Data Collection and Preprocessing Phase

Date	15 March 2024
Team ID	SWTID1727180793
Project Title	SMS- Spam Detection Using NLP
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

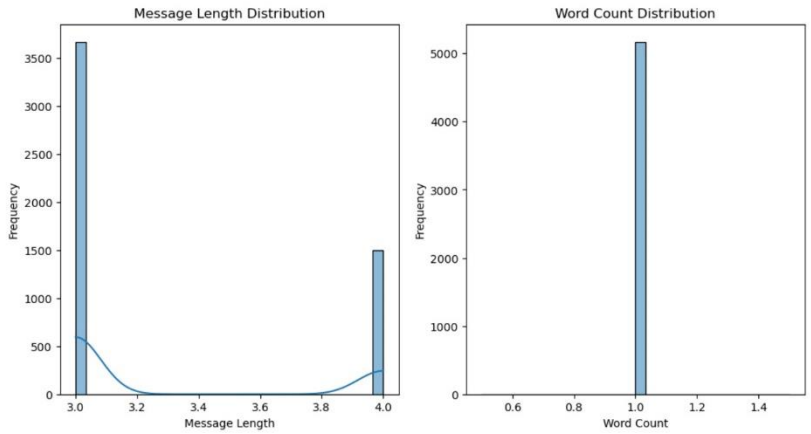
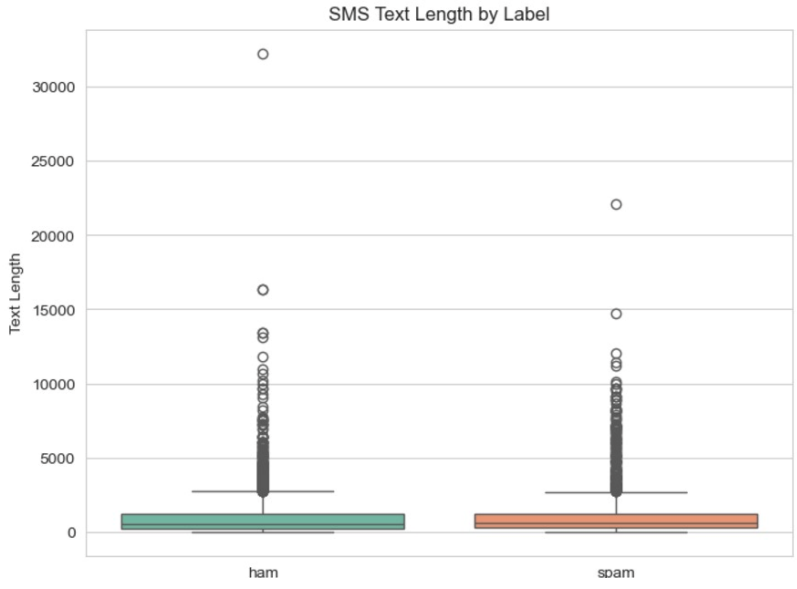
Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

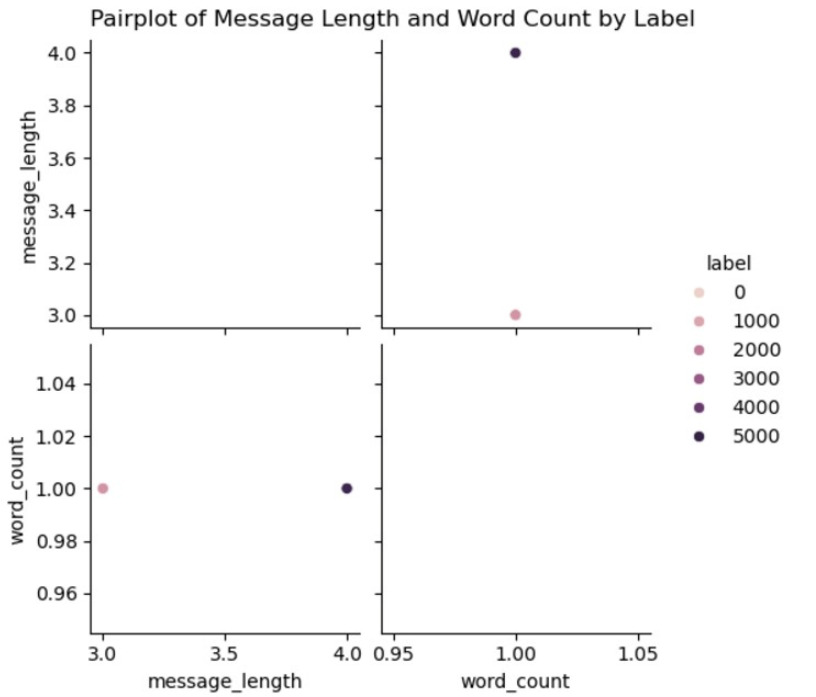
Section: Data Overview

Description:

```
Dimension:
5171 rows × 4 columns
```

	Unnamed: 0	label_num
count	5171.000000	5171.000000
mean	2585.000000	0.289886
std	1492.883452	0.453753
min	0.000000	0.000000
25%	1292.500000	0.000000
50%	2585.000000	0.000000
75%	3877.500000	1.000000
max	5170.000000	1.000000

Univariate Analysis	
Bivariate Analysis	

Multivariate Analysis	
Outliers and Anomalies	-

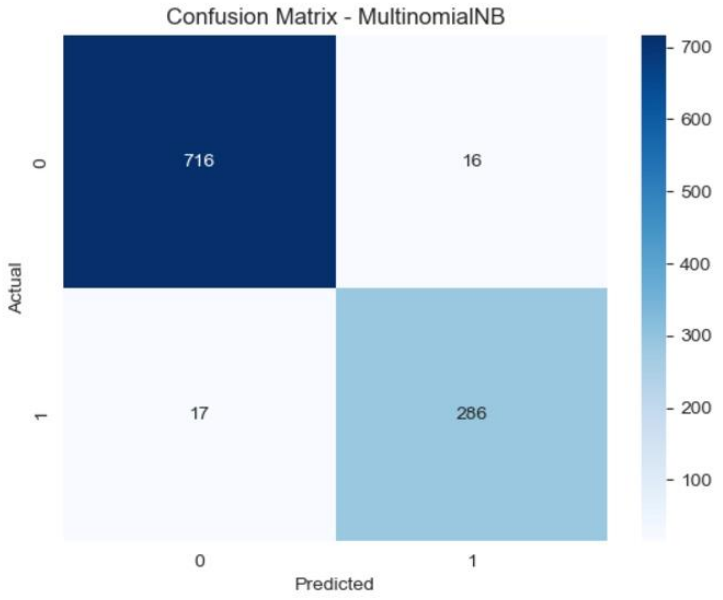
Data Preprocessing Code Screenshots

Loading Data

```
#Load our dataset
df = pd.read_csv("spam_ham_dataset.csv")
```

```
#top 5 rows of the dataframes
df.head()
```

	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0

Handling Missing Data	[5]: ``` # Drop the column'Unnamed: 0' df = df.drop("Unnamed: 0", axis=1) ``` [6]: ``` # Return the shape of the data df.shape ``` [6]: (5171, 3) [7]: ``` # Return the number of dimensions df.ndim ``` [7]: 2 [8]: ``` # Return the size of the data df.size ``` [8]: 15513 [9]: ``` # Returns the sum of all NA values df.isna().sum() ``` [9]: ``` label 0 text 0 label_num 0 dtype: int64 ```
Data Transformation	``` import seaborn as sns import matplotlib.pyplot as plt # Confusion matrix for Naive Bayes sns.heatmap(cm, annot=True, fmt="d", cmap="Blues") plt.title("Confusion Matrix - MultinomialNB") plt.ylabel("Actual") plt.xlabel("Predicted") plt.show() ``` 

Feature Engineering	

```
# Download stopwords if necessary
nltk.download('stopwords')

# Initialize variables
corpus = []  # List to store preprocessed text
pe = PorterStemmer()  # Initialize stemmer
stopword = stopwords.words("english")  # List of stopwords

# Loop through all rows in the dataset for text preprocessing
for i in range(len(df)):
    # Remove non-alphanumeric characters
    text = re.sub("[^a-zA-Z0-9]", " ", df["text"][i])

    # Convert text to lowercase
    text = text.lower()

    # Split the text into words
    text = text.split()

    # Apply stemming and remove stopwords
    text = [pe.stem(word) for word in text if word not in set(stopword)]

    # Join the words back into a single string
    text = " ".join(text)

    # Append the processed text to the corpus
    corpus.append(text)

# Convert the preprocessed text data into numerical features using TfidfVectorizer
tfidf = TfidfVectorizer(max_features=35000)  # Limit to top 35000 features
X = tfidf.fit_transform(corpus).toarray()  # Transform the corpus into a feature matrix

# Extract dependent variable (target labels) from the dataset
y = pd.get_dummies(df['label'])['spam'].values  # Convert 'spam' and 'ham' labels to binary values
```

Save Processed Data	

```
import pickle  ## Importing pickle for dumping models
pickle.dump(cv, open('cv-transform.pkl', 'wb'))  ## Saving into cv-transform.pkl file
```

```
import pickle
pickle.dump(model , open ("spam-sms-mnb-model.pkl" , "wb"))
```