18.4.1

## Elbow Curve

**The** trial-and-error method seemed to work to an extent, but the two of you wonder what happens when data gets more complex, such as in the cryptocurrency dataset.

There is a method that will help you make a more educated guess on the number of clusters called the elbow curve.

An easy method for determining the best number for K is the elbow curve. Elbow curves get their names from their shape: they turn on a specific value, which looks a bit like an elbow!

To create an elbow curve, we'll plot the clusters on the x-axis and the values of a selected objective function on the y-axis.

**Inertia** is one of the most common objective functions to use when creating an elbow curve. While what it's actually doing can get into some pretty complicated math, basically the inertia objective function is measuring the amount of variation in the dataset.

So, for our elbow curve, we'll plot the number of clusters (also known as the values of K) on the x-axis and the inertia values on the y-axis.

Let's see what happens when we plot our K values versus inertia for the preprocessed iris dataset created earlier.

We will first take a look at the elbow curve using this dataset, since we know that there should be three clusters.

Let's first take a look at the elbow curve using the iris dataset, since we know that there should be three clusters:

```
# Initial imports
import pandas as pd
from sklearn.cluster import KMeans
import plotly.express as px
import hvplot.pandas
```

Then enter the code to load in the dataset into a DataFrame:

```
# Loading data
file_path = "Resources/new_iris_data.csv"
df_iris = pd.read_csv(file_path)

df_iris.head(10)
```

## Store Values of *K* to Plot

We'll start with creating an empty list to hold inertia values. We'll also store a range of *K* values we want to test. Enter the code in a new cell:

```
inertia = []
k = list(range(1, 11))
```

## Loop Through *K* Values and Find Inertia

Next, we'll loop through each *K* value, find the inertia, and store it into our list. Enter the code in the next cell:

```
# Looking for the best K
for i in k:
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(df_iris)
    inertia.append(km.inertia_)
```

## Create a DataFrame and Plot the Elbow Curve

We'll create a DataFrame that stores our *K* values and their appropriate inertia values. This will allow for an easy plot of the results with `hvplot`. In another new cell, enter the code:

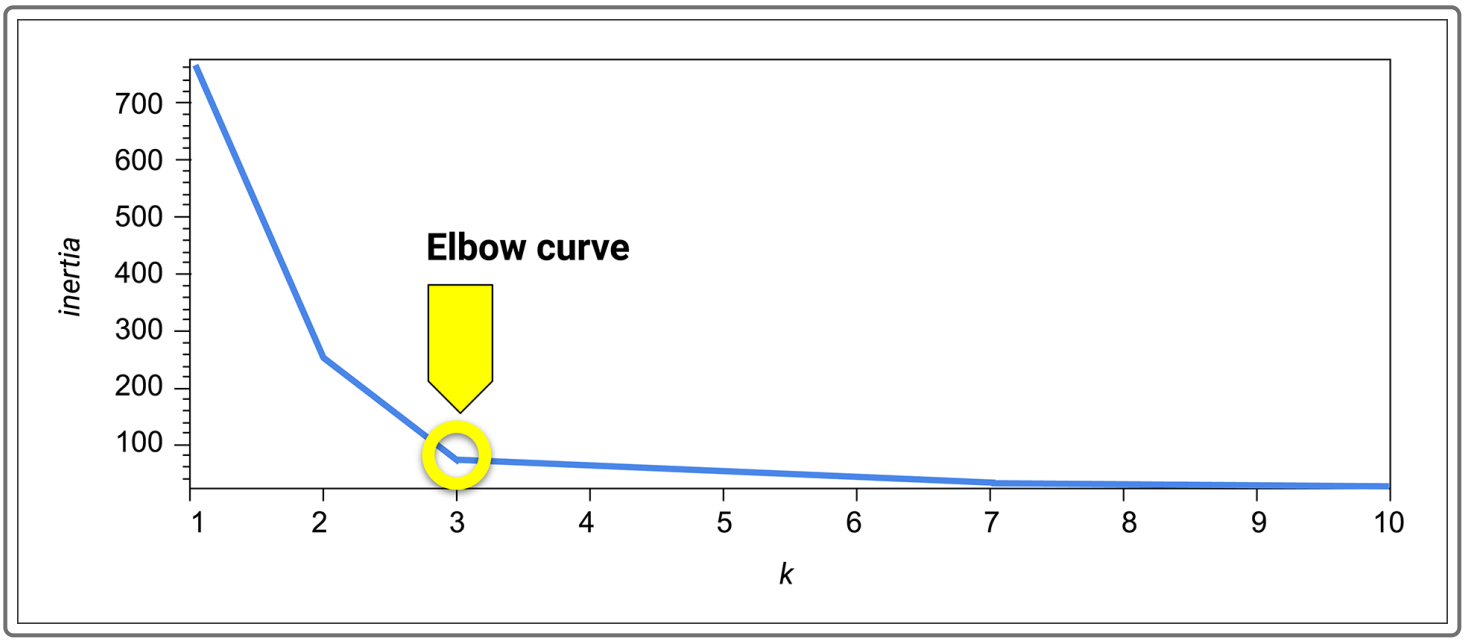
```
# Define a DataFrame to plot the Elbow Curve using hvPlot
elbow_data = {"k": k, "inertia": inertia}
df_elbow = pd.DataFrame(elbow_data)
df_elbow.hvplot.line(x="k", y="inertia", title="Elbow Curve", xticks=k)
```

This will create a graph.

## Use the Elbow Curve to Determine the Best *K* Value

Let's take a look at the graph.

Note the shape of the curve on the following graph. At point 0 (top left), the line starts as a steep vertical slope that breaks at point 2, shifts to a slightly horizontal slope, breaks again at point 3, then shifts to a strong horizontal line that reaches to point 10. The angle at point 3 looks like an elbow, which gives this type of curve its name:




© 2020 - 2022 Trilogy Education Services, a 2U, Inc. brand. All Rights Reserved.