Clustering Crypto

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

Deliverable 1: Preprocessing the Data for PCA

In [2]:
```
# Load the crypto_data.csv dataset.
crypto_df = pd.read_csv("crypto_data.csv")
crypto_df.sample(10)
```

Out[2]:

	Unnamed: 0	CoinName	Algorithm	IsTrading	ProofType	TotalCoinsMined	TotalCoinSupply
1020	LTCP	LitecoinPro	Scrypt	True	PoW	NaN	17500000
70	COMM	Community Coin	Scrypt	True	PoW/PoS	NaN	1000000000
628	MAC	MachineCoin	Time Travel	True	PoW	NaN	35000000
237	NKT	NakomotoDark	X11	True	PoW/PoS	NaN	0
1049	RYO	Ryo	Cryptonight-GPU	True	PoW	4.890897e+06	88188888
873	SPK	SparksPay	NeoScrypt	True	PoW	7.847417e+06	21000000
825	WSC	WiserCoin	Scrypt	False	PoW	NaN	22105263
927	LOT	LottoCoin	Scrypt	True	PoW	1.449101e+10	18406979840
1041	BSPM	Bitcoin Supreme	Scrypt	True	PoS	NaN	21000000
436	MUDRA	MudraCoin	X13	True	PoS	5.000000e+06	200000000

In [3]:
```
# Keep all the cryptocurrencies that are being traded.
traded_df = crypto_df.loc[crypto_df["IsTrading"] == True]
traded_df.set_index('Unnamed: 0', inplace=True)
traded_df.index.name = None
traded_df.head(10)
```

Out[3]:

	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 [4]:
```
# Keep all the cryptocurrencies that have a working algorithm.
algorithm_df = traded_df[traded_df["Algorithm"].notnull()]
algorithm_df.head(10)
```

Out[4]:

	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 [5]:
```
# Remove the "IsTrading" column.
removed_df = algorithm_df.drop(["IsTrading"], axis=1)
removed_df.head(10)
```

Out[5]:

	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
1337	EliteCoin	X13	PoW/PoS	2.927942e+10	314159265359
2015	2015 coin	X11	PoW/PoS	NaN	0
BTC	Bitcoin	SHA-256	PoW	1.792718e+07	21000000
ETH	Ethereum	Ethash	PoW	1.076842e+08	0
LTC	Litecoin	Scrypt	PoW	6.303924e+07	84000000

In [6]:
```
# Remove rows that have at least 1 null value.
null_rows_df = removed_df.dropna()
null_rows_df.head(10)
```

Out[6]:

	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
ETH	Ethereum	Ethash	PoW	1.076842e+08	0
LTC	Litecoin	Scrypt	PoW	6.303924e+07	84000000
DASH	Dash	X11	PoW/PoS	9.031294e+06	22000000
XMR	Monero	CryptoNight-V7	PoW	1.720114e+07	0
ETC	Ethereum Classic	Ethash	PoW	1.133597e+08	210000000

In [7]:
```
# Keep the rows where coins are mined.
mined_df = null_rows_df[null_rows_df["TotalCoinsMined"] > 0]
mined_df.head(10)
```

Out[7]:

	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
LTC	Litecoin	Scrypt	PoW	6.303924e+07	84000000
DASH	Dash	X11	PoW/PoS	9.031294e+06	22000000
XMR	Monero	CryptoNight-V7	PoW	1.720114e+07	0
ETC	Ethereum Classic	Ethash	PoW	1.133597e+08	210000000
ZEC	ZCash	Equihash	PoW	7.383056e+06	21000000

In [8]:
```
# Create a new DataFrame that holds only the cryptocurrencies names.
crypto_names_df = mined_df[["CoinName"]]
crypto_names_df
```

Out[8]:

	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 [9]:
```
# Drop the 'CoinName' column since it's not going to be used on the clustering algorithm.
clean_df = mined_df.drop(["CoinName"], axis=1)
clean_df.head(10)
```

Out[9]:

	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
LTC	Scrypt	PoW	6.303924e+07	84000000
DASH	X11	PoW/PoS	9.031294e+06	22000000
XMR	CryptoNight-V7	PoW	1.720114e+07	0
ETC	Ethash	PoW	1.133597e+08	210000000
ZEC	Equihash	PoW	7.383056e+06	21000000

In [10]:
```
# Use get_dummies() to create variables for text features.
## X is variable
## Features = algorithms and prooftype
X = pd.get_dummies(clean_df, columns=["Algorithm", "ProofType"])
X.head(10)
```

Out[10]:

	TotalCoinsMined	TotalCoinSupply	Algorithm_1GB AES Pattern Search	Algorithm_536	Algorithm_Argon2d	Algorithm_BLAKE256	Algorithm_Blake	Algorithm_Blake2S	Algorithm_Blake2b	Algorithm_C11	...	ProofType_PoW/PoS	ProofTyp
42	4.199995e+01	42	0	0	0	0	0	0	0	0	...	1	
404	1.055185e+09	532000000	0	0	0	0	0	0	0	0	...	1	
1337	2.927942e+10	314159265359	0	0	0	0	0	0	0	0	...	1	
BTC	1.792718e+07	21000000	0	0	0	0	0	0	0	0	...	0	
ETH	1.076842e+08	0	0	0	0	0	0	0	0	0	...	0	
LTC	6.303924e+07	84000000	0	0	0	0	0	0	0	0	...	0	
DASH	9.031294e+06	22000000	0	0	0	0	0	0	0	0	...	1	
XMR	1.720114e+07	0	0	0	0	0	0	0	0	0	...	0	
ETC	1.133597e+08	210000000	0	0	0	0	0	0	0	0	...	0	
ZEC	7.383056e+06	21000000	0	0	0	0	0	0	0	0	...	0	

10 rows × 98 columns

In [11]:
```
# Standardize the data with StandardScaler()
scaled = StandardScaler().fit_transform(X)
scaled
```

Out[11]:
```
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 [12]:
```
# Using PCA to reduce dimension to three principal components
## components = 3
pca = PCA(n_components=3)
crypto_pca = pca.fit_transform(scaled)
crypto_pca
```

Out[12]:
```
array([[-0.33866189,  0.83045622, -0.38746672],
       [-0.32197207,  0.83024051, -0.38773548],
       [ 2.33433133,  1.28925273, -0.36102348],
       ...,
       [ 0.32231716, -2.04370271,  0.27349153],
       [-0.14656277, -1.76338209,  0.24309206],
       [-0.30432054,  0.7247283 , -0.23465375]])
```

In [13]:
```
# Create a DataFrame with the three principal components
pca_df = pd.DataFrame(
    data=crypto_pca, columns=["PC 1", "PC 2", "PC 3"], index=clean_df.index
)
pca_df.head(10)
```

Out[13]:

	PC 1	PC 2	PC 3
42	-0.338662	0.830456	-0.387467
404	-0.321972	0.830241	-0.387735
1337	2.334331	1.289253	-0.361023
BTC	-0.135413	-1.105673	0.149984
ETH	-0.149681	-1.765713	0.280109
LTC	-0.182488	-0.989514	-0.041826
DASH	-0.382843	0.926315	-0.272051
XMR	-0.165349	-1.924613	0.267573
ETC	-0.148119	-1.765851	0.280096
ZEC	-0.146562	-1.763382	0.243092

Deliverable 3: Clustering Crytocurrencies Using K-Means

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

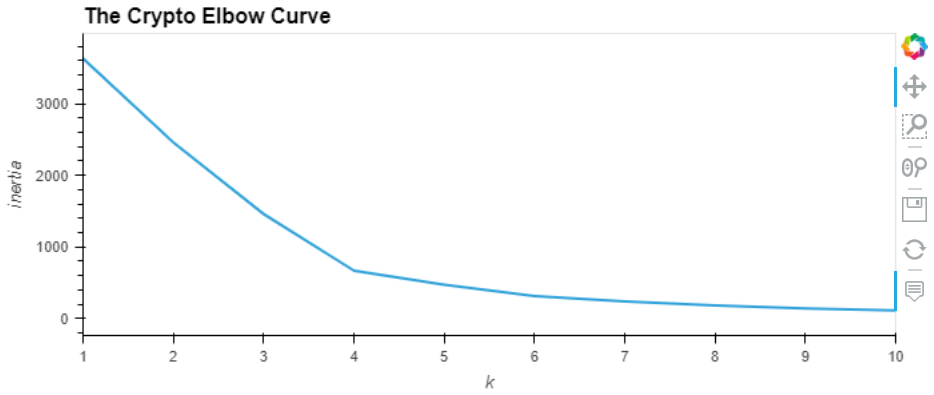
In [14]:
```
# Create an elbow curve to find the best value for K.
# Create variables
inertia = []
k = list(range(1,11))
```

```
# For loop using inertia
for i in k:
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(pca_df)
    inertia.append(km.inertia_)

# Elbow curve code
elbow_curve = {"k": k, "inertia": inertia}
elbow_curve_df = pd.DataFrame(elbow_curve)
elbow_curve_df.hvplot.line(x="k", y="inertia", xticks=k, title="The Crypto Elbow Curve")
```

```
C:\Users\saman\anaconda3\envs\mlenv\lib\site-packages\sklearn\cluster\_kmeans.py:1037: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=3.
  "KMeans is known to have a memory leak on Windows "
```

Out[14]:



Running K-Means with `k=4`

In [15]:

```
# Initialize the K-Means model.
model = KMeans(n_clusters=4, random_state=0)

# Fit the model
model.fit(pca_df)

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

Out[15]:
```
array([2, 2, 2, 0, 0, 0, 2, 0, 0, 0, 2, 0, 2, 2, 0, 2, 0, 0, 2, 2, 0, 0,
       0, 0, 0, 2, 0, 0, 0, 2, 0, 2, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 2, 2,
       0, 0, 0, 0, 0, 2, 2, 0, 2, 0, 0, 0, 0, 2, 0, 0, 2, 0, 2, 2, 2, 0,
       0, 0, 2, 2, 2, 2, 2, 0, 0, 0, 2, 2, 0, 2, 0, 2, 2, 0, 0, 0, 0, 2,
       2, 0, 2, 0, 0, 2, 2, 0, 2, 2, 0, 0, 2, 2, 0, 2, 2, 0, 2, 0, 2, 0,
       2, 0, 2, 2, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 2,
       0, 2, 0, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 0, 0, 2, 2, 0, 2, 0, 2, 2,
       2, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2,
       2, 0, 2, 0, 2, 2, 0, 2, 0, 2, 2, 0, 2, 0, 2, 0, 2, 0, 2, 2, 2, 2,
       0, 2, 2, 2, 2, 2, 0, 0, 2, 2, 0, 0, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2,
       2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2,
       2, 0, 2, 0, 0, 2, 0, 0, 2, 0, 2, 2, 2, 0, 2, 2, 0, 2, 2, 2, 2, 2,
       2, 2, 0, 2, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 0, 0, 0, 2, 0, 2, 2, 0,
       2, 0, 0, 0, 2, 0, 2, 0, 0, 0, 2, 0, 2, 2, 0, 0, 2, 0, 0, 0, 0, 0,
       0, 0, 2, 2, 0, 2, 2, 2, 0, 2, 0, 2, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2,
       0, 2, 2, 2, 0, 0, 0, 0, 2, 2, 2, 2, 0, 2, 0, 0, 0, 2, 2, 0, 0, 2,
       2, 0, 2, 0, 0, 0, 2, 0, 0, 2, 2, 2, 0, 0, 0, 2, 2, 2, 0, 0, 2, 0,
       0, 0, 0, 2, 1, 2, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 0, 0, 0, 2, 2, 2,
       0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 0, 0, 2, 0, 2, 0, 0, 0, 0,
       2, 2, 0, 2, 0, 2, 2, 2, 2, 2, 2, 0, 0, 0, 2, 2, 2, 2, 2, 2, 0, 2,
       0, 0, 0, 0, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 0, 2, 0, 2, 0, 0, 2, 2,
       0, 2, 0, 0, 2, 0, 0, 2, 0, 2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0,
       2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 0, 2, 0, 2,
       2, 2, 0, 0, 2, 2, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 2, 2, 3, 2, 2,
       2, 0, 0, 2])
```

In [16]:
```
# Create a new DataFrame including predicted clusters and cryptocurrencies features
# Concatentate the crypto_df and pcs_df DataFrames on the same columns
# Add a new column, "CoinName" to the clustered_df DataFrame that holds the names of the cryptocurrencies
# Add a new column, "Class" to the clustered_df DataFrame that holds the predictions
clusters_df = pd.DataFrame({
    "Algorithm": clean_df["Algorithm"],
    "ProofType": clean_df["ProofType"],
    "TotalCoinsMined": clean_df["TotalCoinsMined"],
    "TotalCoinSupply": clean_df["TotalCoinSupply"],
    "PC 1": pca_df["PC 1"],
    "PC 2": pca_df["PC 2"],
    "PC 3": pca_df["PC 3"],
    "CoinName": crypto_names_df["CoinName"],
    "Class": predictions
})

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

```
(532, 9)
```

Out[16]:

	Algorithm	ProofType	TotalCoinsMined	TotalCoinSupply	PC 1	PC 2	PC 3	CoinName	Class
42	Scrypt	PoW/PoS	4.199995e+01	42	-0.338662	0.830456	-0.387467	42 Coin	2
404	Scrypt	PoW/PoS	1.055185e+09	532000000	-0.321972	0.830241	-0.387735	404Coin	2
1337	X13	PoW/PoS	2.927942e+10	314159265359	2.334331	1.289253	-0.361023	EliteCoin	2
BTC	SHA-256	PoW	1.792718e+07	21000000	-0.135413	-1.105673	0.149984	Bitcoin	0
ETH	Ethash	PoW	1.076842e+08	0	-0.149681	-1.765713	0.280109	Ethereum	0
LTC	Scrypt	PoW	6.303924e+07	84000000	-0.182488	-0.989514	-0.041826	Litecoin	0
DASH	X11	PoW/PoS	9.031294e+06	22000000	-0.382843	0.926315	-0.272051	Dash	2
XMR	CryptoNight-V7	PoW	1.720114e+07	0	-0.165349	-1.924613	0.267573	Monero	0
ETC	Ethash	PoW	1.133597e+08	210000000	-0.148119	-1.765851	0.280096	Ethereum Classic	0
ZEC	Equihash	PoW	7.383056e+06	21000000	-0.146562	-1.763382	0.243092	ZCash	0

Deliverable 4: Visualizing Cryptocurrencies Results

3D-Scatter with Clusters

In [17]:
```
# Creating a 3D-Scatter with the PCA data and the clusters
fig = px.scatter_3d(
    clusters_df,
    x="PC 1",
    y="PC 2",
    z="PC 3",
    hover_data=["Algorithm"],
    hover_name="CoinName",
    color="Class",
    symbol="Class",
    width=800
)

fig.update_layout(legend=dict(x=0,y=1))
fig.show
```

Out[17]:
```
<bound method BaseFigure.show of Figure({
    'data': [{'customdata': array([['Scrypt'],
                                   ['Scrypt'],
                                   ['X13'],
                                   ...,
                                   ['SHA-256'],
                                   ['Scrypt'],
                                   ['Scrypt']], dtype=object),
              'hovertemplate': ('<b>%{hovertext}</b><br><br>Cla' ... '{customdata[0]}<extra></extra>'),
              'hovertext': array(['42 Coin', '404Coin', 'EliteCoin', ..., 'ZEPHYR', 'Gapcoin',
                                  'BitcoinPlus'], dtype=object),
              'legendgroup': '2',
              'marker': {'color': array([2, 2, 2, ..., 2, 2, 2]),
                         'coloraxis': 'coloraxis',
                         'symbol': 'circle'},
              'mode': 'markers',
              'name': '2',
              'scene': 'scene',
              'showlegend': True,
              'type': 'scatter3d',
              'x': array([-0.33866189, -0.32197207,  2.33433133, ...,  2.46479414, -0.33670313,
                          -0.30432054]),
              'y': array([0.83045622, 0.83024051, 1.28925273, ..., 1.30188594, 0.83029285,
                          0.7247283 ]),
              'z': array([-0.38746672, -0.38773548, -0.36102348, ...,  0.06981759, -0.38748449,
                          -0.23465375])},
             {'customdata': array([['SHA-256'],
                                   ['Ethash'],
                                   ['Scrypt'],
                                   ...,
                                   ['Ethash'],
                                   ['CryptoNight'],
                                   ['Equihash']], dtype=object),
              'hovertemplate': ('<b>%{hovertext}</b><br><br>Cla' ... '{customdata[0]}<extra></extra>'),
              'hovertext': array(['Bitcoin', 'Ethereum', 'Litecoin', ..., 'Reality Clash', 'Beldex',
                                  'Horizen'], dtype=object),
              'legendgroup': '0',
              'marker': {'color': array([0, 0, 0, ..., 0, 0, 0]),
                         'coloraxis': 'coloraxis',
                         'symbol': 'diamond'},
              'mode': 'markers',
              'name': '0',
              'scene': 'scene',
              'showlegend': True,
              'type': 'scatter3d',
              'x': array([-0.1354134 , -0.14968072, -0.18248803, ..., -0.15052467,  0.32231716,
                          -0.14656277]),
              'y': array([-1.10567337, -1.76571302, -0.98951369, ..., -1.76573993, -2.04370271,
                          -1.76338209]),
              'z': array([ 0.1499844 ,  0.2801093 , -0.04182563, ...,  0.28012667,  0.27349153,
                           0.24309206])},
             {'customdata': array([['Proof-of-BibleHash']], dtype=object),
              'hovertemplate': ('<b>%{hovertext}</b><br><br>Cla' ... '{customdata[0]}<extra></extra>'),
              'hovertext': array(['BiblePay'], dtype=object),
              'legendgroup': '1',
              'marker': {'color': array([1]), 'coloraxis': 'coloraxis', 'symbol': 'square'},
              'mode': 'markers',
              'name': '1',
              'scene': 'scene',
              'showlegend': True,
              'type': 'scatter3d',
              'x': array([-0.18720232]),
              'y': array([1.81106086]),
              'z': array([31.55298689])},
             {'customdata': array([['TRC10']], dtype=object),
              'hovertemplate': ('<b>%{hovertext}</b><br><br>Cla' ... '{customdata[0]}<extra></extra>'),
              'hovertext': array(['BitTorrent'], dtype=object),
              'legendgroup': '3',
```

```
                'marker': {'color': array([3]), 'coloraxis': 'coloraxis', 'symbol': 'x'},
                'mode': 'markers',
                'name': '3',
                'scene': 'scene',
                'showlegend': True,
                'type': 'scatter3d',
                'x': array([34.09375783]),
                'y': array([1.60042876]),
                'z': array([-0.45024235])}],
    'layout': {'coloraxis': {'colorbar': {'title': {'text': 'Class'}},
                             'colorscale': [[0.0, '#0d0887'], [0.1111111111111111,
                                            '#46039f'], [0.2222222222222222,
                                            '#7201a8'], [0.3333333333333333,
                                            '#9c179e'], [0.4444444444444444,
                                            '#bd3786'], [0.5555555555555556,
                                            '#d8576b'], [0.6666666666666666,
                                            '#ed7953'], [0.7777777777777778,
                                            '#fb9f3a'], [0.8888888888888888,
                                            '#fdca26'], [1.0, '#f0f921']]},
               'legend': {'title': {'text': 'Class'}, 'tracegroupgap': 0, 'x': 0, 'y': 1},
               'margin': {'t': 60},
               'scene': {'domain': {'x': [0.0, 1.0], 'y': [0.0, 1.0]},
                         'xaxis': {'title': {'text': 'PC 1'}},
                         'yaxis': {'title': {'text': 'PC 2'}},
                         'zaxis': {'title': {'text': 'PC 3'}}},
               'template': '...',
               'width': 800}
})>
```

In [18]:
```
# Create a table with tradable cryptocurrencies.
columns = ["CoinName", "Algorithm", "ProofType", "TotalCoinSupply", "TotalCoinsMined", "Class"]
clusters_df.hvplot.table(columns)
```

Out[18]:

#	CoinName	Algorithm	ProofType	TotalCoinSupply	TotalCoinsMined	Class
0	42 Coin	Scrypt	PoW/PoS	42	41.999954	2
1	404Coin	Scrypt	PoW/PoS	532000000	1,055,184,902.04	2
2	EliteCoin	X13	PoW/PoS	314159265359	29,279,424,622.5027(	2
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	2
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	2

In [19]:
```
# Print the total number of tradable cryptocurrencies.
print(f"There are {clusters_df.CoinName.size} tradeable cryptocurrencies.")
```

```
There are 532 tradeable cryptocurrencies.
```

In [20]:
```
# Scaling data to create the scatter plot with tradable cryptocurrencies.
scaler = MinMaxScaler(feature_range=(0,1))
scaler_cluster = scaler.fit_transform(clusters_df[["TotalCoinSupply", "TotalCoinsMined"]])
scaler_cluster
```

Out[20]:
```
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 [21]:
```
# Create a new DataFrame that has the scaled data with the clustered_df DataFrame index.
scaled_df = pd.DataFrame(scaler_cluster, columns=["TotalCoinSupply", "TotalCoinsMined"], index=clusters_df.index)

# Add the "CoinName" column from the clustered_df DataFrame to the new DataFrame.
# Add the "Class" column from the clustered_df DataFrame to the new DataFrame.
scaled_df = pd.DataFrame({
    "TotalCoinSupply": scaled_df["TotalCoinSupply"],
    "TotalCoinsMined": scaled_df["TotalCoinsMined"],
    "CoinName": clusters_df["CoinName"],
    "Class": clusters_df["Class"]
})


scaled_df.head(10)
```

Out[21]:

	TotalCoinSupply	TotalCoinsMined	CoinName	Class
42	4.200000e-11	0.000000	42 Coin	2
404	5.320000e-04	0.001066	404Coin	2
1337	3.141593e-01	0.029576	EliteCoin	2
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	2
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 [22]:
```
# Create a hvplot.scatter plot using x="TotalCoinsMined" and y="TotalCoinSupply".
scaled_df.hvplot(
    kind="scatter",
    x="TotalCoinsMined",
    y="TotalCoinSupply",
    by="Class",
    hover_cols=["CoinName"]
)
```

Out[22]: