ISSN 2355-0082

Ensemble Learning - Random Forest Algorithm to Classify Obesity Level

Tesalonika Abigail[1], Marlinda Vasty Overbeek[2]
[1,2] Dept. of Informatics, Universitas Multimedia Nusantara, Tangerang, Banten
[1]tesalonika.abigail@student.umn.ac.id, [2]marlinda.vasty@umn.ac.id

Accepted 5 July 2024
Approved 15 November 2024

***Abstract*— Obesity is one of the serious global health problems caused by excessive accumulation of body fat. According to the World Health Organization (WHO), the prevalence of obesity has tripled in the last 40 years, with 650 million out of 1.9 billion overweight adults suffering from obesity. Obesity is a non-communicable disease that increases the risks of more dangerous diseases, such as heart disease and cancer. Therefore, early detection of obesity level is crucial. Currently, Body Mass Index (BMI) serves as a measurement indicator, but it tends to overestimate obesity for those with high muscle mass and vice versa, making it ineffective as it only relies on height and weight, without considering body composition and daily activities. To address this limitation, a Random Forest model was developed and selected based on the results of model selection, feature selection, and hyperparameter tuning. This model improved accuracy by 1.4% and was implemented into a web-based system for classifying obesity levels. Evaluation of the model yielded Precision, Recall, F1-Score, and Accuracy of 97%, 97%, 97%, and 96.8%, respectively. Based on these results, it can be concluded that this system is highly effective in classifying obesity levels.**

***Index Terms*— Feature Selection; Hyperparameter Tuning; Model Selection; Obesity Level Classification; Random Forest**

I. INTRODUCTION

Obesity is a serious global health problem caused by excessive accumulation of fat [1]. The World Health Organization (WHO) states that the global obesity rate has tripled in the last 40 years, with over 1.9 billion adults suffering from overweight and 650 million of those classified as obese [2]. Obesity, primarily caused by unhealthy lifestyles, increases the risk of diseases such as type-2 diabetes, cancer, heart disease, hypertension, and stroke [2,3]. Therefore, obesity should not just be seen as a lifestyle issue but as a disease posing significant health risks. Consequently, early detection of obesity levels is crucial [1].

BMI is a common indicator for measuring obesity, but it tends to overestimate obesity in individuals with high muscle mass and underestimate it in those with low muscle mass [4]. This is because BMI relies solely on height and weight, ignoring body composition and daily activities. Alternative approaches by classifying obesity levels based on lifestyle and dietary habits with machine learning which is increasingly popular in medical studies for classification, clustering, and anomaly detection can be used [1,5].

Previous research using the K-Nearest Neighbor (KNN) algorithm to classify obesity levels with the "PIMA Indian Diabetes" dataset achieved a 78.98% accuracy rate, with a recommendation to use other algorithms to improve model accuracy [1]. Another study comparing Random Forest, Decision Tree, Support Vector Machine, and KNN for diabetes classification, found that Random Forest is the best algorithm to classify diabetes with 97.5% accuracy, 97.4% precision, 96.6% recall, and 97% f1-score [6]. These results show that Random Forest performs exceptionally well in classifying diabetes.

To improve model accuracy, this research will use the Random Forest algorithm to classify obesity levels using the "PIMA Indian Diabetes" dataset as the development from the first previous study. This choice is based on its excellent performance in diabetes classification from the second previous study which is closely related to obesity itself. Random Forest itself is widely used in medical classification due to its ability to handle mixed datasets, high efficiency, excellent accuracy, and low error rates [5,7,8]. The expectation is that it will similarly excel in obesity classification just like how it achieved an exceptionally good performance in classifying diabetes.

During development, a model selection by doing feature selection techniques and hyperparameter tuning will be implemented to select the best model with the highest accuracy. The selected model will then be integrated into a system for accurate early detection of obesity levels. Therefore, this research will focus on designing and developing an obesity levels classification system using the Random Forest algorithm to provide high-accuracy classification results.

II. LITTERATURE REVIEW

A. Obesity

Indonesia obesity prevalence has been increasing, with rates reported by the *Badan Penelitian dan Pengembangan Kesehatan* in 2019 rising from 8.6% in

2007 to 11.5% in 2013, and 13.6% in 2018 [9]. Obesity has thus become a significant health issue in Indonesia, contribute to the dual burden of disease [10]. While infectious diseases remain a leading cause of health problems and deaths, non-communicable diseases are also on the rise, necessitating focused prevention and management efforts [10].

B. Machine Learning

Machine learning, a subset of Artificial Intelligence (AI), empowers systems or applications to autonomously learn from data, enhancing their reliability and predictive accuracy without human intervention [11]. One of its main approaches is supervised learning [12]. Supervised learning itself it utilized when datasets include outputs or classes, rely on accurate class assignments for effective model training [12]. The learning process involves dataset partitioning into training and testing subsets, model training to learn feature-class relationship, and evaluation of model performance using testing data [13].

Classification is a common supervised learning method, involve teaching algorithms to categorize data into predefined classes [11,13]. This classification encompasses dual and multiple class distinctions, with binary outcomes in the former and multi-category outcomes in the latter [11]. Random Forest stands out as one of the top algorithms for classification tasks [11].

C. Decision Tree

Decision tree learning is a predictive model approach commonly used in machine learning [14] and is often employed for classification tasks as it doesn't require extensive information [15]. It takes the form of a tree with subdivisions that repeatedly divide data into smaller subsets based on specific criteria [16]. It is constructed using nodes and branches with root as the top node of the tree for tree construction [16]. Branches stemming from the root, which still have child branches, are called internal nodes thats connected to the root, other internal nodes, and leaf nodes through branches [16]. Leaf nodes, on the other hand, are branches with no further branches, representing the decision outcomes of the decision tree. Fig 1 illustrates the structure of a decision tree [16].

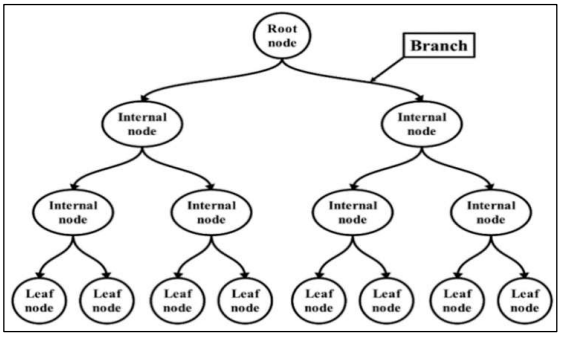

Fig. 1. Decision Tree Structure [16]

D. Random Forest

Random Forest is an ensemble algorithm [17], stemming from the development of decision trees. It operates by constructing multiple decision trees for prediction and making final decisions based on the most voted outcome [18]. Below are the steps in creating a random forest model [19].

1. Selecting n random samples from the training data to be used in the dataset bootstrapping process.
2. Selecting m random features with m < p, where p is the total number of features, to select the best feature in each node as a node splitting separator with optimization criteria, such as "gini", "entropy", or "log_loss".
3. The second process will continue until the minimum number of observations at the node is reached.
4. The entire process above will be repeated until k decision trees are formed. From each decision tree, one class will be obtained and used in the voting process to determine the final classification class

The overview/ illustration of how the Random Forest algorithm works can be seen in Fig 2.

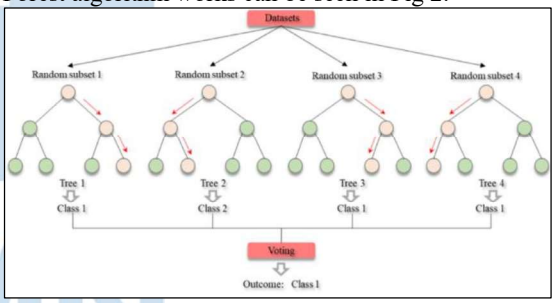

Fig. 2. Flow of the Random Forest Algorithm [20]

Next, here's the formula for calculating the Random Forest algorithm to obtain the final classification result. The formula represents the predicted class of the b-th tree for data x [21], which will be used in the voting process by considering the classification results from the 1st tree to the Bth tree to determine the final classification outcome based on the majority vote.

$$\hat{C}_{rf}^{B} = majority\ vote\{\hat{C}_b(x)\}_1^B \quad (1)$$

However, this algorithm inherently cannot achieve optimal performance if relying solely on one value for each hyperparameter used, as it may not be the optimal value. This is because its performance heavily depends on the hyperparameters used [22]. To address this limitation, hyperparameter tuning can be conducted to find the most optimal hyperparameter set to be used for the model construction through trial and error testing of all hyperparameter combinations [22].

E. Randomized Search

Randomized Search is a hyperparameter tuning technique that utilizes randomness or probability, because it explores the search space randomly using only a few randomly selected hyperparameter settings without following any specific pattern or sequence [23]. Thus, Randomized Search can quickly evaluate several important combinations to achieve optimal results, making computational time more efficient.

The Randomized Search algorithm is evaluated using cross-validation methods [23], which divide the data into k subsets called folds. The evaluation process is repeated k times, using 1 fold as validation data to test the model after being trained on the other k-1 folds as training data. Below is the pseudocode illustrating the workflow of Randomized Search [23].

Algorithm 1: Randomized Search Algorithm

```
Data: Input data
Result: Output Result
Initializing criterion, max_depth, n_estimators, max_features,
 min_samples_split, and max_leaf_nodes hyperparameters;
Initializing estimators, search space, number of iterations, and
 number of k-folds;
while Stop criteria is not fulfilled do
    Randomly select parameters from the search space;
    Split the dataset into K-Folds evenly;
    for each fold k in K-Fold do
        Set k as the validation data, the rest as training data;
        Training the data using the estimator with the
         hyperparameter combination;
        Evaluate model performance;
        Calculate the average score from each fold;
    end
end
Return the best hyperparameters;
```

F. Confusion Matrix

The confusion matrix is a metric used to assess the performance of classification outcomes and is easy to interpret because the model's performance can be directly observed from the distribution of values in the confusion matrix [24]. A good model will only have values along the diagonal line of the matrix and vice versa [24]. Confusion matrix itself is divided into binary and multi-class classification [25].

The binary classification confusion matrix is a 2x2 matrix with positive and negative labels for each actual class and classification outcome [14]. Each cell represents the outcome between the actual and predicted class [24]. There are four terms used to represent the classification outcomes [14]. True Positive (TP) represents the number of classifications predicted as positive by the model that match reality [26]. True Negative (TN) represents the number of classifications predicted as negative by the model that match reality [26]. False Positive (FP) represents the number of classifications predicted as positive that should have been classified as negative [26]. False Negative (FN) represents the number of classifications predicted as negative that should have been classified as positive [26]. Table I shows the representation of the confusion matrix for binary classification.

TABLE I. CONFUSION MATRIX FOR BINARY CLASSIFICATION

	Predicted Positive	**Predicted Negative**
Actual Positive	TP	FN
Actual Negative	FP	TN

From the confusion matrix, accuracy and several other evaluation metrics can be calculated, including precision, recall (or sensitivity), and F1-score. Here are the formulas for calculating these metrics [27].

- Accuracy : used to measure how well the model predicts correctly.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

- Precision : used to measure the model's ability to predict positive class correctly from all positive predictions.

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

- Recall : used to measure the model's ability to predict true positive cases.

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

- Recall : used to measure the model's ability to predict true positive cases.

$$F1 - Score = \frac{2\ x\ Precision\ x\ Recall}{Precision+Recall} \quad (5)$$

Meanwhile, in multi-class classification, the confusion matrix has a dimension of $n \times n$, where n is the number of classes and $n > 2$. Therefore, the characterization of TP, TN, FP, and FN does not apply in multi-class classification confusion matrix [14]. However, analysis for classification results can still be conducted by focusing the analysis on a specific class, as shown in Table II with class C_2 as the main focus of analysis [14].

TABLE II. MULTI-CLASS CONFUSION MATRIX

	Predicted C_1	**Predicted C_2**	**…**	**Predicted C_N**
Actual C_1	$C_{1,1}$	FP	…	$C_{1,N}$
Actual C_2	FP	TP	…	FN
…	…	…	…	…
Actual C_N	$C_{N,1}$	FP	…	$C_{N,N}$

III. METHODS

A. Collecting Data

In this study, the "Estimation of obesity levels UCI" dataset obtained from the Kaggle website is used. The dataset was collected through a 30 day online survey accessible via a website platform with the subject of

people from Mexico, Peru, and Colombia, aged 14-61, regarding various dietary patterns and physical conditions [28]. In addition, the data were also synthetically generated using the SMOTE filter to balance the sample sizes between majority and minority classes [28]. SMOTE is an oversampling technique that generates new synthetic samples for the minority class by interpolation [29]. It first selects a random sample from the minority class and then selects another random sample from its nearest neighbors to perform interpolation by estimating new values from both samples [29].

From this entire process, a dataset with a total of 2111 instances and 17 attributes was created [28]. The features include gender, age, height, weight, family history of overweight, frequent consumption of high-caloric food (FAVC), frequency of vegetable consumption (FCVC), number of main meals (NCP), consumption of food between meals (CAEC), smoking habits (SMOKE), daily water consumption (CH2O), calories consumption monitoring (SCC), physical activity frequency (FAF), time using technology devices (TUE), alcohol consumption (CALC), and transportation used (MTRANS). Additionally, there are "NObeyesdad" class with values such as "Insufficient Weight," "Normal Weight," "Overweight Level I," "Overweight Level II," "Obesity Type I," "Obesity Type II," and "Obesity Type III." The feature data were obtained from questionnaires and the SMOTE filter, while the class data were obtained from BMI calculations [28]. Therefore, this dataset combines BMI values and features that can influence obesity levels with the aim of addressing the limitations of BMI as an indicator of obesity measurement.

B. Pre-processing

In this stage, data checking and deletion will be performed to remove all null and duplicated instances, ensuring data consistency, uniqueness, and preventing bias in the analysis. Furthermore, the processed data will be re-represented to be understood by machine learning by performing Label Encoding to convert all categorical data into numerical data.

C. Model Construction

This stage marks the initial step in the model selection process, starting with the construction and accuracy check of the initial model using the Random Forest algorithm to classify obesity levels and obtain the initial accuracy level of the model. This process involves dividing the dataset using a holdout scenario (8:2), which splits the dataset into training and testing data [30]. From the splitting process, 80% of the data is used to train the model with the training data, and the remaining 20% is used to evaluate how well the model classifies with the testing data. Subsequently, the model's accuracy is checked using the "*accuracy_score()*" function to obtain the initial accuracy level of the model.

D. Feature Selection and Model Construction

This stage marks the second step in model selection process, starting with feature selection using the ANOVA (Analysis of Variance) F-score method aided by the "*SelectKBest()*" function. The feature selection process iterates 16 times in accordance to the dataset's total features. In each iteration, data is split using an 8:2 holdout ratio to construct the second model for 16 times, which will be trained with the chosen features. The accuracy of the new model is then assessed using "*accuracy_score()*". The optimal number and the best-selected features are determined based on the model yielding the highest accuracy and will be used as the question lists in the classification system.

E. Hyperparameter Tuning and Model Construction

This stage marks the third stage in the model selection process, starting with hyperparameter tuning to find the best combination of hyperparameter values for optimal model classification performance. Hyperparameter tuning is performed using Randomized Search aided by the "*RandomizedSearchCV()*" function with 10-fold cross-validation, as it offers the most optimal k-fold value [23]. After that, the data is divided again into 2 parts using the 8:2 holdout scenario, which will be trained using the best combination of hyperparameter values obtained from the tuning results. Subsequently, the accuracy of this new model is assessed using the "*accuracy_score()*" function. Since this stage marks the final step in the model selection process, the next step involves comparing all three models to identify the one with the highest accuracy. This step is crucial as the best-performing model is chosen for further evaluation using various metrics and implementation into the classification system.

F. Model Performance

In this stage, firstly, the multi-class classification confusion matrix will be used to observe the distribution of predicted class results generated by the model against the actual classes. Then, further evaluation will assess the overall model performance for each class based on precision, recall, and F1-score, along with macro, weighted, and micro averages from the "*classification_report()*" function and micro average calculation. Lastly, log loss metrics will be used to check the model's error level in predicting class probabilities because it can be used for multi-class classification problems and aligns with the Random Forest algorithm's operation, which also computes class probabilities at each node for classification.

G. System Development

In this stage, firstly, a flowchart will be created to outline the system's workflow. Then, the system will be designed in prototype form to design the user interface and system flow to achieve the final classification results for each user. The system will be made like an online form with three pages according to each group

of questions, namely personal data, family health and eating habits, and daily routines, with a total of 16 questions corresponding to the original number of features in the dataset and will be made in Bahasa Indonesia. The final classification results will be displayed in a popup, which will either show an error message if an error occurs during the classification process because of the required fields validation error or show the classification results if the process is successful.

IV. Result and Analysis

A. Collecting Data

The first step in building the classification model is to read and store the dataset to be used. This dataset is taken from the Kaggle.com platform entitled "Estimation of Obesity Levels UCI" dataset consists of 2111 data with a total of 17 attributes. The implementation results can be seen in Fig 3.

Fig. 3. Data Collection from Kaggle.com

B. Model Construction

In this stage, the dataset is split into training and testing data using the "*train_test_split()*" function, with 20% of the data used for testing data. Then, the construction of the Random Forest model will start with training using the training data first to find relationships/ patterns between features and classes. After that, the model will use this trained data to predict on the testing data. In the final step, the accuracy of the model is checked by comparing the true classes with the classification results, resulting in an accuracy score of 95.4%. The implementation results can be seen in Fig 4.

Fig. 4. First Model Construction Code

C. Feature Selection and Model Construction

In this stage, the model will be rebuilt for the second time after feature selection to improve the accuracy obtained previously. This process is repeated 16 times to determine the number of features that produce the highest accuracy of the model. Feature selection technique is performed using "*SelectKBest()*" function which retains only the top *k* features in the dataset by involving the F-score analysis of variance (ANOVA) to evaluate the significance of the relationship between each feature and the class using f-score. Then, it will learn the patterns and relationships between features and classes and select only the top *k* features based on the f-score from the learning process.

In this loop, model building is also performed using different datasets in each iteration from *k=1* to *k=16* to select the optimal number and the best features. In its construction, the model will learn and predicts classes from the training and testing data in each iteration to check the model's accuracy score to determine the optimal number of features. The implementation code is shown in Fig 5.

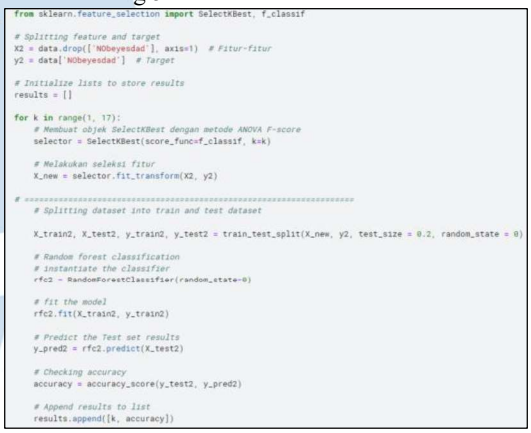

Fig. 5. Feature Selection and Second Model Construction

Next, the *k* value and its corresponding accuracy, which have been stored, will be displayed collectively for easier comparison. From this feature selection result, it is known that the optimal number of features is 12 with an accuracy of 96.4%. The implementation result is shown in Fig 6.

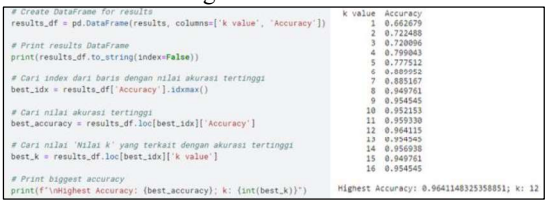

Fig. 6. Accuracy Summary at each Iteration

In the previous stage, it was determined that the optimal number of features is 12, so the next step is to

find out what the selected features are and retrain the model with *k=12,* because the model was last trained with 16 features (*k=16*) from the iteration result. The initial step in this process is to redefine the "*SelectKBest()"* object with *k=12*, then perform the feature selection. From the feature selection result, the F-score for each selected feature is displayed using. Then, the model will be rebuilt using the dataset from feature selection results. The selected features include "Gender", "Age", "Height", "Weight", "family history with overweight", "FAVC", "FCVC", "NCP", "CAEC", "SCC", "CALC", and "MTRANS". Meanwhile, "SMOKE", "CH2O", "FAF", and "TUE" are not used. From the model building result, an accuracy of 96.4% is obtained, indicating that the implementation of feature selection can increase the classification accuracy of the model by 1%, from 95.4% to 96.4%. The implementation result can be seen in Fig 7.

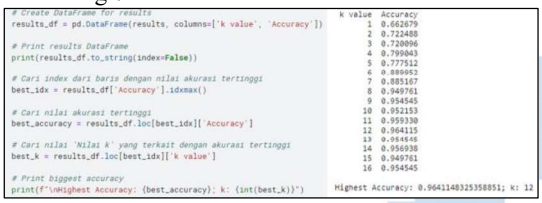

Fig. 7. Selected Features and Second Model Reconstruction

D. Hyperparameter Tuning and Model Construction

In this stage, the model will be rebuilt for the third time after performing hyperparameter tuning to improve the accuracy obtained previously. The hyperparameters used are "*criterion*", "*max_depth*", "*n_estimators*", "*max_features*", "*min_samples_split*", and "*max_leaf_nodes*". Criterion determines the criteria for measuring the quality of splits in tree building. Max_depth determines the maximum depth of the tree. *n_estimators* determines the number of decision trees to be built. *max_features* determines the maximum number of features to consider for splitting a node. *Min_samples_split* determines the minimum number of samples required to split a node. *Max_leaf_nodes* determines the maximum number of leaf nodes in the tree. Fig 8 represents the hyperparameters used along with their respective values.

```
param_dist = {
    'criterion': ['gini', 'entropy', 'log_loss'],
    'max_depth': [i for i in range(1, 101)], # 101
    'n_estimators': [i for i in range(10, 201)], # 201
    'max_features': ['sqrt', 'log2', None],
    'min_samples_split': [i for i in range(2, 6)], # 6
    'max_leaf_nodes': [i for i in range(10, 501)] # 501
}
```

Fig. 8. Hyperparameter Tuning Values

Next, hyperparameter tuning is performed using the values in Fig 8 using the "*RandomizedSeachCV()"* function to obtain the best combination of values for each hyperparameter. Firstly, the Randomized Search is initialized with several parameters. Then, the initialized result is used to train the model, which simultaneously searches for parameters to obtain the best parameter combination. From the tuning result, the best hyperparameter combination values for *n_estimators, min_samples_split, max_leaf_nodes, max_features, max_depth, and criterion* are 131, 2, 361, "*None*", 68, and "*log_loss*" respectively, as seen in Fig 9.

Fig. 9. Hyperparameter Tuning with Randomized Search

Then, the model is rebuilt for the third time to improve the accuracy from the previously obtained value. This process involves learning and predicting by utilizing the best hyperparameter results obtained earlier. From this model building, a new accuracy value of 96.8% is obtained that successfully increases the model's accuracy by 0.4%. This indicates that the implementation of feature selection and hyperparameter tuning can improve the model's accuracy in classification. The implementation result can be seen in Figure 10.

Fig. 10. Third Model Construction using the Tuning Results

Based on the results of the entire model development process, it can be observed that the use of feature selection and hyperparameter tuning successfully increased the model accuracy by a total of 1.4%. This improvement amounted to 1% and 0.4% in each respective stage. It indicates that implementing feature selection and hyperparameter tuning can enhance the model's accuracy in classifying obesity levels. The summary of accuracy from all model developments can be seen in the Table III.

TABLE III. SUMMARY OF THE ACCURACY LEVEL FROM ALL MODEL

Model	Feature Selection	Hyper-parameter Tuning	Accuracy	Increase
1	-	-	95.4%	-
2	✓	-	96.4%	1%
3	✓	✓	96.8%	0.4%

The third model achieved the highest accuracy of 96.8% from the model selection process. Therefore, it is chosen as the best selected model in the study that will be evaluated and implemented into the obesity level classification system.

E. Model Performance Evaluation

a) Evaluation with Confusion Matrix: This evaluation process begins with the calculation of confusion matrix, which will provide results in the form of a 2D array. This calculation process continues until the calculation for the last class is completed. The implementation result of the TP, TN, FP, and FN values for each class can be seen in Table IV.

TABLE IV. TP, TN, FP, AND FN VALUES FOR EACH CLASS

	TP	FN	FP	TN
Insufficient_Weight	45	1	2	370
Normal_Weight	45	6	1	366
Overweight_Level_I	54	2	5	357
Overweight_Level_II	58	2	3	355
Obesity_Type_I	74	1	2	341
Obesity_Type_II	69	1	0	348
Obesity_Type_III	60	0	0	358

From the summary in Table IV, it's evident that the largest misclassified class involves 4 samples incorrectly classified as "*Overweight_Level_I*" from "*Normal_Weight*" class. To conduct further checks, the 4 instances with its original values to before label encoding was performed from the most misclassified "*Normal_Weight*" class from the testing data will first be obtained.

b) Evaluation with Classification Report: From the result of classification report, several evaluation metrics are obtained, including the precision, recall, and f1-score for each class, as well as the accuracy, macro average, and weighted average values from the overall evaluation of the model. Precision represents how well the model classifies positive classes out of all positive predictions. Recall represents how well the model classifies positive classes out of all actual positive classes. F1-score represented how well the model balances precision and recall in classifying positive classes. Accuracy represents how accurate the model is in overall predictions. Macro avg represents the average evaluation metrics based on the precision, recall, and f1-score values from each class, without considering class distribution. Weighted Avg represents the average evaluation metrics based on the precision, recall, and f1-score values from each class, considering class distribution.

Additionally, there is one evaluation metric not included in classification report, which is the micro average, representing the average evaluation metrics based on the TP, FP, and FN values from all classes without considering class distribution. The implementation result along with the model evaluation can be seen in Fig 11, and the result of the classification report and the calculation of the micro average can be seen in Fig 12.

```
from sklearn.metrics import classification_report

print('Report: ' + classification_report(y_test2, y_pred3))
```

```
Report:                      precision    recall  f1-score   support

Insufficient_Weight       0.96      0.98      0.97        46
      Normal_Weight       0.98      0.88      0.93        51
     Obesity_Type_I       0.97      0.99      0.98        75
    Obesity_Type_II       1.00      0.99      0.99        70
   Obesity_Type_III       1.00      1.00      1.00        60
 Overweight_Level_I       0.92      0.96      0.94        56
Overweight_Level_II       0.95      0.97      0.96        60

           accuracy                           0.97       418
          macro avg       0.97      0.97      0.97       418
       weighted avg       0.97      0.97      0.97       418
```

Fig. 11. Classification Report Result

```
from sklearn.metrics import precision_recall_fscore_support

# Menghitung presisi, recall, dan F1-score menggunakan micro averaging
precision, recall, f1_score, _ = precision_recall_fscore_support(y_test2, y_pred3, average='micro')

# Menampilkan nilai metrik evaluasi
print("Micro average metrics:")
print("Precision: {:.2f}".format(precision))
print("Recall: {:.2f}".format(recall))
print("F1-score: {:.2f}".format(f1_score))
```

```
Micro average metrics:
Precision: 0.97
Recall: 0.97
F1-score: 0.97
```

Fig. 12. Micro Average Calculation Result

c) Evaluation with Log Loss: After evaluating the model's prediction results with Confusion Matrix and Classification Report, the next step is evaluating the model's uncertainty/ error level in predicting class probabilities. Log loss is a evaluation metric that measures how well the model predicts the true class probabilities. The smaller the value, the better the model's performance. In other words, log loss measures how accurate and reliable a model is in providing probabilities that a sample/entity belongs to a certain class. The steps calculating the probability values for each entity in each class, then use it to calculate the log loss. From this calculation, a log loss value of 0.09 or 9% is obtained. Fig13 represents the implementation result of log loss.

```
from sklearn.metrics import log_loss

y_proba = rfc3.predict_proba(X_test2)
logloss = log_loss(y_test2, y_proba)

print(f'Log loss {logloss}')
```

```
Log loss 0.09504090778292554
```

Fig. 13. Log Loss Result

F. User Interface Implementation

Overall, the system's interface implementation will resemble the prototyping results, but with only 12 questions as this is the optimal number of features determined by feature selection. The first page, will include questions about gender, age, height, weight, and an additional question for the user name to be displayed with the classification result. The second page will include questions about family histories of overweight, frequency of eating high-calorie foods, frequency of vegetable consumption, and the number of main meals per day. The third page will include questions about snacking frequency, daily calorie

monitoring, alcohol consumption frequency, and usual mode of transportation.

Additionally, there will be an additional feature on pages 2 and 3 to view more information about non-quantifiable answer options like “Sometimes”, “Rarely”, etc. This aims to clarify and standardize these options, which can vary between individuals. Furthermore, the classification result popup will now also include personalized health suggestions based on each user’s classification level. These additions were derived from an online interview with a doctor called dr. Jesslyn Valentina, M.M., conducted on April 16, 2024. Below is a detailed explanation of the three pages and the classification result popup in this classification system.

In the first page, users can fill in the five fields and navigate through the pages. Fig 14 represents the implementation for page 1.

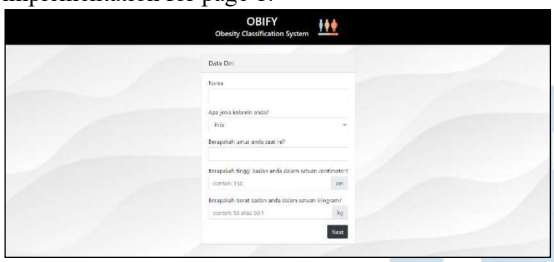

Fig. 14. Implementation Result for Page 1

The second page contains a total of 4 questions as previously described. In page 2, users can also fill in the four fields and navigate through the pages. Fig 115 represents the implementation for page 2.

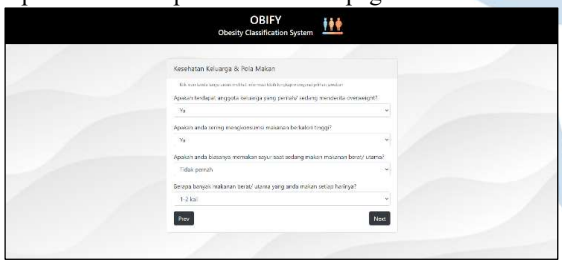

Fig. 15. Implementation Result for Page 2

The answer options for third question on the second page is unmeasurable, and this question is one of the questions that needs the “view more information” feature that was explained before that was obtained from the online interview with a doctor. Fig16 represents a more detailed explained for the third question.

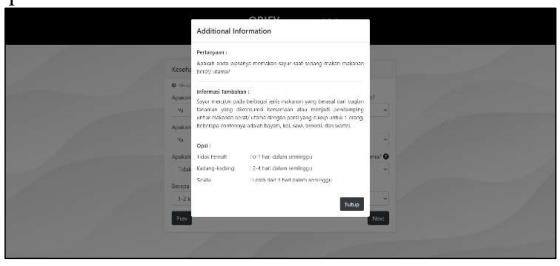

Fig. 16. Detailed Information for the Third Question on the Second Page

The third page contains a total of 4 questions as previously described. In page, users can also fill in the four fields and navigate through the pages. In this page, there is also the submit button “Cek Hasil” to process all inputs and returns the obesity level classification result. Fig 17 represents the implementation for page 3.

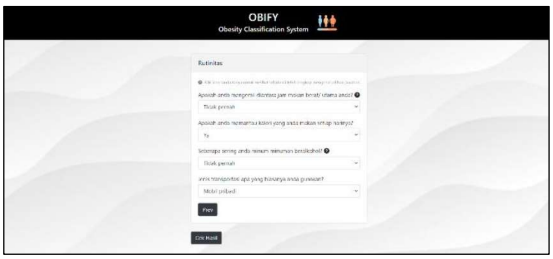

Fig. 17. Implementation Result for Page 3

The answer options for first and third questions on the third page are also unmeasurable and needs the “view more information” feature that was explained before. Fig 18 and 19 represents a more detailed explained for the first and third question respectively.

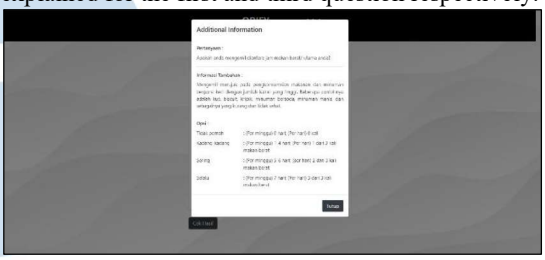

Fig. 18. Detailed Information for the First Question on the Third Page

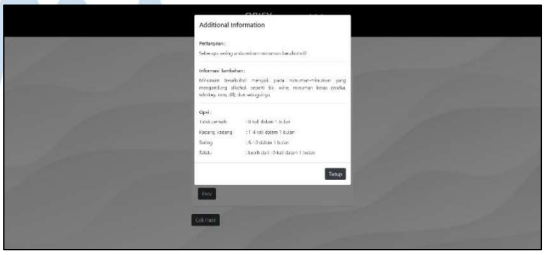

Fig. 19. Detailed Information for the Third Question on the Third Page

Then, these are the user interface for the error popup because of the failed validation in required fields and server/ API error respectively in Fig 20 and 21.

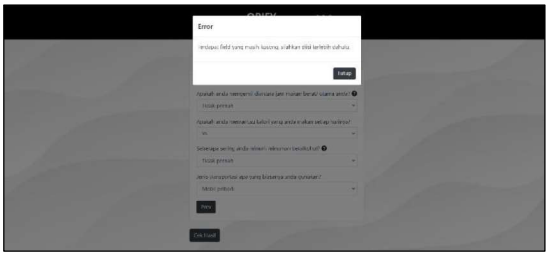

Fig. 20. Error Popup Interface for Required Field Validation Error

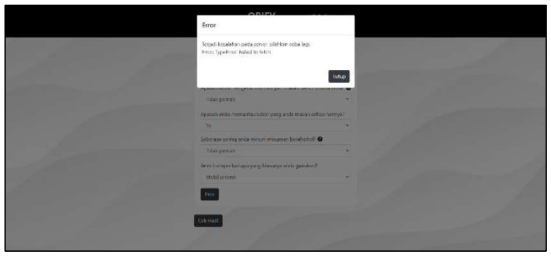

Fig. 21. Error Popup Interface for Server/ API Error

Then, these are the user interface for the success popup for class "Normal_Weight" and all other 6 classes that outputs the classification result and the personalized suggestions based on the obtained obesity level. The icon/ picture representing insufficient weight, normal weight, overweight, and obesity was obtained from the flaticon.com website. The personalized suggestions for each level are also obtained from the online interview. Here is the implementation respectively in Fig 22 and Fig 23.

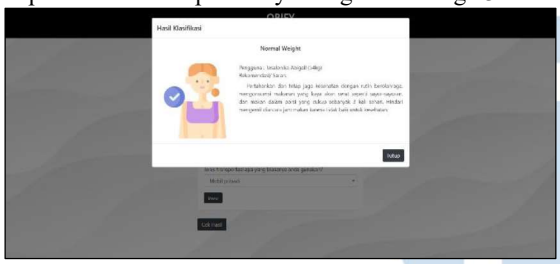

Fig. 22. Success Popup Interface for Class "Normal_Weight"

Fig. 23. Success Popup Interface for Six Other Classes

V. CONCLUSION

This research shows that the Random Forest algorithm can effectively classify obesity levels. The model also successfully implemented into a system using HTML, CSS, JS, and Flask with the assistance of the Pickle module, accurately processes user input and provides reliable classification results and health advice. The model itself exhibits excellent performance with 96.8% accuracy, 97% precision, 97% recall, and 97% f1-score. However, it shows a relatively high misclassification of "Normal_Weight" into "Overweight_Level_I" class.

For further development, the previously ignored features from the feature selection process can be combined to enhance the model performance. Additionally, using or collecting a new dataset reflecting Indonesian adults' dietary and daily habits can also be conducted to better match local characteristics. Lastly, a classification history features can also be added to the system to allows users to track and monitor their own health progress.

REFERENCES

[1] A. M. S. I. Dewi and I. B. G. Dwidasmara, "Implementation of the k-nearest neighbor (knn) algorithm for classification of obesity levels," *JELIKU (Jurnal Elektronik Ilmu Komputer Udayana)*, vol. 9, 2020.

[2] G. Castelnuovo, B. D. Cuevillas, S. Navas-Carretero, and J. A. Mart´ ınez, "Body fat mass assessment and obesity classification: A review of the available methods for adiposity estimation," *Progress in Nutrition*, vol. 23, 2021.

[3] S. Jeon, M. Kim, J. Yoon, S. Lee, and S. Youm, "Machine learningbased obesity classification considering 3d body scanner measurements," *Scientific Reports*, vol. 13, 2023.

[4] K. M. Heinrich, K. G. Gurevich, A. N. Arkhangelskaia, O. P. Karazhelyaskov, and W. S. Poston, "Despite low obesity rates, body mass index under-estimated obesity among russian police officers when compared to body fat percentage," *International Journal of Environmental Research and Public Health*, vol. 17, 2020.

[5] E. Erlin, Y. Desnelita, N. Nasution, L. Suryati, and F. Zoromi, "Dampak smote terhadap kinerja random forest classifier berdasarkan data tidak seimbang," *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 21, 2022.

[6] M. Phongying and S. Hiriote, "Diabetes classification using machine learning techniques," *Computation*, vol. 11, 2023.

[7] B. O. Macaulay, B. S. Aribisala, S. A. Akande, B. A. Akinnuwesi, and O. A. Olabanjo, "Breast cancer risk prediction in african women using random forest classifier," *Cancer Treatment and Research Communications*, vol. 28, 2021.

[8] R. Susetyoko, W. Yuwono, E. Purwantini, and N. Ramadijanti, "Perbandingan metode random forest, regresi logistik, na¨ ıve bayes, dan multilayer perceptron pada klasifikasi uang kuliah tunggal (ukt)," *Jurnal Infomedia*, vol. 7, 2022.

[9] T. Kok, V. Wiriantono, J. Bakhriansyah, and L. Aditama, "The factors affecting the occurrence of obesity in college students," *Unnes Journal of Public Health*, vol. 12, 2023.

[10] D. S. Harbuwono, L. A. Pramono, E. Yunir, and I. Subekti, "Obesity and central obesity in indonesia: Evidence from a national health survey," *Medical Journal of Indonesia*, vol. 27, 2018.

[11] S. Madakam, T. Uchiya, S. Mark, and Y. Lurie, "Artificial intelligence, machine learning and deep learning (literature: Review and metrics)," *Asia-Pacific Journal of Management Research and Innovation*, vol. 18, 2022.

[12] A. Alanazi, "Using machine learning for healthcare challenges and opportunities," 2022.

[13] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to machine learning, neural networks, and deep learning," *Translational Vision Science and Technology*, vol. 9, 2020.

[14] I. Markoulidakis, I. Rallis, I. Georgoulas, G. Kopsiaftis, A. Doulamis, and N. Doulamis, "Multiclass confusion matrix reduction method and its application on net promoter score classification problem," *Technologies*, vol. 9, 2021.

[15] H. Khalid, A. Khan, M. Z. Khan, G. Mehmood, and M. S. Qureshi, "Machine learning hybrid model for the prediction of chronic kidney disease," *Computational Intelligence and Neuroscience*, vol. 2023, 2023.

[16] C. S. Lee, P. Y. S. Cheang, and M. Moslehpour, "Predictive analytics in business analytics: Decision tree," *Advances in Decision Sciences*, vol. 26, 2022.

[17] S. Hamsa, I. Shahin, Y. Iraqi, and N. Werghi, "Emotion recognition from speech using wavelet packet transform cochlear filter bank and random forest classifier," *IEEE Access*, vol. 8, 2020.

[18] C. A. Bugeac, R. Ancuceanu, and M. Dinu, "Qsar models for active substances against pseudomonas aeruginosa using disk-diffusion test data," *Molecules*, vol. 26, 2021.

[19] Sriyanto and A. R. Supriyatna, "Teknika 17 (1): 163-172 prediksi penyakit diabetes menggunakan algoritma random forest," *Jurnal Teknika*, vol. 17, 2023.

[20] J. Yang, J. Gong, W. Tang, Y. Shen, C. Liu, and J. Gao, "Delineation of urban growth boundaries using a patch-based cellular automata model under multiple spatial and socio-economic scenarios," *Sustainability (Switzerland)*, vol. 11, 2019.

[21] T. J. Kiely and N. D. Bastian, "The spatially conscious machine learning model," *Statistical Analysis and Data Mining*, vol. 13, 2020.

[22] I. M. M. Matin, "Hyperparameter tuning menggunakan gridsearchcv pada random forest untuk deteksi malware," *MULTINETICS*, vol. 9, 2023.

[23] F. Rahmadayana and Y. Sibaroni, "Sentiment analysis of work from home activity using svm with randomized search optimization," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 5, 2021.

[24] H. Yun, "Prediction model of algal blooms using logistic regression and confusion matrix," *International Journal of Electrical and Computer Engineering*, vol. 11, 2021.

[25] A. Tharwat, "Classification assessment methods," *Applied Computing and Informatics*, vol. 17, 2018.

[26] T. F. Monaghan, S. N. Rahman, C. W. Agudelo, A. J. Wein, J. M. Lazar, K. Everaert, and R. R. Dmochowski, "Foundational statistical principles in medical research: Sensitivity, specificity, positive predictive value, and negative predictive value," 2021.

[27] H. Agatha, F. Putri, and A. Suryadibrata, "Sentiment Analysis on Song Lyrics for Song Popularity Prediction Using BERT", Ultimatics : Jurnal Teknik Informatika, vol. 15, no. 2, pp. 99-105, Jan. 2024.

[28] F. M. Palechor and A. de la Hoz Manotas, "Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from colombia, peru and mexico," *Data in Brief*, vol. 25, 2019.

[29] Yusuf Gregorius Neven and Gunawan Dennis,SMOTE and Random Forest Algorithm for Detecting DoS Attacks in Wireless Sensor Networks,"ICIC Express Letters, Part B: Applications",21852766,ICIC International, 2022,13,07,715,https://cir.nii.ac.jp/crid/1390855190129179776,https://doi.org/10.24507/icicelb.13.07.715.

[30] F. Tempola, R. Rosihan, and R. Adawiyah, "Holdout validation for com parison classfication na¨ ıve bayes and knn of recipient kartu indonesia pintar," *IOP Conference Series: Materials Science and Engineering*, vol. 1125, 2