Machine Learning Models for Multi-Class Obesity Risk Prediction

Surangi Jayasinghe(RedID: 815904382), Reina Hastings(RedID: 824084541), Trevor Mugoya(RedID:131122027)

***Abstract* - Obesity Risk detection is a critical task for the healthcare industry, especially as the ability to detect obesity has the chance to save lives. In this report, we present a machine learning approach to predict if a patient has obesity by using preprocessed measurements and lifestyle information such as physical activities, eating habits, family history for obesity and many more. We explore various classification techniques and models to effectively identify at risk patients. Our findings portray the significance of machine learning approaches in enhancing diagnostic accuracy and possibly help facilitate efficient classifications in obesity diagnosis. Overall, our experiments show that, with the data available, an obesity risk is well identified with a random forest model that achieved an F1-score of 96%**

# INTRODUCTION

Obesity is a significant public health concern, associated with numerous chronic diseases such as diabetes, cardiovascular diseases, and certain cancers. CDC estimates the prevalence of obesity in people aged 20 and above in the US is 41.9%. Therefore, accurate, reliable and cost-effective methods for preventing its occurrence and progression are required. In this study, we developed an obesity risk prediction system based on machine learning techniques, aiming to achieve personalized comprehensive health management for obesity. Machine learning techniques have emerged as promising tools for enhancing the accuracy of obesity diagnosis, offering potential solutions to challenges i.n traditional diagnostic methods.

We therefore aim to test multiple data mining and machine learning approaches to predict and assess various obesity risk factors.

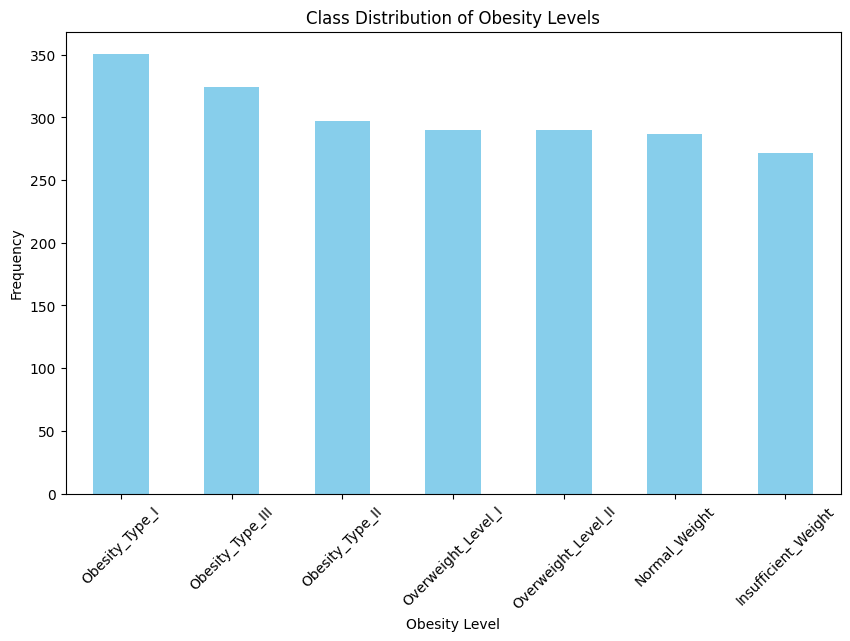
# TASK

The task involves building classification machine learning models to predict if a patient is at risk of developing Obesity using various metrics of health and lifestyle.

# DATA DESCRIPTION

Our dataset is a 16-feature dataset with 2111 instances that include 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 records within the data set are labeled with the class variable NObesity (Obesity Level), that allows classification of the data using the values of Insufficient Weight, Normal Weight, Overweight Level I, Overweight Level II, Obesity Type I, Obesity Type II and Obesity Type III. Among the dataset 77% of the data was generated synthetically using the Weka tool and the SMOTE filter, 23% of the data was collected directly from users through a web platform. To evaluate our models we used accuracy in training and f1 score to determine the effectiveness of the trained model. Additionally, we generated a few visualizations of other metrics such as an AOC, a confusion matrix, and others.

Below is a figure showing the distribution of class labels.



# METHODS

## KNN

### Selecting Optimal Hyperparameters (k and p)

The performance of the KNN model is highly sensitive to two key hyperparameters:

* 𝑘: The number of neighbors to consider.
* 𝑝: The Minkowski distance power parameter (𝑝=1 for Manhattan, 𝑝=2 for Euclidean, etc).

Poor choices for 𝑘 or 𝑝 could lead to overfitting (the model relying too heavily on local patterns, including noise) or underfitting (large 𝑘-values overly smooth the decision boundaries, ignoring useful local variations).

Given the KNN algorithm’s reliance on distance measurements, we initially tested the model using Euclidean Distance, Manhattan Distance, Cosine Similarity, and Hamming Distance. The exploration of different distance metrics and hyperparameters was computationally expensive due to the iterative nature of hyperparameter tuning.

Through experimentation, it was determined that the Minkowski distance was ideal for our dataset because it generalizes both Manhattan distance and Euclidean distance through the parameter 𝑝. It works well with mixed data types (numeric and encoded categorical features) when appropriate scaling and encoding are applied.

To systematically identify the best hyperparameters, a grid search was performed:

* 𝑘-values: Odd values in the range [5, 79] to avoid ties in majority voting.
* 𝑝-values: A range of Minkowski 𝑝-values, including 𝑝=1, 1.5 ,2 ,3.

To ensure robust evaluation, a 5-fold cross-validation strategy was used to measure validation accuracy for each combination of 𝑘 and 𝑝. Models where the gap between training and validation accuracy exceeded 0.05 were filtered out to combat potential overfitting.

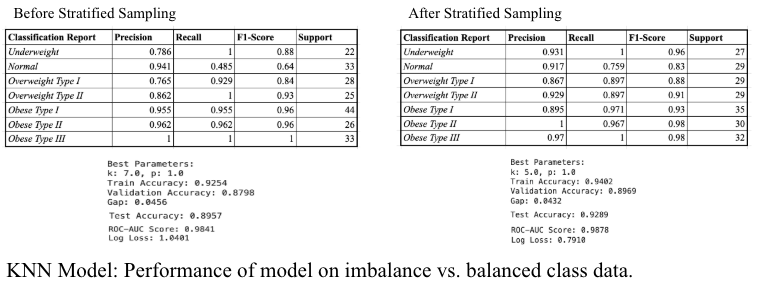
The grid search results indicated that the best-performing hyperparameters were:

𝑘 = 5

𝑝 = 1

### Class Imbalance

While the classes were generally balanced, slight variations in class sizes could bias predictions toward majority classes. The solution for this was stratified splitting. During the train-test-validation split, stratified sampling was used to maintain proportional class distributions across all subsets (training, validation, and test sets). Beyond accuracy, metrics like the ROC-AUC score, log loss, and classification reports were used to assess performance across all classes, ensuring minority classes were well represented.



## Random Forest

Next, we decided to train a Random Forests model on the data, again using Scikit-Learn’s RandomForestClassifier. Random Forest can be scaled to handle large datasets with many classes and features. It's computationally efficient enough for multiclass problems.

## Logistic Regression

We first started by testing the data using a Logistic Regression Module. Logistic Regression offers a simple and interpretable approach to multiclass classification. However, it might not be the most powerful model in terms of accuracy compared to more complex algorithms.

## XGBOOST

In addition, we also applied the XGBoost method to the dataset. XGBoost is a model that builds decision trees in a sequential manner. It is well-suited for multi-class classification tasks and can handle non-linear relationships between features. Our overall metrics for this model was an Accuracy of 96% and F1 score of 95%. After running the XGBoost model we plotted the features based on their importance. Feature importance helps you understand which features are driving your model’s predictions. In a complex model like XGBoost, this provides some transparency into how the model is making decisions.

## DBSCAN

We employed DBSCAN (Density-Based Spatial Clustering of Applications with Noise), an unsupervised learning algorithm that identifies clusters of points in feature space, including non-globular shapes. The key hyperparameters for optimizing clustering are epsilon (eps) and minimum number of points. Epsilon defines the radius of a hypersphere in multidimensional feature space. Core points are defined as points that have at least the minimum number of points within the epsilon radius.

First, we encoded our categorical variables into indicator variables in the obesity dataset with the *pd.get\_dummies* function from the pandas package. We then normalized the features before dimensionality reduction with Principal Components Analysis (PCA). We created a PCA object using the *PCA* function from the *sklearn.decomposition* package in scikit-learn. Using the *pca.fit\_transform* function, we fit the PCA with the normalized data and selected principal components 1 and 2. These components were used to fit our DBSCAN models.

### Clustering Evaluation

Since conventional cross-validation techniques used in supervised learning are not directly applicable to DBSCAN clustering, we used an indirect methodology to validate our results. We built a custom function that iterates through and fits DBSCAN models across all permutations of epsilon and minimum samples hyperparameters. We tested eps values from 0.085 to 0.13 with a step size of 0.005, and minimum samples ranging from 24 to 55 with a step size of 1. We evaluated each DBSCAN model using the silhouette score implementation from the *scikit.learn* metrics package using the *shs* function.

The silhouette score is particularly relevant for evaluating DBSCAN clustering as it measures both the cohesion within clusters and separation between clusters, providing a quantitative metric between -1 and 1, where higher values indicate better-defined clusters with clear separation from neighboring clusters.

# MAJOR RESULTS AND CHALLENGES.

## Results Summary

Our aim for this project was to test multiple machine learning models to find the optimal performing model for obesity risk prediction.

|  |  |  |
| --- | --- | --- |
|  | Accuracy | F1-Score |
| Logistic Regression | 71% | 69% |
| XGBOOST | 96% | 95% |
| Random Forest | 97% | 96% |

One major challenge we faced with our project was the combination of a limited data set and many features. We decided early on that we would not attempt to alter the size of the data set unless our model seemed to suffer as a result. Additionally, because the data set consisted of a large number of features, we wanted see explore the features further by generating data visualizations to help understand the relationships between features related to obesity, health behaviors and family history.

## KNN

The K-Nearest Neighbors (KNN) model was evaluated for its ability to classify obesity status based on a combination of demographic, lifestyle, and behavioral features. The major findings are outlined below:

Using a grid search with cross-validation, the best parameters for the KNN model were identified as 𝑘 = 5 and 𝑝 = 1.

### Evaluation Metrics

#### Accuracy

Accuracy measures the proportion of correctly classified instances among all predictions.

The accuracy was computed using the accuracy\_score function from scikit-learn.

Train Accuracy: 94.02%

Validation Accuracy: 89.69%

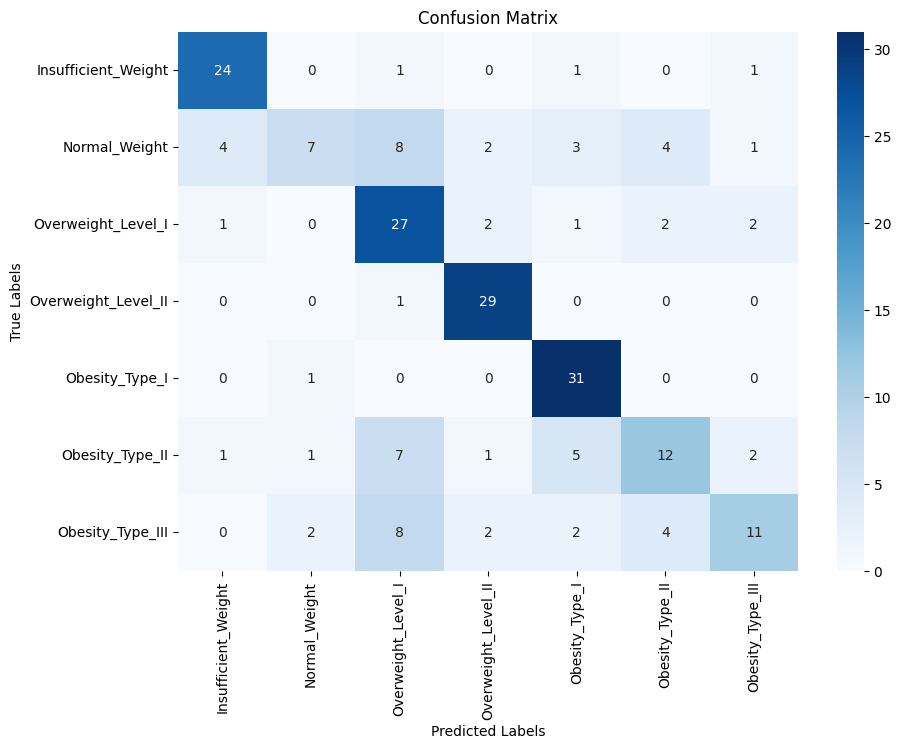
Test Accuracy: 92.89%

The test accuracy demonstrates the model's strong performance on unseen data while maintaining a small training-validation accuracy gap of 0.0432, indicating that the model is not overfitting.

#### Confusion Matrix

A confusion matrix provides a detailed breakdown of the model's predictions across all classes. Each cell in the matrix represents the count of true versus predicted classes. The confusion matrix helps identify which classes are being misclassified and if there is a bias toward certain classes. The matrix was visualized as a heatmap using seaborn.heatmap for interpretability.

"Insufficient Weight" and "Obesity Type III" were predicted with perfect accuracy, as seen by the diagonal dominance of the matrix for these classes. The model had slight misclassifications between adjacent obesity classes, such as "Normal Weight" and "Overweight Level I," which is expected given their close proximity in feature space.



#### Classification Report

The classification report provides key class-level metrics, including Precision, Recall, F1-Score, and Support for each class:

Precision measures how many predicted positive cases were actually positive.

Recall (sensitivity) measures the proportion of actual positive cases correctly identified.

F1-Score is the harmonic mean of Precision and Recall.

Support is the number of true instances for each class.

The classification\_report function generated the results, which were then formatted for readability.

#### ROC-AUC Score

The ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) score measures the model's ability to distinguish between classes. For multi-class problems, a one-vs-rest (OvR) strategy is used, where each class is treated as "positive" while the others are "negative."

The ROC-AUC score evaluates the model’s performance in ranking positive examples higher than negative ones. A higher ROC-AUC indicates better discrimination between classes.

The score was computed using roc\_auc\_score with multi\_class='ovr'.

ROC-AUC Score: 0.9878. The high ROC-AUC score indicates excellent discriminatory power across all classes, suggesting the model effectively distinguishes between obesity categories.

#### Log Loss

Log loss (or cross-entropy loss) measures the model's ability to predict the probability distribution of classes. It penalizes incorrect predictions more heavily when the predicted probability is far from the true class. Log loss is particularly useful for probabilistic models like KNN (when predict\_proba is used). It evaluates not just the correctness of predictions but also the confidence of the model.

The log\_loss function was used to compute the value. We obtained a Log Loss: 0.7910. This demonstrates reliable probability estimates for class predictions.

#### Permutation Feature Importance

Permutation feature importance evaluates the importance of each feature by measuring the decrease in model accuracy when the feature's values are shuffled (permuted). This metric helps identify the most influential features in the model, guiding feature selection and interpretation.

A horizontal bar chart was plotted to show the mean accuracy decrease for each feature.

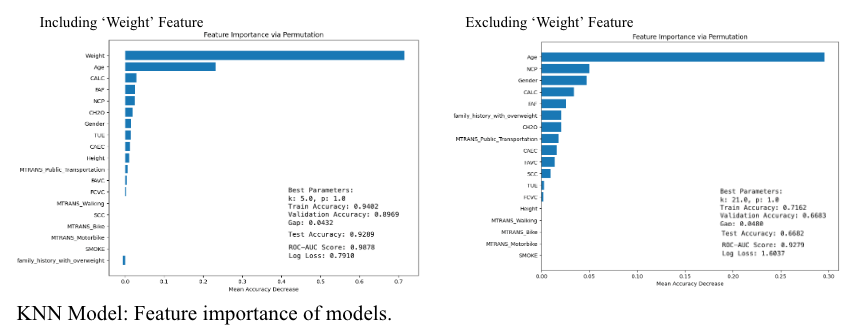
### KNN Model Challenges

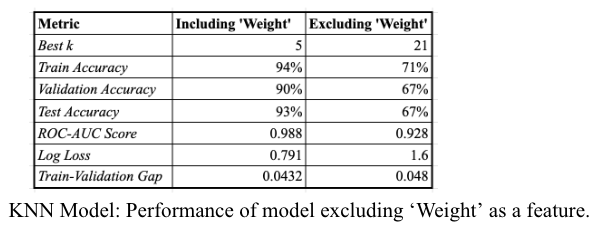
#### Handling Mixed Data Types (Numeric and Categorical Features)

The dataset included both numeric features (e.g., residueCount, resolution) and categorical features (e.g., Gender, family\_history\_with\_overweight, CALC), which cannot directly be used in distance-based algorithms like KNN.

The solution to this issue included label encoding for binary and ordinal features. Binary and ordinal categorical variables were encoded using LabelEncoder to retain their natural order and represent them numerically (e.g., CALC = "Always", "Frequently", "Sometimes"). Features without an inherent order, such as MTRANS (transportation mode), were one-hot encoded to prevent misleading numeric relationships.

1. Removing ‘Weight’ Feature



The Weight feature emerged as the most critical predictor in the initial KNN model, as demonstrated by its high permutation importance. To further investigate the model's reliance on this feature and assess its ability to generalize without it, the model was re-trained and evaluated after excluding Weight. See figure below for the results of this experiment.  Excluding Weight resulted in a significant decline in accuracy and model confidence. Age and other lifestyle features (e.g., caloric intake, meal frequency) gained importance but could not fully compensate for the absence of Weight.

Conclusion & Future Directions

Feature Selection to Improve Model Performance & Elucidate Important Features

While the current model achieved strong results, there is an opportunity to enhance performance further by removing features that appear unimportant or contribute noise to the algorithm. I would like to explore the feature permutation importance. I identified features with minimal or negligible contributions to the model’s accuracy. For example, features such as transportation modes (MTRANS) and lifestyle variables like SMOKE exhibited low importance in the KNN model.

Future work could involve systematically removing such features to reduce dimensionality and computational cost without sacrificing performance. Features with very low or negative importance during permutation analysis may introduce unnecessary noise, distorting distance calculations in the KNN algorithm. The model could focus on the most predictive variables, such as Weight, Age, and CALC, which were identified as key drivers of performance.

A future direction could include an iterative feature selection process, such as recomputing permutation importance after removing unimportant features.Then, retraining and evaluating the model on the reduced feature set. Automated methods, such as Recursive Feature Elimination (RFE) or other wrapper-based approaches, could be explored to identify the optimal subset of features.

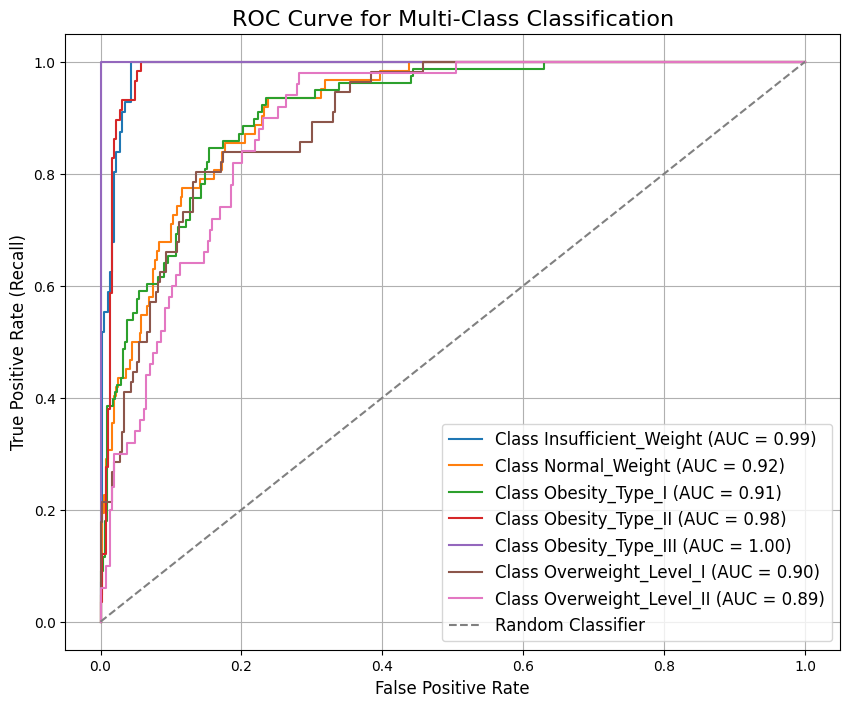
Beyond improving the technical efficiency of the model, identifying and prioritizing these especially important features holds practical value in real-world obesity prevention and intervention efforts. For example, we already use weight as a dominant predictor. Early monitoring and targeted weight management interventions could help identify individuals at risk of developing obesity. Age, another important feature, can lead us to focus on age-specific health programs to promote healthy behaviors during critical life stages. The CALC feature highlights the importance of managing alcohol consumption in order to prevent obesity.

By focusing on the most predictive variables, healthcare practitioners can design targeted prevention strategies and personalized interventions to address obesity more effectively.

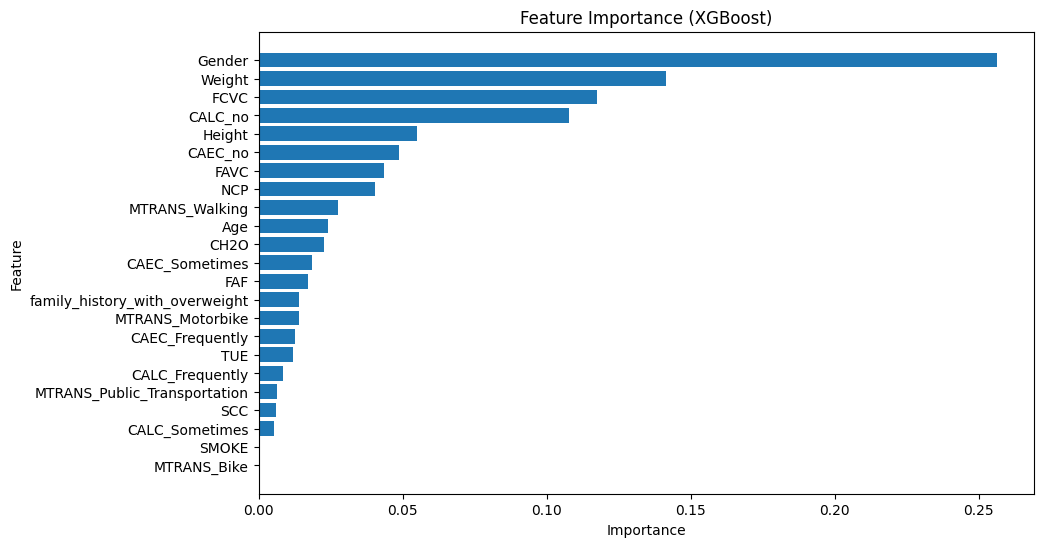
## Random Forest

The model had a high performance overall with an accuracy of ~92.6% and F1 Macro of ~0.93, which is quite strong for a multiclass classification task. Based on the classification report, Obesity\_Type\_III had the highest performance with 100% precision, recall, and F1-score, indicating that this class was very easy to predict. Normal Weight has a lower precision (0.78) but relatively high recall (0.89), indicating the model is better at detecting it when it is truly present, but it also misclassifies some instances as "Normal Weight". Obesity Type II and Obesity Type I are also performing well, with high F1 scores. Overweight Level I and II are also predicted well, but with some minor differences in precision and recall. To ensure quality of performance we also tested the model using cross validation. The results of the model's cross-validation provide a comprehensive overview of its performance. The cross-validated accuracy of the model is reported as 0**.**9295 ± 0.1066. This metric indicates that, on average, the model correctly classifies approximately 92.95% of the instances in your dataset. A high F1 Macro Score of 0.9296 indicates that the model not only performs well in terms of accuracy but also maintained a good balance between precision and recall across all classes. Again, cross validation provided comparable results to the original accuracy and F1 macro scores.

## Logistic Regression

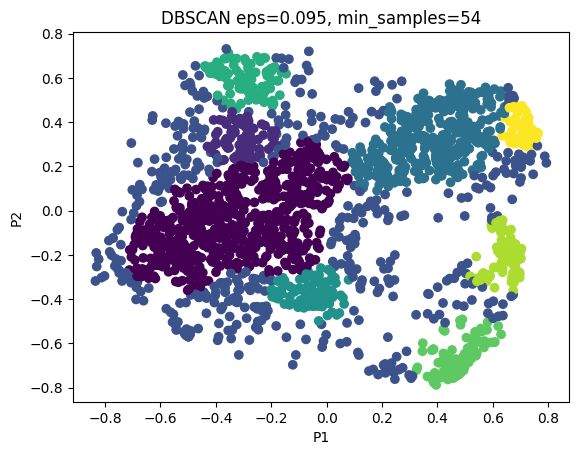
Overall, the model achieved an accuracy of 71%, which indicates room for improvement in performance. This could be due to having a large multi class data set, some classes may be slightly unbalanced to each other, where one class is more dominant than the others. However, after graphing the ROC curve for the model, shown below, the results seem promising.   
  
  
The AUC or Area Under the Curve measures the models ability to distinguish between classes, across all possible classification thresholds. As you can see in the figure the AUC values all within the range of 90s indicates that the model is very good at distinguishing between classes. Specifically class Obesity III had a great true positive rate. Overall, the model achieves a strong true positive rate without much increase in the false positive rate.

*D. XGBOOST*  
In addition, we also applied the XGBoost method to the dataset. XGBoost is a model that builds decision trees in a sequential manner. It is well-suited for multi-class classification tasks and can handle non-linear relationships between features. Our overall metrics for this model was an Accuracy of 96% and F1 score of 95%. After running the XGBoost model we plotted the features based on their importance. Feature importance helps you understand which features are driving your model’s predictions. In a complex model like XGBoost, this provides some transparency into how the model is making decisions. As you can see in the figure found below the feature gender, weight, and FCVC which represented if the patients eat vegetables or not in their diet, plays a key role in feature selection. Following that was how often the patient had alcoholic drinks and height were features ranked highest in importance.

  
Some of the lowest ranked features of importance was SCC which references if the patient monitors their calorie intake or not and lastly modes of transportation. This overall provides a better understanding of the features and how much they can affect the models performance. Based on our preliminary findings (see code) and the findings after applying cross validation; we discovered that next steps in the future would be to implement more fine-tuning for XGBoost hyperparameters, which we can optimize model performance and better align the results.

## DBSCAN

After evaluating 278 hyperparameter permutations, we found that an epsilon of 0.095 and minimum samples of 54 produced the highest silhouette score of 0.726. This resulted in 8 clusters with 468 noise points. This finding is particularly interesting given that we have 7 ground truth labels.



# CONCLUSIONS.

## In conclusion, our study emphasizes the importance of using different machine learning methods to tackle the challenge of identifying the risk of developing obesity. Based on our testing, random forest achieved the highest accuracy.

# FUTURE DIRECTIONS.

For next steps, we believe it would be ideal to validate our developed models on external obesity datasets to assess the robustness across different demographics including other obesity data sets from different institutions, as well as expand the features to daily calorie intake and minutes of exercise. Testing different parameters would also be helpful to measure the model's capabilities. Moreover, it may be interesting to investigate if our classification models can expand its function to longitudinal data such as after diagnosis response to obesity or be used on supplemental data collected from patients in the next ten years. Long term follow up studies could provide valuable insights into the prognostic value of predictive models. In the end, our research highlights the need for a flexible approach in the fight against obesity. By constantly testing and refining our methods, we can make meaningful progress in detecting obesity early and improving patient health and longevity.

# REFERENCES

[1] **Pandas**: Python Data Analysis Library. Retrieved from <https://pandas.pydata.org/>.

[2] **Seaborn**: Statistical Data Visualization. Retrieved from <https://seaborn.pydata.org/>.

[3] **Matplotlib**: Visualization with Python. Retrieved from <https://matplotlib.org/>.

[4] **NumPy**: The fundamental package for scientific computing with Python. Retrieved from <https://numpy.org/>.

[5] **Scikit-learn**: Machine Learning in Python. Retrieved from <https://scikit-learn.org/>.

[6] **Scikit-learn: Model Selection**. Retrieved from <https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection>.

[7] **Scikit-learn: Preprocessing Data**. Retrieved from <https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing>.

[8] **Scikit-learn: Random Forest**. Retrieved from <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>.

[9] **Scikit-learn: Decision Trees**. Retrieved from <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>.

[10] **Scikit-learn: Logistic Regression**. Retrieved from <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>.

[11] **Scikit-learn: Model Evaluation**. Retrieved from <https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics>.

[12] **TensorFlow**: An open-source machine learning framework for everyone. Retrieved from <https://www.tensorflow.org/>.

[13] **TensorFlow: Keras API**. Retrieved from <https://www.tensorflow.org/api_docs>

/python/tf/keras.