



Obesity Prediction with Classification Model Based on Anthropometric Data and Lifestyle Factors using Random Forest

Rizky Adha Hardiman¹, Ari Rosemalatriasari²

^{1,2}Technology Information, Industry Technology, Gunadarma University, Indonesia

Article History

Received : 25 May 2025

Revised : 29 May 2025

Accepted : 15 June 2025

Available Online

17 June 2025

Corresponding author*:

rizky@gmail.com

Cite This Article:

Rizky Adha Hardiman, & Ari Rosemalatriasari. (2025). Obesity Prediction with Classification Model Based on Anthropometric Data and Lifestyle Factors using Random Forest. *Jurnal Ilmiah Teknik*, 4(2), 123-144.

DOI:

<https://doi.org/10.56127/juit.v4i2.2084>

Abstract: Overweight and obesity have become a significant health problem worldwide, including in Indonesia. This study aims to solve this problem by developing a machine learning model based on the Random Forest Classifier algorithm. This model can be used to classify various types of obesity based on relevant variables. This study uses data from the trusted Kaggle data source to find and manage community problems with obesity and overweight. The machine learning approach allows this model to predict obesity types with high accuracy, exceeding 90%. The advantage of this approach is that it can create solutions based on a person's gender, age, weight, and height, as well as other relevant factors.

Keywords: *Machine Learning*, Model, Algorithm, *Random Forest*, Dataset, Classification, Prediction

INTRODUCTION

Obesity is a medical condition characterized by excessive accumulation of body fat, so that a person's weight is greater than normal. This condition usually occurs when the body uses or burns more energy from food and drink intake than it uses or burns. As a result of this imbalance, the body stores excess energy in the form of fat. Body Mass Index (BMI) is a common tool used to measure obesity levels. BMI is calculated as the ratio of a person's weight to height, and a value above 25.0 is considered overweight, and a value above 30.0 is considered obese according to health standards that apply worldwide. Therefore, obesity not only affects your appearance, but also has a negative impact on health, increasing the risk of type 2 diabetes, heart disease, hypertension, and other

metabolic disorders. Therefore, obesity requires serious treatment and effective prevention efforts (Annurullah et al. 2021).

In Indonesia, this problem is also a serious concern, considering the number of cases. Overweight and obesity that has continued to increase in recent decades. Indonesia, as the fourth most populous country in the world, faces complex health challenges, including overweight and obesity. Changes in lifestyle and unhealthy eating patterns. Overeating habits often occur as a result of these changes in eating patterns, especially when high-calorie foods are consumed (A. B. Putri and A. Makmun, 2021). For example, eating too much fast food causes the body to consume more energy, fat, and sugar, and fast food usually has a lot of sodium but little fiber. Consuming it more often increases the risk of obesity (D. Alfora, E. Saori, and L. N. Fajriah, 2023). In addition, a lack of fiber, vitamins, and minerals from vegetables and fruits can also cause obesity and non-communicable diseases such as diabetes, colon cancer, hypertension, stroke, and heart disease (E. Yuniarti, 2023). In addition, snacks that are usually consumed outside of main meal times are often consumed several times a day, for example in the morning, afternoon, evening, or night (D. I. Nutrition, F. Medicine, and U. Diponegoro, 2018).

Based on the problems above, this research was conducted by building a model. Machine Learning which can classify obesity type by predicting body weight type based on several factors. With the availability of datasets and the existence of these problems, the author conducted a study using the Random Forest Classifier algorithm. This study was conducted by building a machine learning model that can classify obesity types based on the problems above. by using a number of variables to predict a person's weight type.

The decision tree technique is the basis of the Random Forest algorithm, which allows for deeper data analysis. This algorithm can find complex patterns and predict outcomes based on multiple correlated variables. Its ability to handle large and varied datasets—including data with many independent variables—is a major advantage of Random Forest. In addition, Random Forest can handle the problem of multicollinearity, which is when independent variables are correlated with each other. Multicollinearity is a common problem in health data analysis. Random Forest is now a great tool for studying risk factors that cause obesity and related diseases (Suci Amaliah, Nusrang, and Aswi 2022).

RESEARCH METHOD

This scientific research writing uses the method *Machine Learning Life Cycle* (MLLC). *Machine Learning Life Cycle* (MLLC) is a process used to develop, train, and evaluate models. *Machine Learning*. The following is the MLLC concept that can be applied to the creation of this model (Gartler, Khaydarov, Klopper, and Urbas, 2021).

Research Methods

To achieve research objectives, research methods have several organized stages that are used as tools.

Model Requirements

This stage involves understanding the needs and goals of creating an obesity type prediction model. In this case, we need to understand the algorithm used for the classification method, namely the Random Forest Classifier.

Data Collection

The data was obtained from a valid and relevant source, namely the open-source dataset database site, Kaggle. The dataset used in this model is 'ObesityDataSet_raw_and_data_synthetic.csv'.

Data Preprocessing

At this stage, the data is processed to be cleaned, formatted, and prepared before being used in modeling. In this model, several data pre-processing steps are carried out such as replacing 0 values with NaN, label encoding, filling missing values with mean, and removing outliers using the IQR method.

Model Development

This stage involves creating, developing, and training a predictive model using the Random Forest Classifier algorithm. The dataset is split into training data (X_train, y_train) and testing data (X_test, y_test) using the train_test_split function. The features are also normalized using StandardScaler before training the model.

Model Evaluation:

At this stage, the model is evaluated using metrics such as confusion matrix and classification report to measure the performance and accuracy of the model. The prediction results of the model are evaluated by comparing the actual values to the test data.

Discussion Material

Overweight and Obesity

Overweight and obesity are two health conditions associated with excessive weight gain. *Overweight* refers to weight gain above normal limits, while obesity is a more serious stage where the accumulation of body fat exceeds healthy limits (Banjarnahor, Banurea, Panjaitan, Pasaribu, & Hafni, 2019). Both of these conditions have a negative impact on a person's health and can increase the risk of various chronic diseases such as diabetes, heart disease, and some types of cancer.

Machine Learning

Machine Learning is a field of computer science that deals with the development of algorithms and models that can learn and make decisions or make predictions based on data. The main goal of *Machine Learning* is to give machines the ability to learn and adapt automatically without having to be explicitly programmed. *Supervised learning* is an approach *Machine Learning* where the algorithm learns from previously labeled data or known datasets. *Random Forest*, included in the algorithm *Supervised learning*. *Random Forest* is an algorithm that combines several *decision tree* to improve accuracy and reduce *overfitting*. Every tree in *Random Forest* given a subset of training data and a random subset of features to build a decision tree. The final prediction is based on the majority of predictions from all trees. (Trehan, 2023)

Model

Model on *Machine Learning* is a mathematical representation of a system or problem to be solved. This model is created based on data given to the algorithm. *Machine Learning* to be studied. The process of creating a model involves selecting an algorithm or method *Machine Learning* that is appropriate to the type of problem to be solved. In other words, the model in the context *Machine Learning* is the result of a learning process from

data that is used to find patterns, trends, or relationships in the data so that it can be used to understand and generalize new data.

Python

Python is a high-level programming language created by Guido van Rossum and first released in 1991. The name "*Python*" taken from Guido's favorite comedy show, "*MontyPython's Flying Circus*". *Python* designed with a focus on syntax clarity, code readability, and developer productivity.

Google Colaboratory

Google Colaboratory is a servicecloud computing provided by *Google* to write, run, and share code*Python*. al using the environment *Jupyter Notebook* which is interactive inbrowser web.

Kaggle

Kaggle is a popular competition platform and data resource among practitioners.data science And Machine Learning. *Kaggle* become a popular platform among the communitydata science because it provides access to rich data resources, challenging competitions, and a means to interact with other practitioners. This allows users to learn, practice, and compete in a collaborative and competitive environment.

RESULT AND DISCUSSION

Model Requirements

Model requirements refer to the steps that must be followed and fulfilled to create, train, and use an obesity type prediction model with an algorithm.*Random Forest Classifier*. At this stage, several random decision trees (called estimators) will be formed using the bagging technique. Algorithm*Random Forest* takes the concept by building many decision trees randomly and combining the prediction results from each tree to produce the final prediction.

Finally, the algorithm*Random Forest* used to predict body weight type based on new data entered by the user through input. This new data is then processed and predictions are made using a previously trained model.

Data Collection (Data Collection)

The dataset used in this model is "ObesityDataSet_raw_and_data_synthetic.csv", obtained from the site *open-source* Kaggle. This dataset consists of 2112 records and 17 attributes. The dataset contains a number of features such as gender, age, weight, height, and other relevant features. Each row in the dataset represents an individual with various attributes, and the columns in the dataset represent the features used for model training and prediction.

Load the dataset

In this model, the “load” function is used to load the dataset in the CSV file named 'ObesityDataSet_raw_and_data_synthetic.csv'. Then the “data” variable is defined as the dataset. Next, Pandas (a Python library for data analysis) is used to read the CSV file and save it in the form of a Dataframe (df).

Table 1. Labels, Label Meanings, and Values based on Dataset Attributes

No	Label	Meaning of Labels	Mark
1	Gender	Gender	1: Man 0: Female
2	Age	Age	User age in number form
3	Height	Height (in cm)	User's height in centimeters
4	Weight	Body Weight (in kg)	User's body weight in kilograms
5	family_history with_overweight	Family History of Obesity	1: Of 0: No
6	CVF	Frequency of Food Consumption High Calories	1: Of 0: No
7	FCVC	Frequency Consumption Vegetables per Day	A value between 1 and 3 indicating the frequency of vegetable consumption per day.
8	NCP	Main Meal Amount per day	A value between 1 and 4 that indicates the number of main meals per day
9	CAEC	Food Consumption in between Main Meal	0: No, 1: Sometimes, 2: Often, 3: Always
10	SMOKE	Smoking Habit	1: Of 0: No
11	CH2O	Daily Water Consumption	A value between 1 and 3 indicating the level of water consumption per day.
12	SCC	Monitoring Calorie Consumption	1: Of 0: No
13	FAF	Frequency of Physical Activity	A value between 0 and 3 indicating the level of frequency of physical activity.

No	Label	Meaning of Labels	Mark
14	TUE	Time Spent on Technology Devices	A value between 0 and 2 indicating the time spent on device technology
15	CALC	Alcohol Consumption	0: No, 1: Sometimes, 2: Often, 3: Always
16	MTRANS	Transportation Used	1: Automobile, 2: Motorbike, 3: Bike, 4: Public Transportation, 5: Walking
17	NObeyesdad	Types of Obesity	Depending on the model's prediction results after inputting other attribute values, these values will be generated by the model based on the user's input. enter

Examining Information and Statistics of a Dataframe

Function “df.info()” is used to display summary information about a DataFrame, including the number of rows and columns, the data type of each column, and whether there are any missing values in the DataFrame. The output will contain information such as index, column name, number of non-null data, data type, and memory usage. This information helps in understanding the size and structure of the data in the DataFrame and the type of data contained in each column. If there are any missing values, researchers must handle the data before conducting further analysis.

Checking Missing Values in Each Dataframe Column

```
[3] # Check information and statistics
print(df.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2111 entries, 0 to 2110
Data columns (total 17 columns):
 #   Column                                     Non-Null Count  Dtype  
---  --
 0   Gender                                   2111 non-null   object  
 1   Age                                       2111 non-null   float64 
 2   Height                                   2111 non-null   float64 
 3   Weight                                   2111 non-null   float64 
 4   family_history_with_overweight          2111 non-null   object  
 5   FAVC                                     2111 non-null   object  
 6   FCVC                                     2111 non-null   float64 
 7   NCP                                       2111 non-null   float64 
 8   CAEC                                     2111 non-null   object  
 9   SMOKE                                    2111 non-null   object  
10  CH2O                                     2111 non-null   float64 
11  SCC                                       2111 non-null   object  
12  FAF                                       2111 non-null   float64 
13  TUE                                       2111 non-null   float64 
14  CALC                                     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
```

Figure 1. Missing Values in the Dataset

The code in this model is used to check if there are missing values in each column of the DataFrame df. The method used is `isnull()` to detect the presence of null (NaN) values in the DataFrame, and then the `sum()` operation is performed to calculate the number of null values in each column.

Target Class Data Visualization (NObeyesdad)

The graph generated from the code in this model is a bar plot. This bar plot is used to visualize the distribution of target class data (NObeyesdad) in the dataset.

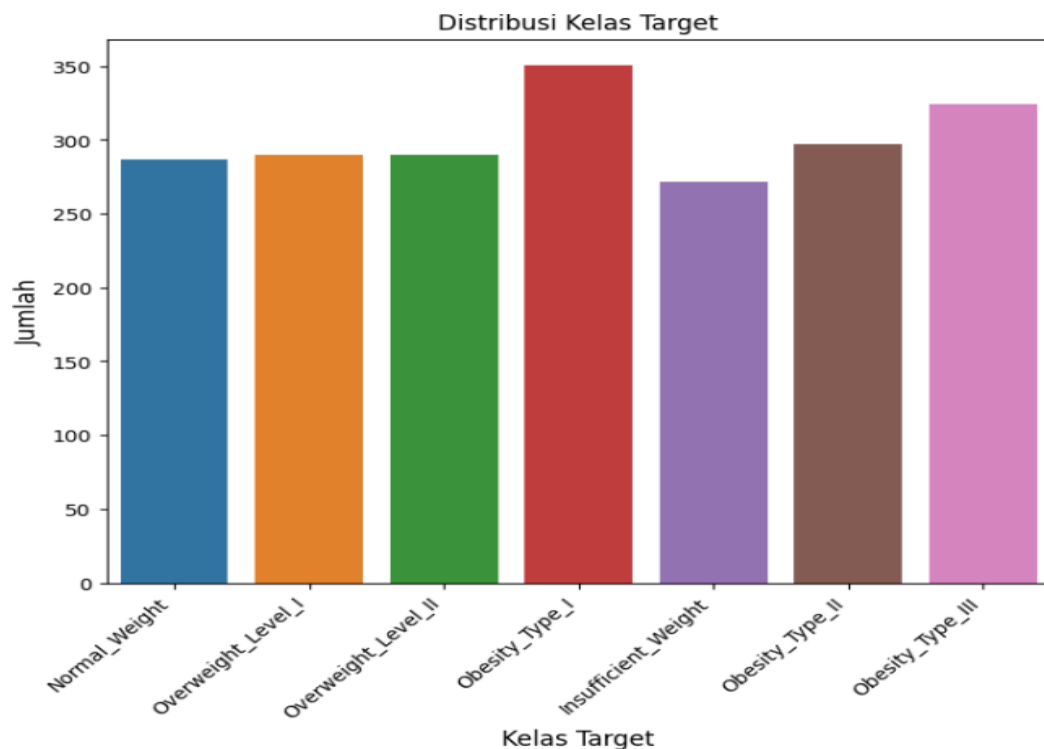


Figure 2. Visualization of Prediction Target Class

Graphics Description:

- The x-axis (horizontal axis) shows the target class or data label (e.g., "Insufficient_Weight", "Normal_Weight", "Obesity_Type_I", and so on).
- The y-axis (vertical axis) shows the amount of data contained in each target class.

This graph provides a visual representation of the distribution of data within each target class. By looking at this graph, the author can see the comparison of the amount of data between different target classes. If there is an imbalance in the distribution of target classes (class imbalance), this graph can help the author identify target classes that may have less data compared to other target classes.

Data Preprocessing

Data Purification

```
[7] # Replace 0 values with NaN
    df['FAF'] = df['FAF'].replace(0, np.NaN)
    df['TUE'] = df['TUE'].replace(0, np.NaN)
```

Figure 3. Data Cleaning

Replacing 0 values with NaN in the “FAF” and “TUE” features. In the dataset, there are several 0 values in the “FAF” and “TUE” features, which indicate that the data is missing or irrelevant. So that these 0 values do not affect the calculation and prediction results, it is necessary to replace them with NaN (Not a Number) values that indicate missing or invalid data.

Label Encoding

```
[8] # Perform label encoding
    encoder = LabelEncoder()
    df['NObeyesdad'] = encoder.fit_transform(df['NObeyesdad'])
    df['Gender'] = encoder.fit_transform(df['Gender'])
    df['family_history_with_overweight'] = encoder.fit_transform(df['family_history_with_overweight'])
    df['CAEC'] = encoder.fit_transform(df['CAEC'])
    df['FAVC'] = encoder.fit_transform(df['FAVC'])
    df['SCC'] = encoder.fit_transform(df['SCC'])
    df['SMOKE'] = encoder.fit_transform(df['SMOKE'])
    df['CALC'] = encoder.fit_transform(df['CALC'])
    df['MTRANS'] = encoder.fit_transform(df['MTRANS'])
```

Figure 4. Label Encoding

Perform label encoding on some features. Some features in the dataset such as “Gender”, “family_history_with_overweight”, “CAEC”, “FAVC”, “SCC”, “SMOKE”, “CALC”, and “MTRANS” have categorical values, for example "Male" and "Female" for “Gender” or "Yes" and "No" for therefore it is necessary to convert these categorical values into numeric values using Scikit-learn’s LabelEncoder. This is done to facilitate further data processing in model building.

Impute Data

```
[9] # Fill missing values with means based on NObeyesdad classes
    for i in range(7):
        faf_mean = df[df['NObeyesdad'] == i]['FAF'].mean()
        tue_mean = df[df['NObeyesdad'] == i]['TUE'].mean()
        df.loc[(df['NObeyesdad'] == i) & (df['FAF'].isnull()), 'FAF'] = faf_mean
        df.loc[(df['NObeyesdad'] == i) & (df['TUE'].isnull()), 'TUE'] = tue_mean
```

Figure 5. Data Imputation

Data imputation is a technique to fill missing values or NaNs with values derived from existing data. In this case, it is necessary to fill the NaN values in the “FAF” and “TUE” features with the average based on the “NObeyesdad” class. After replacing the 0 values with NaNs in the “FAF” and “TUE” features, it is necessary to fill these NaN values with the average values of “FAF” and “TUE” based on the “NObeyesdad” class. This means that the average of “FAF” and “TUE” will be calculated for each “NObeyesdad” class (e.g. “Insufficient_Weight”, “Normal_Weight”, etc.) and fill the NaN values with the appropriate average based on their class. This step is done to retain relevant information in the dataset and prevent bias in the prediction due to the deletion of data that has NaNs.

Outlier Handling

```
[10] # Remove outliers using IQR method
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
df = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axis=1)]
```

Figure 6. Outlier Handling

Remove outliers using the IQR method. Outliers are data that have extreme values and are very different from other values in the dataset. Outliers can affect the results of the model and reduce the quality of predictions. To identify and remove outliers, the IQR (Interquartile Range) method is used. IQR measures the distribution of data by calculating the difference between the first quartile (Q1) and the third quartile (Q3). Data that falls outside the range of $Q1 - 1.5 * IQR$ to $Q3 + 1.5 * IQR$ are considered outliers and will be removed from the dataset. This helps ensure that the model is not influenced by unrepresentative or irrelevant data.

Feature and Label Preparation

```
[11] # Prepare features and labels
features = df[['Gender', 'Age', 'Weight', 'Height', 'family_history_with_overweight', 'FAVC',
              'FCVC', 'NCP', 'CAEC', 'SMOKE', 'CH2O', 'SCC', 'FAF', 'TUE', 'CALC', 'MTRANS']]
label = df['NObeyesdad']
X = features
y = label
```

Figure 7. Features and Labels

The "Feature and Label Preparation" process aims to separate the features that will be used as input to train the model and the labels (targets) that will be predicted by the model.

- The first step is to select the features to be used in model training. In this model, the features are selected from the "df" dataset by taking the following columns: "Gender", "Age", "Weight", "Height", "family_history_with_overweight", "FAVC", "FCVC", "NCP", "CAEC", "SMOKE", "CH2O", "SCC", "FAF",
- "TUE", "CALC", and "MTRANS". All these features will be inputs (independent variables) for the model.
- Next, choose the label that will be used as the target (dependent variable) in model training. The label chosen in the code in this model is "NObeyesdad", which is the obesity class. 'NObeyesdad' will be the target that the model wants to predict.
- After that, the features are stored in the variable "X" and the labels in the variable "y". The variable "X" will contain all the features that will be used as input, while the variable "y" will contain the obesity class that is the target for prediction.

Model Training

Dataset Division

```
[12] # Split the dataset into train and test sets
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
```

Figure 8. Dataset Division

The "Dataset Splitting" process aims to divide the dataset into two parts: training data and test data. Training data will be used to train the model, while test data will be used to test the performance of the trained model.

By dividing the dataset into training data and test data, the model can be trained using the training data, and then tested using the test data to measure how well the model can predict on new data that has never been seen before. This helps ensure that the developed model can generalize well to data that has never been seen before.

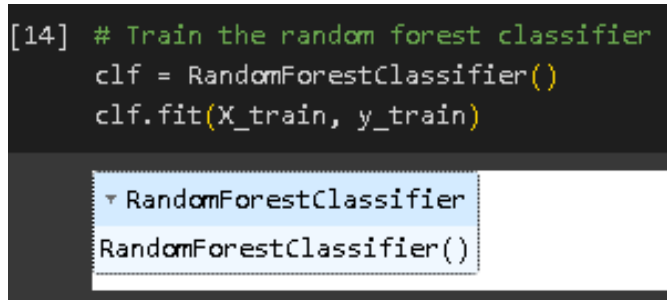
Feature Scaling

```
[13] # Scale the features
      scaler = StandardScaler()
      X_train = scaler.fit_transform(X_train)
      X_test = scaler.transform(X_test)
```

Figure 9. Feature Scaling

The “Feature Scaling” process is an important step to rescale the features in a dataset so that they have a mean of 0 and a variance of 1. This is useful because some features may have a larger range of values than others, and this difference in scale can affect the performance of the model.

Model Training



```
[14] # Train the random forest classifier
      clf = RandomForestClassifier()
      clf.fit(X_train, y_train)
```

▼ RandomForestClassifier
RandomForestClassifier()

Figure 10. Model Training

In the “Model Training” stage, an algorithm is used *Random Forest Classifier* to build a classification model that will be used to classify obesity types based on previously processed anthropometric features.

Steps for training a model using *Random Forest Classifier* are as follows:

- Import class *RandomForestClassifier* from *library* Scikit-learn:
 - from sklearn.ensemble import *RandomForestClassifier*
- Create a *clf* object from the *RandomForest* class *Classifier*:
 - *clf* = *RandomForestClassifier*()
- Train the model using the training data (*X_train* and *y_train*) using the *fit* method:
 - *clf.fit*(*X_train*, *y_train*)

The *fit* method is used to train the model with training data. The model will learn from the features and labels contained in the training data to be able to perform the classification correctly.

Prediction

In the “Prediction” stage, the trained model uses *Random Forest Classifier* used to make predictions on test data (*X_test*). This process aims to test the performance of the

model that has been built by measuring the extent to which the model is able to classify obesity types based on anthropometric features on data that has never been seen before.

The steps for predicting and evaluating model performance are as follows:

- Import the `confusion_matrix` and `classification_report` functions from *library* Scikit-learn:
- `from sklearn.metrics import confusion_matrix, classification_report`
- Make predictions on the test data (`X_test`) using the trained model (`clf`):

```
[15] # Make predictions
      pred = clf.predict(X_test)
```

Figure 11. Test Data Prediction

The `predict` function is used to predict classes on test data. The prediction result will be an array containing prediction labels for each data on `X_test`.

Prediction Results

In the "Show Prediction Results" stage, the user is asked to enter attribute values that are used as input to predict obesity types based on the previously trained classification model. This process is done by converting user input into a form that matches the scale that has been applied to the previous training data.

The steps and code explanation for this model are as follows:

- Reading input from user
Users are asked to enter attribute values such as gender, age, weight, height, history of obesity, habit of consuming high-calorie foods, frequency of vegetable consumption per day, number of main meals per day, etc. Each of these inputs is stored in an appropriate variable (e.g., gender, age, weight, etc.) with an appropriate data type (e.g., integer or float).
- Converts user input into the appropriate form
Once the user input is obtained, the data needs to be transformed into a form that matches the previously processed training data. This is done by forming a list containing all the attributes entered by the user. For example, the `input_data` variable contains a list of attribute values entered by the user.
- Make predictions using trained models

The prediction function is used to predict the type of obesity based on the input provided by the user. This process is done by calling the predict function on the previously trained classification model (clf). The prediction results are stored in the prediction variable.

- Displaying prediction results

After the prediction is complete, the prediction result in the form of a numeric label (prediction result class) is stored in the predicted_class variable. Furthermore, the numeric label is changed to the type of obesity that matches the label value using obesity_types. If the numeric label value does not match the label in obesity_types, then the label "Unknown" will be given.

- Prediction result output

The predicted obesity type results based on user input are displayed to the user by printing the predicted label and the predicted obesity type.

Thus, the user can see the predicted obesity type based on the anthropometric data that has been entered. This process helps in providing information about the weight and obesity categories that may suit the physical characteristics of the user.

Model Evaluation

The following code aims to evaluate the performance of a classification model using a confusion matrix and various evaluation metrics such as accuracy, error rate, precision, recall, and F1-score.

The steps taken are as follows:

Calculating Confusion Matrix (cm)

The confusion matrix is calculated by comparing the actual label (y_test) with the predicted label (pred) from the model. The confusion matrix is an N x N matrix (where N is the number of classes, i.e. 7) that represents the number of data correctly or incorrectly classified in each class. The value cm[i, j] represents the number of data that are actually class i and predicted by the model as class j.

Showing Confusion Matrix in Heatmap Form.



Figure 12. Confusion Matrix Heatmap Graph

The calculated confusion matrix is displayed in the form of a heatmap using *library* seaborn and matplotlib. Heatmap provides a more intuitive visualization of the distribution of data in each cell in the confusion matrix.

Calculating Metrics for Each Class

Next, various evaluation metrics are calculated for each class in the confusion matrix.

```
[17] # Calculate metrics for each class
num_classes = len(encoder.classes_)
for i in range(num_classes):
    tp = cm[i, i]
    fn = np.sum(cm[i, :]) - tp
    fp = np.sum(cm[:, i]) - tp
    tn = np.sum(cm) - (tp + fp + fn)

    accuracy = (tp + tn) / (tp + tn + fp + fn)
    error = (fp + fn) / (tp + tn + fp + fn)
    precision = tp / (tp + fp)
    recall = tp / (tp + fn)
    f1_score = 2 * (precision * recall) / (precision + recall)

    print(f"\nMetrics for class {i}:")
    print("Accuracy:", accuracy)
    print("Error Rate:", error)
    print("Precision:", precision)
    print("Recall:", recall)
    print("F1-score:", f1_score)
```

Figure 13. Calculating Metrics for Each Class

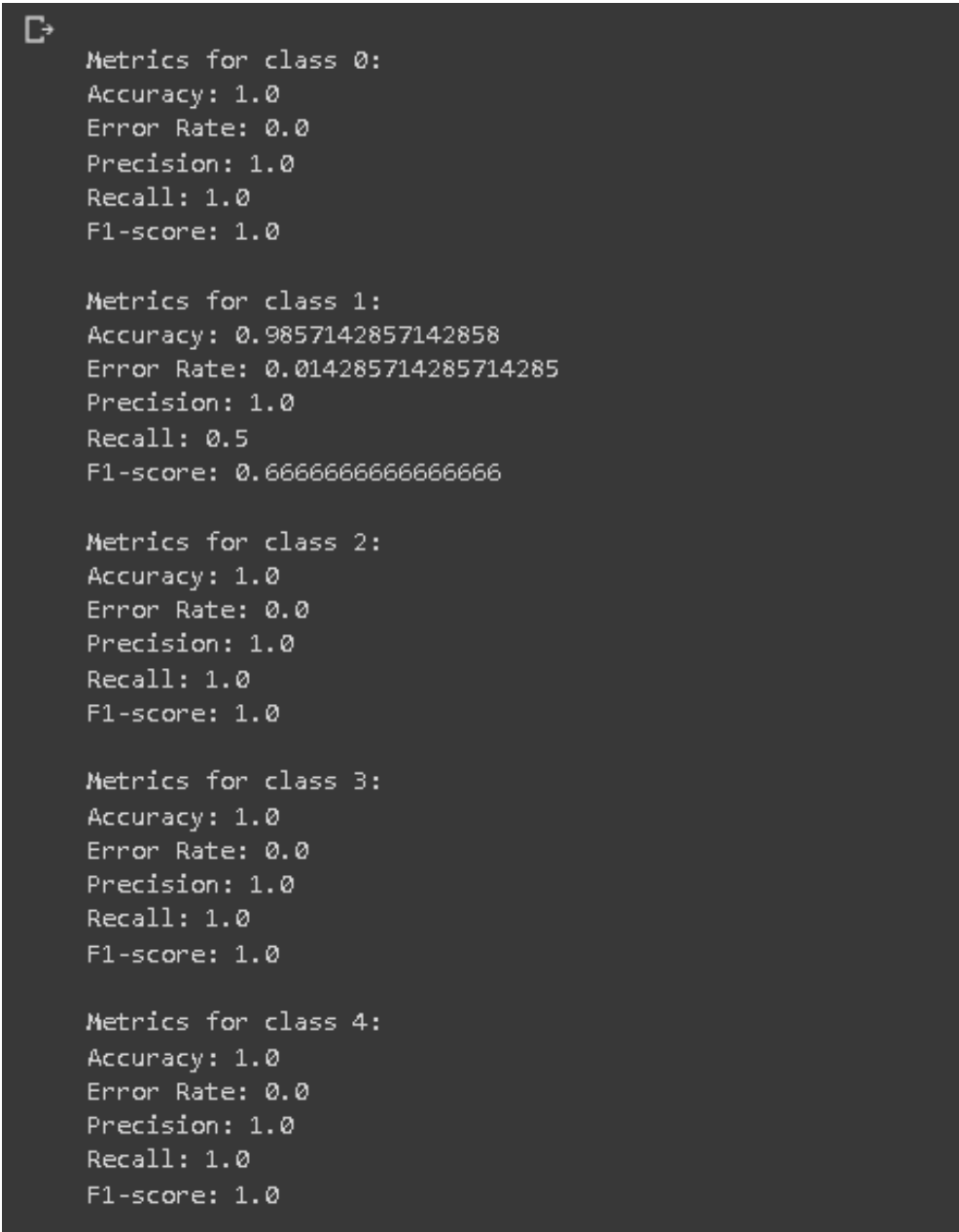


Figure 14. Results of Metric Calculation for Each Class

By doing the steps above, the author can measure and understand the extent to which the model performs in classifying each class as a whole. Here is an example of a confusion matrix in class 1:

Table 2. Confusion Matrix in Class 1

		Predicted Label	
		Class 1	Class is not 1
True Label	Class 1	3	1
	Class is not 1	0	136

Based on the given code, class 5 and class 6 in the matrix are not included in the output because they do not appear in the test set `y_test`. If class 5 and class 6 are not present in the test set, then there will be no true positive (TP), false positive (FP), true negative (TN), or false negative (FN) values in the confusion matrix `cm`. As a result, the metrics (accuracy, error rate, precision, recall, and F1 score) for these classes cannot be calculated.

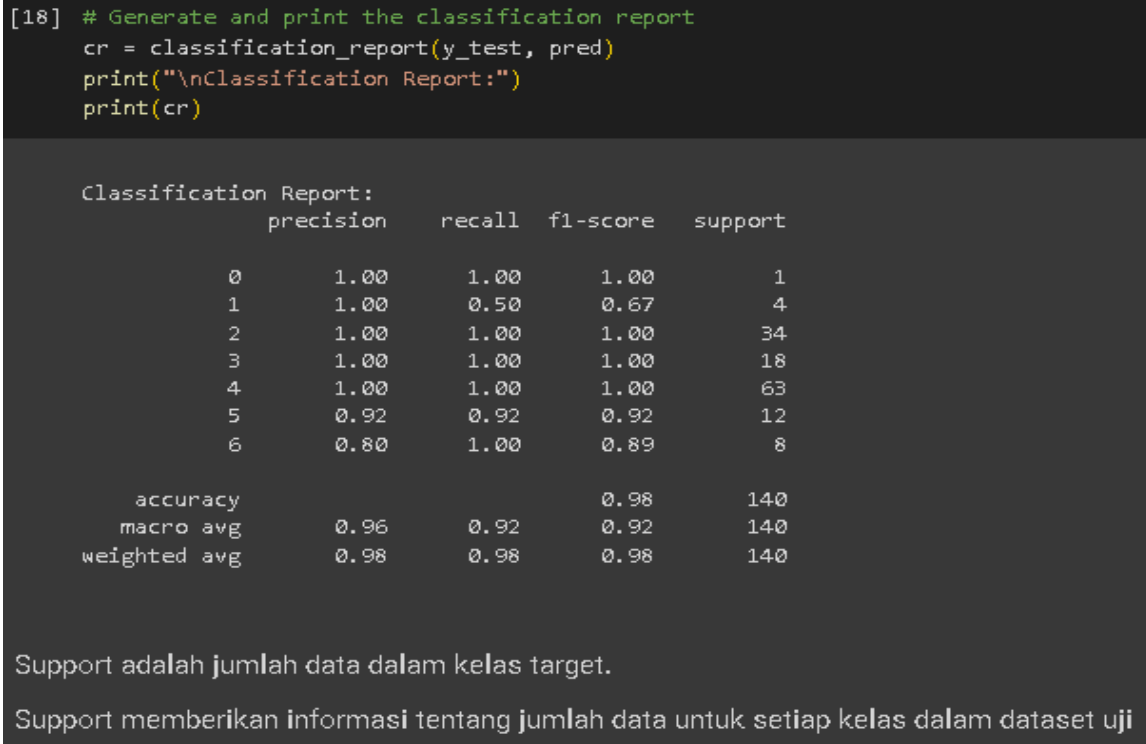


Figure 15. Classification Report

Model Deployment (Model Presentation)

New Data Input

In the previous model, the features used were limited to five, namely "age", "height", "weight", "family_history_with_overweight", and "favc". For the output, it uses one label, namely "NObeyesdad" as the target class.

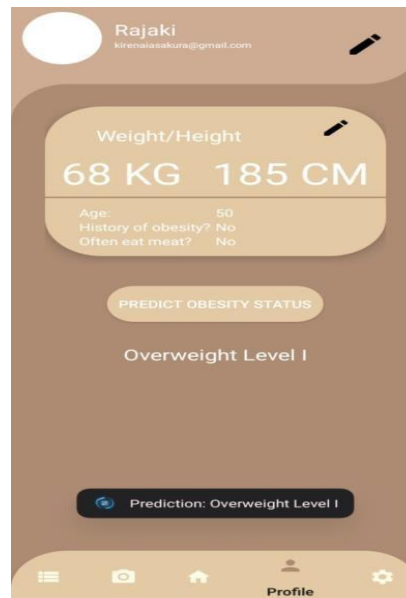


Figure 16. Model Presentation in Android Application

However, the model is deployed using cloud computing techniques and implemented on an android application. In this author's developed model, the features used are all attributes contained in the dataset, which is 16 features and one label for the same target class as the previous model. The difference lies in the presentation of the model because the results of this development were not deployed due to the limitations of the author's learning scope when following an independent study program, so the prediction of the weight type was only carried out directly on Google Colaboratory.

Saving Input Results in Drive

The following code is an implementation to save the prediction results into a text file (.txt) with a name that is adjusted based on the specified MODELNAME.

Here is an explanation of the code steps:

- In the first line, we define the MODELNAME variable using the #@param comment to give it a default value and a data type of "string". The given value will be used as the name for the output text file.

•

```
MODELNAME : "Kyda"
```

Figure 17. Modelname Definition

```
[ ] MODELNAME = "Kyda" #@param {type:"string"}
```

Figure 18. Input Modelname

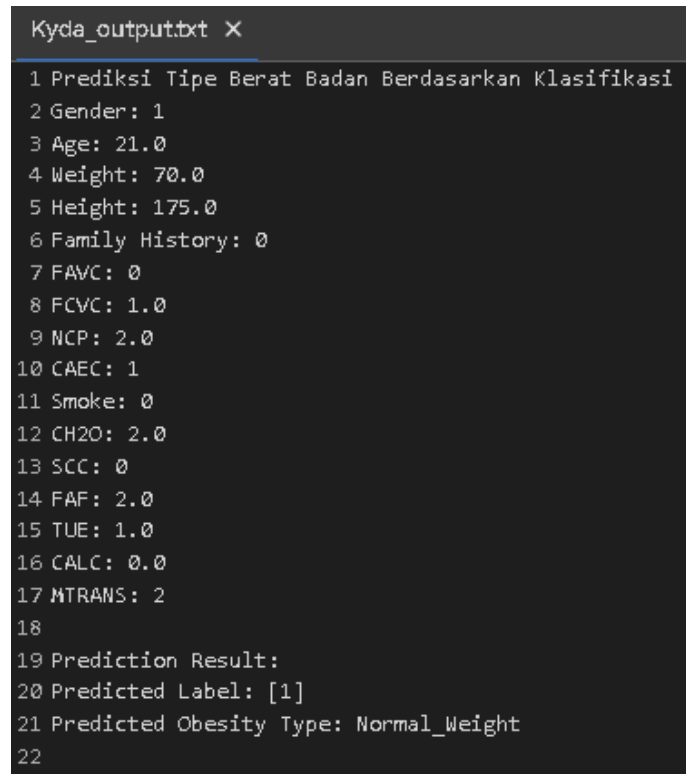
- After that, we create a variable `output_filename` which is a combination of the directory `"/content/drive/MyDrive/"` with the values `MODELNAME` and `"_output.txt"`. This output file will later store the prediction results.

```
[ ] # Create the output file name with MODELNAME
output_filename = f"/content/drive/MyDrive/{MODELNAME}_output.txt"
```

Figure 19. Saving Prediction Results

- Then, we open a text file with a name that matches the `output_filename` that was created earlier. The file is opened with "w" (write) mode which means we will write data into the file. The file will be automatically closed after the process is complete thanks to the use of "with open(...) as output_file:".
- Next, we write some information such as "Prediction of Body Weight Type Based on Classification" and various input data such as gender, age, weight, height, obesity history, and so on into the `output_file` file using the `write()` method.
- After writing the input information, we also call the `prediction()` function with `input_data` and `output_file` as arguments. The `prediction()` function will make predictions based on the input data and write the prediction results to the `output_file` file.
- Next, let's see how the `prediction()` function works:
 - The `prediction()` function will predict body weight based on classification using a previously trained model (`RandomForest.Classifier`).
 - The input data provided in the form of an `input_data` list will be transformed using a scaler to match the form of the data used for model training.
 - Weight prediction is then done by calling the `predict()` function from the `RandomForest model.Classifier` on the input data that has been changed.
 - The prediction results in the form of prediction labels will be printed using the `print()` statement.

- Next, the predicted label value will be used to search for the type of obesity based on the predicted label index in the obesity_types list.
- The obesity type result will be printed using the print() statement.
- After the entire process is complete, the output_file file will be closed and the prediction results will be saved in a file with the name according to the specified MODELNAME.



```

1 Prediksi Tipe Berat Badan Berdasarkan Klasifikasi
2 Gender: 1
3 Age: 21.0
4 Weight: 70.0
5 Height: 175.0
6 Family History: 0
7 FAVC: 0
8 FCVC: 1.0
9 NCP: 2.0
10 CAEC: 1
11 Smoke: 0
12 CH2O: 2.0
13 SCC: 0
14 FAF: 2.0
15 TUE: 1.0
16 CALC: 0.0
17 MTRANS: 2
18
19 Prediction Result:
20 Predicted Label: [1]
21 Predicted Obesity Type: Normal_Weight
22
  
```

Figure 20. Output Results in .txt Format

So, the code above is used to predict body weight based on input data and save the prediction results into a text file with a name that matches the specified MODELNAME.

CONCLUSION

Based on the discussion of the analysis stages and discussions in the previous chapter, it was concluded that this obesity prediction classification model has been successfully developed by utilizing all the features contained in the database used. Use of the algorithm *Random Forest Classifier* for this model, it is the right choice after comparing it with several other algorithms used for classification type models by looking at the evaluation metrics.

Based on the test results, this model can predict a person's obesity type based on the results of the model training test using the dataset, so that if the classification process carried out with new data using the data training test shows good accuracy, then the prediction results will also be good and can approach the actual data results from the dataset.

This model has achieved accuracy and validation accuracy of more than 93%, which is a very good result and does not cause overfitting, so if this model is deployed to the application, it is expected that this model can be accurate in classifying the type of obesity based on user input.

REFERENCES

- A, B., M, Z., & R, S. (2013). Python To Learn Programming. Journal Of Physics: Conference Series, 1-5.
- A. B. Putri And A. Makmun, "Diet Patterns Against Obesity,"Indonesia. J. Heal., Vol. Xx, No. Xx, Pp. 68–76, 2021, Doi:10.33368/Inajoh.V2i1.39.
- Annurullah, Gifita Alifa, Maulyda Shakeela Jasmine, Nandita Ardrafritri Saraswati, And Yabsutur Rizka. 2021. "Risk Factors Of Obesity In Office Workers: A Systematic Review."Tambusai Health Journal 2(2): 80–88.
- Banjarnahor, R. O., Banurea, F. F., Panjaitan, J. O., Pasaribu, R. S., & Hafni, I. (2019). Risk Factors For Overweight And Obesity In .Trophico: Tropical Public Health Journal Faculty Of Public Health, 35-45.
- Cholossodin, I., Sutrisno, Soebroto, A. A., Hasanah, U., & Febiola, Y. I. (2019).Ai, Machine Learning & Deep Learning (Theory & Implementation).Poor.
- Cutler, A., Cao, X., Polat, K., & Zhang, J. (2020). Confusion Matrix. Retrieved From Science Direct: <https://www.sciencedirect.com/topics/engineering/confusion-matrix>
- Dinata, R. K., & Hasdyna, N. (2020).Machine Learning. English: Oxford University Press.
- D. Alfora, E. Saori, And L. N. Fajriah, "The Effect Of Fast Food Consumption On Adolescent Nutrition,"Florona J. Health Science., Vol. 2, No. 1, Pp. 43–49, 2023, Doi: 10.55904/Florona.V2i1.688.
- D. I. Nutrition, F. Medicine, And U. Diponegoro, "Snack Energy Intake, Sleep Duration And Quality In Obesity And Non-Obesity Adolescents," Vol. 7, No. 2010, Pp. 147–154, 2018.
- E. Yuniarti, "The Relationship Between Vegetable And Fruit Consumption And Adolescent Obesity In Padang City,"J. Healthy Independent, Vol. 18, No. 1, Pp. 137–145, 2023, Doi: 10.33761/Jsm.V18i1.974.
- Elwirehardja, G. N., Suparyanto, T., & Pardamean, B. (2022).Introduction To Machine Learning Concepts. Yogyakarta: Instiper Press.

- Gartler, M., Khaydarov, V., Klopper, B., & Urbas, L. (2021). The Machine Learning Life Cycle In Chemicaloperations – Status And Open Challenges. 18.
- Ministry Of Health Of The Republic Of Indonesia. (2018, 2).Obesity Epidemic.Retrieved From Ministry Of Health:
https://P2ptm.Kemkes.Go.Id/Uploads/N2vaaxixzgzwfpe11v1rfdqq3Zrzz09/2018/02/Factsheet_Obesitas_Kit_Informasi_Obesitas.Pdf
- Ministry Of Health Of The Republic Of Indonesia. (2021, 7 1).How To Measure Body Mass Index (Bmi) / Normal Body Weight?Retrieved From Ministry Of Health:
<https://P2ptm.Kemkes.Go.Id/Infographic-P2ptm/Obesitas/Bagaimana-Cara-Mengukur-Indeks-Massa-Tubuh-Imt-Berat-Badan-Normal>
- Luo, T., Chen, S., Xu, G., & Zhou, J. (2013).Trust-Based Collection View Prediction.New York: Springer Science+Business Media.
- Mu'alim, F., & Hidayati, R. (2022). Implementation Of The Methodrandom Forest For Majoring.Jupiter, 116-125.
- Muhamad, I. M., Wardana, S. A., Wanto, A., & Windarto, A. P. (2022). Machine Learning Algorithm For Determining Production Prediction Models.Journal Of Informatics, Electrical And Electronics Engineering, 126-134.
- Mulia, S. (2022, 8 31).Getting To Know Matplotlib For Data Visualization, Check Out Explanation Following This! Retrieved From Idmetaphor:
<https://Idmetafora.Com/News/Read/1254/Mengenal-Matplotlib-Untuk-Do-Visualisasi-Data-Simak-Penjelasan-Berikut-Ini.Html>
- Nugroho, Y. S., & Emiliyawati, N. (2017). Springer Science+Business Media. Journal Of Electrical Engineering, 24-29.
- Puji, A. (2023, 7 10).Formula For Measuring Body Mass Index (Bmi) In Adults. Retrieved From Hellosehat: <https://Hellosehat.Com/Nutrisi/Cara-Hitung-Indeks-Massa-Tubuh/>
- Ramadhan, A., & Susetyo, B. (2019). Application Of Classification Methodsrandom Forest In Identifying Important Factors In Assessing The Quality Of Education.Journal Of Education And Culture, 169-182.
- Roihan, A., Sunarya, P. A., & Rafika, S. A. (2019). Utilization Of Machine Learning In Various Fields: .Ijcit (Indonesian Journal On Computer And Information Technology), 75-82.
- Septiani, R., & Raharjo, B. B. (2017). Fast Food Consumption Patterns, Physical Activity And Hereditary Factors On The Incidence Of .Public Health Perspective, 262- 269.
- Siswoyo, B., Encep, & Firman. (2019). Machine Learning Bankruptcy Prediction.Informatics Buffer Journal, 8-15.
- Trehan, D. (2023, 7 2). Why Choose Random Forest And Not Decision Trees. Retrieved From Towardsai: <https://Towardsai.Net/P/Machine-Learning/Why-Choose-Random-Forest-And-Not-Decision-Trees>
- Wuryani, N., & Agustiani, S. (2021).Random Forest Classifierfor Ct Scan Image-Based Covid-19 Sufferer Detection.Journal Of Computer Engineering Amik Bsi, 187-193.