OPEN SOURCE SOFTWARE LAB
(15B17CI575)
PROJECT REPORT

PROJECT TITLE: SPAM MAIL DETECTION USING ML

विद्या तत्व ज्योतिसमः

GROUP MEMBERS

BATCH: B5

NAME	ENROLLMENT NUMBER
ANSHUMAN SINGH	21103115
BHAVYA VATS	21103134
SURYANSH RATHORE	21103137
SARTHAK PAINULI	21103282

UNDER SUPERVISION OF:-
Mrs. Purtee Kohli

ABSTRACT

Spam Mail Detection Using Machine Learning is a project designed to tackle the pervasive problem of spam emails in today's digital communication landscape. The project leverages machine learning techniques to automatically classify incoming emails as either "spam" or "non-spam" (ham). By analyzing email content and metadata, this system provides an effective means of protecting users from unsolicited and potentially harmful emails.

PROJECT DESCRIPTION

The project involves the collection of a diverse and labeled dataset of emails, including both spam and non-spam examples. After data preprocessing, relevant features are extracted, and machine learning models are employed for classification. Various machine learning algorithms are evaluated, and the best-performing model is selected. Continuous improvement mechanisms are also implemented to ensure the model remains effective in identifying evolving spam tactics. The project has the following benefits:

1. **Improved Email Filtering**: The system enhances email services by automatically filtering out spam, reducing inbox clutter, and improving user experience.

2. **Security**: It safeguards users from potentially harmful content often found in spam emails, such as phishing attempts, malware, and scams.

3. **Time and Resource Efficiency**: Users save time and effort by not having to manually sift through spam emails.

4. **Customization**: The system can be tailored to individual preferences, allowing users to customize the level of aggressiveness in spam filtering.

5. **Reduced False Positives**: The system's machine learning models can be fine-tuned to minimize false positives, ensuring that legitimate emails are not mistakenly classified as spam. This leads to a more accurate and reliable spam filtering process, preserving important emails and preventing potential loss of critical information.

BACKGROUND STUDY

A fundamental task in natural language processing, involves categorizing text into predefined classes or categories. Naive Bayes algorithms are widely employed for this purpose due to their simplicity and effectiveness. Specifically, the Multinomial Naive Bayes algorithm is a popular choice for text-based tasks, including spam detection. Multinomial Naive Bayes for Text Classification Multinomial Naive Bayes is an extension of the Naive Bayes algorithm, tailored for text classification tasks. It operates under the assumption that the presence and frequency of each term (or word) contribute independently to the probability of a particular class.

1.Advantages in Spam Detection:

In the realm of spam detection, Multinomial Naive Bayes demonstrates several advantages. Its simplicity allows for efficient training and inference, even with large datasets comprising numerous features (in this case, words).

2.Challenges and Limitations :

However, it's crucial to acknowledge the limitations of Multinomial Naive Bayes. The algorithm assumes independence among features, meaning it doesn't consider the relationships or dependencies between words. In real-world scenarios, this assumption might not always hold true, impacting the model's accuracy. Additionally, rare or previously unseen words in the training data might pose challenges for classification.

3. State-of-the-Art Practices and Advancements :

Current advancements in spam detection using Multinomial Naive Bayes involve various strategies to enhance its performance. Techniques like TF-IDF (Term Frequency-Inverse Document Frequency) help in weighing the importance of words, giving more significance to terms that are rare yet essential in distinguishing between spam and non-spam content.

4. **Comparative Analysis and Future Directions:**

In comparison to other state-of-the-art algorithms used in spam detection, Multinomial Naive Bayes exhibits strengths in computational efficiency and handling large feature spaces. However, its performance might lag behind more complex models that capture intricate relationships among words.

ALGORITHM:

Overview:
The Multinomial Naive Bayes (MNB) algorithm is widely used for text classification, including spam detection. It assumes independence between features (word frequencies) and computes probabilities based on Bayes' theorem.

Steps:

1.Data Preparation:

Tokenization: Convert text into words.
Feature Extraction: Count occurrences of words in documents.

2.Model Building:

Probability Estimation: Calculate word probabilities for spam and non-spam messages.
Prior Probabilities: Compute prior probabilities for each class.

3.Classification:

<u>Posterior Probability Calculation:</u> Assess the likelihood of a message belonging to each class based on word frequencies.
<u>Class Prediction:</u> Assign the class with the highest probability.

Discussion:

1.Strengths:
Efficient with large text datasets.
Handles high-dimensional data (word frequencies).

2.Limitations:
Assumes independence among words, which might not hold in all cases.
May struggle with unseen words.

3.Performance:

Efficient and suitable for real-time applications.

Might not capture complex word relationships compared to more intricate algorithms.

4.Adaptability:
Can be extended with additional features.
Often used with techniques like Laplace smoothing or TF-IDF for enhanced performance.

Conclusion:
Multinomial Naive Bayes is an efficient choice for spam detection, offering speed and scalability with large text datasets. While simplistic, it performs well in many cases, but its assumptions might limit its accuracy in capturing intricate word relationships.

DESCRIPTION OF STEPS OF THE PROJECT

The following stages collectively form a comprehensive approach to building a robust and reliable Spam Mail Detection system using Machine Learning, from data collection and preprocessing to model evaluation and deployment in a user-friendly web interface.

1. **DATA CLEANING**

   Data Collection: Collect a diverse dataset of emails, including both spam and non-spam examples, ensuring that it's representative of real-world scenarios.
   Handling Missing Data: Check for and address any missing or incomplete data in the dataset.
   Deduplication: Remove duplicate emails if they exist in the dataset.
   Noise Removal: Eliminate irrelevant or redundant information, such as HTML tags, special characters, or unnecessary whitespace.
   Standardization: Standardize features like email addresses, URLs, and dates for consistent formatting.

2. **EXPLORATORY DATA ANALYSIS (EDA)**

   Data Visualization: Use plots and charts to explore the dataset's characteristics, like the distribution of spam vs. non-spam emails.
   Feature Analysis: Examine the importance of different features and their potential predictive power.
   Correlation Analysis: Investigate correlations between features and target variables.
   Outlier Detection: Identify and handle outliers that may adversely affect model training.

3. **TEXT PREPROCESSING**

   Tokenization: Break down the text into individual words or tokens.
   Stop Word Removal: Eliminate common and uninformative words (e.g., "the," "and," "in") that may not contribute to classification.
   Stemming or Lemmatization: Reduce words to their root forms for consistency.

Text Vectorization: Convert text data into numerical representations, such as TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings.

4. **MODEL BUILDING**

Feature Selection: Choose relevant features and attributes that contribute to classification while reducing dimensionality.
Model Selection: Experiment with various machine learning algorithms, such as Naive Bayes, Support Vector Machines, Random Forest, and neural networks.
Hyperparameter Tuning: Fine-tune model parameters to optimize performance.
Cross-Validation: Implement techniques like k-fold cross-validation to assess model generalization and avoid overfitting.

5. **MODEL EVALUATION**

Performance Metrics: Assess the model using metrics like accuracy, precision, recall, F1-score, and ROC-AUC.
Confusion Matrix: Analyze false positives and false negatives, understanding the model's behavior.
ROC Curves: Visualize the trade-off between true positive rate and false positive rate for different threshold settings.

6. **MODEL IMPROVEMENT**

Ensemble Methods: Consider using ensemble techniques like bagging (e.g., Random Forest) or boosting (e.g., AdaBoost) to improve model performance.
Feature Engineering: Experiment with creating new features or transformations that enhance the model's ability to distinguish between spam and non-spam emails.
Feedback Loop: Implement mechanisms to continuously improve the model based on user feedback and evolving spam tactics.

7. WEBSITE

Develop a user-friendly web application interface for users to interact with the spam mail detection system.

Incorporate user settings for customization, allowing users to adjust the level of aggressiveness in spam filtering.

Ensure a responsive and intuitive design that provides feedback and guidance to users.

IMPLEMENTATION

```
import numpy as np
import pandas as pd
df = pd.read_csv('spam.csv', encoding='latin1')
df.sample(4)
df.shape
df.info()
# 1)cleaning and dropping last 3 columns
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
df.sample(5)
df.rename(columns={'v1':'target','v2':'text'},inplace=True)
df.sample(5)
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
df.head()
df['target'] = encoder.fit_transform(df['target'])
#missing values
df.isnull().sum()
#check for duplicate values
df.duplicated().sum()
df.head()
df=df.drop_duplicates(keep='first')# removing duplicates
df.duplicated().sum()
df['target'].value_counts()
import matplotlib.pyplot as plt
plt.pie(df['target'].value_counts(),labels=['ham','spam'],autopct="%0.2f")
plt.show()
#data is imbalanced
import nltk
nltk.download('punkt')
df['num_characters']=df['text'].apply(len)
#num of words
df['num_words']=df['text'].apply(lambda x:len(nltk.word_tokenize(x)))
df['num_sentences']=df['text'].apply(lambda x:len(nltk.sent_tokenize(x)))
df[['num_characters','num_words','num_sentences']].describe()
#ham
df[df['target']==0][['num_characters','num_words','num_sentences']].describe()
#spam
df[df['target']==1][['num_characters','num_words','num_sentences']].describe()
import seaborn as sns
```

```
sns.histplot(df[df['target']==0]['num_characters'])
sns.histplot(df[df['target']==1]['num_characters'],color='red')
sns.pairplot(df,hue='target')
#correlation coeffcient
# Check if the DataFrame contains numeric columns and select only those for correlation
numeric_df = df.select_dtypes(include=['int64', 'float64'])

# Create a correlation matrix
correlation_matrix = numeric_df.corr()

# Create the heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
# 3) DATA PROCESSING
#Lower Case
#Tokenization
#Removing special characters
#Removing stop words and punctuations
#stemming
from nltk.corpus import stopwords
import string

import nltk
nltk.download('stopwords')

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)
df['transformed_text']=df['text'].apply(transform_text)
from wordcloud import WordCloud
wc=WordCloud(width=500,height=500,min_font_size=10,background_color='white')
spam_text = " ".join(df[df['target'] == 1]['transformed_text'])  # Concatenate
transformed_text for spam target
spam_wc = wc.generate(spam_text)  # Generate WordCloud from concatenated text
plt.imshow(spam_wc) # showing all spam words
ham_text = " ".join(df[df['target'] == 0]['transformed_text'])
ham_wc = wc.generate(ham_text)
plt.imshow(ham_wc)              # showing ham words
spam_corpus=[]
for msg in df[df['target']==1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)

len(spam_corpus)  # total spam words
from collections import Counter
import matplotlib.pyplot as plt
word_count = Counter(spam_corpus).most_common(30)
df_word_count = pd.DataFrame(word_count, columns=['Word', 'Count'])

# Creating the barplot
plt.figure(figsize=(10, 6))  # Adjust the figure size as needed
plot = sns.barplot(x='Word', y='Count', data=df_word_count)
plot.set_xticklabels(plot.get_xticklabels(), rotation=45)  # Rotate x-axis labels by 45
degrees

# Display the plot
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
# 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
y=df['target'].values
y
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(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))  # 86%accuracy # precision score less is 0.50
mnb.fit(X_train,y_train)
y_pred2=mnb.predict(X_test)
print(accuracy_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
print(precision_score(y_test,y_pred2))  # 100% precision and 97%accuracy
bnb.fit(X_train,y_train)
y_pred3=bnb.predict(X_test)
print(accuracy_score(y_test,y_pred3))
print(confusion_matrix(y_test,y_pred3))
print(precision_score(y_test,y_pred3))   # 99% precision and 98%accuracy
import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(mnb,open('model.pkl','wb'))
```

```python
import streamlit as st
import pickle
import string
from nltk.corpus import stopwords
import nltk
from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()

tfidf = pickle.load(open('vectorizer.pkl', 'rb'))
model = pickle.load(open('model.pkl', 'rb'))
st.title("Email/SMS Spam Classifier")
input_sms = st.text_area("Enter The Message")
if st.button('Predict'):
   # Preprocess
   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_sms = transform_text(input_sms)
   # Vectorize
   vector_input = tfidf.transform([transform_sms])
   # Predict
   result = model.predict(vector_input)[0]
   # Display
   if result == 1:
       st.header("SPAM")
   else:
       st.header("NOT SPAM")
```

EXPERIMENTATION AND RESULTS

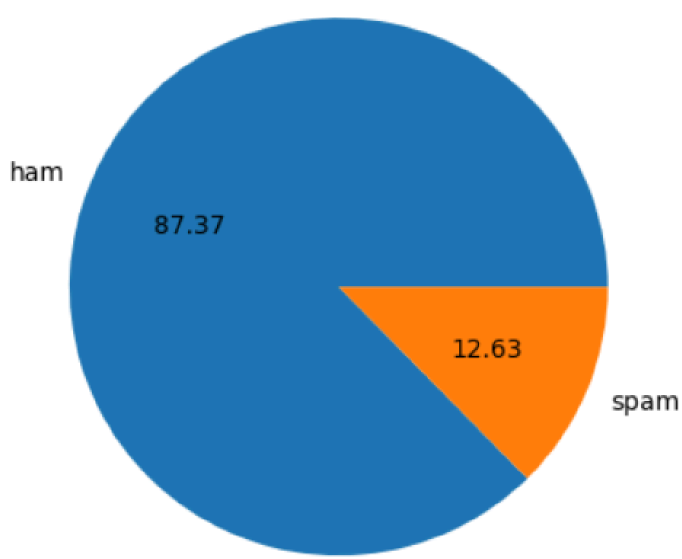
```
import numpy as np
import pandas as pd
```

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

```
df.sample(4)
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
1950	ham	Oh ic. I thought you meant mary jane.	NaN	NaN	NaN
3074	ham	Take us out shopping and Mark will distract Is...	NaN	NaN	NaN
5196	spam	Spook up your mob with a Halloween collection ...	NaN	NaN	NaN
4882	ham	New Theory: Argument wins d SITUATION, but los...	NaN	NaN	NaN

[157]:
```
import matplotlib.pyplot as plt
plt.pie(df['target'].value_counts(),labels=['ham','spam'],autopct="%0.2f")
plt.show()
```



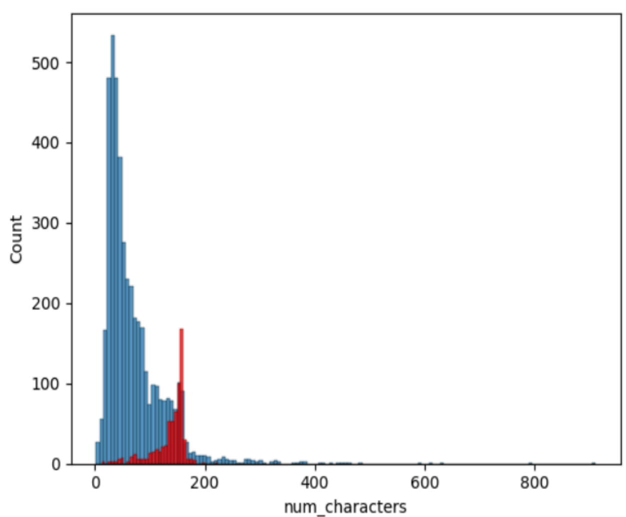
```
df[['num_characters','num_words','num_sentences']].describe()
```

	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

[170]:

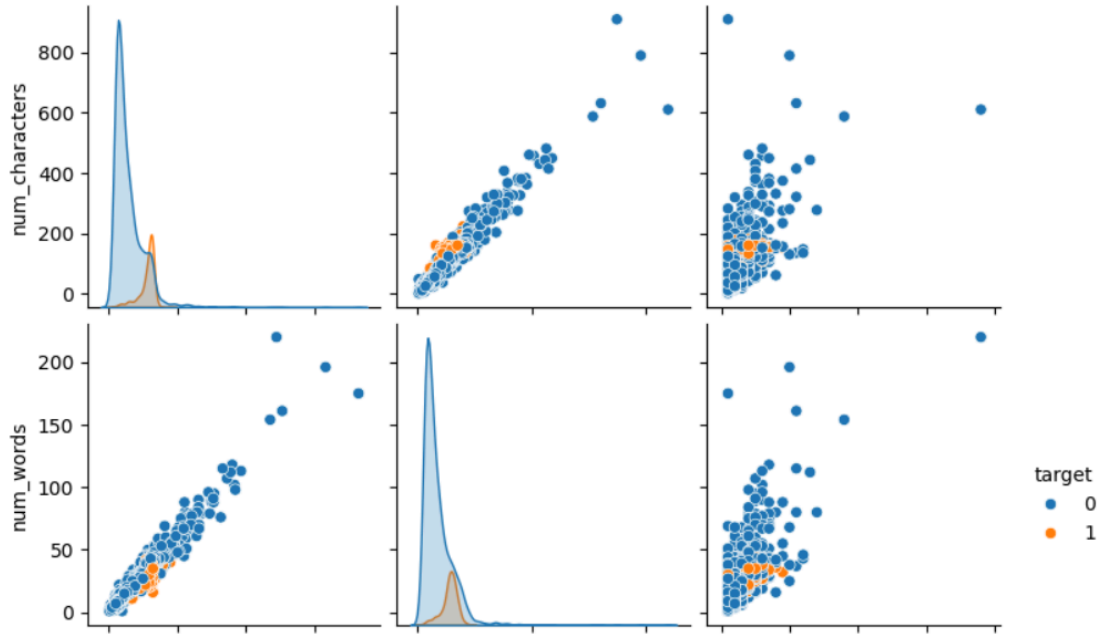
```
import seaborn as sns
sns.histplot(df[df['target']==0]['num_characters'])
sns.histplot(df[df['target']==1]['num_characters'],color='red')
```

[170]: <Axes: xlabel='num_characters', ylabel='Count'>



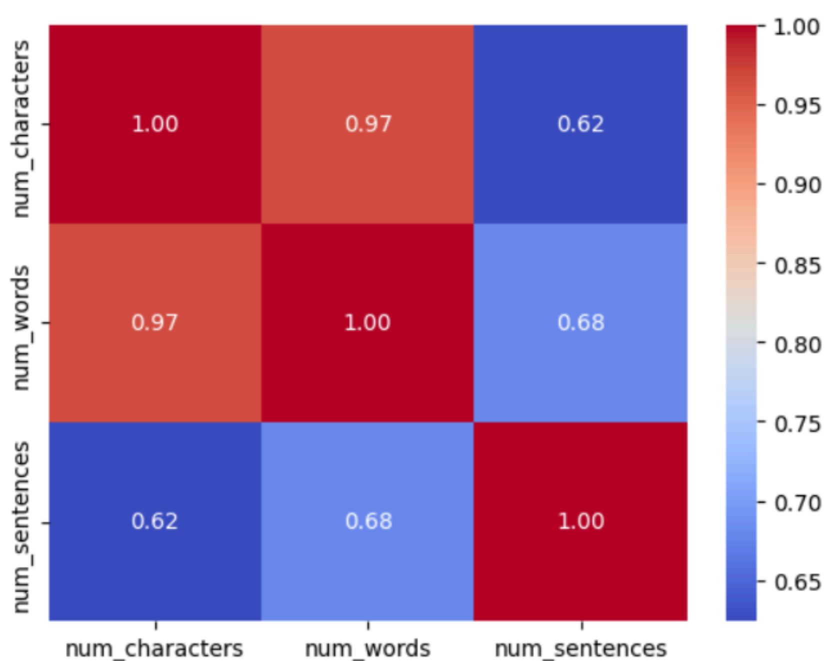
[171]:
```python
sns.pairplot(df,hue='target')
```

[171]: <seaborn.axisgrid.PairGrid at 0x2459886d990>



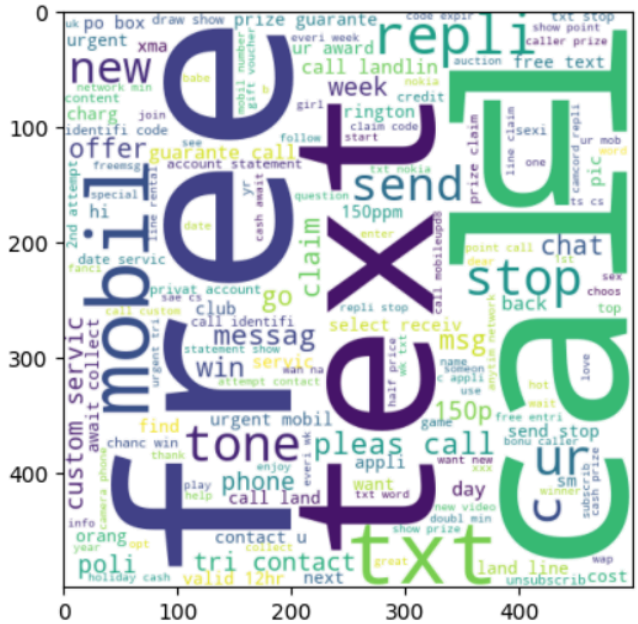

```python
# Create a correlation matrix
correlation_matrix = numeric_df.corr()

# Create the heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
```

[172]: <Axes: >



```python
plt.imshow(spam_wc) # showing all spam words
```

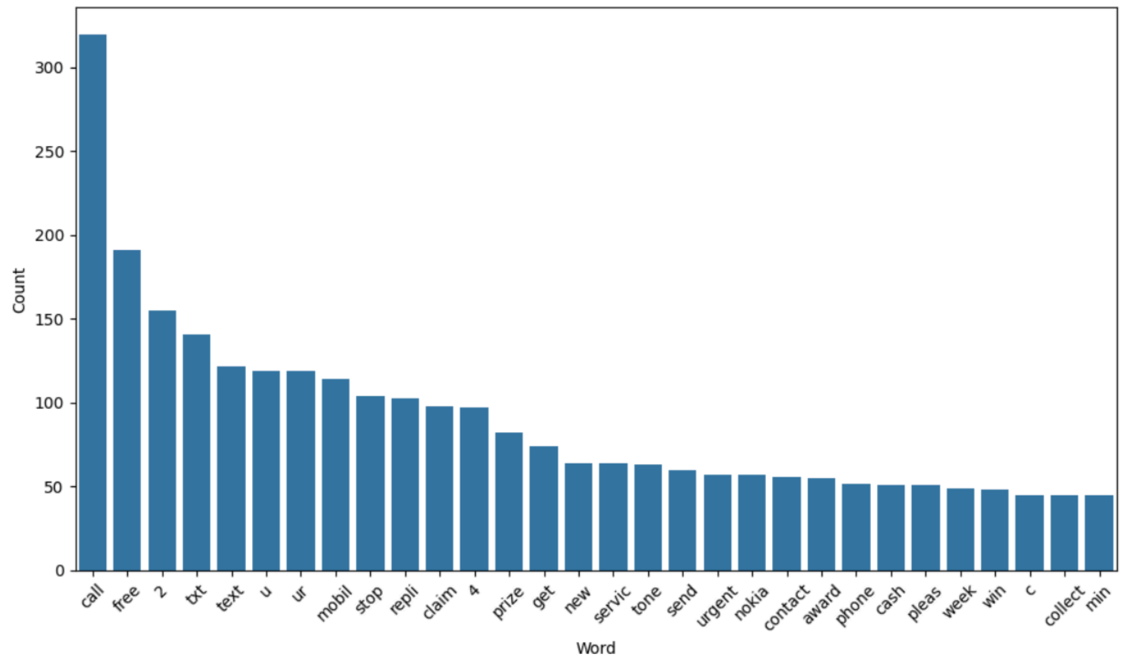
```
<matplotlib.image.AxesImage at 0x245995ec910>
```

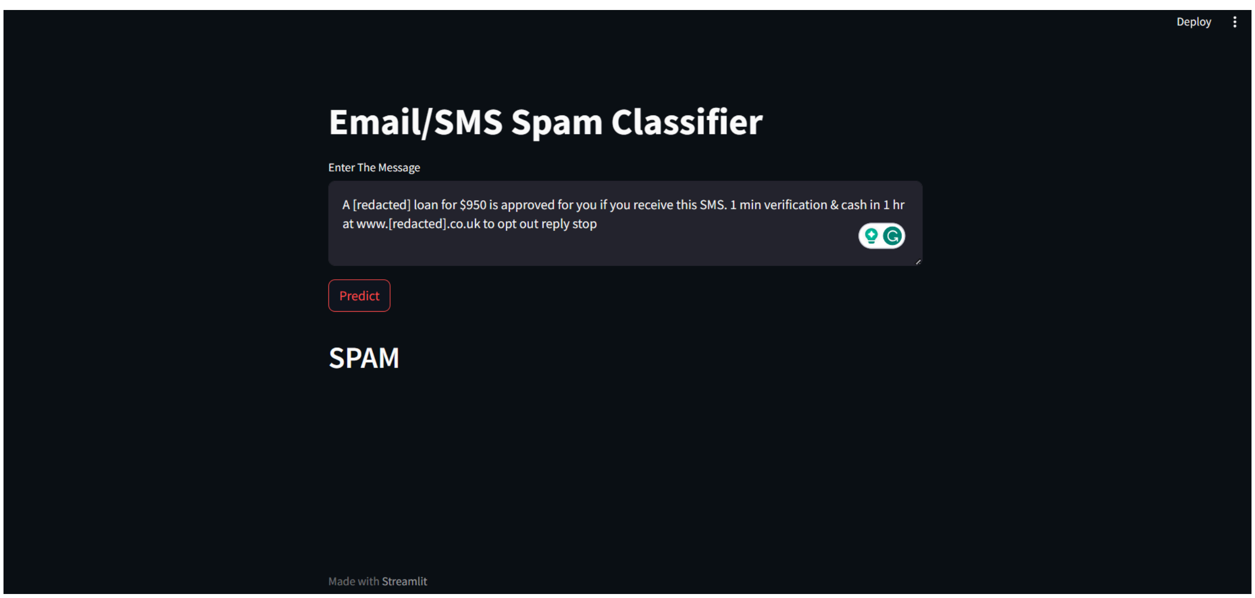



```python
ham_text = " ".join(df[df['target'] == 0]['transformed_text'])
ham_wc = wc.generate(ham_text)
plt.imshow(ham_wc)               # showing ham words
```

```
<matplotlib.image.AxesImage at 0x245994fbad0>
```

```
after set_ticks() or using a FixedLocator.
  plot.set_xticklabels(plot.get_xticklabels(), rotation=45)  # Rotate x-axis labels by 45 degrees
```





CONCLUSION

Spam Mail Detection Using Machine Learning provides an effective solution to the perennial problem of spam emails. By applying machine learning algorithms to classify incoming emails, the system significantly improves email services, enhances user security, and streamlines email management. The use of modern technologies, such as Python, machine learning libraries, and NLP, ensures the system's accuracy and efficiency. Continuous improvement mechanisms make this project adaptive to evolving spam tactics. Overall, this project contributes to a safer and more convenient digital communication experience for users, making it an invaluable addition to email services.

REFERENCES

- https://perfectelearning.com/blog/spam-mail-detection-using-machine-learning
- https://towardsdatascience.com/email-spam-detection-1-2-b0e06a5c0472
- https://www.hindawi.com/journals/scn/2022/1862888/
- Email Spam Detection Using MachineLearning by Mrs. Anitha Reddy1*, Kanthala Harivardhan Reddy2 , A. Abhishek3 , Myana Manish4 , G. Viswa Sai Dattu5 , Noor Mohammad Ansari
- https://www.researchgate.net/publication/344050184_Email_Spam_Detection_Using_Machine_Learning_Algorithms