**Cryptocurrency Price Prediction with Machine Learning**

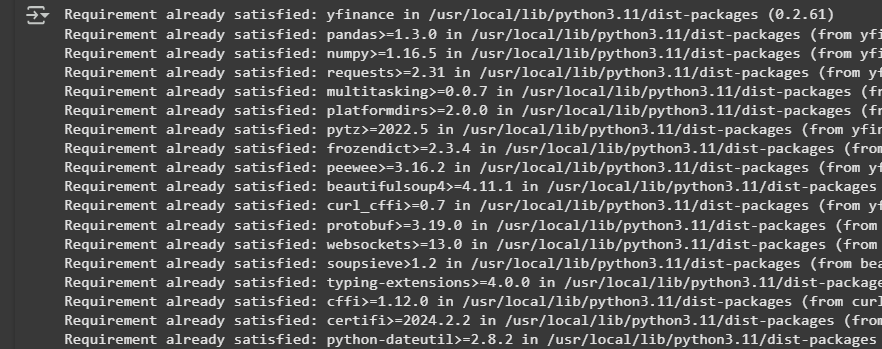
**Code Snippet #1: Code Snippet to Install yfinance Library**

pip install yfinance

**Function:**

**This command installs the yfinance library, which is a Python package used to fetch historical market data from Yahoo Finance. yfinance provides an easy-to-use interface for retrieving stock prices, financial data, and historical data for various financial instruments. By running this command, you ensure that the yfinance package is available in your Python environment, allowing you to use it in your scripts or projects to gather financial data.**

**Figure#1:**

****

**Description:**

**When you run pip install yfinance, the Python package manager (pip) downloads and installs the yfinance library and all of its dependencies. After installation, you can use yfinance to download stock prices, dividends, historical data, and even real-time market data for various companies or financial assets. This tool is widely used for financial analysis and research tasks, such as tracking stock performance over time or building financial models.**

**Code Snippet #2: Code Snippet of Downloading Historical Data for Bitcoin (BTC-USD)**

import pandas as pd

import yfinance as yf

import datetime

from datetime import date, timedelta

today = date.today()

d1 = today.strftime("%Y-%m-%d")

end\_date = d1

d2 = date.today() - timedelta(days=768)

d2 = d2.strftime("%Y-%m-%d")

start\_date = d2

data = yf.download('BTC-USD',

                      start=start\_date,

                      end=end\_date,

                      progress=False)

data["Date"] = data.index

data = data[["Date", "Open", "High", "Low", "Close", "Volume"]]

data.reset\_index(drop=True, inplace=True)

**Function:**

**This code downloads historical price data for Bitcoin (BTC-USD) from Yahoo Finance using the yfinance library. First, it calculates the current date (today) and sets the date range for the data, which spans the last 768 days. The yf.download() function retrieves the Bitcoin data for this period, including the open, high, low, close prices, and trading volume. The retrieved data is stored in a Pandas DataFrame, and a new Date column is added using the index (which corresponds to the date of the record). Finally, the columns are rearranged to include Date, Open, High, Low, Close, and Volume, and the index is reset for easier access.**

**Description:**

**The output will be a Pandas DataFrame containing Bitcoin's historical data for the last 768 days. The DataFrame will include the following columns: Date, Open, High, Low, Close, and Volume. The Date column will represent the date for each data point, and the other columns will show Bitcoin’s price data for that date. This historical data can be used for further analysis, such as studying price trends, performing technical analysis, or building financial models. The reset\_index(drop=True) call ensures the DataFrame’s index is reset, making the data easier to work with.**

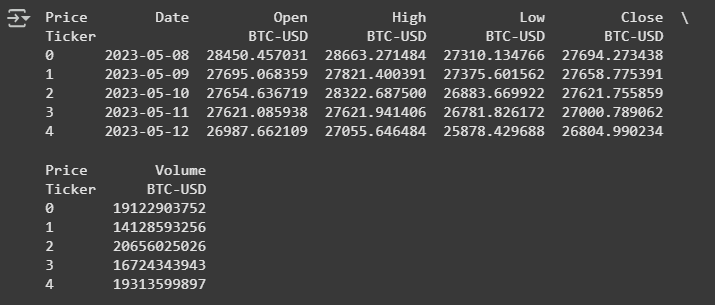
**Code Snippet #2: Code Snippet to Display the First 5 Rows of Bitcoin Data**

print(data.head())

**Function:**

**The print(data.head()) command displays the first five rows of the DataFrame data which contains the historical Bitcoin (BTC-USD) data. The data is retrieved using yfinance, and after processing, it includes columns such as Date, Open, High, Low, Close, and Volume. By printing the first five rows, you can quickly view a snapshot of the data, which helps in inspecting the dataset and verifying its correctness.**

**Figure#2:**

****

**Description:**

**The output will display the first five rows of Bitcoin's historical price data with the following columns: Date, Open, High, Low, Close, and Volume.**

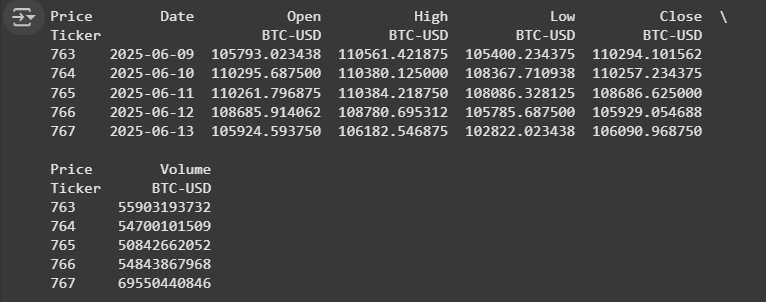
**Code Snippet #3: Code Snippet to Display the Last 5 Rows of Bitcoin Data**

print(data.tail())

**Function:**

**The print(data.tail()) command displays the last five rows of the data DataFrame. This DataFrame contains the historical Bitcoin (BTC-USD) data that has been fetched from Yahoo Finance. The .tail() method returns the last five entries in the dataset, which allows you to check the most recent data points for Bitcoin's price and trading volume.**

**Figure#3:**

****

**Description:**

**The output will show the last five rows of Bitcoin's historical price data, providing insights into the most recent trading days. Each row will include the Date, Open, High, Low, Close, and Volume columns.**

**Code Snippet #1: Code Snippet to Get the Shape of the Data**

data.shape

**Function:**

**The data.shape command is used to get the dimensions of the DataFrame data. It returns a tuple that represents the number of rows and columns in the DataFrame. This is useful for quickly understanding the size of your dataset.**

**Figure#4:**

****

**Description:**

**The output will be a tuple that shows the number of rows and columns in the dataset. For example, if the dataset has 768 rows (one for each day in the last 768 days) and 6 columns (Date, Open, High, Low, Close, and Volume).**

**Code Snippet #5: Code Snippet of Creating a Candlestick Chart for Bitcoin Price Analysis**

import plotly.graph\_objects as go

data.columns = data.columns.get\_level\_values(0) #<- kani i dungag

data["Date"] = pd.to\_datetime(data["Date"])

figure = go.Figure(data=[go.Candlestick(x=data["Date"],

                                        open=data["Open"],

                                        high=data["High"],

                                        low=data["Low"],

                                        close=data["Close"])])

figure.update\_layout(title = "Bitcoin Price Analysis",

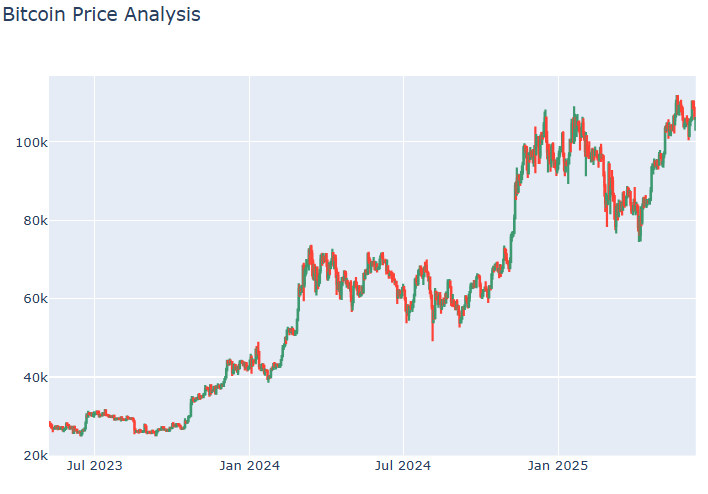
                     xaxis\_rangeslider\_visible=False)

figure.show()

**Function:**

**This code creates a Candlestick chart to visualize Bitcoin's price movement over time using the plotly library. It begins by cleaning the column names in the data DataFrame and converting the Date column to a datetime format for proper handling in the chart. The go.Candlestick() function is used to create the chart, where each candlestick represents the open, high, low, and close prices for a given day. The update\_layout() method customizes the chart’s title and disables the range slider for the x-axis. Finally, figure.show() displays the chart.**

**Figure#5:**

****

**Description:**

**The output will be an interactive Candlestick chart that shows Bitcoin’s price movements over the period defined by your dataset. The chart will display the title "Bitcoin Price Analysis" and will not include a range slider for zooming on the x-axis (date).**

**Code Snippet #6: Code Snippet for Calculating Correlation with Bitcoin Closing Price**

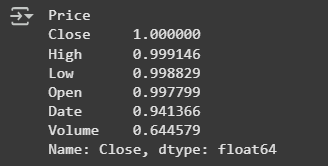
correlation = data.corr()

print(correlation["Close"].sort\_values(ascending=False))

**Function:**

**This code computes the correlation matrix of the data DataFrame, which includes numeric columns such as Open, High, Low, Close, and Volume. The .corr() method calculates the pairwise correlation between each of these columns. The code then isolates the correlation values for the Close column and sorts them in descending order. This helps in identifying which columns have the strongest relationship with Bitcoin’s closing price.**

**Figure#6:**

****

**Description:**

**The output will display the correlation of each numeric column in the dataset with the Close price of Bitcoin. For instance, you might observe that the High, Low, and Open columns have a very high positive correlation with the Close price, as they all reflect Bitcoin's price movements during the same day. On the other hand, the Volume column may show a weak or even negative correlation, indicating that trading volume has little to no direct influence on the closing price of Bitcoin.**

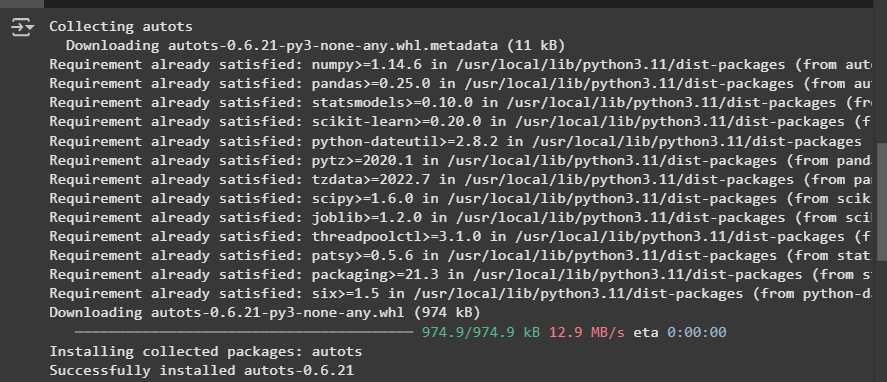
**Code Snippet #7: Code Snippet to Install AutoTS Library**

pip install autots

**Function:**

**This command installs the AutoTS library using the Python package manager pip. AutoTS is a Python library designed for automatic time series forecasting. It provides tools for model selection, hyperparameter tuning, and forecasting, simplifying the process of applying machine learning models to time series data.**

**Figure#7:**

****

**Description:**

**The output will show the installation process of the AutoTS package. If the installation is successful, you'll see messages indicating that the necessary packages have been installed. After installation, you can use AutoTS to easily build and evaluate forecasting models for time series data. For example, it supports automatic feature engineering, model selection, and hyperparameter optimization, making it useful for forecasting tasks in various domains like finance, sales, and demand prediction.**

**Code Snippet #8: Code Snippet for Time Series Forecasting with AutoTS**

from autots import AutoTS

model = AutoTS(forecast\_length=30, frequency='infer', ensemble='simple')

model = model.fit(data, date\_col='Date', value\_col='Close', id\_col=None)

prediction = model.predict()

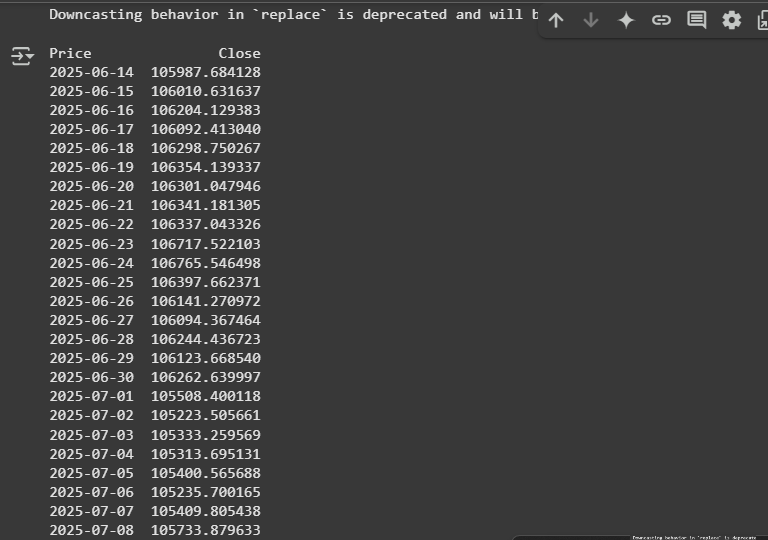
forecast = prediction.forecast

print(forecast)

**Function:**

**This code uses the AutoTS library to perform time series forecasting on Bitcoin's closing price (Close). It first initializes an AutoTS model with a forecast length of 30 days, an inferred frequency for the data (which means it automatically detects the time intervals), and a simple ensemble method for model predictions. The .fit() method trains the model using the data DataFrame, specifying the Date column as the time column and the Close column as the target value. After training, the .predict() method is used to generate the forecast, and the forecast values are stored in prediction.forecast.**

**Figure#8:**

****

**Description:**

**The output will be the predicted Bitcoin closing prices for the next 30 days. The forecast is generated by the AutoTS model, which has learned from historical data (the Close prices over time). The printed forecast will show the estimated closing prices for the next 30 days, based on the patterns identified by the model.**