# SOCCER MATCH OUTCOME PREDICTION & BETTING FOR MAXIMUM PROFIT

**Group 22**
Merlin Beaufils (mb4988)
Tempe Gu (yg2831)
Karpagam Murugappan (km3702)
Sitong Qian (sq2239)

# 1 Introduction

With the recent celebration of the World Cup and our passion for soccer, we wanted to predict soccer matches in order to maximize our profits in betting. It's known that predicting soccer matches is extremely hard due to the vast uncertainties associated with it. When we learned from an article [1] that incorporating bidding information improved prediction accuracy, we decided to implement a machine learning model with the historical match data as well as the corresponding bidding data. In this project, we aimed to use different classification machine learning techniques to predict real-life soccer matches' outcomes and develop a systematic betting strategy to help us bet our money wisely.

# 2 Data Collection

## 2.1 Data scraping

To ensure the authenticity of the dataset, we decided to scrape the raw data from reliable websites and make the datasets ourselves by using **BeautifulSoup** and **Selenium** from python packages. Our dataset contained all the available regular season games from 2017 to 2022 for the major five leagues in Europe (Bundesliga, La Liga, Ligue-1, Premier League, Serie A) from these two websites:

- **Odds Portal** [2]**:** We scraped these following numeric features:
    - ***1***: Average/biggest League betting odds offered on the home team to win
    - ***X***: Average/biggest League betting odds offered on the home team to draw
    - ***2***: Average/biggest League betting odds offered on the away team to win
- **FBref** [3]**:** We scraped these following numeric features:
    - ***xG***: Expected goals for both the home team (*xG*) and away team (*xG.1*)
    - ***Score***: Final scores of each match

We also scraped other descriptive data of matches like *Home*/*Away* team name, *Date*, *Time*, *Attendance*, *Venue*, *Referee*, etc. Please refer to *1.a. Data_scraping.ipynb* for data collected, *new_odds.csv* for Odds Portal data, and *total.csv* for FBref data.

## 2.2 Data Cleaning

After collecting the data, we 1) crosschecked two tables we obtained from different sources, 2) dropped missing values, 3) joined two tables, and 4) converted betting odds to probability, as the betting odds were in an non-continuous American Odds format. We followed these two math formulas to convert the betting odds to probability [4]. Please refer to *1.b. Data_cleaning.ipynb* for the code and *combined_checkpoint.csv* for the data after cleaning.

- **For Negative American Odds:** Implied Probability = (-1*(Odds)) / (-1(Odds) + 100)
- **For Positive American Odds:** Implied Probability = 100 / (Odds + 100)

# 3 Insights From Data Exploration

The initial data exploration was very limited, as the data needed further preprocessing before data exploration. We briefly checked the columns' data types, missing values, basic statistics (i.e., mean, standard deviation), and categories of the categorical variables.

We then performed the following preprocessing steps: 1) sorted the data by date and reset the index; 2) extracted home team's score (*home_score*) and away team's score (*away_score*) from *Score* 3) added *season*: each game season lasts from August to May next year (e.g., season 1 refers to the period of 2017 August to 2018 May); 4) created **target variable** - *home_win*. If the home team wins, the value is 1. Else, the value is 0. Our project focused on predicting *home_win*. We observed that the two classes were roughly balanced.

We also did feature engineering by creating two main functions. The first function created **new features containing metrics for the last N games**. The function takes a row of data (i.e. a specific game) and calculates the home team's or away team's performance metrics for the last N games (prior to this game). The types of metrics (use_case) include win%, average score, and xG difference. The function has an option to give more weights to recent games. We created the following 18 features for both home team and the away team (totalling 36 features): the last 5/8/12 games with equal weights and with exponential weights for win%, average score, and xG difference. We also created another 18 features that calculate the difference between the home team and away team for each of the above 18 metrics.

The second function created **new features containing cumulative metrics** for the current season or for all data collected. The function takes a row of data (i.e. a specific game) and calculates the home team's or away team's performance metrics (prior to this game). The function asks a user whether the metric is for the current season or for all seasons. The types of metrics (use_case) include win%, average score, and xG difference. We created the following 6 features for both the home team and the away team (totalling 12 features): cumulative for the current season and for all seasons with equal weights for win%, average score, and xG difference. We also created another 6 features that calculate the difference between the home team and away team for each of the above 6 metrics.

For data exploration, we first checked the features' **basic statistics** (i.e., mean and standard deviation) for each class. We also ran the **correlation matrix** for all features (existing and new). We observed that: 1) *1_PROB* has the strongest correlation with *home_win*; 2) cumulative metrics have stronger correlation with *home_win*; 3) for the last N games metrics, the larger the N, the higher correlation the metric has with *home_win*. After selecting the features, we checked the final correlation matrix (Appendix A). Some features were still highly correlated. To address this, we would use **PCA** in the machine learning process.

We also created **histograms** for numerical features to check if any features help to distinguish the classes. We found that *1_PROB* and *diff_cumulative_all_avg_xG_diff* were potentially strong. We also created **pairplots** to check correlations between features. Please refer to Appendix B for visualizations.

We checked the **null values** in the columns that would be used for machine learning, and we decided to drop all the null values as they did not provide much insights about the team's performance. Please refer to *2. Data_preprocessing_exploration.ipynb* for the code and *joined.csv* for the dataset used in machine learning.

# 4 Machine Learning & Profit Maximization

Since our data is time series, we used **structured splitting** for the train-validation-test split. Season 1 to 3 were used as the training set; Season 4 was used as the validation set; Season 5 was used as the test set.

## 4.1 Prediction

We built a pipeline to train the following models: Logistic Regression, Random Forest, AdaBoost, Gradient Boosting, Decision Tree and SVC. For each model, we used **GridSearchCV** to choose the best set of parameters and **PCA** due to multicollinearity in the features. After the models were trained, we collected the best set of parameters and the best score for each model and cross-checked **feature importance** of top performing models. Please refer to Appendix C for samples of feature importance plots. For the top performing models, *1_PROB* was the strongest feature, followed by *season*.

Additionally, we built a few Neural Network models. We generated the **training vs validation loss plot and accuracy plots** (Appendix D) to understand if our model was overfitting.

To compare the models' performance, we calculated the **test set's accuracy, precision, recall and F1 score** on 69 models (Appendix E) and plotted their **precision-recall curves** (Appendix F). As our ultimate goal is to maximize the bidding profits, the predicted probability is important to us.
We tried **calibration** to further tune our models, but it did not improve the accuracy of predicted probabilities.

Our best model for prediction is "AdaBoostClassifier Last 8 games" with an accuracy of 68% precision of 0.65, recall of 0.54, and F1 score of 0.59. Due to the high degree of randomness in soccer games, we used 0.5 accuracy as our benchmark. Compared with the benchmark accuracy, our model performance increased by 40%. Please refer to *AML_Project_Machine_Learning_Models_final_copy.ipynb* for detailed code for developing and evaluating the machine learning models.

### 4.2 Profit Maximization

After obtaining a strong model for predicting match outcomes, we focused on betting. Our goal was not only to predict match outcomes, but to make a profit by betting on these games. For this, we needed a new performance metric relating to profit.

We thus defined ROI, the percentage of profit over the quantity invested. A simple betting strategy would be to bet an investment (I) on home win or home loss as predicted and sum the profits based on the multipliers given by bookmakers. We immediately saw that this was a losing strategy. This was because of the margins that bookmakers keep creating a negative profit expectation. To address this we only bet on games where our prediction was higher than the probability proposed by the bookmaker. This allowed for a positive expectation and showed profits in the long run. We then went further to define a threshold of confidence required to place a bet.

After testing our models, we saw that we were able to get a positive return on investment even without tweaking the models. Not satisfied yet, we wanted to go further and aim to maximize profit as opposed to just prediction accuracy. Looking back at feature engineering and exploration, we had decided to drop team ids and use each team's past game metrics to encode their relative strength (cummulative_win%, lastn_xg_diff, etc). In the aim of profit maximization, we decided to add a new feature, false_pred. This represented the difference between the game result (home_win) and the bookmaker prediction (1_prob). Once again, we created features using the cumulative value of this variable in each team's past games.

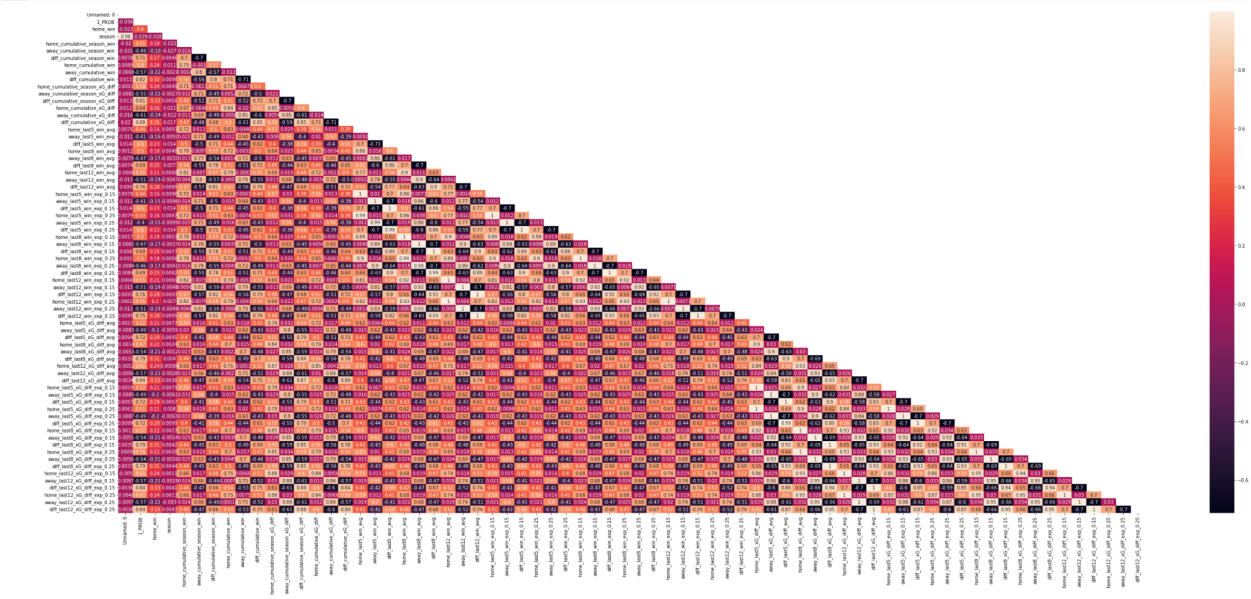
Finally, we take a closer look at the loss functions of our models. We notice that the loss function for Logistic Regression is not in line with the betting strategy outlined above. As a final attempt, we try to improve this loss function by weighing the loss of a sample according to the profitability of that bet. This looks like: L(pred, label, prob) = label * ln(pred) * prob + (1-label) * ln(1-pred) * (1-prob). This would hopefully successfully incorporate betting information directly into the model.

These strategies allowed us to improve our Return On Investment (ROI) all the way to 2.51%. Please refer to the notebook *Custom_loss_with_pytorch_lightning.ipynb* for this implementation and results. This was done in Pytorch Lightning. We plotted the profit over time for the final model(Appendix G).

## 5 Potential Improvements

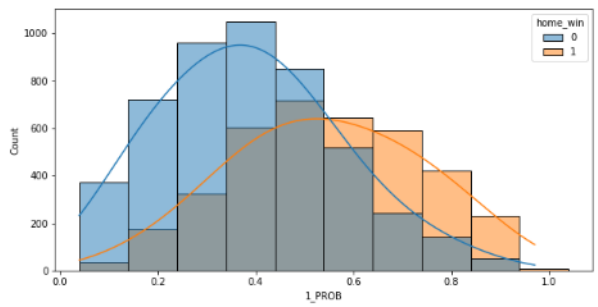
To further improve our results, we would need a lot more data as we currently only have ~8000 data-points. Paid subscriptions to larger databases exist which we could utilize. Finally, we could dig deeper into customizing models for the sake of profitability and use much deeper models.

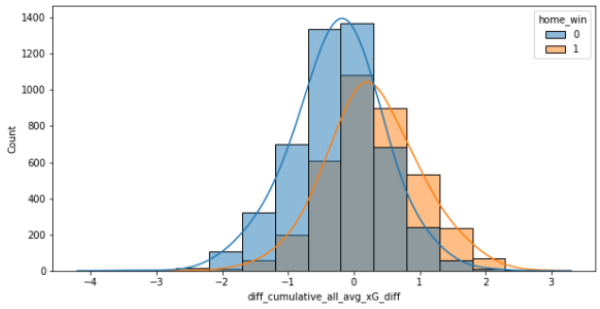
# Appendix A: Correlation Matrix



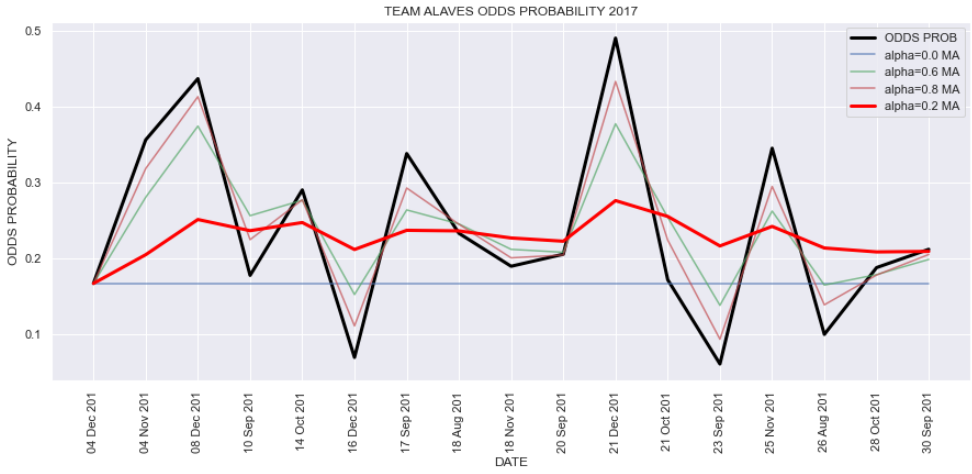
# Appendix B: Data Visualization

## B.1 Samples of Histograms

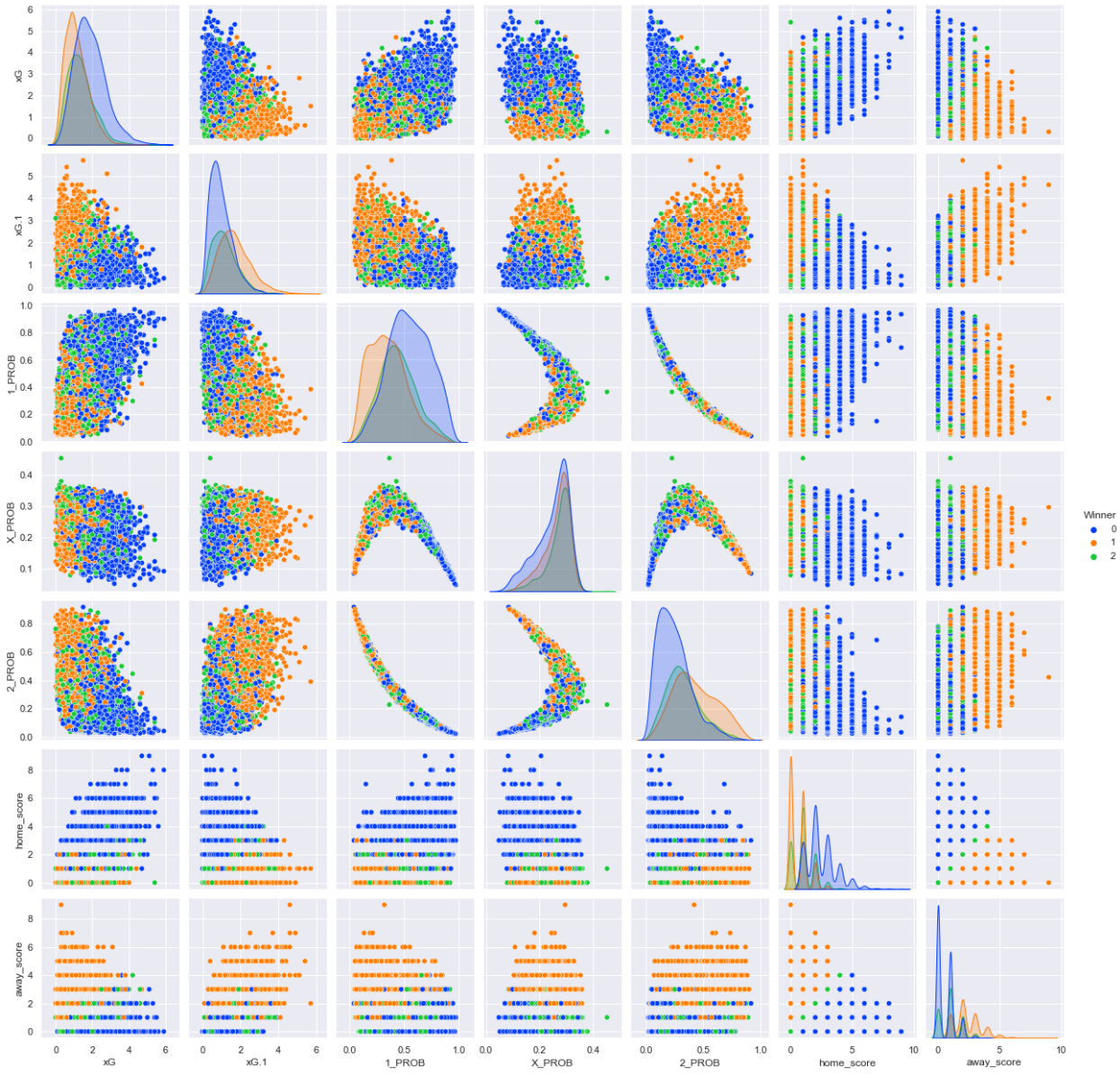



