Python Programming
(Project Report)

Submitted By: Soumya Jain
Sap: 500119436
Roll: R2142230988
Batch: 29
Submitted To: Dr. Manish Kumar Giri

Spam Email Detection Using Data Visualisation

- **Aim**: The primary aim of this project is to leverage data visualization techniques to uncover patterns inherent in spam emails. By analyzing these patterns, we aim to:

  Identify distinguishing characteristics of spam emails: This will involve exploring various email features within a dataset of labeled spam and ham emails. Data visualization will be used to identify trends and relationships between features such as:

  - **Word frequency:** Analyze the frequency of specific words or phrases commonly found in spam emails (e.g., "free", "urgent", "win a million dollars").
  - **Email length:** Visualize the distribution of email lengths across spam and legitimate emails.
  - **Presence of urgency triggers:** Identify the use of words or phrases that create a sense of urgency, often used in phishing attempts.
  - **Special characters and symbols:** Explore the prevalence of uncommon symbols or excessive use of punctuation in spam emails.

Concepts Used:

- **Variables:** Variables are used to store data. Examples are: df, styled_df, values, etc.
- **Functions:** Functions are blocks of reusable code. Examples are: read_csv, head(), info(), isnull(), etc.
- **Importing Library:** importing certain libraries like numpy, pandas, matplotlib.pyplot, seaborn, NLTK, wordcloud, etc.
- **OOPs:** Using Object Oriented Programming to create a parent class i.e WordDistributionPlot.

PROCEDURE:

Importing all libraries:

First of all in this jupyter file we create a cell in which we import all the necessary libraries required.

```python
# Importing necessary libraries
import numpy as np        # For numerical operations
import pandas as pd       # For data manipulation and analysis
import matplotlib.pyplot as plt  # For data visualization
%matplotlib inline

# Importing WordCloud for text visualization
from wordcloud import WordCloud

# Importing NLTK for natural language processing
import nltk
from nltk.corpus import stopwords    # For stopwords


# Downloading NLTK data
nltk.download('stopwords')   # Downloading stopwords data
nltk.download('punkt')     # Downloading tokenizer data
nltk.download('all')
```

Reading the data set:

We read our data set for the project and displaying starting data from the whole data set.

```python
[3]
df = pd.read_csv("D:\Semester - II (Projects)\spam Email Detection\Source code\data.csv")

[4]
styled_df = df.head()
styled_df = styled_df.style.set_table_styles([
    {"selector": "th", "props": [("color", 'black'), ("background-color", "yellow")]}
])
styled_df
```

	Category	Message
0	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
1	ham	Ok lar... Joking wif u oni...
2	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 08452810075over18's
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 around here though

⬆️ Back to the Top ⬆️

Data Cleaning:

Displaying the information of the data set

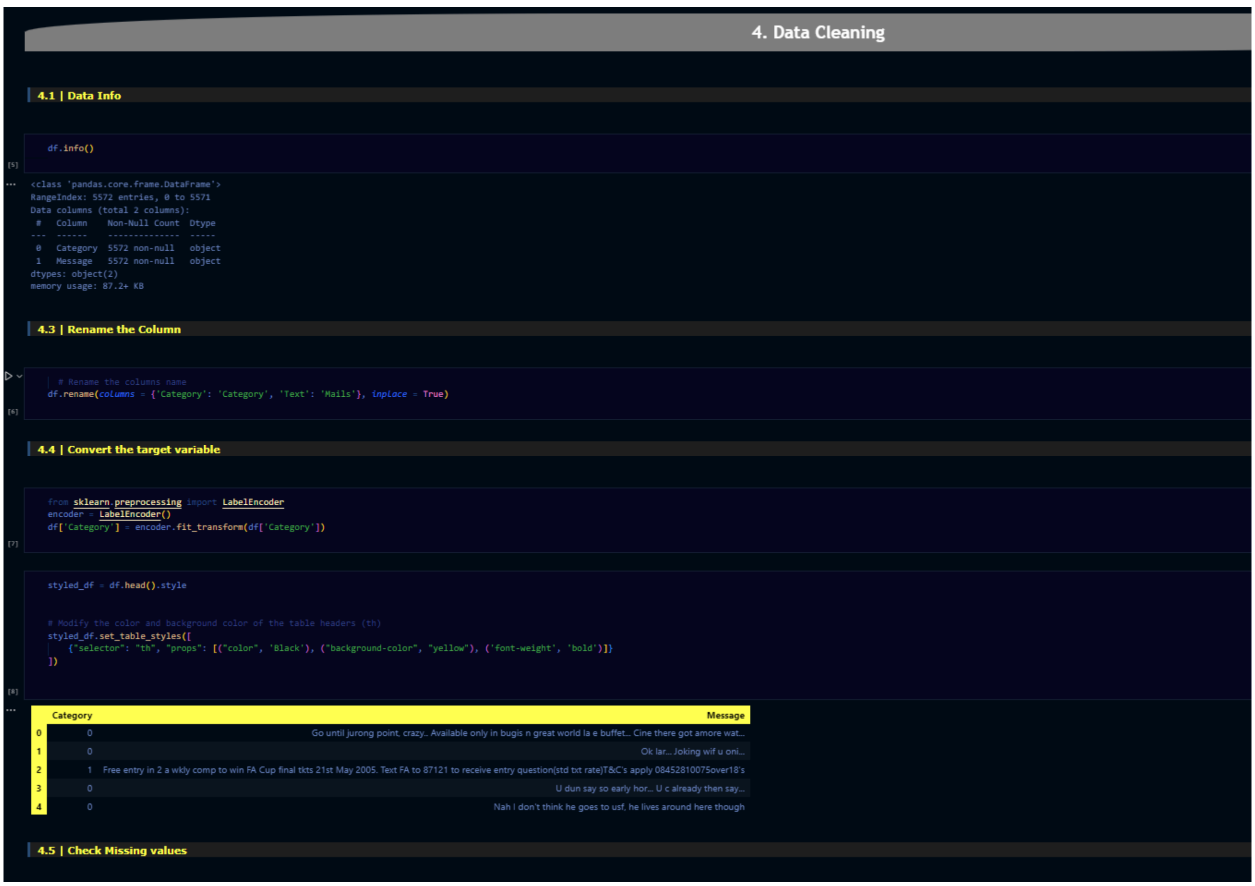
Renaming the column

Checking missing values

Checking duplicate values

Remove duplicate values

Shape of the dataset



4. Data Cleaning

4.1 | Data Info

```
df.info()
```

[5]

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Category  5572 non-null   object
 1   Message   5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

4.3 | Rename the Column

```
# Rename the columns name
df.rename(columns = {'Category': 'Category', 'Text': 'Mails'}, inplace = True)
```

[6]

4.4 | Convert the target variable

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df['Category'] = encoder.fit_transform(df['Category'])
```

[7]

```
styled_df = df.head().style


# Modify the color and background color of the table headers (th)
styled_df.set_table_styles([
    {"selector": "th", "props": [("color", 'Black'), ("background-color", "yellow"), ('font-weight', 'bold')]}
])
```

[8]

	Category	Message
0	0	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
1	0	Ok lar... Joking wif u oni...
2	1	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 08452810075over18's
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 around here though

4.5 | Check Missing values

4.5 | Check Missing values

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

```
Category    0
Message     0
dtype: int64
```

4.6 | Check Duplicate values

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

```
415
```

4.7 | Remove Duplicate values

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

4.8 | Shape of the Dataset

```
df.shape
```

```
(5157, 2)
```

Back to the Top

Data Analysis:

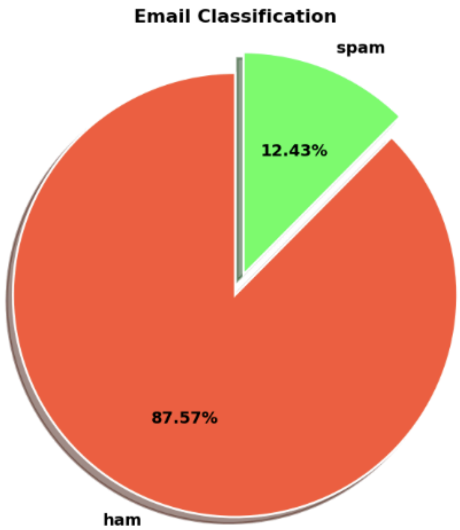
Performing the data analysis on the data set.

5.1 | Percentage of Ham and Spam

```
values = df['Category'].value_counts()
total = values.sum()

percentage_0 = (values[0] /total) * 100
percentage_1 = (values[1]/ total) *100

print('percentage of 0 :' ,percentage_0)
print('percentage of 1 :' ,percentage_1)
```

