RAMNIRANJAN JHUNJHUNWALA COLLEGE

Of Arts, Science & Commerce (Empowered Autonomous Status)

Ghatkopar (West), Mumbai - 86

Project Report On

"Spam Message Detector"

By

Satyam Kumar Dubey

M.Sc.-II (Computer Science)

University Of Mumbai

2024-2025

Project Guide

Prof. Anita Gaikwad

RAMNIRANJAN JHUNJHUNWALA COLLEGE

Of Arts Science & Commerce(Empowered Autonomous Status)

Ghatkopar (West), Mumbai - 86

2024-2025

A project report on

“Spam Message Detector”

Towards partial fulfillment of the degree of M.Sc. (Computer Science) as prescribed by the University of Mumbai.

By

Satyam Kumar Dubey

Project Guide

Prof. Anita Gaikwad

RAMNIRANJAN JHUNJHUNWALA COLLEGE

Of Arts Science & Commerce(Empowered Autonomous Status)
Ghatkopar (West), Mumbai - 86

2024-2025

CERTIFICATE

This is to certify that **Mr. Satyam Kumar Dubey** has successfully completed the project titled,

"Spam Message Detector"

Under the guidance towards the partial fulfillment of degree of Master of Science (Computer Science) submitted to Ramniranjan Jhunjhunwala College, Ghatkopar(W) during the academic year 2023-2024 as specified by the UNIVERSITY OF MUMBAI.

Guide	Course Coordinator
(Prof. Anita Gaikwad)	Department Of Computer Science
	Prof. Anita Gaikwad

4

ACKNOWLEDGMENTS

I would like to extend my heartfelt gratitude. First & foremost, I express my sincere appreciation to The Principle **Dr. Himanshu Dawda** for their unwavering support, mentorship, dedication to the academic & personal growth of students & encouragement throughout this endeavor.

I would also like to acknowledge MSC. Coordinator **Prof. Anita Gaikwad** for their support & guidance in navigating the administrative aspects of this project. Your leadership has been indispensable.

I am deeply thankful to my Project Guide **Prof. Anita Gaikwad**, my mentor & guide, for their invaluable guidance, expertise & continuous encouragement. Your insights have immensely enriched the quality of this project.

I also owe to my fellow friends who have been a constant source of help to solve the problems & help with the challenges faced during the project development phase.

Thank You,
Satyam Kumar Dubey

INDEX

1. Introduction

In the age of instant communication, the spread of fake messages has become a critical challenge, fueling misinformation, social unrest, and even cybersecurity threats. Messaging platforms and social media networks are primary vectors for false information, where deceptive messages can go viral within seconds. Conventional fact-checking approaches, while effective, are slow, labor-intensive, and inadequate for countering large-scale misinformation campaigns. This calls for an advanced, intelligent system that can detect and mitigate fake messages with high accuracy and efficiency.

To address this issue, this project introduces an AI-powered Spam Message Detection System**,** integrating Natural Language Processing (NLP), Deep Learning (DL), and Graph-based Analysis to identify and classify deceptive messages in real time. Unlike traditional keyword-based detection, this system employs context-aware analysis, sentiment evaluation, and linguistic forensics to understand the deeper meaning and intent behind messages.

Data Collection and Preprocessing

The dataset used in this project consists of news headlines and full-text articles stored in TSV files. Each entry in the dataset contains attributes such as:

- Title – The headlines of spam /not spam messages
- Source – Kaggle.com
- Label – Classification as "not spam" or "spam"

Since raw text data often contains noise, redundancies, and inconsistencies, a crucial step in this project is data preprocessing, which includes:

- **Tokenization** – Breaking down news articles into individual words or phrases.
- **Stopword Removal** – Eliminating common words (e.g., "is," "the," "and") that do not contribute to meaning.
- **TF-IDF (Term Frequency-Inverse Document Frequency)** – Assigning importance to words based on their frequency in the dataset while reducing the influence of common words.

Text Extraction and Feature Engineering

Once preprocessing is completed, text feature extraction is performed to convert unstructured text into a structured format suitable for machine learning. The following techniques are used:

- TF-IDF Vectorization – Measures word importance in the document.

Machine Learning and Data Modeling

To classify message as not spam or spam, various machine learning models are trained and evaluated. The key models considered include:

- Logistic Regression – A simple yet effective linear model for binary classification.
- Naïve Bayes Classifier – A probabilistic model useful for text-based classification.
- Support Vector Machines (SVM) – A model that finds the optimal boundary for classification.
- Random Forest – A decision-tree-based ensemble method for robust predictions.
- Deep Learning (LSTMs, Transformers) – Advanced models that capture long-term dependencies in text.

The model is trained using labeled news articles, and its performance is evaluated based on accuracy, precision, recall, and F1-score. The best-performing model is then deployed for real-time news classification.

2. Project Background Studies

1. Introduction to Spam Message and Its Impact

Fake messages have become a major concern in the digital age, where misinformation spreads rapidly through messaging platforms, social media, and online communication channels. The circulation of deceptive messages can mislead individuals, cause unnecessary panic, and contribute to the spread of false narratives. Traditional verification methods, which rely on manual fact-checking, are slow and ineffective in managing the vast volume of messages exchanged daily.

To tackle this challenge, Natural Language Processing (NLP) and Deep Learning have emerged as powerful solutions for automated fake message detection. Transformer-based models like BERT (Bidirectional Encoder Representations from Transformers) and RoBERTa have demonstrated state-of-the-art performance in text classification tasks, making them highly effective in identifying misleading content. By leveraging these advanced techniques, fake message detection systems can analyze linguistic patterns, detect anomalies, and classify messages in real time, improving the accuracy and efficiency of misinformation detection.

2. Evolution of Spam Message Detection Techniques

A. Early Approaches: Rule-Based and Machine Learning Models

Initial spam message detection relied on rule-based approaches, where handcrafted linguistic rules were used to identify misleading content. However, these systems were limited in adaptability and failed against evolving misinformation tactics.

Machine learning methods, such as Naïve Bayes, Logistic Regression, and Support Vector Machines (SVMs), improved accuracy by leveraging word frequency analysis (TF-IDF, Bag of Words) to differentiate between real and fake news. However, these models struggled with:

- Contextual understanding (word meaning changes based on context).
- Generalization (dificulty in detecting new types of misinformation).

B. Deep Learning and Transformer Models

With advancements in deep learning, models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs) were introduced to handle sequential dependencies in text. However, these models suffered from vanishing gradient issues and struggled with long-range dependencies.

The introduction of Transformers, particularly BERT, revolutionized NLP tasks, including fake news detection. Unlike previous models, BERT uses:

- Bidirectional Context Understanding – It considers both left and right context of a word, unlike traditional word embeddings.

- Pretrained Knowledge – Trained on massive corpora (e.g., Wikipedia, BooksCorpus) and fine-tuned for specific tasks.
- Self-Attention Mechanism – Assigns different importance to words in a sentence, improving text classification accuracy.

3. Implementation of naïve Bayes for Spam Message Detection

This project leverages naïve bayes based classification, using the following components:

1. Implementation of Spam message classification with help of non-conditional probability model of naïve bayes.
2. Trainer API (Hugging Face) – Handles training, evaluation, and optimization with minimal manual tuning.
3. TrainingArguments – Defines hyperparameters like learning rate, bais, and epochs to optimize model performance.

4. Dataset and Preprocessing

The dataset used in this project is stored in TSV (Tab-Separated Values) format, which is widely used in NLP tasks due to its structured nature. Each record in the dataset includes:

- Title – The spam/not spam messages.
- Label – A binary classification (1 = Spam, 0 = Not Spam).

Preprocessing Steps for BERT

Before training the model, raw text undergoes several preprocessing steps:
Tokenization – Converting sentences into tokens using CountVectorizer.
Padding & Truncation – Ensuring uniform input lengths by padding shorter texts and truncating longer ones.
Attention Masks – Identifying important words while ignoring padding tokens.

5. Challenges in Spam Message Detection with Naïve Bayes

Despite its effectiveness, Spam message detection using naïve bayes faces several challenges:

1. Computational Cost – Transformer models require high GPU/TPU resources, making them expensive to train.

2. Handling of Ambiguous and Evolving Spam

Spam messages continuously evolve with new patterns, word variations, and obfuscation techniques (e.g., replacing "free" with "fr33" or inserting spaces like "f r e e"). Naïve Bayes relies on word frequency distributions, making it vulnerable to such manipulations and reducing its effectiveness against adversarial spam tactics.

3. High Sensitivity to Training Data Quality

The accuracy of a Naïve Bayes spam filter heavily depends on the quality and balance of the training dataset. If the dataset contains biased or insufficient examples of spam and ham (legitimate messages), the model may suffer from poor generalization and high false positive/false negative rates.

4. Difficulty in Handling Short Messages

Many spam messages, especially SMS and instant messages, are short and lack sufficient word diversity. Since Naïve Bayes relies on word occurrences for classification, it may struggle with messages that contain too few words to make a reliable classification.

5. Lack of Context Awareness and Sentiment Analysis

Naïve Bayes treats words as individual features and does not consider semantic meaning, tone, or intent. This can lead to incorrect classification, especially when spam messages mimic legitimate communication styles (e.g., phishing emails that appear professional).

Conclusion

Despite its simplicity and efficiency, Naïve Bayes has several limitations in spam message detection. To improve performance, it is often combined with **advanced NLP techniques, deep learning models, or hybrid approaches** that incorporate contextual understanding and adaptive learning mechanisms.

3. Objectives and key features

1. Project Objectives

The primary objective of this project is to develop an efficient and real-time spam message classification system using a Naïve Bayes-based or deep learning model**.** The system aims to classify messages as spam or legitimate (ham) based on their textual content. The following objectives guide the project:

A. Build an Effective Spam Classification Model

- Implement a Naïve Bayes or Transformer-based model to differentiate between spam and legitimate messages with high accuracy.
- Fine-tune the classification model using hyperparameter optimization to enhance performance.

B. Process and Analyze Text Data

- Use TSV-based datasets to store and preprocess message for eficient handling.
- Apply text preprocessing techniques, including tokenization, stopword removal, stemming.
- Perform data cleaning and preprocessing, including removing unnecessary characters.

C. Train and Evaluate the Model

- Train the model using labeled datasets to learn patterns in fake and real news articles.
- Evaluate accuracy using Trainer.evaluate(), and track performance with precision, recall, F1-score, and loss functions.
- Ensure robust generalization to avoid overfiting by using appropriate validation strategies.

D. Real-Time Spam Message Detection

- Deploy the trained model to analyze and classify news articles in real-time.
- Implement a user-friendly interface or API for real-time predictions.

E. Improve Model Eficiency and Scalability

- Optimize model performance using techniques like distillation (DistilBERT) or quantization.
- Ensure scalability by designing the system to handle large-scale news articles and datasets.

2. Key Features of the Project

Transformer-Based Text Classification

- Uses BERTForSequenceClassification, a deep learning model that captures context and meaning in news text.
- Bidirectional attention mechanism helps the model understand the full sentence structure.

TSV-Based Dataset Handling

- The system works with Tab-Separated Values (TSV) files for easy storage and processing.
- The dataset consists of news headlines, full content, and labels (real/fake).

Advanced Text Preprocessing

- Tokenization using BertTokenizer ensures eficient text representation.
- Automatic padding and truncation handle varying text lengths.
- Attention masks help the model focus on meaningful words while ignoring padding.

4.Software Specifiactions and technology stack

1. Software Specifications

This section outlines the software requirements and dependencies used for building the fake news detection system. The project utilizes deep learning techniques, particularly a BERT-based model, to classify news articles as real or fake.

1.1 Operating System

- The model and scripts are designed to run on:
    - Windows 10/11
    - Linux (Ubuntu 18.04 and above)
    - macOS (with proper Python and PyTorch support)

1.2 Programming Language

- **Python 3.8+**: Used for all development, including data preprocessing, model training, and evaluation.

1.3 Development Environment and IDEs

- Jupyter Notebook (for prototyping and experimentation)
- VS Code / PyCharm (for structured development)
- Google Colab (for training models on cloud GPUs)

1.4 Libraries and Frameworks

The following libraries and frameworks are used for data processing, model training, and evaluation:

Machine Learning & Deep Learning

- **Transformers** (by Hugging Face): Used for **BertTokenizer** and **BertForSequenceClassification**.
- **Torch (PyTorch)**: Backbone deep learning framework for model implementation.

Data Processing

- **Pandas**: For handling tabular data (TSV files) and dataset manipulation.
- **NumPy**: For numerical computations and array handling.
- **scikit-learn**: For dataset spliting, evaluation metrics, and preprocessing.

Training and Evaluation

- **Trainer API (Hugging Face)**: Simplifies training, evaluation, and fine-tuning of transformer models.
- **TrainingArguments (Hugging Face)**: Defines hyperparameters such as learning rate, batch size, and number of epochs.

1.5 Hardware Requirements

- **Processor**: Intel i5/i7, AMD Ryzen 5/7, or Apple M1/M2.
- **RAM**: Minimum 8GB (16GB+ recommended for large datasets).
- **GPU** (Recommended for faster training): NVIDIA GPU (RTX 2060 or better) with CUDA support or Google Colab's free GPU.

2. Technology Stack

The technology stack consists of various tools, frameworks, and libraries used for model training, data handling, and deployment.

2.1 Data Handling and Storage

- **File Format**: TSV (Tab-Separated Values) format for dataset storage.
- **Pandas**: Used for loading, cleaning, and transforming the dataset.

- **train_test_split (sklearn.model_selection)**: Splits data into training and validation sets.

2.2 Model Architecture and Training

- **BERT (Bidirectional Encoder Representations from Transformers)**: Pre-trained transformer-based model for NLP tasks.
- **BertTokenizer**: Tokenizes the input text to a format suitable for BERT.
- **BertForSequenceClassification**: Pre-trained BERT model fine-tuned for binary classification (real vs. fake news).
- **Trainer API**: Used for eficient training, evaluation, and optimization.
- **TrainingArguments**: Defines hyperparameters like batch size, learning rate, and logging.

2.3 Model Evaluation Metrics

- **Accuracy**: Percentage of correctly classified news articles.
- **Precision**: Measures how many predicted fake news articles were actually fake.
- **Recall**: Measures how many actual fake news articles were correctly classified.
- **F1-score**: Harmonic mean of precision and recall, balancing false positives and false negatives.

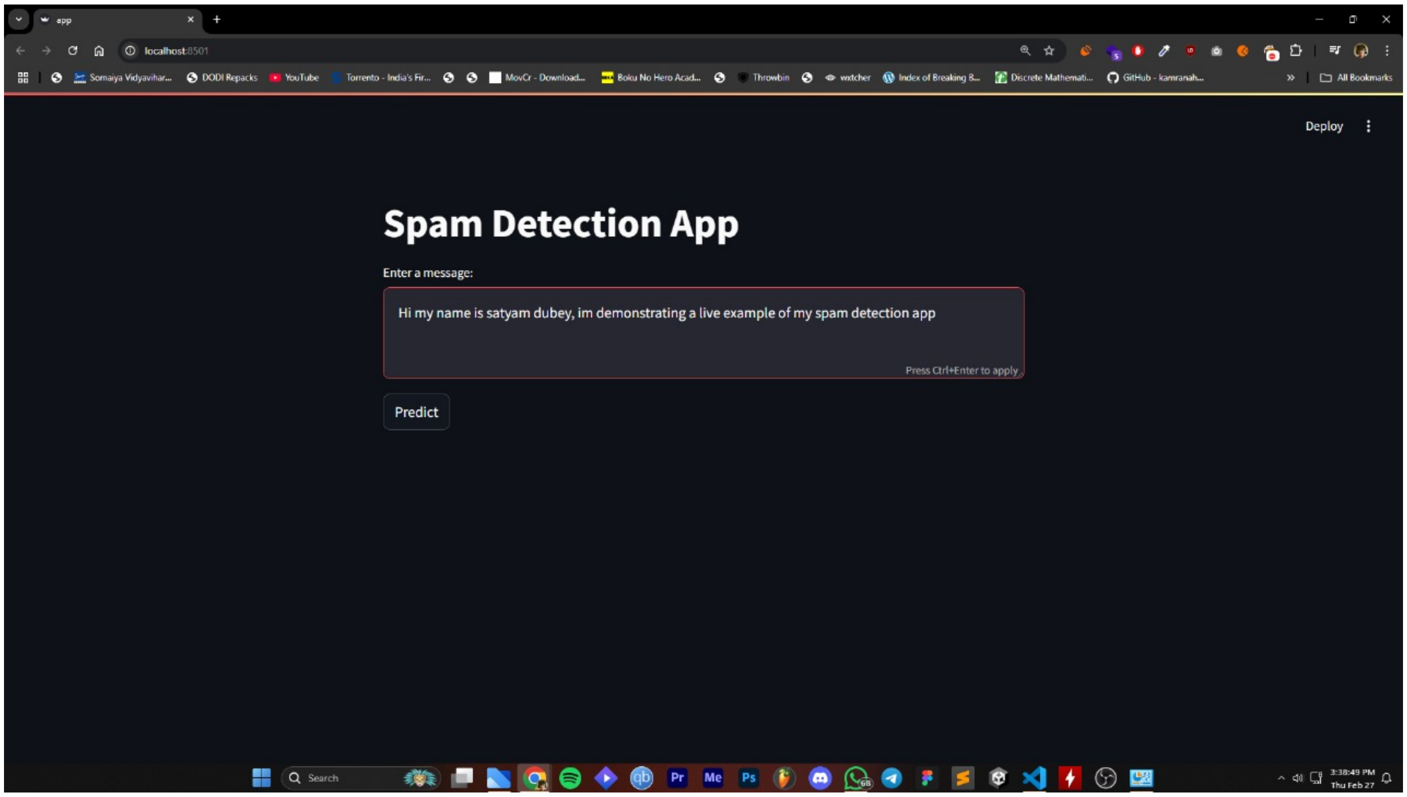
2.4 Model Training and Implementation

- **Dataset**: The LIAR dataset, stored in TSV format, is used for training and evaluation.
- **Data Preprocessing**:
  - Text tokenization using **Sklearn**.
  - Label encoding to convert categorical labels into numerical format.
- **Custom PyTorch Dataset Class**:
  - Created to structure the data for PyTorch's DataLoader.
  - Enables batching and shufling for eficient training.
- **Model Training**:
  - Uses **Trainer API** from Hugging Face.
  - Defines **TrainingArguments** (batch size, learning rate, warmup steps, etc.). o Saves trained model using save_pretrained().

2.5 Deployment Using Streamlit

- **Streamlit**:
  - Used to build an interactive web application for real-time fake news detection. o Allows users to enter a news statement and get instant classification results.
- **Deployment Steps**:
  - Load the pre-trained BERT model and tokenizer.
  - Create a simple UI with a text input field for user-provided news.
  - Tokenize and classify the input using the trained model.
  - Display results in a user-friendly manner.
- **Advantages of Streamlit**:
  - Fast and lightweight web application development.
  - No need for complex backend frameworks like Flask or FastAPI.
  - Easily deployable on cloud platforms like Heroku or AWS.

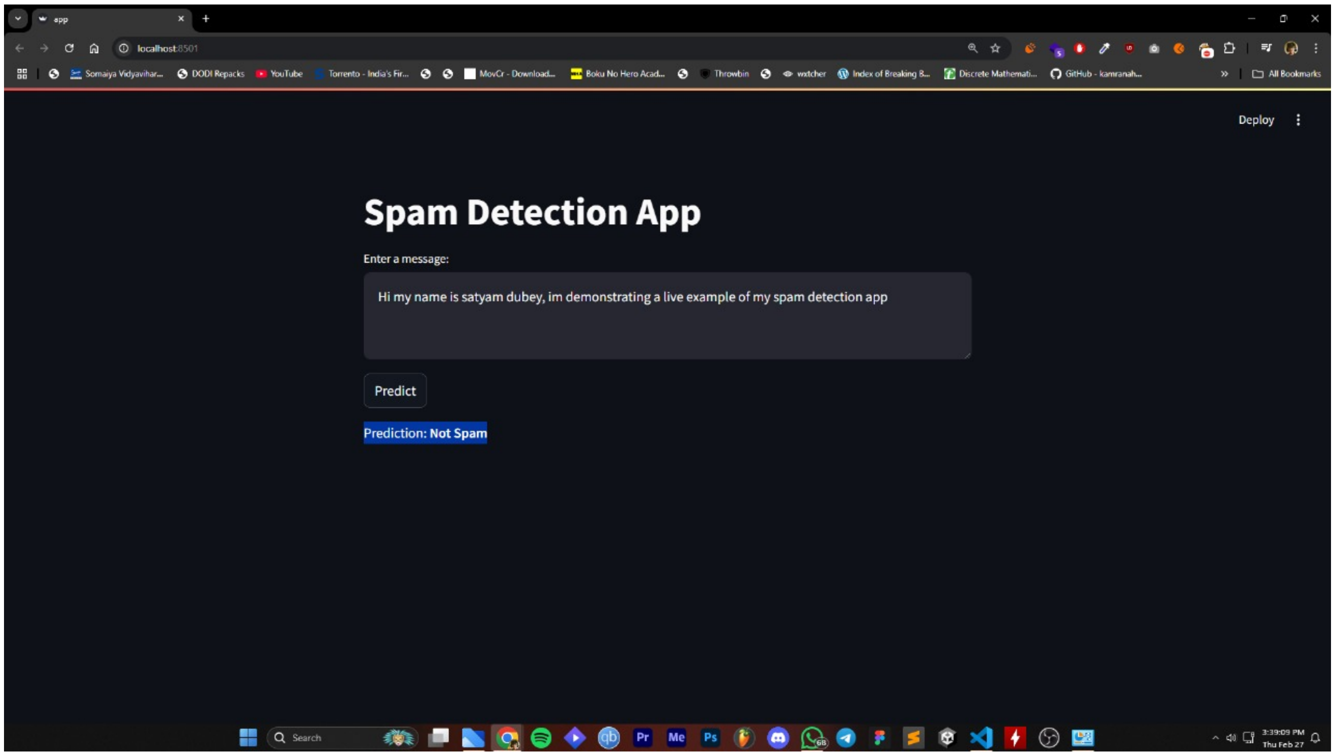
5. Screenshots of Project



Spam Detection App

Enter a message:

Hi my name is satyam dubey, im demonstrating a live example of my spam detection app



Spam Detection App

Enter a message:

Hi my name is satyam dubey, im demonstrating a live example of my spam detection app

Prediction: Not Spam

6. Source Code

Model Train :

```
import pandas as pd
 import numpy as np
 import seaborn as sns
 import matplotlib.pyplot as plt
 from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
 from sklearn.model_selection import train_test_split
 from sklearn.metrics import roc_auc_score, f1_score, confusion_matrix
 from sklearn.naive_bayes import MultinomialNB
  # Load data
 Data = pd.read_csv('/content/spam.csv',encoding = 'ISO-8859-1')
  data.head()
  # Data Statistics
  data.info()
  data.describe

  # Dropping the unwanted columns
   data = data.drop(columns=data.columns[2:5])
    data.head()
   data.columns = ['Category', 'Message']
  data()
  # Checking Null values
   data.isnull().sum()
  # Visualization data
  category_counts = data['Category'].value_counts().reset_index()
   category_counts.columns = ['Category', 'Count']
   plt.figure(figsize=(8, 6)) sns.barplot(x='Category', y='Count',data=category_counts)
   plt.xlabel('Category')
   plt.ylabel('Count')
   plt.title('Category Distribution')
    for i, count in enumerate(category_counts['Count']):
       plt.text(i, count, str(count), ha='center', va='bottom')
   plt.show()
```

```
# Train and Test data

    data['spam']=data['Category'].apply(lambda x:1 if x=='spam' else 0)
    data.head(5)
    from sklearn.feature_extraction.text import CountVectorizer
    featurer = CountVectorizer()
    x_train_count = featurer.fit_transform(x_train.values)
    x_train_count
#Applying naïve bayes
   model_1 = MultinomialNB() model_1.fit(x_train_count,y_train)
    X_test_count = featurer.transform(x_test)
    model_1.score(x_test_count,y_test)
#Applying logistic regression
   From sklearn.linear_model import LogisticRegression
    model = LogisticRegression()
    model.fit(x_train_count,y_train)
    x_test_count = featurer.transform(x_test)
   model.score(x_test_count, y_test)
# Make pipeline
   from sklearn.pipeline import Pipeline
    clf = Pipeline([ ('vectorizer', CountVectorizer()), ('nb', MultinomialNB()) ])
   clf.fit(x_train,y_train)
  clf.score(x_test,y_test)
# Streamlit UI
    st.title("📰 Spam Detection App")
    st.write("Enter a news article and check if it's **spam** or **not spam**.")

    # Text input
    news_text = st.text_area("Enter a message:", height=200)

    if st.button("Analyze"):
        if news_text.strip() == "":
            st.warning("Please enter some text before analyzing.")
        else:
            # Tokenize input text


            inputs = tokenizer(news_text, return_tensors="pt", truncation=True, max_length=500)

            # Run model prediction
            with torch.no_grad():
                logits = model(**inputs).logits
```

```
# Get the label (0 = Fake, 1 = Real)
predicted_label = torch.argmax(logits).item()
label_map = {0: "🛑 spam", 1: "✅ Not spam"}

# Display result
st.subheader(label_map[predicted_label])
```

7.Scope and Limitations

Scope of the Project

The Spam message detection system is designed to classify spam message as spam or notspam using a naive-bayes model. The project aims to provide a reliable solution for identifying misinformation in textual data. Below are the key aspects defining the scope of this project:

- **Dataset Utilization**: The model is trained using the LIAR dataset, which contains labeled news statements categorized into six levels of truthfulness.
- **Real-Time Classification**: The system supports real-time detection of fake news by accepting user input and providing instant classification results.
- **Deep Learning-Based Approach**: Utilizes BERT (Bidirectional Encoder Representations from Transformers) for natural language understanding and classification.
- **Deployment**: The model is deployed using Streamlit, enabling an interactive web-based interface for end users.
- **Scalability**: The model can be fine-tuned with additional datasets to improve accuracy and adapt to different contexts of fake news detection.
- **User-Friendly Interface**: Provides an easy-to-use application where users can input news statements and receive classification results.

Limitations of the Project

While the project offers a robust spam message detection mechanism, certain limitations exist:

- **Dataset Dependency**: The model is trained on the LIAR dataset, and its accuracy depends on the quality and diversity of the training data.
- **Contextual Understanding**: BERT-based models, while powerful, may misinterpret sarcasm, satire, or highly nuanced text that requires deeper contextual knowledge.
- **Binary or Limited Classification**: The model classifies message into predefined categories and may not account for the evolving nature of misinformation.
- **Language Limitations**: The current model is trained on English datasets, making it ineffective for detecting fake news in other languages.
- **Model Bias**: Training data biases may impact classification results, potentially leading to incorrect predictions.
- **Computational Requirements**: Training and fine-tuning the sklearn model require significant computational resources, making it difficult to run on low-end hardware.
- **No Fact-Checking Mechanism**: The model does not verify the authenticity of sources but classifies statements based on learned patterns.

Despite these limitations, the project provides a strong foundation for automated spam message detection and can be improved further with additional data, better NLP techniques, and enhanced fact-checking integrations.

8.Future Enhancements

1. Multilingual Support
Expanding the model to support multiple languages will make it applicable to a broader audience and improve misinformation detection across different linguistic contexts.

2. Integration with Fact-Checking Databases
Linking the system to reliable fact-checking sources such as PolitiFact, Snopes, and FactCheck.org can help cross-verify statements in real-time.

3. Improved Context Understanding
Enhancing the model's ability to differentiate between satire, sarcasm, and actual misinformation by integrating advanced NLP techniques.

4. Enhanced Data Collection
Using larger and more diverse datasets from different sources, including social media, news websites, and independent fact-checking organizations, to improve classification accuracy.

5. Real-Time Web Scraping
Implementing web scraping techniques to analyze news articles as they are published and provide real-time insights into misinformation trends.

6. Explainability and Transparency
Developing an explainable AI approach to help users understand why a particular statement is classified as fake or real, improving trust in the system.

7. Optimized Model for Deployment
Creating lightweight versions of the model for mobile and low-resource devices, ensuring accessibility for a wider user base.

8. Adaptive Learning Mechanism
Incorporating self-learning capabilities so the model can adapt to new misinformation patterns over time and improve its predictions.

9. User Feedback Integration
Allowing users to submit feedback on classification results to enhance the system's performance and ensure continuous improvement.
10. Social Media Integration
Developing plugins or APIs that allow users to fact-check news directly from social media platforms like Twitter and Facebook.

.

9.References and Bibliography

Research Papers and Articles

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention Is All You Need*. Advances in Neural Information Processing Systems (NeurIPS).
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv preprint arXiv:1810.04805.
- Rashkin, H., Choi, E., Jang, J. Y., Volkova, S., & Choi, Y. (2017). *Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking*. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.
- Thorne, J., Vlachos, A., Christodoulopoulos, C., & Mittal, A. (2018). *FEVER: A Large-Scale Dataset for Fact Extraction and Verification*. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics.

Datasets Used

- LIAR: A dataset containing labeled statements from various political sources (Available Online).
- FakeNewsNet: A collection of annotated fake news articles for research in misinformation detection.

Libraries and Frameworks

- Hugging Face Transformers (https://huggingface.co/transformers/)
- PyTorch (https://pytorch.org/)
- Scikit-learn (https://scikit-learn.org/)
- Pandas (https://pandas.pydata.org/)

Tools and Technologies

- Google Colab and Jupyter Notebook for model training and experimentation.
- Streamlit for deploying the fake news detection system.
- GitHub for version control and collaboration.