In [1]:

```
# Initial imports
import pandas as pd
import hvplot.pandas
from pathlib import Path
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from path import Path
from sklearn import tree
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
import plotly.express as px
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import numpy as np
import nbconvert
```

In [2]:

```
df = pd.read_csv("C:/Users/kethr/Desktop/Class/Final Project/Price_Transparency_Analysi
df.head()
```

Out[2]:

	US News Ranking	Hospital Name	DRG Number	DRG Description	Payor	Plan	Charge Amount
0	8	NYU Langone Hospitals Hospital A	1	HEART TRANSPLANT OR IMPLANT OF HEART ASSIST SY...	Aetna	Hospital A1005AETNA HMO	618308.8
1	8	NYU Langone Hospitals Hospital A	2	HEART TRANSPLANT OR IMPLANT OF HEART ASSIST SY...	Aetna	Hospital A1005AETNA HMO	320135.6
2	8	NYU Langone Hospitals Hospital A	3	ECMO OR TRACHEOSTOMY WITH MV >96 HOURS OR PRIN...	Aetna	Hospital A1005AETNA HMO	408571.1
3	8	NYU Langone Hospitals Hospital A	4	TRACHEOSTOMY WITH MV >96 HOURS OR PRINCIPAL DI...	Aetna	Hospital A1005AETNA HMO	254962.7
4	8	NYU Langone Hospitals Hospital A	5	LIVER TRANSPLANT WITH MCC OR INTESTINAL TRANSP...	Aetna	Hospital A1005AETNA HMO	218875.5

In [3]:

```
 #Remove text columns.
#df_clean = df.drop(['Plan','DRG Description', 'Hospital Name', "Specialty _Cancer", "S
df_clean = df.drop(['Plan','DRG Description', 'Hospital Name'], axis=1)
```

In [4]:

```
df_clean.shape
```

Out[4]:

```
(1200922, 4)
```

In [5]:

```
df_clean.describe(include='all')
```

Out[5]:

	US News Ranking	DRG Number	Payor	Charge Amount
count	1.200922e+06	1.200922e+06	1200922	1.200922e+06
unique	NaN	NaN	7	NaN
top	NaN	NaN	Other	NaN
freq	NaN	NaN	789204	NaN
mean	8.643914e+00	4.772373e+02	NaN	5.332684e+04
std	2.885413e+00	2.800987e+02	NaN	9.223762e+04
min	1.000000e+00	1.000000e+00	NaN	9.820000e+00
25%	8.000000e+00	2.410000e+02	NaN	1.548044e+04
50%	8.000000e+00	4.720000e+02	NaN	2.956193e+04
75%	8.000000e+00	7.180000e+02	NaN	5.812355e+04
max	2.000000e+01	9.890000e+02	NaN	3.649520e+07

In [6]:

```
df_clean.dtypes
```

Out[6]:

```
US News Ranking      int64
DRG Number           int64
Payor               object
Charge Amount      float64
dtype: object
```

In [7]:

```
# Use get_dummies() to create variables for text feature.
X = pd.get_dummies(df_clean, columns=["Payor"])
print(X.shape)
X.head(10)
```

```
(1200922, 10)
```

Out[7]:

	US News Ranking	DRG Number	Charge Amount	Payor_Aetna	Payor_BCBS	Payor_Cigna	Payor_Gross Charge	Payor_Other	Payo
0	8	1	618308.8	1	0	0	0	0	
1	8	2	320135.6	1	0	0	0	0	
2	8	3	408571.1	1	0	0	0	0	
3	8	4	254962.7	1	0	0	0	0	
4	8	5	218875.5	1	0	0	0	0	
5	8	6	100432.5	1	0	0	0	0	
6	8	7	247638.3	1	0	0	0	0	
7	8	8	116191.1	1	0	0	0	0	
8	8	10	77413.7	1	0	0	0	0	

	US News Ranking	DRG Number	Charge Amount	Payor_Aetna	Payor_BCBS	Payor_Cigna	Payor_Gross Charge	Payor_Other	Payo
9	8	11	107382.6	1	0	0	0	0	

In [8]:

```
# Standardize the data with StandardScaler().
df_scaled = StandardScaler().fit_transform(X)
print(df_scaled[0:5])
```

```
[[-0.22316178 -1.70024911  6.12529055  4.19628978 -0.44268576 -0.23696036
  -0.05997336 -1.38450558 -0.06052326 -0.26331537]
 [-0.22316178 -1.69667893  2.89262542  4.19628978 -0.44268576 -0.23696036
  -0.05997336 -1.38450558 -0.06052326 -0.26331537]
 [-0.22316178 -1.69310876  3.85140494  4.19628978 -0.44268576 -0.23696036
  -0.05997336 -1.38450558 -0.06052326 -0.26331537]
 [-0.22316178 -1.68953859  2.18604897  4.19628978 -0.44268576 -0.23696036
  -0.05997336 -1.38450558 -0.06052326 -0.26331537]
 [-0.22316178 -1.68596842  1.79480713  4.19628978 -0.44268576 -0.23696036
  -0.05997336 -1.38450558 -0.06052326 -0.26331537]]
```

In [9]:

```
# Using PCA to reduce dimension to three principal components.
pca = PCA(n_components=3)
df_clean_pca = pca.fit_transform(df_scaled)
print(df_clean_pca[0:])
```

```
[[-1.22079767 -1.05563292  5.57105795]
 [-1.47007431 -1.56748803  3.39601788]
 [-1.3962105  -1.41614845  4.03817386]
 ...
 [-2.4256751  -2.02476703 -0.70867093]
 [ 0.09113024 -1.73607051 -1.3675507 ]
 [-2.4204179  -2.01524853 -0.67184707]]
```

In [10]:

```
# Create a DataFrame with the three principal components.
pcs_df = pd.DataFrame(data=df_clean_pca, index=df_clean.index, columns=["PC 1", "PC 2",
print(pcs_df.shape)
pcs_df
```

```
(1200922, 3)
```

Out[10]:

	PC 1	PC 2	PC 3
0	-1.220798	-1.055633	5.571058
1	-1.470074	-1.567488	3.396018
2	-1.396211	-1.416148	4.038174
3	-1.524655	-1.680015	2.916569
4	-1.554871	-1.742283	2.651335
...	...	...	...
1200917	0.105721	-1.705871	-1.238543
1200918	-2.431151	-2.036005	-0.756410

	PC 1	PC 2	PC 3
1200919	-2.425675	-2.024767	-0.708671
1200920	0.091130	-1.736071	-1.367551
1200921	-2.420418	-2.015249	-0.671847

1200922 rows × 3 columns

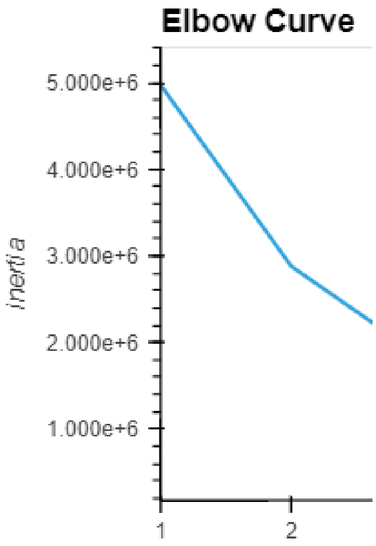
In [11]:

```
# Create an elbow curve to find the best value for K.
inertia = []
k = list(range(1, 11))

# Look for the best K
for i in k:
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(pcs_df)
    inertia.append(km.inertia_)

# Create the elbow curve
elbow_data = {"k": k, "inertia": inertia}
df_elbow = pd.DataFrame(elbow_data)
df_elbow.hvplot.line(x="k", y="inertia", xticks=k, title="Elbow Curve")
```

Out[11]:



In [12]:

```
# Initialize the K-Means model.
model = KMeans(n_clusters=3)

# Fit the model
model.fit(pcs_df)

# Predict clusters
predictions = model.predict(pcs_df)
predictions
```

```
array([1, 1, 1, ..., 1, 0, 1])
```

Out[12]:

In [13]:

```
# Create a new DataFrame including predicted clusters and features.
# Concatentate the crypto_df and pcs_df DataFrames on the same columns.
clustered_df = df_clean.join(pcs_df)
```

In [14]:

```
#  Add a new column, "Class" to the clustered_df DataFrame that holds the predictions.
clustered_df["Class"] = model.labels_
```

In [15]:

```
# Print the shape of the clustered_df
print(clustered_df.shape)
clustered_df.head(10)
```

```
(1200922, 8)
```

Out[15]:

	US News Ranking	DRG Number	Payor	Charge Amount	PC 1	PC 2	PC 3	Class
0	8	1	Aetna	618308.8	-1.220798	-1.055633	5.571058	1
1	8	2	Aetna	320135.6	-1.470074	-1.567488	3.396018	1
2	8	3	Aetna	408571.1	-1.396211	-1.416148	4.038174	1
3	8	4	Aetna	254962.7	-1.524655	-1.680015	2.916569	1
4	8	5	Aetna	218875.5	-1.554871	-1.742283	2.651335	1
5	8	6	Aetna	100432.5	-1.653924	-1.945827	1.785979	1
6	8	7	Aetna	247638.3	-1.530938	-1.693671	2.856391	1
7	8	8	Aetna	116191.1	-1.640859	-1.919522	1.896274	1
8	8	10	Aetna	77413.7	-1.673378	-1.986770	1.609168	1
9	8	11	Aetna	107382.6	-1.648382	-1.935725	1.825281	1

In [16]:

```
# # Creating a 3D-Scatter with the PCA data and the clusters
fig = px.scatter_3d(
    clustered_df,
    x="PC 1",
    y="PC 2",
    z="PC 3",
    color="Class",
    symbol="Class",
    width=800,
    hover_name="US News Ranking",
    hover_data=["Payor"]
)
fig.update_layout(legend=dict(x=0, y=1))
fig.show()
```

info

Class

● 1
◆ 2
■ 0

In [17]:

```
# Create a table
clustered_df.hvplot.table(columns=['US News Ranking', 'DRG Number', 'Payor', 'Charge Am
```

Out[17]:

#	US News Ranking	DR
0	8	1
1	8	2
2	8	3
3	8	4
4	8	5
5	8	6
6	8	7
7	8	8
8	8	10
9	8	11

In [18]:

```
# Scaling data to create the scatter plot
data = df_clean[['US News Ranking', 'Charge Amount']]
new_scaled = MinMaxScaler().fit_transform(data)
print(new_scaled[0:])
```

```
[[3.68421053e-01 1.69419306e-02]
 [3.68421053e-01 8.77172522e-03]
 [3.68421053e-01 1.11949349e-02]
 ...
 [8.42105263e-01 5.30413183e-04]
 [8.42105263e-01 8.53391361e-04]
 [8.42105263e-01 7.11533215e-04]]
```

In [19]:

```
# Create a new DataFrame that has the scaled data with the clustered_df DataFrame index
plot_df = pd.DataFrame(data=new_scaled, index=clustered_df.index, columns=['US News Ran

# Add the "CoinName" column from the clustered_df DataFrame to the new DataFrame.
plot_df = plot_df.join(clustered_df['Payor'], on=plot_df.index)

# Add the "Class" column from the clustered_df DataFrame to the new DataFrame.
plot_df = plot_df.join(clustered_df['Class'], on=plot_df.index)

plot_df.head(10)
```

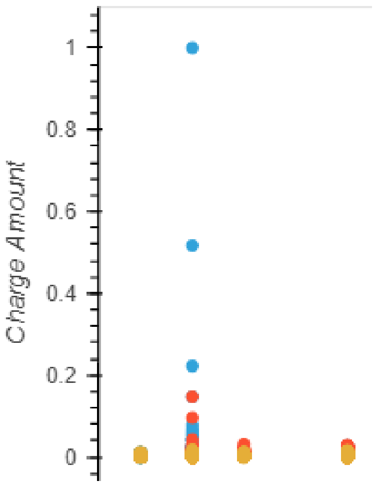
Out[19]:

	US News Ranking	Charge Amount	Payor	Class
0	0.368421	0.016942	Aetna	1
1	0.368421	0.008772	Aetna	1
2	0.368421	0.011195	Aetna	1
3	0.368421	0.006986	Aetna	1
4	0.368421	0.005997	Aetna	1
5	0.368421	0.002752	Aetna	1
6	0.368421	0.006785	Aetna	1
7	0.368421	0.003183	Aetna	1
8	0.368421	0.002121	Aetna	1
9	0.368421	0.002942	Aetna	1

In [20]:

```
# Create a hvplot.scatter plot
plot_df.hvplot.scatter(x="US News Ranking", y="Charge Amount", by="Class")
```

Out[20]:



0
0.2