File Edit View Run Kernel Tabs Settings Help

crypto_clustering.ipynb

Code Python 3

Clustering Crypto

[1]:
```
# Initial imports
import requests
import pandas as pd
import hvplot.pandas
from pathlib import Path
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
```

Fetching Cryptocurrency Data

[2]:
```
# Use the following endpoint to fetch json data
url = "https://min-api.cryptocompare.com/data/all/coinlist"
response = requests.get(url).json()
```

[3]:
```
# Create a DataFrame
crypto_df = pd.DataFrame(response['Data']).T
crypto_df.head()
```

[3]:

	Id	Url	ImageUrl	ContentCreatedOn	Name	Symbol	CoinName	FullName	Descr
42	4321	/coins/42/overview	/media/35650717/42.jpg	1427211129	42	42	42 Coin	42 Coin (42)	Ever about 4 is 42 - fro
300	749869	/coins/300/overview	/media/27010595/300.png	1517935016	300	300	300 token	300 token (300)	300 to an I toke Token wa
365	33639	/coins/365/overview	/media/352070/365.png	1480032918	365	365	365Coin	365Coin (365)	365C Proof o and P St
404	21227	/coins/404/overview	/media/35650851/404-300x300.jpg	1466100361	404	404	404Coin	404Coin (404)	4 Po cryptocu th
433	926547	/coins/433/overview	/media/34836095/433.png	1541597321	433	433	433 Token	433 Token (433)	433 Tok decentr platfo

5 rows × 36 columns

[4]:
```
# Alternatively, use the provided csv file:
# file_path = Path("Resources/crypto_data.csv")

# Create a DataFrame
# crypto_df = pd.read_csv(file_path, index_col=0)
# crypto_df.head(10)

### Data Preprocessing
```

[5]:
```
# Keep only necessary columns:
# 'CoinName','Algorithm','IsTrading','ProofType','TotalCoinsMined','TotalCoinSupply'
crypto_df=crypto_df[['CoinName','Algorithm','IsTrading','ProofType','TotalCoinsMined','MaxSupply']]
crypto_df.head(10)
```

[5]:

	CoinName	Algorithm	IsTrading	ProofType	TotalCoinsMined	MaxSupply
42	42 Coin	Scrypt	True	PoW/PoS	0	0
300	300 token	N/A	True	N/A	300	300
365	365Coin	X11	True	PoW/PoS	0	0
404	404Coin	Scrypt	True	PoW/PoS	0	0
433	433 Token	N/A	False	N/A	NaN	NaN
611	SixEleven	SHA-256	True	PoW	0	0
808	808	SHA-256	True	PoW/PoS	0	0
888	Octocoin	N/A	True	PoW	0	0
1337	EliteCoin	X13	True	PoW/PoS	0	0
2015	2015 coin	X11	True	PoW/PoS	0	0

[6]:
```
# Keep only cryptocurrencies that are trading
crypto_df = crypto_df[crypto_df["IsTrading"] == True]
print(crypto_df.shape)
crypto_df.head(10)
```

```
(5775, 6)
```

[6]:

	CoinName	Algorithm	IsTrading	ProofType	TotalCoinsMined	MaxSupply
42	42 Coin	Scrypt	True	PoW/PoS	0	0
300	300 token	N/A	True	N/A	300	300
365	365Coin	X11	True	PoW/PoS	0	0
404	404Coin	Scrypt	True	PoW/PoS	0	0
611	SixEleven	SHA-256	True	PoW	0	0
808	808	SHA-256	True	PoW/PoS	0	0
888	Octocoin	N/A	True	PoW	0	0
1337	EliteCoin	X13	True	PoW/PoS	0	0
2015	2015 coin	X11	True	PoW/PoS	0	0
BTCD	BitcoinDark	SHA-256	True	PoW/PoS	NaN	NaN

[7]:
```
# Keep only cryptocurrencies with a working algorithm
crypto_df = crypto_df[crypto_df["Algorithm"] != "N/A"]
print(crypto_df.shape)
crypto_df.head(10)
```

```
(1612, 6)
```

[7]:

	CoinName	Algorithm	IsTrading	ProofType	TotalCoinsMined	MaxSupply
42	42 Coin	Scrypt	True	PoW/PoS	0	0
365	365Coin	X11	True	PoW/PoS	0	0
404	404Coin	Scrypt	True	PoW/PoS	0	0
611	SixEleven	SHA-256	True	PoW	0	0
808	808	SHA-256	True	PoW/PoS	0	0
1337	EliteCoin	X13	True	PoW/PoS	0	0
2015	2015 coin	X11	True	PoW/PoS	0	0
BTCD	BitcoinDark	SHA-256	True	PoW/PoS	NaN	NaN
CRAIG	CraigsCoin	X11	True	PoS	NaN	NaN
XBS	Bitstake	X11	True	PoW/PoS	NaN	NaN

[8]:
```
# Remove the "IsTrading" column
crypto_df.drop("IsTrading", axis=1, inplace=True)
print(crypto_df.shape)
crypto_df.head(10)
```

(1612, 5)

[8]:

	CoinName	Algorithm	ProofType	TotalCoinsMined	MaxSupply
42	42 Coin	Scrypt	PoW/PoS	0	0
365	365Coin	X11	PoW/PoS	0	0
404	404Coin	Scrypt	PoW/PoS	0	0
611	SixEleven	SHA-256	PoW	0	0
808	808	SHA-256	PoW/PoS	0	0
1337	EliteCoin	X13	PoW/PoS	0	0
2015	2015 coin	X11	PoW/PoS	0	0
BTCD	BitcoinDark	SHA-256	PoW/PoS	NaN	NaN
CRAIG	CraigsCoin	X11	PoS	NaN	NaN
XBS	Bitstake	X11	PoW/PoS	NaN	NaN

[9]:

```
# Remove rows with at least 1 null value
crypto_df = crypto_df.dropna(axis=0, how="any")
print(crypto_df.shape)
crypto_df.head(10)
```

(413, 5)

[9]:

	CoinName	Algorithm	ProofType	TotalCoinsMined	MaxSupply
42	42 Coin	Scrypt	PoW/PoS	0	0
365	365Coin	X11	PoW/PoS	0	0
404	404Coin	Scrypt	PoW/PoS	0	0
611	SixEleven	SHA-256	PoW	0	0
808	808	SHA-256	PoW/PoS	0	0
1337	EliteCoin	X13	PoW/PoS	0	0
2015	2015 coin	X11	PoW/PoS	0	0
XPD	PetroDollar	SHA-256D	N/A	0	-1
ACOIN	ACoin	SHA-256	PoW	0	0
XMY	MyriadCoin	Multiple	PoW	0	2000000000

[10]:

```
# Remove rows with cryptocurrencies having no coins mined
crypto_df = crypto_df[crypto_df["TotalCoinsMined"] > 0]
print(crypto_df.shape)
crypto_df.head(10)
```

(265, 5)

[10]:

	CoinName	Algorithm	ProofType	TotalCoinsMined	MaxSupply
NSR	NuShares	PoS	PoS	6.16681e+09	0
TRI	Triangles Coin	X13	PoW/PoS	189138	0
CMTC	CometCoin	Scrypt	PoW	872830	0
CHAT	OpenChat	Scrypt	PoW/PoS	1000000000	-1
QRL	Quantum Resistant Ledger	RandomX	PoW	7.51854e+07	105000000
AMB	Amber	Dagger	PoA	819631800	-1
BTCZ	BitcoinZ	Equihash	PoW	1.04953e+10	21000000000
PURA	Pura	X11	PoW	1.88359e+08	-1
BTCP	Bitcoin Private	Equihash	PoW	3.81888e+06	22873588
ADK	Aidos Kuneen	IMesh	PoW	25000000	0

[11]:

```
# Drop rows where there are 'N/A' text values
crypto_df = crypto_df[crypto_df.iloc[:] != 'N/A'].dropna()
crypto_df.head(10)
```

[11]:

	CoinName	Algorithm	ProofType	TotalCoinsMined	MaxSupply

NSR	NuShares	PoS	PoS	6.16681e+09	0
TRI	Triangles Coin	X13	PoW/PoS	189138	0
CMTC	CometCoin	Scrypt	PoW	872830	0
CHAT	OpenChat	Scrypt	PoW/PoS	1000000000	-1
QRL	Quantum Resistant Ledger	RandomX	PoW	7.51854e+07	105000000
AMB	Amber	Dagger	PoA	819631800	-1
BTCZ	BitcoinZ	Equihash	PoW	1.04953e+10	21000000000
PURA	Pura	X11	PoW	1.88359e+08	-1
BTCP	Bitcoin Private	Equihash	PoW	3.81888e+06	22873588
ADK	Aidos Kuneen	IMesh	PoW	25000000	0

[12]:
```
# Store the 'CoinName'column in its own DataFrame prior to dropping it from crypto_df
coins_name = pd.DataFrame(crypto_df["CoinName"], index=crypto_df.index)
print(coins_name.shape)
coins_name.head()
```

```
(124, 1)
```

[12]:

	CoinName
NSR	NuShares
TRI	Triangles Coin
CMTC	CometCoin
CHAT	OpenChat
QRL	Quantum Resistant Ledger

[13]:
```
# Drop the 'CoinName' column since it's not going to be used on the clustering algorithm
crypto_df = crypto_df.drop("CoinName", axis=1)
print(crypto_df.shape)
crypto_df.head(10)
```

```
(124, 4)
```

[13]:

	Algorithm	ProofType	TotalCoinsMined	MaxSupply
NSR	PoS	PoS	6.16681e+09	0
TRI	X13	PoW/PoS	189138	0
CMTC	Scrypt	PoW	872830	0
CHAT	Scrypt	PoW/PoS	1000000000	-1
QRL	RandomX	PoW	7.51854e+07	105000000
AMB	Dagger	PoA	819631800	-1
BTCZ	Equihash	PoW	1.04953e+10	21000000000
PURA	X11	PoW	1.88359e+08	-1
BTCP	Equihash	PoW	3.81888e+06	22873588
ADK	IMesh	PoW	25000000	0

[14]:
```
# Create dummy variables for text features
X = pd.get_dummies(data=crypto_df, columns=["Algorithm", "ProofType"])
print(X.shape)
X.head(10)
```

```
(124, 76)
```

[14]:

	TotalCoinsMined	MaxSupply	Algorithm_Autolykos	Algorithm_BEP-2	Algorithm_BEP-20 Token	Algorithm_BLAKE256	Algorithı
NSR	6.16681e+09	0	0	0	0	0	
TRI	189138	0	0	0	0	0	
CMTC	872830	0	0	0	0	0	
CHAT	1000000000	-1	0	0	0	0	

QRL	7.51854e+07	105000000	0	0	0	0
AMB	819631800	-1	0	0	0	0
BTCZ	1.04953e+10	21000000000	0	0	0	0
PURA	1.88359e+08	-1	0	0	0	0
BTCP	3.81888e+06	22873588	0	0	0	0
ADK	25000000	0	0	0	0	0

10 rows × 76 columns

[15]:
```
# Standardize data
X = StandardScaler().fit_transform(X)
X[:5]
```

[15]:
```
array([[-0.09933113, -0.09656221, -0.09016696, -0.09016696, -0.09016696,
        -0.12803688, -0.09016696, -0.09016696, -0.12803688, -0.12803688,
        -0.12803688, -0.09016696, -0.09016696, -0.2445998 , -0.12803688,
        -0.09016696, -0.09016696, -0.09016696, -0.31200182, -0.09016696,
        -0.09016696, -0.20498002, -0.09016696, -0.09016696, -0.12803688,
        -0.09016696, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
        -0.15745916, -0.09016696, -0.09016696, -0.12803688,  5.47722558,
        -0.09016696, -0.15745916, -0.12803688, -0.31200182, -0.12803688,
        -0.09016696, -0.09016696, -0.09016696, -0.4515346 , -0.09016696,
        -0.09016696, -0.09016696, -0.18257419, -0.09016696, -0.20498002,
        -0.12803688, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
        -0.09016696, -0.26261287, -0.09016696, -0.09016696, -0.12803688,
        -0.09016696, -0.09016696,  3.3763886 , -0.09016696, -0.09016696,
        -0.09016696, -0.98399897, -0.48989795, -0.09016696, -0.09016696,
        -0.09016696, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
        -0.09016696],
       [-0.10297655, -0.09656221, -0.09016696, -0.09016696, -0.09016696,
        -0.12803688, -0.09016696, -0.09016696, -0.12803688, -0.12803688,
        -0.12803688, -0.09016696, -0.09016696, -0.2445998 , -0.12803688,
        -0.09016696, -0.09016696, -0.09016696, -0.31200182, -0.09016696,
        -0.09016696, -0.20498002, -0.09016696, -0.09016696, -0.12803688,
        -0.09016696, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
        -0.15745916, -0.09016696, -0.09016696, -0.12803688, -0.18257419,
        -0.09016696, -0.15745916, -0.12803688, -0.31200182, -0.12803688,
        -0.09016696, -0.09016696, -0.09016696, -0.4515346 , -0.09016696,
        -0.09016696, -0.09016696, -0.18257419, -0.09016696,  4.87852437,
        -0.12803688, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
        -0.09016696, -0.26261287, -0.09016696, -0.09016696, -0.12803688,
        -0.09016696, -0.09016696, -0.29617444, -0.09016696, -0.09016696,
        -0.09016696, -0.98399897,  2.04124145, -0.09016696, -0.09016696,
        -0.09016696, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
        -0.09016696],
       [-0.10297614, -0.09656221, -0.09016696, -0.09016696, -0.09016696,
        -0.12803688, -0.09016696, -0.09016696, -0.12803688, -0.12803688,
        -0.12803688, -0.09016696, -0.09016696, -0.2445998 , -0.12803688,
        -0.09016696, -0.09016696, -0.09016696, -0.31200182, -0.09016696,
        -0.09016696, -0.20498002, -0.09016696, -0.09016696, -0.12803688,
        -0.09016696, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
        -0.15745916, -0.09016696, -0.09016696, -0.12803688, -0.18257419,
        -0.09016696, -0.15745916, -0.12803688, -0.31200182, -0.12803688,
        -0.09016696, -0.09016696, -0.09016696,  2.21466971, -0.09016696,
        -0.09016696, -0.09016696, -0.18257419, -0.09016696, -0.20498002,
        -0.12803688, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
        -0.09016696, -0.26261287, -0.09016696, -0.09016696, -0.12803688,
        -0.09016696, -0.09016696, -0.29617444, -0.09016696, -0.09016696,
        -0.09016696,  1.01626123, -0.48989795, -0.09016696, -0.09016696,
        -0.09016696, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
        -0.09016696],
       [-0.1023855 , -0.09656221, -0.09016696, -0.09016696, -0.09016696,
        -0.12803688, -0.09016696, -0.09016696, -0.12803688, -0.12803688,
        -0.12803688, -0.09016696, -0.09016696, -0.2445998 , -0.12803688,
        -0.09016696, -0.09016696, -0.09016696, -0.31200182, -0.09016696,
        -0.09016696, -0.20498002, -0.09016696, -0.09016696, -0.12803688,
        -0.09016696, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
        -0.15745916, -0.09016696, -0.09016696, -0.12803688, -0.18257419,
        -0.09016696, -0.15745916, -0.12803688, -0.31200182, -0.12803688,
        -0.09016696, -0.09016696, -0.09016696,  2.21466971, -0.09016696,
        -0.09016696, -0.09016696, -0.18257419, -0.09016696, -0.20498002,
        -0.12803688, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
        -0.09016696, -0.26261287, -0.09016696, -0.09016696, -0.12803688,
        -0.09016696, -0.09016696, -0.29617444, -0.09016696, -0.09016696,
        -0.09016696, -0.98399897,  2.04124145, -0.09016696, -0.09016696,
        -0.09016696, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
        -0.09016696],
```

```
[-0.10293221, -0.09650634, -0.09016696, -0.09016696, -0.09016696,
 -0.12803688, -0.09016696, -0.09016696, -0.12803688, -0.12803688,
 -0.12803688, -0.09016696, -0.09016696, -0.2445998 , -0.12803688,
 -0.09016696, -0.09016696, -0.09016696, -0.31200182, -0.09016696,
 -0.09016696, -0.20498002, -0.09016696, -0.09016696, -0.12803688,
 -0.09016696, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
 -0.15745916, -0.09016696, -0.09016696, -0.12803688, -0.18257419,
 -0.09016696, -0.15745916,  7.81024968, -0.31200182, -0.12803688,
 -0.09016696, -0.09016696, -0.09016696, -0.4515346 , -0.09016696,
 -0.09016696, -0.09016696, -0.18257419, -0.09016696, -0.20498002,
 -0.12803688, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
 -0.09016696, -0.26261287, -0.09016696, -0.09016696, -0.12803688,
 -0.09016696, -0.09016696, -0.29617444, -0.09016696, -0.09016696,
 -0.09016696,  1.01626123, -0.48989795, -0.09016696, -0.09016696,
 -0.09016696, -0.09016696, -0.09016696, -0.09016696, -0.09016696,
 -0.09016696]])
```

Reducing Dimensions Using PCA

[16]:
```
# Use PCA to reduce dimension to 3 principal components
n_comp = 3
pca = PCA(n_components=n_comp)
principal_components = pca.fit_transform(X)
principal_components
```

