॥वसुधैव कुटुम्बकम्॥

SYMBIOSIS

INSTITUTE OF COMPUTER STUDIES AND RESEARCH

MACHINE LEARNING CASE STUDY PROJECT ON EMAIL AUTOMATION AND SPAM FILTERING

From,

Aditya Parameswaran(23030141**002**)

Apeksha Chikane(23030141**014**)

Pratik Deshmukh(23030141**018**)

MBA-IT(Div-A)

SICSR(2023-2025**)**

To,

Dr. Farhana Desai,

AIML,

SICSR

Table of Contents:

Objective:

The objective of this project is to develop an email automation and filtering system using machine learning. The system aims to:

Classify Emails: Accurately categorize emails as spam or ham.

Automate Handling: Automatically move spam emails to the spam folder and notify users.

Analyze Content: Extract and visualize key features and patterns in email content.

Integrate with Email Services: Securely connect with email providers to automate email processing.

Notify Users: Inform users about email classifications and provide actionable insights.

Evaluate and Improve: Continuously assess and enhance the model's performance using relevant metrics.

This will improve email management, enhance security, and provide a better user experience.

Algorithm Used: Naïve Bayes Algorithm

Naive Bayes classifiers are widely used for email automation and spam filtering due to several key advantages that make them particularly well-suited for these tasks. Here are the reasons why Naive Bayes is commonly used:

1. Simplicity and Ease of Implementation:

Naive Bayes classifiers are simple to understand and implement. This simplicity makes them a popular choice for initial models in email filtering systems. Despite their simplicity, they often provide strong baseline performance.

2. Probabilistic Model:

Naive Bayes is a probabilistic classifier based on Bayes' Theorem. It calculates the probability that a given email belongs to a particular class (spam or not spam) based on the words in the email. This probabilistic approach is effective in handling the inherent uncertainties in text data.

3. Effectiveness in Text Classification:

Naive Bayes has been shown to perform well on text classification tasks, including spam detection. It is particularly effective when the dataset has a large number of features (words) and when the dataset is large.

4. **Unaffected to Irrelevant Features:**

Naive Bayes is relatively unaffected to irrelevant features. Even if some of the words in an email are not useful for distinguishing spam from non-spam, Naive Bayes can still perform well. This robustness is important in email filtering where not all words are equally informative.

5. **Efficiency and Scalability:**

Naive Bayes classifiers are computationally efficient both in terms of training and prediction. They require relatively little training data and can handle large datasets efficiently, making them suitable for real-time email filtering.

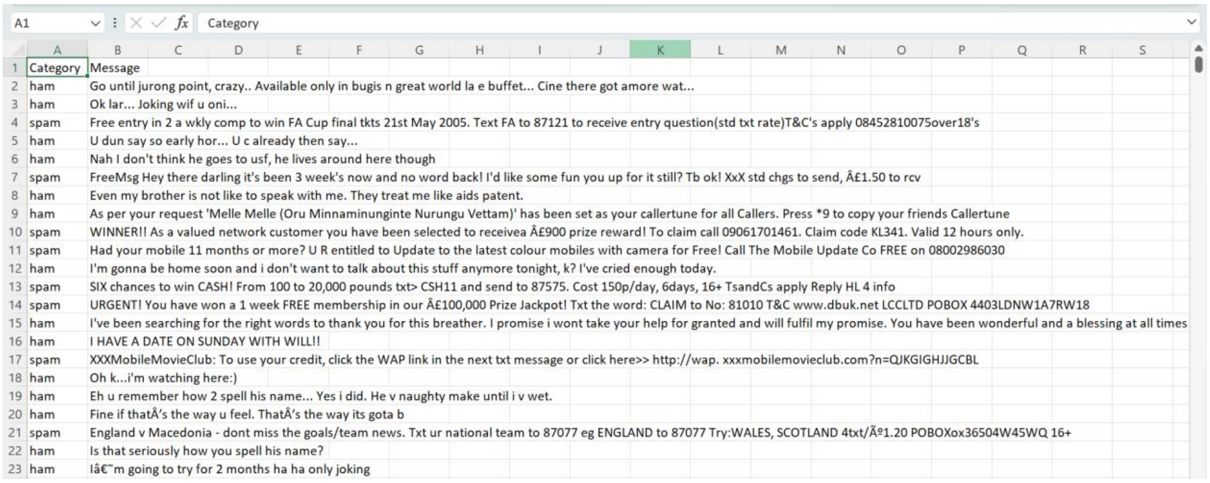
Colab link:
https://colab.research.google.com/drive/102IzlU7L4UzBaBrg0apxF5eV3eL6iXBp#scrollTo=Od9UcCY-4dBp&uniqifier=1

Dataset : https://www.kaggle.com/datasets/phangud/spamcsv

Prerequisites for Building Email Automation System

1. About this Dataset:

The "spam.csv" dataset is primarily used for spam email detection, It contains labelled text messages, including emails, categorized into "ham" (legitimate/non-spam) and "spam" (unsolicited and potentially harmful) messages

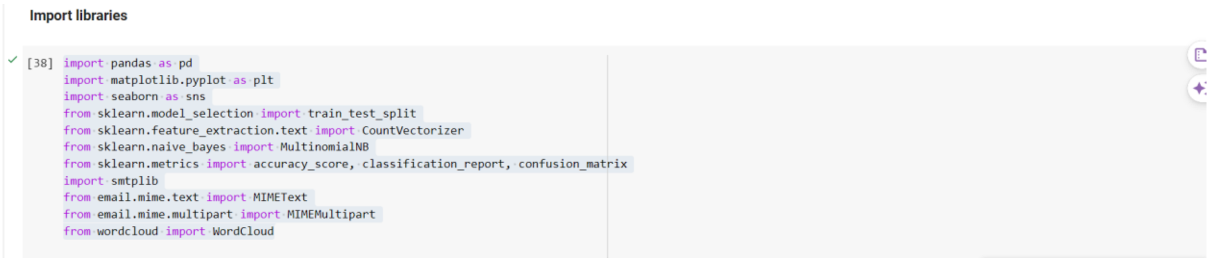


A1 | Category

	A	B
1	Category	Message
2	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
3	ham	Ok lar... Joking wif u oni...
4	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
5	ham	U dun say so early hor... U c already then say...
6	ham	Nah I don't think he goes to usf, he lives around here though
7	spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, Â£1.50 to rcv
8	ham	Even my brother is not like to speak with me. They treat me like aids patent.
9	ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
10	spam	WINNER!! As a valued network customer you have been selected to receivea Â£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
11	spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030
12	ham	I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.
13	spam	SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info
14	spam	URGENT! You have won a 1 week FREE membership in our Â£100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD POBOX 4403LDNW1A7RW18
15	ham	I've been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulfil my promise. You have been wonderful and a blessing at all times
16	ham	I HAVE A DATE ON SUNDAY WITH WILL!!
17	spam	XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap. xxxmobilemovieclub.com?n=QJKGIGHJJGCBL
18	ham	Oh k...i'm watching here:)
19	ham	Eh u remember how 2 spell his name... Yes i did. He v naughty make until i v wet.
20	ham	Fine if thatÂ's the way u feel. ThatÂ's the way its gota b
21	spam	England v Macedonia - dont miss the goals/team news. Txt ur national team to 87077 eg ENGLAND to 87077 Try:WALES, SCOTLAND 4txt/Ã º1.20 POBOXox36504W45WQ 16+
22	ham	Is that seriously how you spell his name?
23	ham	Iâ€˜m going to try for 2 months ha ha only joking

Total row count : 5573

2.Import Libraries:



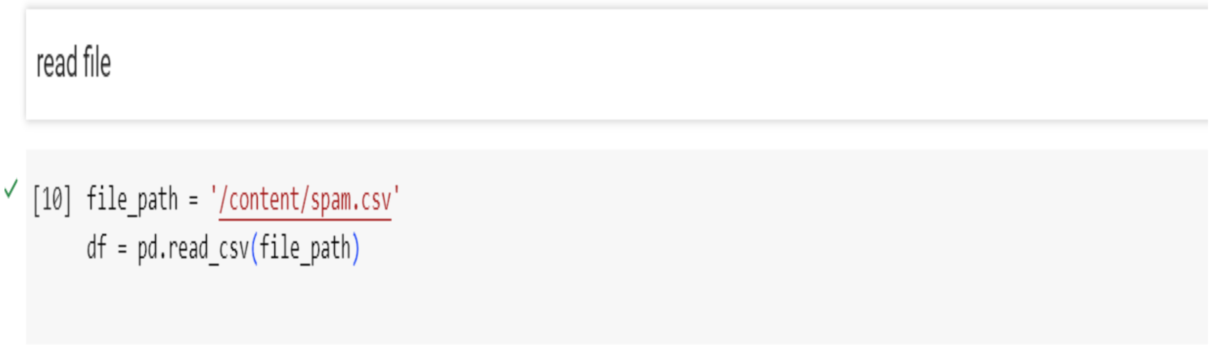
Import libraries

```
[38] import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     from sklearn.feature_extraction.text import CountVectorizer
     from sklearn.naive_bayes import MultinomialNB
     from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
     import smtplib
     from email.mime.text import MIMEText
     from email.mime.multipart import MIMEMultipart
     from wordcloud import WordCloud
```

The following libraries have to be imported to ensure that the entire program is run smoothly:

pandas: Data manipulation and analysis.
matplotlib.pyplot: Plotting and visualizations.
seaborn: Enhanced statistical visualizations.
sklearn.model_selection.train_test_split: Splitting data into train/test sets.
sklearn.feature_extraction.text.CountVectorizer: Converting text to numerical data.
sklearn.naive_bayes.MultinomialNB: Naive Bayes classifier for text data.
sklearn.metrics.accuracy_score, classification_report, confusion_matrix: Evaluating model performance.
smtplib: Sending emails via SMTP.
email.mime.text.MIMEText, email.mime.multipart.MIMEMultipart: Creating email messages.
wordcloud: Generating word cloud visualizations.

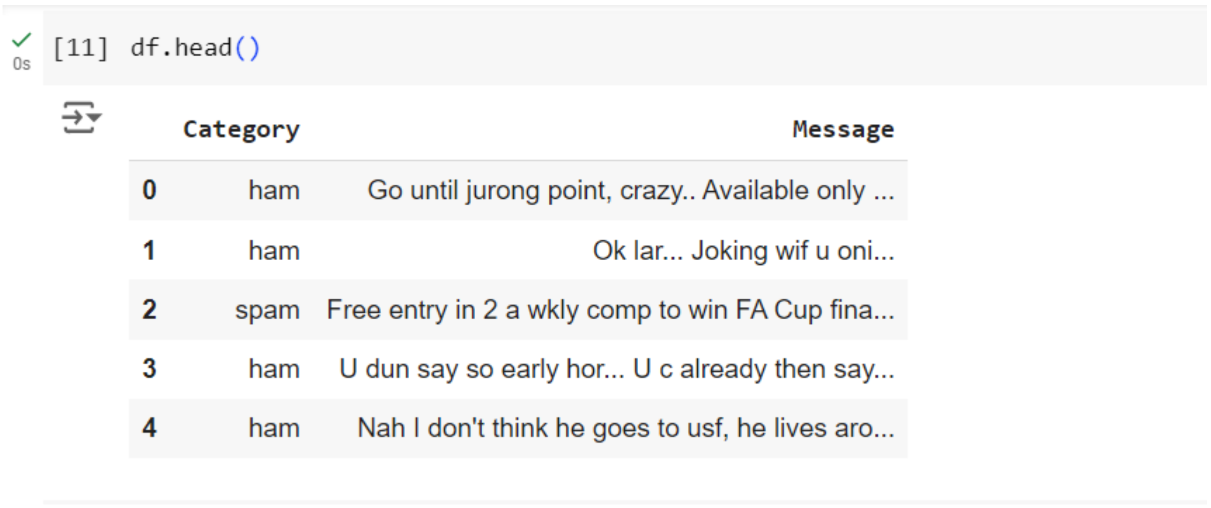
Read file:



read file

```
[10] file_path = '/content/spam.csv'
     df = pd.read_csv(file_path)
```

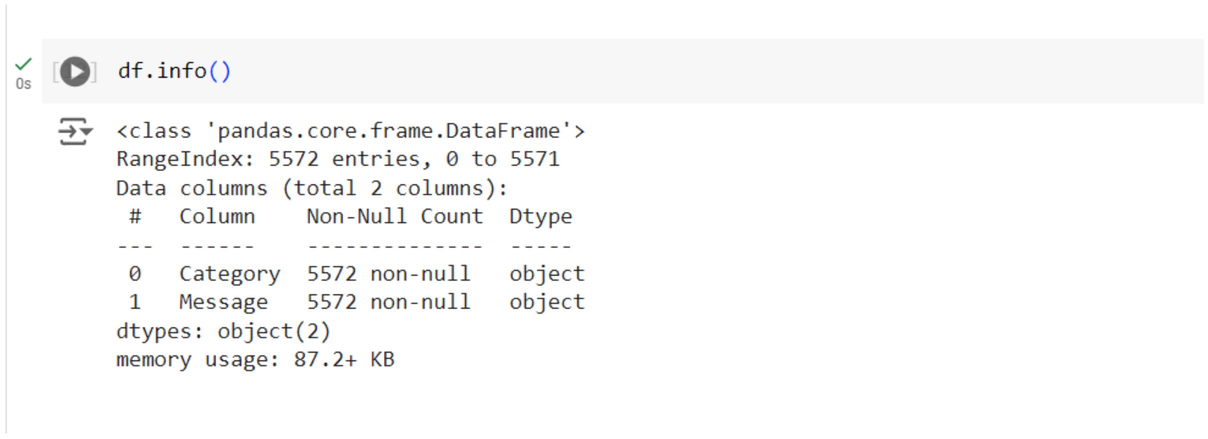
Display first 5 records:



```
[11] df.head()
```

	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...

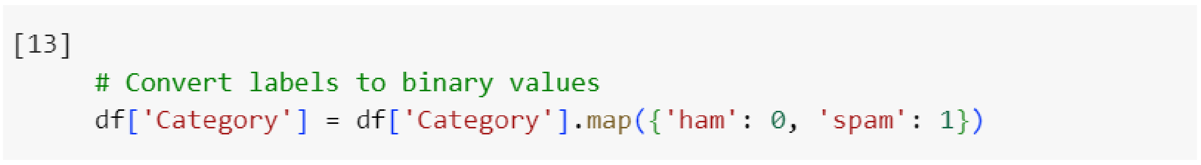
Information of the file:



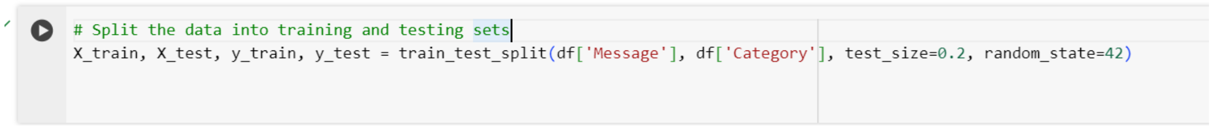
```
df.info()
```

```
<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
```

3. Preprocess the text data:

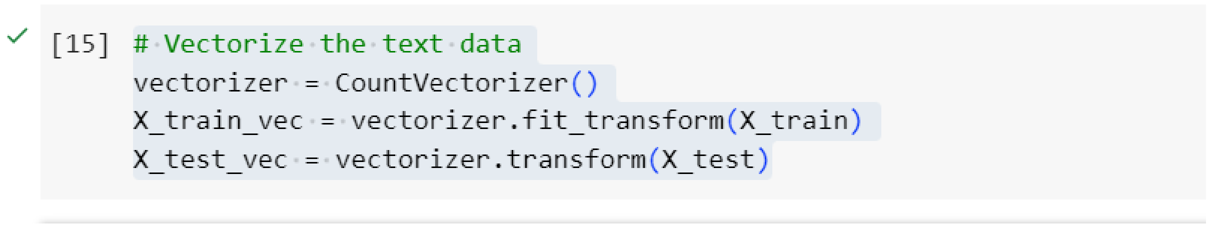


```
[13]
      # Convert labels to binary values
      df['Category'] = df['Category'].map({'ham': 0, 'spam': 1})
```



```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df['Message'], df['Category'], test_size=0.2, random_state=42)
```

It splits the dataset into training and testing sets, where 80% of the data is used for training the model and 20% is used for testing it, ensuring reproducibility with a fixed random state of 42.

