



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING
UTM Johor Bahru

SECB3203-01(PROGRAMMING FOR BIOINFORMATIC)

Semester 01 2025/2026
Section 01

Progress 2

Faculty of Computing

Dataset:

<https://www.kaggle.com/datasets/miadul/tuberculosis-x-ray-dataset-synthetic>

Project Title:

Tuberculosis Disease Classification Using Synthetic Chest X-Ray Images and Machine Learning Techniques

Student Name	Matric Number
NATIJAH BINTI HUDA	A23CS0142
NUR IMAN BINTI MOHAMAD ZAHARI	A23CS0158
LIANA DARWISYAH BINTI AZMAN	A23CS0102

LECTURER'S NAME : DR. SEAH CHOON SEN

SUBMISSION DATE :

Table of Content

2.0 Data Collection and Preprocessing

2.1 Importing Dataset

- 2.1.1 Understanding the data
- 2.1.2 Importing and exporting data in Python
- 2.1.3 Getting started analyzing data in Python
- 2.1.4 Python packages for Data Science

2.2 Data Wrangling (Pandas/Numpy)

- 2.2.1 Identifying and handling missing values
- 2.2.2 Data formatting
- 2.2.3 Data normalization (centering/scaling)
- 2.2.4 Binning
- 2.2.5 Indicator variables

2.1 Importing Dataset

2.1.1 Understanding the data

The dataset selected for this project is a Tuberculosis (TB) Patient Dataset containing clinical and demographic information. The dataset includes information about patients' symptoms, medical history, and TB diagnosis results.

Dataset Features:

- Patient_ID: Unique identifier for each patient
- Age: Patient's age in years
- Gender: Patient's biological sex (Male/Female)
- Chest_Pain: Presence of chest pain (Yes/No)
- Cough_Severity: Intensity of coughing (Low/Medium/High)
- Breathlessness: Difficulty breathing level (numeric scale)
- Fatigue: Level of tiredness (numeric scale)
- Weight_Loss: Amount of weight reduction (numeric scale)
- Fever: Body temperature elevation (Low/Medium/High)
- Night_Sweats: Excessive sweating at night (Yes/No)
- Smoking_History: History of tobacco use (Yes/No)
- Class: TB diagnosis result (Positive/Negative)

This dataset is useful for predicting tuberculosis diagnosis based on patient symptoms and characteristics.

2.1.2 Importing and exporting data in Python

To work with this dataset effectively, we utilize Python's data manipulation capabilities. Below is the systematic approach to importing and exporting data:

```
# Importing data
```

```
import pandas as pd
```

```
import numpy as np
```

```
df = pd.read_csv('tuberculosis_xray_dataset.csv')
```

```
print(df.info())
```

```
# Exporting data
```

```
df.to_csv('processed_tb_data.csv', index=False)
```

2.1.3 Getting started analyzing data in Python

We check the dataset dimensions to understand how much data we have and the target variable to understand class balance.

```
# Check dataset size
```

```
print(f"Number of rows: {df.shape[0]}")
```

```
print(f"Number of columns: {df.shape[1]}")
```

```
# Check class distribution
```

```
print(df['Class'].value_counts())
```

Result:

- Number of rows: 20000
- Number of columns: 15

Class

- Normal : 14082
- Tuberculosis : 5918

2.1.4 Python packages for Data Science

We import the necessary libraries for data handling and preprocessing.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

2.2 Data Wrangling (Numpy)

In this phase, the dataset was prepared for analysis using NumPy. Pandas was only used to load the CSV file, while all processing steps were carried out using NumPy. This ensures the dataset is clean, consistent, and ready for modelling.

2.2.1 Identifying and handling missing values

Missing numeric values were checked using NumPy. The result showed no missing values, so all records were retained.

```
47 numeric_df = df.select_dtypes(include=[int, float])
48 missing_values = np.isnan(numeric_df.to_numpy()).sum()
49
50 print("\nTotal missing numeric values:", missing_values)
```

2.2.2 Data formatting

Several columns in the dataset contained text values such as “Yes/No”, “Male/Female”, and “Low/Medium/High” for fever levels. These values were converted into numbers using NumPy. Yes was converted to 1 and No to 0. Male was converted to 1 and Female to 0. Fever levels were converted to 0, 1 and 2. This step is important because machine learning models require numeric inputs.

```

yes_no_columns = [
    "Chest_Pain",
    "Night_Sweats",
    "Smoking_History"
]

for col in yes_no_columns:
    idx = columns.index(col)
    data[:, idx] = np.where(data[:, idx] == "Yes", 1, 0)

print("Yes/No columns converted to binary.")

```

2.2.3 Data normalization (centering/scaling)

Normalization was applied to selected numeric features such as age, cough severity, breathlessness, fatigue and weight loss. Z-score normalization was used so that all features have similar scale and do not dominate one another during modelling.

```

# -----
# 9. NORMALIZATION (CENTERING / SCALING)
# -----
# Z-score normalization for numeric columns

numeric_columns = [
    "Age",
    "Cough_Severity",
    "Breathlessness",
    "Fatigue",
    "Weight_Loss"
]

for col in numeric_columns:
    idx = columns.index(col)
    values = data[:, idx].astype(float)
    data[:, idx] = (values - values.mean()) / values.std()

print("Numeric features normalized.")

```

2.2.4 Binning

The age feature was grouped into age ranges using NumPy binning. The categories created were young, middle-aged and older. This helps to observe trends and relationships involving age more easily.

```
# -----  
# 8. BINNING (AGE GROUPING)  
# -----  
# Binning BEFORE normalization  
  
age_idx = columns.index("Age")  
age_values = data[:, age_idx].astype(float)  
  
# Age groups:  
# < 40 → Young  
# 40-60 → Middle-aged  
# > 60 → Older  
  
age_bins = np.digitize(age_values, bins=[40, 60])  
  
data = np.column_stack((data, age_bins))  
columns.append("Age_Group")  
  
print("Age values grouped into age categories.")
```

2.2.5 Indicator variables

Indicator variables were generated from categorical features such as gender, symptoms and medical history. These variables were converted into binary numeric values so that they can be processed in statistical analysis and machine learning.