# How to use k means clustering for customer segmentation

K means is one of the most widely used algorithms for clustering data and falls into the unsupervised learning group of machine learning models. It’s ideal for many forms of [customer segmentation](https://practicaldatascience.co.uk/data-science/a-quick-guide-to-customer-segmentation). Unlike supervised learning models, unsupervised learning models work by identifying previously undetected patterns in data without the benefit of a label to teach the model what to do.

For example, in a supervised learning model, such as a decision tree, you provide labelled training data telling the model the class of the target variable (i.e. spam or not spam, terrorist or non-terrorist, fraud or non-fraud). The supervised learning model finds the mathematical relationships between the features and the target variable using the labelled training data, allowing it to make predictions on unseen data based on similarities with the training data.

Unsupervised learning models are simply given a set of data and told to put the data into groups based on the similarities of the data within. For a clustering algorithm on a customer data set, this might be the placement of customers into clusters based on their recency, frequency, or monetary value, the products they purchase, or a combination of many features.

The model creates clearly define customer segments based on a number of different features and is used to assign a cluster label to each group. For example, you might call these gold, silver, and bronze if they’re based on customer values. The segments can then be used to modify the way in which the customers receive marketing or customer service levels based on the segment to which they’ve been assigned.

## How does k means clustering work?

To understand how k means clustering works, the first thing you need to understand is what “k” relates to. In k means clustering “k” is simply the number of clusters or groups created. With k means clustering you define how many clusters the model should create, and the algorithm creates them.

The “means” bit of k means refers to the underlying process by which each observation (or related set of data, such as a customer) is associated to the cluster. If you create a k means model and define that you want to have five clusters, the algorithm will initially create five random points called “centroids”.

Mathematically, the algorithm aims to minimise the within-cluster sum of square distances from the mean or SSE. The algorithm therefore assigns each observation to the nearest cluster, then updates the centroid by calculating the mean of each observation in the cluster until it is unable to gain further improvement.

#### Assumptions in k means clustering

The k means algorithm makes several assumptions about the data that are important to understand. While it often still “works” if you ignore them, you’ll see far better results if you transform the data to work within its assumptions.

* **Equal variance:** K-means assumes that variables have the same variance. The mean and standard deviation of the values should be similar. If they’re not, you’ll need to scale and standardise the variables.
* **Normal distribution:** K-means expects the distribution of each variable to be normal, not highly skewed. If it’s skewed, you’ll need to transform it first using a log transform, Box Cox, or similar.
* **Similarly sized spherical clusters:** It also expects data to form spherical clusters of roughly similar size. If your data aren’t distributed in this way, you may need a different algorithm.

## Dataset:

This Dataset is based on malls' customers. There are a total of 200 rows and 5 columns in this dataset. By using this dataset this data analysis and machine learning project is created. Mall Customer data is an interesting dataset that has hypothetical customer data. It puts you in the shoes of the owner of a supermarket. You have customer data, and on this basis of the data, you have to divide the customers into various groups.

The data includes the following features:

1. Customer ID

2. Customer Gender

3. Customer Age

4. Annual Income of the customer (in Thousand Dollars)

5. Spending score of the customer (based on customer behaviour and spending nature)

Let us proceed with the code.

#Importing the necessary libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

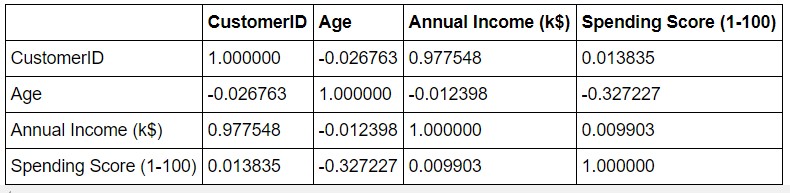
from mpl\_toolkits.mplot3d import Axes3D

%matplotlib inline

#Reading the csv file and check for correlation scores

data=pd.read\_csv('/Users/user/Desktop/dwh&BI/Mall\_Customers.csv')

data.corr()



#Distribution of Annnual Income

plt.figure(figsize=(10, 6))

