Clustering Crypto

In [4]:
```
# 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 [5]:
```
# Load the crypto_data.csv dataset.
crypto_df = pd.read_csv('crypto_data.csv')
```

In [6]:
```
crypto_df
```

Out[6]:

	Unnamed: 0	CoinName	Algorithm	IsTrading	ProofType	TotalCoinsMined	TotalCoinSupply
0	42	42 Coin	Scrypt	True	PoW/PoS	4.199995e+01	42
1	365	365Coin	X11	True	PoW/PoS	NaN	2300000000
2	404	404Coin	Scrypt	True	PoW/PoS	1.055185e+09	532000000
3	611	SixEleven	SHA-256	True	PoW	NaN	611000
4	808	808	SHA-256	True	PoW/PoS	0.000000e+00	0
...	...	...	...	...	...	...	...
1247	XBC	BitcoinPlus	Scrypt	True	PoS	1.283270e+05	1000000
1248	DVTC	DivotyCoin	Scrypt	False	PoW/PoS	2.149121e+07	100000000
1249	GIOT	Giotto Coin	Scrypt	False	PoW/PoS	NaN	233100000
1250	OPSC	OpenSourceCoin	SHA-256	False	PoW/PoS	NaN	21000000
1251	PUNK	SteamPunk	PoS	False	PoS	NaN	40000000

1252 rows × 7 columns

In [7]:
```
# Keep all the cryptocurrencies that are being traded.
crypto_df = crypto_df[crypto_df.IsTrading.eq(True)]
crypto_df.shape
```

Out[7]:
```
(1144, 7)
```

In [8]:
```
# Keep all the cryptocurrencies that have a working algorithm.
pd.isna(crypto_df['Algorithm'])
```

Out[8]:
```
0       False
1       False
2       False
3       False
4       False
        ...
1243    False
1244    False
1245    False
1246    False
1247    False
Name: Algorithm, Length: 1144, dtype: bool
```

In [9]:
```
# Remove the "IsTrading" column.
crypto_df = crypto_df.drop(["IsTrading"],axis = 1)
```

In [10]:
```
# Remove rows that have at least 1 null value.
crypto_df = crypto_df.dropna(how='any',axis=0)
crypto_df
```

Out[10]:

	Unnamed: 0	CoinName	Algorithm	ProofType	TotalCoinsMined	TotalCoinSupply
0	42	42 Coin	Scrypt	PoW/PoS	4.199995e+01	42
2	404	404Coin	Scrypt	PoW/PoS	1.055185e+09	532000000
4	808	808	SHA-256	PoW/PoS	0.000000e+00	0
5	1337	EliteCoin	X13	PoW/PoS	2.927942e+10	314159265359
