PHASE 3

BUILDING A SMARTER AI-POWERED SPAM CLASSIFIER

TEAM MEMBERS:

1. NALINKUMAR S - (2021504304)
2. DINESHKUMAR S - (2021504710)
3. PRAVEEN G R - (2021504307)
4. VENGADESHWARAN S - (2021504708)

Abstract:

Spam Classification using Artificial Intelligence – For business purposes, email is the most widely utilized mode of official communication. Despite the availability of other forms of communication, email usage continues to rise. In today's world, automated email management is critical since the volume of emails grows by the day. More than 55 percent of all emails have been recognized as spam. This demonstrates that spammers waste email users' time and resources while producing no meaningful results. Spammers employ sophisticated and inventive strategies to carry out their criminal actions via spam emails. As a result, it is critical to comprehend the many spam email classification tactics and mechanisms. The main focus of this paper is on spam classification using machine learning algorithms. Furthermore, this research includes a thorough examination and evaluation of research on several machine-learning methodologies and email properties used in various Machine Learning approaches. Future study goals and obstacles in the subject of spam classification are also discussed, which may be valuable to future researchers

Objective:

Machine learning algorithms use statistical models to classify data. In the case of spam detection, a trained machine learning model must be able to determine whether the sequence of words found in an email is closer to those found in spam emails or safe ones.

Introduction:

For the majority of internet users, email has become the most often utilized formal communication channel. In recent years, there has been a surge in email usage, which has exacerbated the problems presented by spam emails. Spam, often known as junk email, is the act of sending unsolicited mass messages to a large number of people. 'Ham' refers to emails that are meaningful but of a different type. Every day, the average email user receives roughly 40-50 emails. Spammers earn roughly 3.5 million dollars per year from spam, resulting in financial damages on both a personal and institutional level. As a result, consumers devote a large amount of their working time to these emails. Spam is said to account for more than half of all email server traffic, sending out a vast volume of undesired and uninvited bulk emails.They squander user resources on useless output, lowering productivity. Spammers use spam for marketing goals to spread malicious

criminal acts such as identity theft, financial disruptions, stealing sensitive information, and reputational damage.

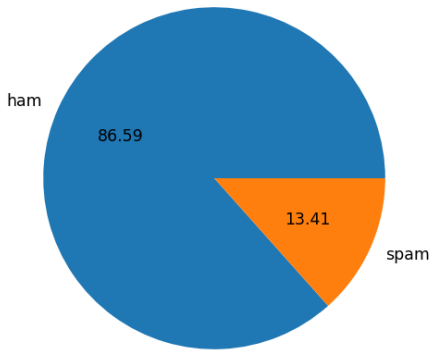
Overview of the Dataset used:

The SMS Spam Collection is a set of SMS-tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according to being ham (legitimate) or spam.

