RAINFALL PREDICTION

Aim:-

The problem statement rainfall prediction has to predict the amount of rainfall by month wise and season wise based on the year.



Libraries that are used in our model:-

PANDAS:- pandas is a open source library which works on data manipulation and data analysis

NUMPY:-numpy is a multidimensional arraywhich works on large set of data to operate arrays

SKLERN:-sklearn works on different types of supervised and unsupervised learning techniques.

SKLEARN.METRICS:-it is a sklearn library and used for both classification and regression problems.

MATH:-math module is used for different mathematical operations.

MATPLOTLIB:-matplotlib is an visualization library for 2d plots of arrays.

SEABORN:-seaborn provides beautiful default styles and colors to make statistical and attractive.

LINEAR REGRESSION:-linear regression is the relationship between a dependent and independent variables

MEAN\_SQUARED\_ERROR:-the average squared difference between the estimated value and true value

R2\_SCORE:-it tells the difference between the samples in the dataset and the predictions made by the model

MEAN\_ABSOLUTE\_ERROR:-it is the amount of error in our measurements.

**Python code**

Import libraries

import pandas as pd

import numpy as np

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

import math

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

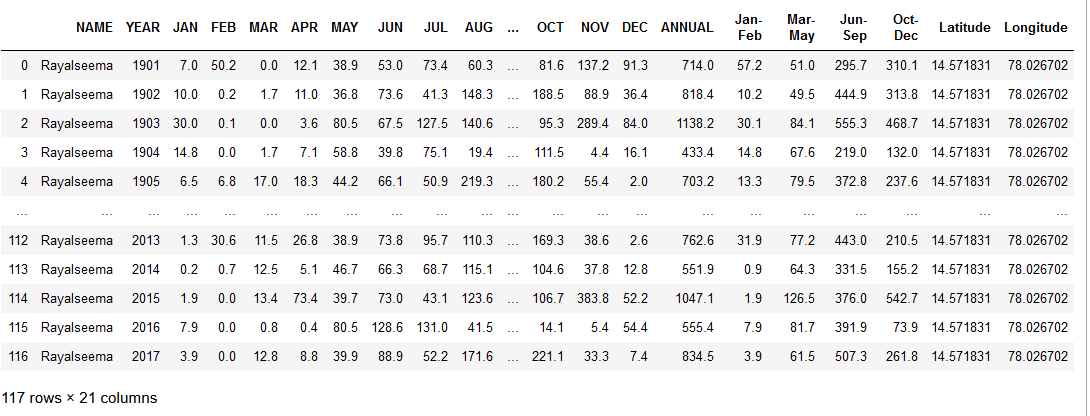
import matplotlib.pyplot as plt

import seaborn as sns

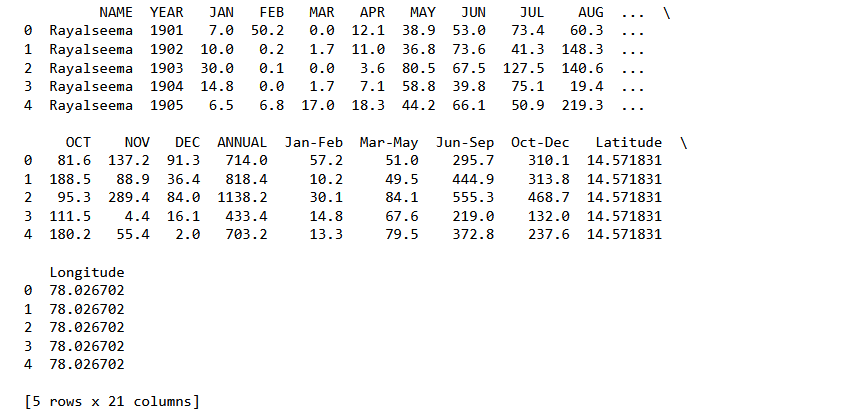
from sklearn import metrics

import data set

data = pd.read\_csv("rayalseemarainfall.csv")

data

print(data.head())



data.duplicated().sum()

output:-

0

data.isnull().any()

output:-

NAME False

YEAR False

JAN False

FEB False

MAR False

APR False

MAY False

JUN False

JUL False

AUG False

SEP False

OCT False

NOV False

DEC False

ANNUAL False

Jan-Feb False

Mar-May False

Jun-Sep False

Oct-Dec False

Latitude False

Longitude False

dtype: bool

data.YEAR.unique()

output:-

array([1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1911,

1912, 1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922,

1923, 1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933,

1934, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944,

1945, 1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955,

1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966,

1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977,

1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988,

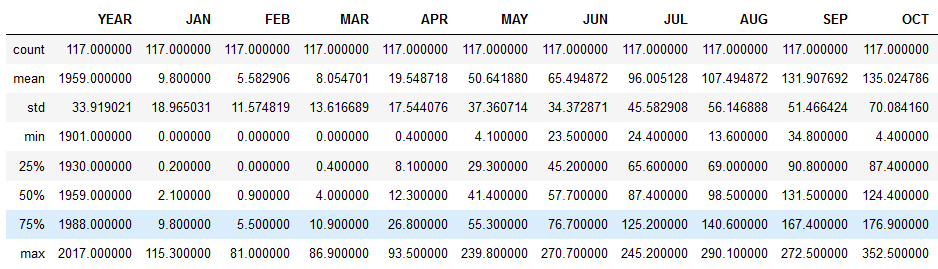
1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999,

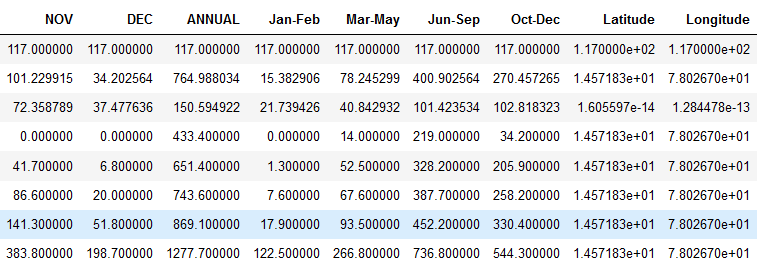
2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,

2011, 2012, 2013, 2014, 2015, 2016, 2017], dtype=int64)

data.describe()

output:-





Month wise annual rainfall v/s subdivision:-

#GRID:-it just create the widegets,and use the grid method to tell the manner in which row and column to place them.

data[["JAN","FEB","MAR","APR","MAY","JUN","JUL","AUG","SEP","OCT","NOV","DEC","ANNUAL"]].groupby("ANNUAL").sum().sort\_values(by='ANNUAL',ascending=False).plot(kind='barh',stacked=True,figsize=(15,10))

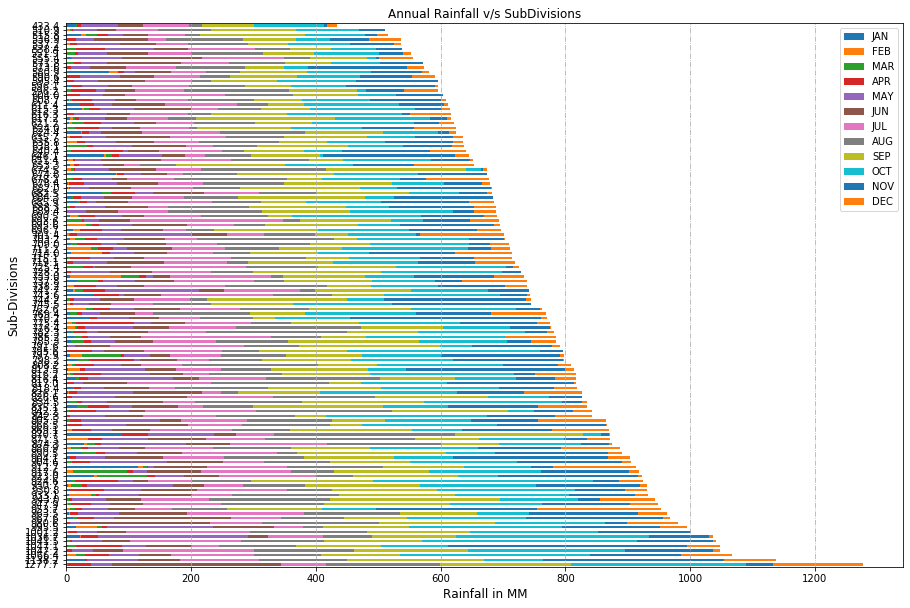
plt.xlabel("Rainfall in MM",size=12)

plt.ylabel("Sub-Divisions",size=12)

plt.title("Annual Rainfall v/s SubDivisions")

plt.grid(axis="x",linestyle="-.")

plt.show()



Seasonal wise annual rainfall v/s subdivision:-

data[['ANNUAL','Jan-Feb','Mar-May','Jun-Sep','Oct-Dec']].groupby("ANNUAL").sum().plot(kind="barh",stacked=True,figsize=(20,10))

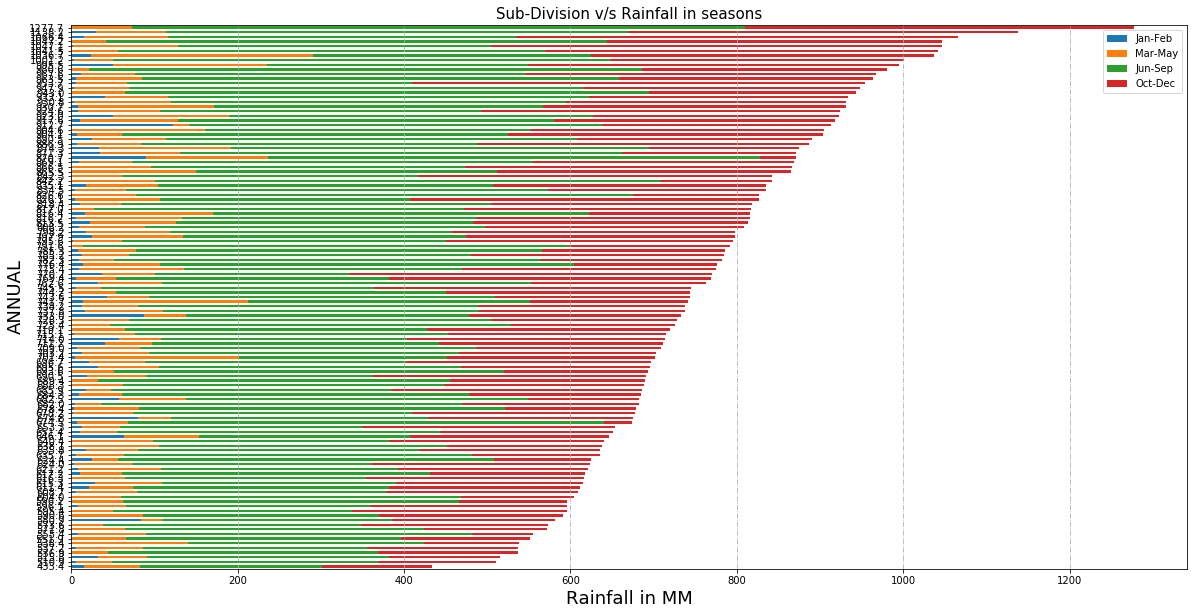
plt.title("Sub-Division v/s Rainfall in seasons",size=15)

plt.xlabel("Rainfall in MM",size=18)

plt.ylabel("ANNUAL",size=18)

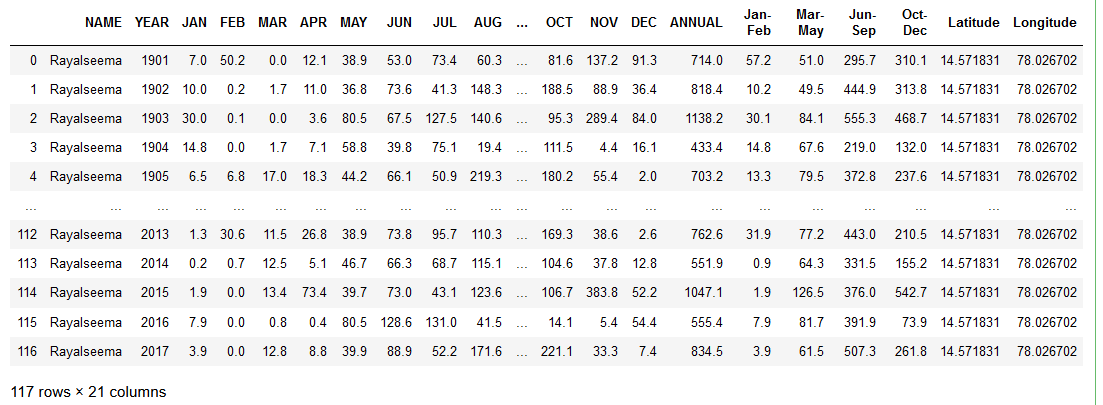
plt.grid(axis="x",linestyle="-.")

plt.show()



data = data.fillna(np.mean(data))

data



print("\nShape: ",data.shape)

output:-

Shape: (117, 21)

GROUPBY:-groupby is used for grouping the data according to the categories and apply a function to the categories.

print("Group by: ")

data.groupby('YEAR').size()

output:-

Group by:

YEAR

1901 1

1902 1

1903 1

1904 1

1905 1

..

2013 1

2014 1

2015 1

2016 1

2017 1

Length: 117, dtype: int64

#pie chat annual indicator:-

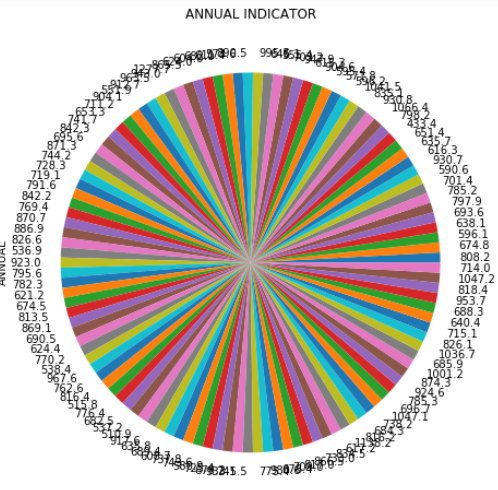
import matplotlib.pyplot as plt

fig = plt.figure(figsize = (7,7))

data.ANNUAL.value\_counts().plot(kind='pie')

plt.title('ANNUAL INDICATOR')

plt.show()



#overall rainfall in each month:-

months = data.columns[2:14]

fig = plt.figure(figsize=(18,10))

ax = fig.add\_subplot(111)

data.groupby('YEAR').mean()[months].plot.line(title='Overall Rainfall in Each Month of Year', ax=ax,fontsize=20)

plt.xticks( rotation = 0)

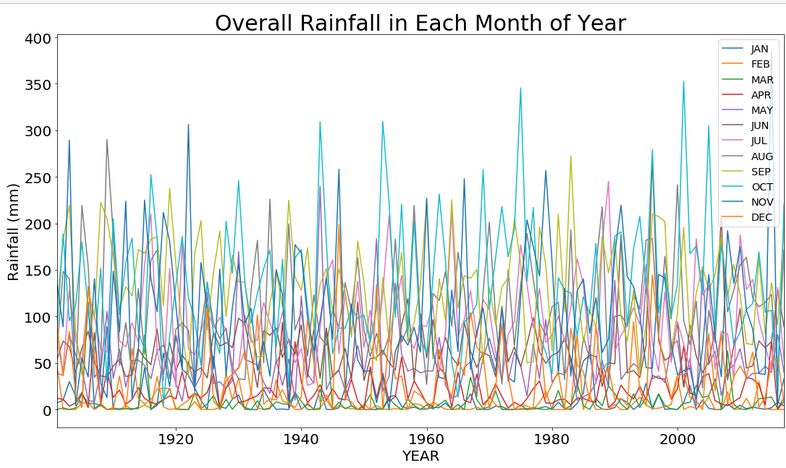
plt.ylabel('Rainfall (mm)')

plt.legend(loc='upper right', fontsize = 'x-large')

ax.title.set\_fontsize(30)

ax.xaxis.label.set\_fontsize(20)

ax.yaxis.label.set\_fontsize(20)



#overall rainfall in seasons:-

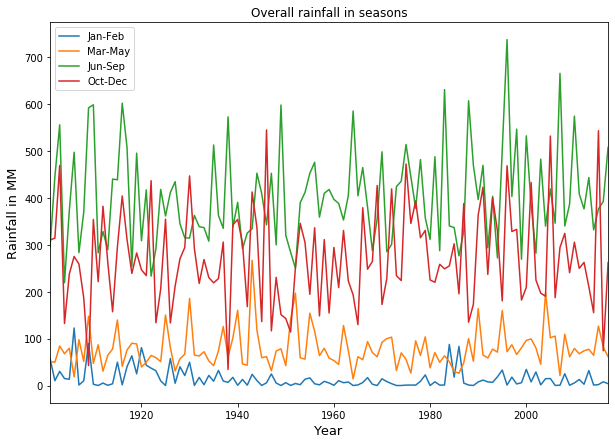
data[['YEAR','Jan-Feb', 'Mar-May','Jun-Sep', 'Oct-Dec']].groupby("YEAR").sum().plot(figsize=(10,7))

plt.title('Overall rainfall in seasons')

plt.xlabel("Year",size=13)

plt.ylabel("Rainfall in MM",size=13)

plt.show()



#correlation:-

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

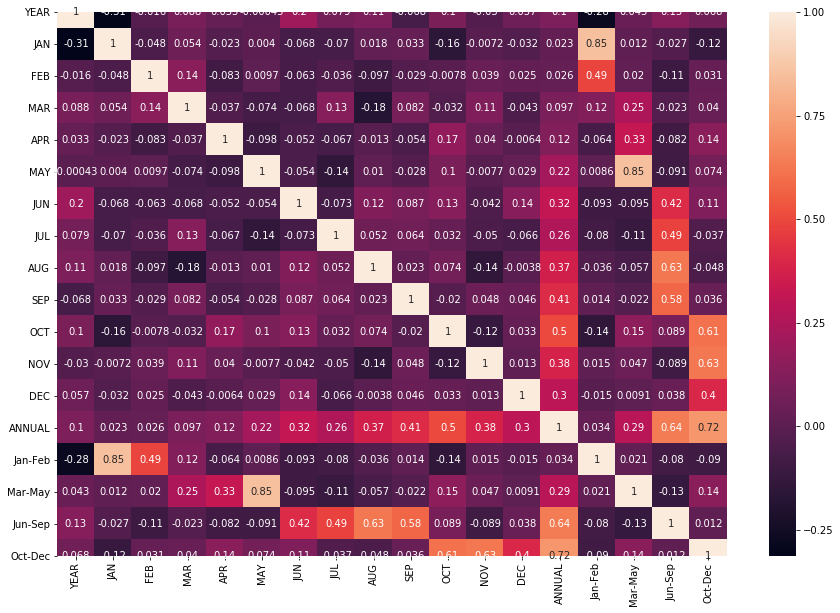
sns.heatmap(data[['YEAR','JAN','FEB','MAR','APR','MAY','JUN',

'JUL','AUG','SEP','OCT','NOV','DEC','ANNUAL',

'Jan-Feb', 'Mar-May','Jun-Sep', 'Oct-Dec']

].corr(),annot=True)

plt.show()



#prediction:-

X=np.asanyarray(data[['YEAR']]).astype('int')

y=np.asanyarray(data[['ANNUAL','JAN','FEB','MAR','APR','MAY','JUN', JUL','AUG','SEP','OCT','NOV','DEC','Jan-Feb','Mar-May', 'Jun-Sep','Oct-Dec']]).astype('int')

#random state is used to generate number of sets randomly

print(X.shape)

print(y.shape)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=10)

#Random Forest Model

from sklearn.ensemble import RandomForestRegressor

random\_forest\_model = RandomForestRegressor(max\_depth=100, max\_features='sqrt', min\_samples\_leaf=4, min\_samples\_split=10, n\_estimators=800)

#min\_sample split specifies the minimum number of samples required to split the internal node

#n\_estimators gives the number of trees you want to build

random\_forest\_model.fit(X\_train, y\_train)

y\_train\_predict=random\_forest\_model.predict(X\_train)

y\_test\_predict=random\_forest\_model.predict(X\_test)

print("-------Test Data--------")

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_test\_predict))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_test\_predict))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_test\_predict)))

print("\n-------Train Data--------")

print('MAE:', metrics.mean\_absolute\_error(y\_train,y\_train\_predict))

print('MSE:', metrics.mean\_squared\_error(y\_train, y\_train\_predict))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_train, y\_train\_predict)))

print("-----------Training Accuracy------------")

print(round(random\_forest\_model.score(X\_train,y\_train),3)\*100)

print("-----------Testing Accuracy------------")

print(round(random\_forest\_model.score(X\_test,y\_test),3)\*100)

predicted = random\_forest\_model.predict([[2020]])

predicted

output:-

(117, 1)

(117, 17)

-------Test Data--------

MAE: 40.643595698088085

MSE: 4541.492185424869

RMSE: 67.39059419106549

-------Train Data--------

MAE: 33.98227732197044

MSE: 2936.731796084506

RMSE: 54.19162108743847

-----------Training Accuracy------------

0.872

-----------Testing Accuracy------------

0.747

array([[

YEAR :846.92222421,

JAN :3.43765609,

FEB :5.44539375,

MAR :15.9429455,

APR :20.79327415,

MAY :6.63692857,

JUN :73.31682716,

JUL :86.95562209,

AUG :106.49425163,

SEP :123.07655613,

OCT :130.61734303,

NOV :98.31561425,

DEC :27.98796044,

JAN-FEB :8.96657953,

MAR-MAY :85.33763813,

JUN-SEP :391.22489029,

OCT-DEC :257.95283407

]])