BUILDING A SMARTER AI-POWERED SPAM CLASSIFIER

TEAM MEMBER
311121205003-Ajayroshan R

PHASE 2- DEVELOPMENT PART 1

Project: Building a Smarter AI-Powered Spam Classifier

Abstract:- Spam Classification using Artificial Intelligence – For business purposes, email is the most widely utilized mode of official communication. Despite the availability of other forms of communication, email usage continues to rise. In today's world, automated email management is critical since the volume of emails grows by the day. More than 55 percent of all emails have been recognized as spam. This demonstrates that spammers waste email users' time and resources while producing no meaningful results. Spammers employ sophisticated and inventive strategies to carry out their criminal actions via spam emails. As a result, it is critical to comprehend the many spam email classification tactics and mechanisms. The main focus of this paper is on spam classification using machine learning algorithms. Furthermore, this research includes a thorough examination and evaluation of research on several machine learning methodologies and email properties used in various Machine Learning approaches. Future study goals and obstacles in the subject of spam classification are also discussed, which may be valuable to future researchers.

Objective: –

Machine learning algorithms use statistical models to classify data. In the case of spam detection, a trained machine learning model must be able to determine whether the sequence of words found in an email is closer to those found in spam emails or safe ones.

Introduction: –

For the majority of internet users, email has become the most often utilized formal communication channel. In recent years, there has been a surge in email usage, which has exacerbated the problems presented by spam emails. Spam, often known as junk email, is the act of sending unsolicited mass messages to a large number of people. ‘Ham’ refers to emails that are meaningful but of a different type. Every day, the average email user receives roughly 40-50 emails. Spammers earn roughly 3.5 million dollars per year from spam, resulting in financial damages on both a personal and institutional level. As a result, consumers devote a large amount of their working time to these emails. Spam is said to account for more than half of all email server traffic, sending out a vast volume of undesired and uninvited bulk emails.

They squander user resources on useless output, lowering productivity. Spammers use spam for marketing goals to spread malicious criminal acts such as identity theft, financial disruptions, stealing sensitive information, and reputational damage.

The existing model of the system: –

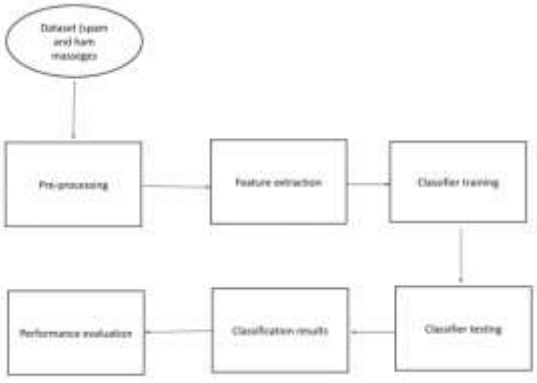
Spam refers to the term, which is related to undesired content with low-quality information, Spam referred to the major drawback of mobile business. When comes to spam detection in the campus network they did the analysis using Incremental Learning. For Collecting Spam detection on web pages. Moreover Sending out a Spam message was also analyzed. Data Collection was done privately by a limited company. From the data Collection. There also anti-spam filter system was evolved. Many parallel and distributed computing system has also processed this spam system. Machine learning algorithm provides accurate result. Text Mining analysis done separates ham and spam separately.

Proposed model of the system: –

As we look at spam detection systems that use Machine Learning (ML) techniques, it's vital to take a look at the history of ML in the field as well as the many methods that are now used to identify spam. Researchers have discovered that the content of spam emails, as well as their operational procedures, evolve with time. As a result, the tactics that are currently effective may become obsolete in the near future. The conceptual drift [8] is a term used to describe this occurrence. Machine Learning is an engineering approach that allows computational instruments to behave without being explicitly programmed. Because of the ML system's ability to evolve, limiting concept drift, this strategy is a significant help in detecting and combating spam.

In the next section, we'll go through a variety of machine learning techniques, approaches, and algorithms, as well as the benefits of each, using Supervised, Unsupervised, and Semi-Supervised Machine Learning algorithms Approaches.

System Architecture: –



Spam classification using Artificial Intelligence

System Requirements: –

Hardware

- OS – Windows 7, 8, and 10 (32 and 64 bit)
- RAM – 4GB

Software:

- Python
- Anaconda navigator
- Python built-in module

DATASET: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

	A	B
2	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
3	ham	Ok lar... Joking wif u oni...
4	spam	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&C's apply 084
5	ham	U dun say so early hor... U c already then say...
6	ham	Nah I don't think he goes to usf, he lives around here though
7	spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, å£1.
8	ham	Even my brother is not like to speak with me. They treat me like aids patent.
9	ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to cop
10	spam	WINNER!! As a valued network customer you have been selected to receivea å£900 prize reward! To claim call 09061701461. Claim code K
11	spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update C
12	ham	I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.
13	spam	SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 in
14	spam	URGENT! You have won a 1 week FREE membership in our å£100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net
15	ham	I've been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulfil my promi
16	ham	I HAVE A DATE ON SUNDAY WITH WILL!!
17	spam	XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap. xxxmobilemovieclub.cor
18	ham	Oh k...i'm watching here:)
19	ham	Eh u remember how 2 spell his name... Yes i did. He v naughty make until i v wet.
20	ham	Fine if thatåÕs the way u feel. ThatåÕs the way its gota b

CODE:

```
import numpy as np
import pandas as pd
df = pd.read_csv('spam.csv')
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

```
df.shape
```

(5572, 5)

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()
```

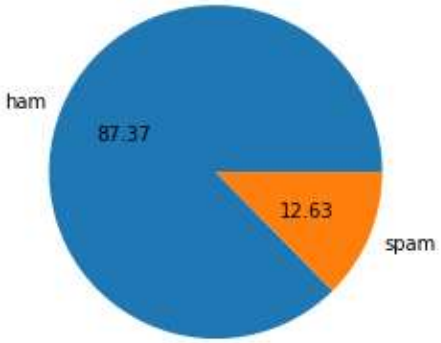
Out[29]:

	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()
```

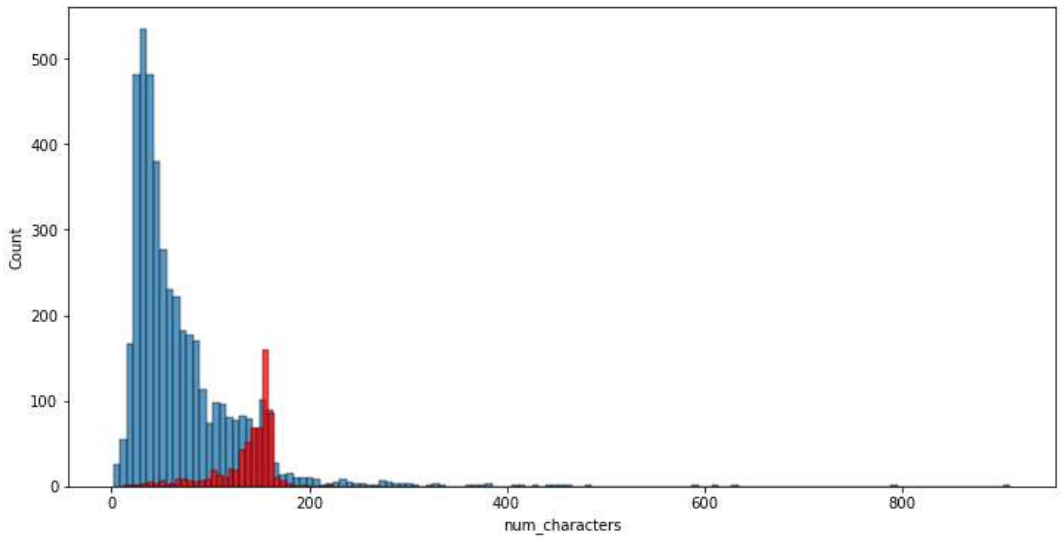
Out[58]:

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

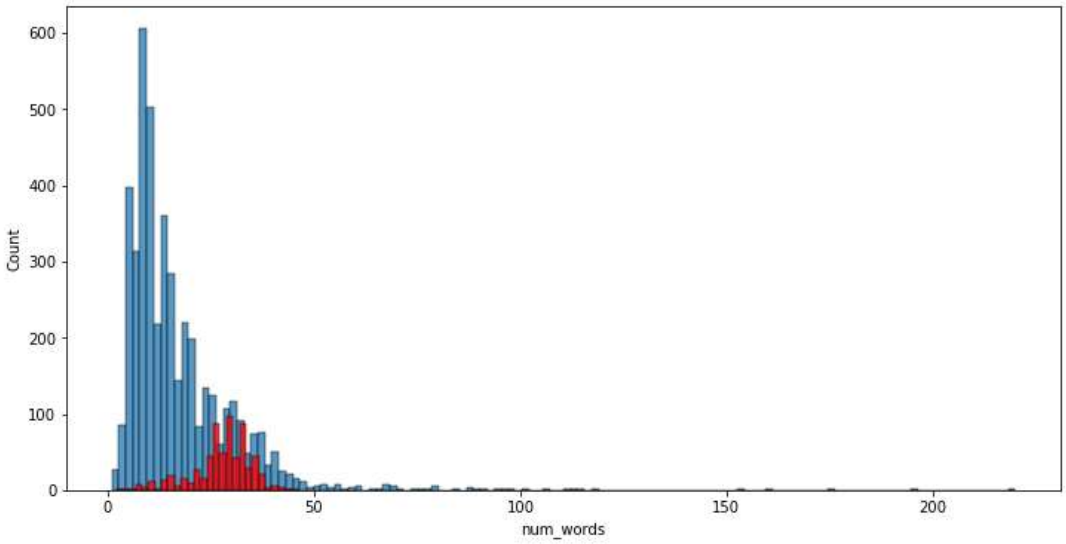
In [84]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
```

