UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING

SEMESTER 1, SESSION 2025/2026

PROJECT PROGRESS 4

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 PREPROCESSING FOR MODELLING

The preprocessing code prepares the dataset for machine learning by converting raw data into a suitable numerical format. First, the target variable ObesityLevel is encoded into numerical labels so it can be used by classification models. Next, all categorical features such as gender, food habits, transportation, and lifestyle variables are encoded into numeric values using label encoding.

After encoding, the dataset is split into training and testing sets with an 80:20 ratio to allow proper model evaluation. Finally, feature scaling using Min-Max normalization is applied to ensure all input variables fall within the same range, which helps improve model performance and prevents bias caused by different feature scales.

CODING

```
# PART 3: PREPROCESSING FOR MODELING
print("\n--- PREPROCESSING ---")

# 1. Encode Target Variable
le_target = LabelEncoder()
df['ObesityLevel_Encoded'] = le_target.fit_transform(df['ObesityLevel'])

# 2. Encode Categorical Features
le_features = LabelEncoder()
# Exclude the Target and the Binned columns if you don't want to use them (optional)
# Here we encode all object/category columns
cat_cols = df.select_dtypes(include=['object', 'category']).columns
for col in cat_cols:
    if col != 'ObesityLevel': # Skip raw target text
        df[col] = le_features.fit_transform(df[col].astype(str))

# 3. Split Data
X = df.drop(['ObesityLevel', 'ObesityLevel_Encoded'], axis=1)
y = df['ObesityLevel_Encoded']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Scaling (Standardization)
# Fit on Train, Transform on Test to prevent leakage
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Keep column names for later visualizations
feature_names = X.columns

print("Data Split & Scaled.")
```

OUTPUT

```
--- PREPROCESSING ---
Data Split & Scaled.
```

Now, the dataset has been properly encoded, divided into training and testing sets, and scaled. As a result, the data is now fully prepared and suitable for training machine learning models in the next stage of the project.

2.0 MODEL DEVELOPMENT

Model development is the stage where several machine learning algorithms are trained and evaluated to determine which model performs best for obesity classification. In this project, four baseline models were implemented and compared using accuracy as the evaluation metric.

Four different classification models were selected:

- Random Forest – an ensemble learning method that combines multiple decision trees to improve prediction accuracy.
- XGBoost – an advanced gradient boosting algorithm known for high performance and efficiency.
- Support Vector Machine (SVM) – a model that finds the optimal decision boundary between classes.
- Logistic Regression – a simple linear model often used as a baseline classifier.

These models were chosen to compare both simple and advanced algorithms.

Each model was trained using the training dataset (X_train, y_train) and evaluated using the testing dataset (X_test, y_test). The accuracy_score function was used to measure how well each model correctly classified obesity levels.

The accuracy results for each model were stored and printed for comparison.

CODING

```python
# PART 4: MODEL DEVELOPMENT
print("\n--- TRAINING BASELINE MODELS ---")

models = {
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "XGBoost": XGBClassifier(eval_metric='mlogloss', random_state=42),
    "SVM": SVC(kernel='linear', probability=True, random_state=42), # probability=True for Log Loss check later
    "Logistic Regression": LogisticRegression(max_iter=1000, random_state=42)
}

results = []
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    results.append({'Model': name, 'Accuracy': acc})
    print(f"{name} Accuracy: {acc:.4%}")

# Quick Visualization of Baseline
results_df = pd.DataFrame(results).sort_values(by='Accuracy', ascending=False)
plt.figure(figsize=(10, 5))
sns.barplot(x='Accuracy', y='Model', data=results_df, palette='viridis')
plt.title('Baseline Model Performance')
plt.xlim(0.6, 1.0)
plt.show()
```

OUTPUT

```
--- TRAINING BASELINE MODELS ---
Random Forest Accuracy: 94.7991%
XGBoost Accuracy: 95.5083%
SVM Accuracy: 83.6879%
Logistic Regression Accuracy: 72.8132%
```

- XGBoost achieved the highest accuracy, indicating that it is the most effective model among the tested algorithms for this dataset.
- Random Forest also performed very well, with accuracy close to XGBoost, showing strong predictive capability.
- SVM showed moderate performance, suggesting it can capture some patterns but may struggle with complex relationships.
- Logistic Regression had the lowest accuracy, which is expected since it is a simpler linear model and may not fully capture non-linear relationships in the data.

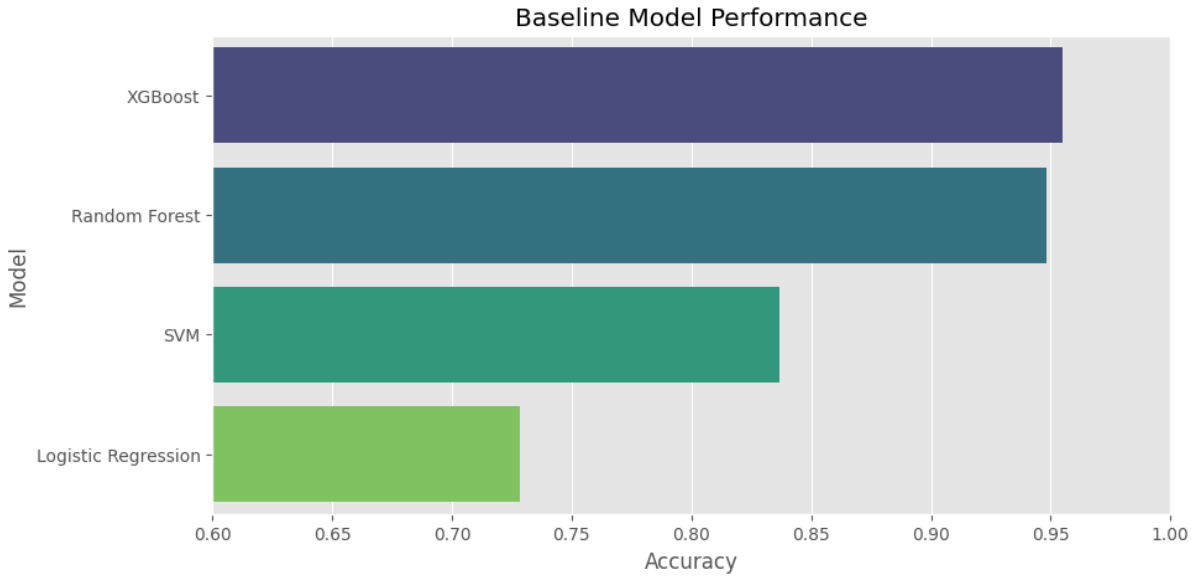

Figure 2.1: Baseline Model Performance Chart

The bar chart compares the accuracy of all four models.

- The horizontal axis represents the accuracy score.
- The vertical axis represents the machine learning models.
- Longer bars indicate higher accuracy.

From the graph:

- XGBoost appears at the top with the highest accuracy.
- Random Forest closely follows as the second-best model.
- SVM shows a noticeable drop in accuracy compared to ensemble models.
- Logistic Regression has the shortest bar, indicating the lowest performance.