# SMS SPAM CLASSIFIER

## Overview :-

The SMS Spam Collection is a dataset for SMS Spam research, featuring 5,574 English SMS messages tagged as "ham" (legitimate) or "spam." Each entry consists of two columns: 'v1' (label - ham/spam) and 'v2' (raw text). The dataset includes messages from various sources, such as the Grumbletext Web site, NUS SMS Corpus, Caroline Tag's PhD Thesis, and SMS Spam Corpus v.0.1 Big.


In [1]:
```
# import libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

In [2]:
```
# read data set

df = pd.read_csv("spam.csv", encoding='latin1')
df.head()
```

Out[2]:

| | 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 |
| 4 | ham | Nah I don't think he goes to usf, he lives aro... | NaN | NaN | NaN |

In [3]:
```
df.shape
```

Out[3]:
```
(5572, 5)
```

# 1. Data Cleaning

In [4]:
```
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 [5]:
```
# drop last 3 columns

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

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

Out[6]:

| | 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... |

In [7]:
```
# Rename the columns

df.rename(columns={'v1':'target','v2':'text'},inplace=True)
df.head()
```

Out[7]:

| | target | text |
|---|---|---|
| **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... |

In [8]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```

In [9]:
```
df['target'] = encoder.fit_transform(df['target'])
```

In [10]:
```
df.head()
```

Out[10]:

| | 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 [11]:
```
df.isnull().sum()
```

Out[11]:
```
target    0
text      0
dtype: int64
```

In [12]:
```
df.duplicated().sum()
```

Out[12]:
```
403
```

In [13]:
```
df = df.drop_duplicates(keep='first')
```

In [14]:
```
df.duplicated().sum()
```

Out[14]:
```
0
```

In [15]:
```
df.shape
```

Out[15]:
```
(5169, 2)
```

## 2. EDA

In [16]:
```
df.head()
```

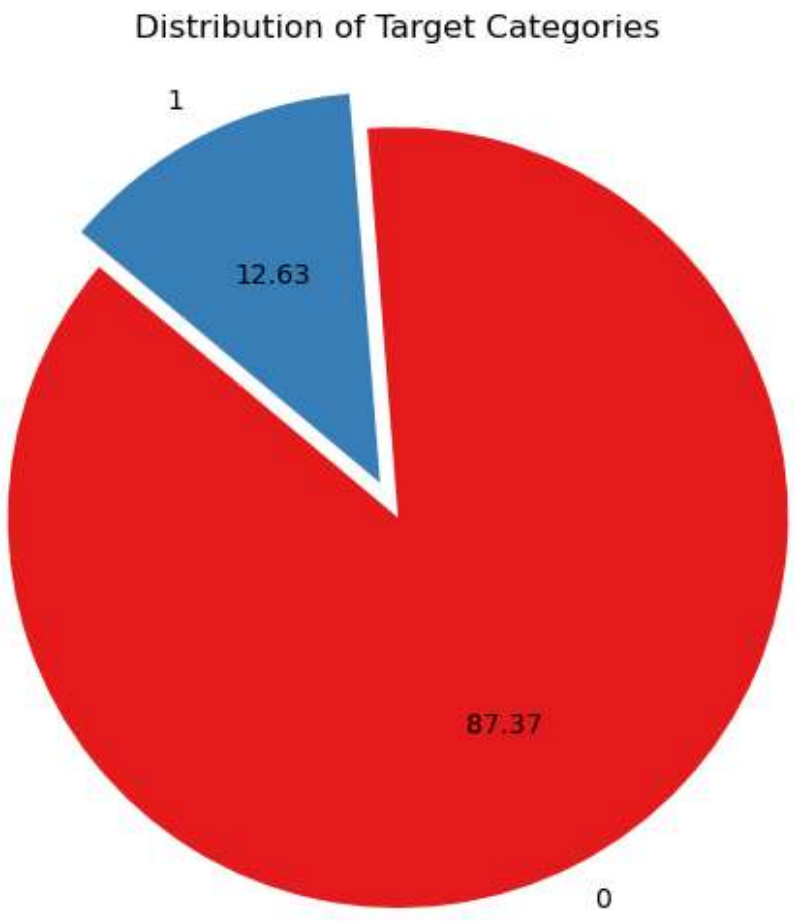
Out[16]:

| | 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... |

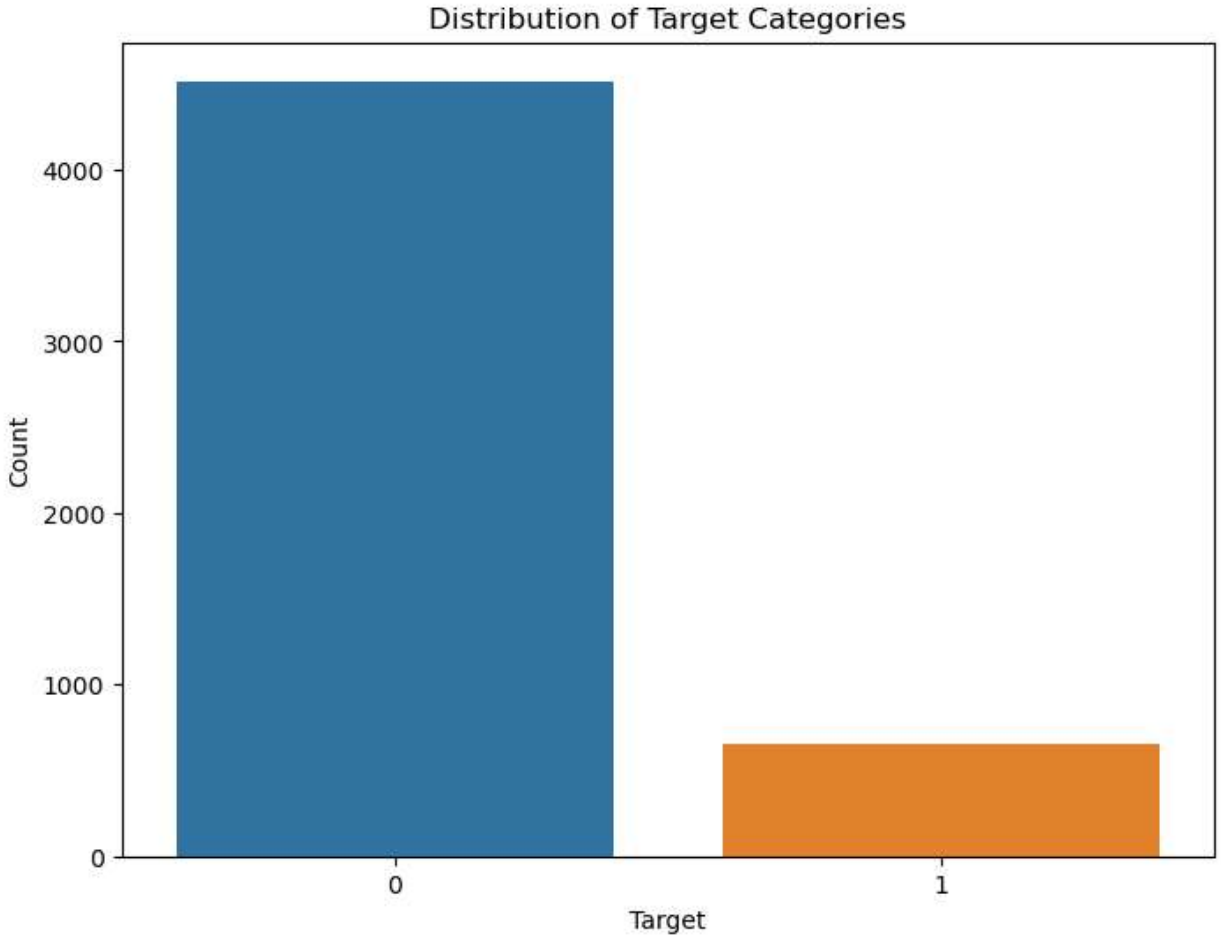
In [17]:
```
df['target'].value_counts()
```

Out[17]:
```
0    4516
1     653
Name: target, dtype: int64
```

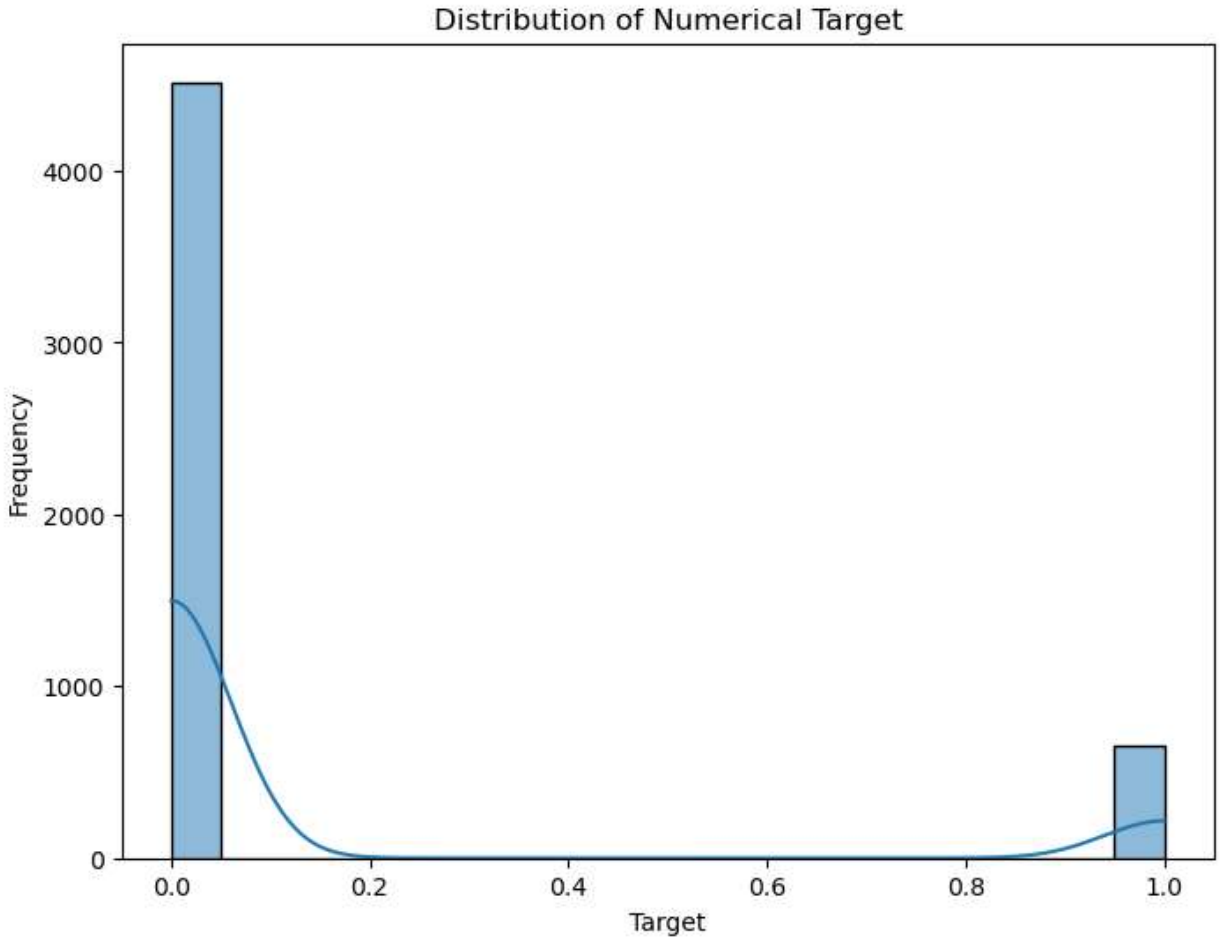
In [18]:
```
# Data
value_counts = df['target'].value_counts()
labels = value_counts.index
explode = (0.1, 0)  # Explode first slice (if needed)

# Plot
plt.figure(figsize=(8, 6))
plt.pie(value_counts, labels=labels, explode=explode, autopct="%0.2f", startangle=140,
plt.title('Distribution of Target Categories')
plt.axis('equal')  # Ensure pie is drawn as a circle
plt.show()
```



In [19]:
```
# Bar plot for categorical target
plt.figure(figsize=(8, 6))
sns.countplot(x='target', data=df)
plt.title('Distribution of Target Categories')
plt.xlabel('Target')
plt.ylabel('Count')
plt.show()
```




In [20]:
```
# Histogram for numerical target
plt.figure(figsize=(8, 6))
sns.histplot(df['target'], bins=20, kde=True)
plt.title('Distribution of Numerical Target')
plt.xlabel('Target')
plt.ylabel('Frequency')
plt.show()
```




In [21]:
```
# Data is imbalance
```

In [22]:
```
import nltk
```

In [23]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\shrut\anaconda3\lib\site-packages (3.
7)
Requirement already satisfied: click in c:\users\shrut\anaconda3\lib\site-packages (f
rom nltk) (8.0.4)
Requirement already satisfied: tqdm in c:\users\shrut\anaconda3\lib\site-packages (fr
om nltk) (4.64.1)
Requirement already satisfied: joblib in c:\users\shrut\anaconda3\lib\site-packages
(from nltk) (1.1.1)
Requirement already satisfied: regex>=2021.8.3 in c:\users\shrut\anaconda3\lib\site-p
ackages (from nltk) (2022.7.9)
Requirement already satisfied: colorama in c:\users\shrut\anaconda3\lib\site-packages
(from click->nltk) (0.4.6)
```

In [24]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\shrut\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[24]:
```
True
```