7	BTC	Bitcoin	SHA-256	PoW	1.792718e+07	21000000
...	...	...	...	...	...	...
1238	ZEPH	ZEPHYR	SHA-256	DPoS	2.000000e+09	2000000000
1242	GAP	Gapcoin	Scrypt	PoW/PoS	1.493105e+07	250000000
1245	BDX	Beldex	CryptoNight	PoW	9.802226e+08	1400222610
1246	ZEN	Horizen	Equihash	PoW	7.296538e+06	21000000
1247	XBC	BitcoinPlus	Scrypt	PoS	1.283270e+05	1000000

685 rows × 6 columns

In [11]:
```
# Keep the rows where coins are mined.
crypto_df = crypto_df[crypto_df.TotalCoinsMined > 0]
crypto_df
```

Out[11]:

	Unnamed: 0	CoinName	Algorithm	ProofType	TotalCoinsMined	TotalCoinSupply
0	42	42 Coin	Scrypt	PoW/PoS	4.199995e+01	42
2	404	404Coin	Scrypt	PoW/PoS	1.055185e+09	532000000
5	1337	EliteCoin	X13	PoW/PoS	2.927942e+10	314159265359
7	BTC	Bitcoin	SHA-256	PoW	1.792718e+07	21000000
8	ETH	Ethereum	Ethash	PoW	1.076842e+08	0
...	...	...	...	...	...	...
1238	ZEPH	ZEPHYR	SHA-256	DPoS	2.000000e+09	2000000000
1242	GAP	Gapcoin	Scrypt	PoW/PoS	1.493105e+07	250000000
1245	BDX	Beldex	CryptoNight	PoW	9.802226e+08	1400222610
1246	ZEN	Horizen	Equihash	PoW	7.296538e+06	21000000
1247	XBC	BitcoinPlus	Scrypt	PoS	1.283270e+05	1000000

532 rows × 6 columns

In [12]:
```
# Create a new DataFrame that holds only the cryptocurrencies names.
names = crypto_df.filter(['CoinName'], axis=1)
names
```

Out[12]:

	CoinName
0	42 Coin
2	404Coin
5	EliteCoin
7	Bitcoin
8	Ethereum
...	...
1238	ZEPHYR
1242	Gapcoin
1245	Beldex
1246	Horizen
1247	BitcoinPlus

532 rows × 1 columns

In [23]:
```
# Drop the 'CoinName' column since it's not going to be used on the cluster
crypto_df = crypto_df.drop(['CoinName'],axis = 1)
crypto_df
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/var/folders/y8/z25h4y997psc483tj59t6x_w0000gn/T/ipykernel_71728/3414334566.py in <module>
      1 # Drop the 'CoinName' column since it's not going to be used on the clustering algorithm.
----> 2 crypto_df = crypto_df.drop(['CoinName'],axis = 1)
      3 crypto_df

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
    309                     stacklevel=stacklevel,
    310                 )
--> 311             return func(*args, **kwargs)
    312
    313         return wrapper

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/pandas/core/frame.py in drop(self, labels, axis, index, columns, level, inplace, errors)
   4911             level=level,
   4912             inplace=inplace,
-> 4913             errors=errors,
   4914         )
   4915

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/pandas/core/generic.py in drop(self, labels, axis, index, columns, level, inplace, errors)
   4148         for axis, labels in axes.items():
   4149             if labels is not None:
-> 4150                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4151
   4152         if inplace:

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/pandas/core/generic.py in _drop_axis(self, labels, axis, level, errors)
   4183                 new_axis = axis.drop(labels, level=level, errors=errors)
   4184             else:
-> 4185                 new_axis = axis.drop(labels, errors=errors)
   4186             result = self.reindex(**{axis_name: new_axis})
   4187

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/pandas/core/indexes/base.py in drop(self, labels, errors)
   6015         if mask.any():
   6016             if errors != "ignore":
-> 6017                 raise KeyError(f"{labels[mask]} not found in axi
```

```
s")
   6018             indexer = indexer[~mask]
   6019         return self.delete(indexer)

KeyError: "['CoinName'] not found in axis"
```

In [13]:
```
crypto_df
```

Out[13]:

	Unnamed: 0	CoinName	Algorithm	ProofType	TotalCoinsMined	TotalCoinSupply
0	42	42 Coin	Scrypt	PoW/PoS	4.199995e+01	42
2	404	404Coin	Scrypt	PoW/PoS	1.055185e+09	532000000
5	1337	EliteCoin	X13	PoW/PoS	2.927942e+10	314159265359
7	BTC	Bitcoin	SHA-256	PoW	1.792718e+07	21000000
8	ETH	Ethereum	Ethash	PoW	1.076842e+08	0
...	...	...	...	...	...	...
1238	ZEPH	ZEPHYR	SHA-256	DPoS	2.000000e+09	2000000000
1242	GAP	Gapcoin	Scrypt	PoW/PoS	1.493105e+07	250000000
1245	BDX	Beldex	CryptoNight	PoW	9.802226e+08	1400222610
1246	ZEN	Horizen	Equihash	PoW	7.296538e+06	21000000
1247	XBC	BitcoinPlus	Scrypt	PoS	1.283270e+05	1000000

532 rows × 6 columns

In [14]:
```
# Use get_dummies() to create variables for text features.
crypto = pd.get_dummies(crypto_df['Algorithm'])
dummy = pd.get_dummies(crypto_df['ProofType'])
combined = pd.concat([crypto,dummy],axis =1)
df = crypto_df.merge(combined,left_index = True,right_index = True)
df = df.drop(['Algorithm','ProofType'],axis = 1)
df
```

Out[14]:

	Unnamed: 0	CoinName	TotalCoinsMined	TotalCoinSupply	1GB AES Pattern Search	536	Argon2d	BLAKE256
0	42	42 Coin	4.199995e+01	42	0	0	0	0
2	404	404Coin	1.055185e+09	532000000	0	0	0	0
5	1337	EliteCoin	2.927942e+10	314159265359	0	0	0	0
7	BTC	Bitcoin	1.792718e+07	21000000	0	0	0	0
8	ETH	Ethereum	1.076842e+08	0	0	0	0	0
...	...	...	...	...	...	...	...	...
1238	ZEPH	ZEPHYR	2.000000e+09	2000000000	0	0	0	0
1242	GAP	Gapcoin	1.493105e+07	250000000	0	0	0	0
1245	BDX	Beldex	9.802226e+08	1400222610	0	0	0	0
1246	ZEN	Horizen	7.296538e+06	21000000	0	0	0	0
1247	XBC	BitcoinPlus	1.283270e+05	1000000	0	0	0	0

532 rows × 100 columns

In [15]:
```
df.head()
```

Out[15]:

	Unnamed: 0	CoinName	TotalCoinsMined	TotalCoinSupply	1GB AES Pattern Search	536	Argon2d	BLAKE256	Bla
0	42	42 Coin	4.199995e+01	42	0	0	0	0	
2	404	404Coin	1.055185e+09	532000000	0	0	0	0	
5	1337	EliteCoin	2.927942e+10	314159265359	0	0	0	0	
7	BTC	Bitcoin	1.792718e+07	21000000	0	0	0	0	
8	ETH	Ethereum	1.076842e+08	0	0	0	0	0	

5 rows × 100 columns

In [16]:
```
df.dtypes
```

Out[16]:
```
Unnamed: 0                 object
CoinName                   object
TotalCoinsMined           float64
TotalCoinSupply            object
1GB AES Pattern Search      uint8
                            ...
Proof of Authority          uint8
Proof of Trust              uint8
TPoS                        uint8
Zero-Knowledge Proof        uint8
dPoW/PoW                    uint8
Length: 100, dtype: object
```

In [18]:
```
# Standardize the data with StandardScaler().
df_scaled = StandardScaler().fit_transform(df[['TotalCoinsMined','TotalCoin
print(df_scaled)
```

```
[[-0.11710817 -0.1528703 ]
 [-0.09396955 -0.145009  ]
 [ 0.52494561  4.48942416]
 ...
 [-0.09561336 -0.13217937]
 [-0.11694817 -0.15255998]
 [-0.11710536 -0.15285552]]
```

In [19]:
```
df_scaled
```

Out[19]:
```
array([[-0.11710817, -0.1528703 ],
       [-0.09396955, -0.145009  ],
       [ 0.52494561,  4.48942416],
       ...,
       [-0.09561336, -0.13217937],
       [-0.11694817, -0.15255998],
       [-0.11710536, -0.15285552]])
```

Deliverable 2: Reducing Data Dimensions Using PCA

In [23]:
```
# Using PCA to reduce dimension to three principal components.
pca = PCA(n_components=2)
pca_df = pca.fit_transform(df_scaled)
pca_df
```

Out[23]:
```
array([[-0.19090361, -0.02528764],
       [-0.16898335, -0.03609034],
       [ 3.54569487,  2.80330967],
       ...,
       [-0.16107379, -0.02585607],
       [-0.19057104, -0.02518136],
       [-0.19089117, -0.02527918]])
```

In [26]:
```
# Create a DataFrame with the three principal components.
pcs_df = pd.DataFrame(
    data = pca_df, columns = ['CRYPT1','CRYPT2'], index = crypto_df.index
)
pcs_df
```

Out[26]:

	CRYPT1	CRYPT2
0	-0.190904	-0.025288
2	-0.168983	-0.036090
5	3.545695	2.803310
7	-0.190406	-0.025346
8	-0.189234	-0.026957
...	...	...
1238	-0.138994	-0.035402
1242	-0.188060	-0.022907
1245	-0.161074	-0.025856
1246	-0.190571	-0.025181
1247	-0.190891	-0.025279

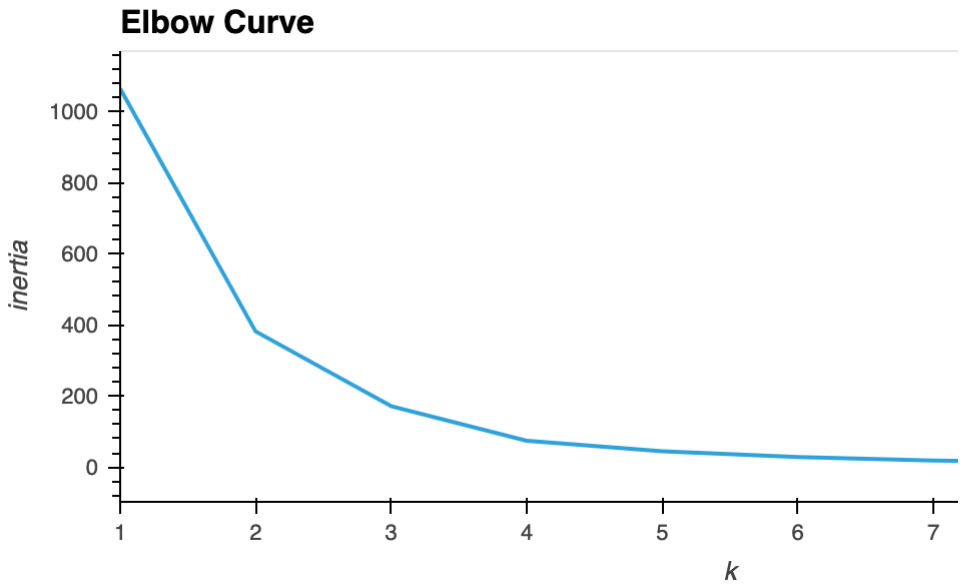
532 rows × 2 columns

Deliverable 3: Clustering Crytocurrencies Using K-Means

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

In [27]:
```
# Create an elbow curve to find the best value for K.
inertia = []
k = list(range(1, 11))
# Calculate the inertia for the range of K values
for i in k:
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(pcs_df)
    inertia.append(km.inertia_)

elbow_data = {"k":k,"inertia":inertia}
elbow_df = pd.DataFrame(elbow_data)
elbow_df.hvplot.line(x="k",y="inertia",xticks=k,title="Elbow Curve")
```

Out[27]:



Running K-Means with `k=4`

In [29]:
```
# Initialize the K-Means model.
model = KMeans(n_clusters=4, random_state=0)
# Fit the model
model.fit(pcs_df)

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

In [31]:
```
# Create a new DataFrame including predicted clusters and cryptocurrencies
# Concatentate the crypto_df and pcs_df DataFrames on the same columns.
clustered_df = pd.concat([crypto_df,pcs_df],axis =1)

#  Add a new column, "CoinName" to the clustered_df DataFrame that holds th
clustered_df['CoinName'] = names['CoinName']

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

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

