Brac University

422

(Artificial Intelligence)

Project -[Football Match Prediction]

Id- 20241003
20101311
21341053
20201045

Sec - 04
Group- 01

Introduction

Predicting the Winning Football Team

- Football is played by 250 million players in over 200 countries (most popular sport globally).
- The English Premier League is the most popular domestic team in the world.
- We will be designing a predictive model which is capable of accurately predicting if the home team will win a football match?

Dataset Description

A prediction system was built to predict whether a home team will win its match or not.

Key to results data:

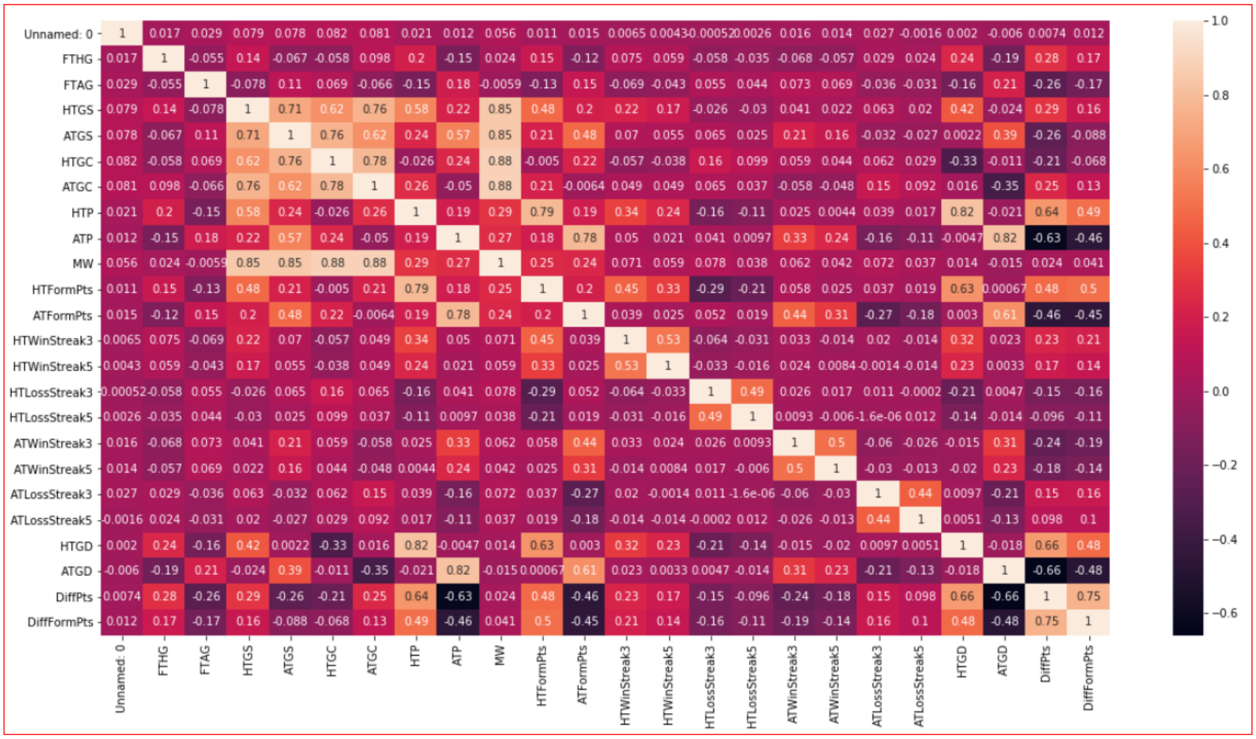
- Div = League Division
- Date = Match Date (dd/mm/yy)
- Time = Time of match kick-off
- HomeTeam = Home Team
- Away team = Away Team
- FTHG and HG = Full Time Home Team Goals
- FTAG and AG = Full-Time Away Team Goals
- FTR and Res = Full-Time Result (H=Home Win, D=Draw, A=Away Win)
- HTHG = Half Time Home Team Goals
- HTAG = Half Time Away Team Goals
- HTR = Half Time Result (H=Home Win, D=Draw, A=Away Win)

Match Statistics (where available)

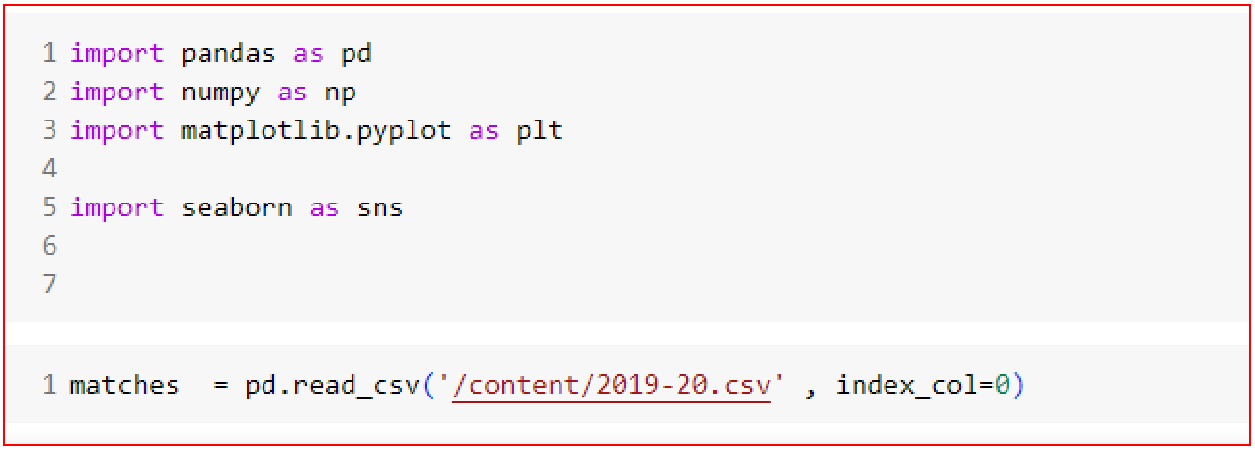
- Attendance = Crowd Attendance
- Referee = Match Referee
- HS = Home Team Shots
- AS = Away Team Shots
- HST = Home Team Shots on Target
- AST = Away Team Shots on Target
- HHW = Home Team Hit Woodwork
- AHW = Away Team Hit Woodwork
- HC = Home Team Corners
- AC = Away Team Corners
- HF = Home Team Fouls Committed
- AF = Away Team Fouls Committed

- HFKC = Home Team Free Kicks Conceded
- AFKC = Away Team Free Kicks Conceded
- HO = Home Team Offsides
- AO = Away Team Offsides
- HY = Home Team Yellow Cards
- AY = Away Team Yellow Cards *HR = Home Team Red Cards AR = Away Team Red Cards HBP = Home Team Bookings Points (10 = yellow, 25 = red) ABP = Away Team Bookings Points (10 = yellow, 25 = red)

So in the Dataset There are more than 25 plus features with 1 label which is Categorical , so it's a classification problem .

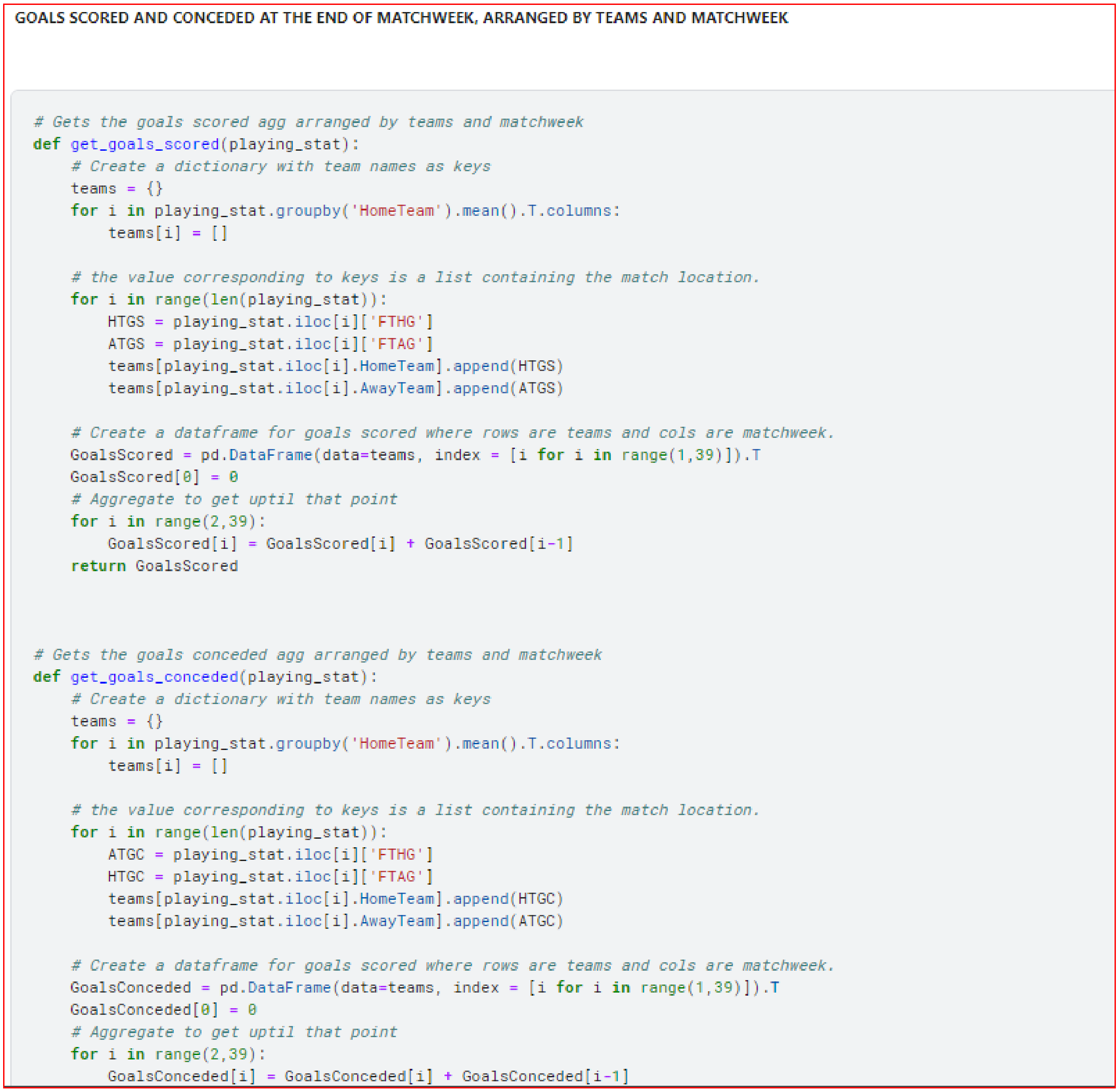


Source - https://www.kaggle.com/competitions/football-match-probability-prediction



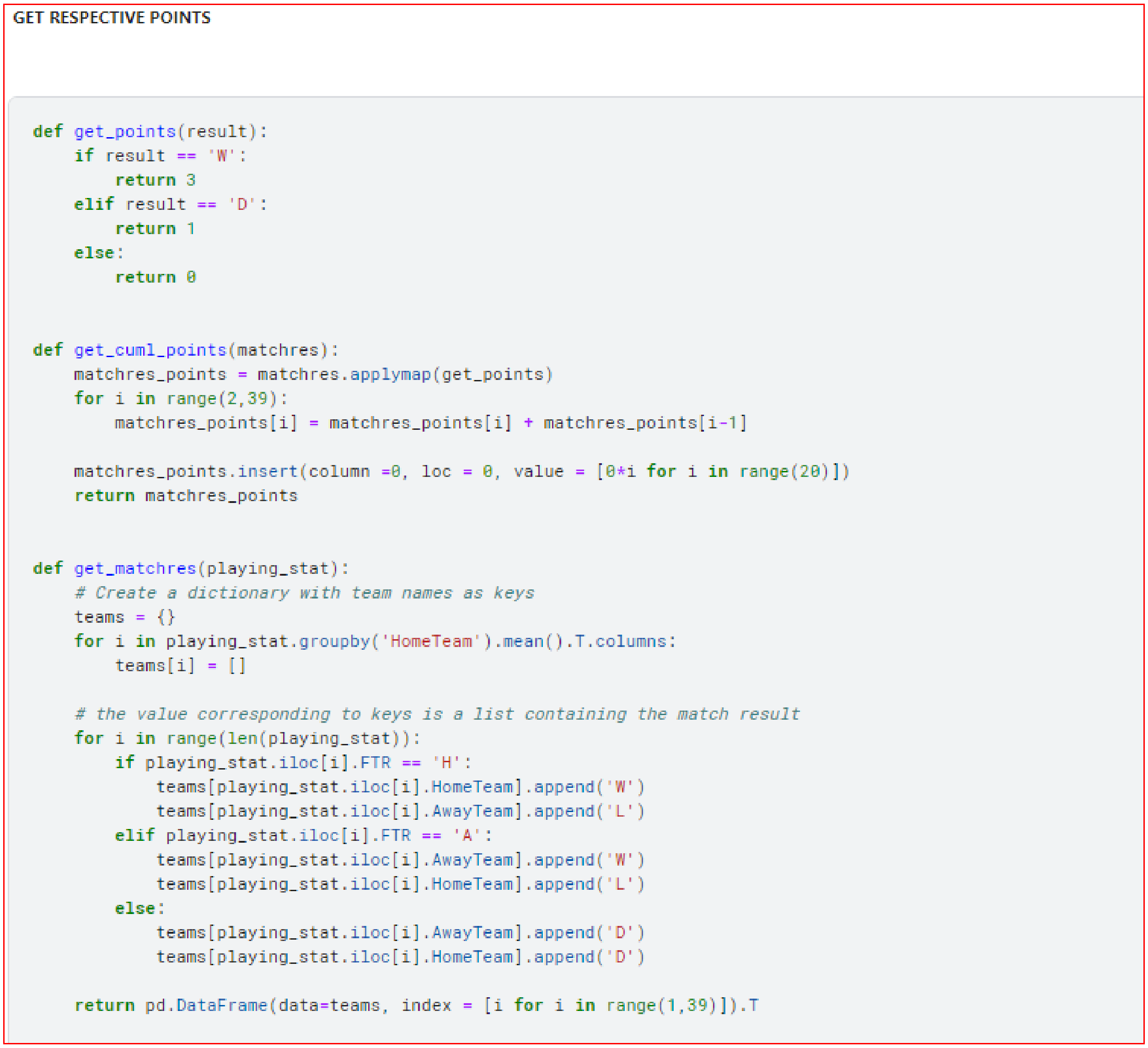
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import seaborn as sns


```

```
matches  = pd.read_csv('/content/2019-20.csv' , index_col=0)
```



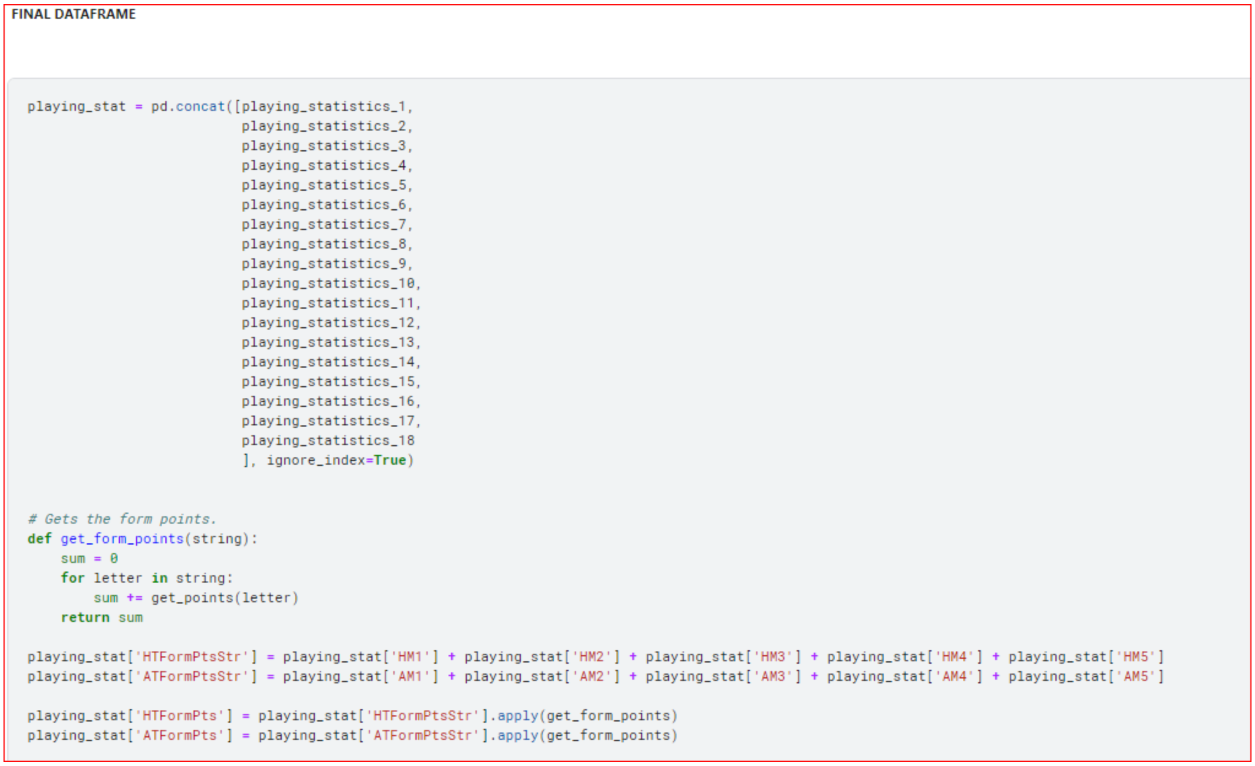
GOALS SCORED AND CONCEDED AT THE END OF MATCHWEEK, ARRANGED BY TEAMS AND MATCHWEEK

```
# Gets the goals scored agg arranged by teams and matchweek
def get_goals_scored(playing_stat):
    # Create a dictionary with team names as keys
    teams = {}
    for i in playing_stat.groupby('HomeTeam').mean().T.columns:
        teams[i] = []

    # the value corresponding to keys is a list containing the match location.
    for i in range(len(playing_stat)):
        HTGS = playing_stat.iloc[i]['FTHG']
        ATGS = playing_stat.iloc[i]['FTAG']
        teams[playing_stat.iloc[i].HomeTeam].append(HTGS)
        teams[playing_stat.iloc[i].AwayTeam].append(ATGS)

    # Create a dataframe for goals scored where rows are teams and cols are matchweek.
    GoalsScored = pd.DataFrame(data=teams, index = [i for i in range(1,39)]).T
    GoalsScored[0] = 0
    # Aggregate to get uptil that point
    for i in range(2,39):
        GoalsScored[i] = GoalsScored[i] + GoalsScored[i-1]
    return GoalsScored



# Gets the goals conceded agg arranged by teams and matchweek
def get_goals_conceded(playing_stat):
    # Create a dictionary with team names as keys
    teams = {}
    for i in playing_stat.groupby('HomeTeam').mean().T.columns:
        teams[i] = []

    # the value corresponding to keys is a list containing the match location.
    for i in range(len(playing_stat)):
        ATGC = playing_stat.iloc[i]['FTHG']
        HTGC = playing_stat.iloc[i]['FTAG']
        teams[playing_stat.iloc[i].HomeTeam].append(HTGC)
        teams[playing_stat.iloc[i].AwayTeam].append(ATGC)

    # Create a dataframe for goals scored where rows are teams and cols are matchweek.
    GoalsConceded = pd.DataFrame(data=teams, index = [i for i in range(1,39)]).T
    GoalsConceded[0] = 0
    # Aggregate to get uptil that point
    for i in range(2,39):
        GoalsConceded[i] = GoalsConceded[i] + GoalsConceded[i-1]
```



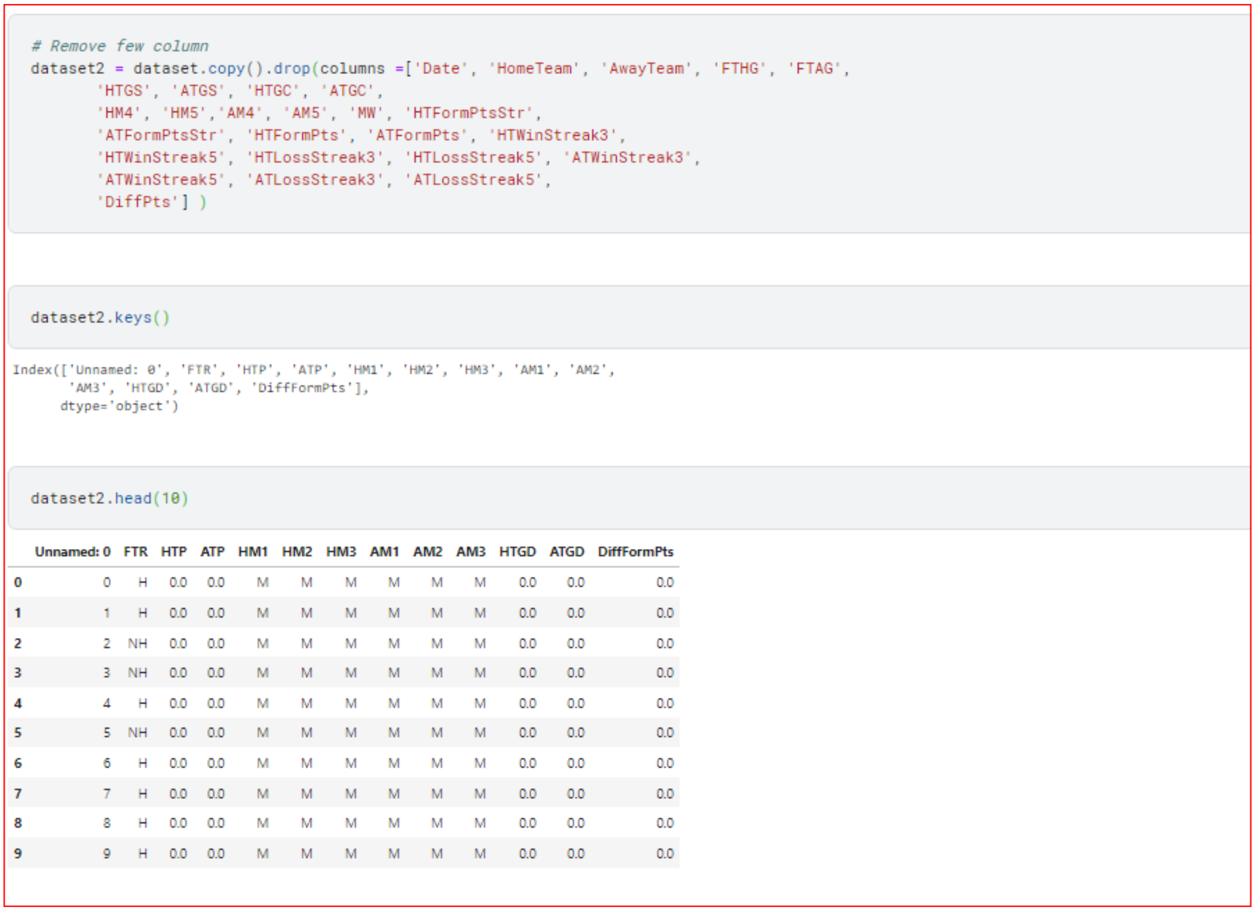
GET RESPECTIVE POINTS

```
def get_points(result):
    if result == 'W':
        return 3
    elif result == 'D':
        return 1
    else:
        return 0


def get_cuml_points(matchres):
    matchres_points = matchres.applymap(get_points)
    for i in range(2,39):
        matchres_points[i] = matchres_points[i] + matchres_points[i-1]

    matchres_points.insert(column =0, loc = 0, value = [0*i for i in range(20)])
    return matchres_points


def get_matchres(playing_stat):
    # Create a dictionary with team names as keys
    teams = {}
    for i in playing_stat.groupby('HomeTeam').mean().T.columns:
        teams[i] = []

    # the value corresponding to keys is a list containing the match result
    for i in range(len(playing_stat)):
        if playing_stat.iloc[i].FTR == 'H':
            teams[playing_stat.iloc[i].HomeTeam].append('W')
            teams[playing_stat.iloc[i].AwayTeam].append('L')
        elif playing_stat.iloc[i].FTR == 'A':
            teams[playing_stat.iloc[i].AwayTeam].append('W')
            teams[playing_stat.iloc[i].HomeTeam].append('L')
        else:
            teams[playing_stat.iloc[i].AwayTeam].append('D')
            teams[playing_stat.iloc[i].HomeTeam].append('D')

    return pd.DataFrame(data=teams, index = [i for i in range(1,39)]).T
```



FINAL DATAFRAME

```
playing_stat = pd.concat([playing_statistics_1,
                          playing_statistics_2,
                          playing_statistics_3,
                          playing_statistics_4,
                          playing_statistics_5,
                          playing_statistics_6,
                          playing_statistics_7,
                          playing_statistics_8,
                          playing_statistics_9,
                          playing_statistics_10,
                          playing_statistics_11,
                          playing_statistics_12,
                          playing_statistics_13,
                          playing_statistics_14,
                          playing_statistics_15,
                          playing_statistics_16,
                          playing_statistics_17,
                          playing_statistics_18
                          ], ignore_index=True)


# Gets the form points.
def get_form_points(string):
    sum = 0
    for letter in string:
        sum += get_points(letter)
    return sum

playing_stat['HTFormPtsStr'] = playing_stat['HM1'] + playing_stat['HM2'] + playing_stat['HM3'] + playing_stat['HM4'] + playing_stat['HM5']
playing_stat['ATFormPtsStr'] = playing_stat['AM1'] + playing_stat['AM2'] + playing_stat['AM3'] + playing_stat['AM4'] + playing_stat['AM5']

