
**Faculty of Engineering & Technology**
**Electrical & Computer Engineering Department**

# MACHINE LEARNING AND DATA SCIENCE - ENCS5341

## Assignment #3 - Project

**Prepared by:**
Osaid Hamza
Student ID: 1200875
Section: 2

Yazeed Hamdan
Student ID: 1201133
Section: 2

**Instructor**: Dr. Yazan Abu Farha

**Date**: 25/1/2024

# Introduction

In this project, we are focusing on building a predictive model using a dataset about obesity. Our goal is to create a model that can predict obesity categories effectively based on various factors like age, height, weight, and more. This work aims to understand better what contributes to obesity and demonstrates how machine learning can be applied in health-related areas.

**Models Explored**

In this project, three different types of models were used:

K-Nearest Neighbors (KNN): This is a simple model that makes predictions based on the closest data points. We tried it with different settings (k=1 and k=3) to see how it performs.

Random Forest: This model works by creating several decision trees and combining their results. It's known for being accurate and reliable. We adjusted the number of trees (n_estimators) to get the best results.

Support Vector Machine (SVM): SVM is a strong model that's particularly good for datasets with many features. It can be customized with different kernel functions. We fine-tuned its C parameter, which helps control the model's complexity, to improve its performance.

**Evaluation Metrics**

To evaluate the performance of our models, we employed several metrics, based on the nature of our classification task and the characteristics of our dataset:

Accuracy: This is like checking how many answers the model got right. It's a quick way to see if the model is doing well, but we have to be careful with it, especially when some types of outcomes are more common than others in our data.

Precision and Recall: These are two ways to look more closely at the model's performance. Precision tells us how accurate the model's predictions are (like how many detected cases are actual cases), while recall tells us how good the model is at finding all the true cases.

F1-Score: This score mixes precision and recall into one number. It's handy when we want to balance being accurate with being thorough.

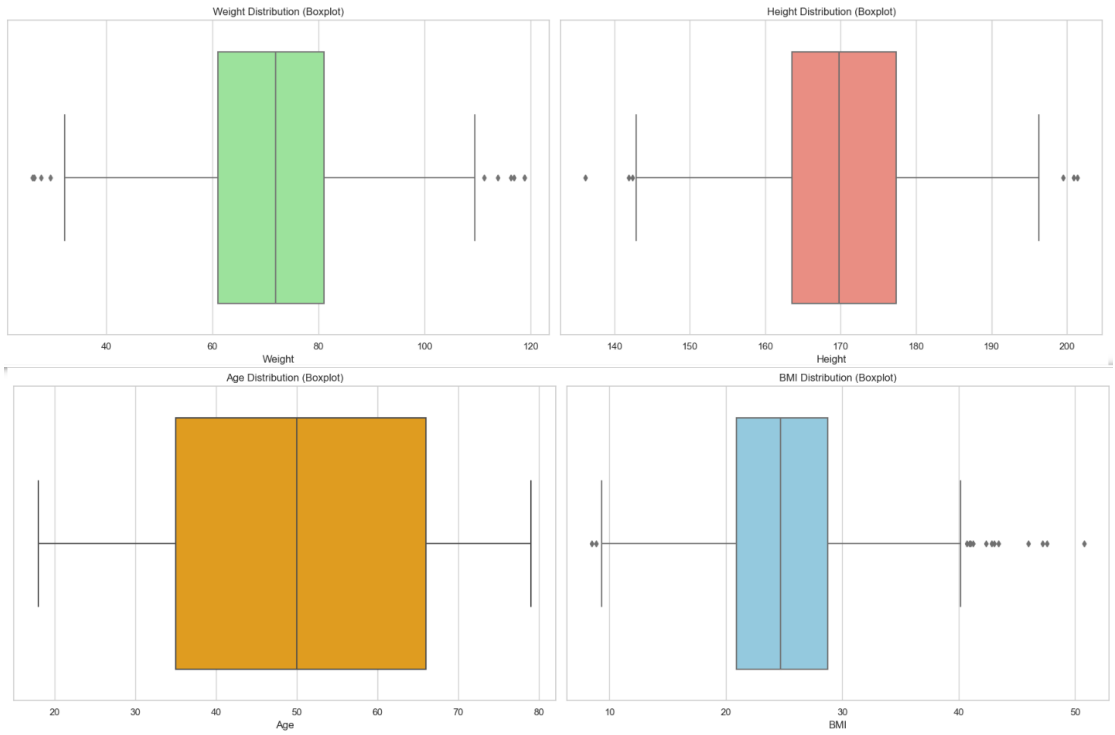
Confusion Matrix: This is a detailed chart that helps us see exactly where the model is getting things right and where it's making mistakes.

All these methods were used to get a full picture of how well each model worked. Our goal was to find models that not only guessed right most of the time but also did so in a way that's reliable and strong for all different kinds of situations in our data.

# Dataset

The dataset utilized in this study offers a comprehensive view of various attributes potentially linked to obesity. It includes a mix of numerical and categorical variables with seven features and 1000 example, such as age, height, weight, BMI, PhysicalActivityLevel, and Gender. The target variable, ObesityCategory, classifies individuals into four different obesity categories, making this a classification task. The following charts shows the percentage of the examples in each feature:


The following Boxplots shows the ranges and outliers of Whight, Hight, Age, BMI features:



The following output shows Feature ranges from minimum value to the maximum for the Dataset:

```
Feature Ranges:
       Age      Height      Weight        BMI  PhysicalActivityLevel
min  18.0  136.115719   26.065730   8.470572                    1.0
max  79.0  201.419670  118.907366  50.791898                    4.0
```

The following output shows Descriptive statistics for the Dataset features:

```
Descriptive Statistics:
                Age       Height       Weight          BMI  \
count  1000.000000  1000.000000  1000.000000  1000.000000
mean     49.857000   170.052417    71.205769    24.888317
std      18.114267    10.309971    15.509849     6.193912
min      18.000000   136.115719    26.065730     8.470572
25%      35.000000   163.514205    61.129629    20.918068
50%      50.000000   169.801665    71.929072    24.698647
75%      66.000000   177.353596    81.133746    28.732132
max      79.000000   201.419670   118.907366    50.791898

       PhysicalActivityLevel
count            1000.000000
mean                2.534000
std                 1.116284
min                 1.000000
25%                 2.000000
50%                 3.000000
75%                 4.000000
max                 4.000000
```

In our Dataset we changed the Gender from words to numbers. Female entries changed to 0 and all Male entries would be 1.
similar process was done with the target ObesityCategory by assigning numbers to different categories of obesity: Underweight is 1, Normal weight is 2, Obese is 3, and Overweight is 4. This way, instead of words, we have a simple number that represents each category.

This standardization of the Dataset features is performed using the StandardScaler. Standardization involves transforming the data in these columns so that they have a mean of 0 and a standard deviation of 1, making it easier for machine learning algorithms to work with them. The code creates a new DataFrame called data_scaled that contains the same data as the original data but with the specified columns scaled. This ensures that the scaled data is ready for further analysis or machine learning modeling, where standardization can improve the performance and interpretability of the models. The following tables shows how the scaled data become for the first 5 rows after scaling.

Out[27]:

| | Age | Gender | Height | Weight | BMI | PhysicalActivityLevel | ObesityCategory |
|---|---|---|---|---|---|---|---|
| **0** | 0.339295 | 1 | 0.341864 | 0.050076 | -0.160970 | 1.313943 | 2 |
| **1** | 1.057320 | 1 | -0.574985 | 1.209739 | 1.374115 | -0.478612 | 3 |
| **2** | -0.213033 | 0 | -0.192164 | 0.111266 | 0.150129 | 1.313943 | 4 |
| **3** | -0.986291 | 1 | -0.154567 | 0.882535 | 0.811514 | 0.417665 | 4 |
| **4** | 0.560226 | 1 | 1.311635 | -0.139776 | -0.710797 | 0.417665 | 2 |

After done these steps, we're making sure our data is clean, consistent, and ready for the next steps where we'll use machine learning models to find patterns and make predictions.

# Experiments and Results

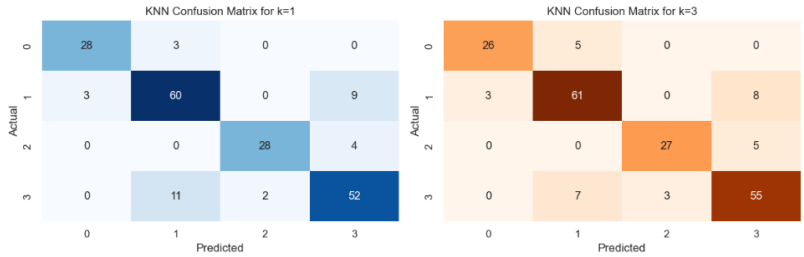
### 1- K-Nearest Neighbors (KNN)

Our first set of experiments involved the K-Nearest Neighbors (KNN) algorithm. We tested two configurations of the KNN model by setting the number of neighbors k to 1 and 3, respectively. In this part the data was divided into two sections: testing, training.

KNN with k=1: The model was perfect in learning from the training data (Training Accuracy: 1.00), which means it got every single training example right. However, this perfection didn't hold when we tested it with new data it hadn't seen before (Testing Accuracy: 0.84), which suggests it may have memorized the training data too well.

KNN with k=3: This model didn't get a better score than KNN when k=1 on the training data (Training Accuracy: 0.95375), but it still did really well, and it performed slightly better on the test data (Testing Accuracy: 0.845) compared to k=1. This tells us it might be better at handling data it hasn't seen before.
The following Figure shows the performance results after apply KNN with K=1 and K=3 model in the Dataset and shows the two confusion matrix for both models.

```
Performance for k=1:
Training Accuracy: 1.0
Testing Accuracy: 0.84
              precision    recall  f1-score   support

           1       0.90      0.90      0.90        31
           2       0.81      0.83      0.82        72
           3       0.93      0.88      0.90        32
           4       0.80      0.80      0.80        65

    accuracy                           0.84       200
   macro avg       0.86      0.85      0.86       200
weighted avg       0.84      0.84      0.84       200

Performance for k=3:
Training Accuracy: 0.95375
Testing Accuracy: 0.845
              precision    recall  f1-score   support

           1       0.90      0.84      0.87        31
           2       0.84      0.85      0.84        72
           3       0.90      0.84      0.87        32
           4       0.81      0.85      0.83        65

    accuracy                           0.84       200
   macro avg       0.86      0.84      0.85       200
weighted avg       0.85      0.84      0.85       200
```



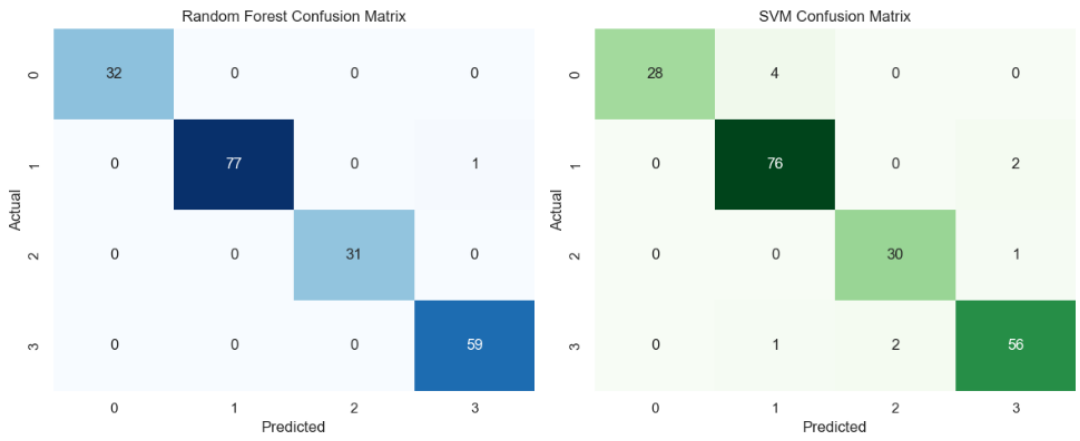

The previous figure shows Accuracy values for each model of KNN and Classification Metrics which includes Precision, Recall, F1-Score and 4x4 confusion matrix because the Dataset have four possible values for the Target variable.

## 2- Random forest and SVM models

Random Forest model was experimented by changing the number of trees (n_estimators). We tried with 10, 20, 50, and 100 trees. We found that 20 trees gave us the best results on the validation data, which is a separate part of the data we use to check how good our model is. The model with 20 trees got 97.5% of the predictions right, which was the highest validation accuracy found in results. For the SVM, we adjusted the C parameter, which controls how strictly the model follows the training data. If C is too high, the model might be too strict and won't adapt well to new data. If it's too low, it might be too relaxed and could make more mistakes. We tried values of 0.1, 1, 10, and 100. The value C=10 worked the best, with 86% validation accuracy.
In this part the data is divided into three sections here: testing, training , and validation, to find the best hyperparameter with higher accuracy from the Validation Set. The following figure shows Accuuracy results for both models:

```
Random Forest Accuracies for Different n_estimators:
n_estimators=10, validation Accuracy: 0.965
n_estimators=20, validation Accuracy: 0.975
n_estimators=50, validation Accuracy: 0.965
n_estimators=100, validation Accuracy: 0.97

Best Parameters for Random Forest: {'n_estimators': 20} with higher Accuracy

SVM Accuracies for Different C Values:
C=0.1, validation Accuracy: 0.505
C=1, validation Accuracy: 0.845
C=10, validation Accuracy: 0.86
C=100, validation Accuracy: 0.845

Best Parameters for SVM: {'C': 10} with higher Accuracy

Random Forest Test Set Accuracy: 0.99
SVM Test Set Accuracy: 0.87
```

After we found the best Accuuracies for both models, we tested them on new data they hadn't seen before (the test set). The Random Forest model with 20 trees was very accurate, getting accuracy with 99% . The SVM with C=10 was also quite good, with an accuracy of 87%.




The previos figure shows the 4x4 Confusion matrix for both SVM and Random forest models.

# Analysis

## 1- KNN ( K=1, K=3)

Based on the results in the previous part, which include performance metrics and confusion matrices for the K-Nearest Neighbors (KNN) algorithm with k=1 and k=3

**KNN with k=1:**

Testing Accuracy was 0.84, which may reflect poor performance due to the possibility of overfitting, and that's because the model classifies only on the 1-nearest neighbor

Classification Metrics: The precision, recall, and f1-score across the different classes are generally high.

Confusion Matrix: The darker shades on the diagonal indicate a high number of correct predictions for each class. The elements on the diagonal represnts the number of correctly classified examples, which were high, and this indicates a good performance.

**KNN with k=3:**

Testing Accuracy: Testing Accuracy = 84.5%, it shows that the model with k=3 generalizes slightly better to the test set.

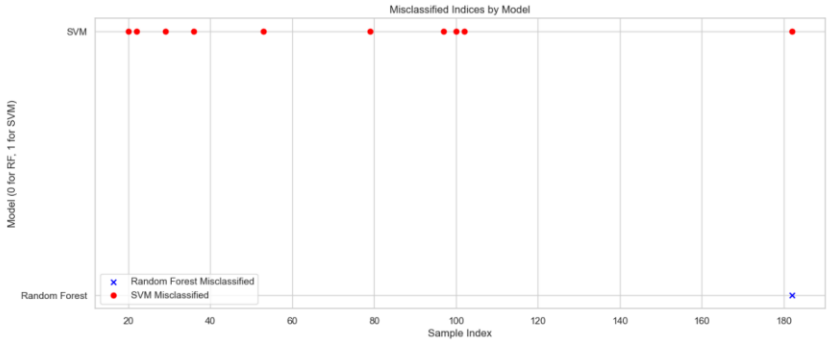
Classification Metrics: The precision, recall, and f1-score were used as the figure in results part shows such that they reflect good performance.

Confusion Matrix: Shows some misclassifications, particularly between classes 1 and 3, but the model with k=3 seems to make fewer mistakes overall compared to k=1.

## 2- Random Forest and SVM models

The following Results shows the missclassified Indices for both models and it can be noticed that random forest has only one missclassified example and SVM have 10 missclassified examples from testing set.

```
Misclassified Indices for Random Forest: [182]
Misclassified Indices for SVM: [ 20  22  29  36  53  79  97 100 102 182]
Classification Report for Random Forest:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        32
           2       1.00      0.99      0.99        78
           3       1.00      1.00      1.00        31
           4       0.98      1.00      0.99        59

    accuracy                           0.99       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      0.99      1.00       200


Classification Report for SVM:
              precision    recall  f1-score   support

           1       1.00      0.88      0.93        32
           2       0.94      0.97      0.96        78
           3       0.94      0.97      0.95        31
           4       0.95      0.95      0.95        59

    accuracy                           0.95       200
   macro avg       0.96      0.94      0.95       200
weighted avg       0.95      0.95      0.95       200
```




The plot above indicates specific instances where each model (Random Forest and SVM) incorrectly predicted the class of a data point in the test set. The x-axis represents the sample index in the test set. The y-axis differentiates between the two models. Red dots represent instances where SVM made incorrect predictions, and blue crosses represent the same for Random Forest.

The spread of red dots at different points along the x-axis suggests SVM misclassified several different instances. The single blue cross indicates that Random Forest had a very low number of misclassifications, only one in this case.

For Random Forest, the classification report shows nearly perfect scores across precision, recall, and f1-score for all classes, with an overall accuracy of 0.99. This means Random Forest was almost best than the SVM in its predictions.

For SVM, while still high, the scores in precision, recall, and f1-score are slightly lower, and the overall accuracy is 0.87. SVM performed well, but not as well as Random Forest.

## Conclusions and Discussion

Our analysis led to the conclusion that the Random Forest model is better than the Support Vector Machine (SVM) model and kNN for our dataset. The Random Forest model showed higher accuracy and fewer misclassifications, as evidenced by the confusion matrices. This superiority could be attributed to Random Forest 's ensemble approach, which combines multiple decision trees to produce a more generalized and robust model. Random Forest tends to perform well when there are complex relationships and interactions between features, which appears to be the case with our dataset.

The SVM model, while still effective, did not achieve the same level of accuracy and correctly classified examples compared with the Random Forest. This may be due to SVM's sensitivity to the choice of kernel and the regularization parameter C. Finding the optimal hyperplane for multi-class classification can be challenging, especially when the data is not linearly separable or if the optimal hyperparameters are not identified.

In our KNN analysis, we found that a k value of 3 yielded better results than k=1. While k=1 achieved high training accuracy, this was a sign of overfitting, with the model being too closely tailored to the training data. By increasing k to 3, we introduced more generalization, reducing overfitting and improving the model's performance on the test data. This suggests that incorporating more neighbors into the decision process helped to make more robust predictions.

In conclusion, the Random Forest gave the least number of misclassified examples, an higher accuracy more than the other models.