import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import StratifiedKFold

from sklearn.preprocessing import StandardScaler

import seaborn as sns

import pandas as pd

import os

df = pd.read\_csv("E:/Users/Shivangi/Geographical.csv")

df.head()

for i in df.columns:

print(df[i].unique(),"\t",df[i].nunique())

sns.countplot(df['-15.75'])

X = df.iloc[:, :68].values

Y = df.iloc[:, 68].values

from sklearn import preprocessing

from sklearn import utils

lab\_enc = preprocessing.LabelEncoder()

encoded = lab\_enc.fit\_transform(Y)

encoded

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

X\_train, X\_test, y\_train, y\_test = tts(X, encoded, test\_size = 0.2)

from sklearn.linear\_model import LogisticRegression

clf = LogisticRegression()

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

print("LogisticRegression")

print(clf.predict(X\_test))

print(clf.score(X\_test, y\_test))

from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()

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

print("DTC")

predicted = clf.predict(X\_test)

Predicted

fig=plt.figure(figsize=(10,6))

sns.heatmap(df.corr(),annot=True)

from sklearn.model\_selection import learning\_curve

lc=learning\_curve(clf,X\_train,y\_train,cv=10,n\_jobs=-1)

size=lc[0]

train\_score=[lc[1][i].mean() for i in range (0,5)]

test\_score=[lc[2][i].mean() for i in range (0,5)]

fig=plt.figure(figsize=(12,8))

plt.plot(size,train\_score)

plt.plot(size,test\_score)

from sklearn.model\_selection import KFold

kf = KFold(n\_splits=3)

def get\_score(model, X\_train, X\_test, y\_train, y\_test):

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

return model.score(X\_test, y\_test)

from sklearn.linear\_model import LogisticRegression

from sklearn import svm

from sklearn.ensemble import RandomForestClassifier

scores\_logistic = []

scores\_svm = []

scores\_rf = []

for train\_index, test\_index in kf.split(X):

X\_train, X\_test, y\_train, y\_test = X[train\_index], X[test\_index], \

encoded[train\_index], encoded[test\_index]

scores\_logistic.append(get\_score(LogisticRegression(), X\_train, X\_test, y\_train, y\_test))

scores\_svm.append(get\_score(svm.SVC(), X\_train, X\_test, y\_train, y\_test))

scores\_rf.append(get\_score(RandomForestClassifier(n\_estimators=40), X\_train, X\_test, y\_train, y\_test))

Scores\_rf

Scores\_svm

Scores\_logistic

from sklearn.neural\_network import MLPClassifier

from sklearn.model\_selection import GridSearchCV

para = { 'learning\_rate' : ['constant', 'invscaling', 'adaptive'],\

'activation' : ['identity', 'logistic', 'tanh', 'relu'], 'solver' : ['lbfgs', 'sgd', 'adam'] }

grid = GridSearchCV(MLPClassifier(), para, refit = True, verbose = 3)

print(grid.fit(X\_train, y\_train))

print("Best param %s"%grid.best\_params\_)

print("BEST SCORE FOUND {}".format(grid.best\_score\_))

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

actual = y\_test

results = confusion\_matrix(actual, predicted)

print ('Confusion Matrix :')

print(results)

print ('Accuracy Score :',accuracy\_score(actual, predicted))

print ('Report : ')

print (classification\_report(actual, predicted))