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



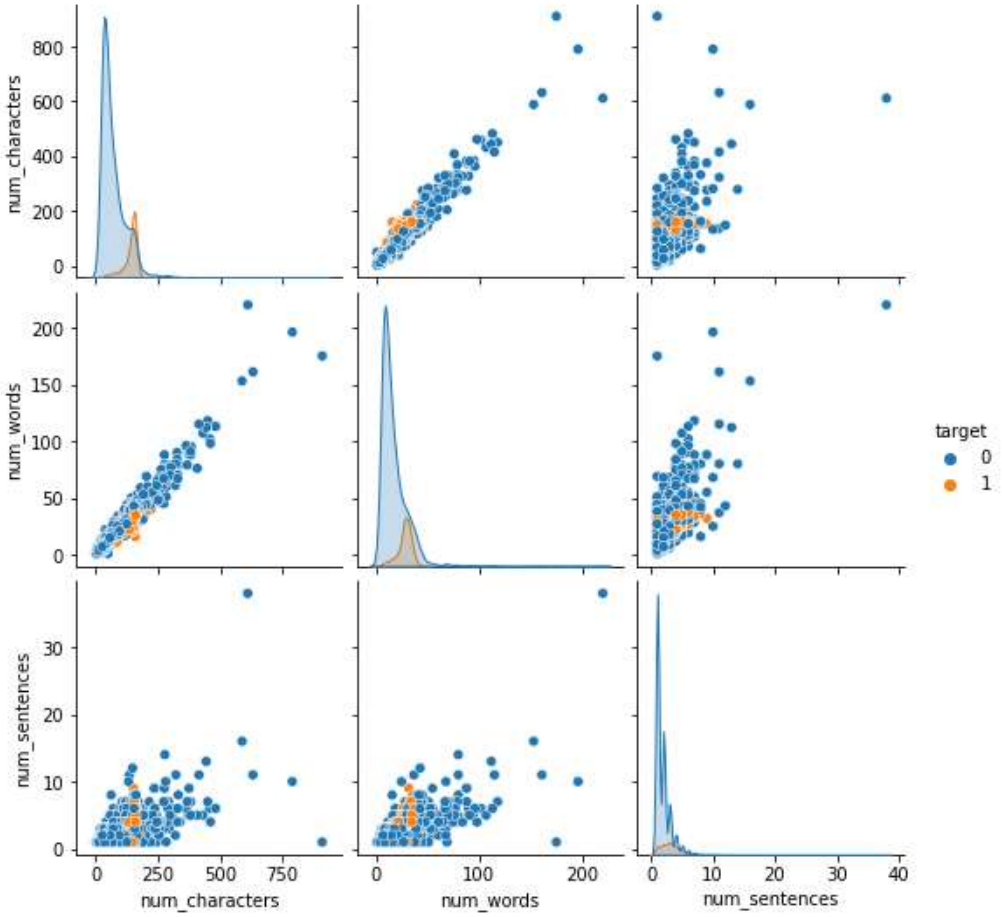
In [85]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
```

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



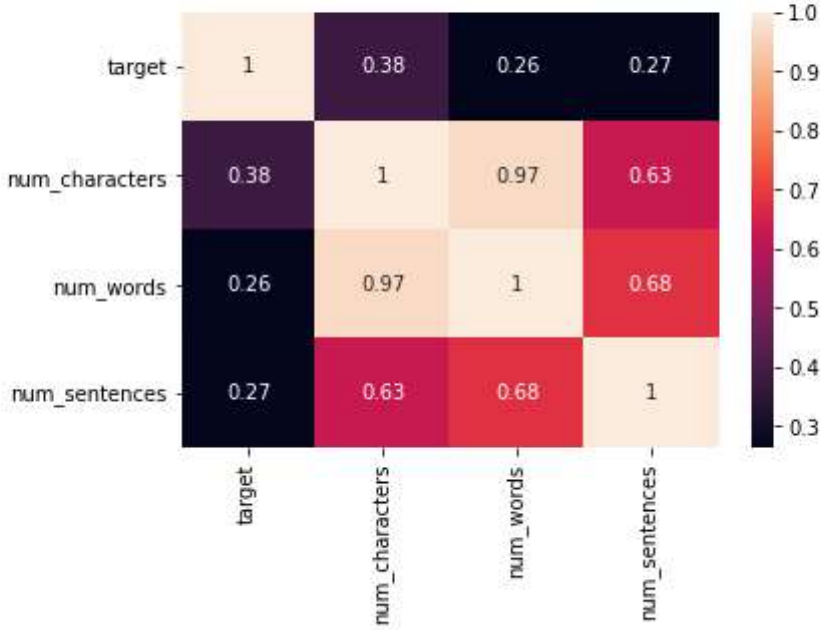
In [86]:
```
sns.pairplot(df,hue='target')
```

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**

```
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)
```

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.")

'gon na home soon want talk stuff anymor tonight k cri enough today'

df['text'][10]

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

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
ps.stem('loving')
```

'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')
```

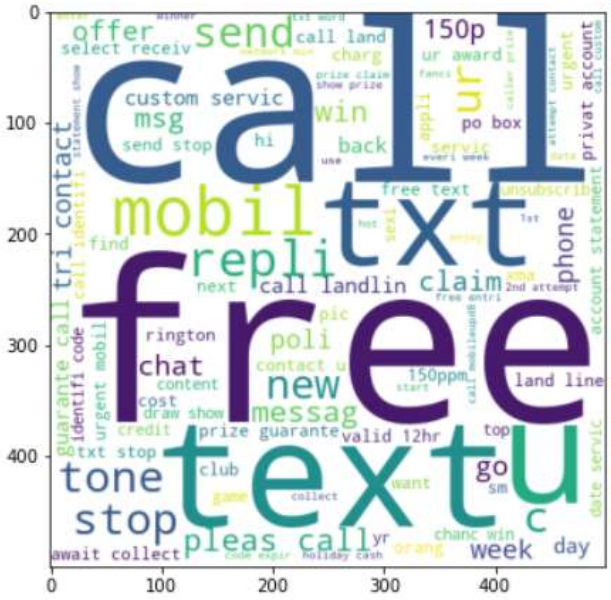
In [233…

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

In [236…

```
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

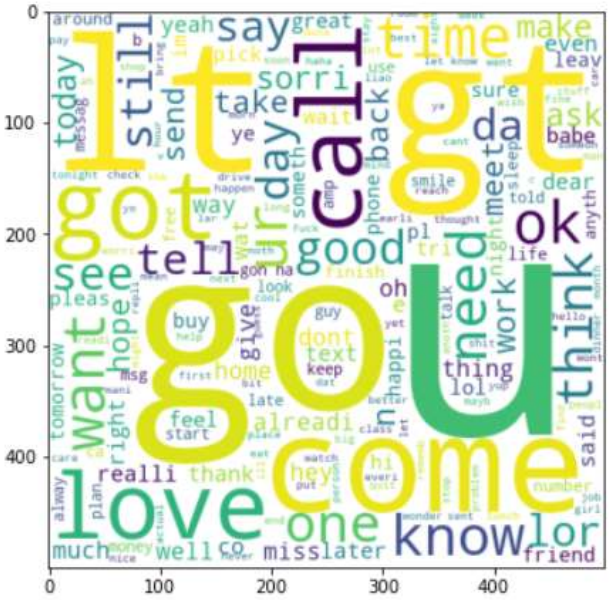
Out[236… <matplotlib.image.AxesImage at 0x16f87ea8cd0>



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)
```

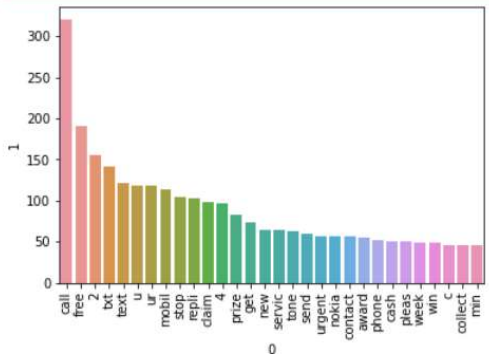
In [274…
```
len(spam_corpus)
```

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…

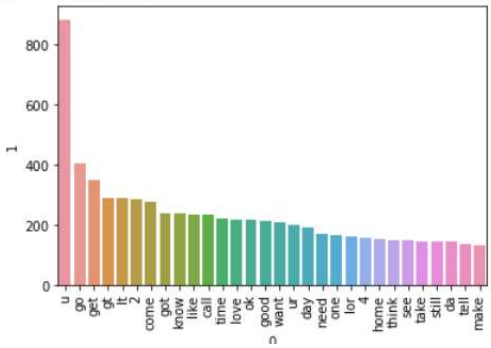
```
len(ham_corpus)
```

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 keywor
rgs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an exp
it keyword will result in an error or misinterpretation.
  warnings.warn(
```



```
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

```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
X = tfidf.fit_transform(df['transformed_text']).toarray()
X.shape
(5169, 3000)
y = df['target'].values
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
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
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
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
# tfidf --> MNB
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
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)
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}
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
train_classifier(svc,X_train,y_train,X_test,y_test)
(0.9729206963249516, 0.9741379310344828)
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)
```

```
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 set 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")
```

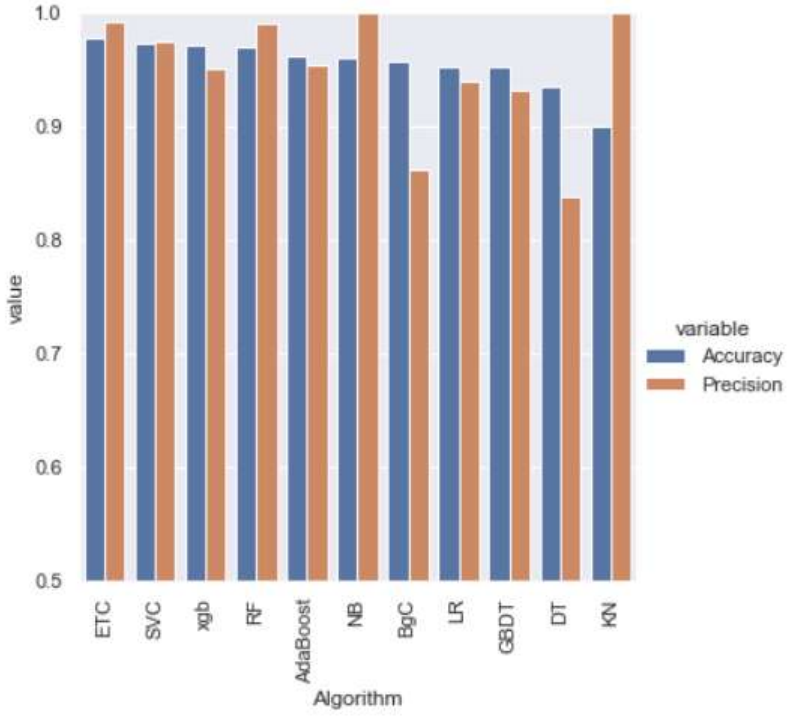
In [365…
```
performance_df1
```

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

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 [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	0.9284
1	NB	0.959381	1.000000	0.971954	1.000000	0.978723	0.946154	0.9400
2	ETC	0.977756	0.991453	0.979691	0.975610	0.979691	0.975610	0.9767
3	RF	0.970019	0.990826	0.975822	0.982906	0.975822	0.982906	0.9748
4	SVC	0.972921	0.974138	0.974855	0.974576	0.971954	0.943089	0.8665
5	AdaBoost	0.962282	0.954128	0.961315	0.945455	0.961315	0.945455	0.9719
6	xgb	0.971954	0.950413	0.968085	0.933884	0.968085	0.933884	0.9700
7	LR	0.951644	0.940000	0.956480	0.969697	0.967118	0.964286	0.9613
8	GBDT	0.951644	0.931373	0.946809	0.927835	0.946809	0.927835	0.9487
9	BgC	0.957447	0.861538	0.959381	0.869231	0.959381	0.869231	0.9680
10	DT	0.935203	0.838095	0.931335	0.831683	0.932302	0.840000	0.9439

svc = SVC(kernel='sigmoid', gamma=1.0,probability=True)
mnb = MultinomialNB()

```
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)

from sklearn.ensemble import VotingClassifier
voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc)],voting='soft')
voting.fit(X_train,y_train)
VotingClassifier(estimators=[('svm',
                              SVC(gamma=1.0, kernel='sigmoid',
                                  probability=True)),
                             ('nb', MultinomialNB()),
                             ('et',
                              ExtraTreesClassifier(n_estimators=50,
                                                   random_state=2))],
                 voting='soft')
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
# Applying stacking
estimators=[('svm', svc), ('nb', mnb), ('et', etc)]
final_estimator=RandomForestClassifier()
from sklearn.ensemble import StackingClassifier
clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)
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
```