MACHINE LEARNING Assignment

Based On Eating Habits and Physical Condition

Aabidh
2060001F

Table of Contents

Introduction

Increased risk of obesity-related diseases like diabetes, cancer, and cardiovascular disorders makes obesity a serious worldwide health concern. In 2022 (WHO, 2024), 1 in 8 individuals globally will be affected by it, since its frequency has more than doubled since 1990. The increasing prevalence of obesity is mostly due to modern lifestyles and food habits. Many factors affect a person's weight and health, therefore understanding and forecasting obesity levels requires a multidisciplinary approach. This dataset includes data for the estimation of obesity levels in individuals from the countries of Mexico, Peru, and Colombia, based on their eating habits and physical condition. The data contains 17 attributes and 2111 records.

We took this data set from the Kaggle platform. Attributes in this dataset are Gender,Age,Height,Weight, family history of being overweight, frequency of consuming high-calorie foods, frequency of consuming vegetables, number of main meals, consumption of food in between meals, habit of smoking, daily water consumption, monitoring calorie consume, frequency of physical activity, time spent using electronic devices, consumption of alcohol, mode of transportation, and classification of obesity level. Out of the total variables, 8 attributes are numerical, and 9 are categorical. In this report, we will demonstrate how to perform a classification task and a regression task using a given dataset. These techniques will help us understand the patterns and structures within the data.

Data Exploration & Preprocessing

Ensuring the data has been loaded correctly and comprehends its structure while working with datasets is essential. Pandas' df.head() function is useful for this. This method gives us a short picture of the data by allowing us to preview the first few rows of a data frame.

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS	NObeyesdad
0	Female	21.0	1.62	64.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	0.0	1.0	no	Public_Transportation	Normal_Weight
1	Female	21.0	1.52	56.0	yes	no	3.0	3.0	Sometimes	yes	3.0	yes	3.0	0.0	Sometimes	Public_Transportation	Normal_Weight
2	Male	23.0	1.80	77.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	2.0	1.0	Frequently	Public_Transportation	Normal_Weight
3	Male	27.0	1.80	87.0	no	no	3.0	3.0	Sometimes	no	2.0	no	2.0	0.0	Frequently	Walking	Overweight_Level_I
4	Male	22.0	1.78	89.8	no	no	2.0	1.0	Sometimes	no	2.0	no	0.0	0.0	Sometimes	Public_Transportation	Overweight_Level_II

Figure 1 - First few rows

The next step is to use df.shape, which returns the number of rows and columns, to determine the size of the dataset. We then ensure data integrity, by using df.duplicated().sum() to look for duplicate entries. Checking the data types of each column is done with df.dtypes, which helps us understand the nature of the data. Lastly, we use df.isnull().sum() to evaluate missing values and find any null entries that need handling.

```
df.shape
✓ 0.0s
(2111, 17)

df.duplicated().sum() #duplicate 12
✓ 0.0s
24

RangeIndex: 2111 entries, 0 to 2110
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Gender                          2111 non-null   object
 1   Age                             2111 non-null   float64
 2   Height                          2111 non-null   float64
 3   Weight                          2111 non-null   float64
 4   family_history_with_overweight  2111 non-null   object
 5   FAVC                            2111 non-null   object
 6   FCVC                            2111 non-null   float64
 7   NCP                             2111 non-null   float64
 8   CAEC                            2111 non-null   object
 9   SMOKE                           2111 non-null   object
 10  CH2O                            2111 non-null   float64
 11  SCC                             2111 non-null   object
 12  FAF                             2111 non-null   float64
 13  TUE                             2111 non-null   float64
 14  CALC                            2111 non-null   object
 15  MTRANS                          2111 non-null   object
 16  NObeyesdad                      2111 non-null   object
```

```
Gender                            0.0
Age                               0.0
Height                            0.0
Weight                            0.0
family_history_with_overweight    0.0
FAVC                              0.0
FCVC                              0.0
NCP                               0.0
CAEC                              0.0
SMOKE                             0.0
CH2O                              0.0
SCC                               0.0
FAF                               0.0
TUE                               0.0
CALC                              0.0
MTRANS                            0.0
NObeyesdad                        0.0
dtype: float64
```

Figure 2 - Data understanding

Then we examined the target variable for classification purposes to identify any imbalances in the data distribution. Since the dataset is not imbalanced, no additional remedies were taken.

```
df["NObeyesdad"].value_counts()
✓ 0.0s
Obesity_Type_I         351
Obesity_Type_III       324
Obesity_Type_II        297
Overweight_Level_I     290
Overweight_Level_II    290
Normal_Weight          287
Insufficient_Weight    272
Name: NObeyesdad, dtype: int64
```

Figure 3 - Target variable

In the preprocessing step, we initially addressed the duplicate values within the dataset to ensure data integrity. Using the appropriate methods, we removed the duplicate entries and then reset the index to maintain a consistent Data Frame structure. Since the dataset did not contain any null values, no additional remedies were necessary. Subsequently, we identified specific columns, namely age, and how many main meals you have daily? Do you usually eat vegetables in your meals? were stored in float format. We converted these columns into integers to align them with their appropriate data types. This transformation ensures that age, the number of main meals consumed daily, and the frequency of vegetable consumption during meals are accurately represented as integer values.

we implemented encoding techniques to convert categorical variables into numerical values. Initially, we mapped binary categorical variables using a dictionary where 'no' was mapped to 0 and yes to 1. This map was applied to columns family_history_with_overweight, SMOKE, FAVC, and SCC.

Next, we handled ordinal categorical variables, such as CAEC (frequency of eating between meals) and CALC (alcohol consumption frequency), by assigning them a numerical scale 'Sometimes' was mapped to 0, 'Frequently' to 1, 'Always' to 2, and 'no' to 3.

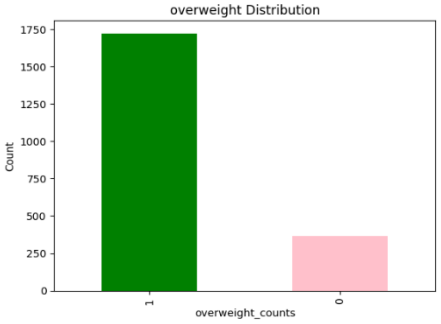
For the MTRANS column (mode of transportation), we categorized the values into three groups: 'Public Transportation', 'Private Motorized Transportation', and 'Non-Motorized Transportation'. This was achieved through a custom function that classified the original values into these groups. However, for further analysis, we employed one-hot encoding for this column. One-hot encoding is particularly suitable for the 'MTRANS' column because it is a nominal variable without a natural order. This method converts the categorical values into binary vectors, ensuring that the model does not infer any ordinal relationship between

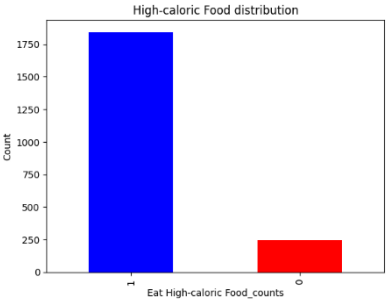
the categories.

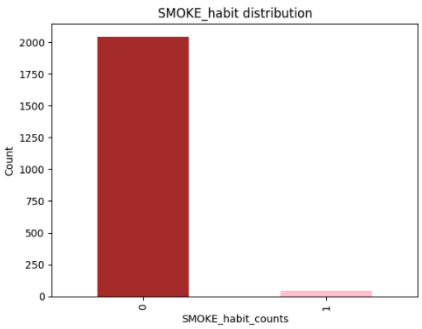
Similarly, the Gender column was encoded by mapping 'Female' to 0 and 'Male' to 1. Finally, we developed mapping for the target variable NObeyesdad (obesity levels) that turned the various levels into numerical values "Obesity_Type_I" to 0, "Obesity_Type_II" to 1, "Obesity_Type_III" to 2, 'Overweight_Level_I' to 3, 'Overweight_Level_II' to 4, 'Normal_Weight' to 5, 'Insufficient_Weight' to 6 All categorical variables are converted into a format that is appropriate for machine-learning models through label encoding.

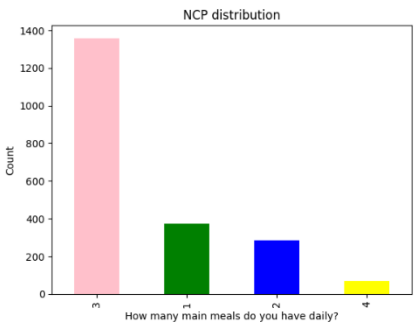
Exploratory data analysis

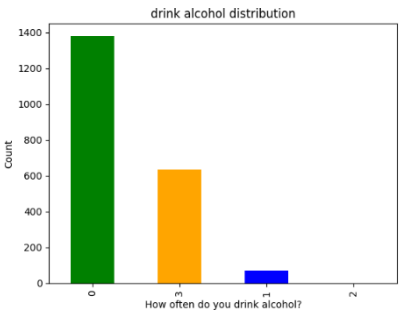
To begin our analysis, we created count plots for each categorical variable in the dataset. Count plots effectively show the frequency of each category within the variables, providing a clear overview of the distribution of data points across different categories. This step is crucial for understanding the underlying structure of the categorical variables before further analysis.

