**1.Problem Definition**

Weather forecasting is the application of science and technology to predict the conditions of the atmosphere for a given location and time. Weather forecasts are made by collecting quantitative data about the current state of the atmosphere at a given place and using meteorology to project how the atmosphere will change.

Rain Dataset is to predict whether or not it will rain tomorrow. The Dataset contains about 10 years of daily weather observations of different locations in Australia.

Here, we are trying to Build Machine Learning Model to predict Two things:

1. Whether or not it will rain Tomorrow - A classification problem
2. Prediction of amount of rainfall - A regression problem

### 2. **Data Analysis**

* Date - The date of observation
* Location -The common name of the location of the weather station
* MinTemp -The minimum temperature in degrees celsius
* MaxTemp -The maximum temperature in degrees celsius
* Rainfall -The amount of rainfall recorded for the day in mm
* Evaporation -The so-called Class A pan evaporation (mm) in the 24 hours to 9am
* Sunshine -The number of hours of bright sunshine in the day.
* WindGustDir- The direction of the strongest wind gust in the 24 hours to midnight
* WindGustSpeed -The speed (km/h) of the strongest wind gust in the 24 hours to midnight
* WindDir9am -Direction of the wind at 9am
* WindDir3pm -Direction of the wind at 3pm
* WindSpeed9am -Wind speed (km/hr) averaged over 10 minutes prior to 9am
* WindSpeed3pm -Wind speed (km/hr) averaged over 10 minutes prior to 3pm
* Humidity9am -Humidity (percent) at 9am
* Humidity3pm -Humidity (percent) at 3pm
* Pressure9am -Atmospheric pressure (hpa) reduced to mean sea level at 9am
* Pressure3pm -Atmospheric pressure (hpa) reduced to mean sea level at 3pm
* Cloud9am - Fraction of sky obscured by cloud at 9am
* Cloud3pm -Fraction of sky obscured by cloud at 3pm
* Temp9am-Temperature (degrees C) at 9am
* Temp3pm -Temperature (degrees C) at 3pm
* RainToday -Boolean: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0
* RainTomorrow -The amount of next day rain in mm. Used to create response variable . A kind of measure of the "risk".

### Steps Included in the Project:

Import Libraries

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.metrics import accuracy\_score, confusion\_matrix, roc\_curve, roc\_auc\_score, classification\_report

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

import pickle

import warnings

warnings.filterwarnings('ignore')

data = pd.read\_csv('C:\\Users\\DELL\\Downloads\\weatherAUS.csv')

data.head()

| **Date** | **Location** | **MinTemp** | **MaxTemp** | **Rainfall** | **Evaporation** | **Sunshine** | **WindGustDir** | **WindGustSpeed** | **WindDir9am** | **...** | **Humidity9am** | **Humidity3pm** | **Pressure9am** | **Pressure3pm** | **Cloud9am** | **Cloud3pm** | **Temp9am** | **Temp3pm** | **RainToday** | **RainTomorrow** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2008-12-01 | Albury | 13.4 | 22.9 | 0.6 | NaN | NaN | W | 44.0 | W | ... | 71.0 | 22.0 | 1007.7 | 1007.1 | 8.0 | NaN | 16.9 | 21.8 | No | No |
| **1** | 2008-12-02 | Albury | 7.4 | 25.1 | 0.0 | NaN | NaN | WNW | 44.0 | NNW | ... | 44.0 | 25.0 | 1010.6 | 1007.8 | NaN | NaN | 17.2 | 24.3 | No | No |
| **2** | 2008-12-03 | Albury | 12.9 | 25.7 | 0.0 | NaN | NaN | WSW | 46.0 | W | ... | 38.0 | 30.0 | 1007.6 | 1008.7 | NaN | 2.0 | 21.0 | 23.2 | No | No |
| **3** | 2008-12-04 | Albury | 9.2 | 28.0 | 0.0 | NaN | NaN | NE | 24.0 | SE | ... | 45.0 | 16.0 | 1017.6 | 1012.8 | NaN | NaN | 18.1 | 26.5 | No | No |
| **4** | 2008-12-05 | Albury | 17.5 | 32.3 | 1.0 | NaN | NaN | W | 41.0 | ENE | ... | 82.0 | 33.0 | 1010.8 | 1006.0 | 7.0 | 8.0 | 17.8 | 29.7 | No | No |

5 rows × 23 columns

In [5]:

data.columns

Out[5]:

Index(['Date', 'Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation',

'Sunshine', 'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',

'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',

'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',

'Temp3pm', 'RainToday', 'RainTomorrow'],

dtype='object')

In [6]:

data.shape

Out[6]:

(8425, 23)

In [7]:

data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 8425 entries, 0 to 8424

Data columns (total 23 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Date 8425 non-null object

1 Location 8425 non-null object

2 MinTemp 8350 non-null float64

3 MaxTemp 8365 non-null float64

4 Rainfall 8185 non-null float64

5 Evaporation 4913 non-null float64

6 Sunshine 4431 non-null float64

7 WindGustDir 7434 non-null object

8 WindGustSpeed 7434 non-null float64

9 WindDir9am 7596 non-null object

10 WindDir3pm 8117 non-null object

11 WindSpeed9am 8349 non-null float64

12 WindSpeed3pm 8318 non-null float64

13 Humidity9am 8366 non-null float64

14 Humidity3pm 8323 non-null float64

15 Pressure9am 7116 non-null float64

16 Pressure3pm 7113 non-null float64

17 Cloud9am 6004 non-null float64

18 Cloud3pm 5970 non-null float64

19 Temp9am 8369 non-null float64

20 Temp3pm 8329 non-null float64

21 RainToday 8185 non-null object

22 RainTomorrow 8186 non-null object

dtypes: float64(16), object(7)

memory usage: 1.5+ MB

data.isnull().sum()

Date 0

Location 0

MinTemp 75

MaxTemp 60

Rainfall 240

Evaporation 3512

Sunshine 3994

WindGustDir 991

WindGustSpeed 991

WindDir9am 829

WindDir3pm 308

WindSpeed9am 76

WindSpeed3pm 107

Humidity9am 59

Humidity3pm 102

Pressure9am 1309

Pressure3pm 1312

Cloud9am 2421

Cloud3pm 2455

Temp9am 56

Temp3pm 96

RainToday 240

RainTomorrow 239

dtype: int64

In [8]:

elements **=** data.select\_dtypes(include**=**['object']).columns

​

print(elements)

​

**for** i **in** elements:

data[i] **=** data[i].fillna(data[i].mode()[0])

Index(['Date', 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm',

'RainToday', 'RainTomorrow'],

dtype='object')

In [9]:

cont **=** data.select\_dtypes(include**=**['float']).columns

​

print(cont)

​

**for** i **in** cont:

data[i] **=** data[i].fillna(data[i].mean())

Index(['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',

'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',

'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm',

'Temp9am', 'Temp3pm'],

dtype='object')

In [11]:

data.isnull().sum()

Out[11]:

Date 0

Location 0

MinTemp 0

MaxTemp 0

Rainfall 0

Evaporation 0

Sunshine 0

WindGustDir 0

WindGustSpeed 0

WindDir9am 0

WindDir3pm 0

WindSpeed9am 0

WindSpeed3pm 0

Humidity9am 0

Humidity3pm 0

Pressure9am 0

Pressure3pm 0

Cloud9am 0

Cloud3pm 0

Temp9am 0

Temp3pm 0

RainToday 0

RainTomorrow 0

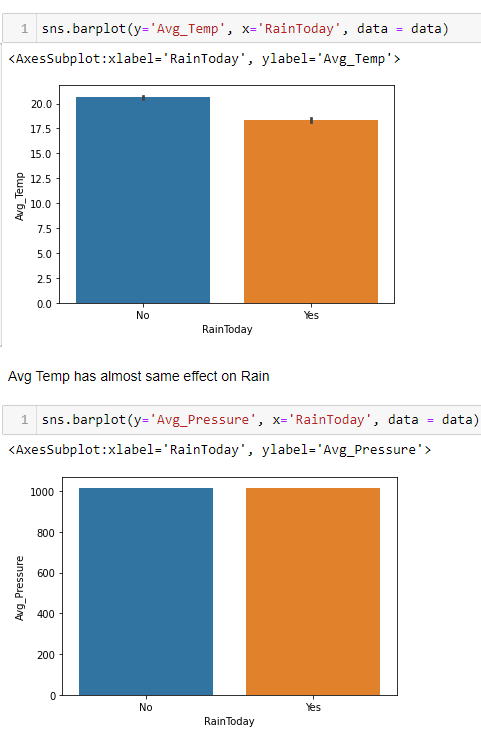
dtype: int64

data.head()

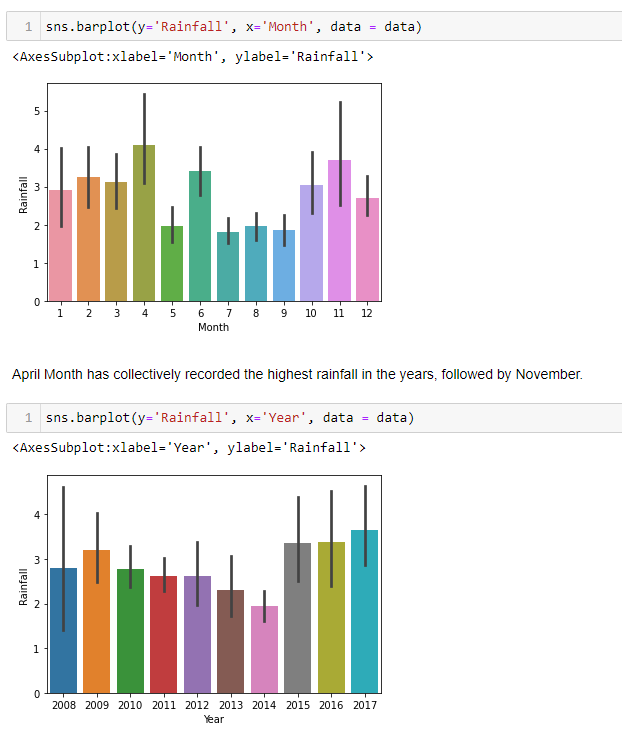
| **Date** | **Location** | **MinTemp** | **MaxTemp** | **Rainfall** | **Evaporation** | **Sunshine** | **WindGustDir** | **WindGustSpeed** | **WindDir9am** | **...** | **Humidity9am** | **Humidity3pm** | **Pressure9am** | **Pressure3pm** | **Cloud9am** | **Cloud3pm** | **Temp9am** | **Temp3pm** | **RainToday** | **RainTomorrow** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2008-12-01 | Albury | 13.4 | 22.9 | 0.6 | 5.389395 | 7.632205 | W | 44.0 | W | ... | 71.0 | 22.0 | 1007.7 | 1007.1 | 8.000000 | 4.503183 | 16.9 | 21.8 | No | No |
| **1** | 2008-12-02 | Albury | 7.4 | 25.1 | 0.0 | 5.389395 | 7.632205 | WNW | 44.0 | NNW | ... | 44.0 | 25.0 | 1010.6 | 1007.8 | 4.566622 | 4.503183 | 17.2 | 24.3 | No | No |
| **2** | 2008-12-03 | Albury | 12.9 | 25.7 | 0.0 | 5.389395 | 7.632205 | WSW | 46.0 | W | ... | 38.0 | 30.0 | 1007.6 | 1008.7 | 4.566622 | 2.000000 | 21.0 | 23.2 | No | No |
| **3** | 2008-12-04 | Albury | 9.2 | 28.0 | 0.0 | 5.389395 | 7.632205 | NE | 24.0 | SE | ... | 45.0 | 16.0 | 1017.6 | 1012.8 | 4.566622 | 4.503183 | 18.1 | 26.5 | No | No |
| **4** | 2008-12-05 | Albury | 17.5 | 32.3 | 1.0 | 5.389395 | 7.632205 | W | 41.0 | ENE | ... | 82.0 | 33.0 | 1010.8 | 1006.0 | 7.000000 | 8.000000 | 17.8 | 29.7 | No | No |

5 rows × 23 columns

**3.EDA Conculding Remarks:**

) [](https://user-images.githubusercontent.com/96686904/181914235-14cd31f4-f2e1-458e-9850-b82290d11a18.png)

Exploring Variables(Data Analysis)

Univarite, Bivariate and Multi Variate Analysis [](https://user-images.githubusercontent.com/96686904/181914242-ef92ed1a-19ba-4dc1-bafb-4982b31c05fb.png)

**plt.figure(figsize = (15,10))**

**plotnumber = 1**

**for column in x:**

**if plotnumber <= 20:**

**ax = plt.subplot(4,5,plotnumber)**

**sns.distplot(x[column])**

**plt.xlabel(column, fontsize = 20)**

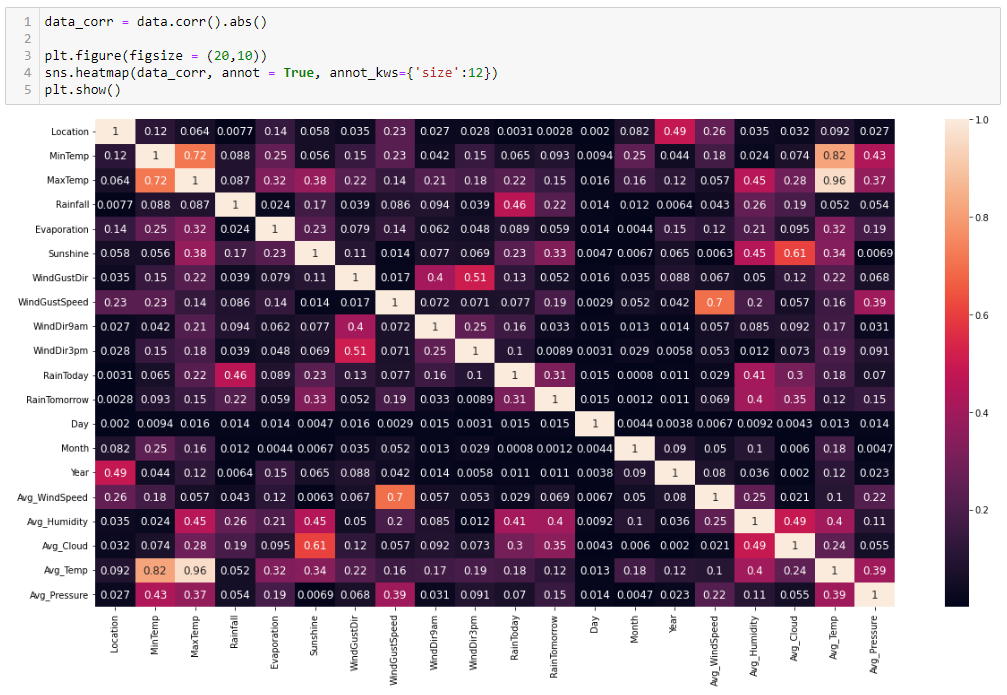
**plotnumber +=1**

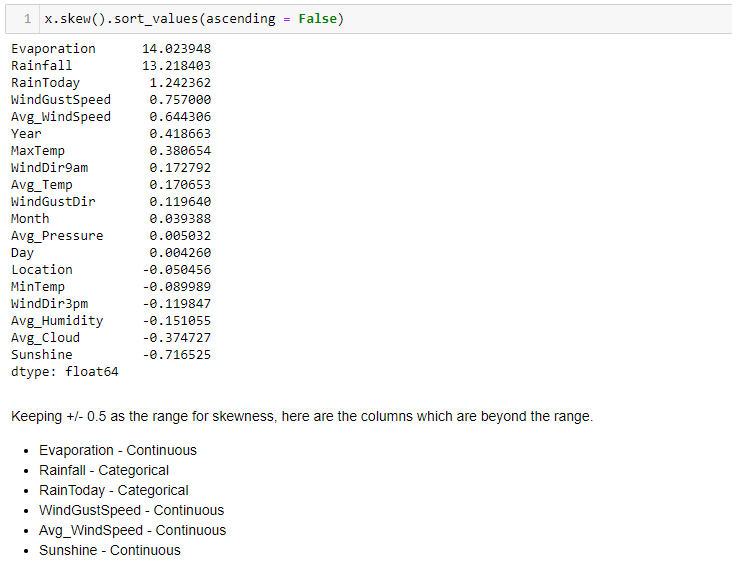
**plt.tight\_layout()**

[](https://user-images.githubusercontent.com/96686904/181914284-a4eab474-be86-48f2-abc6-3096fae015e3.png)

**4.Pre-processing Pipeline**

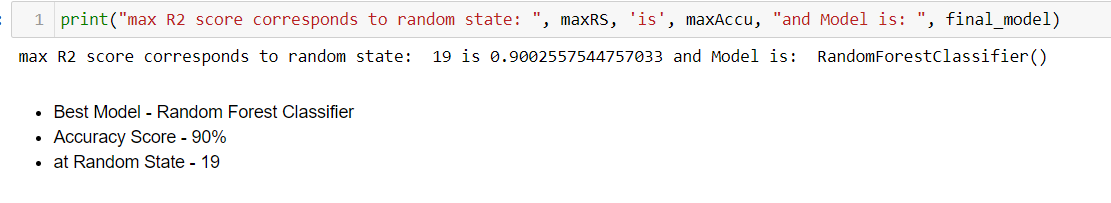
Data Processing like Checking Skewness, Checking Outliers, Checking Corr

[](https://user-images.githubusercontent.com/96686904/181914262-fbf82137-6a5a-4dc6-9401-d572ee39edc2.png)

[](https://user-images.githubusercontent.com/96686904/181914276-669605ab-ae32-48c6-a62b-4a84f885f6c7.png)

**5. Building Machine Learning Models**

Choosing right model

[](https://user-images.githubusercontent.com/96686904/181914313-773f9806-d519-4ebe-8102-afe523dff4e1.png)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

x\_scaled = scaler.fit\_transform(x)

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

vif = pd.DataFrame()

vif['vif'] = [variance\_inflation\_factor(x\_scaled, i) for i in range (x\_scaled.shape[1])]

vif['features'] = x.columns

vif

from sklearn.metrics import accuracy\_score, confusion\_matrix, roc\_curve, roc\_auc\_score, classification\_report

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

import xgboost as xgb

from sklearn.svm import SVC

from sklearn.model\_selection import cross\_val\_score

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_scaled, y, test\_size = 0.25, random\_state = 199)

svc = SVC()

svc.fit(x\_train, y\_train)

y\_pred = svc.predict(x\_test)

print('Accuracy Score: ', accuracy\_score(y\_test, y\_pred))

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

xr\_scaled = scaler.fit\_transform(x\_r)

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

vif = pd.DataFrame()

vif['vif'] = [variance\_inflation\_factor(xr\_scaled, i) for i in range (xr\_scaled.shape[1])]

vif['features'] = x\_r.columns

vif

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error

from sklearn.ensemble import RandomForestRegressor

import xgboost as xgb

from sklearn.neighbors import KNeighborsRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.svm import SVR

max\_r2\_score = 0

maxRS = 0

model = [LinearRegression(),

DecisionTreeRegressor(),

KNeighborsRegressor(),

RandomForestRegressor(),

xgb.XGBRegressor(),

SVR()]

for rs in range(1,200):

xr\_train, xr\_test, yr\_train, yr\_test = train\_test\_split(xr\_scaled, y\_r, test\_size = 0.25, random\_state = rs)

for ir in model:

ir.fit(xr\_train, yr\_train)

yr\_pred = ir.predict(xr\_test)

r2score = r2\_score( yr\_test,yr\_pred)

print('r2\_score', r2score\*100,'%', 'random\_state', rs, 'Model', ir)

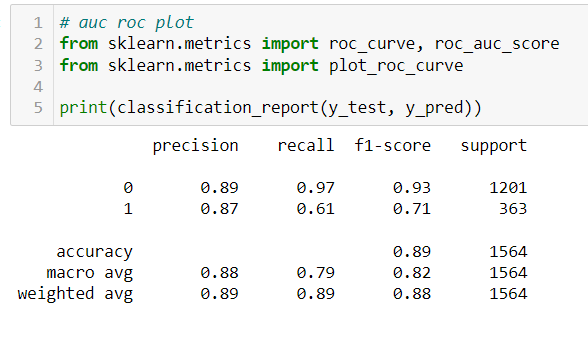
if r2score > max\_r2\_score:

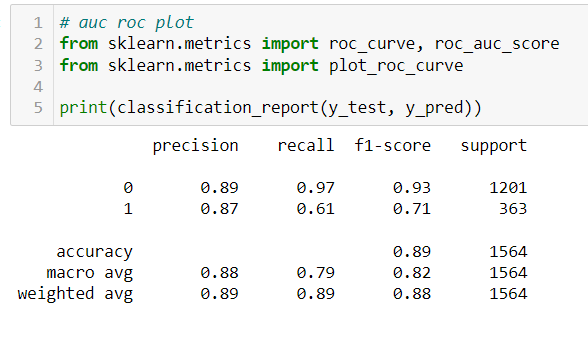
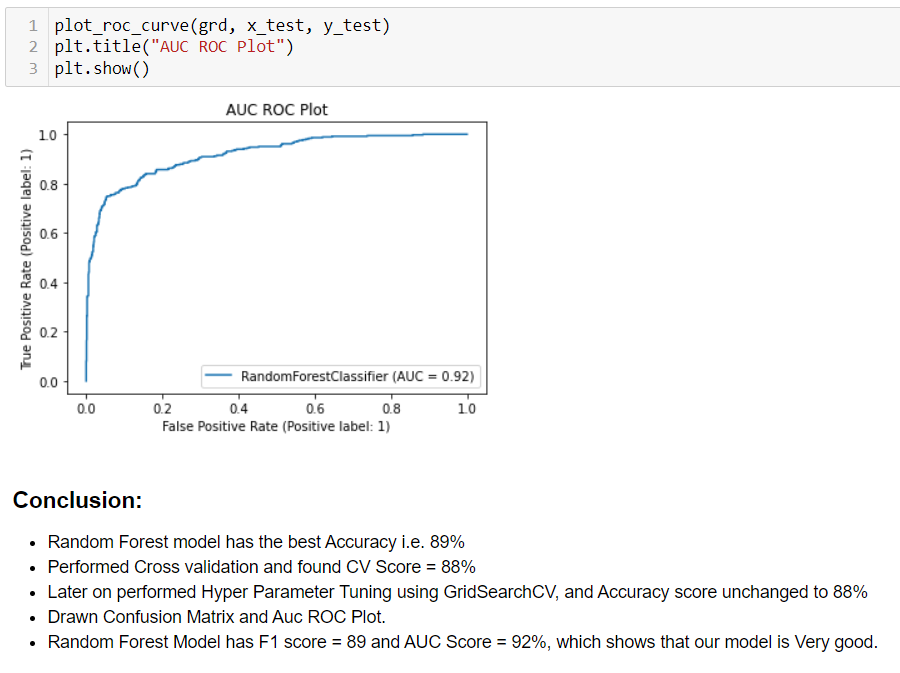
max\_r2\_score = r2score

maxRS = rs

final\_model = ir

print("max R2 score corresponds to random state: ", maxRS, 'is', max\_r2\_score, "and Model is: ", final\_model)

Performing Parameter Tuning [](https://user-images.githubusercontent.com/96686904/181914326-182335d8-f72f-4140-8e4e-2688f1ff6a43.png)

* Performing Parameter Tuning [](https://user-images.githubusercontent.com/96686904/181914326-182335d8-f72f-4140-8e4e-2688f1ff6a43.png) [](https://user-images.githubusercontent.com/96686904/181914330-db29e1a8-f82f-485f-b1e2-cf5467659071.png)

**6.Concluding Remark**

* RandomForest Regressor model has the best Accuracy i.e. 91%
* Performed Hyper Parameter Tuning using GridSearchCV, and Accuracy score was 88%, which is very good.
* Drawn Distribution Plot between Acutal and Predicted Test Data, Which has Normally Distribution and shows that our Model is working very well.