EMAIL SPAM DETECTION

Project Report

SUPERVISOR: Mr. Mahesh Kumar

SUBMITTED BY

Shahnwaz Khan: 20001570046

Nilesh Pandey: 20001570032

Prakash Kumar Singh: 20001570037

2023

Department of Computer Science

ACHARYA NARENDRA DEV COLLEGE

ACKNOWLEDGEMENT

This project was jointly undertaken by Shahnwaz Khan, Prakash Kumar Singh, and Nilesh Pandey as the Semester-VI Data Mining Project under the guidance and supervision of Mr. Mahesh Kumar. We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. We are highly indebted to Mr. Mahesh Kumar for his guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project. Our thanks and appreciation also go to our colleagues in developing the project and the people who have willingly helped us out with their abilities.

Shahnwaz Khan   Prakash Kumar Singh   Nilesh Pandey

ACHARYA NARENDRA DEV COLLEGE
(University of Delhi)

CERTIFICATE

This is to certify that the project entitled, “EMAIL SPAM DETECTION” has been done by Shahnwaz Khan, Nilesh Pandey, and Prakash Kumar Singh in partial fulfilment of the requirements for the award of “Bachelor of Computer Science(Honors)” during semester-VI at the “Acharya Narendra Dev College” under the supervision and guidance of Mr. Mahesh Kumar.

Shahnwaz Khan	Prakash Kumar Singh	Nilesh Pandey

Supervisor
Mr. Mahesh Kumar

Table of Contents

Topic

Chapter 1

PROBLEM STATEMENT

Email spam is a pervasive problem in modern communication, with millions of users receiving unsolicited and unwanted messages every day. Traditional methods of filtering spam, such as keyword-based filtering and blacklisting, are often ineffective due to the dynamic and evolving nature of spam messages. Therefore, there is a need to develop more robust and accurate spam detection systems that can adapt to changing spam patterns.

One potential solution is to use data mining techniques, such as Decision tree classifiers, to analyze large datasets of emails and identify patterns that can be used to distinguish between legitimate emails and spam. However, there are several challenges that need to be addressed in developing such a system, including the high dimensionality of email data, the presence of noisy and irrelevant features, and the need for real-time detection.

The aim of this data mining project is to develop an effective email spam detection system using Decision tree classifiers. The project will involve collecting and preprocessing a large dataset of emails, selecting relevant features, and training a Decision tree model to classify emails as either spam or non-spam. The ultimate goal is to develop a system that can accurately and efficiently detect spam emails, with the potential to be used in real-world applications.

Chapter 2

DATA MINING TECHNIQUES

2.1. DM Techniques

2.1.1. Classification

Classification is a task in data mining that involves assigning a class label to each instance in a dataset based on its features. The goal of classification is to build a model that accurately predicts the class labels of new instances based on their features.

There are two main types of classification: binary classification and multi-class classification. Binary classification involves classifying instances into two classes, such as “spam” or “not spam”, while multi-class classification involves classifying instances into more than two classes.

2.1.2. Association

Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable. It tries to find some interesting relations or associations among the variables of the dataset. It is based on different rules to discover the interesting relations between variables in the database.
Association rule learning is one of the very important concepts of data mining, and it is employed in Market Basket analysis, Web usage mining, continuous production, etc.

2.1.3. Clustering

Clustering is an unsupervised Machine Learning-based Algorithm that comprises a group of data points into clusters so that the objects belong to the same group.

Clustering helps to splits data into several subsets. Each of these subsets contains data similar to each other, and these subsets are called clusters. Now that the data from our customer base is divided into clusters, we can make an informed decision about who we think is best suited for this product.

2.2. Data mining technique used for this project.

2.2.1. Decision Tree

A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.

2.2.2 Naïve Bayes

Naive Bayes is a probabilistic algorithm used for classification and is based on Bayes' theorem. It is called "naive" because it assumes that the features used to classify the data are independent of each other, which may not always be true.

The algorithm calculates the probability of a data point belonging to each class based on the occurrence of its features. It then selects the class with the highest probability as the predicted class for that data point.

Naive Bayes is commonly used in text classification tasks, such as spam filtering and sentiment analysis. It works well with large datasets and is computationally efficient. However, its performance may suffer when the independence assumption is violated or when the training data is imbalanced or contains a large number of irrelevant features.

2.2.3. K NEAREST NEIGHBORS (KNN)

K-Nearest Neighbors (KNN) is a supervised learning algorithm used for classification and regression tasks. It is a non-parametric algorithm, which means it doesn't make any assumptions about the underlying distribution of the data.

The algorithm works by finding the K nearest data points in the training set to a given input data point, based on some distance metric (such as Euclidean distance). It then predicts the class (in the case of classification) or the value (in the case of regression) of the input data point based on the majority class or the average value of its K nearest neighbors.

KNN can be used with any number of features, but it works best when the number of features is small relative to the size of the dataset. It is computationally expensive, as it requires calculating distances between the input data point and all the data points in the training set. However, it can be easily implemented and is effective when the decision boundary between classes is irregular.

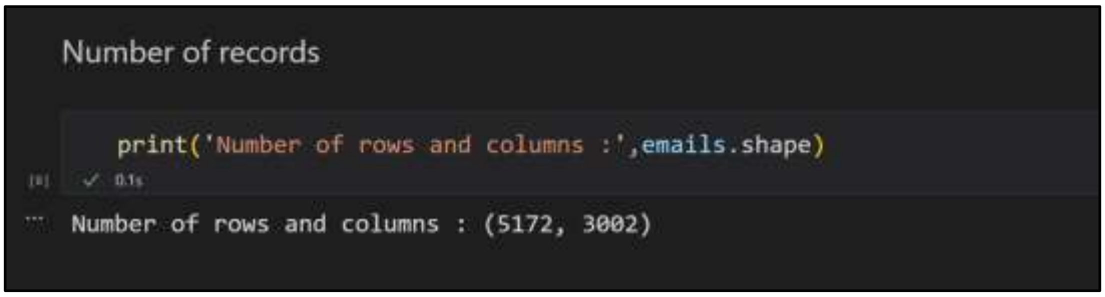
KNN is a versatile algorithm that can be used in a variety of applications, such as image recognition, recommender systems, and anomaly detection. However, its performance can be sensitive to the choice of K, the distance metric, and the scaling of the features.

Chapter 3

Dataset Description

3.1 Dataset

3.1.1. Number of Records

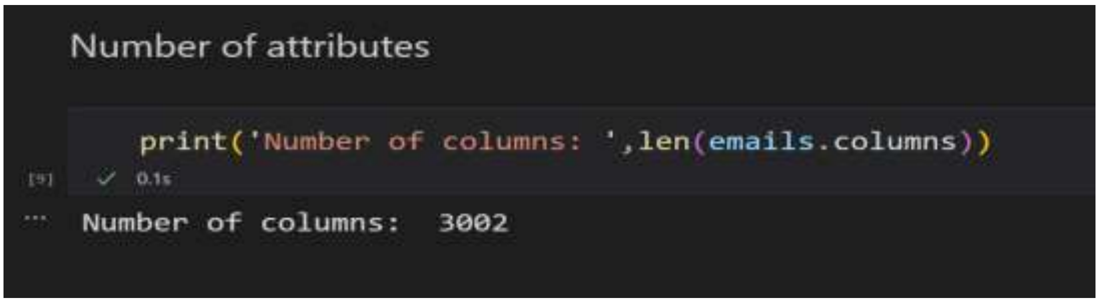


Number of records

```
print('Number of rows and columns :',emails.shape)
```

```
Number of rows and columns : (5172, 3002)
```

3.1.2 Number of Attributes

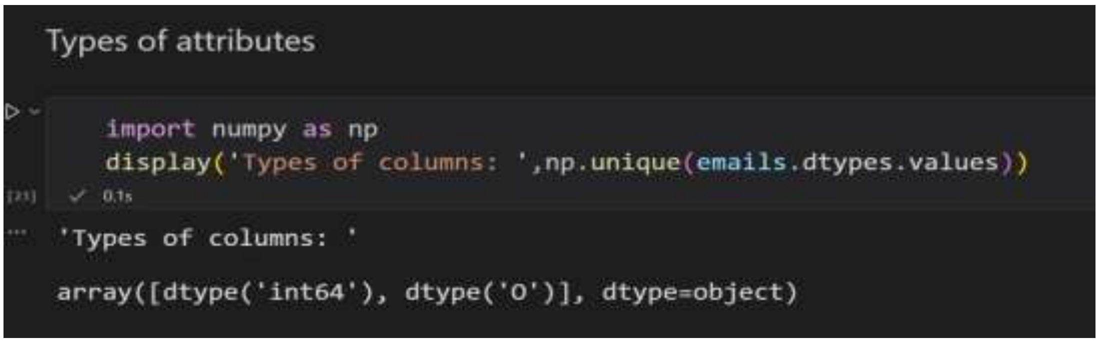


Number of attributes

```
print('Number of columns: ',len(emails.columns))
```

```
Number of columns:  3002
```

3.1.3. Types of Attributes



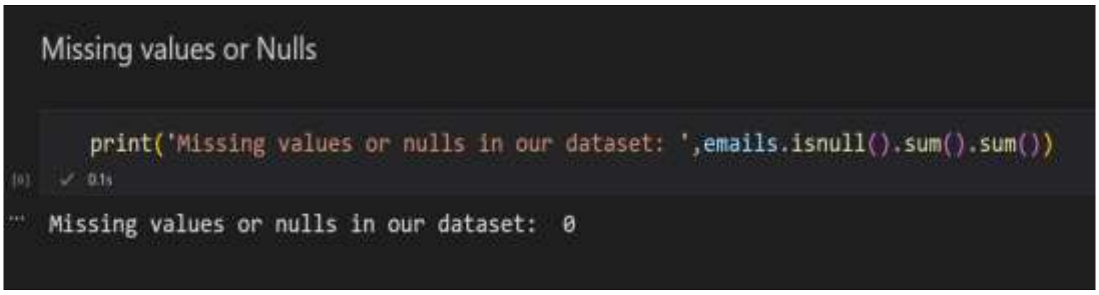
Types of attributes

```
import numpy as np
display('Types of columns: ',np.unique(emails.dtypes.values))
```

```
'Types of columns: '

array([dtype('int64'), dtype('O')], dtype=object)
```

3.1.4. Missing Values or Nulls



Missing values or Nulls

```
print('Missing values or nulls in our dataset: ',emails.isnull().sum().sum())
```

```
Missing values or nulls in our dataset:  0
```

3.1.5. Attributes Description

The first column indicates the Email name. The name has been set with numbers and not the recipient's name to protect privacy. The last column has the labels for prediction: 1 for spam, and 0 for not spam. The remaining 3000 columns are the 3000 most common words in all the emails, after excluding non-alphabetical characters/words.

3.1.6. Distribution/Histograms

Histogram

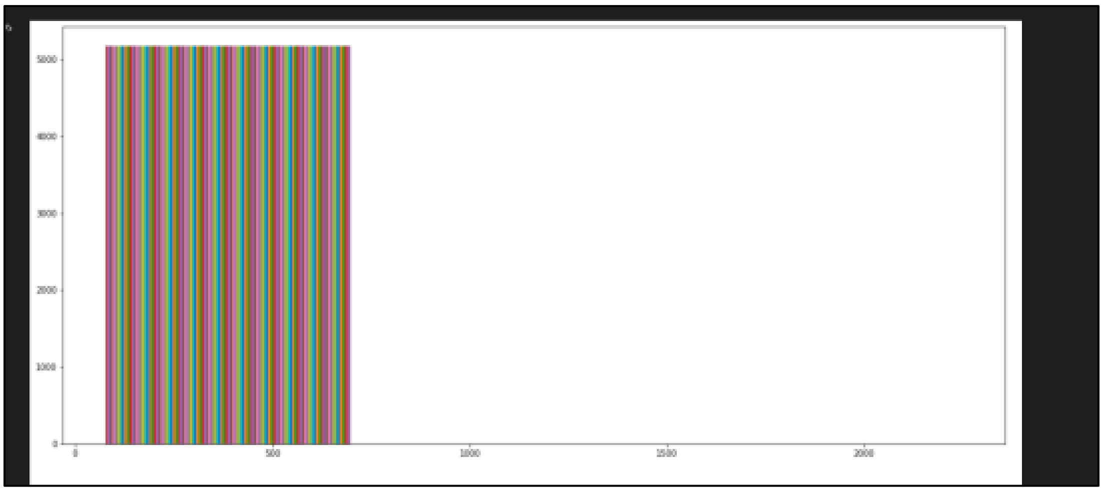


Figure 3.1: Histogram

3.1.7. Detecting Outliers

Box Plot

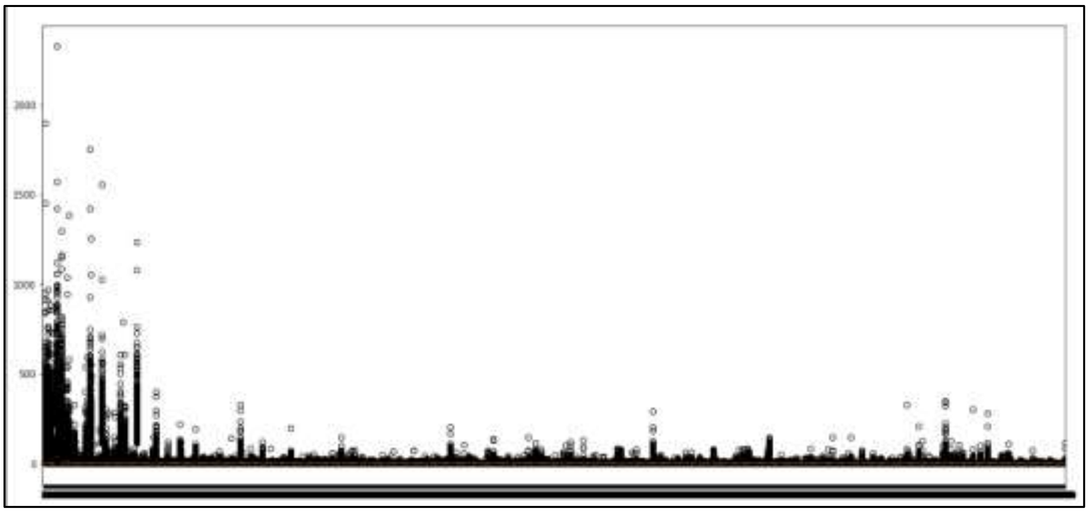


Figure 3.2: Outliers

Chapter 4

DATA PREPROCESSING

Preprocessing data is an important step for data analysis. The following are some benefits of preprocessing data:

- It improves accuracy and reliability. Preprocessing data removes missing or inconsistent data values resulting from human or computer error, which can improve the accuracy and quality of a dataset, making it more reliable.
- It makes data consistent. When collecting data, it's possible to have data duplicates, and discarding them during preprocessing can ensure the data values for analysis are consistent, which helps produce accurate results.
- It increases the data's algorithm readability. Preprocessing enhances the data's quality and makes it easier for machine learning algorithms to read, use, and interpret it.

Steps Involved in Data Preprocessing:

1 Data Cleaning:

The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

- **(a). Missing Data:**
  This situation arises when some data is missing in the data. It can be handled in various ways.
  Some of them are:
  1. **Ignore the tuples:**
     This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.

  2. **Fill the Missing values:**
     There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

- **(b). Noisy Data:**

  Noisy data is meaningless data that can’t be interpreted by machines. It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways:

  1. **Binning Method:**
     This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.

  2. **Regression:**
     Here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).

  3. **Clustering:**
     This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

2. Data Transformation:
This step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:

1. **Normalization:**
   It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)

2. **Attribute Selection:**
   In this strategy, new attributes are constructed from the given set of attributes to help the mining process.

3. **Discretization:**
   This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.

4. **Concept Hierarchy Generation:**
   Here attributes are converted from lower level to higher level in hierarchy. For Example-The attribute “city” can be converted to “country”.

3. Data Reduction:

Since data mining is a technique that is used to handle huge amount of data. While working with huge volume of data, analysis became harder in such cases. In order to get rid of this, we uses data reduction technique. It aims to increase the storage efficiency and reduce data storage and analysis costs.

