crypto_clustering

June 1, 2022

1 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
```

1.0.1 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  \
42      4321   /coins/42/overview           /media/35650717/42.jpg
300   749869  /coins/300/overview          /media/27010595/300.png
365    33639  /coins/365/overview            /media/352070/365.png
404    21227  /coins/404/overview  /media/35650851/404-300x300.jpg
433   926547  /coins/433/overview          /media/34836095/433.png

    ContentCreatedOn Name Symbol   CoinName         FullName  \
42        1427211129   42     42    42 Coin     42 Coin (42)
300       1517935016  300    300  300 token  300 token (300)
365       1480032918  365    365    365Coin    365Coin (365)
404       1466100361  404    404    404Coin    404Coin (404)
433       1541597321  433    433  433 Token  433 Token (433)

                                          Description AssetTokenStatus  …  \
42   Everything about 42 coin is 42 - apart from th…              N/A  …
```

```
300  300 token is an ERC20 token. This Token was cr…              N/A  …
365  365Coin is a Proof of Work and Proof of Stake …              N/A  …
404  404 is a PoW/PoS hybrid cryptocurrency that al…              N/A  …
433  433 Token is a decentralised soccer platform t…         Finished  …

    MaxSupply MktCapPenalty IsUsedInDefi IsUsedInNft PlatformType  \
42         42             0            0           0   blockchain
300       300             0            0           0        token
365        -1             0            0           0   blockchain
404        -1             0            0           0   blockchain
433       NaN           NaN          NaN         NaN          NaN

    AlgorithmType Difficulty BuiltOn  \
42         scrypt   0.504232     NaN
300           NaN        NaN     ETH
365           NaN        NaN     NaN
404           NaN        NaN     NaN
433           NaN        NaN     NaN

                          SmartContractAddress DecimalPoints
42                                         NaN           NaN
300  0xaec98a708810414878c3bcdf46aad31ded4a4557            18
365                                        NaN           NaN
404                                        NaN           NaN
433                                        NaN           NaN

[5 rows x 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','MaxSupp
crypto_df.head(10)
```

[5]:
```
        CoinName Algorithm IsTrading ProofType TotalCoinsMined MaxSupply
42       42 Coin    Scrypt      True   PoW/PoS       41.999952        42
300    300 token       N/A      True       N/A             300       300
365      365Coin       X11      True   PoW/PoS               0        -1
```

404	404Coin	Scrypt	True	PoW/PoS	0	-1
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)
```

```
(6922, 6)
```

[6]:

	CoinName	Algorithm	IsTrading	ProofType	TotalCoinsMined	MaxSupply
42	42 Coin	Scrypt	True	PoW/PoS	41.999952	42
300	300 token	N/A	True	N/A	300	300
365	365Coin	X11	True	PoW/PoS	0	-1
404	404Coin	Scrypt	True	PoW/PoS	0	-1
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
XBS	Bitstake	X11	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)
```

```
(1644, 6)
```

[7]:

	CoinName	Algorithm	IsTrading	ProofType	TotalCoinsMined	MaxSupply
42	42 Coin	Scrypt	True	PoW/PoS	41.999952	42
365	365Coin	X11	True	PoW/PoS	0	-1
404	404Coin	Scrypt	True	PoW/PoS	0	-1
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
XBS	Bitstake	X11	True	PoW/PoS	NaN	NaN
XPY	PayCoin	SHA-256	True	PoS	NaN	NaN
PRC	ProsperCoin	Scrypt	True	PoW	NaN	NaN

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

```
(1644, 5)
```

[8]:

	CoinName	Algorithm	ProofType	TotalCoinsMined	MaxSupply
42	42 Coin	Scrypt	PoW/PoS	41.999952	42
365	365Coin	X11	PoW/PoS	0	-1
404	404Coin	Scrypt	PoW/PoS	0	-1
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
XBS	Bitstake	X11	PoW/PoS	NaN	NaN
XPY	PayCoin	SHA-256	PoS	NaN	NaN
PRC	ProsperCoin	Scrypt	PoW	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)
```

