**CA1 DATA SCIENCE AND**

**MACHINE LEARNING PORTFOLIO LOG**

**Date: February 28th 2025**

**Prepared by:**

Kirubel Temesgen

Student Number: C00260396

**Course:** Data Science and Machine Learning

**Supervisor:** Greg Doyle

## Project Overview

This project focuses on building and evaluating a spam classification model using machine learning. **Three models—Naïve Bayes, Support Vector Machines (SVM), and K-Nearest Neighbors (KNN**)—are compared to determine the best performer in detecting spam emails.

The **Enron Email Dataset**, containing over **500,000 real-world emails**, is used for training and testing**. A CRISP-DM (Cross-Industry Standard Process for Data Mining)** approach structures the workflow, ensuring systematic data preparation, feature engineering, and model evaluation.

## Phase 1: Business Understanding

**Business Objective**

The goal of this project is to build an efficient spam detection model that can automatically classify emails as **spam (1)** or **not spam (0).**

**Key Benefits:**

Reduces time spent on manually filtering spam.

Helps detect phishing emails using text-based patterns.

**Improves email security** by filtering harmful messages.

**Key Questions:**

Which machine learning model performs best in spam detection?

What preprocessing techniques improve classification accuracy?

How do feature extraction methods impact model performance?

Can the model be optimised to minimise false positives and false negatives?

## CRISP-DM Phase 2: Data Understanding

### 1. Data Collection

The Enron Email Dataset was chosen for its real-world applicability. It includes:

Over 500,000 emails from Enron employees.

A mix of spam and legitimate business emails.

Metadata: From, To, Subject, and Body fields.

### 2. Data Description & Structure

Initial exploration of **517,401 emails** revealed:

**Columns:** file, message, From, To, Subject, Body

**Missing Values:** To column had 8,929 missing values (dropped).

**Duplicates:** No exact duplicate emails found.

### 3. Exploratory Data Analysis (EDA)

An overall understanding on how the data can be filtered through effectively.

Most **active senders**: Emails came from employees like kay.mann@enron.com and vince.kaminski**@enron.com**.

Most common subject words: RE:, FW:, and business-related terms like "meeting", "agreement", and "energy".

**Visualisation techniques used:**

Bar charts for sender frequency.

Word frequency analysis for common spam terms.

## CRISP-DM Phase 3: Data Preparation

### Step 1: Handling Missing Values

The **To column was removed** as it was not relevant for spam classification.

Missing values in From, Subject, and Body were dropped to maintain data quality.

### Step 2: Feature Engineering & Text Processing

**Key features extracted for machine learning:**

**Processed\_Subject** – Cleaned subject line (lowercase, no special characters).

**Processed\_Body** – Cleaned body text.

**Sender Domain** – Extracted from the From field.

**Word Count** – Total words in Subject & Body.

**Additional Processing:**

**TF-IDF Vectorization** – Converts **text into numerical** **form** for ML models.

**Stopword Removal** – Removes common words like the, and, and is.

**N-grams** – Uses bigrams (two-word combinations) to improve context understanding.

Example: Instead of analysing "free" and "offer" separately, the model learns "free offer" as a meaningful spam phrase.

### Step 3: Spam Labeling Strategy

Since the dataset does not explicitly label spam, I implemented **rule-based filtering**:

Keyword-Based Filtering: Emails containing **spam-like words** such as **"win**", "l**ottery**", "**free** **offer**", and "**click here**" were classified as spam.

Domain-Based Filtering: Emails from non-Enron domains (**not ending in @enron.com**) were **marked as potential spam**.

**Final Labeling Approach:**

**Spam (1):** If the email contains spam-related words OR comes from a non-Enron sender.

**Not Spam (0):** Otherwise.

### Step 4: Splitting & Training Data

**TF-IDF vectorized features** were used to train the model.

80% of emails used for training - - 20% reserved for testing.

Final Training Set Size: (41392, 1000)

Final Test Set Size: (10348, 1000)

## CRISP-DM Phase 4: Model Evaluation (Naïve Bayes, SVM, KNN)

### 1. Model 1 - Naïve Bayes Classifier

To assess the effectiveness of spam detection, I implemented **a Multinomial Naïve Bayes model** using TF-IDF vectorized features.

**Why Naïve Bayes?**

**Fast and scalable** for large text datasets.

Works well with **word frequency-based** features.

Good for **text classification** tasks like spam detection.

**Model Performance Metrics (Naïve Bayes)**

**Overall Accuracy: 83.91%**

| **Metric** | **Not Spam (0)** | **Spam (1)** | **Macro Avg** | **Weighted Avg** |
| --- | --- | --- | --- | --- |
| **Precision** | 0.87 | 0.75 | 0.81 | 0.83 |
| **Recall** | 0.92 | 0.62 | 0.77 | 0.84 |
| **F1-Score** | 0.89 | 0.68 | 0.79 | 0.83 |

**Confusion Matrix Analysis (Naïve Bayes)**

* **True Positives** (Spam correctly identified): **17,887**
* **True Negatives** (Non-Spam correctly classified): **68,944**
* **False Positives** (Non-Spam mistakenly flagged as Spam): **5,908**
* **False Negatives** (Spam incorrectly classified as Non-Spam): **10,742**

**Key Observations:**

* High precision (0.87) for non-spam emails, meaning few false positives.
* Lower recall (0.62) for spam, meaning some spam emails were missed.
* Balanced F1-score (0.83), but further improvements possible.

### 2. Model 2 - Support Vector Machine (SVM)

To improve classification performance, I implemented an SVM model using TF-IDF vectorized features.

**Why SVM?**

Finds the **optimal decision boundary** (hyperplane) for classification.

Works well for **text classification** and high-dimensional data.

More robust to noisy data compared to Naïve Bayes.

**Model Performance Metrics (SVM)**

**Overall Accuracy: 89.77%**

| **Metric** | **Not Spam (0)** | **Spam (1)** | **Macro Avg** | **Weighted Avg** |
| --- | --- | --- | --- | --- |
| **Precision** | 0.89 | 0.92 | 0.91 | 0.90 |
| **Recall** | 0.98 | 0.69 | 0.83 | 0.90 |
| **F1-Score** | 0.93 | 0.79 | 0.86 | 0.89 |

**Confusion Matrix Analysis (SVM)**

* Higher precision (0.92) for spam detection compared to Naïve Bayes.
* Recall for spam (0.69) is higher than Naïve Bayes, meaning fewer missed spam emails.
* Better overall accuracy (89.77%) due to SVM’s ability to create a more effective separation between spam and non-spam.

**Key Observations:**

* SVM outperforms Naïve Bayes in terms of accuracy and recall for spam detection.
* Training takes longer, but once trained, the model is very efficient.
* May not work well for extremely large datasets due to computational complexity.

### 3. Model 3 - k-Nearest Neighbors (k-NN)

To explore another approach, I implemented a k-NN classifier with TF-IDF vectorized features.

**Why k-NN?**

Simple, intuitive, and **non-parametric** (no need for explicit training).

Useful for datasets with non-linear decision boundaries.

No training phase, but **slow for large datasets**.

**Model Performance Metrics (k-NN)**

**Overall Accuracy: 82.04%**

| **Metric** | **Not Spam (0)** | **Spam (1)** | **Macro Avg** | **Weighted Avg** |
| --- | --- | --- | --- | --- |
| **Precision** | 0.83 | 0.77 | 0.80 | 0.81 |
| **Recall** | 0.94 | 0.49 | 0.72 | 0.82 |
| **F1-Score** | 0.88 | 0.60 | 0.74 | 0.81 |

**Confusion Matrix Analysis (k-NN)**

* High recall for non-spam (0.94) but low recall for spam (0.49), meaning more spam emails were misclassified.
* Accuracy (82.04%) is lower than SVM and Naïve Bayes.
* Performs well with small datasets but struggles with large-scale data.

**Key Observations:**

* Does not explicitly train a model, just stores all data and compares distances during prediction.
* Performance is **highly dependent on the choice of k (number of neighbors)**.
* Computationally expensive, especially with large datasets.

**Comparison of Models**

| **Model** | **Accuracy** | **Pros** | **Cons** |
| --- | --- | --- | --- |
| **Naïve Bayes** | 83.91% | Fast, works well for text, handles large datasets | Assumes feature independence, struggles with complex patterns |
| **SVM** | 89.77% | Effective for high-dimensional data, finds optimal boundaries | Computationally expensive, slower training |
| **k-NN** | 82.04% | Simple, non-parametric, works for non-linear data | Slow prediction, struggles with large datasets |

## CRISP-DM Phase 5: Deployment & Future Improvements

The model successfully classifies emails into **spam or not spam**, improving email security.

**Key Findings:**

* **SVM achieved the highest accuracy (89.77%)**, making it the most effective model in this case.
* **Naïve Bayes performed well (83.91%) but struggled with recall for spam detection**.
* **k-NN had the lowest accuracy (82.04%) and struggled with spam classification**, making it less suitable for this dataset.

**Future Enhancements:**

* Optimise feature engineering (e.g., using word embeddings like Word2Vec instead of TF-IDF).
* Use deep learning models (LSTMs, transformers) to improve classification performance.
* Experiment with hybrid models (e.g., combining Naïve Bayes with SVM) for better results.

## References

* **Pandas**: McKinney, W., 2023. *Pandas: Python Data Analysis Library*. Available at: [General functions — pandas 2.2.3 documentation](https://pandas.pydata.org/docs/reference/general_functions.html#data-manipulations) [Accessed 22 October 2024].
* **Matplotlib**: Hunter, J.D., 2007. *Matplotlib: A 2D Graphics Environment*.[online] Available at: [Plot types — Matplotlib 3.9.2 documentation](https://matplotlib.org/stable/plot_types/index.html) [Accessed 22 October 2024].
* matplotlib.org. (n.d.). Quick start guide — Matplotlib 3.8.4 documentation. [online] Available at: <https://matplotlib.org/stable/users/explain/quick_start.html#types-of-inputs-to-plotting-functions> [Accessed 22 Oct. 2024].
* matplotlib.org. (n.d.). Tight layout guide — Matplotlib 3.8.4 documentation. [online] Available at: <https://matplotlib.org/stable/users/explain/axes/tight_layout_guide.html> [Accessed 22 Oct. 2024].
* scikit-learn.org. (n.d.). Linear Regression — scikit-learn 1.3.1 documentation. [online] Available at: [1.1. Linear Models — scikit-learn 1.5.2 documentation](https://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares) [Accessed 22 Oct. 2024].
* Awan, A.A. and Navlani, A. (2023) *Naive Bayes classifier tutorial: With Python Scikit-Learn*, *DataCamp*. Available at: <https://www.datacamp.com/tutorial/naive-bayes-scikit-learn> (Accessed: 01 March 2025).
* Comment *et al.* (2025) *ML: Naive Bayes scratch implementation using Python*, *GeeksforGeeks*. Available at: <https://www.geeksforgeeks.org/ml-naive-bayes-scratch-implementation-using-python/> (Accessed: 28 Feb. 2025).
* Comment *et al.* (2019) *ML: Implementation of KNN classifier using Sklearn*, *GeeksforGeeks*. Available at: <https://www.geeksforgeeks.org/ml-implementation-of-knn-classifier-using-sklearn/?ref=lbp> (Accessed: 02 March 2025).
* Comment, info, M., aswathisasidharan                                                                                                              *al.* (2025) *Support Vector Machine (SVM) algorithm*, *GeeksforGeeks*. Available at: <https://www.geeksforgeeks.org/support-vector-machine-algorithm/?ref=lbp> (Accessed: 03 March 2025).
* GeeksforGeeks (2024) *Text preprocessing in NLP*, *GeeksforGeeks*. Available at: <https://www.geeksforgeeks.org/text-preprocessing-for-nlp-tasks/> (Accessed: 20 Feb. 2025).
* GeeksforGeeks (2025) *Machine learning model evaluation*, *GeeksforGeeks*. Available at: <https://www.geeksforgeeks.org/machine-learning-model-evaluation/> (Accessed: 03 March 2025).