A REPORT ON

Smishing Detection Using Machine Learning-Based Natural Language Processing

Course: Artificial Intelligence (UCS411)

By:

Garvita Bhatnagar-102303538

Siddharth Malik-102303541

Kashvi Aggarwal-102303564

Under the guidance of

Dr. Ashish Bajaj

Department of Computer Science and Engineering

Department of Computer Science and Engineering
Thapar Institute of Engineering and Technology
Patiala, Punjab – 147004
April, 2025

TABLE OF CONTENTS

ABSTRACT

This project explores the application of machine learning techniques to detect phishing (spam) messages in SMS communications. By leveraging the SMS Spam Collection dataset, we preprocessed text data, extracted features using TF-IDF vectorization, and trained multiple Naive Bayes models. The experimental results demonstrate the effectiveness of simple yet robust models in identifying malicious content with high accuracy, paving the way for smarter communication filtering systems. In addition to classification, we conducted exploratory data analysis to understand linguistic patterns in spam messages. The insights gained from this project can assist in building lightweight, scalable tools for real-time SMS phishing detection in mobile environments.

INTRODUCTION

Smishing attacks remain one of the most widespread and dangerous forms of cybercrime in the digital age. These attacks exploit human psychology by tricking individuals into revealing sensitive personal information such as passwords, credit card numbers, and social security details. Smishing has rapidly evolved and now commonly appears in text messages. Given the ubiquity of smartphones and the trust users often place in text message communication, smishing has become a serious threat to both individual and organizational security.

As communication becomes increasingly digital, there is a growing need for intelligent systems capable of detecting and mitigating these threats automatically. Manual detection is both impractical and inefficient due to the massive volume of messages transmitted daily. This necessitates the development of automated, data-driven approaches that can accurately identify phishing content based on linguistic patterns, message structure, and statistical features.

In recent years, Artificial Intelligence (AI) and Natural Language Processing (NLP**)** have emerged as powerful tools in the fight against cybercrime. By analyzing text data at scale and learning from large labeled datasets, AI models can distinguish between legitimate (ham) and malicious (spam) content. These models benefit from rapid processing capabilities, scalability, and the ability to adapt to new forms of attacks with retraining.

This project aims to develop a machine learning-based system for smishing message detection using the SMS Spam Collection dataset, a widely used corpus of labeled text messages. We explore and compare the effectiveness of various Naive Bayes classifiers, which are particularly well-suited for text classification problems due to their simplicity and robustness.

The primary goal is to evaluate how accurately these models can identify spam messages based on linguistic features, while also considering the computational efficiency of the algorithms. We further analyze the dataset to extract useful insights about the nature of spam messages—such as their average length and vocabulary—and use visual tools like word clouds and confusion matrices to illustrate our findings.

By the end of this project, we aim to demonstrate that effective phishing detection can be achieved using relatively simple machine learning models when combined with appropriate preprocessing and feature engineering. This project not only contributes to the broader field of cyber threat detection but also provides a practical foundation for developing deployable SMS spam filters for consumer and enterprise use.

METHODOLOGY

The core objective of this research project is to design and evaluate a smishing detection system for SMS messages using deep learning-inspired natural language processing (NLP) methods and machine learning classifiers. The system follows a systematic, modular pipeline that begins with data collection and ends with model evaluation. The simplicity of the model architecture ensures that it can be easily deployed in lightweight environments, such as mobile phones or cloud-based SMS filters.

1. Data Collection

The starting point of our methodology is the acquisition of a high-quality dataset. We employed the **SMS Spam Collection Dataset** from Hugging Face, which is widely used in spam classification research. The dataset contains **5,572 SMS messages**, each labeled as either:

- ham: Non-spam (legitimate message)
- spam: Malicious or phishing content

This dataset is particularly suitable for our study due to its:

- Balanced label distribution
- Real-world message content
- Public availability

The dataset was loaded into a pandas DataFrame for exploration and processing. An initial analysis of the label distribution was conducted using pie charts, which revealed a moderate class imbalance (approximately 87% ham and 13% spam).

