Multi-Class Prediction of Obesity Risk - IART

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

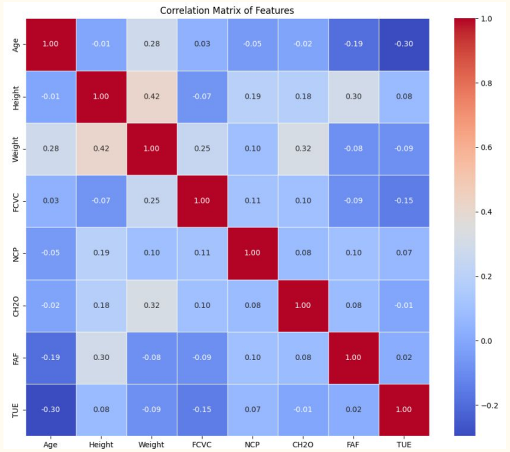
Methodology

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. Interestingly, RFE selected just 3 features: BMI, Weight, and Gender. However, it took longer to run because it tested many feature combinations.

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.

Chosen Parameters:

- N_neighbors = 8 (Controls how many neighbors to consider when making predictions.)

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.

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.

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.

Chosen Parameters:

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

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.

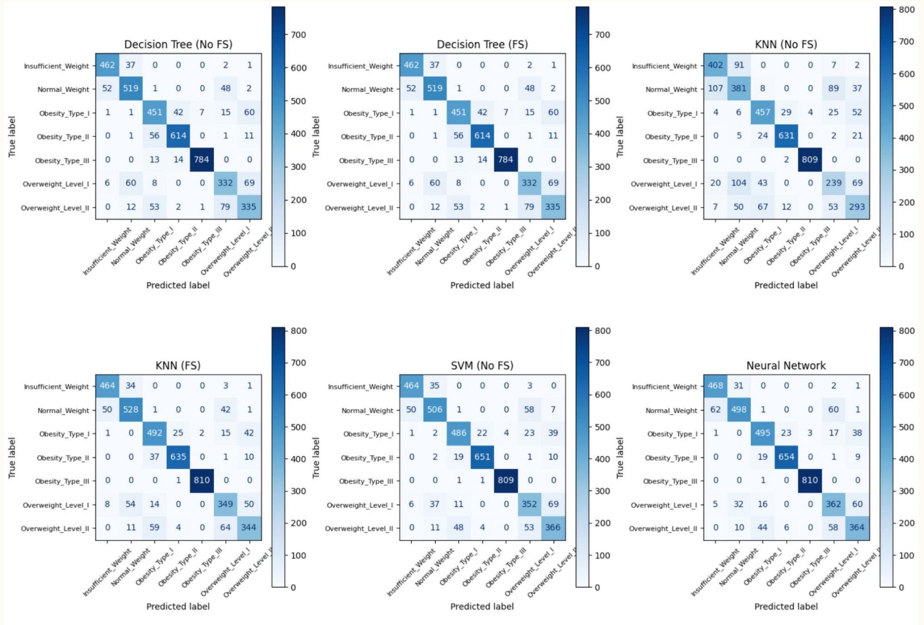
Extra Analysis:

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 - Confusion Matrices

- Neural Networks perform best overall, with minimal confusion and high true positive counts.
- Feature selection greatly enhances KNN but has little impact on Decision Trees.
- Complex models like SVM and Neural Networks handle class distinctions better than simpler models like Decision Trees and KNN.



Decision Tree (No FS)

	Insufficient_Weight	Normal_Weight	Obesity_Type_I	Obesity_Type_II	Obesity_Type_III	Overweight_Level_I	Overweight_Level_II
Insufficient_Weight	462	37	0	0	0	2	1
Normal_Weight	52	519	1	0	0	48	2
Obesity_Type_I	1	1	451	42	7	15	60
Obesity_Type_II	0	1	56	614	0	1	11
Obesity_Type_III	0	0	13	14	784	0	0
Overweight_Level_I	6	60	8	0	0	332	69
Overweight_Level_II	0	12	53	2	1	79	335

Decision Tree (FS)

	Insufficient_Weight	Normal_Weight	Obesity_Type_I	Obesity_Type_II	Obesity_Type_III	Overweight_Level_I	Overweight_Level_II
Insufficient_Weight	462	37	0	0	0	2	1
Normal_Weight	52	519	1	0	0	48	2
Obesity_Type_I	1	1	451	42	7	15	60
Obesity_Type_II	0	1	56	614	0	1	11
Obesity_Type_III	0	0	13	14	784	0	0
Overweight_Level_I	6	60	8	0	0	332	69
Overweight_Level_II	0	12	53	2	1	79	335

KNN (No FS)

	Insufficient_Weight	Normal_Weight	Obesity_Type_I	Obesity_Type_II	Obesity_Type_III	Overweight_Level_I	Overweight_Level_II
Insufficient_Weight	402	91	0	0	0	7	2
Normal_Weight	107	381	8	0	0	89	37
Obesity_Type_I	4	6	457	29	4	25	52
Obesity_Type_II	0	5	24	631	0	2	21
Obesity_Type_III	0	0	0	2	809	0	0
Overweight_Level_I	20	104	43	0	0	239	69
Overweight_Level_II	7	50	67	12	0	53	293

KNN (FS)

	Insufficient_Weight	Normal_Weight	Obesity_Type_I	Obesity_Type_II	Obesity_Type_III	Overweight_Level_I	Overweight_Level_II
Insufficient_Weight	464	34	0	0	0	3	1
Normal_Weight	50	528	1	0	0	42	1
Obesity_Type_I	1	0	492	25	2	15	42
Obesity_Type_II	0	0	37	635	0	1	10
Obesity_Type_III	0	0	0	1	810	0	0
Overweight_Level_I	8	54	14	0	0	349	50
Overweight_Level_II	0	11	59	4	0	64	344

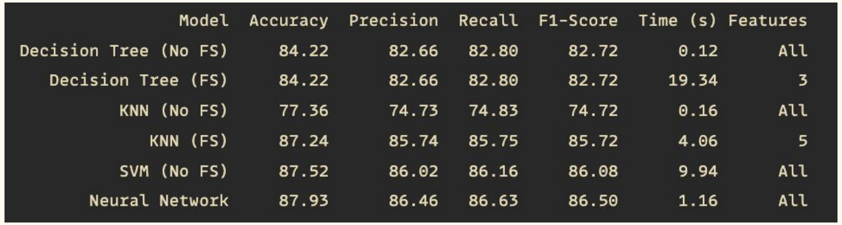
SVM (No FS)

	Insufficient_Weight	Normal_Weight	Obesity_Type_I	Obesity_Type_II	Obesity_Type_III	Overweight_Level_I	Overweight_Level_II
Insufficient_Weight	464	35	0	0	0	3	0
Normal_Weight	50	506	1	0	0	58	7
Obesity_Type_I	1	2	486	22	4	23	39
Obesity_Type_II	0	2	19	651	0	1	10
Obesity_Type_III	0	0	1	1	809	0	0
Overweight_Level_I	6	37	11	0	0	352	69
Overweight_Level_II	0	11	48	4	0	53	366

Neural Network

	Insufficient_Weight	Normal_Weight	Obesity_Type_I	Obesity_Type_II	Obesity_Type_III	Overweight_Level_I	Overweight_Level_II
Insufficient_Weight	468	31	0	0	0	2	1
Normal_Weight	62	498	1	0	0	60	1
Obesity_Type_I	1	0	495	23	3	17	38
Obesity_Type_II	0	0	19	654	0	1	9
Obesity_Type_III	0	0	1	0	810	0	0
Overweight_Level_I	5	32	16	0	0	362	60
Overweight_Level_II	0	10	44	6	0	58	364

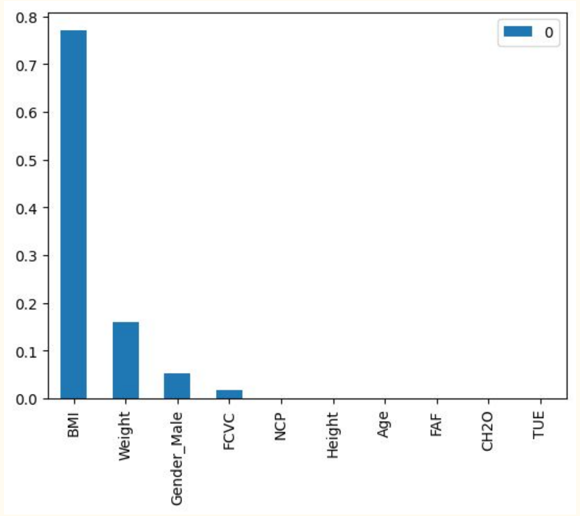
Metrics - Statistics Comparison/Feature Relevance



```
                 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
```

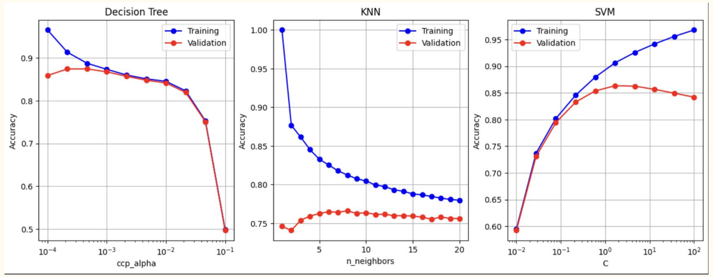
- Neural Networks achieve the highest accuracy (87.93%), slightly outperforming SVM (87.52%) and KNN with feature selection (87.24%).
- Feature selection improves KNN's performance but has no effect on Decision Tree accuracy.
- Decision Tree (No FS) is the fastest model, while Neural Networks balance accuracy and execution time effectively.

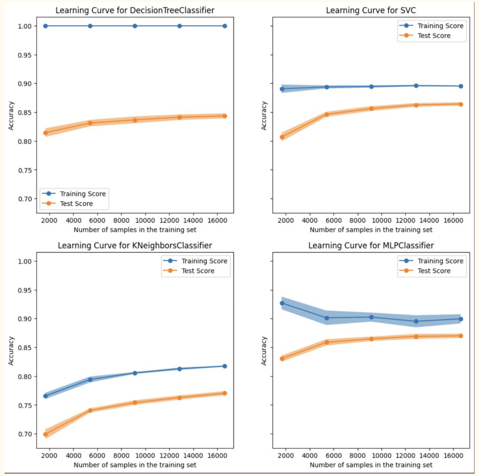
- - The `DecisionTreeClassifier` uses `feature_importances_` to quantify each feature's contribution to predictions.
- - A bar chart of the top 10 features highlights the most influential ones, aiding feature engineering.
- - This analysis is specific to Decision Trees and not applicable to models like KNN, SVM, or Neural Networks.



Metrics - Learning / Validation Curves

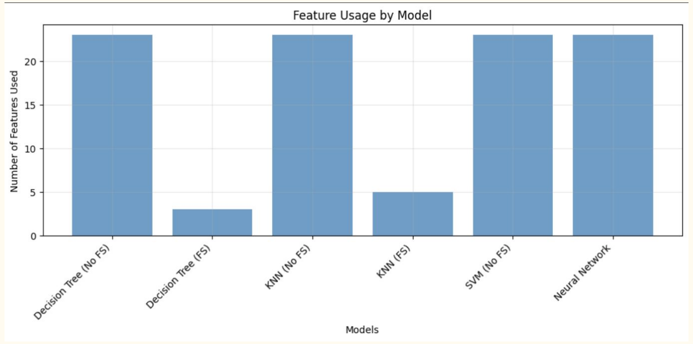
- Decision Tree overfits with perfect training scores but lower test scores.
- KNN underfits, showing low training and test scores.
- SVC and Neural Networks generalize best, with small gaps between training and test scores, benefiting from larger datasets.





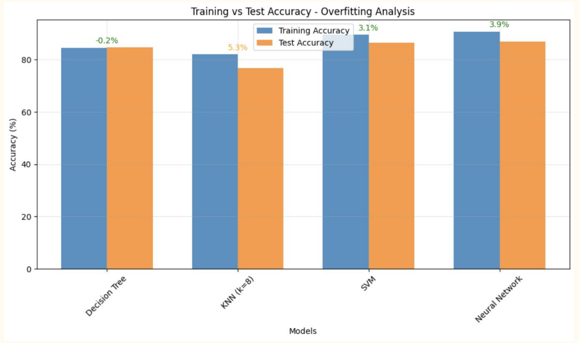
- Decision Tree, KNN, and SVM overfit at extreme parameter values (ccp_alpha, n_neighbors, and C).
- They underfit at the opposite extremes of these parameters.
- Moderate values of ccp_alpha, n_neighbors, and C provide the best balance between training and validation accuracy.

Metrics - Feature Usage / Overfitting

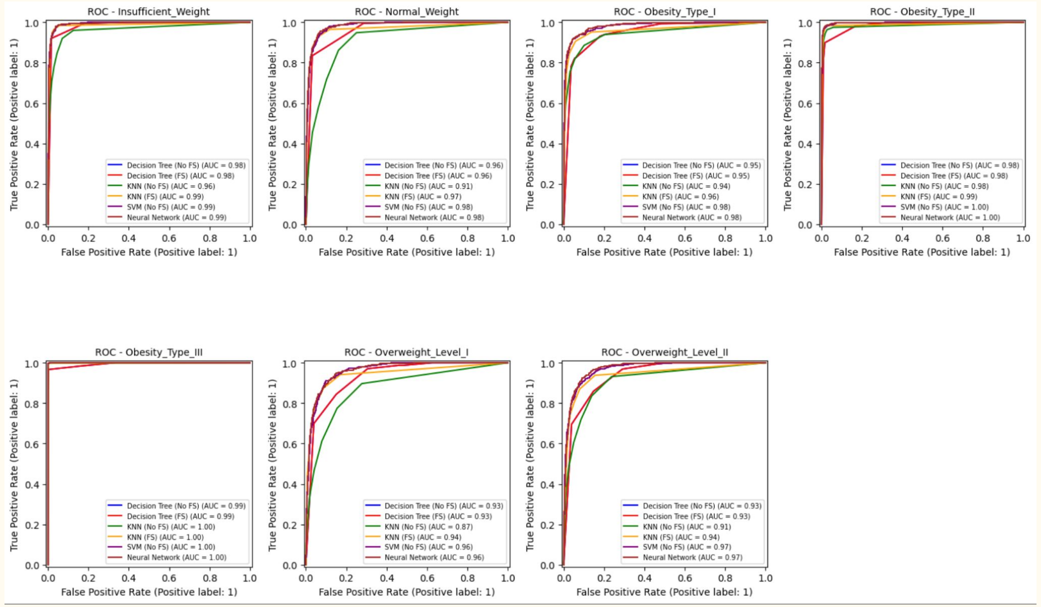


- Feature selection improves KNN's performance and simplifies models like Decision Trees without enhancing their accuracy.
- SVM and Neural Networks perform well without feature selection but may benefit from reduced computational costs.
- Models with feature selection reduce overfitting risks, while those without handle complex patterns effectively but at higher computational costs.

- Decision Tree, SVM, and Neural Networks generalize well, with small gaps between training and test accuracies.
- KNN shows mild overfitting, which feature selection can help mitigate.
- Simpler models like Decision Tree and KNN are more prone to overfitting or underfitting, while complex models like SVM and Neural Networks handle generalization better.



ROC Curves



- Neural Networks achieve the highest AUC scores, followed closely by SVM.
- Feature selection greatly improves KNN but has little effect on Decision Trees.
- SVM and Neural Networks generalize best, though most models struggle with overlapping classes like "Overweight_Level_I" and "Overweight_Level_II."