Module Code & Module Title

CU6051NI Artificial Intelligence

75% Individual Coursework

Submission: Final Submission

Academic Semester: Autumn Semester 2025

Credit: 15 credit semester long module

Student Name: Shubham Khoju Shrestha

London Met ID: 23050343

College ID: NP01CP4A230140

Assignment Due Date: 21/01/2026.

Assignment Submission Date: 21/01/2026

Submitted To: Er. Roshan Shrestha

GitHub Link	https://github.com/ShubhamKhojuShrestha/Email-Spam-Ham-Detection

Table of Contents

List of Figures

1) Introduction

1.1) Introduction of Topic and AI Concept Used

Artificial Intelligence (AI) is a fast-growing area of computer science that aims to develop intelligence systems that are able to execute those tasks that require human intelligence, including learning, problem-solving and decision making. A primary area in the development of Artificial Intelligence is that of Machine Learning (ML), whereby systems learns from experience and improve their performance on a continuous basis without the need to program them explicitly. Within the area of machine learning, the process of supervised learning is the primary method that is employed within classification tasks (Cole Stryker, Eda Kavlakoglu, 2025).

This project also relies heavily on a technique called Natural Language Processing (NLP), which is a sunset of AI and aims to enable machine to interpret and process human language. The reason NLP is employed in this project is because email data contains unstructured text and NLP algorithms need to be used in order tom ap the raw email data to a structured numeric representation. Text pre-processing like removal of unwanted characters, tokenization, removal of stopwords and TF-IDF transformation has been performed on the data in order to make machine learning algorithm ready (Charanarur, 2023).

1.2) Introduction of the Chosen Problem Domain/Topic

The chosen problem domain for this coursework is Spam/Ham Email Detection, a practical and widely researched application of AI within the area of NLP and machine learning. Spam emails are those that are unsolicited, either in the form of advertisement, phishing attacks, fraud or links that are malicious in nature. The rising amount of sophistication in the nature of spam email has raised issues for users as well as organization, thereby requiring the need for automated solutions.

The issue of spam mails can be represented as binary classification task where spam mails manually is rather inefficient and prone to errors with the continuous evolution of spam characteristics to evade conventional rule-based filtering. Through the use if AI based methods, the approach learns effectively from previous emails and adjusts to new spam attributes.

In this project, the design and development of the machine learning based anti-spam solution on the Enron Spam Dataset would be accomplished. The research activities performed in this project are based on the research activities performed in this coursework are based on the research undertaken in project. A functioning application that verifies the applicability of AI solution to the problem of unsolicited emails would be developed. The Algorithms used in this project are:

- Naïve Bayes
- Logistic Regression
- Support Vector Machine (SVM)

2) Background

In Coursework 1, we investigated how AI can solve the problem of spam email detection. The project focused on utilizing Natural Language Processing and supervised machine learning in order to classify if an email is spam or not. Through thorough review, it has been observed that detection of spam is a classic text-classification task, normally tackled by probabilistic models as well as discriminative models.

The research explored commonly used algorithms such as Naïve Bayes, Logistic Regression and Support Vector Machines. These all have extensive application in spam systems. Previous word showed that Naïve Bayes does well on text due to probabilistic nature, but logistic Regression and SVM do well when the features are high-dimensional, typical for technique such as TF-IDF. These insights were used to justify the algorithms of choice for this project.

The study also emphasized text pre-processing steps: tokenization, converting texts to lowercase, removal of stopwords and application of TF-IDF vectorization. The Enron Spam Dataset was identified as a suitable testbed due to its real-world relevance, size and balanced mix of spam and legitimate emails. Overall, Coursework 1 has set that theoretical basis and guided the design decision that have been developed in the following project implementation phase.

2.1) Research on issue:

- Naïve Bayes for Spam Detection
  Naïve Bayes classifier has been defined as the most popular and efficient choice used as a text-based spam classification baseline. The efficiency of Naïve Bayes with TF-IDF has ensured that a high baseline accuracy in email filtering, making it a critical component for integration (Perumal, 2025).

- Superiority of Advanced Linear Models
  The comparative analysis reveals that linear models naturally outperform the Naïve Bayes classification as a standard. Support Vector Machines, particularly Support Vector Classifier are known for their functionality of determining the best separating hyperplane in a feature space with high accuracy in email classification (Budiman, 2024).

- Integration of Topic Modelling and ML Algorithms
  A 2024 research article examined the combination of Latent Dirichlet Allocation (LDA) topic modelling with Logistic Regression, SVM and Naïve Bayes classifiers. Their finding suggested that Logistic Regression might be capable of performing better than traditional models, emphasizing the role of features extraction technique for text classification (Borotic, 2024).

2.2) Advantage, Drawbacks and issues

Advantages:

- Machine Learning Methods are capable of adapting to changes in spam messages.
- NLF features like TF-IDF assists in identifying unique features of text.
- Logistic Regression and SVM may have advantage over rule-based filters, as recent comparative studies have shown.

Drawbacks:

- The models might need careful tuning and pre-processing for preventing overfitting.
- Techniques involving deep learning may require more computation.
- The spam content changes very quickly, which may reduce accuracy if models.

Issues:

- Concept Drift: Adaptation to new spamming methods necessitates recurrent model retains.
- Class imbalance: The class imbalance problem arises if there are more ham massage than spam messages. To address these problems, a strategy for supervised learning.

2.3) Dataset Information and Background

The Dataset used in this project is the Enron Spam Dataset and it is a very large and prominent collection of actual email message. It was initially made available as a result of the Enron legal proceedings. It contains 33,716 emails with 17,171 classified as spam and 16,545 categorized as ham emails. It is very well-balanced and thus acts as an optimum dataset for binary classification problems. It contains two prominent columns columns: “text”, which represent the raw message with no pre-processing and “Spam/Ham”, which identifies whether an email is spam or ham. It has been widely cited as an optimal benchmark because it contains very realistic emails, a large number and they are properly labelled. Various recent research papers have made use of it and its variants for developing and testing NLP and machine learning algorithms for spam classification.

Link to “enron_spam_data”:

https://huggingface.co/datasets/SetFit/enron_spam/tree/main

3) Solution

3.1) Explanation of the Solution

The solution for this project leverages machine learning classification algorithms to classify whether emails are spam or ham based on the text content in the emails. The dataset used in ding project was sourced from the Enron Spam dataset that includes attributes such as Subject, Message and Spam/Ham classification.

During the data exploration and pre-processing stage, missing data in the Subject and Message attribute has been addressed by treating the missing data as empty string and unnecessary attributes of features are removed. The Subject and Message of the emails are then combined into a single feature, which has undergone processing, such as the conversion of all character to lowercase, the removal of all punctuation and special characters and the removal of stopwords. The process ensured that the models are trained on the actual meaningful text.

The given dataset was split into training and testing sets to assess the performance of the models on unknown emails. The project was designed with three machine learning algorithms: Naïve Bayes, Logistic Regression and Support Vector Machine (SVM).

Naïve Bayes was used as the baseline model because it is effective and simple technique and it relies on the probability of words in order to classify an email as spam. Logistic Regression was utilized to express the likelihood of an email being spam in terms of the linear combination of the TF-IDF features. SVM with a linear kernel was used to address high dimensional sparse data created from TF-IDF to address relationship between words and classification into spam.

The performance of each of the models is measured using the typical metrics of the classification. The measures include accuracy, F1. All the metrics has been used to determine the most effective way of spam detection. The approach will thereby provide a scalable and efficient wat of spam detection.

3.2) Explanation of Algorithms used

- Naïve Bayes

  Naïve Bayes is probabilistic classification technique that uses Bayes theorem. Its hypothesis that a message's words are mutually independent, estimating the probability that a message is form either the spam class or ham class. The fact that Naïve Bayes is easy to train, has a fast training speed and is highly efficient for text-classified tasks has made this technique a popular choice for the classification of spam messages (SCIKIT, 2024).

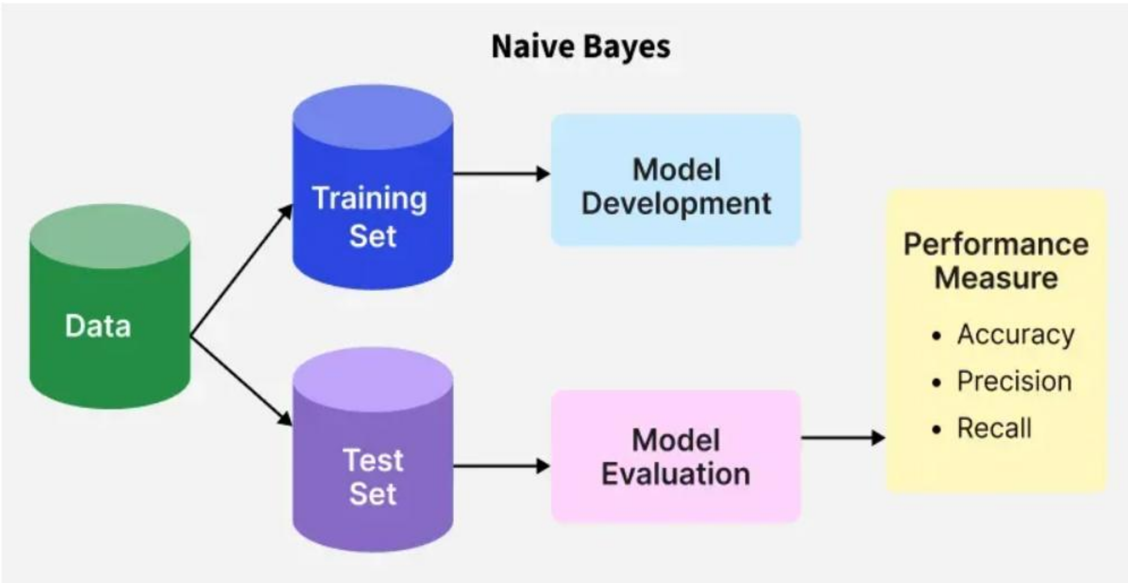


Figure 1: Naive Bayes

- Logistic Regression

  Logistic Regression is a supervised learning algorithm used for predicting the probability of a particular email being marked as spam. It is efficient on high-dimensional spaces, such as the TF-IDF space and gives reliable results. This algorithm helps find a boundary line that distinguishes spam messages from ham messages based on certain features that are identified. (Sarkar, 2025)

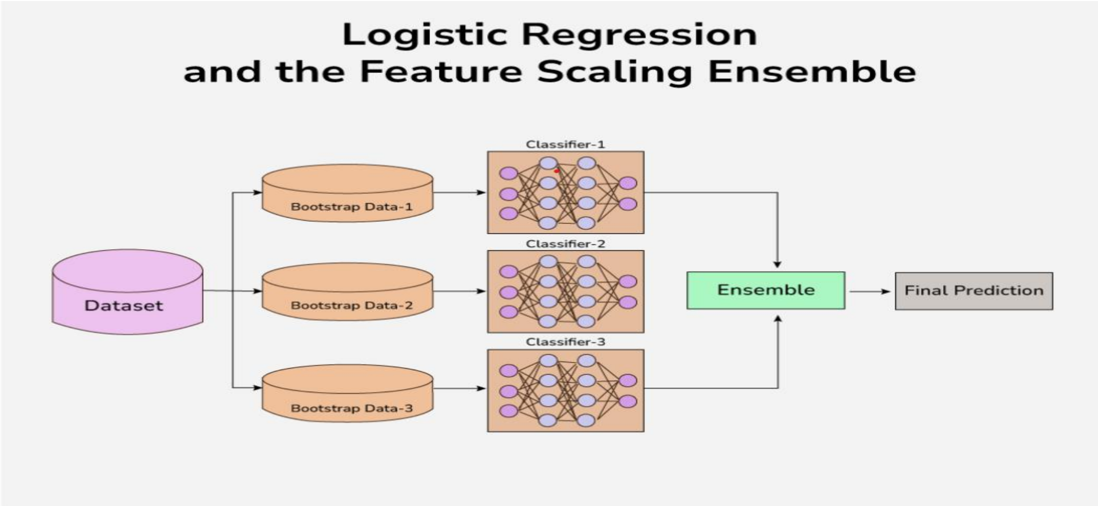


Figure 2: Logistic Regression

- Support Vector Machine (SVM)

  Support Vector Machine (SVM) is a classification technique with a margin that finds the best hyperplane that distinguishes spam emails form ham emails. SVM is generally useful when dealing with high-dimensional text data. The reason that the SVM technique is frequently used in text classification system, including spam filters, is because of its high degree of generalization (SCIKIT, 2024).

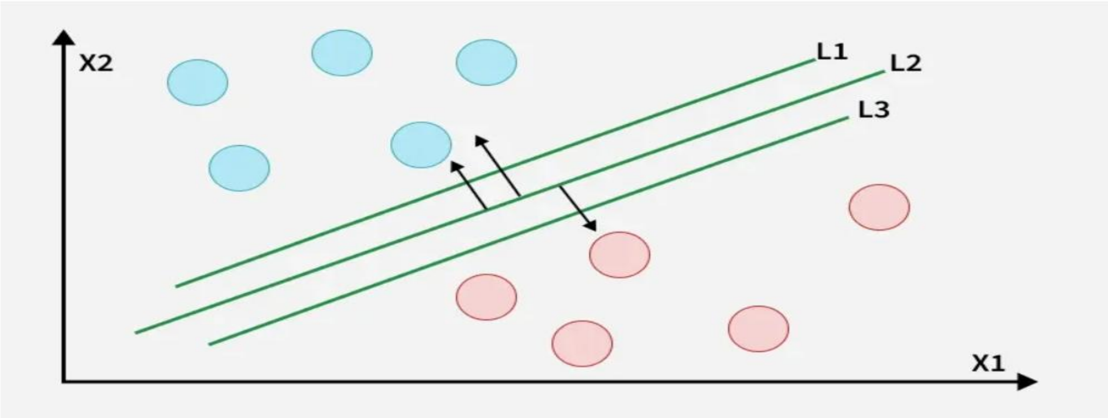


Figure 3: Support Vector Machine (SVM)

3.3) Pseudocode of the Solution

3.3.1) Pseudocode of the Naïve Bayes

START
INPUT training text features and labels
CALCULATE prior probabilities for spam and ham classes
CALCULATE likelihood of words given each class
FOR each email in test dataset
    COMPUTE posterior probability for spam class
    COMPUTE posterior probability for ham class
    ASSIGN class with highest probability
END FOR
OUTPUT predicted labels
END

3.3.2) Pseudocode of the Logistic Regression

START
INPUT training text features and labels
INITIALIZE model parameters
APPLY sigmoid function to estimate probability of spam
OPTIMIZE parameters using gradient descent
FOR each email in test dataset
    CALCULATE probability of spam
    IF probability >= threshold (0.5)
        CLASSIFY as spam
    ELSE
        CLASSIFY as ham
    END FOR
    OUTPUT predicted labels
END

3.3.3) Pseudocode of Support Vector Machine (SVM)

START

INPUT training text features and labels.

MAP text data into high-dimensional features space

FIND optimal hyperplane that maximizes margin between classes

FOR each email in test dataset

DETERMINE position relative to hyperplane

ASSIGN class based on decision boundary

END FOR

OUTPUT predicted labels

END

3.3.4) Pseudocode of System

START

IMPORT required libraries

LOAD Enron spam dataset

HANDLE missing values

REPLACE missing Subject and Message with empty strings

REMOVE rows with missing Spam/Ham labels

COMBINE Subject and Message into a single text field

CONVERT Spam/Ham labels to binary values

Spam Into 1

Ham into 0

PREPROCES text data

CONVERT text to lowercase

REMOVE numbers and special characters

SPLIT dataset into training set and testing set

APPLY TF-IDF vectorization to transform text into numerical features

TRAIN Naïve Bayes model
TRAIN Logistic Regression model
TRAIN Support Vector Machine model

PREDICT spam/ham labels using testing data for each model

EVALUATE models using Accuracy and F1-score

END

3.4) Flowchart

3.4.1) Flowchart of the Naïve Bayes

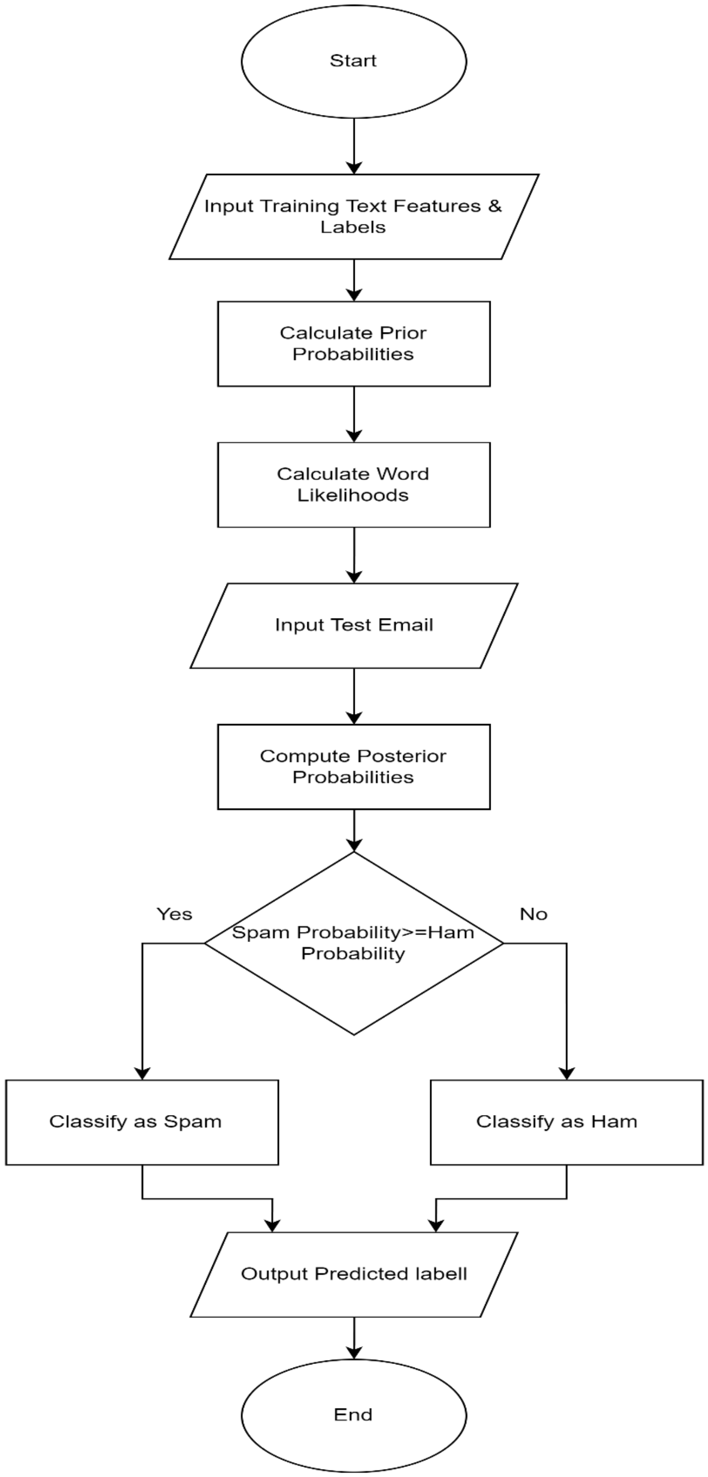


Figure 4: Flowchart of Naive Bayes

3.4.2) Flowchart of the Logistic Regression

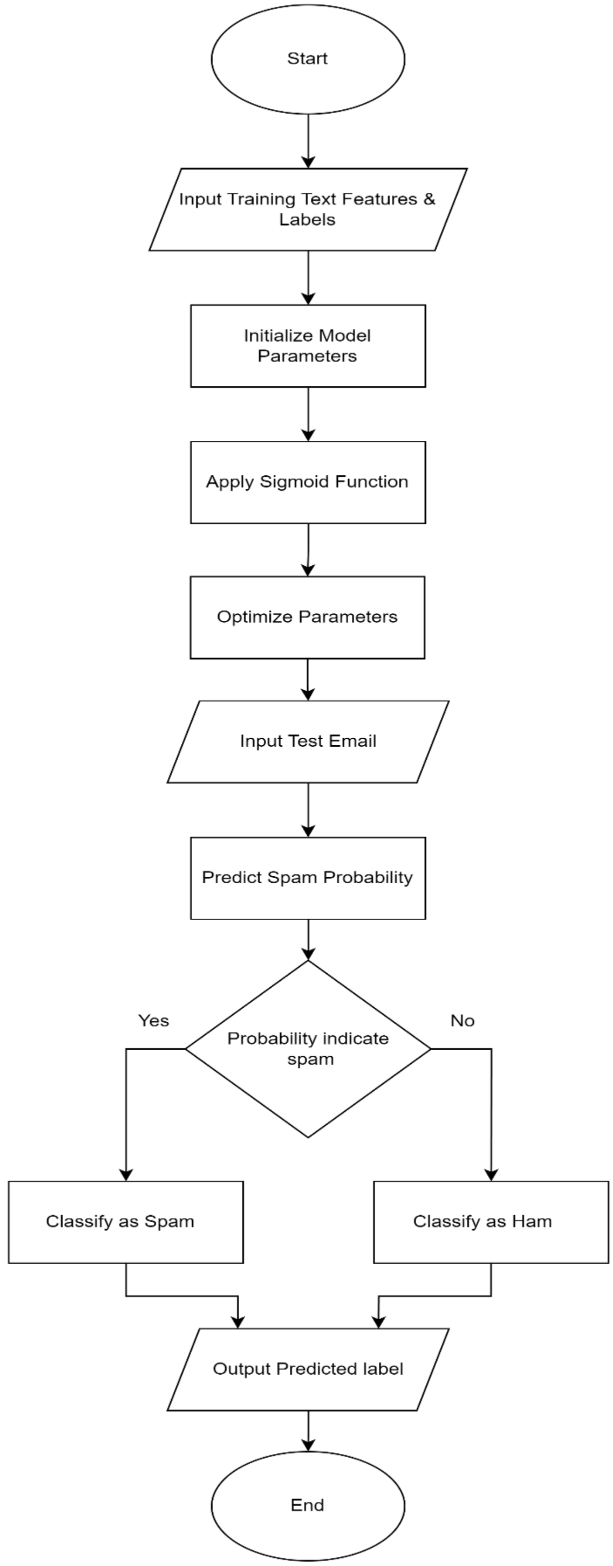


Figure 5: Flowchart of Logistic Regression

3.4.3) Flowchart of Support Vector Machine (SVC)

