Farida Aliyeva

Homework 6

## imports

import numpy as np

import pandas as pd

from sklearn import metrics##used to show the accuracy

from sklearn.metrics import confusion\_matrix #used to describe performance of classifier

from sklearn.metrics import classification\_report##info abouve accuracy, precision and so on...

from sklearn.model\_selection import train\_test\_split## allows us to split the given data

from sklearn.feature\_extraction.text import CountVectorizer##converts text documents to matrix

## our data imported

info = pd.read\_excel(r'C:\Users\ASUS\Desktop\python\azeri\_news.xlsx',sep='\t',names=['Category','Title','news\_article'])

info.head()## first rows from our data

Output:



df\_x=info["news\_article"]+info["Title"]

df\_y=info["Category"]

## Now performing TFID vectorizer

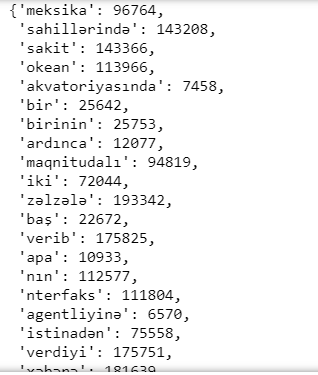
from sklearn.feature\_extraction.text import TfidfVectorizer

ve = TfidfVectorizer()

ve.vocabulary\_

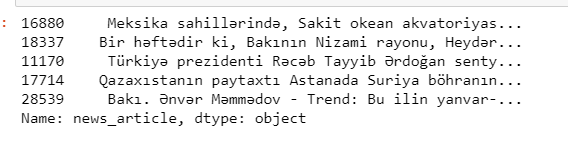
## mapping of terms to idices of features

Output:

 The screenshot is cut.

x\_train.head()

Output:



x\_traincv = ve.fit\_transform(x\_train)

SVM – Support Vector Machine

##performing SVM now

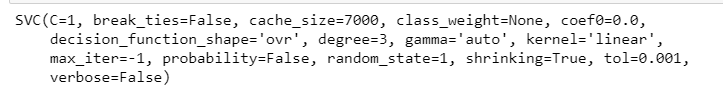
from sklearn.svm import SVC ## with kernel = 'RBF' and gamma = 'scale' the accuracy was 82%

## using cache size to speed up the run process as it takes veeery long

clf = SVC(kernel='linear',gamma= 'auto', C=1, cache\_size=7000, random\_state = 1)

clf.fit(x\_traincv,y\_train)

Output:

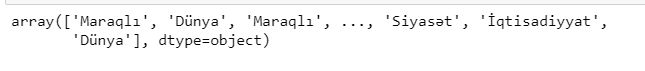


cv\_t = ve.transform(x\_test)

SVM\_predicted =clf.predict(cv\_t)

SVM\_predicted

Output:



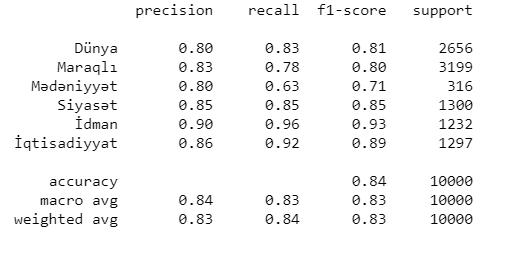
print("Accuracy:",metrics.accuracy\_score(y\_test, SVM\_predicted))

Output:



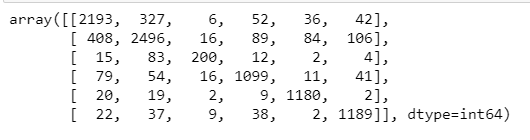
print(classification\_report(y\_test,SVM\_predicted))

Output:



confusion\_matrix(y\_test, SVM\_predicted)

Output:



MLP Artificial Neural Network

##importing neural network

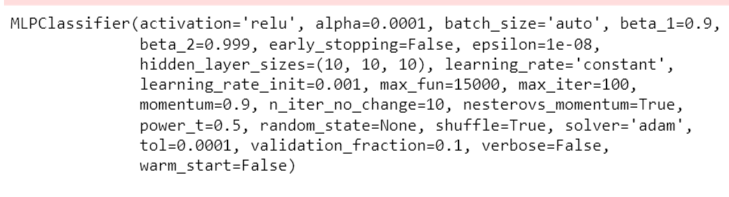
##without any hyperparameters the accuracy was only 76

from sklearn.neural\_network import MLPClassifier

mc = MLPClassifier(hidden\_layer\_sizes=(10,10,10),activation='relu', solver='adam', random\_state = 1, max\_iter=200)

mc.fit(x\_traincv, y\_train)

Output:

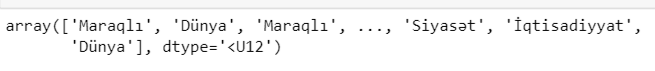


cv\_t = ve.transform(x\_test)

NN\_prediction1 = mc.predict(cv\_t)

NN\_prediction1

Output:



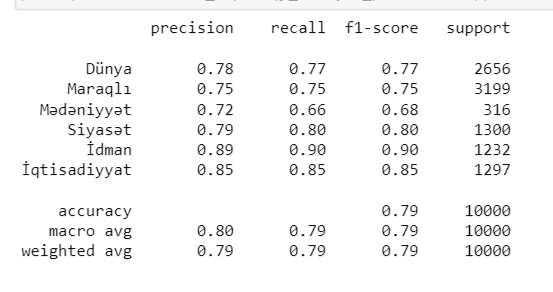
print("Accuracy:",metrics.accuracy\_score(y\_test, NN\_prediction1))

Output:



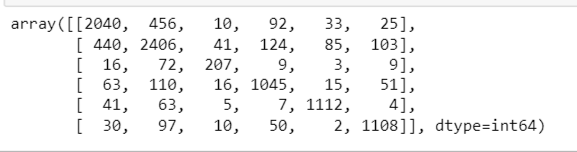
print(classification\_report(y\_test,NN\_prediction1))

Output:



confusion\_matrix(y\_test, NN\_prediction1)

Output:



DTC – Decision Tree Classifier

## with criterion = entropy and splitting = best accuracy was 72

## with criterion = gini and splitting = best accuracy was 73

##importing Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier

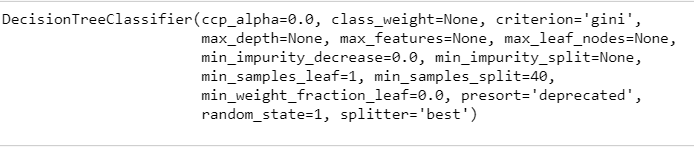
dtree = DecisionTreeClassifier(class\_weight=None, criterion= 'gini', max\_depth=None,

max\_features=None, max\_leaf\_nodes=None,splitter='best', min\_samples\_split=40,random\_state = 1)

cv\_t = ve.transform(x\_test)

dtree.fit(x\_traincv, y\_train)

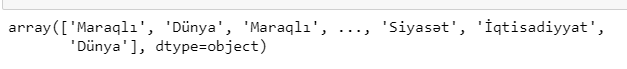
Output:



DTC\_prediction = dtree.predict(cv\_t)

DTC\_prediction

Output:



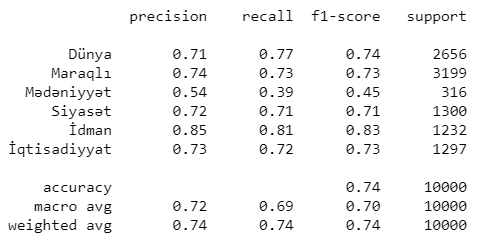
print("Accuracy:",metrics.accuracy\_score(y\_test,DTC\_prediction))

Output:



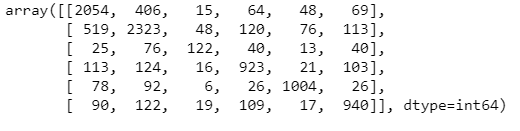
print(classification\_report(y\_test,DTC\_prediction))

Output:



confusion\_matrix(y\_test, DTC\_prediction)

Output:



Report:

Throughout this programming assignment I have been working on training the Azerbaijani news dataset on SVM, MLP – artificial neural network and Decision Tree Algorithm. According to the internet resources, SVM algorithm usually has higher prediction accuracy than a multilayer perceptron. Thus, in this report I will show the actual results from the experimental comparison on Azerbaijani news text classification.

First of all, I started with extracting the features. In this assignment we will be using TfidfVectorizer to perform the feature extraction. The main purpose of Tfidf algorithm is to measure the frequency and uniqueness of the word in a document. Tfidf Vectorizer is helpful for us, as it combines both features of Count Vectorizer that we used in previous assignment and Tdidf Transformer.

The next step was to properly tune our algorithms with hyperparameters to achieve higher accuracy result. For that reason, I started analyzing all the parameters of each of the algorithm. The first was the SVM algorithm which with kernel parameter being equal to ‘rbf’ and gamma parameter being equal to ‘scale’ produced 82% of accuracy. Then setting up the Kernel parameter to ‘linear’ and gamma parameter to ‘auto’ and random\_state to 1 the algorithm produced the highest accuracy of 84%. The second was the MLP Neural Network algorithm which without any parameter specifications produced an accuracy of 76% and then after specifying hidden layer size to 10, random state to 1 and maximum iterations to 200 it gave the highest possible result of 79% of accuracy. It was not possible to set a maximum number of iterations to a higher value than 200 as it took very long to run that and even jupyter notebook ended up giving an error. Lastly, the DTC algorithm produced an accuracy of 72 with criterion ‘enthropy’ and an accuracy of 74% with criterion ‘gini’.

After properly tuning the algorithms with hyperparameters and running all three algorithms on our datasets we can notice that linear SVM (Support Vector Machine) algorithm suits our data the best generating 84% of Accuracy. However, it took considerably long, around an hour to run this algorithm on our dataset. Right after SVM comes MLP -Neural Network whose accuracy reaches 79%, but on the other hand Neural Network was the slowest algorithm among all three used. It took about 2 – 3 hours to run this algorithm on our data. The last one – DTC (Decision Tree Classifier), was the fastest algorithm among all three, it took about 10 minutes to fit and then perform the prediction. However, produced the lowest accuracy percent compared to other algorithms – 74%. Considering all the information mentioned above, we can conclude that SVM is the most productive algorithm to be used in Text Classification for several reasons:

1. It took considerably less time to run this algorithm compared to MLP.
2. It produced the highest Accuracy result of 84%.

Table:

|  |  |  |
| --- | --- | --- |
| Algorithms: | Accuracy (%) | Time Spent(min) |
| SVM | 84 | 40-50 |
| MLP-Neural Network | 79 | 120+ |
| DTC | 74 | 10 |