# SMS_SPAM_DETECTION

August 13, 2023

## 0.1 CODSOFT INTERNSHIP

varad patil

### 0.1.1 Adding the dataset from kaggle

[1]:
```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages
(1.5.16)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-
packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-
packages (from kaggle) (2023.7.22)
Requirement already satisfied: python-dateutil in
/usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from kaggle) (4.66.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-
packages (from kaggle) (8.0.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-
packages (from kaggle) (2.0.4)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages
(from kaggle) (6.0.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-
packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in
/usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->kaggle) (3.4)
```

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

### 0.1.2 make a temporary directory

[3]:
```
import os
os.environ['KAGGLE_CONFIG_DIR'] = '/content/drive/MyDrive/Colab Notebooks/
↪kaggle_dataset'
```

[4]:
```
!pwd
```

```
/content
```

[5]:
```
%cd drive/MyDrive/Colab Notebooks/kaggle_dataset
```

```
/content/drive/MyDrive/Colab Notebooks/kaggle_dataset
```

[6]:
```
!pwd
```

```
/content/drive/MyDrive/Colab Notebooks/kaggle_dataset
```

[7]:
```
!kaggle datasets download -d uciml/sms-spam-collection-dataset
```

```
sms-spam-collection-dataset.zip: Skipping, found more recently modified local
copy (use --force to force download)
```

[8]:
```
!unzip sms-spam-collection-dataset.zip
```

```
Archive:  sms-spam-collection-dataset.zip
replace spam.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: spam.csv
```

### 0.1.3 Importing Library

[9]:
```
from logging import warning
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[10]:
```
df = pd.read_csv('spam.csv', encoding='latin-1')
```

[11]:
```
df.head()
```

[11]:

| | v1 | v2 | Unnamed: 2 | \ |
|---|---|---|---|---|
| 0 | ham | Go until jurong point, crazy.. Available only … | NaN | |

```
1   ham                      Ok lar… Joking wif u oni…        NaN
2  spam  Free entry in 2 a wkly comp to win FA Cup fina…        NaN
3   ham  U dun say so early hor… U c already then say…        NaN
4   ham  Nah I don't think he goes to usf, he lives aro…        NaN

  Unnamed: 3 Unnamed: 4
0        NaN        NaN
1        NaN        NaN
2        NaN        NaN
3        NaN        NaN
4        NaN        NaN
```

[12]:
```
df.dropna(axis=1, inplace=True)
```

[13]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['v1'] = le.fit_transform(df['v1'])
```

[14]:
```
df.head()
```

[14]:
```
   v1                                                 v2
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…
```

[15]:
```
print('df:', df.shape)
```

```
df: (5572, 2)
```

[16]:
```
df.rename(columns={"v1":"Category","v2":"Message"}, inplace = True)
```

[17]:
```
df['num_characters'] = df['Message'].apply(len)
```

[18]:
```
from nltk import corpus
import re
import nltk
nltk.download('stopwords')
nltk.download('punkt')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[19]:
```
df['num_words'] = df['Message'].apply(lambda x:len(nltk.word_tokenize(x)))
```

[20]:
```
df['num_sentences'] = df['Message'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

[21]:
```
df.head()
```

[21]:

| | Category | Message | 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 |

[22]:
```
df.duplicated().sum()
```

[22]:
```
403
```

[23]:
```
df.drop_duplicates(keep='first', inplace=True)
df.reset_index(drop = True, inplace = True)
```

[24]:
```
df.head(105)
```

[24]:

| | Category | Message | 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… | | | |
| .. | … | … | | | |
| 100 | 0 | Okay name ur price as long as its legal! Wen c… | | | |
| 101 | 0 | I'm still looking for a car to buy. And have n… | | | |
| 102 | 0 | wow. You're right! I didn't mean to do that. I… | | | |
| 103 | 0 | Umma my life and vava umma love you lot dear | | | |
| 104 | 0 | Thanks a lot for your wishes on my birthday. T… | | | |

```
4                61          15               1
..              ...         ...             ...
100              81          23               3
101              76          19               2
102             183          44               5
103              44          10               1
104              95          19               2

[105 rows x 5 columns]
```

[25]:
```
df.duplicated().sum()
```

[25]:
```
0
```

[26]:
```
df.isna().sum()
```

[26]:
```
Category          0
Message           0
num_characters    0
num_words         0
num_sentences     0
dtype: int64
```

[27]:
```
df.isnull().sum()
```

[27]:
```
Category          0
Message           0
num_characters    0
num_words         0
num_sentences     0
dtype: int64
```

[28]:
```
df.info()
```

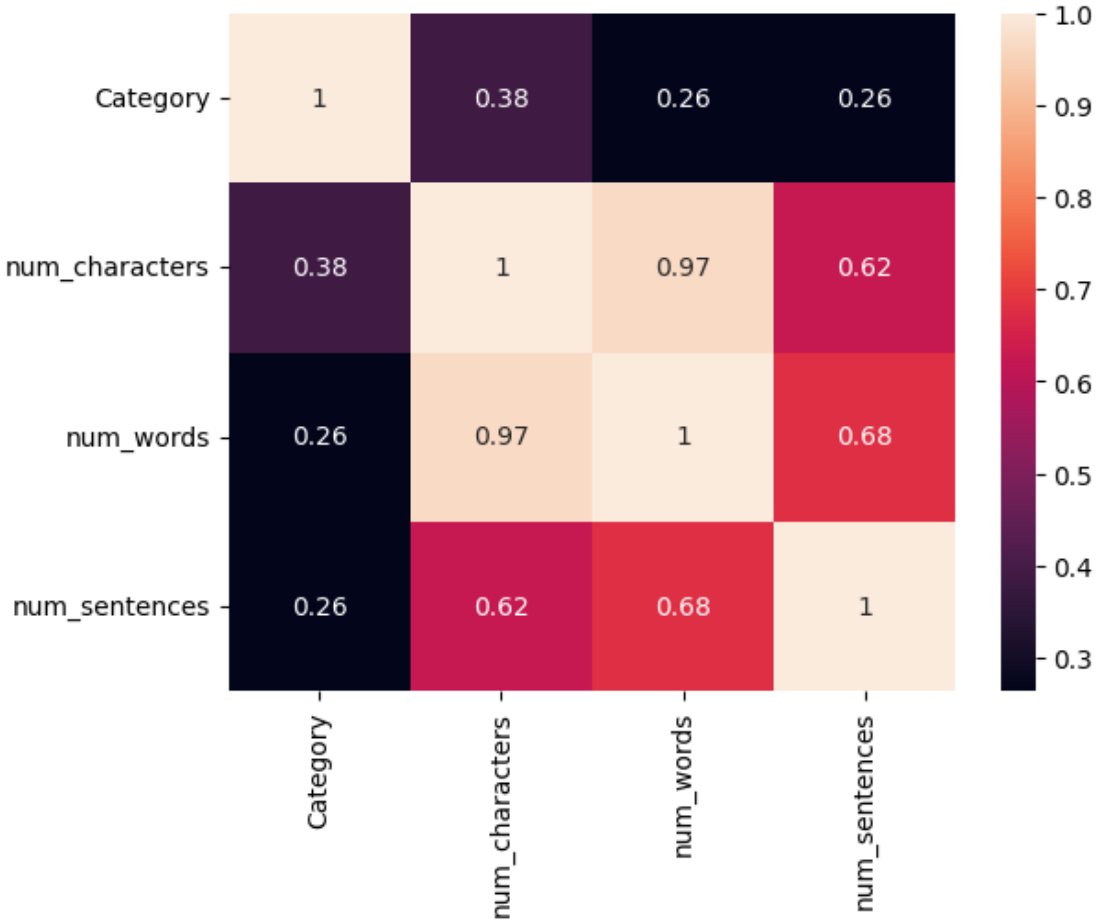
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5169 entries, 0 to 5168
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Category        5169 non-null   int64
 1   Message         5169 non-null   object
 2   num_characters  5169 non-null   int64
 3   num_words       5169 non-null   int64
 4   num_sentences   5169 non-null   int64
dtypes: int64(4), object(1)
memory usage: 202.0+ KB
```

[29]:
```
df.describe()
```

[29]:

| | Category | num_characters | num_words | num_sentences |
|---|---|---|---|---|
| count | 5169.000000 | 5169.000000 | 5169.000000 | 5169.000000 |
| mean | 0.126330 | 78.977945 | 18.455794 | 1.965564 |
| std | 0.332253 | 58.236293 | 13.324758 | 1.448541 |
| min | 0.000000 | 2.000000 | 1.000000 | 1.000000 |
| 25% | 0.000000 | 36.000000 | 9.000000 | 1.000000 |
| 50% | 0.000000 | 60.000000 | 15.000000 | 1.000000 |
| 75% | 0.000000 | 117.000000 | 26.000000 | 2.000000 |
| max | 1.000000 | 910.000000 | 220.000000 | 38.000000 |

[30]:
```
sns.heatmap(df.corr(),annot=True)
plt.xticks(rotation=90)
plt.show()
```



### 0.1.4 Cleaning the data

[31]:
```
all_stopwords = stopwords.words('english')
print(all_stopwords)
```

```
['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',
'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',
'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about',
'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above',
'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under',
'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why',
'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some',
'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn',
"couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn',
"hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't",
'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn',
"shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',
"wouldn't"]
```

[31]:
```
```

[32]:
```
corpus = []
for i in range(0, df.shape[0]):
  review = re.sub('[^a-zA-z]', ' ', df['Message'][i])
  review = review.lower()
  #print(review)
  review = review.split()
  #print(review)
  ps = PorterStemmer()
  review = [ps.stem(word) for word in review if not word in all_stopwords]
  #print(review)
  review = ' '.join(review)
  #print(review)
  corpus.append(review)
```

[33]:
```
corpus[0:5]
```

[33]:
```
['go jurong point crazi avail bugi n great world la e buffet cine got amor wat',
 'ok lar joke wif u oni',
 'free entri wkli comp win fa cup final tkt st may text fa receiv entri question
```

