Assignment 5

Farida Aliyeva

## 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.naive\_bayes import MultinomialNB## Naive Bayes

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

Importing necessary libraries

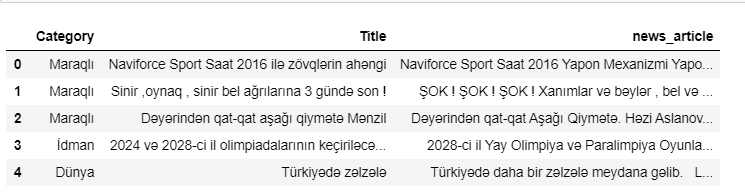
## our data imported

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

Reading our excel file

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

Output:



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

df\_y=info["Category"]

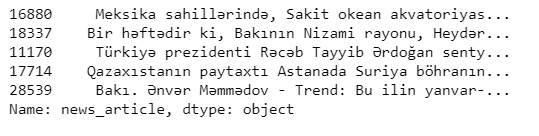
##Count Vectorizer

vectorizer = CountVectorizer()

x\_train, x\_test, y\_train, y\_test = train\_test\_split(df\_x, df\_y, test\_size=0.2, random\_state=4)

x\_train.head()

Output:

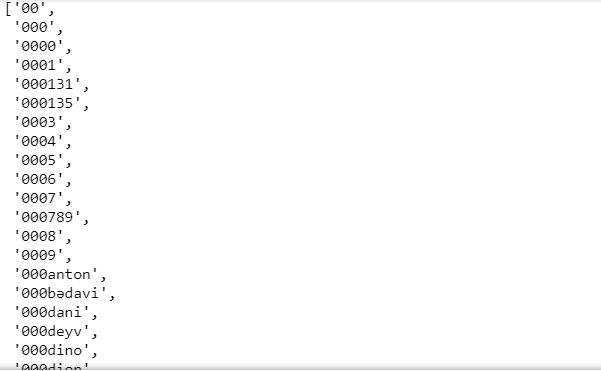


Here with the use of train test split library our scikit learn we split arrays of matrices into random train test subsets.

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

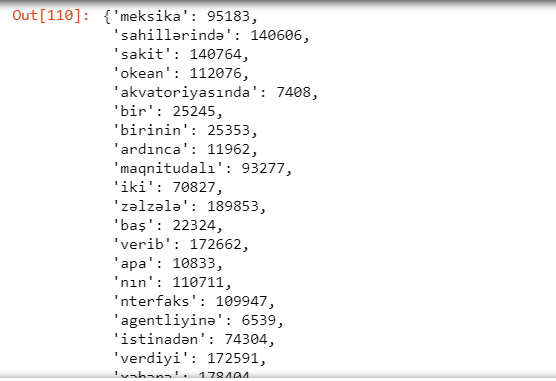
vectorizer.get\_feature\_names()##feature names selected from documents

Output:

The picture was cropped.

vectorizer.vocabulary\_##mapping on the terms of indices of the features

Output:

The picture was cropped.

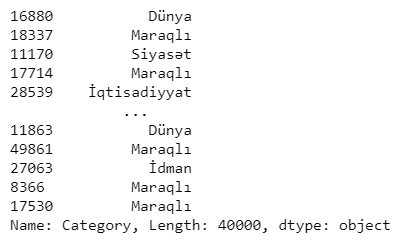
Now we Train a Naïve Bayes classifier on train dataset.

mnb = MultinomialNB()##Multinomial Naive Bayes

y\_train=y\_train.astype('str')

y\_train

Output:



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



crosv\_test = vectorizer.transform(x\_test)

preds=mnb.predict(crosv\_test)

preds

Output:



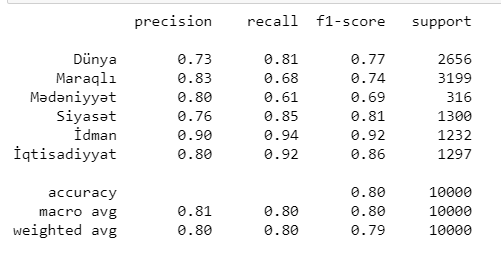
print("Accuracy:",metrics.accuracy\_score(y\_test,preds))

Output:



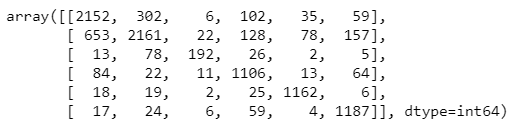
print(classification\_report(y\_test,preds))

Output:



confusion\_matrix(y\_test, preds)

Output:



I also found 2 more vectorizers: Tfid and Hashing vectorizers. Hashing vectorizer didn’t work, there was a negative value error. However, Tfid vectorizer worked for our data. And here is the result:

## Now performing the same with TFID vectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

ve = TfidfVectorizer()

x\_traincv2 = ve.fit\_transform(x\_train)

mnb.fit(x\_traincv2,y\_train)

Output:



cv\_t = vectorizer.transform(x\_test)

predicts=mnb.predict(cv\_t)

predicts

Output:



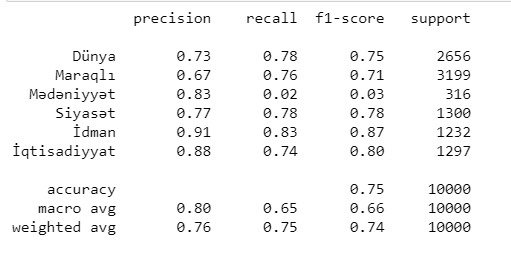
print("Accuracy:",metrics.accuracy\_score(y\_test,predicts))

Output:



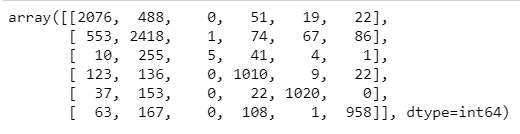
print(classification\_report(y\_test,predicts))

Output:



confusion\_matrix(y\_test, predicts)

Output:



As we can see from the classification report, the accuracy is less with Tfid vectorizer than with the CountVectorizer.