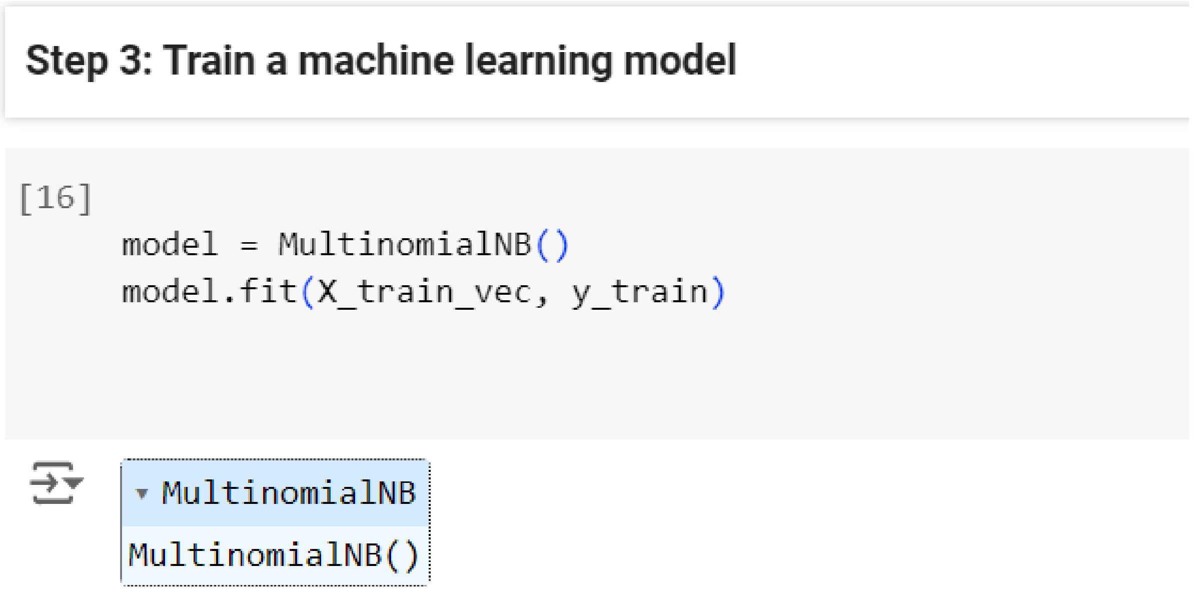


```
[15] # Vectorize the text data
     vectorizer = CountVectorizer()
     X_train_vec = vectorizer.fit_transform(X_train)
     X_test_vec = vectorizer.transform(X_test)
```

- **`vectorizer = CountVectorizer()`**: Initializes a `CountVectorizer` object to convert text data into numerical features based on word counts.
- **`X_train_vec = vectorizer.fit_transform(X_train)`**: Fits the vectorizer to the training data and transforms the text into a matrix of token counts.
- **`X_test_vec = vectorizer.transform(X_test)`**: Transforms the test data into a matrix of token counts using the fitted vectorizer, without fitting again.

<u>How to Build a Automation System Next Steps:</u>

<u>Train a Machine Learning Model using Naive Bayes:</u>



Step 3: Train a machine learning model

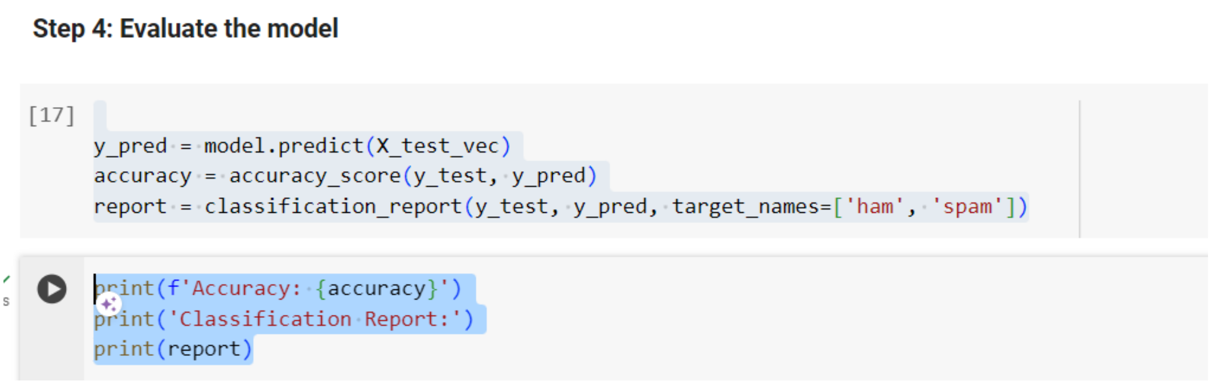
```
[16]
     model = MultinomialNB()
     model.fit(X_train_vec, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

`model = MultinomialNB()`: Initializes the Naive Bayes classifier.

`model.fit(X_train_vec, y_train)`: Trains the model on the training data.

Evaluate the Model:



Step 4: Evaluate the model

```
[17]
y_pred = model.predict(X_test_vec)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=['ham', 'spam'])
```

```
print(f'Accuracy: {accuracy}')
print('Classification Report:')
print(report)
```

`y_pred = model.predict(X_test_vec)`: Makes predictions on the test data.

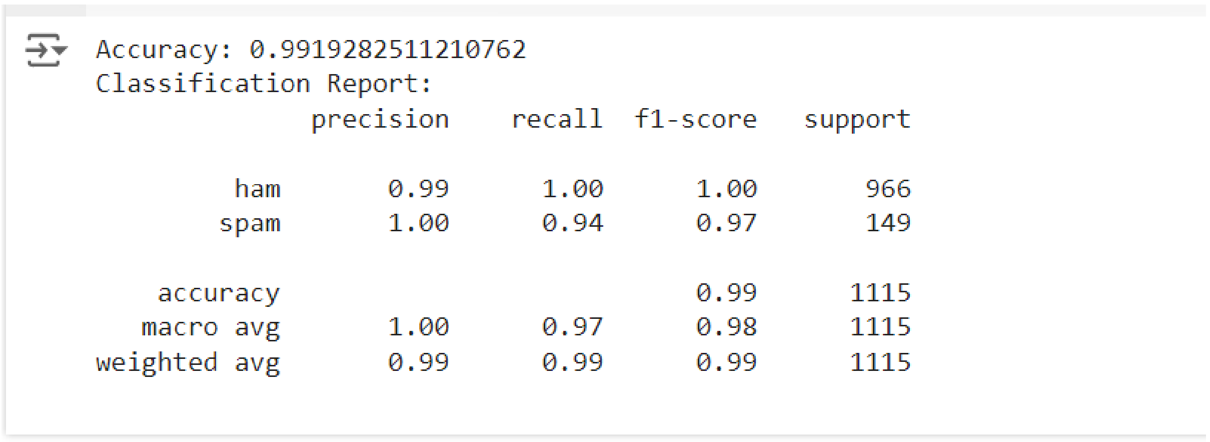
`accuracy = accuracy_score(y_test, y_pred)`: Computes the accuracy.

`report = classification_report(y_test, y_pred, target_names=['ham', 'spam'])`: Generates a classification report.

`print(f'Accuracy: {accuracy}')`: Prints the accuracy.

`print('Classification Report:')`: Prints the classification report.

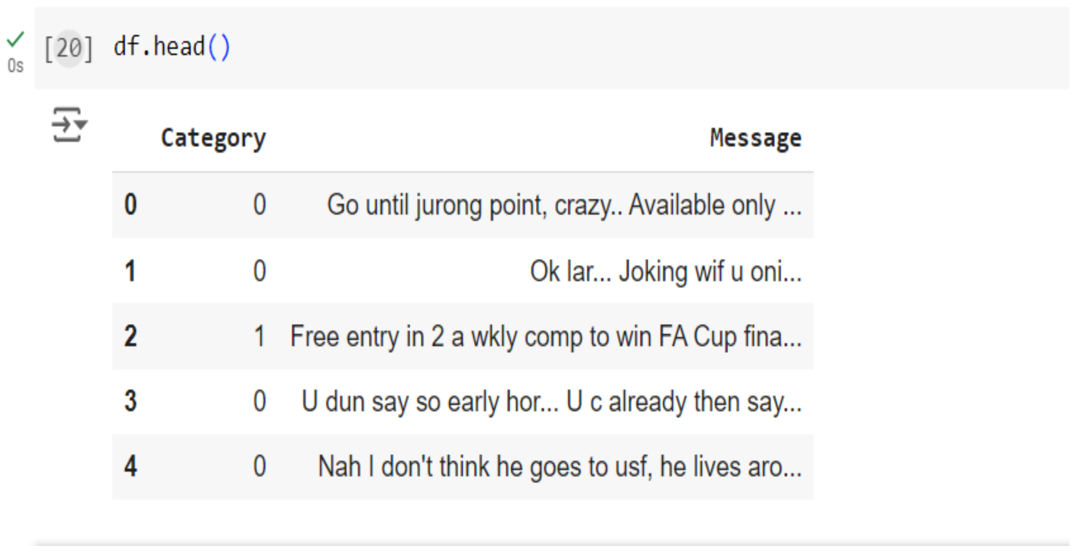
Output:



```
Accuracy: 0.9919282511210762
Classification Report:
              precision    recall  f1-score   support

         ham       0.99      1.00      1.00       966
        spam       1.00      0.94      0.97       149

    accuracy                           0.99      1115
   macro avg       1.00      0.97      0.98      1115
weighted avg       0.99      0.99      0.99      1115
```

The model is highly effective with 99.19% accuracy. It has perfect precision for spam (100%) and nearly perfect precision for ham (99%). The recall is 100% for ham and 94% for spam, indicating that while it misses a few spam emails, it performs excellently overall. The F1-scores are strong for both categories, showing a good balance between precision and recall.

After preprocessing first five records:

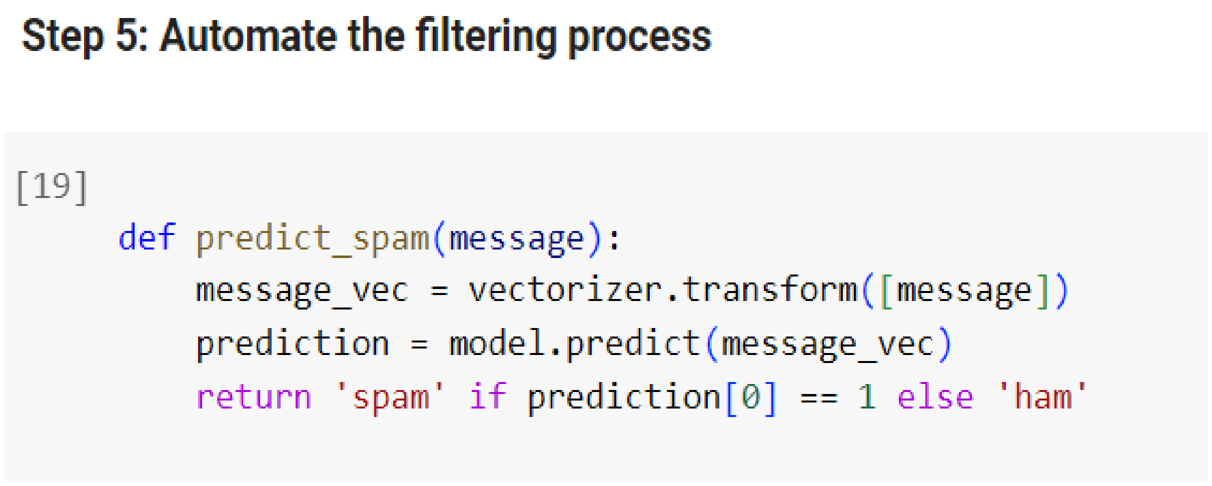


```
[20] df.head()
```

	Category	Message
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
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 aro...

Ham and spam converted to 0 and 1

Automate the Filtering Process:



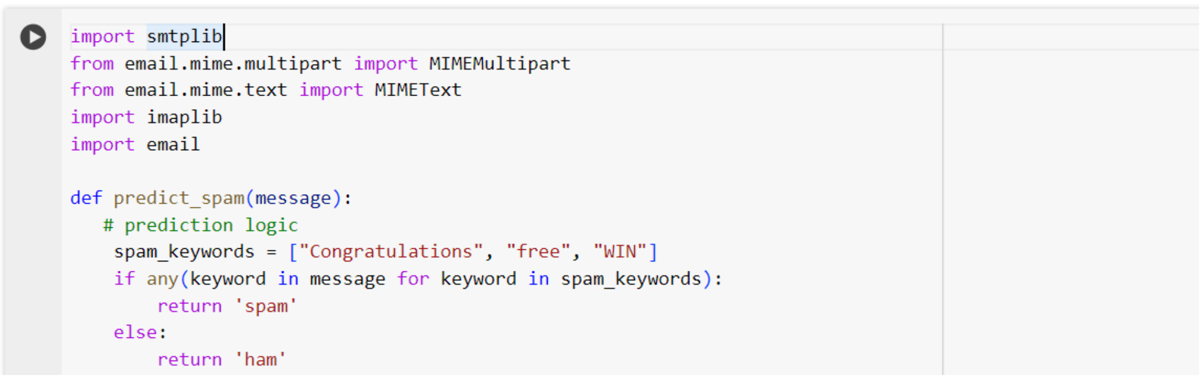
Step 5: Automate the filtering process

```
[19]
def predict_spam(message):
    message_vec = vectorizer.transform([message])
    prediction = model.predict(message_vec)
    return 'spam' if prediction[0] == 1 else 'ham'
```

- **Transform the Message**: `message_vec = vectorizer.transform([message])` converts the input message into the same numerical format used for training.
- **Predict**: `prediction = model.predict(message_vec)` classifies the message using the trained model.

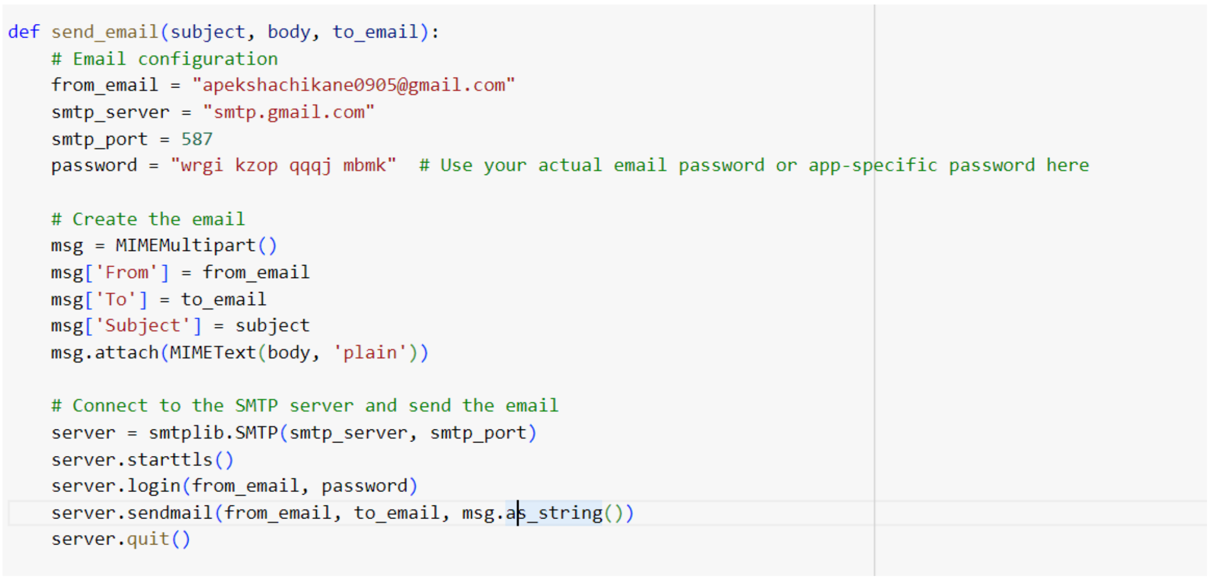
- **Return Result**: `return 'spam' if prediction[0] == 1 else 'ham'` returns 'spam' if the prediction is 1, otherwise 'ham'.

Email Automation send mail to Spam folder:



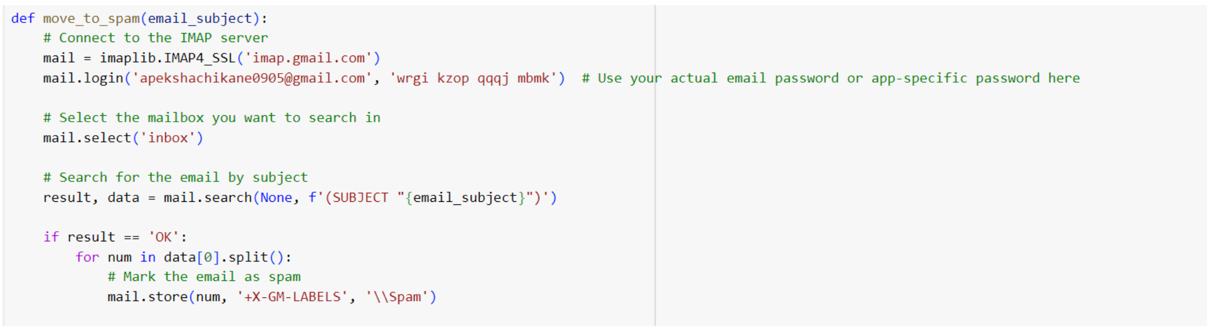
```python
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import imaplib
import email

def predict_spam(message):
    # prediction logic
     spam_keywords = ["Congratulations", "free", "WIN"]
     if any(keyword in message for keyword in spam_keywords):
         return 'spam'
     else:
         return 'ham'
```

Spam Prediction Function: `predict_spam` checks if the message contains specific spam keywords and returns 'spam' or 'ham'.



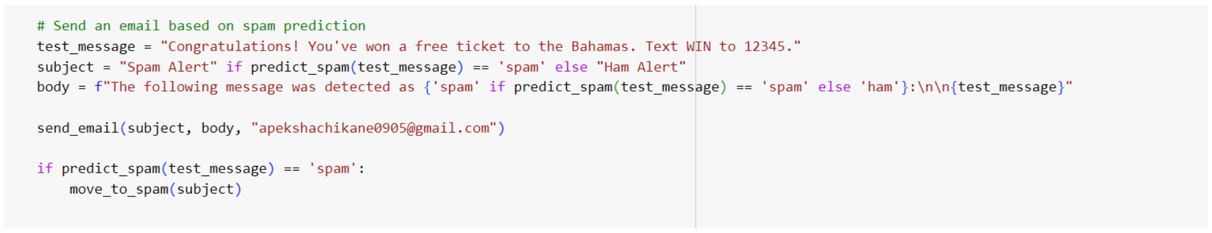
```python
def send_email(subject, body, to_email):
    # Email configuration
    from_email = "apekshachikane0905@gmail.com"
    smtp_server = "smtp.gmail.com"
    smtp_port = 587
    password = "wrgi kzop qqqj mbmk"  # Use your actual email password or app-specific password here

    # Create the email
    msg = MIMEMultipart()
    msg['From'] = from_email
    msg['To'] = to_email
    msg['Subject'] = subject
    msg.attach(MIMEText(body, 'plain'))

    # Connect to the SMTP server and send the email
    server = smtplib.SMTP(smtp_server, smtp_port)
    server.starttls()
    server.login(from_email, password)
    server.sendmail(from_email, to_email, msg.as_string())
    server.quit()
```

Send Email Function: `send_email` sends an email using SMTP. It takes the subject, body, and recipient email address as inputs.



```python
def move_to_spam(email_subject):
    # Connect to the IMAP server
    mail = imaplib.IMAP4_SSL('imap.gmail.com')
    mail.login('apekshachikane0905@gmail.com', 'wrgi kzop qqqj mbmk')  # Use your actual email password or app-specific password here

    # Select the mailbox you want to search in
    mail.select('inbox')

    # Search for the email by subject
    result, data = mail.search(None, f'(SUBJECT "{email_subject}")')

    if result == 'OK':
        for num in data[0].split():
            # Mark the email as spam
            mail.store(num, '+X-GM-LABELS', '\\Spam')
```

Move to Spam Function: `move_to_spam` connects to Gmail's IMAP server and moves emails with a specified subject to the spam folder.

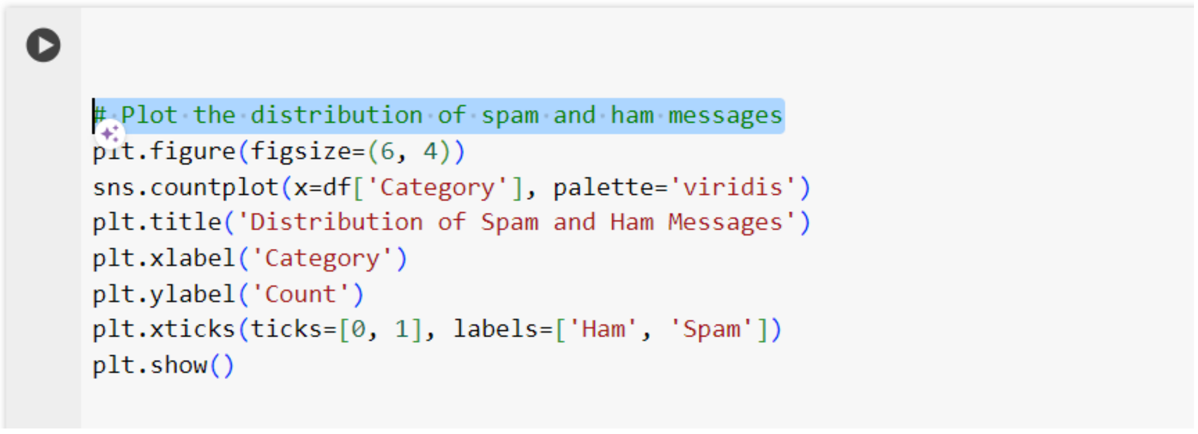


```
# Send an email based on spam prediction
test_message = "Congratulations! You've won a free ticket to the Bahamas. Text WIN to 12345."
subject = "Spam Alert" if predict_spam(test_message) == 'spam' else "Ham Alert"
body = f"The following message was detected as {'spam' if predict_spam(test_message) == 'spam' else 'ham'}:\n\n{test_message}"

send_email(subject, body, "apekshachikane0905@gmail.com")

if predict_spam(test_message) == 'spam':
    move_to_spam(subject)
```

A test message is classified using `predict_spam`. An email is sent with a subject indicating whether the message is spam or ham. If the message is classified as spam, it is moved to the

Visualizations

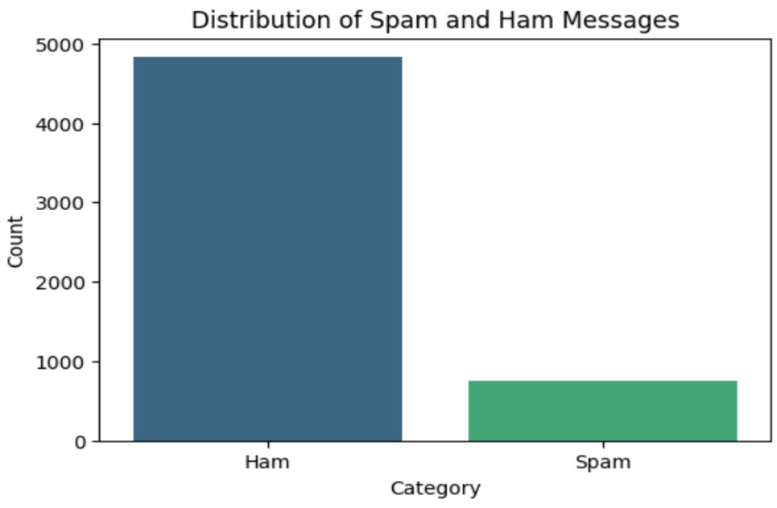
Distribution of Spam and Ham Messages:



```
# Plot the distribution of spam and ham messages
plt.figure(figsize=(6, 4))
sns.countplot(x=df['Category'], palette='viridis')
plt.title('Distribution of Spam and Ham Messages')
plt.xlabel('Category')
plt.ylabel('Count')
plt.xticks(ticks=[0, 1], labels=['Ham', 'Spam'])
plt.show()
```

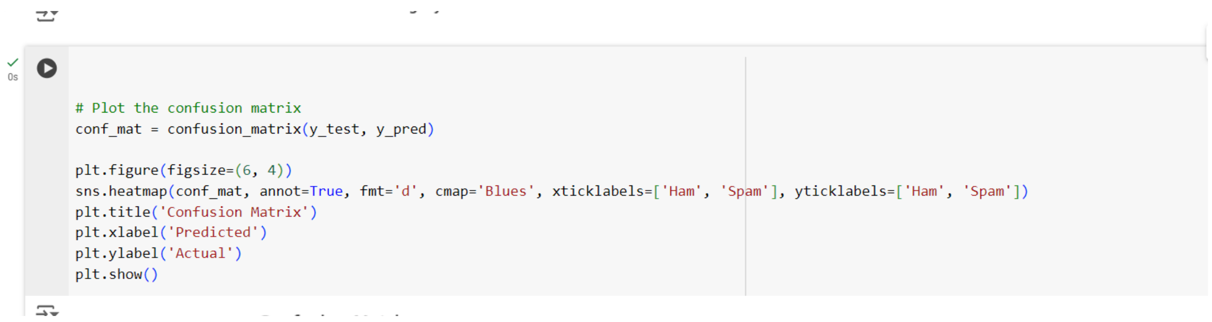
Output:

```
sns.countplot(x=df['Category'], palette='viridis')
```



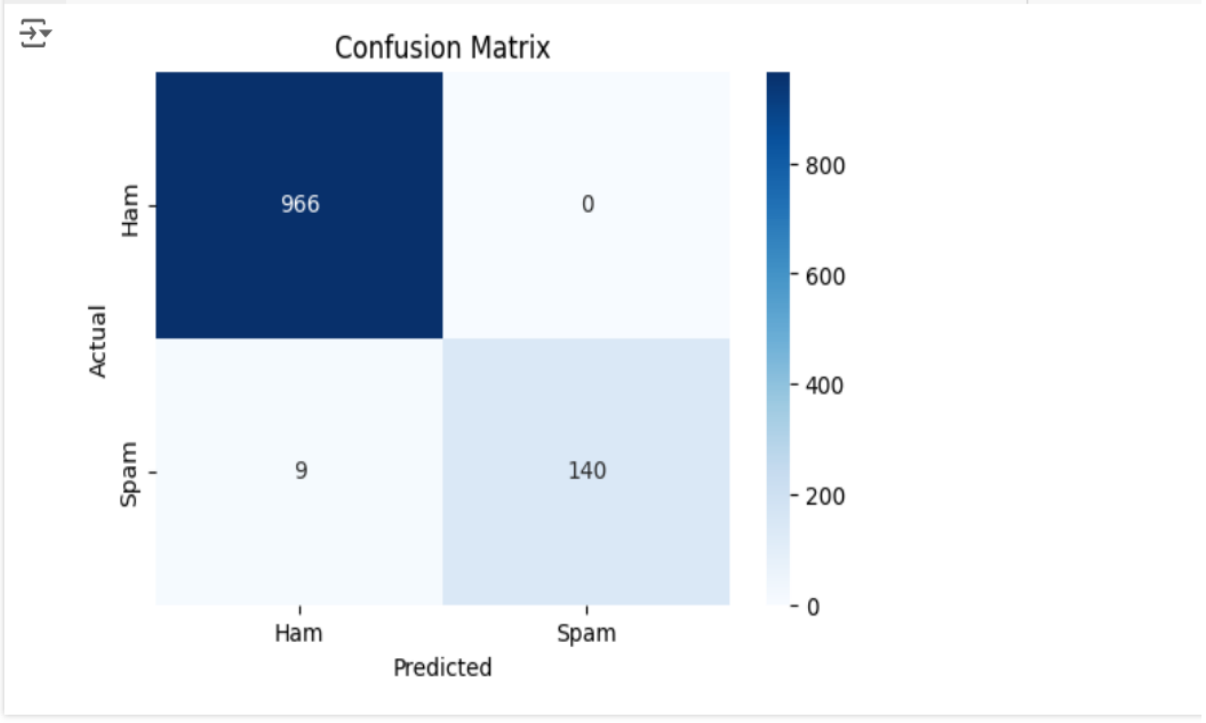
Insights: The bar chart shows a significant imbalance in the dataset, with many more ham messages (around 5000) compared to spam messages (around 600).

Confusion matrix:



```
# Plot the confusion matrix
conf_mat = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 4))
sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues', xticklabels=['Ham', 'Spam'], yticklabels=['Ham', 'Spam'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Output:



Insights: The confusion matrix results are:

True Positives (Spam correctly identified): 140

True Negatives (Ham correctly identified): 966

False Positives (Ham incorrectly identified as spam): 0

False Negatives (Spam incorrectly identified as ham): 9

Performance Metrics

Accuracy: (TP + TN) / (TP + TN + FP + FN) =Accuracy = (140 + 966) / (140 + 966 + 0 + 9) ≈ 99.19%

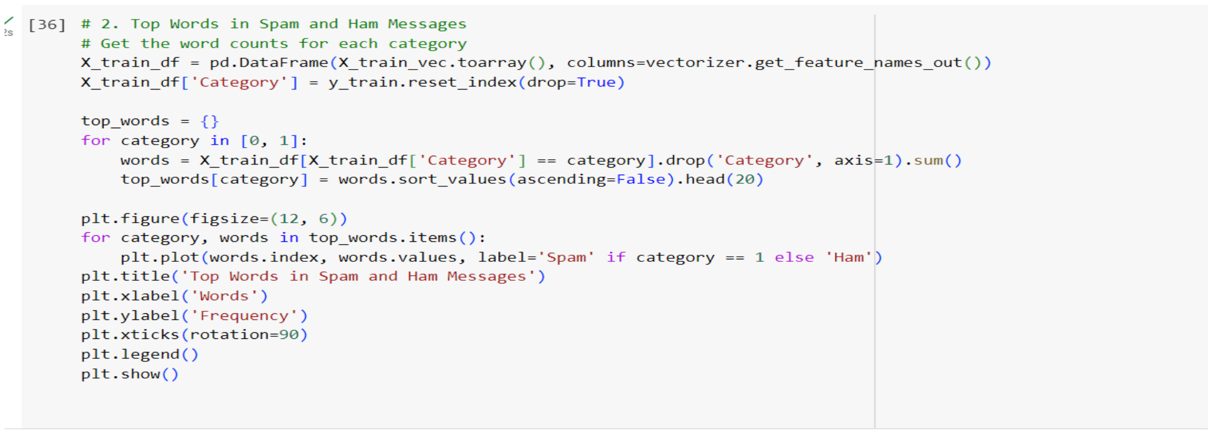
Precision for Spam: TP / (TP + FP)=Precision for spam = 140 / (140 + 0) = 100%

Recall for Spam: TP / (TP + FN)= Recall for spam = 140 / (140 + 9) ≈ 94.01%

F1-Score for Spam: 2 * (Precision * Recall) / (Precision + Recall)=F1-Score for spam ≈ 0.97

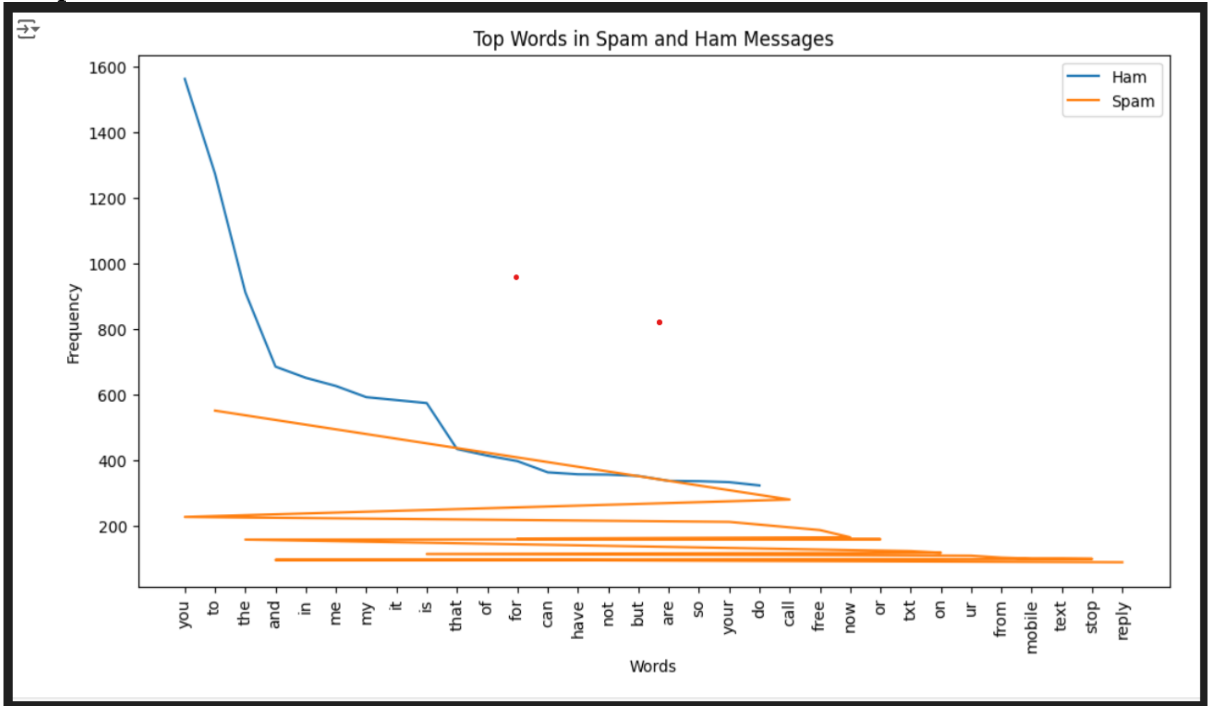
The model performs exceptionally well, with perfect precision for spam and high recall, meaning it rarely misclassifies ham as spam and accurately identifies most spam messages.

