League of Legends Esports Analysis: A Machine Learning and Sentiment Analysis Approach to Predict Pre-Match Winners

By

Karthickeyan Sengodan Rajasekar

Student ID: 2443050

Supervisor: **Dr. Mubashir Ali**

A thesis submitted to the University of Birmingham
For the degree of MSc in Data Science

School of Computer Science
University of Birmingham
Birmingham, UK

September 2023

Abstract

League of Legends is a multiplayer online battle arena (MOBA) game played by millions of players worldwide. Predicting the outcome of esports matches is essential for developing team strategy, esports betting and much more. In this study, we test nine machine learning models and tune their hyperparameters using methods like Grid Search, Random Search, and Bayesian Optimisation. We also introduce a new way to select essential features using mutual importance and CatBoost scores. We perform sentiment analysis on Reddit post comments and incorporate these sentiment scores into our models for predicting match outcomes. By doing this, we evaluate the potential of including public sentiment in predictive analytics. Furthermore, we have developed an interactive dashboard that predicts the winner based on the trained best model and provides valuable visual insights into team performance, which helps the stakeholders and esports organisations strategise and make informed decisions.

Keywords: Esports, League Of Legends, Sentiment Analysis, Predictive Analytics, Machine Learning, Hyperparameter Tuning

Acknowledgement

I want to express gratitude to my supervisor, Dr. Mubashir Ali, as well as project supervisors Dr. Jizheng Wan and Dr. Qamar Natsheh, for their constant support during this project.

I'm also grateful to my family and friends for always being there for me. I would also like to thank the University of Birmingham for providing me the sufficient resources and guidance to complete this project.

Contents

List of Figures

Chapter 1

Introduction

The esports field has grown tremendously with technology development and more internet reach to many people worldwide. This growth has paved the way for a world of opportunities to be explored in this sector. One of the most popularly played esports titles all around the world by millions of players is League of Legends, commonly referred to as LoL. This game has attracted the attention of various stakeholders and esports organisations and is also quite popular among betting markets. Due to the significance of this game in the realm of esports, many companies sponsor teams from technology, merchandise and various other sectors to gain their product reach to people. Hence, a large amount of money is invested in these esports teams. The necessity for advanced, reliable and predictive analytics in League of Legends esports is the motivation behind this research project.

In a game of League of Legends, two teams, namely the red team and the blue team, fight against each other to gain control over a virtual battleground. Winning the game is not only dependent on who clicks faster, but it depends on the whole team working well together, how each player is doing and much more.

This project implements a dual approach. First, it aims to evaluate and fine-tune nine different machine-learning models for their predictive accuracy in forecasting match outcomes. These models are tested rigorously using advanced optimisation techniques for hyperparameter tuning such as Grid Search, Random Search and Bayesian Optimisation. Secondly, it incorporates public sentiment analysis into the prediction model. This involves the extraction of data from Reddit discussions related to the specific esports teams and the year, and then it analyses the text data.

An interactive prediction dashboard that uses the best-performing machine learning model to predict the outcome. The ultimate goal of the project is to improve the predictive power of the existing models and evaluate the effect of public sentiment analysis to provide stakeholders and esports organisations with tools that will help them formulate team strategies and make informed decisions.

The scope of this project is not confined to mere academic research, but it also has real-world applications that could revolutionise the way teams approach game strategy, how sponsors decide which teams to back and how fans engage with the sport. As esports continues to grow, the findings from this research project will contribute towards future studies and practical applications in this and related fields.

In summary, this project aims to take a leap forward in predictive analytics by deploying machine learning algorithms and sentiment analysis to predict match outcomes in League of Legends. This project hopes to contribute towards the growing field of esports analytics.

1.1 Structure of the Report

- **Chapter 2: Literature Review**
  This chapter explores existing studies and research to provide a field background, setting the stage for our own work.

- **Chapter 3: Background Information**
  In this section, we discuss the specifics about the League of Legends game, the nine machine learning algorithms used, and evaluation metrics for our models.

- **Chapter 4: Proposed Methodology**
  This chapter outlines our approach in detail, including the sentiment analysis techniques, feature selection methods, model formulation, evaluation, and our prediction dashboard.

- **Chapter 5: Experimental Results**
  This chapter discusses the specifics of the dataset used, performed exploratory data analysis (EDA), and the outcomes of our various experiments.

- **Chapter 6: Analysis and Discussions**
  This section provides the implications of our findings and how they contribute to the field.

- **Chapter 7: Conclusion and Future Work**
  In this chapter, we summarise the project's key findings and discuss potential areas of further research and development.

- **Chapter 8: References**
  This part of the report contains the list of references cited throughout the report.

- **Chapter 9: Appendices**
  This section of the report contains the appendices.

Chapter 2

Literature Review

1. Existing literature in Winner Prediction models:

Recent work has been done in the domain of machine learning and game prediction, with League of Legends (LoL) being a frequent subject of interest. Hitar-García et al., (2023) conducted a study, presenting a thorough analysis of various machine learning methods for predicting the outcome of LoL games [1]. They revealed that simple models like decision trees and logistic regression could provide results competitive with more advanced models when given relevant features. This work highlights the importance of detailed understanding of game mechanics and their ensemble methods including xgbstack and votestack algorithms obtained 70% and 69% accuracies respectively for the winner prediction models.

Another work by Shen (2022) further explores the significance of individual player performance metrics [2]. They had made use of a dataset that contained the first ten minutes of the LoL esports games. The study introduced metrics such as kill-death-assist (KDA) ratio, creep score, and gold earned, demonstrating how these factors considerably influence game outcomes. This study considered various machine learning algorithms of which their voting classifier predicted the game result and had the highest accuracy of 72.68%.

Yang et al., (2020) also made significant contributions to the field, using machine learning to predict match outcomes based on champion selection and dynamic features [3]. They have used the gold collection, kill score and tower destruction features for the red and blue teams.

Their prediction model showed an accuracy of more than 70%, increasing the potential of predicting LoL game outcomes with high precision. Their study also observed that after fifteen minutes into the game, the prediction accuracy is starts increasing tremendously.

Ani et al., (2019) used ensemble methods to preict League of Legends match outcomes [4]. They have used both prematch and in-game features and have used 60% of their data for training and the remaining 40% of their data for testing purposes. With their Random Forest algorithm, and by using their in-game and prematch data, they have achieved 95.52% accuracy especially due to the 'Ban' feature.

Silva et al., (2018) performed a study where they have compared different neural networks for predicting League of Legends match outcome [5]. They have found that simple RNN is better than recurrent neural network. They have got an accuracy of 63.9% while using in-game data when 0-5 minutes in-game and they got 83.5% accuracy when using 20-25 minutes in-game data. They had performed K fold cross-validation as well.

In a study by Costa et al., (2021), they have evaluated whether the prematch data can be used to predict the match outcome of League of Legends game before the match starts [6]. They have tested out different features and algorithms, of which Random Forest and Logistic Regression got them AUC value of 0.97.

Kim et al., (2020) proposed a research study where they calibrate the condifence for predicting League of Legends winner [7]. Their calibration model takes uncertainites in data into account as well. They achieved an expected calibration error score of 0.57% which is way better than what they got with the traditional method of 1.1%.

Mondal et al., (2022) carried out their research work in League of Legends winner prediction by evaluating if a support role player contributed enough towards securing the win [8]. They also analysed the individual performance of certain roles. They had implemented Monte Carlo simulation method and found that even if the support player performs well, the match might be a loss as well.

2. Existing literature in Sentiment Analysis:

Yu et al., (2023) proposed a research study to mine insights from Esports game reviews using sentiment analysis [9]. Their research concluded that their sentiment analysis framework can identify concerns of players using keywords from their reviews. They also developed a word cloud for the various topics found by their algorithm.

Chandra et al., (2020) had explored sentiment analysis using two methods, namely, machine learning and deep learning [10]. They performed Twitter Sentiment Analysis based on polarity and they achieved accuracy percentages that range from 81 to 90 percent. Their research concludes that by performing sentiment analysis, we can obtain better insights and increase the revenue for businesses.

Pak et al., (2011) carried out their research study on performing twitter sentiment analysis when the resources for language are unavailable [11]. This study states that affective lexicons help in studying emotions and in sentiment analysis. The study has been carried out for major languages and they have used Twittter data and their method managed to perform as good as supervised learning techniques.

Urriza et al., (2021) conducted a research study on aspect based sentiment analysis of game reviews created by the user [12]. Their method collects Steam reviews and carries out sentiment analysis and also their Support Vector Machine classifier provided an accuracy of 97% with aspect-based and for the polarity-based, they got an accuracy of 91% in total.

Chakraborty et al., (2018) conducted a research study on rating the generation of video games by performing sentiment analysis and polarity based on context [13]. They had used multiple machine learning models to perfrom sentiment analysis on Amazon Twitter Data. Their Stochastic Gradient descent model had the highest accuracy of 81% and they implemented a voting classifier along with the other models.

Chapter 3

Background

3.1 Esports

Electronic Sports (Esports) have seen notable growth in recent years, engaging millions of followers worldwide and becoming a significant cultural trend. Esports has been developed into a renowned entertainment sector, attracting several companies sponsoring the esports teams. The revenue generated by this industry has significantly increased due to the constant rise in fan viewership on social media platforms and their lively engagement. The Esports industry's value is projected to reach 5.74 billion US dollars by 2030 [14].

Esports refers to video gaming competitions in which professional teams compete against each other in various multiplayer games. These competitions are streamed online and are watched by millions of viewers across the globe. Originating from local arcades and LAN parties, Esports has developed into a mainstream entertainment industry with massive audiences and multi-million-dollar prize pools.

Several advancements in technology and internet connectivity worldwide have paved the way for the rise of the Esports industry, especially after the COVID-19 pandemic [15]. Streaming platforms like Twitch and YouTube enable players to stream their live gameplay to their viewers and allow them to interact with them in real time. This feature enables the fans to get more involved with the streamers and support them by viewing advertisements that play before their stream starts or offering donations.

Riot Games developed League of Legends (LoL), one of the world's premier online multiplayer battle arena games. Since its launch in 2009, it has become a cultural phenomenon, captivating players and audiences [16]. Its competitive scene has seen exceptional growth, with the annual LoL World Championship breaking viewership records. The 2019 World Championship final reached a peak concurrent viewership of 44 million and a total unique viewership of 100 million, making it one of the most-watched esports events in history [17].

This meteoric rise in popularity underscores the game's impact on the esports industry and reflects the broader trend of online gaming becoming a mainstream entertainment medium. We can analyse historical match data to derive valuable insights to help sponsors plan their marketing and game plans and increase revenue in the upcoming seasons.

The rise of esports has transformed competitive gaming into a global phenomenon with a massive fan base and lucrative opportunities for team sponsors [18]. The steady growth of esports has created a unified environment that connects players, fans, and industry professionals through the shared experience of gaming and entertainment. As this industry expands, the relationship between technology, social interaction, and business in esports highlights an essential change in today's digital culture, offering intriguing opportunities for further study and understanding.

3.2 League Of Legends

League of Legends (LoL) stands as a prominent figure in the world of online gaming. It is known for its engaging and unique game mechanics. As a multiplayer online battle arena game, it offers players a gaming environment filled with strategy, teamwork, and competition.

LoL revolves around two teams containing five players per team, each controlling a character known as a "champion." These teams face off on a battlefield, fighting to destroy the opposing team's main structure, called the Nexus. The path to the Nexus is defended by a series

of turrets and inhibitors, creating a challenging and strategic environment. Players can select their champion of choice from a wide array of champions, each possessing unique abilities and attributes. Champions fall into different roles, such as top, middle, bottom, jungle, and support, that show their responsibilities during a match.

Figure 1: Layout of the Summoner's Rift Map

Summoner's Rift, the standard playing field, consists of three lanes connected by a jungle area, namely the Top Lane, Middle Lane and the Bottom Lane [19]. Players must work together as a team in order to push through the enemy defences in these lanes, battling both computer-controlled minions and the opposing player champions.

Collaboration and formulating team strategy form the essence of match victory in LoL. Players must choose their champions based on their unique skills and roles and coordinate their actions amongst their teammates to increase their chances of winning the game. Tactical decisions, such as deciding when to attack or defend, definitely play a crucial role in determining the outcome of a match. During the game, all the players will earn gold by completing various tasks like defeating their opponents or computer-controlled minions. This gold can be spent on several items that will enhance their champion's capabilities, thereby adding a layer of complexity to the in-game strategy.

LoL has created a thriving community of gamers, spectators, and professional players. Its esports scene is exceptionally vibrant, with competitions taking place around the globe, attracting large audiences and sponsors. From casual gamers to professional esports athletes, LoL appeals to a broad audience, showing its status as a prominent title in modern online gaming. This makes it a rich subject for analysis, whether from a gaming, cultural, or economic perspective. [16]

3.3 Utilised Algorithms

In this section, we discuss about the overview of the nine machine learning algorithms that were selected in order to effectively predict the winner outcome of League of Legends esports matches.

3.3.1 XG Boost

XG Boost is expanded as Extreme Gradient Boosting. It is an optimised gradient boosting library that is very useful for large datasets. It is a very fast and efficient algorithm. This algorithm aims to minimise the loss function $L(y, f(x))$ [20].

3.3.2 Random Forest

When the count of decision trees increase in the Random Forest algorithm, the model performs better with enhanced performance metrics. It considers the majority basis for tree votes [21].

3.3.3 Naïve Bayes

Naïve Bayes is based on the Bayes' theorem and makes the 'naive' assumption that two features are independent of each other [22]. Mathematically, it is represented as follows:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}. \tag{3.1}$$

where

$P(A|B)$: The conditional probability of A given B

$P(B|A)$: The conditional probability of B given A

$P(A)$: The probability of A

$P(B)$: The probability of B

3.3.4 SVM Linear Kernel

SVM stands for Support Vector Machines (SVM). The SVM model works well in high dimensional spaces. When the dimension count is greater than the sample count, SVM is very useful. SVM Linear Kernel is useful when the data can be split linearly [23].

3.3.5 SVM Radial Kernel

The RBF kernel in Support Vector Machines is very similar to the SVM with linear kernel, but it allows the model to create nonlinear boundaries. The RBF stands for Radial Basis Function [24].

3.3.6 KNN

KNN stands for K-Nearest Neighbours. This algorithm works based on the measuremnt of class similarities of their neighbours and classifies them accordingly [25].

3.3.7 Multi-Layer Perceptron

MLP stands for MultiLayer Perceptron. This is neural network is of the artificial type and it contains 3 or more layers. They can be implemented to solve problems with large datasets as well. It is a type of artifical neural network [26].

3.3.8 Logistic Regression (Elastic Net)

The next algorithm is the Logistic Regression algorithm that has Elastic Net regularisation. It brings together the penalties of L1 and L2 regularisations, which will lead to a model with better predictive metrics [27].

3.3.9 AdaBoost (Logistic Regression)

AdaBoost is expanded as Adapative Boosting. This algorithm makes use of the learners which are weak to create a better learner. They learn from the mistakes caused by the learner previously. Logistic Regression is enabled in this AdaBoost algorithm used for our project [28].

3.4 Performance Metrics

The performance of the nine machine learning models that we have used in this project is evaluated using the following metrics, which is a crucial step. The six metrics that we have assessed include accuracy, precision, recall, F1-Score, training time and prediction time.

3.4.1 Accuracy

The proportion of correctly classified instances to the total number of instances in the dataset is called the accuracy. The formula for accuracy is given by:

$$\text{Accuracy} == \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

where TP is True Positives, TN is True Negatives, FP is False Positives, and FN is False Negatives [29].

3.4.2 Precision

Precision focuses on the correctness of positive class predictions made by our model. The formula for precision is given by:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.3}$$

where TP is True Positives and FP is False Positives.

3.4.3 Recall

Recall is the number of True Positives divided by the number of True Positives and the number of False Negatives. Recall is also called as sensitivity. The formula for recall is given by:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.4}$$

where TP is True Positives and FN is False Negatives.

3.4.4 F1- Score

The F1 Score is the weighted average of Precision and Recall. It considers both the false positives and false negatives. The formula for F1-Score is given by:,

$$\frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}. \tag{3.5}$$

3.4.5 Training and Prediction times:

Training time is the time taken to train the model on the training data. Prediction time is the time taken to predict the model on the unseen data. It contributes to the measure of computational cost. They contribute to the measure of computational cost.

Chapter 4

Proposed Methodology

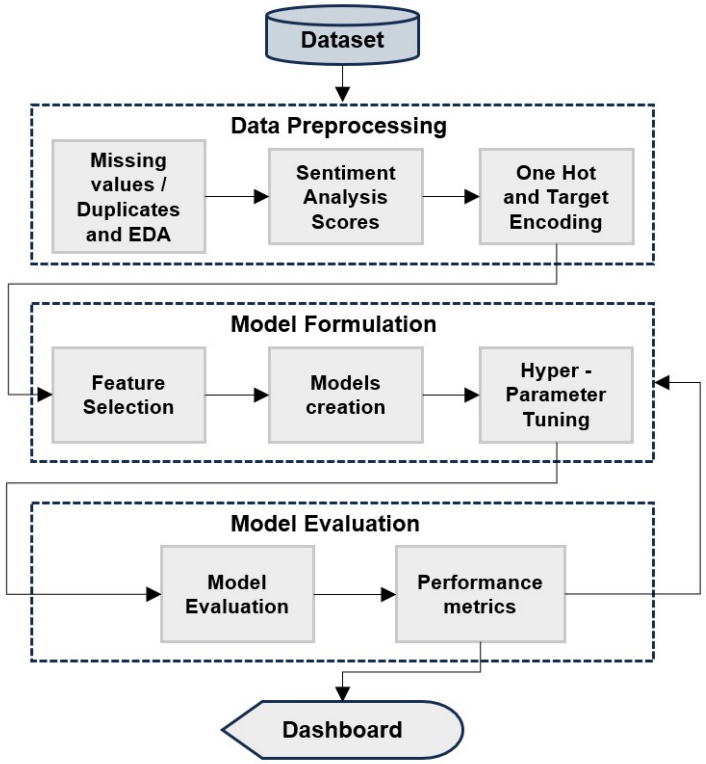

Figure 2: Flow diagram of the Proposed Methodology

4.1 Data Collection

The dataset for our project was obtained from Kaggle [30], which is a platform for machine learning and data science projects. The dataset contains records of professional esports League of Legends matches from various leagues and championships played between the years 2014 and 2018. The dataset is rich in terms of features and comprises many aspects of the esports matches, such as match setup, match progress and team names. The match setup includes data about team names, player names, champions selected, etc. The match progress part of the dataset consists of the data about the in-game statistics, including kills, deaths, gold earnings and objectives taken.

Our project focuses solely on the pre-game data because we aim to make predictions before the game starts to help stakeholders and esports teams formulate strategies well ahead. Hence, we do not consider the match progress or in-game statistics data and discard them for this study while utilising the pre-game data of the teams.

4.2 Sentiment Analysis

In esports, especially in a game like League of Legends, the sentiment of the fan community can play a significant role and affect team performance. The public sentiment influences the team morale, sponsorship decisions for the stakeholders, and the esports betting market. Hence, we incorporate sentiment analysis into our predictive model and evaluate its impact in esports competitive matches.

We implement a specific form of sentiment analysis called the Aspect-Based Sentiment Analysis (ABSA) [31]. Here, we evaluate the sentiment of specific entities instead of focusing on the entire document or sentence. Reddit is a massive discussion forum that enables the fan community, players, esports teams, and general audience to express their views and interact. The 'leagueoflegends' subreddit specifically comprises public sentiment about League of Legends teams and players. By focusing on this subreddit, we increase the relevance and

accuracy of our sentiment scores.

TextBlob [32] is a library for Python that helps process textual data and uses the machine learning algorithm known as the Naïve Bayes Classifier. It predicts the probability of the given text being in a sentiment category and classifies it accordingly. If it is a positive sentiment, it provides a positive sentiment score; if it is a negative sentiment, it provides a negative sentiment score. Initially, we create a Reddit Developer account and configure the PRAW (Python Reddit API Wrapper) [33] library for Python using our Reddit client ID, client secret and user agent details. Then, we create a search query that takes the team names and the specific year as input. Then, it fetches the top post for that query, matching the given input parameters. Finally, we collect the top 5 comments from the top post and then pass them through the TextBlob library for sentiment analysis. The sentiment scores are then updated to our sentiment analysis dataset.

4.3 Data Preprocessing

Data preprocessing is one of the most essential machine learning steps as it significantly impacts the performance of the predictive models. The raw data is transformed into a structured format to make it compatible for the machine learning algorithms to process. The objective behind data preprocessing is to remove anomalies, fill in missing values and convert the data.

Before starting with the data preprocessing, we have divided the primary dataset into training sets and testing subsets. In this project, we utilised 90% of the data for training and preserved the remaining 10% data for testing. This follows the fundamental practice in machine learning to validate the model's performance and prevent overfitting. Duplicate entries in the dataset may lead to overfitting the model; hence, it would perform poorly on unseen data. We identified the duplicates and removed them to increase the robustness of the models.

Categorical variables are generally of two types, namely, Nominal categorical variables and Ordinal Categorical variables. In our dataset, 'League', 'Season', and 'Type' are nominal categorical variables with low cardinality as they have few unique values. For nominal categorical variables, we have used one-hot encoding [34] as it converts such variables into a format that can be provided to machine learning algorithms. The one-hot encoding process converts each unique value into a new category and assigns a binary value of either 1 or 0 to it. To maintain consistency in the number of features between training and testing sets, we have handled the missing columns by setting them to zero in the testing set.

The categorical variables with high cardinality and a high number of unique categories were a challenge to deal with. One-hot encoding of these high cardinality categorical variables would lead to a massive increase in the number of categories and increase the dimensions of the dataset. This would result in computational inefficiency and overfitting. To address this problem, we have used target encoding [35]. Target encoding is a technique that suits well in this scenario as it replaces each category in the variable with the mean of the target variable. This type of encoding captures the valuable information of the dataset while maintaining its dimensions and makes it computationally more efficient. After performing one-hot and target encoding, we combine the encoded variables to form the final testing and training sets. These steps increase the predictive accuracy of the models and make them more reliable.

4.4 Feature Selection

Feature selection is a mandatory step in the machine-learning process, especially when it comes to complex machine-learning models. Removing the features with less importance and selecting the features with the highest importance is crucial to make the models more accurate. This will simplify the model architecture and will lead to an improvement in performance. In our research project, we have utilised a dual approach for feature selection that combines Mutual Information (MI) scores [36] and CatBoost feature importance scores [37]. These methods, especially when combined, provide quantitative

metrics that display the predictive power of each feature and help us build effective models. In our project, the feature selection process has been kept constant for all the nine machine learning models as we have to ensure that we feed each of the nine models with the same set of features to make a fair comparison. The processes from data preprocessing to feature selection are unaltered for all the nine machine learning models.

4.4.1 Mutual Information (MI)

Mutual Information (MI) is a method that measures the statistical dependence between two variables. It quantifies how much information about one variable reduces the uncertainty about the other variable. The MI score is very useful for feature selection as it measures the relationship between the predictor and the target variable. The Scikit library implements MI using the 'mutual_info_classif' function [38]. We obtain the MI scores for the features. The features are then ranked on the basis of their importance obtained from MI scores.

4.4.2 CatBoost

CatBoost (Categorical Boosting) is a machine-learning algorithm based on gradient boosting over decision trees [39]. The significant strength of CatBoost is that it can handle categorical variables easily, which is highly required for our dataset that comprises categorical primary variables. The CatBoost technique has an in-built feature importance option, which is calculated while the model is trained, and it is based on each feature's contribution to the model's predictive accuracy. The feature importances obtained from CatBoost serve as the second pillar, supporting the MI scores.

4.4.3 Combination of MI and CatBoost

Considering the various advantages and disadvantages of the computational methods of Mutual Information and CatBoost Importance,

we have decided to combine these two methods as they will provide a more robust feature selection method. We have normalised the scores from both MI and CatBoost and then subjected them to an average score to get the combined importance score. After combining the importance scores, we have sorted the features and selected the top 22 features to be fed to our predictive models.The dual-method strategy ensures that the selected features are not only mathematically important but also contribute significantly towards the predictive accuracy of the machine learning models.

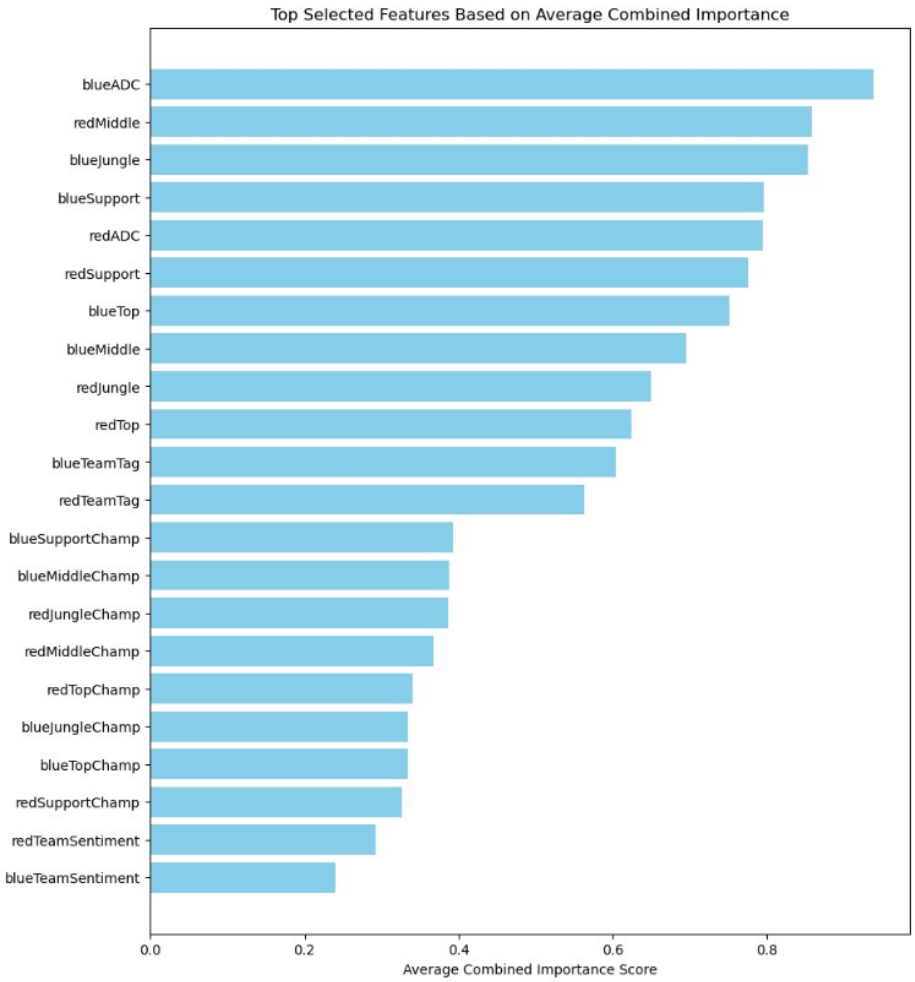

Figure 3: Feature selection based on combined importance scores

4.5 Model Formulations

The model formulation in our project aims to explain the problem in terms of mathematics to help understand how it can be solved using machine learning algorithms. This section for model formulation describes the hyperparameter tuning techniques, model training methodologies, and model evaluation metrics that are employed for all the nine machine learning models we have utilised in this project, including XG Boost, Random Forest, Naïve Bayes, SVM with Linear and Radial Kernels, KNN, Multi-Layer Perceptron, Logistic Regression with Elastic Net, and AdaBoost with Logistic Regression. Two different datasets were used to train the models and the corresponding results were observed and noted. The first dataset did not include the sentiment scores, and the second dataset included the sentiment scores. This dual approach helped us to carry out an in-depth understanding of the effect of sentiment analysis on the accuracy of each of the nine machine-learning models when using their best hyperparameters chosen from the three methods.

4.5.1 Hyperparameter Tuning

Tuning the hyperparameters is essential to any machine learning project, especially in our scenario where we have used nine machine learning models. The aim of carrying out hyperparameter tuning is to determine the most optimal set of hyperparameters. To perform hyperparameter tuning, we deploy three methods, namely, Grid Search, Random Search and Bayesian optimisation.

4.5.1.1 Grid Search

Grid Search performs hyperparameter tuning by executing an exhaustive search of all possible combinations in a hyperparameter grid. The range for the grid is provided while setting up the Grid Search, and the hyperparameters vary from model to model. We have implemented the Grid Search method using the Python library 'scikit-learn', which provides the 'GridSearchCV' function [40]. When a hyperparameter grid is provided, it evaluates all the possible combinations using

cross-validation and then provides the best parameters that lead to the highest performance.

4.5.1.2 Random Search

Random Search is another hyperparameter tuning method that samples the algorithm's parameters from a random collection of hyperparameters. It performs this process repeatedly for a given number of iterations that are set during the search process. Since Random Search randomly find hyperparameters in the hyperparameter space, it may search and find hyperparameter values that Grid Search did not assess earlier.

The Random Search method is computationally less expensive in comparison to Grid Search and takes very little time to find a good combination of hyperparameters. We have implemented Random Search to each of the nine machine learning models using the 'RandomizedSearchCV' function from the 'scikit-learn' Python library by providing the necessary input parameters for each model as the hyperparameters vary from model to model [41].

4.5.1.3 Bayesian Optimisation

Bayesian Optimisation is a probability model-based hyperparameter tuning method. Initially, it develops a model based on probability working in accordance with the hyperparameter values provided whilst initiating the Bayesian Optimisation search, which is then evaluated on the validation set. This process is followed by selecting the best hyperparameters that perform well on this function, and then the parameters for tuning each of the nine models are provided. By doing so, the algorithm saves computational cost, which makes it more efficient.

The reason why we use Bayesian Optimisation in our project is because it carries out hyperparameter tuning with the minimal number of iterations possible. This is crucial for complex models such as Neural Networks, where each evaluation can take a significantly large

amount of time to be performed. We have used the Python library 'BayesianOptimisation' from 'scikit-learn' to implement Bayesian Optimisation for each of the nine machine learning models. [42]

4.5.2 Models Training

After obtaining the best parameters from rigorous hyperparameter tuning, we have subjected the nine machine learning models to training using two datasets separately. The second dataset contains the 'blueTeamSentiment' and 'redTeamSentiment' columns that contain the sentiment scores of the blue teams and reds after sentiment analysis using PRAW and TextBlob. The first dataset contains all the columns in the second dataset, except the last two columns containing the blue team and red team sentiment scores. Our project involves 18 model training scenarios as we train nine machine learning models with and without the sentiment scores to evaluate the impact of sentiment analysis in LoL Esports analysis. Each model used in our project underwent hyperparameter tuning through Grid Search, Random Search, and Bayesian Optimisation, as detailed below:

XGBoost Model Training:

Our XGBoost model was trained using the XGBoost algorithm, which utilises the gradient boosting technique. We have trained our XGBoost model using the following hyperparameters explained in detail.

learning_rate: The individual impact of a tree on the resulting output of the model is known as the learning rate. The learning rate for our XGBoost model ranges from 0.01 to 0.2. We have chosen this range as a lower learning rate, such as 0.01, will help us avoid model overfitting and it is essential for our project where the prediction model has to perform well on unseen data. The upper value of 0.2 allows the model to learn faster, but we capped it to 0.2 as we must prevent our model from converging too quickly; hence, the range from 0.01 to 0.2 will keep it balanced.

max_depth: The max depth signifies how deep the trees of our XGBoost model are. The complexity of the XGBoost model increases as the trees become deeper, which might result in overfitting, and hence, we use it cautiously by setting the ranges from 3 to 10.[43]

Random Forest Model Training:

Our Random Forest model was trained using the Random Forest algorithm, which creates several decision trees during training. Later, it provides the mode of the classes from separate trees as the output. For this model, we focused on the following hyperparameters:

max_depth: The max_depth parameter refers to the random forest algorithm's maximum depth of individual decision trees. This hyperparameter controls how deep the model finds and detects the patterns in the data. The depth range we chose was between 2 and 4. The lower value of 2 makes sure that the trees are not too shallow, which would make the model too simple. The higher value 4 makes sure that the trees do not become too complicated, thus reducing overfitting. The League of Legends esports match data may contain complex information and noise in the data. Hence, a balanced depth is essential to capture the important features correctly.

n_estimators: This parameter is responsible for the count of trees in the random forest algorithm. More trees provide a robust model, but we might not gain significant results after a certain threshold, so balance is crucial. We have chosen the range from 100 to 300 trees for our project. 100 trees are commonly used as the starting range value for random forest models. This ensures the balance between computational efficiency and model performance. The upper limit of 300 increases the model complexity but does not affect the computational efficiency. Hence, we are using a well-optimized algorithm.[44]

Naïve Bayes Model Training:

Our Naive Bayes model was trained using the Gaussian Naive Bayes algorithm, which is particularly suited for data of continuous nature. Naive Bayes classifiers are mainly based upon Bayes' theorem. We have focused on the following hyperparameters for this model:

var_smoothing: The var_smoothing parameter is a supplementary part of the most significant variances of the features [45]. We chose a range of values between 1×10^{-11} and 1×10^{-7} for the `var_smoothing` parameter. For our project, we have selected this range to ensure stable calculations. Gaussian Naive Bayes uses the technique of probabilities and hence can be very sensitive to the size and spread of the data. By adding this smoothing factor, we avoid the zero probabilities.

SVM Model Training (Linear Kernel):

Our SVM model with a Linear Kernel was trained using the Support Vector Machines algorithm. The SVM algorithm is generally used for both classification and regression tasks. SVM attempts to find the hyperplane that best separates the classes of data and thus carries out the classification task [46]. The product of two vectors of two corresponding features in the given input space takes place in the linear kernel that we have used. The key hyperparameter we have tuned for this model is 'C'.

'C': For our model's 'C' value, we chose the values [0.1, 1, 10, 100][0.1, 1, 10, 100]. This is because the range is to cover the most commonly used regularisation strengths. A smaller value like 0.1 will give us a broader margin, but it might also lead to some wrong classifications, and this is also to prevent overfitting [47]. A larger value like 100 will give us a larger margin even if it classifies it as wrong in certain instances. Therefore, we chose this range to balance between underfitting and overfitting.

SVM Model Training (RBF Kernel):

Our SVM model with the Radial Basis Function Kernel was trained using the Support Vector Machines algorithm. We chose the RBF Kernel because it helps us handle non-linear classification problems. We used two key hyperparameters for tuning: 'C' and 'γ'.

C: The values we had chosen for 'C' were [0.1,1,10,100][0.1,1,10,100]. A smaller value like 0.1 may provide a smoother decision surface, but it might also lead to misclassification. A larger value like 100 will help us make the right classifications but may increase the model complexity and lead to outliers sensitivity. Hence, we chose these values to find the right balance to suit our project needs.

Gamma: Smaller values of 'γ' will produce a decision boundary with more flexibility, while larger values will produce a decision boundary with more rigidity. For our model scenario, we have chosen [0.001,0.01,0.1,1][0.001,0.01,0.1,1] to include both flexible and rigid decision boundaries.

KNN Model Training:

Our KNN model was trained using the K-Nearest Neighbors algorithm. In this algorithm, the predictions are made based on the class that majorly falls among its k nearest neighbours. The model was trained by incorporating various hyperparameters as explained below:

n_neighbors: This is the number of neighbours to be considered while making a prediction and is one of the most crucial parameters in KNN. We have chosen the range [3,5,7,9][3,5,7,9] for our KNN model. This is because limited neighbours will make the model very sensitive to noise in the data. On the other hand, a large number of neighbours will make it less sensitive, and hence, the optimal range was selected for our model.

weights: The weights parameter in the KNN model can be 'uniform' or 'distance'. In 'uniform', the weights are weighed uniformly, and in 'distance', the points close to each other have more impact. Hence,

we considered both of these options for our model's hyperparameter tuning.

algorithm: We have considered 'auto', 'ball_tree', 'kd_tree', and 'brute' in our model to ensure that we are not biased towards any specific algorithm and to obtain the results of all these algorithms during hyperparameter tuning.

leaf_size: For the leaf size, we chose the range [20,30,40][20,30,40] to make it optimal during hyperparameter tuning.

p: This parameter sets the power parameter for the Minkowski metric [48]. A value of 1 is equivalent to using Manhattan distance, and a value of 2 is equivalent to using Euclidean distance. Both were considered during the tuning process.

MLP Model Training:

For our Multi-Layer Perceptron model, we have trained it using the best hyperparameters after tuning. The various hyperparameters that we had used for tuning the MLP model are detailed below:

hidden_layer_sizes: The hidden_layer_sizes parameter holds the count of neurons in the hidden layer. We have used values [50, 100, 150] for this parameter. In order to take into account three different model complexity levels, such as low complexity, medium complexity and high complexity, we have chosen 50, 100 and 150, respectively.

activation: Regarding the activation function, we have considered both the 'relu' and 'tanh' activation functions. 'relu' is the Rectified Linear Unit, which is computationally more efficient, while 'tanh' is the Hyperbolic Tangent, which is better for negative activations [49].

solver: In our project, for the solver, which is the optimisation algorithm that is used to optimise the weights, we have considered 'sgd', which stands for Stochastic Gradient Descent and 'adam'. 'adam' works well for large datasets, while 'sgd' offers more options to alter.

alpha: In order to avoid overfitting our model, we have used the regularisation term, alpha. We have used [0.0001,0.001] as the range for alpha. Reducing the alpha value reduces the regularisation and vice versa. Hence, we have defined a range which finds the right balance between underfitting and overfitting.

Our Logistic Regression with Elastic Net regularisation model combines both L1 and L2 regularisations. We have considered the hyperparameters for tuning our model:

'C': The lower the 'C' value, the higher the strength of regularisation. Hence, we have chosen the range [0.1,1,10,100]. Setting the 'C' value very low will lead to underfitting, and setting it to very high will cause overfitting.

l1_ratio: We have set the l1 ratio parameter range of [0.0, 0.5, 1.0] to include L1 regularisation, L2 regularisation, and a combination of both L1 and L2.

max_iter: For the max_iter parameter, which is the maximum count of iterations for the solver to converge, we have set it to 1000 to allow the model to converge completely with enough count of iterations.

solver: We have used the 'saga' solver, which is good for large datasets and also offers support for Elastic Net regularisation [50].

AdaBoost (Logistic Regression) Model Training:

We have used the AdaBoost algorithm with Logistic Regression as the selected base estimator. The various hyperparameters tuned for the AdaBoost model are explained in detail below:

n_estimators: Athough increase in estimators leads to better model robustness, it might also cause model overfitting. Hence, for our AdaBoost model we have selected the range of [50,100,200]. This range can be experimented with as the performance of the model mproves.

learning_rate: We have chosen [0.01,0.1,1.0] as the range for our model's learning rate hyperparameter. A lower value such 0.01 makes the model more robust but increases the computation time of the model, while a higher value of 1.0 will reduce computation time but makes it less robust. So our range should balance these factors to find the best parameters.

4.5.3 Models Evaluation

For each of the nine different models used in our study such as XGBoost, Random Forest, Naive Bayes, SVM with a linear kernel, SVM with an RBF kernel, K Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP), Logistic Regression with Elastic Net, and AdaBoost (with Logistic Regression as the base estimator), we have followed a consistent strategy for model evaluation after the rigorous hyperparameter tuning process to obtain the best parameters. We have followed the following methodology to evaluate all of our models.

Parameter Conversion:

We subjected the hyperparameters to conversion to the correct formats required by the machine learning models, such as converting them to float or integer types as required.

Time Analysis:

We measured both the training and prediction times of each model with and without the sentiment scores to study the computational efficiency of each model and to evaluate the effect of sentiment analysis on League of Legends esports data.

Performance Metrics:

We have calculated the evaluation metrics, including accuracy, precision, recall, and F1-score, to study the model performance.

Results Compilation:

The calculated metrics and times were compiled with and without

sentiment scores for each of the nine machine-learning models.

Method Comparison:

We used the best parameters from the three methods, Grid Search, Randomised Search, and Bayesian Optimization, for hyperparameter tuning to compare and evaluate each model.We have taken into account several metrics to measure the performance for each of the nine machine learning models, not limiting to accuracy because accuracy alone cannot decide the performance of a model and hence we consider model computational efficiency by calculating the training time and prediction time as well.

By implementing uniform data preprocessing and feature selection techniques to all the nine machine learning models during sentiment dataset usage and without sentiment dataset usage, we make sure we perform a fair comparison of all the models and methods. After evaluating all the models, we compared the performance of the models using the best hyperparameters from each of the three hyperparameter tuning methods. This approach enables us to perform a wide range of analyses, ensuring that we understand how each model performs with different hyperparameter tuning methods and how they affect that performance with and without the sentiment scores involved.

Learning Curves:

The learning curves of the nine machine learning models were generated and analysed to check if they have been trained correctly and if they have converged as required. All the models were subjected to learning curve analysis and the training and validation curves were found to converge as the number of samples increased. The models were fine-tuned after selecting the best parameters from the three hyperparameter tuning methods and hence they were trained well. Most of the training and validation curves were close to each other.

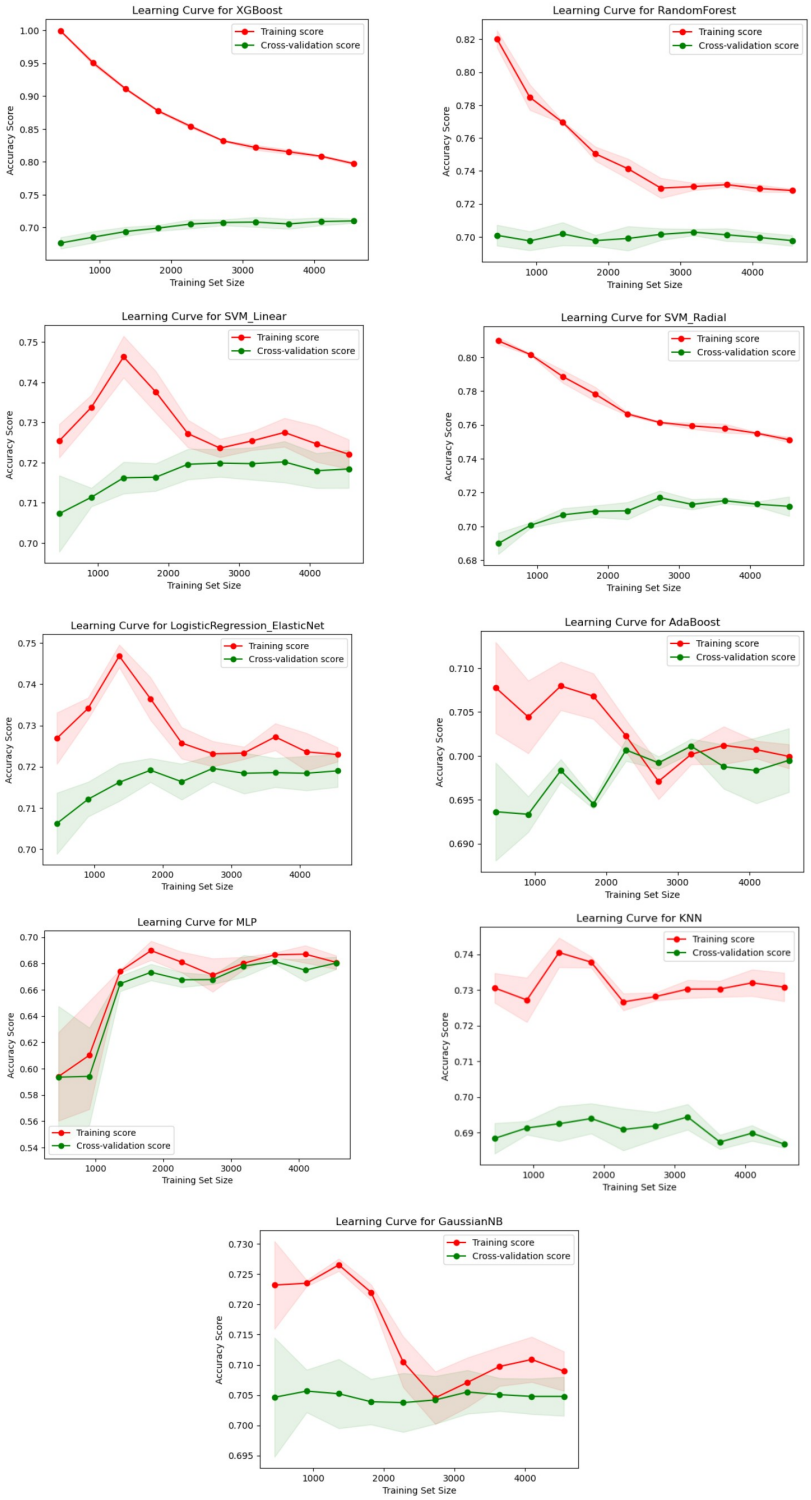

Figure 4: Learning Curves of the Machine Learning Models

4.6 Prediction Dashboard

Our League of Legends winner prediction dashboard is the highlight of our project. It is designed to provide an interactive and user-friendly interface for predicting League of Legends esports winners during the pre-match phase. Our dashboard is powered by our most accurate XGBoost model, with the best parameters obtained from the three hyperparameter tuning methods. The dashboard is built using a combination of Python libraries that includes Pandas, Seaborn and Matplotlib to perform data transformations and data visualisation. The library 'ipywidgets' has been used to create the widgets and interactive elements.

4.6.1 Input widgets

Data Loading:

Initially, our dashboard loads the saved XGBoost model pickle file along with the saved encoder and selected feature files from their saved paths. By doing so, we ensure that our dashboard makes use of the most updated information and data of models and encoders available to make accurate predictions.

Dropdown Creation:

Taking a first look at our dashboard, we can see that the first form of interaction with the dashboard is selecting options from the dropdown widgets that contain various menus including the blue team name, red team name, blue team champion names and red team champion names. These dropdown menus are created based on the unique teams and champions present in the dataset. These team dropdown menus allow the user to select from the available data and also avoid spelling errors while inputting the team and champion names. Thus, the user will have a wide range of real-world options that imitate the esports match scenario and this allows the stakeholders and esports team coaches to make predictions against their opponents or preferred teams.

Figure 5: Prediction Dashboard Dropdowns and Predict Button

Widget Layout:

Each and every dropdown menu in our dashboard has been aligned to make sure the entire text box is readable and to make it easier for the user to select from one of the many options available in each menu. The dropdown widgets contain team and champion selection menus. Selecting options from all these menus is mandatory and by doing so the user will increase the chances of the model predicting the right winner as it has more details to work with. At the end of the dropdowns, we have placed a 'Predict Winner' button. Clicking on this button after selecting the required team and champion details from each menu, the user can view the dashboard output.

4.6.2 Winner Prediction with Visuals

Once the options are selected from the menu by the user, our dashboard displays the predicted winner. Along with the predicted winner, the dashboard also displays many insightful visualisations customised to the Red team and the Blue team that the user selected. The visualisations include win/loss distributions of both the teams individually, the most played champions for each team, recent winning streaks for each team, and head-to-head win rates for the matches played by the teams against each other if they have played before with each other. These visualisations are generated by our dashboard specifically for the user based on the user's inputs provided.

Hence, the visualisations provide insights to the user which they can use to analyse the team performances and make more predictions to formulate game strategy against specific teams and champions.

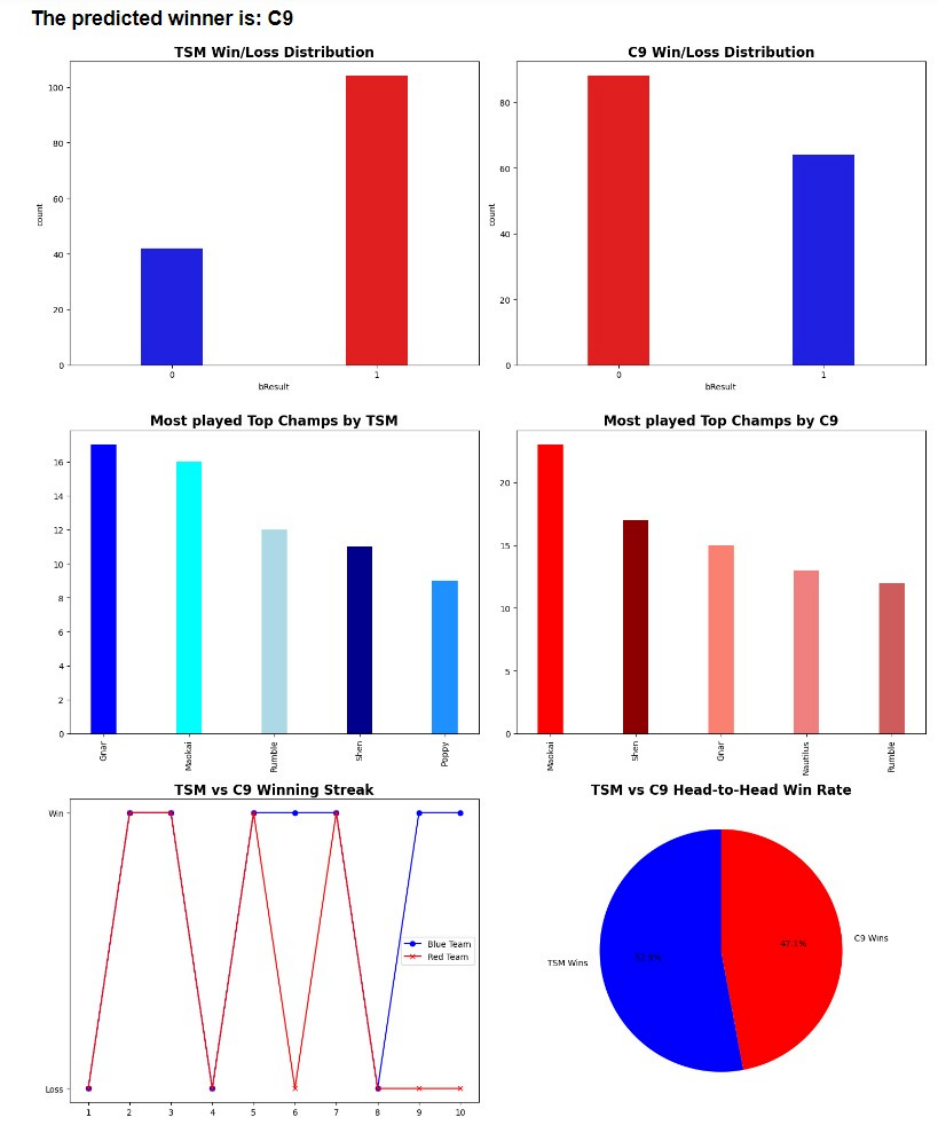

Figure 6: Dashboard Output with Customised Visualisations

4.6.3 Error Handling

Robustness:

After clicking on the 'Predict Winner' button, our dashboard collects the selected inputs from the user and performs logical checks such as the user selecting the same team name for both the Red and Blue team sides. If the user selects the same team names, the dashboard displays an error message saying that the blue team and red team cannot be the same and then displays the dashboard again, so that the user can reselect the team and champion options. Similarly, if two LoL teams have never faced each other in history, the dashboard displays a message on top of the pie chart slot stating that no head-to-head matches have been played by these teams before.

User Experience:

A readable and clear error message that states the issue will be displayed by the dashboard when it cannot make a prediction because of incorrect inputs from the user. This will help the user understand the problem and change their inputs right away. It will also ensure that the user is not confused or annoyed because of not knowing why the error occurs. This process enhances the overall user experience.

Efficiency:

As the XGBoost models is pre-trained and loaded already, our dashboard is ready to predict and is very efficient in terms of prediction time. This provides fast and accurate predictions and does not keep the user waiting.

Our League of Legends Esports Prediction Dashboard invloves the saved XGBoost machine learning model and provides a user-friendly interface, and acts as an esports analytics tool for stakeholders and esports teams. With robust error-handling methods, dropdown layouts and prediction logic, our dashboard provides a reliable solution for predicting League of Legends esports winners.

Chapter 5

Experimental Results

In this section for experimental results, we observe the evaluated results from our nine machine learning models. They contain the results of testing with sentiment scores included and without including the sentiment scores as well. The various experiments denote various hyperparameter tuning methods used to train the nine machine learning models in different scenarios. We have collected the results of the experiments to check and compare the performance of the nine machine learning models to predict the winner of the League of Legends esports matches and evaluate the effect of sentiment scores on the prediction output. The model evaluation metrics for each of the nine models, including accuracy, precision, F1-score, recall, training time and prediction time, are compared amongst each other. Finally, the experiment results of our work are compared with that of related existing literature.

5.1 Dataset

The dataset we used for our project contained both pre-match and in-game statistics of the blue and red teams from 2014 to 2018. Due to our project objective of creating a prediction dashboard which helps stakeholders and esports teams before they start the game, we have decided to discard the in-game statistics and consider only the pre-match statistics for our work. After subjecting the dataset to our previously explained data preprocessing methodologies, we ended

up with a refined dataset that contains 7581 instances in total. It contains 27 columns of, which comprise of the League, Year, Season, Type, blue team name, red team name, five champions each for the blue team and red team, five player names each for the blue team and red team and the final target variable that denotes the result of the match. The champion roles for each team include Top, Jungle, Middle, and ADC, which stands for Attack Damage Carry and Support. The above are contained in the first dataset. The second dataset contains all of the same data columns from the first dataset, and it also contains two extra columns that were updated after sentiment analysis, namely, blue sentiment scores and red sentiment scores.

5.2 Exploratory Data Analysis

Before carrying out data preprocessing, we had subjected the data to exploratory data analysis in search of finding valuable insights that would possibly contribute towards helping us build more accurate League of Legends esports winner prediction models. We had performed statistics computation for carrying out analysis tasks and to visualise the findings to find useful information for our model building process.

Initally, we summarised the dataset to obtain the information related to numerical features. The Year column ranged from 2014 to 2018 for which we calculated the standard deviation to be 0.85. This meant that there were a significant number of matches taking place in the year 2016. The target variable column, which is called 'bResult' holds the result of match based on blue team winning or losing. When we calculated the mean value of this column, we obtained 0.54 which indicates that the blue team has more number of wins than the red team.

After the initial exploration, we focused on the composition of teams based on the champions they selected and the team names. The win rates were calculated for both the blue and red teams for all the matches. The top performing teams were identified based on the win rate calculation. This information helped us understand the team

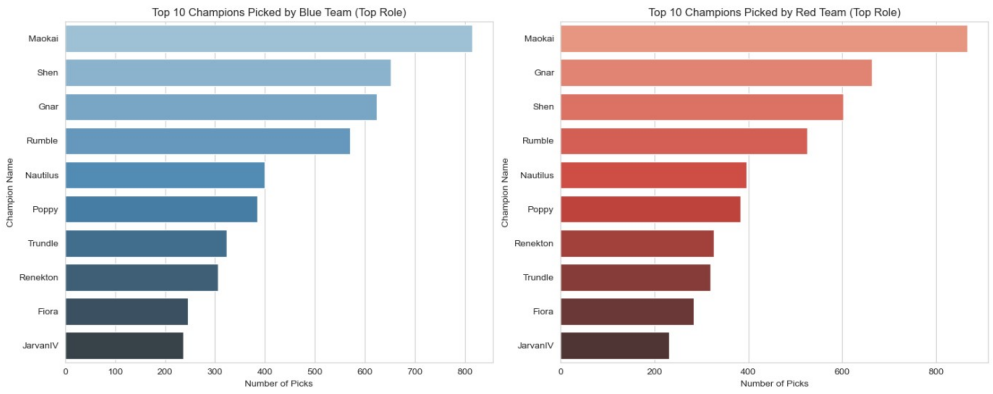

Figure 7: Top 10 Champions picked by the Blue and Red Teams

performance trends and contributed significantly towards the winner prediction model building process.We then created a heatmap to visualise the head-to-head wins between the top five performing teams. This heatmap provided valuable insights into how the top teams battle against each other. This could possibly be a significant aspect while predicting the outcome of the League of Legends esports matches.

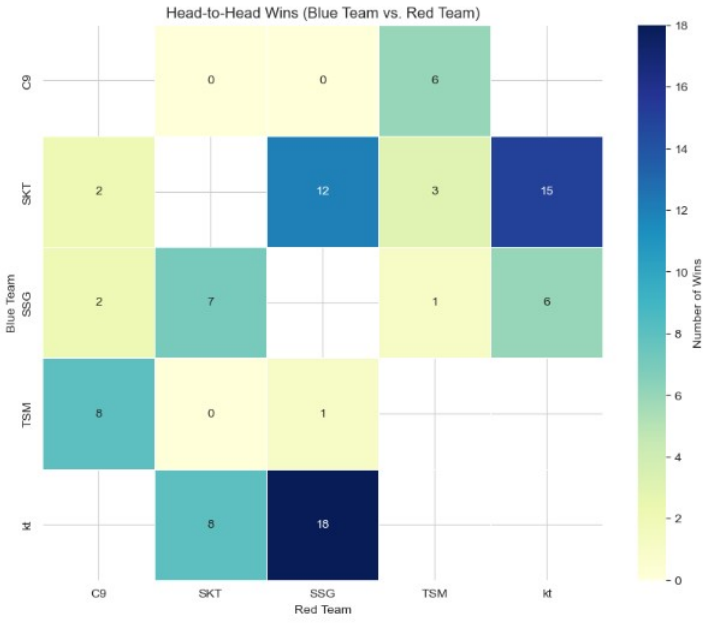

Figure 8: Heatmap for Head-to-Head Wins of top 5 teams

Using the computations carried out earlier, we developed a line plot that helped us visualise the performance of the top ten teams on a yearly basis and in terms of wins. This line plot provides us a clear understanding of the teams that are dominant over the years. This line plot also helps us understand the consistency of the top ten teams in terms of match performances over the years.

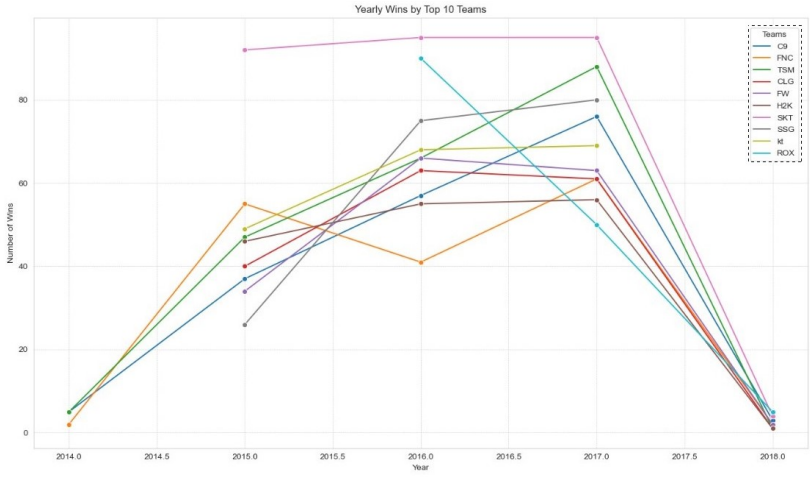

Figure 9: Line Plot for Yearly Wins by Top 10 Teams

5.3 Experiment 1: Grid Search Tuning

After examining the results from the Grid Search hyperparameter tuning carried out for the nine machine learning models with and without the sentiment scores, we can observe that our XG Boost Model had the highest accuracy in this experiment with 66.35% acccuracy, 68.6% precision and 72.38% recall.

At the same time, with just slightly lower accuracy than XGBoost, our MLP model performed better when the sentiment scores were included and obtained an accuracy of 64.11%, precision of 68% and recall of 66.4%. Similarly, our logistic regression model performed better when the sentiment scores were included and recorded an accuracy of 64.77%, a precision of 66.9% and a recall of 71.9%. The time

taken for Grid Search was substantially more than the time taken for Random Search and Bayesian Optimisation methods as it searched for the best parameters within the given range exhaustively to cover all the possible combinations.

Model	Sentiment Scores	Accuracy	Precision	Recall	F1- Score	Training Time (s)	Prediction Time (s)
XG Boost	Not Included	**0.663588**	0.686230	0.723810	0.704519	0.180879	0.006434
	Included	0.646438	0.669643	0.714286	0.691244	0.604659	0.009998
Random Forest	Not Included	0.635884	0.652542	0.733333	0.690583	1.077491	0.049905
	Included	0.634565	0.646817	0.750000	0.694598	0.653968	0.049994
Naïve Bayes	Not Included	0.634565	0.667447	0.678571	0.672963	0.000439	0.004463
	Included	0.630607	0.662791	0.678571	0.670588	0.006233	0.001998
MLP	Not Included	0.609499	0.613971	0.795238	0.692946	13.661889	0.001007
	Included	**0.641161**	0.680488	0.664286	0.672289	12.369790	0.020005
kNN	Not Included	0.621372	0.651481	0.680952	0.665891	0.003968	0.090266
	Included	0.612137	0.641892	0.678571	0.659722	0.004257	0.140374
SVM Linear	Not Included	0.645119	0.667406	0.716667	0.691160	2.485497	0.175507
	Included	0.643799	0.663755	0.723810	0.692483	4.183389	0.140504
SVM Radial	Not Included	0.637203	0.655246	0.728571	0.689966	3.158850	0.492031
	Included	**0.638522**	0.656652	0.728571	0.690745	3.164617	0.590000
Logistic Regression (Elastic Net)	Not Included	0.641161	0.664444	0.711905	0.687356	0.335531	0.010003
	Included	**0.647757**	0.669623	0.719048	0.693456	0.451731	0.009066
Logistic Regression (AdaBoost)	Not Included	0.631926	0.654267	0.711905	0.681870	3.544678	0.075990
	Included	0.630607	0.652838	0.711905	0.681093	3.300578	0.080008

Figure 10: Grid Search Experiment Results

5.4 Experiment 2: Random Search Tuning

Our Random Search experiment performed for the nine machine learning models yielded different results in comparison to the previous experiment. In this experiment, the Logistic Regression model obtained the highest model accuracy of 64.77%, a precision of 66.9% and a recall of 71.9%. This was when the sentiment scores were included and incorporated into the model. Even for our XG Boost and SVM RBF Kernel models, the performance metrics were relatively higher when the sentiment scores were included. However, the other models had better performance metrics when the sentiment scores were not included. Thus, for the second experiment with Random Search, sentiment scores tend to impact the accuracy of winner predictions in League of Legends esports matches.

Model	Sentiment Scores	Accuracy	Precision	Recall	F1- Score	Training Time (s)	Prediction Time (s)
XG Boost	Not Included	0.645119	0.669663	0.709524	0.689017	0.218117	0.010000
	Included	**0.645119**	0.667406	0.716667	0.691160	0.395839	0.005224
Random Forest	Not Included	0.635884	0.652542	0.733333	0.690583	1.218806	0.065305
	Included	0.634565	0.646817	0.750000	0.694598	0.624676	0.049996
Naïve Bayes	Not Included	0.634565	0.667447	0.678571	0.672963	0.005017	0.002009
	Included	0.630607	0.662791	0.678571	0.670588	0.003000	0.002104
MLP	Not Included	0.642480	0.684864	0.657143	0.670717	12.182818	0.009996
	Included	0.641161	0.680488	0.664286	0.672289	12.074345	0.019986
kNN	Not Included	0.620053	0.650685	0.678571	0.664336	0.002001	0.094153
	Included	0.608179	0.638202	0.676190	0.656647	0.020000	0.311360
SVM Linear	Not Included	0.645119	0.667406	0.716667	0.691160	2.525630	0.198934
	Included	0.643799	0.663755	0.723810	0.692483	4.233829	0.170529
SVM Radial	Not Included	0.637203	0.655246	0.728571	0.689966	3.129872	0.580118
	Included	**0.638522**	0.656652	0.728571	0.690745	3.011937	0.589993
Logistic Regression (Elastic Net)	Not Included	0.641161	0.664444	0.711905	0.687356	0.370898	0.002997
	Included	**0.647757**	0.669623	0.719048	0.693456	0.446298	0.003007
Logistic Regression (AdaBoost)	Not Included	0.631926	0.654267	0.711905	0.681870	3.655565	0.075078
	Included	0.630607	0.652838	0.711905	0.681093	3.666663	0.069997

Figure 11: Random Search Experiment Results

5.5 Experiment 3: Bayesian Optimisation

The Bayesian Optimisation experiment for hyperparameter tuning the nine machine learning models yielded similar results to that of the Grid Search hyperparameter tuning experiment. Our XGBoost model obtained an accuracy of 65.6%, a precision of 67.8% and a recall of 72.3% when the sentiment scores were not included. However, our SVM RBF Kernel model and SVM Linear Kernel models displayed better performance metrics when the sentiment scores were incorporated into their model training.

Similarly, our Logistic Regression model also performed better when the sentiment scores were included during training, and it obtained an accuracy of 64.9%, a precision of 67.11% and a recall of 71.9%. Considering all three experiments and comparing the time taken to run these experiments, this experiment using the Bayesian Optimisation technique was the most computationally efficient one as it saved a lot of time in the hyperparameter tuning process.

Model	Sentiment Scores	Accuracy	Precision	Recall	F1- Score	Training Time (s)	Prediction Time (s)
XG Boost	Not Included	**0.656992**	0.678571	0.723810	0.700461	0.180362	0.010000
	Included	0.651715	0.675676	0.714286	0.694444	0.442554	0.009993
Random Forest	Not Included	0.639842	0.656051	0.735714	0.693603	0.889263	0.047545
	Included	0.635884	0.647541	0.752381	0.696035	0.672626	0.049996
Naïve Bayes	Not Included	0.634565	0.667447	0.678571	0.672963	0.003999	0.001998
	Included	0.630607	0.662791	0.678571	0.670588	0.003004	0.002140
MLP	Not Included	0.641161	0.664444	0.711905	0.687356	9.703757	0.002768
	Included	0.642480	0.671264	0.695238	0.683041	8.001773	0.001007
kNN	Not Included	0.635884	0.661435	0.702381	0.681293	0.003002	0.118338
	Included	0.620053	0.646018	0.695238	0.669725	0.002977	0.221242
SVM Linear	Not Included	0.647757	0.668874	0.721429	0.694158	3.879811	0.171004
	Included	**0.647757**	0.666667	0.728571	0.696246	2.691170	0.169991
SVM Radial	Not Included	0.645119	0.660981	0.738095	0.697413	4.353293	0.447879
	Included	**0.651715**	0.668103	0.738095	0.701357	4.268463	0.550003
Logistic Regression (Elastic Net)	Not Included	0.643799	0.667411	0.711905	0.688940	0.489201	0.009804
	Included	**0.649077**	0.671111	0.719048	0.694253	0.319019	0.010003
Logistic Regression (AdaBoost)	Not Included	0.635884	0.651261	0.738095	0.691964	2.908391	0.066425
	Included	0.634565	0.651163	0.733333	0.689810	2.536727	0.049999

Figure 12: Bayesian Optimisation Experiment Results

5.6 Experiment 4: Comparison of Results

The final experiment is the comparison of our best models with the existing literature that is very closely related to this project. For this experiment, we have considered all our models by choosing their best hyperparameters and selecting them from the hyperparameter tuning method that obtains the best performance metrics for each model.

For our XGBoost model, we considered the hyperparameters from the Grid Search experiment. Our Random Forest model got its best performance from the hyperparameters chosen from the Bayesian Optimisation hyperparameter tuning experiment. The best performance for our Naïve Bayes model was observed in all three methods as they yielded similar results. Our MLP model showed the best performance during the Bayesian Optimisation experiment. Similarly, the best performance was seen for our KNN, SVM Linear, SVM RBF, Logistic Regression and AdaBoost models when their best hyperparameters were chosen from the Bayesian hyperparameter tuning experiment.

Model	Sentiment Scores	Accuracy	Training Time (s)	Prediction Time (s)
Naïve Bayes	Not Included	0.634565	0.000439	0.004463
	Included	0.630607	0.006233	0.001998
XG Boost	Not Included	0.663588	0.180879	0.006434
	Included	0.651715	0.442554	0.009998
SVM Linear	Not Included	0.647757	3.879811	0.171004
	Included	0.647757	2.691170	0.169991
kNN	Not Included	0.635884	0.003002	0.118338
	Included	0.620053	0.002977	0.221242
LR (Elastic Net)	Not Included	0.643799	0.489201	0.001098
	Included	0.649077	0.319019	0.010003
LR (AdaBoost)	Not Included	0.635884	2.908391	0.066425
	Included	0.634565	2.536727	0.049999
SVM Radial	Not Included	0.645119	4.353293	0.447879
	Included	0.651715	4.268463	0.550003
MLP	Not Included	0.642480	12.182818	0.009996
	Included	0.642480	8.001773	0.001007
Random Forest	Not Included	0.639842	0.889263	0.047545
	Included	0.635884	0.672626	0.049996

Figure 13: Model Accuracy on Test Partition of the Dataset

After selecting our best models trained based on the best hyperparameters obtained, we compare the performance metrics with the results of existing literature that are recent and related to our field of research work. The existing work that closely resembles our project is the research work done by Hitar-García et al., in which they have used the same dataset that we have also made use of in this project. They had tested out various models, of which their xgbStack model obtained the highest accuracy of 70.41%. However, they have considered only the pre-match data, and they have not incorporated sentiment analysis into their datasets. They had performed one-hot encoding to all of their categorical columns, which increased the computational time of their models during the model training process. Since we have also used some common models, we can compare our test accuracy with theirs, having used the same dataset. For the Naïve Bayes model, they got a test accuracy of 68.96%, while we got an accuracy of 63.45%. But, for the SVM Radial model, we have a better acccuracy of 65.17%, while they have obtained an accuracy of 64.07%. Although we have slightly lower accuracy in some of the models when comparing them

to their work, our models are computationally efficient and have reduced training time due to the target encoding method that we used during the data preprocessing process.

Model	Training instances	Test accuracy
RNN+ LR [White et al.]	70,194	0.7210
k-means + SVM [Ong et al.]	117,000	0.7040
Naive Bayes [Hitar-García et al.]	6826	0.6896
XG Boost [Hitar-García et al.]	6826	0.6882
svm Linear [Hitar-García et al.]	6826	0.6777
kNN [Hitar-García et al.]	6826	0.6658
XG Boost	**6822**	**0.6635**
glmnet [Hitar-García et al.]	6826	0.6605
glmboost [Hitar-García et al.]	6826	0.6579
SVM Radial	**6822**	**0.6517**
SVM Linear	**6822**	**0.6477**
LR (Elastic Net)	**6822**	**0.6490**
MLP	**6822**	**0.6424**
SVM Radial [Hitar-García et al.]	6826	0.6407
Random Forest	**6822**	**0.6398**
SimpleRNN (0-5 minutes) [Silva et al.]	5107	0.6391
kNN	**6822**	**0.6359**
LR (AdaBoost)	**6822**	**0.6359**
Naive Bayes	**6822**	**0.6345**
NeuralAC [Gu et al.]	603,760	0.6209
LR [Chen et al.]	208,090	0.6024

Figure 14: Comparison of Results with Existing Literature

Other related research works in this field that are related to our work share the same game, League of Legends, and have created winner prediction models as well. However, head-to-head comparisons cannot be made as they have considered the complete data of the esports matches, including the in-game statistics such as gold, game time and more. There are not many research works done with only pre-match data of League of Legends esports matches, especially those that incorporate sentiment analysis into their prediction models.

Chapter 6

Analysis and Discussions

This project is focused on creating a prediction model that predicts the winner of a League of Legends esports match based on the pre-match data and also by incorporating sentiment analysis. Even though many prediction models have been created with different forms of League of Legends esports data, not much work has been done by considering fan sentiment as a potential factor that affect performance or, in real-world scenarios, the mentality of the players.

Our model accuracies were slightly lower in some scenarios, as the sentiment scores might have induced a considerable amount of noise into the data. At the same time, certain models, such as the SVM RBF Kernel and MLP model, worked well and provided better performance metrics when the sentiment scores were included.

The novelty of our work lies in our methodology and the way we approach it, especially while carrying out data preprocessing. The biggest challenge in this dataset is that most of the features are categorical variables and not numerical features. Our most related works used One-Hot Encoding to convert the categorical variables into numerics. This increases the computational cost tremendously and also increases the time for computation, thus making it inefficient.

We have One-Hot Encoded the categorical variables with less number of unique categories, and we implemented Target Encoding to convert the other categorical variables that have a large number of

categories. This process makes way for an efficient data preprocessing techniques. For the feature selection method, we have combined the Mutual Information Scores and the CatBoost scores. This makes our feature selection robust and makes sure the right features are fed to the models during the training process.

Based on the results from all our experiments and analysis, we can conclude that this research work provides an in-depth analysis of nine machine learning models that are compared with and without sentiment analysis scores while creating a prediction model for a League of Legends esports match. In this scenario, the sentiment analysis has very little to no impact on the prediction model's performance metrics. This is due to the fact that the sentiment scores were obtained only from one social media platform such as Reddit, and was confined to just one post and its top five comments. This can be addressed by performing an in-depth sentiment analysis by gaining access to the Reddit developer portal with full access, allowing us to retrieve a significantly large number of posts and comments from many sub-Reddits.

One of the most notable achievements of our project is our interactive winner prediction dashboard. Our prediction dashboard works on the XGBoost model that obtained the highest accuracy of all the nine machine learning models that we trained, tested and evaluated. This dashboard provides significant insights to the stakeholders and can contribute towards the development of game strategy and formulate marketing plans based on fan sentiment, and the possibilities are endless. This dashboard makes our project unique as the foundation of the dashboard is built based on rigorous model testing and findings.

Chapter 7

Conclusion and Future Work

The findings from this work contribute to the rising field of esports and the application of machine learning in esports. For future works, the sentiment analysis can be expanded to multiple social media platforms and gaming communities to clearly capture the fan sentiments and perform in-depth sentiment analysis to obtain much more valuable insights and accurate sentiment scores that might improve the accuracy of the winner prediction models. The prediction dashboard can be further improved by adding vector graphics and more gamer-centric visuals of higher resolutions to attract more users from the gaming industry. This can also be the esports betting industry users or esports analysts who constantly work on improving the game strategy to increase the chances of winning. The prediction dashboard, which is the end point of this project, provides practical and intuitive insights to the stakeholders and esports organisations that would help them make informed decisions.

References

[1] Juan Agustín Hitar-García, Laura Morán-Fernández, and Verónica Bolón-Canedo. "Machine Learning Methods for Predicting League of Legends Game Outcome". In: *IEEE Transactions on Games* 15.2 (2023), pp. 171–181. DOI: 10.1109/TG.2022.3153086.

[2] Qiyuan Shen. "A machine learning approach to predict the result of League of Legends". In: *2022 International Conference on Machine Learning and Knowledge Engineering (MLKE)*. 2022, pp. 38–45. DOI: 10.1109/MLKE55170.2022.00013.

[3] Sang-Kwang Lee, Seung-Jin Hong, and Seong-Il Yang. "Predicting Game Outcome in Multiplayer Online Battle Arena Games". In: *2020 International Conference on Information and Communication Technology Convergence (ICTC)*. 2020, pp. 1261–1263. DOI: 10.1109/ICTC49870.2020.9289254.

[4] R. Ani et al. "Victory prediction in League of Legends using Feature Selection and Ensemble methods". In: *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*. 2019, pp. 74–77. DOI: 10.1109/ICCS45141.2019.9065758.

[5] Antonio Luis Cardoso Silva, Gisele Lobo Pappa, and Luiz Chaimowicz. "Continuous outcome prediction of league of legends competitive matches using recurrent neural networks". In: *SBC-proceedings of SBCGames* (2018), pp. 2179–2259.

[6] Lincoln Magalhães Costa et al. "Feature Analysis to League of Legends Victory Prediction on the Picks and Bans Phase". In: *2021 IEEE Conference on Games (CoG)*. 2021, pp. 01–05. DOI: 10.1109/CoG52621.2021.9619019.

[7] Dong-Hee Kim, Changwoo Lee, and Ki-Seok Chung. "A Confidence-Calibrated MOBA Game Winner Predictor". In: *2020 IEEE Conference on Games (CoG)*. 2020, pp. 622–625. DOI: 10.1109/CoG47356.2020.9231878.

[8] Joyanta Jyoti Mondal et al. "Does A Support Role Player really Create Difference towards Triumph? Analyzing Individual Performances of Specific Role Players to Predict Victory in League of Legends". In: *2022 25th International Conference on Computer and Information Technology (ICCIT)*. 2022, pp. 768–773. DOI: 10.1109/ICCIT57492.2022.10055689.

[9] Yang Yu et al. "Mining Insights From Esports Game Reviews With an Aspect-Based Sentiment Analysis Framework". In: *IEEE Access* 11 (2023), pp. 61161–61172. DOI: 10.1109/ACCESS.2023.3285864.

[10] Yogesh Chandra and Antoreep Jana. "Sentiment Analysis using Machine Learning and Deep Learning". In: *2020 7th International Conference on Computing for Sustainable Global Development (INDIACom)*. 2020, pp. 1–4. DOI: 10.23919/INDIACom49435.2020.9083703.

[11] Alexander Pak and Patrick Paroubek. "Twitter for Sentiment Analysis: When Language Resources are Not Available". In: *2011 22nd International Workshop on Database and Expert Systems Applications*. 2011, pp. 111–115. DOI: 10.1109/DEXA.2011.86.

[12] Ian Michael Urriza and Maria Art Antonette Clariño. "Aspect-Based Sentiment Analysis of User Created Game Reviews". In: *2021 24th Conference of the Oriental COCOSDA International Committee for the Co-ordination and Standardisation of Speech Databases and Assessment Techniques (O-COCOSDA)*. 2021, pp. 76–81. DOI: 10.1109/O-COCOSDA202152914.2021.9660559.

[13] Shandro Chakraborty et al. "Rating Generation of Video Games using Sentiment Analysis and Contextual Polarity from Microblog". In: *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*. 2018, pp. 157–161. DOI: 10.1109/CTEMS.2018.8769149.

[14] Sports Pro Media. *Esports industry Statistics*. 2023. URL: https://www.sportspromedia.com/analysis/Esports-industry-2023-gaming-league-of-legneds-csgo-esl-fortnite/?zephr_sso_ott=HxXiPJ.

[15] James P. Crone. "The Impact of the COVID-19 Pandemic on Esports Viewership Trends". PhD thesis. Georgia State University, 2022.

[16] Wikipedia. *League of Legends*. 2023. URL: https://en.wikipedia.org/wiki/League_of_Legends.

[17] Dot Esports. *LoL 2019 World Peak Viewership*. URL: https://dotesports.com/league-of-legends/news/league-of-legends-2019-worlds-peak-viewership.

[18] Medium. *The Rise of Esports: Transforming Gaming into a Global Phenomenon*. 2023. URL: https://medium.com/@robertgabriel3759/the-rise-of-esports-transforming-gaming-into-a-global-phenomenon-2cf199de57aa.

[19] League of Legends Wiki. *Summoner's Rift Map*. URL: https://leagueoflegends.fandom.com/wiki/Map_(League_of_Legends).

[20] Machine Learning Mastery. *XGBoost Introduction*. URL: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/.

[21] Analytics Vidhya. *Random Forest Introduction*. URL: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/#:~:text=Random%20forest%20algorithm%20is%20an,both%20classification%20and%20regression%20problems..

[22] Medium. *Accuracy, Precision, F1-Score, Recall*. URL: https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd.

[23] Towards Data Science. *SVM Linear Kernel Introduction*. URL: https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200.

[24] Wikipedia. *SVM Radial Kernel Introduction*. URL: https://en.wikipedia.org/wiki/Radial_basis_function_kernel.

[25] Analytics Vidhya. *K-Nearest Neighbours Algorithm*. URL: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/.

[26] Wikipedia. *Multi-Layer Perceptron*. URL: https://en.wikipedia.org/wiki/Multilayer_perceptron.

[27] Wikipedia. *Logistic Regression Elastic Net Regularisation*. URL: https://en.wikipedia.org/wiki/Elastic_net_regularization.

[28] Machine Learning Mastery. *AdaBoost Ensemble Method*. URL: https://machinelearningmastery.com/adaboost-ensemble-in-python/.

[29] Medium. *Accuracy, Precision, F1-Score, Recall*. URL: https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd.

[30] Kaggle. *League of Legends competitive matches 2014-2018*. URL: https://www.kaggle.com/chuckephron/leagueoflegends.

[31] Ioannis Pavlopoulos. *Aspect Based Sentiment Analysis*. Technical Report. Athens University of Economics and Business, 2014.

[32] Kaggle. *League of Legends competitive matches 2014-2018*. URL: https://www.kaggle.com/chuckephron/leagueoflegends.

[33] PRAW Library. *PRAW 7.7.1 documentation*. URL: https://praw.readthedocs.io/en/stable/getting_started/quick_start.html.

[34] Wikipedia. *One-Hot Encoding*. URL: https://en.wikipedia.org/wiki/One-hot.

[35] Scikit Learn. *Target Encoder*. URL: https://contrib.scikit-learn.org/category_encoders/targetencoder.html.

[36] Medium. *Mutual Information Score*. URL: https://bobrupakroy.medium.com/mutual-information-score-feature-selection-8eb19071664b#:~:text=The%20MI%20score%20will%20fall,the%20feature%20and%20the%20target..

[37] Catboost AI. *Catboost Feature Selection*. URL: https://catboost.ai/en/docs/concepts/python-reference_catboost_select_features.

[38] Scikit-Learn. *sklearn mutual-info-classif*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html.

[39] Catboost AI. *CatBoost Enables Fast Gradient Boosting on Decision Trees Using GPUs*. URL: https://catboost.ai/news/catboost-enables-fast-gradient-boosting-on-decision-trees-using-gpus.

[40] Scikit-Learn. *sklearn GridSearch CV*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[41] Scikit-Learn. *sklearn RandomSearch CV*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html.

[42] Scikit-Optimize. *sklearn Bayesian Optimization*. URL: https://scikit-optimize.github.io/stable/auto_examples/bayesian-optimization.html.

[43] Machine Learning Mastery. *XGBoost Gradient Boosting*. URL: https://machinelearningmastery.com/tune-learning-rate-for-gradient-boosting-with-xgboost-in-python/#:~:text=Tuning%20Learning%20Rate%20and%20the%20Number%20of%20Trees%20in%20XGBoost&text=The%20number%20of%20decision%20trees,scale%20from%200.0001%20to%201.0.&text=There%20are%205%20variations%20of%20n_estimators%20and%204%20variations%20of%20learning_rate.

[44] Towards Data Science. *Random Forests Guide*. URL: https://towardsdatascience.com/mastering-random-forests-a-comprehensive-guide-51307c129cb1#:~:text=n_estimators%3A%20The%20number%20of%20decision,3%2C%205%2C%20or%207.

[45] Scikit-Learn. *sklearn Naive Bayes Gaussian NB*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html.

[46] Analytics Vidhya. *Support Vector Machines Guide*. URL: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/.

[47] Scikit-Learn. *sklearn Support Vector Machines*. URL: https://scikit-learn.org/stable/modules/svm.html#:~:text=The%20parameter%20C%20%2C%20common%20to,a%20single%20training%20example%20has.

[48] Towards Data Science. *K Nearest Neighbours Classification*. URL: https://towardsdatascience.com/understanding-and-using-k-nearest-neighbours-aka-knn-for-classification-of-digits-a55e00cc746f#:~:text=p%20%3A%20It%20is%20power%20parameter,p%20if%20we%20want%20to.

[49] Built In. *ReLU Activation Function*. URL: https://builtin.com/machine-learning/relu-activation-function.

[50] Analytics Vidhya. *Comprehensive Hyperparameter Tuning Guide*. URL: https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/.

Appendices

Project Repository:
The coding files for our project are located on GitLab, which can be accessed via the following URL:

`https://git.cs.bham.ac.uk/projects-2022-23/kxs250`

Datasets:
This project utilises two datasets:

- `league_of_legends.csv` - The primary dataset containing team and champion features.
- `sentiment_scores.csv` - Similar to the primary dataset but includes additional columns for sentiment scores of the Blue and Red teams, updated after Reddit Sentiment Analysis using Python's PRAW and TextBlob libraries.

Folder Structure: The \main folder structure is as follows:

```
\main\
    \1_Data_Preprocessing_and_EDA\
        Data_Preprocessing_and_EDA.ipynb
    \2_Sentiment_Analysis\
        Sentiment_Analysis.ipynb
    \3_Models_Tuning\
        \1_XG_Boost\
            XG_Boost_Tuning.ipynb
            XG_Boost_Sentiment_Tuning.ipynb
        \2_Random_Forest\
            Random_Forest_Tuning.ipynb
            Random_Forest_Sentiment_Tuning.ipynb
        \3_Naive_Bayes\
            Naive_Bayes_Tuning.ipynb
            Naive_Bayes_Sentiment_Tuning.ipynb
        \4_MLP\
            MLP_Tuning.ipynb
            MLP_Sentiment_Tuning.ipynb
        \5_kNN\
            kNN_Tuning.ipynb
            kNN_Sentiment_Tuning.ipynb
        \6_SVM_Linear\
            SVM_Linear_Tuning.ipynb
            SVM_Linear_Sentiment_Tuning.ipynb
        \7_SVM_Radial\
            SVM_Radial_Tuning.ipynb\
            SVM_Radial_Sentiment_Tuning.ipynb
        \8_Logistic_Regression_with_Elastic_Net\
            Logistic_Elasticnet_Tuning.ipynb
            Logistic_Elasticnet_Sentiment_Tuning.ipynb
    \4_Models_Combined\
        Models_Combined.ipynb
        Models_Combined_Sentiment.ipynb
    \5_Saved_Model\
        encoder.pkl
        selected_features.pkl
        xgboost_model.pkl
    \6_Dashboard\
        Prediction_Dashboard.ipynb
```

Combined Models Code:
The main code files are:
`Models_Combined_Sentiment.ipynb` and `Prediction_Dashboard.ipynb`.

Hyperparameter Tuning:
The hyperparameter tuning files for all machine learning models can be found in \main\3_Models_Tuning\.

Saved Models:
The saved model files for our XGBoost model, including the encoder, features, and model file, are located in:
\main\5_Saved_Model\.

Prediction Dashboard:
The endpoint of this project, the prediction dashboard, can be found in:
\main\Prediction_Dashboard.ipynb.

Prerequisites

- Python 3.11
- Jupyter Notebook