Analytics Systems Technology

ALY6140, WINTER 2022

Capstone Project Draft

Capstone Group: 09

Submitted by: Abhinav Jain

Submitted To: Prof. Richard Zhi

Date: 03/31/2022

Football Match Probability Prediction

Introduction:

Predicting match outcomes is the most difficult undertaking, and it always delights football fans to try to anticipate the conclusion of each match. In the discipline of data science, football takes center stage. We will create models to forecast match results using event recognition and team analysis in this paper. The outcome of a match between two teams is mostly determined by their current form, but the teams' current form may be determined by looking at their recent sequence results against other teams. As a result, the match's probability might differ between two teams.

William McGregor, a Perthshire lord, and Aston Villa F.C. director was the driving force behind meetings in London and Manchester in 1888 with 12 football teams in the hopes of organizing a league. These 12 clubs would eventually become the Football League's 12 founding members.

To forecast the outcome of a football game, the project dataset was acquired from Kaggle.com's "Football Match Probability Prediction." The information was gathered from 150000 past international football matches between 2019 and 2021 by the community prediction competition team. The information was gathered from over 9500 teams and 860 leagues. We'll choose 10-15 variables from this dataset, which comprises 189 variables and 72711 observations, depending on the project's needs, in order to forecast and show the results using multiple regression models.

Football is constantly in high demand since there are sports fanatics all around the world. Fans of football are nervous and interested in the game's uncertain outcomes, which adds to the thrills and tension. Football is the most fun activity for fans to divert their attention away from their daily concerns. Leagues will allow us to deduce patterns from the previous success and failure of matches, as well as another variable in this project team's data, to predict the outcome. It's based on a number of factors, including target, home_team_name, away_team_name, player id, coach id, league id, match date, ratings and other various factors.

Goals:

To assess the chance of a future event, data analytics, statistical algorithms, and machine learning approaches are used to compare data to previous events. Instead of only knowing what has happened, the goal is to make better predictions of what will happen in the future.

Predictive techniques are used to analyze the team's performance during the match. In this study, different models will be employed to estimate the results of a football match, including Random Forest Classifier, K Nearest Neighbor, and Logistics Regression.

The purpose of a data scientist is to collect data, do analysis, interpret the data in a meaningful way, and apply prediction models to the data.

Methods:

I: Predict the matches by getting historical data of the team with the home team coach and opponent team coach.
II. Predict the accuracy by knowing the history match dates of the home and opponent team
III. Predict the motivation of the team by evaluating the team history rating.

Questions to Investigate :

1. What was the team performance history at home play?
2. What was the rating of the opponent team?
3. In the past, when did a match help you?
4. Which coach had a better track record in previous leagues?
5. What were the results of a team's prior leagues?

Exploratory Data Analysis

The process of exploratory data analysis helps in analyzing the insights from the dataset which provide the statistical analysis and graphical representation of the data. In this dataset for the prediction of a football match from the huge dataset, a selected few variables were to apply the predictive models. Initially, after importing the dataset in a jupyter notebook, installing various modules which will require analyzing the dataset helped in the visualization of the statistics.

Description analysis:

In this task, the shape of the dataset consists of 110938 rows and 190 columns, whereas id is an integer, target, home team name, away team name, match date, away team history are float datatype. Selection of the variable is the most difficult task to undertake with the predictive models in mind.

- Shape of the dataset:

```
df.shape
```

```
(110938, 190)
```

- Datatypes of each variable

```
df.dtypes
```

```
id                              int64
target                         object
home_team_name                 object
away_team_name                 object
match_date                     object
                                ...
away_team_history_league_id_6   float64
away_team_history_league_id_7   float64
away_team_history_league_id_8   float64
away_team_history_league_id_9   float64
away_team_history_league_id_10  float64
Length: 190, dtype: object
```

Some common features used from the dataset:

1. Id: Match identity of the data available for the team
2. Target: this feature of the dataset will compare the home team and the away team
3. Home_team_name: team name which is called Home
4. Away_team_name: team name which defines away
5. Match_date: define the time zone of the match
6. League_name: it explains the name of the league
7. League_id: for two team league names can be found identical
8. Is_cup : the match played for the cup
9. Home_team_coach_id: it defines the id of the coach in Hometeam
10. Away_team_coach_id: it defines the id of the coach in the away team

There are more features that we have to use for the comparison to implement the predictive model in the dataset. The historical data are taken from the more than 150000 historical world football matches from 2019-2021 which define our goal in forecasting the outcome of the match by implementing the various models.

Data Extraction

After importing the data from the dataset. These features include historical data of the home team and the away team

```
df.head()
```

	id	target	home_team_name	away_team_name	match_date	league_name	league_id	is_cup	home_team_coach_id	away_team_coach_id	...	away_te
0	11906497	away	Newell's Old Boys	River Plate	2019-12-01 00:45:00	Superliga	636	False	468196.00000	468200.00000	...	
1	11984383	home	Real Estelí	Deportivo Las Sabanas	2019-12-01 01:00:00	Primera Division	752	False	516788.00000	22169161.00000	...	
2	11983301	draw	UPNFM	Marathón	2019-12-01 01:00:00	Liga Nacional	734	False	2510608.00000	456313.00000	...	
3	11983471	away	León	Morelia	2019-12-01 01:00:00	Liga MX	743	False	1552508.00000	465797.00000	...	
4	11883005	home	Cobán Imperial	Iztapa	2019-12-01 01:00:00	Liga Nacional	705	False	429958.00000	426870.00000	...	

