**INFO 6105 Team 25 Report**

**Project Topic:** European Soccer Prediction

**Team Number:** 25

**Team Members:**

1. Akshita Ishvarbhai Barot; NUID: 002704943
2. Gauri Jaydeep Patole; NUID: 002700996

**Introduction:**

**Background:**

The UEFA (Union of European Football Associations) Champions League often known as European Soccer League, is a continental football competition that attracts the best soccer teams from across Europe. The competition, which is among the most renowned in the world of Soccer, initially appeared in 1995 under the name European Champion Club’s Cup. The UEFA Champions League has emerged as the premier event in European football and draws a huge international following.

**Motivation:**

Being one of the most popular sports on the planet, the European Soccer League attracts millions of fans from all over the world following their favorite clubs and players in this league with tremendous interest. Football fans would understand that it may get very difficult and tempting to anticipate the outcome of games. The European soccer dataset analysis offers an additional chance to learn more about the game, its background, and the different elements that affect how it is played. Also outcomes from any sports game can be quite difficult to predict, since Football in general have matches with fixed length of time, along with a single type of scoring: goals unlike other games such as Tennis, Badminton where the game is played until the player wins. The possible outcomes are also fixed in the game of soccer being either win, loss, or draw. Along with the different attributes involved in the game, it becomes lucrative to explore different predictive techniques to predict the outcome of the match.

**Goal:**

The aim is to use and compare prediction methods to predict the match outcomes and perform model evaluation comparing these models with the Bookies accuracy. Primarily, we will be collecting the data, and performing analytical procedures like Data Exploring and Cleaning of the dataset, followed by building suitable train and test pipelines of the Machine Learning algorithms. With the use of data from the European Soccer Database, this model will determine the outcome of a match (win, lose, draw). The possibilities are boundless, ranging from player performance to match results forecasting.

**Methodologies:**

This section presents the process undertaken during the formation of the prediction models, it elaborates the methods such as data collection, preparation along with the process of reducing, resampling the data, performing feature-selection and exploratory data analysis.

**Data Wrangling:**

Primarily we loaded the created a connection to connect the database, reading all the tables separately in different DataFrames in order to observe the data. After that we used the DataFrames, to observe the each table such as its shape, the information of each table, also checking if any DataFrames had any missing values, along with checking the uniqueness of data in each table.

Text

Description automatically generated

After observing the each DataFrame, we searched for missing values in the ‘Match’ table, since we found several columns with NULL values, we decided to drop them.

Table

Description automatically generated Graphical user interface, text

Description automatically generated with medium confidence

Upon further investigation, we found that there were several date columns stored as an object with the type String, we decided to convert these date columns into datetime using the Pandas function, one of which was performed in the ‘Match’ table where we needed to segregate the ‘season’ column into two columns as ‘start\_season’ and ‘end\_season’, dropping the initial columns ‘season’ and ‘date’ after conversion. We also converted the date column ‘birthdate’ in the ‘Player\_Attributes’, and the ‘date ’ column in the ‘Team\_Attributes’ to datetime.

Graphical user interface, text, application, chat or text message

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Since our dataset has relationships between tables, we also performed tasks such as joining the tables on their ids and merging them in order to get only certain data which were needed, and dropping all the extra ids in the ‘Match’ , ‘Team’, ‘Team\_Attributes’, ‘Player’, ‘Player\_Attributes’ and ‘League’ table.

Further, we want to know who wins/loses or whether there is a tie between teams, for that we created a function to determine these and storing them in a new column named ‘winner’ in the ‘Match’ DataFrame.

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Next step we performed was cleaning our data, we checked if there were any duplicate values in the ‘Team’ and ‘Team\_Attributes’ DataFrame. For that we defined a function called ‘unique\_team\_names’ which returned the count of occurrences of each unique Team name in the ‘Team’ DataFrame column ‘team\_long\_name’, and displayed them. If the teams had the same ‘fifa\_api\_id’ we decided to drop them.

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

**Exploratory Data Analysis:**

Now that we have our data cleaned and processed, we decided to answer some questions with our data.

**Q1. Which top 10 teams had the most victories at home in 2015/2016 Season?**

It is important to keep in mind the most recent changes made to the ‘Match’ table. Our focus should be on four specific columns: ‘start\_season’, ‘end\_season’, ‘home\_team\_name’, and ‘winner’. We queried the matches that occurred during the period of 2015-2016, and plotted the Top 10 winning teams along with their number of wins in the Season of 2015-2016. We found that there was a tie between four teams with 16 wins.

