MADE BY GROUP MEMBERS:

MEMBER 1:

NAME: SHUBHAM TYAGI

ENROLL. NO.: 01515603121

BRANCH: INFORMATION TECHNOLOGY

BATCH: 2021-2025

MEMBER 2:

NAME: SAKSHAM JAIN

ENROLL. NO.: 11215603121

BRANCH: INFORMATION TECHNOLOGY

BATCH: 2021-2025

PLEASE DOWNLOAD AND INSTALL ALL THE LIBRARIES BEFORE RUNNING THE PROJECT

In [1]:
```
#Importing the Libraries Required to Perform the Operations

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.naive_bayes import GaussianNB,MultinomialNB, BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier, ExtraTreesClassifier
from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
from xgboost import XGBClassifier, XGBRFClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from wordcloud import WordCloud
from collections import  Counter
import nltk
import pickle
import string


%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
nltk.download('punkt')
nltk.download('stopwords')
```

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

Out[1]: True

In [2]:
```
# Importing Dataset spam.csv into Dataframe df

df = pd.read_csv('spam.csv',encoding='latin-1')
```

In [3]:
```
# Printing First 5 Rows in Dataframe

df.head()
```

Out[3]:

	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 [4]:
```
# Printing any 5 Random Rows in Dataframe

df.sample(5)
```

Out[4]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
744	ham	Men like shorter ladies. Gaze up into his eyes.	NaN	NaN	NaN
357	spam	Ur cash-balance is currently 500 pounds - to m...	NaN	NaN	NaN
4871	ham	Hi dis is yijue i would be happy to work wif Ì...	NaN	NaN	NaN
5178	spam	SMS AUCTION - A BRAND NEW Nokia 7250 is up 4 a...	NaN	NaN	NaN
3778	spam	Claim a 200 shopping spree, just call 08717895...	NaN	NaN	NaN

In [5]:
```
# Returns the Shape (Total number of rows and columns) of DataFrame or in other words,
# Returns a tuple representing the dimensionality of the DataFrame

df.shape
```

Out[5]: (5572, 5)

Operations Performed Throughout the Data Analysis

1. Dataset Cleaning
2. EDA
3. Dataset Preprocessing
4. Models Training Using Various Algorithms
5. Observations (Comparing Accuracy And Precision)
6. Choosing Best Suitable Algorithm For Model
7. Generating Pickle Files

---

1. Data Cleaning

In [6]:
```
#This method prints information about a DataFrame including the index dtype and columns, non-null values and memory usage.

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 columns because they are of no use

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

In [8]:
```
# Printing any 5 Random Values in Dataframe

df.sample(5)
```

Out[8]:

	v1	v2
921	ham	On ma way to school. Can you pls send me ashle...
48	ham	Yeah hopefully, if tyler can't do it I could m...
3984	ham	Whatever, juliana. Do whatever you want.
2423	ham	Lmao but its so fun...
4234	spam	FREEMSG: Our records indicate you may be entit...

In [9]:
```
# Renaming the columns v1 & v2 to target and text
```

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

Out[9]:

	target	text
2511	ham	Er yep sure. Props?
734	ham	Leaving to qatar tonite in search of an opport...
5164	spam	Congrats 2 mobile 3G Videophones R yours. call...
4794	ham	Saw Guys and Dolls last night with Patrick Swa...
4587	ham	I wanted to wish you a Happy New Year and I wa...

In [10]:
```
# Initializing the LabelEncoder

encoder = LabelEncoder()
```

In [11]:
```
# Replacing ham with 0 and spam with 1 in column 'target'

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

In [12]:
```
# Printing First 5 Rows in Dataframe

df.head()
```

Out[12]:

	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 [13]:
```
# Checking For Missing Values in Dataframe

df.isnull().sum()
```

Out[13]:
```
target    0
text      0
dtype: int64
```

In [14]:
```
# Checking For Total Number of Duplicate Values in Dataframe

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

Out[14]:
```
403
```

In [15]:
```
# Removing all the Other Duplicate Values and Keeping Only First One

df = df.drop_duplicates(keep='first')
```

In [16]:
```
# Checking For Total Number of Duplicate Values Again

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

Out[16]:
```
0
```

In [17]:
```
# Checking the Shape of DataFrame Again

df.shape
```

Out[17]:
```
(5169, 2)
```

2.EDA

In [18]:
```
# Printing First 5 Rows in Dataframe

df.head()
```

Out[18]:

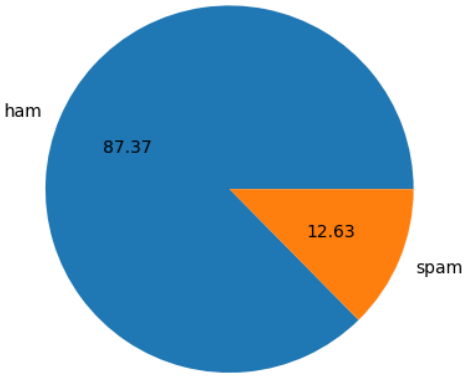
	target	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

