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

```
data = pd.read_csv("sales_data_sample.csv", encoding='Latin-1')
data.head()

# While utf-8 supports all languages according to pandas' documentation, utf-8 has a byte structure that must be respected at all times. Some
```

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

5 rows × 25 columns

```
data.shape
```

```
(2823, 25)
```

```
# Number of NAN values per column in the dataset
data.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
```

```
data.drop(["ORDERNUMBER", "PRICEEACH", "ORDERDATE", "PHONE", "ADDRESSLINE1", "ADDRESSLINE2", "CITY", "STATE", "TERRITORY", "POSTALCODE", "CC
```

```
data.head()
```

| | QUANTITYORDERED | ORDERLINENUMBER | SALES | STATUS | QTR_ID | MONTH_ID | YEAR_ID | PRODUCTLINE | MSRP | PRODUCTCODE | CUSTOMERNAME | COUN |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 30 | 2 | 2871.00 | Shipped | 1 | 2 | 2003 | Motorcycles | 95 | S10_1678 | Land of Toys Inc. | U |
| **1** | 34 | 5 | 2765.90 | Shipped | 2 | 5 | 2003 | Motorcycles | 95 | S10_1678 | Reims Collectables | Fra |
| **2** | 41 | 2 | 3884.34 | Shipped | 3 | 7 | 2003 | Motorcycles | 95 | S10_1678 | Lyon Souveniers | Fra |
| **3** | 45 | 6 | 3746.70 | Shipped | 3 | 8 | 2003 | Motorcycles | 95 | S10_1678 | Toys4GrownUps.com | U |
| | | | | | | | | | | | Corporate Gift Ideas | |

```
data.isnull().sum()
```

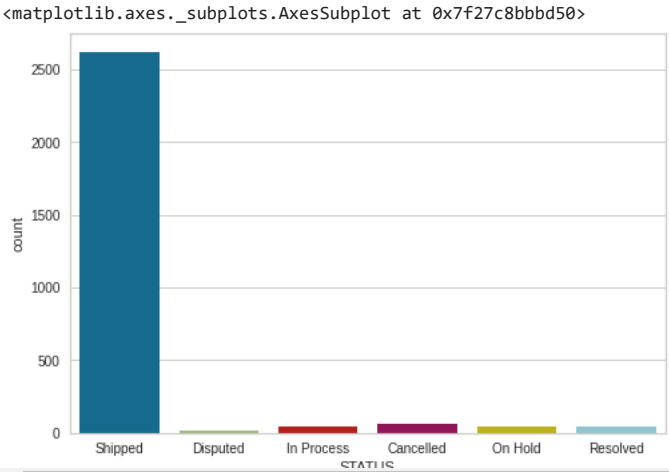
```
QUANTITYORDERED    0
ORDERLINENUMBER    0
SALES              0
STATUS             0
QTR_ID             0
MONTH_ID           0
YEAR_ID            0
PRODUCTLINE        0
MSRP               0
PRODUCTCODE        0
CUSTOMERNAME       0
COUNTRY            0
DEALSIZE           0
dtype: int64
```

```
data.describe()
```

| | QUANTITYORDERED | ORDERLINENUMBER | SALES | QTR_ID | MONTH_ID | YEAR_ID | MSRP |
|---|---|---|---|---|---|---|---|
| **count** | 2823.000000 | 2823.000000 | 2823.000000 | 2823.000000 | 2823.000000 | 2823.00000 | 2823.000000 |
| **mean** | 35.092809 | 6.466171 | 3553.889072 | 2.717676 | 7.092455 | 2003.81509 | 100.715551 |
| **std** | 9.741443 | 4.225841 | 1841.865106 | 1.203878 | 3.656633 | 0.69967 | 40.187912 |
| **min** | 6.000000 | 1.000000 | 482.130000 | 1.000000 | 1.000000 | 2003.00000 | 33.000000 |
| **25%** | 27.000000 | 3.000000 | 2203.430000 | 2.000000 | 4.000000 | 2003.00000 | 68.000000 |
| **50%** | 35.000000 | 6.000000 | 3184.800000 | 3.000000 | 8.000000 | 2004.00000 | 99.000000 |
| **75%** | 43.000000 | 9.000000 | 4508.000000 | 4.000000 | 11.000000 | 2004.00000 | 124.000000 |
| **max** | 97.000000 | 18.000000 | 14082.800000 | 4.000000 | 12.000000 | 2005.00000 | 214.000000 |

```
sns.countplot(data = data , x = 'STATUS')
```

