```
cust=pd.read_csv('customer_data.csv')
```

1) Import Libraries

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

2) To see if there are any null values in the dataset

```
cust=cust.dropna()
```

3) Drop all rows having Family_Size values as 7.0, 8.0 and 9.0.

```
new_df = df[(df['Family_Size']!= 7) & (df['Family_Size']!= 8) & (df['Family_Size']!=
9)]
```

4) Value count

```
data['Columns'].value_counts()
```

Dropping unwanted columns

```
data=data.drop(['Column1',' Column2'],axis=1)

example: customer=customer.drop(['CustomerID'],axis=1)
customer
```

One hot encoding - Changing catogorical to numerical

```
data=pd.get_dummies(data,columns=['Column1','column2])

x=pd.get_dummies(cust2,columns=
['Gender','Graduated','Profession','Spending_Score','Ever_Married'])
```

Outlier Detection

```
from scipy.stats import zscore

z_score=zscore(data)
z_score
```

```
z_score[z_score>3].count()-----------z_score[z_score<-3].count()
```

removing outliers

```
filter_entry= (z_score >=-3) & (z_score <=3)
```

Removing oultiers and assigning the rest of the values to X

```
x=data[filter_entry]
x
```

Dropping null values

```
x=x.dropna()
```

Replacing null values

```
data.fillna(data.median(), inplace=True)
```

X - Independent Variable Y- Target Variable

```
x=data.drop(['column1'],axis=1)


y=data['column1']

example:

Dividing the data into Features and Target variable
X = Data.drop('DefaultPayment_NxtMonth', axis=1) # Features
y = Data['DefaultPayment_NxtMonth']
```

Normalization

```
import pandas as pd
from sklearn.preprocessing import Normalizer

# Load data from CSV into a pandas DataFrame
data = pd.read_csv('your_data.csv')

# Extract the data as a numpy array
X = data.values

# Create a Normalizer object
normalizer = Normalizer()

# Apply normalization to the data
```

```
X_normalized = normalizer.fit_transform(X)
```

Target variable label encoding

```
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
df['species']= label_encoder.fit_transform(df['species'])

df['species'].unique()
```

Comparing clusters

```
calinski_harabasz_score

from sklearn.metrics import calinski_harabasz_score
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
score = calinski_harabasz_score(X, kmeans.labels_)
print("Calinski-Harabasz Index:", score)



silhouette_score
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
score = silhouette_score(X, kmeans.labels_)
print("Silhouette Score:", score)

davies_bouldin_score
from sklearn.metrics import davies_bouldin_score
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
score = davies_bouldin_score(X, kmeans.labels_)
print("Davies-Bouldin Index:", score)
```

Plot for model performance

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import numpy as np

# Generate random data for clustering
X = np.random.rand(100, 2)

# Create a list of different numbers of clusters to test
```

```
cluster_range = range(2, 11)

# Calculate the Silhouette Score for each number of clusters
scores = []
for n_clusters in cluster_range:
    clusterer = KMeans(n_clusters=n_clusters)
    cluster_labels = clusterer.fit_predict(X)
    silhouette_avg = silhouette_score(X, cluster_labels)
    scores.append(silhouette_avg)

# Plot the Silhouette Score for each number of clusters
plt.plot(cluster_range, scores)
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Score')
plt.show()
```