**SPRINT 2**

|  |  |
| --- | --- |
| Team ID | PNT2022TMID17878 |
| Project Name | Exploratory Analysis Of Rainfall Data In India For Agriculture |

**TO GROUP THE VALUES OF DATASET W.R.T. STATES:**

group = df.groupby('SUBDIVISION')['YEAR','JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP','OCT','NOV','DEC']

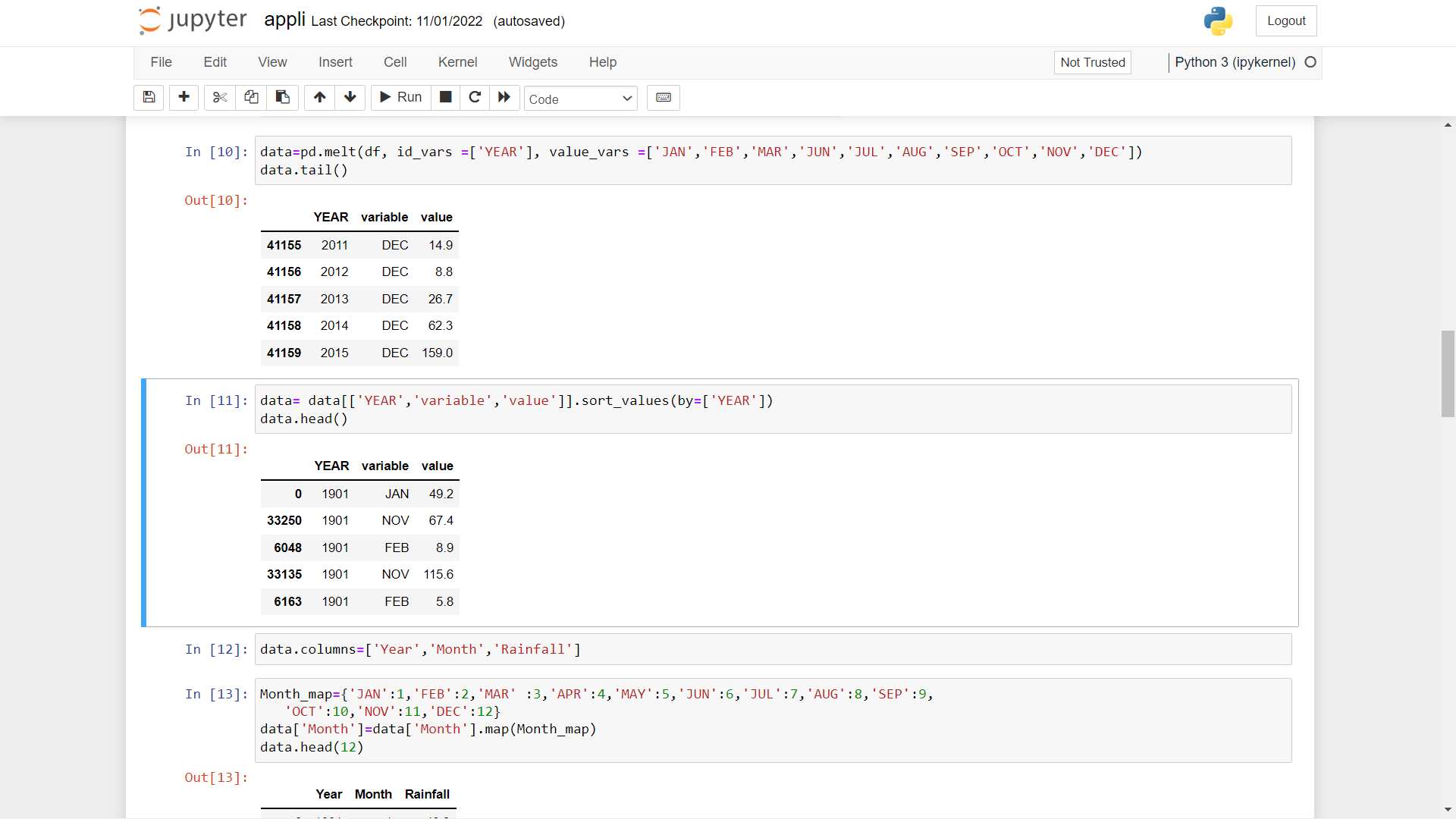
data=group.get\_group(('TAMIL NADU'))

data=pd.melt(df, id\_vars =['YEAR'], value\_vars =['JAN','FEB','MAR','JUN','JUL','AUG','SEP','OCT','NOV','DEC'])

data.tail()

data= data[['YEAR','variable','value']].sort\_values(by=['YEAR'])

data.head()



data.columns=['Year','Month','Rainfall']

