ACCURACY ANALYSIS OF BMI PREDICTION USING REGRESSION ALGORITHMS

Thin Ei Zar[1], Khin Sandar Myint[2], Soe Mya Mya Aye[3]

Abstract

This study conducts an accuracy analysis of BMI prediction using regression algorithms, specifically decision tree and random forest regression. A dataset used in this paper was collected from the Kaggle data repository site. The dataset is split into training and testing sets to predict BMI, BMR and calories needed to maintain weight based on gender, age, weight, height, and activity level using decision tree and random forest regression algorithms. The performance accuracy of the proposed algorithms is evaluated using Root Mean Squared Error, Mean Absolute Error, R-squared, Mean Absolute Percentage Error and accuracy. The goal and content of this research are to study the prediction accuracy of decision tree and random forest regression machine learning algorithms in prediction. Analyzing BMI will be specifically focused on suggesting the daily calories to maintain weight to the user to get a healthy life that leads to a happy life.

Keywords: BMI Prediction, Regression Algorithm, Analysis of BMI, Prediction Accuracy

Introduction

Body Mass Index (BMI) is a commonly used measure that classifies individuals according to body weight and height, helping to identify if a person is underweight, normal weight, overweight, or obese. Body Mass Index (BMI), Basal Metabolic Rate (BMR), and caloric needs are essential metrics in assessing an individual's health status. BMR represents the number of calories required to maintain basic physiological functions. Understanding BMR is crucial for calculating daily caloric needs. Accurate prediction of BMI, BMR, and caloric needs is increasingly important for healthcare professionals and researchers to address various health-related issues.

This study aims to compare BMI prediction accuracy using regression algorithms, specifically decision tree and random forest regression. These algorithms are chosen for their ability to handle complex, non-linear relationships between variables and their robustness in prediction tasks. The dataset for the research consists of 10,726 records with attributes such as age, weight, height, gender, BMI, BMR, activity level and calories required to maintain weight. The dataset is split into training and testing sets, allowing for a systematic approach to predicting BMI, BMR and maintenance calories based on gender, age, weight, height and activity level. Using Python's Jupyter Notebook for data visualization and analysis, the study employs decision tree and random forest regression algorithms for the prediction process.

This comparative analysis offers valuable insights into the effectiveness of decision tree and random forest regression algorithms in predicting BMI. The findings have significant implications for health and a better understanding of the strengths and limitations of these algorithms in BMI prediction tasks. Moreover, the study emphasizes choosing the right algorithm for accurate health predictions, which can directly impact personalized healthcare.

[1] Department of Computer Studies, University of Yangon
[2] Department of Computer Studies, University of Yangon
[3] Department of Computer Studies, University of Yangon

Methods of Machine Learning Algorithms for BMI Prediction

A machine learning algorithm is a set of rules or processes used by an AI system to conduct tasks, most often to discover new data insights and patterns, or to predict output values from a given set of input variables. Machine learning algorithms can be categorized into four types: supervised, unsupervised, semi-supervised and reinforcement learning. This research will use a regression algorithm in supervised learning. Supervised learning is a form of machine learning in which the algorithm is trained on labeled data to make predictions or decisions based on the data inputs. Supervised learning can be further classified into classification and regression.

Classification is a two-step process such as learning (model development) and prediction, suitable for problems with discrete output labels such as email filtering (spam or not spam) and weather categorization (sunny, cloudy, rainy). Regression is predicting continuous values based on input such as housing price prediction.

Regression algorithms are appropriate in the context of BMI and BMR prediction in this research because they predict continuous values rather than discrete categories. Regression will be used for predicting values like BMI, BMR and calories needed to maintain weight based on continuous input variables such as age, height, weight, gender and activity level.

There are many different types of regression algorithms such as Linear Regression, Polynomial Regression, Support Vector Regression (SVR), Decision Tree Regression and Random Forest Regression. Among them, decision tree regression and random forest regression are chosen for their simplicity and accuracy in these predictions.

The implementation process of the research

To develop this research, the dataset collection is the basic step of the process. The dataset used in this research is collected from the Kaggle data repository site and the dataset includes 10726 records with suitable attributes for this research. The prediction process for BMI, BMR and calories needed to maintain weight based on age, height, weight, gender and activity level will be implemented in the Python Jupyter Notebook. After collecting the dataset, the stages of methodology and implementation process have seven stages as shown in Figure 1 as a flow diagram.

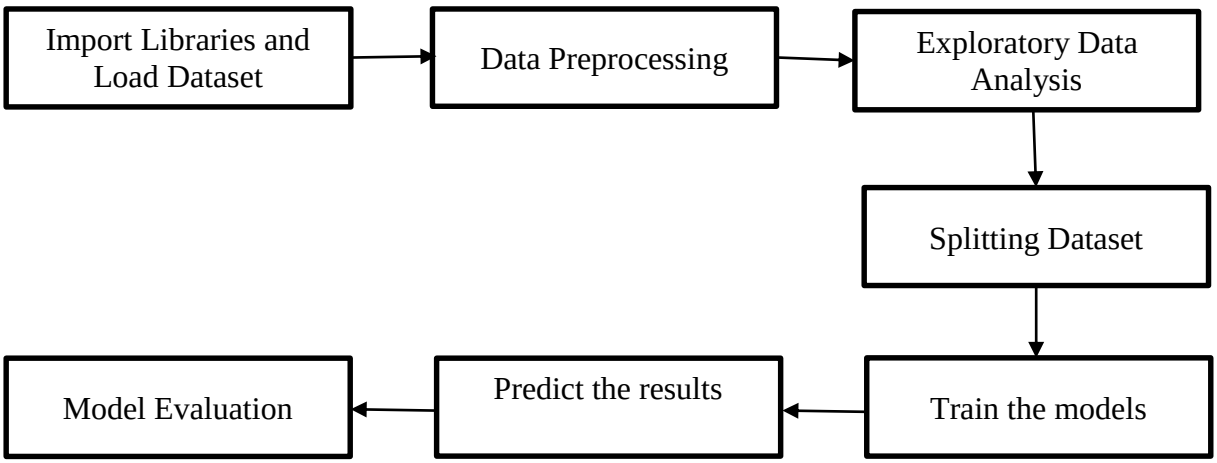


Figure 1 Flow diagram of the research for BMI prediction using regression algorithms

Import Libraries and Load Data

This step involves importing the necessary libraries such as pandas for data manipulation, scikit-learn for machine learning algorithms and seaborn and matplotlib for plotting. With pandas, data from various file formats (e.g., CSV, Excel) can be read, handle missing data and perform complex data transformations. Scikit-learn (sklearn) is a comprehensive library for machine learning in Python. Seaborn is built on top of Matplotlib and provides a high-level interface for drawing attractive and informative statistical graphics. Matplotlib is a widely-used plotting library that offers plots and charts such as line graphs, scatter plots, and more to visually explore and present data.

In the load data step, the dataset is loaded into the environment. The data can be from various sources like CSV files, databases or other formats. The dataset used in this research is a CSV file. It is essential to understand the structure and content of the dataset for effective analysis. Figure 2 shows the code for importing the libraries and loading the dataset into the Jupyter Notebook.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import tree
from sklearn.metrics import accuracy_score
mydata = pd.read_csv('C:/Users/User/Downloads/BMIDataset.csv')
melbourne_data = mydata.dropna(axis=0)
result_data = melbourne_data.dropna(axis=0)

mydata.describe().round(decimals = 2)
```

Figure 2 Importing Python Libraries and Loading the Dataset

Data Preprocessing

Data preprocessing is an essential step that involves handling missing values, encoding categorical features, scaling numerical features and possibly other transformations. This step ensures the data is clean and suitable for model training. Data preprocessing can be performed to ensure the shape of the dataset, no duplicated and null data in the dataset, and the detailed information of the dataset and to drop or remove unnecessary attributes for the dataset.

The info () function is utilized to present the number of columns, their respective labels, data kinds and the count of non-null cells within each column. Kene, et.al, (2023) according to the data presented in Figure 3(a), there is a lack of empty values within the dataset.

```
mydata.isnull().any()

age                          False
weight(kg)                   False
height(m)                    False
gender                       False
BMI                          False
BMR                          False
activity_level               False
calories_to_maintain_weight  False
BMI_tags                     False
Label                        False
dtype: bool
```

Figure 3(a) No null data

```
mydata.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10726 entries, 0 to 10725
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   age                          10726 non-null  int64
 1   weight(kg)                   10726 non-null  float64
 2   height(m)                    10726 non-null  float64
 3   gender                       10726 non-null  object
 4   BMI                          10726 non-null  float64
 5   BMR                          10726 non-null  float64
 6   activity_level               10726 non-null  float64
 7   calories_to_maintain_weight  10726 non-null  float64
 8   BMI_tags                     10726 non-null  int64
 9   Label                        10726 non-null  int64
dtypes: float64(6), int64(3), object(1)
memory usage: 838.1+ KB
```

Figure 3(b) Information of the dataset

Figure 3(b) shows the information of the dataset that the dataset has 10726 entries, 0 to 10725, and 10 data columns (age, weight(kg), height(m), gender, BMI, BMR, activity-level, calories to maintain weight, BMI_tags, Label.

```
mydata.shape

(10726, 10)

mydata.duplicated()

0        False
1        False
2        False
3        False
4        False
        ...
10721    False
10722    False
10723    False
10724    False
10725    False
Length: 10726, dtype: bool
```

Figure 4(a) Shape of data and No Duplicated data in the dataset

```
Dropping Unwanted attributes

analysis= mydata.drop(['BMI_tags','Label'],axis = 1)

analysis.shape

(10726, 8)

pd.DataFrame(analysis[:6]).round(decimals = 2)
```

	age	weight(kg)	height(m)	gender	BMI	BMR	activity_level	calories_to_maintain_weight
0	2	16.10	0.93	Female	18.53	958.58	1.2	1150.30
1	4	14.62	0.92	Female	17.40	932.38	1.7	1585.05
2	4	17.90	1.00	Female	18.00	977.58	1.9	1857.40
3	3	13.53	1.02	Female	12.94	944.69	1.9	1794.91
4	4	17.04	1.05	Male	15.34	799.23	1.9	1518.54
5	3	12.03	1.08	Female	10.34	939.78	1.9	1785.58

Figure 4(b) Dropping unwanted columns

Figure 4(a) describes the shape of the dataset and the data contains 10726 rows, 10 attributes, and no duplicated data in the dataset. Figure 4(b) shows that dropping unnecessary columns for the machine learning process. In this research, the predictive modeling with decision tree and random forest regression will need only 8 columns without 2 columns (BMI_tags, Label). So, during this data pre-processing stage, unwanted columns need to drop for the improvement of the prediction accuracy.

In data preprocessing**,** data exploration is exploring the dataset using Exploratory Data Analysis (EDA) to understand the data before applying machine learning algorithms. EDA identifies data patterns and correlations, preparing the dataset for analysis.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) helps understand the data before applying machine learning algorithms. EDA identifies data patterns and correlations, preparing the dataset for analysis. Visualization is a crucial part of EDA. By analyzing the dataset based on the gender attributes, the data set involves the number of females 5572 and males 5154.

```
people = analysis['gender'].value_counts()
print(type(people))
people.head()

<class 'pandas.core.series.Series'>
gender
Female    5572
Male      5154
Name: count, dtype: int64

fig, ax= plt.subplots( figsize=(3,4))
sns.countplot(x='gender',  data=analysis)
```

Figure 5(a) Data count for Gender

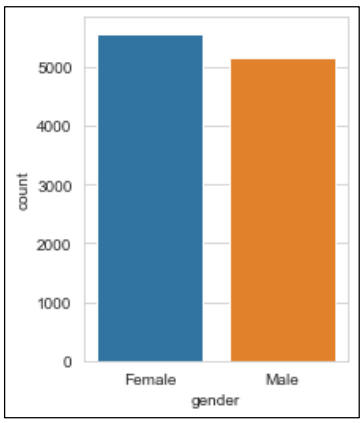
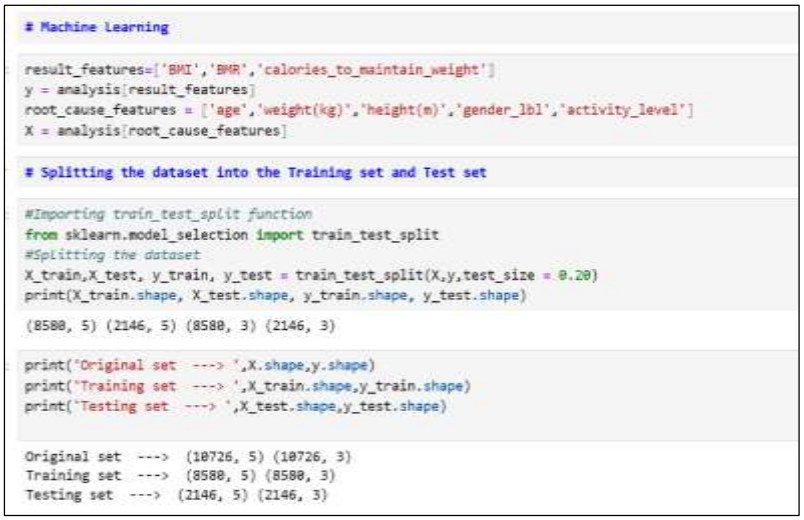


Figure 5(b) Count plot for Gender

Figure 5(a) shows the code for counting data based on the gender attribute of the dataset. Output can be seen in Figure 5(b) as a count plot.

Splitting Dataset

The dataset is divided into training and testing sets. The training set is used to train the model, while the testing set is used to evaluate its performance. For the prediction of the training and testing data of the dataset using the regression algorithm, X and Y must be defined first from the dataset. In this regression analysis, the dependent variable (outcome variable) is denoted "Y" and the independent variables(predictors) are denoted "X".



```
# Machine Learning

result_features=['BMI','BMR','calories_to_maintain_weight']
y = analysis[result_features]
root_cause_features = ['age','weight(kg)','height(m)','gender_lbl','activity_level']
X = analysis[root_cause_features]

# Splitting the dataset into the Training set and Test set

#Importing train_test_split function
from sklearn.model_selection import train_test_split
#Splitting the dataset
X_train,X_test, y_train, y_test = train_test_split(X,y,test_size = 0.20)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

(8580, 5) (2146, 5) (8580, 3) (2146, 3)

print('Original set  ---> ',X.shape,y.shape)
print('Training set  ---> ',X_train.shape,y_train.shape)
print('Testing set  ---> ',X_test.shape,y_test.shape)

Original set  --->  (10726, 5) (10726, 3)
Training set  --->  (8580, 5) (8580, 3)
Testing set  --->  (2146, 5) (2146, 3)
```

Figure 6 The code and output for splitting the dataset

Figure 6 shows the code for defining X and Y and splitting the dataset for training and testing data. For the prediction process, 5 attributes (age, gender, weight(kg), height(m), and activity level) from data will be used as input and will be defined as X. 3 attributes (BMI, BMR, and calories to maintain weight) are defined as Y, will be calculated and predicted as an output result based on X.

The split ratio of 80:20 means that t*he 20% testing data set is represented by the test_size= 0.20 at the end. The output of is original dataset is 10726 records. Training 80% is 8580 and testing 20% is 2146 for each attribute.*

Train the models

In this step, the Decision Tree and Random Forest Regressors are imported and fit the model for the prediction process using the training and testing dataset. These models learn the patterns and relationships within the data to make accurate predictions.

For the training and testing process using a decision tree, need to build and train the model by importing the scikit-learn DecisionTreeRegressor with the following code.

Code for importing the DecisionTree Regressor:

```
from sklearn.tree import DecisionTreeRegressor
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
y_dt=DecisionTreeRegressor ().fit(X_train, y_train)
```

Decision tree regression will be used to predict y (BMI, BMR and calories_to _maintain _weight) based on X (height(m), weight(kg), gender_lbl, age, activity_level) for the training dataset.

Random forest regression will be used to predict y (BMI, BMR and calories_to _maintain _weight) based on X (height(m), weight(kg), gender_lbl, age, activity_level). To use random forest regression, the regressor must be imported by the following code.

Code for Importing the Random Forest Regressor:

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators = 30, random_state = 30)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
y_rf=model.fit(X_train, y_train)
```

The above code fits the decision tree and random forest regression model with training and testing data.

Predict the results

The trained models are used to make predictions on the training and testing data. The predictions on the training and testing dataset using decision tree regression are processed and predict the results of Y (BMI, BMR, and calories_to_maintain_weight) based on X (age, weight(kg), height(m), gender_lbl, and activity_level will be processed by the following code.

Code for prediction on the training dataset using decision tree regression:

```
ytrain_pred_tree= y_dt.predict(X_train)
pd.DataFrame(ytrain_pred_tree,columns=["BMI","BMR","calories_to_maintain_wei
ght"].round (decimals = 2)
```

Code for prediction on the testing dataset using decision tree regression:

```
ytest_pred_tree = y_dt.predict(X_test)
pd.DataFrame(ytest_pred_tree,columns=["BMI",  "BMR",  "calories_to_maintain_
weight"] ). round (decimals = 2)
```

The predictions on the training and testing dataset using random forest regression are processed and the results by the following code.

Code for prediction on the training dataset using random forest regression:

```
ytrain_pred_forest= y_rf.predict(X_train)
pd.DataFrame(ytrain_pred_forest,columns=["BMI","BMR","calories_to_maintain_w
eight"]).round (decimals = 2)
```

Code for prediction on the testing dataset using random forest regression:

```
ytest_pred_forest = y_rf.predict(X_test)
pd.DataFrame(ytest_pred_forest,columns=["BMI","BMR","calories_to_maintain_we
ight"] ).round (decimals = 2)
```

The predicted results for training and testing using decision tree regression and random forest regression are checked to evaluate the accuracy or performance of the model.

Model Evaluation

The models' performance is evaluated using metrics such as R-squared, Root Mean Squared Error (RMSE), Mean Absolute Error(MAE), Mean Absolute Percentage Error(MAPE), and Accuracy (%). These metrics provide insights into the accuracy and effectiveness of the models.

Results

Evaluation for Training and Testing Data using Decision Tree Regression

This study evaluates the performance of Decision Tree and Random Forest Regression models using several standard metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), R-squared (R^2), and Mean Absolute Percentage Error (MAPE) and Accuracy (%). These metrics are applied to assess the models' training and testing phases. The mathematical formulas for these metrics are provided below, while the corresponding Python code for the implementation can be run in the Python Jupyter Notebook environment.

Root Mean Squared Error (RMSE): $\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$

Mean Absolute Error (MAE): $\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$

R-squared (R^2) Coefficient of determination: $R^2 = 1 - \frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$

Mean Absolute Percentage Error (MAPE): $\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{\hat{y}_i - y_i}{y_i} \right| \times 100$

y = actual value of y, $\hat{y}_i$= predicted value of y, n = number of data points

Accuracy (%): Accuracy (%)=100%−MAPE

The metrics for the training dataset of the Decision Tree regression models are computed using Python within a Jupyter Notebook, without the need for manually applying mathematical formulas. An example of Python code for RMSE is shown in Figure 7.

```
Metric Evaluation for prediction of training data using decision tree regression

from sklearn.metrics import mean_absolute_error, mean_squared_error,r2_score

#The RMSE
import numpy as np
print("The RMSE is:", np.sqrt(mean_squared_error(y_train,ytrain_pred_tree)))

The RMSE is: 0.0
```

Figure 7 Examples of Python code for RMSE

The results of metric evaluation for the decision tree and random forest are summarized in Table 1 below.

Table 1: **Performance of Decision Tree and Random Forest on Training and Testing Datasets**

Model Evaluation Metrics	Decision Tree Training	Decision Tree Testing	Random Forest Training	Random Forest Testing
RMSE	0.00	8.06	2.14	5.53
MAE	0.00	3.18	0.95	2.42
R^2	1.00	1.00	1.00	1.00
MAPE	0.00 %	0.49 %	0.13 %	0.30 %
Accuracy(%)	100.00 %	99.51 %	99.87 %	99.70 %

Table 1 compares the Decision Tree and Random Forest models using various evaluation metrics: RMSE, MAE, R^2, MAPE, and Accuracy (%). The metrics are evaluated for both training and testing datasets, providing insights into model performance and its ability to handle new data.

Root Mean Squared Error (RMSE): The metric, representing the square root of the average squared differences between actual and predicted values, shows that the Decision Tree model achieves a training RMSE of 0.00 but a testing RMSE of 8.06, indicating a significant error increase on unseen data. In contrast, the Random Forest model has a training RMSE of 2.14 and a testing RMSE of 5.53, reflecting a more moderate error increase on the test set.

Mean Absolute Error (MAE): MAE represents the average absolute differences between predicted and actual values. The Decision Tree model achieves an MAE of 0.00 during training and 3.18 during testing, reflecting a noticeable rise in error on the test data. The Random Forest model has an MAE of 0.95 for training and 2.42 for testing, demonstrating more robust performance and less variation in error between training and testing datasets.

R-squared (R^2): This statistic indicates the proportion of variance in the target variable that is predictable from the independent variables. Both models achieve an R^2 of 1.00 in training and testing, suggesting a perfect fit. However, such high R^2 values could indicate potential overfitting, particularly when combined with higher error metrics in testing for the Decision Tree model.

Mean Absolute Percentage Error (MAPE): MAPE measures the model's performance as a percentage, with lower values indicating better performance. The Decision Tree model achieves a MAPE of 0.0 in training and 0.49 in testing, while the Random Forest model shows a MAPE of 0.13 in training and 0.30 in testing. The slightly higher MAPE values for the Random Forest in training and testing suggest a more realistic measure of prediction error than those of the Decision Tree.

Accuracy (%): The Decision Tree model has a training accuracy of 100% and a testing accuracy of 99.51%. The Random Forest model has a training accuracy of 99.87% and a testing accuracy of 99.70%.

Overall, the Decision Tree model demonstrates excellent performance in training but suffers from significant performance degradation in testing, potentially indicating overfitting. The Random Forest model, on the other hand, shows more consistent performance across training and testing, suggesting better generalizability.

Comparison of Accuracy (%) for Decision Tree and Random Forest Regression

The comparison between Decision Tree Regression and Random Forest Regression reveals that both models perform exceptionally well in predicting outcomes. However, the Random Forest model demonstrates a slight edge in accuracy and robustness, particularly in handling more complex datasets.

Table 2. Comparison of accuracy (%) for Decision Tree and Random Forest Regression

Model	Training Accuracy (%)	Testing Accuracy (%)
Decision Tree Regression	100.00 %	99.51 %
Random Forest Regression	99.87 %	99.70 %

Table 2 compares the training and testing accuracies of Decision Tree and Random Forest regression models. Both models show high accuracy, with the Decision Tree achieving

perfect accuracy in training (100%) and slightly lower in testing (99.51%). Random Forest performs nearly as well, with a slight drop in training (99.87%) and testing (99.70%) accuracies.

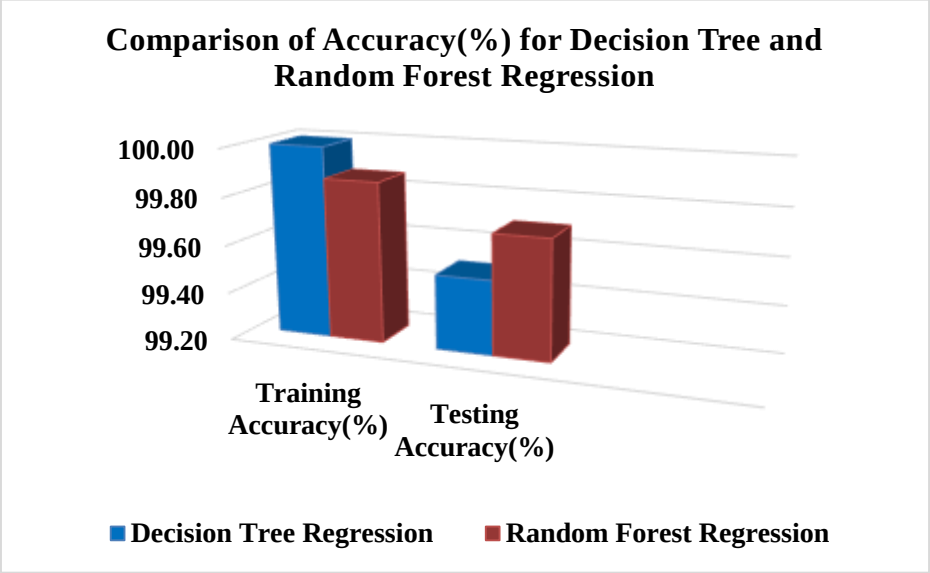


Figure 8 Comparison of Accuracy (%) between the two models

Figure 8 illustrates the performance of Decision Tree and Random Forest models mainly in accuracy (%). While Decision Tree performs slightly better in training accuracy, Random Forest exhibits a slight advantage in testing accuracy. The chart underscores the competitive nature of both models in predictive tasks.

Discussion

The performance comparison between decision tree regression and random forest regression reveals that the training accuracy is higher than the testing accuracy for both models. Decision tree regression shows perfect accuracy on the training data (100%) but a slight drop in testing accuracy (99.51%), suggesting that while the model captures the training data well, it may slightly overfit, leading to a minor reduction in generalization to new data.

Random forest regression achieves nearly perfect accuracy on both training (99.87%) and testing data (99.70%), indicating that this model generalizes slightly better to unseen data compared to the decision tree model. The high testing accuracy demonstrates its robustness and effectiveness in making accurate predictions across different data sets.

The random forest regression model achieves better accuracy (99.70%) compared to the decision tree regression model (99.51%) on the testing dataset. This superior performance can be attributed to the ensemble nature of the random forest, which reduces overfitting and improves generalization by aggregating the predictions of multiple decision trees.

This study predicts three attributes BMI, BMR and the calories needed to maintain weight based on age, weight, height, gender and activity level. Decision Tree Regression is simple to understand and interpret but can have high variance and may overfit the training data. On the other hand, Random Forest Regression, which combines multiple decision trees, delivers improved predictive accuracy and robustness. Both models show high accuracy in predictions, but Random Forest Regression generally offers more reliable results for new data due to its enhanced generalization ability. This reliability makes Random Forest a preferred option for applications that require consistent performance across varied datasets.

Conclusion

This study shows that random forest regression is more accurate and reliable than decision tree regression in predicting BMI, BMR and daily calorie needs based on age, weight, height, gender and activity level. The random forest model handles the complexity of predicting multiple outcomes at once and is less likely to make errors. This research highlights the value of machine learning in improving health assessments and creating personalized dietary recommendations. These advancements can benefit everyone, especially in areas where access to regular healthcare is limited. By making accurate health predictions more accessible, this study supports better health management and healthier lifestyles in our society. Future research could explore more machine learning methods to further improve these predictive systems.

Acknowledgement

I would like to express my profound gratitude to Dr. Soe Mya Mya Aye, Professor and Head of the Department of Computer Studies, University of Yangon, for her kind permission, encouragement, and valuable suggestions to carry out this research. I also would like to thank my supervisor Dr. Khin Sandar Myint, Associate Professor, Department of Computer Studies, University of Yangon, for her close supervision, invaluable suggestions and kind guidance. I wish to thank all who help and give me advice on this research.

References

B. Manoj Kumar, et al., (2022) “Accuracy Analysis of Heart Disease Prediction using Logistic Regression in Comparison with the Linear Regression Algorithm”, Journal of Pharmaceutical Negative Results, Volume 13, Special Issue 4.

F. Ferdowsy, K.S.A. Rahi, Md.I. Jabiullah et al., (2021) “A machine learning approach for obesity risk prediction”, DOI: https://doi.org/10.1016/j.crbeha.2021.100053.

Joy, Jodder, (2020) “An Effective Recommendation System to Forecast the Best Educational Program Using Machine Learning Classification Algorithms”, Journal of IIETA, Vol. 25, No. 5, pp. 559-568

Kamble Rita R (2014) “A Comparison of Decision Tree Algorithms on Healthy Eating System”, International Journal of Innovative Research in Engineering & Management (IJIREM) Volume-1, Issue-3.

K.T. Anyachebelu, M. U. Abdullahi, M. Abdullahi (2023) "Comparative Analysis of Machine Learning Algorithms for Breast Cancer Prediction", Dutse Journal of Pure and Applied Sciences (DUJOPAS), Vol. 9 No. 4b, https://dx.doi.org/10.4314/dujopas.v9i4b.8.

Ms. Madhuvanthi, et.al, (2024) "Comparative Analysis of Diabetic Prediction Using Machine Learning Algorithms", Journal of Advanced Zoology, ISSN: 0253-7214, Volume 45, Page 465-477

https://towardsdatascience.com

https://www.geeksforgeeks.org/regression-in-machine-learning/

https://www.ibm.com/topics/machine-learning-algorithms

https://www.kaggle.com