A Project on

Age, Weight, Height, BMI Analysis Using Machine Learning

By

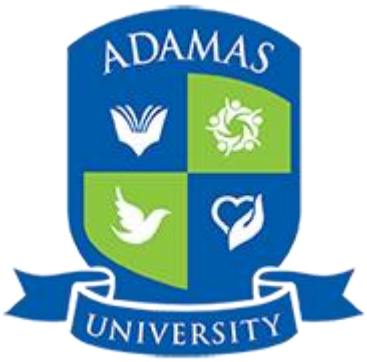
Sahazadi Khatun

Roll No.: UG/05/BSTDA/2021/006

Enrollment No.: AU/2021/0006523

A report submitted in partial fulfillment

of the internship required



Under the supervision of

Dr. Shakti Prasad

(Assistant Professor)

National Institute of Technology (NIT) Arunachal Pradesh

Department of Mathematics

School of Basic and Applied Sciences

ACKNOWLEDGEMENT

I am immensely grateful to my guide Dr. Shakti Prasad sir, for the invaluable guidance and support during my internship project. I would like to express my deepest gratitude to my project guide for their mentorship and insightful direction throughout the project.

Furthermore, I extend my thanks to Adamas University, Kolkata for providing me with the necessary resources and conducive environment to carry out my project effectively. Besides my supervisor and Institutions, I would like to offer my thanks to all the faculty members of Department of Mathematics for their contributions, feedback.

I am privileged to thank all of my classmates for giving me some idea and necessary

atmosphere for the work.

Last but not the least, I am grateful to my family, for their continuous and encouragement, which significantly enhanced the quality and scope of my project.

Thank you once again for the opportunity and support.

Sincerely,

Sahazadi Khatun

Table of Contents

Introduction

In the realm of healthcare, the Body Mass Index (BMI) has been a pivotal metric for assessing general health. However, its conventional interpretation often oversimplifies the intricate web of factors influencing health. In an era dominated by data-driven decision-making, the integration of machine learning techniques has revolutionized the way we interpret and utilize information. In this project, there will be use a secondary data sources for do the analysis and interpretation. The solution is a multi-faceted machine learning toolkit, tailored to consider age, height, weight and bmi.

Aim of the project

- Visualize the potential factor which can help to interpret the BMI Data.
- Using weight, age, height and bmi doing analysis and predict which is the BmiClass ['Obese Class 1', 'Overweight', 'Underweight', 'Obese Class 2', 'Obese Class 3', 'Normal Weight'] of the person.

Dataset

This is taken from a secondary source named kaggle https://www.kaggle.com/datasets/rukenmissonnier/age-weight-height-bmi-analysis , naming (Age, Weight, Height, BMI Analysis).

The dataset in question comprises 741 individual records, each meticulously documented with the following 5 attributes where 4 are numerical and 1 is categorical variables:

SNo.	Variable_Name	Description
1.	Age (in years)	This field quantifies the age of each individual, denominated in years.
2.	Height (in meters):	The "Height" column provides measurements of the subjects' stature in meters.
3.	Weight (in kilograms):	The "Height" column provides measurements of the subjects' stature in meters.
4.	BMI (Body Mass Index):	Derived from the height and weight columns, the BMI column computes the Body Mass Index of each individual. BMI is a vital numerical indicator used for categorizing individuals based on their weight relative to their height. It is expressed as a continuous variable.
5.	BmiClass:	The "BmiClass" column categorizes individuals based on their calculated BMI values.

First few rows in Dataset

	Age	Height	Weight	Bmi	BmiClass
0	61	1.85	109.30	31.935720	Obese Class 1
1	60	1.71	79.02	27.023700	Overweight
2	60	1.55	74.70	31.092612	Obese Class 1
3	60	1.46	35.90	16.841809	Underweight
4	60	1.58	97.10	38.896010	Obese Class 2

Summary of the Dataset

The shape of the dataset is (741, 5).

	Age	Height	Weight	Bmi
count	741.000000	741.000000	741.000000	741.000000
mean	31.618084	1.709427	78.412497	26.365427
std	11.655466	0.085974	32.254547	9.223191
min	15.000000	1.460000	25.900000	12.150497
25%	22.000000	1.670000	63.000000	22.129740
50%	29.000000	1.721000	72.900000	24.132412
75%	40.000000	1.751000	83.300000	27.249306
max	61.000000	2.070000	270.000000	66.301350

Initial Insights about Data

Categorical Features: BmiClass

Numerical Features: Age, Height, Weight & Bmi

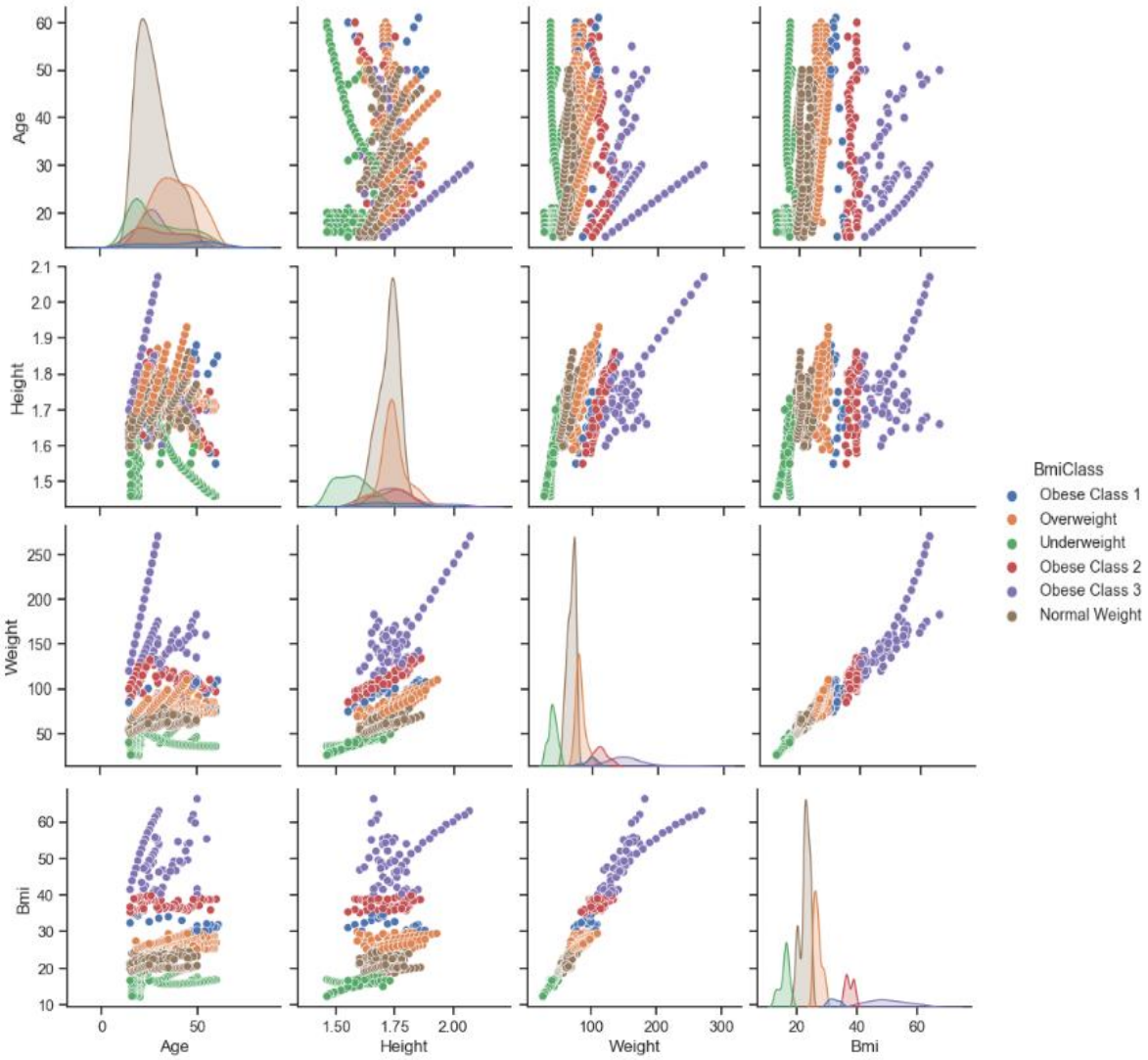
Handling Missing Values

This dataset exhibits a high degree of data integrity, with no missing values across any of the aforementioned columns.

Here, Using data.isnull.sum() shows that in that **data** there is no "NA" value.

Data Visualization

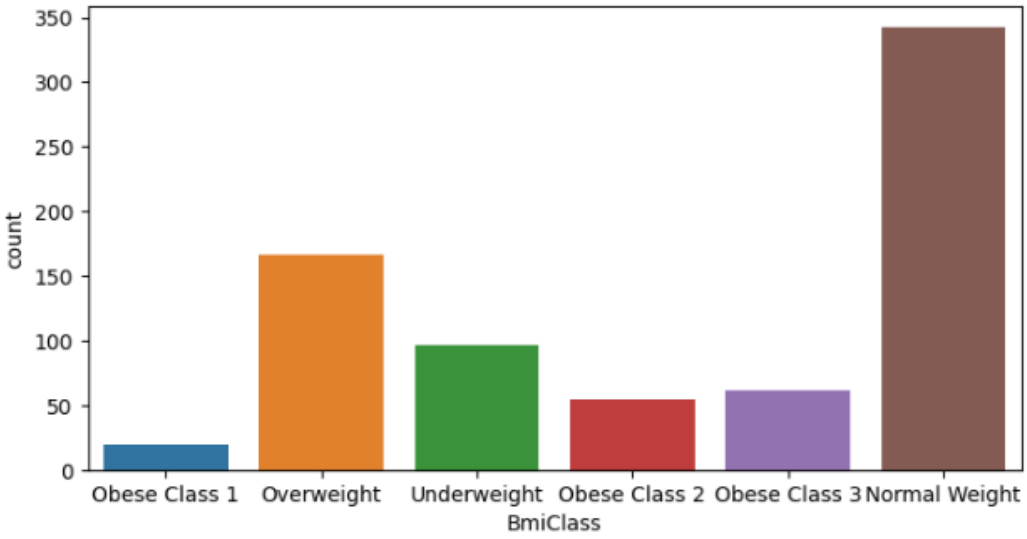
Pair Plot:



Observation:

This is the pairplot between all the features present in the data and with BmiClass ['Obese Class 1', 'Overweight', 'Underweight', 'Obese Class 2', 'Obese Class 3', 'Normal Weight'].

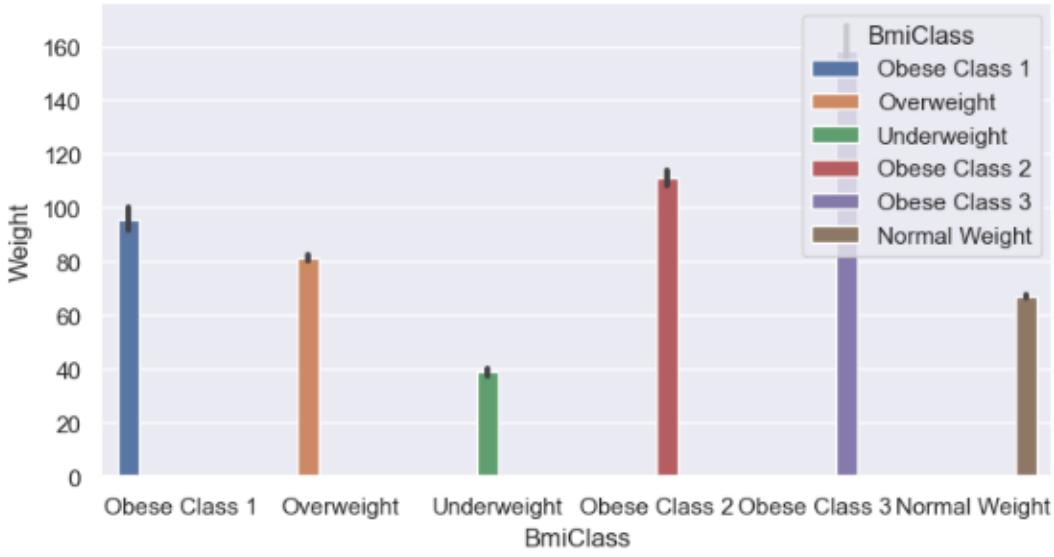
Count Plot of BmiClass in the data



Observation:

In this graph the counts of ['Obese Class 1', 'Overweight', 'Underweight', 'Obese Class 2', 'Obese Class 3', 'Normal Weight'] are plotted. And here we can see No. of "Normal Weight" people are greater in the data.

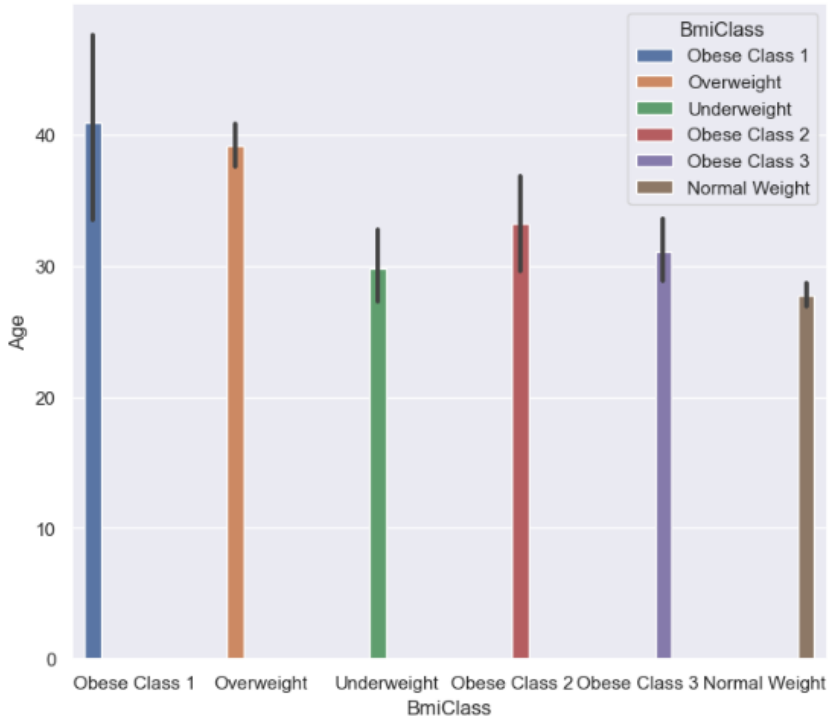
Bar Plot Between Weight and BmiClass



Observation:

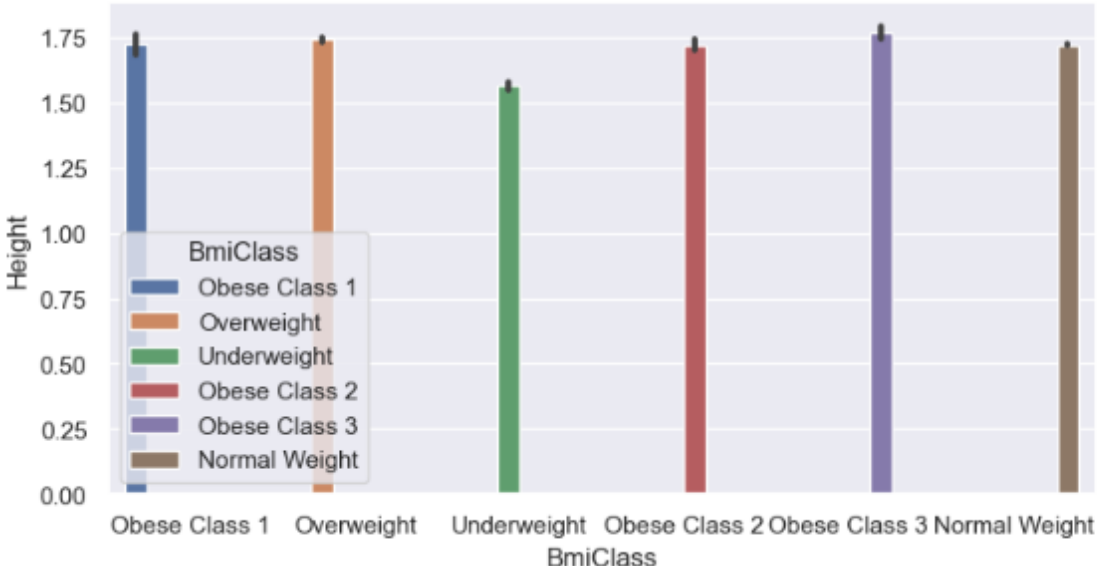
This is the bar plot between Weight and BmiClass. And it can be observed that "Obsese Class 3" people have greater "Weight".

Bar Plot Between Age and BmiClass



Observation:

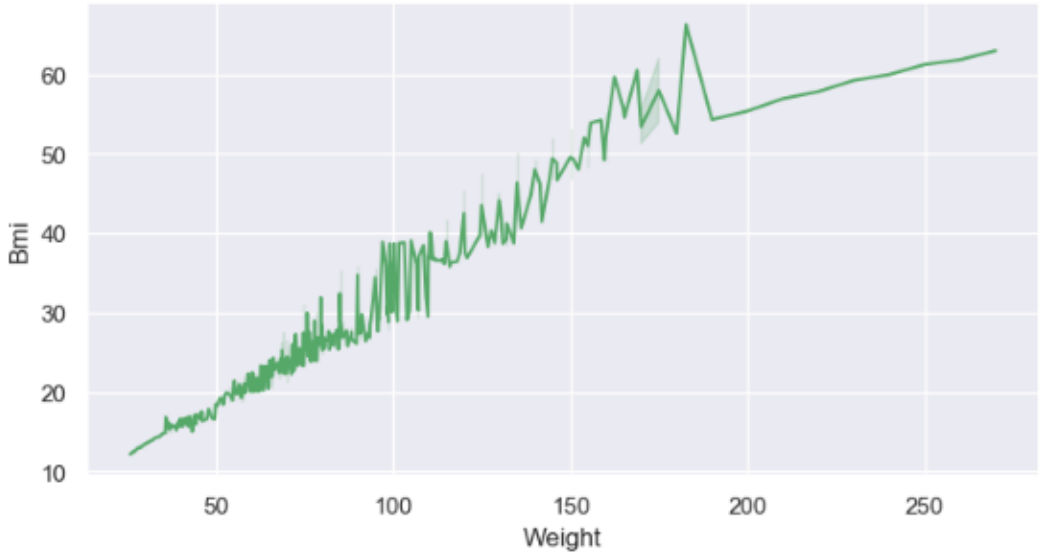
This is the bar plot between Age **and** BmiClass. And it can be observed that greater 'Age ' people are in the "Obsese Class 1" .



Observation:

This is the bar plot between Age and BmiClass. And it can be observed that there is no significant different in "Height" b/w all BmiClass people.

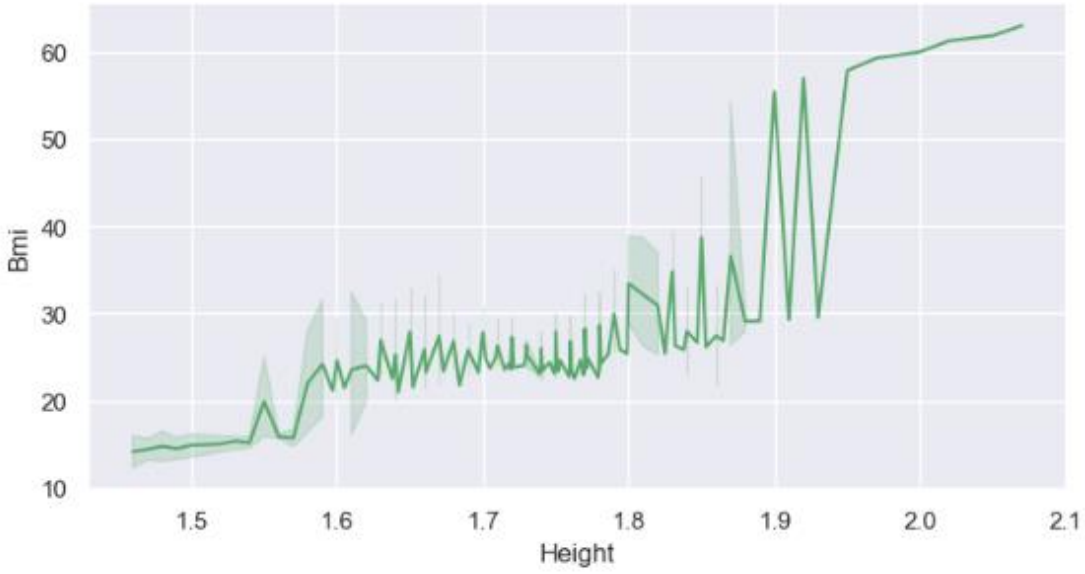
Line Plot Between Weight and Bmi



Observation:

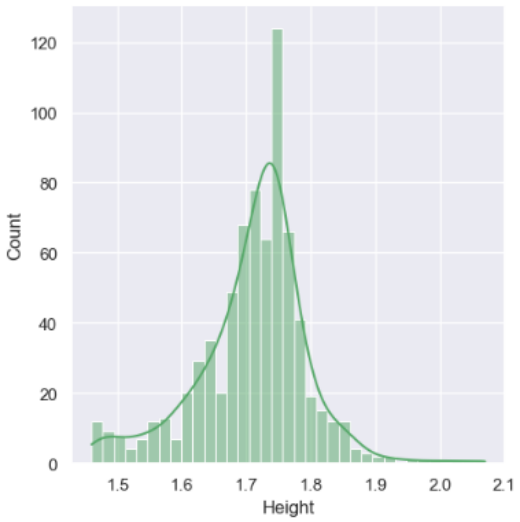
In this is the line plot we can the plot have very much fluctuations but it's clear that Increasing the Weight, Bmi also increases.

Line Plot Between Height and Bmi



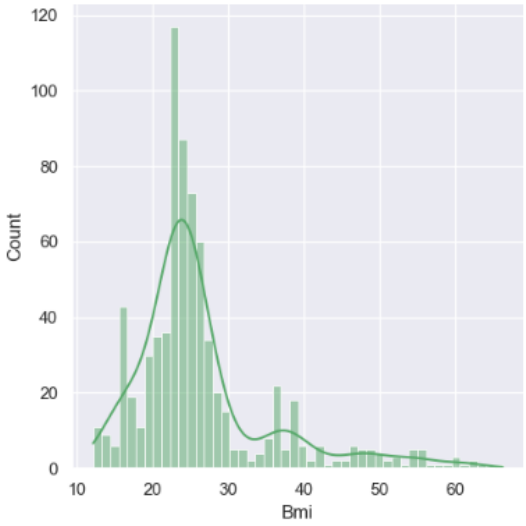
Observation:

In this is the line plot we can the plot have very much fluctuations but it's clear that Increasing the Height, Bmi also increases.



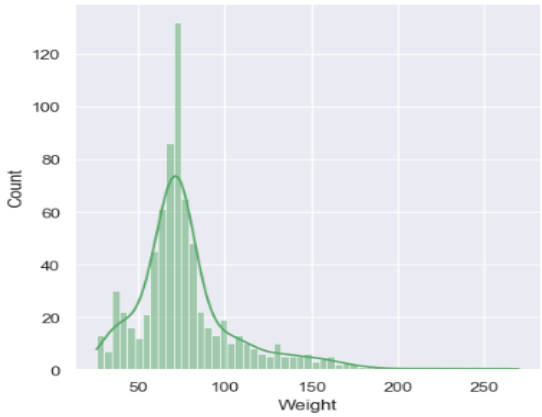
Observation:

Overall Height Distribution in the Data.



Observation:

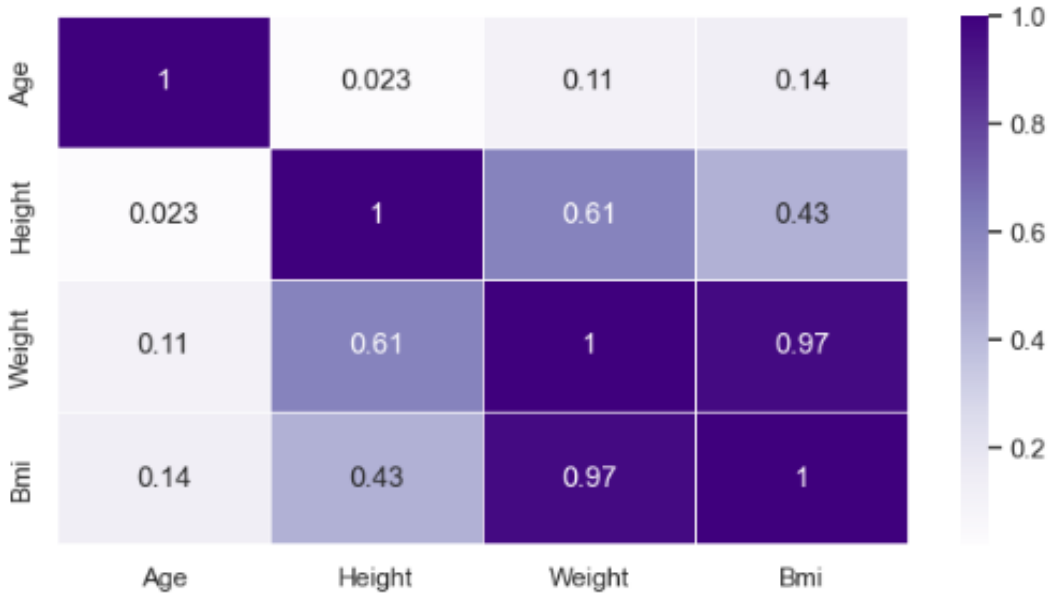
Overall Bmi Distribution in the Data.



Observation:

Overall Weight Distribution in the Data.

HeatMap



Observation:

This is the Heat Map of the data where correlation between all the features are plotted as heat map.

Model

To implement any Machine Learning Model in data there is some common term we need go through for every model implementation such as:

1. **Data Preprocessing:**

   Preprocessing of data is a critical step in the data analysis and machine learning workflow. It involves cleaning and transforming raw data into a format that is suitable for further analysis and modeling. The preprocessing step aims to remove inconsistencies, noise, and irrelevant information from the data, making it more meaningful and easier to work with.
2. **Splitting the Dataset:**

   Data splitting is done to assess the performance of the model. Dataset is split into two subsets, a training set and a testing set. The purpose of this splitting is to train our machine learning model on one subset and evaluate its performance on another to know how well it performs on new, unseen data.

   Dataset is split into training and testing data using train_test_split function present in sklearn.model_selection library. We split our data such that 30% of data is included in test dataset and 70% in train dataset.
3. **Modelling the Dataset:**

   In machine learning, modeling refers to the process of creating a mathematical representation or a computational model of a real-world system based on available data. The goal is to use this model to make predictions or decisions without being explicitly programmed to perform the task.

Choosing the appropriate machine learning algorithm or model based on the nature of the problem (e.g., classification, regression, clustering) and the characteristics of the data.

- **Model Training:** Using a portion of the data (training set) to fit the model by adjusting its parameters. The model learns patterns and relationships in the data during this phase.

- **Model Evaluation:** Assessing the performance of the model on a separate set of data (validation or test set) that it has never seen before. This helps to ensure that the model can generalize well to new, unseen data.
- **Prediction:** Using the trained model to make predictions or decisions on new, unseen data. This is the primary purpose of the machine learning model.

Results:

- **Regression:** In regression tasks, the prediction accuracy is a continuous value. And in order to get the accuracy of the model we can calculate: Mean Squared Error,
    1. **Mean Squared Error:**

       Mean Squared Error (MSE) is a commonly used metric for evaluating the performance of a regression model. It quantifies the average squared difference between the actual values (observed) and the predicted values produced by the model.

       $\mathbf{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y_i})^2$
    2. **Mean Absolute Error (MAE):** This is the average of the absolute differences between the actual and predicted values. It gives a linear measure of the average error.

       $\mathbf{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y_i}|$
    3. **Root Mean Squared Error (RMSE):** This is the square root of the MSE. It provides an interpretable measure of the average error in the same units as the target variable.

       $\mathbf{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y_i})^2}$
    4. **R-squared (R^2):** This metric represents the proportion of variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, where 1 indicates a perfect fit.

       $\mathbf{R\text{-}squared} = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y_i})^2}{\sum_{i=1}^{n} (y_i - \bar{y_i})^2}$
- **Classification:** In classification tasks, the prediction accuracy is a class label or category. For classification model, we can get **Accuracy, Confusion Matrix** and **Classification Report** of the model. And as classification report there will be

Precision, Recall, F-1 Score, Support and we compare them to get the better prediction model.

1. **Accuracy:**

   Accuracy is the ratio of correctly predicted instances to the total number of instances in the dataset. It provides an overall assessment of the model's correctness.

   Accuracy= $\frac{\text{True Positives}+\text{True Negatives}}{\text{Total Instances}}$

2. **Precision:**

   Precision is the ratio of correctly predicted positive observations (True Positives) to the total predicted positive observations (True Positives + False Positives). It measures the accuracy of the positive predictions. $\frac{\text{True Positives}}{\text{True Positives}+\text{False Positives}}$

Precision=

3. **Recall:**

   Recall is the ratio of correctly predicted positive observations (True Positives) to the all observations in actual class (True Positives + False Negatives). It measures the ability of the model to capture all the positive instances.

   Recall= $\frac{\text{True Positives}}{\text{True Positives}+\text{False Negatives}}$

4. **F1-Score:**

   The F1-Score is the harmonic mean of precision and recall. It provides a balanced measure between precision and recall.

   F1-Score= $2 \times \frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$

5. **Support:** Support is the number of actual occurrences of the class in the dataset. It's the number of instances that belong to each class.

Confusion Matrix:

A confusion matrix is a fundamental tool in the evaluation of classification models. It provides a detailed summary of how a classification model is performing, showing the number of true positive, true negative, false positive, and false negative predictions for each class in a classification problem.

Let's break down the components of a confusion matrix:

True Positives (TP): These are the instances that were correctly predicted as belonging to the positive class.

True Negatives (TN): These are the instances that were correctly predicted as belonging to the negative class.

False Positives (FP): These are the instances that were incorrectly predicted as belonging to the positive class when they actually belong to the negative class. Also known as Type I error.

False Negatives (FN): These are the instances that were incorrectly predicted as belonging to the negative class when they actually belong to the positive class. Also known as Type II error.

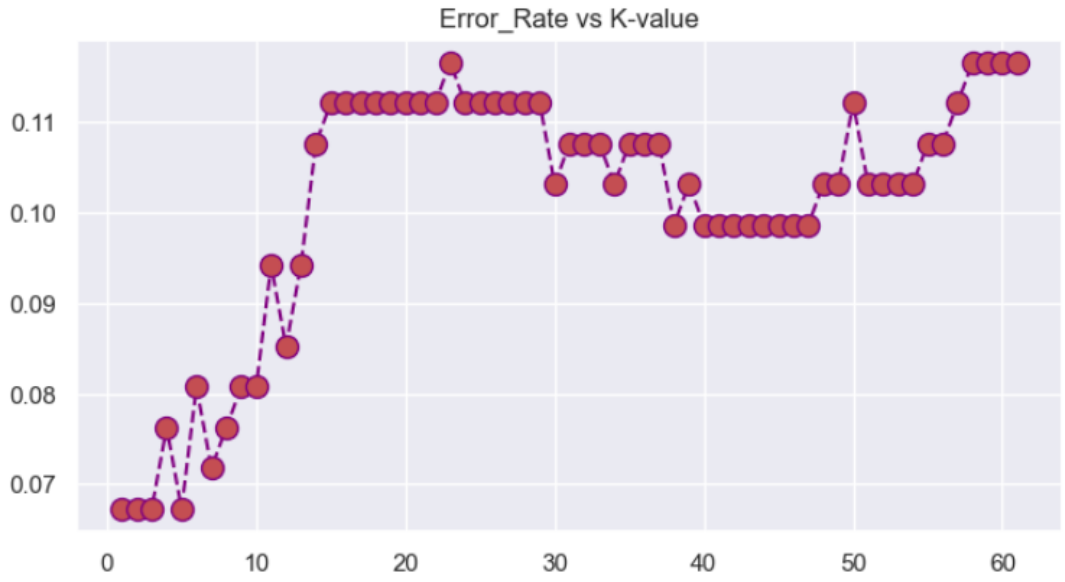
Actual Class			
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Used Machine Learning Models:

1. **KNN Classifier :**

   The k-Nearest Neighbors (KNN) classifier is a non-parametric machine learning model used for both classification and regression tasks. It makes predictions by identifying the k nearest data points in the training set to a given test point. In classification, the mode of the k-nearest neighbors' classes determines the predicted class. KNN is intuitive, easy to implement, and can be effective, particularly in scenarios with well-defined clusters or boundaries.

Error Rate vs. K-value



Observation: In the graph there are plotted the error rate in every k-value.

We assume the appropriate k-value for this dataset is, k=9.

Accuracy in KNN Classifier: 0.9192825112107623

Confusion Matrix in KNN Classifier:

```
array([[101,   0,   0,   0,   0,   0],
       [  0,   1,   2,   0,   3,   0],
       [  0,   0,  10,   3,   0,   0],
       [  0,   0,   2,  17,   0,   0],
       [  8,   0,   0,   0,  49,   0],
       [  0,   0,   0,   0,   0,  27]], dtype=int64)
```

Classification Report of data in KNN Classifier:

```
'              precision    recall  f1-score   support\n\nNormal Weight       0.93      1.00      0.96       101\nObese Class 1       1.00      0.17
0.29         6\nObese Class 2       0.71      0.77      0.74        13\nObese Class 3       0.85      0.89      0.87        19\n  Overweight       0.94
0.86      0.90         57\n  Underweight       1.00      1.00      1.00        27\n\n    accuracy                                        0.92       223\n   macro av
g       0.91      0.78      0.79       223\n weighted avg       0.92      0.92      0.91       223\n'
```

2. Linear Regression Model :

Linear regression is a fundamental machine learning model used for regression tasks. It establishes a linear relationship between independent variables (features) and a dependent variable (target). The model assumes that this relationship can be represented by a straight line.

In simple linear regression, there's only one independent variable, and the relationship is modeled as:

$$y = ax + b$$

Where,

- y is the dependent variable (target),
- x is the independent variable (feature),
- m is the slope of the line, and
- b is the y-intercept.

In the dataset, first we've applied linear regression -- Bmi vs. Height, Bmi vs. Weight.

Bmi vs. Height:

```
# mean_squared_error
Linear_Reg(x,y)[0]
```

```
0.006096558141684684
```

```
# mean_absolute_error
Linear_Reg(x,y)[1]
```

```
0.06067026576255947
```

```
# r2_score
Linear_Reg(x,y)[2]
```

```
0.2013455828819274
```

Bmi vs. Weight:

```
x=data[['Weight']]
y=data[['Bmi']]
```

```
# mean_squared_error
Linear_Reg(x,y)[0]
```

```
5.416325784207428
```

```
# mean_absolute_error
Linear_Reg(x,y)[1]
```

```
1.4607829385305107
```

```
# r2_score
Linear_Reg(x,y)[2]
```

```
0.929236939941767
```

3. Multiple Linear Regression Model :

In multiple linear regression, there are multiple independent variables, and the relationship is extended to:

$$y = b + m_1x_1 + m_2x_2 + \cdots + m_nx_n$$

Where:

- m_i represents the coefficients for each independent variable, and
- x_i is the value of the i-th independent variable.
- **Bmi vs. Height, Weight, Age:**

```
y=data[['Height','Weight','Age']]
x=data[['Bmi']]
```

```
# mean_squared_error
Linear_Reg(x,y)[0]
```

```
67.74670647668675
```

```
# mean_absolute_error
Linear_Reg(x,y)[1]
```

```
4.939740175530802
```

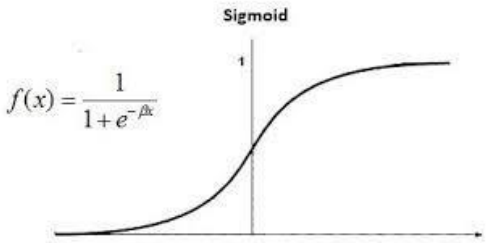
```
# r2_score
Linear_Reg(x,y)[2]
```

```
0.3787883000713144
```

4. Logistic Regression Model :

Logistic regression is a classification algorithm used in machine learning. Despite its name, logistic regression is used for binary classification problems, meaning it's suited for scenarios where the target variable has two possible outcomes, often referred to as classes (e.g., yes/no, 1/0, true/false).

Logistic regression models the probability that a given instance belongs to a particular category. It uses the logistic function (also known as the sigmoid function) to transform the output into a probability score between 0 and 1. The logistic function is defined as:



#Because there is no features as 0 & 1 format....I create a new feature named Having_Thyroid .....Which I take as Randomly True or False---

First Few row of new Dataset:

	Age	Height	Weight	Bmi	BmiClass	Have_Thyroid
0	61	1.85	109.30	31.935720	Obese Class 1	True
1	60	1.71	79.02	27.023700	Overweight	True
2	60	1.55	74.70	31.092612	Obese Class 1	False
3	60	1.46	35.90	16.841809	Underweight	True
4	60	1.58	97.10	38.896010	Obese Class 2	True

In the column Have_Thyroid, the ration of yes or no:

```
Have_Thyroid
True     0.68
False    0.32
Name: proportion, dtype: float64
```

Our dataset is balanced as we can see that about 68% individuals belong to the class having thyroid and 32% is not.

Accuracy in logistic regression:

```
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.6860986547085202
```

Confusion Matrix in logistic regression:

```
print(f'Confusion Matrix: {confusion_matrix}')
```

```
Confusion Matrix: [[  0  70]
 [  0 153]]
```

Classification Report in logistic regression:

```
print(f'Classification Report:\n{report}')
```

```
Classification Report:
              precision    recall  f1-score   support

       False       0.00      0.00      0.00        70
        True       0.69      1.00      0.81       153

    accuracy                           0.69       223
   macro avg       0.34      0.50      0.41       223
weighted avg       0.47      0.69      0.56       223
```

5. Decision Tree :

A Decision Tree is a versatile and intuitive machine learning algorithm used for both classification and regression tasks. It operates by partitioning the feature space into segments, making decisions based on the values of input features. The tree structure consists of nodes, where each node represents a feature, and branches represent possible feature values. It recursively splits the data based on the feature that best

separates the classes or minimizes the variance in regression. Decision Trees are interpretable, making them valuable for understanding model predictions. However, they can be prone to overfitting complex data. Techniques like pruning and ensemble methods are used to mitigate this issue.

Accuracy in Decision Tree:

```
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.9910313901345291
```

Confusion Matrix in Decision Tree:

```
print(f'Confusion Matrix: {confusion_matrix}')
```

```
Confusion Matrix: [[101   0   0   0   0   0]
 [  0   6   0   0   0   0]
 [  0   0  12   1   0   0]
 [  0   0   0  19   0   0]
 [  0   1   0   0  56   0]
 [  0   0   0   0   0  27]]
```

Classification Report in Decision Tree:

```
print(f'Classification Report:\n{report}')
```

```
Classification Report:
                precision    recall  f1-score   support

Normal Weight       1.00      1.00      1.00       101
Obese Class 1       0.86      1.00      0.92         6
Obese Class 2       1.00      0.92      0.96        13
Obese Class 3       0.95      1.00      0.97        19
   Overweight       1.00      0.98      0.99        57
  Underweight       1.00      1.00      1.00        27

     accuracy                           0.99       223
    macro avg       0.97      0.98      0.97       223
 weighted avg       0.99      0.99      0.99       223
```

6. Naive Bayes Classifier :

The Naive Bayes classifier is a probabilistic machine learning algorithm used for classification tasks. It is based on Bayes' theorem and the assumption of independence among features, which is why it's called "naive". Despite its simplicity, Naive Bayes often performs well in a wide range of classification tasks.

Here's how it works:

Bayes' Theorem:

It calculates the probability of a certain event occurring based on the probabilities of related events.

Independence Assumption:

Naive Bayes assumes that features are conditionally independent given the class label. This means that the presence or absence of a particular feature does not influence the presence or absence of any other feature.

Accuracy in Naive Bayes Classifier:

```
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.9506726457399103
```

Classification Report in Naive Bayes Classifier:

```
print(f'Classification Report:\n{report}')
```

```
Classification Report:
                precision    recall  f1-score   support

Normal Weight       0.96      0.96      0.96       110
Obese Class 1       0.67      1.00      0.80         6
Obese Class 2       1.00      0.93      0.97        15
Obese Class 3       1.00      1.00      1.00        17
   Overweight       0.94      0.89      0.91        53
  Underweight       0.96      1.00      0.98        22

     accuracy                           0.95       223
    macro avg       0.92      0.96      0.94       223
 weighted avg       0.95      0.95      0.95       223
```

7. RandomForestRegressor:

The RandomForestRegressor is an ensemble machine learning model widely used for regression tasks. It belongs to the ensemble learning category, which combines the predictions of multiple base estimators to improve overall performance. In the case of RandomForestRegressor, the base estimators are decision trees. This model constructs a forest of decision trees, where each tree is trained on a random subset of the data with replacement (bootstrapped samples) and a random subset of features for each split. During prediction, the individual tree predictions are averaged (regression) to produce the final result. RandomForestRegressor is known for its robustness, accuracy, and resistance to over fitting, making it a popular choice for various regression applications, including finance, ecology, and healthcare.

Mean Squared Error In Random Forest Repressor:

```
mean_squared_error
```

```
92.33247007726995
```

8. **Random Forest Classifier:**

The RandomForestClassifier is an ensemble machine learning algorithm designed for classification tasks. It's a powerful and versatile model known for its high accuracy and robustness. The classifier is composed of a collection of decision trees, each trained on different subsets of the data with bootstrapping. Additionally, it employs feature randomization, considering different subsets of features for each tree. During prediction, the classifier aggregates the results from individual trees, using either a majority vote (for multi-class classification) or a simple vote (for binary classification). RandomForestClassifier is less prone to overfitting compared to individual decision trees, making it particularly effective in complex classification problems. It's widely used in various domains, including image classification, spam detection, and medical diagnosis.

Accuracy in Random Forest Classifier:

```
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.9932885906040269
```

Classification Report in Random Forest Classifier:

```
print(f'Classification Report:\n{report}')
```

```
Classification Report:
                precision    recall  f1-score   support

Normal Weight       1.00      1.00      1.00        59
Obese Class 1       1.00      1.00      1.00         3
Obese Class 2       1.00      0.89      0.94         9
Obese Class 3       0.93      1.00      0.96        13
   Overweight       1.00      1.00      1.00        45
  Underweight       1.00      1.00      1.00        20

     accuracy                           0.99       149
    macro avg       0.99      0.98      0.98       149
 weighted avg       0.99      0.99      0.99       149
```

9. Support Vector Machine :

The Support Vector Machine (SVM) is a powerful supervised machine learning algorithm used for both classification and regression tasks. It aims to find an optimal hyper plane in a high-dimensional space that best separates classes in classification or best fits the data in regression. SVM maximizes the margin between classes, which is the distance between the nearest data points of different classes. It is effective in dealing with non-linearly separable data through the use of kernel functions, which transform the input space into a higher-dimensional space. SVMs are widely employed in tasks such as image classification, text categorization, and medical diagnosis due to their ability to handle complex, high-dimensional data.

Accuracy in Support Vector Machine:

```
accuracy = accuracy_score(np.array(ytest).reshape(-1,1),prediction1)
accuracy
```

```
0.8654708520179372
```

Classification Report in Support Vector Machine:

```
print(f"classification_report: {report}")
```

```
classification_report:                precision    recall  f1-score   support

Normal Weight       0.88      0.94      0.91       105
Obese Class 1       0.00      0.00      0.00         5
Obese Class 2       0.67      0.93      0.78        15
Obese Class 3       0.93      0.81      0.87        16
   Overweight       0.84      0.73      0.78        51
  Underweight       0.97      0.97      0.97        31

     accuracy                           0.87       223
    macro avg       0.71      0.73      0.72       223
 weighted avg       0.85      0.87      0.85       223
```

Conclusion

In conclusion, the internship project delved into a comprehensive exploration of various machine learning techniques, including linear regression, multiple linear regression, K-Nearest Neighbors (KNN) classifier, Naive Bayes classifier, Decision Tree, Random Forest classifier, Random Forest regression, and Support Vector Machine. These methodologies were rigorously applied and meticulously evaluated on the BMI, age, height, weight, and BMI class analysis dataset.

Through a systematic comparative analysis, it was observed that each technique exhibited distinct strengths and weaknesses. Linear regression and multiple linear regression demonstrated commendable performance in predicting BMI based on the given features. The KNN classifier excelled in classifying BMI classes accurately, leveraging proximity-based patterns.

Furthermore, the ensemble methods, including Random Forest classifier and regression, showcased robust predictive capabilities, benefiting from their ability to aggregate multiple decision trees. Naive Bayes and Support Vector Machine algorithms also demonstrated competitive results, emphasizing their versatility in handling complex datasets.

Overall, this project not only provided a comprehensive understanding of various machine learning techniques but also highlighted the importance of selecting the appropriate algorithm based on the specific task at hand. This experience has been invaluable in honing my skills and knowledge in the field of machine learning and data analysis.

Reference

- https://medium.com/analytics-vidhya
- https://github.com/
- **https://www.kaggle.com/**
- Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython-- By Wes McKinney
- Understanding Machine Learning. Shai Shalev-Shwartz and Shai Ben-David. Cambridge University Press. 2017.
- Machine Learning for Big Data, J. Bell Publisher Wiley, 2014