A comparison of predictive analysis methods for bot-detection in video games.

Dissertation Submitted by

**Robert Sharp**

Under Supervision of

**Dr Yang Lu**

In Partial Fulfilment for the Degree of

**Games Development (BSc Hons)**



**Declaration:**

I, Robert Sharp, declare that this report which has been submitted for assessment is my own work, has been written in my own words and no part of it has been previously submitted for any other assessments. Any use of other authors' work, whether that be quotes, images, ideas, statistics etc. are correctly acknowledged and listed in the references section at the end of this report.

**Abstract:**

The use of Bots in MMORPG video games to unfairly gain an advantage has resulted in the development and implementation of neural networks to identify them before they can destabilize the game’s economy. However, there has been no evidence that neural network models are any more efficient than supervised classification models. Using a dataset from the MMORPG Aion: The Tower of Eternity. This research performed a comparative analysis of one neural network model (Multi-Layer perceptron), and three supervised classification models (Logistic regression, Naïve-Bayes, K-Nearest Neighbours) to determine which was the most fit-for-purpose for bot detection in MMORPGs. The study found that the K-nearest neighbours was able to perform with the highest accuracy and also performed faster than the Multi-Layer Perceptron model.

**1. Introduction:**

Massively multiplayer online role-playing games, or MMORPGs, are a genre of video categorized by several simple characteristics: a player can create and control their own character, online multiplayer, open-world exploration, and character specialization/ progression. It is this progression system that is the core mechanic of the MMORPG, with many players considering it to be a measure of a player’s true skill at the game. However, some players feel that the progression systems in these games can be tedious and so choose to circumvent them using a method called Botting.

Botting is the use of an automated program to play a video game for you. In most games, botting is frowned upon by players and even considered a bannable offence by some game companies that will prevent you from playing anymore. In the case of MMORPGs, botting is used to level up the player’s character automatically, as well as automatically collecting high-level abilities and items, all without the user having to play the game themselves. In some instances, the bot account will then be sold to another player who can then continue using the account and all its ill-gotten gains. According to the games company Blizzard, 74,000 bot accounts were banned in the MMORPG game World of Warcraft in 2020 alone [2]. Botting is a constant issue, especially so in video games that allow users to create accounts for free. This practice can also disrupt in-game economies that have supply and demand coded into them by selling copious quantities of botted items and reducing their overall price. Similarly, the MMO Lost Ark banned one million bot accounts in 2022, the botting problem in this game was so prevalent that one user reported when the bot ban happened the queue for the game dropped from 8000 to 950[4]; This made the loading times for the game became long enough to cause a significant number of players to quit altogether.

The removal of game-bots (programs created by players to play the game for them) in multiplayer video games is an especially important process is needed to improve the experience of players, especially in the games where botting has more significant benefits, such as the MMORPGs game genre. Many players of MMORPG video games often feel this is unfair as they must work hard to acquire these high-level items and abilities, but the botting player does not.

The gaming industry’s rapid growth in popularity over the past 20 years, as well as the rising interest in machine learning, has encouraged more companies to invest in developing neural network approaches and machine learning algorithms to see if they can be used to detect these bots. While these algorithms are fit for purpose, there is a lack of research into which of these methods or algorithms is the most efficient for bot-detection. This means that game companies may invest in larger-scale machine learning solutions to identify and remove bots from their games. But those larger-scale solutions could be less efficient or unfit for detecting bots.

The literature review aims to investigate the current methods of detecting game bots and identify which of the investigated methods is the most efficient. The objectives of the research will be as follows: A dataset of players and bots will be collected from a known source that has previously been used for testing other machine learning approaches to bot detection. Then, the data from the dataset will be analysed to establish the efficiency of a specified model or method. Multiple algorithms or models will be tested on this same dataset. The data from these analyses will then be compared to determine which method or algorithm is the most efficient for bot detection.

**2. Literature review:**

In recent years there has been a growing trend towards using machine-learning techniques to identify game-bots with ease. Though there has been a trend toward the use of neural models, for example, see Bernardi (2017). There has been some research into whether supervised classification algorithms can perform at the same level as these complex neural network approaches, but not for bot detection. This literature review will study the current methods of data classification such as neural network approaches, as well as supervised learning models.

**2.1 Neural network models.**

The most common solution companies take to automated bot-detection for MMORPGs is neural networks. One commonly used neural network model for handling large amounts of data is the Multi-Layered Perceptron Network (MLP). Bernardi (2017) proposed a framework for using behavioural features to classify actors in online video games using a Multi-Layered Perceptron Network. The paper found that by using their time-series classification model along with the MLP at its highest parameters, and the behavioural features data collected from a game, they could distinguish between human players and automated game bots with a 0.98 or 98% accuracy (Bernardi, M.L. et al, 2017). However, Tao (2018) suggests there is a flaw with the method this paper shows: a complete lack of futureproofing - The network could become obsolete after an amount of time as it focuses on specific behaviours that may become less common in gameplay over time due to the evolving nature of online games. Another concern that this paper shows is the lack of a literature review, which leaves the reader with a lack of understanding as to where this document fits in the literature. This paper does provide significant background information to understand the author's decisions when designing their experiments.

Another approach to neural networks that can be used to solve this problem is the Long-Term Short memory (LTSM) model. Tao(2018) created an LTSM neural network for bot-detection that relies on pattern recognition over a period of time. The LTSM model used both supervised models to detect bots based on the already collected data, while also using the unsupervised models to investigate new types of bots that the training data did not account for. The theory behind their methodology was that an automated program would tend to repeat certain behaviours and that these patterns could be detected and used as training data for a neural network so that it could detect other bots more easily in the future. The paper found that by taking training data from up to 60 seconds of gameplay from its target, the network model could reliably identify bots with great accuracy, as when the model was run at its highest parameters it produced a precision score of 0.99. The paper also makes no mention of its sample size, or sampling criteria making it much harder to validate its results. Xu (2020) expanded on the LTSM model proposed by Tao (2018). The first addition that Xu (2020) made was adding the interval between the actions a player does and the count of how many times each game action has been performed by the player to the model’s variable list. Together, these variables allowed the authors to chart a specific user’s behavioural sequence much more efficiently. Xu (2020)’s highest accuracy score was 0.99 or 99%. Xu (2020)’s literature demonstrates the author’s understanding of the literature and this paper’s place within it. The paper documents its sample size of 3.9 million game bots, which are divided into multiple distinct categories which the paper clearly explained. This paper does not critically appraise the papers themselves, but solely focuses on critiquing their methodology, which suggests a bias toward these papers.

Another approach to neural networks that can be used to detect game bots is behaviour pattern analysis. Huang (2018) proposed a machine learning model for detecting bots in video games based on the pattern of their actions in the game. The model proposed is a temporal behaviour matrix that shows how the two values of action and level change over time; however, only the value action is used for bot-detection. When evaluated, the proposed model showed that the data of regular players often produced a smooth curve, while the data of game bots showed a steeper curve that was easy to detect. Huang (2018) evaluated its model against some other classification methods: A Support Vector Machine and K-Nearest Neighbour model were tested on a sample of 1332 game bots and detected 1032 and 1276, respectively; meanwhile, the method proposed by the paper detected 1296. Huang(2018) did not provide an accuracy score, but the numbers in its results suggest its accuracy was 0.97 or 97%. This paper had a very clearly defined literature review that allowed the reader to gain an understanding of where the author felt were the gaps in the literature; however, most papers in the literature review were outdated. The literature review was also not given its own section but written into the introduction.

A different type of neural network that can be used to detect game bots is real-time prediction models. Tsikerdekis (2020) aimed to produce a prediction model that identifies whether a player was a human or a bot in real-time and use that data to expand its dataset to allow the model to become more accurate over time. The paper’s highest accuracy result was 0.98 or 98%. The model proposed is generalized in a way that allows it to be easily applied to other games, allowing it to be evaluated outside of the environment it was initially designed for. The literature used by this document is mostly up to date and relevant to the topic at hand, though minimal background information is provided to help orient readers to the problem this paper aims to solve. Also, the critique of these papers is generalized with no specific examples, which implies that the author(s) were trying to present a clear problem in their paper without justifying it to the readers first.

One thing to note about these methods is that they are complex. The methods proposed by these papers have limited scalability, which makes them less useful for large online multiplayer games like MMORPGs, which can have hundreds of thousands of players at any given time. This can cause problems with more complicated approaches as they constantly have to check large numbers of accounts and smaller methods can do that more efficiently.

Next, the literature review will evaluate supervised classification algorithms, which are not often used for bot-detection but could potentially prove to be more fit for purpose than the larger neural network approaches seen previously. Overall, neural network models performed very well and the majority of papers were able to produce an accuracy score of 0.98 or higher when their models were run at their highest parameters.

**2.2 Supervised Classification models.**

Logistic Regression (LR) is a supervised classification algorithm used for predicting results based on existing data. Lee (2016) used a similar method for bot detection to Huang (2018), but the authors used logistic regression as their means of comparing data to determine if a bot was detected. This paper was also an attempt to generalize the designs of three other models used for bot detection into a single framework that could be applied to any MMORPG game in the future. The final model had multiple accuracy scores for three different games, the highest of which was 0.99 or 99% for the game Aion: The Tower of Eternity. The authors also spent a considerable amount of effort on error correction, specifically false positives. One major flaw of Lee (2016) is that it relies on outdated literature, which puts the validity of its results into question. The paper attempts a literature review; however, it is very noticeably short and lacks any meaningful criticism of the individual papers it reviews. This provides no insight into the literature or into what gap the creator intended to fill.

Logistic regression is widely used as a model for machine learning and many attempts at optimizing the model have been made to make it more efficient over time. For example, Zou (2019) proposed obtaining the regression coefficient through the gradient descent method and using it to optimize the sigmoid function so that the number of iterations needed to run the Logistic regression model could be reduced without compromising the accuracy of the model. Zou(2019) calculated its model’s accuracy score as 96.14% or 0.9614. The data presented in this paper is laid out without the use of a table which makes it more difficult to read; this makes the paper's value to the current research hard to determine.

While some are trying to improve the efficiency of the Logistic Regression model, others are trying to test whether it is as efficient as other machine learning models. For example, Christodoulou (2019) performed a systematic review into whether a Logistic regression model method could outperform other machine learning methods. The paper did not have an accuracy score of its own, rather having a comparative one with other models. The researchers found that on average other machine learning techniques had an AUC score of 0.25 points higher; however, the difference between the ACU scores for methods with a lower risk of bias was 0.00. This shows there is no significant difference between using a logistic regression model and using a machine learning model. This paper specified inclusion and exclusion criteria for whether specific studies would be included in the review, and the 71 studies were chosen from a variety of sources that were up to date at the time of the paper’s publishing.

Naïve-Bayes classifiers are another method for predicting information based on existing data they have been given. Naïve-Bayes classifiers are one of the most popular forms of data classification due to their efficiency, which comes from the assumed independence of variables in the given dataset (Chen, S. et al 2020). Naïve-Bayes is a highly scalable classification model that is based on Bayes’ theorem, which states that the probability of events occurring is based on past data of conditions related to those events. Chen(2020) proposed a method to make the Naïve-Bayes classifier work with large amounts of data was proposed by Chen (2020). The paper proposes a selective model that uses cross-validation to rule out values which have no impact on the results, thus increasing efficiency while maintaining its accuracy. The results of their experiment produced an accuracy score of 0.9353, though this result was not written in the result section but was within their experiment procedure. This attempt to obscure their results implies that the researchers were not proud of them. The paper also provides no comparative numbers to show that the accuracy or the efficiency of the model it proposed is any different to the models it was compared against.

Naïve-Bayes is most commonly used for image classification. Sugiarti (2021) proposed using the GLCM function extraction method to collect their data from images of diseased apples, and then using the Naïve-Bayes classifier to identify the apples based on these images. According to their results, the classifier was able to identify the diseases certain apples had with a 96.43 per cent accuracy rating. One flaw with the paper is the literature review, which is short and cluttered; there are no sections to explain why both the GLCM feature extraction and the Naïve-Bayes classifier were chosen as part of the methodology. This makes it hard to understand why the authors chose their methodology.

There have been attempts to use the Naïve-Bayes classifier for bot detection. For example, Lee (2015) uses the Naïve-Bayes classifier for bot detection, specifically by analysing the action sequences of bot players and human players to determine in which order they would perform certain actions in-game. The paper claimed to have a precision score of 1.00, but that is unlikely given the paper's conclusion stating that the dataset used for analysis was not large enough. The paper concluded that the Naïve-Bayes classifier was inefficient for larger amounts of data, which hindered the identification of game bots substantially, which is a recurring problem for this Naïve-Bayes algorithm. This is because the Naïve-Bayes algorithm is better suited for categorical data, not numerical data.

K-Nearest Neighbours (KNN) is a commonly used machine learning model used to solve both classification and regression problems; it works by finding the distance between a given point and classifying it based on how it compares to the example data it is given. However, the KNN algorithm struggles to work with larger data due to the model having to compare the query data to a larger number of points before it can be classified (Saadatfar, H. et al, 2020). While clustering is a good optimization for KNN and drastically improves its handling of large amounts of data, the clustering method can be optimized further. Saadatfar (2020) also proposed a method for pruning excess values to optimize the shapes and densities of the clusters. It did this by using K-means to cluster the initial dataset; then it classifies the data, not by distance from the centre of each group but also by using the shape and density as chrematistics to determine a point's proper placement within a group. The paper's results clearly show that it was consistently performed at the same speed as a clustered KNN while providing slightly higher accuracy. For example, in one test the standard clustered KNN got an accuracy of 0.9275, while the proposed approach had an accuracy of 0.9475. The paper is clearly laid out and provides a clear understanding of the author’s intent, and their research is valuable as it provides the efficiency of a higher-end model at a similar speed to a less computationally expensive one.

There are other solutions to KNN’s struggle with large amounts of data. For example, the solution to this problem is to use a clustering algorithm to group similar values and generate a mean from them, which can then be used to classify new points efficiently rather than wasting computation power on testing a new value against every previous one. With this method, Gallego (2018) was able to correctly identify 98.7% of 1000 points given, while a standard KNN algorithm was only able to correctly 86.9% showing that the clustering method is a viable optimization for a large amount of data, though its results were not presented in the most efficient way.

**2.3 Summary:**

Game bot detection is a significant problem that any online multiplayer game has to solve as botting often provides an unfair advantage to players who partake in it, while players who don’t are rewarded fairly.

Most modern models for bot-detection frequently rely on neural networks that are often difficult to develop, while there has been very little investigation into simpler models. These networks often require a lot of time to develop, but also a large amount of training data that has to be carefully curated. This results not only in lost productivity but also creates an issue of redundancy. This research proposes an investigation into less computationally intensive machine learning approaches to test whether they are fit for game bot detection and to see if any can provide a viable result to the neural network approaches.

**2.4 Research problem:**

In the past few years, there has been a growing need for video games to implement a bot-detection service to remove the unfair advantage botting gains for players. Most research in the field indicated a growing preference for the use of computationally expensive neural network approaches to solve this problem. However, there has been significantly less research into less computationally expensive methods like Supervised classification models. This study proposes a comparative analysis of supervised classification models to determine if they can be more efficient for bot-detection than neural network approaches.

**2.4.1 Research question:**

• Can supervised classification models provide more efficient results than neural network models?

**2.4.2 Research aims:**

1. To identify the most efficient methods between the following machine learning models for bot-detection in MMO video games: Logistic regression, K-nearest Neighbours, Multi-Layer Perceptron (MLP), and Naïve-Bayes.

2. To test if the proposed supervised classification models can provide a competitive result to a more computationally heavy neural network(MLP).

**3. Methodology:**

**3.1 Hypothesis:**

The researcher hypothesised that the logistic regression model would perform the most efficiently out of the tested models as it, unlike Naïve-Bayes, works well with numerical data, and the literature seems to suggest that the K-Nearest Neighbour algorithm is inefficient without the implementation of clustering. The Multi-Layer Perceptron model should in theory produce a higher accuracy score as it is the most computationally expensive model. But the researcher predicts it should also be the slowest of the four models.

**3.2 Experiment Design:**

This research paper follows a quantitative study that compares the efficiency of machine learning methods for bot-detection within video games. The study involved running an individual machine learning model on a bot-detection dataset[3]. This study hopes that by testing the machine learning models on the same dataset, the research can provide a fair comparative analysis of different machine learning methods for bot detection similar to the experiment performed by Christodoulou (2019) which did perform the same tests on two different models to determine if logistic regression could outperform neural networks like Support Vector Machines(SVM). The research used a random sampling of accounts listed in the dataset in the training of the model; crucially, they were all trained with the same sample size. The study considered two important values when determining which methods performed better: the accuracy of the results as well, and the time elapsed to receive that result. The study will evaluate the following machine learning approaches: Logistic regression, Multi-layer Perceptron, K-nearest Neighbours and Naïve-Bayes classification to find out which of the four is the fittest for the purpose of bot-detection. The multi-layer Perceptron was the only neural network method included. Bernardi (2017) used this model to classify actors and it was included to provide a comparison with a neural network against the classification algorithms.

The experiment software was written in the Python programming language and run on the Python IDE Version 3.10.4. The experiment software was designed and run on the Windows 10 Operating System Version 21H2. Though the PC that the software was run on has a six-core AMD Ryzen 5 2600X Six-Core processor, as well as 16GB of available memory to rely on, which may improve the speed at which the results are produced.

The metrics the data would produce were as follows: From the resulting data, an accuracy score was calculated using the formula: (Humans predicted correctly + Bots predicted correctly)/ (Total number of tested records) which produces a percentage of correct results called an accuracy score. The time taken for the experiment in seconds would be read once the model had finished running, which would also be recorded.

The models will be tested without any optimizations, and if time permits later the research will be expanded to include optimized versions of these methods as well. The study was conducted by testing these models against a single dataset and then comparing their results to determine which results produced the most accurate results in the shortest amount of time. By changing the parameters, you can drastically alter the results of a neural network model. The parameter set for the KNN model was the N\_neighbours value of 5, which is the default value. It was set at this value to prevent the Model from having an unfair advantage during the testing phase. Both the Naïve-Bayes model and the logistic regression models were given no additional parameters other than the defaults the library provided. The MLP model was given a higher alpha regularization parameter to prevent overfitting of the data and skewing the results in its favour. It was also given a maximum iteration value of 700. Another parameter that the model was given was hidden\_layer\_sizes which was set to 5 layers, with 2 nodes each. These modifications to the MLP model were not the default values but were made so that the model could refine its results as much as possible; however, these parameter changes did not seem to impact the model at all during the testing phase.

**3.3 Data Source:**

The dataset for this project came from user Nanjangpan on GitHub[3] and was collected from the MMORPG game Aion: The Tower of Eternity. The dataset was created by researchers from Yonsei University’s Hacking Response Technology Lab as part of a competition. The goal of the project was to collect data for detecting game bots which is the exact purpose the current research seeks to use it for, and all the variables in the dataset are actions that players and bots in the game could perform meaning the bots are being investigated based on their relative action frequency like in Tsikerdekis (2020). An example of these variables is ‘avg\_money’, which is the average amount of money a player has on their account – this variable is useful for bot-detection as bot-players often sell large amounts of items to gain a significant advantage for themselves which means players with a higher ‘avg\_money’ variable are often more likely to be bots.

This dataset was chosen for this research because it contained 8101 samples, of which 600 were game bots – this meant that any machine learning approach would be biased towards detecting human players. Successful bot detection models need to correctly identify human players as much as possible to prevent false positives that would get a legitimate player banned from the game. A crucial step that was taken with the dataset was determining which values the research would be using, and which were pruned to make the models more efficient. Several variables were removed from the dataset due to the majority of their values being 0, these values were: ‘exp\_repair\_count’, ‘exp\_repair\_count\_perday’, ‘abyss’, ‘use\_portal’. ‘use\_portal\_count\_perday’ and ‘guild\_join.’ While some accounts did have values in these variables, the majority of accounts did not. This means this variable would contribute little to the final result and would only increase the cost of computation if they were included.

**3.4 Experiment Procedure:**

Once validation of the database had been completed, the database was saved as a .csv file which was then imported into Python. From there the study would use the Sklearn library, a free library designed for the purpose of machine learning, to evaluate whether the dataset was working correctly. The study chose to use this library as implementing the machine-learning models from scratch proved too difficult without prior experience in doing so, which could introduce significant errors with the results of the algorithm. After the library was up and running, the parameters were set for each model.

The experiment software was broken down into four python files, one for each model the researcher was testing. The programs all followed the same procedure; first, they would import the edited training data file, and assign the labels to it using the pandas library. There were a few issues with this library, such as importing the dataset from, for example, the data would be read as strings were specified to be floats. By using the Pandas DataFrame format this could be prevented. Next, the first column of the dataset would be removed as it contained the account ID which did not need to be included in the training data. Following this, the clean\_dataset function ran which removed non-numerical values and replaced them with 0.

After that, the software broke down the training data into two parts, X – which contained all of the account variables; and Y – which contained the ground-truth value for that account. Then the data was split into two groups using the inbuilt ‘test\_train\_split’ method. The sample size of 20% was chosen because it provides a sample of 1,620 data values to test the algorithm against which provides more than enough training data to prove the overall efficiency of the model. This project used probability sampling to randomly select 20% of the accounts listed in the dataset and used them as training data. This sampling method was chosen because it provides a representative sample of the data as a whole.

After that, the software ran both of the two groups of data through a standard scaler. It did this because the dataset was too large for the program to iterate over, and implementing a standard scaler was the method the Sklearn library provided for correcting that issue.

