FACULTAD DE ESTUDIOS ESTADÍSTICOS

MÁSTER EN MINERÍA DE DATOS E INTELIGENCIA DE NEGOCIOS

Curso 2021/2022

Trabajo de Fin de Máster

TITULO: ***Predicción de impagos en correduría de seguros***

***Alumno*: Javier Hidalgo García**

***Tutor*: Óscar Garnica y José Manuel Velasco**

Junio (o septiembre) de 2022

UNIVERSIDAD COMPLUTENSE
MADRID

AGRADECIMIENTOS

A Óscar Garnica, José Manuel Velasco, Alberto Gutiérrez y José Ignacio Hidalgo por su paciencia, dedicación y enseñanza durante la realización del trabajo. Y por darme la oportunidad de haber trabajado en el proyecto.

A mis padres por su constante apoyo y esfuerzo en mi formación.

Índice

Índice de figuras.

Índice de tablas.

1. INTRODUCCIÓN

Los seguros, tal y como se conocen hoy en día, se remontan a la Edad Media, más concretamente al siglo XIV en algunas ciudades mediterráneas, ya que los primeros fueron de carácter marítimo. A medida que los intercambios comerciales aumentaron, así lo hicieron también los seguros y, por tanto, se necesitaba un marco regulador. Este llegó en 1538 de la mano de Carlos V con "Las Ordenanzas del Consulado de Burgos" y con él se empezó a evitar en gran medida los fraudes o abusos.

Por otro lado, las primeras sociedades de seguros no se encuentran hasta finales del siglo XVIII cuando aparecen ciertas compañías que se especializan en la cobertura de incendios y vida. Esto impulsó a su vez, el estudio y avance de las ciencias actuariales.

La ciencia actuaria es una disciplina que, a través de modelos estadísticos y las matemáticas, evalúa los riesgos que hay en campos como los seguros o las finanzas entre otros.

Como es conocido, el campo de los seguros es muy antiguo y ha evolucionado mucho; no obstante, se encuentra con el problema de que, en muchos casos, tienen la incertidumbre de no conocer a sus clientes dando servicios poco personales. Todo esto provoca que las aseguradoras reduzcan beneficios, perdiendo a clientes valiosos o, por el contrario, contratando a otros que en un futuro les pueden producir pérdidas. Con la generación de modelos de clasificación basados en aprendizaje automático se pretende evaluar el riesgo de los clientes generando precios o definiciones más exactas que permitan mejorar los resultados anuales de las compañías aseguradoras.

Este trabajo se centra en la modelización de personas con probabilidad alta de producir impagos. Para ello, se sabe que actualmente existen 3 tipos de clientes generando impagos en las compañías aseguradoras. Los tipos de impagadores dan lugar a la siguiente casuística:

- Impago post suscripción

  Este tipo de impago ocurre cuando el contrato se ha suscrito y la aseguradora comienza a cubrir el riesgo. Cualquier siniestro que ocurra y quede cubierto por el contrato de póliza deberá ser afrontado por la aseguradora.

  Otro posible caso de impago post suscripción se genera cuando el cliente recibe (tradicionalmente) el cobro mediante domiciliación bancaria y dispone de varios meses para devolver este recibo desde su banco. Si lo hace, se genera el impago.

  Por último, y siguiendo dentro de este tipo de impago, se encuentra la posibilidad de que el cliente se quiera cambiar de compañía, suscribe un seguro, notifica la baja al seguro anterior, pero recibe una contraoferta por lo que devuelve el nuevo seguro y genera el impago.

- Impago de recibo temporal

  Se produce cuando un cliente paga alguno de los recibos, pero a partir de un recibo comienza a impagar. Por ejemplo, el cliente contrata un seguro de moto en verano, pero deja de pagar en invierno el segundo recibo.

- Impago de renovación

  En este caso el cliente simplemente deja de pagar. La aseguradora podría reclamar judicialmente, y ganaría. Pero el coste de esto no justifica, generalmente, el importe de la prima por lo que no se hace y se pierde ese dinero.

Realizar un modelo que prediga si el cliente actual o futuro va a generar un impago dentro de una compañía de seguros ayudará al crecimiento de estas pudiendo mejorar anualmente en servicios y productos.

1.1. Empresa Biztools y Motopoliza

BizTools es una empresa de software especializada en el desarrollo de proyectos informáticos en internet. Y tras un convenio de colaboración con la Universidad Complutense de Madrid, ambas colaboran en la consecución del proyecto Aictuari. Aictuari es un proyecto que pretende revolucionar la forma en que las compañías aseguradoras gestionan el riesgo de sus carteras de seguros de automóviles, mejorando las capacidades predictivas y el resultado técnico.

Por otro lado, la empresa Motopoliza comparadora de seguros, es la empresa encargada de proporcionar los datos con los que se trabaja en este proyecto. A su vez, y como mejoras futuras, la empresa Allianz pretende proporcionar más datos con los que se podrá realimentar el modelo y hacerlo cada vez más preciso.

2. OBJETIVOS

El objetivo principal de este proyecto es definir un modelo que prediga cual de nuestros clientes va a generar un impago.

Para lograr el objetivo principal se han de abordar una serie de problemas que irán apareciendo a lo largo del proyecto:

- Resolver el problema del desbalanceo de los datos
- Conseguir un dataset más completo o realimentarlo con más información de las variables ya dadas.
- Codificar los datos para que el modelo se pueda entrenar mejor.
- Evaluar la calidad de las variables y hacer selección de estas.

3. METODOLOGÍA Y MATERIALES

3.1. Aprendizaje automático

El aprendizaje automático (ML) es el subapartado de la inteligencia artificial (IA) que se centra en desarrollar sistemas que aprenden, o mejoran el rendimiento, en función de los datos que consumen. Inteligencia artificial es un término amplio que se refiere a sistemas o máquinas que imitan la inteligencia humana.

En este proyecto se aborda un problema de clasificación, siendo el objetivo clasificar si una persona va a pagar o va a producir un impago. Para conseguirlo se debe seguir la siguiente secuencia de pasos:

1. Definir el modelo con sus hiperparámetros.
2. Dividir los datos en los conjuntos de entrenamiento y test. Normalmente, el 75% del conjunto de datos es destinado para entrenamiento del modelo y el 25% restante para test.
3. Entrenar el modelo con el conjunto de datos de entrenamiento.
4. Testear el aprendizaje del modelo con el conjunto de datos de test.
5. Analizar resultados, comparando la salida obtenida del modelo con los valores reales del conjunto de test, obteniendo así el porcentaje de acierto, y comprobando la efectividad del modelo.

3.2. Tipos de modelización

A lo largo de este trabajo voy a utilizar modelos predictivos basados en las técnicas que presento en las siguientes secciones.

3.2.1. Regresión logística.

La regresión logística busca predecir la probabilidad de que una entrada de datos concreta pertenezca a una categoría. De forma similar a como la regresión lineal supone que los datos siguen una función lineal, la regresión logística modela la probabilidad de que un dato pertenezca a una categoría mediante la función sigmoidea y un umbral que determina a partir de qué valor de la función sigmoidea el dato pertenece a una categoría u otra.

La regresión logística se puede convertir en una técnica de clasificación introduciendo un umbral de decisión: si el valor proporcionado por la regresión logística está por encima del valor umbral entonces el dato pertenece a una categoría y si está por debajo, pertenece a otra categoría. La configuración del valor umbral es uno de los aspectos más importantes del modelo y depende del problema de clasificación en sí.

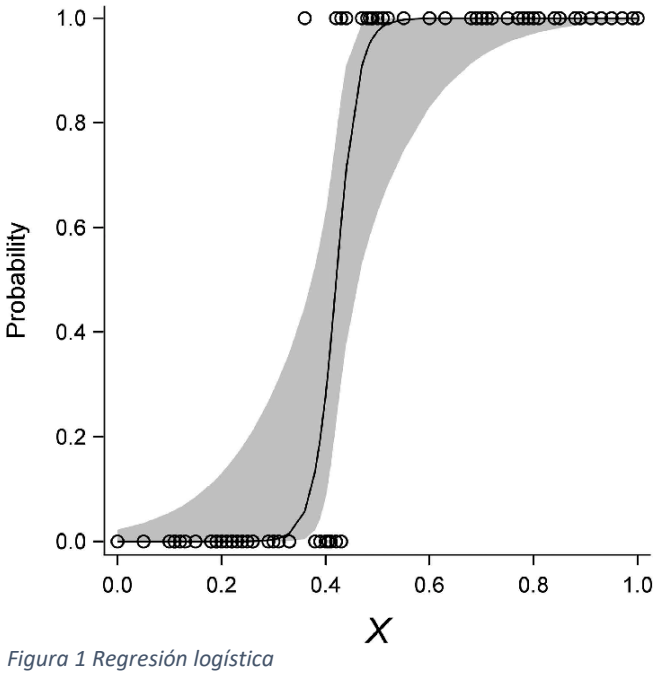


Figura 1 Regresión logística

La Figura 1 muestra un ejemplo de cómo se forma la función sigmoidea en un modelo de regresión logística con valores cualitativos convertidos en cuantitativos entre 0 y 1. Los círculos pequeños representan la distribución de los datos y la curva: si define el cambio de valores de pertenecer a una categoría o a otra.

Los parámetros modificables dentro del modelo en la librería Sklearn (se explicará que es la librería Sklearn en la sección 4.2.1) son los siguientes:

- Class_weight: Pesos asociados con clases en la forma {class_label: peso}. Si no se proporciona, se supone que todas las clases tienen peso uno.
- Max_iter: Número máximo de iteraciones necesarias para que los solucionadores converjan.
- Solver: Algoritmo a utilizar en el problema de optimización. El valor predeterminado es 'lbfgs'.
- Tol: Tolerancia para los criterios de parada.
- Penalty: Especifica la norma de la penalización.

3.2.2. Random forest.

Random forest es un algoritmo de aprendizaje supervisado. Opera construyendo una multitud de árboles de decisión durante la etapa de entrenamiento. Durante la fase de evaluación de una nueva entrada cada uno de los árboles individuales genera una nueva predicción. Posteriormente, se combina el resultado de las múltiples predicciones para generar la predicción final.

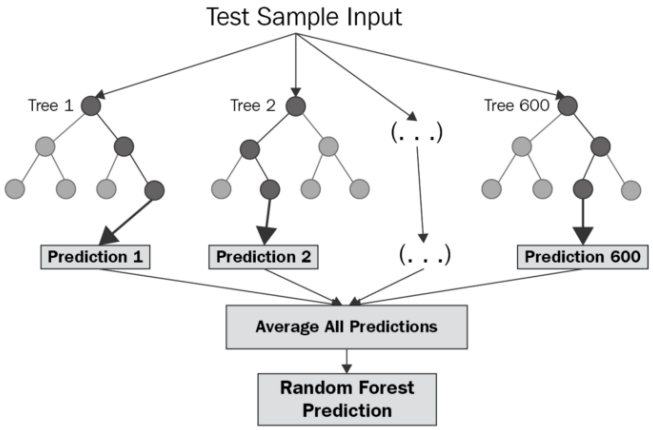


Figura 2 Proceso Random Forest

La Figura 2 ilustra el proceso de creación de la predicción del modelo de Random Forest. Crea, en este caso, 600 árboles y genera una predicción con el promedio de todas esas predicciones.

Los parámetros modificables dentro de este modelo y dentro de la librería Sklearn son los siguientes:

- Ccp_alpha: Parámetro de complejidad utilizado para la poda de complejidad de costo mínimo. Se elegirá el subárbol con la mayor complejidad de costos que sea menor que ccp_alpha. De forma predeterminada, no se realiza ninguna poda.
- Class_weight: Pesos asociados con clases en la forma {class_label: peso}. Si no se proporciona, se supone que todas las clases tienen peso uno.
- Criterion: La función para medir la calidad de una división. Los criterios admitidos son "gini" para la impureza de Gini y "log_loss" y "entropía".
- Max_depth: La profundidad máxima del árbol. Si es Ninguno, los nodos se expanden hasta que todas las hojas estén puras o hasta que todas las hojas contengan menos del número mínimo de muestras necesario para dividir un nodo interno.
- Max_features: El número de características a tener en cuenta al buscar la mejor división.
- N_estimators: El número de árboles.
- Bootstrap: Si se utilizan muestras de arranque al construir árboles. Si es False, se usa todo el conjunto de datos para construir cada árbol.

