Citadel APAC Data Open

Profitable betting strategy for EU football data using machine learning

Chen Ziao, Junwoo Yun, Xiaohe Qiao, Andrew Chan,

28th March 2021

1 Problem Statement

In this digital era, more football clubs start to incorporate data science into strategy development. For example, Liverpool FC and German football teams are using data science to analyze match patterns and maximize each player's potential. With the help of SAP Challenger Insights and Penalty Insights Function, Germany won the World Cup Championship in 2016.

In the Citadel APAC Data Open, European football teams, matches, and players data provided, we planned to prove that there's a systematic betting strategy with the help of machine learning models. Here we present our problem statement to be:

To research if it is possible to design a betting strategy using the machine learning model's prediction to consistently earn profits.

In the meantime, we also want to identify what types of features of the football team that contributes the most to the match outcome.

2 Executive summary

Our team went through EDA with data cleaning, feature selection, model construction, testing, betting strategy testing, and analyzing feature importance.

For EDA and data cleaning, we conducted exploratory data analysis on match, team, and player data separately with null preprocessing and feature creation. We consolidated pre-processed data into a single table for feature selection and later used it for modeling.

For feature selection, we conducted Spearman's Correlation Test using Rolling Window Approach, Chi-square Test, and VIF Test.

For model construction, we used a fined-tuned rolling window approach for training, validation, and testing. We tested linear, bagging, and boosting tree models with hyperparameter tuning. As the test result, we discovered that tree models outperformed over linear and boosting models in terms of accuracy and RMSE.

For betting strategy testing, we implemented two betting strategy using test prediction result and achieved a **positive Sharpe ratio of 0.92 for Extra Tree Model Regressor**

For feature importance, we discovered that best player, ability to mark, to make volley shoots, and to cross accurately makes a good team

3 Technical Exposition

3.1 Data Cleaning and Exploratory Data Analysis

We decided to perform the data pre-processing and EDA separately for players, matches, and teams. After that, we consolidated separate data into a single table for feature selection and modeling in later stages.

3.1.1 Player Data

1. Player Null Values
    a. 836 rows **without all player_attribute** data were dropped.
    b. 911 rows **without the feature "attacking_work_rate"** assumed to be a defender or goalkeeper's attribute which could not be assigned with attacking skills.
    c. 1877 rows **without 7 features (volleys, curve, agility, balance, jumping, vision, sliding_tackle)** assumed could not be assigned due to the player's position or role.
    d. Null values in (b) and (c) were handled in the data consolidation part by averaging all the player's attributes in the team.
2. Player feature creation
    a. We decided to create a feature **"age"** for each player using the **"timestamp of player data"** minus **"birthday"**
    b. We also created the feature "season" for each match based on the match date, this is for later connecting the match and player tables together.

3.1.2 Match Data

1. Within the match data, we divided columns into three-part to preprocess null values
    a. Match Data (from country_id to away_team_goal, information about matches)
    b. Player Data (from home_player_1 to away_player_11, information about players)
    c. betting Data (from B365H to BSA, information about betting)
2. All match data was given without Null values.
3. 243 games **without both "Player Data" and "betting Data"** were dropped. Those rows were only from 2008-2009 with 4 specific leagues.
4. 515 games **without "Player Data"** were dropped. Those rows were only from 2008-2009 with 3 specific leagues, and 87.7% of matches with this filter do not have data.
5. There were 3533 games **differ in player numbers** of home and away team, but no such game was 11 player vs 0 player. Those Null values were handled in the data consolidation part by averaging all the player's attribute in the team

3.1.3 Bid Data

1. In terms of the league, **Poland Ekstraklasa (15722) and Switzerland Super League (24558)** which did not contain any betting information were dropped.

2. Specific betting companies had null value; **PS** (2008/ 2009 ~ 2011/ 2012), **SJ** (2014/ 2015 ~ 2015/ 2016), **GB** and **BS** (2013/2014~) were dropped with null value. Furthermore, **41 matches** without any betting information were **dropped.**

3.1.4 Team Data

1. **Features with the name '_Class'** were dropped since they were perfectly correlated with their continuous feature but had less information.
2. Feature **'buildUpPlayDribbling'** was dropped because it did not have value from **2010 to 2013** which was a crucial period in training data.
3. We dropped one **duplicated** team record (Team ID: 9996)

3.1.5 Target Variable

From the number of goals in match data, we derived our target variable 'match_result' and 'match_result_rel'.

1. We derived categorical target variables, "**match_result** (**win, lose, draw)",** from the number of goals of home and away teams in each match for classification. We decided to label it from the perspective of the home team, meaning we labeled it as a "win" if the home team wins.
2. We derived continuous target variable, **"match_result_rel"**, in a range of -1 to 1, derived with a formula for the regression as below:

$$match\_result\_rel = \ 0 \ \ if \ draw \ else \ \frac{(home\_goal \ - \ away\_goal)}{max(home\_goal, \ away\_goal)}$$

3.2 Data Consolidation

Here are the steps to combine team, player, and match data into a single data file:

1. We matched each match with the **most recently available** players' data and match data.
2. For each match, all continuous player features in each team were aggregated in three ways: **average**, **maximum,** and **standard deviation**.
    a. **Average**: We used them to represent the average skill level of the whole team for each player attribute.
    b. **Maximum**: We used them to navigate the effect of the best skills of the player in each team.
    c. **Standard Deviation**: We used them to represent the consistency of skills among the team members.
3. For each match, all categorical player features were aggregated using a **mode** of the class value of each team.

4. To reduce the dimension of data, we decided to convert all features into a **"Home - Away"**; each Home team's feature was subtracted by the Away team's feature.

After we completed the data consolidation, the single data file comprised **16914 matches** and **129 features**, spanning from **season 2009/2010 to season 2015/2016**.

3.3 Feature Selection

To further reduce the dimensionalities, we have conducted feature selections with the following steps:

1. Data were split into **training data(70%)** and **testing data (30%)**. The data was sorted by match date first (ascending) then split chronologically instead of splitting data randomly to ensure no future information used during model training.
2. Feature selection for continuous features was conducted using Spearman's Correlation.
    a. Due to the time-series nature of our training data, we have decided to conduct the **Spearman's Correlation Test** using **Rolling Window Approach**. The Spearman's Correlation Test statistics are shown below:

$$Z = r * sqrt(n - 1),\ Z = Z \sim N(0, 1)$$

    b. The Rolling Window Approach has conducted the test only within a fixed-sized chronologically moving window. We obtained a time series of correlation values and its p values. The window size was set as 600 (around ¼ of a season).
    c. Using the two-time series, we computed the abs(mean of correlation), mean (p-value), correlation standard deviation, and p-value standard deviation. The mean correlation and standard deviation metrics were ranked descendingly as a high magnitude of correlation and stability of correlation and p-value were highly desired, but mean p-value was ranked ascendingly as low p-value was desired.
    d. We designed a formula to rank the features

$$Final\ Rank = Corr\ Mean\ Rank + P\ Value\ Mean\ Rank + \frac{(Corr\ Std\ Rank + P\ Value\ Std\ Rank)}{2}$$

    e. We discovered that all p values are between 0.3 to 0.6, indicating that the correlation test failed to distinguish useful features well. Hence, we decided to drop extreme features using the threshold and selected the **top 90 features** among the 118 continuous features based on their Final Rank.

	Feature	Cor Mean	Cor Std	P Value Mean	P Value Std	Cor Mean Rank	Cor Std Rank	P Value Mean Rank	P Value Std Rank	Total Rank
0	avg_potential_diff	0.365035	0.021879	1.003928e-06	3.755149e-06	2.0	2.0	7.0	7.0	13.5
1	avg_reactions_diff	0.356791	0.029571	4.911649e-07	1.771572e-06	3.0	17.0	3.0	2.0	15.5
2	max_overall_rating_diff	0.347166	0.028503	4.820443e-07	1.786471e-06	6.0	13.0	2.0	3.0	16.0
3	avg_overall_rating_diff	0.372563	0.029891	9.857007e-07	3.674611e-06	1.0	19.0	6.0	6.0	19.5
4	avg_short_passing_diff	0.348830	0.031098	1.511524e-06	5.478930e-06	5.0	21.0	8.0	9.0	28.0
5	avg_vision_diff	0.335127	0.033117	8.385252e-07	3.114340e-06	7.0	28.0	5.0	5.0	28.5
6	avg_ball_control_diff	0.353890	0.026537	6.657075e-06	2.485721e-05	4.0	9.0	16.0	16.0	32.5
7	avg_long_passing_diff	0.330469	0.034159	2.539599e-06	7.468046e-06	8.0	31.0	12.0	11.0	41.0
8	max_potential_diff	0.330025	0.027676	8.099472e-06	3.015202e-05	9.0	12.0	18.0	19.0	42.5
9	max_reactions_diff	0.312744	0.029776	7.364402e-06	2.643609e-05	12.0	18.0	17.0	17.0	46.5
10	max_ball_control_diff	0.319334	0.027414	2.560156e-05	9.560774e-05	11.0	11.0	22.0	23.0	50.0

