```
import numpy as np
import pandas as pd
```

```
from google.colab import files
uploaded= files.upload()
```

Choose Files EDA_FAT.csv

- **EDA_FAT.csv**(application/vnd.ms-excel) - 21115 bytes, last modified: 12/20/2021 - 100% done

```
Saving EDA_FAT.csv to EDA_FAT (1).csv
```

```
df=pd.read_csv('EDA_FAT.csv')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252 entries, 0 to 251
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  252 non-null    int64  
 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(2)
memory usage: 31.6 KB
```

```
df=df.drop(columns=["Unnamed: 0","Height", "Weight", "Density"],axis=1)
```

```
df.columns
```

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

```
df.describe()
```

	BodyFat	Age	Neck	Chest	Abdomen	Hip	Thi
count	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.00000
mean	19.139038	44.884921	37.967808	100.742163	92.428770	99.735268	59.3281
std	8.330753	12.602040	2.301730	8.161876	10.293612	6.438057	4.9628
min	0.000000	22.000000	31.862500	79.300000	69.400000	85.000000	47.2000
25%	12.475000	35.750000	36.400000	94.350000	84.575000	95.500000	56.0000
50%	19.200000	43.000000	38.000000	99.650000	90.950000	99.300000	59.0000
75%	25.300000	54.000000	39.425000	105.375000	99.325000	103.525000	62.3500
max	44.537500	81.000000	43.962500	121.912500	121.450000	115.562500	71.8750

```
X=df.loc[:,df.columns!="BodyFat"]
y=df.loc[:,df.columns=="BodyFat"]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(X,y, test_size=0.2, random_state= 14)
```

```
df=pd.get_dummies(df,drop_first=True)
```

```
df
```

	BodyFat	Age	Neck	Chest	Abdomen	Hip	Thigh	Knee	Ankle	Biceps	Forearm
0	12.3	23	36.2	93.1	85.2	94.5	59.0	37.3	21.9	32.0	27.4
1	6.1	22	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	28.9
2	25.3	22	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	25.2
3	10.4	26	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	29.4
4	28.7	24	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	27.7
...	...	...	...	...	...	...	...	...	...	...	...
247	11.0	70	34.9	89.2	83.6	88.8	49.6	34.8	21.5	25.6	25.7
248	33.6	72	40.9	108.5	105.0	104.5	59.6	40.8	23.2	35.2	28.6
249	29.3	72	38.9	111.1	111.5	101.7	60.3	37.3	21.5	31.3	27.2
250	26.0	72	38.9	108.3	101.3	97.8	56.0	41.6	22.7	30.5	29.4
251	31.9	74	40.8	112.4	108.5	107.1	59.3	42.2	24.6	33.7	30.0

252 rows × 12 columns

```
from sklearn.tree import DecisionTreeRegressor
```

```
model_dec=DecisionTreeRegressor().fit(X_train,y_train)
```

```
pred_dec=model_dec.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.metrics import r2_score
import numpy
```

```
print(np.sqrt(mean_squared_error(y_test,pred_dec)))
print(r2_score(y_test,pred_dec))
```

```
6.210585523847136
0.4415965360821621
```

```
from sklearn.linear_model import LinearRegression
```

```
from sklearn.svm import SVR
```

```
from sklearn.neighbors import KNeighborsRegressor
```

```
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor,GradientBoostingRegre
```

```
model_ln=LinearRegression().fit(X_train,y_train)
```

```
model_svm=SVR().fit(X_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConversi
  y = column_or_1d(y, warn=True)
```

```
model_knn=KNeighborsRegressor().fit(X_train,y_train)
```

```
model_rf=RandomForestRegressor().fit(X_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DataConversionWarning
  """Entry point for launching an IPython kernel.
```

```
model_ad=AdaBoostRegressor().fit(X_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConversi
  y = column_or_1d(y, warn=True)
```

```
model_gb=GradientBoostingRegressor().fit(X_train,y_train)
```

```
    /usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_gb.py:494: DataConversionWar
      y = column_or_1d(y, warn=True)
```

```
pred_ln=model_ln.predict(X_test)
```

```
pred_rf=model_rf.predict(X_test)
```

```
pred_ad=model_ad.predict(X_test)
```

```
pred_gb=model_gb.predict(X_test)
```

```
pred_svm=model_svm.predict(X_test)
```

```
pred_knn=model_knn.predict(X_test)
```

checking for accuarcy of the model

```
# Linear regression
print(np.sqrt(mean_squared_error(y_test,pred_ln)))
r2_score(y_test,pred_ln)
```

```
    4.273923003247361
    0.7355543679195768
```

```
# Random forest
print(np.sqrt(mean_squared_error(y_test,pred_rf)))
r2_score(y_test,pred_rf)
```

```
    4.737838986268309
    0.6750297497109302
```

```
# KNN
print(np.sqrt(mean_squared_error(y_test,pred_knn)))
r2_score(y_test,pred_knn)
```

```
    4.755433128558963
    0.6726116898947614
```

```
# SVM
print(np.sqrt(mean_squared_error(y_test,pred_svm)))
```

```
r2_score(y_test,pred_svm)
```

```
6.073998978524068
0.4658878721771883
```

```
# Ada boosting
print(np.sqrt(mean_squared_error(y_test,pred_ad)))
r2_score(y_test,pred_ad)
```

```
4.721722664949965
0.6772368391863808
```

```
# Gradient boosting
print(np.sqrt(mean_squared_error(y_test,pred_gb)))
r2_score(y_test,pred_gb)
```

```
4.785141712191264
0.668508331032744
```

hence Linear regression model is having high accuracy almost close to 0.73

✓ 0s completed at 9:50 PM ● ✕