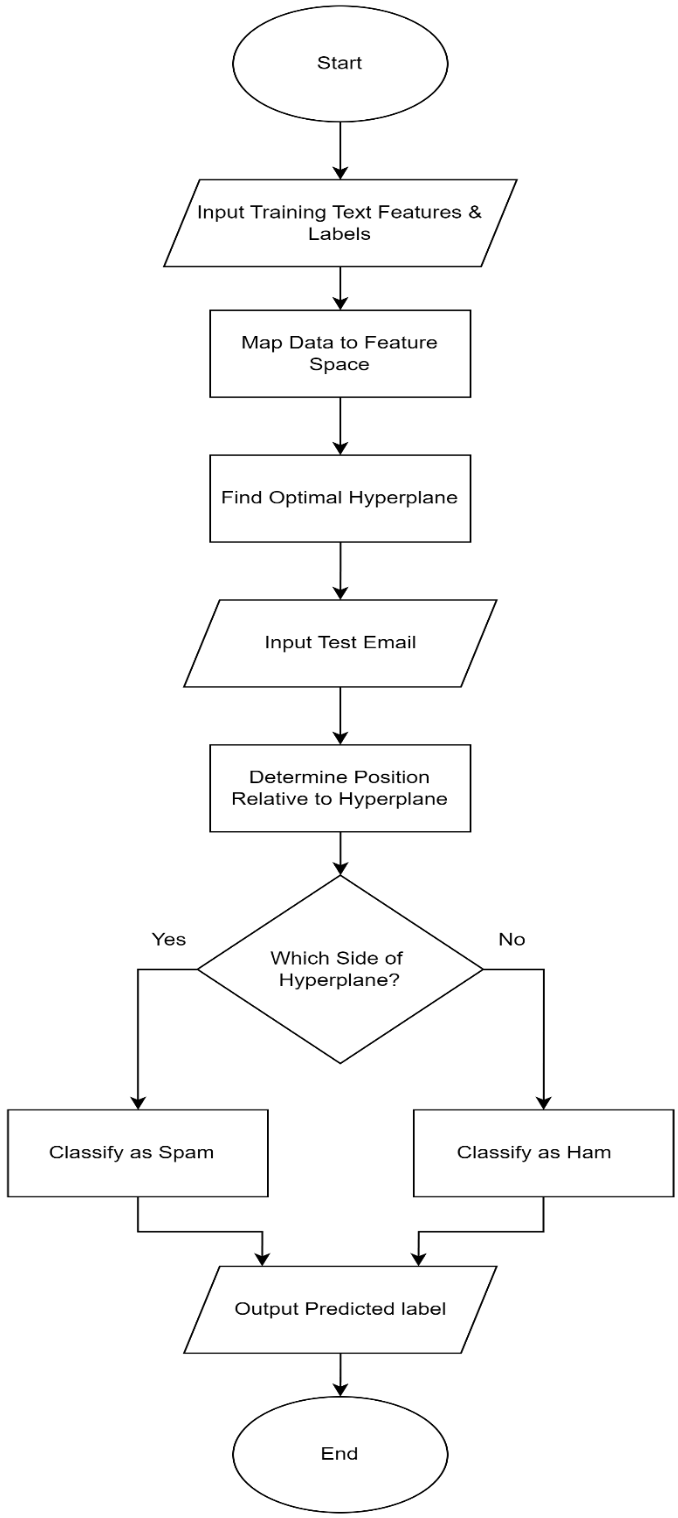


Figure 6: Flowchart of SVM

3.4.4) Flowchart of System

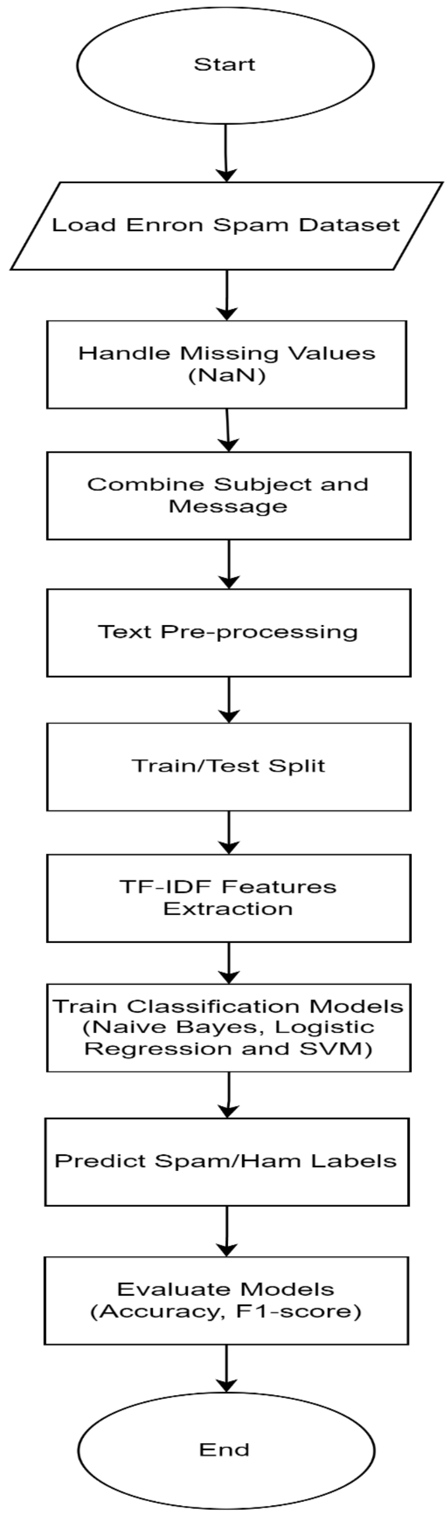


Figure 7: Flowchart of System

3.5) Explanation of Development Process

The whole process involved in the development of the project has been executed using Python programming language, thanks to its strong support for the operational of data analysis, text processing and machine learning. The coding process has been done using Jupiter Notebook environment.

The workflow that was used to develop the model consisted of a systematic process that involved data preparation, pre-processing, feature extraction, model implementation and testing. The following Python libraries were utilized the development process:

- Pandas: For data loading, cleaning and manipulation of the dataset.
- Scikit-learn: For implementing machine learning models, feature extraction (TF-IDF) and evaluation metrics.
- Matplotlib: For creating visualization to analyse data distribution, model performance and prediction results.
- re (Regular Expressions): For text pre-processing and cleaning

This Structured methodology made it possible to develop an effective and explainable spam filter system that makes precise prediction while keeping the process transparent.

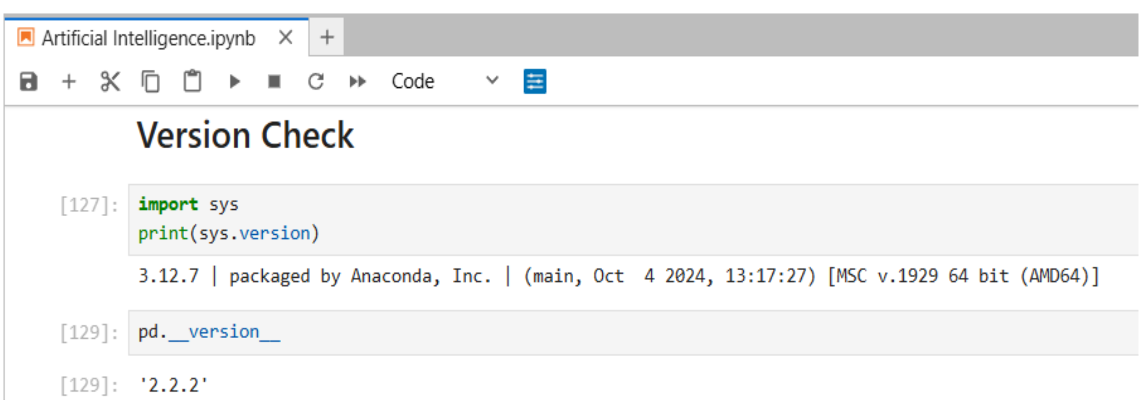


Figure 8: Version Check

3.5.1) Loading Dataset

Artificial Intelligence.ipynb

```
[132]: import pandas as pd
```

Load Dataset

```
[17]: df = pd.read_csv('enron_spam_data.csv')
      df
```

[17]:

	Message ID	Subject	Message	Spam/Ham	Date
0	0	christmas tree farm pictures	NaN	ham	1999-12-10
1	1	vastar resources , inc .	gary , production from the high island larger ...	ham	1999-12-13
2	2	calpine daily gas nomination	- calpine daily gas nomination 1 . doc	ham	1999-12-14
3	3	re : issue	fyi - see note below - already done .\nstella\...	ham	1999-12-14
4	4	meter 7268 nov allocation	fyi .\n- -...	ham	1999-12-14
...	...	...	...	...	...
33711	33711	= ? iso - 8859 - 1 ? q ? good _ news _ c = eda...	hello , welcome to gigapharm onlinne shop .\np...	spam	2005-07-29
33712	33712	all prescript medicines are on special . to be...	i got it earlier than expected and it was wrap...	spam	2005-07-29
33713	33713	the next generation online pharmacy .	are you ready to rock on ? let the man in you ...	spam	2005-07-30
33714	33714	bloow in 5 - 10 times the time	learn how to last 5 - 10 times longer in\nbed ...	spam	2005-07-30
33715	33715	dear sir , i am interested in it	hi :)\ndo you need some softwares ? i can giv...	spam	2005-07-31

33716 rows × 5 columns

Figure 9: Loading Dataset

The dataset used in this case is the Enron Spam Dataset, and this dataset is imported into the system for analysis through the use of the Pandas Library. This allows the dataset to be inspected and reviewed in an organized table layout.

Artificial Intelligence.ipynb

```
[128]: import matplotlib.pyplot as plt
```

```
[130]: df["Spam/Ham"].value_counts().plot(kind="bar")
       plt.title("Spam vs Ham Distribution")
       plt.xlabel("Email Type")
       plt.ylabel("Count")
       plt.show()
```

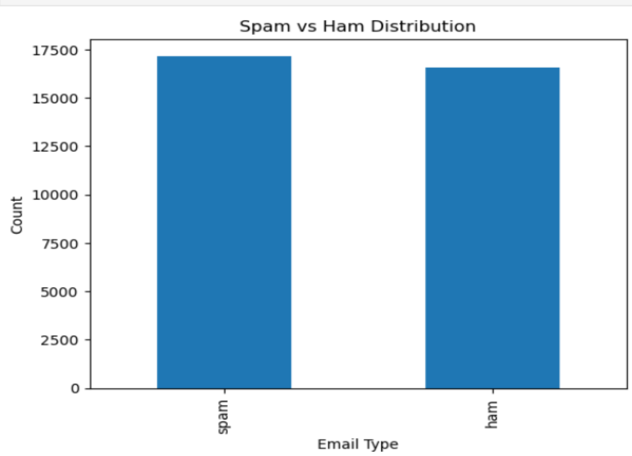
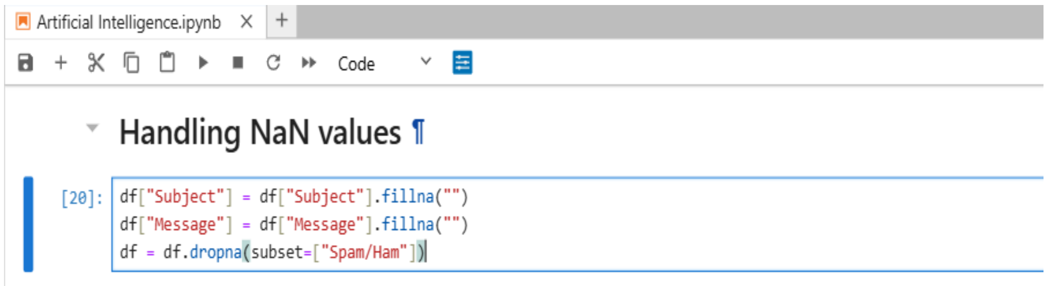


Figure 10: Bar graph of Spam and Ham distribution

The bar graph is used to represent the distribution of spam and ham message within the data. This will aid in analysing the class distribution by knowing the distribution of spam and ham message which will be essential for determining the accuracy level of the developed models.

3.5.2) Handling NaN values



Artificial Intelligence.ipynb

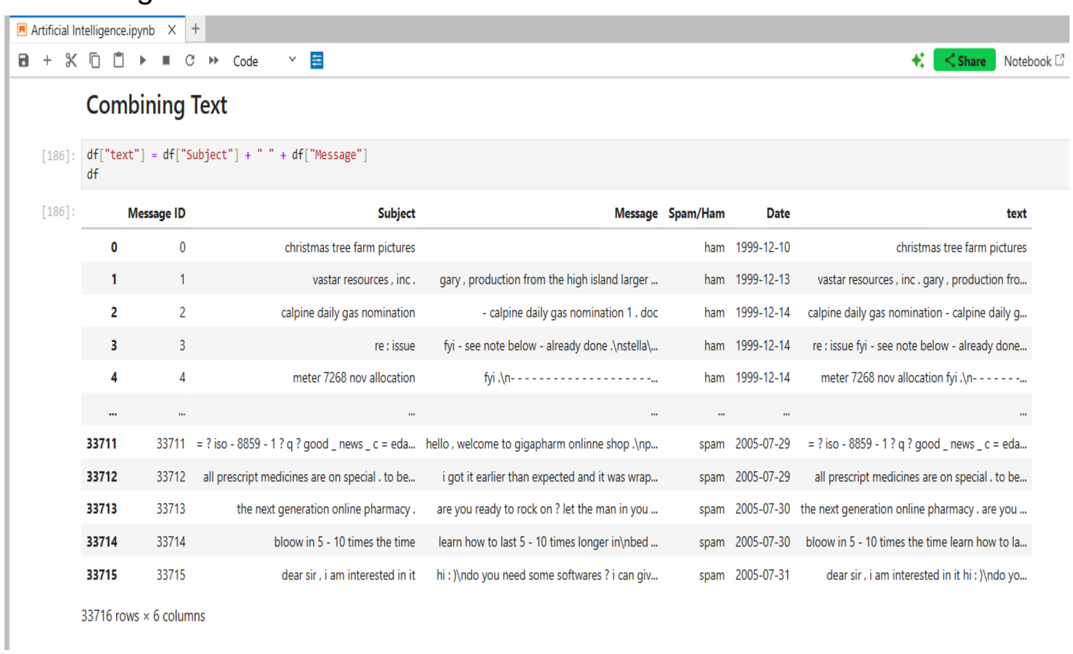
Handling NaN values

```
[20]: df["Subject"] = df["Subject"].fillna("")
df["Message"] = df["Message"].fillna("")
df = df.dropna(subset=["Spam/Ham"])
```

Figure 11: Handling NaN values

Missing values in the email subject and message column are treated by replacing them with empty string. Data with missing values for spam or ham columns is filtered out since data must be present in order for supervised learning processes to take place.

3.5.3) Combining Text



Artificial Intelligence.ipynb

Combining Text

```
[186]: df["text"] = df["Subject"] + " " + df["Message"]
df
```

[186]:

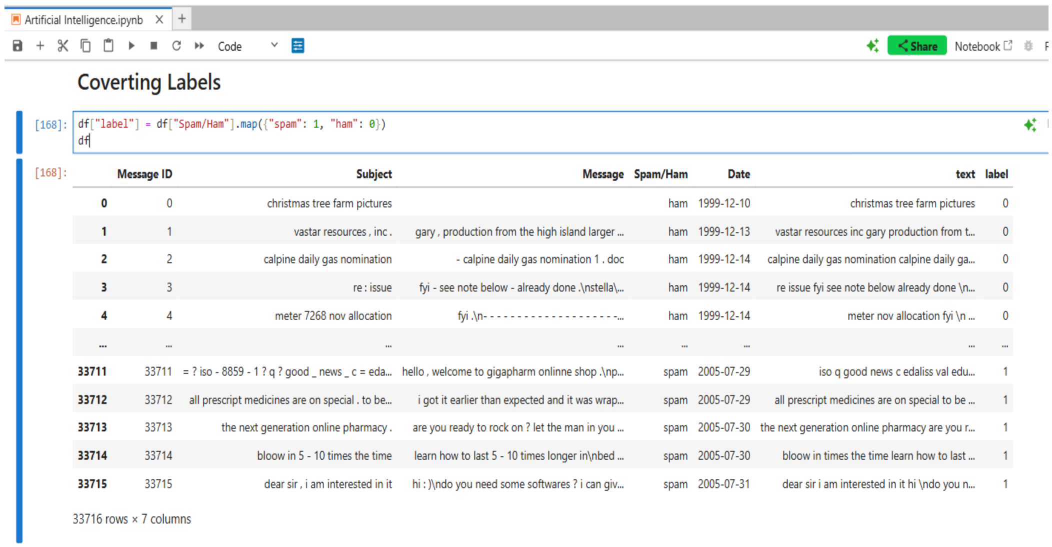
	Message ID	Subject	Message	Spam/Ham	Date	text
0	0	christmas tree farm pictures		ham	1999-12-10	christmas tree farm pictures
1	1	vastar resources , inc .	gary , production from the high island larger ...	ham	1999-12-13	vastar resources , inc . gary , production fro...
2	2	calpine daily gas nomination	- calpine daily gas nomination 1 . doc	ham	1999-12-14	calpine daily gas nomination - calpine daily g...
3	3	re : issue	fyi - see note below - already done .\nstella\...	ham	1999-12-14	re : issue fyi - see note below - already done...
4	4	meter 7268 nov allocation	fyi .\n- - - - - - - - - - - - - - - - - - -...	ham	1999-12-14	meter 7268 nov allocation fyi .\n- - - - - - -...
...	...	...	...	...	...	...
33711	33711	= ? iso - 8859 - 1 ? q ? good _ news _ c = eda...	hello , welcome to gigapharm onlinne shop .\np...	spam	2005-07-29	= ? iso - 8859 - 1 ? q ? good _ news _ c = eda...
33712	33712	all prescript medicines are on special . to be...	i got it earlier than expected and it was wrap...	spam	2005-07-29	all prescript medicines are on special . to be...
33713	33713	the next generation online pharmacy .	are you ready to rock on ? let the man in you ...	spam	2005-07-30	the next generation online pharmacy . are you ...
33714	33714	bloow in 5 - 10 times the time	learn how to last 5 - 10 times longer in\nbed ...	spam	2005-07-30	bloow in 5 - 10 times the time learn how to la...
33715	33715	dear sir , i am interested in it	hi :)\ndo you need some softwares ? i can giv...	spam	2005-07-31	dear sir , i am interested in it hi :)\ndo yo...

33716 rows × 6 columns

Figure 12: Combining Text

The subject and content of each email message are merged into a single text value. The results are a unified representation of an email message, permitting machine learning models to consider all relevant text in an email message when performing feature extraction and classification.

3.5.4) Converting Labels



Coverting Labels

```
[168]: df["label"] = df["Spam/Ham"].map({"spam": 1, "ham": 0})
df
```

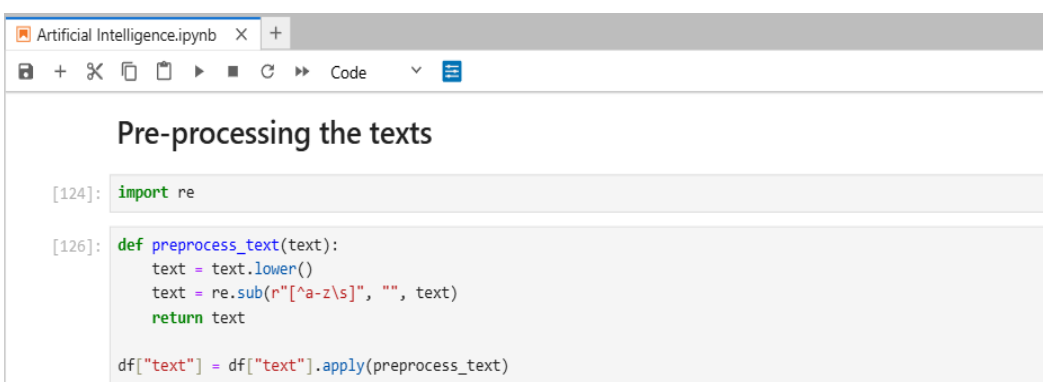
	Message ID	Subject	Message	Spam/Ham	Date	text	label
0	0	christmas tree farm pictures		ham	1999-12-10	christmas tree farm pictures	0
1	1	vastar resources , inc .	gary , production from the high island larger ...	ham	1999-12-13	vastar resources inc gary production from t...	0
2	2	calpine daily gas nomination	- calpine daily gas nomination 1 . doc	ham	1999-12-14	calpine daily gas nomination calpine daily ga...	0
3	3	re : issue	fyi - see note below - already done .\nstella\...	ham	1999-12-14	re issue fyi see note below already done \n...	0
4	4	meter 7268 nov allocation	fyi .\n- - - - - - - - - - - - - - - - - - -...	ham	1999-12-14	meter nov allocation fyi \n ...	0
...	...	...	...	...	...	...	...
33711	33711	= ? iso - 8859 - 1 ? q ? good _ news _ c = eda...	hello , welcome to gigapharm onlinne shop .\np...	spam	2005-07-29	iso q good news c edaliss val edu...	1
33712	33712	all prescript medicines are on special . to be...	i got it earlier than expected and it was wrap...	spam	2005-07-29	all prescript medicines are on special to be ...	1
33713	33713	the next generation online pharmacy .	are you ready to rock on ? let the man in you ...	spam	2005-07-30	the next generation online pharmacy are you r...	1
33714	33714	bloow in 5 - 10 times the time	learn how to last 5 - 10 times longer in\nbed ...	spam	2005-07-30	bloow in times the time learn how to last ...	1
33715	33715	dear sir , i am interested in it	hi :)\ndo you need some softwares ? i can giv...	spam	2005-07-31	dear sir i am interested in it hi \ndo you n...	1

33716 rows × 7 columns

Figure 13: Converting Labels

The spam and ham labels are then encoded into numeric form, where spam is given the value 1 and ham is given the value 0. Coding into numeric from is needed in machine learning algorithms because the computation in machine learning requires numeric input.

3.5.5) Pre-processing the texts



Pre-processing the texts

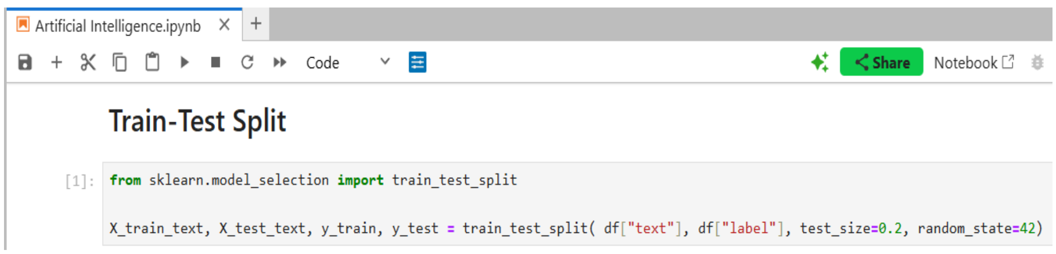
```
[124]: import re
```

```
[126]: def preprocess_text(text):
    text = text.lower()
    text = re.sub(r"[^a-z\s]", "", text)
    return text

df["text"] = df["text"].apply(preprocess_text)
```

Figure 14: Pre-processing the texts

Text pre-processing is performed in order to clean and standardize the content of the emails. A regular expression (re) library is used to separate and eliminate unwanted characters such as punctuation, numbers and special characters. All the text in the mails is converted to lowercase. This is done in order to standardize the content by ensuring all the text in the mails is in lowercase.

3.5.6) Train-Test Split

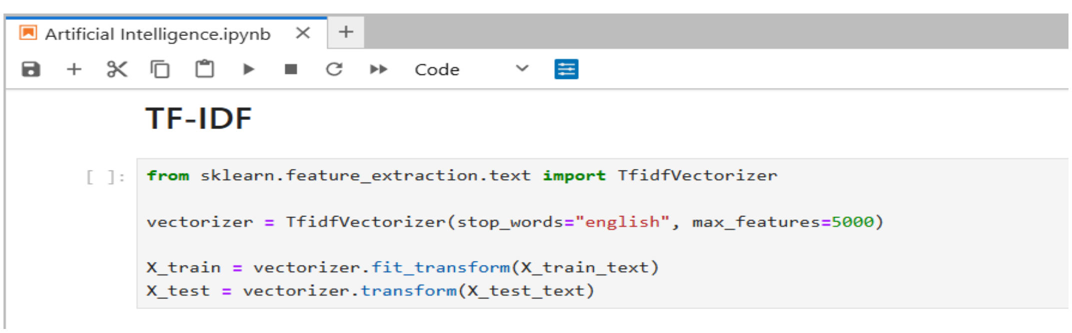



```
from sklearn.model_selection import train_test_split

X_train_text, X_test_text, y_train, y_test = train_test_split( df["text"], df["label"], test_size=0.2, random_state=42)
```

Figure 15: Train-Test Split

The function train_test_split from the module skleran.model_selection is used to split the dataset into training and testing sets. This will enable the training of the machine learning algorithms on the dataset and the evaluation of the results on unseen instances. For the project the dataset is split into 80% training set and 20% testing set allow for proper evaluation of the accuracy of the model.

3.5.7) TF-IDF

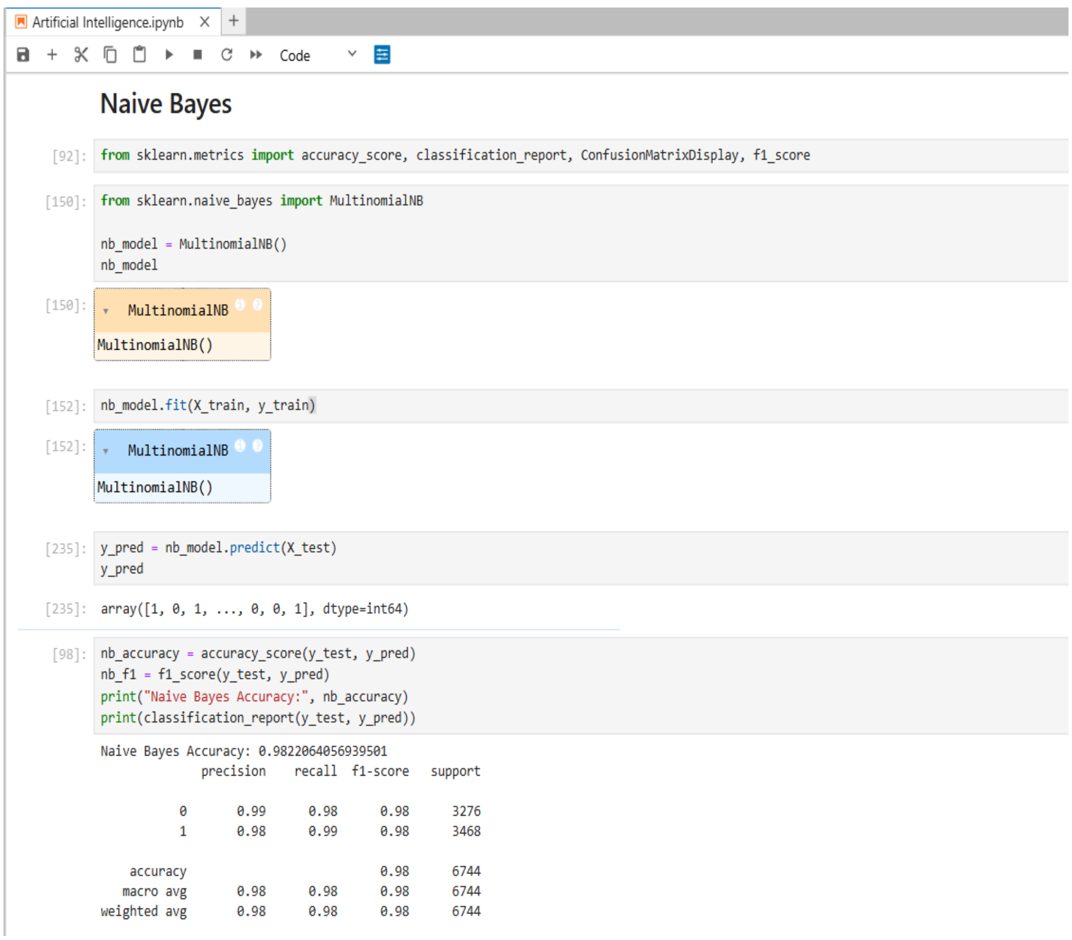



```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop_words="english", max_features=5000)

X_train = vectorizer.fit_transform(X_train_text)
X_test = vectorizer.transform(X_test_text)
```

Figure 16: TF-IDF

The TfidfVectorizer from sklearn.fearures_extraction.text module converts the text to number that the machine learning algorithm can operate on. It applies TF-IDF, giving works a measure of importance based on how often the word appears in that one message versus the entire set of messages. It removes stop words based on the English stopwords, retaining the top 5000 features for faster processing. The vectorizer is fitted only on the training data to avoid data leakage and both training and testing texts are transformed into numerical feature matrices.

3.5.8) Naïve Bayes Algorithm



Artificial Intelligence.ipynb

Code

Naive Bayes

```
[92]: from sklearn.metrics import accuracy_score, classification_report, ConfusionMatrixDisplay, f1_score

[150]: from sklearn.naive_bayes import MultinomialNB

nb_model = MultinomialNB()
nb_model

[150]: MultinomialNB
MultinomialNB()

[152]: nb_model.fit(X_train, y_train)

[152]: MultinomialNB
MultinomialNB()

[235]: y_pred = nb_model.predict(X_test)
y_pred

[235]: array([1, 0, 1, ..., 0, 0, 1], dtype=int64)

[98]: nb_accuracy = accuracy_score(y_test, y_pred)
nb_f1 = f1_score(y_test, y_pred)
print("Naive Bayes Accuracy:", nb_accuracy)
print(classification_report(y_test, y_pred))

Naive Bayes Accuracy: 0.9822064056939501
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      3276
           1       0.98      0.99      0.98      3468

    accuracy                           0.98      6744
   macro avg       0.98      0.98      0.98      6744
weighted avg       0.98      0.98      0.98      6744
```

Figure 17: Naive Bayes Classification

The machine learning process starts by applying a Multinomial Naïve Bayes Classifier in classifying emails int spam and ham messages. The chosen model applies due to its ability to handle word frequency, such as TF-IDF vectors derived earlier.

An instance of the NB class is created through the function call, Multinomial(), followed by the training of the model on the training set that consists of the features in the variable X_train and the response variable for the test set, which is denoted by the variable X_test.

The prediction value are compared to the actual labels (y_test) based on the evaluation metrics from the sklearns.metrics module. The accuracy is used to check the proportion of correctly predicted emails and the F1 score provides a combination of precision and recall to give a clear indication of model performance on an imbalanced dataset. The classification report provides information for each class (spam and ham).

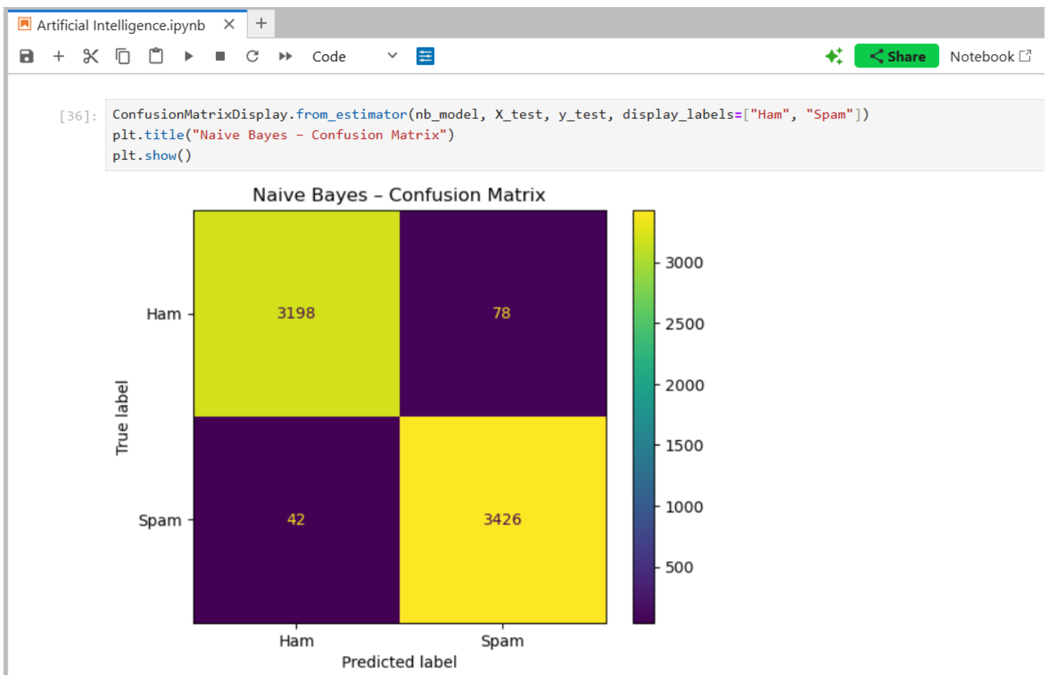


Figure 18: Confusion Matrix of Naive Bayes

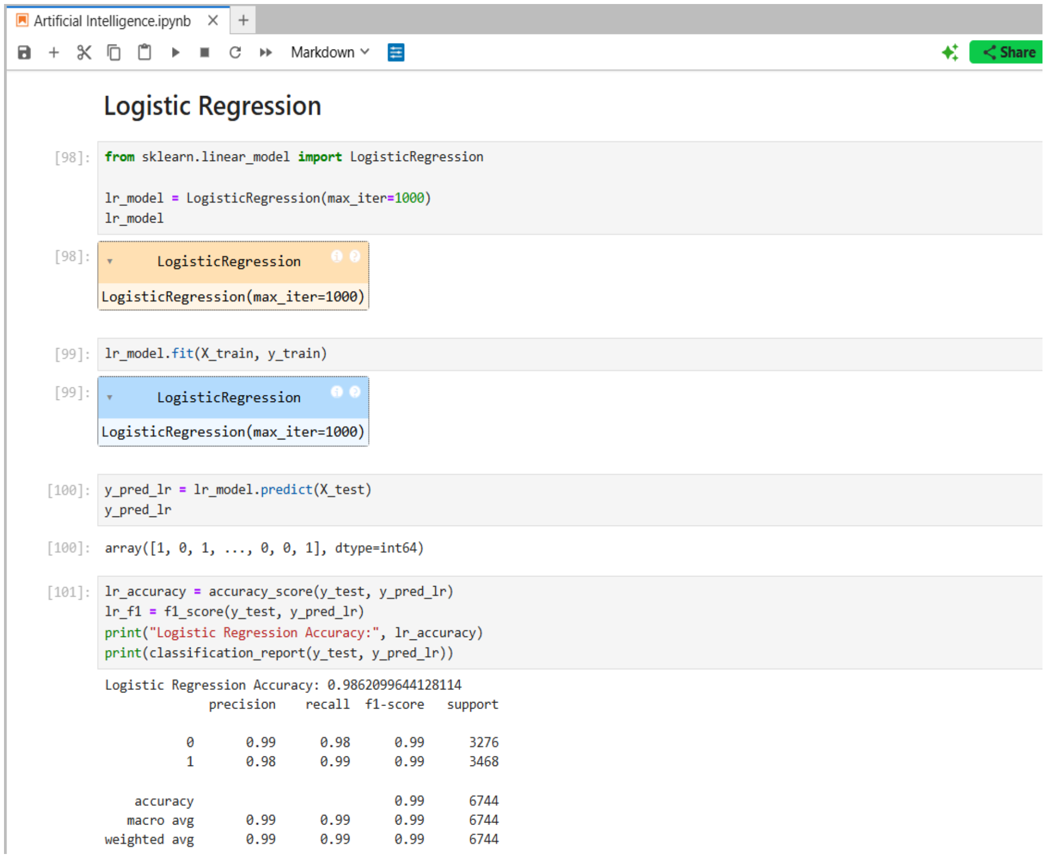
The Naïve Bayes model's email classification performance is visualized using a confusion matrix. ConfusionMatrixDisplay.from_estimator is a sklearn function.Metrics uses the model's prediction for the test set to automatically calculate and plot the matrix.

A clear picture of the model's advantage and disadvantage is provided by the matrix, which displays the number of true positive (spam correctly predicted), true negatives (ham correctly predicted), false positive and false negatives. For the purpose of assessing and enhancing classification performance, visualizing the confusion matrix aids in determining the kinds of mistakes the model makes most frequently.

Figure 19: Accuracy and F1-score Comparison of Naive Bayes

The accuracy and F1 score, two important evaluation metrics are used to create a bar chart that shows the Naïve Bayes models' performance. The model's overall correctness and precision-recall balance can be quickly compared by looking at the heights of the bars, which represents the numerical values of each metric. For clarity he actual metric value are shown in the centre of each bar. This graphic aid makes it simple to assess how well the model classifies emails as spam or ham.

3.5.9) Logistic Regression Algorithm


Artificial Intelligence.ipynb

Markdown

Share

Logistic Regression

```
[98]: from sklearn.linear_model import LogisticRegression

      lr_model = LogisticRegression(max_iter=1000)
      lr_model

[98]: LogisticRegression
      LogisticRegression(max_iter=1000)

[99]: lr_model.fit(X_train, y_train)

[99]: LogisticRegression
      LogisticRegression(max_iter=1000)

[100]: y_pred_lr = lr_model.predict(X_test)
       y_pred_lr

[100]: array([1, 0, 1, ..., 0, 0, 1], dtype=int64)

[101]: lr_accuracy = accuracy_score(y_test, y_pred_lr)
       lr_f1 = f1_score(y_test, y_pred_lr)
       print("Logistic Regression Accuracy:", lr_accuracy)
       print(classification_report(y_test, y_pred_lr))

Logistic Regression Accuracy: 0.9862099644128114
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      3276
           1       0.98      0.99      0.99      3468

    accuracy                           0.99      6744
   macro avg       0.99      0.99      0.99      6744
weighted avg       0.99      0.99      0.99      6744
```

Figure 20: Logistic Regression Classification

Email are categorized as spam or ham using Logistic Regression model. A linear model called logistic regression uses the TF-IDF features to calculate the likelihood that an email will belong to each class. To guarantee convergence during training, the model is instantiated with maximum iteration limit.

The model is used to predict labels for the unseen test set (X_test) after being trained on the training set (X_train and y_train). Accuracy and F1-score, which gauge overall correctness and the ration of precision to recall respectively, are used to assess its performance. A thorough evaluation of model's efficacy is made possible by the classification report, which offers a thorough breakdown of performance for both the spam and ham classes.

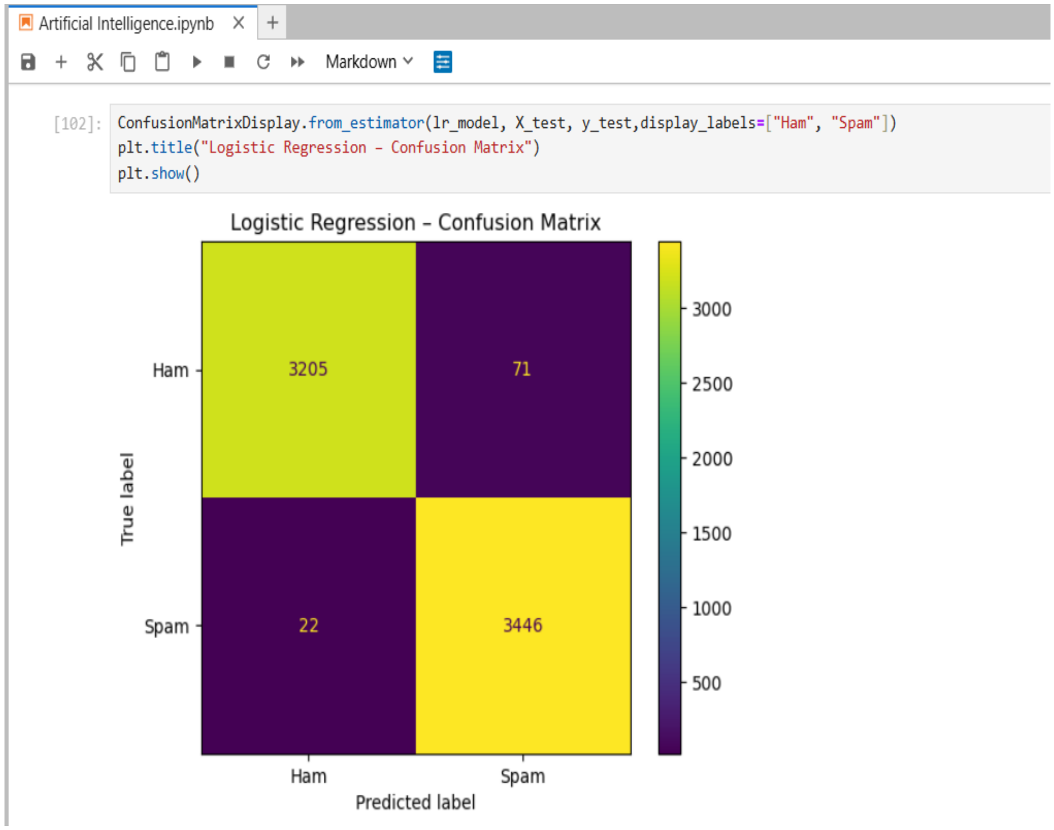


Figure 21: Confusion Matrix of Logistic Regression

To see how the Logistic Regression model performs in email classification, a confusion matrix is created. ConfusionMatrixDisplay is used. The matrix from_estimator displays the quantity of true positive, true negative false negative and false negatives, the model's strengths and weakness in differentiating between spam and ham emails are clearly understood thanks to visual representation which also makes it simple to recognize the kinds of mistakes the model makes.

```
plt.bar(["Accuracy", "F1-score"], [lr_accuracy, lr_f1])
plt.ylim(0, 1)
plt.title("Logistic Regression Performance")
plt.ylabel("Score")
plt.text(0, lr_accuracy / 2, f"{lr_accuracy:.2f}", ha="center", va="center", color="white")
plt.text(1, lr_f1 / 2, f"{lr_f1:.2f}", ha="center", va="center", color="white")
plt.show()
```

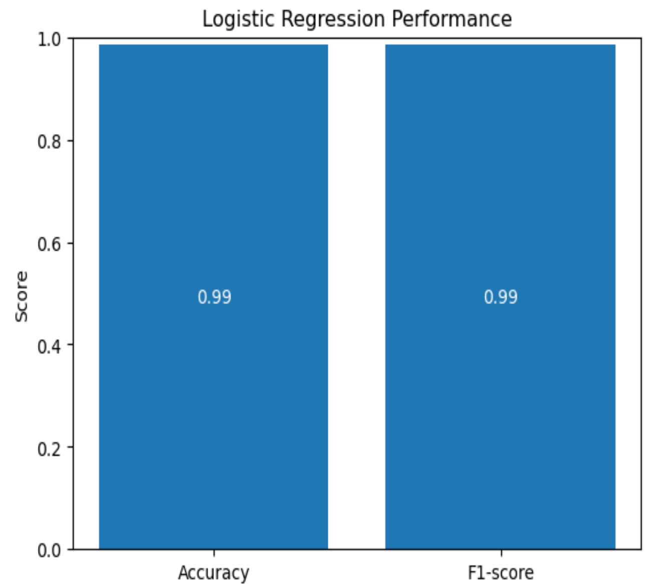
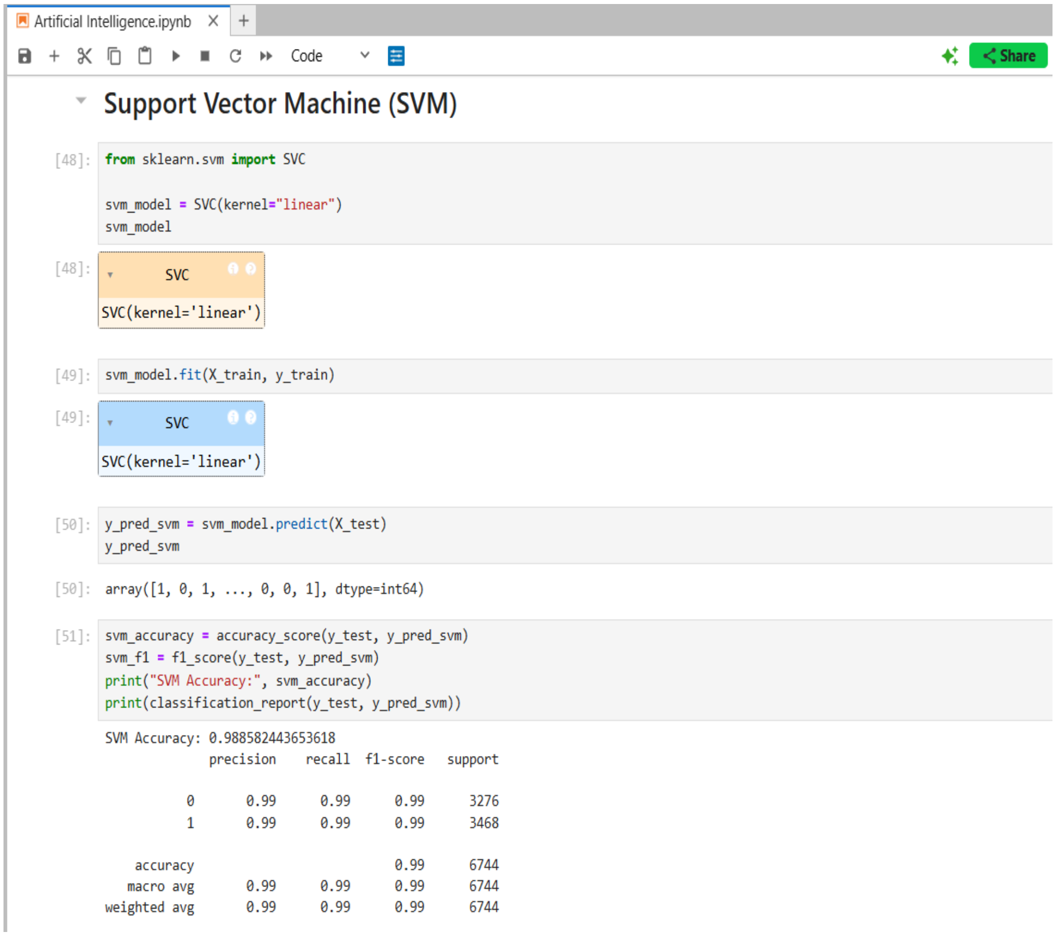


Figure 22: Accuracy and F1-score Comparison of Logistic Regression

The key metrics Accuracy and F1-score are used to visualize the Logistic Regression model's performance in a bar chart. Each metrics numerical values are represented by the heights of the bars, for clarity the values are shown in the centre of the bars. The model's ability to accurately classify emails as spam or ham is summarized in this visualization, which is easy to understand and demonstrates both overall correctness and overall correctness and a balance between precision and recall.

3.5.10) Support Vector Machine (SVM) Algorithm



Support Vector Machine (SVM)

```
[48]: from sklearn.svm import SVC

svm_model = SVC(kernel="linear")
svm_model
```

```
[48]: SVC
SVC(kernel='linear')
```

```
[49]: svm_model.fit(X_train, y_train)
```

```
[49]: SVC
SVC(kernel='linear')
```

```
[50]: y_pred_svm = svm_model.predict(X_test)
y_pred_svm
```

```
[50]: array([1, 0, 1, ..., 0, 0, 1], dtype=int64)
```

```
[51]: svm_accuracy = accuracy_score(y_test, y_pred_svm)
svm_f1 = f1_score(y_test, y_pred_svm)
print("SVM Accuracy:", svm_accuracy)
print(classification_report(y_test, y_pred_svm))
```

```
SVM Accuracy: 0.988582443653618
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      3276
           1       0.99      0.99      0.99      3468

    accuracy                           0.99      6744
   macro avg       0.99      0.99      0.99      6744
weighted avg       0.99      0.99      0.99      6744
```

Figure 23: Support Vector Machine (SVM) Classification

To categorize emails as spam or ham, the Support Vector Machine (SVM) model is applied using a linear kernel. SVM finds the best hyperplane to divide the classes, making it ideal for high-dimensional data like the TF-IDF features produced from email text.

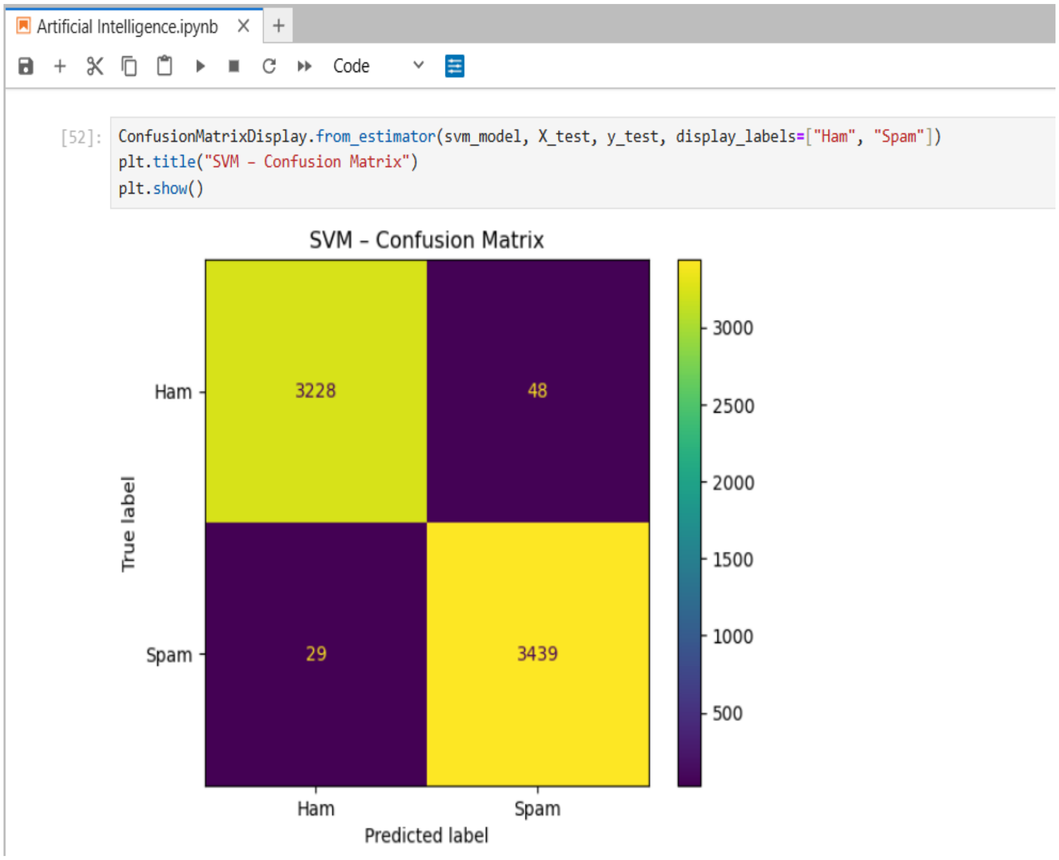


Figure 24: Confusion Matrix of Support Vector Machine (SVM)

To see how the Support Vector Machine model performs in email classification, a confusion matrix is created. ConfusionMatrixDisplay is used. The matrix from_estimator displays the quantity of true positive, true negative false negative and false negatives, the model's strengths and weakness in differentiating between spam and ham emails are clearly understood thanks to visual representation which also makes it simple to recognize the kinds of mistakes the model makes.

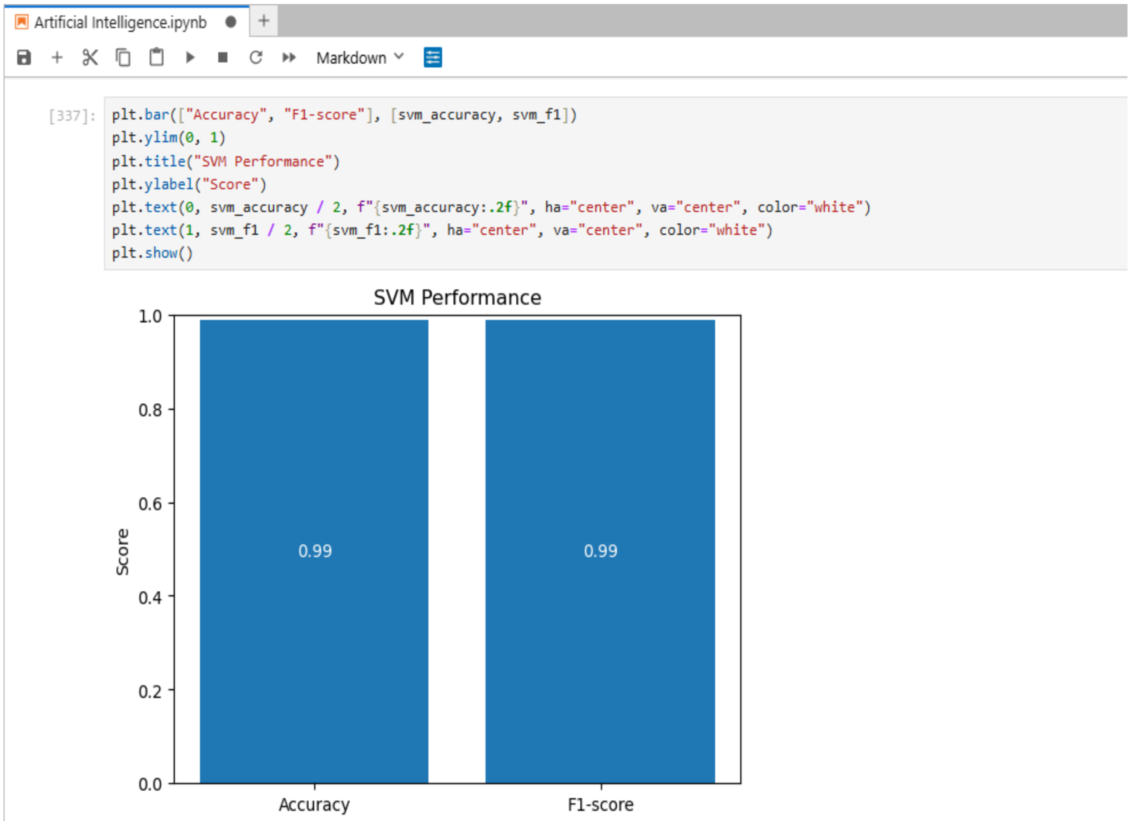


Figure 25: Accuracy and F1-score Comparison of SVM

The key metrics Accuracy and F1-score are used to visualize the Support Vector Machine model's performance in a bar chart. Each metrics numerical values are represented by the heights of the bars, for clarity the values are shown in the centre of the bars. The model's ability to accurately classify emails as spam or ham is summarized in this visualization, which is easy to understand and demonstrates both overall correctness and overall correctness and a balance between precision and recall.

3.6) Achieved Results

3.6.1) User Input

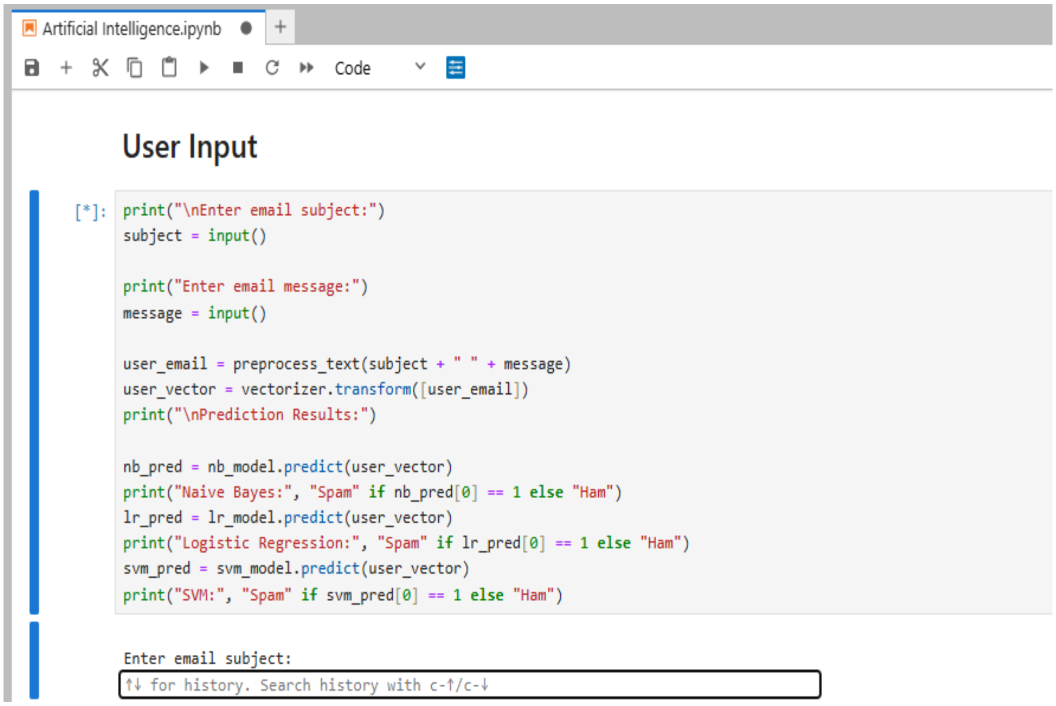


Figure 26: User Input

By entering the subject and message of a new email, the user can input it into the system. The entered emails are pre-processed in the same manner as the training data, which incudes removing punctuation and special characters and converting it to lowercase. To ensure consistency with the featured used to train the models, the previously fitted vectorizer is then used to convert the processed text into a TF-IDF vector.

The emails spam or have status is then predicted using the trained Naïve Bayes, Logistic Regression and SVM models. Each model's predictions are shown so that the classification outcomes on unknown, real-word input can be directly compared. This step shows how the tainted models can be used practically for real time spam detection.

Figure 27: Output of Spam email

All three trained models, Naïve Bayes, Logistic Regression and Support Vector Machine (SVM) correctly identified a sample spam email with promotional keywords like “Congratulation”, “winner”, “cash prize” and “claim your reward” as spam. This shows that the models can effectively generalize to previously unseen email content and successfully learned spam-related patterns during training.

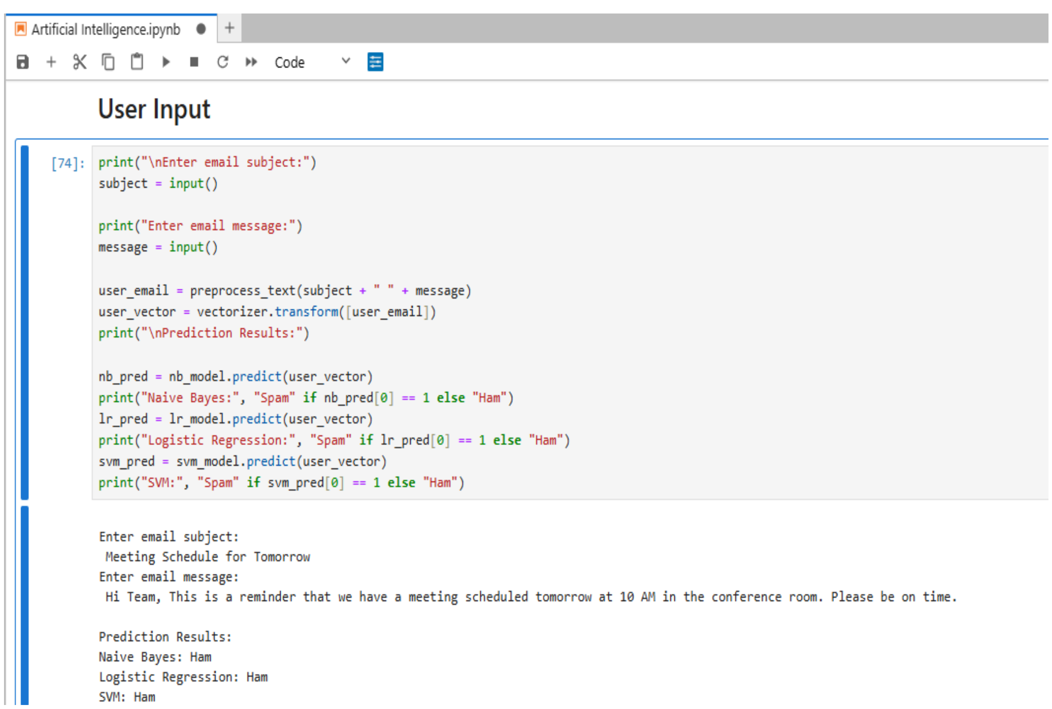
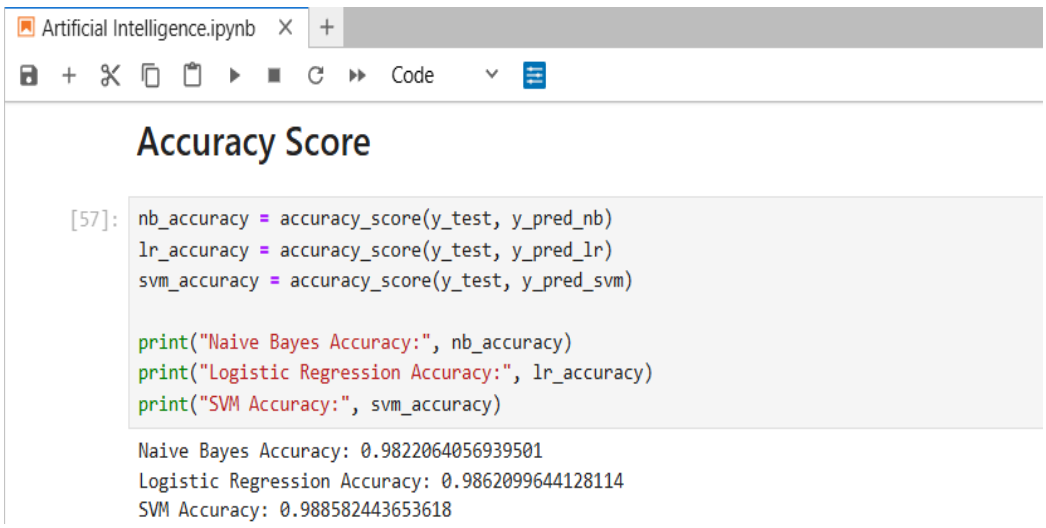


Figure 28: Output of Ham email

All three trained models, Naïve Bayes, Logistic Regression and Support Vector Machine (SVM) correctly identified a typical ham email about a meeting that contained common professional terms like “meeting”, “reminder”, “team” and “conference room”. This shows that the models are capable of differentiating between spam and ham emails and so not mistakenly classify routine correspondent as spam.

3.6.2) Accuracy Score of each Algorithm




```
nb_accuracy = accuracy_score(y_test, y_pred_nb)
lr_accuracy = accuracy_score(y_test, y_pred_lr)
svm_accuracy = accuracy_score(y_test, y_pred_svm)

print("Naive Bayes Accuracy:", nb_accuracy)
print("Logistic Regression Accuracy:", lr_accuracy)
print("SVM Accuracy:", svm_accuracy)
```

```
Naive Bayes Accuracy: 0.9822064056939501
Logistic Regression Accuracy: 0.9862099644128114
SVM Accuracy: 0.988582443653618
```

Figure 29: Accuracy score of reach algorithms

Each model’s accuracy in classifying emails as spam or ham was assessed using test dataset. These high accuracy values show that all three models are very good at differentiating between ham and spam emails. SVM outperformed the others in terms of accuracy, with Logistic Regression and Naïve Bayes coming is second and third respectively. This shoes that the models have effectively learned patterns from the dataset and are able to generalize to emails that have not yet been seen.

Figure 30: Accuracy comparison of each algorithm in bar graph

The accuracy of the Naïve Bayes, Logistic Regression and SVM models are graphically compared in the bar chart. For clarity, the numerical accuracy is shown in the centre of each bar, which displays the model's performance on the test dataset.

3.6.3) Train-Test Generalization check for data Overfitting

Artificial Intelligence.ipynb

Train-Test Generalization Check

```
[160]: print("Train accuracy:", nb_model.score(X_train, y_train))
       print("Test accuracy:", nb_model.score(X_test, y_test))

Train accuracy: 0.9824262197834792
Test accuracy: 0.9822064056939501
```

```
[161]: print("Train accuracy:", lr_model.score(X_train, y_train))
       print("Test accuracy:", lr_model.score(X_test, y_test))

Train accuracy: 0.991101883434673
Test accuracy: 0.9862099644128114
```

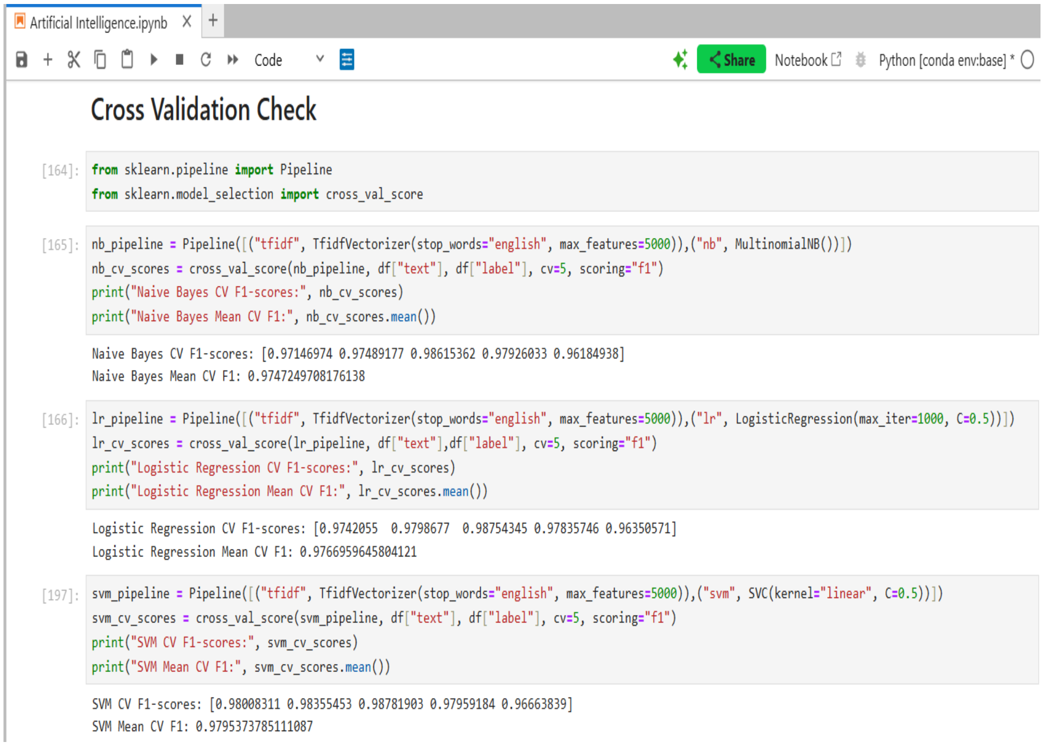
```
[162]: print("Train accuracy:", svm_model.score(X_train, y_train))
       print("Test accuracy:", svm_model.score(X_test, y_test))

Train accuracy: 0.9946240545751149
Test accuracy: 0.988582443653618
```

Figure 31: Train-Test Generalization Check for Overfitting

The code is utilized to estimate and compare the preformation if trained machine learning classifiers on training as well as test dataset. The score() function is used to estimate the accuracy of each classifier based on the number if properly classified emails.

The Accuracy of training depicts the effectiveness of the proposed model in learning from the data it is trained upon, whereas the accuracy of testing reveals the capability of the proposed models in generalization of data. The comparison of both values for Naïve Bayes, Logistic Regression and SVM reveals the extent to which the proposed models overfit, underfit or generalize effectively.

3.6.4) Cross Validation check for Data Overfitting



Cross Validation Check

```
[164]: from sklearn.pipeline import Pipeline
       from sklearn.model_selection import cross_val_score

[165]: nb_pipeline = Pipeline([("tfidf", TfidfVectorizer(stop_words="english", max_features=5000)),("nb", MultinomialNB())])
       nb_cv_scores = cross_val_score(nb_pipeline, df["text"], df["label"], cv=5, scoring="f1")
       print("Naive Bayes CV F1-scores:", nb_cv_scores)
       print("Naive Bayes Mean CV F1:", nb_cv_scores.mean())

Naive Bayes CV F1-scores: [0.97146974 0.97489177 0.98615362 0.97926033 0.96184938]
Naive Bayes Mean CV F1: 0.9747249708176138

[166]: lr_pipeline = Pipeline([("tfidf", TfidfVectorizer(stop_words="english", max_features=5000)),("lr", LogisticRegression(max_iter=1000, C=0.5))])
       lr_cv_scores = cross_val_score(lr_pipeline, df["text"],df["label"], cv=5, scoring="f1")
       print("Logistic Regression CV F1-scores:", lr_cv_scores)
       print("Logistic Regression Mean CV F1:", lr_cv_scores.mean())

Logistic Regression CV F1-scores: [0.9742055  0.9798677  0.98754345 0.97835746 0.96350571]
Logistic Regression Mean CV F1: 0.9766959645804121

[197]: svm_pipeline = Pipeline([("tfidf", TfidfVectorizer(stop_words="english", max_features=5000)),("svm", SVC(kernel="linear", C=0.5))])
       svm_cv_scores = cross_val_score(svm_pipeline, df["text"], df["label"], cv=5, scoring="f1")
       print("SVM CV F1-scores:", svm_cv_scores)
       print("SVM Mean CV F1:", svm_cv_scores.mean())

SVM CV F1-scores: [0.98008311 0.98355453 0.98781903 0.97959184 0.96663839]
SVM Mean CV F1: 0.9795373785111087
```

Figure 32: Cross-Validation Check for Data Overfitting

This code conducts 5-fold cross-validation for the evaluation of the performance of three machine learning algorithms, i.e. Naïve Bayes, Logistic Regression and Support Vector Machine, using pipeline analysis, so that all text feature extraction is learned independently for each cross-validation split. The pipelined consist of features generated using TF-IDF transformation followed by classification algorithms.

The cross_val_score() function calculate the F1 score for each cross validation fold. These F1 scores give a good indication of how well a model is able to generalize. This is a much more reliable way of assessing model quality than using a train/test split and a good way to ensure that there is no overfitting.

3.6.5) Testing each Algorithm with 70% training and 30% Testing

Train-Test Split

```
[111]: from sklearn.model_selection import train_test_split

X_train_text, X_test_text, y_train, y_test = train_test_split( df["text"], df["label"], test_size=0.3, random_state=42)
```

Figure 33: 70% training and 30% testing Train-Test-Split

The train_test_split function is used to split the dataset into training and test subsets, where 70% of the dataset will be used for training and 30% for testing purposes.

```
Naive Bayes Accuracy: 0.9825012357884331
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      4945
           1       0.98      0.99      0.98      5170

    accuracy                           0.98     10115
   macro avg       0.98      0.98      0.98     10115
weighted avg       0.98      0.98      0.98     10115
```

Figure 34: 70% training and 30% Naive Bayes Output

Naïve Bayes output for 70% training and 30% testing.

```
Logistic Regression Accuracy: 0.9865546218487395
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      4945
           1       0.98      0.99      0.99      5170

    accuracy                           0.99     10115
   macro avg       0.99      0.99      0.99     10115
weighted avg       0.99      0.99      0.99     10115
```

Figure 35: 70% training and 30% Logistic Regression Output

Logistic Regression output for 70% training and 30% testing.

```
SVM Accuracy: 0.9884330202669303
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      4945
           1       0.99      0.99      0.99      5170

    accuracy                           0.99     10115
   macro avg       0.99      0.99      0.99     10115
weighted avg       0.99      0.99      0.99     10115
```

Figure 36: 70% training and 30% SVM Output

SVM output for 70% training and 30% testing.

3.6.6) Testing each Algorithm with 60% training and 40% Testing

Train-Test Split

```
[165]: from sklearn.model_selection import train_test_split

X_train_text, X_test_text, y_train, y_test = train_test_split( df["text"], df["label"], test_size=0.4, random_state=42)
```

Figure 37: 60% training and 40% testing Train-Test-Split

The train_test_split function is used to split the dataset into training and test subsets, where 60% of the dataset will be used for training and 40% for testing purposes.

```
Naive Bayes Accuracy: 0.9827241046934084
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      6597
           1       0.98      0.99      0.98      6890

    accuracy                           0.98     13487
   macro avg       0.98      0.98      0.98     13487
weighted avg       0.98      0.98      0.98     13487
```

Figure 38: 60% training and 40% testing Naive Bayes Output

Naïve Bayes output for 60% training and 40% testing.

```
Logistic Regression Accuracy: 0.9869503966782828
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      6597
           1       0.98      1.00      0.99      6890

    accuracy                           0.99     13487
   macro avg       0.99      0.99      0.99     13487
weighted avg       0.99      0.99      0.99     13487
```

Figure 39: 60% training and 40% testing Logistic Regression Output

Logistic Regression output for 60% training and 40% testing.

```
SVM Accuracy: 0.9886557425669162
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      6597
           1       0.98      0.99      0.99      6890

    accuracy                           0.99     13487
   macro avg       0.99      0.99      0.99     13487
weighted avg       0.99      0.99      0.99     13487
```

Figure 40: 60% training and 40% testing SVM Output

SVM output for 60% training and 40% testing.

3.6.7) Testing each Algorithm with 50% training and 50% Testing

Train-Test Split

```
[346]: from sklearn.model_selection import train_test_split

X_train_text, X_test_text, y_train, y_test = train_test_split( df["text"], df["label"], test_size=0.5, random_state=42)
```

Figure 41: 50% training and 50% testing Train-Test-Split

The train_test_split function is used to split the dataset into training and test subsets, where 50% of the dataset will be used for training and 50% for testing purposes.

```
Naive Bayes Accuracy: 0.981492466484755
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      8252
           1       0.98      0.99      0.98      8606

    accuracy                           0.98     16858
   macro avg       0.98      0.98      0.98     16858
weighted avg       0.98      0.98      0.98     16858
```

Figure 42: 50% training and 50% testing Naive Bayes Output

Naïve Bayes output for 50% training and 50% testing.

```
Logistic Regression Accuracy: 0.9855854787044727
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      8252
           1       0.98      0.99      0.99      8606

    accuracy                           0.99     16858
   macro avg       0.99      0.99      0.99     16858
weighted avg       0.99      0.99      0.99     16858
```

Figure 43: 50% training and 50% testing Logistic Regression Output

Logistic Regression output for 50% training and 50% testing.

```
SVM Accuracy: 0.9871277731640764
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      8252
           1       0.98      0.99      0.99      8606

    accuracy                           0.99     16858
   macro avg       0.99      0.99      0.99     16858
weighted avg       0.99      0.99      0.99     16858
```

Figure 44: 50% training and 50% testing SVM Output

SVM output for 50% training and 50% testing.

4) Conclusion

This Project was successful in the application of a spam email classifier trough the application of machine learning algorithms. The Enron Spam Dataset was pre-processed for analysis by applying Natural Language Processing (NLP) techniques, which include text cleaning and TF-IDF. The project applied three different classification algorithms, which include the application of the Naïve Bayes, Logistic Regression and Support Vector Machine. The experiment showed that the three classifiers had a high accuracy and F-score and the Support Vector Machine outperformed the rest.

Spam emails pose one of the most visible problems in the current world because this type of mail might contribute to loss of productivity, security hazards and phishing scams. The application created portrays the aspect of using machine learning to classify emails as spam/ham in real time. The application enables one to enter the content of mail for instant prediction of whether it should be categorized as spam/ham. This application addresses the problems of unsolicited emails by promoting security in digital communication.

Future enhancement may also include using sophisticated deep learning techniques such as Long Short-term Memory (LSTM) network or transformation such as BERT. Furthermore, system enhancement may also include using metadata of emails, incorporating handling of multilingual spam messages and also developing the system as a web or mobile application.

5) Bibliography

1. Borotic, G. (2024) *Paradigm* [Online]. Available from: https://reference-global.com/article/10.2478/crdj-2023-0007.
2. Budiman, D. (2024) *Researchgate* [Online]. Available from: https://www.researchgate.net/publication/378083262_Email_spam_detection_a_comparison_of_svm_and_naive_bayes_using_bayesian_optimization_and_grid_search_parameters.
3. Charanarur, P. (2023) *SPRINGER NATURE Link* [Online]. Available from: https://link.springer.com/article/10.1007/s42979-023-02330-x.
4. Cole Stryker, Eda Kavlakoglu. (2025) *IBM* [Online]. Available from: https://www.ibm.com/think/topics/artificial-intelligence.
5. Perumal, D.A. (2025) *Multisearchjournal* [Online]. Available from: https://multiresearchjournal.theviews.in/uploads/articles/3-2-151.1.pdf.
6. Sarkar, S. (2025) *Geeks for geeks* [Online]. Available from: https://www.geeksforgeeks.org/machine-learning/logistic-regression-and-the-feature-scaling-ensemble/.
7. SCIKIT. (2024) *SCIKIT LEARN* [Online]. Available from: https://scikit-learn.org/stable/modules/naive_bayes.html.
8. SCIKIT. (2024) *SCIKIT LEARN* [Online]. Available from: https://scikit-learn.org/stable/modules/svm.html.