5 rows × 190 columns

```
df.shape
```

```
(110938, 190)
```

Data Cleanup: In this task after analyzing the dataset, we drop many variables from the dataset and took variables as per our requirements which will help in predicting the accuracy of the model are given below:

- 'home_team_history_is_play_home_1'
- 'home_team_history_opponent_goal_1'
- 'home_team_history_rating_1'
- 'home_team_history_coach_1'
- 'home_team_history_league_id_1'
- 'away_team_history_is_play_home_1'
- 'away_team_history_goal_1'
- 'away_team_history_opponent_goal_1'
- 'away_team_history_rating_1'
- 'away_team_history_opponent_rating_1'
- 'away_team_history_coach_1'

```
data= df[[
    'target', 'home_team_name', 'away_team_name', 'league_name',
    'home_team_coach_id', 'away_team_coach_id', 'home_team_history_is_play_home_1',
    'home_team_history_opponent_goal_1', 'home_team_history_rating_1','home_team_history_coach_1',
    'home_team_history_league_id_1', 'away_team_history_is_play_home_1',
    'away_team_history_goal_1', 'away_team_history_opponent_goal_1', 'away_team_history_rating_1',
    'away_team_history_opponent_rating_1', 'away_team_history_coach_1'
]].copy()

data.head()
```

	target	home_team_name	away_team_name	league_name	home_team_coach_id	away_team_coach_id	home_
0	away	Newell's Old Boys	River Plate	Superliga	468196.00000	468200.00000	
1	home	Real Estelí	Deportivo Las Sabanas	Primera Division	516788.00000	22169161.00000	
2	draw	UPNFM	Marathón	Liga Nacional	2510608.00000	456313.00000	
3	away	León	Morelia	Liga MX	1552508.00000	465797.00000	
4	home	Cobán Imperial	Iztapa	Liga Nacional	429958.00000	426870.00000	

```
data.shape
```

```
(110938, 17)
```

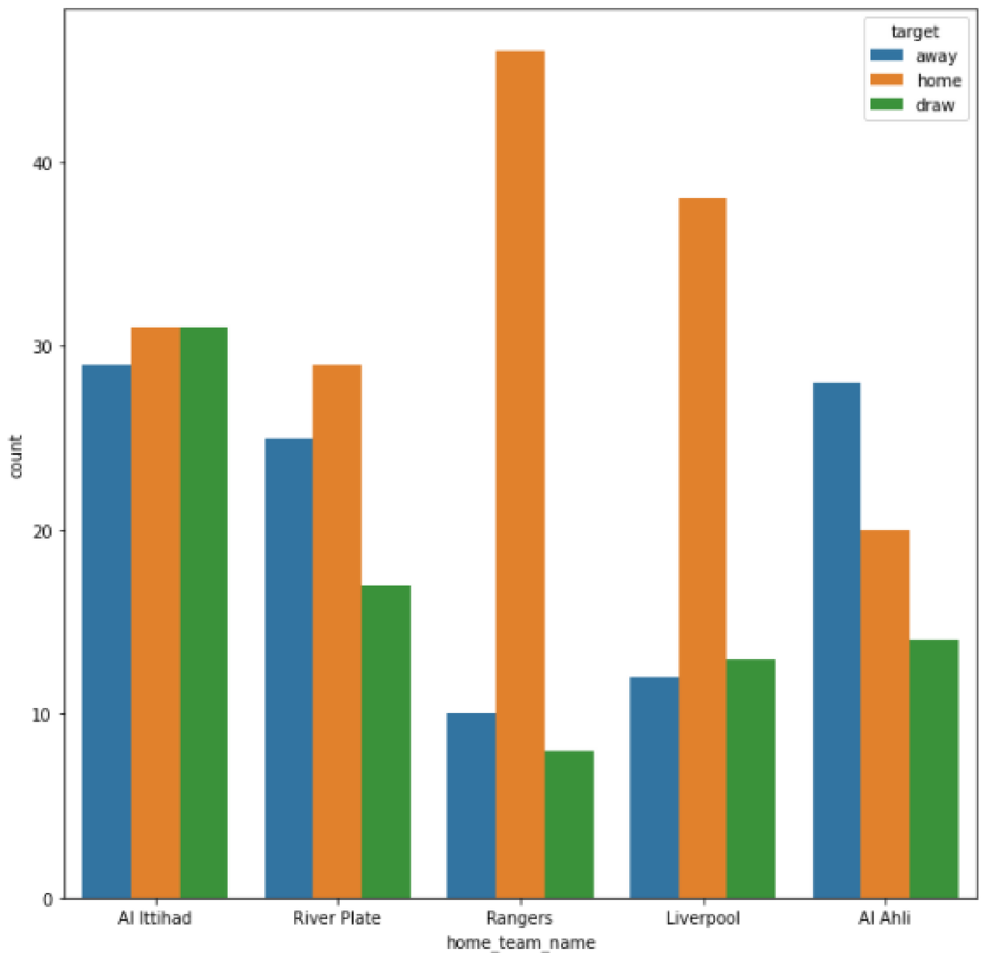
```
df.dtypes
```

```
id                               int64
target                          object
home_team_name                  object
away_team_name                  object
match_date                      object
                                 ...
away_team_history_league_id_6  float64
away_team_history_league_id_7  float64
away_team_history_league_id_8  float64
away_team_history_league_id_9  float64
away_team_history_league_id_10 float64
Length: 190, dtype: object
```

Data Analysis and Interpretation:

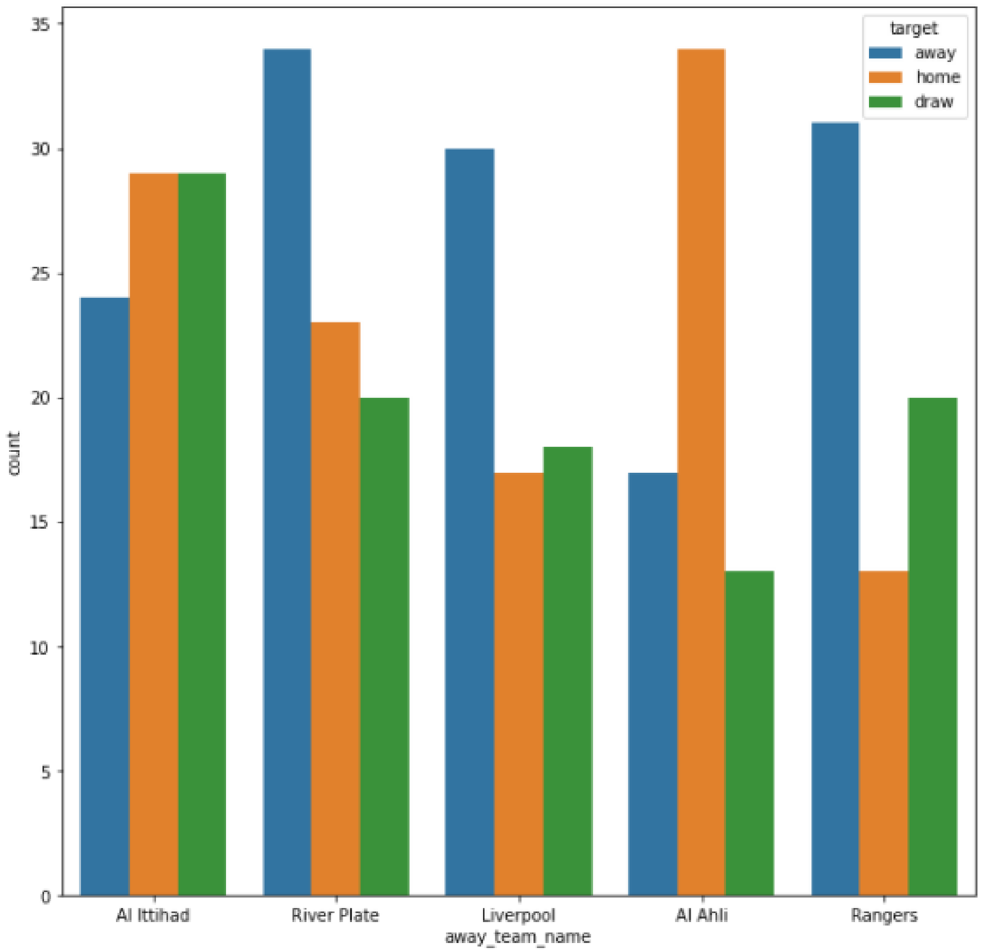
1. In this task analyses the comparison of the home team with the name and target of the team. The below bar graph is shown to describe the home team name with counts with comparison away, home, draw as we are targeting for the prediction of the match.

```
plt.figure(figsize=(10,10))
sns.countplot(x="home_team_name",hue="target",data=df,order=df.home_team_name.value_counts().iloc[:5].i
plt.show()
```



2. In this task analyses the comparison of the away team with name and target of the team The below bar graph is shown to describe the away team name with counts with comparison away, home, draw as we are targeting for the prediction of the match.

```
plt.figure(figsize=(10,10))
sns.countplot(x="away_team_name",hue="target",data=df,order=df.away_team_name.value_counts().iloc[:5].i
plt.show()
```



3. In this task collect the data of the team history at home team and away team

```
df.groupby('home_team_name')[play_home_features].mean()
```

home_team_name	home_team_history_is_play_home_1	home_team_history_is_play_home_2	home_team_history_is_play_home_3
07 Vestur	0.53846	0.38462	0.53846
1. FC M'gladbach	0.00000	1.00000	0.00000
1. FC Merseburg	0.00000	0.66667	0.50000
1. Maj Ruma	0.20000	0.50000	0.50000
12 de Octubre	0.32000	0.64000	0.44000
...	...	...	...
Žilina	0.34483	0.48276	0.55172
Žilina II	0.30000	0.65000	0.50000
Žilina U19	0.14286	0.71429	0.28571
Župa	0.00000	0.50000	0.00000
Žďár nad Sázavou	0.25000	0.50000	0.25000

9813 rows × 20 columns

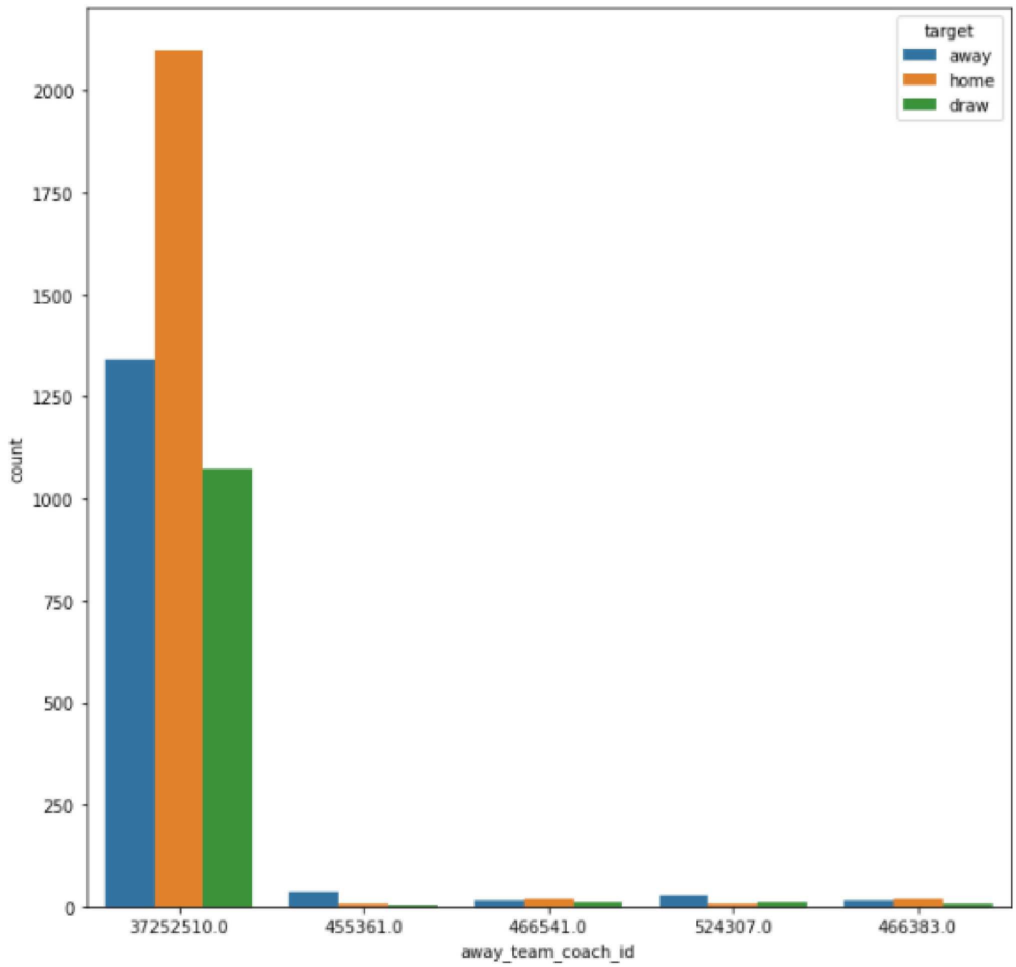
```
df.groupby('home_team_name')[opponent_rating_features].mean()
```

home_team_name	home_team_history_opponent_rating_1	home_team_history_opponent_rating_2	home_team_history_opponent_r
07 Vestur	8.61168	8.75077	[illegible]
1. FC M'gladbach	11.37313	9.09024	1[illegible]
1. FC Merseburg	9.23196	7.66353	[illegible]
1. Maj Ruma	7.18673	5.48778	[illegible]
12 de Octubre	7.52886	7.21602	[illegible]
...	...	...	
Žilina	6.19983	6.05111	[illegible]
Žilina II	8.96911	7.35407	[illegible]
Žilina U19	7.32640	7.03376	[illegible]
Župa	8.55600	6.40655	1[illegible]
Žďár nad Sázavou	11.56933	12.10130	1[illegible]

9813 rows × 20 columns

4. In this task analyze the track record of the coach in the previous matches with the comparison between away, home teams, and draw.

```
plt.figure(figsize=(10,10))
sns.countplot(x="away_team_coach_id",hue="target",data=df,order=df.away_team_coach_id.value_counts().il
plt.show()
```



5. In this task analyze the track record of the history cup of the away team

```
df.groupby('away_team_name')[away_is_cup_features].mean()
```

away_team_name	away_team_history_is_cup_1	away_team_history_is_cup_2	away_team_history_is_cup_3	away_team_history_is
07 Vestur	0.00000	0.00000	0.11111	
1. FC M'gladbach	0.00000	0.00000	0.00000	
1. FC Merseburg	0.00000	0.00000	0.00000	
1. Maj Ruma	0.00000	0.00000	0.00000	
12 de Octubre	0.11538	0.07692	0.03846	
...	...	...	...	
Žilina	0.10714	0.10714	0.21429	
Žilina II	0.00000	0.00000	0.00000	
Žilina U19	0.00000	0.00000	0.00000	
Župa	0.00000	0.00000	0.00000	
Žďár nad Sázavou	0.00000	0.00000	0.00000	

9892 rows × 10 columns

Predictive Models: In the present job market, the future of predictive modeling in statistics is the key emphasis. We may forecast the answer via predictive modeling by adopting modeling solutions based on historical and recent data. Data is obtained and created using predictions, and then confirmed using the most up-to-date information available once the dataset has been trained.

1. **Logistic Regression Model** When the variable is categorical, what type of regression analysis should be used? (binary). The logistic regression, like other regression studies, is a predictive analysis. To describe data and explain the connection between one dependent binary variable and one or more nominal, ordinal, interval, or ratio-level independent variables, logistic regression is utilized.
   - Easier to Implement
   - Efficient to train
   - Fast while classifying unknown records
   - Interpret the Model coefficient as indicators

   Logistics Regression helps in predictive analysis to evaluate the relationship between the nominal ordinal of the interval, by checking the ratio-level independent variables, or one dependent binary variable. The regression studies suggest that the cross-validation is 1.0462 with 46% of accuracy.

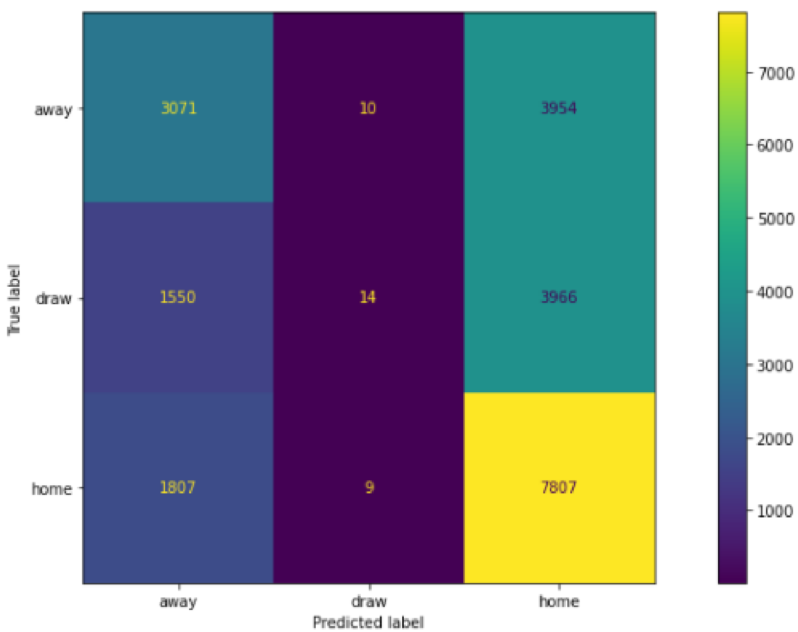
```
Accuracy score: 0.4908959798089057

              precision    recall  f1-score   support

        away       0.48      0.44      0.46      7035
        draw       0.42      0.00      0.01      5530
        home       0.50      0.81      0.62      9623

    accuracy                           0.49     22188
   macro avg       0.47      0.42      0.36     22188
weighted avg       0.47      0.49      0.41     22188
```

```python
# Confusion matrix
cm = confusion_matrix(y_test, y_pred, labels=pipeline.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=pipeline.classes_)
fig, ax = plt.subplots(figsize=(15,7))
disp.plot(ax=ax)
plt.show()
```

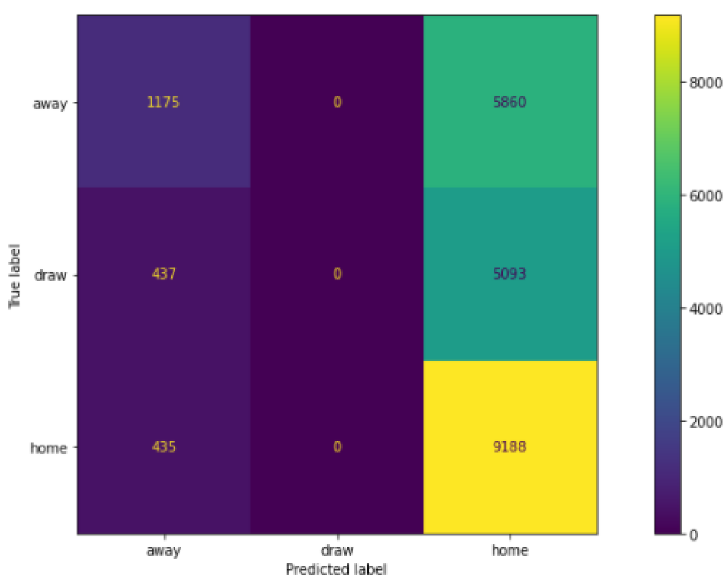


2. **Random Forest Classifier**: is a classifier that combines a number of decision trees on different subsets of a dataset and averages the results to increase the dataset's predicted accuracy." Rather than depending on a single decision tree, the random forest collects predictions from each tree and makes decisions based on them.
   • Works well with large dimensional data
   • Working with a subset
   • Fast to train than the decision tree
   • Easily work with hundreds of features
   • Low correlation is the key
   Random Forest classifier is based on supervised learning and provides the solution for regression and creating multiple decision trees by evaluating the average prediction of that tree Here, the cross-validation is 1.0462 with 46% of accuracy.

```
Accuracy score: 0.46705426356589147
```

	precision	recall	f1-score	support
away	0.57	0.17	0.26	7035
draw	0.00	0.00	0.00	5530
home	0.46	0.95	0.62	9623
accuracy			0.47	22188
macro avg	0.34	0.37	0.29	22188
weighted avg	0.38	0.47	0.35	22188

```
# Confusion matrix
cm = confusion_matrix(y_test, y_pred, labels=pipeline.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=pipeline.classes_)
fig, ax = plt.subplots(figsize=(15,7))
disp.plot(ax=ax)
plt.show()
```



3. **KNN-neighbors:** The supervised learning technique K-nearest neighbors (KNN) is used for both regression and classification. By computing the distance between the test data and all of the training points, KNN tries to predict the proper class for the test data. Then choose the K number of points that are the most similar to the test data.
   - Highly accurate Predictions
   - Solves both classification and regression problem statement
   - KNN algorithm for multiclass classification
   - Recommendation Systems

KNN regression model is defined by the analyst by measuring the size of the neighborhood to check the validation of the model. Here, the cross-validation is 9.79.17 with 39% of accuracy

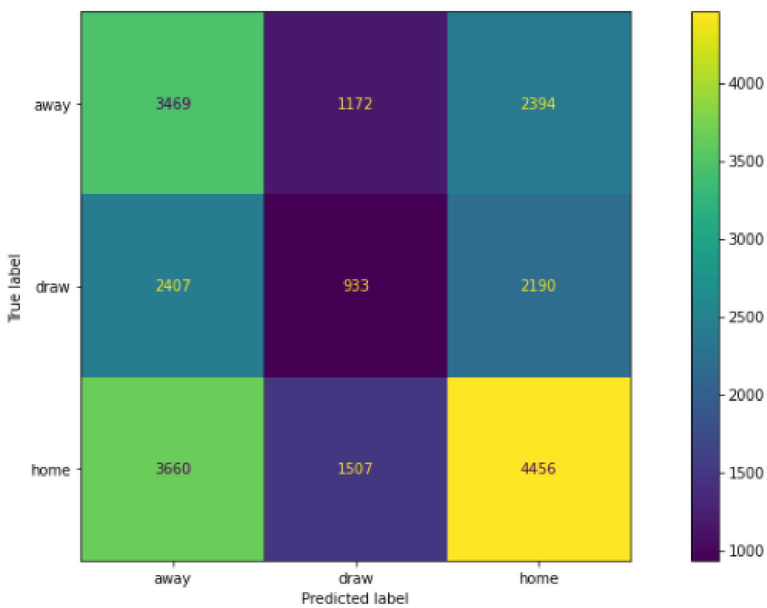
```
Accuracy score: 0.3992248062015504
```

```
              precision    recall  f1-score   support

        away       0.36      0.49      0.42      7035
        draw       0.26      0.17      0.20      5530
        home       0.49      0.46      0.48      9623

    accuracy                           0.40     22188
   macro avg       0.37      0.37      0.37     22188
weighted avg       0.39      0.40      0.39     22188
```

```
# Confusion matrix
cm = confusion_matrix(y_test, y_pred, labels=pipeline.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=pipeline.classes_)
fig, ax = plt.subplots(figsize=(15,7))
disp.plot(ax=ax)
plt.show()
```



Interpretive and Conclusions

Logistic Regression confusion matrix provides the data predictive label which includes home team, away that is opponent team, and draw team which has matched with home and away has 1807 matches, 9 of home and draw, home and home has 7807, draw and away has 1550, draw and draw 14 matches, draw at home has 3966, away and away 3071 matches, away and draw has 10 matches, away and home has 3954 matches which leas to the 49% of accuracy. However, the cross-validation log loss is 1.0208.

Random classifier confusion matrix provides the data predictive label which includes home team, away that is opponent team, and draw team which has matched with home and away has 435 matches, 0 of home and draw, home and home has 9188, draw and away has 437, draw and draw 0 matches, draw at home has 5093, away and away 1175 matches, away and draw has 0 matches, away and home has 5860 matches which least to the 46% of accuracy. In this case, cross-validation log loss is 0.0462

KNN confusion matrix provides the data predictive label which includes home team, away that is opponent team, and draw team which has matched with home and away has 3660 matches, 1507 of home and draw, home and home has 4456, draw and away has 2407, draw and draw 933 matches, draw at home has 2190, away and away 3469 matches, away and draw has 1172 matches, away and home has 2394 matches which lead to the 39% of accuracy. In this model, we can see the cross-validation is 9.7917.

In conclusion, we experimented with a variety of Machine Learning models and features to improve our prediction error as much as possible. The predicting outcome will help determine the instance outcome of the psychology towards the match and create the opportunity. Many organizations are using these forecasting methods to build for the outperformance of the results in the field of data science.

We removed numerous factors from the dataset after evaluating it and replaced them with variables that will help us anticipate the model's accuracy. After that compares the home team's name and target to the team's name and target. The bar graph below depicts the home team name with counts in relation to away, home, and draw since we are aiming for a match prediction. Compares the away side's name and goal to that of the home team The bar graph below depicts the away team's name with counts and a comparison of away, home, and draw, as we are aiming for a match prediction.

Logistics Regression helps in the evaluation of the link between the nominal ordinal of the interval and the ratio-level independent variables, or one dependent binary variable, in predictive analysis. According to regression research, the cross-validation is 1.0462 with a 46 percent accuracy.

The Random Forest classifier is based on supervised learning and provides a solution for regression and multiple decision tree creation by evaluating the tree's average prediction. The cross-validation, in this case, is 1.0462 with a 46 percent accuracy.

K-Nearest Neighbor The analyst defines the KNN regression model by measuring the size of the neighborhood to ensure the model's validity. The cross-validation score is 9.79.17, with a precision of 39%.

We built a model training and testing pipeline using our various model components to quickly and easily change and test new assumptions. We compared our forecasts to benchmark methodologies to better understand the predictive performance of our models. We discovered that this model may be improved to obtain data on the matches and the coach id, which appears to be an outlier. We may have predicted the models with less accuracy due to a lack of data. We need to anticipate the train and test the model in such a way that we can reach that accuracy to develop the model to the level of 90-95 percent accuracy, and we need to provide a better outcome after running the model numerous times.

= =
==
= =

References:

[1]Football Match Probability Prediction

https://www.kaggle.com/c/football-match-probability-prediction/data?select=test.csv

[2]learn

https://scikit-learn.org/stable/

[3]pandas

https://pandas.pydata.org/

[4]seaborn: statistical data visualization — seaborn 0.11.2 ...

seaborn: statistical data visualization. ¶. Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. For a brief introduction to the ideas behind the library, you can read the introductory notes or the paper.

https://seaborn.pydata.org

[5]sklearn.ensemble.RandomForestClassifier

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[6]sklearn.neighbors.KNeighborsClassifier

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[7]sklearn.model_selection.train_test_split

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[8]sklearn.metrics.accuracy_score

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

[9]sklearn.metrics.confusion_matrix

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

[10]sklearn.metrics.classification_report

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

[11]warnings - Warning control¶

https://docs.python.org/3/library/warnings.html

[12]sklearn.linear_model.LogisticRegression

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Appendix:

```python
#!/usr/bin/env python
# coding: utf-8

Run Cell | Run Below | Debug Cell
# In[1]:


#import modules
from sklearn import metrics, model_selection
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.impute import SimpleImputer
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, ConfusionMatrixDisplay
pd.options.display.float_format = '{:.5f}'.format

get_ipython().run_line_magic('matplotlib', 'inline')
import warnings
warnings.filterwarnings('ignore')


Run Cell | Run Above | Debug Cell
# In[2]:


df = pd.read_csv("train.csv")

```

```
Run Cell | Run Above | Debug Cell
# In[3]:


df.head()


Run Cell | Run Above | Debug Cell
# In[4]:


df.shape


Run Cell | Run Above | Debug Cell
# In[5]:


df.columns


Run Cell | Run Above | Debug Cell
# In[6]:


df['target']


Run Cell | Run Above | Debug Cell
# In[7]:


df['home_team_history_rating_5'].value_counts()


```

```
Run Cell | Run Above | Debug Cell
# In[8]:


df['league_id'].value_counts()


Run Cell | Run Above | Debug Cell
# In[9]:


#keep only rating features
rating_features = [x for x in  df if 'rating' in x]
rating_features


Run Cell | Run Above | Debug Cell
# In[ ]:





Run Cell | Run Above | Debug Cell
# In[10]:


data= df[[
    'target', 'home_team_name', 'away_team_name', 'league_name',
    'home_team_coach_id', 'away_team_coach_id', 'home_team_history_is_play_home_1',
    'home_team_history_opponent_goal_1', 'home_team_history_rating_1','home_team_history_coach_1',
    'home_team_history_league_id_1', 'away_team_history_is_play_home_1',
    'away_team_history_goal_1', 'away_team_history_opponent_goal_1', 'away_team_history_rating_1',
    'away_team_history_opponent_rating_1', 'away_team_history_coach_1'
]].copy()

```

```
]].copy()

data.head()


Run Cell | Run Above | Debug Cell
# In[11]:


data.shape


Run Cell | Run Above | Debug Cell
# In[12]:


df.dtypes


Run Cell | Run Above | Debug Cell
# In[13]:


df['away_team_history_coach_1'].value_counts()


Run Cell | Run Above | Debug Cell
# In[14]:


df['home_team_history_is_play_home_1'].value_counts()

```

```
Run Cell | Run Above | Debug Cell
# In[15]:


plt.figure(figsize=(10,10))
sns.countplot(x="home_team_name",hue="target",data=df,order=df.home_team_name.value_counts().iloc[:5].index)
plt.show()


Run Cell | Run Above | Debug Cell
# In[16]:


plt.figure(figsize=(10,10))
sns.countplot(x="away_team_name",hue="target",data=df,order=df.away_team_name.value_counts().iloc[:5].index)
plt.show()


# ## Team history at home

Run Cell | Run Above | Debug Cell
# In[17]:


#keep only play home history features
play_home_features = [x for x in  df if 'play_home' in x]
play_home_features


Run Cell | Run Above | Debug Cell
# In[18]:


df.groupby('home_team_name')[play_home_features].mean()
```

```
df.groupby('home_team_name')[play_home_features].mean()


# ### Rating of the oppenent team

Run Cell | Run Above | Debug Cell
# In[19]:


#keep only oppenent rating features
opponent_rating_features = [x for x in  df if 'opponent_rating' in x]
opponent_rating_features


Run Cell | Run Above | Debug Cell
# In[20]:


df.groupby('home_team_name')[opponent_rating_features].mean()


# ### In the past, when did a match help

Run Cell | Run Above | Debug Cell
# In[21]:


#keep only home team history macth date features
home_date_features = [x for x in  df if 'home_team_history_match_date' in x]
home_date_features
```

```
df['target'].value_counts()


Run Cell | Run Above | Debug Cell
# In[44]:


home_wins = df[df['target'] == "home"]


Run Cell | Run Above | Debug Cell
# In[47]:


home_wins[home_date_features]


# ## Which coach had a better track record

Run Cell | Run Above | Debug Cell
# In[23]:


away_coach_features = [x for x in  df if 'away_team_history_coach' in x]
away_coach_features


Run Cell | Run Above | Debug Cell
# In[24]:


plt.figure(figsize=(10,10))
sns.countplot(x="away_team_coach_id",hue="target",data=df,order=df.away_team_coach_id.value_counts().iloc[:5].index)
plt.show()
```

```
# ## Teams' prior league results

Run Cell | Run Above | Debug Cell
# In[25]:


#keep only away history cup
away_is_cup_features = [x for x in  df if 'away_team_history_is_cup' in x]
away_is_cup_features


Run Cell | Run Above | Debug Cell
# In[26]:


df.groupby('away_team_name')[away_is_cup_features].mean()


# ## Modelling

Run Cell | Run Above | Debug Cell
# In[27]:


#Make X and y
X = df[rating_features]
y = df['target']


Run Cell | Run Above | Debug Cell
# In[28]:


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

```
# ### Logistic Regression

Run Cell | Run Above | Debug Cell
# In[29]:


#build a simple pipeline that fill nans values with the mean and fit a logistic regression classifier without regulariation.
pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ("classifier", LogisticRegression(solver='sag',C=1e5))
])


Run Cell | Run Above | Debug Cell
# In[30]:


#evaluate the model log loss using cross validation
cv_scores = model_selection.cross_val_score(pipeline, X_train, y_train, cv=10,scoring='neg_log_loss')
print(f'The cross validation log loss is {-cv_scores.mean().round(4)}')


Run Cell | Run Above | Debug Cell
# In[31]:


#fit the model
pipeline.fit(X_train, y_train)

# Making predictions
y_pred = pipeline.predict(X_test)

# Measuring the accuracy of the model
print(f'Accuracy score: {accuracy_score(y_test, y_pred)}')
print('\n')
print(f'{classification_report(y_test, y_pred)}')
Run Cell | Run Above | Debug Cell
# In[32]:


# Confusion matrix
cm = confusion_matrix(y_test, y_pred, labels=pipeline.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=pipeline.classes_)
fig, ax = plt.subplots(figsize=(15,7))
disp.plot(ax=ax)
plt.show()


# ## Random Classifier

Run Cell | Run Above | Debug Cell
# In[33]:


#build a simple pipeline that fill nans values with the mean and fit a logistic regression classifier without regulariation.
pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ("classifier", RandomForestClassifier(max_depth = 3, random_state = 0))
])


Run Cell | Run Above | Debug Cell
# In[34]:


#evaluate the model log loss using cross validation
cv_scores = model_selection.cross_val_score(pipeline, X_train, y_train, cv=10,scoring='neg_log_loss')
print(f'The cross validation log loss is {-cv_scores.mean().round(4)}')
```

```
#FIT THE MODEL
pipeline.fit(X_train, y_train)

# Making predictions
y_pred = pipeline.predict(X_test)

# Measuring the accuracy of the model
print(f'Accuracy score: {accuracy_score(y_test, y_pred)}')
print('\n')
print(f'{classification_report(y_test, y_pred)}')


Run Cell | Run Above | Debug Cell
# In[36]:


# Confusion matrix
cm = confusion_matrix(y_test, y_pred, labels=pipeline.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=pipeline.classes_)
fig, ax = plt.subplots(figsize=(15,7))
disp.plot(ax=ax)
plt.show()


# ## kNEIGHBORS

Run Cell | Run Above | Debug Cell
# In[37]:


#build a simple pipeline that fill nans values with the mean and fit a logistic regression classifier without regulariation.
pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ("classifier", KNeighborsClassifier(n_neighbors=3))
])
```

```
Run Cell | Run Above | Debug Cell
# In[38]:


#evaluate the model log loss using cross validation
cv_scores = model_selection.cross_val_score(pipeline, X_train, y_train, cv=10,scoring='neg_log_loss')
print(f'The cross validation log loss is {-cv_scores.mean().round(4)}')


Run Cell | Run Above | Debug Cell
# In[39]:


#FIT THE MODEL
pipeline.fit(X_train, y_train)

# Making predictions
y_pred = pipeline.predict(X_test)

# Measuring the accuracy of the model
print(f'Accuracy score: {accuracy_score(y_test, y_pred)}')
print('\n')
print(f'{classification_report(y_test, y_pred)}')


Run Cell | Run Above | Debug Cell
# In[40]:


# Confusion matrix
cm = confusion_matrix(y_test, y_pred, labels=pipeline.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=pipeline.classes_)
fig, ax = plt.subplots(figsize=(15,7))
disp.plot(ax=ax)
plt.show()
```