

Implement K-Means clustering on sales_data_sample.csv dataset. Determine the number of clusters using the elbow method. Dataset link : <https://www.kaggle.com/datasets/kyanyoga/sample-sales-data>

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

```
# Try reading the file with different encodings if 'utf-8' doesn't work
# ISO-8859-1 or 'latin1' is often a good fallback for non-UTF-8 files
data = pd.read_csv("C:/Users/Atharva/OneDrive/Desktop/LP3 code/sales_data_sample.csv", encoding='ISO-8859-1')

# Display the first few rows of the dataset to check if it's loaded correctly
print(data.head())
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES
0	10107	30	95.70	2	2871.00
1	10121	34	81.35	5	2765.90
2	10134	41	94.74	2	3884.34
3	10145	45	83.26	6	3746.70
4	10159	49	100.00	14	5205.27

	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	...
0	2/24/2003 0:00	Shipped	1	2	2003	...
1	5/7/2003 0:00	Shipped	2	5	2003	...
2	7/1/2003 0:00	Shipped	3	7	2003	...
3	8/25/2003 0:00	Shipped	3	8	2003	...
4	10/10/2003 0:00	Shipped	4	10	2003	...

	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE
0	897 Long Airport Avenue	NaN	NYC	NY
1	59 rue de l'Abbaye	NaN	Reims	NaN
2	27 rue du Colonel Pierre Avia	NaN	Paris	NaN
3	78934 Hillside Dr.	NaN	Pasadena	CA
4	7734 Strong St.	NaN	San Francisco	CA

	POSTALCODE	COUNTRY	TERRITORY	CONTACTLASTNAME	CONTACTFIRSTNAME	DEALSIZE
0	10022	USA	NaN	Yu	Kwai	Small
1	51100	France	EMEA	Henriot	Paul	Small
2	75508	France	EMEA	Da Cunha	Daniel	Medium
3	90003	USA	NaN	Young	Julie	Medium
4	NaN	USA	NaN	Brown	Julie	Medium

[5 rows x 25 columns]

```
# Select relevant numeric features for clustering
# Here, we're using only QUANTITYORDERED, PRICEEACH, and SALES columns for demonstration
# You may choose other columns as per the analysis needs
numeric_data = data[['QUANTITYORDERED', 'PRICEEACH', 'SALES']]
```

```
# Handle missing values (if any) by dropping rows with NaN values
numeric_data = numeric_data.dropna()
```

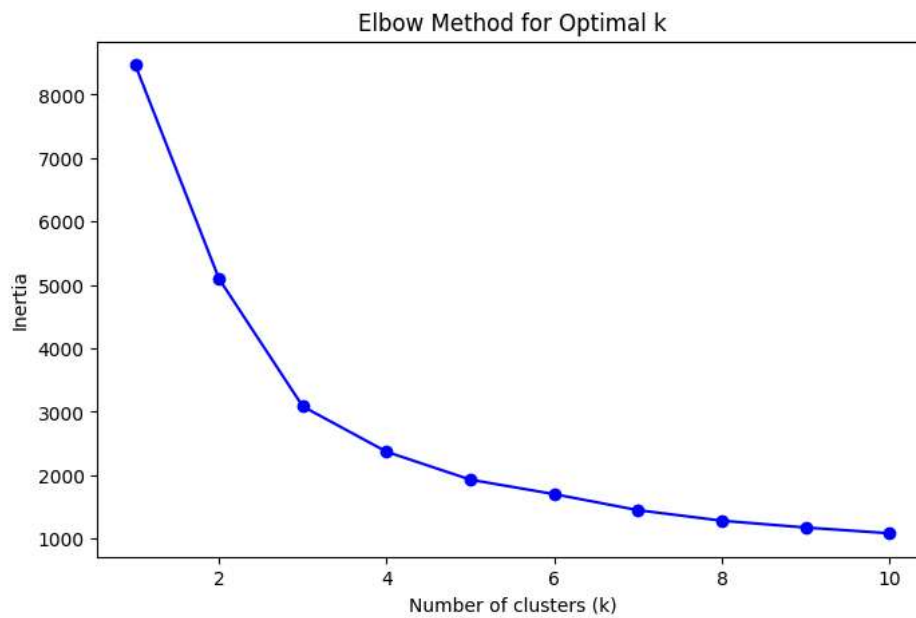
```
# Scale the features for K-Means clustering
scaler = StandardScaler()
scaled_data = scaler.fit_transform(numeric_data)
```

```
# Determine the optimal number of clusters using the elbow method
inertia = []
K = range(1, 11) # Check for 1 to 10 clusters

for k in K:
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(scaled_data)
    inertia.append(kmeans.inertia_)
```

```
# Plot the elbow graph
plt.figure(figsize=(8, 5))
plt.plot(K, inertia, 'bo-')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
```

```
plt.title('Elbow Method for Optimal k')  
plt.show()
```



```
# Once you've identified the elbow point (say 3 clusters), apply KMeans  
optimal_k = 3 # Set this to the elbow point you observed  
kmeans = KMeans(n_clusters=optimal_k, random_state=0)  
kmeans.fit(scaled_data)
```

▼ KMeans ⓘ ?

```
KMeans(n_clusters=3, random_state=0)
```

Start coding or [generate](#) with AI.