Task No. 3 – Implementation of the method

1. A brief description of the language choice (e.g. Java, Python, C++) with justification

PYHTON:
For the usage of the strategy, Python has been chosen. Python could be a flexible and widely-used programming dialect, especially well-suited for errands like normal dialect preparing and machine learning, which are pertinent for mail spam discovery. Its broad environment of libraries, such as scikit-learn for machine learning and pandas for information control, at the side its effortlessness and coherence, make it an great choice for executing the chosen strategy. Also, Python permits for quick advancement and experimentation, which is invaluable for investigating different approaches to mail spam location.

2. The entire code with comments is sent in an editable form to implement the selected method.

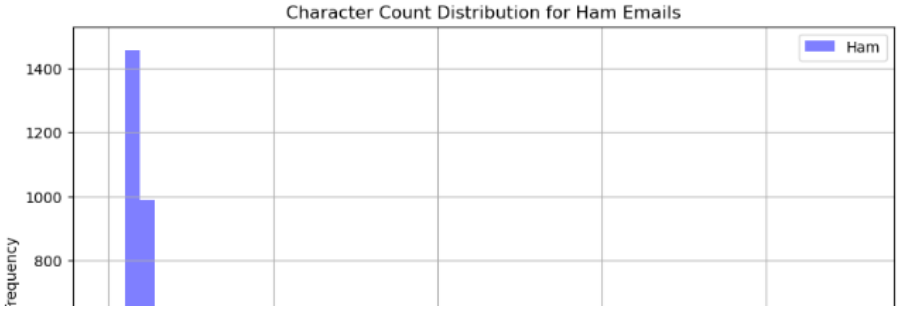
```
# Importing necessary libraries
import pandas as pd  # For data manipulation and analysis
        from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
import numpy as np        # For numerical operations
import matplotlib.pyplot as plt  # For data visualization
        %matplotlib inline
import seaborn as sns
import matplotlib.pyplot as plt


        # Load the Enron Mail Dataset into a pandas DataFrame
        df = pd.read_csv('spam.csv', encoding='latin-1')
        styled_df = df.head()
        styled_df = styled_df.style.set_table_styles([
        {"selector": "th", "props": [("color", 'black'), ("background-
color", "#FF00CC")]}
        ])
        styled_df
        df.info()

        # Data Cleaning: Removing unnecessary columns and handling
missing values
        df = df[['v2', 'v1']]
        df.columns = ['text', 'spam']
        df.dropna(inplace=True)

        # Separate features (email text) and labels (spam or non-spam)
        X = df['text']
        y = df['spam']
import nltk

# Tokenization functions
        def count_characters(text):
        return len(text)
        def count_words(text):
```

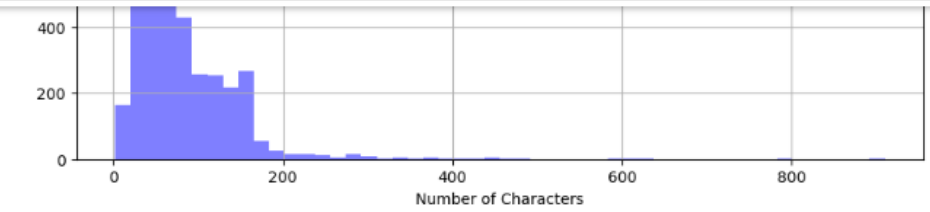
```
        return len(nltk.word_tokenize(text))

        def count_sentences(text):
        return len(nltk.sent_tokenize(text))

        # Apply tokenization functions to create new columns
        df['num_characters'] = df['text'].apply(count_characters)
        df['num_words'] = df['text'].apply(count_words)
        df['num_sentences'] = df['text'].apply(count_sentences)

        # Printing the descriptive statistics of numerical features
        print("\nDescriptive Statistics of Numerical Features:")
        print(df[['num_characters', 'num_words',
'num_sentences']].describe())

import nltk

# Tokenization functions
        def count_characters(text):
        return len(text)
        def count_words(text):
        return len(nltk.word_tokenize(text))
        def count_sentences(text):
        return len(nltk.sent_tokenize(text))

        # Apply tokenization functions to create new columns for ham
emails
        df['num_characters_ham'] = df[df['spam'] ==
'ham']['text'].apply(count_characters)
        df['num_words_ham'] = df[df['spam'] ==
'ham']['text'].apply(count_words)
        df['num_sentences_ham'] = df[df['spam'] ==
'ham']['text'].apply(count_sentences)

        # Apply tokenization functions to create new columns for spam
emails
        df['num_characters_spam'] = df[df['spam'] ==
'spam']['text'].apply(count_characters)
        df['num_words_spam'] = df[df['spam'] ==
'spam']['text'].apply(count_words)
        df['num_sentences_spam'] = df[df['spam'] ==
'spam']['text'].apply(count_sentences)

        # Printing the descriptive statistics of numerical features for
ham emails
        print("\nDescriptive Statistics of Numerical Features for Ham
Emails:")
        print(df[df['spam'] == 'ham'][['num_characters_ham',
'num_words_ham', 'num_sentences_ham']].describe())

        # Printing the descriptive statistics of numerical features for
spam emails
        print("\nDescriptive Statistics of Numerical Features for Spam
Emails:")
        print(df[df['spam'] == 'spam'][['num_characters_spam',
'num_words_spam', 'num_sentences_spam']].describe())
import matplotlib.pyplot as plt
        from sklearn.feature_extraction.text import CountVectorizer
import seaborn as sns
        # Histogram of character count for ham emails
        plt.figure(figsize=(10, 6))
```

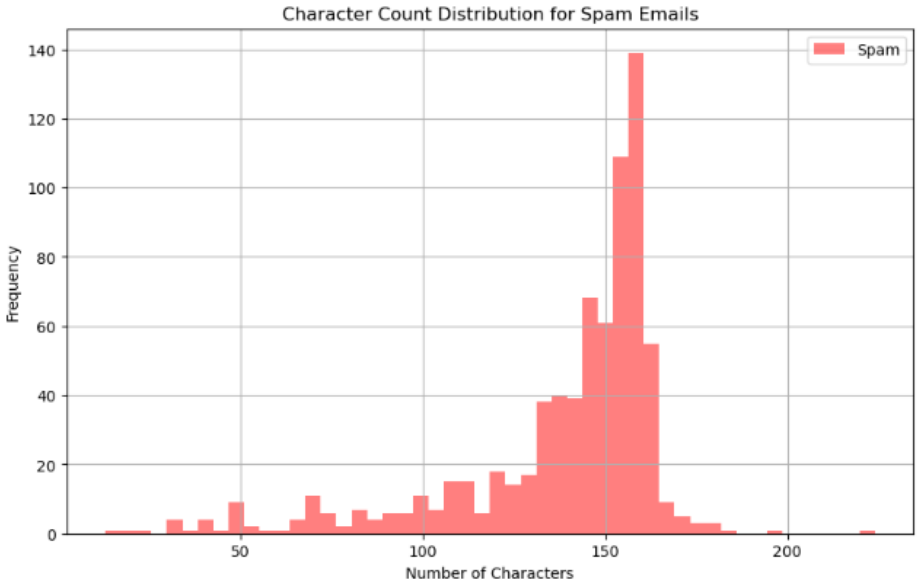
```
        plt.hist(df[df['spam'] == 'ham']['num_characters'], bins=50,
alpha=0.5, color='blue', label='Ham')
        plt.title('Character Count Distribution for Ham Emails')
        plt.xlabel('Number of Characters')
        plt.ylabel('Frequency')
        plt.legend()
        plt.grid(True)
        plt.show()

        # Histogram of character count for spam emails
        plt.figure(figsize=(10, 6))
        plt.hist(df[df['spam'] == 'spam']['num_characters'], bins=50,
alpha=0.5, color='red', label='Spam')
        plt.title('Character Count Distribution for Spam Emails')
        plt.xlabel('Number of Characters')
        plt.ylabel('Frequency')
        plt.legend()
        plt.grid(True)
        plt.show()

        # Scatter plot of word count vs. sentence count
        plt.figure(figsize=(10, 6))
        plt.scatter(df['num_words'], df['num_sentences'], alpha=0.5)
        plt.title('Word Count vs. Sentence Count')
        plt.xlabel('Number of Words')
        plt.ylabel('Number of Sentences')
        plt.grid(True)
        plt.show()

        # Split the dataset into training and testing sets (80% train,
20% test)
        X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
        from sklearn.feature_extraction.text import CountVectorizer
import seaborn as sns
# CountVectorizer ile kelime frekanslarını hesapla
        vectorizer = CountVectorizer(stop_words='english')
X_train_vectorized = vectorizer.fit_transform(X_train)
word_frequencies = pd.DataFrame(X_train_vectorized.toarray(),
columns=vectorizer.get_feature_names_out())
top_20_words =
word_frequencies.sum().sort_values(ascending=False).head(20)
plt.figure(figsize=(10, 6))
        sns.barplot(x=top_20_words.values, y=top_20_words.index,
palette='viridis')
plt.title('Top 20 Most Frequent Words in Emails')
plt.xlabel('Frequency')
plt.ylabel('Word')
plt.show()
# Text Preprocessing: Convert text features into numerical vectors using
CountVectorizer
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
# Naive Bayes Classifier
        naive_bayes_classifier = MultinomialNB()
naive_bayes_classifier.fit(X_train_vectorized, y_train)
# Predictions
        y_pred_train =
naive_bayes_classifier.predict(X_train_vectorized)
y_pred_test = naive_bayes_classifier.predict(X_test_vectorized)
```

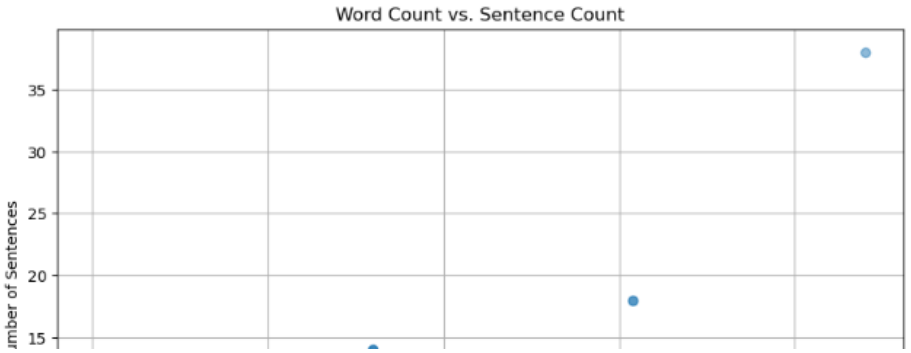
```python
# Model Evaluation
train_accuracy = accuracy_score(y_train, y_pred_train)
test_accuracy = accuracy_score(y_test, y_pred_test)
print("Training Accuracy:", train_accuracy)
print("Testing Accuracy:", test_accuracy)
print("\nClassification Report (Testing Data):\n",
classification_report(y_test, y_pred_test))
```

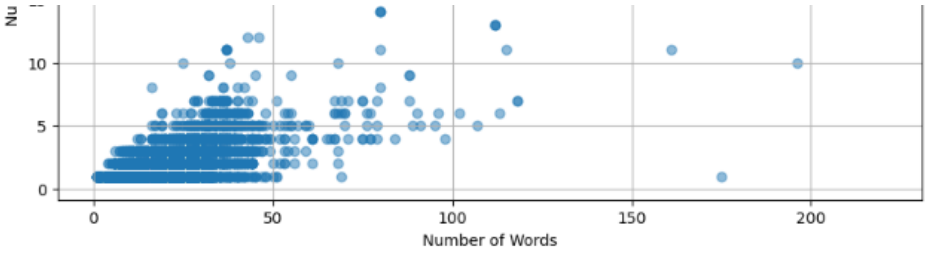
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB

Descriptive Statistics of Numerical Features:
       num_characters    num_words  num_sentences
count     5572.000000  5572.000000    5572.000000
mean        80.118808    18.699390       1.996411
std         59.690841    13.741932       1.520159
min          2.000000     1.000000       1.000000
25%         36.000000     9.000000       1.000000
50%         61.000000    15.000000       1.500000
75%        121.000000    27.000000       2.000000
max        910.000000   220.000000      38.000000

Descriptive Statistics of Numerical Features for Ham Emails:
       num_characters_ham  num_words_ham  num_sentences_ham
count         4825.000000    4825.000000        4825.000000
mean            71.023627      17.276269           1.837720
std             58.016023      13.988585           1.454388
min              2.000000       1.000000           1.000000
25%             33.000000       8.000000           1.000000
50%             52.000000      13.000000           1.000000
75%             92.000000      22.000000           2.000000
max            910.000000     220.000000          38.000000

Descriptive Statistics of Numerical Features for Spam Emails:
       num_characters_spam  num_words_spam  num_sentences_spam
count           747.000000      747.000000          747.000000
mean            138.866131       27.891566            3.021419
std              29.183082        6.867007            1.537580
min              13.000000        2.000000            1.000000
25%             132.500000       25.000000            2.000000
50%             149.000000       29.000000            3.000000
75%             157.000000       32.000000            4.000000
max             224.000000       46.000000            9.000000
```

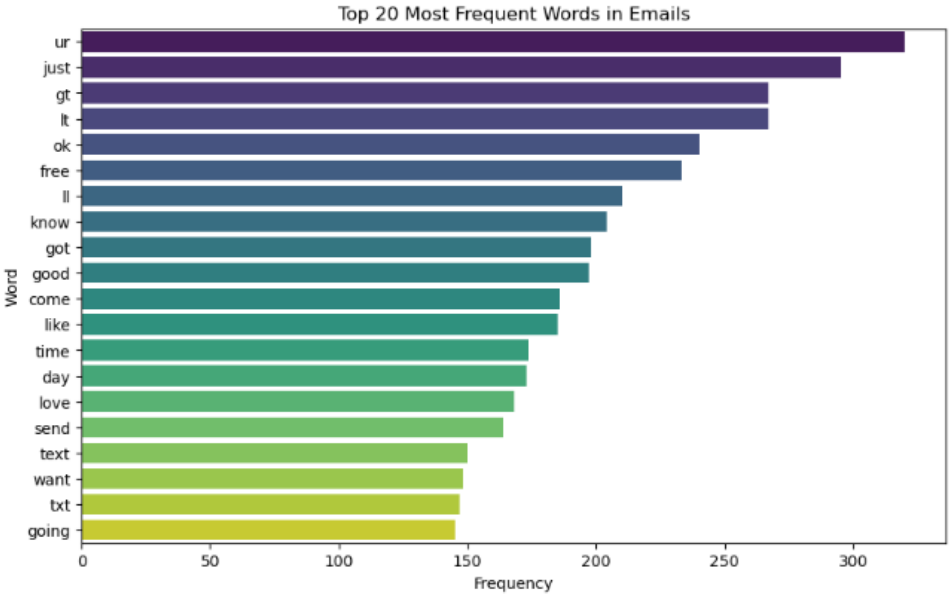



400
200
0
0
200
400
600
800
Number of Characters


Character Count Distribution for Spam Emails
Spam
140
120
100
80
60
40
20
0
Frequency
50
100
150
200
Number of Characters


Word Count vs. Sentence Count
35
30
25
20
15
umber of Sentences






```
Training Accuracy: 0.9943908458604442
Testing Accuracy: 0.9838565022421525

Classification Report (Testing Data):
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99       965
        spam       0.99      0.89      0.94       150

    accuracy                           0.98      1115
   macro avg       0.98      0.95      0.96      1115
weighted avg       0.98      0.98      0.98      1115
```

3. All software in a zip that you can download and run