In [19]:
```
# Counts the Total Number of ham(0) and spam(1) in Dataframe

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

Out[19]:
```
target
0    4516
1     653
Name: count, dtype: int64
```

- Piechart

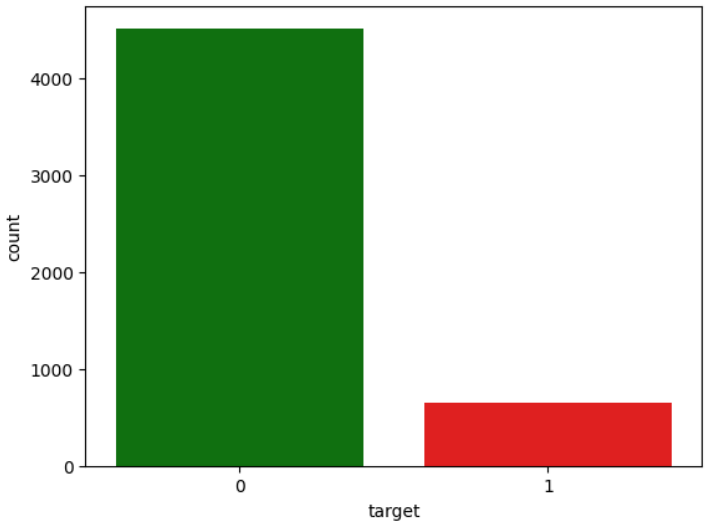
In [20]:
```
# Print the piechart representing the percentage composition of ham and spam in the Dataframe

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



- Bargraph

In [21]:
```
# Prints the BarGraph column 'target' vs their count in the dataframe

sns.countplot(x='target',data=df,palette=['g','r'])
```

Out[21]:
```
<Axes: xlabel='target', ylabel='count'>
```



In [22]:
```
# Counts the total number of characters in column 'text' for each row and saves the count in new column 'num_characters' in the dataframe

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

In [23]:
```
df.head()
```

Out[23]:

	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 [24]:
```
# Counts the total number of words in column 'text' in each row and saves the count in new column 'num_words' in the dataset

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

In [25]:
```
df.head()
```

Out[25]:

	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 [26]:
```
# Counts the total number of sentences in column 'text' in each row and saves the count in new column 'num_sentences' in the dataframe
```

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

In [27]:
```
df.head()
```

Out[27]:

	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 [28]:
```
# Generate descriptive statistics for 'num_characters','num_words','num_sentences' in the Dataframe

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

Out[28]:

	num_characters	num_words	num_sentences
count	5169.000000	5169.000000	5169.000000
mean	78.977945	18.455794	1.965564
std	58.236293	13.324758	1.448541
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 [29]:
```
# Generate descriptive statistics for 'num_characters','num_words','num_sentences' of ham in the Dataframe

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

Out[29]:

	num_characters	num_words	num_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.459256	17.123782	1.820195
std	56.358207	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

In [30]:
```
# Generate descriptive statistics for 'num_characters','num_words','num_sentences' of spam in the Dataframe

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

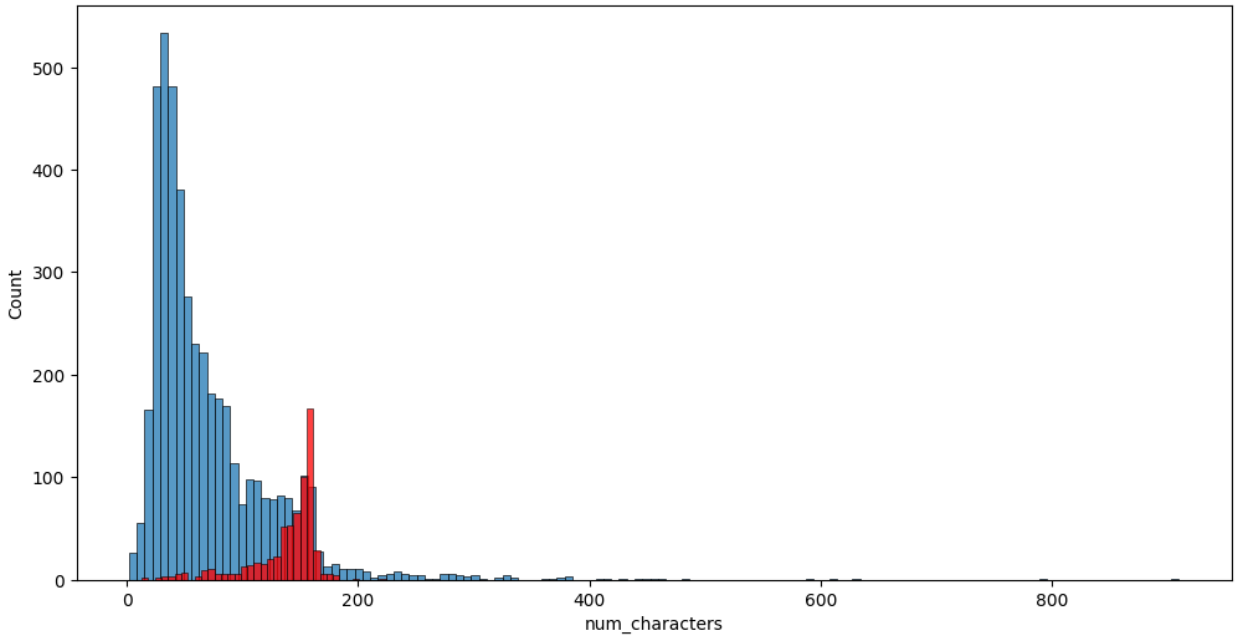
Out[30]:

	num_characters	num_words	num_sentences
count	653.000000	653.000000	653.000000
mean	137.891271	27.667688	2.970904
std	30.137753	7.008418	1.488425
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	9.000000

- Histograms

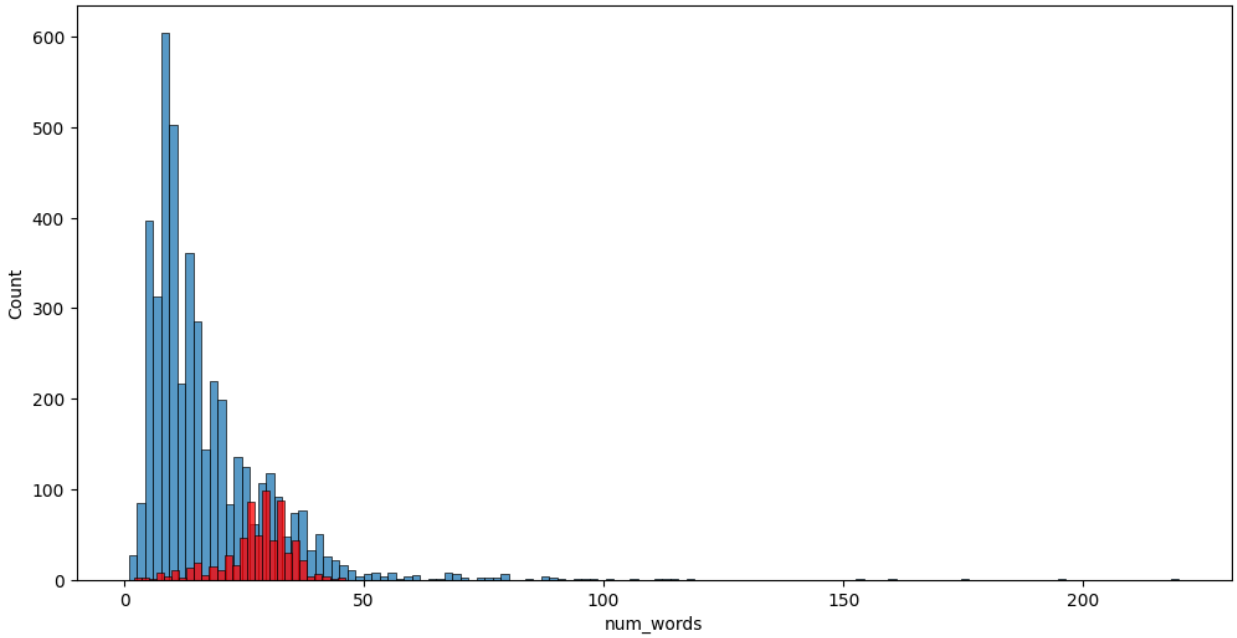
In [31]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
```

Out[31]:
```
<Axes: xlabel='num_characters', ylabel='Count'>
```



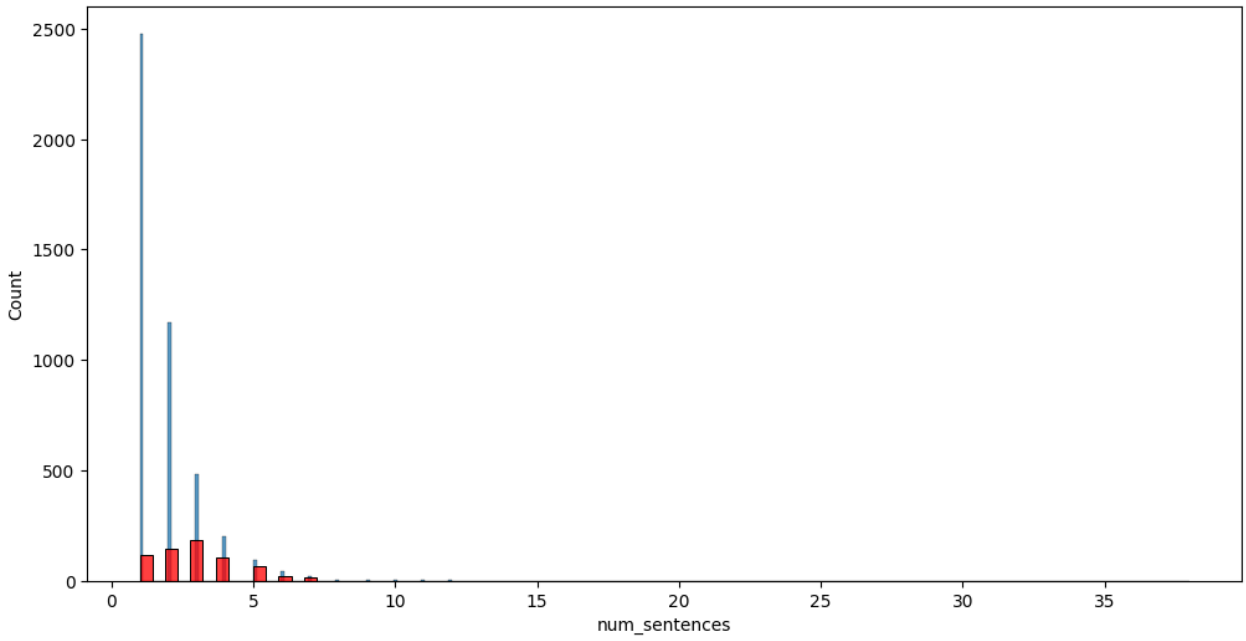
In [32]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
```

Out[32]: <Axes: xlabel='num_words', ylabel='Count'>



In [33]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_sentences'])
sns.histplot(df[df['target'] == 1]['num_sentences'],color='red')
```

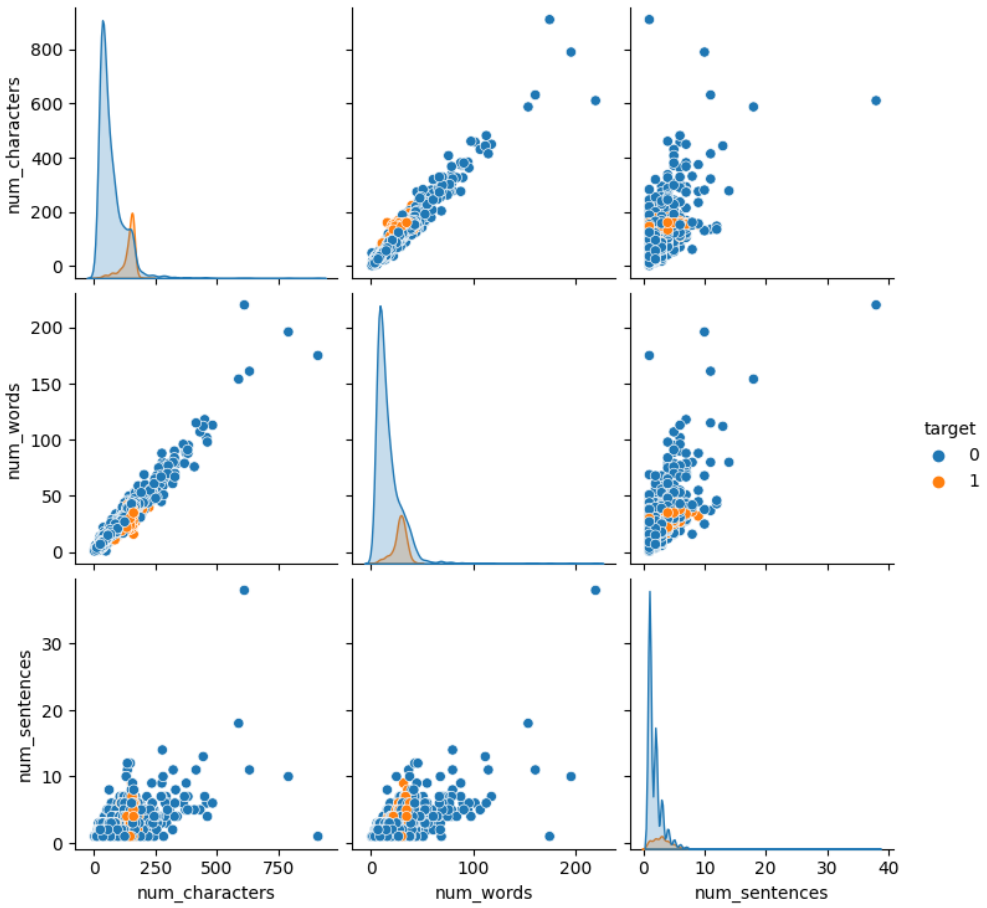
Out[33]: <Axes: xlabel='num_sentences', ylabel='Count'>



- Pairplot

In [34]:
```
# A pairplot plot a pairwise relationships in a dataset

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

Out[34]:
```
<seaborn.axisgrid.PairGrid at 0x27bb8a8f190>
```



- Heatmap

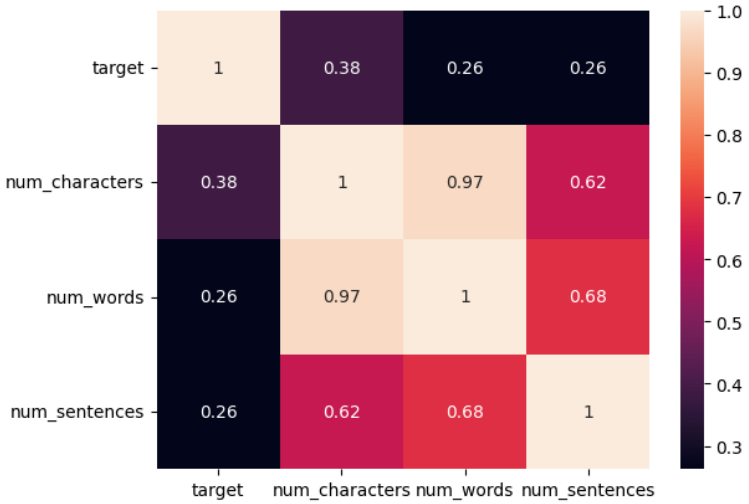
In [35]:
```
df_heatmap=df.select_dtypes(exclude='object')
df_heatmap.corr()
```

Out[35]:

	target	num_characters	num_words	num_sentences
target	1.000000	0.384717	0.262912	0.263939
num_characters	0.384717	1.000000	0.965760	0.624139
num_words	0.262912	0.965760	1.000000	0.679971
num_sentences	0.263939	0.624139	0.679971	1.000000

In [36]:
```
# Generate the heatmap of the for the columns in Dataset

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

Out[36]:
```
<Axes: >
```



---

3. Dataset Preprocessing

- Lower case
- Tokenization
- Removing special characters
- Removing stopwords and punctuation
- Stemming

In [37]:
```
# Initializing PorterStemmer Class

ps = PorterStemmer()
```

In [38]:
```
"""

This Function performs following operations on sentence in text column in each row in Dataset:
1. Converts all uppercase characters to lowercase characters
2. Tokenize the sentence
3. Removes all special characters from sentence
4. Removes stopwords and punctuations from sentence
5. Perform stemming on sentence

"""

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 [39]:
```
# Saves new generated sentence in new coulumn transformed_text after performing trandform_text function on sentence in text column of each row in Dataset

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

In [40]:
```
df.head()
```

Out[40]:

	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

WordCloud

Wordcloud is basically a visualization technique to represent the frequency of words in a text where the size of the word represents its frequency

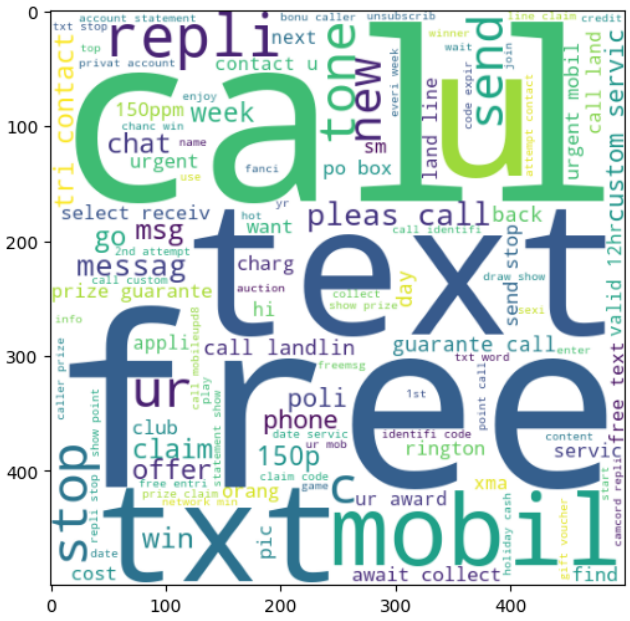
In [41]:
```
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
```

- spam WordCloud

In [42]:
```
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep=" "))
```

In [43]:
```
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

Out[43]:
```
<matplotlib.image.AxesImage at 0x27bbe2b9e90>
```

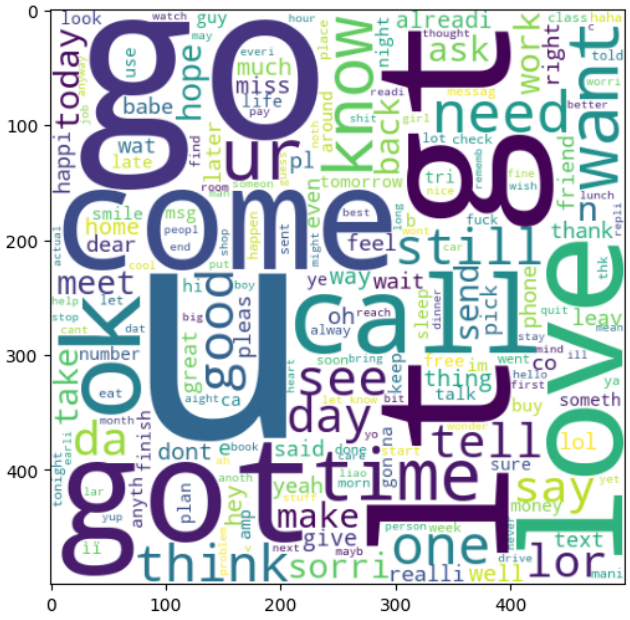


- ham WordCloud

In [44]:
```
ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(sep=" "))
```

In [45]:
```
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
```

Out[45]: <matplotlib.image.AxesImage at 0x27bbe5aa890>



Creating Corpus of ham & spam

A corpus can be defined as a collection of text documents

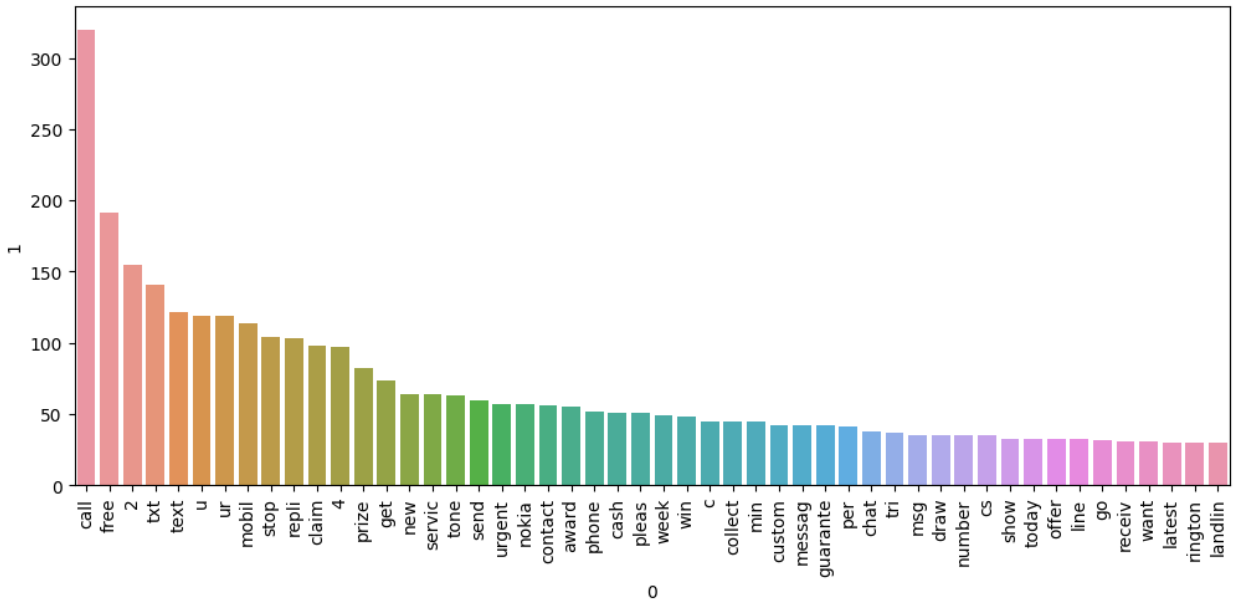
- Creating spam Corpus

In [46]:
```
# Spliting spam sentences in column 'transformed_text' into Words

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

In [47]:
```
# Total number of words in spam corpus

len(spam_corpus)
```

Out[47]: 9939

In [48]:
```
# Plots Bargraph of Top 50 spam Words vs their frequency in the dataset

a=pd.DataFrame(Counter(spam_corpus).most_common(50))[0]
b=pd.DataFrame(Counter(spam_corpus).most_common(50))[1]
plt.figure(figsize=(12,5))
sns.barplot(x= a,y=b)
plt.xticks(rotation=90)

plt.show()
```



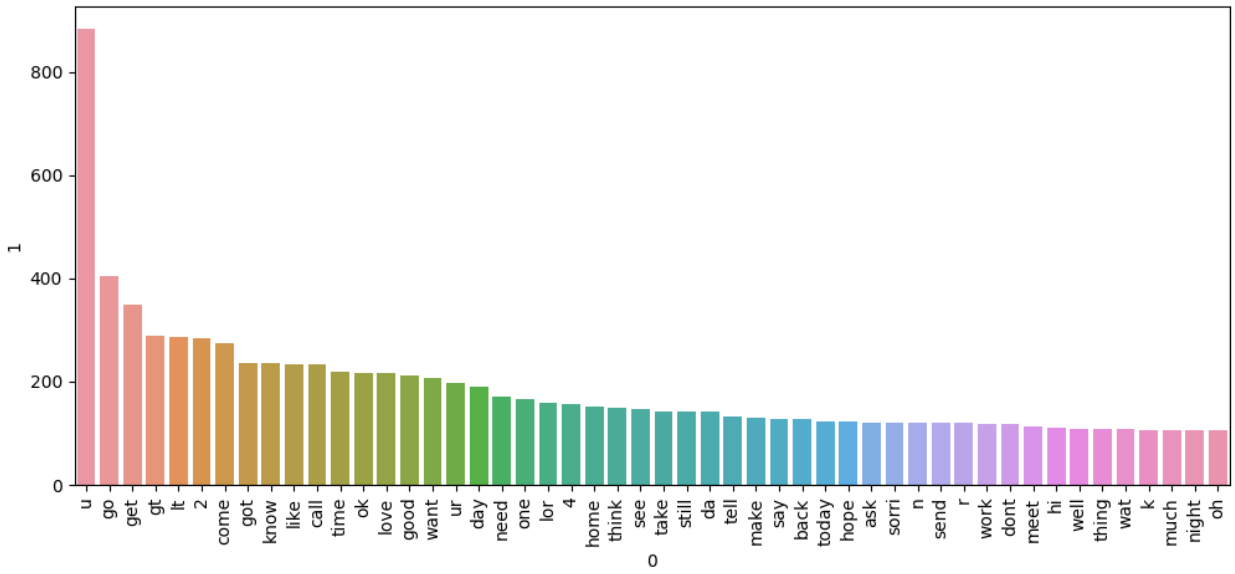
- Creating ham Corpus

In [49]:
```
# Spliting ham sentences in column 'transformed_text' into Words

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

In [50]:
```
# Total number of words in ham corpus

len(ham_corpus)
```

Out[50]: 35404

In [51]:
```
# Plots Bargraph of Top 50 ham Words vs their count in the dataset

a=pd.DataFrame(Counter(ham_corpus).most_common(50))[0]
b=pd.DataFrame(Counter(ham_corpus).most_common(50))[1]
plt.figure(figsize=(12,5))
sns.barplot(x= a,y=b)
plt.xticks(rotation=90)

plt.show()
```



---

4. Models Training Using Various Algorithms

- Vectorization

In [52]:
```
# Initializing TfidfVectorizer

tfidf = TfidfVectorizer(max_features=3000)
```

In [53]:
```
# Independent Feature
```

```
X = tfidf.fit_transform(df["transformed_text"]).toarray()
```

In [54]:
```
# Dependent Feature

y = df["target"].values
```

In [55]:
```
# Performing Train Test Split

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

- Models Training Begins

In [56]:
```
# Models that are going to be trained

models={
    "Gaussian NB" : GaussianNB(),
    "Multinomial NB" : MultinomialNB(),
    "Bernoulli NB" : BernoulliNB(),
    "Logistic Regression" : LogisticRegression(),
    "SVC" : SVC(),
    "Decision Tree" : DecisionTreeClassifier(),
    "KNN" : KNeighborsClassifier(),
    "Bagging CLF" : BaggingClassifier(),
    "Random Forest" : RandomForestClassifier(),
    "ETC" : ExtraTreesClassifier(),
    "Ada Boost" : AdaBoostClassifier(),
    "Gradient Boost" : GradientBoostingClassifier(),
    "XGB" : XGBClassifier(),
    "XGBRF" : XGBRFClassifier()
}
```

In [57]:
```
# Creating a function train each model and calculate & return accuracy and precision

def train_model (model, X_train, y_train, X_test, y_test):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)

    return accuracy, precision
```

In [58]:
```
# A for loop Calls "train_model" for each model and stores accuracy and precision

accuracy_s=[]
precision_s=[]

for name, model in models.items():
    accuracy, precision = train_model(model, X_train, y_train, X_test, y_test)

    accuracy_s.append(accuracy)
    precision_s.append(precision)
```

In [59]:
```
# As Precision matter over Accuracy in this Data, Sorting in DESC order of Precision. All Scores of Models

scores_df = pd.DataFrame({"Algorithm": models.keys(),
                          "Accuracy": accuracy_s,
                         "Precision": precision_s}).sort_values(by="Precision", ascending=False)
```

---

5. Observations (Comparing Accuracy And Precision)

- Observation Table

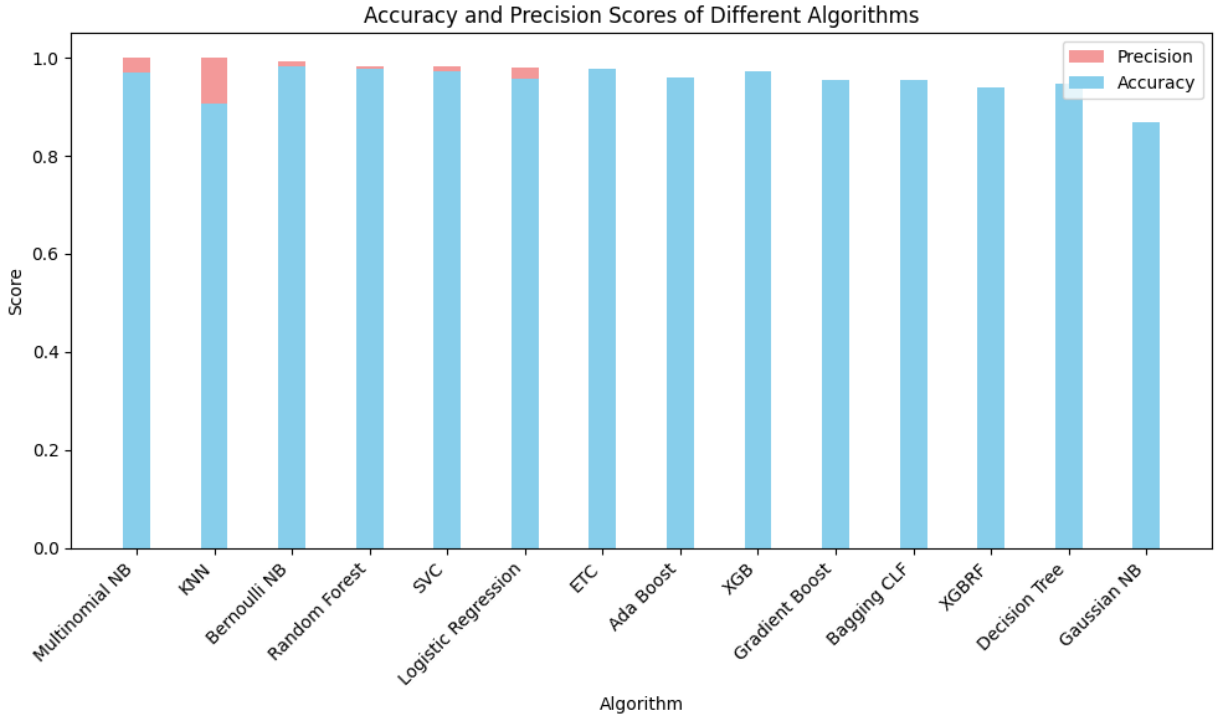
In [60]:
```
# Printing the Accuracy and Scores of all the Models trained with following Algorithms

scores_df
```

Out[60]:

	Algorithm	Accuracy	Precision
1	Multinomial NB	0.970986	1.000000
6	KNN	0.905222	1.000000
2	Bernoulli NB	0.983559	0.991870
8	Random Forest	0.976789	0.983051
4	SVC	0.972921	0.982456
3	Logistic Regression	0.956480	0.979381
9	ETC	0.977756	0.975207
10	Ada Boost	0.960348	0.936937
12	XGB	0.971954	0.936000
11	Gradient Boost	0.955513	0.925926
7	Bagging CLF	0.953578	0.868852
13	XGBRF	0.939072	0.857143
5	Decision Tree	0.947776	0.808824
0	Gaussian NB	0.869439	0.506849

- Observations BarPlot

In [61]:
```
plt.figure(figsize=(10, 6))
bar_width = 0.35

plt.bar(scores_df["Algorithm"], scores_df["Precision"], width=bar_width, label="Precision", color='lightcoral', alpha=0.8)
plt.bar(scores_df["Algorithm"], scores_df["Accuracy"], width=bar_width, label="Accuracy", color='skyblue')
plt.xlabel("Algorithm")
plt.ylabel("Score")
plt.title("Accuracy and Precision Scores of Different Algorithms")
plt.xticks(rotation=45, ha='right')
plt.legend()
plt.tight_layout()
plt.show()
```



---

6. Choosing Best Suitable Algorithm For Model

Choosing Best Suitable Algorithm For Model After Comparing Accuracy and Precision of All Algorithms After Training

In [62]:
```
# Hence Multinomial Naïve Bayes Classifier give excellent precision and accuracy scores.
# According to us MNB is best sutaible for Model

mnb=MultinomialNB()
mnb.fit(X_train, y_train)
```

Out[62]:
```
MultinomialNB()
```

---

7. Generating Pickle Files

In [63]:
```
# Generating 2 Pickle Files : 'vectorizer.pkl' & 'model.pkl'

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