```
 std txt rate c appli',
  'u dun say earli hor u c alreadi say',
  'nah think goe usf live around though']
```

[34]:
```
df['transformed'] = corpus
```

[35]:
```
df.head()
```

[35]:

| | Category | Message | \ |
|---|---|---|---|
| 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… | |

| | num_characters | num_words | num_sentences | \ |
|---|---|---|---|---|
| 0 | 111 | 24 | 2 | |
| 1 | 29 | 8 | 2 | |
| 2 | 155 | 37 | 2 | |
| 3 | 49 | 13 | 1 | |
| 4 | 61 | 15 | 1 | |

| | transformed |
|---|---|
| 0 | go jurong point crazi avail bugi n great world… |
| 1 | ok lar joke wif u oni |
| 2 | free entri wkli comp win fa cup final tkt st m… |
| 3 | u dun say earli hor u c alreadi say |
| 4 | nah think goe usf live around though |

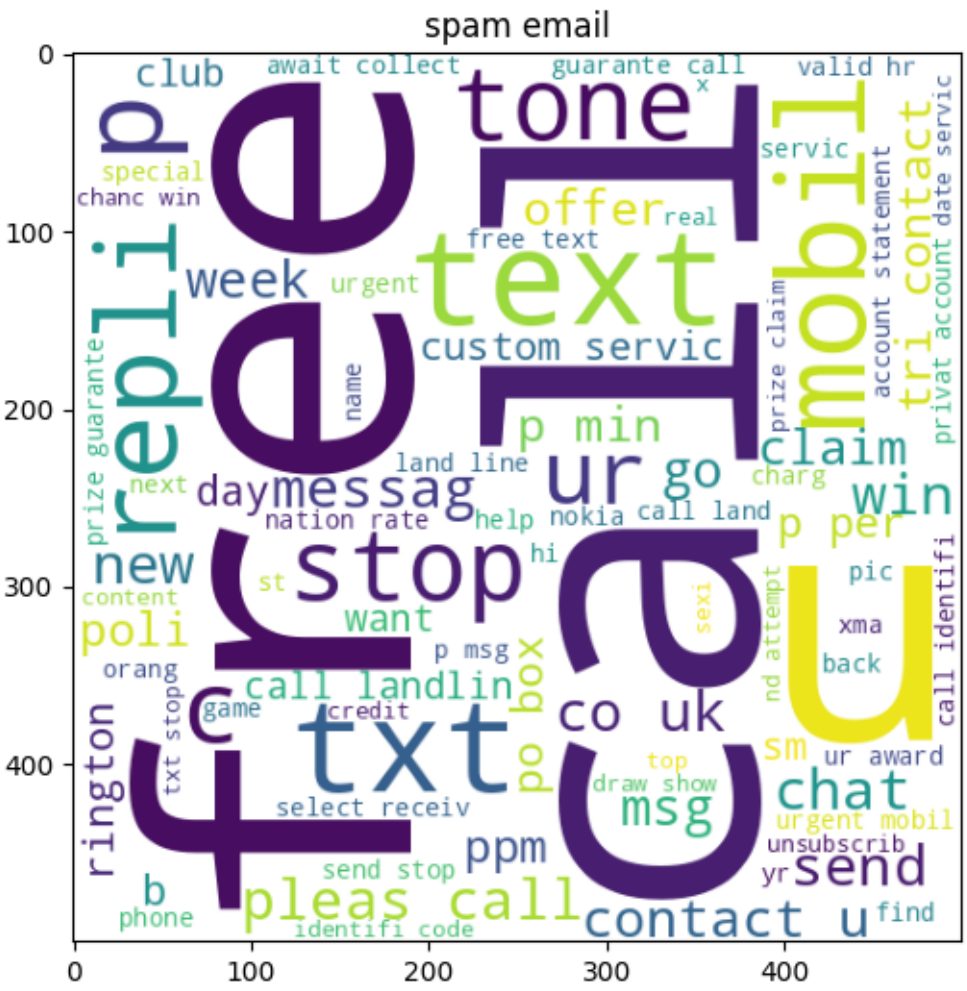
### 0.1.5 Data Visulaization

[36]:
```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
```

[37]:
```
spam_wc = wc.generate(df[df['Category'] == 1]['transformed'].str.cat(sep= ' '))
```

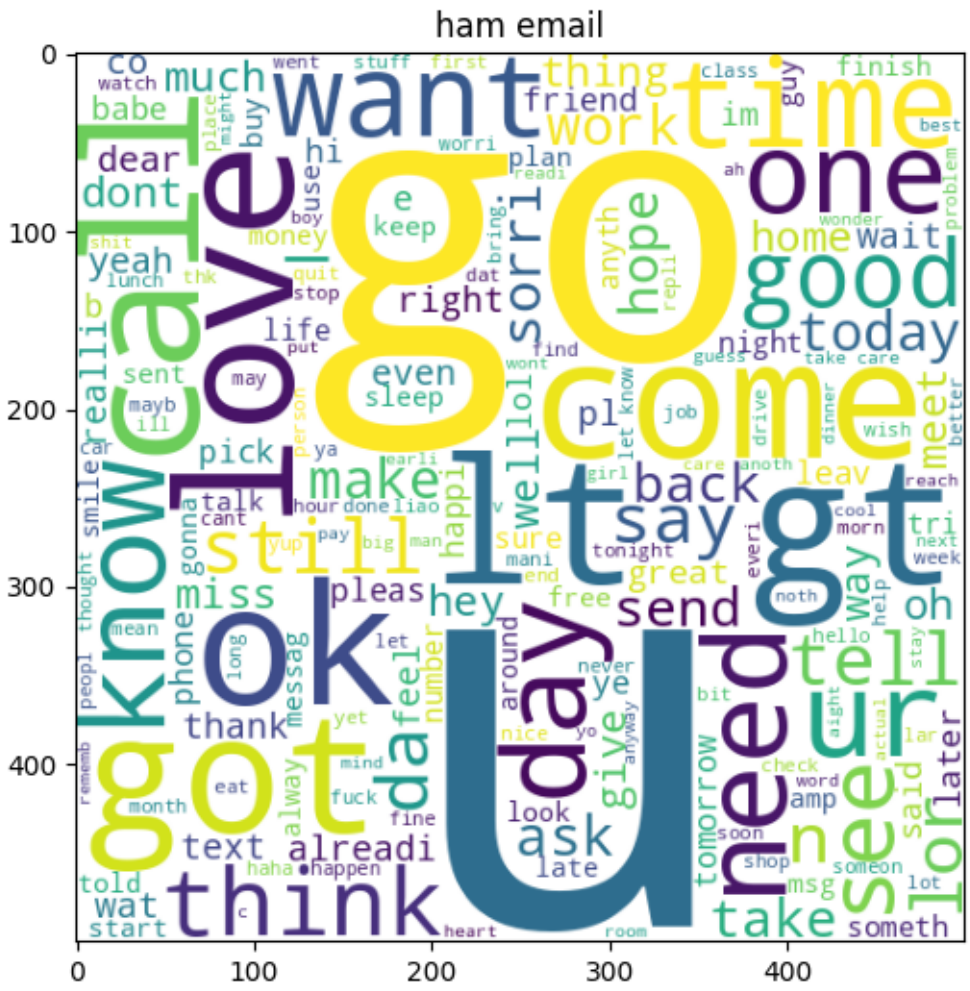
[38]:
```
plt.figure(figsize=(15,6))
plt.title("spam email")
plt.imshow(spam_wc)
```

[38]:
```
<matplotlib.image.AxesImage at 0x7fd931034670>
```

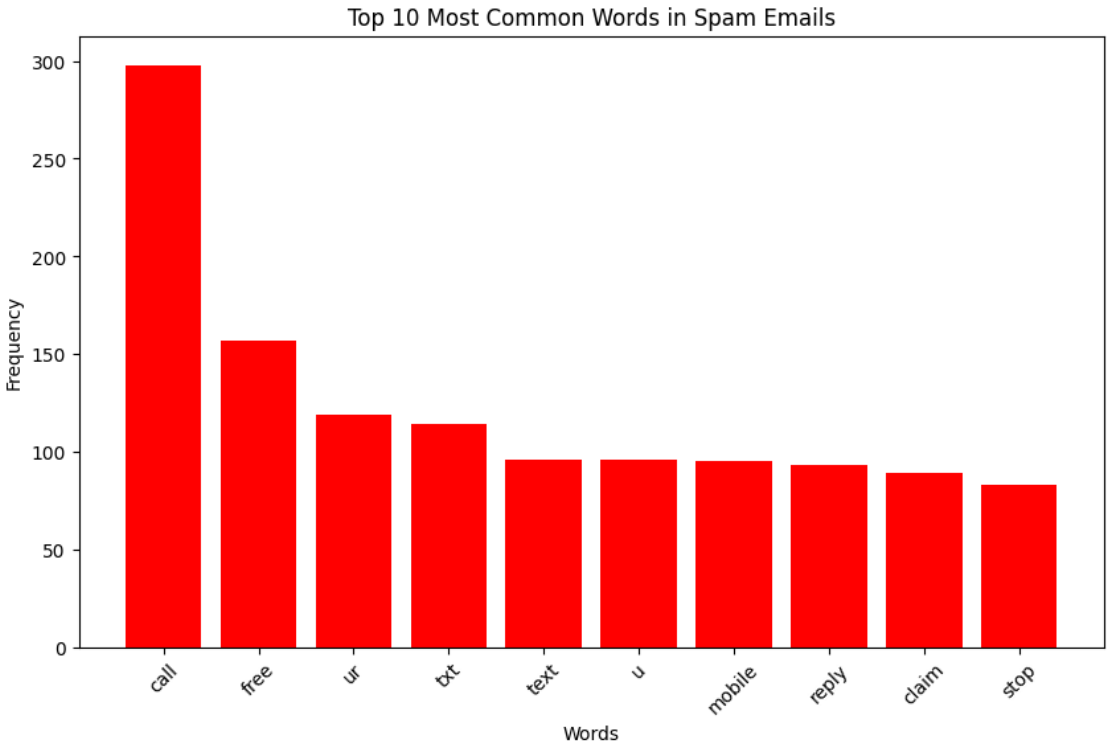



[39]:
```
ham_wc = wc.generate(df[df['Category'] == 0]['transformed'].str.cat(sep= ' '))
plt.figure(figsize=(15,6))
plt.title("ham email")
plt.imshow(ham_wc)
```

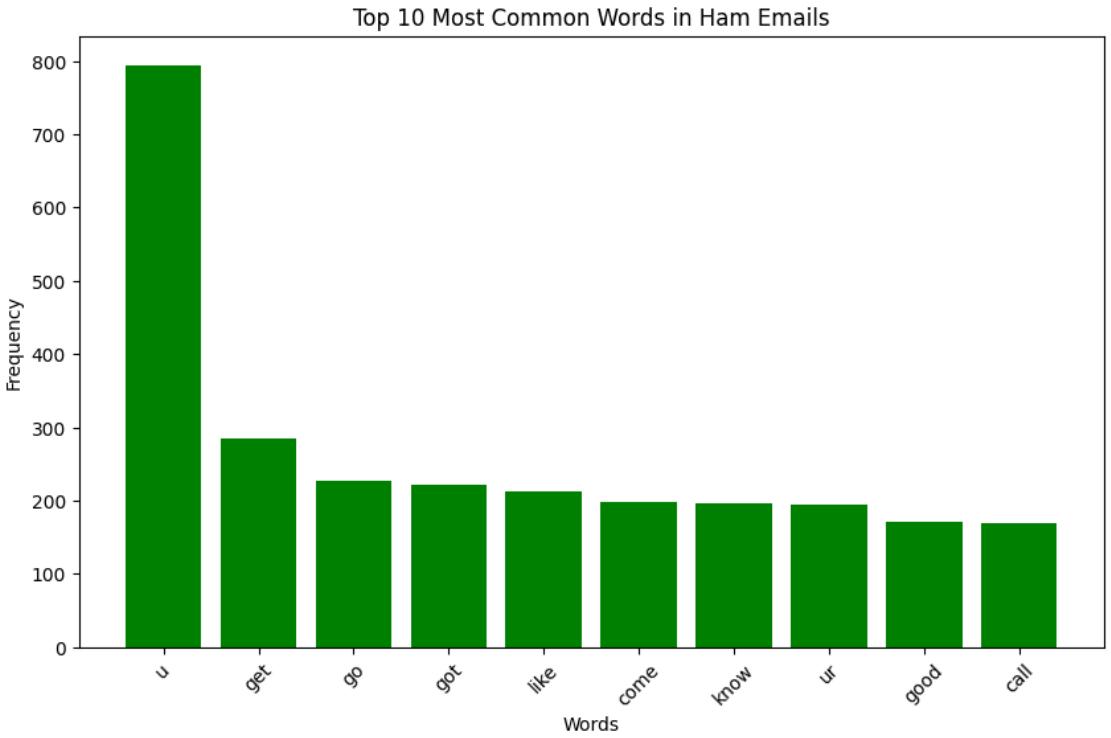
[39]: <matplotlib.image.AxesImage at 0x7fd93093d000>



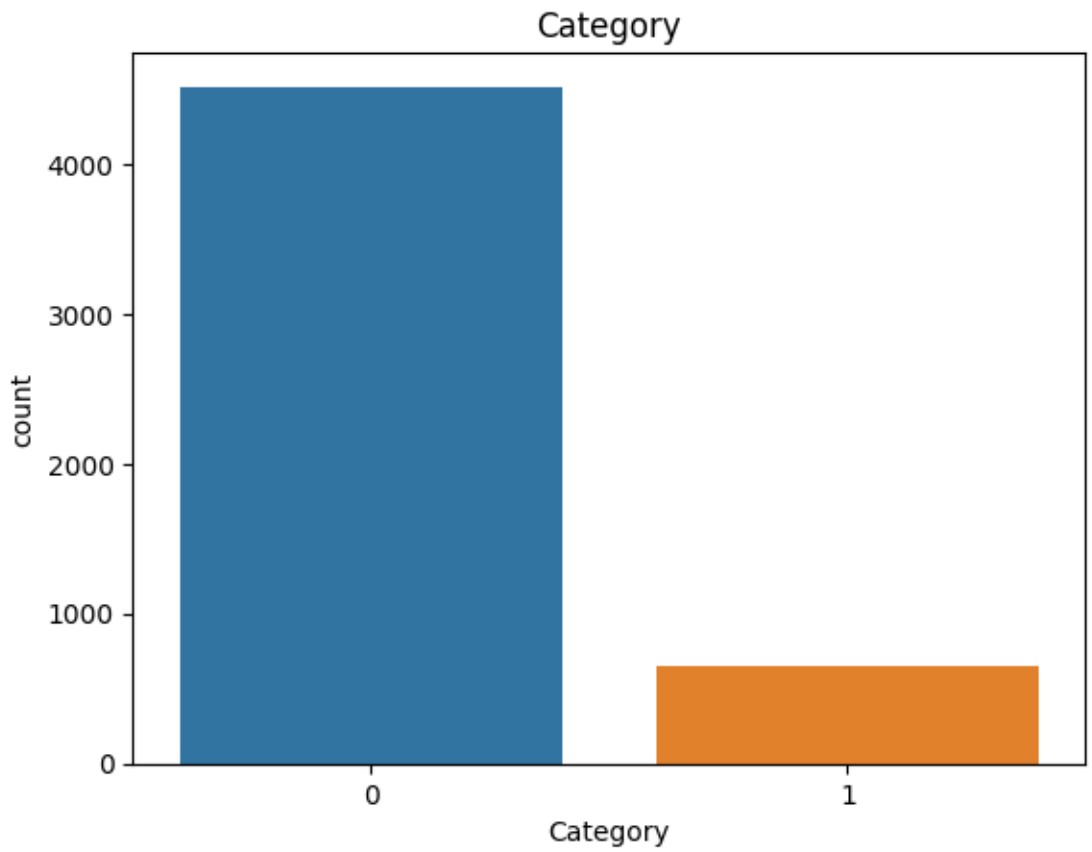

[40]:
```
from collections import Counter

spam_words = " ".join(df[df['Category'] == 1]['Message']).split()
ham_words = " ".join(df[df['Category'] == 0]['Message']).split()

spam_word_freq = Counter([word.lower() for word in spam_words if word.lower()␣
 ↪not in all_stopwords and word.isalpha()])

plt.figure(figsize=(10, 6))
plt.bar(*zip(*spam_word_freq.most_common(10)), color='r')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Top 10 Most Common Words in Spam Emails')
plt.xticks(rotation=45)
plt.show()
```




[41]:
```
ham_word_freq = Counter([word.lower() for word in ham_words if word.lower() not␣
 ↪in all_stopwords and word.isalpha()])

plt.figure(figsize=(10, 6))
plt.bar(*zip(*ham_word_freq.most_common(10)), color='g')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Top 10 Most Common Words in Ham Emails')
plt.xticks(rotation=45)
plt.show()
```




[42]:
```
sns.countplot(x='Category', data=df)
plt.title('Category')
plt.show()
```




### 0.1.6 Data preprocessing

[43]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tf = TfidfVectorizer()
x = tf.fit_transform(corpus).toarray()
y = df.iloc[:, 0:1].values
```

[44]:
```
print('x shape', x.shape)
print('y shape', y.shape)
```

```
x shape (5169, 6251)
y shape (5169, 1)
```

### 0.1.7 Splitting the dataset

[45]:
```
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=42)
```

[46]:
```
print(x_train.shape, x_test.shape)
print(y_train.shape, y_test.shape)
```

```
(4135, 6251) (1034, 6251)
(4135, 1) (1034, 1)
```

[47]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

Training and testing the models

[48]:
```
from sklearn.linear_model import LogisticRegression
log = LogisticRegression(random_state = 42)
log.fit(x_train, y_train)
```

[48]:
```
LogisticRegression(random_state=42)
```

[49]:
```
from sklearn.naive_bayes import GaussianNB, BernoulliNB
GB = GaussianNB()

GB.fit(x_train, y_train)
```

[49]:
```
GaussianNB()
```

[50]:
```
CB = BernoulliNB()
CB.fit(x_train, y_train)
```

[50]:
```
BernoulliNB()
```

[51]:
```
from sklearn.svm import SVC
svc = SVC(kernel = 'linear', random_state = 0)
svc.fit(x_train, y_train)
```

[51]:
```
SVC(kernel='linear', random_state=0)
```

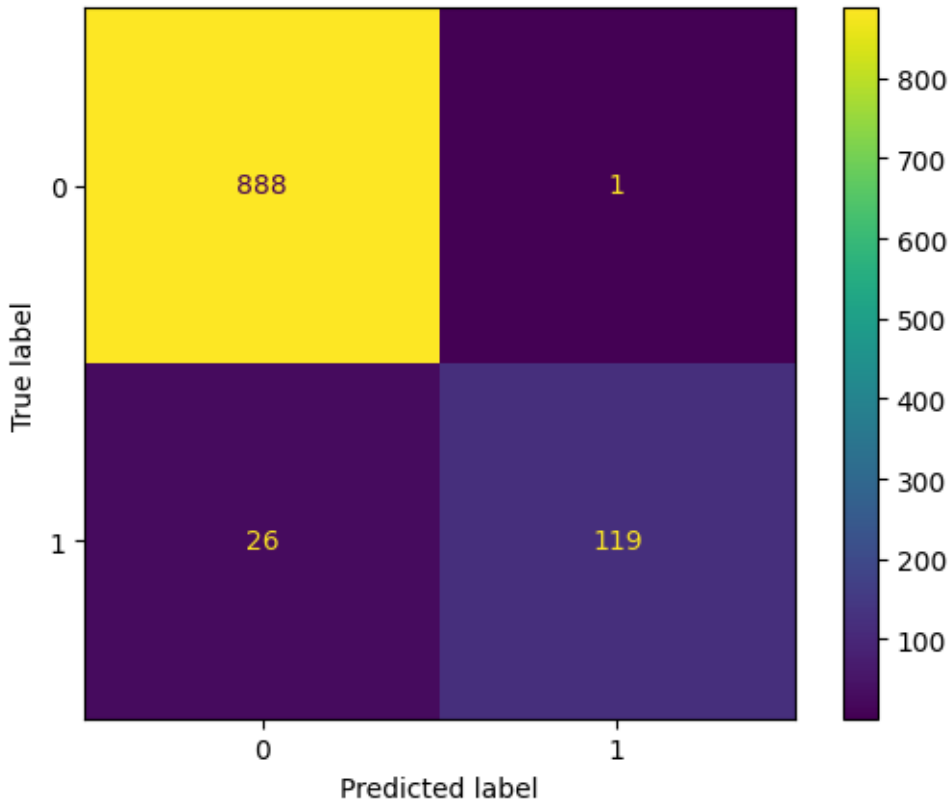
[52]:
```
classifier = [log, svc, GB, CB]
model = ['Logistic', 'Support Vector', 'Naive Bayes GB', 'Naive Bayes CB']
```

