



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING
UTM Johor Bahru

SECB3203-01(PROGRAMMING FOR BIOINFORMATIC)

Semester 01 2025/2026
Section 01

Progress

Faculty of Computing

Dataset:

<https://www.kaggle.com/datasets/uciml/mushroom-classification>

Student Name	Matric Number
NURUL NATALIA BINTI ROSNIZAM	A23CS0165
ADLINA NARISYA BINTI ISMAIL	A23CS0033
NAZATUL NADHIRAH BINTI SABTU	A23CS0144

2.0 Data Collection and Preprocessing

2.1 Importing Dataset

The dataset used in this project is the Mushroom Classification Dataset, contains 8,124 samples of mushroom described by 22 categorical features such as cap shape, cap color, odor, gill size, and stalk characteristics. The target column is class, which indicates whether a mushroom is edible (e) or poisonous (p).

2.1.1 Importing and Exporting Data in Python

The dataset is provided in CSV format. We import it into Python using the Pandas library:

```
import pandas as pd

# Load the dataset
data = pd.read_csv("mushrooms.csv")

# Exporting (optional, after preprocessing)
data.to_csv("mushrooms_clean.csv", index=False)
```

2.1.2 Understanding the Data

After importing, we explore the dataset to understand its structure. We display the dataset in table with attributes and their respective values from the dataset:

```
# Display first few rows
print(data.head())
```

```
# Dataset info
print(data.info())
```

```
# Summary statistics
print(data.describe())
```

Key Observations:

- The dataset has 8,124 rows and 23 columns (the categorical features and class, the target column)
- The target column, class has two categories: edible (e) or poisonous (p)
- No numerical features are present in the raw dataset

2.1.3 Getting Started Analyzing Data in Python

We check the distribution of the target variable to understand class balance:

```
print(data['class'].value_counts())
```

Result:

- Edible: 4,208 samples
- Poisonous: 3,916 samples

This shows the dataset is fairly balanced, which is good for classification tasks.

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
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
```

2.2 Data Wrangling

2.2.1 Identifying and Handling Missing Values

We check for missing values:

```
print(data.isnull().sum())
```

Observation:

The dataset contains no missing values, so no imputation or removal is required.

2.2.2 Data Formatting

Since all features are categorical, we need to encode them into numerical format for machine learning models. We use Label Encoding for simplicity:

```
from sklearn.preprocessing import LabelEncoder  
  
# Apply label encoding to all columns  
labelencoder = LabelEncoder()  
for col in data.columns:  
    data[col] = labelencoder.fit_transform(data[col])  
  
print(data.head())
```

After encoding:

- The target column class is encoded as 0 = edible, 1 = poisonous.
- Each categorical feature is converted into integer codes.

2.2.3 Data Normalization

Most features are categorical, so normalization is not strictly required. However, for models that benefit from scaled inputs (e.g., Logistic Regression, SVM), we can apply Min-Max scaling after encoding:

```
from sklearn.preprocessing import MinMaxScaler  
  
scaler = MinMaxScaler()  
scaled_data = pd.DataFrame(scaler.fit_transform(data), columns=data.columns)  
  
print(scaled_data.head())
```