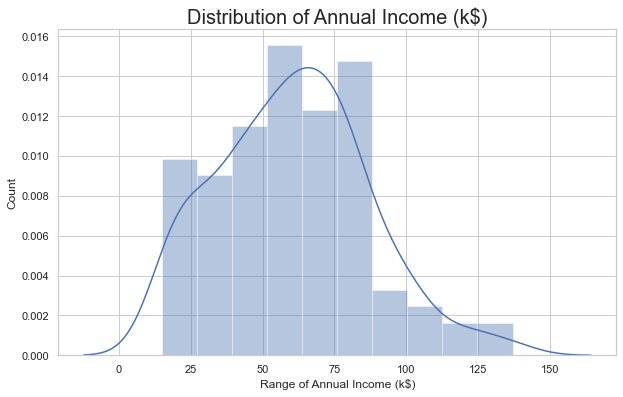
sns.set(style = 'whitegrid')

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

plt.title('Distribution of Annual Income (k$)', fontsize = 20)

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

plt.ylabel('Count')



#Distribution of age

plt.figure(figsize=(10, 6))

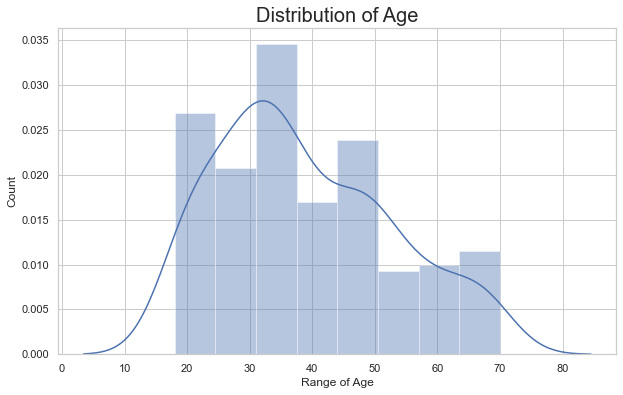
sns.set(style = 'whitegrid')

sns.distplot(data['Age'])

plt.title('Distribution of Age', fontsize = 20)

plt.xlabel('Range of Age')

plt.ylabel('Count')



#Distribution of spending score

plt.figure(figsize=(10, 6))

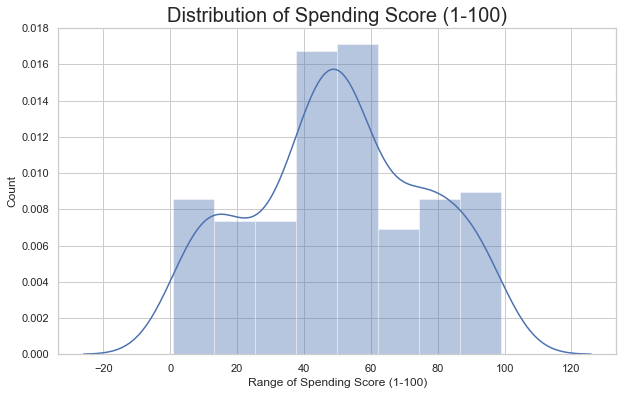
sns.set(style = 'whitegrid')

sns.distplot(data['Spending Score (1-100)'])

plt.title('Distribution of Spending Score (1-100)', fontsize = 20)

plt.xlabel('Range of Spending Score (1-100)')

plt.ylabel('Count')



# Gender analysis

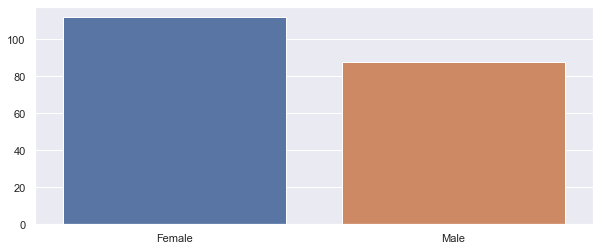
genders = data.Gender.value\_counts()

sns.set\_style("darkgrid")

plt.figure(figsize=(10,4))

sns.barplot(x=genders.index, y=genders.values)

plt.show()



#We take just the Annual Income and Spending score

df1=data[["CustomerID","Gender","Age","Annual Income (k$)","Spending Score (1-100)"]]