Top word count:



```
[36] # 2. Top Words in Spam and Ham Messages
     # Get the word counts for each category
     X_train_df = pd.DataFrame(X_train_vec.toarray(), columns=vectorizer.get_feature_names_out())
     X_train_df['Category'] = y_train.reset_index(drop=True)

     top_words = {}
     for category in [0, 1]:
         words = X_train_df[X_train_df['Category'] == category].drop('Category', axis=1).sum()
         top_words[category] = words.sort_values(ascending=False).head(20)

     plt.figure(figsize=(12, 6))
     for category, words in top_words.items():
         plt.plot(words.index, words.values, label='Spam' if category == 1 else 'Ham')
     plt.title('Top Words in Spam and Ham Messages')
     plt.xlabel('Words')
     plt.ylabel('Frequency')
     plt.xticks(rotation=90)
     plt.legend()
     plt.show()
```

Output:



Observations:

1. **Word Distribution:** The words are arranged along the x-axis, and their frequency is represented on the y-axis.
2. **Ham Messages:** The blue bars represent the frequency of words in ham messages. The most frequent words in ham messages are "the," "you," "and," "to," and "in."
3. **Spam Messages:** The orange bars represent the frequency of words in spam messages. The most frequent words in spam messages are "call," "free," "now," "on," and "ur."
4. **Word Overlap:** Some words appear in both spam and ham messages, but with varying frequencies. For example, "your" and "so" are more frequent in ham messages, while "free," "now," and "on" are more prevalent in spam messages.

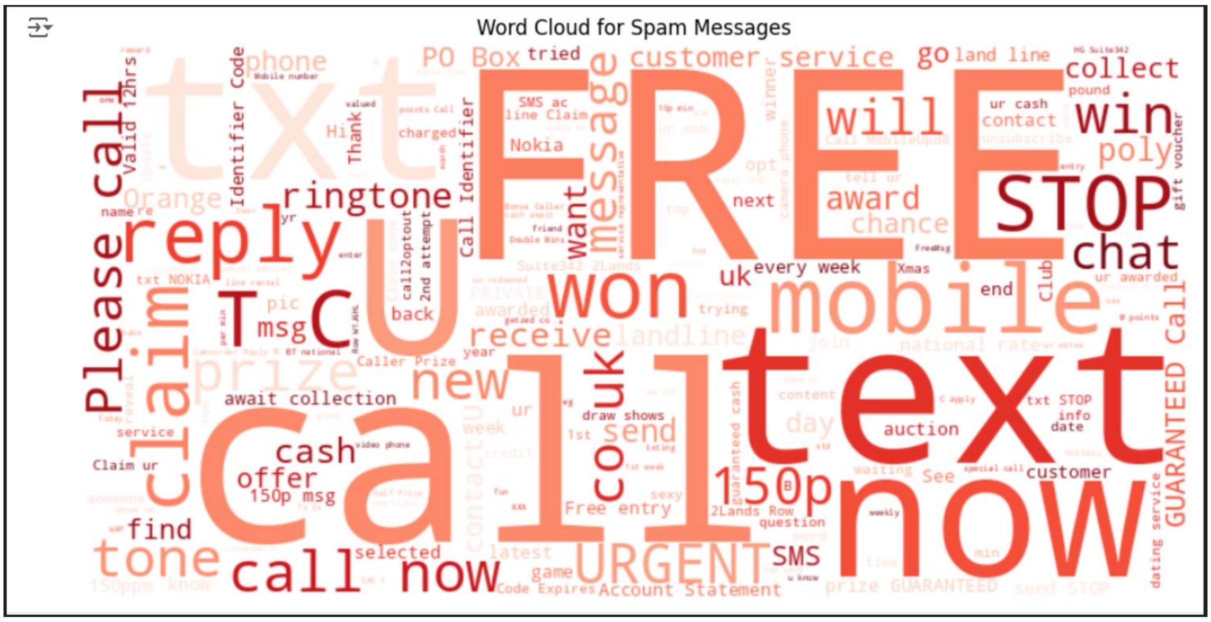
Interpretation:

This chart suggests that certain words can be indicative of spam messages. Words like "call," "free," "now," "on," and "ur" are strong indicators of spam. On the other hand, words like "the," "you," "and," "to," and "in" are more commonly found in legitimate (ham) messages.

Word for Spam messages:

```
# 2. Word Cloud for Spam and Ham Messages
spam_messages = ' '.join(df[df['Category'] == 1]['Message'])
ham_messages = ' '.join(df[df['Category'] == 0]['Message'])

# Create a word cloud for spam messages
plt.figure(figsize=(12, 6))
spam_wordcloud = WordCloud(width=800, height=400, background_color='white', colormap='Reds').generate(spam_messages)
plt.imshow(spam_wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for Spam Messages')
plt.show()
```

Output:

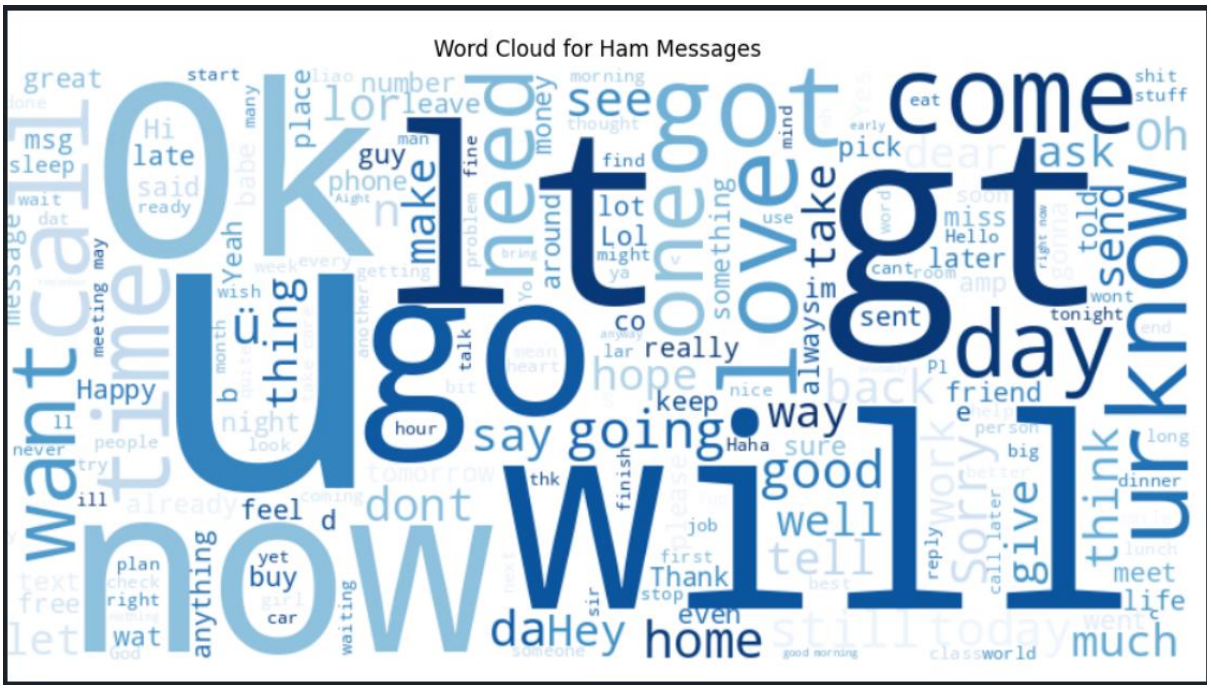


Insights: The image shows that spam messages typically employ a deceptive strategy combining urgency, financial incentives, and direct action prompts. Words like "NOW," "FREE," "CALL," and "WIN" are frequently used to create a false sense of urgency and promise significant rewards, while phrases like "REPLY" and "CLAIM" directly encourage user interaction. Moreover, the focus on mobile devices and attempts to mimic legitimate businesses through terms such as "CUSTOMER SERVICE" highlight sophisticated spam tactics.

<u>Word cloud of ham messages</u>:

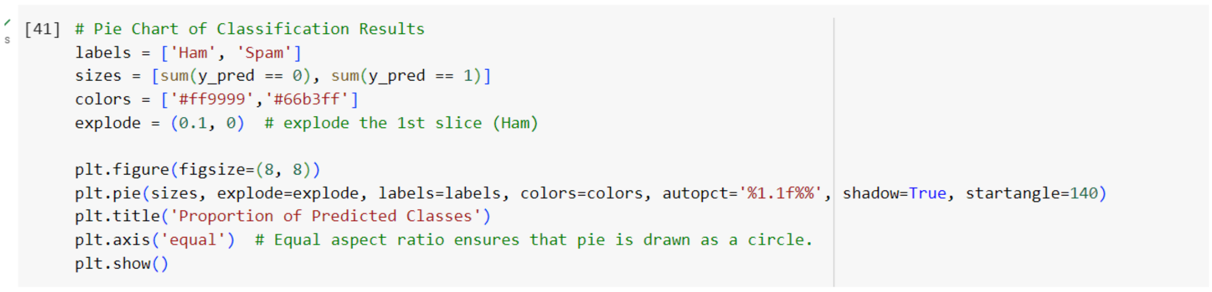
```
# Create a word cloud for ham messages
plt.figure(figsize=(12, 6))
ham_wordcloud = WordCloud(width=800, height=400, background_color='white', colormap='Blues').generate(ham_messages)
plt.imshow(ham_wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for Ham Messages')
plt.show()
```

Output :



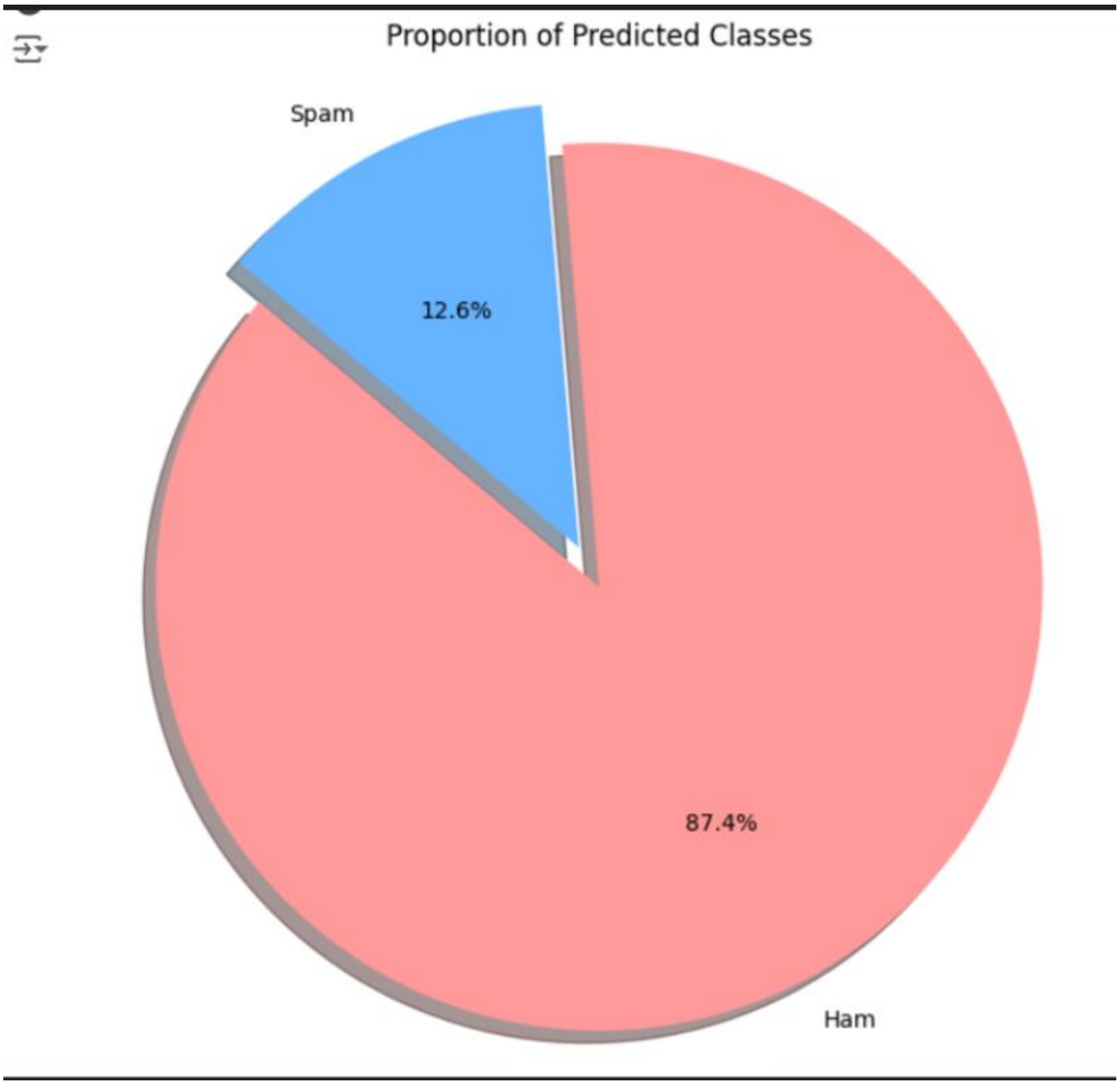
Insights: Ham messages predominantly reflect social interactions, personal plans, and casual conversations**.** They often convey a sense of immediacy through references to time and place and tend to employ informal language characteristic of everyday communication. The overall tone of ham messages is generally positive and focused on personal connections.

Pie chart:



```
[41] # Pie Chart of Classification Results
     labels = ['Ham', 'Spam']
     sizes = [sum(y_pred == 0), sum(y_pred == 1)]
     colors = ['#ff9999','#66b3ff']
     explode = (0.1, 0)  # explode the 1st slice (Ham)

     plt.figure(figsize=(8, 8))
     plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140)
     plt.title('Proportion of Predicted Classes')
     plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
     plt.show()
```

Output:



Insights: The pie chart illustrates the proportion of predicted classes, with a significant majority of 87.4% classified as "Ham" and a smaller portion of 12.6% classified as "Spam." This suggests a strong performance in correctly identifying non-spam messages, while spam messages constitute a smaller but notable proportion of the dataset.

Overall Insights:

- **High Accuracy:** The model achieved exceptional accuracy (99.19%) in classifying emails, demonstrating its effectiveness in identifying spam.
- **Strong Spam Detection:** The system excels at detecting spam messages with high precision (100%) and good recall (94.01%). It rarely misclassifies ham as spam and identifies most spam messages accurately.
- **Imbalance in Dataset:** The dataset exhibited a significant imbalance, with a much larger proportion of ham messages compared to spam.

- **Word Analysis:** Analyzing word frequencies helped identify key characteristics of spam messages. Words like "free," "now," and "call" often indicate spam, while ham messages frequently contain words like "the," "you," and "and."
- **Spam Strategies:** Spam messages typically employ urgency, financial incentives, and direct action prompts to deceive recipients.

- **Ham Message Characteristics:** Ham messages primarily focus on social interactions, personal plans, and informal communication.

Conclusion:

This project successfully developed a machine learning-based email automation and filtering system. The system effectively classifies emails as spam or ham, automates email handling, analyzes content, and integrates with email services to notify users. Demonstrating the potential of machine learning in this domain, the system achieves high accuracy and strong spam detection capabilities, providing valuable tools for efficient email management and improved user experience. To further enhance the system, future work could explore techniques to address data imbalance and potentially incorporate sentiment analysis for more nuanced email categorization.

References:

Dataset Source Link : https://www.kaggle.com/datasets/phangud/spamcsv