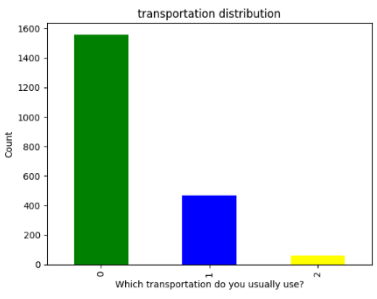












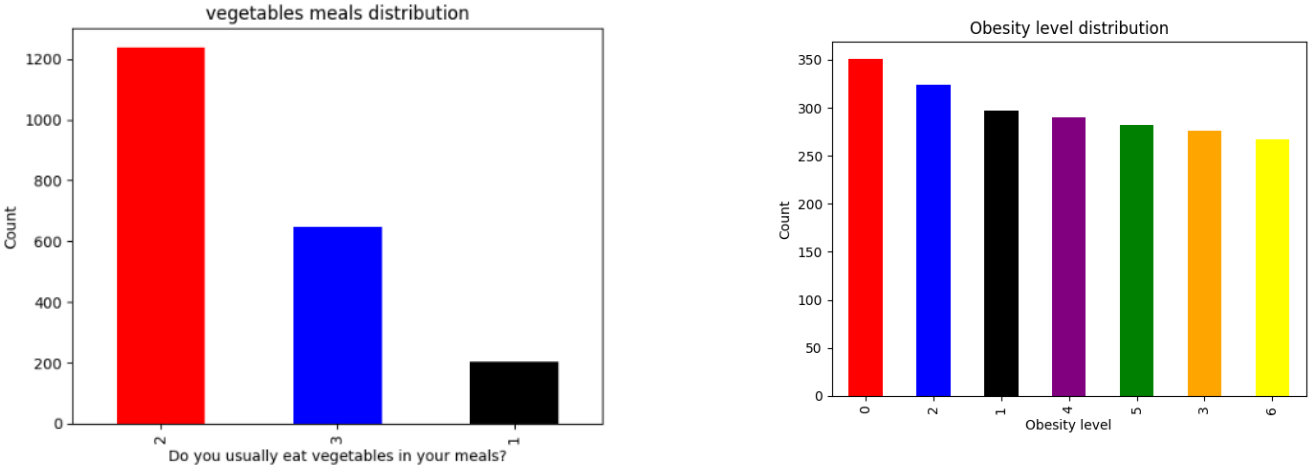


Figure 4 - Categorical counterplot

Then we checked the continuous variable distribution to see whether it was the normal distribution or not. Since its age has a right-skewed distribution we use the Standardization method to scale the variable which will convert it to a mean of 0 and a standard deviation of 1.

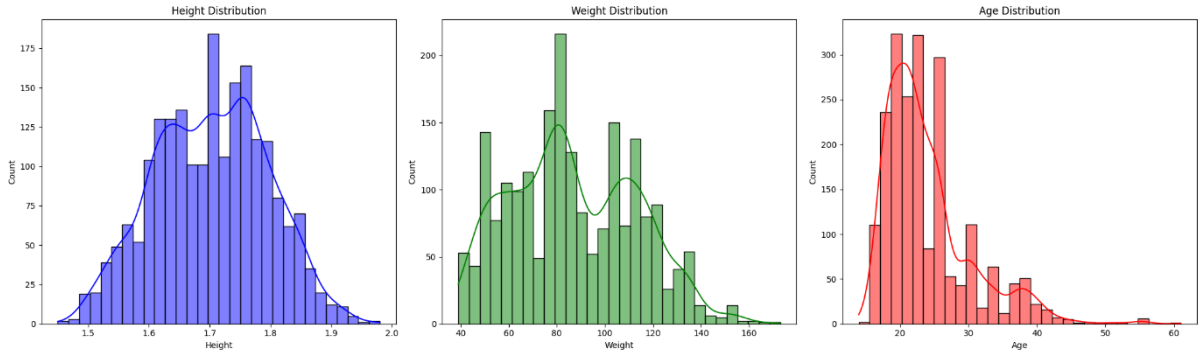


Figure 5 - Distribution plots

Then, we examined the correlation matrix. No further action is required since the analysis revealed no significant multicollinearity among these variables.

