UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING

SEMESTER 1, SESSION 2025/2026

PROJECT PROGRESS 2

OBESITY LEVEL CLASSIFICATION

SECB3203 : PROGRAMMING FOR BIOINFORMATICS

SECTION 02

GROUP MEMBER:

1. MUHAMMAD FARIHIN BIN SALEH	A25CS0102
2. MUHAMMAD MIRZA HASIF BIN MOHD FAHMI	A25CS0108
3. MUHAMMAD NAWFAL BIN MOHD SHAIFUDDIN	A25CS0109

LECTURER NAME : DR. SEAH CHOON SEN

GROUP : GROUP 07

TABLE OF CONTENTS

1.0 IMPORTING DATASET

1.1 Understanding the Data

The dataset used in this project is the **Obesity Levels Dataset** obtained from Kaggle, containing 2,111 records with 17 attributes. This dataset represents a synthetic yet realistic representation of individuals with different obesity levels, based on their dietary habits, physical condition, and lifestyle factors.

Feature Description:

Feature Name	Type	Description
Gender	Categorical	Individual's gender
Age	Continuous	Individual's age in years
Height	Continuous	Height in meters
Weight	Continuous	Weight in kilograms
family_history_ with_overweight	Binary	Family history of overweight (yes/no)
FAVC	Binary	Frequent high-caloric food consumption (yes/no)
FCVC	Integer	Frequency of vegetable consumption (1-3 scale)
NCP	Continuous	Number of main meals daily
CAEC	Categorical	Consumption of food between meals (Never/Sometimes/Frequently/Always)
SMOKE	Binary	Smoking habit (yes/no)
CH2O	Continuous	Daily water consumption (in liters)
SCC	Binary	Calorie intake monitoring (yes/no)
FAF	Continuous	Physical activity frequency (0-3 scale)
TUE	Integer	Technology usage time (0-2 scale)
CALC	Categorical	Alcohol consumption frequency (Never/Sometimes/Frequently)

MTRANS	Categorical	Primary transportation method
NObeyesdad (Target)	Categorical	**Obesity level classification** with 7 classes: ▪ **Insufficient_Weight** - Underweight individuals ▪ **Normal_Weight** - Healthy weight range ▪ **Overweight_Level_I** - Mildly overweight ▪ **Overweight_Level_II** - Moderately overweight ▪ **Obesity_Type_I** - Class I obesity ▪ **Obesity_Type_II** - Class II obesity ▪ **Obesity_Type_III** - Class III obesity (severe obesity)

Table 1.1: Obesity Level Dataset Feature Description

This dataset provides a comprehensive view of lifestyle factors contributing to obesity, making it suitable for developing machine learning models to classify individuals into appropriate obesity categories based on their habits and physical metrics.

1.2 Importing and Exporting Data in Python

CODING

```
# Python Libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, MinMaxScaler

# Import dataset
df = pd.read_csv('ObesityDataSet_raw_and_data_sinthetic.csv')
print(f"Dataset Shape: {df.shape}")
print(df.head())
```

The code imports the necessary Python libraries for data manipulation and loads the Obesity Levels dataset from a CSV file. The `pandas` library is used to read and handle the data, while `sklearn.preprocessing` is imported for later data transformation tasks. The `df.head()` function displays the first five rows of the dataset to provide an initial view of its structure.

OUTPUT

```
Dataset Shape: (2111, 17)
    Age  Gender  Height  Weight        CALC FAVC  FCVC  NCP  SCC SMOKE  CH2O  \
0  21.0  Female    1.62    64.0          no   no   2.0  3.0   no    no   2.0
1  21.0  Female    1.52    56.0   Sometimes   no   3.0  3.0  yes   yes   3.0
2  23.0    Male    1.80    77.0  Frequently   no   2.0  3.0   no    no   2.0
3  27.0    Male    1.80    87.0  Frequently   no   3.0  3.0   no    no   2.0
4  22.0    Male    1.78    89.8   Sometimes   no   2.0  1.0   no    no   2.0

  family_history_with_overweight  FAF  TUE       CAEC                 MTRANS  \
0                            yes  0.0  1.0  Sometimes  Public_Transportation
1                            yes  3.0  0.0  Sometimes  Public_Transportation
2                            yes  2.0  1.0  Sometimes  Public_Transportation
3                             no  2.0  0.0  Sometimes                Walking
4                             no  0.0  0.0  Sometimes  Public_Transportation

           NObeyesdad
0       Normal_Weight
1       Normal_Weight
2       Normal_Weight
3  Overweight_Level_I
4  Overweight_Level_II
```

The output confirms that the dataset contains 2,111 rows (records) and 17 columns (features). The displayed rows show sample values for each attribute, including demographic information (Age, Gender), physical measurements (Height, Weight), lifestyle factors (CALC, FAVC, etc.), and the target variable (NObeyesdad).

1.3 Getting Started Analysing Data in Python

CODING

```
# Basic dataset information
print("Dataset Information:")
print(df.info())
print("\nData Types:")
print(df.dtypes)
print("\nStatistical Summary:")
print(df.describe())
```

This section uses built-in pandas functions to explore the dataset. `df.info()` provides an overview of data types and non-null counts, `df.dtypes` lists each column's data type, and `df.describe()` computes descriptive statistics (mean, standard deviation, min, max, etc.) for numerical columns.

OUTPUT

```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2111 entries, 0 to 2110
Data columns (total 17 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Age                               2111 non-null   float64
 1   Gender                            2111 non-null   object
 2   Height                            2111 non-null   float64
 3   Weight                            2111 non-null   float64
 4   CALC                              2111 non-null   object
 5   FAVC                              2111 non-null   object
 6   FCVC                              2111 non-null   float64
 7   NCP                               2111 non-null   float64
 8   SCC                               2111 non-null   object
 9   SMOKE                             2111 non-null   object
 10  CH2O                              2111 non-null   float64
 11  family_history_with_overweight    2111 non-null   object
 12  FAF                               2111 non-null   float64
 13  TUE                               2111 non-null   float64
 14  CAEC                              2111 non-null   object
 15  MTRANS                            2111 non-null   object
 16  NObeyesdad                        2111 non-null   object
dtypes: float64(8), object(9)
memory usage: 280.5+ KB
None

Data Types:
Age                               float64
Gender                             object
Height                            float64
Weight                            float64
CALC                               object
FAVC                               object
FCVC                              float64
NCP                               float64
SCC                                object
SMOKE                              object
CH2O                              float64
family_history_with_overweight     object
FAF                               float64
TUE                               float64
CAEC                               object
MTRANS                             object
NObeyesdad                         object
dtype: object

Statistical Summary:
               Age       Height       Weight         FCVC          NCP  \
```

```
count  2111.000000  2111.000000  2111.000000  2111.000000  2111.000000
mean     24.312600     1.701677    86.586058     2.419043     2.685628
std       6.345968     0.093305    26.191172     0.533927     0.778039
min      14.000000     1.450000    39.000000     1.000000     1.000000
25%      19.947192     1.630000    65.473343     2.000000     2.658738
50%      22.777890     1.700499    83.000000     2.385502     3.000000
75%      26.000000     1.768464   107.430682     3.000000     3.000000
max      61.000000     1.980000   173.000000     3.000000     4.000000

              CH2O          FAF          TUE
count  2111.000000  2111.000000  2111.000000
mean      2.008011     1.010298     0.657866
std       0.612953     0.850592     0.608927
min       1.000000     0.000000     0.000000
25%       1.584812     0.124505     0.000000
50%       2.000000     1.000000     0.625350
75%       2.477420     1.666678     1.000000
max       3.000000     3.000000     2.000000
```

The output reveals that:

- The dataset has no missing values (all 2111 entries are complete)
- There are 8 numerical (float64) columns and 9 categorical (object) columns.
- Statistical summaries show the distribution of numerical features (e.g., average age is 24.3 years, average weight is 86.6 kg)
- This analysis confirms data quality and helps understand the range of values in each feature.

1.4 Python Packages for Data Science

CODING

```python
# Required Python Libraries
import pandas as pd       # Data manipulation and analysis
import numpy as np        # Numerical computing
from sklearn.preprocessing import LabelEncoder, MinMaxScaler  # Data preprocessing
```

This section lists the essential Python libraries imported for this project. Each library serves a specific purpose: `pandas` for data manipulation, `numpy` for numerical operations, and `sklearn.preprocessing` for data preprocessing tasks like normalization and encoding.

2.0 DATA WRANGLING (Pandas / Numpy)

2.1 Identifying and Handling Missing Values

CODING

```
# Check for missing values
missing_count = df.isnull().sum()
print("Missing Values per Column:")
print(missing_count)
print(f"Total Missing Values: {missing_count.sum()}")

# Since there are no missing values, no action needed
if missing_count.sum() == 0:
    print("No missing values found. Dataset is clean.")
```

The code checks for missing values using `df.isnull().sum()`, which counts null entries in each column. Since missing data can negatively impact machine learning models, this step is crucial for data quality assessment.

OUTPUT

```
Missing Values per Column:
Age                                 0
Gender                              0
Height                              0
Weight                              0
CALC                                0
FAVC                                0
FCVC                                0
NCP                                 0
SCC                                 0
SMOKE                               0
CH2O                                0
family_history_with_overweight      0
FAF                                 0
TUE                                 0
CAEC                                0
MTRANS                              0
NObeyesdad                          0
dtype: int64
Total Missing Values: 0
No missing values found. Dataset is clean.
```

The output shows zeros for all columns, confirming that the dataset contains no missing values. This means no imputation or removal of records is necessary, and the dataset is ready for further processing.

2.2 Data Formatting

CODING

```
# Rename columns
df.rename(columns={
    'FAVC': 'HighCaloricFood',
    'NObeyesdad': 'ObesityLevel'
}, inplace=True)

# Check data types
print("Data Types After Formatting:")
print(df.dtypes)
```

This section renames two columns for better clarity. `FAVC` becomes `HighCaloricFood` and `NObeyesdad` becomes `ObesityLevel`. Clear column names improve code readability and make the dataset more intuitive to work with.

OUTPUT

```
Data Types After Formatting:
Age                                float64
Gender                              object
Height                             float64
Weight                             float64
CALC                                object
FAVC                                object
FCVC                               float64
NCP                                float64
SCC                                 object
SMOKE                               object
CH2O                               float64
family_history_with_overweight      object
FAF                                float64
TUE                                float64
CAEC                                object
MTRANS                              object
NObeyesdad                          object
dtype: object
```

The output displays the updated data types after renaming. All original data types are preserved, and the renamed columns now appear in the DataFrame with their new names.

2.3 Data Normalisation (Centering / Scaling)

CODING

```
# Initialize MinMaxScaler for normalization (0 to 1 range)
scaler = MinMaxScaler()

# Select numerical columns to normalize
numerical_cols = ['Age', 'Height', 'Weight', 'FCVC', 'NCP', 'CH2O',
'FAF', 'TUE']

# Create normalized versions
df_normalized = df.copy()
df_normalized[numerical_cols] = scaler.fit_transform(df[numerical_cols])

# Add suffix to normalized columns for clarity
for col in numerical_cols:
    df[f'{col}_normalized'] = df_normalized[col]

print("Normalization Complete. Example:")
print(df[['Age', 'Age_normalized', 'Weight',
'Weight_normalized']].head())
```

This code normalizes numerical features using Min-Max scaling, which transforms values to a 0-1 range. This is important because features with different scales can bias machine learning models. The normalized values are stored in new columns with the suffix `_normalized`.

OUTPUT

```
Normalization Complete. Example:
    Age  Age_normalized  Weight  Weight_normalized
0  21.0        0.148936    64.0           0.186567
1  21.0        0.148936    56.0           0.126866
2  23.0        0.191489    77.0           0.283582
3  27.0        0.276596    87.0           0.358209
4  22.0        0.170213    89.8           0.379104
```

The output shows a comparison between original and normalized values for Age and Weight. For example, Age 21.0 becomes 0.1489 (scaled), and Weight 64.0 becomes 0.1866 (scaled). This demonstrates how normalization preserves the relative differences between values while bringing them to a common scale.

2.4 Binning

CODING

```
# Binning Physical Activity Frequency (FAF) into categories
bins_faf = [-0.1, 0.9, 1.9, 3.0]  # Adjusted bins to include all values
labels_faf = ['Sedentary', 'Moderate', 'Active']
df['Activity_Level'] = pd.cut(df['FAF'], bins=bins_faf,
labels=labels_faf)

# Binning Technology Usage (TUE) into categories
bins_tue = [-0.1, 0.9, 3.0]
labels_tue = ['Low_Tech_Use', 'High_Tech_Use']
df['Tech_Usage'] = pd.cut(df['TUE'], bins=bins_tue, labels=labels_tue)

print("Binning Complete:")
print(df[['FAF', 'Activity_Level', 'TUE', 'Tech_Usage']].head())
```

Binning converts continuous numerical data into categorical groups. Physical Activity Frequency (FAF) is grouped into three activity levels, and Technology Usage (TUE) is divided into two categories. This simplifies analysis and can sometimes improve model performance by reducing noise.

OUTPUT

```
Normalization Complete. Example:
    Age  Age_normalized  Weight  Weight_normalized
0  21.0        0.148936    64.0           0.186567
1  21.0        0.148936    56.0           0.126866
2  23.0        0.191489    77.0           0.283582
3  27.0        0.276596    87.0           0.358209
4  22.0        0.170213    89.8           0.379104
```

The output displays the original FAF and TUE values alongside their new categorical labels. For example, FAF=0.0 becomes "Sedentary", and TUE=1.0 becomes "High_Tech_Use". This transformation makes it easier to analyze patterns based on activity levels and technology usage.

2.5 Indicator Variables (Encoding Categorical Data)

CODING

```
# Create a copy for processed data
df_processed = df.copy()

# Identify categorical columns (excluding the newly created binned
columns)
categorical_cols = ['Gender', 'HighCaloricFood', 'CALC', 'CAEC', 'SCC',
                    'SMOKE', 'family_history_with_overweight', 'MTRANS']

# Method 1: One-Hot Encoding for multi-class nominal variables
df_encoded = pd.get_dummies(df_processed, columns=['MTRANS'],
prefix='Transport')

# Method 2: Label Encoding for binary and ordinal variables
label_encoders = {}
binary_cols = ['Gender', 'HighCaloricFood', 'SCC', 'SMOKE',
'family_history_with_overweight']

for col in binary_cols:
    le = LabelEncoder()
    df_encoded[col] = le.fit_transform(df_encoded[col].astype(str))
    label_encoders[col] = le

# Ordinal encoding for CALC and CAEC (preserving order)
calc_order = {'no': 0, 'Sometimes': 1, 'Frequently': 2}
caec_order = {'no': 0, 'Sometimes': 1, 'Frequently': 2, 'Always': 3}

df_encoded['CALC_encoded'] = df_encoded['CALC'].map(calc_order)
df_encoded['CAEC_encoded'] = df_encoded['CAEC'].map(caec_order)

# Encode target variable (Obesity Level)
obesity_order = {
    'Insufficient_Weight': 0,
    'Normal_Weight': 1,
    'Overweight_Level_I': 2,
    'Overweight_Level_II': 3,
    'Obesity_Type_I': 4,
    'Obesity_Type_II': 5,
    'Obesity_Type_III': 6
}
df_encoded['ObesityLevel_encoded'] =
df_encoded['ObesityLevel'].map(obesity_order)

print("Encoding Complete. Sample encoded data:")
```

```
print(df_encoded[['Gender', 'HighCaloricFood', 'ObesityLevel',
'ObesityLevel_encoded']].head())
print(f"\nFinal Processed Dataset Shape: {df_encoded.shape}")
```

OUTPUT

```
Encoding Complete. Sample encoded data:
   Gender  HighCaloricFood         ObesityLevel  ObesityLevel_encoded
0       0                0        Normal_Weight                     1
1       0                0        Normal_Weight                     1
2       1                0        Normal_Weight                     1
3       1                0   Overweight_Level_I                     2
4       1                0  Overweight_Level_II                     3

Final Processed Dataset Shape: (2111, 34)
```

Converts categorical text data into numerical format that machine learning models can process. The target variable (ObesityLevel) is also mapped to numerical codes for classification. The final dataset shape expands to (2111, 34) columns due to one-hot encoding creating additional binary columns for transportation methods.