2. Data Preprocessing

Preprocessing is a critical phase in any NLP project. Raw text contains noise that can negatively impact the performance of machine learning algorithms. Our preprocessing pipeline included the following steps:

2.1 Label Encoding

The categorical labels (ham, spam) were converted into binary numeric labels:

- ham → 0
- spam→1

2.2 Message Length Analysis

We added derived features like:

- **Character Length**: Total number of characters in the message

- **Word Count**: Number of words using tokenization
- **Sentence Count**: Number of sentences using sentence tokenization These features are useful as spam messages tend to be longer or have repetitive patterns to attract attention.

2.3 Tokenization and Cleaning

We utilized NLTK's tokenizer to break messages into word-level tokens. Cleaning involved:

- Removing punctuation
- Converting text to lowercase
- Removing stopwords
- Stemming with **Porter Stemmer**, reducing words to their base/root form This reduces dimensionality and groups semantically similar terms (e.g., "running" and "runner" → "run").

2.4 Visual Analysis

To better understand spam message characteristics, we used the WordCloud library to generate a visual representation of the most common words in spam messages. Terms like "free", "win", "urgent", "prize", and "claim" appeared frequently, confirming the manipulative vocabulary typical of smishing attempts.

3. Feature Extraction

With the cleaned and tokenized text, the next step was to transform it into numerical features using **TF-IDF (Term Frequency-Inverse Document Frequency)** vectorization.

Why TF-IDF?

Unlike simple bag-of-words (BoW) models, TF-IDF penalizes commonly occurring words across documents while emphasizing rare, yet significant terms. This is crucial in smishing detection, where certain words (like "claim", "win", or "limited") have high predictive power but might not appear in every message.

We applied the TfidfVectorizer() from sklearn.feature_extraction.text, which generates a sparse matrix representing the TF-IDF score for each word in the corpus.

4. Model Selection and Training

To test the effectiveness of classical machine learning in smishing detection, we selected three types of **Naive Bayes classifiers**:

4.1 Multinomial Naive Bayes

- Best suited for count-based or frequency-based features like TF-IDF
- Assumes the features (words) follow a multinomial distribution

- Performed best in our experiments

4.2 Bernoulli Naive Bayes

- Operates on binary feature vectors (word present or not)
- Works well when text classification focuses on the presence/absence of particular keywords

Each model was trained using an **80:20 train-test split**. The training process involved feeding the TF-IDF-transformed messages and corresponding labels to the classifier.

5. Evaluation Metrics and Analysis

To measure how well our models performed, we used several evaluation metrics and visualizations.

5.1 Accuracy

Measures the percentage of correct predictions. While easy to interpret, accuracy alone can be misleading in imbalanced datasets like ours.

5.2 Precision, Recall, and F1-Score

- **Precision**: Proportion of predicted spam messages that were actually spam
- **Recall**: Proportion of actual spam messages correctly identified
- **F1-Score**: Harmonic mean of precision and recall; balances the two

These metrics are especially important in spam detection, where:

- High precision minimizes **false alarms** (classifying ham as spam)
- High recall ensures **no spam is missed**

5.3 Confusion Matrix

We generated confusion matrices to visualize true positives, false positives, false negatives, and true negatives. These matrices help explain **where the model makes mistakes**—such as misclassifying promotional ham messages as spam.

EXPERIMENTAL DESIGN

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score,classification_report
from wordcloud import WordCloud
wc=WordCloud(width=500,height=500,min_font_size=10,background_color='white')
import pickle
import seaborn as sns
import nltk
from nltk.corpus import stopwords
nltk.download('punkt_tab')
import string
from nltk.stem.porter import PorterStemmer
ps=PorterStemmer()
vectorizer=TfidfVectorizer()
```