3.2.3. Bagging.

Es un método que sirve para generar múltiples versiones de un predictor y usarlas para obtener un predictor agregado. Para conseguirlo toma como base el entrenamiento con cada uno de los clasificadores con conjuntos de datos aleatorios sobre el dataset original y posteriormente agrega dichas predicciones individuales (ya sea por votación o media) para generar una predicción final. Normalmente este tipo de modelo se suele utilizar para reducir la varianza de un modelo de caja negra (árbol de decisión), introduciendo

la aleatorización en su procedimiento de construcción y luego haciendo un conjunto a partir de él.

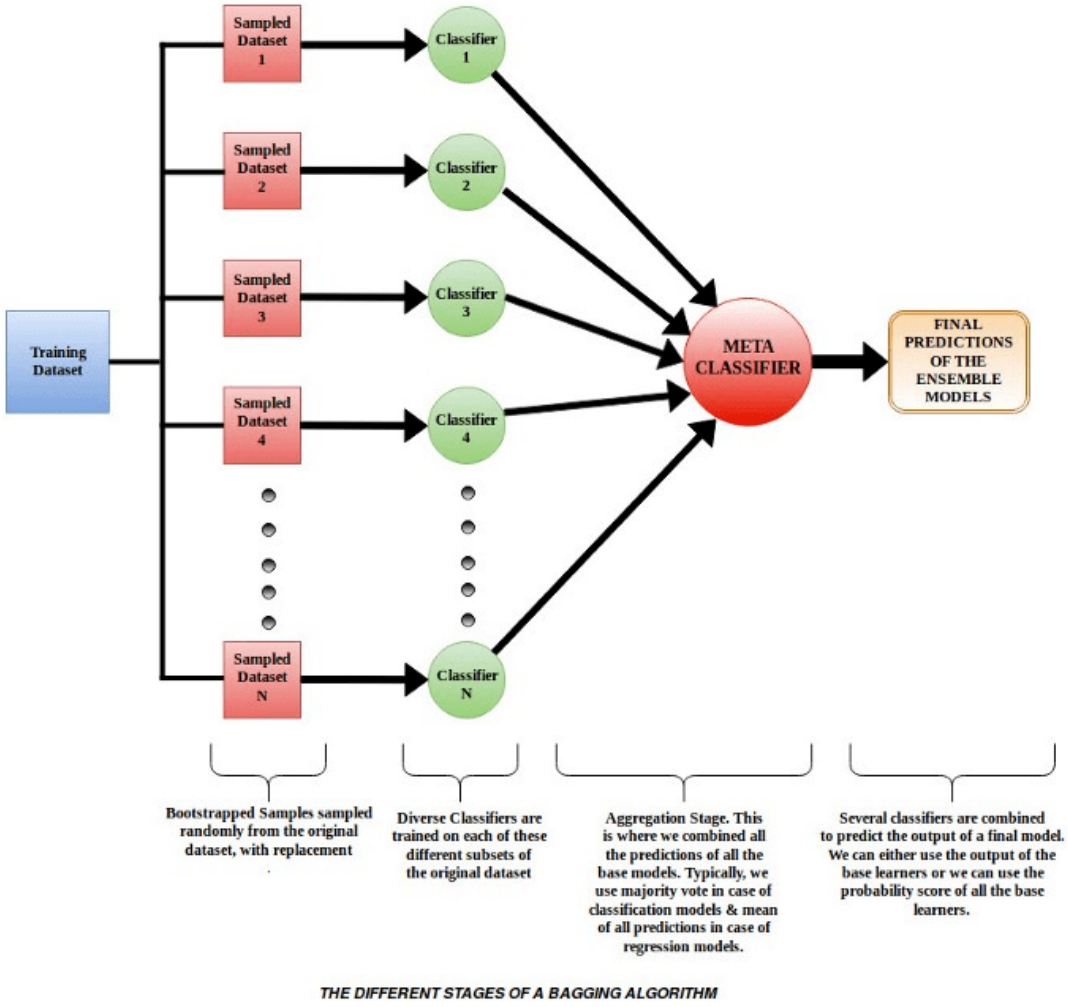


Figura 3 Estados Bagging

La Figura 3 muestra las diferentes etapas por las que pasa el algoritmo para generar la predicción. En primer lugar, escoge diferentes subconjuntos del dataset, genera sus diferentes árboles de decisión, genera una predicción de cada uno y genera una media con todos ellos que será la predicción final.

Los parámetros modificables dentro de este modelo y dentro de la librería Sklearn son los siguientes:

- Max_features: El número de características a tener en cuenta al buscar la mejor división.
- N_estimators: El número de árboles.
- Bootstrap: Si se utilizan muestras de arranque al construir árboles. Si es False, se usa todo el conjunto de datos para construir cada árbol.

3.2.4. Gradient boosting

Gradient Boosting se usa para minimizar una función de pérdida (función que mide la discrepancia entre la predicción de un algoritmo de aprendizaje automático y la salida supervisada y representa el costo de equivocarse). En cada ronda de entrenamiento se genera un modelo y se comparan sus predicciones con el resultado correcto esperado. La distancia entre la predicción y el valor correcto representa la tasa de error del modelo. A partir de estos errores se calcula el gradiente, el cual es la derivada parcial de la función de pérdida, por lo que describe la inclinación de la función de error.

Por lo tanto, se construye un modelo aditivo de manera progresiva hacia el escenario buscado, permitiendo la optimización de funciones de pérdida diferenciables arbitrarias. En cada etapa, un árbol de regresión se ajusta al gradiente negativo de la función de pérdida dada.

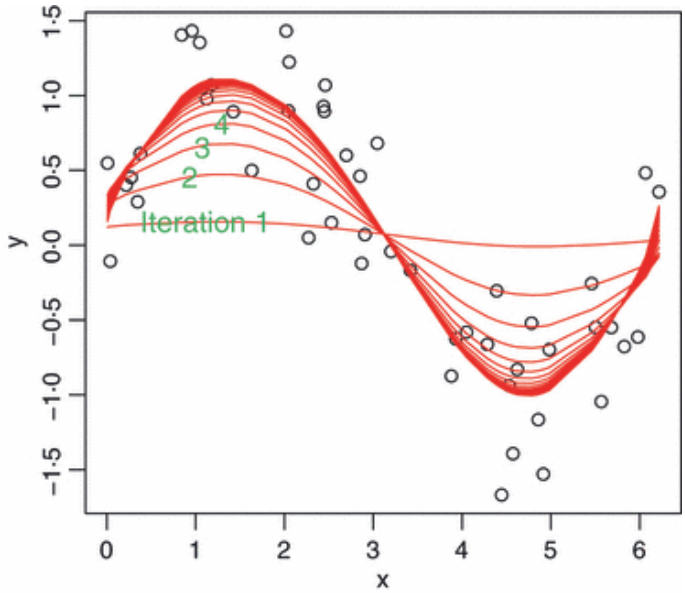


Figura 4 Evolución de Gradient Boosting entre iteraciones

La Figura 4 muestra cómo evolucionan y ajusta el algoritmo cada una de las iteraciones que produce nuestro modelo generando el menor error posible. Las líneas rojas representan cada iteración y los círculos negros pequeños los datos. Como se ve, en cada iteración la línea roja se ajusta más los datos.

Los parámetros modificables dentro de este modelo y dentro de la librería Sklearn son los siguientes:

- Ccp_alpha: Parámetro de complejidad utilizado para la poda de complejidad de costo mínimo. Se elegirá el subárbol con la mayor complejidad de costos que sea menor que ccp_alpha. De forma predeterminada, no se realiza ninguna poda.
- Criterion: La función para medir la calidad de una división. Los criterios admitidos son 'friedman_mse' para el error cuadrático medio con puntuación de mejora de Friedman, 'squared_error' para el error cuadrático medio. El valor

predeterminado de 'friedman_mse' es generalmente el mejor, ya que puede proporcionar una mejor aproximación en algunos casos.

- Learning_rate: La tasa de aprendizaje reduce la contribución de cada árbol por learning_rate. Hay una compensación entre learning_rate y n_estimators. Los valores deben estar en el rango (0.0, inf).
- Loss: La función de pérdida a optimizar. 'log_loss' se refiere a la desviación binomial y multinomial, la misma que se usa en la regresión logística. Es una buena opción para la clasificación con salidas probabilísticas.
- Max_depth: La profundidad máxima de los estimadores de regresión individuales. La profundidad máxima limita el número de nodos en el árbol. Los valores deben estar en el rango [1, inf).
- Max_features: El número de características a tener en cuenta al buscar la mejor división
- N_estimators: El número de etapas de refuerzo a realizar. El aumento de gradiente es bastante resistente al sobreajuste, por lo que un gran número generalmente da como resultado un mejor rendimiento. Los valores deben estar en el rango [1, inf).

3.3. Parámetros utilizados en los modelos.

Tabla 1 Parámetros utilizados en cada uno de los modelos

MODELO	Ccp_alpha	Class_weight	Criterion	Learning_rate	Loss	Max_depth	Max_features	N_estimators	Bootstrap	Max_iter	Solver	Tol	Penalty
Regresión Logística		Balanced								100	lbfgs	0.0001	12
Random Forest	0.0		Gini			None	Auto	100	True				
Bagging							Auto	250	True				
Gradient Boosting	0.0		Friedman_mse	0.1	Deviance	3	None	100					

3.4. Técnicas de remuestreo y evaluación

3.4.1. Desbalanceo de los datos

El desbalanceo de datos se produce cuando el número de observaciones de la variable a predecir pertenecientes a un grupo o clase es mucho mayor a las de la otra clase.

En el caso de este trabajo hay muchas más observaciones de gente que paga que de la gente que no lo hace. Por ello, para comprobar si los resultados mejoran a la hora de igualar los datos, durante la Etapa 3 - Balanceo, se eliminan instancias de la clase mayoritaria para poder igualarla a la minoritaria.

3.4.2. Partición de datos

Con el fin de poder evaluar el modelo final sobre una porción de los datos con la que no se haya trabajado hasta el momento, se realiza una partición de datos por la que se toma el 75% de las observaciones como conjunto de entrenamiento o *train* y un 25% como conjunto de *test* en todos los modelos realizados durante la realización del proyecto y que se verán más adelante en los siguientes apartados.

3.4.3. Técnicas de remuestreo – Validación cruzada

En el caso de este trabajo, la técnica de remuestreo elegida es la validación cruzada. Esta técnica consiste en dividir los datos de forma aleatoria en k grupos de aproximadamente el mismo tamaño, k-1 grupos se emplean para entrenar el modelo y uno de los grupos se emplea como validación. Este proceso se repite k veces utilizando un grupo distinto como validación en cada iteración.

3.4.4. Evaluación de la relación de las variables.

Para estudiar el grado de relación de las variables *input* entre sí y con la variable objetivo, se utiliza el coeficiente V de Cramer aplicado al chi-cuadrado. Este coeficiente toma valores entre 0 y 1; siendo 0 la inexistencia de relación; y 1, relación total.

En este caso se estudia mediante una matriz de correlación. Con esto se pretende observar si existen muchas variables que tienen relaciones altas. Si es así, el modelo tiende a definir la predicción hacia esas variables por lo que el objetivo será eliminar alguna de ellas para que haya el menor “ruido” posible.

3.4.5. Matriz de confusión

La matriz de confusión es una herramienta muy útil para valorar cómo de bueno es un modelo clasificación basado en aprendizaje automático. En particular, sirve para mostrar de forma explícita cuándo una clase es confundida con otra, lo cual nos permite trabajar de forma separada con distintos tipos de error.

Esta es la forma visual con la que se estudia todos los modelos que aparecen en este trabajo.

3.4.6. Features selection

Feature selection es el proceso de seleccionar un subconjunto de las variables de entrada con el propósito de reducir el número de variables de entrada al desarrollar un modelo predictivo. Es deseable reducir el número de variables de entrada para reducir el costo computacional del modelado y, en algunos casos, para mejorar el rendimiento del modelo.

Aunque durante el trabajo se realiza un tratamiento de las variables más profundo y detallado; la cantidad de variables al comienzo es muy grande y con un estudio superficial se pueden rechazar algunas de estas.

3.4.7. Features importance

Features importance se refiere a las técnicas que calculan una puntuación para todas las características de entrada para un modelo dado; las puntuaciones simplemente representan la "importancia" de cada característica. Una puntuación más alta significa que la característica específica tendrá un mayor efecto en el modelo que se utiliza para predecir una determinada variable.

4. MATERIALES

4.1. Bibliotecas empleadas

En este apartado se explica las diferentes tecnologías empleadas a lo largo del proyecto, así como las herramientas empleadas para ello. El proyecto se ha realizado en su totalidad en el lenguaje Python. Este lenguaje es elegido debido a su flexibilidad, permitiendo realizar pruebas en pocos minutos y proporcionando un gran número de bibliotecas con las que poder abordar los diferentes puntos del proyecto.

4.2. Bibliotecas de Python

4.2.1. Sklearn

Sklearn es una biblioteca para aprendizaje automático de software libre para Python. Esta incluye varios algoritmos de clasificación, regresión y análisis de grupos, además de tener diferentes técnicas de preprocesamiento de datos y estudio de los mismos. De dicha biblioteca se han utilizado los modelos citados a lo largo del proyecto, así como algunas técnicas de preprocesamiento y selección de variables.

4.2.2. Graphviz

Graphviz es un conjunto de herramientas software para el diseño de diagramas definido en el lenguaje descriptivo DOT. Estos diagramas se han usado sobre todo para generar los diferentes árboles obtenidos por los modelos Random Forest.

4.2.3. Numpy

Numpy es una biblioteca para Python que permite crear vectores y matrices de gran tamaño y multidimensionales. A su vez cuenta con una gran cantidad de funciones matemáticas de alto nivel.

4.2.4. Pandas

Pandas es una biblioteca de software libre escrita como extensión de Numpy para manipulación y análisis de datos. Uno de los puntos fuertes de esta biblioteca es la creación de DataFrames (tablas de datos), los cuales nos permiten un manejo de datos más rápido y cómodo.

4.2.5. Shap

SHAP es una herramienta basada en la teoría de juego cuya finalidad es dar una explicación al resultado de cualquier modelo de aprendizaje automático. El valor que proporciona SHAP mide cuanto contribuye cada característica a la puntuación del modelo. Cabe destacar que estos valores miden tanto la contribución positiva como la negativa.

Una ventaja que ofrece SHAP es su capacidad de calcular sus valores para cualquier modelo basado en árboles, mientras que con otras tecnologías es necesario emplear modelos de regresión lineal o regresión logística como modelos sustitutos. De todos los métodos que contiene SHAP se elige el siguiente.

4.2.5.1. Beeswarm

Para la explicación de este diagrama se ha escogido primero un ejemplo con una sola variable.

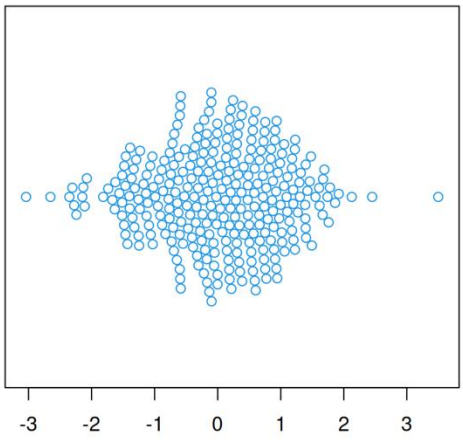


Figura 5 Beeswarn con una sola variable

La Figura 5 muestra una distribución de los datos de una sola variable. En ese caso se agrupan sobre todo entre el -2 y el 2. En el siguiente caso se va a estudiar, también, la relación que tienen los datos con la variable objetivo siendo de color rojo cuando tienen una alta relación y azul cuando es más baja. Cada instancia de la explicación dada está representada por un solo punto en cada flujo de características.

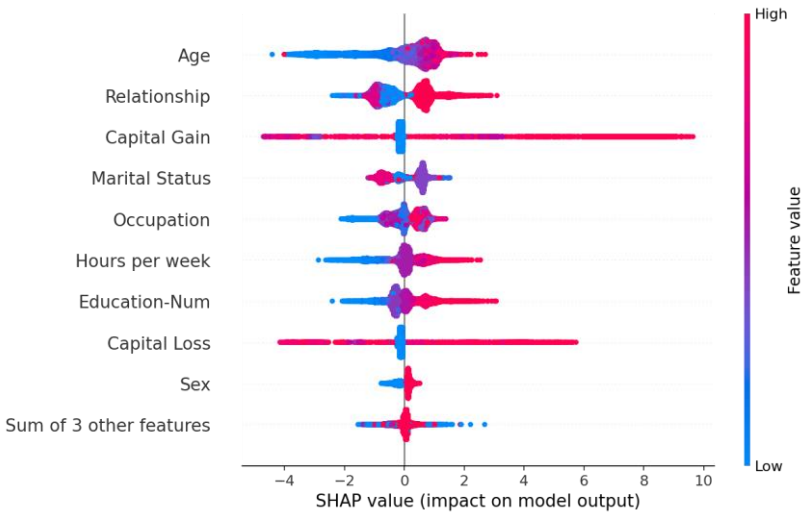


Figura 6 Ejemplo Beeswarm

La Figura 6, expone un ejemplo del diagrama de enjambre de abejas. La posición x del punto está determinada por el valor SHAP de esa característica, y los puntos se "apilan" a lo largo de cada fila de características para mostrar la densidad. El color se utiliza para mostrar el valor original de una característica.

4.3. Software y máquina.

Para la realización de este trabajo se ha utilizado el lenguaje de programación Python. Mediante este programa se ha realizado la depuración, tratamiento previo de los datos y la modelización.

La máquina utilizada para la realización de este trabajo es un *Macbook Pro* con procesador Intel i7, 16GB RAM y macOS Monterey 12.5.1.

Además, se utiliza un servidor para la subida y almacenamiento de los datos con las siguientes características:

- CPU: Realtek RTD1296 SoC
- 2 discos de 2TB
- 512 MB

5. DESCRIPCIÓN DE LOS DATOS

Los datos utilizados para la consecución del proyecto son datos provenientes de la empresa "Biztools". Estos datos han sido recopilados durante años, lo que permite obtener todo tipo de casuísticas que se producen y hace que el modelo sea más fiable.

Estos datos están recopilados con el formato de "Los Estándares EIAC". Estos nacen de la iniciativa del Grupo de Trabajo de Conectividad de UNESPA ante la necesidad de contar con un esquema de datos y unos procesos estándar que permitan a las entidades aseguradoras y a sus mediadores intercambiar información entre ellos de una manera eficaz y eficiente, bajo un modelo de estándar abierto.

Los Estándares EIAC han sido desarrollados por representantes de diferentes entidades aseguradoras y asociaciones de corredurías conjuntamente con TIREA.

Por parte de las asociaciones de corredores:

- ADECOSE
- CONSEJO GENERAL
- FECOR

Por parte de las entidades aseguradoras:

- ALLIANZ
- AXA
- CATALANA OCCIDENTE
- GENERALI
- LIBERTY
- MAPFRE
- NATIONALE SUISSE
- PELAYO
- PLUS ULTRA SEGUROS
- REALE
- SANITAS
- ZURICH

A continuación, se describen todas las variables con las que se ha trabajado.

Tabla 2 Descripción de los datos

VARIABLE	DESCRIPCIÓN
‘IdPoliza'	Número de póliza
'CodigoInterno'	Código interno de Entidad / Correduría
'SituacionPoliza'	Situación actual de la póliza
'ClasePoliza'	Clase de póliza
'DuracionPoliza'	Duración de la póliza
'FechaEfectoInicial'	Fecha de primer efecto. En Suplemento y Recibo se refiere a Póliza.
'FechaEfectoActual'	Fecha de último efecto o actual. En Suplemento es la Fecha de efecto del suplemento. En Recibo es la Fecha de efecto del recibo.
'FechaVencimiento'	Fecha de vencimiento. En Suplemento es la Fecha de vencimiento del suplemento. En Recibo es la Fecha de vencimiento del recibo. Si póliza vitalicia, valor 31/12/9999
'Matricula'	Matrícula del vehículo
'ClaseVehiculo'	Clase de vehículo
'Marca'	Marca del vehículo

'Modelo'	Modelo del vehículo
'Version'	Versión de estándares EIAC (esquema de datos y procesos de seguros estándar) del documento XML
'BaseSIETe'	Referencia Base SIETe: información relativa al vehículo
'UsoVehículo'	Uso del vehículo
'Antigüedad'	Año de construcción
'FechaMatriculacion'	Fecha de matriculación.
'Combustible'	Clase de combustible utilizado por el vehículo.
'CategoriaVehiculo'	Categoría del vehículo
'Cilindrada'	Cilindrada del vehículo (en cm3)
'Potencia'	Potencia del vehículo (en CV)
'Plazas'	Número de plazas
'Valor'	Valor del accesorio
'PMA'	Peso máximo autorizado
'Remolque'	Indicador de si tiene remolque
'TipoIdentificacion_p'	Tipo de identificación
'IdPersona_p'	Identificación (NIF, NIE, Pasaporte) Valor "No identificado" si la persona no dispone de documento identificativo. (TipoIdentificacion = NO)
'IdCliente_p'	Id o Código de cliente de compañía
'Nombre_p'	Nombre de la persona
'Apellido1_p'	Primer apellido
'Apellido2_p'	Segundo apellido
'Sexo_p'	Sexo de la persona
'FechaNacimiento_p'	Fecha de nacimiento
'Idioma_p'	Idioma de preferencia (ISO 639-1)
'EstadoCivil_p'	Estado civil
'CodigoPostal_p'	Código postal
'Poblacion_p'	Población / localidad
'Provincia_p'	Código según tabla Provincias (12.1)
'ClaseVia_p'	Clase de vía (Calle, Plaza, etc.)
'TipoIdentificacion_c0'	Tipo de identificación
'IdPersona_ c0'	Identificación (NIF, NIE, Pasaporte) Valor "No identificado" si la persona no dispone de documento identificativo. (TipoIdentificacion = NO)
'IdCliente_ c0'	Id o Código de cliente de compañía
'Nombre_ c0'	Nombre de la persona
'Apellido1_ c0'	Primer apellido
'Apellido2_ c0'	Segundo apellido
'Sexo_ c0'	Sexo de la persona
'FechaNacimiento_ c0'	Fecha de nacimiento

'Idioma_ c0'	Idioma de preferencia (ISO 639-1)
'EstadoCivil_ c0'	Estado civil
'CodigoPostal_ c0'	Código postal
'Poblacion_ c0'	Población / localidad
'Provincia_ c0'	Código según tabla Provincias (12.1)
'ClaseVia_ c0'	Clase de vía (Calle, Plaza, etc.)
'TipoIdentificacion_ t'	Tipo de identificación
'IdPersona_ t'	Identificación (NIF, NIE, Pasaporte) Valor "No identificado" si la persona no dispone de documento identificativo. (TipoIdentificacion = NO)
'IdCliente_ t'	Id o Código de cliente de compañía
'Nombre_ t'	Nombre de la persona
'Apellido1_ t'	Primer apellido
'Apellido2_ t'	Segundo apellido
'Sexo_ t'	Sexo de la persona
'FechaNacimiento_ t'	Fecha de nacimiento
'Idioma_ t'	Idioma de preferencia (ISO 639-1)
'EstadoCivil_ t'	Estado civil
'CodigoPostal_ t'	Código postal
'Poblacion_ t'	Población / localidad
'Provincia_ t'	Código según tabla Provincias (12.1)
'ClaseVia_ t'	Clase de vía (Calle, Plaza, etc.)
'TipoIdentificacion_a'	Tipo de identificación
'IdPersona_ a'	Identificación (NIF, NIE, Pasaporte) Valor "No identificado" si la persona no dispone de documento identificativo. (TipoIdentificacion = NO)
'IdCliente_ a'	Id o Código de cliente de compañía
'Nombre_ a'	Nombre de la persona
'Apellido1_ a'	Primer apellido
'Apellido2_ a'	Segundo apellido
'Sexo_ a'	Sexo de la persona
'FechaNacimiento_ a'	Fecha de nacimiento
'Idioma_ a'	Idioma de preferencia (ISO 639-1)
'EstadoCivil_ a'	Estado civil
'CodigoPostal_ a'	Código postal
'Poblacion_ a'	Población / localidad
'Provincia_ a'	Código según tabla Provincias (12.1)
'ClaseVia_ a'	Clase de vía (Calle, Plaza, etc.)
'Fecha'	Fecha de la póliza
'siniestro'	Información de siniestro
'FraccionPago'	Clase de fraccionamiento de pago en recibos de la póliza

6. DEPURACIÓN DE LOS DATOS

Para extraer información relevante de los datos, es necesario una etapa de preprocesado. En esta etapa el objetivo es filtrar, en la medida de lo posible, aquellos elementos que tengan potencial y darles la forma adecuada para poder trabajar con los modelos.

Aunque podría realizarse la depuración sin explicar el procedimiento, conviene realizar una memoria de los cambios y acciones llevadas a cabo para el posterior análisis de los resultados. En primer lugar, se estudia el documento en busca de errores en la toma de los datos o incongruencias. En el caso de este proyecto, hay una base de datos muy amplia y en la cual hay que hacer una selección de los datos que son útiles para construir el dataset. A continuación, se construye la variable objetivo mediante el programa Python y la ayuda de otro dataset de recibos. Después, se revisan los datos atípicos del conjunto de datos y se valora si es necesario tratarlos o no. Por último, se deben tratar los datos faltantes o missing y decidir cuál es la mejor decisión con respecto a estos.

Casi cualquier software sería válido para la realización de esta fase, pero en este caso la depuración de este conjunto de datos se llevará a cabo, como ya se ha comentado previamente, en Python.

6.1. Etapas de preprocesado

En este apartado se desarrollan todas las etapas que se han realizado para la construcción de la variable objetivo, la revisión de los datos atípicos y el tratamiento de los valores missing.

6.1.1. Construción de la variable objetivo.

Para la construcción de la variable objetivo, no solo se usa el dataset descrito en el apartado 5, se usa también un dataset proporcionado por la compañía de seguros donde se describen los datos de los recibos de la siguiente forma:

Tabla 3 Descripción variables dataset recibos

VARIABLE	DESCRIPCIÓN
IdRecibo	Número de recibo
SituacionRecibo	Situación actual del recibo
ClaseRecibo	Clase de recibo
Fechas	Fechas del recibo
GestionCobro	Datos de la gestión de cobro del recibo
DatosImportes	Datos de importes
DatosComisiones	Datos de las comisiones
MovimientosRecibo	Datos de movimientos del recibo
IdRemesa	Referencia remesa (si se ha remesado)
RiesgosRecibo	Riesgos del recibo

La columna “clasederecibo” contiene información sobre el estado en el que se encuentra el recibo siendo:

Tabla 4 Descripción variable "clasedemovimiento"

CAMPO	DESCRIPCIÓN
AE	Anulación de extorno
AN	Anulación
CO	Cobro
DE	Devolución
DL	Devolución Liquidación
EM	Emisión
LI	Liquidación

En primer lugar, se decide dar valor 1, considerándoles impagadores, a todos aquellos valores que no sean de cobro y por lo tanto les falte algo por pagar. Por otro lado, 0 a todos aquellos que sean de cobro "CO". Y por último 2 a todos aquellos valores que están vacíos.

Por último, ya que los IdPoliza son los mismos en los dos dataset utilizados, se recorren cada una de las columnas en el dataset original (apartado 5) y se crea una nueva columna con los valores de 0, 1 y 2 explicados previamente. Siendo, por tanto: 0 pagador, 1 impagador y 2 missing o "no se sabe".

6.1.2. Correlación de valores vacios en la variable objetivo.

Para saber si los valores "2" en este caso, valores vacíos de nuestra variable objetivo, predicen o sirven de algo dentro de las predicciones de los modelos, se realizan una serie de modelos Random forest con cada una de las variables predictoras para comprobarlo.

El estudio se realiza tanto con tres valores "pagado, no pagado y no se sabe" como con dos valores. Este último se lleva a cabo nombrando a los valores 0 y 1 como 0 y a los valores 2 como 1. Los resultados son los siguientes:

6.1.2.1. Random Forest con 3 tipos de datos en la variable objetivo y variable predictora IdPoliza.

```
Correct classification rate: 0.6273830155979203
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       436
           1       0.00      0.00      0.00       209
           2       0.63      1.00      0.77      1086

    accuracy                           0.63      1731
   macro avg       0.21      0.33      0.26      1731
weighted avg       0.39      0.63      0.48      1731

Confusion matrix, without normalization
[[   0    0  436]
 [   0    0  209]
 [   0    0 1086]]
```

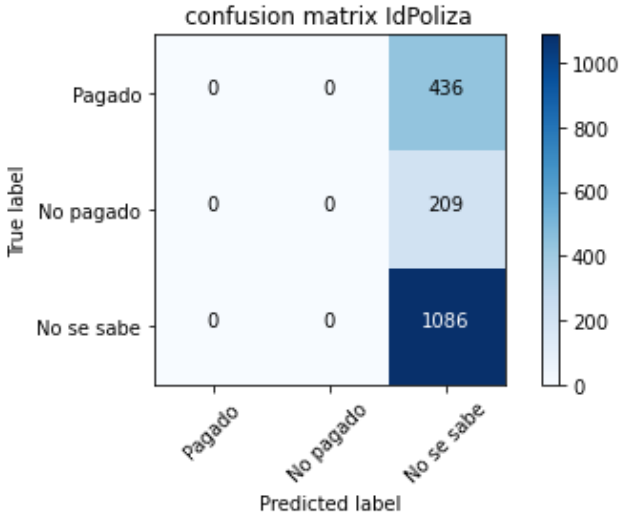


Figura 7 Random Forest 3 variables

La Figura 7 muestra una matriz de confusión. Esta indica que el modelo de Random Forest ha clasificado a todos nuestros datos como valores “no se sabe”. Esto se debe a que el modelo al ver un número elevado de valores “no se sabe” siempre tiende a clasificar a todos los datos hacia la mayoría.

6.1.2.2. Random forest con 2 clases tipos de datos en la variable objetivo y variables IdPoliza, Matrícula y FechaVencimiento.

```
Correct classification rate: 0.6873111782477341
              precision    recall  f1-score   support

           0       0.69      1.00      0.81       455
           1       0.00      0.00      0.00       207

    accuracy                           0.69       662
   macro avg       0.34      0.50      0.41       662
weighted avg       0.47      0.69      0.56       662

Confusion matrix, without normalization
[[455   0]
 [207   0]]
```

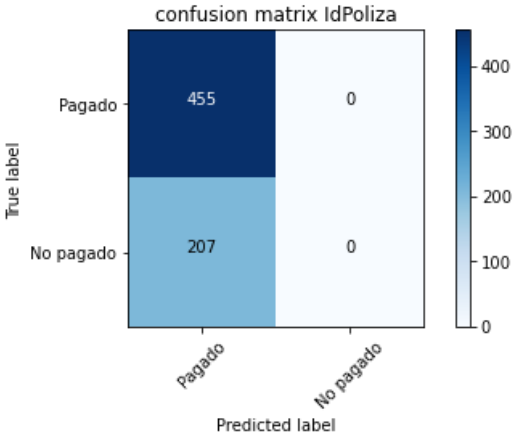


Figura 8 Matriz de confusión para el algoritmo de Random Forest utilizando 2 variables / IdPoliza

La Figura 8 muestra el mismo resultado que la **¡Error! No se encuentra el origen de la referencia.**. El modelo clasifica hacia los datos mayoritarios. En el caso de IdPoliza hacia los valores de 0 y 1 que están incluidos en el nivel de “pagado”.

```
Correct classification rate: 0.6858006042296072
              precision    recall  f1-score   support

           0       0.69      1.00      0.81       454
           1       0.00      0.00      0.00       208

    accuracy                           0.69       662
   macro avg       0.34      0.50      0.41       662
weighted avg       0.47      0.69      0.56       662

Confusion matrix, without normalization
[[454   0]
 [208   0]]
```

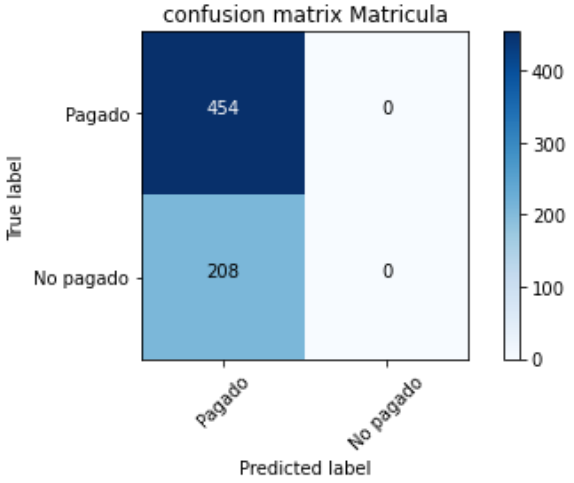


Figura 9 Random Forest 2 variables / Matricula

La Figura 9 mediante la matriz de confusión expone que los datos “no se sabe” siguen sin ser predictores para ninguna de nuestras variables.

```
Correct classification rate: 0.6873111782477341
              precision    recall  f1-score   support

           0       0.69      1.00      0.81       455
           1       0.00      0.00      0.00       207

    accuracy                           0.69       662
   macro avg       0.34      0.50      0.41       662
weighted avg       0.47      0.69      0.56       662

Confusion matrix, without normalization
[[455   0]
 [207   0]]
```

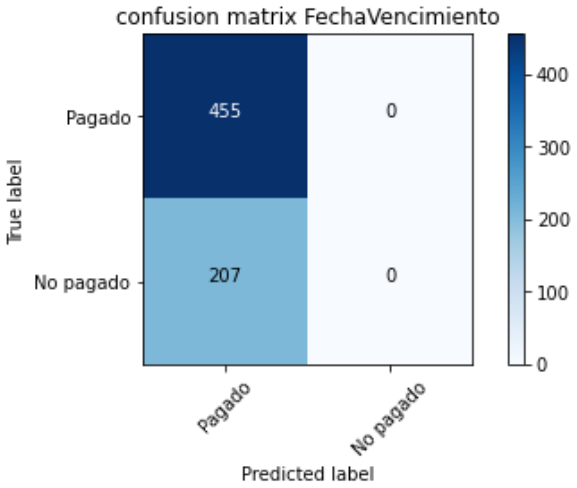


Figura 10 Random Forest 2 variables / FechaVencimiento

En la Figura 10 mediante la matriz de confusión expone que los datos "no se sabe" siguen sin ser predictores para ninguna de nuestras variables al igual que en la Figura 9 Random Forest 2 variables / Matricula.

Por tanto, después de comprobar que para ninguna de las variables predictoras los valores vacíos dentro de la variable objetivo son predictoras o por el contrario indican algo dentro de alguna de esas variables, se decide eliminarlos.

6.1.3. Estudio de los datos atípicos.

En el caso de este proyecto y más específicamente por el dataset utilizado, no se encuentran datos atípicos debido a que la información recogida y suministrada por la compañía de seguros sigue un patrón y estructura que no permite que se generen ese tipo de datos.

6.1.4. Filtrado de NaNs y eliminación de variables.

Para el filtrado de NaNs se estudia, en primer lugar, el porcentaje de valores missing que tiene cada variable. Todas aquellas columnas o variables que son de clase "a = asegurado, p = propietario, C0 = conductor" tienen un porcentaje de missing muy altos tal y como se muestra en la Figura 11 y por ello se decide eliminar las variables que contienen esa clase.

