Final Report

Google Colab Notebook	GitHub Repository

Introduction and Background:

Soccer betting is a multi-billion-dollar industry, offering fans and enthusiasts a thrilling and financially rewarding way to engage with the game. With enormous amounts of historical data and a myriad of influencing factors, soccer betting is an enticing arena for the application of predictive modeling machine learning techniques. Our project focuses on the development of such a model, shedding light on the potential for data-driven success in English Premier League betting.

In terms of literature review, there has been significant work in this field including outcome prediction, player performance analysis, injury prediction, and tactical analysis. Traditional statistical models such as logistic regression and Poisson regression have been widely used for outcome prediction. Machine learning techniques, including but not limited to decision trees, random forests, support vector machines, and neural networks, have been applied to soccer outcome prediction. These models can capture complex relationships between various input features and match results.

Problem Definition:

We want to predict the outcome of future soccer matches in the English Premier League. Given any two participating teams as an input, our model will output the predicted result of the match along with a certain percentage confidence of the prediction. We have narrowed the scope of our problem since the project proposal phase for two main reasons. Firstly, upon further exploration of the international soccer matches database we noticed that most international matches are friendly matches, which is not a good indicator of team performance as the stakes are not very high. Secondly, we noticed that the EPL dataset contained more metrics on player performance which we wanted to incorporate into our model.

Motivation:

Soccer is inherently unpredictable due to the dynamic nature of the sport. Unexpected events, such as injuries, red cards, or referee decisions, can have a profound impact on match outcomes, making it challenging to build accurate predictive models. Managing and selecting the most relevant features for prediction can be complex due to the high dimensionality of the data.

We decided to embrace these challenges in light of the popularity of the sport and potential commercial applications, such as sports betting and fantasy sports.

Data Collection:

We used the European Soccer Database from Kaggle to train our model on English Premier League match results from 2009 to 2017. We specifically used the following tables:

- Matches - Data regarding all premier league matches played between 2009 and 2017, this includes team matchups, date of match, lineups and result.
- Player_Attributes - overall FIFA ranking of the player at the given time stamp
- Player - key value pairs of player API_Id and name
- Team - key value pairs of team API_Id and name

In terms of data cleaning and preprocessing, we started by pruning our data by dropping all null values and duplicates. After dropping duplicates, we reset the dataframe indices to 0 to align the data and make the dataframe contiguous. We filtered the data to only retain all EPL matches, and not other European matches. We then proceeded to select relevant columns of the dataset needed to train the data as this was computationally advantageous and aided in feature building and feature selection.

Methods:

Feature Building

We used the selected columns of the dataset to build the following composite features that would be used to train our model:

- Home/ Away advantage - crowd support and familiarity effect can give the home team a significant advantage over the away team.
- Scoreline - the scoreline of previous matches, including the difference between the number of goals scored by the winning and losing team is an important quantitative factor to consider to account for the 'margin of the win'.
- Player Lineup - The composition of players playing for each team has a significant impact on the outcome of the match. We can consider individual player analytics such as expected goals, expected assists, goal-creating actions, and passes per defensive action. These player analytics are correlated to the scoreline of a match which directly impacts the outcome of any given match.
- Team rankings - The overall team ranking within the league gives a basic indication of which team is likely to win based on previous performance.
- Season Win Percentage - The percentage of matches won by a team in a particular season is a good indicator of their form and can help predict how likely they are to win

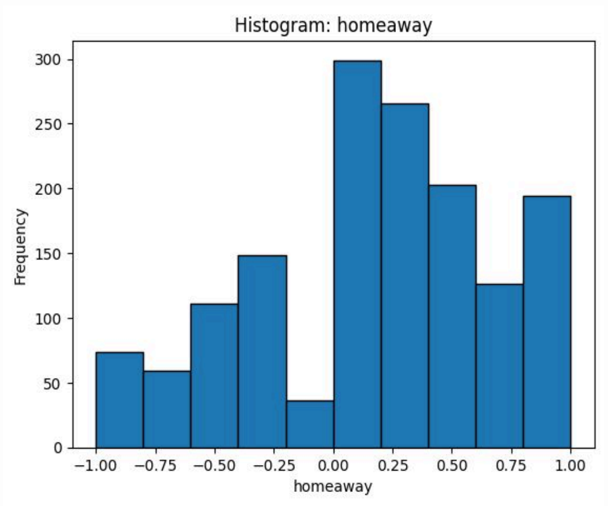
any given match.

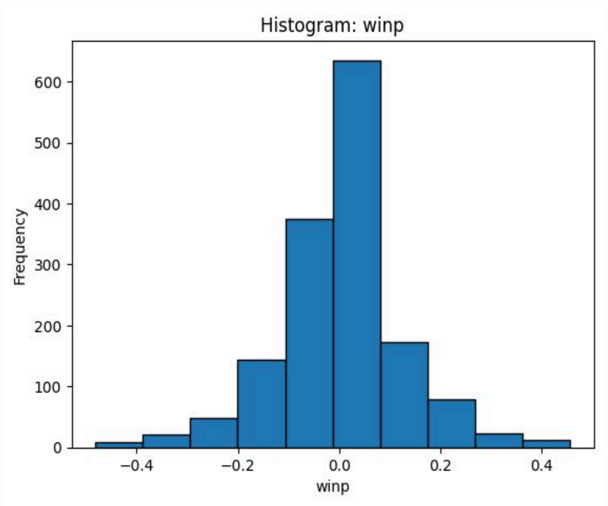
- FIFA Ratings - FIFA is an interactive soccer video game developed by EA Sports that uses complex algorithms to best estimate a particular player's overall rating based on their performance in real life. We used the time series data on FIFA ratings in the Player_Attributes table to calculate the average rating of the team using the most up-to-date player ratings of each player from the lineup at the timestamp right before the match.
- Team impact - To calculate the 'impact' of a particular player, we first calculate the proportion of matches won by a team with that player in their lineup, over the total number of matches won by a team. We then summed up the 'impact' of each player to get the overall impact of the team.

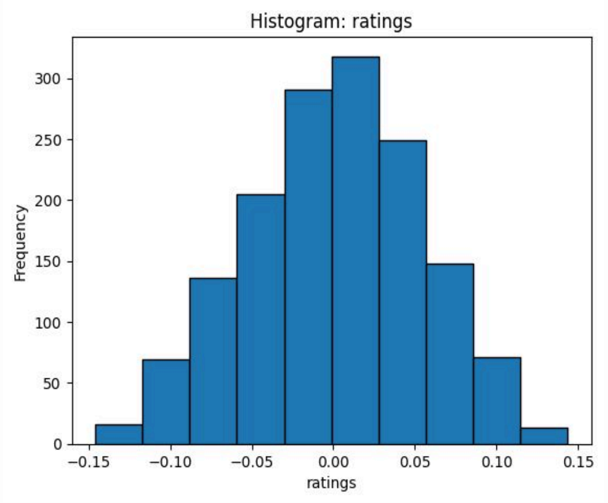
We normalized each feature on a scale from -1 to 1 for feature selection.

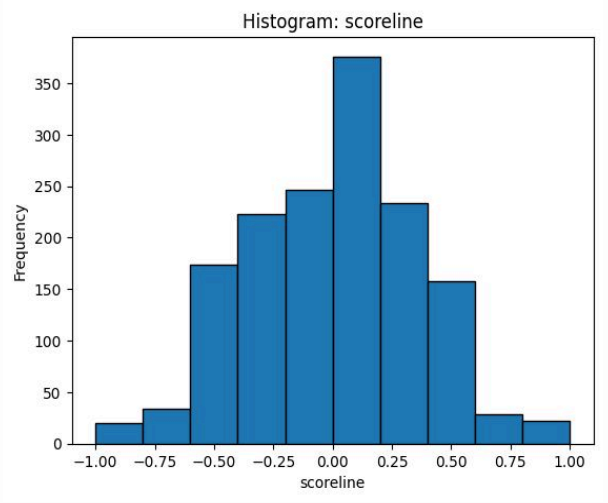
Feature Visualization

We visualized the data in two ways: First we plotted histograms representing the distribution of values for each of the features. The distribution can tell us whether the feature is roughly uniform - each value within the possible range of that feature has an equal chance of happening - or more like a Gaussian distribution.