```
percentage of 0 : 87.5702928058949
percentage of 1 : 12.429707194105099
```

```
import matplotlib.pyplot as plt

# Sample data
# values = [75, 25]  # Example values for 'ham' and 'spam'

# Define custom colors
colors = ['#FF5733', '#33FF57']

# Define the explode parameter to create a gap between slices
explode = (0, 0.1)  # Explode the second slice (spam) by 10%

# Create a figure with a white background
fig, ax = plt.subplots(figsize=(8, 8))
ax.set_facecolor('white')

# Create the pie chart with custom colors, labels, explode parameter, and shadow
wedges, texts, autotexts = ax.pie(
    values, labels=['ham', 'spam'],
    autopct='%0.2f%%',
    startangle=90,
    colors=colors,
    wedgeprops={'linewidth': 2, 'edgecolor': 'white'},
    explode=explode,  # Apply the explode parameter
    shadow=True  # Add shadow
)

# Customize text properties
for text, autotext in zip(texts, autotexts):
    text.set(size=14, weight='bold')
    autotext.set(size=14, weight='bold')

# Add a title
ax.set_title('Email Classification', fontsize=16, fontweight='bold')

# Equal aspect ratio ensures that pie is drawn as a circle
ax.axis('equal')

# Show the pie chart
plt.show()
```



5.2 | Text Length and Structure Analysis

```
df['num_characters'] = df['Message'].apply(len)
df['num_words'] = df['Message'].apply(lambda x: len(nltk.word_tokenize(x)))
df['num_sentence'] = df['Message'].apply(lambda x: len(nltk.sent_tokenize(x)))
```
[13]

```
df[['num_characters', 'num_words', 'num_sentence']].describe()
```
[16]

	num_characters	num_words	num_sentence
count	5157.000000	5157.000000	5157.000000
mean	79.228040	18.544890	1.969750
std	58.451149	13.401415	1.455526
min	2.000000	1.000000	1.000000
25%	36.000000	9.000000	1.000000
50%	61.000000	15.000000	1.000000
75%	118.000000	26.000000	2.000000
max	910.000000	220.000000	38.000000

5.3 | Summary Statistics for Legitimate Messages

```
#ham
df[df['Category'] == 0][['num_characters', 'num_words', 'num_sentence']].describe()
```
[14]

	num_characters	num_words	num_sentence
count	4516.000000	4516.000000	4516.000000
mean	70.869353	17.267715	1.827724
std	56.708301	13.588065	1.394338
min	2.000000	1.000000	1.000000
25%	34.000000	8.000000	1.000000
50%	53.000000	13.000000	1.000000
75%	91.000000	22.000000	2.000000
max	910.000000	220.000000	38.000000

5.4 | Summary Statistics for Spam Messages

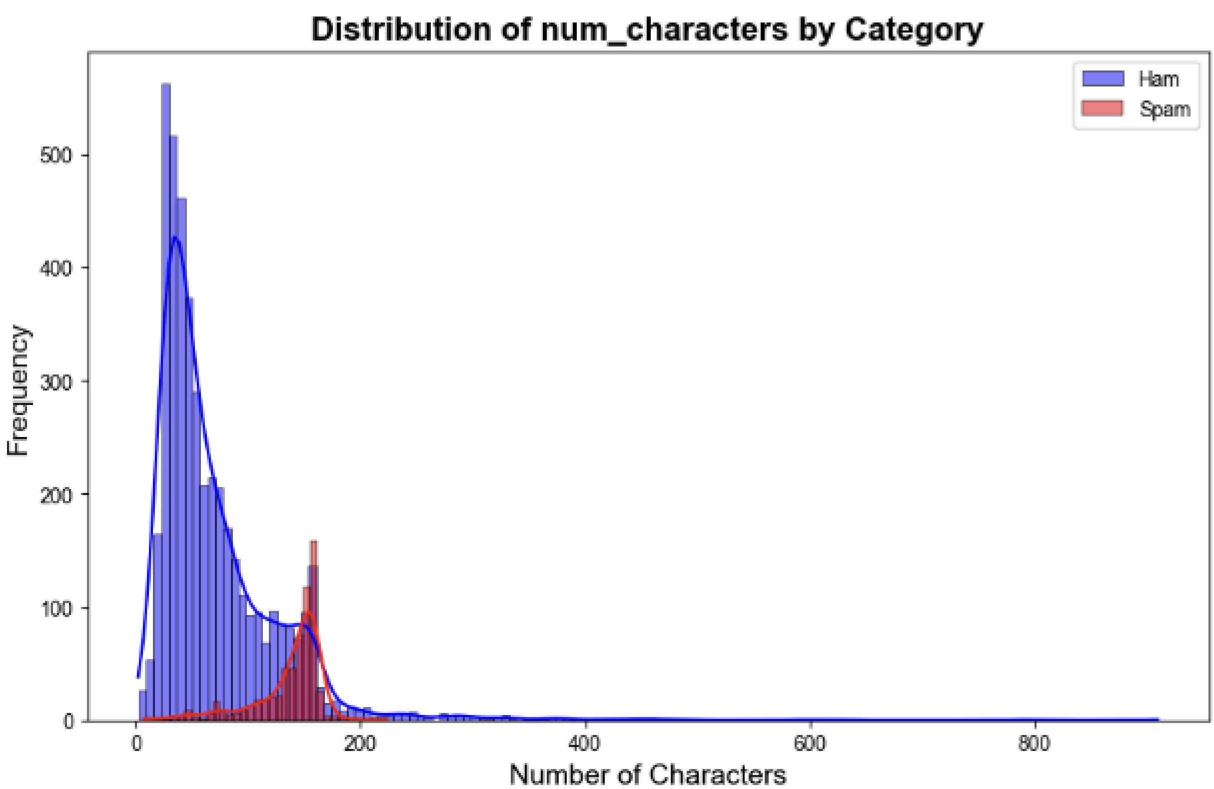
```
#spam
df[df['Category'] == 1][['num_characters', 'num_words', 'num_sentence']].describe()
```

[15]

	num_characters	num_words	num_sentence
count	641.000000	641.000000	641.000000
mean	137.118565	27.667707	2.970359
std	30.399707	7.103501	1.485575
min	7.000000	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

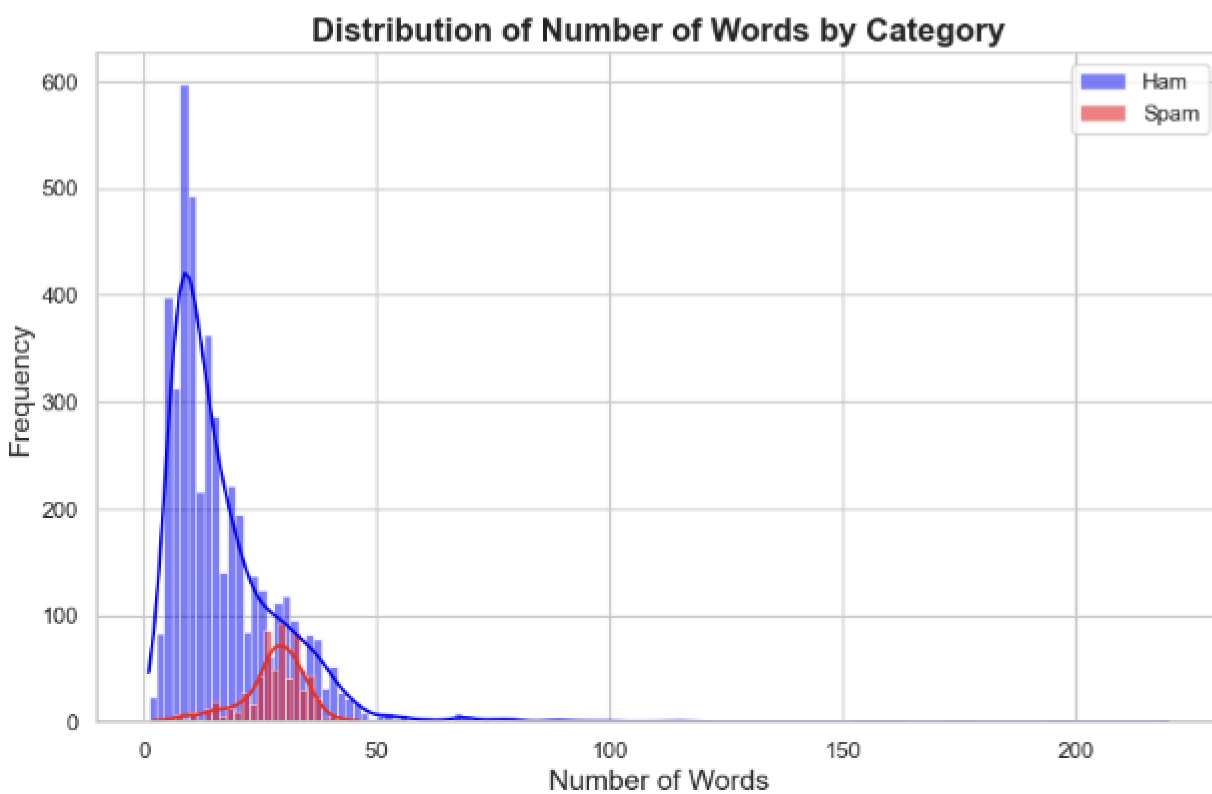
5.5 | Character Length Distribution for Legitimate and Spam Emails

