Analysis on Football Games and Teams using Machine Learning Techniques

Team: ML-Drivers

Authors:

Akanksha Rawat akanksha.rawat@sjsu.edu

Karishma Kuria karishma.kuria@sjsu.edu

Nisha Mohan Devadiga nishamohan.devadiga@sjsu.edu

Abstract

Football is a very popular game worldwide. It was invented in England in the nineteenth century and is now played by more than 240 million people according to International Federation of Association Football.

Also known as soccer in some countries, the motivation for this project is inspired from an opportunity to work with a large database of football data. This data was provided by BBC , ESPN, and several other sport new channel through web scraping.

Application of data mining techniques in football is not something new. However, they are not exploited that as much to reach its full potential. The football analytics is something which is developed and used frequently in western countries to maximize the potential of football team and their players. Identifying the patterns in their gaming style can prove to be a vital information about the team and can help to make difference in winning and losing.

Introduction

We explore the data mining process wherein we apply several Data Mining techniques to the chosen datasets to identify and discover existing patterns.

In this paper, unsupervised learning - Clustering is used to identify the groups of soccer team that are similar to each other with respect to their gaming styles.

We further apply Fractal Clustering to identify best and worst performing teams based on their games - Red Card Vs Goal using Clustering and identify the "GOLDEN CLUSTER".

Objective Functions:

1. Identify and group Soccer Teams based on their Goals and Red Cards received throughout their games.
2. Identify Top 3 Best Performing Soccer teams based on Goals scored and accumulated Red Cards.

3. Identify Worst Performing Soccer teams based on Goals scored and accumulated Red Cards.

Also, Regression and Classification technique would be applied to model a solution to predict whether the Home team or Away Team will win the match.

We also look into the questions like below using the best classifier to compare through the performance matrix and make predictions:-

- Which players are the best finishers?
- Analyzing Goals Scored?
- Analyzing Substitutions?
- Analyzing Yellow/Red Cards?
- Analyzing Penalties?
- Which players have the most "expected goals"?
- Which players are the worst at deciding their shots?
- Which players make the best/most dangerous passes?

With the Soccer team information from new scraped dataset spanning from 2009 to 2019, classification and regression algorithms are implemented to predict which team (Home Team / Away Team) can win the match.

Related Work

An existing paper[1] on applying data mining techniques to Football data from European Championship presents development of decision support system to be used in the match or selection of referee or game location for each match. However the data mining modules were developed to provide adaptive agent behavior in dynamically changing environments using automata data. In our case, the primary focus is on the gaming style of each football team.

Data

The dataset is a result of a very tiresome effort of web-scraping and integrating different data sources. The central element is the text commentary. All the events were derived by reverse engineering the text commentary, using regex.It includes events from more than 7,000 games from the top 5 European Leagues from 2011 to 2016.

The dataset provides a granular view of 9,074 games, totaling 941,009 events from the biggest 5 European football (soccer) leagues: England, Spain, Germany, Italy, France from 2011/2012 season to 2016/2017 season as of 25.01.2017. The dataset is organized in 3 files:

1. events.csv contains event data about each game. Text commentary was scraped from: bbc.com, espn.com and onefootball.com
2. ginf.csv contains metadata and market odds about each game. odds were collected from oddsportal.com
3. dictionary.txt contains a dictionary with the textual description of each categorical variable coded with integers.

Two main datasets were used. The first dataset was chosen because it has the highest level of detail and the lowest levels of missing values and erroneous data. The second dataset includes all matches played in six European countries, including Portugal, for 5 years spanning from 2012 to 2017.

Scraped Dataset : The game information spanning from 2009 - 2019 from datahub.io is also scraped for the better analysis on clustering and regression problem .

Exploratory Data Analysis

The features from event dataset are pre-processed to include problem statement specific features to attain the project objective.

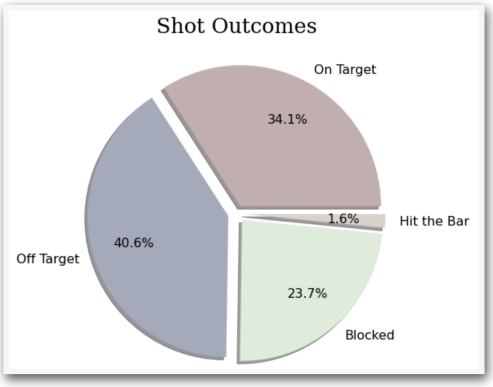


Fig 1

In the pie chart (fig 1), on analyzing the shot outcome results, it can be observed that most of the shots are off-target, but the difference with on-target shows is marginal. Only a fraction of those on-target shots end up as goals, since goalkeeper tries to prevent goals on the other side.

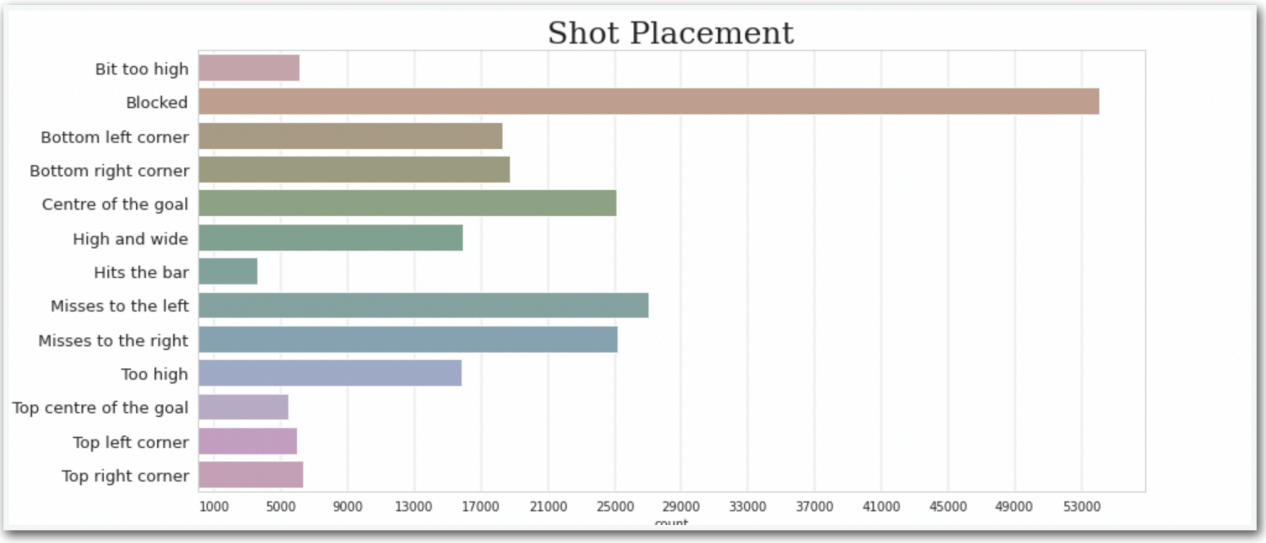


Fig 2

Looking at shot placements in fig 2, it can be noted that most of the shots are blocked.

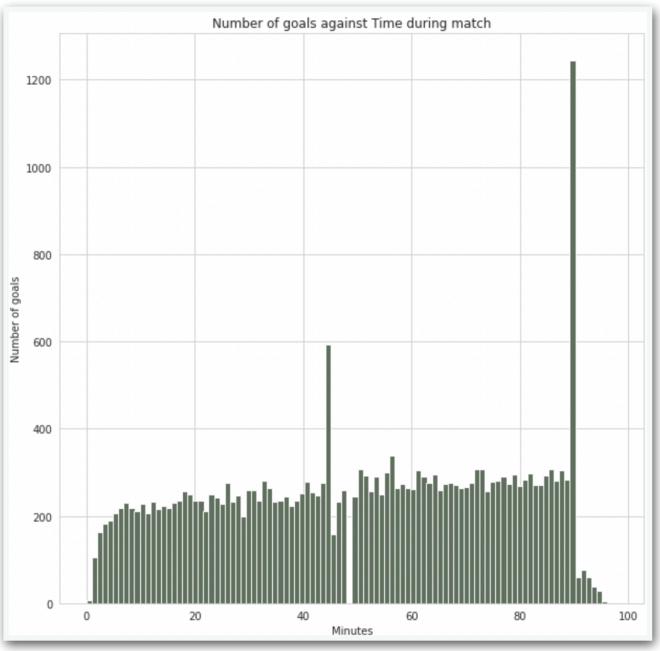


Fig 3

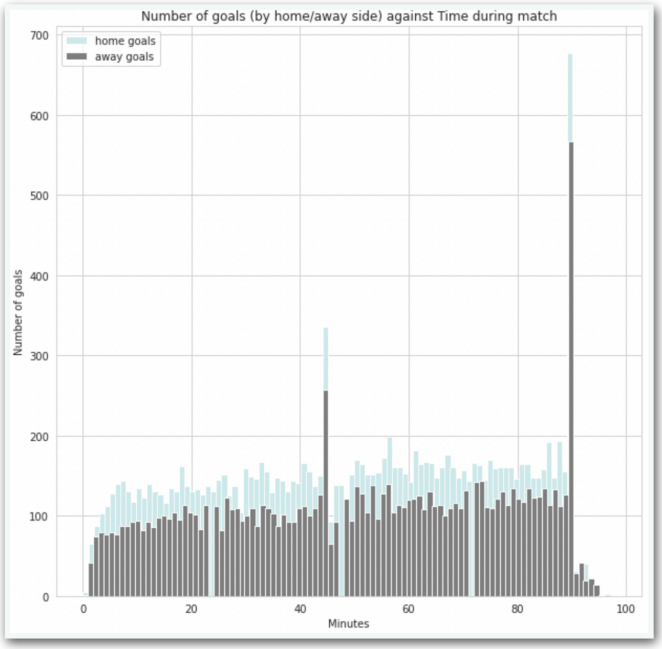


Fig 4

For number of goals against time in a match, it is found that most goals are scored around the Half-Time (45mins + extra time) and around Full-Time (90mins + extra time) (Fig 3).For every minute, most of the goals scored are by the home side.This supports the general notion that the home side has a statistical advantage. (Fig 4).

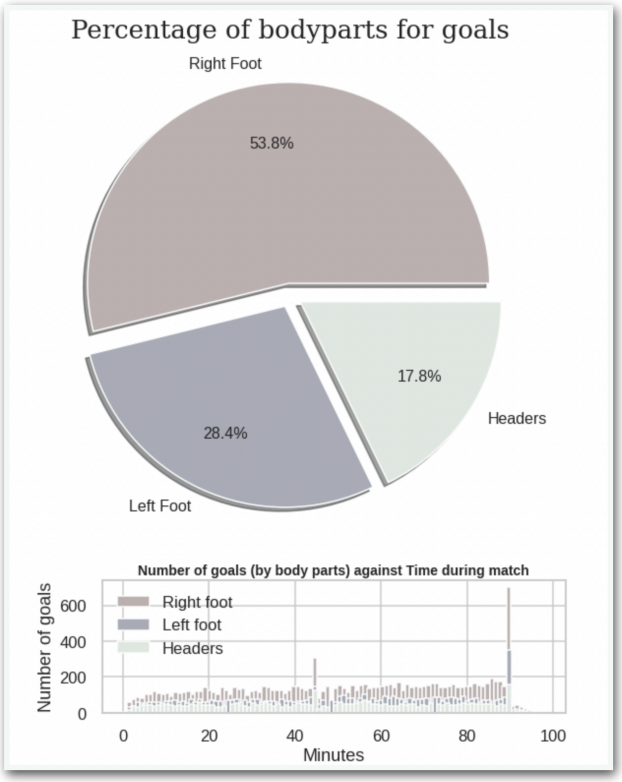


Fig 5

Fig 5.Most of the goals scored are by Right Footed, then followed by Left Footed and lastly, by Headers. Perhaps, this might be because the majority of humans are right-footed and, hence, most players are right-footed.

It is also not surprising that most goals have been scored by foot not head, as after all, soccer is meant to be played by foot.

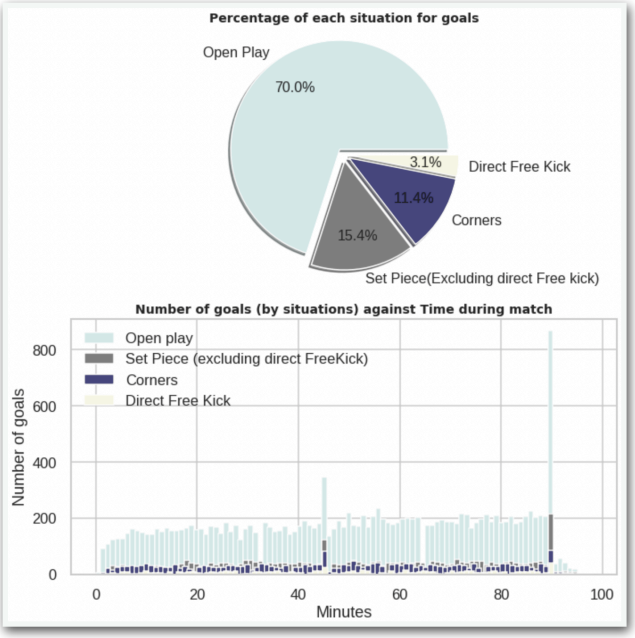


Fig 6

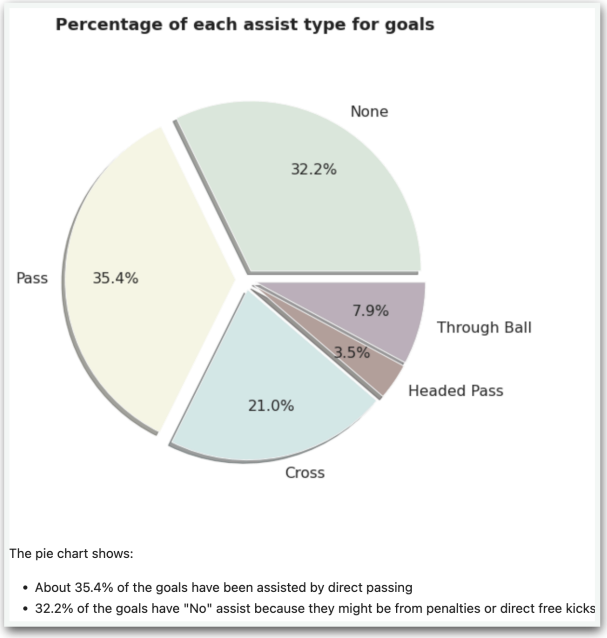


Fig 7

Fig 6 , Nearly 70.8% of the goals scored are from Open Play.

The pie chart in Fig 7 shows:

- About 35.4% of the goals have been assisted by direct passing
- 32.2% of the goals have "No" assist because they might be from penalties or direct free kicks.

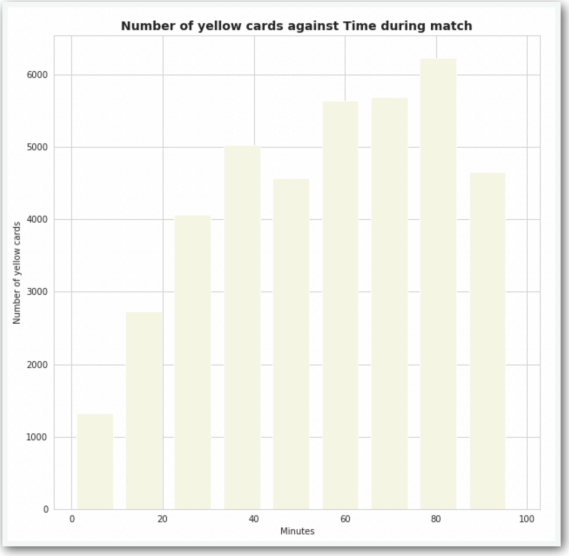


Fig 8

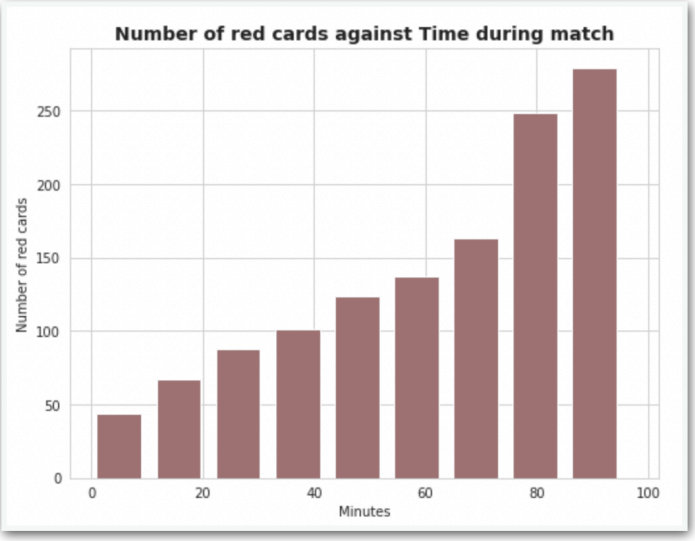


Fig 9

Data Pre-processing

Data Cleaning:

