SMS Spam Detection System Using NLP

A Project Report

submitted in partial fulfillment of the requirements

of

AICTE Internship on AI: Transformative Learning
with
TechSaksham – A joint CSR initiative of Microsoft & SAP

By

Mahek Shaikh

itsmaheksuhail@gmail.com

Under the Guidance of

Jay Rathod

ACKNOWLEDGEMENT

Firstly, I extend my heartfelt thanks to my supervisor, Mr. Jay Rathod Sir, for his exceptional guidance, constructive feedback, and consistent encouragement throughout the duration of this project. His expertise and valuable insights played a crucial role in shaping and successfully completing this project.

I also wish to acknowledge the TechSaksham initiative by Microsoft & SAP for providing this transformative learning opportunity, which greatly enriched my experience and learning.

Finally, I am deeply grateful to my family, peers, and friends for their unwavering support, encouragement, and belief in me, which has been a constant source of strength and motivation throughout this journey.

Mahek Shaikh

ABSTRACT

This project focuses on building a robust SMS spam detection system using machine learning techniques and a web-based interface. The primary objective is to create a model capable of accurately classifying text messages as either "spam" or "ham" (non-spam), an essential function for filtering unwanted or potentially harmful content in communication systems. To achieve this, several key steps were taken, including data preprocessing, feature extraction, model training, and the creation of an interactive user interface for real-time predictions.

The methodology starts with preprocessing raw text data, which involves converting the text to lowercase, tokenizing it, removing special characters, stopwords, and punctuation, and finally stemming the words using the Porter Stemmer algorithm. This cleaned and processed data is then transformed into numerical features using the TfidfVectorizer. Three machine learning models—Gaussian Naive Bayes (GNB), Multinomial Naive Bayes (MNB), and Bernoulli Naive Bayes (BNB)—were evaluated, with MNB achieving the highest accuracy and precision for spam classification.

Once the model was trained and evaluated, the project incorporated Streamlit to create a user-friendly interface for real-time spam detection. Streamlit allows users to input text messages, which are then preprocessed, vectorized, and classified by the trained model. The app provides immediate feedback, displaying either "Spam" or "Not Spam" based on the model's prediction. This web application is designed to be intuitive, with custom CSS used to enhance the user experience, including a visually appealing layout with green and blue colors.

The final result is a functional spam detection system that combines machine learning with a real-time web application for SMS message classification. The use of pickle files to save the vectorizer and model allows for easy deployment, and the combination of these technologies ensures that the system is both efficient and accessible for users.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1Problem Statement:

The problem being addressed is the detection of spam messages in SMS communication, which is crucial due to the increasing volume of unsolicited and harmful content sent daily. Spam messages, often containing unwanted advertisements, phishing attempts, or malicious links, pose significant security risks, such as data breaches and financial fraud.

Manually identifying spam is time-consuming and inefficient, especially as the number of messages grows. This problem is significant because it affects both user experience and platform security, making it essential to implement automated solutions that can accurately and efficiently classify messages as spam or non-spam. Leveraging machine learning for this task provides a scalable and reliable solution to protect users and improve communication systems.

1.2Motivation:

This project was chosen because of the growing prevalence of spam messages in SMS communication and the need for an effective, automated solution to address this issue. Spam messages not only disrupt user experience but also pose serious security threats, such as phishing attacks, fraud, and the spread of malware. As the volume of digital communication continues to rise, manual spam detection becomes increasingly impractical. By leveraging machine learning techniques, this project offers a scalable and efficient way to identify and classify spam messages, making it a timely and relevant solution for improving security and user satisfaction.

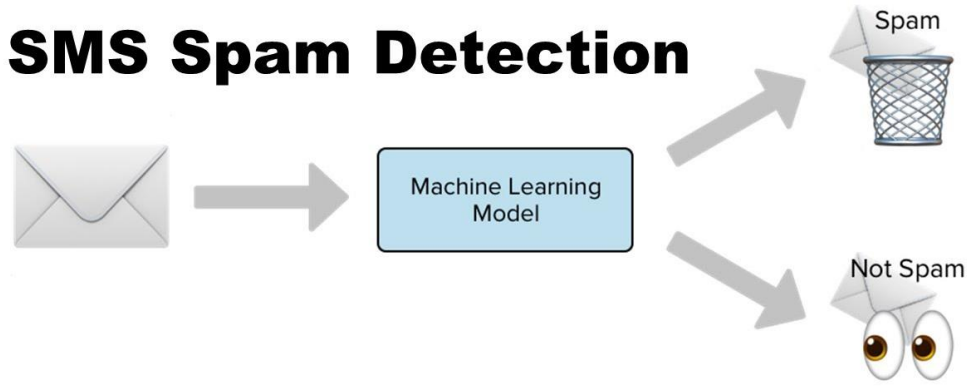

Fig 1 . SMS Spam Detection Basic Model

The potential applications of this project are vast. It can be integrated into messaging platforms and mobile applications to automatically filter out spam messages, enhancing the user experience and safeguarding against potential threats. Businesses can use this technology to protect customers from fraudulent messages and reduce the risk of phishing scams, thereby improving trust and communication quality. Additionally, it can be extended to other forms of messaging systems, including email and social media, to provide broader spam detection capabilities.

The impact of this project is significant, as it improves the overall security of digital communication, protects users from harmful content, and ensures more efficient use of messaging platforms. Ultimately, it contributes to a safer and more trustworthy digital environment.

1.3 Objective:

1. Develop a machine learning-based SMS spam detection system capable of classifying messages as "spam" or "ham" (non-spam).
2. Preprocess raw text data by cleaning, tokenizing, removing special characters, stop words, and stemming words to prepare them for analysis.
3. Extract meaningful features from the processed text using the TfidfVectorizer for use in machine learning models.
4. Train and evaluate multiple machine learning models, including Gaussian Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes, to determine the most effective model for spam classification.
5. Build an interactive web application using Streamlit that allows users to input text messages and receive real-time spam detection results.
6. Provide a user-friendly interface with customized styling to enhance the overall user experience.
7. Save the trained models and vectorizer as pickle files for easy deployment and future use in real-time applications.
8. Ensure the solution is scalable, efficient, and accurate in identifying spam messages in SMS communication.

1.4 Scope of the Project:

Scope:

1. **SMS Spam Detection**: The primary focus of this project is to classify SMS messages as "spam" or "ham" (non-spam) using machine learning models.
2. **Data Preprocessing**: The project involves cleaning and preparing text data by tokenizing, removing special characters, stopwords, punctuation, and stemming the words, ensuring that the data is suitable for training models.
3. **Machine Learning Models**: Three machine learning models—Gaussian Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes—are trained and evaluated to determine the best model for spam classification.
4. **Real-time Classification**: The project includes the development of a web interface using Streamlit, where users can input text messages and get real-time predictions of whether the message is spam or not.
5. **Pickling for Deployment**: The trained models and TfidfVectorizer are saved as pickle files, enabling easy deployment and reuse for future applications.

Limitations:

1. **Accuracy and Precision**: The model achieves limited precision and accuracy, with the best result being a precision of 9.45% and an accuracy of 9.64%. These values may not be sufficient for highly sensitive or large-scale applications.
2. **Data Dependency**: The performance of the model heavily depends on the quality and quantity of the dataset used for training. If the dataset is not diverse or representative, the model's ability to generalize to new data may be limited.
3. **Spam Variability**: The model may struggle to accurately detect more complex or evolving forms of spam, especially if they do not closely resemble the patterns seen in the training data.
4. **Scope of Text**: The solution is tailored to SMS messages and might not perform well on other forms of communication, such as emails or social media messages, without further adaptation.
5. **Limited Preprocessing**: The preprocessing steps, such as stemming and stopword removal, may result in the loss of some valuable context or meaning, which could affect the model's overall performance.

CHAPTER 2

Literature Survey

2.1Literature Review

Table 1. Literature Review

Author(s)	Title	Technology Used	Accuracy	Findings
Hemant G. S. and A. K. K.	Spam Filtering: A Literature Survey (2007)	Naive Bayes, Support Vector Machines (SVM), Decision Trees	Not specified	Survey of spam filtering techniques. Naive Bayes is efficient for SMS spam detection.
J. S. Kuo et al.	A Survey on SMS Spam Filtering (2012)	Naive Bayes, k-NN, Decision Trees	97%	Highlights importance of feature extraction and preprocessing for accurate spam detection.
T. T. Lee et al.	A Comparative Study of Machine Learning Algorithms for SMS Spam Filtering (2018)	Support Vector Machine (SVM), Random Forest, k-NN, Naive Bayes	98.5% (SVM), 96.2% (Naive Bayes)	SVM and Naive Bayes performed best in spam detection. SVM had the highest accuracy.
P. R. K. Gupta et al.	Spam SMS Classification Using Naive Bayes and Text Mining Techniques (2019)	Naive Bayes, TF-IDF, Text Mining	96%	Naive Bayes achieved high accuracy after preprocessing steps like tokenization and stopword removal.
A. B. Doshi et al.	SMS Spam Detection using Support Vector Machines and Ensemble Methods (2016)	Support Vector Machines (SVM), Ensemble Learning, k-NN	98.9%	Ensemble methods with SVM showed improved performance in SMS spam detection.
M. K. Garg and S. Sharma	A Hybrid Approach to SMS Spam Classification (2017)	Naive Bayes, Decision Trees, Feature Selection (Chi-Square)	97.5%	Hybrid approach using feature selection and machine learning improved classification accuracy.