**Graphical user interface, text, application

Description automatically generated**

**Chart, bar chart

Description automatically generated**

**Q2. Which season had the most home and/or away goals over the seasons?**

We will start by summing the number of home team and away team goals, while grouping the ‘start\_season’, ‘end\_season’ in order to calculate the number of goals scored per season. With the number of goals obtained we will calculate ‘home\_team\_goal’, ‘away\_team\_goal’ per seasons and plotting them to compare the difference between the home team and away team goals

**Graphical user interface

Description automatically generated**

**Chart, bar chart

Description automatically generated**

**Q3. What Team Attributes leads to a Team's Victory?**

To address the question, we need to identify the teams with the highest number of wins. To accomplish this, we examined the match table specifically for the 2015-2016 season, while checking only the columns with numerical data. The chart below provides attributes of the winning teams displaying 9 distinct characteristics of the top 10 winners during the 2015-2016 season.

Graphical user interface, text, application

Description automatically generated

Chart, bar chart

Description automatically generated

**Feature Engineering:**

After completing the data cleaning, and performing Exploratory Data Analysis, the next process to select the import features is important.

First we started with the ‘Team’ data, we created a feature function ‘From\_Home\_team’ that returns the sum of the last 10 matches played by the home team before the current match, based on the ‘Result’ column, performing the same tasks for the away team, and populating ‘Home\_Team\_form’ and ‘Away\_Team\_form’ using a ‘lambda’ function.

Text

Description automatically generated



Secondly, we created feature functions on ‘Attacking Form’, ‘Defensive Form’, ‘Head to Head’ by in the same way for home and away team and populating columns by using ‘lambda’ function.

Graphical user interface, text

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Text

Description automatically generated with medium confidence

**Visualizing our Target Variable: Match Results**

Here we visualized our target variable, using the seaborn ‘countplot’ on the ‘Result’ column which represented the outcomes such as ‘Home Win’, ‘Draw’, and ‘Home Loss’ of soccer matches.

Chart, bar chart

Description automatically generated

Since there are different types of player positions in a team, we started with exploring the ‘Player\_Attributes’ all the players as per their playing positions. We used the seaborn ‘distplot’ to first plot the graph for the ‘overall\_rating’ and ‘potential’ for all the players. Next, we moved on to check as per the player positions that are Goalkeeper, Defender, Midfielder, and Striker. Here we have showed the plot for one of the position that is Goalkeeper.

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

After breaking down the Player Attributes by the Player positions, we discarded the attributes which were similar as per the graph result and stored the required attributes in a separate DataFrame as per the Players. Also creating DataFrames for each Player Attribute, and then categorized them under same time period 2008-2011, 2012-2014 and 2015-2016. Later we merged all the Home team and the Away team players as per the same time period.

We also made changes to the ‘Team\_Attribute’ by merging them by the ‘team\_api\_id’ to the match same time period, also dropping columns not required for our model in the process.

A picture containing calendar

Description automatically generatedA picture containing calendar

Description automatically generated

Table

Description automatically generated

**Data Modeling:**

We started with splitting our data into two categories that is the Training data and the Test data. For the Training data, we took into account all the Matches played between the year 2008 and 2014, and for the Test data, we were taking into account the Matches played between the year 2015 and 2016. We started off with dropping all the null values from our Training and Test data, and dropping the columns which were not required, and then feeding it to our training and testing set.

Text

Description automatically generated

**Logistics Regression:**

A Logistics Regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. It works similarly to Linear regression except for the relation function and the types of the class attributes used. The first step we did to implement this model was to create a pipeline with preprocessing and hyperparameters tuning using Scikit-Learn.

The first step is to create a preprocessing object that includes StandardScaler features and PCA. This is done using the FeatureUnion class from scikit-learn, which combines several transformers into a single transformer. In this case, the "std" transformer applies standard scaling to the data, the "pca" transformer performs PCA dimensionality reduction, and the "F\_value" transformer performs feature selection based on F-value. Next, a pipeline is created using the Pipeline class from scikit-learn. The pipeline consists of the preprocessing object created earlier and a logistic regression classifier. A space of candidate values is then defined using a dictionary. In this model, the hyperparameters for PCA, feature selection, logistic regression penalty, and C (inverse of regularization strength) are specified, and their values are given. A grid search is created using the GridSearchCV class from scikit-learn. The grid search fits the pipeline to the training data and evaluates the model performance using cross-validation with five folds. The best combination of hyperparameters is selected based on the mean cross-validation score. Finally, the fitted grid search object is stored in the variable "log\_reg\_test".

Next step, was to obtain the best parameters, and the best cross-validation score for the training data and also obtain the Logistic Regression score for the test data.

Lastly, we plotted the Logistic Regression Confusion Matrix using the Seaborn's heatmap function to visualize the performance of a logistic regression model with three classes: "Home Loss", "Draw", and "Home Win". The diagonal cells of the matrix represent the correct predictions made by the model, while the off-diagonal cells represent the incorrect predictions. The values in each cell of the matrix represent the number of instances where the predicted label matched the true label.

Graphical user interface, text, application

Description automatically generated



Chart

Description automatically generated

**K-Nearest Neighbors:**

K-Nearest Neighbors is a non-parametric algorithm, it does not make any assumptions about the underlying distribution of the data. In the KNN algorithm, the "neighbors" are the K closest points in the training data to the new data point that needs to be classified or predicted.

Similar to Logistic Regression, we first started with creating a pipeline for preprocessing object modeling data, and perform a grid search to find the best hyperparameters for a K-nearest neighbors (KNN) classifier. Next, we created a FeatureUnion object that will apply two preprocessing steps to the data: standard scaling (using the StandardScaler() function) and principal component analysis (PCA, using the PCA() function). We create a pipeline that combines the preprocessing steps by using the KNeighborsClassifier() function. We specified a range of values to try for two hyperparameters: for PCA step (either 85 or 80) and the number of nearest neighbors to consider in the KNN classifier (either 15, 16, or 17). Next we created, a GridSearchCV object, which will perform a grid search over the search space defined in search\_space. The GridSearchCV object fits to the training data x\_train and y\_train, and finds the best combination of hyperparameters that maximizes the cross-validation accuracy.

Next step, was to obtain the best parameters, and the best cross-validation score for the training data and also obtain the K-Nearest Neighbors score for the test data.

Lastly, we plotted the K-Nearest Neighbors Confusion Matrix using the Seaborn's heatmap function to visualize the performance of the model similar to logistic regression model.

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Chart

Description automatically generated

**Random Forest:**

Random Forest is an extension of decision trees, which involves constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. The "random" part of the name comes from the fact that each tree in the forest is constructed using a random subset of the training data and a random subset of the input features.

We used Grid Search Cross Validation to tune hyperparameters for a Random Forest Classifier model. First, the Random Forest Classifier model is instantiated with a random state of 0. Next, a hyperparameter grid is defined, which specifies a range of values to search over for the number of decision trees in the forest (n\_estimators), the maximum depth of the decision trees (max\_depth), the minimum number of samples required to split an internal node (min\_samples\_split), and the minimum number of samples required to be at a leaf node (min\_samples\_leaf). Then, a Grid Search model is instantiated with the Random Forest model, the hyperparameter grid, and a 5-fold cross-validation. Grid Search will search for all possible combinations of hyperparameters in the grid, and evaluate each combination using 5-fold cross-validation on the training data. The Grid Search model is then fit to the training data using the fit() method. After Grid Search is complete, the best hyperparameters are printed to using the best\_params\_ attribute. The best model found by Grid Search is used to make predictions on the test data, and the predictions are stored.

Next, we find the best cross validation score for the test data, to evaluate the model performance on the test data.

Finally, we visualize the confusion matrix for a Random Forest Classifier model using the Seaborn library.

Text

Description automatically generated

Chart

Description automatically generated

**Linear Regression:**

Linear regression is a statistical method used to model the relationship between a dependent variable (also known as the response variable or target variable) and one or more independent variables (also known as the predictor variables or features).

First, we instantiate the Linear Regression model using the LinearRegression() class from Scikit-learn. Next, the model is fitted to the training data using the fit() method, which takes the independent variables (x\_train) and dependent variable (y\_train) as input. After that, the model is used to make predictions on the test data using the predict() method, which takes the independent variables (x\_test) as input and returns the predicted values for the dependent variable. Finally, the performance of the model is evaluated on the test data using the score() method, which calculates the R-squared (coefficient of determination) score. The R-squared score is a measure of how well the linear regression model fits the data, with higher values indicating a better fit.

In this case, the R-squared score is 0.1843, which suggests that the linear regression model explains only a small portion of the variation in the dependent variable.

**Graphical user interface, text

Description automatically generated**

**Datasets:**

**Dataset link:** <https://www.kaggle.com/datasets/hugomathien/soccer>

The data was collected for eleven European League some of the popular one’s are English Premier League(EPL), Spanish League(La Liga), Germany League(Bundesliga), the dataset is stored in the form of ‘sqlite’. The dataset consists of roughly 25,000+ football games and 10000+ players from the top football leagues of 11 European nations. In addition, it includes match data such as scores, corners, fouls, team formations, player names and a pair of coordination indicating their location on the field and covers seasons from years 2008 to 2016. Furthermore, the database also consists of betting odds from up to 10 providers, detailed match events, players, team attributes and team line-up with squad formation. The below table shows the total number of rows and columns of each table in the dataset.

Details of Tables included:

|  |  |  |
| --- | --- | --- |
| Table | Total Rows | Total Columns |
| Country | 11 | 2 |
| League | 11 | 3 |
| Match | 25979 | 115 |
| Player | 11060 | 7 |
| Player\_Attributes | 183978 | 42 |
| Team | 299 | 5 |
| Team\_Attributes | 1458 | 25 |

**Results and Analysis:**

**Model Evaluation:**

For Model Evaluation, we are evaluating three models, Logistic Regression, K-Nearest Neighbors, and Random Forest. We will not be using the Linear Regression model, since the model explains only a small portion of the variation in the dependent variable as per the R-squared score of 0.1843.

After comparing these model we find that the Model Accuracy Score of Logistic Regression is better than K-Nearest Neighbors, and Random Forest. We also visualized our evaluation with a bar graph comparing all the three models and their accuracy score.

Text

Description automatically generated

Chart, bar chart

Description automatically generated

**Model Accuracy vs the Bookies Accuracy:**

To check our model accuracy, we will compare our model accuracy with the bookies prediction data which was available in our dataset. We start with performing the merging of League name to Match for per League Analysis and adding a column for Total Matches. Next, we created a function to check the bookies prediction and if their scores are correct or not. We returned 1 for a correct prediction result and a 0 for an incorrect prediction result for the bookies. Further, we use our predicted result from the Logistic Regression model and also create a function to check whether the model prediction was correct or not. For comparison of the models, we created additional columns, ‘Bookies\_score’ and ‘Model\_score’, and then group the data by League and aggregate both the columns and store it to new DataFrame.

Graphical user interface

Description automatically generated with medium confidence

As per our result, we can see that in most of the cases the Model\_score is better than the Bookies\_score.

Finally, we start with calculating the model and the bookies accuracy using the Bookies\_score, Model\_score and Total\_Matches to plot the Accuracy Scores per League to the Bookies Prediction, and find that our Model accuracy is greater than the Bookies accuracy by 47.76 basis points.

Graphical user interface, text, application, Word

Description automatically generated

Chart, bar chart

Description automatically generated

Text

Description automatically generated with low confidence

**Conclusion:**

Using a combination of FIFA statistics and engineered features, we have built the model to predict the outcomes of the soccer matches. Here, the model with the highest accuracy score on the test match dataset was Logistic Regression model, next was Random Forest and then K-Nearest Neighbors model. We also found out that Linear Regression does not really work with our dataset, as there are only a small number of variations in the dependent variable. We ended with comparing the model accuracy and the bookies accuracy, and we found that our model accuracy is greater than the bookies accuracy by 47.76 basis points.

**References:**

1. <https://www.kaggle.com/code/yoyocm/how-predict-the-outcome-of-48-matches/notebook>
2. <https://medium.com/@rolandshum.shc/neural-networks-football-result-prediction-d8b0f933118b>
3. <https://jiayiwangjw.github.io/2017/06/12/SoccerDataAnalysis/>
4. <https://www.learnpythonwithrune.org/how-to-use-sqlite-database-with-pandas/>
5. <https://github.com/ozlerhakan/soccer/blob/master/investigate-a-dataset-template-main.ipynb>
6. <https://github.com/Ziggaman/Soccer-Match-Prediction>
7. <https://scikit-learn.org/stable/>