



**slington college**  
(इस्लिंग्टन कलेज)

**Module Code & Module Title**

**CU6051NI Artificial Intelligence**

**60% Individual Coursework**

**Submission: Final Submission**

**Academic Semester: Autumn Semester 2025**

**Credit: 15 credit semester long module**

**Student Name: Prashant Rijal**

**London Met ID: 23048683**

**College ID: np01ai4a230142**

**Assignment Due Date: 19/01/2026.**

**Assignment Submission Date: 19/01/2026**

**Submitted To: Mahotsav Bhattarai**

*I confirm that I understand my coursework needs to be submitted online via MST Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.*

## Table of Contents

<b>1</b>	<b>Introduction.....</b>	<b>1</b>
1.1	Machine Learning and NLP .....	1
<b>2</b>	<b>Problem Domain.....</b>	<b>2</b>
2.1	Problem Explanation.....	2
2.2	Dataset Description .....	2
2.3	Societal or Business Relevance.....	3
<b>3</b>	<b>Solution.....</b>	<b>4</b>
3.1	Text Preprocessing Techniques.....	4
3.1.1	Normalization (Lowercasing) .....	4
3.1.2	Removing Punctuation and numbers.....	4
3.1.3	Stopword Removal .....	4
3.1.4	Stemming.....	4
3.2	Feature Extraction .....	5
3.2.1	TF-IDF Vectorization .....	5
3.3	Machine Learning Algorithm/Models .....	5
3.3.1	Multinomial Naïve Bayes: .....	5
3.3.2	Logistic Regression: .....	5
3.3.3	KNN (K Nearest Neighbour): .....	6
3.4	Development Process and Technologies .....	7
3.4.1	Tools and Technologies Used.....	7
3.4.2	Development Pipeline .....	8
3.5	Pseudocodes .....	10
3.5.1	Multinomial Naïve Bayes .....	10
3.5.2	Logistic Regression .....	12
3.5.3	K-Nearest Neighbour .....	14
3.6	State Transition Diagram .....	16
<b>4</b>	<b>Results .....</b>	<b>17</b>
4.1	Model evaluation .....	17
4.2	Comparative analysis .....	19

<b>4.3</b>	<b>Testing.....</b>	<b>20</b>
4.3.1	Testing on Unseen Data .....	20
<b>5</b>	<b>Conclusion .....</b>	<b>22</b>
5.1	Future Work.....	22
	<b>References .....</b>	<b>23</b>

## Table of Figures

Figure 1 Class Distribution .....	2
Figure 2 Frequent words .....	3
Figure 3 logistic regression example figure .....	6
Figure 4 K Nearest Neighbour .....	6
Figure 5 Development Flow .....	9
Figure 6 fig. Multinomial Naïve Bayes Flowchart .....	11
Figure 7 fig. Logistic Regression Flowchart.....	13
Figure 8 fig. KNN flowchart.....	15
Figure 9 fig. Overall System State Transition Diagram.....	16
Figure 10 Multinomial Naive Bayes Performance .....	17
Figure 11 Multinomial Naive Bayes Confusion Matrix .....	17
Figure 12 Logistic Regression Performance .....	18
Figure 13 KNN Perofrmance Index .....	18
Figure 14 Comparative ROC curve .....	19
Figure 15 Comparative Confusion Matrix .....	20
Figure 16 Unseen Data Testing .....	20
Figure 17 Confidence Score on Unseen Data .....	21

## Table Of Tables

Table 1 tbl. Technology Table .....	7
Table 2 Analysis of ML Models .....	19

# 1 Introduction

This project aims to design and develop a machine learning application which is capable of classifying SMS messages as either malicious messages (**Spam**) or real/legit messages (**Ham**), due to the growth of mobile communication SMS has become one of the primary targets for phishing attacks and unprompted marketing. This project aims to filter these types of messages in order to make user experience secure and solicit.

## 1.1 Machine Learning and NLP

The project uses **Supervised Machine Learning** in which the model learns to map input data(Features) to output labels(Target Variable) based on the labelled dataset. For the project following concepts will be utilized:

- Natural Language Processing (NLP):
  - It is the field of AI focused on helping computers understand human language.
  - The use of NLP in this project is to clean the data (raw text) and convert it to numerical format for the computer to understand.
- Vectorization:
  - It is the process of converting non numerical data into numerical vectors for processing.
  - For the project TF-IDF (Term Frequency-Inverse Document Frequency) will be used to transform text into numerical vectors.
  - TF-IDF's main concept is to assign weight based on their importance.
- Classification Algorithm:
  - They are tools in ML which sorts data into categories or classes.
  - The project uses 3 distinct types of learning algorithms for classification model:
    - Naïve Bayes (Probabilistic)
    - Logistic Regression (Linear)
    - KNN K-Nearest Neighbour (Instance based)

## 2 Problem Domain

### 2.1 Problem Explanation

The **SMS Spam Detection** is a type of binary classification problem. The input is a string of text or sentences (SMS message) and the output is a binary label in our case we assign “0 for Ham” and “1 for Spam”. The unstructured structure of the SMS data is a challenge as it contains less characters more often than not messages containing abbreviation, slangs, and misspelling with the addition of lack of metadata.

### 2.2 Dataset Description

Dataset used: UCI SMS Spam Collection

The project utilizes the UCI SMS Spam Collection which is a public dataset sourced from University of California, Irvine’s (UCI) Machine Learning Repository.

Here are some facts that we can get from the dataset: (shown in figure 1)

- Type: .txt (the original dataset is in txt file format)
- No of rows of data: 5572
- Data Type: Unstructured English data (object)
- Class Imbalance: Heavily Imbalanced
- Ham=4824 (86.591276%)
- Spam=747 (13.408724%)
- Implications: Due to the heavy class imbalance the model may have less accuracy so Precision and Recall metrics will be more important than accuracy during the evaluation of the model.

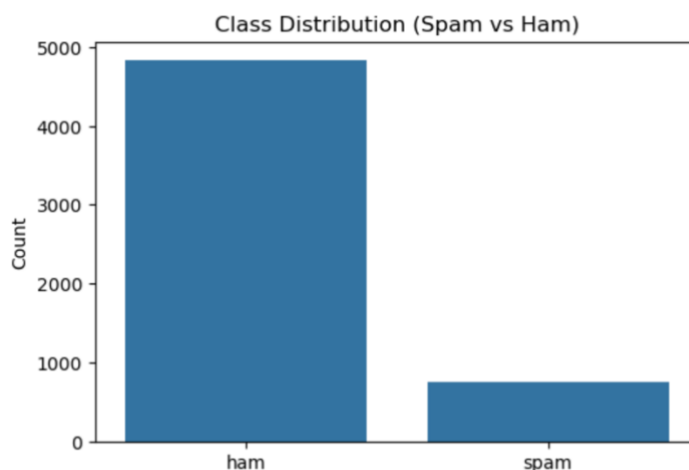


Figure 1 Class Distribution

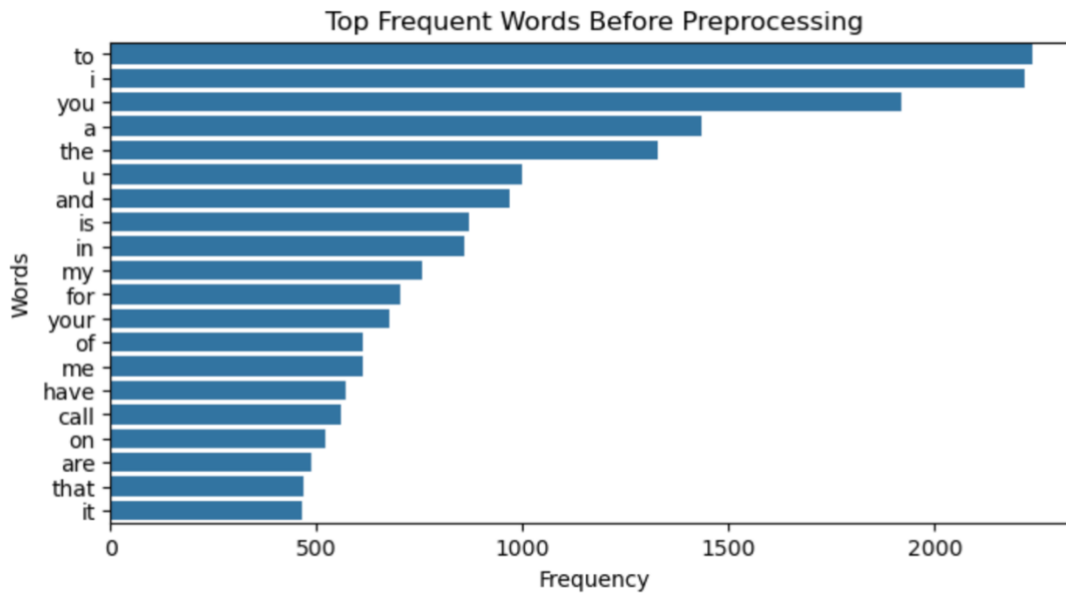


Figure 2 Frequent words

### 2.3 Societal or Business Relevance

- **Cybersecurity:** SMS phishing is a major threat in current time as it is used to steal banking details or other private information, this classifier is like the first line of defence against these types of threats.
- **Business Integrity:** Even now almost all top businesses and companies such as X, Meta etc rely on SMS for one time OTP but if due to constant spam email threats users stop trusting or using SMS it compromises security.
- **User friendly:** the spam filters automatically filters spam SMS saving time for the user and also decluttering mobile storage.

### 3 Solution

The solution developed for this implements a full machine learning pipeline:

- Text Preprocessing
- Feature Extraction
- ML Algorithms

#### 3.1 Text Preprocessing Techniques

The raw data (text) is noisy so we apply the following techniques:

##### 3.1.1 Normalization (Lowercasing)

The data (messages) are in both upper and lower cases and if we tokenize it directly it will see for example “If” and “if” as two completely different tokens so in order to reduce the dimensionality we need to lowercase the label.

##### 3.1.2 Removing Punctuation and numbers

The labels have a lot of punctuations, numbers and also special characters such as @, \$, # etc which are not really that useful in our scenario so removing these make the labels more clear and easy to process.

##### 3.1.3 Stopword Removal

Stopwords are mostly considered unnecessary in NLP and so since our dataset has a lot of characters the less there is the more accurate it becomes and also the faster it is so we remove the Stopwords.

##### 3.1.4 Stemming

Stemming is the process of reducing a word to its root/base form, it mainly removes prefixes and is considered a major part for the dimensionality reduction, here we could use lemmatization too but it takes more time and the end results are almost similar so it is not the most efficient for this reason we are using stemming.

### 3.2 Feature Extraction

Since the data is in textual form, we need to convert it to numerical form and for that following techniques are to be used:

#### 3.2.1 TF-IDF Vectorization

It converts text into numerical vectors. (This method was chosen because it normalizes count of words which prevents longer messages from having unfair weightage.)

### 3.3 Machine Learning Algorithm/Models

Following machine learning algorithm are to be used:

#### 3.3.1 Multinomial Naïve Bayes:

- assumes independence between features
- Speed and high performance on high dimensionality data.
- Uses techniques like Laplace smoothing to handle unseen words preventing zero probabilities.

$$P(c | x) = \frac{P(x | c)P(c)}{P(x)}$$

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

#### 3.3.2 Logistic Regression:

- Learns by creating linear decision boundary between classes.
- Uses sigmoid function.

- Provides probabilities for prediction of class.

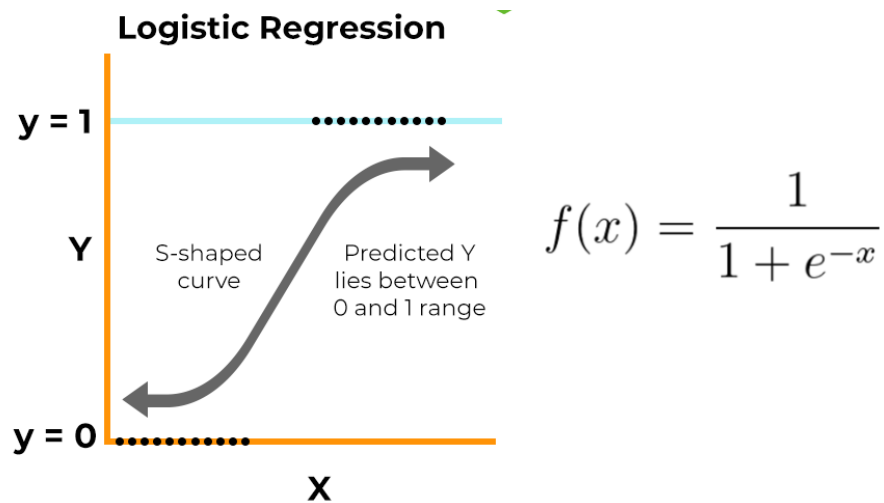


Figure 3 logistic regression example figure

### 3.3.3 KNN (K Nearest Neighbour):

- Classifies based on distance similarity.
- Instance based supervised machine learning algorithm.

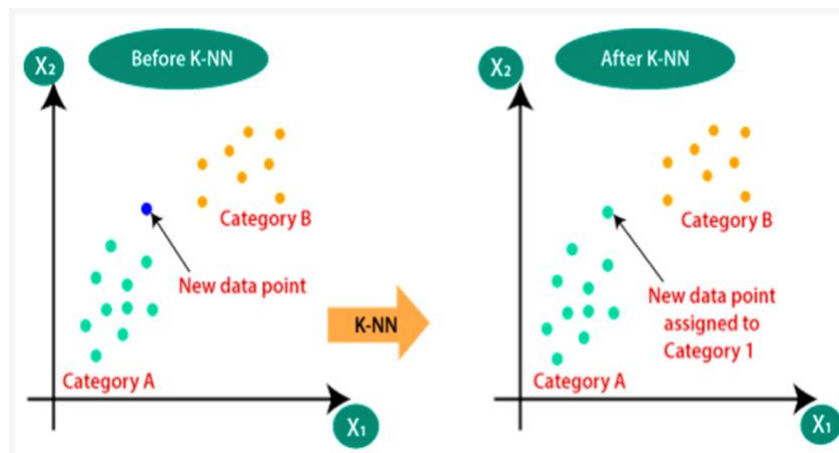


Figure 4 K Nearest Neighbour

### 3.4 Development Process and Technologies

#### 3.4.1 Tools and Technologies Used

For the completion of the project following tools and technologies were used:

- **Programming Language and Environment**
  - **Python 3.0:** This is primary language chosen for the project due to its easiness and vast libraries.
  - **Jupyter Notebook:** This is the IDE (Integrated Development Environment) chosen for the project which is due to its cell-based structure and other auxiliary functions.
- **Data Manipulation and Analysis:**
  - Pandas: used to load and transform the data
- **Machine Learning and NLP**
  - **Scikit-learn:** The most critical library for the project this library holds all the core logic of the models that we ran e.g.
    - TfidfVectorizer
    - Train Test Split
    - Multinomial Naive Bayes, Logistic Regression
    - Evaluation Matrices
  - **Nltk:** The primary library mostly used for text preprocessing tasks such as:
    - Stopword removal
    - Stemming
  - **re (Regular Expression):** The library used to handle removal of punctuations, numbers and special characters.

Table 1 tbl. Technology Table

Category	Technology
Language	Python
Environment	Jupyter Notebook
Data Cleaning	Pandas, Re (Regex)
NLP	NLTK (Stemming, Stopwords)
Vectorization	TF-IDF (Scikit-learn)
ML Algorithms	Naive Bayes, Logistic Regression, KNN

### 3.4.2 Development Pipeline

- **Phase 1: Data Loading and exploration**

This process began with loading the dataset in the notebook. During exploration many problems were found, and solutions were created such as the need of Latin-1 encoding for the dataset to load, the raw data being in comma separated text file form and other details such as the no of data, features, class imbalance etc.

- **Phase 2: Text Cleaning**

The next part is cleaning the text which includes:

- Lowercasing
- Removing signs, numbers and punctuations
- Stop words Removal
- Stemming

- **Phase 3: Tokenization:**

Though this phase we will be tokenizing the cleaned messages.

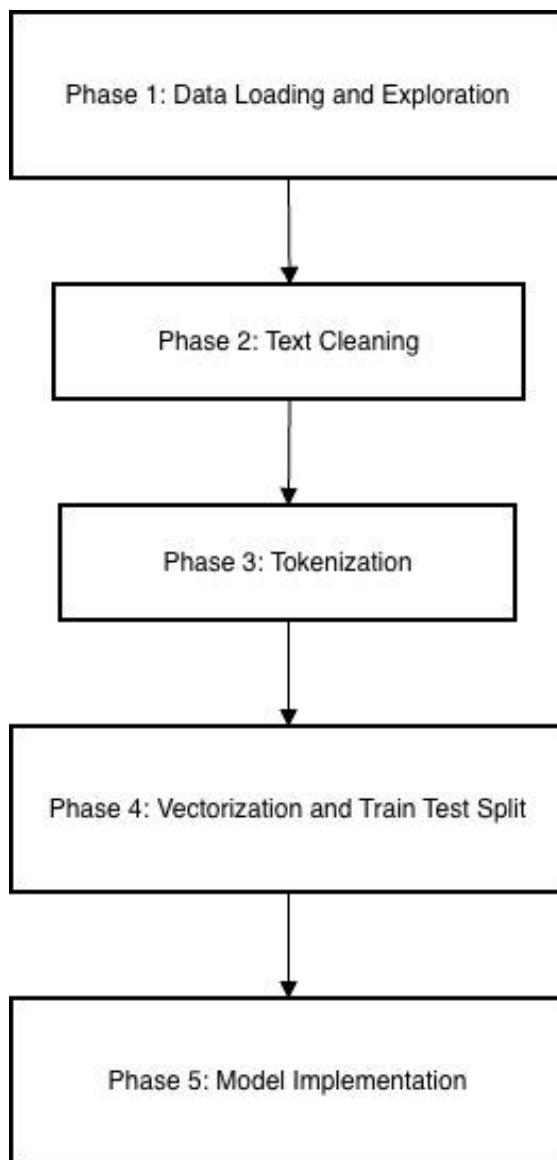
- **Phase 4: Vectorization and Train Test Split:**

In this phase the cleaned words are vectorized as the machine cannot directly read texts, after vectorizing we split the data into test and train data to train the models and test how they perform.

- **Phase 5: Model Implementation:**

Now that the data has been split into train and test data we go into training the data using the following models:

- **Multinomial Naive Bayes:**
- **Logistic Regression:**
- **KNN:**



*Figure 5 Development Flow*

### 3.5 Pseudocodes

#### 3.5.1 Multinomial Naïve Bayes

START

PROCESS NaiveBayes

    READ Training\_Data

    COMPUTE Prior\_Probabilities

        Total\_Count = Total number of messages

        Spam\_Prior = Count of Spam messages / Total\_Count

        Ham\_Prior = Count of Ham messages / Total\_Count

    BUILD Vocabulary

        Collect all unique words from all messages

        Set Vocabulary\_Size to count of unique word

    COMPUTE Word\_Likelihoods (with Laplace Smoothing)

        FOR each word in Vocabulary

            Spam\_Word\_Prob = (Count of word in Spam + 1) / (Total words in Spam + Vocabulary\_Size)

            Ham\_Word\_Prob = (Count of word in Ham + 1) / (Total words in Ham + Vocabulary\_Size)

    PREDICT New\_Message

        Tokenize New\_Message into words

        Initialize Spam\_Score = Log(Spam\_Prior)

        Initialize Ham\_Score = Log(Ham\_Prior)

        FOR each word in New\_Message

IF word exists in Vocabulary

Spam\_Score = Spam\_Score + Log(Spam\_Word\_Prob)

Ham\_Score = Ham\_Score + Log(Ham\_Word\_Prob)

IF Spam\_Score > Ham\_Score

RETURN "Spam"

ELSE

RETURN "Ham"

**END**

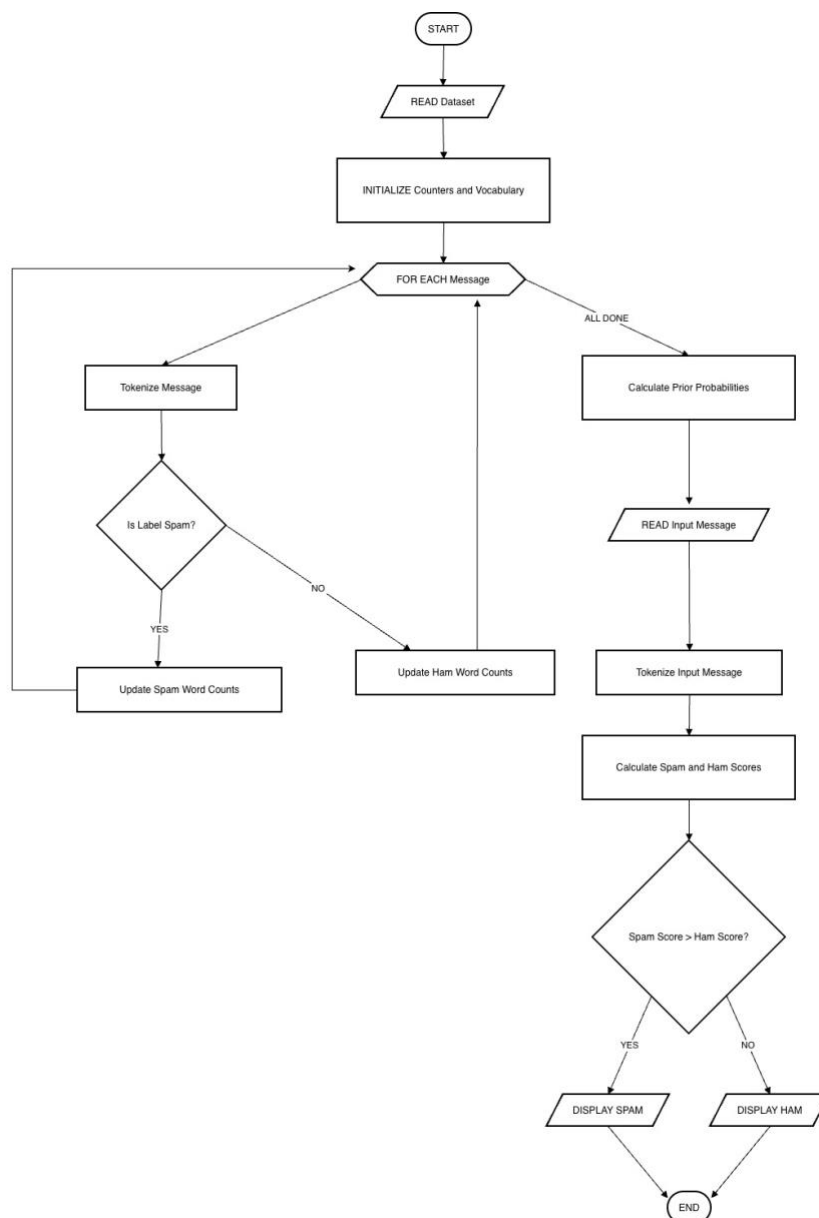


Figure 6 fig. Multinomial Naïve Bayes Flowchart

### 3.5.2 Logistic Regression

#### START

#### PROCESS LogisticRegression

##### INITIALIZE Parameters

Weights = Set all to 0

Bias = 0

Learning\_Rate = Set alpha (e.g., 0.01)

Iterations = Set total loops

##### TRAIN Model

##### FOR each iteration

##### FOR each record in Dataset

Calculate  $\text{Linear\_Output} = (\text{Weights} * \text{Input\_Features}) + \text{Bias}$

Calculate  $\text{Probability} = 1 / (1 + \exp(-\text{Linear\_Output}))$

Calculate  $\text{Error} = \text{Probability} - \text{Actual\_Label}$

##### Update\_Gradients

$\text{Weight\_Gradient} = \text{Input\_Features} * \text{Error}$

$\text{Bias\_Gradient} = \text{Error}$

##### Update\_Parameters

$\text{Weights} = \text{Weights} - (\text{Learning\_Rate} * \text{Weight\_Gradient})$

$\text{Bias} = \text{Bias} - (\text{Learning\_Rate} * \text{Bias\_Gradient})$

##### PREDICT New\_Message

Calculate  $\text{Linear\_Output} = (\text{Weights} * \text{New\_Vector}) + \text{Bias}$

Calculate  $\text{Final\_Prob} = 1 / (1 + \exp(-\text{Linear\_Output}))$

IF Final\_Prob  $\geq$  0.5

RETURN "Spam"

ELSE

RETURN "Ham"

**END**

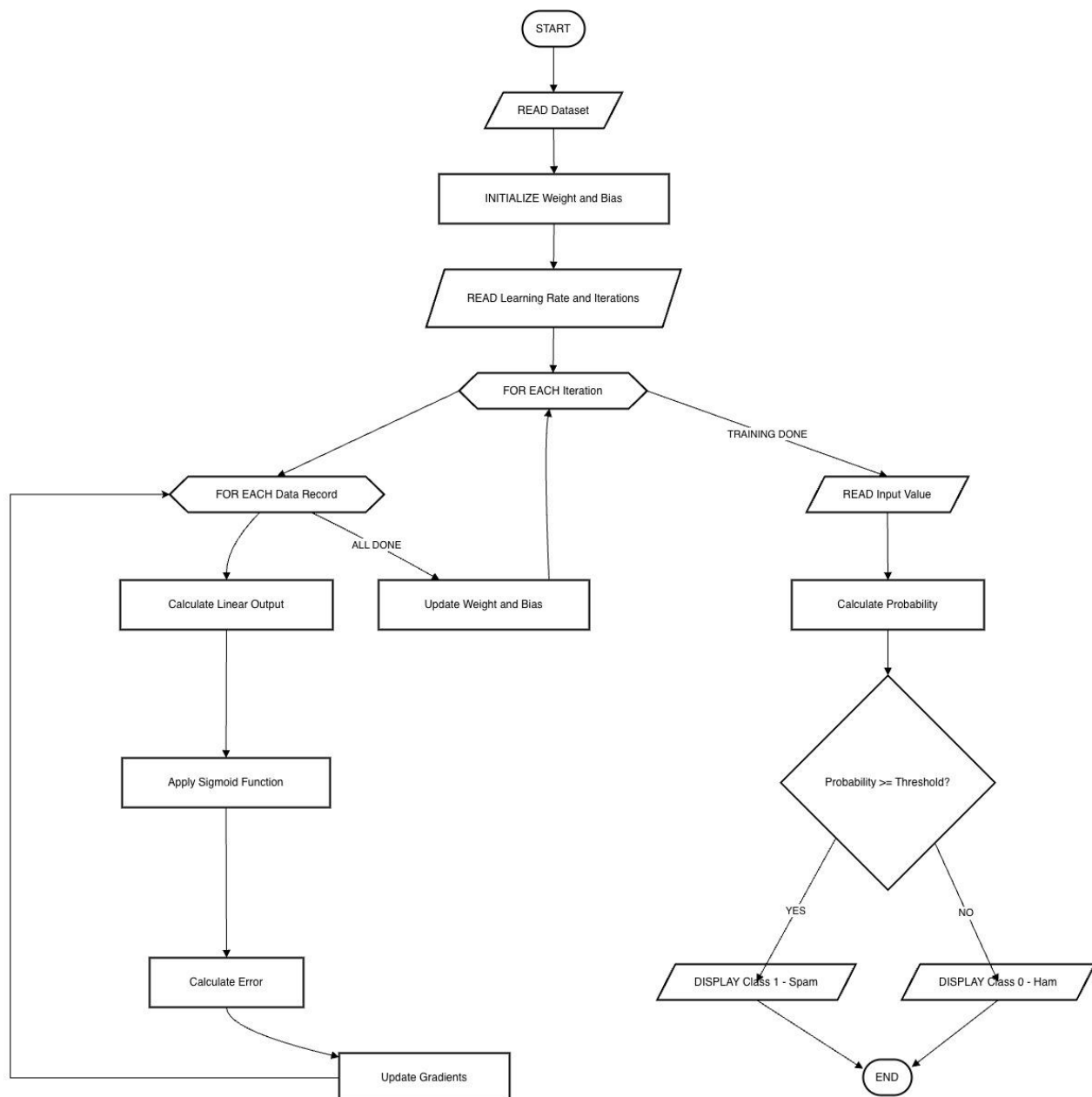


Figure 7 fig. Logistic Regression Flowchart

### 3.5.3 K-Nearest Neighbour

#### START

#### PROCESS KNN

STORE Training\_Data

Load all message vectors and their labels

Set K = number of neighbors (e.g., 5)

PREDICT New\_Message

Initialize Distance\_List = Empty list

Vector\_New = Convert new message to TF-IDF vector

FOR each message in Training\_Data

Calculate Distance = Euclidean distance between Vector\_New and Training\_Vector

Add (Distance, Label) to Distance\_List

SORT Distance\_List by Distance in ascending order

SELECT K\_Neighbors

Identify the first K items in Distance\_List

MAJORITY\_VOTE

Count occurrences of "Spam" in K\_Neighbors

Count occurrences of "Ham" in K\_Neighbors

IF Spam\_Count > Ham\_Count

RETURN "Spam"

ELSE

RETURN "Ham"

END



Figure 8 fig. KNN flowchart

### 3.6 State Transition Diagram

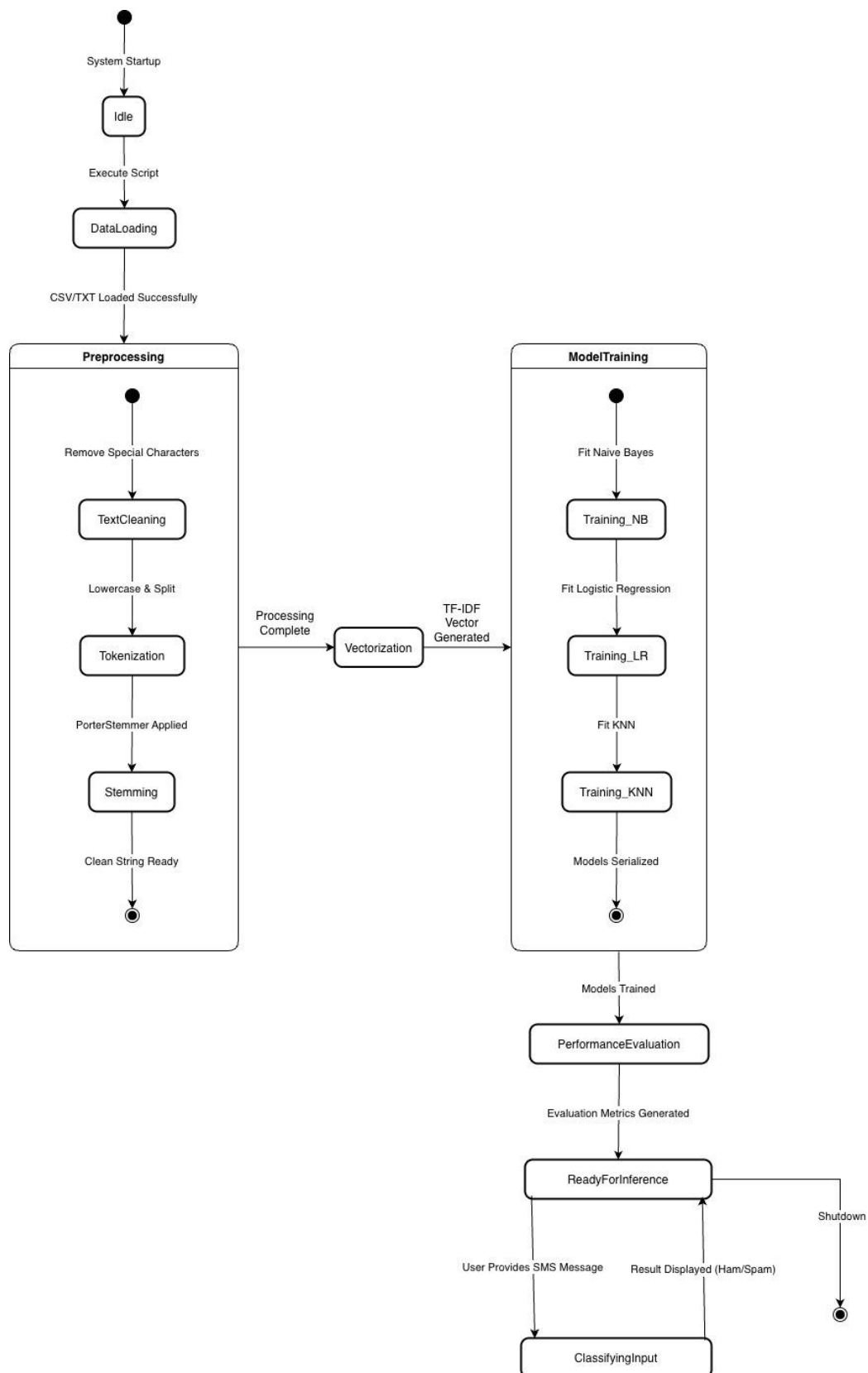


Figure 9 fig. Overall System State Transition Diagram

## 4 Results

This part entails the experimental results obtained after training and evaluating the different machine learning models. The performance of each model is analyzed using Confusion Matrix and ROC (Receiver Operating Characteristics), due to the imbalance in data more emphasis is placed on the precision and recall rather than accuracy.

### 4.1 Model evaluation

- Multinomial Naïve Bayes:

	precision	recall	f1-score	support
0	0.95	0.78	0.86	149
1	0.97	0.99	0.98	966
accuracy			0.97	1115
macro avg	0.96	0.89	0.92	1115
weighted avg	0.96	0.97	0.96	1115

Figure 10 Multinomial Naive Bayes Performance

This model demonstrated very good performance on the SMS spam classification task. It has achieved high accuracy and recall in identifying spam and a bit lower in ham. This result is expected as this model is well suited for such tasks.

The confusion matrix shows that the model correctly classified major ham messages while also maintaining a low number of false negatives for spam messages. This shows that the model is effective at detecting harmful SMS messages

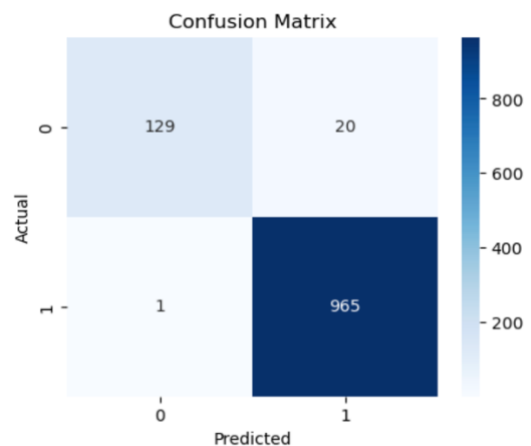


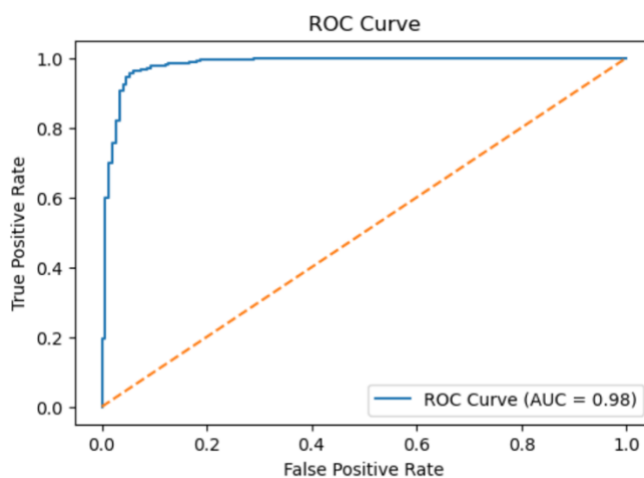
Figure 11 Multinomial Naive Bayes Confusion Matrix

- Logistic Regression:**

The Logistic regression model achieved performance comparable to multinomial naïve bayes with some minor improvement in precision in some cases. The model also has good accuracy but slightly lower than the naïve bayes model.

	precision	recall	f1-score	support
0	0.97	0.73	0.84	149
1	0.96	1.00	0.98	966
accuracy			0.96	1115
macro avg	0.97	0.86	0.91	1115
weighted avg	0.96	0.96	0.96	1115

Figure 12 Logistic Regression Performance



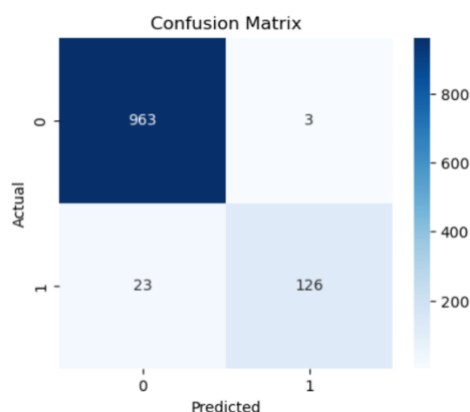
The ROC curve for logistic regression shows a high area under the curve which indicates high separability between spam and ham classes

- K-Nearest Neighbour:**

The KNN model shows similar performance on the dataset model just like naïve bayes and logistic regression models.

	precision	recall	f1-score	support
0	0.97	0.75	0.85	149
1	0.96	1.00	0.98	966
accuracy			0.96	1115
macro avg	0.96	0.87	0.91	1115
weighted avg	0.96	0.96	0.96	1115

Figure 13 KNN Performance Index



The confusion matrix for KNN indicates not many misclassifications particularly for ham messages. This is odd as it should be showing exactly the opposite results.

## 4.2 Comparative analysis

This comparative analysis of the three models highlights the strength and weakness of each approach for the task of SMS spam detection.

Table 2 Analysis of ML Models

Model	Strengths	Weaknesses
Multinomial Naïve Bayes	Fast, efficient, high recall for spam	Assumes feature independence
Logistic Regression	High accuracy, interpretable, strong ROC	Requires feature scaling and tuning
KNN	Simple to implement	Poor performance on high-dimensional text data

Both Multinomial Naïve Bayes and Logistic Regression outperform the KNN in every aspect in the unseen data testing, due to the heavy imbalance in data the recall for spam is crucially important as false negatives may allow ill intended messages to pass through. Overall the Multinomial Naïve Bayes model performs excellently among these.

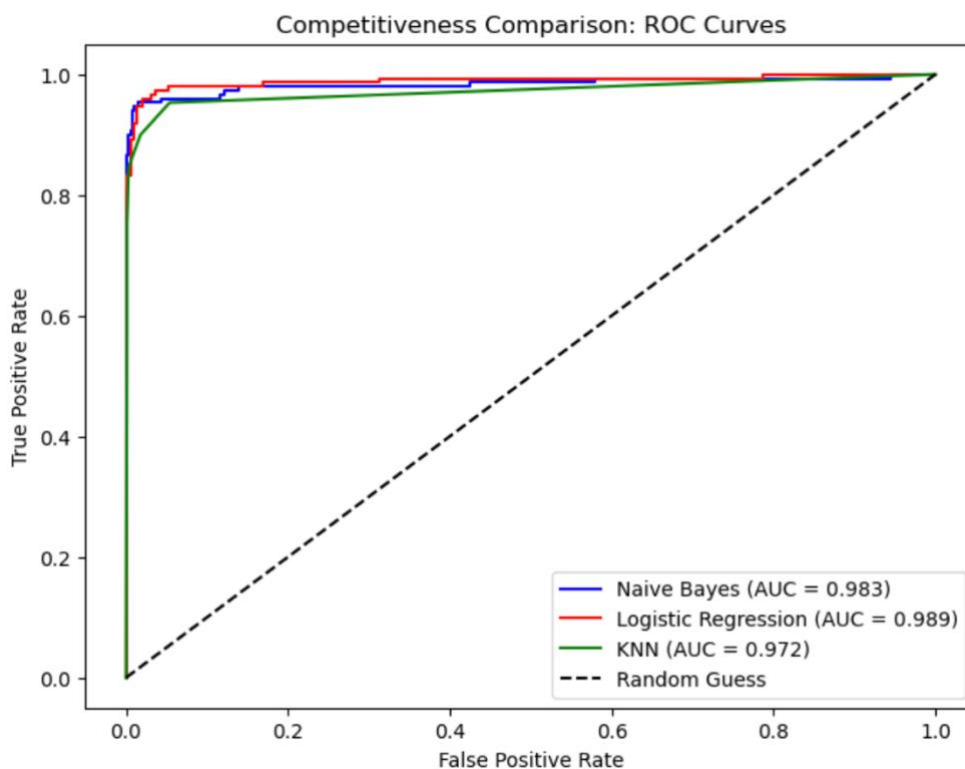


Figure 14 Comparative ROC curve

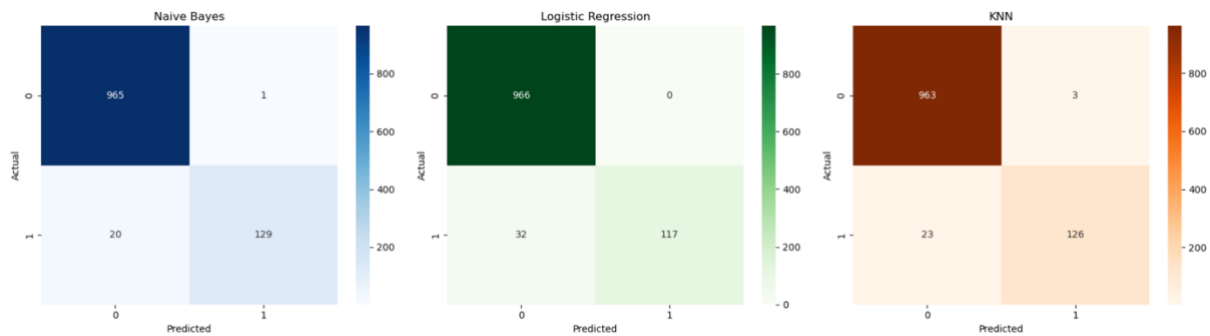


Figure 15 Comparative Confusion Matrix

### 4.3 Testing

The models after training were tested in both labelled and unseen datasets and the results of it are astounding and clearly shows which model is the best among these three.

#### 4.3.1 Testing on Unseen Data

Another testing process was done to test the generalization ability of the trained models through use of unseen SMS messages that were never a component of the training and testing set. This move resembles an actual deployment case, where the system will be required to categorize totally new message entries.

```
[167]: unseen_messages = [
    "Congratulations! You have won a free lottery ticket. Claim now!",
    "Hey, are we still meeting for lunch tomorrow?",
    "URGENT! Your bank account has been suspended. Click the link to verify.",
    "Can you send me the assignment notes?"
]

[169]: unseen_cleaned = [data_preprocessing(message) for message in unseen_messages]
unseen_cleaned = [data_preprocessing_1(messages) for messages in unseen_cleaned]

[171]: unseen_vectorized = tfidf.transform(unseen_cleaned)

[173]: models = {
    "Naive Bayes": nb_model,
    "Logistic Regression": lr_model,
    "KNN": knn_model
}

for name, mdl in models.items():
    preds = mdl.predict(unseen_vectorized)
    print(f"\n{name} Predictions:")
    for msg, pred in zip(unseen_messages, preds):
        print("Spam" if pred == 1 else "Ham")

Naive Bayes Predictions:
Spam
Ham
Spam
Ham

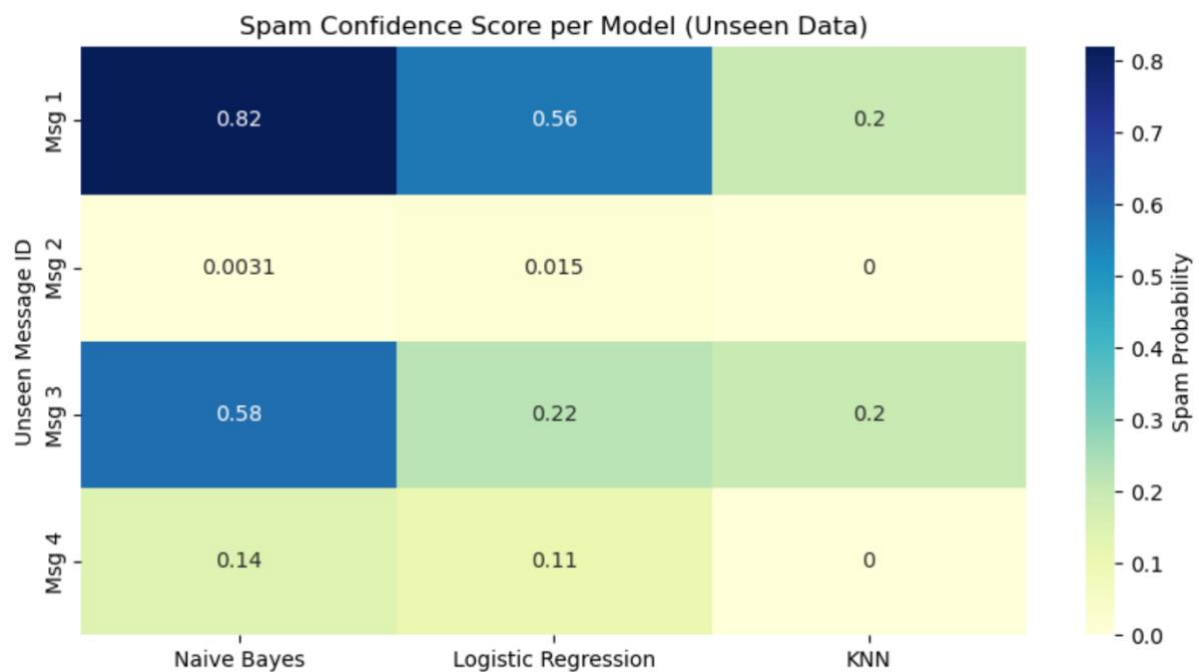
Logistic Regression Predictions:
Spam
Ham
Ham
Ham

KNN Predictions:
Ham
Ham
Ham
Ham
```

Figure 16 Unseen Data Testing

The identical preprocessing pipeline used in the case of the text , which requires text normalization, elimination of stopwords, and stemming, was used to the unseen messages to ensure consistency with the training data. The cleaned messages were then converted to numerical feature vectors of the TF-IDF pre-trained vectorizer.

The hidden messages were then predicted by the trained classification models to use in determining the class. These findings identified that the models could appropriately categorize the spam and ham messages with a high confidence threshold. Spam messages with keywords, promotional messages and messages with phishing were correctly detected and ham messages correctly detected. The capability of the Multinomial Naïve Bayes model to work effectively on unknown data has shown a good potential of application in the real world in SMS filtering systems.



*Figure 17 Confidence Score on Unseen Data*

## 5 Conclusion

The project was able to deploy a machine learning pipeline to recognize SMS messages by either classifying it as Ham or Spam extremely accurately. After comparing three algorithms Multinomial Naive Bayes, Logistic Regression, and K-Nearest Neighbors, the results showed that Naive Bayes and Logistic Regression were more applicable in this field of problem. The scores on recalls, especially the ones in the recognition of malicious spam, validate the fact that text preprocessing and TLT-IDF word vectors steps played an important role in converting unstructured text to meaningful features. Such a strict method enabled the models to solve the major issue of class imbalance in the dataset, both the genuine messages and the malicious ones will be filtered accordingly.

This application would be useful in the actual world as a first-level defense mechanism against both Smishing and automated phishing attacks and helps the users to avoid potential financial fraud and identity theft. Automation of the process of identifying unsolicited marketing and phishing links by the tool contributes to the safety of mobile communication to a large extent. The system may be extended with Deep Learning architectures, e.g. LSTMs or Transformers, to learn more about the semantics of messages. Also, the procedure of implementing the trained models as a real-time API would enable them to directly integrate it into messaging platforms to offer protection against any emerging cyber threats remotely.

### 5.1 Future Work

The current SMS spam detection system shows strong performance using traditional machine learning algorithms and thus there are several opportunities to enhance its capabilities and adapt it into a complex ecosystem in real world scenarios, some of them are:

- Integration of Deep Learning Models.
- Realtime Deployment and API Integration.
- Multi-Lingual handling.
- Hybrid models (Ensemble Learning Models)

## References

GitHub: [https://github.com/Prashant-Rijal-dev/SMS\\_Spam\\_Detection](https://github.com/Prashant-Rijal-dev/SMS_Spam_Detection)