SIR KARL POPPER SCHULE
WIEDNER GYMNASIUM

Vorwissenschaftliche Arbeit im Rahmen der Reifeprüfung

The utilization of Artificial Intelligence in *MOBA* video games

League of Legends: Using Machine Learning models for win prediction

Joshua Gao

8C, 2024/25

Bundesgymnasium und Bundesrealgymnasium Wien 4

Wiedner Gymnasium / Sir Karl Popper Schule
A-1040 Wien, Wiedner Gürtel 68

Betreuungslehrperson: BEd Alexander Höfler

Vorgelegt am 12.02.2025

Abstract

Machine Learning model evaluation for Multiplayer Online Battle Arena (*MOBA*) video game win predictors require a vast amount of dedicated training time when large datasets are provided. As Machine Learning models and algorithms are made of different structures and do not abide by the same learning rules, it is probable that accuracy scores deviate from each other. This work has been carried out to examine the performance of Machine Learning models in order to detect those which are most suitable for video game win predictors. Orthogonal to evaluating model accuracy, this study also proposes another algorithm, based on the Pearson correlation coefficient, to compute the probability of a team winning in *MOBA* video games. The Pearson Correlation Coefficient-Based Algorithm (*PCCBA*) has been implemented to analyze its performance and compare it to its Machine Learning counterparts. Moreover, it is notable to include that the dataset comprises data points of a vast number of video game matches from the *MOBA* League of Legends, which was created and published by the game developer company Riot Games in 2009. The dataset from this study was acquired by utilizing the Riot API.

The Machine Learning models which were applied onto the datasets are the following: Logistic Regression, Decision Tree, Random Forest, K-Nearest-Neighbor, Naive Bayes, Support Vector Machine, Neural Network, Ada Boost and Gradient Boosting. The implementation of the respective models required the scikit-learn Python library (cf. Pedregosa et al. 2011), while other libraries, such as numpy (cf. Harris et al. 2020), matplotlib (cf. Hunter 2007), pandas (cf. McKinney 2010), and seaborn (cf. Waskom 2021) were used to facilitate data preparation. This work has put its focus on comparing the accuracies of the Machine Learning models and plotted multiple graphs to visualize the data. During the evaluation of the different models, the findings suggest that the algorithm which relies on input and output variable relationships should be capable of predicting game outcomes.

Contents

1 Introduction

AI (Artificial Intelligence) plays a crucial role in the field of technology. On top of being capable of handling tasks which usually require human intelligence, it can often outperform conventional algorithms in solving complex problems. Akin to finding its implementation in people's everyday lives, one can also observe its application in video games like League of Legends.

League of Legends is a real-time strategy video game that has been published by the game developer company Riot Games. The company has already organized various tournaments for this game which were available on multiple platforms such as YouTube and Twitch. As the numbers of viewers peaked at 6,941,610 and the prize pool of League of Legends tournaments reached a total of $9,175,408 in 2024, the video game has become the most popular e-sports game (cf. Esports Charts 2025).

The hosting party Riot Games might search for ways to elevate visitor satisfaction. One method which could account for more entertainment in the *MOBA* (multiplayer online battle arena) video game e-sports industry would be the implementation of real time win-probability predictors (cf. Zhong 2024), as viewers are more engaged into the game itself and can interpret game outcomes in a more precise way. However, prior to creating such a model, the evaluation of large datasets, containing game variables, is required to classify supplementary data, as these models must develop a certain familiarity with the information extracted from existing games. These datasets were created during this study by using the developer tools provided by Riot Games.

This work was carried out to examine different Machine Learning algorithms and analyze *MOBA* games with respect to data values from both relative and absolute timestamps. Hereby, Machine Learning refers to a subdomain of Artificial Intelligence. The computation of all model accuracies enables the creation of a blueprint that contains the high performing Machine Learning models which can be implemented into the win predictor. Furthermore, a Non-Machine Learning algorithm capable of classifying games to different game outcomes was developed. After computing training time and accuracy, the results of the algorithm were compared to other Machine Learning models to see whether this algorithm could be a viable option for the win predictor.

As different Machine Learning models are utilized to process different datasets, the following questions arise: Which Machine Learning model returns the highest result in terms of accuracy? How does a change in the current time interval or timestamp affect the accuracy of the model? How does the algorithm, developed in this work, perform when compared to other Machine Learning models?

Regarding the second question, the assumption that accuracy will rise as elapsed time increases has been made. The reason for this assumption lies in the increasing availability of in-game data. Also,

as dataset size increases, the accuracy of the Machine Learning models and the self-made algorithm would presumably rise as well, since a higher number of resources are being provided.

2 Background

This section firstly covers basic definitions such as AI and the game type *MOBA* itself. This serves the purpose of acquiring a foundational overview about the machine-learning models, which will be tested and evaluated in this work. Later, the Riot API, provided by the video game developer *Riot Games*, will be inspected, since it plays a crucial role in data acquisition.

2.1 Artificial Intelligence

Artificial Intelligence (AI) has the capability of automating mentally sophisticated tasks and should be able to think and make decisions like humans. One of the first AI models was trained to learn the rules of chess to play against real players and eventually beat them. In 1997, the chess model "Deep Blue" defeated former World Chess Champion *Garry Kasparov* in a six-game match and set a milestone in AI history (cf. Campbell, Hoane, and Hsu 2002). While these kinds of models turned out to be rather useful, all of them were incapable of classifying images or recognizing languages (cf. Chollet 2018, 22). During this time, scientists were developing new approaches to create an AI capable of having thought patterns similar to the human mind, which eventually resulted in Machine Learning and later Deep Learning. Machine Learning will be relevant later on when certain algorithms are applied to the dataset, extracted through the Riot API (more in chapter 3.1).

As time has passed, Machine Learning models have enabled AI to compute non-linear relationships between certain variables, opening new possibilities in different fields (cf. Emmert-Streib, Yli-Harja, and Dehmer 2020) such as robotics (cf. Brooks 1991), image classification, (cf. Krizhevsky, Sutskever, and Hinton 2012) finance (cf. Bahrammirzaee 2010), and eventually video game win prediction. As for robotics, some AIs are already being implemented in the field of robotic surgery and would require image classification to perform a surgery (cf. Knudsen et al. 2024). In the financial field, the implementation of AI can be found in credit evaluation, portfolio management or financial planning/predicting (cf. Bahrammirzaee 2010).

Nevertheless, the word Artificial Intelligence has no clear definition (cf. Monett and Lewis 2018) and lacks clarity up to this date. Although its denotation always changes over time, many larger computer science communities agree on identifying AI as the techniques developed during its progression and the subdomains it comprises (cf. Wang 2019). This includes, among other things, Machine Learning

and Deep Learning.

2.1.1 Machine Learning

While the domain of AI remains unclear, this work mainly focuses its analysis on its subdomain Machine Learning which can be subcategorized into three different sections:

- **Supervised Learning:**

  Supervised Learning describes the training phase of a model, where labeled datasets are provided to enable prediction and classification. The model basically receives the solution and has to interpret additional data in its own way. Since this method is highly dependent on labeled data, it is often referred to Learning with a Teacher (cf. Haykin 1999, 85), Learning from Labelled Data (cf. Liu and Wu 2012), or Inductive Machine Learning (cf. Kotsiantis, Zaharakis, Pintelas, et al. 2007).

- **Unsupervised Learning:**

  A model based on Unsupervised Learning has to work with unlabeled data to predict or classify certain outcomes. There are many reasons why this method can play a crucial role in Machine Learning. These may include the large amount of unlabeled data which is available, the avoidance of data tagging, or the investigation of unknown or unprocessed data (cf. Naeem et al. 2023).

- **Reinforcement Learning:**

  Lastly, the Reinforcement Learning method emphasizes on letting the model iterate through a certain test case and returns feedback after each repetition, so that it can eventually return the desired output (cf. Khaleel, Jebrel, and Shwehdy 2024). For example, if the model performs poorly, the system punishes the learner for taking certain actions (cf. Sutton and Barto 2018, 21), while correct actions result in receiving a reward signal. In general, these signals are the key components for achieving a certain goal (cf. Sutton and Barto 2018, 7).

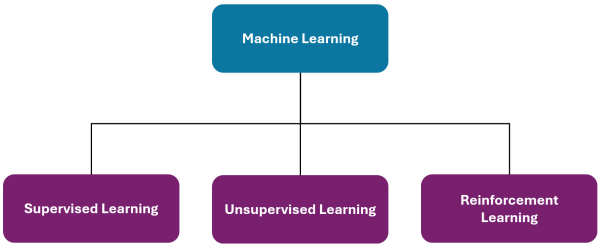


Figure 1: Subcategories of Machine Learning

2.1.1.1 More about Supervised Learning

When a Supervised-Learning-based model receives a dataset, it reads the input, also known as independent variables, and investigates the effects it has on the output, also known as the dependent variables (cf. Hastie, Tibshirani, and Friedman 2001, 9). During the training phase, both independent and dependent variables are labeled and give precise information about their values or categories, whereas in the testing phase, the model does not know the values of the independent variables and needs to predict or classify them based on the independent variables (cf. Liu and Wu 2012). The Machine Learning models and algorithms that were used in this evaluation for *MOBA* win prediction all fall under the Supervised Learning category.

2.1.1.2 Implemented Machine Learning models

Since win predicting models involve categorizing data samples into one out of two outcomes, classifying algorithms are seen as an irrevocable tool. Classification algorithms are capable of telling whether the blue or red team will win, based on only looking at in-game data. The scikit-learn library (cf. Pedregosa et al. 2011) contains a myriad of Machine Learning models, which are specialized in classification. Following Machine Learning models or algorithms were used in this study to predict game outcomes: Logistic Regression, Decision Tree, Random Forest, K-Nearest-Numbers, Naive Bayes, Support Vector Machine, Neural Network, Ada Boost, Gradient Boosting. An explanation of each model follows in chapter 4. Additionally, another algorithm based on the Pearson correlation has been developed to see whether it can return better outcomes than other Machine Learning models.

2.1.2 Misconceptions about AI

Although the word intelligence is included in the term AI, it does not have the same structure as the human brain. This also applies to Neural Networks, which were inspired by physiological neural models like the human brain (cf. Emmert-Streib, Yli-Harja, and Dehmer 2020) and belong to the subdomain of Machine Learning (cf. Chen et al. 2019). As for the methods from AI and Machine Learning, the purposes of these two are not different from each other and maintain the objective of analyzing data (cf. Emmert-Streib, Yli-Harja, and Dehmer 2020).

2.2 *MOBA* video games

The Multiplayer Online Battle Arena (*MOBA*) video game genre belongs to a subcategory of real-time strategy games and has gained a vast amount of popularity over the past decade (cf. Ye et al. 2020). The setup consists of two teams, consisting of five players, who compete against each other with the goal of invading the enemies' territory and destroying their main structure, which is situated on the opposite side of the map. Furthermore, each player has the opportunity to select an in-game character, granted with special abilities, varying from character to character, and can improve their overall statistics by increasing their character level through eliminating enemy players, creeps (serve as another assisting unit) or neutral creatures (cf. Cannizo and Ramírez 2015, 99–100). Creeps are assisting units that attack enemy players or other creeps and appear in a given interval. A typical *MOBA* game map always contains these four areas:

- **River:** divides the map into two parts
- **Base:** covers a small area on the bottom left or top right of the map to keep residing players safe
- **Lane:** allied or enemy creeps move to the opposite bases and attack any hostile player or creeps
- **Jungle:** split into four equally large parts

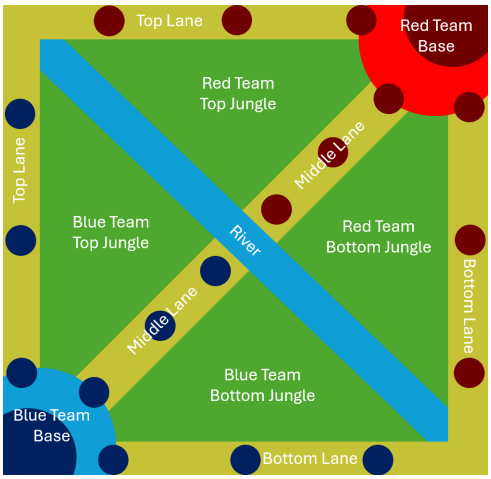


Figure 2: *MOBA* map (smaller dots representing each team's turrets)

Additionally, each lane, being either on the top, the middle or the bottom, has three turrets, guarding the center of the base. Each team must destroy the enemy turrets first to reach the main structure and win the whole game (cf. 102–103). Regarding the *MOBA* video game genre: AI does not only find its implementation in win prediction. The field of its use is broad and comprises intelligent NPCs (non-player characters), game balance and adaptability, and the improvement of player gaming experiences (cf. Zhong 2024). As for intelligent NPCs in *MOBA* matches, OpenAI has already set up a team of five in the DOTA 2 scene called *OpenAI Five*. The in-game characters of the team, built up by OpenAI, are all exclusively controlled by AI. In 2019, *OpenAI Five* was the first AI system to ever beat the word champions (Team OG) at an e-sports game (cf. OpenAI et al. 2019). Just like League of Legends, DOTA 2 falls under the *MOBA* category and is a real-time strategy game. Concerning the game balance and player experience, AI could play a crucial role in finding imbalances and suggesting

optimization steps (cf. Zhong 2024) to leverage user satisfaction. However, developers must handle the use of AI with care, as improper use of AI techniques can make it challenging to improve player experience (cf. Schwab 2009).

2.2.1 League of Legends

In 2009, the game development company Riot Games launched the *MOBA* video game League of Legends. The main structure, being protected by inhibitors and turrets, where one inhibitor and three turrets are placed on each lane, is called the nexus (cf. Junior and Campelo 2023). The most impactful jungle monsters, Baron Nashor and Dragon, can award both teams with different perks if being defeated, thus leading to another advantage for the executor of the jungle objective (cf. Riot Games 2024). Furthermore, League of Legends contains 170 different champions (as of January 2025), which are all equipped with four unique abilities, while the play style of each champion is categorized into either fighter, mage, marksman, tank or support (cf. Junior and Campelo 2023). Each player has the option to select two out of ten secondary abilities which are available to all players. Depending on the team composition, the teams can gain a major advantage, if both players and champions synergize well with each other. Further details about the video game can be read on the official League of Legends website, provided by Riot Games[1].

2.3 APIs

An API (Application Programming Interface) is a type of software with the ability to facilitate communication between the server and the client. Here, the server is responsible for storing information and providing the API, while the client acts as the customer and fetches data from the server through the API (cf. Cooksey 2014, 7–8). A server is essentially a powerful computer designed to handle greater computational tasks. It provides services to other programs or devices, known as clients, which rely on the server for resources or data processing (cf. Dermaku, Demaku, and Bajrami 2013, 1). As an example, the client would send a request to the server to obtain information about video game matches.

2.3.1 Server-client communication

When information is exchanged between client and server, both must adhere to a set of rules which are written in so-called protocols. There are various types of network protocols, each assigned to a specific function. One of the most used protocols on the web is the Hyper-Text Transfer Protocol

1. https://www.leagueoflegends.com/en-gb/how-to-play/

(HTTP), which instructs the browser to follow certain guidelines, declared by HTTP, when talking to the server (cf. Cooksey 2014, 11). In general, the client sends an HTTP request to the server and receives an HTTP response from it. Both request and response contain a certain structure including URL, method, headers and body or status code, headers and body respectively.

2.3.1.1 HTTP request structures

HTTP request structures usually include a URL as the first building block. URLs (Uniform Resource Locators) and URNs (Uniform Resource Names) are sub-terms of the umbrella term URI (Uniform Resource Identifier). The URI serves as a unified method for identifying resources, either by location (similar to the URL) or name (similar to the URN) (cf. Fielding et al. 1999). The fact that all URI, URL and URN are a string of characters must be taken into consideration. URNs on the other hand are used to identify resources based on their names. In general, URLs are preferred over URNs, since these are inefficient due to the potential of mutual names of different resources (cf. Sollins and Masinter 1994). Instead of recognizing the resource by its name, the URL uses the location to pinpoint the targeted resource. Many URLs always consist of two required components, namely the protocol and the domain (cf. Berners-Lee, Masinter, and McCahill 1994). Moreover, the address can encompass supplementary components, also included in APIs, as a means of passing on additional parameters.

The intent of the resource is induced by the HTTP method. The most common functionalities in an API are GET, POST, PUT and DELETE, forming the baseline of the methods block in an HTTP request (cf. Cooksey 2014, 13). The subsequent table shows the purpose of each function:

Method	Functionality
GET	retrieve information in form of an entity
POST	add resources to the server
PUT	modify current version of the origin server by updating enclosed entities
DELETE	remove resources from server

The headers of an HTTP request contain the metadata about the request. These include general information about the client and details on how the server should handle the request. Common types of headers in an HTTP request are "Accept" and "Authorization", allowing us to process data formats and basic authentication (cf. IONOS 2024). Some APIs also operate with the header "X-API-Key" to handle API keys so that the server can authenticate the client. This component will be essential for accessing the Riot API.

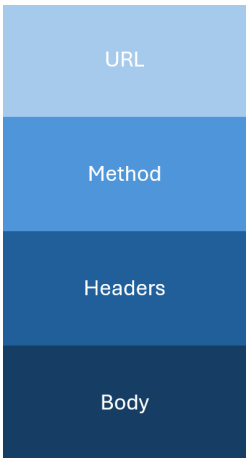


Figure 3: Request structure

An HTTP request body contains additional data which the client wants to transmit to the server. It is used for sending more substantial information, if required, with the unique trait of allowing the client to deliver anything it needs without being confined to the rigid structure of HTTP (cf. Cooksey 2014, 15). However, the content type of the body must be specified in the header so that the server can understand the body content.

2.3.1.2 HTTP response structures

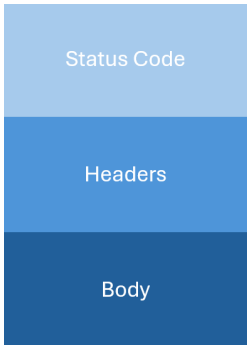


Figure 4: Response structure

HTTP response and request structures almost resemble the same architecture by having headers and body components. In addition to containing headers and a body, the HTTP response is defined by the inclusion of a status code.

Status codes are always included in the HTTP response and are made of three digits. Only the response codes starting with either two, four or five will be relevant when accessing the Riot API (and are also the only ones being mentioned in the Riot API documentation). Status codes, belonging to the type of 2xx, indicate the success of receiving, processing and accepting the client's request. 4xx status codes on the other hand signalize that the client has erred while 5xx response status codes hint out to a server error (cf. Fielding et al. 1999).

2.3.2 API endpoints

Key components of a well-thought API design include the creation of well-structured API endpoints. These endpoints contain URL patterns, which then are being added to the base URL (cf. Cooksey 2014, 48). Those URL patterns often can be found under the documentation of the API.

3 The dataset

Three different datasets, made of five csv files, were created in the process of evaluating Machine Learning models. These adhere to the blueprint of an online provided data table, which contains the information of 64,556 games (cf. Silva Junior and Campelo 2023). It is important to note that the blueVoidGrubKill and redVoidGrubKill variables have been added to the updated dataset, since these did not exist in previous game versions.

Variable	Description	Data Type
Column1	Associated row number	Integer
matchID	Identification string of the corresponding match	String
fullTimeMS	Game duration in milliseconds	Integer
timeStamp	Last timestamp of data collection (in milliseconds)	Integer
blueChampionKill/redChampionKill	Elimination number of blue/red team	Integer
blueFirstBlood/redFirstBlood	Whether blue/red team has received first elimination	Boolean
blueDragonKill/redDragonKill	Dragons slain	Integer
blueDragonHextechKill/redDragonHextechKill	Dragons of category Hextech slain	Integer
blueDragonChemtechKill/redDragonChemtechKill	Dragons of category Chemtech slain	Integer
blueDragonFireKill/redDragonFireKill	Dragons of category Fire slain	Integer
blueDragonAirKill/redDragonAirKill	Dragons of category Air slain	Integer
blueDragonEarthKill/redDragonEarthKill	Dragons of category Earth	Integer
blueDragonWaterKill/redDragonWaterKill	Dragons of category Water	Integer
blueDragonElderKill/redDragonElderKill	Dragons of category Elder slain	Integer
blueVoidGrubKill/redVoidGrubKill	Number of Void Grubs eliminated	Integer
blueRiftHeraldKill/redRiftHeraldKill	Number of Rift Heralds eliminated	Integer
blueBaronKill/redBaronKill	Number of Barons eliminated	Integer
blueTowerKill/redTowerKill	Towers destructed	Integer
blueInhibitorKill/redInhibitorKill	Inhibitors destructed	Integer
blueTotalGold/redTotalGold	Amount of Gold acquired	Integer
blueMinionsKilled/redMinionsKilled	Creep score	Integer
blueJungleMinionsKilled/redJungleMinionsKilled	Number of eliminated non-epic jungle monsters	Integer
blueAvgPlayerLevel/redAvgPlayerLevel	Total sum of all player levels in blue/red team (does not imply average level)	Integer
blueWin/redWin	Wether blue/red team has won the game	Boolean

Table 1: Game Variables (Epic jungle monsters: All dragon types, Void Grub, Rift Herald, Baron)

The first data collection comprises game-state data from 60,435 different games where each csv file corresponds to a specific percentage of elapsed time: 20%, 40%, 60%, 80% and 100%. The game information of the second dataset correlates with a specific timestamp, which accounts for a smaller difference in quantifiable data values. Additionally, the second dataset includes information about 25,082 games and marks its timestamps at 10 minutes, 14 minutes, 20 minutes, 27 minutes and the last minute. The last dataset follows the same principles as the first data table, except that it contains mutual game IDs with the second dataset to examine the impact of data quantity on model accuracy.

The reason why another dataset including timestamps has been created lies in seeing whether the timestamp method delivers results superior to those of the time interval method as this method has been suggested by Jailson Junior and Cláudio Campelo[2].

3.1 Creating the dataset through the Riot API

Accessing games created by Riot Games, such as League of Legends, Teamfight Tactics or Valorant requires the Riot API. It provides developers with match statistics, player details, results, and other relevant data, necessary for the dataset (cf. Junior and Campelo 2023, 3). The API can only be accessed through an API key which can be easily generated on the official Riot Developer website after registration. The HTTP request structure must include the required parameters in the URL and the API-key in the headers section to make an API call. The URL usually contains the endpoint as part of the resource. Also, the value 'application/json' has been passed over as the 'Accept' parameter, indicating that the client expects the format of the response body to be in JSON (JavaScript Object Notation) format. Furthermore, each endpoint on the API provides a different type of information, which is included in the body section of the HTTP response structure. The dataset creation includes various endpoints such as League V4, Summoner V4 and Match V5. In the first step, player puuids were attained through the League V4 and Summoner V4 endpoints and subsequently stored into a txt file. The puuids from before were being used for matchID extraction, which later on served as the key for Match V5 endpoints such as `/lol/match/v5/matches/matchId/timeline`. In the making of the three datasets, requests of the data from four regions (America, Asia, Europe, SEA) have been made and subsequently imported into a csv file via the csv python library.

4 Machine Learning model descriptions

This section aims to elucidate the maths and concepts behind the Machine Learning models. As mentioned before, all models and algorithms are provided by the scikit-learn Python library except for the *PCCBA*. This algorithm will receive its own section (chapter 6) in which the derivation of the formula for the algorithm will be thoroughly explained. Subsequently, this algorithm will be compared to the other Machine Learning models to find out how well it performs.

4.1 Logistic Regression

Logistic Regression is a binary classification algorithm which uses data points to predict boolean values (cf. Dayananda 2023). Hereby the model searches for a relationship between independent and

2. URL: https://arxiv.org/pdf/2309.02449

dependent variables without being confined to $\mathbb{R}^2$.

The logit function plays a crucial role in mapping the input to the output. This function maps a probability π to the log-odds of an event occurring, represented as $\frac{p}{(1-p)}$. The reason for the implementation of the logarithm lies in the creation of a symmetry in the odds ratio.

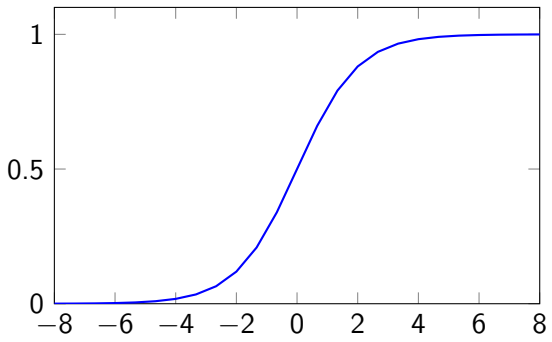
$$\pi' = (1 - \pi)$$

$$\lim_{\pi' \to \infty} \frac{\pi}{\pi'} = 0 \wedge \lim_{\pi \to \infty} \frac{\pi}{\pi'} = \infty$$

$$P(\pi) : \log\left(\frac{\pi}{\pi'}\right) = -\left(\log\left(\frac{\pi'}{\pi}\right)\right)$$

This logit function is crucial because it maps the probability space, lying between 0 and 1, to the real number line, enabling the model to apply linear regression techniques. Since the model needs to output a probability that indicates whether the dependent variable is true or false, it will have to work with the inverse of the logit function, also known as the sigmoid function.

$$\textbf{Sigmoid function: } \sigma(x) = \frac{1}{1 + e^{-x}}$$



While training, the Logistic Regression model uses the logit function to generate a linear combination z made of a bias term β_0 and a weighted sum of features (cf. Hastie, Tibshirani, and Friedman 2001, 119) where each weight β_i is assigned to its corresponding feature x_i:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$$

It is important to note that z equals the output of the logit function. After fitting the dataset to the model, the probability of an event occurring can be calculated through inserting the linear combination into the sigmoid function (cf. Dayananda 2023):

$$P(y = 1|X) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

4.2 Decision Tree

A Decision Tree has the ability to complete classification and regression tasks. Its structure follows a tree structure where each output value ends at a so-called leaf node. Every leaf node is connected to an internal node, representing the parent node and the first node which has access to every leaf is called the root node (cf. Ibm 2024a).

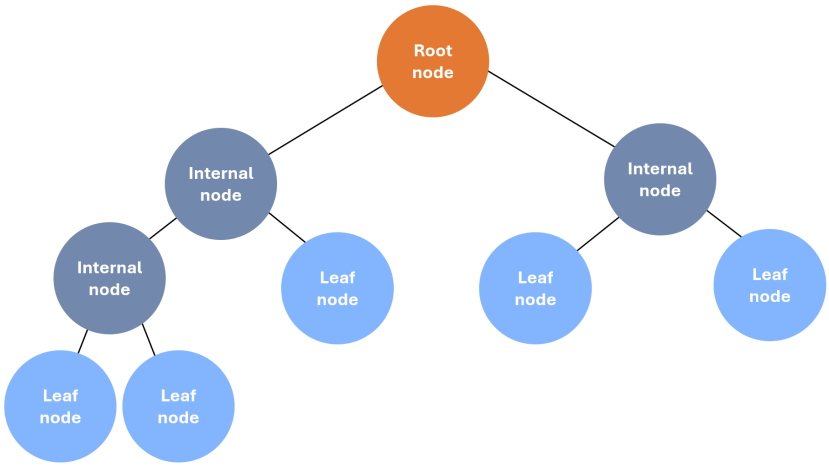


Figure 5: Example of Decision Tree structure

In a Decision Tree, each root or internal node has to establish a condition to classify the data. For example, one query which a node could apply could be whether a variable is true or false. After determining its boolean value, the model compares the input value with the output value (for example blueWin) from the training dataset and creates either two leaf nodes, two internal nodes or one leaf and one internal node to facilitate classification. An empty quantity in a leaf node implies that the node is pure, while a more even distribution among the quantities indicates that the node it impure. However, it will be necessary to quantify the impurities of each leaf to create a certain tree structure. The scikit-learn library implements the Gini Impurity formula to calculate and compare those impurities (cf. Pedregosa et al. 2011).

$$\textbf{Gini Impurity: } 1 - \sum_i (p_i)^2$$

In this formula i represents the different classes (cf. Ibm 2024a). Since the Decision Tree only outputs a boolean value, the function can be simplified to following equation:

$$\text{Gini Impurity} = 1 - \left(\frac{\text{true variables}}{\text{true variables + false variables}}\right)^2 - \left(\frac{\text{false variables}}{\text{true variables + false variables}}\right)^2$$

As for the variables with numeric values, the model calculates the Gini Impurity of all pairwise combination averages. Values with the lowest impurities are then chosen to form the condition of a node. In the making of the Decision Tree structure, the model assigns each node with the lowest total Gini Impurity to its parent node. The total Gini Impurity of a node can be calculated through the sum of its leaf's Gini Impurities (cf. Karabiber 2024).

4.3 Random Forest

The Random Forest model combines multiple Decision Trees to overcome the risk of overfitting. Decision Trees tend to be confined to only one set of data and often face limitations when it comes to handling new information, as is commonly observed in the aforementioned overfitting (cf. Ibm 2024a). With Random Forest this problem can be overcome by creating subsets of the data and assigning one to its corresponding tree. Also, Decision Trees of a Random Forest follow a different set of instructions during their creation. The corresponding data sample, also known as bootstrap sample, is hereby a subset of the whole dataset, where specific rows may occur multiple times (cf. Ibm 2024d). While training, the model randomly chooses one variable from a set of featured variables, which later on serves as a criteria for the internal node. Before training the model, the following parameters must be set: node size, number of trees and the number of features. The scikit-learn library usually sets the parameters to some default values, if the user does not pass on any parameter values into the function (cf. Pedregosa et al. 2011). If the majority of trees output a particular class (probability of the class is highest), the model will return that class (cf. Ibm 2024d). The following image shows an example of a Random Forest structure[3]:

3. Source: https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d

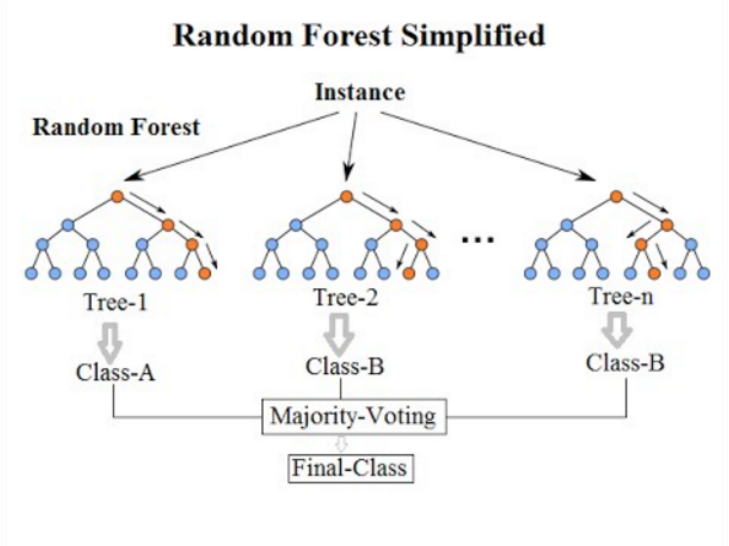


Figure 6: Example of Random Forest structure

4.4 K-Nearest-Neighbor

The K-Nearest-Neighbor (KNN) algorithm maps each row of a dataset to a point in multidimensional space and assigns it to a specific class, such as true or false for blueWin. Before training the model, the parameter k, an odd integer, must be defined (cf. Ibm 2024f). If new rows of data, made of independent variables only, are being added, KNN projects the new input onto a point and calculates the distances between the point and other data points which already contain position and label. In general, the scikit-learn library utilizes the Minkowski distance to compute such distances (cf. Pedregosa et al. 2011). In the following function, A corresponds with the existing point, while B corresponds with the inputted point.

$$\textbf{Minkowski distance: } d(A,B) = \left(\sum_{i=1}^{n} |A_i - B_i|\right)^{1/p}$$

After computing the distances, KNN searches for k points, which are closest to the input value and assigns it to the class where most of the neighbors are in (cf. Ibm 2024f).

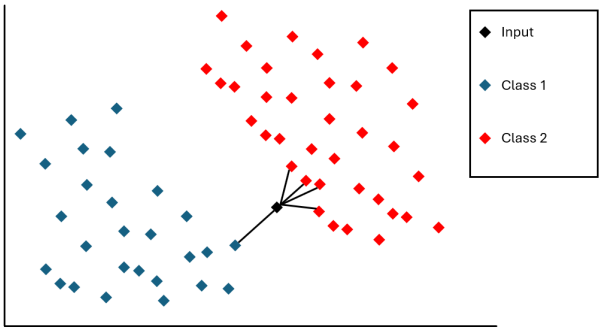


Figure 7: Input being assigned to class 2 (KNN)

4.5 Naive Bayes

The Naive Bayes algorithm is based on applying Bayes' Theorem (cf. Webb, Keogh, and Miikkulainen 2010) and is often used for classification tasks by using probability principles. Bayes' Theorem states that the probability of an event A, given that the event B has already occurred, can be expressed by only using the probability of A and B, and the probability of B, given that A has occurred.

$$\textbf{Bayes' Theorem: } P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

If we portray the independent variables in the given dataset as a vector from x_1 to x_n, we receive following function (knowing that $P(x_1, x_2|y) = P(x_1|y) \cdot P(x_2|y)$ according to the conditional independence assumption (cf. Phillips 1988)):

$$P(y|x_1, ..., x_n) = \frac{P(y) \cdot \prod_{i=1}^{n} P(x_i|y)}{P(x_1, ..., x_n)}$$

Here the probability of y represents the win probability of the blue team and is included in the test dataset, while the x variables constitute the independent variables. Additionally, category classification of game samples requires the highest probability of a given condition (cf. Rish 2001). Since $P(x_1, ..., x_n)$ is a constant which lies in the denominator, it can be excluded out of the equation to get the highest probability.

$$\hat{y} = \arg\max_{y} \left(P(y) \cdot \prod_{i=1}^{n} P(x_i|y) \right)$$

Here, variable $\hat{y}$ returns the class where the probability of an event to occur is highest.

4.5.1 Gaussian Naive Bayes

The Gaussian Naive Bayes, a deviation of the Naive Bayes classifier, was utilized in this study and makes use of the Gaussian normal distribution, which allows the model to work with and predict continuous values. The formula embodies the following form (cf. Peretz, Koren, and Koren 2024):

$$P(x_i|y) = \frac{1}{\sigma_y\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_i-\mu_y}{\sigma_y}\right)^2}$$

(σ = standard deviation of y, μ = mean or average of y)

4.6 Support Vector Machine

The Support Vector Machine (SVM) creates either a line or a plane in multidimensional space to separate different data points into classes. Here, the objective is to maximize the distances, also called margins, between the hyperplane and the closest data points, known as support vectors (cf. Ibm 2024e). When SVM allows misclassification or violation it refers to a soft margin. Otherwise, if there are no falsely classified observations between the support vectors, one would talk about a hard margin (cf. Marín et al. 2022).

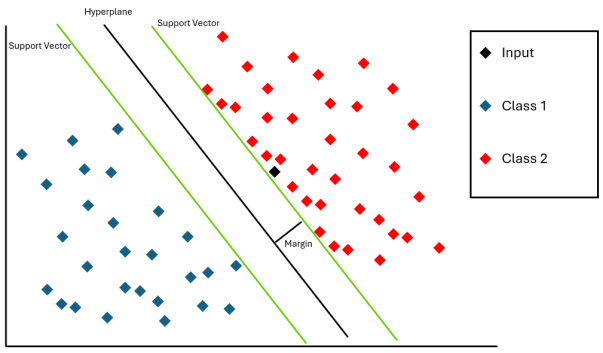


Figure 8: Input being assigned to class 2 (SVM)

Based on where a new observation is being placed, SVM assigns the new data point to a class according to which side of the hyperplane it falls on. However, if the SVM is incapable of drawing optimal support vectors, it uses a so-called kernel function to transform the data into higher dimensions. Scikit-learn usually sets the kernel of its SVC (Support Vector Classifier) to rbf (radial basis function) by default (cf. Pedregosa et al. 2011), which can work with data which overlaps in a certain dimension. The function for the rbf takes two observations, namely a new one and a given data point, as the input and places them into the exponent of the natural number e, where the squared distance of those two data points is being multiplied with the negative of a scalar γ (cf. Bernstein 2017).

Radial basis function: $e^{-\gamma(a-b)^2}$

Hereby, the rbf works similar to a weighted Nearest Neighbor model, since higher distances lead to a lower score in the functions output. Also, the rbf implements the Taylor series expansion to calculate the dot product of the vectors which represent the observations (cf. Bernstein 2017 Bernstein 2017).

$$\textbf{Taylor series: } \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

In order to implement the Taylor series into the radial kernel, multiplying out the square in the exponent is essential to transform e^{ab} into a sum. For the purpose of simplification, the scalar γ has been set to $\frac{1}{2}$. After setting x to ab, e^{ab} can be expressed in following way[4]:

$$e^{ab} = 1 + \frac{1}{1!}ab + \frac{1}{2!}(ab)^2 + ... + \frac{1}{\infty!}(ab)^{\infty}$$

The given Taylor series expansion of e^{ab} can be formed into a dot product where each vector comprises the coordinates of the multidimensional space.

$$e^{-\frac{1}{2}(a-b)^2} = \left(e^{-\frac{1}{2}(a^2+b^2)}, e^{-\frac{1}{2}(a^2+b^2)} \cdot \sqrt{\frac{1}{1!}}a, e^{-\frac{1}{2}(a^2+b^2)} \cdot \sqrt{\frac{1}{2!}}a^2, ..., e^{-\frac{1}{2}(a^2+b^2)} \cdot \sqrt{\frac{1}{\infty!}}a^{\infty}\right) \cdot \left(e^{-\frac{1}{2}(a^2+b^2)}, e^{-\frac{1}{2}(a^2+b^2)} \cdot \sqrt{\frac{1}{1!}}b, e^{-\frac{1}{2}(a^2+b^2)} \cdot \sqrt{\frac{1}{2!}}b^2, ..., e^{-\frac{1}{2}(a^2+b^2)} \cdot \sqrt{\frac{1}{\infty!}}b^{\infty}\right)$$

This equation[5] shows that the radial kernel is capable of transforming data points into higher-dimensional space, where the support vectors can then be initialized (cf. Bernstein 2017).

4.7 Neural Network

A Neural Network consists of multiple layers of nodes where each connection between two layers contains a parameter or weight value, which received its value during the fitting process (cf. Hastie, Tibshirani, and Friedman 2001, 395). The first layer of nodes is called the input nodes and take the

4. Equation 1 in Appendix
5. Equation 2 in Appendix

independent variables as an input. On the contrary, each node of the last layer, also called the output layer, outputs the probability of a given event occurring. The layers between the first and last layer are called hidden layers (cf. Ibm 2024b). When an input x_i passes through the input layer, it first is being multiplied with a weight w_i and then added to a bias b_m to then be passed into an activation function (cf. Davila 2020). Here, the scikit-learn library uses the ReLU function (rectified linear unit) as default in the MLP Classifier (cf. Pedregosa et al. 2011).

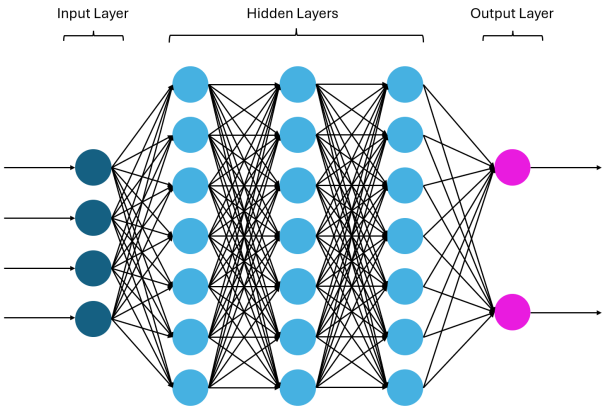
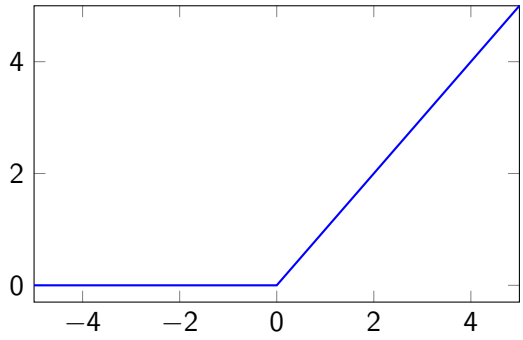


Figure 9: Neural Network structure

The figure above gives an example of a neural network structure with two output nodes. Here, one node could represent the probability of blueWin being true, whereas the other node might portray the probability of blueWin being false.

ReLU function: $\phi(x) = max(0, x)$



A node receives the computed weighted sum of activation values to apply the activation function on it to create a new function which usually deviates from the observations. Neural Networks implement a method called backpropagation to calculate the weights and biases. In this method the model first adjusts the values from the last layers and works its way forward. To calculate the bias of the last connection, the algorithm computes the sum of squared residuals (SSR) first, which can be derived from previous nodes (cf. Hastie, Tibshirani, and Friedman 2001, 395). The SSR is commonly referred

as the mean squared error (MSE) and represents the loss function L (cf. Ibm 2024b). Additionally, a residual is just the difference between an observation y_j and the predicted value of the previous node z_j where j is the number of observations.

$$\textbf{weighted sum: } b + \sum_{i=0}^{n} x_i w_i$$

$$f : x_f \rightarrow x_g, g : x_g \rightarrow \phi(x_g \cdot w_g + b_m), h : x_f \rightarrow \phi(x_g \cdot w_g + b_m)$$

$$z_j = b + \sum_{i=0}^{n} h(x_f) \quad |x = [x_1, x_2, ..., x_n]$$

$$\textbf{SSR: } \sum_{i=1}^{j} (y_i - z_i)^2$$

The optimal value for a bias in a Neural Network can be computed through minimizing the loss function of the error with respect to the bias $\left(\frac{\delta L}{\delta b}\right)$. The minimization of that value can be achieved by utilizing a method called gradient descent (cf. Hastie, Tibshirani, and Friedman 2001, 395). Gradient descent is an optimization algorithm which uses the derivative of the MSE with respect to b to find the optimal value of b. Since b lies in the predicted value which also lies in the MSE function, gradient descent uses the chain rule to compute the slope of each bias[6] in this implementation.

$$\textbf{Loss function: } L = \sum_{i=1}^{j} (y_i - z_i)^2$$

$$\textbf{Application of the chain rule: } \frac{\delta L}{\delta b} = \frac{\delta L}{\delta z} \cdot \frac{\delta z}{\delta b}$$

$$\implies \frac{\delta L}{\delta b} = \sum_{i=1}^{j} (-2) \cdot (y_i - z_i) \cdot 1$$

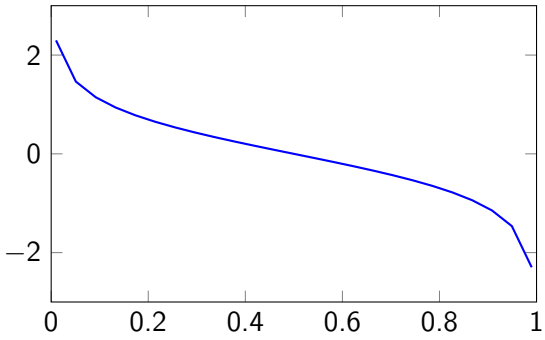
By changing b over and over, gradient descent can find the ideal value of b, which creates the lowest MSE, when plugged into the loss function.

6. Equation 3 in Appendix

4.8 Ada Boost

Just like the Random Forest algorithm, Ada Boost combines multiple trees, also referred as weak learners in this algorithm, to compute the output value. However, in contrast to the Random Forest trees, each tree only comprises one node and two leaves and is thus being called a stump. Initialization of boolean and numeric conditions adhere to Decision Tree instructions, while the impact on the output value, which is determined by the order of stump initialization, varies among every component. The stump that predicts the target variable best, which means that Gini Impurity is lowest, is selected as the first stump (cf. Lang 2024). Then, each data point is assigned a weight w_i, representing the reciprocal of the datasets size. Here, the sum of weights remains equivalent to 1. Summation of incorrect classification and corresponding sample weight products allow stump error rate (ϵ) evaluation and can subsequently be used for stump weight (α) computation (cf. William 2021; Lang 2024).

$$\alpha = \frac{1}{2} \cdot \log\left(\frac{1-\epsilon}{\epsilon}\right)$$



After α has been evaluated, modification of correctly and falsely classified data point weights follow (cf. William 2021):

New value of falsely classified w_i**:** $w_i \cdot e^{\alpha}$

New value of correctly classified w_i**:** $w_i \cdot e^{-\alpha}$

Then, the next stump in the algorithm is determined by the weighted Gini Impurity. After iterating through each component, the sum of α from one class is evaluated, and the category with the highest value determines the class of the added data.

4.9 Gradient Boosting

The Gradient Boosting Machine Learning algorithm can be used for either classification or regression. Just like Ada Boost, Gradient Boosting combines multiple trees with fixed size. However, each tree does not have to consist of two leaves and can thus handle multiple variables (cf. Ibm 2024c).

In the first step, the model creates an initial leaf which represents the initial probability prediction $F_0(x)$ for each data point by implementing a loss function L (cf. Masui 2024).

$$F_0(x) = \arg\max_{\gamma} \sum_{i=0}^{n} L(y_i, \gamma)$$

Here, the loss function takes the observations as input and tries to find the smallest number for γ. This can be achieved through setting the derivative of the loss function to zero (cf. Masui 2024).

In the next step, the residual r_{ij}, which is the difference of the observation and the predicted value, is calculated for each data point and is then assigned to a leaf, representing the set of residuals R_{ij}. Then, the initial prediction has to be added to the product of a learning rate η and the sum of the leaf output values γ_{jm}, to calculate the next probability of an event occurring. The leaf output value of a leaf can be again calculated through the loss function which takes the data points, belonging to a set R_{ij}, and the previous prediction as inputs (cf. Masui 2024).

$$\text{Leaf output value: } \gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma$$

$$\text{Next prediction probability: } F_m(x) = F_{m-1}(x) + \eta \cdot \sum_{j=0}^{J_m} \gamma_{jm} 1(x \in R_{jm})$$

The term $\sum_{j=0}^{J_m} \gamma_{jm} 1(x \in R_{jm})$ suggests that the data point x must be included in the leaf node, so that its value can be added to the sum (cf. Masui 2024). After computing the next prediction, gradient descent iterates through the same process multiple times to reach an accurate prediction of the dependent variable.

5 Procedure of Machine Learning model evaluation

As mentioned before, the Machine Learning models were provided by the scikit-learn Python library (cf. Pedregosa et al. 2011). Apart from scikit-learn, other libraries such as numpy (cf. Harris et al. 2020), matplotlib (cf. Hunter 2007), pandas (cf. McKinney 2010) and seaborn (cf. Waskom 2021) were implemented in a Jupyter Notebook environment to conduct the work in this thesis. After importing the libraries, pandas was used to convert the data of the csv files into a data frame to facilitate the declaration of independent and dependent variables. Also, the variable `blueWin` has been selected as the dependent variable. Regarding model training and accuracy computation, a 5-fold cross-validation was implemented, because it is one of the simplest and most popular methods to estimate prediction error (cf. Hastie, Tibshirani, and Friedman 2001, 241). In cross-validation, the dataset is split into K (in this case 5) roughly equal-sized parts. Here, one component will be used as the testing set while the union of the other sets serve as the training set (cf. Hastie, Tibshirani, and Friedman 2001, 242). After computing the accuracy of the model, another component serves as the testing set. Ultimately, the amount of testing sets will be equal to K.

It will be necessary to compute the mean of all accuracies to calculate the accuracy of a model. Let $\kappa : \{1, ..., N\} \mapsto \{1, ...K\}$ be a function that maps an observation to its partition, where N is the size of the dataset (cf. Hastie, Tibshirani, and Friedman 2001, 242).

$$\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{\sum_{i=1}^{N} 1_{\{k\}}(\kappa(i))}\cdot\sum_{i=1}^{N} 1_{\{\hat{y}_i\}}(y_i)\cdot 1_{\{k\}}(\kappa(i))\right)$$

The indication function $1_{\{k\}}$ checks if a data point is an element of the testing set while $1_{\{\hat{y}_i\}}$ checks whether an observation is equal to the dependent variable. Additionally, the product of the reciprocal of K and the first sum represents the average of the accuracies.

5.1 Fitting the dataset and training the model

Before applying cross-validation onto the dataset, a model has to be defined through a pipeline to handle the data. The `cross_val_score()` function from the scikit-library takes multiple parameters as input (cf. Pedregosa et al. 2011), including the pipeline, dependent and independent variables, number of splits and the scoring parameter that defines an evaluation metric such as accuracy.

After declaring the input parameters, training the model follows. The table below illustrates the amount of time required for training a model when providing a specific dataset and the size of each dataset. Additionally, a graph has been plotted to visualize the training time duration. The benchmark

was conducted using a AMD Ryzen 7 3700X 8-Core Processor.

ML Model and dataset size	1. Dataset	2. Dataset	3. Dataset
Logistic Regression	0min 9s 176ms	0min 1s 795ms	0min 2s 2ms
Decision Tree	0min 18s 781ms	0min 8s 147ms	0min 8s 502ms
Random Forest	4min 44s 179ms	2min 2s 869ms	2min 10s 67ms
K-Nearest-Neighbor	0min 19s 435ms	0min 7s 159ms	0min 7s 209ms
Naive Bayes	0min 1s 297ms	0min 0s 566ms	0min 0s 592ms
Support Vector Machine	45min 24s 618ms	8min 38s 829ms	8min 59s 51ms
Neural Network	19min 10s 762ms	12min 48s 592ms	13min 14s 243ms
Ada Boost	1min 11s 737ms	0min 32s 860ms	0min 30s 497ms
Gradient Boosting	5min 0s 597ms	2min 10s 238ms	2min 11s 466ms
Amount of games	60 435 games	25 082 games	25 082 games

Table 2: ML Model with accuracy at certain time interval

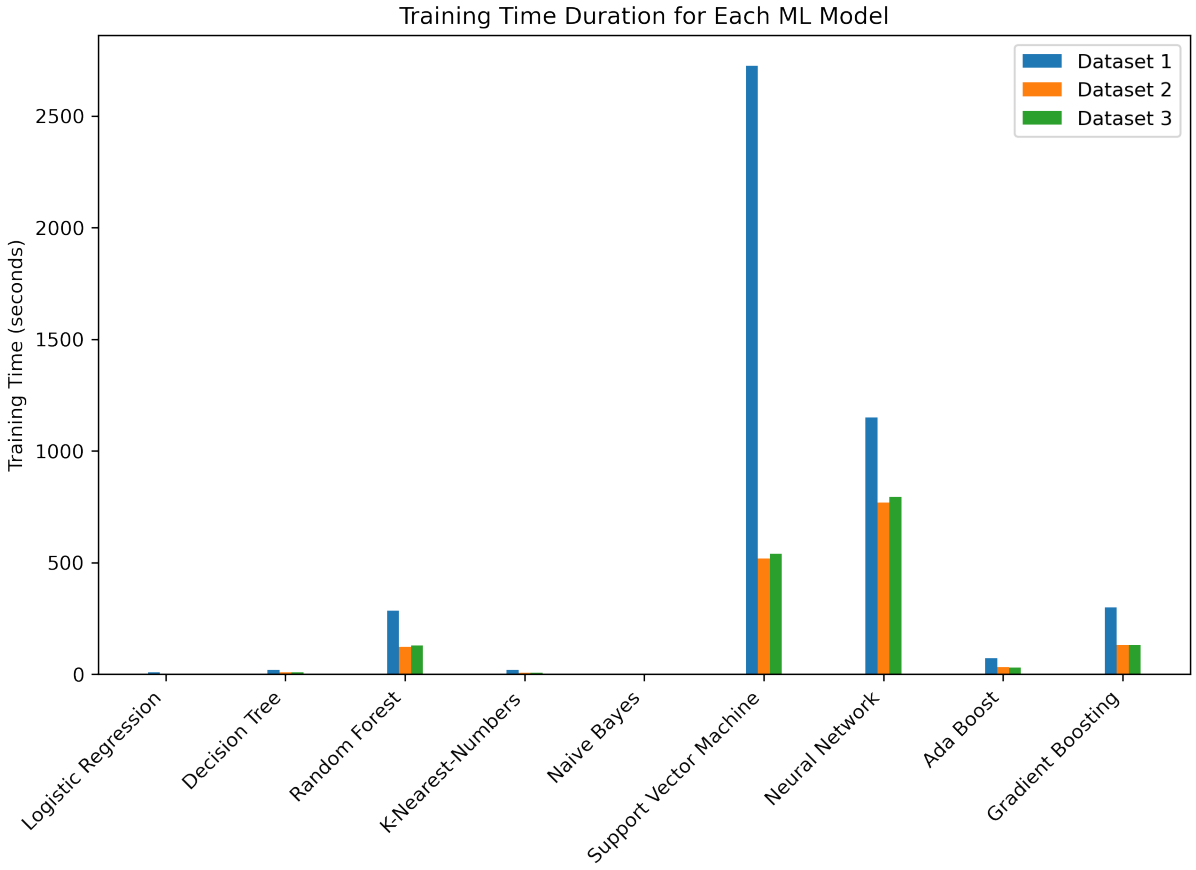


Figure 10: Training time plot

5.2 First time interval dataset accuracies

After providing each model with the first dataset, the following table, containing the accuracies with their corresponding interval and algorithm, has been made:

ML Model	20%	40%	60%	80%	100%
Logistic Regression	64.27%	71.84%	78.69%	87.27%	98.52%
Decision Tree	57.34%	64.50%	71.90%	81.95%	97.54%
Random Forest	63.23%	71.32%	78.50%	87.13%	98.55%
K-Nearest-Neighbor	59.62%	66.63%	73.14%	81.40%	95.09%
Naive Bayes	54.15%	61.13%	70.87%	81.44%	95.33%
Support Vector Machine	64.40%	71.97%	78.78%	87.16%	98.32%
Neural Network	63.30%	70.35%	76.71%	85.32%	98.57%
Ada Boost	63.09%	69.58%	76.37%	85.04%	97.49%
Gradient Boosting	64.00%	71.35%	78.27%	86.73%	98.34%

Table 3: ML Model with accuracy at certain time interval

As can be seen in the table above, the accuracy of every model at the last interval stood out and exceeded the 95% accuracy mark. Additionally, accuracy of each training model increases by time which indicates that the models were functioning, since adding more information about the state of the game lead to a more reliable prediction. After mapping each time mark to its corresponding accuracy, derivation of the visualized graph follows:

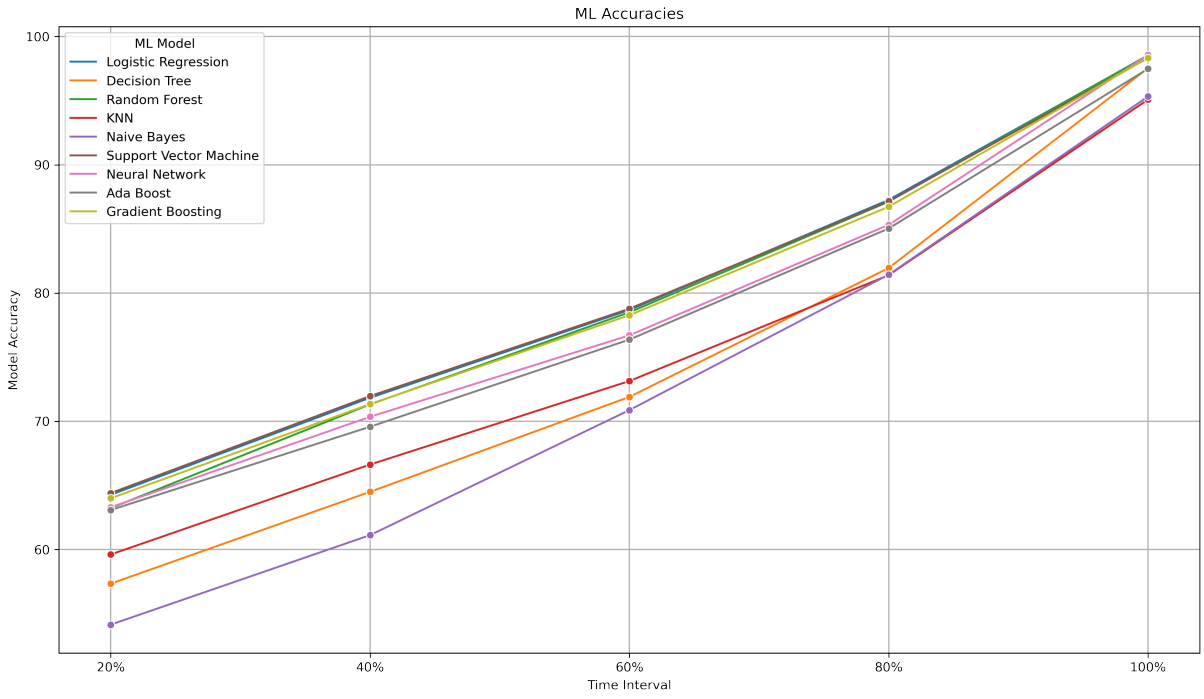
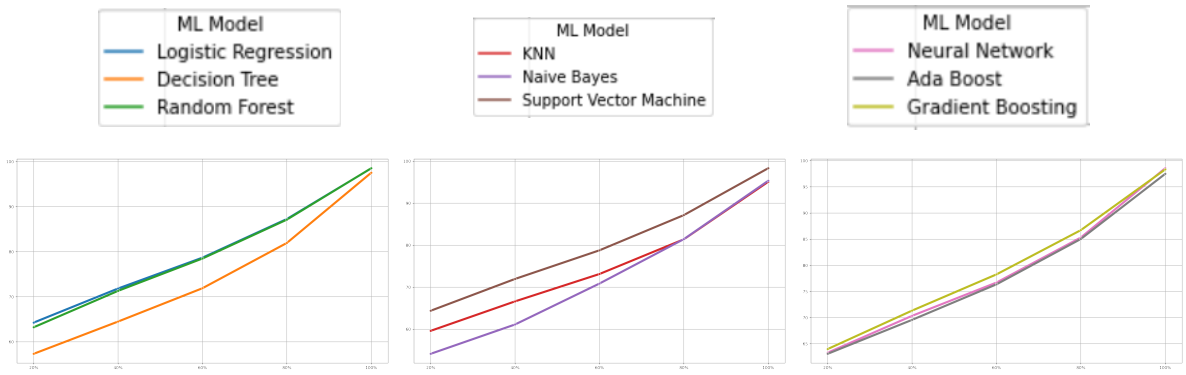


Figure 11: First dataset accuracies



Throughout the whole training process, Logistic Regression and SVM maintained their high accuracy. Gradient Descent also performed well and scored only slightly lower overall. Random Forest, Ada Boost and Neural Network delivered a moderate performance, when only 20% of the game data was provided. However, Neural Network reached an accuracy of 98.57% when trained on 100% of the data and performed best, together with Logistic Regression, Random Forest, SVM and Gradient Boosting models which achieved accuracies of 98.52%, 98.55%, 98.32% and 98.34% respectively. It is essential to acknowledge that the Neural Network only contains two hidden layers, consisting of 64 and 32 nodes respectively, and can therefore be extended to achieve better outcomes. Models which worked worse and started with an accuracy lower than 60 % were KNN, Decision Tree and Naive Bayes. Nevertheless, Decision Tree ended up having similar outcomes as Ada Boost, reaching a score of 97.54%. Although Naive Bayes started of with the worst results, it eventually took over KNN and scored slightly better in terms of accuracy as both models attained 95.33% and 95.09% each.

5.3 Timestamp dataset accuracies

The second table introduces the outcome of the second dataset which contains the accuracies measured at the different timestamps represented by minutes elapsed.

ML Model	min 10	min 14	min 20	min 27	end
Logistic Regression	59.75%	63.02%	69.09%	81.70%	98.53%
Decision Tree	52.50%	54.33%	59.19%	73.02%	96.03%
Random Forest	58.11%	62.04%	68.46%	81.25%	97.78%
K-Nearest-Neighbor	54.25%	56.86%	62.27%	77.41%	94.21%
Naive Bayes	55.95%	59.48%	67.04%	80.34%	94.44%
Support Vector Machine	58.79%	62.77%	68.91%	81.44%	98.19%
Neural Network	54.23%	56.83%	61.99%	75.20%	98.16%
Ada Boost	59.20%	62.24%	69.10%	81.32%	97.69%
Gradient Boosting	59.41%	62.73%	69.16%	81.54%	97.79%

Table 4: ML Model with accuracy at certain minute

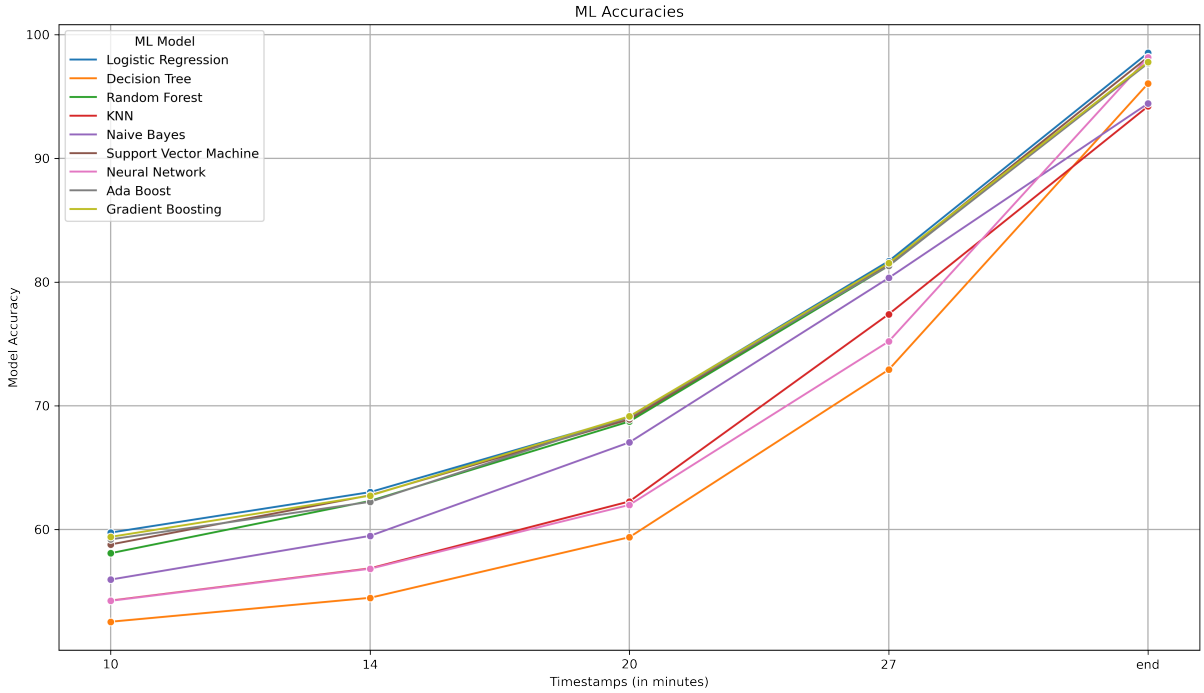
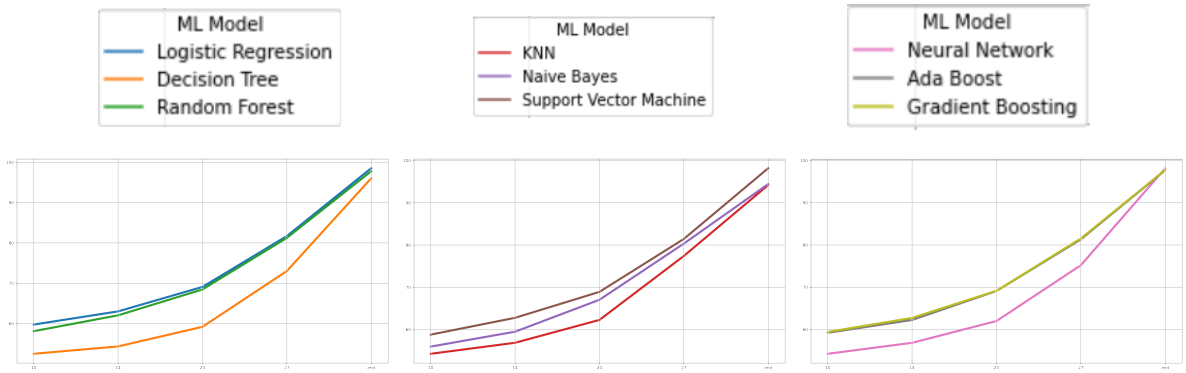


Figure 12: Second dataset accuracies



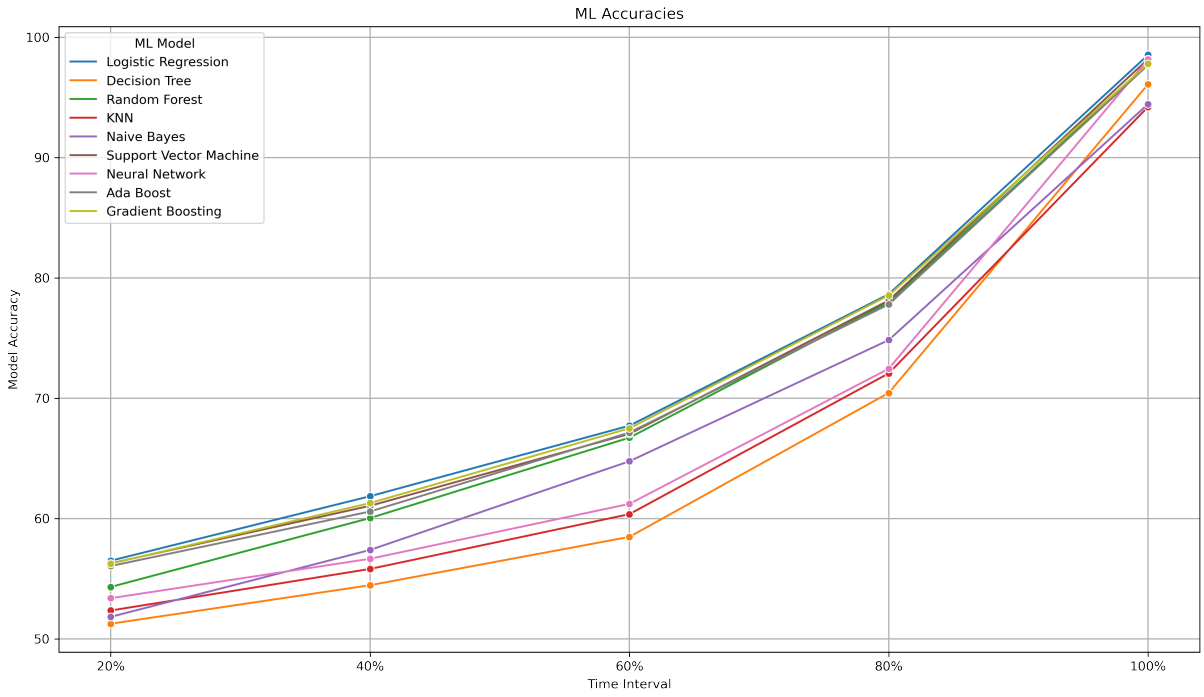
In the second dataset, Logistic Regression still performs best and achieves relatively high scores in comparison to other algorithms. However, all models produce an accuracy lower than 60% at the 10-minute timestamp. In accordance with previous findings, Machine Learning models such as Logistic Regression, Random Forest, SVM and Gradient Boosting outperform other models together with Ada Boost. Also, Naive Bayes' performance rests above Decision Tree, KNN and the Neural Network, but slows down after the 27-minute mark and ultimately converges to an accuracy level comparable to that of KNN. Furthermore, Decision Tree ends up at a higher success rate than KNN and Naive Bayes, after performing worse in the previous minute marks. On top of that, the Neural Network outputs a relatively lower score in the first four time stamps and end up spiking at 98.16%.

