Introduction:

In this report, we are interested in finding the factors that give a team a better chance of winning. To achieve this, we have adopted the following strategy:

First, we argue that bets can be good indicators for prediction. Then, we will use predictions made from betting data as a baseline. In this way, we have benchmarked the average evaluation made by the public. We have also created a betting strategy based on the prediction of the model, which has achieved significant returns.

Then we fit the same model using the dataset that adds in the team talent and team strategy data respectively. If these two models can break even or even surpass the benchmark, it means that the additional information of team talent and team strategy is meaningful. Then we can reverse engineer, and see which features in talent/strategy have the highest feature importance. After that, we can claim that if we improve on these features, we can perform better. We will fit the team talent and team strategy models separately because their variables may have large correlations, which will eschew the result.

Data cleaning:

1. Matches:
    a. There are over 1000/25979 NaN values for each player_id, and over half NaN in some betting systems.
    b. **I drop the column of player_id because if we join the player information, there might be hundreds of features, which makes it hard for feature extraction and learning. (But if you need it, i will only delete the row where player is missing - now it is the match_clean4.csv**)
    c. I count the net goal for the home team and away team before their match happens for each season and each league.
    d. I count the average betting from different systems to avoid missing value in a single betting system.
    e. The final scheme of **match_clean2.csv** has column: country_id,league_id,season,stage,date,match_id,home_team_id,away_team_id,home_team_goal,away_team_goal,home_net_score,away_net_score,avg_h_bet,avg_d_bet,avg_a_bet
2. Team attributes
    a. There are 969/1458 NaN values for buildUpPlayDribbling. I dropped this column with over 60% of NaN values.
    b. The team attributes data contains team play styles data for different seasons. I matched the team attributes with the outcome (as of points per game (under 3/1/0 system in Europe), winning percentage, and net goals per game). ###average
    c. Since each team attribute has a value out of 100 and a class that describes the degree of it, I dropped the non-numerical “class” values to avoid correlation between features and further simplified the model.
    d. The final scheme of **Team_Attributes.csv** has features:

buildUpPlaySpeed, buildUpPlayPassing, chanceCreationPassing,
chanceCreationCrossing, chanceCreationShooting, defencePressure, defenceAggression,
defenceTeamWidth
with 3 target columns:
points, win, net

3. Player attributes
    a. Dealing with Players with Multiple Entries

It will be optimal if we factor in each player's condition at different time periods. However, we may hypothesize that an individual player's attributes do not vary across different timeframes. Therefore, We shall only keep the most recent record for each player.

    b. Missing data of Numerical Variables

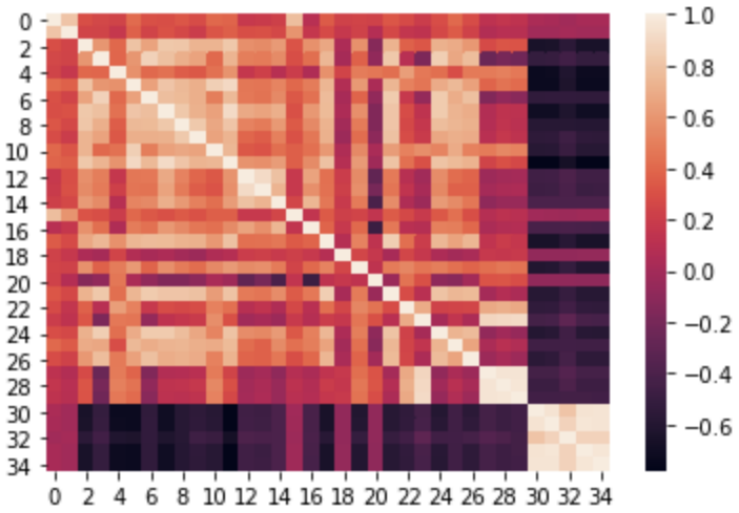
The missing data are clearly not missing completely at random (MCAR), since the missing data is clearly systematically lost. The same rows are missing for different features. In this case, it is not safe to drop the rows with missing data since by doing so we are amplifying bias.

However, it is also not appropriate to perform multiple imputation since there is a large chance that values are not missing at random (MAR). Therefore we will impute the mean for numerical features.

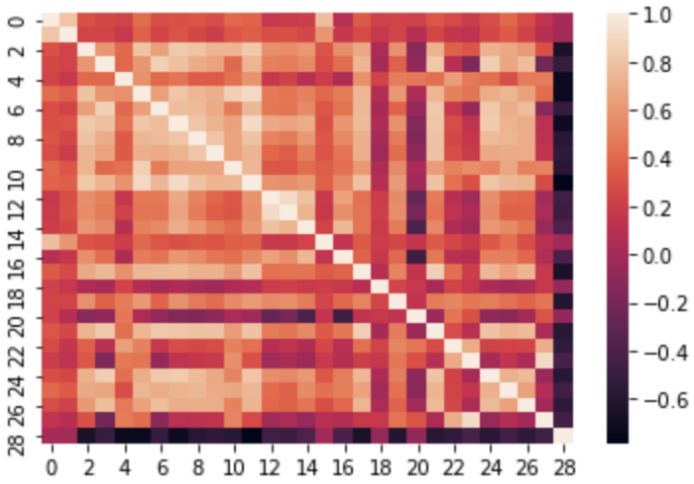
    c. Categorical Variables

We have excluded categorical variables for simplicity.

    d. Feature Correlation and Feature Engineering



Here plotted is a heat map of the feature correlation matrix for player attributes. We can see that features 30-34 (the gatekeeper features) are highly correlated with each other and negatively correlated with other features. Also, features 27-29 (marking, standing_tackle, sliding_tackle) are also highly correlated. Since we want to keep the interpretability of features, we won't perform PCA or other dimensionality reduction techniques since they will affect other features. Therefore we will take a simple average of these features to avoid collinearity during feature selection.



This is the heatmap after engineering.

e. Feature Selection

After we have engineered the features, we will perform feature selection to further reduce dimensionality of the dataset. If we trust that the "overall_rating" feature is representative of the true ability of the player, we can use it as a target and use Lasso regression to select features. The superiority of this approach over PCA or other dimensionality reduction techniques, again, is that we have keeped the interpretability of features. Since potential is highly correlated with overall_rating and has similar underlying meanings, we will drop it. The remaining features and their weights are:
{'short_passing': 0.021, 'long_passing': 0.040, 'reactions': 0.651, 'strength': 0.0333}

Methods:

Baselines:

For Classification: (Models under betting and player attributes)

1. GBTree
2. RF
3. Naive Bayes
4. Logistic Regression

For Regression Models: (Three models under team attributes)

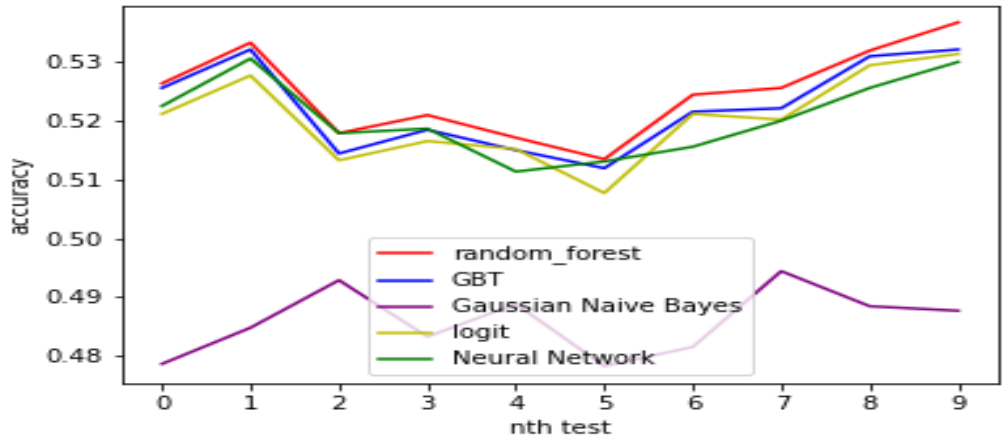
1. GBTree
2. RF
3. Linear Regression

Training Scheme:

