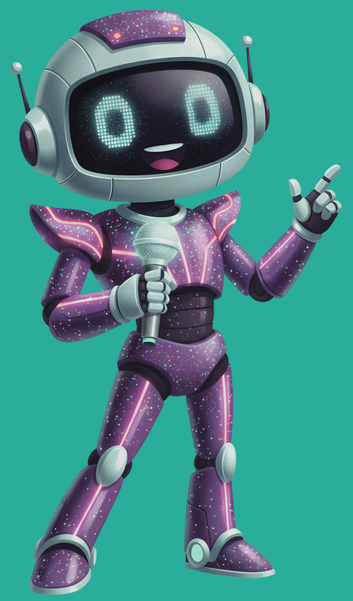
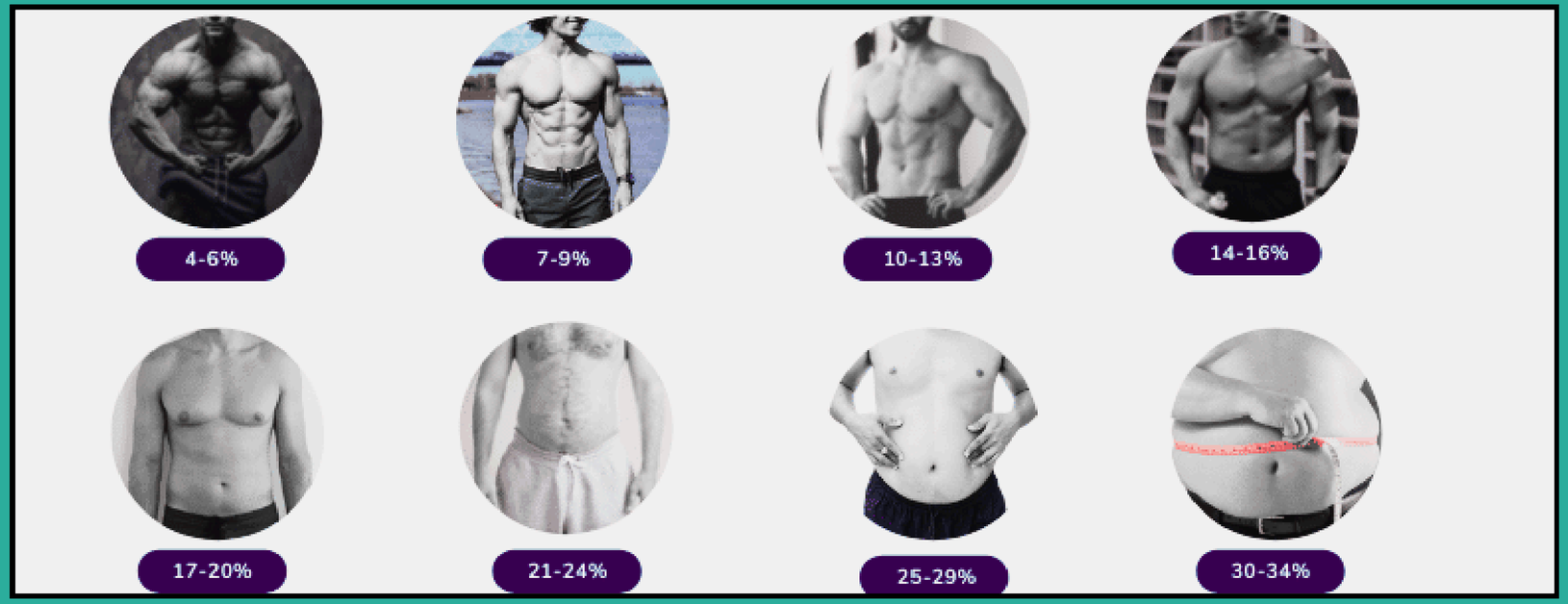
BODY FAT PREDICTION FOR MEN USING AI/ML

PRESENTED BY : RAJARAJESWARI





PROBLEM STATEMENT

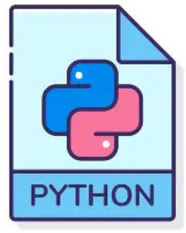
Body fat percentage is a key indicator of health and fitness, but direct measurement methods are expensive and inconvenient. People often rely on BMI, which is inaccurate because it doesn't distinguish between fat and muscle.

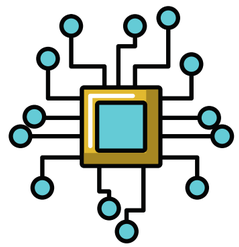
SOLUTION

To develop a machine learning Regression model that predicts body fat percentage using readily available anthropometric and demographic data, such as age, weight, chest circumference, abdominal circumference, and height.

TECHSTACK USED

- DATASET - **KAGGLE**
- PYTHON 3.12
- **SEABORN** to perform univariate and bivariate analysis
- **MACHINE LEARNING REGRESSION MODELS(Linear,SVM,RandomForest,Decision Tree)** to train the dataset
- **django** to develop front end





DATA COLLECTION

- Dataset Lists estimates of the percentage of body fat determined by underwater weighing and various body circumference measurements for 252 men.
- 15 columns and 252 rows

Importing Dataset

```
dataset=pd.read_csv("bodyfat.csv")
dataset
```

	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist
0	1.0708	12.3	23	154.25	67.75	36.2	93.1	85.2	94.5	59.0	37.3	21.9	32.0	27.4	17.1
1	1.0853	6.1	22	173.25	72.25	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	28.9	18.2
2	1.0414	25.3	22	154.00	66.25	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	25.2	16.6
3	1.0751	10.4	26	184.75	72.25	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	29.4	18.2
4	1.0340	28.7	24	184.25	71.25	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	27.7	17.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
247	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	1.0236	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	1.0328	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	1.0399	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	1.0271	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

252 rows × 15 columns

DATA PREPROCESSING

Missing/Null values:

- There are no missing/Null values in the dataset

Outliers:

- Lower bound outliers are available in Density', 'Height', 'Neck', 'Forearm', and 'Wrist' columns
- Upper bound outliers are available in 'BodyFat', 'Weight', 'Neck', 'Chest', 'Abdomen', 'Hip', 'Thigh', 'Knee', 'Ankle', 'Biceps', 'Forearm' and 'Wrist' Columns
- Replaced them with appropriate lower/upper bound range determined using IQR

1.Checking for Missing/null values

```
dataset.isna().sum()
```

```
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
```

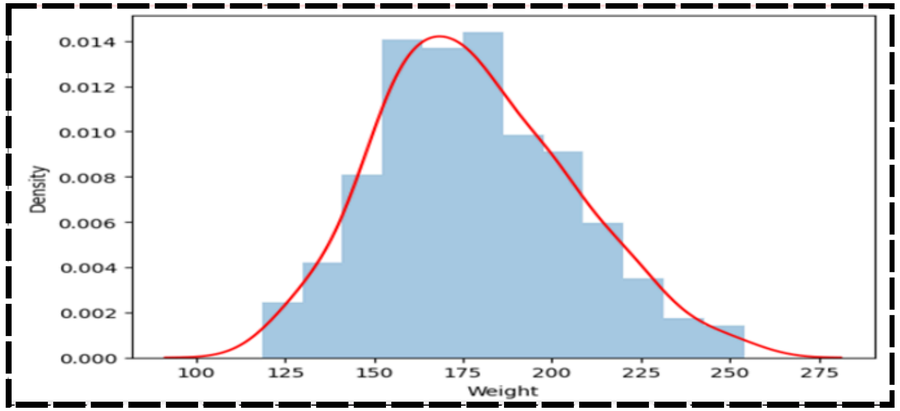
```
# Checking columns which have outliers based on upper and lower range value
lower,upper=Univariate.outliercolumns(quan,descriptive)
print(lower,"\n",upper)
```

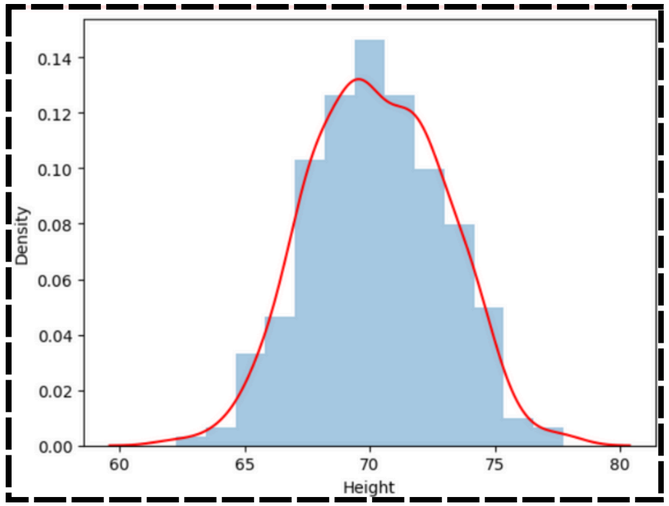
```
['Density', 'Height', 'Neck', 'Forearm', 'Wrist']
 ['BodyFat', 'Weight', 'Neck', 'Chest', 'Abdomen', 'Hip', 'Thigh', 'Knee', 'Ankle', 'Biceps', 'Forearm', 'Wrist']
```

UNIVARIATE ANALYSIS

INSIGHTS (Weight and Height factors)

- Most of the men members weigh from 150 to 200 pounds
- Members with small, medium and large frame are available in the dataset
- Members with different height categories are available. i.e short, average and tall members are available in the dataset

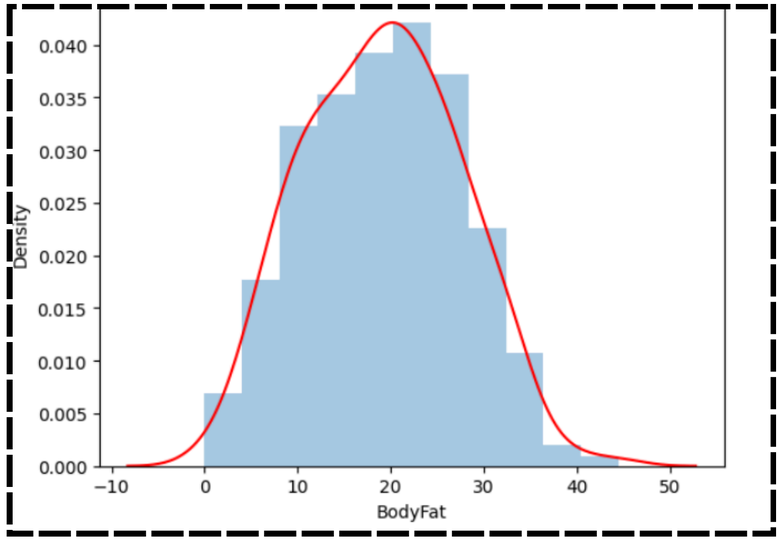


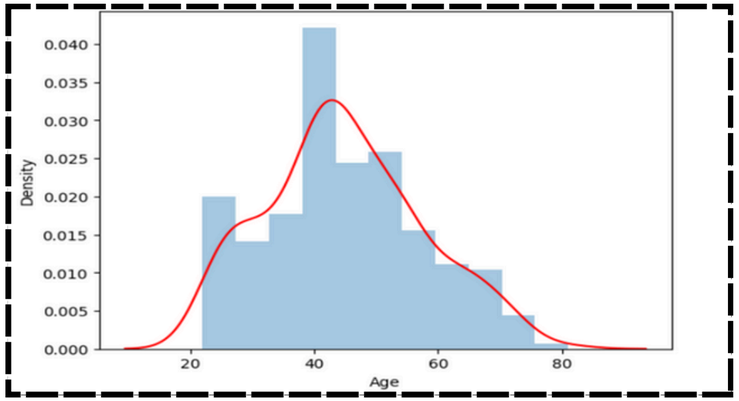


UNIVARIATE ANALYSIS

INSIGHTS(BodyFat and Age)

- The dataset contains participants across low-fat, healthy-fat, and over-fat groups.
- Some members in the dataset are in their seventies
- Most individuals in the dataset are between 40 and 55 years old

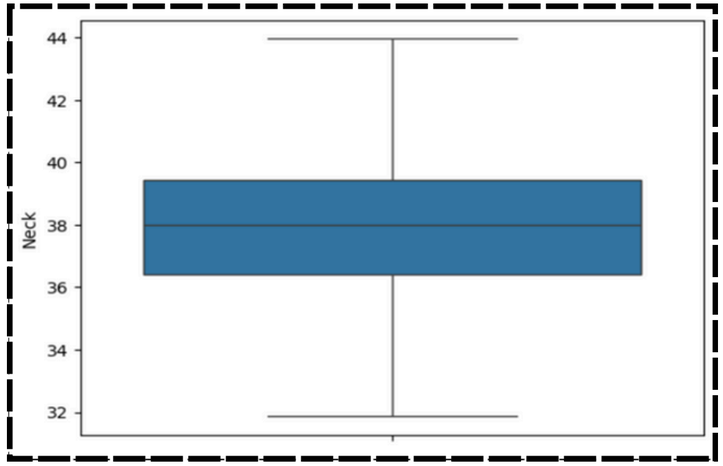


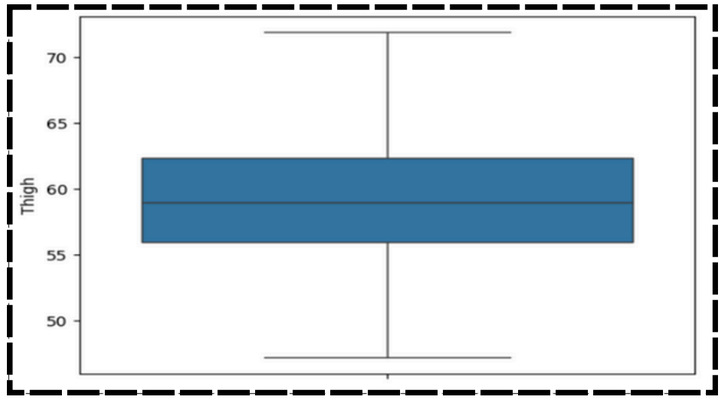


UNIVARIATE ANALYSIS

INSIGHTS(Neck and Thigh)

- Approximately 25% of members have neck measurements near 36.2 (Q1).
- Around 75% of members have neck measurements close to 39.2 (Q3).
- Approximately 25% of members have thigh measurements near 56 (Q1).
- Around 75% of members have thigh measurements close to 62.35 (Q3)

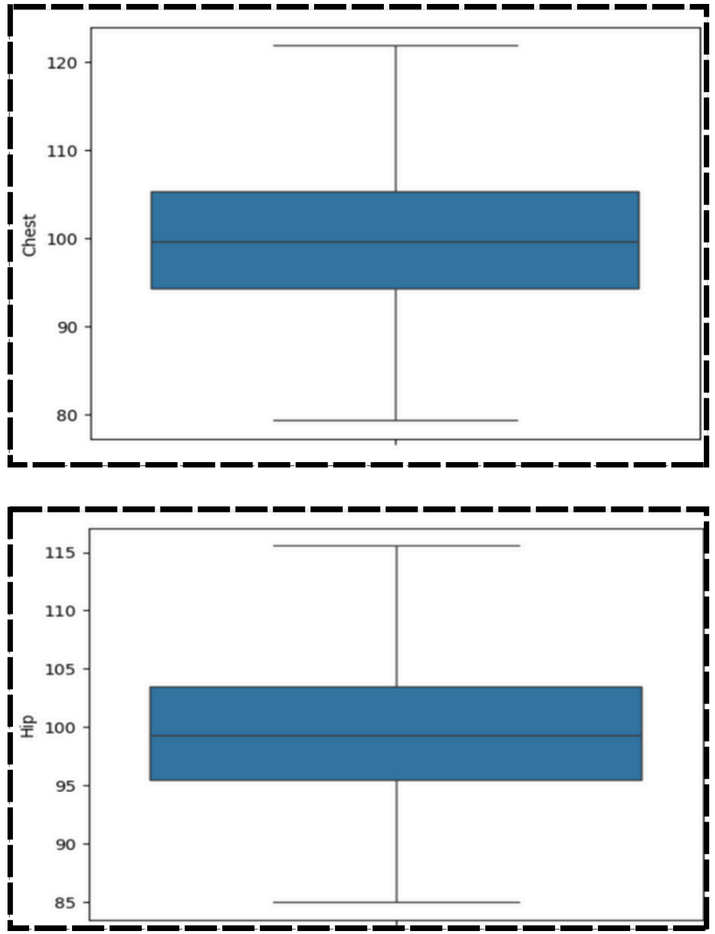




UNIVARIATE ANALYSIS

INSIGHTS(Chest and Hip)

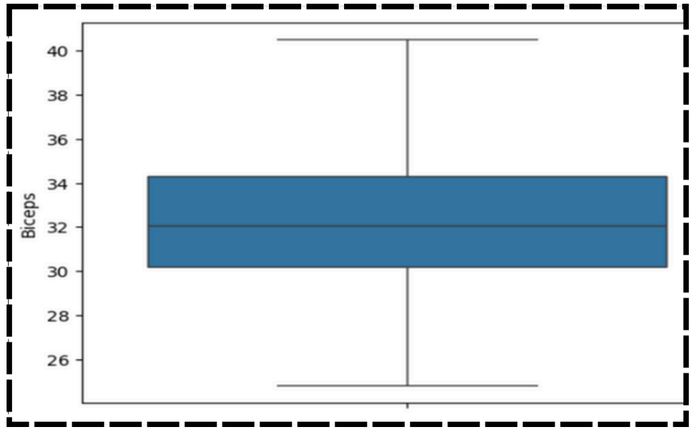
- Approximately 25% of members have chest measurements near 94.35 (Q1).
- Around 75% of members have chest measurements close to 105.37 (Q3).
- Approximately 25% of members have chest measurements near 95.5(Q1).
- Around 75% of members have chest measurements close to 103.5 (Q3).

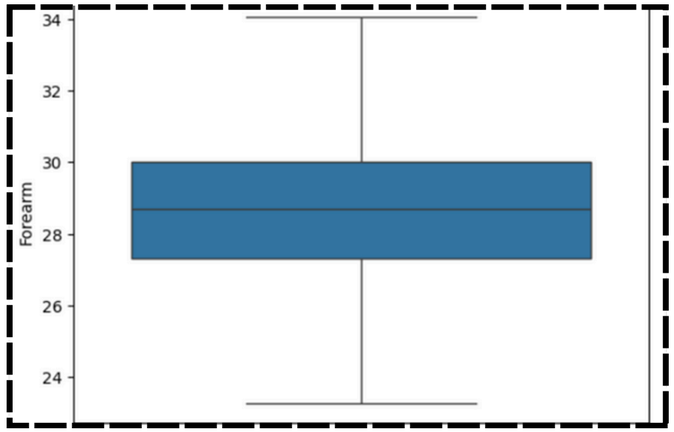


UNIVARIATE ANALYSIS

INSIGHTS(Biceps and forearm)

- Approximately 25% of members have Biceps measurements near 30.2 (Q1).
- Around 75% of members have biceps measurements close to 34.3 (Q3).
- Approximately 25% of members have forearm measurements near 27.3(Q1).
- Around 75% of members have forearm measurements close to 30(Q3)

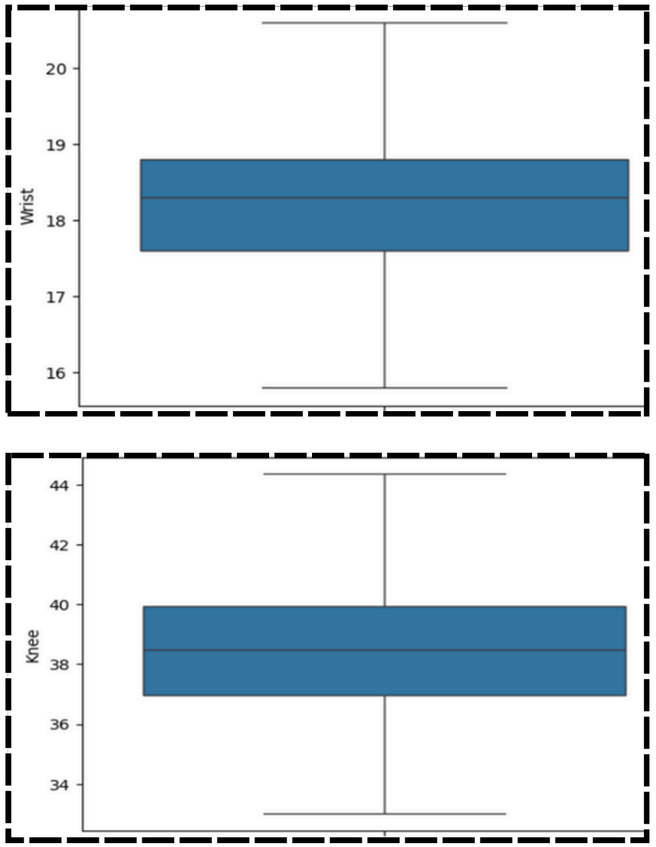




UNIVARIATE ANALYSIS

INSIGHTS(Wrist and knee)

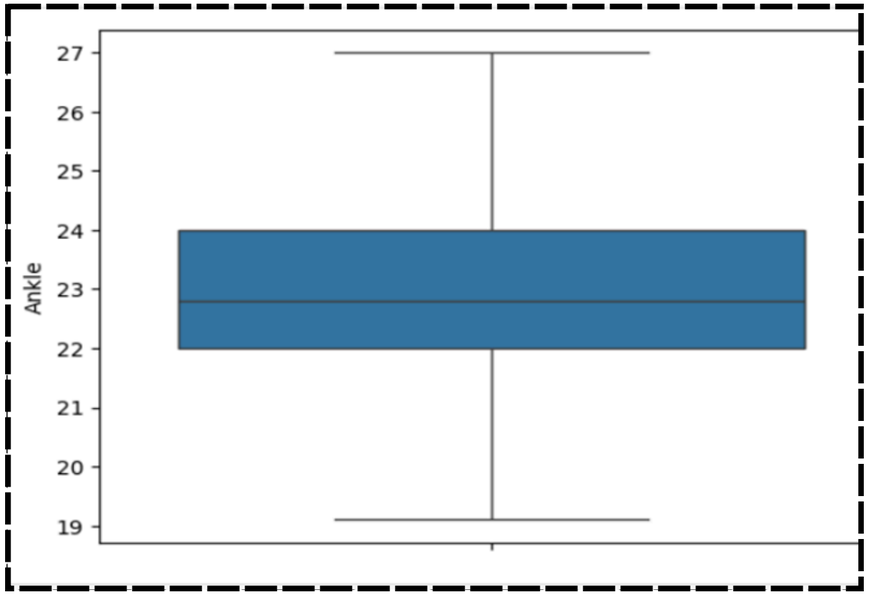
- Approximately 25% of members have wrist measurements near 17.6(Q1).
- Around 75% of members have wrist measurements close to 18.8 (Q3).
- Approximately 25% of members have knee measurements near 36.97(Q1).
- Around 75% of members have knee measurements close to 39.92(Q3)



UNIVARIATE ANALYSIS

INSIGHTS(Ankle)

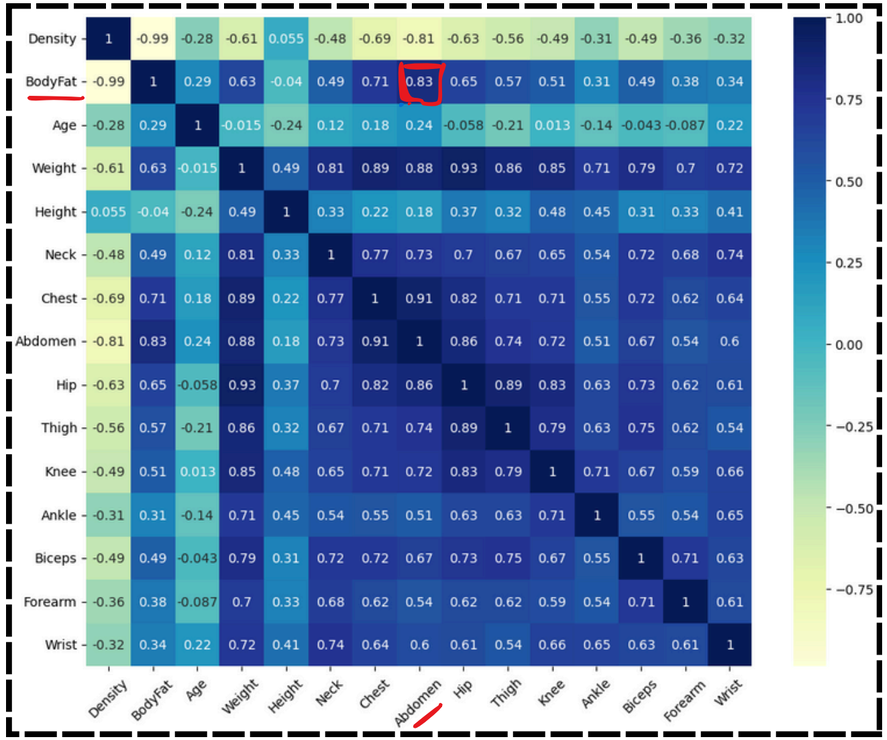
- Approximately 25% of members have ankle measurements near 22(Q1).
- Around 75% of members have ankle measurements close to 24(Q3).



BIVARIATE ANALYSIS

1. Which of the parameters is highly related to body fat?

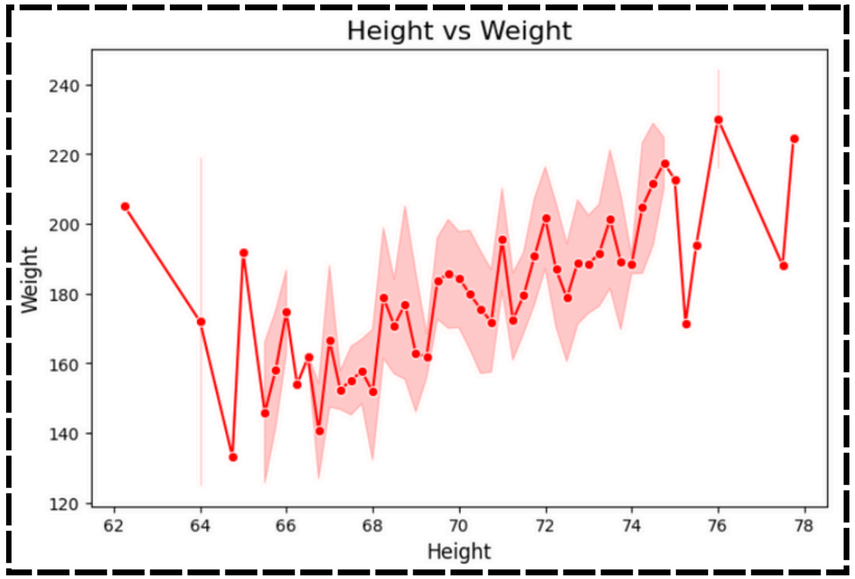
- As per the correlation matrix heat map, abdomen is highly related to body fat - 0.83



BIVARIATE ANALYSIS

2. Do taller people weigh more? What is the relationship between height and weight?

- As the height increases, weight increases and decreases for some data points, so we can say that height doesn't determine the weight of a member
- We didn't have a certain pattern, so height and weight are not related with each other



BIVARIATE ANALYSIS

3. How many members have a minimum and maximum body fat percentage in the dataset?

- Dropped the row which had bodyfat 0.0 from the dataset
- One of the member has bodyfat as 0.7(minimumvalue) but this is not advisable for having such a value to maintain good health
- One of the member has higher bodyfat(44.537) and falls under over fat category

```
# dropping the row with body fat '0' from the dataset
pre_dataset=dataset.drop(dataset[dataset['BodyFat']==0.0].index)
```

```
pre_dataset
```

	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist
0	1.0708	12.3	23	154.25	67.75	36.2	93.1	85.2	94.5	59.0	37.3	21.9	32.0	27.4	17.1
1	1.0853	6.1	22	173.25	72.25	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	28.9	18.2
2	1.0414	25.3	22	154.00	66.25	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	25.2	16.6
3	1.0751	10.4	26	184.75	72.25	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	29.4	18.2
4	1.0340	28.7	24	184.25	71.25	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	27.7	17.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
247	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	1.0236	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	1.0328	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	1.0399	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	1.0271	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	20.6

251 rows × 15 columns

```
def min_max_bodyfat(dataset):
    min_Val_bf=dataset["BodyFat"].min()
    max_Val_bf=dataset["BodyFat"].max()
    min_rows = dataset[dataset["BodyFat"]==dataset["BodyFat"].min()]
    max_rows = dataset[dataset["BodyFat"]==dataset["BodyFat"].max()]
    return min_Val_bf,max_Val_bf,min_rows, max_rows
```

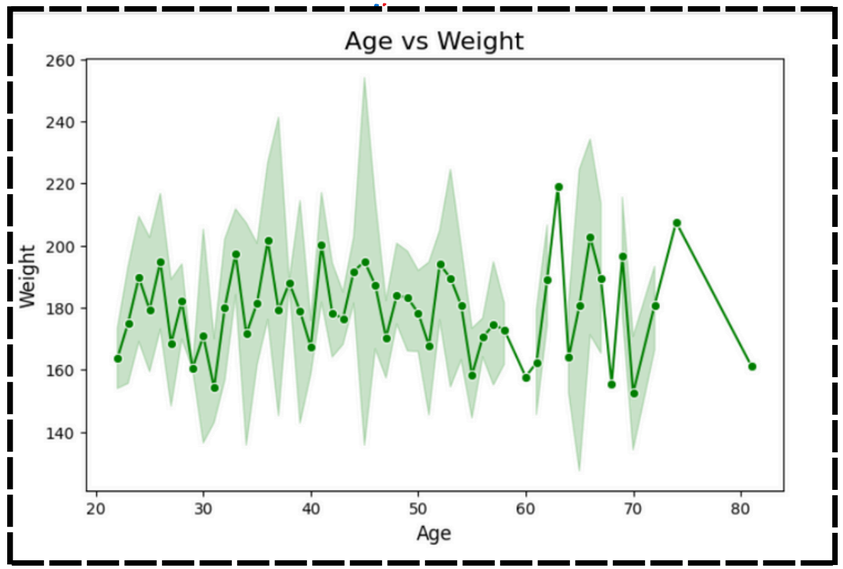
```
min_val,max_val,min_rows,max_rows= min_max_bodyfat(pre_dataset)
print("Minimum bodyfat percentage is:",min_val)
print("Members who has minimum bodyfat percentage:")
print(min_rows.to_string(index=False))
print("Maximum bodyfat percentage is:",max_val)
print("Members who has maximum bodyfat percentage:")
print(max_rows.to_string(index=False))
```

```
Minimum bodyfat percentage is: 0.7
Members who has minimum bodyfat percentage:
 Density  BodyFat  Age  Weight  Height  Neck  Chest  Abdomen  Hip  Thigh  Knee  Ankle  Biceps  Forearm  Wrist
  1.0983      0.7   35  125.75    65.5  34.0   90.8     75.0 89.2   50.0  34.8   22.0    24.8     25.9   16.9
Maximum bodyfat percentage is: 44.5375
Members who has maximum bodyfat percentage:
 Density  BodyFat  Age  Weight  Height  Neck  Chest  Abdomen   Hip  Thigh  Knee  Ankle  Biceps  Forearm  Wrist
  0.9979  44.5375   51   219.0    64.0  41.2  119.8   121.45 112.8   62.5  36.9   23.6    34.7     29.1   18.4
```

BIVARIATE ANALYSIS

4. Find out if people lose weight as they age. What is the relationship between age and weight

- As age increases, weight is increased for some data points but got decreased for some data points
- There is no particular pattern to determine that older people lose weight
- Some of them lose weight while others gain weight as per the graph



BIVARIATE ANALYSIS

5. Classify the members as small, medium and large frame.And determine how many of them fall under each category?

- 124 members fall under the medium frame(based on BMI value) i.e., 49.4%.
- 127 members fall under large frame, i.e., 51%
- None of them belonged to small frame
- Members have BMI greater than or equal to 18.5

```
# adding bmi column to dataframe
df=pre_dataset.copy(deep=True)
df["BMI"] = (703 * df["Weight"]) / (df["Height"] ** 2)

# Classify frame size based on BMI
def classify_frame(bmi):
    if bmi < 18.5:
        return "Small"
    elif 18.5 <= bmi < 25:
        return "Medium"
    else:
        return "Large"

df["Frame"] = df["BMI"].apply(classify_frame)
df
```

	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist	BMI	Frame
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	23.624460	Medium
1	1.0853	6.1	22	173.25	72.25	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	28.9	18.2	23.332048	Medium
2	1.0414	25.3	22	154.00	66.25	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	25.2	16.6	24.666315	Medium
3	1.0751	10.4	26	184.75	72.25	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	29.4	18.2	24.880784	Medium
4	1.0340	28.7	24	184.25	71.25	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	27.7	17.7	25.514854	Large
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
247	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	21.024226	Medium
248	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	29.044437	Large
249	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	30.138946	Large
250	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	26.979981	Large
251	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	20.6	29.769898	Large

```
small_count=(df["Frame"]=="Small").sum()
print("Members falls under small frame:",  small_count)
med_count=(df["Frame"]=="Medium").sum()
print("Members falls under medium frame:", med_count)
Lar_count=(df["Frame"]=="Large").sum()
print("Members falls under large Frame:", Lar_count)
```

```
Members falls under small frame: 0
Members falls under medium frame: 124
Members falls under large Frame: 127
```

BIVARIATE ANALYSIS

6.How many members are short, average and tall in the group

- Most of the members in the group are tall (i.e 71%)
- Some of the members in the group are short(i.e 26%)
- Rest of the members in the group have average height(i.e. 3%)

```
[88]: def apply_height_type(df1):
          Height_category = []
      # If value is a tuple, take the first element
          for value in enumerate(df1["Height"]):
              if isinstance(value, tuple):
                  value = value[0]

              if(value<66):
                  Height_category.append("Short")
              elif(value>72):
                  Height_category.append("Tall")
              else:
                  Height_category.append("Average")
          return Height_category

[89]: df["Height_cat"]=apply_height_type(df)

[90]: df

[90]:
```

	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist	BMI	Frame	Height_cat
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	23.624460	Medium	Short
1	1.0853	6.1	22	173.25	72.25	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	28.9	18.2	23.332048	Medium	Short
2	1.0414	25.3	22	154.00	66.25	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	25.2	16.6	24.666315	Medium	Short
3	1.0751	10.4	26	184.75	72.25	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	29.4	18.2	24.880784	Medium	Short
4	1.0340	28.7	24	184.25	71.25	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	27.7	17.7	25.514854	Large	Short
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
247	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	21.024226	Medium	Tall
248	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	29.044437	Large	Tall
249	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	30.138946	Large	Tall
250	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	26.979981	Large	Tall
251	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	20.6	29.769898	Large	Tall

251 rows × 18 columns

```
short_count=(df["Height_cat"]=="Short").sum()
print("shorter members of the group are:",short_count)
avg_count = (df["Height_cat"]=="Average").sum()
print("Members with average height are:",avg_count)
tall_count=(df["Height_cat"]=="Tall").sum()
print("Taller members of the group are:",tall_count)
```

```
shorter members of the group are: 66
Members with average height are: 7
Taller members of the group are: 178
```

BIVARIATE ANALYSIS

7.How many of them fall under the overfat category with respect to body fat percentage

- Most of the members in the group are tall (i.e 71%)
- Some of the members in the group are short(i.e 26%)
- Rest of the members in the group have average height(i.e. 3%)

```
[92]: def bodyfat_check(df1):
          bodyfat_cat=[]
          for value in df["BodyFat"]:
              if(value>25):
                  bodyfat_cat.append("Overfat")
              elif(value<5):
                  bodyfat_cat.append("Lean")
              else:
                  bodyfat_cat.append("healthy")
          return bodyfat_cat

[94]: df["bodyfat_cat"]=bodyfat_check(df)

[95]: df

[96]:
```

	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist	BMI	Frame	Height_cat
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	23.624460	Medium	Short
1	1.0853	6.1	22	173.25	72.25	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	28.9	18.2	23.332048	Medium	Short
2	1.0414	25.3	22	154.00	66.25	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	25.2	16.6	24.666315	Medium	Short
3	1.0751	10.4	26	184.75	72.25	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	29.4	18.2	24.880784	Medium	Short
4	1.0340	28.7	24	184.25	71.25	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	27.7	17.7	25.514854	Large	Short
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
247	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	21.024226	Medium	Tall
248	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	29.044437	Large	Tall
249	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	30.138946	Large	Tall
250	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	26.979981	Large	Tall
251	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	20.6	29.769898	Large	Tall

251 rows × 19 columns

```
over_count=(df["bodyfat_cat"]=="Overfat").sum()
print("Members fall under overfat category are:",over_count)
```

```
Members fall under overfat category are: 66
```

26% of the members fall under overfat category

BIVARIATE ANALYSIS

8. How many of them fall under the lean category with respect to body fat percentage

- Only few members are lean in the group(7,i.e.3%)
- Most probably the members who are lean might have average height

```
over_count=(df["bodyfat_cat"]=="Overfat").sum()
print("Members fall under overfat category are:",over_count)
```

```
Members fall under overfat category are: 66
```

26% of the members fall under overfat category

BIVARIATE ANALYSIS

9. Test the Analysis of Variance between abdomen and weight at a significance level of 5% (Make a decision using Hypothesis testing)

- H0: There is no significant difference between abdomen and weight of the students
- H1: There is significant difference between abdomen and weight of the students
- Reject null hypothesis: if p<0.05
- Here p-value is (3.3522210068684144e-186) <0.05
- Hence reject null hypothesis and accept alternate hypothesis

```
import scipy.stats as stats
stats.f_oneway(pre_dataset["Abdomen"],pre_dataset['Weight'])
```

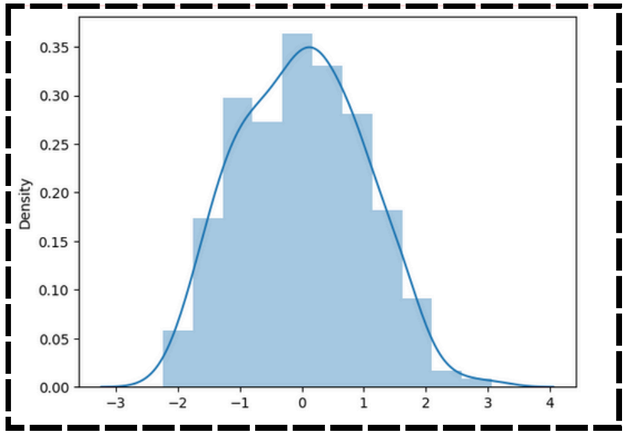
```
F_onewayResult(statistic=np.float64(2224.287717660033), pvalue=np.float64(3.3522210068684144e-186))
```

BIVARIATE ANALYSIS

10.Convert the body fat percentage values into standard normal form.

```
# Converting normal distribution to Standard normal distribution
# defining the function with function name stdNBgraph and pre_dataset to be passed as parameter
def stdNBgraph(dataset):
# import seaborn library for plotting the graph
    import seaborn as sns
# calculating the mean for the given column in the dataset using mean() function and assign it to mean variable
    mean=dataset.mean()
# calculating the standard deviation for the given column in the dataset using std() function and assigne it to std variable
    std=dataset.std()
# each values of particular column in dataset is store in list i and assigned to  list values
    values = [i for i in dataset]
# z_score is calculated for each values in dataset and stored to list z_score
    z_score =[((j-mean)/std) for j in values]
# density plot with respect to the given column in the dataset is plotted using distplot method by passing z_score as parameters
# kde=True so the curve is drawn in the plot
    sns.distplot(z_score,kde=True)
# summation of the z_score divided by number of z_score values is calculated
    (sum(z_score)/len(z_score))

stdNBgraph(pre_dataset["BodyFat"])
```



FEATURE SELECTION & MODEL CREATION

- Select kbest algorithm is used to select the best features
- StandardScaler is used for preprocessing the input after identifying the best features using selectKBest
- Trained the dataset with Linear, SVM, RandomForest and Decision Tree Regression using GridSearchCV

IMPLEMENTATION

Importing Libraries

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
from sklearn.model_selection import GridSearchCV
```

Importing Dataset

```
dataset=pd.read_csv("Preprocessed_dataset.csv")
dataset
```

	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm	Wrist
0	1.0708	12.3	23	154.25	67.75	36.2	93.1	85.2	94.5	59.0	37.3	21.9	32.0	27.4	17.1
1	1.0853	6.1	22	173.25	72.25	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	28.9	18.2
2	1.0414	25.3	22	154.00	66.25	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	25.2	16.6
3	1.0751	10.4	26	184.75	72.25	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	29.4	18.2
4	1.0340	28.7	24	184.25	71.25	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	27.7	17.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
246	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
247	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
248	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
249	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
250	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	20.6

Feature selection using selectKbest

Function for selectkbest

```
def selectkbest(indep_X,dep_Y,n):
    test = SelectKBest(score_func=f_regression, k=n)
    fit1 = test.fit(indep_X,dep_Y)
    # select features using selectkbest for the dataset and return them
    #selectkfeatures=fit1.transform(indep_X)
    #return selectkfeatures
    indices=fit1.get_support(indices=True)
    features_name=fit1.get_feature_names_out()
    return indices,features_name
```

Function for standardscalar

```
def scalar(indep_X):
    sc=StandardScaler()
    indep_X = sc.fit_transform(indep_X)
    return indep_X,sc
```

Function for r2_prediction using grid search

```
def r2_prediction_grid(grid,indep_X,dep_Y):
    grid_predict=grid.predict(indep_X)
    from sklearn.metrics import r2_score
    r_score = r2_score(dep_Y,grid_predict)
    return r_score
```

IMPLEMENTATION

Function for Linear regression using grid search

```
def Linear_grid(indep_X, dep_Y):
    from sklearn.linear_model import LinearRegression
    param_grid={
    'fit_intercept': [True, False],
    'positive': [True, False]
    }
    grid=GridSearchCV(estimator=LinearRegression(), param_grid=param_grid, cv=5, scoring='neg_mean_squared_error')
    grid.fit(indep_X,dep_Y)
    r_score=r2_prediction_grid(grid,indep_X,dep_Y)
    return r_score,grid.best_estimator_
```

Function for SVM using grid search

```
def SVM_grid(indep_X,dep_Y):
    from sklearn.svm import SVR
    param_grid={'kernel':['linear','rbf'],
                'C':[0.1,1,10,100,500],
                'max_iter':[50000],
                'epsilon':[0.1,0.5,1.0],
               'gamma':['scale']}
    grid= GridSearchCV(SVR(),param_grid,refit=True,verbose=3,n_jobs=-1)
    grid.fit(indep_X, dep_Y)
    r_score=r2_prediction_grid(grid,indep_X,dep_Y)
    return r_score,grid.best_estimator_
```

Function for Decision Tree using grid search

```
def Decision_grid(indep_X, dep_Y):
    from sklearn.tree import DecisionTreeRegressor
    param_grid={'criterion':['squared_error','friedman_mse'],
                'max_features':[None,'sqrt','log2'],
                'max_depth':[4,6],
                'splitter':['best','random'],
                'random_state':[0,None]}
    grid= GridSearchCV(DecisionTreeRegressor(),param_grid,refit=True,verbose=3,n_jobs=-1)
    grid.fit(indep_X, dep_Y)
    r_score=r2_prediction_grid(grid,indep_X,dep_Y)
    return r_score,grid.best_estimator_
```

Function for Random Forest using grid search

```
def Random_grid(indep_X, dep_Y):
    from sklearn.ensemble import RandomForestRegressor
    param_grid={'criterion':['squared_error','friedman_mse'],
                'max_features':[None,'sqrt','log2'],
                'max_depth':[4,5],
                'n_estimators':[100,50],
                'random_state':[0,None]}
    grid=GridSearchCV(RandomForestRegressor(),param_grid,refit=True,verbose=3,n_jobs=-1)
    grid.fit(indep_X, dep_Y)
    r_score=r2_prediction_grid(grid,indep_X,dep_Y)
    return r_score,grid.best_estimator_
```

IMPLEMENTATION

Function to create table for the models

```
def selectk_regression(acclin,accsvm,accdes,accrf):
    dataframe=pd.DataFrame(index=['f_regression'],columns=['Linear_grid','SVM_grid','Decision_grid','Random_grid'])

    for number,idex in enumerate(dataframe.index):
        dataframe.loc[idex,'Linear_grid']=acclin[number]
        dataframe.loc[idex,'SVM_grid']=accsvm[number]
        dataframe.loc[idex,'Decision_grid']=accdes[number]
        dataframe.loc[idex,'Random_grid']=accrf[number]
    return dataframe
```

Defining Input and Output data

```
df2=dataset.copy(deep=True)

indep_X=df2.drop('BodyFat', axis=1)
dep_Y=df2['BodyFat']
```

```
kbest,feature_names=selectkbest(indep_X,dep_Y,3)

acclin=[]
accsvm=[]
accdes=[]
accrf=[]
```

Main function to run models for selected k features

```
X_kbest=indep_X.iloc[:,kbest]
X_scaled,sc = scalar(X_kbest)
r2_lin,lin_model=Linear_grid(X_scaled,dep_Y)
acclin.append(r2_lin)
r2_svm,svm_model=SVM_grid(X_scaled,dep_Y)
accsvm.append(r2_svm)
r2_d,dec_model=Decision_grid(X_scaled,dep_Y)
accdes.append(r2_d)
r2_r,ran_model=Random_grid(X_scaled,dep_Y)
accrf.append(r2_r)
```

```
Fitting 5 folds for each of 30 candidates, totalling 150 fits
```

IMPLEMENTATION

R_score of models when k=4

```
#k=4
print(feature_names)
result=selectk_regression(acclin,accsvm,accdes,accrf)
result
```

```
['Density' 'Chest' 'Abdomen' 'Hip']
```

	Linear_grid	SVM_grid	Decision_grid	Random_grid
f_regression	0.978071	0.994376	0.96039	0.991204

R_score of models when k=5

```
# k=5
print(feature_names)
result=selectk_regression(acclin,accsvm,accdes,accrf)
result
```

```
['Density' 'Weight' 'Chest' 'Abdomen' 'Hip']
```

	Linear_grid	SVM_grid	Decision_grid	Random_grid
f_regression	0.978103	0.994362	0.98471	0.990908

R_score of models when k=6

```
# k=6
print(feature_names)
result=selectk_regression(acclin,accsvm,accdes,accrf)
result
```

```
['Density' 'Weight' 'Chest' 'Abdomen' 'Hip' 'Thigh']
```

	Linear_grid	SVM_grid	Decision_grid	Random_grid
f_regression	0.978195	0.994499	0.990508	0.989682

R_score of models when k=3

```
# k=3
print(feature_names)
result=selectk_regression(acclin,accsvm,accdes,accrf)
result
```

```
['Density' 'Chest' 'Abdomen']
```

	Linear_grid	SVM_grid	Decision_grid	Random_grid
f_regression	0.978069	0.994439	0.98471	0.991189

IMPLEMENTATION

Best model and r_score when k=3

```
best_model_name = result.iloc[0].idxmax()
best_score = result.iloc[0].max()

print("Best model:", best_model_name)
print("Best score:", best_score)
```

```
Best model: SVM_grid
Best score: 0.9944387925955688
```

Saving the best model,kbest and preprocessing steps

```
import pickle
pickle.dump(svm_model, open("Bodyfat_model_svm.pkl",'wb'))
pickle.dump({"indices": kbest,"feature_names": feature_names}, open("Bodyfat_kbestfeatures.pkl",'wb'))
pickle.dump(sc,open("Bodyfat_scalar.pkl",'wb'))
```

Preprocessing the raw input from the user

```
# Values for Density, chest and abdomen obtained from the user
preinput=sc.transform([[1.065,89,78]])
```

```
preinput
```

```
array([[ 0.51498749, -1.46955282, -1.42496982]])
```

```
loaded_model=pickle.load(open("Bodyfat_model_svm.pkl",'rb'))
result=loaded_model.predict(preinput)
```

```
result
```

```
array([14.68831857])
```

FINAL MODEL

- Highest r2-score is obtained for SVM when k value is selected as '3' (rscore:0.9944)
- The model is saved and deployed using django framework

Like

& FOLLOW FOR MORE