Detecting Cheating In FPS Online Games Using GPT Technologies

Helin Yağmur Çağine[1]*, Göktuğ Arda Gök[1]†, İlbars Efe Korkmaz[1]‡, ChatGPT[2]
[1]Sabancı University, Istanbul, Turkey
[2]OpenAI§
Supervisor: Enes Eryarsoy¶

May 31, 2023

1 Abstract

Outliers in FPS games are often indicative of cheating behaviors, posing a significant challenge to ensuring fair gameplay. This study explores the application of various outlier detection techniques, including Mahalanobis distance, Principal Component Analysis (PCA), K-Nearest Neighbors (KNN), Isolation Forest, and Local Outlier Factor (LOF), to identify potential cheaters in FPS games. A comprehensive analysis is conducted using a dataset of FPS game sessions, and the performance of each technique is evaluated using accuracy, precision, recall, and other relevant metrics. The study highlights the potential of advanced AI technologies, such as GPT, in enhancing outlier detection techniques to improve gameplay and effectively detect cheating in FPS games.

2 Introduction

In recent years, the prevalence of cheating in FPS (first-person shooter) games has become a significant issue with the rise of online gaming. Cheating can negatively impact the gaming experience of honest players and ultimately result in reduced player engagement and revenue for game developers. Therefore, detecting cheaters in FPS online games is crucial for maintaining the integrity of the game and ensuring a fair playing field for all players.

2.1 Why is it important?

- **Maintaining the integrity of the game:** Detecting cheaters in FPS online games is essential to preserve the game's authenticity and fairness, ensuring that all players have an equal opportunity to succeed.
- **Protecting honest players' experience:** Cheating can ruin the gaming experience for legitimate players, causing frustration and dissatisfaction, which may lead to disengagement from the game.
- **Safeguarding developers' revenue:** Reduced player engagement due to cheating can lead to decreased revenue for game developers, as players may stop purchasing in-game items or quit playing altogether.

2.2 How is it novel?

- **Addressing a growing problem:** With the increasing popularity of online gaming, especially in the FPS genre, cheating has become a more widespread issue. Developing novel methods to detect cheaters helps combat this growing problem.

*helin.cagine@sabanciuniv.edu
†goktug.gok@sabanciuniv.edu
‡ilbars.korkmaz@sabanciuniv.edu
§ChatGPT (Version 3.5) [Software]. Retrieved from `https://chat.openai.com/`
¶enes.eryarsoy@sabanciuniv.edu

- **Utilizing advanced technology:** Detection techniques can leverage cutting-edge technology, such as artificial intelligence and machine learning, to identify cheaters more effectively than traditional methods.

2.3 Research Question

What are the most effective methods and technologies for detecting and preventing cheating in FPS online games to maintain the integrity of the game, and protect honesty?

3 Literature Review

3.1 Why Does Cheating Happen in Online FPS Games?

3.1.1 Psychological Factors

Studies show that several variables, including the games' competitive character, players' goals, and the availability of cheating tools, contribute to the prevalence of cheating in online first- person shooter games. Some players may be tempted to cheat in first-person shooter games because of the intense competition inherent in the genre, as seen by the prevalence of individual and team rankings and awards for excellent performance (Kow and Young, 2013)[1]. The need to prove oneself is a strong motive for dishonest actions (Yee, 2006)[2].

A research from Sharma et al. (2021)[3] investigated the unethical behavioral intention of online gamers. The Theory of Planned Behavior and Social Comparison Theory is used to develop a comprehensive framework to understand gamers' cheating intention The objective of this research is to examine the intention of online gamers to engage in unethical behavior. To comprehensively understand the intention of gamers to cheat, the study makes use of the theory of "Planned Behavior and the Social Comparison Theory." The study employs a quantitative approach and randomly selects 404 gamers to gather data. The results indicate that various factors such as attitude, subjective norms, perceived behavioral control, hedonic motivation, and benign envy affect the intention of gamers to cheat online.

Additionally, it was found that a positive relationship exists between online cheating intention and online cheating behavior. The research also looked at the role of ethical judgment as a moderator and found that it reduces the positive effect of attitude, benign envy, and perceived behavioral control on gamers' online cheating intention.

According to Kow and Young, due to the lack of personal interaction inherent in online gaming, gamers may feel more comfortable cheating there than they would in a face-to-face setting (Kow and Young, 2013)[1]. Especially in first-person shooter games, where players frequently collaborate with total strangers under intense time constraints, this can be the case. (Taylor, 2006)[4] The researcher's result here is that there is no single cause for the widespread problem of cheating in online first-person shooter games; rather, it is a result of many interrelated elements, including as the games' inherent competitiveness, players' goals, the availability of cheating tools, and the anonymity afforded by online gaming. An all-encompassing solution that takes into account these elements is needed to combat cheating in first-person shooter games online and ensure that all players may have a positive and engaging experience.

3.1.2 Technological Advancements

According to the studies, because of technological advancements, the number of online gaming sites has increased dramatically, and a vibrant, competitive gaming culture has emerged. The proliferation of internet gaming has brought with it a host of new problems, cheating being only one of them. The increasing prevalence of cheating in online games poses a serious danger to the honesty and fun of the gaming experience for everyone involved. (Young, 2009)[5] To that end, this literature review will cover definitions, prevalence, types, motivations, repercussions, and preventative techniques related to online gaming cheating. To cheat in an online game is to act in a way that goes against the established norms and standards for fair play in that particular game (Consalvo, 2009)[6]. To acquire an unfair edge, players may resort to a variety of methods, including exploiting software defects, installing third-party software, or even conspiring with other players (Mitterhofer Zimmermann, 2019)[7]. Cheating not only upsets the competitive balance of games but also significantly influences the experiences of legitimate players, potentially leading to disengagement, discontent, and even desertion of the platform (Yee, 2006)[2].

3.1.3 Popularity of the Gaming Business

Experts suggest that as the gaming business has increased, so has the prevalence of cheating in online games. Several studies (Kücklich, 2005[8]; Consalvo, 2009[6]), among others, have found that a sizable percentage of online gamers have engaged in some type of cheating conduct. The growing popularity of competitive esports where substantial financial and social rewards are at risk has also contributed to the cheating epidemic (Taylor, 2012; Hamari Sjöblom, 2017)[9]. Exploiting game mechanics (Kücklich, 2005)[8], using unauthorized software or "hacks" (Mitterhofer Zimmermann, 2019)[7], account sharing (also known as "smurfing") and player collusion (Yee, 2006)[2] are just a few of the many forms of cheating in online games that have been identified in the literature. Players' experiences and the general quality of the game might be negatively impacted by these many forms of cheating (Consalvo, 2009)[6].

Yee (2006)[2], Kow and Young (2013)[1], and Mitterhofer and Zimmermann (2019)[7] all point to a mix of psychological, social, and economic variables as explanations for why people cheat on video games. For example, some players cheat to attain a sense of mastery or domination over others, while others are driven by a desire for social recognition or to alleviate boredom (Yee, 2006[2]; Kow Young, 2013[1]). Players aren't the only ones affected by cheating in video games; programmers and the gaming industry as a whole feel the repercussions as well (Hamari Sjöblom, 2017)[9]. The integrity of a gaming community depends on its members being able to trust one another and play fairly (Consalvo, 2009)[6]. In addition, game makers can lose a lot of money due to cheating. Therefore they have to put in a lot of effort to prevent it and keep their games fair (Mitterhofer Zimmermann, 2019)[7].

Various methods have been proposed to combat cheating in online games, including the use of anti-cheat software (Mitterhofer Zimmermann, 2019)[7], the imposition of penalties and social sanctions (Consalvo, 2009[6]; Kow Young, 2013[1]), and the creation of games with anti-cheat mechanics and structures (Kücklich, 2005)[8]. Yee (2006)[2] and Consalvo (2009)[6] stress the necessity of building a strong gaming community with shared values to deter cheating. There is no silver bullet for preventing cheating in online games, but these methods, when used together, can help create a more level playing field for everyone.

The gaming business, game creators, and gamers all stand to lose in the long run if cheating in online games is not addressed. (Blackburn, 2011)[10] The definitions, prevalence, types, motivations, repercussions, and prevention techniques for cheating in online games have been discussed in this research review. Addressing this issue demands a comprehensive and collaborative strategy, incorporating both technical and social solutions, to ensure that the gaming community remains a fair and entertaining area for all its players. (Lu Wang, 2008) [11]

3.2 Types of Cheating

Scholars and researchers have classified many forms of cheating in online gaming, with considerable disagreement over how many categories exist. This essay is to provide an overview of the various forms of cheating mentioned in the literature, with a particular focus on Kücklich (2005)[8] and Mitterhofer and Zimmermann's (2019) [7]analyses of this phenomenon. Cheating in digital games can be broken down into three broad categories, as proposed by Kücklich (2005)[8]: exploiting game mechanics, using unlicensed software, and social engineering.

3.2.1 Exploiting Game Mechanics

To acquire an unfair edge in a game by the manipulation of its rules and processes is known as "exploiting game mechanics." Exploiting defects, discovering design flaws, and taking unintentional shortcuts are all examples of such strategies. The use of "hacks" or "cheats," which refer to the use of unofficial software to gain an unfair advantage in a game, is a common kind of cheating.

3.2.2 Using Unlicensed Softwares

Using unlicensed software refers to the act of utilizing software without proper authorization or a valid license from the software's copyright owner or developer. This typically involves installing, copying, or distributing software in violation of the terms and conditions outlined by the software's license agreement.

Unlicensed software can take various forms, including proprietary software, open-source software, or freeware. In each case, the unauthorized use of the software violates the rights of the copyright holder and may constitute copyright infringement.

3.2.3 Social Engineeering

Last but not least, social engineering refers to the practice of influencing other players or the game's administrators for personal gain. Methods include cheating in the form of scamming, posing as another player, or working together with others to get an unfair edge.

3.2.4 A More Comprehensive Classification of types of cheating

Mitterhofer and Zimmermann (2019)[7] presents a comprehensive classification of cheating in digital games, distinguishing seven distinct categories: (1) exploiting game mechanics; (2) using unauthorized software; (3) account-related cheating; (4) player collusion; (5) hardware cheating; (6) social engineering; and (7) exploiting game content. Mitterhofer and Zimmermann (2019)[7] expand upon Kücklich's (2005)[8] categorization by recognizing further forms of cheating. Some forms of account-related cheating include "smurfing," in which a talented player creates a new account to compete against less skilled opponents, and "account sharing," in which a more experienced player plays on behalf of a less skilled player. The term "player collusion" is used to describe situations in which two or more players conspire to acquire an unfair advantage in a game. To get an unfair edge, some players use hardware that has been tampered with or is not officially sanctioned by the game's developers. Last but not least, data mining and reverse engineering game files are examples of leveraging game content in order to obtain previously unavailable information or content.

While the number of cheating categories identified by Kücklich (2005)[8] and Mitterhofer and Zimmermann (2019)[7] differ, both publications help us better grasp this nuanced issue. Researchers can learn more about the causes and effects of cheating behavior in online games by categorizing the various forms of dishonesty that occur there.

3.3 Harms of Cheating in FPS Games

Cheating in online games can have several negative consequences, including the de-socialization of gamers, adversely affecting game brands, and monetary losses due to cybercrime. Cheating in online games is a significant problem that affects the gaming industry and players.

Cheating can take many forms, including the use of bots, exploiting bugs, and hacking. Cheating can have several negative consequences, including the de-socialization of gamers, adversely affecting game brands, and monetary losses due to cybercrime (Lee et al., 2021)[12]. Cheating can also lead to a decrease in player satisfaction and loyalty (Lu Wang, 2008)[11]. Passmore et al. (2020)[13] found that cheating behavior can negatively affect game balance, player satisfaction, and enjoyment. Cheating can also lead to feelings of guilt, shame, or frustration among players who do not cheat (Passmore et al., 2020)[13]

3.4 Cheating Detection Methods

As cheating increases in online games, cheater detection is crucial. Using machine learning and AI to detect and prevent cheating is becoming more popular. Machine learning and AI improve cheater detection methods, as this review shows. Here are some of the studies that are focused on this topic.

Jonnalagadda et al. (2021)[14] use computer vision technologies to detect any cheating activities in the famous FPS game Counter-Strike by studying the advantages and disadvantages of different Deep Neural Network architectures operating on a local or global scale.

Alayed, Frangoudes, Neuman (2013)[15] used the method of processing the big data and provided a server-side anti-cheating solution that uses only game logs. Their method is based on defining an honest player.

Yeung, Yeung, Lui (2008)[16] proposed a dynamic Bayesian approach, a machine learning method, for detecting cheats in multiplayer online games. They also proposed some optimizations to overcome the synchronization problems, and they proposed the use of CAPTCHA tests to ensure the participation of human players in online games. This approach is very effective against the use of fully automated robots. Alayed et al. use the Bayesian approach to detect any suspicious activity as well. (Alayed et al., 2013)[15]

Anti-cheat software examines game data and behavior to detect cheating (Mitterhofer Zimmermann, 2019)[7]. Signature-based detection which relies on the identification of known patterns, or signatures, of malicious code, and heuristic-based detection, which involves analyzing the behavior and characteristics of software to identify potential threats, even if their signatures are not yet known are used in these systems (Mitterhofer Zimmermann, 2019)[7]. These approaches can detect existing tricks, but they may not detect new ones. Machine learning and AI can improve cheating detection systems. Machine learning algorithms can predict cheating patterns better than signature- or heuristic-based detection approaches (Yang et al., 2014)[17]. Using historical data, supervised learning may train classification models to detect cheating from non-cheating behavior (Yang et al., 2014)[17]. These models can scan real-time game data to identify potential cheating for further investigation.

Deep learning, a machine learning area, has been used to detect online game cheating (Kim et al., 2018)[18]. Deep learning algorithms like CNNs and RNNs can automatically recognize complex patterns and correlations in data, making them ideal for detecting subtle or hard-to-detect cheating, and CNNs can identify unusual aiming patterns in the provided game's photos and given conditions to detect aimbot use (Kim et al., 2018)[18].

The research of Chapel et al.[19] describes two techniques for detecting cheating solely based on game results. The techniques are built upon the law of large numbers and the Bradley-Terry model and aim to identify statistical patterns that indicate cheating behavior by players. Initial simulation studies indicate that these methods hold great promise in identifying cheaters. (Chapel et al., 2010)[19]

The study of Chen et al.[20] proposes a trajectory-based approach for detecting game bots that can be applied to any game in which the players directly control the avatar's movement. By using real-life data traces, the study demonstrates that human players' trajectories differ significantly from those of game bots. Although game bots may try to simulate human decision-making, certain behavior patterns are challenging to replicate because they are AI-hard. The study evaluates the approach's performance and the results demonstrate that the approach can achieve a detection accuracy of 95% or higher with a trace of 200 seconds or longer. (Chen et al., 2018)[20]

AI-designed game mechanisms that prohibit cheating are another developing study field (Cook Smith, 2017)[21]. Developers can make games that react to players' actions by using AI-generated content and dynamic game environments. This strategy can complement other cheater detection approaches to address online game cheating more comprehensively.

Finally, machine learning and AI can improve online game cheating detection. Game developers and researchers can create more accurate, efficient, and flexible cheating detection systems using this modern technology. Machine learning algorithms, deep learning models, unsupervised learning methods, and AI-powered player behavior models help us comprehend cheating and build new measures to stop it. AI's ability to influence the game design and generate adaptive game environments might also dissuade cheating by making game mechanics harder to exploit. The gaming industry may improve player safety and enjoyment by combining new cheating detection tools with older ones.

3.5 Strategies to Prevent Cheating

The extent of literature on strategies to prevent cheating in online gaming may be broken down into two main groups: technical and social. Anti-cheat software, safe game architecture, and server-side checks are at the heart of technical solutions to the problem of cheating in video games (Mitterhofer Zimmermann, 2019)[7]. Although these methods can be useful, they are not foolproof because cheaters can always find a way around or counteract them (Mitterhofer Zimmermann, 2019)[7]. The use of machine learning and AI to combat cheating is also largely uncharted, despite the fact that these technologies have shown promise in improving cheating detection capabilities.

On the other side, social solutions place emphasis on building a solid gaming community with common values that deter cheating (Yee, 2006 [2]; Consalvo, 2009[6]). Games can be designed to encourage positive player behavior, reporting and moderation tools can be implemented at the community level, and players can be encouraged to create and abide by a code of conduct (Yee, 2006[2]; Consalvo, 2009[6]). Options to address cheating include calling out and blacklisting players, in-game punishments such as fines, temporary progress limits, or starting over as a new character.

Fehr and Gächter (2002)[22]demonstrated that allowing group punishment can integrate cheating into a game, preventing its destruction and creating educational opportunities for dealing with transgressors.

Further study is required to understand the effects of social solutions on cheating behavior, however, the literature on their efficacy is growing.

3.6 The Gap in the Current Literature About Cheating in Online Games

Several possible lines of inquiry can be recognized in light of the gaps in the current literature. To begin, there has to be more research into how machine learning and AI may be used to detect and prevent cheating in digital games. This could involve employing machine learning models to anticipate and preemptively address potential cheating behavior, or it could involve investigating AI-driven game design strategies that intrinsically prohibit cheating.

One gap in the current literature is the lack of standardized datasets for training and testing detection algorithms. Most studies have used small, proprietary datasets or relied on manually labeled data, which may not be representative of real-world cheating behavior. Additionally, few studies have evaluated the performance of detection algorithms in dynamic and unpredictable gaming environments where cheating behavior may vary depending on the situation.

Chambers et al.(2005)[23] suggested a method called 'bitcommitment,' which involves hashing sensitive data and a secret together, to determine if the opponent cheated by verifying the final known positions of the objects with the initial hash after the game. However, this approach has a drawback as the cheating verdict can only be determined at the end of the game, and it is not clear if this solution is scalable.

For instance, Kanervisto et al. (2022)[24] presented a method for using GANs to create aimbot cheating software for online FPS games. However, their study did not focus on detecting cheating bots, but rather on using machine learning to develop these bots. Similarly, other studies have used supervised or unsupervised learning algorithms to detect bots, but these methods do not utilize generative models. This highlights a significant research gap in the detection of cheating bots using generative models. While generative models have been successful in numerous applications, such as image and text generation, there is a lack of research on their effectiveness in detecting cheating bots in FPS online games.

Also, further research is needed to determine if social solutions are beneficial in reducing cheating. For example, we could look at how reporting systems, moderation, and social pressure are used in different gaming communities to combat cheating. Research could also look into how incentives, sanctions, and social norms are implemented within games to encourage desirable player conduct and discourage cheating.

In addition, a more comprehensive strategy is required to combat cheating, one that acknowledges the need to integrate both technical and social approaches. The best way to ensure a safe and fun gaming experience for everyone is to look into how technical and social solutions for preventing cheating may work together. Castronova and Falk (2009)[25] suggest that technical and social approaches should be integrated to ensure a safe and fun gaming experience for everyone.(Castronova, E., Falk, M., 2009)[25]. Johnson and Johnson (2018)[26] apply an actor-network perspective to understand cheating in games and explore how the concept is rhetorically effective in sociotechnical controversies. (Johnson, M. R., 2018)[26]

Another gap in the research on detecting cheating in online FPS games is the lack of consideration for the human factor in cheating behavior. Automated detection methods alone may not capture the nuances of cheaters' motivations, personality traits, and social factors. To address this gap, researchers have proposed incorporating social and psychological factors into the detection process to improve accuracy and effectiveness. Cultural and demographic factors may have an effect on cheating behavior and might be taken into account in future studies. Examining the efficacy of culturally sensitive prevention strategies adapted to specific gaming communities is one possible avenue for illuminating regional differences in cheating frequency, motivations, and attitudes. For instance, Alayed et al. (2013)[15] have proposed a set of behavioral features that are indicative of cheating and use them to train a machine learning classifier. Their approach effectively detects cheating behavior in online FPS games with high accuracy.

Laurens et al. (2007)[27] analyze gameplay data and identify anomalous behavior that may indicate cheating. Their approach was tested on a dataset and effectively-identified cheating behavior with a high degree of accuracy. However, it should be noted that these studies focused on a particular type of cheating, and further research is needed to explore the detection of other types of cheating behavior.

There is a knowledge vacuum when it comes to preventing online gaming cheating, despite the fact that the existing literature has supplied significant insights. Future research can contribute to a more in-depth understanding of cheating prevention strategies and help ensure a fair and enjoyable gaming experience for all players by exploring the potential of machine learning and AI in cheating prevention, investigating the effectiveness of social solutions, adopting a holistic approach to cheating prevention, and considering cultural and demographic factors.

4 Dataset Analyses

4.1 Methodology

This portion of the thesis will be devoted to the development of a Machine Learning model using data accessible to a single individual, with the goal of potentially detecting fraudsters with far fewer data and resources than the sector's companies. This section will initially consist of the methodology associated with data collection. Following this, we will examine in depth the models created during the experimentation phase and then disclose the outcomes of these models. We will conclude with a discussion of these findings and possible future research directions.

4.1.1 Barriers Encountered in Data Extraction

First, in order to better comprehend the methodology and models employed, it is necessary to comprehend the various obstacles encountered by a person desiring to apply Machine Learning to video game data. This model was created in accordance with the numerous aforementioned articles. Using a Machine Learning model, can we successfully detect fraudsters, or at least potential cheaters? In order to construct a relevant and accurate model, it was essential to collect a certain quantity of exploitable data on players, whether they are cheaters or not.

It was very difficult to locate usable data for this thesis because we were not employed by a video game company in a position that would have allowed us access to large databases containing statistics on players and cheaters. It was impossible to obtain data specifically about cheaters, that is, data that had been preclassified as cheaters, because it is confidential data that would enable cheaters or even competing companies to reverse-engineer their method for detecting cheaters, especially with the help of Machine Learning.

On the other hand, it is possible to obtain data on the statistics of participants in general: some games allow data extraction via API. Since we could not have a dataset with a cheater or non-cheater classification, we had to collect information about the game statistics of a large number of players, regardless of whether these players had been previously classified. However, these APIs are limited and do not permit an unidentified user to retrieve data on thousands of participants, making data collection extremely challenging and without apparent solutions. Therefore, it was necessary to innovate and discover an alternative method for locating random participant data.

There are, however, sites outside of the publisher that list a large number of player statistics via an API that they are permitted to use despite the high volume of data requests. This is the case with the website tracker.gg (Tracker Network, 2021), which collects game statistics from more than 200 million users worldwide, focusing on ten very popular online games in particular. Any player whose game statistics are stored on this website can access them at any time and compare himself to other players whose names or IDs he knows.

The purpose of the website is to promote the sharing of player information, which is why they provide an API so that anyone with the player IDs can retrieve their game data. Consequently, a portion of the issue has been resolved: to extract game data from participants, we can create a login on their developer platform and use their API with the Python requests library.

To extract player statistics via the tracker.gg API, we require the Steam, PlayStation and Xbox logins of the participants. Searching for the information of thousands of random players, we must locate thousands of player IDs and players who are playing the video game whose statistics we seek. Due to confidentiality concerns, such databases do not exist publicly on the Internet.

While browsing the website, we observed that in addition to providing information on any player, it also created general rankings based on players' statistics. Everyone could see which participants in the world had the highest elimination or accuracy rates in any game. In the URL of each PC gamer's profile is their Steam ID, which is another essential piece of information. This means that we can recover the Steam IDs of thousands of players by scraping a large number of pages from a general classification that we have previously chosen.

4.1.2 Gathering Steam User IDs via Data Scraping

Using primarily the Python libraries BeautifulSoup and Request, we extracted the Steam IDs from the first 500 pages of a CS:GO classification listing the number of ties. The type of ranking was selected to provide Steam ID with the least amount of bias possible. If we had done the same thing with the ranking of the number of eliminations or wins, for instance, we would have completely skewed data toward one category of player.

Either by selecting ID at the beginning of the ranking, we end up with only very good players and a significant proportion of cheaters, or by selecting ID at the end, we end up with only very poor players and few cheaters. To ensure a random draw between cheaters, excellent, average, and poor players, a hybrid option was considerably more complicated. The solution was to use a ranking with relatively little bias: the number of ties seemed to be the best option, as this number dropped to zero on the second page of the ranking, after which the ranking was determined at random. Due to the possibility of a large number of scraping errors and the uncertainty as to whether or not all the IDs would be usable, we decided to scrape 500 pages to ensure that we had sufficient participant data in the end.

We opted for a CS:GO classification because it offered the most player statistics and was compatible with the website's API platform. It is also among the most popular and competitive first-person shooters in the globe. The fact that each Steam ID begins with the number "7656" allowed us to extract only the links containing Steam IDs from the collected links. We were able to compile a list containing all the scraped IDs by processing these connections so that only the relevant number could be extracted. The entire code written for this thesis is available in Appendix 1's.

4.1.3 Collecting CS:GO Player Data through an API

After retrieving an unbiased list of Steam IDs of CS:GO participants, it was necessary to retrieve these players' information. Therefore, we created an account on the website's API platform and retrieved the request's structure using the requests library. This allowed us to determine what type of data was available, how to retrieve it, and, most importantly, the structure of the Data frame that would receive all of the stored ID's data. There were then a total of 28 primary variables available to each participant, including the number of eliminations, deaths, and headshots, among others. The comprehensive inventory of imported variables is available in Appendix 3.

This implementation resulted in a massive import of user data via Steam ID and the tracker.gg API. The game statistics of 7263 participants could then be retrieved and stored in the previously created dataframe, providing a dataset rich and diverse enough for modeling. Prior to that, it seems essential to investigate and make this dataset usable. While the vast majority of the data consisted of numbers, decimals, and percentages, they were neither floats nor integers. We were able to convert all the variables into floats by replacing the commas with dots and removing the percentage symbols.

User ID and Time Played have been excluded from this standardization and have not been incorporated into the dataset. These are two values that cannot be used in our modeling: the first is random, and the second is deceptive, given that accounts can be recreated indefinitely and account age is not inherently relevant to our analysis.

Finally, the dataset containing a total of 26 columns and, 7263 rows was saved in CSV format. This allowed us to store the data and process it without having to redo the entire data extraction and risk being obstructed by the API or encountering another error. The used dataset is also available on Github via the link provided in Appendix 2.

Having described the data collection methodology, we can now move on to the experimentation phase and construct clustering models based on the collected dataset to detect potential cheaters.

4.2 Execution

4.2.1 Selection Of The ML Model

In our study, the choice to utilize unsupervised machine learning models for outlier detection was largely driven by the nature of the task and the structure of the data. In the gaming context, it's often challenging to label data with certainty, especially when it comes to determining cheaters. Unsupervised learning models excel in such situations where labeled data is either scarce or unreliable. They rely on the inherent patterns and structures in the data, and can identify anomalies or outliers based on these patterns. Moreover, unsupervised methods can handle high-dimensional, complex datasets, such as ours, effectively. These models don't require prior knowledge or assumptions about the presence or characteristics of potential cheaters. Instead, they objectively analyze the patterns in the player data and highlight those who deviate significantly from the norm. Consequently, this makes unsupervised learning models ideally suited for tasks like ours, where the primary objective is to detect outliers that may indicate anomalous, and potentially fraudulent, behavior.

4.2.2 First Model: Mahalanobis

The analytical process initiated with the formulation of a correlation matrix, a principal approach for apprehending the foundational structure of the dataset. This matrix constitutes a visualization of linear interrelationships among the diverse features within the game, enabling a granular comprehension of how distinct variables might influence others. This step is particularly instrumental in guiding the succeeding utilization of the Mahalanobis distance.

The Mahalanobis distance, a multivariate distance metric, is then computed for each data point in the dataset. Unlike the Euclidean distance, which considers each dimension independently, the Mahalanobis distance factors in the variance of each feature and the covariance between features. This is crucial because it ensures that variables with high correlation do not disproportionally dominate the distance calculation. Outliers are then pinpointed by considering the observations with the highest Mahalanobis distances. The underlying rationale is that an observation is more probable to be an outlier if it is significantly distanced from the distribution of 'normal' observations.

To visualize the distribution of Mahalanobis distances, we constructed a histogram using the top 100 distance values as shown in Figure 1. This allowed us to gain insights into the spread and patterns within this subset of distances. The histogram displayed the frequency or count of distances falling into 20 equally sized bins, with the x-axis representing the Mahalanobis distances and the y-axis representing the frequency.

In order to set a threshold for outlier identification, we selected a value of 900 as our threshold. This threshold represents a cutoff point beyond which distances are considered potential outliers. By comparing each distance value with the threshold, we created a boolean mask that identified outlier rows. A True value in the mask indicated that the corresponding data point was classified as an outlier, while a False value indicated that it was within the acceptable range.

We then counted the number of outliers by summing the True values in the mask. This provided us with a quantitative measure of the extent of deviation from the majority of the data. Finally, we printed the number of outliers based on the Mahalanobis distance calculation.

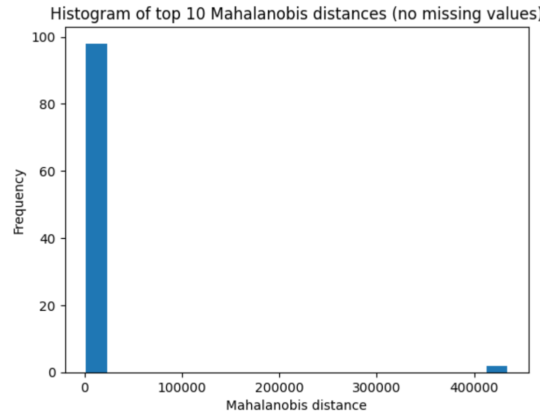


Figure 1:

4.2.3 Second Model: Principal Component Analysis

Principal Component Analysis (PCA), a technique renowned for its proficiency in reducing data dimensionality, is applied. PCA operates by transforming the original variables into a new coordinate system of principal components, which are uncorrelated and account for the maximum possible variance in the data set (see Figure 2). Observations lying considerably distant from others in the PCA score plot, especially along the axes of the first few principal components, are considered outliers.

Following PCA, a density plot (Figure 3) is utilized to evaluate the frequency distribution of outlier scores. Observations residing in the low-density regions of the plot are more likely to be outliers. A threshold is set to discern outliers, ascertained by examining the density plot and selecting an optimal cut-off point.

4.2.4 Third Model: K-Nearest Neighbors (KNN)

The next step is the deployment of the K-Nearest Neighbors (KNN) algorithm. KNN measures the distance between a specific observation and all other observations and then assigns it to the class of its 'k' nearest neighbors. For outlier detection, we look

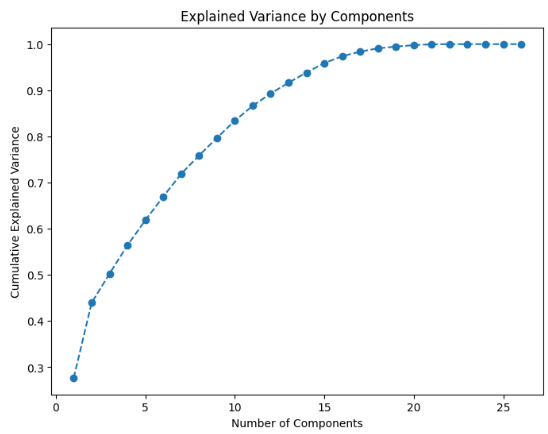


Figure 2:

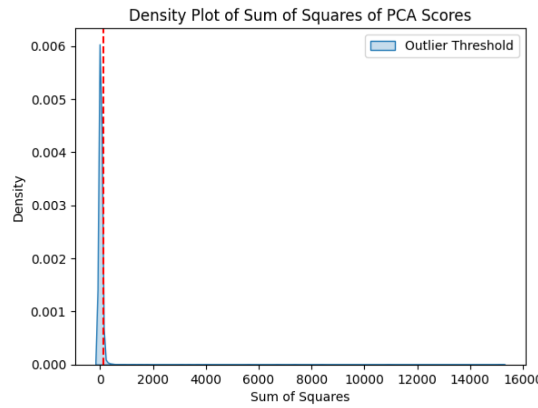


Figure 3:

for observations that deviate significantly from their neighbors. A threshold is established by examining a plot of the KNN distances, beyond which observations are labeled as outliers.

To visualize the distribution, we created a histogram (Figure 4) using the Python programming language and the popular data visualization library, Matplotlib. The histogram was constructed with 20 bins, which partitioned the range of kNN distances into equal intervals. The x-axis represented the distance values, while the y-axis represented the frequency or count of data points falling into each bin.

Additionally, we introduced a vertical red dashed line in the plot to indicate a threshold for identifying potential outliers in the dataset. The threshold was determined by calculating the 99th percentile of the kNN distances using the quantile function. This percentile value serves as a reference point beyond which data points may be considered as outliers, based on their larger distance values.

The resulting histogram provides a visual representation of the distribution of kNN distances and allows us to analyze the spread, skewness, and presence of any potential outliers in the dataset. By understanding the distribution of these distances, we can gain insights into the clustering, separability, or density characteristics of the data, which can further inform decision-making in various domains such as anomaly detection, pattern recognition, or data clustering.

4.2.5 Fourth Model: Local Outlier Factor (LOF)

The process is further amplified by employing the Local Outlier Factor (LOF). LOF estimates the local density deviation of a particular observation with respect to its neighbors. Observations with a substantial density deviation are deemed outliers. A threshold value is determined by plotting the LOF scores, and the observations exceeding this value are identified as outliers.

First, we initialized the LOF model, specifying the number of neighbors to consider as 5 and setting the contamination parameter to 'auto'. The 'auto' setting estimates the proportion of outliers in the dataset automatically. The Euclidean distance metric was chosen to measure the proximity between data points.

Next, we fit the LOF model to the scaled dataset, and predicted the outlier scores for each data point. The scores were then negated to have higher values correspond to outliers.

To identify the top outliers, we created a DataFrame, outliers, which contains the row numbers and corresponding outlier scores. Sorting the DataFrame by the outlier scores in descending order,

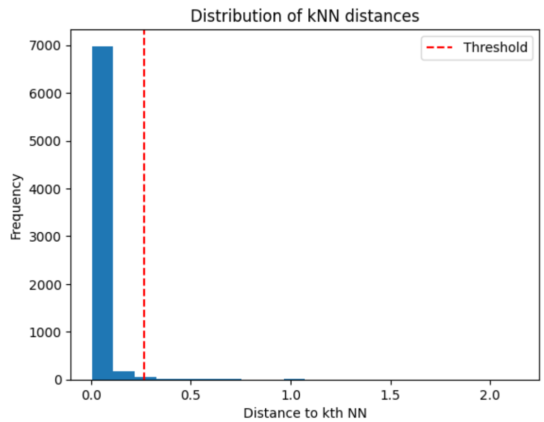


Figure 4:

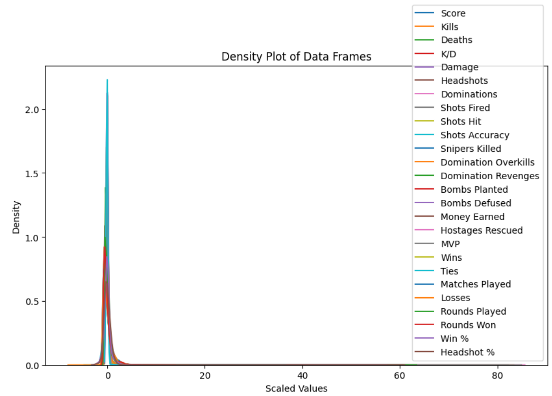


Figure 5:

we obtained a ranked list of the most anomalous data points.

We then printed the top 10 outliers, providing the row numbers and their corresponding scores, for further investigation and analysis.

Additionally, we visualized the density distribution of the dataset using a kernel density estimate (KDE) plot (Figure 5). Each feature's density distribution was plotted individually using the kdeplot() function from the Seaborn library. This allowed us to examine the shape and spread of the scaled values in the dataset, facilitating a better understanding of the overall data distribution and the potential presence of outliers.

By combining the LOF algorithm for outlier detection with the density plot, we gained insights into the anomalous data points and their scores, as well as an overview of the dataset's distribution. This analysis aids in identifying potential outliers and understanding their impact on the overall dataset.

4.2.6 Fifth Model: Isolation Forest (ISO)

A similar methodology is undertaken with the Isolation Forest (ISO) technique. ISO distinguishes anomalies by randomly selecting a feature and then randomly choosing a split value between the maximum and minimum values of the selected feature. It operates on the premise that anomalies are 'few and different', which indicates that they can be isolated more easily.

To begin, we converted the column names of the dataset df to strings to ensure consistent data handling. This step is useful when working with column names that are not already in string format.

Next, we initialized an Isolation Forest model by instantiating the isolation forest class. By default, this initializes the model with 100 base estimators and uses the auto contamination setting, which estimates the proportion of outliers in the dataset.

We then fitted the Isolation Forest model to the dataset, allowing it to learn the underlying patterns and structure of the data. After fitting, we obtained the outlier scores for each data point using the decision function() method.

To define a threshold for outlier identification, we set a value of -0.09 as an example in this case. This threshold represents the cutoff point below which data points are considered outliers based on their outlier scores.

Using the defined threshold, we selected the outliers by filtering the dataset (df) based on the outlier scores. We stored the outlier data points in the outliers variable.

To visually inspect the outlier scores, we created a scatter plot using plt. scatter(). The x-axis represents the samples or data points, while the y-axis represents the outlier scores. Outliers are indicated

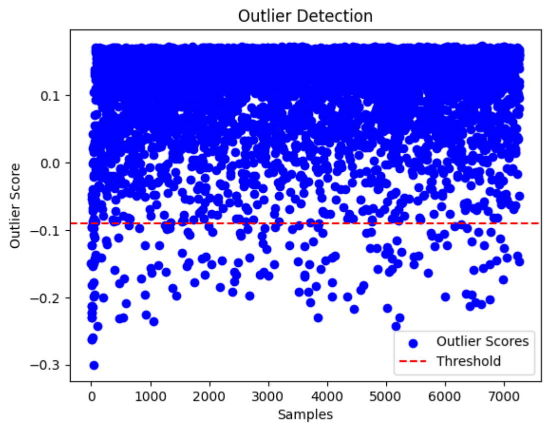


Figure 6:

by higher scores, and they are plotted in blue. Additionally, we drew a horizontal line at the defined threshold value using plt. lashline () to provide a visual reference for distinguishing outliers from non-outliers. The line is drawn in red and is displayed as a dashed line.

To complete the plot, we added labels to the x-axis and y-axis and set a title to describe the purpose of the plot. Furthermore, we added a legend to the plot to clarify the color coding for outliers and the threshold line.

Finally, we displayed the plot using plt. show(), allowing us to visualize the outlier scores and their relationship to the defined threshold.(Figure 6)

This analysis using the Isolation Forest algorithm and the visualization of outlier scores aids in identifying and understanding potential outliers in the dataset.

4.2.7 Decision: Majority Voting

Following the computation of outlier scores via each method, a majority voting scheme is adopted to determine the final set of outliers. Each method 'votes' on whether each observation is an outlier, and observations flagged as outliers by the majority of the methods are deemed to be the true outliers. This approach not only mitigates the potential limitations and biases of individual methods but also bolsters the reliability and robustness of the final set of outlier detections. Based on our calculations, we provided an outlier table.(Table 1)

Row Number	Is Outlier
1	True
3	True
11	True
13	True
26	True
30	True
36	True
39	True
40	True
44	True
70	True
1133	True
2396	True

Table 1: Outlier Rows

4.2.8 ChatGPT

A significant part of the analytical process was carried out utilizing ChatGPT, an AI-powered chatbot developed by OpenAI. Through specific question prompting, we were able to guide the chatbot to provide optimal solutions and data analysis. This amalgamation of human direction and AI capabilities facilitated a more efficient and nuanced approach to data processing and analysis.

5 Conclusion

In conclusion, the multi-method approach led to the identification of a distinct set of outliers, believed to be potential cheaters within the game. The study's methodology, which involved a combination of five different machine learning techniques and the use of a majority voting scheme, resulted in a comprehensive and reliable identification of outliers. This research study not only aimed to identify outliers within a dataset but also showcased the versatility and power of AI-powered tools such as the ChatGPT in executing complex data analysis tasks. Our study is a testament to the significant potential of integrating machine learning techniques and AI-driven tools in conducting thorough data investigations and ensuring fair gaming environments.

References

[1] Kow, Y. M., & Young, A. L. (2013). Cheating in online games: A social network perspective. *Computers in Human Behavior, 29*(6), 2688-2697.

[2] Yee, N. (2006). Motivations for play in online games. *CyberPsychology & Behavior, 9*(6), 772-775.

[3] Sharma, R., Singh, G., & Sharma, S. (2021). Competitors' envy, gamers' pride: An exploration of gamers' divergent behavior. *Psychology & Marketing, 38*, 965– 980.

[4] Taylor, T. L. (2006, April 4). Play Between Worlds: Exploring Online Game Culture.

[5] Kimberly Young (2009) Understanding Online Gaming Addiction and Treatment Issues for Adolescents, *The American Journal of Family Therapy, 37*(5), 355-372,

[6] Consalvo, M. (2009). Cheating: Gaining advantage in videogames. MIT Press.

[7] Mitterhofer, R., & Zimmermann, P. (2019). Cheating in digital games: A systematic review of the literature. *ACM Computing Surveys (CSUR), 52*(2), 1-38.

[8] Kücklich, J. (2005). Precarious playbour: Modders and the digital games industry. *Fibreculture, 5*.

[9] Hamari, J., & Sjöblom, M. (2017). What is eSports and why do people watch it?. *Internet Research, 27*(2), 211-232.

[10] Jeremy Blackburn, Ramanuja Simha, Clayton Long, Xiang Zuo, Nicolas Kourtellis, John Skvoretz, and Adriana Iamnitchi. 2011. Cheaters in a gaming social network. *SIGMETRICS Perform. Eval. Rev., 39*(3), 101–103.

[11] Lu, H. and Wang, S. (2008), "The role of Internet addiction in online game loyalty: an exploratory study", *Internet Research, 18*(5), 499-519.

[12] Lee SJ, Jeong EJ, Lee DY and Kim GM (2021) Why Do Some Users Become Enticed to Cheating in Competitive Online Games? An Empirical Study of Cheating Focused on Competitive Motivation, Self-Esteem, and Aggression. *Frontiers in Psychology, 12*, 768825.

[13] Passmore, C. J., Miller, M. G., Liu, J., Phillips, C., & Mandryk, R. L. (2020). A Cheating Mood: The Emotional and Psychological Benefits of Cheating in Single-Player Games. *Annual Symposium on Computer-Human Interaction in Play.*

[14] Jonnalagadda et al. (2021). Robust Vision-Based Cheat Detection in Competitive Gaming.

[15] H. Alayed, F. Frangoudes and C. Neuman, "Behavioral-based cheating detection in online first person shooters using machine learning techniques," 2013 IEEE Conference on Computational Inteligence in Games (CIG), Niagara Falls, ON, Canada, 2013, pp. 1-8, .

[16] Yeung, Yeung, Lui (2008). Dynamic Bayesian approach for detecting cheats in multi-player online games. *Multimedia Systems, 14*(4), 221-236.

[17] Yang, C., Lin, K., Hsu, S., & Cheng, H. (2014). Cheating detection system for online games using machine learning techniques. *Journal of Computers, 9*(8), 1861-1868.

[18] Kim, J., Kim, H., & Park, S. (2018). A deep learning approach for detecting game bots in MMORPGs using action sequence analysis. *Expert Systems with Applications, 114*, 278-287.

[19] Chapel, Laetitia & Botvich, Dmitri & Malone, David. (2010). Probabilistic Approaches to Cheating Detection in Online Games. Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, CIG2010. 195-201. .

[20] Chen et al. (2018). Game Bot Detection Based on Avatar Trajectory.

[21] Cook, M., & Smith, G. (2017). Artificial intelligence and interactive digital entertainment. *AI Magazine, 38*(2), 9-20.

[22] Fehr, E., & Gächter, S. Altruistic punishment in humans. *Nature, 415*(6868), 137–140 (2002).

[23] Chambers, Chris & Feng, Wu-chang & Feng, Wu-chi & Saha, Debanjan. (2005). Mitigating information exposure to cheaters in real-time strategy games. 7-12. .

[24] Kanervisto, A., Kinnunen, T., & Hautamäki, V. (2022). GAN-Aimbots: Using Machine Learning for Cheating in First Person Shooters. arXiv preprint arXiv:2205.07060.

[25] Castronova, E., & Falk, M. (2009). Virtual Worlds: Petri Dishes, Rat Mazes, and Supercolliders. *Games and Culture, 4*(4), 396–407.

[26] Johnson, M. R. (2018). 'The biggest legal battle in UK casino history': Processes and politics of 'cheating' in sociotechnical networks. *Social Studies of Science, 48*(2), 304–327.

[27] P. Laurens, R. F. Paige, P. J. Brooke and H. Chivers, "A Novel Approach to the Detection of Cheating in Multiplayer Online Games," 12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007), Auckland, New Zealand, 2007, pp. 97-106,

A Appendix

A.1 GitHub Repository 1

You can find the code for this project in our GitHub repository:

https://github.com/ChatGPT-Cheater-Detection-in-Online-Games

A.2 GitHub Repository 2

You can find the dataset we used for this project in Christ Sanchis' GitHub repository: https://github.com/ChristSanchis/memoire.csv

A.3 ChatGPT Prompts

HE

make a density plot for lof # LOF
#Load and preprocess the data

df = pd.read_csv('memoire.csv', sep=';', header=None)

df = df.replace('?', pd.NA).dropna()

df = df.astype(str).apply(lambda x: x.str.replace(',', '.')).astype(float)
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

# Fit the LOF model to the data and predict outlier scores
lof = LocalOutlierFactor(n_neighbors=5, contamination='auto', metric='euclidean')
outlier_scores = -lof.fit_predict(df_scaled)

# Create a scatter plot of the data points with their LOF scores as the color
plt.scatter(df_scaled.iloc[:, 0], df_scaled.iloc[:, 1], c=outlier_scores)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.colorbar()
plt.show()

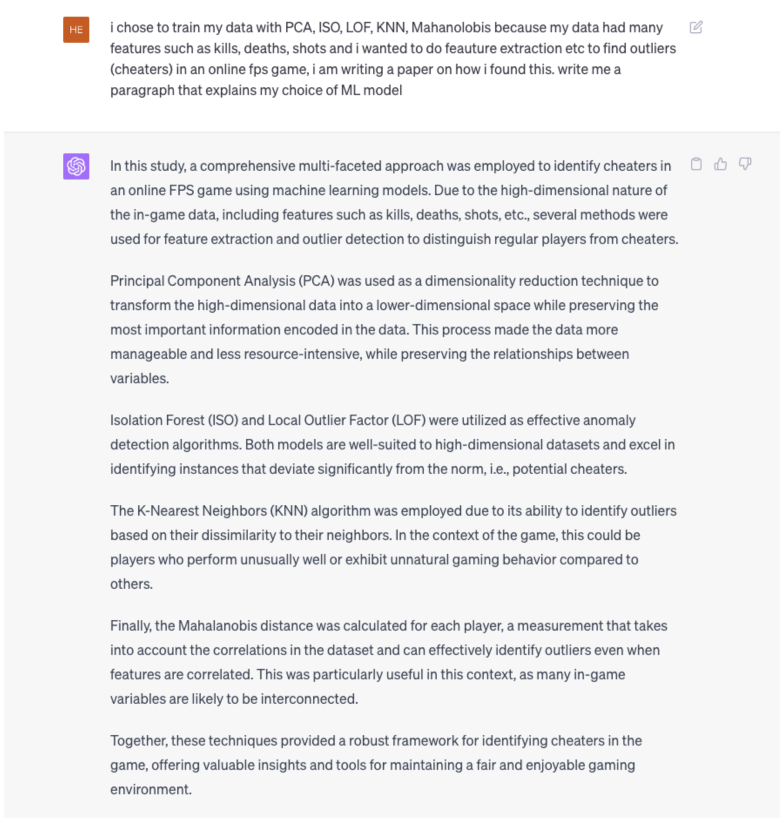
To create a density plot for LOF, you can use the `kdeplot` function from the `seaborn` library. Here's how you can modify your code:

python Copy code

```python
import seaborn as sns

# Load and preprocess the data
df = pd.read_csv('memoire.csv', sep=';', header=None)
df = df.replace('?', pd.NA).dropna()
df = df.astype(str).apply(lambda x: x.str.replace(',', '.')).astype(float)
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

# Fit the LOF model to the data and predict outlier scores
lof = LocalOutlierFactor(n_neighbors=5, contamination='auto', metric='euclid
outlier_scores = -lof.fit_predict(df_scaled)

# Create a density plot of the LOF scores
sns.kdeplot(data=outlier_scores, fill=True)
plt.xlabel('LOF score')
plt.show()
```

This code will create a density plot of the LOF scores using `sns.kdeplot`, and you can customize the plot as needed (e.g., changing the plot title, x-axis label, etc.).

Figure 7: An example of ChatGPT prompts for generating code



HE

i chose to train my data with PCA, ISO, LOF, KNN, Mahanolobis because my data had many features such as kills, deaths, shots and i wanted to do feauture extraction etc to find outliers (cheaters) in an online fps game, i am writing a paper on how i found this. write me a paragraph that explains my choice of ML model

In this study, a comprehensive multi-faceted approach was employed to identify cheaters in an online FPS game using machine learning models. Due to the high-dimensional nature of the in-game data, including features such as kills, deaths, shots, etc., several methods were used for feature extraction and outlier detection to distinguish regular players from cheaters.

Principal Component Analysis (PCA) was used as a dimensionality reduction technique to transform the high-dimensional data into a lower-dimensional space while preserving the most important information encoded in the data. This process made the data more manageable and less resource-intensive, while preserving the relationships between variables.

Isolation Forest (ISO) and Local Outlier Factor (LOF) were utilized as effective anomaly detection algorithms. Both models are well-suited to high-dimensional datasets and excel in identifying instances that deviate significantly from the norm, i.e., potential cheaters.

The K-Nearest Neighbors (KNN) algorithm was employed due to its ability to identify outliers based on their dissimilarity to their neighbors. In the context of the game, this could be players who perform unusually well or exhibit unnatural gaming behavior compared to others.

Finally, the Mahalanobis distance was calculated for each player, a measurement that takes into account the correlations in the dataset and can effectively identify outliers even when features are correlated. This was particularly useful in this context, as many in-game variables are likely to be interconnected.

Together, these techniques provided a robust framework for identifying cheaters in the game, offering valuable insights and tools for maintaining a fair and enjoyable gaming environment.

Figure 8: An example of ChatGPT prompts for generating academic text