File Edit View Run Kernel Tabs Settings Help

crypto_clustering.ipynb

Markdown

Python 3

Clustering Crypto

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

Data Preprocessing

[2]:
```
# Load the cryptocurrencies data
file_path = Path("Resources/crypto_data.csv")
crypto_df = pd.read_csv(file_path, index_col=0)
crypto_df.head(10)
```

[2]:

	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

[3]:
```
# Keep only cryptocurrencies that are on trading
crypto_df = crypto_df[crypto_df["IsTrading"] == True]
print(crypto_df.shape)
crypto_df.head(10)
```

```
(1144, 6)
```

[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

[4]:
```
# Keep only cryptocurrencies with a working algorithm
crypto_df = crypto_df[crypto_df["Algorithm"] != "N/A"]
print(crypto_df.shape)
crypto_df.head(10)
```

```
(1144, 6)
```

[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

[5]:
```
# Remove the "IsTrading" column
crypto_df.drop("IsTrading", axis=1, inplace=True)
print(crypto_df.shape)
crypto_df.head(10)
```

```
(1144, 5)
```

[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

[6]:
```
# 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)
```

```
(685, 5)
```

[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

[7]:
```
# Remove rows with cryptocurrencies without coins mined
crypto_df = crypto_df[crypto_df["TotalCoinsMined"] > 0]
print(crypto_df.shape)
crypto_df.head(10)
```

(532, 5)

[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

[8]:
```
# Fetch the cryptocurrencies names prior to drop them from crypto_df
coins_name = pd.DataFrame(crypto_df["CoinName"], index=crypto_df.index)
print(coins_name.shape)
coins_name.head()
```

(532, 1)

[8]:

	CoinName
42	42 Coin
404	404Coin
1337	EliteCoin
BTC	Bitcoin
ETH	Ethereum

[9]:
```
# Remove the cryptocurrency name 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)
```

(532, 4)

[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

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

(532, 98)

[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	ProofType_PoW/PoS	ProofType_PoW/PoW	ProofType_PoW/nPoS
42	4.199995e+01	42	0	0	0	0	0	0	0	0	...	1	0	0	0
404	1.055185e+09	532000000	0	0	0	0	0	0	0	0	...	1	0	0	0
1337	2.927942e+10	314159265359	0	0	0	0	0	0	0	0	...	1	0	0	0
BTC	1.792718e+07	21000000	0	0	0	0	0	0	0	0	...	0	0	0	0
ETH	1.076842e+08	0	0	0	0	0	0	0	0	0	...	0	0	0	0
LTC	6.303924e+07	84000000	0	0	0	0	0	0	0	0	...	0	0	0	0
DASH	9.031294e+06	22000000	0	0	0	0	0	0	0	0	...	1	0	0	0
XMR	1.720114e+07	0	0	0	0	0	0	0	0	0	...	0	0	0	0
ETC	1.133597e+08	210000000	0	0	0	0	0	0	0	0	...	0	0	0	0
ZEC	7.383056e+06	21000000	0	0	0	0	0	0	0	0	...	0	0	0	0

10 rows × 98 columns

[11]:
```
# Standardize data
X = StandardScaler().fit_transform(X)
X[:5]
```

[11]:
```
array([[-0.11710817, -0.1528703 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.06142951, -0.07530656, -0.0433963 , -0.06142951, -0.06142951,
        -0.0433963 , -0.0433963 , -0.19245009, -0.06142951, -0.09740465,
        -0.0433963 , -0.11547005, -0.07530656, -0.0433963 , -0.0433963 ,
        -0.15191091, -0.0433963 , -0.13118084, -0.0433963 , -0.0433963 ,
        -0.08703883, -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.06142951, -0.0433963 , -0.08703883, -0.08703883, -0.08703883,
        -0.0433963 , -0.13118084, -0.13840913, -0.13840913, -0.0433963 ,
        -0.06142951, -0.0433963 , -0.07530656, -0.18168574, -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.07530656, -0.15826614, -0.31491833,
        -0.0433963 , -0.08703883, -0.07530656, -0.06142951,  1.38675049,
        -0.0433963 , -0.0433963 , -0.06142951, -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.39879994, -0.0433963 , -0.18168574, -0.0433963 ,
        -0.08703883, -0.08703883, -0.10680283, -0.0433963 , -0.13118084,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.07530656,
        -0.43911856, -0.0433963 , -0.06142951, -0.0433963 , -0.0433963 ,
        -0.89632016, -0.0433963 , -0.0433963 ,  1.42222617, -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 ],
       [-0.09396955, -0.145009  , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.06142951, -0.07530656, -0.0433963 , -0.06142951, -0.06142951,
        -0.0433963 , -0.0433963 , -0.19245009, -0.06142951, -0.09740465,
        -0.0433963 , -0.11547005, -0.07530656, -0.0433963 , -0.0433963 ,
        -0.15191091, -0.0433963 , -0.13118084, -0.0433963 , -0.0433963 ,
        -0.08703883, -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.06142951, -0.0433963 , -0.08703883, -0.08703883, -0.08703883,
        -0.0433963 , -0.13118084, -0.13840913, -0.13840913, -0.0433963 ,
        -0.06142951, -0.0433963 , -0.07530656, -0.18168574, -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.07530656, -0.15826614, -0.31491833,
        -0.0433963 , -0.08703883, -0.07530656, -0.06142951,  1.38675049,
        -0.0433963 , -0.0433963 , -0.06142951, -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.39879994, -0.0433963 , -0.18168574, -0.0433963 ,
        -0.08703883, -0.08703883, -0.10680283, -0.0433963 , -0.13118084,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.07530656,
        -0.43911856, -0.0433963 , -0.06142951, -0.0433963 , -0.0433963 ,
        -0.89632016, -0.0433963 , -0.0433963 ,  1.42222617, -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 ],
       [ 0.52494561,  4.48942416, -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.06142951, -0.07530656, -0.0433963 , -0.06142951, -0.06142951,
        -0.0433963 , -0.0433963 , -0.19245009, -0.06142951, -0.09740465,
        -0.0433963 , -0.11547005, -0.07530656, -0.0433963 , -0.0433963 ,
        -0.15191091, -0.0433963 , -0.13118084, -0.0433963 , -0.0433963 ,
        -0.08703883, -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.06142951, -0.0433963 , -0.08703883, -0.08703883, -0.08703883,
        -0.0433963 , -0.13118084, -0.13840913, -0.13840913, -0.0433963 ,
        -0.06142951, -0.0433963 , -0.07530656, -0.18168574, -0.0433963 ,
```

```
        -0.0433963 , -0.0433963 , -0.07530656, -0.15826614, -0.31491833,
        -0.0433963 , -0.08703883, -0.07530656, -0.06142951, -0.72111026,
        -0.0433963 , -0.0433963 , -0.06142951, -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.39879994, -0.0433963 ,  5.50400923, -0.0433963 ,
        -0.08703883, -0.08703883, -0.10680283, -0.0433963 , -0.13118084,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.07530656,
        -0.43911856, -0.0433963 , -0.06142951, -0.0433963 , -0.0433963 ,
        -0.89632016, -0.0433963 , -0.0433963 ,  1.42222617, -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 ],
       [-0.11671506, -0.15255998, -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.06142951, -0.07530656, -0.0433963 , -0.06142951, -0.06142951,
        -0.0433963 , -0.0433963 , -0.19245009, -0.06142951, -0.09740465,
        -0.0433963 , -0.11547005, -0.07530656, -0.0433963 , -0.0433963 ,
        -0.15191091, -0.0433963 , -0.13118084, -0.0433963 , -0.0433963 ,
        -0.08703883, -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.06142951, -0.0433963 , -0.08703883, -0.08703883, -0.08703883,
        -0.0433963 , -0.13118084, -0.13840913, -0.13840913, -0.0433963 ,
        -0.06142951, -0.0433963 , -0.07530656, -0.18168574, -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.07530656, -0.15826614,  3.17542648,
        -0.0433963 , -0.08703883, -0.07530656, -0.06142951, -0.72111026,
        -0.0433963 , -0.0433963 , -0.06142951, -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.39879994, -0.0433963 , -0.18168574, -0.0433963 ,
        -0.08703883, -0.08703883, -0.10680283, -0.0433963 , -0.13118084,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.07530656,
        -0.43911856, -0.0433963 , -0.06142951, -0.0433963 , -0.0433963 ,
         1.11567277, -0.0433963 , -0.0433963 , -0.70312305, -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 ],
       [-0.11474682, -0.1528703 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.06142951, -0.07530656, -0.0433963 , -0.06142951, -0.06142951,
        -0.0433963 , -0.0433963 , -0.19245009, -0.06142951, -0.09740465,
        -0.0433963 , -0.11547005, -0.07530656, -0.0433963 , -0.0433963 ,
        -0.15191091, -0.0433963 ,  7.62306442, -0.0433963 , -0.0433963 ,
        -0.08703883, -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.06142951, -0.0433963 , -0.08703883, -0.08703883, -0.08703883,
        -0.0433963 , -0.13118084, -0.13840913, -0.13840913, -0.0433963 ,
        -0.06142951, -0.0433963 , -0.07530656, -0.18168574, -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.07530656, -0.15826614, -0.31491833,
        -0.0433963 , -0.08703883, -0.07530656, -0.06142951, -0.72111026,
        -0.0433963 , -0.0433963 , -0.06142951, -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.39879994, -0.0433963 , -0.18168574, -0.0433963 ,
        -0.08703883, -0.08703883, -0.10680283, -0.0433963 , -0.13118084,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.07530656,
        -0.43911856, -0.0433963 , -0.06142951, -0.0433963 , -0.0433963 ,
         1.11567277, -0.0433963 , -0.0433963 , -0.70312305, -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 , -0.0433963 ,
        -0.0433963 , -0.0433963 , -0.0433963 ]])
```

Reducing Dimensions Using PCA

[12]:
```
# 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
```

[12]:
```
array([[-0.32847997,  1.02135677, -0.44346298],
       [-0.31180942,  1.02173395, -0.44349819],
       [ 2.29116598,  1.78857593, -0.44741114],
       ...,
       [ 0.31559174, -2.27676317,  0.3076426 ],
       [-0.16999894, -2.15805826,  0.12596361],
       [-0.27958835,  0.79941936, -0.21219861]])
```

[13]:
```
# 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)
```

```
(532, 3)
```

[13]:

	PC 1	PC 2	PC 3
42	-0.328480	1.021357	-0.443463
404	-0.311809	1.021734	-0.443498
1337	2.291166	1.788576	-0.447411
BTC	-0.151399	-1.304196	0.144518
ETH	-0.147775	-2.066985	0.258884
LTC	-0.156849	-1.176076	-0.061527
DASH	-0.399348	1.377149	-0.292472
XMR	-0.148892	-2.229988	0.263554
ETC	-0.146217	-2.067063	0.258882
ZEC	-0.169998	-2.158058	0.125964

Clustering Crytocurrencies Using K-Means

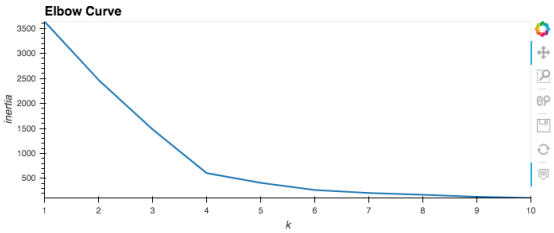
Find the Best Value for `k` Using the Elbow Curve

[14]:
```
inertia = []
k = list(range(1, 11))

# Calculate the inertia for the range ok 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")
```

[14]:



Running K-Means with `k=4`

[15]:
```
# 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)
```

```
(532, 9)
```

	Algorithm	ProofType	TotalCoinsMined	TotalCoinSupply	PC 1	PC 2	PC 3	CoinName	Class
42	Scrypt	PoW/PoS	4.199995e+01	42	-0.328480	1.021357	-0.443463	42 Coin	0
404	Scrypt	PoW/PoS	1.055185e+09	532000000	-0.311809	1.021734	-0.443498	404Coin	0
1337	X13	PoW/PoS	2.927942e+10	314159265359	2.291166	1.788576	-0.447411	EliteCoin	0
BTC	SHA-256	PoW	1.792718e+07	21000000	-0.151399	-1.304196	0.144518	Bitcoin	3
ETH	Ethash	PoW	1.076842e+08	0	-0.147775	-2.066985	0.258884	Ethereum	3
LTC	Scrypt	PoW	6.303924e+07	84000000	-0.156849	-1.176076	-0.061527	Litecoin	3
DASH	X11	PoW/PoS	9.031294e+06	22000000	-0.399348	1.377149	-0.292472	Dash	0
XMR	CryptoNight-V7	PoW	1.720114e+07	0	-0.148892	-2.229988	0.263554	Monero	3
ETC	Ethash	PoW	1.133597e+08	210000000	-0.146217	-2.067063	0.258882	Ethereum Classic	3
ZEC	Equihash	PoW	7.383056e+06	21000000	-0.169998	-2.158058	0.125964	ZCash	3

Visualizing Results

3D-Scatter with Clusters

```
# Create a 3D-Scatter with the PCA data and the clusters
fig = px.scatter_3d(
    clustered_df,
    x="PC 1",
    y="PC 2",
    z="PC 3",
    color="Class",
    symbol="Class",
    hover_name="CoinName",
    hover_data=["Algorithm"],
    width=800,
)
fig.update_layout(legend=dict(x=0, y=1))
fig.show()
```

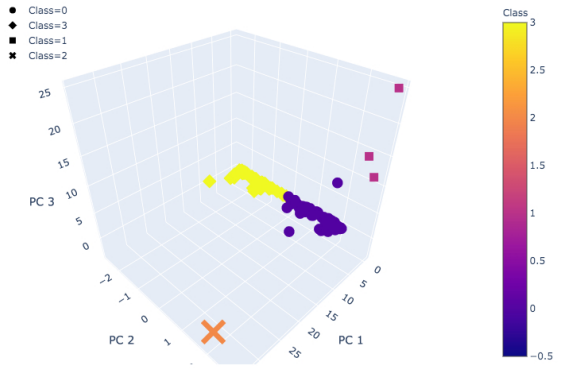


Table of Tradable Cryptocurrencies

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

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

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

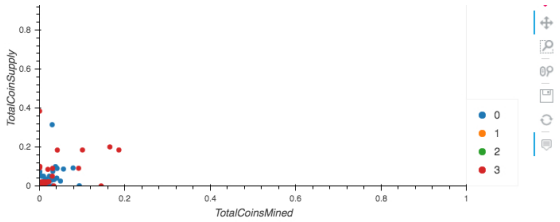
```
There are 532 tradable cryptocurrencies.
```

Scatter Plot with Tradable Cryptocurrencies

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

	TotalCoinSupply	TotalCoinsMined	CoinName	Class
42	4.200000e-11	0.000000	42 Coin	0
404	5.320000e-04	0.001066	404Coin	0
1337	3.141593e-01	0.029576	EliteCoin	0
BTC	2.100000e-05	0.000018	Bitcoin	3
ETH	0.000000e+00	0.000109	Ethereum	3

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


TotalCoinSupply
TotalCoinsMined
0
0.2
0.4
0.6
0.8
1
0
1
2
3