BITS F464 - MACHINE LEARNING

MAJOR PROJECT
PREDICTING THE OUTCOME OF THE EPL

Made By:

Ashutosh Sharma	2018A7PS0179G
Aalok Singh	2018A7PS0198G
Karthik Suresh	2018A3PS0301G
Avdhoot Bhandare	2018A7PS0763G

Table Of Contents

Introduction

The English Premier League is by far the most watched professional football league, with around *12 million viewers per game.* It beats the next most popular league, the Spanish La Liga (2 million viewers per game) by a significant margin. With such a large viewer base and with the stakes so high, we can see why one would want to predict the results of the EPL.

The EPL has 20 teams that compete for 1st place, with each team playing every other team twice, one at home and one away, for a total of 380 matches played from August to May of the next year. The league has a point system, with the team with the highest number of points qualifying as the winner.

We want to predict the winner of the EPL using Machine Learning algorithms. For this it is necessary to choose features that will accurately and relevantly represent the team, like score, passes, shots on target, etc. Not only in game statistics will be used, we will also consider other factors such as expenditure, roster changes, etc which are also indicative of the form of a team. Once we have these features, we will feed them to a Machine Learning algorithm like Logistic Regression, K Nearest Neighbours, Support Network Machine, etc. The aim is to classify each match into one of three classes, either a win, a draw or a loss with respect to the home team.

Abstract

The main objective of our model is to explore different Machine Learning algorithms to predict the result of a football match using in-game match statistics. We will explore different model design hypotheses, feature reduction and engineering techniques, and hyperparameter tuning, and analyze our model's performance against actual results.

Many techniques to predict the result of a professional football match have used the number of goals scored by each team as a measure for evaluating a team's performance and estimating future results. A final target variable which represents home team victory, away team victory or draw is produced. We thus follow a multiclass classification technique and choose the best among various classification techniques.

However the number of goals scored by a team has a random element involved as we see during games and periods of good play during the game. This leads to inconsistencies in many games between the number of goals scored or conceded and the team's performance. So we needed another metric to better evaluate a team's performance and predict results.

In our project, we used an 'expected goals' metric which predicts the expected number of goals scored by each team within a single game. As in football only the number of goals scored can determine the winner, this expected goals function is used as a regression task. Instead of just predicting the outcome of each game for the home team with a classifier, we

build a model to generate synthetic match statistics, and use those to calculate the expected goals for both teams per match.

It is crucial to choose relevant features after analyzing their influence on a match's outcome. We want to avoid too much bias or variance in our model. From intuition we can use several relevant features and later on reduce the dimension using various dimensionality reduction techniques or Principal Component Analysis. We used many in-game statistics like Shots taken, Possession, Corners, Fouls etc as well as season long statistics for teams such as Expenditure, Income. Score (goal) is implicit as it determines the winner and is used to create our ‘target’ variable, which is used to classify between win, loss or draw. We also develop a new feature called ‘form’ which is a measure of “streakiness” and is based on exponentially decaying importance on goal difference as look at the past games. Each feature is available as home and away stats.

After training and analyzing our model on the dataset available from 2007 to 2017 English Premier League data, we will predict the final standings of the league table of year 2018.

After getting an initial result, we will increase the number of runs we do and take the average of final standings. We get a decent model which can predict top teams correctly and looks like a possible finish to a league season.

Machine Learning Techniques

Logistic Regression

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The target variable is usually a binary variable, but the method can be modified to have multiple target classes. It is one of the simplest ML models and is easy to implement, and efficient to train, but the main disadvantage is that it assumes linearity between the target and feature variables, which is rarely the case in the real world.

Naive Bayes

This method is based on the Bayes Theorem, with the added assumption that the features are independent of each other (hence, Naive). Naïve Bayes Classifier is one of the simple and most effective classification algorithms which helps in building the fast machine learning models that can make quick predictions. As may be evident, the main drawback of this method is that we assume independent features, which is also rarely observed in real life.

Gradient Boosting Classifier

Gradient Boosting Classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. The idea behind "gradient boosting" is to take a weak hypothesis or weak learning algorithm and make a series of tweaks to it that will improve the strength of the hypothesis/learner. Decision trees are usually used when doing gradient boosting. Gradient boosting models are becoming popular because of their effectiveness at classifying complex datasets.

Introducing hyperparameters here can be useful in improving model performance. Gradient Boosting Classifier supports both binary and multi-class classification.

Neural Networks

An artificial neural network learning algorithm, or neural network, or just neural net, is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form. The neural net learning algorithm instead learns from processing many labeled examples (i.e. data with "answers") that are supplied during training and using this answer key to learn what characteristics of the input are needed to construct the correct output. Once a sufficient number of examples have been processed, the neural network can begin to process new, unseen inputs and successfully return accurate results. The more examples and variety of inputs the program sees, the more accurate the results typically become because the program learns with experience. The major problems with NNs are that it requires a lot of processing power, and if something goes wrong, it is difficult to determine where exactly the problem lies.

Support Vector Machines (SVMs)

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N - the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence. Again, this is used for binary classification, since the plane splits the data into two classes, so it is of limited use to us since our target has 3 classes.

K Means Clustering

The objective of K-means is simple: group similar data points together and discover underlying patterns. To achieve this objective, K-means looks for a fixed number (k, surprisingly) of clusters in a dataset. We define a target number *k*, which refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the center of the cluster. Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. In other words, the K-means algorithm identifies the ‘*k’* number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible. The *‘means’* in the K-means refers to averaging of the data; that is, finding the centroid. The problem with this method is that we have to specify the number of clusters present, i.e. the algorithm cannot identify this by itself. Another issue that pops up is that the outliers cause significant changes in the means, and need to be trimmed off beforehand.

Methodology

We tested and applied several machine learning models to a dataset describing English Premier League (EPL) to predict the outcomes of matches, as well as the final rankings of the participating teams in the league, with a high degree of accuracy.

Data Set

The dataset used has been scraped from the premier league website (www.premierleague.com) using Selenium. Yearly data from 2007 to 2018 was collected and merged together to form a base for predicting the match outcomes. The session data of the previous nine years was used to simulate the 2018 session. The dataset thus obtained has multiple features including teams names, match_ID, number of shots made and goals scored by both the teams, possession times, touches, passes and a lot of other features which we considered to be of possible relevance to impacting the match outcome, stored in a readily usable .csv format. The output label (target) represents the match outcome: an output of 1 denotes Home team wins, 0 denotes a draw, and -1 denotes Away team's victory.

