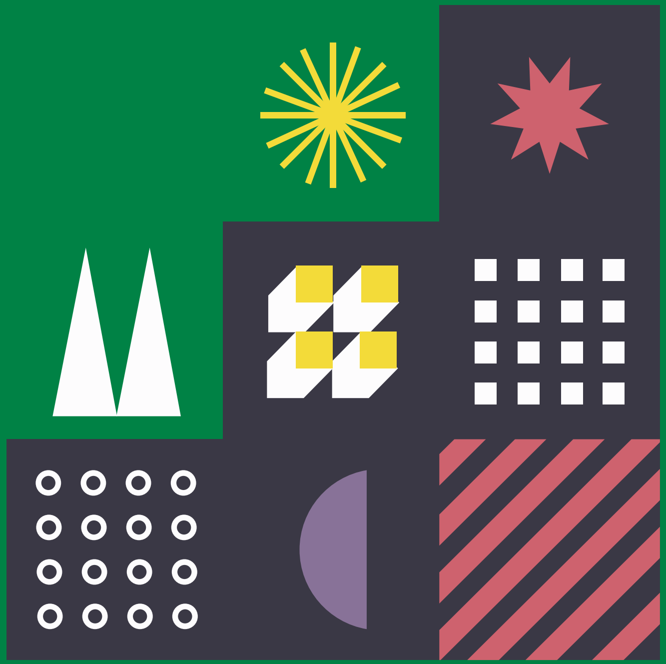
# EDA & Decision Tree Classification with Obesity Classification Dataset



# Table of contents

# 01

# Preparing & Understanding the Data

# Obesity Classification Dataset

Contains information about obesity classification of individuals.

## Columns:

- ID
- Age
- Gender
- Height (cm)
- Weight (kg)
- BMI: The body mass index of the individual, calculated as weight divided by height squared.
- Label: The obesity classification of the individual:
    - Normal Weight
    - Overweight
    - Obese
    - Underweight

## Obesity Classification.csv (3.96 kB)

Detail Compact Column

| ID | Age | Gender | Height | Weight | BMI | Label |
|---|---|---|---|---|---|---|
| 1 | 25 | Male | 175 | 80 | 25.3 | Normal Weight |
| 2 | 30 | Female | 160 | 60 | 22.5 | Normal Weight |
| 3 | 35 | Male | 180 | 90 | 27.3 | Overweight |
| 4 | 40 | Female | 150 | 50 | 20 | Underweight |
| 5 | 45 | Male | 190 | 100 | 31.2 | Obese |
| 6 | 50 | Female | 140 | 40 | 16.7 | Underweight |
| 7 | 55 | Male | 200 | 110 | 34.2 | Obese |
| 8 | 60 | Female | 130 | 30 | 13.3 | Underweight |
| 9 | 65 | Male | 210 | 120 | 37.2 | Obese |
| 10 | 70 | Female | 120 | 20 | 10 | Underweight |

**Source:** https://www.kaggle.com/datasets/sujithmandala/obesity-classification-dataset

# Problem Statement

- Obesity Classification using Machine Learning

**Objective:**

- Build a machine learning model that can accurately classify individuals into different obesity categories based on their gender, age, BMI, weight, and height.

# 02 Exploratory Data Analysis

# Libraries

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

Used for data visualization

# Interpreting Data

- Melihat 5 data teratas

```
d.head()
```

| | ID | Age | Gender | Height | Weight | BMI | Label |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 25 | Male | 175 | 80 | 25.3 | Normal Weight |
| 1 | 2 | 30 | Female | 160 | 60 | 22.5 | Normal Weight |
| 2 | 3 | 35 | Male | 180 | 90 | 27.3 | Overweight |
| 3 | 4 | 40 | Female | 150 | 50 | 20.0 | Underweight |
| 4 | 5 | 45 | Male | 190 | 100 | 31.2 | Obese |

```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 108 entries, 0 to 107
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      108 non-null    int64
 1   Age     108 non-null    int64
 2   Gender  108 non-null    object
 3   Height  108 non-null    int64
 4   Weight  108 non-null    int64
 5   BMI     108 non-null    float64
 6   Label   108 non-null    object
dtypes: float64(1), int64(4), object(2)
memory usage: 6.0+ KB
```

- Daftar column & tipe data masing-masing column

# Interpreting Data

```
d.shape
```

(108, 7)

```
d.describe()
```

| | ID | Age | Height | Weight | BMI |
|---|---|---|---|---|---|
| **count** | 108.000000 | 108.000000 | 108.000000 | 108.000000 | 108.000000 |
| **mean** | 56.046296 | 46.555556 | 166.574074 | 59.490741 | 20.549074 |
| **std** | 31.917939 | 24.720620 | 27.873615 | 28.856233 | 7.583818 |
| **min** | 1.000000 | 11.000000 | 120.000000 | 10.000000 | 3.900000 |
| **25%** | 28.750000 | 27.000000 | 140.000000 | 35.000000 | 16.700000 |
| **50%** | 56.500000 | 42.500000 | 175.000000 | 55.000000 | 21.200000 |
| **75%** | 83.250000 | 59.250000 | 190.000000 | 85.000000 | 26.100000 |
| **max** | 110.000000 | 112.000000 | 210.000000 | 120.000000 | 37.200000 |

- Dataset memiliki 108 row dan 7 column
- Memperlihatkan *statistical* data

# Encoding Columns

```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 108 entries, 0 to 107
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      108 non-null    int64
 1   Age     108 non-null    int64
 2   Gender  108 non-null    object
 3   Height  108 non-null    int64
 4   Weight  108 non-null    int64
 5   BMI     108 non-null    float64
 6   Label   108 non-null    object
dtypes: float64(1), int64(4), object(2)
memory usage: 6.0+ KB
```

- Convert them into numerical representations that can be more easily understood and processed by machine learning algorithms.

```
obj_cols = d.select_dtypes(include=['object']).columns.tolist()
```

```
print(obj_cols)
```

```
['Gender', 'Label']
```

```
le = LabelEncoder()

for obj in obj_cols:
    le.fit(d[obj])
    d[obj] = le.transform(d[obj])
```

data should be represented as numerical values:

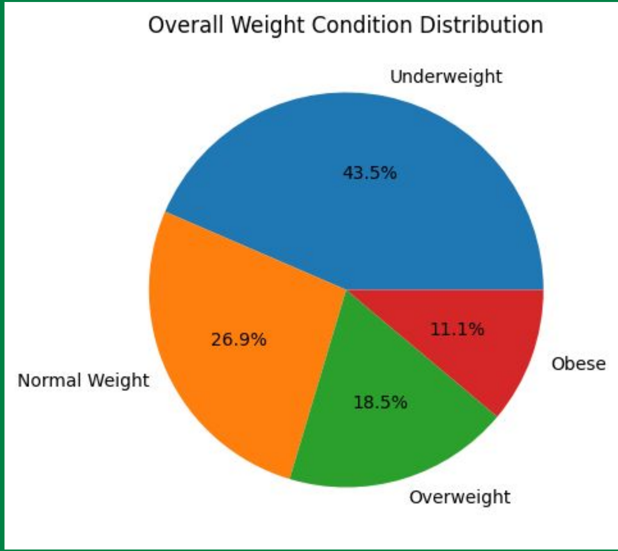
```
d.head()
```

| | ID | Age | Gender | Height | Weight | BMI | Label |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 25 | 1 | 175 | 80 | 25.3 | 0 |
| 1 | 2 | 30 | 0 | 160 | 60 | 22.5 | 0 |
| 2 | 3 | 35 | 1 | 180 | 90 | 27.3 | 2 |
| 3 | 4 | 40 | 0 | 150 | 50 | 20.0 | 3 |
| 4 | 5 | 45 | 1 | 190 | 100 | 31.2 | 1 |

# Visualizing Data

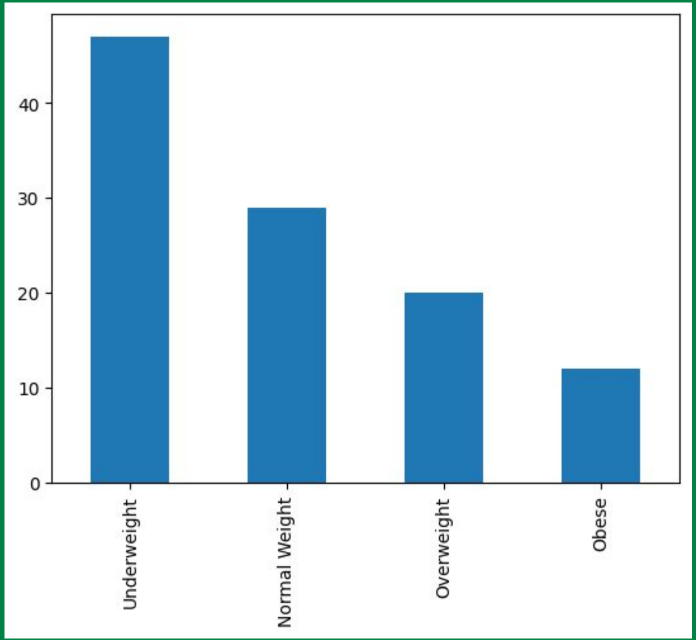
Analyzing distributions

```
plt.figure(figsize=(8,5))

# 0 = Normal Weight
# 1 = Obese
# 2 = Overweight
# 3 = Underweight

plt.title('Overall Weight Condition Distribution')
class_count = d['Label'].value_counts()
vals = class_count.values
labels = class_count.index.tolist()
string_labels = [decoded_labels[label] for label in labels]

plt.pie(vals, labels=string_labels, autopct='%1.1f%%')

plt.show()
```



# Visualizing Data

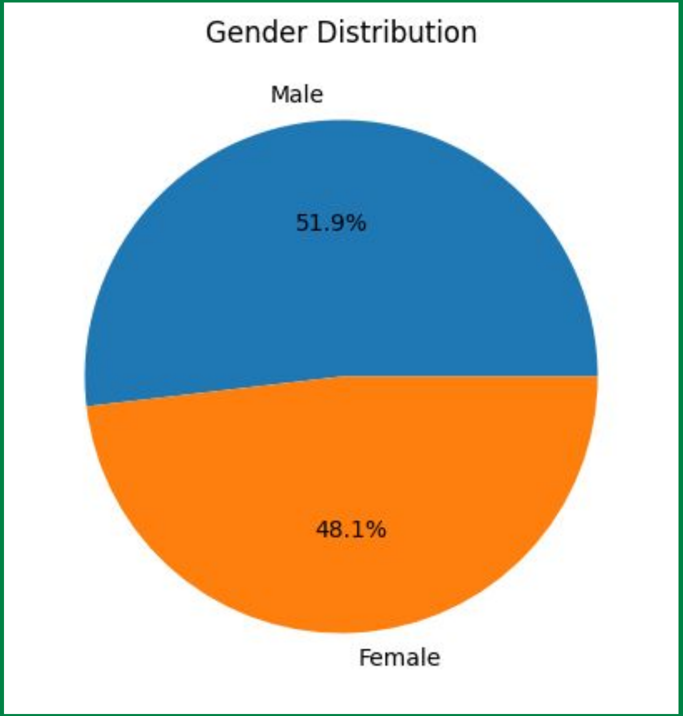
```
d['Label'].value_counts().plot(kind='bar')
```



# Visualizing Data

Analyzing distributions

```
plt.title('Gender Distribution')
vals = d['Gender'].value_counts().values
labels = ['Male', 'Female']

plt.pie(vals, labels=labels, autopct='%1.1f%%')
```



# Visualizing Data

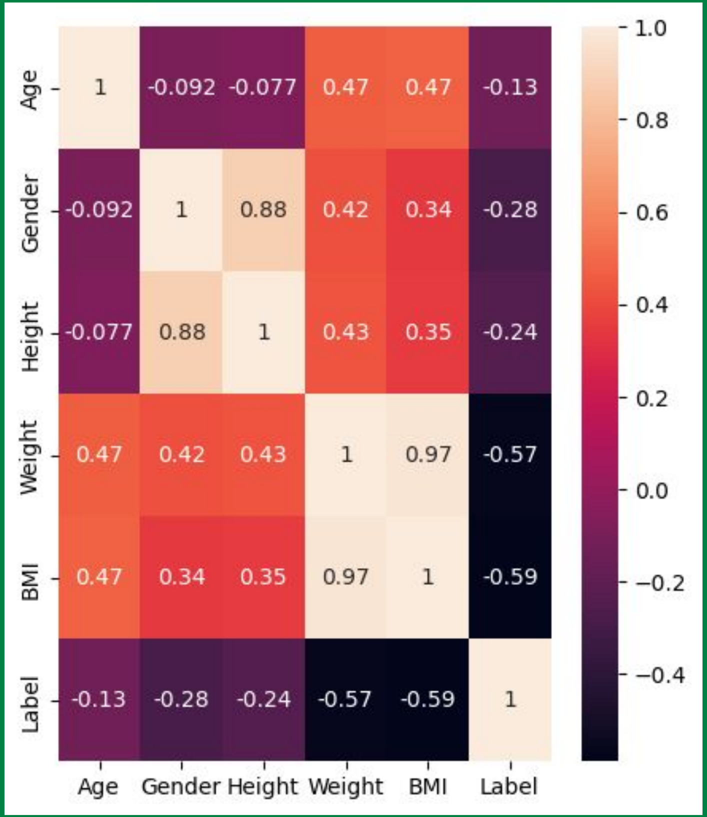
```
d[cols_to_plot].corr()
```

| | Age | Gender | Height | Weight | BMI | Label |
|---|---|---|---|---|---|---|
| **Age** | 1.000000 | -0.091964 | -0.076896 | 0.465106 | 0.474185 | -0.134396 |
| **Gender** | -0.091964 | 1.000000 | 0.876225 | 0.418415 | 0.342342 | -0.281647 |
| **Height** | -0.076896 | 0.876225 | 1.000000 | 0.428890 | 0.354340 | -0.237683 |
| **Weight** | 0.465106 | 0.418415 | 0.428890 | 1.000000 | 0.972829 | -0.565555 |
| **BMI** | 0.474185 | 0.342342 | 0.354340 | 0.972829 | 1.000000 | -0.589237 |
| **Label** | -0.134396 | -0.281647 | -0.237683 | -0.565555 | -0.589237 | 1.000000 |

Analyzing correlation between attributes

Visualized using heatmap →

```python
cols_to_plot = ['Age', 'Gender', 'Height', 'Weight', 'BMI', 'Label']
```




```python
plt.figure(figsize=(5,6))
hm = sns.heatmap(d[cols_to_plot].corr(), annot=True)
```

**Observation:**

High correlation

- BMI <-> Weight
- Height <-> Gender

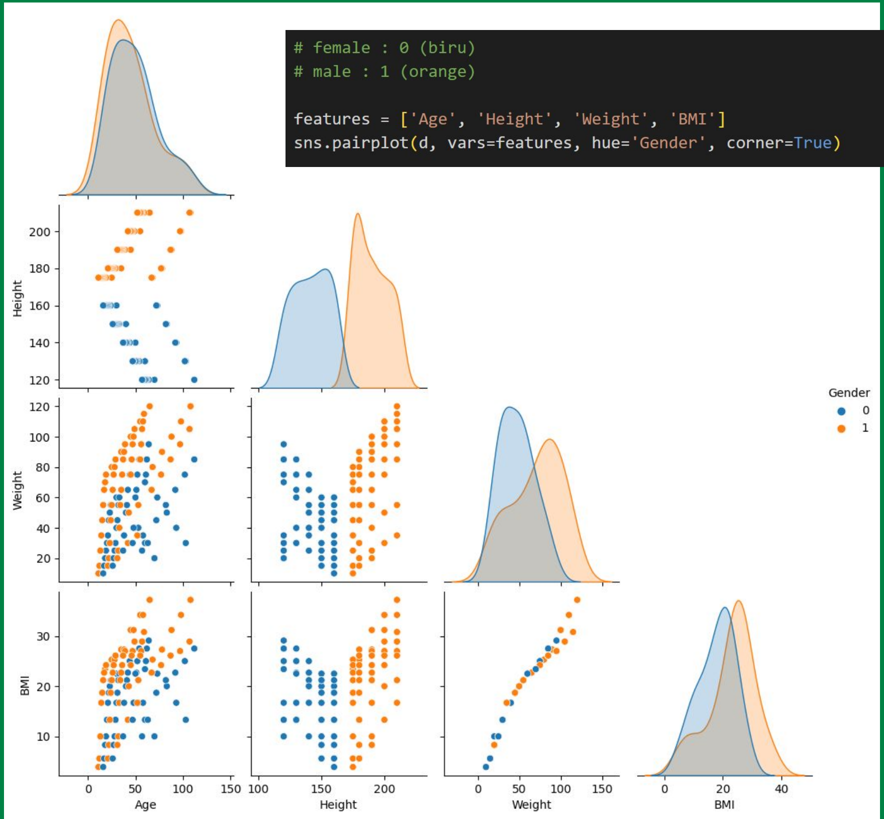
Fair correlation

- Weight <-> Age, Gender, Height
- BMI <-> Age
- Age <-> Weight, BMI
- Gender <-> Weight
- Height <-> Weight

Little correlation

- BMI <-> Gender, Height
- Gender <-> BMI
- Height <-> BMI

- Female: 0 (blue)
- Male: 1 (orange)




```python
# female : 0 (biru)
# male : 1 (orange)

features = ['Age', 'Height', 'Weight', 'BMI']
sns.pairplot(d, vars=features, hue='Gender', corner=True)
```

# Visualizing Data

Comparing feature values based on Gender

## Observation:

- The heavier you weigh the greater the BMI
- Males are taller on average
- Weight & BMI have a high correlation
- Age & Weight / Age & BMI have somewhat a correlation
- Data that has no correlation:
  - Height & Weight
  - Height & BMI
  - Age & Height
  - Gender & BMI

# 03 Machine Learning Modeling

# Libraries

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

Used for machine learning modelling

# Preprocessing Data

```
d.drop('ID', axis=1, inplace=True)
```

```
X = d.drop(["Label"],axis=1)
y = d["Label"]
```

- Removing unused columns

```
       Age  Gender  Height  Weight   BMI
0       25       1     175      80  25.3
1       30       0     160      60  22.5
2       35       1     180      90  27.3
3       40       0     150      50  20.0
4       45       1     190     100  31.2
..     ...     ...     ...     ...   ...
103     11       1     175      10   3.9
104     16       0     160      10   3.9
105     21       1     180      15   5.6
106     26       0     150      15   5.6
107     31       1     190      20   8.3
```

# Splitting Data

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

# Decision Tree

## Classification

```
# training data
dtc = DecisionTreeClassifier()

# train
dtc.fit(X_train, y_train)

# predict labels for the test data
y_pred = dtc.predict(X_test)
```

# Why Decision Tree?

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

Evaluating accuracy of Decision Tree Classification as the chosen model

# Why Decision Tree?

Using classification report

```
print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       1.00      1.00      1.00         4
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         8

    accuracy                           1.00        22
   macro avg       1.00      1.00      1.00        22
weighted avg       1.00      1.00      1.00        22
```

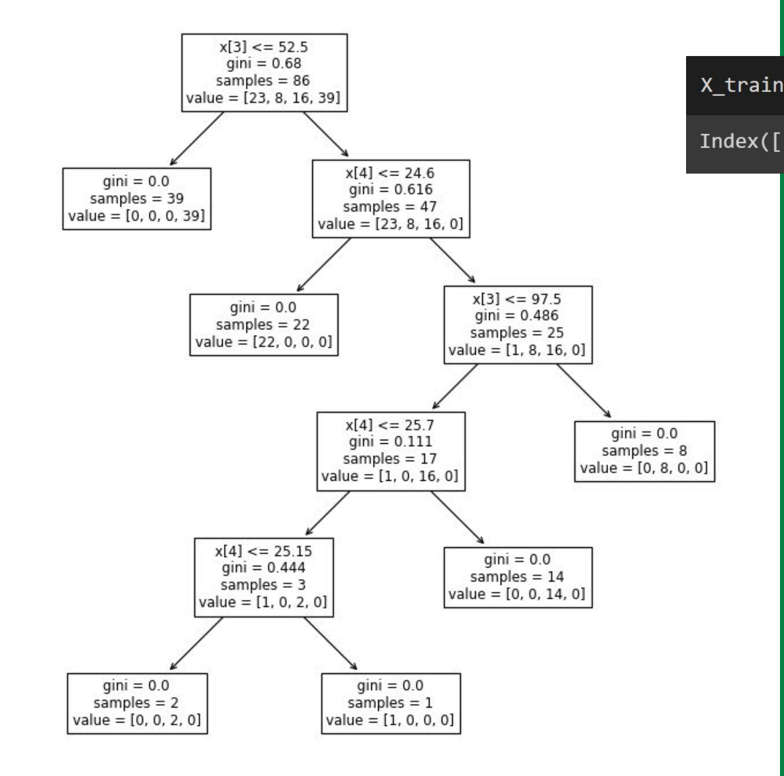
# 04

# Conclusion

# Visualizing Decision Tree

```
plt.rcParams['figure.dpi'] = 85
plt.subplots(figsize=(10, 10))
tree.plot_tree(dtc, fontsize=10)
plt.show()
```

# Visualizing Decision Tree



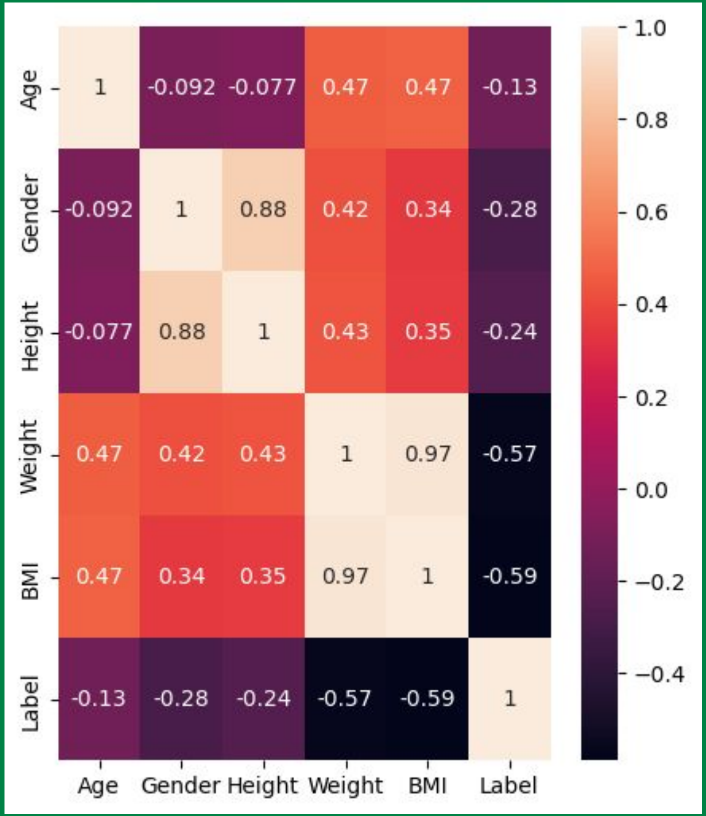

```
X_train.columns
```

```
Index(['Age', 'Gender', 'Height', 'Weight', 'BMI'], dtype='object')
```

- x[3] = Weight
- x[4] = BMI

- Gini impurity
- Samples (before data is split)

*Remember:




# Visualizing Data

Analyzing correlation between attributes

High correlation

- BMI <-> Weight
- Height <-> Gender

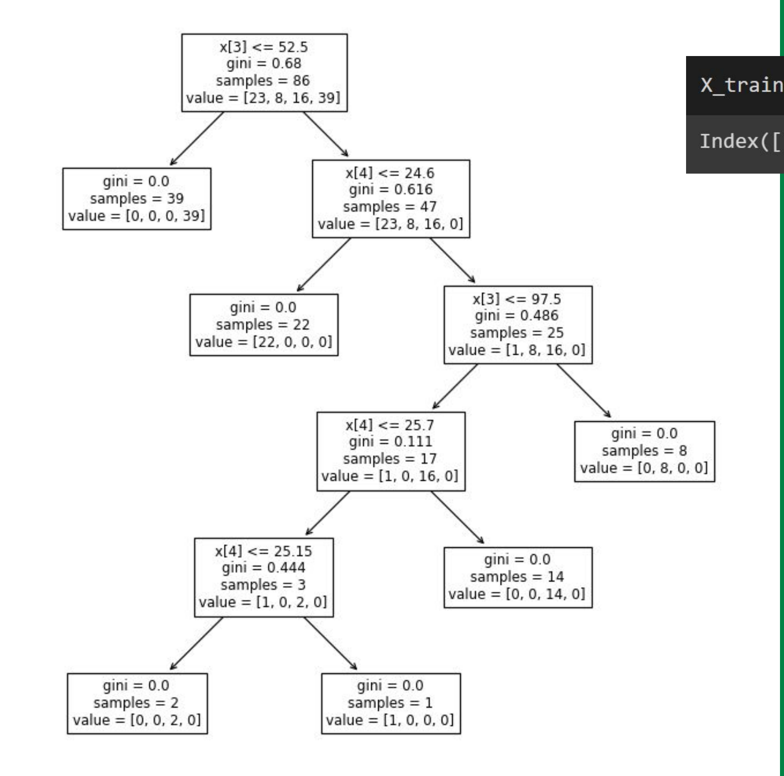
Fair correlation

- Weight <-> Age, Gender, Height
- BMI <-> Age
- Age <-> Weight, BMI
- Gender <-> Weight
- Height <-> Weight

Little correlation

- BMI <-> Gender, Height
- Gender <-> BMI
- Height <-> BMI

# Visualizing Decision Tree




```
X_train.columns
```

```
Index(['Age', 'Gender', 'Height', 'Weight', 'BMI'], dtype='object')
```

- x[3] = Weight
- x[4] = BMI

# Evaluating Trained Model

```
X_train.columns
```

```
Index(['Age', 'Gender', 'Height', 'Weight', 'BMI'], dtype='object')
```

```
new_data = [[30, 1, 180, 20, 27.3], [60, 1, 170, 55, 20]]
preds = dtc.predict(new_data)
preds

# 0 = Normal Weight
# 1 = Obese
# 2 = Overweight
# 3 = Underweight
```

Output:

```
array([3, 0])
```

# Main Summary Points

- Weight and BMI are the most influential attributes used to classify obesity data.
- Using machine learning models is very beneficial to help classify obesity data.
- Decision Tree Classification is a highly effective machine learning model to use for the chosen dataset.

# Thank You!

Tugas Besar Pengantar Kecerdasan Buatan / Introduction to AI Final Project

- 1972034 Leona Rose
- 1972042 Neng Linda Rahayu