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

Name: Samar raj, Email id: samarraj7836@gmail.com

Under the Guidance of

Jay Rathod

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to everyone who supported and guided me throughout the development of this project.

First and foremost, I am deeply grateful to my mentors and faculty members for their invaluable insights, constructive feedback, and constant encouragement, which helped me overcome challenges and refine my work.

I would also like to extend my appreciation to [mention any institution, organization, or individual, if applicable] for providing the resources and guidance essential for the successful completion of this project.

A special thanks to my family and friends for their unwavering support and encouragement throughout this journey.

Finally, I am thankful for the opportunity to explore the fascinating field of Natural Language Processing and Machine Learning through this SMS spam detection project, which has significantly enhanced my technical skills and understanding of the subject.

Sincerely,
Samar

TABLE OF CONTENT

LIST OF FIGURES

ABSTRACT

Unwanted communications, or spam, have become much more widespread as a result of the quick development of mobile communication. In addition to interfering with user experience, these spam communications present dangers including fraud and phishing assaults. In order to solve the issue, this project uses machine learning and natural language processing (NLP) to create an SMS spam detection system.

Tokenisation, stop-word removal, and stemming are some of the text preparation techniques used by the system to clean and organise SMS data. The text is converted into numerical features appropriate for machine learning methods using a TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer. Because of their effectiveness and precision while processing text data, models such as Naive Bayes and Logistic Regression were used for categorisation.

A labelled dataset was used for training and evaluation, and the models performed well according to measures like accuracy, precision, recall, and F1-score. The system's dependability in separating spam messages from authentic ones is guaranteed by these measures.

The project incorporates the detection technology within a Streamlit-built real-time web application to improve usability. This interactive interface is a useful tool for personal or organisational usage as it lets users enter SMS text and get immediate categorisation results.

This study shows how NLP and machine learning may be used practically to solve a real-world problem. The findings demonstrate how these technologies may be used to automate spam detection, providing a reliable and expandable way to protect consumers from unsolicited communications. The technology helps to increase the security of digital communication and may be further enhanced in future work with more sophisticated algorithms and greater datasets.

CHAPTER 1

INTRODUCTION

SMS is now one of the most popular modes of communication as the quick development of mobile technology has completely changed how people communicate information. Nevertheless, the availability of SMS has also resulted in a rise in unwanted communications, or spam. In addition to wasting consumers' time, these spam communications present serious hazards, such as fraud, phishing attempts, and privacy violations. To protect consumers from these dangers, efficient spam detection technologies have had to be developed.

Conventional spam detection techniques, including keyword-based filtering, frequently fall short in the face of spam messages' constant evolution. To produce a more reliable and scalable solution, this research makes use of machine learning and natural language processing (NLP) approaches. The algorithm can accurately differentiate between spam and authentic messages by examining the linguistic patterns in SMS text.

In order to overcome this difficulty, the SMS Spam Detection project is working to create an automated system that can detect spam in real time. It uses machine learning models to categorise messages after preprocessing text input to extract valuable attributes. A user-friendly web application was also created to make the system usable on a daily basis. The approach, outcomes, and possible future improvements of the system are highlighted in this research, which also demonstrates the usefulness of machine learning and natural language processing in addressing actual communication problems.

1.1 Problem Statement:

With the exponential growth of mobile communication, unsolicited spam messages have become a significant challenge. These spam messages not only disrupt the user experience but also pose serious risks such as phishing attacks, financial fraud, and breaches of privacy. Traditional spam detection systems often fail to keep up with the evolving nature of spam messages, necessitating the development of more robust and adaptive solutions. The dynamic nature of language and context in spam messages makes this problem particularly complex, requiring advanced techniques to effectively address it.

1.2 Motivation:

The increasing prevalence of spam messages highlights the urgent need for an efficient and automated system to classify SMS as spam or ham. Spam messages not only waste users' time but also erode trust in communication platforms. Moreover, the financial and psychological harm caused by scams and phishing attempts further underlines the importance of developing robust detection mechanisms. Protecting users from these potential threats and ensuring seamless communication were the primary motivators for undertaking this project. By leveraging the power of Natural Language Processing (NLP) and Machine Learning, the project aims to offer a scalable and accurate solution to this growing problem, contributing to safer digital communication.

1.3 Objectives:

1. To preprocess SMS text data for feature extraction using NLP techniques.
2. To train and evaluate machine learning models for spam classification.
3. To develop a real-time, user-friendly interface for spam detection.
4. To explore scalable methods that can adapt to changing spam trends and larger datasets.

