import pandas as pd

import numpy as np

from sklearn.pipeline import Pipeline

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.svm import LinearSVC

from sklearn.feature\_extraction.text import TfidfTransformer

from sklearn.multiclass import OneVsRestClassifier

#Load the dataset

df = pd.read\_csv("C:\\Users\\priyanshi burad\\Desktop\\IP\_LIVE\_PROJECT\_ MACHINE\_LEARNING\_Priyanshi\_Burad\\dataset.csv",header=None, error\_bad\_lines=False, encoding= 'unicode\_escape')

'''

print(df.head())

print(df.shape)

print(set(df[0]))

'''

from collections import Counter

'''

print(Counter(df[0]))

'''

import re

def clean\_str(string):

"""

Tokenization/string cleaning for dataset

Every dataset is lower cased except

"""

string = re.sub(r"\n", "", string)

string = re.sub(r"\r", "", string)

string = re.sub(r"[0-9]", "digit", string)

string = re.sub(r"\'", "", string)

string = re.sub(r"\"", "", string)

return string.strip().lower()

'''

print(df.columns)

'''

#train test split

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

X = []

for i in range(df.shape[0]):

X.append(clean\_str(df.iloc[i][1]))

y = np.array(df[0])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)

#feature engineering and model selection

from sklearn.svm import LinearSVC

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

#pipeline of feature engineering and model

model = Pipeline([('vectorizer', CountVectorizer()),

('tfidf', TfidfTransformer()),

('clf', OneVsRestClassifier(LinearSVC(class\_weight="balanced")))])

#paramater selection

from sklearn.model\_selection import GridSearchCV

parameters = {'vectorizer\_\_ngram\_range': [(1, 1), (1, 2),(2,2)],

'tfidf\_\_use\_idf': (True, False)}

gs\_clf\_svm = GridSearchCV(model, parameters, n\_jobs=-1)

gs\_clf\_svm = gs\_clf\_svm.fit(X, y)

'''

print(gs\_clf\_svm.best\_score\_)

print(gs\_clf\_svm.best\_params\_)

'''

#preparing the final pipeline using the selected parameters

model = Pipeline([('vectorizer', CountVectorizer(ngram\_range=(1,2))),

('tfidf', TfidfTransformer(use\_idf=True)),

('clf', OneVsRestClassifier(LinearSVC(class\_weight="balanced")))])

#fit model with training data

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

'''

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

'''

#evaluation on test data

pred = model.predict(X\_test)

'''

print(model.classes\_)

'''

from sklearn.metrics import confusion\_matrix, accuracy\_score

'''

print(confusion\_matrix(pred, y\_test))

'''

import sys

if not sys.warnoptions:

import warnings

warnings.simplefilter("ignore")

#print("Accuracy of model is" ,accuracy\_score(y\_test, pred)\*100)

#save the model

from sklearn.externals import joblib

joblib.dump(model, 'model\_question\_topic.pkl', compress=1)

question = input("Enter the event details")

t=model.predict([question])[0]

print(t)

print("Check output.xls file generated")

#sql

import xlrd

import xlwt

from xlwt import Workbook

loc = ("C:\\Users\\priyanshi burad\\Desktop\\IP\_LIVE\_PROJECT\_ MACHINE\_LEARNING\_Priyanshi\_Burad\\CCMLEmployeeData.xlsx")

wb = xlrd.open\_workbook(loc)

sheet = wb.sheet\_by\_index(0)

sheet.cell\_value(0,0)

'''

print(sheet.nrows)

print(sheet.ncols)

'''

count=0

j=0

wb1 = Workbook()

sheet1=wb1.add\_sheet("Sheet 1", cell\_overwrite\_ok=True)

l=[]

for i in range(sheet.nrows):

if(sheet.cell\_value(i,2)=='Internships' or sheet.cell\_value(i,3)=='Internships'):

l.append(sheet.cell\_value(i,0))

p=[','.join(l)]

sheet1.write(1,0,question)

sheet1.write(1,1,p)

wb1.save('output.xls')

file\_errors\_location = 'C:\\Users\\priyanshi burad\\Desktop\\IP\_LIVE\_PROJECT\_ MACHINE\_LEARNING\_Priyanshi\_Burad\\output.xls'

workbook\_errors = xlrd.open\_workbook(file\_errors\_location)