Icon

Description automatically generated****Kingdom of Saudi Arabia Ministry of Education

Al Imam Mohammad Ibn Saud University

College of Computer and Information Sciences

Computer Science Department

**Icon

Description automatically generated**

**Tuwaiq Team**

**Estimation of obesity levels**

**A picture containing icon

Description automatically generatedCS 364 Project**

**Presented By:**

|  |  |
| --- | --- |
| **Name** | **Academic number** |
| Rawan Saad Alabdulkarim | 441020373 |
| Muneera Fahad Almuhareb | 441019039 |
| Najd Mohammed Alqabbani | 441018153 |

Dr. Waad Alhoshan

**Date:**

11/2/2023

Table of Contents

[1.Teamwork Table 4](#_Toc126980081)

[2.Gannt Chart Timeline 4](#_Toc126980082)

[3. Introduction 5](#_Toc126980083)

[4. Problem Description 5](#_Toc126980084)

[5. Dataset 5](#_Toc126980085)

[5.1 Dataset Acquisition 5](#_Toc126980086)

[5.2 Dataset Attributes 5](#_Toc126980087)

[5.3 Goal State: 6](#_Toc126980088)

[6. Machine Learning model selection: Random Forest 7](#_Toc126980089)

[7. Models Training & Testing 8](#_Toc126980090)

[7.1 Model Training 8](#_Toc126980091)

[7.1.1 Feature selection 8](#_Toc126980092)

[7.1.2 Hyperparameter Tuning (GridsearchCV) 8](#_Toc126980093)

[7.2 Model Testing 10](#_Toc126980094)

[7.2.1 Confusion Matrix 11](#_Toc126980095)

[7.2.2 Classification Report 11](#_Toc126980096)

[8. Results & Discussion 13](#_Toc126980097)

[8.1 Performance Results 13](#_Toc126980098)

[8.1.1 Without Feature selection 13](#_Toc126980099)

[8.1.2 Feature selection 13](#_Toc126980100)

[8.1.3 Analysis figures 14](#_Toc126980101)

[8.2 Discussion 15](#_Toc126980102)

[Conclusion 15](#_Toc126980103)

[References 16](#_Toc126980104)

Table of Figures

Figure 1: Timeline………………………………………………………………..…………….4

Figure 2: Obesity Estimations………………………………………………………..……...6

##### Figure 3: Random Forest classifier…………………………………………………......…...7

##### Figure4: Important Features……………………………………………………….….….....8

##### Figure5: Min Samples Leaf……………………………………………………..….………10

Figure 6: confusion matrix………………….…………………………….…...…….….….11

##### Figure 7: without hyperparameter (GridSearchCV)………………………..…….….….14

##### Figure 8: with hyperparameter (GridSearchCV)………………………..……………….14

Table of Tables

Table 1: Teamwork table……………………………………………………………………..4

##### Table 2: Features Table……………………………………………………………………….5

#### Table 3: performance without feature selectin……………...………………………………….13

#### Table 4: performance with feature selection………………………………………………...…13

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Technical Reporting |  | Teamwork |  |  |
| **Tasks** | Rawan | Muneera | Najd |
| Project Overview | Problem description |  |  |  |
| Gannt chart timeline |  |  |  |
| Dataset Description | Dataset description |  |  |  |
| Obesity levels figure |  |  |  |
| ML Model Selection | Random forest selection |  |  |  |
| Implementation | Describe the technical framework |  |  |  |
| Evaluation & Results Discussion | Performance discussion |  |  |  |
| Conclusion | Summary of the project |  |  |  |

# 1.Teamwork Table

Table 1: Teamwork table

# 2.Gannt Chart Timeline

**Diagram

Description automatically generated** We proposed the following timeline to complete our project phases which are:

Figure 2: Timeline

# 3. Introduction

Obesity poses physical and mental health risks, that reduce the level of quality of life and well-being, and it also negatively impacts social well-being. Among its causes are bad eating habits, excessive eating with lack of movement, and there are other factors such as genetic diseases and hypothyroidism.

Certainly, we are not ignorant of its physical dangers. Obesity causes many diseases such as heart disease and strokes.

# 4. Problem Description

The problem is to estimate the level of obesity of an individual, based on their ages, eating habits, their lifestyle in general and many health issues.

# 5. Dataset

## 5.1 Dataset Acquisition

“Level Of Obesity” Dataset from Kaggle [1] consist of 2111 record and 16 features.

## 5.2 Dataset Attributes

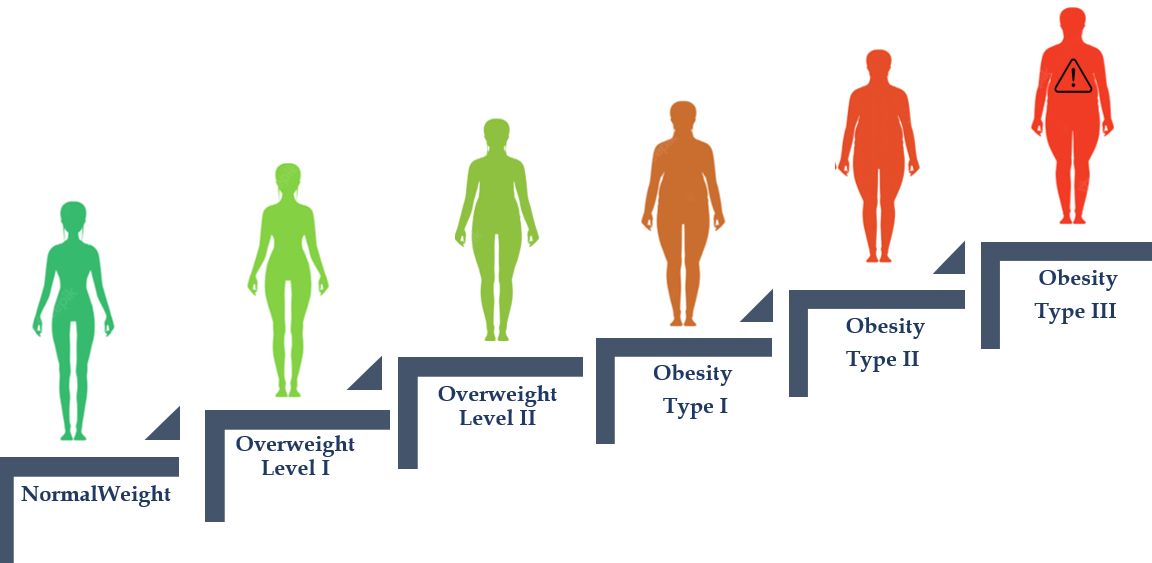
Our dataset has two types numeric and text features [12]:

|  |  |  |  |
| --- | --- | --- | --- |
| Features | | | |
| No. | Text | **No.** | Numeric |
| 1 | Gender | **9** | Age |
| 2 | Family history with overweight | **10** | Height |
| 3 | Frequency consumption of high caloric food (FAVC) | **11** | Weight |
| 4 | Consumption of food between meals (CAEC) | **12** | Frequency of consumption of vegetables (FCVC) |
| 5 | Smoke | **13** | Number of main meals (NCP) |
| 6 | Calories consumption monitoring (SCC) | **14** | Consumption of water daily (CH2O) |
| 7 | Consumption of alcohol (CALC) | **15** | Physical activity frequency (FAF) |
| 8 | Transportation used (MTRANS) | **16** | Time using technology devices (TUE) |

##### Table 2: Features Table

1. The gender value is: Male or Female.
2. The family history with overweight value is: Yes or No.
3. The frequency consumption of high caloric food (FAVC) value is: Yes or No.
4. The consumption of food between meals (CAEC) value is: No or Sometimes or Frequently or Always.
5. The smoke value is: Yes or No.
6. The Calories consumption monitoring (SCC) value is: Yes or No.
7. The consumption of alcohol (CALC) value is: No or Sometimes or Frequently or Always.
8. The transportation used (MTRANS) value is: Automobile or Motorbike or Bike or Public Transportation or Walking.
9. The age is numeric value.
10. The Height is numeric value in meters.
11. The Weight is numeric value in kilograms.
12. The Frequency of consumption of vegetables (FCVC) is numeric value.
13. The Number of main meals (NCP) is numeric value.
14. The Consumption of water daily (CH2O) is numeric value.
15. The Physical activity frequency (FAF) is numeric value.
16. Time using technology devices (TUE) is numeric value.

## 5.3 Goal State:

Our aim is to classify the Level of Obesity of individuals based on estimation features given. in addition, there is an "insufficient weight", which indicates that the model does not have information to determine the obesity level. the levels are:

##### Figure 2: Obesity Estimations

Our main goal is to build a ML model, train it and make it estimate the Level of Obesity of individuals, by **Random Forest** model.

# 6. Machine Learning model selection: Random Forest

As one of the supervised machine learning algorithms that is frequently used in classification and regression, we have selected the Random Forest classification model for our project. It is integrated into multiple decision trees as opposed to just one, and the decision tree with the most votes is chosen based on each tree's projected outcome. One of a random forest's key characteristics is its ability to handle data sets with both continuous and categorical variables for classification and regression. Because of its implementation flexibility and our prior experience, we chose Random Forest [11] .

**Diagram, radar chart

Description automatically generated with medium confidence**

##### Figure 3: Random Forest classifier

# 7. Models Training & Testing

In this section, we present the training procedures for adapting the selected model **random forest classifier.**

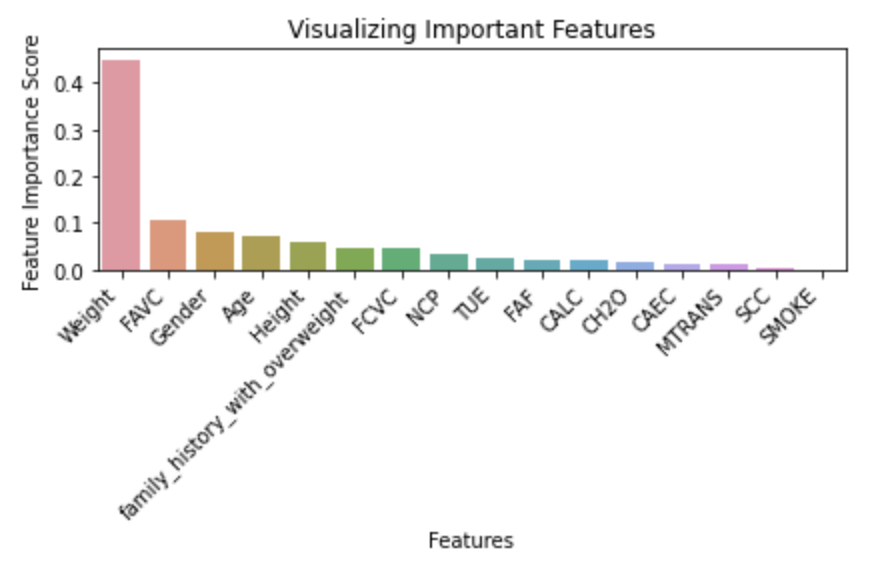
**We used Google colab to implement our model along with detailed notes and the source code of built model, you can see instructions on compiling/running the program in reference [2].**

## 7.1 Model Training

As the common approach to split the dataset, the training set has 70% of the dataset. Also, the random forest is trained by the bagging method (bootstrap aggregating). This involves randomly sampling subsets of the training data, fitting model to these smaller datasets, and aggregating the predictions. Each random sample is the same size as the dataset and sampling with replacement. Random forest method introduces more randomness and diversity by applying a bagging method to the feature space. That is, instead of greedily searching for the best predictor to create branches, the elements of the predictor space are randomly sampled, adding more diversity, and reducing the variance to the tree at the cost of equal or higher bias. This process is also known as “feature bagging” and it is powerful method that leads to a more robust model [3].

### 7.1.1 Feature selection

We used the feature selection technique, which is a common technique used in ML models provide in general a good predictive performance, low overfitting, and easy interpretability [4].it reduces the number of features and only use the important features. This approach is recommended to reduce the size of the model.

In the figure below we apply the feature selection to our model and shows the feature’s level of the importance feature’s. Some features are strongly affecting the prediction value while others barely relied on while training the model.

##### Figure4: Important Features

We built our model **with** and **without** Feature Selection, **with** and **without** the hyperparameter tuning using GridSearchCV to set the difference of performance between them.

### 7.1.2 Hyperparameter Tuning (GridsearchCV)

We will apply the hyperparameter tuning for retraining for both **with** and

**without** feature selection to improve the results using the **GridsearchCV**.

What is **GridSearchCV**?

**GridSearchCV** is the process of performing hyperparameter tuning in order to determine the optimal values for a given model [5].

**GridSearchCV** tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the **Cross-Validation** method [5] . We do this by defining a dictionary in which we mention a particular hyperparameter along with the values it can take. Since we are going to set values for each parameter and set it as a **param\_grid**, after using **GridsearchCV** it will produce the choosing parameter that will applied in random forest classifier

**Param grid** – A dictionary with parameter names as keys and lists of parameter values [6].

**Parameters are:**

1. **min\_sample\_split**:

a parameter that tells the decision tree in a random forest the minimum required number of observations in any given node to split it.[6]

This Random Forest hyperparameter specifies the minimum number of samples that should be present in the leaf node after splitting a node [6].

1. **max\_features:**

we will observe the effect of the **max\_features** hyperparameter. This resembles the number of maximum features provide depth to each tree in a random forest [6] .

1. **max\_depth**: (default=None)

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than **min\_samples\_split** samples [6] .

1. **Criterion**

{“gini”, “entropy”}, default=” gini”

The function to measure the quality of a split.

**Entropy:**  
It helps us to build an appropriate decision tree for selecting the best splitter. Entropy can be defined as a measure of the purity of the sub split. Entropy always lies between 0 to 1[7] .

**Gini Impurity:**

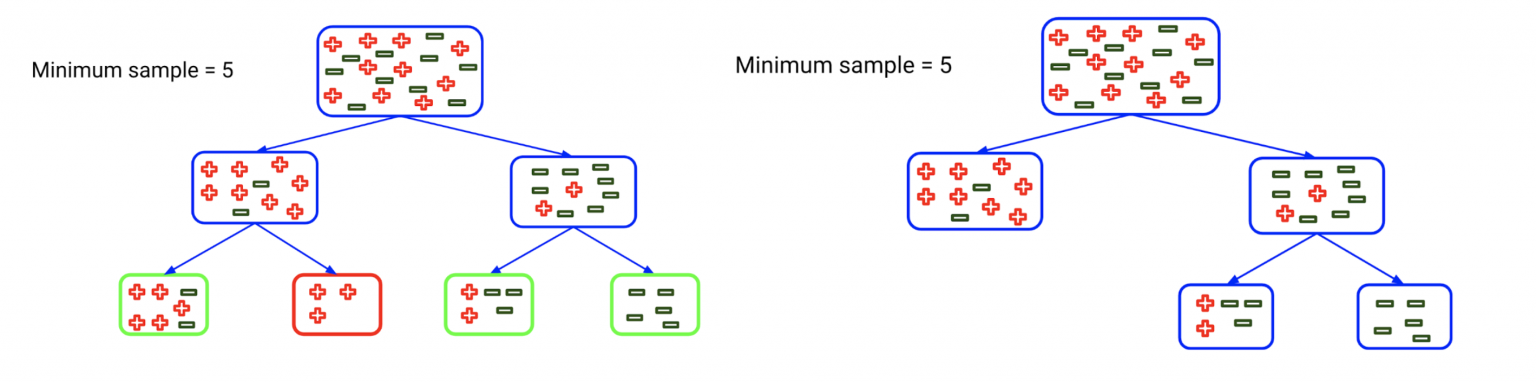
The internal working of Gini impurity is also somewhat similar to the working of entropy in the Decision Tree. In the Decision Tree algorithm, both are used for building the tree by splitting as per the appropriate features but there is quite a difference in the computation of both the methods[7] .

1. **min\_samples\_leaf**

This Random Forest hyperparameter specifies the minimum number of samples that should be present in the leaf node**after splitting** a node.

using an example of **min\_sample\_leaf** Let’s say we have set the minimum [6].

samples for a terminal node as 5, the tree will be shown below:



##### Figure5: Min Samples Leaf

That mean the minimum of samples in each leaf node is 5  [6] .

**6-Bootstrap**

If the Bootstrapequal **True**, that mean drawing with replacement, so some data points might be used more than once and others not[6] .

If the Bootstrapequal **False**, it’s going to use all the sample and all the features iteratively from data [6] .

## 7.2 Model Testing

As common approach to split the dataset, the testing set has 30% of the dataset. our model has been fitted to the training set so we can now predict the test result we can fit and evaluate the model on separate chunks of the dataset. we need accuracy score, confusion matrix, and classification report from random forest classifier- sklearn metrics.

### 7.2.1 Confusion Matrix

Is a tabular breakdown of a classifier's incorrect and correct predictions. It is used to assess the effectiveness of a categorization model. It can be used to assess the performance of a classification model by computing performance metrics such as accuracy, precision, recall, and F1-score[8].

TN (True Negative): When a case was to be negative but predicted negative [9].

TP (True Positive): When a case was to be positive but predicted positive [9].

FN (False Negative): When a case was to be positive but predicted negative [9].

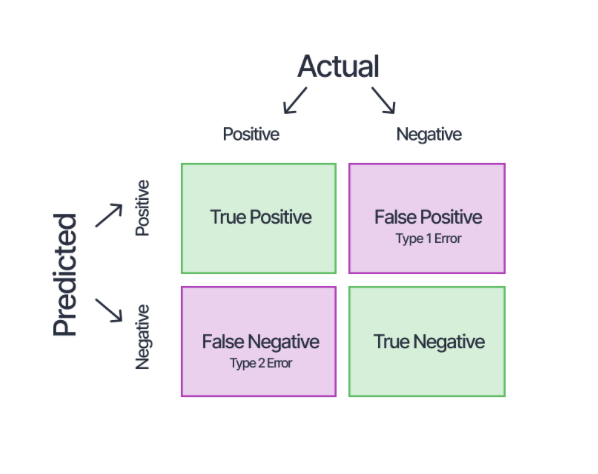
FP (False Positive): When a case was to be negative but predicted positive [9].

Figure 6: confusion matrix

### 7.2.2 Classification Report

It is one of the performance evaluation metrics of a classification-based machine learning model. It displays your model’s precision, recall, F1 score and support. also displays metrics like accuracy, macro average, and weighted average [8].

**Precision**:

The percentage of positive identifications that were truly right.

A model which produces no false positive has a precision of 1.0 [10].

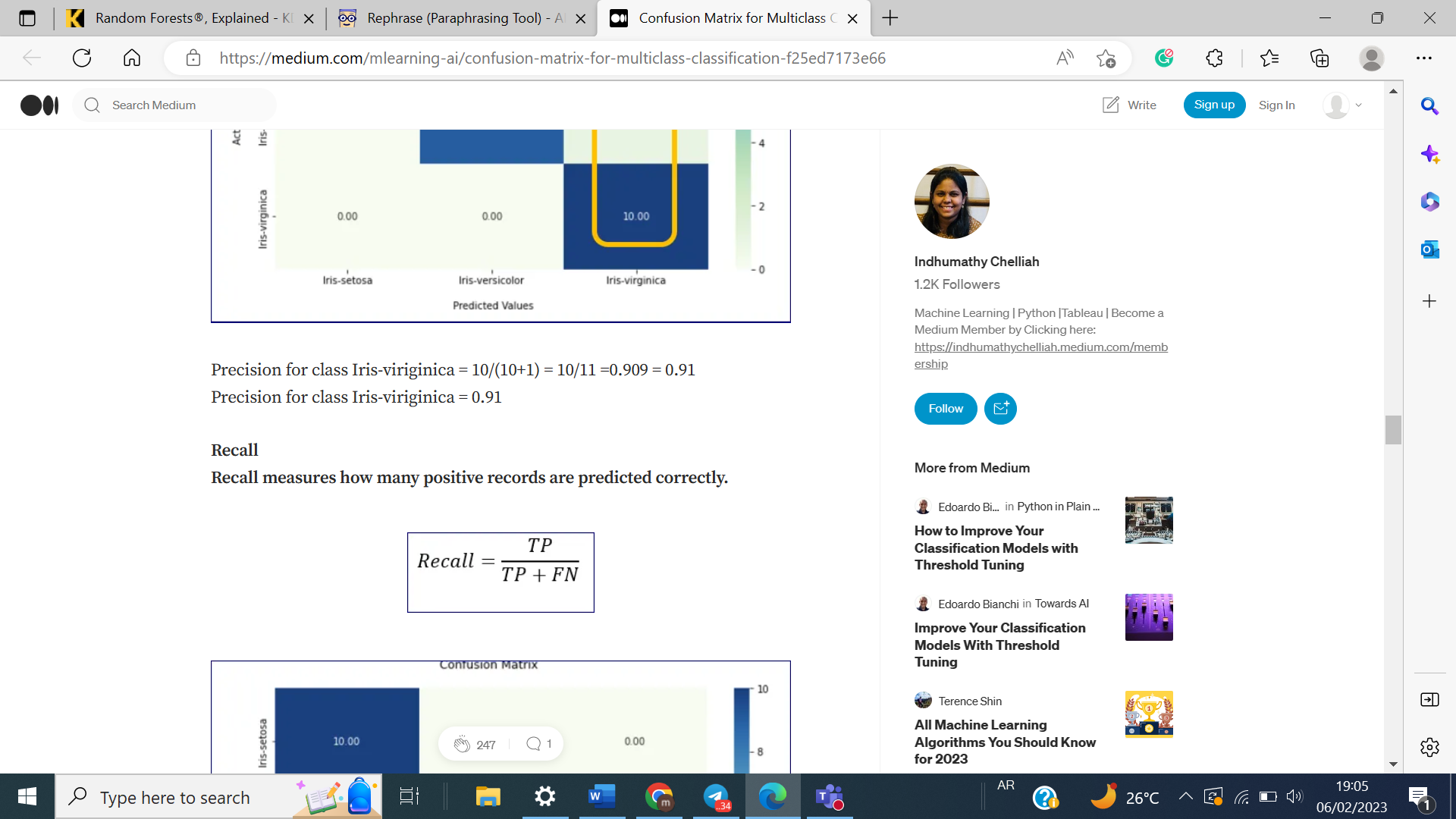
Text

Description automatically generated with medium confidencePrecision is defined as follows [8]:

**Recall:**

The percentage of true positives that were correctly classified.

which produces no false negatives has a recall of 1.0 [10].

Recall is defined as follows [8]:

**F1- score** :

Combination of precision and recall .

A perfect model achieves an F1 score of 1.0 [10].

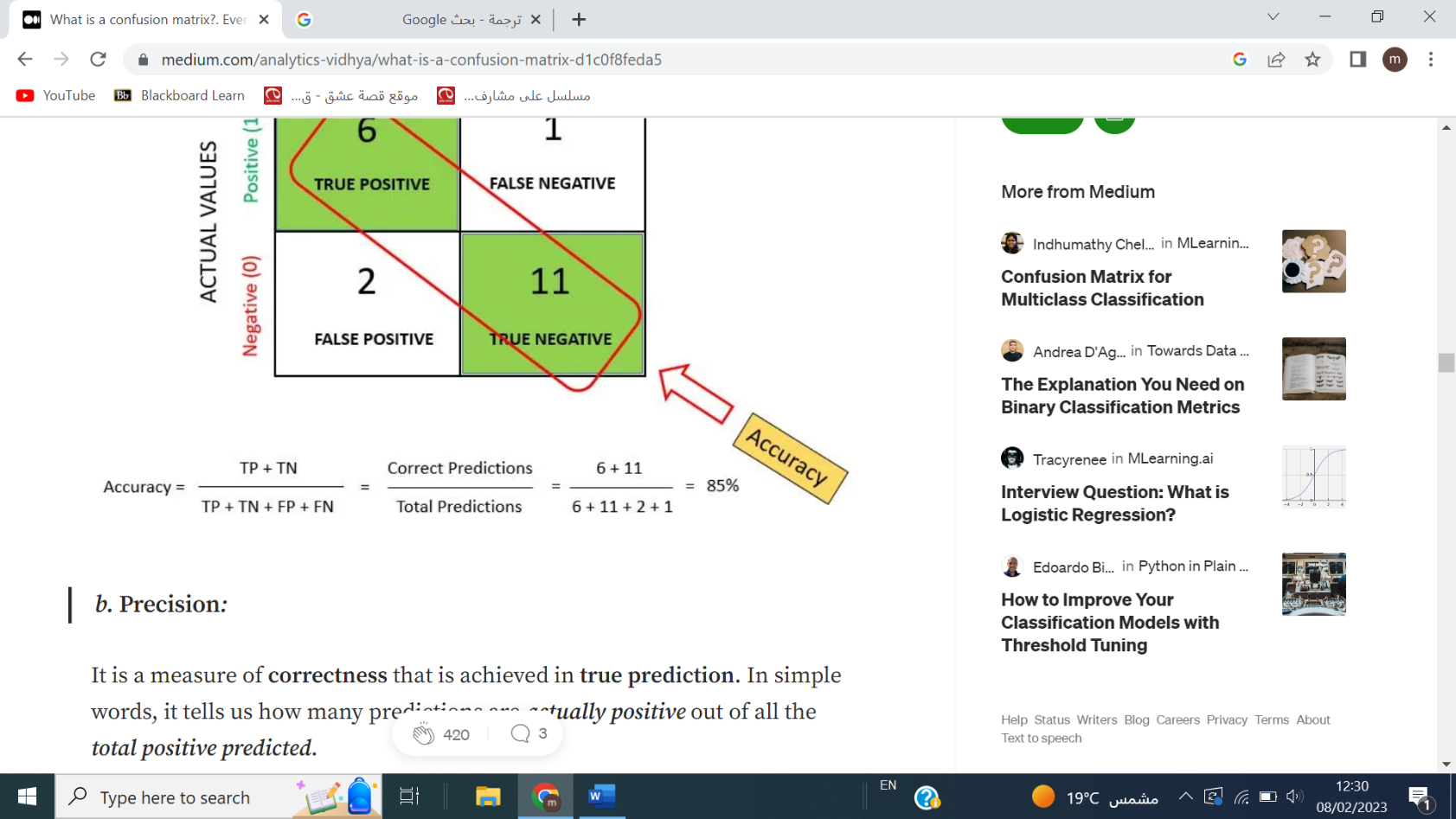
Text

Description automatically generatedF1- score is defined as follows [8]:

**Accuracy**:

The accuracy of the model in decimal form.

Perfect accuracy is equal to 1.0 [10].

Accuracy is defined as follows [8]:

**Support:** the number of samples used to calculate each metric [10].

**Macro average recall:**

Are calculated from the arithmetic mean of all per-class recall scores

[Sum of all the per class recall scores divided by the number of classes] [8].

**Macro average precision:**

 Are calculated from the arithmetic mean of all the per-class precision scores.

[Sum of all the per class precision scores divided by the number of classes] [8].

**Macro average F1 scores**:

 Are calculated from the arithmetic mean of all the per class F1 scores.

[Sum of all the per class f1-scores divided by the number of classes] [8].

**Weighted** **average:**

 Is calculated by taking the mean of per class scores while considering the proportion of each class support.

To compute the weighted average of all classes with the same level of support

The proportion of each class's support is equal to Support of that class/Total support.

[If support varies for different classes, then must calculate the proportion for each class separately] [8].

# 8. Results & Discussion

We are going to present the results and our perspectives on the performance results of the trained models described in the previous section.

## 8.1 Performance Results

### 8.1.1 Without Feature selection

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Without feature selection** | | | | |
| **With Hyperparameter** | | | **Without Hyperparameter** | |
|  | **Train** | **Test** | **Train** | **Test** |
| **1** | 96.2762 | 91.1672 | 100 | 95.4259 |
| **2** | 95.9377 | 92.2713 | 100 | 95.5836 |
| **3** | 96.4116 | 92.1136 | 100 | 95.2681 |
| **4** | 95.9377 | 91.0095 | 100 | 95.1104 |
| **5** | 96.5471 | 92.5868 | 100 | 95.7413 |
| **6** | 95.3961 | 91.0095 | 100 | 94.9527 |
| **7** | 96.0054 | 92.7445 | 100 | 95.2681 |
| **8** | 96.3439 | 92.2713 | 100 | 95.5836 |
| **9** | 96.6825 | 92.9022 | 100 | 95.2681 |
| **10** | 96.1408 | 93.0599 | 100 | 95.7413 |
| **Average:** | 96.1679 | 92.11358 | 100 | 95.39431 |

#### Table 3: performance without feature selection

### 8.1.2 Feature selection

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **With feature selection** | | | | |
| **With Hyperparameter** | | | **Without Hyperparameter** | |
|  | **Train** | **Test** | **Train** | **Test** |
| 1 | 94.9898 | 91.0095 | 100 | 96.53 |
| 2 | 95.5992 | 91.9558 | 100 | 96.6877 |
| 3 | 95.1253 | 92.1136 | 100 | 96.0568 |
| 4 | 95.4638 | 92.7445 | 100 | 96.2145 |
| 5 | 95.5992 | 91.7981 | 100 | 96.0568 |
| 6 | 95.3961 | 92.429 | 100 | 96.0568 |
| 7 | 95.5315 | 92.1136 | 100 | 96.0568 |
| 8 | 95.5315 | 91.3249 | 100 | 96.6877 |
| 9 | 95.3961 | 92.5868 | 100 | 95.4259 |
| 10 | 95.6669 | 92.1136 | 100 | 96.53 |
| **Average:** | 95.42994 | 92.01894 | 100 | 96.2303 |

### 

#### Table 4: performance with feature selection

### 8.1.3 Analysis figures

:

##### Figure 7: without hyperparameter (GridSearchCV)

##### Figure 8: with hyperparameter (GridSearchCV)

## 8.2 Discussion

As we explained above, **with** and **without** feature selection we noticed in both that the **training model** reaches **100%**, which means that there is overfitting.

**Avg test accuracy without feature selection without hyperparameter tuning: 95.39431%**

**Avg test accuracy with feature selection without hyperparameter tuning: 96.2303%**

With unseen data it was better with feature selection, as it reduced the gap between testing and training model, by using **feature\_importance**

**Feature importance** is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples. The higher the value the more important the feature. However, this happens technically within the code [13].

And to solve the problem of overfitting, we decided to do the optimization through the hyperparameter tuning using GridsearchCV, and one of the solutions used is determine the depth values.

**Avg training accuracy with feature selection with hyperparameter tuning: 95.42994%**

**Avg test accuracy with feature selection with hyperparameter tuning: 92.01894%**

**Avg training accuracy without feature selection with hyperparameter tuning :96.1679%**

**Avg test accuracy without feature selection with hyperparameter tuning :92.11358%**

This enhances performance, even though the differences are very small.

So, with feature selection and the hyperparameter tuning, the performance will be better because the hyperparameter tuning without feature selection may take a long time

We have benefited from applying feature selection, which will reduce the time complexity of the model and less possibility of overfighting [14].

# Conclusion

In this report, we presented our work in this project an Obesity Level Estimator application using Random Forest model. We faced several challenges since it is our first time to deal with Random Forest model, also the time taken to find a suitable dataset to implement our project, search/understanding the python libraries and function. After searching and learning we tries to train the model perfectly. Overall results shows that the model shows good prediction results with average of **95.42994%** trainaccuracy and **92.01894%**test accuracy.

# References

**Note: There are some definitions copied from the text, while others are part of the reading and understanding of the course.**

1-<https://www.kaggle.com/datasets/jayitabhattacharyya/estimation-of-obesity-levels-uci-dataset>

2- <https://colab.research.google.com/drive/10GHWQ9yS-cdOKv5YEaOPQ6ujoyVaH_EY>

3- <https://www.kdnuggets.com/2017/10/random-forests-explained.html>

4- <https://towardsdatascience.com/feature-selection-using-random-forest-26d7b747597f>

5- <https://www.mygreatlearning.com/blog/gridsearchcv/>

6-<https://www.analyticsvidhya.com/blog/2020/03/beginners-guide-random-forest-hyperparameter-tuning/>

7- <https://www.geeksforgeeks.org/gini-impurity-and-entropy-in-decision-tree-ml/>

8- <https://medium.com/mlearning-ai/confusion-matrix-for-multiclass-classification-f25ed7173e66>

9- <https://muthu.co/understanding-the-classification-report-in-sklearn/>

10- <https://youtu.be/XiUlqN1Ay0U>

11-<https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501>

12- <https://www.sciencedirect.com/science/article/pii/S2352340919306985?via%3Dihub>

13- <https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3>

14- <https://www.kaggle.com/questions-and-answers/264115>