```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('/content/sales_data_sample.csv',encoding="ISO-8859-1")
```

```
df.head()
```

| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | ORDERDATE | STATUS | QTR_ID | MONTH_ID | YEAR_ID | ... | ADDRESSLINE1 | ADDRESSLINE2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10107 | 30 | 95.70 | 2 | 2871.00 | 2/24/2003 0:00 | Shipped | 1 | 2 | 2003 | ... | 897 Long Airport Avenue | NaN | |
| **1** | 10121 | 34 | 81.35 | 5 | 2765.90 | 5/7/2003 0:00 | Shipped | 2 | 5 | 2003 | ... | 59 rue de l'Abbaye | NaN | |
| **2** | 10134 | 41 | 94.74 | 2 | 3884.34 | 7/1/2003 0:00 | Shipped | 3 | 7 | 2003 | ... | 27 rue du Colonel Pierre Avia | NaN | |
| **3** | 10145 | 45 | 83.26 | 6 | 3746.70 | 8/25/2003 0:00 | Shipped | 3 | 8 | 2003 | ... | 78934 Hillside Dr. | NaN | Pa |
| **4** | 10159 | 49 | 100.00 | 14 | 5205.27 | 10/10/2003 0:00 | Shipped | 4 | 10 | 2003 | ... | 7734 Strong St. | NaN | Fr |

5 rows × 25 columns

```
df.dtypes
```

```
ORDERNUMBER          int64
QUANTITYORDERED      int64
PRICEEACH          float64
ORDERLINENUMBER      int64
SALES              float64
ORDERDATE           object
STATUS              object
QTR_ID               int64
MONTH_ID             int64
```

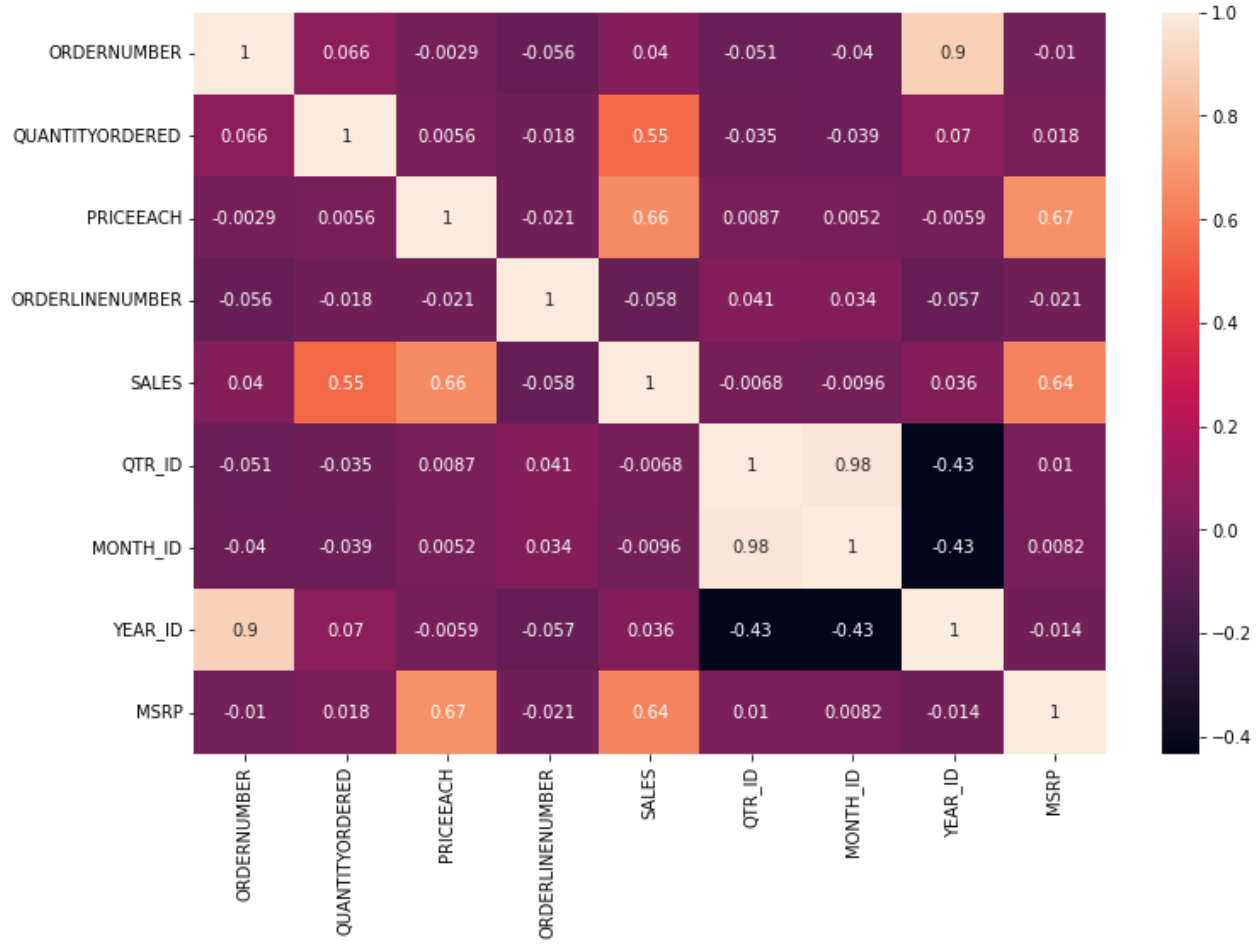
```
YEAR_ID              int64
PRODUCTLINE         object
MSRP                 int64
PRODUCTCODE         object
CUSTOMERNAME        object
PHONE               object
ADDRESSLINE1        object
ADDRESSLINE2        object
CITY                object
STATE               object
POSTALCODE          object
COUNTRY             object
TERRITORY           object
CONTACTLASTNAME     object
CONTACTFIRSTNAME    object
DEALSIZE            object
dtype: object
```

```
df.isnull().sum()
```

```
ORDERNUMBER           0
QUANTITYORDERED       0
PRICEEACH             0
ORDERLINENUMBER       0
SALES                 0
ORDERDATE             0
STATUS                0
QTR_ID                0
MONTH_ID              0
YEAR_ID               0
PRODUCTLINE           0
MSRP                  0
PRODUCTCODE           0
CUSTOMERNAME          0
PHONE                 0
ADDRESSLINE1          0
ADDRESSLINE2       2521
CITY                  0
STATE              1486
POSTALCODE           76
COUNTRY               0
TERRITORY          1074
CONTACTLASTNAME       0
CONTACTFIRSTNAME      0
DEALSIZE              0
dtype: int64
```

```
plt.figure(figsize = (12,8))
sns.heatmap(df.corr(),annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc73c338b50>
```




```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ORDERNUMBER       2823 non-null   int64
 1   QUANTITYORDERED   2823 non-null   int64
```

```
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object
 6   STATUS            2823 non-null   object
 7   QTR_ID            2823 non-null   int64
 8   MONTH_ID          2823 non-null   int64
 9   YEAR_ID           2823 non-null   int64
 10  PRODUCTLINE       2823 non-null   object
 11  MSRP              2823 non-null   int64
 12  PRODUCTCODE       2823 non-null   object
 13  CUSTOMERNAME      2823 non-null   object
 14  PHONE             2823 non-null   object
 15  ADDRESSLINE1      2823 non-null   object
 16  ADDRESSLINE2      302 non-null    object
 17  CITY              2823 non-null   object
 18  STATE             1337 non-null   object
 19  POSTALCODE        2747 non-null   object
 20  COUNTRY           2823 non-null   object
 21  TERRITORY         1749 non-null   object
 22  CONTACTLASTNAME   2823 non-null   object
 23  CONTACTFIRSTNAME  2823 non-null   object
 24  DEALSIZE          2823 non-null   object
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

```
df_drop  = ['ADDRESSLINE1', 'ADDRESSLINE2', 'STATUS','POSTALCODE', 'CITY','PHONE', 'TERRITORY','CITY','STATE','CONTACTLASTNAME','CONTACTFIRSTNAME','CUSTOMERI
df = df.drop(df_drop, axis=1)
```

```
df.shape
```

```
(2823, 14)
```

```
df.head()
```

| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | ORDERDATE | QTR_ID | MONTH_ID | YEAR_ID | PRODUCTLINE | MSRP | PRODUCTCODE | COUNTRY | DEA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10107 | 30 | 95.70 | 2 | 2871.00 | 2/24/2003 0:00 | 1 | 2 | 2003 | Motorcycles | 95 | S10_1678 | USA | |
| **1** | 10121 | 34 | 81.35 | 5 | 2765.90 | 5/7/2003 0:00 | 2 | 5 | 2003 | Motorcycles | 95 | S10_1678 | France | |
| **2** | 10134 | 41 | 94.74 | 2 | 3884.34 | 7/1/2003 0:00 | 3 | 7 | 2003 | Motorcycles | 95 | S10_1678 | France | M |

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ORDERNUMBER      2823 non-null   int64  
 1   QUANTITYORDERED  2823 non-null   int64  
 2   PRICEEACH        2823 non-null   float64
 3   ORDERLINENUMBER  2823 non-null   int64  
 4   SALES            2823 non-null   float64
 5   ORDERDATE        2823 non-null   object 
 6   QTR_ID           2823 non-null   int64  
 7   MONTH_ID         2823 non-null   int64  
 8   YEAR_ID          2823 non-null   int64  
 9   PRODUCTLINE      2823 non-null   object 
 10  MSRP             2823 non-null   int64  
 11  PRODUCTCODE      2823 non-null   object 
 12  COUNTRY          2823 non-null   object 
 13  DEALSIZE         2823 non-null   object 
dtypes: float64(2), int64(7), object(5)
memory usage: 308.9+ KB
```

```
df.shape
```

```
(2823, 14)
```

```
df.isnull().sum()
```

```
ORDERNUMBER        0
QUANTITYORDERED    0
PRICEEACH          0
ORDERLINENUMBER    0
SALES              0
```

```
ORDERDATE          0
QTR_ID             0
MONTH_ID           0
YEAR_ID            0
PRODUCTLINE        0
MSRP               0
PRODUCTCODE        0
COUNTRY            0
DEALSIZE           0
dtype: int64
```

```
df.dtypes
```

```
ORDERNUMBER          int64
QUANTITYORDERED      int64
PRICEEACH          float64
ORDERLINENUMBER      int64
SALES              float64
ORDERDATE           object
QTR_ID               int64
MONTH_ID             int64
YEAR_ID              int64
PRODUCTLINE         object
MSRP                 int64
PRODUCTCODE         object
COUNTRY             object
DEALSIZE            object
dtype: object
```

```
country = pd.get_dummies(df['COUNTRY'])
productline = pd.get_dummies(df['PRODUCTLINE'])
Dealsize = pd.get_dummies(df['DEALSIZE'])
```

```
df = pd.concat([df,country,productline,Dealsize], axis = 1)
```

```
df.head()
```

| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | ORDERDATE | QTR_ID | MONTH_ID | YEAR_ID | PRODUCTLINE | ... | Classic Cars | Motorcycles | Plar |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10107 | 30 | 95.70 | 2 | 2871.00 | 2/24/2003 0:00 | 1 | 2 | 2003 | Motorcycles | ... | 0 | 1 | |
| **1** | 10121 | 34 | 81.35 | 5 | 2765.90 | 5/7/2003 0:00 | 2 | 5 | 2003 | Motorcycles | ... | 0 | 1 | |
| **2** | 10134 | 41 | 94.74 | 2 | 3884.34 | 7/1/2003 0:00 | 3 | 7 | 2003 | Motorcycles | ... | 0 | 1 | |
| **3** | 10145 | 45 | 83.26 | 6 | 3746.70 | 8/25/2003 0:00 | 3 | 8 | 2003 | Motorcycles | ... | 0 | 1 | |
| **4** | 10159 | 49 | 100.00 | 14 | 5205.27 | 10/10/2003 | 4 | 10 | 2003 | Motorcycles | | 0 | 1 | |

```
df_drop = ['COUNTRY','PRODUCTLINE','DEALSIZE']
df = df.drop(df_drop, axis=1)
```

```
df.dtypes
```

```
ORDERNUMBER          int64
QUANTITYORDERED      int64
PRICEEACH          float64
ORDERLINENUMBER      int64
SALES              float64
ORDERDATE           object
QTR_ID               int64
MONTH_ID             int64
YEAR_ID              int64
MSRP                 int64
PRODUCTCODE         object
Australia            uint8
Austria              uint8
Belgium              uint8
Canada               uint8
Denmark              uint8
Finland              uint8
France               uint8
Germany              uint8
Ireland              uint8
Italy                uint8
Japan                uint8
Norway               uint8
Philippines          uint8
```

```
Singapore            uint8
Spain                uint8
Sweden               uint8
Switzerland          uint8
UK                   uint8
USA                  uint8
Classic Cars         uint8
Motorcycles          uint8
Planes               uint8
Ships                uint8
Trains               uint8
Trucks and Buses     uint8
Vintage Cars         uint8
Large                uint8
Medium               uint8
Small                uint8
dtype: object
```

```
df['PRODUCTCODE'] = pd.Categorical(df['PRODUCTCODE']).codes
```

```
df.dtypes
```

```
ORDERNUMBER           int64
QUANTITYORDERED       int64
PRICEEACH           float64
ORDERLINENUMBER       int64
SALES               float64
ORDERDATE            object
QTR_ID                int64
MONTH_ID              int64
YEAR_ID               int64
MSRP                  int64
PRODUCTCODE            int8
Australia            uint8
Austria              uint8
Belgium              uint8
Canada               uint8
Denmark              uint8
Finland              uint8
France               uint8
Germany              uint8
Ireland              uint8
Italy                uint8
Japan                uint8
Norway               uint8
Philippines          uint8
```

```
Singapore            uint8
Spain                uint8
Sweden               uint8
Switzerland          uint8
UK                   uint8
USA                  uint8
Classic Cars         uint8
Motorcycles          uint8
Planes               uint8
Ships                uint8
Trains               uint8
Trucks and Buses     uint8
Vintage Cars         uint8
Large                uint8
Medium               uint8
Small                uint8
dtype: object
```

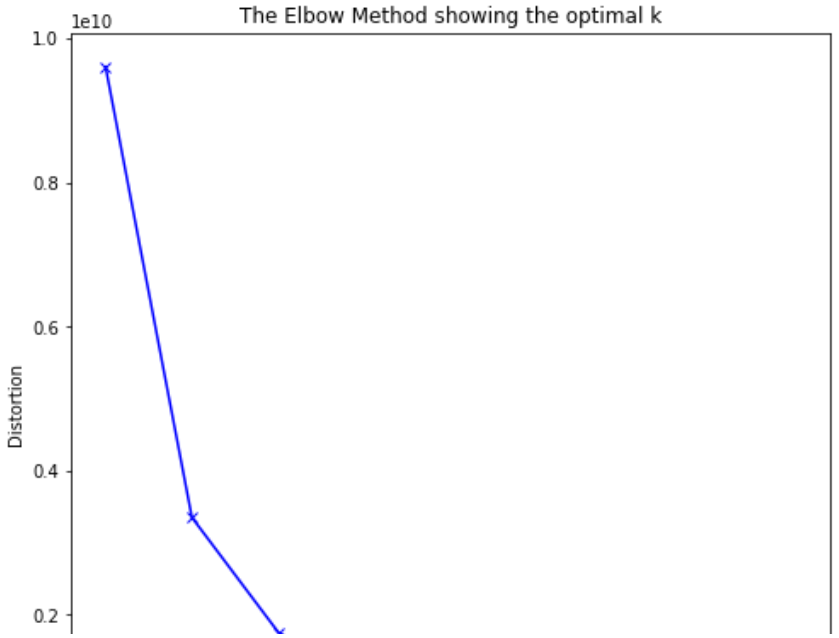
```
df.drop('ORDERDATE', axis=1, inplace=True)
```

```
df.dtypes
```

```
ORDERNUMBER          int64
QUANTITYORDERED      int64
PRICEEACH          float64
ORDERLINENUMBER      int64
SALES              float64
QTR_ID               int64
MONTH_ID             int64
YEAR_ID              int64
MSRP                 int64
PRODUCTCODE           int8
Australia            uint8
Austria              uint8
Belgium              uint8
Canada               uint8
Denmark              uint8
Finland              uint8
France               uint8
Germany              uint8
Ireland              uint8
Italy                uint8
Japan                uint8
Norway               uint8
Philippines          uint8
Singapore            uint8
```

```
Spain                uint8
Sweden               uint8
Switzerland          uint8
UK                   uint8
USA                  uint8
Classic Cars         uint8
Motorcycles          uint8
Planes               uint8
Ships                uint8
Trains               uint8
Trucks and Buses     uint8
Vintage Cars         uint8
Large                uint8
Medium               uint8
Small                uint8
dtype: object
```

```
from sklearn.cluster import KMeans


WCSS = [] # Withhin Cluster Sum of Squares from the centroid



distortions = []
K = range(1,10)
for k in K:
  kmeanModel = KMeans(n_clusters=k)
  kmeanModel.fit(df)
  distortions.append(kmeanModel.inertia_)


plt.figure(figsize=(8,8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```




```
kmeanModel = KMeans(n_clusters=3)
y_kmeans = kmeanModel.fit_predict
```

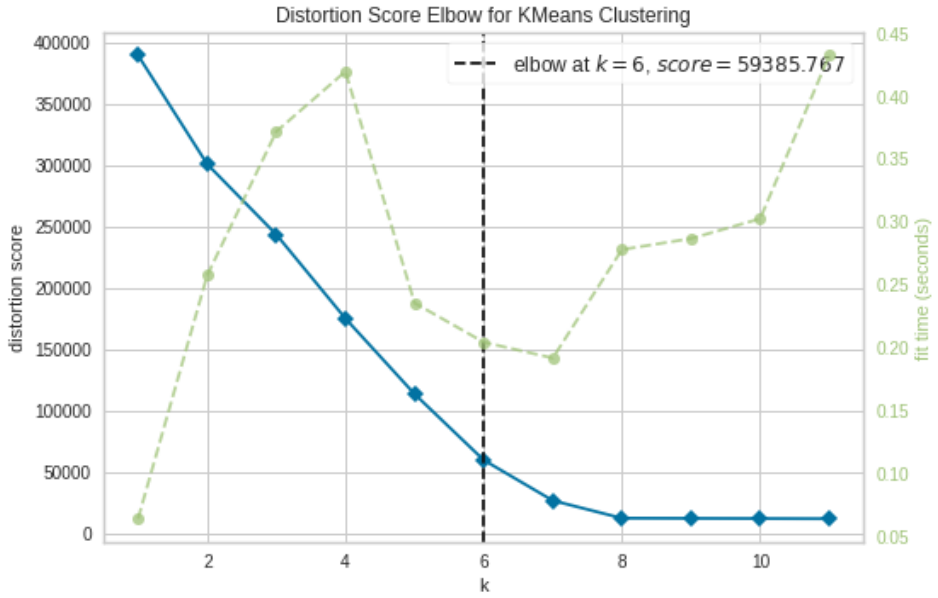
```
print(y_kmeans)
```

```
<bound method KMeans.fit_predict of KMeans(n_clusters=3)>
```

```
!pip install yellowbrick
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: yellowbrick in /usr/local/lib/python3.7/dist-packages (1.5)
Requirement already satisfied: cycler>=0.10.0 in /usr/local/lib/python3.7/dist-packages (from yellowbrick) (0.11.0)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.7/dist-packages (from yellowbrick) (1.21.6)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from yellowbrick) (3.2.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from yellowbrick) (1.7.3)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from yellowbrick) (1.0.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.4.4)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib!=3.0.0,>=2.0.2->yell
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib!=3.0.0,>=2.0.2->yellowbr
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib!=3.0.0,>=2.0.2->yellowbrick) (
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=1.0.0->yellowbrick) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=1.0.0->yellowbrick) (3.1.0)
```

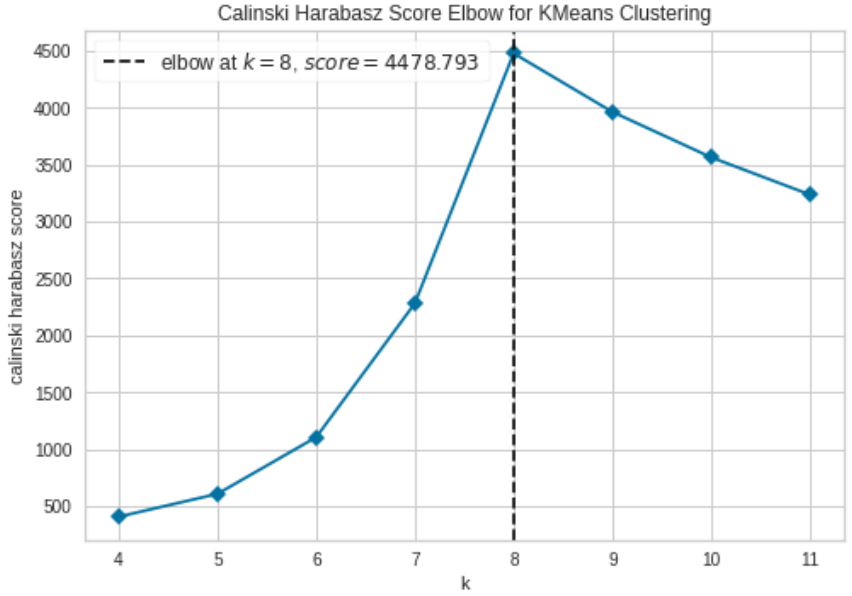
```
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

from yellowbrick.cluster import KElbowVisualizer

# Generate synthetic dataset with 8 random clusters
X, y = make_blobs(n_samples=1000, n_features=12, centers=8, random_state=42)

# Instantiate the clustering model and visualizer
model = KMeans()
visualizer = KElbowVisualizer(model, k=(4,12))

visualizer.fit(X)        # Fit the data to the visualizer
visualizer.show()        # Finalize and render the figure
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc7311f5c50>
```

```
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

from yellowbrick.cluster import KElbowVisualizer

# Generate synthetic dataset with 8 random clusters
```

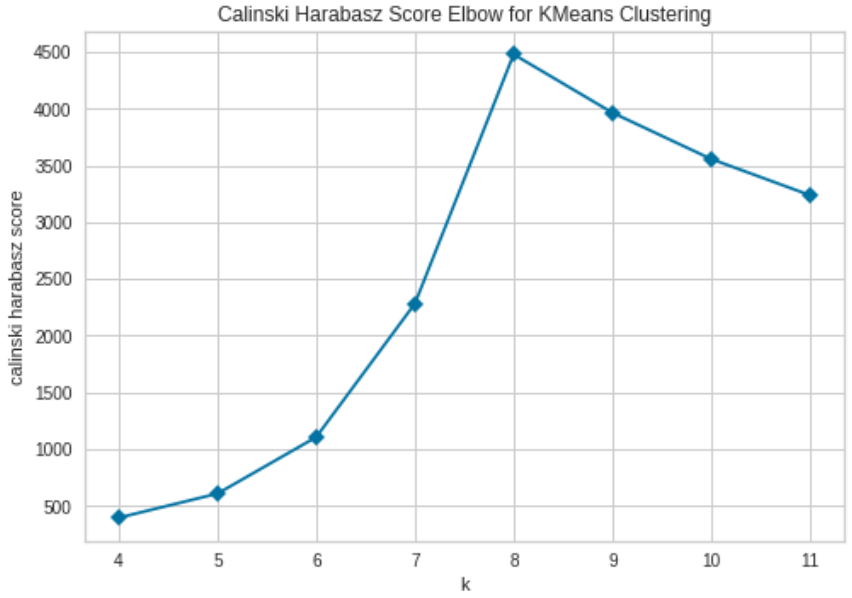
```
X, y = make_blobs(n_samples=1000, n_features=12, centers=8, random_state=42)

# Instantiate the clustering model and visualizer
model = KMeans()
visualizer = KElbowVisualizer(
    model, k=(4,12), metric='calinski_harabasz', timings=False
)

visualizer.fit(X)        # Fit the data to the visualizer
visualizer.show()        # Finalize and render the figure
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc73102d4d0>
```

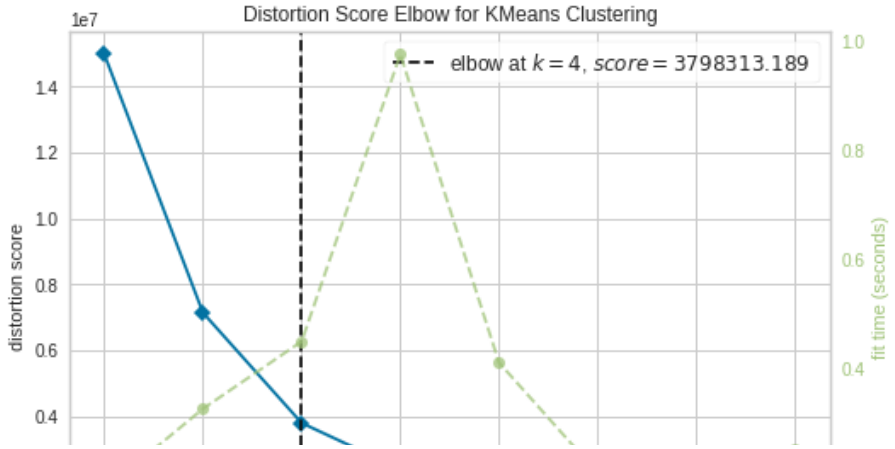
```
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

from yellowbrick.cluster import KElbowVisualizer

# Generate synthetic dataset with 8 random clusters
X, y = make_blobs(n_samples=1000, n_features=12, centers=8, random_state=42)

# Instantiate the clustering model and visualizer
model = KMeans()
visualizer = KElbowVisualizer(
```

```
    model, k=(4,12), metric='calinski_harabasz', timings=False, locate_elbow=False
)

visualizer.fit(X)        # Fit the data to the visualizer
visualizer.show()        # Finalize and render the figure
```

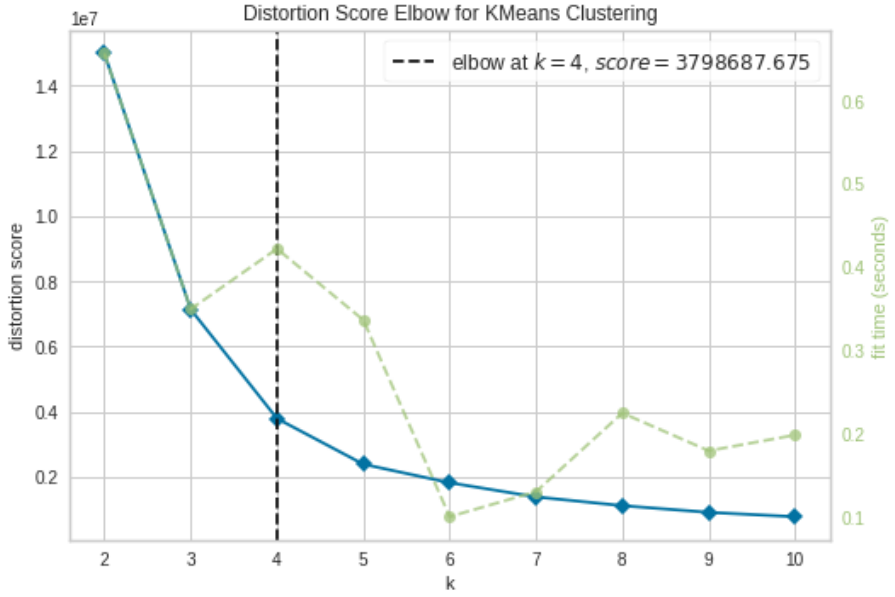



```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc730ea9e90>
```

```
from sklearn.cluster import KMeans
from yellowbrick.cluster.elbow import kelbow_visualizer
from yellowbrick.datasets.loaders import load_nfl

X, y = load_nfl()

# Use the quick method and immediately show the figure
kelbow_visualizer(KMeans(random_state=4), X, k=(2,10))
```




```
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans
model = KElbowVisualizer(KMeans(), k=10)
model.fit(X)
model.show()
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc730cf0d50>
```

Colab paid products - Cancel contracts here

0s completed at 11:35 PM