```
import seaborn as sns
import matplotlib.pyplot as plt

def plot_histogram(df, column, category_column):
    plt.figure(figsize=(10, 6))
    sns.histplot(df[df[category_column] == 0][column], color='blue', label='Ham', kde=True)
    sns.histplot(df[df[category_column] == 1][column], color='red', label='Spam', kde=True)
    plt.xlabel('Number of Characters', fontsize=14)
    plt.ylabel('Frequency', fontsize=14)
    plt.title('Distribution of {} by Category'.format(column), fontsize=16, fontweight='bold')
    plt.legend()
    sns.set(style='whitegrid')
    plt.show()

plot_histogram(df, 'num_characters', 'Category')
```



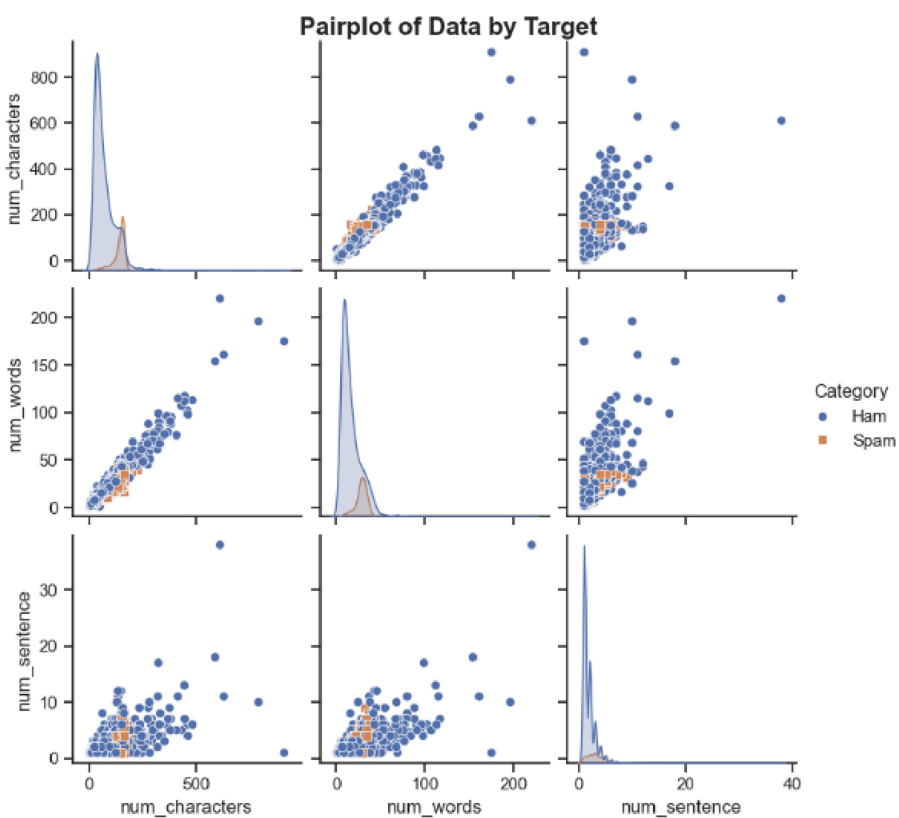
5.6 | Word Count Distribution for Legitimate and Spam Messages

```python
import seaborn as sns
import matplotlib.pyplot as plt

class WordDistributionPlot: #using Object Oriented Programming
    def __init__(self, data):
        self.data = data
        self.fig, self.ax = plt.subplots(figsize=(10, 6))
        sns.set(style='whitegrid')  # Add a white grid background

    def plot(self):
        # Plot the histogram for target 0 in blue
        sns.histplot(self.data[self.data['Category'] == 0]['num_words'], color='blue', label='Ham', kde=True, ax=self.ax)

        # Plot the histogram for target 1 in red
        sns.histplot(self.data[self.data['Category'] == 1]['num_words'], color='red', label='Spam', kde=True, ax=self.ax)

        # Add labels and a title
        self.ax.set_xlabel('Number of Words', fontsize=14)
        self.ax.set_ylabel('Frequency', fontsize=14)
        self.ax.set_title('Distribution of Number of Words by Category', fontsize=16, fontweight='bold')

        # Add a legend
        self.ax.legend()

        # Show the plot
        plt.show()

# Example usage:
# Assuming 'df' is your DataFrame containing the data
plot = WordDistributionPlot(df)
plot.plot()
```



5.7 | Pairplot for Data Visualization

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Create a pairplot with custom styling
sns.set(style='ticks', color_codes=True)
g = sns.pairplot(df, hue='Category', diag_kind='kde', markers=["o", "s"])

# Set a title for the pairplot
g.fig.suptitle("Pairplot of Data by Target", fontsize=16, fontweight='bold')
plt.subplots_adjust(top=0.95)  # Adjust the position of the title

# Customize the legend
g._legend.set_title('Category')
for t, l in zip(g._legend.texts, ["Ham", "Spam"]):
    t.set_text(l)

# Show the pairplot
plt.show()
```



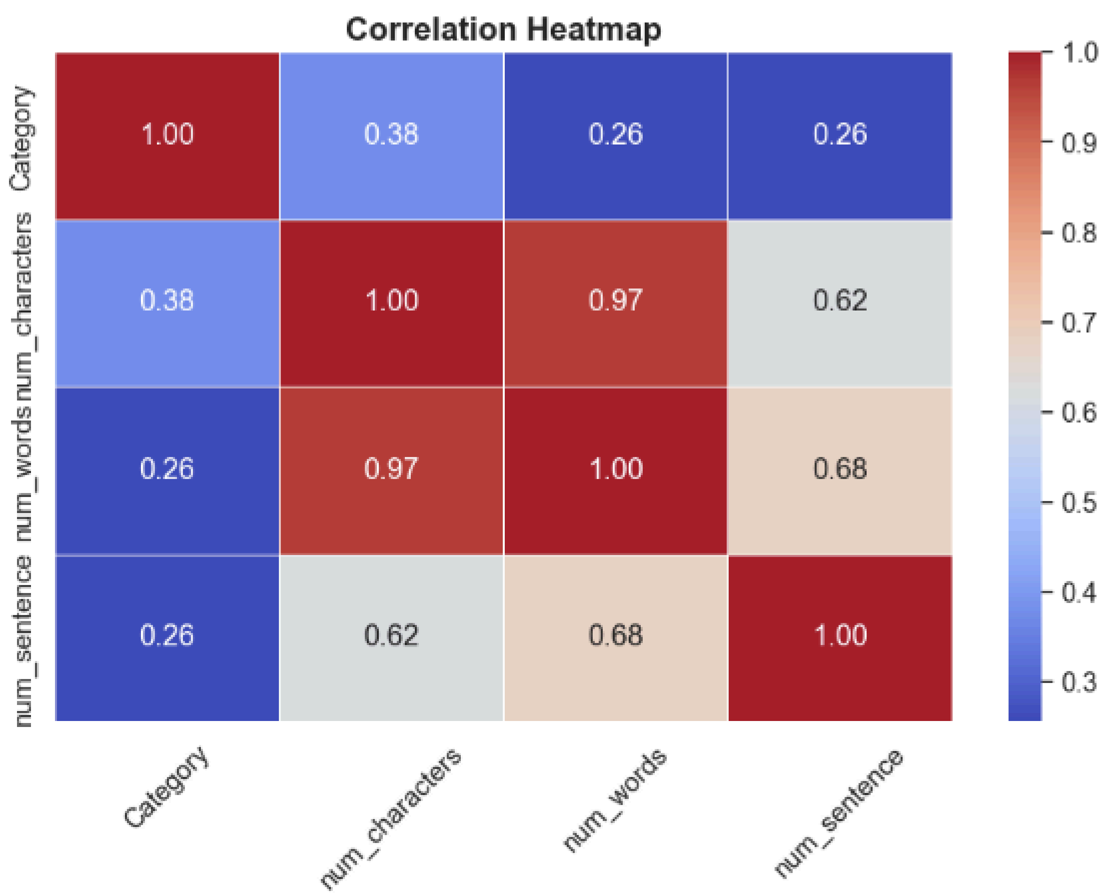
5.8 | Coorelation

```python
df[['Category','num_characters', 'num_words', 'num_sentence']].corr()
```

[22]

	Category	num_characters	num_words	num_sentence
Category	1.000000	0.375897	0.256360	0.259023
num_characters	0.375897	1.000000	0.966006	0.622302
num_words	0.256360	0.966006	1.000000	0.679503
num_sentence	0.259023	0.622302	0.679503	1.000000

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Select the columns for the correlation matrix
correlation_matrix = df[['Category', 'num_characters', 'num_words', 'num_sentence']].corr()

# Create a heatmap with custom styling
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.2)  # Adjust font scale for better readability
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5, fmt=".2f")

# Set a title for the heatmap
plt.title("Correlation Heatmap", fontsize=16, fontweight='bold')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the heatmap
plt.show()
```



Data Preprocessing:

```python
transform_text('Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...')
```

```
'go jurong point crazi avail bugi n great world la e buffet cine got amor wat'
```

6.1 | Creating a New Column: 'transformed_text'

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

```python
styled_df = df.head(5).style

# Modify the color and background color of the table headers (th)
styled_df.set_table_styles([
    {"selector": "th", "props": [("color", 'Black'), ("background-color", "yellow"), ('font-weight', 'bold')]}
])
```

	Category	Message	num_characters	num_words	num_sentence	transformed_text
0	0	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...	111	24	2	go jurong point crazi avail bugi n great world la e buffet cine got amor wat
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 final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's	155	37	2	free entri 2 wkli comp win fa cup final tkt 21st may text fa 87121 receiv entri question std txt rate c appli 08452810075over18
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 around here though	61	15	1	nah think goe usf live around though

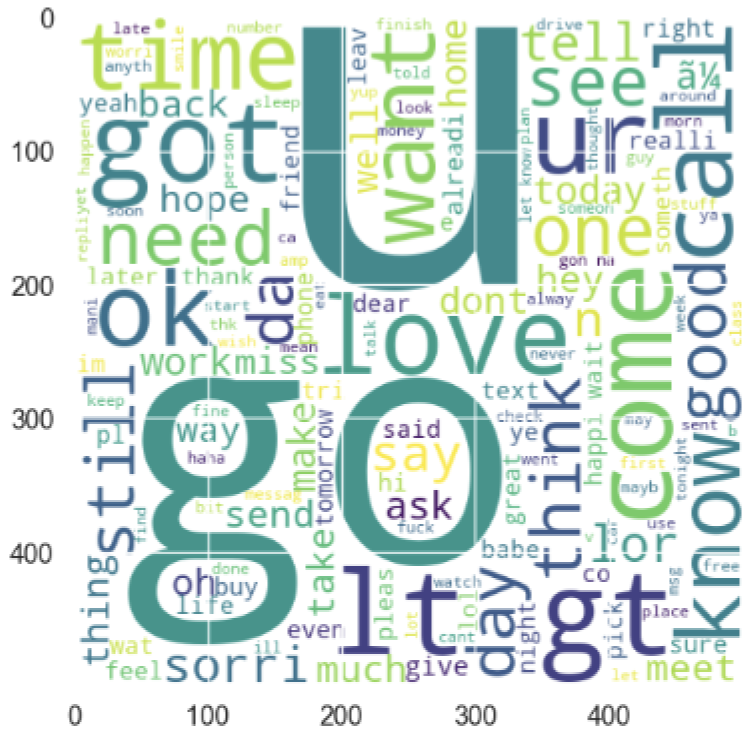
6.2 | Word Cloud for Spam Messages

```python
wc = WordCloud(width = 500, height = 500, min_font_size = 10, background_color = 'white')
spam_wc = wc.generate(df[df['Category'] == 1]['transformed_text'].str.cat(sep = " "))
plt.figure(figsize = (15,6))
plt.imshow(spam_wc)
plt.show()
```

6.2 | Word Cloud for Not spam Messages

```python
ham_wc = wc.generate(df[df['Category'] == 0]['transformed_text'].str.cat(sep = " "))
plt.figure(figsize = (15,6))
plt.imshow(ham_wc)
plt.show()
```

]

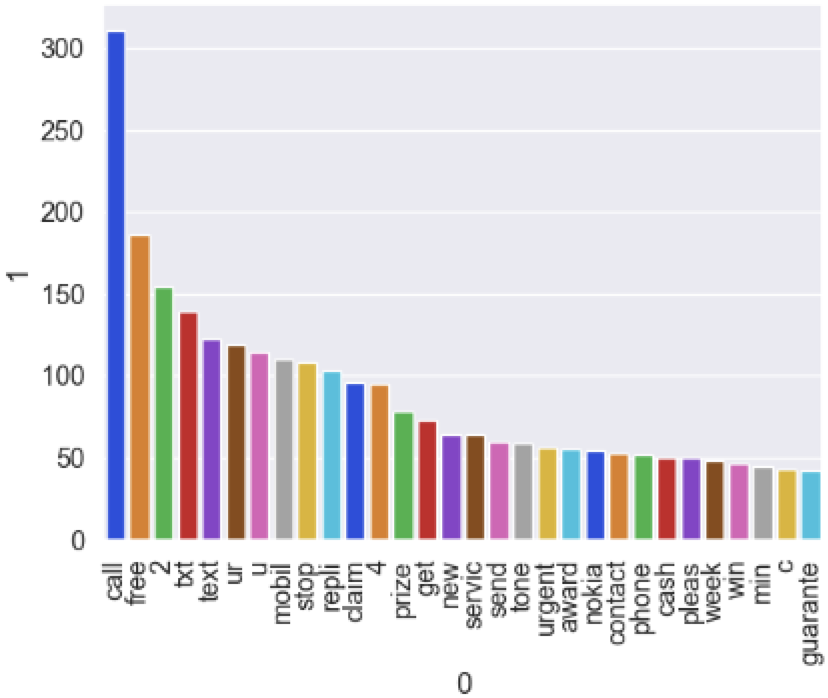


6.3 | Find top 30 words of spam

```python
spam_carpos = []
for sentence in df[df['Category'] == 1]['transformed_text'].tolist():
    for word in sentence.split():
        spam_carpos.append(word)
```

```python
from collections import Counter
filter_df = pd.DataFrame(Counter(spam_carpos).most_common(30))
```

```python
sns.barplot(data = filter_df, x = filter_df[0], y = filter_df[1], palette = 'bright')
plt.xticks(rotation = 90)
plt.show()
```

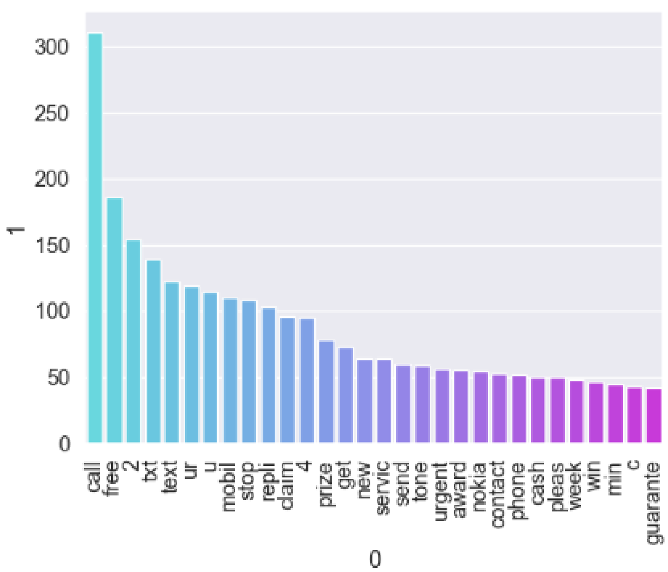


6.4 | Find top 30 words of Not spam Messages

```python
ham_carpos = []
for sentence in df[df['Category'] == 0]['transformed_text'].tolist():
    for word in sentence.split():
        ham_carpos.append(word)
```

```python
filter_ham_df = pd.DataFrame(Counter(spam_carpos).most_common(30))
```

```python
sns.barplot(data = filter_ham_df, x = filter_ham_df[0], y = filter_ham_df[1], palette = 'cool')
plt.xticks(rotation = 90)
plt.show()
```



Conclusion:

Conclusion: In our evaluation of various classification algorithms, we observed the following key insights:

- Support Vector Classifier (SVC) and Random Forest (RF) demonstrated the highest accuracy, both achieving approximately **97.58%**.
- Naive Bayes (NB) achieved a perfect precision score, indicating zero false positives.
- Other models, including Gradient Boosting, Adaboost, Logistic Regression, and Bagging Classifier, displayed competitive performance with accuracy scores ranging from **94.68%** to **96.03%**.

The selection of the optimal model should consider factors beyond just accuracy, such as computational efficiency and the specific requirements of the application. It is advisable to perform further model fine-tuning and validation before making a final choice.