1/19/2018 ML Project(Updated) (1)

http://localhost:8888/notebooks/20th%20Januray/ML%20Project(Updated)%20(1).ipynb# 1/4

Predicting players rating

In this project you are going to predict the overall rating of soccer player based on their attributes

such as 'crossing', 'finishing etc.

The dataset you are going to use is from European Soccer Database

(https://www.kaggle.com/hugomathien/soccer) has more than 25,000 matches and more than

10,000 players for European professional soccer seasons from 2008 to 2016.

Download the data in the same folder and run the following commmand to get it in the environment

About the Dataset

The ultimate Soccer database for data analysis and

machine learning

The dataset comes in the form of an SQL database and contains statistics of about 25,000 football

matches, from the top football league of 11 European Countries. It covers seasons from 2008 to

2016 and contains match statistics (i.e: scores, corners, fouls etc...) as well as the team formations,

with player names and a pair of coordinates to indicate their position on the pitch.

+25,000 matches

+10,000 players

11 European Countries with their lead championship

Seasons 2008 to 2016

Players and Teams' attributes\* sourced from EA Sports' FIFA video game series, including the

weekly updates

Team line up with squad formation (X, Y coordinates)

Betting odds from up to 10 providers

Detailed match events (goal types, possession, corner, cross, fouls, cards etc...) for +10,000

matches

The dataset also has a set of about 35 statistics for each player, derived from EA Sports' FIFA video

games. It is not just the stats that come with a new version of the game but also the weekly

updates. So for instance if a player has performed poorly over a period of time and his stats get

impacted in FIFA, you would normally see the same in the dataset.

Python skills required to complete this project

SQL:

The data is in SQL database so students need to retrive using query language. They also need to

know how to connect SQL database woth python. The library we are using for this in 'sqlite3'.

1/19/2018 ML Project(Updated) (1)

http://localhost:8888/notebooks/20th%20Januray/ML%20Project(Updated)%20(1).ipynb# 2/4

SQLite3 can be integrated with Python using sqlite3 module, which was written by Gerhard Haring.

It provides an SQL interface compliant with the DB-API 2.0 specification described by PEP 249. You

do not need to install this module separately because it is shipped by default along with Python

version 2.5.x onwards.

To use sqlite3 module, you must first create a connection object that represents the database and

then optionally you can create a cursor object, which will help you in executing all the SQL

statements.

Pandas:

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use

data structures and data analysis tools for the Python programming language. Python with Pandas

is used in a wide range of fields including academic and commercial domains including finance,

economics, Statistics, analytics, etc.In this tutorial, we will learn the various features of Python

Pandas and how to use them in practice.

Scikit Learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent

interface in Python.

The library is built upon the SciPy (Scientific Python) that must be installed before you can use

scikit-learn. This stack that includes:

NumPy: Base n-dimensional array package

SciPy: Fundamental library for scientific computing

Matplotlib: Comprehensive 2D/3D plotting

IPython: Enhanced interactive console

Sympy: Symbolic mathematics

Pandas: Data structures and analysis

Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides

learning algorithms and is named scikit-learn.

The vision for the library is a level of robustness and support required for use in production systems.

This means a deep focus on concerns such as easy of use, code quality, collaboration,

documentation and performance.

Machine Learning skills required to complete the

project

Supervised learning

Supervised learning deals with learning a function from available training data. A supervised

learning algorithm analyzes the training data and produces an inferred function, which can be used

for mapping new examples.

1/19/2018 ML Project(Updated) (1)

http://localhost:8888/notebooks/20th%20Januray/ML%20Project(Updated)%20(1).ipynb# 3/4

Regression

Regression is a parametric technique used to predict continuous (dependent) variable given a set of

independent variables. It is parametric in nature because it makes certain assumptions (discussed

next) based on the data set. If the data set follows those assumptions, regression gives incredible

results.

Model evaluation

Student must know how to judge a model on unseen data. What metric to select to judge the

performance

Let's get started.....

Import Libraries

In [1]:

Read Data from the Database into pandas

In [2]:

import sqlite3

import pandas as pd

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

from math import sqrt

*# Create your connection.*

cnx = sqlite3.connect('database.sqlite')

df = pd.read\_sql\_query("SELECT \* FROM Player\_Attributes", cnx)

1/19/2018 ML Project(Updated) (1)

http://localhost:8888/notebooks/20th%20Januray/ML%20Project(Updated)%20(1).ipynb# 4/4

In [3]:

In [ ]:

Out[3]: id player\_fifa\_api\_id player\_api\_id date overall\_rating potential preferred\_foot attacking\_w

0 1 218353 505942

2016-

02-18

00:00:00

67.0 71.0 right

1 2 218353 505942

2015-

11-19

00:00:00

67.0 71.0 right

2 3 218353 505942

2015-

09-21

00:00:00

62.0 66.0 right

3 4 218353 505942

2015-

03-20

00:00:00

61.0 65.0 right

4 5 218353 505942

2007-

02-22

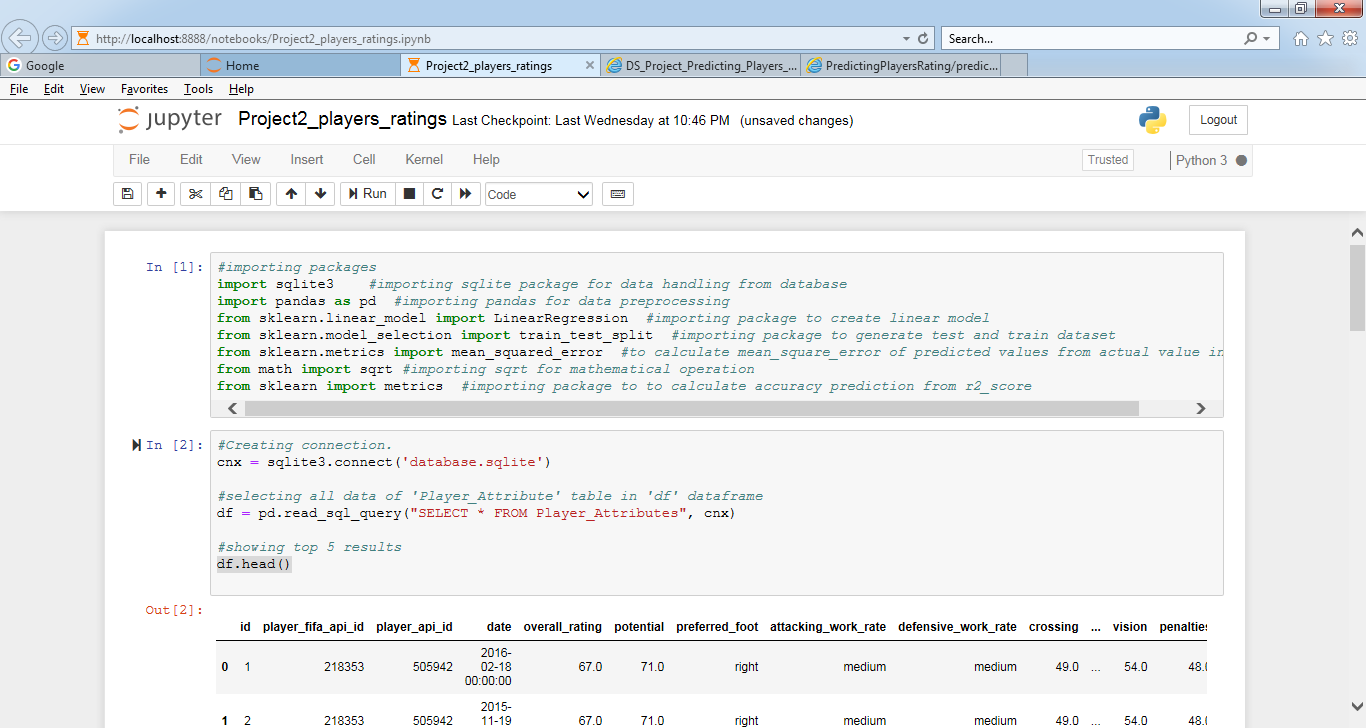
00:00:00

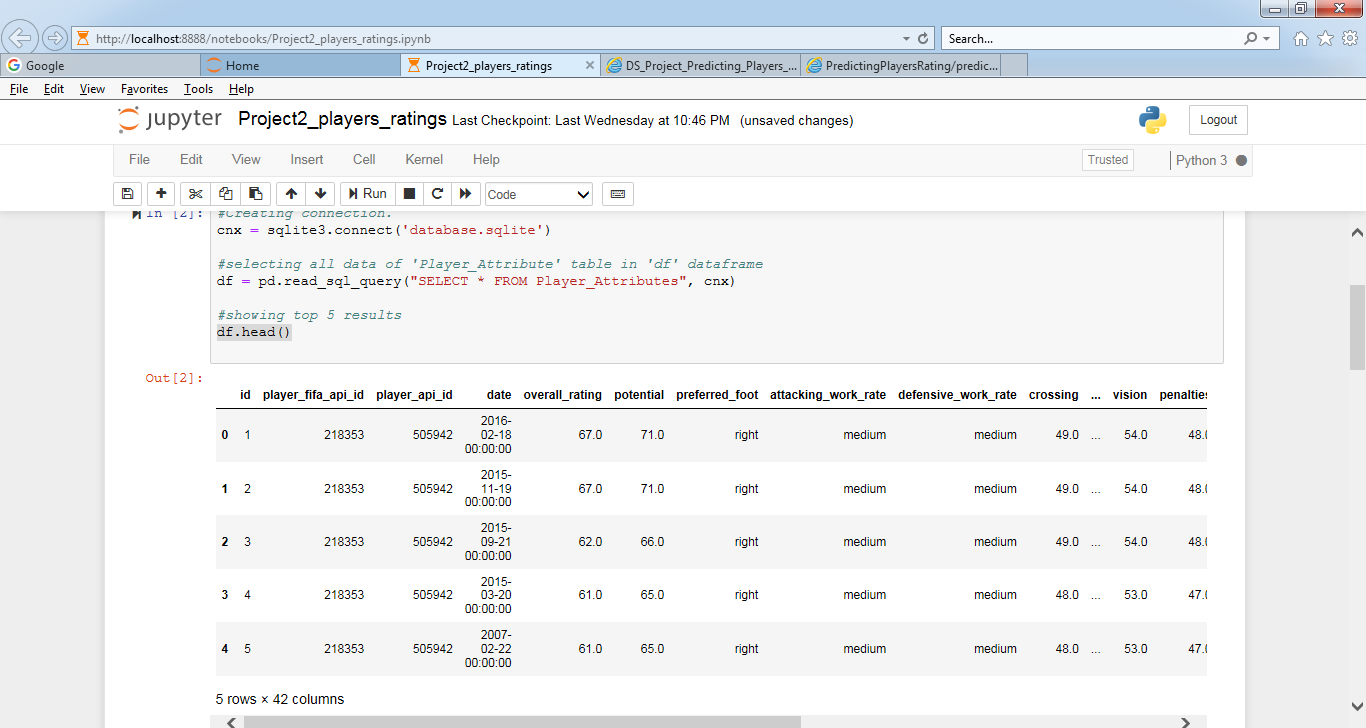
61.0 65.0 right

5 rows × 42 columns

df.head()

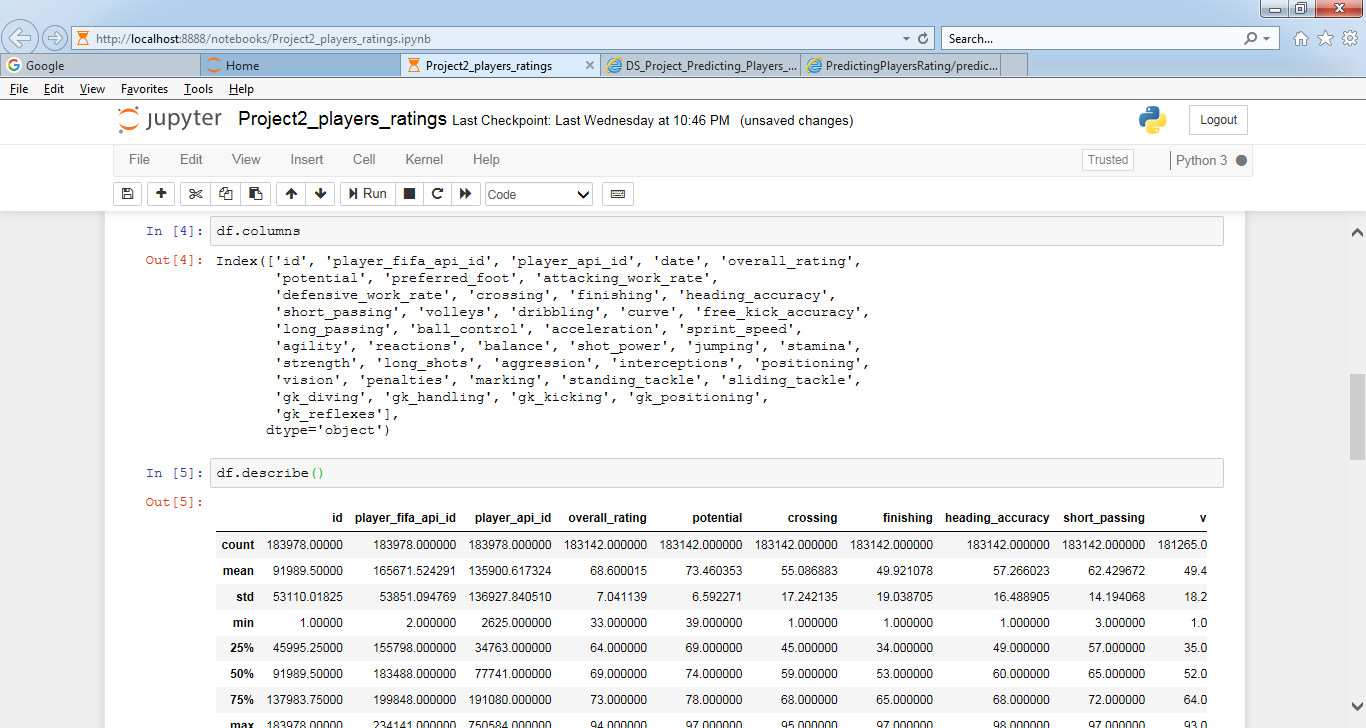
Answer:





df.columns

df.describe()



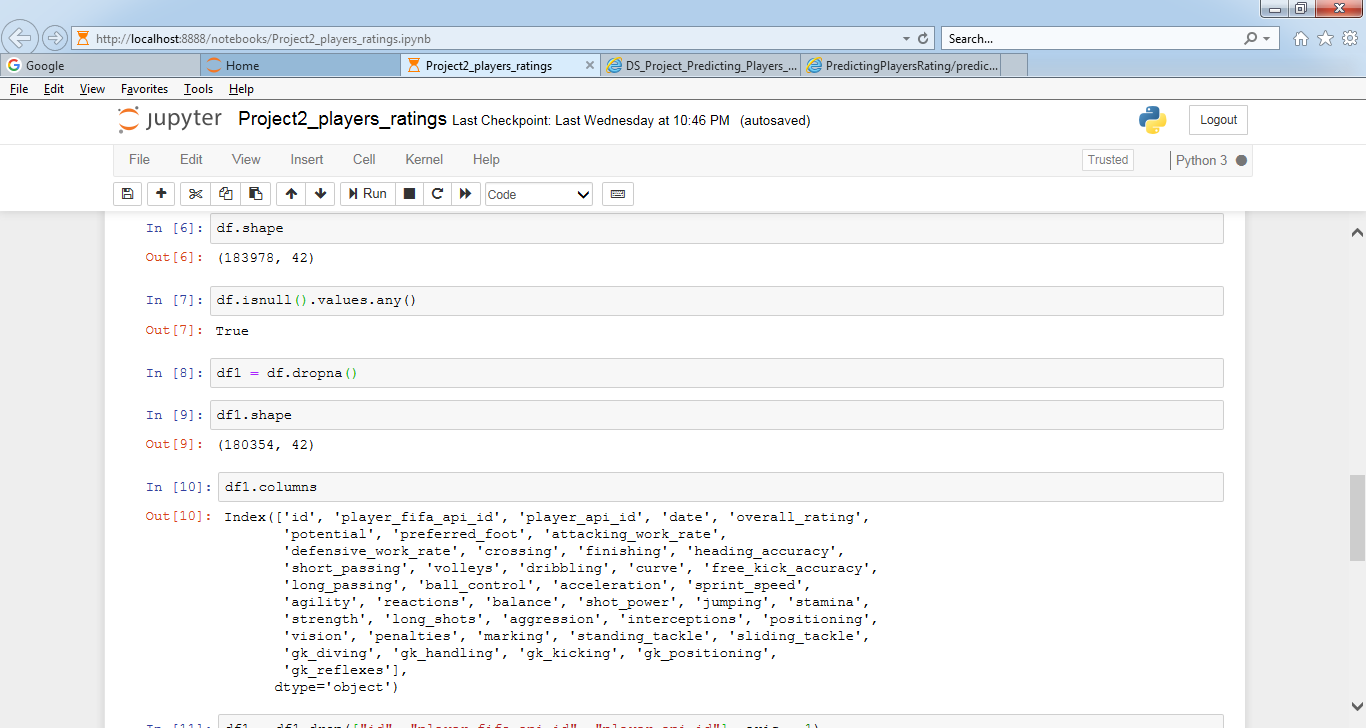
df.shape

df.isnull().values.any()

df1 = df.dropna()

df1.shape

df1.columns



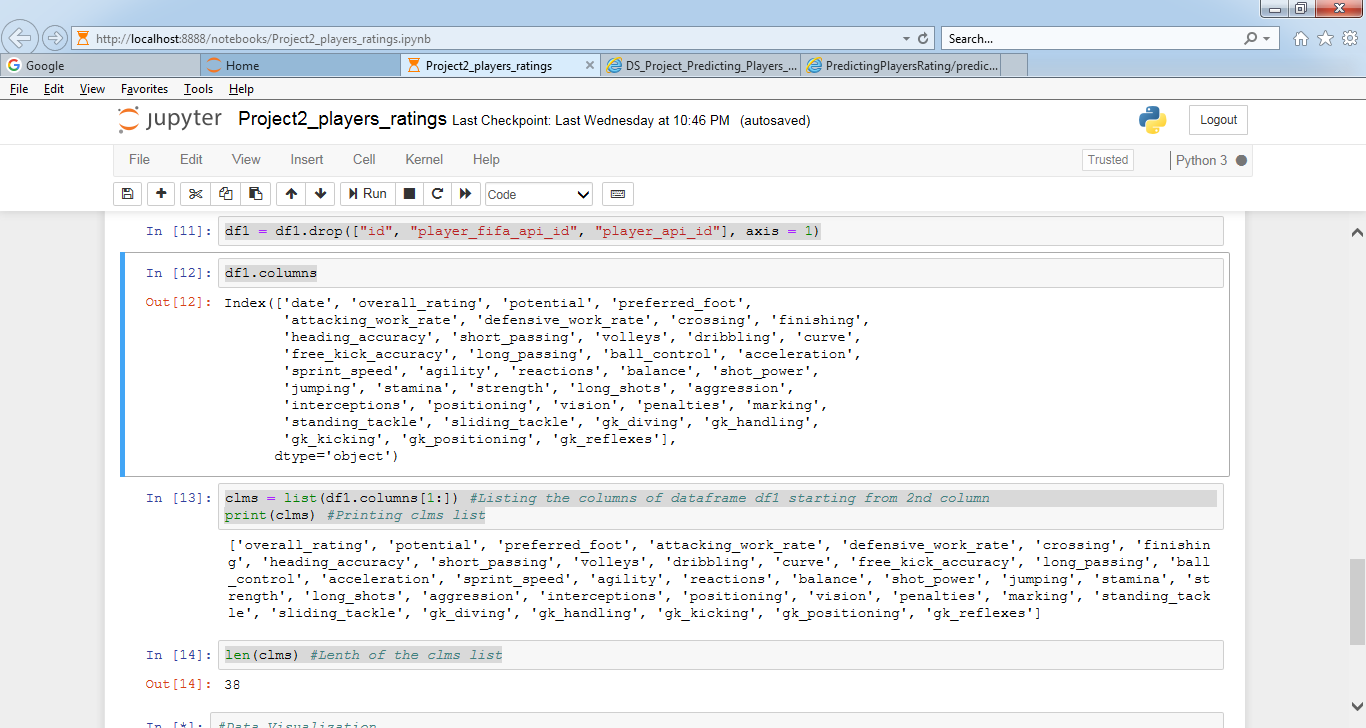
df1 = df1.drop(["id", "player\_fifa\_api\_id", "player\_api\_id"], axis = 1)