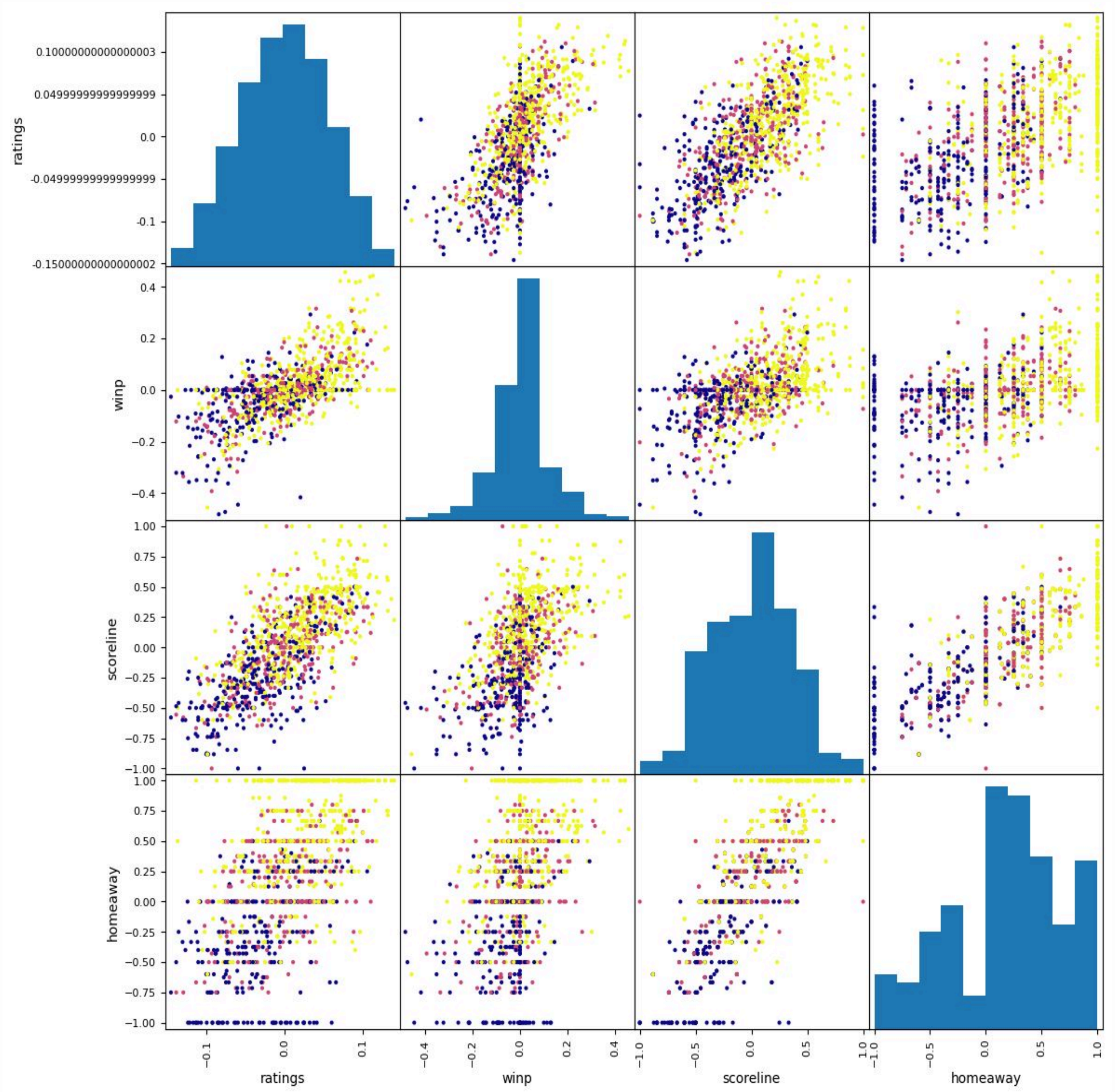








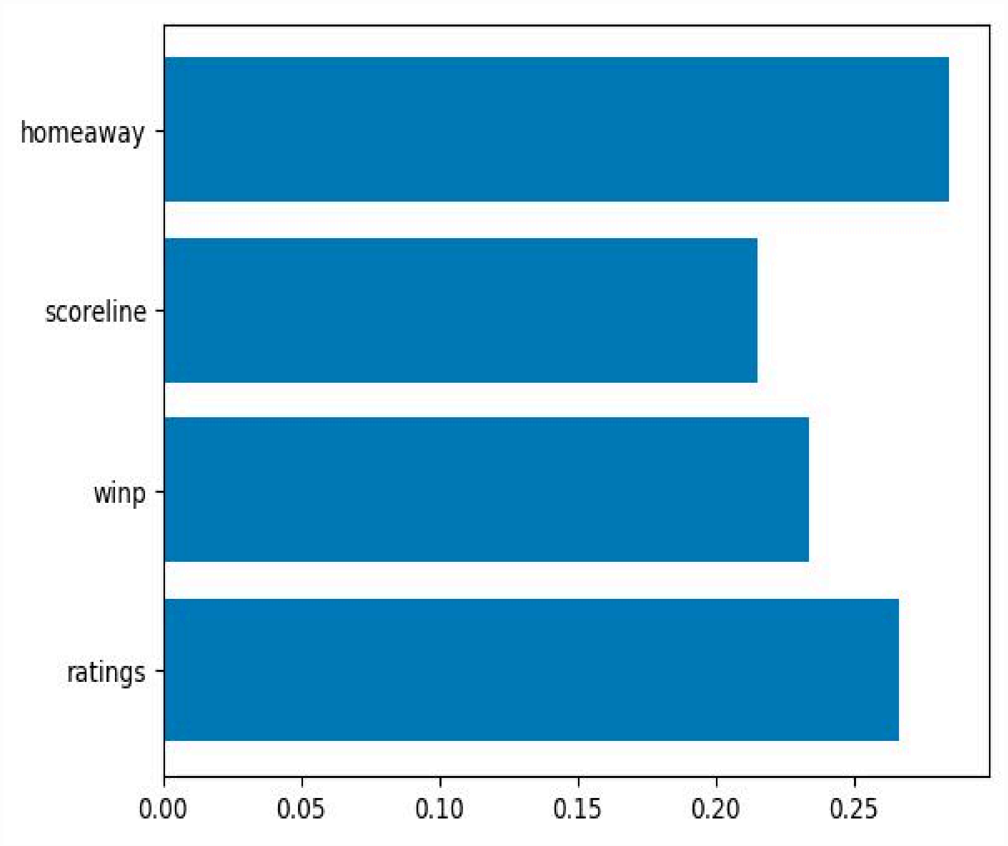
We then created scatter plots with each feature plotted against the other features. This helped us find correlations between various features such as how the combination of player ratings and home/away advantage affects the outcome of the match. Using these correlations, we narrowed down what features may be more useful in our predictive algorithms. We also used these graphs to see how independent our features were - if there are any strongly visible correlations between two features, that might indicate that those two features are more dependent on each other. These graphs are shown below, with yellow representing a win, pink representing a tie, and blue representing a loss (for the home team).



Feature Selection

While building a model, it is crucial to choose relevant features out of a given set of features. Now, we can qualitatively do that to an extent based on our instinctive understanding of the topic. We handpicked 6 features that we thought played an impact on our output. However, it is difficult for humans to tell what features are more important than others and weighing out several features in a model can be computationally intensive and more importantly lead to a lot of randomization while testing and overfitting. So, we had an ML model, namely Random Forest Classifier, interpret our data and do it for us!

A Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. The basic idea behind a Random Forest is to build a diverse set of decision trees and then combine their predictions to create a more robust and accurate model. The classifier was able to look at the data and "SelectFromModel" which features are most important. Using this, we ran a boolean mask to give us all the features that we considered to be the most important (essentially, features that were given a score >= threshold, which in this case was mean were considered). We ran the classifier a few times and eventually we found that FIFA Ratings and Home/Away advantage played the most important role in determining score lines. To that end, we took these two features into our future models. To that end, we took these two features into our future models. Below is a graph showing the relative importance of the features we made.



Model Building

Model Building We chose to build 4 different models using different machine learning techniques - XG Boost, Random Forest Classifier, Naive Bayes, and Logistic Regression. We used the sklearn train_test_split module to split our data into 80% for training and 20% for testing.

- **XG Boost:** We decided to use XGBoost's regression model to predict the odds that a team might win. XGBoost is a machine learning algorithm based on the boosting technique and is known for its highly efficient, yet powerful toolset. The boosting technique works by starting with basic decision trees and combining them such that the learning done by the first tree in the series impacts the way the model learns in the future - when a tree misclassifies a specific data point, the model increases the weights accordingly to improve future predictions.

We selected this model because our dataset includes over a thousand matches, each with several features; therefore, a lack of efficiency would mean significantly longer runtimes for both training and testing. Our feature selection told us that the best features to use would be 'ratings' and 'home/away'; which are average FIFA rating differences between the teams in the match, and home/away advantage, respectively.

- **Random Forest Classifier:** One of the benefits of the Random Forest algorithm is that it can not only be used for Feature selection but also model training. As mentioned above, it is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

  Therefore, once we identified our most important features, we thought it would be interesting to see how the random forest model performs in training a model with its own chosen metrics. In researching this, we inferred that this is a commonly used technique to avoid overfitting data.

  The model has 2 important hyperparameters that we had to determine: max depth and random state. Max depth: This controls the depth of individual trees. As a huge decision tree might lead to overfitting, we took the max depth as 10 Random State: This controls the seed of randomness on which the tree is generated. As we ran our code file multiple times, we wanted to ensure that the results we got from the model remained constant. Therefore, we introduced a seed integer (in our case, 0) to ensure constancy.

- **Naive Bayes:** Naive Bayes is a probabilistic algorithm that is based on the Bayes theorem. It works along with the "naive" assumption that features are conditionally independent given the class label. In our case, as most features are individually computed, we can make this assumption. This assumption makes the algorithm computationally efficient.

  We specifically chose to work with Gaussian Bayes - it works well with continuous features (Bernoulli and Multinomial Bayes assume features to be binary and discrete respectively). Gaussian Bayes works very well with real-valued data and measurements and all our features matched that criterion.

  The primary reason we used Naive Bayes is that our dataset was not very big. Furthermore, as soccer matches can have a lot of anomaly results, our data comes with a fair bit of noise. As the algorithm assumes that the features are independent, it can handle the noise appropriately as well. Lastly, the algorithm is insensitive to hyperparameter tuning. Therefore, we wanted to evaluate a case where we can avoid the programmer's bias of choosing the most fit hyperparameters without affecting the model performance.

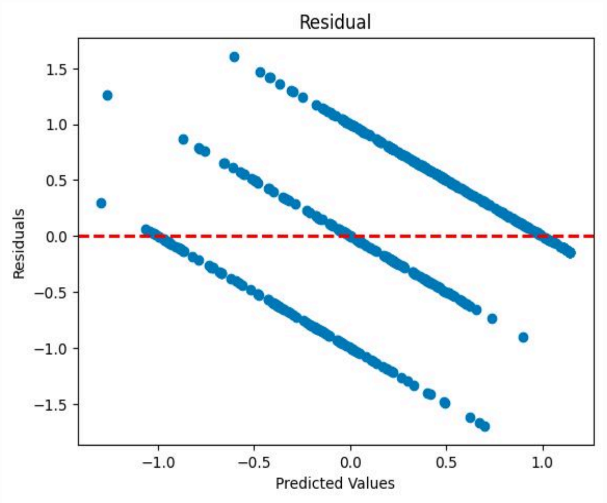
- **Logistic Regression:** This is a statistical method used for classification problems. In our case, classifying the result of a match. This algorithm builds a logistic function, which determines the probability of an instance belonging to a certain class. Based on the function, a decision line is used to assign the result to one of the positive or negative classes. This acts as a soft classifier and provides variety as compared to other models. It also allows us to better understand how boundary data points were created etc. This brought us to conduct regularization on the mode to tune those very boundary points.

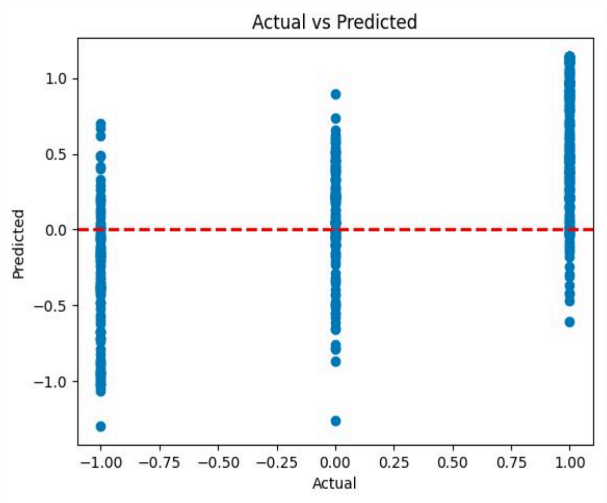
Results and Discussion:

Results: R^2 and Mean Squared Error from XGBoost

We ran the model 10 times and we got an average mean squared error of 0.387 and an R^2 score of 0.425.

