**EMAIL SPAM DETECTION EXPLANATION**

**1. Introduction:**

The Email Spam Detection project is designed to classify SMS messages as spam or ham (non-spam) using natural language processing (NLP) and machine learning techniques. The primary goal is to build a model that can accurately identify spam messages based on their content.

**2. Importing Necessary Packages:**

First, import essential libraries required for data manipulation, visualization, and machine learning:

import nltk

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import string

from sklearn.feature\_extraction.text import CountVectorizer, TfidfTransformer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.model\_selection import train\_test\_split

from sklearn.pipeline import Pipeline

from sklearn.metrics import classification\_report, confusion\_matrix

**3. Data Loading and Exploration:**

Load the dataset from the file `SMSSpamCollection.csv` and explore its structure:

messages = [line.rstrip() for line in open('SMSSpamCollection.csv')]

mess\_data = pd.read\_csv('SMSSpamCollection.csv', sep='\t', names=['Labels', 'Message'])

Examine the first few records and the distribution of message lengths to understand the data:

mess\_data.head()

mess\_data['Length'] = mess\_data['Message'].apply(len)

mess\_data.describe()

mess\_data.groupby('Labels').describe().transpose()

**4. Data Visualization:**

To visualize the distribution of message lengths and how they differ between spam and ham:

mess\_data['Length'].plot(kind='hist', bins=20)

sns.histplot(data=mess\_data, x='Length', hue='Labels', bins=50)

plt.figure(figsize=(10,10))

g = sns.FacetGrid(data=mess\_data, col='Labels', height=5)

g.map(sns.histplot, 'Length', bins=50)

mess\_data.hist(column='Length', by='Labels', bins=50, figsize=(12,4))

**5. Data Preprocessing:**

Preprocess the text data by removing punctuation and stop words to clean the messages:

def text\_process(mess):

no\_punc = [c for c in mess if c not in string.punctuation]

return [word for word in ''.join(no\_punc).split() if word.lower() not in stopwords.words('english')]

mess\_data['Message'].apply(lambda x: text\_process(x))

**6. Data Transformation:**

Transform the text data into numerical form using `CountVectorizer` and `TfidfTransformer`:

bow\_transformer = CountVectorizer(analyzer=text\_process).fit(mess\_data['Message'])

messages\_bow = bow\_transformer.transform(mess\_data['Message'])

tfidf\_transformer = TfidfTransformer().fit(messages\_bow)

messages\_tfidf = tfidf\_transformer.transform(messages\_bow)

**7. Model Training:**

Train a Naive Bayes classifier to predict spam messages:

spam\_detect\_model = MultinomialNB().fit(messages\_tfidf, mess\_data['Labels'])

all\_predictions = spam\_detect\_model.predict(messages\_tfidf)

Evaluate the model using a classification report:

report\_dict = classification\_report(mess\_data['Labels'], all\_predictions, output\_dict=True)

report\_df = pd.DataFrame(report\_dict).transpose()

sns.heatmap(report\_df, annot=True, cmap='Oranges', cbar=False, fmt='.2f', linewidths=0.4, linecolor='white')

**8. Model Training with Train/Test Split:**

Split the data into training and testing sets to evaluate the model's performance on unseen data:

msg\_train, msg\_test, label\_train, label\_test = train\_test\_split(mess\_data['Message'], mess\_data['Labels'], test\_size=0.30, random\_state=42)

pipeline = Pipeline([('bow', CountVectorizer(analyzer=text\_process)), ('tfid', TfidfTransformer()), ('model', MultinomialNB())])

pipeline.fit(msg\_train, label\_train)

predictions = pipeline.predict(msg\_test)

Generate a final classification report and confusion matrix for the test data:

final\_report\_dict = classification\_report(label\_test, predictions, output\_dict=True)

final\_report\_df = pd.DataFrame(final\_report\_dict).transpose()

sns.heatmap(final\_report\_df, annot=True, cmap='Greens', cbar=False, fmt='.2f', linewidths=0.4, linecolor='white')

sns.heatmap(confusion\_matrix(label\_test, predictions), annot=True, fmt='d', cmap='Blues')

**9. Conclusion:**

The project successfully demonstrates how to build a spam detection system using NLP and machine learning. The model can accurately classify messages as spam or ham, providing a valuable tool for filtering unwanted messages.