MACHINE LEARNING FOUNDATION

REPORT (MAJOR PROJECT)

TOPIC - Geographical Original of Music

Name- Shivangi Nagaich

Reg- 11615120(k1621A31)

About Geographical Original of music Dataset:

The dataset was built from a personal collection of 1059 tracks covering 33 countries/area. The music used is traditional, ethnic or `world' only, as classified by the publishers of the product on which it appears. Any Western music is not included because its influence is global - what we seek are the aspects of music that most influence location. Thus, being able to specify a location with strong influence on the music is central.

The geographical location of origin was manually collected the information from the CD sleeve notes, and when this information was inadequate we searched other information sources. The location data is limited in precision to the country of origin.

The country of origin was determined by the artist's or artists' main country/area of residence. Any track that had ambiguous origin is not included. We have taken the position of each country's capital city (or the province of the area) by latitude and longitude as the absolute point of origin.

Attribute Information:

The dataset is present in two files, where each file corresponds to a different feature sets.

Both files contain the audio features of 1059 tracks.

In the 'default\_features\_1059\_tracks.txt' file, the first 68 columns are audio features of the track, and the last two columns are the origin of the music, represented by latitude and longitude.

In the 'default\_plus\_chromatic\_features\_1059\_tracks.txt' file, the first 116 columns are audio features of the track, and the last two columns are the origin of the music.

Model Information:

I have used ​→

1. Support Vector Machine

2. Random ForestClassifier

3. DecisionTreeClassifier

4. MLP Classifier

5.Logistic Regression

6.GridSearchCv

Tool used:

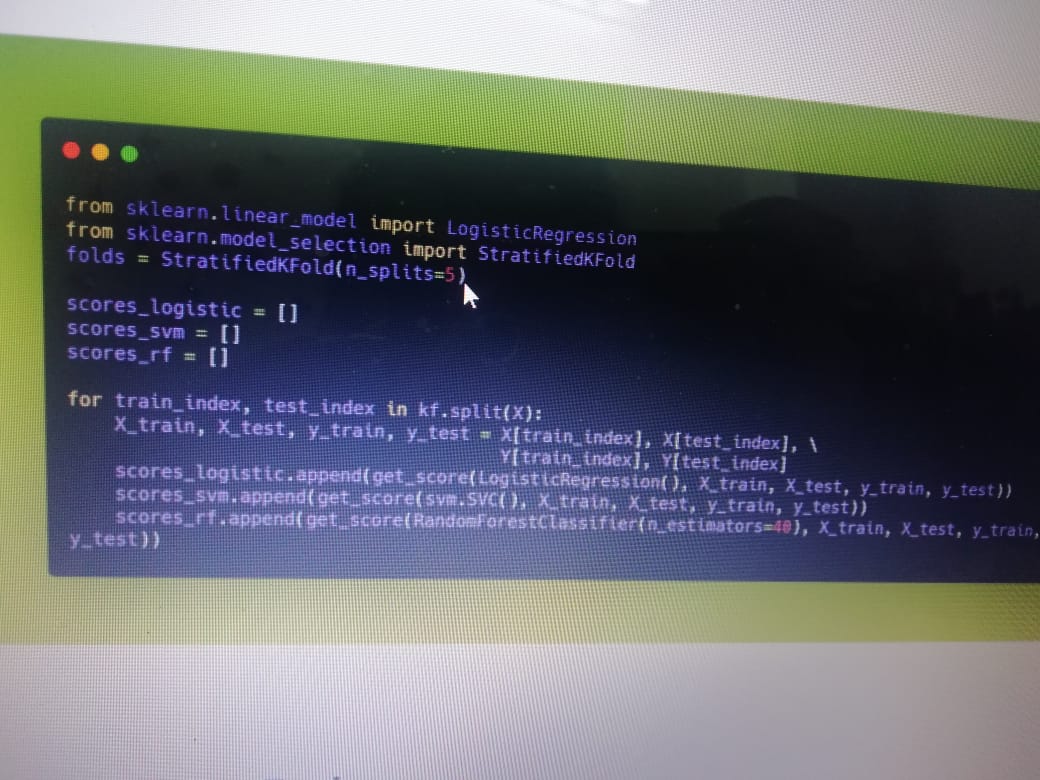
KFold cross-validation:

● That k-fold cross validation is a procedure used to estimate the skill of the model on new data.