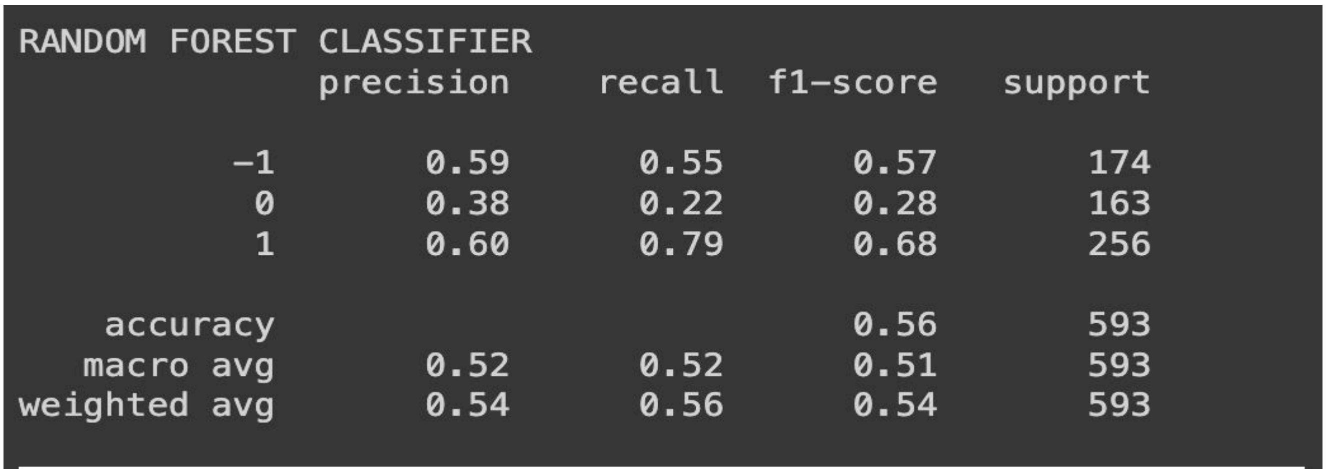
To visualize the results we plotted two different graphs: we first graphed our residuals against our predicted values to highlight the difference between our actual testing values and what the model outputted. We also plotted our actual values versus our predicted values so that we could see the relationship between the two. As evident, our model was largely inaccurate; especially considering that the ideal actual vs predicted graph would be clusters in the bottom left, middle, and the top left for each result respectively.



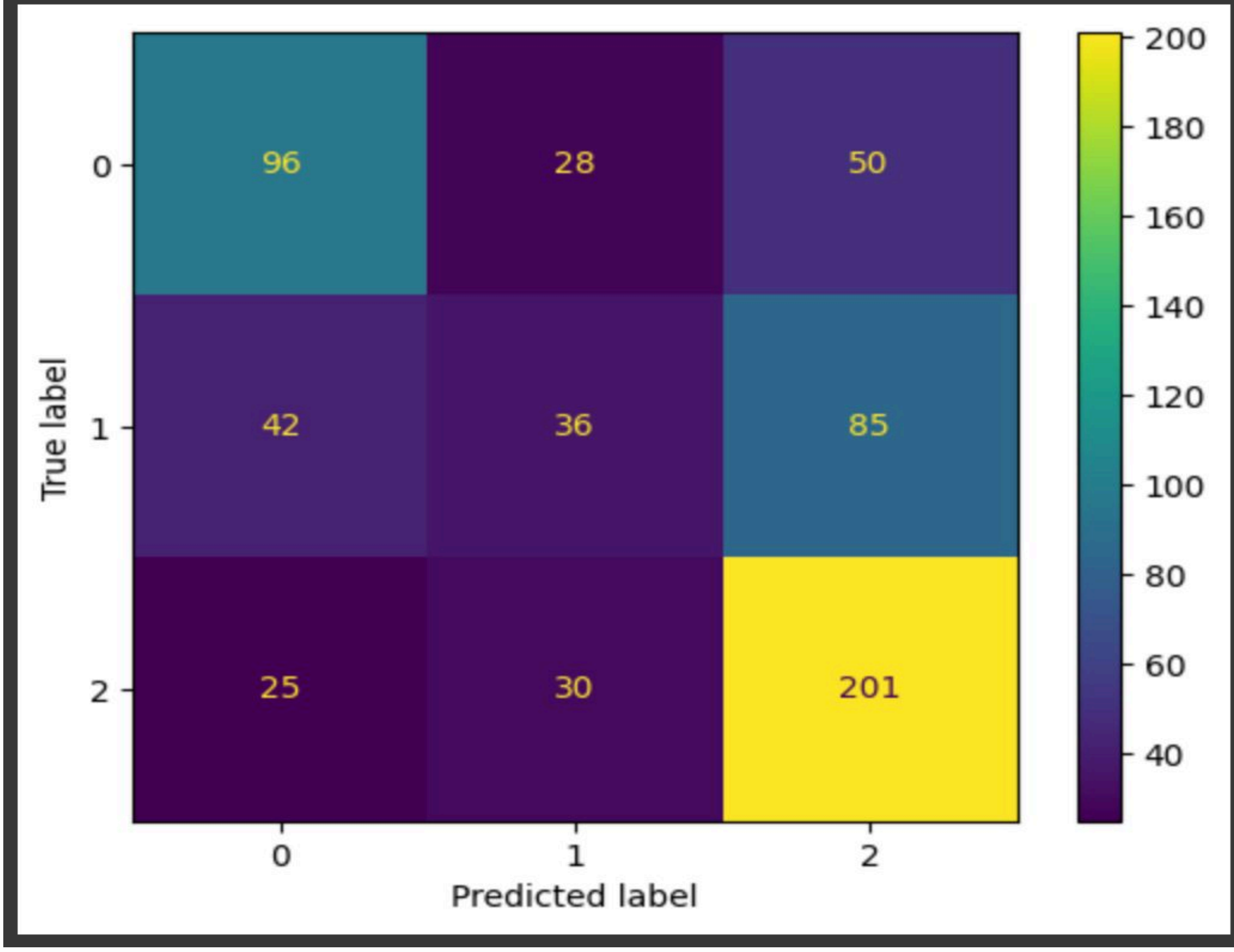


Results: Random Forest Classification

We ran a classification report and confusion matrix on the model to evaluate the model’s performance. The figure is shown below:



```
RANDOM FOREST CLASSIFIER
              precision    recall  f1-score   support

          -1       0.59      0.55      0.57       174
           0       0.38      0.22      0.28       163
           1       0.60      0.79      0.68       256

    accuracy                           0.56       593
   macro avg       0.52      0.52      0.51       593
weighted avg       0.54      0.56      0.54       593
```

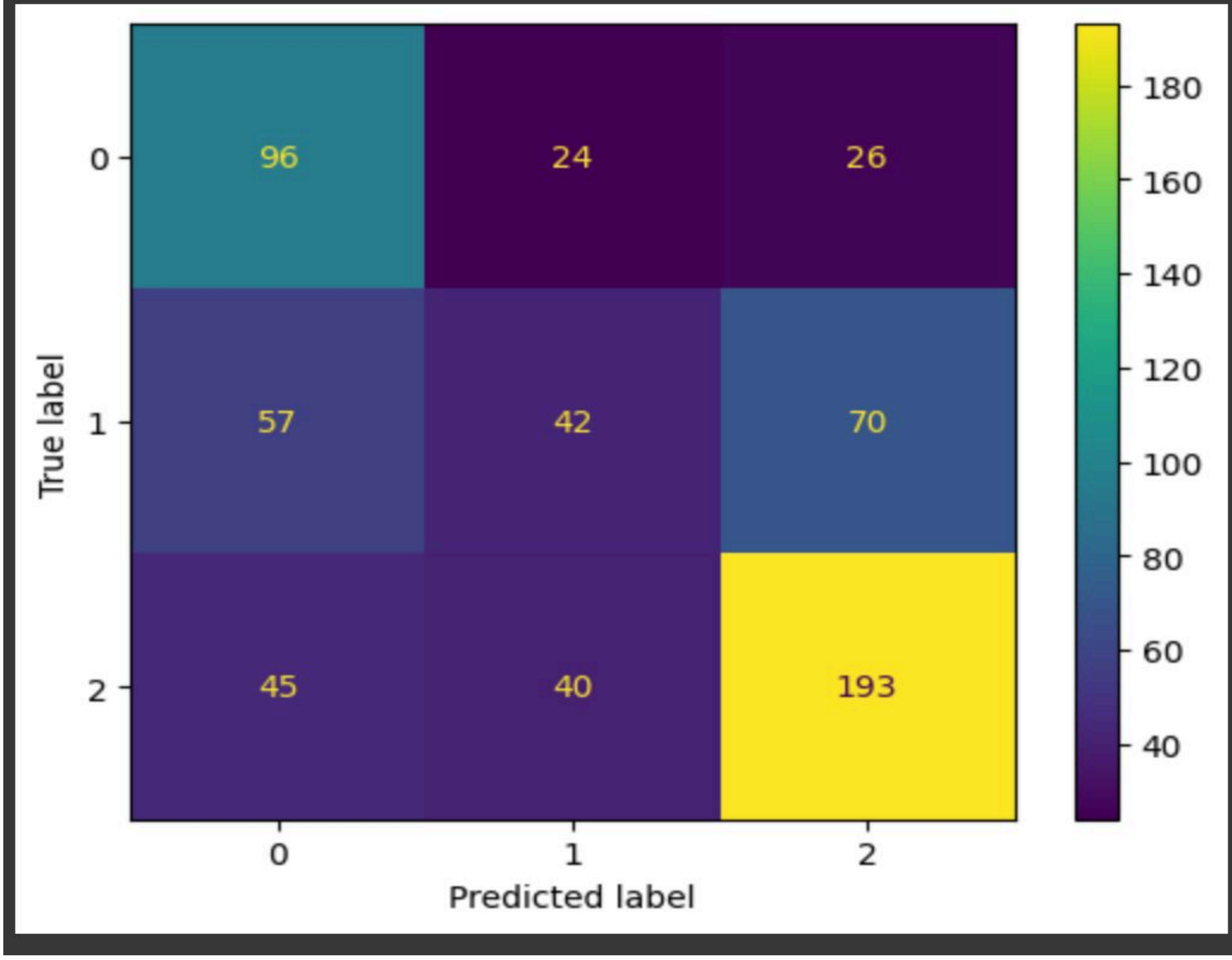


As seen above, the accuracy of the overall model was 56%. Furthermore, based on the recall, we notice that the predictions for losses and wins are more accurate (59% and 79% respectively) as compared to draws (22%). Furthermore, even the diagonal of the confusion matrix signifies this as the top left (losses predicted correctly) and bottom right (wins predicted correctly) have more correctly classified values than the middle (draws predicted correctly).

Results: Naive Bayes

We ran a classification report and confusion matrix on the model to evaluate the model's performance. The figure is shown below:

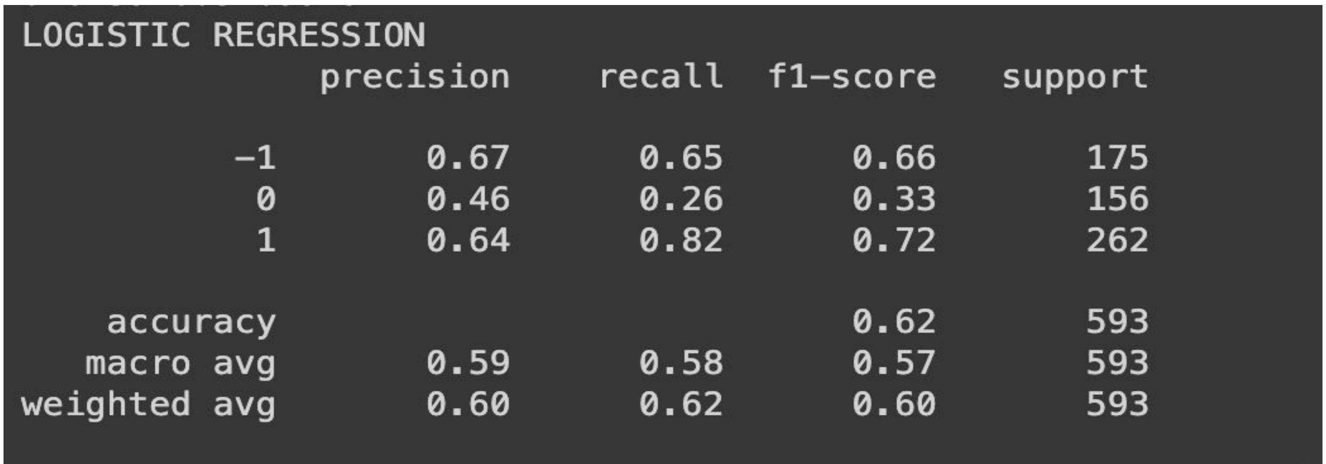
```
NAIVE BAYES
              precision    recall  f1-score   support

          -1       0.48      0.66      0.56       146
           0       0.40      0.25      0.31       169
           1       0.67      0.69      0.68       278

    accuracy                           0.56       593
   macro avg       0.52      0.53      0.51       593
weighted avg       0.55      0.56      0.54       593
```



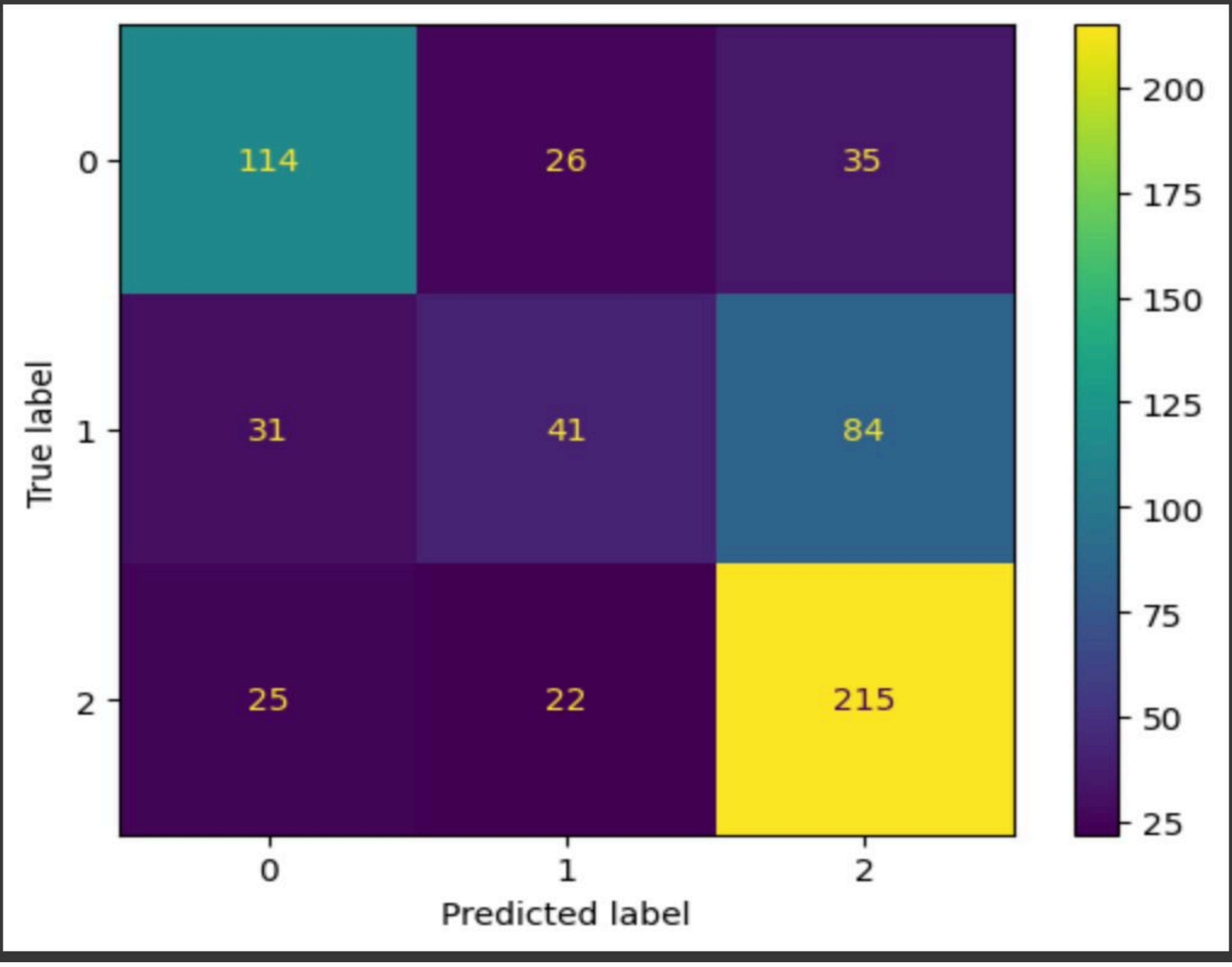
As seen above, the accuracy of the overall model was again 56%. Furthermore, based on the recall, we notice that the predictions for losses and wins are more accurate (66% and 69% respectively) as compared to draws (25%). Furthermore, even the diagonal of the confusion matrix signifies this as the top left (losses predicted correctly) and bottom right (wins predicted correctly) have more correctly classified values than the middle (draws predicted correctly).

Results: Logistic Regression

We ran a classification report and confusion matrix on the model to evaluate the model's performance. The figure is shown below:



```
LOGISTIC REGRESSION
              precision    recall  f1-score   support

          -1       0.67      0.65      0.66       175
           0       0.46      0.26      0.33       156
           1       0.64      0.82      0.72       262

    accuracy                           0.62       593
   macro avg       0.59      0.58      0.57       593
weighted avg       0.60      0.62      0.60       593
```



As seen above, the accuracy of the overall model was again 62%. Furthermore, based on the recall, we notice that the predictions for losses and wins are more accurate (65% and 82% respectively) as compared to draws (26%). Furthermore, even the diagonal of the confusion matrix signifies this as the top left (losses predicted correctly) and bottom right (wins predicted correctly) have more correctly classified values than the middle (draws predicted correctly).

Model Evaluations: Classification Models:

For our remaining 3 classification models, we got similar results:

- Our overall accuracy was around 55% to 65%, which is moderate. This is taking into account, that in an ideal randomized case, the model accuracy would be around 33%
- The correctly predicted proportion of wins and losses (around 65% to 80%) was much higher than that of draws (around 25%)

Here are some reasons why these models performed better than the XG Boost model:

- As mentioned above, these models were classification based while XG Boost was regression-based. As our data was discrete and we wanted to classify the match result as either a win, loss, or draw, these models worked better.
- Gaussian Naive Bayes works very well with real-valued data and measurements and all our features matched that criterion.
- For Logistic Regression, we could avoid overfitting by adding regularization terms
- The Random Forest Classifier was an extension of the same algorithm being used for feature selection. Therefore, in terms of model compatibility, a random forest would work the best. Furthermore, we experimented with different tree depths to ensure that the tree is extensive enough but does not overfit the data.

Here are common shortcomings in the model and potential solutions to the problems:

- Less Data Points: We only had data on Premier League matches from 2009 to 2017. Additionally, we had to split this data into testing and training datasets, which further reduced our training data. Therefore, we were only left with around 3,000 data points. As match results are specific to 2 teams matching up, we did not have a sufficient number of matches played between those 2 teams

- Low Draw Accuracy: As highlighted above, we inferred that our model accuracy for predicting draws was low. One potential reason for this result is the random nature of draws. Let’s consider 2 cases. First, a really good team playing a bad one. The model has sufficient data points to evaluate a positive scoreline in favor of a good team. In most cases, the predicted win and loss of the team will be correct. However, second, consider two good or two bad teams matching up. In reality, this will be a close match-up, which may lead to a draw. However, the model needs the threshold of equality between 2 teams to be very high - almost discrete where the goal difference should be 0 - to qualify it as a draw.

  A potential solution is understanding the boundaries of a draw and tuning the models such that they predict draws more accurately. In our case, this would primarily involve expanding the quantitative threshold for a draw so that more “close wins” or “close losses” are predicted as draws.

References:

- Reference 1
- Reference 2
- Reference 3

- Reference 4

Proposed Timeline:

Gantt Chart

Contribution Table:

- **Krishi Manek**:
  - Created entire slide deck for video
  - Updated GitHub pages
  - Helped edit final project report
  - Helped brainstorm feature selection and model building
- **Vasav Jain**:
  - Wrote the Final Report
  - Wrote the code for features development
  - Worked on the final video for the project
  - Handled data preprocessing and cleaning
- **Samyak Jain**:
  - Feature Engineering - Built some features for the model
  - Data Training - Built training and testing datasets
  - Model Building - Built XGBoost Model and Random Forest Classifier
- **Shreiyas Saraf**:
  - Built video for Project Proposal
  - Feature Engineering - Built some features for the model
  - Feature Selection - Built Random Forest Model Selection
  - Model Building - Built Gaussian Naive Bayes and Logistic Regression Models
  - Built Confusion Matrix Visualisations
- **Aakash Prasad**:
  - Pulled dataset off Kaggle and reformatted into CSVs
  - Built Team Impact/Player Impact feature
  - Built visualizations for dataset - correlation scatter plots for each pair of features
  - Built the framework for the XG Boost model
  - Created some of the visualizations for results/discussion

ML 4641 Project (Group 66)

Aakash Prasad, Krishi Manek,
Shreiyas Saraf, Samyak Jain,
Vasav Jain