import logging

def extract\_features(df,field,training\_data,testing\_data,type="binary"):

"""Extract features using different methods"""

logging.info("Extracting features and creating vocabulary...")

if "binary" in type:

# BINARY FEATURE REPRESENTATION

cv= CountVectorizer(binary=True, max\_df=0.95)

cv.fit\_transform(training\_data[field].values)

train\_feature\_set=cv.transform(training\_data[field].values)

test\_feature\_set=cv.transform(testing\_data[field].values)

return train\_feature\_set,test\_feature\_set,cv

elif "counts" in type:

# COUNT BASED FEATURE REPRESENTATION

cv= CountVectorizer(binary=False, max\_df=0.95)

cv.fit\_transform(training\_data[field].values)

train\_feature\_set=cv.transform(training\_data[field].values)

test\_feature\_set=cv.transform(testing\_data[field].values)

return train\_feature\_set,test\_feature\_set,cv

else:

# TF-IDF BASED FEATURE REPRESENTATION

tfidf\_vectorizer=TfidfVectorizer(use\_idf=True, max\_df=0.95)

tfidf\_vectorizer.fit\_transform(training\_data[field].values)

train\_feature\_set=tfidf\_vectorizer.transform(training\_data[field].values)

test\_feature\_set=tfidf\_vectorizer.transform(testing\_data[field].values)

return train\_feature\_set,test\_feature\_set,tfidf\_vectorizer

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import pandas as pd

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

from sklearn.linear\_model import LogisticRegression

from sklearn.feature\_extraction.text import CountVectorizer,TfidfVectorizer

import logging

# Read the csv file into a pandas DataFrame

df= pd.read\_csv('../Output/tweets1.csv', encoding = "ISO-8859-1")

# Drop Null values from data frame

df= df.dropna(subset=['Tokenized'])

# GET A TRAIN TEST SPLIT (set seed for consistent results)

training\_data, testing\_data = train\_test\_split(df, random\_state = 2000)

# Assign X (data) and y (target)

### Get features

X\_train,X\_test,feature\_transformer=extract\_features(df,'Tokenized',training\_data,testing\_data,type='tfidf')

###Get Labels

y\_train = training\_data["Sentiment"].values

y\_test = testing\_data["Sentiment"].values

print(y\_train)

# INIT LOGISTIC REGRESSION CLASSIFIER

logging.info("Training a Logistic Regression Model...")

scikit\_log\_reg = LogisticRegression(verbose=1, solver='liblinear',random\_state=0, C=5, penalty='l2',max\_iter=1000)

model=scikit\_log\_reg.fit(X\_train,y\_train)

predictions = model.predict(X\_test)

print(predictions)

results=pd.DataFrame({"Prediction": predictions, "Actual": y\_test})

results.head(20)

# Calculate classification report

from sklearn.metrics import classification\_report

from sklearn.metrics import precision\_recall\_fscore\_support

from sklearn.metrics import accuracy\_score

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

print('accuracy:',accuracy\_score(y\_test, predictions))