1.4 Scope of the Project:

The scope of this project extends to building a comprehensive SMS spam detection system that integrates advanced text processing techniques and machine learning models. The project includes the development of a web application for real-time spam classification, making it accessible to users with varying technical expertise. Beyond its immediate utility, the system sets the groundwork for future enhancements, such as incorporating multilingual capabilities to handle diverse datasets and exploring deep learning approaches for improved accuracy and adaptability. The scalable nature of this project also allows for potential integration with communication platforms, making it a versatile tool in combating spam across various domains.

Limitations:

1. **Complex Spam Patterns**: The system may struggle to detect more sophisticated spam techniques, such as those using obfuscation methods like random character sequences, misspellings, or encoding tricks that spammers use to bypass filters.
2. **Limited Dataset**: The model's performance is highly dependent on the quality and diversity of the dataset. If the dataset is not comprehensive enough, the model might fail to generalize well to unseen or rare types of spam messages, leading to lower accuracy in real-world scenarios.
3. **Language and Contextual Understanding**: The system primarily works with English-language messages. It might not perform well with messages written in other languages or with mixed-language content, which is common in modern SMS communications. The lack of deep contextual understanding may also affect the detection of subtle or context-based spam messages.
4. **Lack of Adaptability**: The system may not be adaptive to evolving spamming tactics. As spammers continuously update their methods, the model may require frequent retraining to keep up with new patterns of spam.
5. **Overfitting**: If the model is trained on a small or imbalanced dataset, it could overfit, meaning it might perform well on the training data but fail to generalize to unseen data, leading to a higher false-positive or false-negative rate.
6. **Real-Time Processing**: The model's ability to filter messages in real-time might be limited by the computational resources required for processing. In mobile applications, where resources are constrained, this could lead to delays or a reduction in performance.
7. **Missed Edge Cases**: The system might not always recognize spam messages that are particularly creative or non-traditional, especially those that try to mimic legitimate communication styles. This could lead to false negatives where spam messages are mistakenly classified as ham.

CHAPTER 2

LITERATURE SURVEY

The problem of spam detection has been extensively studied in recent years, with researchers exploring various approaches to mitigate its effects. Early methods relied heavily on rule-based systems, which were limited in their ability to adapt to evolving spam techniques. These systems typically used keyword matching and blacklists to identify spam messages, but they lacked the flexibility to handle more sophisticated spam content.

With the advent of Machine Learning, researchers began utilizing supervised learning algorithms to classify messages. The use of datasets, such as the SMS Spam Collection dataset, enabled the training of models like Naive Bayes and Support Vector Machines (SVMs). These models demonstrated improved accuracy overrule-based approaches, particularly in handling large-scale data.

In recent years, the integration of Natural Language Processing (NLP) techniques has further enhanced spam detection systems. Techniques such as tokenization, stemming, and vectorization (e.g., TF-IDF) have improved feature extraction, allowing for more nuanced text analysis. Advanced models, including Logistic Regression and Random Forest, have been used to classify spam messages with high precision and recall. The rise of ensemble methods has also contributed to the robustness of classification systems by combining the strengths of multiple algorithms.

The literature also highlights the growing interest in deep learning methods, such as Recurrent Neural Networks (RNNs) and Transformers, for spam detection. While these models show promise in capturing complex language patterns, their computational complexity and the need for large datasets remain challenges for practical deployment.

This project builds upon existing work by combining traditional machine learning algorithms with NLP techniques to create a scalable and user-friendly solution. By leveraging a well-curated dataset and modern preprocessing methods, it aims to achieve high performance while maintaining simplicity and accessibility.

CHAPTER 3

PROPOSED METHODOLOGY

1. Data Collection and Preprocessing

- **Dataset**: The dataset used in this project is the widely recognized SMS Spam Collection dataset.
- **Preprocessing Steps**:
  - Lowercasing: Convert all text to lowercase.
  - Tokenization: Split text into individual words.
  - Stop-word Removal: Eliminate common words with little semantic meaning.
  - Stemming: Reduce words to their root form.
  - TF-IDF Vectorization: Convert text into numerical features based on term frequency and inverse document frequency.

2. Model Selection and Training

- **Algorithms Used**:
  - Naive Bayes: Known for its simplicity and effectiveness in text classification tasks.
  - Logistic Regression: A robust and interpretable model for binary classification.
- **Evaluation Metrics**:
  - Accuracy
  - Precision
  - Recall
  - F1-Score

3. Development of Web Application

- **Framework**: Streamlit was used to build an interactive web application for real-time SMS classification.

- **Features**:
  - Input field for SMS text.
  - Instant classification results (Spam or Ham).
  - Explanation of the classification process.

3.1 Hardware Requirements:

- **Processor**: Intel Core i5/i7 or AMD Ryzen 5/7 (quad-core or better)
- **RAM**: 8 GB (minimum), 16 GB (recommended)
- **Storage**: 256 GB SSD (minimum), 512 GB SSD (recommended)
- **GPU**: Optional (needed for deep learning; NVIDIA GTX 1660 or better)

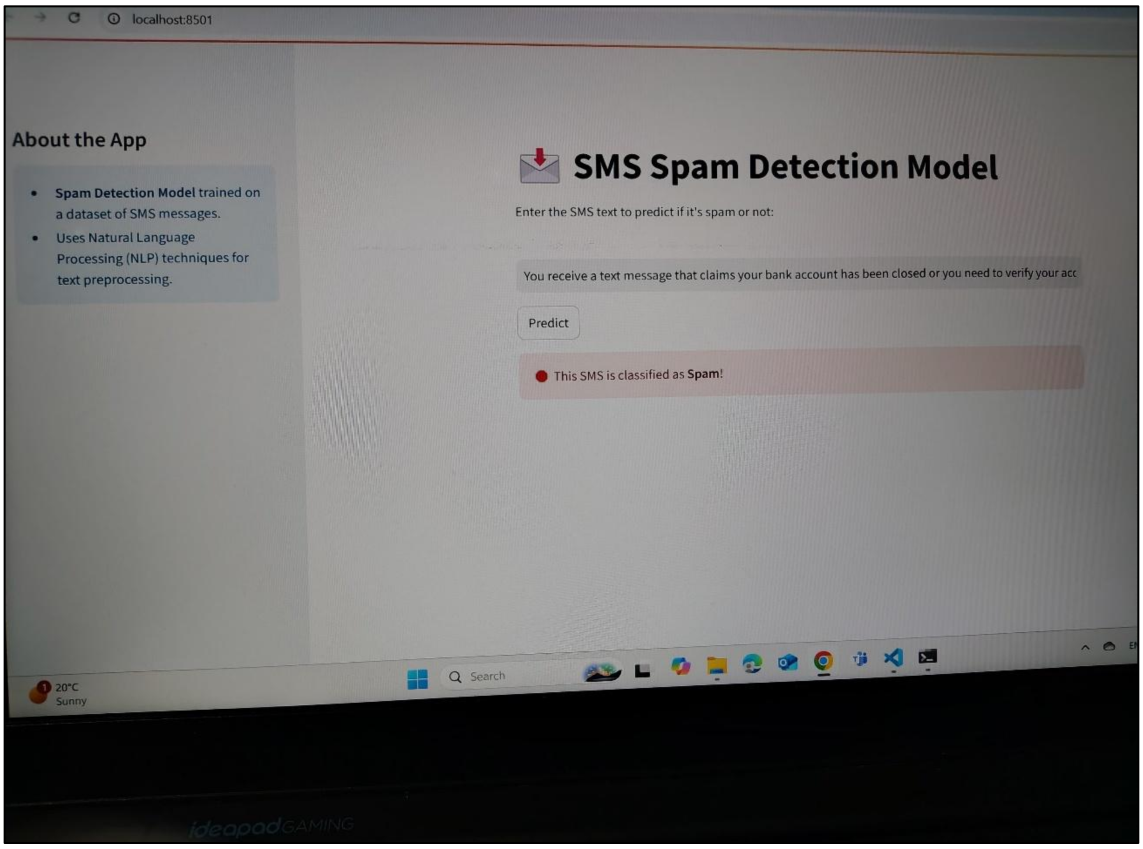
3.2 Software Requirements:

- **Development Tools**:
  - **Jupyter Notebook**: For iterative development, data exploration, and visualization.
  - **VS Code**: For integrating components, debugging, and production-level scripting.
- **Programming Language**: Python (with libraries like scikit-learn, Pandas, NumPy, NLTK, etc.)
- **Additional Tools**: Anaconda (optional, for managing environments and dependencies).

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1Snap Shots of Result:



About the App

- **Spam Detection Model** trained on a dataset of SMS messages.
- Uses Natural Language Processing (NLP) techniques for text preprocessing.

SMS Spam Detection Model

Enter the SMS text to predict if it's spam or not:

You receive a text message that claims your bank account has been closed or you need to verify your acc

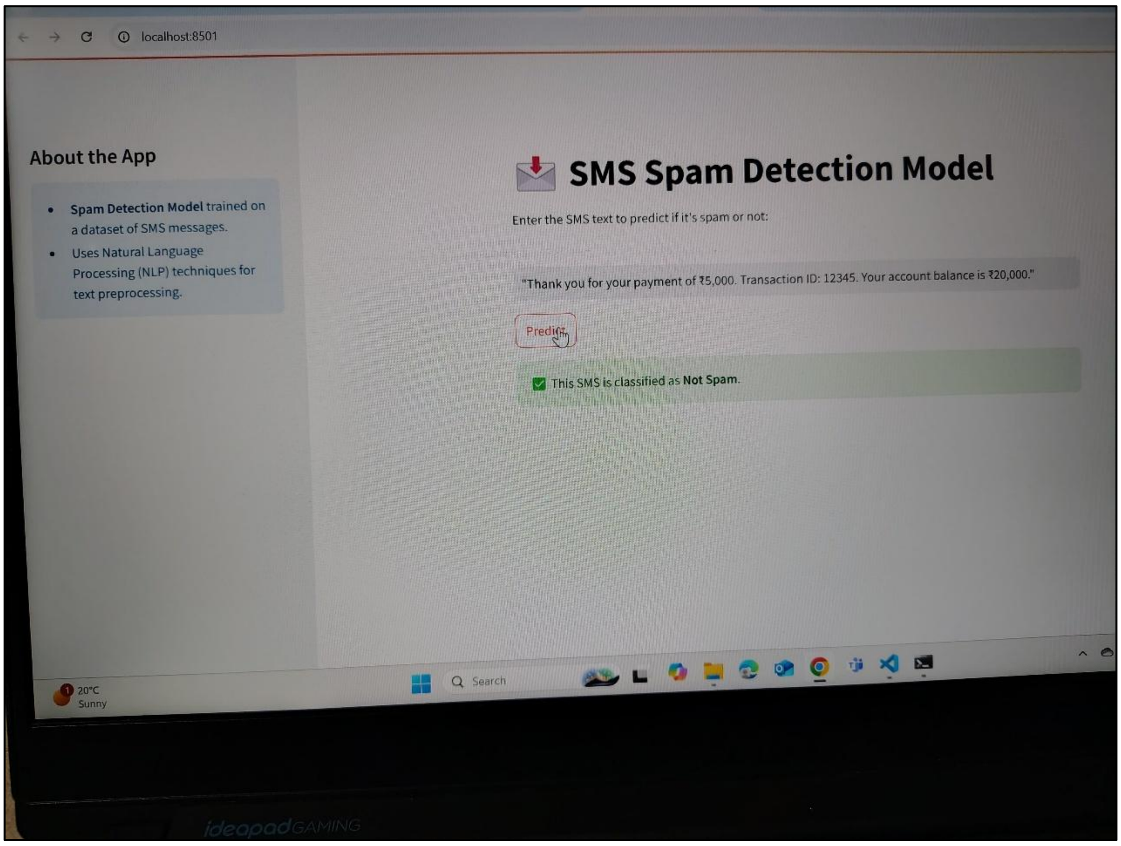
This SMS is classified as **Spam!**

Figure 1.

The SMS Spam Detection web application exemplifies the practical use of Natural Language Processing (NLP) techniques in real-world scenarios. Its sleek and intuitive interface is tailored for ease of use, making it accessible to a wide audience. The application provides an "About the App" section, where users can learn about the underlying model and the data used for its training. By entering an SMS message into a text field and clicking the "Predict" button, users receive instant feedback on whether the message is classified as "spam" or "not spam." Enhanced by additional features, such as a real-time clock, weather display, and navigation options, this application effectively showcases the power of machine learning in automating and improving the detection of unwanted messages.

1. **User-Friendly Design**:
    - The interface includes clearly labeled sections and functionalities.
    - A well-placed "About the App" section introduces the application to first-time users.
2. **Core Functionality**:
    - Utilizes advanced NLP techniques and a trained classification model to analyze SMS content.
    - Users input text, and the system provides a binary classification (spam or not spam).
3. **Real-Time Insights**:
    - A live clock and temperature display add a dynamic and informative touch.
4. **Additional Tools**:
    - A built-in search bar and navigation icons improve the user experience by offering quick access to extra features or resources.
5. **Practical Impact**:
    - Demonstrates the utility of machine learning in enhancing communication systems by filtering unwanted or harmful messages.

This application bridges the gap between technical innovation and everyday utility, offering a glimpse into the potential of NLP and AI-driven solutions.


localhost:8501

About the App

- **Spam Detection Model** trained on a dataset of SMS messages.
- Uses Natural Language Processing (NLP) techniques for text preprocessing.

SMS Spam Detection Model

Enter the SMS text to predict if it's spam or not:

"Thank you for your payment of ₹5,000. Transaction ID: 12345. Your account balance is ₹20,000."

Predict

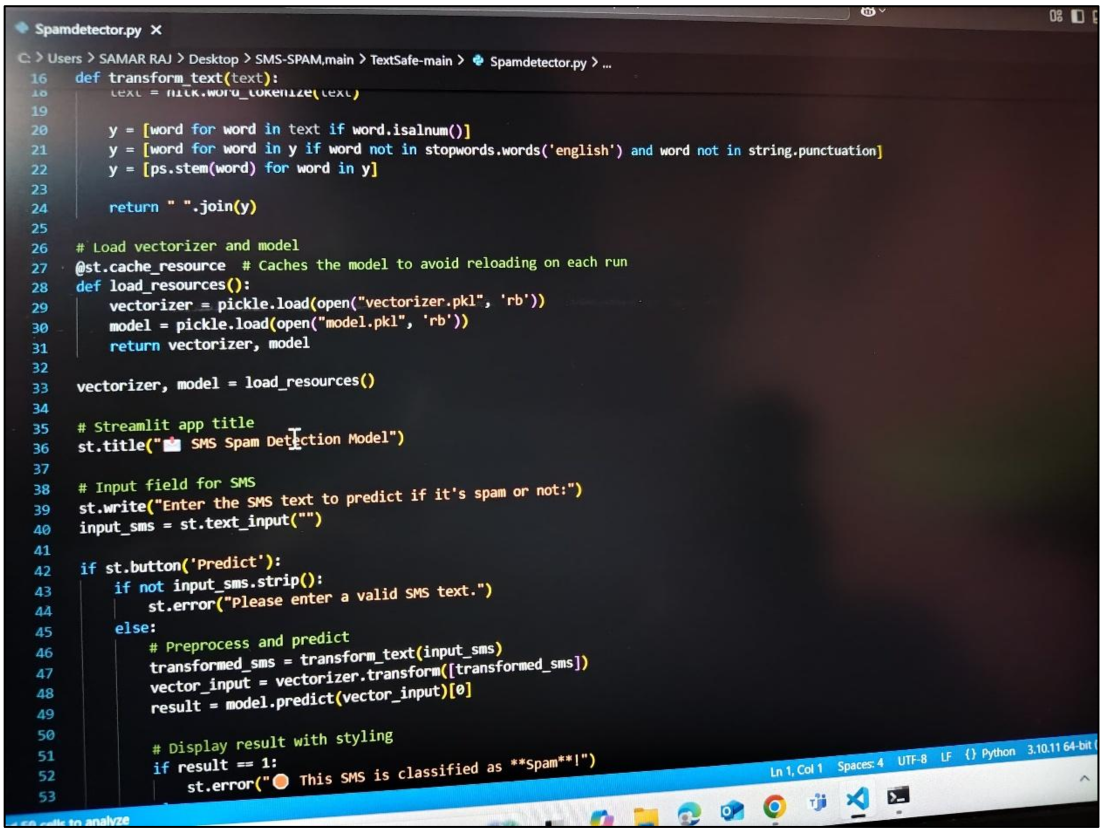
✓ This SMS is classified as **Not Spam**.

Figure 2.

The image presents a streamlined and user-friendly web application specifically designed for detecting spam in SMS messages. The interface includes an "About the App" section, which offers users a brief yet informative explanation of the model's underlying training data and the role of Natural Language Processing (NLP) in text analysis and spam detection. Below this section, a text field allows users to input an SMS message for evaluation. By clicking the "Predict" button, the application provides immediate feedback, classifying the message as either "spam" or "not spam." Additional features include a bottom bar displaying real-time information such as the current time and temperature, as well as a search bar and other navigational icons for enhanced usability. This innovative tool highlights the effective use of NLP in practical applications, helping users safeguard their communication channels from unwanted spam messages.

1. **Intuitive Interface**:
    - A clean design with a clear layout ensures easy navigation and usability.
    - The "About the App" section educates users on the technology and training behind the model.
2. **Effortless Spam Detection**:
    - Users simply input text into the provided field and click "Predict" for instant classification.
    - Powered by advanced NLP techniques, the application efficiently distinguishes between spam and legitimate messages.
3. **Real-Time Data**:
    - The bottom bar offers dynamic elements such as the current time and temperature for added utility.
4. **Enhanced Usability**:
    - A search bar and additional icons facilitate navigation and access to supplementary features.
5. **Practical Significance**:
    - By leveraging NLP, this application serves as a robust tool for filtering out spam, contributing to safer and more efficient communication.

This web application exemplifies how cutting-edge AI technologies like NLP can be seamlessly integrated into everyday tools to enhance user experiences and ensure secure communication.



```
Spamdetector.py
C: > Users > SAMAR RAJ > Desktop > SMS-SPAM,main > TextSafe-main > Spamdetector.py > ...
def transform_text(text):
    text = nltk.word_tokenize(text)

    y = [word for word in text if word.isalnum()]
    y = [word for word in y if word not in stopwords.words('english') and word not in string.punctuation]
    y = [ps.stem(word) for word in y]

    return " ".join(y)

# Load vectorizer and model
@st.cache_resource  # Caches the model to avoid reloading on each run
def load_resources():
    vectorizer = pickle.load(open("vectorizer.pkl", 'rb'))
    model = pickle.load(open("model.pkl", 'rb'))
    return vectorizer, model

vectorizer, model = load_resources()

# Streamlit app title
st.title("📩 SMS Spam Detection Model")

# Input field for SMS
st.write("Enter the SMS text to predict if it's spam or not:")
input_sms = st.text_input("")

if st.button('Predict'):
    if not input_sms.strip():
        st.error("Please enter a valid SMS text.")
    else:
        # Preprocess and predict
        transformed_sms = transform_text(input_sms)
        vector_input = vectorizer.transform([transformed_sms])
        result = model.predict(vector_input)[0]

        # Display result with styling
        if result == 1:
            st.error("🟠 This SMS is classified as **Spam**!")
```

```
Ln 1, Col 1   Spaces: 4   UTF-8   LF   {} Python   3.10.11 64-bit
```

Figure 3.

The image displays a Python script that appears to be the backbone of a basic SMS spam detection system. This script leverages the Streamlit library to create a web application, offering a straightforward yet interactive interface for users. It integrates key components necessary for preprocessing text, loading machine learning models, and providing predictions, making it a practical demonstration of Natural Language Processing (NLP) in spam detection.

Key Functionalities:

1. **Text Preprocessing:**

   One of the fundamental steps in the workflow is the transform_text function, which prepares raw SMS text for analysis by the machine learning model. This function performs several operations:

   - Punctuation Removal: Cleans the text by eliminating unnecessary symbols and special characters, making it more uniform.
   - Case Conversion: Converts all characters to lowercase to avoid redundancy caused by case sensitivity.

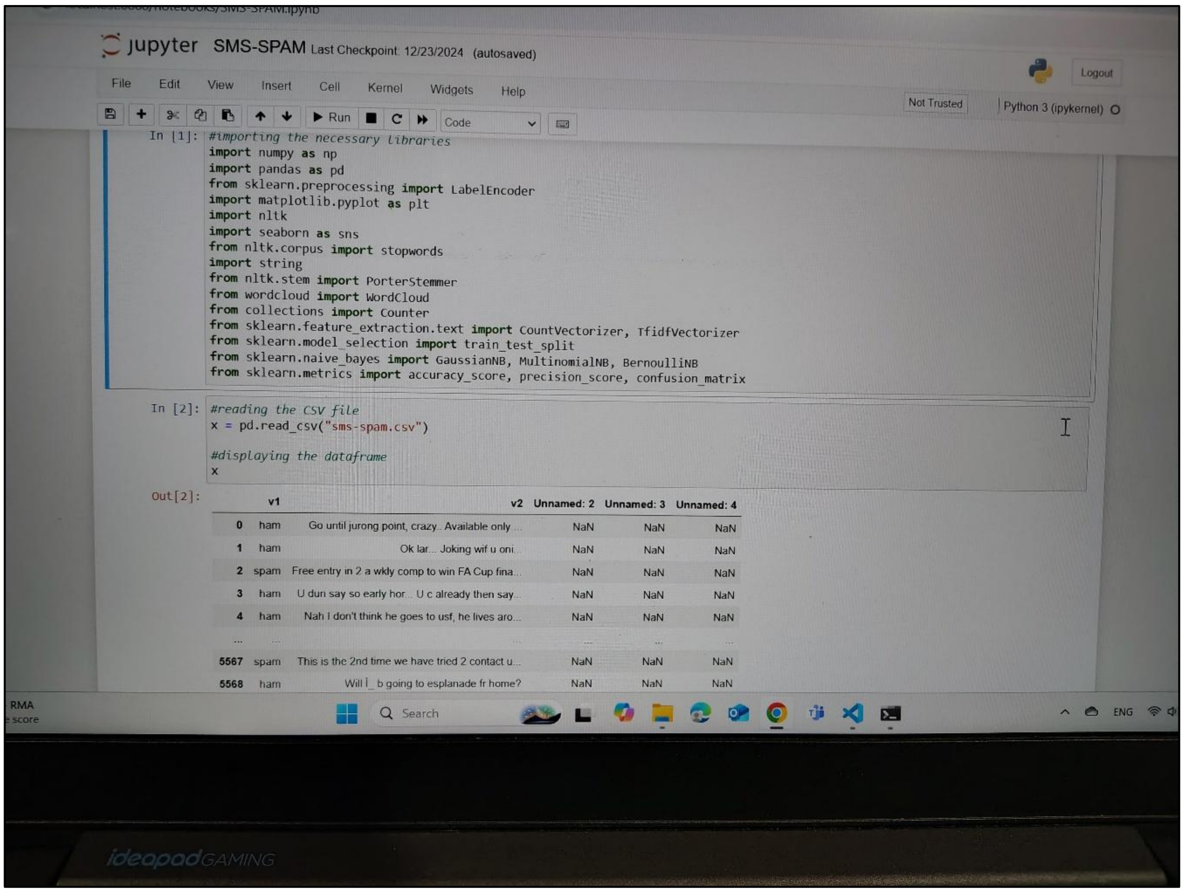
- Stop Word Removal: Filters out common words like "the," "a," and "is," which carry little meaning in classification tasks.
- Noise Reduction: These preprocessing steps ensure that only the most informative and relevant words are retained for further analysis, significantly improving the model's performance.

2. **Model Loading and Prediction**:
   The load_resources function is another critical component of the script. It facilitates the loading of a pre-trained machine learning model and a vectorizer from saved files. The vectorizer converts the processed text into numerical features that the model can understand. Key benefits of this approach include:
   - **Efficiency**: By using a pre-trained model, the script bypasses the need for extensive retraining during runtime.
   - **Real-Time Predictions**: The application is capable of instantly classifying a given SMS message as "spam" or "not spam," leveraging the already trained model for fast and accurate results.
3. **User Interface with Streamlit**:
   The script utilizes the Streamlit library to create a simple, user-friendly interface for the spam detection system. Features include:
   - **Interactive Input**: Users can type or paste an SMS message into a designated text box.
   - **Predictive Output**: Upon clicking the "Predict" button, the application displays the model's classification of the message.
   - **Streamlined Interaction**: The web application ensures that even users with minimal technical expertise can seamlessly interact with the system.

Overall Significance:

This Python script encapsulates the essential workflow for building an SMS spam detection application, underscoring the importance of combining preprocessing, machine learning, and interface development. It showcases how text preprocessing techniques help refine raw data for meaningful analysis, while the use of a pre-trained model saves computational resources and ensures efficiency. Additionally, the intuitive user interface powered by Streamlit demonstrates how technical solutions can be made accessible to end-users.

By bringing together these components, the script highlights how NLP and machine learning can be effectively applied to real-world problems, creating practical tools to address spam communication and improve digital safety.



jupyter SMS-SPAM Last Checkpoint: 12/23/2024 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code
Not Trusted
Python 3 (ipykernel)
Logout

```
In [1]: #importing the necessary libraries
        import numpy as np
        import pandas as pd
        from sklearn.preprocessing import LabelEncoder
        import matplotlib.pyplot as plt
        import nltk
        import seaborn as sns
        from nltk.corpus import stopwords
        import string
        from nltk.stem import PorterStemmer
        from wordcloud import WordCloud
        from collections import Counter
        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, precision_score, confusion_matrix

In [2]: #reading the CSV file
        x = pd.read_csv("sms-spam.csv")

        #displaying the dataframe
        x

Out[2]:
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will Ì_ b going to esplanade fr home?	NaN	NaN	NaN

Figure 4.

The image shows a Jupyter Notebook with Python code related to SMS spam detection.

Key Observations:

1. **Import Statements:** The code starts by importing necessary libraries:
   - numpy and pandas for data manipulation.
   - LabelEncoder for encoding categorical data.

- matplotlib.pyplot and seaborn for data visualization.
- nltk for natural language processing tasks like stop word removal and stemming.
- wordcloud for creating word clouds.
- CountVectorizer and TfidfVectorizer for text vectorization.
- train_test_split for splitting data into training and testing sets.
- GaussianNB, MultinomialNB, BernoulliNB for Naive Bayes classifiers.
- Metrics like accuracy_score, precision_score, and confusion_matrix for evaluating model performance.

2. **Data Loading:** The code reads an SMS spam dataset from a CSV file into a pandas DataFrame.
3. **Data Exploration (Partially Shown):** The code likely continues with data exploration and preprocessing steps, such as:
   - Handling missing values (if any).
   - Cleaning the text data (e.g., removing punctuation, converting to lowercase).
   - Removing stop words (common words like “the,” “a,” “is”).
   - Applying stemming or lemmatization to reduce words to their root form.
   - Creating word clouds to visualize the most frequent words in spam and ham messages.
4. **Feature Extraction:** The code will likely use CountVectorizer or TfidfVectorizer to convert the text data into numerical features that can be used by machine learning models.
5. **Model Training and Evaluation:** The code will then split the data into training and testing sets, train different machine learning models (like Naive Bayes), and evaluate their performance using appropriate metrics.

Overall, this Jupyter Notebook provides a framework for building an SMS spam detection model. It demonstrates the use of common data science and NLP techniques for data preprocessing, feature extraction, model training, and evaluation.

4.1GitHub Link for Code:

You can access the code for the SMS Spam Detection project on GitHub at the following link: SMS Spam Detection GitHub Repository.

CHAPTER 5

DISCUSSION AND CONCLUSION

Future Scope:

While the current SMS Spam Detection system serves as an effective tool for identifying and filtering spam messages, there remains significant potential for improvement and expansion. These enhancements could make the system more versatile, accurate, and capable of addressing diverse user needs and emerging spam tactics. Below are several areas for future development:

1. Multilingual Support

The current model is predominantly designed to process and analyze English-language SMS messages. Expanding its capabilities to handle multiple languages would greatly increase its usability and relevance across different linguistic regions.

- Diverse Language Processing: Many users communicate in languages other than English, including regional and non-Latin script languages. Incorporating multilingual NLP techniques and datasets could help the system analyze messages written in languages such as Spanish, Hindi, Mandarin, and Arabic.
- Universal Application: By supporting multiple languages, the system could cater to a global audience, making it more inclusive and practical for users worldwide.

2. Adoption of Advanced Deep Learning Techniques

While the current system likely relies on traditional machine learning models, exploring deep learning could unlock new possibilities for accuracy and efficiency.

- Recurrent Neural Networks (RNNs): RNNs and their variants, such as Long Short-Term Memory (LSTM) networks, are excellent for sequential data like text. These models could better capture the context and meaning of SMS messages.
- Transformers and Pre-Trained Models: Leveraging state-of-the-art architectures like Transformers (e.g., BERT or GPT) could significantly enhance the system's

ability to understand complex spam patterns, idiomatic expressions, and subtle nuances in language.

- Improved Spam Detection: Deep learning models, trained on large and diverse datasets, would improve detection accuracy and reduce false positives and negatives.

3. Real-Time Adaptation

Spam tactics are constantly evolving, requiring systems to stay updated to remain effective. Implementing real-time adaptation mechanisms would make the system more dynamic.

- Continuous Learning: By incorporating techniques like online learning, the system could update its model incrementally as new spam patterns emerge.
- Periodic Retraining: Scheduling automated retraining sessions with fresh data would ensure the model remains robust against new types of spam.
- Dynamic Rule Updates: Alongside machine learning, updating spam detection rules based on real-time feedback and analysis could further enhance adaptability.

4. Integration with Messaging Platforms

Integrating the SMS Spam Detection system directly into popular messaging platforms would extend its reach and usability.

- Automated Spam Filtering: Messaging apps like WhatsApp, Telegram, and SMS services could use the system to filter spam messages before they reach the user’s inbox.
- Enhanced User Experience: By providing users with built-in spam detection capabilities, these platforms could offer a safer communication environment.
- Cross-Platform Security: Expanding the system to email services and social media platforms could further protect users from phishing attempts and fraudulent communication.

5. Enhanced User-Centric Features

Adding more functionality to the application could improve user interaction and satisfaction.

- Customizable Thresholds: Allowing users to adjust the sensitivity of spam detection based on their preferences could make the system more personalized.
- Spam Reporting and Feedback Loop: Enabling users to report undetected spam messages or incorrectly flagged messages would contribute to improving the model's accuracy over time.
- Visualization and Insights: Providing users with detailed insights and trends about the spam messages they receive could educate them about evolving spam tactics.

6. Scalability and Cloud Integration

To handle large volumes of messages across multiple users, scaling the system is essential.

- Cloud-Based Deployment: Hosting the system on cloud platforms like AWS, Google Cloud, or Azure would ensure high availability and scalability.
- API Services: Offering spam detection as an API could allow developers to integrate the system into various applications seamlessly.

By implementing these enhancements, the SMS Spam Detection system could become a robust, efficient, and scalable solution. It would not only cater to a broader audience but also address emerging challenges in spam detection with greater precision and adaptability. These improvements would make the system a critical tool in safeguarding users against unwanted and malicious communications, contributing to a more secure and seamless digital experience.

Conclusion:

To sum up, the SMS Spam Detection system efficiently detects and filters spam messages using machine learning and Natural Language Processing (NLP) approaches. Tokenisation, stop-word removal, and stemming are examples of preprocessing techniques that the system uses to help convert raw text into a format that is appropriate for model training. By using TF-IDF for feature extraction, classification performance is improved by ensuring that key words are identified.

When it came to managing text-based data, the Naive Bayes classifier outperformed the other algorithms, proving its effectiveness and dependability. Metrics like precision, recall, and F1-score demonstrate the model's excellent accuracy, which shows that it can correctly categorise SMS texts.

The mechanism may be improved, though. It's still difficult to handle more intricate spam patterns, including those with many languages or advanced spamming strategies.

REFERENCES

1. Almeida, T. A., Gómez Hidalgo, J. M., & Yamakami, A. (2011). Contributions to the study of SMS spam filtering: New collection and results.
2. Jurafsky, D., & Martin, J. H. (2021). Speech and Language Processing.
3. Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python.
4. Streamlit Documentation: https://docs.streamlit.io/.