EMAIL SPAM CLASSIFIER

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

MOHAMMAD AZHAR SOFI
[RA2011027010035]
K.VEERENDRANADH
[RA2011027010063]
SAPRATIBH SHYAM
[RA2011027010008]

Under the guidance of

Dr. M. MERCY THERESA

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"EMAIL SPAM CLASSIFIER"** is the bona fide work of **MOHAMMAD AZHAR SOFI [RA2011027010035], K.VEERENDRANADH [RA2011027010063]** and **SAPRATIBH SHYAM [RA2011027010008]** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. M.
MERCY
THERESA
Professor
DSBS

SIGNATURE

Dr. M Lakshmi

HEAD OF THE DEPARTMENT

Professor &
Head DSBS

ABSTRACT

As the usage of mobile phones increased, the use of Short Message Service increased significantly. Due to the lower costs of text messages, people started using it for promotional purposes and unethical activities. This resulted in the ratio of spam messages increasing exponentially and thereby loss of personal and financial data. To prevent data loss, it is crucial to detect spam messages as quick as possible. Thus, the research aims to classify spam messages not only efficiently but also with low latency. Different machine learning models like XGBoost, LightGBM, Bernoulli Naive Bayes that are proven to be very fast with low time complexity have been implemented in the research. The length of the messages was taken as an additional feature, and the features were extracted using Unigram, Bigram and TF-IDF matrix. Chi-Square feature selection was implemented to further reduce the space complexity. The results showcased that Bernoulli Naive Bayes followed by Light GBM with the TF-IDF matrix generated the highest accuracy of 96.5% in 0.157 seconds and 95.4% in 1.708 seconds respectively.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
LR	Logistic Regression
EDA	Exploratory Data Analysis

CHAPTER 1
INTRODUCTION

Short Messaging Service (SMS) is mainly used for unofficial communication such as promoting new products and services but at times also used for official communication like information about any bank transaction or confirmation of the order on an online portal etc.

Due to advancements in technology, the costs of sending an SMS have reduced drastically. This has proved to be a boon for some whereas a bane for many. People are misusing the SMS facility to promote products, services, offers and schemes and so on.

How annoying this has become can be assessed by the fact that people have started ignoring SMS they receive because twenty to thirty percent of the total SMS received is spam. This menace is growing at a rapid rate. As a result, people miss out on genuine informative messages such as bank transactions.

At times the ignorance towards SMS can prove detrimental because some fraud transactions might have been performed, but the information was neglected.

The motive behind this project is to apply machine learning algorithms to separate spam messages from genuine ones. Machine learning techniques along with Natural Language Processing techniques was used to make the process more agile and efficient.

CHAPTER 2

LITERATURE SURVEY

An analysis of spam SMS filtering was done on the UCI machine learning dataset in the year 2015 by (Kim et al. (2015)) who chose the frequency ratio feature selection technique while implementing the algorithms like Naive Bayes, Logistic Regression and J-48 Decision Trees where the 10 fold cross-validation technique was applied. It was seen that Naive Bayes generated results in a minimum time with the highest accuracy of 94 percent. In the year 2018, a similar analysis was conducted by (Gupta et al. (2018)) using 2 sets of data, one with UCI machine learning which is the same corpus as of Kaggle with a total of 5574 ham and spam messages and another dataset contains 2000 spam and ham messages.

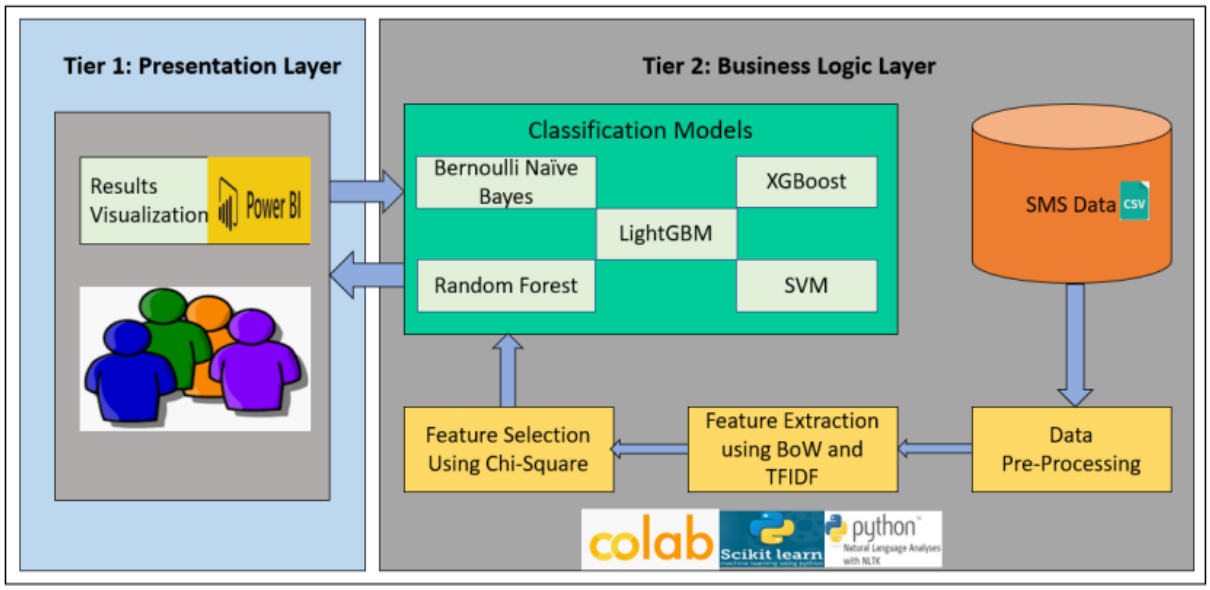
TF-IDF matrix was created and then machine learning algorithms like Naive Bayes, Random Forest, SVM, Decision Tree, Convolutional Neural Network and Artificial Neural Network were applied on both the datasets. The results obtained by CNN were the state-of-art in this area with an accuracy of 99.10 followed by Naive Bayes and SVM. A research conducted by (Ma et al. (2016)), on spam SMS detection proposes a message topic model which is a form of probability topic model. It uses the KNN algorithm to remove the sparsity problem in the messages. The symbol terms, background terms were considered and it was found that the model generated better results than the standard LDA. The classifier Gentle Boost was used for the first time in the research done by (Akbari and Sajedi (2015)) on SMS spam detection in the year 2015. For unbalanced data and binary classification, boosting algorithms work well.

Gentle Boost is a combination of two algorithms, namely AdaBoost and Logit Boost. Gentle Boost is well known for its higher accuracy and less consumption of storage as it removes unwanted features. It obtained an accuracy of 98% on the dataset consisting of 5572 text messages. The author (Agarwal et al. (2016)) states that the short length of the messages and the use of casual words in the text messages do not allow it to perform well with the already established solutions of email spam filtering. In this research, it can be seen that SVM followed by Multinomial Naive Bayes (MNB) shows outstanding results in terms of accuracy with 98.23 and 97.87% respectively. MNB took the least execution time of 2.03 seconds. The researcher further suggests that features like the number of characters in the messages or definite threshold to the length of the message can increase the performance.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1. Architecture diagram of Email Spam Classifier



3.2. Description of Module and Component

Naive Bayes Classifier: This algorithm uses Bayes' theorem to classify messages as spam or not spam. It works by calculating the probability that a message is spam or not spam based on the occurrence of certain words in the message. It is a simple and fast algorithm that can work well with large datasets.

Support Vector Machines (SVM): This algorithm tries to find a hyperplane that separates spam messages from non-spam messages in a high-dimensional space. It is effective in handling both linear and non-linear data and can work well with small to medium-sized datasets.

Decision Trees: This algorithm builds a tree-like model of decisions based on features of the messages to classify them as spam or not spam. It is easy to understand and can handle both numerical and categorical data.

Random Forest: This algorithm is an ensemble learning method that combines multiple decision trees to improve the classification accuracy. It can handle large datasets and can work well with noisy data.

Logistic Regression: This is a binary classification algorithm that models the probability of a message being spam or not as a function of the message features. It works by finding the best decision boundary that separates the two classes. Logistic regression is fast, interpretable, and works well with linearly separable data.

CHAPTER 4

METHODOLOGY

1. Data Selection Phase : Data Selection phase where the UCI machine learning repository provided data to Kaggle, and the data was downloaded from Kaggle which is in .csv format.

2. **Data cleaning** : Data cleaning is an essential step in email spam classification using machine learning. The main goal of data cleaning is to prepare the data in a format that can be easily processed by machine learning algorithms. Irrelevant information is removed from the email text. Here we removed empty cells and duplicate data. With the help of Pandas library we were able to delete column that are not relevant, or contains wrong values, like empty or NULL values.

3. **Exploratory data analysis** : Exploratory Data Analysis is a crucial process where the data is analyzed to uncover underlying patterns, spot abnormality and test the hypothesis . It is the best practice to understand the data and then carry out the data mining process. Missing value analysis was carried out using libraries of python like Pandas. For visualization of most frequent words appearing in spam messages and ham messages, the Matplotlib library with Word Cloud technique was used.

4. **Data Pre-processing** : Data pre-processing is mainly cleaning of data by removing unwanted rows, columns , missing values, outliers, etc. For the research, the following pre-processing steps have been taken:

5. Removal of Unwanted Columns It was found that there were 3 extra columns without data in it, which adds extra noise to the model hence removed .

6. Cleaning of Text Messages For cleaning the text messages, the following steps are involved:

7. **Model building**: In this step, the data is prepared for model building. This involves splitting the data into training and testing sets, and converting the text data into numerical form that can be processed by the machine learning algorithm. The data cleaning steps described earlier, such as removing irrelevant information, tokenization, and stop word removal, are also performed.

8. Feature Extraction is done in which features are extracted from the email text. This involves identifying the most important words or phrases that are likely to

indicate whether an email is spam or not. Feature extraction techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) and bag of words are commonly used for this purpose .

9. Model Selection is also done in which the appropriate machine learning algorithm is selected for the project. Commonly used algorithms for email spam classification include Naive Bayes, Support Vector Machines (SVM), Decision Trees, and Random Forests, the selected machine learning algorithm is trained on the prepared data. The algorithm learns to classify emails as spam or not spam based on the extracted features.

10. **Model Evaluation** : In this step, the trained model is evaluated using the testing set. The accuracy, precision, recall, and F1 score of the model are calculated to determine its performance .

11. **Model Optimization**: In this step, the model is optimized by adjusting the model parameters or feature selection techniques to improve its performance. This step involves iterative testing and refining of the model until the desired performance is achieved.

12. Tokenize the words: In this step, the words are split into tokens based on white spaces or punctuation. To achieve it, word tokenize function from NLTK library was used.

13. Removal of stop words from text messages: Stop words are basically the most commonly used words (such as “a”, “an”, ”the”) which increases the dimensionality and impacts the efficiency of the model

14. Lemmatizing words: Lemmatization in simple terms refers to the removal of duplicate data. For example, words like “study”, “studying”, and “studies” are considered 3different words after the creation of a Document Term Matrix and hence increases the dimensionality.

CHAPTER 5

CODING AND TESTING

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv('spam.csv')
```

In [4]:
```
df.sample(5)
```

Out[4]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
2464	ham	They will pick up and drop in car.so no problem..	NaN	NaN	NaN
1248	ham	HI HUN! IM NOT COMIN 2NITE-TELL EVERY1 IM SORR...	NaN	NaN	NaN
1413	spam	Dear U've been invited to XCHAT. This is our f...	NaN	NaN	NaN
2995	ham	They released vday shirts and when u put it on...	NaN	NaN	NaN
4458	spam	Welcome to UK-mobile-date this msg is FREE giv...	NaN	NaN	NaN

In [5]:
```
df.shape
```

Out[5]:
```
(5572, 5)
```

In [ ]:
```
# 1. Data cleaning
# 2. EDA
# 3. Text Preprocessing
# 4. Model building
# 5. Evaluation
# 6. Improvement
# 7. Website
# 8. Deploy
```

1. Data Cleaning

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

In [7]:
```
# drop last 3 cols
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
```

In [8]:
```
df.sample(5)
```

Out[8]:

	v1	v2
1947	ham	The battery is for mr adewale my uncle. Aka Egbon
2712	ham	Hey you still want to go for yogasana? Coz if ...
4428	ham	Hey they r not watching movie tonight so i'll ...
3944	ham	I will be gentle princess! We will make sweet ...
49	ham	U don't know how stubborn I am. I didn't even ...

In [9]:
```
# renaming the cols
df.rename(columns={'v1':'target','v2':'text'},inplace=True)
df.sample(5)
```

Out[9]:

	target	text
1418	ham	Lmao. Take a pic and send it to me.
2338	ham	Alright, see you in a bit
88	ham	I'm really not up to it still tonight babe
3735	ham	Hows the street where the end of library walk is?
3859	ham	Yep. I do like the pink furniture tho.

In [10]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```

In [12]:
```
df['target'] = encoder.fit_transform(df['target'])
```

In [13]:
```
df.head()
```

Out[13]:

	target	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

In [14]:
```
# missing values
df.isnull().sum()
```

Out[14]:
```
target    0
text      0
dtype: int64
```

In [15]:
```
# check for duplicate values
df.duplicated().sum()
```

Out[15]:
```
403
```

In [17]:
```
# remove duplicates
df = df.drop_duplicates(keep='first')
```

In [18]:
```
df.duplicated().sum()
```

Out[18]:
```
0
```

In [19]:
```
df.shape
```

Out[19]:
```
(5169, 2)
```

2.EDA

In [29]:
```
df.head()
```

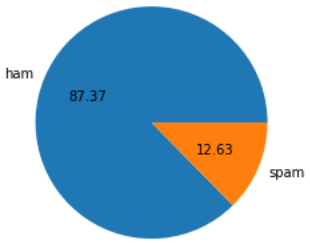
Out[29]:

	target	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

In [31]:
```
df['target'].value_counts()
```

Out[31]:
```
0    4516
1     653
Name: target, dtype: int64
```

In [33]:
```
import matplotlib.pyplot as plt
plt.pie(df['target'].value_counts(), labels=['ham','spam'],autopct="%0.2f")
plt.show()
```



In [34]:
```
# Data is imbalanced
```

In [35]:
```
import nltk
```

In [ ]:
```
!pip install nltk
```

In [37]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\91842\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Out[37]:
```
True
```

In [45]:
```
df['num_characters'] = df['text'].apply(len)
```

In [46]:
```
df.head()
```

Out[46]:

	target	text	num_characters
0	0	Go until jurong point, crazy.. Available only ...	111
1	0	Ok lar... Joking wif u oni...	29
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	0	U dun say so early hor... U c already then say...	49
4	0	Nah I don't think he goes to usf, he lives aro...	61

In [50]:
```
# num of words
df['num_words'] = df['text'].apply(lambda x:len(nltk.word_tokenize(x)))
```

In [51]:
```
df.head()
```

Out[51]:

	target	text	num_characters	num_words
0	0	Go until jurong point, crazy.. Available only ...	111	24
1	0	Ok lar... Joking wif u oni...	29	8
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37
3	0	U dun say so early hor... U c already then say...	49	13
4	0	Nah I don't think he goes to usf, he lives aro...	61	15

In [53]:
```
df['num_sentences'] = df['text'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

In [54]:
```
df.head()
```

Out[54]:

	target	text	num_characters	num_words	num_sentences
0	0	Go until jurong point, crazy.. Available only ...	111	24	2
1	0	Ok lar... Joking wif u oni...	29	8	2
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2
3	0	U dun say so early hor... U c already then say...	49	13	1
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1

In [55]:
```
df[['num_characters','num_words','num_sentences']].describe()
```

Out[55]:

	num_characters	num_words	num_sentences
count	5169.000000	5169.000000	5169.000000
mean	78.923776	18.456375	1.962275
std	58.174846	13.323322	1.433892
min	2.000000	1.000000	1.000000
25%	36.000000	9.000000	1.000000
50%	60.000000	15.000000	1.000000
75%	117.000000	26.000000	2.000000
max	910.000000	220.000000	38.000000

In [58]:
```
# ham
df[df['target'] == 0][['num_characters','num_words','num_sentences']].describe()
```

Out[58]:

	num_characters	num_words	num_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.456820	17.123339	1.815545
std	56.356802	13.491315	1.364098
min	2.000000	1.000000	1.000000
25%	34.000000	8.000000	1.000000
50%	52.000000	13.000000	1.000000
75%	90.000000	22.000000	2.000000
max	910.000000	220.000000	38.000000

In [59]:
```
#spam
df[df['target'] == 1][['num_characters','num_words','num_sentences']].describe()
```

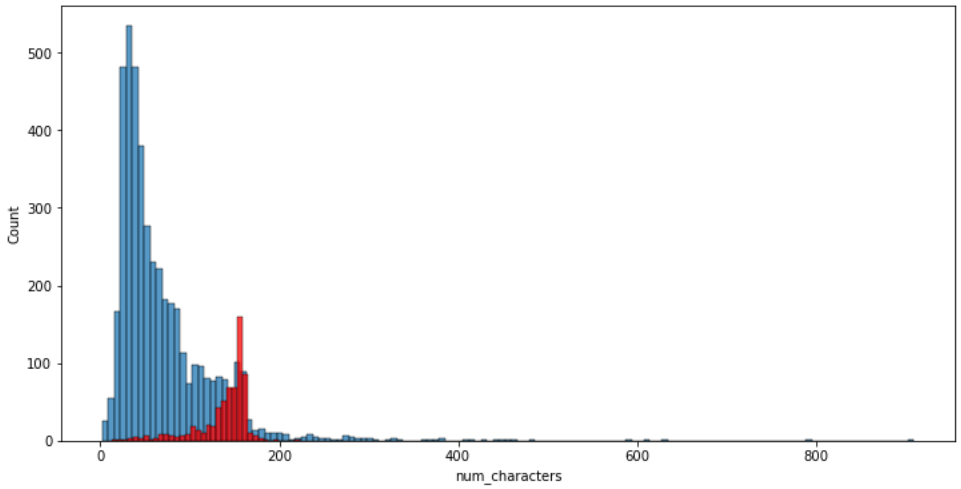
Out[59]:

	num_characters	num_words	num_sentences
count	653.000000	653.000000	653.000000
mean	137.479326	27.675345	2.977029
std	30.014336	7.011513	1.493676
min	13.000000	2.000000	1.000000
25%	131.000000	25.000000	2.000000
50%	148.000000	29.000000	3.000000
75%	157.000000	32.000000	4.000000
max	223.000000	46.000000	9.000000

In [78]:
```
import seaborn as sns
```

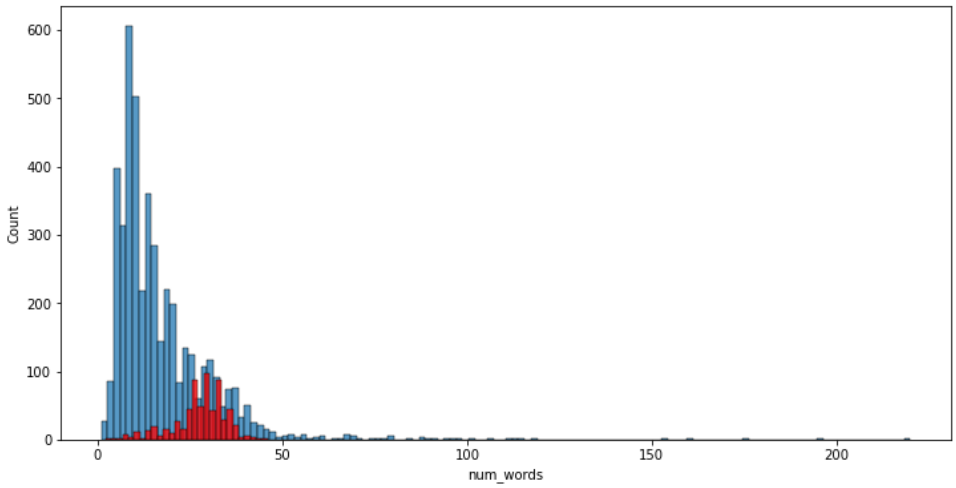
In [84]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
```

Out[84]: <AxesSubplot:xlabel='num_characters', ylabel='Count'>



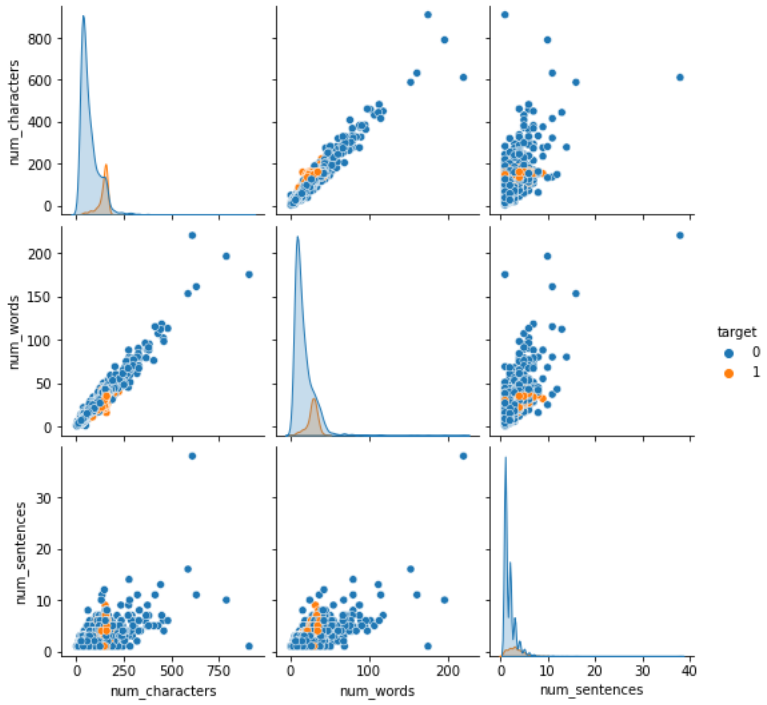
In [85]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
```

Out[85]: <AxesSubplot:xlabel='num_words', ylabel='Count'>



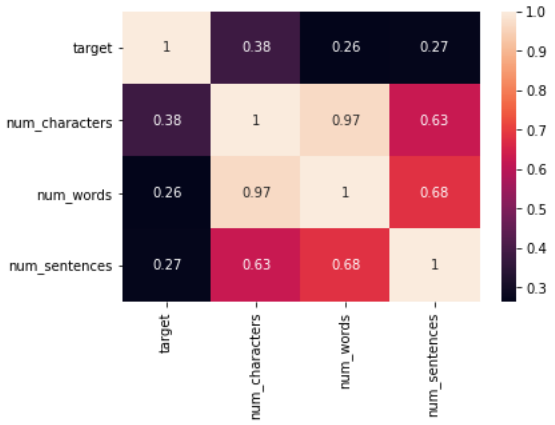
In [86]:
```
sns.pairplot(df,hue='target')
```

Out[86]: <seaborn.axisgrid.PairGrid at 0x16f88c4a4f0>



In [89]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[89]: <AxesSubplot:>



3. Data Preprocessing

- Lower case
- Tokenization
- Removing special characters
- Removing stop words and punctuation
- Stemming

In [187…
```
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))


    return " ".join(y)
```

In [192…
```
transform_text("I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today
```

Out[192…
```
'gon na home soon want talk stuff anymor tonight k cri enough today'
```

In [191…
```
df['text'][10]
```

Out[191…
```
"I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today."
```

In [186…
```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
ps.stem('loving')
```

Out[186…
```
'love'
```

In [194…
```
df['transformed_text'] = df['text'].apply(transform_text)
```

In [195…
```
df.head()
```

Out[195…

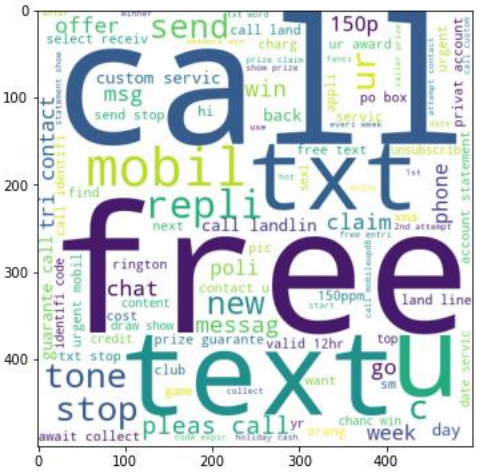
	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

In [232…
```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
```

In [233…
```
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep=" "))
```

In [236…]
```
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

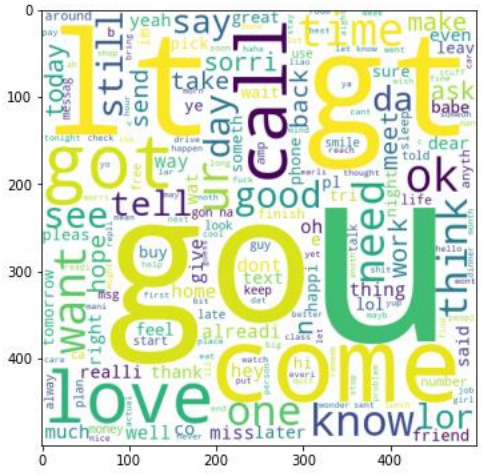
Out[236…] <matplotlib.image.AxesImage at 0x16f87ea8cd0>



In [237…]
```
ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(sep=" "))
```

In [238…]
```
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
```

Out[238…] <matplotlib.image.AxesImage at 0x16f87f6c280>



In [267…]
```
df.head()
```

Out[267…]

	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

In [272…]
```
spam_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

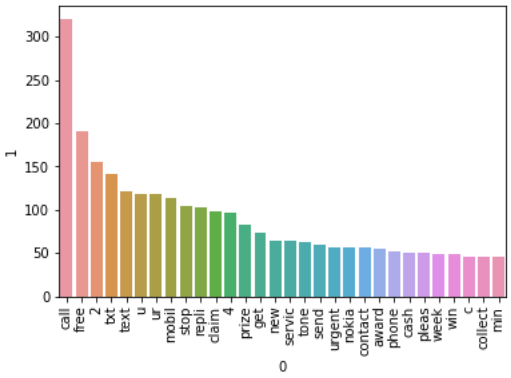
In [274…]
```
len(spam_corpus)
```

Out[274…] 9941

In [280…

```
from collections import Counter
sns.barplot(pd.DataFrame(Counter(spam_corpus).most_common(30))[0],pd.DataFrame(Counter(spam_corpus).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\91842\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword a
rgs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [281…

```
ham_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
```

In [282…

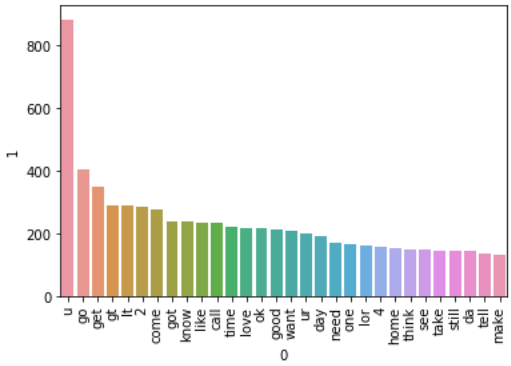
```
len(ham_corpus)
```

Out[282… 35303

In [284…

```
from collections import Counter
sns.barplot(pd.DataFrame(Counter(ham_corpus).most_common(30))[0],pd.DataFrame(Counter(ham_corpus).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\91842\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword a
rgs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [285…

```
# Text Vectorization
# using Bag of Words
df.head()
```

Out[285…

	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

4. Model Building

In [522…
```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

In [523…
```
X = tfidf.fit_transform(df['transformed_text']).toarray()
```

In [470…
```
#from sklearn.preprocessing import MinMaxScaler
#scaler = MinMaxScaler()
#X = scaler.fit_transform(X)
```

In [483…
```
# appending the num_character col to X
#X = np.hstack((X,df['num_characters'].values.reshape(-1,1)))
```

In [524…
```
X.shape
```

Out[524… (5169, 3000)

In [525…
```
y = df['target'].values
```

In [526…
```
from sklearn.model_selection import train_test_split
```

In [527…
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [528…
```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

In [489…
```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [490…
```
gnb.fit(X_train,y_train)
y_pred1 = gnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
```

```
0.8916827852998066
[[808  88]
 [ 24 114]]
0.5643564356435643
```

In [529…
```
mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print(accuracy_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
print(precision_score(y_test,y_pred2))
```

```
0.971953578336557
[[896   0]
 [ 29 109]]
1.0
```

In [492…
```
bnb.fit(X_train,y_train)
y_pred3 = bnb.predict(X_test)
print(accuracy_score(y_test,y_pred3))
print(confusion_matrix(y_test,y_pred3))
print(precision_score(y_test,y_pred3))
```

```
0.9835589941972921
[[895   1]
 [ 16 122]]
0.991869918699187
```

In [493…
```
# tfidf --> MNB
```

In [494…

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
```

In [495…

```
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50,random_state=2)
xgb = XGBClassifier(n_estimators=50,random_state=2)
```

In [496…

```
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}
```

In [497…

```
def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)

    return accuracy,precision
```

In [348…

```
train_classifier(svc,X_train,y_train,X_test,y_test)
```

Out[348…

```
(0.9729206963249516, 0.9741379310344828)
```

In [498…

```
accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf, X_train,y_train,X_test,y_test)

    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)

    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
```

```
C:\Users\91842\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: Precision is ill-d
efined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
For  SVC
Accuracy -  0.8665377176015474
Precision -  0.0
For  KN
Accuracy -  0.9284332688588007
Precision -  0.7711864406779662
For  NB
Accuracy -  0.9400386847195358
Precision -  1.0
For  DT
Accuracy -  0.9439071566731141
Precision -  0.8773584905660378
For  LR
Accuracy -  0.9613152804642167
Precision -  0.9711538461538461
For  RF
Accuracy -  0.9748549323017408
Precision -  0.9827586206896551
For  AdaBoost
Accuracy -  0.971953578336557
Precision -  0.9504132231404959
For  BgC
Accuracy -  0.9680851063829787
Precision -  0.9133858267716536
For  ETC
Accuracy -  0.97678916827853
Precision -  0.975
For  GBDT
Accuracy -  0.9487427466150871
Precision -  0.9292929292929293
C:\Users\91842\anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The use of label encoder in XGBClassifier is
deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder
=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ...,
[num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[14:16:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
For  xgb
Accuracy -  0.9700193423597679
Precision -  0.9421487603305785
```

In [386…
```
performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores,'Precision':precision_scores}).sort_values
```

In [387…
```
performance_df
```

Out[387…

	Algorithm	Accuracy	Precision
1	KN	0.900387	1.000000
2	NB	0.959381	1.000000
8	ETC	0.977756	0.991453
5	RF	0.970019	0.990826
0	SVC	0.972921	0.974138
6	AdaBoost	0.962282	0.954128
10	xgb	0.971954	0.950413
4	LR	0.951644	0.940000
9	GBDT	0.951644	0.931373
7	BgC	0.957447	0.861538
3	DT	0.935203	0.838095

In [364…
```
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
```

In [365…

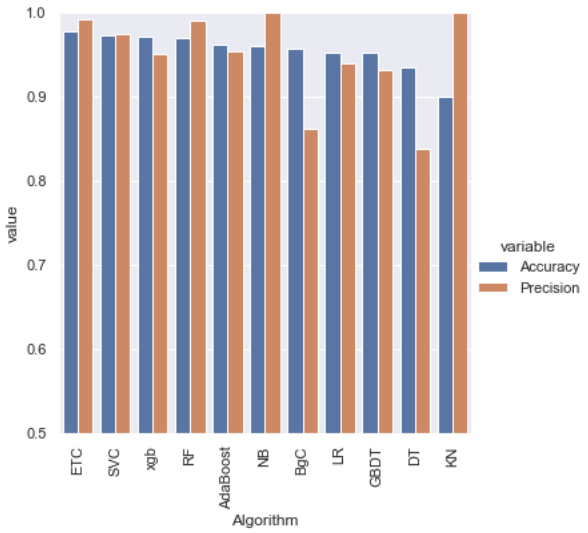
```
performance_df1
```

Out[365…

	Algorithm	variable	value
0	ETC	Accuracy	0.977756
1	SVC	Accuracy	0.972921
2	xgb	Accuracy	0.971954
3	RF	Accuracy	0.970019
4	AdaBoost	Accuracy	0.962282
5	NB	Accuracy	0.959381
6	BgC	Accuracy	0.957447
7	LR	Accuracy	0.951644
8	GBDT	Accuracy	0.951644
9	DT	Accuracy	0.935203
10	KN	Accuracy	0.900387
11	ETC	Precision	0.991453
12	SVC	Precision	0.974138
13	xgb	Precision	0.950413
14	RF	Precision	0.990826
15	AdaBoost	Precision	0.954128
16	NB	Precision	1.000000
17	BgC	Precision	0.861538
18	LR	Precision	0.940000
19	GBDT	Precision	0.931373
20	DT	Precision	0.838095
21	KN	Precision	1.000000

In [385…

```
sns.catplot(x = 'Algorithm', y='value',
               hue = 'variable',data=performance_df1, kind='bar',height=5)
plt.ylim(0.5,1.0)
plt.xticks(rotation='vertical')
plt.show()
```



In [ ]:

```
# model improve
# 1. Change the max_features parameter of TfIdf
```

In [428…

```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_ft_3000':accuracy_scores,'Precision_max_ft_3000':precision_sc
```

In [454…

```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_scaling':accuracy_scores,'Precision_scaling':precision_scores}).s
```

In [452…
```
new_df = performance_df.merge(temp_df,on='Algorithm')
```

In [456…
```
new_df_scaled = new_df.merge(temp_df,on='Algorithm')
```

In [499…
```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_num_chars':accuracy_scores,'Precision_num_chars':precision_scores}
```

In [501…
```
new_df_scaled.merge(temp_df,on='Algorithm')
```

Out[501…

	Algorithm	Accuracy	Precision	Accuracy_max_ft_3000	Precision_max_ft_3000	Accuracy_scaling	Precision_scaling	Accuracy_num_cha
0	KN	0.900387	1.000000	0.905222	1.000000	0.905222	0.976190	0.92843
1	NB	0.959381	1.000000	0.971954	1.000000	0.978723	0.946154	0.94003
2	ETC	0.977756	0.991453	0.979691	0.975610	0.979691	0.975610	0.97678
3	RF	0.970019	0.990826	0.975822	0.982906	0.975822	0.982906	0.97485
4	SVC	0.972921	0.974138	0.974855	0.974576	0.971954	0.943089	0.86653
5	AdaBoost	0.962282	0.954128	0.961315	0.945455	0.961315	0.945455	0.97195
6	xgb	0.971954	0.950413	0.968085	0.933884	0.968085	0.933884	0.97001
7	LR	0.951644	0.940000	0.956480	0.969697	0.967118	0.964286	0.96131
8	GBDT	0.951644	0.931373	0.946809	0.927835	0.946809	0.927835	0.94874
9	BgC	0.957447	0.861538	0.959381	0.869231	0.959381	0.869231	0.96808
10	DT	0.935203	0.838095	0.931335	0.831683	0.932302	0.840000	0.94390

In [514…
```
# Voting Classifier
svc = SVC(kernel='sigmoid', gamma=1.0,probability=True)
mnb = MultinomialNB()
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)

from sklearn.ensemble import VotingClassifier
```

In [515…
```
voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc)],voting='soft')
```

In [516…
```
voting.fit(X_train,y_train)
```

Out[516…
```
VotingClassifier(estimators=[('svm',
                              SVC(gamma=1.0, kernel='sigmoid',
                                  probability=True)),
                             ('nb', MultinomialNB()),
                             ('et',
                              ExtraTreesClassifier(n_estimators=50,
                                                   random_state=2))],
                 voting='soft')
```

In [517…
```
y_pred = voting.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```
Accuracy 0.9816247582205029
Precision 0.9917355371900827
```

In [518…
```
# Applying stacking
estimators=[('svm', svc), ('nb', mnb), ('et', etc)]
final_estimator=RandomForestClassifier()
```

In [519…
```
from sklearn.ensemble import StackingClassifier
```

In [520…
```
clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)
```

In [521…

```
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```
Accuracy 0.9787234042553191
Precision 0.9328358208955224
```

In [530…

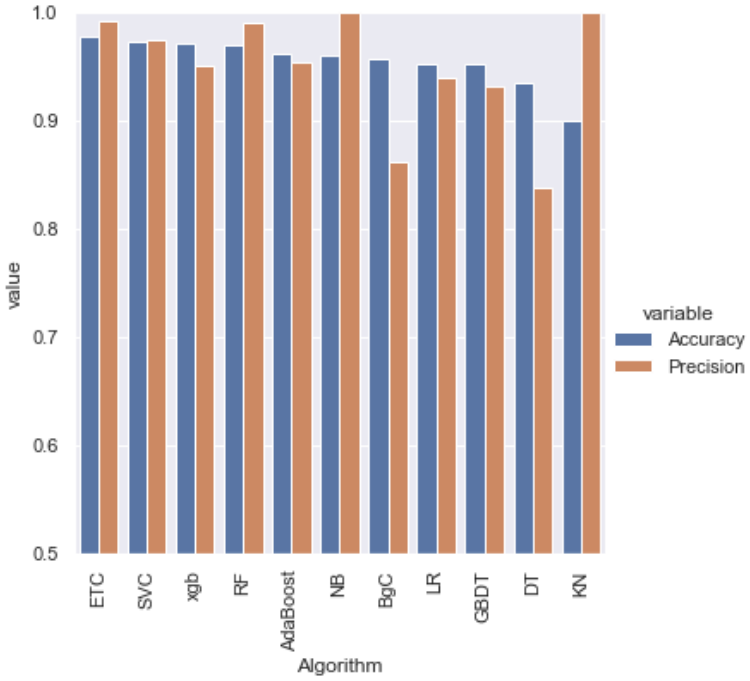
```
import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(mnb,open('model.pkl','wb'))
```

CHAPTER 6

SCREENSHOTS AND RESULTS

```
C:\Users\91842\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: Precision is ill-d
efined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
For  SVC
Accuracy -  0.8665377176015474
Precision -  0.0
For  KN
Accuracy -  0.9284332688588007
Precision -  0.7711864406779662
For  NB
Accuracy -  0.9400386847195358
Precision -  1.0
For  DT
Accuracy -  0.9439071566731141
Precision -  0.8773584905660378
For  LR
Accuracy -  0.9613152804642167
Precision -  0.9711538461538461
For  RF
Accuracy -  0.9748549323017408
Precision -  0.9827586206896551
For  AdaBoost
Accuracy -  0.971953578336557
Precision -  0.9504132231404959
For  BgC
Accuracy -  0.9680851063829787
Precision -  0.9133858267716536
For  ETC
Accuracy -  0.97678916827853
Precision -  0.975
For  GBDT
Accuracy -  0.9487427466150871
Precision -  0.9292929292929293
C:\Users\91842\anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The use of label encoder in XGBClassifier is
deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder
=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ...,
[num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[14:16:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
For  xgb
Accuracy -  0.9700193423597679
Precision -  0.9421487603305785
```

	Algorithm	variable	value
0	ETC	Accuracy	0.977756
1	SVC	Accuracy	0.972921
2	xgb	Accuracy	0.971954
3	RF	Accuracy	0.970019
4	AdaBoost	Accuracy	0.962282
5	NB	Accuracy	0.959381
6	BgC	Accuracy	0.957447
7	LR	Accuracy	0.951644
8	GBDT	Accuracy	0.951644
9	DT	Accuracy	0.935203
10	KN	Accuracy	0.900387
11	ETC	Precision	0.991453
12	SVC	Precision	0.974138
13	xgb	Precision	0.950413
14	RF	Precision	0.990826
15	AdaBoost	Precision	0.954128
16	NB	Precision	1.000000
17	BgC	Precision	0.861538
18	LR	Precision	0.940000
19	GBDT	Precision	0.931373
20	DT	Precision	0.838095
21	KN	Precision	1.000000



CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

Spam SMS is a grave threat and it is getting more and more serious with each day. It can cause significant harm and the consequences can be drastic. Countering this menace with high accuracy and low latency was the main motivation behind this research. A sample dataset was used to find an effective solution to the above problem. In the initial stage, Exploratory Analysis was conducted on the dataset wherein it was established that the length feature was a contributing factor in identifying the ham and spam. This also revealed that there was a high imbalance in the ham and spam class of the dataset. This was taken care of by the down-sampling technique to match the ham and spam class counts. Data pre-processing and cleaning was done to reduce the noise from the data. Furthermore, the features were extracted using the Bag of Words and TF-IDF models. To achieve low latency, the extracted features were selected using the Chi-Square feature selection technique. Then the machine learning models Bernoulli Naive Bayes, Light GBM, and XG Boost were applied along with the traditional base models SVM and Random Forest. The research objectives were accomplished and the spam SMS were filtered with high accuracy within a short time. Hence the research can be termed successful. The results section demonstrated that the suggested models like Bernoulli Naive Bayes and Light GBM combined with TF-IDF were apt for solving the research question since they produced an accuracy of 96.5% and 95.4% respectively. Also, the time taken by these models was 0.157 and 1.708 seconds which was significantly better than the other traditional models.

REFERENCES

- Agarwal, S., Kaur, S. and Garhwal, S. (2016). SMS spam detection
- for Indian messages , proceedings on 2015 1st International Conference on Next Generation
- Computing Technologies, NGCT 2015 (September): 634–638.
- Aich, P., Venu Gopalan, M. and Gupta, D. (2019). Content based spam
- detection in short text messages with emphasis on dealing with imbalanced datasets,
- Proceedings – 2018 4th International Conference on Computing, Communication Control and Automation , ICCUBEA 2018.
- Akbari, F. and Sajedi , H. (2015). SMS spam detection using selected text
- features and Boosting Classifiers, 2015 7th Conference on Information and Knowledge Technology , IKT 2015 pp. 1–5.
- [6] M. Ayaz, M. Ammad-uddin, I. Baig and e. M. Aggoune, "Wireless Possibilities: A Review," in IEEE Sensors Journal, vol. 18, no. 1, pp. 4-30, 1 Jan.1, 2018.