Building a Smarter AI-Powered Spam Classifier

TEAM MEMBERS
G. GANESH GOPAL (821021104019)
S. ABDUR RAZZAK (821021104004)
K. SRI RAM (821021104044)
B. GANESH (821021104302)
K. SIVA PRAKASH (821021104305)

PHASE 4 DEVELOPMENT PART 2

TASK

Building Project Spam Classifier By Selecting A Machine Learning Algorithm, Training The Model , Evaluating Its Performance

DATASET LINK

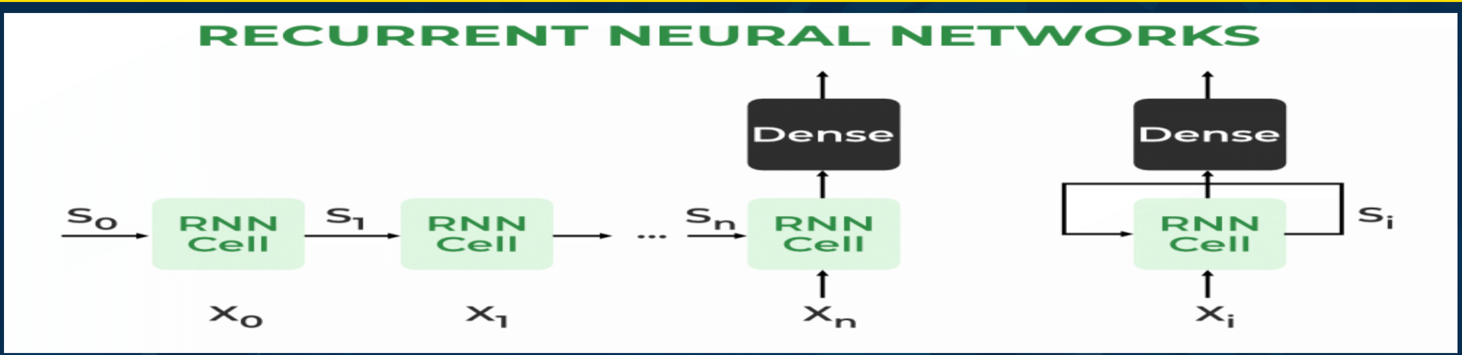
https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

OBJECT DETECTION WITH YOLO

- Object detection is a popular task in computer vision.
- YOLO (You Only Look Once) is a popular object detection model known for its speed and accuracy. It was first introduced by Joseph Redmon et al. in 2016 and has since undergone several iterations, the latest being YOLO v7.
- YOLO divides an input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes.

RECURRENT NEURAL NETWORKS

- Recurrent Neural Network(RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step.
- The main and most important feature of RNN is its **Hidden state**, which remembers some information about a sequence. The state is also referred to as *Memory State* since it remembers the previous input to the network.



Advantages of Recurrent Neural Network

- An RNN remembers each and every piece of information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory.

Disadvantages of Recurrent Neural Network

- Gradient vanishing and exploding problems.
- Training an RNN is a very difficult task.

Applications of Recurrent Neural Network

1. Language Modelling and Generating Text
2. Speech Recognition
3. Machine Translation
4. Image Recognition, Face detection
5. Time series Forecasting

TYPES OF RNN :

1. One to One
2. One to Many
3. Many to One
4. Many to Many

Natural language processing

- Natural language processing (NLP) is a machine learning technology that gives computers the ability to interpret, manipulate, and comprehend human language. Organizations today have large volumes of voice and text data from various communication channels like emails, text messages, social media newsfeeds, video, audio, and more.

Why Nlp Is Important ?

Natural language processing (NLP) is critical to fully and efficiently analyze text and speech data. It can work through the differences in dialects, slang, and grammatical irregularities typical in day-to-day conversations.

How does NLP work?

Natural language processing (NLP) combines computational linguistics, machine learning, and deep learning models to process human language.

- Computational linguistics
- Machine learning
- Deep learning
- NLP implementation steps
- Pre-processing
- Training
- Deployment and inference

What are NLP tasks?

Natural language processing (NLP) techniques, or NLP tasks, break down human text or speech into smaller parts that computer programs can easily understand. Common text processing and analyzing capabilities in NLP are given below.

- Part-f-speech tagging
- Word-sense disambiguation
- Speech recognition
- Machine translation
- Named-entity recognition
- Sentiment analysis

Approaches to NLP?

- Supervised NLP
- Unsupervised NLP
- Natural language understanding
- Natural language generation

Data Collection and Preprocessing:

1. Download the dataset from the provided Kaggle link.
2. Load the dataset and examine its structure.
3. Preprocess the text data by removing punctuation, converting text to lowercase, and tokenizing the messages.

Data Exploration:

Explore the dataset to understand its distribution, class balance, and any patterns in the data.

Feature Extraction:

Convert the text messages into numerical features. You can use TF-IDF (Term Frequency-Inverse Document Frequency) or other techniques like Word Embeddings (Word2Vec, GloVe).

Split Data:

Split the dataset into a training set, a validation set, and a test set. A common split is 70% for training, 15% for validation, and 15% for testing.

Select a Machine Learning Algorithm:

Choose a machine learning algorithm for text classification. For this task, a good starting point is to use Multinomial Naive Bayes, which is a common choice for text classification problems.

Model Training:

Train the selected algorithm on the training data using the features you've extracted.

program

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_score
import nltk
from nltk.corpus import stopwords
from collections import Counter
```

Libraries for visualisation

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

Download the stopwords dataset

In [3]:
```
nltk.download('stopwords')
```

Out [3]:
```
True
```

Reading and Describing Data

In [4]:
```
# Loading the dataset
df = pd.read_csv("/kaggle/input/sms-spam-collection-dataset/spam.csv",encoding='latin-1')

# Displaying the first few rows of the dataset
```

In [5]:
```
df.head()
```

Out [5]:

	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 [6]:
```
# Droping unnecessary columns from the DataFrame
columns_to_drop = ["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"]
df.drop(columns=columns_to_drop, inplace=True)
```

In [7]:
```
# Displaying the data
df
```

Out [7]:

	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...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will Ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

In [8]:
```
# Consice information of the dataset
df.info()
```

Out [8]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   v1      5572 non-null   object
 1   v2      5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

In [9]:

```
df.shape
```

Out [9]:

```
(5572, 2)
```

In [10]:

```
df.describe()
```

Out [10]:

	v1	v2
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

In [11]:

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

Out [11]:

```
v1    0
v2    0
dtype: int64
```

In [12]:

```
df.columns
```

Out [12]:

```
Index(['v1', 'v2'], dtype='object')
```

In [13]:

```
# Rename the columns "v1 and "v2" to new names
new_column_names = {"v1":"Category","v2":"Message"}
df.rename(columns = new_column_names,inplace = True)
```

In [14]:

```
df.head()
```

Out [14]:

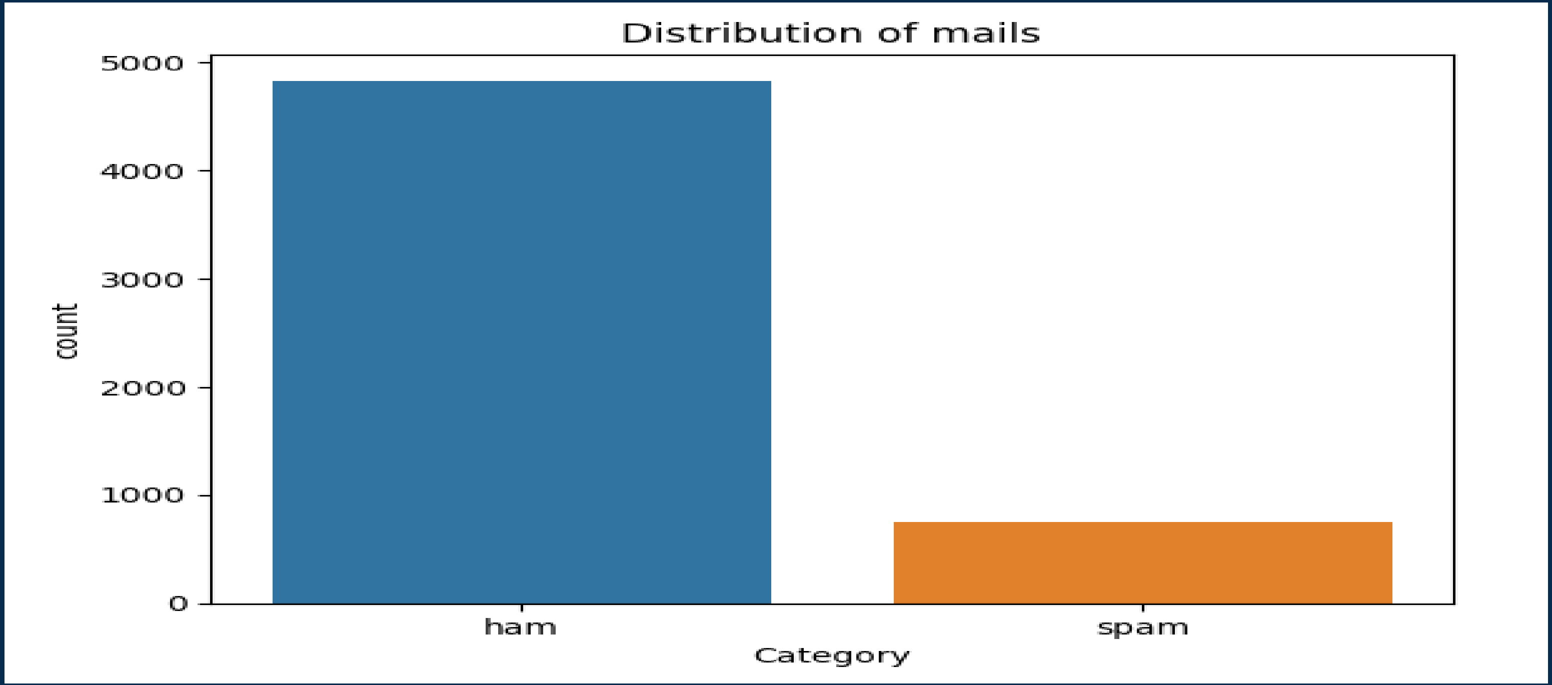
	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Data Visualisation

In [15]:

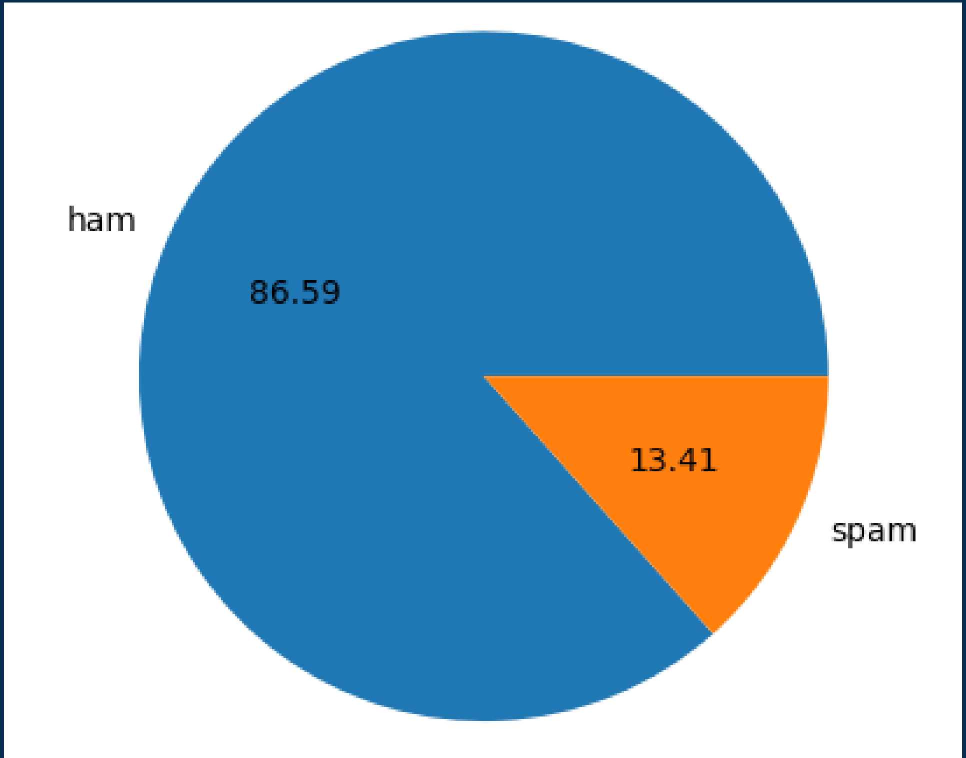
```
sns.countplot(data=df, x='Category')
plt.xlabel('Category')
plt.ylabel('count')
plt.title('Distribution of mails')
plt.show()
```

Out [15]:



In [16]:

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



Data Preprocessing
Label Encoding

In [17]:

```
df.loc[df["Category"] == "spam", "Category"] = 0
df.loc[df["Category"] == "ham", "Category"] = 1
```

In [18]:

```
# Separate the feature (message) and target (category) data
X = df["Message"]
Y = df["Category"]
```

In [19]:

```
print(X)
```

Out [19]:

```
0       Go until jurong point, crazy.. Available only ...
1                           Ok lar... Joking wif u oni...
2       Free entry in 2 a wkly comp to win FA Cup fina...
3       U dun say so early hor... U c already then say...
 :                :                :               :                 :
5568                 Will Ì_ b going to esplanade fr home?
5569    Pity, * was in mood for that. So...any other s...
5570    The guy did some bitching but I acted like i'd...
5571                            Rofl. Its true to its name
Name: Message, Length: 5572, dtype: object
```

In [20]:

```
print(Y)
```

Out [20]:

```
0       1
1       1
2       0
3       1
4       1
5567    0
5568    1
5569    1
5570    1
5571    1
Name: Category, Length: 5572, dtype: object
```

Splitting the data into training data and test data

In [21]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 3)
```

In [22]:

```
# Print the shape of X
print(X.shape)
```

Out [22]:

```
(5572,)
```

In [23]:

```
# Print the shape of X_train and X_test
print(X_train.shape)
print(X_test.shape)
```

Out [23]:

```
(4457,)
(1115,)
```

Feature Extraction

TF-IDF Vectorizer

In [24]:

```
# Initialize TF-IDF Vectorizer
feature_extraction = TfidfVectorizer(min_df=1, stop_words="english", lowercase=True)
```

In [25]:

```
# Feature extraction for training and testing data
X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)
```

In [26]:

```
# Convert Y_train and Y_test to integer type
Y_train = Y_train.astype("int")
Y_test = Y_test.astype("int")
```

In [27]:

```
print(X_train)
```

Out [27]:

```
3075    Mum, hope you are having a great day. Hoping t...
1787                          Yes:)sura in sun tv.:)lol.
1614    Me sef dey laugh you. Meanwhile how's my darli...
4304                Yo come over carlos will be here soon
3266                    Ok then i come n pick u at engin?
                              ...
789                         Gud mrng dear hav a nice day
968             Are you willing to go for aptitude class.
1667    So now my dad is gonna call after he gets out ...
3321    Ok darlin i supose it was ok i just worry too ...
1688                   Nan sonathaya soladha. Why boss?
Name: Message, Length: 4457, dtype: object
```

In [28]:

```
print(X_train_features)
```

Out [28]:

```
(0, 741)      0.3219352588930141
(0, 3979)     0.2410582143632299
(0, 4296)     0.3891385935794867
:    :           :          :          :
(4456, 6133)           0.5304350313291551
(4456, 1386)           0.446003631644607
(4456, 4557)           0.48821933148688146
```

Model Selection and Training
Logistic Regresion

In [29]:
```
# Creating and Fit Logistic Regression Model
model = LogisticRegression()
model.fit(X_train_features, Y_train)
```

Out [29]:
```
>   LogisticRegression
LogisticRegression()
```

Evaluating the trained model

In [30]:
```
 #Make predictions on the training data
predict_train_data=model.predict(X_train_features)
```

In [31]:
```
#Model Evaluation
from sklearn.metrics import accuracy_score,confusion_matrix
accuracy_train_data=accuracy_score(Y_train,predict_train_data)
print("Accuracy on training data: ",accuracy_train_data)
```

Out [31]:
```
Accuracy on training data:  0.9661207089970832
```

In [32]:

```
# Make predictions on the testing data
predict_test_data=model.predict(X_test_features)
```

In [33]:

```
#Model Evaluation
accuracy_test_data=accuracy_score(Y_test,predict_test_data)
print("acuuracy on test data: ",accuracy_test_data)
```

Out [33]:

```
acuuracy on test data:  0.9623318385650225
```

Test the model with an email messages

In [34]:

```
new_mail=["Congratulations on your recent achievement! Well done."]
new_data_features=feature_extraction.transform(new_mail)
prediction=model.predict(new_data_features)
print(prediction)
if(prediction[0]==1):
    print("Ham Mail")
else:
    print("Spam Mail")
```

Out [34]:

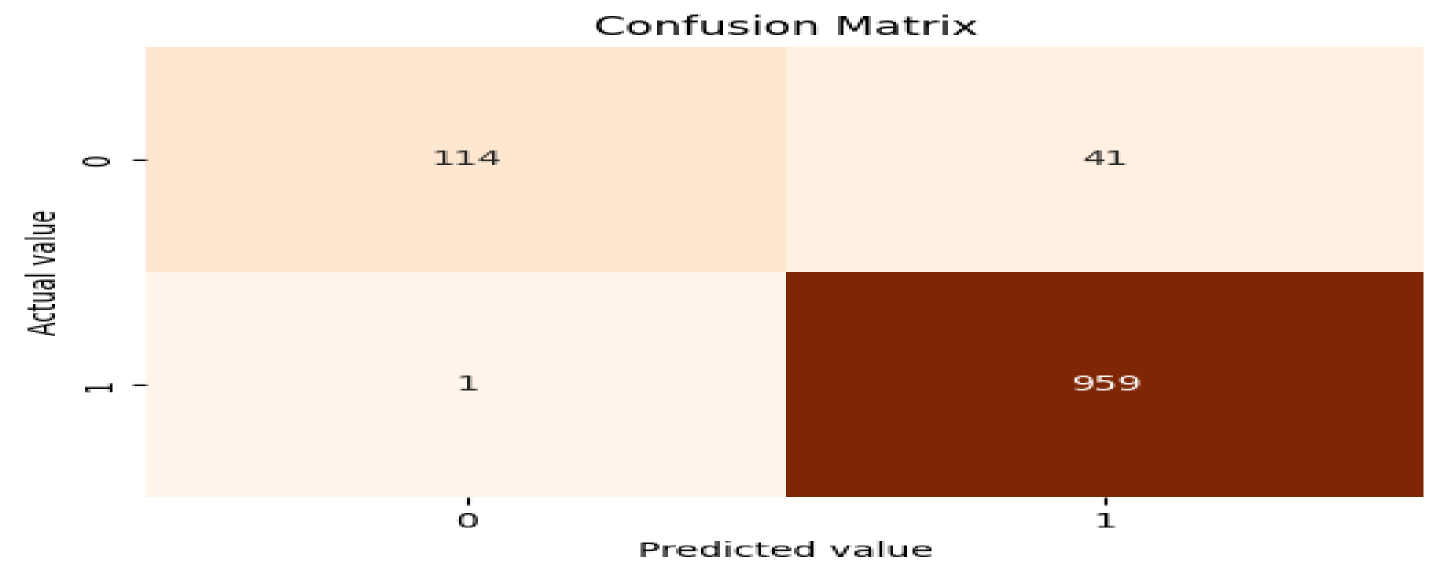
```
[1]
Ham Mail
```

Confusion Matrix

In [35]:

```
conf_matrix=confusion_matrix(Y_test,predict_test_data)
plt.figure(figsize=(6,4))
sns.heatmap(conf_matrix,annot=True,fmt="d",cmap="Oranges",cbar=False)
plt.xlabel("Predicted value")
plt.ylabel("Actual value")
plt.title("Confusion Matrix")
plt.show()
```

Out [35]:



In [36]:

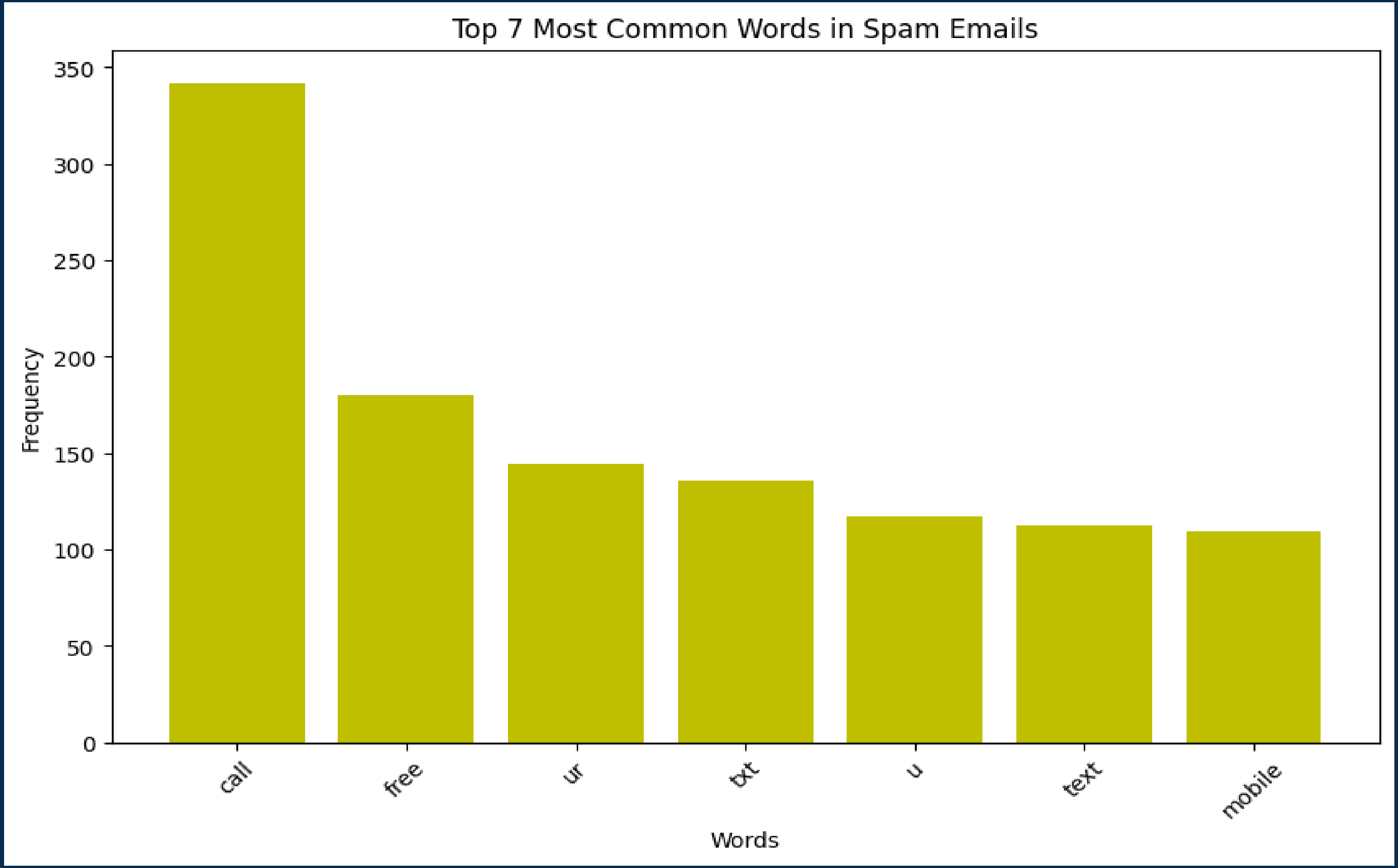
```
# Data visualization - Top 7 Most Common Words in Spam Emails

stop_words = set(stopwords.words('english'))
spam_words = " ".join(df[df['Category'] == 0]['Message']).split()
ham_words = " ".join(df[df['Category'] == 1]['Message']).split()

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

plt.figure(figsize=(10, 6))
plt.bar(*zip(*spam_word_freq.most_common(7)), color='y')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Top 7 Most Common Words in Spam Emails')
plt.xticks(rotation=45)
plt.show()
```

Out [36]:



Top 7 Most Common Words in Spam Emails
Frequency
350
300
250
200
150
100
50
0
call
free
ur
txt
u
text
mobile
Words

Conclusion:

The Development And Implementation Of A Spam Classifier Is A Crucial Step In Enhancing The Efficiency And Security Of Digital Communication. This Project Has Demonstrated The Effectiveness Of Machine Learning Algorithms In Distinguishing Between Legitimate Messages And Unsolicited Spam, Thereby Reducing The Risk Of Falling Victim To Phishing Scams, Malware, And Unwanted Advertisements.

The Classifier, Trained On A Diverse Dataset And Fine-tuned Through Iterative Testing And Optimization, Has Showcased Commendable Accuracy And Reliability. It Has Proven Capable Of Adapting To Evolving Spamming Techniques And Maintaining A Low False-positive Rate, Ensuring That Important Messages Are Not Inadvertently Labeled As Spam.

Thank you