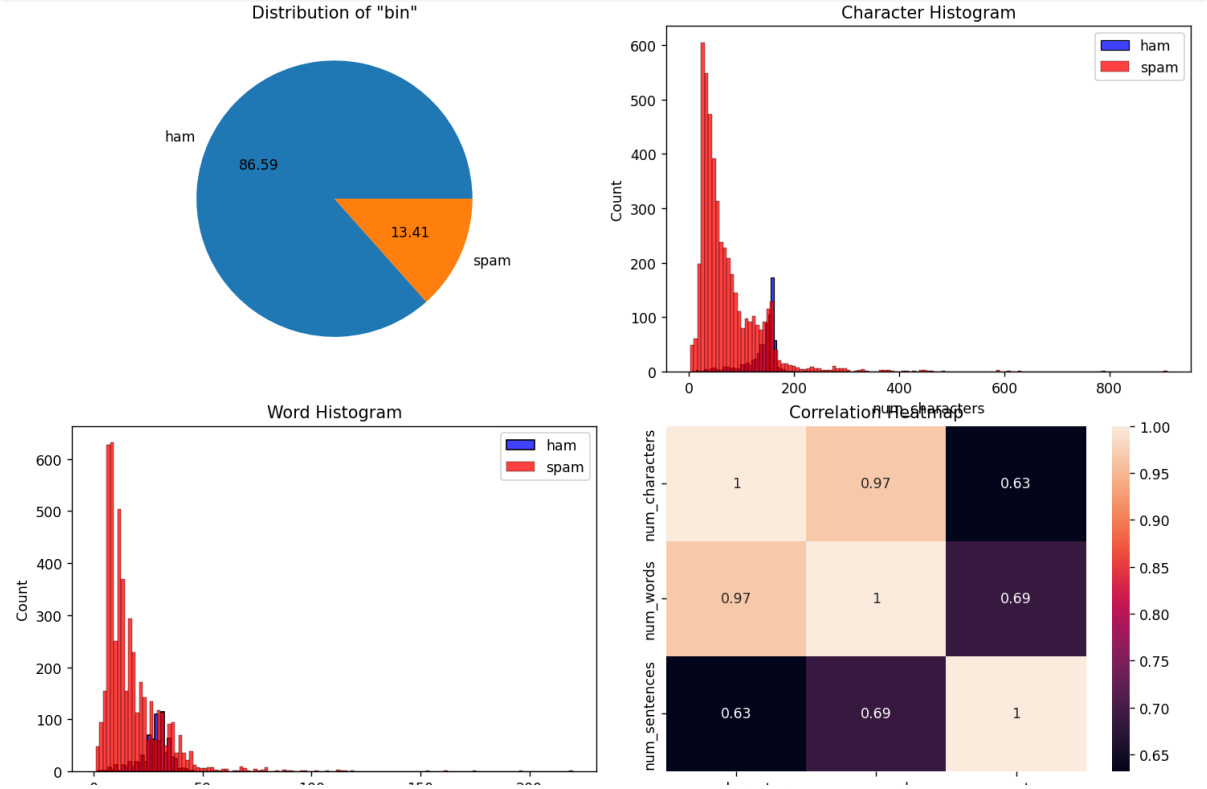
Program:

```
import numpy as np
import nltk
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score

# Load the CSV file
df = pd.read_csv("D:/test/nmp/spam.csv", encoding='latin1')

# Rename the columns
df.rename(columns = {'v1': 'bin', 'v2': 'message'}, inplace=True)

df['bin'] = df['bin'].map({'spam': 0, 'ham': 1})

# Split the data into training and testing sets
x = df['message']
y = df['bin']

nltk.download('punkt')
```

```
PS D:\test> & C:/Users/ACER/AppData/Local/Microsoft/WindowsApps/python3.11.exe "d:/test/nmp/nm phase-3.py"
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 Ì_ 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: 5572, dtype: object
PS D:\test> & C:/Users/ACER/AppData/Local/Microsoft/WindowsApps/python3.11.exe "d:/test/nmp/nm phase-3.py"
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 Ì_ 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: 5572, dtype: object
0       1
1       1
2       0
3       1
4       1
       ..
5567    0
5568    1
5569    1
5570    1
5571    1
Name: bin, Length: 5572, dtype: int64
PS D:\test>
```



```python
# Create subplots
fig, axes = plt.subplots(2, 2, figsize=(12, 12))

# Plot 1: Distribution of 'bin'
d1 = df['bin'].value_counts()
axes[0, 0].pie(d1, labels=['ham', 'spam'], autopct='%0.2f')
axes[0, 0].set_title('Distribution of "bin"')

# Plot 2: Character Histogram
sns.histplot(df[df['bin'] == 0]['num_characters'], ax=axes[0, 1], color='blue', label='ham')
sns.histplot(df[df['bin'] == 1]['num_characters'], ax=axes[0, 1], color='red', label='spam')
axes[0, 1].set_title('Character Histogram')
axes[0, 1].legend()

# Plot 3: Word Histogram
sns.histplot(df[df['bin'] == 0]['num_words'], ax=axes[1, 0], color='blue', label='ham')
sns.histplot(df[df['bin'] == 1]['num_words'], ax=axes[1, 0], color='red', label='spam')
axes[1, 0].set_title('Word Histogram')
axes[1, 0].legend()

# Plot 4: Correlation Heatmap
numeric_columns = df[['num_characters', 'num_words', 'num_sentences']]
correlation = numeric_columns.corr()
sns.heatmap(correlation, annot=True, ax=axes[1, 1])
axes[1, 1].set_title('Correlation Heatmap')

plt.tight_layout()
plt.show()
```



```
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
X = tfidf.fit_transform(df['message']).toarray()
y = df['bin'].values
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)

gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
gnb.fit(X_train,y_train)
y_pred1 = gnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print(accuracy_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
print(precision_score(y_test,y_pred2))
bnb.fit(X_train,y_train)
y_pred3 = bnb.predict(X_test)
print(accuracy_score(y_test,y_pred3))
print(confusion_matrix(y_test,y_pred3))
print(precision_score(y_test,y_pred3))
```

```
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50,random_state=2)
xgb = XGBClassifier(n_estimators=50,random_state=2)
```

```python
# Define a dictionary of classifiers
clfs = {
    'SVC': svc,
    'KN': knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT': gbdt,
    'xgb': xgb
}

def train_classifier(clf, X_train, y_train, X_test, y_test):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    return accuracy, precision

# Train and evaluate each classifier
for name, clf in clfs.items():
    current_accuracy, current_precision = train_classifier(clf, X_train, y_train, X_test, y_test)
    print("For", name)
    print("Accuracy:", current_accuracy)
    print("Precision:", current_precision)
```

```
PS D:\test> & C:/Users/ACER/AppData/Local/Microsoft/WindowsApps/python3.11.exe d:/test/nmp/testnm.py
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ACER\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
0.8860986547085202
[[134  24]
 [103 854]]
0.9726651480637813
0.9695067264573991
[[124  34]
 [  0 957]]
0.9656912209889001
0.97847533632287
[[137  21]
 [  3 954]]
0.9784615384615385
For SVC
Accuracy: 0.9757847533632287
Precision: 0.9744897959183674
For KN
Accuracy: 0.9094170403587444
Precision: 0.9045368620037807
For NB
Accuracy: 0.9695067264573991
Precision: 0.9656912209889001
For DT
Accuracy: 0.9488789237668162
Precision: 0.9473161033797217
For LR
Accuracy: 0.9560538116591928
Precision: 0.9558232931726908
For RF
Accuracy: 0.9704035874439462
Precision: 0.9666666666666667
For AdaBoost
Accuracy: 0.9695067264573991
Precision: 0.9685279187817258
```