In [ ]:

```
#*******************************************************************************************
#
#                   Project 2 : Prediction of Overall Rating of the Soccer Player
#
#
#*******************************************************************************************


# Step0  : importing libraries

import numpy as np
import pandas as pd
import sqlite3

from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.preprocessing import LabelEncoder

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from math import sqrt

# Step1  : Loading Data from SQl DataBase

# Creating a connection to the sqlite DataBase and fetching Data

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


# Data from Player_Attributes Table
print(" The Data from the Player Attributes Table giving information of Soccer Player")
print('-'*80)
print(df.head(2))

print("The attributes of the Player to be compared with are :\n ",'-'*80)
print(df.columns)

```

In [ ]:

```
# Step2 : Data Preprocessing

print("The number of rows and columns\n")
print('-'*80)
print(df.shape)

# Data  has 183978 rows and 42 columns

# Understanding the data
print('-'*80)
print(df.describe())
# all ids and Date columns have high std  , so we can drop them

print('-'*80)
print(df.info())



# check for if any columns have nulls
print(" Checking for nulls in any of the features \n",'-'*80)

print('-'*80)
print((df.isnull().sum()/df.shape[0])*100)

# since the percentage of nulls is less than total number of rows , we can drop the nulls
df = df.dropna()

print("After Dropping the nulls check for the count of nulls and the shape \n",'-'*80)

print((df.isnull().sum()/df.shape[0])*100)
print('-'*80)

print("Number of rows after dropping the nulls \n",'-'*80)
print(df.shape)
```

In [ ]:

```
#Step 3  :  Data Analysing

# From the Data we see there are Categorical Fields like attacking work rate , Defensive Work Rate
#'preferred_foot', 'attacking_work_rate','defensive_work_rate'
# finding unique values for each
print(df.preferred_foot.unique())
print(df.attacking_work_rate.unique())
print(df.defensive_work_rate.unique())

print(" Converting the Categorical attributes to numerical values \n",'-'*80)

number = LabelEncoder()
df['preferred_foot'] = number.fit_transform(df['preferred_foot'].astype('str'))
df['attacking_work_rate'] = number.fit_transform(df['attacking_work_rate'].astype('str'))
df['defensive_work_rate'] = number.fit_transform(df['defensive_work_rate'].astype('str'))

print(df.preferred_foot.unique())
print(df.attacking_work_rate.unique())
print(df.defensive_work_rate.unique())

# Dropping the ID fields
df.drop(['id','player_fifa_api_id','player_api_id', 'date'], axis=1,inplace=True)

print(" The Data after dropping the ids \n",'-'*80)
print(df.head())


```

In [ ]:

```
#Step 4 : Visualising the Data by using pairplot with respect to overall rating

df.head()

features = df.columns

target = 'overall_rating'
count = len(df.columns)

# as lots of records , pairplot done only for sample size

i = 1
while i < count:
    sns.pairplot(df.sample(1000,random_state=2),x_vars= features[i:i+5] ,y_vars = target, aspect=1,kind='reg')
    i = i + 5


```

In [ ]:

```
# Step 5 : Splitting the Data into Training and Test Set

# from the Visualisation we see that colummns
# 'preferred_foot', 'attacking_work_rate','defensive_work_rate' has no much effect on the overall rating
# so we can ignore them
# also by checking for fields 'sliding_tackle', 'gk_handling','gk_reflexes' not much effect on accuracy
# by inclusion, so dropping it

features = ['potential', 'crossing', 'finishing', 'heading_accuracy',
       'short_passing', 'volleys', 'dribbling', 'curve', 'free_kick_accuracy',
       'long_passing', 'ball_control', 'acceleration', 'sprint_speed',
       'agility', 'reactions', 'balance', 'shot_power', 'jumping', 'stamina',
       'strength', 'long_shots', 'aggression','interceptions', 'positioning',
        'vision', 'penalties', 'marking' ,'standing_tackle','gk_diving','gk_kicking',
            'gk_positioning']

#, ,'sliding_tackle', 'gk_handling',,'gk_reflexes',,

target = 'overall_rating'

X = df[features]
y = df[target]

print(X.shape)
print(y.shape)

y = y.values.reshape(-1,1)


X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=2)

print(" The Training and Test size after splitting \n",'-'*80)
# size of the Training and Test set
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

In [ ]:

```
# Step 6  : Model /instantiating the Linear Regression Classifier
lm = LinearRegression()

# fitting / Training data with the training set data
lm.fit(X_train,y_train)

# Predicting the prices for the test set
pred_test = lm.predict(X_test) # test case prediction

print(" Following are the predicted price values for the test set , after fitting the model with train set\n",'-'*80)
print(pred_test)


```

In [ ]:

```
#Step 7 :  Visualize the predicted and observed ratings

# first, plot the observed data

plt.scatter(y_test, pred_test)
plt.xlabel('Observed Rating')
plt.ylabel('Predicted Overall Rating')
plt.title('Actual vs. Predicted Overall Rating')
plt.show()

```

In [ ]:

```
# Step 8 : Model Evaluation
#from sklearn.metrics import mean_squared_error,r2_score



RMSE = sqrt(mean_squared_error(y_test,pred_test))
print("The Root Mean squared error for Model is : %.4f"%RMSE)

R2square = r2_score(y_test,pred_test)

print("The R2 score for the model is  : %.4f"%R2square)

acc = round(R2square,2)*100

print("\n This Model can predict the performance of the Player at %.2f percent accuracy" %acc)

```