Assignment -4

Applied Data Science

Assignment Date	8 October 2022
Student Name	Gokulakrishnan G
Student Roll Number	2116190701053
Maximum Marks	2 Marks

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew
```

```
data = pd.read_csv("C:\IBM\Mall_Customers.csv")
data.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

```
data.tail()
```

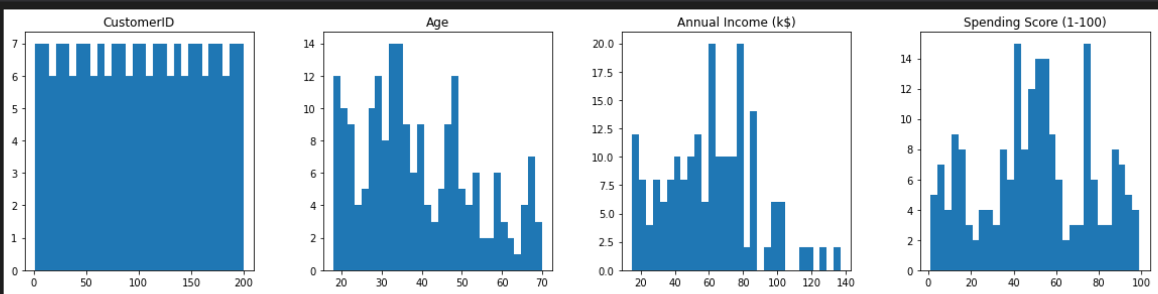
	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

```
data.describe()
```

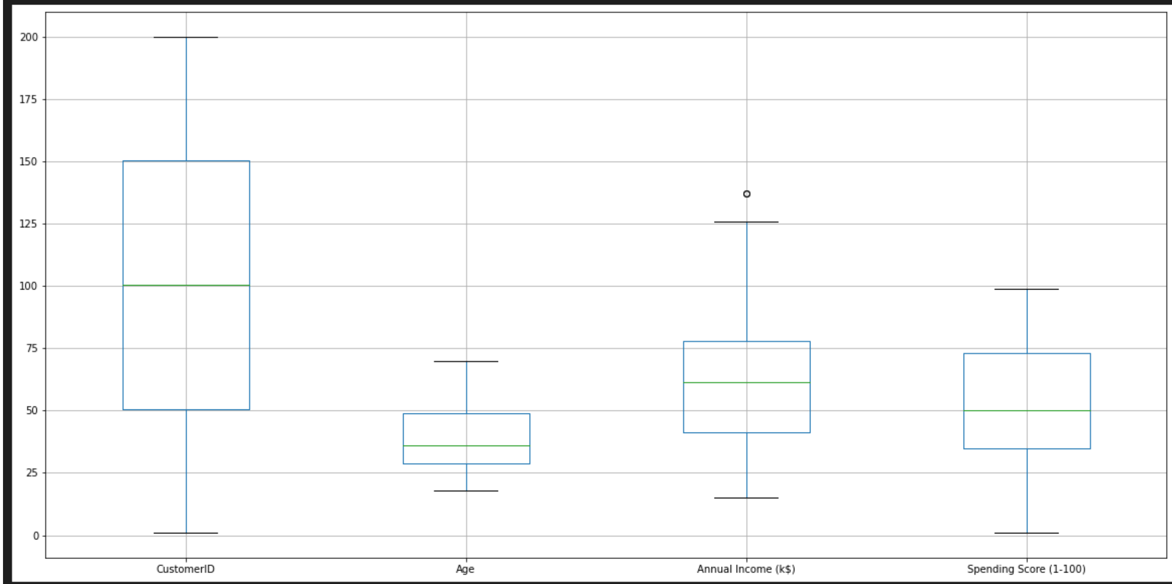
	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522

univariate analysis

```
data.hist(figsize=(20,10), grid=False, layout=(2,4), bins=30)
plt.show()
```

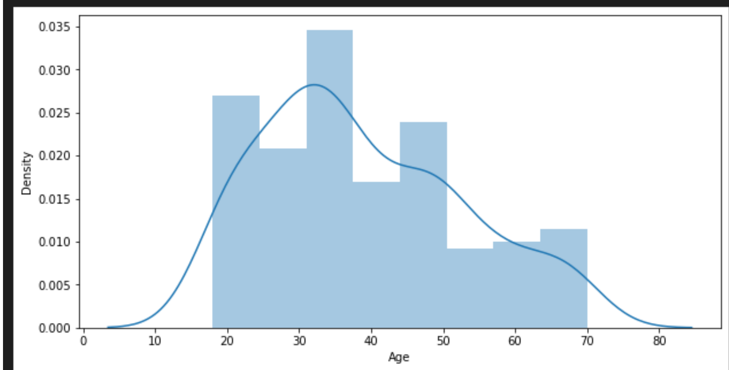


```python
data.boxplot(figsize=(20,10))
plt.show()
```




```python
plt.figure(figsize=(10,5))
sns.distplot(data['Age'])
plt.show()
```

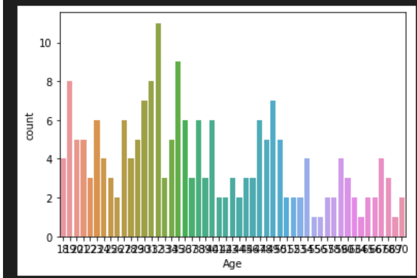
```
C:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



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

```
C:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only v
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

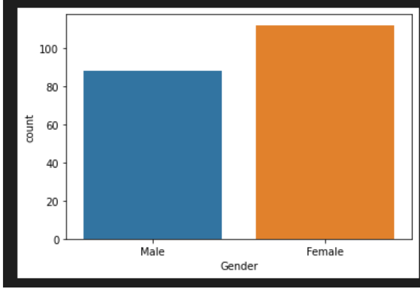
```
<AxesSubplot:xlabel='Age', ylabel='count'>
```




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

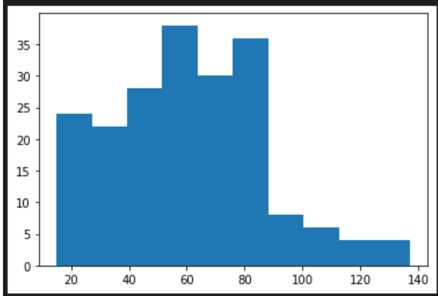
```
C:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only v
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



```
plt.hist(data['Annual Income (k$)'])
```

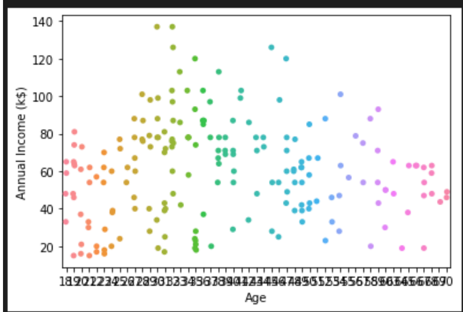
```
(array([24., 22., 28., 38., 30., 36.,  8.,  6.,  4.,  4.]),
 array([ 15. ,  27.2,  39.4,  51.6,  63.8,  76. ,  88.2, 100.4, 112.6,
        124.8, 137. ]),
 <BarContainer object of 10 artists>)
```



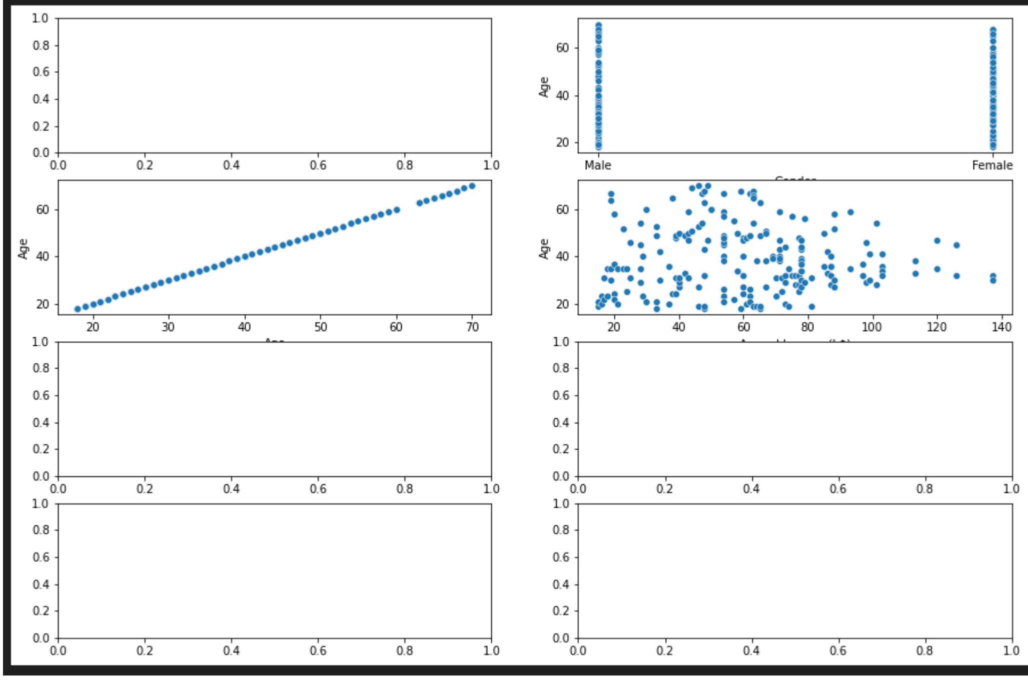
Bivariate Analysis

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

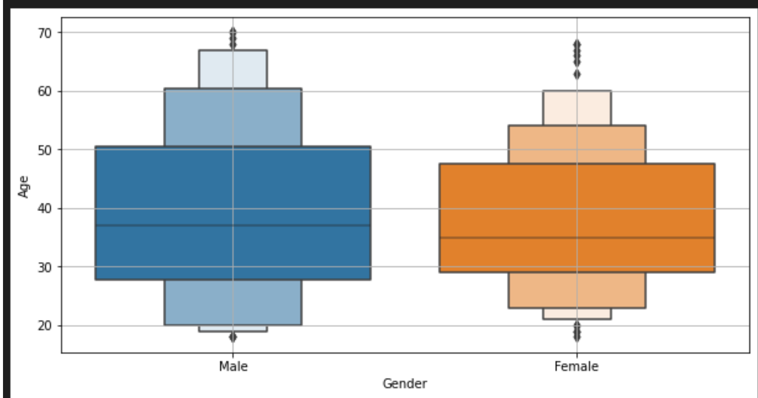
```
<AxesSubplot:xlabel='Age', ylabel='Annual Income (k$)'>
```



```python
fig, axes = plt.subplots(4,2, figsize=(15,10))
axes = axes.flatten()
for i in range(1,len(data.columns)-1):
    sns.scatterplot(x=data.iloc[:,i], y=data['Age'], ax=axes[i])
plt.show()
```



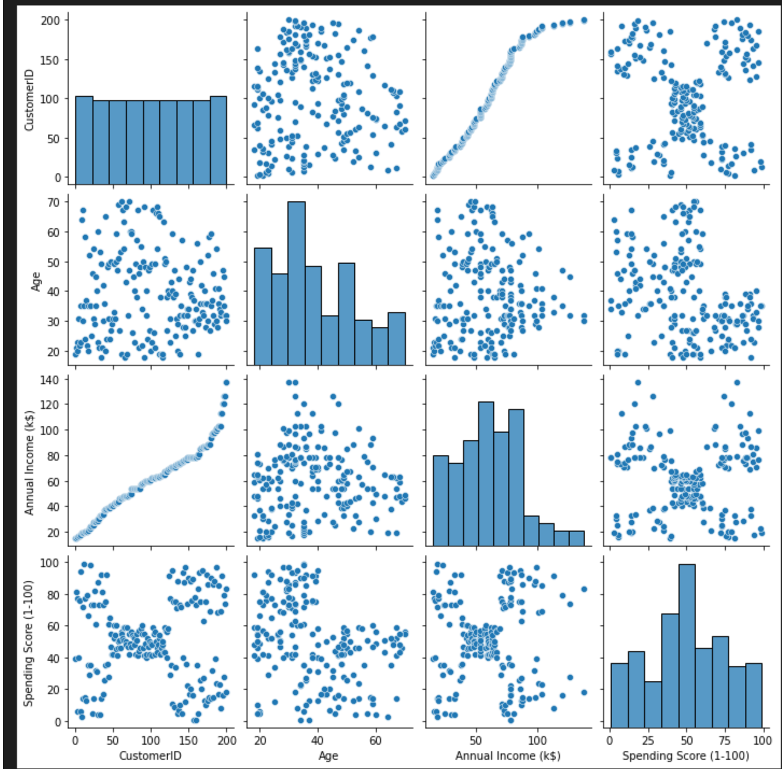
```python
plt.figure(figsize=(10,5))
sns.boxenplot(y=data['Age'], x=data['Gender'])
plt.grid()
plt.show()
data.groupby('Gender')['Age'].describe()
```



	count	mean	std	min	25%	50%	75%	max
Gender								
Female	112.0	38.098214	12.644095	18.0	29.00	35.0	47.5	68.0
Male	88.0	39.806818	15.514812	18.0	27.75	37.0	50.5	70.0

Multivariate Analysis

```
sns.pairplot(data)
plt.show()
```



```python
plt.figure(figsize=(10,5))
sns.heatmap(data.corr(), annot=True)
plt.show()
```

Descriptive Statistics

```python
#mean

data.mean()
```

```
C:\Users\michael\AppData\Local\Temp\ipykernel_7288\4148990336.py:3: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numer
only valid columns before calling the reduction.
  data.mean()

CustomerID                100.50
Age                        38.85
Annual Income (k$)         60.56
Spending Score (1-100)     50.20
dtype: float64
```

```
#mode

data.mode()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Female	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

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

```
Female    112
Male       88
Name: Gender, dtype: int64
```

```
data.shape
```

```
(200, 5)
```

```
data.isnull().sum()
```

```
CustomerID                0
Gender                    0
Age                       0
```

Missing values and deal with them

```
data.isna()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
195	False	False	False	False	False
196	False	False	False	False	False
197	False	False	False	False	False
198	False	False	False	False	False
199	False	False	False	False	False

200 rows × 5 columns

```
data.isna().any()
```

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

```python
data.kurt()
```

```
C:\Users\michael\AppData\Local\Temp\ipykernel_7288\2907027414.py:1: FutureWarning: Dropping of nuisance columns in DataFr
only valid columns before calling the reduction.
  data.kurt()

CustomerID               -1.200000
Age                      -0.671573
Annual Income (k$)       -0.098487
Spending Score (1-100)   -0.826629
dtype: float64
```

```python
data.var()
```

```
C:\Users\michael\AppData\Local\Temp\ipykernel_7288\445316826.py:1: FutureWarning: Dropping of nuisance columns in DataFra
only valid columns before calling the reduction.
  data.var()

CustomerID                3350.000000
Age                        195.133166
Annual Income (k$)         689.835578
Spending Score (1-100)     666.854271
dtype: float64
```

```python
data.std()
```

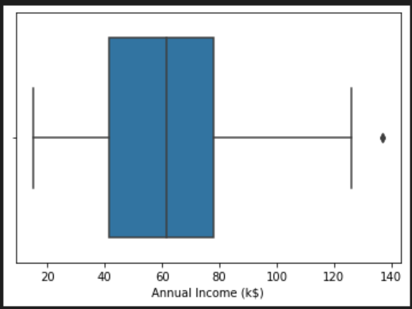
```
C:\Users\michael\AppData\Local\Temp\ipykernel_7288\2723740006.py:1: FutureWarning: Dropping of nuisance columns in DataFr
only valid columns before calling the reduction.
  data.std()

CustomerID                57.879185
Age                       13.969007
Annual Income (k$)        26.264721
Spending Score (1-100)    25.823522
dtype: float64
```

Find the outliers and replace them outliers

```
sns.boxplot(data['Annual Income (k$)'])
```

```
C:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
<AxesSubplot:xlabel='Annual Income (k$)'>
```




```
qnt=data.quantile(q=(0.30,0.45))

qnt
```

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
0.30	60.70	30.0	46.00	40.00
0.45	90.55	35.0	58.55	47.55

```
iqr =qnt.loc[0.45]-qnt.loc[0.30] #iqr calculation
iqr
```

```
CustomerID                29.85
Age                        5.00
Annual Income (k$)        12.55
Spending Score (1-100)     7.55
dtype: float64
```

```
#lower extreme values
lower=qnt.loc[0.30]-1.5*iqr
lower
```

```
CustomerID                15.925
Age                       22.500
Annual Income (k$)        27.175
Spending Score (1-100)    28.675
dtype: float64
```

```
#upper extreme values
upper=qnt.loc[0.45]+1.5*iqr
upper
```

```
CustomerID                135.325
Age                        42.500
Annual Income (k$)         77.375
Spending Score (1-100)     58.875
dtype: float64
```

```
data['CustomerID']=np.where(data['CustomerID']>45,31,data['CustomerID'])
```

Encoding Categorical Values

```
numeric_data = data.select_dtypes(include=[np.number])
categorical_data = data.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']
```

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

```
Female    112
Male       88
Name: Gender, dtype: int64
```

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

```
0    112
1     88
Name: Gender, dtype: int64
```

Scaling the data

```
X = data.drop("Age",axis=1)
Y = data['Age']
```

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

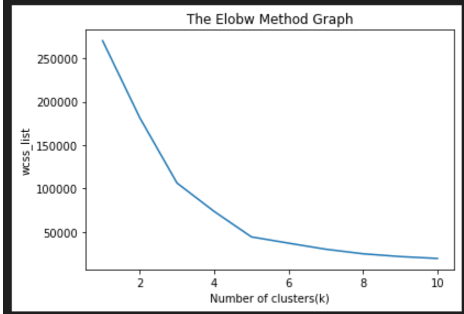
```
Output exceeds the size limit. Open the full output data in a text editor
[[-4.02405716  1.12815215 -1.73899919 -0.43480148]
 [-3.8813601   1.12815215 -1.73899919  1.19570407]
 [-3.73866304 -0.88640526 -1.70082976 -1.71591298]
 [-3.59596597 -0.88640526 -1.70082976  1.04041783]
 [-3.45326891 -0.88640526 -1.66266033 -0.39597992]
 [-3.31057185 -0.88640526 -1.66266033  1.00159627]
 [-3.16787479 -0.88640526 -1.62449091 -1.71591298]
 [-3.02517772 -0.88640526 -1.62449091  1.70038436]
 [-2.88248066  1.12815215 -1.58632148 -1.83237767]
 [-2.7397836  -0.88640526 -1.58632148  0.84631002]
 [-2.59708654  1.12815215 -1.58632148 -1.4053405 ]
 [-2.45438947 -0.88640526 -1.58632148  1.89449216]
 [-2.31169241 -0.88640526 -1.54815205 -1.36651894]
 [-2.16899535 -0.88640526 -1.54815205  1.04041783]
 [-2.02629829  1.12815215 -1.54815205 -1.44416206]
 [-1.88360122  1.12815215 -1.54815205  1.11806095]
 [-1.74090416 -0.88640526 -1.50998262 -0.59008772]
```

Clustering Algorithm

```
x = data.iloc[:, [3, 4]].values
```

```
from sklearn.cluster import KMeans
wcss_list= []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss_list)
plt.title('The Elobw Method Graph')
plt.xlabel('Number of clusters(k)')
plt.ylabel('wcss_list')
plt.show()
```

```
C:\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when t
variable OMP_NUM_THREADS=1.
  warnings.warn(
```




```
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()
```

Split the data into dependent and independent variables.

```
#target variable

y=data['Age']
y.head()
```

```
0    19
1    21
2    20
3    23
4    31
```

```python
data=pd.get_dummies(data,columns=['Age'])

data.head()
```

	CustomerID	Gender	Annual Income (k$)	Spending Score (1-100)	Age_18	Age_19	Age_20	Age_21	Age_22	Age_23	...	Age_59	Age_60	Age_63	Age_64	Age_65	Age_66	Age_67	Age_68	Age_69	Age_70
0	1	1	15	39	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	2	1	15	81	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
2	3	0	16	6	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	4	0	16	77	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
4	5	0	17	40	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 55 columns

```python
#encoding
data = pd.get_dummies(data, drop_first=True)
data.head()
```

	CustomerID	Gender	Annual Income (k$)	Spending Score (1-100)	Age_18	Age_19	Age_20	Age_21	Age_22	Age_23	...	Age_59	Age_60	Age_63	Age_64	Age_65	Age_66	Age_67	Age_68	Age_69	Age_70
0	1	1	15	39	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	2	1	15	81	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
2	3	0	16	6	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	4	0	16	77	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
4	5	0	17	40	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 55 columns

Split the data into training and testing

```python
from sklearn.model_selection import train_test_split
```

```python
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

```python
x_train.shape
```

```
(160, 4)
```

```python
x_test.shape
```

```
(40, 4)
```

```python
y_train.shape
```

```
(160,)
```

```python
y_test.shape
```

```
(40,)
```

Build, test, train the Model

```python
from sklearn.tree import DecisionTreeClassifier
```

```python
#initializing the DT

model=DecisionTreeClassifier()
```

```python
#encoding

data = pd.get_dummies(data, drop_first=True)
data.head()
```

	CustomerID	Gender	Annual Income (k$)	Spending Score (1-100)	Age_18	Age_19	Age_20	Age_21	Age_22	Age_23	...	Age_59	Age_60	Age_63	Age_64	Age_65	Age_66	Age_67	Age_68	Age_69	Age_70
0	1	1	15	39	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	2	1	15	81	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
2	3	0	16	6	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	4	0	16	77	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
4	5	0	17	40	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 55 columns

```python
X = data.drop('Gender', axis=1)
y = data['Gender']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)

from sklearn.preprocessing import StandardScaler
ss = StandardScaler()

X_trains = ss.fit_transform(X_train)
X_tests = ss.transform(X_test)
```

```python
#Base model
from sklearn.linear_model import LinearRegression
lr = LinearRegression()

lr.fit(X_trains, y_train)
pred = lr.predict(X_tests)

from sklearn.metrics import r2_score, roc_auc_score, mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, pred))
r2 = r2_score(y_test, pred)

print("The root mean Sq error calculated from the base model is:",rmse)
print("The r2-score is:",r2)
```

```
The root mean Sq error calculated from the base model is: 849861698448.1481
The r2-score is: -2.932139732215125e+24
```

```python
#selecting best feautre

from sklearn.feature_selection import RFE
lr = LinearRegression()
n = [{'n_features_to_select':list(range(1,10))}]
rfe = RFE(lr)

from sklearn.model_selection import GridSearchCV
gsearch = GridSearchCV(rfe, param_grid=n, cv=3)
gsearch.fit(X, y)

gsearch.best_params_
```

```
{'n_features_to_select': 1}
```

```python
lr = LinearRegression()
rfe = RFE(lr, n_features_to_select=8)
rfe.fit(X,y)

pd.DataFrame(rfe.ranking_, index=X.columns, columns=['Gender'])
```

	Gender
CustomerID	45
Annual Income (k$)	46

Measure the performance using Evaluation Metrics

```python
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import  RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import  GradientBoostingRegressor
from sklearn.linear_model import  Ridge
from sklearn.svm import SVR
from sklearn import model_selection
from sklearn.model_selection import cross_val_predict

models = [   SVR(),
             RandomForestRegressor(),
             GradientBoostingRegressor(),
             KNeighborsRegressor(n_neighbors = 4)]
results = []
names = ['SVM','Random Forest','Gradient Boost','K-Nearest Neighbors']
for model,name in zip(models,names):
    kfold = model_selection.KFold(n_splits=10)
    cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
    rmse = np.sqrt(mean_squared_error(y, cross_val_predict(model, X , y, cv=3)))
    results.append(rmse)
    names.append(name)
    msg = "%s: %f" % (name, rmse)
    print(msg)
```

```
SVM: 0.542347
Random Forest: 0.511006
Gradient Boost: 0.542861
K-Nearest Neighbors: 0.523808
```