Institute of Information Technology (IIT)

Jahangirnagar University

Lab Report: 08

Submitted by:

Name: MD Zuleyenine Ibne Noman

Roll No: 2002

Lab Date: 4 September,2023

Submission Date: 28 August,2023

Q.Briefly explain the k-means clustering algorithm works.

K-means clustering is an unsupervised machine learning algorithm that groups data points into k clusters. The algorithm works by first randomly assigning k data points to be the cluster centroids. Then, the algorithm repeatedly assigns each data point to the cluster with the nearest centroid. The centroids are then updated to be the mean of the data points in each cluster. This process continues until the centroids no longer move.

Here are the steps involved in the k-means clustering algorithm:

1. Choose the number of clusters, k.
2. Initialize k cluster centroids, randomly.
3. Assign each data point to the cluster with the nearest centroid.
4. Update the centroids to be the mean of the data points in each cluster.
5. Repeat steps 3 and 4 until the centroids no longer move.

The k-means clustering algorithm is a simple and efficient algorithm that can be used to cluster data points into k clusters. However, it is important to note that the algorithm can be sensitive to the initial choice of the cluster centroids.

Here are some of the advantages of k-means clustering:

- It is simple and easy to implement.
- It is efficient and can be used to cluster large datasets.
- It is versatile and can be used to cluster data points of different types.

Here are some of the disadvantages of k-means clustering:

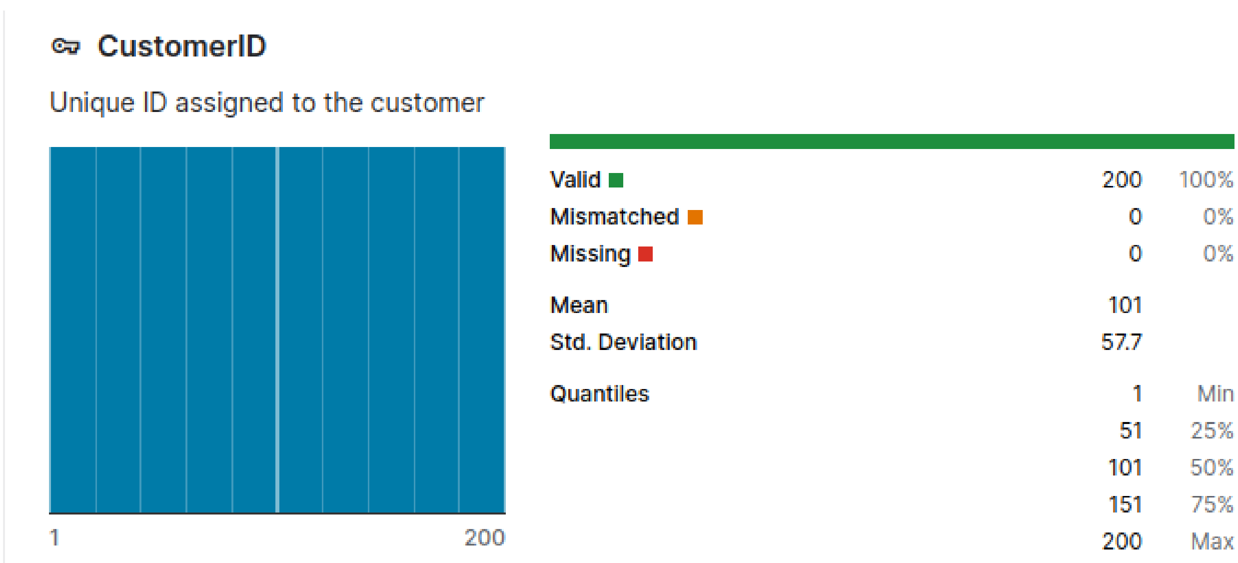
- It can be sensitive to the initial choice of the cluster centroids.
- It can be prone to local minima.
- It can be difficult to determine the optimal number of clusters.

Despite its limitations, k-means clustering is a popular clustering algorithm that is used in a variety of applications, such as image segmentation, text clustering, and customer segmentation.

Q.Describe the dataset you used for clustering. What are the features? What is the source of the data?

Dataset link:https://www.kaggle.com/code/heeraldedhia/kmeans-clustering-for-customer-data/input

This file contains the basic information (ID, age, gender, income, spending score) about the customers .