```
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

```
df=pd.read_csv('SMSSpamCollection.txt', sep='\t', header=None, names=['label', 'message'])
df.columns=['label', 'message']
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   label    5572 non-null   object
 1   message  5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

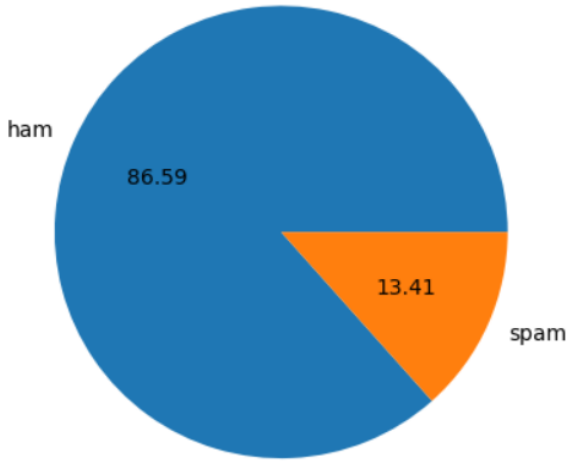
```
df['label_num'] = df['label'].map({'ham': 0, 'spam': 1})
df.sample(5)
```

	label	message	label_num
4607	ham	Oh... Haha... Den we shld had went today too.....	0
574	ham	Waiting for your call.	0
1268	ham	SERIOUSLY. TELL HER THOSE EXACT WORDS RIGHT NOW.	0
4150	ham	Haven't found a way to get another app for you...	0
4207	ham	Get the door, I'm here	0

```
df['label_num'].value_counts()
```

```
label_num
0    4825
1     747
Name: count, dtype: int64
```

```
plt.pie(df['label_num'].value_counts(),labels=['ham','spam'],autopct="%0.2f")
plt.show()
```




```
df['length']=df['message'].apply(len)
df['num_words']=df['message'].apply(lambda x:len(nltk.word_tokenize(x)))
df['num_sent']=df['message'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

```
df.head()
```

	label	message	label_num	length	num_words	num_sent
0	ham	Go until jurong point, crazy.. Available only ...	0	111	24	2
1	ham	Ok lar... Joking wif u oni...	0	29	8	2
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	1	155	37	2
3	ham	U dun say so early hor... U c already then say...	0	49	13	1
4	ham	Nah I don't think he goes to usf, he lives aro...	0	61	15	1

```
df[['length','num_words','num_sent']].describe()
```

	length	num_words	num_sent
count	5572.000000	5572.000000	5572.000000
mean	80.489950	18.842426	2.006102
std	59.942907	13.851947	1.539977
min	2.000000	1.000000	1.000000
25%	36.000000	9.000000	1.000000
50%	62.000000	15.000000	2.000000
75%	122.000000	27.000000	3.000000
max	910.000000	220.000000	38.000000

```
df[df['label_num']==0][['length','num_words','num_sent']].describe()
```

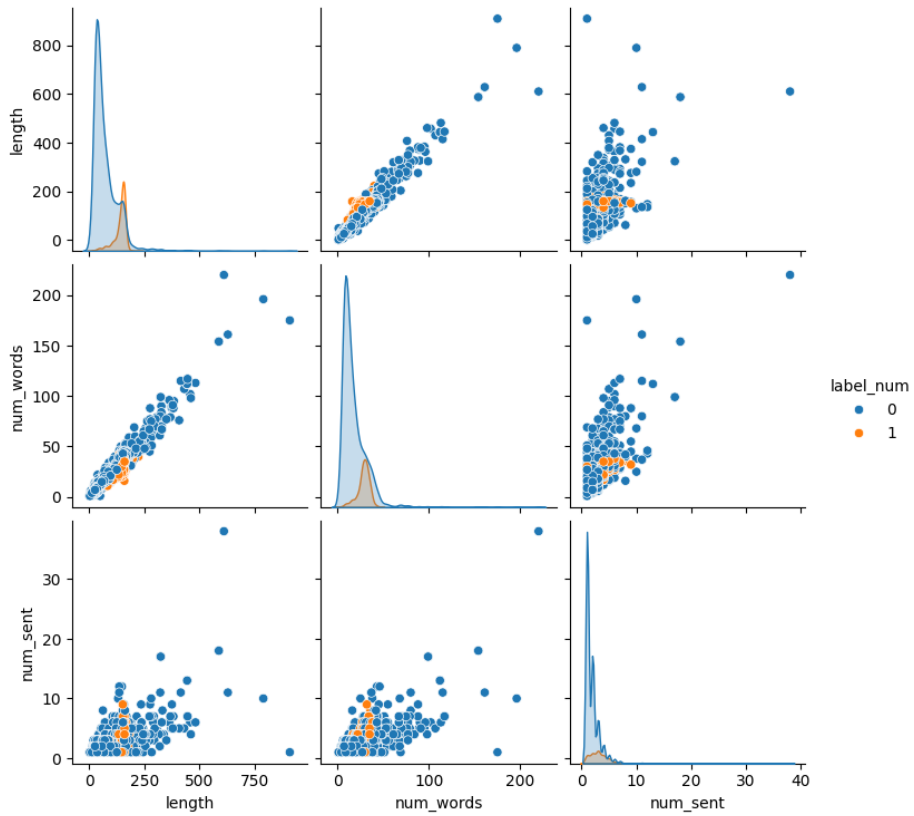
	length	num_words	num_sent
count	4825.000000	4825.000000	4825.000000
mean	71.482487	17.425699	1.846010
std	58.440652	14.118631	1.475377
min	2.000000	1.000000	1.000000
25%	33.000000	8.000000	1.000000
50%	52.000000	13.000000	1.000000
75%	93.000000	23.000000	2.000000
max	910.000000	220.000000	38.000000

```
df[df['label_num']==1][['length','num_words','num_sent']].describe()
```

	length	num_words	num_sent
count	747.000000	747.000000	747.000000
mean	138.670683	27.993307	3.040161
std	28.873603	6.860440	1.548499
min	13.000000	2.000000	1.000000
25%	133.000000	25.000000	2.000000
50%	149.000000	29.000000	3.000000
75%	157.000000	32.000000	4.000000
max	223.000000	46.000000	9.000000

```
sns.pairplot(df,hue='label_num')
```

```
<seaborn.axisgrid.PairGrid at 0x2c7366166c0>
```



```
df.select_dtypes(include=['number']).corr()
```

	label_num	length	num_words	num_sent
label_num	1.000000	0.381937	0.259958	0.264230
length	0.381937	1.000000	0.966435	0.630641
num_words	0.259958	0.966435	1.000000	0.685898
num_sent	0.264230	0.630641	0.685898	1.000000

```
def transform_sms(text):
    text=text.lower()
    text=nltk.word_tokenize(text)
    y=[]
    for i in text:
        if i.isalnum():
            y.append(i)
    text=y[:]
    y.clear()
    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)
    text=y[:]
    y.clear()
    for i in text:
        y.append(ps.stem(i))
    return " ".join(y)
```

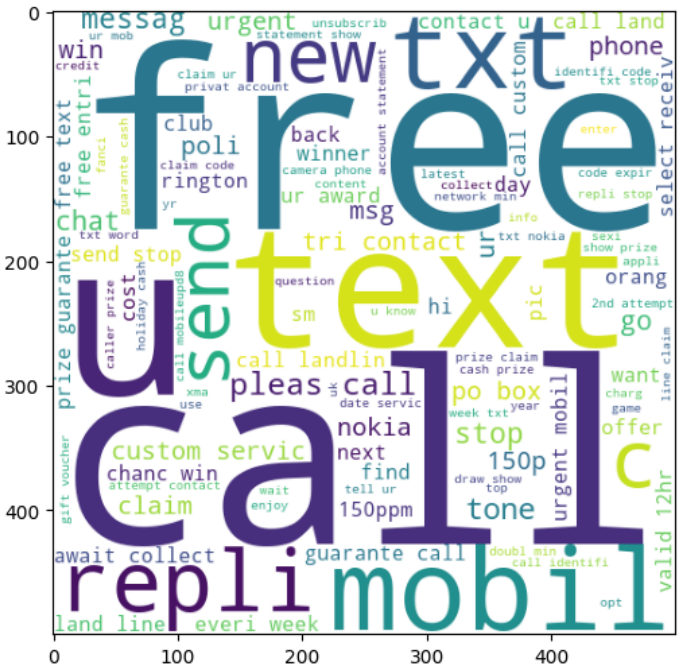
```
df['transformed']=df['message'].apply(transform_sms)
```

```
df.head()
```

	label	message	label_num	length	num_words	num_sent	transformed
0	ham	Go until jurong point, crazy.. Available only ...	0	111	24	2	go jurong point crazi avail bugi n great world...
1	ham	Ok lar... Joking wif u oni...	0	29	8	2	ok lar joke wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	1	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	ham	U dun say so early hor... U c already then say...	0	49	13	1	u dun say earli hor u c alreadi say
4	ham	Nah I don't think he goes to usf, he lives aro...	0	61	15	1	nah think goe usf live around though

```
spam_wc=wc.generate(df[df['label_num']==1]['transformed'].str.cat(sep=" "))
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

```
<matplotlib.image.AxesImage at 0x2c73885dac0>
```



```
ham_wc=wc.generate(df[df['label_num']==0]['transformed'].str.cat(sep=" "))
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
```

```
<matplotlib.image.AxesImage at 0x2c7388cf680>
```


```
X=df['message']
y=df['label_num']
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

```
vectorizer=TfidfVectorizer()
X_train_vectorized=vectorizer.fit_transform(X_train)
```

```
model=BernoulliNB()
model.fit(X_train_vectorized, y_train)
X_test_vectorized=vectorizer.transform(X_test)
y_pred=model.predict(X_test_vectorized)
print(confusion_matrix(y_test,y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
[[956   1]
 [ 31 127]]
Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       957
           1       0.99      0.80      0.89       158

    accuracy                           0.97      1115
   macro avg       0.98      0.90      0.94      1115
weighted avg       0.97      0.97      0.97      1115
```

```
model=MultinomialNB()
model.fit(X_train_vectorized, y_train)
X_test_vectorized=vectorizer.transform(X_test)
y_pred=model.predict(X_test_vectorized)
print(confusion_matrix(y_test,y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
[[957   0]
 [ 53 105]]
Classification Report:
              precision    recall  f1-score   support

           0       0.95      1.00      0.97       957
           1       1.00      0.66      0.80       158

    accuracy                           0.95      1115
   macro avg       0.97      0.83      0.89      1115
weighted avg       0.95      0.95      0.95      1115
```

```
with open('vectorizer.pkl', 'wb') as vec_file:
    pickle.dump(vectorizer, vec_file)
with open('model.pkl', 'wb') as model_file:
    pickle.dump(model, model_file)
```

```
def predict_spam_ham(text_message):
    with open('vectorizer.pkl', 'rb') as vec_file:
        loaded_vectorizer = pickle.load(vec_file)
    with open('model.pkl', 'rb') as model_file:
        loaded_model = pickle.load(model_file)
    input_vector = loaded_vectorizer.transform([text_message])
    prediction = loaded_model.predict(input_vector)[0]
    return "SPAM" if prediction == 1 else "HAM"
```

```
while True:
    user_input=input("Enter an SMS message (or type 'exit' to quit):\n")
    if user_input.lower() == 'exit':
        break
    prediction = predict_spam_ham(user_input)
    print("Prediction:", prediction)
```

```
Enter an SMS message (or type 'exit' to quit):
 Can you send me 100 bucks pls
Prediction: HAM
Enter an SMS message (or type 'exit' to quit):
 congrats on your promotion!
Prediction: HAM
```

CONCLUSION

In this project, we successfully developed a machine learning-based system capable of classifying SMS messages as either spam or ham. The development process involved several key stages, including Exploratory Data Analysis (EDA), data preprocessing, and tokenization. EDA allowed us to gain meaningful insights into the characteristics of spam versus ham messages, such as differences in word count and message length. Preprocessing steps, including removal of noise, handling of stopwords, and normalization helped clean and structure the raw text data, making it suitable for feature extraction.

To convert textual data into a numerical format that could be used by machine learning algorithms, we employed the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique. This helped highlight the importance of specific words in each message while reducing the influence of commonly occurring but less informative terms.

Several classification algorithms were tested, with the Multinomial Naive Bayes model demonstrating the highest precision and computational efficiency. This made it particularly suitable for our application, as spam detection systems require high accuracy with fast decision-making. Our evaluation using metrics like the confusion matrix, precision, recall, and F1-score confirmed that the model performed well in distinguishing spam from ham messages.

The successful deployment of this model highlights the strong potential of machine learning techniques in automating spam detection, thereby improving user experience and enhancing security in communication systems. For future work, the project could be extended by incorporating a larger and more diverse dataset, experimenting with deep learning models such as LSTMs or transformers for even better context understanding, and adapting the system for real-time deployment in messaging platforms or mobile applications.