Feature Selection

In predicting the outcome of a match, while we expect most of the features to be relevant, MatchID, which uniquely identifies each data point, had to be dropped. Team names being non-numerical were also dropped. The year was considered irrelevant for our prediction. Finally, we also had to do away with the respective scores of both the teams, as it directly concludes the match results.

Our features for simulating the 2018 season of the EPL were much different. Its data points contain only the information of a single team's performance in a certain match, along with the year, to be useful to predict the team's performance in our target year.

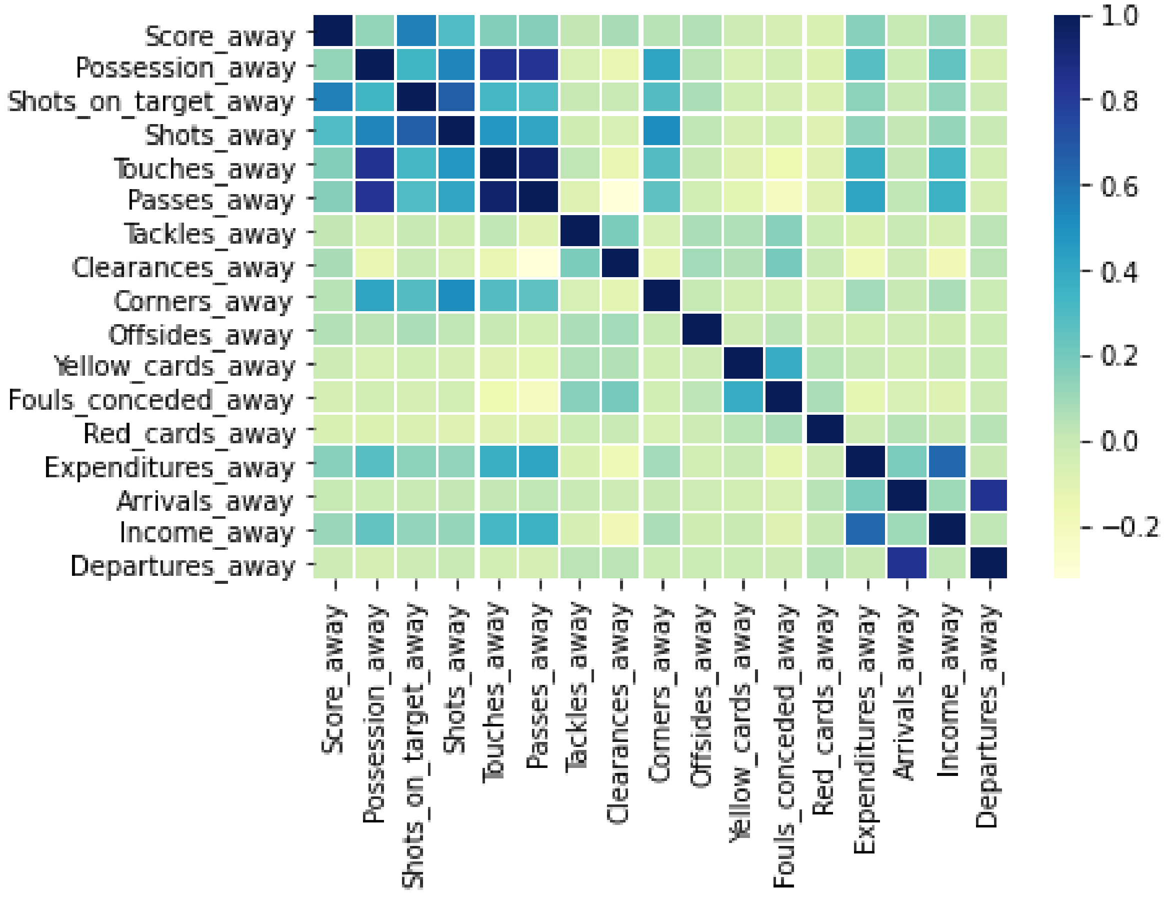
We further employ feature reduction techniques to improve the interpretability and robustness of our program.

Finding Correlations

Our data frame initially has 4556 rows and 39 columns. Each row denotes a game played between two teams with in-game home-away stats as well as season long stats. All the features which are initially captured by the columns include:

```
['MatchID', 'Home_team', 'Away_team', 'Score_home',
'Score_away','Possession_home', 'Possession_away', 'Shots_on_target_home',
'Shots_on_target_away', 'Shots_home', 'Shots_away', 'Touches_home',
'Touches_away', 'Passes_home', 'Passes_away', 'Tackles_home',
'Tackles_away', 'Clearances_home', 'Clearances_away', 'Corners_home',
'Corners_away', 'Offsides_home', 'Offsides_away', 'Yellow_cards_home',
'Yellow_cards_away', 'Fouls_conceded_home', 'Fouls_conceded_away',
'Red_cards_home', 'Red_cards_away', 'year', 'Expenditures_home',
'Arrivals_home', 'Income_home', 'Departures_home', 'Expenditures_away',
'Arrivals_away', 'Income_away', 'Departures_away', 'target']
```

We started by calculating the Pearson's correlation matrix to ensure that our features provided unique information, and to possibly reduce the dimensionality of our dataset. However, we find that our features are mostly independent of each other.



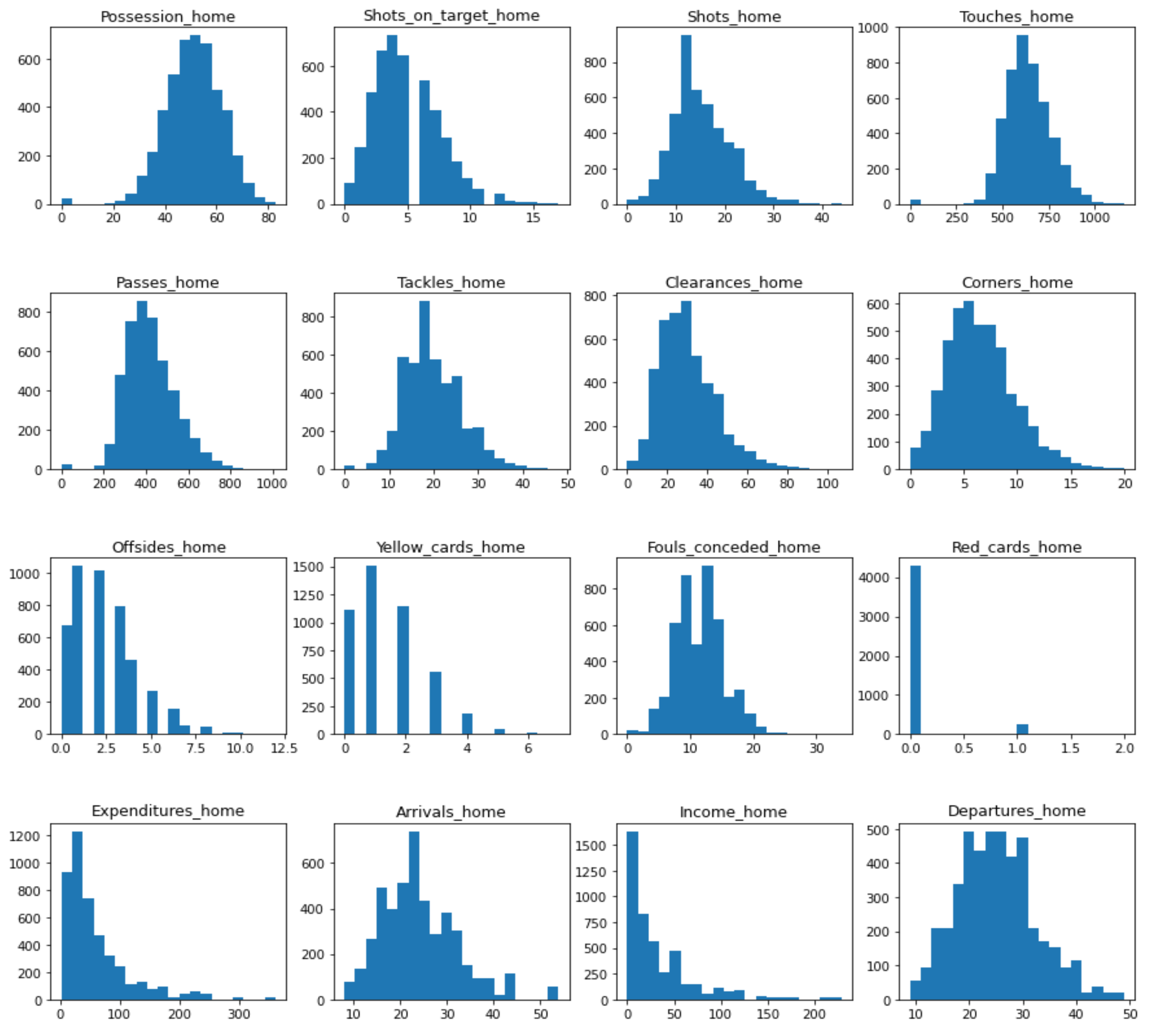
From this we see that few features look strongly correlated. Examples of such pairs are

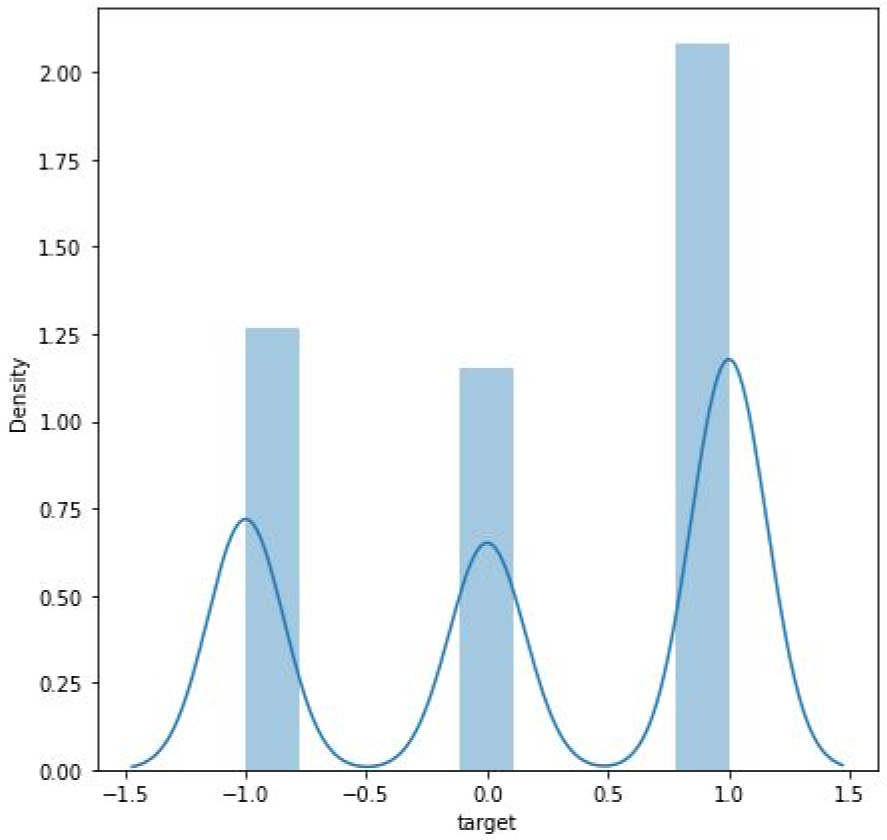
1. Possession home (away) and touches/passes home (away)
2. Shots home (away) and Shots on target home (away)
3. Score home (away) and Shots on target home (away)
4. Departure home (away) and Arrivals home (away)
5. Expenditure home (away) and Income home (away).

These correlations are also fairly intuitive as more possessions leads to more passes and definitely more touches, more shots can lead to more shots on target, and more income can lead to more expenditure in the transfer market.

So we can remove some feature whose essence is captured by some other feature and thus decrease the number of columns (or dimensions).

We also look at the histogram of the features to get an idea about the distribution.





The target variable distribution.

From this we see that the home team wins more often (as shown by more number of 1s) followed by away team wins (shown by lesser count of -1s). So we can conclude here that the home team has an advantage over being the away team for some games. So this feature might be an important one.

Feature Engineering

Here we modify features set of the raw data, to represent the data in a better format, to the predicting models. Recent performance has some predictive power in explaining the team's performance in a game. We create a feature 'form' that uses an exponential decay weight factor.

$$Form(t) = \sum_{i = t-n}^{t} gd(i) * e^{-\lambda(t - i)}$$

Here n is the window size. For i^{th} game, using goal difference we calculate the weight of a particular game.

However later we realize this does not do much good to our model and comprises mostly zeros. So eventually we will do away with this feature.

We will also need to Scale the features so that each feature can have nearly the same impact on the accuracy result. We use StandardScaler which will transform the data such that its distribution has a mean 0 and standard deviation of 1. In case of this multivariate data, this is done feature-wise. Given the distribution of the data, each value in the dataset will have the mean value subtracted, and then divided by the standard deviation of the whole dataset.

Feature Importance

We used a Random Forest Classifier for a preliminary prediction. Using the `feature_importances_` attribute of `RandomForestClassifier` from `sklearn`, we estimated the relevance of our features to the predicted outcome. Unsurprisingly, `Shots_on_target_*` ranked high in importance, while features like `Red_cards_*`, while capable of severely impacting a match, were found to be unimportant by our classifier, because of their rarity.

```
 1) Shots_on_target_home           0.073068

 2) Clearances_away                0.064972

 3) Shots_on_target_away           0.058568

 4) Clearances_home                0.050302

 5) Expenditures_away              0.039533

 6) Expenditures_home              0.038386

 7) Passes_away                    0.036948

 8) Income_away                    0.036030

 9) Touches_away                   0.035701

10) Passes_home                    0.035578

11) Income_home                    0.034710

12) Touches_home                   0.032749

13) Shots_home                     0.032306

14) Shots_away                     0.031355

15) Possession_away                0.029745

16) Possession_home                0.029692
```

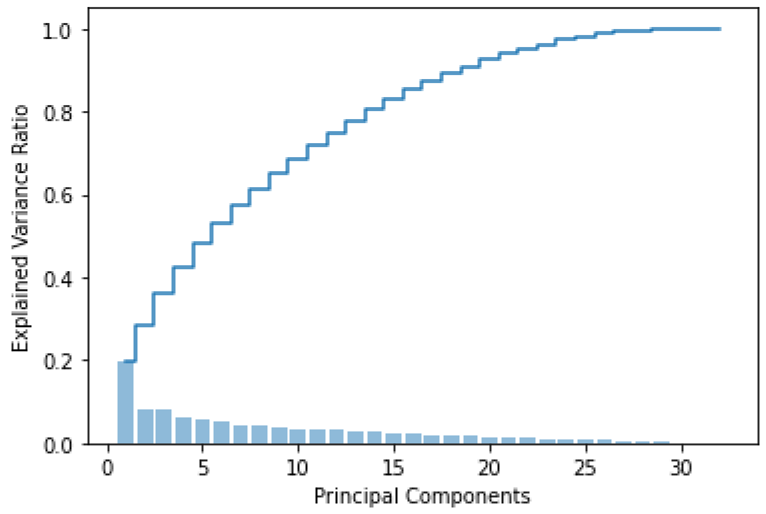
```
17) Tackles_away                   0.028653

18) Tackles_home                   0.028334

19) Arrivals_away                  0.027865

20) Arrivals_home                  0.027565

21) Departures_home                0.027099

22) Departures_away                0.026871

23) Fouls_conceded_away            0.025810

24) Fouls_conceded_home            0.025579

25) Corners_home                   0.022296

26) Corners_away                   0.021941

27) Offsides_home                  0.019105

28) Offsides_away                  0.018365

29) Yellow_cards_away              0.016599

30) Yellow_cards_home              0.016296

31) Red_cards_home                 0.005184

32) Red_cards_away                 0.002795
```

Because of less importance of some features, we must delve deeper into feature selection techniques. For this, we will use Principal Component Analysis.

Principal Component Analysis

Principal component analysis is a process of computing the change of basis on the data by computing the principal component. Principal components of a collection of points in a real p-space are a sequence of p directional vectors, where the ith vector is the direction of a line that best fits the data while being orthogonal to the rest i-1 vectors.

After fitting our pca with X_train, we study the explained variance ratio vs principal components used.



We see that with around 10 components, we see around 60% of the explained variance, with around 15 components around 80% of explained variance, and with 25 components around 90% of explained variance.

Thus we can reduce our dimensions to 15 components with 80% explanation variance if the accuracy is better than using full components or does not fall too much.

Using Principal Component Analysis, we simplify our model and test it on various classification algorithms like LogisticRegression, KNeighborsClassifier, GradientBoostingClassifier, Support Vector Machines and MultiLayered Perceptron Classifier (MLP), keeping the 'target' variable as the variable to be classified.

Accuracy (Score) with Different number of Components

	Dimension: Original	Dimension: 15 components	Dimension: 20 components	Dimension: 25 components
Logistic Regression	0.6261	0.6030	0.6052	0.6129
Random Forest	0.625	0.6008	0.6063	0.6151
Gradient Boosting	0.6316	0.6041	0.5997	0.6052
SVMs	0.5241	0.6162	0.6140	0.6184
Neural Network	0.2861	0.5263	0.5175	0.5219

So reducing the dimension by PCA does not do much good to our model.

We observe best performance by using Gradient Boosting on original dimensions and with reduced dimensions, SVMs perform best with 15 components.

Testing The Model

We reduce the dimensions of our model using the feature importance and Pearson’s correlation matrix. The reduced dimension will have 21 dimensions, and with 4 label features removed we get 17 features.

Testing our model using Support Vector Machines (SVMs) with decision function ‘ovo’, we get a 61.29% accuracy score.

Using Ensemble Model

Ensemble methods are learning models that achieve performance by combining the opinions of multiple learners. We used a hard VotingClassifier.

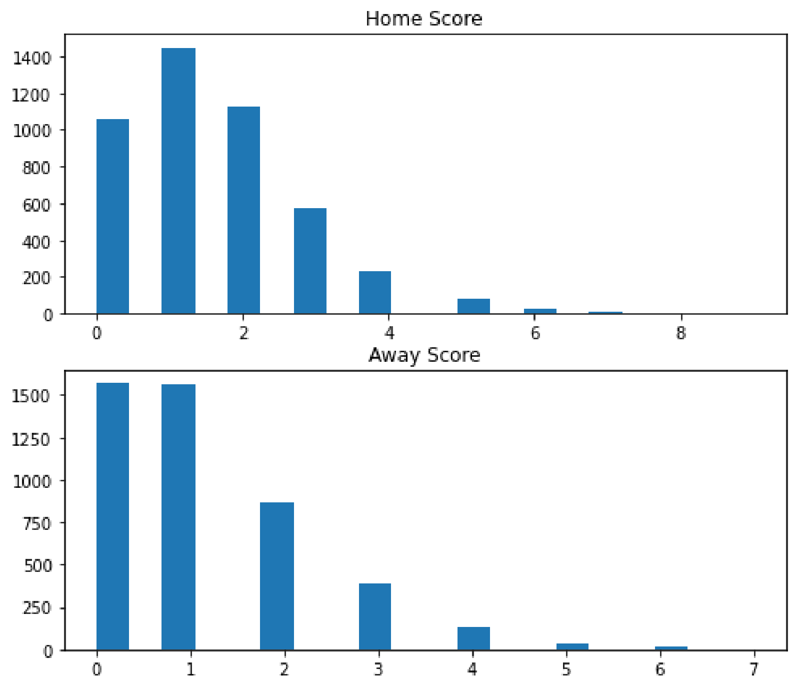
Using Ensemble model on LogisticRegression, Random Forest Classifier, Gradient Boosting Classifier and SVM, we get an improved accuracy of 64 +/- 0.3%.

Ensemble model doesn't improve the result much compared to individual classifiers.

Expected Goals

So far we have used multilabel classification on the target variable to predict home team victory, away team victory or a draw, based on in-game and season long statistics which can have significant impact on games. But as mentioned earlier, there are many times inconsistency involved with a team's performance and the number of goals scored and conceded. Thus we create an expected Goals based on the features and predict victory to the side which scores more. We build a model to generate synthetic match statistics, which can be used to predict the expected number of goals scored.

We can consider both transfer market spending and income, along with the game-level statistics. Using this as a function, we can make our predictions.



This goal scored distribution looks like an exponential or Poisson distribution. This is intuitive in a sense that it is hard to score more goals and their occurrence depends on a rather very short passage of play.

Season Simulation

We attempt to simulate the final rankings of the teams in the 2018 season of the EPL. For this we use individual team performance to calculate expected goals to declare the winner of a particular game, and thus the expected number of points scored by the team at the end of the season.

For this, we try to minimise the RMSE (Root Mean Squared Error) of the predicted vs. real scores (goals).

Selecting The Best Model

First we need to know the best regression model that can predict the score variable using features. We try several regressors to find the ones giving the least RMSE. Keeping 'score' as the target, we see that **GradientBoostingRegressor** works best followed by **Ridge** and **LinearRegression**.

```
('GradientBoostingRegressor', 1.0476679808835914),

('Ridge', 1.0483360868187832),

('LinearRegression', 1.0483361079788438),

('BayesianRidge', 1.0483445635459643),

('MLPRegressor', 1.0559851992947198),

('AdaBoostRegressor', 1.0720350150665876),

('RandomForestRegressor', 1.0796276861756529),

('ExtraTreesRegressor', 1.0943520790161436),

('Lasso', 1.1534239002762374),

('SVR', 1.2123354715319923),

('LassoLars', 1.3101170191594549),
('KNeighborsClassifier', 1.565887031105823),

('SGDRegressor', 20900617051055.145)
```

Hyperparameter Tuning

In our preliminary tests, GradientBoostingRegressor was for to work best for score prediction. Thus we use cross validation techniques to tune our hyperparameters. Using `RandomizedSearchCV` with GradientBoostingRegressor, we obtain the least ls loss (least square) for the following parameter set.

```
{'subsample': 0.2, 'min_samples_split': 2, 'max_depth': 10, 'loss': 'ls'}
: 0.14525918395391843
```

Prediction

The score values (expected number of goals) were predicted using gradient boosting regressor with the hyperparameters obtained above, using the following features: Shots, Passes, Clearances, Fouls and Offsides. We assign some regression models to each of these features which will work well as shown afterwards. The model is trained on the dataset of the premier league games we have.

Using a window of 10 games in the past, we synthesize the selected features using the following Regression Models which we found to perform best. When we have these stats, we can use our game_model (GradientBoostingRegressor in our case) to predict the expected goals for the game in study.

```
'Shots': GradientBoostingRegressor(),

'Passes': LinearRegression(),

 'Clearances': GradientBoostingRegressor(),
 'Fouls': Ridge(),

 'Offsides': GradientBoostingRegressor()
```

The data thus obtained, describes a team's predicted performance in individual matches for the season 2018. Using the MatchID feature, we obtain the relative performance of the two teams in a particular match, by comparing MatchID's for equality between two data points. The comparison of predicted scores (goals) of the two teams gives us the winner of the match.

As we know, in EPL a team is awarded 3 points for a win, 1 for a draw, and 0 for a loss. Using the predicted match results, we calculate the total number of points of all the teams at the end of the season. This, however, doesn't give us very accurate rankings.

For a more reliable prediction, we run our model several (50) times and use the expected value (mean) of the team points to calculate the final standing. As shown in the following section, we obtain very significant results, in spite of the highly random nature of the game.

Results

Using expected goal metric and doing multiple numbers of runs on the League table simulation model and taking the average of the points, we get the final standing which resembles the actual Premier League Table for that season.

Predicted Season 2017/18

Position	Team	Points
1	Chelsea	75
2	Spurs	68
3	Man City	66
4	Liverpool	60
5	Arsenal	60
6	Man Utd	57
7	Watford	56
8	Southampton	55
9	Everton	52
10	Leicester	52
11	Swansea	52
12	Stoke	49
13	Newcastle	42
14	Burnley	41
15	West Ham	41
16	West Brom	40
17	Brighton	39
18	Huddersfield	38
19	Crystal Palace	37
20	Bournemouth	36

Actual Season 2017/18

Position	Team	Points
1	Man City	100
2	Man Utd	81
3	Spurs	77
4	Liverpool	75
5	Chelsea	70
6	Arsenal	63
7	Burnley	54
8	Everton	49
9	Leicester	47
10	Newcastle	44
11	Crystal Palace	44
12	Bournemouth	44
13	West Ham	42
14	Watford	41
15	Brighton	40
16	Huddersfield	37
17	Southampton	36
18	Swansea	33
19	Stoke	33
20	West Brom	31

From the standing we see that All top 6 teams which qualify for European League are predicted correctly. 3 out 4 teams are correctly predicted to be in the top 4.

In general, our model looks like a standard finish to English Premier League.

The regression model which worked best and was used for predicting expected Goals was **Gradient Boosting Regressor** with 'ls' cost function.

In the first part of our project, we did a classification modelling using 'target' feature to classify into win, loss or draw.

The Best model with full dimensions was observed to **Gradient Boosting** with **63.15%** accuracy score. Using **Ensemble model**, we observed an accuracy score of **(64 +/- 0.2)%.**

After reducing the dimensions, we observed that **Logistic Regression** did best and achieved **62.82%** accuracy score. Using **Ensemble** model, we observed an accuracy score of **(64 +/- 0.3)%**.

Reducing dimensions did not have a great impact on the model's performance.

Conclusion

Our main objective to build an expected goals model using various important features and exploring different Machine Learning techniques was accomplished. We used Machine Learning Models such as Logistic Regression, Support Vector Machines, Gradient Boosting to generate match outcome and score predictions.

We did feature reduction using Principal Component Analysis and found out that a model with somewhere close to 15 components works good. By feature engineering, we created a form feature which took into account the last five games with exponentially decreasing importance on goal difference. We found out that this model did not contribute to our model’s accuracy.

To select the best model to predict expected Goals, we ran a regression analysis using various regression models on our last ten years dataset, keeping the score feature as target. We found that Gradient Boosting Regressor worked best in this case. We synthesize the selected feature used to predict expected goals, using a window of past 10 games and adding randomness. Few caveats to it were that newly promoted teams took the statistics of last season’s relegated team and for the first ten games, we used last season’s data.

We found that the expected Goals model was successful in predicting the 2017/18 season to some extent.

References

[1] Premier League official website
[https://www.premierleague.com/results?co=1&se=23&cl=-1]

[2] Predicting Football Results Using Machine Learning Techniques
https://www.imperial.ac.uk/media/imperial-college/faculty-of-engineering/computing/public/1718-ug-projects/Corentin-Herbinet-Using-Machine-Learning-techniques-to-predict-the-outcome-of-profressional-football-matches.pdf

[3] Principal Component Analysis
[https://en.wikipedia.org/wiki/Principal_component_analysis]

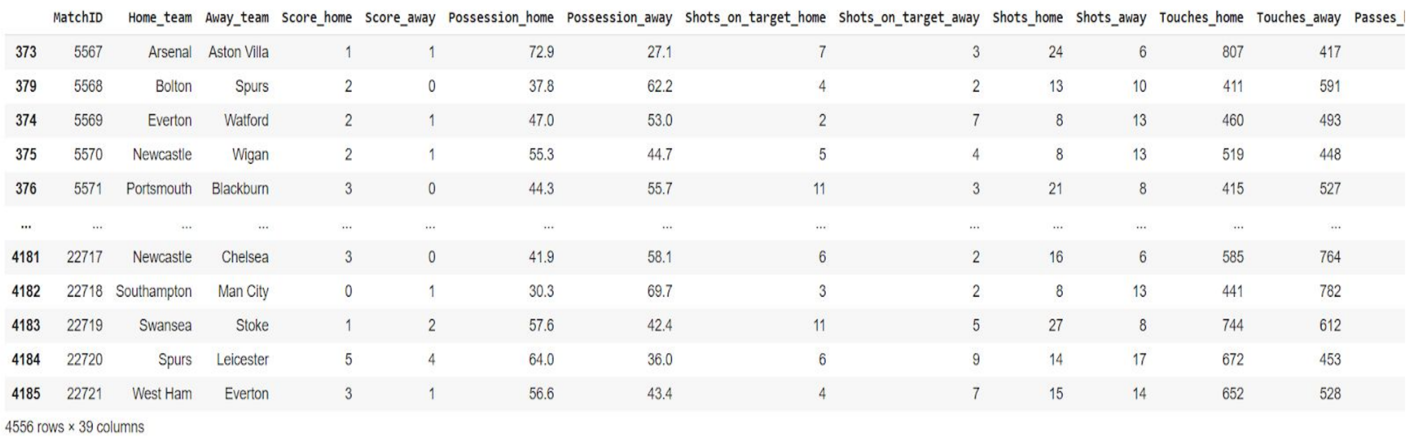
[4] Sourabh Swain & Shriya Mishra - Data Science Approach to Predict the Outcome of a Football Match
[https://www.ijcseonline.org/pdf_paper_view.php?paper_id=1857&20-IJCSE-02970.pdf]

[5] Ben Ulmer & Matthew Fernandez - Predicting Soccer Match Results in the English Premier League
[http://cs229.stanford.edu/proj2014/Ben%20Ulmer,%20Matt%20Fernandez,%20Predicting%20Soccer%20Results%20in%20the%20English%20Premier%20League.pdf]

[6] Expected Goals Model: xG stats explained
[https://www.bundesliga.com/en/bundesliga/news/expected-goals-xg-model-what-is-it-and-why-is-it-useful-sportec-solutions-3177]

Appendix

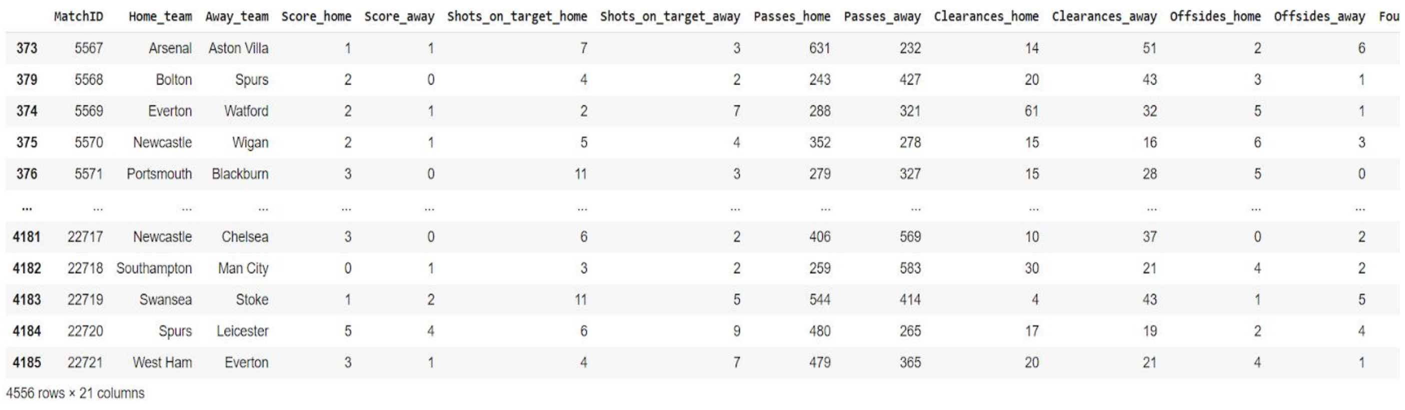
1. Initial dataframe with 39 features



	MatchID	Home_team	Away_team	Score_home	Score_away	Possession_home	Possession_away	Shots_on_target_home	Shots_on_target_away	Shots_home	Shots_away	Touches_home	Touches_away	Passes_
373	5567	Arsenal	Aston Villa	1	1	72.9	27.1	7	3	24	6	807	417	
379	5568	Bolton	Spurs	2	0	37.8	62.2	4	2	13	10	411	591	
374	5569	Everton	Watford	2	1	47.0	53.0	2	7	8	13	460	493	
375	5570	Newcastle	Wigan	2	1	55.3	44.7	5	4	8	13	519	448	
376	5571	Portsmouth	Blackburn	3	0	44.3	55.7	11	3	21	8	415	527	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4181	22717	Newcastle	Chelsea	3	0	41.9	58.1	6	2	16	6	585	764	
4182	22718	Southampton	Man City	0	1	30.3	69.7	3	2	8	13	441	782	
4183	22719	Swansea	Stoke	1	2	57.6	42.4	11	5	27	8	744	612	
4184	22720	Spurs	Leicester	5	4	64.0	36.0	6	9	14	17	672	453	
4185	22721	West Ham	Everton	3	1	56.6	43.4	4	7	15	14	652	528	

4556 rows × 39 columns

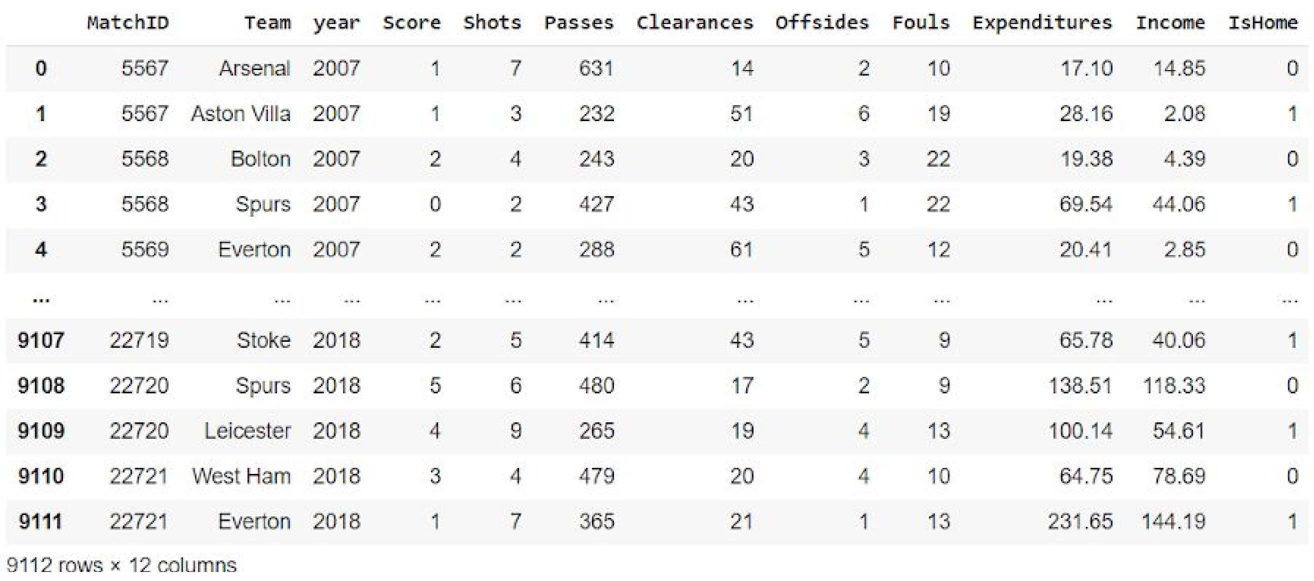
2. Modified dataframe with 21 features



	MatchID	Home_team	Away_team	Score_home	Score_away	Shots_on_target_home	Shots_on_target_away	Passes_home	Passes_away	Clearances_home	Clearances_away	Offsides_home	Offsides_away	Fou
373	5567	Arsenal	Aston Villa	1	1	7	3	631	232	14	51	2	6	
379	5568	Bolton	Spurs	2	0	4	2	243	427	20	43	3	1	
374	5569	Everton	Watford	2	1	2	7	288	321	61	32	5	1	
375	5570	Newcastle	Wigan	2	1	5	4	352	278	15	16	6	3	
376	5571	Portsmouth	Blackburn	3	0	11	3	279	327	15	28	5	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4181	22717	Newcastle	Chelsea	3	0	6	2	406	569	10	37	0	2	
4182	22718	Southampton	Man City	0	1	3	2	259	583	30	21	4	2	
4183	22719	Swansea	Stoke	1	2	11	5	544	414	4	43	1	5	
4184	22720	Spurs	Leicester	5	4	6	9	480	265	17	19	2	4	
4185	22721	West Ham	Everton	3	1	4	7	479	365	20	21	4	1	

4556 rows × 21 columns

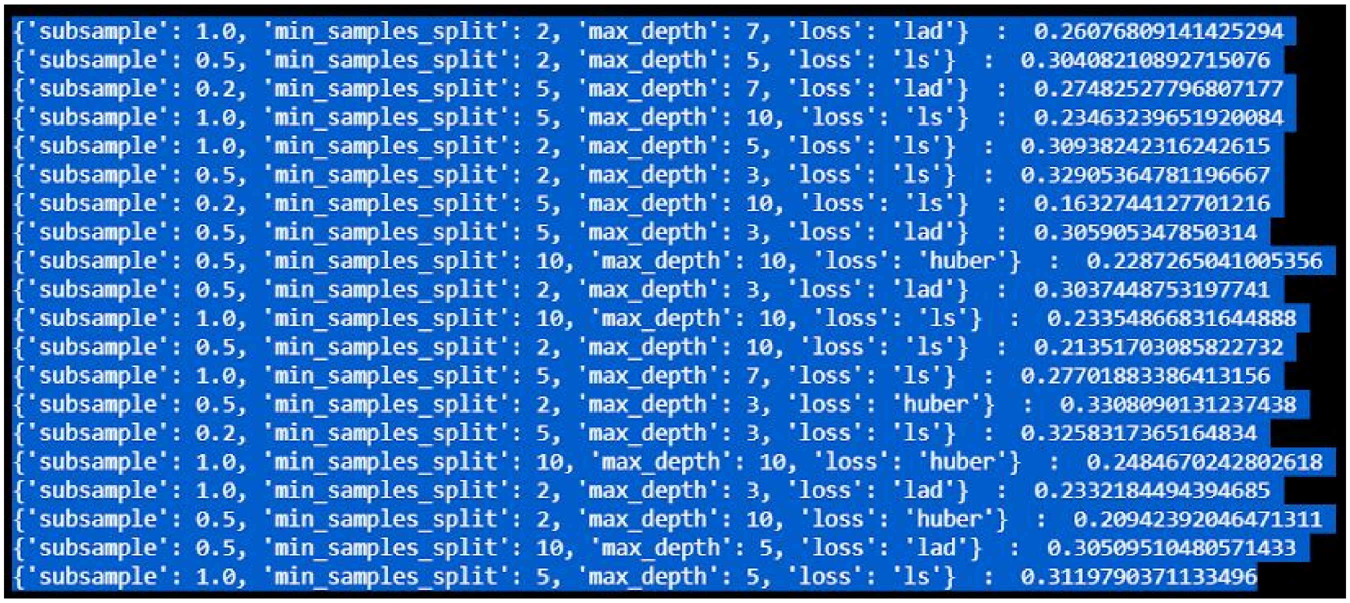
3. Dataframe with each individual row consisting of one team's result.



	MatchID	Team	year	Score	Shots	Passes	Clearances	Offsides	Fouls	Expenditures	Income	IsHome
0	5567	Arsenal	2007	1	7	631	14	2	10	17.10	14.85	0
1	5567	Aston Villa	2007	1	3	232	51	6	19	28.16	2.08	1
2	5568	Bolton	2007	2	4	243	20	3	22	19.38	4.39	0
3	5568	Spurs	2007	0	2	427	43	1	22	69.54	44.06	1
4	5569	Everton	2007	2	2	288	61	5	12	20.41	2.85	0
...	...	...	...	...	...	...	...	...	...	...	...	...
9107	22719	Stoke	2018	2	5	414	43	5	9	65.78	40.06	1
9108	22720	Spurs	2018	5	6	480	17	2	9	138.51	118.33	0
9109	22720	Leicester	2018	4	9	265	19	4	13	100.14	54.61	1
9110	22721	West Ham	2018	3	4	479	20	4	10	64.75	78.69	0
9111	22721	Everton	2018	1	7	365	21	1	13	231.65	144.19	1

9112 rows × 12 columns

4. Result of RandomizedSearchCV on Gradient Boosting Regressor



```
{'subsample': 1.0, 'min_samples_split': 2, 'max_depth': 7, 'loss': 'lad'}  :  0.26076809141425294
{'subsample': 0.5, 'min_samples_split': 2, 'max_depth': 5, 'loss': 'ls'}  :  0.30408210892715076
{'subsample': 0.2, 'min_samples_split': 5, 'max_depth': 7, 'loss': 'lad'}  :  0.27482527796807177
{'subsample': 1.0, 'min_samples_split': 5, 'max_depth': 10, 'loss': 'ls'}  :  0.23463239651920084
{'subsample': 1.0, 'min_samples_split': 2, 'max_depth': 5, 'loss': 'ls'}  :  0.30938242316242615
{'subsample': 0.5, 'min_samples_split': 2, 'max_depth': 3, 'loss': 'ls'}  :  0.32905364781196667
{'subsample': 0.2, 'min_samples_split': 5, 'max_depth': 10, 'loss': 'ls'}  :  0.1632744127701216
{'subsample': 0.5, 'min_samples_split': 5, 'max_depth': 3, 'loss': 'lad'}  :  0.305905347850314
{'subsample': 0.5, 'min_samples_split': 10, 'max_depth': 10, 'loss': 'huber'}  :  0.2287265041005356
{'subsample': 0.5, 'min_samples_split': 2, 'max_depth': 3, 'loss': 'lad'}  :  0.3037448753197741
{'subsample': 1.0, 'min_samples_split': 10, 'max_depth': 10, 'loss': 'ls'}  :  0.23354866831644888
{'subsample': 0.5, 'min_samples_split': 2, 'max_depth': 10, 'loss': 'ls'}  :  0.21351703085822732
{'subsample': 1.0, 'min_samples_split': 5, 'max_depth': 7, 'loss': 'ls'}  :  0.27701883386413156
{'subsample': 0.5, 'min_samples_split': 2, 'max_depth': 3, 'loss': 'huber'}  :  0.3308090131237438
{'subsample': 0.2, 'min_samples_split': 5, 'max_depth': 3, 'loss': 'ls'}  :  0.3258317365164834
{'subsample': 1.0, 'min_samples_split': 10, 'max_depth': 10, 'loss': 'huber'}  :  0.2484670242802618
{'subsample': 1.0, 'min_samples_split': 2, 'max_depth': 3, 'loss': 'lad'}  :  0.2332184494394685
{'subsample': 0.5, 'min_samples_split': 2, 'max_depth': 10, 'loss': 'huber'}  :  0.20942392046471311
{'subsample': 0.5, 'min_samples_split': 10, 'max_depth': 5, 'loss': 'lad'}  :  0.30509510480571433
{'subsample': 1.0, 'min_samples_split': 5, 'max_depth': 5, 'loss': 'ls'}  :  0.3119790371133496
```