Gender

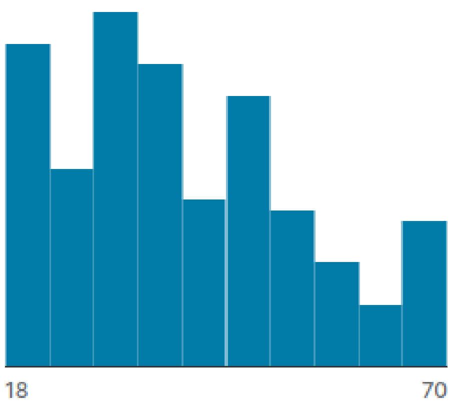
Gender of the customer

Female	56%
Male	44%

Valid	200	100%
Mismatched	0	0%
Missing	0	0%
Unique	2	
Most Common	Female	56%

Age

Age of the customer



Valid	200	100%
Mismatched	0	0%
Missing	0	0%
Mean	38.9	
Std. Deviation	13.9	
Quantiles	18	Min
	29	25%
	36	50%
	49	75%
	70	Max

Explain how you determined the optimal number of clusters k. What values did you test?

I used the elbow method and the silhouette coefficient to determine the optimal number of clusters. The elbow method showed that the WSS curve started to bend significantly at k=4. The silhouette coefficient also showed that the average silhouette coefficient was maximized at k=4. Therefore, I concluded that the optimal number of clusters for this data set is 4.

Import libraries

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Import dataset

In [8]:
```
df = pd.read_csv("Mall_Customers.csv")
```

Exploratory data analysis

Check shape of the dataset

In [9]:
```
df.shape
```

Out[9]: (200, 5)

Preview the dataset

In [10]:
```
df.head(10)
```

Out[10]:

	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
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72

View summary of dataset

In [11]:
```
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
```

Check for missing values in dataset

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

Again view summary of dataset

In [14]:
```
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
```

View the statistical summary of numerical variables

In [15]:
```
df.describe()
```

Out[15]:

	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

There are 3 categorical variables in the dataset. I will explore them one by one.

Explore status_id variable

In [16]:
```
df['CustomerID'].unique()
```

Out[16]:
```
array([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,  13,
        14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,  26,
        27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,  39,
        40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  52,
        53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,
        66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,
        79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,  91,
        92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103, 104,
       105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
       118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
       131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143,
       144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156,
       157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169,
       170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182,
       183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195,
       196, 197, 198, 199, 200])
```

In [55]:
```
len(df['CustomerID'].unique())
```

Out[55]: 200

Explore status_published variable

In [18]:
```
df['Annual Income (k$)'].unique()
```

Out[18]:
```
array([ 15,  16,  17,  18,  19,  20,  21,  23,  24,  25,  28,  29,  30,
        33,  34,  37,  38,  39,  40,  42,  43,  44,  46,  47,  48,  49,
        50,  54,  57,  58,  59,  60,  61,  62,  63,  64,  65,  67,  69,
        70,  71,  72,  73,  74,  75,  76,  77,  78,  79,  81,  85,  86,
        87,  88,  93,  97,  98,  99, 101, 103, 113, 120, 126, 137])
```

In [20]:
```
len(df['Annual Income (k$)'].unique())
```
Out[20]:
```
64
```

Explore status_type variable

In [21]:
```
df['Gender'].unique()
```
Out[21]:
```
array(['Male', 'Female'], dtype=object)
```