X=df1[["Annual Income (k$)","Spending Score (1-100)"]]

#Scatterplot of Annual Income and Spending score

plt.figure(figsize=(10,6))

sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)', data = X ,s = 60 )

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

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

plt.title('Spending Score (1-100) vs Annual Income (k$)')

plt.show()



#Annual Income vs Age and Spending Score:

x = data['Annual Income (k$)']

y = data['Age']

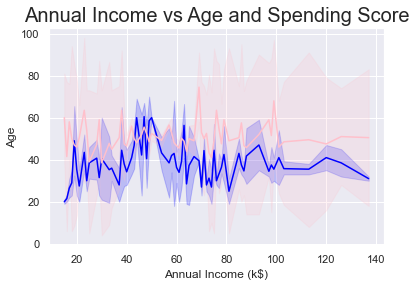
z = data['Spending Score (1-100)']

sns.lineplot(x, y, color = 'blue', palette = 'Accent\_r')

sns.lineplot(x, z, color = 'pink', palette = 'Accent\_r')

plt.title('Annual Income vs Age and Spending Score', fontsize = 20)

plt.show()



The above Plot Between Annual Income and Age represented by a blue color line and a plot between Annual Income and the Spending Score is represented by a pink color. It shows how Age and Spending Vary with Annual Income.

Now we calculate the Within Cluster Sum of Squared Errors (WSS) for different values of k. Next, we choose the k for which WSS first starts to diminish. This value of K gives us the best number of clusters to make from the raw data.

#Importing KMeans from sklearn

from sklearn.cluster import KMeans

wcss=[]

for i in range(1,11):

km=KMeans(n\_clusters=i)

km.fit(X)

wcss.append(km.inertia\_)

#The elbow curve

plt.figure(figsize=(12,6))

plt.plot(range(1,11),wcss)

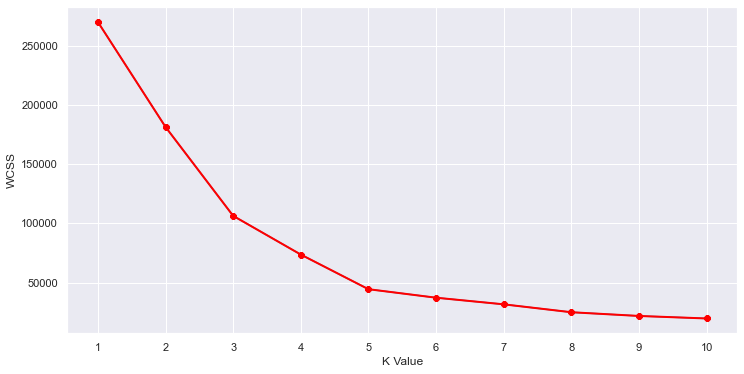
plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="8")

plt.xlabel("K Value")

plt.xticks(np.arange(1,11,1))

plt.ylabel("WCSS")

plt.show()



This is known as the elbow graph, the x-axis being the number of clusters, the number of clusters is taken at the elbow joint point. This point is the point where making clusters is most relevant as here the value of WCSS suddenly stops decreasing. Here in the graph, after 5 the drop is minimal, so we take 5 to be the number of clusters.

#Taking 5 clusters

km1=KMeans(n\_clusters=5)

#Fitting the input data

km1.fit(X)

#predicting the labels of the input data

y=km1.predict(X)

#adding the labels to a column named label

df1["label"] = y

#The new dataframe with the clustering done

df1.head()

#Scatterplot of the clusters

plt.figure(figsize=(10,6))

sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)',hue="label",

palette=['green','orange','brown','dodgerblue','red'], legend='full',data = df1 ,s = 60 )

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

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

plt.title('Spending Score (1-100) vs Annual Income (k$)')

plt.show()



## k-Means Clustering on the basis of 3D data

Now, we shall be working on 3 types of data. Apart from the spending score and annual income of customers, we shall also take in the age of the customers.

#Taking the features

df=data[["CustomerID","Gender","Age","Annual Income (k$)","Spending Score (1-100)"]]

X2=df[["Age","Annual Income (k$)","Spending Score (1-100)"]]

#Now we calculate the Within Cluster Sum of Squared Errors (WSS) for different values of k.

wcss = []

for k in range(1,11):

kmeans = KMeans(n\_clusters=k, init="k-means++")

kmeans.fit(X2)

wcss.append(kmeans.inertia\_)

plt.figure(figsize=(12,6))

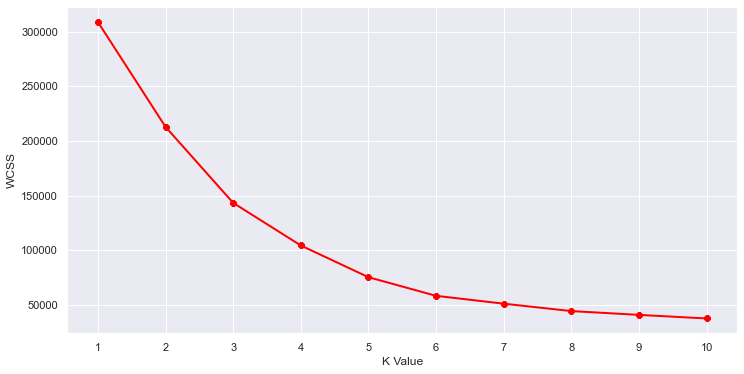
plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="8")

plt.xlabel("K Value")

plt.xticks(np.arange(1,11,1))

plt.ylabel("WCSS")

plt.show()



#We choose the k for which WSS starts to diminish

#We choose the k for which WSS starts to diminish

km2 = KMeans(n\_clusters=5)

y2 = km2.fit\_predict(X2)

df["label"] = y2

#The data with labels

df.head()

#3D Plot as we did the clustering on the basis of 3 input features

fig = plt.figure(figsize=(20,10))

ax = fig.add\_subplot(111, projection='3d')

ax.scatter(df.Age[df.label == 0], df["Annual Income (k$)"][df.label == 0], df["Spending Score (1-100)"][df.label == 0], c='purple', s=60)

ax.scatter(df.Age[df.label == 1], df["Annual Income (k$)"][df.label == 1], df["Spending Score (1-100)"][df.label == 1], c='red', s=60)

ax.scatter(df.Age[df.label == 2], df["Annual Income (k$)"][df.label == 2], df["Spending Score (1-100)"][df.label == 2], c='blue', s=60)

ax.scatter(df.Age[df.label == 3], df["Annual Income (k$)"][df.label == 3], df["Spending Score (1-100)"][df.label == 3], c='green', s=60)

ax.scatter(df.Age[df.label == 4], df["Annual Income (k$)"][df.label == 4], df["Spending Score (1-100)"][df.label == 4], c='yellow', s=60)

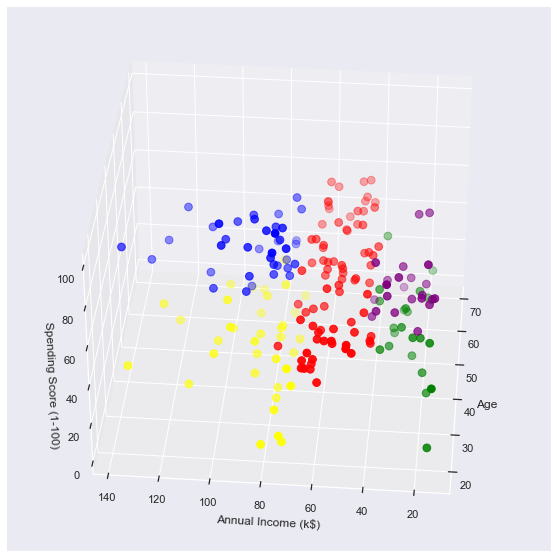
ax.view\_init(35, 185)

plt.xlabel("Age")

plt.ylabel("Annual Income (k$)")

ax.set\_zlabel('Spending Score (1-100)')

plt.show()



What we get is a 3D plot. Now, if we want to know the customer IDs, we can do that too.

cust1=df[df["label"]==1]

print('Number of customer in 1st group=', len(cust1))

print('They are -', cust1["CustomerID"].values)

print("--------------------------------------------")

cust2=df[df["label"]==2]

print('Number of customer in 2nd group=', len(cust2))

print('They are -', cust2["CustomerID"].values)

print("--------------------------------------------")

cust3=df[df["label"]==0]

print('Number of customer in 3rd group=', len(cust3))

print('They are -', cust3["CustomerID"].values)

print("--------------------------------------------")

cust4=df[df["label"]==3]

print('Number of customer in 4th group=', len(cust4))

print('They are -', cust4["CustomerID"].values)

print("--------------------------------------------")

cust5=df[df["label"]==4]

print('Number of customer in 5th group=', len(cust5))

print('They are -', cust5["CustomerID"].values)

print("--------------------------------------------")

Number of customer in 1st group= 79

They are - [ 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 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 127 143]

--------------------------------------------

Number of customer in 2nd group= 39

They are - [124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 154 156 158 160 162 164 166 168 170 172 174 176 178 180 182 184 186 188 190 192 194 196 198 200]

--------------------------------------------

Number of customer in 3rd group= 23

They are - [ 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46]

--------------------------------------------

Number of customer in 4th group= 23

They are - [ 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45]

--------------------------------------------

Number of customer in 5th group= 36

They are - [125 129 131 133 135 137 139 141 145 147 149 151 153 155 157 159 161 163 165 167 169 171 173 175 177 179 181 183 185 187 189 191 193 195 197 199]

## Hierarchical Clustering

#Hierarchical agglomerative clustering (HAC)

import scipy.cluster.hierarchy as sch

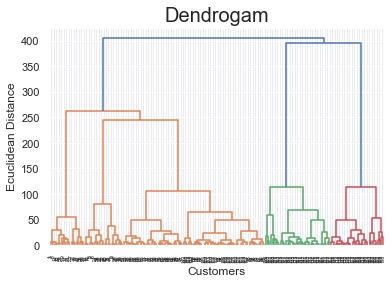
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))

plt.title('Dendrogam', fontsize = 20)

plt.xlabel('Customers')

plt.ylabel('Ecuclidean Distance')

plt.show()



from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n\_clusters = 5, affinity = 'euclidean', linkage = 'ward')

y\_hc = hc.fit\_predict(X)

df2=data[["CustomerID","Gender","Age","Annual Income (k$)","Spending Score (1-100)"]]

df2["label"] = y\_hc

df2

#Scatterplot of the clusters

plt.figure(figsize=(10,6))

sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)',hue="label",

palette=['green','orange','brown','dodgerblue','red'], legend='full',data = df2 ,s = 60 )

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

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

plt.title('Spending Score (1-100) vs Annual Income (k$)')

plt.show()



cust6=df2[df2["label"]==1]

print('Number of customer in 1st group=', len(cust6))

print('They are -', cust1["CustomerID"].values)

print("--------------------------------------------")

cust7=df2[df2["label"]==2]

print('Number of customer in 2nd group=', len(cust7))

print('They are -', cust2["CustomerID"].values)

print("--------------------------------------------")

cust8=df2[df2["label"]==0]

print('Number of customer in 3rd group=', len(cust8))

print('They are -', cust3["CustomerID"].values)

print("--------------------------------------------")

cust9=df2[df2["label"]==3]

print('Number of customer in 4th group=', len(cust9))

print('They are -', cust4["CustomerID"].values)

print("--------------------------------------------")

cust10=df2[df2["label"]==4]

print('Number of customer in 5th group=', len(cust10))

print('They are -', cust5["CustomerID"].values)

print("--------------------------------------------")

Number of customer in 1st group= 85

They are - [ 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 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 127 143]

--------------------------------------------

Number of customer in 2nd group= 39

They are - [124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 154 156 158 160 162 164 166 168 170 172 174 176 178 180 182 184 186 188 190 192 194 196 198 200]

--------------------------------------------

Number of customer in 3rd group= 32

They are - [129 131 135 137 139 141 145 149 151 153 155 157 159 163 165 167 169 171 173 175 177 179 181 183 185 187 189 191 193 195 197 199]

--------------------------------------------

Number of customer in 4th group= 21

They are - [ 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42]

--------------------------------------------

Number of customer in 5th group= 23

They are - [ 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45]

--------------------------------------------