Understand the data

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

```
df = pd.read_csv('/content/Mall_Customers.csv')
df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

```
df.shape
```

```
(200, 5)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

```
df.describe()
```

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

```
df.isnull().any()
```

```
CustomerID              False
Gender                  False
Age                     False
Annual Income (k$)      False
Spending Score (1-100)  False
dtype: bool
```

```
df.isnull().sum()
```

```
CustomerID              0
Gender                  0
Age                     0
Annual Income (k$)      0
Spending Score (1-100)  0
dtype: int64
```

```
df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77

Data Preprocessing

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
df.Gender = le.fit_transform(df.Gender)
```

```
df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	1	19	15	39
1	2	1	21	15	81
2	3	0	20	16	6
3	4	0	23	16	77
4	5	0	31	17	40

```
x = df.drop(columns = ['Gender'], axis = 1)
x.head()
```

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19	15	39
1	2	21	15	81
2	3	20	16	6
3	4	23	16	77
4	5	31	17	40

```
y = df.Gender
y.head()
```

```
0    1
1    1
2    0
3    0
4    0
Name: Gender, dtype: int64
```

```
from sklearn.preprocessing import MinMaxScaler
scale = MinMaxScaler()
x_scaled = pd.DataFrame(scale.fit_transform(x),columns=x.columns)
x_scaled.head()
```

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
0	0.000000	0.019231	0.000000	0.387755
1	0.005025	0.057692	0.000000	0.816327
2	0.010050	0.038462	0.008197	0.051020
3	0.015075	0.096154	0.008197	0.775510
4	0.020101	0.250000	0.016393	0.397959

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_scaled,y,test_size = 0.2, random_state= 0)
```

```
x_train.shape
x_test.shape
```

```
(40, 4)
```

```
x_train.head()
```

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
134	0.673367	0.038462	0.475410	0.040816
66	0.331658	0.480769	0.270492	0.500000
26	0.130653	0.519231	0.106557	0.316327
113	0.567839	0.019231	0.401639	0.459184
168	0.844221	0.346154	0.590164	0.265306

```
y_train.head()
```

```
134    1
66     0
26     0
113    1
168    0
Name: Gender, dtype: int64
```

```
new_df = df.iloc[:,:-1]
new_df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)
0	1	1	19	15
1	2	1	21	15
2	3	0	20	16
3	4	0	23	16
4	5	0	31	17

Machine Learning approach with clustering algorithm

```
from sklearn import cluster
```

```
error = []
for i in range(1,11):
  kmeans = cluster.KMeans(n_clusters = i,init = 'k-means++', random_state = 0)
  kmeans.fit(new_df)
  pred = km_model.predict(new_df)
  error.append(kmeans.inertia_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
```

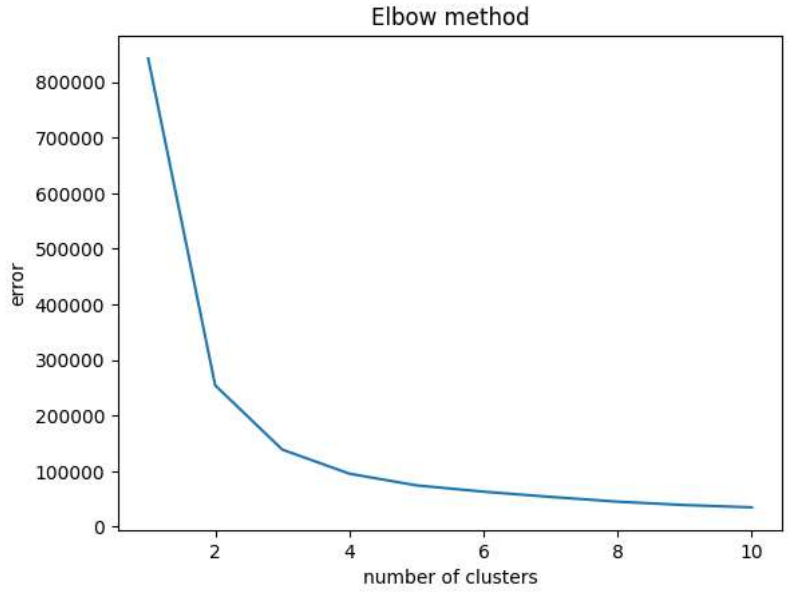
```
error
```

```
[842808.06,
 254375.02790279023,
 138716.48711070663,
 95392.76089176608,
 74339.55121941707,
 63023.88081677608,
 53573.00760399022,
 44998.052643910836,
```

```
 38962.28138017457,
 34773.099870413454]
```

```
import matplotlib.pyplot as plt
```

```
plt.plot(range(1,11),error)
plt.title("Elbow method")
plt.xlabel('number of clusters')
plt.ylabel('error')
plt.show()
```




```
km_model = cluster.KMeans(n_clusters=3,init = 'k-means++',random_state=0)
```

```
km_model.fit(new_df)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_
  warnings.warn(
```

```
KMeans(n_clusters=3, random_state=0)
```

```
pred = km_model.predict(new_df)
pred
```

```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0], dtype=int32)
```

```
km_model.predict([[1.1,2.2,4.3,4.4]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KMeans was fitte
  warnings.warn(
array([2], dtype=int32)
```

```
km_model.predict([[2.2,2.0,1.1,1.2]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KMeans was fitte
  warnings.warn(
array([2], dtype=int32)
```