df1.columns

clms = list(df1.columns[1:]) #Listing the columns of dataframe df1 starting from 2nd column

print(clms) #Printing clms list

len(clms) #Lenth of the clms list



#Data Visualization

import matplotlib.pyplot as plt

fig, axes = plt.subplots(10, 4, figsize=(16, 12))

for i,ax in enumerate(axes.flat):

if i < len(clms):

ax.hist(df1[clms[i]])

ax.set\_title(clms[i])

plt.tight\_layout()

plt.show()

import matplotlib.pyplot as plt

fig, axes = plt.subplots(10, 4, figsize=(16, 12))

for i,ax in enumerate(axes.flat):

if i < len(clms)-1:

ax.scatter(df1[clms[i+1]], df1[clms[0]])

ax.set\_title(clms[i+1])

plt.tight\_layout()

plt.show()

axes[0,0].hist(df1[clms[0]])

plt.hist(df1["preferred\_foot"]) #Histogram for preferred\_foot

#Correlation Matrix

import seaborn as sns

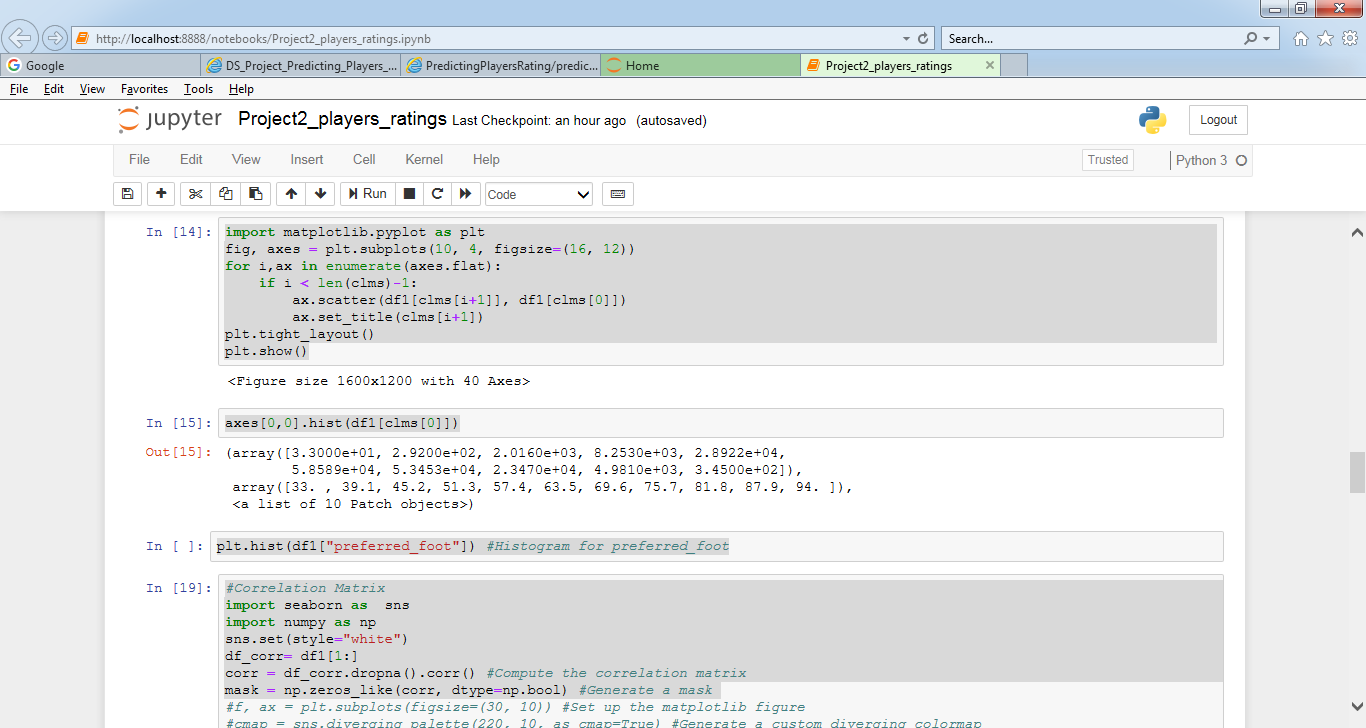
import numpy as np

sns.set(style="white")

df\_corr= df1[1:]

corr = df\_corr.dropna().corr() #Compute the correlation matrix

mask = np.zeros\_like(corr, dtype=np.bool) #Generate a mask



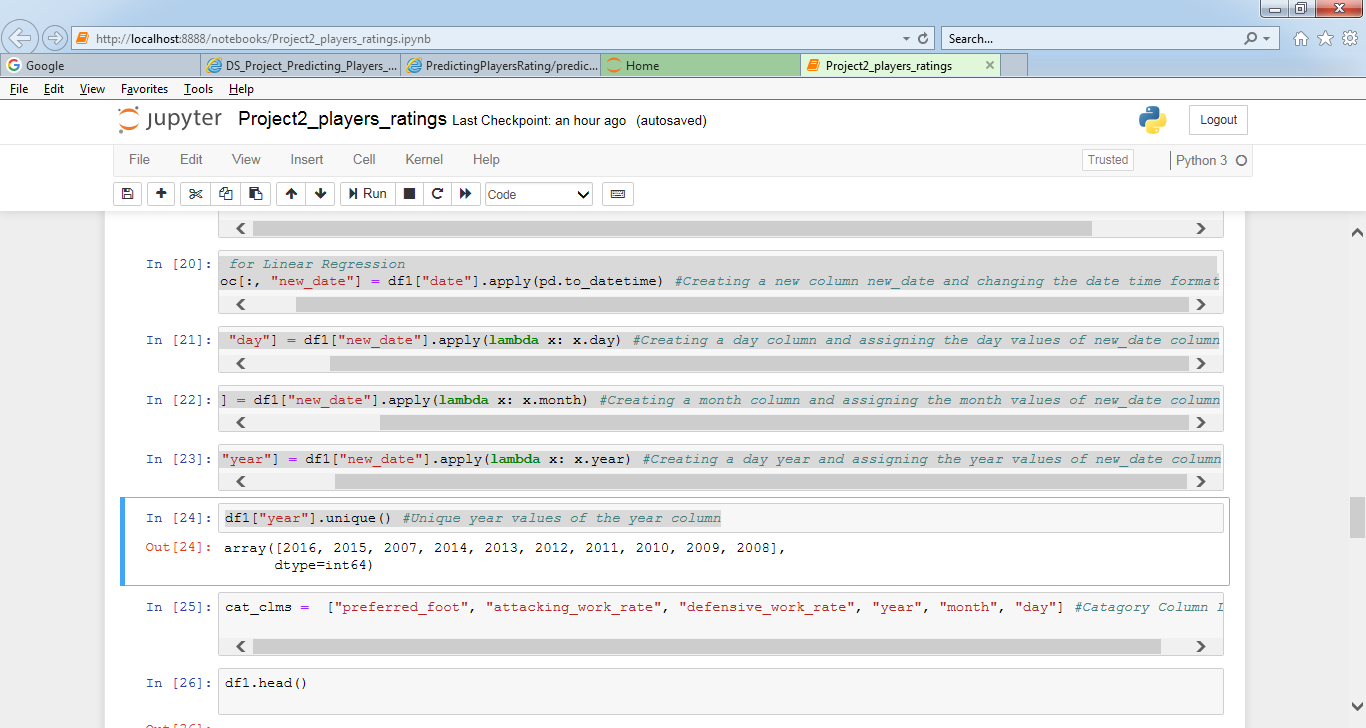
#Data for Linear Regression

df1.loc[:, "new\_date"] = df1["date"].apply(pd.to\_datetime) #Creating a new column new\_date and changing the date time formatdf1.loc[:, "day"] = df1["new\_date"].apply(lambda x: x.day) #Creating a day column and assigning the day values of new\_date column

df1.loc[:, "month"] = df1["new\_date"].apply(lambda x: x.month) #Creating a month column and assigning the month values of new\_date column

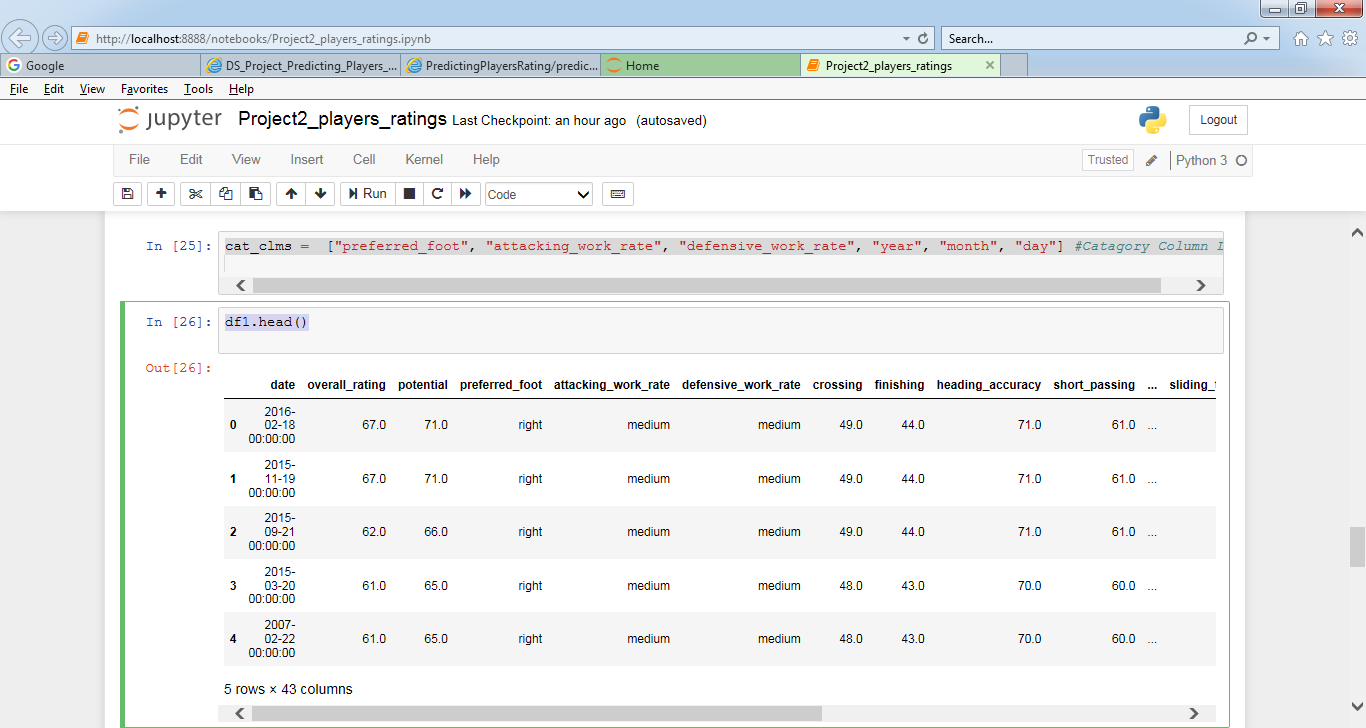
df1.loc[:, "year"] = df1["new\_date"].apply(lambda x: x.year) #Creating a day year and assigning the year values of new\_date column

df1["year"].unique() #Unique year values of the year column



cat\_clms = ["preferred\_foot", "attacking\_work\_rate", "defensive\_work\_rate", "year", "month", "day"] #Catagory Column List

df1.head()



df1 = df1.drop(["date", "new\_date"], axis = 1) #Dropping the date and new\_date columns

for clm in cat\_clms:

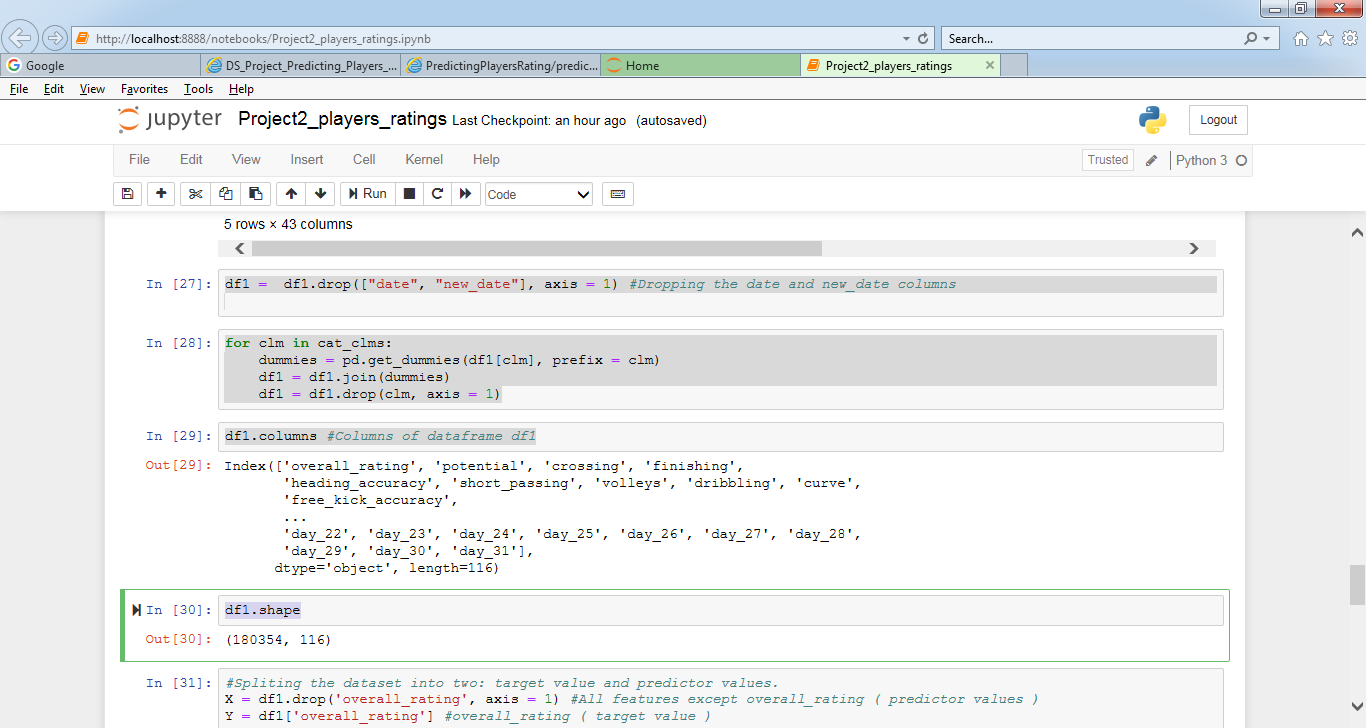
dummies = pd.get\_dummies(df1[clm], prefix = clm)

df1 = df1.join(dummies)

df1 = df1.drop(clm, axis = 1)

df1.columns #Columns of dataframe df1

df1.shape



#Spliting the dataset into two: target value and predictor values.

X = df1.drop('overall\_rating', axis = 1) #All features except overall\_rating ( predictor values )

Y = df1['overall\_rating'] #overall\_rating ( target value )

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2, random\_state = 5)

print(X\_train.shape) #Training data shape (predictor values) : 80%

print(X\_test.shape) #Test data shape (predictor values) : 20%

print(Y\_train.shape) #Training data shape (target values) : 80%

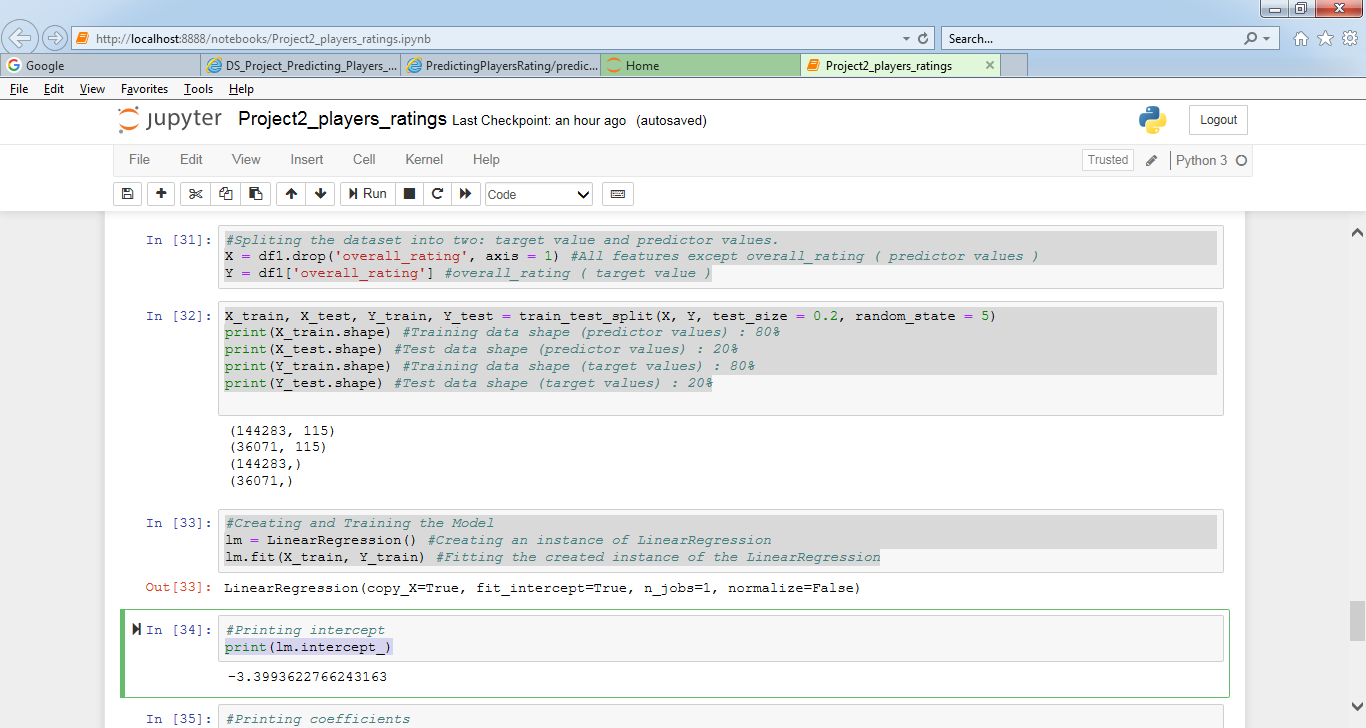
print(Y\_test.shape) #Test data shape (target values) : 20%

#Creating and Training the Model

lm = LinearRegression() #Creating an instance of LinearRegression

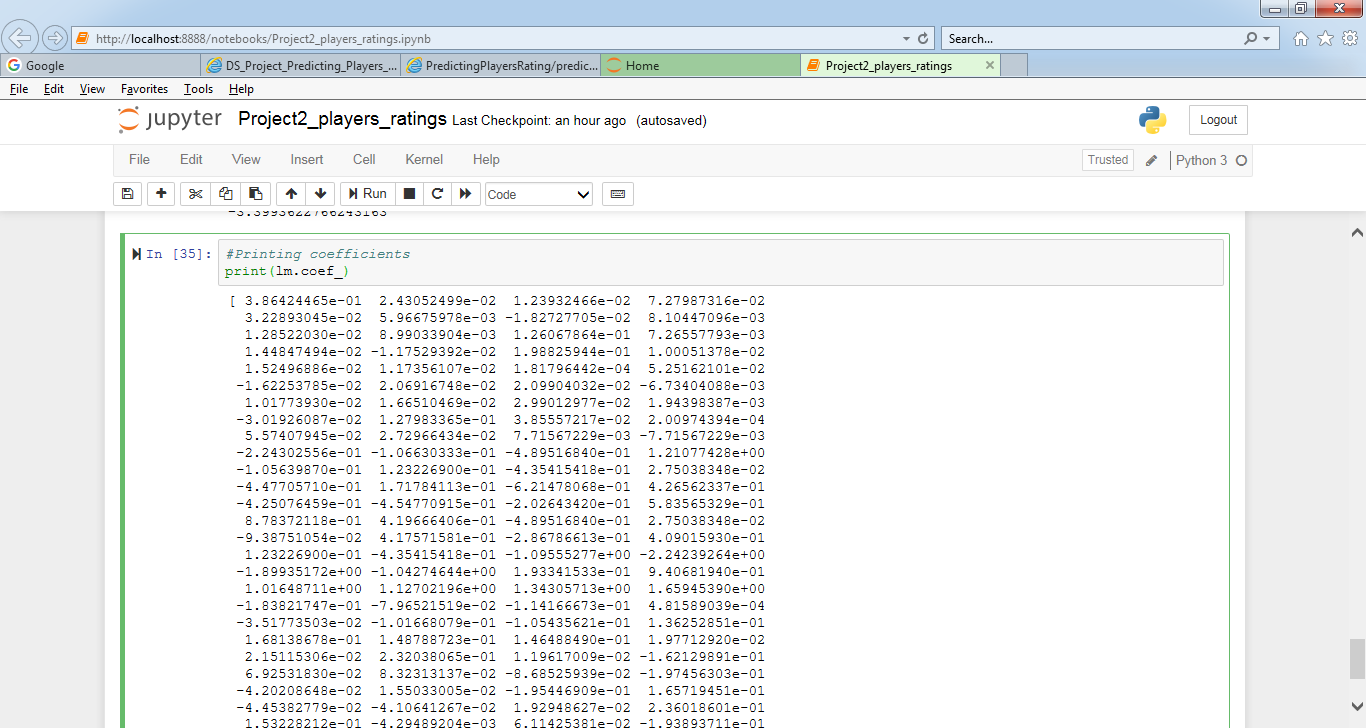
lm.fit(X\_train, Y\_train) #Fitting the created instance of the LinearRegression

print(lm.intercept\_)



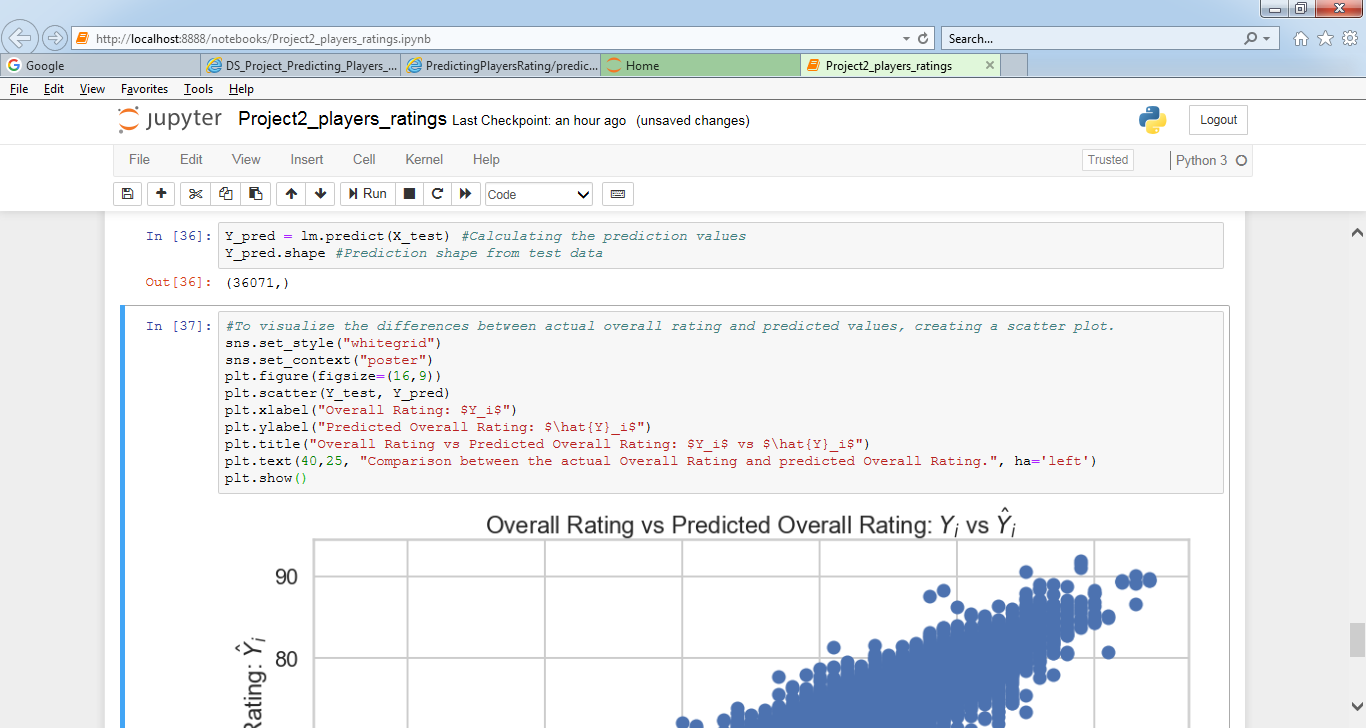
#Printing coefficients

print(lm.coef\_)



Y\_pred = lm.predict(X\_test) #Calculating the prediction values

Y\_pred.shape #Prediction shape from test data



#To visualize the differences between actual overall rating and predicted values, creating a scatter plot.

sns.set\_style("whitegrid")

sns.set\_context("poster")

plt.figure(figsize=(16,9))

plt.scatter(Y\_test, Y\_pred)

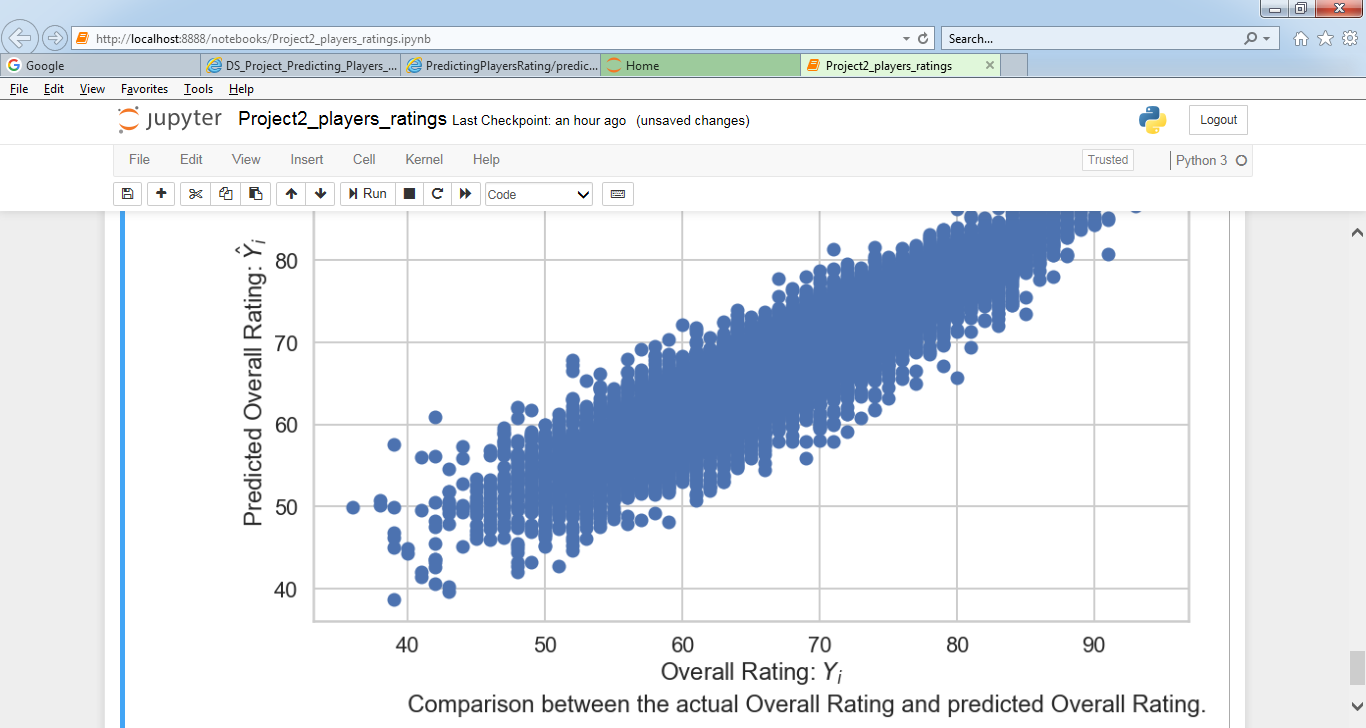
plt.xlabel("Overall Rating: $Y\_i$")

plt.ylabel("Predicted Overall Rating: $\hat{Y}\_i$")

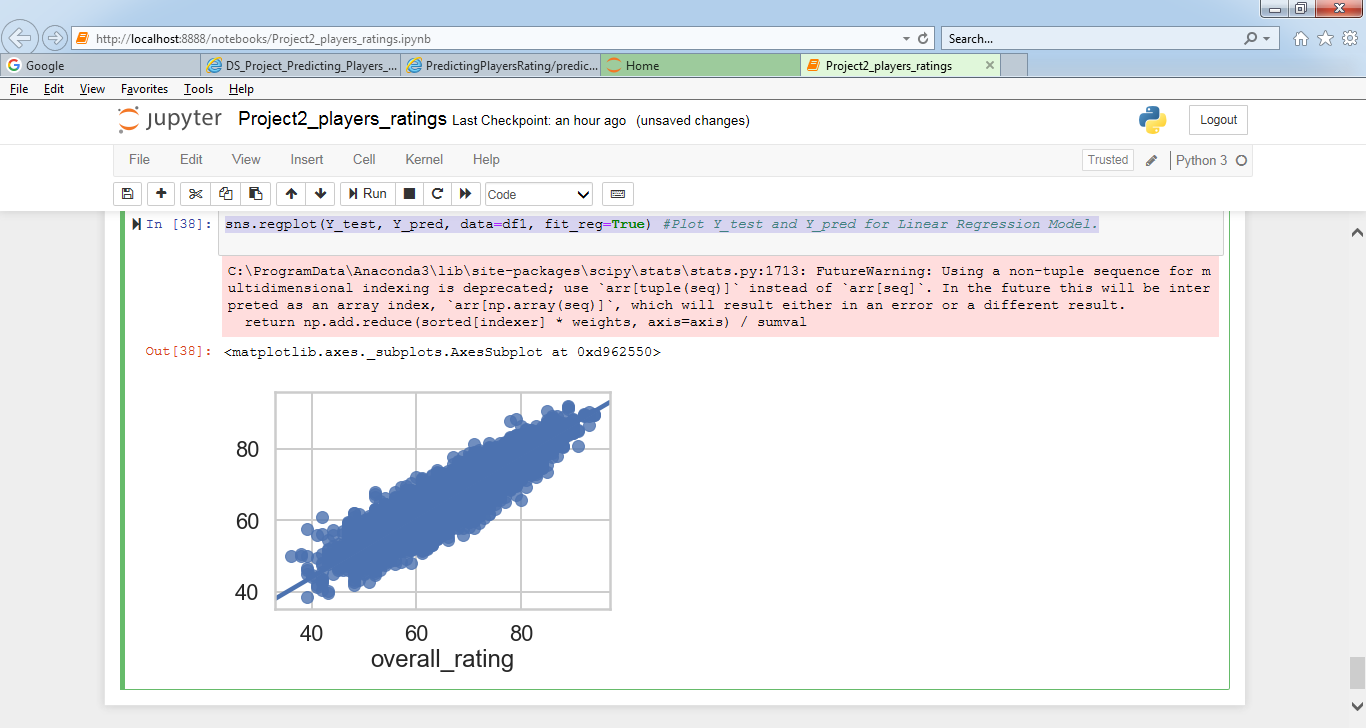
plt.title("Overall Rating vs Predicted Overall Rating: $Y\_i$ vs $\hat{Y}\_i$")

plt.text(40,25, "Comparison between the actual Overall Rating and predicted Overall Rating.", ha='left')

plt.show()



sns.regplot(Y\_test, Y\_pred, data=df1, fit\_reg=True) #Plot Y\_test and Y\_pred for Linear Regression Model.



sns.regplot(x=lm.predict(X), y=df1['overall\_rating'], data=df1, fit\_reg=True) #Plot predicted and actual Overall Rating values.