The various steps to data reduction are:

1. **Data Cube Aggregation:**
   Aggregation operation is applied to data for the construction of the data cube.

2. **Attribute Subset Selection:**
   The highly relevant attributes should be used, rest all can be discarded. For performing attribute selection, one can use level of significance and p- value of the attribute.the attribute having p-value greater than significance level can be discarded.

3. **Numerosity Reduction:**
   This enable to store the model of data instead of whole data, for example: Regression Models.

4. **Dimensionality Reduction:**
   This reduce the size of data by encoding mechanisms.It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction are called lossless reduction else it is called lossy reduction. The two effective methods of dimensionality reduction are:Wavelet transforms and PCA (Principal Component Analysis).

4.1 HANDLING NULL VALUES

When no information is given for one or more elements, a whole unit, or both, this is known as missing data. Missing data poses a serious issue in real-world situations. In pandas, missing data can also refer to NA (Not Available) values. Many datasets in DataFrame occasionally arrive with missing data, either because the data was never collected or because it was present but was not captured. Assume, for instance, that some people being surveyed opt not to disclose their income, and that some users choose not to disclose their addresses. As a result, numerous datasets went missing.

Since the majority of the machine learning models you wish to employ will give you an error if you feed them NaN values, it is vital to fill in the missing data values in data sets. The simplest solution is to simply fill them with 0, however doing so will dramatically lower the accuracy of your model.

There are several ways available for replacing missing values. To fully grasp how to manage missing data in Python, you must first comprehend the type of missing value and its relevance before you can start filling in or removing the data.

There are 8 ways to handle missing values in our dataset:

1. **Drop rows or columns that have a missing value.**
2. **Drop rows or columns that only have missing values.**
3. **Fill with a constant value.**
4. **Fill with an aggregated value.**
5. **Replace with the previous or next value.**

In our dataset, there is no null value.

4.2 FEATURE SCALING

Describe feature scaling and describe

4.2.1 Normalization

Data normalization is a method used in data mining to convert a dataset's values onto a standard scale. This is significant because many machine learning algorithms are sensitive to the magnitude of the input features and benefit from the normalization of the data by producing more accurate results.

4.2.2 Standardization

Standardization or Z-Score Normalization is the transformation of features by subtracting from mean and dividing by standard deviation. This is often called as Z-score.

X_new = (X - mean)/Std

Standardization can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true.

4.2.3 Demonstrate Feature Scaling in your project.

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units. If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

Example: If an algorithm is not using the feature scaling method then it can consider the value 3000 meters to be greater than 5 km but that's actually not true and in this case, the algorithm will give wrong predictions. So, we use Feature Scaling to bring all values to the same magnitudes and thus, tackle this issue.

Techniques to perform Feature Scaling
Consider the two most important ones:

- **Min-Max Normalization:** This technique re-scales a feature or observation value with distribution value between 0 and 1.
- **Standardization:** It is a very effective technique which re-scales a feature value so that it has distribution with 0 mean value and variance equals to 1.

In our project, there is no need to perform feature scaling.

4.3 FEATURE SELECTION AND CONVERSION

What is Feature selection? What features are important for your problem statement?

Choosing the key features for the model is known as feature selection. A feature is a trait that affects or helps solve an issue. While developing the machine learning model, only a few variables in the dataset are useful for building the model, and the rest features are either redundant or irrelevant. If we input the dataset with all these redundant and irrelevant features, it may negatively impact and reduce the overall performance and accuracy of the model. Hence it is very important to identify and select the most appropriate features from the data and remove the irrelevant or less important features, which is done with the help of feature selection in machine learning.

In our dataset, the first column that is EmailNo is of no use. It is just the count of the emails. So if we do not remove this column then we shall end up with some unwanted result.

In feature transformation, we apply a mathematical formula to a specific column (feature) and alter the values to make them more relevant for our further analysis. It is a method by which we can improve the performance of our models. It is often referred to as "feature engineering," and it involves constructing new features out of preexisting ones in order to enhance the performance of the model.

There is not such attribute that requires conversion in our problem statement.

4.4 DATA SAMPLING AND SUBSETTING

A typical method for choosing a portion of the data objects to be analysed is sampling. It has long been used in statistics for both the first analysis of the data and the final analysis of the data. Data mining can also benefit greatly from sampling. But in statistics and data mining, sampling is frequently done for different reasons. While data miners employ sampling because processing all the data would be costly or time-consuming, statisticians use sampling because acquiring the whole set of relevant data is impractical. In some circumstances, a sampling method can help minimise the amount of data to the point where a more effective but pricy approach can be utilised.

Why do we need to split the data?
Every time we train a machine learning model, we are not able to train it on a single dataset, and even if we do, we won't be able to evaluate the performance of our model. We divided our source data into training, testing, and validation datasets as a result.

4.4.1 What are various ways of sampling data?

There are many sampling techniques, but only a few of the most basic ones and their variations will be covered here. The simplest type of sampling is **simple random sampling**. For this type of sampling, there is an equal probability of selecting any particular item. There are two variations on random sampling (and other sampling techniques as well):
(1) sampling without replacement—as each item is selected, it is removed from the set of all objects that together constitute the population.
(2) sampling with replacement—objects are not removed from the population as they are selected for the sample.
In sampling with replacement, the same object can be picked more than once. The samples produced by the two methods are not much different when samples are relatively small compared to the data set size, but sampling with replacement is simpler to analyze since the probability of selecting any object remains constant during the sampling process. When the population consists of different types of objects, with widely different numbers of objects, simple random sampling can fail to adequately represent those types of objects that are less frequent. This can cause problems when the analysis requires proper representation of all object types. For example, when building classification models for rare classes, it is critical that the rare

classes be adequately represented in the sample. Hence, a sampling scheme that can accommodate differing frequencies for the items of interest is needed.

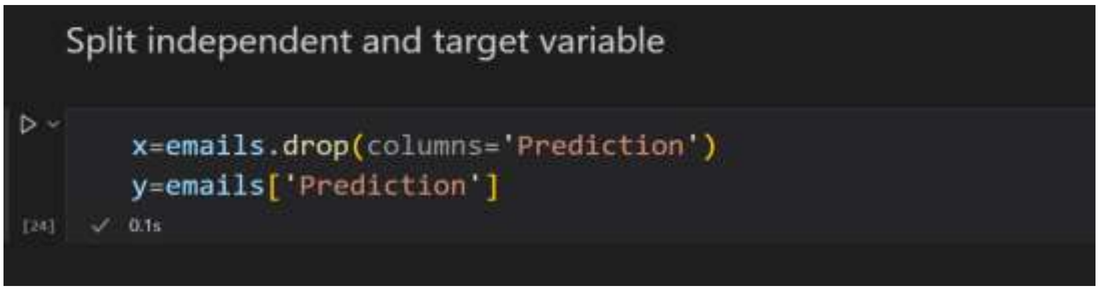
Stratified sampling, which starts with prespecified groups of objects, is such an approach. In the simplest version, equal numbers of objects are drawn from each group even though the groups are of different sizes. In another variation, the number of objects drawn from each group is proportional to the size of that group

4.4.2 TRAIN-TEST SPLIT

The train-test split is a technique for evaluating the performance of a machine learning algorithm.

It can be used for classification or regression problems and can be used for any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

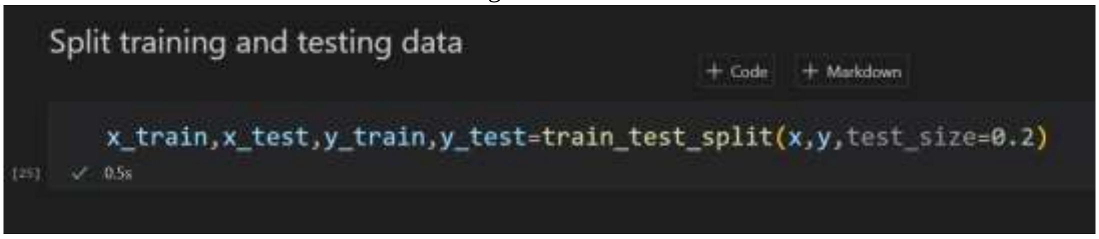


Split independent and target variable

```
x=emails.drop(columns='Prediction')
y=emails['Prediction']
```

```
[24] ✓ 0.1s
```

Figure 4.4.2.1:

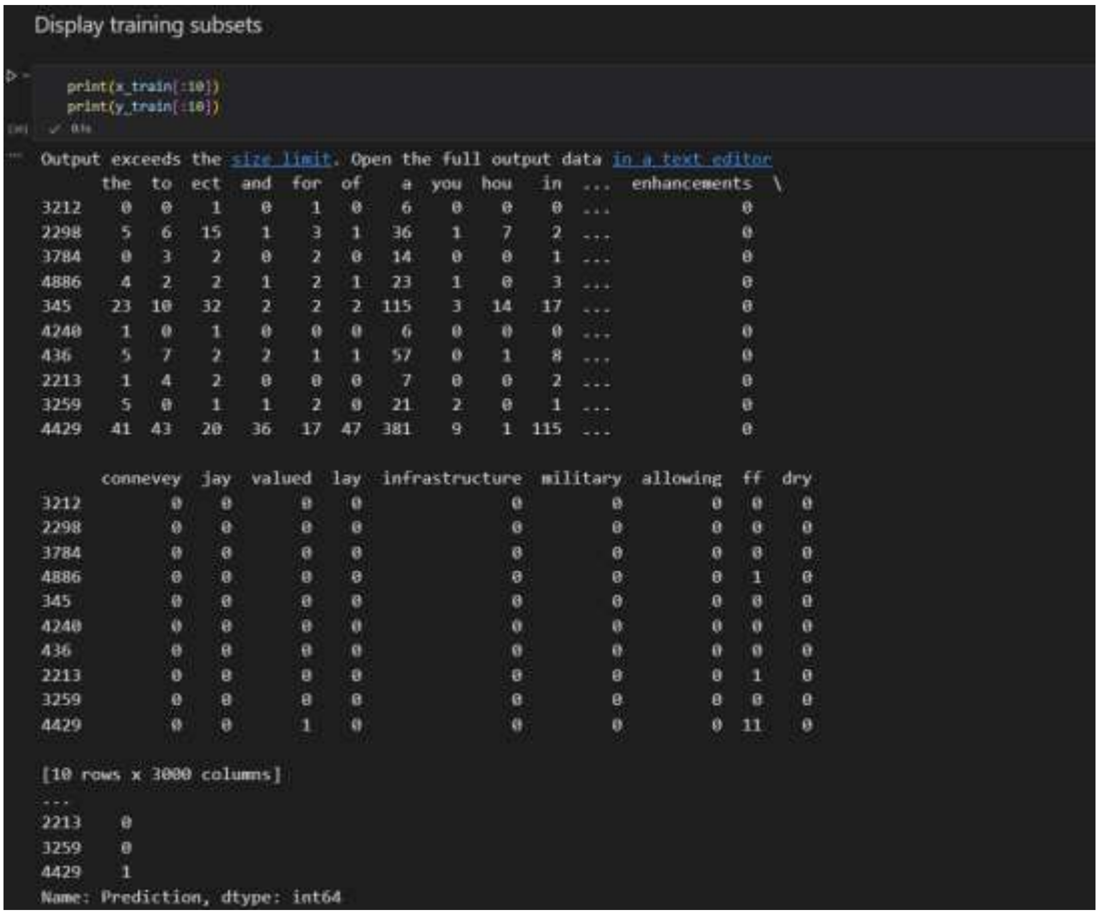


Split training and testing data

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

```
[25] ✓ 0.5s
```

Figure 4.4.2.2



Display training subsets

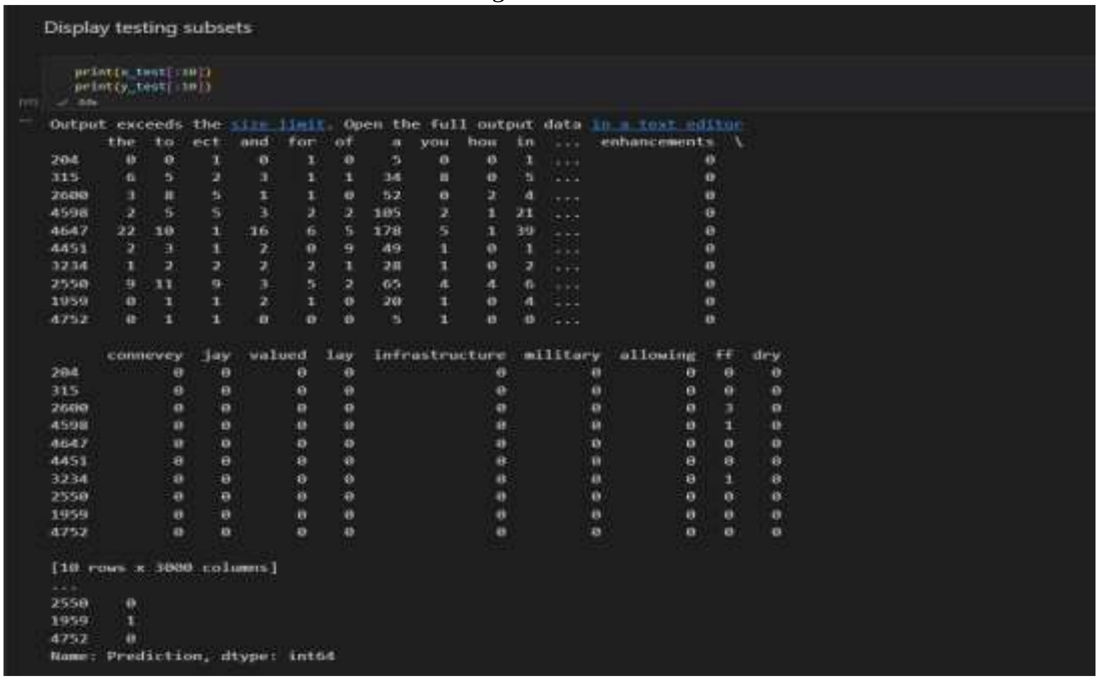
```
print(x_train[:10])
print(y_train[:10])
```

```
Output exceeds the size limit. Open the full output data in a text editor
      the  to  ect  and  for  of    a  you  hou   in  ...  enhancements  \
3212    0   0    1    0    1   0    6    0    0    0  ...             0
2298    5   6   15    1    3   1   36    1    7    2  ...             0
3784    0   3    2    0    2   0   14    0    0    1  ...             0
4886    4   2    2    1    2   1   23    1    0    3  ...             0
345    23  10   32    2    2   2  115    3   14   17  ...             0
4240    1   0    1    0    0   0    6    0    0    0  ...             0
436     5   7    2    2    1   1   57    0    1    8  ...             0
2213    1   4    2    0    0   0    7    0    0    2  ...             0
3259    5   0    1    1    2   0   21    2    0    1  ...             0
4429   41  43   20   36   17  47  381    9    1  115  ...             0

      connevey  jay  valued  lay  infrastructure  military  allowing  ff  dry
3212         0    0       0    0               0         0         0   0    0
2298         0    0       0    0               0         0         0   0    0
3784         0    0       0    0               0         0         0   0    0
4886         0    0       0    0               0         0         0   1    0
345          0    0       0    0               0         0         0   0    0
4240         0    0       0    0               0         0         0   0    0
436          0    0       0    0               0         0         0   0    0
2213         0    0       0    0               0         0         0   1    0
3259         0    0       0    0               0         0         0   0    0
4429         0    0       1    0               0         0         0  11    0

[10 rows x 3000 columns]
...
2213    0
3259    0
4429    1
Name: Prediction, dtype: int64
```

Figure 4.4.2.3



Display testing subsets

```
print(x_test[:10])
print(y_test[:10])
```

```
Output exceeds the size limit. Open the full output data in a text editor
      the  to  ect  and  for  of    a  you  hou  in  ...  enhancements  \
204     0   0    1    0    1   0    5    0    0   1  ...             0
315     6   5    2    3    1   1   34    8    0   5  ...             0
2600    3   8    5    1    1   0   52    0    2   4  ...             0
4598    2   5    5    3    2   2  105    2    1  21  ...             0
4647   22  10    1   16    6   5  178    5    1  39  ...             0
4451    2   3    1    2    0   9   49    1    0   1  ...             0
3234    1   2    2    2    2   1   28    1    0   2  ...             0
2550    9  11    9    3    5   2   65    4    4   6  ...             0
1959    0   1    1    2    1   0   20    1    0   4  ...             0
4752    0   1    1    0    0   0    5    1    0   0  ...             0

      connevey  jay  valued  lay  infrastructure  military  allowing  ff  dry
204          0    0       0    0               0         0         0   0    0
315          0    0       0    0               0         0         0   0    0
2600         0    0       0    0               0         0         0   3    0
4598         0    0       0    0               0         0         0   1    0
4647         0    0       0    0               0         0         0   0    0
4451         0    0       0    0               0         0         0   0    0
3234         0    0       0    0               0         0         0   1    0
2550         0    0       0    0               0         0         0   0    0
1959         0    0       0    0               0         0         0   0    0
4752         0    0       0    0               0         0         0   0    0

[10 rows x 3000 columns]
...
2550    0
1959    1
4752    0
Name: Prediction, dtype: int64
```

Figure 4.4.2.4

Chapter 5

Building Models

Describe why you need to train and test different models?

We need to train and test different models in data mining because it allows us to evaluate the performance of the models and select the one that best suits our needs.

When we train a model, we are teaching it how to make predictions based on the data we provide. We use a portion of the available data (called the training set) to train the model, and we adjust its parameters until it can make accurate predictions.

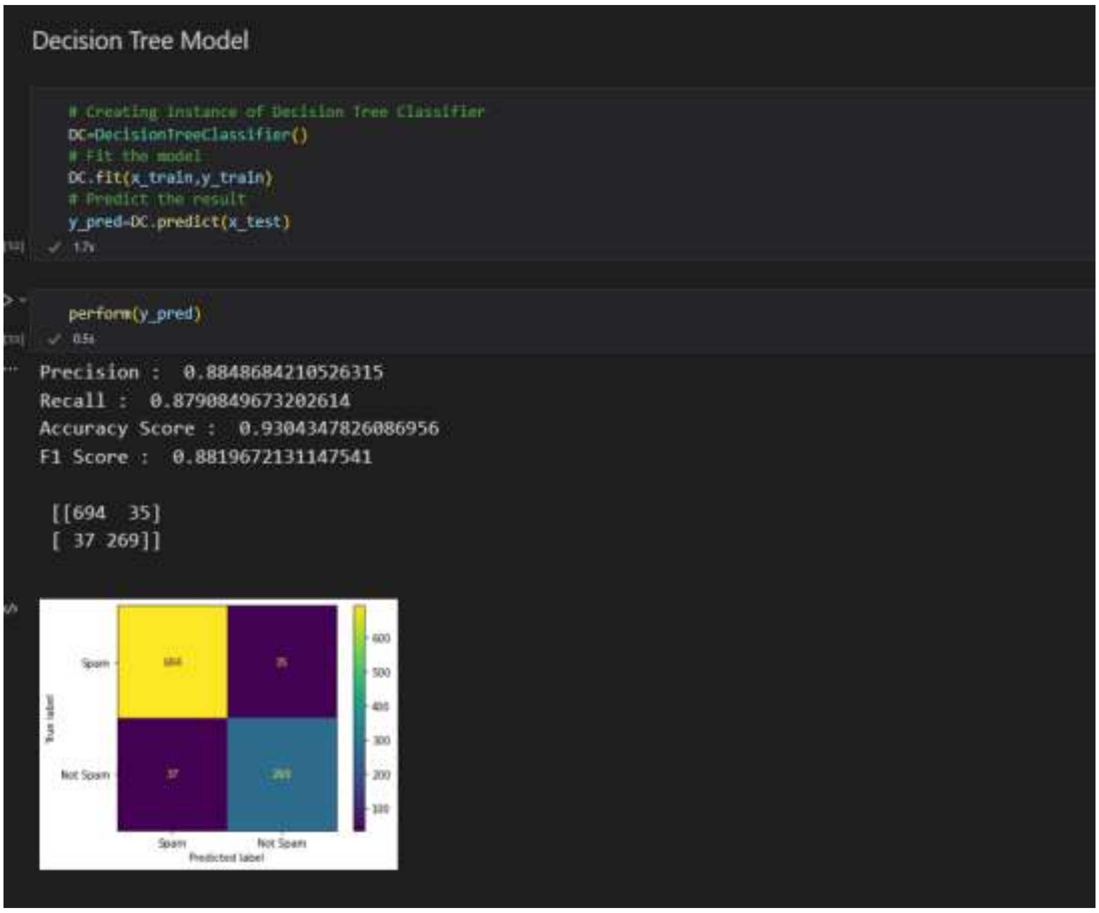
However, the ultimate goal of building a model is to use it to make predictions on new, unseen data. Therefore, it's important to test the model on a separate set of data (called the testing set) to see how well it performs on new data. This is known as model evaluation.

By testing different models on the same testing set, we can compare their performance and choose the one that performs best. This helps us to avoid overfitting (where the model is too closely tailored to the training data and performs poorly on new data) and to ensure that our model is robust and reliable.

In summary, training and testing different models allows us to evaluate their performance, select the best model, and ensure that it will perform well on new, unseen data.

5.1 Model1 (Decision Tree)

A decision tree model is a predictive model used in data mining. It uses a tree-like structure to represent a sequence of decisions and their possible outcomes. The tree is built using a dataset of labeled examples, and the algorithm selects the best attribute to split the dataset at each node. Once the tree is constructed, it can be used to classify or predict new examples by following the sequence of decisions. Decision tree models are easy to interpret and visualize, but they can overfit the training data.


Decision Tree Model

```
# Creating Instance of Decision Tree Classifier
DC=DecisionTreeClassifier()
# Fit the model
DC.fit(x_train,y_train)
# Predict the result
y_pred=DC.predict(x_test)
```

```
perform(y_pred)
```

```
Precision :  0.8848684210526315
Recall :  0.8790849673202614
Accuracy Score :  0.9304347826086956
F1 Score :  0.8819672131147541

 [[694  35]
 [ 37 269]]
```

Figure 5.1: Decision Tree

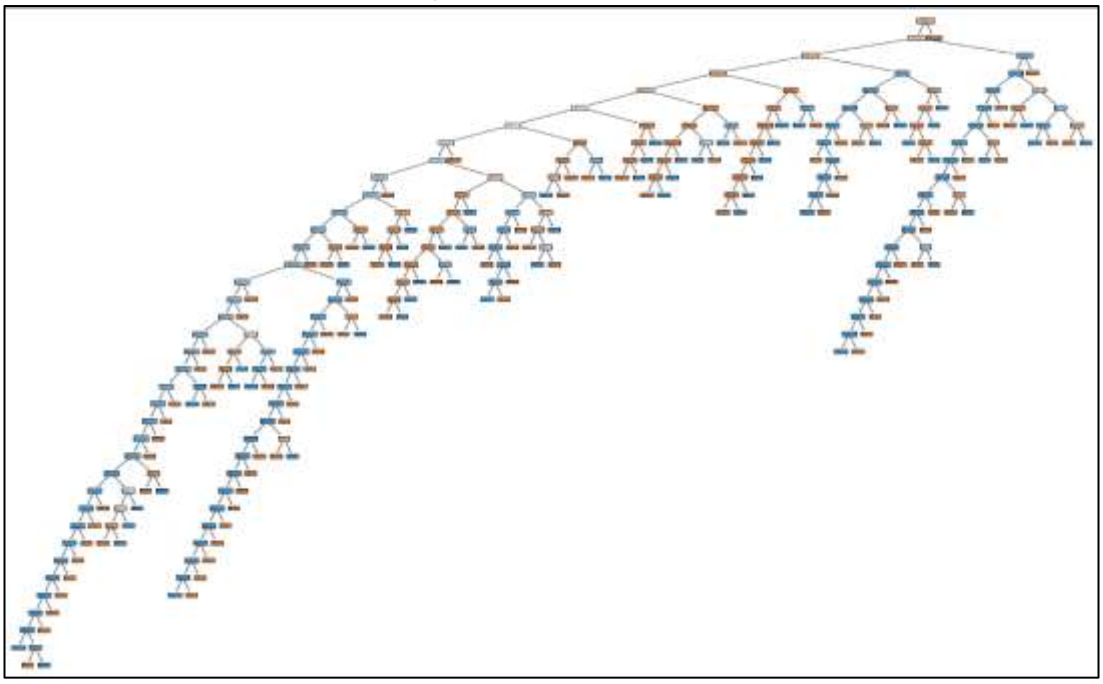
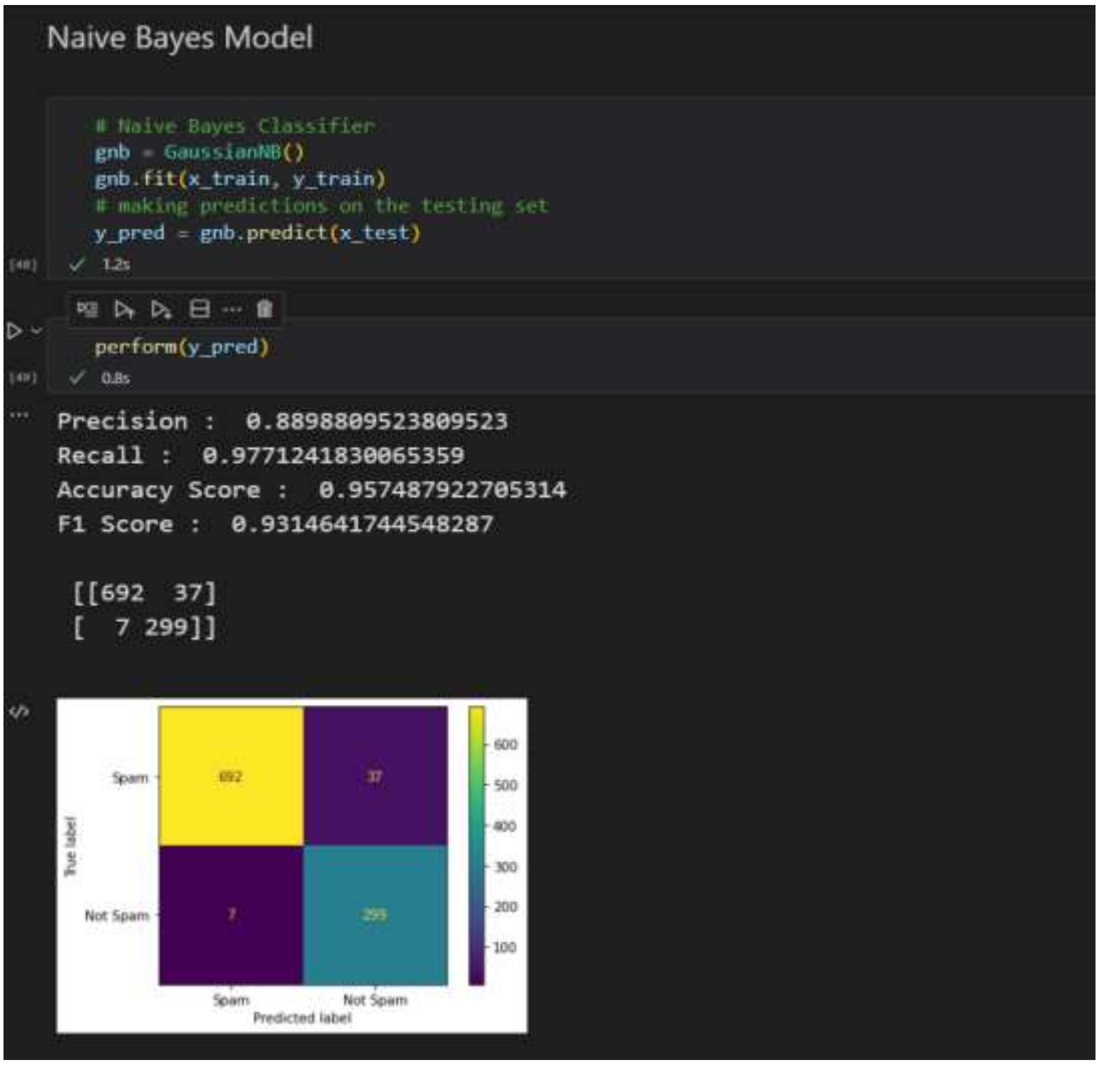


Figure 5.1.2 Decision Tree

5.2 Model2 (Naïve Bayes)

The Naive Bayes model is a probabilistic machine learning algorithm used for classification tasks in data mining. It is based on Bayes' theorem, which describes the probability of an event based on prior knowledge of conditions that might be related to the event.

The Naive Bayes model assumes that the presence or absence of a feature is independent of the presence or absence of any other feature, given the class variable. This assumption simplifies the calculation of probabilities.



Naive Bayes Model

```
# Naive Bayes Classifier
gnb = GaussianNB()
gnb.fit(x_train, y_train)
# making predictions on the testing set
y_pred = gnb.predict(x_test)
```

```
perform(y_pred)
```

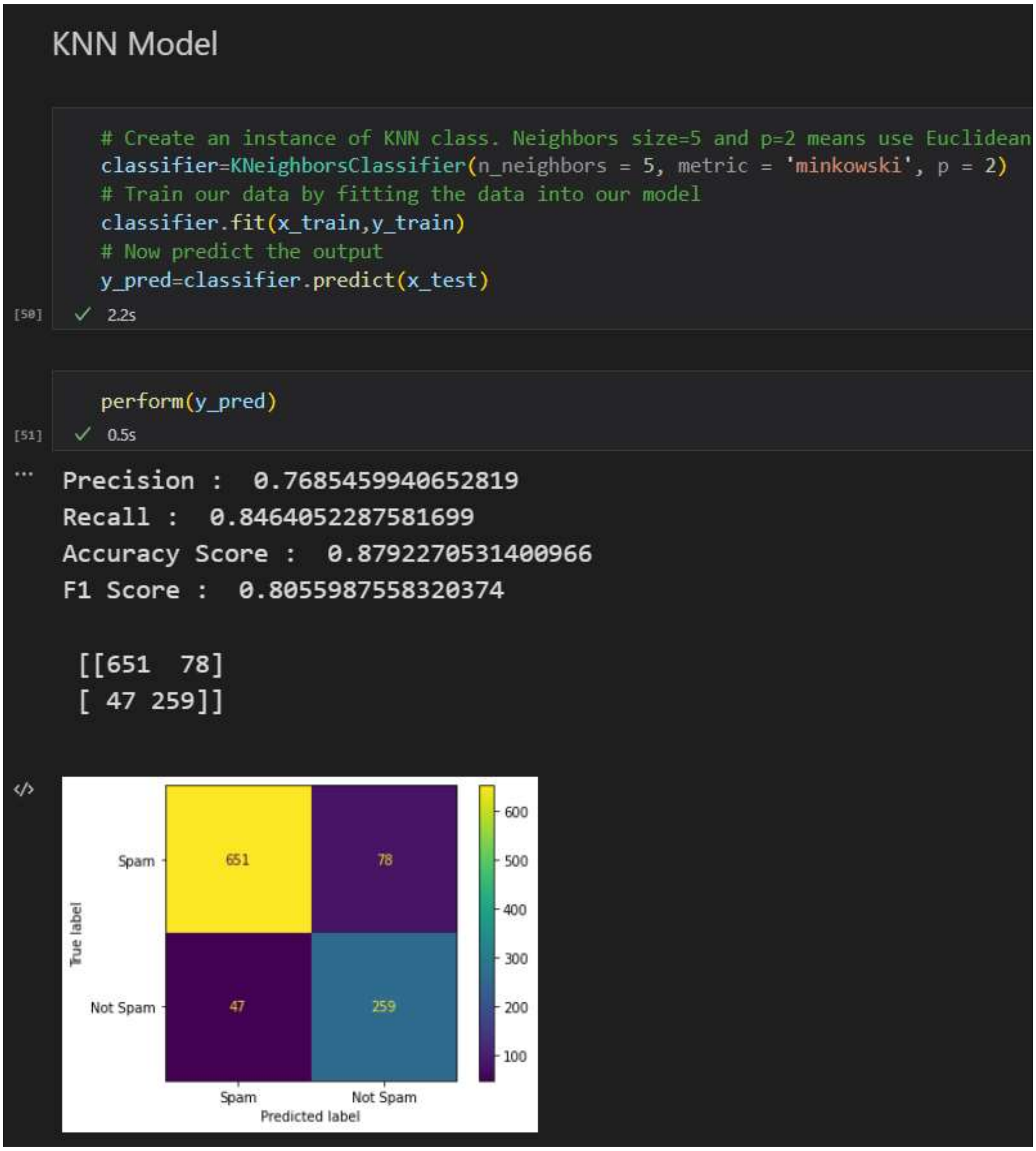
```
Precision :  0.8898809523809523
Recall :  0.9771241830065359
Accuracy Score :  0.957487922705314
F1 Score :  0.9314641744548287

 [[692  37]
 [  7 299]]
```

Figure 5.2: Naïve Bayes

5.3 Model3 (KNN)

The K-Nearest Neighbors (KNN) model is a machine learning algorithm used for both classification and regression tasks. It is a type of instance-based or lazy learning algorithm, which means that it does not create a model during the training phase. Instead, it stores the entire training dataset and uses it to make predictions on new, unseen data.



KNN Model

```
# Create an instance of KNN class. Neighbors size=5 and p=2 means use Euclidean
classifier=KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
# Train our data by fitting the data into our model
classifier.fit(x_train,y_train)
# Now predict the output
y_pred=classifier.predict(x_test)
```

[50] ✓ 2.2s

```
perform(y_pred)
```

[51] ✓ 0.5s

```
Precision :  0.7685459940652819
Recall :  0.8464052287581699
Accuracy Score :  0.8792270531400966
F1 Score :  0.8055987558320374

 [[651  78]
 [ 47 259]]
```

Figure 5.3: KNN Model

Chapter 6

MODEL EVALUATION AND RESULTS

6.1 METRICS

How do we compare models? What are the various metrics used. Describe them briefly.

Comparing models is an important step in machine learning to determine the performance of different models on a given task. There are various metrics used to evaluate the performance of a model, depending on the type of task and the nature of the data. Some commonly used metrics are:

Accuracy: The proportion of correctly classified instances out of the total instances. This metric is commonly used for classification tasks with balanced classes, where the number of instances in each class is roughly the same.

Precision: The proportion of true positives (correctly predicted positive instances) out of all positive predictions. This metric is used to evaluate the performance of a model in identifying positive instances, and is particularly useful when the cost of false positives is high.

Recall: The proportion of true positives out of all actual positive instances. This metric is used to evaluate the performance of a model in detecting all positive instances, and is particularly useful when the cost of false negatives is high.

F1-score: The harmonic mean of precision and recall, which balances the importance of precision and recall. This metric is commonly used in classification tasks with imbalanced classes.

Mean Squared Error (MSE): The average of the squared differences between the predicted and actual values in a regression task. This metric is commonly used to evaluate the performance of a regression model.

Root Mean Squared Error (RMSE): The square root of the MSE. This metric has the same unit as the predicted variable, making it more interpretable.

R-squared (R2): The proportion of the variance in the predicted variable that is explained by the model. This metric ranges from 0 to 1, where higher values indicate a better fit.

These metrics can be used to compare different models and select the best one for a given task. However, it is important to choose the appropriate metric based on the nature of the data and the goals of the task.

6.1.1 CONFUSION MATRIX

A confusion matrix is a table that is used to evaluate the performance of a classification model by comparing the predicted and actual values of the target variable. It is also known as an error matrix.

A confusion matrix consists of four components:

1. True Positives (TP): The number of instances that are correctly classified as positive.
2. False Positives (FP): The number of instances that are incorrectly classified as positive.
3. True Negatives (TN): The number of instances that are correctly classified as negative.
4. False Negatives (FN): The number of instances that are incorrectly classified as negative.

The confusion matrix is typically displayed in a table format with the predicted values along the top row and the actual values along the first column. The entries in the table represent the number of instances in each combination of predicted and actual values.

Using the values in the confusion matrix, several metrics can be calculated to evaluate the performance of a classification model, including accuracy, precision, recall, and F1-score. For example, accuracy is calculated as (TP + TN) / (TP + TN + FP + FN), and precision is calculated as TP / (TP + FP).

The confusion matrix provides a detailed view of the performance of a classification model and can help identify specific areas where the model is making errors. It is particularly useful in cases where the classes are imbalanced, and the number of instances in each class is not equal.

6.1.2 ACCURACY

The proportion of correctly classified instances out of the total instances. This metric is commonly used for classification tasks with balanced classes, where the number of instances in each class is roughly the same.

6.1.3 PRECISION

The proportion of true positives (correctly predicted positive instances) out of all positive predictions. This metric is used to evaluate the performance of a model in identifying positive instances, and is particularly useful when the cost of false positives is high.

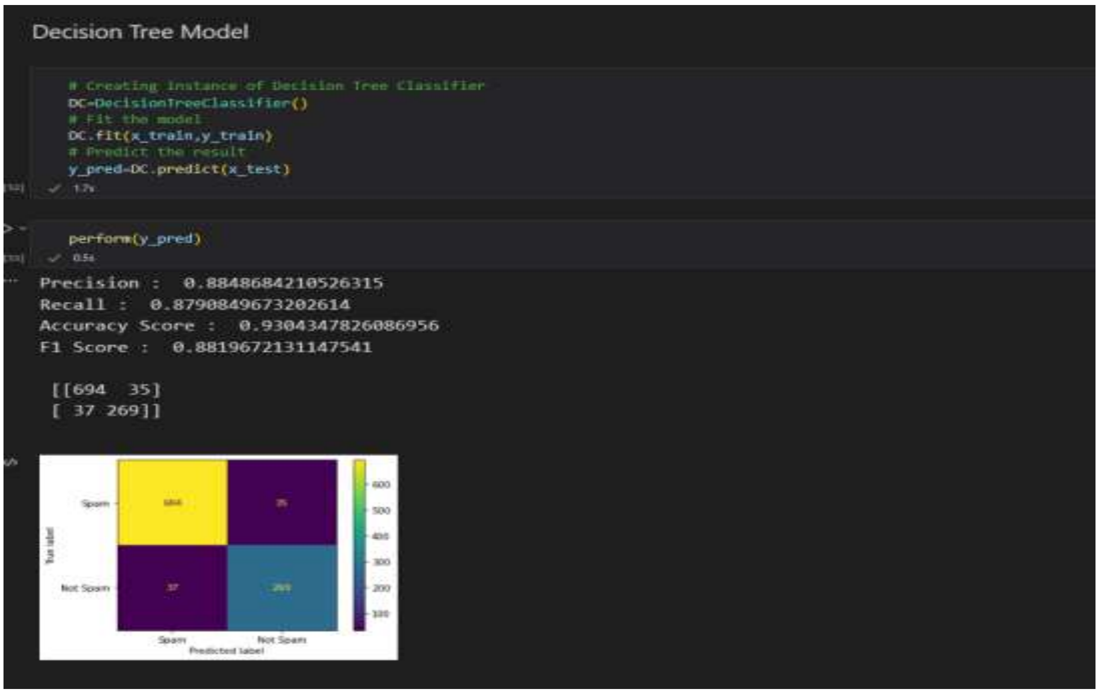
6.1.4 RECALL

The proportion of true positives out of all actual positive instances. This metric is used to evaluate the performance of a model in detecting all positive instances, and is particularly useful when the cost of false negatives is high.

6.1.5 F1-SCORE

The harmonic mean of precision and recall, which balances the importance of precision and recall. This metric is commonly used in classification tasks with imbalanced classes.

6.2. EXPERIMENTAL RESULTS AND COMPARISON



Decision Tree Model

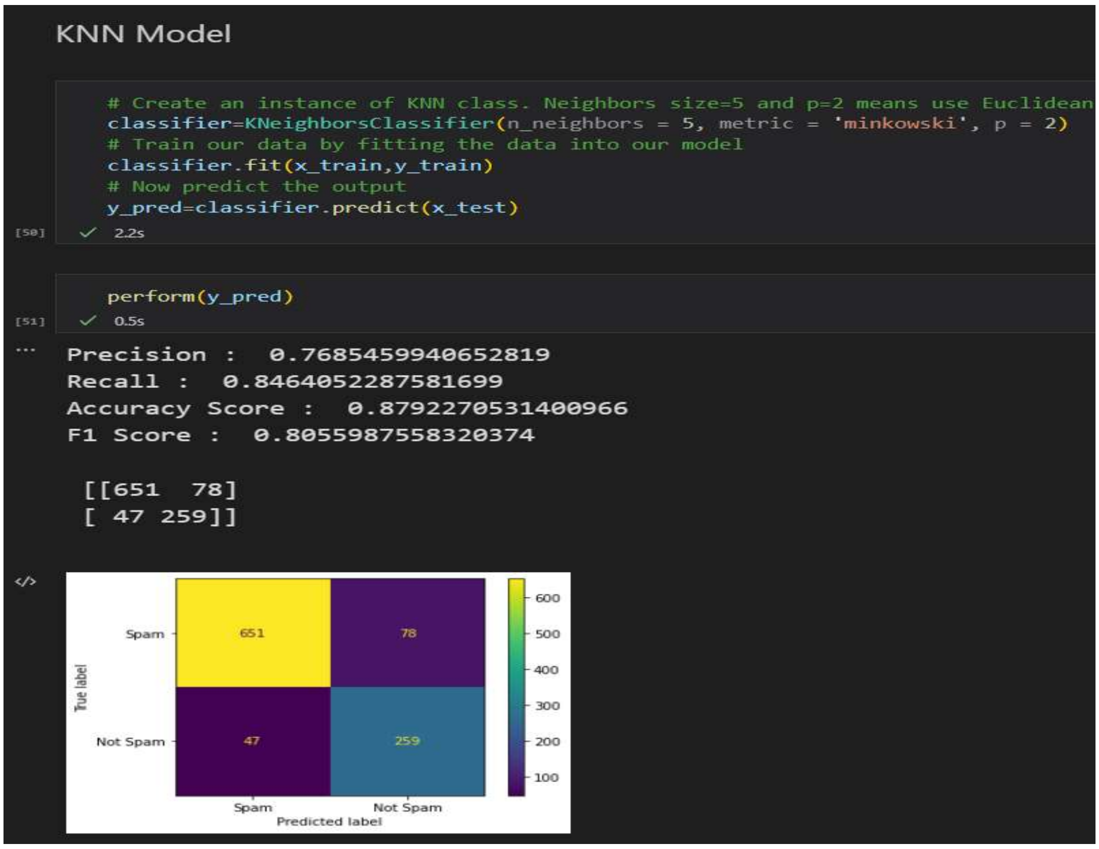
```
# Creating instance of Decision Tree Classifier
DC=DecisionTreeClassifier()
# Fit the model
DC.fit(x_train,y_train)
# Predict the result
y_pred=DC.predict(x_test)
```

```
perform(y_pred)
```

```
Precision :  0.8848684210526315
Recall :  0.8790849673202614
Accuracy Score :  0.9304347826086956
F1 Score :  0.8819672131147541

 [[694  35]
 [ 37 269]]
```

Figure 6.2.1: Decision Tree Model



KNN Model

```
# Create an instance of KNN class. Neighbors size=5 and p=2 means use Euclidean
classifier=KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
# Train our data by fitting the data into our model
classifier.fit(x_train,y_train)
# Now predict the output
y_pred=classifier.predict(x_test)
```

[50] ✓ 2.2s

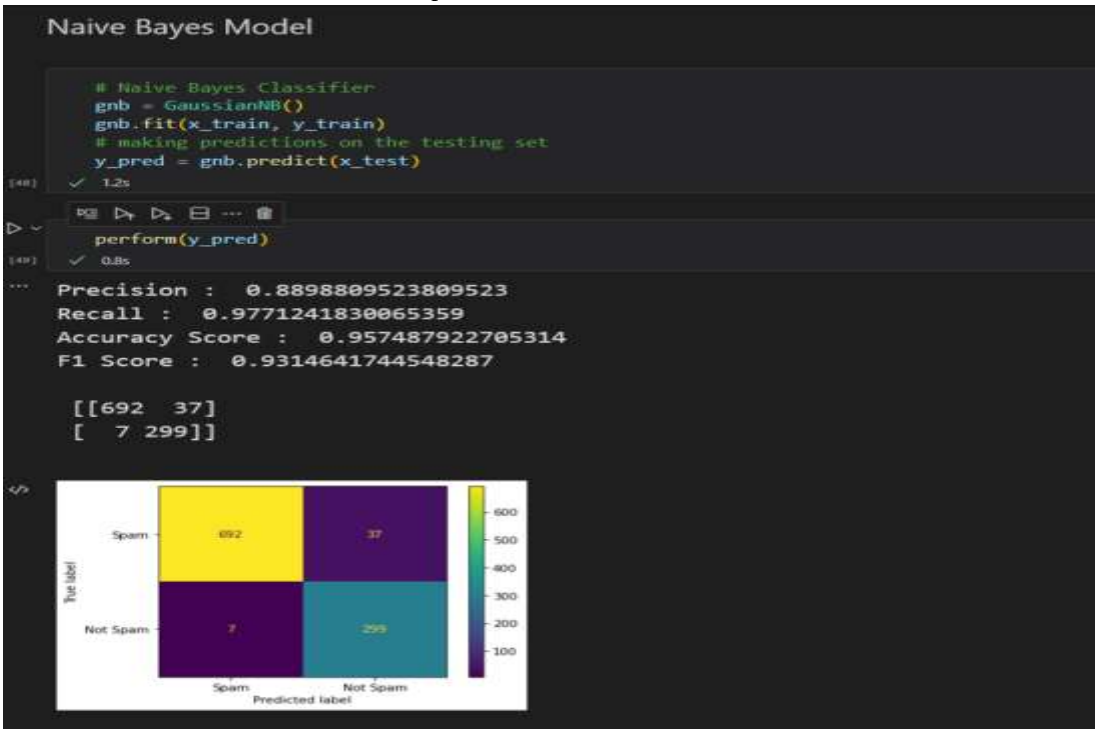
```
perform(y_pred)
```

[51] ✓ 0.5s

```
Precision :  0.7685459940652819
Recall :  0.8464052287581699
Accuracy Score :  0.8792270531400966
F1 Score :  0.8055987558320374

 [[651  78]
 [ 47 259]]
```

Figure 6.2.2: KNN Model



Naive Bayes Model

```
# Naive Bayes Classifier
gnb = GaussianNB()
gnb.fit(x_train, y_train)
# making predictions on the testing set
y_pred = gnb.predict(x_test)
```

[48] ✓ 1.2s

```
perform(y_pred)
```

[49] ✓ 0.8s

```
Precision :  0.8898809523809523
Recall :  0.9771241830065359
Accuracy Score :  0.957487922705314
F1 Score :  0.9314641744548287

 [[692  37]
 [  7 299]]
```

Figure 6.2.3: Naïve Bayes Model

Chapter 7

INFERENCES AND CONCLUSION

- Objective of this project is to find a model that classifies an email as spam or not spam with better accuracy. The given dataset is trained using three models KNN, DecisionTree, and Naïve Bayes.
- All the three models resulted in accuracy greater than 85%, which is quite good. Table 7.1 is a comparison table that compares all the three models to find a model better than all.

Attribute	**KNN**	**Decision Tree**	**Naïve Bayes**
Accuracy	0.88	0.92	0.95
Precision	0.78	0.88	0.89
Recall	0.87	0.87	0.94
F1 Score	0.82	0.88	0.92

Table 7.1

- After comparing all the attributes, it is clear that Naïve Bayes is the best model among all the models with better accuracy, precision, recall and F1 score.

Reference

- Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Pearson Education.
- Data Mining: Concepts and Techniques, 3nd edition,Jiawei Han and Micheline Kamber
- Data Mining: A Tutorial Based Primer, Richard Roiger, Michael Geatz, Pearson Education 2003.
- Introduction to Data Mining with Case Studies, G.K. Gupta, PHI 2006
- Insight into Data mining: Theory and Practice, Soman K. P., DiwakarShyam, Ajay V., PHI 2006
- https://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv