

Mathematica for Research Project Adipose Determination using Machine Learning Algorithms Raj Shekhar - 18200277

Chapter 1 - Introduction

Adipose tissue, is an anatomical term for loose connective tissue that compose of adipocytes. Its main role is to store energy in the form of fat, although it cushions and insulates the body which is necessary upto some extent in the body but excess of it may lead to unwanted disease. Far from being hormonally inert, it has been recognized as a major endocrine organ as it produces hormones and thus it is an important part of body composition. In recent years diseases like cancer, heart disease, and diabetes have been reported to be the leading causes of death in most countries of the world. One of the most common risk factors for those diseases is the obesity and excess of adipose often leads to this obesity.

In this project we will explore and analyse the data using data analysis methodologies of EDA(exploratory data analysis), Data Visualization and Analysis (using different machine learning techniques). We will mainly follow below steps:

- 1) **Pre - Processing** : In this step we will work on data cleansing, dependence / independence of variable check, detecting/removing of outlier and many other processes.
- 2) **Data Visualization** : In this step we will use Bar plot, Box plot, Histogram, ListPlot and other plotting methods for the pictorial representation of the data.
- 3) **Analysis** : In this step we will use multiple **Machine Learning** techniques and making comparisons among them to predict a sustainable model for the input dataset.

A pictorial representation of how we will analyze and process the data :

```
In[ ]:= PieChart[{33.3, 33.3, 33.3}, SectorOrigin -> {Automatic, 1},
  ChartLabels -> {"Pre-Processing", "Visualization", "Analysis"},
  ChartElementFunction -> "GlassSector", ChartStyle -> "Pastel"]
```

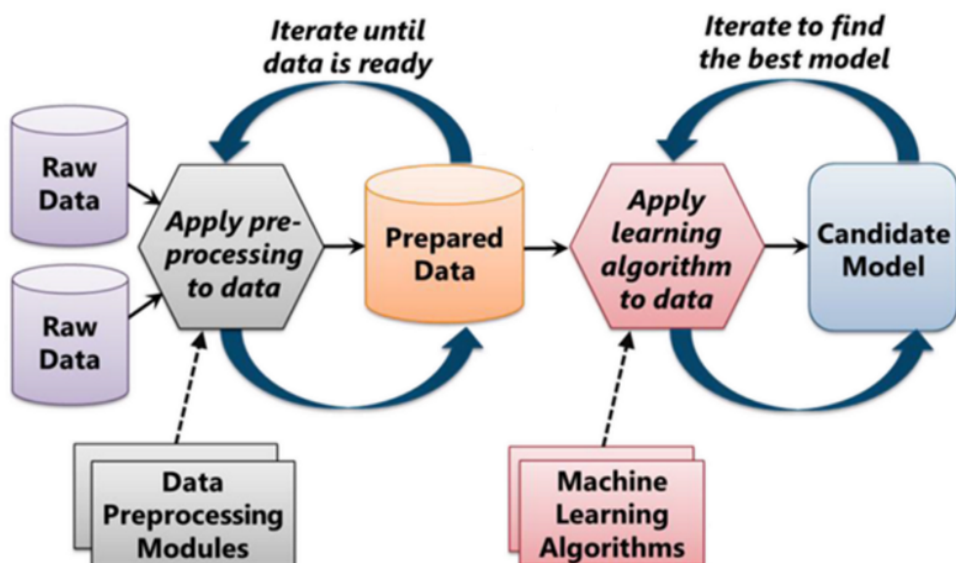
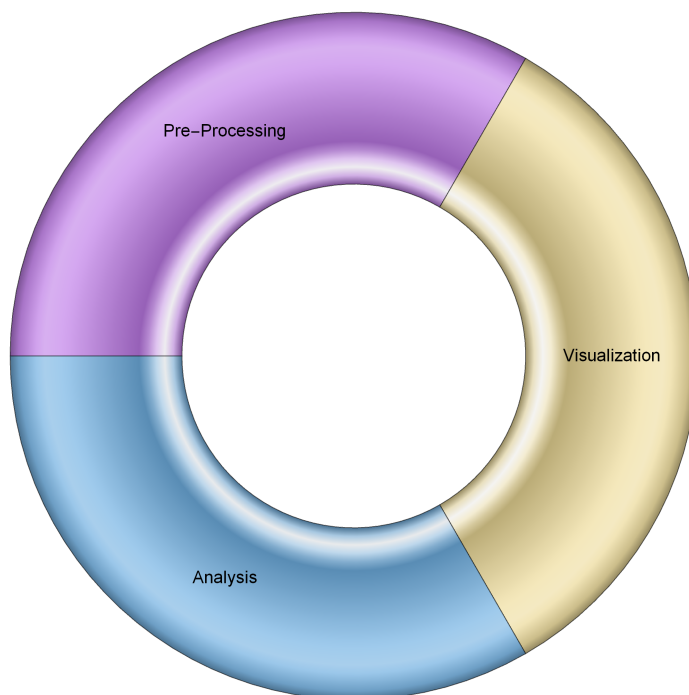


Figure 1 : Pie Chart

Figure 2: Flow Chart (reference taken from en.wikipedia.org)

1.1 Project Description

In this project we will be using data visualization and multiple machine learning techniques for predicting adipose in the human body. Our data set consists of different body parts measurement, which may contribute to the body adipose.

As we can see in the below body part figures that each body part comprises of different layers of tissues, muscles, nerves and many other components and hence it is quite difficult to determine the actual composition of adipose in the entire body in normal circumstances. In medical science it's quite expensive task to get the percentage of adipose in human body.

Although there are several ways to measure it, the accurate methods are often associated with a lot of hassle or have very high costs. Traditional approaches for predicting it may use certain body measurements or explanatory variables. Diverging from existing approaches, this study proposes new intelligent hybrid approaches to obtain fewer explanatory variables, and the proposed forecasting models shown in this project are able to effectively predict it up to a very large extent.

1.2 About Dataset

The data set used has in total 241 rows and 15 columns that comprises of different body parts quantification having data of people between the age range of 19 yrs to 65 yrs.

(* The data set being used in this project is taken from www.kaggle.com *)

Description of different variables:

The variables used in the dataset are listed below:

- 1) Age : The current age of an individual when measurement was taken. It lies between 19 to 65 yrs.
- 2) Weight : The body weight of an individual.
- 3) Height : The height with measurement in inches.
- 4) Thigh : The thickness of individual thigh circumference in centimetres.
- 5) Knee : The measurement of individual knee circumference in centimetres.
- 6) Ankle : The measurement of individual ankle circumference in centimetres.
- 7) Biceps : The measurement of individual biceps circumference in centimetres.
- 8) Forearm : The measurement of individual forearm circumference in centimetres.
- 8) Wrist : The measurement of individual wrist circumference in centimetres.
- 9) Neck : The measurement of individual neck circumference in centimetres.

- 10) Chest : The measurement of individual chest circumference in centimetres.
- 11) Abdomen : The measurement of individual abdomen circumference in centimetres.
- 12) Hip : The measurement of individual hip circumference in centimetres.
- 13) Density : The measured density of an individual .
- 14) Adipose : The interpreted adipose in an individual.

Importing the data:

```
In[ ]:= Clear["Global`*"]

(* Selecting the data set stored in the file project_data.csv *)

path = SystemDialogInput["FileOpen"];
data = Import[path, {"CSV", "Dataset"}, HeaderLines -> 1]
```

Variable Names of Data Set


```
In[ ]:= columnName = First@Keys@data
```

Age	Weight	Height	Thigh	Knee
Ankle	Biceps	Forearm	Wrist	Neck
Chest	Abdomen	Hip	Density	Adipose

We can visualize the columns present in the data set in the below pop up list :

```
In[ ]:= colname = {"Age", "Weight", "Height", "Thigh", "Knee", "Ankle", "Biceps",
  "Forearm", "Wrist", "Neck", "Chest", "Abdomen", "Hip", "Density", "Adipose"};

In[ ]:= PopupView[colname]
```

Out[]:= 

Dataset Dimension

```
In[ ]:= Dimensions[data]

Out[ ]:= {241, 15}
```

Chapter 2 - Processing

```
Mouseover[Please Hover, START Processing]
(* A small mouse hover animation to begin the processing of data *)
```

Out[]:= Hover Please

In this section we will perform different steps to prepare data set for the next step of model building.

2.1 Pre – Processing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real world data is often inconsistent, incomplete, incorrect and may lack in certain behaviors and is likely to contain errors. Data preprocessing is an efficient proven method of resolving such issues and transforming a raw data into an understandable format so that it can be used by machine learning techniques for the prediction analysis.

Missing Value:

The concept of missing values is important to understand in order to successfully manage data. If the missing values are not handled properly then we may end up drawing an inappropriate and inaccurate inference about the data. Due to improper handling, the result obtained will actually differ from ones where the missing values are present. Hence it's quite important to identify and address missing values before proceeding with any sort of analysis.

```
In[ ]:= Table[data[Count[_Missing], i], {i, 1, 15, 1}]
Out[ ]:= {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

From the above output we can say that our data is free from any missing value.

2.2 Variable Selection

It is one of the core concepts in machine learning which hugely impacts the performance of a model. The data features that you use to train your machine learning models have a huge influence on the performance you can achieve. Irrelevant or partially relevant features can adversely impact model performance. We are using correlation technique here.

Correlation is a technique for investigating the relationship between two quantitative continuous variables and is a measure of the strength of the association between them.

```
In[ ]:= predictorvariable = {"Age", "Weight", "Height", "Thigh", "Knee", "Ankle",
    "Biceps", "Forearm", "Wrist", "Neck", "Chest", "Abdomen", "Hip", "Density"};

(* Using Correlation function to find
    correlation among all the variables and storing it *)

In[ ]:= corr = Correlation[Normal[Values[data]]];

(* Correlation Plot of all the variables *)
```

```

In[ ]:= Column[{GraphicsRow[colname, ImageSize → 800, Frame → All],
  Row[{GraphicsColumn[colname, ImageSize → {100, 800}, Frame → All],
    Overlay[{ArrayPlot[corr, ColorFunction → (ColorData["TemperatureMap"][(1 + #)/2] &),
      PlotRangePadding → 0, ImageSize → {800, 800}, Frame → None, Mesh → True,
      ColorFunctionScaling → False], GraphicsGrid[Map[NumberForm[#, 2] &, corr, {2}],
      Spacings → 0, ImageSize → {800, 800}]}]}], Alignment → Right]

```

Age	Weight	Height	Thigh	Knee	Ankle	Biceps	Forearm	Wrist	Neck	Chest	Abdomen
-----	--------	--------	-------	------	-------	--------	---------	-------	------	-------	---------

Age
Weight
Height
Thigh
Knee
Ankle
Biceps
Forearm
Wrist
Neck
Chest
Abdomen
Hip

Out[]:=

Density
Adipose

1.	0.013	-0.14	-0.12	0.027	-0.17	0.021	-0.021	0.14	0.084	0.16	0.21
0.013	1.	0.31	0.88	0.86	0.63	0.8	0.63	0.76	0.82	0.9	0.9
-0.14	0.31	1.	0.14	0.29	0.31	0.2	0.22	0.35	0.26	0.14	0.1
-0.12	0.88	0.14	1.	0.82	0.55	0.76	0.55	0.61	0.7	0.75	0.8
0.027	0.86	0.29	0.82	1.	0.63	0.68	0.54	0.68	0.66	0.72	0.75
-0.17	0.63	0.31	0.55	0.63	1.	0.49	0.42	0.6	0.5	0.49	0.46
0.021	0.8	0.2	0.76	0.68	0.49	1.	0.67	0.67	0.73	0.74	0.71
-0.021	0.63	0.22	0.55	0.54	0.42	0.67	1.	0.62	0.63	0.58	0.52
0.14	0.76	0.35	0.61	0.68	0.6	0.67	0.62	1.	0.75	0.67	0.62
0.084	0.82	0.26	0.7	0.66	0.5	0.73	0.63	0.75	1.	0.78	0.74
0.16	0.9	0.14	0.75	0.72	0.49	0.74	0.58	0.67	0.78	1.	0.92
0.21	0.9	0.1	0.8	0.75	0.46	0.71	0.52	0.62	0.74	0.92	1.
-0.021	0.94	0.17	0.9	0.83	0.57	0.73	0.53	0.65	0.73	0.84	0.89
-0.31	-0.46	0.052	-0.44	-0.42	-0.096	-0.4	-0.33	-0.25	-0.32	-0.5	-0.6
0.27	0.62	-0.073	0.6	0.52	0.23	0.51	0.38	0.32	0.47	0.7	0.81



From the above correlation plot we can define relation between each of the predictor variable and dependent variable.

- a) There is very strong positive correlation of Adipose with Abdomen of 0.81
- b) There is very moderate positive correlation of Adipose with Hip of 0.64, Weight of 0.62 , Knee of 0.52 and Biceps of 0.51.
- c) Rest of the variables are weakly correlated with the dependent variables.
- d) The variable Height is weakly correlated with the Adipose with lowest collinearity among all of -0.073. Hence it is fine to remove it from the model but for our testing we can consider it as it will not impact the analysis.

Chapter 3 - Data Visualization

There are several techniques of data visualization that uses an array of static and interactive visuals within a specific context to help understand and make sense of large amounts of data in an effective and efficient way.

3.1 Word Cloud:

A Word cloud showing the columns present in the data set :

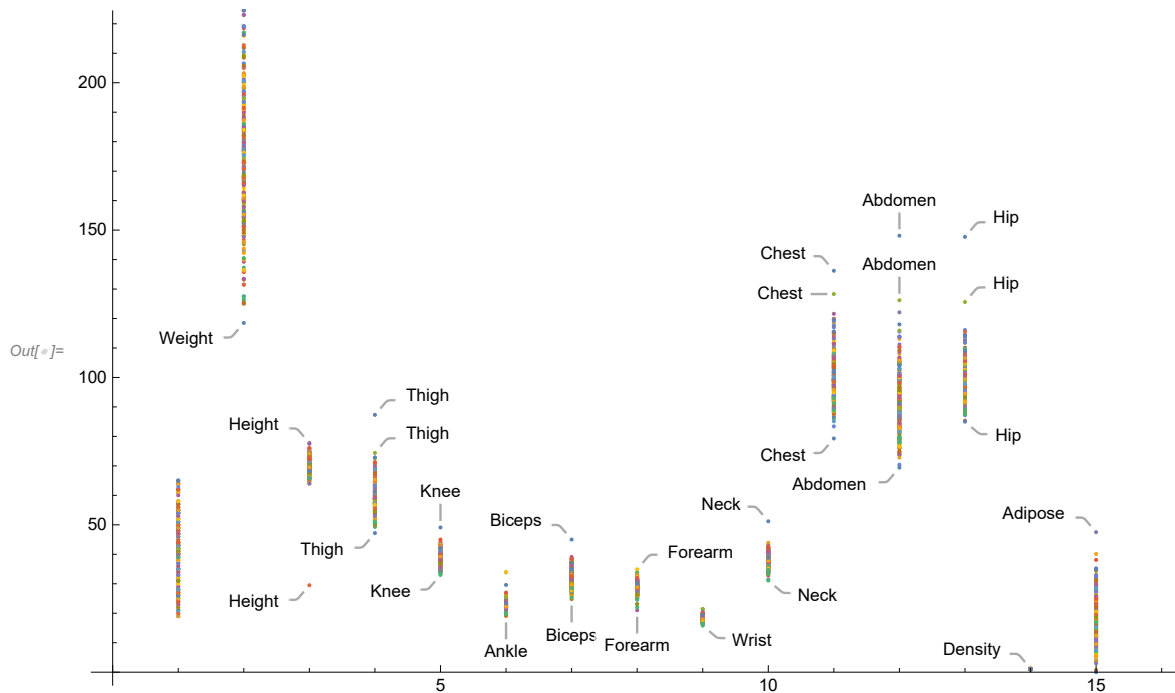
```
In[ ]:= WordCloud[colname, Disk[], ImageSize -> 300]
```

Out[]:=



3.2 List Plot:

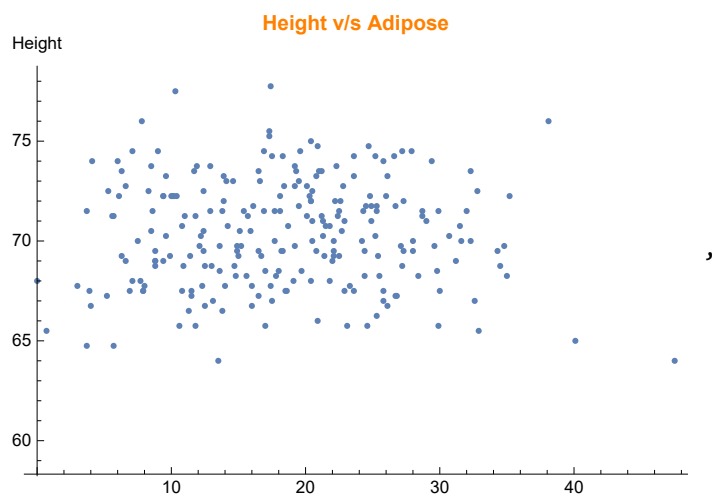
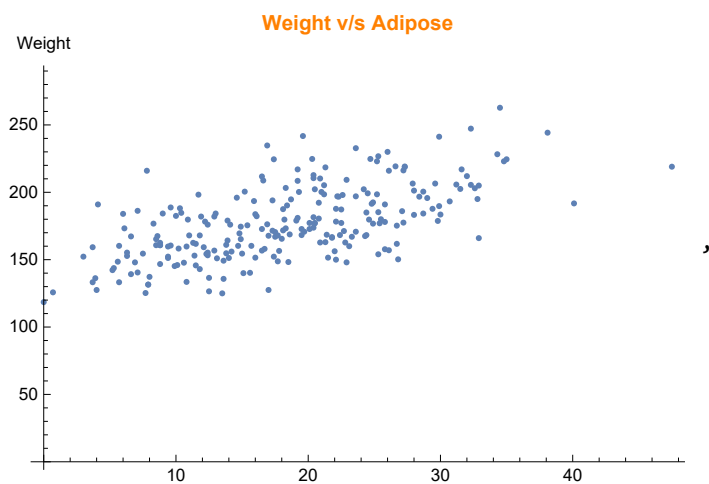
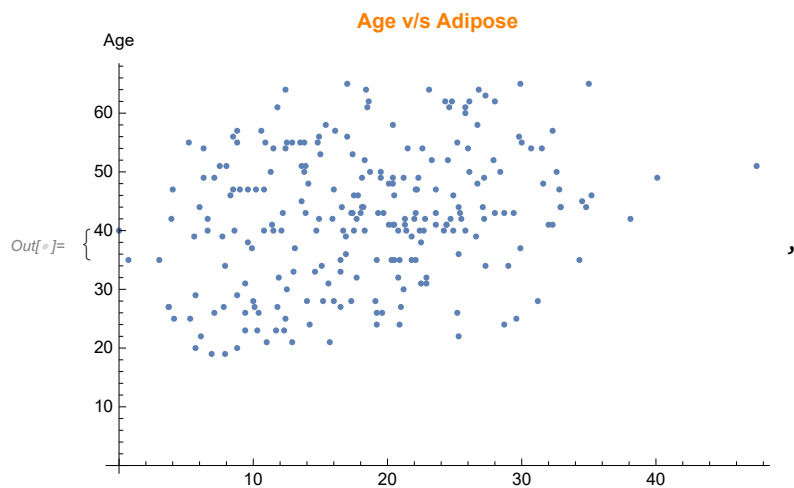
```
In[ ]:= ListPlot[data, ImageSize → Large]
```

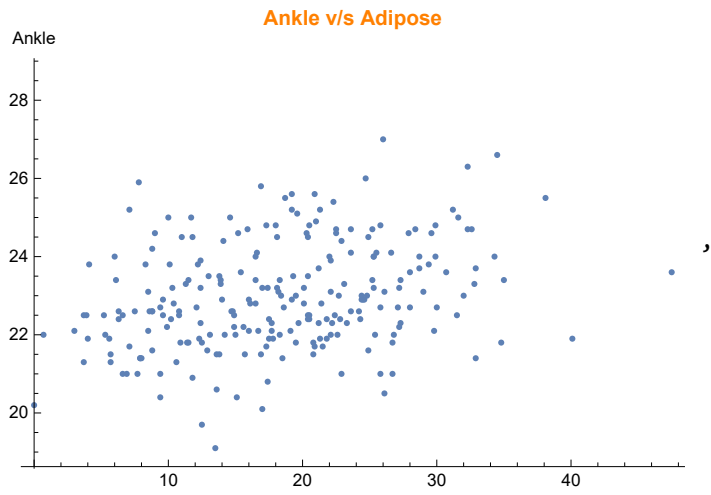
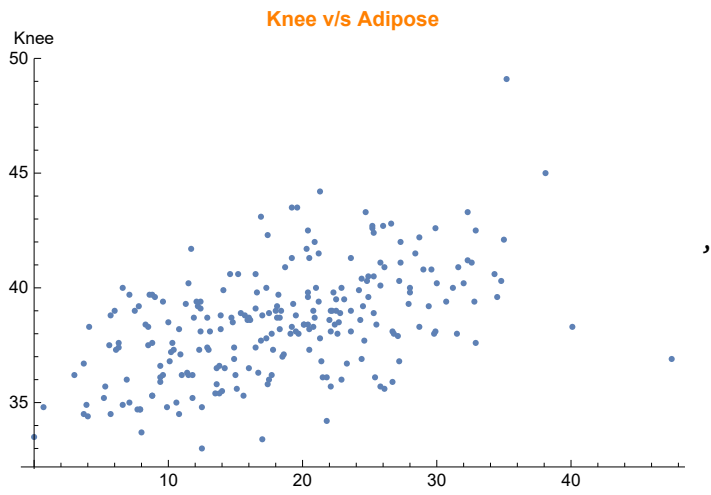
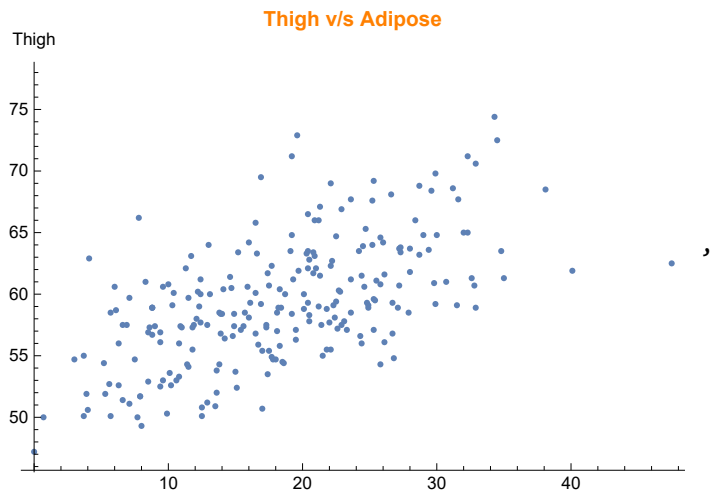


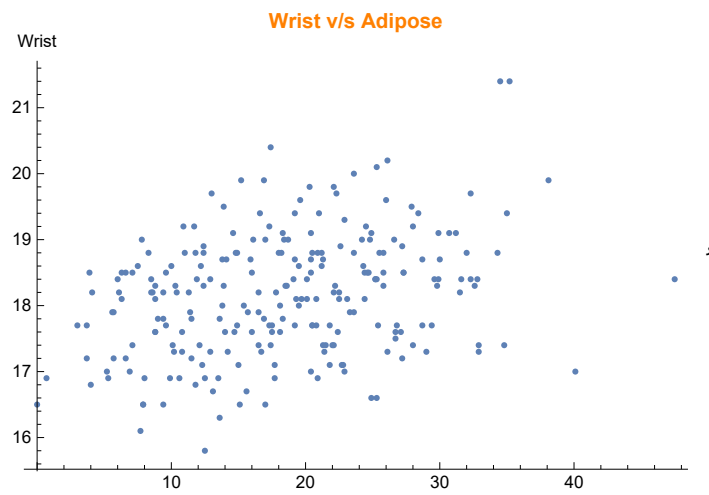
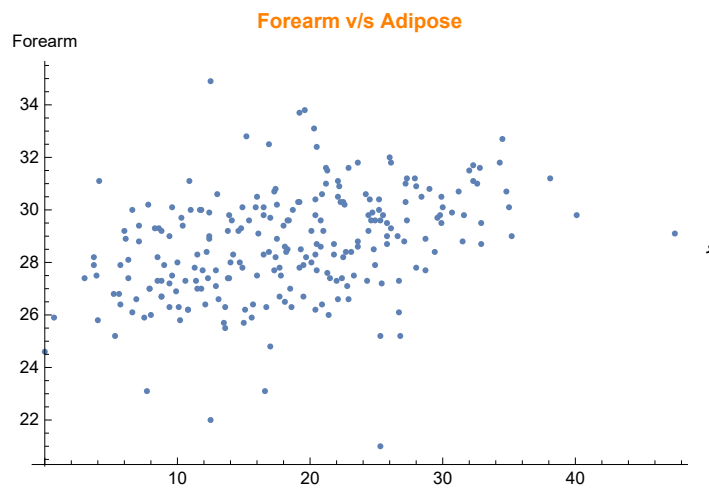
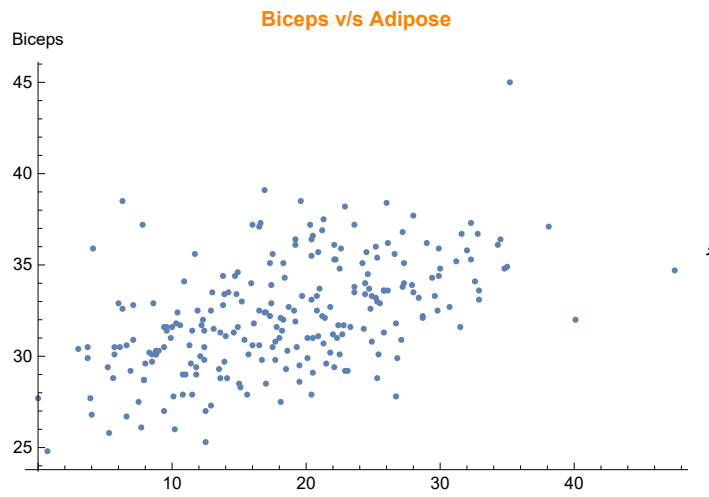
From the above list plot we can infer that Weight is considerably high among all variables which is obvious and rest all variables are almost grouped closely with very few having exception of outlier points that may be due some individual measurements.

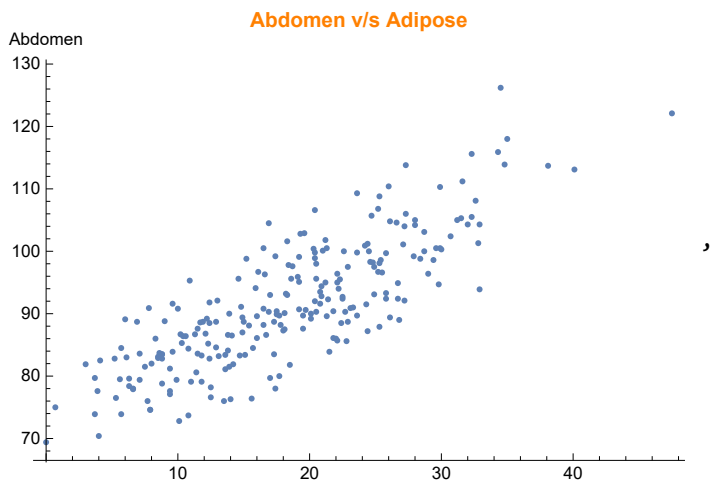
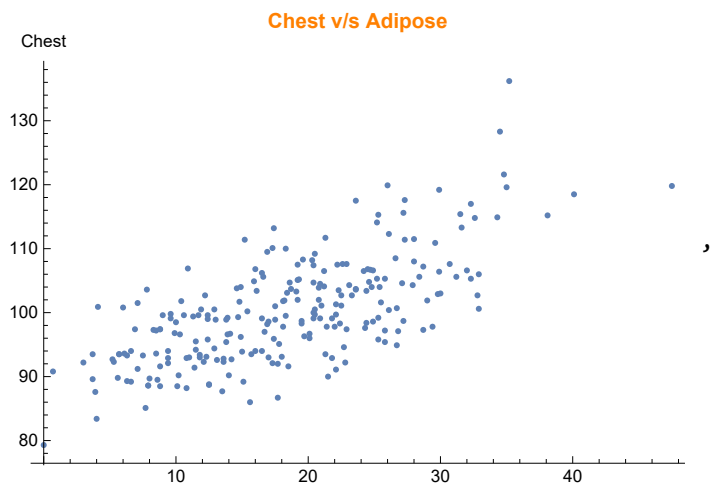
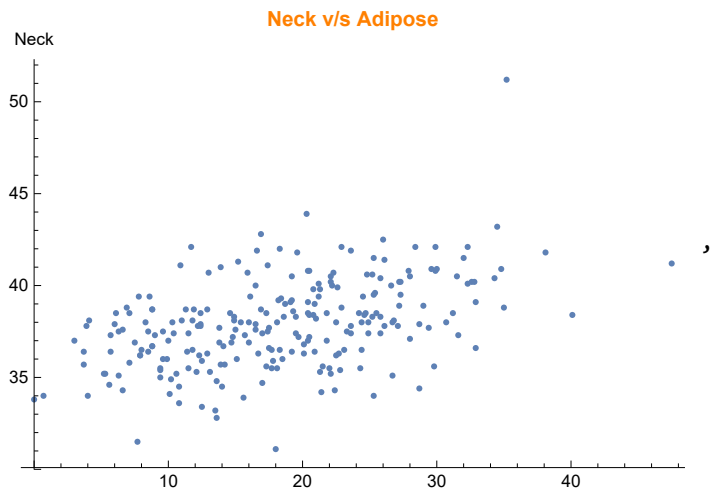
3.3 Scatter Plots:

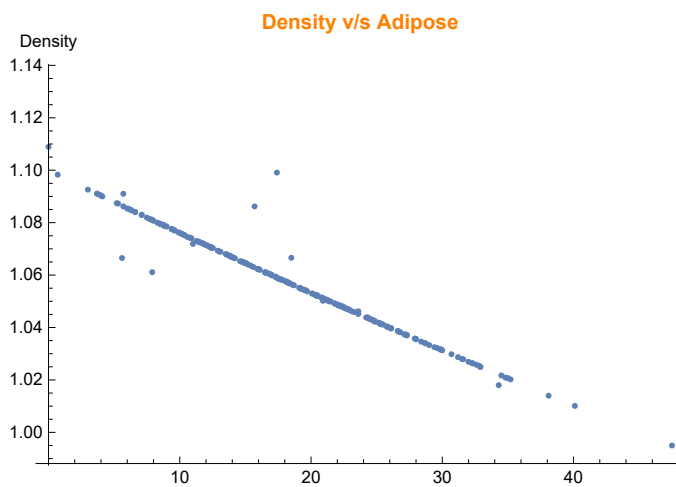
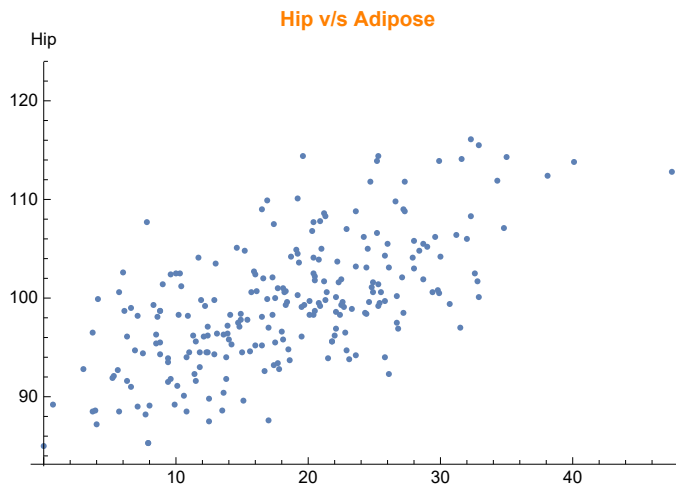
```
In[ ]:= Table[Show[ListPlot[Transpose[{data[[All, 15]] // Normal, data[[All, i]] // Normal}],
  AxesLabel → {"", colname[[i]]},
  PlotLabel → Style[StringForm["` v/s Adipose", colname[[i]]], Orange, Bold],
  ImageSize → Medium]], {i, 1, 14}]
```







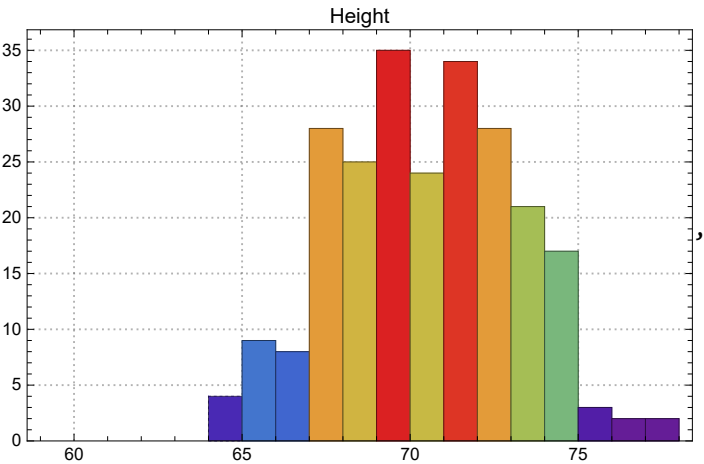
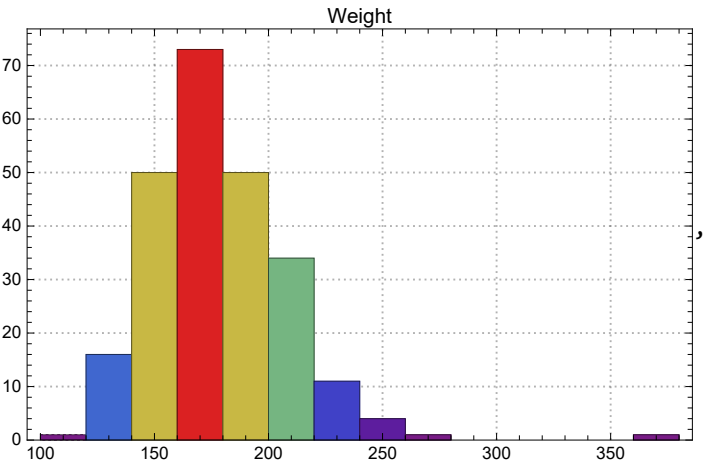
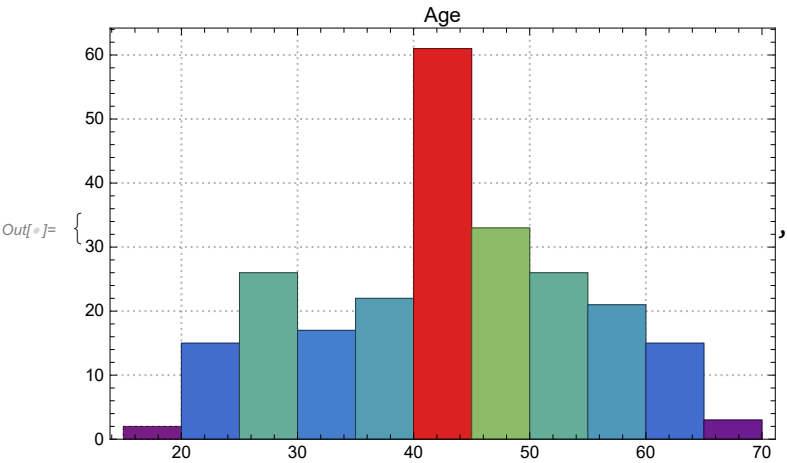


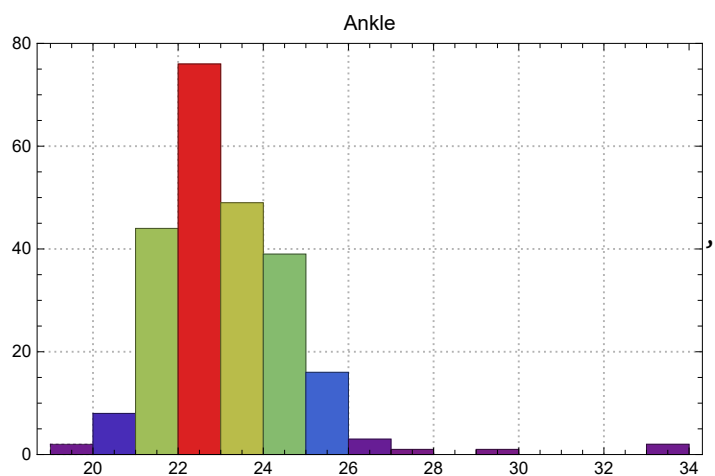
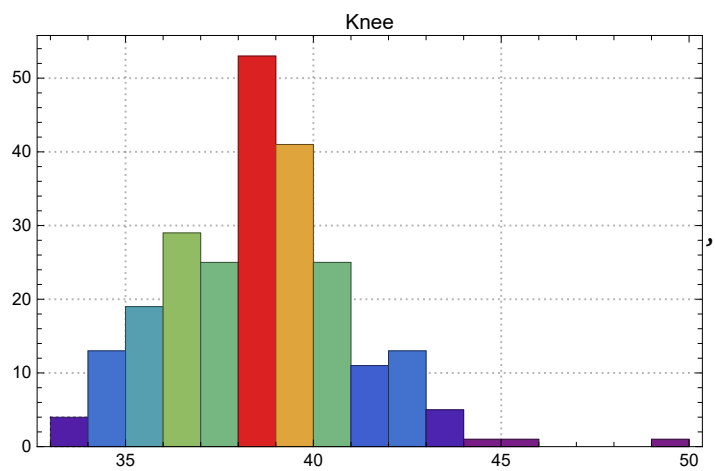
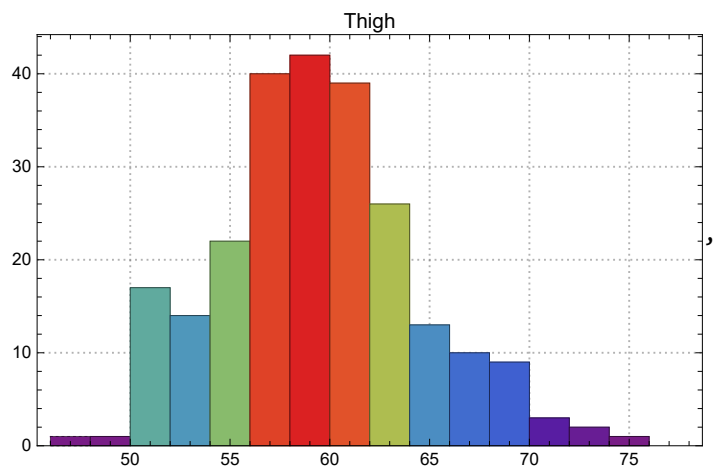


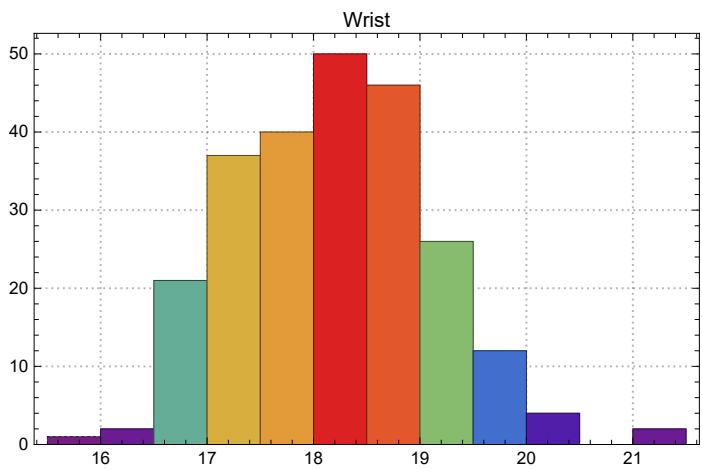
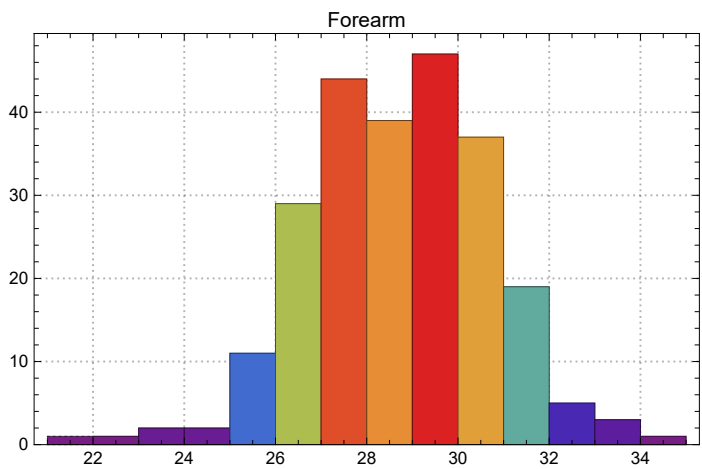
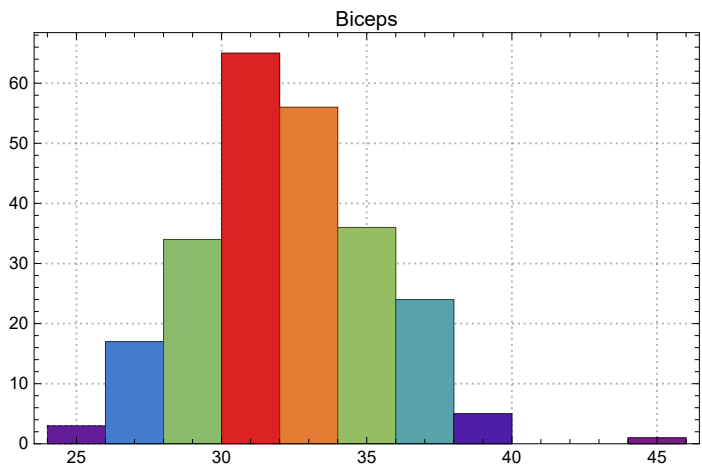
From the above scatter plot we can infer that one plot that is standing out is Density Vs Adipose, as it has strong negative linear relationship which means as Adipose increases Density tends to decrease and have very few outliers that can surely be neglected here.

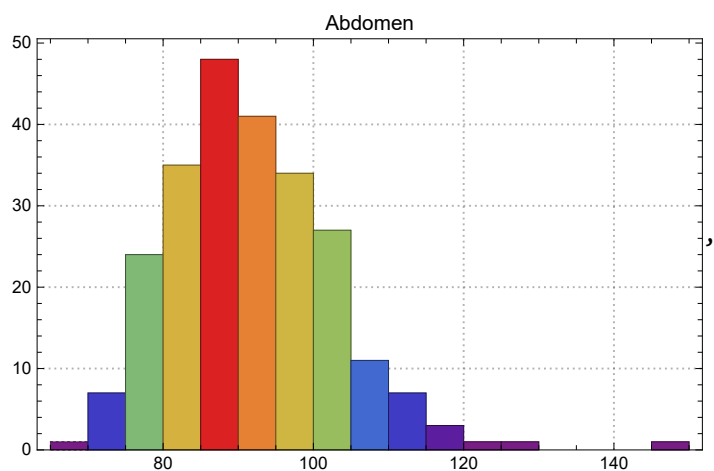
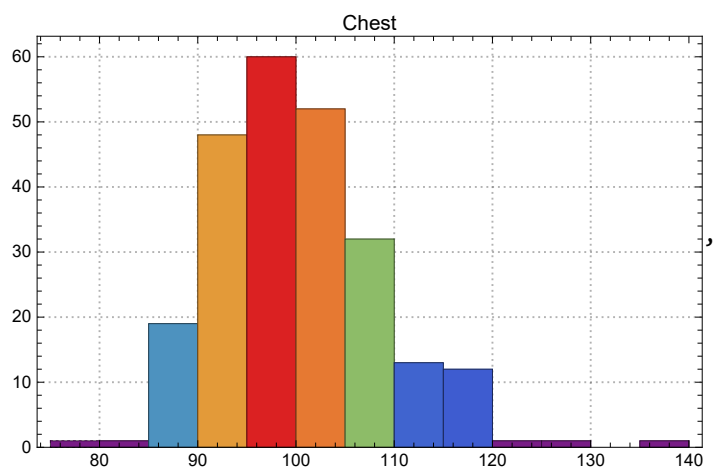
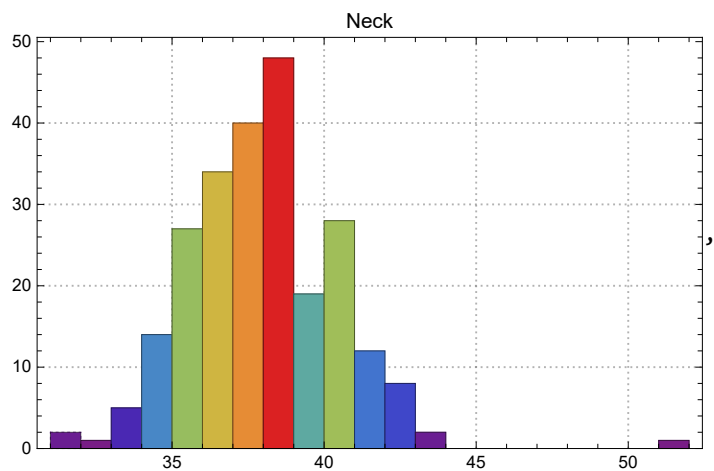
3.4 Histogram :

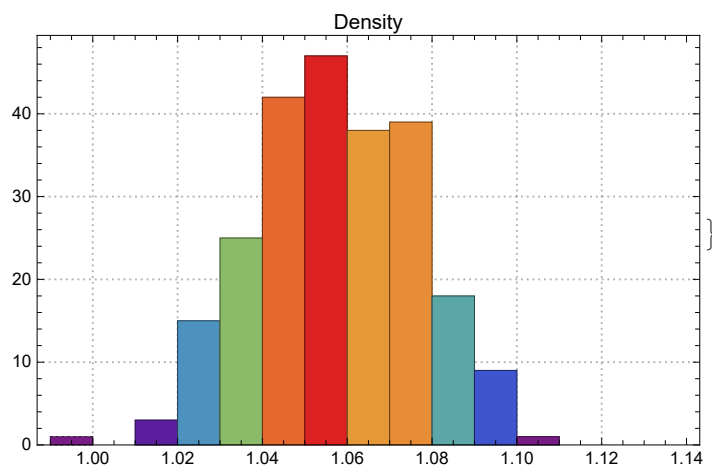
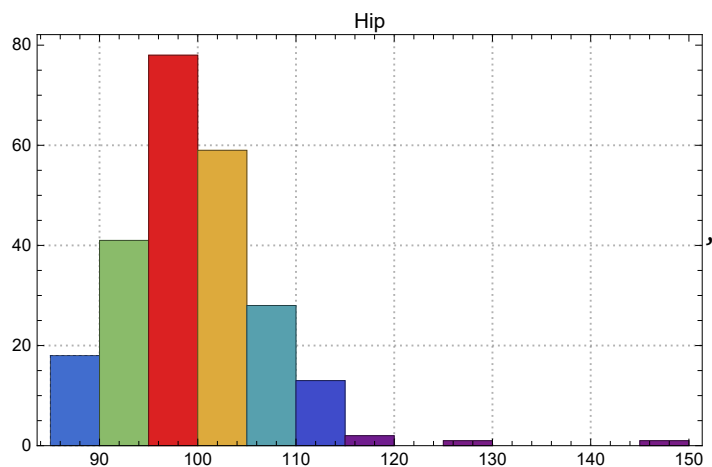
```
In[ ]:= Table[ Histogram[data[All, i] // Normal,
  PlotTheme -> "Detailed", ColorFunction -> "Rainbow",
  PlotLabel -> predictorvariable[[i]], ImageSize -> Medium], {i, 1, 14, 1}]
```











Above plot shows the range of data per variable in each of their categories. As we have all numerical variables thus can't be categorized explicitly.

Scaling:

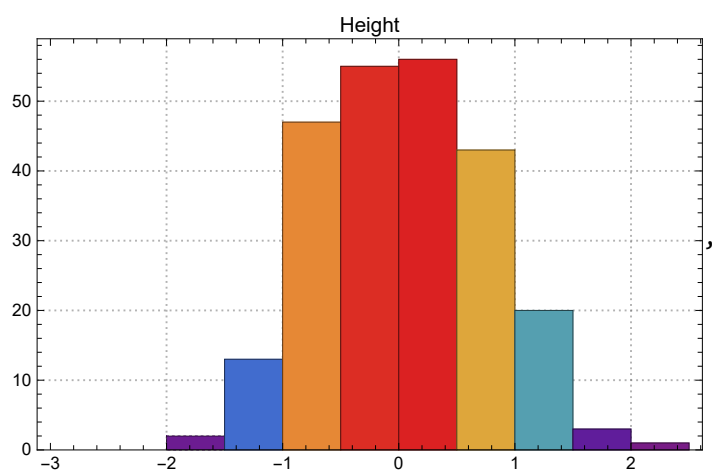
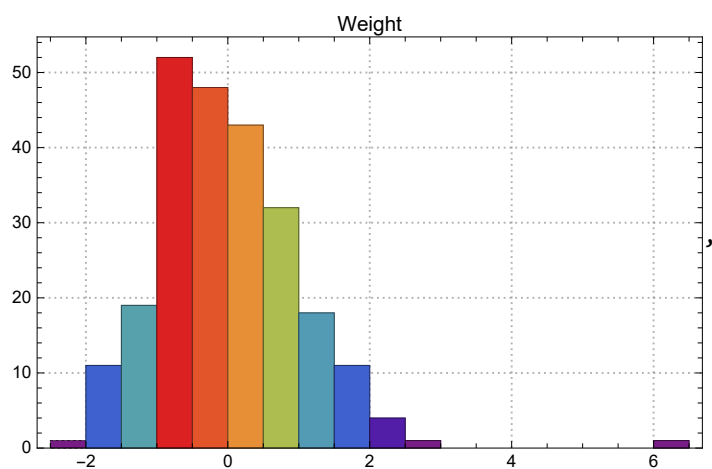
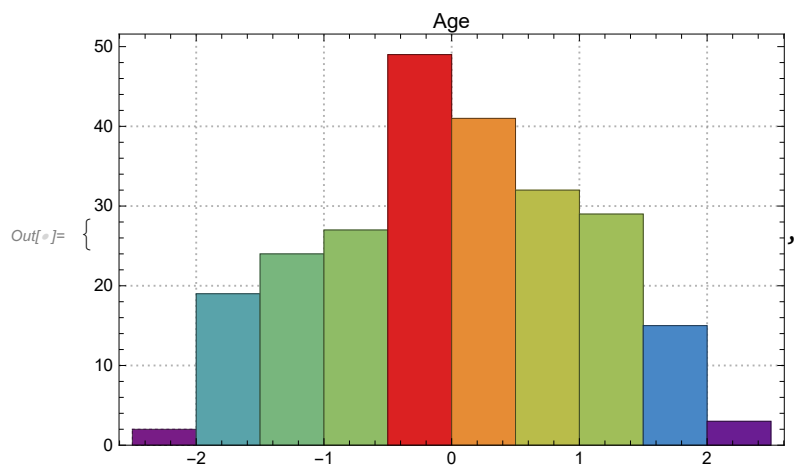
It is the method that standardizes the variables. By standardizing the data set it shifts and rescales the elements of the list to have a zero mean and unit sample variance.

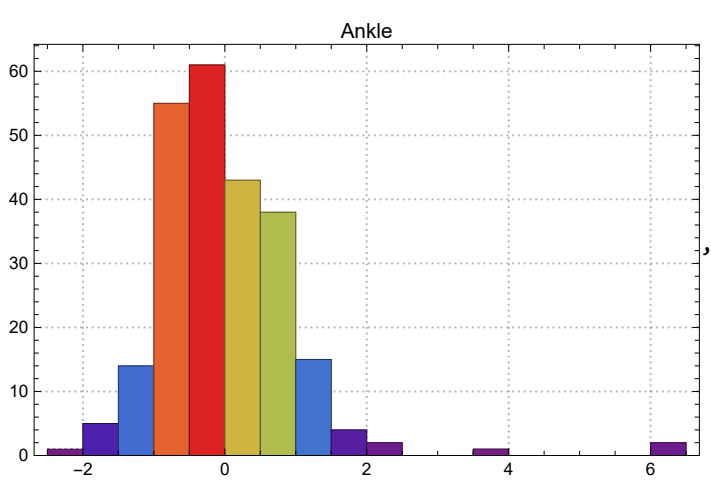
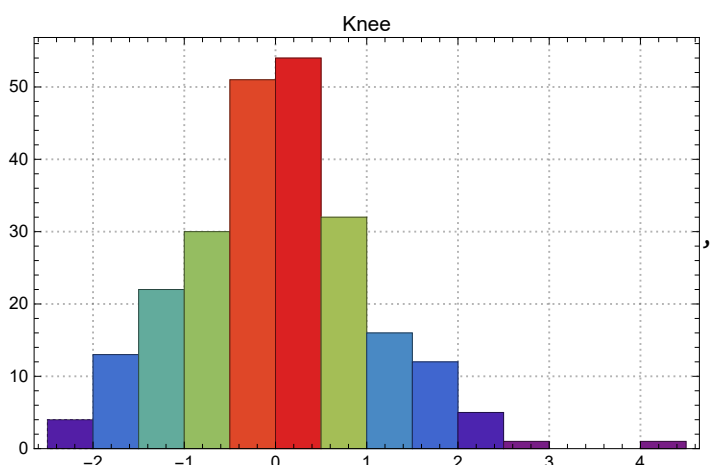
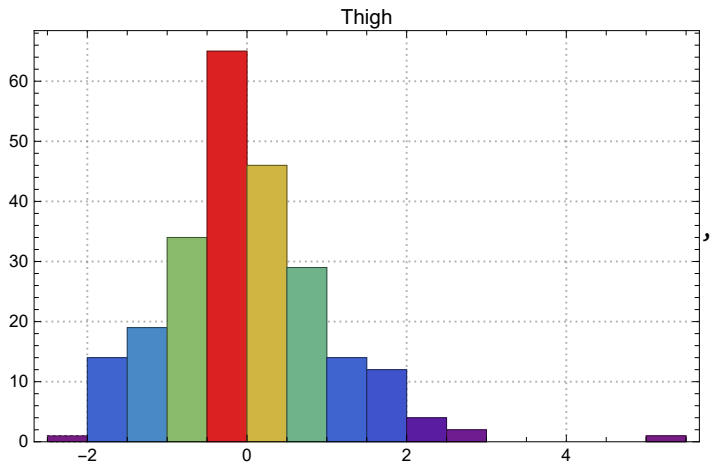
(*Creating the Standardized data set for better interpretation *)

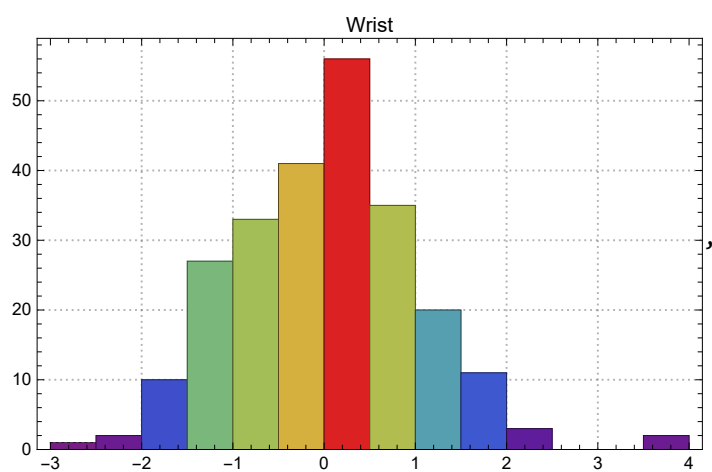
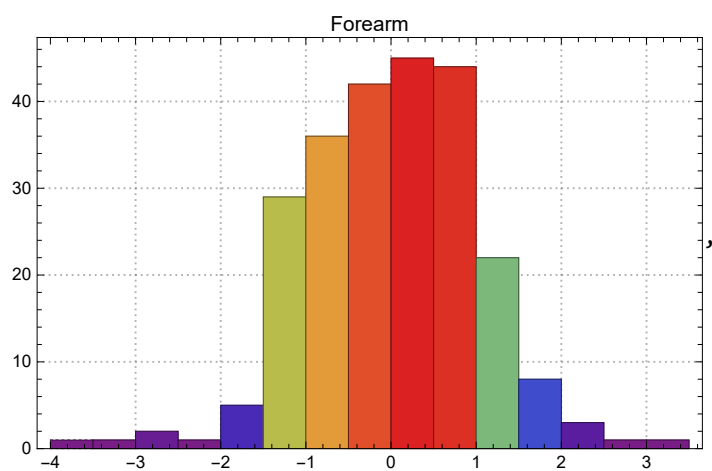
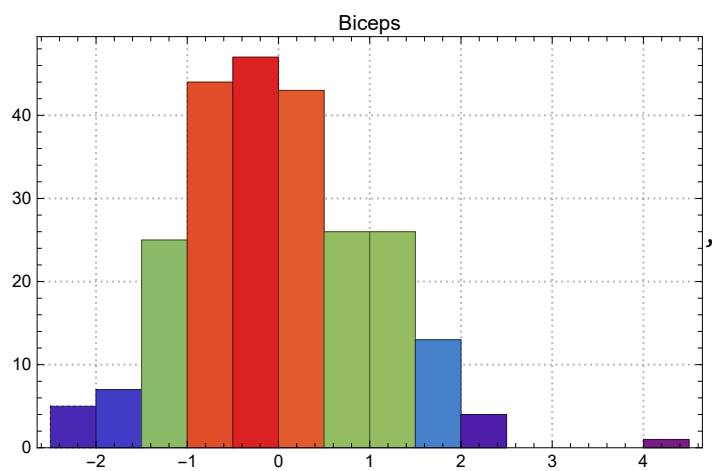
```
newdata = Dataset[Standardize[Values[Normal[data]]]];
```

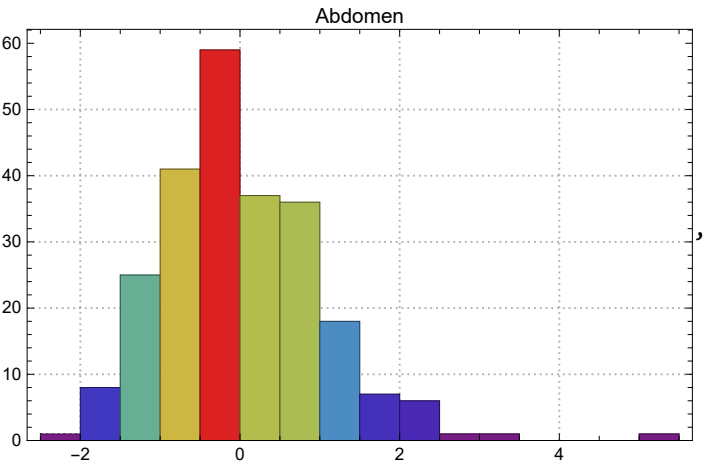
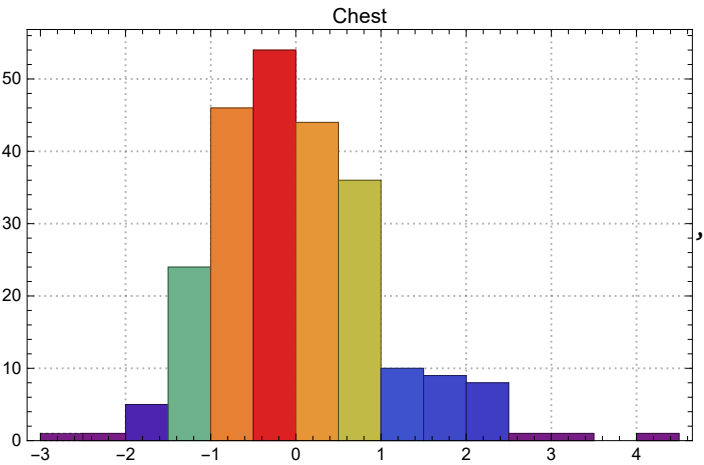
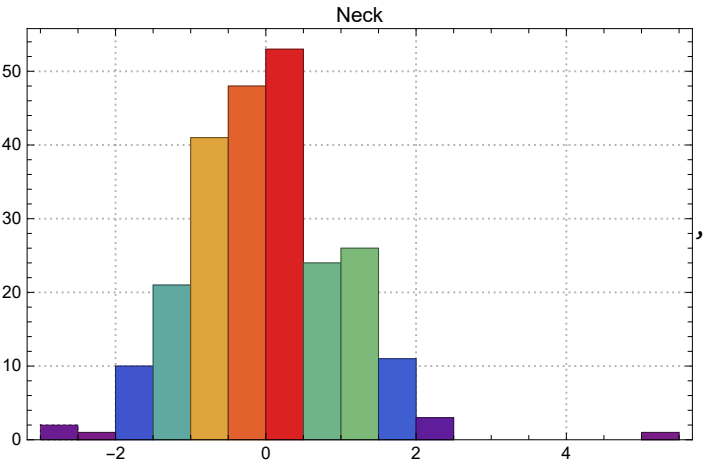
Histogram plot with the new standardized data set:

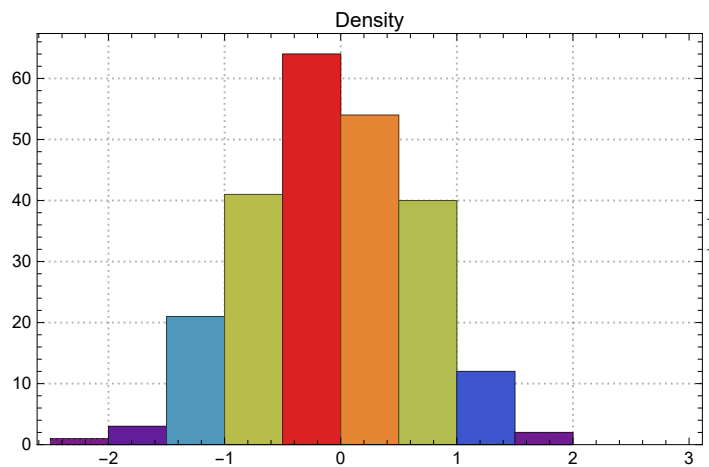
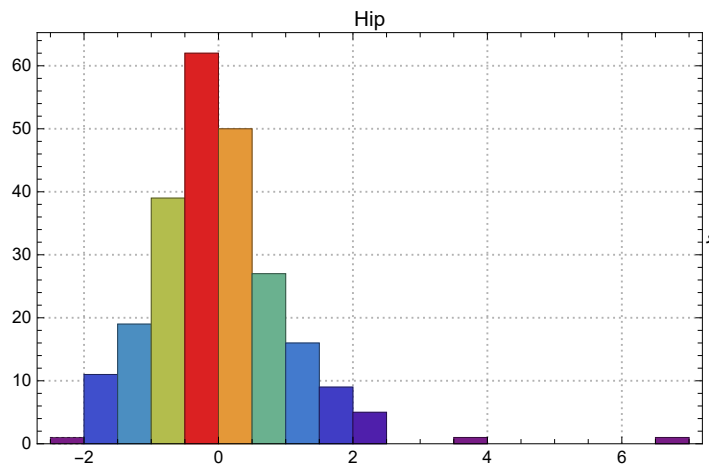
```
In[ ]:= Table[Histogram[newdata[All, i] // Normal,
  PlotTheme -> "Detailed", ColorFunction -> "Rainbow",
  PlotLabel -> predictorvariable[[i]], ImageSize -> Medium], {i, 1, 14, 1}]
```











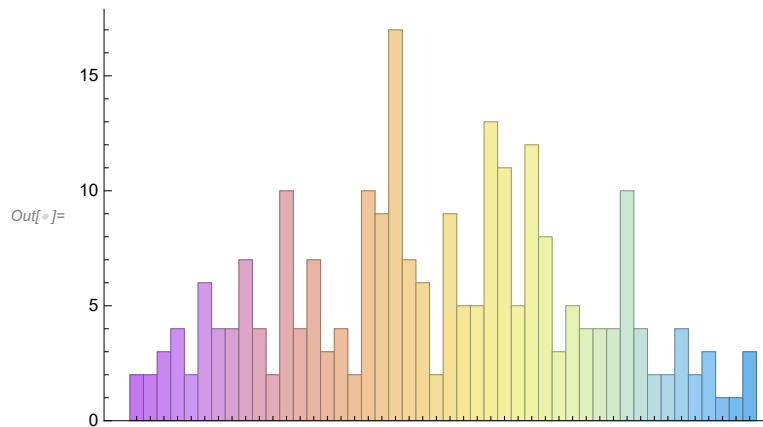
From the above Histograms we can infer that there are bell shaped curves (knee plot having perfect bell shape curve with just one outlier leading to its little skewness) with very few plot which are skewed and that may be due to the presence of outliers in the data set.

After standardizing the data set we can clearly see that we have more symmetric histogram plots with relatively less skewness.

3.5 Bar-chart :

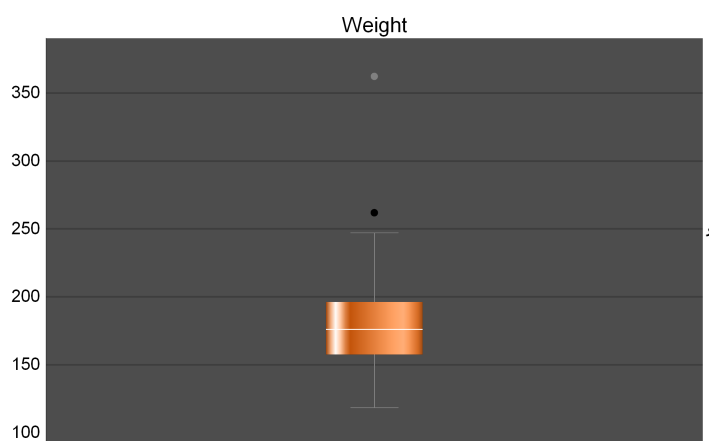
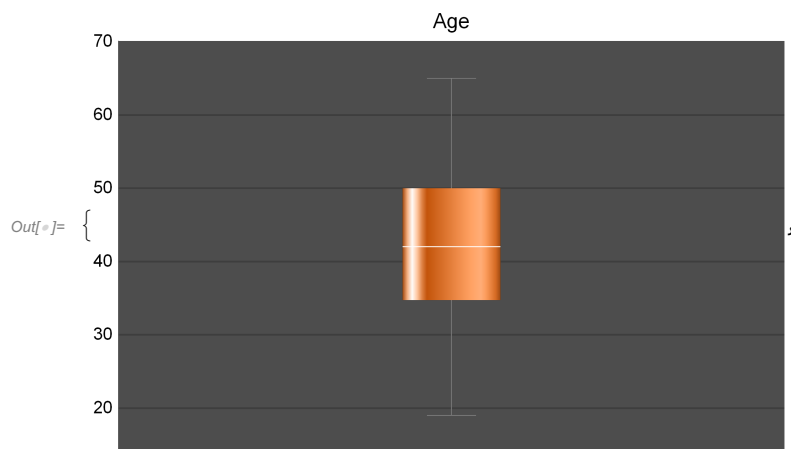
As each of the variables in the dataset have wide range of data and thus bar chart pictorial representation of each of them does not provide meaningful picture.Hence just plotting 1 st variable of the data set.

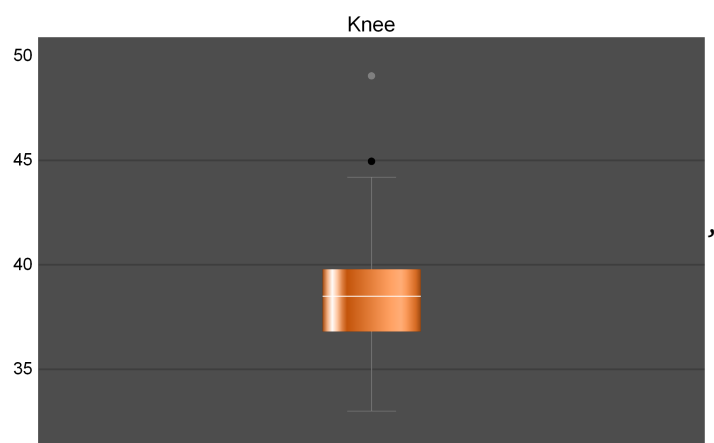
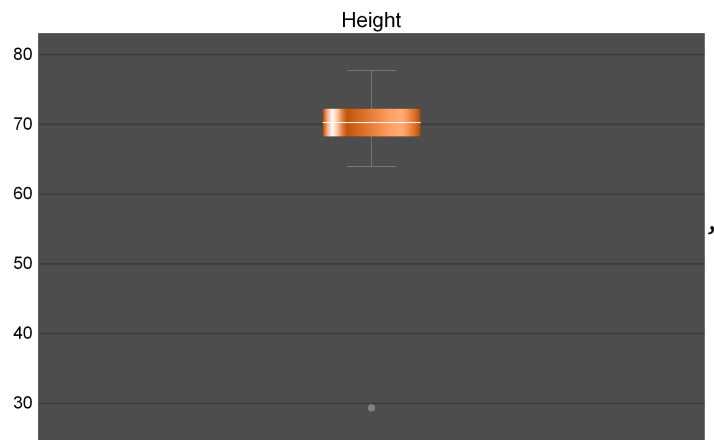

```
In[ ]:= BarChart[Counts[data[All, 1]], ChartStyle -> "Pastel", Frame -> {{True, None}, {True, None}}]
```

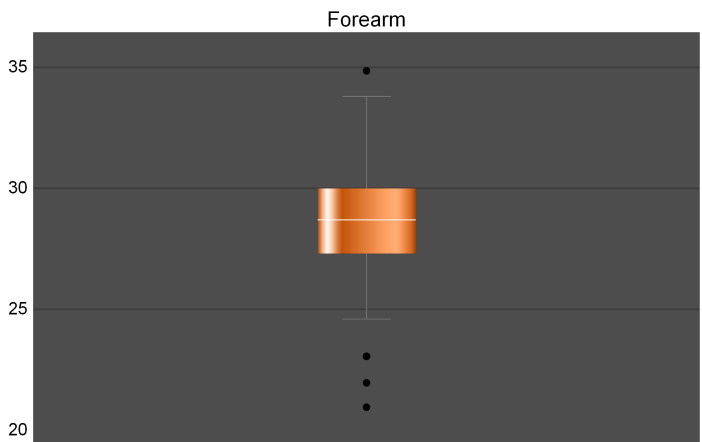
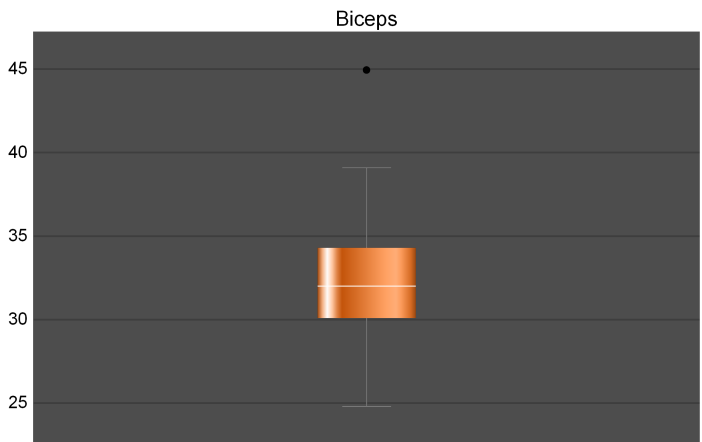
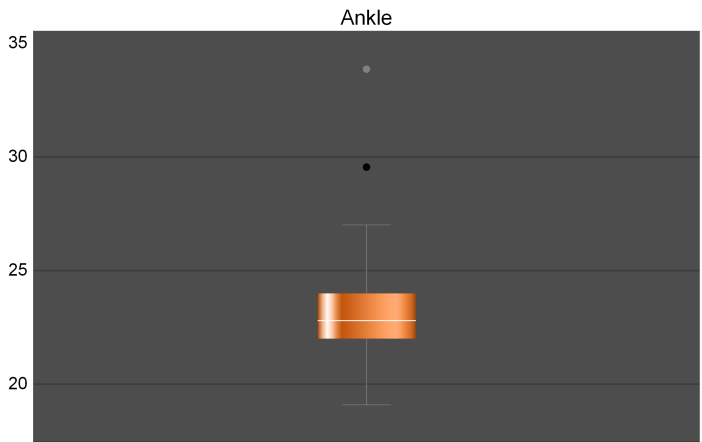


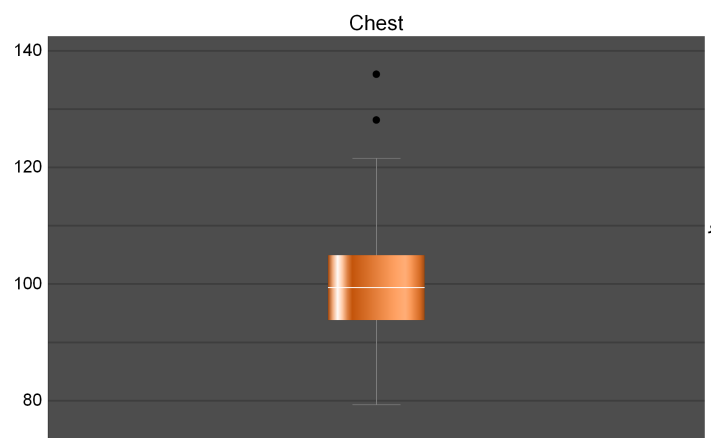
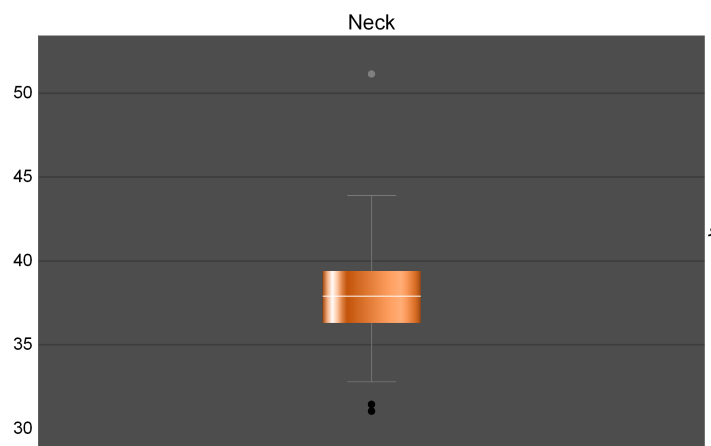
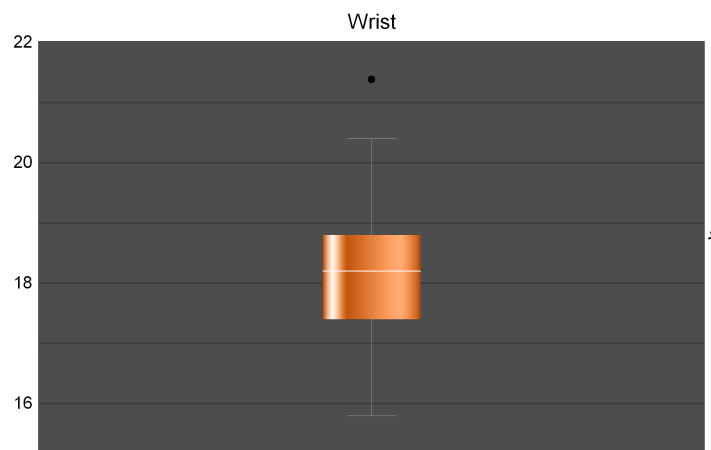
3.6 Box Plot :

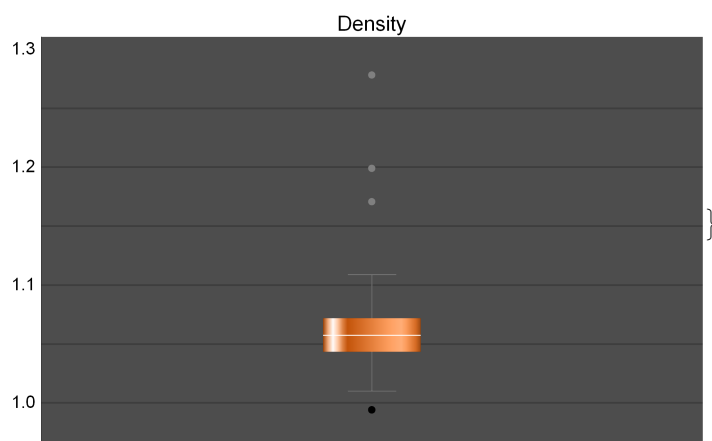
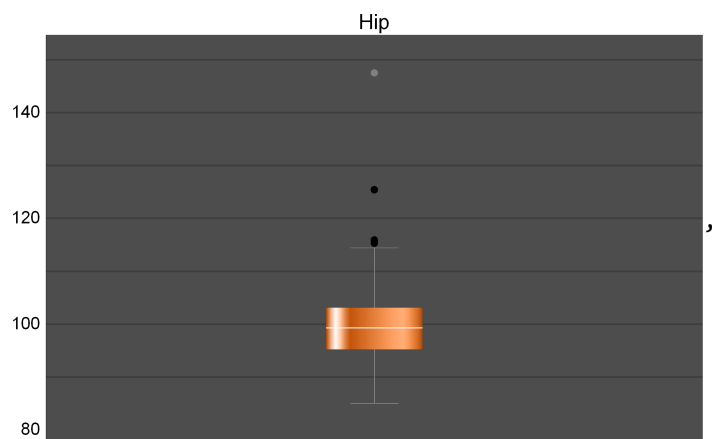
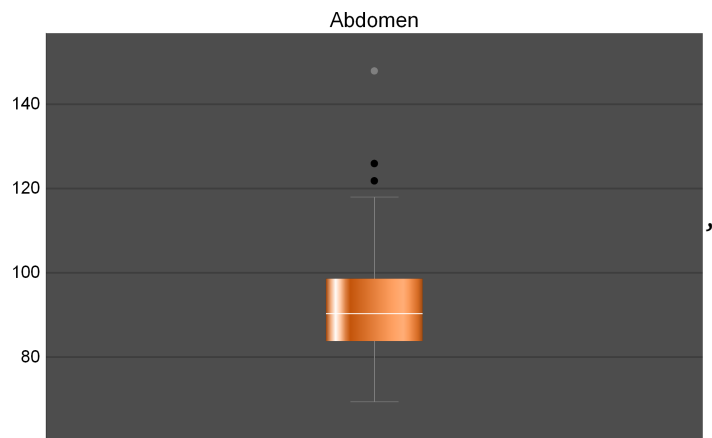
```
In[ ]:= Table[BoxWhiskerChart[data[All, i] // Normal, "Outliers", PlotTheme -> "Marketing",  
PlotLabel -> predictorvariable[[i]], ImageSize -> Medium], {i, 1, 14, 1}]
```











Box plot shows the Inter quartile range, 1st quartile, 3rd quartile, minimum, maximum and median for each of the variable. Outliers in the plot represent unusual values.

From the above Box Plot we can infer that maximum data for all variables lie in between upper and lower whiskers with very few outliers that may be present because of more weight composition of some individual. Hence outlier here is not any data entry error and we can't eliminate it from our data

set as they will also play important part in the model building and prediction.

Age Box plot has maximum variability because of large range of data in that category with no outliers, where as Height box plot has the minimum variability in it.

Chapter 4 - Modelling

Model selection is a task of selecting the statistical model from a set of candidate models for a given dataset. The model can also be a mathematical equation that describes the relationships among the predictor variables and response variables for the dataset. The model is divided into train and test data for the better prediction.

(a) Train data set is used to train the model for predicting the desired output by using different machine learning techniques.

(b) Test data is used for validating the predicted model for the prediction of response variable and finally comparing the error with the previous value of response variable before prediction analysis.

4.1 Training and Test Dataset :

Training Dataset :

The sample of data used to fit the model. The model sees and learns from this data. It is the actual dataset that we use to train the model.

80% of the actual data will be used for training.

```

In[ ]:= (* Diving the complete standardized dataset and taking 80% of it as train data *)
traindata = newdata[[0 ;; 186]];
(* Taking all Predictor variables as X *)
traindataX = traindata[[All, {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}]];
(* Taking Adipose (dependent variable) as Y *)
traindataY = traindata[[All, 15]];
(* Creating the complete Training data set
   with Dependent (Y) variable and Predictor (X) variables *)
traindataset = Table[N[Normal[traindataX[[i]]] -> N[Normal[traindataY[[i]]], {i, 1, 186}]];

```

Test Dataset :

The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. The Test dataset provides the standard used to evaluate the model.

20% of the remaining data will be used for testing

```

In[ ]:= (* Taking the rest of the standardized dataset i.e 20% of it as the test data *)
testdata = newdata[186 ;; 241];
(* Taking all Predictor variables as X *)
testdataX = testdata[All, {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}];
(* Taking Adipose(dependent variable) as Y *)
testdataY = testdata[All, 15];
(* Creating the complete Test data set with
Dependent(Y) variable and Predictor(X) variables *)
testdataset = Table[N[Normal[testdataX[i]]] -> N[Normal[testdataY[i]]], {i, 1, 55}];

```

Now we will use divided data sets : **traindataset** and **testdataset** to obtain the best model using the different Machine Learning algorithms.

4.2 : Model Building

1) Random Forest :

The random forest is a classification algorithm consisting of multiple decisions trees. It is basically implemented by using bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction is more accurate than that of any individual tree.

(*Applying the Random Forest model on the earlier obtained Training data-set*)

```

In[ ]:= PredictRandomF = Predict[traindataset, Method -> "RandomForest"]

```

```

Out[ ]:= PredictorFunction[   Input type: NumericalVector (length: 14)
Method: RandomForest ]

```

(*Applying the Predict model on the earlier obtained Test data-set*)

```

In[ ]:= PredictionRandomF = Table[PredictRandomF[Normal[N[testdataX[i]]]], {i, 1, 55}]
(*Use the model on the test dataset*)
Out[ ]:= {0.733377, 0.350069, 0.27947, 0.717173, -0.940664, 1.46391, -0.0630135, 0.933688,
0.00200066, 0.78784, 0.0793037, -0.016724, -1.13788, 0.384047, -0.348001, -0.0430626,
1.19406, -0.617152, 1.49496, -0.0990176, 0.937278, 1.09651, -0.861123, -0.87194,
-1.28669, 1.09076, 0.169179, 0.183598, 0.0381696, 1.51953, -0.664767, -1.20161,
0.710084, -0.336293, -0.36454, 0.984669, -0.641002, -1.29491, -0.284405, -1.09399,
-0.21673, 0.810831, -0.190428, -0.0309619, -0.85575, -0.0560834, -0.325743,
0.726872, 0.627778, 0.204079, 0.625286, 1.3969, -0.783538, 1.02853, -0.815791}

```

Accuracy of the model

```

In[ ]:= (*Finding the Regression Sum of Squared, Sum of Squared Error and Degree of Freedom*)
SSRRandom =
Total[Table[Power[PredictionRandomF[[i]] - Mean[testdataY] // N, 2], {i, 1, 55}]];
SSERandom = Total[Table[Power[PredictionRandomF[[i]] - N[testdataY[i]], 2], {i, 1, 55}]];
dfRandom = Length[PredictionRandomF];

```

```
In[ ]:= (*Finding the Mean squared error*)
```

```
MSERandom = SSERandom / dfRandom
```

```
Out[ ]:= 0.176945
```

```
In[ ]:= (*Finding the R2:Coefficient of Determination*)
```

```
RSquareRandom = SSRRandom / (SSRRandom + SSERandom)
```

```
Out[ ]:= 0.779055
```

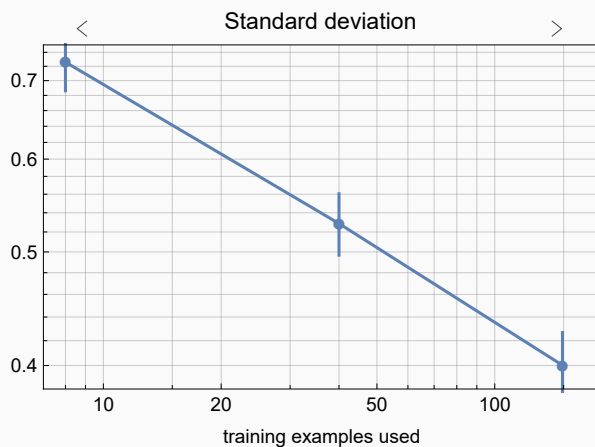
RSquare value is more than 50 % i.e around 77 % of variation in Adipose is being explained by predictor variables.

```
In[ ]:= PredictorInformation[PredictRandomF]
```

Predictor information

Input type	NumericalVector (length: 14)
Method	RandomForest
Standard deviation	0.400 ± 0.027
Loss	0.262 ± 0.050
Single evaluation time	5.63 ms/example
Batch evaluation speed	16.6 examples/ms
Predictor memory	211. kB
Training examples used	186 examples
Training time	1.69 s

```
Out[ ]:=
```

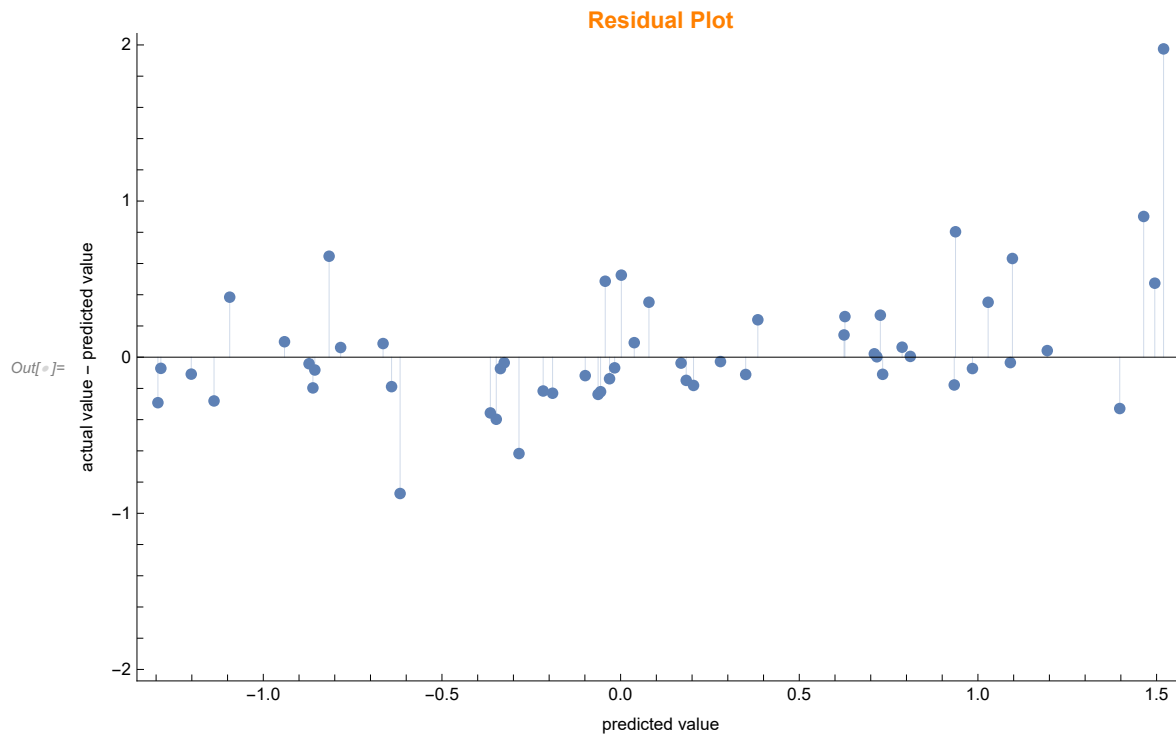


Validate Test Models

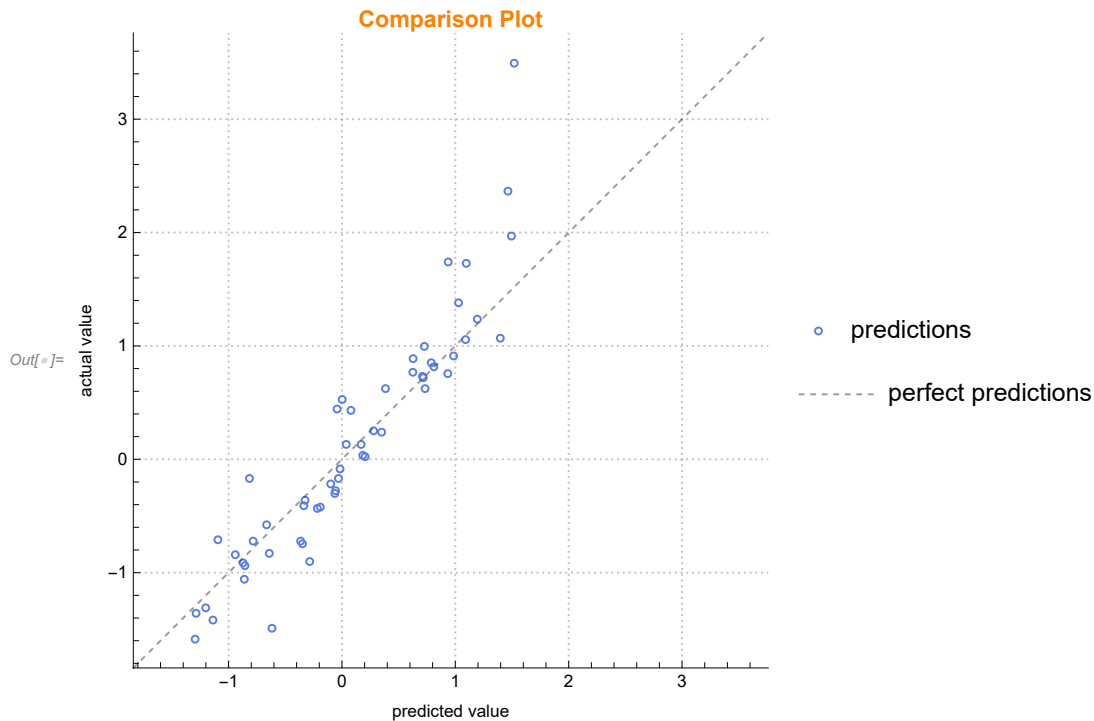
In[]:= **PredictRF = PredictorMeasurements[PredictRandomF, Normal[testdataset]]**

Out[]:= PredictorMeasurementsObject [ Predictor: RandomForest
Number of test examples: 55]

In[]:= **Show[PredictRF["ResidualPlot"], ImageSize → Large,
PlotLabel → Style["Residual Plot", Bold, 12, Orange]]**



```
In[ ]:= Show[PredictRF ["ComparisonPlot"], ImageSize → Medium,
PlotLabel → Style["Comparison Plot", Bold, 12, Orange]]
```




Actual value is closely distributed across the predicted value i.e the linear regression line. Hence we can say that model is a good fit.

2) Linear Regression:

It is a linear approach of modeling the relationship between a dependent variable and one or more independent/predictor variables.

```
In[ ]:= (*Applying the Linear Regression model on the earlier obtained Training data-set*)
PredictLinear = Predict[traindataset, Method → "LinearRegression"]
```

```
Out[ ]:= PredictorFunction[  Input type: NumericalVector (length: 14)
Method: LinearRegression ]
```

```
In[ ]:= (*Applying the Predict model on the earlier obtained Test data-set*)
PredictionLinear = Table[PredictLinear[Normal[N[testdataX[i]]]], {i, 1, 55}]
```

```
Out[ ]:= {1.01451, 0.360951, 0.473892, 0.832623, -0.910806, 1.48153, -0.023152, 0.86776,
-0.0491751, 0.591789, 0.0757132, 0.115299, -1.38479, 0.139953, -0.310223, -0.0920332,
1.21474, -0.533953, 2.07733, -0.415645, 0.812897, 1.01204, -0.567192, -0.557381,
-0.804215, 0.974393, -0.199028, 0.286756, -0.284581, 2.84931, -0.710202, -1.10937,
0.675485, -0.139791, 0.124629, 1.33323, -0.280872, -0.463838, 0.0292857, -0.564701,
-0.0227644, 0.535019, -0.0920435, 0.0949141, 0.0168164, 0.237322, -0.174428,
0.664198, 0.389817, 0.405317, 0.412246, 1.72257, -0.419312, 0.955177, -0.236735}
```

Accuracy of the model:

```
In[ ]:= (*Finding the Regression Sum of Squared, Sum of Squared Error and Degree of Freedom*)
SSRLinear =
  Total[Table[Power[PredictionLinear[[i]] - Mean[testdataY] // N, 2], {i, 1, 55}]];
SSELinear = Total[Table[Power[PredictionLinear[[i]] - N[testdataY[i]], 2], {i, 1, 55}]];
dfLinear = Length[PredictionLinear];

(*Finding the Mean squared error*)
```

```
In[ ]:= MSELinear = SSELinear / dfLinear
```

```
Out[ ]:= 0.229367
```

```
In[ ]:= (*Finding the R2:Coefficient of Determination*)
RsquareLinear = SSRLinear / (SSRLinear + SSELinear)
```

```
Out[ ]:= 0.73006
```

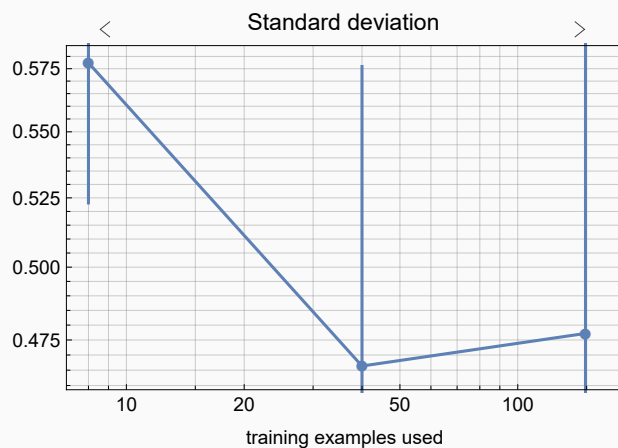
RSquare value is more than 50 % i.e around 73 % of variation in Adipose is being explained by predictor variables.

```
In[ ]:= PredictorInformation[PredictLinear]
```

Predictor information

Input type	NumericalVector (length: 14)
Method	LinearRegression
Standard deviation	0.464 ± 0.15
Loss	0.551 ± 0.93
Single evaluation time	2.04 ms/example
Batch evaluation speed	48.7 examples/ms
Predictor memory	252. kB
Training examples used	186 examples
Training time	7.61 s

```
Out[ ]:=
```



Validate Test Models

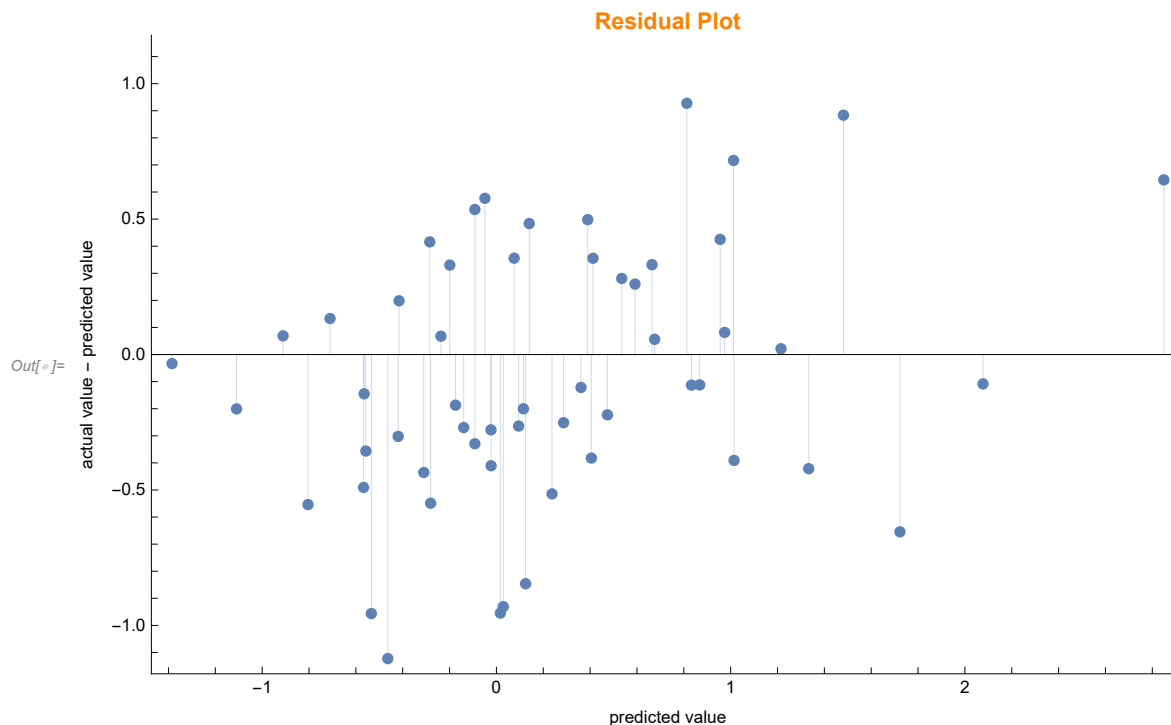
```
In[ ]:= PredictLR = PredictorMeasurements[PredictLinear, Normal[testdataset]]
```

```
Out[ ]:= PredictorMeasurementsObject [
```

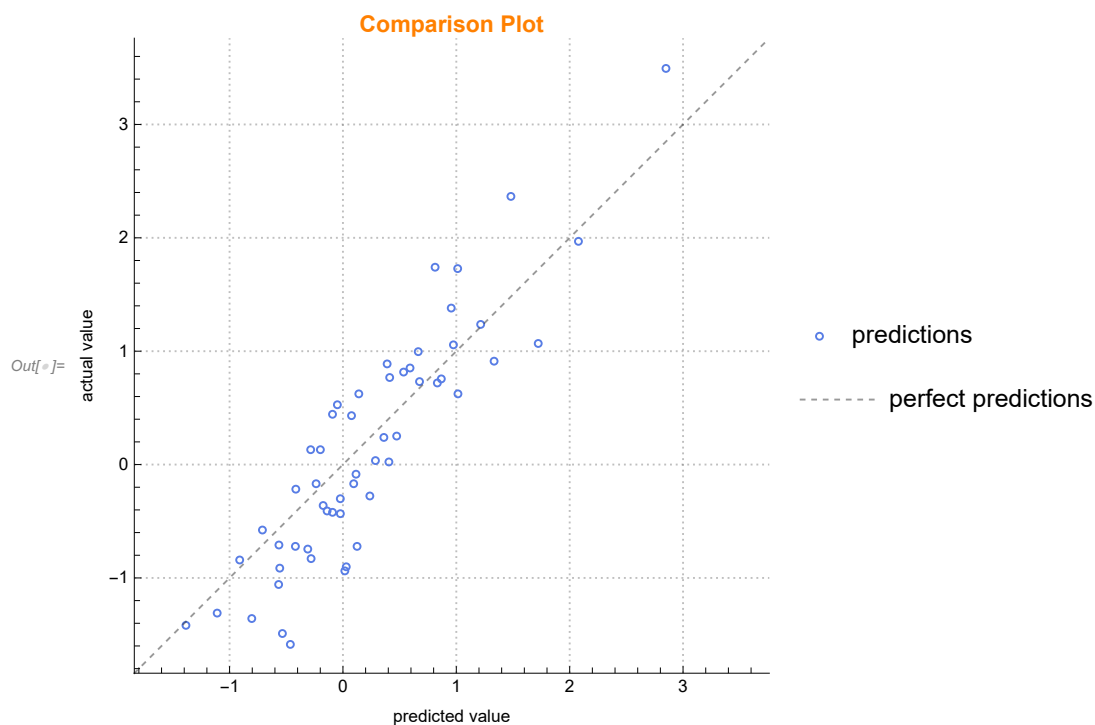


Predictor: LinearRegression
Number of test examples: 55

```
In[ ]:= Show[PredictLR["ResidualPlot"], ImageSize → Large,  
PlotLabel → Style["Residual Plot", Bold, 12, Orange]]
```



```
In[ ]:= Show[PredictLR["ComparisonPlot"], ImageSize → Medium,  
PlotLabel → Style["Comparison Plot", Bold, 12, Orange]]
```



Actual value is closely distributed across the predicted value i.e the linear regression line. Hence we can

say that model is a good fit.

3) Decision Tree

Decision trees algorithm comprises of the tree like model having different nodes where each node have different decisions and subsequent sub - node has its consequences. They are widely used in decision analysis

```
In[ ]:= (*Applying the Decision Tree model on the earlier obtained Training data-set*)
PredictDecisionTree = Predict[traindataset, Method -> "DecisionTree"]
```

```
Out[ ]:= PredictorFunction[  Input type: NumericalVector (length: 14)
Method: DecisionTree ]
```

```
In[ ]:= (*Applying the Predict model on the earlier obtained Test data-set*)
PredictionDecisionTree = Table[PredictDecisionTree[Normal[N[testdataX[i]]]], {i, 1, 55}]
(*Use the model on the test dataset*)
```

```
Out[ ]:= {0.365089, 0.365089, 0.365089, 0.365089, -0.514858, 1.36074, -0.514858, 0.365089,
0.365089, 1.36074, 0.365089, -0.514858, -1.27996, 0.365089, -0.514858, 0.365089,
1.36074, -1.27996, 1.36074, -0.514858, 1.36074, 1.36074, -1.27996, -1.27996,
-1.27996, 1.36074, 0.365089, 0.365089, 0.365089, 1.36074, -0.514858, -1.27996,
0.365089, -0.514858, -0.514858, 1.36074, -0.514858, -1.27996, -0.514858, -0.514858,
-0.514858, 0.365089, -0.514858, -0.514858, -1.27996, -0.514858, -0.514858,
1.36074, 1.36074, 0.365089, 0.365089, 1.36074, -0.514858, 1.36074, -0.514858}
```

Accuracy of the model

```
In[ ]:= (*Finding the Regression Sum of Squared, Sum of Squared Error and Degree of Freedom*)
SSRDecison =
Total[Table[Power[PredictionDecisionTree[[i]] - Mean[testdataY] // N, 2], {i, 1, 55}]];
SSEDecison = Total[Table[Power[PredictionDecisionTree[[i]] - N[testdataY[i]], 2],
{i, 1, 55}]];
dfDecison = Length[PredictionDecisionTree];
```

```
In[ ]:= (*Finding the Mean squared error*)
MSEDecison = SSEDecison / dfDecison
```

```
Out[ ]:= 0.185457
```

```
In[ ]:= (*Finding the R2:Coefficient of Determination*)
RsquareDecison = SSRDecison / (SSRDecison + SSEDecison)
```

```
Out[ ]:= 0.813154
```

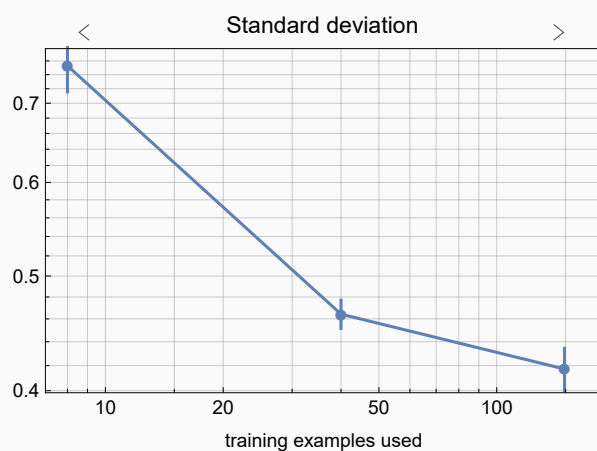
RSquare value is more than 80 % i.e around 81 % of variation in Adipose is being explained by predictor variables.

In[]:= **PredictorInformation[PredictDecisionTree]**

Predictor information

Input type	NumericalVector (length: 14)
Method	DecisionTree
Standard deviation	0.417 \pm 0.017
Loss	0.290 \pm 0.032
Single evaluation time	993. μ s/example
Batch evaluation speed	606. examples/ms
Predictor memory	129. kB
Training examples used	186 examples
Training time	1.53 s

Out[]:=



Validate Test Models

In[]:= **PredictDT = PredictorMeasurements[PredictDecisionTree, Normal[testdataset]]**

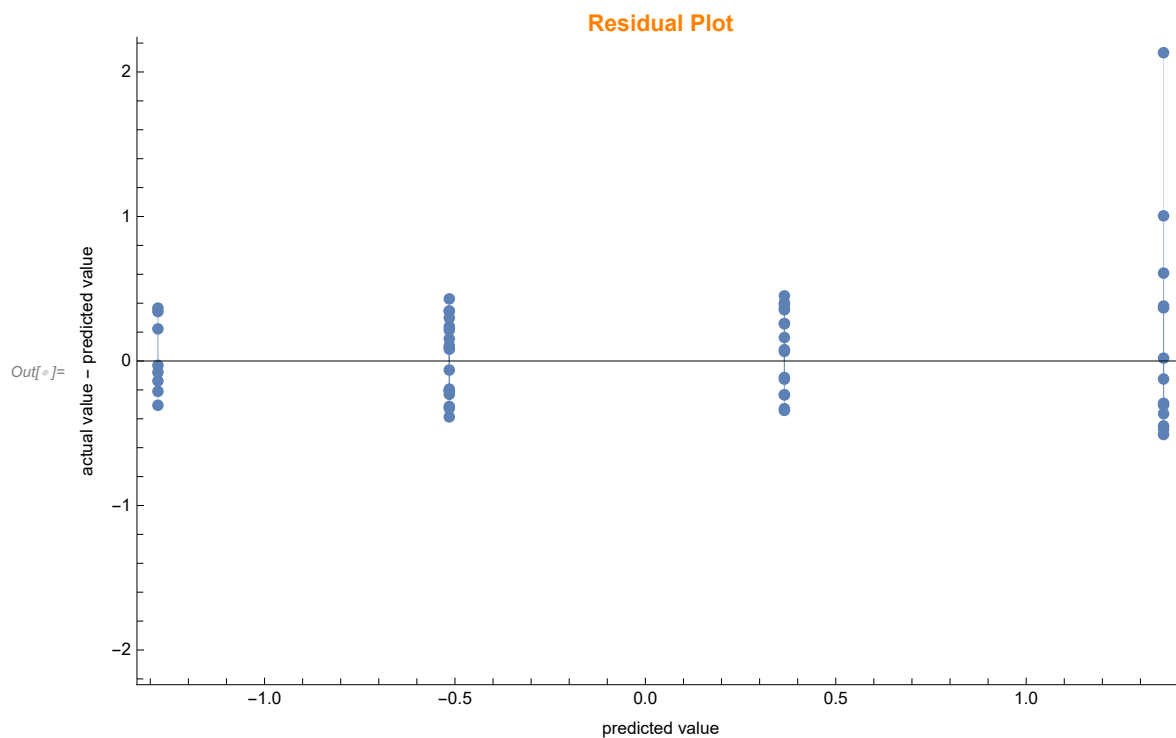
Out[]:= **PredictorMeasurementsObject** [



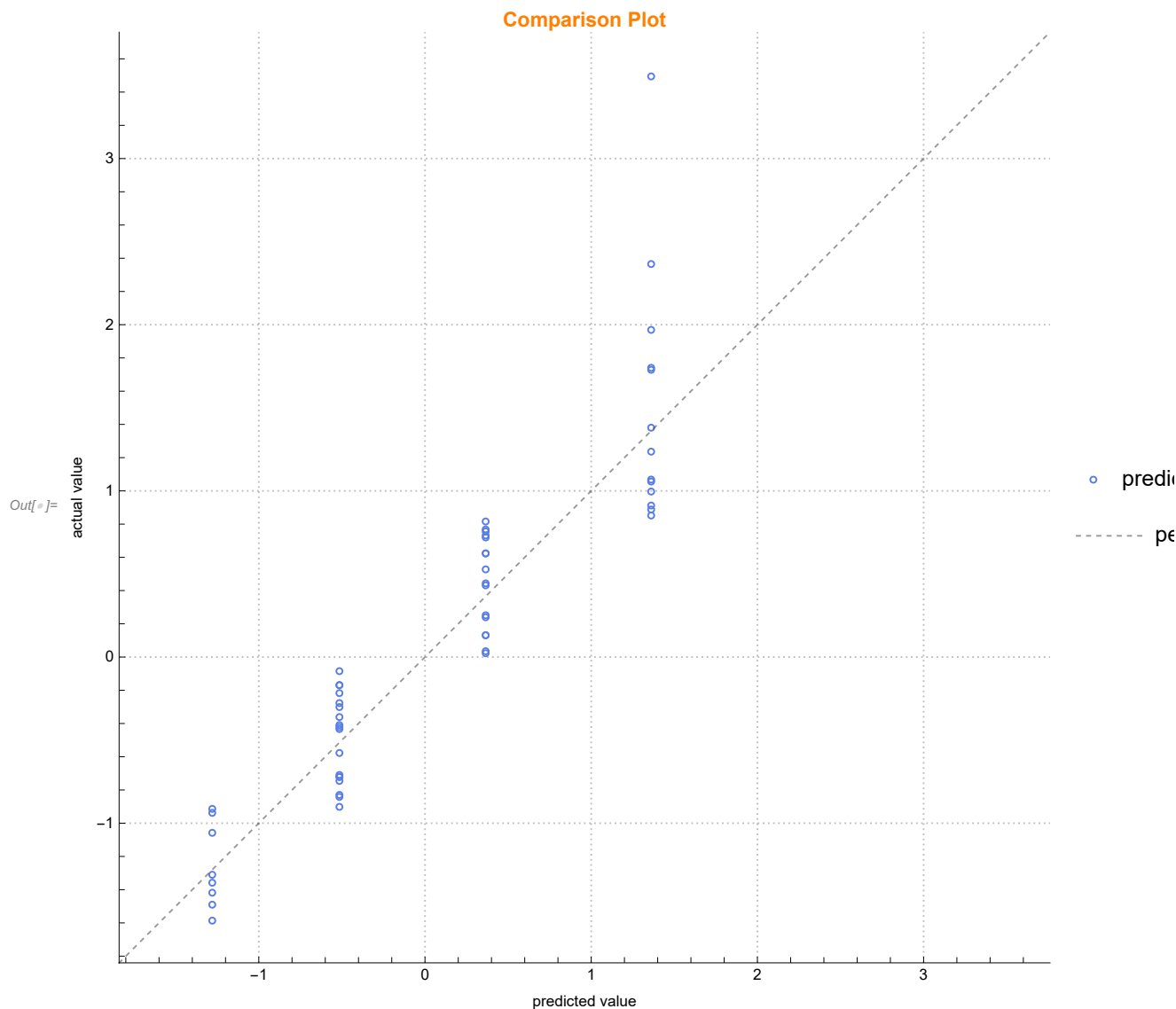
Predictor: DecisionTree

Number of test examples: 55

```
In[ ]:= Show[PredictDT["ResidualPlot"], ImageSize → Large,  
PlotLabel → Style["Residual Plot", Bold, 12, Orange]]
```




```
In[ ]:= Show[PredictDT ["ComparisonPlot"], ImageSize → Large,
  PlotLabel → Style["Comparison Plot", Bold, 12, Orange]]
```



4) Gradient Boosting :

It is machine learning algorithm that make predictions by making cluster of considerable weak predictions and further analyze them.

```
In[ ]:= (*Applying the Gradient Boosted Trees model on the earlier obtained Training data-set*)
PredictGradient = Predict[trainedataset, Method → "GradientBoostedTrees"]
```

```
Out[ ]:= PredictorFunction[   Input type: NumericalVector (length: 14)
Method: GradientBoostedTrees ]
```

(*Applying the Predict model on the earlier obtained Test data-set*)

```

In[ ]:= PredictionGradient = Table[PredictGradient[N[testdataX[i]] // Normal], {i, 1, 55}]
Out[ ]:= {0.531328, 0.205763, 0.212439, 0.625216, -0.79135, 1.66127, -0.313159, 0.634869, 0.438101,
0.719413, 0.380008, -0.0846578, -1.19443, 0.475012, -0.57073, 0.386058, 1.07074,
-0.86763, 1.65269, -0.153374, 1.57601, 1.58532, -0.974143, -0.786123, -1.25864,
0.915309, 0.131832, -0.00764074, 0.131832, 1.68586, -0.558053, -0.991611, 0.625216,
-0.447747, -0.559177, 0.795325, -0.684067, -1.35651, -0.706901, -0.686768,
-0.364934, 0.702333, -0.344828, -0.178289, -0.725027, -0.326812, -0.350625,
0.862014, 0.76568, 0.00711968, 0.66811, 0.927557, -0.574734, 1.18066, -0.308053}

```

Accuracy of the model

```

In[ ]:= (*Finding the Regression Sum of Squared, Sum of Squared Error and Degree of Freedom*)
SSRGradient =
  Total[Table[Power[PredictionGradient[[i]] - Mean[testdataY] // N, 2], {i, 1, 55}]];
SSEGradient = Total[Table[Power[PredictionGradient[[i]] - N[testdataY[i]], 2],
  {i, 1, 55}]];
dfGradient = Length[PredictionGradient];

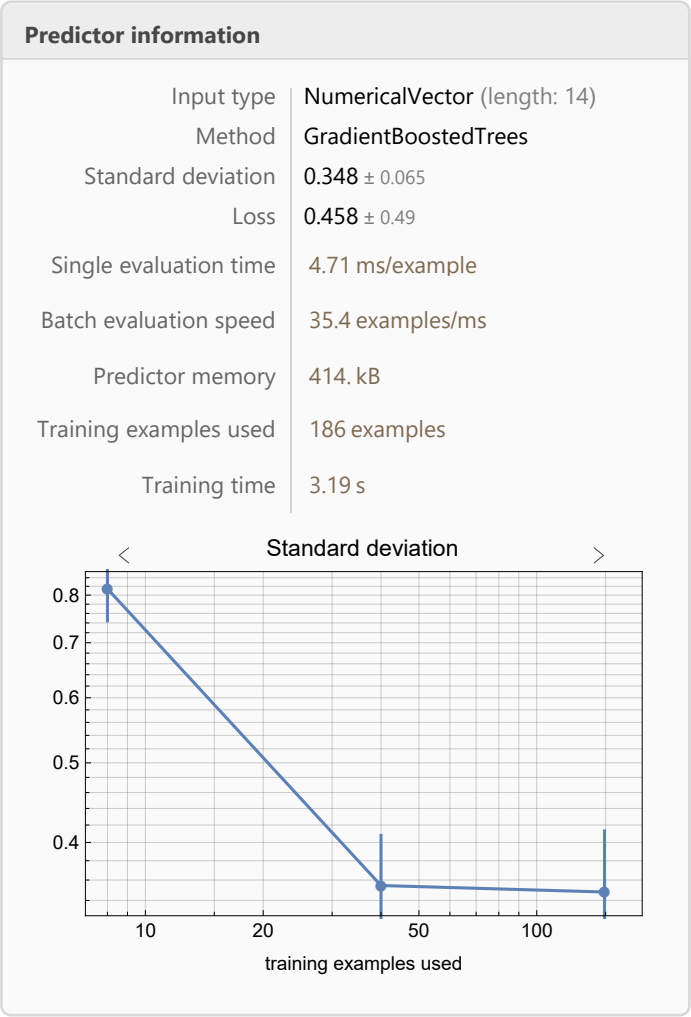
In[ ]:= (*Finding the Mean squared error*)
MSEGradient = SSEGradient / dfGradient
Out[ ]:= 0.0916512

In[ ]:= (*Finding the R2:Coefficient of Determination*)
RsquareGradient = SSRGradient / (SSRGradient + SSEGradient)
Out[ ]:= 0.877402

```

RSquare value is more than 85 % i.e around 87 % of variation in Adipose is being explained by predictor variables.

```
In[ ]:= PredictorInformation[PredictGradient]
```



Validate Test Models

```
In[ ]:= PredictGR = PredictorMeasurements[PredictGradient, Normal[testdataset]]
```

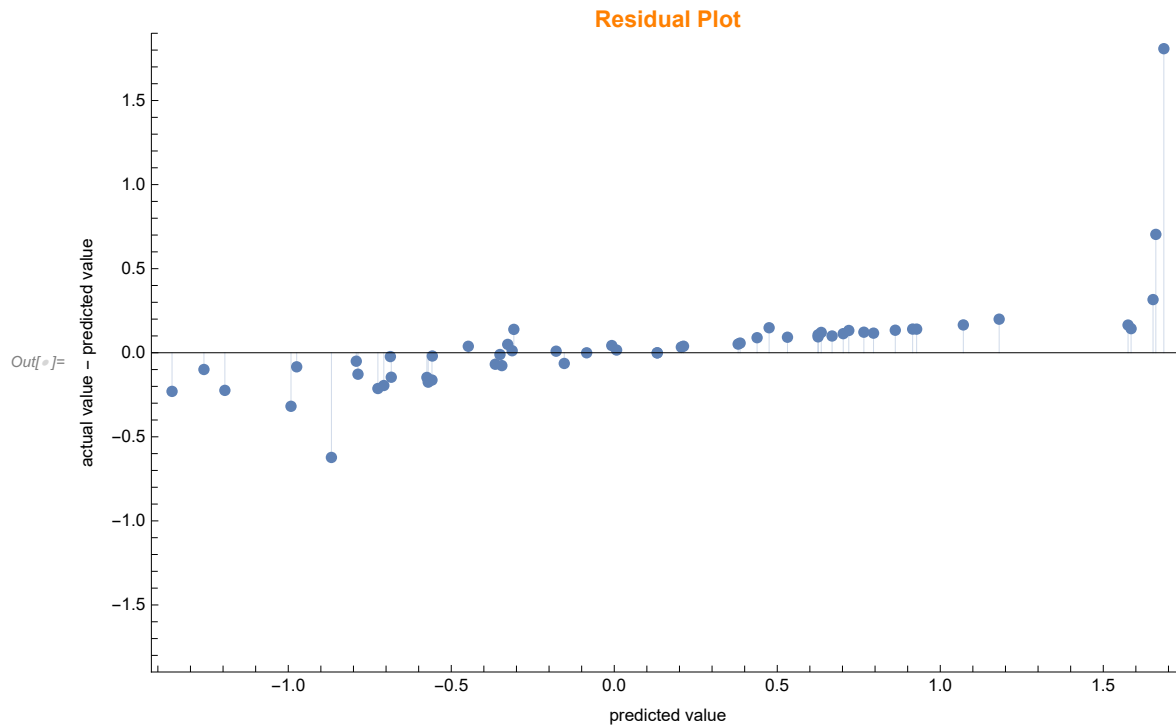
Out[]:= PredictorMeasurementsObject [

Predictor: GradientBoostedTrees

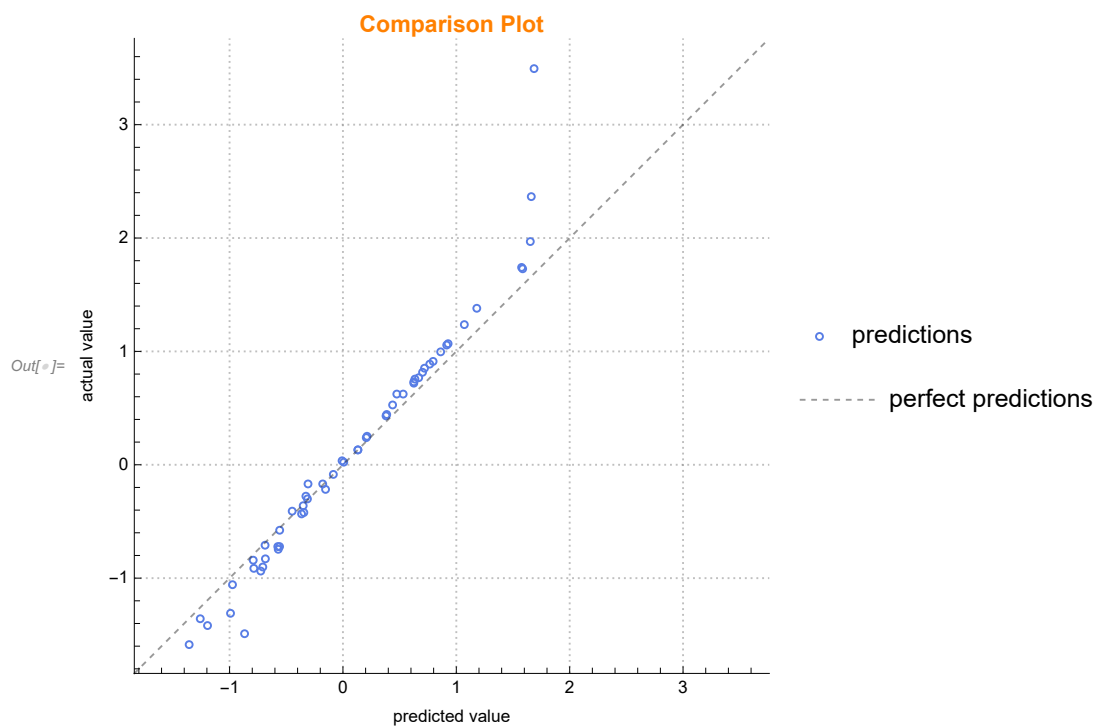
Number of test examples: 55

]

```
In[ ]:= Show[PredictGR["ResidualPlot"], ImageSize → Large,  
PlotLabel → Style["Residual Plot", Bold, 12, Orange]]
```



```
In[ ]:= Show[PredictGR["ComparisonPlot"], ImageSize → Medium,  
PlotLabel → Style["Comparison Plot", Bold, 12, Orange]]
```



Actual value is closely distributed across the predicted value i.e the linear regression line. Hence we can

say that model is a good fit.

5) Nearest Neighbour :

This algorithm collates all cases and classifies the combined data based on a similarity.

Train Models

```
In[ ]:= (*Applying the Nearest Neighbour model on the earlier obtained Training data-set*)
PredictNearest = Predict[traindataset, Method -> "DecisionTree"]
```

```
Out[ ]:= PredictorFunction[  Input type: NumericalVector (length: 14)
Method: DecisionTree ]
```

```
In[ ]:= (*Applying the Predict model on the earlier obtained Test data-set*)
PredictionNearest = Table[PredictNearest[N[testdataX[i]] // Normal], {i, 1, 55}]

Out[ ]:= {0.365089, 0.365089, 0.365089, 0.365089, -0.514858, 1.36074, -0.514858, 0.365089,
0.365089, 1.36074, 0.365089, -0.514858, -1.27996, 0.365089, -0.514858, 0.365089,
1.36074, -1.27996, 1.36074, -0.514858, 1.36074, 1.36074, -1.27996, -1.27996,
-1.27996, 1.36074, 0.365089, 0.365089, 0.365089, 1.36074, -0.514858, -1.27996,
0.365089, -0.514858, -0.514858, 1.36074, -0.514858, -1.27996, -0.514858, -0.514858,
-0.514858, 0.365089, -0.514858, -0.514858, -1.27996, -0.514858, -0.514858,
1.36074, 1.36074, 0.365089, 0.365089, 1.36074, -0.514858, 1.36074, -0.514858}
```

Accuracy of the model

```
In[ ]:= (*Finding the Regression Sum of Squared, Sum of Squared Error and Degree of Freedom*)
SSRNearest =
Total[Table[Power[PredictionNearest[[i]] - Mean[testdataY] // N, 2], {i, 1, 55}]];
SSENearest = Total[Table[Power[PredictionNearest[[i]] - N[testdataY[i]], 2], {i, 1, 55}]];
dfNearest = Length[PredictionNearest];
```

```
In[ ]:= (*Finding the Mean squared error*)
MSENearest = SSENearest / dfNearest
```

```
Out[ ]:= 0.185457
```

```
In[ ]:= (*Finding the R2:Coefficient of Determination*)
RsquareNearest = SSRNearest / (SSRNearest + SSENearest)
```

```
Out[ ]:= 0.813154
```

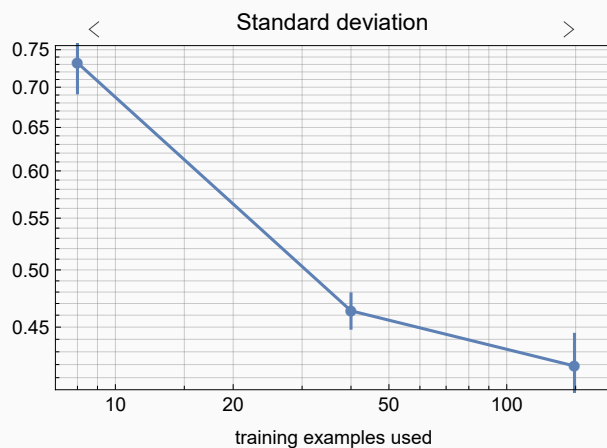
RSquare value is more than 80 % i.e around 81 % of variation in Adipose is being explained by predictor variables.

```
In[ ]:= PredictorInformation[PredictNearest]
```

Predictor information

Input type	NumericalVector (length: 14)
Method	DecisionTree
Standard deviation	0.419 \pm 0.025
Loss	0.313 \pm 0.050
Single evaluation time	1.64 ms/example
Batch evaluation speed	369. examples/ms
Predictor memory	125. kB
Training examples used	186 examples
Training time	1.54 s

```
Out[ ]:=
```



Validate Test Models

```
In[ ]:= PredictNN = PredictorMeasurements[PredictNearest, Normal[testdataset]]
```

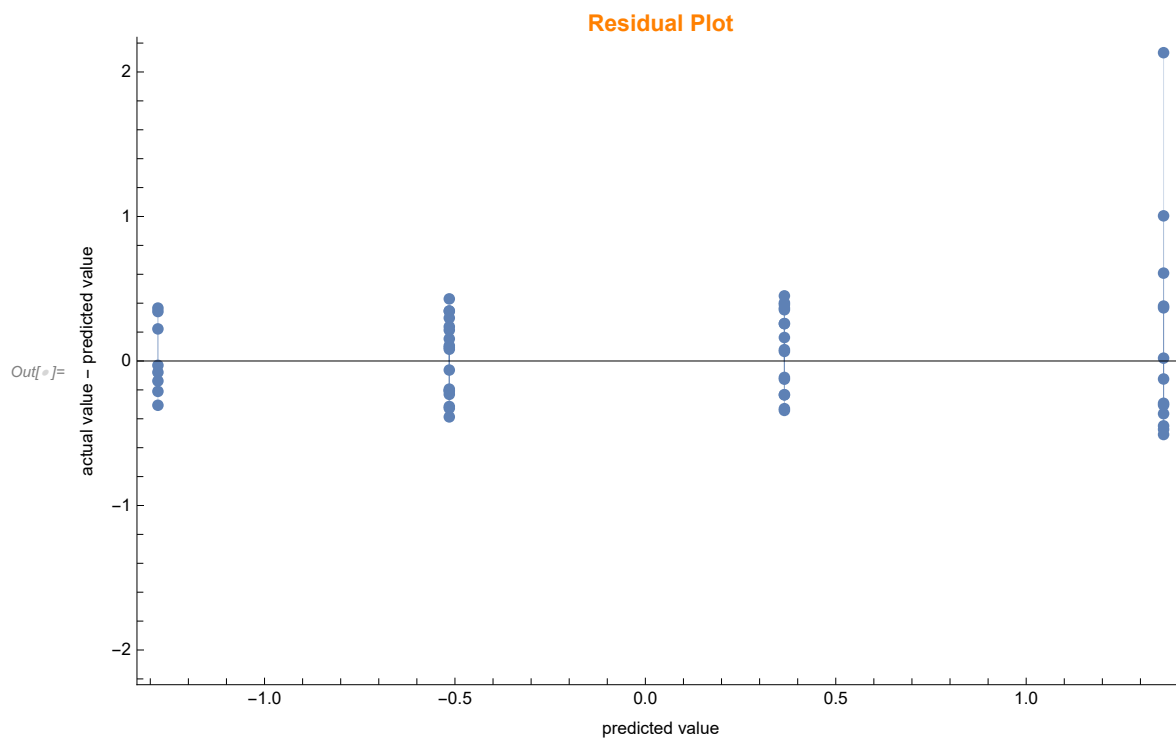
```
Out[ ]:= PredictorMeasurementsObject [
```



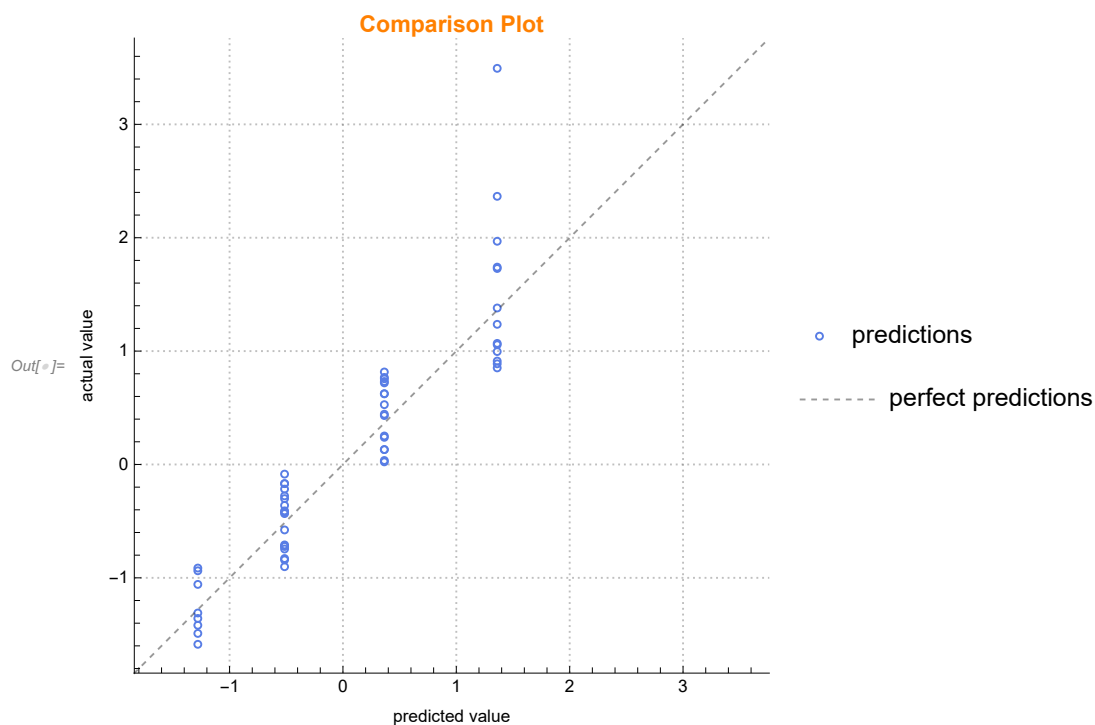
Predictor: DecisionTree

Number of test examples: 55

```
In[ ]:= Show[PredictNN["ResidualPlot"], ImageSize → Large,  
PlotLabel → Style["Residual Plot", Bold, 12, Orange]]
```



```
In[ ]:= Show[PredictNN["ComparisonPlot"], ImageSize → Medium,  
PlotLabel → Style["Comparison Plot", Bold, 12, Orange]]
```





6) Neural Network

This technique works closely like the human brain. It comprises of connected neural networks and it interprets the data through machine perception.

Train Models

```
In[ ]:= (*Applying the Neural Network model on the earlier obtained Training data-set*)
PredictNeural = Predict[traindataset, Method -> "NeuralNetwork"]
```

```
Out[ ]:= PredictorFunction[
   Input type: NumericalVector (length: 14)
  Method: NeuralNetwork
   Data not in notebook; Store now »
]
```

(*Applying the Predict model on the earlier obtained Test data-set*)

```
In[ ]:= PredictionNeural = Table[PredictNeural[N[testdataX[i]] // Normal], {i, 1, 55}]
(*Use the model on the test dataset*)
Out[ ]:= {-0.178799, -0.123987, -0.217988, -0.0137988, -0.0134318, -0.019696, -0.198528,
-0.0581394, 0.00834404, -0.0647276, -0.152547, 0.000422292, -0.118825, -0.162224,
-0.130768, 0.00676463, 0.0148463, -0.195758, -0.106286, 0.0260879, -0.0441419,
-0.168758, -0.196553, -0.00358598, -0.202022, -0.173528, -0.0239107, -0.171536,
0.0119973, -0.0531618, -0.0141655, 0.0113871, -0.0640388, -0.0488886, -0.204239,
-0.228788, -0.00206829, -0.191875, -0.207833, 0.0180893, -0.182372, -0.0458961,
-0.19986, -0.176056, -0.207628, -0.0761177, -0.190954, 0.0236572, -0.0550842,
-0.105178, -0.0358422, -0.0676905, -0.0348333, -0.183505, -0.0458462}
```

Accuracy of the model

```
In[ ]:= (*Finding the Regression Sum of Squared, Sum of Squared Error and Degree of Freedom*)
SSRNeural =
  Total[Table[Power[PredictionNeural[[i]] - Mean[testdataY] // N, 2], {i, 1, 55}]];
SSENeural = Total[Table[Power[PredictionNeural[[i]] - N[testdataY[i]], 2], {i, 1, 55}]];
dfNeural = Length[PredictionNeural];
```

```
In[ ]:= (*Finding the Mean squared error*)
MSENeural = SSENeural / dfNeural
```

```
Out[ ]:= 1.1005
```

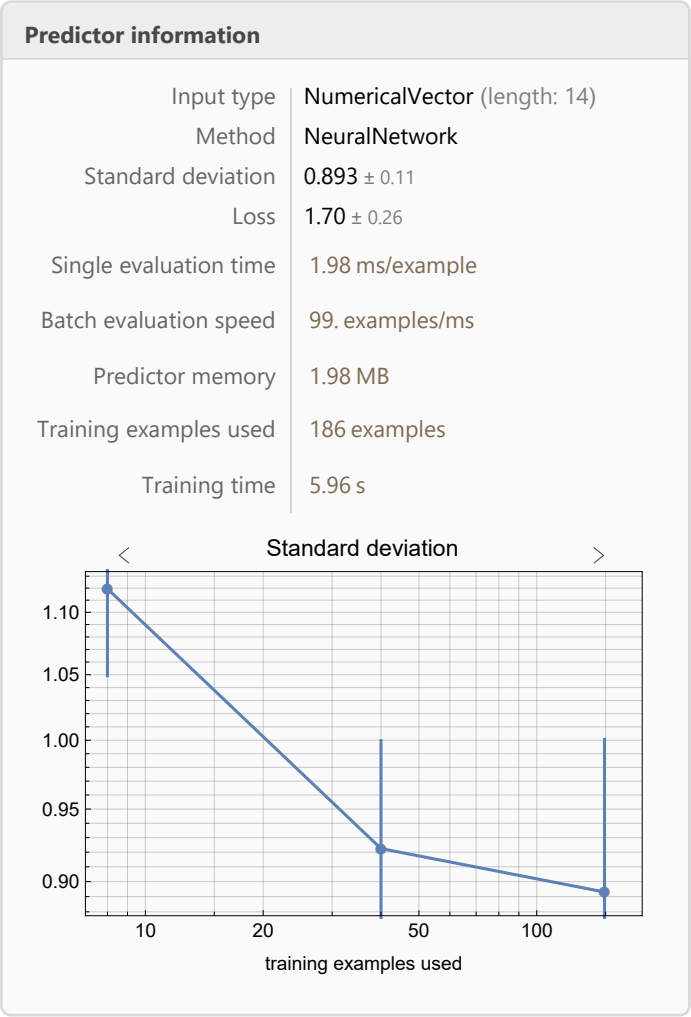
```
In[ ]:= (*Finding the R2:Coefficient of Determination*)
RSquareNeural = SSRNeural / (SSRNeural + SSENeural)
```

```
Out[ ]:= 0.0689622
```

RSquare value is more than 50 % i.e around 68 % of variation in Adipose is being explained by predictor variables.


```
In[ ]:= PredictorInformation[PredictNeural]
```

Out[]:=



Validate Test Models

```
In[ ]:= PredictNeu = PredictorMeasurements[PredictNeural, Normal[testdataset]]
```

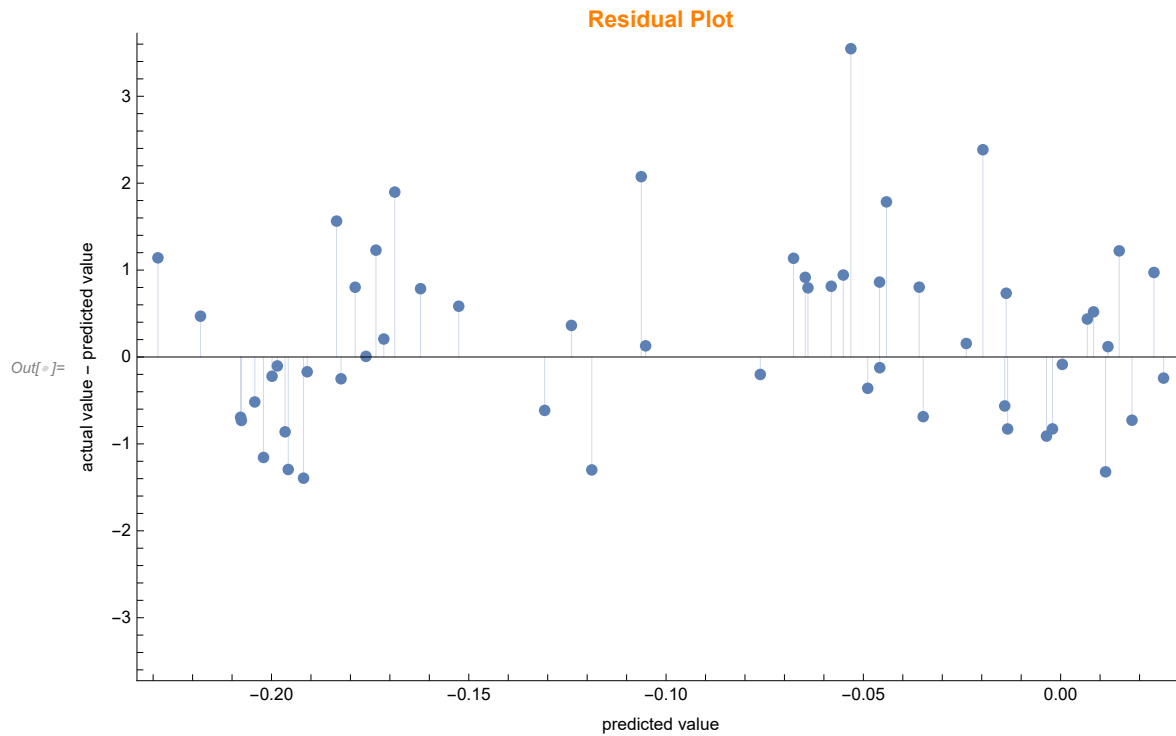
Out[]:= PredictorMeasurementsObject [

+

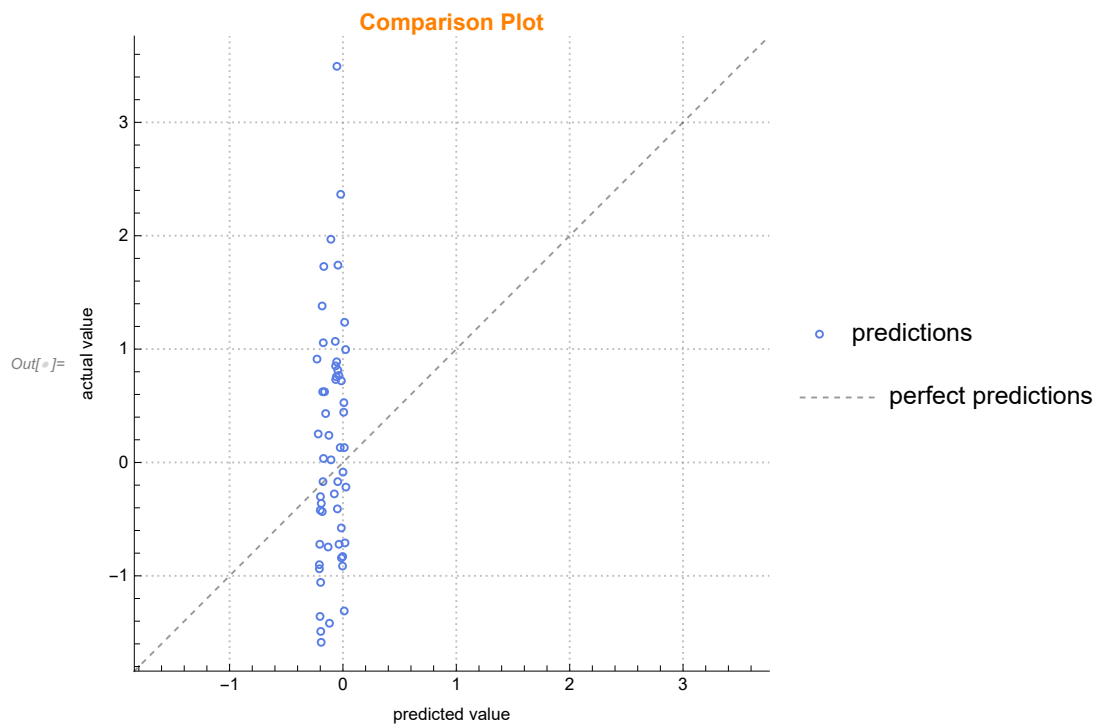
Predictor: NeuralNetwork
Number of test examples: 55

Data not in notebook; Store now »

```
In[ ]:= Show[PredictNeu["ResidualPlot"], ImageSize → Large,  
PlotLabel → Style["Residual Plot", Bold, 12, Orange]]
```



```
In[ ]:= Show[PredictNeu["ComparisonPlot"], ImageSize → Medium,  
PlotLabel → Style["Comparison Plot", Bold, 12, Orange]]
```



Conclusion

Summary

Below are the results of the models created using different algorithms:

1) Random Forest:

MSE = 0.1769

R-Square = 0.7790

2) Linear Regression:

MSE = 0.2293

R-Square = 0.7300

3) Decision Tree:

MSE = 0.1854

R-Square= 0.8131

4) Gradient Boosted Trees:

MSE = 0.0916

R-Square = 0.8774

5) Nearest Neighbour:

MSE = 0.1854

R-Square = 0.8131

6) Neural Network:

MSE = 1.1005

R-Square = 0.0689

Gradient Boosted Trees has the least Mean Square Error value of 0.0916 and the highest R - square value of 0.8774 as compared to the other models and hence we can say that it is the best model among all and provides the best fit for the data. From above calculations its clear that determining adipose in the body is not an straightforward ask. Our dataset had 14 different body measurements but we can say that the proposed models would have provided better predictions with more body measurements. Moreover, the proposed hybrid modeling is not the only prediction technique that can be employed. One may combine other data mining techniques, such as rough set or genetic algorithms basically to refine the structure of models.

The dataset being used was comparatively small for easier processing of the data and that may be the reason some of the machine learning techniques have not produced output as expected. But surely

there is scope to implement various other techniques taking into other aspects of the data that may lead to better prediction. The possibility to apply the same procedure to combine other methods as are the evolving systems deserves further in-depth research and analysis.

For an instance we have included a future aspect of research and analysis below :

(* Reference has been taken from wikipedia for in-depth understanding of some topics *)

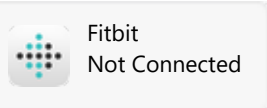
Research Scope:

In this project we have used static data set using the data file that has all the required aesthetics pre filled in it. For future prospects and research purpose we might need latest data set to make interpretation more accurate and meaningful with the changing data. To achieve this we can use dynamic dataset which has several other body composition measurements and we can get that from various activity tracker devices.

For our case we may take data from Fitbit by using **ServiceConnect** function. Fitbit is basically an activity tracker that is worn on the wrist, which basically monitors and track the distance you walk, run, swim or cycle, as well as the other important aesthetics like number of calories you burn.

```
In[ ]:= fitbit = ServiceConnect["Fitbit", "New"]
```

```
Out[ ]:= ServiceObject[
```

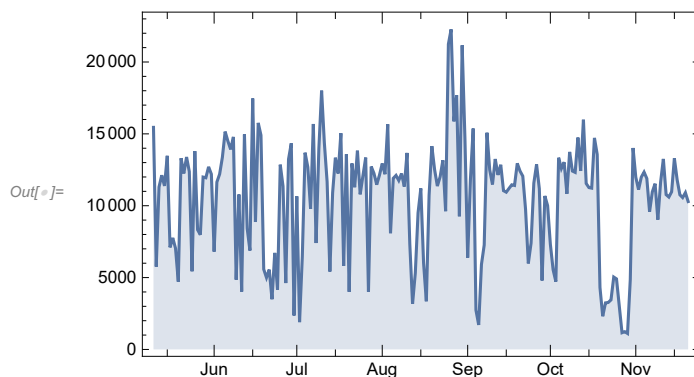


```
ServiceExecute[fitbit, "ActivityData", {"Date" → "Dec. 12, 2018"}]
```

```
Out[ ]:= <|ActivityCalories → 1125, CaloriesBMR → 1732, CaloriesOut → 2464,
FairlyActiveMinutes → 54, LightlyActiveMinutes → 264, MarginalCalories → 636,
SedentaryMinutes → 1114, Steps → 13 298, VeryActiveMinutes → 8|>
```


(*Plot the total steps for each day in a range*)

```
ServiceExecute[fitbit, "StepsPlot",
{"StartDate" → "Feb. 2, 2019", "EndDate" → "May. 13 2019"}]
```




(*Get data of daily calories burned*)

```
calories = ServiceExecute[fitbit, "CaloriesTimeSeries",
  {"StartDate" → "Feb. 2, 2019", "EndDate" → "May. 13 2019"}]
```

Out[]:= TimeSeries [ Time: 3 577 132 800 to 3 593 894 400
Data points: 195]

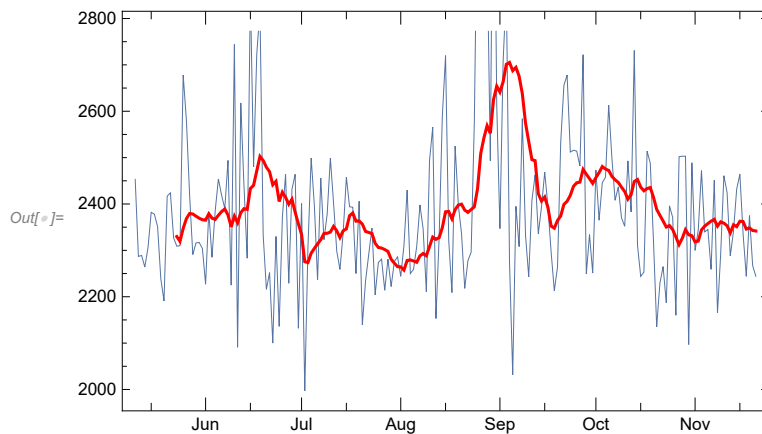
(*Creating average of movemnet*)

```
ln[ ]:= smoothed = MovingMap[Mean, calories, {14, "Day"}]
```

Out[]:= TimeSeries [ Time: 3 578 256 000 to 3 593 894 400
Data points: 182]

(*Time Series Plot of Calories and movement average*)

```
ln[ ]:= DateListPlot[{calories, smoothed}, PlotStyle → {Thin, Red}]
```



Mathematica being modern technical computing system, offers wide range of options for data analysis using data visualization and interpretation using several Machine Learning techniques.

For futuristic advancements in this field we can use dynamic data as shown above that will have better interpretation with accuracy and precision.