## 0.2 Making the Confusion Matrix

[53]:
```
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score, ConfusionMatrixDisplay
for i in range(0, len(classifier)):
  y_pred = classifier[i].predict(x_test)
  cm = confusion_matrix(y_test, y_pred)
```

```
  accuracy = accuracy_score(y_test, y_pred)*100
  print('\nfor ' + str(model[i]) + ':\n')
  disp = ConfusionMatrixDisplay(confusion_matrix=cm)
  plt.rcParams['axes.grid'] = False
  disp.plot()
  print('Accuracy: ',accuracy)
  print(classification_report(y_test, y_pred))
  plt.show()
```

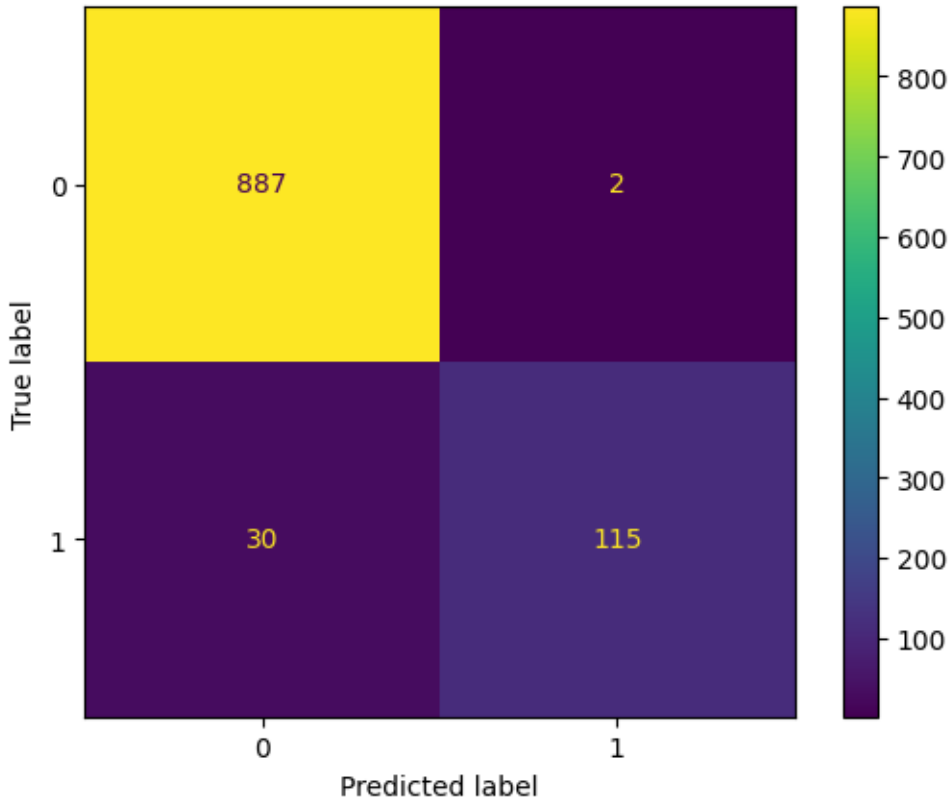
```
for Logistic:

Accuracy:  97.38878143133462
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       889
           1       0.99      0.82      0.90       145

    accuracy                           0.97      1034
   macro avg       0.98      0.91      0.94      1034
weighted avg       0.97      0.97      0.97      1034
```



for Support Vector:

Accuracy:  96.90522243713733

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 1.00 | 0.98 | 889 |
| 1 | 0.98 | 0.79 | 0.88 | 145 |
| accuracy | | | 0.97 | 1034 |
| macro avg | 0.98 | 0.90 | 0.93 | 1034 |
| weighted avg | 0.97 | 0.97 | 0.97 | 1034 |

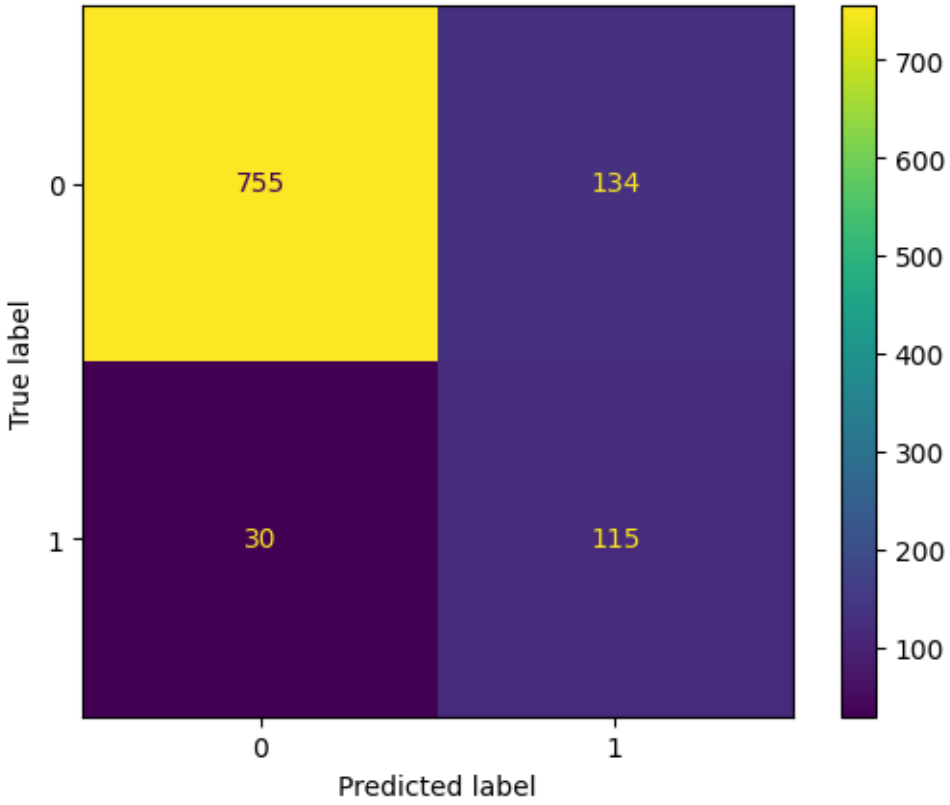



for Naive Bayes GB:

Accuracy:  84.13926499032883

| | precision | recall | f1-score | support |
|---|---|---|---|---|

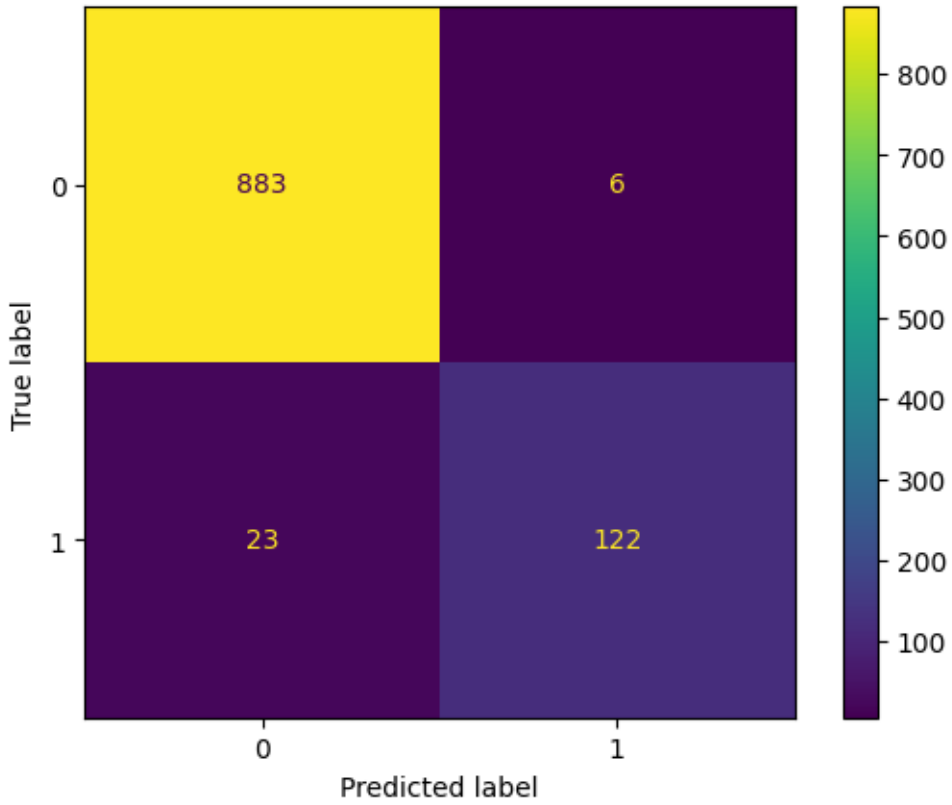
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.96 | 0.85 | 0.90 | 889 |
| 1 | 0.46 | 0.79 | 0.58 | 145 |
| accuracy | | | 0.84 | 1034 |
| macro avg | 0.71 | 0.82 | 0.74 | 1034 |
| weighted avg | 0.89 | 0.84 | 0.86 | 1034 |




for Naive Bayes CB:

Accuracy: 97.1953578336557

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 0.99 | 0.98 | 889 |
| 1 | 0.95 | 0.84 | 0.89 | 145 |
| accuracy | | | 0.97 | 1034 |
| macro avg | 0.96 | 0.92 | 0.94 | 1034 |
| weighted avg | 0.97 | 0.97 | 0.97 | 1034 |




### 0.3 Applying k-Fold Cross Validation

[54]:
```
from sklearn.model_selection import cross_val_score
for i in classifier:
  accuracies  = cross_val_score(estimator=i, X = x_train, y = y_train, cv = 10)
  print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
  print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 96.78 %
Standard Deviation: 0.47 %
Accuracy: 96.42 %
Standard Deviation: 0.64 %
Accuracy: 85.10 %
Standard Deviation: 0.89 %
Accuracy: 97.22 %
Standard Deviation: 0.79 %
```