

**NANYANG  
TECHNOLOGICAL  
UNIVERSITY**  

---

**SINGAPORE**

## SC4020 Data Analytics and Mining

Academic Year 2025/2026

Semester 1

Group 18

**LUNBERRY NOAH IWATA (N2503869H)**

**PIKERINGA ANTONINA DAILA (N2504101A)**

**RAHLFS FREDERIC MAURITZ (N2504096K)**

**SARAH EMILY ONG XIN WEI (U2440124G)**

---

## Table of Contents

---

<b>Chapter 1</b>	<b>Task 1: Analysis of Symptom Co-occurrence Patterns ....</b>	<b>1</b>
1.1	Data Preprocessing.....	1
1.2	Implementation of Apriori Algorithm .....	1
1.3	Results.....	1
<b>Chapter 2</b>	<b>Task 2: Mining Cancer Feature Patterns .....</b>	<b>3</b>
2.1	Data Preprocessing.....	3
2.2	Building Sequences .....	3
2.3	Pattern Mining with GSP .....	3
2.4	Results.....	4
2.5	Discussion.....	4
2.6	Conclusion for Task 2.....	4
<b>Chapter 3</b>	<b>Task 3: Open Advanced Task .....</b>	<b>5</b>
3.1	Technique explanation.....	5
3.1.1	Random forest .....	5
3.1.2	Gradient boosting.....	5
3.1.3	Linear classifier .....	6
3.1.4	Autoencoder with linear classifier .....	6
3.2	Results and comparison .....	6
<b>Appendix A</b>	<b>- Breakdown of Contributions .....</b>	<b>A-1</b>

# 1 Task 1: Analysis of Symptom Co-occurrence Patterns

---

In this task, we analysed the co-occurrence pattern of different symptoms within disease profiles using the Apriori algorithm.

## 1.1 Data Preprocessing

The dataset is loaded from a CSV file from the Disease Symptom Prediction Dataset. It was then cleaned to remove null values. Each row represents a transaction. We then extract the symptoms (items) associated with each case from every row (transaction).

The symptoms are then converted to lowercase and we remove any whitespace. This ensures that when we store the symptoms, there are no duplicate symptoms within transactions. Using the `TransactionEncoder` from the `mlxtend` library, the transactions are transformed via one-hot encoding. This facilitates the implementation of the Apriori algorithm.

## 1.2 Implementation of Apriori Algorithm

The Apriori algorithm is implemented using the `mlxtend` library, with the goal of mining frequent itemsets from the preprocessed transaction data. The minimum support threshold is set to 0.1, meaning that an itemset must appear in at least 10% of the transactions to be considered frequent.

Following the discovery of frequent itemsets, the `association_rules` function is used to generate association rules with a minimum confidence threshold of 0.6. This ensures that only rules with a confidence greater than or equal to 60% are considered valid.

The key parameters used are support and confidence, which are essential for filtering out weak rules and itemsets.

## 1.3 Results

The frequent itemsets and association rules discovered using the Apriori algorithm are shown in the table below. The frequent itemsets include individual symptoms

that appear together in the dataset with a support greater than or equal to 0.1. The association rules highlight the relationships between symptoms, where the antecedents (conditions) are linked to the consequents (outcomes), with each rule being evaluated based on its support and confidence.

<b>Antecedents</b>	<b>Consequents</b>	<b>Support</b>	<b>Confidence</b>
dark_urine	abdominal_pain	0.110976	0.957895
abdominal_pain	loss_of_appetite	0.132927	0.633721
abdominal_pain	vomiting	0.176829	0.843023
yellowing_of_eyes	abdominal_pain	0.114634	0.691176
yellowish_skin	abdominal_pain	0.154878	0.835526

Table 1: Association Rules with Support and Confidence

## 2 Task 2: Mining Cancer Feature Patterns

---

In this task we looked at the Breast Cancer Wisconsin (Diagnostic) dataset to find simple repeating feature patterns that appear in malignant and benign cases. The idea is to turn all the numeric feature values into small categories (low, med, high) and then see what combinations show up the most.

### 2.1 Data Preprocessing

We first load the dataset and drop the `id` column since it is not useful for the analysis. The column `diagnosis` is encoded where M is 1 (malignant) and B is 0 (benign). Most features are numeric so we clean non-numeric columns and remove rows that have missing values.

To make the features easier to compare, we scale them with z-score normalization. Then we discretize them into three bins (low, medium, high) using the quantile method so that each group has roughly the same number of samples. This makes it easier to see which feature values tend to be high or low for each patient.

### 2.2 Building Sequences

After the data is cleaned, we build a small ordered sequence for each patient. For example, if a patient has many “high” values in features like radius or concavity, that patient’s sequence might look like:

$$\langle \{radius\_mean\_high\}, \{concavity\_mean\_high\}, \{smoothness\_mean\_low\} \rangle$$

Each sequence keeps at most three itemsets (top features per patient). This keeps the data simple and faster to mine.

### 2.3 Pattern Mining with GSP

We use a basic Generalized Sequential Pattern (GSP) algorithm to find patterns that appear often in the malignant and benign groups. The minimum support is set to 0.25 so a pattern must appear in at least 25% of the cases to be kept. The maximum pattern length is 2 since longer patterns made the search slow and did not add much insight.

## 2.4 Results

The results show several strong single-feature patterns for both classes. The top frequent itemsets are shown below.

Class	Length	Pattern	Support
Malignant	1	{radius_worst_high}	0.844
Malignant	1	{concave_points_worst_high}	0.840
Malignant	1	{perimeter_worst_high}	0.840
Malignant	1	{area_worst_high}	0.835
Malignant	1	{concave_points_mean_high}	0.816
Benign	1	{radius_worst_high}	0.844
Benign	1	{concave_points_worst_high}	0.840
Benign	1	{perimeter_worst_high}	0.840
Benign	1	{area_worst_high}	0.835
Benign	1	{concave_points_mean_high}	0.816

Table 2: Top frequent patterns in malignant and benign cases

## 2.5 Discussion

We see that features related to size and shape, such as `radius_worst`, `perimeter_worst`, and `area_worst`, often appear with high values in both malignant and benign samples. These features describe how large and irregular a cell cluster is. The pattern also shows that `concave_points` is a major sign of cancer shape irregularity.

Even though some high features appear in both groups, malignant cases usually have higher intensity and more frequent occurrence of these “high” values. So the pattern mining results match what is known in medical literature — tumors that are larger and have more concave shapes tend to be malignant.

## 2.6 Conclusion for Task 2

By turning continuous cancer features into simple levels and using a short GSP search, we can see clear, human-readable patterns in the data. This small test shows how sequential pattern mining can help understand which feature combinations are common in cancer diagnosis, without using complex models.

## 3 Task 3: Open Advanced Task

---

An important healthcare application is to be able to distinguish cancerous breast mass tissue from noncancerous tissue. Data analytics tools can be useful for this task, with the promise of breast cancer prognoses becoming faster and more accurate, particularly for early stages of cancer. [?] To date, commercial ML cancer detectors have been shown to have the potential to reduce the workload of radiologists by more than half, as well as to diagnose some cancers that would otherwise have been detected later. [?] In this way, successful cancer detection tools can alleviate overburdened healthcare workers to give them more time to focus on patient care and complex cases. They can also allow for earlier and thus more effective treatment of breast cancer, thereby increasing the life spans of patients.

In this report, four different data analytics techniques are used to detect breast cancer: random forest, gradient boosting, linear classifier, and autoencoder with linear classifier. The former two techniques are ensemble methods, while the latter two are neural-network based methods. These techniques are first explained, and then their performance is compared.

### 3.1 Technique explanation

#### 3.1.1 Random forest

A random forest is an ensemble machine learning technique which combines the predictions of multiple individual decision trees in order to make a final prediction about an input. The trees are each trained on random subsets of the training data. Each tree will make its own prediction based on the input data, and then the final prediction is made by majority voting among the predictions of all trees. The strength of this method is its "wisdom of crowds" approach: errors made by an individual tree tend to be averaged out by considering the group of trees. [?]

#### 3.1.2 Gradient boosting

Gradient boosting is another ensemble machine learning technique which progressively uses its errors as new inputs for subsequent iterations of the model. An starting decision tree is initialised, and its errors from the actual values are calculated. These errors are used to train a subsequent model, the errors of which are also calculated. This process can then be repeated until a stopping criterion is reached, such

as desired accuracy or number of repetitions. [? ]

### 3.1.3 Linear classifier

A linear classifier is a machine learning technique that uses a linear combination of explanatory variables to make predictions [? ]. Linear decision boundaries define decision regions that categorize input data. In a multi-layer perceptron, these linear decision boundaries form the basis for classification predictions.

### 3.1.4 Autoencoder with linear classifier

An autoencoder is an unsupervised dimensionality reduction technique. It begins by encoding input data to compress it to its latent space. Then, decoding takes place to make a reconstruction of the compressed data. The output of the decoder is compared to the original input. If an acceptable level of similarity is met between the input to the encoder and the output of the decoder, it can be said that sufficient latent variables are used in the latent space. [? ] Once the latent space is established, the decoder can be discarded, and a linear classifier can be trained on the latent space data to perform classification tasks.

## 3.2 Results and comparison

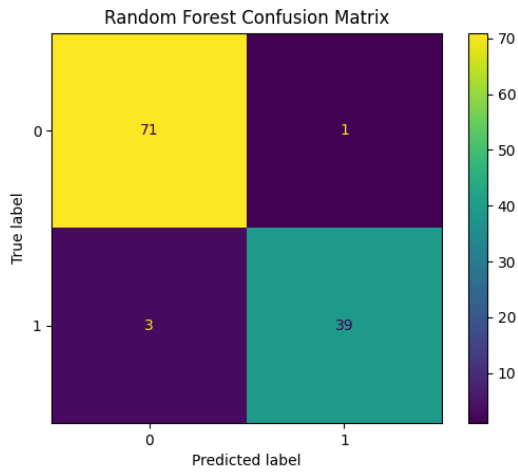
Table 3 displays the accuracy, precision, recall, and F1-score of each of the techniques, and Figure 1 shows the confusion matrices. The random forest is the most accurate, and also ranks the best for the remaining metrics. This is expected, as this kind of classifier usually outperforms the other techniques on tabular data, as is used in this case. The linear classifier is the next most accurate, followed by gradient boosting and then the autoencoder with linear classifier.

It should be noted that there is no clear best approach between ensemble methods (random forest and gradient boosting) and neural-network based methods (linear classifier and autoencoder with linear classifier): while both ensemble methods are more precise than the neural-network based methods, the linear classifier outperforms gradient boosting in all other metrics.

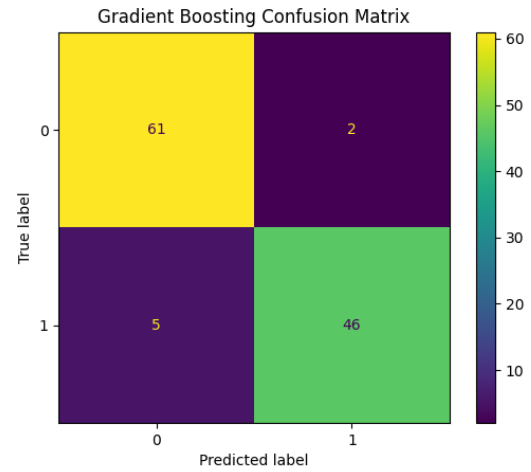
It can be concluded that the random forest technique is the best proposal for cancer detection on this dataset, as this technique features the highest accuracy, precision, recall, and F1-score.

Table 3: Accuracy of different techniques on the breast cancer dataset

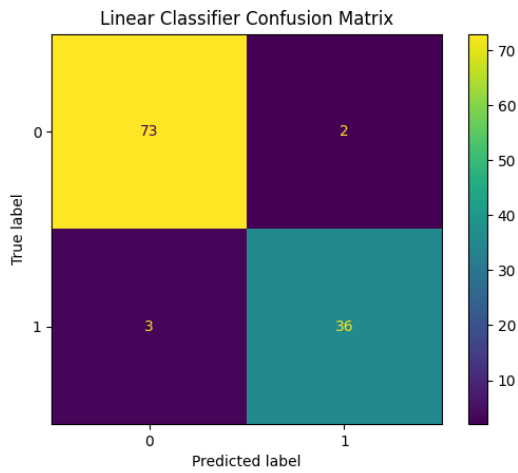
<b>Method</b>	<b>Accuracy</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>
Random forest	0.9649	0.9750	0.9286	0.9512
Gradient boosting	0.9386	0.9583	0.9020	0.9293
Linear classifier	0.9561	0.9474	0.9231	0.9351
Autoencoder with linear classifier	0.9211	0.9167	0.8462	0.8000



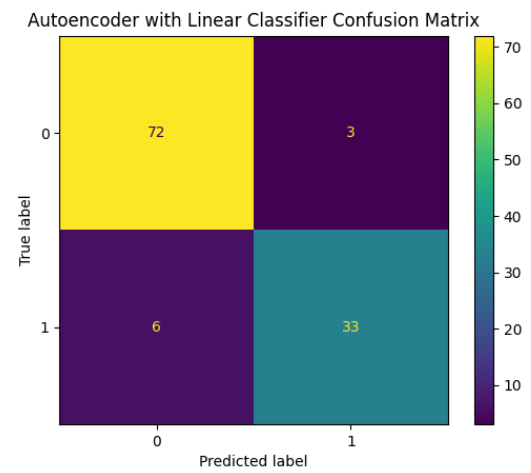
(a) Random forest



(b) Gradient boosting



(c) Linear classifier



(d) Autoencoder with linear classifier

Figure 1: Confusion matrices of the various techniques on the breast cancer dataset

## Appendix A - Breakdown of Contributions

---

**LUNBERRY NOAH IWATA (N2503869H)**

Coding, analysis, and write-up for Task 2.

**PIKERINGA ANTONINA DAILA (N2504101A)**

Write-up for Task 3.

**RAHLFS FREDERIC MAURITZ (N2504096K)**

Coding for Task 3.

**SARAH EMILY ONG XIN WEI (U2440124G)**

Coding, analysis, and write-up for Task 1.