

Faculty of Science and Technology

Assignment Coversheet

Student ID number & Student Name	U3242228 Erique
Unit name	Software Technology 1
Unit number	4483
Unit Tutor	Prof. Girija Chetty
Assignment name	ST1 Capstone Project – Semester 1 2023
Due date	12 May at 21:59
Date submitted	11 May 2023

Student declaration

I certify that the attached assignment is my own work. Material drawn from other sources has been appropriately and fully acknowledged as to author/creator, source and other bibliographic details.

Signature of student: __Erique Hew__

Date: _May 11 2023_____

Table of Contents

Introduction	3
Context.....	3
Methodology.....	5
Stage 1: Algorithm Design Stage.....	5
Dataset Description.....	6
Mount Google Drive as “/content/drive”	8
Importing Libraries and Loading dataset	8
Understanding Dataset.....	9
Exploratory Data Analysis	12
Stage 2: Algorithm Implementation Stage.....	26
Stage 3: Software Deployment Stage	27
Conclusions	27
References	28
Source	28

Introduction

This report describes the details of Python Capstone Project for ST1 unit within the scope of the project requirements provided in the assignment handout [1]. I have decided to work on the project using a body fat prediction dataset available in Kaggle data repositories [2].

Context

Obesity, associated with having excess body fat, is a critical public health problem that can cause serious diseases. Although a range of techniques for body fat estimation have been developed to assess obesity, these typically involve high-cost tests requiring special equipment. Thus, the accurate prediction of body fat percentage based on easily accessed body measurements is important for assessing obesity and its related diseases. By considering the characteristics of different features (e.g. body measurements), this study investigates the effectiveness of feature extraction for body fat prediction. It evaluates the performance of three feature extraction approaches by comparing four well-known prediction models. Experimental results based on two real-world body fat datasets show that the prediction models perform better on incorporating feature extraction for body fat prediction, in terms of the mean absolute error, standard deviation, root mean square error and robustness. These results confirm that feature extraction is an effective pre-processing step for predicting body fat. In addition, statistical analysis confirms that feature extraction significantly improves the performance of prediction methods. Moreover, the increase in the number of extracted features results in further, albeit slight, improvements to the prediction models. The findings of this study provide a baseline for future research in related areas [10]

A variety of popular health books suggest that the readers assess their health, at least in part, by estimating their percentage of body fat. In Bailey (1994), for instance, the reader can estimate body fat from tables using their age and various skin-fold measurements obtained by using a calliper. Other texts give predictive equations for body fat using body circumference measurements (e.g. abdominal circumference) and/or skin-fold measurements. See, for instance, Behnke and Wilmore (1974), pp. 66-67; Wilmore (1976), p. 247; or Katch and McArdle (1977), pp. 120-132).

The percentage of body fat for an individual can be estimated once body density has been determined. Folks (e.g. Siri (1956)) assume that the body consists

of two components - lean body tissue and fat tissue. Letting:

- D = Body Density (gm/cm^3)
- A = proportion of lean body tissue
- B = proportion of fat tissue ($A+B=1$)
- a = density of lean body tissue (gm/cm^3)
- b = density of fat tissue (gm/cm^3)

we have:

- $D = 1/[(A/a) + (B/b)]$
- solving for B we find:
- $B = (1/D) * [ab/(a-b)] - [b/(a-b)]$.
- Using the estimates $a=1.10 \text{ gm/cm}^3$ and $b=0.90 \text{ gm/cm}^3$ (see Katch and McArdle (1977), p. 111 or Wilmore (1976), p. 123) we come up with "Siri's equation":
- Percentage of Body Fat (i.e. $100*B$) = $495/D - 450$.

Volume, and hence body density, can be accurately measured in a variety of ways. The technique of underwater weighing "computes body volume as the difference between body weight measured in air and weight measured during water submersion. In other words, body volume is equal to the loss of weight in

water with the appropriate temperature correction for the water's density" (Katch and McArdle (1977), p. 113). Using this technique,

$$\text{Body Density} = \frac{WA}{[(WA-WW)/c.f. - LV]}$$

where:

- WA = Weight in the air (kg)
- WW = Weight in water (kg)
- $c.f.$ = Water correction factor ($=1$ at 39.2 deg F as one gram of water occupies exactly one cm^3 at this temperature, $=.997$ at $76-78 \text{ deg F}$)
- LV = Residual Lung Volume (litres)

(Katch and McArdle (1977), p. 115). Other methods of determining body volume are given in Behnke and Wilmore (1974), p. 22 ff.

This report presents the details of a prototype software platform, in terms of several Python software tools developed as part of this capstone project, based on a data-driven scientific approach, involving exploratory data analysis, predictive analytics and implementation as a desktop Tkinter application.

Methodology

The methodology used for developing the software platform involves 3 stages as outlined below:

1. Design and development of decision support algorithms based on exploratory data analysis and predictive analytics, for identifying the best-performing algorithm for solving a real-world problem.
2. Implementation of the best-performing algorithm as a desktop Tkinter software tool.
3. Deployment of the tool as a web or cloud-enabled platform tool.

Stage 1: Algorithm Design Stage

Stage 1 is the most important preliminary stage and depending on the complexity of the problem and dataset used, the design of algorithms for exploratory data analysis and predictive analytics algorithms will vary. However, the workflow for algorithm development will be as outlined in the Figure 1 schematic shown below:

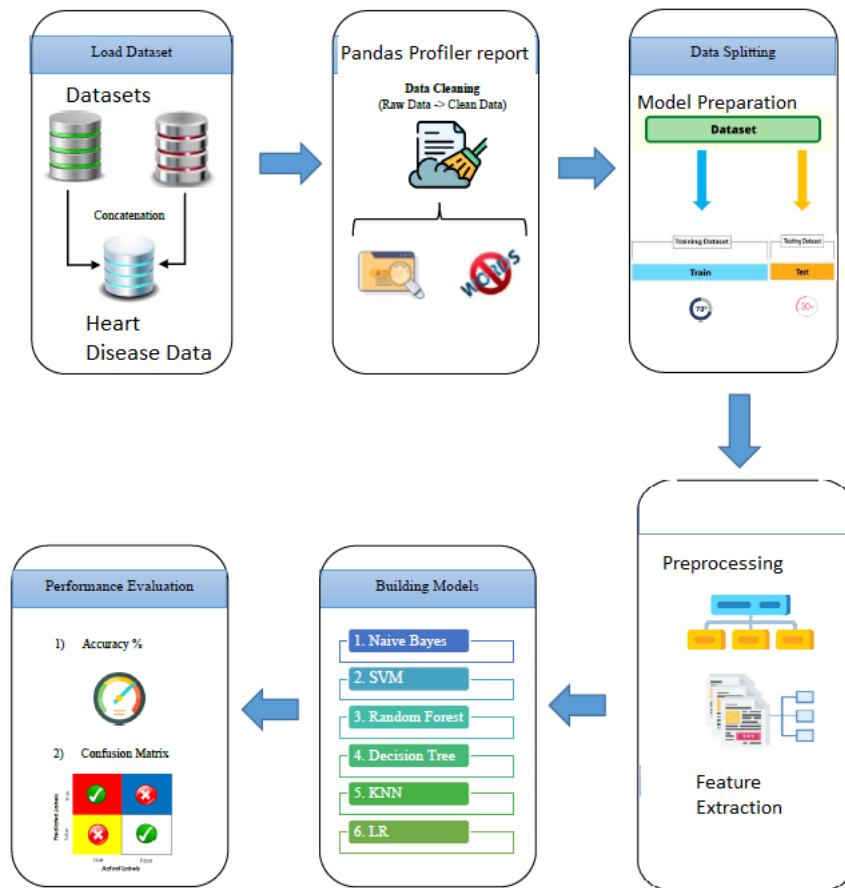
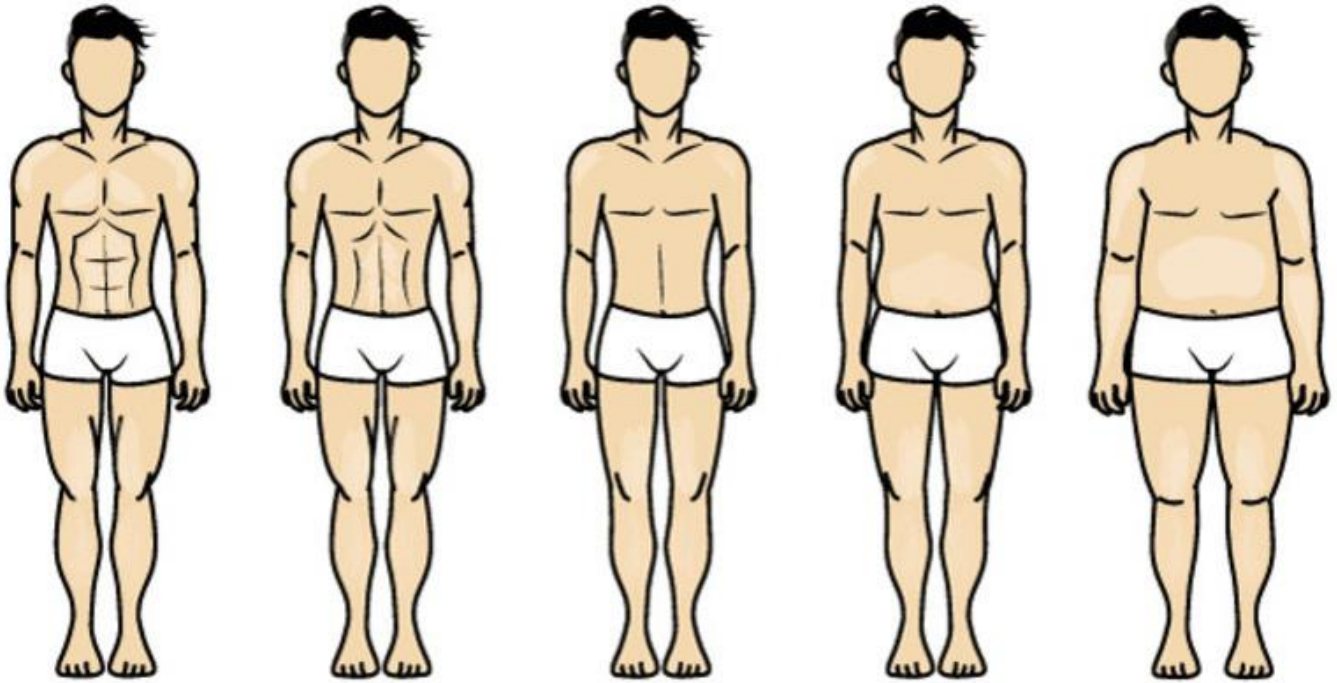


Figure 1: Schematic for Algorithm Design Methodology for Body Fat Prediction

The details of each building block in Figure 1 schematic for algorithm design is described in the follofing.

Dataset Description



There is only one dataset used for this project and it is publicly available in Kaggle [2]. The dataset consists of 252 observations and 16 attributes.

This dataset can be used to illustrate multiple regression techniques. Accurate measurement of body fat is inconvenient/costly and it is desirable to have easy methods of estimating body fat that are not inconvenient/costly.

Lists estimates of the percentage of body fat determined by underwater weighing and various body circumference measurements for 252 men.

Content

The variables listed below, from left to right, are:

1. Age Group
2. Density determined from underwater weighing
3. Percent body fat from Siri's (1956) equation
4. Age (years)
5. Weight (lbs)
6. Height (inches)
7. Neck circumference (cm)
8. Chest circumference (cm)
9. Abdomen 2 circumference (cm)
10. Hip circumference (cm)
11. Thigh circumference (cm)

12. Knee circumference (cm)
13. Ankle circumference (cm)
14. Biceps (extended) circumference (cm)
15. Forearm circumference (cm)
16. Wrist circumference (cm)

(Measurement standards are apparently those listed in Benhke and Wilmore (1974), pp. 45-48 where, for instance, the abdomen 2 circumference is measured "laterally, at the level of the iliac crests, and anteriorly, at the umbilicus".)

These data are used to produce the predictive equations for lean body weight given in the abstract "Generalized body composition prediction equation for men using simple measurement techniques", K.W. Penrose, A.G. Nelson, A.G. Fisher, FACSM, Human Performance Research Center, Brigham Young University, Provo, Utah 84602 as listed in Medicine and Science in Sports and Exercise, vol. 17, no. 2, April 1985, p. 189. (The predictive equation was obtained from the first 143 of the 252 cases that are listed below).

Exploratory Data Analysis

The first phase of the software development activity involved understanding the data, basic exploratory data analysis and visualisation. Google Colab was chosen as the experimental environment as it incorporates virtual hardware and resources which does not require additional physical hardware requirement that can be run directly from a web browser. The Python language was used to create the scripts which ran directly on the online Jupyter Notebook using Google Colab with the help of a free Google account, then saving all the notebook files virtually on google drive without additional configurations. Before the exploratory data analysis can begin, some of the Python libraries for EDA need to be imported and the dataset acquired, by using the following Python script

Mount Google Drive as “/content/drive”

```
from google.colab import drive
drive.mount("/content/drive")
```

Importing Libraries and Loading dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from sklearn.preprocessing import PowerTransformer
```



```

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression , ElasticNet , Lasso ,
Ridge
from sklearn.metrics import r2_score
from sklearn.svm import SVR
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import SGDRegressor
from sklearn.linear_model import BayesianRidge
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from lightgbm import LGBMRegressor
from sklearn.kernel_ridge import KernelRidge
plt.style.use('fivethirtyeight')
colors=['#ffcd94','#eac086','#ffad60','#ffe39f']
sns.set_palette(sns.color_palette(colors))

#Read the dataset/s

df = pd.read_csv('/content/drive/....../heart.csv')

```

Understanding Dataset

#1. Checking description(first 5 and last 5 rows)

```
df.head()
```

	AgeGroup	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist
0	YoungAge	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	YoungAge	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	YoungAge	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	YoungAge	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	YoungAge	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

```
df.tail() #last 5 rows
```

	AgeGroup	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist
247	OldAge	1.0736	11.0	70	134.25	67.00	34.9	89.2	83.6	88.8	49.6	34.8	21.5	25.6	25.7	18.5
248	OldAge	1.0236	33.6	72	201.00	69.75	40.9	108.5	105.0	104.5	59.6	40.8	23.2	35.2	28.6	20.1
249	OldAge	1.0328	29.3	72	186.75	66.00	38.9	111.1	111.5	101.7	60.3	37.3	21.5	31.3	27.2	18.0
250	OldAge	1.0399	26.0	72	190.75	70.50	38.9	108.3	101.3	97.8	56.0	41.6	22.7	30.5	29.4	19.8
251	OldAge	1.0271	31.9	74	207.50	70.00	40.8	112.4	108.5	107.1	59.3	42.2	24.6	33.7	30.0	20.9

#rows and columns-data shape(attributes & samples)

df.shape

```
df.shape
```

```
(252, 16)
```

name of the attributes

df.columns

```
Index(['AgeGroup', 'Density', 'BodyFat', 'Age', 'Weight', 'Height',  
'Neck', 'Chest', 'Abdomen', 'Hip', 'Thigh', 'Knee', 'Ankle', 'Biceps',  
'Forearm', 'Wrist'], dtype='object')
```

```
df.columns
```

```
Index(['AgeGroup', 'Density', 'BodyFat', 'Age', 'Weight', 'Height', 'Neck',  
      'Chest', 'Abdomen', 'Hip', 'Thigh', 'Knee', 'Ankle', 'Biceps',  
      'Forearm', 'Wrist'],  
      dtype='object')
```

#unique values for each attribute

df.nunique()

```
df.nunique()
```

AgeGroup	4
Density	218
BodyFat	176
Age	51
Weight	197
Height	48
Neck	90
Chest	174
Abdomen	185
Hip	152
Thigh	139
Knee	90
Ankle	61
Biceps	104
Forearm	77
Wrist	44
dtype:	int64

#Complete info about data frame

df.info()



df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252 entries, 0 to 251
Data columns (total 16 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   AgeGroup    252 non-null    object
1   Density     252 non-null    float64
2   BodyFat     252 non-null    float64
3   Age         252 non-null    int64
4   Weight      252 non-null    float64
5   Height      252 non-null    float64
6   Neck        252 non-null    float64
7   Chest       252 non-null    float64
8   Abdomen     252 non-null    float64
9   Hip         252 non-null    float64
10  Thigh       252 non-null    float64
11  Knee        252 non-null    float64
12  Ankle       252 non-null    float64
13  Biceps      252 non-null    float64
14  Forearm     252 non-null    float64
15  Wrist       252 non-null    float64
dtypes: float64(14), int64(1), object(1)
memory usage: 31.6+ KB
```

#Analysis on the numerical columns

df.describe()

	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist
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	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	32.273413	28.663889	18.229762
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	3.021274	2.020691	0.933585
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	24.800000	21.000000	15.800000
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	30.200000	27.300000	17.600000
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	32.050000	28.700000	18.300000
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	34.325000	30.000000	18.800000
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	45.000000	34.900000	21.400000

#Check for null values

df.isnull().sum()

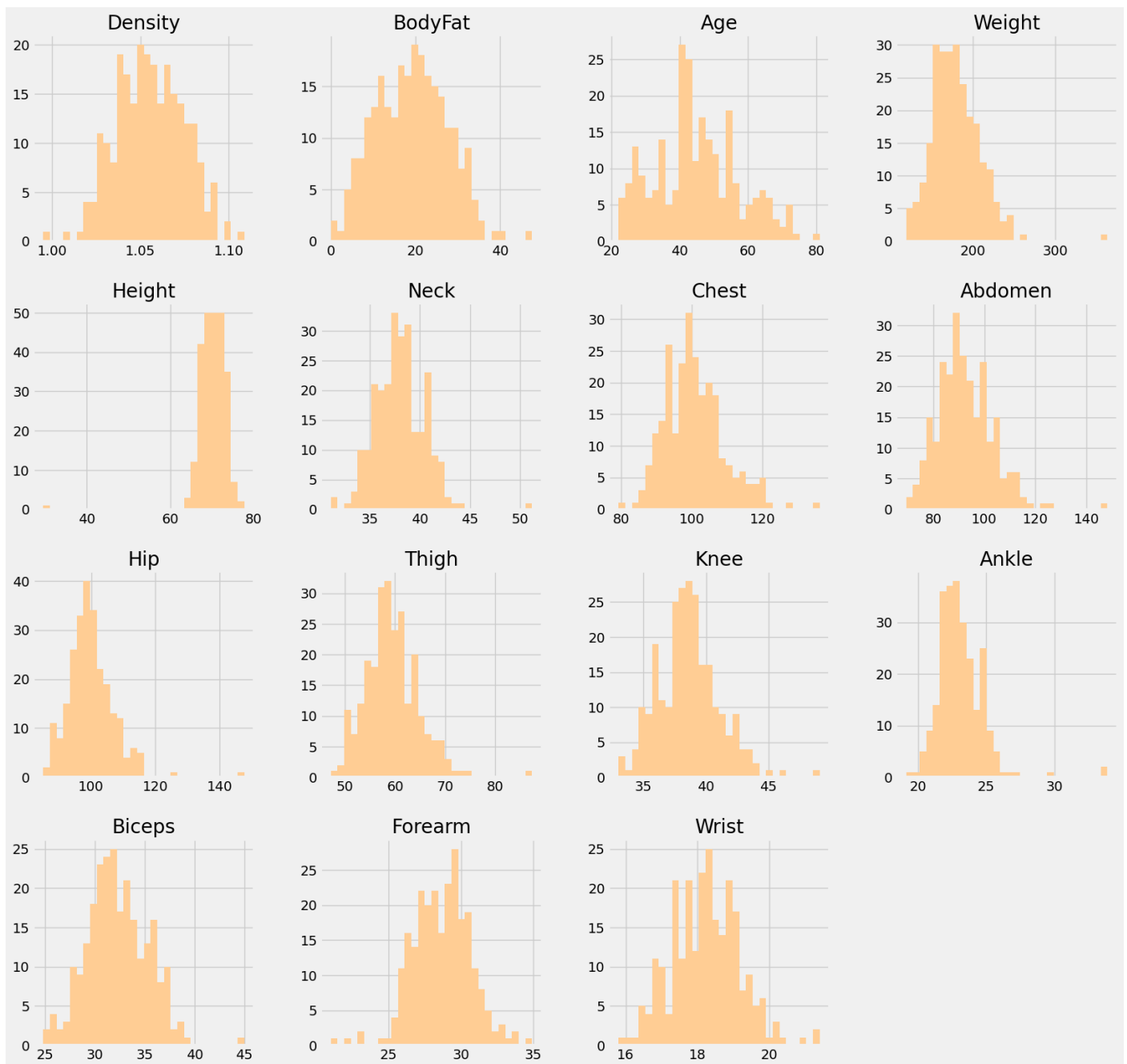
```
df.isnull().sum()
AgeGroup      0
Density        0
BodyFat        0
Age            0
Weight         0
Height         0
Neck           0
Chest          0
Abdomen        0
Hip            0
Thigh          0
Knee           0
Ankle          0
Biceps         0
Forearm        0
Wrist          0
dtype: int64
```

```
#Check for duplicates in the dataset
df.duplicated().sum()
```

```
df.duplicated().sum()
0
```

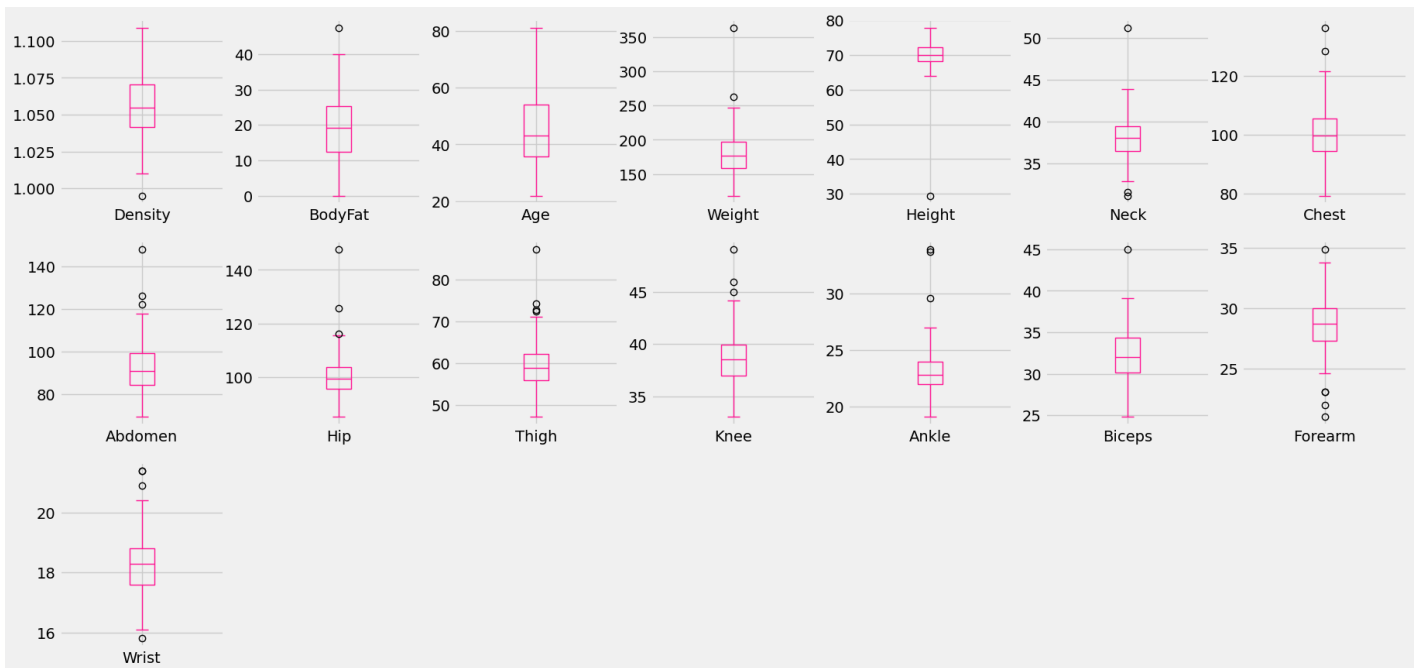
Exploratory Data Analysis

```
#Visualising data distribution in detail
fig = plt.figure(figsize =(18,18))
ax=fig.gca()
df.hist(ax=ax,bins =30)
plt.show()
```



```
#Detecting Outliers
```

```
df.plot(kind='box', subplots=True, layout=(3,7), sharex=False, sharey=False,
figsize=(20, 10), color='deeppink');
```



```
# Identify the outliers
# define continuous variable & plot
continous_features = ['Density', 'BodyFat', 'Age', 'Weight', 'Chest']
def outliers(df_out, drop = False):
    for each_feature in df_out.columns:
        feature_data = df_out[each_feature]
        Q1 = np.percentile(feature_data, 25.) # 25th percentile of the d
        Q3 = np.percentile(feature_data, 75.) # 75th percentile of the d
        IQR = Q3-Q1 #Interquartile Range
        outlier_step = IQR * 1.5 #That's we were talking about above
        outliers = feature_data[~((feature_data >= Q1 - outlier_step) &
        (feature_data <= Q3 + outlier_step))].index.tolist()
        if not drop:
            print('For the feature {}, No of Outliers is {}'.format(each
            _feature, len(outliers)))
        if drop:
            df.drop(outliers, inplace = True, errors = 'ignore')
            print('Outliers from {} feature removed'.format(each_feature
            ))
    outliers(df[continous_features])
```

```
Outliers from Age feature removed
Outliers from Weight feature removed
Outliers from Height feature removed
Outliers from Neck feature removed
Outliers from Chest feature removed
```

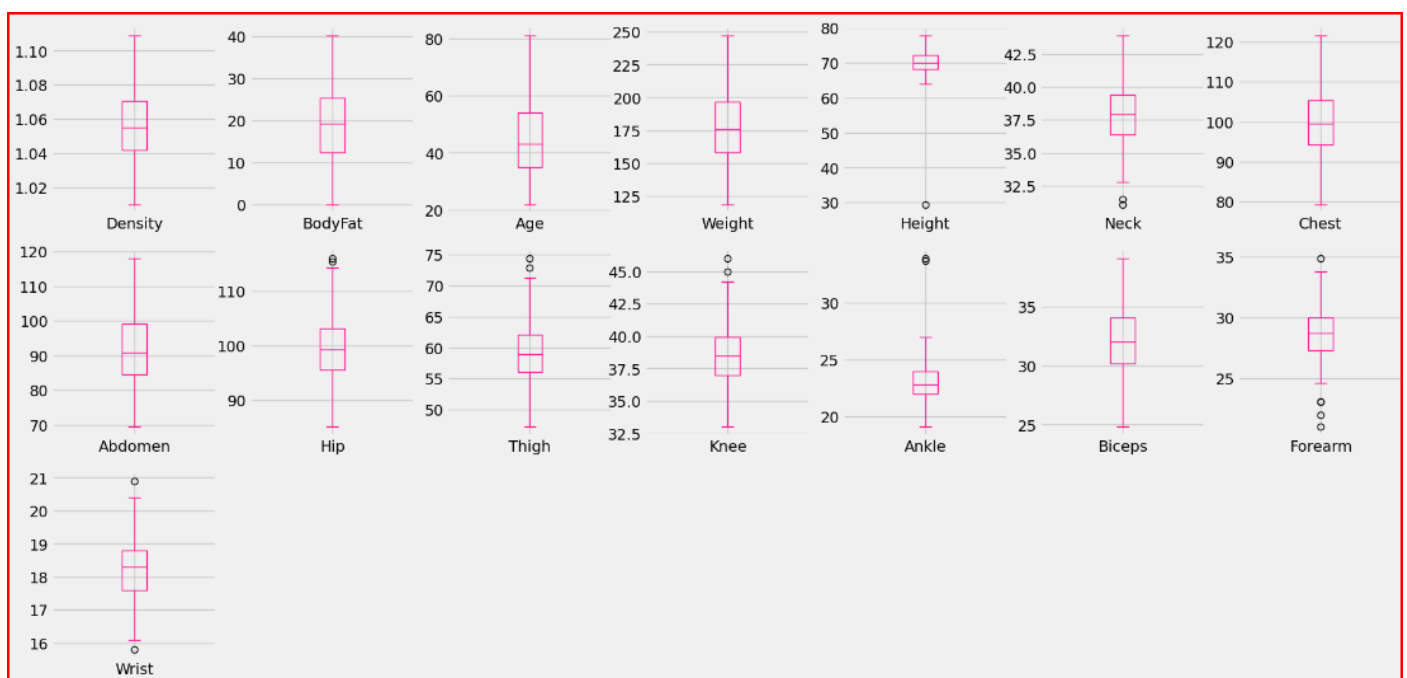
```
#drop the outliers
```

```
outliers(df[continous_features], drop = True)
```

```
Outliers from Density feature removed
Outliers from BodyFat feature removed
Outliers from Age feature removed
Outliers from Weight feature removed
Outliers from Chest feature removed
```

```
#check if outliers got removed
```

```
df.plot(kind='box', subplots=True, layout=(3,7), sharex=False, sharey=False,
figsize=(20, 10), color='deeppink');
```



```
#Check data shape after outlier removal
```

```
df.shape
```

```
(249, 16)
```

```
#checking target value distribution
```

```
#This checking is irrelevant for Body Fat Prediction because the dataset has no result
```

```
print(df.AgeGroup.value_counts())
```

```
fig, ax = plt.subplots(figsize=(5,4))
```

```
name = ["MiddleAge", "SeniorAge", "YoungAge", "OldAge"]
```

```
ax = df.AgeGroup.value_counts().plot(kind='bar')
```

```
ax.set_title("Age Category", fontsize = 13, weight = 'bold')
```

```
ax.set_xticklabels (name, rotation = 0)
```

```

# To calculate the percentage
totals = []
for i in ax.patches:
    totals.append(i.get_height())
total = sum(totals)
for i in ax.patches:
    ax.text(i.get_x()+.09, i.get_height()-50, \
            str(round((i.get_height()/total)*100, 2))+'%', fontsize=14,
            color='white', weight = 'bold')

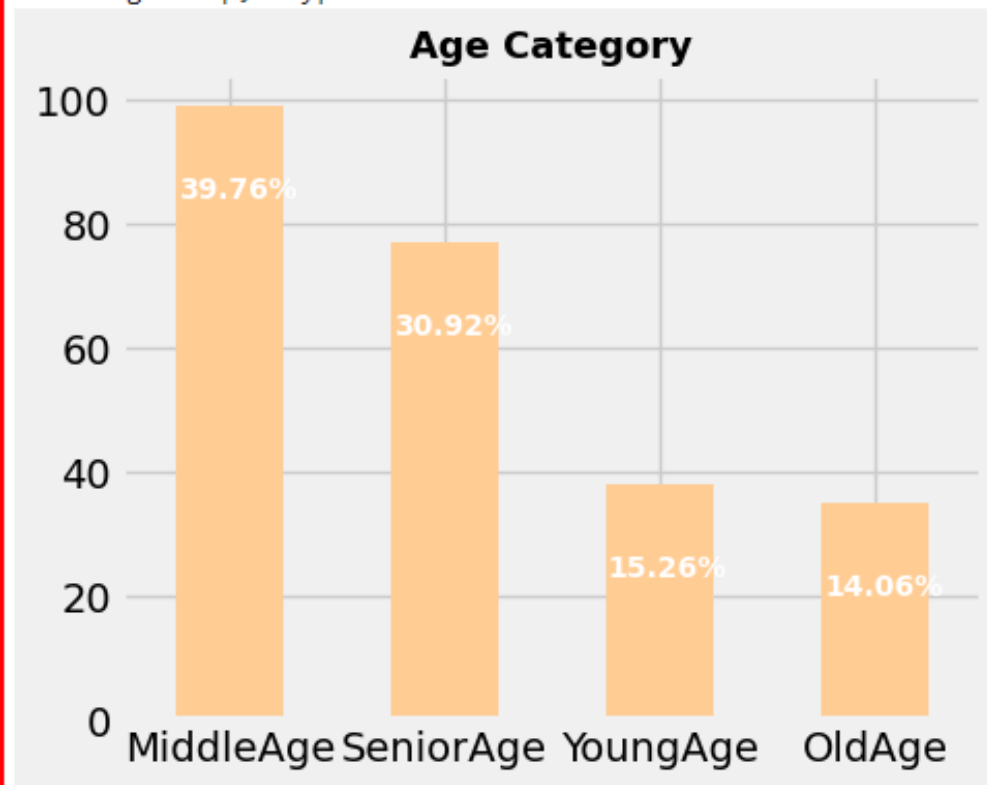
plt.tight_layout()

```

```

MiddleAge    99
SeniorAge    77
YoungAge     38
OldAge       35
Name: AgeGroup, dtype: int64

```

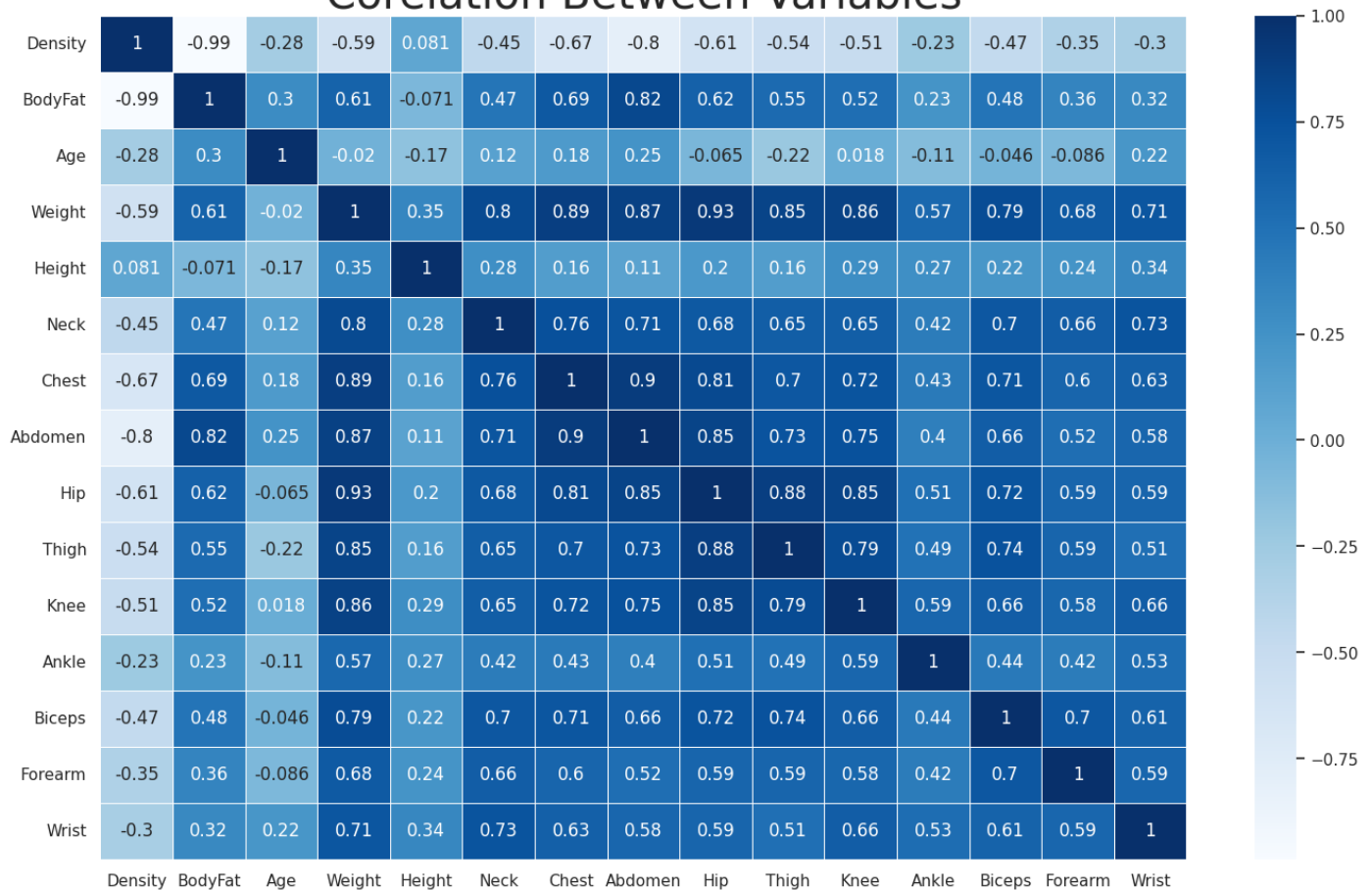


```

#check correlation between variables
sns.set(style="white")
plt.rcParams['figure.figsize'] = (15, 10)
sns.heatmap(df.corr(), annot = True, linewidths=.5, cmap="Blues")
plt.title('Correlation Between Variables', fontsize = 30)
plt.show()

```


Correlation Between Variables



#Install Pandas profiling report

!pip install <https://github.com/pandas-profiling/pandas-profiling/archive/master.zip>

```
[57] #Install Pandas profiling report
!pip install https://github.com/pandas-profiling/pandas-profiling/archive/master.zip

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting https://github.com/pandas-profiling/pandas-profiling/archive/master.zip
  Downloading https://github.com/pandas-profiling/pandas-profiling/archive/master.zip (22.6 MB)
    22.6/22.6 MB 21.6 MB/s eta 0:00:00
```

Obtain full profiler report

```
import pandas as pd
import numpy as np
from pandas_profiling import ProfileReport
profile = ProfileReport(df,title="Body Fat Prediction",
                        html={ 'style':{ 'full_width':True} })
profile.to_notebook_iframe()
```

```
<ipython-input-58-b99648625c27>:3: DeprecationWarning: `import pandas_profiling` is going to be deprecated by April 1st. Please use `import ydat
from pandas_profiling import ProfileReport
Summarize dataset: 100% ██████████ 250/250 [01:02<00:00, 2.25it/s, Completed]
Generate report structure: 100% ██████████ 1/1 [00:12<00:00, 12.26s/it]
Render HTML: 100% ██████████ 1/1 [00:06<00:00, 6.48s/it]
```

Body Fat Prediction

Overview

Variables

Interactions

Correlations

Missing values

Sample

Overview

Alerts 16

Reproduction

Dataset statistics

Number of variables	16
Number of observations	249
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	33.1 KiB
Average record size in memory	136.0 B

Variable types

Categorical	1
Numeric	15

Predictive Data Analytics Stage

For predictive analytics, several processing steps are required. These include pre-processing, classifier comparison to identify the best machine learning classifier and performance evaluation with different objective metrics, such as accuracy, classification report, confusion matrix, ROC-AUC curve and prediction report obtained using the Python scikit-learn package. Each of these steps is described next.

- Pre-processing: Since the dataset consists of a combination of continuous and categorical attributes/variables, there is a need to pre-process the data with attribute transformation, standardization and normalisation. We used scikit-learn's `OrdinalEncoder()` function to perform attribute transformation.
- Normalisation of the independent values of the dataframe by was done by dropping the target from the dataframe, normalising it, and then reattaching the target to the dataframe:-

```
#pre-processing
from sklearn.exceptions import DataDimensionalityWarning
#encode object columns to integers
from sklearn import preprocessing
from sklearn.preprocessing import OrdinalEncoder

for col in df:
    if df[col].dtype == 'object':
```

```
df[col]=OrdinalEncoder().fit_transform(df[col].values.reshape(-1,1))
df
```

	AgeGroup	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist
0	3.0	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	3.0	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	3.0	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	3.0	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	3.0	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
...
247	1.0	1.0736	11.0	70	134.25	67.00	34.9	89.2	83.6	88.8	49.6	34.8	21.5	25.6	25.7	18.5
248	1.0	1.0236	33.6	72	201.00	69.75	40.9	108.5	105.0	104.5	59.6	40.8	23.2	35.2	28.6	20.1
249	1.0	1.0328	29.3	72	186.75	66.00	38.9	111.1	111.5	101.7	60.3	37.3	21.5	31.3	27.2	18.0
250	1.0	1.0399	26.0	72	190.75	70.50	38.9	108.3	101.3	97.8	56.0	41.6	22.7	30.5	29.4	19.8
251	1.0	1.0271	31.9	74	207.50	70.00	40.8	112.4	108.5	107.1	59.3	42.2	24.6	33.7	30.0	20.9

249 rows x 16 columns

```
class_label =df['AgeGroup']
df = df.drop(['AgeGroup'], axis =1)
df = (df-df.min())/(df.max()-df.min())
df['AgeGroup']=class_label
df
```

	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist	AgeGroup
0	0.614372	0.306733	0.016949	0.277670	0.792746	0.398438	0.326241	0.325103	0.305466	0.433824	0.330769	0.189189	0.503497	0.460432	0.254902	3.0
1	0.761134	0.152120	0.000000	0.425243	0.886010	0.578125	0.338061	0.279835	0.440514	0.422794	0.330769	0.290541	0.398601	0.568345	0.470588	3.0
2	0.316802	0.630923	0.000000	0.275728	0.761658	0.226562	0.390071	0.380658	0.456592	0.455882	0.453846	0.331081	0.279720	0.302158	0.156863	3.0
3	0.657895	0.259352	0.067797	0.514563	0.886010	0.492187	0.531915	0.349794	0.520900	0.474265	0.330769	0.250000	0.531469	0.604317	0.470588	3.0
4	0.241903	0.715711	0.033898	0.510680	0.865285	0.257812	0.425532	0.629630	0.543408	0.588235	0.707692	0.331081	0.517483	0.482014	0.372549	3.0
...
247	0.642713	0.274314	0.813559	0.122330	0.777202	0.296875	0.234043	0.292181	0.122186	0.088235	0.138462	0.162162	0.055944	0.338129	0.529412	1.0
248	0.136640	0.837905	0.847458	0.640777	0.834197	0.765625	0.690307	0.732510	0.627010	0.455882	0.600000	0.277027	0.727273	0.546763	0.843137	1.0
249	0.229757	0.730673	0.847458	0.530097	0.756477	0.609375	0.751773	0.866255	0.536977	0.481618	0.330769	0.162162	0.454545	0.446043	0.431373	1.0
250	0.301619	0.648379	0.847458	0.561165	0.849741	0.609375	0.685579	0.656379	0.411576	0.323529	0.661538	0.243243	0.398601	0.604317	0.784314	1.0
251	0.172065	0.795511	0.881356	0.691262	0.839378	0.757812	0.782506	0.804527	0.710611	0.444853	0.707692	0.371622	0.622378	0.647482	1.000000	1.0

249 rows x 16 columns

#pre-processing

```
le = preprocessing.LabelEncoder()
agegroup = le.fit_transform(list(df["AgeGroup"]))
density = le.fit_transform(list(df["Density"]))
bodyfat = le.fit_transform(list(df["BodyFat"]))
age = le.fit_transform(list(df["Age"]))
weight = le.fit_transform(list(df["Weight"]))
height = le.fit_transform(list(df["Height"]))
neck = le.fit_transform(list(df["Neck"]))
chest = le.fit_transform(list(df["Chest"]))
abdomen = le.fit_transform(list(df["Abdomen"]))
hip = le.fit_transform(list(df["Hip"]))
thigh = le.fit_transform(list(df["Thigh"]))
```

```
knee = le.fit_transform(list(df["Knee"]))
ankle = le.fit_transform(list(df["Ankle"]))
biceps = le.fit_transform(list(df["Biceps"]))
forearm = le.fit_transform(list(df["Forearm"]))
wrist = le.fit_transform(list(df["Wrist"]))
```

Model Preparation and Development

- Convert the dataframe to training and validation/test subsets by taking a random sample of 80% of the data and defining it as train subset. This leaves 20% of the data for validation/testing
- Create the validation/test set by dropping all of the rows that comprise the training set from the dataframe.
- Create y_train by using using the last column of train (target class).
- Create x_train by using all of the columns in train except the last one.
- The validation set of y_val and x_val or (y_test and x_test), can be created using the same methodology that used to create y_train and x_train

```
x = list(zip(bodyfat, age))
y = list(agegroup)
# Test options and evaluation metric
num_folds = 5
seed = 7
scoring = 'accuracy'

# Model Test/Train
# Splitting what we are trying to predict into 4 different arrays -
# X train is a section of the x array(attributes) and vice versa for Y(features)
# The test data will test the accuracy of the model created
import sklearn.model_selection
x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(x, y, test_size = 0.20, random_state=seed)
#splitting 20% of our data into test samples. If we train the model with higher data it already has seen that information and knows

#size of train and test subsets after splitting
np.shape(x_train), np.shape(x_test)

((199, 2), (50, 2))

# Predictive analytics model development by comparing different Scikit-learn classification algorithms
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score
```

```

from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier

models = []
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
models.append(('GBM', GradientBoostingClassifier()))
models.append(('RF', RandomForestClassifier()))
# evaluate each model in turn
results = []
names = []
print("Performance on Training set")
for name, model in models:
    kfold = KFold(n_splits=num_folds, shuffle=True, random_state=seed)
    cv_results = cross_val_score(model, x_train, y_train, cv=kfold, scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    msg += '\n'
    print(msg)

Performance on Training set
NB: 0.969872 (0.009939)

SVM: 0.683333 (0.051908)

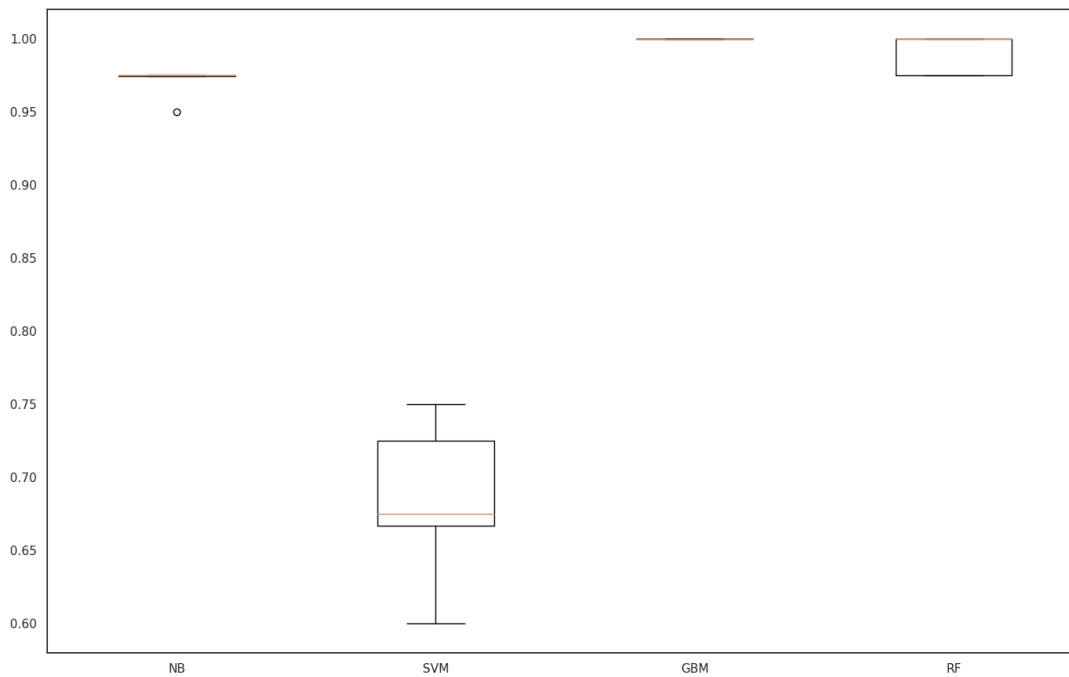
GBM: 1.000000 (0.000000)

RF: 0.990000 (0.012247)

# Compare Algorithms' Performance
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()

```

Algorithm Comparison



```
#Model Evaluation by testing with independent/external test data set.
#Make predictions on validation/test dataset
```

```
#Model Evaluation by testing with independent/external test data set.
#Make predictions on validation/test dataset
```

```
models.append(('DT', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
models.append(('GBM', GradientBoostingClassifier()))
models.append(('RF', RandomForestClassifier()))
dt = DecisionTreeClassifier()
nb = GaussianNB()
gb = GradientBoostingClassifier()
rf = RandomForestClassifier()

best_model = gb
best_model.fit(x_train, y_train)
y_pred = best_model.predict(x_test)
print("Best Model Accuracy Score on Test Set:", accuracy_score(y_test, y_pred))
```

Best Model Accuracy Score on Test Set: 1.0

```
#Model Performance Evaluation Metric 1 - Classification Report
print(classification_report(y_test, y_pred))
```

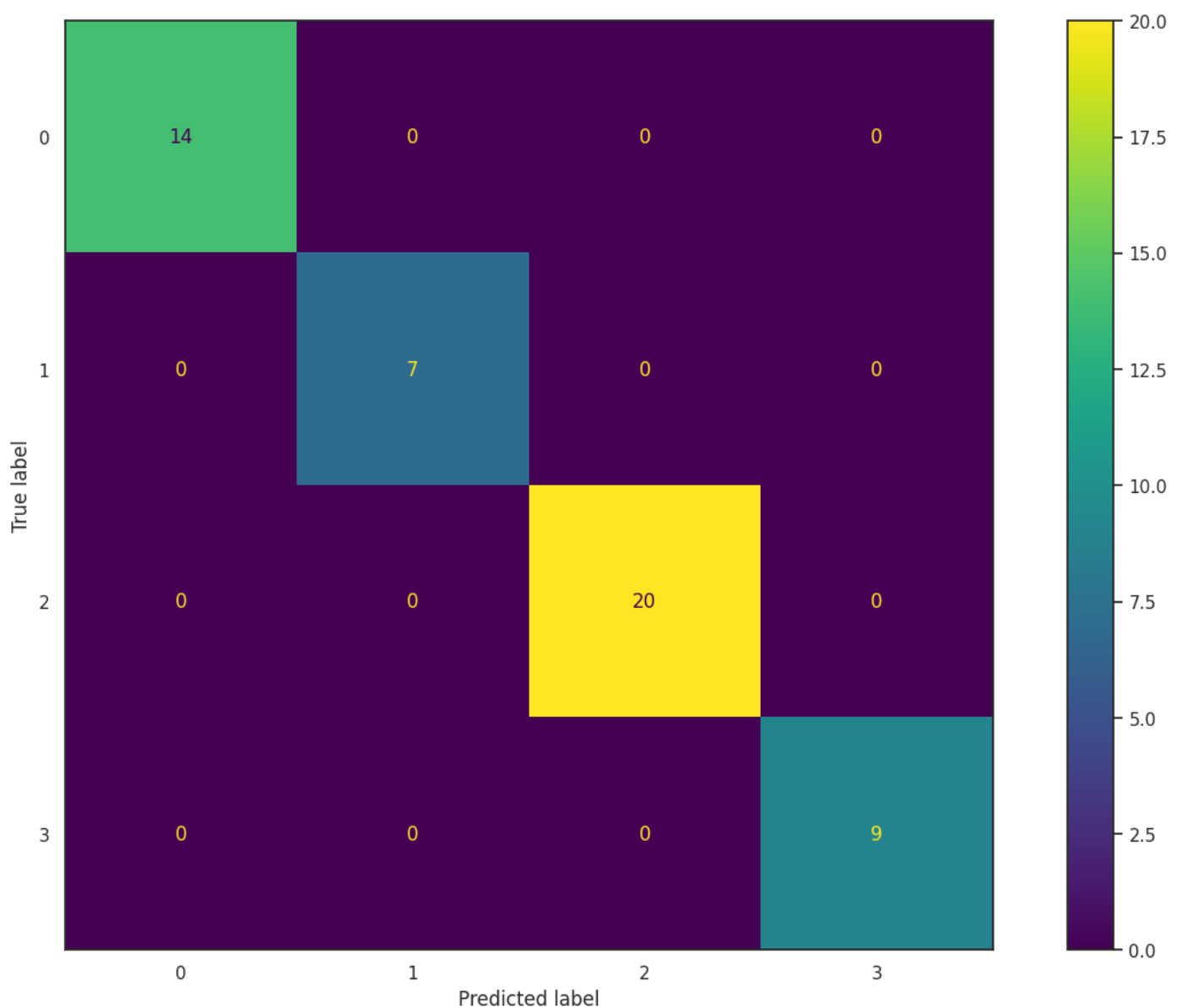
	precision	recall	f1-score	support
0	1.00	1.00	1.00	14

1	1.00	1.00	1.00	7
2	1.00	1.00	1.00	20
3	1.00	1.00	1.00	9
accuracy			1.00	50
macro avg	1.00	1.00	1.00	50
weighted avg	1.00	1.00	1.00	50

#Model Performance Evaluation Metric 2

#Confusion matrix

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```



#Model Evaluation Metric 3- ROC-AUC curve

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.preprocessing import LabelBinarizer
```

```

#Check actual/ground truth vs predicted diagnosis
for x in range(len(y_pred)):
    print("Predicted: ", y_pred[x], "Actual: ", y_test[x], "Data: ",
x_test[x],)

label_binarizer = LabelBinarizer().fit(y_train)
y_onehot_test = label_binarizer.transform(y_test)
y_onehot_test.shape # (n_samples, n_classes)

#ROC curve for class 3 ('YoungAge')
from sklearn.metrics import RocCurveDisplay
class_id = 3
class_of_interest = "Age"

RocCurveDisplay.from_predictions(y_onehot_test[:, class_id], y_pred,
                                name=f"{class_of_interest} vs the
rest", color="darkorange",)
plt.plot([0, 1], [0, 1], "k--", label="chance level (AUC = 0.5)")
plt.axis("square")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("One-vs-Rest ROC curves:\nYoungAge vs Rest(OldAge, MiddleAge,
SeniorAge )")
plt.legend()
plt.show()

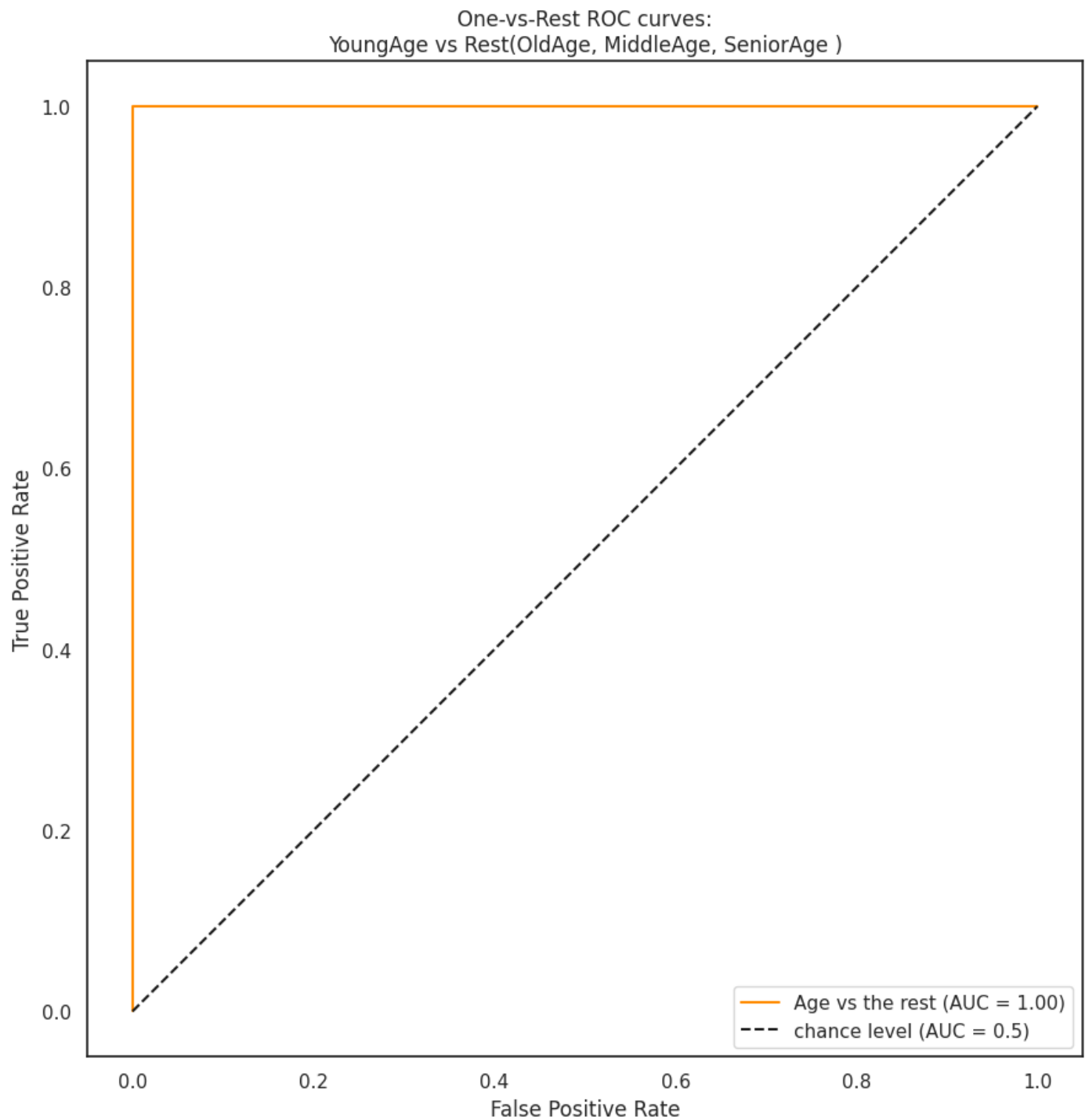
```

```

Predicted: 3 Actual: 3 Data: (26, 4)
Predicted: 2 Actual: 2 Data: (40, 32)
Predicted: 0 Actual: 0 Data: (27, 16)
Predicted: 1 Actual: 1 Data: (62, 48)
Predicted: 3 Actual: 3 Data: (33, 4)
Predicted: 2 Actual: 2 Data: (69, 25)
Predicted: 2 Actual: 2 Data: (110, 25)
Predicted: 1 Actual: 1 Data: (152, 42)
Predicted: 0 Actual: 0 Data: (128, 18)
Predicted: 2 Actual: 2 Data: (54, 28)
Predicted: 2 Actual: 2 Data: (93, 28)
Predicted: 3 Actual: 3 Data: (76, 6)
Predicted: 0 Actual: 0 Data: (84, 18)
Predicted: 2 Actual: 2 Data: (21, 24)
Predicted: 1 Actual: 1 Data: (85, 41)
Predicted: 0 Actual: 0 Data: (4, 20)
Predicted: 0 Actual: 0 Data: (67, 9)
Predicted: 0 Actual: 0 Data: (168, 13)
Predicted: 2 Actual: 2 Data: (111, 27)
Predicted: 2 Actual: 2 Data: (66, 36)
Predicted: 0 Actual: 0 Data: (97, 13)
Predicted: 2 Actual: 2 Data: (63, 31)
Predicted: 0 Actual: 0 Data: (140, 22)
Predicted: 2 Actual: 2 Data: (24, 35)
Predicted: 2 Actual: 2 Data: (165, 25)
Predicted: 2 Actual: 2 Data: (20, 29)
Predicted: 2 Actual: 2 Data: (99, 24)
Predicted: 3 Actual: 3 Data: (18, 5)
Predicted: 0 Actual: 0 Data: (142, 12)

```


Predicted:	3	Actual:	3	Data:	(130, 0)
Predicted:	3	Actual:	3	Data:	(56, 6)
Predicted:	1	Actual:	1	Data:	(125, 38)
Predicted:	2	Actual:	2	Data:	(96, 26)
Predicted:	2	Actual:	2	Data:	(103, 27)
Predicted:	2	Actual:	2	Data:	(13, 32)
Predicted:	0	Actual:	0	Data:	(107, 17)
Predicted:	1	Actual:	1	Data:	(139, 48)
Predicted:	3	Actual:	3	Data:	(47, 3)
Predicted:	2	Actual:	2	Data:	(156, 32)
Predicted:	3	Actual:	3	Data:	(48, 8)
Predicted:	1	Actual:	1	Data:	(65, 45)
Predicted:	0	Actual:	0	Data:	(123, 19)
Predicted:	3	Actual:	3	Data:	(6, 3)
Predicted:	1	Actual:	1	Data:	(158, 44)
Predicted:	0	Actual:	0	Data:	(117, 9)
Predicted:	0	Actual:	0	Data:	(123, 19)
Predicted:	2	Actual:	2	Data:	(163, 35)
Predicted:	2	Actual:	2	Data:	(129, 33)
Predicted:	2	Actual:	2	Data:	(54, 33)
Predicted:	0	Actual:	0	Data:	(28, 15)



Stage 2: Algorithm Implementation Stage

1. Implementation of best performing algorithm as a desktop Tkinter software tool.
2. Deployment of the tool as a web or cloud enabled platform tool.

Once the best performing algorithm and machine learning model for heart disease prediction has been identified from stage 1, the implementation of the algorithm as a desktop software tool using the python Tkinter package.

The Pycharm project for the implementation is available at this google drive link:

https://drive.google.com/drive/folders/1-KppRzk7MaVAxT6M0299NXU_pG0CD6u-?usp=sharing

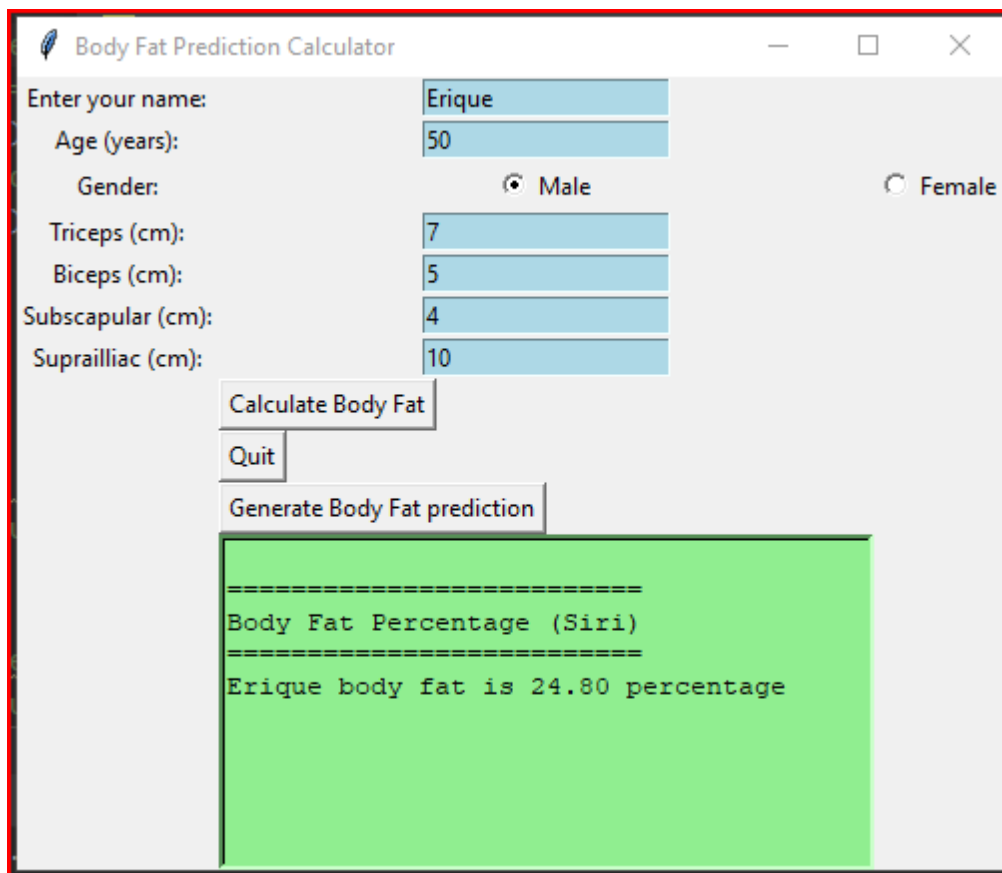
Stage 3: Software Deployment Stage

The deployment of software as a desktop tool as in stage 3, the software application is developed using the python tkinter. The preliminary package that needs to be installed before running the application is called the “fitness-tool” package which is available at <https://fitness-tools.readthedocs.io/en/latest/quickstart.html>.

The application will require 6 inputs which are age, gender, triceps, biceps, subscapular and suprailliac. These inputs will be used to calculate body density value. The result value is in float data type. The module to calculate the body definitely is “DurninWomersley”

The body density value will be used to calculate body fat by using the Siri formula and the value is float.

The interface of this application is using the TKinter and the screenshot is as follows.



The screenshot shows a Tkinter window titled "Body Fat Prediction Calculator". The interface includes input fields for "Enter your name:" (Erique), "Age (years):" (50), "Gender:" (Male selected, Female unselected), "Triceps (cm):" (7), "Biceps (cm):" (5), "Subscapular (cm):" (4), and "Suprailliac (cm):" (10). Below these fields are three buttons: "Calculate Body Fat", "Quit", and "Generate Body Fat prediction". A green text box at the bottom displays the result: "=====
Body Fat Percentage (Siri)
=====
Erique body fat is 24.80 percentage".

Conclusions

This report presents the work done towards the ST1 capstone project for design, development, implementation and deployment of data driven heart disease prediction software platform using Python. This platform allows leveraging the studies of non-invasive clinical tests run on patients, and building a data driven disease prediction platform, where the historical data and insight about a problem or an event can be used to predict the future risk or probability of the event. As can be seen

from the outcomes of this project, we can train models that can predict coronary heart disease with substantial agreement with the results of invasive coronary angiography. The availability of predictive analytics tools as both desktop software tools and a web-based tool allows the wider application of this project to clinical, non-clinical and allied healthcare teams for managing heart disease for the patients.

References

1. "Software Technology 1 (4483)" <https://uclearn.canberra.edu.au/courses/13373/> (accessed May 02, 2023).
2. Roger W. Johnson, "Body Fat Prediction Dataset," Department of Mathematics & Computer Science. South Dakota School of Mines & Technology. 501 East St. Joseph Street. Rapid City, SD 57701. email address: rwjohnso@silver.sdsmt.edu / web address: <http://silver.sdsmt.edu/~rwjohnso>
3. Fitness Tools, "Healthy Lifestyles With Python", 2004 January Apache 2.0 License.<https://fitness-tools.readthedocs.io/en/latest/license-link.html> (accessed May 02, 2023).
4. Bailey, Covert (1994). Smart Exercise: Burning Fat, Getting Fit, Houghton-Mifflin Co., Boston, pp. 179-186.
5. Behnke, A.R. and Wilmore, J.H. (1974). Evaluation and Regulation of Body Build and Composition, Prentice-Hall, Englewood Cliffs, N.J.
6. Siri, W.E. (1956), "Gross composition of the body", in Advances in Biological and Medical Physics, vol. IV, edited by J.H. Lawrence and C.A. Tobias, Academic Press, Inc., New York.
7. Katch, Frank and McArdle, William (1977). Nutrition, Weight Control, and Exercise, Houghton Mifflin Co., Boston.
8. Wilmore, Jack (1976). Athletic Training and Physical Fitness: Physiological Principles of the Conditioning Process, Allyn and Bacon, Inc., Boston.
9. Fan Z, Chiong R, Hu Z, Keivanian F, Chiong F. Body fat prediction through feature extraction based on anthropometric and laboratory measurements. PLoS One. 2022 Feb 22;17(2):e0263333. doi: 10.1371/journal.pone.0263333. PMID: 35192644; PMCID: PMC8863283. (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8863283/>)

Source

The data were generously supplied by Dr. A. Garth Fisher who gave permission to freely distribute the data and use it for non-commercial purposes.

Roger W. Johnson

Department of Mathematics & Computer Science

South Dakota School of Mines & Technology

501 East St. Joseph Street

Rapid City, SD 57701

email address: rwjohnso@silver.sdsmt.edu

web address: <http://silver.sdsmt.edu/~rwjohnso>