République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
ECOLE SUPÉRIEURE EN INFORMATIQUE
8 Mai 1945 - Sidi-Bel-Abbès

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
المدرسة العليا للإعلام الآلي
8 ماي 1945 - سيدي بلعباس

Ecole Supérieure en Informatique 08-MAI-1945
Sidi Bel Abbes

Mini-project report

SMS spam detection

Présenté par:
- Benghenima Hafsa
- Ghandouz Amina

Encadrant :
- Dr. Boussmaha Rabab

1. Introduction

SMS remains one of the most widely used communication tools, especially in personal communication, but it also plays a role in professional contexts such as authentication, alerts, and marketing. However, its widespread use makes it a prime target for spam—unsolicited messages that often contain advertisements, scams, or malicious links. The ability to accurately detect and filter spam SMS is crucial for maintaining security, reducing distractions, and improving user experience.

Traditional rule-based spam filters have largely been replaced by machine learning approaches, which learn to distinguish spam from legitimate messages based on patterns in the data. With the increasing availability of labeled datasets and advancements in natural language processing (NLP), the performance of spam detection systems has significantly improved.

This project aims to develop and compare two different approaches for SMS spam detection:

1. A baseline model using classical machine learning techniques such as TF-IDF for feature extraction and classifiers like Naive Bayes and Logistic Regression.

2. A more advanced model leveraging transformer-based architectures such as BERT or ALBERT, which are capable of understanding deeper contextual information in text.

2. Dataset Overview

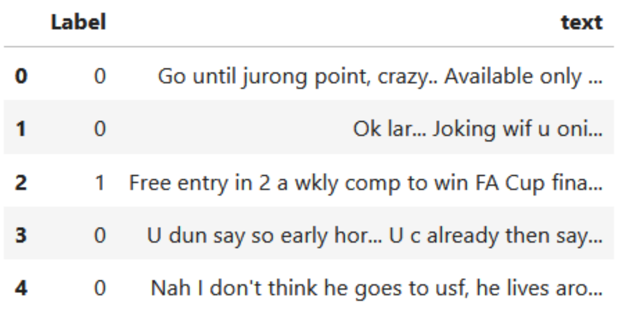
For this project, we used the publicly available **SMS Spam Detection** dataset provided on Kaggle. The dataset consists of SMS messages labeled as either "**spam**" or "**ham**" (i.e., non-spam). Each row contains the raw text of an SMS and a corresponding label indicating whether it is spam.

2.1 Dataset Structure

The dataset includes the following columns:

- text: The full content of the SMS.
- label: The class label (spam or ham).

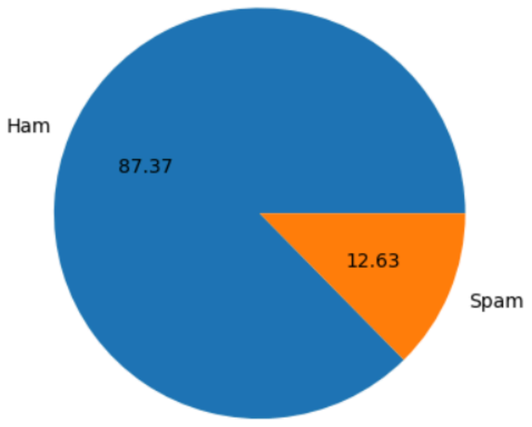
This structure makes the dataset well-suited for text classification tasks and allows us to directly apply standard natural language processing techniques.



	Label	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

2.2 Class Distribution

An important characteristic of the dataset is the imbalance in class distribution, with a significantly larger number of "ham" SMSs compared to "spam". This is a common issue in real-world spam detection scenarios, where legitimate SMSs tend to outnumber spam messages.



This imbalance poses a challenge during training, as models can become biased toward predicting the majority class. Proper strategies to mitigate this issue are discussed in later sections of the report.

