Unsupervised Learning

ScholarHat

Last lecture reminder

We learned about:

- Random Forest - Introduction
- Random Forest vs Single Decision Tree
- Random Forest - Parameters
- Bootstrapping in Random Forest
- Out-Of-Bag (OOB) Error
- Random Forest - Python Example

Unsupervised Learning - Introduction

Until now we discussed only on supervised learning algorithms. What considering it as supervised algorithms is the fact that for each set of feature we also have the corresponding label. According to that known label we could train our model and test it's accuracy.

Unsupervised learning → Unsupervised learning is a type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision.
In contrast to supervised learning that usually uses human-labeled data, unsupervised learning, doesn't have labeled data so it can't learn from it and test itself accordingly.
In unsupervised learning the model have only the features data without the label so it will try to find specific patterns according to those features.

In addition, unlike in supervised learning which it was very easy for us to test our model accuracy. In unsupervised learning it will be much harder to create high accuracy models.

Unsupervised Learning - Introduction

Just like with supervised learning that we separated the different models to 2 tasks - classification problems and regression problems, the same thing applying to unsupervised learning models as well.

Unsupervised learning models are separated to three main tasks:

- **Clustering** → Clustering is a data mining technique for grouping unlabeled data based on their similarities or differences.
- **Association** → Association is another type of unsupervised learning method that uses different rules to find relationships between variables in a given dataset.
- **Dimensionality reduction** → Dimensionality reduction is a learning technique used when the number of features (or dimensions) in a given dataset is too high. It reduces the number of data inputs to a manageable size while also preserving the data integrity. Often, this technique is used in the preprocessing data stage.

Supervised vs Unsupervised Learning

Main differences between supervised learning and unsupervised learning:

- **Labeled data** → The main distinction between the two approaches is the use of labeled datasets. To put it simply, supervised learning uses labeled input and output data, while an unsupervised learning algorithm does not.
  In supervised learning, the algorithm “learns” from the training dataset by iteratively making predictions on the data and adjusting for the correct answer.
  Unsupervised learning models, in contrast, work on their own to discover the inherent structure of unlabeled data.

- **Model goals** → In supervised learning, the goal is to predict outcomes for new data. You know up front the type of results to expect. With an unsupervised learning algorithm, the goal is to get insights from large volumes of new data. The machine learning itself determines what is different or interesting from the dataset.

Supervised vs Unsupervised Learning

Main differences between supervised learning and unsupervised learning:

- **Complexity** → Supervised learning is a simple method for machine learning, typically calculated through the use of basic programs like R or Python.
  In unsupervised learning, you need powerful tools for working with large amounts of unclassified data. Unsupervised learning models are computationally complex because they need a large training set to produce intended outcomes.

- **Drawbacks** → Supervised learning models can be time-consuming to train, and the labels for input and output variables require expertise. Meanwhile, unsupervised learning methods can have wildly inaccurate results unless you have human intervention to validate the output variables.

K-Mean Clustering - Introduction

K-Mean Clustering → K-Means clustering is a type of unsupervised machine learning algorithm used for classification or to make inferences from the dataset in order to organize the data into distinct groups (clusters) based on their features.

Unlike in classification problems, in clustering problems the algorithm goal is not to identity the cluster label but to organize the observation into different clusters (groups). The algorithm itself can't identify what is the meaning of those groups and its the task of the data science to try and understand what each group represent.

This is why the algorithm cluster names will be group_1, group_2, … with no actual group labels.

K-Mean Clustering - Introduction

K-Mean Clustering algorithm steps:

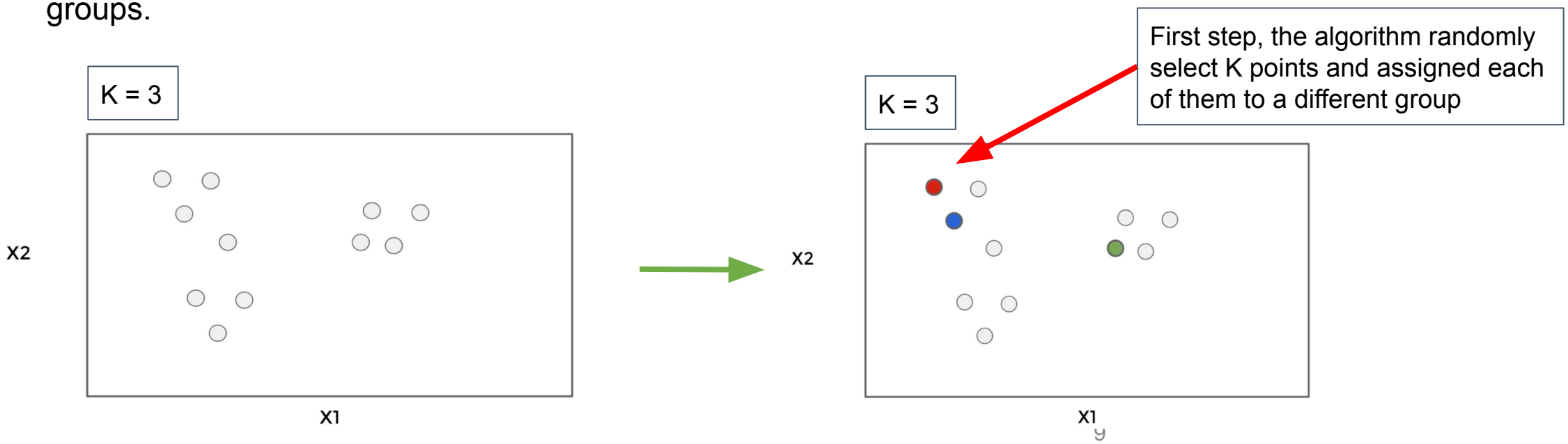
1. **Initialization** → Firstly, 'K' centroids are randomly initialized. K is predetermined by the user (and thus a crucial decision to make). This K is the number of clusters that the user wants to form from the given dataset.
2. **Assignment** → Each data point is then assigned to the closest centroid based upon some distance measure, typically Euclidean distance.
3. **Centroid update** → Once all data points have been assigned to the existing centroids, a new centroid of each cluster is calculated. This is typically calculated as the mean of all the points assigned to the cluster.
4. **Reassignment** → Iterate on all data points and calculate for each data point it's distance to each cluster centroid. In case re-assign is need it's been done at this step.
5. **Reprocessing** → Steps 2 - 4 are repeated until the assignments of data points to clusters no longer change or the algorithm reaches a predetermined number of iterations.

K-Mean Clustering - Introduction

Let’s take a simple example to see how K-Mean Clustering is working:

Let’s say that we have a dataset with 2 different features (X1 & X2) and we want to use K-Mean Clustering to order the data into a different clusters (groups).

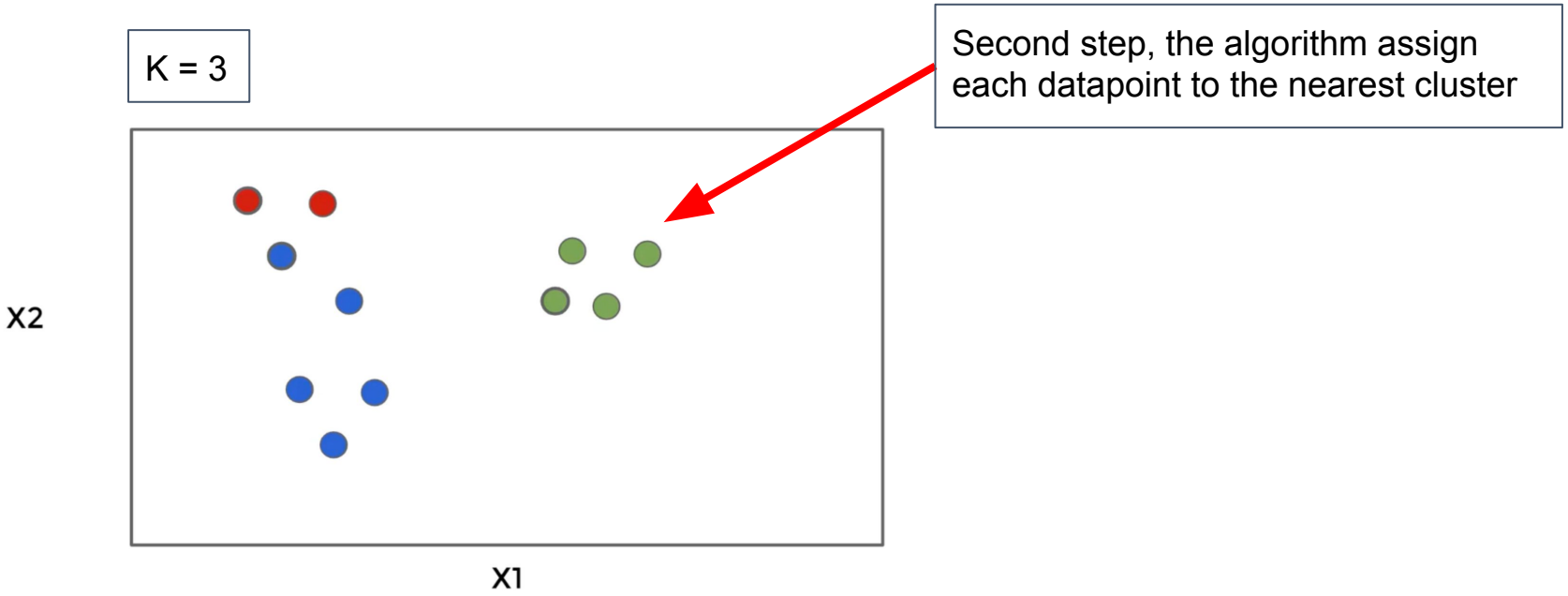
For the example we will use K=3 meaning we want the algorithm to order the observations into 3 different groups.



K-Mean Clustering - Introduction

In the second step, the algorithm will assign each datapoint to the closest distance existing cluster.

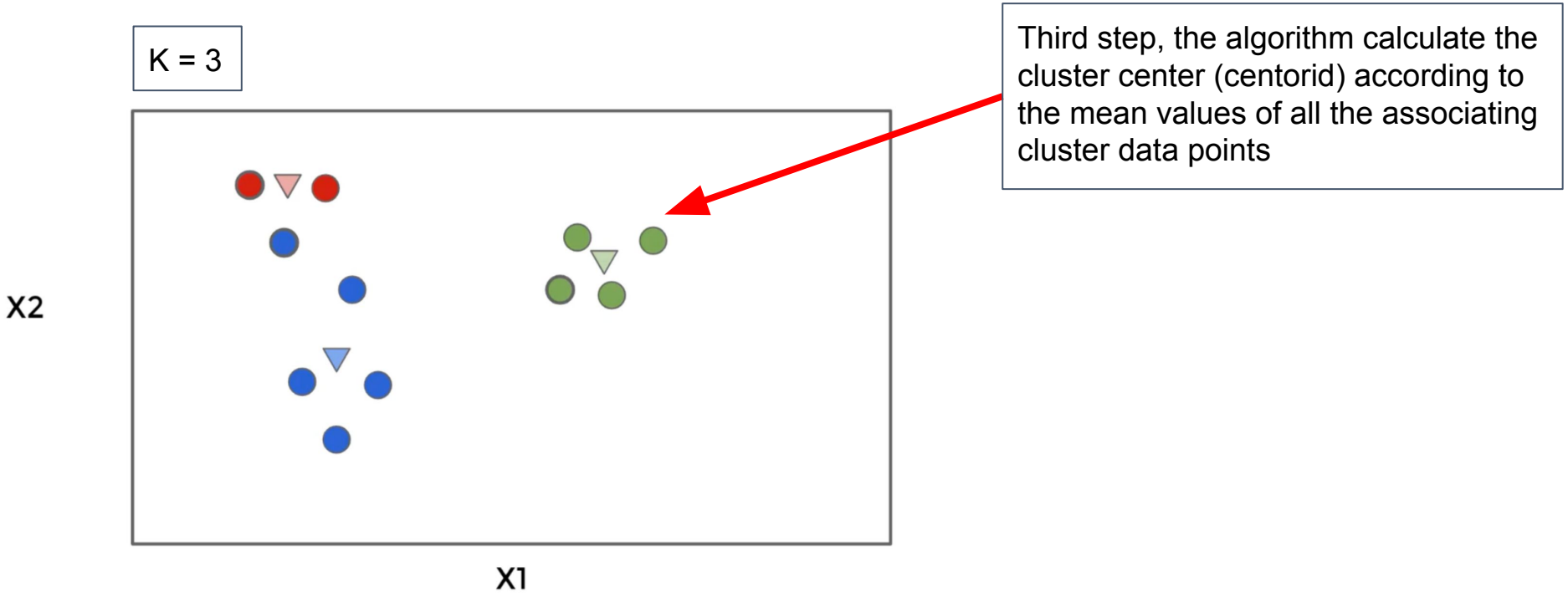
This calculation been done by representing each datapoint as a vector in space and calculate this vector distance from the center of the cluster (centroids).



K-Mean Clustering - Introduction

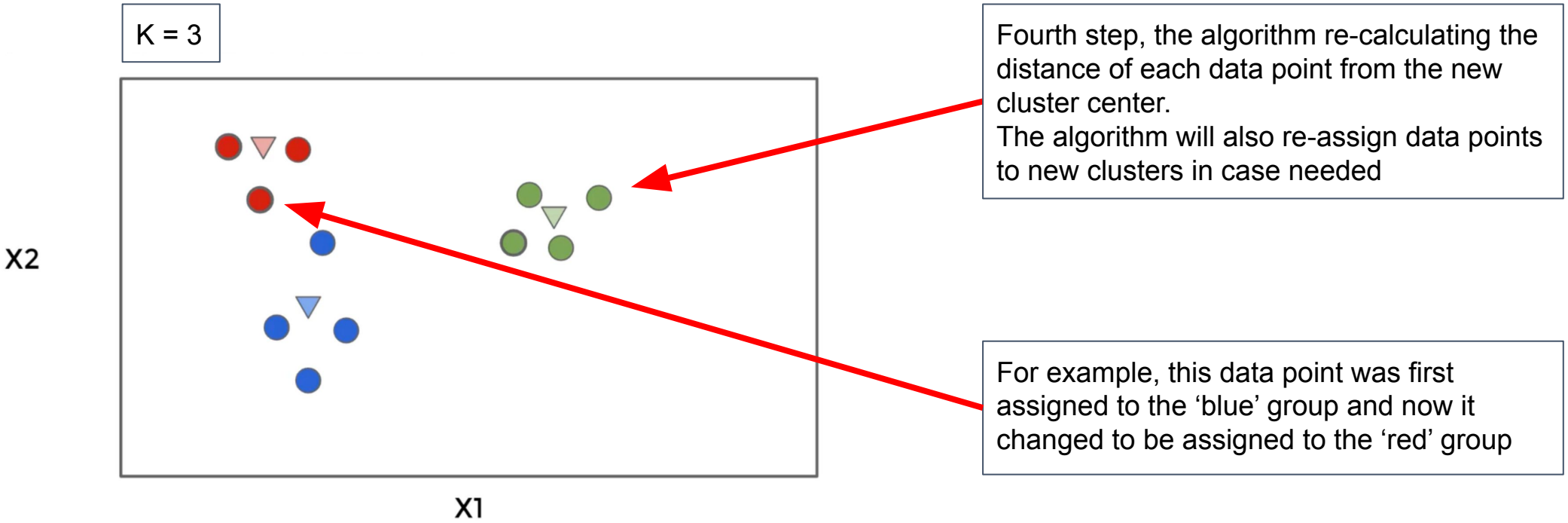
In the third step, the algorithm will calculate the new center of each cluster (centroid).

This calculation been done by calculating the mean value of each vector in the cluster.



K-Mean Clustering - Introduction

In the fourth step, the algorithm iterate on all data points and calculate their distance from the new cluster center. The algorithm reassigned all data points to the nearest cluster center. This action can cause changing the cluster association in some of the data points.

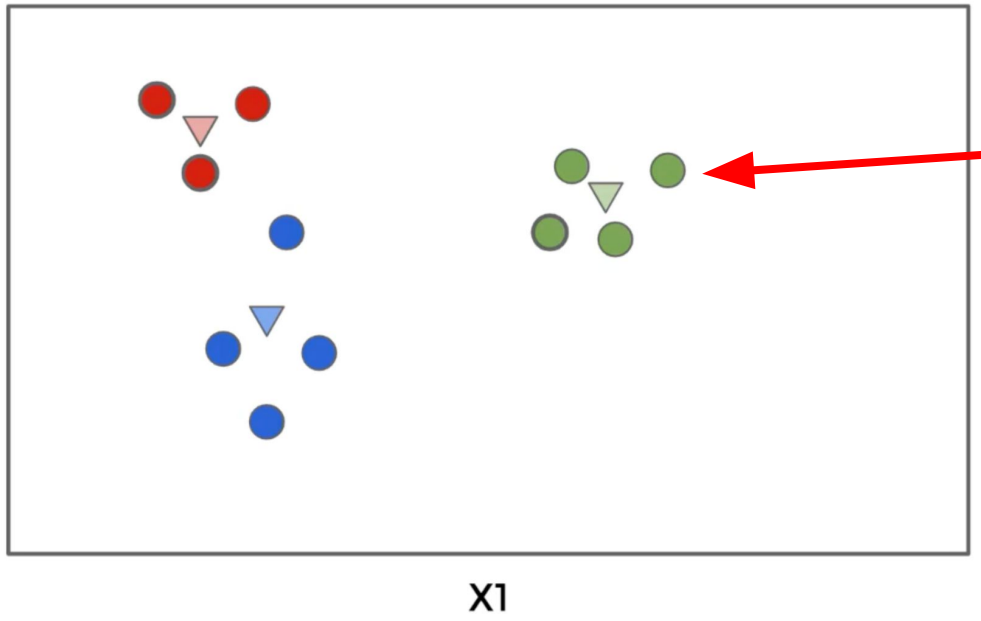


K-Mean Clustering - Introduction

In the final step, the keep repeating steps 2 - 4 meaning it keep calculating the new centroid of each cluster and keep re-assign data points to the nearest cluster centroid.

This step will continue until there are no more changes in the cluster association or the algorithm reached it's iteration limit.

K = 3



Final result, each data point is been associated to a different cluster and there are 3 clusters in total as determined by the K value

K-Mean Clustering - Python Example

For this example we will use the **‘bank-full.csv’** file. This file is a very large dataset containing data about bank customers marketing phone calls.

Some main features in the dataset:

- ‘job’ → The customer job category
- ‘marital’ → The customer marriage status
- ‘education’
- ‘housing’ → Does the customer own a house (yes / no)
- ‘loan’ → Does the customer has a lown (yes / no)
- Call details → like contact (by telephone or cellphone), duration of the call, campaign id, pdays (the amount of days that passed from the previous customer call).

K-Mean Clustering - Python Example

Before we start to train our K-Mean model let's first explore the given dataset. Also remember that the model will provide the clustering allocation but it's our task to understand what each group actually means. This is why it's important in unsupervised learning to analyze the dataset before the model training.

First let's take a look at the complete dataset:

```
In [48]: df = pd.read_csv('/Users/ben.meir/Downloads/bank-full.csv')
         print(f'dataset rows: {len(df)}')
         df.head()

         dataset rows: 41188

Out[48]:
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cc
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	1.1	

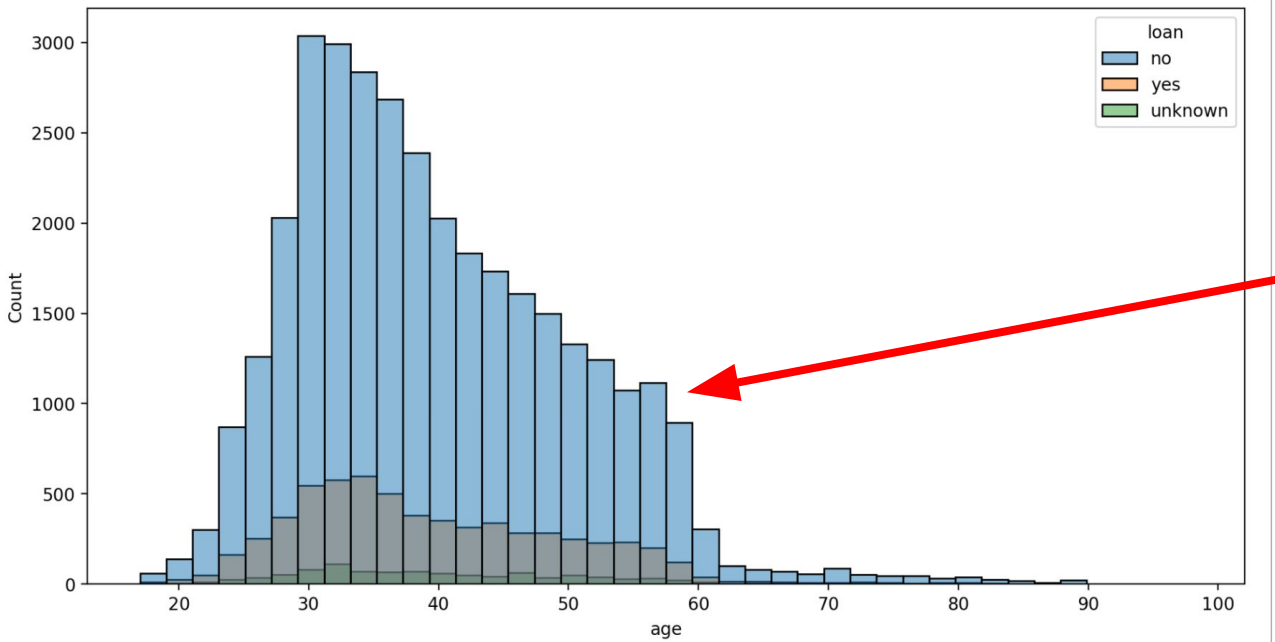
5 rows × 21 columns

The dataset contain almost 42,000 rows which is considering a large dataset

K-Mean Clustering - Python Example

Now, let's explore some of the features in the dataset:

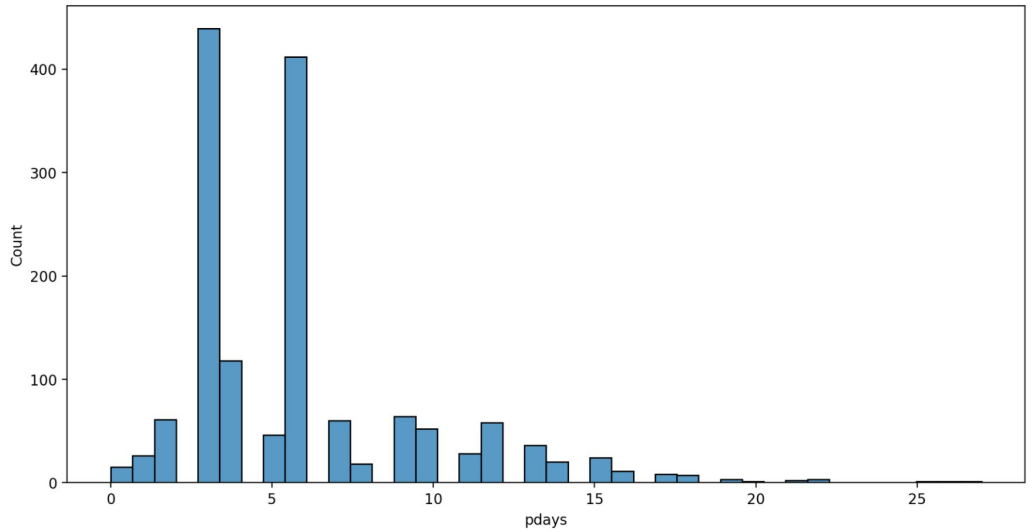
```
In [49]: plt.figure(figsize=(12,6), dpi=200)
         sns.histplot(data=df, x='age', bins=40, hue='loan')
         plt.show()
```



The distribution of customer ages regarding customers with or without loans

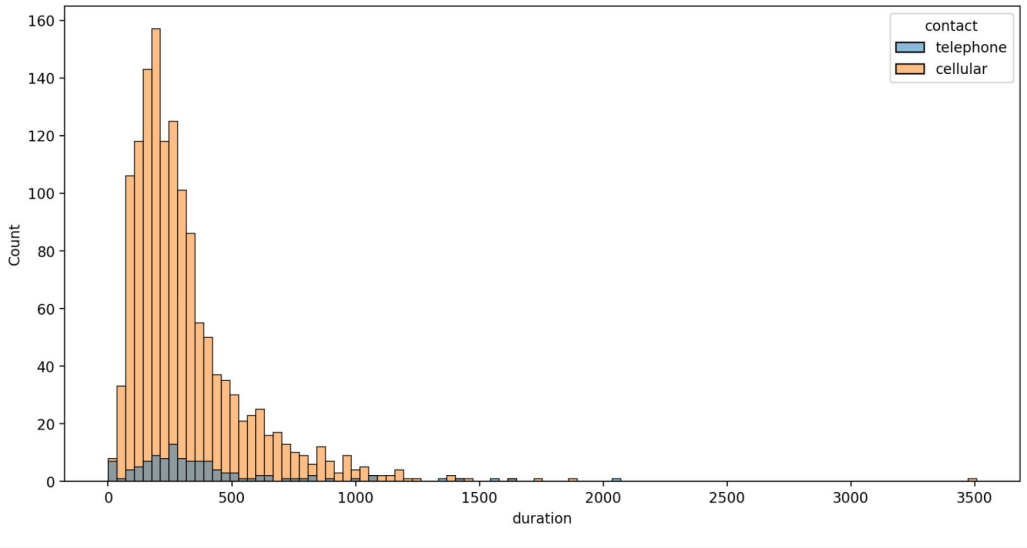
K-Mean Clustering - Python Example

```
In [50]: data_df = df[df['pdays'] != 999]
plt.figure(figsize=(12,6), dpi=200)
sns.histplot(data=data_df, x='pdays', bins=40)
plt.show()
```



The distribution of the amount of days passed before the previous marketing call without those with 999 (meaning there was no previous marketing call to those customers)

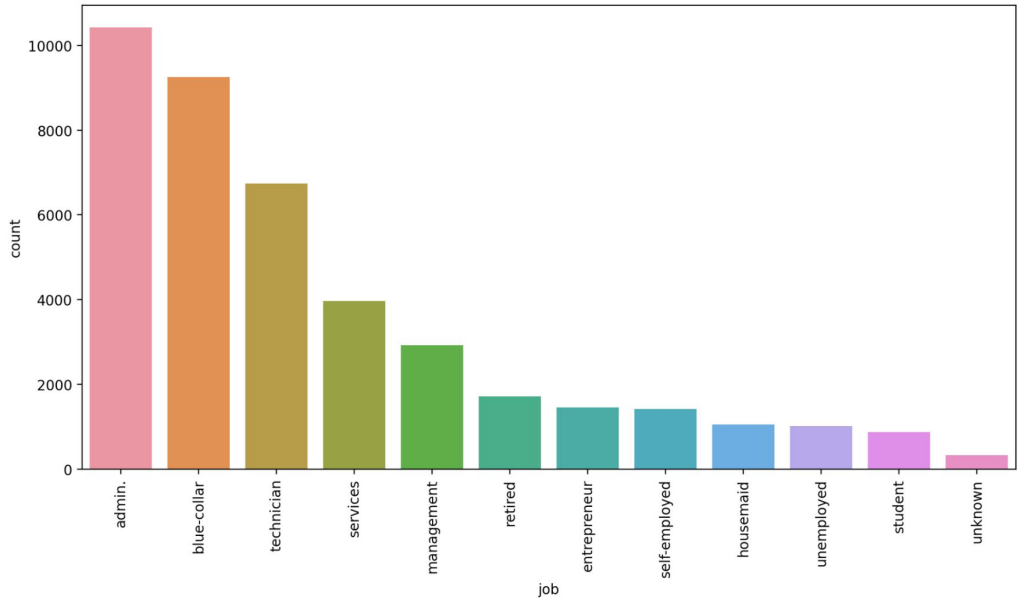
```
In [51]: plt.figure(figsize=(12,6), dpi=200)
sns.histplot(data=data_df, x='duration', bins=100, hue='contact')
plt.show()
```



The distribution of the call duration regarding the contact type (telephone or celular)

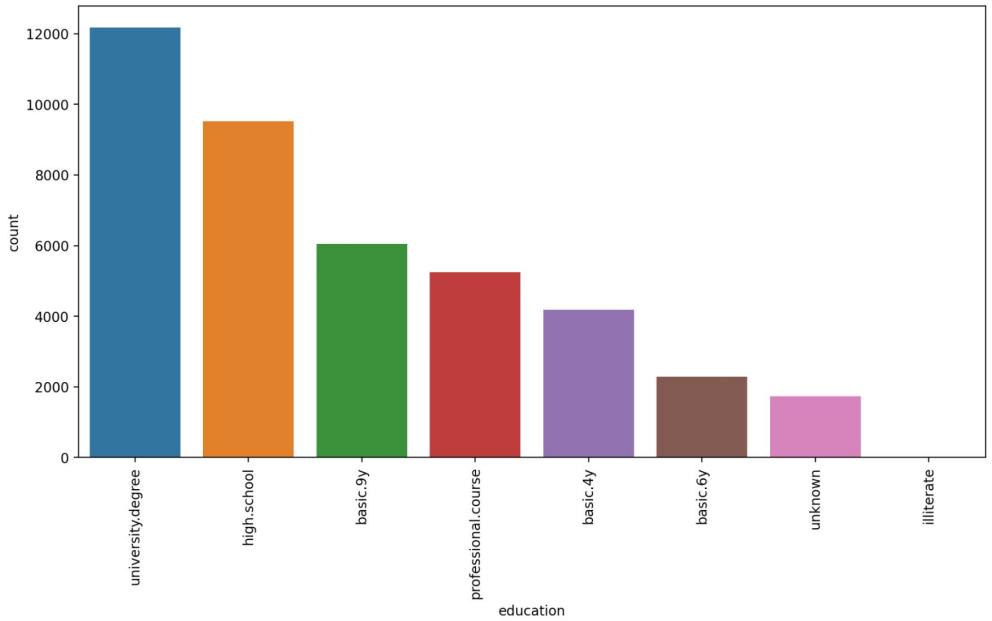
K-Mean Clustering - Python Example

```
In [52]: order_jobs_df = df['job'].value_counts().index

         plt.figure(figsize=(12,6), dpi=200)
         sns.countplot(data=df, x='job', order=order_jobs_df)
         plt.xticks(rotation=90)
         plt.show()
```



Customers count regarding the customer job category. The chart is sorted from highest count to the lowest

```
In [53]: order_jobs_df = df['education'].value_counts().index

         plt.figure(figsize=(12,6), dpi=200)
         sns.countplot(data=df, x='education', order=order_jobs_df)
         plt.xticks(rotation=90)
         plt.show()
```



Customers count regarding the customer education category. The chart is sorted from highest count to the lowest

K-Mean Clustering - Python Example

After analyzing and examine the given dataset we can start train our K-Mean Clustering model.

When training a K-Mean Clustering model its very important to handle and prepare the data first.

- **Handle categorical data** → We don’t want categorical data so we will use the pd.get_dummies() method to transform those categorical values into binary values (True / False values).
- **Data Scaling** → Given that the K-Means Clustering model computes the distance of each data point as a vector in space, it becomes vital to scale all values prior to training. This ensures that features with larger values don't disproportionately influence the distance calculation compared to features with smaller values.

Note: In unsupervised learning we don’t have labeled data, meaning we can’t perform train / test split.

So this means that we can train our model on the entire dataset without worried about test set.

K-Mean Clustering - Python Example

Let's now preparing our dataset and train the K-Mean Clustering model.

For the example we will choose the K value to be 2 but later we will see methods to find the optimal K value for the model.

```
In [54]: from sklearn.preprocessing import StandardScaler
         from sklearn.cluster import KMeans

         df = pd.read_csv('/Users/ben.meir/Downloads/bank-full.csv')
         X = pd.get_dummies(df)

         scaler = StandardScaler()
         scaled_X = scaler.fit_transform(X)

         model = KMeans(n_clusters=2)

         cluster_labels = model.fit_predict(scaled_X)
         cluster_labels

Out[54]: array([0, 0, 0, ..., 1, 1, 1], dtype=int32)
```

We handle the categorial features with pd.get_dummies() method

We apply feature scaling on the entire dataset. We can also apply fit_transform() on the entire dataset because we don't use train / test split.

We train our model with K = 2 and we use fit_predict() which it similar to fit_transfom(). Again, we can use it because we don't use train / test split.

The model prediction is the group (0 or 1) that each data point is associated to. Remember that we need to understand ourselves what every group represent.

K-Mean Clustering - Optimal K Value

Until now we didn't choose an optimal K value but just decided randomly what is the K value that we want. In order to find an optimal K value we need a way to evaluate our model performance and find the K value that provide the best model performance.

The problem is that in unsupervised learning we don't have labels so we can't compare the prediction results to the actual results in order to evaluate the model.

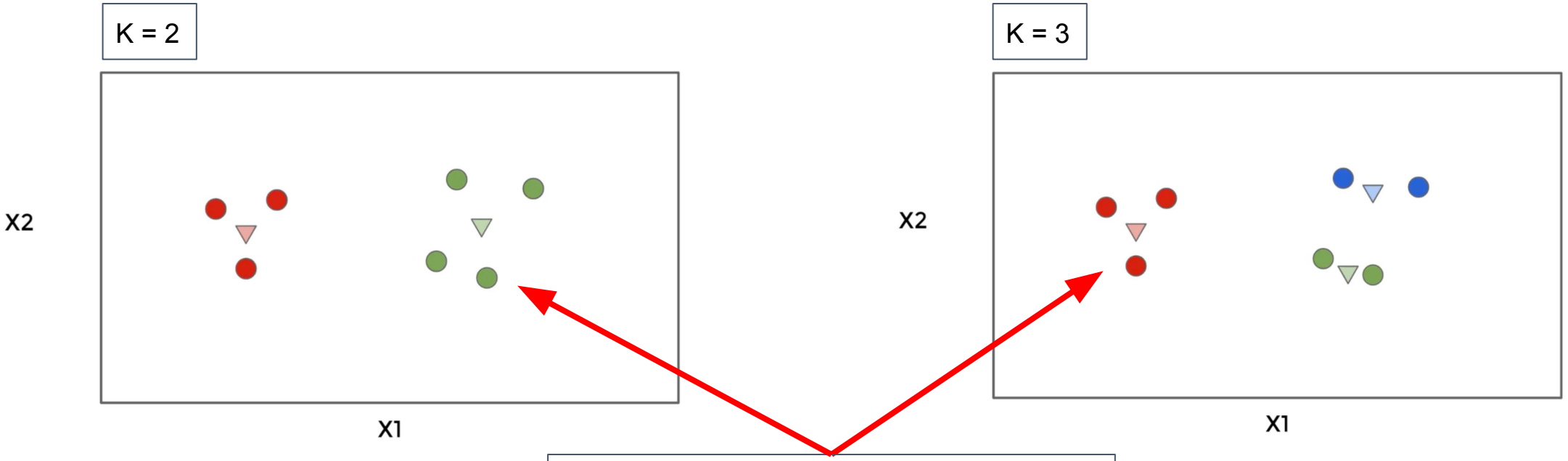
However, what we can do is to calculate for each K value the corresponding **SSD**.

SSD (Sum of Squared Distances) → SSD is used as a measure of the total within-cluster variation. The SSD is calculated by summing the squared distances of each data point to its closest cluster centroid. The goal of K-Means Clustering is to minimize the SSD, which essentially means reducing the variance within each cluster. This reflects that the data points in the same cluster are as similar as possible.

K-Mean Clustering - Optimal K Value

Let’s see an example of how SSD calculation is working.

Let’s say we have the following dataset and we want to find the optimal K value.

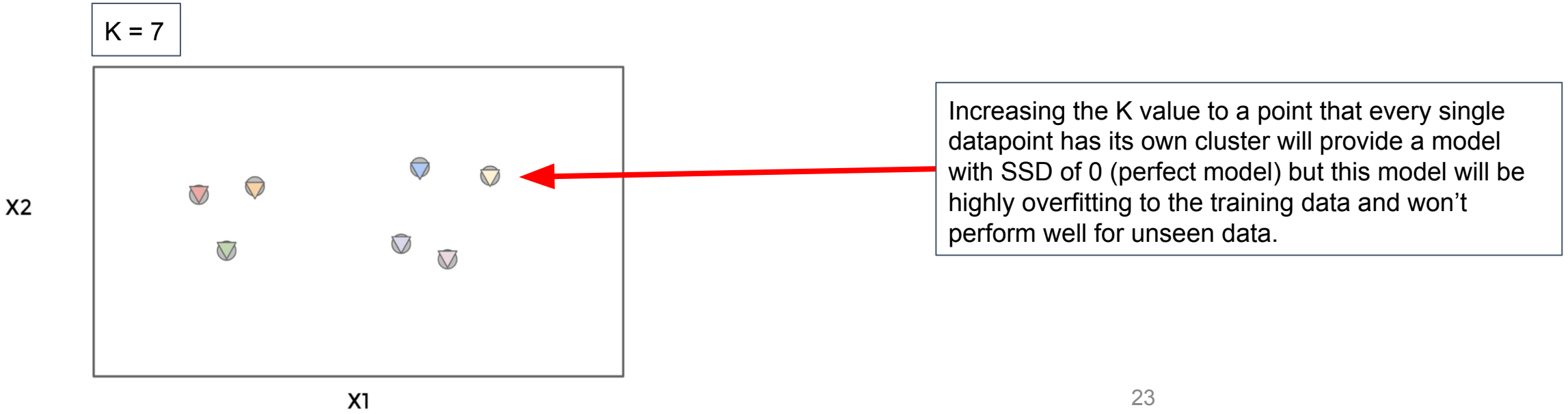


For each K value we run the K-Mean Clustering algorithm and calculate the SSD value. This value represent the ‘accuracy’ of the model clustering.
We want SSD as low as possible.

K-Mean Clustering - Optimal K Value

Every time we increase the K value we will see a drop in the model SSD values because by increasing the number of clusters, each data point might be closer to its designated centroid. Therefore, the overall SSD (which is the sum of these distances) will intuitively decrease.

If we will keep increasing the K until it will match the number of observation in the dataset we will model with perfect SSD with 0 value. However, this will cause overfitting and won't perform well for unseen data.



Optimal K Value - Python Example

So in order to find the optimal K value we can use some kind of elbow method and see where the drop in SSD is not justifying the increasing of the K value.

Let's use the previous example and generate an elbow method to find the optimal K value:

```
In [66]: from sklearn.preprocessing import StandardScaler
         from sklearn.cluster import KMeans

         df = pd.read_csv('/Users/ben.meir/Downloads/bank-full.csv')
         X = pd.get_dummies(df)

         scaler = StandardScaler()
         scaled_X = scaler.fit_transform(X)
```

Preparing the data with pd.get_dummies() and data scaling.

```
In [*]: ssd = []

        for k in range(2,10):
            model = KMeans(n_clusters=k)
            model.fit(scaled_X)

            ssd.append(model.inertia_)
```

We run a for loop that will generate a K-Mean Clustering model with the relevant K value. For each K value we save the corresponding SSD value whitch in Scikit-Learn provided by the model.inertia_ property.

Optimal K Value - Python Example

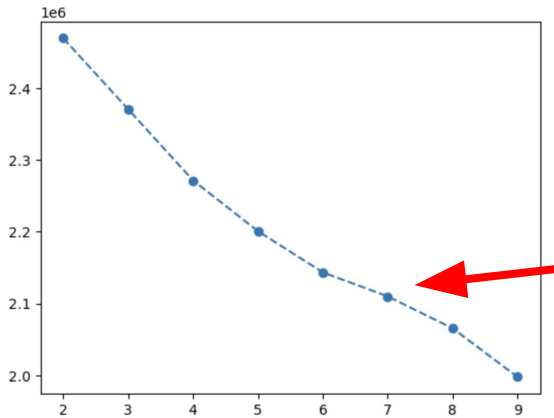
We can take a look at the ssd array to see the results:

```
In [73]: ssd

Out[73]: [2469792.361662753,
          2370786.469991024,
          2271502.7381159575,
          2200694.0788070476,
          2143252.542312437,
          2109764.2978770975,
          2065183.97599472,
          1997403.8477992783]
```

We can see that for each K value the SSD value decreased. What we also can see is that some K values provided sharper SSD decrease then others.

We can also generate this as a line plot and see from which K value the decrease started to reduce:

```
In [75]: plt.plot(range(2,10), ssd, 'o--')
         plt.show()
```



We can see from the graph that starting from K = 6 the decrease is SSD is not as sharp as with the previous K values. This can indicate that the optimal K value should be 6.

Class Exercise - K-Mean Clustering

Instructions:

Use the **'wine_data.csv'** file that contain data about different wines. This dataset contain only features as as Alcohol, Color intensity, Flavonoids, Total phenols and more.

Out[83]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od3
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	
...	...	...	...	...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.06	7.70	0.64	
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.41	7.30	0.70	
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35	10.20	0.59	
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46	9.30	0.60	
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35	9.20	0.61	

178 rows × 13 columns

Class Exercise - K-Mean Clustering

Instructions:

Your mission is to implement the following:

- Run a simple K-Mean Clustering model with K = 3.

  Remember to handle the data properly before start training the model.
- Print the cluster center values with cluster_centers_ method and print the prediction values.
- Add to the existing dataframe the corresponding cluster prediction values
- Run elbow method for K values between 2 and 8 and find what is the optional K value.
- Generate a line plot of the elbow method to support your answer.

Class Exercise Solution - K-Mean Clustering