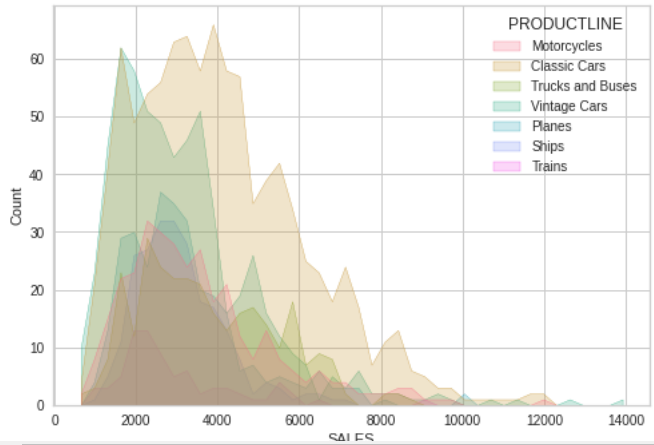
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f27c8bbbd50>
```




```
import seaborn as sns
```

```
sns.histplot(x = 'SALES' , hue = 'PRODUCTLINE', data = data,
             element="poly")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f27c8b6f210>
```




Here we can see all the catagory lies in the range of price and hence in this we be creating a cluster on targeting the same

```
data['PRODUCTLINE'].unique()
```

```
array(['Motorcycles', 'Classic Cars', 'Trucks and Buses', 'Vintage Cars',
       'Planes', 'Ships', 'Trains'], dtype=object)
```

```
#checking the duplicated values
data.drop_duplicates(inplace=True)
```

```
data.info()
```

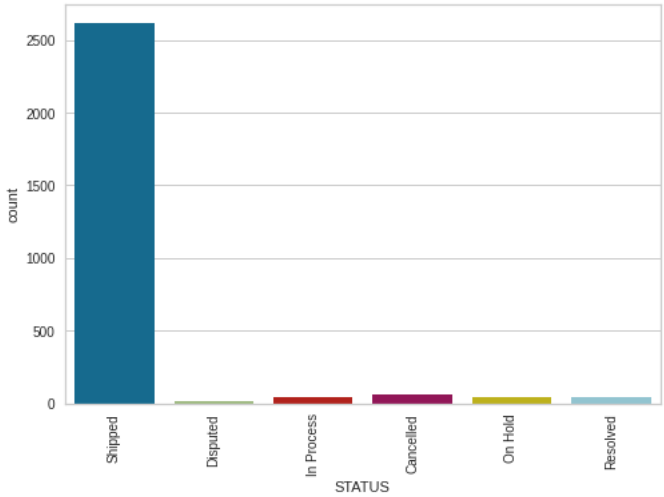
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2823 entries, 0 to 2822
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   QUANTITYORDERED  2823 non-null   int64
 1   ORDERLINENUMBER  2823 non-null   int64
 2   SALES            2823 non-null   float64
 3   STATUS           2823 non-null   object
 4   QTR_ID           2823 non-null   int64
 5   MONTH_ID         2823 non-null   int64
 6   YEAR_ID          2823 non-null   int64
 7   PRODUCTLINE      2823 non-null   object
 8   MSRP             2823 non-null   int64
 9   PRODUCTCODE      2823 non-null   object
 10  CUSTOMERNAME     2823 non-null   object
 11  COUNTRY          2823 non-null   object
 12  DEALSIZE         2823 non-null   object
dtypes: float64(1), int64(6), object(6)
memory usage: 308.8+ KB
```

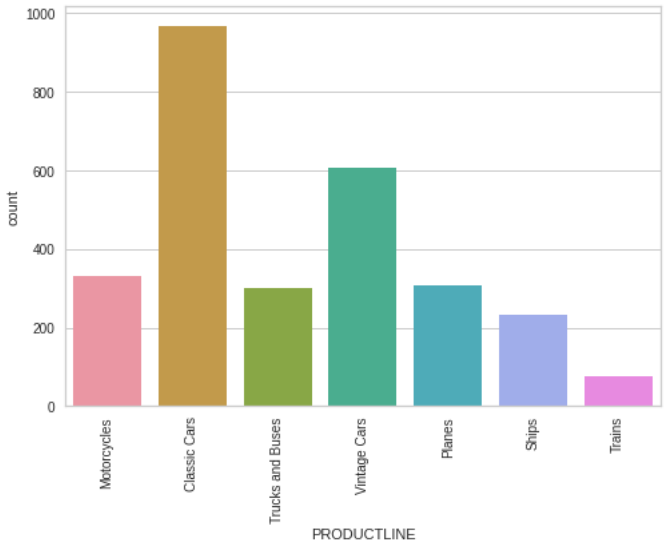
```
list_cat = data.select_dtypes(include=['object']).columns.tolist()
```

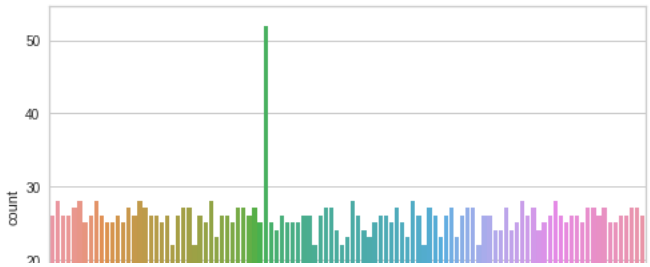
```
list_cat
```

```
['STATUS', 'PRODUCTLINE', 'PRODUCTCODE', 'CUSTOMERNAME', 'COUNTRY', 'DEALSIZE']
```

```
for i in list_cat:
  sns.countplot(data = data ,x = i)
  plt.xticks(rotation = 90)
  plt.show()
```




```
#dealing with the catagorical features
from sklearn import preprocessing
le = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
for i in list_cat:
  data[i]= le.fit_transform(data[i])
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2823 entries, 0 to 2822
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   QUANTITYORDERED  2823 non-null   int64
 1   ORDERLINENUMBER  2823 non-null   int64
 2   SALES            2823 non-null   float64
 3   STATUS           2823 non-null   int64
 4   QTR_ID           2823 non-null   int64
 5   MONTH_ID         2823 non-null   int64
 6   YEAR_ID          2823 non-null   int64
 7   PRODUCTLINE      2823 non-null   int64
 8   MSRP             2823 non-null   int64
```

```
 9   PRODUCTCODE      2823 non-null   int64
 10  CUSTOMERNAME     2823 non-null   int64
 11  COUNTRY          2823 non-null   int64
 12  DEALSIZE         2823 non-null   int64
dtypes: float64(1), int64(12)
memory usage: 373.3 KB
```

```
data['SALES'] = data['SALES'].astype(int)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2823 entries, 0 to 2822
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   QUANTITYORDERED  2823 non-null   int64
 1   ORDERLINENUMBER  2823 non-null   int64
 2   SALES            2823 non-null   int64
 3   STATUS           2823 non-null   int64
 4   QTR_ID           2823 non-null   int64
 5   MONTH_ID         2823 non-null   int64
 6   YEAR_ID          2823 non-null   int64
 7   PRODUCTLINE      2823 non-null   int64
 8   MSRP             2823 non-null   int64
 9   PRODUCTCODE      2823 non-null   int64
 10  CUSTOMERNAME     2823 non-null   int64
 11  COUNTRY          2823 non-null   int64
 12  DEALSIZE         2823 non-null   int64
dtypes: int64(13)
memory usage: 373.3 KB
```

```
data.describe()
```

| | QUANTITYORDERED | ORDERLINENUMBER | SALES | STATUS | QTR_ID | MONTH_ID | YEAR_ID | PRODUCTLINE | MSRP | PRODUCT |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 2823.000000 | 2823.000000 | 2823.000000 | 2823.000000 | 2823.000000 | 2823.000000 | 2823.00000 | 2823.000000 | 2823.000000 | 2823.00 |
| **mean** | 35.092809 | 6.466171 | 3553.421537 | 4.782501 | 2.717676 | 7.092455 | 2003.81509 | 2.515055 | 100.715551 | 53.77 |
| **std** | 9.741443 | 4.225841 | 1841.865754 | 0.879416 | 1.203878 | 3.656633 | 0.69967 | 2.411665 | 40.187912 | 31.58 |
| **min** | 6.000000 | 1.000000 | 482.000000 | 0.000000 | 1.000000 | 1.000000 | 2003.00000 | 0.000000 | 33.000000 | 0.00 |
| **25%** | 27.000000 | 3.000000 | 2203.000000 | 5.000000 | 2.000000 | 4.000000 | 2003.00000 | 0.000000 | 68.000000 | 27.00 |
| **50%** | 35.000000 | 6.000000 | 3184.000000 | 5.000000 | 3.000000 | 8.000000 | 2004.00000 | 2.000000 | 99.000000 | 53.00 |
| **75%** | 43.000000 | 9.000000 | 4508.000000 | 5.000000 | 4.000000 | 11.000000 | 2004.00000 | 5.000000 | 124.000000 | 81.00 |
| **max** | 97.000000 | 18.000000 | 14082.000000 | 5.000000 | 4.000000 | 12.000000 | 2005.00000 | 6.000000 | 214.000000 | 108.00 |

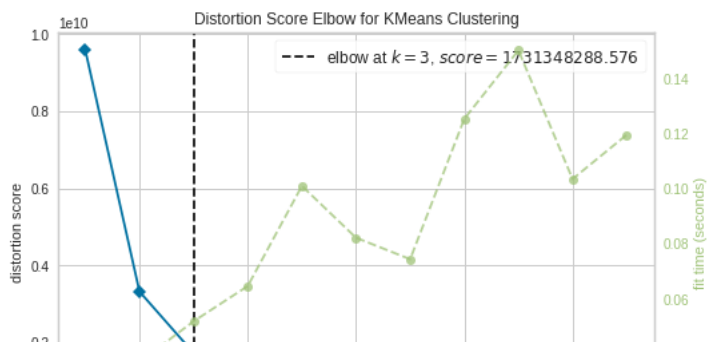
```
## taget feature are Sales and productline
X = data[['SALES','PRODUCTCODE']]
```

```
data.columns
```

```
Index(['QUANTITYORDERED', 'ORDERLINENUMBER', 'SALES', 'STATUS', 'QTR_ID',
       'MONTH_ID', 'YEAR_ID', 'PRODUCTLINE', 'MSRP', 'PRODUCTCODE',
       'CUSTOMERNAME', 'COUNTRY', 'DEALSIZE'],
      dtype='object')
```

## K Means implementation

```
from yellowbrick.cluster import KElbowVisualizer
model = KMeans()
visualizer = KElbowVisualizer(model, k=(1,12)).fit(X)
visualizer.show()
```




```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0).fit(X)
```

```
kmeans.labels_
```

```
array([0, 0, 0, ..., 3, 2, 0], dtype=int32)
```

```
kmeans.inertia_
```

```
1042223216.6249831
```

```
kmeans.n_iter_
```

```
24
```

```
kmeans.cluster_centers_
```

```
array([[3416.59686888,   56.3072407 ],
       [7983.1758794 ,   28.05025126],
       [1879.28363988,   63.25072604],
       [5289.27065026,   41.01230228]])
```

```
#getting the size of the clusters
from collections import Counter
Counter(kmeans.labels_)
```

```
Counter({0: 1024, 3: 565, 2: 1035, 1: 199})
```

Hence the NUmber of Clusters to be choosen Will be 4 according to the elbow method

```
sns.scatterplot(data=X, x="SALES", y="PRODUCTCODE", hue=kmeans.labels_)
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1],
            marker="X", c="r", s=80, label="centroids")
plt.legend()
plt.show()
```