```
import numpy as np
import pandas as pd
```

```
df = pd.read_csv('/spam.csv')
```

```
df.head(5)
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN

```
df.shape
```

```
(5572, 5)
```

Steps Involved in this Process :

1)Data Cleaning 2)EDA 3)Text Preprocessing 4)ModelBuilding 5)Evaluation 6)Improvement

1.Data Cleaning

```
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
```

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

```
df.head(5)
```

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

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

	target	text
3133	ham	Ok going to sleep. Hope i can meet her.
5015	ham	Hey gals...U all wanna meet 4 dinner at n��te?
3465	ham	Actually fuck that, just do whatever, do find ...
2397	ham	Neshanth..tel me who r u?
4212	ham	I attended but nothing is there.

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```

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

```
df.head()
```

	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...

```
data = df['text'].copy()
```

```
# missing values
df.isnull().sum()
```

```
target    0
text      0
dtype: int64
```

```
# check for duplicate values
df.duplicated().sum()
```

```
409
```

```
# remove duplicates
df = df.drop_duplicates(keep='first')
```

```
df.duplicated().sum()
```

```
0
```

```
df.shape
```

```
(5163, 2)
```

```
df.head()
```

	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...

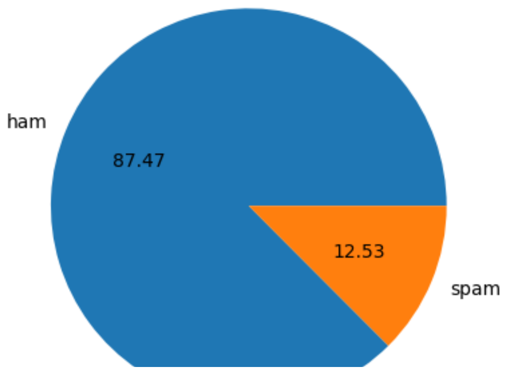
```
data = df['text'].copy()
```

**2)EDA **

```
df['target'].value_counts()
```

```
0    4516
1     647
Name: target, dtype: int64
```

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



Data is Imbalanaced

```
import nltk
!pip install nltk
nltk.download('punkt')
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.6)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2022.10.31)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.65.0)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

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

```
df.head()
```

	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

```
df['num_words'] = df['text'].apply(lambda x : len(nltk.word_tokenize(x)))
```

```
df.head()
```

	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

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

```
df.head()
```

	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

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

	num_characters	num_words	num_sentences
count	5163.000000	5163.000000	5163.000000
mean	78.811156	18.444122	1.963974
std	58.153491	13.327940	1.448008
min	2.000000	1.000000	1.000000
25%	36.000000	9.000000	1.000000
50%	60.000000	15.000000	1.000000
75%	116.000000	26.000000	2.000000
max	910.000000	220.000000	38.000000

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

	num_characters	num_words	num_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.428919	17.123782	1.820195
std	56.356310	13.493970	1.383657
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

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

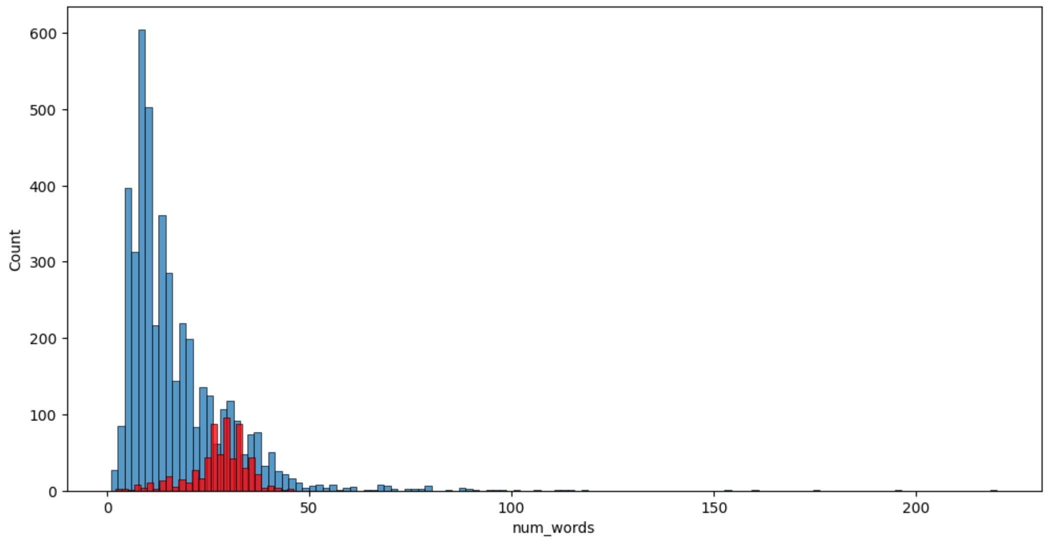
	num_characters	num_words	num_sentences
count	647.000000	647.000000	647.000000
mean	137.318393	27.659969	2.967543
std	30.096861	7.038065	1.490069
min	13.000000	2.000000	1.000000
25%	130.000000	25.000000	2.000000
50%	148.000000	29.000000	3.000000
75%	157.000000	32.000000	4.000000
max	223.000000	46.000000	9.000000

```
import seaborn as sns
plt.figure(figsize=(12,6))
sns.histplot(df[df['target']==0]['num_characters'])
sns.histplot(df[df['target']==1]['num_characters'],color='red')
```

```
<Axes: xlabel='num_characters', ylabel='Count'>
```

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

```
<Axes: xlabel='num_words', ylabel='Count'>
```

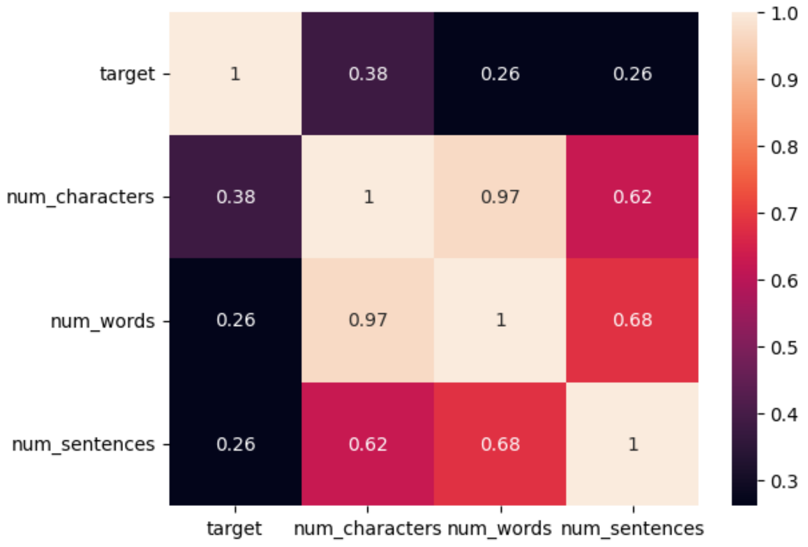


```
sns.pairplot(df,hue='target')
```

```
<seaborn.axisgrid.PairGrid at 0x7fec31339840>
```

```
sns.heatmap(df.corr(),annot=True)
```

```
<ipython-input-35-8df7bcac526d>:1: FutureWarning: The default value of numeric_only in DataFrame.corr i
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```



	target	num_characters	num_words	num_sentences
target	1	0.38	0.26	0.26
num_characters	0.38	1	0.97	0.62
num_words	0.26	0.97	1	0.68
num_sentences	0.26	0.62	0.68	1

3.Data Preprocessing

Steps involved in this process are :

a.Lower case

b.Tokenization

c.Removing special characters

d.Removing stop words and punctuation

e.Stemming

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stopwords.words('english')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers',
 'herself',
 'it',
 "it's",
 'its',
 'itself',
 'they',
 'them',
 'their',
 'theirs',
 'themselves',
 'what',
 'which',
 'who',
 'whom',
 'this',
 'that',
 "that'll",
 'these',
 'those',
 'am',
 'is',
 'are',
 'was',
 'were',
 'be',
 'been',
 'being',
 'have',
 'has',
 'had',
 'having',
```

```
import string
string.punctuation
```

```
     '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

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

```
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('Hi How are You 20')
```

```
     'hi 20'
```

```
df['text'][2000]
```

```
     'But i'll b going 2 sch on mon. My sis need 2 take smth.'
```

```
transform_text(df['text'][2000])
```

```
     'b go 2 sch mon si need 2 take smth'
```

```
df['transformed_text'] = df['text'].apply(transform_text)
df.head()
```

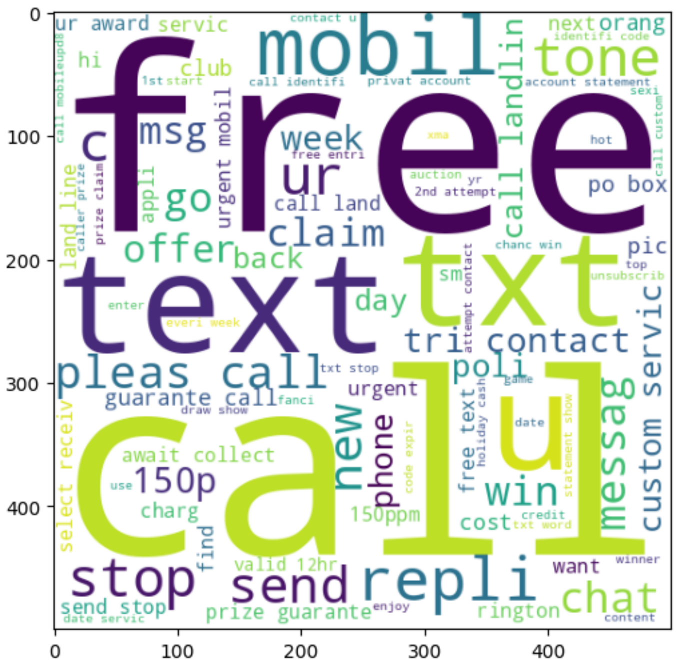
	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
2	1	comp to win FA Cup fina	155	37	2	cup final tkt 21
[illegible]	0	U c already then sav...	[illegible]	[illegible]	1	sav

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

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

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

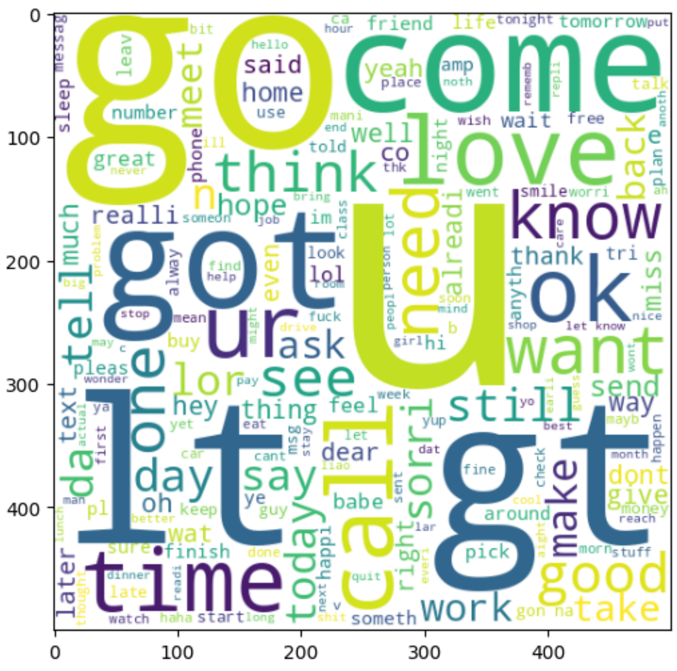
```
<matplotlib.image.AxesImage at 0x7fec2cc25510>
```



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

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

```
<matplotlib.image.AxesImage at 0x7fec2cb4eb00>
```



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

```
len(spam_corpus)
```

```
9861
```

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

```
len(ham_corpus )
```

```
35306
```

```
from collections import Counter
pd.DataFrame(Counter(ham_corpus).most_common(10))
```

	0	1
0	u	883
1	go	404
2	get	349
3	gt	288
4	lt	287
5	2	284
6	come	275
7	got	236
8	know	236
9	like	234

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

	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 alreadv then sav...	49	13	1	u dun say earli hor u c alreadi sav

4.Model Building

```
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
```

```
(5163, 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(precision_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
```

```
0.861568247821878
0.452991452991453
[[784 128]
 [ 15 106]]
```

```
mnb.fit(x_train,y_train)
y_pred2 = mnb.predict(x_test)
print(accuracy_score(y_test,y_pred2))
print(precision_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
```

```
0.9719264278799613
0.9893617021276596
[[911   1]
 [ 28  93]]
```

```
bnb.fit(x_train,y_train)
y_pred3 = bnb.predict(x_test)
print(accuracy_score(y_test,y_pred3))
print(precision_score(y_test,y_pred3))
print(confusion_matrix(y_test,y_pred3))
```

```
0.978702807357212
0.9714285714285714
[[909   3]
 [ 19 102]]
```

```
# 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)
```

```
(0.978702807357212, 0.9805825242718447)
```

```
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 -  0.978702807357212
Precision -  0.9805825242718447
For  KN
Accuracy -  0.9225556631171346
Precision -  1.0
For  NB
Accuracy -  0.9719264278799613
Precision -  0.9893617021276596
For  DT
Accuracy -  0.936108422071636
Precision -  0.8021978021978022
For  LR
Accuracy -  0.9554695062923524
Precision -  0.9310344827586207
For  RF
Accuracy -  0.972894482090997
Precision -  0.979381443298969
For  AdaBoost
Accuracy -  0.9622458857696031
Precision -  0.9270833333333334
For  BgC
Accuracy -  0.9641819941916747
Precision -  0.868421052631579
For  ETC
Accuracy -  0.9777347531461762
Precision -  0.9803921568627451
For  GBDT
Accuracy -  0.9535333978702807
Precision -  0.9397590361445783
For  xgb
Accuracy -  0.9622458857696031
Precision -  0.91
```

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

	Algorithm	Accuracy	Precision
1	KN	0.922556	1.000000
2	NB	0.971926	0.989362
0	SVC	0.978703	0.980583
8	ETC	0.977735	0.980392
5	RF	0.972894	0.979381
9	GBDT	0.953533	0.939759
4	LR	0.955470	0.931034
6	AdaBoost	0.962246	0.927083
10	xgb	0.962246	0.910000
7	BgC	0.964182	0.868421
3	DT	0.936108	0.802198

```
n=67
input = x[n]
input_reshaped=input.reshape(1,-1)
prediction=mnb.predict(input_reshaped)
if prediction == 1 :
    print(data[n])
    print('SPAM')
else:
    print(data[n])
    print('NOT SPAM')
```

```
Urgent UR awarded a complimentary trip to EuroDisinc Trav, Aco&Entry41 Or �1000. To claim txt DIS to 87121 18+6*�1.50(moreFrmMob.
SPAM
```