Assignment Report

Spam Classification of Text Messages

MTech CSE(AI)

Submitted by: Ranjan Kumar
Roll No: 2422118
Supervised by: Prof. Ramanujam Sir

1 Dataset Used

Dataset Details:

- Total samples: 2,000
- Class Distribution: 1,237 negative (ham), 763 positive (spam)
- Features: 8 features to be extracted from text messages.

1.1 Dataset Sample

The first five samples of the final dataset are shown below:

Message	Category
मास्टरी और अल्टीमेट लेग स्ट्रेंथ प्रोग्राम। अभी ट्राई करें!	spam
एक्सप्लोर करें और अपने बिजनेस के लिए सर्वश्रेष्ठ डील पाएं।	spam
अपने आप को सरकारी एजेंसीयों/रिज़र्व बैंक (RBI) के अधिकारी बताकर पैसे ट्रांसफ़र करने की मांग करने वाले साइबर अपराधियों के व	ham
हो सकते हैं और इसे दूरसंचार विभाग के चक्षु मॉड्यूल https://sancharsaathi.gov.in/sfc के माध्यम से सूचित करें। धोखाधड़ी के	ham
Jio के साथ जुड़े रहें और Jio की विश्व स्तरीय सेवाओं का आनंद लेते रहें जो आज भी सबसे किफ़ायती है	ham

Figure 1: Sample Dataset

1.2 Spam Distribution

- **Ham (Negative Messages):** 1,237 samples
- **Spam (Positive Messages):** 763 samples

2 Features Extracted

The following features were extracted from the messages:

- **message length:** The length of the message.
- **word_count:** Total number of words in the message.
- **has_url:** Whether the message contains a URL.
- **has_phone:** Whether the message contains a phone number.
- **has_money:** Whether the message mentions monetary values.
- **special_char_count:** Number of special characters in the message.
- **spam_word_count:** Number of spam-related words present.
- **TF-IDF score:** The Term Frequency-Inverse Document Frequency (TF-IDF) score tells the importance of words with respect to a document in a corpus.

```
print("Final Dataset Sample:")
print("-" * 50)
df=result_df
print(df.head())

print("\nSpam Distribution:")
print(df['is_spam'].value_counts())
```

```
Final Dataset Sample:
--------------------------------------------------
                                             Message  message_length  \
0  "बधाई हो! हमने सीरीज़ B फ़ंडिंग में $29 मिलियन...             207
1  अंतिम मौका! संचार मास्टरक्लास पर 70% की छूट, स...             178
2  फिटनेस पर 60% छूट! अपनी प्रीमियम मेंबरशिप अपग्...             178
3  Alibaba.com पर थोक में खरीदें। फैशन, प्लेड ड्...             180
4  अपने आप को सरकारी एजेंसीयों/रिज़र्व बैंक (RBI)...             200

   word_count  has_url  has_phone  has_money  special_char_count  \
0          38        0          0          1                   5
1          32        0          0          1                   7
2          27        0          0          0                   4
3          32        0          0          0                   4
4          34        0          0          0                   4

   spam_word_count  tfidf_score  is_spam
0                2         0.03        1
1                2         0.02        1
2                0         0.03        1
3                1         0.04        1
4                1         0.03        0

Spam Distribution:
is_spam
0    1237
1     763
Name: count, dtype: int64
```

Figure 2: Extracted Features

3 Classification and Evaluation:

For classification, the Lazy Predict library was used to evaluate various machine learning models. The code and results are as follows:

Code Snippet

```
clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric=None)
models, predictions = clf.fit(X_train, X_test, y_train, y_test)

# Display results
print(models)
```

```
 97%|                                                                                 | 28/29 [01:05<00:01,  1.86s/it]
[LightGBM] [Info] Number of positive: 631, number of negative: 969
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.005695 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 4040
[LightGBM] [Info] Number of data points in the train set: 1600, number of used features: 166
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.394375 -> initscore=-0.428959
[LightGBM] [Info] Start training from score -0.428959
100%|                                                                                | 29/29 [01:06<00:00,  2.30s/it]
                               Accuracy  Balanced Accuracy  ROC AUC  F1 Score  \
Model
ExtraTreesClassifier               0.96               0.95     0.95      0.96
XGBClassifier                      0.92               0.91     0.91      0.92
RandomForestClassifier             0.92               0.91     0.91      0.92
LGBMClassifier                     0.91               0.90     0.90      0.91
NearestCentroid                    0.92               0.89     0.89      0.92
Perceptron                         0.90               0.89     0.89      0.90
AdaBoostClassifier                 0.91               0.89     0.89      0.90
BaggingClassifier                  0.90               0.89     0.89      0.90
DecisionTreeClassifier             0.89               0.87     0.87      0.88
PassiveAggressiveClassifier        0.88               0.86     0.86      0.88
SGDClassifier                      0.84               0.86     0.86      0.84
RidgeClassifierCV                  0.89               0.85     0.85      0.89
LinearDiscriminantAnalysis         0.88               0.85     0.85      0.88
RidgeClassifier                    0.88               0.85     0.85      0.88
BernoulliNB                        0.89               0.85     0.85      0.88
```

Figure 3: Code Snippet for Lazy Predict Evaluation

3.1 Observations

- The **ExtraTreesClassifier** achieved the highest accuracy (96%) and balanced accuracy (95%).
- The **RandomForestClassifier** achieved an accuracy of 92%.

4 Conclusion

The Lazy Predict evaluation provides a quick benchmarking of models. Based on the results:

- The **ExtraTreesClassifier** is recommended for its high accuracy and balanced accuracy.
- Future work could include addressing class imbalance and testing ensemble methods.