***CODE FOR EMAIL SPAM VERIFICATION***

#importing required packages import numpy as np

import pandas as pd

import matplotlib.pyplot as plt import nltk

from nltk.corpus import stopwords import string

from nltk.tokenize import word\_tokenize nltk.download('punkt')

#dataset: https://[www.kaggle.com/karthickveerakumar/spam-filter](http://www.kaggle.com/karthickveerakumar/spam-filter) emails = pd.read\_csv('/content/emails.csv')

emails

emails.info()

emails = emails.drop\_duplicates(keep = 'last') #remove all duplicate emails from the dataframe emails

#data visualization using matplotlib emails.spam.value\_counts().plot(kind='pie',

explode=[0,.1], figsize=(6,6),

autopct='%.2f%%') plt.title('Normal Mails vs Spam mails') plt.legend(['Normal','Spam'])

plt.show()

emails.spam.value\_counts()

# allocating data to the variables

spam\_messages = emails[emails['spam']==1]['text'] notspam\_messages = emails[emails['spam']==0]['text']

spam\_words = [] notspam\_words = []

#creating a function for tokenizing the text using nltk def tokenize\_spam\_words(text):

words = [w.lower() for w in word\_tokenize(text) if w.lower() not in stopwords.words('english') and w.lower().isalpha()]

spam\_words.extend(words)

def tokenize\_notspam\_words(text):

words = [w.lower() for w in word\_tokenize(text) if w.lower() not in stopwords.words('english') and w.lower().isalpha()]

notspam\_words.extend(words) #tokenizing the spam messages import nltk nltk.download('stopwords')

spam\_messages.apply(tokenize\_spam\_words) print(spam\_words[:100])

#tokenizing the not spam messages notspam\_messages.apply(tokenize\_notspam\_words) print(notspam\_words[:100])

#stemming

from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

# creating a function for stemming the words def cleanup\_text(message):

message = message.translate(str.maketrans('','',string.punctuation))

words = [stemmer.stem(w) for w in message.split() if w.lower() not in stopwords.words('english') ] return ' '.join(words)

emails.text = emails.text.apply(cleanup\_text) emails.head()

emails.tail()

#feautre extraction using count vectorizer

from sklearn.feature\_extraction.text import CountVectorizer vect = CountVectorizer(stop\_words = 'english')

features = vect.fit\_transform(emails.text) features.shape

# saving the feautures using the pickle import pickle

with open('count\_vectorizer.pkl','wb') as f: pickle.dump(vect,f)

print('done')

# data preprocessing for training the model

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

#labeling the data

category = LabelEncoder()

emails.spam = category.fit\_transform(emails.spam) emails.head()

#splitting the data into training and testing data

x\_train, x\_test, y\_train,y\_test = train\_test\_split(features.toarray(), emails.spam,test\_size=.2)

#creating a machine learning model

from sklearn.naive\_bayes import GaussianNB

from sklearn.model\_selection import cross\_val\_score from sklearn.metrics import confusion\_matrix

model = GaussianNB() model.fit(x\_train,y\_train) y\_pred = model.predict(x\_test)

from sklearn.metrics import accuracy\_score

# Assuming y\_test and y\_pred are already defined accuracy = accuracy\_score(y\_test, y\_pred) print("Accuracy:", accuracy)

#confusion matrix confusion\_matrix(y\_test,y\_pred) #saving the builded model using pickle import pickle

with open('spam\_classifier.pkl','wb') as f: pickle.dump(model,f)

print('done')