The objective of this project is to build a **machine learning model** that can classify SMS messages as either **Spam** or **Ham (Not Spam)**.
This task demonstrates the application of **Natural Language Processing (NLP)** techniques for real-world text classification problems such as spam filtering in messaging platforms.

Dataset Used

- **Source:** SMS Spam Collection Dataset (UCI Machine Learning Repository)
- **Total Records:** 5574 messages
- **Classes:**
    - **Ham (Normal):** ~87%
    - **Spam:** ~13%

The dataset is **imbalanced**, meaning the number of normal messages far exceeds the number of spam messages — a critical factor that initially affected model performance.

Steps & Methodology

Data Preprocessing

- Converted all text to **lowercase**
- Removed **numbers**, **punctuation**, and **special characters**
- **Tokenized** messages into words
- Removed **stopwords** (like "the", "and", "is")
- Applied **stemming** to reduce words to their base form (e.g., "running" → "run")

Why:
These steps clean and normalize the text so the model focuses only on meaningful patterns rather than irrelevant formatting or noise.

Feature Extraction (TF-IDF)

- Used **TF-IDF (Term Frequency–Inverse Document Frequency)** to convert text into numeric features.

- Included **n-grams (1–2)** to capture important word combinations like “free ticket” or “won prize”.

Why TF-IDF over Bag of Words (BoW):

- BoW only counts how often a word appears.
- TF-IDF gives **more importance to rare but meaningful words** (e.g., “lottery”, “claim”, “free”).
- This improves the model’s ability to detect spam indicators.

Model Training

Trained two supervised learning models:

- **Naive Bayes Classifier**
- **Logistic Regression**

Also, handled **class imbalance** by **upsampling spam messages** to match ham messages.

Why These Models:

Model	**Reason**
Naive Bayes	Simple and effective for text classification, assumes word independence, fast to train.
Logistic Regression	Performs well with high-dimensional sparse data like TF-IDF, provides better recall and interpretability.

Evaluation Metrics

Evaluated models using:

- **Accuracy**
- **Precision**
- **Recall**
- **F1-Score**
- **Confusion Matrix**

Why These Metrics:
In spam detection, **Recall and F1-score** are more critical than accuracy.
We want to **minimize false negatives** (spam misclassified as ham), since missing a spam message is more harmful than mistakenly flagging a ham message.

Results & Comparison

Metric	Naive **Bayes**	**Logistic Regression**
Accuracy	0.95	0.97
Precision (Spam)	0.90	0.95
Recall (Spam)	0.91	0.96
F1-Score (Spam)	0.90	0.95

Final Chosen Model: Logistic Regression with TF-IDF (1–2 n-grams)
Dataset Handling: Balanced via upsampling
Vectorization: TF-IDF (captures rare, meaningful words)

Why Logistic Regression (Final Choice)

Aspect	Naive Bayes	Logistic Regression
Handling Correlated Words	Weak	Strong
Works with TF-IDF weights	Limited	Excellent
Interpretability	Moderate	High
Generalization	Good	**Better**
Final Spam Recall	0.91	**0.96 (Higher)**

Conclusion: Logistic Regression achieved **higher recall and precision** on spam messages, meaning it correctly flagged spam more often without many false alarms.
This makes it more reliable for real-world spam filtering systems.

Example Predictions

Message	Predicted Class
"Congratulations! You have won a free lottery ticket!"	Spam ✅
"Hey, can we meet tomorrow for lunch?"	Ham ✅
"URGENT! Claim your prize by replying now."	Spam ✅
"Your appointment is scheduled at 4 PM."	Ham ✅

Deliverables

- naive_bayes_sms_model.pkl
- logistic_regression_sms_model.pkl
- tfidf_vectorizer.pkl
- Confusion Matrix Visualizations
- End-to-End Colab Notebook with code + results

Conclusion

This project successfully demonstrates an end-to-end **text classification pipeline** using NLP. Through preprocessing, feature extraction, and model training, we achieved a **97% accurate** spam detector.
Balancing the dataset and using **TF-IDF with Logistic Regression** improved recall and reduced spam misclassification.

Key Takeaway:
Choosing the right feature representation (TF-IDF) and handling class imbalance were more impactful than simply changing algorithms.