Directions for Unsupervised Machine Learning K-Means and DBSCAN

Image from: [https://www.kaggle.com/nareshbhat/outlier-the-silent-killer (https://www.kaggle.com/nareshbhat/outlier-the-silent-killer) (Links to an external site.)]

Anomaly Detection Using Unsupervised Algorithms Overview

Anomalies or outliers

Anomalies and outliers are often used interchangeably. Outliers are extreme data values that are distant from other values in your dataset. Outliers can distort the conclusions you draw from descriptive statistics (like mean and standard deviation) and from data visuals (like histograms and scatter plots). Many machine learning algorithms and statistical models are also sensitive to the range and distribution of data values used in your training data. Some of the most common causes of outliers in a dataset include data entry errors (human errors), measurement errors (instrument errors), and data processing errors (data manipulation errors). Consequently, outliers can come from multiple sources and hide in a single feature or n-dimensional feature space. Aside from the specific task of investigating these extreme data values (for anomaly detection) or in the case of naturally occurring outliers (novelties), the goal is often to remove or ignore outliers in your analysis.

The following Anomaly Detection Techniques are good examples that can be used for Customer Segmentation.

We all know that human diversity, whether it be gender, age, education, gender expression, or even food preferences, is a quality to be celebrated. While this is easier to accomplish in our dayto- day activities, from a business perspective, treating everyone as an individual is unrealistic, if not impossible, even though the business community is working on it. This is what motivates market segmentation. While you may not be able to serve your clients on an individual basis right now, you can do the next best thing by identifying their commonalities and differences, grouping them into small subgroups, and catering to their requirements. Even if ten thousand of your other clients received the same discount code in their email, this could give them the impression that you are communicating to them as an individual.

Clustering

Clustering is an example of an unsupervised machine learning algorithm. When new data is presented, the fundamental idea behind employing a clustering algorithm for anomaly detection is to group data based on the structure and variables without any prior knowledge about the data. Another way to say this is that the purpose is to find "natural" groups in an unlabeled dataset. This information is then used to determine whether or not there are any anomalous points present. Customer segmentation, text clustering, and image segmentation are the most popular clustering techniques. There are many clustering algorithms. It is important to choose the best algorithm, and the best one to use depends on the database, client requirements, and client expectations.

How do you segment your market?

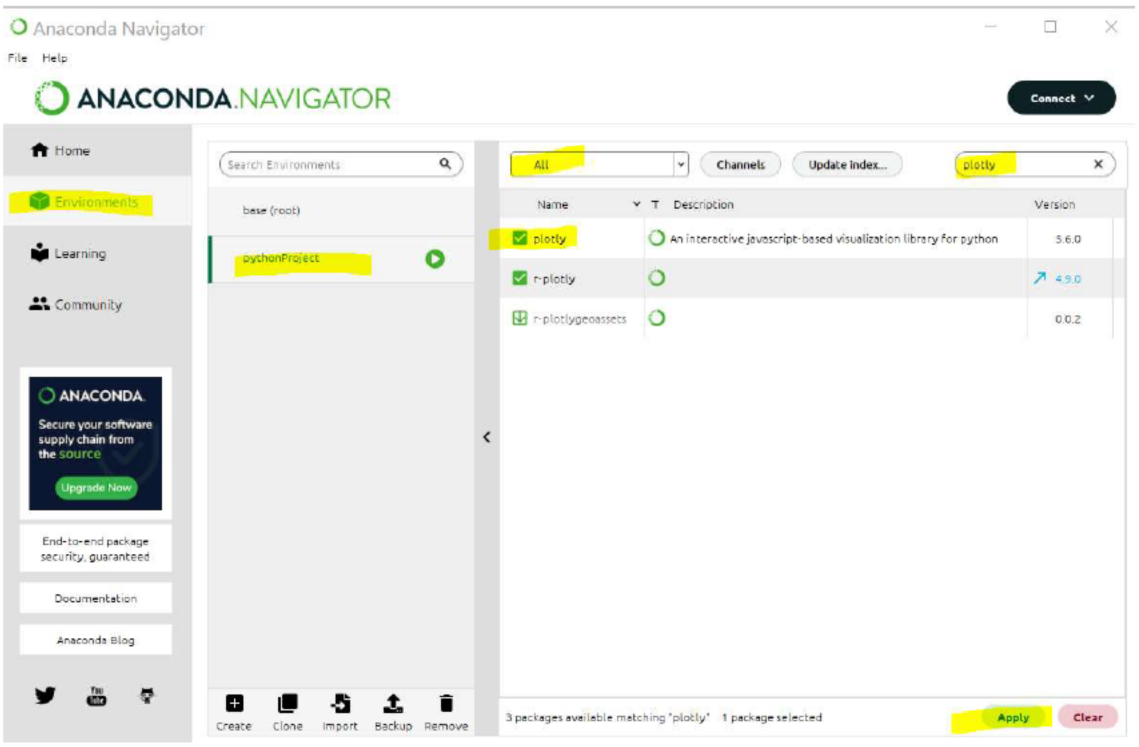
K-means clustering is usually one of the first methods used. Because k-means is one of, if not the only, data clustering techniques taught in statistics classrooms, it's only natural that business analysts would use it when a project requires market segmentation. As we learned in the lecture on k-means, k-means is simple to learn and effective with massive data sets. But, like all statistical approaches, k-means clustering relies on specific assumptions, and if those assumptions prove correct, the method is useless. Every data point is given to a cluster in the k-means clustering result, and the mean of the points is the centroid of each cluster. As a result, each point contributed to the cluster's formation. In reality, it's not uncommon for certain data points to stray from the "norm", resulting in anomalies or outliers. A single outlier in a dataset can cause the entire clustering result to be skewed. Furthermore, clusters formed using k-means are more spherical, yet, inaccuracies can develop if clusters are of other odd or elliptical shapes. Scree plots and silhouette plots can help you decide on parameter k; however, model sensitivity to anomalies significantly influences them. The goal of developing a new strategy is to overcome the drawbacks of existing methods. The majority of k-means drawbacks are dealt with or handled by DBSCAN. DBSCAN is a density-based non-parametric unsupervised learning algorithm. The DBSCAN has fewer assumptions, a more flexible model, and a larger range of applications than k-means. However, DBSCAN, like k-means, groups data according to their similarities using distance functions. DBSCAN also uses densities to form groups of data. The term "density" refers to the number of points in the region indicated by the model parameters.

Let's Get Started with Unsupervised Machine Learning

In the assignment, we will focus k-means and DBSCAN algorithms by using the Scikit-Learn Python library. From this dataset, we need to calculate some patterns; as it is an unsupervised method, we don't know what to calculate exactly. The column names will help you make sense of the data.

STEP 0: Install plotly by going to Anaconda Navigator and selecting the

Environments Tab. Select pythonProject, All packages, and search for plotly. Select plotly, and select Apply to start the installation. Once it has been installed go to Home and open Jupytner Notebook. Next, create a new Jupyter Notebook file, Name the file, and then Enter your name and date at the top of the file. SAVE file.



Name	Description	Version
plotly	An interactive javascript-based visualization library for python	5.6.0
r-plotly		4.9.0
r-plotlygeoassets		0.0.2

3 packages available matching "plotly" 1 package selected

STEP 1: Import Libraries

In [163]:
```python
# Import Python Libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
from sklearn.metrics import silhouette_score
from sklearn.cluster import DBSCAN
from sklearn.cluster import KMeans
from itertools import product
from mpl_toolkits.mplot3d import Axes3D
import plotly as py
import plotly.graph_objs as go
#filter warnings
import warnings
warnings.filterwarnings("ignore")

plt.rcParams.update({'font.size': 8})
```

STEP 2: Load Dataset & Get Dataset Shape

In [164]:
```python
mall_data = pd.read_csv('Mall_Customers.csv')
print('There are {} rows and {} columns in our dataset'.format(mall_data.shape[0],mall_data.shape[1]))

mall_data.head()
mall_data.describe()

```

```
There are 200 rows and 5 columns in our dataset
```

Out[164]:

	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

STEP 3: Clean the data

In [165]:
```
# We see there are no missing data points
mall_data.isnull().sum()
```

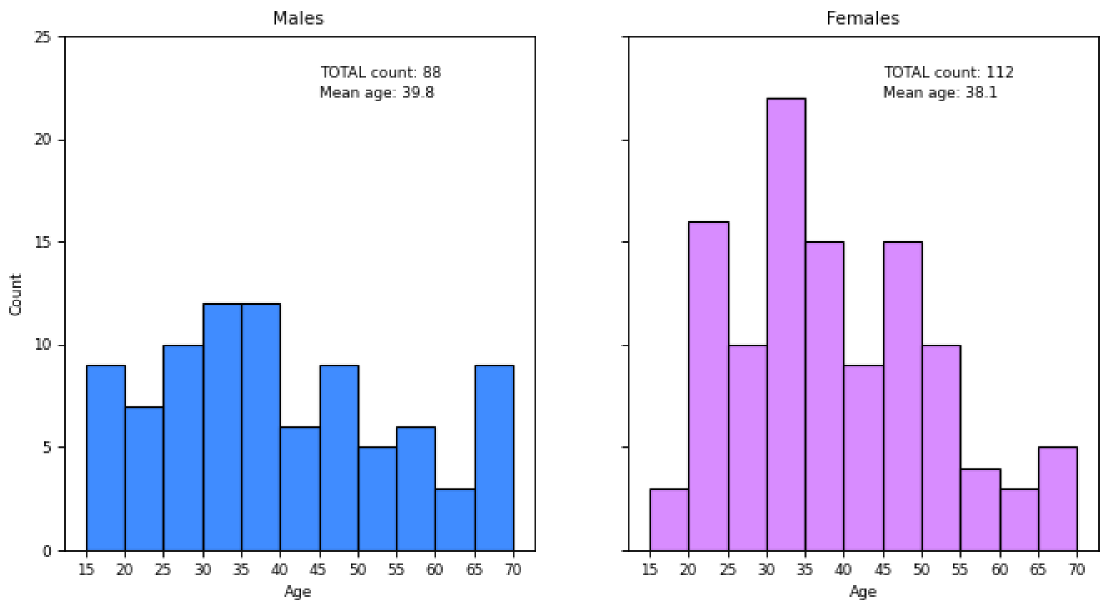
Out[165]:
```
CustomerID                0
Genre                     0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

STEP 4: Performing the Exploratory Data Analysis (EDA)

Distribution - Age

In [166]:
```
males_age = mall_data[mall_data['Genre']=='Male']['Age'] # subset with males age
females_age = mall_data[mall_data['Genre']=='Female']['Age'] # subset with females age
age_bins = range(15,75,5)
# males histogram
fig2, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5), sharey=True)
sns.histplot(males_age,bins=age_bins, color='#0066ff', ax=ax1)
ax1.set_xticks(age_bins)
ax1.set_ylim(top=25)
ax1.set_title('Males')
ax1.set_ylabel('Count')
ax1.text(45,23, "TOTAL count: {}".format(males_age.count()))
ax1.text(45,22, "Mean age: {:.1f}".format(males_age.mean()))
# females histogram
sns.histplot(females_age, bins=age_bins, color='#cc66ff', ax=ax2, )
ax2.set_xticks(age_bins)
ax2.set_title('Females')
ax2.set_ylabel('Count')
ax2.text(45,23, "TOTAL count: {}".format(females_age.count()))
ax2.text(45,22, "Mean age: {:.1f}".format(females_age.mean()))
plt.show()
print('Kolgomorov-Smirnov test p-value:{:.2f}'.format(stats.ks_2samp(males_age, females_age)[1]))
```

Figure 103



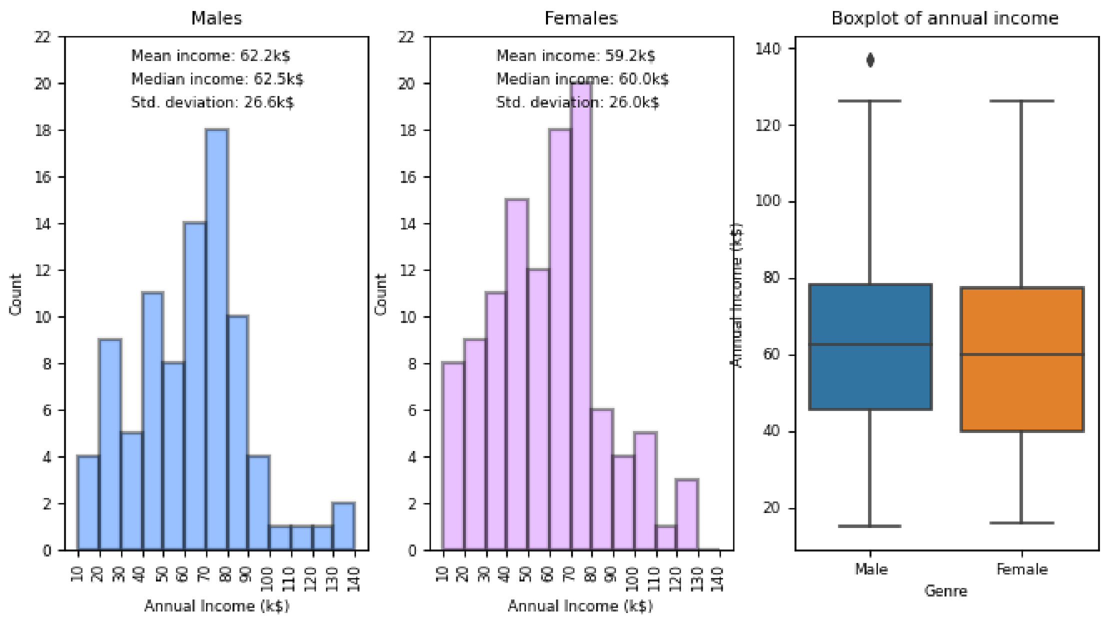
```
Kolgomorov-Smirnov test p-value:0.49
```

Male age is more evenly distributed than females. The biggest age group for women is between 30-35, and for men, it is between 30-40. The Kolmogorov-Smirnov test shows the difference between the males and females age groups is statistically insignificant as it calculates the P-value. A P-value that is less than or equal to 0.05 is statistically significant.

Distribution - Annual Income (k$)

In [167]:
```python
males_income = mall_data[mall_data['Genre']=='Male']['Annual Income (k$)']
# subset with males income
females_income = mall_data[mall_data['Genre']=='Female']['Annual Income (k$)']
# subset with females income

my_bins = range(10,150,10)
# males histogram
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(10,5))
sns.distplot(males_income, bins=my_bins, kde=False, color='#0066ff', ax=ax1,
hist_kws=dict(edgecolor="k", linewidth=2))

ax1.set_xticks(my_bins)
ax1.set_xticklabels(ax1.get_xticks(), rotation = 90)

ax1.set_yticks(range(0,24,2))
ax1.set_ylim(0,22)
ax1.set_title('Males')
ax1.set_ylabel('Count')
ax1.text(35,21, "Mean income: {:.1f}k$".format(males_income.mean()))
ax1.text(35,20, "Median income: {:.1f}k$".format(males_income.median()))
ax1.text(35,19, "Std. deviation: {:.1f}k$".format(males_income.std()))
# females histogram
sns.distplot(females_income, bins=my_bins, kde=False, color='#cc66ff',
ax=ax2, hist_kws=dict(edgecolor="k", linewidth=2))
ax2.set_xticks(my_bins)
ax2.set_xticklabels(ax2.get_xticks(), rotation = 90)

ax2.set_yticks(range(0,24,2))
ax2.set_ylim(0,22)
ax2.set_title('Females')
ax2.set_ylabel('Count')
ax2.text(35,21, "Mean income: {:.1f}k$".format(females_income.mean()))
ax2.text(35,20, "Median income: {:.1f}k$".format(females_income.median()))
ax2.text(35,19, "Std. deviation: {:.1f}k$".format(females_income.std()))
# boxplot
sns.boxplot(x='Genre', y='Annual Income (k$)', data=mall_data, ax=ax3)
ax3.set_title('Boxplot of annual income')
plt.show()
print('Kolgomorov-Smirnov test p-value:{:.2f}'.format(stats.ks_2samp(males_income, females_income)[1]))
```

Figure 104

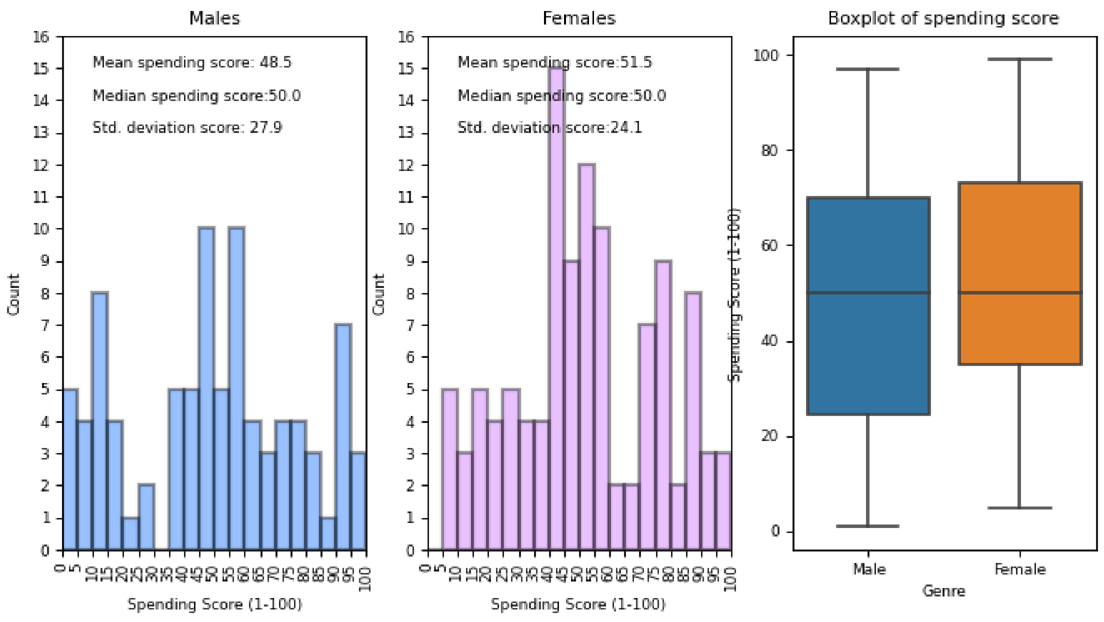


```
Kolgomorov-Smirnov test p-value:0.78
```

Distribution - Spending Score (1-100)

In [168]:
```python
males_spending = mall_data[mall_data['Genre']=='Male']['Spending Score (1-100)']
# subset with males age
females_spending = mall_data[mall_data['Genre']=='Female']['Spending Score (1-100)']
# subset with females age
spending_bins = range(0,105,5)
# males histogram
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(10,5))
sns.distplot(males_spending, bins=spending_bins, kde=False, color='#0066ff',
ax=ax1, hist_kws=dict(edgecolor="k", linewidth=2))
ax1.set_xticks(spending_bins)
ax1.set_xticklabels(ax1.get_xticks(), rotation = 90)

ax1.set_xlim(0,100)
ax1.set_yticks(range(0,17,1))
ax1.set_ylim(0,16)
ax1.set_title('Males')
ax1.set_ylabel('Count')
ax1.text(10,15, "Mean spending score: {:.1f}".format(males_spending.mean()))
ax1.text(10,14, "Median spending score:{:.1f}".format(males_spending.median()))
ax1.text(10,13, "Std. deviation score: {:.1f}".format(males_spending.std()))
# females histogram
sns.distplot(females_spending, bins=spending_bins, kde=False,
color='#cc66ff', ax=ax2, hist_kws=dict(edgecolor="k", linewidth=2))
ax2.set_xticks(spending_bins)
ax2.set_xticklabels(ax2.get_xticks(), rotation = 90)

ax2.set_xlim(0,100)
ax2.set_yticks(range(0,17,1))
ax2.set_ylim(0,16)
ax2.set_title('Females')
ax2.set_ylabel('Count')
ax2.text(10,15, "Mean spending score:{:.1f}".format(females_spending.mean()))
ax2.text(10,14, "Median spending score:{:.1f}".format(females_spending.median()))
ax2.text(10,13, "Std. deviation score:{:.1f}".format(females_spending.std()))
# boxplot
sns.boxplot(x='Genre', y='Spending Score (1-100)', data=mall_data, ax=ax3)
ax3.set_title('Boxplot of spending score')
plt.show()
print('Kolgomorov-Smirnov test p-value:{:.2f}'.format(stats.ks_2samp(males_spending, females_spending)[1]))
```

Figure 105

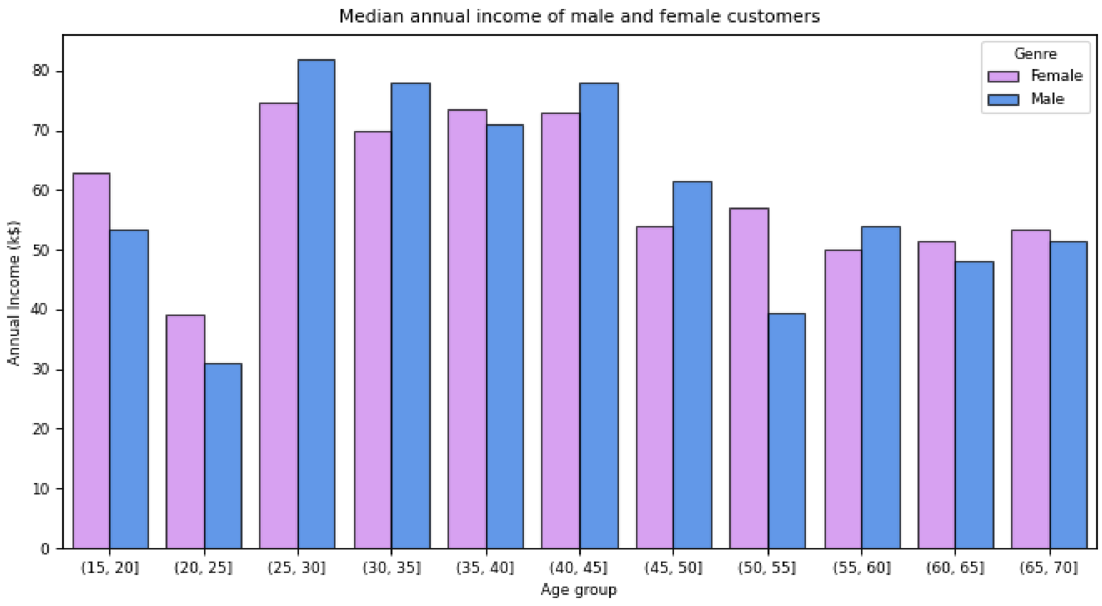


```
Kolgomorov-Smirnov test p-value:0.29
```

Distribution - Median Annual Income of Male and Female Customers

```
In [169]:
medians_by_age_group = mall_data.groupby(["Genre",pd.cut(mall_data['Age'], age_bins)]).median()
medians_by_age_group.index = medians_by_age_group.index.set_names(['Genre', 'Age_group'])
medians_by_age_group.reset_index(inplace=True)
fig, ax = plt.subplots(figsize=(10,5))
sns.barplot(x='Age_group', y='Annual Income (k$)', hue='Genre',
            data=medians_by_age_group, palette=['#cc66ff','#0066ff'],
            alpha=0.7,edgecolor='k',ax=ax)
ax.set_title('Median annual income of male and female customers')
ax.set_xlabel('Age group')
plt.show()
```

Figure 106

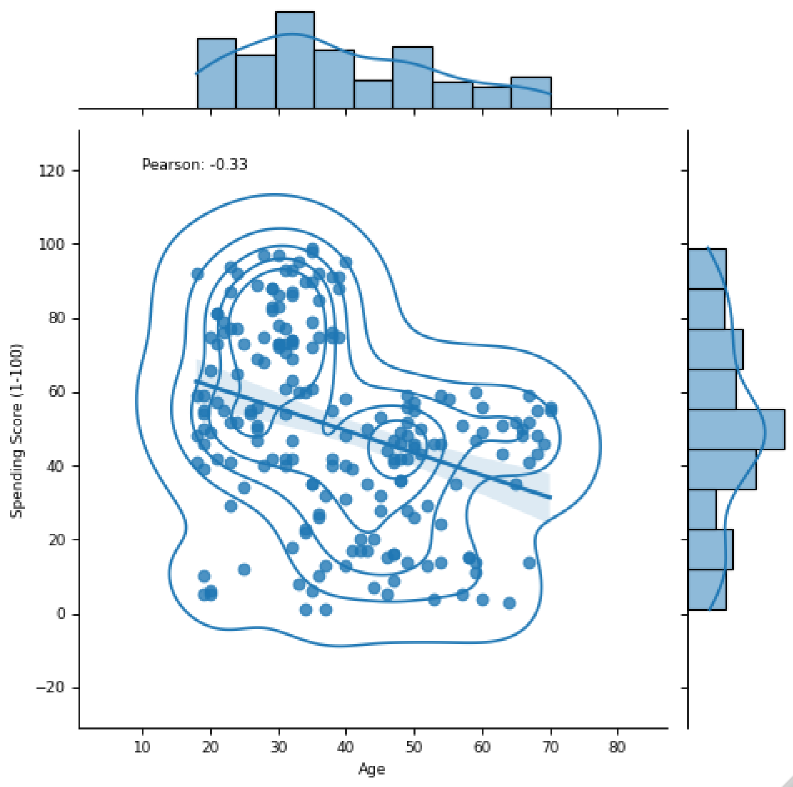


Correlations

Pearson's Correlation for Age & Spending Score (1-100) with jointplot

In [197]:
```python
from scipy.stats import pearsonr
# calculating Pearson's correlation
corr, _ = pearsonr(mall_data['Age'], mall_data['Spending Score (1-100)'])
jp = sns.jointplot(x='Age', y='Spending Score (1-100)', data=mall_data, kind='reg').plot_joint(
                    sns.kdeplot, zorder=0, n_levels=6)

plt.text(10,120, 'Pearson: {:.2f}'.format(corr))
plt.show()
```

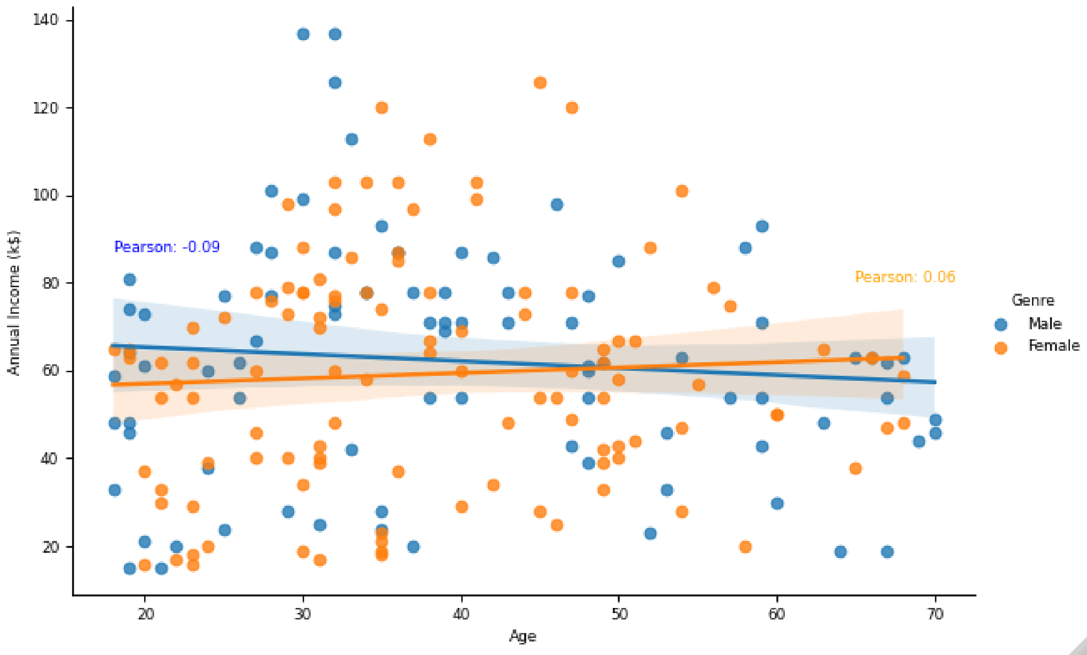
Figure 120



Pearson's Correlation for Age & Annual Income with lineplot (lmplot)

In [202]:
```python
# calculating Pearson's correlations
corr1, _ = pearsonr(males_age.values, males_income.values)
corr2, _ = pearsonr(females_age.values, females_income.values)
sns.lmplot(x='Age', y='Annual Income (k$)', data=mall_data, hue='Genre',aspect=1.5)
plt.text(18,87, 'Pearson: {:.2f}'.format(corr1), color='blue')
plt.text(65,80, 'Pearson: {:.2f}'.format(corr2), color='orange')
plt.show()
```

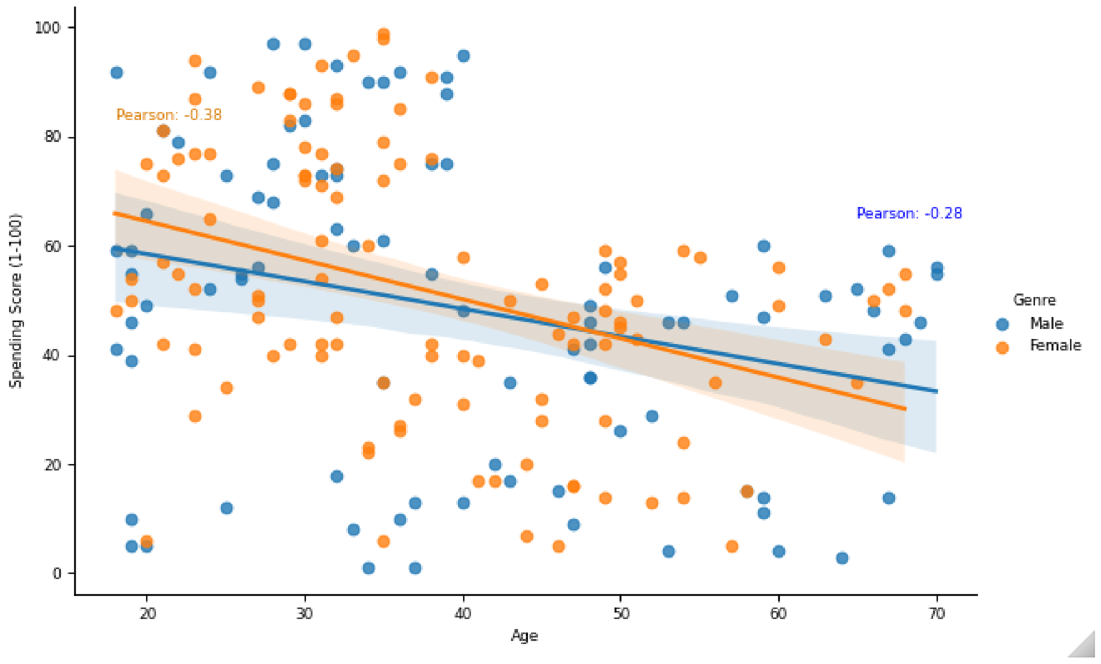
Figure 125



Pearson's Correlation for Age & Spending Score (1-100) with line plot (lmplot)

In [203]:
```
# calculating Pearson's correlations
corr1, _ = pearsonr(males_age.values, males_spending.values)
corr2, _ = pearsonr(females_age.values, females_spending.values)
sns.lmplot(x='Age', y='Spending Score (1-100)', data=mall_data, hue='Genre', aspect=1.5)
plt.text(65,65, 'Pearson: {:.2f}'.format(corr1), color='blue')
plt.text(18,83, 'Pearson: {:.2f}'.format(corr2), color='#d97900')
plt.show()
```

Figure 126

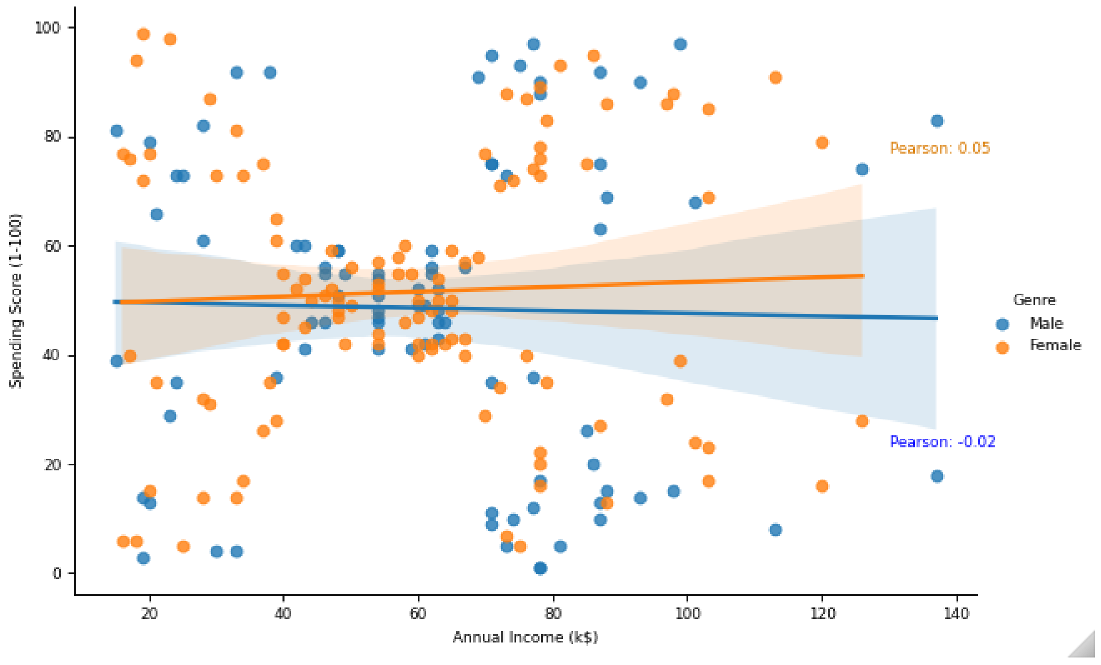


x=57.8 y=25.1

Pearson's Correlation for Annual Income & Spending Score (1-100) with line plot (lmplot)

In [173]:
```
# calculating Pearson's correlations
corr1, _ = pearsonr(males_income.values, males_spending.values)
corr2, _ = pearsonr(females_income.values, females_spending.values)
sns.lmplot(x='Annual Income (k$)', y='Spending Score (1-100)', data=mall_data,
hue='Genre', aspect=1.5)
plt.text(130,23, 'Pearson: {:.2f}'.format(corr1), color='blue')
plt.text(130,77, 'Pearson: {:.2f}'.format(corr2), color='#d97900')
plt.show()
```

Figure 110



STEP 5: K-Means

The k-means algorithm is the most used partitional clustering technique. It gained popularity since it is so easy and simple to implement, and it has had many successes in many diverse sectors. As we learned in our lecture for this module, k-means is a greedy algorithm. Remember with the greedy algorithm to be aware of global optimization. Also, remember that the Euclidean distance is implemented, and the number of clusters has to be predefined. In this assignment, you will use the Elbow Method and Silhouette Score to calculate the k-value. For clustering, you will only use numerical columns.

In [174]:
```
X_numerics = mall_data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]
# subset with numeric variables only
```

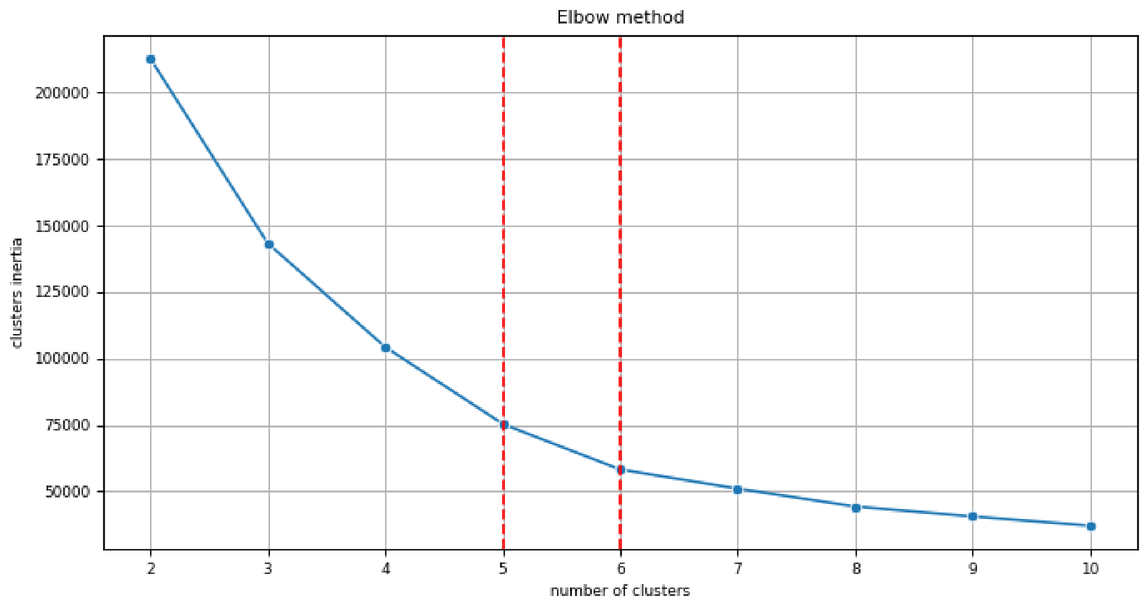
Elbow Method

In order to find the optimal K number, look at the "elbow" in the graph.

In [175]:
```
n_clusters = [2,3,4,5,6,7,8,9,10] # number of clusters
clusters_inertia = [] # inertia of clusters
s_scores = [] # silhouette scores
for n in n_clusters:
    KM_est = KMeans(n_clusters=n, init='k-means++').fit(X_numerics)
    clusters_inertia.append(KM_est.inertia_) # data for the elbow method
    silhouette_avg = silhouette_score(X_numerics, KM_est.labels_)
    s_scores.append(silhouette_avg) # data for the silhouette score method
```

In [176]:
```
fig, ax = plt.subplots(figsize=(10,5))
ax = sns.lineplot(x=n_clusters, y=clusters_inertia, marker='o', ax=ax)
ax.set_title("Elbow method")
ax.set_xlabel("number of clusters")
ax.set_ylabel("clusters inertia")
ax.axvline(5, ls="--", c="red")
ax.axvline(6, ls="--", c="red")
plt.grid()
plt.show()
```

Figure 111

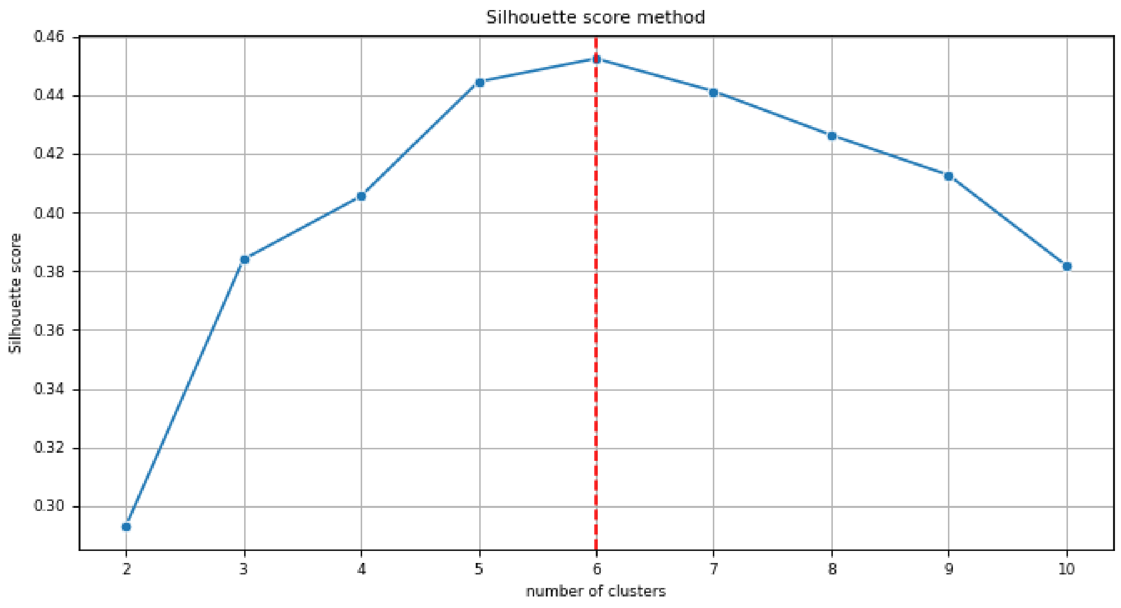


Silhouette Score Method

Since there is no clear "elbow", the Silhouette Score will also be utilized. The Silhouette Score illustrates the distance between clusters, ranging between -1 and 1, and the best score is the number closest to 1.

In [177]:
```python
fig, ax = plt.subplots(figsize=(10,5))
ax = sns.lineplot(x=n_clusters, y=s_scores, marker='o', ax=ax)
ax.set_title("Silhouette score method")
ax.set_xlabel("number of clusters")
ax.set_ylabel("Silhouette score")
ax.axvline(6, ls="--", c="red")
plt.grid()
plt.show()
```

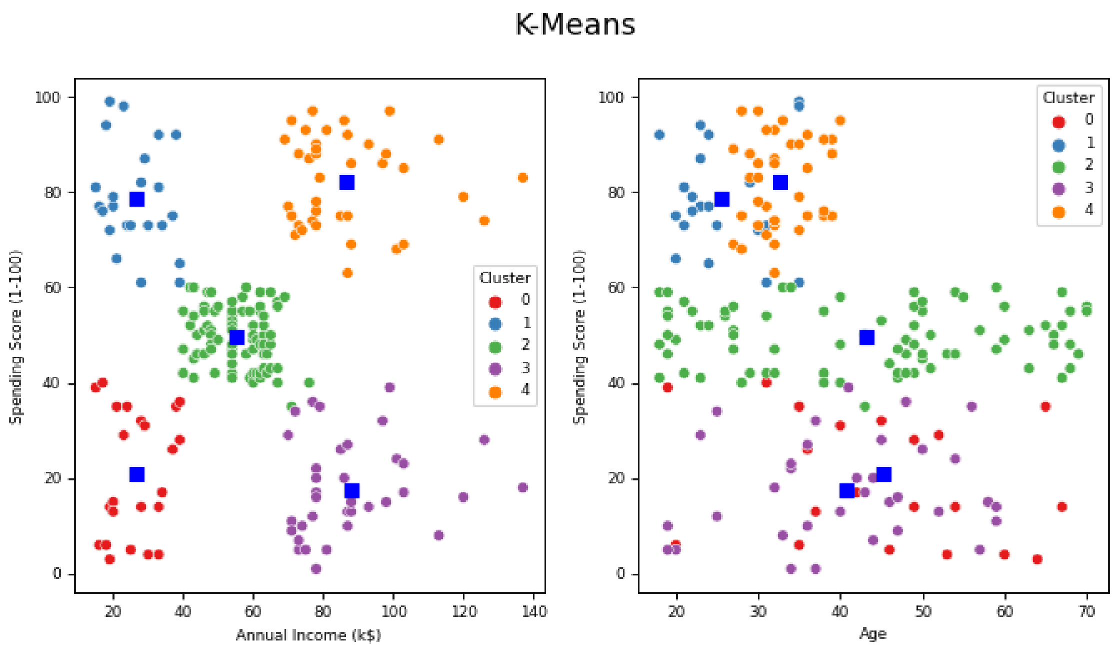
Figure 112



Let's try 5 Clusters

In [178]:
```python
# initialize and fit K-Means model
KM_5_clusters = KMeans(n_clusters=5, init='k-means++').fit(X_numerics)
KM5_clustered = X_numerics.copy()
# append labels to points
KM5_clustered.loc[:,'Cluster'] = KM_5_clusters.labels_
fig1, (axes) = plt.subplots(1,2,figsize=(10,5))
scat_1 = sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',
data=KM5_clustered,hue='Cluster', ax=axes[0], palette='Set1', legend='full')
sns.scatterplot(x='Age', y='Spending Score (1-100)', data=KM5_clustered,
hue='Cluster', palette='Set1', ax=axes[1], legend='full')
axes[0].scatter(KM_5_clusters.cluster_centers_[:,1],KM_5_clusters.cluster_centers_[:,2], marker='s', s=40, c="blue")
axes[1].scatter(KM_5_clusters.cluster_centers_[:,0],KM_5_clusters.cluster_centers_[:,2], marker='s', s=40, c="blue")
plt.show()
```

Figure 113



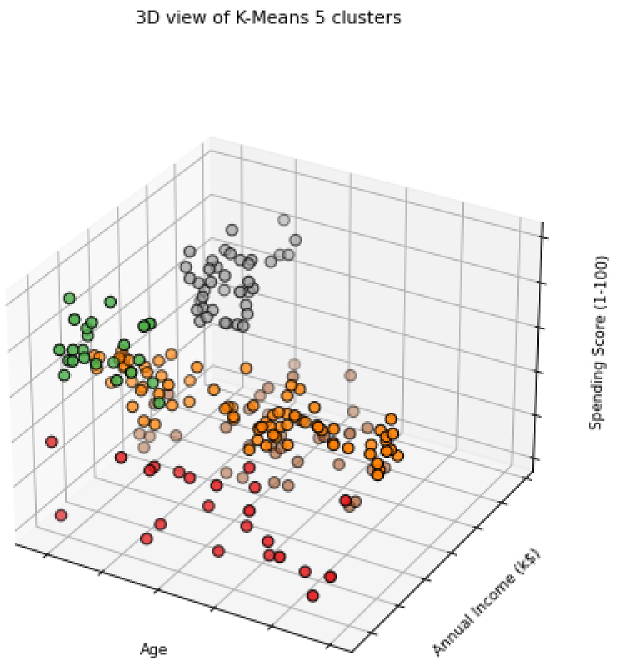
In [179]:
```python
# Size of clusters
KM_clust_sizes = KM5_clustered.groupby('Cluster').size().to_frame()
KM_clust_sizes.columns = ["KM_size"]
KM_clust_sizes
```

Out[179]:

Cluster	KM_size
0	23
1	23
2	79
3	36
4	39

In [180]:
```python
# if not installed
#!pip install ipympl

#Enable Javascript in your browser

#Create a 3D projection of 5 generated clusters
from mpl_toolkits.mplot3d import Axes3D
# for creating a responsive plot
%matplotlib widget

# creating figure
fig = plt.figure(figsize=(7, 7))
ax=fig.add_subplot(111,projection='3d')

ax.scatter(KM5_clustered['Age'],
    KM5_clustered['Annual Income (k$)'],
    KM5_clustered['Spending Score (1-100)'],
    c=KM5_clustered['Cluster'],
    s=35, edgecolor='k', cmap=plt.cm.Set1)
ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_xlabel('Age')
ax.set_ylabel('Annual Income (k$)')
ax.set_zlabel('Spending Score (1-100)')
ax.set_title('3D view of K-Means 5 clusters')
ax.dist = 12
plt.show()
```

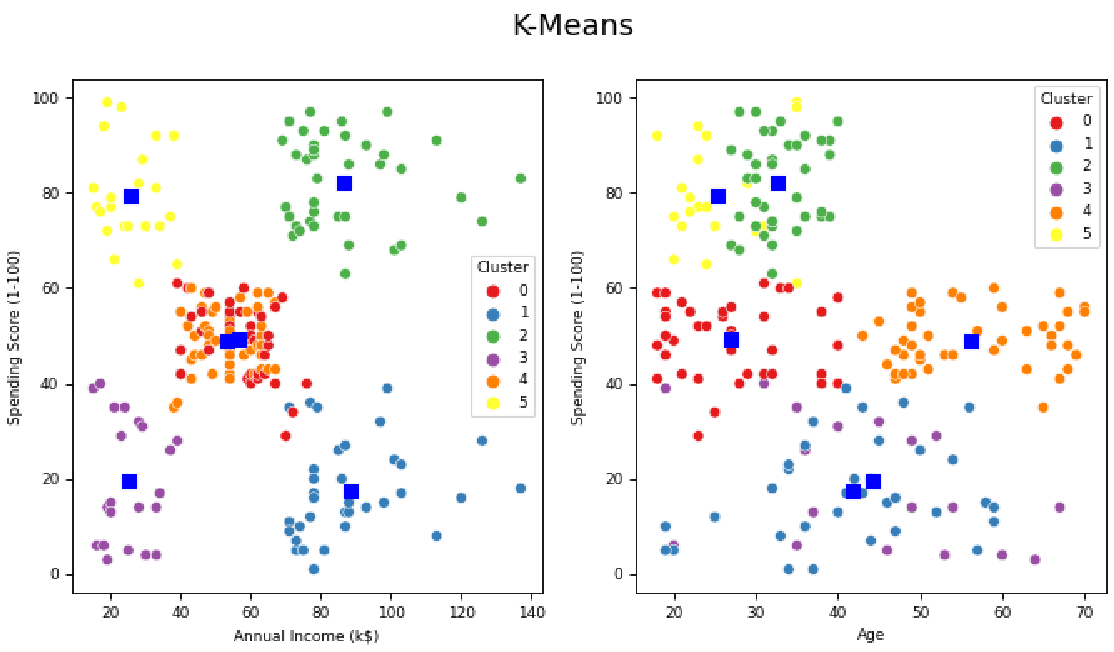
Figure 114



Let's try 6 Clusters

In [181]:
```
#initialize and fit K-Means model
KM_6_clusters = KMeans(n_clusters=6, init='k-means++').fit(X_numerics)
KM6_clustered = X_numerics.copy()
# append labels to points
KM6_clustered.loc[:,'Cluster'] = KM_6_clusters.labels_
fig11, (axes) = plt.subplots(1,2,figsize=(10,5))
sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',
data=KM6_clustered,hue='Cluster', ax=axes[0], palette='Set1', legend='full')
sns.scatterplot(x='Age', y='Spending Score (1-100)', data=KM6_clustered,
hue='Cluster', palette='Set1', ax=axes[1], legend='full')
# plotting centroids
axes[0].scatter(KM_6_clusters.cluster_centers_[:,1],
KM_6_clusters.cluster_centers_[:,2], marker='s', s=40, c="blue")
axes[1].scatter(KM_6_clusters.cluster_centers_[:,0],
KM_6_clusters.cluster_centers_[:,2], marker='s', s=40, c="blue")
plt.show()
```

Figure 115

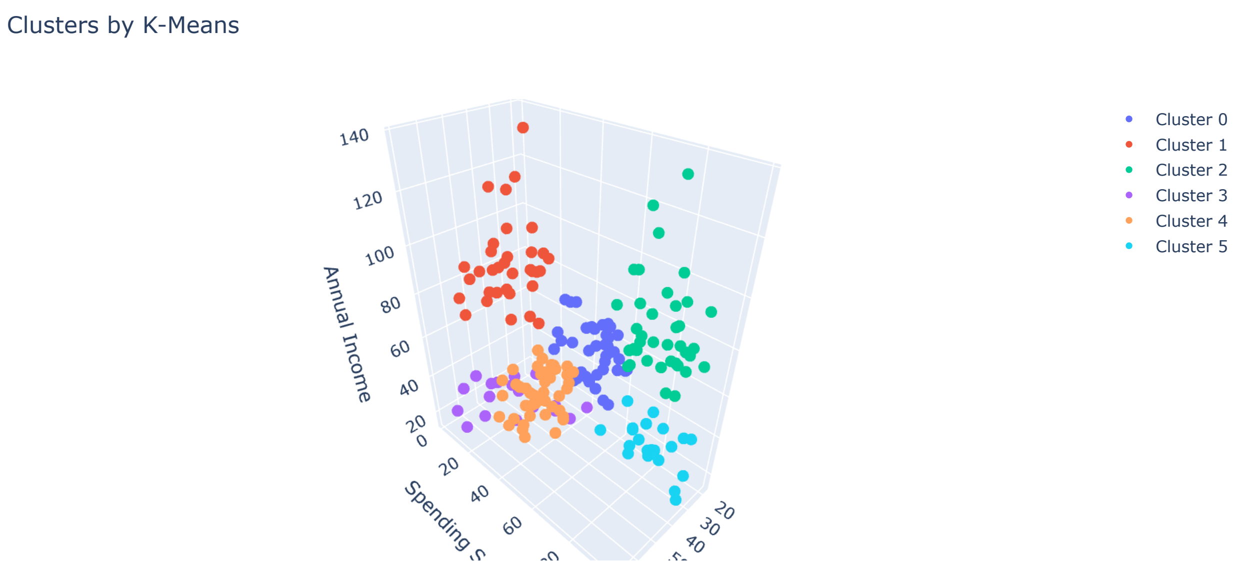


In [182]:
```
# Size of Clusters
KM6_clust_sizes = KM6_clustered.groupby('Cluster').size().to_frame()
KM6_clust_sizes.columns = ["KM_size"]
KM6_clust_sizes
```

Out[182]:

	KM_size
Cluster	
0	38
1	35
2	39
3	21
4	45
5	22

In [204]:
```
# Create an interactive 3D view of K-Means with 6 clusters with Plotly
def tracer(db, n, name):
    '''
    This function returns trace object for Plotly
    '''
    return go.Scatter3d(
        x = db[db['Cluster']==n]['Age'],
        y = db[db['Cluster']==n]['Spending Score (1-100)'],
        z = db[db['Cluster']==n]['Annual Income (k$)'],
        mode = 'markers',
        name = name,
        marker = dict(size = 5)
    )
trace0 = tracer(KM6_clustered, 0, 'Cluster 0')
trace1 = tracer(KM6_clustered, 1, 'Cluster 1')
trace2 = tracer(KM6_clustered, 2, 'Cluster 2')
trace3 = tracer(KM6_clustered, 3, 'Cluster 3')
trace4 = tracer(KM6_clustered, 4, 'Cluster 4')
trace5 = tracer(KM6_clustered, 5, 'Cluster 5')
data = [trace0, trace1, trace2, trace3, trace4, trace5]
layout = go.Layout(
    title = 'Clusters by K-Means',
    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)
```



STEP 6: DBSCAN (Density-Based Spatial Clustering of Applications with

Noise) The basis of this algorithm is based on the concept of density. We will use the distance (eps) and the minimum number of points within the distance as parameters. Again, we will use the Euclidean distance for the DBSCAN. Create a matrix of investigated combinations and the number of generated clusters.

In [184]:
```
# create a matrix of investigated combinations.
eps_values = np.arange(8,12.75,0.25) # eps values to be investigated
min_samples = np.arange(3,10) # min_samples values to be investigated
DBSCAN_params = list(product(eps_values, min_samples))
```

In [185]:
```
# collecting number of generated clusters
no_of_clusters = []
sil_score = []
for p in DBSCAN_params:
    DBS_clustering = DBSCAN(eps=p[0], min_samples=p[1]).fit(X_numerics)
    no_of_clusters.append(len(np.unique(DBS_clustering.labels_)))
    sil_score.append(silhouette_score(X_numerics, DBS_clustering.labels_))
```

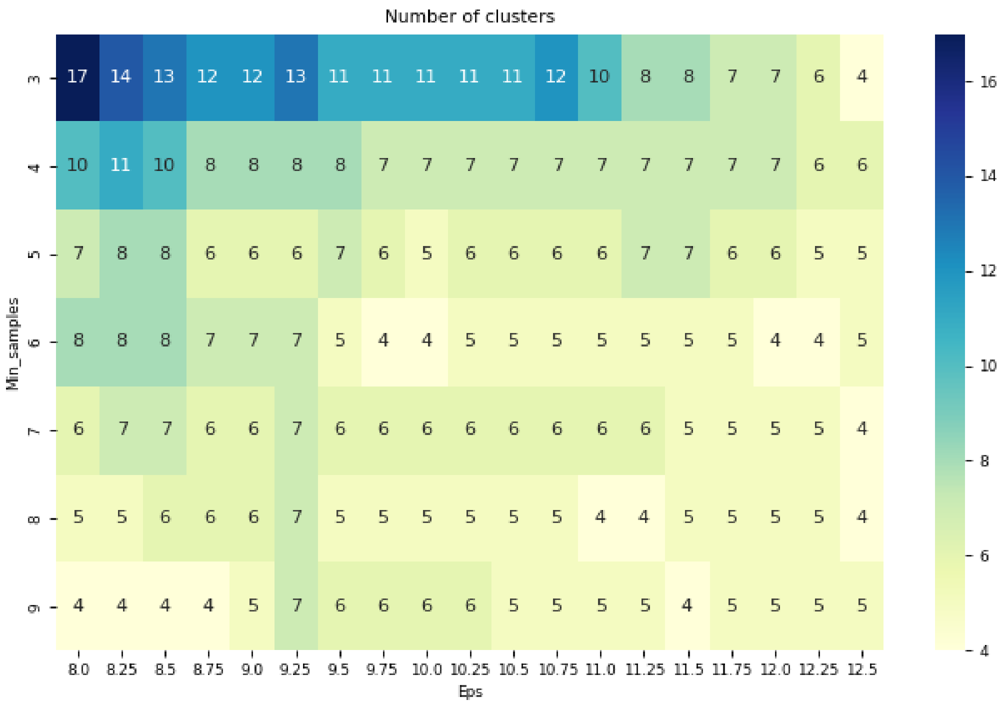
Create Heatmaps

The heatplot illustrates how many clusters were generated by the algorithm with the respective parameters combinations created above. You can see there are clusters from 17 to 4.

In [186]:

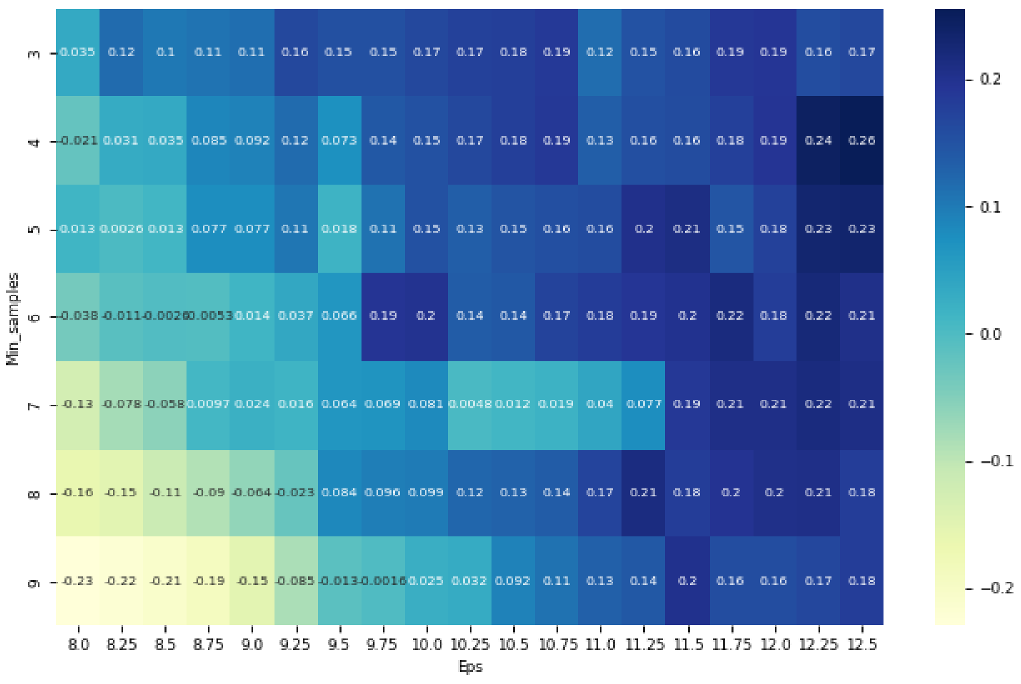
```python
tmp = pd.DataFrame.from_records(DBSCAN_params, columns =['Eps', 'Min_samples'])
tmp['No_of_clusters'] = no_of_clusters
pivot_1 = pd.pivot_table(tmp, values='No_of_clusters', index='Min_samples',
columns='Eps')
fig, ax = plt.subplots(figsize=(10,6))
sns.heatmap(pivot_1, annot=True,annot_kws={"size": 10}, cmap="YlGnBu", ax=ax)
ax.set_title('Number of clusters')
plt.show()
```

Figure 116



In [187]:
```python
# In the heatmap below, we see the Global maximum is 0.26
# for eps=12.5 and
min_samples=4.
tmp = pd.DataFrame.from_records(DBSCAN_params, columns =['Eps', 'Min_samples'])
tmp['Sil_score'] = sil_score
pivot_1 = pd.pivot_table(tmp, values='Sil_score', index='Min_samples',
columns='Eps')
fig, ax = plt.subplots(figsize=(10,6))
sns.heatmap(pivot_1, annot=True, annot_kws={"size": 7}, cmap="YlGnBu", ax=ax)
plt.show()
```

Figure 117



In [188]:
```python
# Based on the graph above let's use the Global
# maximum settings (eps = 12.5 and min_samples = 4)
DBS_clustering = DBSCAN(eps=12.5, min_samples=4).fit(X_numerics)
DBSCAN_clustered = X_numerics.copy()
DBSCAN_clustered.loc[:,'Cluster'] = DBS_clustering.labels_
# append labels to points
```

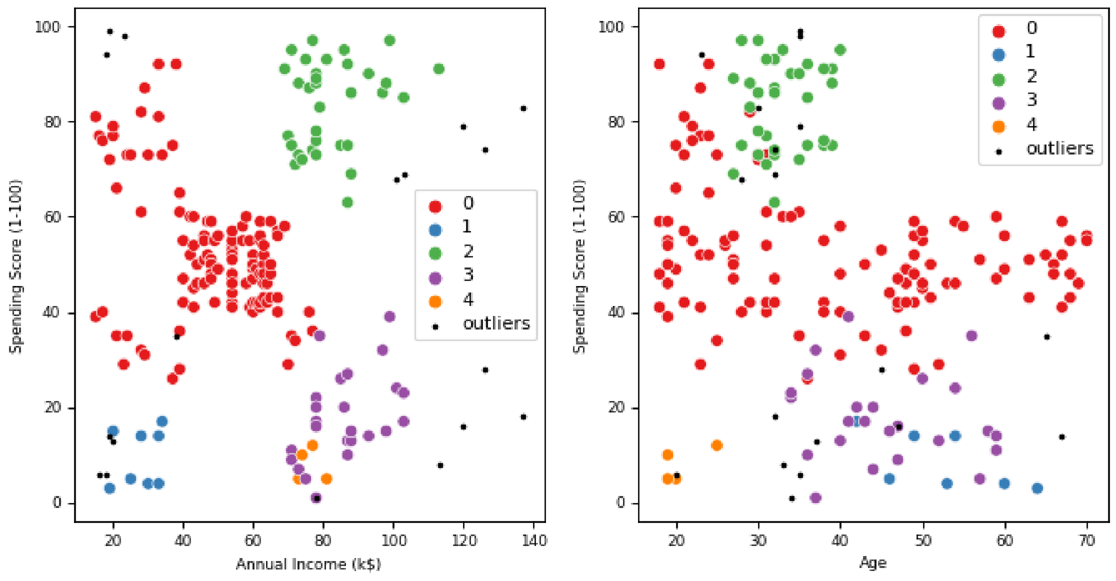
In [189]:
```python
# With the following code, you will see that 5
# clusters are created AND one outlier cluster (-1) was created.
# You can also see that the size of clusters varies greatly,
# and some only have a few observations.
# Lastly, you see there are 18 outliers (-1).
DBSCAN_clust_sizes = DBSCAN_clustered.groupby('Cluster').size().to_frame()
DBSCAN_clust_sizes.columns = ["DBSCAN_size"]
DBSCAN_clust_sizes
```

Out[189]:

	DBSCAN_size
Cluster	
-1	18
0	112
1	8
2	34
3	24
4	4

In [205]:
```python
# With this code, you will see that there are
# some outliers (small black dots).
#These points do not meet distance and minimum
# samples requirements to be recognized as a cluster.
outliers = DBSCAN_clustered[DBSCAN_clustered['Cluster']==-1]
fig2, (axes) = plt.subplots(1,2,figsize=(10,5))
sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',
data=DBSCAN_clustered[DBSCAN_clustered['Cluster']!=-1],
hue='Cluster', ax=axes[0], palette='Set1', legend='full', s=45)
sns.scatterplot(x='Age', y='Spending Score (1-100)',
data=DBSCAN_clustered[DBSCAN_clustered['Cluster']!=-1],
hue='Cluster', palette='Set1', ax=axes[1], legend='full', s=45)
axes[0].scatter(outliers['Annual Income (k$)'], outliers['Spending Score (1-100)'],
s=5, label='outliers', c="k")
axes[1].scatter(outliers['Age'], outliers['Spending Score (1-100)'], s=5,
label='outliers', c="k")
axes[0].legend()
axes[1].legend()
plt.setp(axes[0].get_legend().get_texts(), fontsize='10')
plt.setp(axes[1].get_legend().get_texts(), fontsize='10')
plt.show()
```

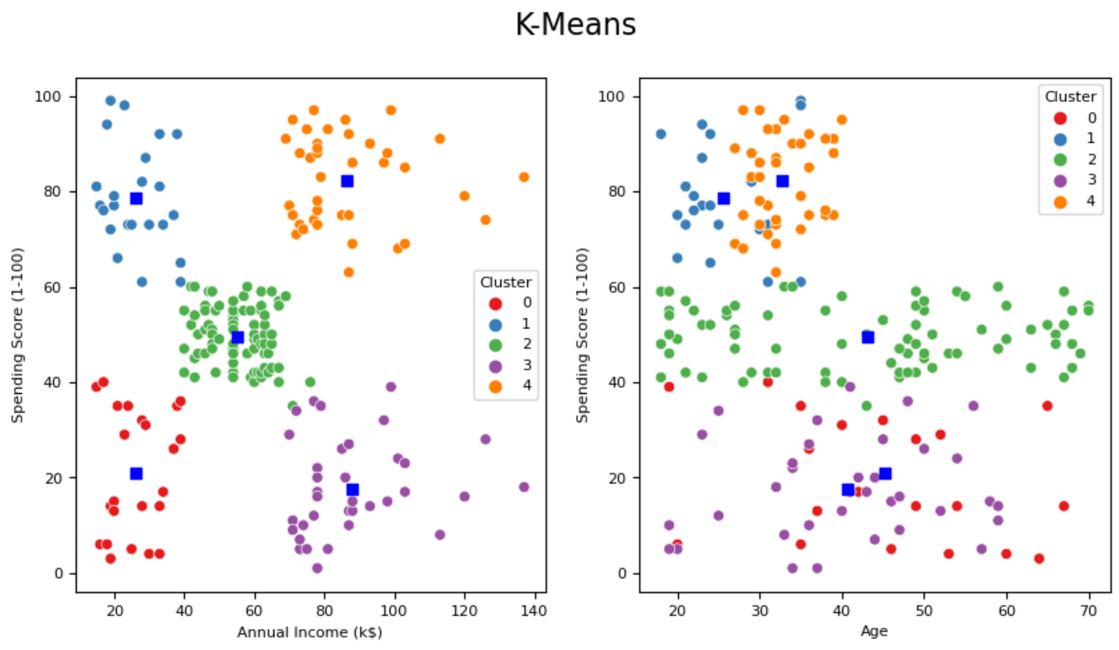
Figure 127



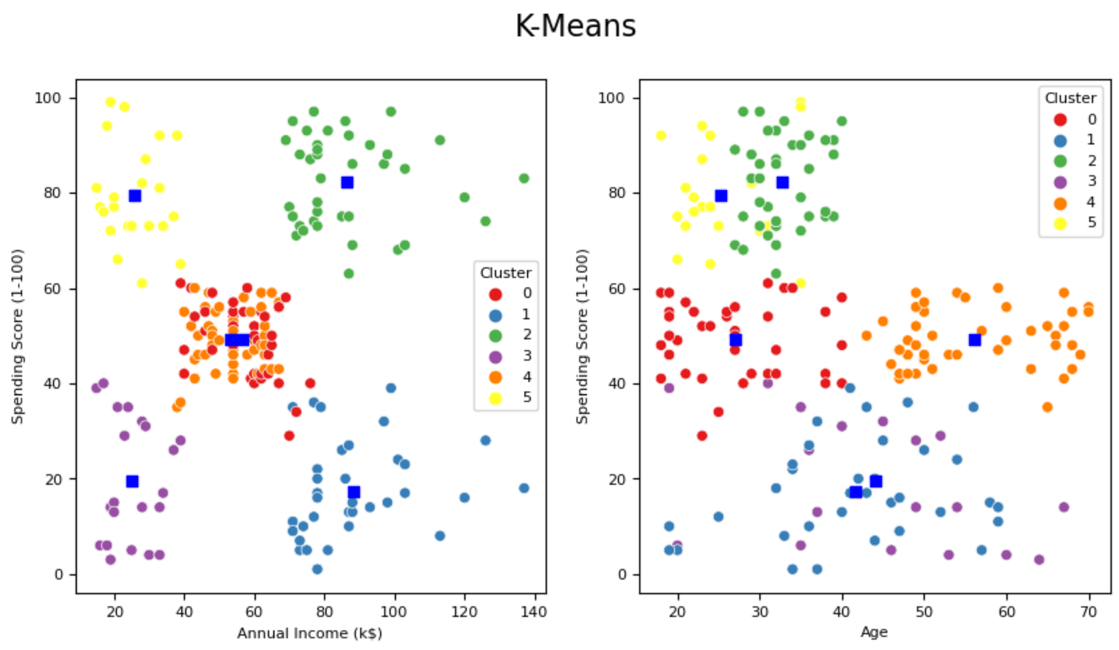
STEP 7: Evaluate

Compare Algorithms

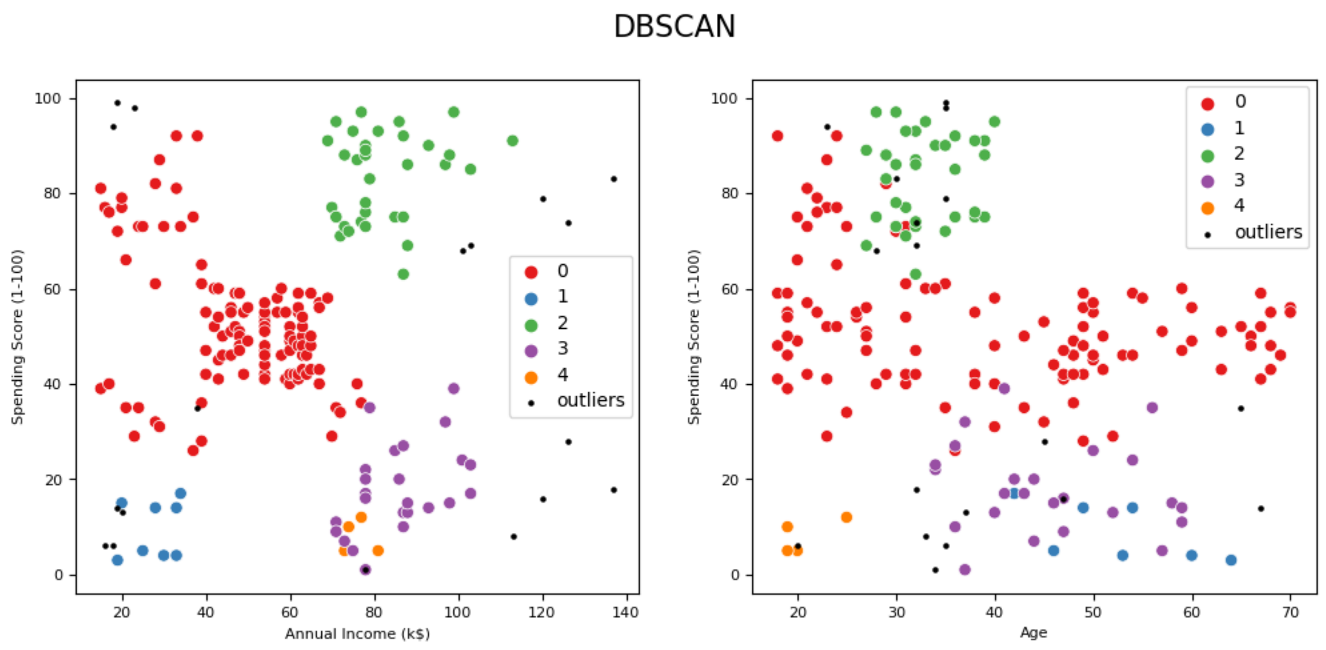
In [191]:
```python
# K-Means with K=5
fig1.suptitle('K-Means', fontsize=16)
fig1
```
Out[191]:



In [192]:
```python
# K-Means with K=6
fig11.suptitle('K-Means', fontsize=16)
fig11
```
Out[192]:



In [193]:
```python
# DBSCAN with 5 clusters AND outliers
fig2.suptitle('DBSCAN', fontsize=16)
fig2
```
Out[193]:



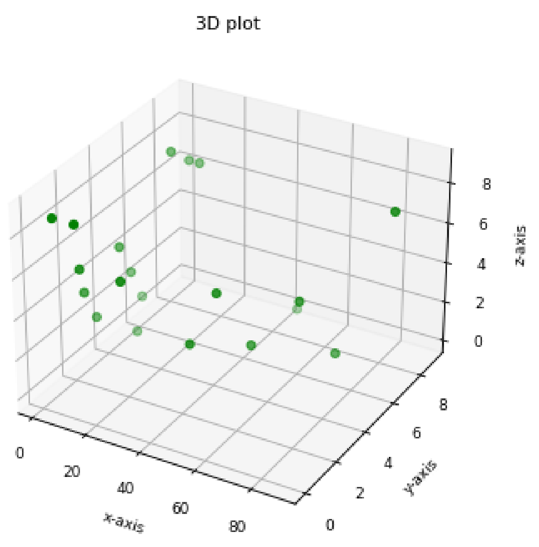
In [194]:
```python
clusters = pd.concat([KM6_clust_sizes, DBSCAN_clust_sizes],axis=1,sort=False)
clusters
```
Out[194]:

	KM_size	DBSCAN_size
Cluster		
0	38.0	112.0
1	35.0	8.0
2	39.0	34.0
3	21.0	24.0
4	45.0	4.0
5	22.0	NaN
-1	NaN	18.0

In [195]:
```python
# Print the outliers
print(DBSCAN_clustered[DBS_clustering.labels_==-1])
```
```
     Age  Annual Income (k$)  Spending Score (1-100)  Cluster
2     20                  16                       6       -1
6     35                  18                       6       -1
7     23                  18                      94       -1
10    67                  19                      14       -1
11    35                  19                      99       -1
14    37                  20                      13       -1
19    35                  23                      98       -1
40    65                  38                      35       -1
158   34                  78                       1       -1
187   28                 101                      68       -1
191   32                 103                      69       -1
192   33                 113                       8       -1
194   47                 120                      16       -1
195   35                 120                      79       -1
196   45                 126                      28       -1
197   32                 126                      74       -1
198   32                 137                      18       -1
199   30                 137                      83       -1
```

In [196]:
```python
# creating 3d plot using matplotlib
# in python

# for creating a responsive plot
%matplotlib widget

# importing required libraries
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

# creating random dataset
xs = [14, 24, 43, 47, 54, 66, 74, 89, 12, 44, 1, 2, 3, 4, 5, 9, 8, 7, 6, 5]

ys = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 6, 3, 5, 2, 4, 1, 8, 7, 0, 5]

zs = [9, 6, 3, 5, 2, 4, 1, 8, 7, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0]

# creating figure
fig=plt.figure()
ax=fig.add_subplot(111,projection='3d')

# creating the plot
plot_geeks = ax.scatter(xs, ys, zs, color='green')

# setting title and labels
ax.set_title("3D plot")
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')

# displaying the plot
plt.show()

```

Figure 119



Great Job! You are done.

Real-world application questions to consider.

Now that we have identified the outliers, now what?

- Why are they occurring?
- Where—and what—might the meaning be?

The answer could differ from organization to organization, but you can't simply ignore the data, regardless of the significance. In addition to the above questions, it is more important to know

- How outliers affect your testing efforts?
- How do I deal with the identified outliers?
- Should outliers be removed from the analysis?