5.4 Second time interval dataset accuracies

From the last table, which is based on the dataset containing time intervals of the identical games included in the second dataset, the following results can be deduced:

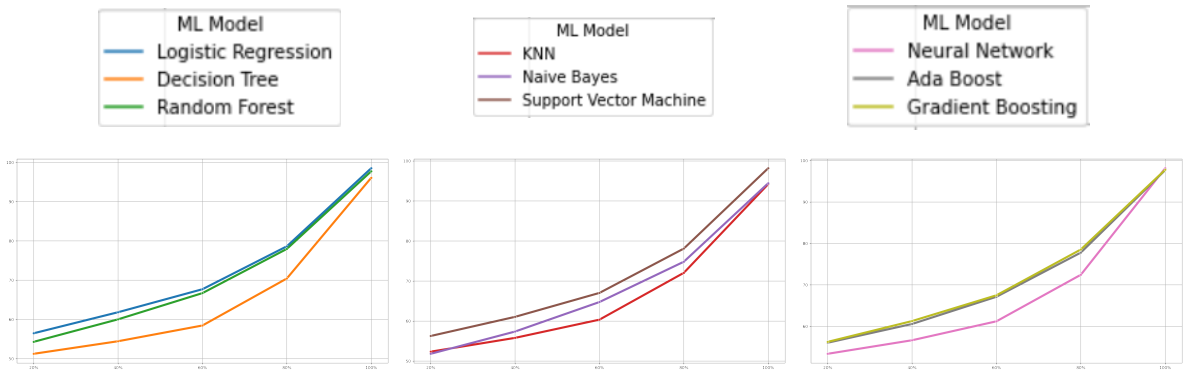
ML Model	20%	40%	60%	80%	100%
Logistic Regression	56.49%	61.85%	67.71%	78.64%	98.53%
Decision Tree	51.26%	54.46%	58.47%	70.44%	96.09%
Random Forest	54.31%	60.04%	66.72%	77.97%	97.74%
K-Nearest-Neighbor	52.36%	55.81%	60.36%	72.06%	94.21%
Naive Bayes	51.83%	57.39%	64.76%	74.83%	94.44%
Support Vector Machine	56.28%	61.06%	67.04%	78.12%	98.19%
Neural Network	53.38%	56.65%	61.22%	72.45%	98.16%
Ada Boost	56.05%	60.58%	67.14%	77.80%	97.69%
Gradient Boosting	56.26%	61.28%	67.50%	78.55%	97.79%

Table 5: ML Model with accuracy at certain time interval



1

Figure 13: Third dataset accuracies



Here, the high accuracy of Logistic Regression, SVM, Ada Boost and Gradient Boosting is still apparent. However, the values which were returned deviate from each other at the 40% time interval, but still come together at the last time interval. While Random Forest achieves scores lower than previously listed models at the 20% time interval, it ends up scoring an accuracy of 97.74%. Also, the models Decision Tree, KNN, Naive Bayes and Neural Network all perform worse than Random Forest. As KNN and Naive Bayes attain a score lower than 95%, both Neural Network and Decision Tree end up achieving an accuracy of 98.16% and 96.09% respectively. One characteristic of the plot which remains apparent is the performance of Naive Bayes at the 40%, 60% and 80% time interval as it exceeds the values of Neural Network, KNN and Decision Tree. Moreover, the Decision Tree model delivers relatively low accuracies, but ultimately overtakes Naive Bayes and KNN.

5.5 General observations

Overall, Logistic Regression, SVM and Gradient Boosting deliver the highest accuracies throughout all datasets while Decision Tree, KNN and Naive Bayes consistently scored below average.

As Neural Network shows a recurring pattern of underperforming at early intervals and significantly improving at later intervals, it does not maintain this behavior in the first dataset. Moreover, the Random Forest classifier always returns accuracies superior to those from the Decision Tree. This result can be deduced from the risk of overfitting (cf. Ibm 2024a) and therefore implies that a Random Forest should almost always be chosen over a Decision Tree, unless memory and computation time are of relevance. Furthermore, out of all models which adhere to a tree structure, Decision Tree performed worst while Gradient Boosting returned the best results in the second and third dataset. Regarding Ada Boost and Random Forest, both models yielded similar results with slight deviations, except in the first dataset, where the performance notably varies. As for the first dataset, Random Forest eventually outperforms Gradient Boosting, while Ada Boosts outcomes fall between Decision Tree and Gradient Boosting.

Accuracies from the 20%/10-minute time point or 40%/14-minute time point also deliver mentionable findings, as the results of Logistic Regression, SVM and Gradient Boosting are always better than the other models (one exception occurs in the second dataset where Ada Boost scores higher than SVM). This indicates that these models would be a better fit for win predictors, when little information about a match is given. The models in the second and third dataset never score better than 60% in the first time point. However, when increasing the number of matches from 25,082 to 64,556, the majority of models exceed 60%.

The observations made from the graphs suggest that it would be more suitable to set the time points to timestamps instead of time intervals, since the second dataset scored higher than the third one although the size of both data collections were identical. Furthermore, a higher score in the first dataset which comprises a larger amount of data indicates that the sample size matters and could elevate the model's performance in terms of accuracy.

6 The Pearson Correlation Coefficient-Based Algorithm (PCCBA)

Akin to training and evaluating Machine Learning models, a custom heuristic that does not rely on Machine Learning was developed, utilizing the Pearson correlation ρ_{xy}. This correlation has been discovered when producing heat maps with the seaborn library. These heat maps contain the correlations between the independent and dependent variables and set the correlation method to 'pearson' as default. Thus, this correlation was selected. The algorithm makes use of the following formula to calculate the probability of the blue team winning:

$$P(X=1) = \sum_{c=1}^{\gamma} \left(\frac{\left(|\rho_{B_c Y}| + |\rho_{R_c Y}| \cdot \frac{1}{2}\right)}{\sum_{d=1}^{\gamma} \left(|\rho_{B_d Y}| + |\rho_{R_d Y}| \cdot \frac{1}{2}\right)} \cdot \frac{b_{tc}+1}{b_{tc}+r_{tc}+2} \right)$$

Here, the index t represents a data point of the testing set, while indices c and d represent the predictor variables. Also, γ is set equivalent to the half of all independent variables, as two variables which correlate with either the blue team or red team can be assigned to the same category. The left term of the sums inner product normalizes the Pearson correlation between an independent variable and the output variable with respect to the sum of all correlations. While variable B represents the independent variables related to the blue team, variable R indicates an association with the red team. The calculation of the Pearson correlation between variables X and Y occurs through dividing their covariance by the product of their respective standard deviations (cf. Berman 2016, 168).

$$\text{Pearson Correlation } (\rho_{xy}): \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2}\sqrt{\sum(y_i - \bar{y})^2}}$$

Additionally, the numerator of the normalization fraction represents the computation of the mean between the correlations absolute values which are in relation with the same independent variable (e.g. blueFirstBlood and redFirstBlood), as red team associated variables tend to suggest a negative correlation towards the blueWin variable when increased. The average of a pairwise correlation represents a portion of the win probability. Since the sum of all averages exceeds the value 1, normalization for each average must be applied.

Conversely, the right term of the sum's inner product determines the amount of the portion which will be assigned the win probability through dividing the value of a blue team correlated independent variable by the total value of the independent variable. Also, both variables b and r will be incremented by 1 to prevent division by zero.

After adding up all percentages which contribute to the chances of the blue team winning, a probability is obtained.

6.1 Results of the *PCCBA*

The subsequent table illustrates the amount of time the algorithm required to process the different datasets. The time and space complexity of the program both go by $O(n)$.

Dataset	**Time Intervals (60 435 games)**	**Timestamps (25 082 games)**	**Time Intervals (25 082 games)**
Time	3min 36s 41ms	1min 30s 289ms	1min 30s 374ms

Table 6: Time required to handle different datasets

Additionally, following table and graph deliver the accuracies which can be deduced from the respective provided datasets that contain either time interval or timestamp information.

Dataset	**20%/10th minute**	**40%/14th minute**	**60%/20th minute**	**80%/27th minute**	**100%/end**
1st dataset	61.68%	68.64%	75.33%	83.76%	93.84%
2nd dataset	59.01%	62.51%	67.84	80.75%	91.25%
3rd dataset	56.20%	60.52%	66.62%	75.45%	91.25%

Table 7: *PCCBA* accuracies

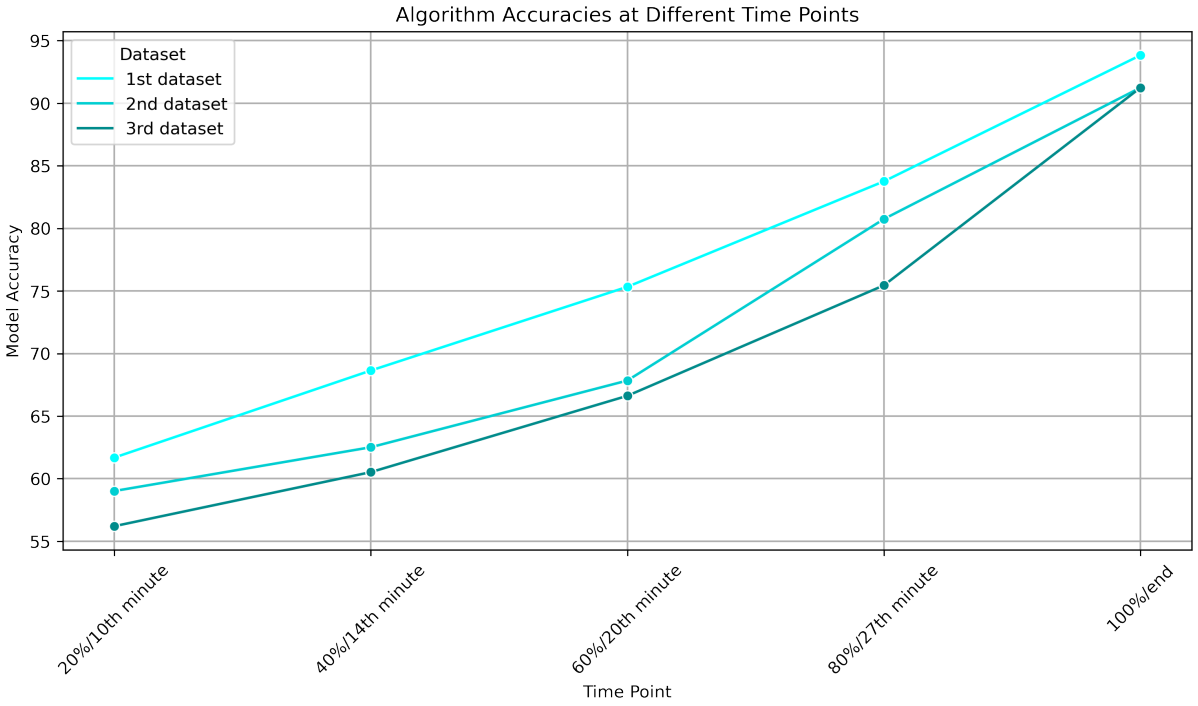


Figure 14: Accuracies of *PCCBA*

Similar to previous findings, the performance of the algorithm, when applying the first dataset was best, while the second dataset shows inferior results, and the third dataset attains the worst scores. Moreover, the line attributed to the first dataset expresses linear behaviour, as the other sets of data receive a surge in terms of accuracy after the 20-minute mark or 80% interval. The continuous increase in accuracy after every time point indicates that the algorithm is suitable for win prediction tasks.

6.2 Comparison with other models

In order to compare the Pearson correlation-based algorithm with other Machine Learning models, following three plots have been made to visualize overall accuracies. As in previous instances, the first plot includes data deduced from the first dataset, while the second plot correlates with the second dataset and the third plot with the third dataset.

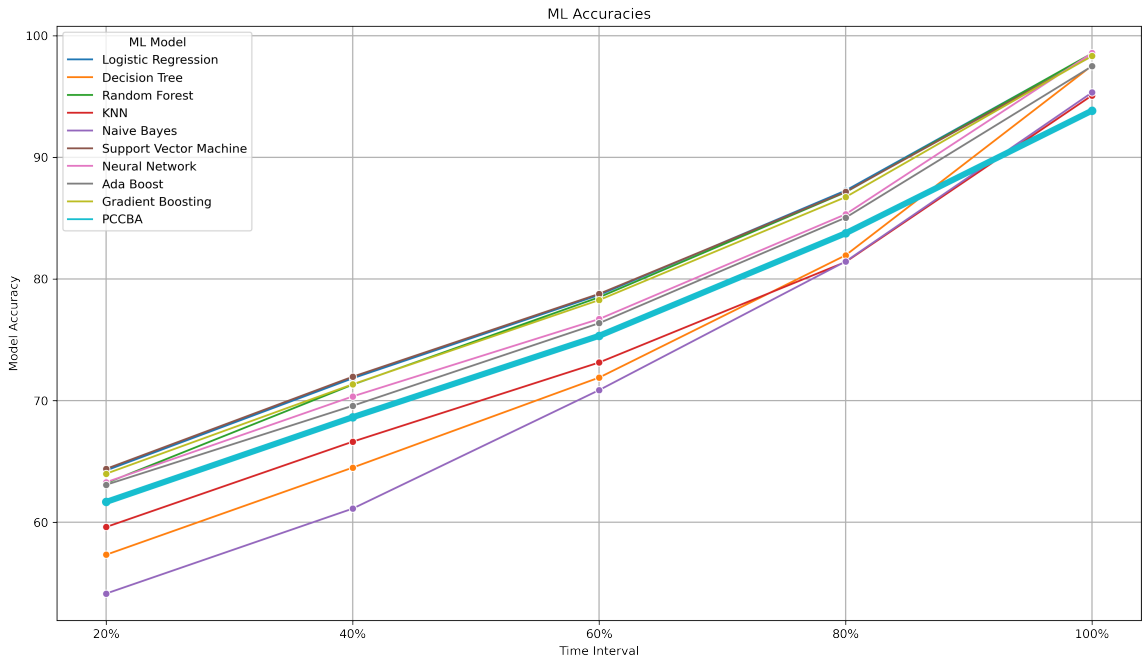


Figure 15: First dataset accuracies with *PCCBA*

In the first dataset the *PCCBA* performs mediocre when compared to other models and ends up scoring lowest at the last time interval. However, the algorithms' accuracies were consistently higher than those of KNN, Decision Tree and Naive Bayes throughout the first four time points.

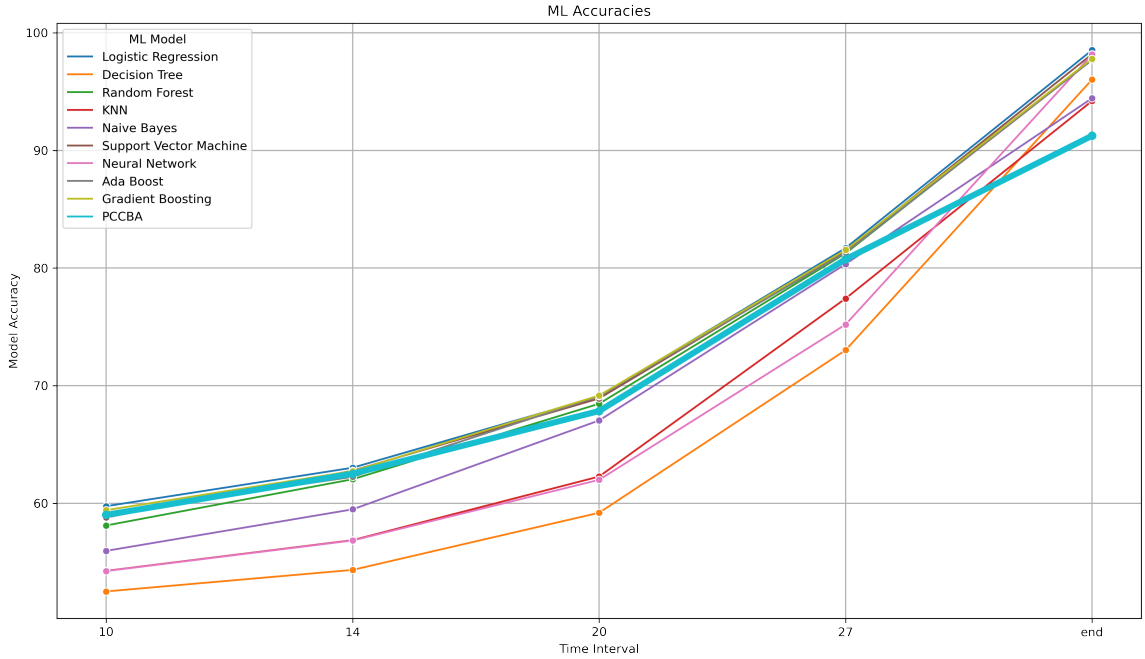


Figure 16: Second dataset accuracies with *PCCBA*

Regarding the second dataset, as the *PCCBA* can keep up with other Machine Learning models that

returned the highest accuracies (e.g. Logistic Regression, Gradient Boosting and SVM), it ends up performing poorly at the last minute mark. In addition to exceeding the capabilities of KNN, Decision Tree and Naive Bayes until the 4th time point it also scored higher than Neural Network in terms of accuracy.

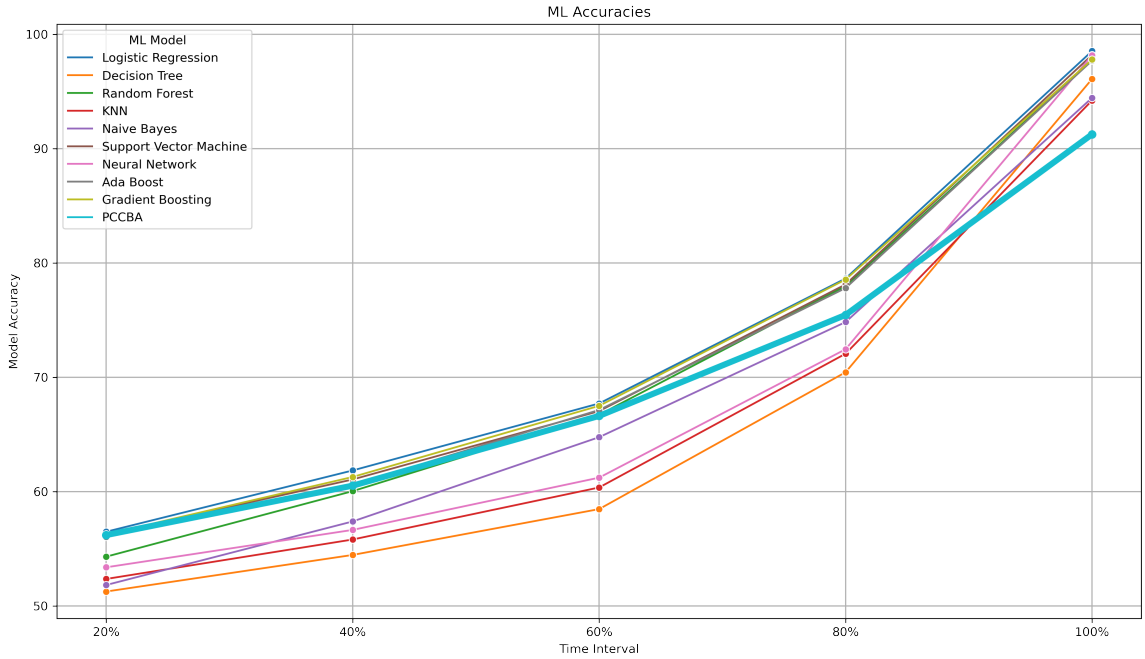


Figure 17: Third dataset accuracies with *PCCBA*

In the third dataset, the *PCCBA* delivers similar results when compared to the second dataset. However, in contrast to the previous dataset, the Pearson correlations' performance already starts to decrement at the 80% time interval.

7 Discussion

As previously mentioned, the Machine Learning models were provided by the scikit-learn library, Nevertheless, other libraries such as PyTorch, TensorFlow and Keras also contain Machine Learning models that might perform better than the models from scikit-learn. Akin to changing the Machine Learning library, one could also replace the Pearson correlation with other correlation methods such as the Spearman correlation or the Kendall rank correlation.

It is also notable to include that the Neural Network was only made of two hidden layers and can be further extended. One could for example include more hidden layers with more nodes, but it is important to note that a more complex Neural Network would require more time for training. However, as the models do not depend on the time variable in a real-time scenario, an extension of the Neural

Network can be taken into consideration.

As there are many factors which can influence the accuracy of a model, future studies could take a look at other Machine Learning libraries, extend the structure of the Neural Network, implement other correlations, feed the model with more data, or implement other Machine Learning models to create a win predictor which yields the highest accuracies and attribute to a more engaging experience in the e-sports industry.

8 Conclusion

The plots deduced from calculated accuracies clearly show which Machine Learning models would fit best for *MOBA* game win prediction. The set of models which represents the upper accuracy boundary comprises Logistic Regression, SVM and Gradient Boosting. Although each models returns a high value in terms of accuracy, training time does not come with similarities, as SVM requires the longest and second longest amount of time among all models, while Logistic Regression fitted the data almost in an instant and Gradient Boosting only needed a few of minutes. However, as the fitting process occurs before evaluating supplementary data, all of the previously mentioned models can be taken into consideration when creating a real-time win predictor.

The fact that each model returns a higher accuracy as elapsed time increases, confirms the stated hypothesis. Also, the time point is not the only parameter which has an impact on the result, since a larger dataset also contributed to higher accuracies. It is also important to determine the data point type, as timestamps were superior to time intervals.

Ultimately, the PCCBA was not able to outperform high-ranking Machine Learning models. However, when dataset size was smaller, the algorithm was able to keep up with the highest scorers in earlier time points but ended up performing poorly at later time points.

Acknowledgments

I would like to express my sincere gratitude to Michael Eickmeyer without whom this work would have never been materialized, as a lot of valuable feedback and resources have been provided. Also, I am very thankful for his guidance, the time he devoted to helping refine and improve this work, and the countless meetings we have had.

References

Bahrammirzaee, Arash. 2010. "A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems." *Neural Computing and Applications* 19 (8): 1165–1195. ISSN: 1433-3058. https://doi.org/10.1007/s00521-010-0362-z. https://doi.org/10.1007/s00521-010-0362-z.

Berman, Jules J. 2016. *Data simplification: Taming information with open source tools.* 1st ed. Cambridge: Morgan Kaufmann.

Berners-Lee, Tim, Larry M Masinter, and Mark P. McCahill. 1994. *Uniform Resource Locators (URL).* RFC 1738, 1738, December. https://doi.org/10.17487/RFC1738. https://www.rfc-editor.org/info/rfc1738.

Bernstein, Matthew N. 2017. *The Radial Basis Function Kernel.* December 26, 2024. https://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/svms/RBFKernel.pdf.

Brooks, R. A. 1991. "New approaches to robotics" [in eng]. *Science* 253, no. 5025 (September): 1227–1232. ISSN: 0036-8075. https://doi.org/10.1126/science.253.5025.1227.

Campbell, Murray, A.Joseph Hoane, and Feng-hsiung Hsu. 2002. "Deep Blue." *Artificial Intelligence* 134 (1): 57–83. ISSN: 0004-3702. https://doi.org/https://doi.org/10.1016/S0004-3702(01)00129-1. https://www.sciencedirect.com/science/article/pii/S0004370201001291.

Cannizo, Alejandro, and Esmitt Ramírez. 2015. "Towards Procedural Map and Character Generation for the MOBA Game Genre." *Ingeniería y Ciencia* (11):95–119.

Chen, Mingzhe, Ursula Challita, Walid Saad, Changchuan Yin, and Mérouane Debbah. 2019. "Artificial neural networks-based machine learning for wireless networks: A tutorial." *IEEE Communications Surveys & Tutorials* 21 (4): 3039–3071.

Chollet, François. 2018. *Depp Learning mit Python und Keras.* Frechen: mitp.

Cooksey, Brian. 2014. *An Introduction to APIs.* Zapier, Inc. https://cdn.zapier.com/storage/learn_ebooks/e06a35cfcf092ec6dd22670383d9fd12.pdf.

Davila, Annette Lopez. 2020. *Neural Networks: The Backpropagation Algorithm.* December 2, 2024. https://cklixx.people.wm.edu/teaching/math400/Annette-paper.pdf.

Dayananda, Sandun. 2023. *Logistic Regression.* December 26, 20224. https://sandundayananda.medium.com/logistic-regression-55512384851b.

Dermaku, A., N. Demaku, and Xh. Bajrami. 2013. "Reducing of the latency between the client and server using Heuristic Partitioning Approaches on Cloud Computing Architecture." *IFAC Proceedings Volumes* 46 (8): 64–68.

Emmert-Streib, Frank, Olli Yli-Harja, and Matthias Dehmer. 2020. "Artificial Intelligence: A Clarification of Misconceptions, Myths and Desired Status." *Frontiers in Artificial Intelligence* 3. ISSN: 2624-8212. https://doi.org/10.3389/frai.2020.524339. https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2020.524339.

Esports Charts. 2025. *Most Popular Esports Games 2024.* January 5, 2025. https://escharts.com/top-games?order=peak.

Fielding, Roy, Jim Gettys, Jeff Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. 1999. *Hypertext Transfer Protocol – HTTP/1.1.* Technical report. RFC 2616, Internet Engineering Task Force, June. https://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. "Array programming with NumPy." *Nature* 585, no. 7825 (September): 357–362. https://doi.org/10.1038/s41586-020-2649-2. https://doi.org/10.1038/s41586-020-2649-2.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2001. *The Elements of Statistical Learning.* New York: Springer New York Inc.

Haykin, Simon. 1999. *Neural Networks: A Comprehensive Foundation.* Upper Saddle River: Prentice Hall.

Hunter, J. D. 2007. "Matplotlib: A 2D graphics environment." *Computing in Science & Engineering* 9 (3): 90–95. https://doi.org/10.1109/MCSE.2007.55.

Ibm. 2024a. *What is a decision tree?* December 2, 2024. https://www.ibm.com/topics/decision-trees#:~:text=A%20decision%20tree%20is%20a,internal%20nodes%20and%20leaf%20nodes..

———. 2024b. *What is a neural network?* December 2, 2024. https://www.ibm.com/topics/neural-networks.

———. 2024c. *What is boosting?* December 2, 2024. https://www.ibm.com/topics/boosting.

———. 2024d. *What is Random Forest?* December 2, 2024. https://www.ibm.com/topics/random-forest.

Ibm. 2024e. *What is support vector machine?* December 2, 2024. https://www.ibm.com/topics/support-vector-machine#:~:text=What%20are%20SVMs%3F,in%20an%20N-dimensional%20space..

———. 2024f. *What is the KNN algorithm?* December 2, 2024. https://www.ibm.com/topics/knn.

IONOS. 2024. *Der HTTP-Header - eine Übersicht für Anwender.* September 25, 2024. https://www.ionos.at/digitalguide/hosting/hosting-technik/http-header/.

Junior, Jailson, and Cláudio Campelo. 2023. "League of Legends: Real-Time Result Prediction." In *Anais do XVI Congresso Brasileiro de Inteligência Computacional.* CBIC 2023. SBIC.

Karabiber, Fatih. 2024. *Gini Impurity.* December 26, 2024. https://www.learndatasci.com/glossary/gini-impurity/?utm_source=chatgpt.com.

Khaleel, Mohamed, Abdullatif Jebrel, and Dunia M. Shwehdy. 2024. "Artificial Intelligence in Computer Science." *International Journal of Electrical Engineering and Sustainability (Int. J. Electr. Eng. And Sustain.)* 2 (2): 01–21. https://doi.org/10.5281/zenodo.10937515. https://ijees.org/index.php/ijees/article/view/80.

Knudsen, J Everett, Umar Ghaffar, Runzhuo Ma, and Andrew J Hung. 2024. "Clinical applications of artificial intelligence in robotic surgery." *Journal of Robotic Surgery* 18, no. 1 (March): 102. ISSN: 1863-2491. https://doi.org/10.1007/s11701-024-01867-0.

Kotsiantis, Sotiris B, Ioannis Zaharakis, P Pintelas, et al. 2007. "Supervised machine learning: A review of classification techniques." *Emerging artificial intelligence applications in computer engineering* 160 (1): 3–24.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. 2012. "ImageNet Classification with Deep Convolutional Neural Networks." In *Advances in Neural Information Processing Systems,* edited by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, vol. 25. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Lang, Niklas. 2024. *What is AdaBoost?* December 2, 2024. https://databasecamp.de/en/ml/adaboost-en.

Liu, Qiong, and Ying Wu. 2012. "Supervised Learning." In *Encyclopedia of the Sciences of Learning,* edited by Norbert M. Seel, 3243–3245. Boston, MA: Springer US. ISBN: 978-1-4419-1428-6. https://doi.org/10.1007/978-1-4419-1428-6_451. https://doi.org/10.1007/978-1-4419-1428-6_451.

Marín, Alfredo, Luisa I. Martínez-Merino, Justo Puerto, and Antonio M. Rodríguez-Chía. 2022. "The soft-margin Support Vector Machine with ordered weighted average." *Knowledge-Based Systems* 237:107705. ISSN: 0950-7051. https://doi.org/https://doi.org/10.1016/j.knosys.2021.107705. https://www.sciencedirect.com/science/article/pii/S0950705121009576.

Masui, Tomonori. 2024. *All you need to know about gradient boosting algorithm - Part 2. classification.* December 2, 2024. https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-2-classification-d3ed8f56541e.

McKinney, Wes. 2010. "Data Structures for Statistical Computing in Python." In *Proceedings of the 9th Python in Science Conference,* edited by Stéfan van der Walt and Jarrod Millman, 56–61. https://doi.org/10.25080/Majora-92bf1922-00a.

Monett, Dagmar, and Colin W. P. Lewis. 2018. "Getting Clarity by Defining Artificial Intelligence—A Survey." In *Philosophy and Theory of Artificial Intelligence 2017,* edited by Vincent C. Müller, 212–214. Cham: Springer International Publishing. ISBN: 978-3-319-96448-5.

Naeem, Samreen, Aqib Ali, Sania Anam, and Munawar Ahmed. 2023. "An Unsupervised Machine Learning Algorithms: Comprehensive Review." *IJCDS Journal* 13 (April): 911–921. https://doi.org/10.12785/ijcds/130172.

OpenAI, : Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, et al. 2019. *Dota 2 with Large Scale Deep Reinforcement Learning.* arXiv: 1912.06680 [cs.LG]. https://arxiv.org/abs/1912.06680.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12:2825–2830.

Peretz, Or, Michal Koren, and Oded Koren. 2024. "Naive Bayes classifier – An ensemble procedure for recall and precision enrichment." *Engineering Applications of Artificial Intelligence* 136:108972. ISSN: 0952-1976. https://doi.org/https://doi.org/10.1016/j.engappai.2024.108972. https://www.sciencedirect.com/science/article/pii/S0952197624011308.

Phillips, Peter C.B. 1988. "Conditional and unconditional statistical independence." *Journal of Econometrics* 38 (3): 341–348. ISSN: 0304-4076. https://doi.org/https://doi.org/10.1016/0304-4076(88)90049-8. https://www.sciencedirect.com/science/article/pii/0304407688900498.

Riot Games. 2024. *Welcome to the rift - learn the basics.* August 19, 2024. https://www.leagueoflegends.com/en-gb/how-to-play/.

Rish, Irina. 2001. "An Empirical Study of the Naïve Bayes Classifier." *IJCAI 2001 Work Empir Methods Artif Intell* 3 (January).

Schwab, Brian. 2009. *AI Game Engine Programming* [in English]. 2nd ed. xxx, 710 pages. Illustrations; 24 cm + 1 CD-ROM (3 4/3 in.) Boston, MA: Course Technology, Cengage Learning. ISBN: 9781584505723, 1584505729. https://worldcat.org/title/263295961.

Silva Junior, Jailson Barros da, and Claudio Campelo. 2023. *League of Legends Match Data at Various Time Intervals,* August. https://doi.org/10.5281/zenodo.8303397. https://doi.org/10.5281/zenodo.8303397.

Sollins, K., and L. Masinter. 1994. *Functional Requirements for Uniform Resource Names.* Technical report. RFC 1737, Internet Engineering Task Force, December. https://www.rfc-editor.org/rfc/rfc1737.txt.

Sutton, Richard S., and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction.* Cambridge: A Bradford Book.

Wang, Pei. 2019. "On Defining Artificial Intelligence." *Journal of Artificial General Intelligence* 10 (2): 1–37. https://doi.org/doi:10.2478/jagi-2019-0002. https://doi.org/10.2478/jagi-2019-0002.

Waskom, Michael L. 2021. "seaborn: statistical data visualization." *Journal of Open Source Software* 6 (60): 3021. https://doi.org/10.21105/joss.03021. https://doi.org/10.21105/joss.03021.

Webb, Geoffrey I, Eamonn Keogh, and Risto Miikkulainen. 2010. "Naıve Bayes." *Encyclopedia of machine learning* 15 (1): 713–714.

William, Andrew. 2021. *A comprehensive mathematical approach to understand AdaBoost.* December 2, 2024. https://towardsdatascience.com/a-comprehensive-mathematical-approach-to-understand-adaboost-f185104edced.

Ye, Deheng, Guibin Chen, Wen Zhang, Sheng Chen, Bo Yuan, Bo Liu, Jia Chen, et al. 2020. "Towards Playing Full MOBA Games with Deep Reinforcement Learning." In *Advances in Neural Information Processing Systems,* edited by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, 33:621–632. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/06d5ae105ea1bea4d800bc96491876e9-Paper.pdf.

Zhong, Yuheng. 2024. "The application of artificial intelligence in MOBA games." *Applied and Computational Engineering* 47 (March): 238–241. https://doi.org/10.54254/2755-2721/47/20241384.

List of Figures

Appendix

Equation 1

$$f(ab) = e^{ab} \tag{1}$$

$$= \sum_{n=0}^{\infty} \frac{f^{(n)}(c)}{n!}(ab-c)^n$$

$$= e^c + \frac{e^c}{1!}(ab-c) + \frac{e^c}{2!}(ab-c)^2 + ... + \frac{e^c}{\infty!}(ab-c)^\infty \qquad |c=0$$

$$= e^0 + \frac{e^0}{1!}(ab-0) + \frac{e^0}{2!}(ab-0)^2 + ... + \frac{e^0}{\infty!}(ab-0)^\infty$$

$$= 1 + \frac{1}{1!}ab + \frac{1}{2!}(ab)^2 + ... + \frac{1}{\infty!}(ab)^\infty$$

Equation 2

$$e^{-\frac{1}{2}(a-b)^2} \tag{2}$$

$$= e^{-\frac{1}{2}(a^2+b^2-2ab)}$$

$$= e^{-\frac{1}{2}(a^2+b^2)}e^{ab}$$

$$= e^{-\frac{1}{2}(a^2+b^2)}\left[\left(1, \sqrt{\frac{1}{1!}}a, \sqrt{\frac{1}{2!}}a^2, ..., \sqrt{\frac{1}{\infty!}}a^\infty\right) \cdot \left(1, \sqrt{\frac{1}{1!}}b, \sqrt{\frac{1}{2!}}b^2, ..., \sqrt{\frac{1}{\infty!}}b^\infty\right)\right]$$

$$= \left(e^{-\frac{1}{2}(a^2+b^2)}, e^{-\frac{1}{2}(a^2+b^2)} \cdot \sqrt{\frac{1}{1!}}a, e^{-\frac{1}{2}(a^2+b^2)} \cdot \sqrt{\frac{1}{2!}}a^2, ..., e^{-\frac{1}{2}(a^2+b^2)} \cdot \sqrt{\frac{1}{\infty!}}a^\infty\right) \cdot$$
$$\left(e^{-\frac{1}{2}(a^2+b^2)}, e^{-\frac{1}{2}(a^2+b^2)} \cdot \sqrt{\frac{1}{1!}}b, e^{-\frac{1}{2}(a^2+b^2)} \cdot \sqrt{\frac{1}{2!}}b^2, ..., e^{-\frac{1}{2}(a^2+b^2)} \cdot \sqrt{\frac{1}{\infty!}}b^\infty\right)$$

Equation 3

$$\frac{\delta L}{\delta z} = \frac{\delta}{\delta z}\sum_{i=1}^{j}(y_i - z_i)^2 \tag{3}$$

$$= \sum_{i=1}^{j} 2 \cdot (y_i - z_i) \cdot (-1) \qquad \left|\frac{\delta}{\delta z}y_j - z_j = (-1)\right.$$

$$= \sum_{i=1}^{j}(-2) \cdot (y_i - z_i)$$

$$\frac{\delta z}{\delta b} = \frac{\delta}{\delta b}b + \frac{\delta}{\delta b}\sum_{i=0}^{n} h(x_f)$$

$$= 1$$

Begleitprotokoll

Name des Schülers / der Schülerin: Joshua Gao

Thema der Arbeit: The utilization of Artificial Intelligence in *MOBA* video games

Name der Betreuungsperson: BEd Alexander Höfler

Datum	Arbeitsschritt
09.10.2023	Erstes Gespräch mit der Betreuungslehrperson
08.04.2024	Gespräch Erwartungshorizont
12.04.2024	VwA-Mentoring: Zeitmanagement und VwA-Plan erstellt
30.05.2024	VwA-Mentoring: Einblicke in die Riot API
26.06.2024	VwA-Mentoring: Paper über MOBA win prediction besprochen
09.08.2024	Mit Background angefangen
16.08.2024	VwA Mentoring + GitHub Repo erstellt
21.08.2024	LolWatcher API Wrapper untersucht
24.08.2024	Neues Programm mit LCU wrapper geschrieben
07.09.2024	VwA-Mentoring: Sklearn/Tensorflow für ML model Evaluation
05.09.2024	VwA-Mentoring: Korrelationen der Variablen betrachtet
05.10.2024	ML Algorithmen getestet: Linear Regression, Logistic Regression
13.10.2024	Puuids der Spieler mithilfe der Riot-API extrahiert (Challenger und Grandmaster Spieler)
14.10.2024	Match-IDs mithilfe der API extrahiert
23.10.2024	KNN, Decision Tree und Random Forest getestet
26.10.2024	VwA-Mentoring: Vorbereitung auf Erstellung des Datensets
26.10.2024	Naive Bayes getestet
28.10.2024-03.11.2024	eigene Datensets erstellt (3 Arten)
20.11.2024-03.12.2024	Kapitel Machine Learning Modelle + Datenset und API
27.11.2024	Genauigkeit der ML Modelle evaluiert (eigene Datensets als Input verwendet)
23.11.2024	VwA-Mentoring: Machine Learning Model Abschnitt besprochen
21.12.2024	VwA-Mentoring: Besprechung eigener Algorithmus
23.12.2024	Background Überarbeitung
24.12.2024-27.12.2024	ML Modelle Überarbeitung
13.01.2025	VwA Erstversion
31.01.2025	letzte Überarbeitung

Die Arbeit hat eine Länge von 52 871 Zeichen.

Wien, 09.02.2025

Joshua Gao

Joshua Gao

Eigenständigkeitserklärung VWA

Name: Joshua Gao

Ich erkläre, dass ich die Vorwissenschaftliche Arbeit selbstständig verfassen und ausschließlich Quellen (Literatur, Interviews, Medienbeiträge...) verwenden werde, die ich als Zitate belege und im Quellenverzeichnis anführe. Mir ist bewusst, dass es sich andernfalls um eine vorgetäuschte Leistung handeln würde, die nicht zu beurteilen ist.

Ich versichere, dass ich meine Vorwissenschaftliche Arbeit weder im In- noch im Ausland als Prüfungsarbeit vorgelegt habe oder vorlegen werde.

Wien, 09.02.2025 — Joshua Gao

Ort, Datum — Unterschrift bzw. Handysignatur

Zustimmung zur Aufstellung in der Schulbibliothek

Ich gebe mein Einverständnis, dass ein Exemplar meiner Vorwissenschaftlichen Arbeit in der Schulbibliothek meiner Schule aufgestellt wird.

Wien, 09.02.2025 — Joshua Gao

Ort, Datum — Unterschrift bzw. Handysignatur

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen Hilfsmittel als die angegebenen benützt habe. Die Stellen, die anderen Werken (gilt ebenso für Werke aus elektronischen Datenbanken oder aus dem Internet) wörtlich oder sinngemäß entnommen sind, habe ich unter Angabe der Quelle und Einhaltung der Regeln wissenschaftlichen Zitierens kenntlich gemacht. Diese Versicherung umfasst auch in der Arbeit verwendete bildliche Darstellungen, Tabellen, Skizzen und Zeichnungen. Für die Erstellung der Arbeit habe ich auch folgende Hilfsmittel generativer KI-Tools _____ (z. B. ChatGPT, Grammarly Go, Midjourney) zu folgendem Zweck verwendet: [*Bitte hier Einsatzgebiet anführen.*]. Die verwendeten Hilfsmittel wurden vollständig und wahrheitsgetreu inkl. Produktversion und Prompt ausgewiesen. -

Wien, 09.02.2025
Ort, Datum

Joshua Gao
Unterschrift bzw. Handysignatur