DATA MINING TECHNIQUES IN VIDEO GAMES

Diana Sofía Lora Ariza

MÁSTER EN INGENIERÍA INFORMÁTICA. FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Máster en Ingeniería Informática

Madrid, Septiembre de 2015

Calificación obtenida: 10

Director:

Pedro A. González Calero

Autorización de difusión

Autor
Diana Sofía Lora Ariza

Fecha
Madrid, Septiembre de 2015

La abajo firmante, matriculada en el Máster en Ingeniería Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "Minería de Datos en Trazas de Videojuegos", realizado durante el curso académico 2013-2015 bajo la dirección de Pedro A. González Calero en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen en castellano

Debido a los diferentes modelos de negocio utilizados en torno a los videojuegos, ha aumentado la necesidad de conocer los gustos y mejorar la experiencia de los jugadores con el objetivo de aumentar las ganancias. Una de las razones por la que una persona deja de jugar es debido a la frustración que le causa no poder avanzar; por ello, tener el balance adecuado de dificultad es fundamental. El enfoque llamado *banda de caucho* es uno de los más populares para el ajuste de dificultad en videojuegos, el cual crea un vínculo entre el jugador y sus enemigos. Así, dependiendo de las habilidades demostradas por el jugador, la dificultad del juego es perzonalizada [46].

A través de la minería de datos es posible recuperar información vital producida por los jugadores para la toma de decisiones, pruebas, mejora del diseño e investigación. Este trabajo de fin de Máster se centra en el análisis de videojuegos y la aplicación de técnicas de minería de datos para la clasificación del comportamiento de los usuarios y la personalización del juego.

Palabras clave

Minería de datos, Aprendizaje Automático, Clasificación, Clustering, Predicción del comportamiento, Videojuegos, Telemetría, Métricas, Ajuste de Dificultad, Tetris.

Abstract

Due to the different business models employed in video games, the need of knowing user preferences and improve their experience has increased aiming to boost revenue. One of the reasons a person leaves a game is because the frustration of not being able to move forward; therefore, having the right balance of difficulty is essential. One popular approach of dynamic difficult adjustment (DDA) is called *rubber band*, which builds a virtual connection between the player and his enemies. Thus, depending on the skills displayed by the player, the game difficulty is customized [46].

Through Data Mining, it is possible to obtain vital information from raw data produced by players' interaction with the game. This information is used for decision making, testing, improving game design and user research. This Master's final project focuses on game analytics and the employment of data mining techniques for behavioral player classification and difficult adjustment.

Keywords

Data Mining, Machine Learning, Classification, Clustering, Predict player behavior, Games, Telemetry, Metrics, Difficult Adjustment, Tetris.

Índice general

Agradecimientos

Mis mas sinceros agradecimientos a mi tutor Pedro A. González Calero, tus consejos y dirección han sido fundamentales para el desarrollo de este trabajo. También a Marco Antonio Gómez Martín por tu paciencia y tiempo.

Dedicatoria

A mis padres, quienes a través del ejemplo me han enseñado a luchar por mis metas hasta alcanzarlas.

"*El éxito es la suma de pequeños esfuerzos repetidos día tras día.*"

Robert Collier

Capítulo 1

Introducción

Los avances en la tecnología han permitido persistir gran cantidad de datos. En consecuencia, se crearon las base de datos estructuradas y sistemas gestión de bases de datos que permite almacenar y recuperar datos de forma eficiente y eficaz. Asimismo, a través del poder de la computación, se ha encontrado la posibilidad de navegar en el inmenso océano de datos creado a través de los años, con el fin de encontrar sentido a estos. En el ámbito de los videojuegos no es diferente, éstos han evolucionado hasta convertirse en sistemas refinados de información.

En los últimos años, el análisis de videojuegos se ha introducido en el ciclo de desarrollo y producción a causa de la creciente necesidad de tener mayor conocimiento sobre los jugadores, sus gustos y su interacción con el juego. La aplicación de métricas con datos obtenidos a través de telemetría, se ha convertido en una tendencia utilizada en el desarrollo de juegos e investigación desde hace 5 años[21]. La mayoría de los datos recuperados vía telemetría son utilizados para realizar reportes de marketing, calidad, comparativas (benchmark), mejoras en el diseño y pruebas. Esto ha causado que el análisis de videojuegos se convierta en un área importante de la inteligencia de negocio de la industria[20, 38, 44].

Las dos preguntas más importantes del análisis de datos de videojuegos son: Qué variables rastrear y cómo analizar los datos[20]. En el mundo real, no es posible analizar todas variables de comportamiento de los jugadores debido a la limitada capacidad computacional. Por ello, es necesario desarrollar un enfoque que permita analizar las variables a partir de su

costo/beneficio[28]. También dependiendo de la fase en la que se encuentre el proyecto, las variables de interés cambian; por ejemplo, en las últimas fases de desarrollo, refinamiento del diseño y pruebas, no es necesario seguir las variables relacionadas con las ganancias porque aún no está disponible al público. Otro ejemplo común, ocurre en los MMORPG (Massively Multi-Player Online Role Playing Game) cuando se lanzan actualizaciones, es habitual rastrear variables adicionales del juego para validar que las características agregadas o modificadas funcionan de la forma deseada.

Por medio de la telemetría se recuperan datos generados remotamente. Éstos son transmitidos a un servidor, formateados y persistidos. Las métricas permite la interpretación de los datos adquiridos; y así, es posible analizar el proceso de desarrollo del juego (métricas de proceso), su desempeño técnico y de infraestructura (métricas de desempeño) y el comportamiento que tienen los usuarios dentro del juego (métricas de usuario). En particular, las métricas de usuario describen cómo juega una persona, qué tan sencillo es y qué tan buena es su experiencia; además, éstas son utilizadas para mejorar el diseño y calidad del sistema[57]. Los objetivos principales del análisis de usuarios en los juegos pueden ser concretados en tres categorías: análisis estratégico, análisis táctico y análisis operacional. Por medio del análisis estratégico se tiene una visión global de cómo el juego debe evolucionar basándose en el comportamiento de los jugadores y el modelo de negocio. El análisis táctico se centra en la evaluación del diseño del juego con el fin de agregar nuevas características. Y el análisis operacional se encarga del estudio y evaluación del estado actual del juego; por ejemplo, conocer que cambios deben realizarse en la infraestructura para mejorar la experiencia del usuario[20].

Con la minería de datos es posible reconocer patrones y tendencias escondidos en los datos. De acuerdo con el Gartner Group[31] la minería de datos es: *"The process of discovering meaningful correlations, patterns and trends by sifting through large amounts of data stored in repositories. Data mining employs pattern recognition technologies, as well as statistical and mathematical techniques"*. La minería de datos contiene una cantidad amplia

de métodos e ideas de diferentes campos de la ciencia, uno de éstos es el aprendizaje automático. Ésta se encuentra relacionada con el diseño y desarrollo de algoritmos que permiten a las computadoras adquirir conocimientos a partir de datos, con la finalidad de realizar predicciones[27].

En este trabajo se utilizan diferentes técnicas de aprendizaje automático con el fin de encontrar sentido a los datos recuperados de la interacción jugador-juego. A partir de los datos de comportamiento es posible responder muchos interrogantes los cuales son vitales para la evolución y maduración del juego.

1.1. Motivación

Las técnicas de aprendizaje automático tienes muchas áreas de aplicación debido a que éstas son capaces de extraer información importante de grandes conjuntos de datos. Con la aplicación de estas herramientas es posible encontrar errores de diseño en el videojuego, creación de NPC (*Non-Player Character*) que emulen el comportamiento de jugadores reales, control de calidad, mejorar las ganancias, predecir cuándo un usuario dejará el juego, entre otras[33, 34, 38, 47, 48]. Usualmente el aprendizaje automático se encuentra divido en dos categorías: Aprendizaje No Supervisado y Aprendizaje Supervisado.

Por un lado, Clustering es una de las técnicas de Aprendizaje No Supervisado más utilizadas para la clasificación de los usuarios basándose en la interacción que estos tienen en el juego. Existe gran variedad de algoritmos que permiten obtener información específica a partir de un conjunto de datos. Por ejemplo, k-means[41], debido a su naturaleza, es posible obtener los comportamientos más comunes entre los jugadores de un grupo; mientras que Archetypal Analysis[13] vía Simplex Volume Maximization (SIVM) adquiere los comportamientos extremos. Existe gran variedad de casos de estudio[22, 25, 26, 65] donde se realiza clasificación de los jugadores con estás técnicas. Así es posible reconocer los diferentes tipos de jugadores existentes, sus gustos, cómo juegan, cómo se relacionan entre ellos, etc.

Por otro lado, el objetivo de los algoritmos de Aprendizaje Supervisado es de predecir

el valor de la función para cualquier entrada válida después de aprender con un conjunto de entrenamiento [27, 59]. La salida de los algoritmos puede ser continúo (regresión) o discreto (clasificación). En el contexto de videojuegos, los algoritmos de predicción son utilizados para la creación de NPC que imiten el comportamiento de jugadores reales, el desarrollo de juegos personalizables como es el ajuste de la dificultad, predecir el desgaste de un jugador, entre otros. En concreto, el ajuste de la dificultad emplea un enfoque llamado *banda de caucho*[46]. Este enfoque crea un vínculo entre el jugador y sus enemigos. Así, dependiendo de las habilidades demostradas por el jugador, sus enemigos son graduados; en otras palabras, si un jugador es principiante, entonces sus enemigos manifiestan un comportamiento sencillo, pero si el jugador muestra altas habilidades en el juego, así mismo sus enemigos exhibirán comportamientos más complejos. En un videojuego es fundamental mantener un equilibrio de los retos propuestos a los jugadores; de esta forma, los desafíos no son tan difíciles como para evocar frustración, ni muy fáciles para despertar aburrimiento.

El propósito de este trabajo es el empleo de algoritmos de clasificación y predicción con el fin de predecir el nivel de habilidad de un usuario a partir la interacción que éste tiene con el juego. Una vez se tenga el nivel del jugador, es posible realizar el ajuste de dificultad adecuado.

1.2. Objetivos

El objetivo general de este trabajo es **evaluar la suficiencia de una metodología para la clasificación de comportamiento de jugadores y el ajuste de dificultad a partir de trazas de videojuegos**. Con este fin, se emplean técnicas de Aprendizaje Automático, tales como clustering y clasificación, para la extracción de patrones y predicción de comportamiento de jugadores por medio de las trazas generadas durante una partida de Tetris. El juego utilizado se llama *TetrisAnalytics*, el cual ha sido desarrollado por Marco Antonio Gómez Martín integrante activo del grupo GAIA (Group of Artificial Intelligence Applications).

Para lograr el objetivo general, es necesario la realización de los siguientes objetivos específicos:

1. Diseñar una metodología que integre las diferentes etapas del descubrimiento del conocimiento para la creación de perfiles de jugadores, predicción de éstos y personalización del juego.

2. Implantar exitosamente la metodología en un caso concreto (TetrisAnalytics).

3. Desarrollar herramientas para la preparación de los datos exportados del juego.

4. Aplicar técnicas estadísticas y de aprendizaje automático para la clasificación y predicción de los tipos de jugadores.

5. Diseñar ajuste de dificultad para TetrisAnalytics.

1.3. Estructura del Documento

En este trabajo se presentan varios temas en el análisis de videojuegos tales como: selección de variables, creación de métricas, clasificación de jugadores y personalización del juego acorde con las habilidades del usuario. Su estructura está organizada de la siguiente manera: En el capítulo 2 se presenta el estado del arte del análisis de videojuegos, el análisis de variables, las estrategias de medición, las métricas utilizadas y técnicas de minería de datos para la clasificación del comportamiento y predicción de tipos de jugadores. También se exponen casos de estudios realizados por analistas expertos. En el capítulo 3 se desarrolla una metodología general que permite la clasificación de jugadores y predicción de la dificultad y se realiza la implantación de la metodología en un juego de Tetris. En el capítulo 4 se exhiben los resultados obtenidos de los experimentos realizados a las trazas del juego. En estos experimentos se realiza la evaluación de las variables del juego, se analiza las trazas que describen la interacción del jugador con el sistema, se clasifican los tipos de jugadores

y propone cómo realizar el ajuste de dificultad de acuerdo a las habilidades expuestas por el jugador. Finalmente en el capítulo 5 se presentan las conclusiones y el trabajo futuro.

Introduction

The advances in technology had allowed us to persist great amount of data. As a consequence, relational databases and database management systems were built to persist and retrieve data in an efficient and effective manner. Moreover, with computing it is possible to navigate in the vast ocean of data created through the years to make sense of them. In the context of video games it is not different, because these have become sophisticated information systems.

In recent years, game analytics have been introduced in production and development cycle due to the growing need of having more knowledge about the player, their preferences and their interaction in the game. The use of metrics is the way of harness information from data obtain via telemetry. This is why metrics have become popular in game development and game user research from 5 years till now[21]. Most data acquire through telemetry are used for marketing, quality assurance, benchmark, design enhancement and testing. Therefore, game analytics has become a relevant area of the business intelligence in the industry[20, 38, 44].

The most important questions about game analytics is what variables most be monitored and how analyze them[20]. In reality, it is not possible to analyze all behavioral features because of the limited computational capabilities. Therefore, it is necessary to develop an approach which lets us analyze variables from cost/benefit point of view[28]. On top of that, depending on the stage of the project, the variables change; e.g. in the last phases of development, design refinement and testing, there is no need of monitor revenue variables because the game is not available to the public yet; or in patch updates of MMORPG (Massively Multi-Player Online Role Playing Game) it is useful to track additional variables regarding the new or modified features to verify it correct functionality.

Through telemetry, data can be acquired remotely from the client's platform to the

server, give a specific format and persist it. Metrics enable the interpretation of the data and, therefore, it is possible to analyze the game development process (process metrics), the performance of the technical and software infrastructure (performance metrics), and the user behavior in the game (user metrics). In particular, user metrics describe how the person plays the game, how easy is to play it and how good is the user experience. What's more, these metrics are used to enhance the design and the quality of the game[57]. The main goals of behavioral analysis in games are divided in three categories: strategic analysis, tactical analysis and operational analysis. Through strategic analysis you have an overview of how the game should evolve based on player behavior and the business model. The tactical analysis focuses on the assessment of game design in order to add new features. And finally, the operational analysis evaluates the current game state; for example, know what changes must be made in infrastructure to improve user experience[20].

Data mining is the process extracting useful information from big amounts of data. The Gartner Group[31] defines data mining as: *"The process of discovering meaningful correlations, patterns and trends by sifting through large amounts of data stored in repositories. Data mining employs pattern recognition technologies, as well as statistical and mathematical techniques"*. Data mining has cross-cutting techniques and ideas from other sciences fields like Machine Learning. This is a branch of Artificial Intelligence is related to the design and development of algorithms that allow computers to gain knowledge based on information obtained from the datasets, in order to make predictions[27].

In this project different machine learning techniques are used in order to make sense of data retrieved from player-game interaction. From the behavioral data, it is possible to answer many questions which are vital for the development and maturation of the game.

Motivation

Machine learning techniques have many application areas because they are able to extract important information from large datasets. In the context of video games it is not

different, with these techniques is possible to find game design errors, create NPC (*Non-Player Character*) that behave like real players, quality assurance, improve revenue, predict players churn, etc [33, 34, 38, 47, 48]. Usually, Machine Learning is divided in two categories: Unsupervised Learning and Supervised Learning.

On one hand, Unsupervised Learning has several techniques for searching patterns embedded in raw data. A popular example is Clustering which is frequently used for behavioral player classification for all the data produces by users during gameplay. There are a variety of algorithms that enables us to obtain specific information from a dataset, e.g. k-means[41] which due to its nature, can obtain common behavior between objects in the same cluster; whereas Archetypal Analysis[13] via Simplex Volume Maximization (SIVM) gets extreme behaviors. There are a variety of case studies[22, 25, 26, 65] where behavior player classification is made from these techniques. Using them is possible to classify players by his preferences, how he plays, how he relates to other players, etc.

In the other hand, Supervised Learning goals are predicting the value of the function for any valid input after learning from a training dataset [27, 59]. The output of the algorithms might be continuous (regression) or discrete (classification). In video games, Supervised Learning techniques have several uses, e.g. creation of Non-Player Characters, difficult adjustment, predict players "chrun", etc. In particular, difficult adjustment employs an approach called *rubber band*[46] which builds a virtual connection between the player and his enemies. If the player exhibit a high level of abilities, his enemies will have a more complex and difficult behavior; but if the player shows to be a beginner, then his enemies will have a simple behavior. Difficult adjustment maintain the flow of the game, and this is essential to keep a balance in the challenges proposed to the players. This way, the challenges are not too difficult to evoke frustration, but not too easy to raise boredom.

The purpose of this project is to employ classification and prediction algorithms to predict players level from its interaction with the game. Once the player level is obtained, it is possible to make a suitable difficult adjustment.

Objectives

The overall objective of this project is to **evaluate the adequacy of a player behavior classification and difficulty adjustment methodology via video games traces**. To this end, Machine Learning techniques, such as clustering and classification, are employed for pattern extraction and player behavior prediction through raw data produced by the game. The video game used was developed by Marco Antonio Gómez Martín active member of GAIA (Group of Artificial Intelligence Applications) and its name is *TetrisAnalytics*.

To achieve the overall objective, it is necessary to successfully accomplish the following specific objectives:

1. Design a methodology that integrates the different stages of the knowledge discovery for creating player's profiles, their prediction and game customization.
2. Successfully implement the methodology in a specific case (TetrisAnalytics).
3. Develop tools for data preparation.
4. Apply machine learning and statistics techniques for classification and prediction of players' types.
5. Designing TetrisAnalytics difficulty adjustment.

Document Structure

This project will present several topics of game analytics like: feature selection, metrics analysis, behavioral player classification and difficult adjustment of games according to players' abilities. Its structure is organized as follows: Chapter 2 shows the most popular data mining techniques used in video games, how to analyze features, measurement strategies, metrics used and a popular standard employed to get information from raw data. Also have case studies of Machine Learning techniques for player behavior classification and difficult

adjustment. Chapter 3 presents a general methodology for player classification and difficult adjustment. This methodology is developed from CRISP-DM standard and the analysis of several case studies. Also shows the methodology in action with a Tetris game. This is a simple Tetris game that saves the player moves during gameplay. Chapter 4 shows the results of several experiments made with the data acquire from a diverse set of players, e.g. evaluate which variables are better fit for behavior analysis, verify performance from two cluster algorithms, player behavior classification and ability prediction of a player by its behavior to adjust the difficulty during the session. Chapter 5 contains the conclusions and future work.

Capítulo 2

Minería de Datos en Videojuegos

"You are no longer an individual,
you are a data cluster bound to a vast global network"
Trailer del juego "Whatch Dogs" (Ubisoft). (2012).

La falta de estándares y métricas en la industria de los videojuegos hacen que sea complejo la evaluación de los proyectos. La gran competencia en el mercado ha permitido el desarrollo e inclusión de nuevas herramientas y técnicas de otros sectores de TI con la finalidad de solventar estas necesidades. Entre las técnicas incluidas se destacan las prácticas empresariales, las métricas, el análisis, la gestión de proyectos y las pruebas de usuario [23].

Los videojuegos han pasado de ser aplicaciones simples a sofisticados sistemas de información, con el fin de tomar decisiones importantes a partir del registro de datos de las acciones de los jugadores [27]. El registro de datos es conocido como telemetría en el ámbito de las ciencias de la computación. Las telemetría y métricas son indispensables para la medición de diferentes aspectos del rendimiento de los sistemas, la finalización exitosa de los proyectos y el análisis del comportamiento de los jugadores[45]. El desarrollo basado en métricas, en inglés *Metrics Driven Development (MDD)*, es el proceso que permite cuantificar las tareas y riesgos de un proyecto para gestionar de manera eficiente sus recursos. El análisis de los datos brindan un panorama global del estado actual del proyecto y del progreso que este tiene en el tiempo. Asimismo, la medición del comportamiento de los jugadores es vital para mejorar las características del juego y su validación. Esto se puede alcanzar analizando

los datos para conocer las cualidades más populares, las diferentes formas cómo los usuarios interactúan con el juego, cómo se relacionan entre ellos, entre otras.

Existe una gran cantidad de información escondida en los datos adquiridos vía telemetría. Sin embargo, para obtener esa información es necesario aplicar diferentes técnicas que permitan extraer patrones de los datos[27]. En el momento en que las bases de datos crecieron, muchos de los algoritmos utilizados se convirtieron en obsoletos. Por ello se crearon nuevos métodos diseñados para grandes bases de datos. Este conjunto de métodos es conocido como minería de datos.

La minería de datos es considerada una fase en el proceso de descubrimiento de conocimiento en las bases de datos (*Knowledge Discovery in Databases - KDD*). KDD es un proceso general utilizado para el descubrimiento de información útil a partir de los datos, mientras que "minar"datos es la aplicación de algoritmos particulares para extraer patrones de los datos.

Existen muchas aplicaciones de la minería de datos en el contexto de los videojuegos. Algunas de estas aplicaciones son[27]:

- Encontrar debilidades en el diseño de videojuegos.
- Descubrir cómo atraer nuevos jugadores[28].
- Descubrir qué hacen los usuarios cuando están jugando y predecir su comportamiento.
- Conocer cuáles son los elementos menos utilizados[21].
- Desarrollar oponentes inteligentes o juegos que se adapten a sus jugadores[10, 46, 51, 56, 65].

En los últimos años, el almacenamiento y análisis de datos se han convertido en una herramienta fundamental para el desarrollo, el rendimiento y la escalabilidad de los proyectos de videojuegos. En este capítulo se abordaran temas importantes tales como el análisis, la

telemetría, las métricas y algunas técnicas de minería de datos aplicado en el contexto de los videojuegos.

2.1. Análisis de Videojuegos

El análisis de videojuegos es el proceso de descubrir patrones en los datos con el fin de resolver problemáticas empresariales o la realización de predicciones para soportar decisiones de gestión. Este proceso no solo se basa en la consulta y visualización de datos, también incluye su análisis. Los fundamentos del análisis de videojuegos son las estadísticas, la minería de datos, las matemáticas, la programación y la investigación[14]. El beneficio principal de su implantación en proyectos de desarrollo es la toma de decisiones. Este método permite el análisis de los videojuegos como un producto para proveer una mejor experiencia al usuario y como proyecto permite realizar comparaciones con otros videojuegos en el mercado[23].

El-Nasr, M.S. et al.[23] se centran en dos aspectos importantes del análisis de los videojuegos: la **telemetría** y los **usuarios**. La telemetría es la transmisión digital de datos de manera remota. En este caso particular de videojuegos, es muy común que se realice la transmisión de datos desde el cliente hacia el servidor[54]. Un cliente puede ser programado en diferentes plataformas (consolas de videojuegos, PC's, móviles, tabletas, etc.); esta mantiene una comunicación constante con el servidor, la cual es principalmente en una dirección y los datos son persistidos para su posterior análisis.

En la actualidad, una de las aplicaciones más importantes de análisis de videojuegos es obtener información sobre cómo los usuarios juegan. *Game User Research (GUR)* es la combinación de diferentes técnicas y metodologías (psicología experimental, inteligencia computacional, aprendizaje automático e interacción humano-computadora), que permiten evaluar cómo las personas juegan y la calidad de la interacción que existe entre estos[21, 23]. GUR se centra en los usuarios, quienes también son considerados clientes; por esto, se ha convertido ampliamente aceptada la noción que ayuda a mejorar el juego y las ventas, lo que ha aumentado el crecimiento de esta área considerablemente [19, 50].

En el contexto de GUR, el valor de las métricas de juego yace en monitorear el comportamiento del usuario y su conversión en datos que pueden ser cuantificables y de fácil manipulación. Por medio de la telemetría, es posible evaluar el diseño y la experiencia del usuario a un nivel superior que con otras técnicas no es posible[21, 23, 54, 57]. El verdadero reto se tiene frente a las métricas de juego es saber qué buscar, por qué buscarlo y cómo hacerlo valioso para las diferentes áreas interesadas.

2.1.1. Telemetría

Como se ha mencionado previamente, la telemetría es la transmisión de datos desde el cliente hacia el servidor con el fin de ser persistido y posteriormente analizado. Un cliente envía información a uno o varios servidores manteniendo un flujo continuo de datos. Por lo general, este tipo de comunicación se da en una sola dirección, donde el cliente está enviando continuamente datos al servidor, los datos son procesados (para dar formato o realizar cálculos simples) y finalmente persistidos[23]. Drachen A.[15] define la telemetría como la medición de atributos de un objeto. Un objeto es cualquier cosa que tenga uno o más atributos que pueden ser medidos. Por ejemplo, la recopilación de datos de la ubicación de un jugador en un mundo virtual. El objeto es el jugador y el atributo medido es su ubicación.

La telemetría de usuarios son datos cuantitativos sobre la interacción jugador-juego o jugador-jugador, los cuales son persistidos en bases de datos a partir de trazas generadas por cada usuario. Los datos recopilados son transformados en métricas como el total de tiempo jugado por usuario, los usuarios activos por día o mes, el promedio de ganancias por usuario, etc[23]. Con el fin de trabajar con estos datos, se debe decidir una forma en la que los datos son persistidos. Este proceso es conocido como *operacionalización*[15]. En el caso que se esté recuperando datos de la ubicación de un jugador en el mundo virtual, se puede representar en un número que contenga los movimientos que este ha realizado en metros. Cuando un dato se operacionaliza, se convierte en una variable o característica.

Las variables/características hacen parte de un dominio donde tienen un cantidad finita de valores. A pesar que en el análisis de videojuegos no hay estándares establecidos, en el ámbito de investigación del comportamiento de los jugadores y minería de datos, es más utilizada la palabra *característica*.

La telemetría de videojuegos tiene retos que se presentan en grandes bases de datos orientadas al usuario. De acuerdo con El-Nasr, M.S. et al.[23] algunos de estos retos son:

1. Los datos tienen una alta dimensionalidad debido a que miles de características son medidas para cada usuario.

2. Las bases de datos que contienen la información medida ascienden a la escala de terabytes.

3. Es común reunir datos de diferentes fuentes.

4. Obtención de datos de videojuegos desde los clientes con soporte a múltiples plataformas. En estos casos se necesita un buen diseño en el sistema del lado del servidor para garantizar que todos los datos son enviados. En especial aquellos sistemas que no siempre están conectados a Internet cuando el usuario está jugando, como es el caso de los juegos de móviles.

Los retos para el análisis de grandes bases de datos con métricas de videojuegos son muchos. Los mencionados anteriormente son solo una pequeña parte de todos inconvenientes que se pueden tener al momento de realizar análisis sobre estas métricas.

Estrategias de Medición

El-Nasr, M. S. at al.[23] exponen tres formas fundamentales por medio de las cuales se realiza la transmisión de datos en sistemas de telemetría. Éstas son:

Eventos: La telemetría basada en eventos es la obtención de datos a partir de acciones iniciadas por una persona o un sistema. Por ejemplo, cuando se vende un juego, un ju-

gador compra un accesorio o realiza un movimiento específico pueden ser considerados como eventos.

Frecuencia: Los datos también pueden ser recuperados desde el cliente con una frecuencia estimada. Por ejemplo, obtener la ubicación de un objeto cada segundo.

Manual: En algunas ocasiones, no es necesario la constante medición de ciertos atributos. Por esta razón, es posible iniciar o detener la recopilación de datos de algunas variables. Por ejemplo, cuando se distribuye un parche, es necesario la medición de atributos que generalmente no son necesarios para la verificación del correcto funcionamiento de las características agregadas o modificadas en el juego.

2.1.2. Métricas

Las métricas han estado en la industria de videojuegos desde sus inicios. Sin embargo, solo en los últimos años, la aplicación de métricas en el desarrollo y diseño de videojuegos se ha expandido y madurado[16, 21]. Mientras que la telemetría es la transmisión remota de datos desde el cliente al servidor, las métricas son generadas automáticamente cada vez que una variable es medida[8].Por medio de métricas se puede conocer cómo las personas juegan, qué tan fácil es jugar y qué tan buena es la experiencia del usuario[57].

Las métricas de videojuegos son las medidas interpretables de algo. Por lo general, las métricas son calculadas como una función de alguna característica. La unidad más utilizada es el tiempo; no obstante, también son empleadas otras como el país, el progreso que se tiene en el juego, el artículo más utilizado, entre otras. Con las métricas es posible realizar predicciones para conocer el comportamiento que tendría en el futuro un jugador. Más adelante se explicarán algunas técnicas de minería de datos utilizadas para predicción del comportamiento.

El tiempo de juego es una de las métricas más utilizadas puesto que permite conocer el total de tiempo que el usuario dedica al juego. Cuanto menos tiempo una persona le dedique a un juego, menos interesante es. La optimización de esta métrica es uno de los desafíos del

diseño de videojuegos, sobre todo para los Free-to-Play (F2P). El tiempo de juego y sus derivados hacen parte de los indicadores claves de desempeño (KPIs) en la industria. Entre ellos se encuentran: usuarios activos por día, usuarios activos por mes, promedio de ganancias por usuario y cálculo del desgaste ("*churn*") que tiene el jugador[23, 26, 28].

Figura 2.1: Diagrama jerárquico de las fuentes de datos en el análisis de videojuegos[20].

Las métricas de videojuegos se dividen en tres categorías[20, 23, 45]:

Métricas de usuario: Estas métricas están relacionadas con los usuarios. Los usuarios son considerados como clientes y como jugadores. Cuando son considerados clientes, las métricas son utilizadas para calcular las ganancias que estos proveen. Por ejemplo, promedio de ganancias por usuario, la cantidad de usuarios activos diariamente, análisis de desempeño de la atención al cliente, entre otros. Cuando son considerados como jugadores, las métricas obtenidas son utilizadas para la investigación de cómo juegan, cuáles son los elementos más utilizados, como se relacionan con otros jugadores, el

total de tiempo jugado por usuario, promedio de amigos que tiene en el juego, etc.

Métricas de desempeño: Estas métricas se encuentran relacionadas con el desempeño técnico y de infraestructura. Con éstas se puede determinar cómo se está ejecutando el juego desde el cliente, la estabilidad del servidor, el impacto que tienen las actualizaciones del sistema, etc. Una métrica que es utilizada desde hace mucho tiempo es la cantidad de erratas encontradas en un periodo de tiempo determinado.

Métricas de proceso: Las métricas de proceso están relacionadas con el proceso de desarrollo de videojuegos. Éste es un proceso con cierto grado de creatividad donde es necesario implementar metodologías de desarrollo ágil; por ello, se ha vuelto necesaria su monitorización. Un ejemplo de estas métricas es la medición del tiempo promedio que puede tomar realizar una nueva versión del juego en producción.

Particularmente, las métricas de usuario son utilizadas para investigar el comportamiento que estos tienen en el juego (tiempo promedio de juego, daño infligido por sesión, daño recibido por sesión, entre otros). Las métricas de usuario están divididas en tres categorías que contienen tres intereses diferentes: **métricas de cliente**, **métricas de comunidad** y **métricas de juego**. La primera abarca todos los aspectos del usuario como cliente (costo de adquisición y retención de clientes) y son de interés para el área de marketing, gestión y desarrollo de juegos. La segunda engloba los movimientos de los usuarios en la comunidad; por ejemplo, la actividad en los foros. Y la última, incluye cualquier variable relacionada con el comportamiento del usuario dentro del juego. Las métricas de juego son importantes para cualquier área interesada en conocer cómo juegan las personas incluyendo diseño, investigación de usuario y control de calidad[20].

Métricas de Juego

Las métricas de juego es una forma de Inteligencia de Negocio (BI), las cuales son derivadas de las personas, los clientes y los procesos involucrados en el negocio de los juegos. Al igual que BI, las métricas de juego son utilizadas para soportar decisiones de marketing y

la forma de trabajar con éstas es a través del análisis[21, 61]. Estas son medidas a partir del comportamiento que tiene el usuario dentro del juego (navegación, elementos en posesión y uso, comercio, etc.). En una sola sesión, un jugador puede generar miles de datos como es el caso de los MMORPGs. La medición de los jugadores en el MMORPG *World of Warcraft* involucra guardar la posición del personaje, la salud, el mana, la resistencia, el tiempo en que recibe algún daño, las acciones como correr o utilizar su arma, si se encuentra en combate, viajando o realizando alguna transacción, los atributos de los enemigos que atacan al personaje, el nombre, la raza, el nivel, el equipamiento, el dinero, etc.

Las métricas de juego son fundamentales para la evaluación del diseño y la experiencia del usuario; sin embargo, éstas no son consideradas prioridad, puesto que no se encuentran relacionadas directamente con las ganancias obtenidas por el juego[19, 39, 50]. A pesar de lo anterior, su análisis permite solucionar preguntas importantes como conocer qué áreas del juego son muy utilizadas y cuales no tanto, si los jugadores utilizan los artículos de la forma correcta, o si existen problemas para el progreso del jugador.

El análisis de videojuegos se realiza a través de análisis o síntesis. Análisis es cuando se divide un todo en diferentes partes o componentes; mientras que síntesis es lo contrario, cuando a partir de un conjunto de componentes, se construye un todo. Por ejemplo, dividir datos de comportamiento (como tiempo gastado en un punto de control o cantidad de botones presionados) para obtener mayor información de los componentes individuales es análisis, y una figura mostrando la cantidad de usuarios activos por día o mes es síntesis. Los enfoques utilizados para encontrar respuestas a las preguntas son aquellos en los que se desea confirmar una idea que se tiene (investigación basada en hipótesis), o una idea más abierta porque no se está seguro de cómo encontrar la respuesta a una pregunta dada (investigación basada en exploración)[21].

Investigación exploratoria de métricas es cuando no se sabe exactamente porqué ocurre algo. Por ejemplo, en F2P MMORPG existen muchas razones por las cuales un jugador decide gastar dinero en un artículo virtual. Un método tradicional para la investigación ex-

ploratoria es el análisis de profundidad, o *drill-down analysis*, donde los datos de las métricas de juego son examinados en mayor detalle hasta encontrar la respuesta[27].

Investigación basada en hipótesis de métricas es cuando la búsqueda tiene la finalidad de confirmar alguna idea o cuando la respuesta se puede predecir. Por ejemplo, se puede tener una hipótesis que indica que la cantidad de muertes en un mapa está directamente relacionada con la dificultad de dicho mapa. Verificar los datos de las muertes de los personajes de un conjunto de participantes puede llevar a una confirmación o rechazo de la hipótesis.

De acuerdo con Drachen, A. et al.[21], las preguntas exploratorias consumen mayor tiempo (y por lo general requieren análisis) que aquellas donde se tienen hipótesis (preguntas concretas) y son tratadas usando síntesis de los datos relevantes. Los análisis de métricas de juego pueden ser realizados en conjuntos de datos grandes o pequeños. Por lo general, los objetivos en los análisis con conjuntos de datos pequeños son más detallados que los grandes, puesto que en esta última, la cantidad de datos y jugadores permite tomar conclusiones amplias. En casos donde el conjunto de datos es muy grande, es posible seleccionar una muestra o subconjunto representativo, con el fin de realizar análisis más detallados[24, 64].

2.2. Descubrimiento del Conocimiento

Cross-Industry Standard Process for Data Mining (CRISP-DM) es un estándar desarrollado en 1996, el cual representa un enfoque para los procesos de minería de datos[27]. CRISP es un proceso iterativo, el cual se desenvuelve bien en el ambiente ágil de la industria de desarrollo de videojuegos [45]. En la práctica, algunas fases serán abordadas de forma rápida, especialmente si el análisis ha sido desarrollado en iteraciones previas. Este modelo de procesamiento jerárquico consiste en un conjunto de tareas inmersas en cuatro niveles de abstracción. Éstos van desde un marco general a uno específico[9]. Los cuatro niveles de abstracción son:

Fases: En el primer nivel, el proceso de minería de datos es organizado en un conjunto de fases, cada fase consiste en varias tareas genéricas de segundo nivel. Estas fases serán explicadas más adelante.

Tareas genéricas: Estas tareas son llamadas genéricas porque se desea que sean lo suficientemente generales para acoger todas las posibles situaciones en minería de datos. Estas tareas genéricas deben ser lo más completas y estables posibles. Es decir, deben amparar todo el proceso de minería de datos y todas sus aplicaciones; y el modelo debe ser válido incluso para nuevas técnicas de modelado.

Tareas especializadas: En este nivel, se debe describir como las acciones del nivel superior (tareas genéricas) deben ser desarrolladas en situaciones específicas. Por ejemplo, si en nivel de tareas genéricas existe una tarea llamada limpieza de datos, en el tercer nivel, se debe describir cómo debe realizarse dicho proceso en diferentes situaciones, ya sea limpieza de valores numéricos, caracteres, etcétera.

Procesado de instancias: Este nivel es un registro de las acciones, decisiones y resultados obtenidos del proceso de minería de datos. El procesado de instancias es organizado de acuerdo con las tareas especializadas, sin embargo, cada uno representa un procesado específico.

La descripción de fases y tareas son realizadas en un orden específico en la teoría. Pero en la práctica, muchas tareas pueden ser realizadas en diferente orden e incluso puede requerir realizar nuevamente tareas previas.

2.2.1. Fases CRISP-DM

Como se puede observar en la figura 2.3, el ciclo de vida de un proyecto de minería de datos consiste en 6 fases. La secuencia de estas fases no son rígidas, ya que es posible moverse de una fase a otra si es necesario[9]. Las flechas representan las dependencias más

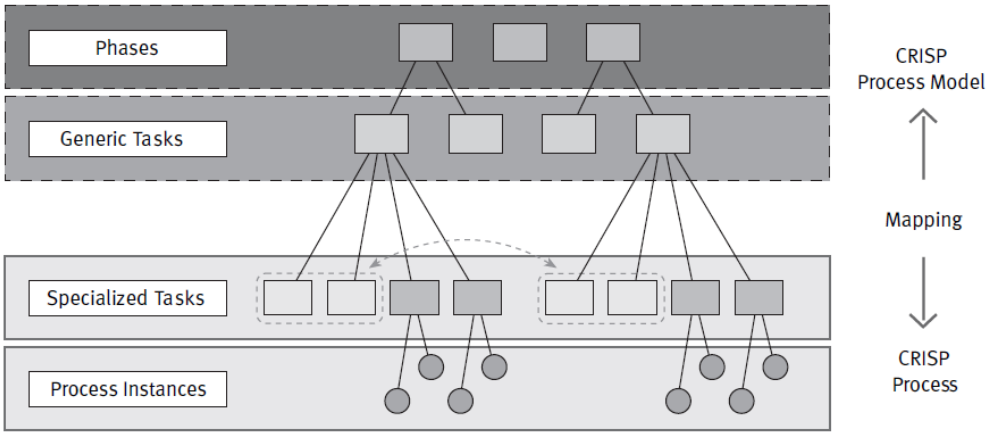


Figura 2.2: Niveles jerárquicos de la metodología CRISP-MD. Obtenida de [9]

frecuentes entre estas fases. El circulo exterior representa el ciclo natural de la minería de datos.

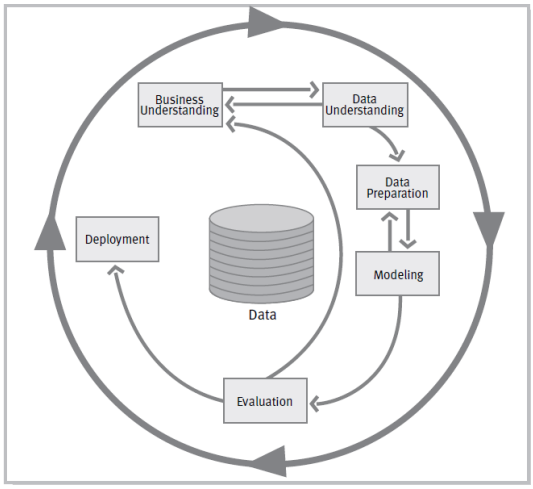


Figura 2.3: Fases del modelo CRISP-DM. Obtenida de [9]

De acuerdo con [9, 27], las fases de CRISP y sus actividades son:

1. **Entendimiento del negocio/investigación:** Esta fase se centra en la compresión de los objetivos del proyecto y sus requerimientos desde la perspectiva del negocio o la investigación. Entre las actividades principales a realizar en esta etapa están:

- Definir los objetivos y requerimientos del proyecto.
- Definir los problemas de minería de datos que representen los objetivos y restricciones del proyecto.
- Realizar un plan que permita alcanzar los objetivos.

2. **Entendimiento de los datos:** Esta fase inicia con la colección de datos en bruto. Se realizan actividades que permitan familiarizarse con los datos, identificar problemas de calidad entre los datos y descubrir la información escondida en los datos. Entre las tareas a realizar están:

   - Extraer los datos necesarios.
   - Explorar los datos para familiarizarse con los datos con el fin de obtener algunas ideas iniciales (a través de *EDA*[1]).
   - Evaluar la calidad de los datos.
   - Seleccionar subconjunto de datos que contengan patrones.

3. **Preparación de los datos:** Esta fase es una de las más extensas y se centra en todas aquellas tareas necesarias para construir el conjunto de datos que finalmente será interesado en la siguiente fase. Las actividades a realizar son:

   - Preparar los datos en bruto para que puedan ser utilizados para las siguientes fases.
   - Seleccionar los casos y variables a analizar.
   - Realizar las transformaciones necesarias a las variables, si es necesario. Es decir, darle el formato deseado para realizar el proceso de minería de datos.
   - Limpiar los datos, como quitar datos irrelevantes o errados.

[1]EDA (Exploratory Data Analysis) es un enfoque utilizado para analizar conjuntos de datos con el fin de obtener sus principales características.

4. **Modelado:** Este proceso de modelado puede realizarse varias veces con el fin de integrar nuevas variables o refinar el resultado obtenido.

    - Seleccionar y aplicar técnicas de minería de datos apropiadas.
    - Calibrar la configuración del modelo para optimizar resultados si el resultado de la técnica aplicada es un modelo.

5. **Evaluación:** Esta fase se centra en la evaluación y validación de todos los pasos realizados para obtener el modelo, el cual debe alcanzar los objetivos propuestos en la primera fase. El resultado de esta fase son las decisiones tomadas a partir del modelo adquirido en el proceso de minería de datos.

6. **Despliegue:** En el despliegue del conocimiento descubierto es importante presentar la información a los interesados de una forma sencilla, por medio de gráficos y reportes, y estos son los encargados de darla a conocer dentro de la organización.

2.3. Clasificación de Jugadores

Los datos adquiridos a través de telemetría sobre el comportamiento de los usuarios permite conocer las características más atractivas del juego, los artículos más utilizados e incluso, la forma como los usuarios interactúan entre sí. La clasificación de jugadores basándose en su comportamiento hace parte de una línea de investigación que se centra en el desarrollo de juegos personalizables[27, 65]. Esta permite realizar un análisis completo de las bases de datos de comportamiento y obtener resultados a través de la generación de perfiles. Con estos resultados se realizan mejoras en el diseño del juego y en las estrategias de monetización utilizadas[22, 43]. El análisis del comportamiento en videojuegos ha pasado a ser una práctica popular debido a que esta provee información vital sobre la población de jugadores.

A través de técnicas de Aprendizaje No Supervisado es posible extraer patrones de los datos de comportamiento de los jugadores donde no existe una definición a priori [27,

60]. Los métodos utilizados se centran en la estructura y la relaciones entre los datos; es decir, buscan patrones entre las características. En el caso en que se desee clasificar el comportamiento de los jugadores, se usa Aprendizaje No Supervisado si se desconoce cómo varía el comportamiento de estos o si no existen clases definidas para su clasificación. Con técnicas como *Clustering*, es posible disminuir la dimensionalidad de un conjunto de datos y obtener aquellas características más relevantes entre los jugadores.

Clustering es uno de los métodos más utilizados en diferentes campos de aplicación. Este método ha sido utilizado en el análisis de los videojuegos como una forma de encontrar patrones en el comportamiento de los jugadores/clientes, realizar comparativas entre videojuegos (bechmark), evaluación del desempeño de la infraestructura y para el diseño y entrenamiento de personajes no controlados por el jugador (*Non-player Character - NPC*).

2.3.1. Clustering

Clustering es el proceso de agrupar un conjunto de objetos de tal forma que los elementos pertenecientes al mismo grupo (cluster), son semejantes entre sí, y diferentes a aquellos que integran a otros grupos. Además, permite disminuir la dimensionalidad del conjunto de datos; es decir, a partir de una base de datos con una gran cantidad de variables, es capaz de seleccionar aquellas que son más importantes[17, 58]. Por ejemplo, a través de esta técnica se puede encontrar grupos de jugadores que exhiben un comportamiento similar, e identificar sus características principales.

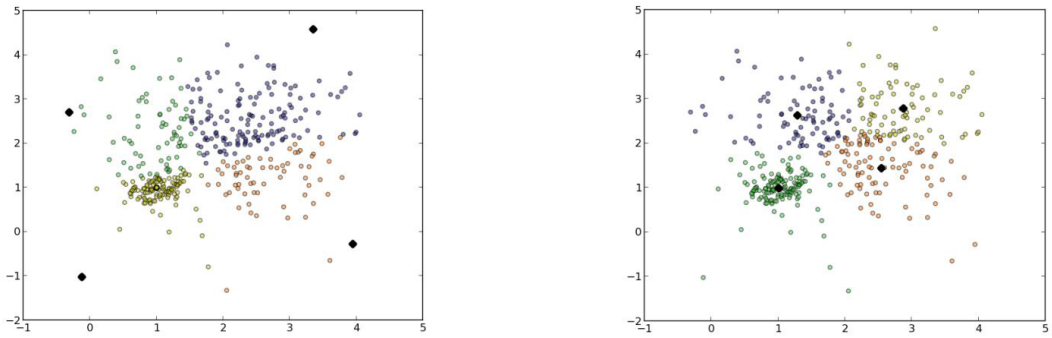
Los objetos son expuestos como puntos (vectores) en un espacio multi-dimensional, donde cada dimensión es una característica. Los puntos son agrupados en una matriz $A_{m,n}$, donde existen m filas (una por objeto) y n columnas (una por característica). La asignación de los objetos a un cluster puede ser parcial (*"soft"*) o total (*"hard"*); es total, cuando un objeto es identificado completamente por dicho cluster; y es parcial, cuando un objeto puede pertenecer a diferentes grupos por tener cierto grado de semejanza con cada uno.

Entre las categorías principales de los métodos de clustering se encuentran los *métodos*

de particionamiento, también conocido como clustering de centroides[18, 32]. Estos métodos construyen grupos que cumplen dos requerimientos: cada grupo debe contener por lo menos un objeto y cada objeto solo puede pertenecer a un grupo. Los métodos de particionamiento empiezan realizando una división inicial entre los datos. Posterior a esta, utiliza una técnica iterativa de re-ubicación de objetos, donde son intercambiados de un grupo a otro con el fin de validar la semejanza de los elementos pertenecientes al mismo grupo. Entre los métodos de particionamiento, existen dos criterios populares para la selección de los objetos que harán parte de un cluster. El primero es el utilizado en el algoritmo *k-means*[41], donde cada cluster es representado por el valor medio de sus objetos[32]. El segundo es utilizado en el algoritmo *k-medoids*[37], donde cada cluster es representado por el valor del objeto más cercano a su centro. Thurau C. et al.[55] expone un método de clustering que al contrario de los mencionados previamente, no buscan por las características comunes entre sus elementos; por el contrario, busca por objetos atípicos que no residen en las regiones densas de los grupos. En otras palabras, este método se centra en los puntos que residen en los extremos de los clusters. *Simplex Volume Maximization* (SIVM) es un algoritmo de *Archetypal Analysis* (AA) adaptado para grandes bases de datos, el cual realiza clasificación de comportamientos extremos en el contexto de métricas de videojuegos[6, 7, 26].

En las técnicas de clustering se tiene una matriz $V^{d\times n}$ de d-dimensiones y n muestras. Su propósito principal es encontrar un conjunto de k vectores expresados $W^{d\times k}$ para $k << n$. Los valores de pertenencia de cada punto a su respectivo centroide se define como una matriz de coeficientes $H^{k\times n}$ [26, 55]. Por ejemplo, para el algoritmo k-means, cada muestra de datos pertenece a un cluster específico. Las columnas de H son todas ceros, exceptuando la fila i-ésima del cluster centroide al cual pertenece el dato; en ese caso el valor es 1, asumiendo que el i-ésimo cluster centroide es el más cercano[27]. Clustering puede ser interpretado como una factorización de matrices cuya finalidad es minimizar la norma de Frobenius[30] $\| V - WH \|$.

En la figura 2.4 se muestra un ejemplo de los algoritmos k-means y SIVM de una base de



(a) Clustering con SIVM.

(b) Clustering con k-means.

Figura 2.4: Gráficas de clustering de dos dimensiones con base de datos sencilla.

datos sencilla, donde $k = 4$, y el resultado es visualizado en una gráfica de dos dimensiones. En estas gráficas, los diamantes negros representan los centroides de cada cluster. Cómo se puede observar, los centroides de los datos analizados con SIVM se encuentran en los puntos extremos de cada cluster; mientras que los centroides de los datos analizados con k-means en la media de los puntos.

2.3.2. Caso de Estudio: Tera

Un ejemplo interesante de clasificación de los jugadores se realiza con MMORPG *Tera*[25]. El caso de estudio se centra en la aplicación de clustering en telemetría del comportamiento de los jugadores con alta dimensionalidad. Su objetivo es reducir la dimensionalidad de los datos de comportamiento, encontrar las características más importantes de estos y localizar patrones; los cuales son expresados en terminos del comportamiento del usuario como función de las características seleccionadas para probar y refinar el diseño del juego.

La historia presentada en Tera (The Exiled Real of Arborea) se da en un mundo de fantasía el cual está basado en niveles y clases. En un juego basado en niveles al comenzar el personaje tiene una puntuación baja en todas sus características. Cuando el jugador aumenta su nivel, la puntuación aumenta, pero este grado de incremento de cada característica depende de las preferencias del jugador. Por ejemplo, existen características opcionales, tales

como Minería y Plantación, donde el jugador puede decidir invertir tiempo en esta habilitad para aumentar su puntaje.

Los datos utilizados en el caso de estudio fueron obtenidos en la fase de prueba del juego, antes de su lanzamiento. Este juego es un MMORPG tradicional con características tales como: quests, mejorar habilitades, batallas y una economía integrada. Los jugadores pueden crear uno o más personajes de diferentes razas y roles. La base de datos de Tera analizada en este caso de estudio contiene información de 250.000 personajes con dos grupos de características:

- **Características de la habilidad del personaje:** Estas características están relacionadas con las habilidades de los personajes; por ejemplo, clase, raza, fuerza, ataque, defensa, entre otras. En total son 8 características.
- **Características del juego:** Estas características describen como juega el personaje; por ejemplo, los monstruos matados, amigos en el juego, misiones completadas, etc.

El alto valor de desviación estándar en las características se debe al diseño de nivelación de este juego, esto es debido a que en el conjunto de datos de los personajes se encuentran 32 niveles diferentes. De un total de 33 características preseleccionadas, se evalúan 18 de ellas. A continuación los resultados del análisis del conjunto de datos:

- **Misiones completadas:** Cantidad de misiones terminadas ($\mu = 40{,}9$, $\sigma = 59{,}3$)
- **Amigos:** Cantidad de amigos en el juego ($\mu = 0{,}44$, $\sigma = 1{,}4$)
- **Logros:** La cantidad de logros alcanzados ($\mu = 10{,}7$, $\sigma = 13{,}4$)
- **Minería y Plantación:** Nivel de habilidad de minería ($\mu = 5{,}3$, $\sigma = 16$) y Plantación ($\mu = 2{,}6$, $\sigma = 8{,}9$).
- **Matar monstruos:** La cantidad de enemigos controlados por agentes inteligentes que ha matado el personaje ($\mu = 625{,}2$, $\sigma = 1242{,}9$).

- **Total de artículos obtenidos:** El total de artículos que el personaje ha encontrado durante el juego ($\mu = 100{,}9$, $\sigma = 200{,}5$).
- **Muerte por monstruos:** El número de veces que un enemigo controlado por un agente inteligente ha matado al personaje ($\mu = 2{,}7$, $\sigma = 12{,}1$).
- **Subasta:** La cantidad de veces que un personaje ha participado en una subasta, ya sea creación o compra ($\mu = 818{,}3$, $\sigma = 1667{,}9$).
- **Nivel del personaje:** El rango de niveles está entre 1 y 32 ($\mu = 7{,}96$, $\sigma = 7{,}6$). Es importante destacar que un jugador puede tener varios personajes, así que la cantidad de jugadores reales probablemente sea menor a la cantidad de personajes del conjunto de datos.

Para el análisis del conjunto de datos se realizó una división de los datos a partir de los niveles de los jugadores, esto con el fin de determinar las tendencias que se dan en los jugadores en función de su nivel. Se seleccionaron cuatro grupos donde los niveles varían de la siguiente forma: 1-10 (166.003 personajes), 11-20 (48.270 personajes), 21-31 (21.703 personajes) y por último, se creó un solo conjunto para los personajes que tienen el máximo nivel, 32 (4.241 personajes). En MMORPGs los personajes tienen un nivel máximo a alcanzar, después de alcanzarlo los jugadores se dedican principalmente a mejorar su equipamiento y artículos del personaje. Así que en este último grupo, los personajes se diferencian por la puntuación de sus habilidades. Realizar el análisis de los datos de comportamiento de los jugadores en función del nivel permite conocer el comportamiento y las tendencias que se dan en el progreso del jugador durante el juego.

Después de dividir los datos en conjuntos agrupados por el nivel de cada personaje, se aplican los algoritmos de Aprendizaje Automático k-means y SIVM (Simplex Volume Maximization). Para ambos se realizan ejecuciones con k variando de 2 a 24, y se determinó a través del error cuadrado y gráficos la cantidad de clusters óptimo. Se concluye que la mejor opción es realizar 6-7 clusters, independientemente del algoritmo aplicado. Los clusters se

mantienen constantes a través de las diferentes agrupaciones realizadas dependiendo de los niveles. Sobresale un clusters de personajes que tienen las puntuaciones más altas en la mayoría de las habilidades, este grupo es considerado como "La Elite"; también se encuentra otro grupo con las puntuaciones más bajas y es considerado como los personajes de bajo desempeño. Los cuatro clusters restantes se dividen en dos grupos: dos con puntuación media, uno mejor que el otro, pero con las habilidades de Minería y Plantación bajo; y dos con puntuaciones semejantes, pero con alta calificación en las habilidades de Minería y Plantación.

Drachen, A. et al.[22] realizan un estudio inicial de cómo identificar diferentes tipos de jugadores en videojuegos comerciales populares a través de aprendizaje no supervisado. Por medio de esa estrategia, se realiza la identificación de los diferentes tipos de jugadores que presenta el análisis de datos y relacionando cada una de las características con el diseño del juego.

En el Cuadro 2.1 se encuentran los resultados de clasificación de comportamiento de los jugadores obtenidos con k-means. Entre los grupos resalta uno pequeño llamado **Elite**, quienes sobresalen por tener las puntuaciones más altas en todas las características, excepto por muertes ocasionadas por monstruos y una baja puntuación en la habilidad de Minería y Plantación. Esto quiere decir que el jugador se enfoca en el desempeño dejando a un lado aquellas habilidades que no impactan en este; en este caso, las habilidades de Minería y Plantación que proveen acceso a recursos y equipamiento del personaje, aunque este último puede ser adquirido a través de subastas. El grupo de los **Atrasados**, quienes tienen baja puntuación en todas sus características, incluyendo muerte por monstruos representan el 39.4 % de los jugadores. Además, entre los grupos de bajo rendimiento se encuentran dos clusters con mejores puntajes: **Joes Promedios** y **Los Dependientes**; este último con las mejores puntuaciones exceptuando la Elite. Los dos últimos grupos, **Trabajador 1** y **Trabajador 2** tienen puntuación semejante a los dos mencionados anteriormente (Joes Promedios y Los Dependientes); sin embargo, presentan mejor puntuación en Minería y

Título	%P	Características
Elite	5.78	Alta puntuación en todas las características excepto Minería y Plantación, las cuales son las habilidades más bajas en el juego.
Atrasados	39.4	Baja puntuación en todas las características.
Joes Promedio	12.7	Mejores puntuaciones que los de bajo rendimiento en todas las categorías. Cuartos en el ranking general.
Dependientes	18.6	Puntuaciones promedio en todas las categorías, gran cantidad de amigos, tercero en el ranking general y segundos matando monstruos.
Trabajador 1	15.9	Semejante a Joes promedio, pero con alta puntuación en Minería y Plantación, y tercero en el ranking de total de artículos obtenidos.
Trabajador 2	7.6	Semejante que los Dependientes, pero con mejor puntuación en las habilidades de Minería y Plantación. Segundo en el ranking general. Segundo en el ranking de total de artículos obtenidos.

Cuadro 2.1: Comportamiento interpretado de los clusters,
Grupo Nivel 32, K-means, %P= % de jugadores en el grupo. Tomada de [25, p.169].

Plantación, y han adquirido más artículos. Los resultados adquiridos a través de SIVM (Cuadro 2.2) también identifican los perfiles de comportamiento Elite y Atrasados, ya que estos dos representan comportamientos extremos identificados en k-means. No obstante, los otros cuatro grupos exhiben comportamientos más extremos. Las habilidades de Minería y Plantación tienen puntuaciones promedio, pero un poco más altas de lo normal, lo cual demuestra que los jugadores se especializan en estas competencias. El perfil de **Maestro de Subasta** representa los jugadores que se centran en usar las subastas del juego para obtener logros, como la obtención de artículos, y estos tienen grandes redes sociales y altas puntuaciones en Minería con el fin de obtener recursos para las subastas. El grupo de **Amigables Pros** tienen un comportamiento similar a los Maestros de Subasta, pero no utilizan tanto las características de subasta del juego, y tienen puntuaciones altas en las características de desempeño. Estos dos perfiles muestran jugadores enfocados en la obtención de ganancias a través de los medios provistos en el juego (adquisición de artículos, habilidades y subastas).

Título	%P	Características
Elite	3.9	Altas puntuaciones en general, excepto en Minería y Plantación, y muerte por monstruos. No crean subastas.
Atrasados	7.6	Bajas puntuaciones en general, altas muertes por monstruos.
Plantadores	21.6	Puntuaciones promedio, pero con alta habilidad en Plantación.
Mineros	15.0	Puntuaciones promedio, pero con alta habilidad en Minería.
Maestros de Subasta	1.1	Altas puntuaciones en subastas y logros. Segundos en adquisición de artículos y matando enemigos. Segundos haciendo amigos y alta puntuación en minería.
Amigables Pros	50.8	Alta puntuación haciendo amigos, puntuaciones similares a los Maestros de Subasta. Penúltimos en adquisición de artículos.

Cuadro 2.2: Comportamiento interpretado de los clusters,
Grupo Nivel 32, SIVM, %P= % de jugadores en el grupo. Tomada de [25, p.169].

Sólo dos de los seis clusters aprenden habilidades diferentes a las de combate, lo cual podría representar un problema en el diseño de Tera. La adquisición de recursos es una habilidad importante para la economía de un MMORPG, y solo el 25-35 % de los jugadores desarrollan estas habilidades. Los jugadores clasificados como de bajo rendimiento son aquellos en mayor riesgo de dejar el juego. Por el contrario, el perfil de Elite y aquellos grupos con altas redes sociales son de mayor interés porque estos garantizan una comunidad sostenible.

En la estrategia utilizada en este caso de estudio de clasificación de comportamiento de los jugadores, se puede observar que se encuentra intrínsecamente relacionado con el diseño del juego. Más aún para la selección de las características a ser evaluadas. A través de los algoritmos utilizados, k-means y SIVM, se pudo observar las ventajas y dificultades de cada uno por medio del análisis de los resultados obtenidos. En particular, k-means provee resultados generales para cada cluster identificado; mientras que los resultados de SIVM presenta comportamientos extremos en cada grupo.

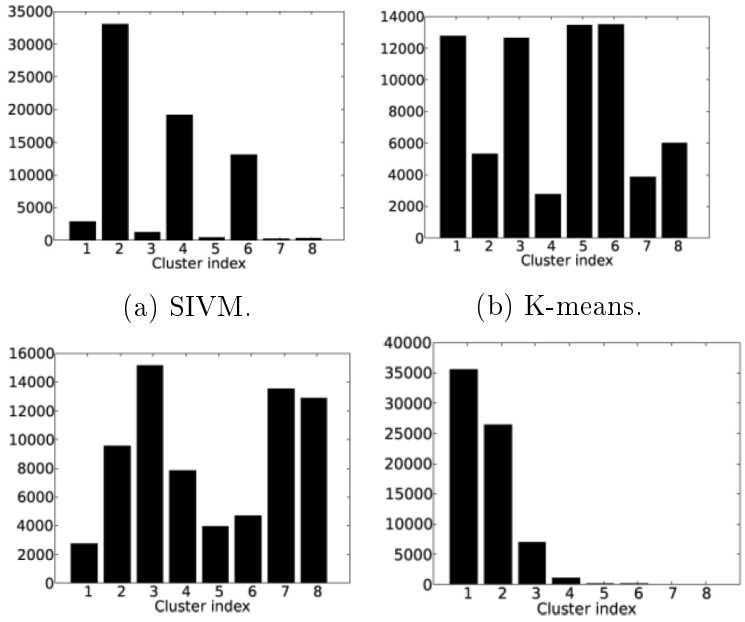
2.3.3. Caso de Estudio: World of Warcraft

Drachen, A. et al.[26], se realiza una comparación de los diferentes algoritmos utilizados para categorizar jugadores. El análisis es realizado con datos del popular MMORPG *World of Warcraft*. En este se realizan evaluaciones del tiempo de juego y la velocidad en que los jugadores aumentan de nivel.

En los resultados obtenidos, resaltan los algoritmos k-means y Simplex Volume Maximization (SIVM) debido a que producen resultados interpretables de los perfiles de comportamiento. Los centroides obtenidos a través del algoritmo k-means son bastante similares y muestran un tiempo invertido semejante en el aumento de nivel de los usuarios. Mientras que los resultados obtenidos con SIVM son más variados. Por otro lado, los algoritmos Non-Negative Matrix Factorization (NMF) y Análisis de Componentes Principales (PCA) generan perfiles que no son interpretables en términos de la mecánica del juego; por ejemplo, los jugadores disminuyen su nivel en el tiempo lo cual no es posible en este World of Warcraft.

El objetivo de este análisis es investigar como las diferentes técnicas de clustering se desempeñan en cuanto a la separación de los clusters y qué tan interpretable son sus resultados.

Los experimentos se realizaron configurando ocho clases diferentes ($k = 8$) entre los algoritmos utilizados. La figura 2.5 está basada en la de la varianza explicada vs los vectores base con el fin de producir clases de jugadores que sean significativamente diferentes en comportamiento. Para SIVM, existen tres grupos grandes y en k-means cuatro grupos grandes y cuatro pequeños. En las figuras 2.6a, 2.6b, 2.6c, 2.6d se muestra el histórico de nivel/tiempo de un jugador. En 2.6a se percibe que el jugador aumenta su nivel de experiencia muy lentamente; mientras que la segunda muestra un jugador que aumenta su nivel rápidamente a nivel 70 y después de un tiempo aumenta a nivel 80. Estos dos tipos de jugadores pueden ser etiquetados como *jugador casual* y *jugador experto*. En la figura 2.6b se muestran los centroides obtenidos por medio del algoritmo k-means, estos tienen cierto



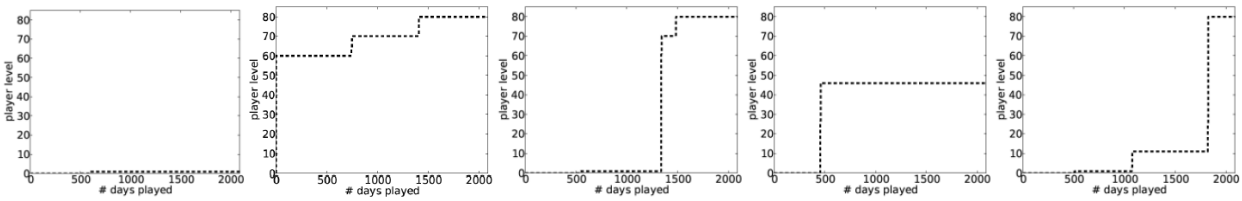
(a) SIVM. (b) K-means.

(c) NMF. (d) PCA.

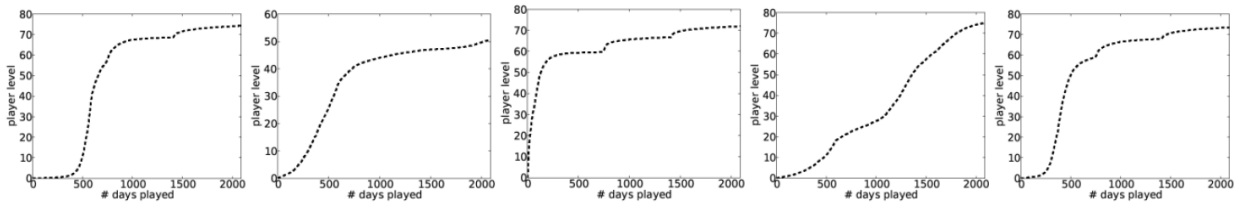
Figura 2.5: Centroides generados a partir de base de datos de World of Warcraft.

grado de similaridad y por ello, no se tiene una interpretación directa de los resultados. Esto quiere decir que los perfiles de comportamiento resultantes, expresados en tiempo de juego y velocidad de aumento de nivel, son en cierto grado similares. Mientras que los centroides en la figura 2.6a, fueron resultado del algoritmo SIVM y estos muestran grandes diferencias. Igualmente, los jugadores suben al nivel 80 en un rango de tiempo similar, esto es debido al lanzamiento de una expansión que modifica el nivel máximo de 70 a 80. Las figuras 2.6c y 2.6d muestran alguna similitud debido a que ambas tienen varias pendientes. Esto significa que los perfiles de comportamiento resultantes, expresado en términos de tiempo de juego y velocidad de nivelación, son de algún modo parecidos.

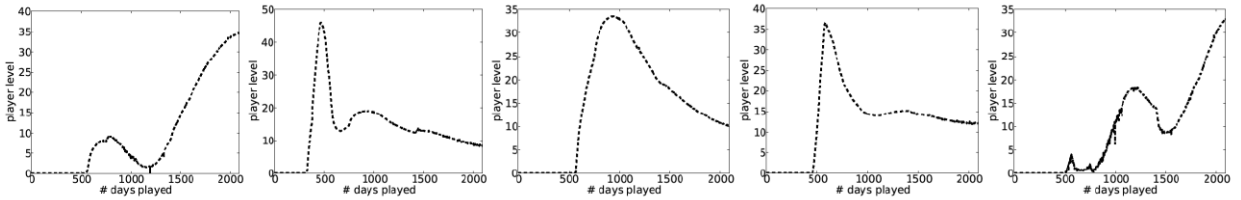
Además de una descripción de los clusters, también se desea una discriminación de los tipos de jugadores; es decir, la cantidad de jugadores que pertenecen a cada clase. Debido a que k-means clustering realiza una asignación total de los objetos, donde cada objeto pertenece a un único cluster, no se puede conocer el grado de pertenecia que tiene un objeto en los diferentes clusters creados. Por el contrario, SIVM realiza una asignación parcial de



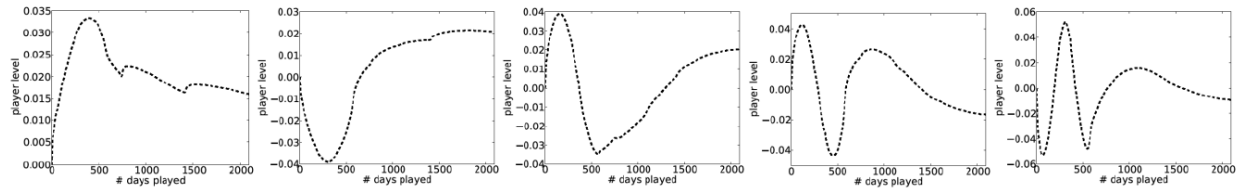
(a) Los primeros 5 vectores base de SIVM. Todos los vectores base corresponden a valores legales de comportamiento (comportamiento permitido en el juego).



(b) Los primeros 5 cluster centroides de k-means clustering. Estos representan una gran cantidad de jugadores, son similares entre ellos y no permite una interpretación inmediata.



(c) Los primeros 5 clusters centroides de NMF. Los resultados pueden ser interpretados, sin embargo, no corresponden a comportamientos posibles según el diseño del juego; por ejemplo, los jugadores no pueden perder nivel.



(d) Los primeros 5 clusters centroides de PCA no corresponden a comportamientos posibles en el juego (perder nivel a través del tiempo).

los objetos, por ello los jugadores son expresados en la relación que tienen con cada uno de los 8 perfiles de comportamiento definidos.

Los resultados de estos algoritmos muestran diferentes enfoques hacia la clasificación del comportamiento de los jugadores en los videojuegos, donde se pueden observar tanto fortalezas como puntos por mejorar. El agrupamiento de jugadores por medio de k-means ubica a cada jugador en un único grupo. La asignación del grupo se lleva a cabo por medio

del centroide más cercano, el cual se encuentra definido por un comportamiento específico. Por otro lado, NMF, PCA y SIVM proveen un vector base (centroide) el cual determina un espacio en el que los jugadores pueden hacer parte; es decir, los jugadores pueden ser descritos en términos de la relación que tienen con cada vector base (tipo de comportamiento) y son agrupados de acuerdo con su ubicación respecto a dicho vector base. Tanto SIVM y k-means brindan centroides que pueden ser interpretados desde la perspectiva del comportamiento que representan. Aunque SIVM presenta vectores base significativamente diferentes en comparación con k-means. Por otro lado, NMF y PCA presentan resultados que no son posibles ya que contradicen el diseño del juego; por ejemplo, los jugadores no pueden perder nivel a lo largo del tiempo. A pesar que estos dos métodos brindan centroides válidos desde una perspectiva metodológica a partir de los datos procesados, no es posible interpretar el resultado en términos del comportamiento de los jugadores.

2.4. Predicción de la dificultad

La finalidad de un juego es que sus usuarios se diviertan. De acuerdo con el modelo psicológico de los jugadores de Daniel Cook [11], éstos son impulsados por el deseo de aprender nuevas habilidades. Los jugadores sienten el deseo de aprender nuevas habilidades ya que reciben recompensas. La diversión adquirida en un juego se deriva del dominio de las habilidades puesto que esto brinda satisfacción al jugador. Después de dominar las habilidades, el juego da una recompensa al usuario por su buen trabajo y crea nuevos objetivos por alcanzar. Así se crea un ciclo donde el jugador se encuentra constantemente aprendiendo, dominando habilidades y obteniendo recompensas, con lo que lo mantiene interesado. Los juegos deben mantener el ciclo de aprendizaje-dominio-recompensa en el nivel adecuado con el fin que no sea lo suficientemente fácil para ser aburrido o demasiado difícil para ser frustrante. Ambos casos llevan al mismo resultado, donde el usuario abandona el juego. Por medio del ajuste dinámico de la dificultad, en inglés *Dynamic Difficulty Adjustment* (DDA), es posible modular cómo el juego responde a las habilidades particulares de sus jugadores

durante la sesión de juego.

La dificultad es considerada un factor subjetivo que se deriva de la interacción entre el jugador y el reto propuesto. Ésta no es estática, puesto que entre mayor dominio tiene el jugador sobre una habilidad concreta, más compleja se convierten las tareas a realizar[35, 46]. En el caso que el juego presente consecuencias no deseadas, estas pueden frustrar las metas y aspiraciones del jugador. Por esta razón, tener un nivel balanceado de dificultad y consistencia es muy importante en los videojuegos[1].

Un enfoque popular utilizado para el ajuste dinámico de la dificultad es la "banda de caucho", en ingles *Rubber band AI*[46]. Este enfoque crea una unión virtual entre el jugador y su oponente, similar a una banda de caucho; si el jugador tira en una dirección (si juega mejor o peor que su oponente), la banda de caucho hace que su oponente se mueva en la misma dirección (que este juegue mejor o peor respectivamente).

Un sistema de ajuste dinámico de la dificultad exitoso debe mantener un balance interno del juego y los mecanismos de retroalimentación. Con este fin, el sistema debe ser adaptado al diseño de los objetivos fundamentales del juego con respecto a la experiencia del jugador; es decir, es necesario conocer el diseño del juego para poder ajustar la experiencia de un jugador particular[35]. El marco de trabajo que permite alcanzar este objetivo es MDA (Mecánicas, Dinámica y Estética). Este marco de trabajo realiza la división de los videojuegos en tres componentes: mecánicas, dinámica y estética[36]. El primero describe las partes del juego a nivel técnico; es decir, la representación de datos y algoritmos. El componente dinámico describe el comportamiento de los mecanismos a partir de las interacciones que tiene el usuario con el juego en tiempo de ejecución. Y por último, la estética describe las respuestas emocionales deseadas que debe evocar el jugador cuando interactúa con el juego.

A través de métodos de Aprendizaje Supervisado es posible realizar ajuste de la dificultad en videojuegos. En los juegos comerciales, la finalidad de hacerlos divertidos e interesantes es aumentar las ganancias [29]; mientras que en juegos serios permite mejorar el nivel de aprendizaje del jugador. Dos ejemplos clásicos de aprendizaje supervisado es clasificación y

regresión, donde la salida es categórica (las clases) y numérica respectivamente.

Entre las técnicas populares de Aprendizaje Supervisado se encuentran redes neuronales, árboles de decisión, máquinas de soporte vectorial e inferencia bayesiana. En el contexto de los videojuegos, el uso de Aprendizaje Supervisado se centra en la imitación, análisis y predicción del comportamiento del jugador; desarrollo de mejores oponentes y la adaptación dinámica de la dificultad del juego [40, 46, 65]. Con un vasto conjunto de datos de métricas de comportamiento recopilados de jugadores es posible imitar el comportamiento de un jugador real y realizar predicciones en otros aspectos. En la siguiente sección se presenta un caso de estudio realizado por Missura y Gärtner [46], donde muestran una forma de realizar ajuste dinámico de dificultad en un juego sencillo de disparo.

2.4.1. Caso de Estudio: Modelado de Jugadores para el Ajuste Dinámico de la Dificultad

La forma tradicional en la que los juegos ajustan los diferentes niveles de dificultad es por medio de la selección del usuario. Por lo general estos permiten que el jugador seleccione entre principiante, medio y experto. Sin embargo, no todos los jugadores encajan en estos tres niveles, así que la selección puede no llegar a ser tan satisfactoria como se espera. Es posible aumentar la cantidad de niveles de dificultad, pero esto se convierte en un problema en el momento de seleccionar el nivel adecuado. Una forma óptima de ajuste de de la dificultad es que el juego sepa identificar el nivel de un jugador y ajustar la dificultad para dicho nivel.

Missura y Gärtner [46] presentan una forma de implementar ajuste dinámico de la dificultad en videojuegos a través del aprendizaje supervisado. En este caso de estudio, se recolectan datos de la fase de desarrollo del juego, con los que se realiza el modelo de dificultad, para realizar un ajuste de la dificultad adecuado. El enfoque utilizado para construir el modelo de dificultad consta de tres partes: realizar clustering del conjunto de datos de comportamiento de los jugadores, obtener el promedio de cada cluster y aprender a predecir la dificultad de cada cluster en un corto tiempo.

En los experimentos realizados se infiere que entre todos los jugadores posibles se pueden clasificar en tres tipos: principiantes, promedio y experto. A partir de dos fuentes de información, se realizan los análisis para determinar la dificultad apropiada para un jugador particular. Estas fuentes de información son:

1. Los datos obtenidos en la fase de pruebas, denominada como la fase fuera de línea.

2. Los datos obtenidos del nuevo jugador o fase en línea.

En la primera fase, los testers son los encargados de realizar la configuración de la dificultad. De esta forma, el juego aprende a identificar el ajuste de dificultad apropiado para un tipo de jugador determinado. Así, en la segunda fase, el juego solo necesita determinar a qué tipo de jugador pertenece el usuario y aplicar el modelo aprendido. Con el fin de adaptar el juego de forma rápida al nuevo jugador, se propone dividir las trazas del juego en dos partes: una parte pequeña de los datos iniciales se utiliza para entrenar el algoritmo en la fase fuera de línea, para luego en la parte en línea realizar la predicción; y el resto de datos se utilizan para realizar el clustering.

Los experimentos realizados por Missura y Gärtner contienen datos de como 17 jugadores interactúan con el juego. Se utilizan el algoritmo K-means [32, p. 402] para la fase de clustering y el algoritmo SVM, en ingles *Support Vector Machine*, con función gaussiana[12] para la parte de predicción. Para el ajuste de la dificultad se realizan dos modelos: el modelo constante, donde dado un cluster, se obtiene el promedio de sus instancias durante el tiempo, lo cual nos resulta en un ajuste estático de dificultad; y el modelo de regresión, donde dado el cluster, se entrena el RLSC (Regularized Least-Squares Classification)[53] con función gaussiana de sus instancias.

Para hacer las pruebas del enfoque propuesto, realizan un juego sencillo donde el jugador controla un cañón con las que puede disparar a naves espaciales extraterrestres, y estas a su vez, le disparan al cañón. Un máximo de 5 naves pueden estar al tiempo en la pantalla. Aparecen en el lado derecho de la pantalla, y se mueven a una altura constante de derecha

a izquierda. A las naves se les asigna una velocidad aleatoria en un intervalo específico. Cuando una de las naves cae o sale de la pantalla, una nueva es generada. Al inicio, el cañón hace un daño determinado a las naves; sin embargo, cada vez que este recibe un impacto de las naves, el daño que realiza se disminuye. Cada determinado tiempo un kit de reparación cae desde la parte superior de la pantalla y luego desaparece. Si el jugador consigue llegar al kit, el daño que hacen las balas del cañón aumenta a uno. El juego puede terminar de dos formas, si el daño realizado por el cañón disminuye a cero o si alcanza el tiempo límite, el cual es 100 segundos.

Figura 2.7: Pantallazo del juego [46, p. 6]

En los controles del jugador, también tiene dos botones para aumentar o disminuir la dificultad en cualquier momento del juego. La dificultad del juego es controlada por la velocidad promedio del las naves. Por cada nave destruida, el jugador recibe una cantidad específica de puntos, los cuales aumentan al cuadrado con el nivel de dificultad. Entre las variables que fueron guardadas, aquellas que mejor representaban el estado del jugador son: el nivel de dificultad, la puntuación y la salud.

Para las pruebas realizadas se tuvieron en cuenta las siguientes consideraciones técnicas:

1. Para disminuir la carga computacional de los datos guardados, solo se persistieron datos cuando el juego o el estado del jugador cambiaban. Este el caso particular de un

juego sencillo, esta preocupación puede parecer innecesaria; sin embargo, para juegos complejos con gran cantidad de variables definidas llega a ser un tema complejo.

2. Debido a que la partida podía terminar antes de los 100 segundos, es posible encontrar irregularidades en la duración de las trazas. con el fin de obtener un conjunto de datos homogéneo, en el caso de las partidas no tuviera el máximo de instancias registradas, se realiza la interpolación de datos para conseguir una cantidad uniforme de estas durante todas las partidas.

La última consideración técnica es debido a que en los experimentos realizados los primeros 30 segundos el juego observa cómo interactúa el usuario es necesario tener una cantidad homogénea de datos. En la parte inicial, las trazas son utilizadas para el entrenamiento y evaluación del predictor. En la segunda fase, las trazas son utilizadas para realizar clustering.

Para evaluar el rendimiento del predictor SVM se realiza validación cruzada separando los datos en dos grupos, uno para el entrenamiento y otro para las pruebas. El conjunto de datos de entrenamiento consiste en todas las instancias de los jugadores excepto uno. El conjunto de datos de prueba consiste en todas las instancias del jugador que no fue incluido en el conjunto de datos de entrenamiento. La construcción de los conjuntos de entrenamiento y pruebas se desarrollan esta forma debido a que se ajusta a situaciones reales. Como variable de evaluación de rendimiento se utiliza la diferencia del error absoluto entre el comportamiento exhibido en el conjunto de pruebas y el comportamiento descrito en el modelo del cluster.

Para la construcción de una línea base para el desempeño de la evaluación, se construye para cada instancia una secuencia de predictores: el primero selecciona el cluster que entrega el error absoluto menor, el segundo selecciona el cluster con menor error absoluto entre los restantes, y así. Para cada predictor se reliza el promedio de todas las instancias de prueba y el error del predictor SVM es comparado con estos valores. La figura 2.10 muestra el desempeño del predictor SVM, las mejores y peores líneas base para un solo jugador y 7 clusters. Se puede observar que el predictor SVM se acerca bastante al mejor cluster. En

la figura 2.13 se muestra el rendimiento del predictor SVM realizando una división en los datos, una parte para el conjunto de entrenamiento y los restantes para las pruebas.

En la parte de resultados se realizan validaciones estadísticas (pruebas de los rangos con signo de Wilcoxon), comparación del modelo estático contra el dinámico, calidad del predicción de clusters, entre otras, que permiten demostrar que la hipótesis presentada en este caso de estudio de predecir la dificultad para cada jugador basándose en cortos periodos de tiempo es viable. El ajuste dinámico de la dificultad a través del modelo de regresión siempre mejora significativamente el modelo constante para seleccionar el mejor cluster. De este modo, los resultados experimentales confirman que el uso de ambos, el ajuste dinámico y la predicción de clusters, superan un ajuste estático.

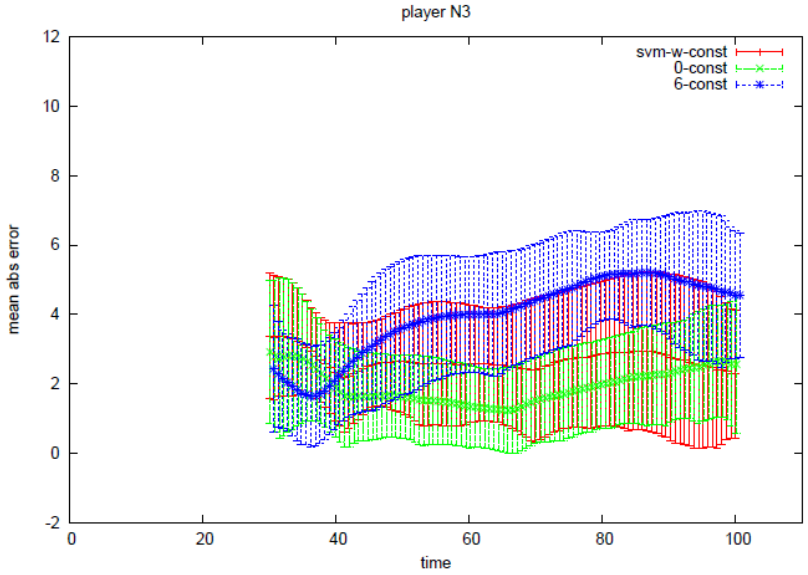


Figura 2.8: Usando el modelo constante.

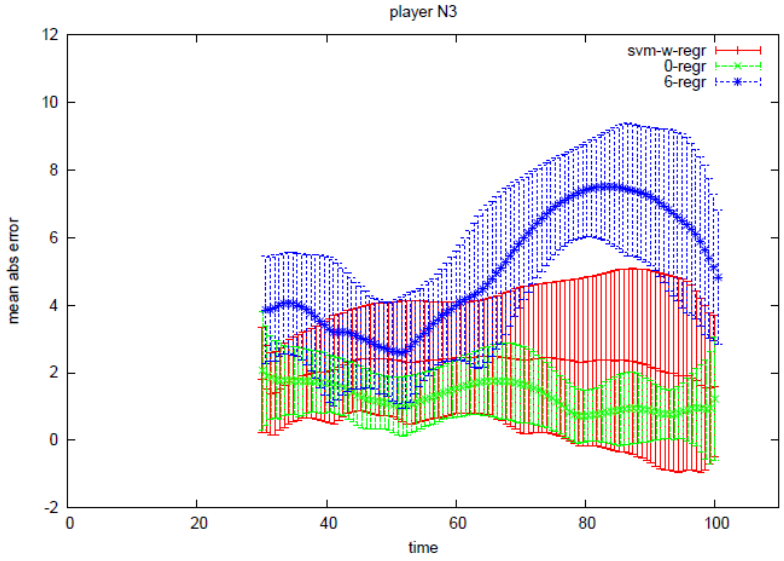


Figura 2.9: Usando el modelo de regresión.

Figura 2.10: Ejemplo del desempeño del predictor para un jugador[46, p. 10].

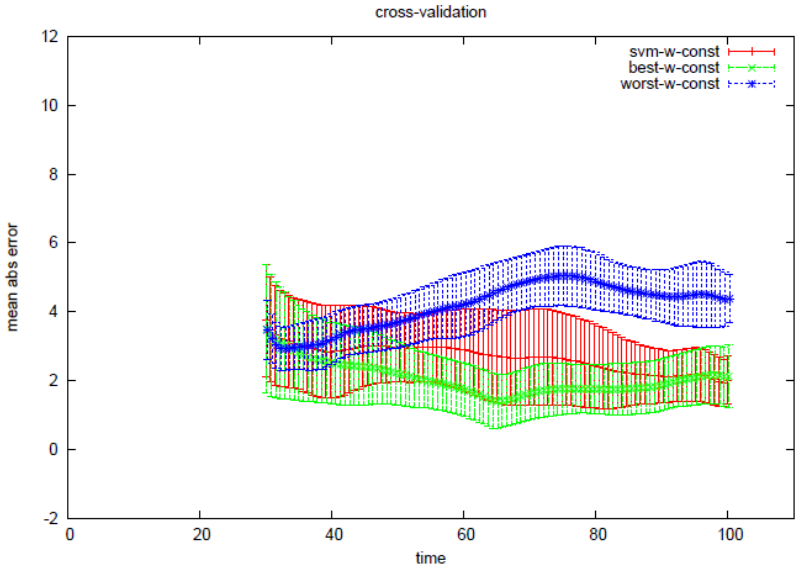


Figura 2.11: Usando el modelo constante.

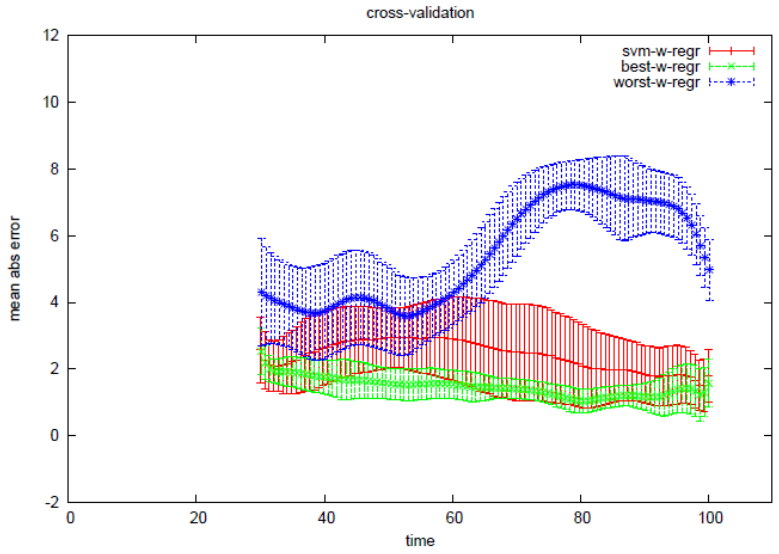


Figura 2.12: Usando el modelo de regresión.

Figura 2.13: Desempeño del predictor al dividir los datos de entrenamiento y pruebas para 7 clusters[46, p. 11].

Capítulo 3

Metodología de Clasificación de Jugadores y Ajuste de Dificultad a partir de Trazas de Videojuegos

En este capítulo se introduce una metodología cuya finalidad es la clasificación de jugadores y ajuste de dificultad a partir de trazas de videojuegos. En el capítulo 2 se presentó el proceso de descubrimiento del conocimiento a través del estándar CRISP-DM, el cual es popular en la industria de los videojuegos, y diferentes técnicas de Aprendizaje Automático para la clasificación de jugadores y ajuste dinámico de la dificultad. Con las diferentes etapas de CRISP es posible analizar, preparar y modelar el comportamiento de los jugadores con los datos que estos generan en las sesiones de juego. Con el modelo, se evalúan datos de nuevas partidas y se asigna un perfil mientras la partida esté activa. Y así una vez identificado el perfil del jugador, es posible personalizar la dificultad de la sesión en curso.

La metodología para el ajuste de la dificultad propuesta en este trabajo se basa en el estándar CRISP-DM y el caso de estudio de Missura y Gärtner[46]. En el caso de estudio se tiene un juego sencillo de naves espaciales en el que se realiza el ajuste de la velocidad de las naves enemigas dependiendo de las habilidades del usuario. Por partida siempre genera la misma cantidad de trazas, puesto que ésta tiene un tiempo límite. Aunque la habilidad del jugador no sea la mejor(pierda antes del tiempo límite), es posible tener un conjunto de datos homogéneo de las partidas por medio de la interpolación de datos. En el caso de

un juego de Tetris, no es posible tener la misma cantidad de trazas en todas las partidas evaluadas, independientemente si el usuario es bueno o malo, principalmente porque las sesiones de juego no tienen límite de tiempo y cada instancia representa una pieza ubicada en el tablero; por lo cual, si se agregan más instancias para forzar que todas las partidas tengan una cantidad homogénea de datos deteriora su veracidad. La manera de tolerar este inconveniente es agregar partidas de jugadores de diferentes niveles de habilidad hasta que la cantidad de instancias sea semejante en cada conjunto.

Después de tener el modelo de comportamiento de los jugadores, Missura y Gärtner realizan la predicción de nuevas partidas con datos recopilados al inicio de la partida. De forma similar, se realiza la predicción con TetrisAnalytics; sin embargo, no se toma un rango de tiempo determinado al inicio de la partida como en el estudio en cuestión, sino que se crean "límites" en el tablero, que al ser superados se realiza la predicción con todos los datos recopilados hasta el momento.

La metodología propuesta contiene dos ciclos importantes: el modelado de comportamiento y el ajuste de dificultad. En el primero se tienen las fases de análisis de diseño, entendimiento de variables, preparación de datos y modelado de perfiles. A partir del modelo, es posible determinar y asignar un perfil a un nuevo jugador durante la partida. El segundo ciclo inicia con la evaluación de nuevas trazas, se predice el perfil al que corresponde el jugador tomando un porcentaje de los datos generados durante la partida en curso y así, realizar el ajuste de dificultad correspondiente. El modelo de ajuste de la dificultad es estático, tiene en consideración el diseño del juego y los niveles de habilidad exhibidos en la fase de modelado.

El proceso de dificultad de un videojuego (Figura 3.1) tiene las siguientes actividades:

1. **Análisis de diseño:** Esta fase se centra en el análisis de diseño del juego, sus reglas, la interacción permitida y las habilidades que son evaluadas a sus jugadores.

2. **Entendimiento de variables:** Esta fase consta del análisis de variables que se recopilan a partir de la interacción jugador-juego. Entre las actividades a realizar están:

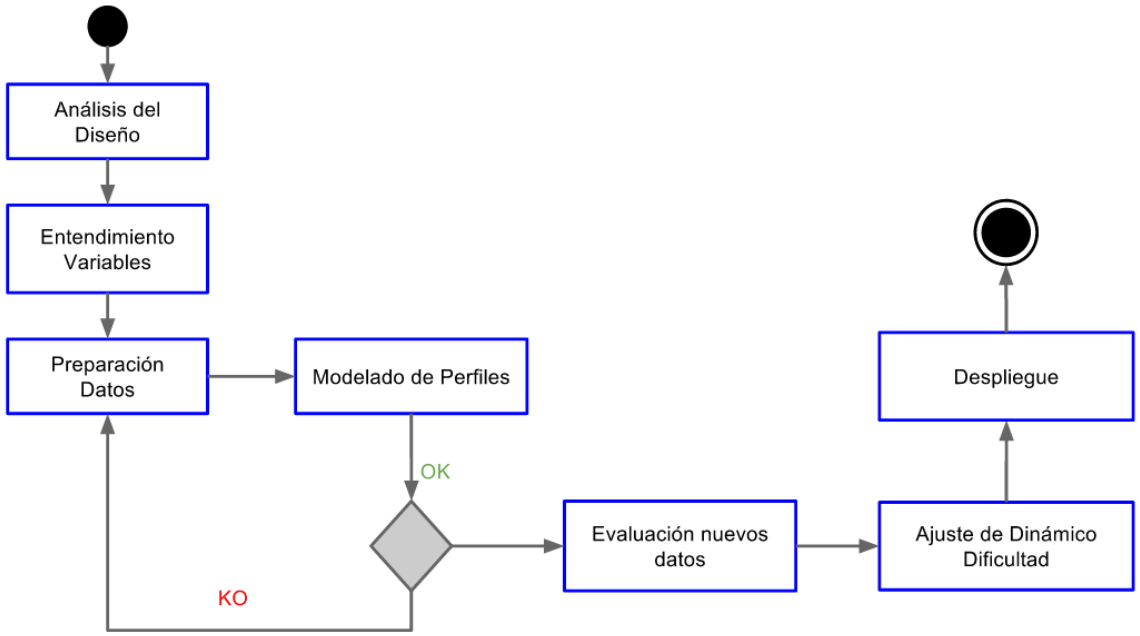


Figura 3.1: Metodología propuesta para realizar ajuste dinámico de la dificultad en videojuegos.

- Extracción de datos.
- Análisis y evaluación de la calidad de los datos.
- Selección de variables que son utilizados en las etapas posteriores.

3. **Preparación de datos:** Esta es una de las actividades más extensas y tediosas. Como resultado de esta fase se obtiene el conjunto de datos final a evaluar en la próxima fase. Entre las actividades más comunes en esta etapa se encuentran:

   - Preparación de los datos.
   - En caso de ser necesario, dar formato deseado a los datos.
   - En caso de ser necesario, se crean variables a partir de la agregación de dos o más datos básicos.
   - Eliminar datos irrelevantes o errados.

4. **Modelado de perfiles:** La etapa de modelado de perfiles se puede realizar varias veces con el fin de refinar los resultados obtenidos, agregar o quitar variables, utilizar diferentes técnicas de minería de datos y modificar parámetros de entrada para optimizar los resultados.

5. **Evaluación nuevos datos:** Al igual que la etapa de modelado, esta fase se puede realizar varias veces. En una primera instancia se hace para asignarle un perfil al jugador dependiendo de sus trazas. Luego se realiza con el fin de validar que la asignación inicial continúa siendo la correcta.

6. **Ajuste de dificultad:** Los diferentes perfiles de los jugadores se encuentran relacionados con un ajuste específico de la dificultad. En el caso de estudio de Missura y Gärtner[46], desarrollan un modelo que define la velocidad en que las naves extraterrestres se desplazan en la escena del juego. Dependiendo del perfil que se le es asignado al jugador (principiante, medio o experto), se ajusta la velocidad de sus enemigos. Asimismo, en esta fase se debe relacionar el nivel de dificultad a un perfil del jugador. De esta forma, cada perfil de usuario tiene asociado su ajuste de dificultad correspondiente.

7. **Despliegue:** En en despliegue se realiza el ajuste de dificultad sobre el juego; es decir, dependiendo de la habilidad predicha del usuario, se realiza un cambio en la dinámica del juego para que este sea más sencillo o más complejo.

3.1. Implantación de la Metodología con TetrisAnalytics

3.1.1. Análisis del Diseño

Para probar la metodología de clasificación de jugadores y ajuste de dificultad se utiliza el juego TetrisAnalytics. La implementación de este juego de Tetris es sencilla, se tiene 7 diferentes piezas y al pasar el tiempo, la ficha cae más rápido por el tablero. El jugador puede realizar los movimientos tradicionales (izquierda, derecha, rotación y bajar la ficha).

La aplicación TetrisAnalytics permite jugar una única partida de Tetris, sin ningún tipo de interfaz previo ni posterior. Al terminar la partida la aplicación acaba y se genera un archivo binario que contiene la información de toda la sesión. Con este binario es posible reproducir la partida, obtener datos de la duración de esta, puntuación total, nombre del jugador, si este lo registró al iniciar la sesión.

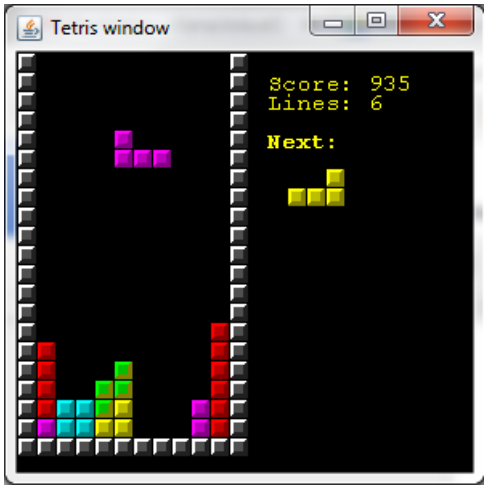


Figura 3.2: Pantallazo de TetrisAnalytics.

TetrisAnalytics tiene varias IA's pre-configuradas con las cuales se puede generar trazas. Las IA's AlwaysLeft, AlwaysRight y RandomAI son sencillas y solo mueven la ficha siempre a la izquierda, siempre a la derecha y de forma aleatoria respectivamente. Otra es HardCodedTacticalAI que tiene un nivel alto de habilidad y obtiene buenas puntuaciones. También con datos de jugadores reales generar IA's que jueguen de forma similar. Existen dos tipos de IA en TetrisAnalytics: aquellas encargadas de decidir dónde se coloca la ficha (IA táctica), y las encargadas de mover la ficha hasta su posición final (IA de movimiento). El sistema permite separar ambas IA's, de esta forma es posible generar posteriormente trazas que contengan los movimientos tácticos; es decir, cada instancia tenga la información de dónde se ubicó la ficha. Mientras que las IA's de movimiento generan trazas con los datos celda a celda de la pieza mientras baja por el tablero. Las tres IA's descritas hacen parte del conjunto total de datos, al igual que otros jugadores con diferentes niveles de habilidad.

Como se menciona previamente, al finalizar la partida se guarda un archivo binario con todos los datos de esta. A partir de este binario se pueden generar archivos ARFF que posteriormente serán ingresados en Weka[49, 62] para su análisis. Los ficheros ARFF son documentos de texto que empiezan con un preámbulo en el que aparece información de las variables (cuántas entradas hay y sus tipos) y luego contiene los datos[63]. Cada instancia tiene el estado de la partida y la acción que realizó el jugador; es decir, los datos del estado del tablero, ficha actual que baja (tipo de ficha, posición y rotación) y acción que realizó el jugador (izquierda, derecha, rotación, bajar). La información que contienen estos archivos puede ser configurada en el momento de generarlos. En la práctica, el caso de entrenamiento puede contener más o menos información dependiendo de las variables de interés. Existen diferentes políticas de generación de ARFF donde se puede parametrizar la variables a exportar. Las políticas de generación existentes son:

- **PosFichaARFF:** guarda la posición de la ficha actual y la acción del usuario.
- **PosYTipoFichaARFF:** guarda todos los atributos de la política anterior más el tipo de ficha actual.
- **LastXLinesARFF:** guarda los datos de las fichas (posición, rotación, tipo de ficha y ficha siguiente). Al igual que la parte superior del tablero; es decir, las últimas filas no vacías. La X del nombre puede variar entre 1-4, lo cual indica la cantidad de filas que se guardan. La posición de la ficha se almacena relativa a la Y de la fila superior.
- **EstadoCompletoARFF:** guarda todo el estado del juego, esto incluye: posición de la ficha, rotación, tipo, ficha siguiente y estado completo del tablero (casilla libre=0, casilla ocupada=1).

3.1.2. Entendimiento de variables

TetrisAnalytics permite generar dos tipos de trazas: aquellas que contienen los datos de la pieza en un instante de tiempo determinado, y los movimientos tácticos de los jugadores.

Este último posee los datos de la ubicación final de la pieza; o sea, cada instancia contiene dónde el jugador ubica la ficha actual, la acción efectuada y el estado del tablero, si es necesario. Las trazas que se utilizan son aquellas que incluyen los movimientos tácticos de los jugadores, así se puede analizar el estilo de juego de cada usuario a partir de la ubicación de las fichas en el tablero.

Al generar ARFF con movimientos tácticos, las variables que tiene cada instancia son las siguientes:

- **PosX:** posición X donde el jugador ubica la pieza. Esta variable es una enumeración con valores: x0, x1, x2, x3, x4, x5, x6, x7, x8, x9; donde x0 es la posición más a la izquierda del tablero.
- **PosY:** posición Y donde el jugador ubica la pieza. Esta variable es una enumeración con valores: y0, y1, y2, y3, y4, y5, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15, y16, y17, y18, y19; donde y0 es la posición superior del tablero.
- **Rot:** rotación de la ficha. Esta variable es una enumeración con valores: r0, r1, r2, r3.
- **NextPiece:** pieza siguiente que saldrá a continuación (valor entero entre 0 y 6).
- **Celda_X_RelY:** X es un entero de 0 al ancho - 1 del tablero. Y corresponde a las K últimas filas del tablero. Cada valor contiene el estado de la celda concreta del tablero (libre=0, ocupada=1).
- **TipoFicha:** tipo de ficha actual. Esta variable es una enumeración con valores: p0, p1, p2, p3, p4, p5, p6.

Con el conjunto de variables generadas con movimientos tácticos, se realiza el modelado de perfiles de jugadores y la evaluación de nuevos jugadores.

3.1.3. Preparación de datos

En la fase de preparación de datos se realizan las siguientes actividades:

- **Extracción de archivos ARFF:** Dependiendo de la política de generación de datos (PosYTipoFichaARFF, Last4LinesARFF y EstadoCompletoARFF), se tienen más o menos características de la partida. Por esta razón, se realiza la evaluación de los datos generados con cada una de las políticas. Así, dependiendo de los resultados obtenidos se puede conocer qué variables son más relevantes para la clasificación y predicción de nivel. Además, la puntuación y tiempo total de la partida son características que distinguen a un jugador experto de uno principiante. A partir de éstas se determinan los niveles de los jugadores que componen el conjunto de entrenamiento.

- Adición de identificador del jugador, puntuación total, tiempo total y nivel de habilidad (clase).

- **Unión de 2 o más movimientos tácticos consecutivos por partida:** Resulta complejo determinar si un usuario es bueno o no, a partir de un solo movimiento; especialmente, si el espacio de acciones posibles dentro de el juego es restringido. Una forma de determinar el nivel de un usuario es a partir de la ubicación de piezas consecutivas y de las acciones realizadas. Por ejemplo, si después de ubicar 5 piezas en el tablero la cantidad de posiciones en Y aumenta rápidamente, puede significar que el jugador tiene una habilidad baja. De igual forma, la cantidad de rotaciones realizadas a las piezas puede distinguir a un jugador bueno de uno no tan bueno.

3.1.4. Modelado de perfiles y Evaluación de nuevos datos

En la fase de modelado se realizan varias evaluaciones sobre los datos de las partidas. Estas actividades son:

1. **Selección de variables:** Se evalúan las trazas de las tres políticas de generación (PosYTipoFichaARFF, Last4LinesARFF y EstadoCompletoARFF) en el modo "classes-to-clusters" donde la clase es el identificador de los jugadores y se utilizan los algoritmos SimpleKMeans [4, 5] y EM (Expectation Maximization) [3] con Weka Explorer.

2. **Selección de algoritmo:** Se realiza la prueba T-Test pareado para determinar cuál algoritmo tiene mejor desempeño. El conjunto de entrenamiento que se utiliza es aquel con mayor porcentaje de clasificación en la actividad previa.

3. **Creación de clases:** Las clases asignadas a cada instancia corresponde al nivel de habilidad. Por medio de la puntuación total de la partida se determina el nivel de habilidad del jugador. La *discretización de la puntuación* de todas las partidas que conforman el conjunto de entrenamiento permite conocer los rangos en los que se encuentran la mayoría de los jugadores.

4. **Descripción de características principales de tipos de jugadores:** Analizando las características principales de cada cluster de jugadores se asocia el nivel de habilidad con los movimientos y estado del tablero.

5. **Predicción nuevas instancias:** Con la predicción de nuevas instancias se valida cuán efectivo es el modelo creado. Para verificar la predicción se utilizan partidas completas cuyas instancias tienen una clase asignada (la clase se obtiene a partir de la puntuación total de la partida). Así al utilizar el algoritmo de predicción, se podrá comprar la cantidad de aciertos que se obtienen. Sin embargo, al implementar este enfoque en el juego, no se tiene disponible toda esta información por lo que la clasificación de instancias se realiza sin un etiquetado previo; sino que se conoce a partir de las características del grupo al que pertenece.

3.1.5. Ajuste de dificultad y Despliegue

La dificultad de un juego como Tetris se centra en la velocidad de caída de la pieza y el tipo de ficha actual y siguiente disponible. La habilidad del jugador se mide por la cantidad de líneas que este puede hacer antes que la velocidad del juego sea lo suficientemente alta, no pueda ubicar las piezas en el sitio deseado, las fichas alcancen la parte superior del tablero y pierda. La velocidad de la caída de la pieza no se puede alterar, ya que esto hace que

el juego se convierta, hasta cierto punto, en aburrido e interminable, la mejor opción para ajustar la dificultad es a partir del tipo de ficha disponible para el usuario. Existen piezas en Tetris que son más complejas que otras porque debido a su forma crean espacios en las líneas. Una manera de ajustar la dificultad es evaluando el estado del tablero e identificar qué ficha es ideal para que el jugador obtenga mejor puntuación. Este ajuste será efectuado proporcional a la habilidad del jugador; es decir, cuan menor habilidad tenga, mayor ayuda tendrá del juego. Análogamente, los jugadores considerados como expertos reciben poca o ninguna ayuda del sistema.

La forma para determinar la habilidad del jugador es por medio de su clasificación. Los datos tomados para la evaluación serán las primeras trazas generadas de la partida. El tablero tiene 20 celdas de altura, así que este es divido en 4 partes iguales, si el jugador pasa alguno de estos umbrales, se evalúa la habilidad que tiene con las trazas almacenadas hasta el momento. Cuando más de la mitad de los movimientos tácticos que componen una instancia superan dicho umbral, entonces se realiza la clasificación. Por ejemplo, si se tiene un jugador que supera las primeras 5 filas inferiores del tablero, y 6 piezas de la misma traza son ubicadas por encima de las primeras 5 filas, entonces se clasifica de acuerdo a su comportamiento y se ajusta acorde a su categoría. El nivel del jugador actual se calcula a partir de la clase predicha para cada instancia y se selecciona aquella clase con mayor ocurrencia.

La implementación del ajuste de dificultad es posible a través de la reutilización de métodos ya existentes de la clase `HardCodedTacticalAI`. Esta IA evalúa el tablero con el método `rateBoard` para cada rotación de la pieza actual y así, ubicar la ficha en la mejor posición. Este método devuelve un valor numérico que representa la valoración que tiene el tablero en ese momento, cuan más pequeño sea el valor, mejor es el estado del tablero. El método `rateBoard` evalúa la altura máxima que alcanzan las piezas en el tablero (la posición y con menor valor) y la cantidad de espacios que ha dejado el jugador entre las piezas. El método `numHuecos` es el encargado de contar la cantidad de espacios que tiene

una columna a partir de la pieza más al norte.

Las IA's tácticas implementan la interfaz `TacticalAI`. Esta interfaz contiene el método `getTacticalAction`, cuya funcionalidad es decidir dónde ubicar la pieza. Para el caso particular de `HardCodedTacticalAI`, se evalúan todas las posibles configuraciones sin tener en cuenta la pieza siguiente. A través del método `tryFinalPosition`, se copia el estado actual del tablero con la nueva pieza y devuelve la calificación del tablero. Esto se repite para cada una de las 4 rotaciones posibles. Y Finalmente, se devuelve la mejor acción táctica.

Por otro lado, en el bucle principal del juego se realiza el llamado al método `pickPiece`. Este método es el encargado de devolver la siguiente pieza al jugador. Si se pasa como parámetro el valor -1, entonces el método selecciona una pieza de manera aleatoria. Pero si se ingresa un valor específico, la siguiente pieza es la correspondiente a ese valor (entero de 0 a 6). Previo a este método, se crea el un método que evalúe el tablero para cada una de las piezas del juego, acorde al resultado se solicita a `pickPiece` que ésta sea la ficha siguiente. Así dependiendo de la habilidad del jugador, se le entrega la pieza con mejor calificación de tablero.

Capítulo 4

Experimentos Realizados con TetrisAnalytics

En este capítulo se ejecutan diferentes experimentos que tienen como finalidad el ajuste de dificultad en partidas activas de Tetris. Para ello, se desarrollan un conjunto de pruebas que permiten conocer cuáles son las variables y el algoritmo óptimo, cómo se relacionan las variables con el nivel de habilidad y cómo predecirlo, y cuándo es necesario realizar un ajuste de dificultad.

Después de conocer la interacción que tiene el usuario con el juego y las variables que son monitoreadas y almacenadas, es necesario evaluar cuáles de éstas son idóneas para la clasificación de un jugador por habilidad. En la primera etapa se determinan las variables que mejor clasifican a un jugador a partir de su habilidad. En Tetris, el nivel de un jugador es directamente proporcional a la puntuación que éste obtenga en la partida, el tiempo de duración y la cantidad de líneas que realice. Estas variables deben ser relacionadas con los movimientos tácticos que realiza el jugador; así, se vincula el nivel con el estilo de juego. Los pasos a seguir son los siguientes [42]:

- Seleccionar las variables que distinguen el estilo de juego: Desarrollar pruebas con movimientos tácticos individuales y consecutivos vía SimpleKMeans y EM. Para cada uno de los algoritmos y políticas de generación se evalúa el conjunto de datos en modo "classes-to-clusters", donde k es el número de jugadores y la clase es el identificador de

cada jugador.

- Determinar el nivel de habilidad de los grupos: Determinar las características que hacen a un jugador mejor que otro a partir de los movimientos que éste realiza.
- Elegir el algoritmo con mayor desempeño: Se realiza la prueba T-Test pareado para seleccionar el algoritmo con mejor desempeño.

En segunda instancia, a partir de un conjunto de entrenamiento y de las variables seleccionadas, se desarrolla un modelo que describe el comportamiento de cada tipo de jugador dependiendo de su habilidad. En otras palabras, el modelo identifica el nivel de un jugador basándose en los movimientos que éste realiza durante la partida. Esta fase es llamada evaluación de estilos y su finalidad es definir los diferentes comportamientos que tienen los jugadores y las características principales de cada grupo. Así es posible realizar el modelado de los jugadores a partir de su habilidad y estilo de juego.

Finalmente, se realiza la predicción de los jugadores con los movimientos tácticos a partir del modelo de comportamiento generado. Para cada jugador con un nivel de habilidad específico, se evalúan sus trazas y se determina qué tan acertado es el modelo construido. Una vez calculada la habilidad del jugador, es posible efectuar el ajuste de dificultad.

4.1. Resultados

4.1.1. Consideraciones Técnicas

La preparación de los datos es la etapa más extensa de todo el proceso. A partir de los binarios generados por el juego, se producen archivos ARFF de cada partida. Con la clase `CookTrace` de TetrisAnalytics es posible reproducir una partida. Con `PointsTraceWriter` y `GameDurationTraceWriter` se exporta la puntuación y el tiempo de la partida que se está reproduciendo respectivamente.

Una vez generados los archivos ARFF con la política de generación designada, se realizan las actividades agregan los atributos que faltan; es decir, la puntuación y el tiempo de la

partida a la que corresponde la instancia. Y se hace la unión de n movimientos tácticos consecutivos.

4.1.2. Selección de Variables

TetrisAnalytics tiene tres políticas de generación de archivos ARFF, dependiendo de la política de generación seleccionada, las trazas resultantes tienen más o menos variables como se explica en la sección 3.1.1. Con el fin de verificar las variables y algoritmo que mejor clasifican a los jugadores de Tetris se realizan 3 experimentos en este apartado: evaluación de movimientos tácticos individuales, evaluación de movimientos tácticos consecutivos y comparación de desempeño de los algoritmos[42]. Las evaluaciones de los movimientos tácticos individuales y consecutivos se realizan para cada una de las políticas de generación (PosYTipoFichaARFF, Last4LinesARFF y EstadoCompletoARFF) en modo "classes-to-clusters" con los algoritmos SimpleKMeans y EM vía Weka Explorer. La clase utilizada corresponde al identificador del jugador. El objetivo de esto es conocer las características más importantes, si es posible diferenciar movimientos de diferentes jugadores y seleccionar el mejor algoritmo.

El análisis efectuado con los movimientos tácticos individuales se desarrollaron con trazas de 6 jugadores. Tres de los jugadores son IA's, una posee buenas habilidades para jugar Tetris (jugador 3). Las otras dos son "tontas" y sus movimientos consisten en poner las fichas siempre en el lado izquierdo o derecho del tablero (jugador 5 y jugador 4 respectivamente); mientras que los demás jugadores distribuyen las fichas sin preferencias notorias a lo ancho del tablero. Esto se puede examinar en la figura 4.1, donde cada color representa un jugador. Con la política de generación PosYTipoFichaARFF se exporta el tipo de ficha actual, el tipo de ficha siguiente, la posición x y y donde se ubica la ficha, y la rotación que tiene la pieza en su ubicación final. Con Last4LinesARFF y EstadoCompletoARFF, se agrega la información de las 4 filas no vacías superiores del tablero y el estado completo del tablero respectivamente.

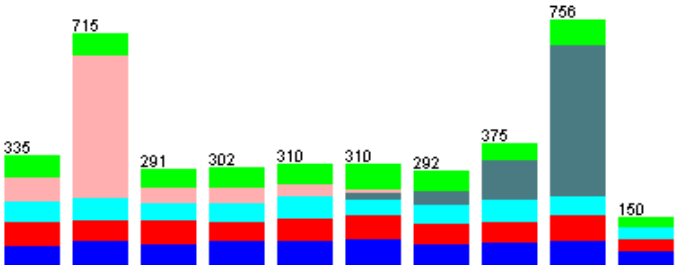


Figura 4.1: Característica PosX de ubicación de la ficha. Varia de $x0$ a $x9$. Donde $x0$ es la posición más a la izquierda.

La tasa de error para las pruebas con los movimientos tácticos individuales son las siguientes:

- **PosYTipoFichaARFF:** SimpleKMeans 70.61 %, EM 71.12 %.
- **Last4LinesARFF:** SimpleKMeans 44.15 %, EM 48.46 %.
- **EstadoCompletoARFF:** SimpleKMeans 45.06 %, EM 53.43 %.

Para todos los casos SimpleKMeans clasifica mejor que EM. Asimismo, aunque se esperaba que al tener la información del tablero completo diera mejores resultados que la información parcial de este, la mejor clasificación fue realizada teniendo la información de las 4 filas superiores no vacías del tablero. Debido a que la evaluación es realizada con los movimientos tácticos individuales de los jugadores, es posible que el estado completo del tablero, hasta cierto punto, no sea tan relevante debido a que solo se tiene en cuenta la ubicación de cada ficha, no de un conjunto de fichas consecutivas. Además existen zonas en el tablero a las que el jugador no puede afectar.

En la matriz de confusión (Cuadro 4.1), las columnas representan el número de predicciones del algoritmo y las filas las instancias en la clase real. La mayoría de las instancias de los jugadores 4 y 5 pertenecen a los clusters 0 y 3 respectivamente. Estos dos jugadores, debido a tener comportamientos extremos, su clasificación es bastante acertada. Sin embargo, el resto de jugadores pueden llegar a tener comportamientos similares y así ser clasificados

0(J6)	1(J2)	2(J4)	3(J3)	4(J1)	5(J5)	Clusters Asignados
131	104	28	147	203	56	J1
190	204	113	104	99	82	J2
29	76	1	371	194	30	J3
0	0	725	0	0	34	J4
0	0	0	0	4	760	J5
145	118	14	168	149	33	J6

Cuadro 4.1: Asignación de clases a clusters con SimpleKMeans, configuración Last4LinesARFF y movimientos individuales.

0(J1)	1(J3)	2(J6)	3(J5)	4(J4)	5(J2)	Clusters Asignados
176	107	170	40	36	139	J1
124	52	139	71	122	283	J2
84	398	156	21	0	41	J3
0	0	0	0	758	0	J4
2	0	0	761	0	0	J5
116	134	174	28	12	162	J6

Cuadro 4.2: Asignación de clases a clusters con SimpleKMeans, configuración Last4LinesARFF y unión de 2 movimientos consecutivos.

erróneamente. Esto ocurre porque el espacio de movimientos permitidos a los jugadores es bastante reducido. En consecuencia, la evaluación de partidas con movimientos individuales por jugador lleva a una mala clasificación. Por esta razón, se unen la instancia actual con la siguiente de una partida. Esta unión se realiza considerando que durante la partida el jugador puede observar la pieza siguiente a ubicar en el tablero. Así, se puede considerar que el jugador ubica la ficha actual de manera diferente si conoce cuál será la siguiente.

La tasa de error que presentan las pruebas al unir la traza de la pieza actual con la siguiente son:

- **PosYTipoFichaARFF:** SimpleKMeans 60.03 %, EM 68.33 %.
- **Last4LinesARFF:** SimpleKMeans 40.78 %, EM 47.08 %.
- **EstadoCompletoARFF:** SimpleKMeans 52.32 %, EM 53.032 %.

0(J3)	1(J5)	2(J1)	3(Sin clase)	4(J2)	5(J6)	Clusters Asignados
35	5	68	54	29	58	J1
28	41	54	93	120	94	J2
243	0	0	59	0	97	J3
0	65	0	0	0	0	J4
0	65	0	0	0	0	J5
56	0	39	44	69	107	J6

Cuadro 4.3: Asignación "classes-to-clusters" con SimpleKMeans, configuración Last4LinesARFF y unión de 10 movimientos consecutivos.

En el caso de PosYTipoFichaARFF, se puede observar una disminución importante en el porcentaje de instancias incorrectamente clasificadas para ambos algoritmos. De modo similar, la configuración EstadoCompletoARFF disminuye respecto a la prueba anterior. Y Last4LinesARFF es la mejor clasificada. Para el caso de los jugadores 4 y 5, los cuales presentan comportamientos extremos, se puede observar una clasificación correcta de todas sus instancias en la tabla de confusión (Cuadro 4.2) resultante.

La cantidad de trazas producidas por partida depende la habilidad que tiene el jugador, se calcula la cantidad mínima de instancias que un jugador realiza por sesión en el conjunto de entrenamiento; por ello, se unen 10 movimientos tácticos consecutivos generados con la política Last4LinesARFF por partida. A pesar que el porcentaje de instancias correctamente clasificadas es menor que al unir dos instancias consecutivas, se observa en la tabla de confusión (Cuadro 4.3) que los dos jugadores principiantes (jugadores 4 y 5) son clasificados en el mismo grupo, lo cual no ocurría anteriormente. Estos dos jugadores tienen estilos de juego similares aunque uno ubica las piezas del lado izquierdo y el otro del lado derecho. Al unir 10 movimientos tácticos consecutivos, es posible reconocer que el estilo de ambos es similar y ubicarlos en un mismo cluster.

Prueba Estadística: T-Test Pareado

Con el fin de comparar el desempeño de los algoritmos utilizados, se realiza la prueba T-Test Pareada Corregida vía Weka Experimenter. Este método realiza la comparación de los

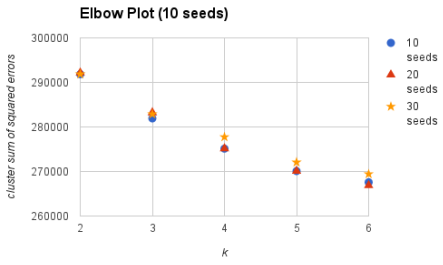
resultados de N algoritmos de clasificación sobre el mismo conjunto de datos[52]. Por defecto, el nivel significativo es de 0.05 y en el campo de comparación se selecciona *percent_correct*. El conjunto de datos ingresado es el generado a partir de la política Last4LinesARFF con dos movimientos consecutivos por instancia. En el resultado muestra que con la configuración SimpleKMeans ($t = 50{,}05$) supera a EM ($t = 46{,}48$).

4.1.3. Evaluación de Estilo de Juego

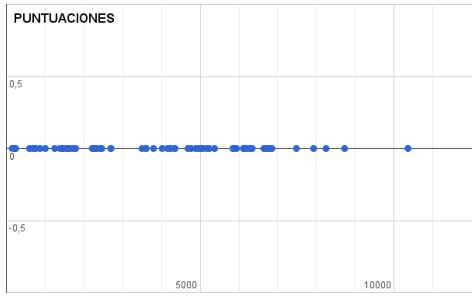
De acuerdo con las pruebas realizas en la sección 4.1.2, la unión de movimientos tácticos consecutivos mejora la clasificación de los jugadores. Además, se demuestra que la política de generación de archivos Last4LinesARFF clasifica mejor que las otras. En este apartado se describen los experimentos ejecutados para determinar si utilizando clustering es posible segmentar las trazas de los jugadores de acuerdo a su habilidad en el juego. El conjunto de entrenamiento está conformado por 174 partidas de 9 jugadores.

Con el fin de definir los niveles (clases), se genera un gráfico de dispersión (Figura 4.2a) que contiene en el eje horizontal a k variando de 2 a 6 y en el eje vertical la sumatoria de errores cuadrados de los clusters; y otro gráfico con las puntuaciones de los jugadores. Para el gráfico de dispersión, se selecciona SimpleKMeans como algoritmo de agrupamiento para 10, 20 y 30 centroides iniciales. Puesto que k-means realiza múltiples iteraciones con diferentes centroides, su variabilidad radica en la selección de los centroides iniciales; sin embargo, como se puede observar, no supone mayor diferencia así que se utiliza la configuración por defecto de 10 semillas. Además, los datos se dividen 70 % para el conjunto de entrenamiento y 20 % para el conjunto de pruebas. El gráfico de puntuación (Figura 4.2b) se puede observar 3 rangos de puntuaciones: $0 - 2999$, $3000 - 6999$ y $7000 - \infty$. Por consiguiente, $k = 3$ es la mejor opción para la división de grupos, donde cada uno corresponde a un nivel de habilidad: principiante, medio y experto.

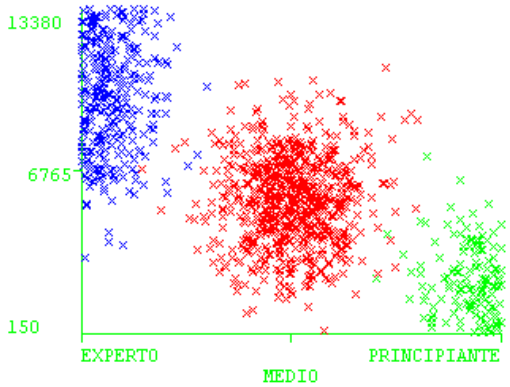
Por medio de la variable PosY se analiza cuán rápido aumenta verticalmente la posición de las piezas a lo largo de la partida; es decir, es posible determinar qué tan rápido el jugador



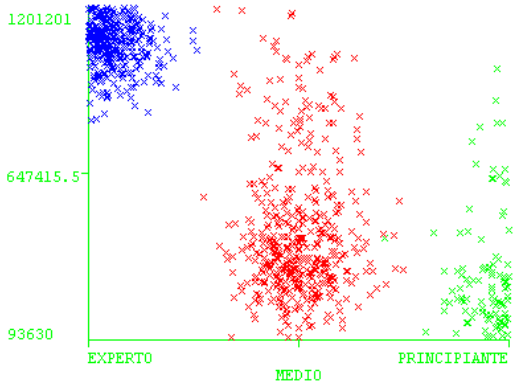
(a) Gráfico de dispersión para k de 2 a 6.



(b) Puntuaciones.



(c) Clases Vs. Puntuación total por partida (Para $k = 3$).



(d) Clases Vs. Tiempo total por partida (Para $k = 3$).

permite que las piezas alcancen la parte superior del tablero. En el caso de los jugadores con habilidades medias y expertas, se evidencia que las filas inferiores del tablero tienen mayor ubicación de piezas que aquellas situadas en la parte superior. En el caso de los principiantes, esta diferencia no observa. Con la variable PosX se conoce dónde el jugador posiciona horizontalmente las piezas en el tablero. Existen muchos jugadores que tienen preferencia de colocar las fichas primero en los extremos y luego en el centro, o lo contrario. Un ejemplo claro de esto es la barra (ficha roja), debido a su forma, los jugadores tienen a ubicar esta pieza en los extremos del tablero independientemente de si este es principiante, medio o experto. De manera similar ocurre con el cuadrado (ficha cían). Mientras que para otras piezas, la tendencia es ubicarlas en el centro del tablero.

Cluster	%P	Características
PRINCIPIANTE	46 %	La distribución de las piezas a lo largo del tablero es semejante. Tendencia a ubicar las piezas en los extremos. Baja rotación.
MEDIO	21 %	La mayor cantidad de piezas durante la partida son ubicadas en la mitad inferior del tablero (y10-y19). Ubica de manera homogénea las fichas a lo ancho del tablero. Alta rotación.
EXPERTO	33 %	La mayor cantidad de piezas durante la partida son ubicadas en la mitad inferior del tablero (y13-y19). Ubica de manera homogénea las piezas a lo ancho del tablero. Alta rotación.

Cuadro 4.4: Comportamiento interpretado de los clusters,
K-means, %P = % de jugadores en el grupo.

4.1.4. Predicción de Habilidades

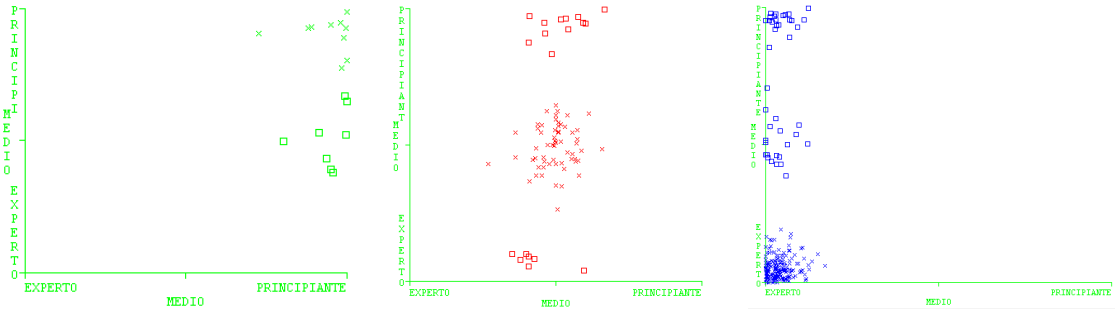
En esta sección se verifica la efectividad del modelo construido en el apartado 4.1.3. El conjunto de entrenamiento es el descrito en la sección anterior (174 partidas de 9 jugadores) y el conjunto de datos de prueba son nuevas partidas de jugadores con diferentes niveles de habilidad. Para cada nivel, se toma una partida completa cuyas instancias han sido previamente etiquetadas con los diferentes niveles de habilidad (principiante, medio y experto) a partir de la puntuación total; es decir, si un jugador tiene una puntuación entre 3000 y 7000, entonces sus instancias son consideradas como nivel de habilidad medio. La asignación de la clase tiene como finalidad verificar que la predicción las de instancias sea correcta; sin embargo, en la ejecución tradicional del juego no se asignaría una etiqueta previa puesto que no se tiene la puntuación total de la partida.

`ClassificationViaClustering` es un meta-clasificador sencillo que utiliza clusters para la clasificación en Weka[2]. Con este clasificador se predice la habilidad que demuestran los jugadores en nuevas partidas a partir de un modelo de comportamiento.

Clasificación de Principiantes En la clasificación de principiantes se utilizan varias partidas de un nuevo jugador. Esto es porque al ser principiantes, crean pocas trazas por partida y cuando se unen 10 movimientos tácticos consecutivos las instancias resultantes son pocas. Así que se toman 10 partidas de este nuevo jugador principiante y se evalúan con `ClassificationViaClustering`. En la figura 4.3a se tiene la gráfica de errores de clasificación. Varias instancias son clasificadas con clase MEDIO. En estas instancias se puede observar una rotaciones en la mayoría de los movimientos tácticos. Además, la diferencia de entre la variable `PosY_1` (primera movimiento táctico en la instancia) y `PosY_10` (décimo movimiento táctico en la instancia) oscila entre los rangos descritos para el jugador medio; en otras palabras, en estas instancias en particular, no se muestra un ascenso rápido de las piezas en el tablero. A pesar que existen varias instancias mal clasificadas, la mayoría son clasificadas correctamente. Sin embargo, surge la duda de qué hacer en los casos donde las habilidades del jugador sean muy bajas y pierda antes obtener los datos suficientes para realizar el ajuste.

Clasificación de Medios Para la clasificación de jugadores con habilidad media se utiliza inicialmente la partida completa. La puntuación total de la partida es de 5850, así que este jugador es considerado con habilidad alta entre aquellos de su rango. La partida completa tiene 78 instancias, cada una con 10 movimientos tácticos consecutivos. Es decir, durante toda la partida el jugador ubica 780 piezas en el tablero. El 75.6 % de las instancias de esta partida son correctamente clasificadas con una razón FP (*False Positive*) de 0.09 clasificados como expertos y 0.154 como principiantes. En el gráfico 4.3b se observan las instancias incorrectamente clasificadas como rectángulos.

Uno de los propósitos es poder realizar una predicción a partir de una porción de los datos con el fin de adaptar la sesión activa. Si se ha superado la posición y15, entonces se evalúa la habilidad del jugador. A partir de la instancia 13, más de la mitad de los movimientos tácticos que la componen son inferiores a y15 y se considera que el jugador a superado ese umbral. Las primeras 13 instancias son evaluadas con 62 % de acierto; es decir, 8 instancias



(a) (b) (c)

Figura 4.3: Clase (x) Vs. Clase predicha (y).

han sido asignadas al cluster MEDIO y 5 al cluster de PRINCIPIANTE. Debido a que la cantidad de instancias del cluster MEDIO es mayor, se considera que el jugador tiene una habilidad media y el ajuste se realiza acorte a esta.

Clasificación de Expertos El jugador clasificado como experto en la partida evaluada genera 232 instancias de 10 movimientos tácticos consecutivos y una puntuación de 10690. Este jugador ha producido 2320 movimientos tácticos durante toda su partida. El 82.3 % de las trazas son clasificadas correctamente. Las instancias restantes son clasificadas como medio ($FPRate = 0{,}087$) o principiante ($FPRate = 0{,}091$). En el gráfico 4.3c se observan las instancias incorrectamente clasificadas como rectángulos. De forma similar al jugador promedio, se busca la instancia a partir de la cual se tenga más de la mitad de movimientos tácticos inferiores a y15 y se realiza la clasificación sólo con las trazas previas. El 62.7 % de estas instancias son clasificadas correctamente. Las instancias restantes son clasificadas como medio ($FPRate = 0{,}176$) o principiante ($FPRate = 0{,}196$).

Capítulo 5

Conclusiones y Trabajo Futuro

En este trabajo se investiga la aplicación de técnicas de Aprendizaje Automático para la clasificación y predicción de habilidades de jugadores de Tetris. El objetivo principal de este estudio, es el ajuste de dificultad del juego a partir de la predicción del nivel de habilidad del jugador. El enfoque empleado fue utilizar las trazas generadas de la interacción del usuario con el juego y determinar su nivel de habilidad. Para esto, se realizaron varios experimentos con el fin de determinar el grupo de variables que mejor describe cómo juega un usuario, y la evaluación, modelado y predicción de los tipos de jugadores.

En primera instancia, se determina el conjunto de variables que mejor clasifican a los jugadores basándose en la interacción que estos tienen con el juego. TetrisAnalytics tiene diversas políticas de generación de datos a partir de las trazas de las partidas. Cada política produce determinadas variables que son utilizadas para la clasificación del comportamiento de los jugadores. Para cada una de éstas, se evalúa cuál produce un modelo con mejor porcentaje de clasificación y a partir de este, se describen las características principales de cada uno de los perfiles.

En segundo lugar, el espacio de movimientos posibles en Tetris es restringido, puede ocurrir que varios jugadores con diferentes niveles de habilidad realicen movimientos semejantes; por esto, no es suficiente evaluar los movimientos tácticos individualmente sino que se realiza la unión de movimientos tácticos consecutivos. Esto aumenta la complejidad de las pruebas, ya que es necesario concatenar movimientos tácticos y evaluar su clasificación

hasta encontrar el óptimo.

Finalmente, con el modelo del comportamiento de los jugadores, se predice el nivel de nuevos usuarios mientras la sesión está activa y así, realizar un ajuste de dificultad. En esta etapa final, se desarrollaron experimentos con jugadores de cada uno de los grupos (principiante, medio y experto). Para los niveles medio y experto, se realizó la predicción de las instancias de toda la partida con las que se obtiene un porcentaje alto de clasificación correcta. Sin embargo, el propósito es realizar la predicción mientras la sesión de juego está activa. De manera que se divide el tablero en 4 zonas y en el momento que el jugador sobrepase alguno de estos umbrales, las instancias recopiladas hasta el momento son evaluadas y se toma la clase con mayor ocurrencia. A partir de esta información es posible determinar el ajuste de dificultad acorde a las habilidades del jugador.

La metodología utilizada para la clasificación y predicción del nivel de habilidad es posible aplicarla en otros juegos; sin embargo, este proceso no es automático, y depende mucho de su diseño. Las fases de preparación de datos y modelado son las más extensas y tediosas, puesto que dependiendo del análisis, selección y formato de las variables, los resultados tienden a cambiar. Particularmente en juegos similares a Tetris, dónde el espacio de movimientos posibles es restringido, la clasificación del comportamiento de los jugadores tiende a ser más compleja a causa de la alta probabilidad que existe que dos o más jugadores realicen movimientos similares. En consecuencia, se opta por unir movimientos tácticos consecutivos para disminuir la posibilidad de que esto ocurra. La cantidad de movimientos a unir depende de la población evaluada; si las partidas evaluadas pertenecen a jugadores con alto nivel, entonces tendrán más movimientos en sus partidas y la cantidad utilizada en la construcción de una instancia es mayor; lo que lleva a una alta precisión. Por el contrario, cuando la cantidad de movimientos por partida es poca, o sea el jugador pierde rápidamente, la construcción de una instancia contiene pocos movimientos tácticos lo que supone una exactitud menor. Como trabajo futuro, es necesario encontrar alternativas de predicción de jugadores con bajas habilidades. Además, se implementará el ajuste de dificultad en

TetrisAnalytics y se comparará la experiencia que tiene el usuario con el juego sin el ajuste contra aquel que si lo tiene.

Conclusions and Future Work

In this work, the application of Machine Learning techniques for classification and prediction of skills in video games is under investigation. Its main goal is the difficult adjustment of video games based on level player prediction. The approach used include the gameplay traces from each player and the establishment of its level. For this purpose, several experiments were made to determine the set of features that better describe how a user plays, and the evaluation, classification and prediction of the different types of players in the training set.

First of all, the set of features for the evaluation and classification of player is determined. TetrisAnalytics has several data generation policies. Each one takes the log produced from a session of the game and brings out the respective features. To be able to select the variables that best fit, i.e. the features that produces the highest correctly classified instances percentage, it is necessary to evaluate the outcome features from each policy.

Secondly, the available moves for Tetris players is limited, two player with different levels might have similar tactical movements. Because its not enough to assess tactical moves individually. That's why, joining consecutive tactical movements is a better fit to describe the style of a player. This increases the complexity of the experiments because is necessary to associate n consecutive tactical moves, check how good is the classification results and if is not, assess with $n + 1$ until find the optimum value.

Finally, through the behavioral model, can be predicted the level of new players while the session still active and make the difficult adjustment. In this final stage, several tests were made to verify the correct classification of players with distinct levels (beginner, medium and expert). For expert and medium levels, the predictions were made with all the instances from the session; however, the purpose is to predict the level of a player while he is playing. Therefore, the Tetris board space is divided in 4 zones and if the player crosses one of them,

all the moves made until that moment are used for its classification and the level is set by the higher occurrence of a class, e.g. if a player have 15 moves, and 10 of them are classify like medium, 3 like expert and 2 like beginner, then the player is level is medium. According to the player's level, it is possible to figure out the appropriate difficult adjustment.

The methodology adopted for classification and prediction of the player's level can be applied in other games; however, this process is not automatic and depends on the game design. The data preparation and modeling phases are the most extensive and tedious due to design analysis and feature selection. So it might be necessary to repeat those phases several times until get it right. In games like Tetris, where the space of possible player's moves are restricted, the behavioral classification trends to be more complex because the high probability of two or more players make the same moves, whether they have different levels or not. In consequence, it is best to join consecutive moves to low this probability. The number of moves to join depends of the population that is been assess. If the apprise player has a high level, then the session will contain a lot of moves and more instances can be build joining consecutive actions which results in a higher accuracy. On the other hand, if there are few actions, e.i. the player loses quickly, the instances build has less join consecutive moves which results in low accuracy. For future work is required to consider a better way to predict low level players. Also, the modification of TetrisAnalytics to include the difficult adjustment explain here and compare the user experience.

Bibliografía

[1] E. Adams. Balancing games with positive feedback. Gamasutra, 2002. 38

[2] Weka API. Class classificationviaclustering. http://weka.sourceforge.net/doc.stable/weka/classifiers/meta/ClassificationViaClustering.html. Consultado: 2015-05-17. 65

[3] Weka API. Class em. http://weka.sourceforge.net/doc.dev/weka/clusterers/EM.html. Consultado: 2015-05-17. 53

[4] Weka API. Class simplekmeans. http://weka.sourceforge.net/doc.dev/weka/clusterers/SimpleKMeans.html. Consultado: 2015-05-17. 53

[5] D. Arthur and S. Vassilvitskii. k-means++: the advantages of carefull seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2007. 53

[6] Christian Bauckhage and Christian Thurau. Making archetypal analysis practical. In Joachim Denzler, Gunther Notni, and Herbert Süße, editors, *Pattern Recognition*, volume 5748 of *Lecture Notes in Computer Science*, pages 272–281. Springer Berlin Heidelberg, 2009. 27

[7] K. Kersting C. Thurau and C. Bauckhage. Yes we can: Simplex volume maximization for descriptive web-scale matrix factorization. In Proceedings 19th ICIKM, 2010, 2010. 27

[8] A. Canossa. Meaning in gameplay: Filtering variables, defining metrics, extracting features and creating models for gameplay analysis. In *Game Analytics, Maximizing the Value of Player Data*, pages 255–283. Springer, 2013. 17

[9] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth. Crisp-dm 1.0. http://the-modeling-agency.com/crisp-dm.pdf, Agosto 2000. Consultado: 2015-03-19. 21, 22, 23

[10] D. Charles and M. Black. Dynamic player modelling: A framework for player-centered digital games, 2004. 13

[11] D. Cook. The chemistry of game design. Gamasutra, 2007. Consultado: 2015-05-26. 37

[12] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. 40

[13] A. Cutler and L. Breiman. Archetypal analysis. Technometrics, 36(4), 1994. 3, 9

[14] T.H. Davenport and J.G. Harris. *Competing on Analytics: The New Science of Winning*. General management / Harvard Business School Press. Harvard Business School Press, 2007. 14

[15] A. Drachen. What are game metrics? http://blog.gameanalytics.com/blog/what-are-game-metrics.html, Julio 2012. Consultado: 2015-06-10. 15

[16] A. Drachen. What is game telemetry? http://blog.gameanalytics.com/blog/what-is-game-telemetry.html, Agosto 2012. Consultado: 2015-06-10. 17

[17] A. Drachen. Introducing clustering i: Behavioral profiling for game analytics. http://blog.gameanalytics.com/blog/introducing-clustering-behavioral-profiling-game-analytics.html, Mayo 2014. Consultado: 2015-04-17. 26

[18] A. Drachen. Introducing clustering ii: Clustering algorithms. http://blog.gameanalytics.com/blog/introducing-clustering-ii-clustering-algorithms.html, 2014. Consultado: 2015-04-17. 27

[19] A. Drachen and A. Canossa. *Towards Gameplay Analysis via Gameplay Metrics*, pages 202–209. Association for Computing Machinery, 2009. 14, 20

[20] A. Drachen, A. Canossa, and M. Seif El-Nasr. Intro to user analytics. Gamasutra, 2013. 1, 2, 7, 8, 18, 19

[21] A. Drachen, A. Canossa, and J. Sørensen. Gameplay metrics in game user research: Examples from the trenches. In M. Seif El-Nasr, A. Drachen, and A. Canossa, editors, *Game Analytics, Maximizing the Value of Player Data*, pages 285–319. Springer London, 2013. 1, 7, 13, 14, 15, 17, 20, 21

[22] A. Drachen, A. Canossa, and G. Yannakakis. Player modeling using self-organization in tomb raider: Underworld. In Proceedings of IEEE Computational Intelligence in Games (CIG) 2009 (Milan, Italy), 2009. 3, 9, 25, 31

[23] A. Drachen, M. Seif El-Nasr, and A. Canossa. Game analytics - the basics. In *Game Analytics, Maximizing the Value of Player Data*, pages 13–40. Springer, 2013. 12, 14, 15, 16, 18

[24] A. Drachen, A. Gagné, and M. Seif El-Nasr. Sampling for game user research. In M. Seif El-Nasr, A. Drachen, and A. Canossa, editors, *Game Analytics*, pages 143–167. Springer London, 2013. 21

[25] A. Drachen, R. Sifa, C. Bauckhage, and C. Thurau. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *2012 IEEE Conference on Computational Intelligence and Games, CIG 2012, Granada, Spain, September 11-14, 2012*, pages 163–170, 2012. 3, 9, 28, 32, 33

[26] A. Drachen, C. Thurau, R. Sifa, and C. Bauckhage. A comparison of methods for player clustering via behavioral telemetry. In *Foundations of Digital Games 2013 (Chenia, Greece)*. Society for the Advancement of the Science of Digital Games (SASDG), 2013. 3, 9, 18, 27, 34

[27] A. Drachen, C. Thurau, J. Togelius, G.N. Yannakakis, and C. Bauckhage. Game data mining. In *Game Analytics, Maximizing the Value of Player Data*, pages 205–253. Springer, 2013. 3, 4, 8, 9, 12, 13, 21, 23, 25, 27

[28] T. Fields. Game industry metrics terminology and analytics case study. In M. Seif El-Nasr, A. Drachen, and A. Canossa, editors, *Game Analytics*, pages 53–71. Springer London, 2013. 2, 7, 13, 18

[29] T. Fields and B. Cotton. *Social Game Design: Monetization Methods and Mechanics*. MK Pub., 2011. 38

[30] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins studies in the mathematical sciences. Johns Hopkins University Press, 1996. 27

[31] Gartner group. http://www.gartner.com/it-glossary/data-mining. Consultado: 2015-06-10. 2, 8

[32] J. Han and M. Kamber. *Data Mining: Concepts and techniques*. Morgan Kaufmann, 2006. 27, 40

[33] G. Hasson. Story design tips: Better npc interaction, part i. Gamasutra, Junio 2012. Consultado: 2015-04-15. 3, 9

[34] G. Hasson. Story design tips: Better npc interaction, part ii. Gamasutra, Julio 2012. Consultado: 2015-04-16. 3, 9

[35] R. Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 429–433. ACM, 2005. 38

[36] R. Hunicke, M. LeBlanc, and R. Zubek. Mda: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, volume 4, 2004. 38

[37] L. Kaufman and P. Rousseeuw. *Clustering by Means of Medoids*. Reports of the Faculty of Mathematics and Informatics. Faculty of Mathematics and Informatics, 1987. 27

[38] D. Kennerly. Better game design through data mining. Gamasutra, Agosto 2003. Consultado: 2015-04-15. 1, 3, 7, 9

[39] Jun H. Kim, Daniel V. Gunn, Eric Schuh, Bruce Phillips, Randy J. Pagulayan, and Dennis R. Wixon. Tracking real-time user experience (TRUE): a comprehensive instrumentation solution for complex systems. In *Proceedings of the 2008 Conference on Human Factors in Computing Systems, CHI 2008, 2008, Florence, Italy, April 5-10, 2008*, pages 443–452, 2008. 20

[40] D. King and S. Chen. Metrics for social games. Presentation at the social games summit 2009, game developers conference, 2009. San Francisco, CA. 39

[41] S.P. Lloyd. Least squares quantization in pcm. IEEE Transactions on Information Theory 28(2), 1982. 3, 9, 27

[42] D. Lora. El clustering de jugadores de tetris. II CONGRESO DE LA SOCIEDAD ESPAÑOLA PARA LAS CIENCIAS DEL VIDEOJUEGO, Barcelona., 2015. 57, 59

[43] T. Mahlman, A. Drachen, A. Canossa, J. Togelius, and G.N. Yannakakis. Predicting player behavior in tomb raider: Underworld. In Proceedings of the 2010 IEEE Conference on Computational Intelligence in Games, 2010. 25

[44] L. McWilliams. The metrics aren't the message. Gamasutra, Marzo 2013. Consultado: 2015-04-15. 1, 7

[45] L. Mellon. Applying metrics driven development to mmo costs and risks. *Versant Corporation, Tech. Rep.*, 2009. 12, 18, 21

[46] O. Missura and T. Gärtner. Player modeling for intelligent difficulty adjustment. In

Discovery Science, 12th International Conference, DS 2009, Porto, Portugal, October 3-5, 2009, pages 197–211, 2009. 1, 2, 4, 9, 13, 38, 39, 41, 44, 45, 46, 49

[47] D. Nozhnin. Predicting churn: Data-mining your game. Gamasutra, Mayo 2012. Consultado: 2015-06-03. 3, 9

[48] D. Nozhnin. Predicting churn: When do veterans quit? Gamasutra, Agosto 2012. Consultado: 2015-05-21. 3, 9

[49] Weka The University of Waikato. Weka 3: Data mining software in java. http://www.cs.waikato.ac.nz/ml/weka/. Consultado: 2015-05-17. 51

[50] R.J. Pagulayan, K. Keeker, D. Wixon, Ramon L. Romero, and T. Fuller. User-centered design in games. In Julie A. Jacko and Andrew Sears, editors, *The Human-computer Interaction Handbook*, pages 883–906. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2003. 14, 20

[51] C. Pedersen, J. Togelius, and G.N. Yannakakis. Modeling player experience for content creation. *Computational Intelligence and AI in Games, IEEE Transactions on*, 2(1):54–67, March 2010. 13

[52] R. Rabbany. Comparison of different classification methods. http://webdocs.cs.ualberta.ca/~rabbanyk/research/603/short-paper-rabbany.pdf. Consultado: 2015-03-08. 63

[53] R.M. Rifkin. *Everything Old is new again: A fresh Look at Historical Approaches to Machine Learning*. PhD thesis, PhD Thesis, MIT, 2004. 40

[54] S. Santhosh and M. Vaden. Telemetry and analytics best practices and lessons learned. In M. Seif El-Nasr, A. Drachen, and A. Canossa, editors, *Game Analytics*, pages 85–109. Springer London, 2013. 14, 15

[55] C. Thurau and A. Drachen. Introducing archetypal analysis for player classification in games. In *2011 Foundations of Digital Games Conference (Bordeaux, France)*. ACM, 2011. 27

[56] C. Thurau, T. Paczian, G. Sagerer, and C. Bauckhage. Bayesian imitation learning in game characters. *Int. J. Intell. Syst. Technol. Appl.*, 2(2/3):284–295, February 2007. 13

[57] A. Tychsen. Crafting user experience via game metrics analysis. https://andersdrachen.files.wordpress.com/2014/07/crafting-user-experience-via-game-metrics-analysis.pdf, 2014. Consultado: 2015-06-10. 2, 8, 15, 17

[58] Springer US. Clustering. In Claude Sammut and GeoffreyI. Webb, editors, *Encyclopedia of Machine Learning*, pages 180–180. Springer US, 2010. 26

[59] Springer US. Supervised learning. In C. Sammut and G.I. Webb, editors, *Encyclopedia of Machine Learning*, pages 941–941. Springer US, 2010. 4, 9

[60] Springer US. Unsupervised learning. In C. Sammut and G.I. Webb, editors, *Encyclopedia of Machine Learning*, pages 1009–1009. Springer US, 2010. 26

[61] C. Vercellis. *Business intelligence: Data mining and optimization for decision making*. Chichester: Wiley, 2009. 20

[62] Ian H. Witten, Eibe Frank, and Mark A. Hall. Chapter 10 - introduction to weka. In Ian H. Witten and Eibe Frank Mark A. Hall, editors, *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 403 – 406. Morgan Kaufmann, Boston, third edition edition, 2011. 51

[63] Ian H. Witten, Eibe Frank, and Mark A. Hall. Chapter 2 - input: Concepts, instances, and attributes. In Ian H. Witten and Eibe Frank Mark A. Hall, editors, *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 39 – 60. Morgan Kaufmann, Boston, third edition edition, 2011. 51

[64] Ian H. Witten, Eibe Frank, and Mark A. Hall. Chapter 7 - data transformations. In Ian H. Witten and Eibe Frank Mark A. Hall, editors, *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 305 – 349. Morgan Kaufmann, Boston, third edition edition, 2011. 21

[65] G.N. Yannakakis and J. Hallam. Real-time game adaptation for optimizing player satisfaction. *IEEE Trans. Comput. Intellig. and AI in Games*, 1(2):121–133, 2009. 3, 9, 13, 25, 39