DATA ANALYSIS
Technical Report

Student Information: Alexios Petrou –alexisptr204@gmail.com

Introduction to the Topic of the Project

In the context of this project, data analysis was conducted on the dataset titled *"Estimation of Obesity Levels Based On Eating Habits and Physical Condition."* The dataset contains information on 2,111 individuals and focuses on characteristics related to eating habits, physical activity, and other lifestyle factors, aiming to estimate each individual's level of obesity. The main objective of this project is to become familiar with techniques of data preprocessing, clustering, classification, and regression, as well as to explore the relationships between features and the Body Mass Index or obesity level.

Contents

1-Data processing

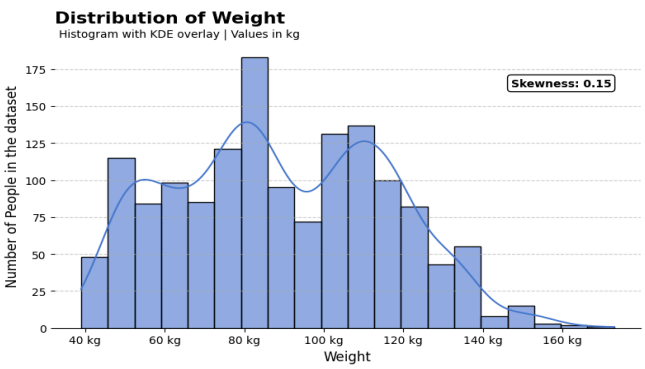
1.1-Data cleaning

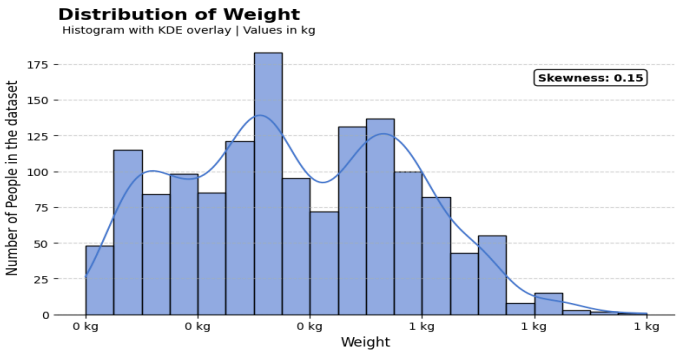
In order to fulfill the objectives of this study, the dataset was partitioned into training, validation, and test subsets, following a ratio of 70%, 10%, and 20%, respectively. Prior to this partitioning, a thorough examination was conducted to identify the presence of duplicate entries, missing values, and outliers. Initially, 24 duplicate records were identified and removed, resulting in a reduction of the dataset size by approximately 1.13% (from 2,111 to 2,087 entries). Subsequently, the dataset was assessed for missing values, of which none were detected. The next step involved the identification and removal of outliers. This was carried out using the Interquartile Range (IQR) method, applying a distinct IQR factor for each variable to account for differences in their distributions. A total of 303 entries—representing 14.5% of the dataset—were excluded as outliers. Following the data cleansing process, the dataset was shuffled to ensure a randomized distribution of samples, and subsequently divided into the aforementioned training, validation, and test subsets. Given that the primary aim of this project is the prediction of an individual's obesity classification, the target variable “NObeyesdad” was isolated from the input features and stored separately in the corresponding target datasets: y_train, y_val, and y_test.

1.2-Data Normalization and Discretization

Following the data cleansing process, normalization and discretization procedures were applied. The dataset includes features with continuous numerical values, such as age, height, weight, FCVC, NCP, TUE, FAF, and CH2O, which were selected for normalization.The Min-Max Scaling method was employed for this purpose, implemented using the MinMaxScaler class from the *scikit-learn* library. This approach was chosen due to the subsequent application of neural networks within the project; Min-Max Scaling ensures that input features fall within a standardized value range, which is beneficial for neural network training and performance.It is important to note that the fit_transform method was applied exclusively to the x_train dataset to compute the scaling parameters, while the transform method was used on x_val and x_test to preserve consistency in the scaling process.

Before Normalization **vs** **After Normalization**



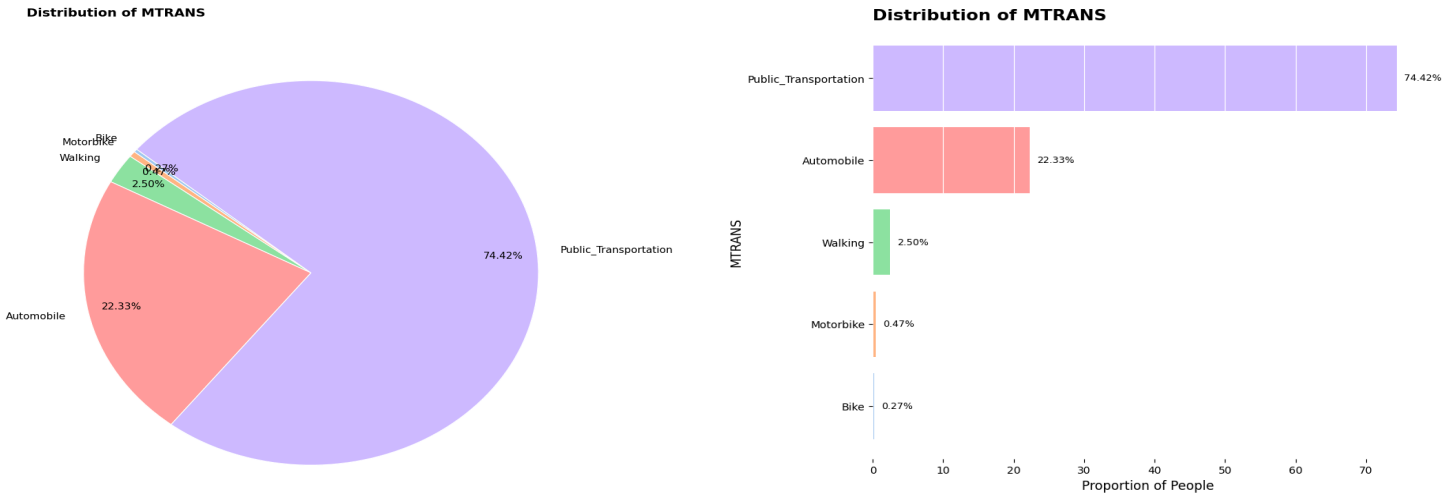


1-Figure thats shows the difference -before and after normalization

1.3-Data Encoding

At this stage of the technical report, encoding is applied to the features that do not contain numerical values. These categorical variables include: *Gender*, *family_history_with_overweight*, *FAVC*, *CAEC*, *SMOKE*, *SCC*, *CALC*, and *MTRANS*.

Before proceeding with the encoding process, an example of a categorical feature from the dataset is



presented below:

2-Figure thats shows the distribution of a categorical feature

There are several available techniques for the categorization (encoding) of data, including *one-hot encoding*, *label encoding*, *ordinal encoding*, among others. For the implementation of this project, a combination of different encoding methods was deemed necessary, as the categorical features in the dataset vary significantly in terms of their nature, role, and the type of information they aim to represent.

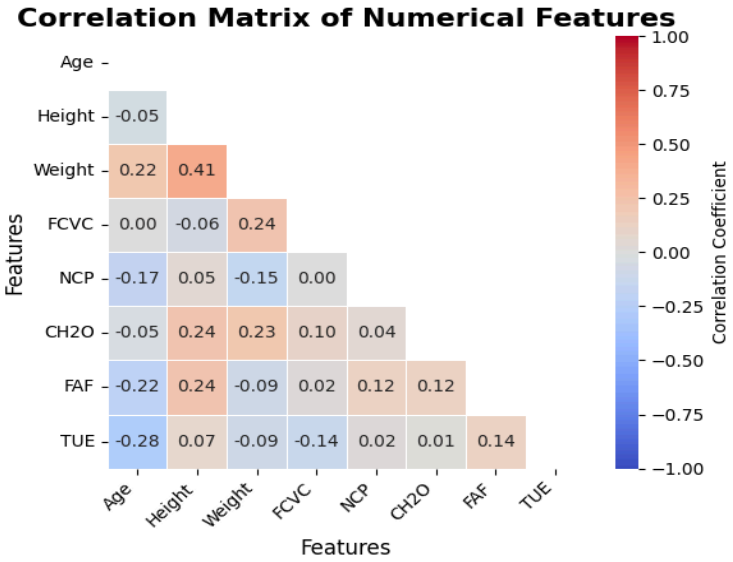
More specifically, the rationale applied in this context is as follows:

- For categorical variables with binary values such as *Yes* or *No*, **Label Encoding** will be applied.
- For categorical variables whose values represent an inherent order or scale, **Ordinal Encoding** will be used.
- For categorical variables with multiple distinct, non-ordinal categories, **One-Hot Encoding** will be employed.

1.4-Data Volume Reduction

The purpose of data reduction is to enable the production of reliable and meaningful results without the necessity of utilizing all available data—particularly when certain features do not significantly contribute to the final outcome. To this end, a correlation analysis was conducted to assess whether the presence of one feature could effectively account for the influence of another. If such redundancy was detected, only one of the two features would be retained. Ultimately, no features were identified as suitable for removal based on this criterion.

Although the dimensionality reduction technique **Principal Component Analysis (PCA)** was implemented experimentally, it was deliberately excluded from the final analysis. This decision was driven by the need to preserve the original dietary features in their raw form in order to maintain the interpretability of the results during the clustering phase. Applying PCA would have transformed the original features into abstract linear combinations, thus impeding meaningful interpretation of the patterns identified through clustering in the context of nutritional behaviors.



Furthermore, during the classification phase, the **Random Forest** algorithm was employed. This algorithm does not require dimensionality reduction and is not negatively affected by correlated or redundant features. On the contrary, it effectively utilizes the full scope of the original variables and also provides a mechanism for evaluating the relative importance of each feature—a capability that would be compromised if transformed features derived from PCA were used instead.

3-Figure thats shows the correlation between numerical features

2-Clustering Analysis

At this stage of the project, clustering of the data will be performed using two algorithms: K-Means and DBSCAN. The implementation begins with the selection of the specific parameters that will be included in the clustering process. According to the assignment requirements, the focus is to be placed exclusively on the diet-related features of individuals, rather than on all available attributes in the dataset. Therefore, features such as age, height, weight, and smoking status will be excluded from clustering. Instead, emphasis will be given to variables such as:

- FAVC (frequent consumption of high-calorie food),
- FCVC (frequency of vegetable consumption),
- NCP (number of main meals per day),
- CAEC (consumption of food between meals), and
- CALC (frequency of alcohol consumption).

Algorithms Used: K-Means vs DBSCAN

K-Means: K-Means is one of the most widely used clustering algorithms due to its speed and simplicity. However, it requires the number of clusters (k) to be defined in advance and may occasionally produce suboptimal results. The first step involves determining the optimal number of clusters using an appropriate method, such as the Elbow Method, visualized in a diagram.
The computational complexity of the K-Means algorithm is:
O(number_of_samples × number_of_clusters × number_of_iterations × number_of_dimensions).

DBSCAN (Density-Based Spatial Clustering of Applications with Noise): DBSCAN is a density-based clustering algorithm that does not require a pre-defined number of clusters. Instead, it identifies clusters as areas of high data point density and labels points in sparse regions as "noise."
DBSCAN relies on two key parameters:

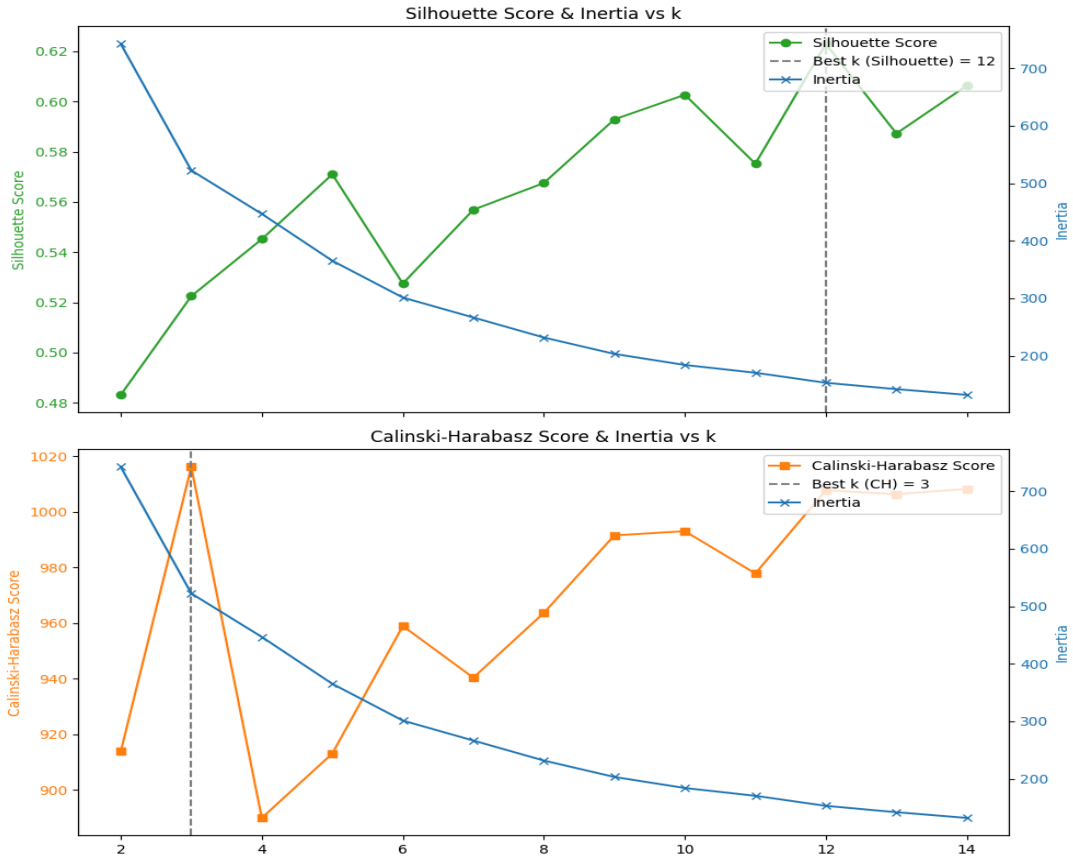
- ε (epsilon), which defines the neighborhood radius, and
- MinPts (minimum number of points), which defines the minimum number of neighboring data points required to form a dense region.

The advantages of DBSCAN include its ability to discover clusters of arbitrary shape and handle noise effectively. However, the quality of the results is highly dependent on the proper selection of its parameters.
The computational complexity of DBSCAN is: O(n × log n), where *n* is the number of samples.

2.1-K-Means

Initially, as previously mentioned, the optimal number of clusters must be determined. To achieve this, the **Elbow method** will be utilized, which provides a plot enabling the identification of the candidate number(s) of clusters that best serve the implementation of K-Means.

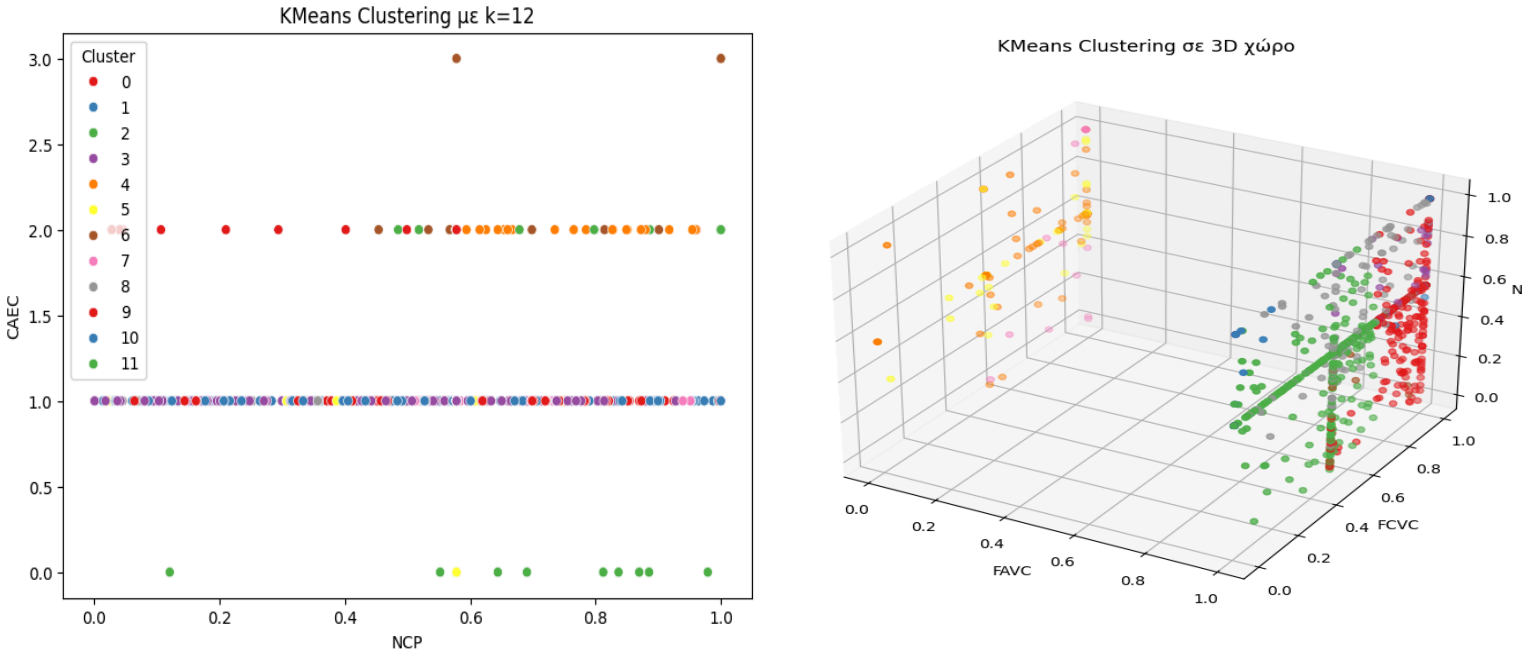
The number of clusters was selected as **k = 12**, as it offers the best balance between compact clustering and adequate separation of the data points. Specifically, at this point, a significant decrease in the inertia value is observed, indicating that data points are closer to their respective cluster centers. Simultaneously, the **silhouette score** reaches a local maximum of **0.6230**, reflecting good cohesion within clusters and clear separation between them. Furthermore, the algorithm's training time was **0.0169 seconds**.



4-Figure thats shows the relation between the number of clusters and silhouette score/Calinski-Harabasz score

Below is the visualization of the **K-Means** clustering:

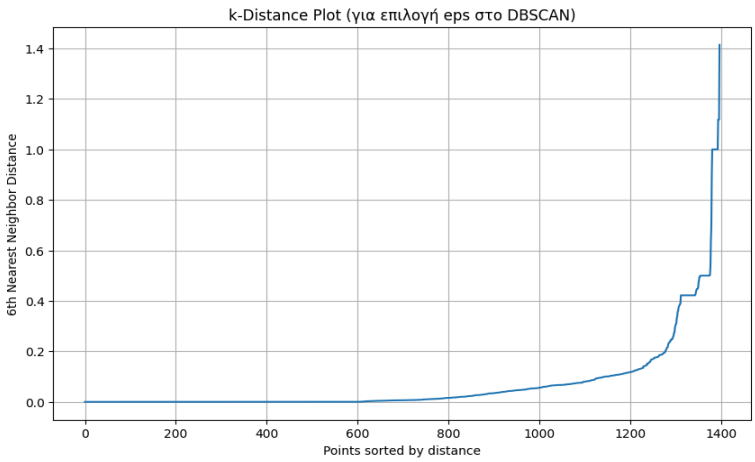
2-dimensional space **3-dimensional space**



5-Figure for visualization of k-means

2.2-DBSCAN

As previously mentioned, the DBSCAN algorithm determines the number of clusters autonomously. However, it requires the specification of two key parameters: **eps (epsilon)** and **min_samples**. The parameter **eps** defines the maximum distance between two samples for them to be considered neighbors, while **min_samples** represents the minimum number of data points required to form a cluster.

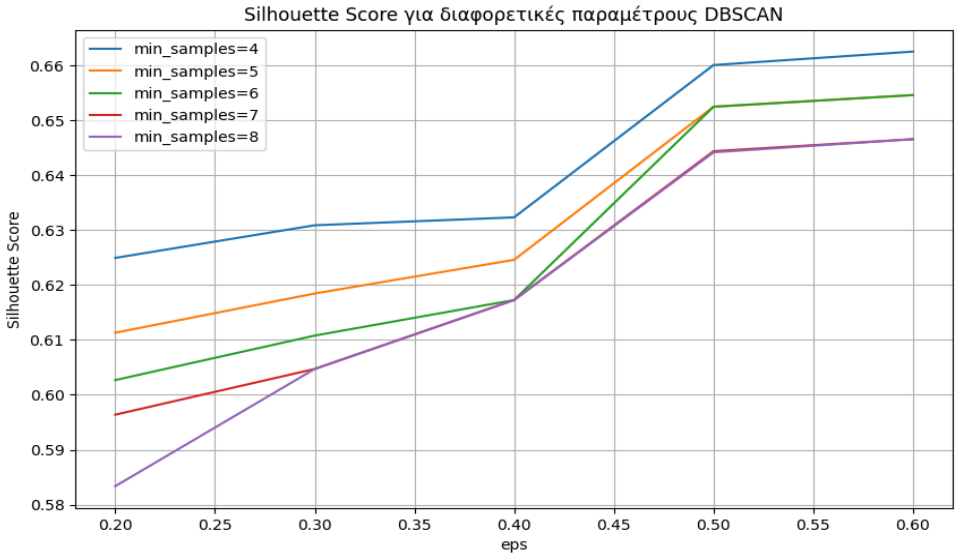


The selection of these parameters followed the following rationale: According to the literature, a common choice for **min_samples** is **n + 1**, where **n** is the number of features. Therefore, in this study, **min_samples** is set to **6**.

Regarding the parameter **eps**, its value is determined based on the following plot:
As observed in the plot on the right, an appropriate value for **eps** lies approximately within the interval **[0.35, 0.5]**.

6-Figure that helps to choose eps value

However, further investigation is required to determine the optimal values for the DBSCAN parameters. The following plot was used to evaluate the silhouette score performance across various parameter values.



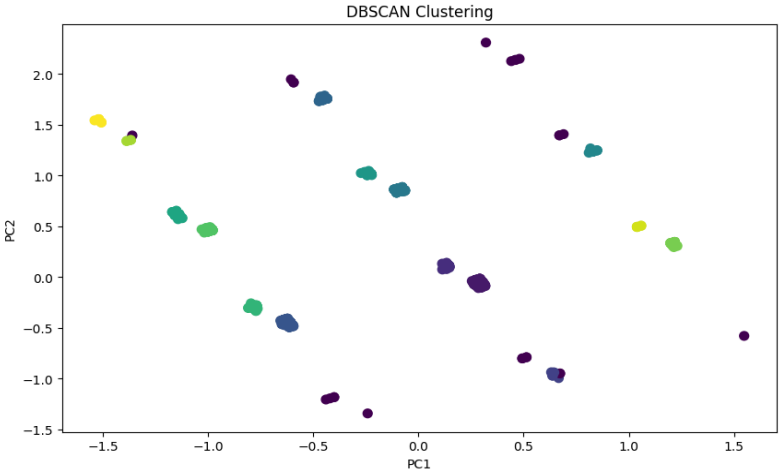
Based on this plot, the parameters were set to **eps = 0.5** and **min_samples = 6**. The choice of **min_samples** was guided by research referenced as the fifth source in the bibliography, following the rule **min_samples = dimensions + 1**.

With these parameters, the DBSCAN algorithm identified **15 clusters** with a silhouette score of **0.652**, and the training time was **0.0715 seconds**.

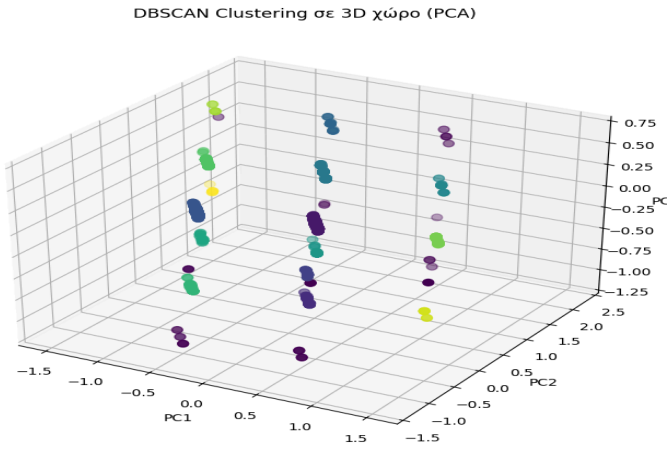
7-Figure that compares the silhouette score for each value of the parameter min_samples

Below is the visualization of the **DBSCAN** clustering:

2-dimensional space



3-dimensional space



8-Figure for visualization of DBSCAN

2.3-Comparison of the Two Clustering Algorithms

The comparison between the two algorithms reveals that, although both achieved satisfactory results, they differ in how they "perceive" the structure of the data. K-means produced a more predictable and symmetrical partitioning, which is advantageous for data with clear, evenly sized clusters, whereas DBSCAN identified more complex and varied clusters by leveraging density-based criteria. Notably, DBSCAN achieved higher cohesion among observations without requiring prior knowledge of the number of clusters, indicating greater adaptability. Ultimately, K-means appears better suited for controlled environments with predictable structures, while DBSCAN excels in uncovering complex patterns within data characterized by vague or heterogeneous distributions.

3-Classification & Regression

In the context of the present analysis, the third step of the experimental procedure focuses on the application of classification and regression techniques aimed at estimating obesity (NObesity) and Body Mass Index (BMI), respectively.

3.1 - Prediction of Obesity Category(Classification)

The initial phase of this process involves selecting the features to be used in classification. The task requested natural features and lifestyle-related features. Therefore, the following were selected: age, height, weight, CALC, FAF, TUE.

For predicting the obesity category (NObesity), multiple classifiers from the Scikit-Learn library were examined, including Naive Bayes, SVM, KNN, Decision Trees, Random Forest, Gradient Boosting, and Neural Networks. However, the primary models chosen were:

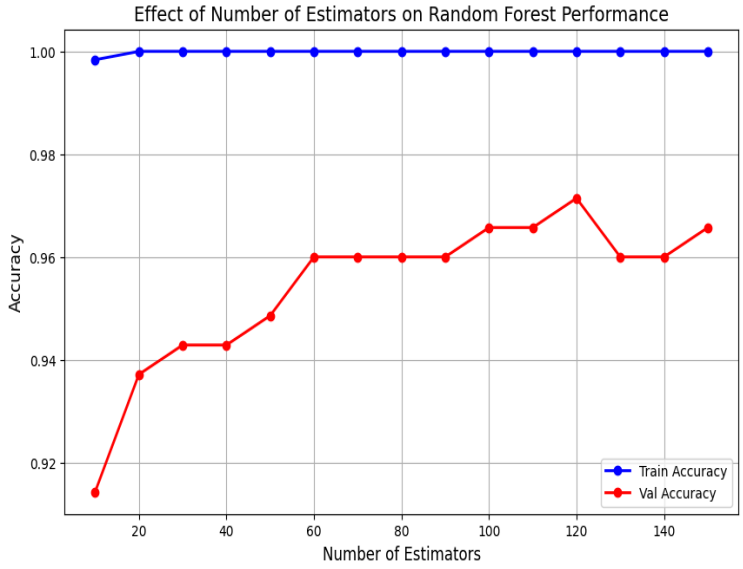
- **Random Forest** – Due to its capability to effectively handle small to medium-sized datasets, deliver high accuracy, and provide interpretability via feature importance.

- **Neural Networks** – Selected for their ability to model complex nonlinear relationships within the data. Additionally, feature normalization (performed in Step 1.b) enhanced their performance, as the transformed data is more suitable for neural network training.

The comparison between these two models will be based on metrics such as accuracy, precision, recall, F1-score, alongside confusion matrices and ROC-AUC curves.

3.1.1-Random Forest

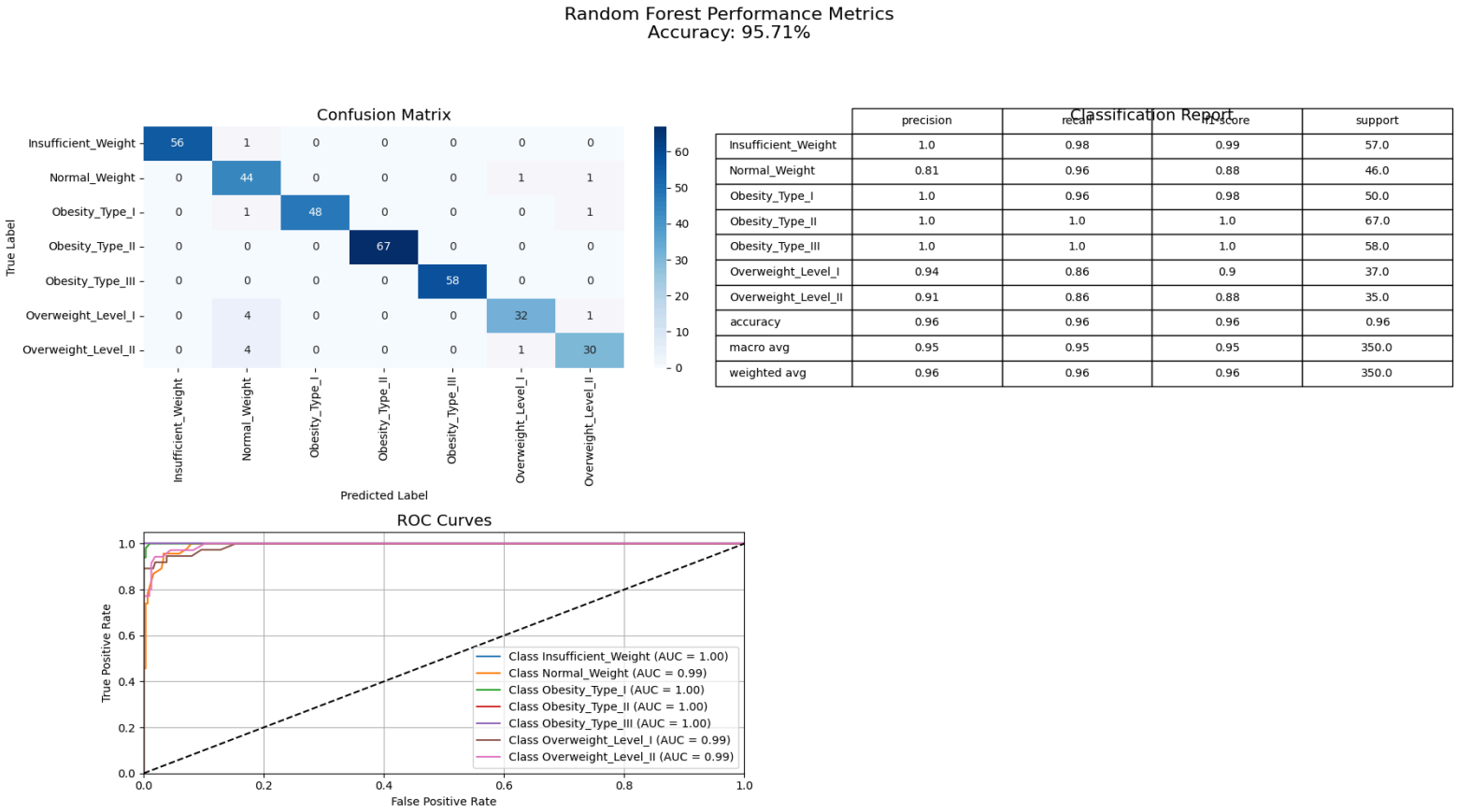
Initially trained on the training dataset with some predefined parameters, it was later deemed necessary to further tune these parameters using the validation dataset. The classifier's key parameters include n_estimators, criterion, min_samples_leaf, min_samples_split, max_depth, and random_state.



9-Figure that shows the accuracy for the training and validation dataset

First, the optimal value for n_estimators (the number of trees in the forest) was investigated using a plot showing classifier accuracy for each candidate value. The graph revealed that validation accuracy improved most significantly between 40 and 60 trees, reaching a global maximum at 120. The value n_estimators=60 was selected, as accuracy stabilized beyond this point and higher values (like 120) increased computational complexity without tangible benefits.

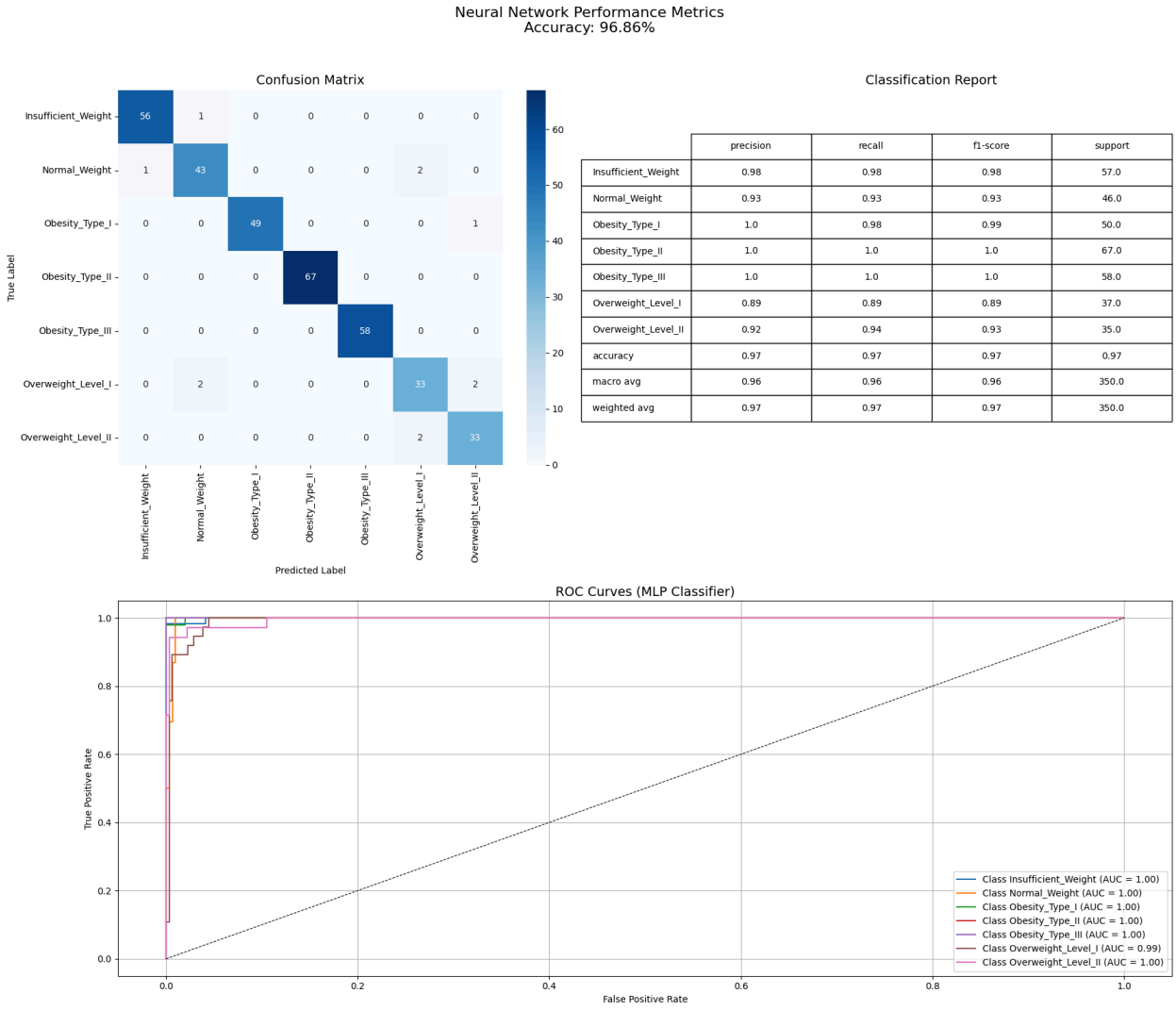
Subsequently, the best values for the remaining hyperparameters (max_depth, min_samples_split, min_samples_leaf) were identified via GridSearchCV with 5-fold cross-validation based on accuracy. The best estimator was evaluated on the validation set, producing a classification report including precision, recall, and F1-score. The optimal hyperparameters were found to be: max_depth=20, min_samples_split=2, and min_samples_leaf=1.



10-Figure that shows the performance of random forest

3.1.2-Neural Network Multi-Layer Perceptron Model

Similar to the Random Forest classifier, the MLP model requires specification of several training parameters. Key parameters included activation function, learning rate, and number of layers. The selected values were: activation function = ReLU, learning rate = constant, solver = Adam, layers = [150, 50]. The final results were as follows:



11-Figure that shows the performance of neural network

3.1.3-Neural Network Multi-Layer Perceptron Model vs Random Forest

---

- **Accuracy:** The neural network achieved slightly higher accuracy (96.86%) compared to Random Forest (95.71%), a difference of 1.15 percentage points. This improvement indicates superior overall classification ability of the MLP, particularly for challenging categories such as Normal_Weight and Overweight_Level_I/II, where Random Forest showed more errors.

- **Precision:** The MLP maintained higher and more consistent precision values (0.89–1.00), whereas Random Forest exhibited greater variability (0.81–1.00).

- **Recall:** Both models demonstrated high recall for Obesity_Type_I–III categories, with MLP outperforming in other categories.

- **F1-Score:** The MLP showed a slightly higher macro-average F1-score (0.96 vs 0.95), indicating a better balance between precision and recall.

ROC-AUC Analysis:

The ROC curves and corresponding AUC values confirm the excellent discriminatory power of both models:

- The MLP achieves an AUC ≈ 1.00 for nearly all categories, except Overweight_Level_I (AUC = 0.99), demonstrating exceptional class separation capability.

- Random Forest also shows strong AUC values (0.99–1.00) but slightly lower in the same categories.

ROC-AUC curves highlight a small but consistent advantage of the MLP, especially in more difficult categories, reflected by visually steeper curves and larger area under the curve (AUC).

Conclusions: The neural network stands out with marginally better performance across key metrics and class discrimination (ROC-AUC). Although both models exceed 95% accuracy, the MLP is recommended when maximum reliability is required, particularly in applications demanding precise prediction ofdifferent obesity and weight levels.

3.2 - Prediction of Body Mass Index (BMI) Value(Regression)

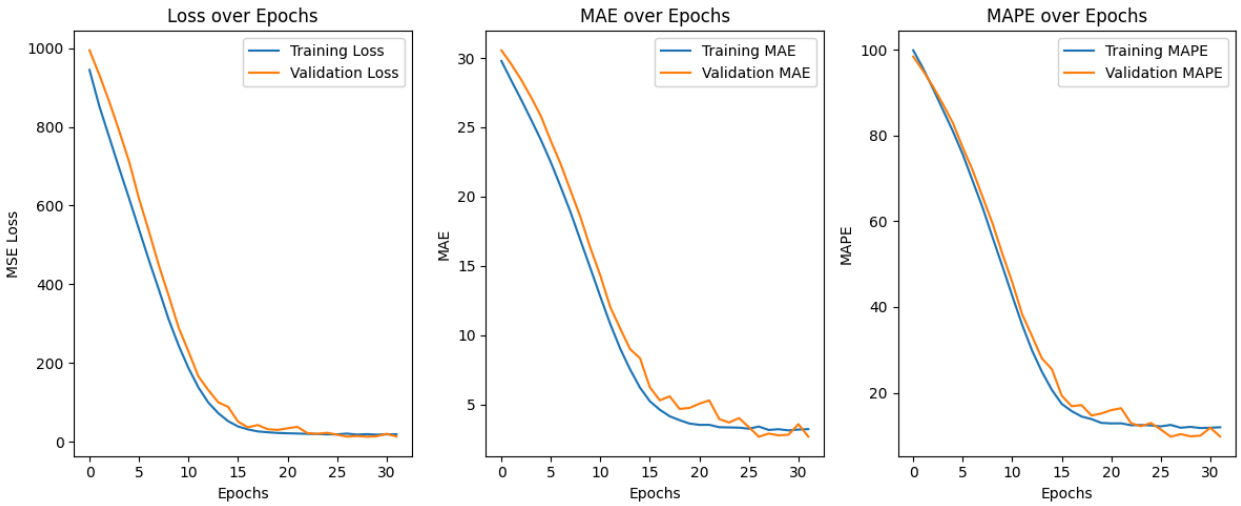
At this stage, the technical report discusses methods used for predicting BMI values. Initially, a BMI column must be created; however, BMI calculation requires weight and height, which have already been normalized. Therefore, the original (pre-normalized) weight and height features are used to generate the BMI target. Consequently, a new training target dataset (train_y) contains the BMI feature, while train_X contains all features from previous datasets.

3.2.1-Feedforward Model

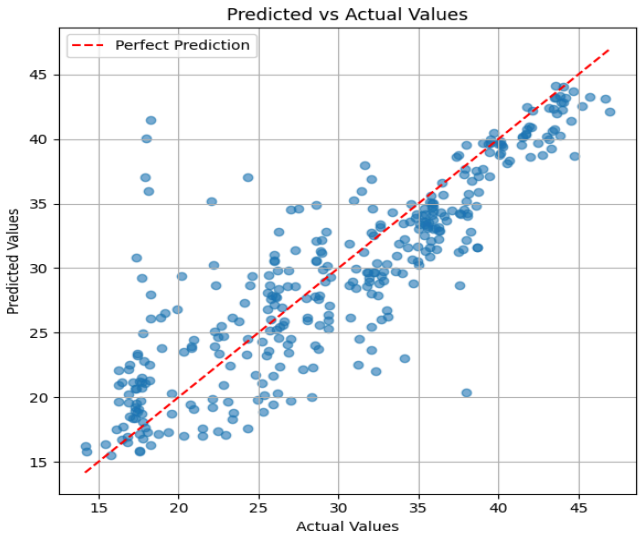
This method involves constructing and training a neural network using the Keras library within TensorFlow. Specifically, a sequential model is created with an input layer matching the number of features in the dataset X_train_3b. The model includes two hidden dense layers with 64 and 32 neurons respectively, both using ReLU activation, and a final output layer with a single neuron for regression (no activation function).

The loss function is mean squared error (MSE), and metrics used are mean absolute error (MAE) and mean absolute percentage error (MAPE). EarlyStopping is employed to prevent overfitting by monitoring validation loss (val_loss) and halting training if no improvement occurs for 3 consecutive epochs, restoring the best weights.

The model is trained over a maximum of 100 epochs with a batch size of 32, using 20% of the data for validation.



12-Figure that shows the performance of the feedforward model



The comparison between predicted and actual values (illustrated in the accompanying plot) shows satisfactory performance. Most points cluster around the perfect prediction line (y = x), indicating small deviations and adequate generalization. Higher BMI values are predicted more accurately, while lower values display greater dispersion, possibly due to insufficient data or higher noise in that range. Overall, the model demonstrates it can effectively learn and reproduce the general data behavior, confirming the success of the training and feature selection process.

13-Figure that shows the accuracy of the prediction

3.2.2-Transfer Learning Model

Transfer Learning is a machine learning technique wherein a model trained on a large dataset for a specific problem is reused or adapted to a new but related task with less data. Practically, this means the pretrained model "transfers" learned knowledge—such as feature representations—to a new task, avoiding training from scratch. This method is particularly useful when resources (data or computational power) are limited. Transfer Learning finds applications in many fields, including image processing, natural language processing, and tabular data through models like TabNet or AutoML frameworks.

Application of Transfer Learning

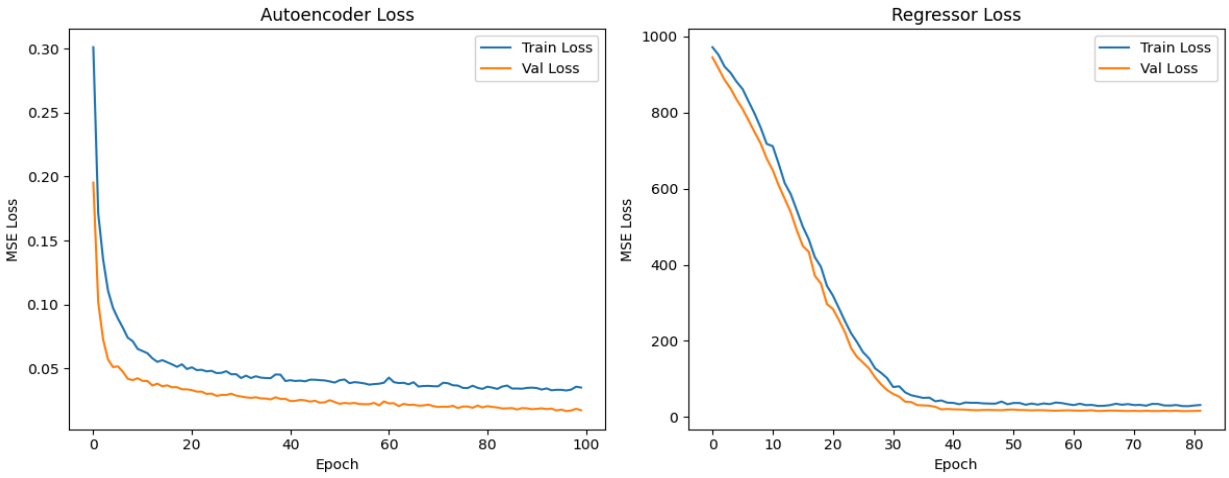
In this approach, an Autoencoder is initially trained in an unsupervised manner to extract latent representations from the input data. The encoder part of the Autoencoder is then employed as a pretrained component of a regression system that learns to predict BMI based on the compressed representations.

Using the encoder as a fixed data transformation mechanism constitutes a form of knowledge transfer from an unsupervised training phase to a supervised regression task. Regularization techniques (dropout, batch normalization) and early stopping were applied to improve generalization.

Model evaluation used metrics such as R^2, RMSE, MAE, and MAPE, with supplementary plots showing loss curves and predicted vs. actual values.

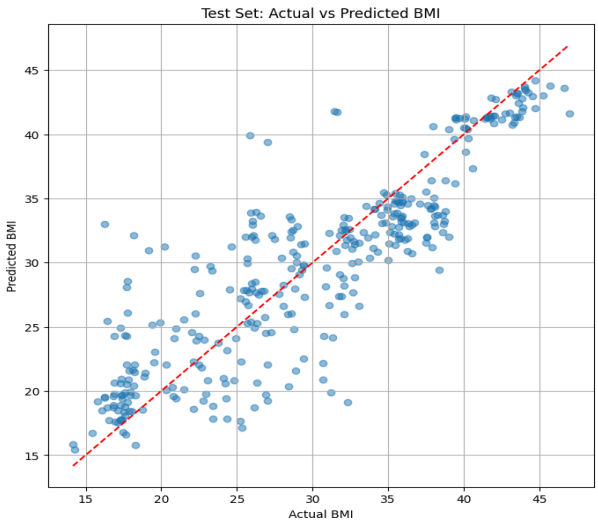
This strategy was deemed appropriate to enhance prediction accuracy and stability by leveraging prior knowledge from data reconstruction in the final regression task.

Screenshots and Commentary from Transfer Learning Application



14-Figure that shows the performance of the transfer learning model

The performance analysis during training demonstrates steady and substantial improvement in both training and validation loss during early epochs. Specifically, from epoch 1 to about epoch 30, a significant decreasing trend in both losses is observed, indicating effective learning of input-target relationships.



After epoch 30, validation loss reduction slows and exhibits minor fluctuations, as expected when approaching optimal generalization. Crucially, no significant overfitting occurs, since validation loss remains low and stable without deterioration.

Moreover, early stopping ensures that the final model is selected based on the best generalization performance (validation loss ≈ 15.4). The large reduction from the initial validation loss (~945) is clear evidence of successful model adaptation to the regression task.

15-Figure that shows the accuracy of the prediction

3.2.3-Comparison of Feedforward and Transfer Learning Regressors

---

Comparing the simple feedforward neural network with the transfer learning approach shows both achieved good performance, with transfer learning slightly outperforming in overall stability and consistency.

Specifically:

- Feedforward model: $R^2 = 0.8132$, RMSE = 3.6612, MAPE = 10.26% on final evaluation
- Transfer learning model: $R^2 \approx 0.79$, RMSE ≈ 3.91, MAPE ≈ 11.04% on validation set, with similarly good test set performance

While the feedforward network showed faster error reduction in early epochs and marginally better final test results, transfer learning provided greater stability during training, especially in data-scarce environments.

Overall, both methods are effective, with transfer learning excelling in generalization and the feedforward network in accuracy.

4-Bibliographic Sources

- Tan, P.-N., Steinbach, M., Karpatne, A., & Kumar, V. (2019). *Introduction to data mining* (2nd ed.). Pearson.
- Arulkumar, A. R. K. (n.d.). *Handling numerical data: Numerical data*. Medium. https://medium.com/@arulprakash/handling-numerical-data-numerical-data-302e6955db2a
- CS221. (n.d.). *How K-means works*. Stanford University. https://cs221.stanford.edu
- scikit-learn. (2024). *DBSCAN — scikit-learn 1.6.1 documentation*. https://scikit-learn.org/stable/modules/clustering.html#dbscan
- GeeksforGeeks. (2023). *DBSCAN clustering in ML – Density-based clustering (How to choose min_samples value)*. https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/
- FluCoMa. (n.d.). *Understanding the parameters of neural networks and what are the differences between each activation function*. https://learn.flucoma.org/
- GeeksforGeeks. (2023). *Random forest algorithm in machine learning*. https://www.geeksforgeeks.org/random-forest-classifier/
- DataScienceDojo. (n.d.). *Titanic dataset*. https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv
- Deep Learning Tutorial 27. (n.d.). *Transfer learning | TensorFlow, Keras & Python*. YouTube. https://www.youtube.com/watch?v=xyz (Insert actual link if different)
- Wikipedia. (n.d.). *Transfer learning*. https://en.wikipedia.org/wiki/Transfer_learning