Zelta Labs, IIT Kharagpur Data Science Hackathon- 2024 Final Report

By - Team Data.py

1.Introduction

Embarking on our journey in the dynamic realm of BTC/USDT algorithmic trading, we confront the distinctive challenges inherent in this cryptocurrency market. Thriving in its fast-paced landscape, every moment unfolds as a potential opportunity or risk. As active participants, we immerse ourselves in the intricacies of the digital domain, crafting resilient trading techniques that not only weather market fluctuations but also surpass established benchmarks.

Our collective efforts yield a fusion of ingenuity and strategic thinking, culminating in algorithms meticulously designed to navigate the nuances of the BTC/USDT market. Beyond the pursuit of profits, our focal point is the essential facet of risk management, particularly vital in the unpredictable landscape of cryptocurrency trading. Integrated with sentiment analysis, we harness the power of market sentiment to inform our strategies, enhancing our risk management protocols.

In an ever-evolving landscape that demands adaptability and acute market awareness, this report transcends a mere summary of our work. It stands as a chapter in an ongoing narrative of innovation and expertise in BTC/USDT algorithmic trading. Our intent is to spark novel ideas, foster collaborative endeavors, and advocate for methodologies where algorithms play a pivotal role in maximizing returns and controlling risks.

2.Proposed system

In envisioning an advanced and comprehensive system for BTC/USDT algorithmic trading, we propose the integration of our LSTM-based predictive model with robust backtesting mechanisms and sentiment analysis. This amalgamation aims to create a holistic approach, enhancing the precision and adaptability of our trading strategies.

Our envisioned system leverages the cutting-edge capabilities of Long Short-Term Memory (LSTM) networks to redefine BTC/USDT market forecasting. Operating as a specialized type of recurrent neural network (RNN), the LSTM model showcases exceptional prowess in capturing intricate temporal dependencies within market data. Its unique ability to retain and utilize information over extended sequences positions the model as a frontrunner in delivering accurate predictions, offering an insightful perspective into the dynamic BTC/USDT landscape.

3.System Requirements

The implementation of the proposed system for BCT/USDT market requires the following system requirements:

3.1.Hardware requirements:

- A computer or server with sufficient computational power and the memory to handle the training and evaluation of deep learning models.
- Adequate storage capacity to store the dataset and trained models.
- A stable internet connection for data retrieval and updates.

3.2.Software requirements:

- Python programming language(version 3.x) for model implementation and evaluation.
- Appropriate Python libraries and frameworks,such as TensorFlow, PyTorch,Keras , and scikit-learn, for deep learning model development and analysis
- Integrated Development Environment (IDE) or code editor such as Google Colab, PyCharm,Jupyter Notebook, or Visual Studio Code, for coding and model development.
- Required Python packages and dependencies, including pandas, NumPy, Matploylib and seaborn for data preprocessing , visualization and analysis.
- Data retrieval library, such as yahoo finance , for obtaining BCT/USDT market data.

4.Data Acquisition:

- The yahoo finance dataset or similar BCT/USDT market dataset , containing historical price and volume data for the specific crypto being analyzed can be used.
- Sufficient historical data to train and evaluate the models effectively is a requirement , here spanning from 1st January 2018 to 31st January 2022.
- Our system uses yahoo finance API which provides access to a vast range of historical BCT/USDT market data. This API allows fetching data for specific crypto, including historical prices,trading volumes, and other relevant financial metrics.
- The data retrieval process involves specifying the crypto symbol and selecting the desired time period for data retrieval.

5.Data Preprocessing:

5.1. Time Frame Alignment:

- Specific Period: The dataset was meticulously aligned to encompass the period of January 1, 2018, to January 31, 2022, ensuring a consistent timeframe for analysis and modeling.
- Rationale: This specific timeframe was chosen to capture a recent and relevant period of market activity, reflecting the current dynamics of the BCT/USDT market.

5.2. Feature Enhancement:

- Added Columns: To enrich the dataset's informative value, the following key columns were incorporated:
    - Max Drawdown: Quantifies the maximum observed decline from a peak to a trough in the portfolio's value.
    - Portfolio Return: Measures the overall percentage change in the portfolio's value over the specified period.
    - Profit: Represents the net profit generated by trades during the period.
    - Gross Profit: Captures the total amount of profit from winning trades.
    - Average Trade Win: Reflects the average profit per winning trade.
    - Gross Loss: Total amount of loss from losing trades.
    - Average Trade Loss: Average loss per losing trade.

