**PYTHON FILE FOR THE ALGORITHM (RANDOM FOREST)**

import numpy as np

import pandas as pd

import os

for dirname, \_, filenames in os.walk('/content/PoductDemand.csv.zip'):

    for filename in filenames:

        print(os.path.join(dirname, filename))

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.metrics import f1\_score, accuracy\_score, confusion\_matrix ,classification\_report

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier

from plotly.offline import iplot

import plotly as py

import plotly.tools as tls

import pickle

data  = pd.read\_csv("/content/PoductDemand.csv.zip")

for ind in data.index:

    continue

def remove\_space\_between\_word(dataset):

    for col in dataset.columns:

        for i in range(len(dataset[col])):

            if (type(dataset[col][i]) == str ):

                dataset[col][i] = dataset[col][i].strip()

                dataset[col][i] = dataset[col][i].replace(" ", "\_")

    return data

new\_df = remove\_space\_between\_word(data)

new\_df.head()

fig = plt.figure(figsize=(5, 5))

ax = fig.gca()

data["ID"].hist(ax =ax)

plt.ylabel("number of apparution in data")

plt.xlabel("UNITS SOLD")

new\_df.hist()

def encode\_data(dataset , data\_dict\_weigth):

    cols = dataset.columns

    for columnName in cols:

        for i in range(len(dataset[columnName])):

            try:

            #print(data\_dict[data2[columnName][i]]["weight"])

                dataset[columnName][i] = data\_dict[dataset[columnName][i]]["weight"]

            except:

                pass

    dataset = dataset.fillna(0) # put empty cell to 0

    dataset = dataset.replace("Store ID" , 5)

    dataset = dataset.replace("Total Price" , 6)

    dataset = dataset.replace("Base price" , 6)

    return dataset

data\_dict = new\_df.set\_index('ID').T.to\_dict()

df = encode\_data(new\_df , data\_dict)

df.head()

new\_df\_data = df.drop('ID' , axis =1)

label = data["ID"]

import plotly.express as px

import seaborn as sns

correlations = new\_df\_data.corr(method='pearson')

plt.figure(figsize=(5, 5))

sns.heatmap(correlations, cmap="coolwarm", annot=True)

plt.show()

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

X\_train, X\_test, y\_train, y\_test = train\_test\_split(new\_df\_data, label, shuffle=True, train\_size = 0.70 )

def fit\_eval\_model(model, train\_features, y\_train, test\_features, y\_test):

    """

    Function: train and evaluate a machine learning classifier.

    Args:

      model: machine learning classifier

      train\_features: train data extracted features

      y\_train: train data lables

      test\_features: train data extracted features

      y\_test: train data lables

    Return:

      results(dictionary): a dictionary of classification report

    """

    results = {}

    # Train the model

    model.fit(train\_features, y\_train)

    # Test the model

    train\_predicted = model.predict(train\_features)

    test\_predicted = model.predict(test\_features)

     # Classification report and Confusion Matrix

    results['classification\_report'] = classification\_report(y\_test, test\_predicted)

    results['confusion\_matrix'] = confusion\_matrix(y\_test, test\_predicted)

    return results

rf = RandomForestClassifier(random\_state = 1)

ab = AdaBoostClassifier(random\_state = 1)

gb = GradientBoostingClassifier(random\_state = 1)

# Fit and evaluate models

results = {}

for cls in [ rf, ab, gb]:

    cls\_name = cls.\_\_class\_\_.\_\_name\_\_

    results[cls\_name] = {}

    results[cls\_name] = fit\_eval\_model(cls, X\_train, y\_train, X\_test, y\_test)

for result in results:

    print (result)

    print()

    for i in results[result]:

        print (i, ':')

        print(results[result][i])

        print()

    print ('-----')

    print()