o3tddztce

January 19, 2025

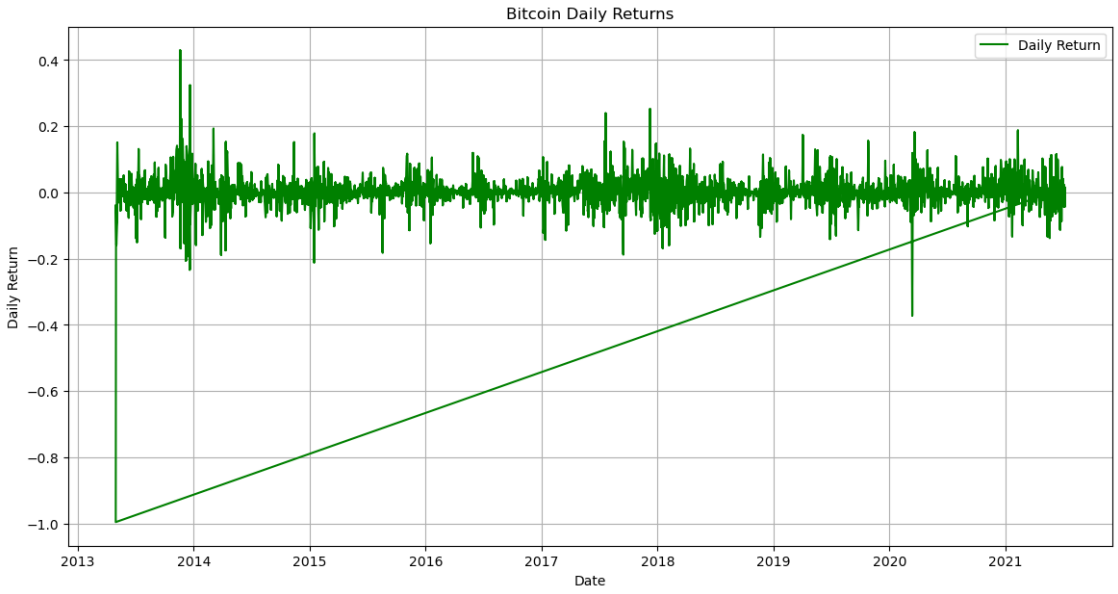
[33]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pymongo import MongoClient
import requests
from datetime import datetime
import json

# Configure MongoDB connection
mongo_client = MongoClient("mongodb+srv://ismini6:9LogpPCZJUJEZyjl@cluster0.
 ↪8xihe.mongodb.net/")
db = mongo_client['Crypto']
collection = db['Cryptocurrency']

# Data Ingestion
def load_data(file_path):
    df = pd.read_csv(file_path)
    return df

# Load all datasets from the 'archive' folder
import os

data_folder = 'C:/Users/Nantia/Downloads/archive/'
crypto_files = [f for f in os.listdir(data_folder) if f.endswith('.csv')]
crypto_data = pd.DataFrame()

for file in crypto_files:
    file_path = os.path.join(data_folder, file)
    data = load_data(file_path)
    crypto_data = pd.concat([crypto_data, data], ignore_index=True)

print(crypto_data.head())

# Data Cleaning and Transformation
crypto_data['Date'] = pd.to_datetime(crypto_data['Date'])
```

```
# Drop duplicates and handle missing values
crypto_data.drop_duplicates(inplace=True)
crypto_data.fillna(method='ffill', inplace=True)

# Store cleaned data to MongoDB
collection.insert_many(crypto_data.to_dict('records'))

# Relevant Queries for Data Extraction
# Example query: Get all data for Bitcoin
bitcoin_data = list(collection.find({"Symbol": "BTC"}))
bitcoin_df = pd.DataFrame(bitcoin_data)

# Convert 'Date' back to datetime for plotting
bitcoin_df['Date'] = pd.to_datetime(bitcoin_df['Date'])

# Analyzing Price Movements
plt.figure(figsize=(14, 7))
plt.plot(bitcoin_df['Date'], bitcoin_df['Close'], label='Close Price',␣
 ↪color='blue')
plt.title('Bitcoin Price Movements Over Time')
plt.xlabel('Date')
plt.ylabel('Price in USD')
plt.legend()
plt.grid()
plt.show()

# Big Data Processing Solution (Batch Analysis)

bitcoin_df['Daily Return'] = bitcoin_df['Close'].pct_change()

# Visualize Daily Returns
plt.figure(figsize=(14, 7))
plt.plot(bitcoin_df['Date'], bitcoin_df['Daily Return'], label='Daily Return',␣
 ↪color='green')
plt.title('Bitcoin Daily Returns')
plt.xlabel('Date')
plt.ylabel('Daily Return')
plt.legend()
plt.grid()
plt.show()
```

```
C:\Users\Nantia\anaconda3\Lib\site-packages\cryptography\x509\base.py:594:
CryptographyDeprecationWarning:

Parsed a negative serial number, which is disallowed by RFC 5280.

   SNo  Name Symbol                 Date        High         Low        Open  \
```

0	1	Aave	AAVE	2020-10-05 23:59:59	55.112358	49.787900	52.675035
1	2	Aave	AAVE	2020-10-06 23:59:59	53.402270	40.734578	53.291969
2	3	Aave	AAVE	2020-10-07 23:59:59	42.408314	35.970690	42.399947
3	4	Aave	AAVE	2020-10-08 23:59:59	44.902511	36.696057	39.885262
4	5	Aave	AAVE	2020-10-09 23:59:59	47.569533	43.291776	43.764463

	Close	Volume	Marketcap
0	53.219243	0.000000e+00	8.912813e+07
1	42.401599	5.830915e+05	7.101144e+07
2	40.083976	6.828342e+05	6.713004e+07
3	43.764463	1.658817e+06	2.202651e+08
4	46.817744	8.155377e+05	2.356322e+08



[7]:
```python
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import glob
import os
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Download VADER sentiment analysis model from nltk
nltk.download('vader_lexicon')

# Define the path to your dataset folder
data_folder = 'C:/Users/Nantia/Downloads/archive/'

# Load all CSV files into a single DataFrame
all_files = glob.glob(os.path.join(data_folder, "*.csv"))
dfs = []

for filename in all_files:
    df = pd.read_csv(filename)
    dfs.append(df)
```

```
# Concatenate all DataFrames into one
data = pd.concat(dfs, ignore_index=True)

# Display the first few rows
data.head()
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\Nantia\AppData\Roaming\nltk_data…
[nltk_data]   Package vader_lexicon is already up-to-date!
```

[7]:

	SNo	Name	Symbol	Date	High	Low	Open	Close	Volume	Marketcap
0	1	Aave	AAVE	2020-10-05 23:59:59	55.112358	49.787900	52.675035	53.219243	0.000000e+00	8.912813e+07
1	2	Aave	AAVE	2020-10-06 23:59:59	53.402270	40.734578	53.291969	42.401599	5.830915e+05	7.101144e+07
2	3	Aave	AAVE	2020-10-07 23:59:59	42.408314	35.970690	42.399947	40.083976	6.828342e+05	6.713004e+07
3	4	Aave	AAVE	2020-10-08 23:59:59	44.902511	36.696057	39.885262	43.764463	1.658817e+06	2.202651e+08
4	5	Aave	AAVE	2020-10-09 23:59:59	47.569533	43.291776	43.764463	46.817744	8.155377e+05	2.356322e+08

[8]:
```
# Check for missing values
print(data.isnull().sum())

# Fill or drop missing values
data.dropna(inplace=True)

# Convert 'Date' column to datetime type
data['Date'] = pd.to_datetime(data['Date'])

# Reset index
data.reset_index(drop=True, inplace=True)

# Display cleaned data info
data.info()
```

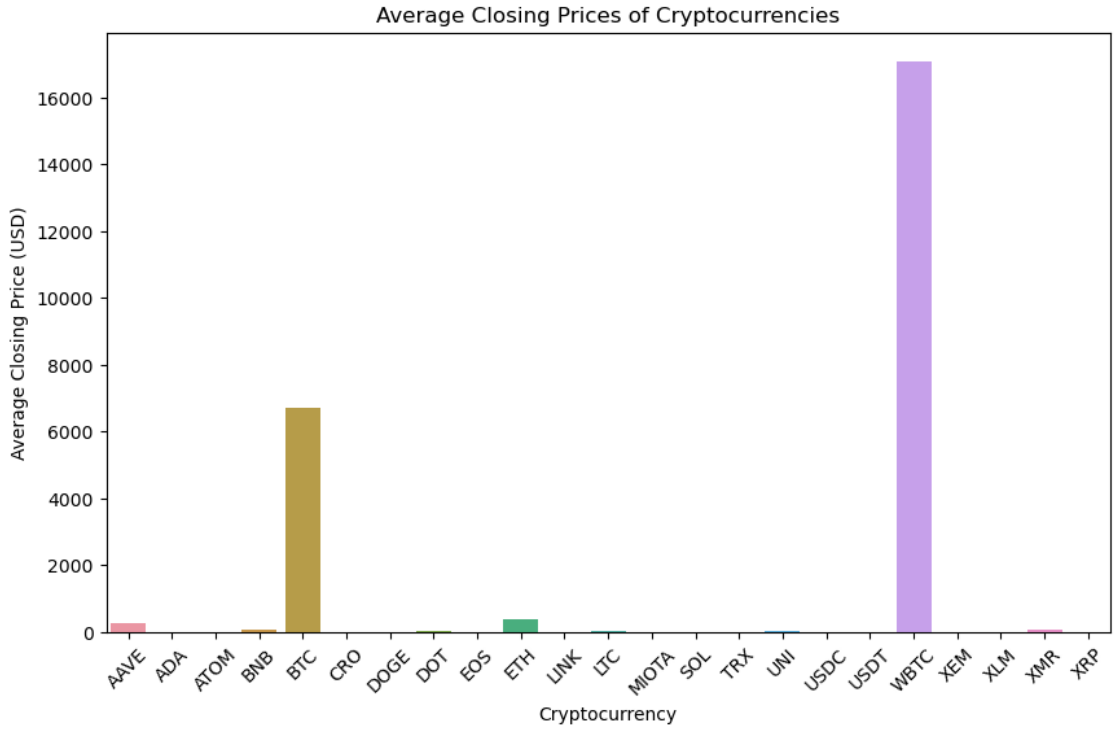
```
SNo          0
Name         0
Symbol       0
Date         0
High         0
Low          0
Open         0
Close        0
Volume       0
```

```
Marketcap    0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37082 entries, 0 to 37081
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   SNo        37082 non-null  int64
 1   Name       37082 non-null  object
 2   Symbol     37082 non-null  object
 3   Date       37082 non-null  datetime64[ns]
 4   High       37082 non-null  float64
 5   Low        37082 non-null  float64
 6   Open       37082 non-null  float64
 7   Close      37082 non-null  float64
 8   Volume     37082 non-null  float64
 9   Marketcap  37082 non-null  float64
dtypes: datetime64[ns](1), float64(6), int64(1), object(2)
memory usage: 2.8+ MB
```

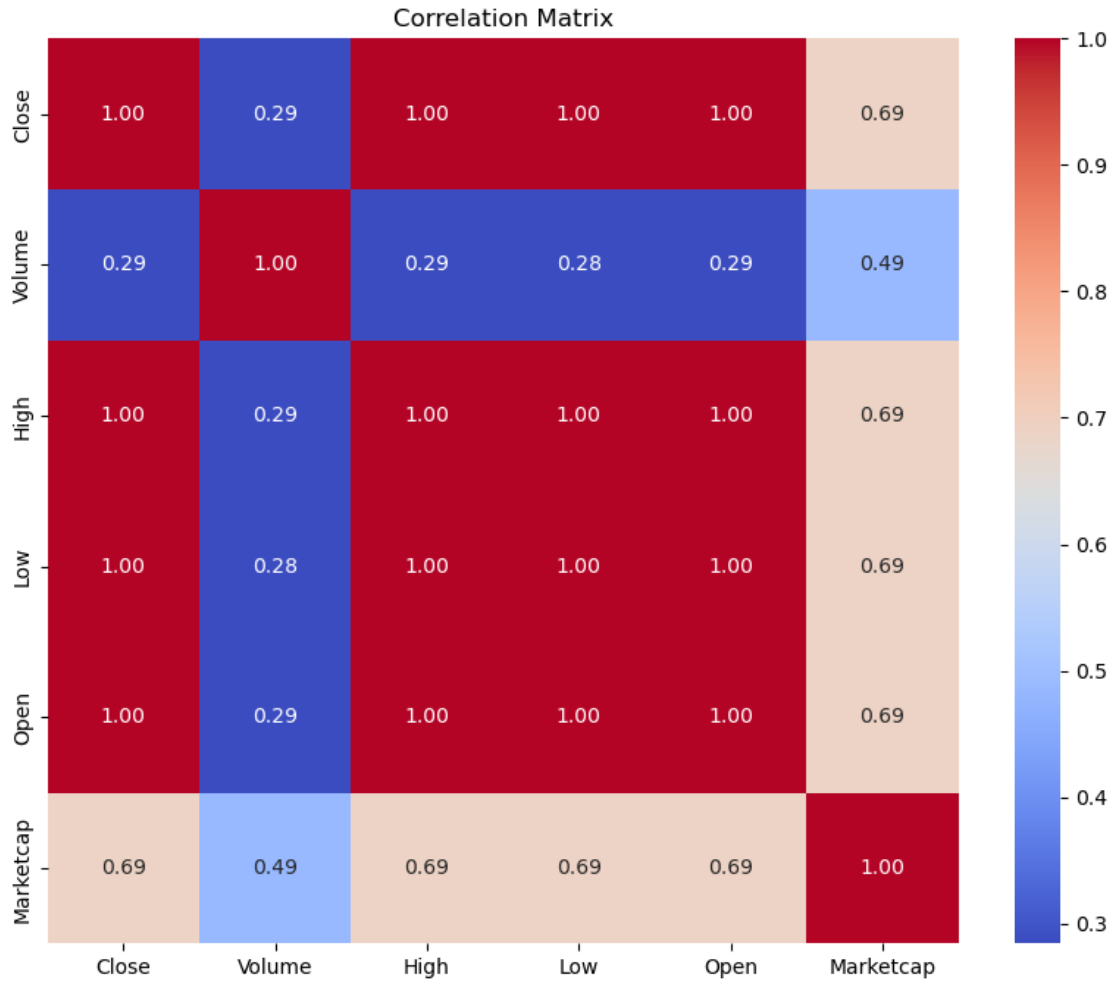
[9]:
```python
# Example: Analyzing Average Closing Price for each cryptocurrency
avg_close = data.groupby('Symbol')['Close'].mean().reset_index()
print(avg_close)

# Visualizing average closing prices
plt.figure(figsize=(10, 6))
sns.barplot(data=avg_close, x='Symbol', y='Close')
plt.title('Average Closing Prices of Cryptocurrencies')
plt.xlabel('Cryptocurrency')
plt.ylabel('Average Closing Price (USD)')
plt.xticks(rotation=45)
plt.show()
```

```
   Symbol         Close
0    AAVE    255.525845
1     ADA      0.256313
2    ATOM      6.768099
3     BNB     52.250308
4     BTC   6711.290443
5     CRO      0.081912
6    DOGE      0.013763
7     DOT     18.143080
8     EOS      4.624088
9     ETH    383.910691
10   LINK      6.308583
11    LTC     49.279008
12  MIOTA      0.729370
13    SOL     10.471388
```

```
14     TRX       0.032585
15     UNI      17.077256
16    USDC       1.003791
17    USDT       1.000696
18    WBTC   17086.573875
19     XEM       0.124662
20     XLM       0.101509
21     XMR      74.134773
22     XRP       0.234790
```



[10]:
```python
# Correlation Matrix
correlation = data[['Close', 'Volume', 'High', 'Low', 'Open', 'Marketcap']].
 ↪corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```



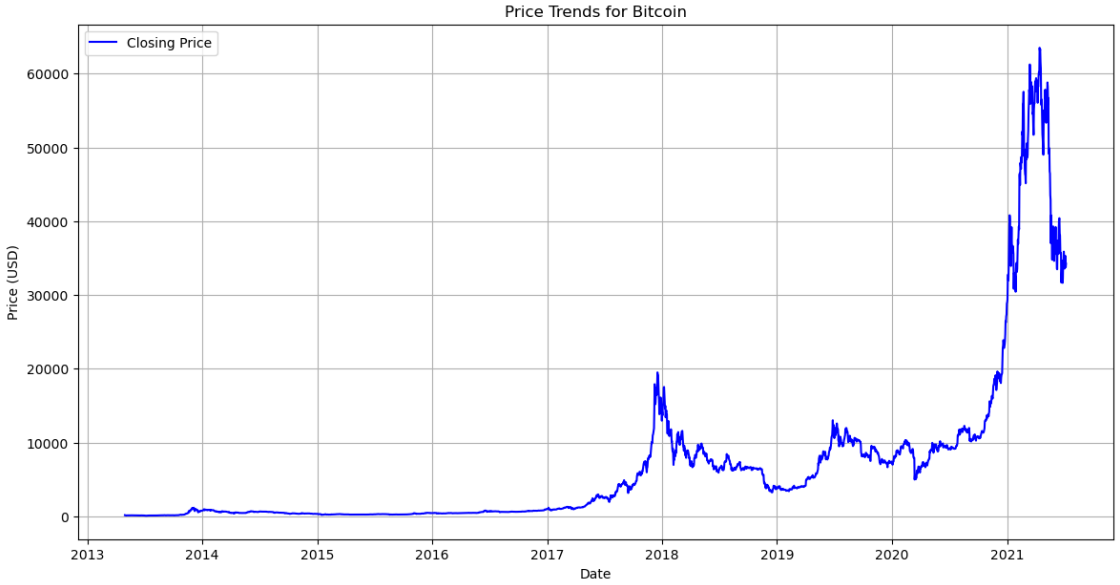
[11]:
```
# Prepare data for regression model
X = data[['Volume', 'High', 'Low', 'Open']]
y = data['Close']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Create and fit the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
```

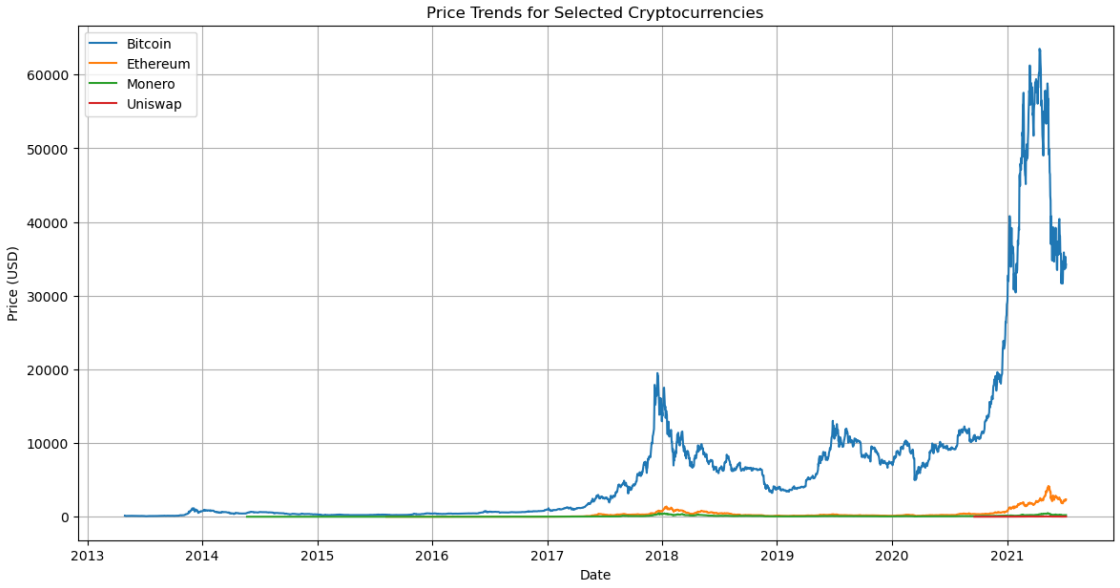
```
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 15899.032311721452
```

[28]:
```
import pandas as pd
import os

# Function to load CSV files from a directory
def load_crypto_data(folder_path):
    data = {}
    for filename in os.listdir(folder_path):
        if filename.endswith('.csv'):
            filepath = os.path.join(folder_path, filename)
            # Load the data into a DataFrame
            df = pd.read_csv(filepath)
            # Store DataFrame in a dictionary with the cryptocurrency name as␣
 ↪the key
            crypto_name = df['Name'][0]  # Assumes the first entry represents␣
 ↪the crypto
            data[crypto_name] = df
    return data

# Load data from the "archive" folder
data_folder = 'C:/Users/Nantia/Downloads/archive/'
crypto_data = load_crypto_data(data_folder)

# Display the keys of loaded data
crypto_data.keys()

# Function to clean and preprocess the data
def clean_crypto_data(df):
    # Convert 'Date' to datetime format
    df['Date'] = pd.to_datetime(df['Date'])

    # Drop unnecessary columns if needed
    df.drop(columns=['SNo', 'Name', 'Symbol'], inplace=True)

    # Rename columns for consistency
    df.rename(columns={'High': 'high', 'Low': 'low', 'Open': 'open',
                       'Close': 'close', 'Volume': 'volume', 'Marketcap':␣
 ↪'marketcap'}, inplace=True)

    # Handle missing values
    df.fillna(method='ffill', inplace=True)

    return df
```

```
# Clean all loaded data
cleaned_crypto_data = {name: clean_crypto_data(df) for name, df in crypto_data.
 ↪items()}

import matplotlib.pyplot as plt
import seaborn as sns

# Function to visualize price trends
def plot_price_trends(df, crypto_name):
    plt.figure(figsize=(14, 7))
    plt.plot(df['Date'], df['close'], label='Closing Price', color='blue')
    plt.title(f'Price Trends for {crypto_name}')
    plt.xlabel('Date')
    plt.ylabel('Price (USD)')
    plt.legend()
    plt.grid()
    plt.show()

# Plotting price trends for Bitcoin as an example
if 'Bitcoin' in cleaned_crypto_data:
    plot_price_trends(cleaned_crypto_data['Bitcoin'], 'Bitcoin')

# Function to visualize price trends for multiple cryptocurrencies
def plot_multiple_price_trends(cleaned_data, crypto_names):
    plt.figure(figsize=(14, 7))

    for crypto_name in crypto_names:
        if crypto_name in cleaned_data:
            df = cleaned_data[crypto_name]
            plt.plot(df['Date'], df['close'], label=crypto_name)

    plt.title('Price Trends for Selected Cryptocurrencies')
    plt.xlabel('Date')
    plt.ylabel('Price (USD)')
    plt.legend()
    plt.grid()
    plt.show()

# List of cryptocurrencies to plot
cryptocurrencies_to_plot = ['Bitcoin','Ethereum', 'Monero', 'Uniswap']

# Plotting price trends for selected cryptocurrencies
plot_multiple_price_trends(cleaned_crypto_data, cryptocurrencies_to_plot)
```





Explanation: MongoDB Connection: Connects to your online MongoDB database using the connection string. Data Loading Function: Reads all CSV files from a specified directory and combines them into a single DataFrame. MongoDB Insertion: Converts the DataFrame to a dictionary format compatible with MongoDB and inserts it into the specified collection.

Explanation: Data Cleaning: Removes rows with null values and converts the Date column into a datetime format. Feature Engineering: Adds a new column for daily returns, which quantifies price changes.

Volume: Handling large datasets may require batch loading and optimized queries. Velocity: To address real-time data influx, consider implementing a streaming solution (e.g., using Apache Kafka). Variety: Different data formats (structured and unstructured) necessitate diverse processing techniques. For instance, you could incorporate sentiment analysis from social media.

[27]:
```
from itertools import cycle
import plotly.express as px
Monero = pd.read_csv("C:/Users/Nantia/Downloads/archive/coin_Monero.csv")
Monero
projection_Monero = 5
#creation of a new column with a name prediction
Monero['Prediction'] = Monero[['Close']].shift(-projection_Monero)
Monero
visualize_Monero = cycle(['Open','Close','High','Low','Prediction'])

fig = px.line(Monero, x=Monero.Date, y=[Monero['Open'], Monero['Close'],
                                        Monero['High'], Monero['Low'],␣
 ↪Monero['Prediction']],
              labels={'Date': 'Date','value':'Price'})
fig.update_layout(title_text='Monero', font_size=15,␣
 ↪font_color='black',legend_title_text='Parameters')
fig.for_each_trace(lambda t:  t.update(name = next(visualize_Monero)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```

[31]:
```
from itertools import cycle
import plotly.express as px
Bitcoin = pd.read_csv("C:/Users/Nantia/Downloads/archive/coin_Bitcoin.csv")
Bitcoin
projection_Bitcoin = 5
#creation of a new column with a name prediction
Bitcoin['Prediction'] = Bitcoin[['Close']].shift(-projection_Bitcoin)
Bitcoin
visualize_Bitcoin = cycle(['Open','Close','High','Low','Prediction'])

fig = px.line(Bitcoin, x=Bitcoin.Date, y=[Bitcoin['Open'], Bitcoin['Close'],
                                        Bitcoin['High'], Bitcoin['Low'],␣
 ↪Bitcoin['Prediction']],
              labels={'Date': 'Date','value':'Price'})
fig.update_layout(title_text='Bitcoin', font_size=15,␣
 ↪font_color='black',legend_title_text='Parameters')
fig.for_each_trace(lambda t:  t.update(name = next(visualize_Bitcoin)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
```

```
fig.show()
```

[30]:
```
from itertools import cycle
import plotly.express as px
Ethereum = pd.read_csv("C:/Users/Nantia/Downloads/archive/coin_Ethereum.csv")
Ethereum
projection_Ethereum = 5
#creation of a new column with a name prediction
Ethereum['Prediction'] = Ethereum[['Close']].shift(-projection_Ethereum)
Ethereum
visualize_Ethereum= cycle(['Open','Close','High','Low','Prediction'])

fig = px.line(Ethereum, x=Ethereum.Date, y=[Ethereum['Open'], Ethereum['Close'],
                                            Ethereum['High'], Ethereum['Low'],␣
 ↪Ethereum['Prediction']],
              labels={'Date': 'Date','value':'Price'})
fig.update_layout(title_text='Bitcoin', font_size=15,␣
 ↪font_color='black',legend_title_text='Parameters')
fig.for_each_trace(lambda t:  t.update(name = next(visualize_Ethereum)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```

[32]:
```
from itertools import cycle
import plotly.express as px
Uniswap = pd.read_csv("C:/Users/Nantia/Downloads/archive/coin_Uniswap.csv")
Uniswap
projection_Uniswap = 5
#creation of a new column with a name prediction
Uniswap['Prediction'] = Uniswap[['Close']].shift(-projection_Uniswap)
Uniswap
visualize_Uniswap= cycle(['Open','Close','High','Low','Prediction'])

fig = px.line(Uniswap, x=Uniswap.Date, y=[Uniswap['Open'], Uniswap['Close'],
                                            Uniswap['High'], Uniswap['Low'],␣
 ↪Uniswap['Prediction']],
              labels={'Date': 'Date','value':'Price'})
fig.update_layout(title_text='Bitcoin', font_size=15,␣
 ↪font_color='black',legend_title_text='Parameters')
fig.for_each_trace(lambda t:  t.update(name = next(visualize_Uniswap)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```