***Customer Segmentation***

In [1]:
```
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.cluster import KMeans

import joblib

from tkinter import *
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
dataset = pd.read_csv('Mall_Customers.csv')
```

In [4]:
```
dataset.head()
```

Out[4]:

| | CustomerID | Genre | 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 |

In [5]:
```
dataset.tail()
```

Out[5]:

| | CustomerID | Genre | 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 |

In [6]:
```
dataset.shape
```

Out[6]:
```
(200, 5)
```

In [7]:
```
print('Number of Rows:',dataset.shape[0])
print('Number of Columns:',dataset.shape[1])
```

```
Number of Rows: 200
Number of Columns: 5
```

In [8]:
```
dataset.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   Genre                   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
```

In [9]:
```
dataset.isnull().sum()
```

Out[9]:
```
CustomerID                0
Genre                     0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [10]:
```
dataset.duplicated().sum()
```

Out[10]:
```
0
```

In [11]:
```
dataset.describe()
```

Out[11]:

| | 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 |

In [12]:
```
dataset.corr()
```

Out[12]:

| | CustomerID | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| **CustomerID** | 1.000000 | -0.026763 | 0.977548 | 0.013835 |
| **Age** | -0.026763 | 1.000000 | -0.012398 | -0.327227 |
| **Annual Income (k$)** | 0.977548 | -0.012398 | 1.000000 | 0.009903 |
| **Spending Score (1-100)** | 0.013835 | -0.327227 | 0.009903 | 1.000000 |

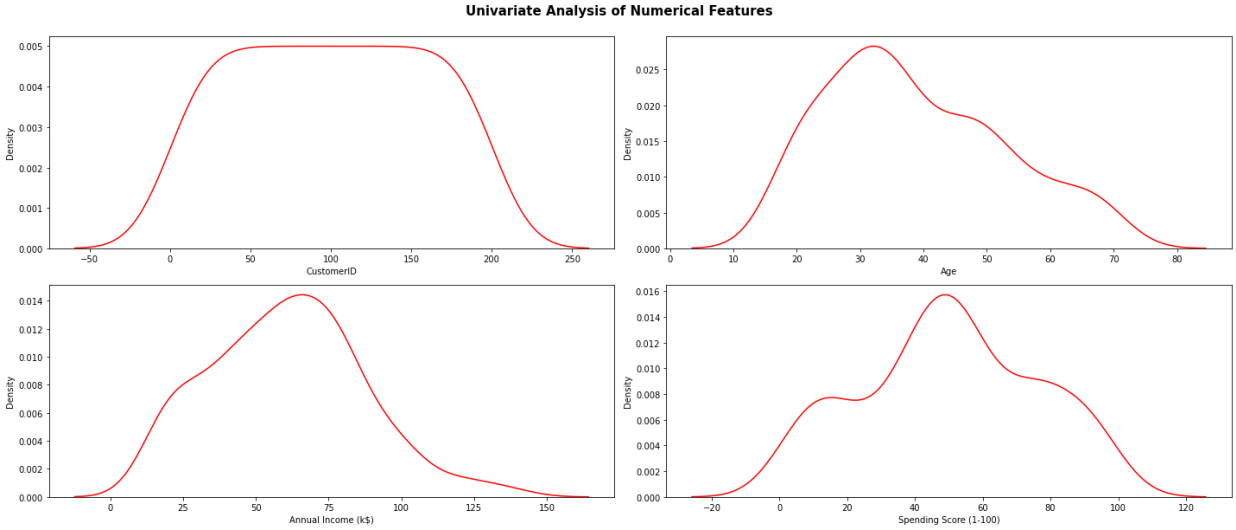
In [13]:
```
dataset.cov()
```

Out[13]:

| | CustomerID | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| **CustomerID** | 3350.000000 | -21.638191 | 1486.050251 | 20.678392 |
| **Age** | -21.638191 | 195.133166 | -4.548744 | -118.040201 |
| **Annual Income (k$)** | 1486.050251 | -4.548744 | 689.835578 | 6.716583 |
| **Spending Score (1-100)** | 20.678392 | -118.040201 | 6.716583 | 666.854271 |

In [14]:
```
numerical_cols = [features for features in dataset.columns if dataset[features]
```

In [15]:
```
plt.figure(figsize=(20,20))
plt.suptitle('Univariate Analysis of Numerical Features',fontweight='bold',font

for i in range(0,len(numerical_cols)):
    plt.subplot(5,2,i+1)
    sns.kdeplot(data=dataset,x=numerical_cols[i],color='red')
    plt.xlabel(numerical_cols[i])
    plt.tight_layout()
```




In [16]:
```
X = dataset[['Annual Income (k$)','Spending Score (1-100)']]
```

In [17]:
```
k_means = KMeans(n_clusters=5)
k_means.fit(X)
```

Out[17]:
```
KMeans(n_clusters=5)
```

In [18]:
```
k_means.fit_predict(X)
```

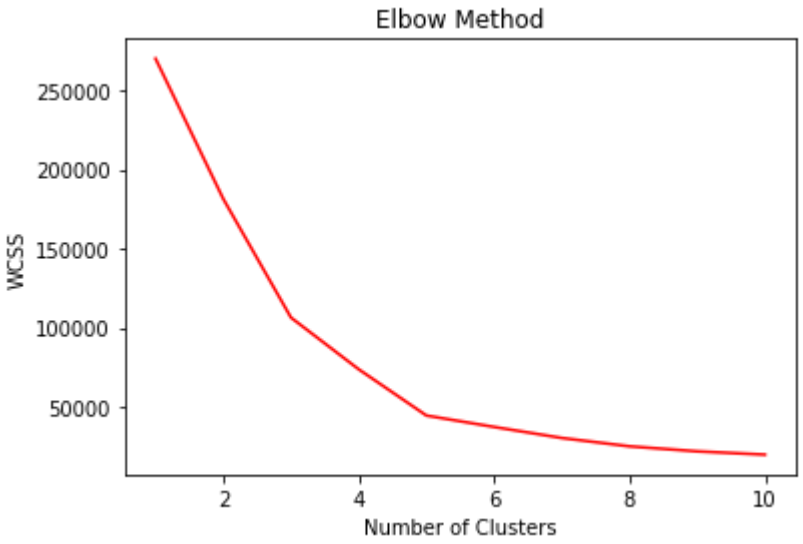
Out[18]:
```
array([4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0,
       4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 4, 1,
       4, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 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, 2, 3, 2, 1, 2, 3, 2, 3, 2,
       1, 2, 3, 2, 3, 2, 3, 2, 3, 2, 1, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2,
       3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2,
       3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2,
       3, 2])
```

In [19]:
```
wcss = []

for i in range(1,11):
    k_means = KMeans(n_clusters=i)
    k_means.fit(X)
    wcss.append(k_means.inertia_)
```

In [20]:
```
wcss
```

Out[20]:
```
[269981.28,
 181363.59595959596,
 106348.37306211119,
 73679.78903948834,
 44448.45544793371,
 37233.81451071001,
 30259.65720728547,
 24986.52564064288,
 21818.114588452183,
 19740.010370359305]
```

In [21]:
```
plt.plot(range(1,11),wcss,color='red')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

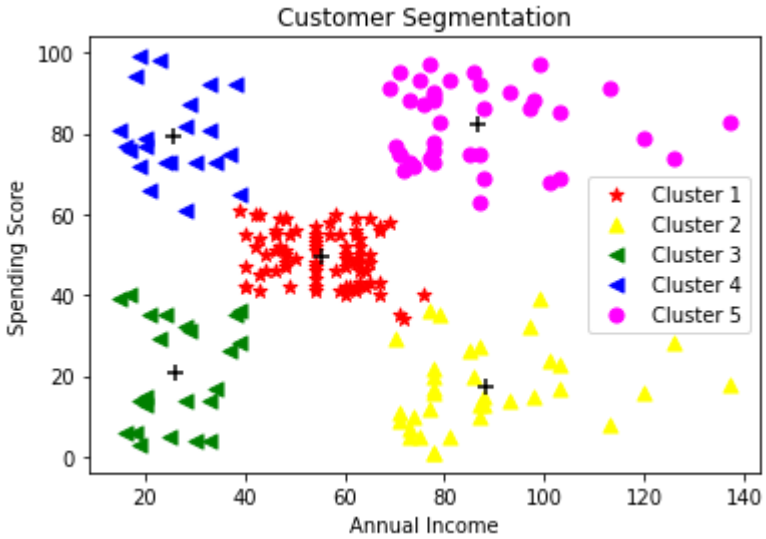



In [22]:
```
k_means = KMeans(n_clusters=5,random_state=42)
y_means = k_means.fit_predict(X)
```

In [23]:
```
y_means
```

Out[23]:
```
array([2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3,
       2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 0,
       2, 3, 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, 0, 0, 4, 1, 4, 0, 4, 1, 4, 1, 4,
       0, 4, 1, 4, 1, 4, 1, 4, 1, 4, 0, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4,
       1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4,
       1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4,
       1, 4])
```

In [24]:
```
plt.scatter(X.iloc[y_means==0,0],X.iloc[y_means==0,1],s=50,color='red',label='(
plt.scatter(X.iloc[y_means==1,0],X.iloc[y_means==1,1],s=50,color='yellow',label
plt.scatter(X.iloc[y_means==2,0],X.iloc[y_means==2,1],s=50,color='green',label=
plt.scatter(X.iloc[y_means==3,0],X.iloc[y_means==3,1],s=50,color='blue',label=
plt.scatter(X.iloc[y_means==4,0],X.iloc[y_means==4,1],s=50,color='magenta',labe
plt.scatter(k_means.cluster_centers_[:,0],k_means.cluster_centers_[:,1],s=50,c
plt.title('Customer Segmentation')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.legend()
plt.show()
```




In [25]:
```
k_means.predict([[15,39]])
```

Out[25]:
```
array([2])
```

In [26]:
```
joblib.dump(k_means,'customer_segmentation')
```

Out[26]:
```
['customer_segmentation']
```

In [27]:
```
model = joblib.load('customer_segmentation')
```

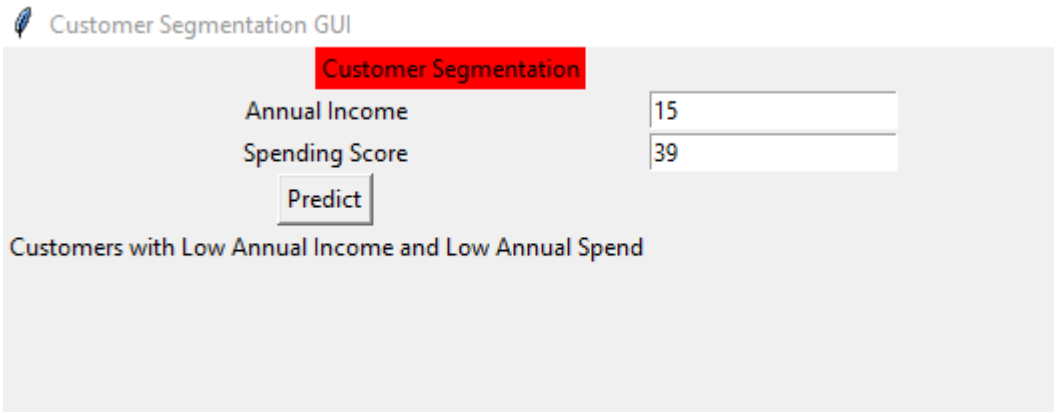
In [28]:
```
model.predict([[15,39]])
```

Out[28]:
```
array([2])
```

*GUI*

In [29]:
```
def show_entry_fields():
    p1=int(e1.get())
    p2=int(e2.get())

    model = joblib.load('customer_segmentation')
    result=model.predict([[p1,p2]])
    print('This Customer Belongs to Cluster No:', result[0])

    if result[0] == 0:
        Label(master, text='Customers with Medium Annual Income and Medium Annu
    elif result[0]==1:
        Label(master, text='Customers with High Annual Income but Low Annual Sp
    elif result[0]==2:
        Label(master, text='Customers with Low Annual Income and Low Annual Spe
    elif result[0]==3:
        Label(master, text='Customers Low Annual Income but High Annual Spend'
    elif result[0]==4:
        Label(master, text='Customers with High Annual Income and High Annual S

master = Tk()
master.title('Customer Segmentation GUI')


label = Label(master, text = 'Customer Segmentation'
                          , bg='red',fg='black'). \
                            grid(row=0,columnspan=2)

Label(master,text='Annual Income').grid(row=1)
Label(master, text='Spending Score').grid(row=2)


e1 = Entry(master)
e2 = Entry(master)

e1.grid(row=1, column=1)
e2.grid(row=2, column=1)

Button(master, text='Predict', command=show_entry_fields).grid()

mainloop()
```

```
This Customer Belongs to Cluster No: 2
```




***Thank You***

`github.com/prasadmjadhav2`