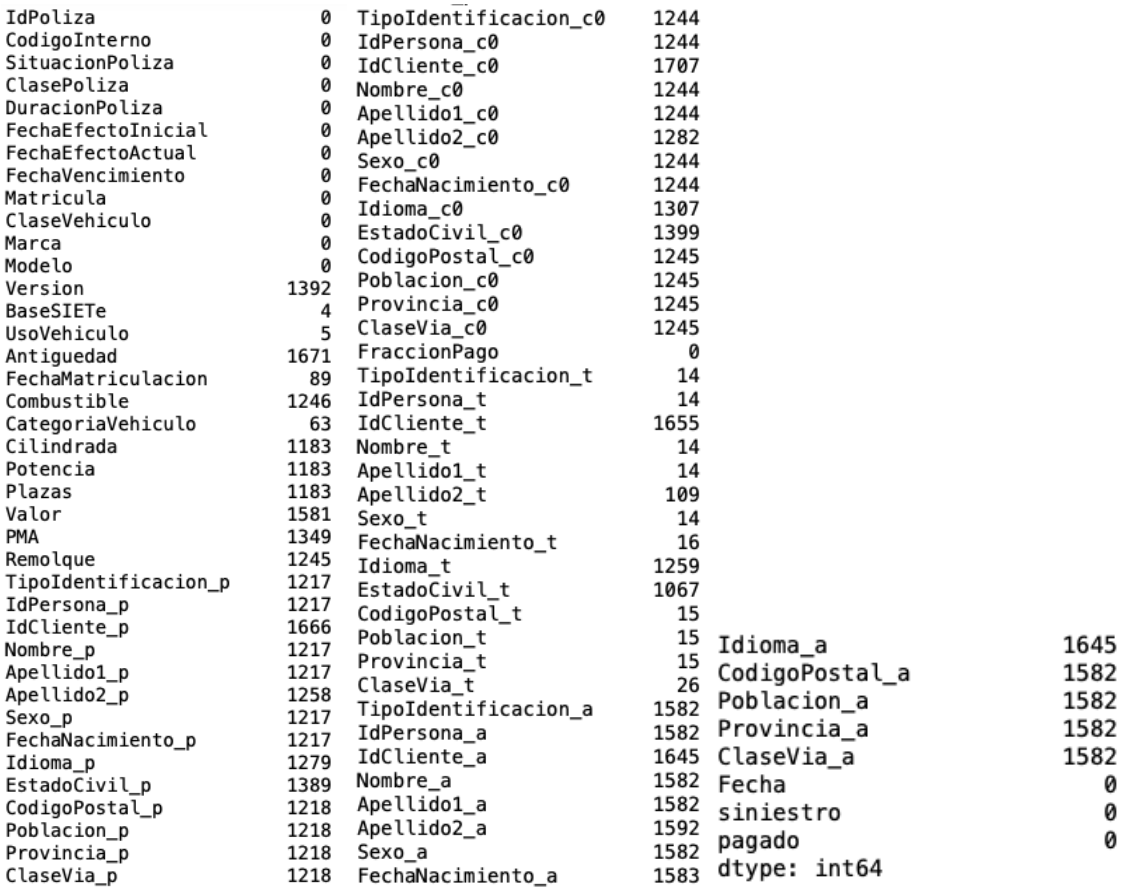


Figura 11 Número de missing por variable

En la Figura 11 se observa el número de valores missing que hay en cada una de las variables. (Previo a la eliminación de las variables de "a = asegurado, p = propietario, C0 = conductor").

A continuación se nombran las variables restantes una vez eliminado las clases de asegurado, propietario y conductor: 'SituacionPoliza', 'ClasePoliza', 'DuracionPoliza', 'FechaEfectoInicial', 'FechaEfectoActual', 'FechaVencimiento', 'Matricula', 'ClaseVehiculo', 'Marca', 'Modelo', 'Version', 'BaseSIETe', 'UsoVehiculo', 'Antiguedad', 'FechaMatriculacion', 'Combustible', 'CategoriaVehiculo', 'Cilindrada', 'Potencia', 'Plazas', 'Valor', 'PMA', 'Remolque', 'FraccionPago', 'TipoIdentificacion_t', 'Sexo_t', 'FechaNacimiento_t', 'Idioma_t', 'EstadoCivil_t', 'CodigoPostal_t', 'Poblacion_t', 'Provincia_t', 'ClaseVia_t', 'Fecha', 'siniestro', 'pagado'.

Una vez que se han eliminado las variables se pasa al estudio y tratamiento de los valores missing del resto de ellas. Los porcentajes de valores missing que nos encontramos son los siguientes:

```
IdPoliza                0.000000
SituacionPoliza         0.000000
ClasePoliza             0.000000
DuracionPoliza          0.000000
FechaEfectoInicial      0.000000
FechaEfectoActual       0.000000
FechaVencimiento        0.000000
Matricula               0.000000
ClaseVehiculo           0.000000
Marca                   0.000000
Modelo                  0.000000
BaseSIETe               7.123248
UsoVehiculo             0.086693
FechaMatriculacion      2.441844
CategoriaVehiculo       1.892790
FraccionPago            0.000000
TipoIdentificacion_t    0.809132
Sexo_t                  0.809132
FechaNacimiento_t       0.881376
CodigoPostal_t          0.823580
Poblacion_t             0.823580
Provincia_t             0.823580
ClaseVia_t              8.568126
Fecha                   0.000000
siniestro               0.000000
pagado                  0.000000
dtype: float64
```

Figura 12 Porcentaje de missing

La Figura 12 muestra que las variables que contienen valores missing lo hacen con porcentajes muy bajos, siendo, por ejemplo, de un 0,08% en "UsoVehiculo" y un 7,1% como valor más alto en "BaseSIETe".

Debido a la poca importancia de estos valores dentro de cada variable se decide eliminar los valores missing.

```
IdPoliza                0.0
SituacionPoliza         0.0
ClasePoliza             0.0
DuracionPoliza          0.0
FechaEfectoInicial      0.0
FechaEfectoActual       0.0
FechaVencimiento        0.0
Matricula               0.0
ClaseVehiculo           0.0
Marca                   0.0
Modelo                  0.0
BaseSIETe               0.0
UsoVehiculo             0.0
FechaMatriculacion      0.0
CategoriaVehiculo       0.0
FraccionPago            0.0
TipoIdentificacion_t    0.0
Sexo_t                  0.0
FechaNacimiento_t       0.0
CodigoPostal_t          0.0
Poblacion_t             0.0
Provincia_t             0.0
ClaseVia_t              0.0
Fecha                   0.0
siniestro               0.0
pagado                  0.0
dtype: float64
```

Figura 13 Porcentaje de missing después de la eliminación de ellos.

En la Figura 13 se observa que, tras el filtrado y eliminación, no hay ningún missing dentro de las variables predictoras.

6.1.5. Codificación

Para el tratamiento de los modelos se va a convertir a todas las variables categóricas en numéricas y, para ello, a través de una codificación mediante el programa Python se transforman los datos.

Esta codificación se hace a través de un proceso de la librería Sklearn el cual agrupa los datos y los convierte en números en función de los valores.

La codificación se produce en las siguientes variables de la siguiente forma:

Tabla 5 Codificación variables categóricas a numéricas

DESCRIPCIÓN	VALORES
CategoriaVehiculo	0, 1
ClasePoliza	0, 1
ClaseVehiculo	0, 1, 2, 3, 4, 5, 6, 7
ClaseVia_t	0, 1, … 51, 52
DuracionPoliza	0
FraccionPago	0, 1, 2
IdPoliza	0, …, 2028
Marca	0, …, 149
Matricula	0, …, 2407
Modelo	0, …, 1147
Poblacion_t	0, …, 1295
Sexo_t	0, 1
SituacionPoliza	0, 1, 2, 3
TipoIdentificacion_t	0, 1, 2
UsoVehiculo	0, 1

7. MODELIZACIÓN

Una vez explicada y analizada la etapa de preprocesado, se pasa a una fase de modelización de los datos que tiene como objetivo conseguir el mejor resultado modelando los datos a través de diferentes algoritmos de aprendizaje automático en diferentes etapas.

7.1. Etapa 1 - Variables Edad y Antigüedad de poliza.

Para seguir alimentando el dataset y con la intención de conseguir unos mejores resultados, se decide sacar dos nuevas variables, fecha y antigüedad de póliza, a través de las otras variables que ya existen.

La edad se obtiene a partir de la resta de las variables “fecha actual” y “fecha de nacimiento”.
En cuanto a la antigüedad de la póliza, se recorren todas aquellas pólizas con un mismo Id y se suma 1 a los años en los que se emitió. Siendo 1 el primer año, 2 el segundo, 3 el tercero y así sucesivamente con los años que el tomador haya emitido la póliza. En la Figura 14 se muestra un ejemplo final de estas dos variables incluidas en el dataset.

edad	antiguedad_pol
58	0
58	1
58	2
54	0
54	1
...	...
46	0
51	0
46	0
51	0
43	0

Figura 14 Nuevas variables edad y antigüedad de la póliza

Una vez incluidas en el dataset, se estudia la importancia de éstas dos nuevas variables con respecto a la variable objetivo.

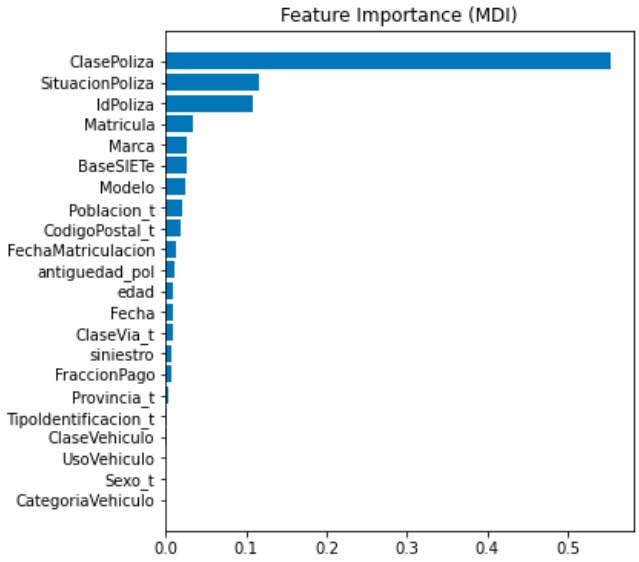


Figura 15 Importancia de las variables

En la Figura 15 se muestra la importancia de cada una de las variables en el modelo, la escala de valores se mueve entre 0 y 1 y la longitud de la barra azul indica la importancia

medida como valor de la impureza de disminución media (MDI) que calcula la media y la desviación estándar de la acumulación de la disminución de impurezas dentro de cada árbol. En el "eje y" se nombran todas las variables. Se observa que las variables que más influyen en la precisión del modelo son "Clase Póliza, Situación de la Póliza e id Póliza".

Se observa que la variable predictora "ClasePoliza" es la variable que más importancia tiene con respecto a la variable objetivo con mucha diferencia llegando su barra de impureza de disminución media por encima de 0,5. Por su parte, "CategoriaVehiculo" es la que menos importancia presenta.

Se observa como las nuevas dos variables no tienen una gran importancia dentro de las predicciones y se lanzan los modelos para estudiar sus resultados.

7.1.1. Regresión logística

```
Correct classification rate: 0.4205729166666667
              precision    recall  f1-score   support

           0       0.42      1.00      0.59       646
           1       0.00      0.00      0.00       890

    accuracy                           0.42      1536
   macro avg       0.21      0.50      0.30      1536
weighted avg       0.18      0.42      0.25      1536

Confusion matrix, without normalization
[[646   0]
 [890   0]]
```

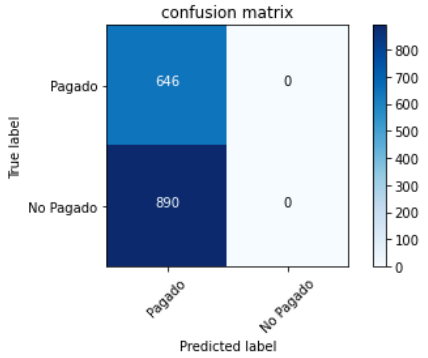


Figura 16 Modelo Regresión logística con nuevas variables de edad y antigüedad de la póliza

La Figura 16 expone una matriz de confusión. Esta nos indica que el modelo clasifica bien a los 646 datos de pagadores con un 100% de recall pero que no consigue clasificar a ninguno de los 890 datos a impagadores.

7.1.2. Random Forest

```
Correct classification rate: 0.8161648177496038
              precision    recall  f1-score   support

           0       0.80      0.97      0.88       419
           1       0.89      0.51      0.65       212

    accuracy                           0.82       631
   macro avg       0.85      0.74      0.76       631
weighted avg       0.83      0.82      0.80       631

Confusion matrix, without normalization
[[406  13]
 [103 109]]
```

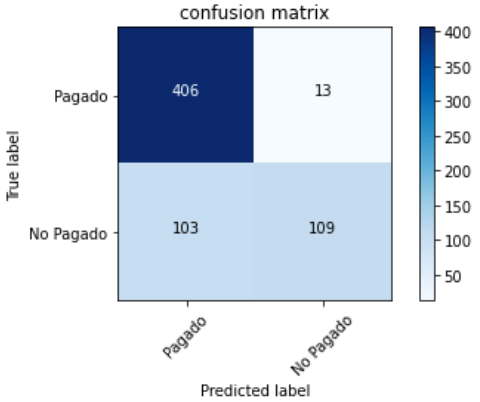


Figura 17 Modelo Random Forest con nuevas variables de edad y antigüedad de la póliza

En la Figura 17 se observa una matriz de confusión con los resultados del modelo de Random Forest. Esta indica que el modelo clasifica bien a 406 datos de pagadores con un 97% de recall pero que tan solo consigue clasificar a 109 de los 631 datos a impagadores.

7.1.3. Gradient Boosting

```
Correct classification rate: 0.820919175911252
              precision    recall  f1-score   support

           0       0.80      0.98      0.88       424
           1       0.94      0.49      0.64       207

    accuracy                           0.82       631
   macro avg       0.87      0.74      0.76       631
weighted avg       0.84      0.82      0.80       631

Confusion matrix, without normalization
[[417   7]
 [106 101]]
```

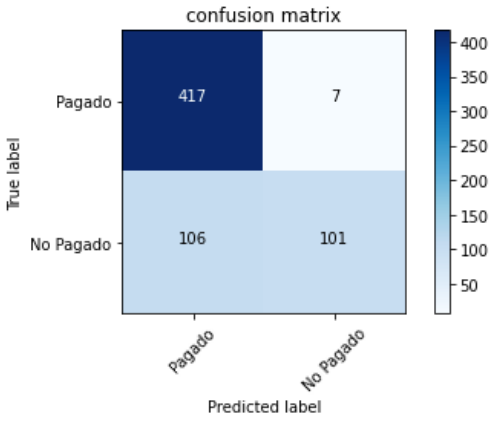


Figura 18 Modelo Gradient Boosting con nuevas variables de edad y antigüedad de la póliza

En la Figura 18 se ve que el modelo mejora con respecto al de regresión logística visto previamente en la Figura 16,. pero aun así no se consiguen buenos resultados. Se consigue clasificar bien a casi la mitad de los datos de impagadores (101) con un recall de un 49%.

Tal y como se puede observar en los resultados de las figuras 15, 16 y 17, se sigue sin obtener una buena clasificación de impagadores en ninguno de los modelos, por lo que se procede a seguir modelando los datos para acercarse a unos resultados que sean correctos.

7.2. Etapa 2 – Base Siete

En esta etapa lo que se pretende conseguir es la alimentación del dataset con una nueva serie de datos del dataset de Base Siete.

El dataset Base Siete, contiene información relativa al vehículo con una serie de variables que se exponen a continuación:

Tabla 6 Descripción variables dataset BaseSIETe

VARIABLE	DESCRIPCIÓN
CodTipoVehiculo	Tipo de vehículo
CodCategoría	Categoría de vehículo
CodModelo	Modelo de vehículo

CodProhibición	Código de prohibición
Version	Versión del modelo
Puertas	Número de puertas
Chasis	Chasis
Techo	Tipo de techo
Potencia	Potencia del vehículo
Combustible	Tipo de combustible
Tara	Tara
PMA	Peso máximo autorizado
Cilindrada	Cilindrada del vehículo
FechaLanzamiento	Fecha de lanzamiento
Plazas	Número de plazas
ClaseVehiculo	Clase de vehículo
CodVersion	Código de versión del vehículo
Volumen	Volumen del vehículo
Longitud	Longitud del vehículo
Caja	Número de caja
Activo	Activo
PermiteCarnetB	Permite o no carnet tipo B

Una vez incluidas las nuevas variables en nuestro dataset, se realiza la codificación de los datos a través de la librería Sklearn tal y como se ha hecho previamente con el resto de los datos. Las codificaciones resultan de la siguiente manera:

Tabla 7 Codificación variables categóricas a numéricas BaseSIETe

DESCRIPCIÓN	VALORES
CodProhibicion	-1, 0, 1, 2, 3, 4
Activo	-1, 0
Combustible	-1, 0, 1, 2
Versión	-1, …, 60

A continuación, se hace una limpieza de estos con especial hincapié en los datos faltantes o missing. Se observan los siguientes resultados:

```
CodVersion              483
CodTipoVehiculo         483
CodCategoria            483
CodModelo               483
Version                   0
Puertas                 483
Potencia                483
Combustible               0
Tara                    483
Pma                     483
Cilindrada              483
Volumen                 483
Chasis                 6144
Longitud               6144
Caja                   6144
Techo                  6144
FechaLanzamiento       1375
Plazas                  483
CodProhibicion            0
Activo                    0
PermiteCarnetB         6144
```

Figura 19 Número de datos missing

En la Figura 19 se ve como la mayoría de las nuevas variables contienen valores missing. Las únicas que no contienen son las variables “Version”, “Combustible”, “CodProhibicion” y “Activo”.

Debido al gran número de valores missing se decide eliminar las variables 'Chasis', 'Longitud', 'Caja', 'Techo', 'FechaLanzamiento', 'PermiteCarnetB', 'Puertas', 'Tara' ya que se considera que no hay suficientes datos como para que el modelo trabaje con ellas.

```
CodVersion              483
CodTipoVehiculo         483
CodCategoria            483
CodModelo               483
Version                   0
Potencia                483
Combustible               0
Pma                     483
Cilindrada              483
Volumen                 483
Plazas                  483
CodProhibicion            0
Activo                    0
```

Figura 20 Resto variables BaseSIETe

En la Figura 20 se observan las variables que se van a incluir en el dataset y el número de valores missing que tienen cada una de ellas, procediendo a la limpieza.

Para la limpieza de esas variables se decide dar un valor de -1 a los valores missing. De esta forma se obtiene el nuevo dataset con los datos de BaseSIETe incluidos.

Una vez realizada esa inclusión y limpieza de los datos se procede a realizar un estudio de la importancia de las variables. En la Figura 21 se observa como matrícula y población son las dos variables que más influyen en los modelos. Pudiendo interpretar que cuanto más nueva sea una matrícula menos impagos se van a producir y mucho más fácil es predecir el impago.

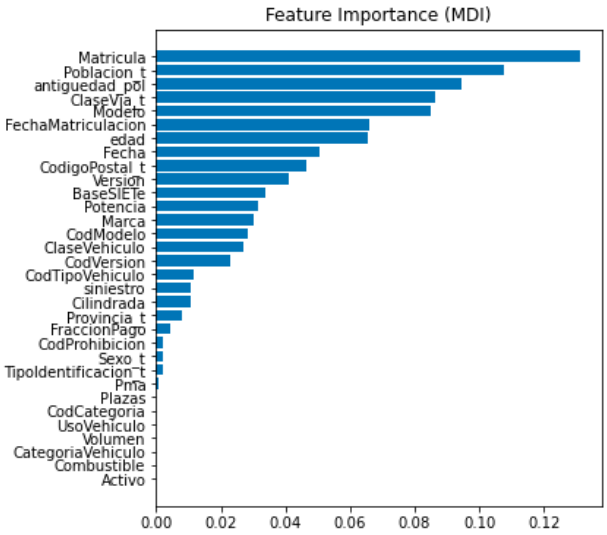


Figura 21 Importancia de las variables tras modelo de Gradient Boosting.

Además, mediante una matriz de correlación se estudia cuáles de las variables están más correlacionadas entre sí y con la variable objetivo (en este caso llamada “label”)

Figura 22 Matriz de correlación

En la Figura 22 se observa tanto en el "eje x" como en el "eje y" todas las variables predictoras y la variable objetivo, llamada "label". Los valores positivos son aquellos que aparecen con un color azul, siendo más azules cuanto más cerca estén del uno. Por otro lado, los valores negativos son representados con un color rojo, siendo de un rojo más intenso cuanto más cerca esté de -1. Los valores representan el coeficiente V de Cramer aplicado al chi-cuadrado. Este coeficiente toma valores entre 0 y 1; siendo 0 la inexistencia de relación; y 1, relación total. Se observa como, por ejemplo, la variable "plazas" con la variable "volumen" tienen una relación muy alta, de un 0,97 ya que cuantas más plazas tiene el vehículo más grande suele ser por lo que más capacidad de volumen tiene. En cambio, se puede ver como con respecto a la variable objetivo, no hay ninguna variable que tenga una relación muy alta.

Seguido del estudio de la importancia y relación de las variables, se comprueban los resultados para el modelo de Gradient Boosting en la parte de entrenamiento.

```
Correct classification rate: 0.7662612374405077
              precision    recall  f1-score   support

           0       0.75      0.98      0.85      1273
           1       0.88      0.33      0.48       618

    accuracy                           0.77      1891
   macro avg       0.81      0.65      0.66      1891
weighted avg       0.79      0.77      0.73      1891

Confusion matrix, without normalization
[[1245   28]
 [ 414  204]]
```

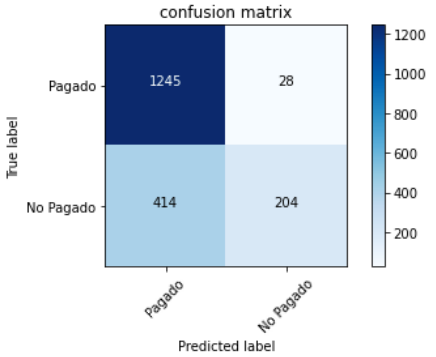


Figura 23 Modelo Gradient Boosting datos entrenamiento

En la Figura 23 la matriz de confusión indica que el modelo clasifica bien a un 85% de los pagadores, pero tan solo un 48% de impagadores por lo que siguen siendo porcentajes muy alejados de lo que se quiere llegar a obtener.

Con los datos test:

```
Correct classification rate: 0.7068145800316957
              precision    recall  f1-score   support

           0       0.72      0.94      0.82       438
           1       0.56      0.18      0.27       193

    accuracy                           0.71       631
   macro avg       0.64      0.56      0.55       631
weighted avg       0.67      0.71      0.65       631

Confusion matrix, without normalization
[[411  27]
 [158  35]]
```

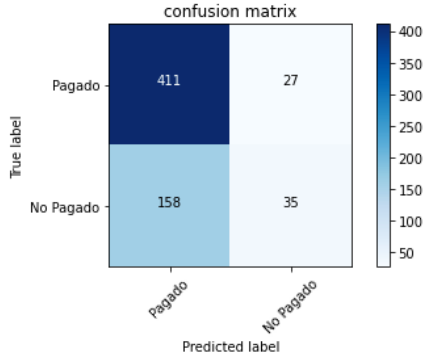


Figura 24 Modelo Gradient Boosting datos test

Al igual que en la Figura 23, en la Figura 24 se muestra que la matriz de confusión no da unos resultados buenos con tan solo un 18% de impagadores bien clasificados.

7.3. Etapa 3 - Balanceo

Debido a que se sigue sin tener unos buenos resultados de predicción de personas que no pagan, se decide realizar un balanceo de los datos mediante un 50% de personas que cumplen con los pagos y 50% de personas que no cumplen con los datos. Esto se realiza debido a que en el dataset se encuentran muchas más observaciones de gente que paga que de la gente que no lo hace. Por ello, se eliminan instancias de la clase mayoritaria para poder igualarla a la minoritaria.

Una vez realizado el balanceo se estudian de nuevo la importancia de las variables.

