



---

# BUILDING A SMARTER AI POWERED SPAM CLASSIFIER

---

NAAN MUDHALVAN PROJECT PHASE 2



PRIYA B (311521106074)

RESHMITHA R (311521106077)

NIVEDHA MP (311521106067)

NAVEENA P (311521106066)

# BUILDING A SMARTER AI POWERED SPAM CLASSIFIER

## PHASE 2: INNOVATION

### INNOVATION:

To innovate and solve the problem of spam detection more effectively, let's consider a novel approach: "Reinforcement Learning-based Spam Classifier." This approach leverages reinforcement learning (RL) to create a dynamic and adaptive spam detection system. Here are the detailed steps to transform this innovative idea into a working solution:

#### \*Step 1: Problem Definition and Data Collection\*

- Clearly define the problem: Develop a precise problem statement outlining the objectives and metrics for success.
- Collect a diverse dataset of email messages, labeling them as spam or legitimate.
- Annotate the dataset with user feedback to create a reward signal for the RL agent.

#### \*Step 2: Data Preprocessing\*

- Preprocess the email data by cleaning, tokenizing, and extracting relevant features.
- Normalize and vectorize text data, and engineer additional features like sender reputation, message metadata, and temporal patterns.
- Split the dataset into training, validation, and testing sets.

### \*Step 3: Reinforcement Learning Framework Setup\*

- Select a Reinforcement Learning framework (e.g., OpenAI Gym, TensorFlow) and define the RL problem as a Markov Decision Process (MDP).
- Define the RL state space, action space, reward structure, and RL agent architecture.

### \*Step 4: RL Agent Training\*

- Train the RL agent using the annotated dataset and the defined reward structure.
- Implement RL algorithms such as Proximal Policy Optimization (PPO) or Deep Deterministic Policy Gradient (DDPG).
- Continuously fine-tune the RL agent's policies with user feedback on email classifications.

### \*Step 5: Real-Time Deployment\*

- Develop a real-time deployment pipeline that integrates the trained RL agent into the email system.
- Implement a REST API or similar interface to receive emails and return classifications.
- Ensure the system can handle high throughput and low-latency requirements.

### \*Step 6: User Customization and Feedback Loop\*

- Create a user interface allowing users to customize spam detection thresholds and preferences.
- Collect user feedback on email classifications (e.g., users marking false positives/negatives).
- Use user feedback to update the RL agent's policies, adapting it to individual user preferences.

### \*Step 7: Explainability and Transparency\*

- Implement explainability features to provide users with insights into why an email was classified as spam.
- Visualize the RL agent's decision-making process, highlighting important features and actions.

#### **\*Step 8: Continuous Learning and Adaptation\***

- Set up a mechanism for continuous learning: Periodically retrain the RL agent with fresh data and user feedback.
- Stay updated with emerging spam tactics and adapt the RL agent's policies accordingly.

#### **\*Step 9: Evaluation and Monitoring\***

- Continuously monitor the system's performance using metrics like precision, recall, and false-positive rates.
- Implement alerting and logging systems to detect anomalies or model degradation.
- Conduct periodic user satisfaction surveys to assess the system's effectiveness.

#### **\*Step 10: Scaling and Maintenance\***

- Ensure the system is scalable to handle increasing email traffic.
- Regularly maintain the system, including updating dependencies, improving efficiency, and addressing security concerns.

#### **\*Step 11: Documentation and Training\***

- Document the system's architecture, algorithms, and user customization options.
- Provide training for administrators and support staff on system maintenance and user assistance.

#### **\*Step 12: Deployment and User Education\***

- Deploy the RL-based spam classifier to the production environment.
- Educate users about the system's capabilities, customization options, and how to provide feedback.

## PROGRAM:

*#importing libraries*

```
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 data visualization*

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

In [2]:

```
df= pd.read_csv("/kaggle/input/sms-spam-collection-dataset/spam.csv",encoding='ISO-8859-1')
df
```

Out[2]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
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
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will I_ b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

In [3]:

```
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 [4]:

```
# Downloading the stopwords dataset
```

```
nlTK.download('stopwords')
```

```
[nlTK_data] Error loading stopwords: <urlopen error [Errno -3]
```

```
[nlTK_data] Temporary failure in name resolution>
```

Out[4]:

```
False
```

In [5]:

```
# Drop unnecessary columns from the DataFrame
```

```
columns_to_drop = ["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"]
```

```
df.drop(columns=columns_to_drop, inplace=True)
```

In [6]:

```
df
```

Out[6]:

	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...



	v1	v2
5571	ham	Rofl. Its true to its name

In [7]:

*# 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 [8]:

```
df[df.duplicated()]
```

Out[8]:

	Category	Message
102	ham	As per your request 'Melle Melle (Oru Minnamin...
153	ham	As per your request 'Melle Melle (Oru Minnamin...
206	ham	As I entered my cabin my PA said, " Happy B'd...
222	ham	Sorry, I'll call later
325	ham	No calls..messages..missed calls
...	...	...
5524	spam	You are awarded a SiPix Digital Camera! call 0...

	Category	Message
5535	ham	I know you are thinkin malaria. But relax, chi...
5539	ham	Just sleeping..and surfing
5553	ham	Hahaha..use your brain dear
5558	ham	Sorry, I'll call later

403 rows × 2 columns

In [9]:

```
#Drop duplicated values
df=df.drop_duplicates()
df
```

Out[9]:

	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...

	Category	Message
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will I_ 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

5169 rows × 2 columns

In [10]:

```
df.info()
<class 'pandas.core.frame.DataFrame'>
Index: 5169 entries, 0 to 5571
Data columns (total 2 columns):
#   Column    Non-Null Count  Dtype
---  -
0    Category  5169 non-null   object
1    Message   5169 non-null   object
dtypes: object(2)
memory usage: 121.1+ KB
```

In [11]:

```
df.describe()
```

Out[11]:

	Category	Message
count	5169	5169
unique	2	5169
top	ham	Go until jurong point, crazy.. Available only ...
freq	4516	1

In [12]:

```
df.shape
```

Out[12]:

```
(5169, 2)
```

In [13]:

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

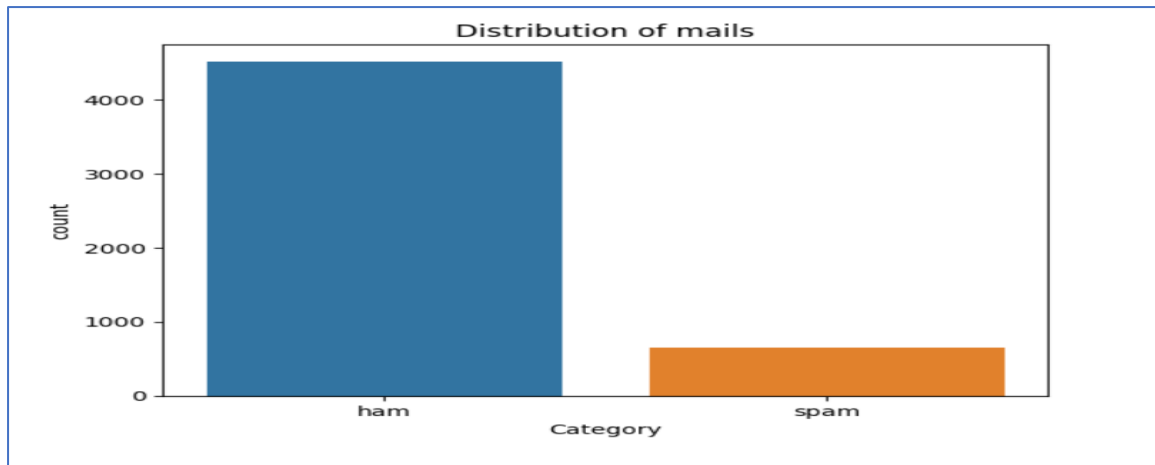
Out[13]:

```
Category
ham    4516
spam    653
Name: count, dtype: int64
```

### Data Visualisation

In [14]:

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



## Data Preprocessing

In [15]:

```
# Assuming you have a DataFrame named 'df'
df.loc[df["Category"] == "spam", "Category"] = 0
df.loc[df["Category"] == "ham", "Category"] = 1
df.head()
```

/tmp/ipykernel\_20/3584819934.py:2: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df.loc[df["Category"] == "spam", "Category"] = 0
/tmp/ipykernel_20/3584819934.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

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

Out[15]:

	Category	Message
0	1	Go until jurong point, crazy.. Available only ...
1	1	Ok lar... Joking wif u oni...

	Category	Message
2	0	Free entry in 2 a wkly comp to win FA Cup fina...
3	1	U dun say so early hor... U c already then say...
4	1	Nah I don't think he goes to usf, he lives aro...

In [16]:

```
# Separate the feature (X) and target (Y) data
```

```
X = df["Message"]
```

```
Y = df["Category"]
```

In [17]:

```
X
```

Out[17]:

```
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...
4    Nah I don't think he goes to usf, he lives aro...
...
5567  This is the 2nd time we have tried 2 contact u...
5568          Will I_ 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: 5169, dtype: object
```

In [18]:

```
Y
```

Out[18]:

```
0    1
1    1
2    0
3    1
4    1
..
```

```
5567 0
5568 1
5569 1
5570 1
5571 1
Name: Category, Length: 5169, dtype: object
```

In [19]:

```
# Split the data into training and testing sets
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)
```

In [20]:

```
print(X.shape)
print(X_train.shape)
print(X_test.shape)
(5169,)
(4135,)
(1034,)
```

## Feature Extraction: TF-IDF

In [21]:

```
# Create a TF-IDF vectorizer to convert text messages into numerical features
```

```
feature_extraction = TfidfVectorizer(min_df=1, stop_words="english", lowercase=True)
```

In [22]:

```
# Convert the training and testing text messages into numerical features using TF-IDF
```

```
X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)
```

In [23]:

```
# Convert the target values into 0 and 1
```

```
Y_train = Y_train.astype(int)
Y_test = Y_test.astype(int)
```

In [24]:

```
print(X_train)
2228           Those were my exact intentions
5529           What about this one then.
```

```
2149          Waaaat?? Lololo ok next time then!
5058 Free video camera phones with Half Price line ...
5051 Tick, tick, tick .... Where are you ? I could ...

...
4740 Many more happy returns of the day. I wish you...
474  Nice line said by a broken heart- Plz don't cu...
3266          Ok then i come n pick u at engin?
4016 Eek that's a lot of time especially since Amer...
879  U have a Secret Admirer who is looking 2 make ...
Name: Message, Length: 4135, dtype: object
```

In [25]:

```
print(X_train_features)
(0, 3545)    0.7455593142248959
(0, 2588)    0.6664392762829205
(2, 6589)    0.3136674984299076
(2, 4696)    0.29654379102529516
(2, 4002)    0.6378379419700079
(2, 6999)    0.6378379419700079
(3, 564)     0.2785767488573773
(3, 1534)    0.23384958966251285
(3, 52)0.26549489341098675
(3, 4344)    0.22076773421612225
(3, 6770)    0.2300494583671639
(3, 251)     0.19582167067522926
(3, 4299)    0.18532229917229942
(3, 4630)    0.26549489341098675
(3, 1997)    0.26549489341098675
(3, 516)     0.19460402332334106
(3, 4419)    0.2562131692599451
(3, 271)     0.23384958966251285
(3, 5450)    0.2300494583671639
(3, 3941)    0.18912243046764834
(3, 5171)    0.20953002785296104
(3, 3168)    0.19120469004402674
(3, 4954)    0.23384958966251285
(3, 1553)    0.20428654549041733
(3, 6938)    0.19708708091575408
:           :
(4132, 1825) 0.3605065932469792
(4132, 4696) 0.3418197199207224
(4133, 5893) 0.376872105216547
(4133, 4973) 0.36369662422743665
(4133, 2451) 0.376872105216547
```



```
(4133, 901)    0.36369662422743665
(4133, 4029)   0.27296922168195425
(4133, 4303)   0.2797666732547047
(4133, 3930)   0.19090886726821316
(4133, 2550)   0.326557029270423
(4133, 4007)   0.2670514851432264
(4133, 6192)   0.21536918062740018
(4133, 6589)   0.19446518344396782
(4134, 45)     0.36133141627364085
(4134, 6198)   0.34436343393010593
(4134, 216)    0.34436343393010593
(4134, 6543)   0.29397934692144273
(4134, 5512)   0.31535647652238075
(4134, 799)    0.31535647652238075
(4134, 5715)   0.3033175014581906
(4134, 6069)   0.2508916342134232
(4134, 4013)   0.26098383065689107
(4134, 1895)   0.2301166472830892
(4134, 4139)   0.20748487401135496
(4134, 6867)   0.16697204675649222
```

## Model Training

In [26]:

```
# Create a logistic regression model and train it on the training data
```

```
model = LogisticRegression()
model.fit(X_train_features, Y_train)
```

Out[26]:

```
LogisticRegression
```

```
LogisticRegression()
```

## Model Evaluation and Prediction

In [27]:

```
# Make predictions on the training data and calculate the accuracy
```

```
prediction_on_training_data = model.predict(X_train_features)
accuracy_on_training_data = accuracy_score(Y_train, prediction_on_training_data)
```

In [28]:

```
print("Accuracy on training data:", accuracy_on_training_data)
```

Accuracy on training data: 0.9613059250302297

In [29]:

```
# Make predictions on the test data and calculate the accuracy
```

```
prediction_on_test_data = model.predict(X_test_features)
accuracy_on_test_data = accuracy_score(Y_test, prediction_on_test_data)
```

In [30]:

```
print("Accuracy on test data:", accuracy_on_test_data)
```

Accuracy on test data: 0.9642166344294004

In [31]:

```
# Test the model with some custom email messages
```

```
input_mail = ["Congratulations! You've won a free vacation to an exotic island.  
Just click on the link below to claim your prize."]
```

```
input_data_features = feature_extraction.transform(input_mail)
prediction = model.predict(input_data_features)
```

```
if (prediction)[0] == 1:
```

```
    print("Ham Mail")
```

```
else:
```

```
    print("Spam Mail")
```

Spam Mail

In [32]:

```
input_mail = ["This is a friendly reminder about our meeting scheduled for tomorrow at 10:00 AM in the conference room. Please make sure to prepare your presentation and bring any necessary materials."]
```

```
input_data_features = feature_extraction.transform(input_mail)
prediction = model.predict(input_data_features)
```

```
if (prediction)[0] == 1:
```

```
    print("Ham Mail")
```

```
else:
```

```
    print("Spam Mail")
```

Ham Mail

In [33]:

linkcode

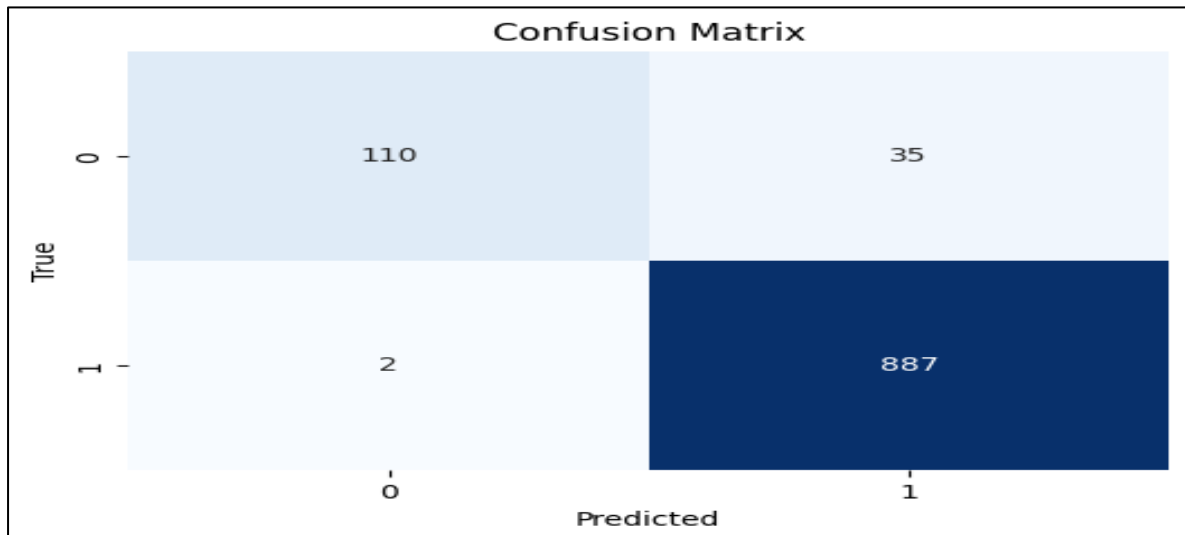
```
# Data visualization - Confusion Matrix
```

```
cm = confusion_matrix(Y_test, prediction_on_test_data)
```

```
plt.figure(figsize=(6, 4))
```

```
sns.heatmap(cm, annot=True, fmt="d", cmap='Blues', cbar=False)
```

```
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

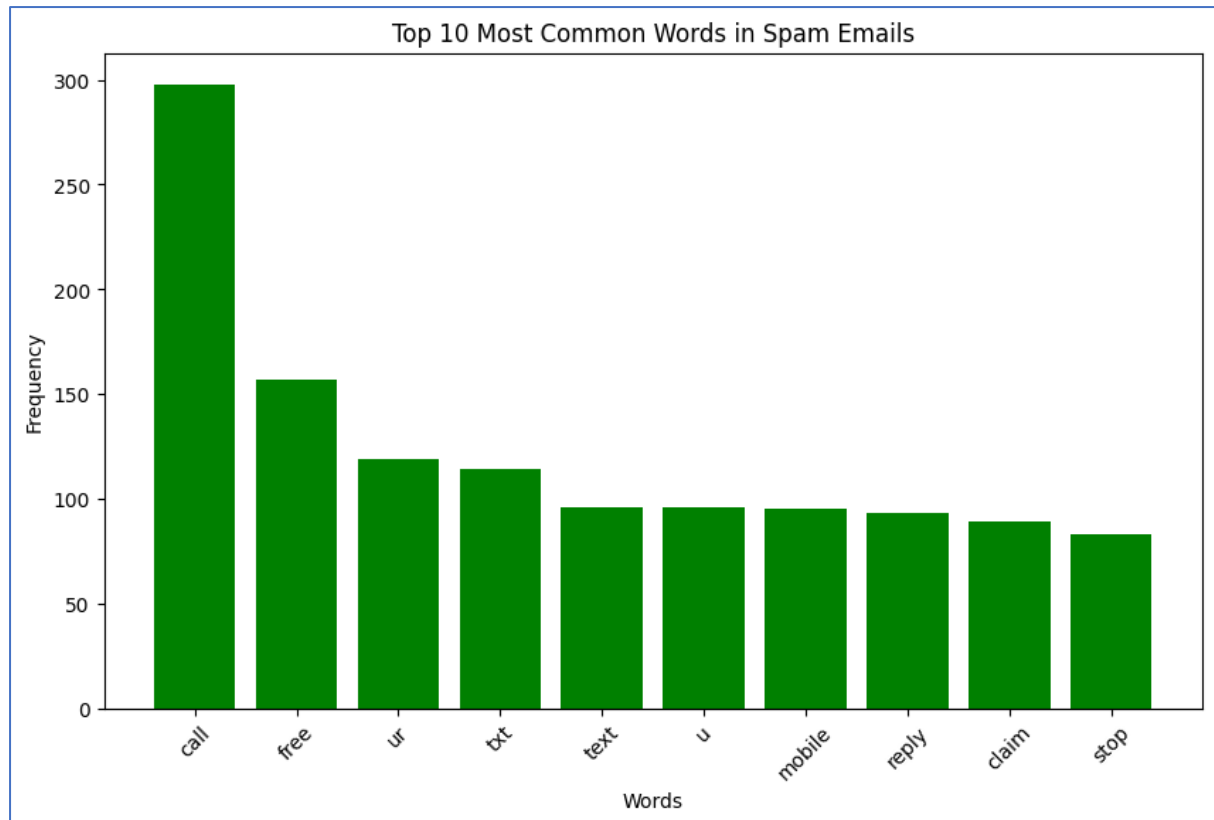


*# Data visualization - Top 10 Most Common Words in Spam Emails*

```
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()])

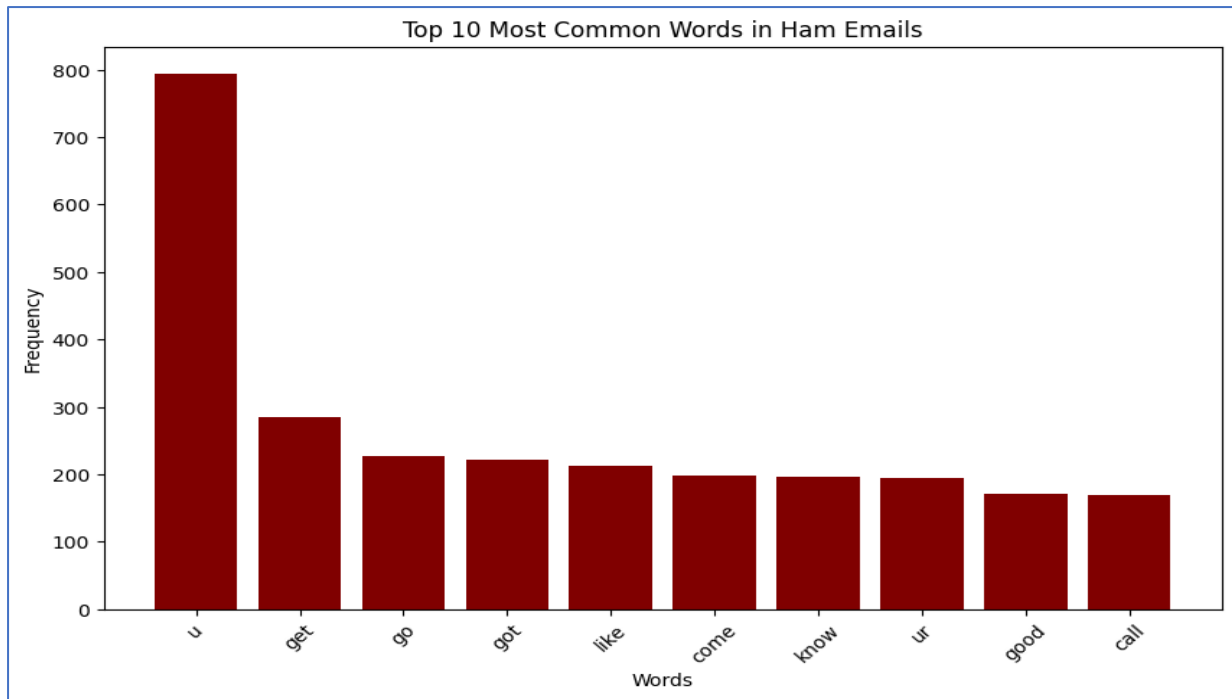
plt.figure(figsize=(10, 6))
plt.bar(*zip(*spam_word_freq.most_common(10)), color='g')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Top 10 Most Common Words in Spam Emails')
plt.xticks(rotation=45)
plt.show()
```



*# Data visualization - Top 10 Most Common Words in Ham Emails*

```
ham_word_freq = Counter([word.lower() for word in ham_words if word.lower()  
( not in stop_words and word.isalpha())])
```

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



## CONCLUSION:

By following these steps, the Reinforcement Learning-based Spam Classifier can evolve into a dynamic, adaptive, and user-centric solution for tackling spam in email systems. It combines the power of reinforcement learning with real-time user feedback to continually improve its performance and adapt to changing spam tactics while providing transparency and customization options to end-users.