FACULTAD DE ESTUDIOS ESTADÍSTICOS

MÁSTER EN MINERÍA DE DATOS E INTELIGENCIA DE NEGOCIOS

Curso 2020/2021

Trabajo de Fin de Máster

TÍTULO: Leveraging Entities Knowledge to Bypass the Cold-Start Recommender Problem on Microsoft News Dataset

***Alumno*: Carlos Ignacio Pérez Sechi**

***Tutor*: Javier Portela García-Miguel**

Junio de 2021

UNIVERSIDAD COMPLUTENSE
MADRID

Content

Abstract 1
Background and state of the art 3
Problem description 3
Intervention description 3
Why did the intervention succeed? 3
Why is this study critical? 4
Materials 5
Data sources 5
Variables 6
Hypothesis 9
Objectives 9
Main objective 9
Secondary objective 9
Methodology 11
Analysis unit 11
Statistic methods 11
Technology and programs 11
Machine Learning Concepts 13
Eligible Algorithms 13
Logistic Regression 13
Decision Trees 14
Neural Networks 15
Discarded Algorithms 17
Knowledge Graphs and Graph Neural Networks 17
Differences Between Graph Neural Networks and Tabular Methods 18
Knowledge Graph Embeddings 19
MKR 21
KGAT 22
RippleNet 23
Implementation 25
Environment Setup 25
Sampling, Exploration and Modification 26
Knowledge Graph Construction 27
RecBole Formatting 27
Tabular data construction 29
Missing Value Assignment and Low Representativity Class Value Imputation 32
Variable Selection 33
Transformation of Categorical Variables with High Cardinality 34
Modeling 35

Tuning Strategy for Graph Neural Network Methods 36
Graph Neural Network Methods Modeling 37
Tuning Strategy for Tabular Methods in SAS 38
Tuning Strategy for Tabular Methods in R 41
Tabular Methods Implementation 42
Tabular Model Execution Results 45
Assessment 46
SAS Base Macro Output Retrieval 46
Python AUC Metrics 47
Results & Discussion 49
AUC Comparison 49
Categorical Variables Transformation for Best Models 51
Complexity Differences 51
Model selected 52
Conclusions 53
Performance: Graph vs. Tabular Methods 53
Complexity: Graph vs. Tabular Methods 54
Graph Algorithms: what impacts their performance? 55
Tabular Methods Software: what impacts their performance? 57
Future Work and Open Research Lines 59
Next Steps 59
Bibliography 61
Annexes 63
Annex 1: Code 63
Annex 1.1: WikiData API entity information fetch script 63
Annex 1.2: MIND behavior.tsv transformation into RecBole format 63
Annex 1.3: MKR training code 64
Annex 1.4: KGAT training code 65
Annex 1.5: RippleNet training code 65
Annex 1.6: Early Stopping calculation macro 66
Annex 1.7: Binary pack in SAS Base 68
Annex 1.8: avNNet and Logistic Regression training functions in R 76
Annex 1.9: Regression Tree training function in R 80
Annex 1.10: Gradient Boosting training function in R 81
Annex 1.11: Random Forest training function in R 83
Annex 1.12: XGBoost training function in R 85
Annex 1.13: Two methods variable selection model training in SAS Base 86
Annex 1.14: Three methods variable selection model training in SAS Base 90
Annex 1.15: Four methods variable selection model training in SAS Base 93
Annex 1.16: Five methods variable selection model training in SAS Base 103
Annex 1.17: Two methods variable selection model training in R 119

Annex 1.18: Three methods variable selection model training in R 129
Annex 1.19: Four methods variable selection model training in R 135
Annex 1.20: Five methods variable selection model training in R 140
Annex 1.21: AutoML Implementation in R 146
Annex 1.22: SAS Base scoring information transformation into AUC CSV files 148
Annex 1.23: AUC Boxplot chart for tabular models 149
Annex 1.24: AUC Boxplot chart for Graph Neural Networks models 150
Annex 1.25: AUC Boxplot chart for the models over 0.93 threshold 151
Annex 2: Results 152
Annex 2.1: Graph Neural Networks results 152
Annex 2.2: Tabular methods results 155

Figures

Figure 1 MIND emblem 1
Figure 2 WikiData item parts (from https://www.wikidata.org/wiki/Wikidata:Glossary) 6
Figure 3 Neural Network Example (Portela García-Miguel, 2019) 15
Figure 4 Full graph (Hamilton, 2020) 18
Figure 5 Overview of how a single node aggregates messages from its local (Hamilton, 2020) 19
Figure 6 Overview of the encoder-decoder approach (Hamilton, 2020) 20
Figure 7 The framework of MKR (H. Wang et al., 2019) 21
Figure 8 Illustration of the proposed KGAT model (X. Wang et al., 2019) 23
Figure 9 MSI GTX GeForce 1050 Ti card 25
Figure 10 MSI GTX GeForce 1050 Ti card mounted in the computer 26
Figure 11 MSI AfterBurner resources monitor 26
Figure 12 Complete graph visualization of user, article, and entity relations. 28
Figure 13 Head, property, and tail example 31
Figure 14 Variable selection used methods 33
Figure 15 Target encoding representation from https://brendanhasz.github.io/2019/03/04/target-encoding 35
Figure 16 Neural Network N6 Levenberg Marquardt TanH Early Stopping Calculation 39
Figure 17 Scoring results for Gradient Boosting model of two methods selection input variables, maximum depth of six and shrinkage 0.1 45
Figure 18 AUC information for ten hidden layer nodes Neural Network implemented with aVNNet with shrinkage of 0.1 46
Figure 19 Boxplot for AUC metrics of SAS Base models trained using fifty variables selected from two selection methods 48
Figure 20 Best models for each algorithm 51
Figure 21 Complexity of Graph Methods vs. Neural Networks vs. Gradient Boosting by the number of users 54
Figure 22 Complexity of Neural Networks vs. Gradient Boosting by the number of users 55
Figure 23 MKR complete AUC Boxplot grid results divided into eight regions 55
Figure 24 KGAT complete AUC Boxplot grid results divided into four training regions 56
Figure 25 RippleNet complete AUC Boxplot grid results divided into three regions 56
Figure 26 Best Model Mean AUC by Input Variable Set and Software 57
Figure 27 KGAT Best Results 153
Figure 28 Left. MKR Best Results 153
Figure 29 Right. RippleNet Best Results 153
Figure 30 MKR full model results 154
Figure 31 KGAT full model results 154
Figure 32 RippleNet full model results 155

Figure 33 Top Left. SAS Base best models for variables selected by two methods. 156
Figure 34 Top Right. SAS Base best models for variables selected by three methods... 156
Figure 35 Bottom Left. SAS Base best models for variables selected by four methods. 156
Figure 36 Bottom Right. SAS Base best models for variables selected by five methods. 156
Figure 37 SAS Base two methods selection variable set model results..... 156
Figure 38 SAS Base three methods selection variable set model results..... 157
Figure 39 SAS Base four methods selection variable set model results 157
Figure 40 SAS Base five methods selection variable set model results..... 158
Figure 41 H2O GBM variance study fed with 50 input variables selected by two methods 158
Figure 42 R best models for variables selected by two methods. 159
Figure 43 R best models for variables selected by three methods. 159
Figure 44 Left. R best models for variables selected by four methods. 159
Figure 45 Right. R best models for variables selected by five methods. 159
Figure 46 R two methods selection variable set model results..... 160
Figure 47 R three methods selection variable set model results 160
Figure 48 R four methods selection variable set model results 161
Figure 49 R five methods selection variable set model results..... 161

Tables

Table 1 Sample of behavior.tsv file..... 5
Table 2 Sample of news.tsv file 5
Table 3 Variable's description for behavior.tsv file..... 6
Table 4 Variable's description for news.tsv file 7
Table 5 Variable's description of a Wikidata triplet..... 7
Table 6 File format for RecBole engine..... 27
Table 7 RecBole files dataset 29
Table 8 Tabular dataset variables description 31
Table 9 Top 5 variables with the most significant number of categorical levels 32
Table 10 Top 5 variables with the most significant number of missing values..... 32
Table 11 Variable selection set based on joined selection methods..... 34
Table 12 Hyper-parameters tuned in each method..... 36
Table 13 Tuning Grid for MKR 37
Table 14 Tuning grid for KGAT 37
Table 15 Tuning grid for RippleNet 37
Table 16 Hyper-parameters for Neural Networks in SAS Base..... 39
Table 17 Each of the figures in this table corresponds to an Early Stopping point for each of the Neural Networks from two to ten nodes, Back Propagation and Levenberg Marquardt optimization functions in the grid. For example the second column shows that the point of Early Stopping is 13 for all the models with Back Propagation optimization function, eight input variables, and two to six nodes 40
Table 18 Transformation from eight nominal variables selected by four selection methods to 82 dummy variables 40
Table 19 Parameters for Decision Trees, Gradient Boosting and Random Forest in SAS Base 41
Table 20 Hyper-parameters for avNNet, Decision Trees, Bagging, Random Forest, Gradient Boosting, and XGBoost in R..... 41
Table 21 H2O AutoML Setup..... 42
Table 22 Top ten models from AutoML tuning..... 45
Table 23 Number of combinations for each algorithm..... 49
Table 24 Best model AUC result for each algorithm 50
Table 25 Categorical Variables versus Transformed Variables for models trained with 50 variables 51
Table 26 Three best models' comparison and best model selected..... 52
Table 27 Ranking of Algorithms' Performance by Mean AUC. In yellow, Graph Methods 53

Table 28 Inverse Ranking of Algorithms' Performance by AUC Variance. In yellow, Graph Methods 54

Code

Code 1 SAS Base Error Output from SVM Method 17
Code 2 Node Q34433 embedding with TransE method..... 20
Code 3 Pseudo-code for entity information retrieval from WikiData 27
Code 4 Entity and article link extraction from news.tsv pseudo-code..... 28
Code 5 Filter relevant properties algorithm 31
Code 6 Test grid example..... 38
Code 7 Training loop for each test case and CPU process..... 38
Code 8 Execution time for a Neural Network with two nodes in the hidden layer, activation function Tanh, optimization algorithm Levenberg Marquardt, and 28 estimators 38
Code 9 Capture of the Logistic Regression score out of the cruzadalogistica method 43
Code 10 AUC calculation function example for Logistic Regression in R 43
Code 11 Gradient Boosting training in SAS Base with fifty input variables selected by two selection methods, a maximum depth of six levels and shrinkage of 0.1 44
Code 12 Neural Network training with avNNet package in R with two nodes in the hidden layer, the Learning Rate of 0.1 and fifty input variables selected by two selection methods 44
Code 13 AUC extraction function in Python for Gradient Boosting results in SAS Base format 47
Code 14 Boxplot chart for SAS models with fifty variables as input selected by two selection methods..... 47
Code 15 AutoML Implementation 147
Code 16 Best 10 AutoML Models Variance Study..... 147

Abstract

Online news has been a hot topic since 2002 when the New York Times published its news RSS feed (Doree, 2007). At this time, users subscribed to the feed using their Netscape Navigator browser and received a daily update of the titles published in the newspaper. This feed was a service with no cost for the user and no monetary income for the journal. Nowadays, user engagement to services is one of the most profitable features for information companies, so they pay more attention to what they show to the users.

To keep the user engaged, news aggregators offer their clients relevant information based on their interests. In the early days, companies asked the user the definitive source of the information he wanted to read to gather the user's preferences. Later, the aggregators ask the user about the kind and features of the information of interest. Still, most current systems do not ask for explicit information from the users but model their behavior from their navigation history.

Recommendations arise on top of user's interest models, matching interests and news features. The intersection of both was first done by exact classification match and become fuzzier every moment till now where we have stickiness probabilities. Users and news featuring have walked a long path from its manual classification to the machine learning classification techniques used here, improving the user's recommendations.

This study dives into the Microsoft News Dataset (Wu et al., 2020), analyzing the users' behavior in it. The main objective was to estimate the click gesture given the news clicked by users previously. This prediction is helpful for any news aggregator portal to show the most relevant news for the user's interest, saving time to the user, reducing the resources consumed by the portal, and, most importantly, improving its engagement score.

NEWS MIND

Figure 1 MIND emblem

Showing the users the most relevant news is a cold-start problem, where portals do not have enough collaborative information about news themselves before they become obsolete. News is a volatile asset, which dramatically depreciates as time passes. There is not enough time for an article to get relevant information from the users' community to profile collaborative filtering (CF) score. Therefore, this technique is excluded from the hypothesis explained hereabove. Instead, featuring engineering and featuring inference were used to predict the importance of a news article for a user.

Background and state of the art

Microsoft News Dataset (Wu et al., 2020), MIND in its English initials, was released in July 2020 for research purposes. There was no other public English news-related dataset of this size until now. Most of the academic studies about user interaction with news consider only non-English dataset as the one of Adressa (Gulla et al., 2017) in Norwegian language or Yahoo! Japan (Okura et al., 2017) in English but only containing 14180 news, ten times fewer news than MIND with 161013 news.

At the annual meeting of the Association for Computational Linguistics in July 2020 (ACL 2020), Wu et al. presented their paper (Wu et al., 2020), describing the motivations, insights, and future work regarding this dataset. One of their main conclusions was the importance of selecting the user interest model related to news recommendations performance. They mostly use interest models based on the relation between words in news and the previous click history of the users through linguistic techniques. The method which performs better is the Neural News Recommendation with Multi-Head Self-Attention, which represents news from its title words and users from their click history.

Problem description

"Is an article interesting for a given user in the Microsoft News ecosystem?" This statement is the main problem to solve when analyzing the Microsoft News Dataset, which is the object of this study. Knowing the user's interest in a specific content is an arduous task given the temporary information of news. As it is not feasible to use the Collaborative Recommendation schema to recommend articles, there is no possibility to use old scores that other users have assigned to news articles. Therefore, the challenge was to use a different approach to score news articles for each reader, regardless of the subjective relevance that other users give to such pieces.

Intervention description

While it is not convenient to use Collaborative Scoring, it was still possible to take advantage of the relationship between articles, content, and previous user interactions. We can interpret it as an unconscious preference scoring behavior, where users give more relevance to certain news while their contents are more relevant in their navigation history.

It was proposed in this study the use of the relations between content, news articles, and readers to figure out the latent users' interest in news articles that they have not read yet. This study has dived into the entities listed in each article and has related these entities with all the articles containing them and the readers' visit history, resulting in a relation graph of readers, articles, and entities to extract each user interest.

Why did the intervention succeed?

User attraction to public entities is a reality since humans live in society. This attractiveness has repeatedly been increasing while communications and media delivery technology evolve. From newspapers to social networks, celebrities have trapped our attention and have developed various feelings among their followers.

The kind of interaction that media distribution systems as radio or tv, social network platforms, or news aggregators, for the sake of this study, offer to its users towards celebrities derived from what is called parasocial relations. It is well distinguished when, for example, a new user comes into an unknown social network for him. As users arrive at the platform, they establish links to their closest related circle and famous people they

know in each community (Yu & Kak, n.d.). This process allows users to get enough information about each of the categories of the user's interest.

This relation is helpful to gather news and an acknowledgment of which products are good, healthy, and even needed by those users following celebrities who are the source of brand and product endorsement (Hwang & Zhang, 2018) in social media. The impact of celebrities' endorsements can be measured financially by reviewing the effect over equities in the stock market after a specific announcement of a celebrity endorsement (Ford, 2018).

Why is this study critical?

The result of this study can be applied to deliver information and news even more relevant than before to users subscribed to the Microsoft News portal, retaining the interest of the user in this platform.

This exercise outperforms the current results of the linguistic models that describe the user interest in MIND. Every slight improvement in the accuracy of the user interest model can derive into several users that stick to the informative channel.

Hence, any improvement in the model that describes the user interest means that the probability of hitting a relevant news article for the user is higher: the number of relevant articles per user delivered to the user. Therefore, the number of relevant articles sent to accurate users increases. Consequently, the number of satisfied users increases.

Customers' satisfaction is directly proportional to the perceived brand value (O'Sullivan & McCallig, 2012). This study is critical because it will lead to a better brand perception.

Materials

Data sources

The central part of the data used in this study came from the Microsoft News Dataset (MIND) released at the Association of Computational Linguistic meeting in July 2020 (Wu et al., 2020). This data contains 1M randomly sampled users with at least five interactions with a 160K news articles population in the Microsoft News portal during the observation period of 6 weeks from October 12th to November 22nd of 2019.

In particular, the MIND dataset is composed of four files: *behavior.tsv*, *news.tsv*, *entity_embedding.vec* and *relation_embedding.vec*. The first two *.tsv* files correspond to user interactions and news properties. The latter files provide the encoding calculation of the entities in news and their relations when using the *TransE* embedding method. However, the *TransE* embedding is not used in this study. Instead, in this research, algorithms will create their own representation of entities included in the news.

The *behavior.tsv* and *news.*tsv files have this aspect:

User Identifier	Date	Previously clicked	Display log
U80234	11/15/2019 12:37:50 PM	N55189 N46039 N51741 …	N28682-0 N48740-0 N31958-1 …
U60458	11/15/2019 7:11:50 AM	N58715 N32109 N51180 …	N20036-0 N23513-1 N32536-0 …
U44190	11/15/2019 9:55:12 AM	N56253 N1150 N55189 …	N36779-0 N62365-0 N58098-0 …
U87380	11/15/2019 3:12:46 PM	N63554 N49153 N28678 …	N6950-0 N60215-0 N6074-0 …

Table 1 Sample of behavior.tsv file

New ID	Category	Subcategory	…	Body Entities
N55528	lifestyle	lifestyleroyals	…	[{"Label": "Prince Philip, Duke of Edinburgh", "Type": "P", "WikidataId": "Q80976", …}]
N18955	health	medical	…	[{"Label": "Drug Enforcement Administration", "Type": "O", "WikidataId": "Q622899", …}]
N61837	news	newsworld	…	[{"Label": "Ukraine", "Type": "G", "WikidataId": "Q212", …}]
N53526	health	voices	…	[{"Label": "National Basketball Association", "Type": "O", "WikidataId": "Q155223", …}]
N38324	health	medical	…	[{"Label": "Skin tag", "Type": "C", "WikidataId": "Q3179593", …}]

Table 2 Sample of news.tsv file

The tables above give us scarce information to work with, apart from the linguistic study, not considered in this work. To extend the data contained in MIND, this study uses the Wikidata database content related to the entities in the news articles.

Reviewing the JSON object shown in the *Body Entities* column of Table 2, we see it contains a label, a type identifier, and a Wikidata identifier. This last identifier has the link point with extended information hosted in the Wikidata database, which reached 71,611,020 entities in February 2020. An entity in Wikidata looks like the schema shown in Figure 2, and as we can see in its *property* value, entities also point to other entities, building this way a graph representation of the entities' relations. This study will take advantage of these relations.

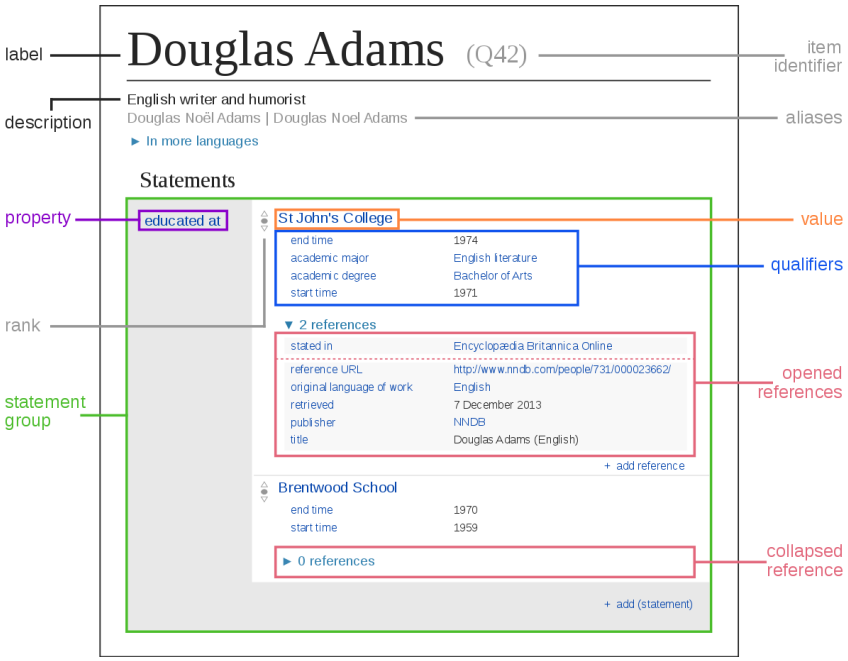


Figure 2 WikiData item parts (from https://www.wikidata.org/wiki/Wikidata:Glossary)

Variables

Columns in *behavior.tsv* file represents user unique identifier, timestamp of the news display event, historical data of the user's previous clicked news articles, the displayed unique article identifiers, the information about clicked or not clicked in each article. This last column, *Display Log*, is a list of tuples, each tuple containing the news identifiers and the boolean indicator telling if the user clicked the article: *identifier-clicked*. The following table describes each of the columns in *behavior.tsv* file.

Column	Information	Type	Cardinality
User Identifier	Unique identifier string of the user which performs the behavior of the observation	Identifier	Very High, over 1000 unique values
Date	Date string of the observation	Date time	Low: six weeks of study
Previously clicked	List of the news identifier strings clicked by the user before the observation	List of identifiers	Very high, over 1000 unique values
Display log	Tuple list of the news identifier strings and click indicator of the articles displayed in the observation	List of tuples of identifier and boolean	Very high, over 1000 unique values

Table 3 Variable's description for behavior.tsv file

The *news.tsv* file describes the features of the news articles used in this study. It has the information about the article identifier, the classification that Microsoft redactors give to each article, title, body, URL address, and Wikidata information.

Column	Information	Type	Cardinality
New ID	Unique identifier of the new described	Identifier	Very high, over 1000 unique values
Category	First categorization provided by redactors of the Microsoft News Portal	Categorical	Low: 15 unique values

Subcategory	Subsequent categorization provided by redactors of the Microsoft News Portal	Categorical	High: 204 unique values
Title	Title string of the article	String	Very high, over 1000 unique values
Body	The content of the article limited to 2600 characters	String	Very high, over 1000 unique values
URL	The URL address of the article	String	Very high, over 1000 unique values
Title Entities	Information about the entities contained in the title of the article described as Wikidata identifiers	JSON structure of strings and identifiers	Very high, over 1000 unique values
Body Entities	Information about the entities contained in the body of the article described as Wikidata identifiers	JSON structure of strings and identifiers	Very high, over 1000 unique values

Table 4 Variable's description for news.tsv file

When dealing with Wikidata information, there are only three variables in a triplet format that this study has handled: head entity, relation type, and destination entity. A triplet is a three variables piece of information, and on this occasion, it contains the information of a relation between two entities. A relation between two entities in the Wikidata database could contain information about how Madrid and Spain are related politically, as an example. Madrid would be the head entity in this relation, Spain would be the tail entity, and the relation type would be *the capital city*. With this kind of information, more variables are derived later on to build a larger dataset, but we will review how they are transformed in the chapter Tabular data construction.

Variable	Information	Type	Cardinality
Head Entity ID	Unique identifier of the source entity of a Wikidata relation triple	Identifier	Very high: 23093 unique values
Relation Type ID	Unique identifier of the relationship type between head and tail entities	Identifier	High: 872 unique values
Tail Entity ID	Unique identifier of the destination entity of a Wikidata relation triple	Identifier	Very high: 147834 unique values

Table 5 Variable's description of a Wikidata triplet

Hypothesis

This study postulates the hypothesis that the strategy of using Graph Neural Networks to model the user interest in news articles enhances the description of more profound relations between user behaviors, news articles, and entities included in those articles. The methods used to model the user's interest for the articles in this database up to this study imply the linguistics of the content and a single user click history. This hypothesis goes a step forward relating entities contained in the news of not only one but several users.

Objectives

Main objective

The objective of this study was to find a model that could recommend relevant news articles to the users of the Microsoft News portal in a better way than nowadays method: this research improves the results of the user interest models for the Microsoft News Dataset presented during the annual meeting in 2020 of the Association of Computational Linguistics.

The most performant method presented in the meeting was the News Recommendation Multi-Head Self-Attention model, which recommends news articles to users in the MIND. The score achieved by the mentioned method is 66.91 for the Area Under the Curve (AUC) and 40.85 for the Discounted Cumulative Gain at 10 (DCG@10). The first score tells us how sensitive and specific the model for different cut-off points is. The second score evaluates the relevance of the list of documents returned by the model regarding the information gained.

The AUC score was the target to beat for the models presented in this study. This score helped compare every model's Goodness-of-fit. However, those scores were not the only criteria when selecting the candidate models. The simplicity of each model and the ability to be self-explained were the other two non-numeric criteria that this study cared about when selecting the candidate model. Even when the score achieved for the AUC was good, a significant difference in the simplicity of the model might have tipped the balance towards a less accurate alternative.

Secondary objective

During this study, the quality of traditional models was compared against Graph Neural Networks. This study has approached the objective in first-degree feature relation and a more profound degree relationship model. Based on the hypothesis, a higher degree approach would have better explained the user's preference for a news article than a superficial first-degree model. The comparison in both models was the secondary objective of this study.

Methodology

Analysis unit

This study analyzed the relation between users' and news articles' features to determine their relevance with users, leaving aside the study of any linguistic feature. The focus was on the link that users may have with the explicitly defined characteristics included in the dataset of the news articles they read.

Statistic methods

This study has applied these techniques to follow the first-degree relation approach: Logistic Regression, Classification Trees, Bagging or Gradient Boosting techniques, and feed-forward multilayer perceptron combining convergence algorithm, number of nodes, and activation function.

On the other hand, to compare the deeper degree relations of the news articles features with the user preference, this study has applied Graph Neural Network (GNN) techniques. This study's specific implementations of GNN are the Knowledge Graph Attention Network, Ripple Net, and Multi-task Feature Learning. The higher-order relation between users and news articles amplifies the effect of those neighbor nodes in the graph relevant to the current node embedding calculation.

Technology and programs

The first step to begin working with the MIND is to format the data into the desired structure. This analysis has used Python 3.7, Microsoft Excel, Unix Bash, SED, AWK, and GREP to perform this transformation. This first structural step resulted in the curated dataset. SAS Base and SAS Enterprise Miner were the software for sampling, exploring, and modifying the curated dataset.

For modeling and assessment, this study has used R language, SAS Base, and Python language with its native library for data mining: pandas, sklearn, scipy, numpy, tensor flow, plotly, caret, nnet, H2O, sas7bdat, and more, in terms of infrastructure.

Machine Learning Concepts

This chapter summarizes the most relevant concepts of different machine learning disciplines as part of the techniques used in this study.

It is necessary to know basic linear algebra concepts to follow the upcoming explanations. To better understand the mathematical basics applied here, please visit the online publication *Deep Learning* (Goodfellow et al., 2016). It holds introductory content about scalars, vectors, matrices, operations, treatment, probability and information theory, explanation of a random variable, probabilistic independence, density, and distributions probabilistic functions.

Also, to extend any of the statistical learning concepts here, please refer to the publication *An Introduction to Statistical Learning* (James et al., 2013). The reader could find a more detailed explanation of algorithms and techniques here overviewed in this publication.

This section describes three main topics of machine learning: first, a review of possible algorithms used to analyze and model the propensity of click given the MIND dataset; afterward, it studies Logistic Regression, Decision Trees, and Neural Networks in their parameterization and modeling; and finally, it jumps into the deep learning techniques, analyzing three algorithms used during this study.

Eligible Algorithms

Several algorithms and machine learning techniques could be applied to the MIND dataset to model the click prediction given a newspaper article. Nonetheless, only a few of them are depicted here because of one principal reason: familiarity with their usage. Most of the algorithms evaluated here are part of the training taught in this job's master's degree subject.

Logistic Regression

Logistic Regression is the first model to study due to its simplicity and explanatory capacity. The Logistic Regression estimates the event probability of an objective variable given the value of its estimators.

When optimizing the Logistic Regression model, changing the default optimization function from the least square error function to the maximum likelihood method is necessary.

$$L(\beta) = \prod_{i/y_i=1} p_{1i} \prod_{j/y_j=0} (1 - p_{1j})$$

$$p_{1i} = P(Y = 1|x_{1i}, x_{2i}, x_{3i}, \dots, x_{mi}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_{1i} + \dots + \beta_m x_{mi})}} \quad (1)$$

Using an iterative process is the only way to optimize this likelihood function. There is no direct mathematical formula that could be derived and optimized. All along the iterative process, the algorithm reviews the β_k to maximize the likelihood formula (1).

Two more fitting optimization techniques help in finding the best model: variable selection and model selection. In the variable selection process, variables are added or dropped from the Logistic Regression model to contribute to the Wald test statistic model. The algorithm adds variables to the model in a stepwise strategy order: first, there are no variables in the model, and then every step decides if including or removing a new or existent variable in terms of the Wald test. It is also possible to use the forward or backward strategies, but the default stepwise is the one we used for this case.

Among all the models generated in each variable selection step process, this method selects the best of those models using the overall Goodness-of-fit test. So, the algorithm eliminates the possible information bias introduced by the Wald test. We must consider that the Wald test is not a Goodness-of-fit test but an information contribution test, meaning that in each step, the variable that contributes most to the information provided about the objective variable is the one included in the model. Outperforming the Wald test does not imply that including this variable makes the model fit better the objective variable.

The model calculation, variable selection, and model selection processes are repeated several times, one for each partition of the train and validation sets and one for each seed used to bootstrap the partitions. This repetitive process leads to $Repetitions = Number\ of\ partitions\ X\ Number\ of\ seed$ different best logistic models in each repetition that are compared using the test partition fit test wanted: maybe AUC, or Miss Classification Rate, or Sensitivity.

Decision Trees

In the second position of the preferences list, the Decision Trees are the base model for a range of techniques and algorithms with high impact in recent data science competitions: Gradient Boosting, Bagging, Random Forest, or XG Boost.

Decision Trees output comprehensible information about the model of the objective variables, being easy to read and describe. This technique predicts by information segmentation, based on sequential rules optimized in each training iteration following a hierarchy of decisions. This hierarchy starts in the root node and keeps dividing the observations of the datasets by choosing the correct variable and the suitable threshold for each variable in each decision in a binary way. When choosing the variable for each decision node, it is possible to use three different measures: Chi-Square test, Gini's index, or Entropy.

Some advantages of Decision Trees are their clarity, simplicity, generalization capacity, missing value, and against outlier's robustness. Nevertheless, they are very vague when predicting linear relations, and each training is highly demanding in computer power compared with logistic regression. These problems are solved with advanced ensemble techniques, achieving better accuracy.

Gradient Boosting, and its implementation XG Boost, are the most recent decision tree ensemble techniques used here. As its names say, this method consists of finding as many Decision Trees as the maximum number allowed or the limitation in improvement achieved. Each tree is built up from the residual value of the previous tree prediction through a transformation function. Boosting, in this case, consists of building the successive regression trees. The gradient part comes from the usage residual or difference between the observed target value with the predicted one.

Random Forest and Bagging are very similar techniques. Both create a chosen number of decision trees from a different sample subset in each tree to average their predictions as an ensemble output. The difference between Bagging and Random Forest is that the latter also chooses a subset of variables to be evaluated in each tree node, not all the variables set available in the original dataset.

The essential parameters in these Decision Tree techniques are standard: the depth of the trees chosen, the number of leaves or final nodes, the minimum number of observations in each node o for each category, and in the case of Gradient Boosting, the learning rate.

Neural Networks

This first step, Neural Networks, is very close to testing deep learning methods. The connection neurons have in brains inspires this technique. Each neuron would compute an output when receiving stimulation from its neighbor neurons.

This model is described mathematically with standard components common to any neural network. First, information coming from initial nodes must be numerically encoded, meaning that there is no place for categorial information through the network, and so, it must be transformed into magnitudes.

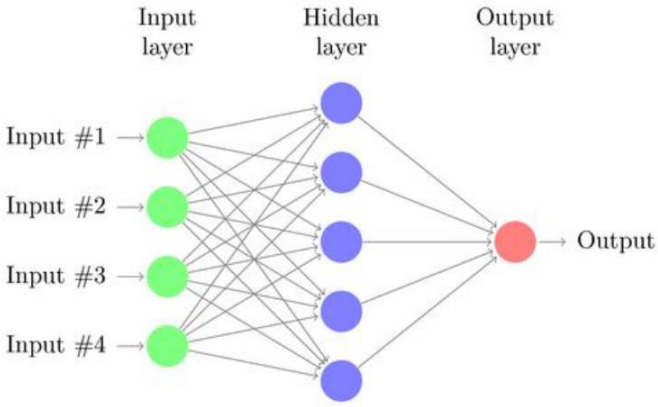


Figure 3 Neural Network Example (Portela García-Miguel, 2019)

From Figure 3, three types of nodes can be distinguished: green, blue, and red. The green nodes correspond to the entry layer and contain the estimator variables' information in our model. The blue nodes are the so-called hidden layer nodes, and their number consists of one of the most powerfull tuning parameters for Neural Networks. Last, the red node is the output node, and its importance comes from the aggregation of all the previous calculations.

Three functions are designed for each node to improve the heterogeneity of the network: the transfer function, the aggregation function, and the activation or output function. The transfer function modifies the value coming from the previous node. The aggregation function joins the information coming from all the connected nodes, and finally, the activation function permits or denies the output of information from the node being studied.

Additionally, latent values in the network are modeled by the bias of each node, adding an optimized constant value to each node aggregation function. With all these ingredients, the mathematical function that describes the output of the network is as follows:

$$output = \mathcal{F}_{out}\left[\beta_{out} \sum_{i \,\epsilon\, nodes} w_{i,out} \cdot \mathcal{F}_i\left(\beta_i \sum_{j \,\epsilon\, inputs} w_{j,i} \cdot input_j\right)\right] \quad (2)$$

Where $\mathcal{F}_{out}$ comes from the output activation function, β_{out} is the output bias, $w_{i,out}$ weights are assigned to the connection between the hidden layer nodes and the output node. For each hidden node, there is a bias parameter represented by β_i and for each input node to each hidden layer node, there is a weight parameter depicted as $w_{j,i}$. The output of the hidden layer is modulated by every activation function of every hidden node by $\mathcal{F}_i$.

All the neural network parameters mentioned in the above paragraph are set to improve the model's output. While some of them are tuned automatically, some others must be reviewed using iterative techniques to set different values for further fit test comparison. Activation functions may also vary among all sorts of continuous and monotonic rising in the dynamic area and bounded out of it: from rectified linear unit function to hyperbolic tangent through all sigmoid functions. This study only uses the hyperbolic tangent function, but any other could be implemented. We selected this function because is the most used in the literature, and using more would multiply the competitional time.

Two techniques are used in this study to optimize bias parameters and weights: Back Propagation and the Levenberg Marquardt method. The former is a technique of adjusting weights and biases to reduce the error magnitude by walking the path back of each node and applying the correction in the direction towards the correct output of the neural network in each iteration. The latter technique takes the same mechanic of Back Propagation, but instead of Gradient Descent, it uses a mix between Gradient Descent and Newton's method, choosing one or another by setting a value from zero to infinity in a parameter called lambda.

Nevertheless, the optimization may have a high cost when dealing with a high amount of data and lots of training iterations. This cost translates into overfitting or model bias when the parameters are so adapted to the training samples that new coming data that was not used to train the model would lead to too inaccurate predictions.

The Early Stopping technique can be used to mitigate over-fitted neural network parameters. As its name says, the Early Stopping method will stop optimizing the network parameters in an early stage given a fulfilled condition. In most cases, the condition to stop the training is that the validation sample set fit starts to deteriorate compared to the training set fit. Usually, the Early Stopping method outputs a particular maximum number of iterations to be performed in a determined network setup to optimize its parameters. The maximum number of iterations is computed in a previous step for a network given its hyperparameters: number of nodes, type of activation function, and optimization technique. Afterward, the neural network is trained using the same hyperparameters and an additional one: the early stopping point.

SAS Base *neural* procedure needs the early stopping parameter provided manually. Therefore, the early stopping point should be calculated in the previous step. Later on in this document, such an Early Stopping study is done for the SAS Base case in the Implementation chapter.

In the case of R Caret *avNNet* package, to avoid model overfitting, the algorithm ensemble several Neural Networks by aleatory parameters exploration, being the output the average of them all. We may tune up the iteration number that *avNNet* repeats this process with the *maxIt* parameters.

And not only the weights, biases, activation functions, optimization methods, Early Stopping point, and probably the most crucial setup, the number of nodes of the hidden layer is parametrized. The importance of this parameter is such that if there are not enough observations in the training set, it may not be possible to perform a coherent model when choosing an inappropriate number of nodes in the hidden layer. Formula (3) describes the relationship between the number of nodes and the training sample size to achieve a minimum coherence in the model predictions. It says that there should be at least twenty observations for each optimizable parameter. The number of parameters being tuned is the denominator of the ratio. Meanwhile, the numerator is the number of observations of the training sample.

$$\frac{observations}{h(k+1)+h+1} > 20 \quad (3)$$

$$h = number\ of\ nodes\ in\ the\ hidden\ layer$$
$$k = number\ of\ input\ nodes$$

Discarded Algorithms

Two of the most significant classification algorithms were discarded from the study because of their incompatibility with the set of predictive variables used: K Nearest Neighbors and Support Vector Machine.

The K Nearest Neighbors algorithm needs at least one continuous dimension to calculate the distance between observations in the sample. In this case, the dataset does not contain any continuous variable, being all of them categorical ones, so it was discarded as an algorithm.

In its duty, Support Vector Machine struggles when leading with a high number of predictive variables that are categorical. The problems come when building the kernel matrix for categorical variables. It reaches the maximum size of the matrix even with the previous selection of eight final input variables.

```
NOTE: Method LSSVM is very unusual and not very efficient for 55488 observations and
82 variables.
WARNING: For large problems like this, the iterative CG method is preferred.
WARNING: Problem too large for storing the kernel matrix, you must use the iterative
CG method.
ERROR: Estimation cannot be performed.
```

Code 1 SAS Base Error Output from SVM Method

Both methods should be removed from the study and could not be compared with the previous algorithms exposed in this chapter.

Knowledge Graphs and Graph Neural Networks

Beyond Neural Networks, the next step in complexity is deep learning networks. Specifically, Graph Neural Networks are chosen from various techniques available for this study.

Graph Neural Networks tries to model the topology, relations, and properties of a linked network of objects where each link or object might be different. It can describe real-world relation topographies, design a statistical problem, or sequence events for a time-relevant scenario.

The advantage of graph model representation is its simplicity when dealing with complex data structures. With graph models, it is possible to link objects of different types, and it is also possible to relate objects with different kinds of relations. Moreover, we could add the orientation of those relations and even have more than one relation coming or going from an object.

Typically, Knowledge Graph models are used to predict complex categorization or relations. One example of usage may be predicting an unseen relation between two nodes in the graph. Knowledge Graph models are also used to classify a node in the graph given its surrounding relations. They may also help classify graphs based on their structure and node composition, such as gregarious classification.

The output of a Knowledge Graph method tends to be a probability of occurrence in a node-to-node relation prediction or the probability of belonging to a category when classifying nodes and graphs.

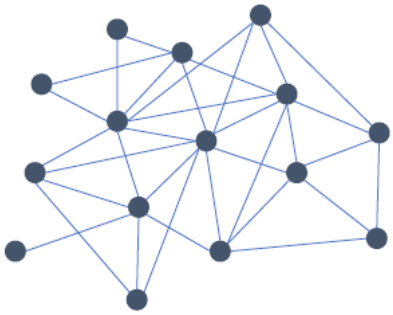


Figure 4 Full graph (Hamilton, 2020)

Moreover, we can easily represent massive information structures with all their complexity in graph representation thanks to the heterogeneity in relations orientation, types, object cardinality, and object types.

Graph models and Neural Networks also look alike. Both have connected nodes with arbitrary order and size. However, Neural Networks have some properties that general graphs may not accomplish. Neural Networks are always:

- Directed graphs
- Connected graphs
- Graphs with order over or equal to three
- Graphs with size greater or equal to two
- Are Triangle-Free graphs
- Are k-partite graphs with k=2 when setting the odd layers nodes in one vertex group and the even layer nodes in the opposite.
- Edges in Neural Networks have the transfer and activation function properties
- Vertex in Neural Networks have the aggregation function and the bias properties

Additionally, Neural Networks use the information from their input nodes following a specific routine through the hidden layer nodes to the output layer nodes. Their implementation is tightened to the waterfall schematic from input to output.

On the other hand, Graph Neural Networks are less restrictive in the scheme of linking the information contained in the graph, the representation of this information, and the algorithm that infers knowledge from it. Graph Neural Networks is an open approach to knowledge inference that allows researchers to develop new algorithms and theories in the Data Mining discipline.

Differences Between Graph Neural Networks and Tabular Methods

A significant difference between Graph Neural Networks and Neural Networks, Decision Trees, or Logistic Regression resides in the fact that the former takes into consideration the surrounding information of a concrete observation when predicting a target value.

It comes from how a node and a relation are represented in this type of Machine Learning algorithm. When representing nodes and relations, the graph's structure is captured, building the dimensions on top of the information coming from surrounding objects in several levels of proximity.

When computing a prediction, graph algorithms' input will contain information of one individual object and its environment. There is a strong correlation between observations in the training dataset, given that observations were computed using the information of their surroundings. This kind of behavior is usually avoided in tabular methods as Logistic Regression and Decision Trees.

Some well-known techniques represent graphs, vertex, and edges, following this philosophy of structure catch, allowing us to bootstrap the definition of models in Graph Neural Networks. They consist of embedding the representation of each element in a newly generated Euclidean vector space. This technique is reviewed hereunder.

Knowledge Graph Embeddings

Knowledge Graphs represent heterogeneous data with complex and numerous relations and high number entities on top of linked, directed, or undirected graphs. The graph's edges represent the relations between entities, and the vertices or nodes of the graph represent the entities included in the knowledge ecosystem.

Literature typically uses the approximation of nodes, relations, and full graph embeddings to represent the information of each set. Embeddings are a low-dimensional representation of the set transformed. We shorten the number of dimensions with embeddings, reducing the amount of data to handle, but compromising the information contained. There is barely little information about the relations or the graph's structure in node embeddings. For the relations embeddings, there is no information of nodes or graph structure in them, while graph embeddings do not have information about nodes or relation types but for the structure itself.

Neural Networks help calculate the embeddings while converging to the minimum value of a loss function when finding the values of each of the embedding vector dimensions in a representation. The loss function usually measures how far the decoding of the embedding was from reality, what is called the Encoder-Decoder perspective.

The pseudo-code followed to build a node embedding representation is:

algorithm embedding **is**
    input: Graph **as** G,
        Node **as** u
    $embeddings_{t=0}$ = hot vector(u)
    $neighbor_{t=0}$ = hot vector(v) ∀ v ∈ N(u)
    for K iterations **do**
        $aggregated_t$ = Aggregation Function($neighbor_{t-1}$)
        $embeddings_t$ = Update Function($embeddings_{t-1}$, $aggregated_t$)

    return $embeddings_K$

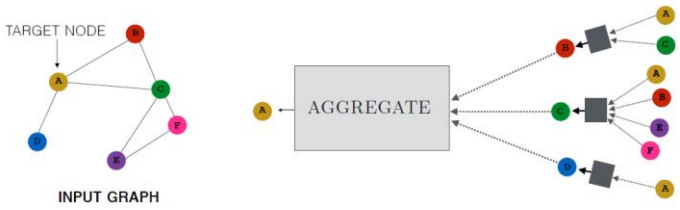


Figure 5 Overview of how a single node aggregates messages from its local (Hamilton, 2020)

The Aggregation and Update functions are continuously differentiable functions. The hot vector function represents a node as an array with as many dimensions set to zero as

nodes are in the graph, while the position of the argument is set to one. Moreover, $N(u)$ is the subset of neighbors of u.

Both Aggregate and Update functions optimize parameters in each of the K iterations by minimizing the selected loss function. The loss function used in the algorithm is under the choice of the researcher. The following uses negative sampling to calculate the Cross-Entropy loss to provide an example of a commonly used function.

$$\mathcal{L} = \sum_{(u,\tau,v)\epsilon\mathcal{E}} -\log\left(\sigma\left(DEC(z_u,\tau,u_v)\right)\right) - \gamma \mathrm{E}_{v_n\sim P_{n,u}(v)}[\log\left(\sigma(-DEC(z_u,\tau,u_v))\right)] \quad (4)$$

Where σ is a sigmoid function, DEC is the decoding function of the system which translates back the information contained in the intermediate embeddings, $\mathrm{E}_{v_n\sim P_{n,u}(v)}$ is the expectation of the negative sampling distribution (Hamilton, 2020), and γ is a parametrizable value greater than zero. The decoding function can be any of the operations supported by the Euclidean space where, $\forall\, u\,\epsilon\,\mathcal{E}, z_u$ lives. Algorithms as *TransE* uses the norm of the aggregation of head, tail, and relation representation. Some use the dot product function as a decoder, and others use the product of head embeddings by a trainable relation parameters matrix by the tail embeddings. It is up to the method to choose all Aggregation, Update, decoding, and loss function.

The following picture shows in an illustrative manner how the Encoder-Decoder scheme works. It helps to assess the Goodness-of-fit of the representation embeddings while optimizing the parameters included in the Aggregation and Update functions.

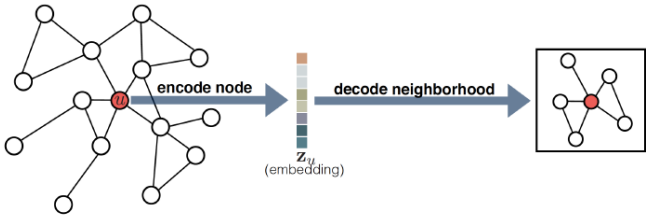


Figure 6 Overview of the encoder-decoder approach (Hamilton, 2020)

Out of this process, a node embedding consists of an array of dimensions. An example is Code 2, where the entity Q34433 is represented in the embedded space by 100 dimension values. It was computed using the *TransE* method (Wu et al., 2020).

```
Q34433 0.017808 -0.073256 0.102521 -0.059926 -0.060665 0.027027 -0.091728 -0.003057 -
0.170798 0.111819 0.006821 -0.049873 -0.050532 -0.003127 -0.074472 -0.115891 -0.067093
-0.095272 0.019178 -0.083725 -0.060890 0.017744 0.049417 -0.026014 -0.048549 0.017528
0.044163 0.022111 -0.081519 0.046278 -0.183939 -0.063143 -0.014518 -0.080644 -0.099994
0.085905 -0.083003 -0.092844 -0.216481 0.125441 0.179819 0.036735 -0.085375 0.021276 -
0.154971 0.039009 0.016059 0.067725 -0.148213 0.158773 -0.028527 0.125790 0.006361
0.067541 0.077552 0.060792 -0.044511 -0.005862 -0.068080 -0.063204 -0.094127 0.115441
-0.016472 0.106616 0.047839 -0.151805 -0.111083 -0.142330 -0.120680 -0.050393 -0.073787
0.017424 -0.081620 0.062599 -0.022102 -0.102688 -0.128149 -0.075895 0.095134 0.000984
0.010143 -0.068552 -0.026573 0.019735 -0.000981 -0.126635 0.008300 0.170557 0.002250 -
0.157175 -0.077962 0.013433 0.045894 -0.071253 0.086445 -0.120466 0.059235 -0.071865
0.058854 0.024765
```

Code 2 Node Q34433 embedding with TransE method

Once explained how to obtain the representation of the information in Graph Networks, in the following three chapters, we will see the three methods selected to learn and predict the link between users, news articles, and entities contained in those articles.

MKR

The first method studied is the Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation (H. Wang et al., 2019). It consists of a framework that takes advantage of both Knowledge Graph Embedding (KGE) and the Recommendation Systems (RS) by linking them together in the *cross&compress* unit.

This algorithm will input the user representation, like the one in Code 2, the target item, and it will suggest the probability of a relationship between them, given the already observed relations in the graph.

To predict the existence of a relation between a user with embeddings vector u_L and an item with embeddings vector v_L MKR uses in its publication the inner product of both embeddings, threshold by the sigmoid function (H. Wang et al., 2019).

$$\hat{y} = \sigma(\langle u_L, v_l \rangle) \tag{5}$$

MKR claims to be a method that made a step forward by finding the benefits of regularizing the learning process of either side of the KGE-RS by the other. It means that KGE influences RS and vice-versa during the optimization of the inner parameters of both. When the loss function of the embeddings calculations corrects the value of an item representations, this representation is updated in the recommendation training. And vice-versa.

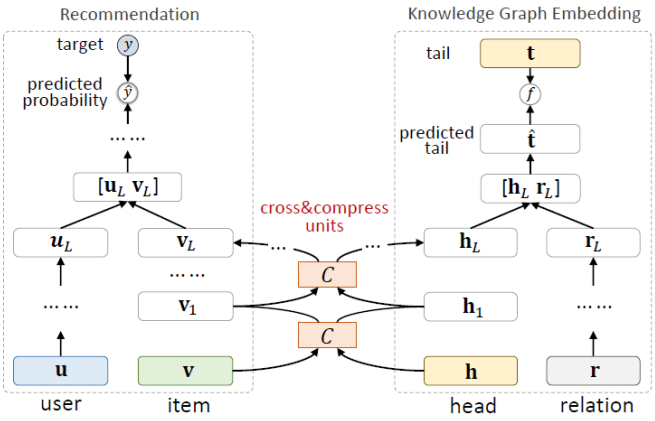


Figure 7 The framework of MKR (H. Wang et al., 2019)

H. Wang et al. called the connection mechanism *cross&compress*. The principle of this connection uses a combination of the item and related entities, expanding first and compressing the result afterward, multiplying by a weights vector that could be optimized. This framework uses Multi-Layer Perceptron to improve the explanatory capacity of each side of the system, and the number of layers used is a hyper-parameter composed of two values: the number of high and low layers.

$$C_l = v_l e_l{}^T$$

$$\begin{aligned} v_{l+1} &= C_l w_l{}^{VV} + C_l{}^T w_l{}^{EV} + b_l{}^V \\ e_{l+1} &= C_l w_l{}^{VE} + C_l{}^T w_l{}^{EE} + b_l{}^V \end{aligned} \tag{6}$$

The above equation describes the *cross&compress* mechanism, where v_l is the representation of item v at layer l, same for e_l and the representation for the entities related to v, and w_l¨ represent the optimizable weights for compression. Only the lower layers apply the *cross&compress* connection to restrict the influence of uncorrelated tail items on the recommendation prediction.

When assessing the Goodness-of-fit, MKR uses the following loss function, where the last part is the regularization one, the first part is the Cross-Entropy loss for the recommendation predictions, and the middle term describes the relation between existing Knowledge Graph triplets and non-existing ones, leveraging the former, penalizing the latter.

$$\mathcal{L} = \sum_{u \epsilon U, v \epsilon V} LogLoss(\hat{y}_{uv}, y_{uv}) - \lambda_1 \left(\sum_{(h,r,t) \epsilon G} score(h,r,t) - \sum_{(h,r,t) \epsilon G} score(h',r,t') \right) - \lambda_2 \|W\|_2^2 \quad (7)$$

As data mining specialists, we could tune this model by setting the following hyperparameters:

- regularization weight, *reg_weight*
- number of lower layers, *low_layers_num*
- number of higher layers where MKR does not perform *cross&compress*, *high_layers_num*
- drop out probability when building the Knowledge Graph Embeddings, *dropout_prob*
- number of loops performed for the recommendation training before continuing with the Knowledge Graph training, *kge_interval*
- the number of dimensions for the embedding of both KGE and RS, *embedding_size* and *kg_embedding_size*.

KGAT

Like MKR, KGAT uses Knowledge Graphs and Knowledge Graph Embeddings to exploit the relation between users, items, and entities for interest prediction. It is a method that overcomes state-of-the-art methods in plugging the Knowledge Graph structure directly into the recommendation learning network. The direct connection of the KG differs from the current regularization-based methods that make this connection implicitly. KGAT also advantages path-based methods by capturing the structure more straightforwardly, eliminating the meta path computational overload.

KGAT's output would be predicting a relation between a user and an item in the way equation (8) shows. It computes the product of two vectors, the user representation vector, e_u^*, and the item representation vector, e_i^*. Both vectors are defined in the modeling process.

$$\hat{y}(u,i) = e_u^{*T} e_i^* \quad (8)$$

During the modeling process, KGAT implements three different tasks: Collaborative Knowledge Graph embedding, Attentive Embedding Propagation, and Prediction. Figure 8 depicts this procedure. During the first tasks, KGAT calculates the entire graph embeddings for each node using the *TransR* method. After that, it propagates the relevance of each node neighbor along with the graph, weighting higher to those nodes that are closer to the interest node while the embedding calculation.

$$e_h{}^l = f\left(e_h{}^{l-1}, e_{N_h}{}^{l-1}\right)$$

$$e_{N_h}{}^{l-1} = \sum_{(h,r,t) \in N_h} \pi(h,r,t)^{l-1} e_t{}^{l-1} \quad (9)$$

$$\pi(h,r,t)^{l-1} = \frac{\exp\left((W_r e_t{}^{l-1})^T \tanh\left(W_r e_h{}^{l-1} + e_r{}^{l-1}\right)\right)}{\sum_{(h,r',t') \in N_h} \exp\left((W_{r'} e_{t'}{}^{l-1})^T \tanh\left(W_{r'} e_h{}^{l-1} + e_{r'}{}^{l-1}\right)\right)}$$

The equations in (9) resume the attention mechanism, where $e_h{}^l$ represents the embeddings after l layers of a head item in a triplet *(h, r, t)* of the CKG. N_h are the neighbor triplets of $e_h{}^l$, W_r and $W_{r'}$ are trainable projection matrices to represent nodes in the relationship space.

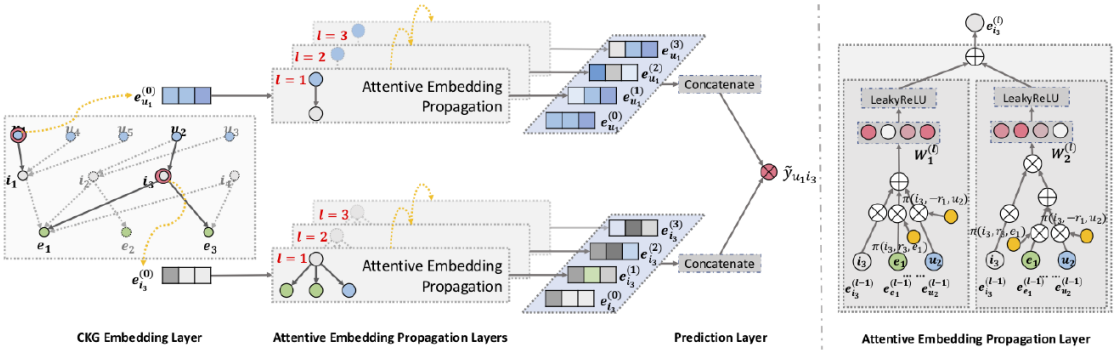


Figure 8 Illustration of the proposed KGAT model (X. Wang et al., 2019)

Finally, KGAT optimizes the weights in the embeddings, attention embeddings, and prediction layers, minimizing the loss function here shown:

$$\mathcal{L}_{KGAT} = \sum_{(h,r,t,t')\in\mathcal{T}} -\ln\,\sigma(\|W_r e_h + e_r - W_r e_{t'}\| - \|W_r e_h + e_r - W_r e_t\|) + \sum_{(u,i,j)\in O} -\ln\sigma(\hat{y}(u,i) - \hat{y}(u,j)) + \lambda\|\Theta\|_2^2 \tag{10}$$

For the middle term $O = \{(u,i,j)|(u,i) \in \wp^+, (u,j) \in \wp^-\}$ denotes the training set with positive, existing, and negative, randomly sampled, triplets. In the first term $\mathcal{T} = \{(h,r,t,t')|(h,r,t) \in \wp^+, (h,r,t') \in \wp^-\}$ and at the end of the equation λ and $\Theta = \{E, W_r\, \forall r \in \wp, W_1{}^l, W_2{}^l, \forall l \in \{1,\dots,L\}$ are the L^2 regularization model parameters.

RippleNet

Resembling raindrops ripples propagation in the water, RippleNet extends the user interest in items all over the Knowledge Graph. This method proposes to solve the problem of handcrafted paths in Knowledge Graph path-based methods and the focus problem of actual Knowledge Graph methods in predicting user-item relation different from inter-item link prediction.

RippleNets prediction mechanism is quite the same as MKR or KGAT; it is based on the inner product of the representations of both user and item nodes. The following paragraphs describe how are those user and item nodes representations computed.

$$\hat{y}_{uv} = \sigma(u^T v)$$
$$\sigma(x) = \frac{1}{1 + exp\,(-x)} \tag{11}$$

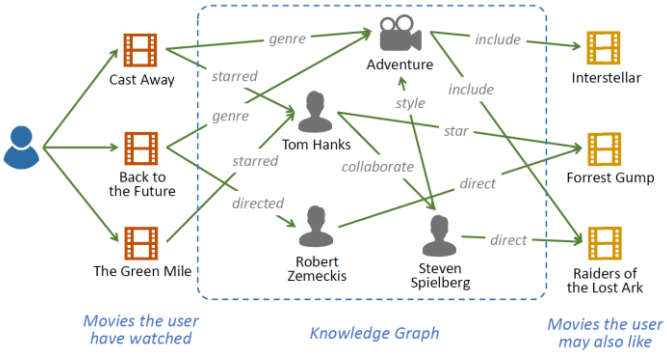
To reflect the user's historical data, RippleNet designs the user's previous interactions by seeding the user embedding representation with the information from the items visited by the user. It is achieved by weighting and joining K chained items' information of the Knowledge Graph from the user interaction with an item first in the chain as a *seed*.

$$u = \sum_{j=\{1..H\}} \sum_{(h_i,r_i,t_i)\in S_u^j} \frac{\exp\left(t_i{}^T R_i h_i\right)}{\sum_{(h,r,t)\in S_u^j} \exp\left(t^T R h\right)} t_i$$

$$S_u^j = \{(h,r,t)|(h,r,t) \in G \text{ and } h \in \mathcal{E}_u^{j-1}\} \tag{12}$$

$$\mathcal{E}_u^j = \{t|(h,r,t) \in G \text{ and } h \in \mathcal{E}_u^{j-1}\};\ \mathcal{E}_u^0 = \{v|y_{uv} = 1\}$$

The above equation, which computes the embeddings for the user u, R represents the relation embeddings for r.



(H. Wang et al., 2018)

Equation (12) is helpful to build the overall loss function (13) for the RippleNet framework. The first term of this loss function calculates the *Cross-Entropy* of the predictions of item match with users. The second term measures the discrepancy between the original adjacency matrix and the generated relations and item embeddings. Last, the regularization term prevents over-fitting.

$$\mathcal{L} = \sum_{(u,v)\epsilon Y} -\left(y_{uv} \log \sigma\left(u^t v\right) + (1 - y_{uv}) log\left(1 - \sigma(u^t v)\right)\right) + \frac{\lambda_2}{2} \sum_{r\in R} \|I_r - E^T R E\|_2^2 + \frac{\lambda_1}{2}\left(\|V\|_2^2 + \|E\|_2^2 + \sum_{r\in R} \|R\|_2^2\right) \tag{13}$$

Implementation

Because of the objective of comparing both graph-based and tabular algorithms, the implementation of models was in R, SAS Base, and Python. Graph-based models were trained in Python and tabular models in R and SAS Base. Tabular models are those models that are trained from tabular and unrelated samples. Such models are Logistic Regression, tree-based models, and Neural Networks.

In particular, to train the graph-based models, the RecBole library was an excellent help, having already implemented the MKR, KGAT, and RippleNet algorithm. It is a Python library with no other programming or statistics language implementation. Hence Python is also used as a modeling language.

Since only one computer was available to develop this entire study, some of the training and tuning procedures have been shortened, excluding, for example, the SAS Enterprise Miner as a modeling engine or some further research in the ensemble area. The future work chapter describes how the work carried out can be improved. Nevertheless we will see the very acceptable results with this limitations.

Environment Setup

As mentioned earlier, to develop this study, only a Windows 10, AMD FX 8120 Eight-Core Processor and 12 GBytes RAM capacity computer was available. It was necessary to keep the computer running for more than 15 weeks with no rest regarding this computing power to achieve the objectives of this study.

Before getting into the data mining process, this computer had to be upgraded in software and hardware capacities to be prepared. Talking about software, this computer received the installation of SAS enterprise Miner, which comes with SAS Base, Rstudio, which requires R software, Python 3.6.5, the only compatible with RecBole, finishing with Microsoft Office Student Suite to have Microsoft Excel available.

In terms of hardware, the graphic card MSI GTX GeForce 1050 Ti was installed to gain GPU capacity compatible with RecBole. The MSI card required specific driver software installed: Nvidia Monitor, GeForce Experience, and MSI AfterBurner.

Figure 9 MSI GTX GeForce 1050 Ti card

Figure 10 MSI GTX GeForce 1050 Ti card mounted in the computer

Figure 11 MSI AfterBurner resources monitor

Once the installation of the listed needed component was completed, the correct integration of them could be checked by running the *torch* Python library check:

```
import torch

torch.cuda.is_available()
>>> True
```

Sampling, Exploration and Modification

The Microsoft News Dataset is not complete to be used in Knowledge Graph studies. It comes with the nodes and relations embeddings previously calculated, but this is not enough information to be plugged into the MKR, KGAT, neither RippleNet methods. Such methods require complete information about nodes and relations, and not only their embeddings representation.

MIND contains information about user click history, display history, display time, article identifier, article title, category, subcategory, article-related entities, and node and relations embeddings of article-related entities from WikiData Knowledge Graph. This work uses the WikiData Rest API to extend the information contained in MIND, querying the properties of those entities included.

Afterward, the retrieved information is arranged in the form RecBole could read and use in its model implementations.

Independently from the graph algorithm, the format used in the tabular algorithm is way different. The data enclosed in graphs must be flattened for its correct usage by algorithms like Logistic Regression. However, it is not feasible to use it all. The huge cardinality and number of variables compounding all the graph information made it useless until it was reduced into a handier dataset.

The following episodes describe the data retrieval and transformation to build two datasets: graph and tabular datasets.

Knowledge Graph Construction

In order to construct the Knowledge Graph Dataset linked to the user interactions part of MIND, this study has used the WikiData Rest API. It is an HTTPS interface that helps query information from entities to properties through statements.

This inquiry only needs to fetch the information related to entities. Entities are a type of information piece compound of several parts: label, description, statement groups, properties, and references. Figure 2 describes an example of an entry in the WikiData database. For this purpose, the endpoint *http://www.wikidata.org/entity/* is used in the script that fetches entity information for every distinguished noun in the news articles in MIND. The pseudo-code of the fetching script is described below, and the complete code is shown in Annex 1.1: WikiData API entity information fetch script.

```
algorithm build graph is
        input: Entities as entities
        knowledge graph file = open file('some location')
        endpoint = 'http://www.wikidata.org/entity/'
        for e in entities do
                item = fetch(endpoint + e)
                for property in item do
                        write e, type(property), property to knowledge graph file
```

Code 3 Pseudo-code for entity information retrieval from WikiData

Up to this point, we have the information of all the properties for all the entities in MIND articles. Now it is necessary to link the articles with their entities and users with the articles they have visited. This labor is described in the following episode.

RecBole Formatting

The RecBole library implements up to seventy-three (73) recommendation models, eight based on Knowledge Graphs. This library allows data miners to explore those models from a single starting point, a single input dataset shared with all implementations. Therefore, the database format is crucial for the correct execution of any model we may want to train.

Suffix	Content	Format
.inter	User-News interaction	*user_id*, *item_id*, *timestamp*
.item	News feature	*item_id*, *category*, *subcategory*
.kg	Triplets in a knowledge graph	*head_entity*, *tail_entity*, *relation*
.link	News-entity linkage data	*entity*, *item_id*

Table 6 File format for RecBole engine

As well as model exploration, RecBole allows users to customize how trainer functions work when dealing with models, how data is loaded into the trainer, and how to sample the data

in the training, evaluation, and test phases. Another relevant feature is the ability of RecBole to program tuning hyper-parameters by configuration, managing the epochs and results for each hyper-parameter combination. Despite this helpful feature, during this work, it is not used. Instead, the training procedure was built manually to collect the results differently.

The input data format must follow the specifications above to use RecBole. This format is divided into four parts: user interactions with news, news description, entities knowledge graphs triplets, and news and entities linkage. Each of the four parts is present in one file correspondingly: *.inter, .item, .kg, .link*. This study uses the code in Annex 1.2 below to get from MIND users' behavior information the format of the RecBole interactions achieving the first needed file.

To get the second file, *.item*, and the fourth file, *.link*, similar code as for the interactions file has been used. The difference resides in the information manipulated. To show more in-depth how to transform the *news.tsv* file of MIND to the linkage file, we have the pseudo-code below:

```
algorithm from news to linkage is
        input: Articles as articles
        news.link = open file('some location.link')
        for article in articles do
                wikidata ids = extract("WikidataId" from "Entities" in article)
                for id in wikidata ids do
                        write article, id to news.link
```

Code 4 Entity and article link extraction from news.tsv pseudo-code

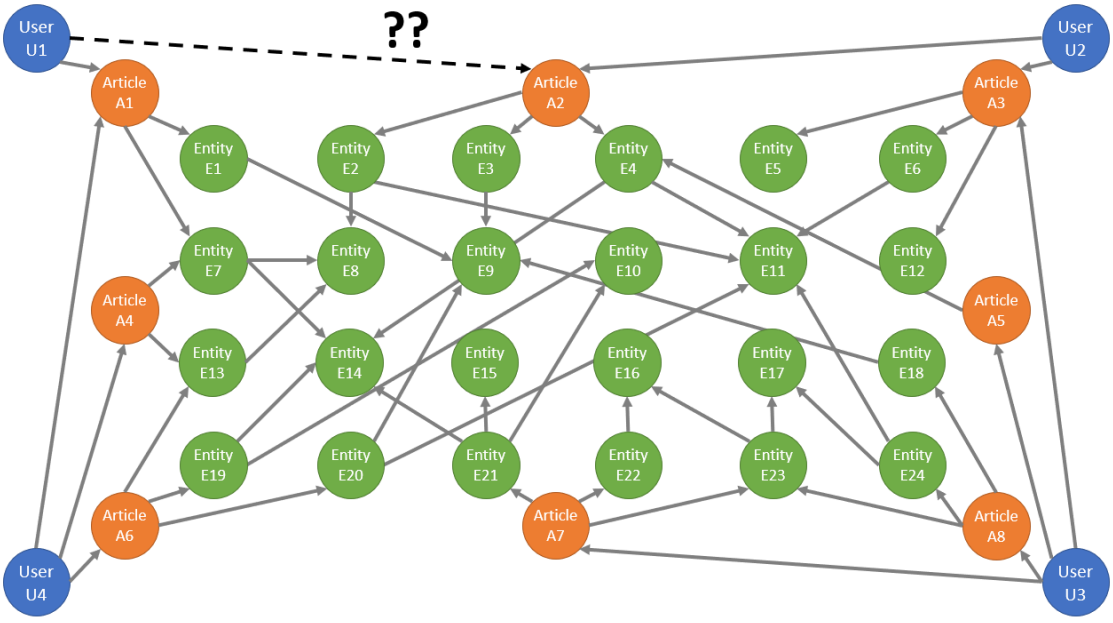


Figure 12 Complete graph visualization of user, article, and entity relations.

Once we have the four files with the needed information for users, interactions, news, and entities, we can visualize an example of how this information is linked into a graph visualization like the one presented in Figure 12. We see the user nodes, in blue, related to the article's nodes by interactions. In green, there are the entities contained in the articles and the relations among them.

The exact figure represents in dotted line the kind of relations this study tries to predict: would user U1 be interested in article A2?

The outcome of the RecBole file formatting process were the four mentioned files with the following dimensions:

File	Rows	Columns
mind_small_positive.inter	2473898	user_id, item_id, label
mind_small_positive.item	42417	item_id, category, subcategory
mind_small_positive.kg	6330557	head_id, relation_id, tail_id
mind_small_positive.link	117638	item_id, entity_id

Table 7 RecBole files dataset

Tabular data construction

In summary, the dataset used to start debugging the tabular algorithms, shown in Table 8, consists of a sample of 70 multinomial predictive variables, one binary objective variable, and 73985 rows. This tabular dataset does not contain either the User ID or the date column of the *behavior.tsv* file, neither the News ID, the Title, the Body, nor the URL column of the *news.tsv* files. These variables were discarded because of their type: identifiers, dates, and long string with very high cardinality.

The following paragraphs describe how this dataset was obtained from the initial information in MIND files *news.tsv*, *behavior.tsv*, and the information coming from the Wikidata API.

Variable	Role	Type	Levels	Missing
Clicked	Target	Binary	2	0
Category	Input	Nominal	15	0
Subcategory	Input	Nominal	204	0
P102_head	Input	Nominal	307	64411
P102_tail	Input	Nominal	45	64411
P103_head	Input	Nominal	223	62245
P103_tail	Input	Nominal	37	62245
P1050_head	Input	Nominal	121	66708
P1050_tail	Input	Nominal	53	66708
P106_head	Input	Nominal	513	3213
P106_tail	Input	Nominal	218	45961
P118_head	Input	Nominal	238	68298
P118_tail	Input	Nominal	33	68298
P131_head	Input	Nominal	513	10050
P131_tail	Input	Nominal	513	45207
P1340_head	Input	Nominal	130	64449
P1340_tail	Input	Nominal	15	64449
P1343_head	Input	Nominal	513	6677
P1343_tail	Input	Nominal	101	43389
P1411_head	Input	Nominal	253	64117
P1411_tail	Input	Nominal	130	64117
P1412_head	Input	Nominal	513	22967
P1412_tail	Input	Nominal	38	54995

P1532_head	Input	Nominal	416	67574
P1532_tail	Input	Nominal	26	67574
P1622_head	Input	Nominal	99	67340
P1622_tail	Input	Nominal	3	67340
P166_head	Input	Nominal	513	11664
P166_tail	Input	Nominal	413	51742
P172_head	Input	Nominal	380	61431
P172_tail	Input	Nominal	105	61431
P17_head	Input	Nominal	513	1435
P17_tail	Input	Nominal	119	33074
P1884_head	Input	Nominal	126	63863
P1884_tail	Input	Nominal	12	63863
P206_head	Input	Nominal	224	59584
P206_tail	Input	Nominal	120	59584
P21_head	Input	Nominal	513	3049
P21_tail	Input	Nominal	8	45748
P27_head	Input	Nominal	513	3200
P27_tail	Input	Nominal	77	45887
P2936_head	Input	Nominal	206	59521
P2936_tail	Input	Nominal	115	59521
P30_head	Input	Nominal	187	60994
P30_tail	Input	Nominal	9	60994
P31_head	Input	Nominal	513	386
P31_tail	Input	Nominal	513	2462
P361_head	Input	Nominal	513	8719
P361_tail	Input	Nominal	513	37505
P37_head	Input	Nominal	181	60692
P37_tail	Input	Nominal	59	60692
P39_head	Input	Nominal	255	64973
P39_tail	Input	Nominal	157	64973
P421_head	Input	Nominal	460	54746
P421_tail	Input	Nominal	59	54746
P463_head	Input	Nominal	462	55137
P463_tail	Input	Nominal	197	55137
P495_head	Input	Nominal	513	33557
P495_tail	Input	Nominal	42	64605
P5008_head	Input	Nominal	365	59937
P5008_tail	Input	Nominal	24	59937
P530_head	Input	Nominal	105	67259
P530_tail	Input	Nominal	25	67259
P552_head	Input	Nominal	65	67384
P552_tail	Input	Nominal	4	67384

P641_head	Input	Nominal	513	20280
P641_tail	Input	Nominal	54	58361
P6886_head	Input	Nominal	100	66655
P6886_tail	Input	Nominal	9	66655
P937_head	Input	Nominal	156	65917
P937_tail	Input	Nominal	50	65917

Table 8 Tabular dataset variables description

When building the data file to feed tabular algorithms, the problem is the amount of information collected. When flattened into a plain table, the sample size reached 5,1 million records. The number of predictive variables was again unmanageable. To build the variable set from the relations in the graph in Figure 12, each relation type, so-called property, was decomposed into two variables: *property_head* and *property_tail*. Considering that there are 873 different properties in the graph, it would result in 1746 variables.

For the purpose of selecting a starting property set, this study discarded those property values with lower representation than 5% of the most represented property. First, the property structure must be clear to understand this reasoning. In Figure 13, Pedro Sánchez, identified as the entity E8, has a property "President Of" identified as property P67, which value is Spain identified by entity E102.

Figure 13 Head, property, and tail example

Keeping only those properties with enough representation means that when grouping by property name and property value, and counting the number of the appearance of the most represented property value, all other properties with any of its value count higher than that 5% are kept. The following pseudo-code describes this filter.

```
algorithm filter relevant properties is
    input: Knowledge Graph as kg
    property value group = count(group(kg by property name and property value))
    max count = max(property value group)
    selected = filter(property value group with count > 0.05 * max count)
    return unique(selected)
```

Code 5 Filter relevant properties algorithm

Doing so, the number of properties that pass the threshold of 5% reduced from the initial 873 properties to 34. In this scenario, it is possible to model these relations with 68 variables: *head_property* variable and *tail_property* variable. With this number of variables, it is more feasible to start debugging which are the most relevant to model the objective click prediction variable.

This approach differs from selecting the most relevant variables with the most overall representation. Given that the most represented property is represented in 1,3% of the sample, filtering by the 5% representativeness would result in no properties filtered.

One characteristic that tabular algorithms need from the dataset used for modeling is that this information must contain both positive and negative clicks. It differs from the sampling that Graph Neural Networks (GNN) need. In GNN, the algorithms extract a negative sample by randomly creating a triple that is not contained in the dataset. For tabular algorithms, those samples indicating that the user did not click the article must be included in the dataset before using the algorithm itself. Indeed, negative samples for tabular algorithms could be generated in a previous step, but this mechanism is not included in algorithms like Neural Network models or Classification Trees.

As a means to get the negative sample in place for the tabular dataset, the *display_log* information is used. It is a log containing information about the click event when articles were displayed to the user. If a user clicks an article when displayed, it will be represented as "articleid-1" in the *behavior.tsv* file. On the contrary, it will be represented as "articleid-0" if it is not clicked. This information is used to build the positive and negative records in the tabular dataset. It is achieved by splitting those with the indicator of click "1" and setting them as positive samples, while the rest are set as negative samples.

Nonetheless, after filtering the number of property variables to 68 and building the negative samples, there are still 5.1 million records in the table. It is still too many records to be handled by the available resources, and two of the modeling and algorithm methods selected are Neural Networks and Random Select. A uniform random sample of one thousand (1000) users was extracted from the users' list to reduce the sample size. These thousand users' selections helped choose only those records related to them, resulting in a total of 73985 rows.

Missing Value Assignment and Low Representativity Class Value Imputation

A first exploration of the tabular dataset results in 75.7% (53 over 70) of the variables have more than 50 value levels, and 88.5% (62 over 70) have more than 10% of missing values. The following tables show those variables with higher cardinality and the most missing values.

Variable	Levels
P106_head	513
P131_head	513
P131_tail	513
P1343_head	513
P1412_head	513

Table 9 Top 5 variables with the most significant number of categorical levels

Variable	Missing
P118_head	68298
P118_tail	68298
P1532_head	67574
P1532_tail	67574
P552_head	67384

Table 10 Top 5 variables with the most significant number of missing values

The imputation strategy is followed to solve the property variables' emptiness, replacing those missing values with the keyword _EMPTY_. Then, to reduce the dispersion problem, the variable values with less than 1% of the overall representativity were classified in a common category called _OTHER_.

In this case, we must notice that this 1% used for category imputation differs from the percentage chosen in the previous section to select those variables relevant to the study. The percentage in this section is computed over the reduced sample, 73985 rows, while the previous 5% was computed over the most relevant property value count in a 5.1 million records sample.

This strategy helps the study eliminate Neural Networks and Logistic Regression's missing values.

Variable Selection

Independently from the variable selection done while building the tabular dataset that was performed merely in representativeness, seventy (70) variables are still too many, and a selection of the most relevant, regarding the modeling of the objective variable, must be done.

This study has used five (5) different methods and has arranged the variables in order of appearance to sort the relevance of the seventy variables regarding the objective variable.

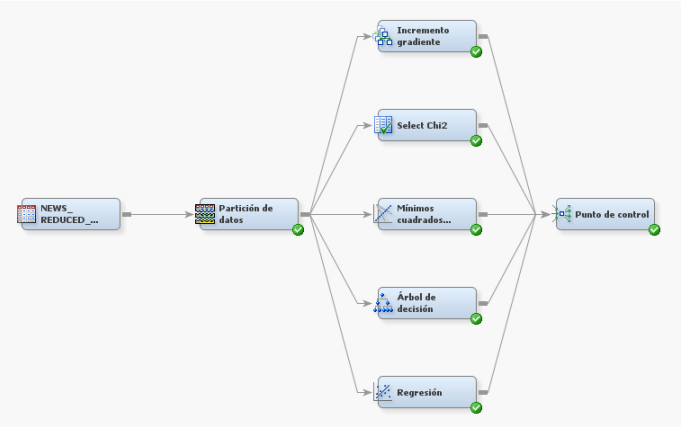


Figure 14 Variable selection used methods

The five methods used to select the most relevant estimators were: Partial Least Square (PLS), Gradient Boosting (GB), Chi-Square Selection, Decision Tree, and Logistic Regression. The parameters used for each of these algorithms in SAS Enterprise Miner were the following: PLS with 0.1 as estimators threshold and 0.8 for VIP importance threshold; GB with minimum category and leaf size to 74 (1‰ of sample size), max depth of two and 50 iterations max; Chi-Square with minimum R^2 set to $5 \cdot 10^{-8}$; Decision Tree with minimum category and leaf size to 74 (1‰ of sample size), max depth of eight levels; Logistic Regression with Akaike Information Criteria as selection criteria and stepwise selection algorithm.

The output of aggregating the number of variables selected by each method is shown in Table 11.

Nº of methods	Nº of variables	Variables
one method	68	P102_head P102_tail P103_head P103_tail P1050_head P1050_tail P106_head P106_tail P118_head P118_tail P131_head P131_tail P1340_head P1340_tail P1343_head

		P1343_tail P1411_tail P1412_head P1412_tail P1532_head P1532_tail P1622_head P1622_tail P166_head P166_tail P172_head P172_tail P17_head P17_tail P1884_head P1884_tail P206_head P206_tail P21_head P21_tail P27_head P27_tail P2936_head P2936_tail P30_head P30_tail P31_head P31_tail P361_head P361_tail P37_head P37_tail P39_head P39_tail P421_head P421_tail P463_head P463_tail P495_head P495_tail P5008_head P5008_tail P530_tail P552_head P552_tail P641_head P641_tail P6886_head P6886_tail P937_head P937_tail category subcategory
two methods	50	P102_head P103_head P103_tail P1050_head P1050_tail P106_head P106_tail P118_head P131_tail P1340_tail P1343_head P1343_tail P1412_head P1412_tail P1532_head P1622_head P166_head P166_tail P172_head P172_tail P17_tail P1884_tail P206_head P206_tail P21_head P27_tail P2936_head P2936_tail P30_head P30_tail P31_head P31_tail P361_head P361_tail P37_head P37_tail P39_tail P421_tail P463_tail P495_head P5008_head P5008_tail P552_tail P641_tail P6886_head P6886_tail P937_head P937_tail category subcategory
three methods	22	P102_head P106_tail P131_tail P1340_tail P1343_tail P1532_head P166_tail P172_head P17_tail P206_tail P27_tail P2936_tail P31_head P31_tail P361_tail P37_head P421_tail P463_tail P641_tail P6886_head category subcategory
four methods	8	P1343_tail P17_tail P27_tail P31_head P31_tail P361_tail category subcategory
five methods	3	P31_head P31_tail subcategory

Table 11 Variable selection set based on joined selection methods

The Random Select method was also included during this study, but with negative results. This method is parametrized usually with a high number of iterations, in hundreds, to score later the variables' relevance, counting the number of occurrence of variables in the linear models performed. However, this was a problem. The Random Select procedure took around 4 hours to complete when running one iteration. To run a flawed test with Random Select, it might have used at least one hundred (100) seeds, meaning that it would have taken over 400 hundred hours, more than two weeks. For this reason, the Random Select method was discarded from the selection algorithms.

Transformation of Categorical Variables with High Cardinality

Up to this point, the data transformation consisted of creating new variables, reducing the cardinality of the variables created, assigning values to missing data, grouping disperse categories, and reducing the sample size. In order to contrast the performance of all the categorical predictors included in our dataset, this study uses a technique to transform those variables into continuous numerical variables. The technique used is called Target Encoding.

Target encoding is a Bayesian method that transforms input variables regarding the value the target variable has for a given range of values in the variables to transform. In our case, the transformation performed is a "mean" target encoding transformation. It will replace all the categories' values for a categorical variable into each category's clicked mean probability.

This transformation is applied in input variables only in those scenarios where tabular methods perform best without the transformation to check whether it exceeds the given results.

The library used to apply target encoding is the H2O library for R. Apart from encoding categorical variables in terms of the target means, this library prevents overfitting the model of the new variables generated by using only part of the training variables in the encoding process and adding a regularization parameter called *noise.*

When partitioning the training dataset in the encoding process, the library uses the so-called *K-Fold* method: it splits the dataset into *K* parts and uses *K-1* of them to compute the target mean. The computed *K-1* mean target values are used as the value in the remaining fold to replace the categorical variable. This process is repeated *K* times.

Figure 15 Target encoding representation from https://brendanhasz.github.io/2019/03/04/target-encoding

To profound in the code used to calculate the Target Encoding transformation, please refer to annexes 1.8 to 1.12 below.

Modeling

This study has used three different software during the modeling process: SAS Base, R, and Python. The first and second were used to implement Logistic Regression, Decision Trees, Bagging, Gradient Boosting, XGBoost, Random Forest, and AutoML execution. Python was used to implement MKR, KGAT, and RippleNet.

The order of implementation was first the graph methods in Python, and later the tabular methods with SAS and R. To compare all these models, this study has chosen the metric called Area Under the Curve (AUC), and to be fair for them all, it has used the same number of random seeds in the repetition of each training case: ten.

AUC measures the Goodness-of-fit of a model by comparing the sensitivity and one minus the model's specificity in a range of the event probability threshold points. This metric was chosen mainly because it was the only available among all the training methods and software. The restriction came with the RecBole library for model fit metrics. There were only two metrics available: AUC and Log Loss, also called Cross-Entropy. Cross-Entropy is not among the metrics that could be easily extracted from the tools available in this study. On the contrary, AUC was easy to extract in all cases, and so, it was chosen as the comparison metric.

To summarize the strategy used by all the models in this study, Table 12 shows all hyper-parameters and methods combined. Meanwhile, the following chapters describe what those hyper-parameters mean and which values they take.

Model Name	Hyper-parameters tuned
MKR	Learning Rate, Predictive Function, Number of Layers, L^2, Dropout probability
KGAT	Learning Rate, Embedding Size, L^2, Dropout probability
RippleNet	Learning Rate, Embedding Size, Negative Sampling, L^2, Memory, Graph Weight
Neural Network	Nodes in hidden Layer, Optimization Function, Input Variables
Logistic Regression	Input variables
Trees in general	Maximum Depth, Category Size, Split Size, Leaf Minimum Size, and Input Variables
Gradient Boosting and XGBoost	Learning Rate, Shrinkage, Input Variables
Random Forest and Bagging	Maximum Number of trees to pool, Number of Randomly selected variables in each split, Input Variables
AutoML	Included Algorithms and Maximum Execution Time

Table 12 Hyper-parameters tuned in each method

Tuning Strategy for Graph Neural Network Methods

Each of the Graph Neural Network algorithms used in this study has its own and shared hyper-parameters. One of the most mentioned hyper-parameters is the L^2 regularization factor. For MKR, it is the λ_2 factor for the equation (14), concerning KGAT, the regularization factor is λ in equation (15), while for RippleNet the factor is $\frac{\lambda_1}{2}$ in equation (16). The regularization factor allows the optimization algorithm to generalize the model better, adding a component that does not depend on the error between ground truth and predicted value.

$$\mathcal{L}_{MKR} = \cdots - \lambda_2 \|W\|_2^2 \tag{14}$$

$$\mathcal{L}_{KGAT} = \cdots + \lambda \|\Theta\|_2^2 \tag{15}$$

$$\mathcal{L}_{RippleNet} = \cdots + \frac{\lambda_1}{2}\left(\|V\|_2^2 + \|E\|_2^2 + \sum_{r \in R} \|R\|_2^2\right) \tag{16}$$

Another common hyper-parameter is the Learning Rate. It is a positive decimal factor, typically much lower than one, modulating the increment speed in the Gradient Descent optimization process. It could also be seen as a regularization factor because it shrinks the overfitting by dropping the speed of adjustment of the model to the training dataset.

A parameter that MKR and KGAT share, but RippleNet does not, is the dropout probability. This parameter tells the Aggregation function in the embedding process how feasible it is to include the weights of a specific neighbor in the sum. The result of increasing the dropout probability is that the Aggregation function will often ignore the weights of a neighbor in the embedding update step. It is also a regularization method, preventing the model from overfitting the graph structure.

The last parameter common to MKR, KGAT, and RippleNet is the embedding size. It helps to define the number of dimensions that embeddings may have. RecBole, by default, sets it to 64, meaning that the array describing a node in embedding format will have 64 positions. Increasing this parameter leads to a better separation between nodes in the embed Euclidean space.

All other parameters like *low_layers_num* or *attention_layer* are specific to each algorithm. When training the three algorithms, a hyper-parameters grid was defined. Below it is shown for all the cases.

Parameter	Possible Values
learning_rate	0.0001 or 0.001
low_layers_num	1 or 2
high_layers_num	1 or 2
use_inner_product	True or False
reg_weight (L^2)	10^{-5} or 10^{-6}
dropout_prob	0 or 0.05

Table 13 Tuning Grid for MKR

Parameter	Possible Values
learning_rate	0.001, 0.005, or 0.01
embedding_layer	2, 4, 8, 64, or 128
attention_layer	2, 4, 8, 64, or 128
prediction_layer	2, 4, 8, 64, or 128
reg_weight (L^2)	10^{-4}, 10^{-5}, or 10^{-6}
mess_dropout	0.5, 0.3, 0.1, 0.01, or 0.001

Table 14 Tuning grid for KGAT

Parameter	Possible Values
learning_rate	0.0001, 0.0005, 0.001, or 0.01
embedding_size	64 or 128
training_neg_sample_num	2 or 3
n_memory	2, 3, 4, or 8
reg_weight (L^2)	0.01 or 0.001
kg_weight	0.01 or 0.001

Table 15 Tuning grid for RippleNet

Graph Neural Network Methods Modeling

As mentioned, the RecBole library has been used to implement the GNN models. Instead of using the RecBole implementation for grid hyper-parameter tuning, a manual approach is chosen in this study. This approach aims to have more flexibility in the configuration used in each iteration and when obtaining the results from each epoch of the training process.

Each training case had its own configuration and its shared configuration that may be built on the run. Also, by implementing the training grid manually, the specialist can take advantage of the CPU process parallelization, gaining time when running several training setups simultaneously.

The flexibility needed to build a different output from the automated grid trainer of RecBole regards the fact that this study uses the BoxPlot chart extensively as a comparison method. Nonetheless, RecBole does not automatically provide an integrated way of building the required information for the BoxPlot chart, so it must be done manually.

To enhance the training grid, this study uses JSON files to describe the configuration of each training case. The format of each of these files will be as shown next:

```
{"case": 1, "learning_rate": 0.01, "layers": [8, 4, 2], "reg_weight": 0.0001,
"mess_dropout": 0.1, "seed": 5678}
{"case": 2, "learning_rate": 0.01, "layers": [8, 4, 2], "reg_weight": 0.0001,
"mess_dropout": 0.1, "seed": 5778}
{"case": 3, "learning_rate": 0.01, "layers": [8, 4, 2], "reg_weight": 0.0001,
"mess_dropout": 0.1, "seed": 5878}
```

Code 6 Test grid example

In each line of the test grid file, the information about the hyper-parameters to be used in each case could be seen. This kind of file feeds the loop to train each case split into several CPU processes. The following Code 7 shows how the manual trainer splits the cases into the desired number of processes, where *run_iteration* is the function that trains a case given its parameters, *n_threads* is the desired number of processes, and *config_dict* is the variable containing the information of one line of the above test grid file.

```
for i in range(0, len(cases), n_threads):
    configs = cases[i:i + n_threads]
    threads = list()
    for config_dict in configs:
        x = threading.Thread(target=run_iteration, args=(config_dict,))
        threads.append(x)
        x.start()
```

Code 7 Training loop for each test case and CPU process

For more detail about the code that performs the model training, please refer to annexes 1.3 to 1.5 below.

Tuning Strategy for Tabular Methods in SAS

Tabular methods are compound by Logistic Regression, Neural Networks, Decision Trees, Bagging, Gradient Boosting, XGBoost, Random Forest, along with an AutoML execution in r, each of them accepting different tuning hyper-parameters.

Nonetheless, what is common to almost all of them are the specific variable sets used to train the models. The selection is based on five methods described before, discarding the selection made by only one method because this selection left out only two variables out of seventy.

Neural Network Tuning in SAS

In addition, in the cases for those sets selected by two and three methods, the number of variables, 50 and 28, was high, leading to a dilated time of training in SAS Base for Neural Networks and were discarded from the training grid in this environment. It caused execution times of the order of hours for one epoch of a single hyper-parameter test case:

```
NOTE: The data set WORK.DATOS has 59187 observations and 31 variables.
NOTE: The data set WORK.VALIDA has 14797 observations and 31 variables.
NOTE: Closing the Neural Application.
NOTE: PROCEDIMIENTO NEURAL used (Total process time):
      real time           9:31:47.21
      cpu time            9:28:57.98
```

Code 8 Execution time for a Neural Network with two nodes in the hidden layer, activation function Tanh, optimization algorithm Levenberg Marquardt, and 28 estimators

Two more hyper-parameters were tuned in this study to enhance the Neural Network models in SAS Base: the number of nodes in the hidden layer and the optimization algorithm. With the number of nodes in the hidden layer, we can modulate the explanatory

capacity of the Neural Network, trading off the computation time and the number of samples per parameter. The optimization algorithm impacts the convergence time of the optimization process and the quality of the fit. These two hyper-parameters are decomposed in Table 16 in the cases used during this study.

Parameter	Possible Values
number of nodes of the hidden layer	2 to 10
optimization algorithm	Levenberg Marquardt or Back Propagation
nº of input variables	8 or 3

Table 16 Hyper-parameters for Neural Networks in SAS Base

An additional setup is usually included when parametrizing Neural Networks. The Early Stopping point. The calculation of this parameter in SAS Base is done by iteration over a maximum of 10000 repetitions and scoring the error against the objective variable using two datasets: train and validation dataset. Afterward, the macro that executes this process compares the descent in error for each iteration, stopping when the error difference is incremental instead of decremental after three consecutive iterations. The stop iteration point is called the Early Stopping point.

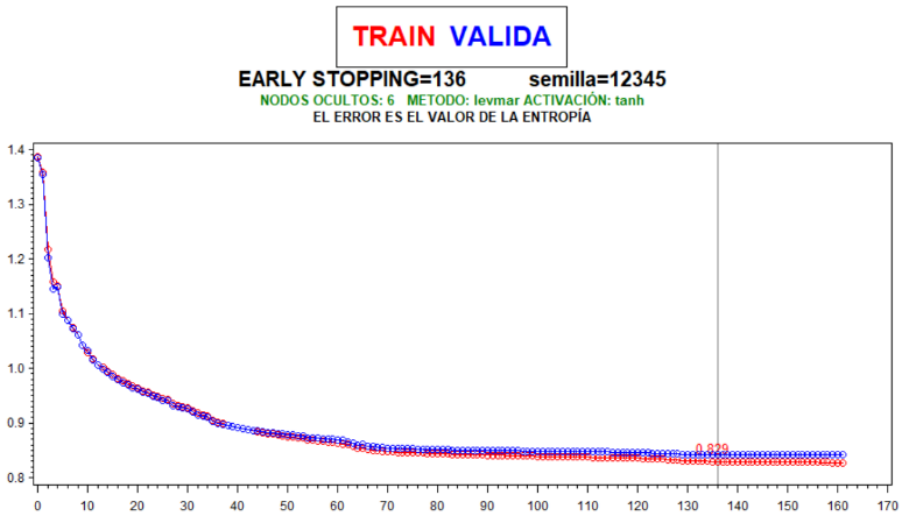


Figure 16 Neural Network N6 Levenberg Marquardt TanH Early Stopping Calculation

The inputs to the Early Stopping macro are the optimization algorithm, the number of nodes in the hidden layer, the activation function, the train/validation ratio, and the variable set description.

Figure 16 shows the output of the Early Stopping macro. This figure shows how at the 136th iteration, the Neural Network started to perform negatively for three consecutive iterations. The macro's code to calculate the Early Stopping comes from the subject "Machine Learning Techniques," taught by Mr. Javier Portela, responsible of this master's degree. As for the rest of the code, macro's code for Early Stopping calculation is attached to this document in the annexes section below.

Nodes in the hidden layer	Back Prop eight variables	Back Prop three variables	Levmar eight variables	Levmar three variables
two	13	13	234	141
three	13	13	431	166
four	13	173	504	59
five	13	191	541	42
six	13	184	136	36

seven	-	185	-	40
eight	-	165	-	30
nine	-	169	-	24
ten	-	167	-	27

Table 17 Each of the figures in this table corresponds to an Early Stopping point for each of the Neural Networks from two to ten nodes, Back Propagation and Levenberg Marquardt optimization functions in the grid. For example the second column shows that the point of Early Stopping is 13 for all the models with Back Propagation optimization function, eight input variables, and two to six nodes

This previous table shows the Early Stopping point for both optimization algorithms, variable set parameters, and the number of nodes. Nonetheless, there is no information about the Early Stopping point for those Neural Networks over six nodes in its hidden layer and eight variables as input. It is because each hidden node's number increase signified a considerable delay of the Early Stopping point calculation due to the number of parameters to optimize. The equation (17) below shows the formula to obtain the number of parameters for a Neural Network given a certain number of estimators and nodes in the hidden layer.

$$Number\ of\ parameters = k(h+1) + h + 1$$

$$k = number\ of\ inputs \tag{17}$$

$$h = number\ of\ nodes\ in\ the\ hidden\ layer$$

REP_P1343_tail
REP_P17_tail
REP_P27_tail
REP_P31_head
REP_P31_tail
REP_P361_tail
REP_category
REP_subcategory

TI_REP_P1343_tail1
TI_REP_P1343_tail2
TI_REP_P1343_tail3
TI_REP_P1343_tail4
TI_REP_P1343_tail5
TI_REP_P1343_tail6
TI_REP_P1343_tail7
… 68 more
TI_REP_subcategory29
TI_REP_subcategory30
TI_REP_subcategory31
TI_REP_subcategory32
TI_REP_subcategory33
TI_REP_subcategory34
TI_REP_subcategory35

Table 18 Transformation from eight nominal variables selected by four selection methods to 82 dummy variables

For the case of eight input variables and seven nodes in the hidden layer, the number of parameters to be optimized would have been $k(h+1)+h+1 = 8(7+1)+7+1 = 73$ when all the input variables were continuous. However, the input variables in this study are categorical, which means that before feeding the Neural Network algorithm, each category for each nominal variable must be exploded into dummy variables. Mainly, for the eight input variables selected by four selection methods, the dummy transformation generates 82 dummy variables.

Recalculating the number of parameters in the neural network after the dummy variables transformation, we obtain $82(7+1)+7+1 = 664$. With one hidden node less, the number of parameters decreases 12,5%, and the computation time also decreases 22,1% from 2 hours 11 minutes to 1 hour and 42 minutes, for only one hyper-parameter test case and one seed. Given this information, it was decided to stop at six nodes in the hidden layer for the case where eight variables formed the input, in order to save time in the already elapsed training process.

Tree-Based Algorithms Tuning in SAS

To tune tree algorithms, this study has used in SAS four different hyper-parameters apart from the previous variable selection: shrinkage, depth, the maximum number of trees, and in the case of Random Forest, the number of variables selected at the node level.

Shrinkage is another form to name the Learning Rate parameter. When reducing its value, the Gradient Boosting algorithm decreases its learning speed, therefore dropping the risk of overfitting the model to the training dataset. Depth is a standard parameter for tree algorithms that defines the maximum number of levels in the tree hierarchy. Maximum trees and the number of variables selected at node level hyper-parameters refer to the Random Forest algorithm tuning. The former limits the number of trees used when pooling results, and the latter limits the input variables number to the selection algorithm when deciding the splitting variable in a node. Down here, we can find how the configuration is distributed for each of the tree-based algorithms.

Algorithm	Hyper-parameter	Values
Decision Trees and Gradient Boosting	maximum depth	4 or 6
Gradient Boosting	Learning Rate, shrinkage	0.001, 0.03 or 0.1
Random Forest	maximum number of trees to pool	100 or 200
Random Forest	randomly selected variables in each split	6 or 12
All	number of input variables	50, 22, 8, or 3

Table 19 Parameters for Decision Trees, Gradient Boosting and Random Forest in SAS Base

Tuning Strategy for Tabular Methods in R

A similar training process is repeated in R. R is statistical software that implements plenty of machine learning models apart from statistical and data manipulation functions. In this case, the machine learning algorithms implemented are Logistic Regression, Neural Networks optimized by Back Propagation, Decision Trees, Bagging, Random Forest, Gradient Boosting, and XGBoost.

With R, the hyper-parameters tuned up were slightly different. For Neural Networks, this study used the number of nodes in the hidden layer, the maximum number of iterations during the optimization process, and the decay factor, also called shrinkage. The decay is a hyper-parameter for the *avNNet* package that impacts the convergence speed of the Back Propagation algorithm when optimizing the Neural Network parameter. The same hyper-parameter as Learning Rate in SAS Base or Graph Neural Networks seen previously.

Algorithm	Hyper-parameter	Values
avNNet	hidden layer nodes	2 to 10
avNNet	Learning Rate, decay	0.01 or 0.1
Random Forest and Bagging	randomly selected variables in each split	50, 22, 8, 3, or 2
Random Forest and Bagging	maximum number of trees	50
Decision Trees, Random Forest, and Bagging	minimum sample size in each node	74
Gradient Boosting and XGBoost	Learning Rate, shrinkage, eta	0.001, 0.008, 0.03, or 0.1
Gradient Boosting and XGBoost	maximum tree depth	2, 4, or 6

Table 20 Hyper-parameters for avNNet, Decision Trees, Bagging, Random Forest, Gradient Boosting, and XGBoost in R

For tree algorithms, the hyper-parameters used were the number of variables to select in each node split randomly, the maximum number of trees to pool, the minimum number of

observations per node, the tree depth, and shrinkage. The table above shows the tuning used in each case in R.

H20 AutoML Tuning Strategy

A second approach to find a better tuning of models is to use the AutoML functionality offered by statistical packages as H2O. This method randomly sets the values of the hyper-parameters used in the models selected during a limited number of iteration or a restricted time. Once the time is over, or the loop has reached the maximum number of repetitions, AutoML builds the leader board of the trained models, sorted in descending order of performance.

In this study, the limitation imposed was ten hours spent trying to find the best tuning. The algorithms included in the AutoML tuning were H20 Deep Learning, H2O Gradient Boosting Machine, and H20 Random Forest. Logistic Regression was left apart because of its time spent building interactions among input variables. It consumes all the ten hours training time limitations before building any model.

Trained Algorithms	Overall Maximum Execution Time
GBM, RF, and Deep Learning	10 Hours

Table 21 H2O AutoML Setup

Tabular Methods Implementation

Thanks to the procedures provided by Javier Portela García-Miguel during the "Machine Learning Techniques", links to this master's degree study, it was possible to carry out the training grid proposed in the previous chapters. Without those already implemented methods, the development time would have increased notoriously to achieve similar functionality.

Despite the time saved in the development process, some extra effort had to be invested in adapting the training functions to the particular case of the dataset herein used. First, the methods had to be adapted to collect the Area Under the Curve (AUC) indicator, feature which was not implemented previously.

Code 9 highlights the lines in the provided *cruzadalogistica* method to capture each iteration's scoring information in the grid process. Afterward, the scoring information was used to compute the AUC indicator for each iteration. The AUC indicator is crucial because it is the transversal comparison metric used to evaluate the Goodness-of-fit for each model trained in this study. The Assessment chapter describes how the AUC computation was done for cases that were needed.

```
%macro
cruzadalogistica(archivo=,vardepen=,categor=,ngrupos=,sinicio=,sfinal=,objetivo=tasafall
os);
        title ' ';

        data final;
        run;

        data sal_final;
        run;
        %do semilla=&sinicio %to &sfinal;
…
                                proc logistic data=tres noprint;
                                        class &categor;
                                        model vardep=&categor;
                                        output out=sal p=predi;
                                run;
```

```
                                data sal_final;
                                       set sal_final sal;
                                run;
…
```

Code 9 Capture of the Logistic Regression score out of the cruzadalogistica method

A similar modification was done for the rest of the procedures in SAS Base that did not calculate the AUC: *cruzadabinneural*, *cruzadarandomforestbin, cruzadabaggingbin, cruzadatreeboostbin.* Annex 1.7 below contains the modified code for the mentioned procedures.

In R, similar situation happened, the procedures provided in this master's degree lacked the ability to compute the AUC indicator for each iteration in the grid training process. This study has modified the R functions to enhance them in the way shown in Code 10.

```
cruzadalogistica <- function(data = data,
                             vardep = NULL,
                             listclass = NULL,
                             grupos = 4,
                             sinicio = 1234,
                             repe = 5) {
…
  auc_func <- function(x, y) {
    curvaroc <- roc(response = x, predictor = y)
    auc_value <- curvaroc$auc
    return(as.numeric(auc_value))
  }

  mediasbis <- preditest %>%
    group_by(Rep) %>%
    summarize(auc = auc_func(obs, Yes))
…
```

Code 10 AUC calculation function example for Logistic Regression in R

The remaining code in annexes 1.8 to 1.12 below describes how the functions were modified for training algorithms as avNNet, Logistic Regression, Gradient Boosting, Random Forest, and XGBoost achieving the computation of the AUC indicator.

Thereafter comes the most accessible part: invoking the macro defined for training. This chapter shows an example of each SAS Base and R environment. Nevertheless, the full implementation of all the test cases described in the tuning strategy sections resides in Annexes 1.13 to 1.20.

Both codes 12 and 13 above implement the training macros' invocation for two of the best scoring models in this study: Gradient Boosting of depth six and shrinkage 0.1 in SAS Base and the ten nodes Neural Network with Learning Rate of 0.1 in R. The two of them use the fifty variables selected from two selection methods. It also shows how they save the relevant information used later to build the AUC in the case of the Gradient Boosting model and compare the results with Python in either case.

```
%cruzadatreeboostbin(
        archivo=news_reduced_clean_two_methods,
        vardepen=REP_clicked,
        categor=REP_P102_head REP_P103_head REP_P103_tail REP_P1050_head REP_P1050_tail
REP_P106_head REP_P106_tail REP_P118_head REP_P131_tail REP_P1340_tail REP_P1343_head
REP_P1343_tail REP_P1412_head REP_P1412_tail REP_P1532_head REP_P1622_head REP_P166_head
REP_P166_tail REP_P172_head REP_P172_tail REP_P17_tail REP_P1884_tail REP_P206_head
REP_P206_tail REP_P21_head REP_P27_tail REP_P2936_head REP_P2936_tail REP_P30_head
REP_P30_tail REP_P31_head REP_P31_tail REP_P361_head REP_P361_tail REP_P37_head
REP_P37_tail REP_P39_tail REP_P421_tail REP_P463_tail REP_P495_head REP_P5008_head
REP_P5008_tail REP_P552_tail REP_P641_tail REP_P6886_head REP_P6886_tail REP_P937_head
REP_P937_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
```

```
        sfinal=12354,
        leafsize=74,
        iteraciones=100,
        shrink=0.1,
        maxbranch=2,
        maxdepth=6,
        mincatsize=74,
        minobs=74,
        objetivo=tasafallos
);

data Results.gbd601v2;
        set final;
run;

data Results.sal_gbd601v2;
        set sal_final;
run;
```

Code 11 Gradient Boosting training in SAS Base with fifty input variables selected by two selection methods, a maximum depth of six levels and shrinkage of 0.1

```
avnnetn2c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(2),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn2c01$modelo = "avnnet-n2-c0.1-v2"
save.image("avnnet-n2-c0.1-v2.RData")
```

Code 12 Neural Network training with avNNet package in R with two nodes in the hidden layer, the Learning Rate of 0.1 and fifty input variables selected by two selection methods

When talking about Neural Networks implementation for the two-methods selection variable set, we consider a significant number of parameters. For the largest Neural Network computed in R, fifty nominal input variables were transformed into 272 dummy input variables, and at the same time, those dummy variables turned into 3003 model parameters. The *MaxNWts* setup parameter of the avNNet package had to be modified considering the number of parameters handled. *MaxNWts* is a setup parameter that allows avNNet to exceed the default limit of 1000 parameters for Neural Network models.

H2O AutoML Implementation

The code for the AutoML implementation could be reviewed in annex 1.21 below, which outputs 218 models. From those 218, the top ten models were selected to study their variance with the code hosted in the same annex.

Model	AUC	Lear Rate	Trees	Max Depth	Max Leaves
GBM_model_24	0.9472226	0.1	61	17	2736
GBM_model_49	0.9471711	0.1	70	16	2935
GBM_model_20	0.9470626	0.1	77	15	1969
GBM_model_122	0.9469963	0.1	73	17	1218
GBM_model_119	0.9469763	0.1	75	16	1547
GBM_model_36	0.9467395	0.1	96	14	935
GBM_model_104	0.9466398	0.1	65	17	1657

GBM_model_42	0.9462066	0.1	70	14	1213
GBM_model_32	0.9460080	0.1	60	16	1279
GBM_model_35	0.9459466	0.1	70	15	1029

Table 22 Top ten models from AutoML tuning

All top ten models extracted from H20 AutoML after ten hours of execution are Gradient Boosting Machine models. The first model different from GBM is a Deep Learning model in position 45 of the leader board.

To evaluate the variance of these top ten models, they are trained again with the same hyper-parameters but different seeds. The hyper-parameters used were Learning Rate, Number of Trees, and Maximum Depth, repeated for ten different seeds, like the rest of the tabular algorithms in this study.

Tabular Model Execution Results

As soon as a training macro finished its execution, the results were saved into files that, depending on the implementation, contained raw scoring results or computed AUC results.

In general, all SAS Base results hold raw scoring results in *sas7bdat* file format. The scoring results consist of event probability and the ground-truth event value for each observation in the test dataset used for each test seed, in each cross-validation group, and each test case.

	Predicted: vardep=1	Predicted: vardep=0	Replacement: clicked	Replacement: P103_head	Replacement: P103_tail	Replacement: P102_head	Replacement: P1050_head	Replacement: P1050_tail
1	.	.	.					
2	0.3183294617	0.6816705383	1	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_
3	0.9003722506	0.0996277494	1	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_
4	0.9720667885	0.0279332115	1	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_
5	0.794351987	0.205648013	1	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_
6	0.9392680106	0.0607319894	1	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_
7	0.235786525	0.764213475	1	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_
8	0.1960275773	0.8039724227	0	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_
9	0.9656312492	0.0343687508	1	_OTHER_	Q1860	_EMPTY_	_EMPTY_	_EMPTY_
10	0.0541570282	0.9458429718	0	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_
11	0.9690385096	0.0309614904	1	_EMPTY_	_EMPTY_	_OTHER_	_EMPTY_	_EMPTY_
12	0.0927273393	0.9072726607	0	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_
13	0.2864167636	0.7135832364	0	Q22686	Q1860	_OTHER_	Q22686	Q84263196
14	0.1458251866	0.8541748134	0	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_
15	0.1973884627	0.8026115373	0	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_	_EMPTY_

Figure 17 Scoring results for Gradient Boosting model of two methods selection input variables, maximum depth of six and shrinkage 0.1

For R, the results include the AUC information of each validation group and test case inside a *.RData* file that could be recovered later for comparison with other models.

	Rep	tasa	auc	modelo
1	Rep01	0.1870945	0.8822566	avnnet-n2-c0.1-v2
2	Rep02	0.1896221	0.8811859	avnnet-n2-c0.1-v2
3	Rep03	0.1842696	0.8843690	avnnet-n2-c0.1-v2
4	Rep04	0.1842966	0.8814715	avnnet-n2-c0.1-v2
5	Rep05	0.1882029	0.8820611	avnnet-n2-c0.1-v2
6	Rep06	0.1837289	0.8831596	avnnet-n2-c0.1-v2
7	Rep07	0.1860537	0.8817731	avnnet-n2-c0.1-v2
8	Rep08	0.1819583	0.8856857	avnnet-n2-c0.1-v2
9	Rep09	0.1874324	0.8784061	avnnet-n2-c0.1-v2
10	Rep10	0.1887976	0.8818541	avnnet-n2-c0.1-v2

Figure 18 AUC information for ten hidden layer nodes Neural Network implemented with aVNNet with shrinkage of 0.1

Assessment

The most crucial question in this study when assessing the models' results is the comparison metric. As mentioned in the Modeling chapter, the metric named to set side by side each model was the Area Under the Curve. It was the most accessible metric to obtain from each of the procedures used in the training phase in each software. In any case, it was not straightforward for all the models.

Another topic is the number of seeds used to train and check the models. In this study, the number of ten seeds was selected because of the inflated execution time of Neural Networks and Graph Neural Networks models.

Finally, cross-validation methodology was not applied in all the cases. For Graph Neural Network, only repeated training with ten seeds was used. On the other hand, repeated cross-validation of ten seeds and four validation groups was done for the tabular methods. It has its implication in the results section when choosing the best model for the best use case.

SAS Base Macro Output Retrieval

In chapter Tabular Methods Implementation, we saw how SAS Base macros saved the result of each training test case into a file, which keeps the scoring information about each record of the test dataset check. Mainly, to use the outputs' scoring information, specific functions were built using Python, *Pandas* library, and *SKLearn* library.

```
def gb_auc(name, model, directory):
    df = pandas.read_sas(name)
    df = df.dropna(subset=['REP_clicked', 'P_vardep1'])
    group = df.iloc[0]['grupo']
    i = 0
    with open(f'{directory}{os.path.sep}auc{os.path.sep}{model}.csv', 'w') as f:
        for index, row in df.iterrows():
            if group != row['grupo']:
                y = df['REP_clicked'][i:index]
                pred = df['P_vardep1'][i:index]
                fpr, tpr, thresholds = metrics.roc_curve(y, pred, pos_label=1)
                auc = metrics.auc(fpr, tpr)
                f.write(f'{model}; {auc}\n')
```

```
                i = index
                group = row['grupo']
```

Code 13 AUC extraction function in Python for Gradient Boosting results in SAS Base format

The code above converts the results for a specific model, Gradient Boosting, initially hosted in *sas7bdat* format with scoring information to a CSV file with AUC calculation divided by validation groups. After obtaining the CSV files with the AUC information of all the model training in SAS, we can build the comparative charts with the other R and RecBole models. By contrast, not all the SAS models use the same transformation function in Python. Annex 1.22 below points out how the process is carried out for the rest of the models.

Python AUC Metrics

All steps, previous to this point, serve to collect the Area Under the Curve (AUC) information about the selected models trained by a grid strategy in different training environments. Now, to visualize how well those models behave, it is necessary to build the BoxPlot charts that compare the AUC metrics performed by each model in terms of maximum, minimum, quartiles, mean, and outliers.

The BoxPlot charts are built using a standard representation for all models, independently of the software origin. Python helped harmonize and compare in this study all the AUC results using the *Pandas* and *Matplotlib* libraries.

```
from os import path
import pandas as pd
import glob
import matplotlib.pyplot as plt


df = None
for result_file in
glob.glob(f'..{path.sep}results{path.sep}Base{path.sep}all{path.sep}*v2.csv'):
    if df is None:
        df = pd.read_csv(result_file, sep=';', names=['modelo', 'auc'])
    else:
        df = pd.concat([df, pd.read_csv(result_file, sep=';', names=['modelo', 'auc'])],
ignore_index=True)
ax = df.boxplot(column=['auc'], by=['modelo'], rot=45, fontsize=10)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("SAS Base Software V2 Variable Selection Models")
plt.suptitle('')
plt.show()
```

Code 14 Boxplot chart for SAS models with fifty variables as input selected by two selection methods

This code shown above builds the chart displayed right below, corresponding to those models implemented in SAS Base with the previous selection of fifty variables as input from two selection methods.

As we could see in the code and chart, each representation comes from many CSV files explicitly named using a concrete nomenclature and location. In this example, the directory containing the CSV files with the AUC information is the *Base* for SAS Base, and the file name ends width **v2.csv* for two methods variable set selection.

Annexes 1.23 and 1.24 below describe the fragments of code used to build all the AUC charts for each of the models and input variable sets.

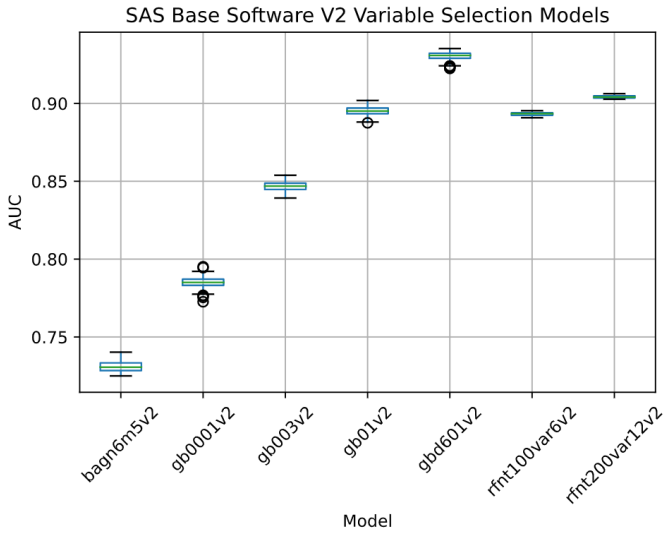


Figure 19 Boxplot for AUC metrics of SAS Base models trained using fifty variables selected from two selection methods

Results & Discussion

The output of this study comprises the AUC and complexity metrics for each of the models implemented: Logistic Regression, Decision Trees, Bagging, Random Forest, Gradient Boosting, H2O AutoML, XGBoost, MKR, KGAT, and RippleNet.

In general, the AUC metrics shown in this chapter compare the scores within an environment, SAS, R, or Python, and an input variable set selection, two methods, three, four, or five.

AUC Comparison

During this study, 441 hyper-parameter combinations with ten different seeds in each combination were trained among all methods. More in detail, Table 23 shows the number of combinations performed for each algorithm.

Algorithm	Combinations
MKR	64 in Python
KGAT	112 in Python
RippleNet	44 in Python
Logistic Regressions	10 (3 in SAS, 4 in R, and 3 Target Encoded in R)
Neural Networks	129 (54 in SAS, 72 in R, and 3 Target Encoded in R)
Decision Trees	4 in R
Bagging	12 (4 in SAS, 4 in R, and 4 Target Encoded in R)
Random Forest	20 (8 in SAS, 8 in R, and 4 Target Encoded in R)
Gradient Boosting	42 (16 in SAS, 12 in R, 10 from AutoML, and 4 Target Encoded)
XGBoost	4 in R

Table 23 Number of combinations for each algorithm

The following table shows those models and hyper-parameters setup that best performed in terms of mean AUC for their ten seed in each algorithm to summarize all these combinations. We can see that four algorithms achieved a mean AUC over 0.930 in their best setup: Recbole RippleNet, R Caret *avNNet*, R H20 *GBM*, and SAS *Treebost* proc.

Algorithm	Best Model	Sampling Method	Mean AUC	Max AUC	Min AUC
MKR	Learning Rate 10^{-3}, one lower layer, one higher layer, with the inner product as prediction function, Regression Weight of 10^{-6}, and Dropout of 5 10^{-2}	Ten seeds resampling without cross-validation, Random Ordering, Ratio Based 80%-10%-10%	0.929	0.934	0.924
KGAT	Learning Rate 10^{-3}, 64 dimensions for embedding, attention and prediction layers, Regression Weight of 10^{-5}, and Dropout of $1 \cdot 10^{-2}$	Ten seeds resampling without cross-validation, Random Ordering, Ratio Based 80%-10%-10%	0.801	0.760	0.841
RippleNet	Learning Rate 10^{-3}, 64 dimensions for the embedding layer, Negative samples of two, Memory of four, Regression Weight of 10^{-7}, and Knowledge Graph Weight of 10^{-2}	Ten seeds resampling without cross-validation, Random Ordering, Ratio Based 80%-10%-10%	0.930	0.935	0.922
Logistic Regressions	R Caret GLM with 22 input variables	Ten seed resampling with four groups cross-validation, Random	0.804	0.810	0.795

		Ordering, Ratio Based 80%-10%-10%			
Neural Networks	R Caret avNNet, ten nodes in the hidden layer, Weight Decay of 10^{-1}, and 50 input variables	Ten seed resampling with four groups cross-validation, Random Ordering, Ratio Based 80%-10%-10%	0.930	0.932	0.929
Decision Trees	R Caret RPart with 50 input variables with CP = 0	Ten seed resampling with four groups cross-validation, Random Ordering, Ratio Based 80%-10%-10%	0.925	0.926	0.922
Bagging	R Caret Random Forest with 50 maximum split variables, 50 input variables, and 50 maximum trees	Ten seed resampling with four groups cross-validation, Random Ordering, Ratio Based 80%-10%-10%	0.884	0.886	0.882
Random Forest	SAS HPForest proc with 12 maximum variables split variables, 200 trees, and 22 input variables	Ten seed resampling with four groups cross-validation, Random Ordering, Ratio Based 80%-10%-10%	0.910	0.911	0.908
Treeboost	SAS Treeboost proc with a maximum depth of six levels, shrinkage of 10^{-1}, minimum observations for splitting, minimum category size and minimum leaf size of 74, a maximum number branch of two, 100 iterations, and 50 input variables	Ten seed resampling with four groups cross-validation, Random Ordering, Ratio Based 80%-10%-10%	0.930	0.935	0.922
H2O GBM	H2O GBM method with 61 trees, maximum depth of 17 and 0.1 for learning rate which best performed in the AutoML execution	Ten seed resampling with four groups cross-validation, Random Ordering, Ratio Based 80%-10%-10%	0.947	0.949	0.943
XGBoost	R Caret xgbTree with Eta parameter of 10^{-3}, a maximum depth of six, minimum child weight of 74, and 50 input variables	Ten seed resampling with four groups cross-validation, Random Ordering, Ratio Based 80%-10%-10%	0.808	0.809	0.807

Table 24 Best model AUC result for each algorithm

Moreover, enclosed in the annexes, we can find the illustrations for the best models in terms of AUC for each environment and input variable setup. Even so, to find deeper insights with all the AUC results, not only the best ones, please refer to annexes 2.1 and 2.2 below.

In the case of Graph Neural Network models, the performance is quite similar for MKR, and RippleNet algorithms, achieving both results over 0.92 mean AUC. Not for KGAT, which barely beats the 0.80 barriers.

Comparing the overall mean AUC results and model stability, the best option is the Gradient Boosting Machine computed by the *H2O* package in R with a mean AUC of 0.9475, over the 0.9308 mean AUC of *avNNet*, 0.9306 of RippleNet, and 0.9302 of SAS *Treeboost* Gradient Boosting.

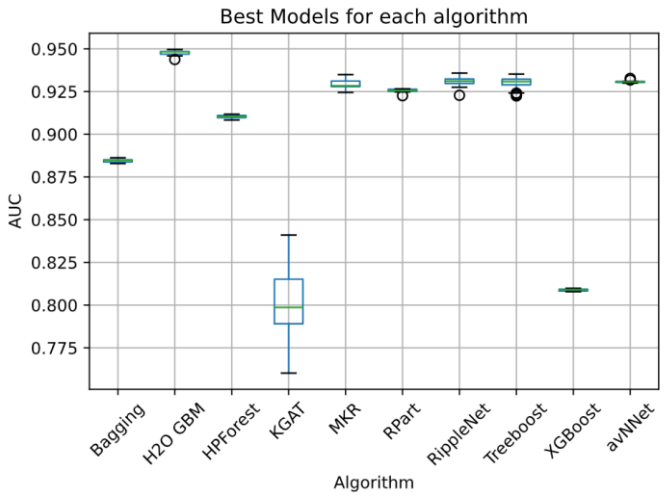


Figure 20 Best models for each algorithm

Still, AUC comparison is not the only indicator to think about when deciding on which is the best model.

Categorical Variables Transformation for Best Models

Earlier in this study, a Target Encoding transformation was proposed to enhance the model performances when converting those categorical variables with high cardinality. In order to assess the contribution of the Target Encoding to the AUC results, this table reviews how this transformation affected the algorithms accepting fifty input variables.

Algorithm	Best Mean AUC with Categorical Variables	Best Mean AUC with Encoded Variables
Logistic Regressions	0.804	0.764
Neural Networks	0.930	0.859
Bagging	0.884	0.739
Random Forest	0.910	0.749
Gradient Boosting	0.947	0.831

Table 25 Categorical Variables versus Transformed Variables for models trained with 50 variables

In the literature, there are specific occasions when Target Encoding transformation helped make progress in the error rate and log loss results, but it is not the case. Compared in terms of AUC, the transformation did not help to improve the marks achieved by algorithms before the transformation.

Complexity Differences

Throughout the best models in terms of AUC, their complexity is irregular, although high in any case. We arrive at the finish line with four algorithms: Gradient Boosting Machine of *H20* package in R, Neural Networks R Caret *avNNet*, Gradient Boosting SAS *Treeboost* proc, and RippleNet RecBole implementation.

Let us compute the number of model parameters for each selected algorithm to give a glance at complexity. For ten Neural Network, the number of parameters can be computed from its formula $k(h+1)+h+1$, k for the number of input variables, 274, and h for the number of nodes in the hidden layer, 10. Obtaining 3003 parameters in this case.

The number of parameters used by the Gradient Boosting Machine can be expressed as a formula compounding the maximum number of leaves and the number of trees. If we declare a minimum leaf size, it is a way to set the maximum number of leaves, given that the training dataset has a limited number of samples. Equation (18) uses a multiplier of two, considering that each split has two parameters: split variable and split threshold. Then it computes the maximum number of parameters by counting the theoretical number of splits.

$$N_{Gradient\ Boosting} = 2 \cdot trees \cdot (max.leaves - 1) = 2 \cdot trees \cdot \left(\frac{Train\ Sample\ size}{min.leaf\ size} - 1\right)$$
$$N_{Gradient\ Boosting} = \begin{cases} 1.59 \cdot 10^5, & SAS\ Treebost\ with\ 100\ trees\ and\ min.leaf\ size\ 74 \\ 3.33 \cdot 10^5, & H20\ GBM\ with\ 61\ trees\ and\ 2736\ max.leaves \end{cases} \quad (18)$$

When it comes to RippleNet, to compute the theoretic number of parameters, we should look at the size of embeddings calculated for each node and relationship in the graph. The best RippleNet model in this study uses 64 dimensions for each node and relationship embeddings. Multiplying the dimensions of a node embedding by the number of nodes, 50000 users, plus the number of articles, 42417 articles, the number of entities, 23520 entities, and the number of relations, 873 relations, we obtain a total number of $4.33 \cdot 10^6$ model parameters.

$$N = dimensions(users + articles + entities + relations) = 7.47 \cdot 10^6 \quad (19)$$

RippletNet is the most complex model, 22 times more complex than H2O GBM, 47 times more complex than Treesbost, and 2489 times more complex than the Neural Network used in this study.

Model selected

When comparing the four methods that perform best in terms of AUC, we see that avNNet with ten nodes and 0.1 Weight Decay Neural Network implementation of Caret has the lowest AUC variance and the least complex model with 3003 parameters.

But, the big difference in terms of mean AUC tips the scales towards the H2O Gradient Boosting Machine with 61 trees, 17 levels, and 2736 maximum leaves. This model is the one to choose when deciding which articles to send to a reader in the Microsoft News Dataset.

Algorithm	Sampling Method	Mean AUC	AUC Variance	Complexity
RippleNet	Repeated Validation with 10 different seeds and 80%-20% train-test split	0.9306	$1.28 \cdot 10^{-5}$	$7.47 \cdot 10^6$ parameters
avNNet	Repeated Cross-Validation with 10 different seeds, 4 validation groups, and 80%-20% train-test split	0.9308	$6.76 \cdot 10^{-7}$	$3 \cdot 10^3$ parameters
Treeboost	Repeated Cross-Validation with 10 different seeds, 4 validation groups, and 80%-20% train-test split	0.9302	$8.56 \cdot 10^{-6}$	$1.59 \cdot 10^5$ parameters
H2O GBM	Repeated Cross-Validation with 10 different seeds, 4 validation groups, and 80%-20% train-test split	0.9475	$2.98 \cdot 10^{-6}$	$3.33 \cdot 10^5$ parameters

Table 26 Three best models' comparison and best model selected

Conclusions

After developing this study, one concludes that we must take care of the complexity of the Graph Neural Networks models because it reaches much higher levels than its counterparts for tabular models.

Also, this study needed higher computational power, which could be fulfilled thanks to the GPU special equipment, reducing the training time of KGAT, MKR, and RippleNet from weeks to hours. Without this crucial piece, the Graph Neural Network models could not have taken part in this research.

Building the tabular dataset from graph-structured data may be tedious and prone to errors, but it is worth saying that it is practical when using tabular algorithms, and for this particular dataset, it worked better than expected.

Ending with the most significant conclusion on this work: Graph Neural Networks may perform well when dealing with graph-structured information, but in this case, it is not enough compared with tabular methods. Tabular methods are less complex, more stables, need less computational power, need less training data, and, most important, perform better under AUC criteria. In short, tabular methods are better for this dataset.

Performance: Graph vs. Tabular Methods

In terms of performance, tabular methods are better overall. The Top 2 algorithms in performance are based on tabular structured data. The first graph algorithm is in the third position. However, it is more impressive to see algorithms like Logistic Regression outperforming KGAT, 10^5 times more complex than the Logistic Regression.

Algorithm	Mean AUC
H2O GBM	0.947
Neural Networks	0.930
RippleNet	0.930
Treeboost	0.930
MKR	0.929
Decision Trees	0.925
Random Forest	0.910
Bagging	0.884
XGBoost	0.808
Logistic Regressions	0.804
KGAT	0.801

Table 27 Ranking of Algorithms' Performance by Mean AUC. In yellow, Graph Methods

When observing the variance of the best models for each algorithm, the difference between tabular and graph models is clear: all tabular models are more stable than graph-based ones. The most stable algorithm, XGBoost, is 1000 times more stable than the least stable algorithm, KGAT.

Algorithm	AUC Variance
XGBoost	$4.51 \cdot 10^{-7}$
Neural Networks	$6.76 \cdot 10^{-7}$
Random Forest	$6.88 \cdot 10^{-7}$
Bagging	$9.72 \cdot 10^{-7}$
Decision Trees	$1.36 \cdot 10^{-6}$

H2O GBM	$2.98 \cdot 10^{-6}$
Treeboost	$8.56 \cdot 10^{-6}$
Logistic Regressions	$9.75 \cdot 10^{-6}$
MKR	$1.06 \cdot 10^{-5}$
RippleNet	$1.28 \cdot 10^{-5}$
KGAT	$4.52 \cdot 10^{-4}$

Table 28 Inverse Ranking of Algorithms' Performance by AUC Variance. In yellow, Graph Methods

Complexity: Graph vs. Tabular Methods

Previous chapters showed how graph methods are tens of times more complex than ensembled tree-based methods and thousands of times more complex than one hidden layer neural network given the dataset enclosed here.

However, we have to consider that the data used in graph methods was 121 times bigger, implying more users, more articles, and more entities, than the 73985 interactions dataset used for tabular methods. Now the question is, how does the size of the dataset impacts the complexity of the models.

The following table shows how the complexity of graph methods scales 68 times faster than the complexity of the single hidden layer Neural Network as the number of users increases. It means that even using the same sample size in tabular methods, the number of parameters for graph methods would continue to be much bigger than tabular methods.

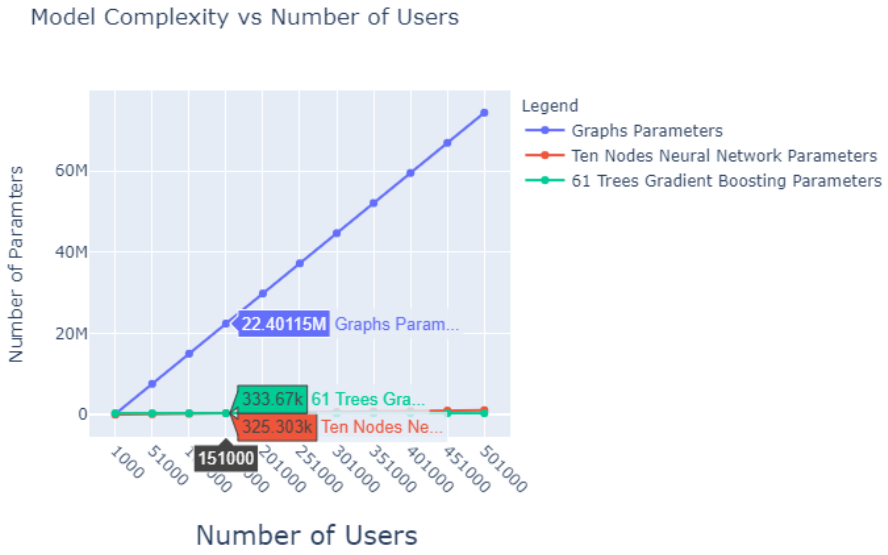


Figure 21 Complexity of Graph Methods vs. Neural Networks vs. Gradient Boosting by the number of users

On the other hand, Figure 22 shows a breakpoint where the number of parameters for Neural Networks would overtake the number of parameters for the used Gradient Boosting setup. When the number of users exceeds 151000 as the input to the model trainer, the number of parameters for ten nodes Neural Networks will keep rising higher than the number of parameters of the Gradient Boosting algorithm. The reason is that the number of the Gradient Boosting Machine parameters is not related to the number of training sample size but the number of trees and maximum leaves.

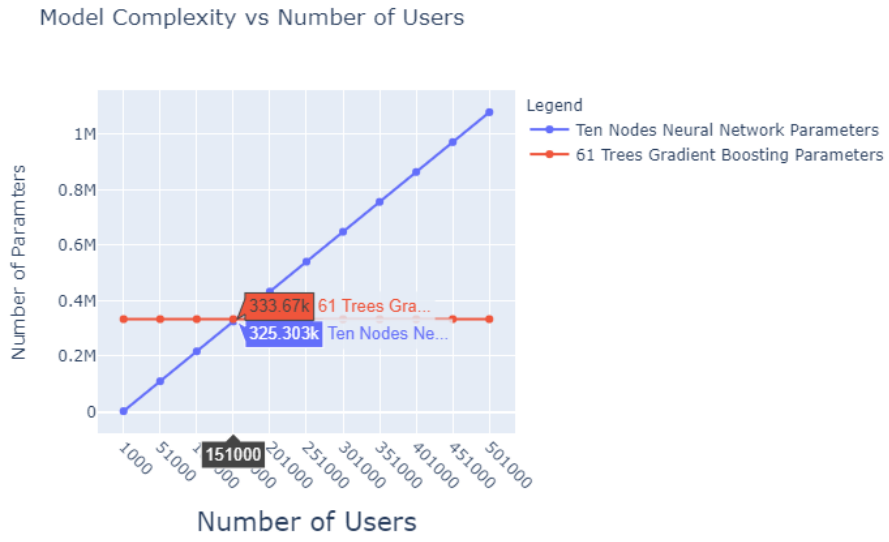


Figure 22 Complexity of Neural Networks vs. Gradient Boosting by the number of users

Graph Algorithms: what impacts their performance?

Figure 23, Figure 24, and Figure 25 refer to MKR, KGAT, and RippleNet Area Under the Curve (AUC) results. In each scenario, ten (10) epochs were repeated with different seeds, and the results of those ten AUC metrics conform to each bar in the Boxplot chart below.

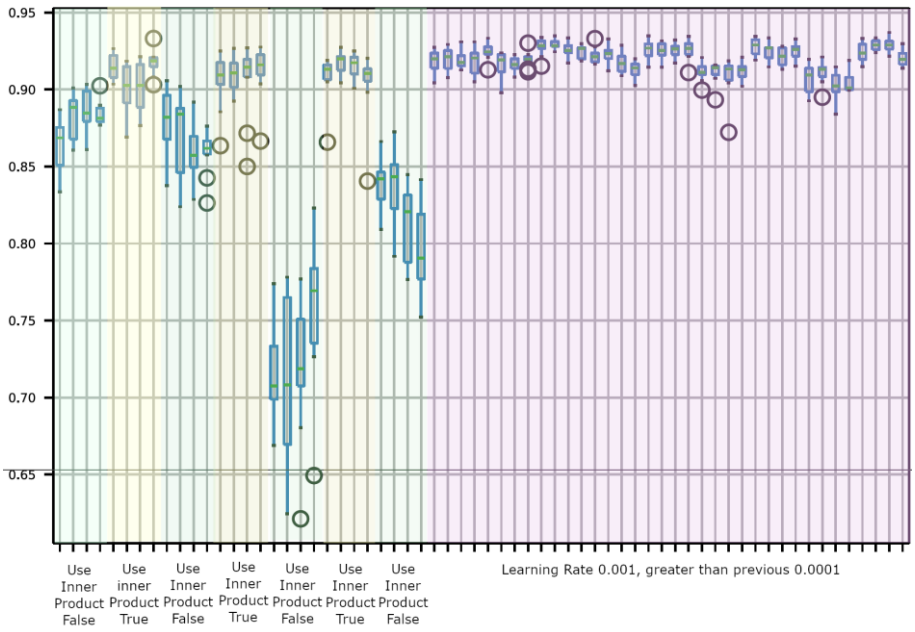


Figure 23 MKR complete AUC Boxplot grid results divided into eight regions

The charts have been shadowed with three different colors, dividing the regions where the model performs differently regarding the hyper-parameters setup. In Figure 23, MKR performance better in terms of AUC when the inner product is used as a prediction function and the learning rate is 0.0001, compared to the case where this inner function is not used, and the same learning rate is used. Overall, the best performance is achieved for learning rate 0.001, again with some differences between the prediction function.

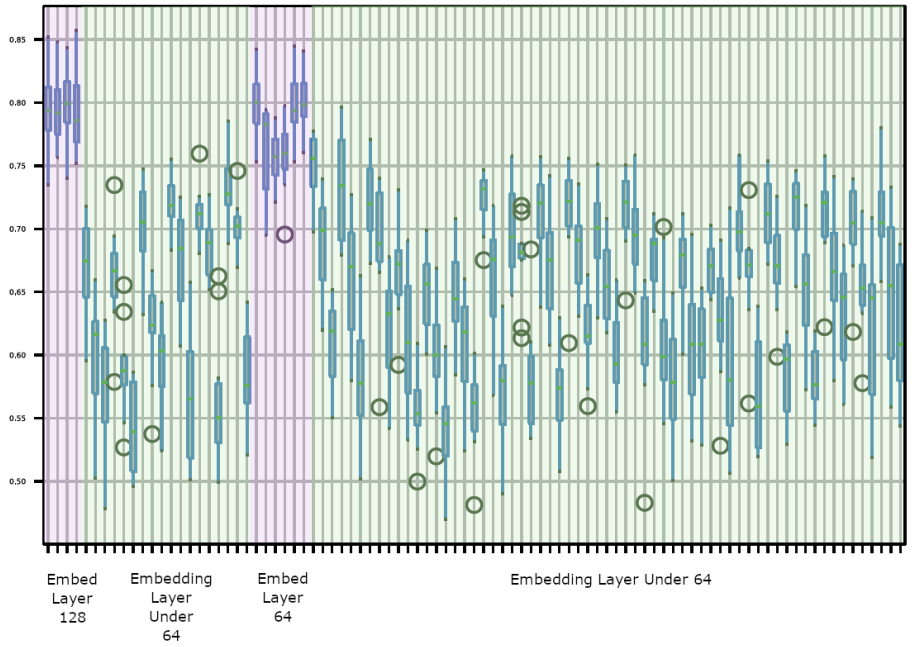


Figure 24 KGAT complete AUC Boxplot grid results divided into four training regions

When talking about KGAt, what impacts performance is the number of dimensions used for describing the embedding layer. Figure 24 shows that the best AUC results are achieved when setting the embedding layer hyper-parameter over or equal to 64: 64 and 128. When lower, 2, 4, 8, the performance decreases significantly.

In RippleNet, the differences come from the learning rate hyper-parameter. The best window is for a learning rate under 0.01 and over 0.0005. The worst configuration happens when it is under 0.0005.

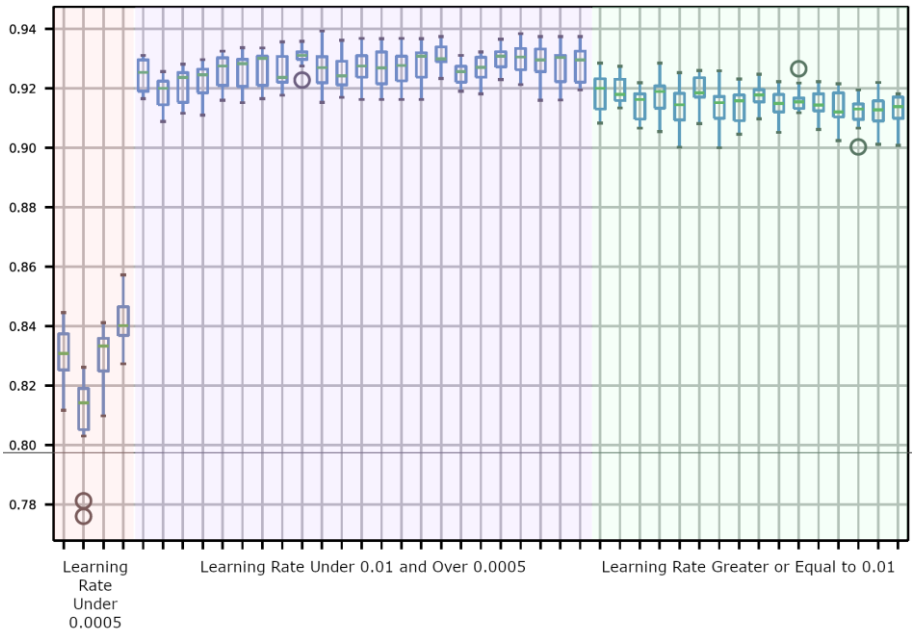


Figure 25 RippleNet complete AUC Boxplot grid results divided into three regions

Regarding the results per environment, we can see a notorious difference in tabular AUC best results when reducing the number of input variables. In SAS Environment, the metric for the best models drops from 0.930 mean AUC marked by the Gradient Boosting Model with fifty input variables to 0.785 mean AUC for the same algorithm with three input variables.

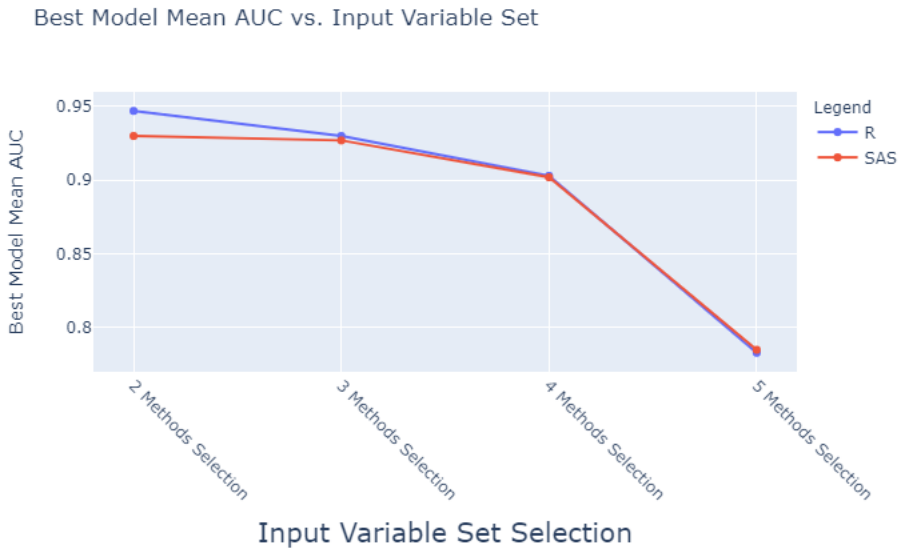


Figure 26 Best Model Mean AUC by Input Variable Set and Software

R shows an identical behavior, decreasing the AUC performance when reducing the number of input variables. The best AUC mean metric of 0.947 in R is achieved by the Gradient Boosting Machine with a maximum depth of 17 levels and 61 trees when fifty input variables fed the training. More information about AUC performance could be obtained in annex 2.2 below.

Future Work and Open Research Lines

This work leaves open concrete questions about the models that could not be trained because of the resources available. How would it have performed the Neural Networks models in SAS Base for a more extensive variable set? How would the variance and bias behave in Graph Neural Networks when using cross-validation? What about the ensemble? How would an ensemble model have performed?

It also opens to door to keep increasing the number of nodes in the hidden layer for Neural Networks. The trend of the AUC metric from two nodes did not stop increasing when it arrived at ten nodes. In future works, this hyper-parameter might be increased.

More Graph Neural Network algorithms could also be compared with the ones selected in this study: Collaborative Knowledge Base Embedding, Heterogeneous Knowledge Base Embeddings for Explainable Recommendation, or Knowledge Graph Convolutional Networks for Recommender.

Next Steps

To tackle the questions raised, this study proposes the following work plan in five steps for future works:

1. Integrate Cross-Validation in RecBole Graph Methods
2. Increase the sample size and variable set used for Tabular Methods.
3. Take advantage of Ensemble Methods to join the results of Neural Networks, Gradient Boosting, and RippleNet achieving better performance
4. Implement bigger and deeper Neural Networks beyond the ten nodes single layer multi-perceptron.
5. Add more comparison metric as Logarithmic Loss or Miss-Classification Rate to select the best model, comparing not only complexity or AUC.

Bibliography

Doree, J. (2007). RSS: A Brief Introduction. *The Journal of Manual & Manipulative Therapy*, *15*(1), 57–58. PubMed. https://doi.org/10.1179/106698107791090169

Ford, J. B. (2018). What Do We Know About Celebrity Endorsement in Advertising? *Journal of Advertising Research*, *58*(1), 1–2. https://doi.org/10.2501/JAR-2018-006

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. https://www.deeplearningbook.org

Gulla, J. A., Zhang, L., Liu, P., Özgöbek, Ö., & Su, X. (2017). The Adressa Dataset for News Recommendation. *Proceedings of the International Conference on Web Intelligence*, 1042–1048. https://doi.org/10.1145/3106426.3109436

Hamilton, W. L. (2020). Graph Representation Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *14*(3), 1–159. https://doi.org/10.2200/S01045ED1V01Y202009AIM046

Hwang, K., & Zhang, Q. (2018). Influence of parasocial relationship between digital celebrities and their followers on followers' purchase and electronic word-of-mouth intentions, and persuasion knowledge. *Computers in Human Behavior*, *87*, 155–173. https://doi.org/10.1016/j.chb.2018.05.029

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning* (Vol. 103). Springer New York. https://doi.org/10.1007/978-1-4614-7138-7

Okura, S., Tagami, Y., Ono, S., & Tajima, A. (2017). Embedding-Based News Recommendation for Millions of Users. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1933–1942. https://doi.org/10.1145/3097983.3098108

O'Sullivan, D., & McCallig, J. (2012). Customer satisfaction, earnings and firm value. *European Journal of Marketing*, *46*(6), 827–843. https://doi.org/10.1108/03090561211214627

Portela García-Miguel, J. (2019). *Machine Learning. Introducción*. UCM.

Wang, H., Zhang, F., Wang, J., Zhao, M., Li, W., Xie, X., & Guo, M. (2018). RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 417–426. https://doi.org/10.1145/3269206.3271739

Wang, H., Zhang, F., Zhao, M., Li, W., Xie, X., & Guo, M. (2019). Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. *The World Wide Web Conference on - WWW '19*, 2000–2010. https://doi.org/10.1145/3308558.3313411

Wang, X., He, X., Cao, Y., Liu, M., & Chua, T.-S. (2019). KGAT: Knowledge Graph Attention Network for Recommendation. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 950–958. https://doi.org/10.1145/3292500.3330989

Wu, F., Qiao, Y., Chen, J.-H., Wu, C., Qi, T., Lian, J., Liu, D., Xie, X., Gao, J., Wu, W., & Zhou, M. (2020, July). MIND: A Large-scale Dataset for News Recommendation. *ACL 2020*. https://www.microsoft.com/en-us/research/publication/mind-a-large-scale-dataset-for-news-recommendation/

Yu, S., & Kak, S. (n.d.). *An Empirical Study of How Users Adopt Famous Entities*. 7.

Annexes

Annex 1: Code

Annex 1.1: WikiData API entity information fetch script

```
import sys
import os
import json
import re
import urllib.request, json

base_url = 'http://www.wikidata.org/entity/'
news = open('news-json.tsv', 'r',encoding='utf-8')
lines = news.readlines()
kg = open('kg.tsv', 'a', encoding='utf-8')
item entity = open('item_entity.tsv', 'a', encoding='utf-8')

for line in lines:
    try:
        print("New line...")
        line tokens = re.split(r'\t+', line.rstrip('\t'))
        new_id = line_tokens[0]
        title_entities = json.loads(line_tokens[1])
        body_entities = json.loads(line_tokens[2]) if len(line_tokens) == 3 else []
        for entity in title_entities + body_entities:
            print(f'{new_id}\t{entity["WikidataId"]}\n')
            item_entity.write(f'{new_id}\t{entity["WikidataId"]}\n')
            url = f'{base_url}{entity["WikidataId"]}'
            response = urllib.request.urlopen(url)
            data = json.loads(response.read())
            for property_key in data['entities'][entity['WikidataId']]['claims']:
                claims = data['entities'][entity['WikidataId']]['claims'][property_key]
                for claim in claims:
                    if 'type' in claim and claim['type'] == 'statement' \
                        and 'mainsnak' in claim and 'datatype' in claim['mainsnak'] \
                        and claim['mainsnak']['datatype'] == 'wikibase-item' \
                        and 'datavalue' in claim['mainsnak'] and 'type' in
claim['mainsnak']['datavalue'] \
                        and claim['mainsnak']['datavalue']['type'] == 'wikibase-
entityid':

print(f'{entity["WikidataId"]}\t{claim["mainsnak"]["property"]}\t{claim["mainsnak"]["dat
avalue"]["value"]["id"]}\n')

kg.write(f'{entity["WikidataId"]}\t{claim["mainsnak"]["property"]}\t{claim["mainsnak"]["
datavalue"]["value"]["id"]}\n')
    except Exception as e:
        continue

kg.close()
item_entity.close()
```

Annex 1.2: MIND behavior.tsv transformation into RecBole format

```
behaviors =
pd.read_csv(f'..{path.sep}data{path.sep}MINDsmall_dev{path.sep}behaviors.tsv', sep='\t',
header=None, names=[
    'id', 'user', 'date', 'history', 'printlog'
])

links = []
for _, event in behaviors.iterrows():
    history = str(event['history']).split()
    for new_clicked in history:
        links.append([event['user'], new_clicked, 1])
    printlog = event['printlog'].split()
    for new_printed in printlog:
        new_id, clicked = new_printed.split(sep='-')
        links.append([event['user'], new_id, clicked])
```

```
with
open(f'..{path.sep}data{path.sep}mind_small_positive{path.sep}mind_small_positive.inter'
, 'w', newline='') as f:
    writer = csv.writer(f, delimiter='\t')
    writer.writerow(['user_id:token', 'item_id:token', 'label:float'])
    writer.writerows(links)
```

Annex 1.3: MKR training code

```
import os
import glob
import json
import datetime
import threading
from logging import getLogger
from recbole.config import Config
from recbole.data import create_dataset, data_preparation
from recbole.utils import init_logger, get_model, get_trainer, init_seed

saved = True

n_threads = 3

def run_iteration(config_dict: dict):
    file_name_prefix = f"results{os.path.sep}MKR{os.path.sep}case-{config_dict['case']}"
    if len(glob.glob(f'{file_name_prefix}-*.txt')) > 0:
        return

    config = Config(config_file_list=[
        f'models{os.path.sep}MKR{os.path.sep}config.yaml'
    ], config_dict=config_dict)
    init_seed(config['seed'], config['reproducibility'])
    init_logger(config)
    logger = getLogger()
    dataset = create_dataset(config)

    train_data, valid_data, test_data = data_preparation(config, dataset)
    model = get_model(config['model'])(config, train_data).to(config['device'])
    trainer = get_trainer(config['MODEL_TYPE'], config['model'])(config, model)

    _, best_valid_result = trainer.fit(
        train_data, valid_data, saved=saved, show_progress=config['show_progress']
    )

    test_result = trainer.evaluate(
        test_data, load_best_model=saved, show_progress=config['show_progress'])

    iteration_info = f'Config: {config_dict}\r\nBest valid result:
{best_valid_result}\r\nTest result: {test_result}'
    logger.info(iteration_info)

    result_file = open(f"{file_name_prefix}-
{datetime.datetime.now().strftime('%Y%m%d%H%M%S%f')}.txt", 'w')
    result_file.write(f'Config: {config_dict}\r\n')
    result_file.write(f'Best valid result: {best_valid_result}\r\n')
    result_file.write(f'Test result: {test_result}\r\n')
    result_file.close()

if __name__ == "__main__":
    cases = []
    with open(f'models{os.path.sep}MKR{os.path.sep}train.test', 'r') as f:
        for line in f:
            try:
                cases.append(json.loads(line))
            except Exception as e:
                print(e)
                print(f'Failed to load line from test case file: {line}')
    for i in range(0, len(cases), n_threads):
        configs = cases[i:i + n_threads]
        threads = list()
        for config_dict in configs:
            x = threading.Thread(target=run_iteration, args=(config_dict,))
            threads.append(x)
            x.start()
```

```
        for _, thread in enumerate(threads):
            thread.join()
```

Annex 1.4: KGAT training code

```
import os
import json
import datetime
import threading
from logging import getLogger
from recbole.config import Config
from recbole.data import create_dataset, data_preparation
from recbole.utils import init_logger, get_model, get_trainer, init_seed

saved = True

n_threads = 1

def run_iteration(config_dict: dict):
    config = Config(config_file_list=[
        f'models{os.path.sep}KGAT{os.path.sep}config.yaml'
    ], config_dict=config_dict)
    init_seed(config['seed'], config['reproducibility'])
    init_logger(config)
    logger = getLogger()
    dataset = create_dataset(config)

    train_data, valid_data, test_data = data_preparation(config, dataset)
    model = get_model(config['model'])(config, train_data).to(config['device'])
    trainer = get_trainer(config['MODEL_TYPE'], config['model'])(config, model)

    _, best_valid_result = trainer.fit(
        train_data, valid_data, saved=saved, show_progress=config['show_progress']
    )

    test_result = trainer.evaluate(
        test_data, load_best_model=saved, show_progress=config['show_progress'])

    iteration_info = f'Config: {config_dict}\r\nBest valid result:
{best_valid_result}\r\nTest result: {test_result}'
    logger.info(iteration_info)

    result_file = open(f"results{os.path.sep}KGAT{os.path.sep}case-
{config_dict['case']}-{datetime.datetime.now().strftime('%Y%m%d%H%M%S%f')}.txt", 'w')
    result_file.write(f'Config: {config_dict}\r\n')
    result_file.write(f'Best valid result: {best_valid_result}\r\n')
    result_file.write(f'Test result: {test_result}\r\n')
    result_file.close()

if __name__ == "__main__":
    cases = []
    with open(f'models{os.path.sep}KGAT{os.path.sep}train-IV.test', 'r') as f:
        for line in f:
            try:
                cases.append(json.loads(line))
            except Exception as e:
                print(e)
                print(f'Failed to load line from test case file: {line}')
    for i in range(0, len(cases), n_threads):
        configs = cases[i:i + n_threads]
        threads = list()
        for config_dict in configs:
            x = threading.Thread(target=run_iteration, args=(config_dict,))
            threads.append(x)
            x.start()

        for _, thread in enumerate(threads):
            thread.join()
```

Annex 1.5: RippleNet training code

```
import os
import glob
import json
import datetime
import threading
from logging import getLogger
from recbole.config import Config
from recbole.data import create_dataset, data_preparation
from recbole.utils import init_logger, get_model, get_trainer, init_seed

saved = True

n_threads = 1

def run_iteration(config_dict: dict):
    file_name_prefix = f"results{os.path.sep}RippleNet{os.path.sep}case-
{config_dict['case']}"
    if len(glob.glob(f'{file_name_prefix}-*.txt')) > 0:
        return

    config = Config(config_file_list=[
        f'models{os.path.sep}RippleNet{os.path.sep}config.yaml'
    ], config_dict=config_dict)
    init_seed(config['seed'], config['reproducibility'])
    init_logger(config)
    logger = getLogger()
    dataset = create_dataset(config)

    train_data, valid_data, test_data = data_preparation(config, dataset)
    model = get_model(config['model'])(config, train_data).to(config['device'])
    trainer = get_trainer(config['MODEL_TYPE'], config['model'])(config, model)

    _, best_valid_result = trainer.fit(
        train_data, valid_data, saved=saved, show_progress=config['show_progress']
    )

    test_result = trainer.evaluate(
        test_data, load_best_model=saved, show_progress=config['show_progress'])

    iteration_info = f'Config: {config_dict}\r\nBest valid result:
{best_valid_result}\r\nTest result: {test_result}'
    logger.info(iteration_info)

    result_file = open(f"{file_name_prefix}-
{datetime.datetime.now().strftime('%Y%m%d%H%M%S%f')}.txt", 'w')
    result_file.write(f'Config: {config_dict}\r\n')
    result_file.write(f'Best valid result: {best_valid_result}\r\n')
    result_file.write(f'Test result: {test_result}\r\n')
    result_file.close()

if __name__ == "__main__":
    cases = []
    with open(f'models{os.path.sep}RippleNet{os.path.sep}train-II.test', 'r') as f:
        for line in f:
            try:
                cases.append(json.loads(line))
            except Exception as e:
                print(e)
                print(f'Failed to load line from test case file: {line}')
    for i in range(0, len(cases), n_threads):
        configs = cases[i:i + n_threads]
        threads = list()
        for config_dict in configs:
            x = threading.Thread(target=run_iteration, args=(config_dict,))
            threads.append(x)
            x.start()

        for _, thread in enumerate(threads):
            thread.join()
```

Annex 1.6: Early Stopping calculation macro

```
%macro
redneuronalbinaria(archivo=,listclass=,vardep=,porcen=,semilla=,ocultos=,meto=levmar,act
i=);
	proc dmdb data=&archivo dmdbcat=catauno;
		target &vardep;
		class &vardep &listclass;
	run;

	data ooo;
		set &archivo;
	run;

	data datos;
		set ooo nobs=nume;
		tr=int(&porcen*nume);
		call symput('tr',left(tr));
		u=ranuni(&semilla);
	run;

	proc sort data=datos;
		by u;
	run;

	data datos valida;
		set datos;if _n_>tr then output valida;
		else output datos;
	run;

	proc neural data=datos dmdbcat=catauno validata=valida graph;
		input &listclass /level=nominal;
		target &vardep /level=nominal id=o error=ENT;
		hidden &ocultos /id=h act=&acti;
		nloptions maxiter=10000;
		netoptions randist=normal ranscale=0.1 random=15115;
		prelim 0;
		train maxiter=10000 outest=mlpest estiter=1 technique=&meto;
		score data=datos out=mlpout outfit=mlpfit;
		score data=valida out=mlpout2 outfit=mlpfit2 role=valid;
	run;

	data mlpest2 ;
		k=3;
		retain iterepocas 0;
		set mlpest nobs=nume;
		call symput('numeroit',left(nume));
		eval=_VOBJERR_;
		x3=lag3(eval);
		x6=lag6(eval);
		if _n_>6 and eval>x3 and eval>x6 then iterepocas=_n_;
	run;

	data;
		set mlpest2 nobs=nume;
		if iterepocas ne 0 then do;
			call symput('earlystop',left(iterepocas));
			stop;
		end;
		else if _n_=nume and iterepocas=0 then do;
			iterepocas=&numeroit;
			call symput('earlystop',left(iterepocas));
			stop;
		end;
	run;

	data fin;
		j=&earlystop;
		set mlpest point=j;
		output;
		stop;
	run;

	data mlpest;
		set mlpest nobs=nume;
		if _n_=&earlystop then do;
			cosa1=put(_OBJERR_,20.6);
			cosa2=put(_VOBJERR_,20.6);
		end;
```

```
		else do;cosa1=' ';cosa2=' ';end;
	run;

	title1
	h=2 box=1 j=c c=red 'TRAIN' c=blue '  VALIDA'
	h=1.5 j=c c=black "EARLY STOPPING=&earlystop " "semilla=&semilla"
	h=1 j=c c=green "NODOS OCULTOS: &ocultos  " " METODO: &meto "  "ACTIVACIÓN:
&acti"
	h=1 j=c c=black "EL ERROR ES EL VALOR DE LA ENTROPÍA";

	symbol1 c=red v=circle i=join pointlabel=("#cosa1" h=1 c=red position=bottom
j=c);
	symbol2 c=blue v=circle i=join pointlabel=("#cosa2" h=1 c=blue position=top j=c);

	axis1 label=none;
	proc gplot data=mlpest;
		plot _OBJERR_ *_iter_=1 _VOBJERR_*_iter_=2 /overlay href=&earlystop
vaxis=axis1 haxis=axis1;
	run;

	proc print data=fin;
		var  _iter_ _OBJERR_  _AVERR_  _VNOBJ_   _VOBJ_  _VOBJERR_  _VAVERR_;
	run;
%mend;
```

Annex 1.7: Binary pack in SAS Base

```
%macro
cruzadalogistica(archivo=,vardepen=,categor=,ngrupos=,sinicio=,sfinal=,objetivo=tasafall
os);
	title ' ';

	data final;
	run;

	data sal_final;
	run;

	%do semilla=&sinicio %to &sfinal;
		data dos;
			set &archivo;
			u=ranuni(&semilla);
		run;

		proc sort data=dos;
			by u;
		run;

		data dos (drop=nume);
			retain grupo 1;
			set dos nobs=nume;
			if _n_>grupo*nume/&ngrupos then grupo=grupo+1;
		run;

		data fantasma;
		run;

		%do exclu=1 %to &ngrupos;
			data tres;
				set dos;
				if grupo ne &exclu then vardep=&vardepen;
			run;

			proc logistic data=tres noprint;
				class &categor;
				model vardep=&categor;
				output out=sal p=predi;
			run;

			data sal_final;
				set sal_final sal;
			run;

			data sal2;
				set sal;pro=1-predi;
```

```
                        if pro>0.5 then pre11=1;
                        else pre11=0;
                        if grupo=&exclu then output;
                run;

                proc freq data=sal2;
                        tables pre11*&vardepen/out=sal3;
                run;

                data estadisticos (drop=count percent pre11 &vardepen);
                        retain vp vn fp fn suma 0;
                        set sal3 nobs=nume;
                        suma=suma+count;
                        if pre11=0 and &vardepen=0 then vn=count;
                        if pre11=0 and &vardepen=1 then fn=count;
                        if pre11=1 and &vardepen=0 then fp=count;
                        if pre11=1 and &vardepen=1 then vp=count;
                        if _n_=nume then do;
                        porcenVN=vn/suma;
                        porcenFN=FN/suma;
                        porcenVP=VP/suma;
                        porcenFP=FP/suma;
                        sensi=vp/(vp+fn);
                        especif=vn/(vn+fp);
                        tasafallos=1-(vp+vn)/suma;
                        tasaciertos=1-tasafallos;
                        precision=vp/(vp+fp);
                        F_M=2*Sensi*Precision/(Sensi+Precision);
                        output;
                        end;
                run;

                data fantasma;
                        set fantasma estadisticos;
                run;
        %end;

        proc means data=fantasma sum noprint;
                var &objetivo;
                output out=sumaresi sum=suma mean=media;
        run;

        data sumaresi;
                set sumaresi;
                semilla=&semilla;
        run;

        data final (keep=suma media semilla);
                set final sumaresi;
                if suma=. then delete;
        run;
%end;

proc print data=final;
run;
%mend;


%macro cruzadabinarianeural(archivo=, vardepen=, categor=, ngrupos=, sinicio=, sfinal=,
nodos=, algo=, objetivo=, early=, acti=tanh, directorio=);
        title ' ';
        data final;
        run;

        data sal_final;
        run;

        proc printto print="&directorio\basura.txt";
        run;

        /* Bucle semillas */
        %do semilla=&sinicio %to &sfinal;
                data dos;
                        set &archivo;
                        u=ranuni(&semilla);
                run;
```

```
proc sort data=dos;
	by u;
run;

data dos (drop=nume);
	retain grupo 1;
	set dos nobs=nume;
	if _n_> grupo * nume / &ngrupos then grupo = grupo + 1;
run;

data fantasma;
run;
%do exclu=1 %to &ngrupos;

	data trestr tresval;
		set dos;
		if grupo ne &exclu then output trestr;
		else output tresval;
	run;

	proc dmdb data=trestr dmdbcat=catatres;
		target &vardepen;
		class &categor &vardepen;
	run;

	proc neural data=trestr dmdbcat=catatres random=789;
		input &categor /level=nominal;
		target &vardepen /level=nominal;
		hidden &nodos /act=&acti;
		netoptions randist=normal ranscale=0.15 random=15459;
		prelim 15 preiter=10 pretech=&algo;
		%if &early ne %then %do;
			train maxiter=&early outest=mlpest technique=&algo;
		%end;
		%else %do;
			train outest=mlpest technique=&algo;
		%end;
		score data=tresval role=valid out=sal;
	run;

	data sal2;
		set sal;
		pro=1-%str(p_&vardepen)0;
		if pro>0.5 then pre11=1;
		else pre11=0;
	run;

	data sal_final;
		set sal_final sal;
	run;

	proc freq data=sal2;
		tables pre11 * &vardepen /out=sal3;
	run;

	data estadisticos (drop=count percent pre11 &vardepen);
		retain vp vn fp fn suma 0;
		set sal3 nobs=nume;
		suma=suma+count;
		if pre11=0 and &vardepen=0 then vn=count;
		if pre11=0 and &vardepen=1 then fn=count;
		if pre11=1 and &vardepen=0 then fp=count;
		if pre11=1 and &vardepen=1 then vp=count;
		if _n_=nume then do;
			porcenVN=vn/suma;
			porcenFN=FN/suma;
			porcenVP=VP/suma;
			porcenFP=FP/suma;
			sensi=vp/(vp+fn);
			especif=vn/(vn+fp);
			tasafallos=1-(vp+vn)/suma;
			tasaciertos=1-tasafallos;
			precision=vp/(vp+fp);
			F_M=2*Sensi*Precision/(Sensi+Precision);
			output;
		end;
	run;
```

```
                        data fantasma;
                                set fantasma estadisticos;
                        run;
                %end;

                proc means data=fantasma sum noprint;
                        var &objetivo;
                        output out=sumaresi sum=suma mean=media;
                run;

                data sumaresi;
                        set sumaresi;
                        semilla=&semilla;
                run;

                data final (keep=suma media semilla);
                        set final sumaresi;
                        if suma=. then delete;
                run;
        %end;
        proc printto;
        run;

        proc print data=final;
        run;
%mend;


%macro
cruzadarandomforestbin(archivo=,vardep=,categor=,maxtrees=100,variables=3,porcenbag=0.80
,maxbranch=2,tamhoja=5,maxdepth=10,pvalor=0.1,ngrupos=4,sinicio=12340,sfinal=12345,objet
ivo=tasafallos);

        data final;
        run;

        data sal_final;
        run;

        %do semilla=&sinicio %to &sfinal;
                data dos;
                        set &archivo;
                        u=ranuni(&semilla);
                run;

                proc sort data=dos;
                        by u;
                run;

                data dos ;
                        retain grupo 1;
                        set dos nobs=nume;
                        if _n_>grupo*nume/&ngrupos then grupo=grupo+1;
                run;

                data fantasma;
                run;

                %do exclu=1 %to &ngrupos;
                        data tres;
                                set dos;
                                if grupo ne &exclu then vardep=&vardep;
                        run;

                        ods listing close;
                        proc hpforest data=tres maxtrees=&maxtrees vars_to_try=&variables
trainfraction=&porcenbag leafsize=&tamhoja maxdepth=&maxdepth alpha=&pvalor
exhaustive=5000 missing=useinsearch;
                                target vardep/level=nominal;
                                input &categor/level=nominal;
                                score out=salo;
                        run;

                        ods listing;

                        data sal_final;
```

```
                        set sal_final salo;
                run;

                data salo;
                        merge salo tres;
                        if p_vardep1>0.5 then pre11=1;else pre11=0;
                        if grupo=&exclu;
                run;

                proc freq data=salo;
                        tables pre11*&vardep/out=sal3;
                run;

                data estadisticos (drop=count percent pre11 &vardep);
                        retain vp vn fp fn suma 0;
                        set sal3 nobs=nume;
                        suma=suma+count;
                        if pre11=0 and &vardep=0 then vn=count;
                        if pre11=0 and &vardep=1 then fn=count;
                        if pre11=1 and &vardep=0 then fp=count;
                        if pre11=1 and &vardep=1 then vp=count;
                        if _n_=nume then do;
                                porcenVN=vn/suma;
                                porcenFN=FN/suma;
                                porcenVP=VP/suma;
                                porcenFP=FP/suma;
                                sensi=vp/(vp+fn);
                                especif=vn/(vn+fp);
                                tasafallos=1-(vp+vn)/suma;
                                tasaciertos=1-tasafallos;
                                precision=vp/(vp+fp);
                                F_M=2*Sensi*Precision/(Sensi+Precision);
                                output;
                        end;
                run;

                data fantasma;
                        set fantasma estadisticos;
                run;

        %end;

        proc means data=fantasma sum noprint;
                var &objetivo;
                output out=sumaresi sum=suma mean=media;
        run;

        data sumaresi;
                set sumaresi;
                semilla=&semilla;
        run;

        data final (keep=suma media semilla);
                set final sumaresi;
                if suma=. then delete;
        run;
%end;

proc print data=final;
run;
%mend;


%macro
cruzadabaggingbin(archivo=,vardepen=,listcategor=,ngrupos=4,sinicio=12340,sfinal=12345,s
iniciobag=12340,sfinalbag=12345,porcenbag=0.80,maxbranch=2,nleaves=6,tamhoja=5,reemplazo
=1,objetivo=tasafallos);

data final;
run;

data sal_final;
run;

%do semilla=&sinicio %to &sfinal;

        data dos;
```

```
                set &archivo;
                u=ranuni(&semilla);
        run;

        proc sort data=dos;
                by u;
        run;

        data dos;
                retain grupo 1;
                set dos nobs=nume;
                if _n_>grupo*nume/&ngrupos then grupo=grupo+1;
        run;

        data fantasma;
        run;

        %do exclu=1 %to &ngrupos;
                data tres;
                        set dos;
                        if grupo ne &exclu then vardep=&vardepen*1;
                run;

                data tresbis trespred;
                        set tres;
                        if grupo ne &exclu then output tresbis;
                        else output trespred;
                run;

                %do sem=&siniciobag %to &sfinalbag;
                        data;
                                numero=&sem-&siniciobag+1;
                                call symput('numero',left(numero));
                                total=&sfinalbag-&siniciobag+1;
                                call symput('total',left(total));
                        run;

                        %if &reemplazo=0 %then %do;
                                proc surveyselect data=tresbis out=muestra2 outall
method=srs seed=&sem samprate=&porcenbag noprint;
                                run;
                        %end;
                        %else %do;
                                proc surveyselect data=tresbis out=muestra2 outall
method=urs seed=&sem samprate=&porcenbag noprint;
                                run;
                        %end;

                        data entreno1;
                                set muestra2;
                                if selected=1 then output entreno1;
                                drop selected;
                        run;

                        proc arbor data=entreno1 ;
                                input &listcategor/level=nominal;
                                target vardep /level=nominal;
                                interact largest;
                                train maxbranch=&maxbranch leafsize=&tamhoja;
                                subtree nleaves=&nleaves;
                                score data=trespred out=sal;
                        run;

                        data sal;
                                set sal;
                                vardepen&numero=p_vardep1;
                        run;

                        data sal_final;
                                set sal_final sal;
                        run;

                        %if &numero=1 %then %do;
                                data uni;
                                        set sal;ypredi=vardepen&numero;
                                        keep ypredi &vardepen;
                                run;
```

```
                        %end;
                        %else %do;
                                data uni;
                                        merge uni sal;
                                        ypredi=vardepen&numero+ypredi;
                                        keep ypredi &vardepen;
                                run;
                        %end;
                %end;

                data sal2;
                        set uni ;ypredi=ypredi/&total;
                        if ypredi>0.5 then pre11=1;
                        else pre11=0;
                run;

                proc freq data=sal2;
                        tables pre11*&vardepen/out=sal3;
                run;
                data estadisticos (drop=count percent pre11 &vardepen);
                        retain vp vn fp fn suma 0;
                        set sal3 nobs=nume;
                        suma=suma+count;
                        if pre11=0 and &vardepen=0 then vn=count;
                        if pre11=0 and &vardepen=1 then fn=count;
                        if pre11=1 and &vardepen=0 then fp=count;
                        if pre11=1 and &vardepen=1 then vp=count;
                        if _n_=nume then do;
                                porcenVN=vn/suma;
                                porcenFN=FN/suma;
                                porcenVP=VP/suma;
                                porcenFP=FP/suma;
                                sensi=vp/(vp+fn);
                                especif=vn/(vn+fp);
                                tasafallos=1-(vp+vn)/suma;
                                tasaciertos=1-tasafallos;
                                precision=vp/(vp+fp);
                                F_M=2*Sensi*Precision/(Sensi+Precision);
                                output;
                        end;
                run;

                data fantasma;
                        set fantasma estadisticos;
                run;
        %end;

        proc means data=fantasma sum noprint;
                var &objetivo;
                output out=sumaresi sum=suma mean=media;
        run;

        data sumaresi;
                set sumaresi;
                semilla=&semilla;
        run;

        data final (keep=suma media semilla);
                set final sumaresi;
                if suma=. then delete;
        run;
    %end;

    proc print data=final;
    run;
%mend;


%macro
cruzadatreeboostbin(archivo=,vardepen=,categor=,ngrupos=,sinicio=,sfinal=,leafsize=5,ite
raciones=100,shrink=0.01,maxbranch=2,maxdepth=4,mincatsize=15,minobs=20,objetivo=tasafal
los);
    data final;
    run;

    data sal_final;
    run;
```

```
	%do semilla=&sinicio %to &sfinal;
		data dos;
			set &archivo;
			u=ranuni(&semilla);
		run;

		proc sort data=dos;
			by u;
		run;

		data dos;
			retain grupo 1;
			set dos nobs=nume;
			if _n_>grupo*nume/&ngrupos then grupo=grupo+1;
		run;

		data fantasma;
		run;

		%do exclu=1 %to &ngrupos;
			data tres;
				set dos;
				if grupo ne &exclu then vardep=&vardepen;
			run;

			proc treeboost data=tres exhaustive=1000 intervaldecimals=max
leafsize=&leafsize iterations=&iteraciones maxbranch=&maxbranch maxdepth=&maxdepth
mincatsize=&mincatsize missing=useinsearch shrinkage=&shrink splitsize=&minobs;
				input &categor/level=nominal;
				target vardep /level=binary;
				save fit=iteraciones importance=impor model=modelo
rules=reglas;
				subseries largest;
				score out=sal;
			run;

			data sal2;
				set sal;
				pro=1-p_vardep0;
				if pro>0.5 then pre11=1;
				else pre11=0;
				if grupo=&exclu then output;
			run;

			data sal_final;
				set sal_final sal;
			run;

			proc freq data=sal2;
				tables pre11*&vardepen/out=sal3;
			run;

			data estadisticos (drop=count percent pre11 &vardepen);
				retain vp vn fp fn suma 0;
				set sal3 nobs=nume;
				suma=suma+count;
				if pre11=0 and &vardepen=0 then vn=count;
				if pre11=0 and &vardepen=1 then fn=count;
				if pre11=1 and &vardepen=0 then fp=count;
				if pre11=1 and &vardepen=1 then vp=count;
				if _n_=nume then do;
					porcenVN=vn/suma;
					porcenFN=FN/suma;
					porcenVP=VP/suma;
					porcenFP=FP/suma;
					sensi=vp/(vp+fn);
					especif=vn/(vn+fp);
					tasafallos=1-(vp+vn)/suma;
					tasaciertos=1-tasafallos;
					precision=vp/(vp+fp);
					F M=2*Sensi*Precision/(Sensi+Precision);
					output;
				end;
			run;

			data fantasma;
```

```
                        set fantasma estadisticos;
                run;
        %end;

        proc means data=fantasma sum noprint;
                var &objetivo;
                output out=sumaresi sum=suma mean=media;
        run;

        data sumaresi;
                set sumaresi;
                semilla=&semilla;
        run;

        data final (keep=suma media semilla);
                set final sumaresi;
                if suma=. then delete;
        run;
%end;

proc print data=final;
run;
%mend;
```

Annex 1.8: avNNet and Logistic Regression training functions in R

```
cruzadalogistica <- function(data = data,
                             vardep = NULL,
                             listclass = NULL,
                             grupos = 4,
                             sinicio = 1234,
                             repe = 5) {
  library(dplyr)
  library(pROC)

  databis <- data[, c(vardep, listclass)]
  databis <- dummy.data.frame(databis, listclass, sep = ".")

  x <- databis[, names(databis) != vardep]
  y <- as.factor(databis[, vardep])


  set.seed(sinicio)
  control <-
    trainControl(
      method = "repeatedcv",
      number = grupos,
      repeats = repe,
      savePredictions = "all",
      classProbs = TRUE
    )


  regresion <- train(
    x = x,
    y = y,
    trControl = control,
    method = "glm",
    family = binomial(link = "logit")
  )
  preditest <- regresion$pred

  preditest$prueba <- strsplit(preditest$Resample, "[.]")
  preditest$Fold <- sapply(preditest$prueba, "[", 1)
  preditest$Rep <- sapply(preditest$prueba, "[", 2)
  preditest$prueba <- NULL

  tasafallos <- function(x, y) {
    confu <- confusionMatrix(x, y)
    tasa <- confu[[3]][1]
    return(tasa)
  }

  medias <- preditest %>%
    group_by(Rep) %>%
```

```
    summarize(tasa = 1 - tasafallos(pred, obs))

  auc_func <- function(x, y) {
    curvaroc <- roc(response = x, predictor = y)
    auc_value <- curvaroc$auc
    return(as.numeric(auc_value))
  }

  mediasbis <- preditest %>%
    group_by(Rep) %>%
    summarize(auc = auc_func(obs, Yes))

  medias$auc <- mediasbis$auc

  return(medias)

}

cruzadalogencode <- function(data = data,
                             vardep = NULL,
                             listclass = NULL,
                             grupos = 4,
                             sinicio = 1234,
                             repe = 5,
                             lambda = 0,
                             interactions = FALSE) {
  library(h2o)
  h2o.init()

  df <- data.frame(matrix(ncol = 2, nrow = 0))
  colnames(df) <- c("seed", "auc")

  sfin <- sinicio + repe
  for (seed in sinicio:sfin) {
    splits <- h2o.splitFrame(data, seed = seed, ratios = c(0.8))

    train <- splits[[1]]
    test <- splits[[2]]


    # For k_fold strategy we need to provide fold column
    train$fold <-
      h2o.kfold_column(train, nfolds = grupos, seed = seed)

    # Train a TE model
    target_encoder <- h2o.targetencoder(
      training_frame = train,
      x = listclass,
      y = vardep,
      fold_column = "fold",
      data_leakage_handling = "KFold",
      blending = TRUE,
      inflection_point = 3,
      smoothing = 10,
      noise = 0.15,
      seed = seed
    )

    # New target encoded train and test sets
    transformed_train <-
      h2o.transform(target_encoder, train, as_training = TRUE)
    transformed_test <-
      h2o.transform(target_encoder, test, noise = 0)

    features_with_te <-
      setdiff(names(transformed train),
              c(vardep, listclass))

    if (interactions == TRUE) {
      log_with_te <- h2o.glm(
        x = features_with_te,
        y = vardep,
        training_frame = transformed_train,
        fold_column = "fold",
        family = "binomial",
        lambda = lambda,
        interactions = features_with_te
```

```
      )
    }
    else {
      log with te <- h2o.glm(
        x = features_with_te,
        y = vardep,
        training_frame = transformed_train,
        fold_column = "fold",
        family = "binomial",
        lambda = lambda
      )
    }

    # Measuring performance on a transformed_test split
    with_te_test_predictions <-
      predict(log with te, transformed test)

    auc <-
      h2o.auc(h2o.performance(log_with_te, transformed_test))
    df[nrow(df) + 1, ] <- c(seed, auc)
  }

  return(df)
}

cruzadaavnnetbin <- function(data = data,
                             vardep = "vardep",
                             listclass = "listclass",
                             grupos = 4,
                             sinicio = 1234,
                             repe = 5,
                             size = c(5),
                             decay = c(0.01),
                             repeticiones = 5,
                             itera = 100) {

  databis <- data[, c(vardep, listclass)]
  databis <- dummy.data.frame(databis, listclass, sep = ".")

  x <- databis[, names(databis) != vardep]
  y <- as.factor(databis[, vardep])

  set.seed(sinicio)
  control <- trainControl(
    method = "repeatedcv",
    number = grupos,
    repeats = repe,
    savePredictions = "all",
    classProbs = TRUE
  )

  avnnetgrid <-  expand.grid(size = size,
                             decay = decay,
                             bag = FALSE)

  avnnet <- train(
    x = x,
    y = y,
    method = "avNNet",
    linout = FALSE,
    maxit = itera,
    repeats = repeticiones,
    trControl = control,
    tuneGrid = avnnetgrid,
    MaxNWts = 10000000
  )

  print(avnnet$results)

  preditest <- avnnet$pred

  preditest$prueba <- strsplit(preditest$Resample, "[.]")
  preditest$Fold <- sapply(preditest$prueba, "[", 1)
  preditest$Rep <- sapply(preditest$prueba, "[", 2)
  preditest$prueba <- NULL

  tasafallos <- function(x, y) {
```

```
    confu <- confusionMatrix(x, y)
    tasa <- confu[[3]][1]
    return(tasa)
  }

  medias <- preditest %>%
    group_by(Rep) %>%
    summarize(tasa = 1 - tasafallos(pred, obs))

  auc func <- function(x, y) {
    curvaroc <- roc(response = x, predictor = y)
    auc_value <- curvaroc$auc
    return(as.numeric(auc_value))
  }

  mediasbis <- preditest %>%
    group_by(Rep) %>%
    summarize(auc = auc_func(obs, Yes))

  medias$auc <- mediasbis$auc

  return(medias)

}

cruzadaneuralencode <- function(data = data,
                                vardep = NULL,
                                listclass = NULL,
                                grupos = 4,
                                sinicio = 1234,
                                repe = 5,
                                hidden = c(1),
                                activation = "Tanh",
                                l2 = 0,
                                epochs = 50
                                ) {

  library(h2o)
  h2o.init()

  df <- data.frame(matrix(ncol = 2, nrow = 0))
  colnames(df) <- c("seed", "auc")

  sfin <- sinicio + repe
  for (seed in sinicio:sfin) {
    splits <- h2o.splitFrame(data, seed = seed, ratios = c(0.8))

    train <- splits[[1]]
    test <- splits[[2]]


    # For k_fold strategy we need to provide fold column
    train$fold <-
      h2o.kfold_column(train, nfolds = grupos, seed = seed)

    # Train a TE model
    target_encoder <- h2o.targetencoder(
      training_frame = train,
      x = listclass,
      y = vardep,
      fold_column = "fold",
      data_leakage_handling = "KFold",
      blending = TRUE,
      inflection_point = 3,
      smoothing = 10,
      noise = 0.15,
      seed = seed
    )

    # New target encoded train and test sets
    transformed_train <-
      h2o.transform(target encoder, train, as training = TRUE)
    transformed_test <-
      h2o.transform(target_encoder, test, noise = 0)

    features_with_te <-
      setdiff(names(transformed_train),
```

```
            c(vardep, listclass))

    neural_with_te <- h2o.deeplearning(
      x = features with te,
      y = vardep,
      training_frame = transformed_train,
      fold_column = "fold",
      epochs = epochs,
      hidden = hidden,
      activation = activation,
      l2 = l2
    )

    # Measuring performance on a transformed_test split
    with_te_test_predictions <-
      predict(neural with te, transformed test)

    auc <-
      h2o.auc(h2o.performance(neural_with_te, transformed_test))
    df[nrow(df) + 1,] <- c(seed, auc)
  }

  return(df)
}
```

Annex 1.9: Regression Tree training function in R

```
cruzadaarbolbin <- function(data = data,
                            vardep = "vardep",
                            listclass = "listclass",
                            grupos = 4,
                            sinicio = 1234,
                            repe = 5,
                            cp = c(0),
                            minbucket = 20) {
  databis <- data[, c(vardep, listclass)]
  databis <- dummy.data.frame(databis, listclass, sep = ".")

  x <- databis[, names(databis) != vardep]
  y <- as.factor(databis[, vardep])

  set.seed(sinicio)
  control <-
    trainControl(
      method = "repeatedcv",
      number = grupos,
      repeats = repe,
      savePredictions = "all",
      classProbs = TRUE
    )

  arbolgrid <-  expand.grid(cp = cp)

  arbol <- train(
    x = x,
    y = y,
    method = "rpart",
    trControl = control,
    tuneGrid = arbolgrid,
    minbucket = minbucket
  )

  preditest <- arbol$pred

  preditest$prueba <- strsplit(preditest$Resample, "[.]")
  preditest$Fold <- sapply(preditest$prueba, "[", 1)
  preditest$Rep <- sapply(preditest$prueba, "[", 2)
  preditest$prueba <- NULL

  tasafallos <- function(x, y) {
    confu <- confusionMatrix(x, y)
    tasa <- confu[[3]][1]
    return(tasa)
  }
```

```
  medias <- preditest %>%
    group_by(Rep) %>%
    summarize(tasa = 1 - tasafallos(pred, obs))

  auc_func <- function(x, y) {
    curvaroc <- roc(response = x, predictor = y)
    auc_value <- curvaroc$auc
    return(as.numeric(auc_value))
  }

  mediasbis <- preditest %>%
    group_by(Rep) %>%
    summarize(auc = auc_func(obs, Yes))

  medias$auc <- mediasbis$auc

  return(medias)

}
```

Annex 1.10: Gradient Boosting training function in R

```
cruzadagbmbin <-  function(data = data,
                           vardep = "vardep",
                           listclass = "listclass",
                           grupos = 4,
                           sinicio = 1234,
                           repe = 5,
                           n.minobsinnode = 20,
                           shrinkage = 0.1,
                           n.trees = 100,
                           interaction.depth = 2) {
  databis <- data[, c(vardep, listclass)]
  databis <- dummy.data.frame(databis, listclass, sep = ".")

  x <- databis[, names(databis) != vardep]
  y <- as.factor(databis[, vardep])

  set.seed(sinicio)
  control <-
    trainControl(
      method = "repeatedcv",
      number = grupos,
      repeats = repe,
      savePredictions = "all",
      classProbs = TRUE
    )

  gbmgrid <- expand.grid(
    n.minobsinnode = n.minobsinnode,
    shrinkage = shrinkage,
    n.trees = n.trees,
    interaction.depth = interaction.depth
  )

  gbm <- train(
    x = x,
    y = y,
    method = "gbm",
    trControl = control,
    tuneGrid = gbmgrid,
    distribution = "bernoulli",
    verbose = FALSE
  )

  print(gbm$results)

  preditest <- gbm$pred


  preditest$prueba <- strsplit(preditest$Resample, "[.]")
  preditest$Fold <- sapply(preditest$prueba, "[", 1)
  preditest$Rep <- sapply(preditest$prueba, "[", 2)
  preditest$prueba <- NULL
```

```
  tasafallos <- function(x, y) {
    confu <- confusionMatrix(x, y)
    tasa <- confu[[3]][1]
    return(tasa)
  }

  medias <- preditest %>%
    group_by(Rep) %>%
    summarize(tasa = 1 - tasafallos(pred, obs))

  auc_func <- function(x, y) {
    curvaroc <- roc(response = x, predictor = y)
    auc_value <- curvaroc$auc
    return(as.numeric(auc_value))
  }

  mediasbis <- preditest %>%
    group_by(Rep) %>%
    summarize(auc = auc_func(obs, Yes))

  medias$auc <- mediasbis$auc

  return(medias)

}

cruzadagbmencode <- function(data = data,
                             vardep = NULL,
                             listclass = NULL,
                             grupos = 4,
                             sinicio = 1234,
                             repe = 5,
                             ntrees = 200,
                             max_depth = 5,
                             min_rows = 100,
                             nbins_cats = 1024
                             ) {

  library(h2o)
  h2o.init()

  df <- data.frame(matrix(ncol = 2, nrow = 0))
  colnames(df) <- c("seed", "auc")

  sfin <- sinicio + repe
  for (seed in sinicio:sfin) {
    splits <- h2o.splitFrame(data, seed = seed, ratios = c(0.8))

    train <- splits[[1]]
    test <- splits[[2]]


    # For k_fold strategy we need to provide fold column
    train$fold <-
      h2o.kfold_column(train, nfolds = grupos, seed = seed)

    # Train a TE model
    target_encoder <- h2o.targetencoder(
      training_frame = train,
      x = listclass,
      y = vardep,
      fold_column = "fold",
      data_leakage_handling = "KFold",
      blending = TRUE,
      inflection_point = 3,
      smoothing = 10,
      noise = 0.15,
      seed = seed
    )

    # New target encoded train and test sets
    transformed train <-
      h2o.transform(target_encoder, train, as_training = TRUE)
    transformed_test <-
      h2o.transform(target_encoder, test, noise = 0)

    features_with_te <-
```

```
      setdiff(names(transformed_train),
              c(vardep, listclass))

    gbm_with_te <- h2o.gbm(
      x = features_with_te,
      y = vardep,
      training_frame = transformed_train,
      fold_column = "fold",
      ntrees = ntrees,
      max_depth = max_depth,
      min_rows = min_rows,
      nbins_cats = nbins_cats
    )

    # Measuring performance on a transformed_test split
    with_te_test_predictions <-
      predict(gbm_with_te, transformed_test)

    auc <-
      h2o.auc(h2o.performance(gbm_with_te, transformed_test))
    df[nrow(df) + 1,] <- c(seed, auc)
  }

  return(df)
}
```

Annex 1.11: Random Forest training function in R

```
cruzadarfbin <- function(data = data,
                         vardep = "vardep",
                         listclass = "listclass",
                         grupos = 4,
                         sinicio = 1234,
                         repe = 5,
                         nodesize = 20,
                         mtry = 2,
                         ntree = 50,
                         replace = TRUE) {
  databis <- data[, c(vardep, listclass)]
  databis <- dummy.data.frame(databis, listclass, sep = ".")

  x <- databis[, names(databis) != vardep]
  y <- as.factor(databis[, vardep])

  set.seed(sinicio)
  control <-
    trainControl(
      method = "repeatedcv",
      number = grupos,
      repeats = repe,
      savePredictions = "all",
      classProbs = TRUE
    )

  rfgrid <- expand.grid(mtry = mtry)

  rf <- train(
    x = x,
    y = y,
    method = "rf",
    trControl = control,
    tuneGrid = rfgrid,
    nodesize = nodesize,
    replace = replace,
    ntree = ntree
  )

  print(rf$results)

  preditest <- rf$pred

  preditest$prueba <- strsplit(preditest$Resample, "[.]")
  preditest$Fold <- sapply(preditest$prueba, "[", 1)
  preditest$Rep <- sapply(preditest$prueba, "[", 2)
```

```
  preditest$prueba <- NULL

  tasafallos <- function(x, y) {
    confu <- confusionMatrix(x, y)
    tasa <- confu[[3]][1]
    return(tasa)
  }

  medias <- preditest %>%
    group_by(Rep) %>%
    summarize(tasa = 1 - tasafallos(pred, obs))

  auc_func <- function(x, y) {
    curvaroc <- roc(response = x, predictor = y)
    auc_value <- curvaroc$auc
    return(as.numeric(auc value))
  }

  mediasbis <- preditest %>%
    group_by(Rep) %>%
    summarize(auc = auc_func(obs, Yes))

  medias$auc <- mediasbis$auc

  return(medias)

}

cruzadarfencode <- function(data = data,
                            vardep = NULL,
                            listclass = NULL,
                            grupos = 4,
                            sinicio = 1234,
                            repe = 5,
                            ntrees = 200,
                            max_depth = 5,
                            min_rows = 100,
                            mtries = -1,
                            nbins_cats = 1024
                            ) {

  library(h2o)
  h2o.init()

  df <- data.frame(matrix(ncol = 2, nrow = 0))
  colnames(df) <- c("seed", "auc")

  sfin <- sinicio + repe
  for (seed in sinicio:sfin) {
    splits <- h2o.splitFrame(data, seed = seed, ratios = c(0.8))

    train <- splits[[1]]
    test <- splits[[2]]


    # For k_fold strategy we need to provide fold column
    train$fold <-
      h2o.kfold_column(train, nfolds = grupos, seed = seed)

    # Train a TE model
    target_encoder <- h2o.targetencoder(
      training_frame = train,
      x = listclass,
      y = vardep,
      fold_column = "fold",
      data_leakage_handling = "KFold",
      blending = TRUE,
      inflection_point = 3,
      smoothing = 10,
      noise = 0.15,
      seed = seed
    )

    # New target encoded train and test sets
    transformed_train <-
      h2o.transform(target_encoder, train, as_training = TRUE)
    transformed_test <-
```

```
      h2o.transform(target_encoder, test, noise = 0)

    features_with_te <-
      setdiff(names(transformed train),
              c(vardep, listclass))

    rf_with_te <- h2o.randomForest(
      x = features_with_te,
      y = vardep,
      training frame = transformed train,
      fold_column = "fold",
      ntrees = ntrees,
      max_depth = max_depth,
      min_rows = min_rows,
      nbins_cats = nbins_cats,
      mtries = mtries,
      binomial_double_trees = TRUE
    )

    # Measuring performance on a transformed_test split
    with_te_test_predictions <-
      predict(rf_with_te, transformed_test)

    auc <-
      h2o.auc(h2o.performance(rf_with_te, transformed_test))
    df[nrow(df) + 1,] <- c(seed, auc)
  }

  return(df)
}
```

Annex 1.12: XGBoost training function in R

```
cruzadaxgbmbin <- function(data = data,
                           vardep = "vardep",
                           listclass = "listclass",
                           grupos = 4,
                           sinicio = 1234,
                           repe = 5,
                           min_child_weight = 20,
                           eta = 0.1,
                           nrounds = 100,
                           max_depth = 2,
                           gamma = 0,
                           colsample_bytree = 1,
                           subsample = 1,
                           alpha = 0,
                           lambda = 0,
                           lambda_bias = 0) {
  databis <- data[, c(vardep, listclass)]
  databis <- dummy.data.frame(databis, listclass, sep = ".")

  x <- databis[, names(databis) != vardep]
  y <- as.factor(databis[, vardep])

  set.seed(sinicio)
  control <-
    trainControl(
      method = "repeatedcv",
      number = grupos,
      repeats = repe,
      savePredictions = "all",
      classProbs = TRUE
    )

  xgbmgrid <- expand.grid(
    min_child_weight = min_child_weight,
    eta = eta,
    nrounds = nrounds,
    max_depth = max_depth,
    gamma = gamma,
    colsample_bytree = colsample_bytree,
    subsample = subsample
  )
```

```
  xgbm <- train(
    x = x,
    y = y,
    method = "xgbTree",
    trControl = control,
    tuneGrid = xgbmgrid,
    objective = "binary:logistic",
    verbose = FALSE,
    alpha = alpha,
    lambda = lambda,
    lambda_bias = lambda_bias
  )

  print(xgbm$results)

  preditest <- xgbm$pred


  preditest$prueba <- strsplit(preditest$Resample, "[.]")
  preditest$Fold <- sapply(preditest$prueba, "[", 1)
  preditest$Rep <- sapply(preditest$prueba, "[", 2)
  preditest$prueba <- NULL

  tasafallos <- function(x, y) {
    confu <- confusionMatrix(x, y)
    tasa <- confu[[3]][1]
    return(tasa)
  }

  medias <- preditest %>%
    group_by(Rep) %>%
    summarize(tasa = 1 - tasafallos(pred, obs))

  auc_func <- function(x, y) {
    curvaroc <- roc(response = x, predictor = y)
    auc_value <- curvaroc$auc
    return(as.numeric(auc_value))
  }

  mediasbis <- preditest %>%
    group_by(Rep) %>%
    summarize(auc = auc_func(obs, Yes))

  medias$auc <- mediasbis$auc

  return(medias)

}
```

Annex 1.13: Two methods variable selection model training in SAS Base

```
%cruzadalogistica(
        archivo=news_reduced_clean_two_methods,
        vardepen=REP_clicked,
        categor=REP_P102_head REP_P103_head REP_P103_tail REP_P1050_head REP_P1050_tail
REP_P106_head REP_P106_tail REP_P118_head REP_P131_tail REP_P1340_tail REP_P1343_head
REP_P1343_tail REP_P1412_head REP_P1412_tail REP_P1532_head REP_P1622_head REP_P166_head
REP_P166_tail REP_P172_head REP_P172_tail REP_P17_tail REP_P1884_tail REP_P206_head
REP_P206_tail REP_P21_head REP_P27_tail REP_P2936_head REP_P2936_tail REP_P30_head
REP_P30_tail REP_P31_head REP_P31_tail REP_P361_head REP_P361_tail REP_P37_head
REP_P37_tail REP_P39_tail REP_P421_tail REP_P463_tail REP_P495_head REP_P5008_head
REP_P5008_tail REP_P552_tail REP_P641_tail REP_P6886_head REP_P6886_tail REP_P937_head
REP_P937_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        objetivo=tasafallos
);

data Results.llv2;
        set final;
run;
```

```
data Results.sal_llv2;
	set sal_final;
run;

%cruzadatreeboostbin(
	archivo=news_reduced_clean_two_methods,
	vardepen=REP_clicked,
	categor=REP_P102_head REP_P103_head REP_P103_tail REP_P1050_head REP_P1050_tail
REP_P106_head REP_P106_tail REP_P118_head REP_P131_tail REP_P1340_tail REP_P1343_head
REP P1343 tail REP P1412 head REP P1412 tail REP P1532 head REP P1622 head REP P166_head
REP_P166_tail REP_P172_head REP_P172_tail REP_P17_tail REP_P1884_tail REP_P206_head
REP_P206_tail REP_P21_head REP_P27_tail REP_P2936_head REP_P2936_tail REP_P30_head
REP_P30_tail REP_P31_head REP_P31_tail REP_P361_head REP_P361_tail REP_P37_head
REP_P37_tail REP_P39_tail REP_P421_tail REP_P463_tail REP_P495_head REP_P5008_head
REP_P5008_tail REP_P552_tail REP_P641_tail REP_P6886_head REP_P6886_tail REP_P937_head
REP P937 tail REP_category REP_subcategory,
	ngrupos=4,
	sinicio=12345,
	sfinal=12354,
	leafsize=74,
	iteraciones=100,
	shrink=0.03,
	maxbranch=2,
	maxdepth=4,
	mincatsize=74,
	minobs=74,
	objetivo=tasafallos
);

data Results.gb003v2;
	set final;
run;

data Results.sal_gb003v2;
	set sal_final;
run;

%cruzadatreeboostbin(
	archivo=news_reduced_clean_two_methods,
	vardepen=REP_clicked,
	categor=REP_P102_head REP_P103_head REP_P103_tail REP_P1050_head REP_P1050_tail
REP_P106_head REP_P106_tail REP_P118_head REP_P131_tail REP_P1340_tail REP_P1343_head
REP_P1343_tail REP_P1412_head REP_P1412_tail REP_P1532_head REP_P1622_head REP_P166_head
REP_P166_tail REP_P172_head REP_P172_tail REP_P17_tail REP_P1884_tail REP_P206_head
REP_P206_tail REP_P21_head REP_P27_tail REP_P2936_head REP_P2936_tail REP_P30_head
REP_P30_tail REP_P31_head REP_P31_tail REP_P361_head REP_P361_tail REP_P37_head
REP_P37_tail REP_P39_tail REP_P421_tail REP_P463_tail REP_P495_head REP_P5008_head
REP_P5008_tail REP_P552_tail REP_P641_tail REP_P6886_head REP_P6886_tail REP_P937_head
REP_P937_tail REP_category REP_subcategory,
	ngrupos=4,
	sinicio=12345,
	sfinal=12354,
	leafsize=74,
	iteraciones=100,
	shrink=0.1,
	maxbranch=2,
	maxdepth=4,
	mincatsize=74,
	minobs=74,
	objetivo=tasafallos
);

data Results.gb01v2;
	set final;
run;

data Results.sal_gb01v2;
	set sal_final;
run;

%cruzadatreeboostbin(
	archivo=news reduced clean_two_methods,
	vardepen=REP_clicked,
	categor=REP_P102_head REP_P103_head REP_P103_tail REP_P1050_head REP_P1050_tail
REP_P106_head REP_P106_tail REP_P118_head REP_P131_tail REP_P1340_tail REP_P1343_head
REP_P1343_tail REP_P1412_head REP_P1412_tail REP_P1532_head REP_P1622_head REP_P166_head
REP_P166_tail REP_P172_head REP_P172_tail REP_P17_tail REP_P1884_tail REP_P206_head
```

```
REP_P206_tail REP_P21_head REP_P27_tail REP_P2936_head REP_P2936_tail REP_P30_head
REP_P30_tail REP_P31_head REP_P31_tail REP_P361_head REP_P361_tail REP_P37_head
REP_P37_tail REP_P39_tail REP_P421_tail REP_P463_tail REP_P495_head REP_P5008_head
REP_P5008_tail REP_P552_tail REP_P641_tail REP_P6886_head REP_P6886_tail REP_P937_head
REP_P937_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        leafsize=74,
        iteraciones=100,
        shrink=0.1,
        maxbranch=2,
        maxdepth=6,
        mincatsize=74,
        minobs=74,
        objetivo=tasafallos
);

data Results.gbd601v2;
        set final;
run;

data Results.sal_gbd601v2;
        set sal_final;
run;

%cruzadatreeboostbin(
        archivo=news_reduced_clean_two_methods,
        vardepen=REP_clicked,
        categor=REP_P102_head REP_P103_head REP_P103_tail REP_P1050_head REP_P1050_tail
REP_P106_head REP_P106_tail REP_P118_head REP_P131_tail REP_P1340_tail REP_P1343_head
REP_P1343_tail REP_P1412_head REP_P1412_tail REP_P1532_head REP_P1622_head REP_P166_head
REP_P166_tail REP_P172_head REP_P172_tail REP_P17_tail REP_P1884_tail REP_P206_head
REP_P206_tail REP_P21_head REP_P27_tail REP_P2936_head REP_P2936_tail REP_P30_head
REP_P30_tail REP_P31_head REP_P31_tail REP_P361_head REP_P361_tail REP_P37_head
REP_P37_tail REP_P39_tail REP_P421_tail REP_P463_tail REP_P495_head REP_P5008_head
REP_P5008_tail REP_P552_tail REP_P641_tail REP_P6886_head REP_P6886_tail REP_P937_head
REP_P937_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        leafsize=74,
        iteraciones=100,
        shrink=0.001,
        maxbranch=2,
        maxdepth=4,
        mincatsize=74,
        minobs=74,
        objetivo=tasafallos
);

data Results.gb0001v2;
        set final;
run;

data Results.sal_gb0001v2;
        set sal_final;
run;

%cruzadabaggingbin(
        archivo=news_reduced_clean_two_methods,
        vardepen=REP_clicked,
        listcategor=REP_P102_head REP_P103_head REP_P103_tail REP_P1050_head
REP_P1050_tail REP_P106_head REP_P106_tail REP_P118_head REP_P131_tail REP_P1340_tail
REP_P1343_head REP_P1343_tail REP_P1412_head REP_P1412_tail REP_P1532_head
REP_P1622_head REP_P166_head REP_P166_tail REP_P172_head REP_P172_tail REP_P17_tail
REP_P1884_tail REP_P206_head REP_P206_tail REP_P21_head REP_P27_tail REP_P2936_head
REP_P2936_tail REP_P30_head REP_P30_tail REP_P31_head REP_P31_tail REP_P361_head
REP_P361_tail REP_P37_head REP_P37_tail REP_P39_tail REP_P421_tail REP_P463_tail
REP_P495_head REP_P5008_head REP_P5008_tail REP_P552_tail REP_P641_tail REP_P6886_head
REP_P6886_tail REP_P937_head REP_P937_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        siniciobag=12345,
        sfinalbag=12354,
        porcenbag=0.80,
```

```
        maxbranch=2,
        nleaves=6,
        tamhoja=5,
        reemplazo=1,
        objetivo=tasafallos
);

data Results.bagn6m5v2;
        set final;
run;

data Results.sal_bagn6m5v2;
        set sal_final;
run;

%cruzadarandomforestbin(
        archivo=news_reduced_clean_two_methods,
        vardep=REP_clicked,
        categor=REP_P102_head REP_P103_head REP_P103_tail REP_P1050_head REP_P1050_tail
REP_P106_head REP_P106_tail REP_P118_head REP_P131_tail REP_P1340_tail REP_P1343_head
REP_P1343_tail REP_P1412_head REP_P1412_tail REP_P1532_head REP_P1622_head REP_P166_head
REP_P166_tail REP_P172_head REP_P172_tail REP_P17_tail REP_P1884_tail REP_P206_head
REP_P206_tail REP_P21_head REP_P27_tail REP_P2936_head REP_P2936_tail REP_P30_head
REP_P30_tail REP_P31_head REP_P31_tail REP_P361_head REP_P361_tail REP_P37_head
REP_P37_tail REP_P39_tail REP_P421_tail REP_P463_tail REP_P495_head REP_P5008_head
REP_P5008_tail REP_P552_tail REP_P641_tail REP_P6886_head REP_P6886_tail REP_P937_head
REP_P937_tail REP_category REP_subcategory,
        ngrupos=4,
        maxtrees=100,
        variables=6,
        porcenbag=0.80,
        maxbranch=2,
        tamhoja=5,
        maxdepth=10,
        pvalor=0.1,
        sinicio=12345,
        sfinal=12354,
        objetivo=tasafallos
);

data Results.rfnt100var6v2;
        set final;
run;

data Results.sal_rfnt100var6v2;
        set sal_final;
run;

%cruzadarandomforestbin(
        archivo=news_reduced_clean_two_methods,
        vardep=REP_clicked,
        categor=REP_P102_head REP_P103_head REP_P103_tail REP_P1050_head REP_P1050_tail
REP_P106_head REP_P106_tail REP_P118_head REP_P131_tail REP_P1340_tail REP_P1343_head
REP_P1343_tail REP_P1412_head REP_P1412_tail REP_P1532_head REP_P1622_head REP_P166_head
REP_P166_tail REP_P172_head REP_P172_tail REP_P17_tail REP_P1884_tail REP_P206_head
REP_P206_tail REP_P21_head REP_P27_tail REP_P2936_head REP_P2936_tail REP_P30_head
REP_P30_tail REP_P31_head REP_P31_tail REP_P361_head REP_P361_tail REP_P37_head
REP_P37_tail REP_P39_tail REP_P421_tail REP_P463_tail REP_P495_head REP_P5008_head
REP_P5008_tail REP_P552_tail REP_P641_tail REP_P6886_head REP_P6886_tail REP_P937_head
REP_P937_tail REP_category REP_subcategory,
        ngrupos=4,
        maxtrees=200,
        variables=12,
        porcenbag=0.80,
        maxbranch=2,
        tamhoja=5,
        maxdepth=10,
        pvalor=0.1,
        sinicio=12345,
        sfinal=12354,
        objetivo=tasafallos
);

data Results.rfnt200var12v2;
        set final;
run;
```

```
data Results.sal_rfnt200var12v2;
	set sal_final;
run;
```

Annex 1.14: Three methods variable selection model training in SAS Base

```
%cruzadalogistica(
	archivo=news_reduced_clean_three_methods,
	vardepen=REP_clicked,
	categor=REP_P102_head REP_P106_tail REP_P131_tail REP_P1340_tail REP_P1343_tail
REP_P1532_head REP_P166_tail REP_P172_head REP_P17_tail REP_P206_tail REP_P27_tail
REP_P2936_tail REP_P31_head REP_P31_tail REP_P361_tail REP_P37_head REP_P421_tail
REP_P463_tail REP_P641_tail REP_P6886_head REP_category REP_subcategory ,
	ngrupos=4,
	sinicio=12345,
	sfinal=12354,
	objetivo=tasafallos
);

data Results.llv3;
	set final;
run;

data Results.sal_llv3;
	set sal_final;
run;

%cruzadatreeboostbin(
	archivo=news_reduced_clean_three_methods,
	vardepen=REP_clicked,
	categor=REP_P102_head REP_P106_tail REP_P131_tail REP_P1340_tail REP_P1343_tail
REP_P1532_head REP_P166_tail REP_P172_head REP_P17_tail REP_P206_tail REP_P27_tail
REP_P2936_tail REP_P31_head REP_P31_tail REP_P361_tail REP_P37_head REP_P421_tail
REP_P463_tail REP_P641_tail REP_P6886_head REP_category REP_subcategory ,
	ngrupos=4,
	sinicio=12345,
	sfinal=12354,
	leafsize=74,
	iteraciones=100,
	shrink=0.03,
	maxbranch=2,
	maxdepth=4,
	mincatsize=74,
	minobs=74,
	objetivo=tasafallos
);

data Results.gb003v3;
	set final;
run;

data Results.sal_gb003v3;
	set sal_final;
run;

%cruzadatreeboostbin(
	archivo=news_reduced_clean_three_methods,
	vardepen=REP_clicked,
	categor=REP_P102_head REP_P106_tail REP_P131_tail REP_P1340_tail REP_P1343_tail
REP_P1532_head REP_P166_tail REP_P172_head REP_P17_tail REP_P206_tail REP_P27_tail
REP_P2936_tail REP_P31_head REP_P31_tail REP_P361_tail REP_P37_head REP_P421_tail
REP_P463_tail REP_P641_tail REP_P6886_head REP_category REP_subcategory ,
	ngrupos=4,
	sinicio=12345,
	sfinal=12354,
	leafsize=74,
	iteraciones=100,
	shrink=0.1,
	maxbranch=2,
	maxdepth=4,
	mincatsize=74,
	minobs=74,
```

```
        objetivo=tasafallos
);

data Results.gb01v3;
        set final;
run;

data Results.sal_gb01v3;
        set sal_final;
run;

%cruzadatreeboostbin(
        archivo=news_reduced_clean_three_methods,
        vardepen=REP_clicked,
        categor=REP_P102_head REP_P106_tail REP_P131_tail REP_P1340_tail REP_P1343_tail
REP P1532 head REP P166 tail REP P172 head REP P17 tail REP P206 tail REP P27 tail
REP_P2936_tail REP_P31_head REP_P31_tail REP_P361_tail REP_P37_head REP_P421_tail
REP_P463_tail REP_P641_tail REP_P6886_head REP_category REP_subcategory ,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        leafsize=74,
        iteraciones=100,
        shrink=0.1,
        maxbranch=2,
        maxdepth=6,
        mincatsize=74,
        minobs=74,
        objetivo=tasafallos
);

data Results.gbd601v3;
        set final;
run;

data Results.sal_gbd601v3;
        set sal_final;
run;

%cruzadatreeboostbin(
        archivo=news_reduced_clean_three_methods,
        vardepen=REP_clicked,
        categor=REP_P102_head REP_P106_tail REP_P131_tail REP_P1340_tail REP_P1343_tail
REP_P1532_head REP_P166_tail REP_P172_head REP_P17_tail REP_P206_tail REP_P27_tail
REP_P2936_tail REP_P31_head REP_P31_tail REP_P361_tail REP_P37_head REP_P421_tail
REP_P463_tail REP_P641_tail REP_P6886_head REP_category REP_subcategory ,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        leafsize=74,
        iteraciones=100,
        shrink=0.001,
        maxbranch=2,
        maxdepth=4,
        mincatsize=74,
        minobs=74,
        objetivo=tasafallos
);

data Results.gb0001v3;
        set final;
run;

data Results.sal_gb0001v3;
        set sal_final;
run;

%cruzadabaggingbin(
        archivo=news_reduced_clean_three_methods,
        vardepen=REP_clicked,
        listcategor=REP_P102_head REP_P106_tail REP_P131_tail REP_P1340_tail
REP P1343 tail REP P1532 head REP P166 tail REP P172 head REP P17 tail REP P206 tail
REP_P27_tail REP_P2936_tail REP_P31_head REP_P31_tail REP_P361_tail REP_P37_head
REP_P421_tail REP_P463_tail REP_P641_tail REP_P6886_head REP_category REP_subcategory ,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
```

```
        siniciobag=12345,
        sfinalbag=12354,
        porcenbag=0.80,
        maxbranch=2,
        nleaves=6,
        tamhoja=5,
        reemplazo=1,
        objetivo=tasafallos
);

data Results.bagn6m5v3;
        set final;
run;

data Results.sal_bagn6m5v3;
        set sal_final;
run;

%cruzadarandomforestbin(
        archivo=news_reduced_clean_three_methods,
        vardep=REP_clicked,
        categor=REP_P102_head REP_P106_tail REP_P131_tail REP_P1340_tail REP_P1343_tail
REP_P1532_head REP_P166_tail REP_P172_head REP_P17_tail REP_P206_tail REP_P27_tail
REP_P2936_tail REP_P31_head REP_P31_tail REP_P361_tail REP_P37_head REP_P421_tail
REP_P463_tail REP_P641_tail REP_P6886_head REP_category REP_subcategory ,
        ngrupos=4,
        maxtrees=100,
        variables=6,
        porcenbag=0.80,
        maxbranch=2,
        tamhoja=5,
        maxdepth=10,
        pvalor=0.1,
        sinicio=12345,
        sfinal=12354,
        objetivo=tasafallos
);

data Results.rfnt100var6v3;
        set final;
run;

data Results.sal_rfnt100var6v3;
        set sal_final;
run;

%cruzadarandomforestbin(
        archivo=news_reduced_clean_three_methods,
        vardep=REP_clicked,
        categor=REP_P102_head REP_P106_tail REP_P131_tail REP_P1340_tail REP_P1343_tail
REP_P1532_head REP_P166_tail REP_P172_head REP_P17_tail REP_P206_tail REP_P27_tail
REP_P2936_tail REP_P31_head REP_P31_tail REP_P361_tail REP_P37_head REP_P421_tail
REP_P463_tail REP_P641_tail REP_P6886_head REP_category REP_subcategory ,
        ngrupos=4,
        maxtrees=200,
        variables=12,
        porcenbag=0.80,
        maxbranch=2,
        tamhoja=5,
        maxdepth=10,
        pvalor=0.1,
        sinicio=12345,
        sfinal=12354,
        objetivo=tasafallos
);

data Results.rfnt200var12v3;
        set final;
run;

data Results.sal_rfnt200var12v3;
        set sal_final;
run;
```

Annex 1.15: Four methods variable selection model training in SAS Base

```
%cruzadalogistica(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        objetivo=tasafallos
);

data Results.llv4;
        set final;
run;

data Results.sal_llv4;
        set sal_final;
run;

%cruzadatreeboostbin(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        leafsize=74,
        iteraciones=100,
        shrink=0.03,
        maxbranch=2,
        maxdepth=4,
        mincatsize=74,
        minobs=74,
        objetivo=tasafallos
);

data Results.gb003v4;
        set final;
run;

data Results.sal_gb003v4;
        set sal_final;
run;

%cruzadatreeboostbin(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        leafsize=74,
        iteraciones=100,
        shrink=0.1,
        maxbranch=2,
        maxdepth=4,
        mincatsize=74,
        minobs=74,
        objetivo=tasafallos
);

data Results.gb01v4;
        set final;
run;

data Results.sal_gb01v4;
        set sal_final;
run;

%cruzadatreeboostbin(
```

```
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP P361 tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        leafsize=74,
        iteraciones=100,
        shrink=0.1,
        maxbranch=2,
        maxdepth=6,
        mincatsize=74,
        minobs=74,
        objetivo=tasafallos
);

data Results.gbd601v4;
        set final;
run;

data Results.sal_gbd601v4;
        set sal_final;
run;

%cruzadatreeboostbin(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        leafsize=74,
        iteraciones=100,
        shrink=0.001,
        maxbranch=2,
        maxdepth=4,
        mincatsize=74,
        minobs=74,
        objetivo=tasafallos
);

data Results.gb0001v4;
        set final;
run;

data Results.sal_gb0001v4;
        set sal_final;
run;

%cruzadabaggingbin(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        listcategor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        siniciobag=12345,
        sfinalbag=12354,
        porcenbag=0.80,
        maxbranch=2,
        nleaves=6,
        tamhoja=5,
        reemplazo=1,
        objetivo=tasafallos
);

data Results.bagn6m5v4;
        set final;
run;

data Results.sal_bagn6m5v4;
        set sal_final;
run;
```

```
%cruzadarandomforestbin(
        archivo=news_reduced_clean_four_methods,
        vardep=REP_clicked,
        categor=REP P1343 tail REP P17 tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        maxtrees=100,
        variables=6,
        porcenbag=0.80,
        maxbranch=2,
        tamhoja=5,
        maxdepth=10,
        pvalor=0.1,
        sinicio=12345,
        sfinal=12354,
        objetivo=tasafallos
);

data Results.rfnt100var6v4;
        set final;
run;

data Results.sal_rfnt100var6v4;
        set sal_final;
run;

%cruzadarandomforestbin(
        archivo=news_reduced_clean_four_methods,
        vardep=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        maxtrees=200,
        variables=12,
        porcenbag=0.80,
        maxbranch=2,
        tamhoja=5,
        maxdepth=10,
        pvalor=0.1,
        sinicio=12345,
        sfinal=12354,
        objetivo=tasafallos
);

data Results.rfnt200var12v4;
        set final;
run;

data Results.sal_rfnt200var12v4;
        set sal_final;
run;

%redneuronalbinaria(
        archivo=news_reduced_clean_four_methods,
        vardep=REP_clicked,
        listclass=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=2,
        meto=levmar,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news reduced_clean_four_methods,
        vardep=REP_clicked,
        listclass=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=2,
        meto=bprop,
        acti=tanh
);

%redneuronalbinaria(
```

```
        archivo=news_reduced_clean_four_methods,
        vardep=REP_clicked,
        listclass=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP P361 tail REP category REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=3,
        meto=levmar,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_four_methods,
        vardep=REP_clicked,
        listclass=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP P361 tail REP category REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=3,
        meto=bprop,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_four_methods,
        vardep=REP_clicked,
        listclass=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=4,
        meto=levmar,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_four_methods,
        vardep=REP_clicked,
        listclass=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=4,
        meto=bprop,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_four_methods,
        vardep=REP_clicked,
        listclass=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=5,
        meto=levmar,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_four_methods,
        vardep=REP_clicked,
        listclass=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=5,
        meto=bprop,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_four_methods,
        vardep=REP_clicked,
        listclass=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        porcen=0.8,
```

```
        semilla=12345,
        ocultos=6,
        meto=levmar,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_four_methods,
        vardep=REP_clicked,
        listclass=REP P1343 tail REP P17 tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=6,
        meto=bprop,
        acti=tanh
);

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=2,
        algo=bprop,
        objetivo=tasafallos,
        early=13,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\bproptanhearlyXn2v4
);

data Results.bproptanhearlyXn2v4;
        set final;
run;

data Results.sal_bproptanhearlyXn2v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=3,
        algo=levmar,
        objetivo=tasafallos,
        early=431,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\levtanhearlyXn3v4
);

data Results.levtanhearlyXn3v4;
        set final;
run;

data Results.sal_levtanhearlyXn3v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=3,
        algo=bprop,
        objetivo=tasafallos,
```

```
        early=13,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\bproptanhearlyXn3v4
);

data Results.bproptanhearlyXn3v4;
        set final;
run;

data Results.sal bproptanhearlyXn3v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=4,
        algo=levmar,
        objetivo=tasafallos,
        early=504,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\levtanhearlyXn4v4
);

data Results.levtanhearlyXn4v4;
        set final;
run;

data Results.sal_levtanhearlyXn4v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=4,
        algo=bprop,
        objetivo=tasafallos,
        early=13,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\bproptanhearlyXn4v4
);

data Results.bproptanhearlyXn4v4;
        set final;
run;

data Results.sal_bproptanhearlyXn4v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=5,
        algo=levmar,
        objetivo=tasafallos,
        early=541,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\levtanhearlyXn5v4
);
```

```
data Results.levtanhearlyXn5v4;
        set final;
run;

data Results.sal_levtanhearlyXn5v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news reduced clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=5,
        algo=bprop,
        objetivo=tasafallos,
        early=13,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\bproptanhearlyXn5v4
);

data Results.bproptanhearlyXn5v4;
        set final;
run;

data Results.sal_bproptanhearlyXn5v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=6,
        algo=levmar,
        objetivo=tasafallos,
        early=136,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\levtanhearlyXn6v4
);

data Results.levtanhearlyXn6v4;
        set final;
run;

data Results.sal_levtanhearlyXn6v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=6,
        algo=bprop,
        objetivo=tasafallos,
        early=13,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\bproptanhearlyXn6v4
);

data Results.bproptanhearlyXn6v4;
        set final;
run;

data Results.sal_bproptanhearlyXn6v4;
```

```
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=2,
        algo=levmar,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\levtanhearly0n2v4
);

data Results.levtanhearly0n2v4;
        set final;
run;

data Results.sal_levtanhearly0n2v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=2,
        algo=bprop,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\bproptanhearly0n2v4
);

data Results.bproptanhearly0n2v4;
        set final;
run;

data Results.sal_bproptanhearly0n2v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=3,
        algo=levmar,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\levtanhearly0n3v4
);

data Results.levtanhearly0n3v4;
        set final;
run;

data Results.sal levtanhearly0n3v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
```

```
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=3,
        algo=bprop,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\bproptanhearly0n3v4
);

data Results.bproptanhearly0n3v4;
        set final;
run;

data Results.sal_bproptanhearly0n3v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=4,
        algo=levmar,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\levtanhearly0n4v4
);

data Results.levtanhearly0n4v4;
        set final;
run;

data Results.sal_levtanhearly0n4v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=4,
        algo=bprop,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\bproptanhearly0n4v4
);

data Results.bproptanhearly0n4v4;
        set final;
run;

data Results.sal_bproptanhearly0n4v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news reduced clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
```

```
        sfinal=12354,
        nodos=5,
        algo=levmar,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\levtanhearly0n5v4
);

data Results.levtanhearly0n5v4;
        set final;
run;

data Results.sal_levtanhearly0n5v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=5,
        algo=bprop,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\bproptanhearly0n5v4
);

data Results.bproptanhearly0n5v4;
        set final;
run;

data Results.sal_bproptanhearly0n5v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=6,
        algo=levmar,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\levtanhearly0n6v4
);

data Results.levtanhearly0n6v4;
        set final;
run;

data Results.sal_levtanhearly0n6v4;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_four_methods,
        vardepen=REP_clicked,
        categor=REP_P1343_tail REP_P17_tail REP_P27_tail REP_P31_head REP_P31_tail
REP_P361_tail REP_category REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=6,
        algo=bprop,
        objetivo=tasafallos,
        early=,
```

```
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\4 methods\Tmp\bproptanhearly0n6v4
);

data Results.bproptanhearly0n6v4;
        set final;
run;

data Results.sal_bproptanhearly0n6v4;
        set sal_final;
run;
```

Annex 1.16: Five methods variable selection model training in SAS Base

```
%cruzadalogistica(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        objetivo=Area
);

data Results.llv5;
        set final;
run;

data Results.sal_llv5;
        set sal_final;
run;

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=2,
        meto=levmar,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=2,
        meto=bprop,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=3,
        meto=levmar,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=3,
        meto=bprop,
```

```
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=4,
        meto=levmar,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=4,
        meto=bprop,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=5,
        meto=levmar,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=5,
        meto=bprop,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=6,
        meto=levmar,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=6,
        meto=bprop,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=7,
        meto=levmar,
```

```
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=7,
        meto=bprop,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=8,
        meto=levmar,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=8,
        meto=bprop,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=9,
        meto=levmar,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=9,
        meto=bprop,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=10,
        meto=levmar,
        acti=tanh
);

%redneuronalbinaria(
        archivo=news_reduced_clean_five_methods,
        vardep=REP clicked,
        listclass=REP_P31_head REP_P31_tail REP_subcategory,
        porcen=0.8,
        semilla=12345,
        ocultos=10,
        meto=bprop,
```

```
        acti=tanh
);

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=2,
        algo=levmar,
        objetivo=tasafallos,
        early=141,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearlyXn2v5
);

data Results.levtanhearlyXn2v5;
        set final;
run;

data Results.sal_levtanhearlyXn2v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=2,
        algo=bprop,
        objetivo=tasafallos,
        early=13,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearlyXn2v5
);

data Results.bproptanhearlyXn2v5;
        set final;
run;

data Results.sal_bproptanhearlyXn2v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=3,
        algo=levmar,
        objetivo=tasafallos,
        early=166,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearlyXn3v5
);

data Results.levtanhearlyXn3v5;
        set final;
run;

data Results.sal_levtanhearlyXn3v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
```

```
        sinicio=12345,
        sfinal=12354,
        nodos=3,
        algo=bprop,
        objetivo=tasafallos,
        early=13,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearlyXn3v5
);

data Results.bproptanhearlyXn3v5;
        set final;
run;

data Results.sal_bproptanhearlyXn3v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=4,
        algo=levmar,
        objetivo=tasafallos,
        early=59,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearlyXn4v5
);

data Results.levtanhearlyXn4v5;
        set final;
run;

data Results.sal_levtanhearlyXn4v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=4,
        algo=bprop,
        objetivo=tasafallos,
        early=173,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearlyXn4v5
);

data Results.bproptanhearlyXn4v5;
        set final;
run;

data Results.sal_bproptanhearlyXn4v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news reduced clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=5,
        algo=levmar,
        objetivo=tasafallos,
        early=42,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearlyXn5v5
```

```
);

data Results.levtanhearlyXn5v5;
	set final;
run;

data Results.sal_levtanhearlyXn5v5;
	set sal_final;
run;

%cruzadabinarianeural(
	archivo=news_reduced_clean_five_methods,
	vardepen=REP_clicked,
	categor=REP_P31_head REP_P31_tail REP_subcategory,
	ngrupos=4,
	sinicio=12345,
	sfinal=12354,
	nodos=5,
	algo=bprop,
	objetivo=tasafallos,
	early=191,
	acti=tanh,
	directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearlyXn5v5
);

data Results.bproptanhearlyXn5v5;
	set final;
run;

data Results.sal_bproptanhearlyXn5v5;
	set sal_final;
run;

%cruzadabinarianeural(
	archivo=news_reduced_clean_five_methods,
	vardepen=REP_clicked,
	categor=REP_P31_head REP_P31_tail REP_subcategory,
	ngrupos=4,
	sinicio=12345,
	sfinal=12354,
	nodos=6,
	algo=levmar,
	objetivo=tasafallos,
	early=36,
	acti=tanh,
	directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearlyXn6v5
);

data Results.levtanhearlyXn6v5;
	set final;
run;

data Results.sal_levtanhearlyXn6v5;
	set sal_final;
run;

%cruzadabinarianeural(
	archivo=news_reduced_clean_five_methods,
	vardepen=REP_clicked,
	categor=REP_P31_head REP_P31_tail REP_subcategory,
	ngrupos=4,
	sinicio=12345,
	sfinal=12354,
	nodos=6,
	algo=bprop,
	objetivo=tasafallos,
	early=184,
	acti=tanh,
	directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearlyXn6v5
);

data Results.bproptanhearlyXn6v5;
	set final;
run;

data Results.sal_bproptanhearlyXn6v5;
	set sal_final;
```

```
run;

%cruzadabinarianeural(
        archivo=news reduced clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=7,
        algo=levmar,
        objetivo=tasafallos,
        early=40,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearlyXn7v5
);

data Results.levtanhearlyXn7v5;
        set final;
run;

data Results.sal_levtanhearlyXn7v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=7,
        algo=bprop,
        objetivo=tasafallos,
        early=185,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearlyXn7v5
);

data Results.bproptanhearlyXn7v5;
        set final;
run;

data Results.sal_bproptanhearlyXn7v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=8,
        algo=levmar,
        objetivo=tasafallos,
        early=30,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearlyXn8v5
);

data Results.levtanhearlyXn8v5;
        set final;
run;

data Results.sal_levtanhearlyXn8v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
```

```
        sfinal=12354,
        nodos=8,
        algo=bprop,
        objetivo=tasafallos,
        early=165,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearlyXn8v5
);

data Results.bproptanhearlyXn8v5;
        set final;
run;

data Results.sal_bproptanhearlyXn8v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=9,
        algo=levmar,
        objetivo=tasafallos,
        early=24,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearlyXn9v5
);

data Results.levtanhearlyXn9v5;
        set final;
run;

data Results.sal_levtanhearlyXn9v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=9,
        algo=bprop,
        objetivo=tasafallos,
        early=169,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearlyXn9v5
);

data Results.bproptanhearlyXn9v5;
        set final;
run;

data Results.sal_bproptanhearlyXn9v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=10,
        algo=levmar,
        objetivo=tasafallos,
        early=27,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearlyXn10v5
);
```

```
data Results.levtanhearlyXn10v5;
	set final;
run;

data Results.sal_levtanhearlyXn10v5;
	set sal_final;
run;

%cruzadabinarianeural(
	archivo=news_reduced_clean_five_methods,
	vardepen=REP_clicked,
	categor=REP_P31_head REP_P31_tail REP_subcategory,
	ngrupos=4,
	sinicio=12345,
	sfinal=12354,
	nodos=10,
	algo=bprop,
	objetivo=tasafallos,
	early=167,
	acti=tanh,
	directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearlyXn10v5
);

data Results.bproptanhearlyXn10v5;
	set final;
run;

data Results.sal_bproptanhearlyXn10v5;
	set sal_final;
run;

%cruzadabinarianeural(
	archivo=news_reduced_clean_five_methods,
	vardepen=REP_clicked,
	categor=REP_P31_head REP_P31_tail REP_subcategory,
	ngrupos=4,
	sinicio=12345,
	sfinal=12354,
	nodos=2,
	algo=levmar,
	objetivo=tasafallos,
	early=,
	acti=tanh,
	directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearly0n2v5
);

data Results.levtanhearly0n2v5;
	set final;
run;

data Results.sal_levtanhearly0n2v5;
	set sal_final;
run;

%cruzadabinarianeural(
	archivo=news_reduced_clean_five_methods,
	vardepen=REP_clicked,
	categor=REP_P31_head REP_P31_tail REP_subcategory,
	ngrupos=4,
	sinicio=12345,
	sfinal=12354,
	nodos=2,
	algo=bprop,
	objetivo=tasafallos,
	early=,
	acti=tanh,
	directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearly0n2v5
);

data Results.bproptanhearly0n2v5;
	set final;
run;

data Results.sal_bproptanhearly0n2v5;
	set sal_final;
run;
```

```
%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=3,
        algo=levmar,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearly0n3v5
);

data Results.levtanhearly0n3v5;
        set final;
run;

data Results.sal_levtanhearly0n3v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=3,
        algo=bprop,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearly0n3v5
);

data Results.bproptanhearly0n3v5;
        set final;
run;

data Results.sal_bproptanhearly0n3v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=4,
        algo=levmar,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearly0n4v5
);

data Results.levtanhearly0n4v5;
        set final;
run;

data Results.sal_levtanhearly0n4v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news reduced clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
```

```
        nodos=4,
        algo=bprop,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearly0n4v5
);

data Results.bproptanhearly0n4v5;
        set final;
run;

data Results.sal_bproptanhearly0n4v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=5,
        algo=levmar,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearly0n5v5
);

data Results.levtanhearly0n5v5;
        set final;
run;

data Results.sal_levtanhearly0n5v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=5,
        algo=bprop,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearly0n5v5
);

data Results.bproptanhearly0n5v5;
        set final;
run;

data Results.sal_bproptanhearly0n5v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=6,
        algo=levmar,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearly0n6v5
);
```

```
data Results.levtanhearly0n6v5;
	set final;
run;

data Results.sal_levtanhearly0n6v5;
	set sal_final;
run;

%cruzadabinarianeural(
	archivo=news reduced clean_five_methods,
	vardepen=REP_clicked,
	categor=REP_P31_head REP_P31_tail REP_subcategory,
	ngrupos=4,
	sinicio=12345,
	sfinal=12354,
	nodos=6,
	algo=bprop,
	objetivo=tasafallos,
	early=,
	acti=tanh,
	directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearly0n6v5
);

data Results.bproptanhearly0n6v5;
	set final;
run;

data Results.sal_bproptanhearly0n6v5;
	set sal_final;
run;

%cruzadabinarianeural(
	archivo=news_reduced_clean_five_methods,
	vardepen=REP_clicked,
	categor=REP_P31_head REP_P31_tail REP_subcategory,
	ngrupos=4,
	sinicio=12345,
	sfinal=12354,
	nodos=7,
	algo=levmar,
	objetivo=tasafallos,
	early=,
	acti=tanh,
	directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearly0n7v5
);

data Results.levtanhearly0n7v5;
	set final;
run;

data Results.sal_levtanhearly0n7v5;
	set sal_final;
run;

%cruzadabinarianeural(
	archivo=news_reduced_clean_five_methods,
	vardepen=REP_clicked,
	categor=REP_P31_head REP_P31_tail REP_subcategory,
	ngrupos=4,
	sinicio=12345,
	sfinal=12354,
	nodos=7,
	algo=bprop,
	objetivo=tasafallos,
	early=,
	acti=tanh,
	directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearly0n7v5
);

data Results.bproptanhearly0n7v5;
	set final;
run;

data Results.sal_bproptanhearly0n7v5;
	set sal_final;
run;
```

```
%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=8,
        algo=levmar,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearly0n8v5
);

data Results.levtanhearly0n8v5;
        set final;
run;

data Results.sal_levtanhearly0n8v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=8,
        algo=bprop,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearly0n8v5
);

data Results.bproptanhearly0n8v5;
        set final;
run;

data Results.sal_bproptanhearly0n8v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=9,
        algo=levmar,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearly0n9v5
);

data Results.levtanhearly0n9v5;
        set final;
run;

data Results.sal_levtanhearly0n9v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=9,
```

```
        algo=bprop,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearly0n9v5
);

data Results.bproptanhearly0n9v5;
        set final;
run;

data Results.sal_bproptanhearly0n9v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=10,
        algo=levmar,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\levtanhearly0n10v5
);

data Results.levtanhearly0n10v5;
        set final;
run;

data Results.sal_levtanhearly0n10v5;
        set sal_final;
run;

%cruzadabinarianeural(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        nodos=10,
        algo=bprop,
        objetivo=tasafallos,
        early=,
        acti=tanh,
        directorio=C:\Users\secci\Workspace\TFM\Base\5 methods\Tmp\bproptanhearly0n10v5
);

data Results.bproptanhearly0n10v5;
        set final;
run;

data Results.sal_bproptanhearly0n10v5;
        set sal_final;
run;

%cruzadatreeboostbin(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        leafsize=74,
        iteraciones=100,
        shrink=0.03,
        maxbranch=2,
        maxdepth=4,
        mincatsize=74,
        minobs=74,
        objetivo=tasafallos
);
```

```
data Results.gb003v5;
        set final;
run;

data Results.sal_gb003v5;
        set sal_final;
run;

%cruzadatreeboostbin(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        leafsize=74,
        iteraciones=100,
        shrink=0.1,
        maxbranch=2,
        maxdepth=4,
        mincatsize=74,
        minobs=74,
        objetivo=tasafallos
);

data Results.gb01v5;
        set final;
run;

data Results.sal_gb01v5;
        set sal_final;
run;

%cruzadatreeboostbin(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        leafsize=74,
        iteraciones=100,
        shrink=0.1,
        maxbranch=2,
        maxdepth=6,
        mincatsize=74,
        minobs=74,
        objetivo=tasafallos
);

data Results.gbd601v5;
        set final;
run;

data Results.sal_gbd601v5;
        set sal_final;
run;

%cruzadatreeboostbin(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        leafsize=74,
        iteraciones=100,
        shrink=0.001,
        maxbranch=2,
        maxdepth=4,
        mincatsize=74,
        minobs=74,
        objetivo=tasafallos
);

data Results.gb0001v5;
```

```
        set final;
run;

data Results.sal gb0001v5;
        set sal_final;
run;

%cruzadabaggingbin(
        archivo=news_reduced_clean_five_methods,
        vardepen=REP clicked,
        listcategor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        sinicio=12345,
        sfinal=12354,
        siniciobag=12345,
        sfinalbag=12354,
        porcenbag=0.80,
        maxbranch=2,
        nleaves=6,
        tamhoja=5,
        reemplazo=1,
        objetivo=tasafallos
);

data Results.bagn6m5v5;
        set final;
run;

data Results.sal_bagn6m5v5;
        set sal_final;
run;

%cruzadarandomforestbin(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        maxtrees=100,
        variables=6,
        porcenbag=0.80,
        maxbranch=2,
        tamhoja=5,
        maxdepth=10,
        pvalor=0.1,
        sinicio=12345,
        sfinal=12354,
        objetivo=tasafallos
);

data Results.rfnt100var6v5;
        set final;
run;

data Results.sal_rfnt100var6v5;
        set sal_final;
run;

%cruzadarandomforestbin(
        archivo=news_reduced_clean_five_methods,
        vardep=REP_clicked,
        categor=REP_P31_head REP_P31_tail REP_subcategory,
        ngrupos=4,
        maxtrees=200,
        variables=12,
        porcenbag=0.80,
        maxbranch=2,
        tamhoja=5,
        maxdepth=10,
        pvalor=0.1,
        sinicio=12345,
        sfinal=12354,
        objetivo=tasafallos
);

data Results.rfnt200var12v5;
        set final;
run;
```

```
data Results.sal_rfnt200var12v5;
       set sal_final;
run;
```

Annex 1.17: Two methods variable selection model training in R

```
# Logistic

logistic <- cruzadalogistica(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 12345,
  repe = 10
)

logistic$modelo = "ll-v2"
save.image("ll-v2.RData")

# avNNet

avnnetn2c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(2),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn2c01$modelo = "avnnet-n2-c0.1-v2"
save.image("avnnet-n2-c0.1-v2.RData")

avnnetn2c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(2),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn2c001$modelo = "avnnet-n2-c0.01-v2"
save.image("avnnet-n2-c0.01-v2.RData")

avnnetn3c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(3),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn3c01$modelo = "avnnet-n3-c0.1-v2"
save.image("avnnet-n3-c0.1-v2.RData")

avnnetn3c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
```

```
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(3),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn3c001$modelo = "avnnet-n3-c0.01-v2"
save.image("avnnet-n3-c0.01-v2.RData")

avnnetn4c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(4),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn4c01$modelo = "avnnet-n4-c0.1-v2"
save.image("avnnet-n4-c0.1-v2.RData")

avnnetn4c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(4),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn4c001$modelo = "avnnet-n4-c0.01-v2"
save.image("avnnet-n4-c0.01-v2.RData")

avnnetn5c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(5),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn5c01$modelo = "avnnet-n5-c0.1-v2"
save.image("avnnet-n5-c0.1-v2.RData")

avnnetn5c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(5),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn5c001$modelo = "avnnet-n5-c0.01-v2"
save.image("avnnet-n5-c0.01-v2.RData")

avnnetn6c01 <- cruzadaavnnetbin(
```

```
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(6),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn6c01$modelo = "avnnet-n6-c0.1-v2"
save.image("avnnet-n6-c0.1-v2.RData")

avnnetn6c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(6),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn6c001$modelo = "avnnet-n6-c0.01-v2"
save.image("avnnet-n6-c0.01-v2.RData")

avnnetn7c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(7),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn7c01$modelo = "avnnet-n7-c0.1-v2"
save.image("avnnet-n7-c0.1-v2.RData")

avnnetn7c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(7),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn7c001$modelo = "avnnet-n7-c0.01-v2"
save.image("avnnet-n7-c0.01-v2.RData")

avnnetn8c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(8),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn8c01$modelo = "avnnet-n8-c0.1-v2"
```

```
save.image("avnnet-n8-c0.1-v2.RData")

avnnetn8c001 <- cruzadaavnnetbin(
  data = news reduced clean two methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(8),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn8c001$modelo = "avnnet-n8-c0.01-v2"
save.image("avnnet-n8-c0.01-v2.RData")

avnnetn9c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(9),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn9c01$modelo = "avnnet-n9-c0.1-v2"
save.image("avnnet-n9-c0.1-v2.RData")

avnnetn9c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(9),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn9c001$modelo = "avnnet-n9-c0.01-v2"
save.image("avnnet-n9-c0.01-v2.RData")

avnnetn10c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(10),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn10c01$modelo = "avnnet-n10-c0.1-v2"
save.image("avnnet-n10-c0.1-v2.RData")

avnnetn10c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(10),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
```

```
)

avnnetn10c001$modelo = "avnnet-n10-c0.01-v2"
save.image("avnnet-n10-c0.01-v2.RData")

arbol <- cruzadaarbolbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  cp = c(0),
  minbucket = 74
)

arbol$modelo = "arbol-v2"
save.image("arbol-v2.RData")

bagging <- cruzadarfbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  nodesize = 74,
  mtry = 50,
  ntree = 50,
  replace = TRUE
)

bagging$modelo = "bagging-v2"
save.image("bagging-v2.RData")

randomforest <- cruzadarfbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  nodesize = 74,
  mtry = 3,
  ntree = 50,
  replace = TRUE
)

randomforest$modelo = "rf-v2"
save.image("rf-v2.RData")

randomforestm2 <- cruzadarfbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  nodesize = 74,
  mtry = 2,
  ntree = 50,
  replace = TRUE
)

randomforestm2$modelo = "rf-m2-v2"
save.image("rf-m2-v2.RData")

gradientboosting <- cruzadagbmbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  n.minobsinnode = 74,
  shrinkage = 0.008,
  n.trees = 250,
```

```
  interaction.depth = 2
)

gradientboosting$modelo = "gbm-v2"
save.image("gbm-v2.RData")


gradientboosting01d4 <- cruzadagbmbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  n.minobsinnode = 74,
  shrinkage = 0.1,
  n.trees = 250,
  interaction.depth = 4
)

gradientboosting01d4$modelo = "gbm-01-d4-v2"
save.image("gbm-01-d4-v2.RData")

gradientboosting003d4 <- cruzadagbmbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  n.minobsinnode = 74,
  shrinkage = 0.03,
  n.trees = 250,
  interaction.depth = 4
)

gradientboosting003d4$modelo = "gbm-003-d4-v2"
save.image("gbm-003-d4-v2.RData")

xgboost <- cruzadaxgbmbin(
  data = news_reduced_clean_two_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  min_child_weight = 74,
  eta = 0.001,
  nrounds = 1250,
  max_depth = 6,
  gamma = 0,
  colsample_bytree = 1,
  subsample = 1,
  alpha = 0,
  lambda = 0,
  lambda_bias = 0
)

xgboost$modelo = "xgbm-v2"
save.image("xgbm-v2.RData")

library(h2o)

h2o.init()
data.hex <- as.h2o(news_reduced_clean_two_methods, destination_frame = "data.hex")
source("../cruzada gbm encode.R")

gbmencode_nt200_d6_nr74_nb100 <- cruzadagbmencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  ntrees = 200,
  max_depth = 6,
  min_rows = 74,
```

```
  nbins_cats = 100
)

gbmencode_nt200_d6_nr74_nb100$modelo = "gbmencode-nt200-d6-nr74-nb100-v2"
save.image("gbmencode-nt200-d6-nr74-nb100-v2.RData")

gbmencode_nt400_d6_nr74_nb1024 <- cruzadagbmencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  ntrees = 400,
  max_depth = 6,
  min_rows = 74,
  nbins_cats = 1024
)

gbmencode_nt400_d6_nr74_nb1024$modelo = "gbmencode-nt400-d6-nr74-nb1024-v2"
save.image("gbmencode-nt400-d6-nr74-nb1024-v2.RData")

gbmencode_nt200_d3_nr148_nb100 <- cruzadagbmencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  ntrees = 200,
  max_depth = 3,
  min_rows = 148,
  nbins_cats = 100
)

gbmencode_nt200_d3_nr148_nb100$modelo = "gbmencode-nt200-d3-nr148-nb100-v2"
save.image("gbmencode-nt200-d3-nr148-nb100-v2.RData")

gbmencode_nt400_d3_nr148_nb1024 <- cruzadagbmencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  ntrees = 400,
  max_depth = 3,
  min_rows = 148,
  nbins_cats = 1024
)

gbmencode_nt400_d3_nr148_nb1024$modelo = "gbmencode-nt400-d3-nr148-nb1024-v2"
save.image("gbmencode-nt400-d3-nr148-nb1024-v2.RData")

h2o.init()
data.hex <- as.h2o(news_reduced_clean_two_methods, destination_frame = "data.hex")
source("../cruzada rf encode.R")

rfencode_nt200_d6_nr74_mtries_minus1_nb100 <- cruzadarfencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  ntrees = 200,
  max_depth = 6,
  min_rows = 74,
  mtries = -1,
  nbins_cats = 100
)

rfencode_nt200_d6_nr74_mtries_minus1_nb100$modelo = "rfencode-nt200-d6-nr74-mtries-
minus1-nb100-v2"
save.image("rfencode-nt200-d6-nr74-mtries-minus1-nb100-v2.RData")

rfencode_nt400_d6_nr74_mtries_minus1_nb1024 <- cruzadarfencode(
```

```
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  ntrees = 400,
  max_depth = 6,
  min_rows = 74,
  mtries = -1,
  nbins_cats = 1024
)

rfencode_nt400_d6_nr74_mtries_minus1_nb1024$modelo = "rfencode-nt400-d6-nr74-mtries-
minus1-nb1024-v2"
save.image("rfencode-nt400-d6-nr74-mtries-minus1-nb1024-v2.RData")

rfencode_nt200_d3_nr148_mtries_minus1_nb100 <- cruzadarfencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  ntrees = 200,
  max_depth = 3,
  min_rows = 148,
  mtries = -1,
  nbins_cats = 100
)

rfencode_nt200_d3_nr148_mtries_minus1_nb100$modelo = "rfencode-nt200-d3-nr148-mtries-
minus1-nb100-v2"
save.image("rfencode-nt200-d3-nr148-mtries-minus1-nb100-v2.RData")

rfencode_nt400_d3_nr148_mtries_minus1_nb1024 <- cruzadarfencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  ntrees = 400,
  max_depth = 3,
  min_rows = 148,
  mtries = -1,
  nbins_cats = 1024
)

rfencode_nt400_d3_nr148_mtries_minus1_nb1024$modelo = "rfencode-nt400-d3-nr148-mtries-
minus1-nb1024-v2"
save.image("rfencode-nt400-d3-nr148-mtries-minus1-nb1024-v2.RData")

h2o.init()
data.hex <- as.h2o(news_reduced_clean_two_methods, destination_frame = "data.hex")
source("../cruzada rf encode.R")

baggingencode_nt200_d6_nr74_nb100 <- cruzadarfencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  ntrees = 200,
  max depth = 6,
  min_rows = 74,
  mtries = 50,
  nbins_cats = 100
)

baggingencode nt200 d6 nr74 nb100$modelo = "baggingencode-nt200-d6-nr74-nb100-v2"
save.image("baggingencode-nt200-d6-nr74-nb100-v2.RData")

baggingencode_nt400_d6_nr74_nb1024 <- cruzadarfencode(
  data = data.hex,
  vardep = vardep,
```

```
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  ntrees = 400,
  max_depth = 6,
  min_rows = 74,
  mtries = 50,
  nbins_cats = 1024
)

baggingencode_nt400_d6_nr74_nb1024$modelo = "baggingencode-nt400-d6-nr74-nb1024-v2"
save.image("baggingencode-nt400-d6-nr74-nb1024-v2.RData")

baggingencode_nt200_d3_nr148_nb100 <- cruzadarfencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  ntrees = 200,
  max_depth = 3,
  min_rows = 148,
  mtries = -1,
  nbins_cats = 100
)

baggingencode_nt200_d3_nr148_nb100$modelo = "baggingencode-nt200-d3-nr148-nb100-v2"
save.image("baggingencode-nt200-d3-nr148-nb100-v2.RData")

baggingencode_nt400_d3_nr148_nb1024 <- cruzadarfencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  ntrees = 400,
  max_depth = 3,
  min_rows = 148,
  mtries = -1,
  nbins_cats = 1024
)

baggingencode_nt400_d3_nr148_nb1024$modelo = "baggingencode-nt400-d3-nr148-nb1024-v2"
save.image("baggingencode-nt400-d3-nr148-nb1024-v2.RData")

h2o.init()
data.hex <- as.h2o(news_reduced_clean_two_methods, destination_frame = "data.hex")
source("../cruzada neural encode.R")

neuralencode_h10_tanh_l2_0001_e100 <- cruzadaneuralencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  activation = "Tanh",
  hidden = c(10),
  l2 = 0.001,
  epochs = 100
)

neuralencode_h10_tanh_l2_0001_e100$modelo = "neuralencode-h10-tanh-l2-0001-e100-v2"
save.image("neuralencode-h10-tanh-l2-0001-e100-v2.RData")

neuralencode_h10_recti_l2_0001_e100 <- cruzadaneuralencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  activation = "Rectifier",
  hidden = c(10),
```

```
  l2 = 0.001,
  epochs = 100
)

neuralencode_h10_recti_l2_0001_e100$modelo = "neuralencode-h10-recti-l2-0001-e100-v2"
save.image("neuralencode-h10-recti-l2-0001-e100-v2.RData")

neuralencode_h10_maxout_l2_0001_e100 <- cruzadaneuralencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  activation = "Maxout",
  hidden = c(10),
  l2 = 0.001,
  epochs = 100
)

neuralencode_h10_maxout_l2_0001_e100$modelo = "neuralencode-h10-maxout-l2-0001-e100-v2"
save.image("neuralencode-h10-maxout-l2-0001-e100-v2.RData")

h2o.init()
data.hex <- as.h2o(news_reduced_clean_two_methods, destination_frame = "data.hex")
source("../cruzada log binaria encode.R")

logbinaria_alliter_l0001 <- cruzadalogencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  lambda = 0.001,
  interactions = TRUE
)

logbinaria_alliter_l0001$modelo = "logbinaria-alliter-l0001-v2"
save.image("logbinaria-alliter-l0001-v2.RData")

logbinaria_noiter_l0001 <- cruzadalogencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  lambda = 0.001,
  interactions = FALSE
)

logbinaria_noiter_l0001$modelo = "logbinaria-noiter-l0001-v2"
save.image("logbinaria-noiter-l0001-v2.RData")

logbinaria_alliter_l001 <- cruzadalogencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  lambda = 0.01,
  interactions = TRUE
)

logbinaria_alliter_l001$modelo = "logbinaria-alliter-l001-v2"
save.image("logbinaria-alliter-l001-v2.RData")

logbinaria_noiter_l001 <- cruzadalogencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  lambda = 0.01,
```

```
  interactions = FALSE
)

logbinaria_noiter_l001$modelo = "logbinaria-noiter-l001-v2"
save.image("logbinaria-noiter-l001-v2.RData")

logbinaria_alliter_l01 <- cruzadalogencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  lambda = 0.1,
  interactions = TRUE
)

logbinaria_alliter_l01$modelo = "logbinaria-alliter-l01-v2"
save.image("logbinaria-alliter-l01-v2.RData")

logbinaria_noiter_l01 <- cruzadalogencode(
  data = data.hex,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  lambda = 0.1,
  interactions = FALSE
)

logbinaria_noiter_l01$modelo = "logbinaria-noiter-l01-v2"
save.image("logbinaria-noiter-l01-v2.RData")
```

Annex 1.18: Three methods variable selection model training in R

```
# Logistic

logistic <- cruzadalogistica(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 12345,
  repe = 10
)

logistic$modelo = "ll-v3"
save.image("ll-v3.RData")

# avNNet

avnnetn2c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(2),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn2c01$modelo = "avnnet-n2-c0.1-v3"
save.image("avnnet-n2-c0.1-v3.RData")

avnnetn2c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
```

```
  size = c(2),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn2c001$modelo = "avnnet-n2-c0.01-v3"
save.image("avnnet-n2-c0.01-v3.RData")

avnnetn3c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(3),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn3c01$modelo = "avnnet-n3-c0.1-v3"
save.image("avnnet-n3-c0.1-v3.RData")

avnnetn3c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(3),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn3c001$modelo = "avnnet-n3-c0.01-v3"
save.image("avnnet-n3-c0.01-v3.RData")

avnnetn4c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(4),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn4c01$modelo = "avnnet-n4-c0.1-v3"
save.image("avnnet-n4-c0.1-v3.RData")

avnnetn4c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(4),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn4c001$modelo = "avnnet-n4-c0.01-v3"
save.image("avnnet-n4-c0.01-v3.RData")

avnnetn5c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
```

```
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(5),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn5c01$modelo = "avnnet-n5-c0.1-v3"
save.image("avnnet-n5-c0.1-v3.RData")

avnnetn5c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(5),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn5c001$modelo = "avnnet-n5-c0.01-v3"
save.image("avnnet-n5-c0.01-v3.RData")

avnnetn6c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(6),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn6c01$modelo = "avnnet-n6-c0.1-v3"
save.image("avnnet-n6-c0.1-v3.RData")

avnnetn6c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(6),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn6c001$modelo = "avnnet-n6-c0.01-v3"
save.image("avnnet-n6-c0.01-v3.RData")

avnnetn7c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(7),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn7c01$modelo = "avnnet-n7-c0.1-v3"
save.image("avnnet-n7-c0.1-v3.RData")

avnnetn7c001 <- cruzadaavnnetbin(
```

```
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(7),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn7c001$modelo = "avnnet-n7-c0.01-v3"
save.image("avnnet-n7-c0.01-v3.RData")

avnnetn8c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(8),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn8c01$modelo = "avnnet-n8-c0.1-v3"
save.image("avnnet-n8-c0.1-v3.RData")

avnnetn8c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(8),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn8c001$modelo = "avnnet-n8-c0.01-v3"
save.image("avnnet-n8-c0.01-v3.RData")

avnnetn9c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(9),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn9c01$modelo = "avnnet-n9-c0.1-v3"
save.image("avnnet-n9-c0.1-v3.RData")

avnnetn9c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(9),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn9c001$modelo = "avnnet-n9-c0.01-v3"
```

```
save.image("avnnet-n9-c0.01-v3.RData")

avnnetn10c01 <- cruzadaavnnetbin(
  data = news reduced clean three methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(10),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn10c01$modelo = "avnnet-n10-c0.1-v3"
save.image("avnnet-n10-c0.1-v3.RData")

avnnetn10c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(10),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn10c001$modelo = "avnnet-n10-c0.01-v3"
save.image("avnnet-n10-c0.01-v3.RData")

arbol <- cruzadaarbolbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  cp = c(0),
  minbucket = 74
)

arbol$modelo = "arbol-v3"
save.image("arbol-v3.RData")

bagging <- cruzadarfbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  nodesize = 74,
  mtry = 22,
  ntree = 50,
  replace = TRUE
)

bagging$modelo = "bagging-v3"
save.image("bagging-v3.RData")

randomforest <- cruzadarfbin(
  data = news reduced clean three methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  nodesize = 74,
  mtry = 3,
  ntree = 50,
  replace = TRUE
)
```

```
randomforest$modelo = "rf-v3"
save.image("rf-v3.RData")

randomforestm2 <- cruzadarfbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  nodesize = 74,
  mtry = 2,
  ntree = 50,
  replace = TRUE
)

randomforestm2$modelo = "rf-m2-v3"
save.image("rf-m2-v3.RData")

gradientboosting <- cruzadagbmbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  n.minobsinnode = 74,
  shrinkage = 0.008,
  n.trees = 250,
  interaction.depth = 2
)

gradientboosting$modelo = "gbm-v3"
save.image("gbm-v3.RData")


gradientboosting01d4 <- cruzadagbmbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  n.minobsinnode = 74,
  shrinkage = 0.1,
  n.trees = 250,
  interaction.depth = 4
)

gradientboosting01d4$modelo = "gbm-01-d4-v3"
save.image("gbm-01-d4-v3.RData")

gradientboosting003d4 <- cruzadagbmbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  n.minobsinnode = 74,
  shrinkage = 0.03,
  n.trees = 250,
  interaction.depth = 4
)

gradientboosting003d4$modelo = "gbm-003-d4-v3"
save.image("gbm-003-d4-v3.RData")

xgboost <- cruzadaxgbmbin(
  data = news_reduced_clean_three_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  min_child_weight = 74,
  eta = 0.001,
```

```
  nrounds = 1250,
  max_depth = 6,
  gamma = 0,
  colsample bytree = 1,
  subsample = 1,
  alpha = 0,
  lambda = 0,
  lambda_bias = 0
)

xgboost$modelo = "xgbm-v3"
save.image("xgbm-v3.RData")
```

Annex 1.19: Four methods variable selection model training in R

```
# Logistic

logistic <- cruzadalogistica(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 12345,
  repe = 10
)

logistic$modelo = "ll-v4"
save.image("ll-v4.RData")

# avNNet

avnnetn2c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(2),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn2c01$modelo = "avnnet-n2-c0.1-v4"
save.image("avnnet-n2-c0.1-v4.RData")

avnnetn2c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(2),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn2c001$modelo = "avnnet-n2-c0.01-v4"
save.image("avnnet-n2-c0.01-v4.RData")

avnnetn3c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(3),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)
```

```r
avnnetn3c01$modelo = "avnnet-n3-c0.1-v4"
save.image("avnnet-n3-c0.1-v4.RData")

avnnetn3c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(3),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn3c001$modelo = "avnnet-n3-c0.01-v4"
save.image("avnnet-n3-c0.01-v4.RData")

avnnetn4c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(4),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn4c01$modelo = "avnnet-n4-c0.1-v4"
save.image("avnnet-n4-c0.1-v4.RData")

avnnetn4c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(4),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn4c001$modelo = "avnnet-n4-c0.01-v4"
save.image("avnnet-n4-c0.01-v4.RData")

avnnetn5c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(5),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn5c01$modelo = "avnnet-n5-c0.1-v4"
save.image("avnnet-n5-c0.1-v4.RData")

avnnetn5c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(5),
  decay = c(0.01),
```

```
  repeticiones = 5,
  itera = 200
)

avnnetn5c001$modelo = "avnnet-n5-c0.01-v4"
save.image("avnnet-n5-c0.01-v4.RData")

avnnetn6c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(6),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn6c01$modelo = "avnnet-n6-c0.1-v4"
save.image("avnnet-n6-c0.1-v4.RData")

avnnetn6c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(6),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn6c001$modelo = "avnnet-n6-c0.01-v4"
save.image("avnnet-n6-c0.01-v4.RData")

avnnetn7c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(7),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn7c01$modelo = "avnnet-n7-c0.1-v4"
save.image("avnnet-n7-c0.1-v4.RData")

avnnetn7c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(7),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn7c001$modelo = "avnnet-n7-c0.01-v4"
save.image("avnnet-n7-c0.01-v4.RData")

avnnetn8c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
```

```
  repe = 10,
  size = c(8),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn8c01$modelo = "avnnet-n8-c0.1-v4"
save.image("avnnet-n8-c0.1-v4.RData")

avnnetn8c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(8),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn8c001$modelo = "avnnet-n8-c0.01-v4"
save.image("avnnet-n8-c0.01-v4.RData")

avnnetn9c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(9),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn9c01$modelo = "avnnet-n9-c0.1-v4"
save.image("avnnet-n9-c0.1-v4.RData")

avnnetn9c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(9),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn9c001$modelo = "avnnet-n9-c0.01-v4"
save.image("avnnet-n9-c0.01-v4.RData")

avnnetn10c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(10),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn10c01$modelo = "avnnet-n10-c0.1-v4"
save.image("avnnet-n10-c0.1-v4.RData")

avnnetn10c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
```

```
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(10),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn10c001$modelo = "avnnet-n10-c0.01-v4"
save.image("avnnet-n10-c0.01-v4.RData")

arbol <- cruzadaarbolbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  cp = c(0),
  minbucket = 74
)

arbol$modelo = "arbol-v4"
save.image("arbol-v4.RData")

bagging <- cruzadarfbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  nodesize = 74,
  mtry = 8,
  ntree = 50,
  replace = TRUE
)

bagging$modelo = "bagging-v4"
save.image("bagging-v4.RData")

randomforest <- cruzadarfbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  nodesize = 74,
  mtry = 3,
  ntree = 50,
  replace = TRUE
)

randomforest$modelo = "rf-v4"
save.image("rf-v4.RData")

randomforestm2 <- cruzadarfbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  nodesize = 74,
  mtry = 2,
  ntree = 50,
  replace = TRUE
)

randomforestm2$modelo = "rf-m2-v4"
save.image("rf-m2-v4.RData")

gradientboosting <- cruzadagbmbin(
  data = news_reduced_clean_four_methods,
```

```
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  n.minobsinnode = 74,
  shrinkage = 0.008,
  n.trees = 250,
  interaction.depth = 2
)

gradientboosting$modelo = "gbm-v4"
save.image("gbm-v4.RData")


gradientboosting01d4 <- cruzadagbmbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  n.minobsinnode = 74,
  shrinkage = 0.1,
  n.trees = 250,
  interaction.depth = 4
)

gradientboosting01d4$modelo = "gbm-01-d4-v4"
save.image("gbm-01-d4-v4.RData")

gradientboosting003d4 <- cruzadagbmbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  n.minobsinnode = 74,
  shrinkage = 0.03,
  n.trees = 250,
  interaction.depth = 4
)

gradientboosting003d4$modelo = "gbm-003-d4-v4"
save.image("gbm-031-d4-v4.RData")

xgboost <- cruzadaxgbmbin(
  data = news_reduced_clean_four_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  min_child_weight = 74,
  eta = 0.001,
  nrounds = 1250,
  max_depth = 6,
  gamma = 0,
  colsample_bytree = 1,
  subsample = 1,
  alpha = 0,
  lambda = 0,
  lambda_bias = 0
)

xgboost$modelo = "xgbm-v4"
save.image("xgbm-v4.RData")
```

Annex 1.20: Five methods variable selection model training in R

```
# Logistic

logistic <- cruzadalogistica(
  data = news_reduced_clean_five_methods,
```

```
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 12345,
  repe = 10
)

logistic$modelo = "ll-v5"
save.image("ll-v5.RData")

# avNNet

avnnetn2c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(2),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn2c01$modelo = "avnnet-n2-c0.1-v5"
save.image("avnnet-n2-c0.1-v5.RData")

avnnetn2c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(2),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn2c001$modelo = "avnnet-n2-c0.01-v5"
save.image("avnnet-n2-c0.01-v5.RData")

avnnetn3c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(3),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn3c01$modelo = "avnnet-n3-c0.1-v5"
save.image("avnnet-n3-c0.1-v5.RData")

avnnetn3c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(3),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn3c001$modelo = "avnnet-n3-c0.01-v5"
save.image("avnnet-n3-c0.01-v5.RData")

avnnetn4c01 <- cruzadaavnnetbin(
```

```
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(4),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn4c01$modelo = "avnnet-n4-c0.1-v5"
save.image("avnnet-n4-c0.1-v5.RData")

avnnetn4c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(4),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn4c001$modelo = "avnnet-n4-c0.01-v5"
save.image("avnnet-n4-c0.01-v5.RData")

avnnetn5c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(5),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn5c01$modelo = "avnnet-n5-c0.1-v5"
save.image("avnnet-n5-c0.1-v5.RData")

avnnetn5c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(5),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn5c001$modelo = "avnnet-n5-c0.01-v5"
save.image("avnnet-n5-c0.01-v5.RData")

avnnetn6c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(6),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn6c01$modelo = "avnnet-n6-c0.1-v5"
```

```
save.image("avnnet-n6-c0.1-v5.RData")

avnnetn6c001 <- cruzadaavnnetbin(
  data = news reduced clean five methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(6),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn6c001$modelo = "avnnet-n6-c0.01-v5"
save.image("avnnet-n6-c0.01-v5.RData")

avnnetn7c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(7),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn7c01$modelo = "avnnet-n7-c0.1-v5"
save.image("avnnet-n7-c0.1-v5.RData")

avnnetn7c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(7),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn7c001$modelo = "avnnet-n7-c0.01-v5"
save.image("avnnet-n7-c0.01-v5.RData")

avnnetn8c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(8),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn8c01$modelo = "avnnet-n8-c0.1-v5"
save.image("avnnet-n8-c0.1-v5.RData")

avnnetn8c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(8),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
```

```
)

avnnetn8c001$modelo = "avnnet-n8-c0.01-v5"
save.image("avnnet-n8-c0.01-v5.RData")

avnnetn9c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(9),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn9c01$modelo = "avnnet-n9-c0.1-v5"
save.image("avnnet-n9-c0.1-v5.RData")

avnnetn9c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(9),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn9c001$modelo = "avnnet-n9-c0.01-v5"
save.image("avnnet-n9-c0.01-v5.RData")

avnnetn10c01 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(10),
  decay = c(0.1),
  repeticiones = 5,
  itera = 200
)

avnnetn10c01$modelo = "avnnet-n10-c0.1-v5"
save.image("avnnet-n10-c0.1-v5.RData")

avnnetn10c001 <- cruzadaavnnetbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  size = c(10),
  decay = c(0.01),
  repeticiones = 5,
  itera = 200
)

avnnetn10c001$modelo = "avnnet-n10-c0.01-v5"
save.image("avnnet-n10-c0.01-v5.RData")

arbol <- cruzadaarbolbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  cp = c(0),
```

```
  minbucket = 74
)

arbol$modelo = "arbol-v5"
save.image("arbol-v5.RData")

bagging <- cruzadarfbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  nodesize = 74,
  mtry = 3,
  ntree = 50,
  replace = TRUE
)

bagging$modelo = "bagging-v5"
save.image("bagging-v5.RData")

randomforest <- cruzadarfbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  nodesize = 74,
  mtry = 3,
  ntree = 50,
  replace = TRUE
)

randomforest$modelo = "rf-v5"
save.image("rf-v5.RData")

randomforestm2 <- cruzadarfbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  nodesize = 74,
  mtry = 2,
  ntree = 50,
  replace = TRUE
)

randomforestm2$modelo = "rf-m2-v5"
save.image("rf-m2-v5.RData")

gradientboosting <- cruzadagbmbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  n.minobsinnode = 74,
  shrinkage = 0.008,
  n.trees = 250,
  interaction.depth = 2
)

gradientboosting$modelo = "gbm-v5"
save.image("gbm-v5.RData")


gradientboosting01d4 <- cruzadagbmbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
```

```
  repe = 10,
  n.minobsinnode = 74,
  shrinkage = 0.1,
  n.trees = 250,
  interaction.depth = 4
)

gradientboosting01d4$modelo = "gbm-01-d4-v5"
save.image("gbm-01-d4-v5.RData")

gradientboosting003d4 <- cruzadagbmbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  n.minobsinnode = 74,
  shrinkage = 0.03,
  n.trees = 250,
  interaction.depth = 4
)

gradientboosting003d4$modelo = "gbm-003-d4-v5"
save.image("gbm-003-d4-v5.RData")

xgboost <- cruzadaxgbmbin(
  data = news_reduced_clean_five_methods,
  vardep = vardep,
  listclass = categoricas,
  grupos = 4,
  sinicio = 1234,
  repe = 10,
  min_child_weight = 74,
  eta = 0.001,
  nrounds = 1250,
  max_depth = 6,
  gamma = 0,
  colsample_bytree = 1,
  subsample = 1,
  alpha = 0,
  lambda = 0,
  lambda_bias = 0
)

xgboost$modelo = "xgbm-v5"
save.image("xgbm-v5.RData")
```

Annex 1.21: AutoML Implementation in R

```
splits <-
  h2o.splitFrame(data.hex,
                 seed = seed,
                 ratios = c(0.8))

train <- splits[[1]]
test <- splits[[2]]

y <- "REP_clicked"
x <- setdiff(names(train), y)

train$fold <-
  h2o.kfold_column(train, nfolds = nfolds, seed = seed)

aml <- h2o.automl(
  x = x,
  y = y,
  include_algos = c("GBM", "DeepLearning", "DRF"),
  training_frame = train,
  fold_column = "fold",
  seed = seed,
  max_runtime_secs = 36000
)

lb <- h2o.get_leaderboard(object = aml, extra_columns = 'ALL')
```

```
print(lb, n = nrow(lb))
```

Code 15 AutoML Implementation

```
result <- data.frame(matrix(ncol = 3, nrow = 0))
colnames(result) <- c("model", "AUC", "seed")

best_automl_params <- data.frame(
  ntrees = c(
    61,
    70,
    77,
    73,
    75,
    96,
    65,
    70,
    60,
    70
  ),
  max_depth = c(
    17,
    16,
    15,
    17,
    16,
    14,
    17,
    14,
    16,
    15
  )
)


for (j in 1:nrow(best_automl_params)) {
  for (seed in sinicio:sfin) {
    splits <-
      h2o.splitFrame(data.hex,
                     seed = seed,
                     ratios = c(0.8))

    train <- splits[[1]]
    test <- splits[[2]]

    y <- "REP_clicked"
    x <- setdiff(names(train), y)

    train$fold <-
      h2o.kfold_column(train, nfolds = nfolds, seed = seed)

    rcv_gbm <- h2o.gbm(
      x = x,
      y = y,
      training_frame = train,
      fold_column = "fold",
      seed = seed,
      ntrees = best_automl_params[j, 'ntrees'],
      max_depth = best_automl_params[j, 'max_depth'],
      learn_rate = 0.1
    )

    best_gbm_perf1 <- h2o.performance(model = rcv_gbm,
                                      newdata = test)
    rcv_auc <- h2o.auc(best_gbm_perf1)
    result[nrow(result) + 1, ] <-
      c(paste("GBM_", best_automl_params[j, 'ntrees'], "_", best_automl_params[j,
'max_depth'], "_0.1_V2", sep = ""),
        rcv_auc,
        seed)
    print(result)
  }
}
```

Code 16 Best 10 AutoML Models Variance Study

Annex 1.22: SAS Base scoring information transformation into AUC CSV files

```
import os
import re
import sys
import glob
import pandas

from sklearn import metrics
from sklearn.metrics import roc_auc_score


def rf_auc(name, model, directory):
    df = pandas.read_sas(name)
    df = df.dropna(subset=['F_vardep', 'P_vardep1'])
    group_size = 55837
    with open(f'{directory}{os.path.sep}auc{os.path.sep}{model}.csv', 'w') as f:
        for k in range(0, len(df) - group_size, group_size):
            y = df['F_vardep'][k:k + group_size].astype(float)
            pred = df['P_vardep1'][k:k + group_size]
            auc = roc_auc_score(y, pred)
            f.write(f'{model}; {auc}\n')


def neural_auc(name, model, directory):
    df = pandas.read_sas(name)
    df = df.dropna(subset=['REP_clicked', 'P_REP_clicked1'])
    group = df.iloc[0]['grupo']
    i = 0
    with open(f'{directory}{os.path.sep}auc{os.path.sep}{model}.csv', 'w') as f:
        for index, row in df.iterrows():
            if group != row['grupo']:
                y = df['REP_clicked'][i:index]
                pred = df['P_REP_clicked1'][i:index]
                fpr, tpr, thresholds = metrics.roc_curve(y, pred, pos_label=1)
                auc = metrics.auc(fpr, tpr)
                f.write(f'{model}; {auc}\n')
                i = index
                group = row['grupo']


def logistic_auc(name, model, directory):
    df = pandas.read_sas(name)
    df = df.dropna(subset=['REP_clicked', 'predi'])
    group = df.iloc[0]['grupo']
    i = 0
    with open(f'{directory}{os.path.sep}auc{os.path.sep}{model}.csv', 'w') as f:
        for index, row in df.iterrows():
            if group != row['grupo']:
                y = df['REP_clicked'][i:index]
                pred = df['predi'][i:index]
                fpr, tpr, thresholds = metrics.roc_curve(y, pred, pos_label=0)
                auc = metrics.auc(fpr, tpr)
                f.write(f'{model}; {auc}\n')
                i = index
                group = row['grupo']


def gb_auc(name, model, directory):
    df = pandas.read_sas(name)
    df = df.dropna(subset=['REP_clicked', 'P_vardep1'])
    group = df.iloc[0]['grupo']
    i = 0
    with open(f'{directory}{os.path.sep}auc{os.path.sep}{model}.csv', 'w') as f:
        for index, row in df.iterrows():
            if group != row['grupo']:
                y = df['REP_clicked'][i:index]
                pred = df['P_vardep1'][i:index]
                fpr, tpr, thresholds = metrics.roc_curve(y, pred, pos_label=1)
                auc = metrics.auc(fpr, tpr)
                f.write(f'{model}; {auc}\n')
                i = index
                group = row['grupo']
```

```
if __name__ == '__main__':
    if len(sys.argv) != 2:
        print('Usage: python build_auc_from_sas7bdat.py directory')
    directory = sys.argv[1]

    for name in glob.glob(f'{directory}{os.path.sep}sal_bproptanhearly*.sas7bdat'):
        x = re.search(r"bproptanhearly[^.]+", name)
        model = x.group(0)
        neural_auc(name, model, directory)

    for name in glob.glob(f'{directory}{os.path.sep}sal_levtanhearly*.sas7bdat'):
        x = re.search(r"levtanhearly[^.]+", name)
        model = x.group(0)
        neural_auc(name, model, directory)

    for name in glob.glob(f'{directory}{os.path.sep}sal_ll*.sas7bdat'):
        x = re.search(r"ll[^.]+", name)
        model = x.group(0)
        logistic_auc(name, model, directory)

    for name in glob.glob(f'{directory}{os.path.sep}sal_rfnt*.sas7bdat'):
        x = re.search(r"rfnt[^.]+", name)
        model = x.group(0)
        rf_auc(name, model, directory)

    for name in glob.glob(f'{directory}{os.path.sep}sal_gb*.sas7bdat'):
        x = re.search(r"gb[^.]+", name)
        model = x.group(0)
        gb_auc(name, model, directory)

    for name in glob.glob(f'{directory}{os.path.sep}sal_bag*.sas7bdat'):
        x = re.search(r"bag[^.]+", name)
        model = x.group(0)
        gb_auc(name, model, directory)
```

Annex 1.23: AUC Boxplot chart for tabular models

```
from os import path
import pandas as pd
import glob
import matplotlib.pyplot as plt


# v2
df = None
for result_file in
glob.glob(f'..{path.sep}results{path.sep}Base{path.sep}all{path.sep}*v2.csv'):
    if df is None:
        df = pd.read_csv(result_file, sep=';', names=['modelo', 'auc'])
    else:
        df = pd.concat([df, pd.read_csv(result_file, sep=';', names=['modelo', 'auc'])],
ignore_index=True)
ax = df.boxplot(column=['auc'], by=['modelo'], rot=45, fontsize=10)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("SAS Base Software V2 Variable Selection Models")
plt.suptitle('')
plt.show()

df_best = df[df['auc'] > 0.9]
df_best = df[df['modelo'].isin(df_best['modelo'])]
ax = df_best.boxplot(column=['auc'], by=['modelo'], rot=45, fontsize=10)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("SAS Base Software V2 Variable Selection Models with AUC over 0.9")
plt.suptitle('')
plt.show()

# v3
df = None
for result_file in
glob.glob(f'..{path.sep}results{path.sep}Base{path.sep}all{path.sep}*v3.csv'):
    if df is None:
        df = pd.read_csv(result_file, sep=';', names=['modelo', 'auc'])
```

```
    else:
        df = pd.concat([df, pd.read_csv(result_file, sep=';', names=['modelo', 'auc'])],
ignore_index=True)
ax = df.boxplot(column=['auc'], by=['modelo'], rot=45, fontsize=10)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("SAS Base Software V3 Variable Selection Models")
plt.suptitle('')
plt.show()

df_best = df[df['auc'] > 0.9]
df_best = df[df['modelo'].isin(df_best['modelo'])]
ax = df_best.boxplot(column=['auc'], by=['modelo'], rot=45, fontsize=10)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("SAS Base Software V3 Variable Selection Models with AUC over 0.9")
plt.suptitle('')
plt.show()

# v4
df = None
for result_file in
glob.glob(f'..{path.sep}results{path.sep}Base{path.sep}all{path.sep}*v4.csv'):
    if df is None:
        df = pd.read_csv(result_file, sep=';', names=['modelo', 'auc'])
    else:
        df = pd.concat([df, pd.read_csv(result_file, sep=';', names=['modelo', 'auc'])],
ignore_index=True)
ax = df.boxplot(column=['auc'], by=['modelo'], rot=80, fontsize=8)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("SAS Base Software V4 Variable Selection Models")
plt.suptitle('')
plt.show()

df_best = df[df['auc'] > 0.88]
df_best = df[df['modelo'].isin(df_best['modelo'])]
ax = df_best.boxplot(column=['auc'], by=['modelo'], rot=45, fontsize=10)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("SAS Base Software V4 Variable Selection Models with AUC over 0.88")
plt.suptitle('')
plt.show()

# v5
df = None
for result_file in
glob.glob(f'..{path.sep}results{path.sep}Base{path.sep}all{path.sep}*v5.csv'):
    if df is None:
        df = pd.read_csv(result_file, sep=';', names=['modelo', 'auc'])
    else:
        df = pd.concat([df, pd.read_csv(result_file, sep=';', names=['modelo', 'auc'])],
ignore_index=True)
ax = df.boxplot(column=['auc'], by=['modelo'], rot=80, fontsize=6)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("SAS Base Software V5 Variable Selection Models")
plt.suptitle('')
plt.show()

df_best = df[df['auc'] > 0.79]
df_best = df[df['modelo'].isin(df_best['modelo'])]
ax = df_best.boxplot(column=['auc'], by=['modelo'], rot=45, fontsize=10)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("SAS Base Software V5 Variable Selection Models with AUC over 0.79")
plt.suptitle('')
plt.show()
```

Annex 1.24: AUC Boxplot chart for Graph Neural Networks models

```
from os import path
import pandas as pd
import matplotlib.pyplot as plt
```

```
# KGAT

df = pd.read_csv(f'..{path.sep}results{path.sep}KGAT{path.sep}full_results.csv',
sep=';')
ax = df.boxplot(column=['test_auc'], by=['model'], rot=90, fontsize=3)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("RecBole KGAT Models")
plt.suptitle('')
plt.show()

df_best = df[df['test_auc'] > 0.80]
df_best = df[df['model'].isin(df_best['model'])]
ax = df_best.boxplot(column=['test_auc'], by=['model'], rot=45, fontsize=10)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("RecBole KGAT Models with AUC over 0.80")
plt.suptitle('')
plt.show()

# MKR

df = pd.read_csv(f'..{path.sep}results{path.sep}MKR{path.sep}full_results.csv', sep=';')
ax = df.boxplot(column=['test_auc'], by=['model'], rot=90, fontsize=5)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("RecBole MKR Models")
plt.suptitle('')
plt.show()

df_best = df[df['test_auc'] > 0.934]
df_best = df[df['model'].isin(df_best['model'])]
ax = df_best.boxplot(column=['test_auc'], by=['model'], rot=45, fontsize=10)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("RecBole MKR Models with AUC over 0.934")
plt.suptitle('')
plt.show()

# RippleNet

df = pd.read_csv(f'..{path.sep}results{path.sep}RippleNet{path.sep}full_results.csv',
sep=';')
ax = df.boxplot(column=['test_auc'], by=['model'], rot=90, fontsize=5)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("RecBole RippleNet Models")
plt.suptitle('')
plt.show()

df_best = df.groupby(['model']).filter(lambda x: x['test_auc'].mean() > 0.93)
ax = df_best.boxplot(column=['test_auc'], by=['model'], rot=45, fontsize=10)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("RecBole RippleNet Models with mean AUC over 0.93")
plt.suptitle('')
plt.show()
```

Annex 1.25: AUC Boxplot chart for the models over 0.93 threshold

```
from os import path
import pandas as pd
import glob
import matplotlib.pyplot as plt

# R
df_r = None
for result_file in
glob.glob(f'..{path.sep}results{path.sep}R{path.sep}all{path.sep}*.csv'):
    if df_r is None:
        df r = pd.read csv(result file)
    else:
        df_r = pd.concat([df_r, pd.read_csv(result_file)], ignore_index=True)
df_r_best = df_r.groupby(['modelo']).filter(lambda x: x['auc'].mean() > 0.93)[['auc',
'modelo']]
```

```
# Base
df_base = None
for result_file in
glob.glob(f'..{path.sep}results{path.sep}Base{path.sep}all{path.sep}*.csv'):
    if df_base is None:
        df_base = pd.read_csv(result_file, sep=';', names=['modelo', 'auc'])
    else:
        df_base = pd.concat([df_base, pd.read_csv(result_file, sep=';', names=['modelo',
'auc'])], ignore index=True)
df_base_best = df_base.groupby(['modelo']).filter(lambda x: x['auc'].mean() >
0.93)[['auc', 'modelo']]

# Graph

df_kgat = pd.read csv(f'..{path.sep}results{path.sep}KGAT{path.sep}full_results.csv',
sep=';')
df_kgat = df_kgat[['test_auc', 'model']].rename(columns={'test_auc': 'auc', 'model':
'modelo'})
df_kgat_best = df_kgat.groupby(['modelo']).filter(lambda x: x['auc'].mean() > 0.93)

df_mkr = pd.read_csv(f'..{path.sep}results{path.sep}MKR{path.sep}full_results.csv',
sep=';')
df_mkr = df_mkr[['test_auc', 'model']].rename(columns={'test_auc': 'auc', 'model':
'modelo'})
df_mkr_best = df_mkr.groupby(['modelo']).filter(lambda x: x['auc'].mean() > 0.93)

df_rn = pd.read_csv(f'..{path.sep}results{path.sep}rippleNet{path.sep}full_results.csv',
sep=';')
df_rn = df_rn[['test_auc', 'model']].rename(columns={'test_auc': 'auc', 'model':
'modelo'})
df_rn_best = df_rn.groupby(['modelo']).filter(lambda x: x['auc'].mean() > 0.93)

df_best = pd.concat([df_r_best, df_base_best, df_kgat_best, df_mkr_best, df_rn_best],
ignore_index=True)

ax = df_best.boxplot(column=['auc'], by=['modelo'], rot=45, fontsize=10)
ax.set_xlabel("Model")
ax.set_ylabel("AUC")
plt.title("Best Models with mean AUC over 0.93")
plt.suptitle('')
plt.show()
```

Annex 2: Results

Annex 2.1: Graph Neural Networks results

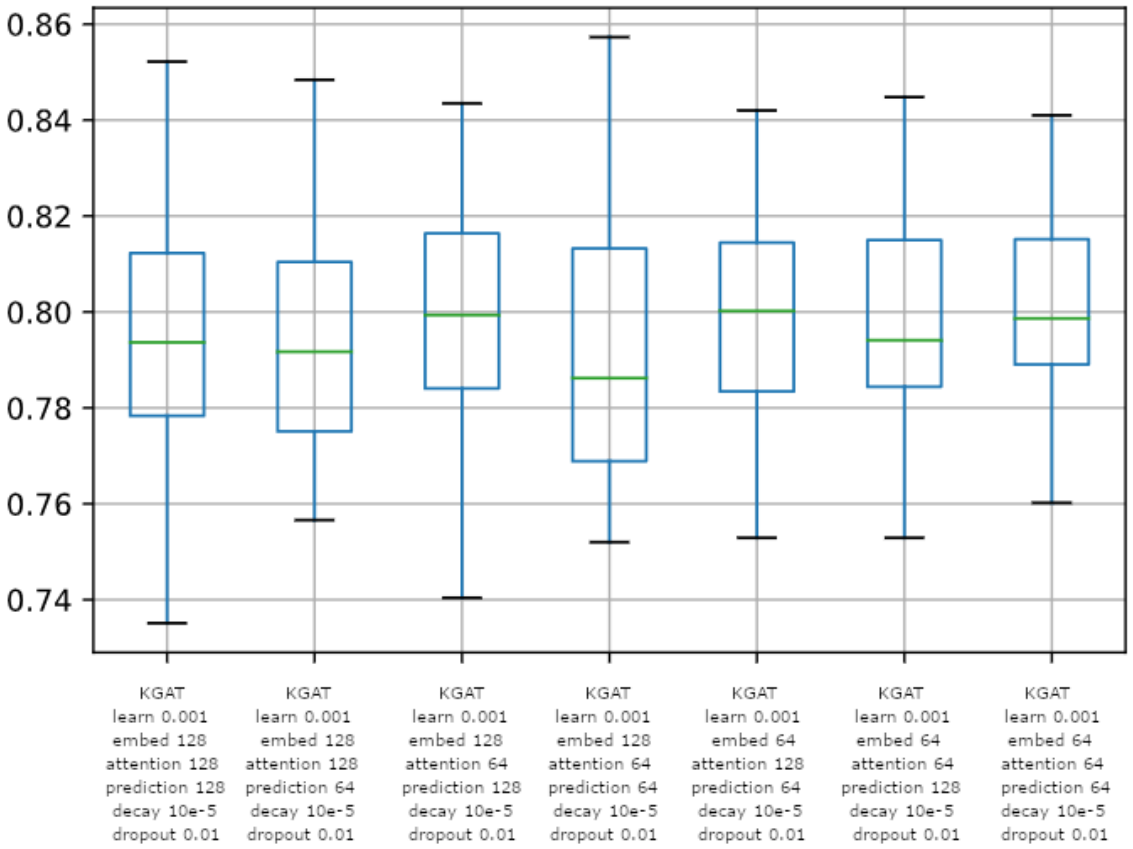
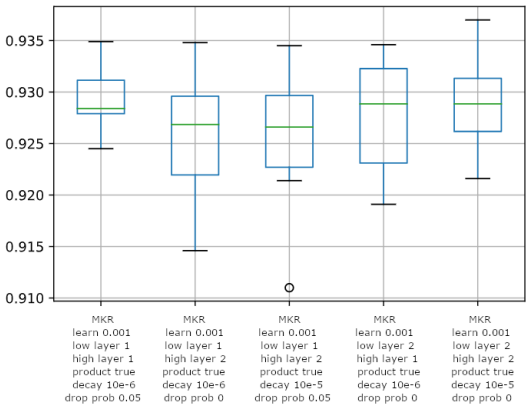


Figure 27 KGAT Best Results



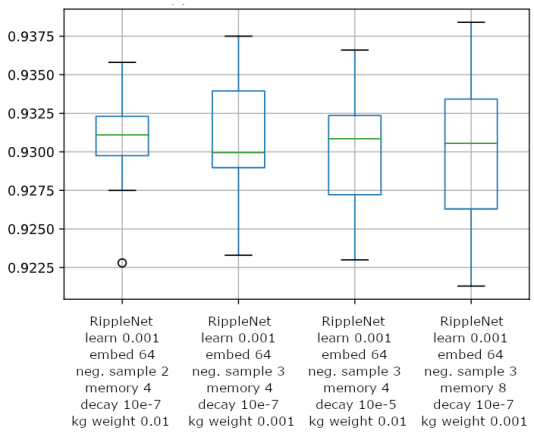


Figure 28 Left. MKR Best Results
Figure 29 Right. RippleNet Best Results

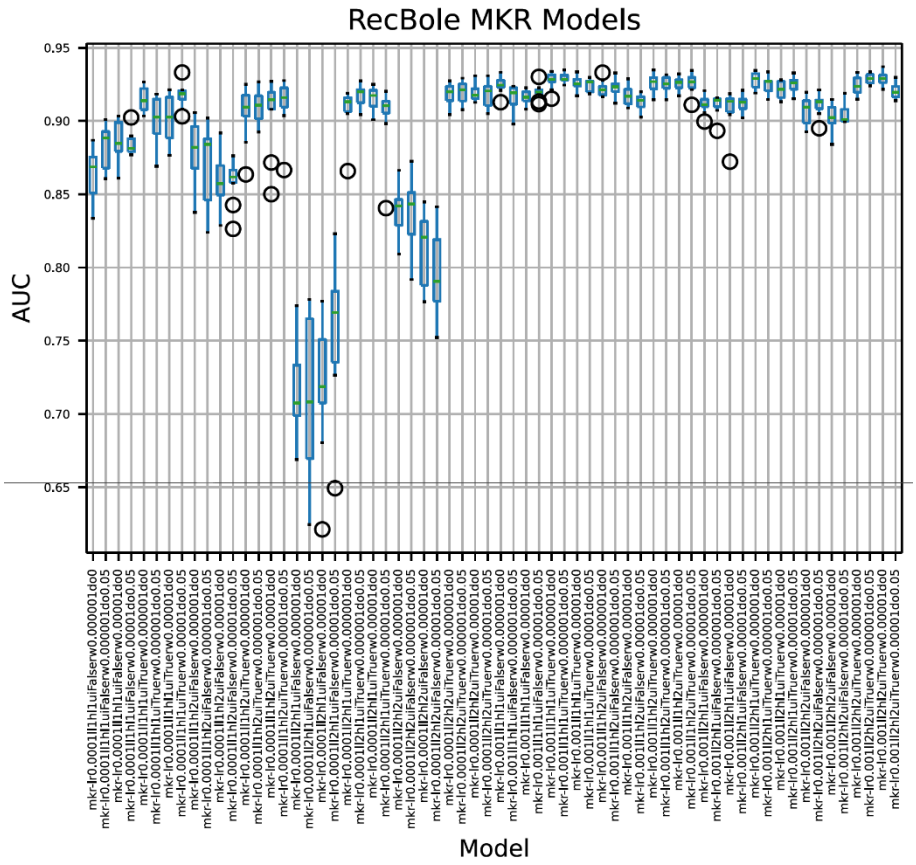


Figure 30 MKR full model results

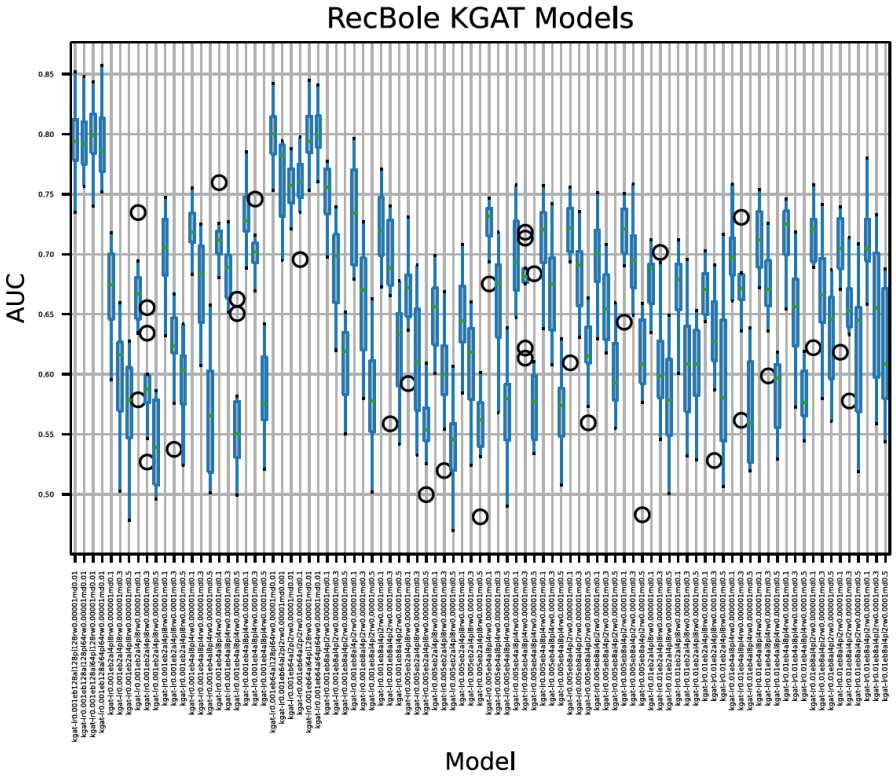


Figure 31 KGAT full model results

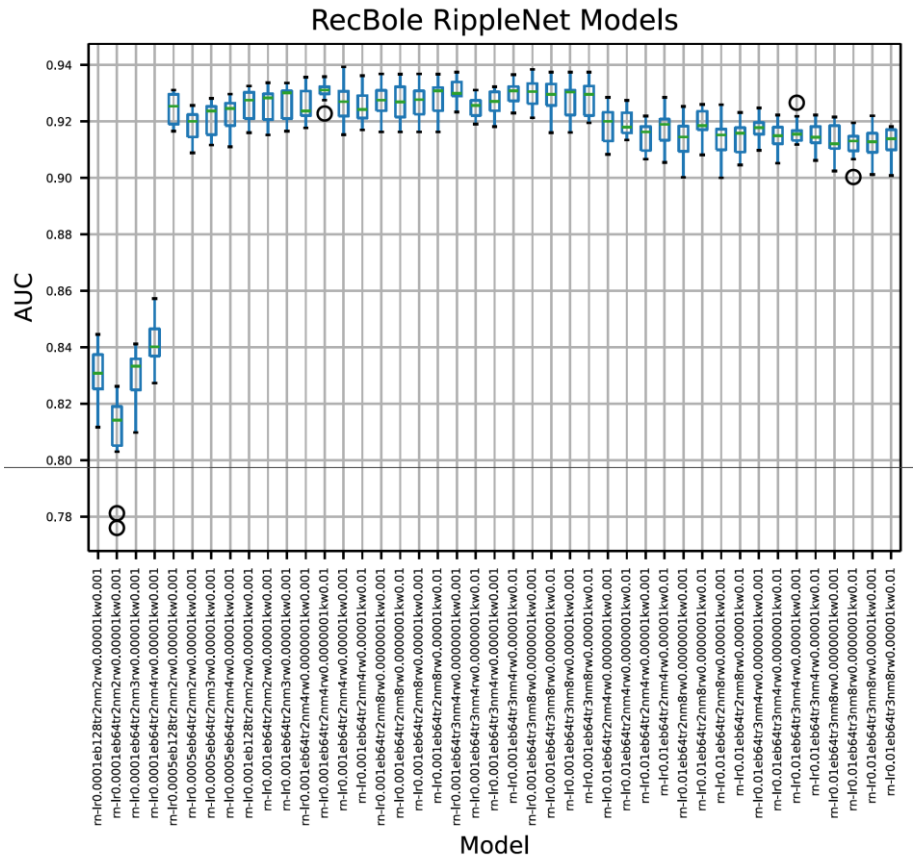
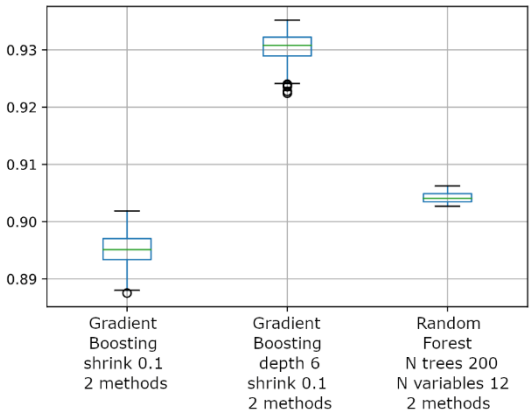
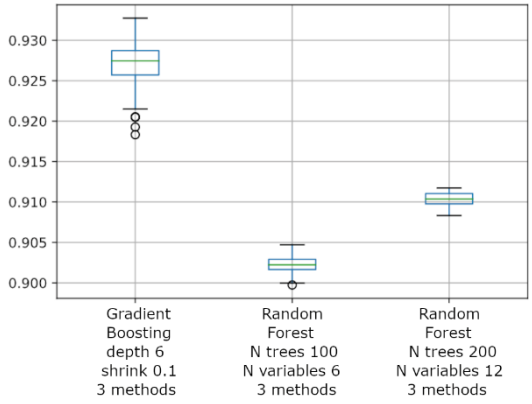
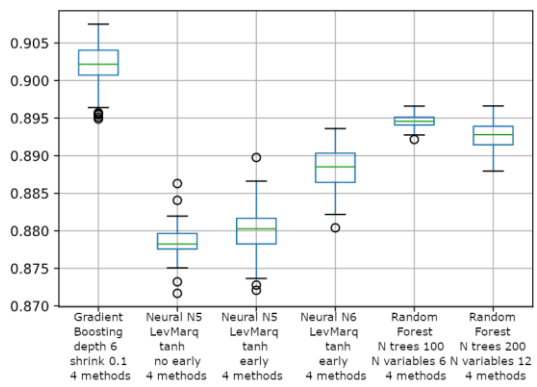


Figure 32 RippleNet full model results

Annex 2.2: Tabular methods results







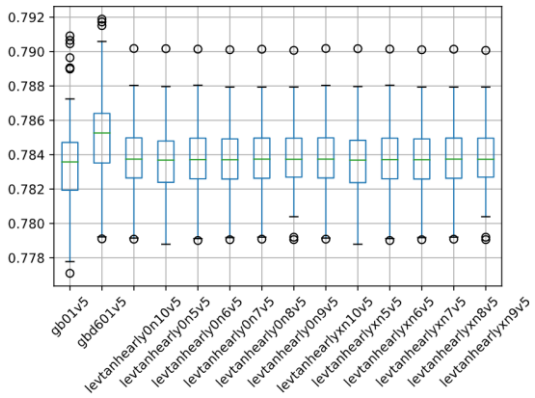


Figure 33 Top Left. SAS Base best models for variables selected by two methods.
Figure 34 Top Right. SAS Base best models for variables selected by three methods.
Figure 35 Bottom Left. SAS Base best models for variables selected by four methods.
Figure 36 Bottom Right. SAS Base best models for variables selected by five methods.

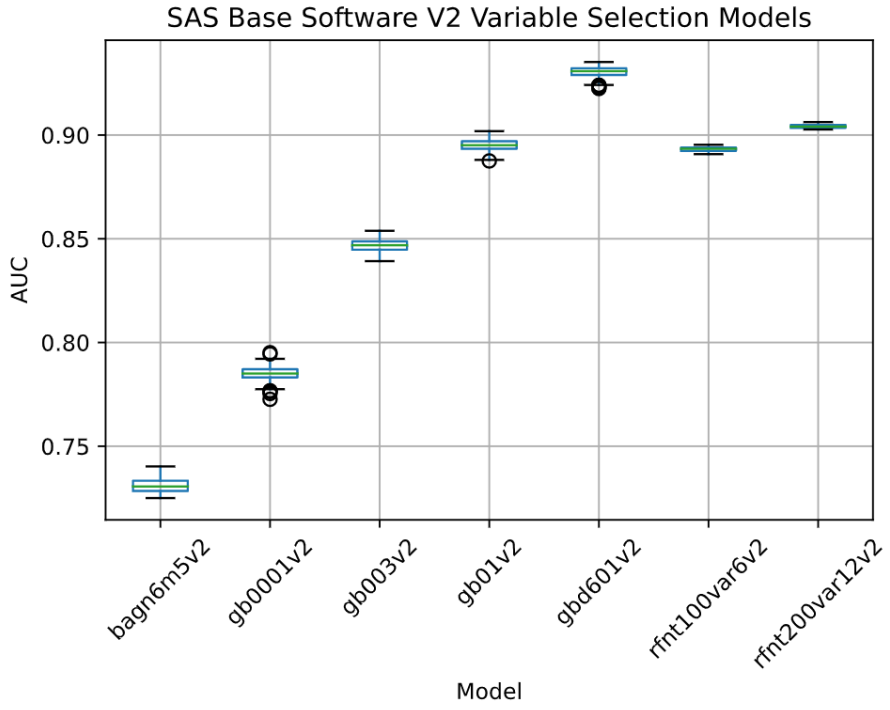


Figure 37 SAS Base two methods selection variable set model results

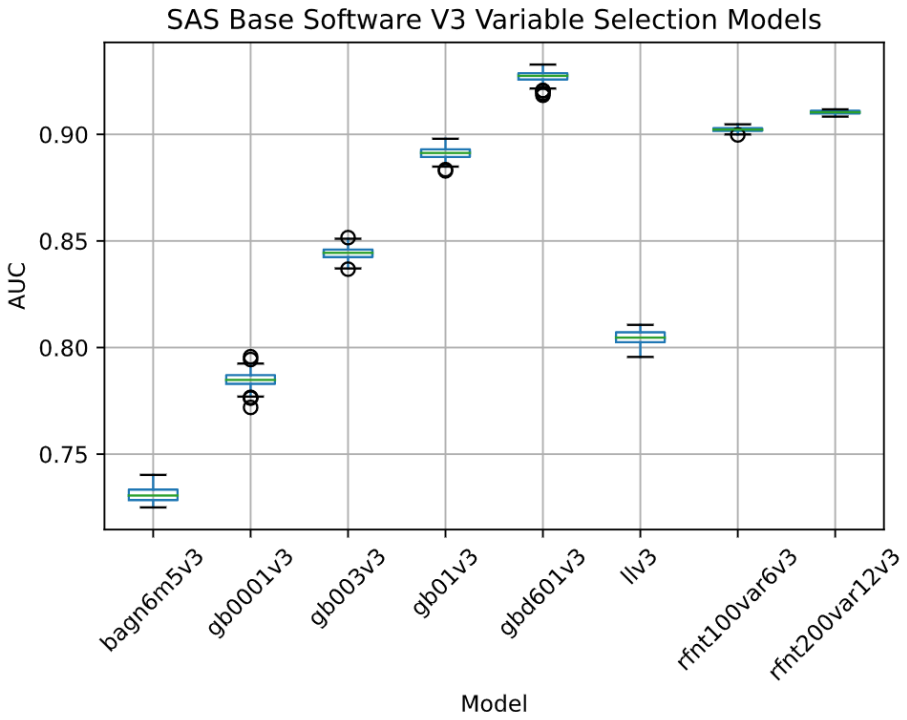


Figure 38 SAS Base three methods selection variable set model results

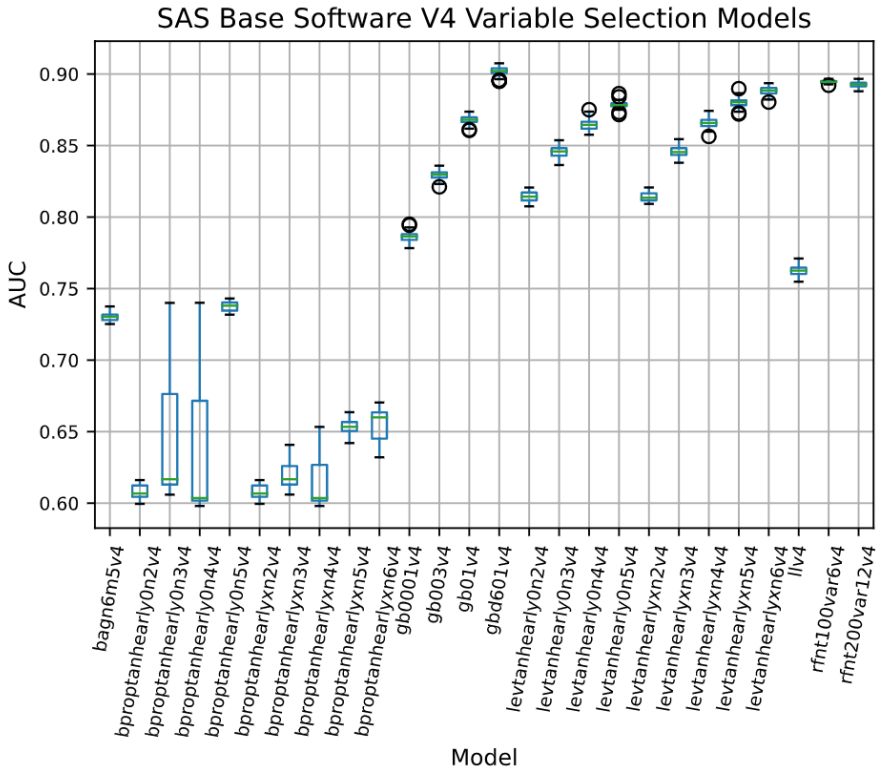


Figure 39 SAS Base four methods selection variable set model results

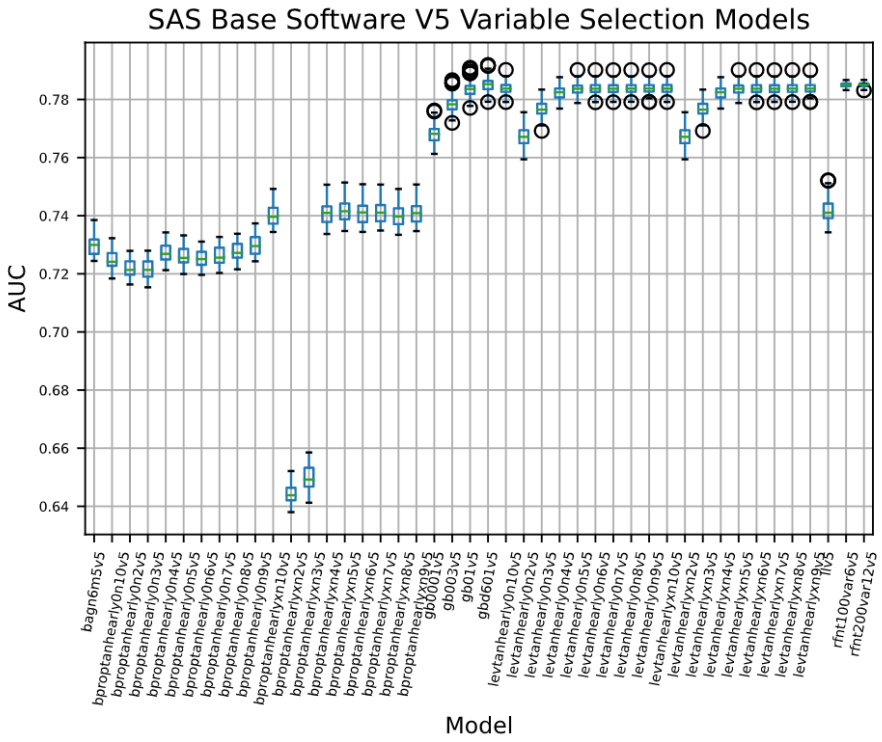


Figure 40 SAS Base five methods selection variable set model results

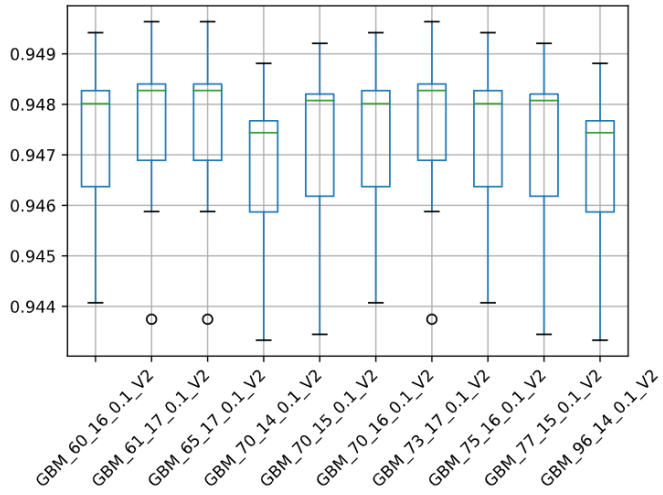


Figure 41 H2O GBM variance study fed with 50 input variables selected by two methods

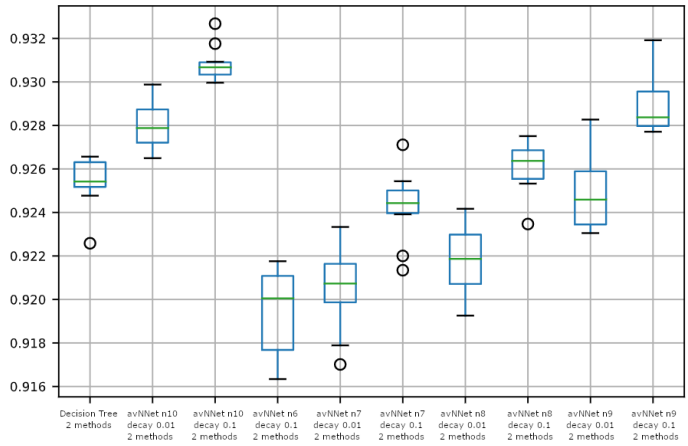


Figure 42 R best models for variables selected by two methods.

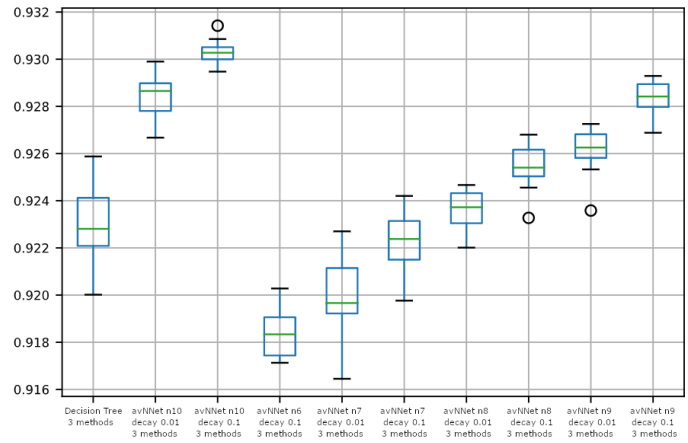
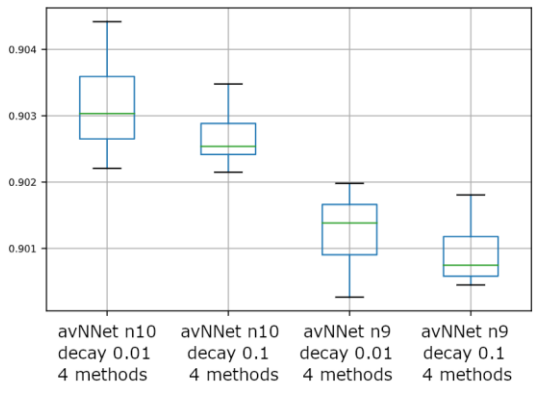


Figure 43 R best models for variables selected by three methods.



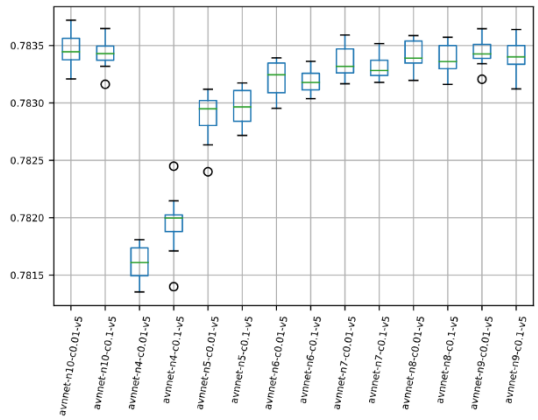


Figure 44 Left. R best models for variables selected by four methods.
Figure 45 Right. R best models for variables selected by five methods.

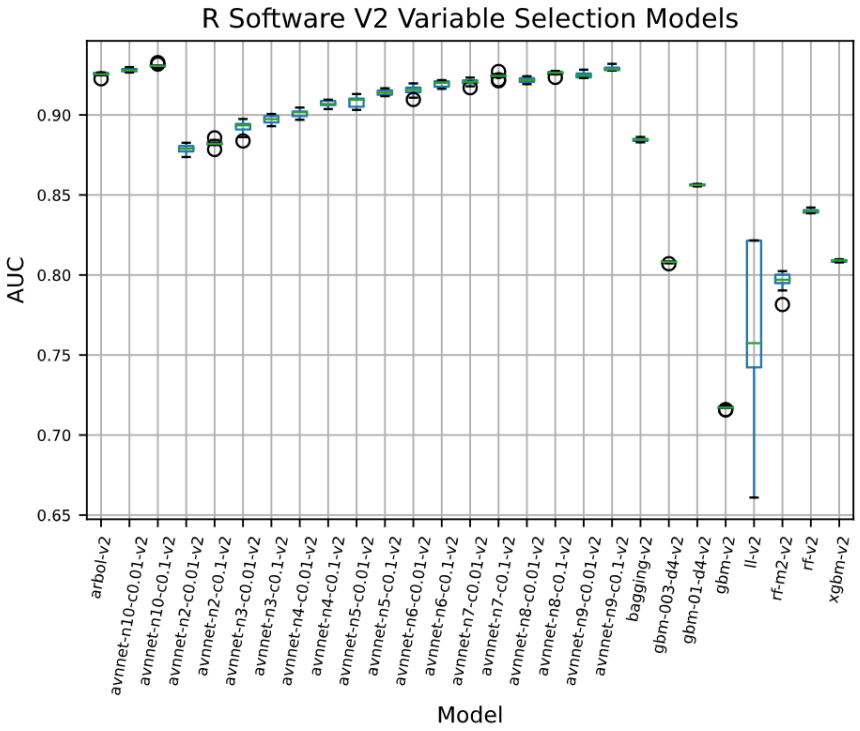


Figure 46 R two methods selection variable set model results

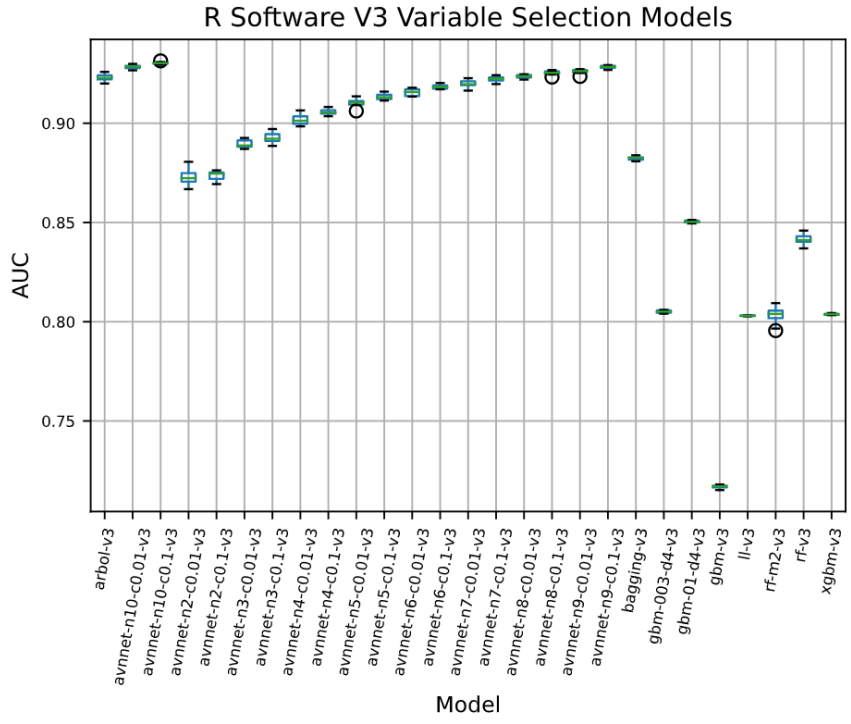


Figure 47 R three methods selection variable set model results

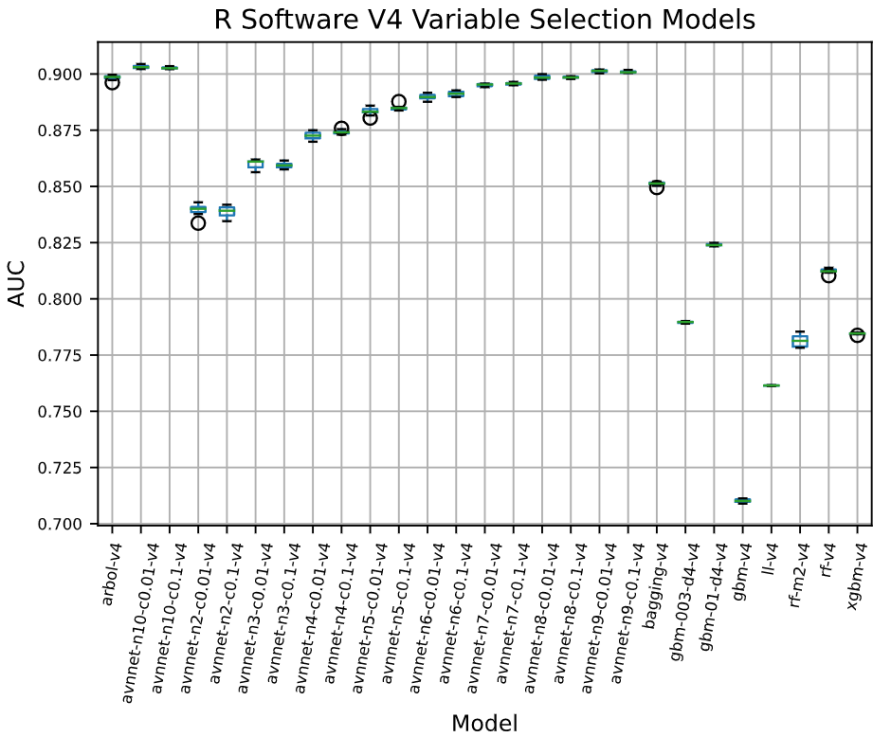


Figure 48 R four methods selection variable set model results

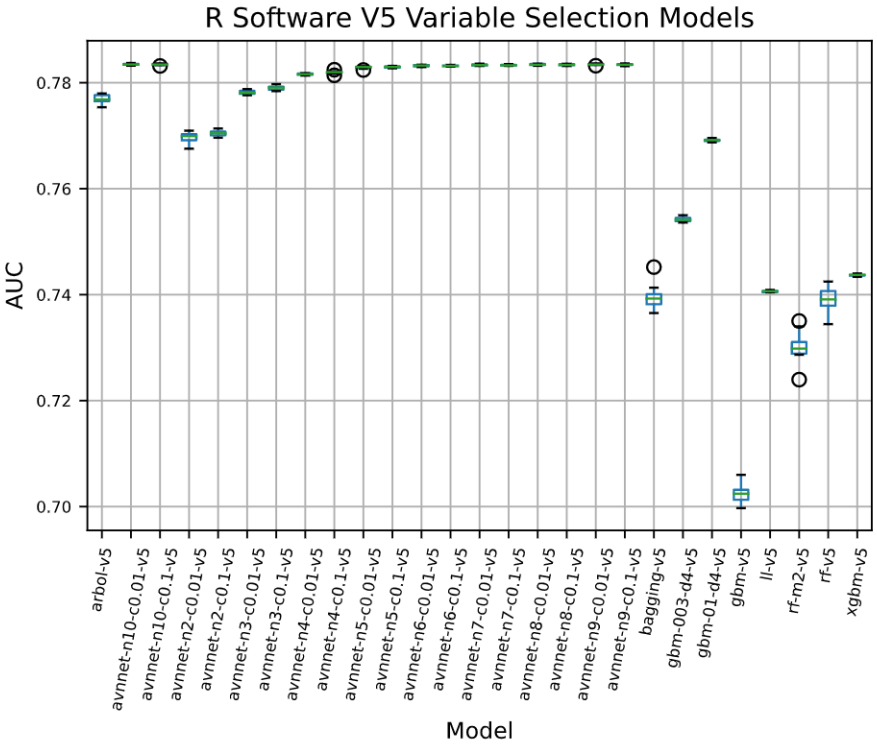


Figure 49 R five methods selection variable set model results