<table showing top 10 features according to Final Rank>

3. We selected Categorical Feature univariately using the **Chi-square Test** based on a contingency table.

a. There are in total 8 categorical features.
b. We dropped **"season"** since we utilized chronologically moving window functions
c. We discovered **"league"** was an important factor because each league had unique characteristics
d. For each of the 6 remaining categorical features, we conducted a **Chi-Square Test** on it against the categorical target variable "match_result" using Rolling Window Approach.
    i. Derived the contingency table using feature vs match outcome
    ii. Computed the Chi-square test statistics based on the contingency table.
    iii. Using the chi-square distribution, computed the p-value.
e. We noticed that all 6 categorical features had their mean p-value in the 0.25 to 0.4 range, which was not statistically significant. Hence, we failed to reject the null hypothesis that they had no relationship with the "match_result" and we decided to drop them.

4. For the all selected variables, we conducted a **VIF Test** with a threshold of VIF score of 5 and dropped features with high multicollinearity to reduce the data dimension and the chance of overfitting.

After feature selection, we **selected 78 features from 129 features** for model training and testing. We intentionally loosened the feature selection threshold since some features could be useful as control variables in different models.

3.4 Model Construction and Fine-tuning

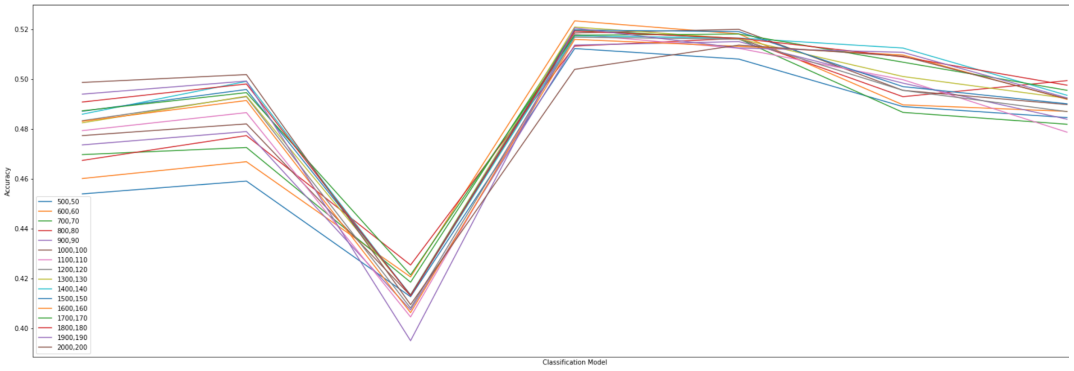
With given data and features, we trained and tuned **7 classification models** and **10 regression models.**

1. Model Selection
    a. For the classification, we chose **two linear models, three tree models,** and **two boosting models**.
    b. We chose linear models due to speed, generalization, and ease of tuning.
    c. We chose bagging and boosting tree models for regularized and powerful models.
    d. For the regression, we chose **five linear models, three tree models, and two boosting models** for the same reasons above.

Model \ Purpose	Classification	Regression
Linear Model	Linear Discriminant Analysis Ridge Classifier	Orthogonal Matching Pursuit Linear Regression Huber Regressor Bayesian Ridge Lasso
Tree Bagging Model	Decision Tree Classifier ExtraTrees Classifier RandomForest Classifier	Decision Tree Regressor ExtraTrees Regressor Random Forest Regressor
Tree Boosting Model	XGBoost Classifier LightGBM Classifier	XGBoost Regressor LightGBM Regressor

<Table of model summary>

2. Window Size Tuning



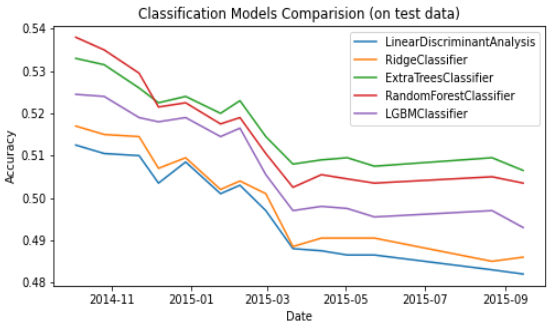
< model vs accuracy with each line representing
[training window size, validation window size] >

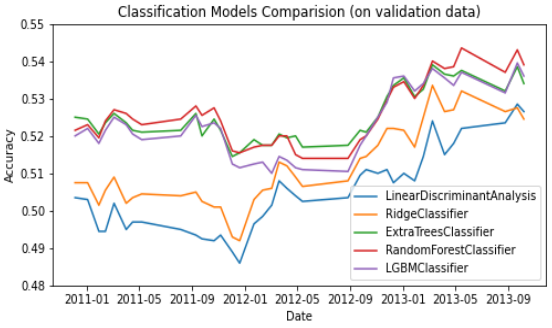
a. We tuned the window size since it decided the amount of information available for the model to be trained and fine-tuned
b. Considering each season have 2000 games available, we tuned the model with a training window size (100~2000) and a validation window size (10~200)
c. As **bigger window** sizes gave much information, the brown line (2000, 200) in the graph had **at most 10%** difference in predicting power with the **best stability** compared to other window sizes.
d. After tuning window size, we performed hyperparameter tuning since data availability will affect predictive power and thus results in different best hyperparameters.

3. Model Tuning
   We used **Optuna** for hyperparameter tuning. Most models had **at least 5% improvement** in validation results and adjusted not to be overfitted using hyperparameters.

3.5 Model Result and Discussion

1. Classification





Classification Models (on test data)

	model	Accuracy
0	LinearDiscriminantAnalysis	0.496200
1	RidgeClassifier	0.502600
2	DecisionTreeClassifier	0.427800
3	ExtraTreesClassifier	0.522600
4	RandomForestClassifier	0.521200
6	LGBMClassifier	0.511000

Classification Models (on validation data)

	model	Accuracy
0	LinearDiscriminantAnalysis	0.505612
1	RidgeClassifier	0.511122
2	DecisionTreeClassifier	0.417245
3	ExtraTreesClassifier	0.524694
4	RandomForestClassifier	0.525102
6	LGBMClassifier	0.523061

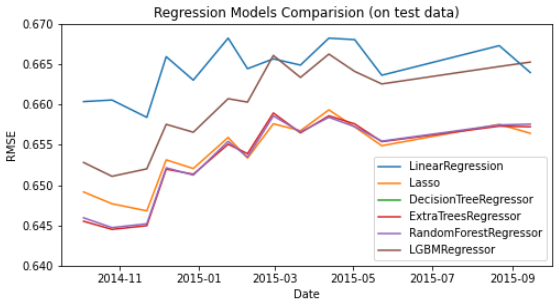
We discovered that as time went by the validation accuracy went up while the test accuracy went down.

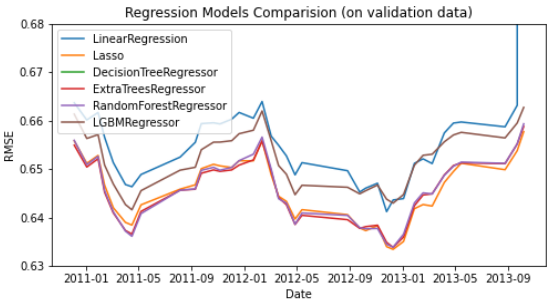
a. Due to **seasonality** (team attribute and player attribute was updated at a certain time period), the prediction could be affected
b. Due to **y ticks size**, the graph seems fluctuating but considering Extra Tree and Random Forest had only 0.01 difference in accuracy, we discovered the graph didn't show significant fluctuations of predication power.

Extra Trees and **Random Forest** outstood other classifiers on both validation and test datasets. Tree models are more robust. Random Forest and ExtraTrees could limit overfitting without substantially increasing eros due to bias.

Decision Tree gave the weakest performance because it's prone to overfit with non-bagging and low complexity.

2. Regression





Regression Models (on test data)

	model	RMSE
0	LinearRegression	0.441297
1	Lasso	0.426996
2	DecisionTreeRegressor	0.602963
3	ExtraTreesRegressor	0.425598
4	RandomForestRegressor	0.425974
6	LGBMRegressor	0.434852

Regression Models (on validation data)

	model	RMSE
0	LinearRegression	708785527748719935488.000(
1	Lasso	0.421508
2	DecisionTreeRegressor	0.580675
3	ExtraTreesRegressor	0.421253
4	RandomForestRegressor	0.422145
6	LGBMRegressor	0.429307

Lasso, Random Forest, and **Extra Trees** Regressor were three of the strongest learners with an average RMSE of around 0.4 on both the validation and test datasets. While Decision Tree Regressor save the weakest performance with an average of RMSE of around 0.6

Lasso regression performed well due to regularization with L1-norm penalty, reducing the coefficients to zero. We implemented Lasso for generalized betting strategy having continuous profits

Linear regression performed poorly when there're no linear relationships. Based on extremely high and fluctuated RMSE, we can conclude that data exhibits non-linear relationships.

3.6 Betting Strategy Testing

Referring back to our problem statement, we decided to implement two simple betting strategies for classification and regression instead of finding the best strategy.

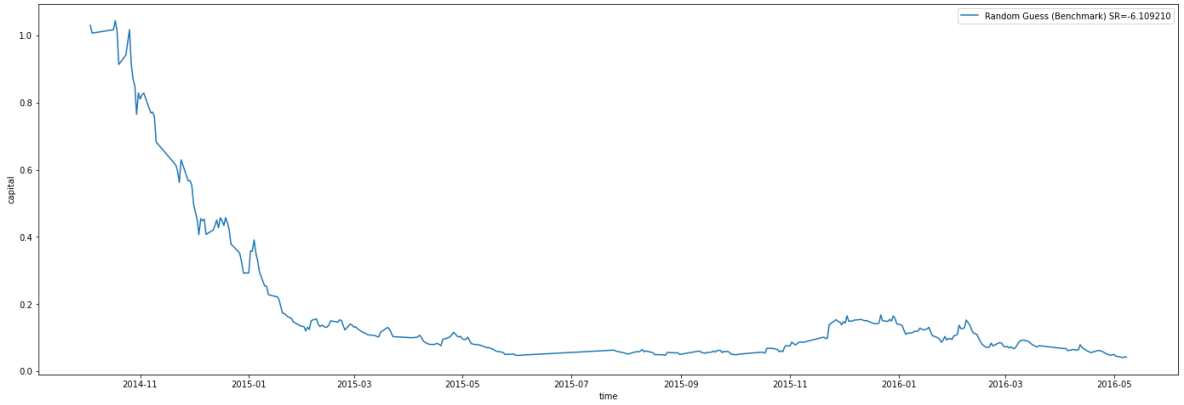
We used the predictions from test data (2014-10-04 to 2016-05-08) and constructed two betting strategies and Random Guess Strategy as our benchmark. We decided to use annualized Sharpe Ratio and cumulative return to evaluate the strategy performance.

3.6.1 Random Guess Strategy (Benchmark)

Before applying any betting strategy with test predictions, we selected Random Guess Strategy as a benchmark to show whether the machine learning models can outperform it.

a. It randomly chose the outcome of the matches to bet and always bets for all matches.
b. We assumed for each day it would only invest 10% of his current capital (so that it would never lose all capital and could keep betting all the time).
c. We also assumed that it would allocate the 10% equally to all the matches in a given day.
d. We assumed the strategy starts with 1 as its initial capital.

The above assumptions were also applied for the betting strategy using machine learning predictions.



<Benchmark: capital cumulative return vs time with Sharpe Ratio>

From the line charts, we discovered Random Guess Strategy lost the capital very fast in early periods and gave a very negative Sharpe Ratio -6.1. **It implied that we should never bet randomly!**