Month\_map={'JAN':1,'FEB':2,'MAR' :3,'APR':4,'MAY':5,'JUN':6,'JUL':7,'AUG':8,'SEP':9,

'OCT':10,'NOV':11,'DEC':12}

data['Month']=data['Month'].map(Month\_map)

**TO SPLIT THE INPUT AND OUTPTU FEATURE COLUMN**

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

y=np.asanyarray(data['Rainfall']).astype('int')

print(X.shape)

print(y.shape)

OUTPUT:

(41160, 1)

(41160,)

**TO SPLIT THE TRAIN AND TEST DATA**

# splitting the dataset into training and testing

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)

**TO NORMALIZE THE INPUT DATA**

from sklearn.preprocessing import StandardScaler

sc\_X = StandardScaler()

X\_train = sc\_X.fit\_transform(X\_train)

X\_test = sc\_X.transform(X\_test)

**USING DIFFERENT MODELS TO TRAIN ON THE DATASET AND FIND THE BEST FITTING MODEL**

RANDOM FOREST REGRESSION

from sklearn.ensemble import RandomForestRegressor

random\_forest\_model = RandomForestRegressor(max\_depth=10, max\_features='sqrt', n\_estimators=5000)

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

OUTPUT:

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

MAE: 82.0659922456978

MSE: 20287.117086747196

RMSE: 142.4328511501023

**LASSO REGRESSION**

from sklearn.linear\_model import Lasso

reg = Lasso(alpha=0.001)

reg.fit(X\_train, y\_train)

y\_test\_predict=reg.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)))

OUTPUT:

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

MAE: 128.99256559491246

MSE: 34470.61880070557

RMSE: 185.66264783392907

**RIDGE REGRESSION**

# use automatically configured the ridge regression algorithm

from numpy import arange

from sklearn.linear\_model import RidgeCV

cv = RepeatedKFold(n\_splits=10, n\_repeats=3, random\_state=1)

# define model

model = RidgeCV(alphas=arange(0, 1, 0.01), cv=cv, scoring='neg\_mean\_absolute\_error')

# fit model

model.fit(X\_train, y\_train)

# summarize chosen configuration

print('alpha: %f' % model.alpha\_)

model = Ridge(alpha=0.0)

# define model evaluation method

cv = RepeatedKFold(n\_splits=10, n\_repeats=5, random\_state=1)

# evaluate model

scores = cross\_val\_score(model, X\_train, y\_train, scoring='neg\_mean\_absolute\_error', cv=cv, n\_jobs=-1)

# force scores to be positive

scores = absolute(scores)

print('Mean MAE: %.3f (%.3f)' % (mean(scores), std(scores)))

model = Ridge(alpha=0.0000)

# fit model

model.fit(X\_train, y\_train)

y\_test\_predict=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)))

OUTPUT:

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

MAE: 128.99244468794677

MSE: 34470.61608555598

RMSE: 185.66264052187768

**DECISION TREE REGRESSION**

from sklearn.tree import DecisionTreeRegressor

regressor = DecisionTreeRegressor(random\_state = 10)

regressor.fit(X\_train, y\_train)

y\_test\_predict=regressor.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)))

OUTPUT:

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

MAE: 82.06811302080845

MSE: 20287.10126311729

RMSE: 142.43279560240782

**TO FORM THE PICKLE FILE**

import pickle

file = open("model.pkl","wb")

pickle.dump(random\_forest\_model,file)

file.close()

# print(y\_predict)

Similarly the different models can be constructed for various sates in the dataset.

**CONCLUSION:**

**THE BEST MODEL IS RANDOM FOREST REGRESSION**