In [25]:
```
# num of characters
```

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

In [26]:
```
df.head()
```

Out[26]:

| | 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 [27]:
```
# num of words

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

In [28]:
```
df.head()
```

Out[28]:

| | 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 [29]:
```
# num of sentences
df['num_sentences'] = df['text'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

In [30]:
```
df.head()
```

Out[30]:

| | 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 [31]:
```
df[['num_characters','num_words','num_sentences']].describe()
```

Out[31]:

| | num_characters | num_words | num_sentences |
|---|---|---|---|
| **count** | 5169.000000 | 5169.000000 | 5169.000000 |
| **mean** | 78.977945 | 18.453279 | 1.947185 |
| **std** | 58.236293 | 13.324793 | 1.362406 |
| **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 | 28.000000 |

In [32]:
```
# ham msg

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

Out[32]:

| | num_characters | num_words | num_sentences |
|---|---|---|---|
| **count** | 4516.000000 | 4516.000000 | 4516.000000 |
| **mean** | 70.459256 | 17.120903 | 1.799601 |
| **std** | 56.358207 | 13.493725 | 1.278465 |
| **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 | 28.000000 |

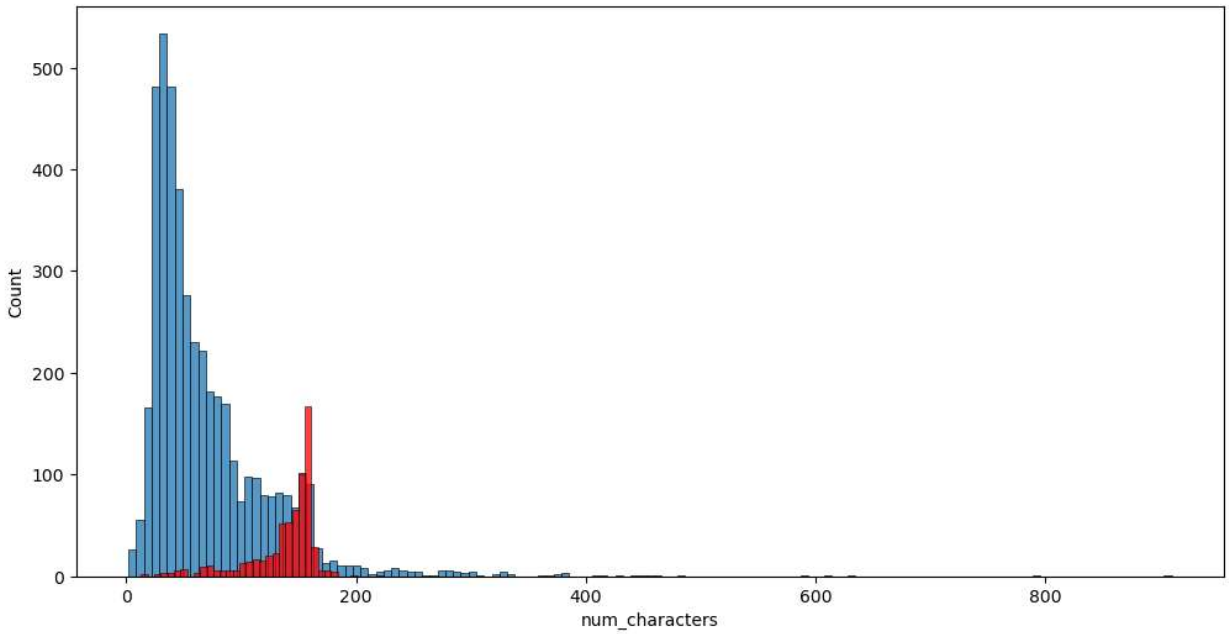
In [33]:
```
# spam msg

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

Out[33]:

| | num_characters | num_words | num_sentences |
|---|---|---|---|
| **count** | 653.000000 | 653.000000 | 653.000000 |
| **mean** | 137.891271 | 27.667688 | 2.967841 |
| **std** | 30.137753 | 7.008418 | 1.483201 |
| **min** | 13.000000 | 2.000000 | 1.000000 |
| **25%** | 132.000000 | 25.000000 | 2.000000 |
| **50%** | 149.000000 | 29.000000 | 3.000000 |
| **75%** | 157.000000 | 32.000000 | 4.000000 |
| **max** | 224.000000 | 46.000000 | 8.000000 |

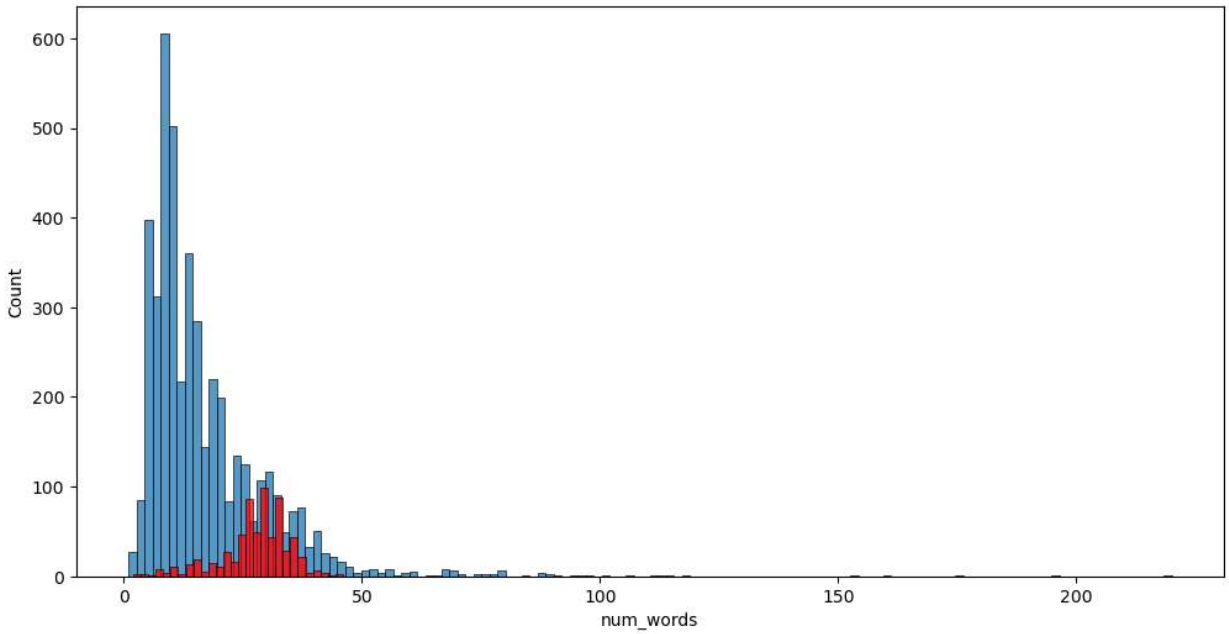
In [34]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
```

Out[34]:
```
<Axes: xlabel='num_characters', ylabel='Count'>
```



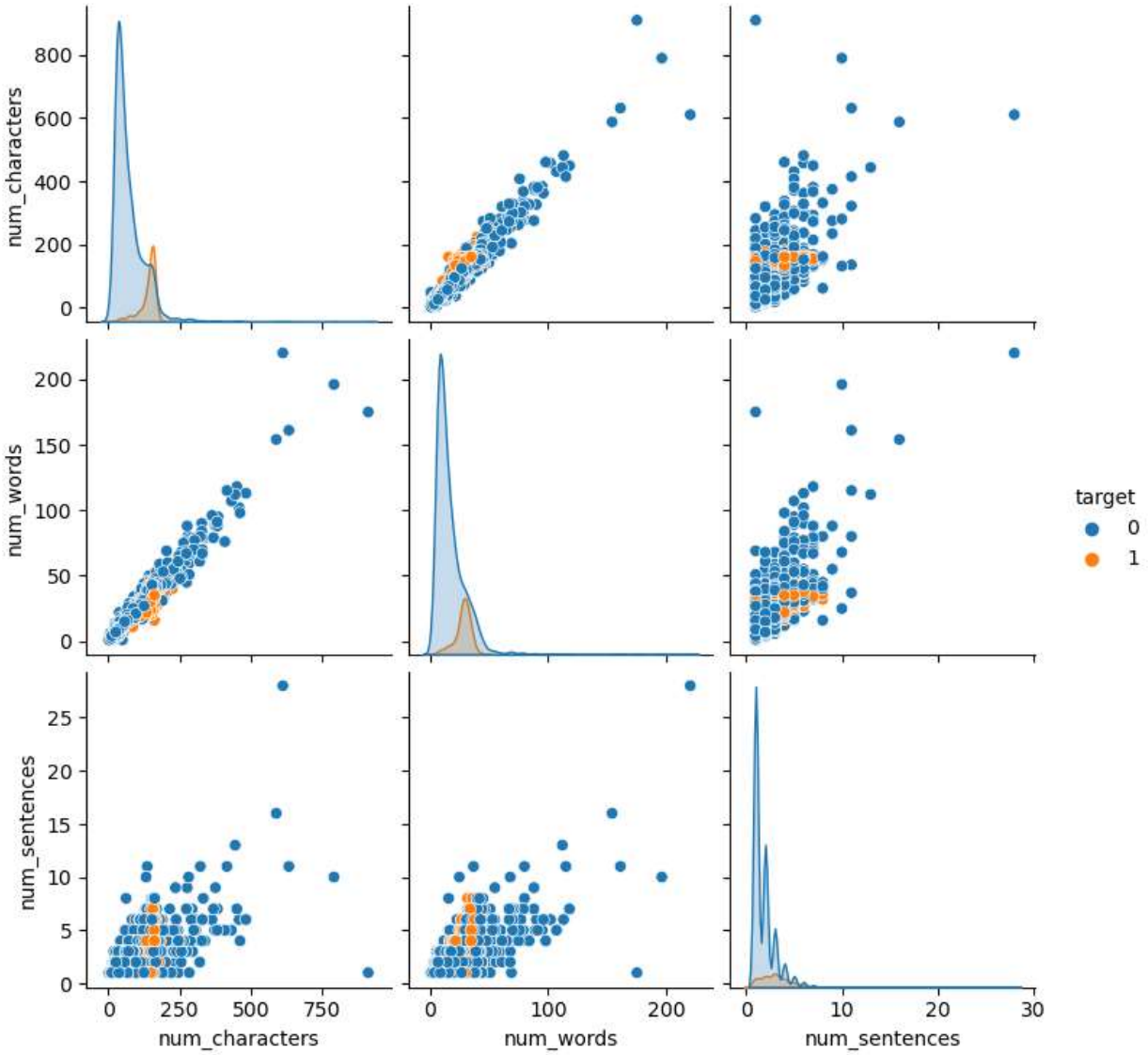

In [35]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
```

Out[35]:
```
<Axes: xlabel='num_words', ylabel='Count'>
```




In [36]:
```
sns.pairplot(df,hue='target')
```

Out[36]:
```
<seaborn.axisgrid.PairGrid at 0x20193012ce0>
```




In [37]:
```
numeric_values = df.select_dtypes( include = ['number'])
sns.heatmap(numeric_values.corr(),annot=True)
```

Out[37]:
```
<Axes: >
```


## 3. Data Preprocessing

In [38]:
```
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import string
```

In [39]:
```
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\shrut\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\shrut\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[39]:
```
True
```

In [ ]:
```
```

In [40]:
```
def transform_text(text):
    ps = PorterStemmer()
    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 [41]:
```
transform_text("I'm gonna be home soon and i don't want to talk about this stuff anymo
```

Out[41]:
```
'gon na home soon want talk stuff anymor tonight k cri enough today'
```

In [42]:
```
df['text'][10]
```

Out[42]:
```
"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 [43]:
```
ps = PorterStemmer()
ps.stem('loving')
```

Out[43]:
```
'love'
```

In [44]:
```
ps.stem('dancing')
```

Out[44]:
```
'danc'
```

In [45]:
```
df['transformed_text'] = df['text'].apply(transform_text)
```

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

Out[46]:

| | 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 [47]:
```
!pip install wordcloud
```

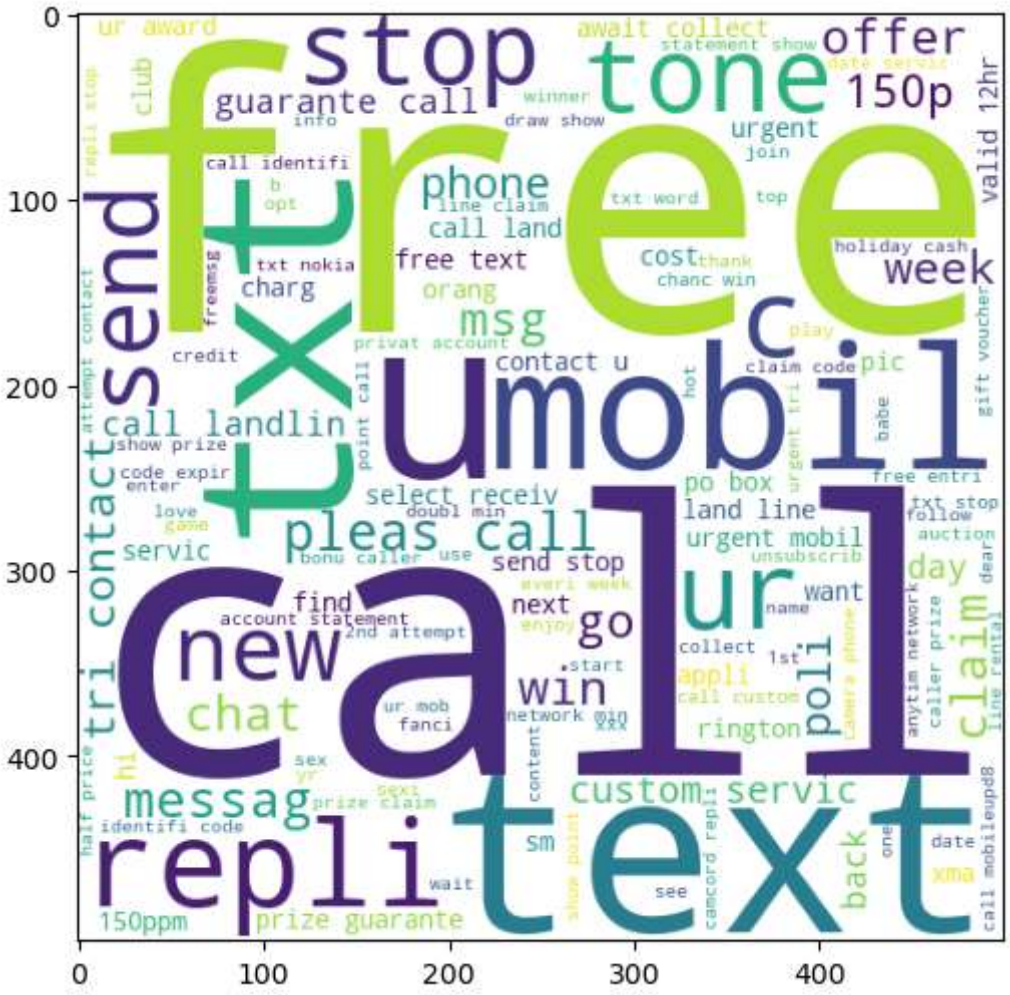
```
Requirement already satisfied: wordcloud in c:\users\shrut\anaconda3\lib\site-package
s (1.9.3)
Requirement already satisfied: matplotlib in c:\users\shrut\anaconda3\lib\site-packag
es (from wordcloud) (3.7.0)
Requirement already satisfied: numpy>=1.6.1 in c:\users\shrut\anaconda3\lib\site-pack
ages (from wordcloud) (1.23.5)
Requirement already satisfied: pillow in c:\users\shrut\anaconda3\lib\site-packages
(from wordcloud) (9.4.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\shrut\anaconda3\lib\site-
packages (from matplotlib->wordcloud) (1.0.5)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\shrut\anaconda3\lib\site
-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\shrut\anaconda3\lib\s
ite-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\shrut\anaconda3\lib\site-
packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: packaging>=20.0 in c:\users\shrut\anaconda3\lib\site-p
ackages (from matplotlib->wordcloud) (22.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\shrut\anaconda3\lib\site
-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: cycler>=0.10 in c:\users\shrut\anaconda3\lib\site-pack
ages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: six>=1.5 in c:\users\shrut\anaconda3\lib\site-packages
(from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
```

In [48]:
```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
```

In [49]:
```
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep=" "))
```

In [50]:
```
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

Out[50]:
```
<matplotlib.image.AxesImage at 0x20194fcb850>
```




In [51]:
```
ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(sep=" "))
```

In [52]:
```
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
```

Out[52]:
```
<matplotlib.image.AxesImage at 0x20195427640>
```


In [53]:
```
df.head()
```

Out[53]:

| | 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 [54]:
```
spam_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

In [55]:
```
len(spam_corpus)
```

Out[55]:
```
9939
```

In [56]:
```
from collections import Counter

pd.DataFrame(Counter(spam_corpus).most_common(30))
```

Out[56]:

| | 0 | 1 |
|---|---|---|
| 0 | call | 320 |
| 1 | free | 191 |
| 2 | 2 | 155 |
| 3 | txt | 141 |
| 4 | text | 122 |
| 5 | u | 119 |
| 6 | ur | 119 |
| 7 | mobil | 114 |
| 8 | stop | 104 |
| 9 | repli | 103 |
| 10 | claim | 98 |
| 11 | 4 | 97 |
| 12 | prize | 82 |
| 13 | get | 74 |
| 14 | new | 64 |
| 15 | servic | 64 |
| 16 | tone | 63 |
| 17 | send | 60 |
| 18 | urgent | 57 |
| 19 | nokia | 57 |
| 20 | contact | 56 |
| 21 | award | 55 |
| 22 | phone | 52 |
| 23 | cash | 51 |
| 24 | pleas | 51 |
| 25 | week | 49 |
| 26 | win | 48 |
| 27 | c | 45 |
| 28 | collect | 45 |
| 29 | min | 45 |

In [57]:
```
ham_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
```

In [58]:
```
len(ham_corpus)
```

Out[58]:
```
35394
```

In [59]:
```
pd.DataFrame(Counter(ham_corpus).most_common(30))
```

Out[59]:

| | 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 |
| 10 | call | 233 |
| 11 | time | 219 |
| 12 | ok | 217 |
| 13 | love | 216 |
| 14 | good | 213 |
| 15 | want | 208 |
| 16 | ur | 197 |
| 17 | day | 190 |
| 18 | need | 170 |
| 19 | one | 165 |
| 20 | lor | 159 |
| 21 | 4 | 156 |
| 22 | home | 152 |
| 23 | think | 149 |
| 24 | see | 147 |
| 25 | take | 143 |
| 26 | still | 143 |
| 27 | da | 138 |
| 28 | tell | 133 |
| 29 | make | 129 |

In [60]:
```
df.head()
```

Out[60]:

| | 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 [61]:
```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
```

In [62]:
```
x = cv.fit_transform(df['transformed_text']).toarray()
```

In [63]:
```
x.shape
```

Out[63]:
```
(5169, 6708)
```

In [64]:
```
y = df['target'].values
```

In [65]:
```
from sklearn.model_selection import train_test_split
```

In [66]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=2)
```

In [67]:
```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

In [68]:
```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [69]:
```
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))
```

```
0.8800773694390716
[[792 104]
 [ 20 118]]
0.5315315315315315
```

In [70]:
```
mnb.fit(x_train,y_train)
y_pred1 = mnb.predict(x_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
```

```
0.9642166344294004
[[871  25]
 [ 12 126]]
0.8344370860927153
```

In [71]:
```
bnb.fit(x_train,y_train)
y_pred1 = bnb.predict(x_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
```

```
0.9700193423597679
[[893   3]
 [ 28 110]]
0.9734513274336283
```

In [72]:
```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\shrut\anaconda3\lib\site-packages
(2.0.3)
Requirement already satisfied: scipy in c:\users\shrut\anaconda3\lib\site-packages (f
rom xgboost) (1.10.0)
Requirement already satisfied: numpy in c:\users\shrut\anaconda3\lib\site-packages (f
rom xgboost) (1.23.5)
```

In [73]:
```
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 [74]:
```
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 [75]:
```
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}
```

In [76]:
```
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 [77]:
```
train_classifier(svc,x_train,y_train,x_test,y_test)
```

Out[77]:
```
(0.9284332688588007, 0.7580645161290323)
```

In [80]:
```
accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf, x_train,y_train,x_test,

    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)
    print("="*50)

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

```
For  SVC
Accuracy -  0.9284332688588007
Precision -  0.7580645161290323
==================================================
For  KN
Accuracy -  0.90715667311412
Precision -  1.0
==================================================
For  NB
Accuracy -  0.9642166344294004
Precision -  0.8344370860927153
==================================================
For  DT
Accuracy -  0.9245647969052224
Precision -  0.9166666666666666
==================================================
For  LR
Accuracy -  0.9709864603481625
Precision -  0.9736842105263158
==================================================
For  RF
Accuracy -  0.971953578336557
Precision -  1.0
==================================================
For  AdaBoost
Accuracy -  0.9632495164410058
Precision -  0.9464285714285714
==================================================
For  BgC
Accuracy -  0.9622823984526112
Precision -  0.9159663865546218
==================================================
For  ETC
Accuracy -  0.9777562862669246
Precision -  0.9914529914529915
==================================================
For  GBDT
Accuracy -  0.9439071566731141
Precision -  0.9444444444444444
==================================================
For  xgb
Accuracy -  0.9729206963249516
Precision -  0.9661016949152542
==================================================
```

In [81]:
```
performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores,'Pre
performance_df
```

Out[81]:

| | Algorithm | Accuracy | Precision |
|---|---|---|---|
| **1** | KN | 0.907157 | 1.000000 |
| **5** | RF | 0.971954 | 1.000000 |
| **8** | ETC | 0.977756 | 0.991453 |
| **4** | LR | 0.970986 | 0.973684 |
| **10** | xgb | 0.972921 | 0.966102 |
| **6** | AdaBoost | 0.963250 | 0.946429 |
| **9** | GBDT | 0.943907 | 0.944444 |
| **3** | DT | 0.924565 | 0.916667 |
| **7** | BgC | 0.962282 | 0.915966 |
| **2** | NB | 0.964217 | 0.834437 |
| **0** | SVC | 0.928433 | 0.758065 |

# Conclusion :-

The Random Forest (RF) classifier achieves the highest accuracy (97.20%) and precision (100%) among all algorithms evaluated for SMS spam classification. It demonstrates superior performance in accurately identifying spam messages while minimizing false positives. RandomForestClassifier is recommended as the optimal choice for this task due to its excellent overall performance.