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

```
df = pd.read_csv('sales_data_sample.csv',encoding='unicode_escape')
df.head()
```

```
df.info()
```

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

```
df.info()
```

```
for col in df.columns.values:
    print(df[col].value_counts())
```

```
df.drop(columns=['ORDERDATE','STATUS','MONTH_ID','QTR_ID','YEAR_ID'],inplace=True)
df.head()
```

```
from sklearn.preprocessing import LabelEncoder
def convert_categories(col):
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col].values)
```

```
categories = ['PRODUCTLINE','PRODUCTCODE','COUNTRY','DEALSIZE']
for col in categories:
    convert_categories(col)
```

```
df.head()
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
data = sc.fit_transform(df)
```

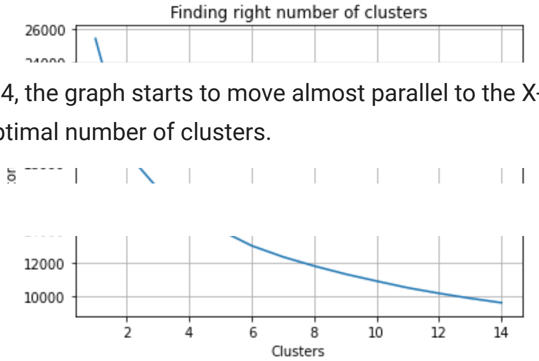
Elbow Method

Finding optimal numbers of clusters is elbow method

For each value of K, we are calculating WCSS (Within-Cluster Sum of Square). WCSS is the sum of squared distance between each point and the centroid in a cluster.When we plot the WCSS with the K value, the plot looks like an Elbow

```
from sklearn.cluster import KMeans
wcss = []
for k in range(1,15):
    kmeans = KMeans(n_clusters=k,init='k-means++',random_state=15)
    kmeans.fit(data)
    wcss.append(kmeans.inertia_)
```

```
k = list(range(1,15))
plt.plot(k,wcss)
plt.xlabel('Clusters')
plt.ylabel('scores')
plt.title('Finding right number of clusters')
plt.grid()
plt.show()
```



At k=4, the graph starts to move almost parallel to the X-axis. The K value corresponding to this point is the optimal K value or an optimal number of clusters.