Multi-Class Prediction of Obesity Risk - IART (Final Delivery)

Rubem Neto - 202207086
Diogo Goiana - 202207944
Leandro Martins - 202208001

Problem Definition

Problem Statement

- The main objective of the competition is to build a predictive model to classify individuals into multiple obesity risk classes

Target Variable: NObeyesdad

- The target variable (NObeyesdad) indicates the **obesity level** and includes the following categories:
  1. Insufficient Weight
  2. Normal Weight
  3. Overweight Level I and II
  4. Obesity Type I, II and III

Dataset Description

Name	Data Type	Column Name
Age	Integer	Age
Gender	Categorical	Gender
Height	Numeric	Height
Weight	Numeric	Weight
Family History	Categorical	family_history_with_overweight
Frequent consumption of high-calorie food	Categorical	FAVC
Frequency of Vegetable Consumption	Numeric	FCVC
Main meals per day	Numeric	NCP
Consumption of food between meals	Categorical	CAEC
Individual smokes	Categorical	SMOKE
Daily Water Consumption	Numeric	CH20
Monitoring Calorie Consumption	Categorical	SCC
Frequency of physical activity	Numeric	FAC
Time spent using technology devices	Numeric	TUE
Frequency of alcohol consumption	Categorical	CALC
Main mode of Transportation	Categorical	MTRANS

Related Work

Amialito's Lopez Solution

- Solution with hyperparameter tuning using Optuna
- https://github.com/amaliogomezlopez/Obesity-Classification?utm_source=chatgpt.com

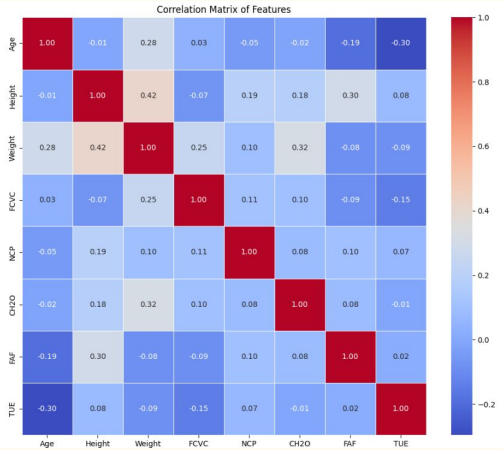
Understanding "NObeyesdad"

- Kaggle discussion explaining the NObeyesdad target variable
- https://www.kaggle.com/competitions/playground-series-s4e2/discussion/477095

Methodology

1. **Data Preprocessing**
    a. Handle missing values, outliers, and categorical encoding
    b. Normalize numerical values like height and weight
2. **Feature Selection**
    a. Derive new features (e.g., BMI from height and weight)
    b. Reduce dimensionality
3. **Model Selection**
    a. Try multiple classifiers
    b. Use ensemble methods for better performance
4. **Model Tuning**
    a. Cross-validation to optimize hyperparameters
    b. Apply class weighting to balance the influence of minority classes
5. **Model Evaluation**

Work Progress



Data Analysis and Preprocessing:

- Explored dataset structure and verified no missing/duplicate values
- Generated descriptive stats and correlation matrix (see right)
- Dropped non-informative ID column
- Prepared data for subsequent modeling steps

Initial Models:

- Decision Trees
  - Splits data into branches to make decisions based on feature values
- Neural Networks
  - Mimics the human brain to detect complex patterns using layers of nodes
- k-Nearest Neighbors (k-NN)
  - Classifies new data based on the majority label of its closest neighbors
- Support Vector Machines (SVM)
  - Finds the optimal boundary that best separates classes in the feature space

Decision Tree

Overview:

A tree-structured model that splits data based on feature values to make decisions. It's easy to interpret and works well for both classification and regression tasks.

Chosen Parameters:

- ccp_alpha = 0.01
  - Enables cost-complexity pruning to reduce overfitting by simplifying the tree

Extra Analysis:

We used RFE to remove features that didn't improve—or even hurt—the model's performance. By keeping only the important ones, the model became simpler and more accurate.

Interestingly, RFE selected just 3 features: BMI, Weight, and Gender. However, it took longer to run because it tested many feature combinations. In the end, the final feature set matched what the Decision Tree would have chosen naturally.

k-Nearest Neighbors (k-NN)

Overview:

K-Nearest Neighbors (KNN) is a simple, instance-based learning algorithm that classifies a data point based on the majority label of its nearest neighbors. It's intuitive and effective for classification tasks, especially when the dataset is well-scaled and features are relevant.

Chosen Parameters:

- N_neighbors = 8
  - Controls how many neighbors to consider when making predictions. We tested different values over multiple runs, the optimal number fell between 6 and 10, providing the best average accuracy.

Extra Analysis:

We used SelectKBest to evaluate different feature subsets. Although this took more time due to testing many combinations, it helped identify a smaller set of 5 key features.

Using these features improved the model's performance significantly, increasing accuracy and other metrics by around 10%.

Support Vector Machines (SVM)

Overview:

Support Vector Machines (SVM) is a powerful classification algorithm that finds the optimal boundary (hyperplane) separating classes by maximizing the margin between data points.

It works well with high-dimensional data and can be adapted to nonlinear problems using kernels. In this case, we use a linear kernel for interpretability and efficiency.

Chosen Parameters:

- kernel = "linear"
- probability = True (enables probability estimates for predictions)

Extra Analysis:

We used RFE to remove features that didn't improve—or even hurt—the model's performance. By keeping only the important ones, the model became simpler and more accurate.

This process tests all possible combinations of features, making it computationally expensive and time-consuming, so we don't have definite results for now.

Neural Networks

Overview:

Neural Networks, specifically Multi-Layer Perceptrons (MLPs), are powerful models inspired by the brain's structure.

They consist of layers of interconnected nodes (neurons) that learn complex patterns through nonlinear transformations.

This makes them suitable for classification tasks with complex relationships between features.

Chosen Parameters:

- hidden_layer_sizes = (128, 64, 32)
- activation = 'relu'
- solver = 'adam'
- learning_rate = 'adaptive'
- max_iter = 1000
- early_stopping = True

Extra Analysis:

The model requires label encoding for categorical target variables and scaled input features for optimal performance.

Training with early stopping prevents overfitting by monitoring validation performance.

While no explicit feature selection was applied here, tuning architecture and training parameters helped balance accuracy and training time effectively.

Ensemble

Overview:

Combines multiple diverse classifiers to improve overall performance by leveraging their complementary strengths.

Uses our previously defined models with their respective parameters. This approach often yields better accuracy and robustness than individual models.

Chosen Parameters:

- voting='hard'
  - Makes it choose the prediction from the first model incase of a tie.

Extra Analysis:

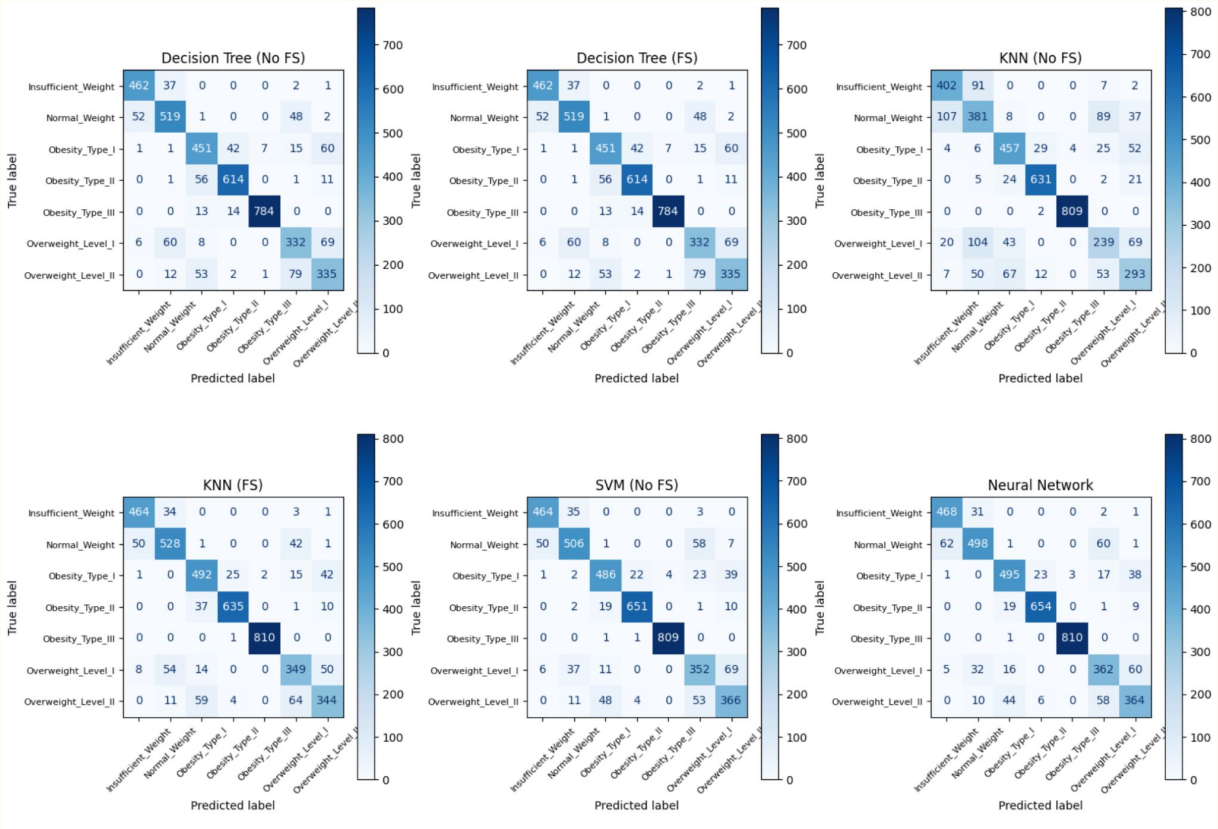
Each individual model is scaled and tuned before ensembling to ensure balanced contributions.

By combining different learning algorithms, the ensemble reduces the risk of individual model biases or errors, resulting in improved accuracy and generalization on the test data.

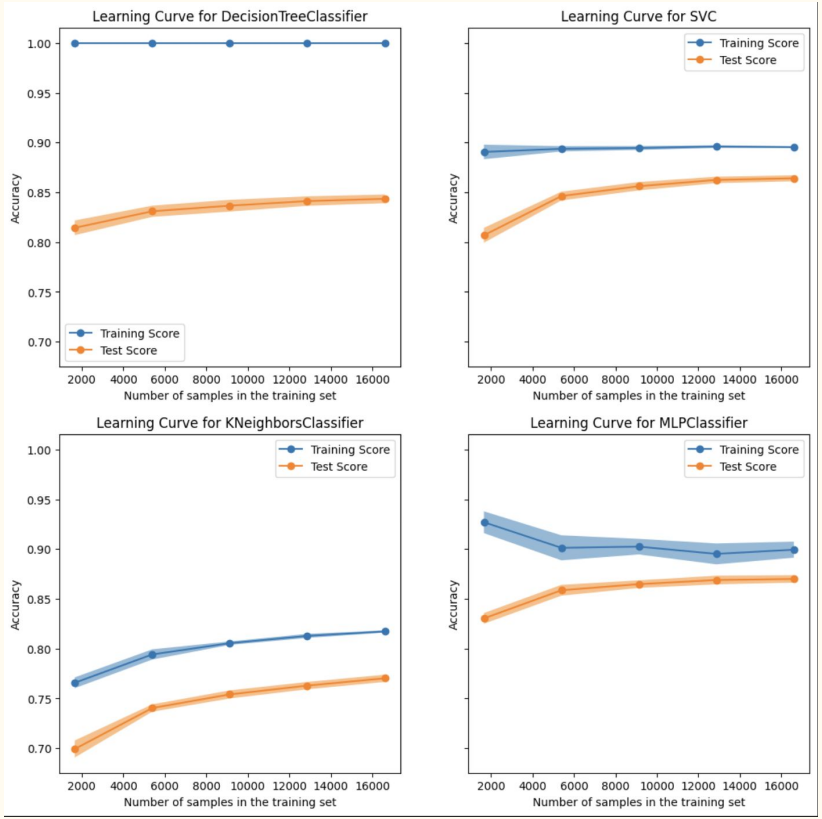
Metrics - Statistics Comparison

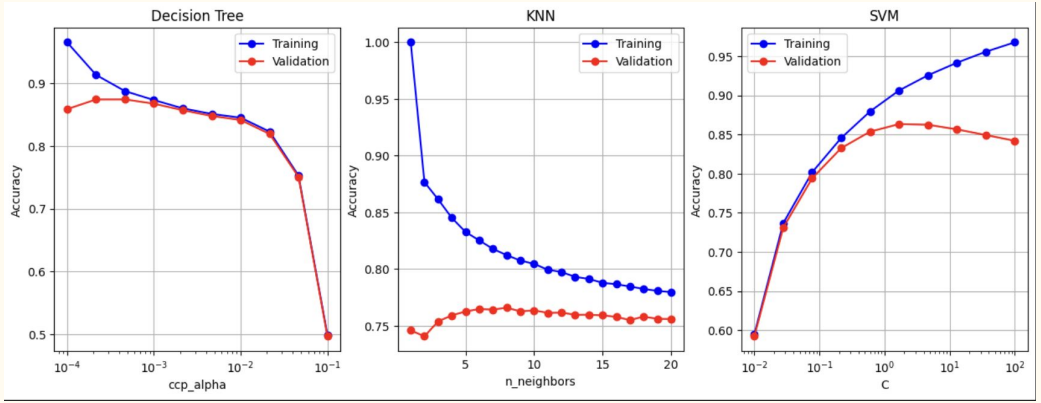
Model	Accuracy	Precision	Recall	F1-Score	Time (s)	Features
Decision Tree (No FS)	84.22	82.66	82.80	82.72	0.12	All
Decision Tree (FS)	84.22	82.66	82.80	82.72	19.34	3
KNN (No FS)	77.36	74.73	74.83	74.72	0.16	All
KNN (FS)	87.24	85.74	85.75	85.72	4.06	5
SVM (No FS)	87.52	86.02	86.16	86.08	9.94	All
Neural Network	87.93	86.46	86.63	86.50	1.16	All

Metrics - Confusion Matrices

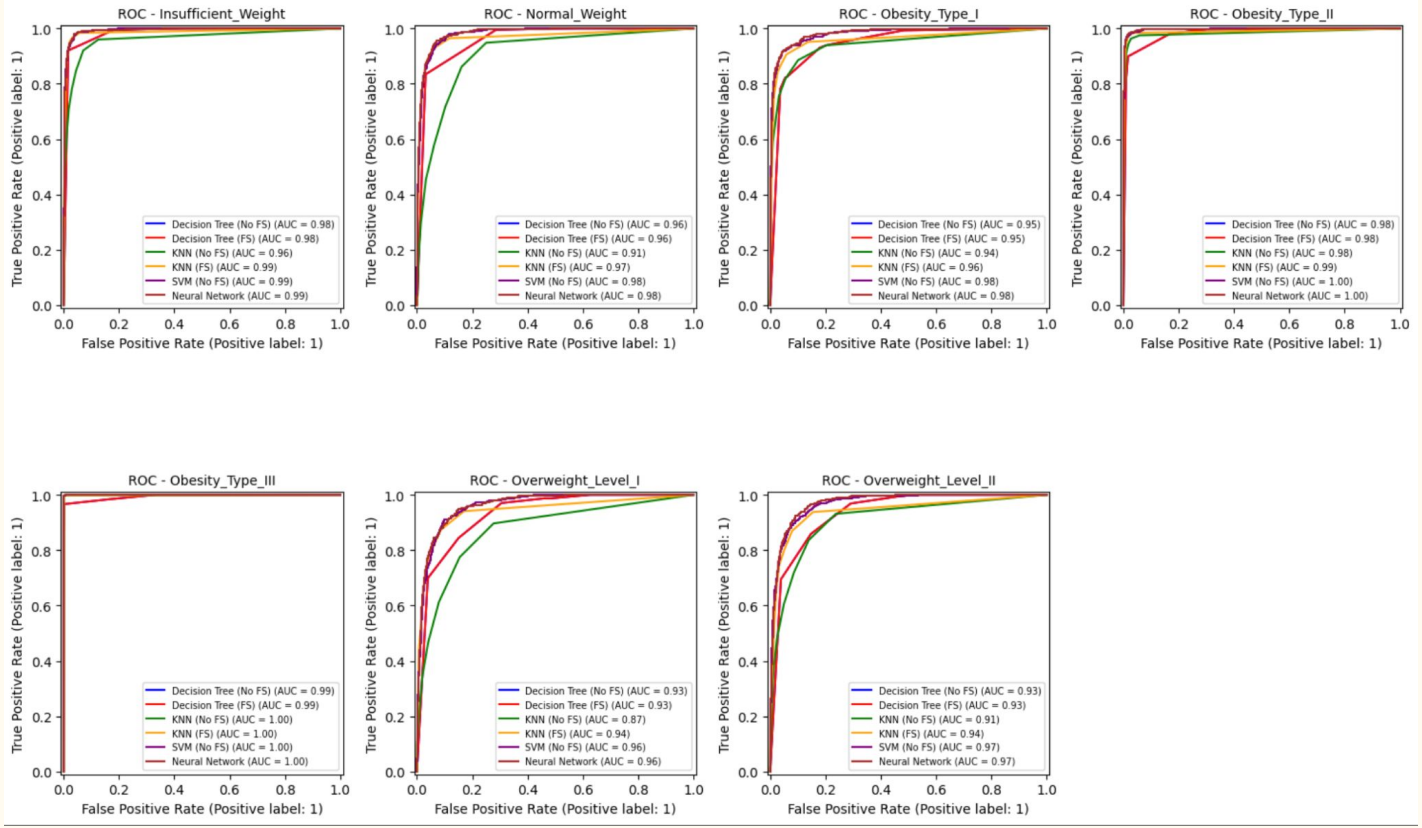


Metrics - Learning / Validation Curves





Metrics - ROC Curves



Metrics - Feature Usage / Overfitting