● There are common tactics that you can use to select the value of k for your dataset.

● There are commonly used variations on cross-validation such as stratified and repeated that are available in sci-kit-learn

K-fold Part in code:



Hyperparameter Tuning:

Hyperparameter tuning is choosing a set of optimal hyperparameters for a learning algorithm. There are two kinds of tuning strategies

1. Grid Search 2. Randomized Search

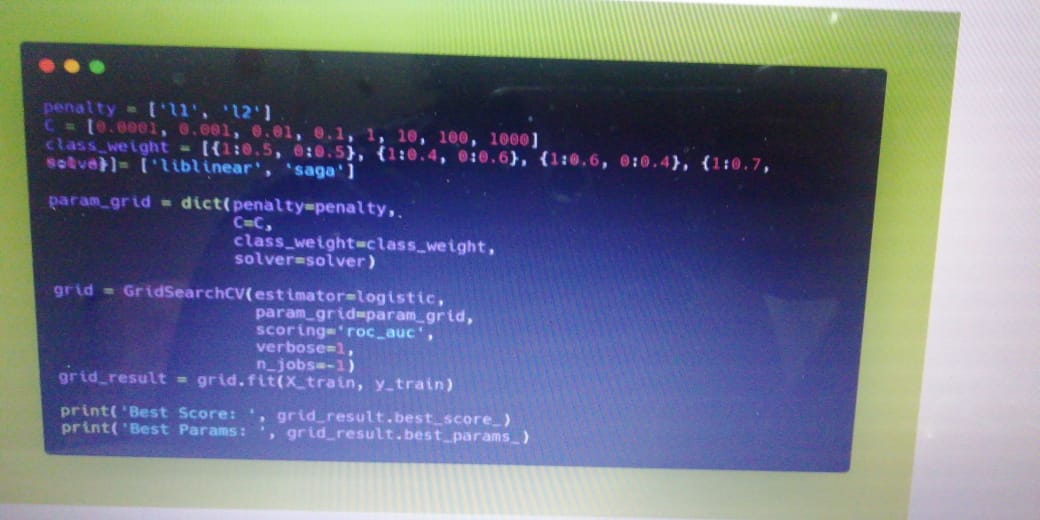
I have used the grid search cross-validation technique over here.

A grid search is a traditional way to perform hyperparameter optimization. It works by searching exhaustively through a specified subset of hyperparameters.Using sklearn’s ​GridSearchCV​, I have

defined my grid of parameters to search over and then run the grid search.

The benefit of grid search is that it is guaranteed to find the optimal combination of parameters supplied. The drawback is that it can be very time consuming and computationally expensive.

GRID SEARCH IN CODE:

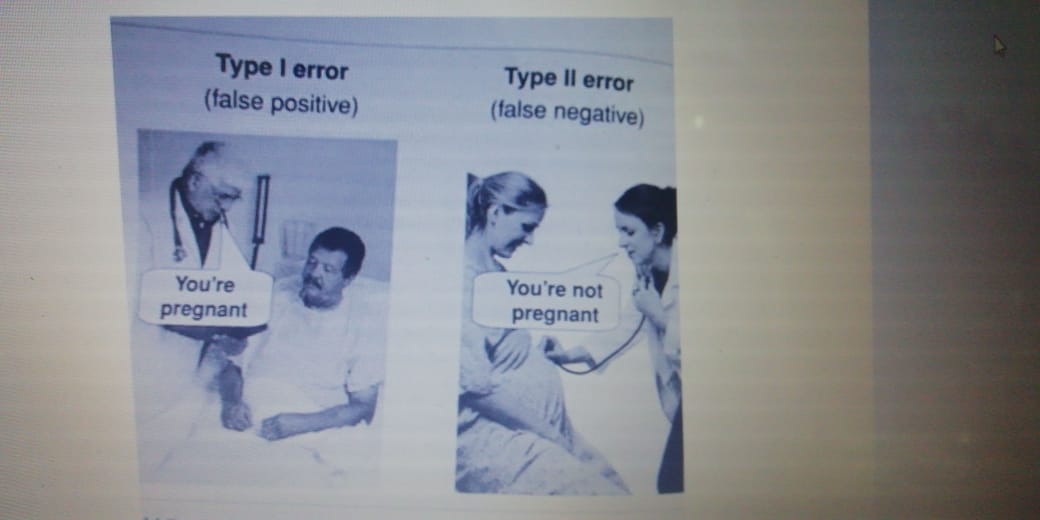


Confusion Matrix and Classification Report:

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix. A confusion matrix is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm. It allows easy identification of confusion between classes e.g. one class is commonly mislabeled as the other. Most performance measures are computed from the confusion matrix.

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions.

It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

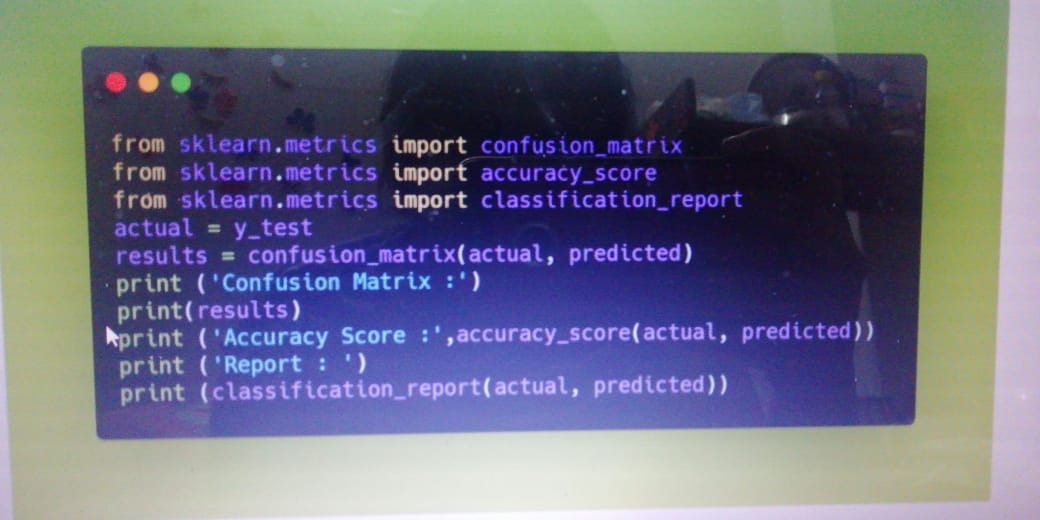


Definition of the Terms:

• Positive (P): Observation is positive (for example: is an apple). • Negative (N): Observation is not positive (for example: is not an apple). • True Positive (TP): Observation is positive, and is predicted to be positive.

• False Negative (FN): Observation is positive, but is predicted negative. • True Negative (TN): Observation is negative, and is predicted to be negative.

• False Positive (FP): Observation is negative, but is predicted positive.

* Code for Confusion Matrix and Classification Report:

CODE SNIPPETS:

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

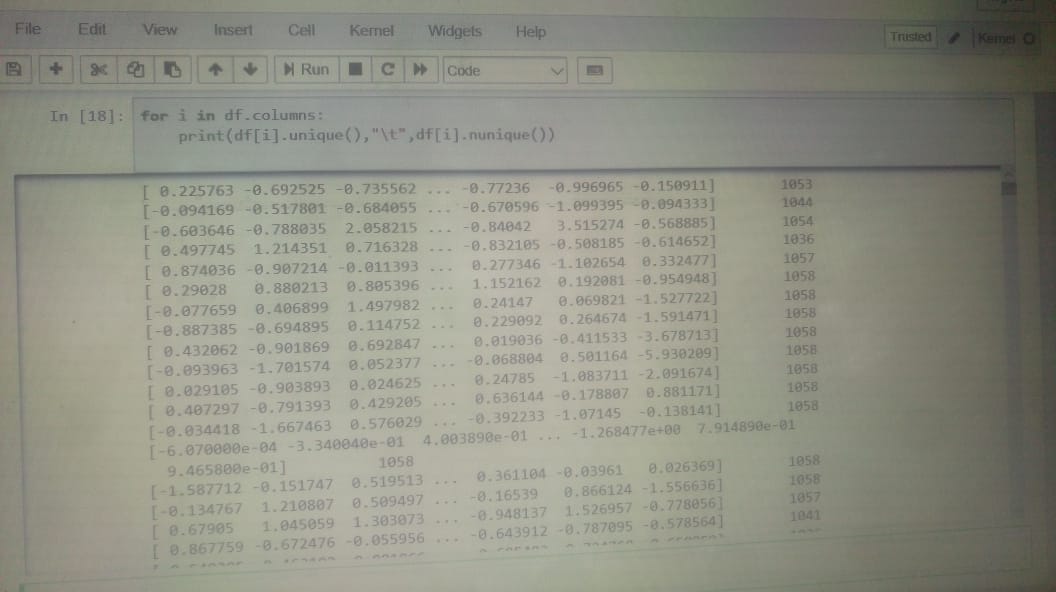
import os

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

df.head()

for i in df.columns:

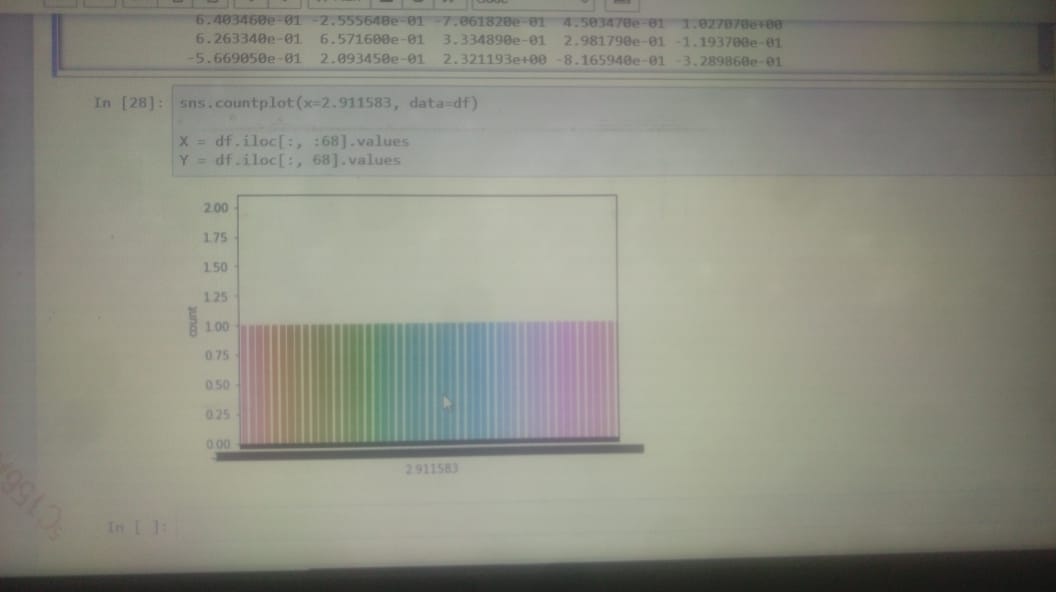
print(df[i].unique(),"\t",df[i].nunique())



sns.countplot(X = 2.911583,data =df)

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

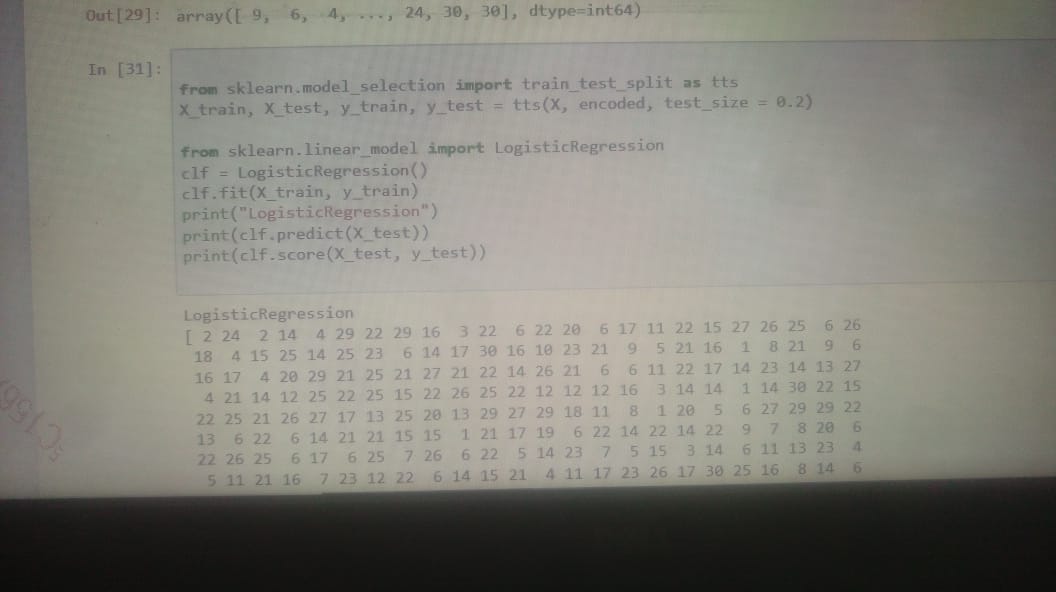
clf = LogisticRegression()

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)

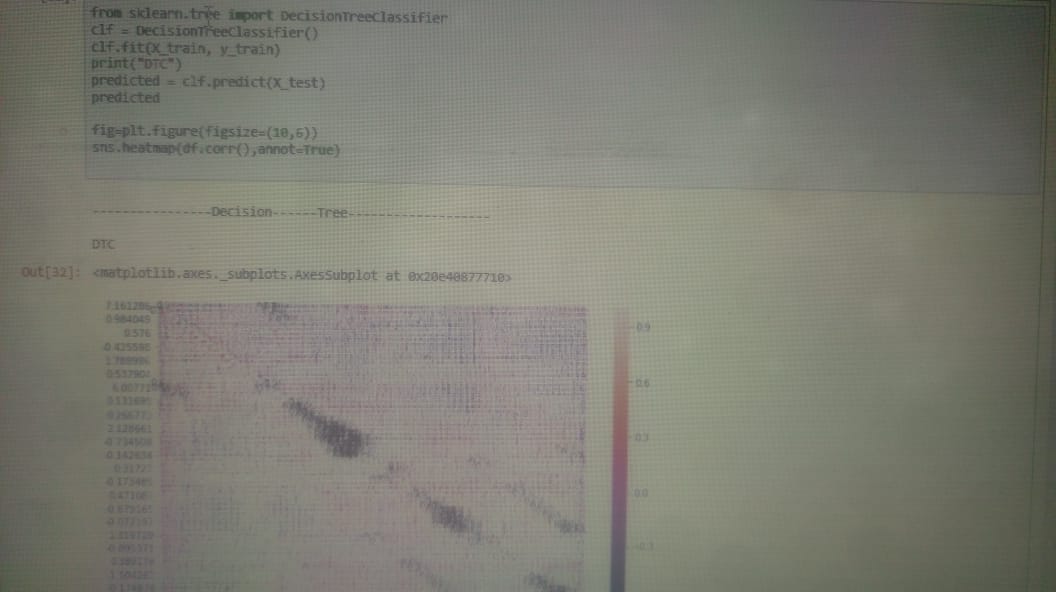
print("DTC")

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))

ACCURACY:

1.Randomforest- 100- ErrorReduction[0.3597733711048159, 0.38243626062322944, 0.35795454545454547]

So accuracy in rf [0.6402266288951, 0.61756377678, 0.65888220022]

2.SVM- 100-ErrorReduction[0.3286118980169972, 0.36827195467422097, 0.30113636363636365]

So accuracy in SVM[0.682467890, 0.6437773789, 0.7099228118]

3.LogisticRegression- 100-ErrorReduction[0.311614730878187, 0.3031161473087819, 0.3522727272727273]

So accuracy in Logistic[0.6999430245, 0.7079927334, 0.65883838383]

4.GridSearch- BEST SCORE FOUND [0.37535410764872523]