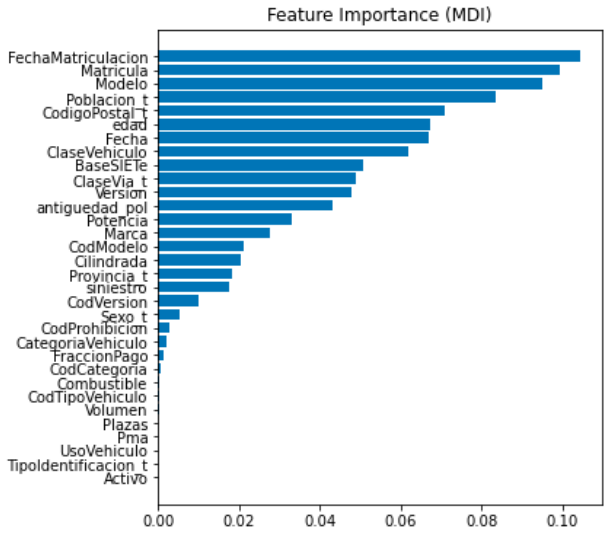


Figura 25 Importancia de las variables tras modelo Gradient Boosting.

En esta ocasión, tal y como vemos en la Figura 25, “FechaMatriculación” es la variable que más importancia tiene dentro del modelo. Esto se puede interpretar de la forma que cuanto más nuevo esté el vehículo, menos se tiende a no pagar, siendo muy fácil la clasificación.

Una vez realizado el balanceo se procede, de nuevo, a realizar los modelos. Los resultados obtenidos son los siguientes:

7.3.1. Gradient Boosting

```
Correct classification rate: 0.5435816164817749
              precision    recall  f1-score   support

           0       0.76      0.50      0.60       438
           1       0.36      0.65      0.47       193

    accuracy                           0.54       631
   macro avg       0.56      0.57      0.53       631
weighted avg       0.64      0.54      0.56       631

Confusion matrix, without normalization
[[217 221]
 [ 67 126]]
```

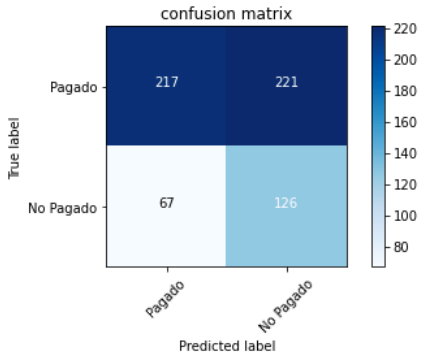


Figura 26 Modelo Gradient Boosting tras balanceo

En la Figura 26 se observa que la clasificación de impagadores empieza a mejorar. La matriz de confusión muestra una clasificación buena del 65% de los impagadores. Por otro lado, se puede observar que, tras el balanceo, la clasificación de los pagadores ha bajado y clasifica bien a un 60%.

7.3.2. Random Forest

```
Correct classification rate: 0.5705229793977813
              precision    recall  f1-score   support

           0       0.80      0.50      0.62       438
           1       0.39      0.72      0.51       193

    accuracy                           0.57       631
   macro avg       0.60      0.61      0.56       631
weighted avg       0.68      0.57      0.59       631

Confusion matrix, without normalization
[[221 217]
 [ 54 139]]
```

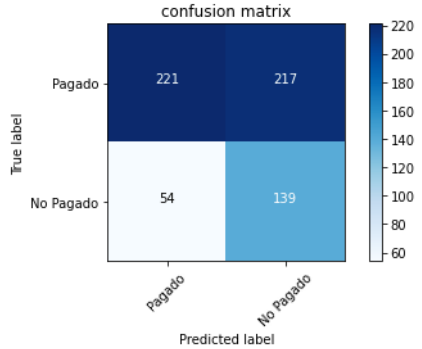


Figura 27 Modelo Random Forest tras balanceo

En el modelo de Random forest, se observa que la matriz de confusión de la Figura 27 muestra una clasificación correcta del 72% de los impagadores y un 50% de los pagadores. Se repite la mejora en la clasificación de los impagadores, pero una disminución en los pagadores.

7.3.3. Regresión Logística

```
Correct classification rate: 0.6465927099841522
              precision    recall  f1-score   support

           0       0.69      0.88      0.78       438
           1       0.30      0.11      0.16       193

    accuracy                           0.65       631
   macro avg       0.50      0.50      0.47       631
weighted avg       0.57      0.65      0.59       631

Confusion matrix, without normalization
[[386  52]
 [171  22]]
```

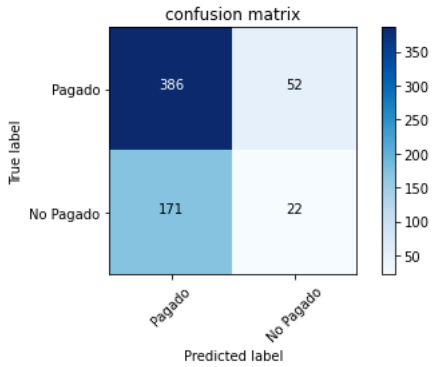


Figura 28 Modelo Regresión logística tras balanceo

En la matriz de confusión de la Figura 28 se observa una clasificación de un 11% de los impagadores de forma correcta y un 88% de los pagadores. En el caso de este modelo, los impagadores no mejoran.

7.3.4. Bagging

```
Correct classification rate: 0.5768621236133122
              precision    recall  f1-score   support

           0       0.81      0.51      0.62       438
           1       0.40      0.74      0.52       193

    accuracy                           0.58       631
   macro avg       0.60      0.62      0.57       631
weighted avg       0.69      0.58      0.59       631

Confusion matrix, without normalization
[[222 216]
 [ 51 142]]
```

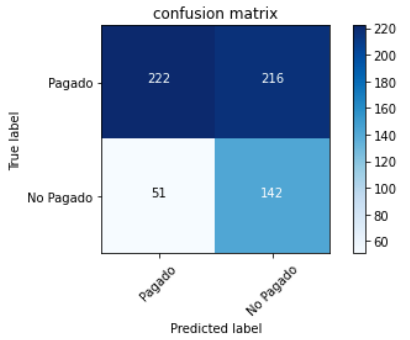


Figura 29 Modelo Bagging tras balanceo

En nuestro último modelo estudiado tras el balanceo, se muestra en la Figura 29, que al igual que en anteriores modelos, la clasificación de los impagadores mejora con un 74% de buena clasificación mientras que los pagadores bajan a un 51% de buena clasificación.

Tras observar cada uno de los modelos y tras haber realizado el balanceo, se afirma que los resultados de clasificación de los impagadores mejoran considerablemente pero que, en cambio, la clasificación de los pagadores ha bajado rondando el 50% en todos los modelos excepto en el de regresión logística que no empeora.

7.4. Etapa 4 – Método Ensemble

Tras la obtención de las predicciones de los mejores modelos; estas pueden combinarse para reducir la varianza del error. Por ello, para seguir mejorando los resultados, en esta última etapa se realiza un método de ensamblado.

Los métodos de ensamblado (métodos de ensemble) utilizan múltiples algoritmos de aprendizaje para obtener un rendimiento predictivo que mejore el que podría obtenerse por medio de cualquiera de los algoritmos de aprendizaje individuales que lo constituyen.

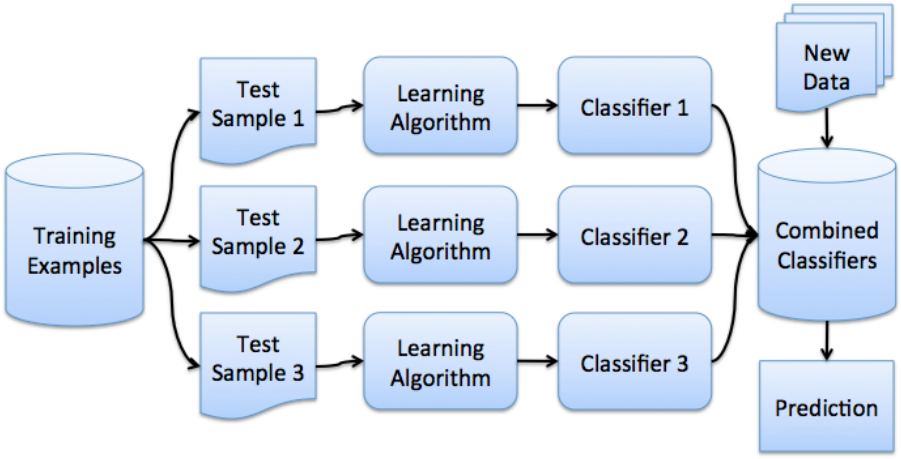


Figura 30 Método Ensemble

La Figura 30 muestra cómo funciona el ensamblado. Con los conjuntos de entrenamiento se realizan diferentes modelos mediante diferentes algoritmos de clasificación. Por último, una vez que tenemos esos resultados los combinamos para obtener una combinación de todas ellas.

En este caso se cogen las 3 mejores predicciones (Bagging, Random forest y Gradient Boosting) que se han obtenido tras la realización del balanceo, vistas en el Etapa 3 - Balanceo y se realiza el ensamblado.

Se obtiene el siguiente resultado:

```
Correct classification rate: 0.595879556259905
              precision    recall  f1-score   support

           0       0.76      0.58      0.66       426
           1       0.42      0.62      0.50       205

    accuracy                           0.60       631
   macro avg       0.59      0.60      0.58       631
weighted avg       0.65      0.60      0.61       631

Confusion matrix, without normalization
[[248 178]
 [ 77 128]]
```

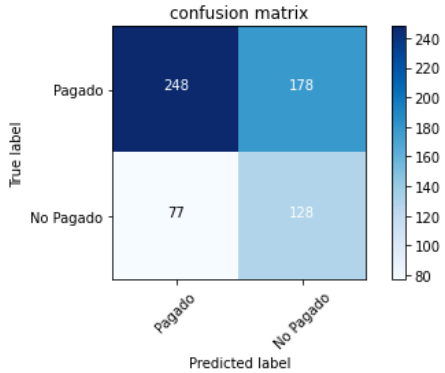


Figura 31 Modelo final método ensemble

Tras la realización del ensamblado, se puede observar en la matriz de confusión de la Figura 31 que los resultados de clasificación son de un 62% para los impagadores,

clasificando a 128 de 205 datos bien y un 58% para los pagadores, clasificando bien a 248 de 426 datos.

8. SELECCIÓN DEL MEJOR MODELO

Para la consecución del proyecto se han realizado diferentes etapas para alimentar el dataset e ir mejorando los modelos. Los resultados obtenidos en las diferentes etapas son los mostrados en la Tabla 8 Resumen de modelos.

Tabla 8 Resumen de modelos

MODELO	RECALL PAGADORES	RECALL IMPAGADORES	ACCURACY
Regresión Logística nuevas variables edad y antigüedad de póliza	100%	0%	42%
Random Forest nuevas variables edad y antigüedad de póliza	97%	51%	82%
Gradient Boosting nuevas variables edad y antigüedad de póliza	98%	49%	82%
Gradient Boosting BaseSIETe	94%	18%	71%
Gradient Boosting balanceo	50%	65%	54%
Random Forest balanceo	50%	72%	57%
Regresión logística balanceo	88%	11%	65%
Bagging balanceo	51%	74%	58%
Ensamblado	58%	62%	60%

Se puede observar que el modelo más equilibrado es el obtenido tras realizar el ensamblado con un 42% de error en los pagadores y un 38% en los impagadores.

Se puede ver a su vez, que, en la etapa de balanceo de los datos, los resultados han mejorado considerablemente con respecto a la etapa en la que se ha incluido al dataset los datos de BaseSIETe; sobre todo en los resultados de clasificación de los impagadores, pasando de un 18% a un 65% en los modelos de Gradient Boosting siendo la mejora de un 47%.

Además, se puede decir que los modelos de regresión logística son los que menos funcionan para clasificar a los impagadores, siendo un 11% en el de balanceo y un 0% al introducir las nuevas variables de antigüedad de póliza y edad del tomador.

Si se estudia el "accuracy" -medida utilizada para determinar qué modelo es mejor para identificar relaciones y patrones entre variables en un conjunto de datos en función de los datos de entrada o de entrenamiento- se observa que se obtiene el mejor resultado en el modelo de Gradient Boosting, una vez introducidas las variables de edad del tomador y antigüedad de la póliza, y Random Forest también una vez introducidas las variables de edad del tomador y antigüedad de la póliza con un 82%.

Por último, hablando de la elección del mejor modelo se pueden sacar varias conclusiones. En primer lugar, no se puede afirmar que ninguno de los modelos sea mejor que otro en general ya que de ello depende los objetivos empresariales que se quieran alcanzar. Si se quiere tener una precisión muy alta clasificando a los impagadores habría que escoger el modelo de bagging tras balanceo (color rojo en la Tabla 8) con una clasificación del 72% de los datos de forma correcta. Si, por el contrario, se pretende escoger un modelo que sea más equilibrado y clasifique de forma correcta aproximadamente al mismo número de pagadores que de impagadores habría que escoger el modelo de ensamblado (color verde en la Tabla 8).

9. CONCLUSIONES

Al comienzo de este proyecto, se planteaba el objetivo de definir un modelo que predijese cuál de los clientes va a generar un impago. Y para lograrlo, se ha trabajado con entrenamiento automático que se define en el apartado 3.1 de este trabajo.

Los modelos de predicción utilizados han sido Regresión logística, Random Forest, Bagging y Gradient Boosting, y para el trabajo y precisión de los resultados, se ha utilizado una división de un 75% de datos destinado a entrenamiento y un 25% destinado a test en cada uno de los modelos.

El trabajo se ha realizado en diferentes etapas con el objetivo de alimentar los modelos e ir consiguiendo en cada etapa unos mejores resultados.

Previamente a estas etapas, se ha hecho una preparación del dataset con el que se ha trabajado, sacando la variable objetivo mediante la unión de dos bases de datos y definiendo los datos con un 0 en el caso de ser pagador y un 1 en el caso de no serlo. Además, durante esta preparación del dataset con el que se quería trabajar se han eliminado algunas variables ya que no eran importantes o ya que esa información que proporcionaban ya lo hacían otras variables. Por último, se han eliminado los datos missing dentro de las variables con las que se ha trabajado posteriormente y se han codificado los datos para que todas las variables categóricas pasasen a ser numéricas.

En cuanto a las etapas de trabajo, se definen cuatro. En la primera etapa (descrita en el apartado 7.1) se alimenta el dataset introduciendo dos variables nuevas, la variable "edad del tomador" y la variable "antigüedad de la póliza". Se estudian los modelos consiguiendo el mejor resultado en un modelo de Gradient Boosting que reporta unos resultados de una clasificación de un 98% de pagadores, un 49% de impagadores y un 82% de accuracy. En la etapa 2 (apartado 7.2) se vuelve a alimentar el dataset, pero esta vez introduciendo y conexionando los datos de nuestro dataset original con otro llamado "BaseSIETe" que contiene información relativa al vehículo. Y para esta etapa se consiguen unos resultados que no son buenos y que, en este caso empeoran a los anteriores, reportando una clasificación buena de tan solo un 14% de los impagadores. Durante la etapa 3 se ha realizado un balanceo de datos ya que la clase de pagadores tenía un numero mucho mayor de datos que la clase impagadores. Este balanceo se ha hecho igualando al 50% los datos de pagadores e impagadores. Y los resultados obtenidos son una mejora muy significativa en la clasificación de los impagadores mejorando un 47% en los modelos de Gradient Boosting con respecto a los resultados obtenidos en la etapa 2 cuando se ha introducido los datos de "BaseSIETe". Por último, en la etapa 4 (apartado 7.4) se realiza un ensamblado de los datos cogiendo las 3 mejores predicciones (Bagging, Random forest y Gradient Boosting) que se ha obtenido tras la realización del balanceo. Y se reportan unos resultados finales mucho más equilibrados que en los modelos de las etapas anteriores acercándose a la clasificación de forma correcta a 6 de cada 10 individuos y un accuracy de ese último modelo de un 60%.

Si nos centramos en cual es el mejor modelo, como se comenta en el apartado anterior, el mejor modelo habría que elegirlo en función de los objetivos empresariales que se

quieren conseguir. Aun así, a pesar de que se ha conseguido acercar a unos resultados que empiezan a ser razonables; debido a la escasez de los datos que nos ha proporcionado la aseguradora, no podemos concluir de esta forma el estudio y la construcción del modelo, ya que los resultados no son perfectos y están lejos de predecir cual de nuestros clientes generará en un futuro un impago.

Para la mejora de nuestros modelos y la consecución de nuestros objetivos, durante los próximos meses con la posible inclusión de nuevos datos por parte de nuevas aseguradoras realimentaremos y mejoraremos los modelos queriendo que los modelos finales clasifiquen de forma correcta a 9 de cada 10 individuos.

10. TRABAJO FUTURO

El objetivo empresarial marcado termina en el mes de diciembre de 2022. Durante los próximos meses se espera la incorporación al proyecto de una compañía más grande con la cual se pretende aumentar el tamaño del dataset, lo que supondrá una mejora en los entrenamientos de los modelos. Además, durante los próximos meses se seguirá probando nuevos modelos con nuevos parámetros para ver cuál de todos responde mejor, así como desarrollando etapas que determinar si con ellas los resultados mejoran.

A partir del mes de enero de 2023 se espera poner en marcha la prueba piloto con la empresa con la que se colabora.

11. BIBLIOGRAFÍA

Bilgin, A., Ellson, J., Gansner, E., Hu, Y., North, S. y contribuciones. Graph- viz. https://graphviz.org/

Breiman, L. Bagging predictors. vol. 421, páginas –20, 1994. Chakure, A. Random forest regression. https://medium.com/swlh/random-forest-and-its-implementation-71824ced454f, 2019.

Copeland, B. Artificial intelligence. Encyclopædia Britannica, (2), páginas –24, 2020.

Cournapeau, D. Scikit-learn. https://scikit-learn.org/stable, 2012.

GeeksforGeeks. Understanding logistic regression. https://www.geeksforgeeks.org/understanding-logistic-regression/, 2019.

Glander, S. Machine learning basics - gradient boosting xgboost. https://www.shirin-glander.de/2018/11/ml_basics_gbm/, 2018.

Herman, J., Usher, W., Mutel, C., Trindade, B., Hadka, D., Woodruff, M., Rios, F., Hyams, D. y xantares. Salib - sensitivity analysis library in python. https://salib.readthedocs.io/en/latest/

De la Hoz Manotas, A., De la Hoz Correa, E., Mendoza, F., Morales, R. y Sanchez, B. Obesity level estimation software based on decision trees. Journal of Computer Science, vol. 15, páginas –10, 2019.

Iooss, B. y Lemaître, P. A Review on Global Sensitivity Analysis Methods, páginas 101–122. Springer US, Boston, MA, 2015. ISBN 978-1-4899-7547-8.

Khalaf, M., Hussain, A. J., Keight, R., Al-Jumeily, D., Fergus, P., Keenan, R. y Tso, P. Machine learning approaches to the application of disease modifying therapy for sickle cell using classification models. Neurocomputing, vol. 228, páginas 154 – 164, 2017. ISSN 0925-2312. Advanced Intelligent Computing: Theory and Applications.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A., van Ginneken, B. y Sánchez, C. I. A survey on deep learning in medical image analysis. Medical Image Analysis, vol. 42, páginas 60 – 88, 2017. ISSN 1361-8415.

Lundberg, S. M. y Lee, S.-I. Shapley additive explanations. https://shap.readthedocs.io/en/latest/, 2017a.

Lundberg, S. M. y Lee, S.-I. A unified approach to interpreting model predictions. En Advances in Neural Information Processing Systems (editado por I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan y R. Garnett), vol. 30, páginas 4765–4774. Curran Associates, Inc., 2017b.

Maloney, K. O., Schmid, M. y Weller, D. E. Applying additive modelling and gradient boosting to assess the effects of watershed and reach characteristics on riverine assemblages. Methods in Ecology and Evolution, vol. 3(1), páginas 116–128, 2012.

Marsland, S. Machine Learning: An Algorithmic Perspective, Second Edition. Chapman and amp; Hall/CRC, 2nd edición, 2014. ISBN 1466583282.

McKinney, W. Pandas. https://pandas.pydata.org/

Muhamad Adnan, M. H. B., Husain, W. y Abdul Rashid, N. A hybrid approach using naïve bayes and genetic algorithm for childhood obesity prediction. En 2012 International Conference on Computer Information Science (ICCIS), vol. 1, páginas 281–285. 2012.

Oliphant, T. Numpy. https://numpy.org/, 1995.

Organization, W. H. Obesity: preventing and managing the global epidemic. 2000.

Paul, S. Ensemble learning — bagging, boosting, stacking and casca- ding classifiers in machine learning using sklearn and mlextend libraries.

https://medium.com/@saugata.paul1010/ensemble-learning-bagging-boosting-stacking- and-cascading-classifiers-in-machine-learning-9c66cb271674, 2018.

Ministerio de Sanidad, C. y. B. S. Encuesta nacional de salud. españa 2017. https://www.mscbs.gob.es/estadEstudios/estadisticas/encuestaNacional/encuestaNac 2017/ ENSE2017_notatecnica.pdf, 2017.

Singh, B. y Tawfik, H. Machine learning approach for the early prediction of the risk of overweight and obesity in young people. En Computational Science – ICCS 2020 (editado por V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos y J. Teixeira), páginas 523–535. Springer International Publishing, Cham, 2020. ISBN 978-3-030-50423-6.

Wickham, J., Stehman, S. y Homer, C. Spatial patterns of the United States national land cover dataset (nlcd) land-cover change thematic accuracy (2001–2011). Internatio- nal Journal of Remote Sensing, vol. 39, páginas 1729–1743, 2018.

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, *20*(2), 215–232.

Breiman, L. Bagging predictors. *Machine Learning* **24**, 123–140 (1996). Springer Ed. https://doi.org/10.1007/BF00058655

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). New York, NY, USA: ACM. https://doi.org/10.1145/2939672.2939785

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 4768–4777.

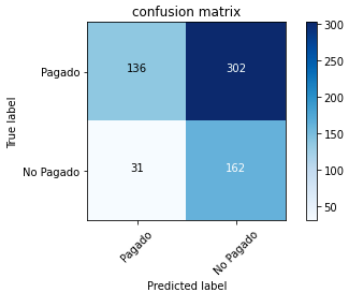
12. ANEXOS

OTROS MODELOS TRAS BALANCEO.

4.10 - DecisionTreeClassifier

In [22]:
```
from sklearn import tree

modelo = tree.DecisionTreeClassifier(random_state=0,max_depth=6)
modelo.fit(train_features, train_labels)
prediccion = modelo.predict(test_features)
report = classification_report(test_labels, prediccion)
print("Correct classification rate:", accuracy_score(test_labels, prediccion))
print(report)

cnf_matrix = confusion_matrix(test_labels, prediccion)
plot_confusion_matrix(cnf_matrix, classes=array_clases,
                      title='confusion matrix')
plt.show()
```

```
Correct classification rate: 0.4722662440570523
              precision    recall  f1-score   support

           0       0.81      0.31      0.45       438
           1       0.35      0.84      0.49       193

    accuracy                           0.47       631
   macro avg       0.58      0.57      0.47       631
weighted avg       0.67      0.47      0.46       631

Confusion matrix, without normalization
[[136 302]
 [ 31 162]]
```



4.13 - ExtraTreesClassifier ¶

In [25]:
```
from sklearn.ensemble import ExtraTreesClassifier
modelo = ExtraTreesClassifier(n_estimators=250,
                              random_state=0)
modelo.fit(train_features, train_labels)
prediccion = modelo.predict(test_features)
report = classification_report(test_labels, prediccion)
print("Correct classification rate:", accuracy_score(test_labels, prediccion))
print(report)

cnf_matrix = confusion_matrix(test_labels, prediccion)
plot_confusion_matrix(cnf_matrix, classes=array_clases,
                      title='confusion matrix')
plt.show()
```

```
Correct classification rate: 0.5277337559429477
              precision    recall  f1-score   support

           0       0.80      0.42      0.55       438
           1       0.37      0.77      0.50       193

    accuracy                           0.53       631
   macro avg       0.59      0.59      0.53       631
weighted avg       0.67      0.53      0.54       631

Confusion matrix, without normalization
[[185 253]
 [ 45 148]]
```