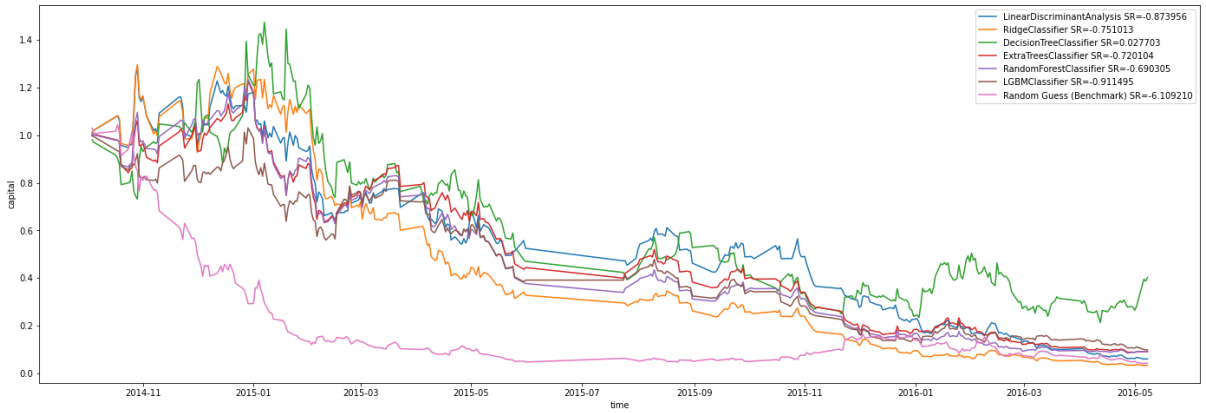
3.6.2 Betting Strategy for Classification Result

A brief introduction to the strategy is as below:

1. Took the test data classification predictions and extracted the probabilities of win, lose and draw for each test match.
2. Chose the predicted match outcome with the highest predicted probability.
3. Ensured that the strategy only bet when the machine learning model was confident on the predicted results, implementing a probability threshold at 0.4. For example, if the model predicted three outcomes in probability 0.31, 0.31, 0.38, then the strategy would not bet for this match.
4. For all matches on the same day, the strategy would simply equally weigh the capital allocated to each match for betting. We assumed a rational investor only reinvests a portion

of his total capital each day because it was unreasonable for a risk-averse investor to invest all his capital at one time. We have set the reinvestment rate for each day as 10%.

After applying the strategy to all classification models' prediction on test data, we obtained betting strategy performance in time series for each model. Assuming a **risk-free rate** during 2014 to 2016 to be 2.1% (from the US 10 Year Treasury Bill), we computed Sharpe Ratio for each model. We tested different betting companies and the results were close and hence, we chose to only show the result from betting company B365 as a sample graph below.



<Classification models: capital cumulative return vs time with Sharpe Ratio>

From the line charts, we discovered all models appeared to earn some profits at the early stages between 2014-11 to 2015-01, but started to lose capital and converged to 0. Before the convergence, all models greatly outperformed the benchmark, meaning machine learning models could help to predict the result slightly more correctly but not up to the point to make the strategy consistently earn profits.

Model	**Sharpe Ratio**
LDA	-0.87
Ridge Classifier	-0.75
Decision Tree Classifier	0.03
Extra Tree Classifier	-0.72
Random Forest Classifier	-0.69
Light GBM Classifier	-0.91
Random Guess (Benchmark)	-6.11

Although all models outperformed the benchmark, only **Decision Tree Classifier** was able to achieve a positive Sharpe Ratio. It was expected that LDA was almost the worst model due to its low complexity. However, we were surprised that **Extra Tree** and **Random Forest** did not outperform Decision Tree Classifier even if the test accuracy in section 3.5 implied the contrary. We guessed it

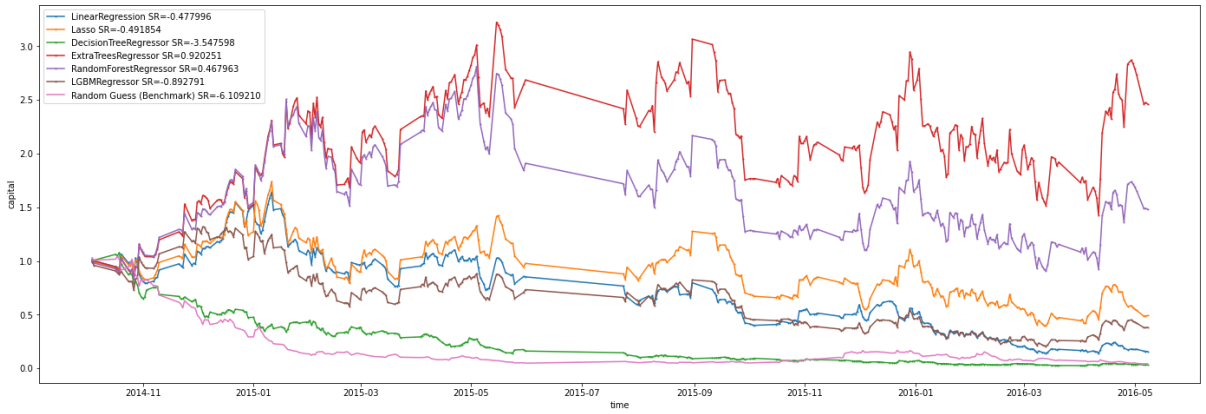
might be due to the fact that Bagging methods were good at betting matches with small odds so that even if they exhibited high accuracy, they failed to predict matches with high odds/return correctly.

3.6.3 Betting Strategy for Regression Result

The betting strategy for regression is very similar to the classification strategy described in 3.6.2, with the following modifications:

1. Since the regression target variable was a continuous variable in [-1, 1], we decided to simply split the range equally for loss [-1, 0.33), draw (-0.33, 0.33) and win (0.33, 1].
2. Same as classification, we assumed that the model would not perform well when its prediction was close to the range boundary of each outcome. Hence, we reduced the range for lose, draw and win be [-1, -0.4), [-0.3,0.3] and (0.4,1]

Same as 3.6.2, after applying the strategy on all regression models' predictions, we obtained the strategy performance in time series format for each model. Assuming the risk-free rate to be 2.1%, we computed Sharpe Ratio using the same betting company with the reason described above.



<Regression models: Capital cumulative return vs time with Sharpe Ratio>

From the line chart, we could discover that all betting strategies with test prediction data outperformed the benchmark except for the Decision Tree Regressor. **Extra Tree** and **Random Forest** Regressor outperformed other models and they were the only two that had final capital greater than 1. Lasso slightly outperformed Linear Regression if we just looked at the line charts.

Model	**Sharpe Ratio**
Linear Regression	-0.48
Lasso Regression	-0.50
Decision Tree Regressor	-3.55
Extra Tree Regressor	0.92
Random Forest Regressor	0.47
Light GBM Classifier	-0.89
Random Guess (Benchmark)	-6.11

From the Sharpe Ratio table, even if Lasso showed better performance on the line chart compared to Linear Regression, its Sharpe Ratio was lower due to its high volatility (risk).
Extra Tree Regressor gave the highest Sharpe Ratio as 0.92, followed by Random Forest Regressor 0.47 and they were the two models that gave positive Sharpe Ratio. Furthermore, all machine learning models would help the betting strategy achieve a much better Sharpe Ratio than our Random Guess benchmark.

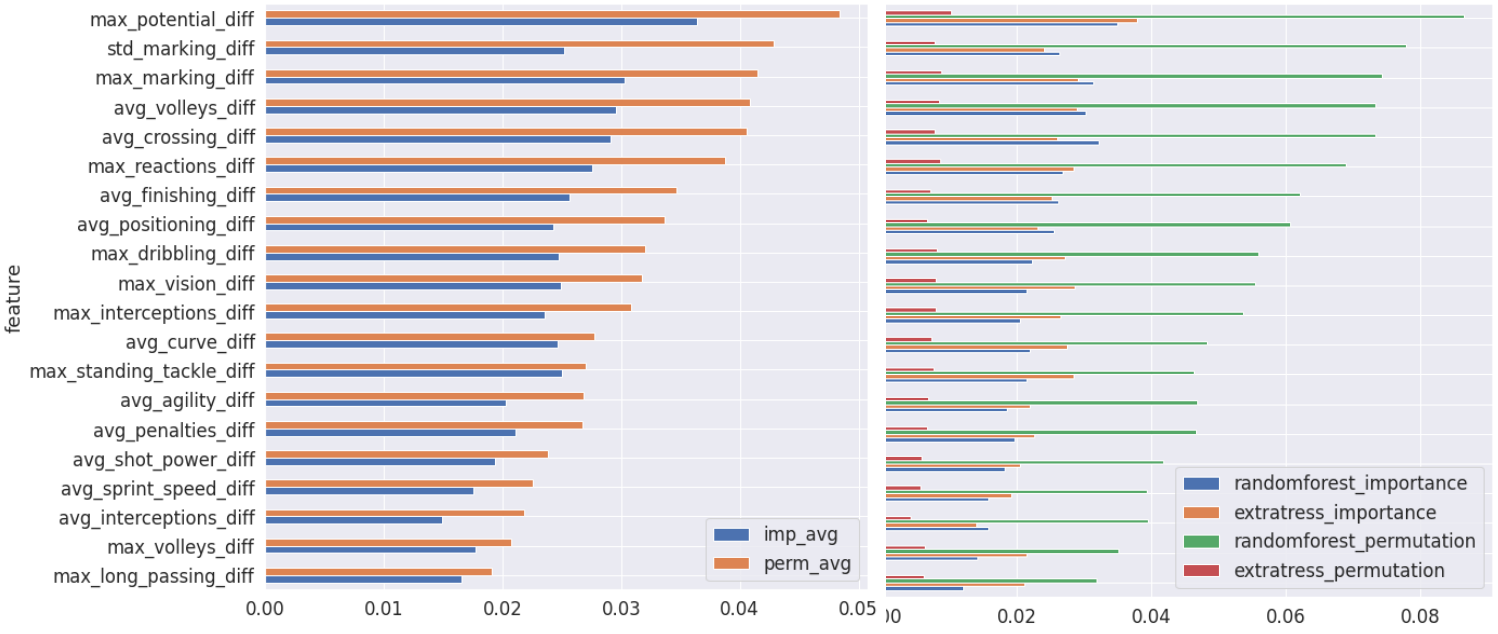
Results from Regression predictions were much better than classification prediction in terms of Sharpe Ratio. Since classification target variables are categorical (win, lose or draw), it naturally contains less information available compared to the continuous regression target variable [-1, 1]. We assumed information of the score differences between two teams in a match, which to some extent, included the relative strength information of two teams. And this relative strength information really helped the model to boost performance.

3.7 Feature Importance Discussion

There are three types of feature importance computation methods; Filter method, Wrapper method, and Embedded method.

a. Filter methods (EDA)
   We conducted Spearman's Correlation Test using Rolling Window Approach, Chi-square Test, and VIF Test.
b. Wrapper methods
   Model selectively used feature subset with specific learning algorithms to see the usefulness of features for prediction. We conducted **permutation importance** with the **eli5** library measuring the increase in the prediction error of the model after we permuted the feature's values.
c. Embedded methods (Tree-based models)
   Models accumulate the feature importance by measuring the impurity of the division of trees.

Two best models based on betting strategy Sharpe Ratio, Random Forest, and Extra Trees were selected and derived feature importance, results in below graphs.



Evaluation of top 5 feature importance

1. **max_potential_difference:**
   max potential represents the best player in the team, called “star players” like Messi. By seeing avg_potential_difference, not in the top 20 features, the difference in the best player’s ability is the most important feature for predicting the winner.
2. **max & std_marking_difference:**
   zone/man-marking is an essential skill in soccer. The player could prevent opponents from gaining control of the ball and limiting their influence by marking them.
3. **avg_volleys_difference:**
   volley shot is a difficult skill, requiring the ability to ball-touch, and good timing with high shooting accuracy. If average volley shot ability is higher, players on average seem better in different attribute skills as well, resulting in victory.
4. **avg_crossing_difference:**
   To make the accurate crossing, the player should consider the available zone and position of both allies and opponents, which is a high-demanding skill and directly associated with the success of attacking or making a goal. Thus, a higher average crossing difference is associated with more chances to make successive attacks.

4 Future Work

1. Ensemble and Stacking best models for the better predictive performance
2. Add more features such as winning/losing momentum of each team at a certain time.
3. Optimize the betting strategy by even tuning the reinvestment rate and prediction threshold during the training and validation stage
4. Design more sophisticated mathematical betting strategy using Kelly’s formula

5 Conclusion

Based on the betting strategy performance with test data, we could conclude that with the help of the team, player, and match information and specific machine learning models, we could bet much smarter than Random Guess Strategy. Given positive Sharpe Ratio in tree models, we successfully proved that we could design a **betting strategy achieving a positive Sharpe Ratio 0.92** using given data and advanced machine learning models. Using feature importance from modeling, we could conclude the best player, ability to mark, to make volley shoots, and to cross accurately result in the best team likely to win for betting