image source: YouTube Mario Tomic (https://www.youtube.com/c/MarioTomicOfficial)

Body Fat Level Predictions - Machine Learning vs. US Navy Formula

instead of the original dataset, we are gonna use the data generated with my first notebook (https://www.kaggle.com/code/alexandrepetit881234/bodyfat-visualization)

in adddition to the original data, a few more column were created:

- BMI, a column with the individual Body Mass Index
- LeanPercent, the percent of lean tissues
- FatFreeMass, the lean mass in kg
- FFMI, the fat-free mass index. Similar to BMI, but for the lean mass
- Overweight, BMI over 25
- Obese, BMI over 30
- HighFat, body-fat above 22 percent

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
/kaggle/input/body-fat-prediction-dataset/bodyfat.csv
/kaggle/input/bodyfat-prediction-cleaned-extended/BodyFatCleaned.cs
v
```

1.Data Preparation

In [2]:

```
df = pd.read_csv('/kaggle/input/bodyfat-prediction-cleaned-extended/BodyFatCleaned.csv')
df.drop('Unnamed: 0', axis=1, inplace=True)
df.head()
```

Out[2]:

	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh
0	1.0708	12.3	23	70.1	1.7	36.2	93.1	85.2	94.5	59.0
1	1.0853	6.1	22	78.8	1.8	38.5	93.6	83.0	98.7	58.7
2	1.0414	25.3	22	70.0	1.7	34.0	95.8	87.9	99.2	59.6
3	1.0751	10.4	26	84.0	1.8	37.4	101.8	86.4	101.2	60.1
4	1.0340	28.7	24	83.8	1.8	34.4	97.3	100.0	101.9	63.2

5 rows × 22 columns

let's add a new column for people under 10% body-fat,

it will be used to compare the density based body-fat percent with the US navy formula fat percent

In [3]:

```
df['LowFat'] = 0

lf = 10

df['LowFat'][df['BodyFat'] < lf] = 1
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy
  """
```

This formula is used by the US navy to estimate the body-fat percent, let's see how it compare with the density based body-fat calculation

%BF = 495 / (1.0324 - 0.19077 *log10(waist - neck) + 0.15456* log10(height)) - 450

In [4]:

```
df['NavyFat'] = 495 / (1.0324 - 0.19077 * np.log10(df['Abdomen'] - df['Neck']) +
0.15456 * np.log10(df['Height'] * 100)) - 450
```

In [5]:

```
df['NavyFat'] = round(df['NavyFat'], 1)
df.head()
```

Out[5]:

	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	...
0	1.0708	12.3	23	70.1	1.7	36.2	93.1	85.2	94.5	59.0	...
1	1.0853	6.1	22	78.8	1.8	38.5	93.6	83.0	98.7	58.7	...
2	1.0414	25.3	22	70.0	1.7	34.0	95.8	87.9	99.2	59.6	...
3	1.0751	10.4	26	84.0	1.8	37.4	101.8	86.4	101.2	60.1	...
4	1.0340	28.7	24	83.8	1.8	34.4	97.3	100.0	101.9	63.2	...

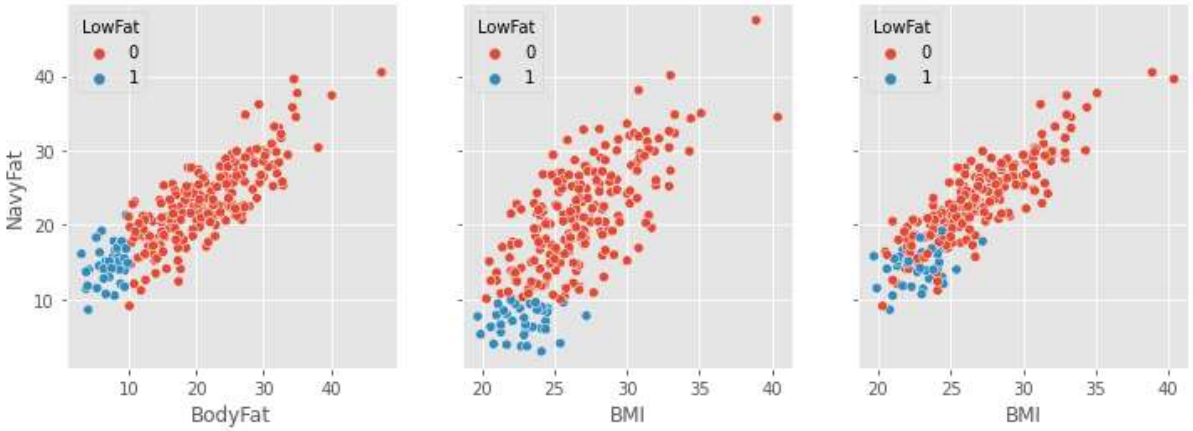
5 rows × 24 columns

In [6]:

```
plt.style.use('ggplot')
fig, ax = plt.subplots(1,3, figsize=(12, 4), sharey=True)

sns.scatterplot(ax=ax[0], data=df, x='BodyFat', y='NavyFat', hue='LowFat')
sns.scatterplot(ax=ax[1], data=df, x='BMI', y='BodyFat', hue='LowFat')
sns.scatterplot(ax=ax[2], data=df, x='BMI', y='NavyFat', hue='LowFat')

plt.ylabel('FatPercent')
plt.show()
```



In [7]:

```
df[['BodyFat', 'NavyFat', 'Abdomen']][df['BodyFat'] < 10]
```

Out[7]:

	BodyFat	NavyFat	Abdomen
1	6.1	14.1	83.0
8	4.1	14.0	82.5
10	7.1	13.0	83.6
11	7.8	17.8	90.9
25	3.7	13.7	79.7
26	7.9	10.5	74.6
28	3.7	11.4	73.9
29	8.8	17.6	83.5
31	5.7	16.3	84.5
42	7.7	15.8	76.0
45	5.6	14.5	79.5
47	4.0	8.6	70.4
49	6.6	15.1	77.9
50	8.0	16.6	82.0
51	6.3	14.1	79.6
52	3.9	11.8	77.6
66	6.3	12.8	78.4
69	8.8	15.5	82.8
70	8.5	15.7	82.9
74	8.8	12.1	78.8
86	8.3	16.9	86.0
90	8.5	15.0	83.1
92	9.0	17.8	88.8
94	9.6	21.3	91.6
141	9.4	11.7	77.1
146	5.3	11.5	76.5
148	9.4	13.9	77.6
158	9.4	15.5	81.2
168	3.0	16.1	81.9
172	9.9	14.9	79.4
181	8.6	14.0	83.7
194	6.6	10.7	78.0
199	6.0	19.2	89.1
204	9.6	16.8	83.9
206	7.1	15.1	79.4
213	7.5	14.2	81.5
219	5.2	18.3	82.8

In [8]:

```
df[df['NavyFat'] < 10]
```

Out[8]:

	Density	BodyFat	Age	Weight	Height	Neck	Chest	Abdomen	Hip	Thigh	..
47	1.0903	4.0	47	58.0	1.7	34.0	83.4	70.4	87.2	50.6	..
150	1.0758	10.1	27	66.4	1.8	34.1	88.5	72.8	91.1	53.6	..

2 rows × 24 columns

with the navy body fat formula, only two people have an estimated bofy fat level under 10%. Much less than the people under 10% body-fat calculated with the density

2. Machine Learning Model

we will use the measurements of the different body part to estimate the body-fat percent of an individual. Because the density is used to calculate the BodyFat column, we will not keep it with the training data

In [9]:

```
df.columns
```

Out[9]:

```
Index(['Density', 'BodyFat', 'Age', 'Weight', 'Height', 'Neck', 'Ch
est',
       'Abdomen', 'Hip', 'Thigh', 'Knee', 'Ankle', 'Biceps', 'Forea
rm',
       'Wrist', 'BMI', 'LeanPercent', 'FatFreeMass', 'FFMI', 'Overw
eight',
       'Obese', 'HighFat', 'LowFat', 'NavyFat'],
      dtype='object')
```

Data Selection and Separation

In [10]:

```
X = df[['Age', 'Weight', 'Height', 'Neck', 'Chest',
       'Abdomen', 'Hip', 'Thigh', 'Knee', 'Ankle', 'Biceps', 'Forearm',
       'Wrist']]
y_b = df['BodyFat']
y_n = df['NavyFat']
```

In [11]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train_b, y_test_b, y_train_n, y_test_n = train_test_split(X,
y_b, y_n,
                                                                             test_
size=0.25,
                                                                             rando
m_state=42)
```

In [12]:

```
print(X_train.shape, X_test.shape)
print(y_train_b.shape, y_test_b.shape)
print(y_train_n.shape, y_test_n.shape)
```

```
(186, 13) (62, 13)
(186,) (62,)
(186,) (62,)
```

For regression problem, we cannot use the accuracy score. We are going to use the mean squared error

In [13]:

```
from sklearn.metrics import mean_squared_error
from time import perf_counter
```

Linear Regression

In [14]:

```
from sklearn.linear_model import LinearRegression

start = perf_counter()
reg = LinearRegression()
reg.fit(X_train, y_train_b)

y_pred_b = reg.predict(X_test)
reg_mse_b = mean_squared_error(y_test_b, y_pred_b)

print(f'the mean squared error for linear regression on density body-fat is {rou
nd(reg_mse_b, 2) }')

reg.fit(X_train, y_train_n)

y_pred_n = reg.predict(X_test)
reg_mse_n = mean_squared_error(y_test_n, y_pred_n)

print(f'the mean squared error for linear regression on navy body-fat is {round(
reg_mse_n, 2)}')

end = perf_counter()

elapsed_time = round(end - start, 3)

print(f'it took {elapsed_time} second for predicting the body fat percent using
 linear regression')
```

```
the mean squared error for linear regression on density body-fat is
19.45
the mean squared error for linear regression on navy body-fat is 0.
15
it took 0.029 second for predicting the body fat percent using line
ar regression
```

Decision Tree

In [15]:

```
from sklearn.tree import DecisionTreeRegressor

start = perf_counter()
dtr = DecisionTreeRegressor()
dtr.fit(X_train, y_train_b)

y_pred_b = dtr.predict(X_test)
tree_mse_b = mean_squared_error(y_test_b, y_pred_b)

print(f'the mean squared error for decision tree on density body-fat is {round(t
ree_mse_b, 2) }')

dtr.fit(X_train, y_train_b)

y_pred_n = dtr.predict(X_test)
tree_mse_n = mean_squared_error(y_test_n, y_pred_n)

print(f'the mean squared error for decision tree on navy body-fat is {round(tree
_mse_n, 2)}')

end = perf_counter()

elapsed_time = round(end - start, 3)

print(f'it took {elapsed_time} second to predict the body fat percent using a de
cision tree')
```

```
the mean squared error for decision tree on density body-fat is 40.
42
the mean squared error for decision tree on navy body-fat is 33.97
it took 0.014 second to predict the body fat percent using a decisi
on tree
```

Linear regression gives us predictions closer to the value we calculate before for both the density based body-fat and the navy formula based body-fat. Decision Tree also took a little more time.

For some problems, we don't need a complex model

In [16]:

```
mse_navy_fat = mean_squared_error(y_b, y_n)
mse_navy_fat = round(mse_navy_fat, 2)
print(f'The mean squared error for the Navy bod-fat formula is {mse_navy_fat}')
```

```
The mean squared error for the Navy bod-fat formula is 27.4
```

Wait, was machine learning necessary for predicting the body-fat?

we saw just before that the navy body-fat formula was giving us a good estimation of the body-fat level. The mean squared error for the formula is even smaller than the mean squared error of the decision tree