```
(710, 5)
```

[9]:

	CoinName	Algorithm	ProofType	TotalCoinsMined	MaxSupply
42	42 Coin	Scrypt	PoW/PoS	41.999952	42
365	365Coin	X11	PoW/PoS	0	-1
404	404Coin	Scrypt	PoW/PoS	0	-1
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
XMY	MyriadCoin	Multiple	PoW	0	2000000000
SXC	SexCoin	Scrypt	PoW	0	0

[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)
```

```
(312, 5)
```

[10]:

	CoinName	Algorithm	ProofType	TotalCoinsMined	MaxSupply
42	42 Coin	Scrypt	PoW/PoS	41.999952	42
NSR	NuShares	PoS	PoS	6178782525.8373	0

```
TRI    Triangles Coin       X13       PoW/PoS     199294.064798            0
CMTC       CometCoin    Scrypt           PoW           872830            0
CHAT        OpenChat    Scrypt       PoW/PoS       1000000000           -1
PURA            Pura       X11           PoW  188358976.839698           -1
ADK     Aidos Kuneen     IMesh           PoW         25000000            0
DAPS       DAPS Coin    Dagger  PoW/PoS/PoA       62319462900  70000000000
FOIN            Foin   SHA-256           N/A     92631000.8161    100000000
NVL           Nevula     NEP-5           N/A       40000000000  40000000000
```

[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  \
42              42 Coin    Scrypt       PoW/PoS           41.999952
NSR            NuShares       PoS           PoS     6178782525.8373
TRI      Triangles Coin       X13       PoW/PoS       199294.064798
CMTC          CometCoin    Scrypt           PoW              872830
CHAT           OpenChat    Scrypt       PoW/PoS          1000000000
PURA               Pura       X11           PoW    188358976.839698
ADK         Aidos Kuneen     IMesh           PoW            25000000
DAPS           DAPS Coin    Dagger  PoW/PoS/PoA         62319462900
VEIL               VEIL     X16RT       PoW/PoS    119516479.714871
RVC   Ravencoin Classic      X16R           PoW  10501536386.860544

        MaxSupply
42             42
NSR             0
TRI             0
CMTC            0
CHAT           -1
PURA           -1
ADK             0
DAPS  70000000000
VEIL    300000000
RVC   21000000000
```

[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()
```

```
(140, 1)
```

[12]:
```
             CoinName
42           42 Coin
```

```
NSR          NuShares
TRI    Triangles Coin
CMTC        CometCoin
CHAT         OpenChat
```

[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)
```

```
(140, 4)
```

[13]:

	Algorithm	ProofType	TotalCoinsMined	MaxSupply
42	Scrypt	PoW/PoS	41.999952	42
NSR	PoS	PoS	6178782525.8373	0
TRI	X13	PoW/PoS	199294.064798	0
CMTC	Scrypt	PoW	872830	0
CHAT	Scrypt	PoW/PoS	1000000000	-1
PURA	X11	PoW	188358976.839698	-1
ADK	IMesh	PoW	25000000	0
DAPS	Dagger	PoW/PoS/PoA	62319462900	70000000000
VEIL	X16RT	PoW/PoS	119516479.714871	300000000
RVC	X16R	PoW	10501536386.860544	21000000000

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

```
(140, 83)
```

[14]:

	TotalCoinsMined	MaxSupply	Algorithm_Autolykos	Algorithm_BEP-2	\
42	41.999952	42	0	0	
NSR	6178782525.8373	0	0	0	
TRI	199294.064798	0	0	0	
CMTC	872830	0	0	0	
CHAT	1000000000	-1	0	0	
PURA	188358976.839698	-1	0	0	
ADK	25000000	0	0	0	
DAPS	62319462900	70000000000	0	0	
VEIL	119516479.714871	300000000	0	0	
RVC	10501536386.860544	21000000000	0	0	

	Algorithm_BEP-20 Token	Algorithm_BLAKE256	Algorithm_BMW512 / Echo512	\
42	0	0	0	
NSR	0	0	0	

```
TRI                          0                    0                           0
CMTC                         0                    0                           0
CHAT                         0                    0                           0
PURA                         0                    0                           0
ADK                          0                    0                           0
DAPS                         0                    0                           0
VEIL                         0                    0                           0
RVC                          0                    0                           0

      Algorithm_Blake2B + SHA3  Algorithm_Blake2b  Algorithm_C31  …  \
42                           0                   0              0  …
NSR                          0                   0              0  …
TRI                          0                   0              0  …
CMTC                         0                   0              0  …
CHAT                         0                   0              0  …
PURA                         0                   0              0  …
ADK                          0                   0              0  …
DAPS                         0                   0              0  …
VEIL                         0                   0              0  …
RVC                          0                   0              0  …

      ProofType_PoW/PoSe  ProofType_PoW/nPoS  ProofType_ProgPoW/PoS  \
42                     0                   0                      0
NSR                    0                   0                      0
TRI                    0                   0                      0
CMTC                   0                   0                      0
CHAT                   0                   0                      0
PURA                   0                   0                      0
ADK                    0                   0                      0
DAPS                   0                   0                      0
VEIL                   0                   0                      0
RVC                    0                   0                      0

      ProofType_Proof of Authority  ProofType_Proof-of-Work  ProofType_SPoS  \
42                               0                        0               0
NSR                              0                        0               0
TRI                              0                        0               0
CMTC                             0                        0               0
CHAT                             0                        0               0
PURA                             0                        0               0
ADK                              0                        0               0
DAPS                             0                        0               0
VEIL                             0                        0               0
RVC                              0                        0               0

      ProofType_TPoS  ProofType_Zero-Knowledge Proof  ProofType_dPoW  \
42                 0                               0               0
```

```
NSR                0                                  0                0
TRI                0                                  0                0
CMTC               0                                  0                0
CHAT               0                                  0                0
PURA               0                                  0                0
ADK                0                                  0                0
DAPS               0                                  0                0
VEIL               0                                  0                0
RVC                0                                  0                0

      ProofType_dPoW/PoW
42                     0
NSR                    0
TRI                    0
CMTC                   0
CHAT                   0
PURA                   0
ADK                    0
DAPS                   0
VEIL                   0
RVC                    0

[10 rows x 83 columns]
```

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

[15]:
```
array([[-0.08660438, -0.09087225, -0.08481889, -0.08481889, -0.08481889,
        -0.12038585, -0.08481889, -0.08481889, -0.12038585, -0.12038585,
        -0.14797909, -0.08481889, -0.08481889, -0.08481889, -0.24618298,
        -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.29201253,
        -0.08481889, -0.08481889, -0.24618298, -0.08481889, -0.08481889,
        -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
        -0.08481889, -0.08481889, -0.14797909, -0.08481889, -0.08481889,
        -0.12038585, -0.19245009, -0.08481889, -0.08481889, -0.14797909,
        -0.12038585, -0.29201253, -0.12038585, -0.08481889, -0.08481889,
        -0.08481889,  2.19848433, -0.08481889, -0.08481889, -0.08481889,
        -0.08481889, -0.08481889, -0.21160368, -0.08481889, -0.19245009,
        -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
        -0.08481889, -0.26211122, -0.08481889, -0.08481889, -0.12038585,
        -0.12038585, -0.08481889, -0.31994094, -0.08481889, -0.08481889,
        -0.08481889, -0.94440028,  2.        , -0.08481889, -0.08481889,
        -0.08481889, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
        -0.08481889, -0.08481889, -0.08481889],
       [-0.08653027, -0.09087225, -0.08481889, -0.08481889, -0.08481889,
        -0.12038585, -0.08481889, -0.08481889, -0.12038585, -0.12038585,
```

```
 -0.14797909, -0.08481889, -0.08481889, -0.08481889, -0.24618298,
 -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.29201253,
 -0.08481889, -0.08481889, -0.24618298, -0.08481889, -0.08481889,
 -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
 -0.08481889, -0.08481889, -0.14797909, -0.08481889, -0.08481889,
 -0.12038585,  5.19615242, -0.08481889, -0.08481889, -0.14797909,
 -0.12038585, -0.29201253, -0.12038585, -0.08481889, -0.08481889,
 -0.08481889, -0.45485883, -0.08481889, -0.08481889, -0.08481889,
 -0.08481889, -0.08481889, -0.21160368, -0.08481889, -0.19245009,
 -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
 -0.08481889, -0.26211122, -0.08481889, -0.08481889, -0.12038585,
 -0.12038585, -0.08481889,  3.12557687, -0.08481889, -0.08481889,
 -0.08481889, -0.94440028, -0.5       , -0.08481889, -0.08481889,
 -0.08481889, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
 -0.08481889, -0.08481889, -0.08481889],
[-0.08660438, -0.09087225, -0.08481889, -0.08481889, -0.08481889,
 -0.12038585, -0.08481889, -0.08481889, -0.12038585, -0.12038585,
 -0.14797909, -0.08481889, -0.08481889, -0.08481889, -0.24618298,
 -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.29201253,
 -0.08481889, -0.08481889, -0.24618298, -0.08481889, -0.08481889,
 -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
 -0.08481889, -0.08481889, -0.14797909, -0.08481889, -0.08481889,
 -0.12038585, -0.19245009, -0.08481889, -0.08481889, -0.14797909,
 -0.12038585, -0.29201253, -0.12038585, -0.08481889, -0.08481889,
 -0.08481889, -0.45485883, -0.08481889, -0.08481889, -0.08481889,
 -0.08481889, -0.08481889, -0.21160368, -0.08481889,  5.19615242,
 -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
 -0.08481889, -0.26211122, -0.08481889, -0.08481889, -0.12038585,
 -0.12038585, -0.08481889, -0.31994094, -0.08481889, -0.08481889,
 -0.08481889, -0.94440028,  2.        , -0.08481889, -0.08481889,
 -0.08481889, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
 -0.08481889, -0.08481889, -0.08481889],
[-0.08660437, -0.09087225, -0.08481889, -0.08481889, -0.08481889,
 -0.12038585, -0.08481889, -0.08481889, -0.12038585, -0.12038585,
 -0.14797909, -0.08481889, -0.08481889, -0.08481889, -0.24618298,
 -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.29201253,
 -0.08481889, -0.08481889, -0.24618298, -0.08481889, -0.08481889,
 -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
 -0.08481889, -0.08481889, -0.14797909, -0.08481889, -0.08481889,
 -0.12038585, -0.19245009, -0.08481889, -0.08481889, -0.14797909,
 -0.12038585, -0.29201253, -0.12038585, -0.08481889, -0.08481889,
 -0.08481889,  2.19848433, -0.08481889, -0.08481889, -0.08481889,
 -0.08481889, -0.08481889, -0.21160368, -0.08481889, -0.19245009,
 -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
 -0.08481889, -0.26211122, -0.08481889, -0.08481889, -0.12038585,
 -0.12038585, -0.08481889, -0.31994094, -0.08481889, -0.08481889,
 -0.08481889,  1.05887304, -0.5       , -0.08481889, -0.08481889,
```

```
        -0.08481889, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
        -0.08481889, -0.08481889, -0.08481889],
       [-0.08659238, -0.09087225, -0.08481889, -0.08481889, -0.08481889,
        -0.12038585, -0.08481889, -0.08481889, -0.12038585, -0.12038585,
        -0.14797909, -0.08481889, -0.08481889, -0.08481889, -0.24618298,
        -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.29201253,
        -0.08481889, -0.08481889, -0.24618298, -0.08481889, -0.08481889,
        -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
        -0.08481889, -0.08481889, -0.14797909, -0.08481889, -0.08481889,
        -0.12038585, -0.19245009, -0.08481889, -0.08481889, -0.14797909,
        -0.12038585, -0.29201253, -0.12038585, -0.08481889, -0.08481889,
        -0.08481889,  2.19848433, -0.08481889, -0.08481889, -0.08481889,
        -0.08481889, -0.08481889, -0.21160368, -0.08481889, -0.19245009,
        -0.12038585, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
        -0.08481889, -0.26211122, -0.08481889, -0.08481889, -0.12038585,
        -0.12038585, -0.08481889, -0.31994094, -0.08481889, -0.08481889,
        -0.08481889, -0.94440028,  2.        , -0.08481889, -0.08481889,
        -0.08481889, -0.08481889, -0.08481889, -0.08481889, -0.08481889,
        -0.08481889, -0.08481889, -0.08481889]])
```

1.0.2 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([[ 2.22848785e-01, -1.32606183e+00, -1.34365811e+00],
       [ 6.96262339e-01, -1.16929985e+00, -3.12068027e-01],
       [ 6.54791911e-01, -1.97456592e+00, -1.62691027e+00],
       [-8.53522827e-01,  4.47471463e-01, -3.64626266e-01],
       [ 2.22853570e-01, -1.32605797e+00, -1.34365834e+00],
       [-5.55574962e-01,  1.16552388e-01, -3.38081156e-01],
       [-9.29555832e-01,  8.89836627e-01,  2.89141460e-01],
       [ 8.61758472e-01, -1.95192307e+00,  6.34026044e+00],
       [ 6.19078495e-01, -1.98033461e+00, -1.71363992e+00],
       [-1.21994362e+00,  1.24793343e+00,  2.26503417e-01],
       [ 6.19098671e-01, -1.98036517e+00, -1.71364551e+00],
       [-1.29969826e+00,  1.33905475e+00,  1.71861716e-01],
       [ 8.83591721e-01, -1.41029875e+00, -4.01970685e-01],
       [-1.01092116e+00,  9.05266812e-01,  8.33258045e-02],
       [ 6.25508531e-01, -2.00421431e+00, -1.73851162e+00],
       [-1.26320025e+00,  1.28048827e+00,  1.62596984e-01],
       [-1.10355713e+00,  1.04594313e+00,  1.36671223e-01],
       [ 6.54902882e-02, -8.68234312e-01, -8.95707984e-01],
       [-1.26319870e+00,  1.28048556e+00,  1.62596572e-01],
```

```
[-1.26319860e+00,  1.28048564e+00,  1.62596567e-01],
[ 1.42320541e-02, -7.76267901e-01, -8.78540981e-01],
[-1.86825988e-01, -4.93048702e-01, -8.16435482e-01],
[-1.22551220e+00,  1.25442329e+00,  2.27977461e-01],
[-1.21264217e+00,  1.23889648e+00,  2.24603312e-01],
[-1.21913503e+00,  1.24675596e+00,  2.26308365e-01],
[-1.10356779e+00,  1.04595766e+00,  1.36673711e-01],
[ 8.07576334e-01, -1.65592044e+00, -9.62926494e-01],
[-1.21263561e+00,  1.23888651e+00,  2.24601491e-01],
[-1.10356056e+00,  1.04594776e+00,  1.36672020e-01],
[-4.11281238e-01,  3.10076352e-01,  1.13910889e-01],
[-1.15222974e-01,  2.68011183e-02,  4.15779974e-01],
[ 8.52012077e-01, -1.89145772e+00,  3.35616167e+00],
[-1.21233703e+00,  1.23914370e+00,  2.24584535e-01],
[-1.21277505e+00,  1.23909243e+00,  2.24639633e-01],
[ 2.22849263e-01, -1.32606145e+00, -1.34365813e+00],
[ 6.96232421e-01, -1.16932322e+00, -3.12066500e-01],
[-2.53624664e-01,  5.47040342e-02,  8.29619381e-03],
[ 8.70074284e-01, -1.96904903e+00,  6.36883534e+00],
[-1.10437055e+00,  1.04713519e+00,  1.36868083e-01],
[-2.03890438e+00,  2.50300301e+00,  3.57614971e-01],
[ 6.96237556e-01, -1.16931987e+00, -3.12066818e-01],
[-1.01092622e+00,  9.05271533e-01,  8.33267997e-02],
[ 2.96599087e-01, -6.03292657e-01, -1.92394888e-01],
[ 4.65481740e-01, -1.22791599e+00, -6.52711667e-01],
[ 8.49669919e-01, -1.88606656e+00,  3.36544091e+00],
[ 1.42250389e-02, -7.76256785e-01, -8.78538991e-01],
[-1.10355820e+00,  1.04594428e+00,  1.36671447e-01],
[ 6.25507233e-01, -2.00421536e+00, -1.73851155e+00],
[-8.53521766e-01,  4.47472320e-01, -3.64626318e-01],
[ 2.22861227e-01, -1.32605179e+00, -1.34365872e+00],
[-1.21883921e+00,  1.24625650e+00,  2.26208970e-01],
[-8.53524373e-01,  4.47473912e-01, -3.64625876e-01],
[-1.21263863e+00,  1.23889062e+00,  2.24602285e-01],
[ 3.45605117e+00,  1.41373589e+00, -4.43931199e-02],
[ 5.20795800e-01, -1.65698160e+00, -1.31711296e+00],
[-1.21263575e+00,  1.23888639e+00,  2.24601500e-01],
[-1.01066384e+00,  9.05474617e-01,  8.33132427e-02],
[-1.21263537e+00,  1.23888672e+00,  2.24601478e-01],
[ 6.73579777e-01, -1.33833647e+00, -6.53129160e-01],
[ 5.21269332e-01, -1.37694053e+00, -8.25988021e-01],
[ 1.93477694e+00, -1.99152009e-01, -2.37300127e-01],
[ 8.83588695e-01, -1.41029390e+00, -4.01969917e-01],
[-8.53589605e-01,  4.47565452e-01, -3.64610434e-01],
[ 5.20794938e-01, -1.65698035e+00, -1.31711275e+00],
[ 2.22812093e-01, -1.32600342e+00, -1.34364883e+00],
[ 8.33918525e-01, -1.84162976e+00,  3.43477023e+00],
```

```
[-1.21263715e+00,  1.23888860e+00,  2.24601896e-01],
[-2.53668426e-01,  5.47702196e-02,  8.30830904e-03],
[-1.10355808e+00,  1.04594421e+00,  1.36671427e-01],
[ 6.25507522e-01, -2.00421512e+00, -1.73851157e+00],
[-1.10355805e+00,  1.04594424e+00,  1.36671426e-01],
[-4.11281716e-01,  3.10075941e-01,  1.13910916e-01],
[ 6.22260737e-01, -1.99212905e+00, -1.72589819e+00],
[-1.21264947e+00,  1.23890824e+00,  2.24605400e-01],
[ 6.19098707e-01, -1.98036514e+00, -1.71364551e+00],
[-1.21883476e+00,  1.24624968e+00,  2.26207776e-01],
[-1.10355808e+00,  1.04594421e+00,  1.36671427e-01],
[-9.32154183e-01,  8.51048764e-01,  1.22559432e-01],
[ 6.19164499e-01, -1.98030847e+00, -1.71364925e+00],
[-1.21296409e+00,  1.23940265e+00,  2.24694223e-01],
[-1.26319775e+00,  1.28048448e+00,  1.62596368e-01],
[-2.71922257e-02, -7.27577092e-01, -8.42358721e-01],
[-8.53522720e-01,  4.47471549e-01, -3.64626271e-01],
[ 8.52165859e-01, -1.89124399e+00,  3.35603299e+00],
[-1.26323141e+00,  1.28053548e+00,  1.62604657e-01],
[-8.53528769e-01,  4.47480752e-01, -3.64624779e-01],
[-1.21264248e+00,  1.23889737e+00,  2.24603437e-01],
[ 6.22258285e-01, -1.99212548e+00, -1.72589760e+00],
[ 8.79108654e-01, -1.40196624e+00, -3.99118082e-01],
[-1.21850468e+00,  1.24652978e+00,  2.26190276e-01],
[ 2.22848790e-01, -1.32606183e+00, -1.34365811e+00],
[ 2.09963867e+00,  2.05372792e-01,  1.81587409e-01],
[ 1.32579807e-01, -9.11526406e-01, -8.07767685e-01],
[ 3.45605068e+00,  1.41373549e+00, -4.43930958e-02],
[-8.53522419e-01,  4.47471792e-01, -3.64626286e-01],
[ 6.19102720e-01, -1.98036169e+00, -1.71364574e+00],
[-1.21875371e+00,  1.24634040e+00,  2.26203430e-01],
[ 5.59756717e-01, -1.07004913e+00, -1.50747972e-01],
[ 8.79109207e-01, -1.40196576e+00, -3.99118114e-01],
[ 3.45605536e+00,  1.41373926e+00, -4.43933241e-02],
[ 2.22832534e-01, -1.32603798e+00, -1.34365417e+00],
[ 6.54784214e-01, -1.97455365e+00, -1.62690832e+00],
[-1.26319861e+00,  1.28048563e+00,  1.62596567e-01],
[ 3.12220321e+00,  1.14647600e+00,  4.87500281e-02],
[ 1.15302349e+01,  8.47038769e+00, -4.75083437e-01],
[ 3.75628716e-01, -1.00741117e+00, -6.79673365e-01],
[-9.38887404e-01,  8.61852974e-01,  1.24268586e-01],
[-1.01117545e+00,  9.05633750e-01,  8.33868602e-02],
[ 8.76211425e-01, -1.39727009e+00, -3.98286857e-01],
[ 3.45605504e+00,  1.41373901e+00, -4.43933087e-02],
[ 8.52001384e-01, -1.89147259e+00,  3.35617061e+00],
[-1.21264114e+00,  1.23889518e+00,  2.24603050e-01],
[-1.22612520e+00,  1.25538584e+00,  2.28131273e-01],
```

```
       [-1.21265726e+00,  1.23892156e+00,  2.24607699e-01],
       [-5.32920716e-01,  2.85566139e-01,  2.98135833e-03],
       [-1.21263707e+00,  1.23888861e+00,  2.24601885e-01],
       [ 3.45605006e+00,  1.41373498e+00, -4.43930654e-02],
       [ 2.22997119e-01, -1.32594204e+00, -1.34366535e+00],
       [ 3.45605477e+00,  1.41373879e+00, -4.43932955e-02],
       [ 8.52025731e-01, -1.89143875e+00,  3.35615024e+00],
       [-8.53522352e-01,  4.47471846e-01, -3.64626289e-01],
       [ 3.45605078e+00,  1.41373557e+00, -4.43931006e-02],
       [-8.53525603e-01,  4.47475582e-01, -3.64625590e-01],
       [-1.21883358e+00,  1.24624887e+00,  2.26207569e-01],
       [-1.21263564e+00,  1.23888649e+00,  2.24601493e-01],
       [ 8.52003015e-01, -1.89147032e+00,  3.35616925e+00],
       [-1.26319861e+00,  1.28048563e+00,  1.62596567e-01],
       [-1.26319861e+00,  1.28048563e+00,  1.62596567e-01],
       [-1.21263909e+00,  1.23889150e+00,  2.24602430e-01],
       [ 6.54791938e-01, -1.97456590e+00, -1.62691027e+00],
       [ 5.21267028e-01, -1.37693950e+00, -8.25987610e-01],
       [-1.22631410e+00,  1.25562466e+00,  2.28173758e-01],
       [-8.53525962e-01,  4.47476327e-01, -3.64625484e-01],
       [-8.52878420e-01,  4.47991867e-01, -3.64657725e-01],
       [-1.21883515e+00,  1.24625053e+00,  2.26207986e-01],
       [-1.01092168e+00,  9.05266396e-01,  8.33258296e-02],
       [ 6.54792263e-01, -1.97456564e+00, -1.62691029e+00],
       [ 8.51982056e-01, -1.89143201e+00,  3.35624388e+00],
       [ 2.22848803e-01, -1.32606182e+00, -1.34365811e+00],
       [ 8.68581907e-01, -1.96445460e+00,  6.33771859e+00]])
```

[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)
```

```
(140, 3)
```

[17]:

	PC 1	PC 2	PC 3
42	0.222849	-1.326062	-1.343658
NSR	0.696262	-1.169300	-0.312068
TRI	0.654792	-1.974566	-1.626910
CMTC	-0.853523	0.447471	-0.364626
CHAT	0.222854	-1.326058	-1.343658
PURA	-0.555575	0.116552	-0.338081
ADK	-0.929556	0.889837	0.289141
DAPS	0.861758	-1.951923	6.340260
VEIL	0.619078	-1.980335	-1.713640

```
RVC  -1.219944  1.247933  0.226503
```

1.0.3 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]:
```
:Curve   [k]   (inertia)
```

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)
```

```
(140, 9)
```

[19]:

	Algorithm	ProofType	TotalCoinsMined	MaxSupply	PC 1	\
42	Scrypt	PoW/PoS	41.999952	42	0.222849	
NSR	PoS	PoS	6178782525.8373	0	0.696262	
TRI	X13	PoW/PoS	199294.064798	0	0.654792	
CMTC	Scrypt	PoW	872830	0	-0.853523	

```
CHAT    Scrypt       PoW/PoS          1000000000           -1  0.222854
PURA       X11           PoW    188358976.839698           -1 -0.555575
ADK      IMesh           PoW            25000000            0 -0.929556
DAPS    Dagger   PoW/PoS/PoA         62319462900  70000000000  0.861758
VEIL     X16RT       PoW/PoS    119516479.714871    300000000  0.619078
RVC       X16R           PoW  10501536386.860544  21000000000 -1.219944

          PC 2      PC 3           CoinName  Class
42   -1.326062 -1.343658            42 Coin      0
NSR  -1.169300 -0.312068           NuShares      0
TRI  -1.974566 -1.626910     Triangles Coin      0
CMTC  0.447471 -0.364626          CometCoin      2
CHAT -1.326058 -1.343658           OpenChat      0
PURA  0.116552 -0.338081               Pura      2
ADK   0.889837  0.289141       Aidos Kuneen      2
DAPS -1.951923  6.340260          DAPS Coin      3
VEIL -1.980335 -1.713640               VEIL      0
RVC   1.247933  0.226503  Ravencoin Classic      2
```

1.0.4 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
42    2.047619e-12     0.000000e+00         42 Coin      0
NSR   4.761905e-14     6.241194e-06        NuShares      0
TRI   4.761905e-14     2.012647e-10  Triangles Coin      0
CMTC  4.761905e-14     8.816040e-10       CometCoin      2
CHAT  0.000000e+00     1.010101e-06        OpenChat      0
```

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

[21]: :NdOverlay [Class]
:Scatter [TotalCoinsMined] (MaxSupply,CoinName)

Table of Tradable Cryptocurrencies

[22]:
```
# Table with tradable cryptos
clustered_df[
    [
        "CoinName",
        "Algorithm",
        "ProofType",
        "MaxSupply",
        "TotalCoinsMined",
        "Class",
    ]
].hvplot.table()
```

[22]: :Table [CoinName,Algorithm,ProofType,MaxSupply,TotalCoinsMined,Class]

[23]:
```
# Print the total number of tradable cryptocurrencies
print(f"There are {clustered_df.shape[0]} tradable cryptocurrencies.")
```

```
There are 140 tradable cryptocurrencies.
```