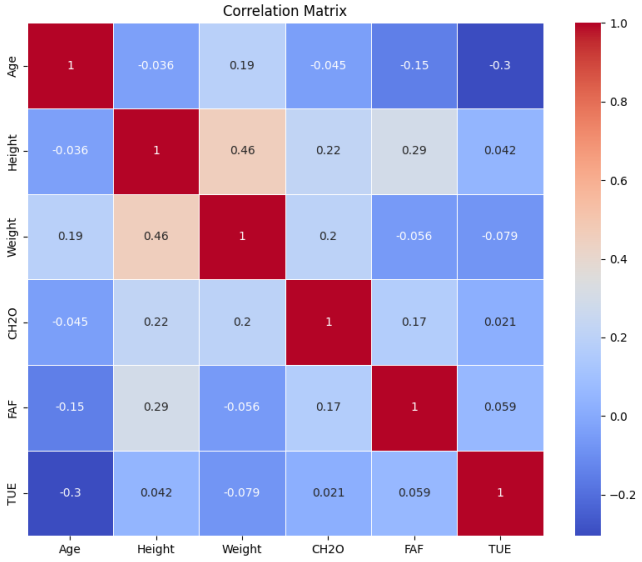


Figure 6 -Correlation matrix

To enhance the performance and interpretability of our predictive model, we conducted a Chi-square test to evaluate the association between categorical features and the target variable. The Chi-square

test provides a statistical measure of how much-observed frequencies deviate from expected frequencies. Higher Chi-square values indicate a stronger association with the target variable. Based on the Chi-square test results, all variables are significant which say there is an association with target variable.

Feature	Chi2	p-value
CALC	558.747807	1.836141e-117
CAEC	516.896206	1.925273e-108
Gender	326.048793	2.129154e-67
FCVC	222.927691	2.471139e-45
SCC	122.141584	5.782949e-24
family_history_with_overweight	108.032828	5.260990e-21
NCP	103.832354	3.975509e-20
SMOKE	31.183130	2.338777e-05
FAVC	26.928662	1.493320e-04

Figure 7 - Chi-Square test table

Then examining the dataset using boxplots, we identified the presence of outliers in the Weight and Age columns. Then we removed those outlier rows from the data set.

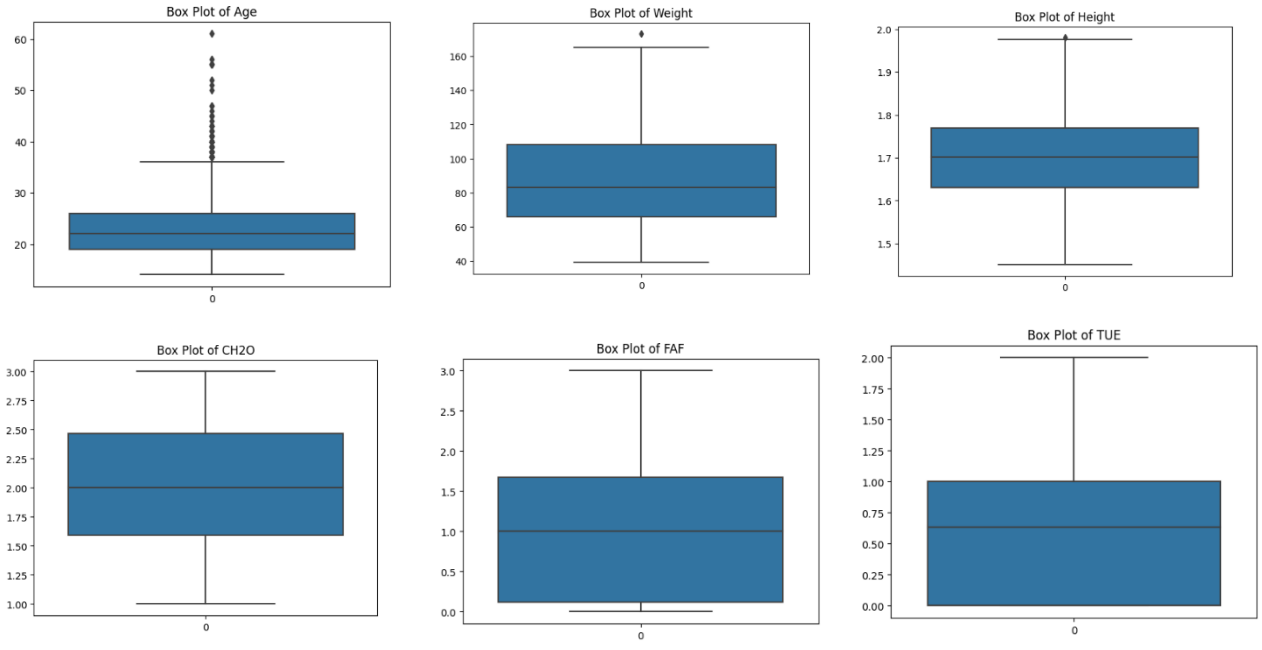


Figure 8 - Box plots

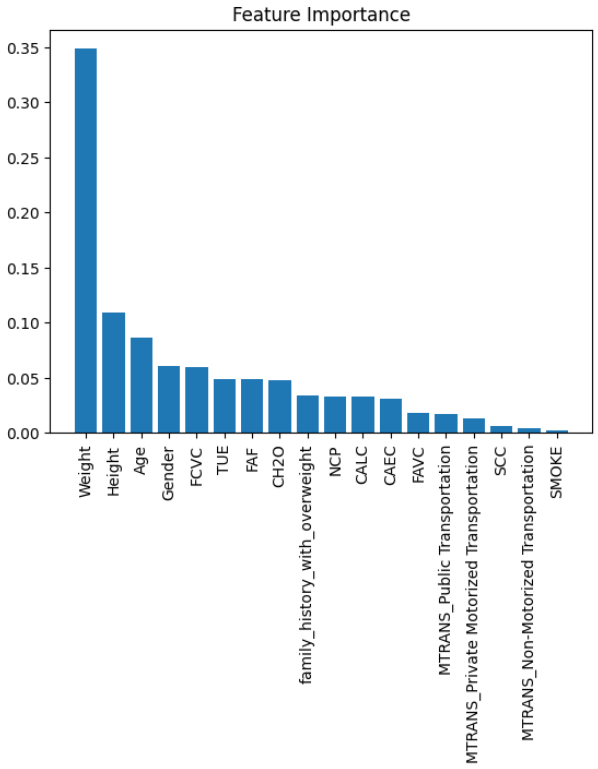


Figure 9 - RF feature selection

In our analysis, we employed the Random Forest algorithm to perform feature selection, leveraging its ability to rank features by importance. The features that contributed to at least 5% of the overall importance were retained for further analysis. This methodology was applied to both classification and Regression tasks.

Model Selection

Classification Task

The dataset comprises various attributes including Gender, Age, Height, Weight, family history of being overweight, frequency of consuming high-calorie foods, frequency of consuming vegetables, number of main meals, consumption of food in between meals, habit of smoking, daily water consumption, monitoring calorie consumption, frequency of physical activity, time spent using electronic devices, consumption of alcohol, mode of transportation, and classification of obesity level. There are 8 numerical attributes and 9 categorical attributes. We used random forest feature selection to reduce dimension which has reduced to 8 variables. The target variable for this classification model is the classification of obesity level. The goal is to build a model that can accurately predict an individual's obesity level based on these attributes, utilizing machine learning techniques to analyze the relationships between the features and the target variable.

We divided our dataset into 80% for training and 20% for testing to evaluate the performance of our models effectively. For classification tasks, we selected four different models SVC, KNN, Random Forest, and Gradient Boosting. Every model has certain advantages that make it useful for various facets of the issue. When class separation is visible SVC performs well in high-dimensional spaces and is robust across a range of data distributions. KNN offers simplicity and versatility, making it a reliable baseline model due to its non-parametric nature and proximity-based classification. Random Forest produces stable forecasts that increase accuracy and decrease overfitting Gradient Boosting performs better on complicated datasets by iteratively creating models to fix prior mistakes and concentrating on hard-to-predict cases.

Regression Task

The goal of this regression study was to predict "Weight" using multiple factors from the dataset, such as "Age," "Gender," "FCVC," "TUE," "FAF," "CH2O," and "NObeyesdad." In the meantime, behavioral variables including "FCVC," "TUE," "FAF," and "CH2O" provide the spotlight on lifestyle choices that could influence weight. By looking at the connections between these variables and weight, we want to create a reliable regression model that can precisely predict weight based on these important variables.

To assess our regression models efficiently, we divided the dataset into two groups 80% for training and 20% for testing. With this method, we can ensure that the models work well when applied to previously unseen data by training them on a sizable piece while keeping a separate set for testing.

We used the Decision Tree Regressor, Random Forest Regressor, and Support Vector Regressor as three distinct models for our regression tasks. Because of its interpretability and simplicity, the Decision Tree Regressor was selected to assist us in comprehending how different features affect the predictions. To use of the Random Forest Regressor's ensemble learning capabilities which increase accuracy and resilience by averaging predictions from several decision trees and minimize overfitting while boosting generalization. Ultimately, the Support Vector Regressor was chosen due to its high-dimensionality efficacy and capacity to handle non-linear interactions through the utilization of various kernel functions. Regression can be approached comprehensively by combining these models, which enables us to identify a variety of patterns and relationships in the data.

Model Training and Evaluation

Classification

To provide a thorough evaluation, we considered several performance criteria when assessing our categorization model. A full assessment of the model's performance is provided by the F1 score, which strikes a compromise between precision and recall. Accuracy offers an overall performance snapshot, while precision demonstrates the correctness of positive predictions. Recall displays the capacity to collect all positives. We also used a confusion matrix, which displays the counts of true positive, true negative, false positive, and false negative predictions and offers a comprehensive analysis of the model's performance. This aids in identifying specific locations where errors may be occurring in the model. All these indicators combined give a comprehensive picture of the advantages and disadvantages of a model.

Training accuracy tables

Before hyperparameter tunning

Performance metrics	SVC	KNN	Random forest	Gradient Boosting
Accuracy	0.95889	0.88755	1	1
Precision	0.96046	0.90968	1	1
Recall	0.95889	0.88755	1	1
F1 Score	0.95899	0.89361	1	1

After hyperparameter tunning

Performance metrics	SVC	KNN	Random forest	Gradient Boosting
Accuracy	0.98823	1	1	1
Precision	0.98839	1	1	1
Recall	0.98823	1	1	1
F1 Score	0.98821	1	1	1

Testing accuracy tables

Before hyperparameter tunning

Performance metrics	SVC	KNN	Random forest	Gradient Boosting
Accuracy	0.90135	0.79643	0.95444	0.96101
Precision	0.91440	0.85788	0.95502	0.96290
Recall	0.90135	0.79643	0.95444	0.96101
F1 Score	0.90130	0.80642	0.95394	0.96112

After hyperparameter tunning

Performance metrics	SVC	KNN	Random forest	Gradient Boosting
Accuracy	0.97885	0.88164	0.95928	0.97585
Precision	0.97848	0.89913	0.95953	0.97626
Recall	0.97885	0.88164	0.95928	0.97585
F1 Score	0.97878	0.88578	0.95859	0.97584

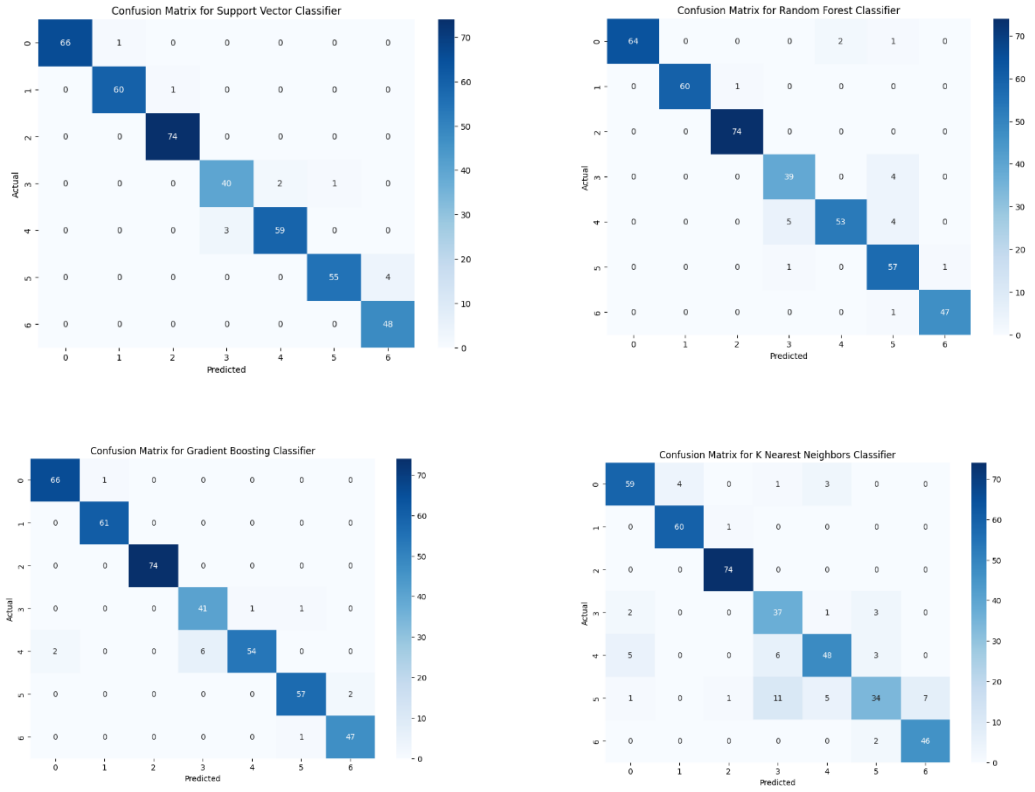


Figure 10 - Confusion matrix

The confusion matrix reveals the classification performance of each model across seven classes. For the Support Vector Classifier (SVC), out of 414 total instances, 402 were correctly classified, indicating a high level of accuracy. The Random Forest Classifier accurately classified 394 instances, showing strong performance as well. The K-Nearest Neighbors (KNN) model classified 358 instances correctly, which is lower compared to the other models, suggesting room for improvement. Finally, the Gradient Boosting Classifier achieved 400 correct classifications, demonstrating its effectiveness in handling the dataset. These results provide a clear comparison of how each model performs in terms of correctly classifying instances across multiple classes.

Comparison and Conclusion

1. **Support Vector Classifier (SVC)**: Shows small improvement in both training and testing metrics after hyperparameter tuning.
2. **K Nearest Neighbors (KNN)**: Improved significantly in training and testing accuracy, achieving good scores after tuning.
3. **Random Forest**: Maintains perfect scores
4. in training metrics both before and after tuning. Testing accuracy improved slightly after tuning, indicating it was already performing well.
5. **Gradient Boosting**: Shows excellent performance both before and after tuning, with slight improvements in testing metrics.

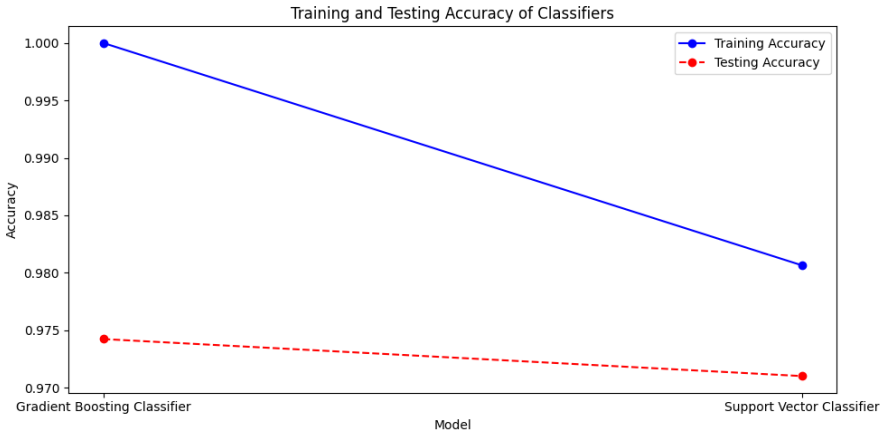


Figure 11 – Classification accuracy plots

Hyperparameter tuning greatly enhances the generalization and accuracy of KNN. After fine-tuning, the Random Forest, SVC, and Gradient Boosting models showed slight performance improvements. While all models exhibit strong performance, SVC, and Gradient Boosting stand out due to their persistent superiority on all criteria.

Regression

Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2 Score are the three main metrics we used to assess our regression models. With the same units as the objective variable, MAE provides an understanding of prediction accuracy by measuring the mean number of errors in forecasts. MSE, on the other hand, emphasizes total prediction variance and gives more weight to larger errors by calculating the average of the squared differences between the actual and anticipated values. The R2 Score provides information about the explanatory power of the model by indicating the percentage of the dependent variable's variance that can be predicted from the independent variables.

Before hyperparameter tunning

Performance metrics	Decision Tree	Random Forest	Support Vector
Mean Absolute Error	2.149	1.686	6.591
Mean Squared Error	17.031	8.632	89.624
R^2 Score	0.976	0.987	0.875

After hyperparameter tunning

Performance metrics	Decision Tree	Random Forest	Support Vector
Mean Absolute Error	2.218	2.059	5.387
Mean Squared Error	12.821	10.750	60.576
R^2 Score	0.982	0.985	0.916

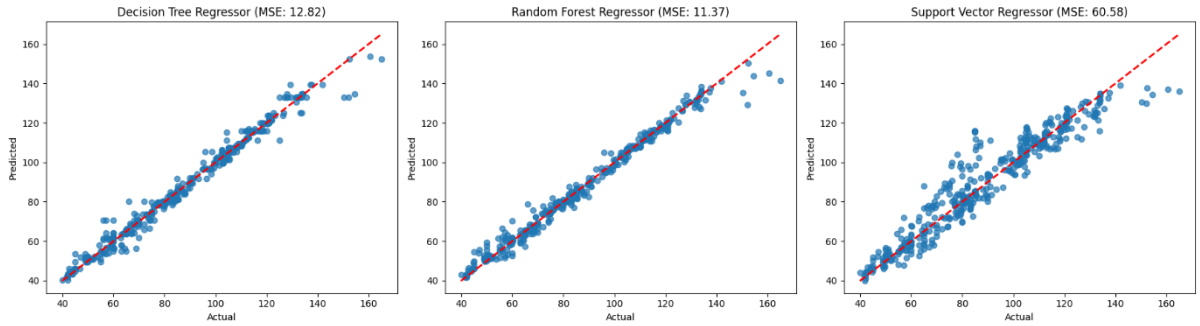


Figure 12 - Regression performance plots

Before hyperparameter adjustment, the Random Forest Regressor showed the best fit and highest R^2 Score (0.987), with the lowest Mean Absolute Error (1.686) and Mean Squared Error (8.632) among the models. Despite having somewhat higher error metrics, the Decision Tree Regressor functioned well all around. Without any adjustment, the Support Vector Regressor was ineffective for this problem, as evidenced by its much greater errors and lower R^2 Score. The Random Forest Regressor continued to perform at its peak after hyperparameter adjustment, with a Mean Absolute Error of 2.059 and a Mean Squared Error of 10.750. While the Support Vector Regressor performed better with smaller errors and a higher R2 Score of 0.916, the Decision Tree Regressor's error metrics climbed marginally. This shows that the Support Vector Regressor's performance was enhanced and made a better model by hyperparameter adjustment.

The plot illustrates the relationship between the actual and predicted weights for the model. Each scatter plot shows the actual weight on the x-axis and the predicted weight on the y-axis. The red dashed line represents the ideal scenario where the predicted weight perfectly matches the actual weight.

If the points lie close to this line, it indicates a good model fit, meaning the model's predictions are accurate. The distance of the points from this line indicates the residuals or the differences between the observed and predicted values. Smaller residuals suggest better model performance.

In this analysis, all three models Decision Tree Regressor, Random Forest Regressor, and Support Vector Regressor show good predictive accuracy. However, the Decision Tree Regressor and Random Forest Regressor outperform the Support Vector Regressor, as their predicted values are closer to the actual values, indicating higher accuracy and better performance in predicting weight.

Challenges Faced and Solutions

1. A major problem was making sure the dataset was free of missing or incorrect values. Model performance can be strongly impacted by noisy or incomplete data. Strict data before treatment procedures were used to address this, including encoding category variables, scandalizing numerical characteristics, and addressing missing values.

2. Overfitting Due to Lack of Scaling - Initially, we encountered overfitting issues with our classification model. This problem was traced back to the feature scaling process. Scaling features can sometimes lead to overfitting if not handled properly. To address this, we applied feature scaling techniques, specifically standardization. Standardization transformed the features to have a mean of 0 and a standard deviation of 1, thereby ensuring that all features contributed equally to the model. This step significantly improved the model's performance and generalization.

3. Handling Outliers - Outliers in the dataset were causing skewed results and affecting the model's accuracy. Outliers can distort statistical analyses and model predictions. We used box plots to visually identify and assess the extent of the outliers. Based on this analysis, we removed the outliers from the dataset, which led to a more robust model with improved performance

4. Hyperparameter Tuning - Selecting optimal hyperparameters is crucial for model performance. Using grid search for hyperparameter tuning was too time-consuming, which was a significant bottleneck. We opted for Randomized Search Cross-Validation (Random Search CV) instead of Grid Search CV. Random Search CV proved to be more efficient as it randomly samples from the parameter space, significantly reducing computation time while still providing effective hyperparameter optimization.

5. Identifying the most relevant features for predicting weight requires careful analysis. Irrelevant or redundant features can introduce noise, reducing model accuracy. To tackle this, the Random Forest algorithm was used for feature selection. Random Forest is an ensemble learning method that combines multiple decision trees to enhance prediction accuracy and reduce overfitting. By analyzing the feature importance provided by the Random Forest model, we were able to identify and retain the most informative features while discarding those that contributed little. This process not only improved the model's performance but also simplified it, making it more interpretable and efficient.

6. Multiple models tuning with huge parameter grids can be a tedious and computationally demanding procedure. To mitigate this, the hyperparameter tuning process was accelerated using distributed computing resources and effective parallel processing.

Conclusion

Several important conclusions were drawn from the evaluation of our regression and classification models. Performance measures for classification, including recall, accuracy, precision, and F1 score, were carefully evaluated both before and after hyperparameter adjustment. Gradient Boosting and Support Vector Classifier (SVC) models showed remarkable gains after adjusting and consistently performed exceptionally well. While K-Nearest Neighbors (KNN) showed significant gains after tuning, but it still trailed behind the best performers, the Random Forest Classifier maintained perfect scores in training and had minor gains in testing accuracy after tuning. The confusion matrix also showed that the most dependable methods for accurately identifying instances across several classes were SVC and Gradient Boosting. Initially, the Random Forest Regressor performed the best in regression tasks, exhibiting a high R2 Score, the lowest Mean Absolute Error (MAE) and Mean Squared Error (MSE). The Random Forest performed better after hyperparameter adjustment, but the Support Vector Regressor improved and outperformed the Random Forest and Decision Tree Regressors in terms of accuracy. All the models were effective, according to the scatter plots, but the Random Forest and Decision Tree regressors produced predictions that were more accurate and had smaller residuals than the Support Vector regressor. The most dependable model overall was the Random Forest model, which performed well in both the classification and regression tasks. SVC and Gradient Boosting also produced good results, particularly in the classification challenge. The performance of the model was greatly enhanced by hyperparameter tuning, highlighting its significance in maximizing predicted accuracy.

References

WHO. (2024, March 1). Retrieved from https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight