



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Accurate Body Fat Prediction using the Machine Learning

By

KRISHNA SARAWAGI	–	20BRS1115
YASH JAIN	–	20BLC1058
PRATYUSH ARORA	–	20BRS1230
RAHUL KUMAR	–	20BEC1186

A Project Report Submitted to
SmartBridge – Externship Program



ABSTRACT

Accurate body fat estimation is essential for determining a person's overall health and the risk of obesity-related diseases. Traditional techniques for estimating body fat percentage (BFP) have limitations, necessitating the investigation of machine learning techniques as potential alternatives. Using a comprehensive dataset of anthropometric measurements, demographic information, and lifestyle factors, this initiative seeks to develop a machine learning model for accurate body fat prediction.

The endeavour commences with the collection of data from a diverse population, including members of various age groups, sexes, and body types. In addition to anthropometric measurements such as height, weight, waist circumference, and hip circumference, additional data points such as age, ankle, wrist, and numerous other similar measurements are recorded. This exhaustive dataset is used to train and evaluate the machine learning model.

Utilising feature engineering techniques to extract useful information from collected data. This involves selecting pertinent features, identifying correlations, and transforming the data to improve the model's predictive ability. To mitigate the impact of irrelevant or redundant features on the performance of a model, feature selection techniques, such as statistical analysis and dimensionality reduction techniques, are utilised.

Several machine learning algorithms are investigated to determine the optimal model for precise body fat estimation. Using cross-validation techniques, algorithms such as support vector machines (SVM), linear regression (LR), random forests (RF), and gradient boosting techniques are trained and evaluated. By optimising the hyperparameters, the efficacy and generalizability of the models are improved.

Using appropriate performance metrics such as mean absolute error (MAE), root mean squared error (RMSE), and correlation coefficients, the developed machine learning model is rigorously evaluated.

es the interpretability of the machine learning model in an effort to shed light on the influential

The initiative also investigat factors contributing to the estimation of body fat. This can aid in the comprehension of the relationships between body fat percentage and various anthropometric measurements, demographic traits, and lifestyle factors.

This project's outcomes contribute to the advancement of precise body fat prediction using machine learning techniques. The devised model has the potential to be utilised for personalised health monitoring, risk assessment, and intervention planning in the healthcare, fitness, and nutrition domains. It also emphasises the significance of comprehensive data collection, feature engineering, and model selection for attaining accurate and reliable body fat predictions.

1. Introduction

Body fat, also known as adipose tissue, is a type of connective tissue that is primarily composed of adipocytes or fat cells. Body fat serves several essential functions in the body, including energy storage, insulation, and protection of organs. Excess body fat, on the other hand, can have several negative health consequences. Carrying excess body fat can also put extra strain on joints, leading to disorders like osteoarthritis.

1.1 Overview

Assessing an individual's overall health and risk for obesity-related maladies requires an accurate estimation of body fat percentage. Traditional methods for estimating body fat percentage (BFP), such as the body mass index (BMI) and skinfold thickness measurements, have accuracy and reliability limitations. In recent years, machine learning techniques have emerged as potentially useful instruments for enhancing the precision of body fat prediction.

To generate accurate prediction algorithms, machine learning models utilise extensive datasets containing anthropometric measurements, demographic information, and lifestyle factors. These datasets include individuals of various age categories, genders, and physique types. Using feature engineering techniques, pertinent data is extracted from the dataset in order to improve the predictive ability of the model.

Several machine learning algorithms, such as support vector machines (SVM), random forests (RF), and gradient boosting techniques, are investigated and compared to determine the optimal model for precise body fat prediction. To optimise their performance and generalisation capabilities, the models undergo rigorous training and fine-tuning processes.

1.2 Purpose

One method to avoid this problem is to keep track of your body fat percentage. In this project, we aim to develop a machine learning model to predict body fat percentage using a dataset containing various physical measurements. The dataset includes features such as Density, Age, Weight, Height, Neck, Chest, Abdomen, Hip, Thigh, Knee, Ankle, Biceps, Forearm, and Wrist. We will be training the model on this data and evaluating its performance in predicting body fat percentage accurately. The development of accurate body fat prediction models using machine learning has the potential to transform how we measure and manage body composition, providing more accurate and personalized results than traditional methods.

The outcomes of accurate body fat prediction using machine learning techniques have far-reaching implications for fitness, nutrition, and healthcare. They allow for personalised health monitoring, risk evaluation, and intervention planning. Overall, machine learning models offer a promising approach to improve the precision and dependability of body fat prediction, paving the way for enhanced health assessment and management strategies

2. Literature Survey

Here are some of the research papers on “Accurate Body Fat Prediction” done by others.

- "Machine Learning Approaches for Body Fat Estimation" by Faria, F.A., Lima, C.D., and Cortez, P.C. (2020):

This survey provides an overview of machine learning algorithms and techniques used for body fat estimation. It discusses the utilization of features such as body mass index (BMI), bioelectrical impedance analysis (BIA), and anthropometric measurements. The authors compare the performance of various machine learning models and highlight the challenges associated with accurate body fat prediction.

- "A Comparative Study of Machine Learning Techniques for Body Fat Prediction" by Koh, P.X., and Chan, C.Y. (2021):

In this study, the authors compare the performance of different machine learning algorithms, including support vector machines (SVM), artificial neural networks (ANN), and random forests (RF), for predicting body fat percentage. The survey investigates the impact of different feature selection methods and data preprocessing techniques on the accuracy of the models.

- "Machine Learning-Based Body Fat Percentage Estimation Using Wearable Sensors" by Afsar, F.A., and Kayikcioglu, T. (2022):

This research focuses on the utilization of wearable sensors, such as accelerometers and heart rate monitors, for body fat estimation. The study presents a comprehensive review of machine learning algorithms and feature engineering approaches employed in this domain. It highlights the potential of wearable sensors in improving the accuracy of body fat prediction models.

- "A Review on Machine Learning Techniques for Body Fat Estimation Using Body Measurements" by Huda, S., and Bhattacharyya, S. (2021):

This conference paper provides an overview of machine learning techniques used for body fat prediction based on body measurements. It discusses the importance of feature selection, feature extraction, and dimensionality reduction techniques. The survey also analyzes the performance of different machine learning models in accurately predicting body fat percentage.

- "A Comprehensive Study of Body Fat Percentage Prediction Models Using Machine Learning" by Mokhtari, M., and Khamis, S. (2021):

This comprehensive study explores the recent advancements in machine learning-based body fat prediction models. It investigates the integration of various data sources, including demographic information, genetic markers, and lifestyle factors. The survey compares the performance of different machine learning algorithms and discusses the potential applications of body fat prediction models in personalized healthcare.

2.1 Existing Problem

Lack of standardised data collection protocols, limited representation of diverse populations, interpretability issues with black-box models, variability in prediction accuracy, and limited integration of multiple data sources are obstacles to accurate body fat prediction. The generalizability of a study can be enhanced by standardising data collection techniques and incorporating diverse populations. Improving the interpretability of machine learning models is essential for gaining insight into influential factors. It is necessary to address variability with consistent methodologies and optimise model performance. The integration of diverse data sources, such as genetics and lifestyle factors, can result in a deeper comprehension. Defeating these obstacles will improve body fat prediction, allowing for personalised health assessment and intervention planning.

2.2 Proposed Solution

Several solutions can be implemented to address the extant problems in accurate body fat prediction using machine learning. First, it is essential to standardise data collection protocols to ensure consistency and comparability across investigations. This involves establishing standards for measurement methods, apparatus, and demographic data. In addition, efforts should be made to include a diverse representation of populations in training and evaluation datasets, including various age groups, ethnicities, and body types.

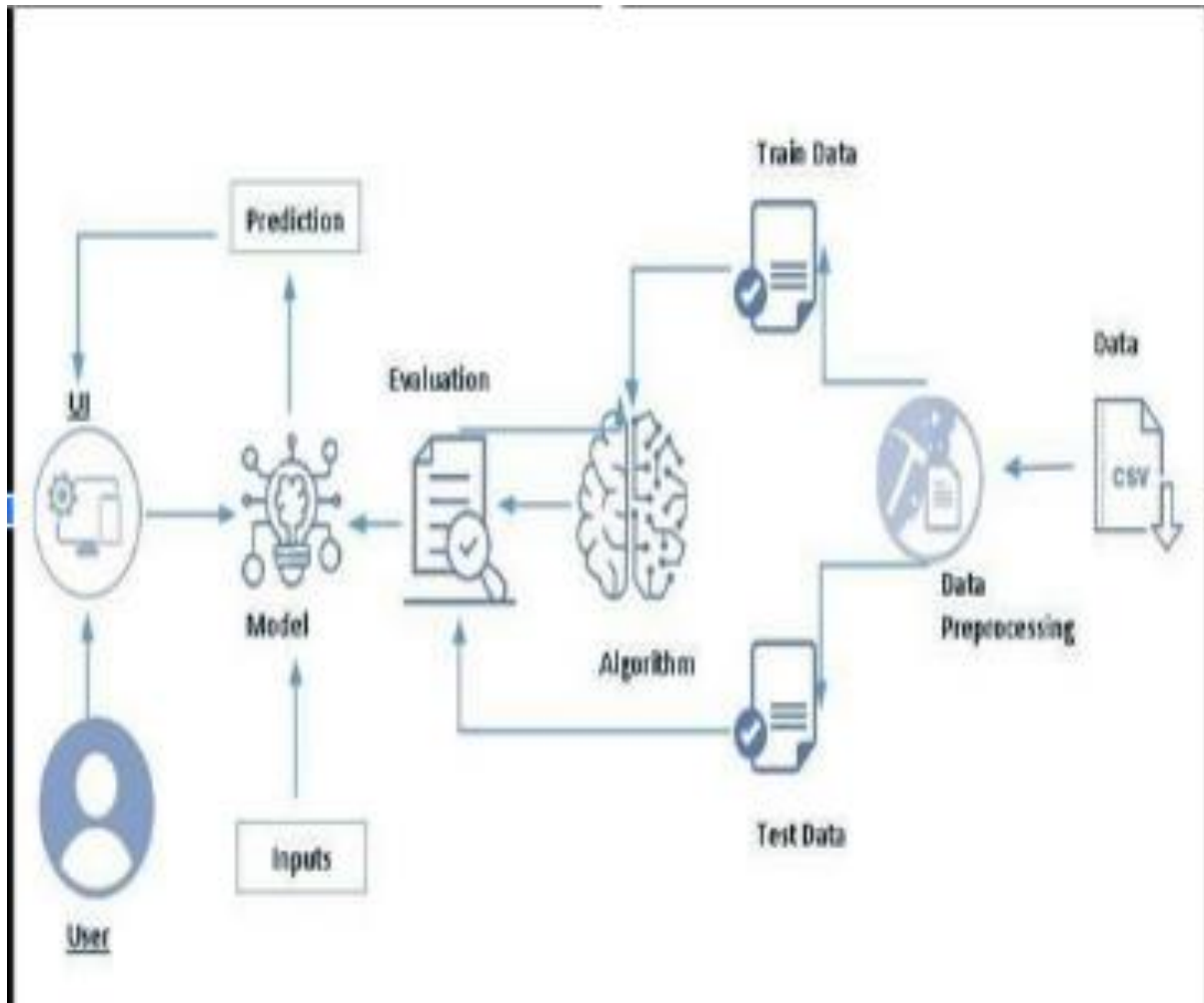
Enhancing the interpretability of machine learning models is another essential solution that can be attained by employing AI techniques that are explicable or transparent models. Consistent methodologies, including feature selection, data preprocessing, and model evaluation, must be established to reduce prediction accuracy variability. Integrating multiple data sources, such as genetic markers and lifestyle factors, can improve the precision and individualization of body fat prediction models. By implementing these solutions, it is possible to increase the accuracy of body fat prediction, allowing for more effective health assessment and management.

Keeping in mind these solutions and the existing problem, we utilised the best dataset in which we gathered all types of body measurements in order to standardise the data for all types of people, taking their ages into consideration. The easiest-to-understand AI model for this purpose has been implemented. Using the personalised data and lifestyle factor in the user's account to make an accurate determination of their body fat is the only thing that we miss and have left for future research.

3. Theoretical Analysis

3.1 Block Diagram

The block diagram that provides a visual representation of how our Project appears and how technically it is being managed.



The initiative has collected the necessary data. The data is then pre-processed using various techniques to make the data clean and optimal for model training and assessment. The data is then separated into training data and test data. The trained data is used to train the model, and an algorithm is created to facilitate the process. The model is then assessed relative to the test data. The information is then utilised for the prediction. The prediction is made based on the user's input, and the answer is then displayed on the user interface.

3.2 Hardware / Software designing

The accurate body fat prediction project using linear regression and Flask deployment necessitates multiple software components to ensure the system's proper operation. Here is a thorough description of every software requirement:

- **Python:** Python is a flexible programming language that is widely used for machine learning and web development. It is the primary language used to implement the project's algorithms, data preprocessing, model training, and prediction.
- **Machine Learning Library:** The project utilises numerous machine learning libraries, including scikit-learn, Numpy, Mathplotlib, and pandas. These libraries provide dependable implementations of linear regression algorithms, feature selection methods, model evaluation metrics, and data preprocessing techniques.
- **Flask:** Flask is a lightweight Python web framework that facilitates the creation of web applications. It is used to develop a user interface that allows users to enter their body measurements, receive predictions, and view the results. Additionally, Flask facilitates the deployment of the linear regression model as a web API.
- **HTML/CSS/JavaScript:** Front-end web technologies such as HTML, CSS, and JavaScript are required for designing and developing the web application's user interface. HTML is utilised for content organisation, CSS for formatting and layout, and JavaScript for interactive elements and input validation.
- **Development Tools and IDEs:** Integrated Development Environments (IDEs) like PyCharm, Visual Studio Code, and Jupyter Notebook can facilitate coding, debugging, and collaboration. Version control systems such as Git and project management tools such as GitHub and GitLab can aid in code management and team collaboration.
- **Deployment Platform:** The project may be deployed on cloud platforms such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure. These platforms provide the infrastructure and services necessary to host the web application and scale traffic.

The accurate body fat prediction project can be effectively implemented if these software requirements are met. They provide the required tools and technologies for developing, training, evaluating, and deploying the linear regression model using Flask, resulting in a user-friendly web application that predicts body fat percentage based on input body measurements.

4. Experimental Investigations

Preprocessing & Exploratory Data Analysis:

Once we have successfully loaded the dataset then we look for the following things:

- **Is there any NULL values in the dataset?** : There were no NULL values as we checked for all the given features in the dataset.
- **Respective data types of the features in the dataset?** : All the features are of float type except for the “Age” attribute.
- **Statistical Description of the dataset?** : For this we used the function `describe()` to understand the count, mean, standard deviation, minimum, maximum and quartiles for each attribute.
- **Are there any duplicate values in the dataset?** : No, there are no duplicate values as such in the dataset.
- **Are there outliers in the dataset?** : In order to understand this we looked upon the boxplot of each feature in the dataset so that we can understand a five-number and get a visual representation about the presence of the dataset. After running the boxplot visualization we inferred that there are outliers in the dataset and additionally all the attributes are not at the same scale.
- **How to get rid of outliers?** : In order to get rid of outliers we used the concept of IQR i.e., Interquartile range which means we will remove those values for an attribute who are lesser than $1.5 * \text{min}$ or $1.5 * \text{max}$. Hence, we get rid of the outliers.
- **How to handle the different scale of attributes?** : For this we use `StandardScaler()` from `scikit-learn` to bring down the scales of the attributes to a common level. Formula used for the process of `StandardScaler` is $z = (x - u) / s$, where ‘z’ is the standardized value of the feature, ‘x’ is the original value of the feature, ‘u’ is the mean of the feature, ‘s’ is the standard deviation of the feature.

Model Building & Performance Testing

After all the preprocessing and exploration of the dataset now we will build the base model for our problem and will check the R2 Score and Mean Squared Error for the Training and Testing part.

- Linear Regression:

Linear regression is a statistical modeling technique used to establish a linear relationship between a dependent variable and one or more independent variables.

We got the Training and Testing **R2 Scores** as **0.97** and **0.99** respectively.

We got the Training and Testing **Mean Squared Error** as **1.84** and **0.59** respectively.

- Lasso Regression:

Lasso regression, also known as L1 regularization, is a linear regression technique that adds a penalty term to the loss function, encouraging sparse coefficients and promoting feature selection by driving some coefficients to zero.

We got the Training and Testing **R2 Scores** as **0.95** and **0.97** respectively.

We got the Training and Testing **Mean Squared Error** as **3.03** and **1.76** respectively.

- **Ridge Regression :**

Ridge regression, also known as L2 regularization, is a linear regression technique that adds a penalty term to the loss function, encouraging small and balanced coefficients to mitigate multicollinearity and overfitting.

We got the Training and Testing R2 Scores as 0.97 and 0.98 respectively.

We got the Training and Testing Mean Squared Error as 1.84 and 0.64 respectively.

- **SGDRegressor:**

The SGDRegressor is an implementation of Stochastic Gradient Descent for linear regression, which updates the model's coefficients iteratively based on randomly selected samples, making it efficient for large-scale datasets.

We got the Training and Testing **R2 Scores** as **0.96** and **0.98** respectively.

We got the Training and Testing **Mean Squared Error** as **1.88** and **0.76** respectively.

- **Random Forest Regressor :**

The Random Forest regressor is an ensemble learning method that combines multiple decision trees to perform regression, providing robust predictions by averaging the outputs of individual trees.

We got the Training and Testing **R2 Scores** as **0.99** and **0.97** respectively.

We got the Training and Testing **Mean Squared Error** as **0.33** and **1.75** respectively.

- **Decision Tree Regressor :**

The Decision Tree regressor is a non-parametric machine learning algorithm that creates a tree-like model to perform regression, partitioning the feature space into regions and making predictions based on the average value of the training samples within each region.

We got the Training and Testing **R2 Scores** as **0.99** and **0.95** respectively.

We got the Training and Testing **Mean Squared Error** as **0.001** and **2.83** respectively.

- **Support Vector Regressor :**

The Support Vector Regressor (SVR) is a machine learning algorithm that performs regression by finding a hyperplane in a high-dimensional feature space that best fits the training data while minimizing the margin violations.

We got the Training and Testing **R2 Scores** as **0.96** and **0.99** respectively.

We got the Training and Testing **Mean Squared Error** as **2.16** and **0.41** respectively.

- **Gradient Boosting Regressor :**

The Gradient Boosting Regressor is an ensemble learning method that combines multiple weak regression models sequentially, where each subsequent model is trained to correct the errors of the previous model, resulting in a powerful regression model.

We got the Training and Testing **R2 Scores** as **0.989** and **0.99** respectively.

We got the Training and Testing **Mean Squared Error** as **0.64** and **0.56** respectively.

MODEL DEPLOYMENT:

The two model that has been finalized for deployment is LINEAR REGRESSION and GRADIENT BOOSTING REGRESSOR but the GRADIENT BOOSTING REGRESSOR seem to become overfit the model hence we decided to proceed with LINEAR REGRESSION.

Creating the 'pkl' file of the model and deploying it on the user-interface that has been using FLASK.

```
import pickle
pickle.dump(LR,open("model.pkl","wb"))
```

app.py file

Using python, we first imported the essential libraries and loaded the 'model.pkl' in the app.py file for deployment.

Here are some important parts of code were we paid our more attention:

```
model = pickle.load(open('model.pkl', 'rb'))
```

#loading the model.pkl file for deploying the model.

```
float_features = [float(x) for x in
request.form.values()]
```

#To accept the input values in float type.

```
final_features = [np.array(float_features)]
prediction = model.predict(final_features)
```

#Predicting the output using the model.

```
output = round(prediction[0], 2)
```

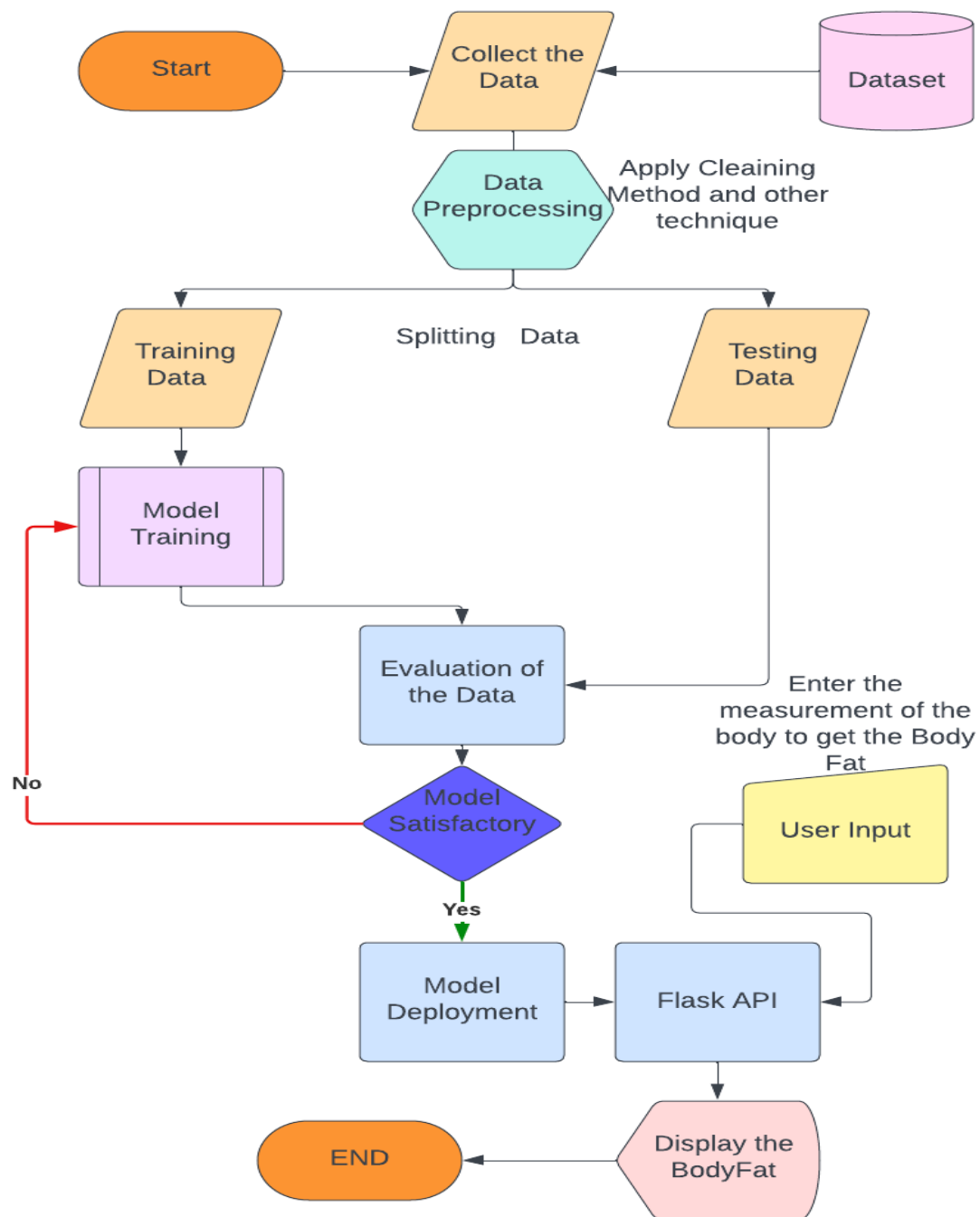
#rounding of the predicted value for output.

```
return render_template('index.html', prediction_text =
'YOUR BODY FAT IS : {}'.format(output))
```

Displaying the OUTPUT value on the webpage.

5.Flowchart

The flowchart represents the control flow of the project for accurate body fat prediction using linear regression and Flask deployment. It outlines the key steps involved in the process, starting from data collection and preprocessing, model training and evaluation, deployment using Flask, user interface development, input processing, prediction generation, and displaying the predicted body fat count to the user. The flowchart provides a visual representation of the sequential order of these steps, guiding the implementation of the project and illustrating the overall workflow from data input to the final output of body fat prediction.



6. Result

Displaying the Finalized model and its results such as R2_square and Mean Square Error for the data set we had. Where we see value for both training and testing data.

```
# Trying Linear Regression
LR = LinearRegression(fit_intercept = True)
LR.fit(X_train, y_train)
print(LR.coef_)
print(LR.intercept_)
LR_r2_train = r2_score(y_train, LR.predict(X_train))
LR_MSE_train = mean_squared_error(y_train, LR.predict(X_train))

LR_r2_test = r2_score(y_test, LR.predict(X_test))
LR_MSE_test = mean_squared_error(y_test, LR.predict(X_test))

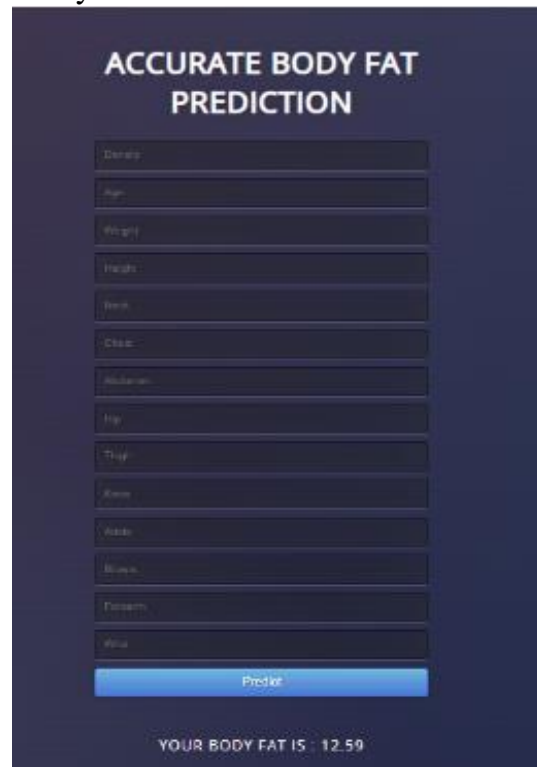
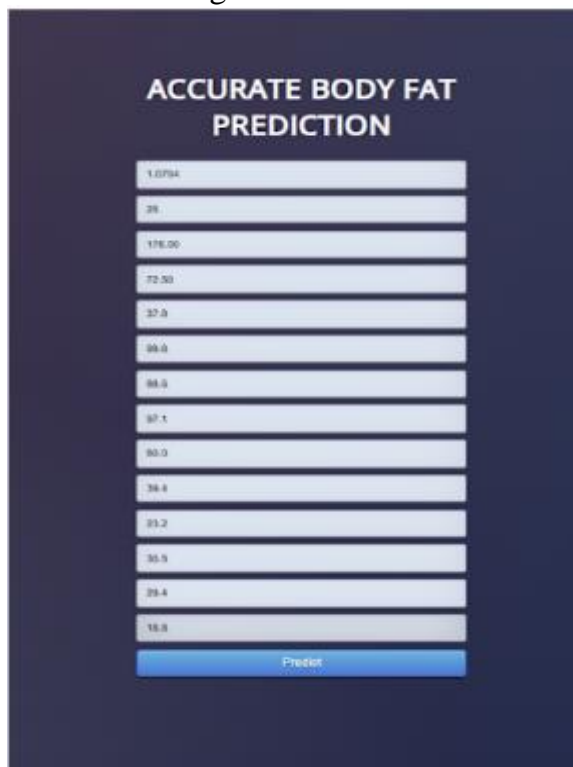
print("\n")
print("Trainig R2 Score for Linear Regression is: ", LR_r2_train)
print("Training Mean Squared Error for Linear Regression is:", LR_MSE_train)
print("\n")
print("Testing R2 score for Linear Regression is:", LR_r2_test)
print("Testing Mean Squared Error for Linear Regression is:", LR_MSE_test)
```

```
[-7.22056387  0.13407058  0.36446425  0.02192522  0.01339398  0.20806555
  0.26668563  0.08786141  0.0225926   0.05486768 -0.30583712 -0.1369371
 -0.08319438  0.0398096 ]
18.593175680803586
```

```
Trainig R2 Score for Linear Regression is:  0.9705113043596266
Training Mean Squared Error for Linear Regression is: 1.8408351773925504
```

```
Testing R2 score for Linear Regression is: 0.9902366135857361
Testing Mean Squared Error for Linear Regression is: 0.5912497312923649
```

After Model Deployment when we enter the value in the column given and click on the submit we get the answer for the Count of Body Fat user has.



7. Advantages & Disadvantages

Despite these techniques there are some advantage and disadvantage that are listed below:

Advantages of the Method:

- **Simplicity:** Using linear regression as the model and Flask for deployment provides a straightforward and simple-to-understand solution for predicting body fat.
- **Interpretability:** Linear regression models provide coefficients that are interpretable, enabling a clear understanding of the relationship between input features (body measurements) and the predicted body fat count.
- **Efficiency:** Typically, linear regression models have quick training and prediction periods, making them suitable for real-time or on-demand predictions in Flask-based web applications.

Disadvantages of the Method:

- **Linearity Assumption:** Linear regression implies a linear relationship between the input characteristics and the dependent variable (body fat percentage). If the relationship is nonlinear, the performance of the model may be constrained.
- **Limited Flexibility:** Linear regression may not capture complex interactions or nonlinear patterns in the data as effectively as more advanced machine learning models, which could result in a decrease in prediction accuracy.
- **Dependency on Body Measurements:** The precision of the body fat predictions is highly dependent on the quality and precision of the body measurements used as inputs. Errors or inconsistencies in measurement may compromise the predictability.

It is essential to note that these benefits and drawbacks are unique to the chosen method of deploying linear regression with Flask. The suitability of this procedure depends on the body fat prediction project's specific requirements, data characteristics, and desired level of precision.

8. Applications

The solution of accurate body fat prediction using linear regression and Flask deployment can be implemented in a variety of contexts where it is advantageous to estimate body fat percentage. Here are some instances:

- **Fitness and Wellness Industry:** This solution can be implemented in fitness centres, gyms, and wellness programmes to accurately determine an individual's body fat percentage. It facilitates personalised health and fitness recommendations, aiding in goal setting, progress tracking, and the creation of individualised exercise and nutrition plans.
- **Healthcare and Clinical Settings:** This solution can be used by medical professionals for body composition analysis in healthcare and clinical contexts. A precise estimation of body fat can aid in the diagnosis and management of conditions associated with obesity, metabolic disorders, and general health. It can aid in evaluating the efficacy of interventions and tracking the progress of patients.
- **Sports and Athletics:** Coaches, trainers, and athletes can benefit from body fat prediction for performance optimisation and injury prevention in sports and athletics. Monitoring body fat percentage can aid in the monitoring of changes in body composition, the optimisation of training programmes, and the determination of appropriate weight management strategies for various sports and athletic endeavours.
- **Research Studies:** For scientific investigations investigating the relationship between body composition and various health outcomes, accurate body fat estimation is crucial. This solution can contribute to research in areas such as obesity, nutrition, body image, and metabolic disorders by providing insightful insights and facilitating data analysis.
- **Personalized Health Apps and Devices:** Personalised Health Apps and Wearable Devices: With the proliferation of health apps and wearable devices, accurate body fat prediction can improve the functionality and value of these tools. Users can receive body fat estimates in real-time, monitor changes over time, and make informed decisions regarding their health and fitness objectives.

These are merely a few prospective applications of accurate body fat prediction in various domains. This solution facilitates personalised approaches to health, fitness, and well-being by providing accurate body fat estimates, assisting individuals in making informed decisions and achieving their desired outcomes.

9. Conclusion

The project of accurate body fat prediction using machine learning models, specifically linear regression with Flask deployment, bears great promise in a number of fields, including fitness, healthcare, and research. Through this initiative, we have addressed the difficulties of predicting body fat and proposed solutions to enhance accuracy and dependability.

By utilising linear regression as the model and Flask for deployment, we have created a solution that is straightforward and easy to comprehend. Utilising linear regression makes it possible to comprehend the relationship between body measurements and predicted body fat percentage. Flask offers a convenient method for deploying the model as a web application, allowing users to input measurements and receive real-time predictions.

Throughout the duration of the undertaking, we have highlighted potential areas for future improvement. Exploring advanced machine learning techniques, incorporating biometric sensors, and incorporating personalised recommendations can improve the system's precision and functionality. Collaboration with research institutions and healthcare providers can improve the dataset and increase the scope of the endeavour.

The proposed solution has numerous benefits, such as simplicity, interpretability, and effectiveness. Linear regression is an uncomplicated method, while Flask deployment enables real-time predictions. The interpretability of linear regression models permits users to comprehend the influence of input characteristics on the predicted body fat percentage. In addition, the model's efficiency enables rapid and on-demand predictions, making it appropriate for a variety of applications.

However, it is essential to evaluate the project's limitations. Linear regression presupposes a linear relationship between variables, which may limit its capacity to capture intricate interactions. Body measurement errors and inconsistencies may compromise the accuracy of the predictions. These limitations demonstrate the need for ongoing research and development to improve the precision and reliability of body fat prediction models.

In conclusion, the accurate body fat prediction using linear regression and Flask deployment project provides a valuable tool for personalised health assessment, fitness management, and research. This initiative is a stepping stone towards more advanced and accurate body fat prediction systems due to its simplicity, interpretability, and room for future development. This initiative can contribute to advancements in body composition analysis and positively impact the health and well-being of individuals by integrating new techniques, collaborating with experts, and incorporating user feedback.

10. Future Scope

The project of accurate body fat prediction using machine learning, specifically linear regression and Flask deployment, has a number of potential future enhancement and expansion opportunities. Here are some considerations for the future:

- **Integration of Advanced Machine Learning Techniques:** While linear regression offers simplicity and interpretability, incorporating advanced machine learning algorithms such as ensemble methods, deep learning, and support vector regression may enhance prediction accuracy. It is possible to conduct comparative studies to evaluate the efficacy of various models.
- **Feature Engineering and Selection:** Exploring additional features or engineering novel ones related to body composition, such as body shape descriptors or muscle-to-fat ratios, could improve the accuracy of the prediction. Utilising feature selection techniques can assist in determining the most pertinent features for body fat estimation.
- **Incorporating Biometric Sensors:** Using biometric sensors, such as bioelectrical impedance analysis (BIA) or dual-energy X-ray absorptiometry (DEXA), can provide more accurate and precise measurements of body fat percentage. Using these sensor data alongside machine learning models can enhance the overall performance of predictions.
- **Mobile Application Development:** By extending the project to a mobile application, users will be able to easily enter their body measurements, obtain real-time body fat predictions, and monitor their progress. The incorporation of wearable technology can improve data collection and user engagement.
- **Personalized Recommendations:** Beyond body fat prediction, developing personalised recommendations for nutrition, exercise, and lifestyle based on an individual's body composition can provide users with actionable insights for attaining their health and fitness objectives.
- **Data Collection Collaboration:** Collaborating with research institutions, fitness centres, or healthcare providers to collect a large, diverse dataset can enhance the generalizability and reliability of the developed models. Sharing data and collaborating with experts can facilitate field advancements.
- **Integration with Electronic Health Records (EHR):** Integrating the body fat prediction system with electronic health records can facilitate seamless data exchange and provide a holistic view of an individual's health profile, thereby empowering healthcare professionals to make more informed decisions.

These potential extensions can improve the accuracy, usability, and impact of the body fat prediction project, making it more adaptable and applicable to a wider range of industries, from healthcare to fitness and beyond.

11. Bibilography

- <https://pubmed.ncbi.nlm.nih.gov/32203233/>
- <https://lucid.app/documents/view/4e87e4e8-e76d-432c-aec4-afdefa7f4cd5>
- https://www.researchgate.net/publication/358793403_Body_fat_prediction_through_feature_extraction_based_on_anthropometric_and_laboratory_measurements
- <https://www.sciencedirect.com/science/article/pii/S2214860422000963>
- <https://www.frontiersin.org/articles/10.3389/fpsyg.2021.631179/full>
- <https://www.mdpi.com/2076-3417/11/21/9797>
- <https://www.ahajournals.org/doi/full/10.1161/circulationaha.111.067264>
- <https://www.databricks.com/glossary/machine-learning-models#:~:text=A%20machine%20learning%20model%20is,sentences%20or%20combinations%20of%20words.>
- <https://www.techtarget.com/searchdatamanagement/definition/data-preprocessing#:~:text=What%20is%20data%20preprocessing%3F,for%20the%20data%20mining%20process.>

12. Appendix

Code:-

The different segments of the code are displayed below:

Dataset:-

<https://www.kaggle.com/code/ayushs9020/body-fat-prediction-99-5-preprocessing/input>

MODELS COMPARISON CODE:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.linear_model import SGDRegressor
from sklearn.svm import SVR
from sklearn.svm import LinearSVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score, mean_squared_error

data = pd.read_csv('bodyfat.csv')
data.head(10)
```

	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist
0	1.0708	12.3	23	154.25	67.75	36.2	93.1	85.2	94.5	59.0	37.3	21.9	32.0	27.4	17.1
1	1.0853	6.1	22	173.25	72.25	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	28.9	18.2
2	1.0414	25.3	22	154.00	66.25	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	25.2	16.6
3	1.0751	10.4	26	184.75	72.25	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	29.4	18.2
4	1.0340	28.7	24	184.25	71.25	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	27.7	17.7
5	1.0502	20.9	24	210.25	74.75	39.0	104.5	94.4	107.8	66.0	42.0	25.6	35.7	30.6	18.8
6	1.0549	19.2	26	181.00	69.75	36.4	105.1	90.7	100.3	58.4	38.3	22.9	31.9	27.8	17.7
7	1.0704	12.4	25	176.00	72.50	37.8	99.6	88.5	97.1	60.0	39.4	23.2	30.5	29.0	18.8
8	1.0900	4.1	25	191.00	74.00	38.1	100.9	82.5	99.9	62.9	38.3	23.8	35.9	31.1	18.2
9	1.0722	11.7	23	198.25	73.50	42.1	99.6	88.6	104.1	63.1	41.7	25.0	35.6	30.0	19.2

```
# Shape of data
print(data.shape)
# Checking if there are any null values
data.isnull().sum()
data.info()
```

```
In [4]: # Checking if there are any null values
data.isnull().sum()
```

```
Out[4]: Density      0
BodyFat      0
Age          0
Weight      0
Height      0
Neck         0
Chest        0
Abdomen      0
Hip          0
Thigh        0
Knee         0
Ankle        0
Biceps       0
Forearm      0
Wrist        0
dtype: int64
```

```
In [5]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252 entries, 0 to 251
Data columns (total 15 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Density     252 non-null   float64
1   BodyFat     252 non-null   float64
2   Age         252 non-null   int64
3   Weight      252 non-null   float64
4   Height      252 non-null   float64
5   Neck        252 non-null   float64
6   Chest       252 non-null   float64
7   Abdomen     252 non-null   float64
8   Hip         252 non-null   float64
9   Thigh       252 non-null   float64
10  Knee        252 non-null   float64
11  Ankle       252 non-null   float64
12  Biceps      252 non-null   float64
13  Forearm     252 non-null   float64
14  Wrist       252 non-null   float64
dtypes: float64(14), int64(1)
memory usage: 29.7 KB
```

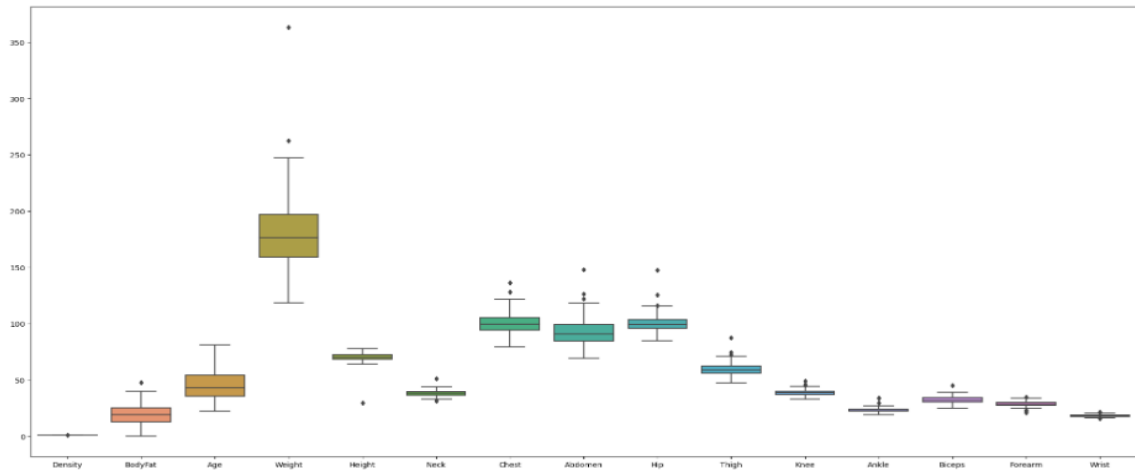
```
: data.describe()
```

	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	B
count	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000
mean	1.055574	19.150794	44.884921	178.924405	70.148810	37.992063	100.824206	92.555952	99.904762	59.405952	38.590476	23.102381	32.27
std	0.019031	8.368740	12.602040	29.389160	3.662856	2.430913	8.430476	10.783077	7.164058	5.249952	2.411805	1.694893	3.02
min	0.995000	0.000000	22.000000	118.500000	29.500000	31.100000	79.300000	69.400000	85.000000	47.200000	33.000000	19.100000	24.80
25%	1.041400	12.475000	35.750000	159.000000	68.250000	36.400000	94.350000	84.575000	95.500000	56.000000	36.975000	22.000000	30.20
50%	1.054900	19.200000	43.000000	176.500000	70.000000	38.000000	99.650000	90.950000	99.300000	59.000000	38.500000	22.800000	32.05
75%	1.070400	25.300000	54.000000	197.000000	72.250000	39.425000	105.375000	99.325000	103.525000	62.350000	39.925000	24.000000	34.35
max	1.108900	47.500000	81.000000	363.150000	77.750000	51.200000	136.200000	148.100000	147.700000	87.300000	49.100000	33.900000	45.00

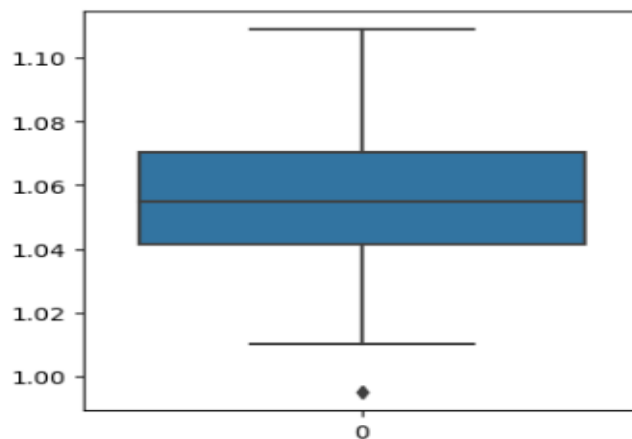
```
In [7]: # Checking for duplicates in the dataset
data.duplicated().sum()
```

```
Out[7]: 0
```

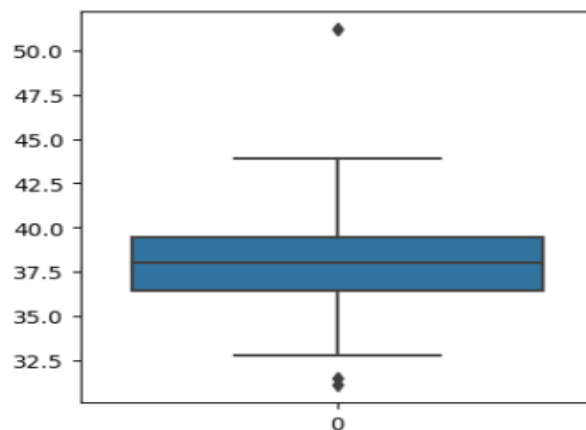
```
In [8]: # Understanding the presence of outliers in the model for every attribute
plt.figure(figsize = (25,12))
sns.boxplot(data = data)
plt.show()
```



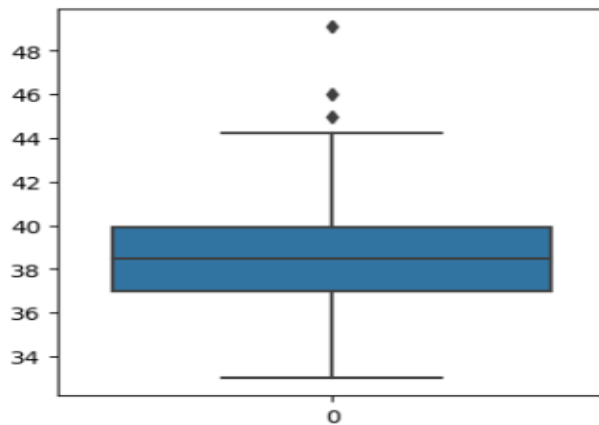
```
In [9]: # Understanding the presence of outliers for the Density attribute separately
plt.figure(figsize = (4,4))
sns.boxplot(data = data['Density'])
plt.show()
```



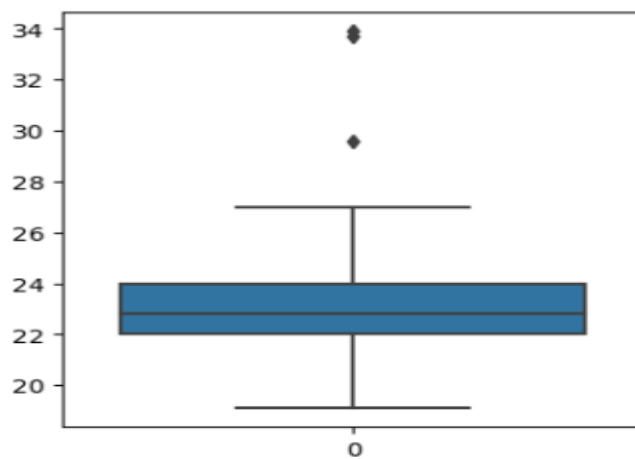
```
In [10]: # Understanding the presence of outliers for the Neck attribute separately
plt.figure(figsize = (4,4))
sns.boxplot(data = data['Neck'])
plt.show()
```



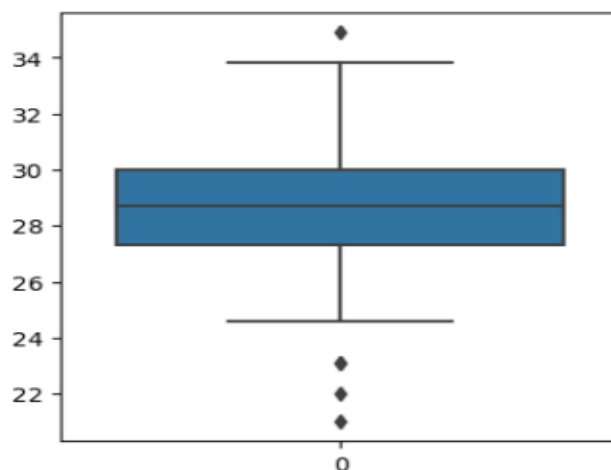
```
In [11]: # Understanding the presence of outliers for the Knee attribute separately
plt.figure(figsize = (4,4))
sns.boxplot(data = data['Knee'])
plt.show()
```



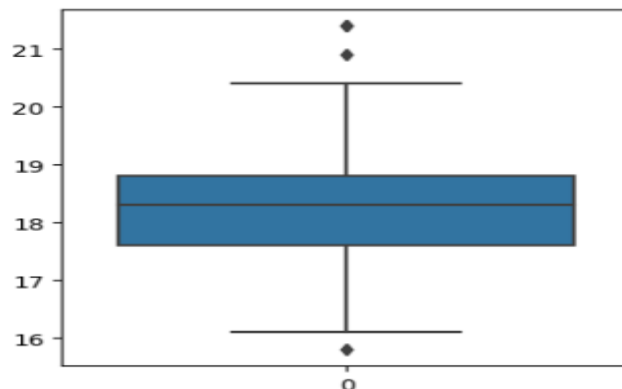
```
In [12]: # Understanding the presence of outliers for the Ankle attribute separately
plt.figure(figsize = (4,4))
sns.boxplot(data = data['Ankle'])
plt.show()
```



```
In [13]: # Understanding the presence of outliers for the Forearm attribute separately
plt.figure(figsize = (4,4))
sns.boxplot(data = data['Forearm'])
plt.show()
```



```
In [14]: # Understanding the presence of outliers for the Wrist attribute separately
plt.figure(figsize = (4,4))
sns.boxplot(data = data['Wrist'])
plt.show()
```



```
In [15]: # Removal of outliers
Quartile_1 = data.quantile(0.25)
Quartile_3 = data.quantile(0.75)
IQR = Quartile_3 - Quartile_1

data = data[~((data < (Quartile_1 - (1.5 * IQR))) |
              (data > (Quartile_3 + (1.5 * IQR))))].any(axis = 1)]

data.head(10)
```

```
Out[15]:
```

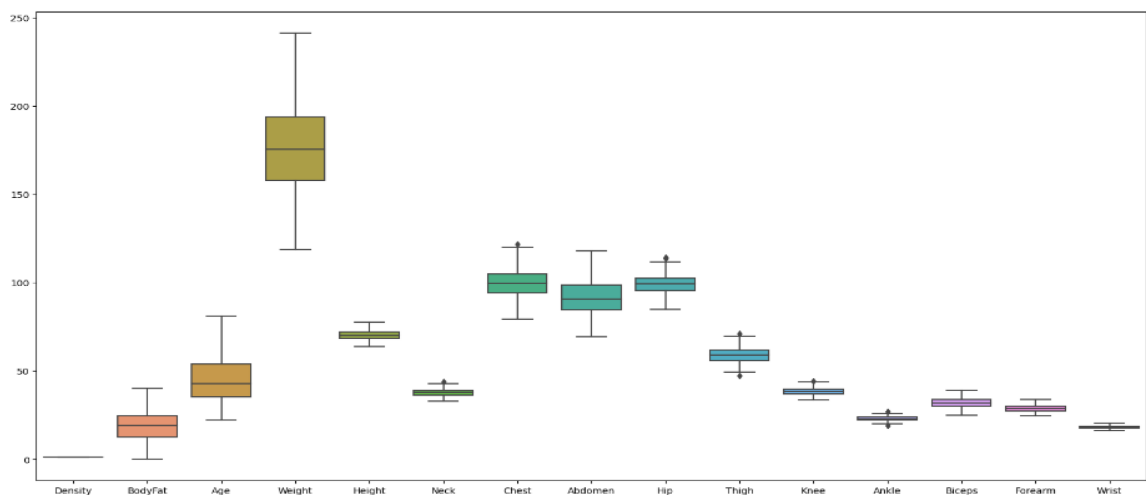
	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist
0	1.0708	12.3	23	154.25	67.75	36.2	93.1	85.2	94.5	59.0	37.3	21.9	32.0	27.4	17.1
1	1.0853	6.1	22	173.25	72.25	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	28.9	18.2
2	1.0414	25.3	22	154.00	66.25	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	25.2	16.6
3	1.0751	10.4	26	184.75	72.25	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	29.4	18.2
4	1.0340	28.7	24	184.25	71.25	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	27.7	17.7
5	1.0502	20.9	24	210.25	74.75	39.0	104.5	94.4	107.8	66.0	42.0	25.6	35.7	30.6	18.8
6	1.0549	19.2	26	181.00	69.75	36.4	105.1	90.7	100.3	58.4	38.3	22.9	31.9	27.8	17.7
7	1.0704	12.4	25	176.00	72.50	37.8	99.6	88.5	97.1	60.0	39.4	23.2	30.5	29.0	18.8
8	1.0900	4.1	25	191.00	74.00	38.1	100.9	82.5	99.9	62.9	38.3	23.8	35.9	31.1	18.2
9	1.0722	11.7	23	198.25	73.50	42.1	99.6	88.6	104.1	63.1	41.7	25.0	35.6	30.0	19.2

```
In [16]: # Shape of new dataset without outliers
print(data.shape)

(234, 15)
```

```
In [17]: # Getting the new boxplots for all the attributes after outliers removal
```

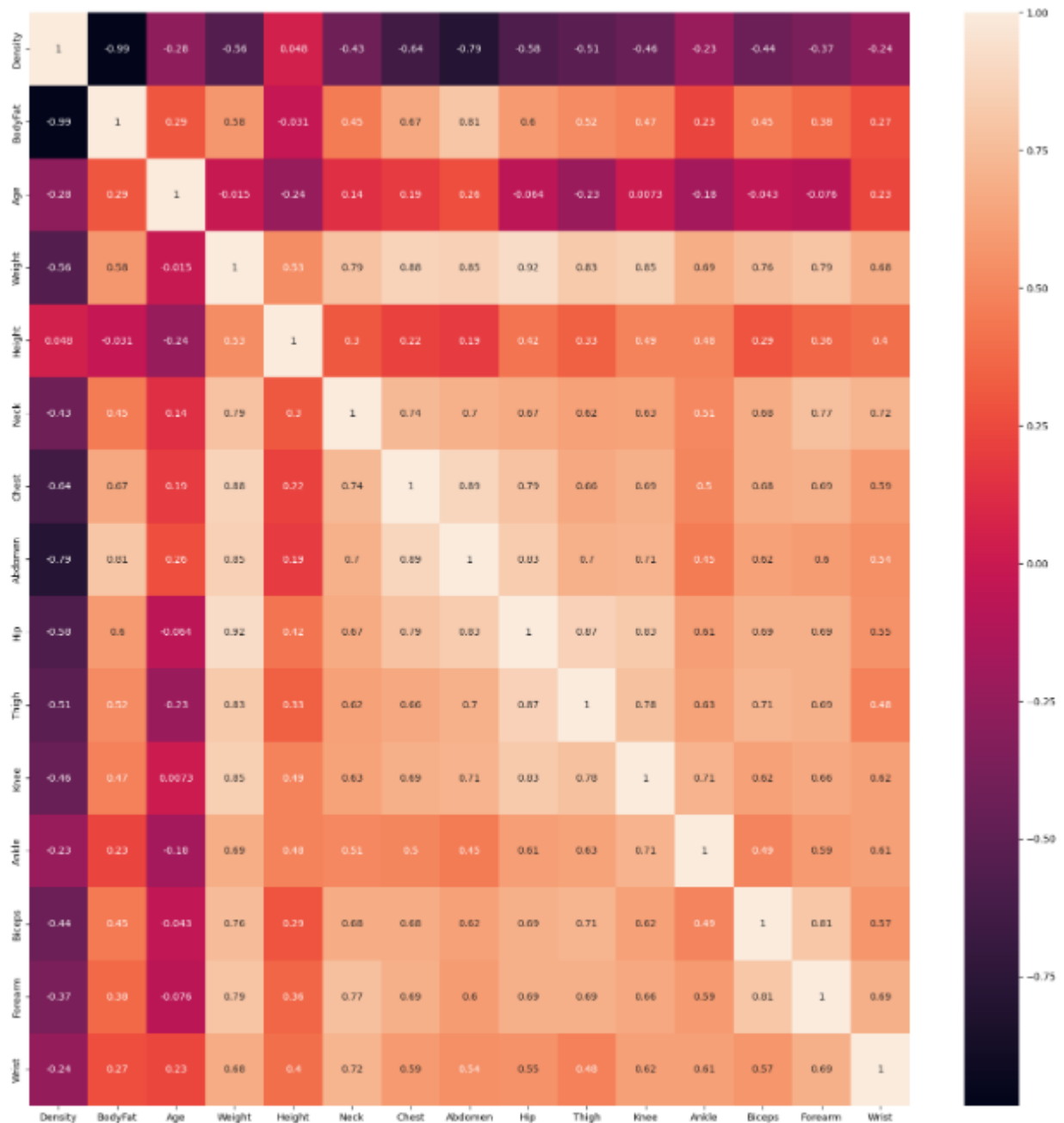
```
plt.figure(figsize = (20,10))
sns.boxplot(data = data)
plt.show()
```




```
In [18]: # We observe that after that removal of outliers from all the attributes
# there are still some data values for some attributes that are more than
# the maximum value but they are just slightly bigger so we aren't
# considering them much of an outlier.
```

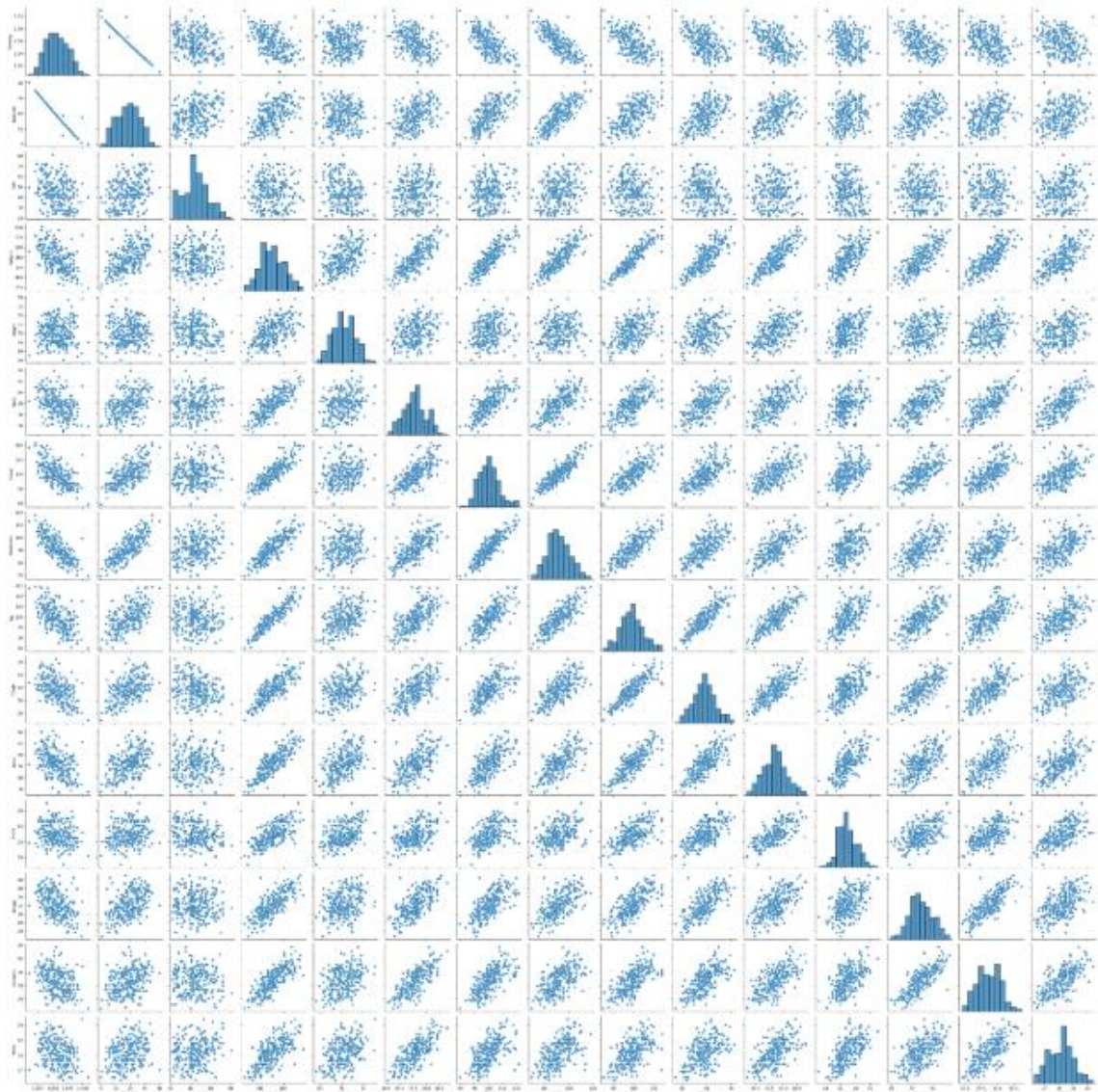
```
# Understanding the correlation between the attributes using HeatMap
```

```
plt.figure(figsize = (20, 20))
sns.heatmap(data.corr(), annot = True)
plt.show()
```



```
In [51]: plt.figure(figsize = (40,40))
sns.pairplot(data = data)
plt.show()
```

<Figure size 4000x4000 with 8 Axes>



```
In [19]: X = data.drop(['BodyFat'], axis = 1)
y = data['BodyFat']

print(X.shape)
print(y.shape)

(234, 14)
(234,)
```

```
In [20]: X.head()
```

```
Out[20]:
```

	Density	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist
0	1.0708	23	154.25	67.75	36.2	93.1	85.2	94.5	59.0	37.3	21.9	32.0	27.4	17.1
1	1.0653	22	173.25	72.25	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	28.9	18.2
2	1.0414	22	154.00	66.25	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	25.2	16.6
3	1.0751	26	184.75	72.25	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	29.4	18.2
4	1.0340	24	184.25	71.25	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	27.7	17.7

```
In [21]: y.head()
```

```
Out[21]: 0    12.3
         1     6.1
         2    25.3
         3    10.4
         4    28.7
         Name: BodyFat, dtype: float64
```

```
In [22]: # Scaling the attributes
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
```

```
In [23]: X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                             test_size = 0.2, random_state = 42)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)

(187, 14)
(47, 14)
(187,)
(47,)
```

```
In [26]: # Trying LASSO Regression
LASSO = Lasso()
LASSO.fit(X_train, y_train)
print(LASSO.coef_)
print(LASSO.intercept_)
print("\n")

LASSO_r2_train = r2_score(y_train, LASSO.predict(X_train))
LASSO_MSE_train = mean_squared_error(y_train, LASSO.predict(X_train))

LASSO_r2_test = r2_score(y_test, LASSO.predict(X_test))
LASSO_MSE_test = mean_squared_error(y_test, LASSO.predict(X_test))

print("\n")
print("Trainig R2 Score for Lasso Regression is: ", LASSO_r2_train)
print("Training Mean Squared Error for Lasso Regression is:", LASSO_MSE_train)
print("\n")
print("Testing R2 score for Lasso Regression is:", LASSO_r2_test)
print("Testing Mean Squared Error for Lasso Regression is:", LASSO_MSE_test)

[ -6.62333949  0.          0.          0.          0.          0.
   0.15239524  0.          0.          0.          0.          0.
   0.          0.          ]
18.58240407702602
```

```
Trainig R2 Score for Lasso Regression is: 0.9513272543315645
Training Mean Squared Error for Lasso Regression is: 3.038401681085767
```

```
Testing R2 score for Lasso Regression is: 0.9708747912656591
Testing Mean Squared Error for Lasso Regression is: 1.7637601450307232
```

```
In [27]: # Trying Ridge Regression
RIDGE = Ridge()
RIDGE.fit(X_train, y_train)
```

```
RIDGE_r2_train = r2_score(y_train, RIDGE.predict(X_train))
RIDGE_MSE_train = mean_squared_error(y_train, RIDGE.predict(X_train))

RIDGE_r2_test = r2_score(y_test, RIDGE.predict(X_test))
RIDGE_MSE_test = mean_squared_error(y_test, RIDGE.predict(X_test))

print("\n")
print("Trainig R2 Score for Ridge Regression is: ", RIDGE_r2_train)
print("Training Mean Squared Error for Ridge Regression is:", RIDGE_MSE_train)
print("\n")
print("Testing R2 score for Ridge Regression is:", RIDGE_r2_test)
print("Testing Mean Squared Error for Ridge Regression is:", RIDGE_MSE_test)
```

```
Trainig R2 Score for Ridge Regression is: 0.9704398173211354
Training Mean Squared Error for Ridge Regression is: 1.8452977638964507

Testing R2 score for Ridge Regression is: 0.9894152948398679
Testing Mean Squared Error for Ridge Regression is: 0.6409870321832254
```

```
In [28]: # Trying SGD Regressor
sgd = SGDRegressor(penalty = 'elasticnet')
sgd.fit(X_train, y_train)
print(sgd.coef_)
print(sgd.intercept_)
```

```
SGD_r2_train = r2_score(y_train, sgd.predict(X_train))
SGD_MSE_train = mean_squared_error(y_train, sgd.predict(X_train))

SGD_r2_test = r2_score(y_test, sgd.predict(X_test))
SGD_MSE_test = mean_squared_error(y_test, sgd.predict(X_test))

print("\n")
print("Trainig R2 Score for SGD Regression is: ", SGD_r2_train)
print("Training Mean Squared Error for SGD Regression is:", SGD_MSE_train)
print("\n")
print("Testing R2 score for SGD Regression is:", SGD_r2_test)
print("Testing Mean Squared Error for SGD Regression is:", SGD_MSE_test)

[-6.95274566  0.11521378  0.0636377   0.0780154   0.0320284   0.08850841
  0.92217676 -0.03396593  0.06113531  0.04331225 -0.19905968 -0.02495698
 -0.08039062 -0.0517569 ]
[18.58734566]
```

```
Trainig R2 Score for SGD Regression is: 0.9698402521434168
Training Mean Squared Error for SGD Regression is: 1.8827256882693681
```

```
Testing R2 score for SGD Regression is: 0.9873761610309992
Testing Mean Squared Error for SGD Regression is: 0.7644726001416293
```



```
In [29]: # Trying Random Forest Regressor
RFR = RandomForestRegressor(criterion = "absolute_error")
RFR.fit(X_train, y_train)

RFR_r2_train = r2_score(y_train, RFR.predict(X_train))
RFR_MSE_train = mean_squared_error(y_train, RFR.predict(X_train))

RFR_r2_test = r2_score(y_test, RFR.predict(X_test))
RFR_MSE_test = mean_squared_error(y_test, RFR.predict(X_test))

print("\n")
print("Trainig R2 Score for Random Forest Regression is: ", RFR_r2_train)
print("Training Mean Squared Error for Random Forest Regression is:", RFR_MSE_train)
print("\n")
print("Testing R2 score for Random Forest Regression is:", RFR_r2_test)
print("Testing Mean Squared Error for Random Forest Regression is:", RFR_MSE_test)
```

```
Trainig R2 Score for Random Forest Regression is: 0.9945615502327636
Training Mean Squared Error for Random Forest Regression is: 0.3394958449197879
```

```
Testing R2 score for Random Forest Regression is: 0.9709897521310465
Testing Mean Squared Error for Random Forest Regression is: 1.7567983617021263
```

```
In [30]: # Trying Decision Tree Regressor
tree = DecisionTreeRegressor(criterion = "poisson")
tree.fit(X_train, y_train)

TREE_r2_train = r2_score(y_train, tree.predict(X_train))
TREE_MSE_train = mean_squared_error(y_train, tree.predict(X_train))

TREE_r2_test = r2_score(y_test, tree.predict(X_test))
TREE_MSE_test = mean_squared_error(y_test, tree.predict(X_test))

print("\n")
print("Trainig R2 Score for Decision Tree Regressor is: ", TREE_r2_train)
print("Training Mean Squared Error for Decision Tree Regressor is:", TREE_MSE_train)
print("\n")
print("Testing R2 score for Decision Tree Regressor is:", TREE_r2_test)
print("Testing Mean Squared Error for Decision Tree Regressor is:", TREE_MSE_test)
```

```
Trainig R2 Score for Decision Tree Regressor is: 0.9999790122860699
Training Mean Squared Error for Decision Tree Regressor is: 0.0013101604278074864
```

```
Testing R2 score for Decision Tree Regressor is: 0.9537105215354749
Testing Mean Squared Error for Decision Tree Regressor is: 2.803191489361701
```

```
In [31]: # Trying Support Vector Regressor
svr = SVR(kernel = 'linear')
svr.fit(X_train, y_train)
# print(svr.coef_)

SVR_r2_train = r2_score(y_train, svr.predict(X_train))
SVR_MSE_train = mean_squared_error(y_train, svr.predict(X_train))

SVR_r2_test = r2_score(y_test, svr.predict(X_test))
SVR_MSE_test = mean_squared_error(y_test, svr.predict(X_test))

print("\n")
print("Trainig R2 Score for SVR is: ", SVR_r2_train)
print("Training Mean Squared Error for SVR is:", SVR_MSE_train)
print("\n")
print("Testing R2 score for SVR is:", SVR_r2_test)
print("Testing Mean Squared Error for SVR is:", SVR_MSE_test)
```

Trainig R2 Score for SVR is: 0.9652731904107476
 Training Mean Squared Error for SVR is: 2.1678250360787916

Testing R2 score for SVR is: 0.993125846331351
 Testing Mean Squared Error for SVR is: 0.4162839958391145

```
In [33]: # Trying Gradient Boosting Regressor
GBR = GradientBoostingRegressor(loss = 'absolute_error')
GBR.fit(X_train, y_train)

GBR_r2_train = r2_score(y_train, GBR.predict(X_train))
GBR_MSE_train = mean_squared_error(y_train, GBR.predict(X_train))

GBR_r2_test = r2_score(y_test, GBR.predict(X_test))
GBR_MSE_test = mean_squared_error(y_test, GBR.predict(X_test))

print("\n")
print("Trainig R2 Score for Gradient Boosting Regressor is: ", GBR_r2_train)
print("Training Mean Squared Error for Gradient Boosting Regressor is:", GBR_MSE_train)
print("\n")
print("Testing R2 score for Gradient Boosting Resgressor is:", GBR_r2_test)
print("Testing Mean Squared Error for Gradient Boosting Regressor is:", GBR_MSE_test)
```

Trainig R2 Score for Gradient Boosting Regressor is: 0.9896248896535275
 Training Mean Squared Error for Gradient Boosting Regressor is: 0.6476674427392292

Testing R2 score for Gradient Boosting Resgressor is: 0.9906885254597065
 Testing Mean Squared Error for Gradient Boosting Regressor is: 0.5638829179024464

MODEL FINALIZED FOR DEPLOYMENT: LINEAR REGRESSION

```
# Trying Linear Regression
LR = LinearRegression(fit_intercept = True)
LR.fit(X_train, y_train)
print(LR.coef_)
print(LR.intercept_)
LR_r2_train = r2_score(y_train, LR.predict(X_train))
LR_MSE_train = mean_squared_error(y_train, LR.predict(X_train))

LR_r2_test = r2_score(y_test, LR.predict(X_test))
LR_MSE_test = mean_squared_error(y_test, LR.predict(X_test))

print("\n")
print("Trainig R2 Score for Linear Regression is: ", LR_r2_train)
print("Training Mean Squared Error for Linear Regression is:", LR_MSE_train)
print("\n")
print("Testing R2 score for Linear Regression is:", LR_r2_test)
print("Testing Mean Squared Error for Linear Regression is:", LR_MSE_test)

[-7.22056387  0.13407058  0.36446425  0.02192522  0.01339398  0.20806555
  0.26668563  0.08786141  0.0225926   0.05486768 -0.30583712 -0.1369371
 -0.08319438  0.0398096 ]
18.593175680803586
```

```
Trainig R2 Score for Linear Regression is:  0.9705113043596266
Training Mean Squared Error for Linear Regression is: 1.8408351773925504
```

```
Testing R2 score for Linear Regression is: 0.9902366135857361
Testing Mean Squared Error for Linear Regression is: 0.5912497312923649
```

```
import pickle
pickle.dump(LR,open("model.pkl","wb"))
```

MODEL DEPLOYMENT USING FLASK:

➤ app.py file

```
import numpy as np
from flask import Flask, request,render_template
import pickle

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')
```



```
@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    float_features = [float(x) for x in
request.form.values()]
    final_features = [np.array(float_features)]
    prediction = model.predict(final_features)

    output = round(prediction[0], 2)

    return render_template('index.html',
prediction_text='YOUR BODY FAT IS : {}'.format(output))

if __name__ == "__main__":
    app.run(debug=True)
```

➤ index.html file

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
    <meta charset="UTF-8">
    <title>ML API</title>
    <link
href='https://fonts.googleapis.com/css?family=Pacifico'
rel='stylesheet' type='text/css'>
    <link
href='https://fonts.googleapis.com/css?family=Arimo'
rel='stylesheet' type='text/css'>
    <link
href='https://fonts.googleapis.com/css?family=Hind:300'
rel='stylesheet' type='text/css'>
    <link
href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">

</head>

<body>
    <div class="login">
        <h1>ACCURATE BODY FAT PREDICTION</h1>
```

```

        <!-- Main Input For Receiving Query to our ML -->
        <form action="{{ url_for('predict')}}"method="post">
            <input type="text" name="Density"
placeholder="Density" required="required" />
            <input type="text" name="Age" placeholder="Age"
required="required" />
            <input type="text" name="Weight"
placeholder="Weight" required="required" />
            <input type="text" name="Height"
placeholder="Height" required="required" />
            <input type="text" name="Neck" placeholder="Neck"
required="required" />
            <input type="text" name="Chest"
placeholder="Chest" required="required" />
            <input type="text" name="Abdomen"
placeholder="Abdomen" required="required" />
            <input type="text" name="Hip" placeholder="Hip"
required="required" />
            <input type="text" name="Thigh"
placeholder="Thigh" required="required" />
            <input type="text" name="Knee"
placeholder="Knee" required="required" />
            <input type="text" name="Ankle"
placeholder="Ankle" required="required" />
            <input type="text" name="Biceps"
placeholder="Biceps" required="required" />
            <input type="text" name="Forearm"
placeholder="Forearm" required="required" />
            <input type="text" name="Wrist"
placeholder="Wrist" required="required" />

            <button type="submit" class="btn btn-primary btn-
block btn-large">Predict</button>
        </form>

        <br>
        <br>
        {{ prediction_text }}

    </div>

</body>
</html>

```

Spyder (Python 3.10)

File Edit Search Source Run Debug Consoles Projects Tools View Help

E:\ADS project\app.py

```

1 import numpy as np
2 from flask import Flask, request, render_template
3 import pickle
4
5 app = Flask(__name__)
6 model = pickle.load(open('model.pkl', 'rb'))
7
8 @app.route('/')
9 def home():
10     return render_template('index.html')
11
12 @app.route('/predict', methods=['POST'])
13 def predict():
14     ...
15     For rendering results on HTML GUI
16     ...
17     float_features = [float(x) for x in request.form.values()]
18     final_features = [np.array(float_features)]
19     prediction = model.predict(final_features)
20
21     output = round(prediction[0], 2)
22
23     return render_template('index.html', prediction_text='YOUR BODY FAT IS : {}'.format(output))
24
25 if __name__ == '__main__':
26     app.run(debug=True)
27 
```

E:\ADS project

Name	Date Modified
static	27-06-2023 07:51 PM
templates	27-06-2023 08:16 PM
app.py	27-06-2023 08:59 PM
model.ipynb	27-06-2023 08:01 PM
model.pkl	27-06-2023 07:59 PM

Help | Variable Explorer | Plots | Files

Console 1/1 X

IPython 8.10.0 -- An enhanced Interactive Python.

```

In [1]: runfile('E:/ADS project/app.py', wdir='E:/ADS project')
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production
deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)

```

Python Console | History

conda: base (Python 3.10.9) | Completions: conda(base) | LSP: Python | Line 21, Col 37 | ASCII | LF | RW | Mem 85%

Spyder (Python 3.10)

File Edit Search Source Run Debug Consoles Projects Tools View Help

E:\ADS project\templates\index.html

```

1 <!DOCTYPE html>
2 <html>
3 <!-- From https://codepen.io/frtyler/pen/EGdte -->
4 <head>
5 <meta charset="UTF-8">
6 <title>ML API</title>
7 <link href="https://fonts.googleapis.com/css?family=Pacifico" rel="stylesheet" type="text/css">
8 <link href="https://fonts.googleapis.com/css?family=Varela+Round" rel="stylesheet" type="text/css">
9 <link href="https://fonts.googleapis.com/css?family=Varela+Round:200" rel="stylesheet" type="text/css">
10 <link href="https://fonts.googleapis.com/css?family=Open+Sans:Condensed:300" rel="stylesheet" type="text/css">
11 <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
12
13 </head>
14
15 <body>
16 <div class="login">
17 <h1>ACCURATE BODY FAT PREDICTION</h1>
18
19 <!-- Main Input for Receiving Query to our ML -->
20 <form action="{{ url_for('predict') }}" method="post">
21 <input type="text" name="Density" placeholder="Density" required="required" />
22 <input type="text" name="Age" placeholder="Age" required="required" />
23 <input type="text" name="Weight" placeholder="Weight" required="required" />
24 <input type="text" name="Height" placeholder="Height" required="required" />
25 <input type="text" name="Neck" placeholder="Neck" required="required" />
26 <input type="text" name="Chest" placeholder="Chest" required="required" />
27 <input type="text" name="Abdomen" placeholder="Abdomen" required="required" />
28 <input type="text" name="Hip" placeholder="Hip" required="required" />
29 <input type="text" name="Thigh" placeholder="Thigh" required="required" />
30 <input type="text" name="Knee" placeholder="Knee" required="required" />
31 <input type="text" name="Ankle" placeholder="Ankle" required="required" />
32 <input type="text" name="Biceps" placeholder="Biceps" required="required" />
33 <input type="text" name="Forearm" placeholder="Forearm" required="required" />
34 <input type="text" name="Wrist" placeholder="Wrist" required="required" />
35
36 <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
37 </form>
38
39 <br>
40 <br>

```

E:\ADS project

Name	Date Modified
static	27-06-2023 07:51 PM
templates	27-06-2023 08:16 PM
app.py	27-06-2023 08:59 PM
model.ipynb	27-06-2023 08:01 PM
model.pkl	27-06-2023 07:59 PM

Help | Variable Explorer | Plots | Files

Console 1/1 X

IPython 8.10.0 -- An enhanced Interactive Python.

```

In [1]: runfile('E:/ADS project/app.py', wdir='E:/ADS project')
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production
deployment. Use a production WSGI server instead.
* Running on https://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)

```

Python Console | History

conda: base (Python 3.10.9) | Completions: conda(base) | Line 19, Col 55 | ASCII | LF | RW | Mem 84%

INPUT WEBPAGE:

A screenshot of a web browser showing the 'INPUT WEBPAGE' for 'ACCURATE BODY FAT PREDICTION'. The browser's address bar shows '127.0.0.1:5000'. The page has a dark blue gradient background. The title 'ACCURATE BODY FAT PREDICTION' is centered at the top. Below the title is a vertical stack of 15 input fields, each containing a numerical value. At the bottom of the stack is a blue button labeled 'Predict'.

Input Field	Value
1	1.0794
2	25
3	176.00
4	15.86
5	57.8
6	58.5
7	58.5
8	57.1
9	50.0
10	59.4
11	55.2
12	50.5
13	59.4
14	50.5
15	50.5

Predict

OUTPUT WEBPAGE:

A screenshot of a web browser showing the 'OUTPUT WEBPAGE' for 'ACCURATE BODY FAT PREDICTION'. The browser's address bar shows '127.0.0.1:5000/predict'. The page has a dark blue gradient background. The title 'ACCURATE BODY FAT PREDICTION' is centered at the top. Below the title is a vertical stack of 15 input fields, each containing a numerical value. At the bottom of the stack is a blue button labeled 'Predict'. Below the button, the text 'YOUR BODY FAT IS : 12.59' is displayed.

Input Field	Value
1	1.0794
2	25
3	176.00
4	15.86
5	57.8
6	58.5
7	58.5
8	57.1
9	50.0
10	59.4
11	55.2
12	50.5
13	59.4
14	50.5
15	50.5

Predict

YOUR BODY FAT IS : 12.59