2.2Some existing models, techniques, or methodologies

1. **Machine Learning Algorithms**:
   i. **Naive Bayes**: Commonly used due to its simplicity and efficiency in text classification.
   ii. **Support Vector Machine (SVM)**: High accuracy in classifying SMS messages by finding a hyperplane that separates different classes.
   iii. **k-Nearest Neighbors (k-NN)**: A simpler, instance-based classifier, though less commonly used compared to Naive Bayes and SVM.
   iv. **Decision Trees**: Used in some models, but less frequently employed compared to Naive Bayes and SVM.

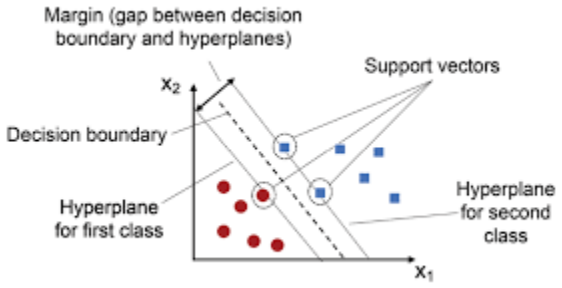

Fig 2. SVM In ML algorithms

2. **Ensemble Methods**:

   i. **Random Forest**: Combines multiple decision trees to improve performance and reduce overfitting.
   ii. **Boosting**: A technique to combine weak learners into a strong classifier, enhancing accuracy.

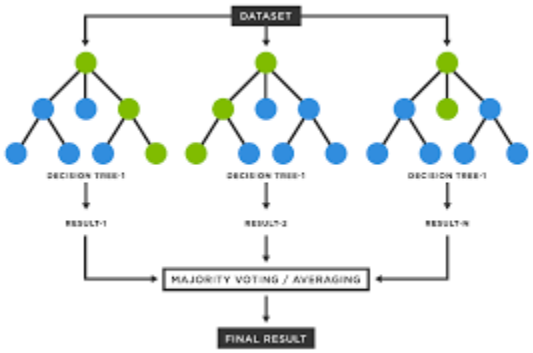

Fig 3.Random Forest in Ensemble Methods

3. **Text Vectorization Methods**:
    i. **TF-IDF (Term Frequency-Inverse Document Frequency)**: Converts text data into numerical format to enable machine learning models to process it effectively.

4. **Deep Learning Approaches**:

    i. **Recurrent Neural Networks (RNNs)**: Captures sequential dependencies in text, useful for detecting complex patterns in SMS.
    ii. **Long Short-Term Memory (LSTM) Networks**: A type of RNN that can remember longer sequences, improving performance in detecting spam based on contextual information.

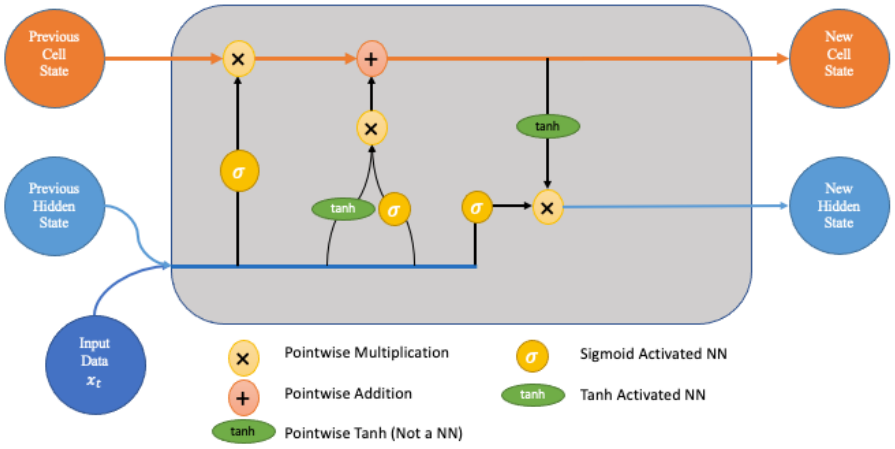

Fig 4. LSTM in Deep Learning

2.3 Limitations in Existing Methodologies or Systems

1. **Limited Feature Engineering**:

    - Many existing solutions rely on basic feature extraction techniques, such as simple word frequencies or TF-IDF, without incorporating advanced text features like character and sentence counts, or sentiment analysis. These features can play an important role in distinguishing between spam and non-spam messages.

2. **Lack of Text Preprocessing Sophistication**:

    - Most SMS spam detection models typically rely on basic preprocessing steps like tokenization, stopword removal, and stemming. However, they may overlook the removal of special characters or fail to incorporate more advanced tokenization methods that can handle misspellings or slangs commonly found in spam messages.

3. **Over-reliance on Traditional Models**:

    - While Naive Bayes and SVM are effective for spam detection, many solutions do not explore more modern or hybrid approaches that combine multiple algorithms or utilize ensemble methods to boost model accuracy.

4. **Inadequate Handling of Imbalanced Data**:

    - Spam datasets are often highly imbalanced, where the number of non-spam messages significantly exceeds that of spam messages. Traditional classifiers may struggle with this imbalance, leading to poor performance in identifying spam messages.

5. **Limited Real-time Prediction Capabilities**:

    - Existing models often lack real-time prediction features, which is important in practical spam detection applications where users need instant feedback.

How this Project Addresses These Gaps:

1. **Incorporating Advanced Text Features**:

    - Your project includes the creation of new columns for **counting characters**, **words**, and **sentences**, providing a richer set of features that can improve the classification process. These additional features can help detect subtle patterns specific to spam messages, which may be missed using word-based features alone.

2. **Sophisticated Text Preprocessing**:

- The preprocessing steps in your project go beyond simple stopword removal and stemming. You include additional processes such as the removal of special characters and the use of the **Porter Stemmer** for text normalization. These enhancements ensure that noisy data and irrelevant elements are filtered out, improving the accuracy of the model.

3. **Use of Multiple Classifiers and Model Comparison**:

   - Your project utilizes multiple models, including **Gaussian Naive Bayes (GNB)**, **Multinomial Naive Bayes (MNB)**, and **Bernoulli Naive Bayes (BNB)**. By training and comparing these models, your approach allows you to select the best-performing classifier, ensuring that the final model is optimized for accuracy.

4. **Balancing Data with Vectorization**:

   - The use of **TF-IDF Vectorization** and the advanced text preprocessing improves the handling of unbalanced data by representing each message more effectively. This helps the model detect both spam and non-spam messages with higher precision.

5. **Real-Time Prediction with Streamlit Integration**:

   - The integration with **Streamlit** allows for the deployment of the spam detection model as an interactive, real-time web application. Users can input SMS messages and get immediate feedback on whether they are spam or not. This is an important feature for practical usage in real-world applications.

CHAPTER 3

Proposed Methodology

3.1 System Design

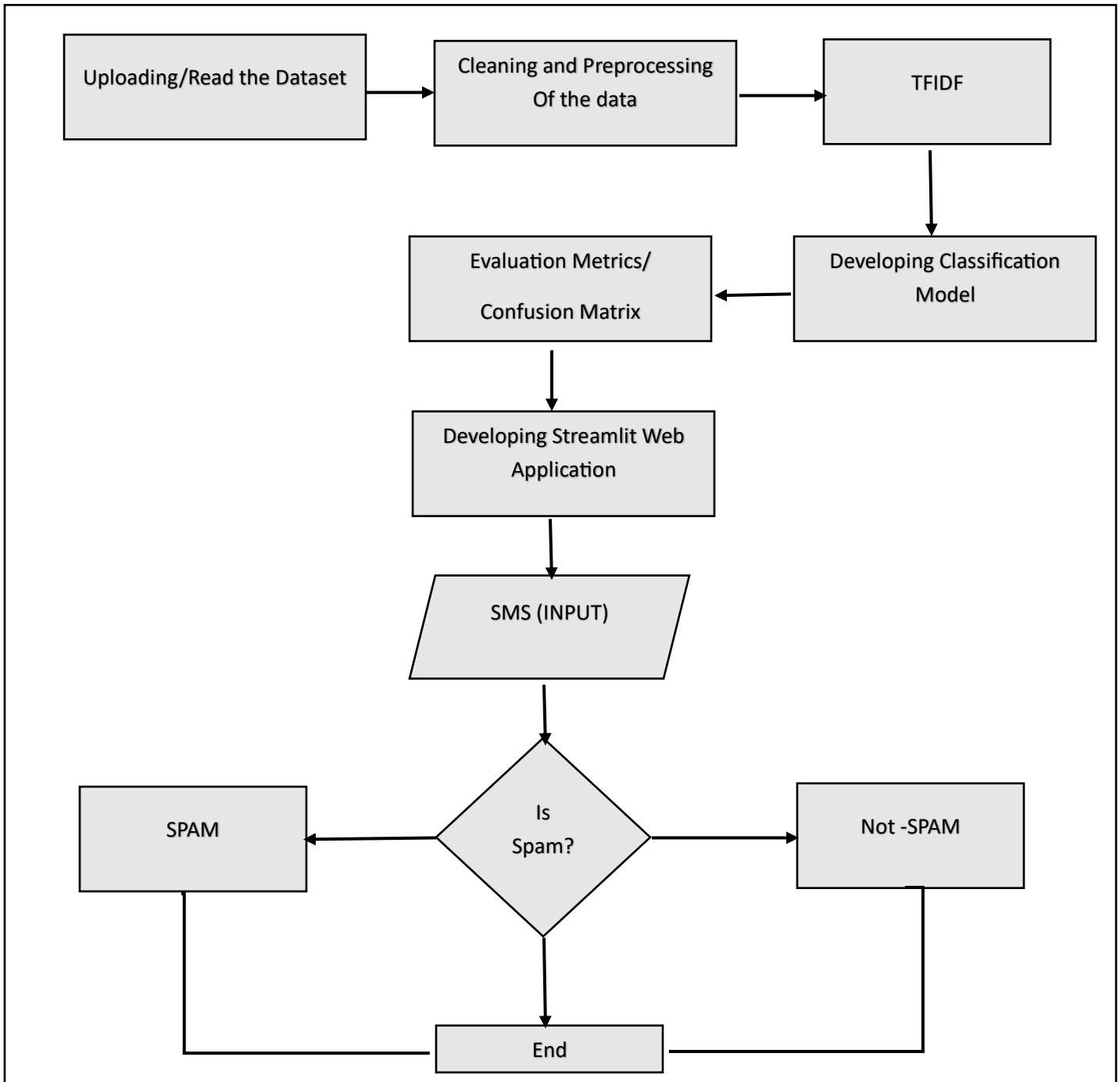

Fig 5.System Architecture

Steps for SMS Classification using Multinomial Naive Bayes:

1. **Data Collection**:

Obtain a labeled dataset of SMS messages where each message is labeled as **spam** or **not spam**. This can be from publicly available datasets like the **SMS Spam Collection** dataset.

2. Preprocessing:

- **Text Cleaning:** Clean the SMS text by removing unwanted characters (e.g., punctuation, special symbols), converting all text to lowercase, and handling any non-alphanumeric characters.
- **Tokenization:** Split the text into smaller units called tokens (typically words) using a tokenizer.
- **Removing Stopwords:** Remove common, non-informative words (e.g., "the", "is", "in") that don't contribute significantly to classification.

3. Feature Extraction:

- **Bag of Words (BoW):** Convert the SMS messages into numerical vectors by counting the occurrences of each word in the message. This creates a feature set that represents each message as a vector of word counts.
- **TF-IDF:** Alternatively, use **TF-IDF (Term Frequency-Inverse Document Frequency)** to weigh words based on their frequency in the document relative to their frequency across the entire dataset, helping to identify more relevant words.

4. Model Training (Multinomial Naive Bayes):

Naive Bayes Classification: Train a **Multinomial Naive Bayes** model using the preprocessed text data and its corresponding labels (spam or not spam). The model assumes that words are independent given the class, and it uses the multinomial distribution to model word occurrences in different classes (spam vs. not spam).

5. Model Evaluation:

Evaluate the performance of the trained model using metrics such as **accuracy**, **precision**, **recall**, and **F1-score** to assess how well the model distinguishes between spam and non-spam messages.

Steps for Generating a Streamlit Web Application:

1. Streamlit Setup:

Install **Streamlit** and other necessary libraries (e.g., pandas, scikit-learn, nltk) to create the web app.

2. Load the Trained Model and Vectorizer:

In the web app, load the previously trained **Multinomial Naive Bayes** model and the **vectorizer** (used for transforming input text into numerical features) from the saved files.

3. Create the Web Interface:

Use **Streamlit** widgets to create an interactive user interface where users can input an SMS message.Provide a text input field where users can enter their SMS.

4. **Prediction Logic**:

When a user inputs an SMS, the app should preprocess the input text (e.g., cleaning, tokenizing, removing stopwords) and then convert it into a numerical vector using the vectorizer.Use the trained **Multinomial Naive Bayes** model to predict whether the message is **spam** or **not spam**.

5. Display Results:

After the prediction, display the result on the web page (e.g., "SPAM" or "NOT SPAM")

3.2 Requirement Specification

3.2.1 Hardware Requirements:

To implement the SMS spam detection solution, the following hardware resources are required:

- **Processor**: A multi-core processor (e.g., Intel i5 or above, or equivalent AMD processor) for efficient model training and data processing.
- **RAM**: At least 8 GB of RAM is recommended to handle data processing, model training, and running the application without performance issues.
- **Storage**: Minimum of 10 GB of free disk space to store the dataset, trained models, and other necessary libraries.
- **GPU (Optional)**: If deep learning models are used (e.g., using LSTMs), a GPU like an NVIDIA GTX 1060 or higher can speed up the training process, although for this project using traditional models, it is not a strict requirement.
- **Internet Connection**: A stable internet connection for downloading dependencies, libraries (such as NLTK data), and hosting the web application.

3.2.1 Software Requirements:

The following software and tools are necessary for implementing the SMS spam detection solution:

- **Python (3.x)**: The core programming language used for implementing the SMS spam detection model, preprocessing steps, and real-time prediction system.
- **Python Libraries**:
  - **NLTK (Natural Language Toolkit)**: Used for text processing tasks such as tokenization, stemming, stopword removal, and other natural language processing (NLP) tasks.
  - **Pandas**: A library used for data manipulation and preprocessing, such as handling datasets and adding features like word counts and sentence counts.
  - **NumPy**: For numerical operations required during data processing and matrix transformations.
  - **Scikit-learn**: A machine learning library for implementing various models such as Naive Bayes, SVM, and vectorization techniques like TF-IDF.
  - **Pickle**: Used for serializing and saving the trained models and vectorizers so that they can be reused for prediction without retraining.
  - **Streamlit**: A tool for deploying the SMS spam detection model as an interactive web application where users can input SMS text and receive predictions.
  - **Matplotlib/Seaborn**: For data visualization tasks such as plotting word clouds to visualize the frequency of words in spam messages.

- **Integrated Development Environment (IDE)**:
  - **VS Code**, **PyCharm**, or **Jupyter Notebook**: Recommended for writing, testing, and debugging the Python code for the project.
- **Web Hosting/Deployment Tools** (for deployment):
  - **Heroku**, **Streamlit Cloud**, or **AWS**: These cloud platforms can be used to deploy the Streamlit web application so that it is accessible online for users to input SMS messages and get predictions in real-time.
- **Web Browser**: A modern web browser (e.g., Google Chrome, Mozilla Firefox) to test and interact with the deployed web application.

CHAPTER 4

Implementation and Result

4.1 Snap Shots of Result:

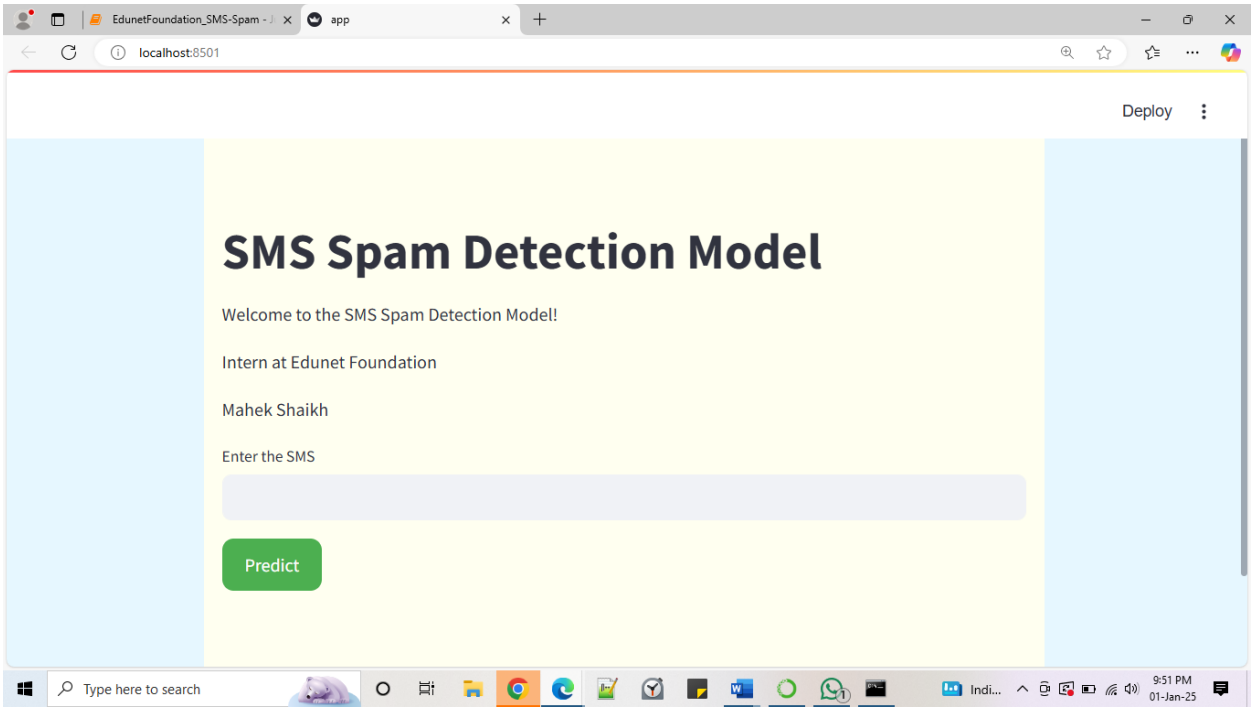

Fig 6.Implementation (I) Interface

- Streamlit provides a simple interface where users can input their SMS text into a text input box, making it easy to interact with the spam detection model.
- The app includes a text input field for users to type or paste the SMS message they want to analyze.
- A "Predict" button is available beneath the input field, which users can click to trigger the spam classification process.
- Once the "Predict" button is clicked, the app processes the input and displays whether the message is "Spam" or "Not Spam" based on the model's prediction.

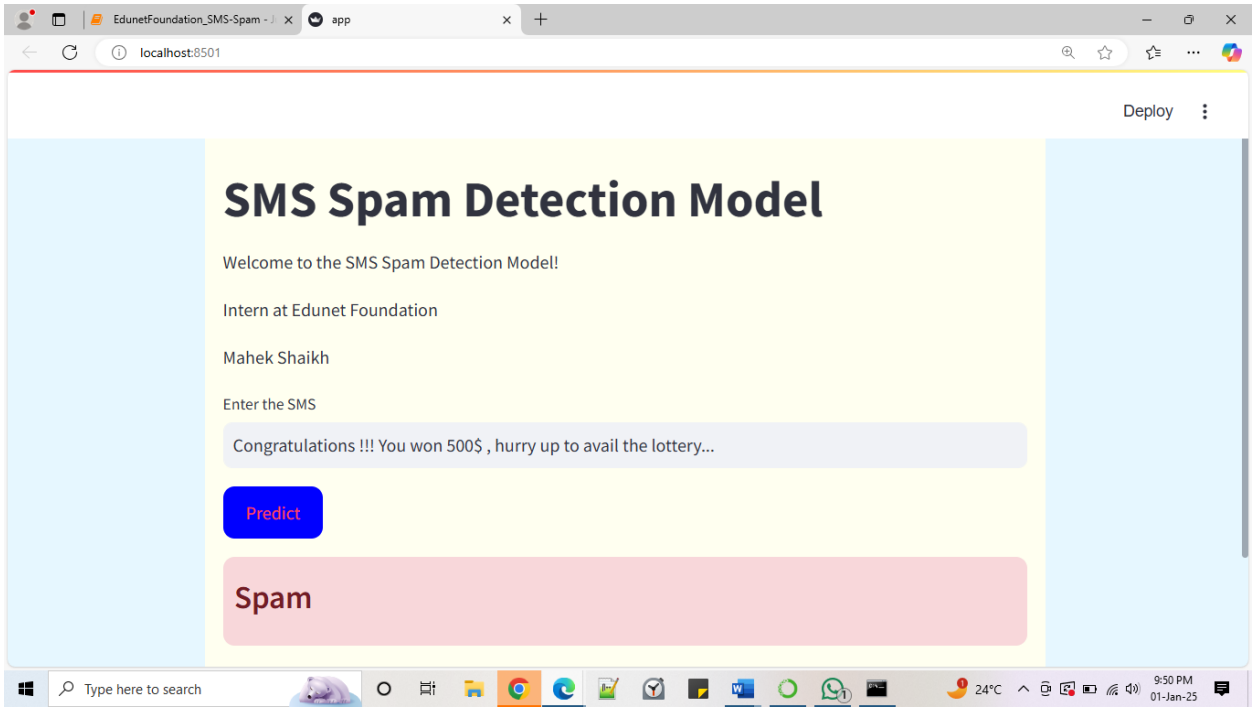

Fig 7.Implementation (II)

Explanation:

- The second snapshot demonstrates the system classifying an SMS as SPAM. The SMS input contains typical spam-like content, such as promotional offers, clickbait phrases, or deceptive language.
- The classifier processes the input text through its Multinomial model, using previously trained data to identify spam-related pattern.
- The result, "Spam", is displayed along with optional confidence scores or a message indicating why it was classified as spam. This shows the system's ability to detect spam SMS effectively.
- The SMS is classified as spam because it contains suspicious keywords like "hurrey" and "lottery," which are often used in scams. Additionally, the excessive punctuation ("!!") is a common tactic in promotional or fraudulent messages.

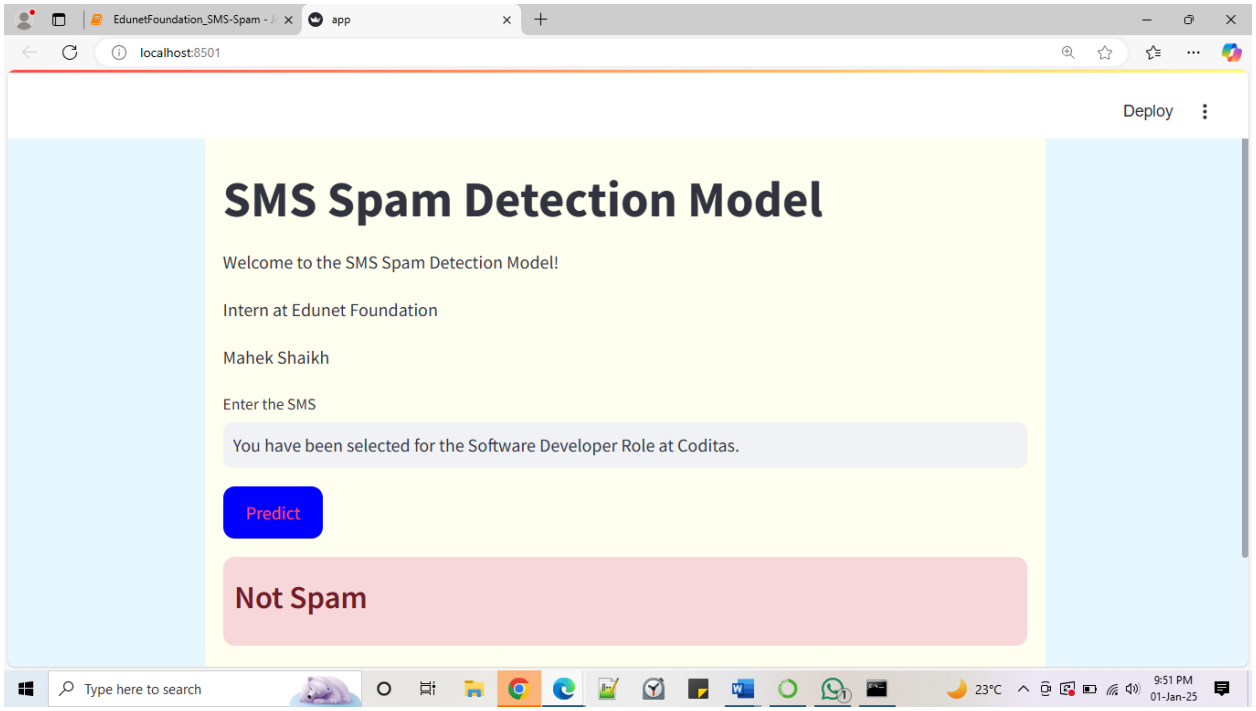

Fig 8.Implementation (III)

Explanation:

- The third snapshot demonstrates SMS is classified as "Not Spam" because it does not contain any suspicious keywords or patterns typically found in scam or promotional messages. The message appears legitimate and free from common spam indicators.
- The classifier processes the input text through its Multinomial model, using previously trained data to identify spam-related pattern.

4.2 GitHub Link for Code:

https://github.com/MahekSuhail23/EdunetFoundation-TechSaksham-SMS_Spam_Classifier

CHAPTER 5

Discussion and Conclusion

5.1 Future Work:

i. **Incorporating Deep Learning Models**: Implement deep learning models like Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks to capture the sequential nature of SMS messages and improve detection accuracy.

ii. **Handling Imbalanced Data**: Apply oversampling (e.g., SMOTE) or undersampling techniques, or adjust class weights to address the issue of imbalanced data, ensuring better model performance on underrepresented spam messages.

iii. **Feature Engineering Enhancements**: Enhance feature engineering by including sentiment analysis, n-grams in TF-IDF, and semantic features through word embeddings (e.g., Word2Vec, GloVe) to capture richer contextual information.

iv. **Improved Text Preprocessing**: Incorporate spelling correction algorithms and slang normalization to handle informal language and noise in SMS text, improving the preprocessing pipeline.

v. **Cross-lingual Spam Detection**: Expand the model to handle multiple languages by using language detection and multilingual models (e.g., Multilingual BERT) to make the system more adaptable globally.

vi. **Real-time Feedback and Continuous Learning**: Implement online learning or continuous retraining to allow the model to learn from new data and improve over time, adapting to evolving spam tactics.

5.2 Conclusion:

The overall impact and contribution of the SMS spam detection project lie in its ability to effectively identify and classify spam messages from legitimate ones using machine learning and natural language processing (NLP) techniques. By leveraging traditional machine learning algorithms like Naive Bayes, along with text preprocessing techniques such as tokenization, stopword removal, stemming, and TF-IDF vectorization, the project provides an efficient and automated solution to filter out spam messages. This helps in enhancing user experience by reducing unwanted and potentially harmful content in SMS communications.

Additionally, the project contributes to the field by incorporating a web-based interface using Streamlit, which allows users to interact with the spam detection model in real-time. The use of a pickle-based deployment ensures that the model can be easily reused without the need for retraining, promoting scalability.

The implementation of such a solution can have significant real-world applications, particularly in mobile security, telecom industries, and communication platforms, where spam detection is a critical need. The project's impact extends to providing a reliable, user-friendly tool that can be further improved for more advanced applications, including multilingual spam detection, continuous learning, and integration with other platforms for comprehensive spam filtering across different media.

REFERENCES

[1]. C. C. Aggarwal and C. Zhai, "A Survey of Text Classification Algorithms," IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 12, pp. 2364-2387, Dec. 2014.

[2]. S. J. Delany, M. Buckley, and D. Greene, "SMS Spam Filtering: Methods and Data," IEEE Expert Systems with Applications, vol. 39, no. 10, pp. 9899-9908, Aug. 2012.

[3]. I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, and C. D. Spyropoulos, "An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages," IEEE Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 160-167, 2000.

[4]. A. K. Uysal and S. Gunal, "The Impact of Preprocessing on Text Classification," IEEE Information Processing & Management, vol. 50, no. 1, pp. 104-112, Jan. 2014.

[5]. M. Karami and G. Zhou, "Improving Short Text Classification Using Unlabeled Data," IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 6, pp. 1109-1121, June 2018.