## B.2 Alaves Odds Probability in 2017

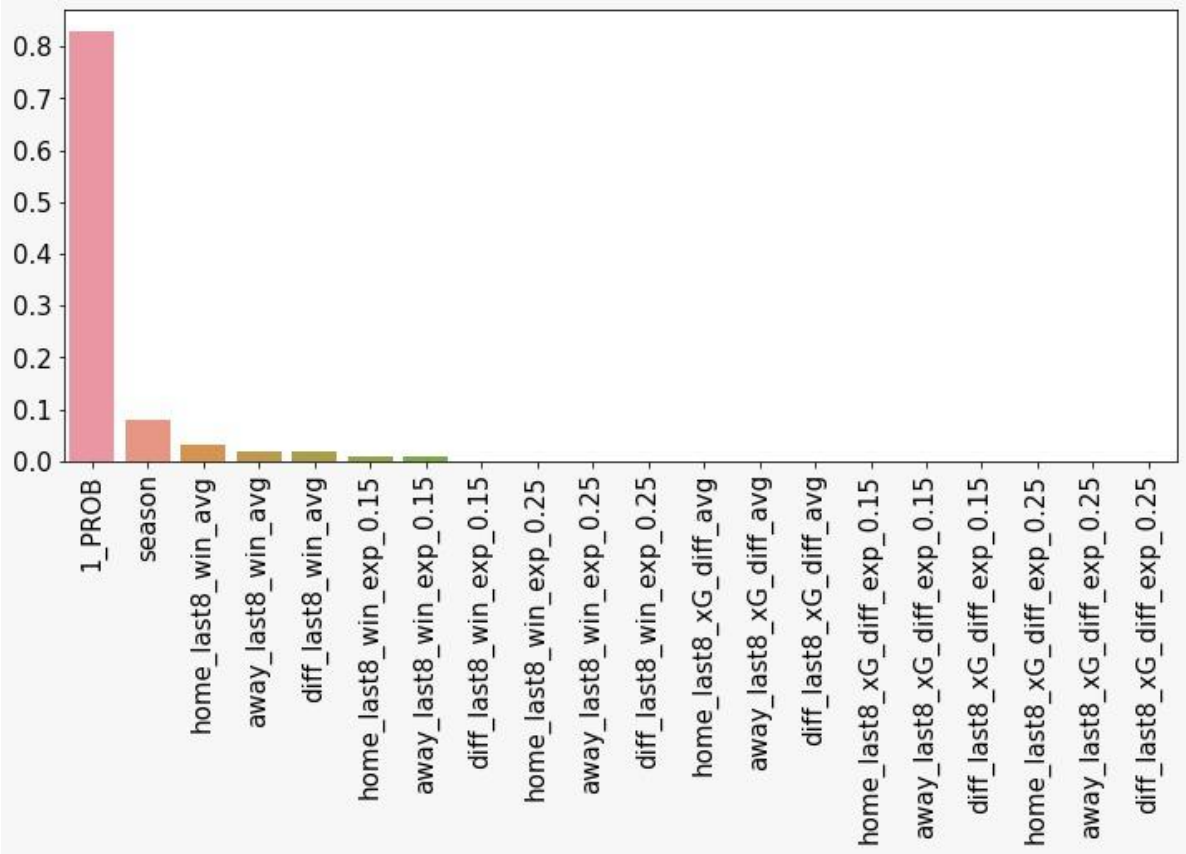


## B.3 Initial data exploration - Pairplots

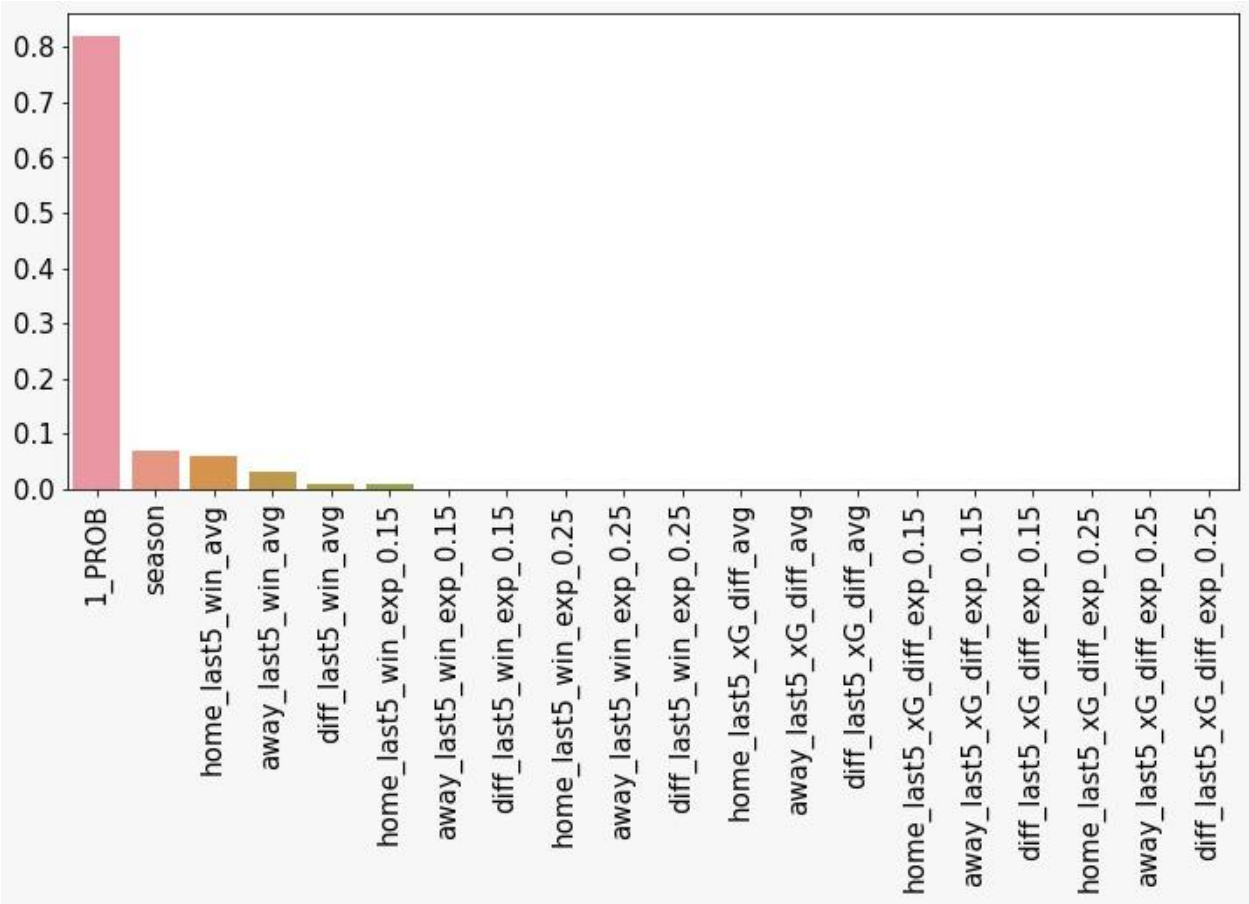


# Appendix C: Feature Importance Plots for the Top 5 Models

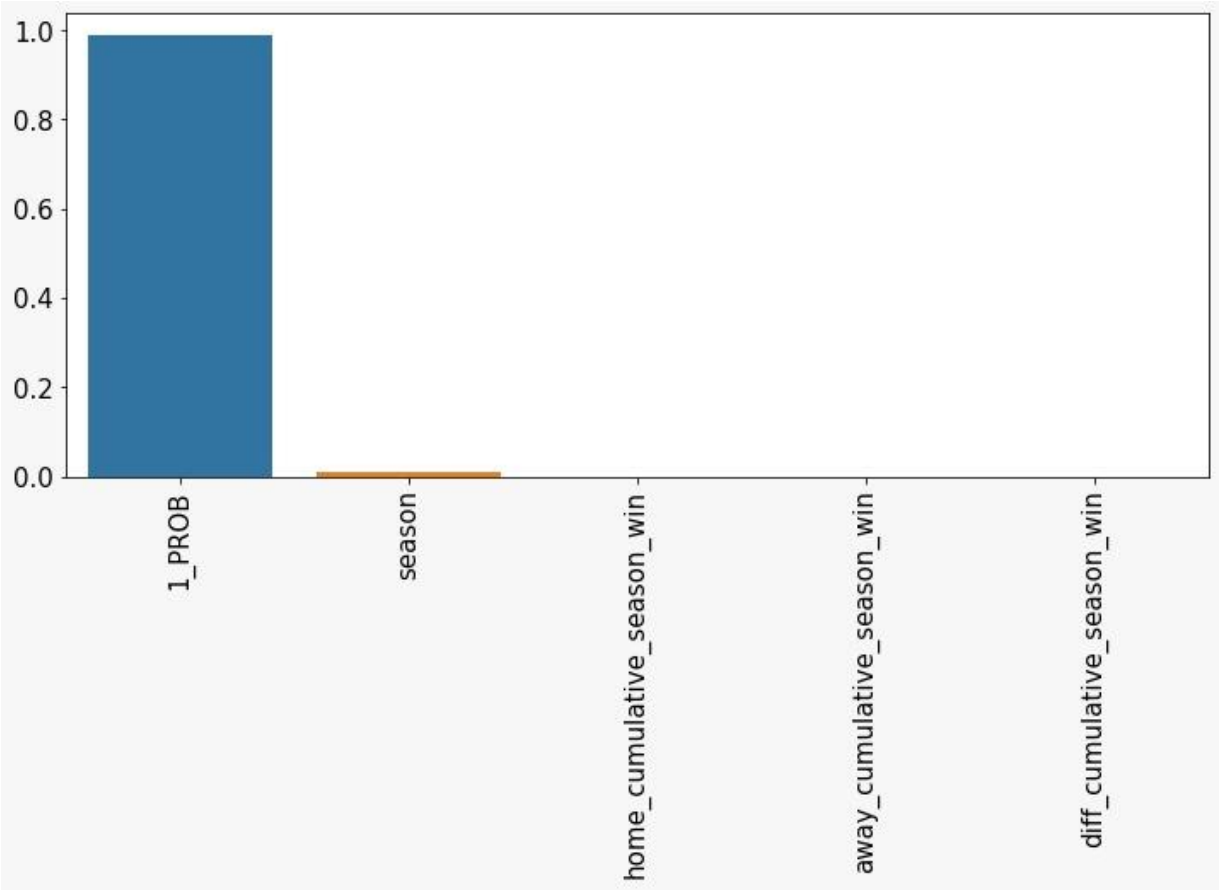
## C.1 Feature Importance - AdaBoostClassifier Last 8 Games