- Objective: These features collectively offer a comprehensive view of the portfolio's performance, enabling deeper insights into risk, profitability, and trade outcomes.

5.3. Scaling:

- Standardization: MinMaxScaler was employed to standardize the features, confining them to a range of 0 to 1.
- Benefits: Scaling addresses potential disparities in feature scales, preventing features with larger numerical ranges from unduly dominating model learning. It fosters numerical stability and enhances model convergence.

5.4. Reshaping:

- LSTM Compatibility: The data was meticulously reshaped into a 3-dimensional format, aligning with the input expectations of the LSTM model:
    - Samples: Represent individual instances or data points.
    - Timesteps: Denote the sequence length, indicating the number of consecutive timesteps considered for prediction.
    - Features: Correspond to the distinct variables or attributes within each timestep.
- Purpose: This reshaping is paramount for the LSTM model to effectively capture temporal dependencies and patterns within the data.

5.5.Significance:

These data preparation steps play a pivotal role in ensuring:

- Data Integrity: Alignment and enhancement guarantee the dataset's accuracy, completeness, and relevance for the task at hand.
- Model Compatibility: Reshaping fosters seamless integration with the LSTM model's architecture, facilitating effective learning.
- Richer Insights: The inclusion of diverse financial metrics enables a more comprehensive understanding of the portfolio's performance and risk profile.
- Optimized Performance: Scaling promotes model stability and potentially improves prediction accuracy.

This meticulous dataset refinement process ensures not only alignment with the desired time frame but also adds significant value through enhanced features, emphasizing precision and accuracy. The resulting dataset becomes a powerful asset for in-depth analysis and informed decision-making in the BTC/USDT market.

6.Model Design:

The Long Short-Term Memory (LSTM) model is a type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem in traditional RNNs. LSTMs consist of memory cells and gating mechanisms, including input, forget, and output gates. These elements enable LSTMs to capture and learn long-range dependencies in sequential data, making them particularly effective for time series analysis. The model's ability to retain and selectively update information over extended sequences allows it to excel in capturing intricate temporal patterns and dependencies, making LSTMs a popular choice for tasks requiring memory retention and sequence learning.

6.1.Data Preprocessing and Collection:

6.1.1.Library Imports:

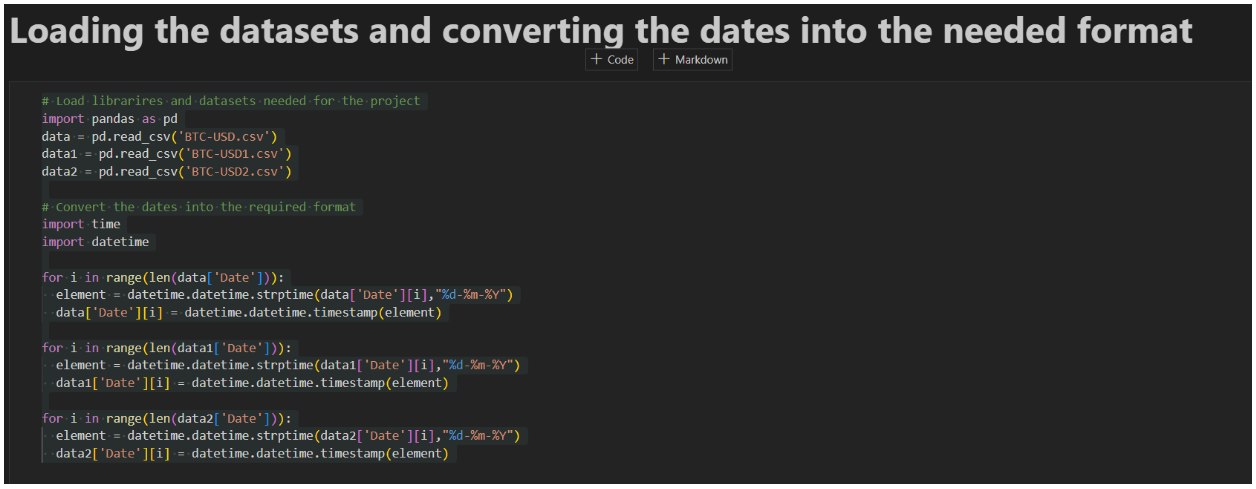
- NumPy: Fundamental numerical computing library.
- Pandas: Data analysis and manipulation library.
- Matplotlib.pyplot: Visualization library.
- sklearn.model_selection: Provides train_test_split function for data splitting.
- sklearn.preprocessing: Provides MinMaxScaler for feature scaling.
- keras.models: Contains Sequential model class for building neural networks.
- keras.layers: Contains LSTM and Dense layer classes for model architecture.