playing_stat['HTFormPts'] = playing_stat['HTFormPtsStr'].apply(get_form_points)
playing_stat['ATFormPts'] = playing_stat['ATFormPtsStr'].apply(get_form_points)
```



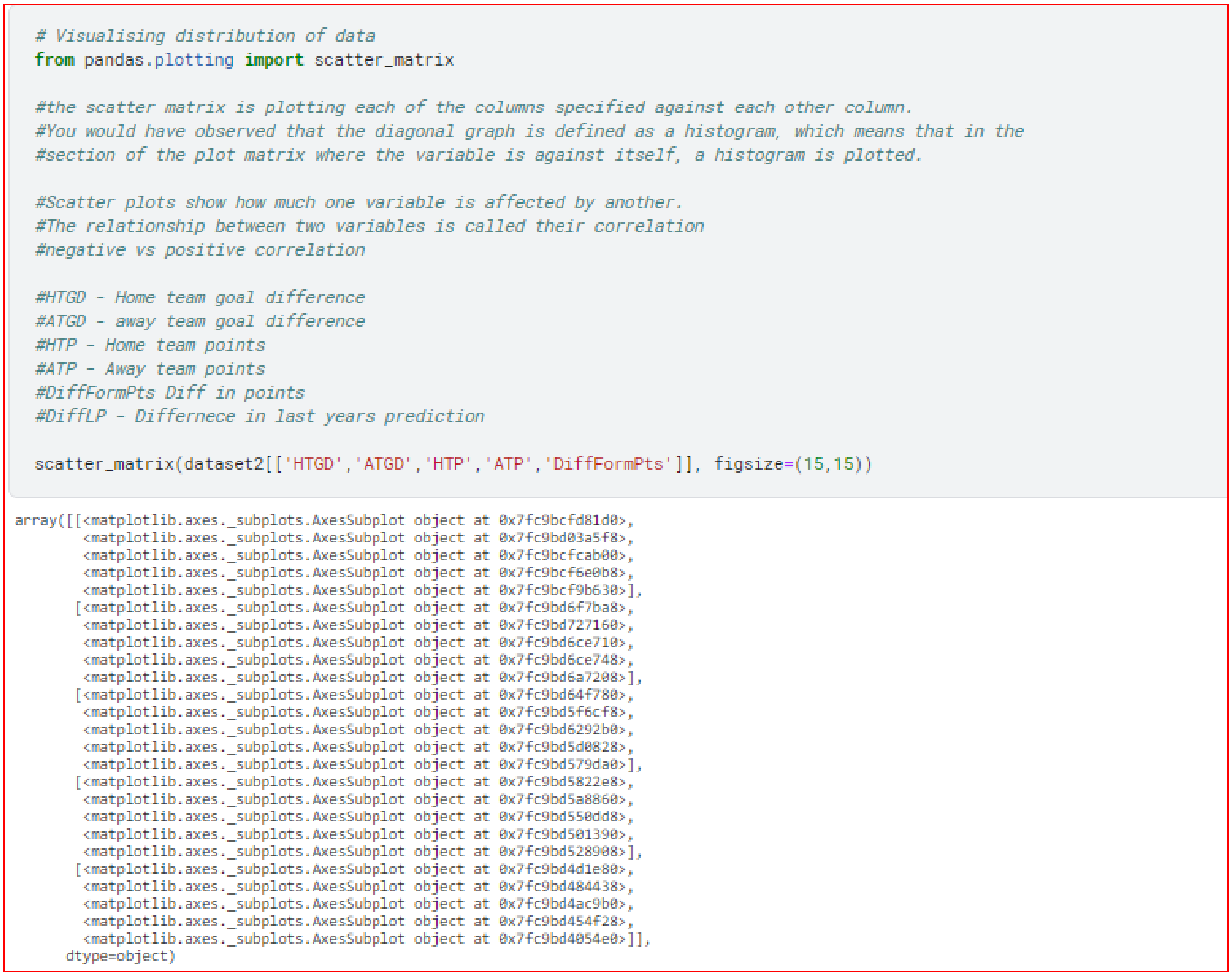
```
# Remove few column
dataset2 = dataset.copy().drop(columns =['Date', 'HomeTeam', 'AwayTeam', 'FTHG', 'FTAG',
        'HTGS', 'ATGS', 'HTGC', 'ATGC',
        'HM4', 'HM5','AM4', 'AM5', 'MW', 'HTFormPtsStr',
        'ATFormPtsStr', 'HTFormPts', 'ATFormPts', 'HTWinStreak3',
        'HTWinStreak5', 'HTLossStreak3', 'HTLossStreak5', 'ATWinStreak3',
        'ATWinStreak5', 'ATLossStreak3', 'ATLossStreak5',
        'DiffPts'] )
```

```
dataset2.keys()
```

```
Index(['Unnamed: 0', 'FTR', 'HTP', 'ATP', 'HM1', 'HM2', 'HM3', 'AM1', 'AM2',
       'AM3', 'HTGD', 'ATGD', 'DiffFormPts'],
      dtype='object')
```

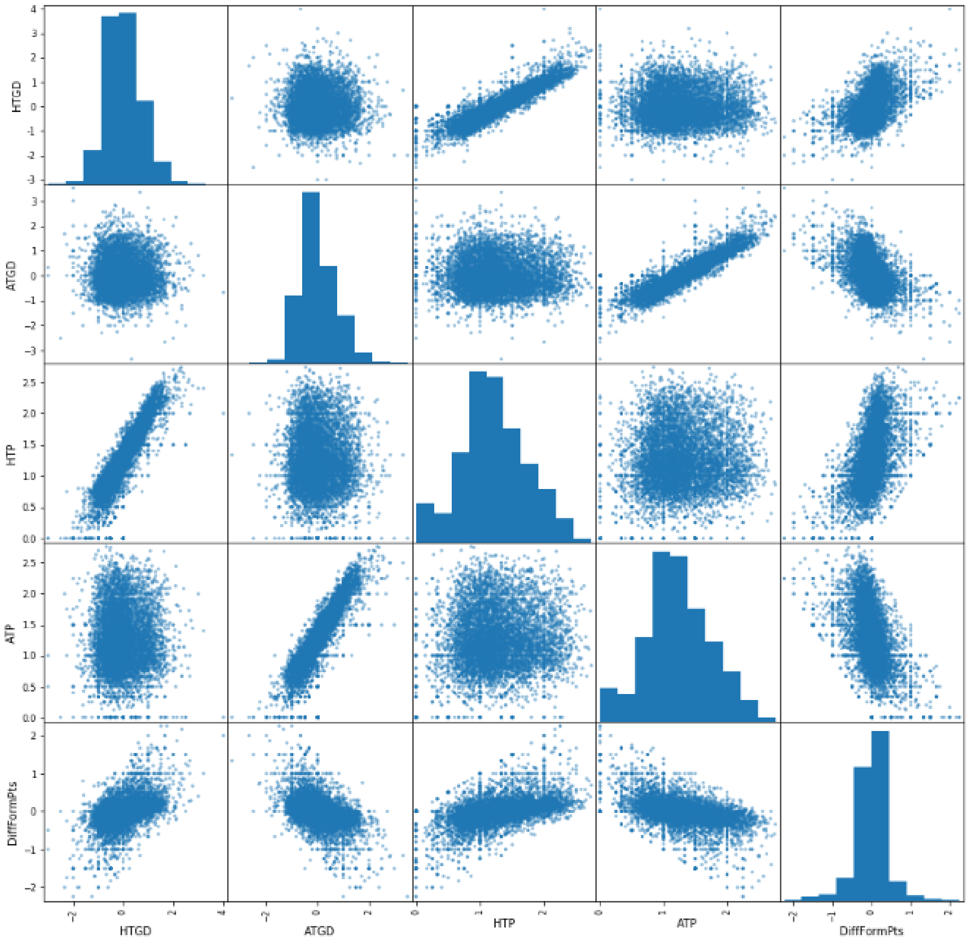
```
dataset2.head(10)
```

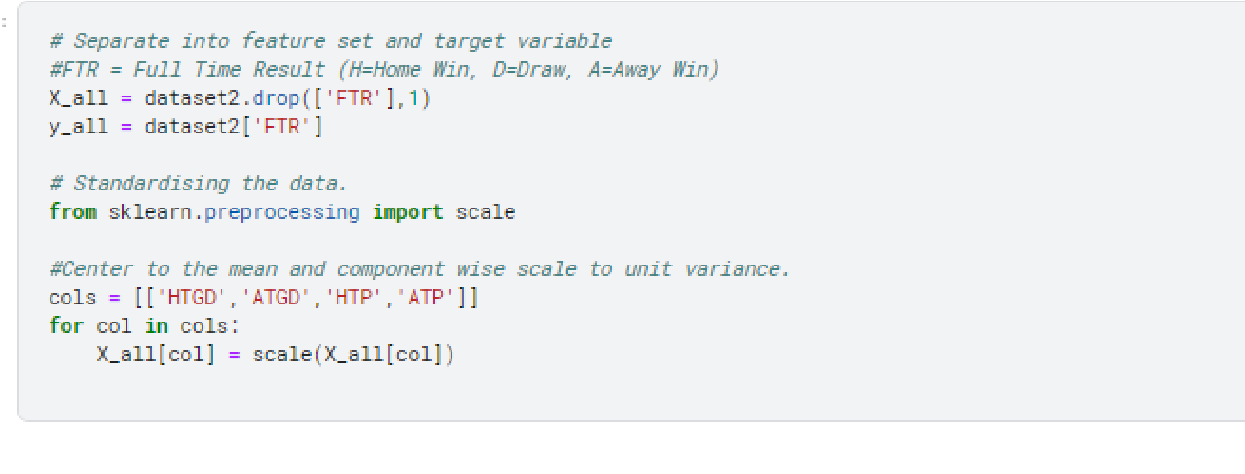
	Unnamed: 0	FTR	HTP	ATP	HM1	HM2	HM3	AM1	AM2	AM3	HTGD	ATGD	DiffFormPts
0	0	H	0.0	0.0	M	M	M	M	M	M	0.0	0.0	0.0
1	1	H	0.0	0.0	M	M	M	M	M	M	0.0	0.0	0.0
2	2	NH	0.0	0.0	M	M	M	M	M	M	0.0	0.0	0.0
3	3	NH	0.0	0.0	M	M	M	M	M	M	0.0	0.0	0.0
4	4	H	0.0	0.0	M	M	M	M	M	M	0.0	0.0	0.0
5	5	NH	0.0	0.0	M	M	M	M	M	M	0.0	0.0	0.0
6	6	H	0.0	0.0	M	M	M	M	M	M	0.0	0.0	0.0
7	7	H	0.0	0.0	M	M	M	M	M	M	0.0	0.0	0.0
8	8	H	0.0	0.0	M	M	M	M	M	M	0.0	0.0	0.0
9	9	H	0.0	0.0	M	M	M	M	M	M	0.0	0.0	0.0



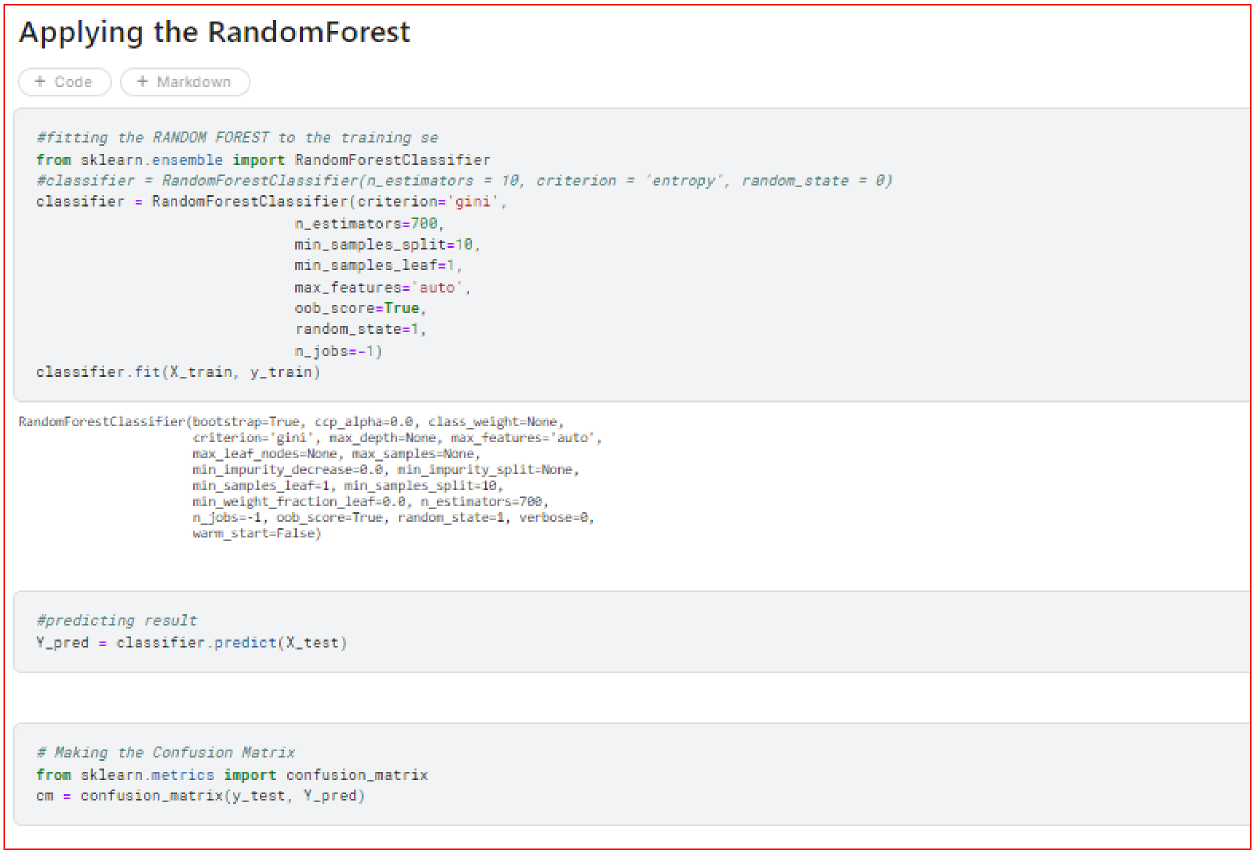
```python
# Visualising distribution of data
from pandas.plotting import scatter_matrix

#the scatter matrix is plotting each of the columns specified against each other column.
#You would have observed that the diagonal graph is defined as a histogram, which means that in the
#section of the plot matrix where the variable is against itself, a histogram is plotted.

#Scatter plots show how much one variable is affected by another.
#The relationship between two variables is called their correlation
#negative vs positive correlation

#HTGD - Home team goal difference
#ATGD - away team goal difference
#HTP - Home team points
#ATP - Away team points
#DiffFormPts Diff in points
#DiffLP - Differnece in last years prediction

scatter_matrix(dataset2[['HTGD','ATGD','HTP','ATP','DiffFormPts']], figsize=(15,15))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bcfd81d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd03a5f8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bcfcab00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bcf6e0b8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bcf9b630>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd6f7ba8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd727160>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd6ce710>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd6ce748>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd6a7208>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd64f780>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd5f6cf8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd6292b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd5d0828>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd579da0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd5822e8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd5a8860>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd550dd8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd501390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd528908>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd4d1e80>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd484438>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd4ac9b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd454f28>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc9bd4054e0>]],
      dtype=object)
```





```
# Separate into feature set and target variable
#FTR = Full Time Result (H=Home Win, D=Draw, A=Away Win)
X_all = dataset2.drop(['FTR'],1)
y_all = dataset2['FTR']

# Standardising the data.
from sklearn.preprocessing import scale

#Center to the mean and component wise scale to unit variance.
cols = [['HTGD','ATGD','HTP','ATP']]
for col in cols:
    X_all[col] = scale(X_all[col])
```



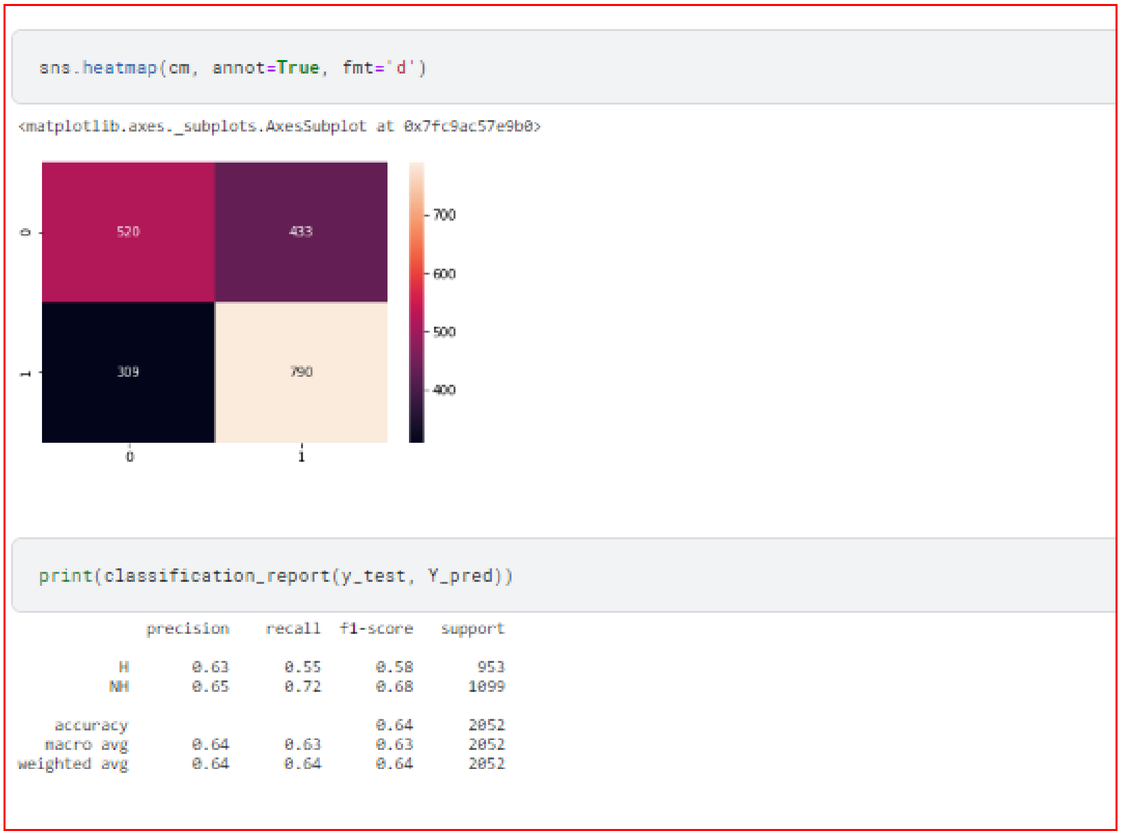
Applying the RandomForest

```
#fitting the RANDOM FOREST to the training se
from sklearn.ensemble import RandomForestClassifier
#classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier = RandomForestClassifier(criterion='gini',
                             n_estimators=700,
                             min_samples_split=10,
                             min_samples_leaf=1,
                             max_features='auto',
                             oob_score=True,
                             random_state=1,
                             n_jobs=-1)
classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=10,
                       min_weight_fraction_leaf=0.0, n_estimators=700,
                       n_jobs=-1, oob_score=True, random_state=1, verbose=0,
                       warm_start=False)
```

```
#predicting result
Y_pred = classifier.predict(X_test)
```

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, Y_pred)
```

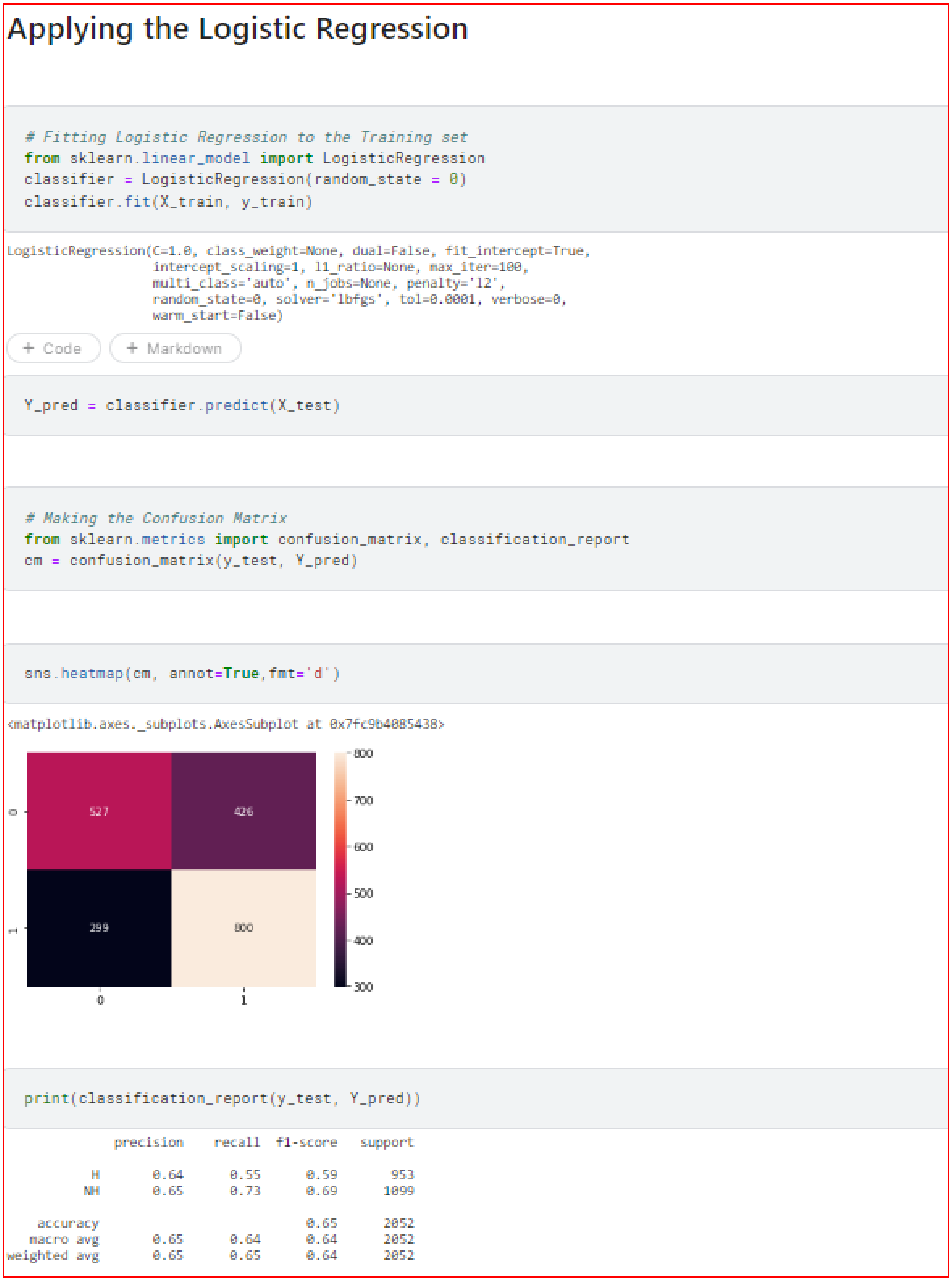


```
sns.heatmap(cm, annot=True, fmt='d')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc9ac57e9b0>
```

```
print(classification_report(y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           H       0.63      0.55      0.58       953
          NH       0.65      0.72      0.68      1099

    accuracy                           0.64      2052
   macro avg       0.64      0.63      0.63      2052
weighted avg       0.64      0.64      0.64      2052
```



Applying the Logistic Regression

```
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
Y_pred = classifier.predict(X_test)
```

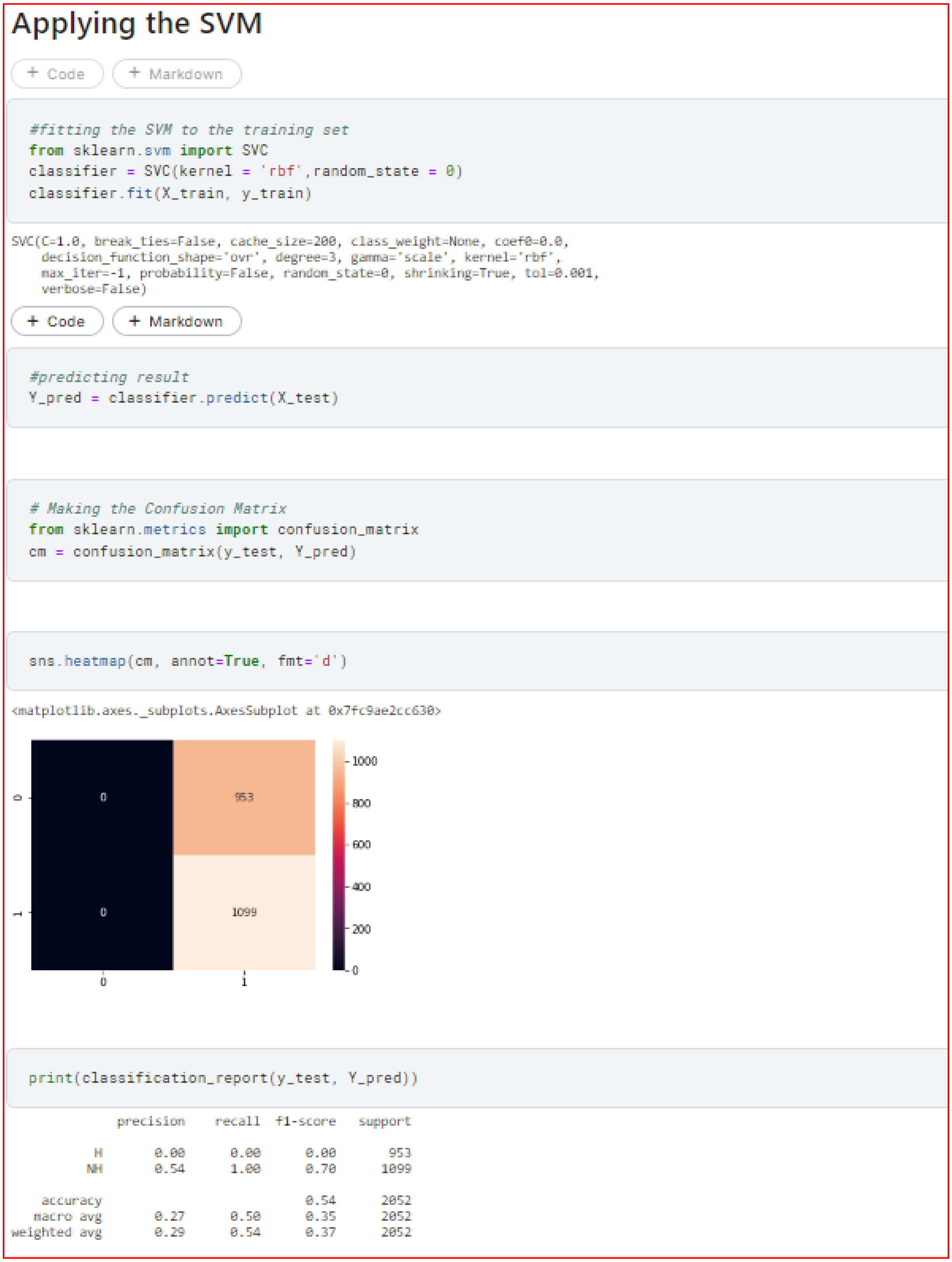
```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, classification_report
cm = confusion_matrix(y_test, Y_pred)
```

```
sns.heatmap(cm, annot=True,fmt='d')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc9b4085438>
```

```
print(classification_report(y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           H       0.64      0.55      0.59       953
          NH       0.65      0.73      0.69      1099

    accuracy                           0.65      2052
   macro avg       0.65      0.64      0.64      2052
weighted avg       0.65      0.65      0.64      2052
```



Applying the SVM

```
#fitting the SVM to the training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf',random_state = 0)
classifier.fit(X_train, y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=0, shrinking=True, tol=0.001,
    verbose=False)
```

```
#predicting result
Y_pred = classifier.predict(X_test)
```

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, Y_pred)
```

```
sns.heatmap(cm, annot=True, fmt='d')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc9ae2cc630>
```

```
print(classification_report(y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           H       0.00      0.00      0.00       953
          NH       0.54      1.00      0.70      1099

    accuracy                           0.54      2052
   macro avg       0.27      0.50      0.35      2052
weighted avg       0.29      0.54      0.37      2052
```

Conclusion -

Probably little best!!!!!!!!

Accuracy is not so good but it can be improved.

Actually it only depends upon past year match dataset,we can improve the accuracy by putting twitter data related to match, sentiment analysis, chances of a player to play a specific match,player performance in a recent series,etc..