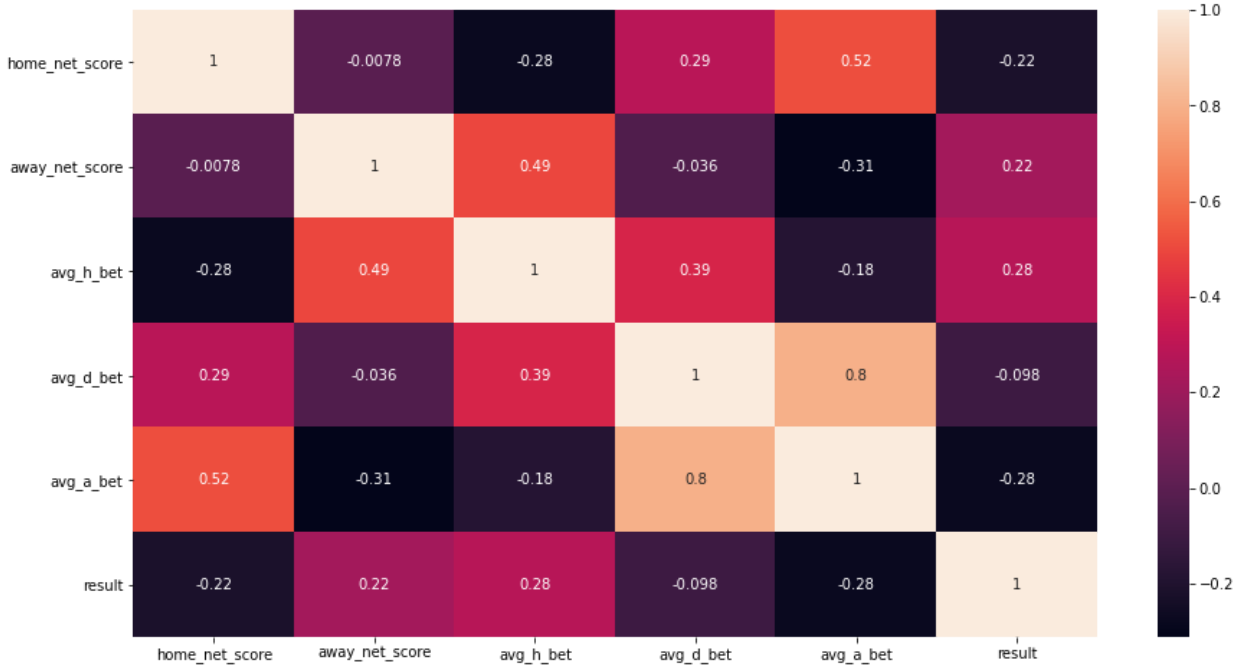
1. *Using bets to make predictions (modeling_bets.ipynb)*

Task 1: Using bets before the match to make predictions

We first use the bets and net goal before matches for home and away teams to predict whether the result is home_win, home_draw, away_win. This has served as our baseline model for football outcome predictions. We compare different ML models like random forest, Gradient Boost, and etc. The average accuracy has an average of 52.5% by Random Forest, which is our best predictor. The comparison from different methods are:



The correlation map on features:



We could notice that the feature of home_net_score, away_net_score, avg_h__bet, and avg_a_bet has large correlation with the result (indicated by 0, 1, 2 for home win, draw, and away win)

Task 2: Possible Betting Strategies to Explore

The betting information can also be used to develop successful betting strategies. Although this is not our main goal here, it is interesting to explore a little. We propose that if you invest k dollars for each match in a group of matches betting that you want to attend, the returns will be estimated as the equation below:

$$money\ return\ =\ k\ *\ [(accuracy\ *\ (avg\ bets\ -\ 1)\ -\ (1\ -\ accuracy))]\ =\ k\ *\ (accuracy\ *\ avg$$

Where accuracy is the prediction accuracy in the selected group in percentage, avg bets is the average bet odds that you choose to invest in, and k is the dollar amount you spent for each bet.

From this equation, we could notice that

(1) In order to win money, we should have accuracy * avg bets > 1

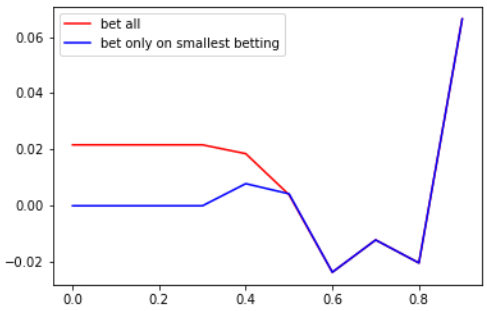
(2) The dollar amount you invest in for each bet, accuracy and avg bets determine how much you will win or lose. The higher value of them, the more you will win.

From our prediction (shown in the section below) by the random forest using team talent features, betting features, and net goal features, we have accuracy is around 0.538, and the bet and bet odds we choose based on our prediction should have at least 1.86 so that we could win money, which is unlikely. However, we could choose a subgroup of all the match outcome we predicted that have the highest (accuracy * avg bets) so that we could win money. Therefore, we test (accuracy * avg bets - 1) on groups by the combination below:

(1) Whether our bet has the lowest bet odds among home win, draw and away wins to ensure our bet has a lower risk.

(2) Does our prediction have over 0%, 10%, 20%, …, 90% probability that the prediction is correct. If the probability by random forest is over 80%, it means that the model is 80% sure that the prediction is correct. This ensures that accuracy will be large.

The result is presented in the following graph:



Where the x axis are model prediction probability, and y axis are (testing accuracy * avg bet odds we bet on that group - 1).

This means that we should bet on matches in the group where our model is over 90% sure about the prediction.

After having our strategy, we use 100 train-test splits to evaluate our returns. We choose to invest 1 dollar for each match in the group we described above. We found that we will get an average return of 1.04 dollars overall, with 80 of the tests having positive returns. That means if we have M dollars prepared and there are N matches in the group, we will win an average total 1.04 * dollars, and has average rate of return $\frac{104}{N}$%.

2. *Team building strategy*

We are trying to understand if there is a dominant team playstyle that could give us the maximum points per game, winning percentage, and net goals per game. Under each target value, we train Linear Regression, Gradient Boost Tree Regressor, and Random Forest Regressor models.

Under all three target columns, Linear Regression gives us the best MSE. We thus choose this model, and interpret the outcome from the coefficients of Linear Regression.

In order to get more points per game, buildUpPlaySpeed (feature 1), chanceCreationCrossing (feature 4), defencePressure (feature 6) have direct (positive) contribution, while buildUpPlayPassing (feature 2), chanceCreationPassing (feature 3), chanceCreationShooting (feature 5), defenceAggression (feature 7), defenceTeamWidth (feature 8) have negative contribution.

In order to get higher winning percentage, buildUpPlaySpeed (feature 1), buildUpPlayPassing (feature 2), defencePressure (feature 6) are factors have positive contribution to it, and chanceCreationPassing (feature 3), chanceCreationCrossing (feature 4), chanceCreationShooting (feature 5), defenceAggression (feature 7), defenceTeamWidth (feature 8) are factors with negative relation.

Factors that are positively related to net goals per game are buildUpPlaySpeed (feature 1), chanceCreationPassing (feature 3), chanceCreationCrossing (feature 4), defencePressure (feature 6), and factors that are negatively related are buildUpPlayPassing (feature 2), chanceCreationShooting (feature 5), defenceAggression (feature 7), defenceTeamWidth (feature 8).

Aggregating the previous outputs, we interpret if there is a dominant team playstyle. Feature 1 and 6 always positively contribute to our model and thus are great attributes to build a winning team; feature 5, 7, and 8 are always negatively related, thus should be considered as negative features. Other features, including feature 2, 3, and 4, should be considered as features with less importance.

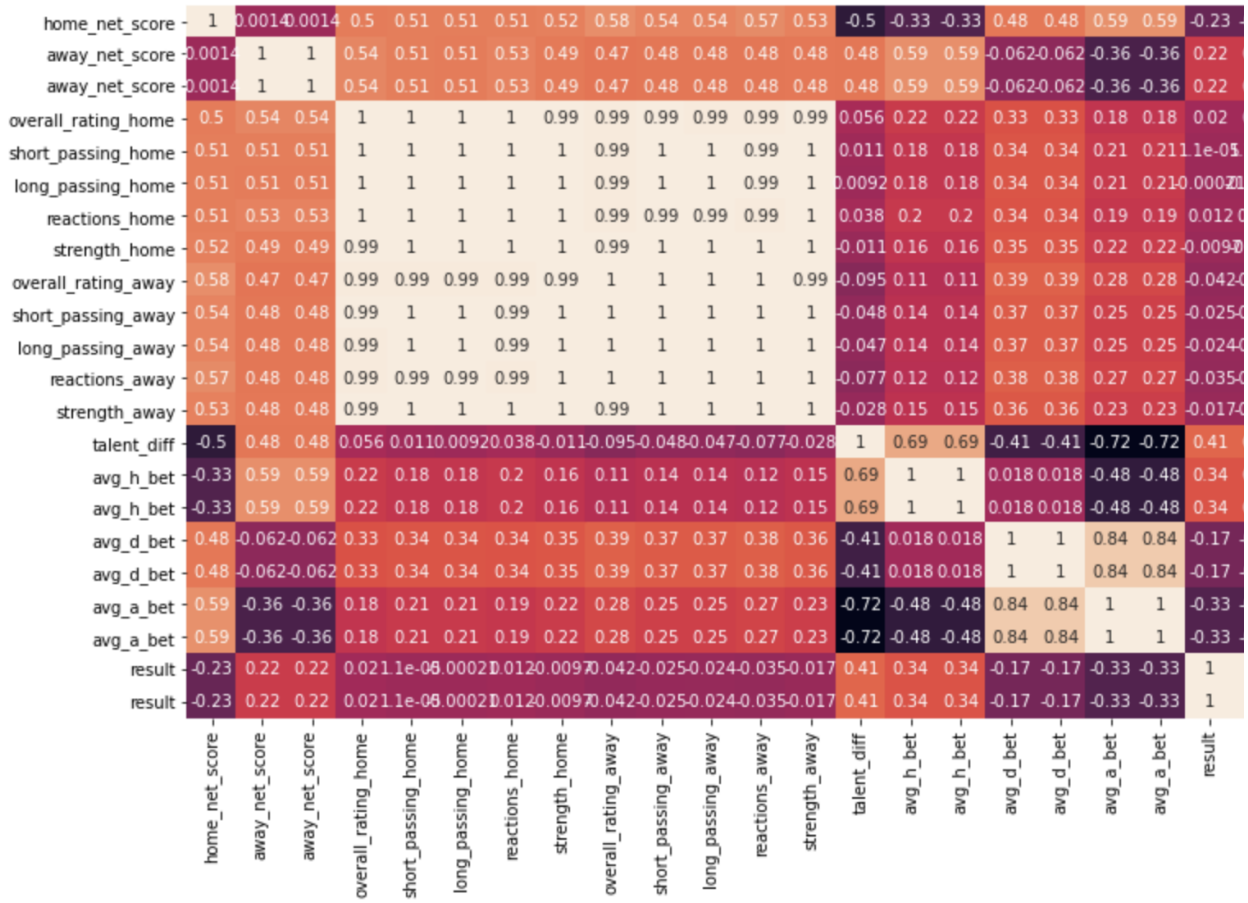
In building our team, the team should have higher pace in offense and higher defense pressure, and it should weigh less on chance creation shoot, in defense aggressiveness and width. If we want a team with better fan appreciation (higher net score per game), we should care more about the quality of crossing and passing in offense, so adding players that are good at these attributes should help. For a higher winning percentage, we should add players with specialties in passing.

3. *Using team talent to make predictions*

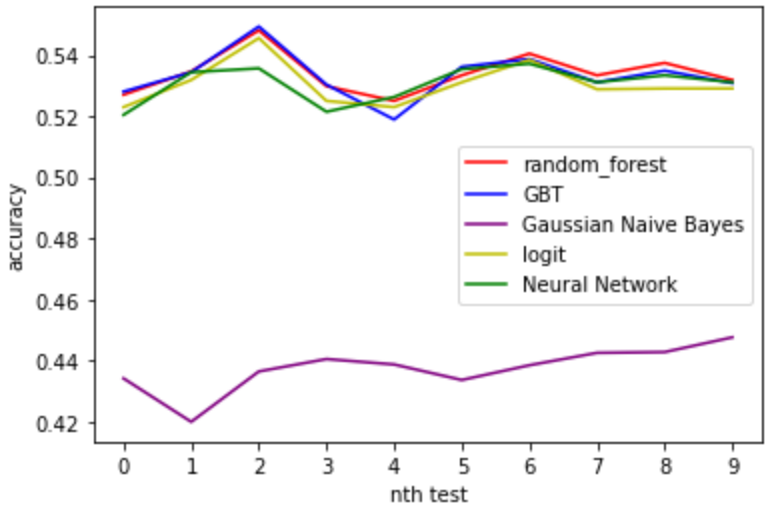
We want to see if team talent is a good predictor of match outcomes. We built models with the same model structure as the betting models but using team talent features as predictors.

Since the number of matches with missing player IDs is relatively small (less than 1/25) and we hypothesize that interpolation will add more bias than dropping in this case, we have dropped entries with missing player IDs.

We have trained the models separately, one time without the betting data and one time with the betting data added to the dataset. By using this scheme, we have a better understanding of whether team talents are useful information in addition to the betting data.



From the correlation heat map, we will eliminate highly correlated terms. Therefore, we will use only the talent_diff as additional input into the betting model.



By adding the talent difference into the model, we can see that prediction accuracy has increased for all models. The average accuracy has an average of 53.4% by Random Forest, which is our best predictor. This result means that talent difference is significant in determining the winning team.

Let's look back on what are the most important features that determine team talent level. From the Lasso feature selector, we get the following coefficients:

'short_passing': 0.021, 'long_passing': 0.040, 'reactions': 0.651, 'strength': 0.0333

We argue that if team improves on the above talents, especially reactions, they will have a better chance of winning.