```
(532, 9)
```

Out[31]:

	Unnamed: 0	CoinName	Algorithm	ProofType	TotalCoinsMined	TotalCoinSupply	CRYPT1	C
0	42	42 Coin	Scrypt	PoW/PoS	4.199995e+01	42	-0.190904	-0.
2	404	404Coin	Scrypt	PoW/PoS	1.055185e+09	532000000	-0.168983	-0.
5	1337	EliteCoin	X13	PoW/PoS	2.927942e+10	314159265359	3.545695	2.
7	BTC	Bitcoin	SHA-256	PoW	1.792718e+07	21000000	-0.190406	-0.
8	ETH	Ethereum	Ethash	PoW	1.076842e+08	0	-0.189234	-0.
9	LTC	Litecoin	Scrypt	PoW	6.303924e+07	84000000	-0.189048	-0.
10	DASH	Dash	X11	PoW/PoS	9.031294e+06	22000000	-0.190534	-0.
11	XMR	Monero	CryptoNight-V7	PoW	1.720114e+07	0	-0.190637	-0.
12	ETC	Ethereum Classic	Ethash	PoW	1.133597e+08	210000000	-0.186952	-0.
13	ZEC	ZCash	Equihash	PoW	7.383056e+06	21000000	-0.190570	-0.

Deliverable 4: Visualizing Cryptocurrencies Results

3D-Scatter with Clusters

In [32]:
```
# Creating a 3D-Scatter with the PCA data and the clusters
fig = px.scatter_3d(
    clustered_df,
    hover_name="CoinName",
    hover_data=["Algorithm"],
    x="CRYPT1",
    y="CRYPT2",
    color="Class",
    symbol="Class",
    width=800,
)
fig.update_layout(legend=dict(x=0, y=1))
fig.show()
```

In [33]:
```
# Create a table with tradable cryptocurrencies.
clustered_df.hvplot.table(sortable=True, selectable=True)
```

Out[33]:

#	Unnamed: 0	CoinName	Algorithm	ProofType	TotalCoinsMin	TotalCoinSupp
0	42	42 Coin	Scrypt	PoW/PoS	41.999954	42
1	404	404Coin	Scrypt	PoW/PoS	1,055,184,902	532000000
2	1337	EliteCoin	X13	PoW/PoS	29,279,424,62	31415926535
3	BTC	Bitcoin	SHA-256	PoW	17,927,175.0	21000000
4	ETH	Ethereum	Ethash	PoW	107,684,222.(	0
5	LTC	Litecoin	Scrypt	PoW	63,039,243.3(	84000000
6	DASH	Dash	X11	PoW/PoS	9,031,294.37	22000000
7	XMR	Monero	CryptoNight-\	PoW	17,201,143.1	0
8	ETC	Ethereum Cla	Ethash	PoW	113,359,703.(	210000000
9	ZEC	ZCash	Equihash	PoW	7,383,056.25	21000000
10	BTS	Bitshares	SHA-512	PoS	2,741,570,00(	3600570502

In [34]:
```
# Print the total number of tradable cryptocurrencies.
print(clustered_df.count)
```

```
<bound method DataFrame.count of      Unnamed: 0     CoinName    Algorith
m ProofType  TotalCoinsMined  \
0            42      42 Coin       Scrypt   PoW/PoS     4.199995e+01
2           404      404Coin       Scrypt   PoW/PoS     1.055185e+09
5          1337    EliteCoin          X13   PoW/PoS     2.927942e+10
7           BTC      Bitcoin      SHA-256       PoW     1.792718e+07
8           ETH     Ethereum       Ethash       PoW     1.076842e+08
...         ...          ...          ...       ...              ...
1238       ZEPH       ZEPHYR      SHA-256      DPoS     2.000000e+09
1242        GAP      Gapcoin       Scrypt   PoW/PoS     1.493105e+07
1245        BDX       Beldex  CryptoNight       PoW     9.802226e+08
1246        ZEN      Horizen     Equihash       PoW     7.296538e+06
1247        XBC  BitcoinPlus       Scrypt       PoS     1.283270e+05

     TotalCoinSupply    CRYPT1    CRYPT2  Class
0                 42 -0.190904 -0.025288      0
2          532000000 -0.168983 -0.036090      0
5       314159265359  3.545695  2.803310      3
7           21000000 -0.190406 -0.025346      0
8                  0 -0.189234 -0.026957      0
...              ...       ...       ...    ...
1238      2000000000 -0.138994 -0.035402      0
1242       250000000 -0.188060 -0.022907      0
1245      1400222610 -0.161074 -0.025856      0
1246        21000000 -0.190571 -0.025181      0
1247         1000000 -0.190891 -0.025279      0

[532 rows x 9 columns]>
```

In [35]:
```
# Scaling data to create the scatter plot with tradable cryptocurrencies.
scaled =MinMaxScaler().fit_transform(clustered_df[["TotalCoinSupply","Total
print(scaled)
```

```
[[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]]
```

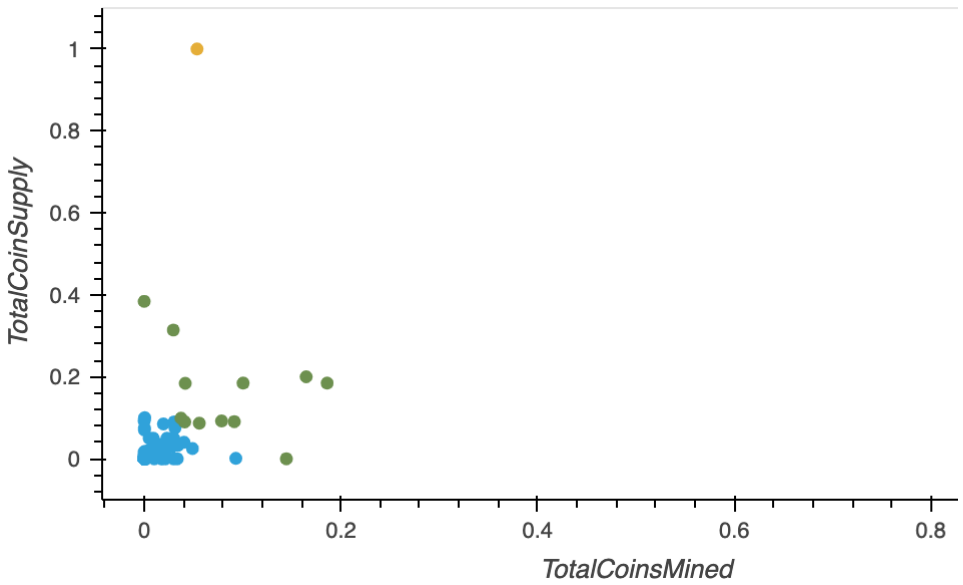
In [36]:
```
# Create a new DataFrame that has the scaled data with the clustered_df Dat
new_df = pd.DataFrame(
    data = scaled,columns = ["TotalCoinSupply","TotalCoinsMined"], index =
)
# Add the "CoinName" column from the clustered_df DataFrame to the new Data
new_df = pd.concat([new_df,clustered_df['CoinName']],axis =1)

# Add the "Class" column from the clustered_df DataFrame to the new DataFra
plot_df = pd.concat([new_df,clustered_df["Class"]],axis =1)


plot_df.head(10)
```

Out[36]:

	TotalCoinSupply	TotalCoinsMined	CoinName	Class
0	4.200000e-11	0.000000	42 Coin	0
2	5.320000e-04	0.001066	404Coin	0
5	3.141593e-01	0.029576	EliteCoin	3
7	2.100000e-05	0.000018	Bitcoin	0
8	0.000000e+00	0.000109	Ethereum	0
9	8.400000e-05	0.000064	Litecoin	0
10	2.200000e-05	0.000009	Dash	0
11	0.000000e+00	0.000017	Monero	0
12	2.100000e-04	0.000115	Ethereum Classic	0
13	2.100000e-05	0.000007	ZCash	0

In [37]:
```
# Create a hvplot.scatter plot using x="TotalCoinsMined" and y="TotalCoinSu
plot_df.hvplot.scatter(x="TotalCoinsMined", y="TotalCoinSupply", by="Class"
```

Out[37]:



In [ ]: