Clustering Crypto

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

Fetching Cryptocurrency Data

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

In [3]:
```
# Create a DataFrame
# HINT: You will need to use the 'Data' key from the json response, then transpose the DataFrame.
```

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

# Create a DataFrame
df = pd.read_csv(file_path)
```

Data Preprocessing

In [5]:
```
# Keep only necessary columns:
# 'CoinName','Algorithm','IsTrading','ProofType','TotalCoinsMined','TotalCoinSupply'
df = df.drop(columns=['Unnamed: 0'])
```

In [6]:
```
# Keep only cryptocurrencies that are trading
df = df.loc[df['IsTrading'] == True]
```

In [7]:
```
# Keep only cryptocurrencies with a working algorithm
df = df.loc[df['Algorithm'] != 'N/A']
```

In [8]:
```
# Remove the "IsTrading" column
df = df.drop(columns=['IsTrading'])
```

In [9]:
```
# Remove rows with at least 1 null value
df = df.dropna()
```

In [10]:
```
# Remove rows with cryptocurrencies having no coins mined
df = df.loc[df['TotalCoinsMined'] != 0]
```

In [11]:
```
# Drop rows where there are 'N/A' text values
df = df.loc[df['TotalCoinSupply'] != 'N/A']
df = df.loc[df['ProofType'] != 'N/A']
```

In [12]:
```
# Store the 'CoinName'column in its own DataFrame prior to dropping it from crypto_df
coin_name = df.copy()
coin_name = coin_name.drop(columns=['Algorithm', 'ProofType', 'TotalCoinsMined', 'TotalCoinSupply'])
coin_name.head()
```

Out[12]:

	CoinName
0	42 Coin
2	404Coin
5	EliteCoin
7	Bitcoin
8	Ethereum

In [13]:
```
# Drop the 'CoinName' column since it's not going to be used on the clustering algorithm
crypto_df = df.drop(columns=['CoinName'])
df.head()
```

Out[13]:

	CoinName	Algorithm	ProofType	TotalCoinsMined	TotalCoinSupply
0	42 Coin	Scrypt	PoW/PoS	4.199995e+01	42
2	404Coin	Scrypt	PoW/PoS	1.055185e+09	532000000
5	EliteCoin	X13	PoW/PoS	2.927942e+10	314159265359
7	Bitcoin	SHA-256	PoW	1.792718e+07	21000000
8	Ethereum	Ethash	PoW	1.076842e+08	0

In [14]:
```
# Create dummy variables for text features
df_encoded = pd.get_dummies(data=df, columns=['CoinName', 'Algorithm', 'ProofType'])
```

In [15]:
```
# Standardize data
crypto_df_scaled = StandardScaler().fit_transform(df_encoded)
```

Reducing Dimensions Using PCA

In [16]:
```
# Use PCA to reduce dimensions to 3 principal components
pca = PCA(n_components=3)
crypto_pca = pca.fit_transform(crypto_df_scaled)
```

In [17]:
```
# Create a DataFrame with the principal components data
pca_df = pd.DataFrame(data=crypto_pca, columns=["PC1", "PC2", "PC3"], index=df.index)
pca_df.head()
```

Out[17]:

	PC1	PC2	PC3
0	-0.367977	1.508842	-0.415745
2	-0.347439	1.505998	-0.415285
5	2.777346	2.374562	-0.364379
7	-0.182963	-1.706033	0.091023
8	-0.156623	-2.367520	0.134513

Clustering Crytocurrencies Using K-Means

Find the Best Value for k Using the Elbow Curve

In [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(pca_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 Visualization", xlabel="Number of Clusters", ylabel="Inertia")
```

Out[18]:

Running K-Means with k=7

In [19]:
```
# Initialize the K-Means model
kmodel = KMeans(n_clusters=7, random_state=0)
# Fit the model
kmodel.fit(pca_df)
# Predict clusters
predictions = kmodel.predict(pca_df)
# Create a new DataFrame including predicted clusters and cryptocurrencies features
clustered_df = pd.DataFrame({"CoinName": coin_name.CoinName,
                             "Algorithm": df.Algorithm,
                             "ProofType": df.ProofType,
                             "TotalCoinsMined": df.TotalCoinsMined,
                             "TotalCoinSupply": df.TotalCoinSupply,
                             "PC 1": pca_df['PC1'],
                             "PC 2": pca_df['PC2'],
                             "PC 3": pca_df['PC3'],
                             "Class": kmodel.labels_,
                             "Predictions": predictions})
clustered_df.head(10)
```

Out[19]:

	CoinName	Algorithm	ProofType	TotalCoinsMined	TotalCoinSupply	PC 1	PC 2	PC 3	Class	Predictions
0	42 Coin	Scrypt	PoW/PoS	4.199995e+01	42	-0.367977	1.508842	-0.415745	0	0
2	404Coin	Scrypt	PoW/PoS	1.055185e+09	532000000	-0.347439	1.505998	-0.415285	0	0
5	EliteCoin	X13	PoW/PoS	2.927942e+10	314159265359	2.777346	2.374562	-0.364379	6	6
7	Bitcoin	SHA-256	PoW	1.792718e+07	21000000	-0.182963	-1.706033	0.091023	5	5
8	Ethereum	Ethash	PoW	1.076842e+08	0	-0.156623	-2.367520	0.134513	1	1
9	Litecoin	Scrypt	PoW	6.303924e+07	84000000	-0.187302	-1.265171	-0.071848	5	5
10	Dash	X11	PoW/PoS	9.031294e+06	22000000	-0.420389	1.697986	-0.387648	0	0
11	Monero	CryptoNight-V7	PoW	1.720114e+07	0	-0.115673	-2.574003	0.109935	1	1
12	Ethereum Classic	Ethash	PoW	1.133597e+08	210000000	-0.154049	-2.366668	0.134710	1	1
13	ZCash	Equihash	PoW	7.383056e+06	21000000	-0.112817	-2.040530	0.259027	1	1

Visualizing Results

3D-Scatter with Clusters

In [25]:
```
# 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",
    hover_name= 'CoinName',
    hover_data= ['Algorithm'],
    color="Class",
    symbol="Class")
fig.update_layout(legend=dict(x=0, y=1))
fig.show()
plt.savefig('crypto_scatter.png')
```

```
<Figure size 432x288 with 0 Axes>
```

Table of Tradable Cryptocurrencies

In [28]:
```
# Table with tradable cryptos
columns = ['CoinName', 'Algorithm', 'ProofType', 'TotalCoinSupply', 'TotalCoinsMined', 'Class']
tradeables = clustered_df.hvplot.table(columns)
tradeables
hvplot.save(tradeables, 'tradeable_cryptos.png', fmt='png')
```

In [22]:
```
# Print the total number of tradable cryptocurrencies
number_of_rows = len(tradeables)
print(f"Total number of tradeable coins are {number_of_rows}.")
```

```
Total number of tradeable coins are 533.
```

Scatter Plot with Tradable Cryptocurrencies

In [23]:
```
# Scale data to create the scatter plot
clustered_df['TotalCoinsMined / 1B'] = clustered_df['TotalCoinsMined'].astype(float) / 1000000000
clustered_df['TotalCoinSupply / 1B'] = clustered_df['TotalCoinSupply'].astype(float) / 1000000000
```

In [24]:
```
# Plot the scatter with x="TotalCoinsMined" and y="TotalCoinSupply"
clustered_df.hvplot(
    kind="scatter",
    x="TotalCoinsMined",
    y="TotalCoinSupply",
    c='Class',
    colormap="Inferno_r",
    hover_cols=['CoinName']
)
```

```
/Users/erenturkmenoglu/opt/anaconda3/lib/python3.7/site-packages/holoviews/plotting/util.py:697: MatplotlibDeprecationWarning:

The global colormaps dictionary is no longer considered public API.

/Users/erenturkmenoglu/opt/anaconda3/lib/python3.7/site-packages/holoviews/plotting/util.py:697: MatplotlibDeprecationWarning:

The global colormaps dictionary is no longer considered public API.
```

Out[24]:

In [ ]:
```
```