[16]:
```
array([[-1.28007236e+00,  5.42020350e-01, -6.31283826e-01],
       [-1.53785651e+00, -8.67238903e-01, -4.17227662e-01],
       [ 6.95502744e-01, -8.02489190e-01, -2.71525543e-01],
       [-9.45793776e-01, -1.01545044e+00, -3.78498633e-01],
       [ 1.26904375e+00, -6.07995743e-01, -1.55236160e-01],
       [-1.58156735e+00,  1.09616704e+00,  6.41567909e+00],
       [ 1.08380024e+00, -6.65546644e-01, -1.67605267e-01],
       [ 4.83508319e-01, -6.41042131e-01, -1.49879129e-01],
       [ 1.07760247e+00, -6.72058640e-01, -1.69143551e-01],
       [ 8.62219996e-01, -5.32608653e-01, -9.32431751e-02],
       [-1.52501114e+00,  1.15897094e+00,  6.39817279e+00],
       [-1.49431079e+00, -1.01429970e+00, -3.65759049e-01],
       [ 1.26900831e+00, -6.08035208e-01, -1.55245584e-01],
       [ 1.27695154e+00, -5.98904469e-01, -1.53044549e-01],
       [-1.49444574e+00, -1.01446219e+00, -3.65798650e-01],
       [ 2.22391477e+00,  6.27382672e-01,  1.37314253e-01],
       [-1.60281438e+00,  7.42432887e-01, -7.40440300e-01],
       [ 1.05614868e+00, -6.64231458e-01, -1.81435234e-01],
       [-1.51134103e+00, -1.03018320e+00, -3.71796047e-01],
       [ 1.82346935e+00,  2.04512308e-01,  3.87356451e-02],
       [-1.56540566e+00,  7.56772194e-01,  2.98466587e+00],
       [ 1.07760380e+00, -6.72057078e-01, -1.69143283e-01],
       [-5.83608897e-01, -8.75544637e-01, -2.88038257e-01],
       [ 1.82344721e+00,  2.04488869e-01,  3.87302189e-02],
       [ 1.82345162e+00,  2.04493593e-01,  3.87312796e-02],
       [-5.99815180e-01, -8.93634704e-01, -3.21587254e-01],
       [ 1.81946904e-01, -8.69049045e-03, -6.82919239e-02],
       [ 6.95562332e-01, -8.02426332e-01, -2.71510848e-01],
       [ 1.27990968e+00, -6.07681496e-01, -1.55276023e-01],
       [ 1.26197706e+00, -6.03370602e-01, -1.53990263e-01],
       [ 1.27111127e+00, -6.05034728e-01, -1.54496437e-01],
       [ 1.07766195e+00, -6.71996373e-01, -1.69128699e-01],
       [-1.84115384e+00,  3.63744756e+00, -2.13832611e+00],
       [ 1.26212616e+00, -6.03190006e-01, -1.53946932e-01],
       [ 1.07760376e+00, -6.72057242e-01, -1.69143244e-01],
       [-1.59706635e+00,  4.10623936e-02, -6.03534960e-01],
       [ 1.26193616e+00, -6.03419764e-01, -1.54002295e-01],
       [-5.99757703e-01, -8.93565263e-01, -3.21570480e-01],
       [ 1.05612701e+00, -6.64255261e-01, -1.81440216e-01],
       [ 1.26194106e+00, -6.03413991e-01, -1.54000811e-01],
       [ 1.26199115e+00, -6.03353357e-01, -1.53986236e-01],
       [ 6.95935905e-01, -8.02035345e-01, -2.71417544e-01],
       [ 1.26193528e+00, -6.03420843e-01, -1.54002543e-01],
       [-1.50277848e+00, -1.02219962e+00, -3.68748255e-01],
       [ 4.18871255e-02, -6.42046924e-01, -1.18951137e-01],
       [ 1.26449129e+00, -6.00335420e-01, -1.53255570e-01],
       [ 1.07762146e+00, -6.72038667e-01, -1.69138844e-01],
       [ 1.26901911e+00, -6.08023527e-01, -1.55243034e-01],
       [ 4.19073843e-02, -6.42022013e-01, -1.18945395e-01],
       [-3.19299049e-01, -3.72597527e-01,  3.13921932e-02],
       [-1.57592000e+00,  7.00703715e-01,  2.93707939e+00],
       [ 1.27400924e+00, -5.88574130e-01, -1.50580789e-01],
       [ 1.82345117e+00,  2.04493111e-01,  3.87311714e-02],
       [-1.76953686e+00,  3.54993271e+00, -2.06460235e+00],
       [-1.51139721e+00, -1.03024337e+00, -3.71809557e-01],
       [-9.39587561e-01, -1.00880413e+00, -3.77006233e-01],
       [-9.45876218e-01, -1.01554123e+00, -3.78517498e-01],
```

```
       [-1.59231174e+00,  1.05821768e+00,  6.37278454e+00],
       [-1.06282087e+00,  3.56627428e-01, -4.53456375e-01],
       [ 1.05612550e+00, -6.64256284e-01, -1.81440808e-01],
       [ 1.26204281e+00, -6.03291052e-01, -1.53971119e-01],
       [-1.53785535e+00, -8.67237659e-01, -4.17227382e-01],
       [-1.59359741e+00,  7.37144572e-01, -7.34416386e-01],
       [ 1.29482346e+00, -5.70813533e-01, -1.46096902e-01],
       [ 1.26273524e+00, -6.02460667e-01, -1.53766615e-01],
       [-1.84117955e+00,  3.63742003e+00, -2.13833229e+00],
       [-1.57537022e+00,  7.03719152e-01,  2.93945527e+00],
       [ 6.95507429e-01, -8.02484173e-01, -2.71524416e-01],
       [-1.49427334e+00, -1.01425019e+00, -3.65749791e-01],
       [-1.28115781e+00,  5.40857928e-01, -6.31544844e-01],
       [-1.84135583e+00,  3.63723125e+00, -2.13837468e+00],
       [ 1.82374282e+00,  2.04800267e-01,  3.88032849e-02],
       [-1.57628637e+00,  6.98694235e-01,  2.93549610e+00],
       [ 1.82345118e+00,  2.04493117e-01,  3.87311728e-02],
       [-1.53777422e+00, -8.67152024e-01, -4.17207392e-01],
       [ 1.07760527e+00, -6.72055624e-01, -1.69142880e-01],
       [ 1.82345118e+00,  2.04493117e-01,  3.87311727e-02],
       [-1.43996271e+00,  4.10592093e+00, -2.03613960e+00],
       [-1.00520917e+00, -1.07369871e-01, -5.64855133e-01],
       [ 4.19457008e-01, -1.54365965e-01, -3.38238373e-01],
       [-1.72505693e+00,  2.11253551e+00, -1.43062580e+00],
       [-1.60278140e+00,  7.42467720e-01, -7.40432178e-01],
       [ 1.36840251e+00, -4.93118448e-01, -1.27982089e-01],
       [ 1.05797998e+00, -6.62309113e-01, -1.80980017e-01],
       [-1.56797344e+00,  7.68134935e-01, -7.26951151e-01],
       [-1.15802945e+00, -8.54261865e-01, -2.56910182e-01],
       [-9.55859699e-01, -5.82437267e-01, -8.08730942e-02],
       [-1.50279712e+00, -1.02221920e+00, -3.68752881e-01],
       [ 1.28664284e+00, -6.00596277e-01, -1.53608941e-01],
       [-1.49160435e+00, -1.01096827e+00, -3.64993400e-01],
       [ 7.23181687e-01, -7.72847508e-01, -2.64869626e-01],
       [ 6.95549219e-01, -8.02439419e-01, -2.71514367e-01],
       [-1.84117158e+00,  3.63742856e+00, -2.13833038e+00],
       [-9.45982877e-01, -1.01565295e+00, -3.78544106e-01],
       [ 9.78832588e+00,  8.60876709e+00,  2.00200097e+00],
       [-5.63776690e-01, -8.85109504e-01, -2.76136086e-01],
       [ 1.07760378e+00, -6.72057230e-01, -1.69143241e-01],
       [-9.45614512e-01, -1.01526443e+00, -3.78453242e-01],
       [-1.57635261e+00,  6.98330919e-01,  2.93520984e+00],
       [ 1.26197870e+00, -6.03368428e-01, -1.53989860e-01],
       [ 1.28614542e+00, -6.01102431e-01, -1.53738709e-01],
       [-1.39551887e+00,  1.67629006e+00,  3.73531772e+00],
       [-1.56991076e+00,  7.33663263e-01,  2.96304847e+00],
       [ 1.26194784e+00, -6.03405409e-01, -1.53998986e-01],
       [-9.55825372e-01, -5.82394839e-01, -8.08634513e-02],
       [ 1.28290029e+00, -5.91747610e-01, -1.51523220e-01],
       [-1.84138652e+00,  3.63719839e+00, -2.13838206e+00],
       [-9.46003640e-01, -1.01567518e+00, -3.78549099e-01],
       [-5.99069705e-01, -8.92718031e-01, -3.21375986e-01],
       [ 6.95523569e-01, -8.02466889e-01, -2.71520535e-01],
       [-1.51138476e+00, -1.03023003e+00, -3.71806562e-01],
       [-1.59356573e+00,  7.37183526e-01, -7.34407408e-01],
       [-9.46003132e-01, -1.01567464e+00, -3.78548977e-01],
       [-1.57631299e+00,  6.98535723e-01,  2.93539781e+00],
       [-1.53784142e+00, -8.67222740e-01, -4.17224032e-01],
       [ 1.27262285e+00, -6.02754635e-01, -1.53994319e-01],
       [ 1.82344695e+00,  2.04488715e-01,  3.87301081e-02],
       [ 1.26901175e+00, -6.08029384e-01, -1.55244095e-01],
       [ 6.95520121e-01, -8.02470581e-01, -2.71521364e-01],
       [ 6.95518900e-01, -8.02472140e-01, -2.71521562e-01],
       [ 6.95533019e-01, -8.02457270e-01, -2.71518071e-01],
       [-1.84135921e+00,  3.63722763e+00, -2.13837550e+00],
       [-1.68979240e+00,  3.21598840e+00, -1.88945329e+00],
       [ 1.26194341e+00, -6.03411090e-01, -1.54000145e-01]])
```

[17]:
```
# Create a DataFrame with the principal components data
col_names = [f"PC {i}" for i in range(1, n_comp + 1)]
pcs_df = pd.DataFrame(principal_components, columns=col_names, index=crypto_df.index)
print(pcs_df.shape)
pcs_df.head(10)
```

```
(124, 3)
```

[17]:

	PC 1	PC 2	PC 3
NSR	-1.280072	0.542020	-0.631284
TRI	-1.537857	-0.867239	-0.417228
CMTC	0.695503	-0.802489	-0.271526
CHAT	-0.945794	-1.015450	-0.378499

QRL	1.269044	-0.607996	-0.155236
AMB	-1.581567	1.096167	6.415679
BTCZ	1.083800	-0.665547	-0.167605
PURA	0.483508	-0.641042	-0.149879
BTCP	1.077602	-0.672059	-0.169144
ADK	0.862220	-0.532609	-0.093243

Clustering Crytocurrencies Using K-Means

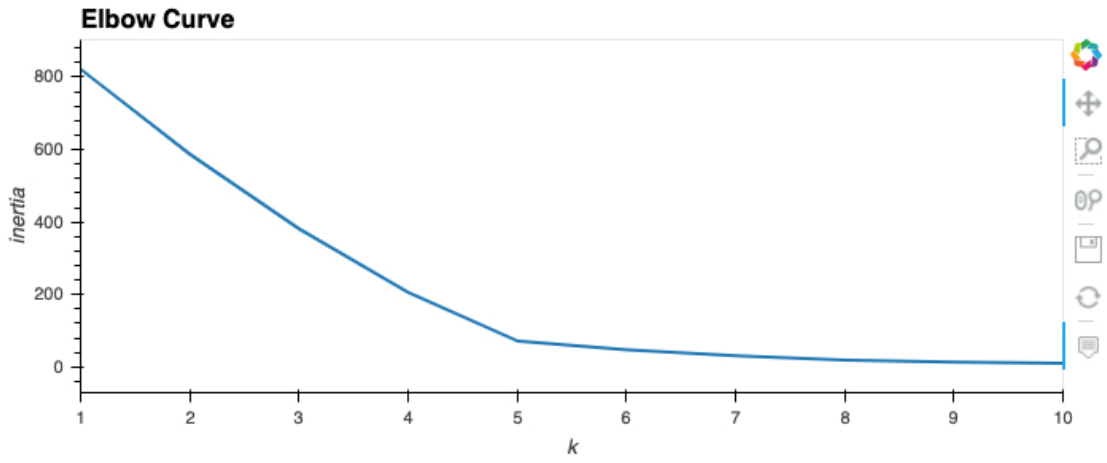
Find the Best Value for `k` Using the Elbow Curve

[18]:
```
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_)

# Create the Elbow Curve using hvPlot
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")
```

[18]:



Running K-Means with `k=4`

[19]:
```
# 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)

# Create a new DataFrame including predicted clusters and cryptocurrencies features
clustered_df = pd.concat([crypto_df, pcs_df], axis=1, sort=False)
clustered_df["CoinName"] = coins_name["CoinName"]
clustered_df["Class"] = model.labels_
print(clustered_df.shape)
clustered_df.head(10)
```

```
(124, 9)
```

[19]:

	Algorithm	ProofType	TotalCoinsMined	MaxSupply	PC 1	PC 2	PC 3	CoinName	Class
NSR	PoS	PoS	6.16681e+09	0	-1.280072	0.542020	-0.631284	NuShares	1
TRI	X13	PoW/PoS	189138	0	-1.537857	-0.867239	-0.417228	Triangles Coin	1
CMTC	Scrypt	PoW	872830	0	0.695503	-0.802489	-0.271526	CometCoin	1

CHAT	Scrypt	PoW/PoS	1000000000	-1	-0.945794	-1.015450	-0.378499	OpenChat	1
QRL	RandomX	PoW	7.51854e+07	105000000	1.269044	-0.607996	-0.155236	Quantum Resistant Ledger	1
AMB	Dagger	PoA	819631800	-1	-1.581567	1.096167	6.415679	Amber	2
BTCZ	Equihash	PoW	1.04953e+10	21000000000	1.083800	-0.665547	-0.167605	BitcoinZ	1
PURA	X11	PoW	1.88359e+08	-1	0.483508	-0.641042	-0.149879	Pura	1
BTCP	Equihash	PoW	3.81888e+06	22873588	1.077602	-0.672059	-0.169144	Bitcoin Private	1
ADK	IMesh	PoW	25000000	0	0.862220	-0.532609	-0.093243	Aidos Kuneen	1

Visualizing Results

Scatter Plot with Tradable Cryptocurrencies

[20]:
```
# Scale data to create the scatter plot
mm_scaler = MinMaxScaler()
plot_data = mm_scaler.fit_transform(
    clustered_df[["MaxSupply", "TotalCoinsMined"]]
)
plot_df = pd.DataFrame(
    plot_data, columns=["MaxSupply", "TotalCoinsMined"], index=clustered_df.index
)
plot_df["CoinName"] = clustered_df["CoinName"]
plot_df["Class"] = clustered_df["Class"]
plot_df.head()
```

[20]:

	MaxSupply	TotalCoinsMined	CoinName	Class
NSR	4.761905e-14	3.265908e-04	NuShares	1
TRI	4.761905e-14	9.882636e-09	Triangles Coin	1
CMTC	4.761905e-14	4.609061e-08	CometCoin	1
CHAT	0.000000e+00	5.295937e-05	OpenChat	1
QRL	5.000000e-06	3.981645e-06	Quantum Resistant Ledger	1

[21]:
```
# Plot the scatter with x="TotalCoinsMined" and y="TotalCoinSupply"
plot_df.hvplot.scatter(
    x="TotalCoinsMined", y="MaxSupply", hover_cols=["CoinName"], by="Class"
)
```

[21]:

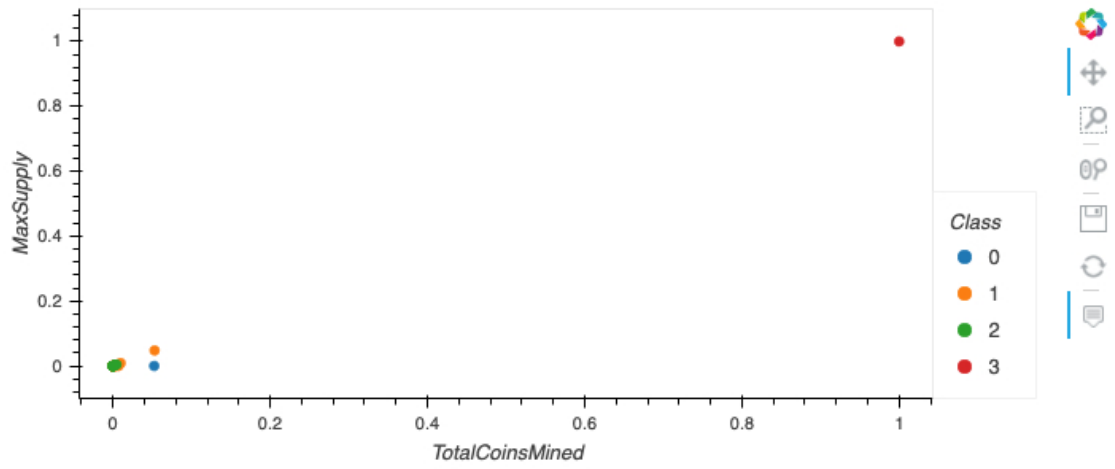


Table of Tradable Cryptocurrencies

[22]:
```
# Table with tradable cryptos
clustered_df[
    [
        "CoinName",
        "Algorithm",
        "ProofType",
```

```
        "MaxSupply",
        "TotalCoinsMined",
        "Class",
    ]
].hvplot.table()
```

[22]:

#	CoinName	Algorithm	ProofType	MaxSupply	TotalCoinsMined	Class
0	NuShares	PoS	PoS	0	6166805595.8311	1
1	Triangles Coin	X13	PoW/PoS	0	189138.196177	1
2	CometCoin	Scrypt	PoW	0	872830	1
3	OpenChat	Scrypt	PoW/PoS	-1	1000000000	1
4	Quantum Resistant L	RandomX	PoW	105000000	75185353.36876895	1
5	Amber	Dagger	PoA	-1	819631800	2
6	BitcoinZ	Equihash	PoW	21000000000	10495346127.27862	1
7	Pura	X11	PoW	-1	188358976.8396980	1
8	Bitcoin Private	Equihash	PoW	22873588	3818878.387801544	1
9	Aidos Kuneen	IMesh	PoW	0	25000000	1
10	DAPS Coin	Dagger	PoW/PoS/PoA	70000000000	62319462900	2

[23]:

```
# Print the total number of tradable cryptocurrencies
print(f"There are {clustered_df.shape[0]} tradable cryptocurrencies.")
```

```
There are 124 tradable cryptocurrencies.
```

[ ]: