Simple Linear Regression

In [ ]:

```
# Import libraries
import numpy as np
import pandas as pd
```

Import Dataset

In [ ]:

```
# Load data
df = pd.read_csv('../datasets/mldata2.csv')
df.head()
```

Out[ ]:

	age	height	weight	gender	likeness
0	27	170.688	76.0	Male	Biryani
1	41	165.000	70.0	Male	Biryani
2	29	171.000	80.0	Male	Biryani
3	27	173.000	102.0	Male	Biryani
4	29	164.000	67.0	Male	Biryani

Data Cleaning

In [ ]:

```
# Find missing values
df.isnull().sum()
```

Out[ ]:

```
age         0
height      0
weight      0
gender      0
likeness    0
dtype: int64
```

In [ ]:

```
# Data information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245 entries, 0 to 244
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       245 non-null    int64  
 1   height    245 non-null    float64
 2   weight    245 non-null    float64
 3   gender    245 non-null    object 
 4   likeness  245 non-null    object 
dtypes: float64(2), int64(1), object(2)
memory usage: 9.7+ KB
```

In [ ]:

```
# Convert categorical columns (gender , likeness) data type to numeric
from sklearn.preprocessing import LabelEncoder
df.gender = LabelEncoder().fit_transform(df.gender)
df.likeness = LabelEncoder().fit_transform(df.likeness)
```

In [ ]:

```
# View dataset
df.head()
```

Out[ ]:

	age	height	weight	gender	likeness
0	27	170.688	76.0	1	0
1	41	165.000	70.0	1	0
2	29	171.000	80.0	1	0
3	27	173.000	102.0	1	0
4	29	164.000	67.0	1	0

In [ ]:

```
# Unique values in 'gender' and 'likeness'
print('Unique values in gender: {}'.format(pd.unique(df['gender'])))
print('Unique values in likeness: {}'.format(pd.unique(df['likeness'])))
```

```
Unique values in gender: [1 0]
Unique values in likeness: [0 1 2]
```

In [ ]:

```
# Check information and data type again
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245 entries, 0 to 244
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       245 non-null    int64  
 1   height    245 non-null    float64
 2   weight    245 non-null    float64
 3   gender    245 non-null    int32  
 4   likeness  245 non-null    int32  
dtypes: float64(2), int32(2), int64(1)
memory usage: 7.8 KB
```

Split Data into Train and Test

In [ ]:

```
# Split data into input (X) and output (y)
X = df[['age', 'height', 'weight', 'gender']]
y = df['likeness']
```

In [ ]:

Out[ ]:

	age	height	weight	gender
0	27	170.688	76.0	1
1	41	165.000	70.0	1
2	29	171.000	80.0	1
3	27	173.000	102.0	1
4	29	164.000	67.0	1

In [ ]:

```
y.head()
```

Out[ ]:

```
0    0
1    0
2    0
3    0
4    0
Name: likeness, dtype: int32
```

In [ ]:

```
# Split data into train and test
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=0)
```

In [ ]:

```
# View training dataset
print(f'Training inputs:\n{X_train.head()}')
print(f'Training outputs:\n{y_train.head()}')
print(f'Training input shape: {X_train.shape}')
print(f'Training input shape: {y_train.shape}')
```

```
Training inputs:
     age  height  weight  gender
15    27   174.0    78.0       1
158   22   181.0    67.0       1
7     34   176.5    98.0       1
159   27   152.4    75.0       1
207   28     5.3    62.0       0
Training outputs:
15     2
158    0
7      0
159    0
207    0
Name: likeness, dtype: int32
Training input shape: (196, 4)
Training input shape: (196,)
```

Fit Linear Regression Model

In [ ]:

```
from sklearn.linear_model import LinearRegression
# Create model
model = LinearRegression()
# Fit model
model = model.fit(X_train, y_train)
print('Training completed!!')
```

```
Training completed!!
```

Make Predictions

In [ ]:

```
y_preds = model.predict(X_test)
y_preds[:10] # first 10 predictions
```

Out[ ]:

```
array([0.54849352, 0.46098079, 0.53362028, 0.5515754 , 0.71265416,
       0.55948045, 0.51826946, 0.50782544, 0.3327458 , 0.56214246])
```

Model Evaluation

In [ ]:

```
# Find MAE and MSE on models predictions with test labels
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error

mae = mean_absolute_error(y_test, y_preds)
mse = mean_squared_error(y_test, y_preds)
print(f'Model MAE score: {mae}')
print(f'Model MSE score: {mse}')
```

```
Model MAE score: 0.6665392013180987
Model MSE score: 0.5918437171374316
```