Load Dataset

```
import pandas as pd
import numpy as np
df=pd.read_csv(r'C:\Users\Gokul\Downloads\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

Univariate Analysis Visualization

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.plot(df['Annual Income (k$)'])
plt.show()
```



```
plt.hist(df['Annual Income (k$)'])
(array([24., 22., 28., 38., 30., 36.,  8.,  6.,  4.,  4.]),
 array([ 15. ,  27.2,  39.4,  51.6,  63.8,  76. ,  88.2, 100.4, 112.6,
        124.8, 137. ]),
 )
```



```
data=np.array(df['Annual Income (k$)'])
```

```
plt.plot(data,linestyle = 'dotted')
```

```
[]
```

```
sns.boxplot(df['Age'])
```

```
C:\Users\govin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
sns.countplot(df['Age'])
```

```
C:\Users\govin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
sns.countplot(df['Gender'])
```

```
C:\Users\govin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
df['Age'].plot(kind='density')
```

Bivariate Analysis Visualization

```
sns.stripplot(x=df['Annual Income (k$)'],y=df['Spending Score (1-100)'])
```

```
plt.scatter(df['Annual Income (k$)'],df['Age'],color='pink')
plt.xlabel("Annual Income (k$)")
plt.ylabel("Age")
Text(0, 0.5, 'Age')

sns.stripplot(x=df['Annual Income (k$)'],y=df['Age'])

sns.violinplot(x ='Annual Income (k$)', y ='Spending Score (1-100)', data = df)
```

Multivariate Analysis Visualization

```
sns.pairplot(df)

sns.heatmap(df.corr(),annot=True)
```

Descriptive Statistics

```
df.shape
(200, 5)
df.info()
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.isnull().sum()

CustomerID 0

Gender 0

Age 0

Annual Income (k$) 0

Spending Score (1-100) 0

dtype: int64

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.mean()

CustomerID 100.50

Age 38.85

Annual Income (k$) 60.56

Spending Score (1-100) 50.20

dtype: float64

df['Age'].mean()

38.85

df.mode()

CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	0.0	32.0	54.0	42.0
1	2	NaN	NaN	78.0	NaN
2	3	NaN	NaN	NaN	NaN
3	4	NaN	NaN	NaN	NaN
4	5	NaN	NaN	NaN	NaN
...	...	...	...	...	...
195	196	NaN	NaN	NaN	NaN
196	197	NaN	NaN	NaN	NaN
197	198	NaN	NaN	NaN	NaN
198	199	NaN	NaN	NaN	NaN
199	200	NaN	NaN	NaN	NaN

200 rows × 5 columns

df.median()

CustomerID 100.5

Age 36.0

Annual Income (k$) 61.5

Spending Score (1-100) 50.0

dtype: float64

```
df['Gender'].value_counts()

Female        112

Male           88

Name: Gender, dtype: int64
```

Handle Missing Values

```
df.isna().sum()

CustomerID                    0

Gender                        0

Age                           0

Annual Income (k$)            0

Spending Score (1-100)        0

dtype: int64
```

Handling Outliers

```
sns.boxplot(df['Spending Score (1-100)'])
```

```
C:\Users\govin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

  warnings.warn(
```

```
Q1 = df['Spending Score (1-100)'].quantile(0.25)

Q3 = df['Spending Score (1-100)'].quantile(0.75)

IQR = Q3 - Q1

whisker_width = 1.5

lower_whisker = Q1 -(whisker_width*IQR)

upper_whisker = Q3 +(whisker_width*IQR)

df['Spending Score (1-100)']=np.where(df['Spending Score
```

(1-100)']>upper_whisker,upper_whisker,np.where(df['Spending Score (1-100)']<lower_whisker,lower_whisker,df['Spending Score (1-100)']))

sns.boxplot(df['Spending Score (1-100)'])

C:\Users\govin\anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

  warnings.warn(

Categorical Variable and Encoding

numeric_data = df.select_dtypes(include=[np.number])

categorical_data = df.select_dtypes(exclude=[np.number])

print("Number of numerical variables: ", numeric_data.shape[1])

print("Number of categorical variables: ", categorical_data.shape[1])

Number of numerical variables: 4

Number of categorical variables: 1

print("Number of categorical variables: ", categorical_data.shape[1])

Categorical_variables = list(categorical_data.columns)

Categorical_variables

Number of categorical variables: 1

['Gender']

df['Gender'].value_counts()

Female 112

Male 88

Name: Gender, dtype: int64

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

```
label = le.fit_transform(df['Gender'])

df["Gender"] = label

df['Gender'].value_counts()

0    112

1     88

Name: Gender, dtype: int64

df.head()
```

CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	1	19	15	39.0
1	2	1	21	15	81.0
2	3	0	20	16	6.0
3	4	0	23	16	77.0
4	5	0	31	17	40.0

Independent and Dependent Variables

```
X = df.drop("Spending Score (1-100)",axis=1)

Y = df['Spending Score (1-100)']

X[:5]
```

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

```
Y[:5]

0    39.0
```

1 81.0

2 6.0

3 77.0

4 40.0

Name: Spending Score (1-100), dtype: float64

Scale Independent Variables

X

	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
...	...	...	...	...
195	196	0	35	120
196	197	0	45	126
197	198	1	32	126
198	199	1	32	137
199	200	1	30	137

200 rows × 4 columns

```
from sklearn.preprocessing import StandardScaler
object= StandardScaler()
scale = object.fit_transform(X)
print(scale)
```

```
[[-1.7234121       1.12815215 -1.42456879 -1.73899919]
 [-1.70609137     1.12815215 -1.28103541 -1.73899919]
 [-1.68877065 -0.88640526 -1.3528021     -1.70082976]
 [-1.67144992 -0.88640526 -1.13750203 -1.70082976]
 [-1.6541292     -0.88640526 -0.56336851 -1.66266033]
 [-1.63680847 -0.88640526 -1.20926872 -1.66266033]
 [-1.61948775 -0.88640526 -0.27630176 -1.62449091]
 [-1.60216702 -0.88640526 -1.13750203 -1.62449091]
 [-1.5848463       1.12815215     1.80493225 -1.58632148]
 [-1.56752558 -0.88640526 -0.6351352     -1.58632148]
 [-1.55020485     1.12815215     2.02023231 -1.58632148]
 [-1.53288413 -0.88640526 -0.27630176 -1.58632148]
 [-1.5155634     -0.88640526     1.37433211 -1.54815205]
 [-1.49824268 -0.88640526 -1.06573534 -1.54815205]
 [-1.48092195     1.12815215 -0.13276838 -1.54815205]
 [-1.46360123     1.12815215 -1.20926872 -1.54815205]
 [-1.4462805     -0.88640526 -0.27630176 -1.50998262]
 [-1.42895978     1.12815215 -1.3528021     -1.50998262]
 [-1.41163905     1.12815215     0.94373197 -1.43364376]
 [-1.39431833 -0.88640526 -0.27630176 -1.43364376]
 [-1.3769976       1.12815215 -0.27630176 -1.39547433]
 [-1.35967688     1.12815215 -0.99396865 -1.39547433]
 [-1.34235616 -0.88640526     0.51313183 -1.3573049 ]
 [-1.32503543     1.12815215 -0.56336851 -1.3573049 ]
 [-1.30771471 -0.88640526     1.08726535 -1.24279661]
```

[-1.29039398 1.12815215 -0.70690189 -1.24279661]

[-1.27307326 -0.88640526 0.44136514 -1.24279661]

[-1.25575253 1.12815215 -0.27630176 -1.24279661]

[-1.23843181 -0.88640526 0.08253169 -1.20462718]

[-1.22111108 -0.88640526 -1.13750203 -1.20462718]

[-1.20379036 1.12815215 1.51786549 -1.16645776]

[-1.18646963 -0.88640526 -1.28103541 -1.16645776]

[-1.16914891 1.12815215 1.01549866 -1.05194947]

[-1.15182818 1.12815215 -1.49633548 -1.05194947]

[-1.13450746 -0.88640526 0.7284319 -1.05194947]

[-1.11718674 -0.88640526 -1.28103541 -1.05194947]

[-1.09986601 -0.88640526 0.22606507 -1.01378004]

[-1.08254529 -0.88640526 -0.6351352 -1.01378004]

[-1.06522456 -0.88640526 -0.20453507 -0.89927175]

[-1.04790384 -0.88640526 -1.3528021 -0.89927175]

[-1.03058311 -0.88640526 1.87669894 -0.86110232]

[-1.01326239 1.12815215 -1.06573534 -0.86110232]

[-0.99594166 1.12815215 0.65666521 -0.82293289]

[-0.97862094 -0.88640526 -0.56336851 -0.82293289]

[-0.96130021 -0.88640526 0.7284319 -0.82293289]

[-0.94397949 -0.88640526 -1.06573534 -0.82293289]

[-0.92665877 -0.88640526 0.80019859 -0.78476346]

[-0.90933804 -0.88640526 -0.85043527 -0.78476346]

[-0.89201732 -0.88640526 -0.70690189 -0.78476346]

[-0.87469659 -0.88640526 -0.56336851 -0.78476346]

[-0.85737587 -0.88640526 0.7284319 -0.70842461]

[-0.84005514 1.12815215 -0.41983513 -0.70842461]

[-0.82273442 -0.88640526 -0.56336851 -0.67025518]

[-0.80541369 1.12815215 1.4460988 -0.67025518]

[-0.78809297 -0.88640526 0.80019859 -0.67025518]

[-0.77077224 1.12815215 0.58489852 -0.67025518]

[-0.75345152 -0.88640526 0.87196528 -0.63208575]

[-0.73613079 1.12815215 2.16376569 -0.63208575]

[-0.71881007 -0.88640526 -0.85043527 -0.55574689]

[-0.70148935 1.12815215 1.01549866 -0.55574689]

[-0.68416862 1.12815215 2.23553238 -0.55574689]

[-0.6668479   1.12815215 -1.42456879 -0.55574689]

[-0.64952717 -0.88640526 2.02023231 -0.51757746]

[-0.63220645 -0.88640526 1.08726535 -0.51757746]

[-0.61488572 1.12815215 1.73316556 -0.47940803]

[-0.597565   1.12815215 -1.49633548 -0.47940803]

[-0.58024427 -0.88640526 0.29783176 -0.47940803]

[-0.56292355 -0.88640526 2.091999   -0.47940803]

[-0.54560282 1.12815215 -1.42456879 -0.47940803]

[-0.5282821 -0.88640526 -0.49160182 -0.47940803]

[-0.51096138 1.12815215 2.23553238 -0.4412386]

[-0.49364065 -0.88640526 0.58489852 -0.4412386]

[-0.47631993 -0.88640526 1.51786549 -0.40306917]

[-0.4589992 -0.88640526 1.51786549 -0.40306917]

[-0.44167848 1.12815215 1.4460988 -0.25039146]

[-0.42435775 1.12815215 -0.92220196 -0.25039146]

[-0.40703703 -0.88640526 0.44136514 -0.25039146]

[-0.3897163   1.12815215 0.08253169 -0.25039146]

[-0.37239558 -0.88640526 -1.13750203 -0.25039146]

[-0.35507485 -0.88640526 0.7284319 -0.25039146]

[-0.33775413 1.12815215 1.30256542 -0.25039146]

[-0.3204334   1.12815215 -0.06100169 -0.25039146]

[-0.30311268 1.12815215 2.02023231 -0.25039146]

[-0.28579196 -0.88640526 0.51313183 -0.25039146]

[-0.26847123 -0.88640526 -1.28103541 -0.25039146]

[-0.25115051 1.12815215 0.65666521 -0.25039146]

[-0.23382978 -0.88640526 1.15903204 -0.13588317]

[-0.21650906 -0.88640526 -1.20926872 -0.13588317]

[-0.19918833 -0.88640526 -0.34806844 -0.09771374]

[-0.18186761 -0.88640526 0.80019859 -0.09771374]

[-0.16454688 -0.88640526 2.091999   -0.05954431]

[-0.14722616 1.12815215 -1.49633548 -0.05954431]

[-0.12990543 1.12815215 0.65666521 -0.02137488]

[-0.11258471 -0.88640526 0.08253169 -0.02137488]

[-0.09526399 -0.88640526 -0.49160182 -0.02137488]

[-0.07794326 1.12815215 -1.06573534 -0.02137488]

[-0.06062254 -0.88640526 0.58489852 -0.02137488]

[-0.04330181 -0.88640526 -0.85043527 -0.02137488]

[-0.02598109 1.12815215 0.65666521 0.01679455]

[-0.00866036 1.12815215 -1.3528021   0.01679455]

[0.00866036 -0.88640526 -1.13750203 0.05496398]

[0.02598109 -0.88640526 0.7284319   0.05496398]

[0.04330181 1.12815215 2.02023231 0.05496398]

[0.06062254 1.12815215 -0.92220196 0.05496398]

[0.07794326 1.12815215 0.7284319   0.05496398]

[0.09526399 -0.88640526 -1.28103541 0.05496398]

[0.11258471 -0.88640526 1.94846562 0.09313341]

[0.12990543 1.12815215 1.08726535 0.09313341]

[0.14722616 1.12815215 2.091999   0.09313341]

[0.16454688 1.12815215 1.94846562 0.09313341]

[0.18186761 1.12815215 1.87669894 0.09313341]

[0.19918833 -0.88640526 -1.42456879 0.09313341]

[0.21650906 -0.88640526 -0.06100169 0.13130284]

[0.23382978 1.12815215 -1.42456879 0.13130284]

[0.25115051 -0.88640526 -1.49633548 0.16947227]

[0.26847123 -0.88640526 -1.42456879 0.16947227]

[0.28579196 -0.88640526 1.73316556 0.16947227]

[0.30311268 -0.88640526 0.7284319   0.16947227]

[0.3204334 -0.88640526 0.87196528 0.24581112]

[0.33775413 -0.88640526 0.80019859 0.24581112]

[0.35507485 1.12815215 -0.85043527 0.24581112]

[0.37239558 -0.88640526 -0.06100169 0.24581112]

[0.3897163 -0.88640526 0.08253169 0.32214998]

[0.40703703 1.12815215 0.010765   0.32214998]

[0.42435775 -0.88640526 -1.13750203 0.36031941]

[0.44167848 -0.88640526 -0.56336851 0.36031941]

[0.4589992   1.12815215 0.29783176 0.39848884]

[0.47631993 1.12815215 0.08253169 0.39848884]

[0.49364065 1.12815215 1.4460988   0.39848884]

[0.51096138 1.12815215 -0.06100169 0.39848884]

[0.5282821   1.12815215 0.58489852 0.39848884]

[0.54560282 1.12815215 0.010765   0.39848884]

[0.56292355 -0.88640526 -0.99396865 0.43665827]

[0.58024427 -0.88640526 -0.56336851 0.43665827]

[0.597565   1.12815215 -1.3528021   0.4748277]

[0.61488572 -0.88640526 -0.70690189 0.4748277]

[0.63220645 -0.88640526 0.36959845 0.4748277]

[0.64952717 1.12815215 -0.49160182 0.4748277]

[0.6668479   1.12815215 -1.42456879 0.51299713]

[0.68416862 -0.88640526 -0.27630176 0.51299713]

[0.70148935 -0.88640526 1.30256542 0.55116656]

[0.71881007 1.12815215 -0.49160182 0.55116656]

[0.73613079 -0.88640526 -0.77866858 0.58933599]

[0.75345152 -0.88640526 -0.49160182 0.58933599]

[0.77077224 1.12815215 -0.99396865 0.62750542]

[0.78809297 1.12815215 -0.77866858 0.62750542]

[0.80541369 1.12815215 0.65666521 0.62750542]

[0.82273442 -0.88640526 -0.49160182 0.62750542]

[0.84005514 -0.88640526 -0.34806844 0.66567484]

[0.85737587 1.12815215 -0.34806844 0.66567484]

[0.87469659 1.12815215 0.29783176 0.66567484]

[0.89201732 1.12815215 0.010765   0.66567484]

[0.90933804 -0.88640526 0.36959845 0.66567484]

[0.92665877 -0.88640526 -0.06100169 0.66567484]

[0.94397949 -0.88640526 0.58489852 0.66567484]

[0.96130021 -0.88640526 -0.85043527 0.66567484]

[0.97862094 1.12815215 -0.13276838 0.66567484]

[0.99594166 -0.88640526 -0.6351352   0.66567484]

[1.01326239 1.12815215 -0.34806844 0.66567484]

[1.03058311 -0.88640526 -0.6351352   0.66567484]

[1.04790384 -0.88640526 1.23079873 0.70384427]

[1.06522456 -0.88640526 -0.70690189 0.70384427]

[1.08254529 1.12815215 -1.42456879 0.78018313]

[1.09986601 -0.88640526 -0.56336851 0.78018313]

[1.11718674 1.12815215 0.80019859 0.93286085]

[1.13450746 -0.88640526 -0.20453507 0.93286085]

[1.15182818 1.12815215 0.22606507 0.97103028]

[1.16914891 -0.88640526 -0.41983513 0.97103028]

[1.18646963 -0.88640526 -0.20453507 1.00919971]

[1.20379036 1.12815215 -0.49160182 1.00919971]

[1.22111108 1.12815215 0.08253169 1.00919971]

[1.23843181 1.12815215 -0.77866858 1.00919971]

[1.25575253 1.12815215 -0.20453507 1.00919971]

[1.27307326 1.12815215 -0.20453507 1.00919971]

[1.29039398 -0.88640526 0.94373197 1.04736914]

[1.30771471 -0.88640526 -0.6351352 1.04736914]

[1.32503543 1.12815215 1.37433211 1.04736914]

[1.34235616 1.12815215 -0.85043527 1.04736914]

[1.35967688 1.12815215 1.4460988 1.23821628]

[1.3769976 1.12815215 -0.27630176 1.23821628]

[1.39431833 -0.88640526 -0.13276838 1.390894]

[1.41163905 -0.88640526 -0.49160182 1.390894]

[1.42895978 1.12815215 0.51313183 1.42906343]

[1.4462805 -0.88640526 -0.70690189 1.42906343]

[1.46360123 -0.88640526 0.15429838 1.46723286]

[1.48092195 1.12815215 -0.6351352 1.46723286]

[1.49824268 -0.88640526 1.08726535 1.54357172]

[1.5155634 1.12815215 -0.77866858 1.54357172]

[1.53288413 -0.88640526 0.15429838 1.61991057]

[1.55020485 -0.88640526 -0.20453507 1.61991057]

[1.56752558 -0.88640526 -0.34806844 1.61991057]

[1.5848463 -0.88640526 -0.49160182 1.61991057]

[1.60216702 1.12815215 -0.41983513 2.00160487]

[1.61948775 -0.88640526 -0.06100169 2.00160487]

[1.63680847 -0.88640526 0.58489852 2.26879087]

[1.6541292 -0.88640526 -0.27630176 2.26879087]

[1.67144992 -0.88640526 0.44136514 2.49780745]

[1.68877065 1.12815215 -0.49160182 2.49780745]

[1.70609137 1.12815215 -0.49160182 2.91767117]

[1.7234121 1.12815215 -0.6351352 2.91767117]]

```
X_scaled=pd.DataFrame(scale,columns=X.columns)
```

```
X_scaled
```

	CustomerID	Gender	Age	Annual Income (k$)
0	-1.723412	1.128152	-1.424569	-1.738999
1	-1.706091	1.128152	-1.281035	-1.738999
2	-1.688771	-0.886405	-1.352802	-1.700830
3	-1.671450	-0.886405	-1.137502	-1.700830
4	-1.654129	-0.886405	-0.563369	-1.662660
...	...	...	...	...
195	1.654129	-0.886405	-0.276302	2.268791
196	1.671450	-0.886405	0.441365	2.497807
197	1.688771	1.128152	-0.491602	2.497807
198	1.706091	1.128152	-0.491602	2.917671
199	1.723412	1.128152	-0.635135	2.917671

200 rows × 4 columns

KMeans Clustering Algorithm

Train and Test Split

```
from sklearn.model_selection import train_test_split
# split the dataset
X_train, X_test, Y_train, Y_test = train_test_split(X_scaled, Y, test_size=0.20, random_state=0)
```

```
X_train.shape
```

(160, 4)

```
X_test.shape
```

(40, 4)

```
Y_train.shape

(160,)

Y_test.shape

(40,)

Build the Model

x = df.iloc[:,[3,4]].values

#training the K-means model on a dataset

kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)

y_predict= kmeans.fit_predict(x)

#visulaizing the clusters

plt.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #for first cluster

plt.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #for second cluster

plt.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #for third cluster

plt.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') #for fourth cluster

plt.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5') #for fifth cluster

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroid')

plt.title('Clusters of customers')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()
```