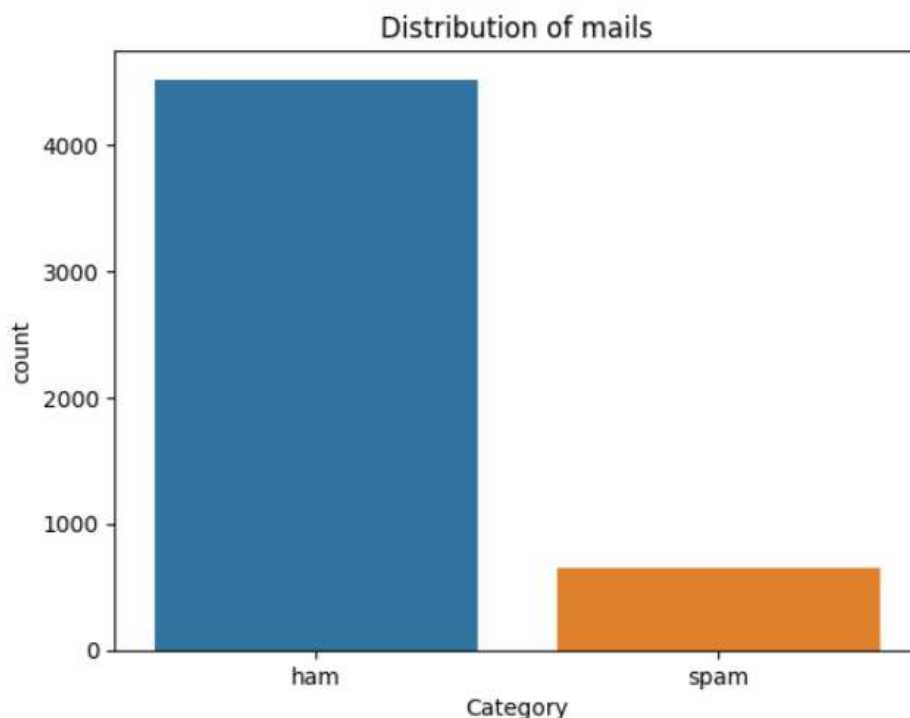


▼ Data Visualisation

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



▼ Data Preprocessing

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

<ipython-input-18-860e27392fb1>: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.

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

<ipython-input-18-860e27392fb1>: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.

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

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

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

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

```
[ ] X
```

```
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
```

```
[ ] Y
```

```
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
```

```
[ ] # 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)
```

```
[ ] print(X.shape)
print(X_train.shape)
print(X_test.shape)
```

```
(5169,)
(4135,)
(1034,)
```

▼ Feature Extraction: TF-IDF

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

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

```
[ ] # 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)
```

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

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

```
[ ] 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
```

```
[ ] 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
```

```
[ ] 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

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

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

```
• LogisticRegression
LogisticRegression()
```



```
LogisticRegression()
```

▼ Model Evaluation and Prediction

```
[ ] # 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)
```

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

Accuracy on training data: 0.9613059250302297
```

```
[ ] # 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)
```

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

Accuracy on test data: 0.9642166344294004
```

```
[ ] # 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 to claim your prize. Don't miss out!"]
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

```
[ ] input_mail = ["This is a friendly reminder about our meeting scheduled for tomorrow at 10:00 AM in the conference room. Please be on time."]
input_data_features = feature_extraction.transform(input_mail)
prediction = model.predict(input_data_features)

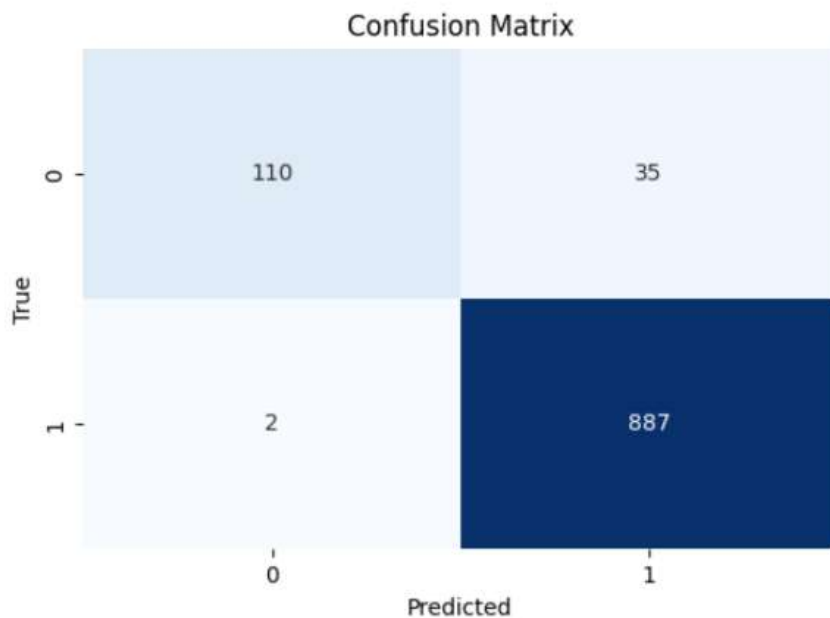
if (prediction)[0] == 1:
    print("Ham Mail")
else:
    print("Spam Mail")
```

Ham Mail

```
[ ] # 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()
```

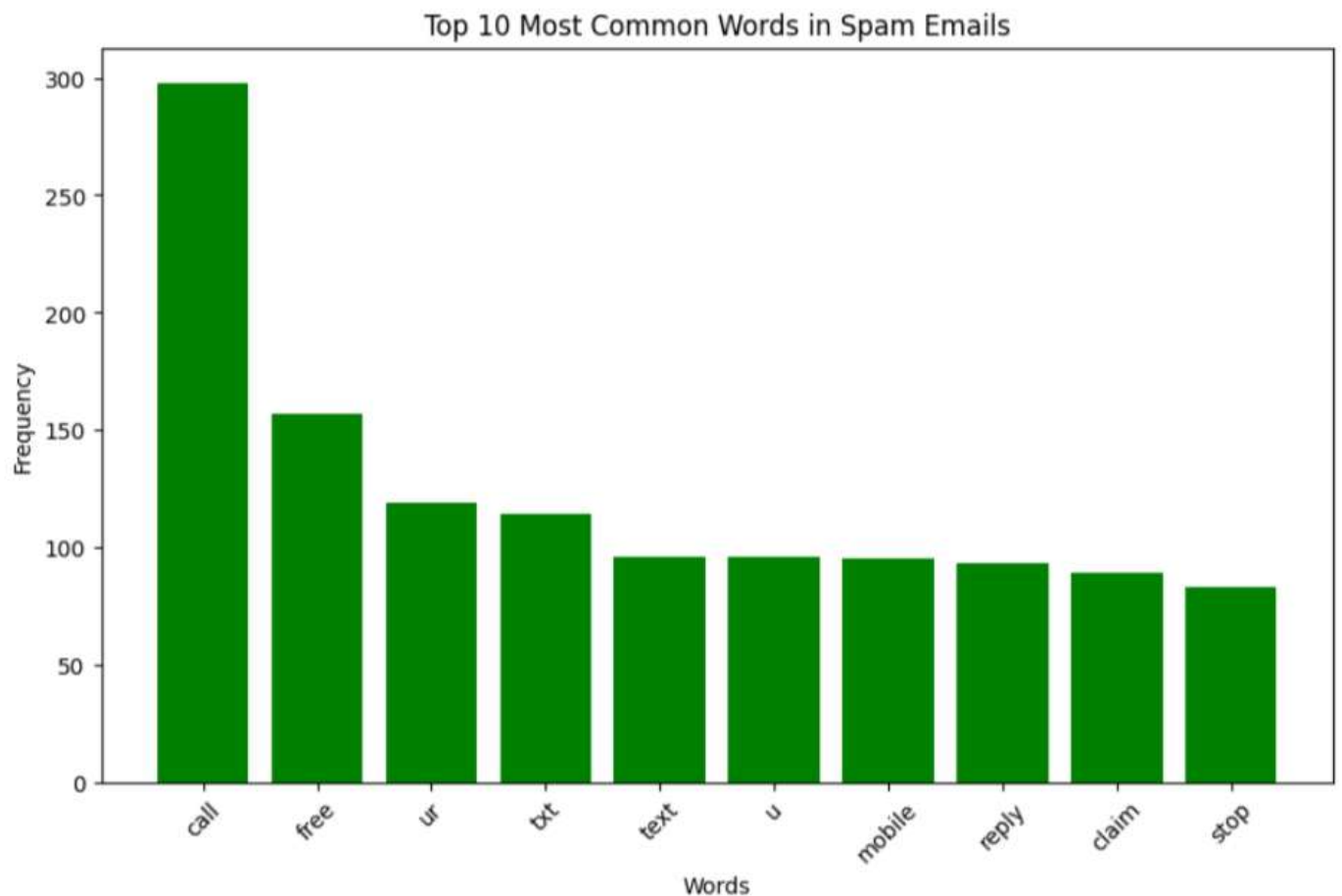


```
[ ] # Data visualization - Top 10 Most Common Words in Spam Emails
```

```
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 w

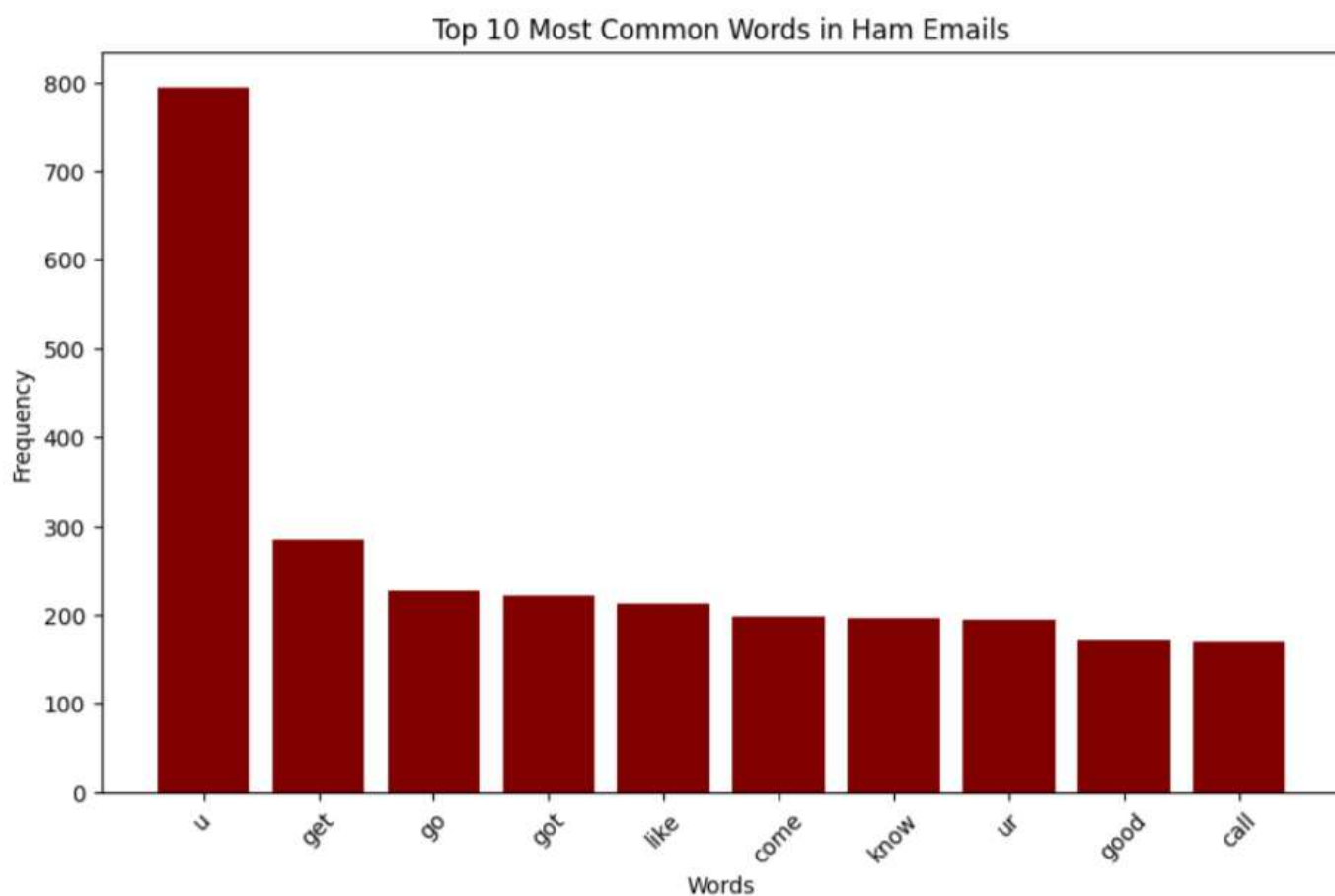
plt.figure(figsize=(10, 6))
plt.bar(*zip(*spam_word_freq.most_common(10)), color='g')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Top 10 Most Common Words in Spam Emails')
plt.xticks(rotation=45)
plt.show()
```



```
[ ] # 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])

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()
```



```
[ ]
```