Player_Rating _Prediction

March 17, 2019

```
In [1]: # Importing basic libraries for data preprocessing and visualization

        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import sqlite3
```

```
In [2]: # To read the dataset we have to create a connection to sqlite3

        cnx = sqlite3.connect('database.sqlite')
        df = pd.read_sql_query("SELECT * FROM Player_attributes", cnx)
        df.head()
```

```
Out[2]:    id  player_fifa_api_id  player_api_id                 date  overall_rating  \
        0   1              218353         505942  2016-02-18 00:00:00            67.0
        1   2              218353         505942  2015-11-19 00:00:00            67.0
        2   3              218353         505942  2015-09-21 00:00:00            62.0
        3   4              218353         505942  2015-03-20 00:00:00            61.0
        4   5              218353         505942  2007-02-22 00:00:00            61.0

           potential preferred_foot attacking_work_rate defensive_work_rate  crossing  \
        0       71.0          right              medium              medium      49.0
        1       71.0          right              medium              medium      49.0
        2       66.0          right              medium              medium      49.0
        3       65.0          right              medium              medium      48.0
        4       65.0          right              medium              medium      48.0

              ...        vision  penalties  marking  standing_tackle  sliding_tackle  \
        0     ...          54.0       48.0     65.0             69.0            69.0
        1     ...          54.0       48.0     65.0             69.0            69.0
        2     ...          54.0       48.0     65.0             66.0            69.0
        3     ...          53.0       47.0     62.0             63.0            66.0
        4     ...          53.0       47.0     62.0             63.0            66.0

           gk_diving  gk_handling  gk_kicking  gk_positioning  gk_reflexes
        0        6.0         11.0        10.0             8.0          8.0
        1        6.0         11.0        10.0             8.0          8.0
```

```
2        6.0         11.0        10.0             8.0          8.0
3        5.0         10.0         9.0             7.0          7.0
4        5.0         10.0         9.0             7.0          7.0

[5 rows x 42 columns]
```

```
In [3]: # Let's see what are the columns we have.

        [(f"column {i+1} : {column}") for i,column in enumerate(df.columns)]
```

```
Out[3]: ['column 1 : id',
         'column 2 : player_fifa_api_id',
         'column 3 : player_api_id',
         'column 4 : date',
         'column 5 : overall_rating',
         'column 6 : potential',
         'column 7 : preferred_foot',
         'column 8 : attacking_work_rate',
         'column 9 : defensive_work_rate',
         'column 10 : crossing',
         'column 11 : finishing',
         'column 12 : heading_accuracy',
         'column 13 : short_passing',
         'column 14 : volleys',
         'column 15 : dribbling',
         'column 16 : curve',
         'column 17 : free_kick_accuracy',
         'column 18 : long_passing',
         'column 19 : ball_control',
         'column 20 : acceleration',
         'column 21 : sprint_speed',
         'column 22 : agility',
         'column 23 : reactions',
         'column 24 : balance',
         'column 25 : shot_power',
         'column 26 : jumping',
         'column 27 : stamina',
         'column 28 : strength',
         'column 29 : long_shots',
         'column 30 : aggression',
         'column 31 : interceptions',
         'column 32 : positioning',
         'column 33 : vision',
         'column 34 : penalties',
         'column 35 : marking',
         'column 36 : standing_tackle',
         'column 37 : sliding_tackle',
         'column 38 : gk_diving',
         'column 39 : gk_handling',
         'column 40 : gk_kicking',
         'column 41 : gk_positioning',
         'column 42 : gk_reflexes']
```

```
In [4]: #Create a new dataframe after dropping some columns which are not useful to predict pl
        soccer_data = df.drop(["id", "player_fifa_api_id", "player_api_id", "date"], axis = 1)

        #Check whether there are duplicates entries present or not
        soccer_data.duplicated().any()
```

```
Out[4]: True
```

```
In [5]: #Drop duplicates entries from soccer_data dataframe
        soccer_data.drop_duplicates(inplace=True)

        #check dataframe shape after dropping duplicate entries
        soccer_data.shape
```

```
Out[5]: (138440, 38)
```

```
In [6]: # Steps to handle missing data
        # As we have quite so many observations, we can drop few Nan value rows will not impac

        soccer_data = soccer_data.dropna()
        soccer_data.isnull().any().any(),soccer_data.shape
```

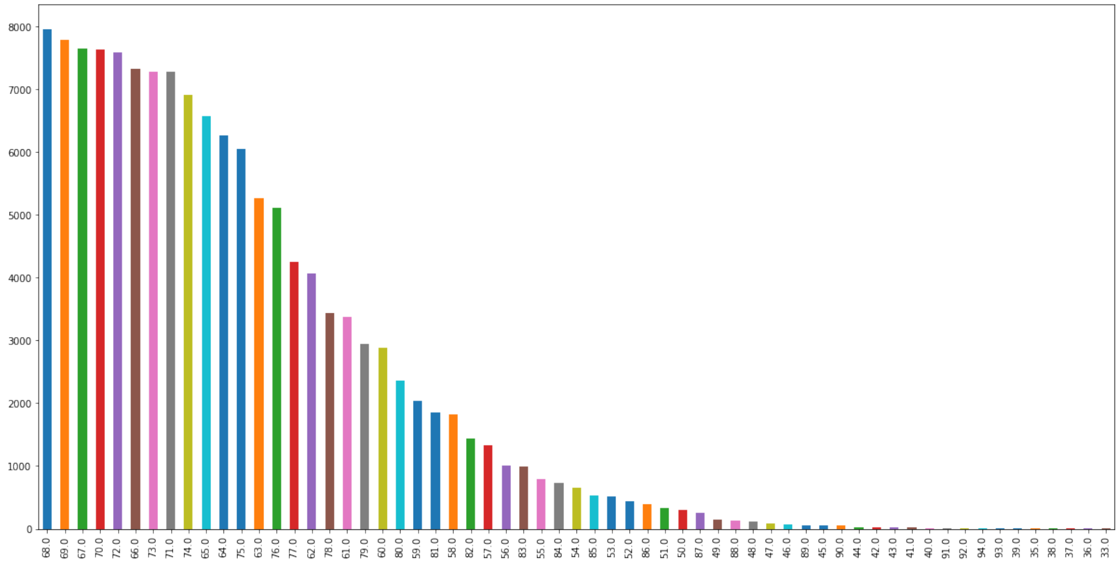
```
Out[6]: (False, (136284, 38))
```

```
In [7]: # Categorical Variables into dummies

        soccer_data = pd.get_dummies(soccer_data)
        soccer_data.shape
```

```
Out[7]: (136284, 63)
```

```
In [8]: #Visualize column overall_rating of the dataframe

        soccer_data['overall_rating'].value_counts().plot(kind = 'bar',figsize = (20,10))
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x124f912bef0>
```



```
In [12]: # Splitting the data into dependent and independent variables

         X = soccer_data.drop(['overall_rating'],axis = 1)
         y = np.array(soccer_data['overall_rating'])
         y

Out[12]: array([67., 62., 61., ..., 77., 78., 80.])

In [13]: # splitting the data into train test parts

         from sklearn.model_selection import train_test_split

         X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.33,random_state=10

In [16]: # Applying XGBoost

         import xgboost as xgb

         from sklearn.metrics import accuracy_score


         Boosting = xgb.XGBRegressor()
         Boosting.fit(X_train,y_train)
         y_pred = Boosting.predict(X_test)
         score = accuracy_score(y_test,y_pred.round())
         score

Out[16]: 0.26875528082892336
```

```
In [17]: # Applying Decesion Tree

         from sklearn.tree import DecisionTreeRegressor
         dtr = DecisionTreeRegressor(min_samples_split=10, random_state=55)
         dtr.fit(X_train,y_train)
         y_pred = dtr.predict(X_test)

         score = accuracy_score(y_test,y_pred.round())
         score

Out[17]: 0.4445235024680927
```

```
In [20]: # Applying Linear Regression

         from sklearn.linear_model import LinearRegression

         lr = LinearRegression()
         lr.fit(X_train,y_train)
         y_pred = lr.predict(X_test)
         score = accuracy_score(y_test,y_pred.round())
         score

Out[20]: 0.1664072575265709
```

```
In [22]: # Applying Random Forest Model

         from sklearn.ensemble import RandomForestRegressor
         rfr = RandomForestRegressor()
         rfr.fit(X_train,y_train)
         y_pred = rfr.predict(X_test)
         score = accuracy_score(y_test,y_pred.round())
         score

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


Out[22]: 0.5088495575221239
```

1 As we are getting a better accuracy with Random Forest Regressor,we should go with the random fprest model