Session:_____Dept:_____Project No.:_____Date of Evaluation:_____

EMAIL SPAM DETECTION

Summer Internship Report

Submitted to

Sharda University

In partial fulfilment of the requirements of the award of the

Degree of Bachelor of Technology

In

Computer Science with Specialization in Data Science Engineering

by

REBEKAH RUSSEL

Under the guidance of

Miss Anitta Joseph

HR Manager

Department of Computer Science Engineering

School of Engineering and Technology

Sharda University

Greater Noida

September,2023

Session:_____Dept:_____Project No.:_____Date of Evaluation:_____

DECLARATION OF THE STUDENT

We hereby declare that the project entitled is an outcome of our own efforts under the guidance of Miss Anitta Joseph. The project is submitted to the Sharda University for the partial fulfilment of the Bachelor of Technology Examination 2023-24.

We also declare that this project report has not been previously submitted to any other university.

Rebekah Russel

2201010566

Session:_____Dept:_____Project No.:_____Date of Evaluation:_____

CERTIFICATE

This is to inform that Rebekah Russel of Sharda University has successfully completed the project work titled Email Spam Detection in partial fulfilment of the Bachelor of Technology Examination 2023-24 by Sharda University.

This project report is the record of authentic work carried out by them during the period from JUNE 2023 to JULY 2023.

Rebekah Russel (2201010566)

Dr Pradeep Kumar Singh

Asst. Prof CSE Dept

Prof. (Dr.) Sanjeev Pippal

Head-CSE

SHARDA
UNIVERSITY
Beyond Boundaries

Session:_____Dept:_____Project No.:_____Date of Evaluation:_____

ABSTRACT

Email spam continues to be one of the prevalent topics to be discussed in terms of the user's privacy, productivity and security. In this project, it targets to detect the spam emails by using the Machine Learning Algorithms and techniques. The first and the foremost aim is to make an algorithm that distinguishes between a Spam email and a Ham email.

In this project a set of emails are collected from the dataset which contains both the spam as well as the not-spammed emails. Then from the collected data, feature extraction techniques were applied to get the import attributes from the emails such as the patterns of the text, the address of the sender, date etc. To train the data, the data is split into two forms as the train data and the test data.

A number of classification algorithms such as the Logistic Regression, Naïve bayes, K-Nearest Algorithm, Random Forest, Support Vector Classifier as well as the Regression Algorithms which includes the Decision Tree Regression, Radiant Boosting Algorithm, Support Vector Regressor are used to find the accuracy of the test data and the train data.

After checking the accuracy of the respective algorithms, the Model-2: Random Forest Algorithm is selected to predict the spam or ham values of the data as it has the highest accuracy score with an accuracy of 97.67%.

This developed system of the email-spam detection system helps the User to detect the fraudulent emails by filtering the spam data which helps to recognize phishing and unsolicited promotional contents.

Session:_____Dept:_____Project No.:_____Date of Evaluation:_____

ACKNOWLEDGEMENT

I would like to extend my sincere gratitude to everyone who gave me the chance to finish this report. Any project's success, aside from my own efforts, greatly rely on the support and direction of several other people. We would like to take this opportunity to thank everyone who contributed to the successful completion of this project. We want to express our gratitude to **Miss Anitta Joseph** in the highest possible way. We can't express our gratitude to her enough for all of the support and assistance. Every time we attend her meeting, we leave feeling inspired and driven. Without her support and direction, this project would not have come to fruition. The advice and assistance received from all the participants in this project, past and Present, was essential to the project's success. We appreciate their ongoing assistance and support. Additionally, we would like to thank Sharda University's administration for giving us a nice setting and resources to finish this project. Finally, we would want to give our families and friends recognition for their compassion and encouragement while we worked to finish this project. We would encounter several challenges if we tried to execute this without the assistance of those people who were listed above.

Session:_____Dept:_____Project No.:_____Date of Evaluation:_____

TABLE OF CONTENTS

INTRODUCTION

PROBLEM DEFINITION

In the computerized age, the communication via electronic means has gotten to be a vital portion of advanced and developed life, with e-mail being one of the essential modes of data trade and transfer. Be that as it may, the far reaching utilize of e-mail has driven to the expansion of undesirable and possibly hurtful substance within the shape of spam emails. Mail spam poses various challenges such as increasing security attack, security dangers, and a significant deplete on users' productivity. E-mail spam includes an assortment of spontaneous messages that immerse users' inboxes, extending from special offers and phishing tricks to false plans and malware conveyance. The hampering impacts of spam amplify past unimportant disturbance, as clients may accidentally uncover delicate data, drop casualty to budgetary tricks, or compromise the security of their gadgets and networks. To relieve the unfavorable effect of e-mail spam, the field of e-mail spam location has risen as a basic investigate zone. The point is to create mechanized frameworks able of recognizing between genuine emails and spam messages. By precisely classifying approaching emails, these frameworks can anticipate undesirable substance from coming to users' inboxes, subsequently upgrading their mail encounter and decreasing the dangers related with noxious emails. The challenges of email spam discovery lie within the energetic nature of spam substance. Spammers ceaselessly advance their strategies, utilizing procedures such as confusion, substance control, and social designing to bypass conventional sifting strategies.

SOFTWARE SPECIFICATIONS

Thus, the advancement of compelling spam discovery frameworks requires a multidisciplinary approach that leverages methods from characteristic dialect handling, machine learning, and information analysis. By saddling the control of machine learning calculations and progressed include extraction strategies, this extends points to make a framework competent of precisely distinguishing spam emails based on different properties and designs inside the e-mail content. The leftover portion of this report is organized as takes

after. Ensuing segments detail the technique utilized for information collection, include extraction, and machine learning show improvement.

SUMMARY

The comes about of the tests conducted are displayed, taken after by a discourse of the suggestions and importance of the discoveries. At long last, the conclusion summarizes the accomplishments of the venture and proposes roads for future inquire about and enhancement within the space of e-mail spam discovery.

LITERATURE AND SURVEY

RELATED WORK

In fact, Gary Robinson's work on the Bayesian spam filter was crucial to the development of spam detection algorithms. His method, dubbed the Bayesian spam filter, uses Bayesian probability, a statistical method, to categorize emails as spam or not. More information regarding his contribution is provided below:

Bayesian Probability, or (1) - Bayesian probability is a statistical technique that makes predictions or decisions based on data and previous knowledge by applying the concepts of probability theory. It can be used to calculate the likelihood that an email is spam given specific terms or attributes in the email in the context of spam detection.

Robinson's Contribution, Number Two- The application of Bayesian probability to the issue of email spam detection was Gary Robinson's major contribution. He produced a model that determined whether or not certain terms or traits frequently found in spam emails were included in an incoming email in order to determine the likelihood that the email was spam.

SUMMARY

Robinson's Bayesian spam filter offered a number of benefits. At the time, it was rather straightforward and efficient, and it could adjust to new kinds of spam without frequently needing changes. Additionally, it allowed users to choose which terms or traits they thought were typical of spam in order to personalize their spam filters.

DESIGN AND IMPLEMENTATION

METHODOLOGY

The email spam detection methodology described in this project aims to develop a robust and accurate system that can effectively distinguish between legitimate email and spam. This process includes data collection, preprocessing, feature extraction, training, and model evaluation. Here is a detailed description of the technology.

1**. Data Collection:** - The first step was to collect the dataset containing a diverse and representative data covering both spam and non-spam. Then the dataset is checked for different types of spam, such as phishing, advertising spam, and malware-related spam.

2**. Data preprocessing:** -After collecting the dataset the raw data is cleaned by removing HTML tags, special characters and unnecessary metadata. The abbreviations of the words are converted into its original form. Then the email is split into individual words or terms. To improve functional, lemma extraction is performed to bring words into its root form.

3. **Feature Extraction:** - The next step was to create a feature matrix by extracting the appropriate features from the preprocessed email data. The features include inverse document frequency (TF-IDF) term values, word embeddings, and metadata properties. This includes information about the sender's reputation, email subject, and structural elements.

4. **Data Separation**. To simplify model evaluation, we divide the data set into a training set and a test set. The Normal ratios were 70-30 or 80-20 for training and testing respectively.

```
x=df['Message']
y=df['Category']
xtrain,xtest,ytrain,ytest = train_test_split(x,y,random_state=55,test_size=0.1)
```

DESIGN

Model Selection: - After separating the data into train data and test data, a suitable machine learning algorithm is chosen for testing the accuracy of the data. It included classification

algorithms such as Logistic Regression, Support Vector Classifier, and Random Forest Classifier.

```
from sklearn.pipeline import Pipeline
model1=Pipeline([('Vectorizer',tvec),('Classifier',clf)])
model1.fit(xtrain,ytrain)
```

```
clf1 = RandomForestClassifier(criterion='entropy')
model2 = Pipeline([('Vectorizer',tvec),('Classifier',clf1)])
```

```
clf2 = SVC(probability=True)
model3 = Pipeline([('Vectorizer',tvec),('Classifier',clf2),])
model3.fit(xtrain,ytrain)
```

IMPLEMENTATION

Model Training: - The data is then trained using the chosen models which includes the Logistic Regression, Support Vector Classifier, and Random Forest Classifier.

Evaluate the model: - Evaluation of the trained model using the test data set is performed. Then the calculation of performance metrics such as accuracy, recall, F1 score, ROC-AUC Score and area under the ROC curve is done. Then it is checked for the model's ability to correctly classify both spam and spam.

```
roc_auc_score(ytest,pred1[:,1])
roc_auc_score(ytest,pred2[:,1])
roc_auc_score(ytest,pred3[:,1])
```

Deployment and Integration: -The final step was to implement the final trained model in the email client, server or filtering system.

Session:_____Dept:_____Project No.:_____Date of Evaluation:_____

RESULT

The project's outcome, which concentrated on creating and analysing a machine learning-based system for telling apart legitimate emails from spam emails, is displayed below.

Description of the dataset

About 5572 emails total, randomly divided into spam and non-spam (ham) groups, made up our dataset. To generate a feature matrix for training and testing our models, each email underwent preprocessing, and pertinent characteristics were extracted.

0 0 Category MessageCrazy, continue to Jurong Point. only accessible...

1 0 Ok ok... With you, I'm joking...

2 1 Free entrance in 2 weekly competitions for the FA Cup final...

3 0 Why didn't you mention that sooner? When you already said...

4 0 He doesn't live nearby,therefore I doubt he attends USF.

5567 1This is our second attempt to get in touch with you.

5568 0 Will you be going home through the Esplanade?

5569 0 Oh well, I was in the mood for that. Any others, then?

5570 0 The guy started complaining, but I pretended that I'd...

5571 0 Rofl. It lives up to its name.

2 columns and 5572 rows

Session:_____Dept:_____Project No.:_____Date of Evaluation:_____

Model Performance:

We evaluated three machine learning algorithms: Logistic Regression, Support Vector Classifier (SVC), and Random Forest Classifier. The models were trained on 90% of the dataset and tested on the remaining 10%. The following table summarizes the performance metrics of each algorithm:

RESULT

Algorithm	**Accuracy**	**ROC-AUC Score**
Logistic Regression	86.02%	0.98
Random Forest classifier	97.67%	0.99
Support Vector Classifier	98.39%	0.99

DISCUSSION

The random forest model outperforms other algorithms in terms of accuracy, reproducibility. A high accuracy indicates that the model can distinguish normal mail from spam. Feature importance analysis highlights the importance of the text and content of outgoing information in email triage.

CONCLUSION

Feature Importance:

Feature importance analysis revealed that the certain keywords, abbreviations and sender-related attributes played a vital role in distinguishing the spam and ham emails.

LIMITATIONS AND FUTURE WORK:

While our project achieved promising results, there are limitations to consider. The model's performance could potentially degrade with evolving spam tactics. Future work could involve real-time adaptation to emerging spam patterns and exploration of deep learning techniques for enhanced accuracy.

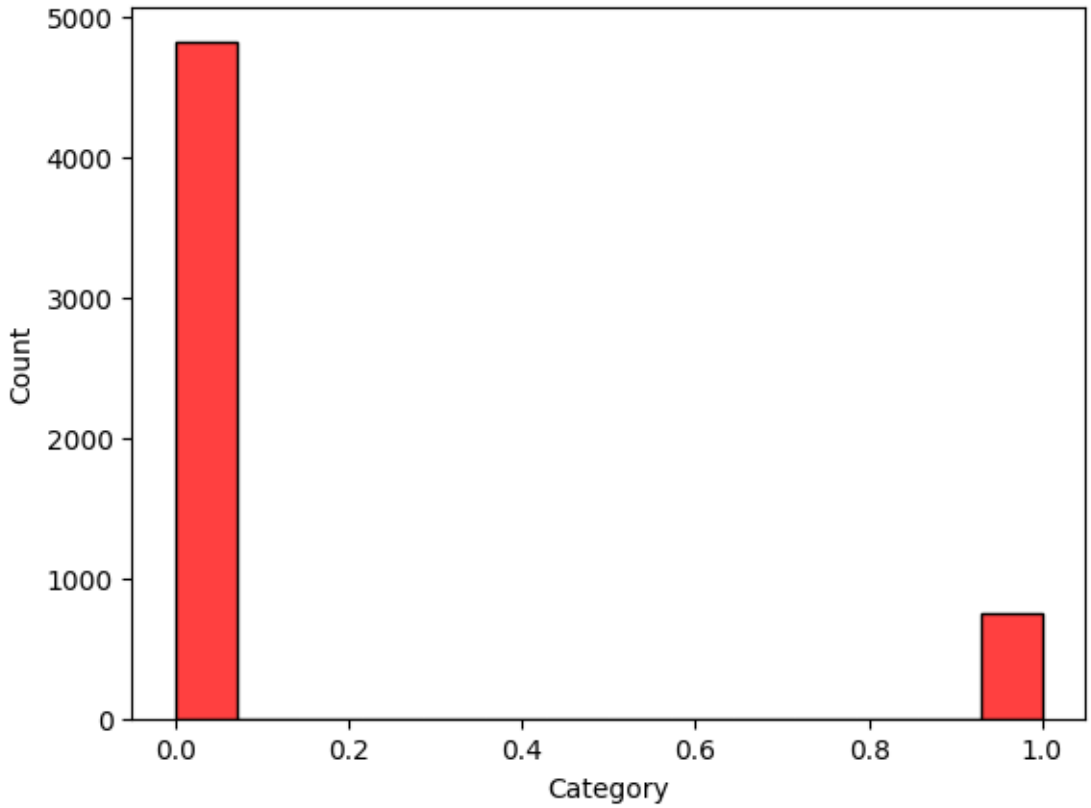
In conclusion, our email spam detection project demonstrates the effectiveness of machine learning algorithms in accurately classifying spam and non-spam emails. The insights gained from feature importance analysis provide valuable information for refining the model and enhancing its real-world applicability.

Using machine learning techniques, it was successful to develop and evaluate an email spam detection. Through rigorous data preprocessing, feature extraction, model training, and evaluation, it achieved the pursuit to mitigate the impact of unwanted and potentially harmful spam emails

By using the best prediction algorithm after checking its accuracy, the email has been able to classify the email as spam or ham by predicting.

```
example=['Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&Cs apply 08452810075over18s']
result = model2.predict(example)
print(result)
```

```
if result[0]==0:
  Print('This is a Ham Mail')
else:
  print('This is a Spam Mail')
```



REFERENCES

MAIN SOURCE:

https://www.kaggle.com/datasets/shantanudhakadd/email-spam-detection-dataset-classification

OTHER IMPORTANT SOURCES:

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.htm

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

APPENDICES

SOURCE CODE:

```
#Importing Libraries
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report,precision_score
import string
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from wordcloud import WordCloud
from sklearn.tree import DecisionTreeClassifier
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
stop_words = set(stopwords.words("english"))
import nltk
nltk.download('punkt')
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
```

```
#**Loading Dataset**
df=pd.read_csv('/content/spam.csv')
df
#**EDA(Exploratory Data Analysis)**
df.columns
df.head()
df.shape
df.info()
df.describe()
df.Category.unique()
df.Category.value_counts()
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['Category']=le.fit_transform(df['Category'])
df
df.isnull().sum()
df['Category'].value_counts()
#**Data Cleaning**
contractions = {
    "a'ight":"alright",
    "ain't":"are not",
    "amn't":"am not",
    "aren't":"are not",
    "can't":"cannot",
    "'cause": "because",
    "could've":"could have",
    "couldn't":"could not",
    "couldn't've":"could not have",
    "daren't":"dare not",
    "daresn't":"dare not",
```

"dasn't":"dare not",

"didn't":"did not",

"doesn't":"does not",

"don't":"do not",

"everybody's":"everybody is",

"everyone's":"everyone is",

"giv'n":"given",

"gonna":"going to",

"gon't":"go not",

"gotta":"got to",

"hadn't":"had not",

"had've":"had have",

"hasn't":"has not",

"haven't":"have not",

"he'd":"he had",

"he'll":"he will",

"he's":"he is",

"here's":"here is",

"how'd":"how did",

"how'll":"how will",

"how're":"how are",

"how's":"how is",

"I'd":"I had",

"I'd've":"I would have",

"I'd'nt":"I would not",

"I'd'nt've":"I would not have",

"I'll":"I will",

"I'm":"I am",

"I've":"I have",

"isn't":"is not",

"it'd":"it would",

"it'll":"it will",

"it's":"it is",

"let's":"let us",

"ma'am":"madam",

"mayn't":"may not",

"may've":"may have",

"mightn't":"might not",

"might've":"might have",

"mustn't":"must not",

"mustn't've":"must not have",

"must've":"must have",

"needn't":"need not",

"needn't've":"need not have",

"o'clock":"of the clock",

"oughtn't":"ought not",

"oughtn't've":"ought not have",

"shan't":"shall not",

"she'd":"she would",

"she'll":"she will",

"she's":"she is",

"should've":"should have",

"shouldn't":"should not",

"shouldn't've":"should not have",

"somebody's":"somebody is",

"someone's":"someone is",

"something's":"something is",

"so're":"so are",

"soâ€ ™s":"so is",

"soâ€ ™ve":"so have",

"that'll":"that will",

"that're":"that are",

"that's":"that is",

"that'd":"that would",

"there'd":"there would",

"there'll":"there will",

"there're":"there are",

"there's":"there is",

"these're":"these are",

"these've":"these have",

"they'd":"they would",

"they'll":"they will",

"they're":"they are",

"they've":"they have",

"this's":"this is",

"those're":"those are",

"those've":"those have",

"to've":"to have",

"wasn't":"was not",

"we'd":"we would",

"we'd've":"we would have",

"we'll":"we will",

"we're":"we are",

"we've":"we have",

"weren't":"were not",

"what'd":"what did",

"what'll":"what will",

"what're":"what are",

"what's":"what is",

"what've":"what have",

"when's":"when is",

"where'd":"where did",

"where'll":"where will",

"where're":"where are",

"where's":"where is",

"where've":"where have",

"which'd":"which would",

"which'll":"which will",

"which're":"which are",

"which's":"which is",

"which've":"which have",

"who'd":"who would",

"who'd've":"who would have",

"who'll":"who will",

"who're":"who are",

"who's":"who is",

"who've":"who have",

"why'd":"why did",

"why're":"why are",

"why's":"why is",

"won't":"will not",

"would've":"would have",

"wouldn't":"would not",

"wouldn't've":"would not have",

"y'at":"you at",

"yesâ€ ™m":"yes madam",

"you'd":"you would",

"you'll":"you will",

"you're":"you are",

"you've":"you have"

```
}
abbreviations = {
    "aamof" : "as a matter of fact",
    "acct" : "account",
    "adih" : "another day in hell",
    "afaic" : "as far as i am concerned",
    "afaict" : "as far as i can tell",
    "afaik" : "as far as i know",
    "afair" : "as far as i remember",
    "afk" : "away from keyboard",
    "app" : "application",
    "approx" : "approximately",
    "apps" : "applications",
    "asap" : "as soon as possible",
    "asl" : "age, sex, location",
    "atk" : "at the keyboard",
    "ave." : "avenue",
    "aymm" : "are you my mother",
    "ayor" : "at your own risk",
    "b&b" : "bed and breakfast",
    "b+b" : "bed and breakfast",
    "b.c" : "before christ",
    "b2b" : "business to business",
    "b2c" : "business to customer",
    "b4" : "before",
    "b4n" : "bye for now",
    "b@u" : "back at you",
    "bae" : "before anyone else",
    "bak" : "back at keyboard",
    "bbbg" : "bye bye be good",
```

"bbc" : "british broadcasting corporation",

"bbias" : "be back in a second",

"bbl" : "be back later",

"bbs" : "be back soon",

"be4" : "before",

"bfn" : "bye for now",

"blvd" : "boulevard",

"bout" : "about",

"brb" : "be right back",

"bros" : "brothers",

"brt" : "be right there",

"bsaaw" : "big smile and a wink",

"btw" : "by the way",

"bwl" : "bursting with laughter",

"c/o" : "care of",

"cet" : "central european time",

"cf" : "compare",

"cia" : "central intelligence agency",

"csl" : "can not stop laughing",

"cu" : "see you",

"cul8r" : "see you later",

"cv" : "curriculum vitae",

"cwot" : "complete waste of time",

"cya" : "see you",

"cyt" : "see you tomorrow",

"dae" : "does anyone else",

"dbmib" : "do not bother me i am busy",

"diy" : "do it yourself",

"dm" : "direct message",

"dwh" : "during work hours",

"e123" : "easy as one two three",

"eet" : "eastern european time",

"eg" : "example",

"embm" : "early morning business meeting",

"encl" : "enclosed",

"encl." : "enclosed",

"etc" : "and so on",

"faq" : "frequently asked questions",

"fawc" : "for anyone who cares",

"fb" : "facebook",

"fc" : "fingers crossed",

"fig" : "figure",

"fimh" : "forever in my heart",

"ft." : "feet",

"ft" : "featuring",

"ftl" : "for the loss",

"ftw" : "for the win",

"fwiw" : "for what it is worth",

"fyi" : "for your information",

"g9" : "genius",

"gahoy" : "get a hold of yourself",

"gal" : "get a life",

"gcse" : "general certificate of secondary education",

"gfn" : "gone for now",

"gg" : "good game",

"gl" : "good luck",

"glhf" : "good luck have fun",

"gmt" : "greenwich mean time",

"gmta" : "great minds think alike",

"gn" : "good night",

"g.o.a.t" : "greatest of all time",

"goat" : "greatest of all time",

"goi" : "get over it",

"gps" : "global positioning system",

"gr8" : "great",

"gratz" : "congratulations",

"gyal" : "girl",

"h&c" : "hot and cold",

"hp" : "horsepower",

"hr" : "hour",

"hrh" : "his royal highness",

"ht" : "height",

"ibrb" : "i will be right back",

"ic" : "i see",

"icq" : "i seek you",

"icymi" : "in case you missed it",

"idc" : "i do not care",

"idgadf" : "i do not give a damn fuck",

"idgaf" : "i do not give a fuck",

"idk" : "i do not know",

"ie" : "that is",

"i.e" : "that is",

"ifyp" : "i feel your pain",

"IG" : "instagram",

"iirc" : "if i remember correctly",

"ilu" : "i love you",

"ily" : "i love you",

"imho" : "in my humble opinion",

"imo" : "in my opinion",

"imu" : "i miss you",

"iow" : "in other words",

"irl" : "in real life",

"j4f" : "just for fun",

"jic" : "just in case",

"jk" : "just kidding",

"jsyk" : "just so you know",

"l8r" : "later",

"lb" : "pound",

"lbs" : "pounds",

"ldr" : "long distance relationship",

"lmao" : "laugh my ass off",

"lol" : "laughing out loud",

"ltd" : "limited",

"ltns" : "long time no see",

"m8" : "mate",

"mfw" : "my face when",

"mph" : "miles per hour",

"mr" : "mister",

"mrw" : "my reaction when",

"ms" : "miss",

"mte" : "my thoughts exactly",

"nagi" : "not a good idea",

"nbc" : "national broadcasting company",

"nbd" : "not big deal",

"nfs" : "not for sale",

"ngl" : "not going to lie",

"nhs" : "national health service",

"nrn" : "no reply necessary",

"nsfl" : "not safe for life",

"nsfw" : "not safe for work",

"nth" : "nice to have",

"nvr" : "never",

"nyc" : "new york city",

"oc" : "original content",

"og" : "original",

"ohp" : "overhead projector",

"oic" : "oh i see",

"omdb" : "over my dead body",

"omg" : "oh my god",

"omw" : "on my way",

"p.a" : "per annum",

"p.m" : "after midday",

"pm" : "prime minister",

"poc" : "people of color",

"pov" : "point of view",

"pp" : "pages",

"ppl" : "people",

"prw" : "parents are watching",

"ps" : "postscript",

"pt" : "point",

"ptb" : "please text back",

"pto" : "please turn over",

"qpsa" : "what happens",

"ratchet" : "rude",

"rbtl" : "read between the lines",

"rlrt" : "real life retweet",

"rofl" : "rolling on the floor laughing",

"roflol" : "rolling on the floor laughing out loud",

"rotflmao" : "rolling on the floor laughing my ass off",

"rt" : "retweet",

"ruok" : "are you ok",

"sfw" : "safe for work",

"sk8" : "skate",

"smh" : "shake my head",

"sq" : "square",

"srsly" : "seriously",

"ssdd" : "same stuff different day",

"tbh" : "to be honest",

"tbs" : "tablespooful",

"tbsp" : "tablespooful",

"tfw" : "that feeling when",

"thks" : "thank you",

"tho" : "though",

"thx" : "thank you",

"tia" : "thanks in advance",

"til" : "today i learned",

"tl;dr" : "too long i did not read",

"tldr" : "too long i did not read",

"tmb" : "tweet me back",

"tntl" : "trying not to laugh",

"ttyl" : "talk to you later",

"u" : "you",

"u2" : "you too",

"u4e" : "yours for ever",

"utc" : "coordinated universal time",

"w/" : "with",

"w/o" : "without",

"w8" : "wait",

"wassup" : "what is up",

"wb" : "welcome back",

```
    "wtf" : "what the fuck",
    "wtg" : "way to go",
    "wtpa" : "where the party at",
    "wuf" : "where are you from",
    "wuzup" : "what is up",
    "wywh" : "wish you were here",
    "yd" : "yard",
    "ygtr" : "you got that right",
    "ynk" : "you never know",
    "zzz" : "sleeping bored and tired"
}
def expand_cotractions(x):
  if type(x) is str:
      x = x.replace('\\',")
      for key in contractions:
        value = contractions[key]
        x = x.replace(key, value)
      return x
  else:
    return x
def expand_abbreviations(sentence):
    final_words=[]
    words=sentence.split()
    final_words= [abbreviations[w.lower()] if w.lower() in abbreviations.keys() else w for w in
words]
    return " ".join(final_words)
def clean_text(text):
    text=text.lower()
    text = re.sub('\[.*?\]', ", text)
    text = re.sub('[%s]' % re.escape(string.punctuation),",text)
```

```
    text = re.sub('\w*\d\w*', '', text)
    text = re.sub(r"[-()\"#/@;:<>~|.?,]","",text)
    text = re.sub('\n', '', text)
    text = re.sub(r'[^\x00-\x7F]+', '', text)
    tokens = word_tokenize(text)
    stop_words = set(stopwords.words("english"))
    words = [w for w in tokens if not w in stop_words]
    return " ".join(words)
cleaned1 = lambda x: clean_text(x)
df['cleaned Message']=pd.DataFrame(df['Message'].apply(cleaned1))
df['cleaned Message'].unique()
#**Data Visualization**
text=' '.join([word for word in df['Message']])
plt.figure(figsize=(20,15),facecolor='None')
wordcloud=WordCloud(max_words=500,width=1600,height=800).generate(text)
plt.imshow(wordcloud,interpolation='bilinear')
plt.axis("off")
plt.title('Most frequent words in positive reviews', fontsize=25)
plt.show()
sns.histplot(data=df,x='Category',color='r')
#**Data Splitting**
x=df['Message']
y=df['Category']
xtrain,xtest,ytrain,ytest = train_test_split(x,y,random_state=55,test_size=0.1)
print(xtrain.shape)
print(ytrain.shape)
print(xtest.shape)
print(ytest.shape)
y.unique()
```

```
#Model Building
#1.Logistic Regression
tvec=CountVectorizer()
clf=LogisticRegression(tol=0.01,max_iter=80,n_jobs=1,verbose=20,C=0.002)
from sklearn.pipeline import Pipeline
model1=Pipeline([('Vectorizer',tvec),('Classifier',clf)])
model1.fit(xtrain,ytrain)
p=model1.predict(xtrain)
p
a=accuracy_score(ytrain,p)*100
print('training accuracy:',round(a,2),"%")
p1=model1.predict(xtest)
p1
a1=accuracy_score(ytest,p1)*100
print('testing accuracy:',round(a1,2),"%")
from sklearn.metrics import
confusion_matrix,classification_report,precision_score,recall_score,f1_score,roc_curve,roc_a
uc_score
confusion_matrix(ytest,p1)
sns.heatmap(confusion_matrix(ytest,p1),annot=True)
#2.Random Forest Classifier
clf1 = RandomForestClassifier(criterion='entropy')
model2 = Pipeline([('Vectorizer',tvec),('Classifier',clf1)])
model2.fit(xtrain,ytrain)
p=model2.predict(xtrain)
p
a=accuracy_score(ytrain,p)*100
print('training accuracy:',round(a,2),"%")
p1=model2.predict(xtest)
p1
a1=accuracy_score(ytest,p1)*100
```

```
print('testing accuracy:',round(a1,2),"%")

from sklearn.metrics import
confusion_matrix,classification_report,precision_score,recall_score,f1_score,roc_curve,roc_a
uc_score

confusion_matrix(ytest,p1)

sns.heatmap(confusion_matrix(ytest,p1),annot=True)

# 3.SVM-SVC

clf2 = SVC(probability=True)

model3 = Pipeline([('Vectorizer',tvec),('Classifier',clf2),])

model3.fit(xtrain,ytrain)

p=model3.predict(xtrain)

p

a=accuracy_score(ytrain,p)*100

print('training accuracy:',round(a,2),"%")

p1=model3.predict(xtest)

p1

a1=accuracy_score(ytest,p1)*100

print('testing accuracy:',round(a1,2),"%")

from sklearn.metrics import
confusion_matrix,classification_report,precision_score,recall_score,f1_score,roc_curve,roc_a
uc_score

confusion_matrix(ytest,p1)

sns.heatmap(confusion_matrix(ytest,p1),annot=True)

#**Predict_Proba**

pred1=model1.predict_proba(xtest)

pred1

pred2=model2.predict_proba(xtest)

pred1

pred3=model3.predict_proba(xtest)

pred1
```

```
#roc_curve
fpr,tpr,thresh=roc_curve(ytest,pred1[:,1],pos_label=1)
fpr1,tpr1,thresh1=roc_curve(ytest,pred2[:,1],pos_label=1)
fpr2,tpr2,thresh2=roc_curve(ytest,pred3[:,1],pos_label=1)
plt.plot(fpr,tpr,'r--',label='logistic regression')
plt.plot(fpr1,tpr1,'g--',label='Random Forest Classifier')
plt.plot(fpr2,tpr2,'b--',label='SVC')
plt.xlabel('fpr rate')
plt.ylabel('tpr rate')
plt.title('roc_curve')
plt.show()
#**Roc_Auc_Score**
roc_auc_score(ytest,pred1[:,1])
roc_auc_score(ytest,pred2[:,1])
roc_auc_score(ytest,pred3[:,1])
#**Final Prediction Using the best Model**
example=['Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&Cs apply 08452810075over18s']
result = model2.predict(example)
print(result)
if result[0]==0:
  Print('This is a Ham Mail')
else:
  print('This is a Spam Mail')
```