Clustering Crypto

In [1]:
```python
# Initial imports
import pandas as pd
import hvplot.pandas
from path import Path
import plotly.express as px
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
```

Deliverable 1: Preprocessing the Data for PCA

In [7]:
```python
# Load the crypto_data.csv dataset.
crypto_df = pd.read_csv("Resources/crypto_data.csv", index_col=0)
crypto_df.head(10)
```

Out[7]:

	CoinName	Algorithm	IsTrading	ProofType	TotalCoinsMined	TotalCoinSupply
42	42 Coin	Scrypt	True	PoW/PoS	4.199995e+01	42
365	365Coin	X11	True	PoW/PoS	NaN	2300000000
404	404Coin	Scrypt	True	PoW/PoS	1.055185e+09	532000000
611	SixEleven	SHA-256	True	PoW	NaN	611000
808	808	SHA-256	True	PoW/PoS	0.000000e+00	0
1337	EliteCoin	X13	True	PoW/PoS	2.927942e+10	314159265359
2015	2015 coin	X11	True	PoW/PoS	NaN	0
BTC	Bitcoin	SHA-256	True	PoW	1.792718e+07	21000000
ETH	Ethereum	Ethash	True	PoW	1.076842e+08	0
LTC	Litecoin	Scrypt	True	PoW	6.303924e+07	84000000

In [8]:
```python
crypto_df.shape
```

Out[8]:
```
(1252, 6)
```

In [9]:
```python
# Keep all the cryptocurrencies that are being traded.
crypto_df_traded = crypto_df[crypto_df['IsTrading'] == True]
crypto_df_traded.head()
```

Out[9]:

	CoinName	Algorithm	IsTrading	ProofType	TotalCoinsMined	TotalCoinSupply
42	42 Coin	Scrypt	True	PoW/PoS	4.199995e+01	42
365	365Coin	X11	True	PoW/PoS	NaN	2300000000
404	404Coin	Scrypt	True	PoW/PoS	1.055185e+09	532000000
611	SixEleven	SHA-256	True	PoW	NaN	611000
808	808	SHA-256	True	PoW/PoS	0.000000e+00	0

In [10]:
```python
crypto_df_traded.shape
```

Out[10]:
```
(1144, 6)
```

In [11]:
```python
# Keep all the cryptocurrencies that have a working algorithm.
crypto_df_traded_work_algo = crypto_df_traded[crypto_df_traded['Algorithm'].notnull()]
```

In [12]:
```python
crypto_df_traded_work_algo
```

Out[12]:

	CoinName	Algorithm	IsTrading	ProofType	TotalCoinsMined	TotalCoinSupply
42	42 Coin	Scrypt	True	PoW/PoS	4.199995e+01	42
365	365Coin	X11	True	PoW/PoS	NaN	2300000000
404	404Coin	Scrypt	True	PoW/PoS	1.055185e+09	532000000
611	SixEleven	SHA-256	True	PoW	NaN	611000
808	808	SHA-256	True	PoW/PoS	0.000000e+00	0
...	...	...	...	...	...	...
SERO	Super Zero	Ethash	True	PoW	NaN	1000000000
UOS	UOS	SHA-256	True	DPoI	NaN	1000000000
BDX	Beldex	CryptoNight	True	PoW	9.802226e+08	1400222610
ZEN	Horizen	Equihash	True	PoW	7.296538e+06	21000000
XBC	BitcoinPlus	Scrypt	True	PoS	1.283270e+05	1000000

1144 rows × 6 columns

In [14]:
```python
# Remove the "IsTrading" column.
crypto_df_drop_IsTrading = crypto_df_traded_work_algo.drop(['IsTrading'], axis=1)
crypto_df_drop_IsTrading
```

Out[14]:

	CoinName	Algorithm	ProofType	TotalCoinsMined	TotalCoinSupply
42	42 Coin	Scrypt	PoW/PoS	4.199995e+01	42
365	365Coin	X11	PoW/PoS	NaN	2300000000
404	404Coin	Scrypt	PoW/PoS	1.055185e+09	532000000
611	SixEleven	SHA-256	PoW	NaN	611000
808	808	SHA-256	PoW/PoS	0.000000e+00	0
...	...	...	...	...	...
SERO	Super Zero	Ethash	PoW	NaN	1000000000
UOS	UOS	SHA-256	DPoI	NaN	1000000000
BDX	Beldex	CryptoNight	PoW	9.802226e+08	1400222610
ZEN	Horizen	Equihash	PoW	7.296538e+06	21000000
XBC	BitcoinPlus	Scrypt	PoS	1.283270e+05	1000000

1144 rows × 5 columns

In [15]:
```python
crypto_df_drop_null = crypto_df_drop_IsTrading.dropna()
crypto_df_drop_null
```

Out[15]:

	CoinName	Algorithm	ProofType	TotalCoinsMined	TotalCoinSupply
42	42 Coin	Scrypt	PoW/PoS	4.199995e+01	42
404	404Coin	Scrypt	PoW/PoS	1.055185e+09	532000000
808	808	SHA-256	PoW/PoS	0.000000e+00	0
1337	EliteCoin	X13	PoW/PoS	2.927942e+10	314159265359
BTC	Bitcoin	SHA-256	PoW	1.792718e+07	21000000
...	...	...	...	...	...
ZEPH	ZEPHYR	SHA-256	DPoS	2.000000e+09	2000000000
GAP	Gapcoin	Scrypt	PoW/PoS	1.493105e+07	250000000
BDX	Beldex	CryptoNight	PoW	9.802226e+08	1400222610
ZEN	Horizen	Equihash	PoW	7.296538e+06	21000000
XBC	BitcoinPlus	Scrypt	PoS	1.283270e+05	1000000

685 rows × 5 columns

In [16]:
```python
# Keep the rows where coins are mined.
crypto_df_coins_mined = crypto_df_drop_null[crypto_df_drop_null['TotalCoinsMined'] >= 1]
crypto_df_coins_mined
```

Out[16]:

	CoinName	Algorithm	ProofType	TotalCoinsMined	TotalCoinSupply
42	42 Coin	Scrypt	PoW/PoS	4.199995e+01	42
404	404Coin	Scrypt	PoW/PoS	1.055185e+09	532000000
1337	EliteCoin	X13	PoW/PoS	2.927942e+10	314159265359
BTC	Bitcoin	SHA-256	PoW	1.792718e+07	21000000
ETH	Ethereum	Ethash	PoW	1.076842e+08	0
...	...	...	...	...	...
ZEPH	ZEPHYR	SHA-256	DPoS	2.000000e+09	2000000000
GAP	Gapcoin	Scrypt	PoW/PoS	1.493105e+07	250000000
BDX	Beldex	CryptoNight	PoW	9.802226e+08	1400222610
ZEN	Horizen	Equihash	PoW	7.296538e+06	21000000
XBC	BitcoinPlus	Scrypt	PoS	1.283270e+05	1000000

532 rows × 5 columns

In [17]:
```python
# Create a new DataFrame that holds only the cryptocurrencies names.
crypto_df_names = pd.DataFrame(crypto_df_coins_mined['CoinName'])
crypto_df_names
```

Out[17]:

	CoinName
42	42 Coin
404	404Coin
1337	EliteCoin
BTC	Bitcoin
ETH	Ethereum
...	...
ZEPH	ZEPHYR
GAP	Gapcoin
BDX	Beldex
ZEN	Horizen
XBC	BitcoinPlus

532 rows × 1 columns

In [18]:
```python
# Drop the 'CoinName' column since it's not going to be used on the clustering algorithm.
crypto_df_drop_CoinName = crypto_df_coins_mined.drop(['CoinName'], axis=1)
crypto_df_drop_CoinName
```

Out[18]:

	Algorithm	ProofType	TotalCoinsMined	TotalCoinSupply
42	Scrypt	PoW/PoS	4.199995e+01	42
404	Scrypt	PoW/PoS	1.055185e+09	532000000
1337	X13	PoW/PoS	2.927942e+10	314159265359
BTC	SHA-256	PoW	1.792718e+07	21000000
ETH	Ethash	PoW	1.076842e+08	0
...	...	...	...	...
ZEPH	SHA-256	DPoS	2.000000e+09	2000000000
GAP	Scrypt	PoW/PoS	1.493105e+07	250000000
BDX	CryptoNight	PoW	9.802226e+08	1400222610
ZEN	Equihash	PoW	7.296538e+06	21000000
XBC	Scrypt	PoS	1.283270e+05	1000000

532 rows × 4 columns

In [19]:
```python
# Use get_dummies() to create variables for text features.
X = pd.get_dummies(crypto_df_drop_CoinName, columns=['Algorithm','ProofType'])
X
```

Out[19]:

	TotalCoinsMined	TotalCoinSupply	Algorithm_1GB AES Pattern Search	Algorithm_536	Algorithm_Argon2d	Algorithm_BLAKE256	Algorithm_Blake
42	4.199995e+01	42	0	0	0	0	0
404	1.055185e+09	532000000	0	0	0	0	0
1337	2.927942e+10	314159265359	0	0	0	0	0
BTC	1.792718e+07	21000000	0	0	0	0	0
ETH	1.076842e+08	0	0	0	0	0	0
...	...	...	...	...	...	...	...
ZEPH	2.000000e+09	2000000000	0	0	0	0	0
GAP	1.493105e+07	250000000	0	0	0	0	0
BDX	9.802226e+08	1400222610	0	0	0	0	0
ZEN	7.296538e+06	21000000	0	0	0	0	0
XBC	1.283270e+05	1000000	0	0	0	0	0

532 rows × 98 columns

In [20]:
```python
# Standardize the data with StandardScaler().
X_scaled = StandardScaler().fit_transform(X)
X_scaled
```

Out[20]:
```
array([[-0.11710817, -0.1528703 , -0.0433963 , ..., -0.0433963 ,
        -0.0433963 , -0.0433963 ],
       [-0.09396955, -0.145009  , -0.0433963 , ..., -0.0433963 ,
        -0.0433963 , -0.0433963 ],
       [ 0.52494561,  4.48942416, -0.0433963 , ..., -0.0433963 ,
        -0.0433963 , -0.0433963 ],
       ...,
       [-0.09561336, -0.13217937, -0.0433963 , ..., -0.0433963 ,
        -0.0433963 , -0.0433963 ],
       [-0.11694817, -0.15255998, -0.0433963 , ..., -0.0433963 ,
        -0.0433963 , -0.0433963 ],
       [-0.11710536, -0.15285552, -0.0433963 , ..., -0.0433963 ,
        -0.0433963 , -0.0433963 ]])
```

Deliverable 2: Reducing Data Dimensions Using PCA

In [24]:
```python
# Using PCA to reduce dimension to three principal components.
pca = PCA(n_components=3)
crypto_pca = pca.fit_transform(X_scaled)
```

In [25]:
```python
crypto_pca
```

Out[25]:
```
array([[-0.34037704,  1.02605335, -0.46499479],
       [-0.32371468,  1.0262267 , -0.46545608],
       [ 2.3102145 ,  1.64478026, -0.57032548],
       ...,
       [ 0.32839027, -2.3299913 ,  0.3420979 ],
       [-0.14082619, -2.06908111,  0.36593453],
       [-0.29612192,  0.81332633, -0.18243945]])
```

In [29]:
```python
# Create a DataFrame with the three principal components.
df_crypto_pca = pd.DataFrame(
    data=crypto_pca,
    columns=["principal component 1", "principal component 2", "principal component 3"], index=X.index
)

df_crypto_pca.head(10)
```

Out[29]:

	principal component 1	principal component 2	principal component 3
42	-0.340377	1.026053	-0.464995
404	-0.323715	1.026227	-0.465456
1337	2.310215	1.644780	-0.570325
BTC	-0.142496	-1.329328	0.145082
ETH	-0.146645	-2.014620	0.312857
LTC	-0.171284	-1.130821	0.013700
DASH	-0.394653	1.266771	-0.473432
XMR	-0.150164	-2.227637	0.294874
ETC	-0.145086	-2.014721	0.312834
ZEC	-0.140825	-2.069081	0.365934

In [30]:
```python
df_crypto_pca.shape
```

Out[30]:
```
(532, 3)
```

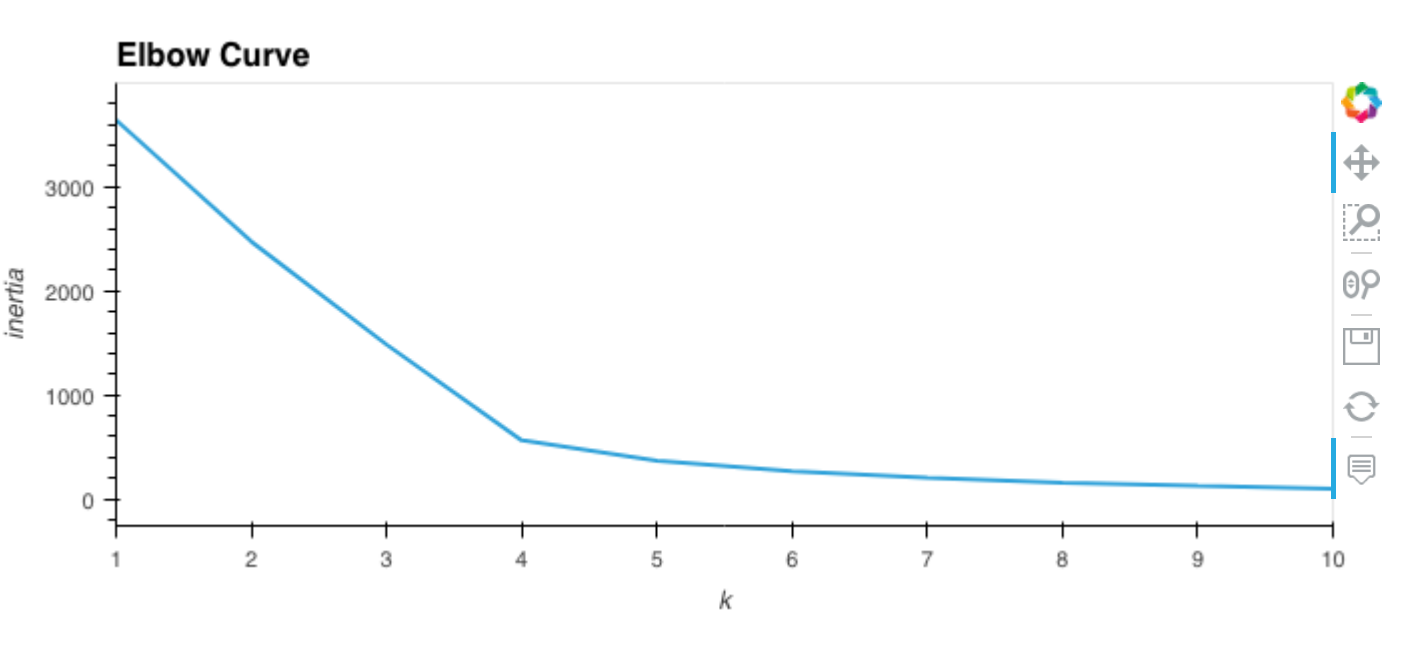
Deliverable 3: Clustering Crytocurrencies Using K-Means

Finding the Best Value for `k` Using the Elbow Curve

In [31]:
```python
inertia = []
k = list(range(1, 11))

for i in k:
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(df_crypto_pca)
    inertia.append(km.inertia_)
```

In [32]:
```python
# Create an elbow curve to find the best value for K.
elbow_data = {"k": k, "inertia": inertia}
df_elbow = pd.DataFrame(elbow_data)
df_elbow.hvplot.line(x="k", y="inertia", title="Elbow Curve", xticks=k)
```

Out[32]:



Running K-Means with `k=4`

In [33]:
```python
# Initialize the K-Means model.
model = KMeans(n_clusters=4, random_state=1)

# Fit the model
model.fit(df_crypto_pca)

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

Out[33]:
```
array([1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1,
       1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 3, 3, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 2, 1, 1,
       1, 0, 0, 1], dtype=int32)
```

In [34]:
```python
# Create a new DataFrame including predicted clusters and cryptocurrencies features.
# Concatentate the crypto_df and pcs_df DataFrames on the same columns.
clustered_df = pd.concat([crypto_df_drop_CoinName,df_crypto_pca], axis= 1, join="inner")

#  Add a new column, "CoinName" to the clustered_df DataFrame that holds the names of the cryptocurrencies.
clustered_df = pd.concat([clustered_df,crypto_df_names], axis= 1, join="inner")

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

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

```
(532, 9)
```

Out[34]:

	Algorithm	ProofType	TotalCoinsMined	TotalCoinSupply	principal component 1	principal component 2	principal component 3	CoinName	Class
42	Scrypt	PoW/PoS	4.199995e+01	42	-0.340377	1.026053	-0.464995	42 Coin	1
404	Scrypt	PoW/PoS	1.055185e+09	532000000	-0.323715	1.026227	-0.465456	404Coin	1
1337	X13	PoW/PoS	2.927942e+10	314159265359	2.310215	1.644780	-0.570325	EliteCoin	1
BTC	SHA-256	PoW	1.792718e+07	21000000	-0.142496	-1.329328	0.145082	Bitcoin	0
ETH	Ethash	PoW	1.076842e+08	0	-0.146645	-2.014620	0.312857	Ethereum	0
LTC	Scrypt	PoW	6.303924e+07	84000000	-0.171284	-1.130821	0.013700	Litecoin	0
DASH	X11	PoW/PoS	9.031294e+06	22000000	-0.394653	1.266771	-0.473432	Dash	1
XMR	CryptoNight-V7	PoW	1.720114e+07	0	-0.150164	-2.227637	0.294874	Monero	0
ETC	Ethash	PoW	1.133597e+08	210000000	-0.145086	-2.014721	0.312834	Ethereum Classic	0
ZEC	Equihash	PoW	7.383056e+06	21000000	-0.140825	-2.069081	0.365934	ZCash	0

Deliverable 4: Visualizing Cryptocurrencies Results

3D-Scatter with Clusters

In [35]:
```python
# Creating a 3D-Scatter with the PCA data and the clusters
fig = px.scatter_3d(
    clustered_df,
    x='principal component 1',
    y='principal component 2',
    z='principal component 3',
    color='Class',
    symbol='Class',
    hover_name= 'CoinName',
    hover_data= ['Algorithm'],
    width=800,
)
fig.update_layout(legend=dict(x=0, y=1))
fig.show()
```

In [36]:
```python
# Create a table with tradable cryptocurrencies.
clustered_df.hvplot.table(
    columns=['CoinName', 'Algorithm', 'ProofType','TotalCoinSupply','TotalCoinsMined', 'Class'],
    sortable=True,
    selectable=True
)
```

Out[36]:

#	CoinName	Algorithm	ProofType	TotalCoinSupply	TotalCoinsMined	Class
0	42 Coin	Scrypt	PoW/PoS	42	41.999954	1
1	404Coin	Scrypt	PoW/PoS	532000000	1,055,184,902.04	1
2	EliteCoin	X13	PoW/PoS	314159265359	29,279,424,622.5027(	1
3	Bitcoin	SHA-256	PoW	21000000	17,927,175.0	0
4	Ethereum	Ethash	PoW	0	107,684,222.6865	0
5	Litecoin	Scrypt	PoW	84000000	63,039,243.300005	0
6	Dash	X11	PoW/PoS	22000000	9,031,294.375634	1
7	Monero	CryptoNight-V7	PoW	0	17,201,143.144913	0
8	Ethereum Classic	Ethash	PoW	210000000	113,359,703.0	0
9	ZCash	Equihash	PoW	21000000	7,383,056.25	0
10	Bitshares	SHA-512	PoS	3600570502	2,741,570,000.0	1

In [37]:
```python
# Print the total number of tradable cryptocurrencies.
print(f'There are {len(clustered_df)}  tradable cryptocurrencies.')
```

```
There are 532  tradable cryptocurrencies.
```

In [38]:
```python
# Scaling data to create the scatter plot with tradable cryptocurrencies.
clustered_scaled = MinMaxScaler().fit_transform(clustered_df[['TotalCoinSupply', 'TotalCoinsMined']])
clustered_scaled
```

Out[38]:
```
array([[4.20000000e-11, 0.00000000e+00],
       [5.32000000e-04, 1.06585544e-03],
       [3.14159265e-01, 2.95755135e-02],
       ...,
       [1.40022261e-03, 9.90135079e-04],
       [2.10000000e-05, 7.37028150e-06],
       [1.00000000e-06, 1.29582282e-07]])
```

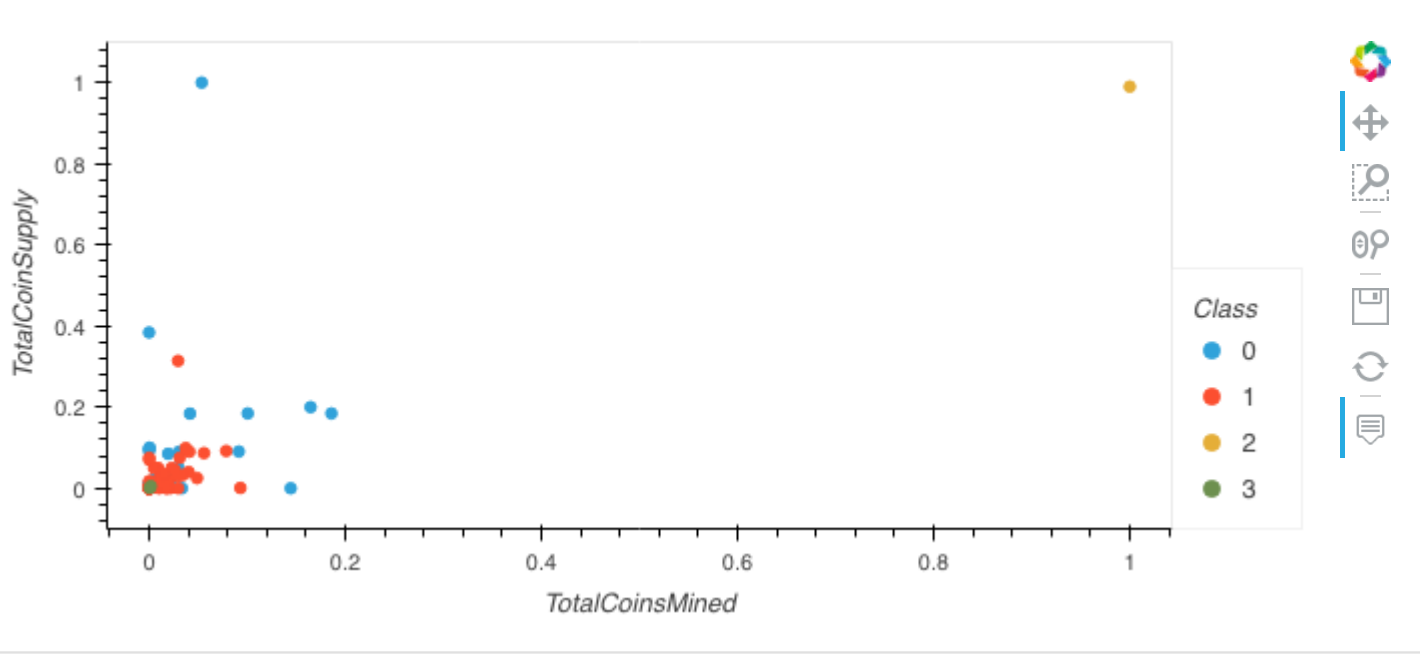
In [39]:
```python
# Create a new DataFrame that has the scaled data with the clustered_df DataFrame index.
plot_df = pd.DataFrame(
    data=clustered_scaled, columns=["TotalCoinSupply", "TotalCoinsMined"], index=clustered_df.index.copy()
)

# Add the "CoinName" column from the clustered_df DataFrame to the new DataFrame.
plot_df['CoinName'] = clustered_df['CoinName']

# Add the "Class" column from the clustered_df DataFrame to the new DataFrame.
plot_df['Class'] = clustered_df['Class']

plot_df.head(10)
```

Out[39]:

	TotalCoinSupply	TotalCoinsMined	CoinName	Class
42	4.200000e-11	0.000000	42 Coin	1
404	5.320000e-04	0.001066	404Coin	1
1337	3.141593e-01	0.029576	EliteCoin	1
BTC	2.100000e-05	0.000018	Bitcoin	0
ETH	0.000000e+00	0.000109	Ethereum	0
LTC	8.400000e-05	0.000064	Litecoin	0
DASH	2.200000e-05	0.000009	Dash	1
XMR	0.000000e+00	0.000017	Monero	0
ETC	2.100000e-04	0.000115	Ethereum Classic	0
ZEC	2.100000e-05	0.000007	ZCash	0

In [40]:
```python
# Create a hvplot.scatter plot using x="TotalCoinsMined" and y="TotalCoinSupply".
plot_df.hvplot.scatter(
    x="TotalCoinsMined",
    y="TotalCoinSupply",
    by="Class",
    hover_cols="CoinName"
)
```

Out[40]:



In [ ]: