!pip install kaggle

!mkdir -p ~/.kaggle

!cp kaggle.json ~/.kaggle/

!chmod 600 ~/.kaggle/kaggle.json

!kaggle datasets download -d kazanova/sentiment140

from zipfile import ZipFile

dataset = '/content/sentiment140.zip'

with ZipFile(dataset, 'r') as zip:

  zip.extractall()

print("The dataset is extracted")

import numpy as np

import pandas as pd

import re

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

from sklearn.feature\_extraction.text import TfidfVectorizer

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

import nltk

nltk.download("stopwords")

print(stopwords.words("english"))

twitter\_data = pd.read\_csv("/content/training.1600000.processed.noemoticon.csv", encoding="ISO-8859-1")

twitter\_data.shape

twitter\_data.head()

column\_names = ['target', 'id', 'date', 'flag', 'user', 'text']

twitter\_data = pd.read\_csv("/content/training.1600000.processed.noemoticon.csv",names=column\_names, encoding="ISO-8859-1")

twitter\_data.head()

twitter\_data.isnull().sum()

twitter\_data["target"].value\_counts()

twitter\_data.replace({"target":{4: 1}}, inplace=True)

twitter\_data["target"].value\_counts()

port\_stem = PorterStemmer()

def stemming(content):

  stemmed\_content = re.sub("[^a-zA-Z]", ' ', content)

  stemmed\_content = stemmed\_content.lower()

  stemmed\_content = stemmed\_content.split()

  stemmed\_content = [port\_stem.stem(word) for word in stemmed\_content if not word in stopwords.words("english")]

  stemmed\_content = ' '.join(stemmed\_content)

  return stemmed\_content

twitter\_data['stemmed\_content'] = twitter\_data["text"].apply(stemming)

twitter\_data.head()

print(twitter\_data["stemmed\_content"])

print(twitter\_data["target"])

X = twitter\_data["stemmed\_content"].values

Y = twitter\_data["target"].values

print(X)

print(Y)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2,stratify=Y, random\_state=2)

print(X.shape, X\_train.shape, X\_test.shape)

print(X\_train)

print(X\_test)

vectorizer = TfidfVectorizer()

X\_train = vectorizer.fit\_transform(X\_train)

X\_test = vectorizer.transform(X\_test)

print(X\_train)

print(X\_test)

model = LogisticRegression(max\_iter=1000)

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

X\_train\_prediction = model.predict(X\_train)

training\_data\_accuracy = accuracy\_score(Y\_train, X\_train\_prediction)

print("Accuracy score of train data : ", training\_data\_accuracy)

X\_test\_prediction = model.predict(X\_test)

test\_data\_accuracy = accuracy\_score(Y\_test, X\_test\_prediction)

print("Accuracy score of test data : ", test\_data\_accuracy)

import pickle

filename = "trained\_model.sav"

pickle.dump(model, open(filename, "wb"))

loaded\_model = pickle.load(open("/content/trained\_model.sav", "rb"))

X\_new = X\_test[200]

print(Y\_test[200])

prediction = model.predict(X\_new)

print(prediction)

if(prediction[0] == 0):

  print("Negative Tweet")

else:

  print("Positive Tweet")

twitter\_data.dtypes

from wordcloud import WordCloud

import matplotlib.pyplot as plt

# Separate tweets by sentiment

positive\_tweets = twitter\_data[twitter\_data['target'] == 1]['stemmed\_content']

negative\_tweets = twitter\_data[twitter\_data['target'] == 0]['stemmed\_content']

# Generate word clouds

positive\_wordcloud = WordCloud(width=800, height=500, random\_state=21,max\_font\_size=110).generate(' '.join(positive\_tweets))

negative\_wordcloud = WordCloud(width=800, height=500, random\_state=21,max\_font\_size=110).generate(' '.join(negative\_tweets))

# Display the generated image

plt.figure(figsize=(10, 7))

plt.imshow(positive\_wordcloud, interpolation="bilinear")

plt.axis('off')

plt.show()

plt.figure(figsize=(10, 7))

plt.imshow(negative\_wordcloud, interpolation="bilinear")

plt.axis('off')

plt.show()

import seaborn as sns

import matplotlib.pyplot as plt

# Plot the distribution of sentiments

sns.countplot(x='target', data=twitter\_data)

plt.title('Distribution of Sentiments')

plt.xlabel('Sentiment')

plt.ylabel('Count')

plt.show()

from sklearn.metrics import confusion\_matrix

# Assume y\_test are the actual values and y\_pred are the predicted values

cm = confusion\_matrix(Y\_test, X\_test\_prediction)

import matplotlib.pyplot as plt

import numpy as np

# Normalize the confusion matrix

cm\_norm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

fig, ax = plt.subplots(figsize=(8, 8))

sns.heatmap(cm\_norm, annot=True, fmt=".2f", linewidths=.5, square=True,cmap='Blues', ax=ax)

plt.ylabel('Actual label')

plt.xlabel('Predicted label')

all\_sample\_title = 'Accuracy Score: {0}'.format(np.trace(cm\_norm)/2)

plt.title(all\_sample\_title, size=15)

plt.show()