SPAM HAM CLASSIFIER

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import re

import string

import nltk

from nltk.corpus import stopwords

from sklearn.feature\_extraction.text import CountVectorizer

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

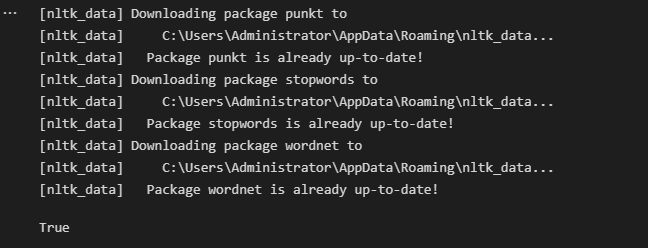
from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import accuracy\_score, f1\_score, precision\_score

nltk.download('punkt')

nltk.download('stopwords')

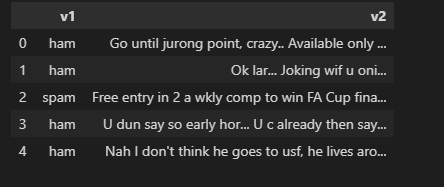
nltk.download('wordnet')



df = pd.read\_csv('spam.csv', encoding='latin-1')

df = df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1)

df.head()



def cleaning\_reviews(df):

messages = list()

lines = df['v2'].values.tolist()

for i in lines:

text = i.lower()

pattern = re.compile('http[s]?://(?:[A-Za-z]|[0-9]|[$-\_@.&+]|[!\*\(\),]|(?:[0-9a-fA-F][0-9a-fA-F]))+')

text = pattern.sub("", text)

tokens = nltk.word\_tokenize(text)

table = str.maketrans('', '', string.punctuation)

stripped = [w.translate(table) for w in tokens]

words = [word for word in stripped if word.isalpha()]

stop\_word = set(stopwords.words('english'))

stop\_word.discard("not")

words = [nltk.WordNetLemmatizer().lemmatize(w) for w in words if not w in stop\_word]

words = ' '.join(words)

messages.append(words)

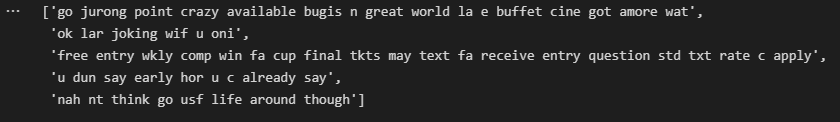
return(messages)

corpus = cleaning\_reviews(df)

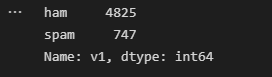
corpus.\_\_len\_\_()



corpus[0:5]



df.v1.value\_counts()



CV = CountVectorizer(max\_features= 500)

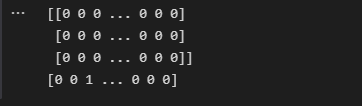
X = CV.fit\_transform(corpus).toarray()

Y=pd.get\_dummies(df['v1'])

Y = Y.iloc[:, 1].values

print(X[0:3])

print(Y)

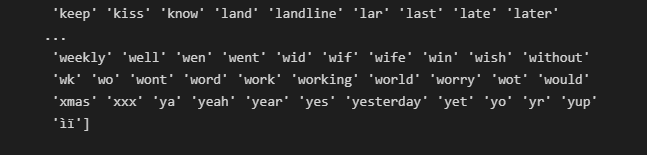


feature\_names = CV.get\_feature\_names\_out()

print(feature\_names.\_\_len\_\_())

print(feature\_names)





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

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.3, random\_state=42)

from sklearn.naive\_bayes import MultinomialNB

spam\_detect = MultinomialNB().fit(X\_train, Y\_train)

Y\_pred = spam\_detect.predict(X\_test)

print(accuracy\_score(Y\_test, Y\_pred))

print(f1\_score(Y\_test, Y\_pred))

print(precision\_score(Y\_test, Y\_pred))

