EWD Project

Student: Bohdan Andoniiev, s23005
Tutor: Stanisław Horawa

POLSKO-JAPOŃSKA
AKADEMIA TECHNIK
KOMPUTEROWYCH

Goal of the experiment

The goal of the project is to build the predicting model to predict the obesity level of persons.

Data

Link to the data: https://www.kaggle.com/datasets/fatemehmehrparvar/obesity-levels/data

Features:

Gender: Feature, Categorical, "Gender"
Age : Feature, Continuous, "Age"
Height: Feature, Continuous
Weight: Feature Continuous
family_history_with_overweight: Feature, Binary, " Has a family member suffered or suffers from overweight? "
FAVC : Feature, Binary, " Do you eat high caloric food frequently? "
FCVC : Feature, Integer, " Do you usually eat vegetables in your meals? "
NCP : Feature, Continuous, " How many main meals do you have daily? "
CAEC : Feature, Categorical, " Do you eat any food between meals? "
SMOKE : Feature, Binary, " Do you smoke? "
CH2O: Feature, Continuous, " How much water do you drink daily? "
SCC: Feature, Binary, " Do you monitor the calories you eat daily? "
FAF: Feature, Continuous, " How often do you have physical activity? "
TUE : Feature, Integer, " How much time do you use technological devices such as cell phone, videogames, television, computer and
CALC : Feature, Categorical, " How often do you drink alcohol? "
MTRANS : Feature, Categorical, " Which transportation do you usually use? "
NObeyesdad : Target, Categorical, "Obesity level"

Exploratory Data Analysis – head of the dataset

```
obesity_df.head()
```

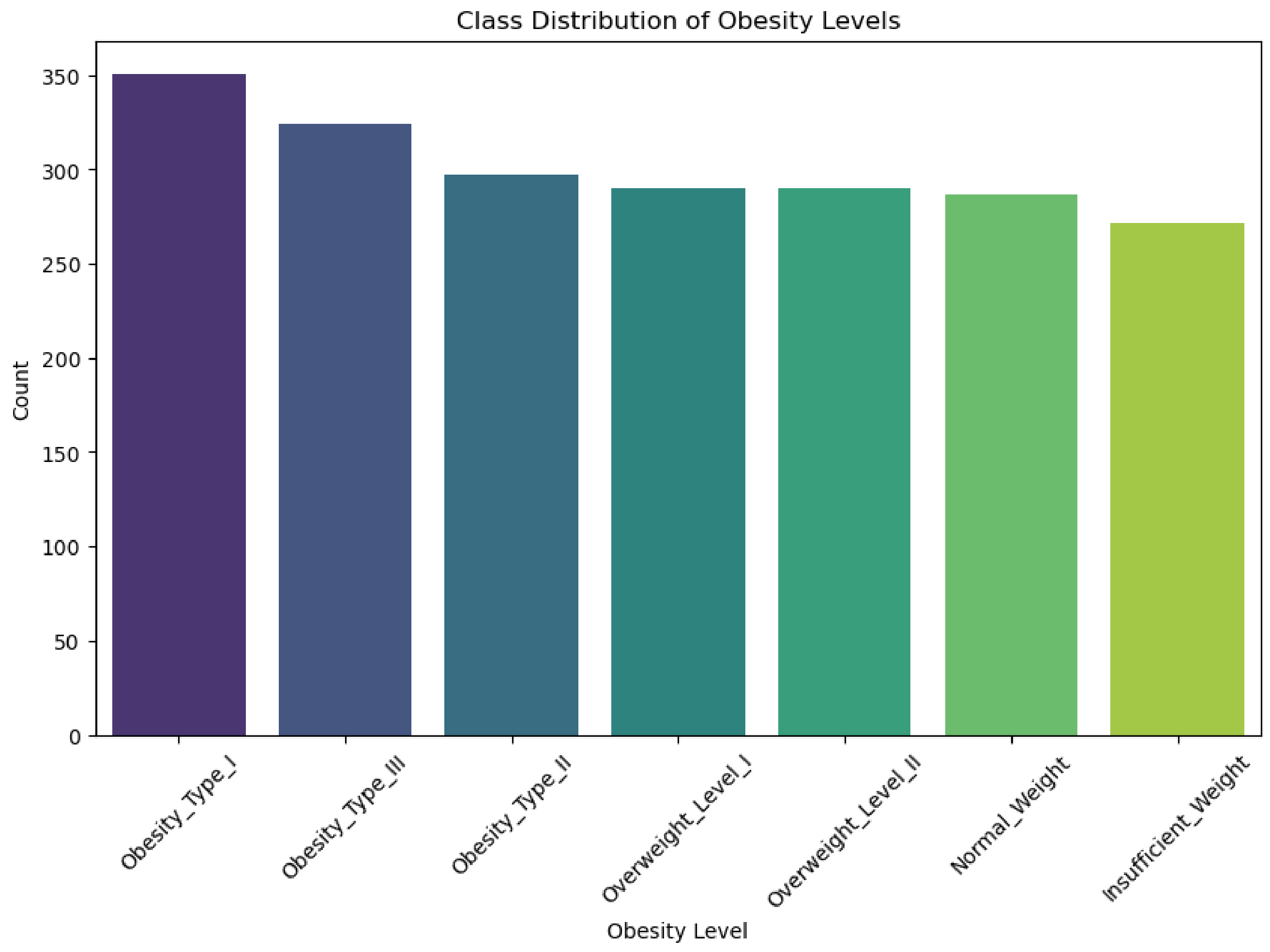
	Age	Gender	Height	Weight	CALC	FAVC	FCVC	NCP	SCC	SMOKE	CH2O	family_history_with_overweight	FAF	TUE	CAEC	MTRANS	NObeyesdad
0	21.0	Female	1.62	64.0	no	no	2.0	3.0	no	no	2.0	yes	0.0	1.0	Sometimes	Public_Transportation	Normal_Weight
1	21.0	Female	1.52	56.0	Sometimes	no	3.0	3.0	yes	yes	3.0	yes	3.0	0.0	Sometimes	Public_Transportation	Normal_Weight
2	23.0	Male	1.80	77.0	Frequently	no	2.0	3.0	no	no	2.0	yes	2.0	1.0	Sometimes	Public_Transportation	Normal_Weight
3	27.0	Male	1.80	87.0	Frequently	no	3.0	3.0	no	no	2.0	no	2.0	0.0	Sometimes	Walking	Overweight_Level_I
4	22.0	Male	1.78	89.8	Sometimes	no	2.0	1.0	no	no	2.0	no	0.0	0.0	Sometimes	Public_Transportation	Overweight_Level_II

Exploratory Data Analysis – information about dataset

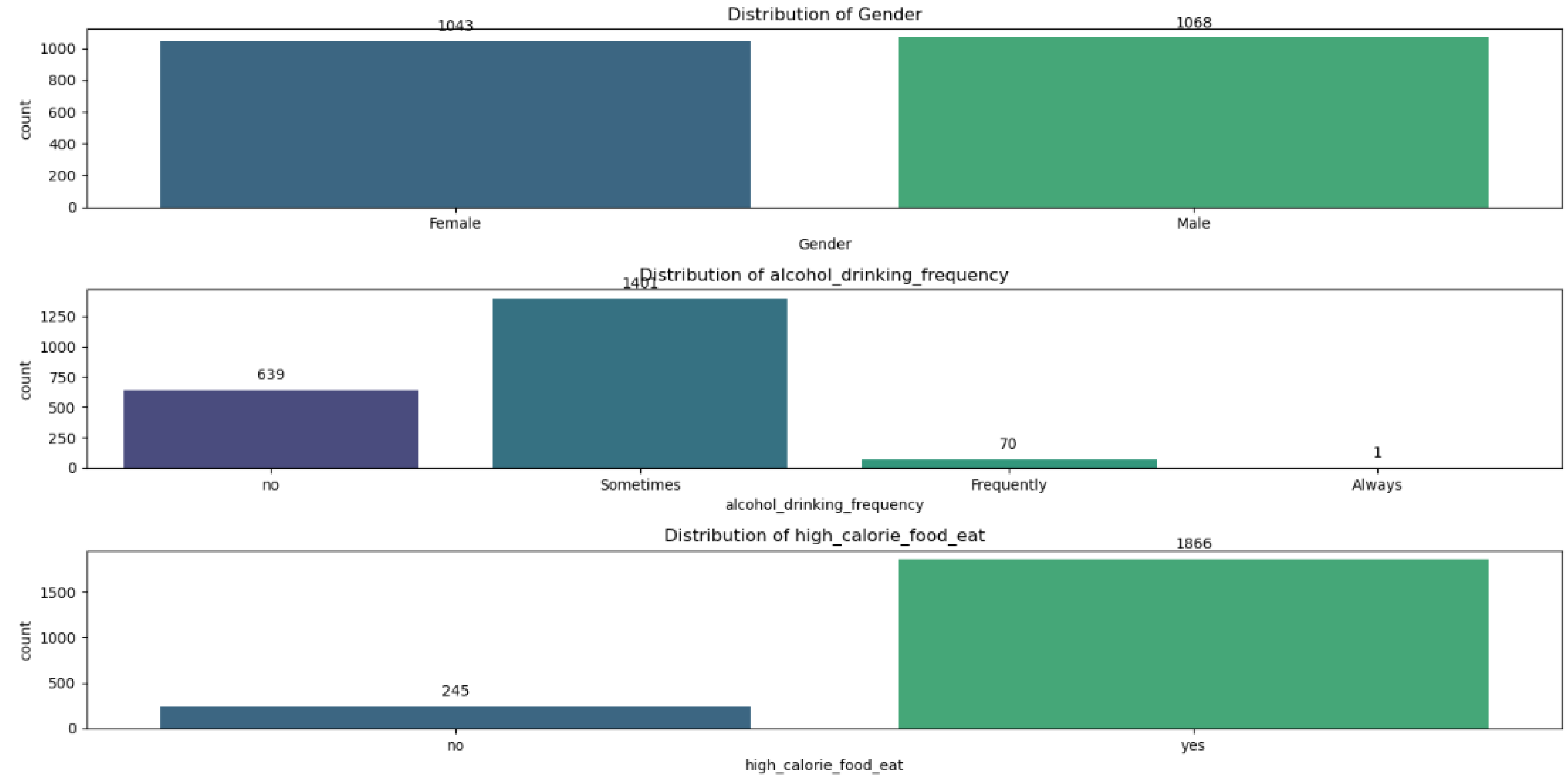
```
obesity_df.info()
```

```
<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
```

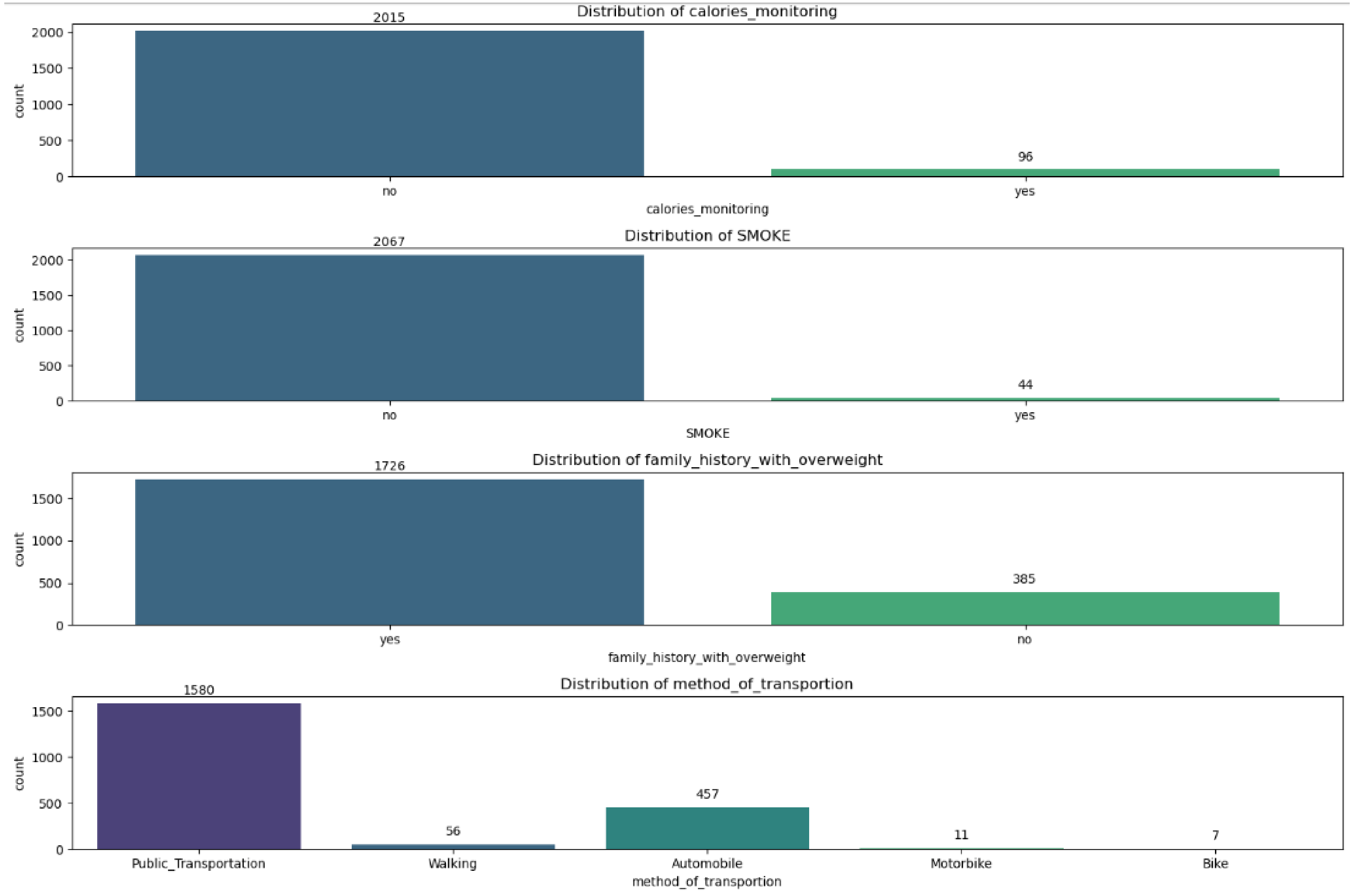
Exploratory Data Analysis – distribution of instances for every class



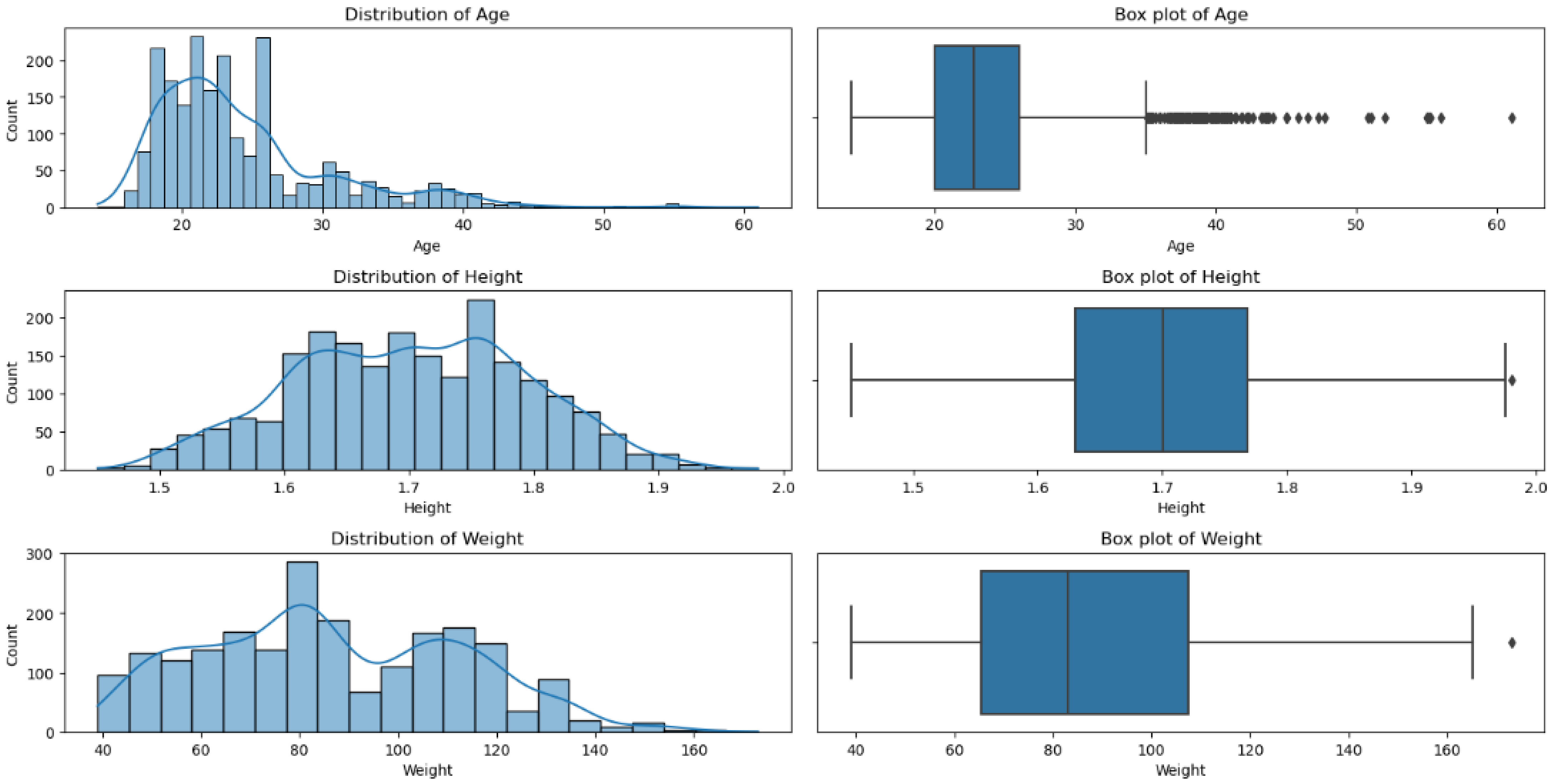
Exploratory Data Analysis – distribution of categorical features



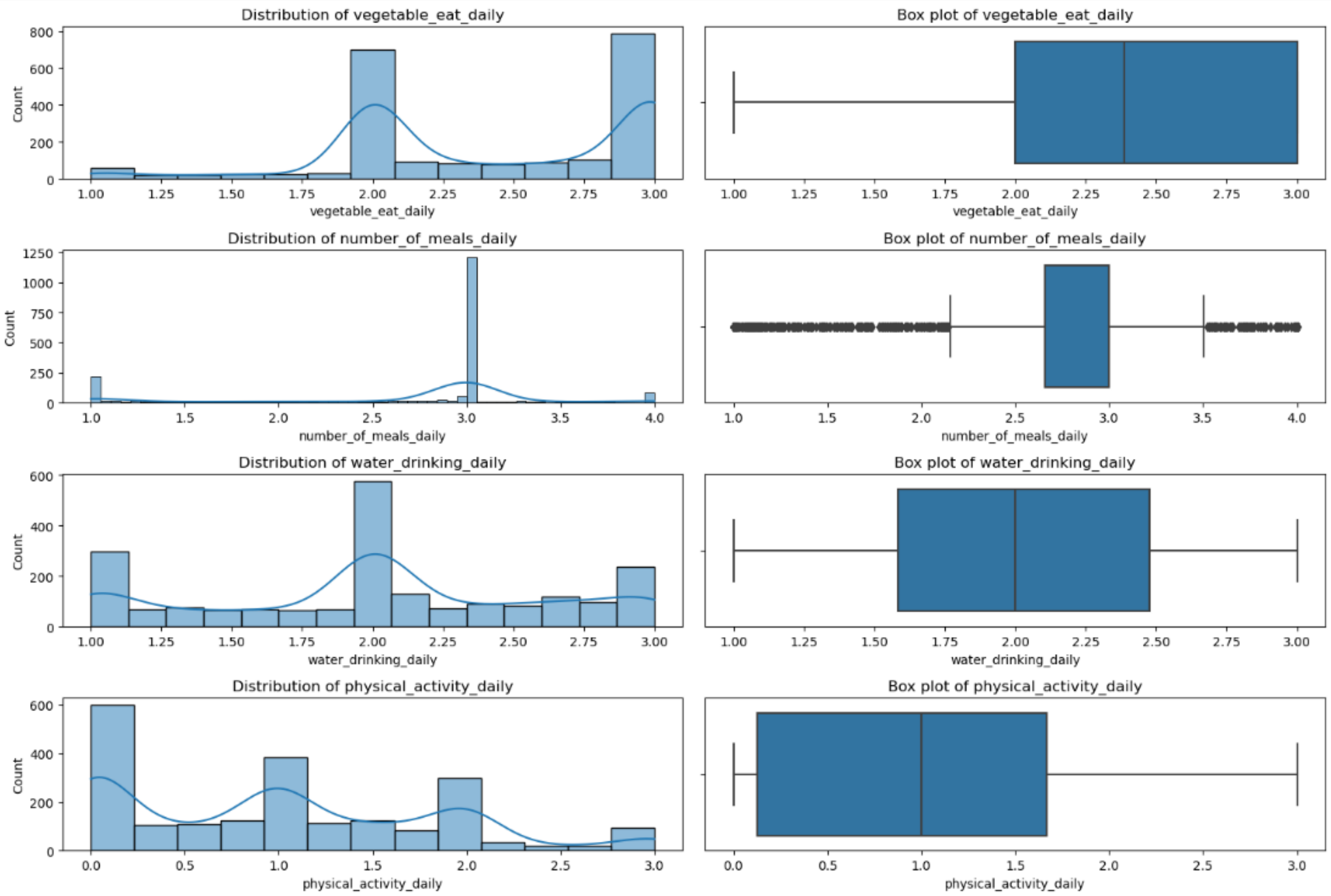
Exploratory Data Analysis – distribution of categorical features



Exploratory Data Analysis – distribution of continuous features



Exploratory Data Analysis – distribution of continuous features



Data Preprocessing

Now, we need to encode our categorical variables and scale continuous values.

The categorical variables that need encoding: gender, alcohol_drinking_frequency, high_calorie_food_eat, food_between_meals, calories_monitoring, calories_monitoring, SMOKE, family_history_with_overweight, method_of_transportatio NObeyesdad(our target variable).

```
label_encoders = {}

for col in categorical_features:
    le = LabelEncoder()
    obesity_df[col] = le.fit_transform(obesity_df[col])
    label_encoders[col] = le

# Encoding the target variable
target_le = LabelEncoder()
obesity_df['NObeyesdad'] = target_le.fit_transform(obesity_df['NObeyesdad'])
label_encoders['NObeyesdad'] = target_le

# Scaling continuous variables
scaler = StandardScaler()
obesity_df[continuous_features] = scaler.fit_transform(obesity_df[continuous_features])
```

The continuous variables that need scaling: Age, Height, Weight, vegetable_eat_daily, number_of_meals_daily, water_drinking_daily, physical_activity_daily, electronics_usage_daily.

Model training

For the training, I picked the following models

- Logistic Regression
- Decision Tree Classifier
- Random Forest Classifier
- Support Vector Machine
- K-Nearest Neighbors

For each model we will:

1) Train the model
2) Evaluate the model and prepare a classification report along with confusion matrix
3) Perform cross-validation to validate the model performance

```python
# Function to evaluate model
def evaluate_model(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred))
    cm = confusion_matrix(y_test, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot()
    plt.show()
```

Model training

```python
# Initialize variables to store the best classifier and its performance
best_classifier = None
best_classifier_name = ""
best_cv_score = 0

# Classifier names and their instances
classifiers = {
    "Logistic Regression": LogisticRegression(max_iter=10000, random_state=42),
    "Decision Tree Classifier": DecisionTreeClassifier(random_state=42),
    "Random Forest Classifier": RandomForestClassifier(random_state=42),
    "Support Vector Machine (SVM)": SVC(random_state=42),
    "K-Nearest Neighbors (KNN)": KNeighborsClassifier()
}

# Define a function to evaluate each model
def evaluate_and_cross_validate(model, X_train, X_test, y_train, y_test, model_name):
    # Train the model
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Classification report and confusion matrix
    print(f"{model_name} Classification Report:")
    print(classification_report(y_test, y_pred))
    print(f"{model_name} Confusion Matrix:")
    cm = confusion_matrix(y_test, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot()
    plt.title(f"Confusion Matrix for {model_name}")
    plt.show()

    # Cross-validation
    cv_scores = cross_val_score(model, X, y, cv=10)
    mean_cv_score = cv_scores.mean()
    print(f"{model_name} Cross-Validation Scores: {cv_scores}")
    print(f"Mean CV Score: {mean_cv_score}\n")

    return mean_cv_score
```

```python
# Evaluate each classifier and store the best one
for name, clf in classifiers.items():
    mean_cv_score = evaluate_and_cross_validate(clf, X_train, X_test, y_train, y_test, name)

    # Check if this is the best classifier
    if mean_cv_score > best_cv_score:
        best_cv_score = mean_cv_score
        best_classifier = clf
        best_classifier_name = name
```

Evaluation

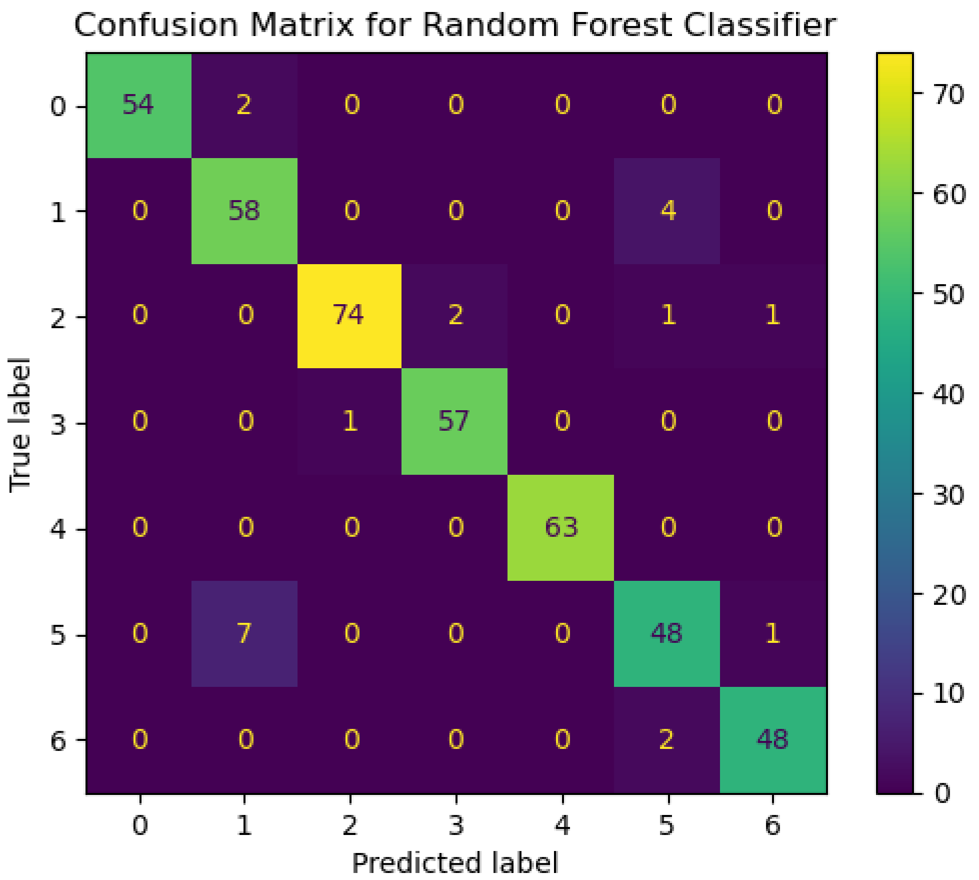
```
print(f"Best classifier: {best_classifier_name}\nAverage accuracy:{best_cv_score}")
```

```
Best classifier: Random Forest Classifier
Average accuracy:0.9470446213001876
```

Random Forest Classifier Classification Report:

	precision	recall	f1-score	support
0	1.00	0.96	0.98	56
1	0.87	0.94	0.90	62
2	0.99	0.95	0.97	78
3	0.97	0.98	0.97	58
4	1.00	1.00	1.00	63
5	0.87	0.86	0.86	56
6	0.96	0.96	0.96	50
accuracy			0.95	423
macro avg	0.95	0.95	0.95	423
weighted avg	0.95	0.95	0.95	423

Random Forest Classifier Confusion Matrix:




```
Random Forest Classifier Cross-Validation Scores: [0.73584906 0.83412322 0.99052133 0.99526066 0.98578199 0.97630332
 0.98578199 0.98578199 0.99052133 0.99052133]
Mean CV Score: 0.9470446213001876
```

Thank You for Your Attention