SPRINT 2

Team ID	PNT2022TMID30419
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']

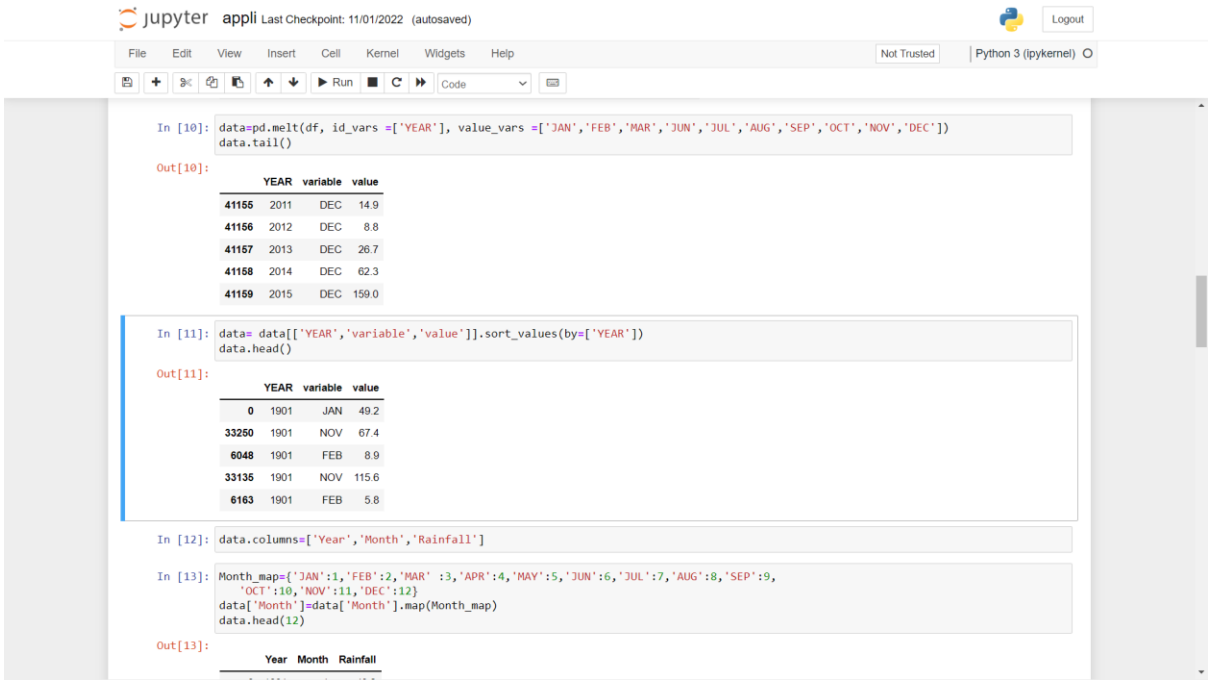
data=group.get_group(('TAMIL NADU'))

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