In the data cleaning step, the dataset for analysis is prepped by removing or modifying the data is incorrect, incomplete, irrelevant, duplicated or improperly formatted.Checking for missing or null values, converting the categorical value to numeric is performed. Additional checking for if data is balanced or not (pair plot) or whether there is a need to whiten out the data (remove different weights in the data)

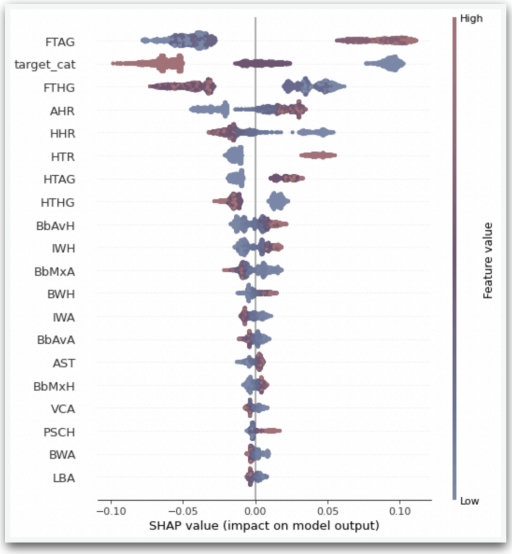
Feature Selection:

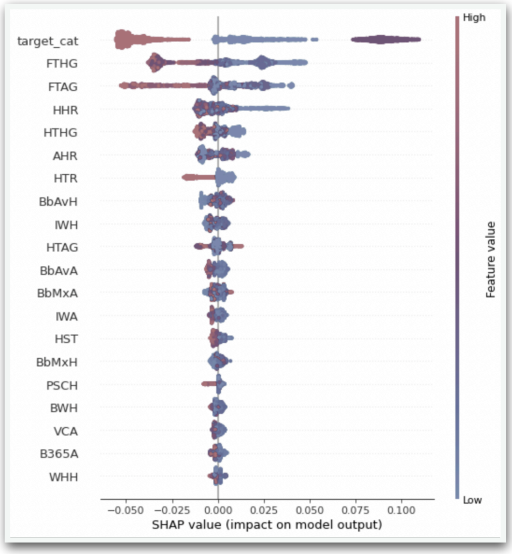
In lines to problem statement, we have focused on two important features - Red Cards and Goals for the clustering problem statement i.e. applying Fractal Clustering to identify best and worst football teams on these features.

In the first step of data pre-processing, event dataset is filtered on shot outcome as ‘on target’ to get all rows associated with goals and then, grouped on teams. The counts of yellow cards, red cards, fouls and goals each team during the match are calculated and used for further processing. Using Dictionary dataset, the features like ‘situation’,'location','assist_method' are renamed with more user friendly names.

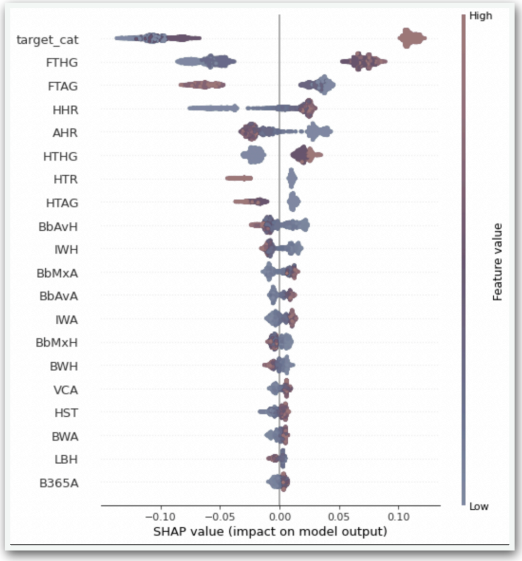
For classification problem, scraped dataset provides better accuracy and finer details for matches spanning from 2009 to 2019.

We use SHAP technique for Feature Selection by importing summary plot and TreeExplainer from shap libraries. Run one of the classifiers and input the fitted model to TreeExplainer to output Shap values. Proceed to summary_plot that will show feature rankings based on SHAP values on a per class basis. For class 3 this will be





We can see the top Features on the top of the graph.



Introduction to Latent Variables:

Latent variables are introduced when it is determined that a variable is not directly observable and is assumed to affect the response variables. Based on the selected football domain following three latent variables are identified.

1. Latent variable #1 : The Total Shots Ratio (TSR) is used to determine how well teams fare in a match when it comes to taking and conceding shots.

   The TSR is determined by the following formula:

   TSR = Total Shots for / (Total Shots for + Total Shots against)

2. Latent Variable# 2 : Home Hit Rate is used to determines the rate at which home teams serves a goal against shots played.

   The HHR is determined by the following formula:

   Home Hit Rate = Full Time Home Goal / Home Shots

3. Latent Variable# 3 : Away Hit Rate is used to determine the rate at which away teams serves a goal against shots played.

   The AWR is determined by the following formula:

   Away Hit Rate = Full Time Away Goal / Away Shots

Feature Analysis

Features Used :

Although there are many features in the game information dataset. We will purely consider the features related to football game and discard the information related to various betting scores for machine learning. Therefore, following features are included - Full Time Home Goal, Full Time Away Goal, Half time Away Goal , Half Time Home Goal , Full Time Result , Half Time Result, Total Shot Ratio, Home Hit Rate and Away Hit Rate

- *Univariate Analysis :*

We use the simplest form of analysis - univariate analysis to see out information of one one feature at a time. Using univariate analysis, we can observe that features such as :- Full time Home Goal, Full TIme Away Goal, Half time Away Goal , Half Time Home Goal , Total shot ratio, Home Hit Rate and Away Hit Rate are gaussian distributed. Hence MinMax scaling is required.

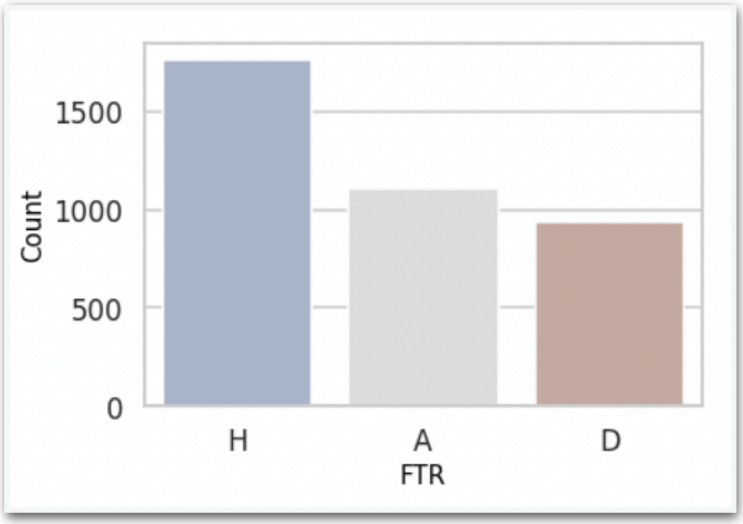
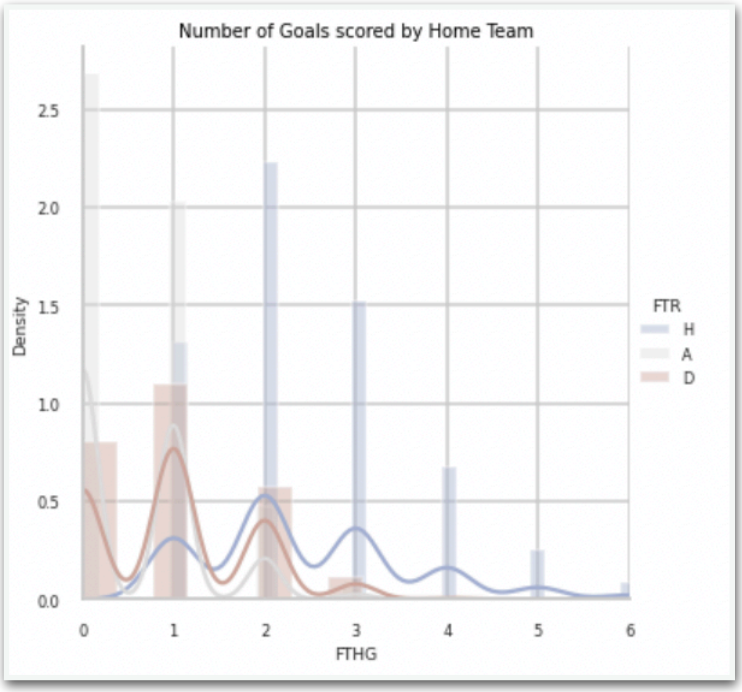
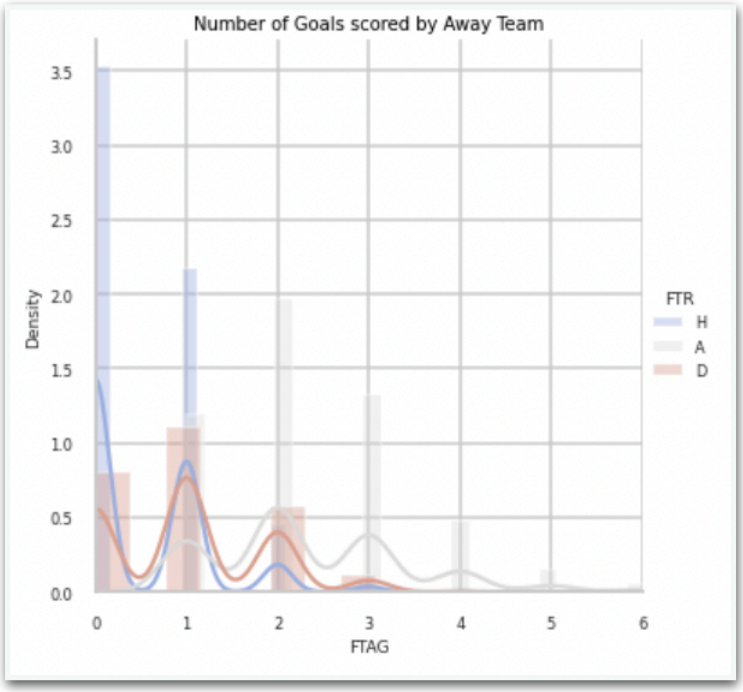


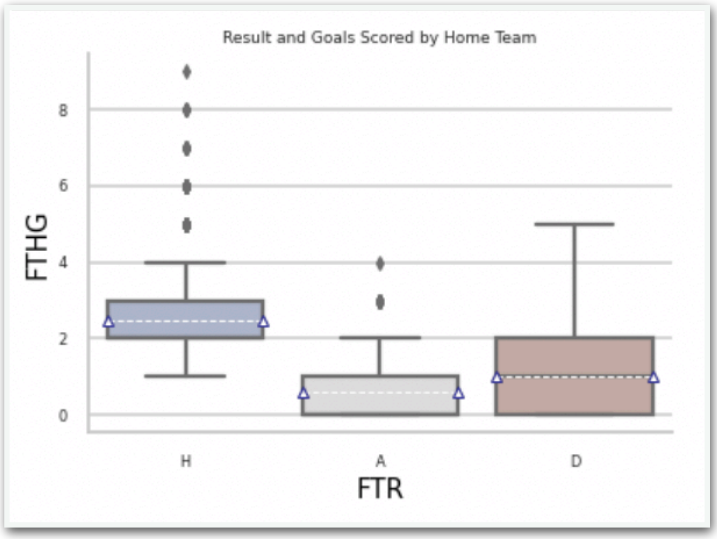
Fig 10

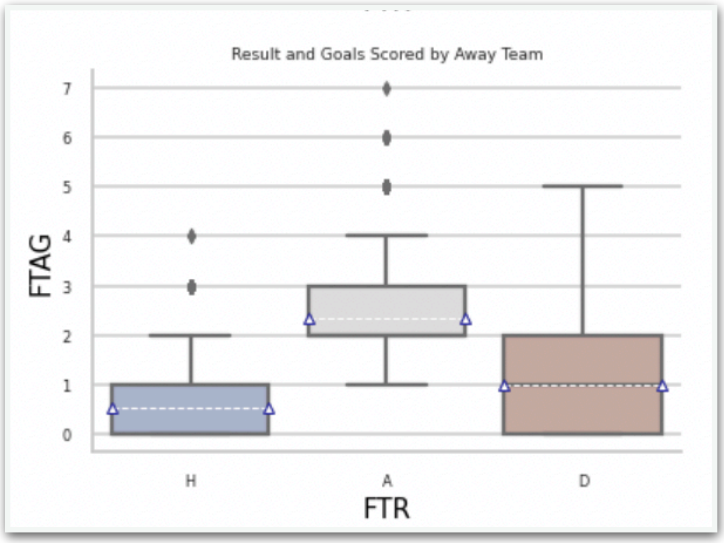
Just By looking at the Counts we can get the Understanding that the Home team has Significantly more wins. Another way of looking at is Away Team is more likely to get a Draw or a loss more often. Looking at this it looks like Playing at home is a big advantage.





Most of the times, Both Home and Away Team scores 1 goal, Frequent scores are (1,0,2 in this sequence).Away team slightly ahead here. However, When it come to more than 2 goals, Home Teams are ahead.





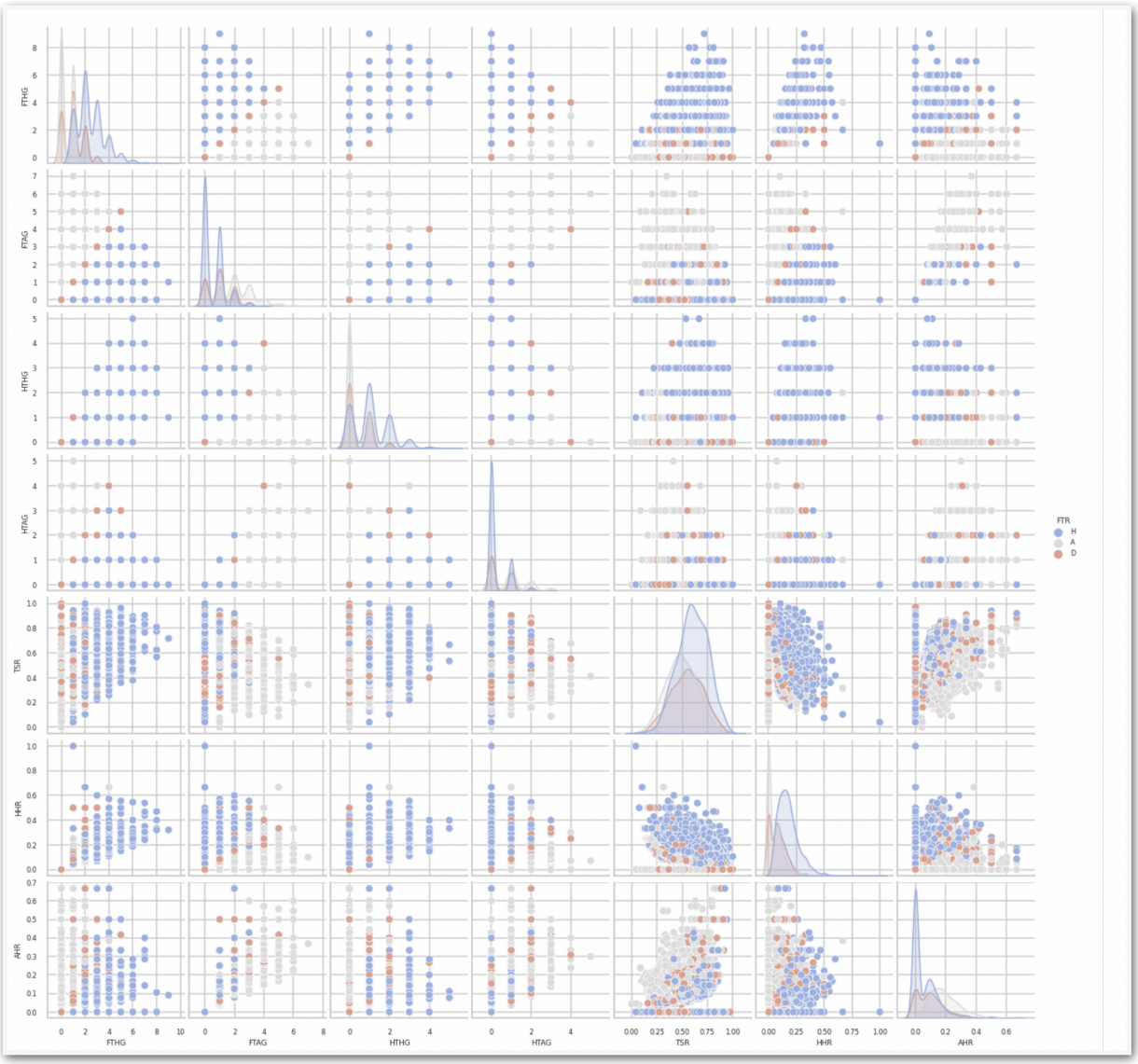
Home Team :

1. While winning score mean of 2.5 Goals.
2. When Drawn mean is 0.9 Goals
3. When Lost Mean is 0.5 Goals.

Away Team :

1. While winning score mean of 2.1 Goals.
2. When Drawn mean is 0.9 Goals.
3. When Lost Mean is 0.6 Goals -> Overall Home Team is Scoring more goals, While will be a huge factor in winning the game.

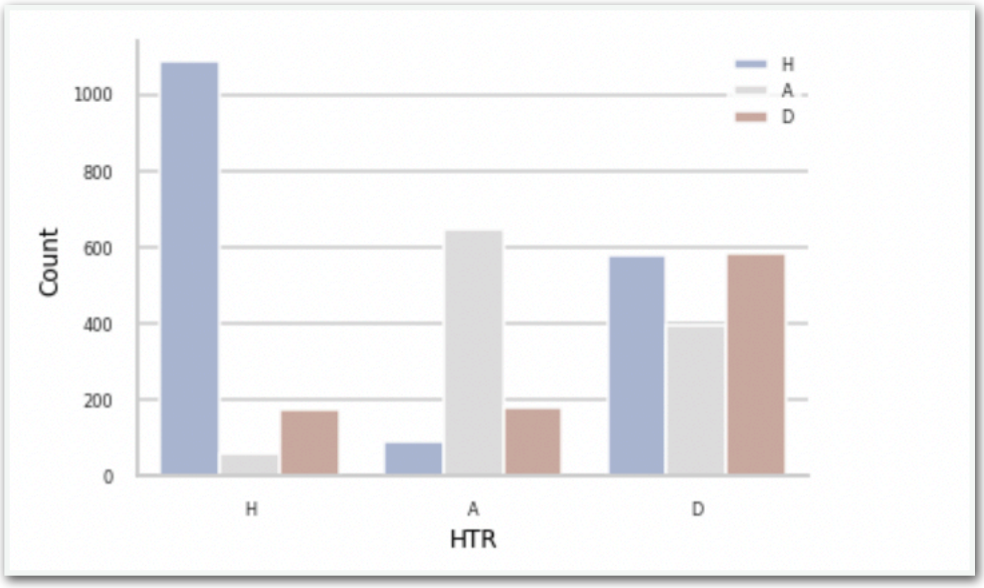
- *Bivariate Analysis:*



For Bivariate analysis, each feature is plotted again remaining features to understand the relationship between each other.

FTHG *(Full time Home Goal)* and FTAG *(Full time Away Goal)* are the values which clearly indicate who will win. So studying these 2 variables can be the best way to to predict FTR.

Whichever feature has higher value that team wins which indicates that the team which score more Goals at Full Time wins the match. Basically this is how football works.



Following points is observed from the bar graph:

1. The team Leading at Half Time almost always goes on to win the game at Full time.

2. If the game is level at Half Time it is more likely Home team will win than the Away team. Although the most likely outcome is a Draw only.

3. So HTR is a very important variable to determine who wins at Full time.

And below mentioned points can be concluded:-

1. There is a Higher percentage of Home team winning, so clearly the team playing at Home has an advantage.
2. Goals Scored at Full time (FTHG - *Home Goals*, FTAG - *Away Goals*) determine FTR - *Full Time Result* i.e. which team will go on to win the game, team which score more Goals at FT wins the match.
3. The Home team usually score more goals. Ex While winning Home team score mean of 2.5 Goals as compared to 2.1 Goals by Away team while winning.
4. HTR (*Half Time Result*) is a very important variable to determine who wins at Full time. As we saw the Team winning at Half team does not usually end up Losing at Full time. So this Variable can effectively predict who is likely to win at full time.

Methods and Algorithms

Clustering Techniques:

Now we have the dataset ready for modeling. Before we model, we will scale the dataset using MinMaxScaler and apply K-Means Clustering. To find optimal number of clusters, silhouette analysis and Elbow method is applied. K-Means is implemented with cluster k=4.

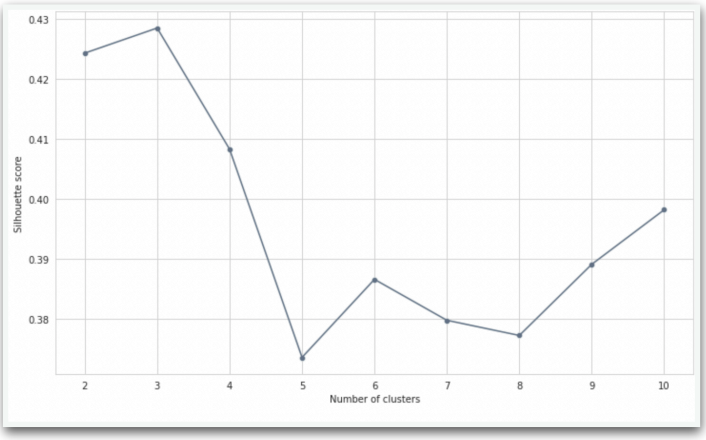
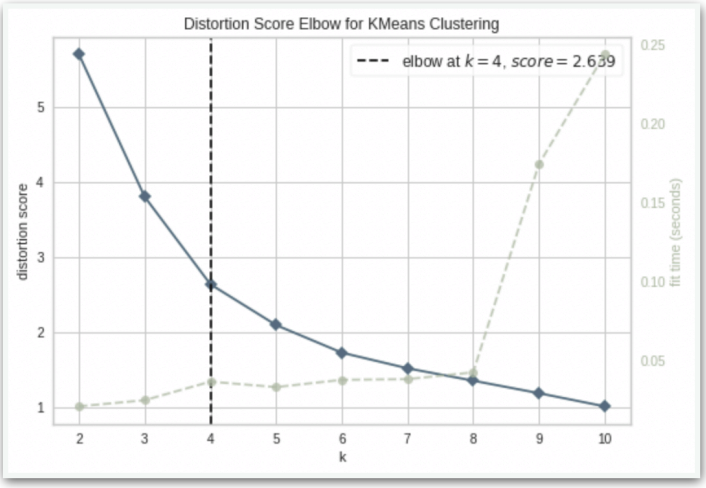
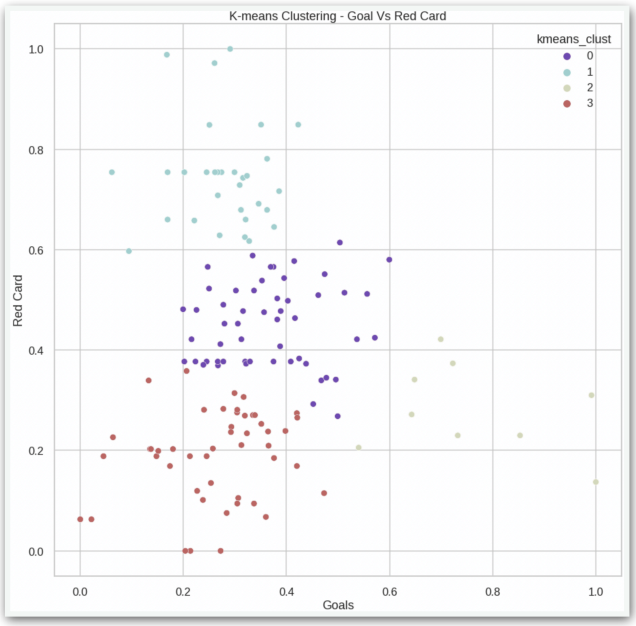


Fig 9 - Silhouette Analysis



Elbow Method

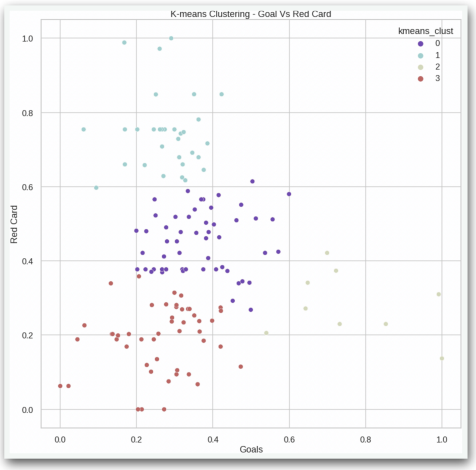


K-Means Clustering

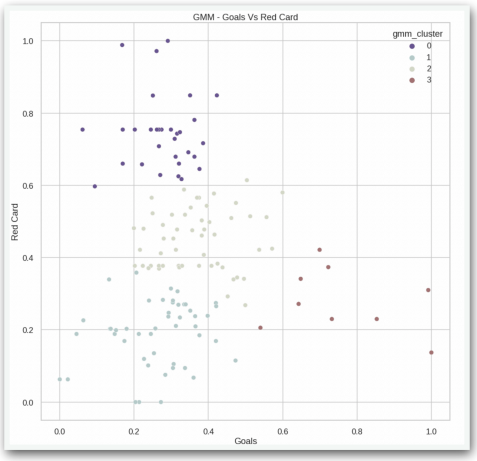
From the above scatterplot, following clusters can be observed:-

- Cluster 0 - Team with Average Goals and Average number of Red Cards.
- Cluster 1 - Teams with Less Goals but highest number of Red Cards. This team calls for a special attention for a good coach.
- Cluster 2 - Teams with more Goals and less Red Cards - This indicates a Good Team.
- Cluster 3 - Team with Less Goal and Less Red Cards.

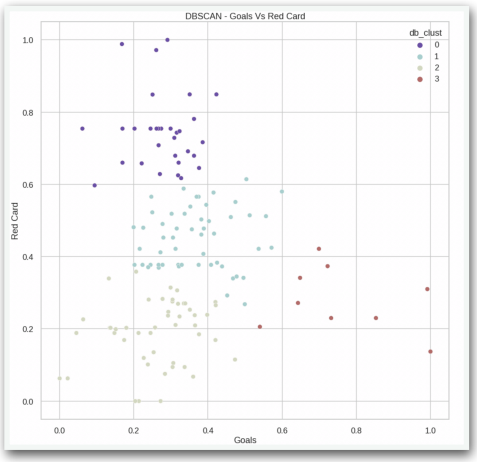
Applying various clustering techniques - Kmeans, Gaussian and Dbscan shows same results.



K-Means Clustering



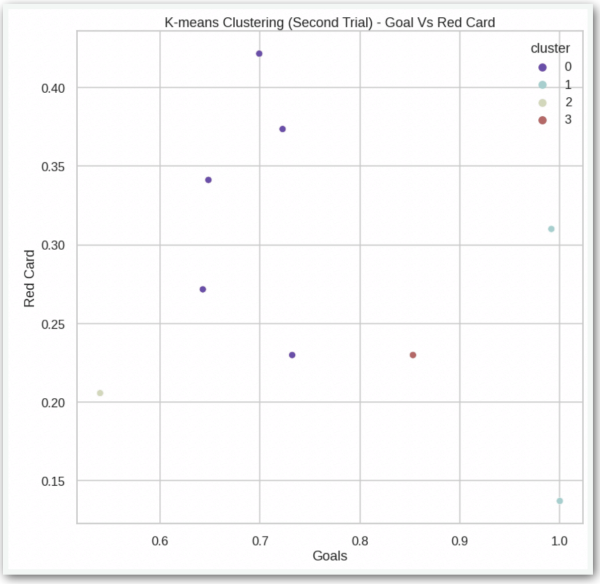
Gaussian Clustering



DBSCAN Clustering

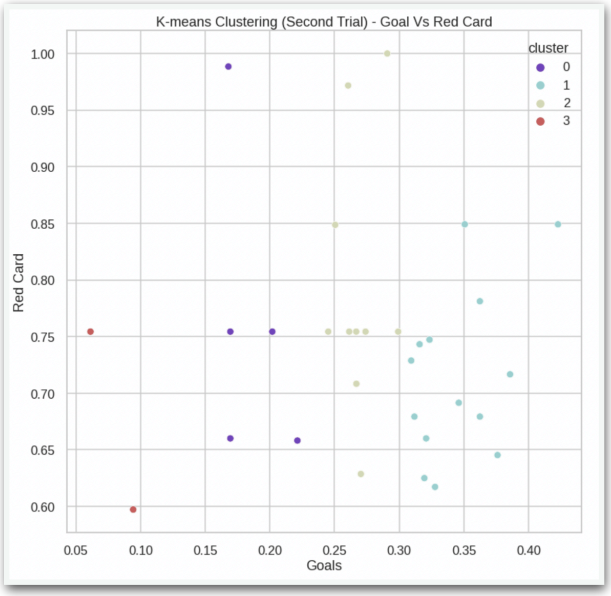
Next step, apply Fractal Clustering to identify the best and worst performing soccer team who has Good Goal score to Red Card ratio (aka Sharpe Ratio). i.e. *GOLDEN CLUSTER*.

- Identify Top 2 Best Performing Soccer teams based on Goals scored to accumulated Red Cards ratio by applying K-means clustering repetitively on the resultant cluster till the objective is achieved.



From the plot, the cluster 1 represents the group of best soccer team in terms of Goal-RedCard ratio.

- Identify Top 2 Worst Performing Soccer teams based on Goals scored to accumulated Red Cards ratio by applying K-means clustering repetitively on the resultant cluster till group of bad performing teams are found.



From the plot, the cluster 3 represents the group of bad performing soccer team in terms of Goal-RedCard ratio.

Classification And Regression Technique:

Before Data Modeling, encode the categorical feature with OneHot Encoding and scale numerical features using MinMax Scaler. Use Column Transformer as shown to transform all the columns before splitting the dataset into training and test set.

```
X = df[categorical_features + numeric_features]

colum_transformer = ColumnTransformer([('numerical', MinMaxScaler(), numeric_features),('categorical', OneHotEncoder(), categorical_feature

X = colum_transformer.fit_transform(X)

print(X)
```

```
[[0.22222222 0.14285714 0.2        ... 0.         0.         1.        ]
 [0.22222222 0.         0.2        ... 0.         0.         1.        ]
 [0.         0.28571429 0.         ... 1.         0.         0.        ]
 ...
 [0.44444444 0.         0.4        ... 0.         0.         1.        ]
 [0.11111111 0.14285714 0.2        ... 0.         1.         0.        ]
 [0.22222222 0.14285714 0.2        ... 0.         1.         0.        ]]
```

- For Regression , implement Muller Loop to run the training dataset against - “MLPRegressor", "LinearRegression", "RandomForestRegressor","KNNRegressor", "LogisticRegression", "AdaBoost" and display their accuracy.

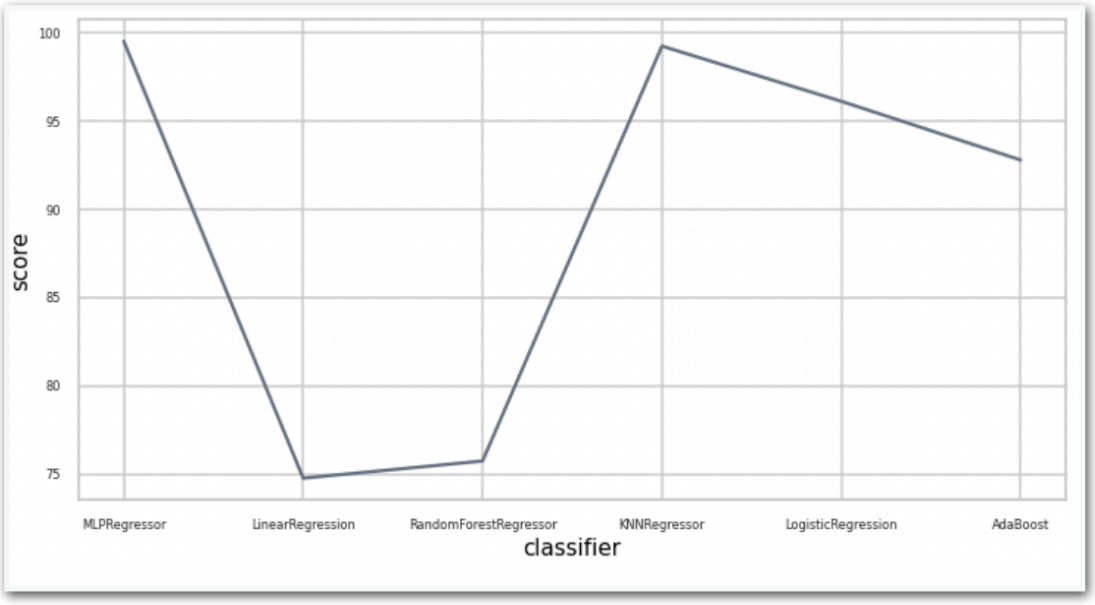
Popular regression algorithm are selected as follows:

- MLP Regressor : it is Multi Layer Perceptron regressor which optimizes the square error using LBFGS or stochastic gradient descent.
- Linear Regressor : it is an ordinary least squares Linear Regression.
- KNN Regressor : This algorithm learning is based on K nearest neighbors of each query point, where k is an integer value specified by the user.
- Random Forest: It is basically a set of decision trees the randomly selected subset of the training dataset. Then it collects the votes from different decision trees to decide the final value or label.
- Logistic Regressor : This algorithm measures the relationship between the categorical dependent variables and one or more independent variable , by estimating the probability of occurrence of an event, using its logistics function.:
- AdaBoost: Is an optimized distributed gradient boosting algorithm popular for its efficiency and flexibility.It provides a parallel tree boosting to solve many data science problems.

