cleaning

April 25, 2025

1 Bitcoin Price Predictor – AI306

[2]:
```
!pip install kagglehub
```

```
Requirement already satisfied: kagglehub in
/home/mohammed/anaconda3/envs/Crypto/lib/python3.12/site-packages (0.3.4)
Requirement already satisfied: packaging in
/home/mohammed/anaconda3/envs/Crypto/lib/python3.12/site-packages (from
kagglehub) (24.1)
Requirement already satisfied: requests in
/home/mohammed/anaconda3/envs/Crypto/lib/python3.12/site-packages (from
kagglehub) (2.32.3)
Requirement already satisfied: tqdm in
/home/mohammed/anaconda3/envs/Crypto/lib/python3.12/site-packages (from
kagglehub) (4.67.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/home/mohammed/anaconda3/envs/Crypto/lib/python3.12/site-packages (from
requests->kagglehub) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/home/mohammed/anaconda3/envs/Crypto/lib/python3.12/site-packages (from
requests->kagglehub) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/home/mohammed/anaconda3/envs/Crypto/lib/python3.12/site-packages (from
requests->kagglehub) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/home/mohammed/anaconda3/envs/Crypto/lib/python3.12/site-packages (from
requests->kagglehub) (2024.8.30)
```

[3]:
```
import kagglehub
import shutil
import os

import numpy as np
import pandas as pd
```

1.0.1 Downloading the dataset

[4]:
```
# Download latest version
path = kagglehub.dataset_download("mczielinski/bitcoin-historical-data")

print("Path to dataset files:", path)
```

```
Warning: Looks like you're using an outdated `kagglehub` version, please
consider updating (latest version: 0.3.12)
Path to dataset files:
/home/mohammed/.cache/kagglehub/datasets/mczielinski/bitcoin-historical-
data/versions/216
```

1.0.2 Copying the dataset to the project directory

[ ]:
```
# Source and destination paths
source_path = os.path.join(path, 'btcusd_1-min_data.csv')
destination_path = './data/btcusd_dataset.csv'

# Create destination directory if it doesn't exist
os.makedirs('./data', exist_ok=True)

# Only copy if the file doesn't already exist
if not os.path.exists(destination_path):
    shutil.copy(source_path, destination_path)
    print("File copied successfully.")

else:
    print("File already exists. Skipping copy.")
```

```
File copied successfully.
```

1.0.3 Reading dataset

[6]:
```
data = pd.read_csv(destination_path)
```

```
/tmp/ipykernel_8164/1927658998.py:1: DtypeWarning: Columns (6) have mixed types.
Specify dtype option on import or set low_memory=False.
  data = pd.read_csv(destination_path)
```

[8]:
```
data.head()
```

[8]:
```
      Timestamp  Open  High   Low  Close  Volume                   datetime
0  1.325412e+09  4.58  4.58  4.58   4.58     0.0  2012-01-01 10:01:00+00:00
1  1.325412e+09  4.58  4.58  4.58   4.58     0.0  2012-01-01 10:02:00+00:00
2  1.325412e+09  4.58  4.58  4.58   4.58     0.0  2012-01-01 10:03:00+00:00
3  1.325412e+09  4.58  4.58  4.58   4.58     0.0  2012-01-01 10:04:00+00:00
4  1.325412e+09  4.58  4.58  4.58   4.58     0.0  2012-01-01 10:05:00+00:00
```

1.0.4 Info about dataset

[10]:
```
print('Shape of the dataset: ', data.shape)
```

```
Shape of the dataset:  (7001004, 7)
```

[11]:
```
print(f"info of the dataset: \n{data.info()}\n")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7001004 entries, 0 to 7001003
Data columns (total 7 columns):
 #   Column     Dtype
---  ------     -----
 0   Timestamp  float64
 1   Open       float64
 2   High       float64
 3   Low        float64
 4   Close      float64
 5   Volume     float64
 6   datetime   object
dtypes: float64(6), object(1)
memory usage: 373.9+ MB
info of the dataset:
None
```

[12]:
```
print(f"describe of the dataset: \n{data.describe()}\n")
```

```
describe of the dataset:
          Timestamp          Open          High           Low         Close  \
count  7.001004e+06  7.001004e+06  7.001004e+06  7.001004e+06  7.001004e+06
mean   1.535443e+09  1.729476e+04  1.730170e+04  1.728760e+04  1.729476e+04
std    1.212619e+08  2.389940e+04  2.390744e+04  2.389117e+04  2.389938e+04
min    1.325412e+09  3.800000e+00  3.800000e+00  3.800000e+00  3.800000e+00
25%    1.430427e+09  4.239100e+02  4.240000e+02  4.237600e+02  4.239300e+02
50%    1.535442e+09  6.575210e+03  6.578515e+03  6.572320e+03  6.575290e+03
75%    1.640457e+09  2.720000e+04  2.720400e+04  2.719600e+04  2.720000e+04
max    1.745542e+09  1.091110e+05  1.093560e+05  1.087940e+05  1.090360e+05

             Volume
count  7.001004e+06
mean   5.308327e+00
std    2.253495e+01
min    0.000000e+00
25%    1.815710e-02
50%    4.703309e-01
75%    3.039586e+00
max    5.853852e+03
```

[13]:
```
print(f"null values of the dataset: \n{data.isnull().sum()}\n")
```

```
null values of the dataset:
Timestamp         0
Open              0
High              0
Low               0
Close             0
Volume            0
datetime     218724
dtype: int64
```

[14]:
```
print(f"duplicated values of the dataset: \n{data.duplicated().sum()}\n")
```

```
duplicated values of the dataset:
0
```

1.0.5 Cleaning Dataset

[19]:
```
# Drop the 'datetime' column
data = data.drop(columns=['datetime'])

# view the updated DataFrame
print(data.head())
print(f"\n\nnull values of the dataset: \n{data.isnull().sum()}\n")
```

```
      Timestamp  Open  High   Low  Close  Volume
0  1.325412e+09  4.58  4.58  4.58   4.58     0.0
1  1.325412e+09  4.58  4.58  4.58   4.58     0.0
2  1.325412e+09  4.58  4.58  4.58   4.58     0.0
3  1.325412e+09  4.58  4.58  4.58   4.58     0.0
4  1.325412e+09  4.58  4.58  4.58   4.58     0.0


null values of the dataset:
Timestamp    0
Open         0
High         0
Low          0
Close        0
Volume       0
dtype: int64
```

```
[21]: # Convert Unix timestamp (seconds since 00:00:00 UTC January 1, 1970) to␣
       ↪datetime
      data['datetime'] = pd.to_datetime(data['Timestamp'], unit='s')

      print(data.head())
```

```
      Timestamp  Open  High   Low  Close  Volume            datetime
0  1.325412e+09  4.58  4.58  4.58   4.58     0.0 2012-01-01 10:01:00
1  1.325412e+09  4.58  4.58  4.58   4.58     0.0 2012-01-01 10:02:00
2  1.325412e+09  4.58  4.58  4.58   4.58     0.0 2012-01-01 10:03:00
3  1.325412e+09  4.58  4.58  4.58   4.58     0.0 2012-01-01 10:04:00
4  1.325412e+09  4.58  4.58  4.58   4.58     0.0 2012-01-01 10:05:00
```

```
[22]: # ensure the data is continuous and their are no missing values or rows,
      # Reindexes the data to have a row for every minute - even if that minute was␣
       ↪missing in the original data.
      continuous_data = data.set_index('datetime').asfreq('min')
      print('data Null/NA Values before fill:', continuous_data.isnull().values.sum())

      # fill in and interpolate missing values after re-indexing is done
      continuous_data.interpolate(method='time', inplace=True)  # Time-based␣
       ↪interpolation
      continuous_data.ffill(inplace=True) # forwards fill missing values

      continuous_data.reset_index(inplace=True) # Moves 'datetime' back from the␣
       ↪index to a regular column
      print('data Null/NA Values after fill:', continuous_data.isnull().values.sum())
```

```
data Null/NA Values before fill: 6960
data Null/NA Values after fill: 0
```

```
[25]: first_nonzero_row = data[data['Volume'] > 0].head(1)
      print(first_nonzero_row)
```

```
        Timestamp  Open  High   Low  Close  Volume            datetime
627  1.325450e+09  4.84  4.84  4.84   4.84    10.0 2012-01-01 20:28:00
```