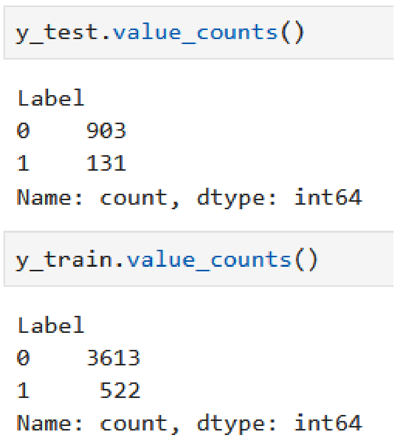
2.3 Data Quality

The dataset is generally well-structured, with no missing values. However, during initial exploration, we identified a small number of duplicate entries, which were subsequently removed to avoid bias and redundancy during training. The remaining SMS texts exhibit a wide range of styles, lengths, and linguistic patterns, reflecting the natural complexity of real-world SMS communication. This diversity provides a solid foundation for evaluating both basic and advanced classification models.

3. Baseline Approach

3.1 Data Preprocessing

To ensure robust evaluation, we first split the dataset into training and testing sets with 80% of the data allocated for training and 20% for testing. To preserve the proportion of each class in both sets, we used stratified splitting. This approach ensures that the class distribution (spam vs. ham) is consistent across both the training and testing sets, preventing class imbalance from affecting the model's performance evaluation.



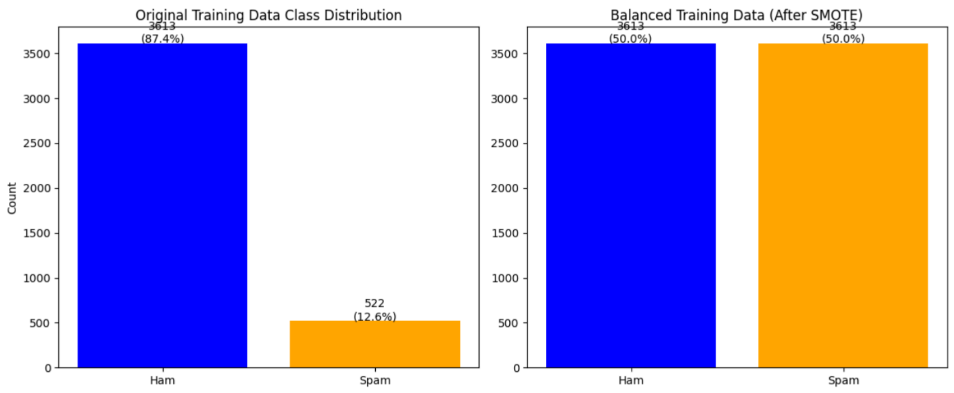
```
y_test.value_counts()
```

```
Label
0    903
1    131
Name: count, dtype: int64
```

```
y_train.value_counts()
```

```
Label
0    3613
1     522
Name: count, dtype: int64
```

To address the class imbalance, we applied SMOTE (Synthetic Minority Over-sampling Technique) on the training data. However, since SMOTE works only with numerical features, we first applied TF-IDF (Term Frequency-Inverse Document Frequency) to transform the raw text into numerical features. TF-IDF was applied only to the training data to prevent data leakage, ensuring that the test data remains unseen during the preprocessing phase. This step is crucial because IDF (Inverse Document Frequency) uses information from the entire dataset, and applying it to the test set could inadvertently give the model access to information it would not have in a real-world scenario. By fitting the TF-IDF transformer solely on the training data, we avoid this issue, ensuring the integrity of our evaluation process.



3.2 Text Normalization

We experimented with various text normalization techniques to prepare the SMS text for classification. As part of the preprocessing, we performed the following steps:

- Lowercasing: All SMS text was converted to lowercase to ensure uniformity, as the model should not distinguish between words based solely on case .

- Punctuation Removal: We removed all punctuation marks from the text to avoid treating them as part of the words, which might introduce unnecessary noise in the feature representation.

Interestingly, stemming did not improve performance in this context. Spam detection often relies on the presence of specific keywords or word forms (e.g., "win", "winner", "winning"), and stemming tends to remove this nuance, which could lead to losing important context for identifying spam. Therefore, we opted to retain the original word forms in our final preprocessing pipeline.

3.3 Model Training Process

After preprocessing, we trained and evaluated three traditional machine learning classifiers:

- **Logistic Regression (LR)**
- **Gaussian Naive Bayes (GNB)**
- **Random Forest (RF)**

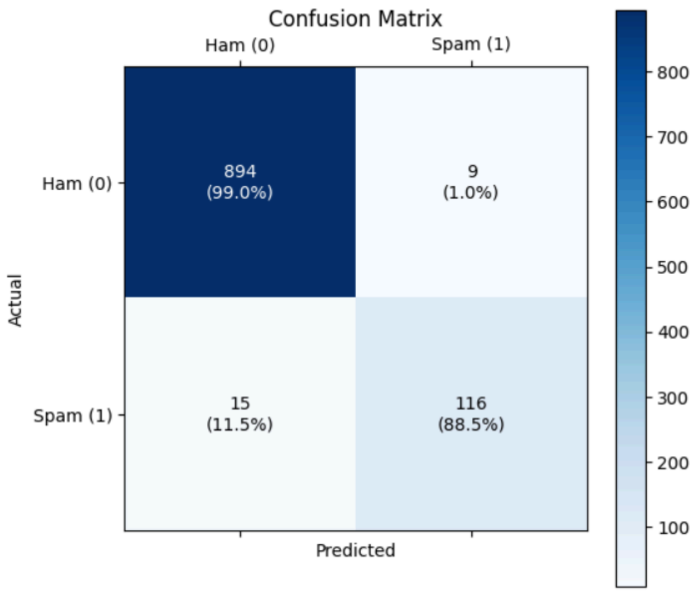
These models were chosen for their simplicity, interpretability, and proven performance in text classification tasks. We evaluated each model based on several performance metrics, and the results are summarized in the table below.

Model	Recall	F1-Score	ROC-AUC
Logistic Regression	0.98	0.98	0.99
Gaussian Naive Bayes	0.88	0.89	0.87
Random Forest	0.95	0.95	0.97

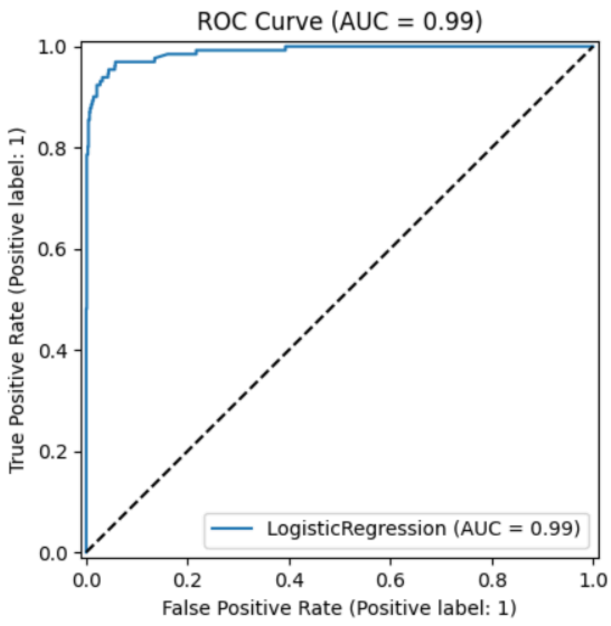
The Logistic Regression (LR) model outperformed the other classifiers, yielding the best balance of accuracy, precision, and recall, particularly for detecting spam. Based on this, we selected Logistic Regression as the best model for our baseline approach.

3.4 Best Model: Logistic Regression

Confusion Matrix:



ROC Curve:



Given the combination of high accuracy, strong performance in detecting spam, and favorable confusion matrix and AUC scores, we conclude that Logistic Regression (LR) is the best model for this task in the baseline approach. This model strikes an optimal balance between complexity and interpretability while providing reliable results for spam detection.

4. Transformer-based Approach

While traditional machine learning models like Logistic Regression (LR) offer simplicity and efficiency, they often struggle to capture the rich contextual relationships between words. In tasks like SMS spam detection, subtle patterns and phrasing can indicate whether a message is spam or ham. To address the limitations of classical models, we turn to a more advanced solution using transformer-based architectures, specifically BERT (Bidirectional Encoder Representations from Transformers).

BERT is pre-trained on large corpora using a masked language modeling objective, allowing it to capture deep contextual information and semantic relationships within text. This makes it particularly effective for classification tasks where meaning and word order are important, such as spam detection. In this approach, we fine-tune a pre-trained BERT model on the SMS spam dataset and compare its performance with the traditional models used in the baseline.

4.1 Understanding BERT: Architecture and Mechanism

BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based model developed by Google in 2018. It marked a major advancement in the field of Natural Language Processing (NLP) due to its deep contextual understanding of language. Unlike traditional models that read text either left-to-right or right-to-left, BERT reads in both directions simultaneously, capturing richer linguistic patterns and dependencies.

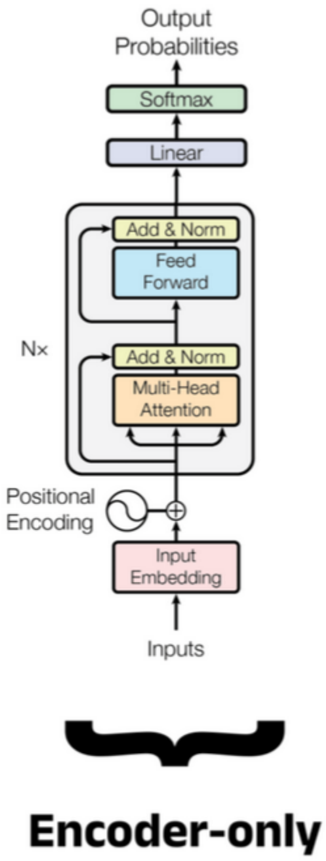
4.1.1 Architecture Overview

BERT is built solely on the encoder portion of the original Transformer architecture introduced by Vaswani et al. (2017). The core building block of BERT is the transformer encoder layer, which includes:

- Multi-head Self-Attention Mechanism:Lets BERT understand how each word relates to others in the sequence.

- Feedforward Neural Networks:Applies learned transformations to capture complex patterns.

- Residual Connections and Layer Normalization: Used to stabilize and accelerate training.



In this project, we use BERT Base (bert-base-uncased), which includes:

- 12 transformer encoder layers
- Add a final classification layer with 2 neurons, corresponding to the Ham and Spam classes.
- 768-dimensional hidden representations
- 12 self-attention heads
- A total of around 110 million trainable parameters

4.1.2 Input Representation

BERT takes as input a sequence of tokens preceded by a **[CLS]** token and optionally separated by a **[SEP]** token for sentence-pair tasks. Each input token is represented by the sum of three embeddings:

- Token Embedding: Represents the vocabulary item.
- Segment Embedding: Distinguishes different parts (e.g., Sentence A and Sentence B).

- Position Embedding: Encodes the position of each token in the sequence.

These embeddings allow BERT to encode not just word identities, but also their positions and relationships, then they are passed through multiple Transformer encoder layers.

4.1.3 Fine-tuning for Spam Detection

In this project, we fine-tune BERT for binary text classification (Spam vs. Ham). During fine-tuning:

- The input sequence is tokenized and padded to a fixed length (128 tokens).
- The hidden representation of the [CLS] token (from the final layer) is passed through a classification head (a linear layer) to predict the class.
- The model is trained using cross-entropy loss with class weights to handle the data imbalance.
- Only minimal preprocessing (e.g., lowercasing and punctuation removal) is required since BERT handles raw language effectively.

4.2 Data Preparation

We used the cleaned version of the SMS dataset, ensuring no missing values or duplicates were present. The dataset was split into 80% training and 20% test sets, with stratification applied to preserve the proportion of spam and ham messages across both sets.

4.3 Tokenization and Input Formatting

We used the bert-base-uncased tokenizer to preprocess the messages. Each SMS was:

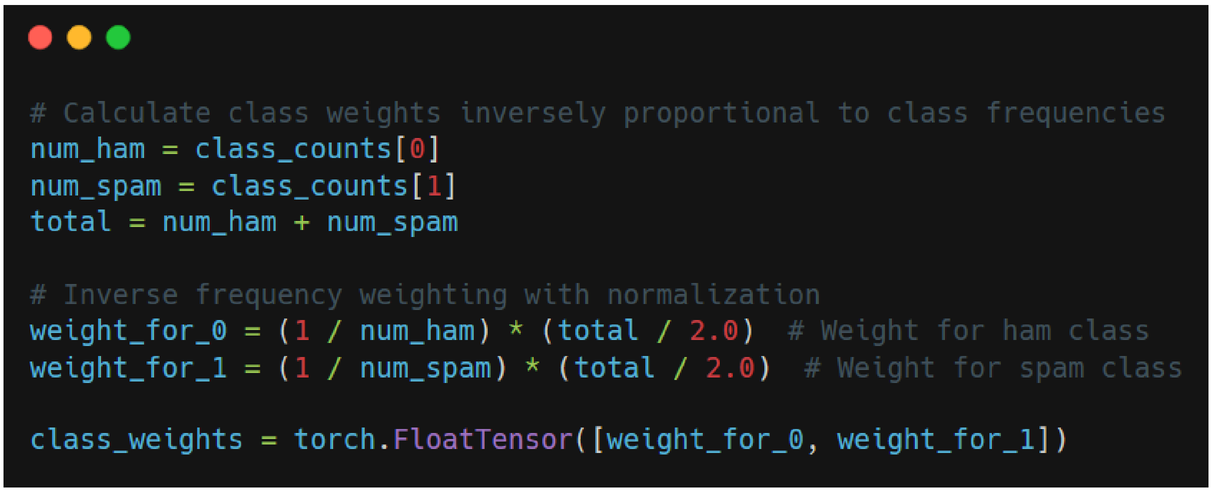
- Lowercased automatically by the tokenizer
- Tokenized and converted to token IDs
- Padded/truncated to a fixed length of 128 tokens
- Encoded to return input IDs and attention masks (that is a binary mask (1 for real tokens, 0 for padding) ensures the model ignores padding during self-attention)

A custom PyTorch Dataset class was implemented to convert the tokenized inputs and labels into a format compatible with BERT's DataLoader.

4.4 Addressing Class Imbalance

The dataset exhibited class imbalance, with the ham class being significantly more frequent than spam (4516 ham vs. 653 spam). To mitigate this:

- We computed class weights inversely proportional to class frequencies.



```
# Calculate class weights inversely proportional to class frequencies
num_ham = class_counts[0]
num_spam = class_counts[1]
total = num_ham + num_spam

# Inverse frequency weighting with normalization
weight_for_0 = (1 / num_ham) * (total / 2.0)  # Weight for ham class
weight_for_1 = (1 / num_spam) * (total / 2.0)  # Weight for spam class

class_weights = torch.FloatTensor([weight_for_0, weight_for_1])
```

- These weights were integrated into the cross-entropy loss function during training. This helped penalize misclassified spam messages more heavily, encouraging the model to be more sensitive to them.
- This approach: assigns higher weights to the minority class (spam)and makes the model pay more attention to spam samples during training.

4.5 Model Configuration and Training

We fine-tuned the pre-trained BertForSequenceClassification model with the following setup:

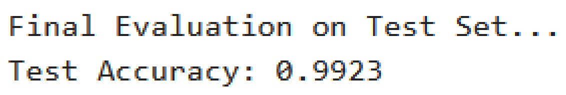
- **Loss function**: CrossEntropyLoss with class weights
- **Optimizer**: AdamW with learning rate 2e-5
- **Scheduler**: Linear schedule with warm-up and no decay
- **Epochs**: 3
- **Batch size**: 16

The training loop included gradient clipping, scheduler stepping, and evaluation at each epoch.