Implement **Muller Loop** to run the training dataset against - "MLPRegressor", "LinearRegression", "RandomForestRegressor","KNNRegressor", "LogisticRegression", "AdaBoost" and display their accuracy.

```
names = ["MLPRegressor", "LinearRegression", "RandomForestRegressor",
         "KNNRegressor", "LogisticRegression",
         "AdaBoost"]

classifiers = [
    MLPRegressor(random_state=1, max_iter=800),
    LinearRegression(),
    RandomForestRegressor(max_depth=2, random_state=0),
    KNeighborsRegressor(n_neighbors=2),
    LogisticRegression(),
    AdaBoostRegressor(random_state=0, n_estimators=100)]


max_score = 0.0
max_class = ''
score_list = []
clf_list = []

# iterate over classifiers
for name, clf in zip(names, classifiers):
    start_time = time.time()
    clf.fit(X_train, y_train)
    score = 100.0 * clf.score(X_test, y_test)
    score_list.append(score)
    clf_list.append(name)
    print('Regression  Classifier = %s, Score (test, accuracy) = %.2f,' %(name, score), 'Training time = %.2f seconds' % (time.time() - sta

    if score > max_score:
        clf_best = clf
        max_score = score
        max_class = name

print(80*'-' )
print('Best --> Regression Classifier = %s, Score (test, accuracy) = %.2f' %(max_class, max_score))
#plot the output of the various algorithms
```

```
Regression  Classifier = MLPRegressor, Score (test, accuracy) = 99.51, Training time = 1.94 seconds
Regression  Classifier = LinearRegression, Score (test, accuracy) = 74.78, Training time = 0.00 seconds
Regression  Classifier = RandomForestRegressor, Score (test, accuracy) = 75.75, Training time = 0.24 seconds
Regression  Classifier = KNNRegressor, Score (test, accuracy) = 99.24, Training time = 0.02 seconds
Regression  Classifier = LogisticRegression, Score (test, accuracy) = 96.12, Training time = 0.08 seconds
Regression  Classifier = AdaBoost, Score (test, accuracy) = 92.79, Training time = 0.29 seconds
--------------------------------------------------------------------------------
Best --> Regression Classifier = MLPRegressor, Score (test, accuracy) = 99.51
```



From the above plot, it is concluded that MLP regressor gives highest accuracy.

- For Classification, we have selected the following algorithms:

- K Nearest Neighbors: It is used to create models which does the prediction based on similarity. It stores all the available cases provided in the training dataset and classifies the new data based on similarity.
- Gaussian Naïve Bayes: It is a probabilistic algorithm used for classification. It is based on probability models that incorporate strong independence assumptions.
- Random Forest: It is basically a set of decision trees the randomly selected subset of the training dataset. Then it collects the votes from different decision trees to decide the final value or label
- XGBoost: Is an optimized distributed gradient boosting algorithm popular for its efficiency and flexibility.It provides a parallel tree boosting to solve many data science problems.

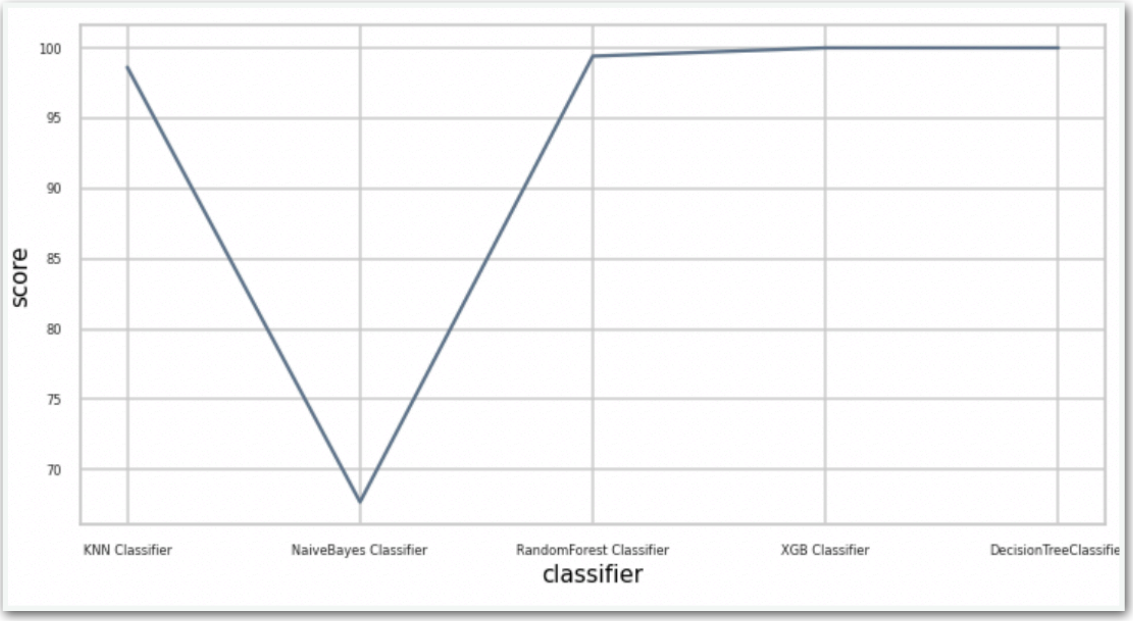
Ensemble learning: We are finally using Stacking method which is an ensemble technique that uses predictions from multiple nodes to build new model. The new model in our project corresponds to voting classifier which will decide the prediction label for the given data depending on voting.

Popular classification algorithms are applied to the reduced dataset, using classification techniques. The optimized dataset is split into testing and training datasets prior to applying the algorithm. The algorithm is then trained with the training dataset, and the trained classifier is applied in the testing phase. **The objective is to use the algorithm/classifier with the best score**

and use it for crop prediction *or* to come up with an ensemble based voting technique to classify the label for given parameters.

```
names = ["KNN Classifier", "NaiveBayes Classifier", "RandomForest Classifier",
         "XGB Classifier", "DecisionTreeClassifier"]

classifiers = [
    KNeighborsClassifier(),
    GaussianNB(),
    ensemble.RandomForestClassifier(),
    XGBClassifier(),
    DecisionTreeClassifier()]

max_score = 0.0
max_class = ''
score_list = []
clf_list = []

# iterate over classifiers
for name, clf in zip(names, classifiers):
    start_time = time.time()
    clf.fit(X_train, y_train)
    score = 100.0 * clf.score(X_test, y_test)
    score_list.append(score)
    clf_list.append(name)
    print('Classification  Classifier = %s, Score (test, accuracy) = %.2f,' %(name, score), 'Training time = %.2f seconds' % (time.time() -

    if score > max_score:
        clf_best = clf
        max_score = score
        max_class = name

print(80*'-' )
print('Best --> Classification Classifier = %s, Score (test, accuracy) = %.2f' %(max_class, max_score))
#plot the output of the various algorithms
```

```
Classification  Classifier = KNN Classifier, Score (test, accuracy) = 98.62, Training time = 0.05 seconds
Classification  Classifier = NaiveBayes Classifier, Score (test, accuracy) = 67.70, Training time = 0.00 seconds
Classification  Classifier = RandomForest Classifier, Score (test, accuracy) = 99.41, Training time = 0.21 seconds
Classification  Classifier = XGB Classifier, Score (test, accuracy) = 100.00, Training time = 0.26 seconds
Classification  Classifier = DecisionTreeClassifier, Score (test, accuracy) = 100.00, Training time = 0.00 seconds
--------------------------------------------------------------------------------
Best --> Classification Classifier = XGB Classifier, Score (test, accuracy) = 100.00
```



From the above plot, it is concluded that XGB Classifier gives highest accuracy.

Cross-validation-k-Fold:

It is a statistical method commonly used in applied machine learning to compare and select a model for a given predictive modeling problem. In this technique, input data is split into 3 folds with preserving the percentage of sample for each class. A 'kf' object (StratifiedKFold splitting strategy) will be used to as determine the scores for each fold and average score for each Classifiers.

Classifiers used are:

1. Random Forest Classifier
2. Naive Bayes Classifier
3. XGB Classifier
4. K Neighbors Classifier

Performance metrics like Accuracy, Precision, Recall and F1 score is used to measure and evaluate the efficiency of the model.

```
pred_result = classification_report(y_test, y_pred)
print(pred_result)
```

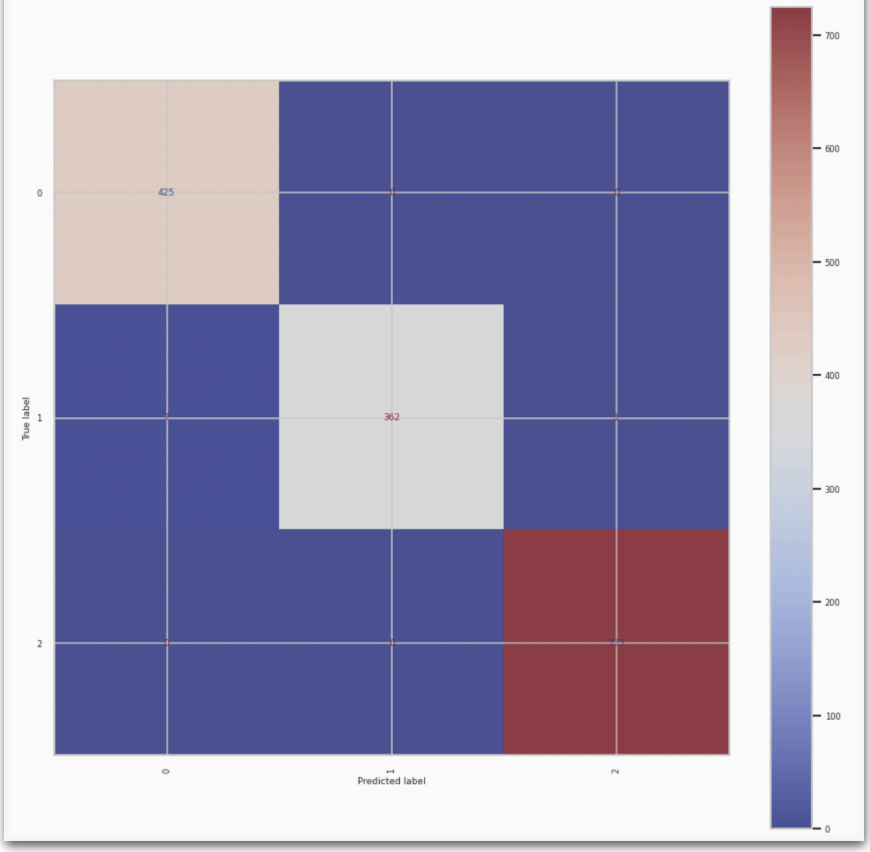
```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       425
           1       1.00      0.98      0.99       370
           2       1.00      1.00      1.00       725

    accuracy                           0.99      1520
   macro avg       0.99      0.99      0.99      1520
weighted avg       0.99      0.99      0.99      1520
```

The confusion matrix is used for evaluating the performance of a classification model. You can notice that this model is a good model since has high True Positive and True Negative, while low False Positive and False Negatives for test dataset.

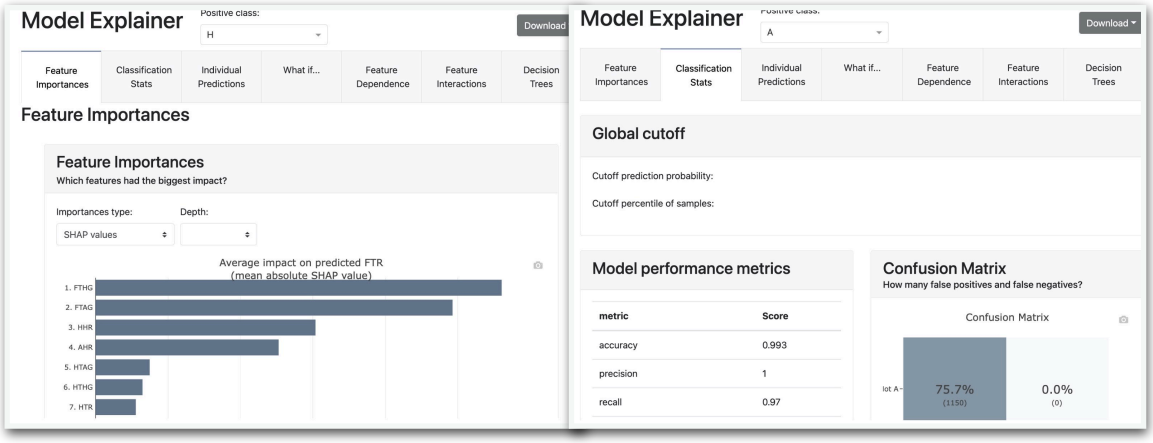
Confusion matrix to analyse the result.

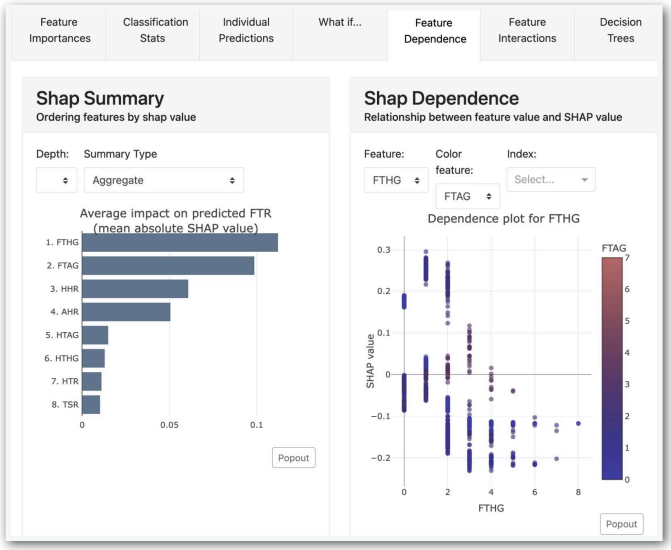
```
con_matrix = confusion_matrix(y_test, y_pred, labels=voting_clf.classes_)
figure, axes = plt.subplots(1, figsize=(15,15))
disp = ConfusionMatrixDisplay(confusion_matrix=con_matrix, display_labels=voting_clf.classes_)
disp.plot(ax = axes, xticks_rotation='vertical',cmap = "coolwarm")
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fec91095fd0>
```



Below is the screenshot of the dashboard:





Contributions Table

How has each feature contributed to the prediction?

Index: 1068 | Depth: 8 | Sorting: Ab

Reason	Effect
Average of population	24.84%
HHR = 0.0	-10.52%
FTHG = 0.0	-4.59%
HTR = 1.0	-3.67%
AHR = 0.1	-2.69%
HTHG = 0.0	-1.84%
FTAG = 1.0	-1.06%
HTAG = 1.0	-0.61%
TSR = 0.6296296296296297	+0.14%
Other features combined	+0.0%
Final prediction	-0.0%

Experiments and Results

We performed different univariate and bivariate analysis. Also, performed encoding of categorical variables. Based on manifold learning added three latent variables (details) to model with better accuracy. Performed Shap method to find out most important features. Performed baseline model prediction.

Top features were also selected and explain it with explainer dashboard.

To make the data balanced, SMOTE technique was applied for upsampling imbalanced data and again performed baseline model prediction.

Next, Muller loop was implemented to check different classification and regression techniques. We concluded by performing Cross fold validation for optimization and visualizing the Confusion metrics for performance evaluation.

Conclusion

The empirical result show that we can classify the outcome of the football match accurately by applying the given predictive model on the game info provided by Sports News channel. This study can also be extended to include player performance if the feature extraction is done with large volume of data correctly and optimally with increased accuracy of our model.

References:

[1]Applying Data Mining Techniques to Football Data from European Championships- https://paginas.fe.up.pt/~prodei/comic06/p6.pdf

[2] https://scikit-learn.org/

[3]kaggle.com

[4] datahub.io