Finally, the data would be evaluated by the model. In the initial testing phase, a single value from the dataset was evaluated against the model first, this value was run through the experiment software to confirm that the model was functioning correctly and could make accurate predictions. Next, the model was run on a training sample a total of ten times, to guarantee that the results were repeatable and were not the result of an error. Then a confusion matrix of the results was produced to properly display the results. Another feature that was added to the program was the time library, which was used to identify when the program started and ended to determine how fast it was at producing the results. This would be important for identifying the most efficient of the four programs.

The data collected from all of the tests would be used to determine which of the methods was faster, and which method was more accurate.

**4. Results:**

In total, the four machine learning models took a total of three days to read the documentation for the Sklearn library and gain a good understanding of how each model was supposed to be implemented. Having done that, the models themselves only took around 2 hours to implement including debugging. The models were tested against 1620 randomly selected values.

The following equation was used to calculate the accuracy score: Correctly guessed human accounts + correctly guessed bot accounts / by total number of tested accounts.

**4.1 Naïve-Bayes classifier tests:**

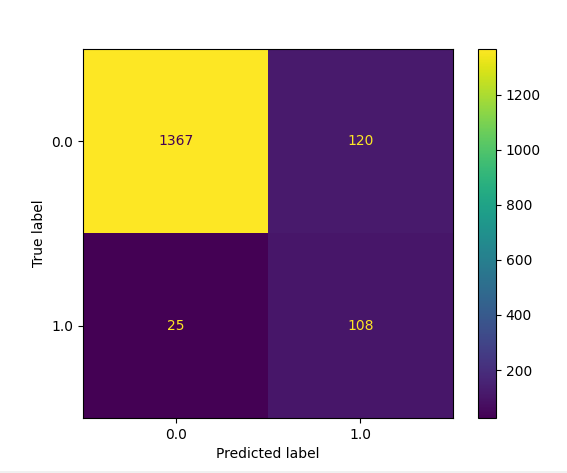


Figure 1. Confusion Matrix of the Naïve-Bayes model

Table I. A table of the results of the Naïve-Bayes model on the dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision:  (%) | Recall:  (%) | f1-score:  (%) | Total Values: |
| Value:  0 | 0.98 | 0.92 | 0.95 | 1487 |
| Value:  1 | 0.47 | 0.81 | 0.60 | 113 |
| Accuracy:  (%) | 0.73 | 0.87 | 0.77 | 0.91 |
| Time Elapsed:  (s) |  |  |  | 0.29 |

The model predicted correctly predicted that 1367 accounts were humans (True positives) and predicted that 120 humans were bots (False positives); the models predicted that 25 bots were human (False negatives) and that 108 bots were bots (True negatives); see Figure 1 for details. This test was repeated several times to ensure the result wasn’t an outlier. Using the equation, the precision score of the model was calculated to be 0.91 or 91%. The time elapsed between starting the program and the result being produced was 0.29 seconds, meaning it was able to investigate the 1620 accounts given in that time. See Table I, for details.

**4.2 K nearest neighbours tests:**

Chart, Confusion matrix chart

Description automatically generated

Figure 2. Confusion Matrix of the KNN model

Table II. A table of the results of the KNN model on the dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision:  (%) | Recall:  (%) | f1-score:  (%) | Total Values: |
| Value:  0 | 0.98 | 0.99 | 0.98 | 1504 |
| Value:  1 | 0.88 | 0.69 | 0.98 | 116 |
| Accuracy:  (%) | 0.93 | 0.84 | 0.88 | 0.97 |
| Time Elapsed:  (s) |  |  |  | 0.36 |

The model predicted correctly predicted that 1493 accounts were humans (True positives) and predicted that 11 humans were bots (False positives); the models predicted that 36 bots were human (False negatives) and that 80 bots were bots (True negatives); see Figure 2 for details. This test was repeated several times to ensure the result wasn’t an outlier. Using the equation, the precision score of the model was calculated to be 0.97 or 97%. The time elapsed between starting the program and the result being produced was 0.36 seconds meaning, it was able to investigate the 1620 accounts given in that time. See Table II, for details.

**4.3 Logistic regression tests:**

Chart, treemap chart

Description automatically generated

Figure 3. Confusion Matrix of the Logistic Regression model

Table III. A table of the results of the Logistic Regression model on the dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision:  (%) | Recall:  (%) | f1-score:  (%) | Total Values: |
| Value:  0 | 0.96 | 0.98 | 0.97 | 1496 |
| Value:  1 | 0.68 | 0.47 | 0.56 | 116 |
| Accuracy:  (%) | 0.82 | 0.72 | 0.76 | 0.94 |
| Time Elapsed:  (s) |  |  |  | 0.14 |

The model predicted correctly predicted that 1469 accounts were humans (True positives) and predicted that 27 humans were bots (False positives); the models predicted that 66 bots were human (False negatives) and that 58 bots were bots (True negatives); see Figure 3 for details. This test was repeated several times to ensure the result wasn’t an outlier. Using the equation, the precision score of the model was calculated to be 0.94 or 94%. The time elapsed between starting the program and the result being produced was 0.14 seconds meaning, it was able to investigate the 1620 accounts given in that time. See Table III, for details.

**4.4 Multi-Layer Perceptron network test:**

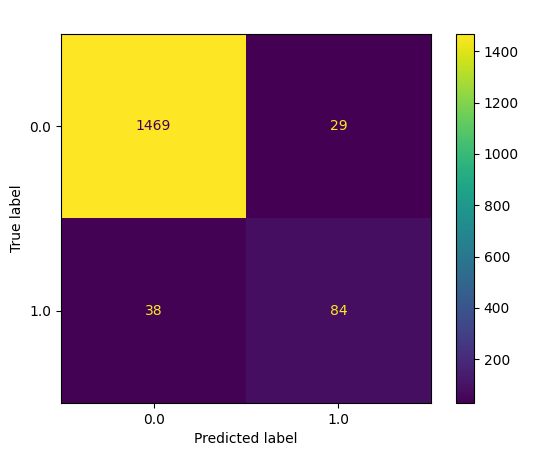


Figure 3. Confusion Matrix of the Multi-Layer Perceptron model

Table IV. A confusion matrix of the results of the Multi-Layer Perceptron model on the dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision:  (%) | Recall:  (%) | f1-score:  (%) | Total Values: |
| Value:  0 | 0.97 | 0.98 | 0.99 | 1498 |
| Value:  1 | 0.74 | 0.69 | 0.71 | 122 |
| Accuracy:  (%) | 0.86 | 0.83 | 0.85 | 0.96 |
| Time Elapsed:  (s) |  |  |  | 1.63 |

The Multi-Layer Perceptron model predicted correctly predicted that 1469 accounts were humans (True positives) and predicted that 29 humans were bots (False positives); the models predicted that 38 bots were human (False negatives) and that 84 bots were bots (True negatives); see Figure 4 for details. This test was repeated several times to ensure the result wasn’t an outlier. Using the equation, the precision score of the model was calculated to be 0.96 or 96%. The time elapsed between starting the program and the result being produced was 1.63 seconds, meaning it was able to investigate the 1620 accounts given in that time. See Table IV, for details.

**4.5 Comparative analysis:**

Table X. The average results of the AI models

|  |  |  |
| --- | --- | --- |
| Model | Accuracy (%) | Time  (s) |
| Logistic Regression | 0.94 | 0.14 |
| K-Nearest Neighbours | 0.97 | 0.36 |
| Naïve-Bayes Classifier | 0.91 | 0.29 |
| Multi-Layer Perceptron | 0.96 | 1.63 |

The Naïve-Bayes model performed the fastest overall but provided the worst results on average. The Naïve-Bayes model was also the only model to produce a drastically different outlier while the testing process occurred, which decreases its usefulness for bot detection. The Logistic regression model was able to provide better Accuracy than Naïve-Bayes but was slower on average when it came to classifying records. The K-Nearest neighbours algorithm was the slowest of the classification models when it came to classifying records, but had a higher accuracy than all of the other methods.

Finally, the Multi-layer Perceptron network was the slowest of all of the models tested. It also produced the second-highest average accuracy score, though it consistently performed slower than the K-Nearest Neighbours method.

**5. Discussion:**

The results appear to indicate that of the four models tested, the one which was most fit for purpose was the K-nearest neighbours model because while it did perform slower than the Logistic regression and Naïve-Bayes models, it produced a consistently higher accuracy rating while not being slower than the most Multi-Layer Perceptron method.

**5.1 Interpretations:**

The comparative analysis of the four machine learning models confirms that of the four models tested, the most fit-for-purpose for game-bot detection would be K-Nearest Neighbours. While this model was the slowest of the classification algorithms, its speed only varied from the fastest classification model (Logistic Regression) by 0.22 seconds which is not a substantial difference. The model also produced the highest accuracy score more consistently than all of the other models, including the Multi-Layer Perceptron(MLP) neural network, which the researcher had anticipated would outperform all of the other models. Detecting game bots with the KNN model would provide the most consistent and accurate result meaning there would be less chance of a player being accidentally mistaken for a bot and banned.

Another important fact to note is that the MLP neural network performed worse on average than the K-Nearest Neighbours model. For bot accounts specifically, the KNN model had a precision of 0.88, a recall value of 0.69, and an f1-score of 0.98. Meanwhile, the MLP model had a precision of 0.74, a recall value of 0.69, and an f1-score of 0.71. See Tables II and IV for more details. This data suggests that the MLP model was worse on average for identifying bot accounts. While their overall accuracy scores were close in value(0.97 and 0.96 respectively), the MLP network would be the worse option of the two for correctly identifying bot accounts. The KNN model had the highest precision, recall, and F-1 score of all of the models when it came to detecting bot accounts. The models tested it was the most fit for purpose for detecting bots; see Tables I-IV for details.

The hypothesis made by this paper stated that logistic regression would be the most fit for purpose of the supervised classification methods for bot detection. This hypothesis was incorrect as K-Nearest Neighbours proved to be more provide more accurate and consistent results, See table V for details. The results contradict the claims of Saadatfar (2020) that the K-Nearest Neighbour would not perform when having to classify large amounts of data. It should also be noted that the Multi-Layer perceptron network was expected to perform with the highest accuracy due to it being a neural model; however, it proved to perform worse than the KNN model as well. While this study only tested the model and dataset meant for bot-detection, this could be interpreted as the KNN model being more well-suited for problems like classifying bot-detection datasets than others. However, the assumption that the naïve-bayes classifier’s difficulty interpreting numerical data would hinder it in this research as it produced the lowest highest accuracy score.

Bernardi (2017) created a model for identifying whether or not an account was a game bot; their study’s model performed with an accuracy rating of 0.98 at maximum, though this result was only provided by their most high-end model. The dataset used by Bernardi(2017) had a total of 57,441 values as stated by the paper itself, meaning that the K-nearest neighbour model implemented by this study was to produce a result of similar accuracy while only having a sample of 1620 data values to train it against. Bernardi(2017) also listed how long its highest accuracy model took to build, which was a total of 614 seconds / 10 minutes and 14 seconds. The model proposed by Bernardi(2017) took longer to develop than any of the models proposed by this study, and the KNN model from this study provided a competitive result.

This study also recreated the Multi-layer Perceptron model used by Bernardi (2017), which performed at the same level as the K Nearest neighbours model in only 1 of the 10 tests. to this example from literature and could potentially outperform it if optimized using a clustering algorithm like suggested by Gallego (2018).

**5.2 Implications:**

These results should be taken into account when considering how to develop a machine learning model for bot-detection in MMO video games because the results suggest that a simple classification algorithm can perform at the same level as many of the more computationally heavy methods that are currently used for the same purpose.

The findings of this research have the potential to change how bot-detection systems in MMORPGs work as the two most common approaches to detecting bots currently are to either have someone review a player’s gameplay by hand to determine if they are a bot or to use a neural network model to identify bots based on the behaviour of other bots that has been collected in the past.

Previous research focused heavily on neural network models as a means to automate bot detection for video games. The MLP neural network implemented performed worse than the KNN model proving there are circumstances where more computationally heavy methods are not the most suitable approach to a given problem, and bot-detection is one of them.

**5.3 Limitations:**

There are potential limitations to the current study. One limitation of this study is that there had been little effort in the literature into comparing the use of different machine learning models for bot detection. Christodoulou (2019) provided the only example where a supervised classification algorithm was tested and compared to a machine learning method and found there was no significant difference between the two.

There are three major methodical limitations in this study that could be addressed in future research. First, the study focused primarily on supervised classification models. Second, this study gave little time to more complex neural network models; it is entirely possible that a neural network model could outperform those used in the study. Third, the models used for testing the data were implemented from the Sklearn library, rather than them being implemented from scratch; this was done to compensate for the researcher’s lack of experience using machine learning algorithms, but it also took a level of control away from the researcher that they would otherwise have when developing the models from scratch. Finally, by using a dataset collected from GitHub[3] the researcher had no control over how the data was collected, or what data was collected, and so any biases in the dataset could not be accounted for. Another issues was that the GitHub page was written in Korean, which made understanding what each of the variables was supposed to be rather challenging, but not impossible due to other research using similar values. If the research was repeated there would be time devoted to collecting a dataset from an MMORPG first so the data could more easily be understood and validated.

Another limitation of this study is the time elapsed during the tests, which the researcher is currently unsure of the validity of. Despite the models being tested on 1620 data values each, all of the models were able to process the data and provide an accuracy score in under 2.3 seconds. A possible explanation for this is that the computer used to process the results had a Six-Core processor as well as an in-built graphics processing unit which could have handled a load of such as a large amount of data very quickly. Whether or not those components had a significant impact on the result will require more testing.

The final limitation of this study is a lack of researcher experience. This study was proposed by an undergraduate student who had never worked with machine learning technologies before. This problem led to a large amount of time spent troubleshooting the models used in the study, time which could have otherwise been spent testing other models like the Support Vector Machine(SVM) model. This means that there is a potential that the models were simply not implemented correctly or working properly, and the researcher had no means of comparing the results to find out for certain. This was the biggest hindrance to the study. The models tested in this study required a large amount of troubleshooting and data validation to make them function correctly, which ultimately limited the amount of time that could be spent on the experiment. The researcher would have tested several more models if it were not for this hindrance, such as the Long Term Short Memory(LTSM) model proposed by Tao(2018).

There is an alternative explanation for the results, which is that machine learning methods are often better suited to different types of data. For example, the Naïve-Bayes classifier is better suited to categorical data, rather than the numerical data that was used in these experiments. If the dataset used in this study was modified into a categorical form, it could be expected that the Naïve-Bayes model would work better than the other models when classifying the data. This explanation also provides an explanation as to why the Multi-layer Perceptron network the researcher implemented performed better than it did when used by Bernardi (2017) as it used time-series data rather than numerical data, the MLP model may operate better with the type of data this study used, though a larger dataset and more research would be needed to confirm that.

Despite these limitations, there are still advantages of the methodology proposed by this study. The first advantage is that it can easily be modified to use a different bot-detection dataset by simply replacing which file is being used by the programs, allowing the machine learning models to be adapted to any new game that has collected data on the bot and human accounts. Another advantage is that the methodology was very simple to implement, allowing game companies to do so without the considerable commitment that designing a more complex machine learning approach would provide.

This study also provides a simple approach for testing the efficiency of the machine learning model for bot detection. However, this could be generalised and used as a means for comparing and contrasting the usefulness of machine learning models as a whole.

**5.4 Recommendations & Future work:**

The first piece of future work this study recommends is more studies to build upon the current research. Starting with more comparative analyses of machine learning methods for bot-detection. These analyses could save gaming companies money that they would otherwise use to develop large-scale machine learning models for bot-detection, which they may not need. This study already showed that one neural network approach could be less useful than a supervised classification algorithm, and more analyses of this type could do the same.

The second piece of future work this study recommends is a reassessment of this study, specifically its methodology and its results. The following limitations may need to be addressed with more research; this study’s focus on supervised classification models, this study’s lack of investigation into larger and more computationally expensive neural networks, the study’s use of the Sklearn library and whether or a hand-written code could outperform it. It would also be a valuable research topic to investigate the results produced by the models used in this study and validate them for further use. It may also prove useful to investigate whether or not having a higher-end computer affected the speed at which the results were produced. This study could be improved by addressing those concerns.

**6. Conclusion:**

The purpose of this chapter is to conclude the study. It will do this by summarising the key research findings, to what degree the research aims were met, and discussing the value of the research and its contribution to the literature going forward. This chapter will also review the limitations of the study and propose opportunities for future research.

In recent years, there has been a growing trend towards neural network-based bot-detection models for bot-detection in MMORPG. The literature hadn’t tested supervised classification models for this purpose. This study proposed a comparative analysis of supervised classification models to determine if they can be more efficient for bot-detection than neural network approaches.

This study ran a comparative analysis to determine which of the following machine learning models was the most efficient: K-Nearest Neighbours, Logistic Regression, Multi-Layer Perceptron, and Naïve-Bayes classifier. Multi-Layer Perceptron was included to provide the comparison to a neural network method. The study determined the most efficient by taking note of which model produced the highest accuracy score in the shortest amount of time.

The study hypothesised that logistic regression would perform the best out of the supervised classification models. However, the results appeared to show that K-nearest Neighbours was the most efficient of the models and even outperformed the result of the MLP network showing that supervised classification algorithms can be more efficient than neural network models. For bot-detection, K-Nearest Neighbours is a computationally cheap, yet effective model.

This result does appear to show that a K-Nearest Neighbours supervised classification model can outperform an MLP neural network model. Therefore showing that a supervised classification model can be more efficient than a neural network model for bot-detection. This matters because game companies often devote large amounts of resources to designing, implementing, and maintaining networks for bot detection when they rely on a simpler and more effective model instead. A games company could implement a supervised classification model instead, which could potentially save a considerable amount of money that they would otherwise have to spend. KNN is by no means the perfect or most efficient model for bot detection, but it is cheaper and still very effective for such a simple model.

In conclusion, the K-Nearest neighbours model was the most efficient model for bot-detection in MMORPG video games, among those tested by this study at least. If given enough time and resources to expand this study, the researcher would do so by testing more models with the hope of identifying one that could outperform the KNN model’s level of accuracy, but at a much faster speed.

**7. Appendices:**

**[1]** – Project source code. Available from: https://github.com/RobertSharp2001/DissertationFiles

**8. References:**

Bernardi, M.L., Cimitile, M., Martinelli, F. and Mercaldo, F., 2017. A time series classification approach to game bot detection. In Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics (pp. 1-11).

Blizzard Entertainment (2020) Actions Taken to Address Exploitative Gameplay. Available from https://us.forums.blizzard.com/en/wow/t/actions-taken-to-address-exploitative-gameplay/558339. [Accessed 15th January 2022] **[2]**

Chen, S., Webb, G.I., Liu, L. and Ma, X., 2020. A novel selective Naïve-Bayes algorithm. Knowledge-Based Systems, 192, p.105361.

Christodoulou, E., Ma, J., Collins, G.S., Steyerberg, E.W., Verbakel, J.Y. and Van Calster, B., 2019. A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models. Journal of clinical epidemiology, 110, pp.12-22.

Gallego, A.J., Calvo-Zaragoza, J., Valero-Mas, J.J. and Rico-Juan, J.R., 2018. Clustering-based k-nearest neighbor classification for large-scale data with neural codes representation. Pattern Recognition, 74, pp.531-543.

Huang, X., Fan, J., Gao, S. and Hu, W., 2018. Online game bot detection based on extreme learning machine. In Transactions on Edutainment XIV (pp. 155-165). Springer, Berlin, Heidelberg.

Lee, E., Woo, J., Kim, H., Mohaisen, A. and Kim, H.K., 2016, February. You are a Game Bot! Uncovering Game Bots in MMORPGs via Self-similarity in the Wild. In Ndss.

Lee, J., Lim, J., Cho, W. and Kim, H.K., 2015. In-game action sequence analysis for game bot detection on the big data analysis platform. In Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems-Volume 2 (pp. 403-414). Springer, Cham.

Baek, J/ Nanjangpan. GitHub (2019) Game-Bot-Detection. Available from https://github.com/Nanjangpan/Game-bot-detection. [Accessed 7th January 2022] **[3]**

PC Gamer (2022) Lost Ark is banning 'over a million accounts for using bots. Available from https://www.pcgamer.com/uk/lost-ark-is-banning-over-a-million-accounts-for-using-bots/. [Accessed 23rd March 2022] **[4]**

Saadatfar, H., Khosravi, S., Joloudari, J.H., Mosavi, A. and Shamshirband, S., 2020. A new K-nearest neighbors classifier for big data based on efficient data pruning. Mathematics, 8(2), p.286.

Sugiarti, Y., Supriyatna, A., Carolina, I., Amin, R. and Yani, A., 2021, September. Model Naïve-Bayes Classifiers For Detection Apple Diseases. In 2021 9th International Conference on Cyber and IT Service Management (CITSM) (pp. 1-4). IEEE.

Tao, J., Xu, J., Gong, L., Li, Y., Fan, C. and Zhao, Z., 2018, July. Nguard: A game bot detection framework for netease mmorpgs. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 811-820).

Tsikerdekis, M., Barret, S., Hansen, R., Klein, M., Orritt, J. and Whitmore, J., 2020. Efficient Deep Learning Bot Detection in Games Using Time Windows and Long Short-Term Memory (LSTM). IEEE Access, 8, pp.195763-195771.

Xu, J., Luo, Y., Tao, J., Fan, C., Zhao, Z. and Lu, J., 2020. NGUARD+ An Attention-based Game Bot Detection Framework via Player Behavior Sequences. ACM Transactions on Knowledge Discovery from Data (TKDD), 14(6), pp.1-24.

Zou, X., Hu, Y., Tian, Z. and Shen, K., 2019, October. Logistic regression model optimization and case analysis. In 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT) (pp. 135-139). IEEE.