LlaMa

January 12, 2025

[1]:
```
# Install necessary libraries
!pip install transformers yfinance ccxt pandas matplotlib torch
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-
packages (4.47.1)
Requirement already satisfied: yfinance in /usr/local/lib/python3.10/dist-
packages (0.2.51)
Collecting ccxt
  Downloading ccxt-4.4.47-py2.py3-none-any.whl.metadata (117 kB)
                           117.7/117.7

kB 5.5 MB/s eta 0:00:00
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-
packages (2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (3.10.0)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages
(2.5.1+cu121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from transformers) (3.16.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.27.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.21.0)
Requirement already satisfied: safetensors>=0.4.1 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.5.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-
packages (from transformers) (4.67.1)
```

```
Requirement already satisfied: multitasking>=0.0.7 in
/usr/local/lib/python3.10/dist-packages (from yfinance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in /usr/local/lib/python3.10/dist-
packages (from yfinance) (5.3.0)
Requirement already satisfied: platformdirs>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from yfinance) (4.3.6)
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.10/dist-
packages (from yfinance) (2024.2)
Requirement already satisfied: frozendict>=2.3.4 in
/usr/local/lib/python3.10/dist-packages (from yfinance) (2.4.6)
Requirement already satisfied: peewee>=3.16.2 in /usr/local/lib/python3.10/dist-
packages (from yfinance) (3.17.8)
Requirement already satisfied: beautifulsoup4>=4.11.1 in
/usr/local/lib/python3.10/dist-packages (from yfinance) (4.12.3)
Requirement already satisfied: html5lib>=1.1 in /usr/local/lib/python3.10/dist-
packages (from yfinance) (1.1)
Requirement already satisfied: setuptools>=60.9.0 in
/usr/local/lib/python3.10/dist-packages (from ccxt) (75.1.0)
Requirement already satisfied: certifi>=2018.1.18 in
/usr/local/lib/python3.10/dist-packages (from ccxt) (2024.12.14)
Requirement already satisfied: cryptography>=2.6.1 in
/usr/local/lib/python3.10/dist-packages (from ccxt) (43.0.3)
Requirement already satisfied: typing-extensions>=4.4.0 in
/usr/local/lib/python3.10/dist-packages (from ccxt) (4.12.2)
Collecting aiohttp<=3.10.11 (from ccxt)
  Downloading aiohttp-3.10.11-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (7.7 kB)
Collecting aiodns>=1.1.1 (from ccxt)
  Downloading aiodns-3.2.0-py3-none-any.whl.metadata (4.0 kB)
Requirement already satisfied: yarl>=1.7.2 in /usr/local/lib/python3.10/dist-
packages (from ccxt) (1.18.3)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.2)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
```

```
packages (from torch) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch) (3.1.5)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch) (2024.10.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-
packages (from torch) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from sympy==1.13.1->torch) (1.3.0)
Collecting pycares>=4.0.0 (from aiodns>=1.1.1->ccxt)
  Downloading pycares-4.5.0-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.1 kB)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<=3.10.11->ccxt) (2.4.4)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<=3.10.11->ccxt) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp<=3.10.11->ccxt) (24.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<=3.10.11->ccxt) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<=3.10.11->ccxt) (6.1.0)
Requirement already satisfied: async-timeout<6.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<=3.10.11->ccxt) (4.0.3)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-
packages (from beautifulsoup4>=4.11.1->yfinance) (2.6)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.10/dist-
packages (from cryptography>=2.6.1->ccxt) (1.17.1)
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.10/dist-
packages (from html5lib>=1.1->yfinance) (1.17.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-
packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.3.0)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from yarl>=1.7.2->ccxt) (0.2.1)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch) (3.0.2)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-
packages (from cffi>=1.12->cryptography>=2.6.1->ccxt) (2.22)
Downloading ccxt-4.4.47-py2.py3-none-any.whl (5.6 MB)
                         5.6/5.6 MB
81.4 MB/s eta 0:00:00
Downloading aiodns-3.2.0-py3-none-any.whl (5.7 kB)
```

```
Downloading
aiohttp-3.10.11-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2
MB)
                         1.2/1.2 MB
72.8 MB/s eta 0:00:00
Downloading
pycares-4.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (288
kB)
                         288.6/288.6 kB
26.3 MB/s eta 0:00:00
Installing collected packages: pycares, aiohttp, aiodns, ccxt
  Attempting uninstall: aiohttp
    Found existing installation: aiohttp 3.11.11
    Uninstalling aiohttp-3.11.11:
      Successfully uninstalled aiohttp-3.11.11
Successfully installed aiodns-3.2.0 aiohttp-3.10.11 ccxt-4.4.47 pycares-4.5.0
```

[2]:
```python
# Import libraries
import yfinance as yf
import ccxt
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from transformers import LlamaTokenizer, LlamaForCausalLM
import torch
```

[3]:
```python
# Function to fetch stock data using yfinance
def fetch_stock_data(ticker, start_date='2010-01-01', end_date=None):
    stock = yf.Ticker(ticker)
    df = stock.history(start=start_date, end=end_date)
    df = df[['Open', 'High', 'Low', 'Close', 'Volume']].reset_index()
    df.columns = ['timestamp', 'open', 'high', 'low', 'close', 'volume']
    return df

# Function to fetch cryptocurrency data using ccxt
def fetch_crypto_data(symbol='BTC/USD', exchange_name='kraken', timeframe='1d',␣
 ↪limit=1000):
    exchange_class = getattr(ccxt, exchange_name)
    exchange = exchange_class()
    ohlcv = exchange.fetch_ohlcv(symbol, timeframe=timeframe, limit=limit)
    data = pd.DataFrame(ohlcv, columns=['timestamp', 'open', 'high', 'low',␣
 ↪'close', 'volume'])
    data['timestamp'] = pd.to_datetime(data['timestamp'], unit='ms')
    return data

# Fetch data for stocks and cryptocurrencies
tesla_data = fetch_stock_data('TSLA', start_date='2020-01-01')
```

```python
apple_data = fetch_stock_data('AAPL', start_date='2020-01-01')
btc_data = fetch_crypto_data('BTC/USD')
eth_data = fetch_crypto_data('ETH/USD')

# Display sample data
print("Tesla Data:\n", tesla_data.head())
print("Apple Data:\n", apple_data.head())
print("Bitcoin Data:\n", btc_data.head())
print("Ethereum Data:\n", eth_data.head())
```

```
Tesla Data:
                    timestamp       open       high        low      close  \
0 2020-01-02 00:00:00-05:00  28.299999  28.713333  28.114000  28.684000
1 2020-01-03 00:00:00-05:00  29.366667  30.266666  29.128000  29.534000
2 2020-01-06 00:00:00-05:00  29.364668  30.104000  29.333332  30.102667
3 2020-01-07 00:00:00-05:00  30.760000  31.441999  30.224001  31.270666
4 2020-01-08 00:00:00-05:00  31.580000  33.232666  31.215334  32.809334

      volume
0  142981500
1  266677500
2  151995000
3  268231500
4  467164500
Apple Data:
                    timestamp       open       high        low      close  \
0 2020-01-02 00:00:00-05:00  71.799866  72.856606  71.545380  72.796013
1 2020-01-03 00:00:00-05:00  72.020454  72.851784  71.862915  72.088318
2 2020-01-06 00:00:00-05:00  71.206062  72.701485  70.953995  72.662704
3 2020-01-07 00:00:00-05:00  72.672409  72.929322  72.100418  72.320976
4 2020-01-08 00:00:00-05:00  72.022843  73.787300  72.022843  73.484337

      volume
0  135480400
1  146322800
2  118387200
3  108872000
4  132079200
Bitcoin Data:
    timestamp     open     high      low    close       volume
0 2023-01-24  22926.0  23158.7  22455.9  22633.8  3077.643596
1 2023-01-25  22636.0  23829.3  22320.0  23056.5  5020.204657
2 2023-01-26  23063.2  23293.3  22857.5  23010.6  3753.163921
3 2023-01-27  23010.6  23500.0  22492.8  23077.5  3420.533974
4 2023-01-28  23079.9  23191.9  22900.0  23031.1  1006.279351
Ethereum Data:
    timestamp     open     high      low    close        volume
```

```
0 2023-01-24  1626.40  1640.04  1531.35  1556.01  40939.533459
1 2023-01-25  1555.63  1639.60  1518.00  1611.62  33939.635980
2 2023-01-26  1611.62  1633.39  1578.78  1602.22  20970.688727
3 2023-01-27  1602.14  1621.59  1551.00  1597.31  38471.693576
4 2023-01-28  1597.31  1606.82  1557.24  1572.63  17893.514388
```

[4]:
```
# Function to create a text prompt for LLaMA
def create_text_prompt(data, asset_name="Asset", max_entries=5):
    prompt = f"Here is the historical price data for {asset_name}. Predict the␣
 ↪next closing price:\n\n"
    data = data.tail(max_entries)
    for _, row in data.iterrows():
        prompt += f"Date: {row['timestamp'].strftime('%Y-%m-%d')}, Open:␣
 ↪{row['open']}, High: {row['high']}, Low: {row['low']}, Close:␣
 ↪{row['close']}\n"
    prompt += "\nThe next closing price is: "
    return prompt

# Generate prompts for all four assets
tesla_prompt = create_text_prompt(tesla_data, asset_name="Tesla", max_entries=5)
apple_prompt = create_text_prompt(apple_data, asset_name="Apple", max_entries=5)
btc_prompt = create_text_prompt(btc_data, asset_name="Bitcoin", max_entries=5)
eth_prompt = create_text_prompt(eth_data, asset_name="Ethereum", max_entries=5)

# Display prompts
print("Tesla Prompt:\n", tesla_prompt)
print("\nApple Prompt:\n", apple_prompt)
print("\nBitcoin Prompt:\n", btc_prompt)
print("\nEthereum Prompt:\n", eth_prompt)
```

```
Tesla Prompt:
 Here is the historical price data for Tesla. Predict the next closing price:

Date: 2025-01-03, Open: 381.4800109863281, High: 411.8800048828125, Low:
379.45001220703125, Close: 410.44000244140625
Date: 2025-01-06, Open: 423.20001220703125, High: 426.42999267578125, Low:
401.70001220703125, Close: 411.04998779296875
Date: 2025-01-07, Open: 405.8299865722656, High: 414.3299865722656, Low: 390.0,
Close: 394.3599853515625
Date: 2025-01-08, Open: 392.95001220703125, High: 402.5, Low: 387.3999938964844,
Close: 394.94000244140625
Date: 2025-01-10, Open: 391.3999938964844, High: 399.2799987792969, Low:
377.2900085449219, Close: 394.739990234375

The next closing price is:

Apple Prompt:
 Here is the historical price data for Apple. Predict the next closing price:
```

```
Date: 2025-01-03, Open: 243.36000061035156, High: 244.17999267578125, Low:
241.88999938964844, Close: 243.36000061035156
Date: 2025-01-06, Open: 244.30999755859375, High: 247.3300018310547, Low:
243.1999969482422, Close: 245.0
Date: 2025-01-07, Open: 242.97999572753906, High: 245.5500030517578, Low:
241.35000610351562, Close: 242.2100067138672
Date: 2025-01-08, Open: 241.9199981689453, High: 243.7100067138672, Low:
240.0500030517578, Close: 242.6999969482422
Date: 2025-01-10, Open: 240.00999450683594, High: 240.16000366210938, Low:
233.0, Close: 236.85000610351562

The next closing price is:

Bitcoin Prompt:
 Here is the historical price data for Bitcoin. Predict the next closing price:

Date: 2025-01-08, Open: 96932.2, High: 97232.8, Low: 92501.0, Close: 95060.4
Date: 2025-01-09, Open: 95060.1, High: 95251.1, Low: 91168.5, Close: 92531.7
Date: 2025-01-10, Open: 92523.5, High: 95771.2, Low: 92200.0, Close: 94698.8
Date: 2025-01-11, Open: 94698.9, High: 94986.6, Low: 93865.0, Close: 94562.1
Date: 2025-01-12, Open: 94562.1, High: 95300.0, Low: 93652.2, Close: 93875.9

The next closing price is:

Ethereum Prompt:
 Here is the historical price data for Ethereum. Predict the next closing price:

Date: 2025-01-08, Open: 3380.67, High: 3414.55, Low: 3210.73, Close: 3326.34
Date: 2025-01-09, Open: 3326.4, High: 3355.0, Low: 3159.16, Close: 3219.04
Date: 2025-01-10, Open: 3218.8, High: 3320.0, Low: 3194.91, Close: 3266.05
Date: 2025-01-11, Open: 3266.06, High: 3317.8, Low: 3220.26, Close: 3282.32
Date: 2025-01-12, Open: 3281.68, High: 3295.83, Low: 3225.29, Close: 3234.97

The next closing price is:
```

[5]:
```python
# Load LLaMA tokenizer and model
model_name = "huggyllama/llama-7b"  # Adjust based on available model
tokenizer = LlamaTokenizer.from_pretrained(model_name)
model = LlamaForCausalLM.from_pretrained(model_name, torch_dtype=torch.float16,␣
 ↪device_map="auto")

print("LLaMA model loaded successfully.")
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94:
UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
```

```
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab
and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
public models or datasets.
  warnings.warn(

tokenizer_config.json:   0%|          | 0.00/2.28k [00:00<?, ?B/s]

tokenizer.model:   0%|          | 0.00/500k [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/411 [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/1.84M [00:00<?, ?B/s]

You are using the default legacy behaviour of the <class
'transformers.models.llama.tokenization_llama.LlamaTokenizer'>. This is
expected, and simply means that the `legacy` (previous) behavior will be used so
nothing changes for you. If you want to use the new behaviour, set
`legacy=False`. This should only be set if you understand what it means, and
thoroughly read the reason why this was added as explained in
https://github.com/huggingface/transformers/pull/24565 - if you loaded a llama
tokenizer from a GGUF file you can ignore this message

config.json:   0%|          | 0.00/594 [00:00<?, ?B/s]

model.safetensors.index.json:   0%|          | 0.00/26.8k [00:00<?, ?B/s]

Downloading shards:   0%|          | 0/2 [00:00<?, ?it/s]

model-00001-of-00002.safetensors:   0%|          | 0.00/9.98G [00:00<?, ?B/s]

model-00002-of-00002.safetensors:   0%|          | 0.00/3.50G [00:00<?, ?B/s]

Loading checkpoint shards:   0%|          | 0/2 [00:00<?, ?it/s]

generation_config.json:   0%|          | 0.00/137 [00:00<?, ?B/s]

LLaMA model loaded successfully.
```

[6]:
```python
# Function to generate predictions using LLaMA
def predict_next_price(prompt):
    inputs = tokenizer(prompt, return_tensors="pt").to('cuda')  # Move input to␣
 ↪GPU
    with torch.no_grad():
        outputs = model.generate(
            inputs.input_ids,
            max_new_tokens=10,
            pad_token_id=tokenizer.eos_token_id
        )
    prediction = tokenizer.decode(outputs[:, inputs.input_ids.shape[-1]:][0],␣
 ↪skip_special_tokens=True)
    return prediction
```

```python
# Predict prices for all four assets
predicted_tesla_price = predict_next_price(tesla_prompt)
predicted_apple_price = predict_next_price(apple_prompt)
predicted_btc_price = predict_next_price(btc_prompt)
predicted_eth_price = predict_next_price(eth_prompt)

# Display predictions
print("Predicted Tesla Price:", predicted_tesla_price)
print("Predicted Apple Price:", predicted_apple_price)
print("Predicted Bitcoin Price:", predicted_btc_price)
print("Predicted Ethereum Price:", predicted_eth_price)
```

```
The attention mask is not set and cannot be inferred from input because pad
token is same as eos token. As a consequence, you may observe unexpected
behavior. Please pass your input's `attention_mask` to obtain reliable results.
```

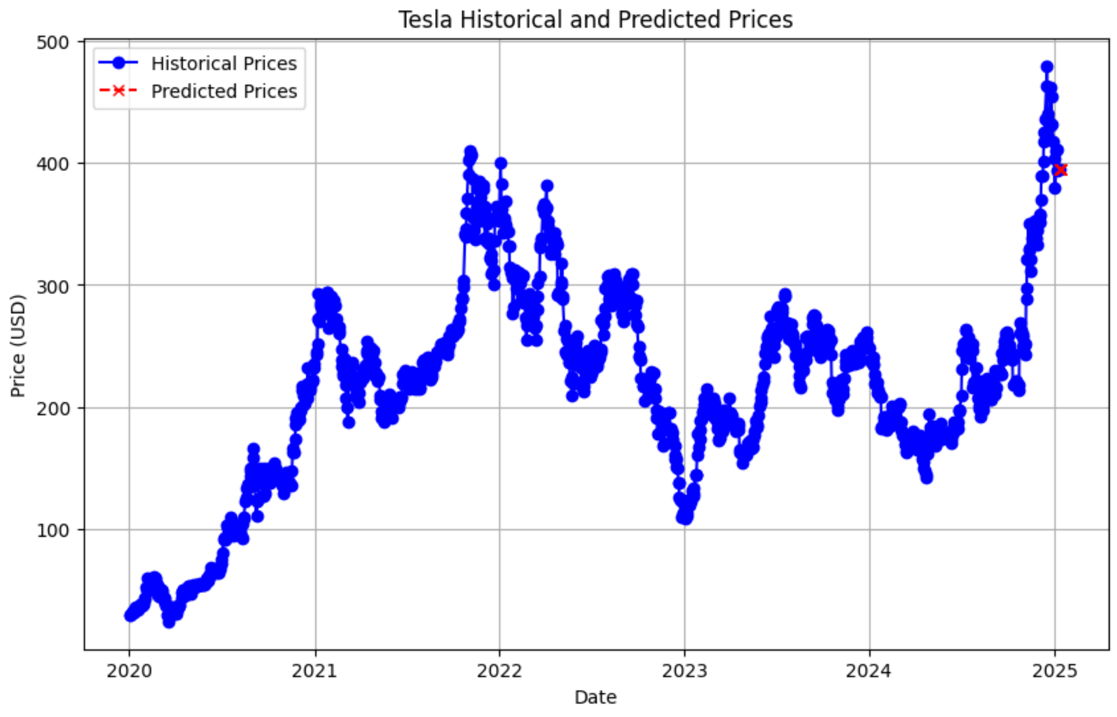
```
Predicted Tesla Price: 394.739990
Predicted Apple Price: 236.850006
Predicted Bitcoin Price: 93875.9

Answer
Predicted Ethereum Price: 3234.97

Answer
```

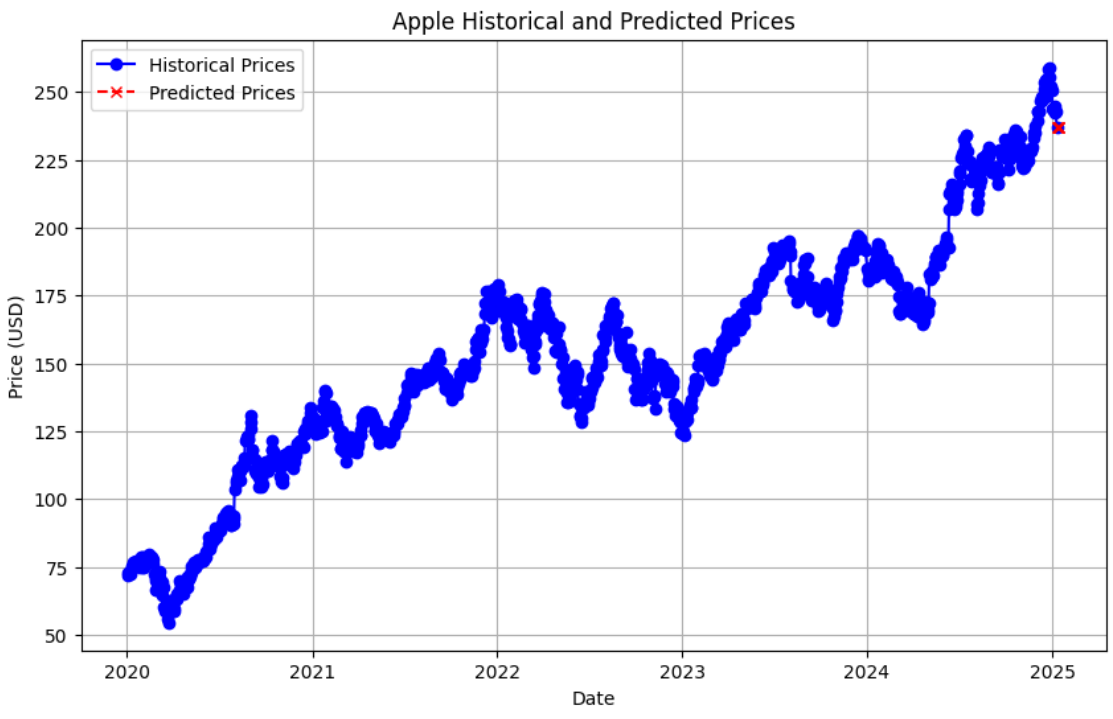
[7]:
```python
# Function to generate predictions for multiple days
def generate_predictions_for_days(data, num_days=5, asset_name="Asset"):
    future_data = data.copy()
    predictions = []

    for i in range(num_days):
        prompt = create_text_prompt(future_data, asset_name=asset_name,␣
 ↪max_entries=5)
        predicted_price = predict_next_price(prompt)
        predicted_price_value = float(predicted_price.split()[0])
        predictions.append(predicted_price_value)

        # Add the predicted price to the dataset for the next iteration
        next_date = future_data['timestamp'].iloc[-1] + pd.DateOffset(1)
        new_row = pd.DataFrame({
            'timestamp': [next_date],
            'open': [np.nan],
            'high': [np.nan],
            'low': [np.nan],
            'close': [predicted_price_value],
            'volume': [np.nan]
        })
        future_data = pd.concat([future_data, new_row], ignore_index=True)

    return predictions

# Generate predictions for all four assets
tesla_predictions = generate_predictions_for_days(tesla_data, num_days=5,␣
 ↪asset_name="Tesla")
apple_predictions = generate_predictions_for_days(apple_data, num_days=5,␣
 ↪asset_name="Apple")
btc_predictions = generate_predictions_for_days(btc_data, num_days=5,␣
 ↪asset_name="Bitcoin")
eth_predictions = generate_predictions_for_days(eth_data, num_days=5,␣
 ↪asset_name="Ethereum")

# Display predictions
print("Predicted Tesla Prices for Next 5 Days:", tesla_predictions)
print("Predicted Apple Prices for Next 5 Days:", apple_predictions)
print("Predicted Bitcoin Prices for Next 5 Days:", btc_predictions)
print("Predicted Ethereum Prices for Next 5 Days:", eth_predictions)
```

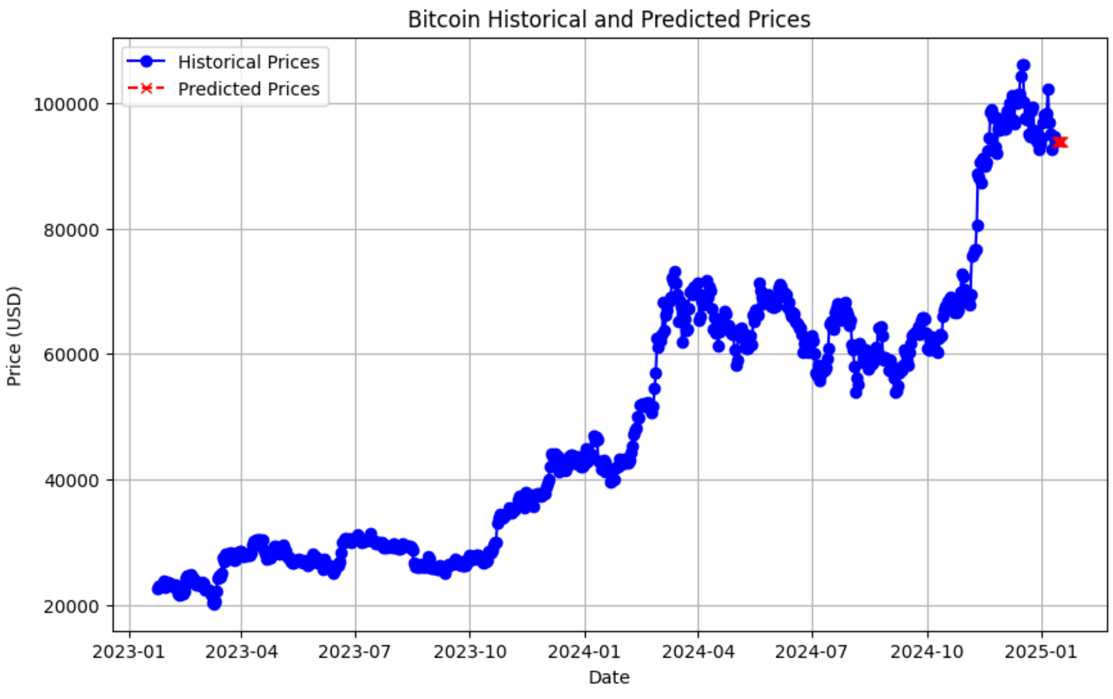
```
Predicted Tesla Prices for Next 5 Days: [394.73999, 394.73999, 394.73999,
394.73999, 394.73999]
Predicted Apple Prices for Next 5 Days: [236.850006, 236.850006, 236.850006,
236.850006, 236.850006]
Predicted Bitcoin Prices for Next 5 Days: [93875.9, 93875.9, 93875.9, 93875.9,
93875.9]
Predicted Ethereum Prices for Next 5 Days: [3234.97, 3234.97, 3234.97, 3234.97,
3234.97]
```

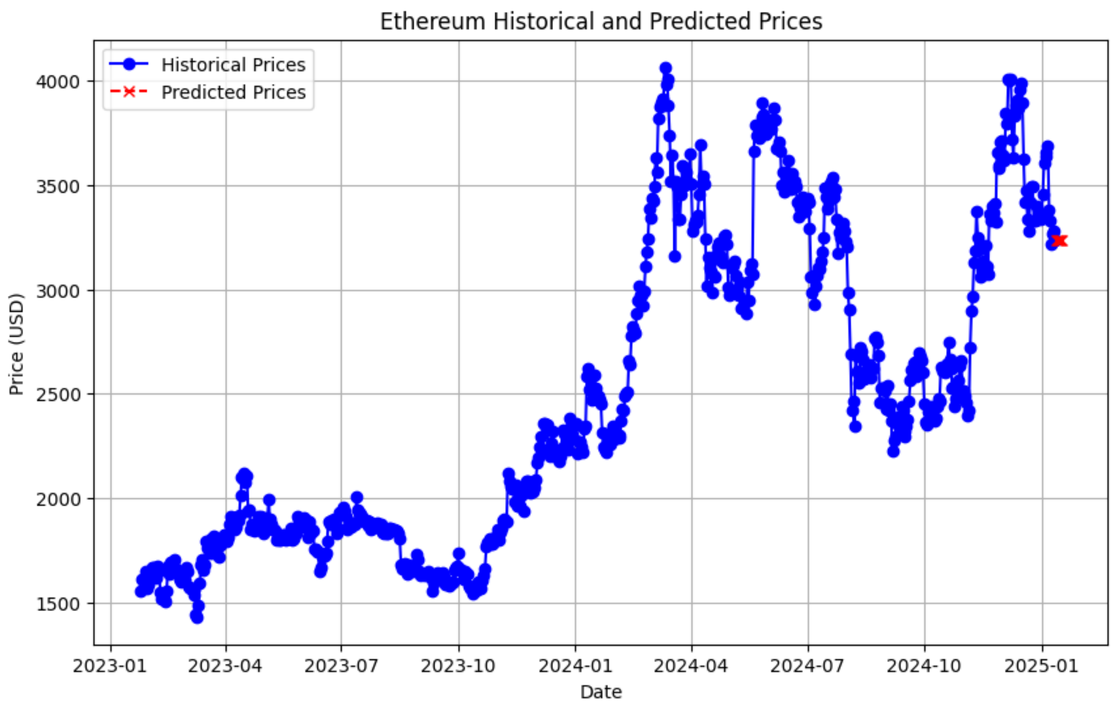
[8]:
```
# Function to plot predictions along with historical data
def plot_predictions(data, predicted_prices, asset_name="Asset", num_days=5):
    data['timestamp'] = pd.to_datetime(data['timestamp'])
    plt.figure(figsize=(10, 6))

    # Plot historical closing prices
    plt.plot(data['timestamp'], data['close'], label='Historical Prices',␣
 ↪marker='o', color='blue')

    # Generate future dates
    last_date = data['timestamp'].iloc[-1]
    future_dates = [last_date + pd.DateOffset(days=i+1) for i in␣
 ↪range(num_days)]

    # Plot predicted prices
    plt.plot(future_dates, predicted_prices, label='Predicted Prices',␣
 ↪marker='x', color='red', linestyle='--')
```

```
    plt.title(f"{asset_name} Historical and Predicted Prices")
    plt.xlabel("Date")
    plt.ylabel("Price (USD)")
    plt.legend()
    plt.grid(True)
    plt.show()

# Plot predictions for all four assets
plot_predictions(tesla_data, tesla_predictions, asset_name="Tesla", num_days=5)
plot_predictions(apple_data, apple_predictions, asset_name="Apple", num_days=5)
plot_predictions(btc_data, btc_predictions, asset_name="Bitcoin", num_days=5)
plot_predictions(eth_data, eth_predictions, asset_name="Ethereum", num_days=5)
```




Apple Historical and Predicted Prices
Historical Prices
Predicted Prices
Price (USD)
250
225
200
175
150
125
100
75
50
2020
2021
2022
2023
2024
2025
Date


Bitcoin Historical and Predicted Prices
Historical Prices
Predicted Prices
Price (USD)
100000
80000
60000
40000
20000
2023-01
2023-04
2023-07
2023-10
2024-01
2024-04
2024-07
2024-10
2025-01
Date



[9]:
```
# Function to compute evaluation metrics
def compute_metrics(actual, predicted):
    mse = np.mean((np.array(actual) - np.array(predicted)) ** 2)
    rmse = np.sqrt(mse)
    mae = np.mean(np.abs(np.array(actual) - np.array(predicted)))
    return mse, rmse, mae
```

[10]:
```
# Predict prices for Tesla
predicted_tesla_price = predict_next_price(tesla_prompt)

# Create a list of identical predictions (as per the original behavior)
predicted_tesla_prices = [float(predicted_tesla_price.split()[0])] * 5

# Compute evaluation metrics for Tesla
actual_tesla_prices = tesla_data['close'].tail(5).tolist()
tesla_mse, tesla_rmse, tesla_mae = compute_metrics(actual_tesla_prices,␣
 ↪predicted_tesla_prices)

# Display Tesla metrics
print(f"Tesla - MSE: {tesla_mse}, RMSE: {tesla_rmse}, MAE: {tesla_mae}")
```

```
Tesla - MSE: 102.53816543530188, RMSE: 10.126113046737226, MAE:
6.518005511718764
```

[11]:
```python
# Predict prices for Apple
predicted_apple_price = predict_next_price(apple_prompt)

# Create a list of identical predictions
predicted_apple_prices = [float(predicted_apple_price.split()[0])] * 5

# Compute evaluation metrics for Apple
actual_apple_prices = apple_data['close'].tail(5).tolist()
apple_mse, apple_rmse, apple_mae = compute_metrics(actual_apple_prices,␣
 ↪predicted_apple_prices)

# Display Apple metrics
print(f"Apple - MSE: {apple_mse}, RMSE: {apple_rmse}, MAE: {apple_mae}")
```

```
Apple - MSE: 34.35088675480286, RMSE: 5.860962954566669, MAE: 5.173996075195305
```

[12]:
```python
# Predict prices for Bitcoin
predicted_btc_price = predict_next_price(btc_prompt)

# Create a list of identical predictions
predicted_btc_prices = [float(predicted_btc_price.split()[0])] * 5

# Compute evaluation metrics for Bitcoin
actual_btc_prices = btc_data['close'].tail(5).tolist()
btc_mse, btc_rmse, btc_mae = compute_metrics(actual_btc_prices,␣
 ↪predicted_btc_prices)

# Display Bitcoin metrics
print(f"Bitcoin - MSE: {btc_mse}, RMSE: {btc_rmse}, MAE: {btc_mae}")
```

```
Bitcoin - MSE: 871589.7480000046, RMSE: 933.5897107402184, MAE:
807.5600000000035
```

[13]:
```python
# Predict prices for Ethereum
predicted_eth_price = predict_next_price(eth_prompt)

# Create a list of identical predictions
predicted_eth_prices = [float(predicted_eth_price.split()[0])] * 5

# Compute evaluation metrics for Ethereum
actual_eth_prices = eth_data['close'].tail(5).tolist()
eth_mse, eth_rmse, eth_mae = compute_metrics(actual_eth_prices,␣
 ↪predicted_eth_prices)

# Display Ethereum metrics
print(f"Ethereum - MSE: {eth_mse}, RMSE: {eth_rmse}, MAE: {eth_mae}")
```

```
Ethereum - MSE: 2362.046140000023, RMSE: 48.60088620591216, MAE:
```

```
37.146000000000186
```

[ ]:

```
```