**SMS Spam Classifier – A Data Analytics Case Study**

**Introduction**

Spam messages are not only inconvenient but can also pose serious threats, including scams and phishing attempts. As mobile communication continues to grow, so does the importance of intelligent systems that can automatically detect and filter out these unwanted messages.

In this project, I applied natural language processing (NLP) and machine learning to build a model capable of classifying SMS messages as either spam or ham (legitimate). The goal was not just to build a working classifier, but to go through the full cycle of a real-world data analytics project—from problem definition to model evaluation—while maintaining a focus on clarity, interpretability, and business value.

**Problem Statement**

The dataset used in this project consists of labelled SMS messages from the UCI Machine Learning Repository. Each message is marked as either "spam" or "ham." The core challenge was to develop a model that could accurately identify spam messages, minimizing false positives (flagging legitimate messages as spam) and false negatives (missing actual spam).

A significant hurdle was the class imbalance present in the data: approximately 87% of the messages were legitimate, and only 13% were spam. This imbalance needed to be addressed to prevent biased model predictions.

**Objectives**

* Clean and prepare unstructured text data for analysis.
* Address data imbalance through appropriate sampling techniques.
* Evaluate multiple machine learning models and select the best-performing one.
* Understand which preprocessing and vectorization techniques yield the best results.
* Interpret model outcomes and assess business impact.

**Data Understanding and Exploration**

The dataset contained two main columns:

* label: the target variable (spam or ham)
* message: the raw SMS text content

Initial exploration revealed:

* Significant class imbalance between spam and ham messages
* Variations in message length and language structure between the two classes

These observations helped shape the preprocessing and modelling strategies.

**Preprocessing and Feature Engineering**

Text data was pre-processed using the following steps:

* Converted all text to lowercase
* Removed punctuation and special characters
* Removed common stop words using NLTK
* Applied stemming to reduce words to their root forms

Vectorization was done using:

* **Count Vectorizer**: a simple term-frequency approach
* **TF-IDF Vectorizer**: which accounts for word importance across messages

TF-IDF generally produced better results, especially when paired with certain models like Naive Bayes.

**Handling Class Imbalance**

Given the skewed distribution of classes, I applied techniques to balance the dataset:

* **Random Oversampling**: duplicating minority class (spam) examples
* **SMOTE (Synthetic Minority Oversampling Technique)**: creating synthetic spam samples

These techniques significantly improved the model’s ability to detect spam, particularly in terms of recall.

**Modelling and Evaluation**

Three models were trained and compared:

* Naive Bayes
* Logistic Regression
* Support Vector Machine (SVM)

Evaluation metrics included:

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

**Naive Bayes with TF-IDF features** emerged as the top performer, offering a strong balance between precision and recall, with an overall accuracy of approximately **97%**. It was particularly effective at identifying spam while minimizing false positives.

**Key Insights**

* Basic NLP techniques, when properly applied, can produce highly effective spam filters.
* Addressing class imbalance is essential in classification tasks involving skewed datasets.
* Simpler models like Naive Bayes, when paired with strong features, can outperform more complex models in certain scenarios.
* TF-IDF is a powerful and interpretable feature extraction method for text classification tasks.

**Tools and Technologies**

* **Python**: Core programming language
* **Pandas and NumPy**: Data manipulation
* **NLTK**: Text preprocessing
* **Scikit-learn**: Machine learning models and evaluation
* **Matplotlib and Seaborn**: Data visualization
* **Jupyter Notebook**: Interactive development environment

**Conclusion**

This project offered a comprehensive exercise in text classification, model evaluation, and problem-solving from a data analytics perspective. It demonstrated that with the right preprocessing and thoughtful handling of imbalance, even relatively simple models can deliver excellent performance.

The approach taken here not only provides immediate value in terms of classification accuracy but also serves as a scalable and interpretable framework for similar problems in other domains.

**Future Directions**

If extended further, this project could include:

* Deep learning models such as LSTM or BERT for improved semantic understanding
* A deployment-ready web application using Flask or Stream lit
* Integration with real-time SMS monitoring systems

**About the Analyst**

I’m Harshika, a recent graduate with a strong interest in data analysis and machine learning. Through projects like this SMS Spam Classifier, I’ve gained practical experience in working with real-world data, applying NLP techniques, and building predictive models.