4.6 Evaluation

After training, we evaluated the model on the test set using:

- **Accuracy**



```
Final Evaluation on Test Set...
Test Accuracy: 0.9923
```

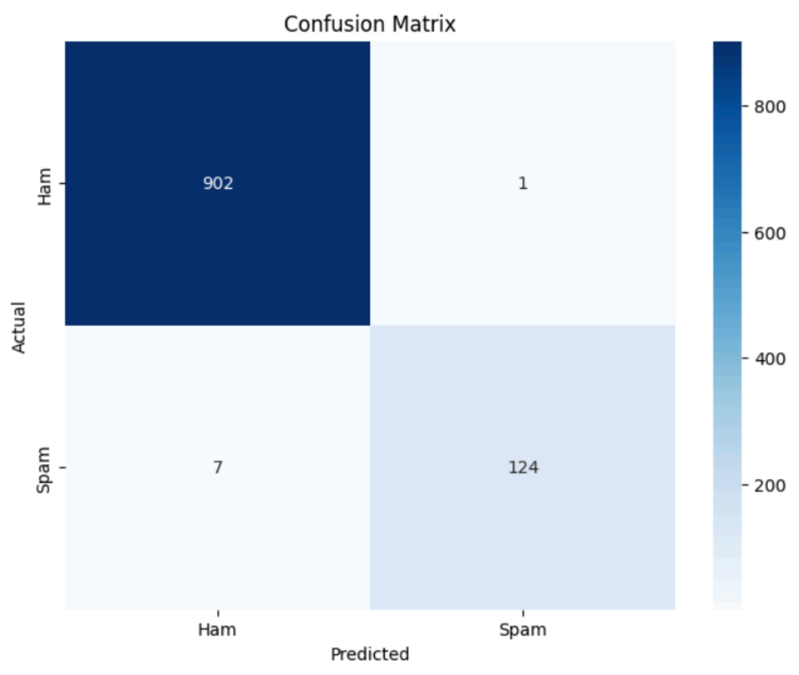
- **Classification report**

```
Classification Report:
              precision    recall  f1-score   support

         Ham       0.99      1.00      1.00       903
        Spam       0.99      0.95      0.97       131

    accuracy                           0.99      1034
   macro avg       0.99      0.97      0.98      1034
weighted avg       0.99      0.99      0.99      1034
```

- **Confusion matrix**

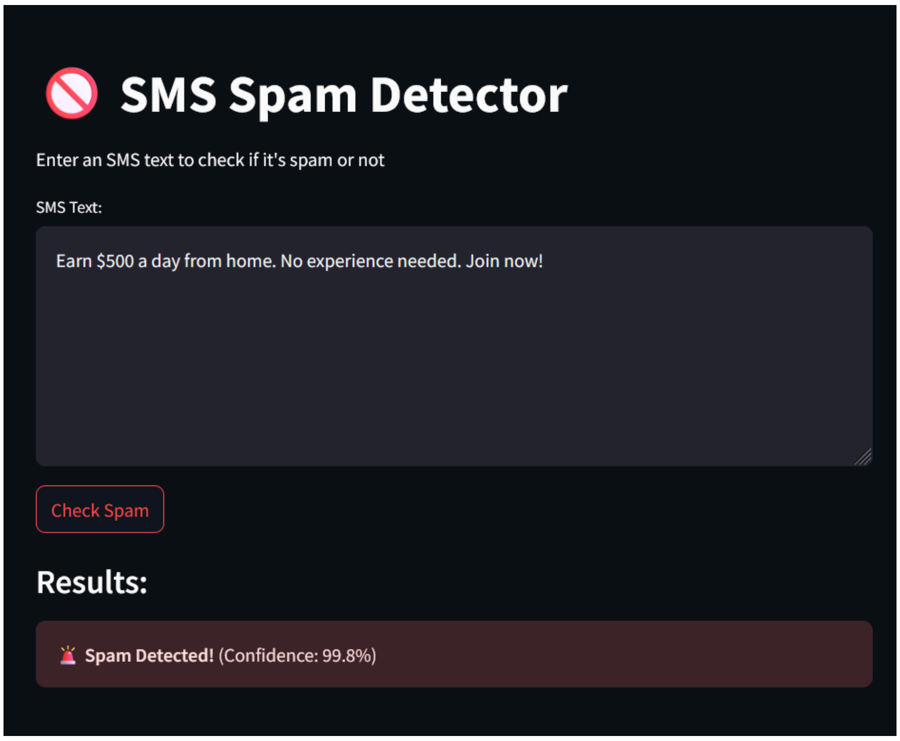


The model demonstrated **high accuracy and balanced performance**, particularly in identifying spam messages, highlighting BERT's capacity to generalize without explicit feature engineering.

4.7 Streamlit Interface

To make the BERT-based spam detection model accessible, a simple user interface was built using Streamlit. The user can input a message, and the app returns whether it is Spam or Ham, along with the model's confidence.

Example Tests



🚫 SMS Spam Detector

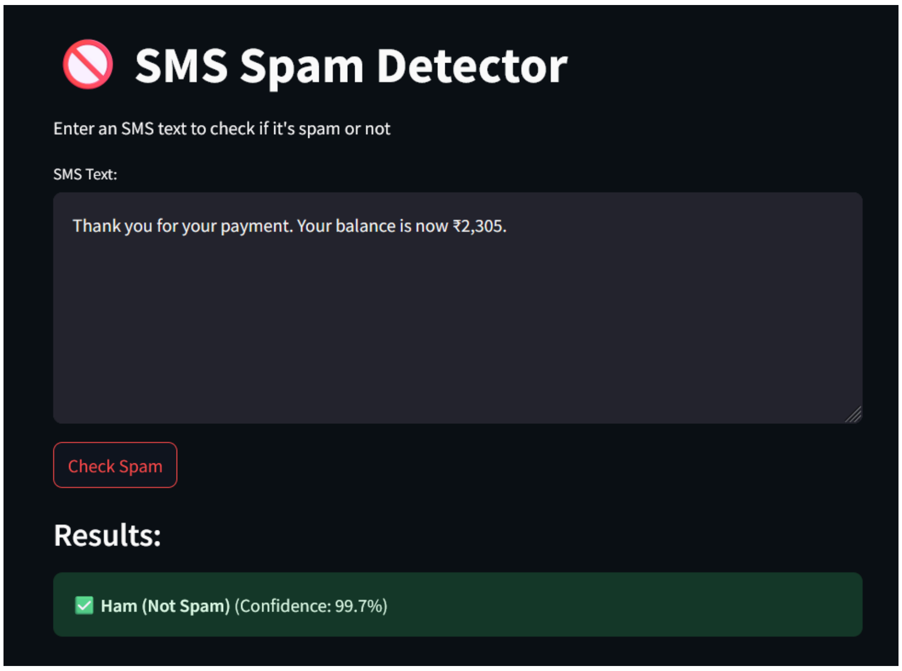
Enter an SMS text to check if it's spam or not

SMS Text:

Earn $500 a day from home. No experience needed. Join now!

Results:

🚨 **Spam Detected!** (Confidence: 99.8%)



🚫 SMS Spam Detector

Enter an SMS text to check if it's spam or not

SMS Text:

Thank you for your payment. Your balance is now ₹2,305.

Results:

✅ **Ham (Not Spam)** (Confidence: 99.7%)

Conclusion

In this project, we explored two approaches for SMS spam detection: a traditional machine learning pipeline using TF-IDF with models like Logistic Regression, and an advanced deep learning method using BERT. The classical approach, after proper preprocessing and handling class imbalance, achieved strong results, with Logistic Regression performing the best among tested models.

However, the transformer-based approach (BERT) outperformed traditional models by effectively capturing contextual meaning without heavy preprocessing. It also handled class imbalance through weighted loss functions and demonstrated high accuracy and robustness on real-world examples.

Finally, a simple Streamlit interface was developed to make the system user-friendly and easily testable. This project highlights how both traditional and modern NLP methods can be leveraged for text classification, with BERT offering a more powerful yet computationally intensive solution.

References

- Project Repository: GitHub Link
- Article: “A Complete Guide to BERT with Code” – towardsdatascience