```
# Importing Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from kmodes.kprototypes import KPrototypes

# Load the Titanic Dataset and Process it
DT = sns.load_dataset("titanic")
DT = pd.DataFrame(DT)

# Display the shape of the original dataset
print("Original dataset shape:", DT.shape)

# Drop unnecessary columns
D = DT.drop(['survived', 'pclass', 'deck', 'embark_town'], axis=1)

# Check for missing values before dropping rows
print("Missing values before dropping rows:\n", D.isna().sum())

# Drop rows with any missing values
D.dropna(how="any", inplace=True)

# Display the shape of the dataset after dropping missing values
print("Dataset shape after dropping missing values:", D.shape)

# Sample 90% of the preprocessed dataset without replacement
np.random.seed(201681)
data = D.sample(frac=0.9, replace=False, random_state=17)
```

```
# Display the first few rows of the sampled data
print("Sampled data preview:\n", data.head())

# Display the data types of the columns
print("Data types:\n", data.dtypes)

# Looping over a range of cluster numbers to compute Cost
cost = []

for i in range(1, 7):
    KP = KPrototypes(n_clusters=i, init="Huang", max_iter=500, random_state=17, verbose=0)
    KP.fit(data, categorical=[0, 5, 6, 7, 8, 9, 10])
    cost.append(KP.cost_)

# Display the computed cost for different numbers of clusters
print("Cost for different numbers of clusters:", cost)

# Plotting the Elbow Method to determine the optimal number of clusters
number_clusters = range(1, 7)
plt.figure(figsize=(6, 4))
plt.plot(number_clusters, cost, color="red", linestyle="solid", linewidth=2, marker='o',
markerfacecolor="blue", markersize=5)
plt.axvline(x=3, color="blue", ls="--", linewidth=2)
plt.xlabel("Number of clusters")
plt.ylabel("Cost")
plt.title("Elbow Method for Optimal value of k")
plt.grid(True)
plt.show()
```

```
# Fit KPrototypes clustering with the optimal number of clusters (k=3)
Kproto = KPrototypes(n_clusters=3, init="Huang", n_init=100, verbose=2, max_iter=500)
labels = Kproto.fit_predict(data, categorical=[0, 5, 6, 7, 8, 9, 10])

# Display cluster centroids, cost, and labels
print("Cluster centroids:\n", Kproto.cluster_centroids_)
print("Cluster labels:\n", labels)
print("Final cost:", Kproto.cost_)

# Insert cluster labels into the original dataset
data.insert(0, "Cluster", labels, True)

# Visualize the cluster distribution
sns.countplot(x='Cluster', data=data)
plt.title('Cluster Distribution')
plt.xlabel('Cluster')
plt.ylabel('Count')
plt.show()
```