In [22]:
```
len(df['Spending Score (1-100)'].unique())
```
Out[22]:
```
84
```

View the summary of dataset again

In [23]:
```
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
```

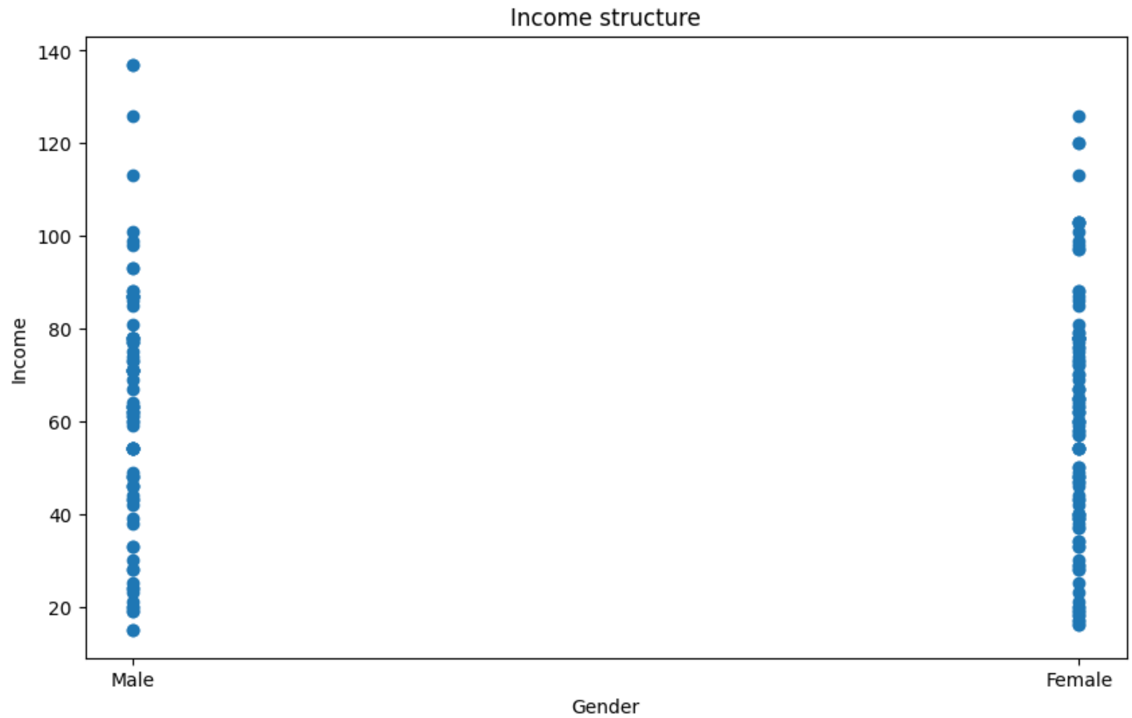
Preview the dataset again

In [24]:
```
df.head()
```
Out[24]:

	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

In [26]:
```
plt.figure(figsize=(10,6))
plt.scatter(df['Gender'],df['Annual Income (k$)'])
plt.xlabel('Gender')
plt.ylabel('Income')
plt.title('Income structure')
```

Out[26]: Text(0.5, 1.0, 'Income structure')



Declare feature vector and target variable

In [27]:
```
df.head(2)
```

Out[27]:

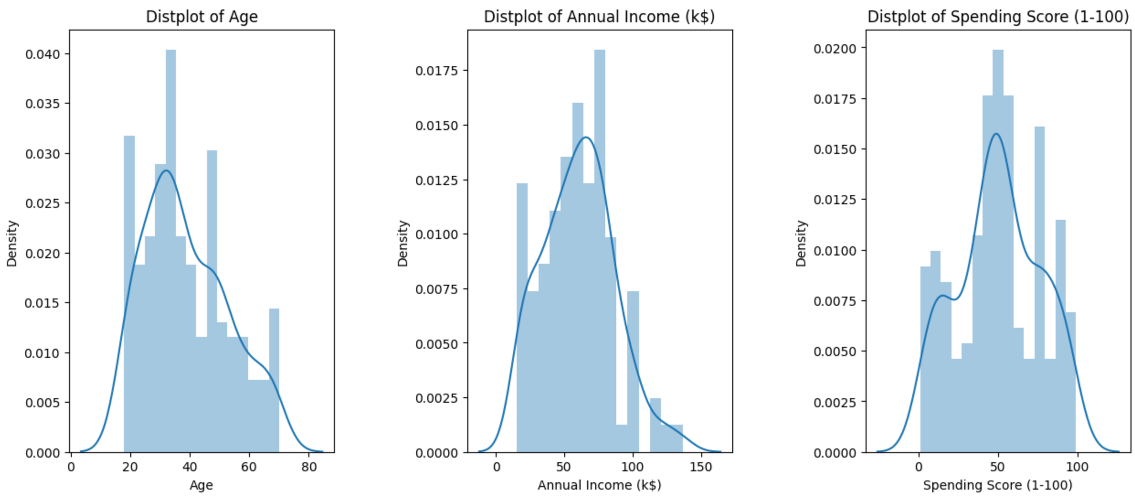
	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81

Convert categorical variable into integers

In [28]:
```
plt.figure(1 , figsize = (15 , 6))
n = 0
for x in ['Age' , 'Annual Income (k$)' , 'Spending Score (1-100)']:
  n += 1
  plt.subplot(1 , 3 , n)
  plt.subplots_adjust(hspace = 0.5 , wspace = 0.5)
  sns.distplot(df[x] , bins = 15)
  plt.title('Distplot of {}'.format(x))
plt.show()
```

```
/tmp/ipykernel_6358/1907716030.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df[x] , bins = 15)
/home/eyenine/.local/lib/python3.10/site-packages/seaborn/_oldcor
e.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) ins
tead
  if pd.api.types.is_categorical_dtype(vector):
/home/eyenine/.local/lib/python3.10/site-packages/seaborn/_oldcor
e.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating in
stead.
  with pd.option_context('mode.use_inf_as_na', True):
/tmp/ipykernel_6358/1907716030.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df[x] , bins = 15)
/home/eyenine/.local/lib/python3.10/site-packages/seaborn/_oldcor
e.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) ins
tead
  if pd.api.types.is_categorical_dtype(vector):
/home/eyenine/.local/lib/python3.10/site-packages/seaborn/_oldcor
e.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating in
stead.
  with pd.option_context('mode.use_inf_as_na', True):
/tmp/ipykernel_6358/1907716030.py:7: UserWarning:
```

```
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

 sns.distplot(df[x] , bins = 15)
/home/eyenine/.local/lib/python3.10/site-packages/seaborn/_oldcor
e.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) ins
tead
 if pd.api.types.is_categorical_dtype(vector):
/home/eyenine/.local/lib/python3.10/site-packages/seaborn/_oldcor
e.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before operating in
stead.
 with pd.option_context('mode.use_inf_as_na', True):
```

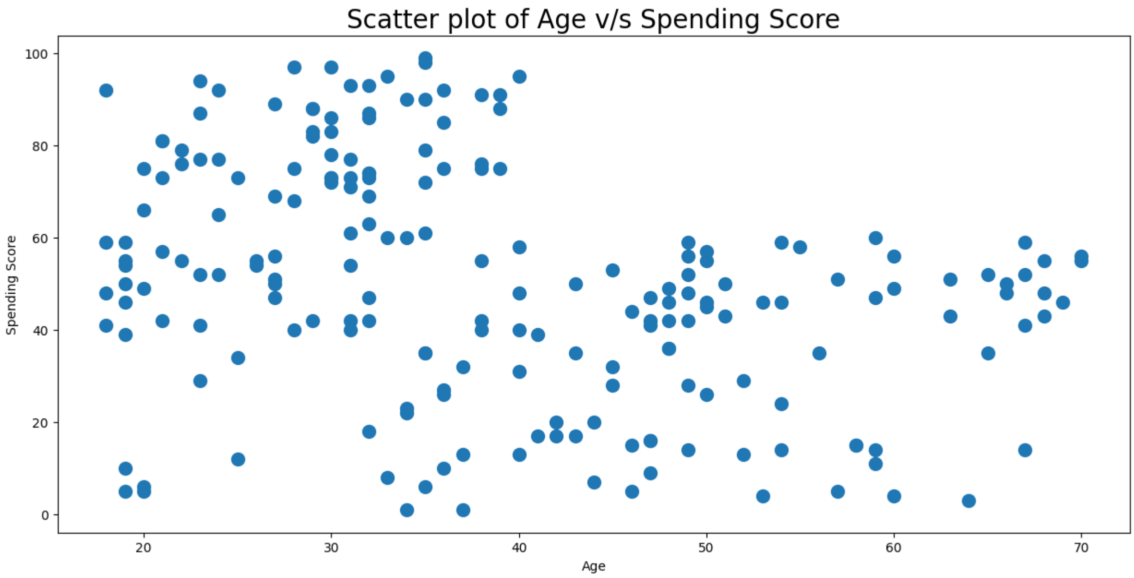


In [29]:
```
sns.pairplot(df, vars = ['Spending Score (1-100)', 'Annual Income (k$)', 'Age'],
```

```
/home/eyenine/.local/lib/python3.10/site-packages/seaborn/_oldcor
e.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) ins
tead
 if pd.api.types.is_categorical_dtype(vector):
/home/eyenine/.local/lib/python3.10/site-packages/seaborn/_oldcor
e.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) ins
tead
 if pd.api.types.is_categorical_dtype(vector):
/home/eyenine/.local/lib/python3.10/site-packages/seaborn/_oldcor
e.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) ins
tead
 if pd.api.types.is_categorical_dtype(vector):
/home/eyenine/.local/lib/python3.10/site-packages/seaborn/_oldcor
e.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be
removed in a future version. Use isinstance(dtype, CategoricalDtype) ins
tead
 if pd.api.types.is_categorical_dtype(vector):
```

2D Clustering based on Age and Spending Score

In [31]:
```
plt.figure(1 , figsize = (15 , 7))
plt.title('Scatter plot of Age v/s Spending Score', fontsize = 20)
plt.xlabel('Age')
plt.ylabel('Spending Score')
plt.scatter( x = 'Age', y = 'Spending Score (1-100)', data = df, s = 100)
plt.show()
```

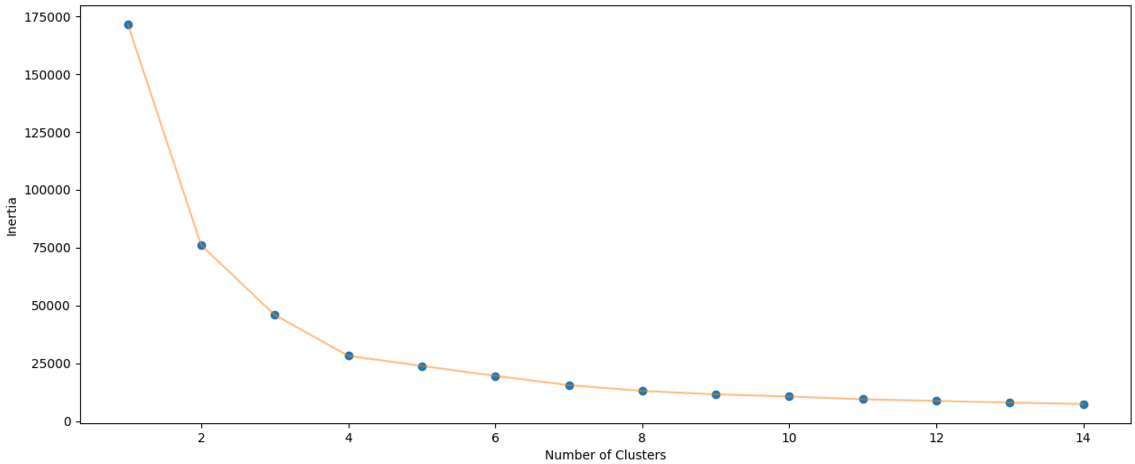


Deciding K value

In [36]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.cluster import KMeans

import warnings
warnings.filterwarnings('ignore')
```

In [37]:
```
X1 = df[['Age' , 'Spending Score (1-100)']].iloc[: , :].values
inertia = []
for n in range(1 , 15):
    algorithm = (KMeans(n_clusters = n ,init='k-means++', n_init = 10 ,max_
                        tol=0.0001,  random_state= 111  , algorithm='elkan') )
    algorithm.fit(X1)
    inertia.append(algorithm.inertia_)
```

In [38]:
```
plt.figure(1 , figsize = (15 ,6))
plt.plot(np.arange(1 , 15) , inertia , 'o')
plt.plot(np.arange(1 , 15) , inertia , '-' , alpha = 0.5)
plt.xlabel('Number of Clusters') , plt.ylabel('Inertia')
plt.show()
```

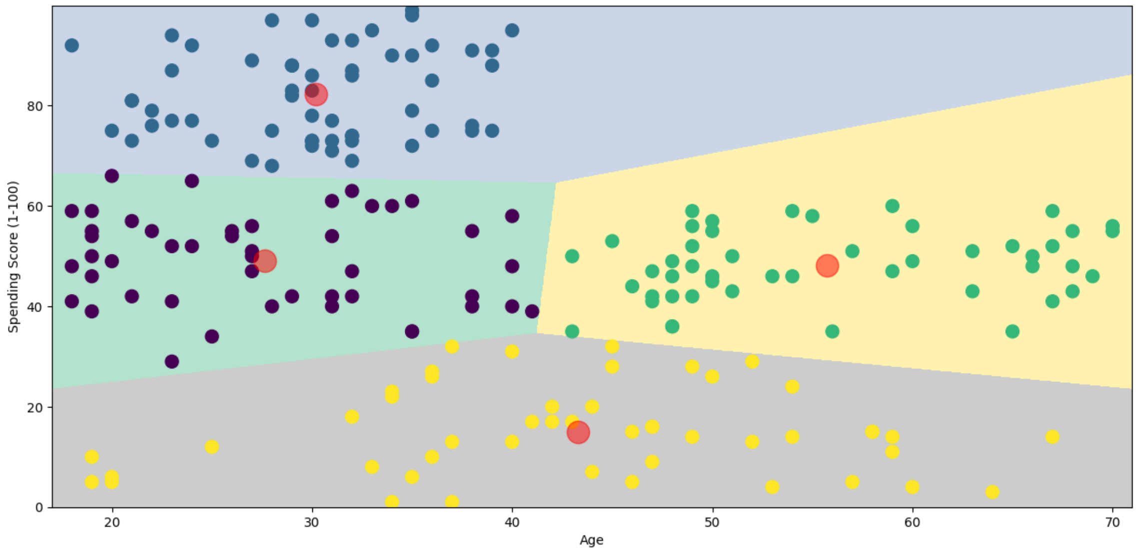


Applying KMeans for k=4

In [39]:
```
algorithm = (KMeans(n_clusters = 4 ,init='k-means++', n_init = 10 ,max_it
                tol=0.0001,  random_state= 111  , algorithm='elkan') )
algorithm.fit(X1)
labels1 = algorithm.labels_
centroids1 = algorithm.cluster_centers_
```

In [40]:
```
h = 0.02
x_min, x_max = X1[:, 0].min() - 1, X1[:, 0].max() + 1
y_min, y_max = X1[:, 1].min() - 1, X1[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h
Z = algorithm.predict(np.c_[xx.ravel(), yy.ravel()])
```

In [41]:
```
plt.figure(1 , figsize = (15 , 7) )
plt.clf()
Z = Z.reshape(xx.shape)
plt.imshow(Z , interpolation='nearest',
      extent=(xx.min(), xx.max(), yy.min(), yy.max()),
      cmap = plt.cm.Pastel2, aspect = 'auto', origin='lower')

plt.scatter( x = 'Age', y = 'Spending Score (1-100)', data = df, c = labels1, s =
plt.scatter(x = centroids1[: , 0] , y =  centroids1[: , 1] , s = 300 , c = 'red' , al
plt.ylabel('Spending Score (1-100)') , plt.xlabel('Age')
plt.show()
```

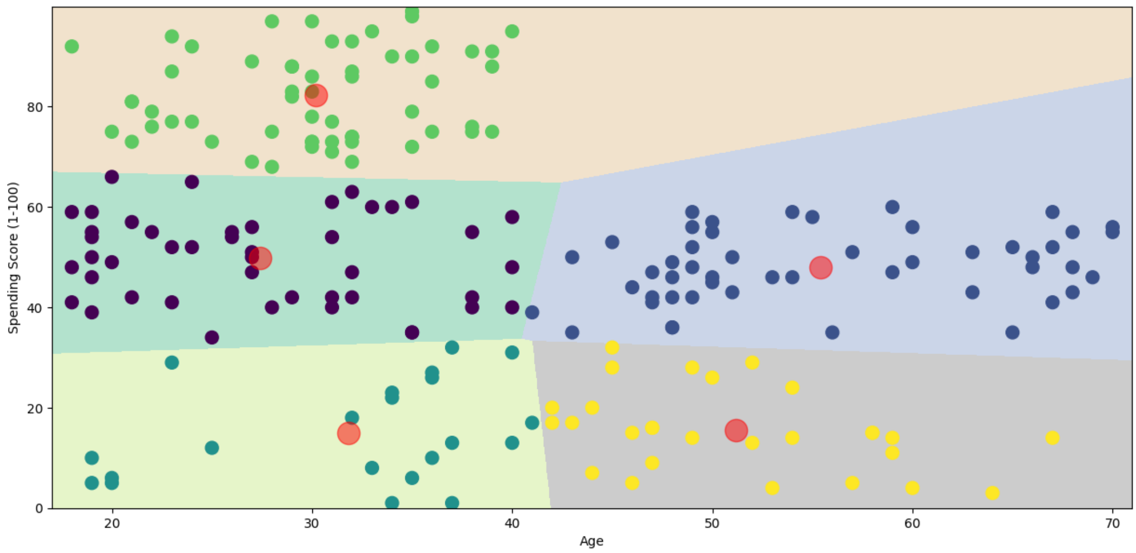


Applying KMeans for k=5

In [42]:
```
algorithm = (KMeans(n_clusters = 5, init='k-means++', n_init = 10, max_it
               tol=0.0001, random_state= 111 , algorithm='elkan'))
algorithm.fit(X1)
labels1 = algorithm.labels_
centroids1 = algorithm.cluster_centers_
```

In [43]:
```
h = 0.02
x_min, x_max = X1[:, 0].min() - 1, X1[:, 0].max() + 1
y_min, y_max = X1[:, 1].min() - 1, X1[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h
Z = algorithm.predict(np.c_[xx.ravel(), yy.ravel()])
```

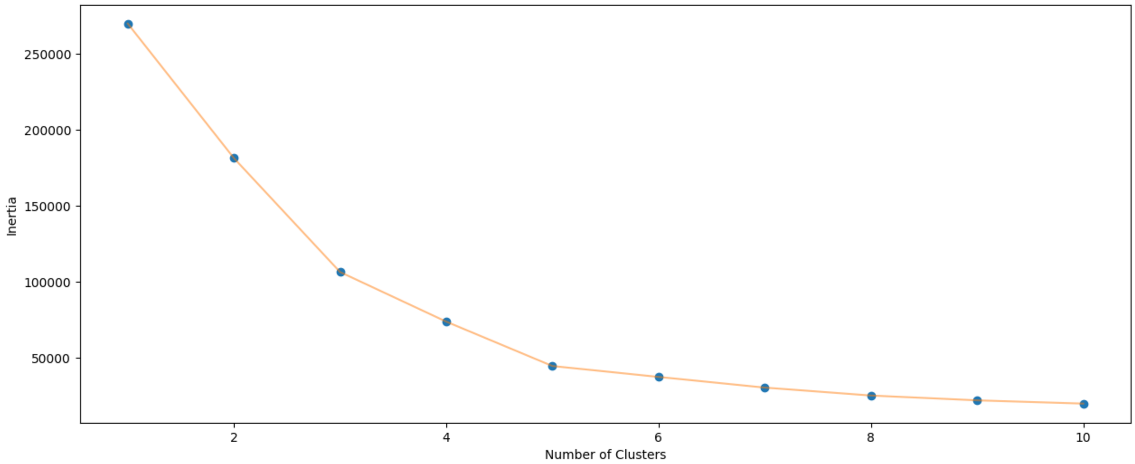
In [45]:
```
plt.figure(1 , figsize = (15 , 7) )
plt.clf()
Z = Z.reshape(xx.shape)
plt.imshow(Z , interpolation='nearest',
      extent=(xx.min(), xx.max(), yy.min(), yy.max()),
      cmap = plt.cm.Pastel2, aspect = 'auto', origin='lower')

plt.scatter( x = 'Age', y = 'Spending Score (1-100)', data = df, c = labels1, s =
plt.scatter(x = centroids1[: , 0] , y =  centroids1[: , 1] , s = 300 , c = 'red' , al
plt.ylabel('Spending Score (1-100)') , plt.xlabel('Age')
plt.show()
```



2D Clustering based on Annual Income and Spending Score

In [46]:
```
X2 = df[['Annual Income (k$)' , 'Spending Score (1-100)']].iloc[: , :].values
inertia = []
for n in range(1 , 11):
  algorithm = (KMeans(n_clusters = n ,init='k-means++', n_init = 10 ,max_
            tol=0.0001,  random_state= 111  , algorithm='elkan') )
  algorithm.fit(X2)
  inertia.append(algorithm.inertia_)
```

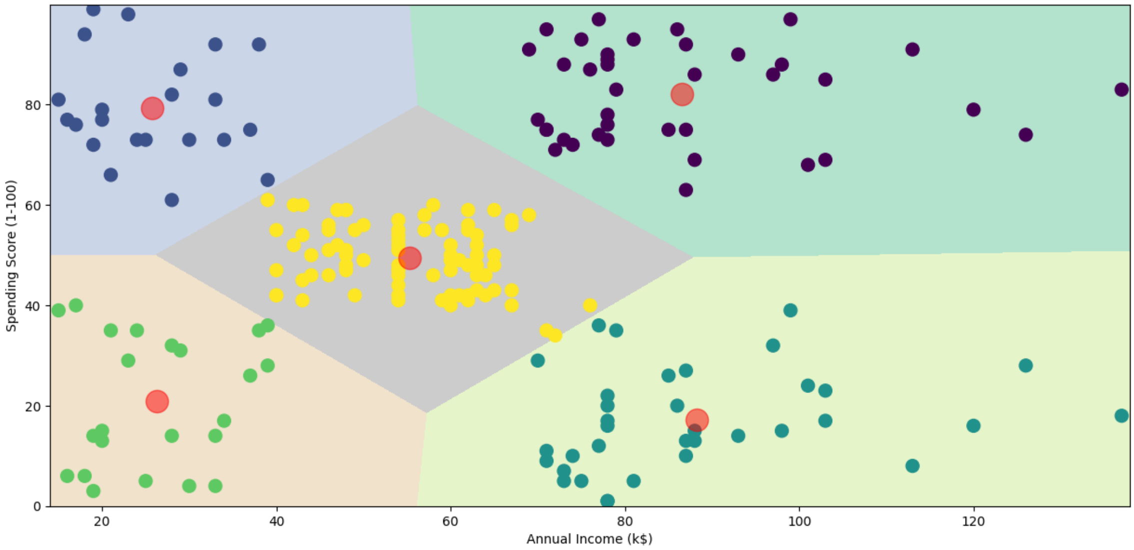
In [47]:
```
plt.figure(1 , figsize = (15 ,6))
plt.plot(np.arange(1 , 11) , inertia , 'o')
plt.plot(np.arange(1 , 11) , inertia , '-' , alpha = 0.5)
plt.xlabel('Number of Clusters') , plt.ylabel('Inertia')
plt.show()
```



In [48]:
```
algorithm = (KMeans(n_clusters = 5 ,init='k-means++', n_init = 10 ,max_it
                        tol=0.0001,  random_state= 111  , algorithm='elkan') )
algorithm.fit(X2)
labels2 = algorithm.labels_
centroids2 = algorithm.cluster_centers_
```

In [49]:
```
h = 0.02
x_min, x_max = X2[:, 0].min() - 1, X2[:, 0].max() + 1
y_min, y_max = X2[:, 1].min() - 1, X2[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h
Z2 = algorithm.predict(np.c_[xx.ravel(), yy.ravel()])
```

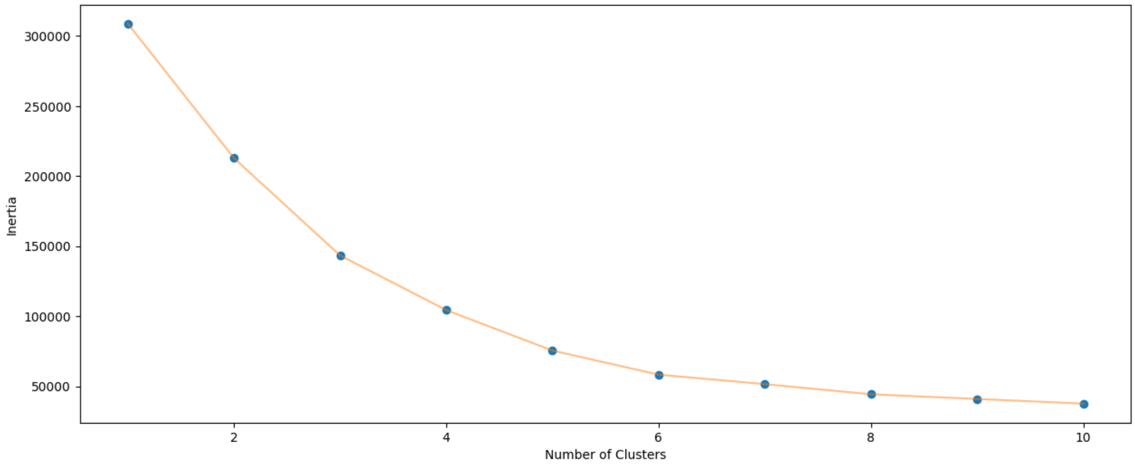
In [50]:
```
plt.figure(1 , figsize = (15 , 7) )
plt.clf()
Z2 = Z2.reshape(xx.shape)
plt.imshow(Z2 , interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap = plt.cm.Pastel2, aspect = 'auto', origin='lower')

plt.scatter( x = 'Annual Income (k$)' ,y = 'Spending Score (1-100)' , data = df ,
            s = 100 )
plt.scatter(x = centroids2[: , 0] , y =  centroids2[: , 1] , s = 300 , c = 'red' , al
plt.ylabel('Spending Score (1-100)') , plt.xlabel('Annual Income (k$)')
plt.show()
```



3D Clustering Age , Annual Income and Spending Score

In [51]:
```
X3 = df[['Age' , 'Annual Income (k$)' ,'Spending Score (1-100)']].iloc[: , :].values
inertia = []
for n in range(1 , 11):
    algorithm = (KMeans(n_clusters = n, init='k-means++', n_init = 10, max_
                        tol=0.0001, random_state= 111, algorithm='elkan'))
    algorithm.fit(X3)
    inertia.append(algorithm.inertia_)
```

In [52]:
```
plt.figure(1 , figsize = (15 ,6))
plt.plot(np.arange(1 , 11) , inertia , 'o')
plt.plot(np.arange(1 , 11) , inertia , '-' , alpha = 0.5)
plt.xlabel('Number of Clusters') , plt.ylabel('Inertia')
plt.show()
```



In [53]:
```
algorithm = (KMeans(n_clusters = 6 ,init='k-means++', n_init = 10 ,max_it
              tol=0.0001,  random_state= 111  , algorithm='elkan') )
algorithm.fit(X3)
labels3 = algorithm.labels_
centroids3 = algorithm.cluster_centers_

y_kmeans = algorithm.fit_predict(X3)
df['cluster'] = pd.DataFrame(y_kmeans)
df.head()
```

Out[53]:

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

In [54]:
```
import plotly as py
import plotly.graph_objs as go

trace1 = go.Scatter3d(
  x= df['Age'],
  y= df['Spending Score (1-100)'],
  z= df['Annual Income (k$)'],
  mode='markers',
   marker=dict(
    color = df['cluster'],
    size= 10,
    line=dict(
      color= df['cluster'],
      width= 12
    ),
    opacity=0.8
   )
)
data = [trace1]
layout = go.Layout(
  title= 'Clusters wrt Age, Income and Spending Scores',
  scene = dict(
      xaxis = dict(title  = 'Age'),
      yaxis = dict(title  = 'Spending Score'),
      zaxis = dict(title  = 'Annual Income')
    )
)
fig = go.Figure(data=data, layout=layout)
py.offline.iplot(fig)
```

Clusters wrt Age, Income and Spending Scores

In [ ]:

```

```