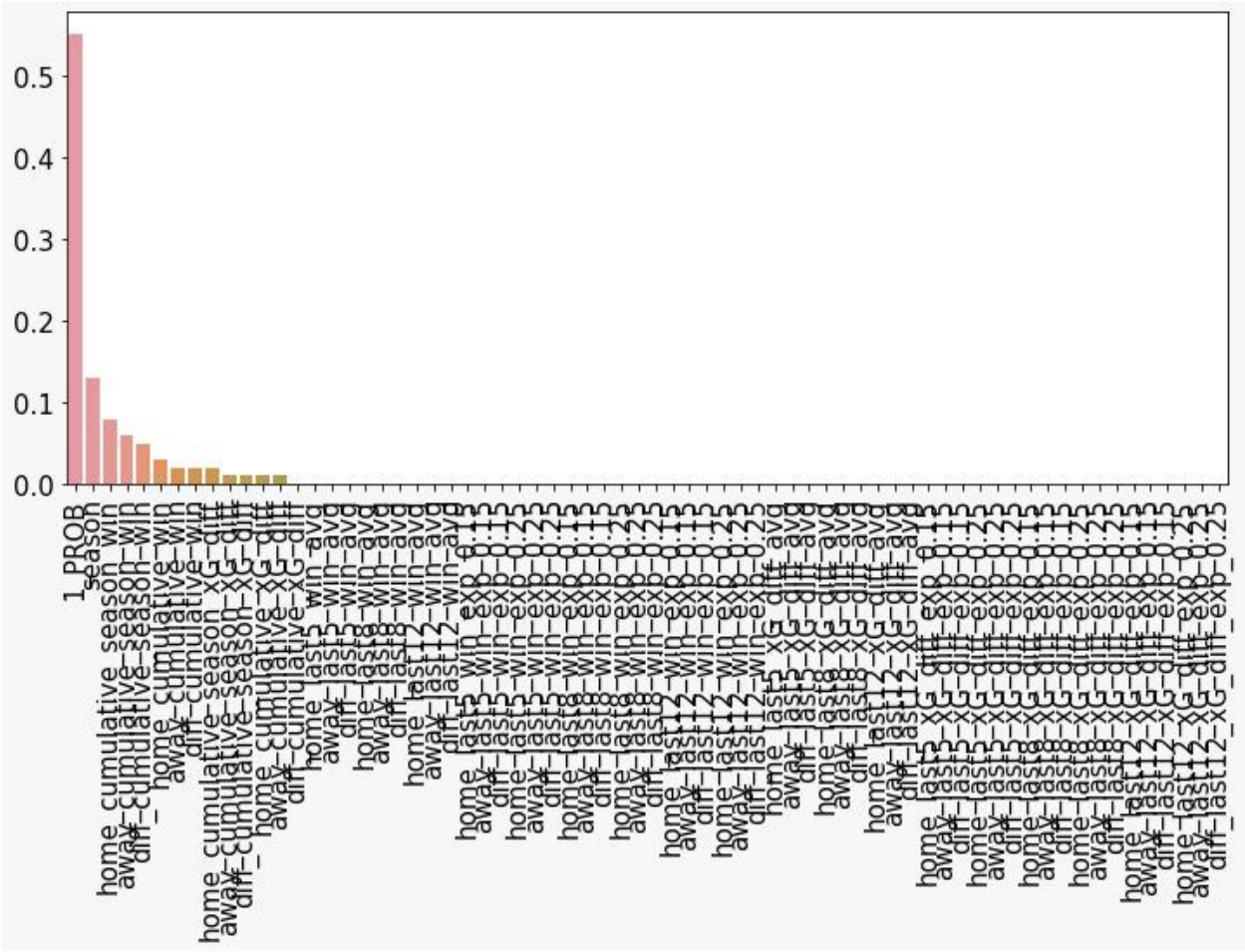
## C.2. Feature Importance - AdaBoostClassifier Last 5 Games



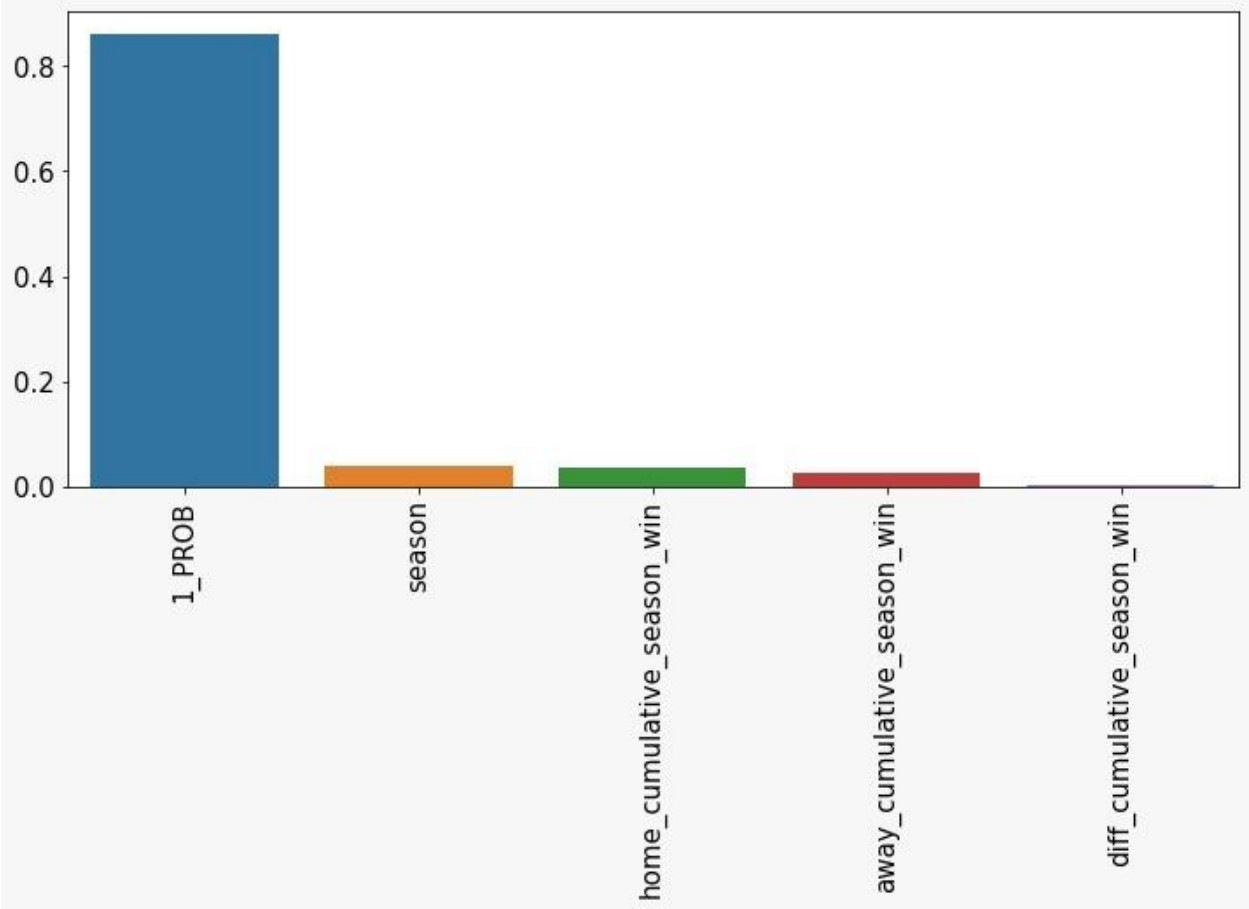
**C.3 Feature Importance - AdaBoostClassifier Top 5 Features**



## C.4 Feature Importance - AdaBoostClassifier

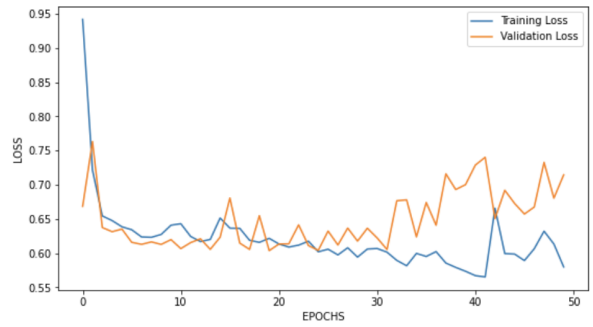


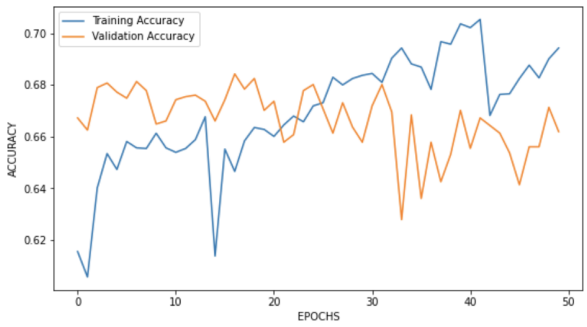
### C.5 Feature Importance - Logistic Regression with PCA Top 5 Features




## Appendix D: Sample of Training vs Validation Plots for Neural Network models

MODEL NN_1



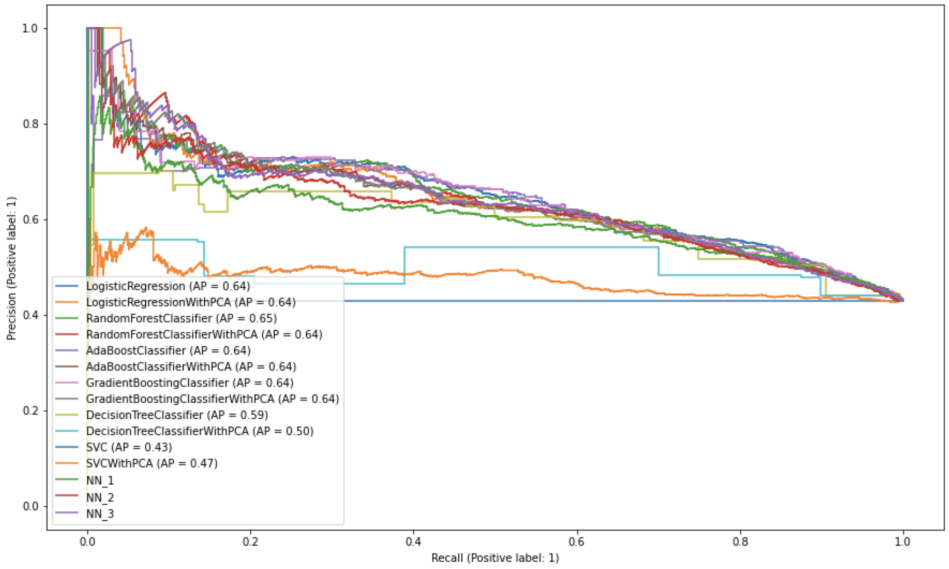



# Appendix E: Top 5 Models Results

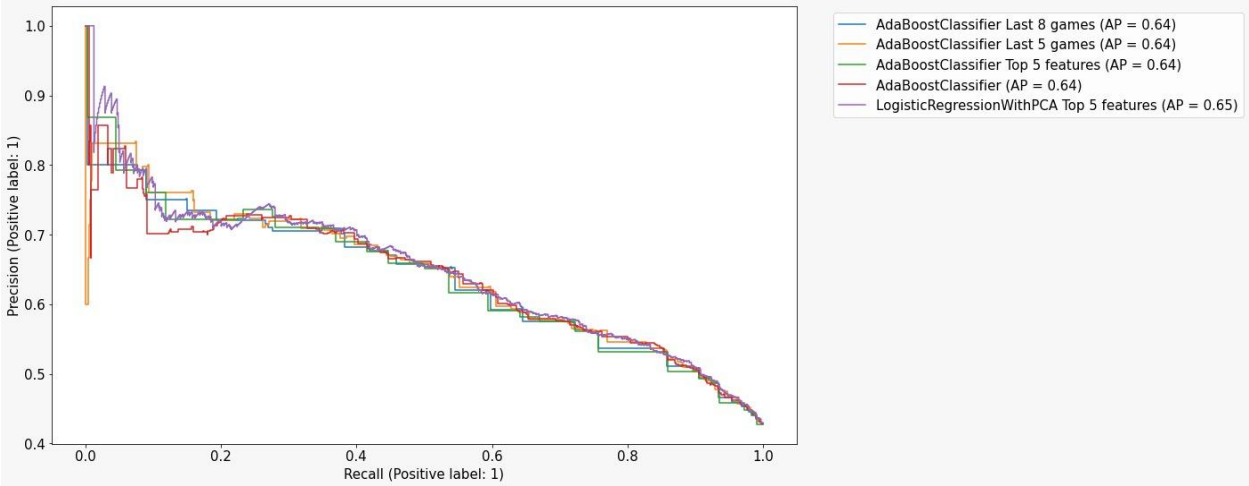
| ModelName | BestParam | Acc_percentage | Profit_Loss_percentage | TestPrecision | TestRecall | TestF1 |
|---|---|---|---|---|---|---|
| AdaBoostClassifier Last 8 games | {'ada__learning_rate': 0.05, 'ada__n_estimators': 100} | 68.0 | 5.9 | 0.651815 | 0.541838 | 0.591760 |
| AdaBoostClassifier Last 5 games | {'ada__learning_rate': 0.05, 'ada__n_estimators': 100} | 67.9 | 5.7 | 0.651667 | 0.536351 | 0.588412 |
| AdaBoostClassifier Top 5 features | {'ada__learning_rate': 0.05, 'ada__n_estimators': 100} | 67.8 | 5.6 | 0.651085 | 0.534979 | 0.587349 |
| AdaBoostClassifier | {'ada__learning_rate': 0.05, 'ada__n_estimators': 100} | 67.8 | 5.6 | 0.653650 | 0.528121 | 0.584219 |
| LogisticRegressionWithPCA Top 5 features | {'logr__C': 10, 'logr__penalty': 'l2', 'pca__n_components': 4} | 67.6 | 5.2 | 0.653979 | 0.518519 | 0.578424 |

# Appendix F: Precision-Recall Curves

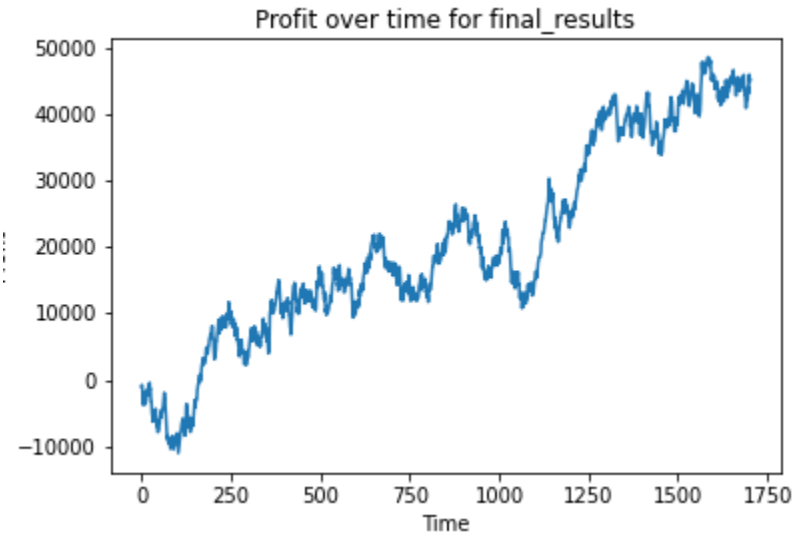
## F.1 All Models



## F.2 Top 5 Models




# Appendix G: Profit-over time



## References

[1] Caldas, A. (2021, June 30). *Beating soccer odds using Machine Learning - project walkthrough*. Medium. Retrieved December 1, 2022, from https://medium.com/analytics-vidhya/beating-soccer-odds-using-machine-learning-project-walkthrough-a1c3445b285a

[2] s.r.o., L. S. (n.d.). *Soccer betting odds comparison*. Oddsportal.com. Retrieved December 1, 2022, from https://www.oddsportal.com/soccer

[3] *Football statistics and history*. FBref.com. (n.d.). Retrieved December 1, 2022, from https://fbref.com/en/

[4] Fisk, W. B. G., & About the Author Geoff Fisk G. (2022, November 30). *Implied probability calculator for sports betting odds*. GamingToday.com. Retrieved December 1, 2022, from https://www.gamingtoday.com/tools/implied-probability/