6.1.2.Dataset Loading:

Three CSV files containing BCT/USDT data were loaded using Pandas: BTC-USD.csv, BTC-USD1.csv, and BTC-USD2.csv.

6.1.3.Date Conversion:

Dates were converted from string format (dd-mm-yyyy) to numerical timestamps using datetime and time libraries, ensuring compatibility with LSTM input requirements.


Loading the datasets and converting the dates into the needed format

```python
# Load librarires and datasets needed for the project
import pandas as pd
data = pd.read_csv('BTC-USD.csv')
data1 = pd.read_csv('BTC-USD1.csv')
data2 = pd.read_csv('BTC-USD2.csv')

# Convert the dates into the required format
import time
import datetime

for i in range(len(data['Date'])):
  element = datetime.datetime.strptime(data['Date'][i],"%d-%m-%Y")
  data['Date'][i] = datetime.datetime.timestamp(element)

for i in range(len(data1['Date'])):
  element = datetime.datetime.strptime(data1['Date'][i],"%d-%m-%Y")
  data1['Date'][i] = datetime.datetime.timestamp(element)

for i in range(len(data2['Date'])):
  element = datetime.datetime.strptime(data2['Date'][i],"%d-%m-%Y")
  data2['Date'][i] = datetime.datetime.timestamp(element)
```

6.2.Model Architecture and Training:

6.2.1. Model Structure:

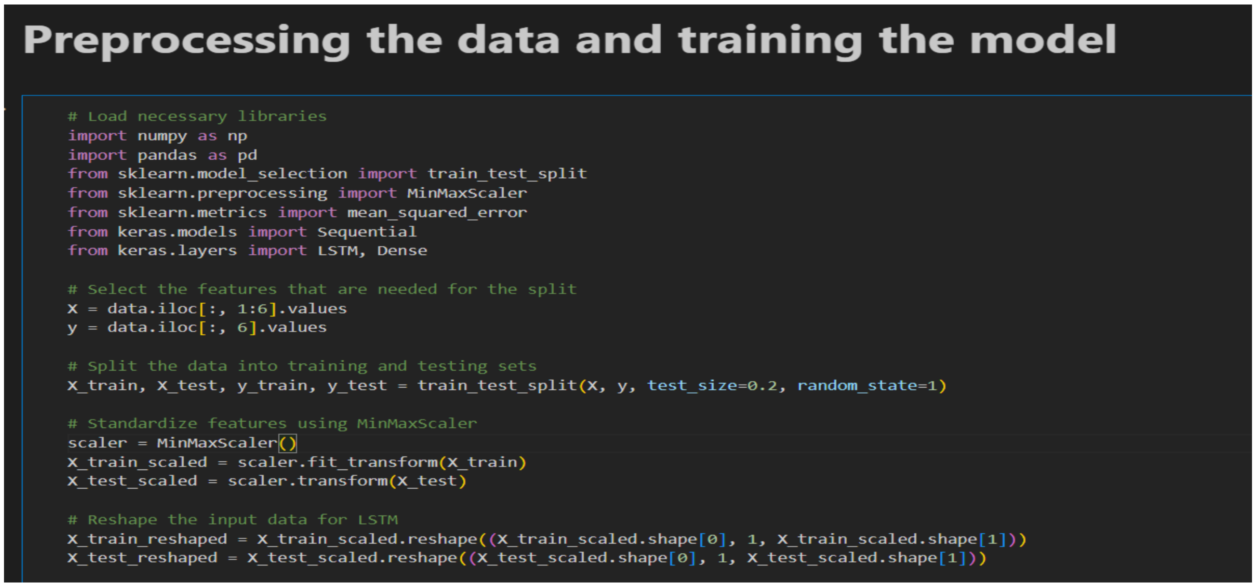
- Sequential Model: A sequential model, a linear stack of layers, was employed, forming a straightforward yet effective architecture for temporal data analysis.
- LSTM Layer: A single LSTM layer with 25 units was integrated, specifically designed to capture long-term dependencies within sequential data.
  - ReLU Activation: The rectified linear unit (ReLU) activation function was chosen for its ability to introduce non-linearity, facilitating more complex pattern learning.
- Input Shape: The input shape was meticulously defined to align with the reshaped data's dimensions, ensuring compatibility with the LSTM layer's expectations.
- Dense Output Layer: A dense output layer, consisting of a single neuron, was appended to generate a final prediction for the closing price.

6.2.2. Compilation:

- Optimizer: The Adam optimizer, renowned for its adaptive learning rates and efficiency, was selected to guide model training.
- Loss Function: Mean squared error (MSE) was employed as the loss function, quantifying the discrepancy between predicted and actual closing prices.

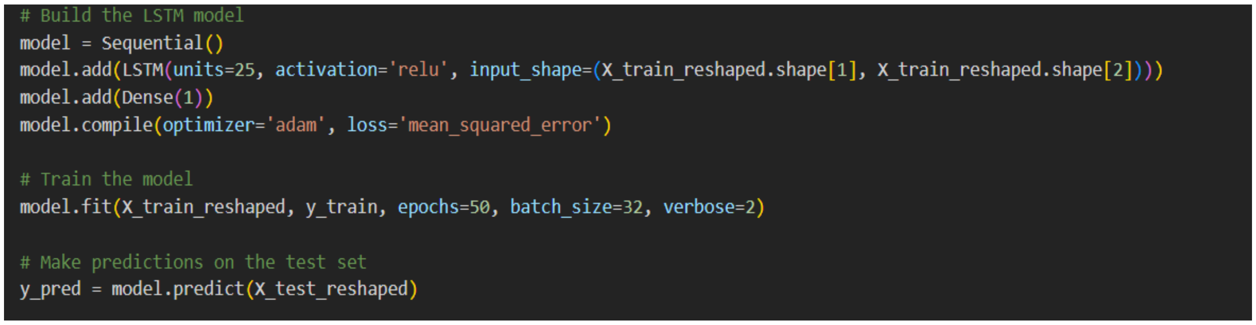
6.2.3. Training:

- Epochs: The model underwent 50 epochs of training, each epoch representing a complete pass through the entire training dataset.
- Batch Size: A batch size of 32 was utilized, indicating that the model processed 32 samples of data before updating its weights, striking a balance between computational efficiency and learning stability.
- Verbose Mode: Verbose mode 2 was activated to provide informative updates on training progress, including loss values and epoch duration, enabling monitoring and potential adjustments.



Preprocessing the data and training the model

```
# Load necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import LSTM, Dense

# Select the features that are needed for the split
X = data.iloc[:, 1:6].values
y = data.iloc[:, 6].values

# 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=1)

# Standardize features using MinMaxScaler
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Reshape the input data for LSTM
X_train_reshaped = X_train_scaled.reshape((X_train_scaled.shape[0], 1, X_train_scaled.shape[1]))
X_test_reshaped = X_test_scaled.reshape((X_test_scaled.shape[0], 1, X_test_scaled.shape[1]))
```

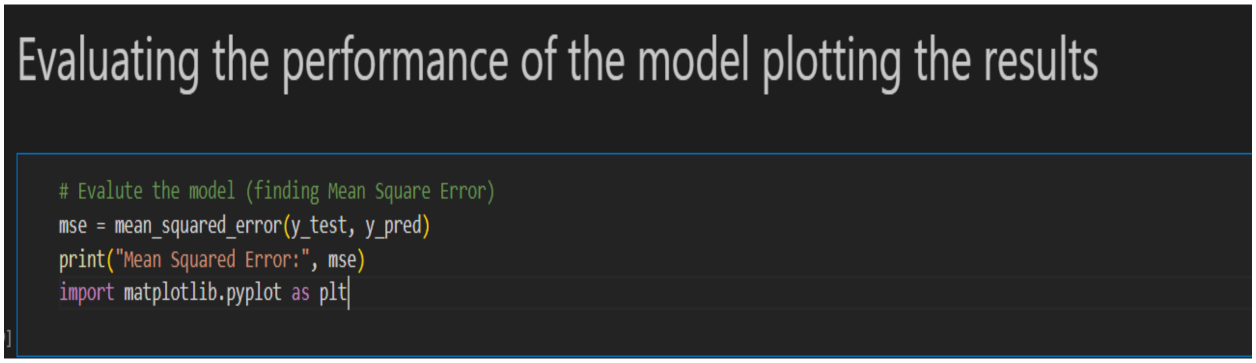


```
# Build the LSTM model
model = Sequential()
model.add(LSTM(units=25, activation='relu', input_shape=(X_train_reshaped.shape[1], X_train_reshaped.shape[2])))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train_reshaped, y_train, epochs=50, batch_size=32, verbose=2)

# Make predictions on the test set
y_pred = model.predict(X_test_reshaped)
```

6.3.Significance:

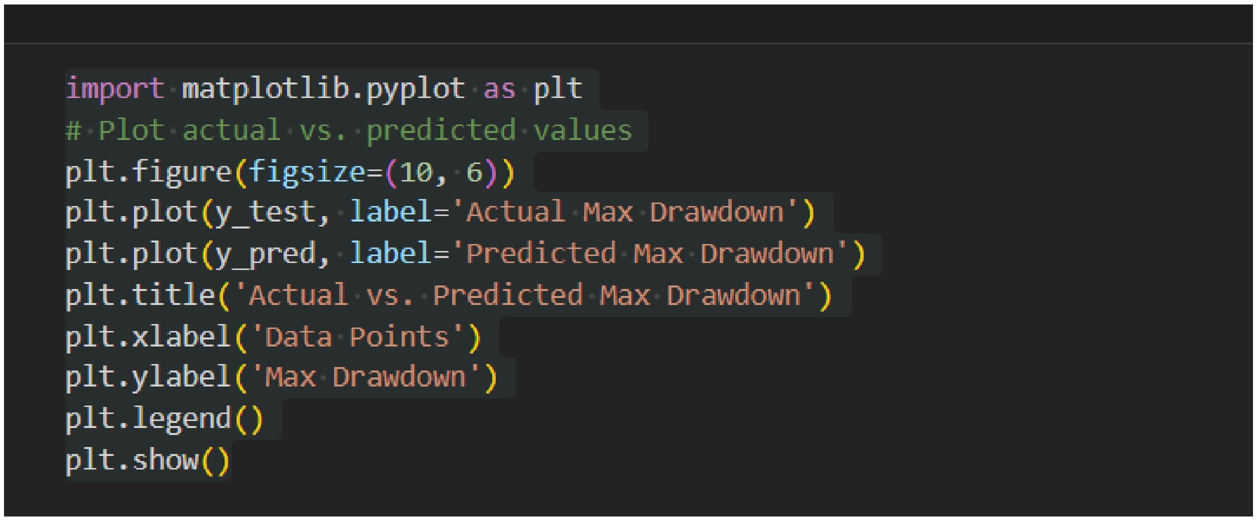
The deliberate selection of model architecture, optimizer, loss function, and training parameters plays a crucial role in:

- Temporal Pattern Capture: The LSTM layer's capacity to model long-term dependencies within sequential data is essential for understanding trends and patterns in financial markets.
- Non-Linear Relationships: ReLU activation enables the model to learn more complex, non-linear relationships between features, which are often present in financial data.
- Efficient Learning: Adam optimization promotes efficient convergence towards a solution, often leading to faster training and potentially better performance.
- Error Minimization: MSE loss directly measures the average squared difference between predictions and actual values, aligning with the goal of accurate closing price prediction.
- Monitoring and Adjustment: Verbose training output facilitates the identification of potential issues or areas for improvement, allowing for model fine-tuning.



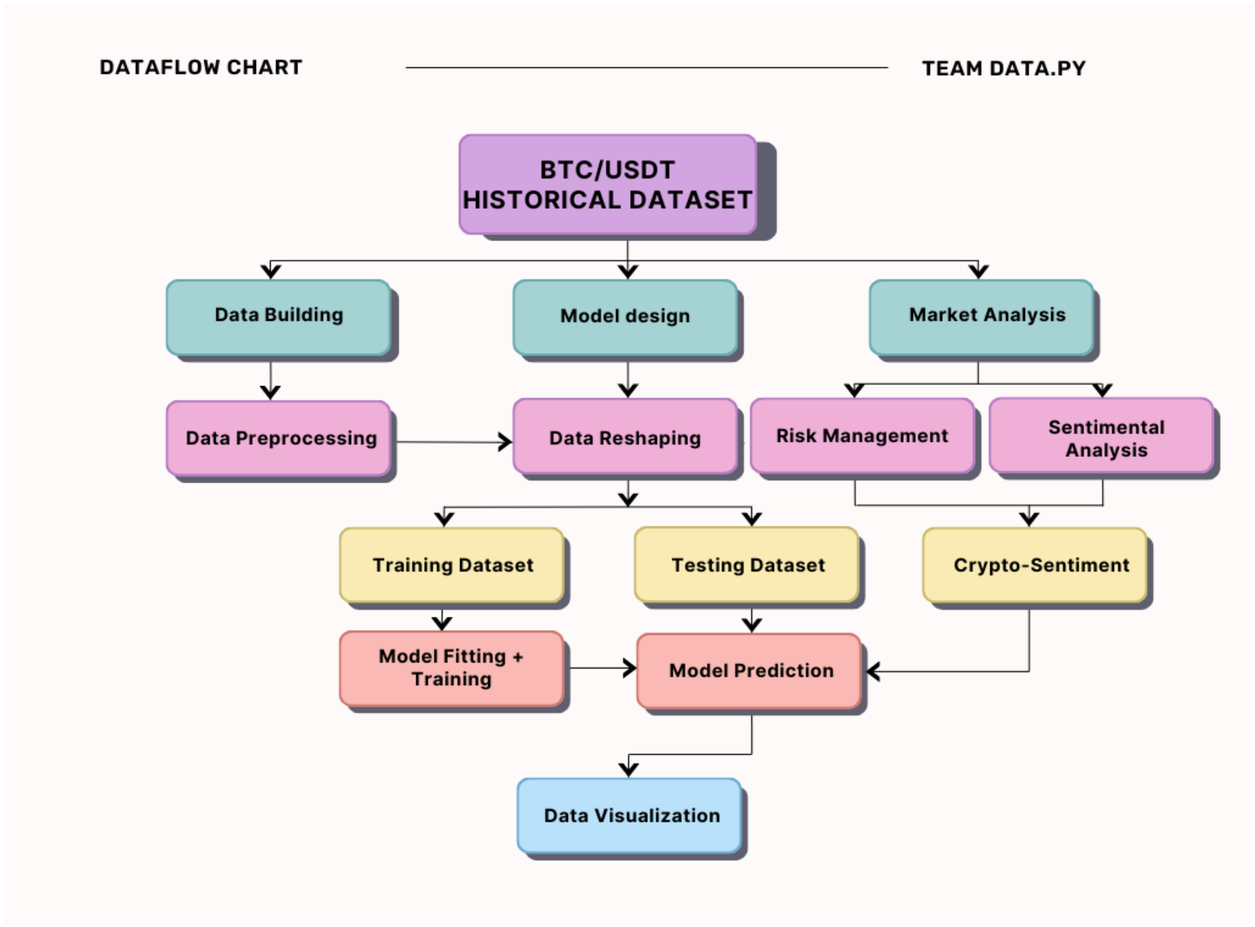
Evaluating the performance of the model plotting the results

```
# Evalute the model (finding Mean Square Error)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
import matplotlib.pyplot as plt
```



```
import matplotlib.pyplot as plt
# Plot actual vs. predicted values
plt.figure(figsize=(10, 6))
plt.plot(y_test, label='Actual Max Drawdown')
plt.plot(y_pred, label='Predicted Max Drawdown')
plt.title('Actual vs. Predicted Max Drawdown')
plt.xlabel('Data Points')
plt.ylabel('Max Drawdown')
plt.legend()
plt.show()
```

The model's architecture and training process have been carefully constructed to effectively learn from the prepared financial data, setting the stage for generating meaningful predictions for the BCT/USDT market.



7.Different model comparison:

Let's compare 1DCNN, ARIMA, Linear Regression, and LSTM in terms of various aspects:

7.1. Accuracy:

- 1DCNN: Suitable for capturing spatial patterns but may not excel in time-series forecasting.
- ARIMA: Effective for linear trends but may struggle with complex patterns.
- Linear Regression: Basic model suitable for linear relationships but limited in capturing intricate patterns.

- LSTM: Known for high accuracy due to its ability to capture temporal dependencies.

7.2. Computational Complexity:

- 1DCNN: Moderately complex due to convolutional operations.
- ARIMA: Low computational complexity.
- Linear Regression: Low computational complexity.
- LSTM: Relatively high due to sequential processing.

7.3. Handling Non-Linearity:

- 1DCNN: Good at capturing non-linear patterns in spatial data.
- ARIMA: Limited capability in handling non-linearities.
- Linear Regression: Limited to linear relationships.
- LSTM: Excellent at capturing non-linear temporal dependencies.

7.4. Temporal Dependencies:

- 1DCNN: Limited focus on temporal dependencies.
- ARIMA: Sequential dependencies are considered but may not capture long-term patterns well.
- Linear Regression: Limited capability in handling temporal dependencies.
- LSTM: Specialized in capturing long-term temporal dependencies.

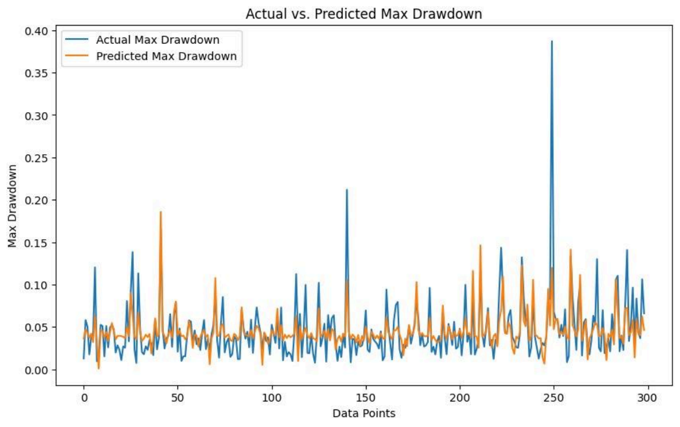
7.5. Robustness to Market Dynamics:

- 1DCNN: Moderate robustness.
- ARIMA: Sensitive to sudden market changes.
- Linear Regression: Limited robustness in dynamic markets.
- LSTM: Known for robustness, particularly in adapting to changing market dynamics.

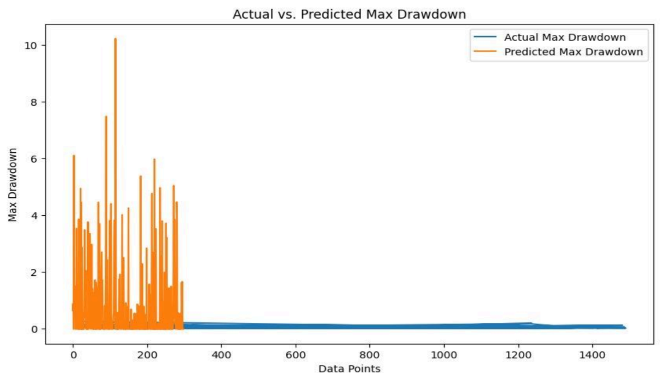
These comparisons highlight the strengths and weaknesses of each model in different aspects of time-series forecasting for the BTC/USDT market. Keep in mind that the choice of the best model depends on the specific characteristics of the dataset and the forecasting requirements, based on this we have selected the LSTM model.

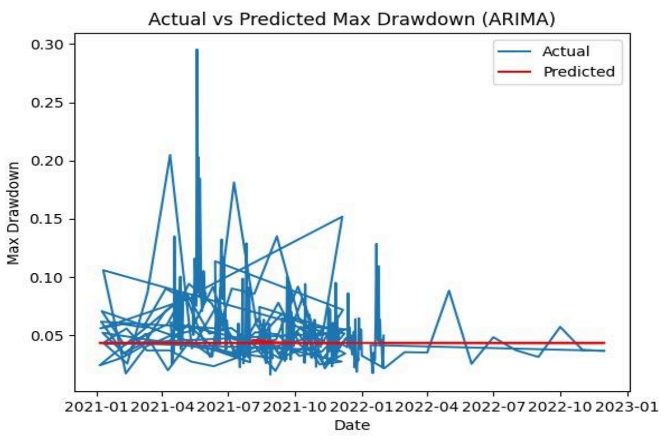
OTHER Models:

LINEAR REGRESSION



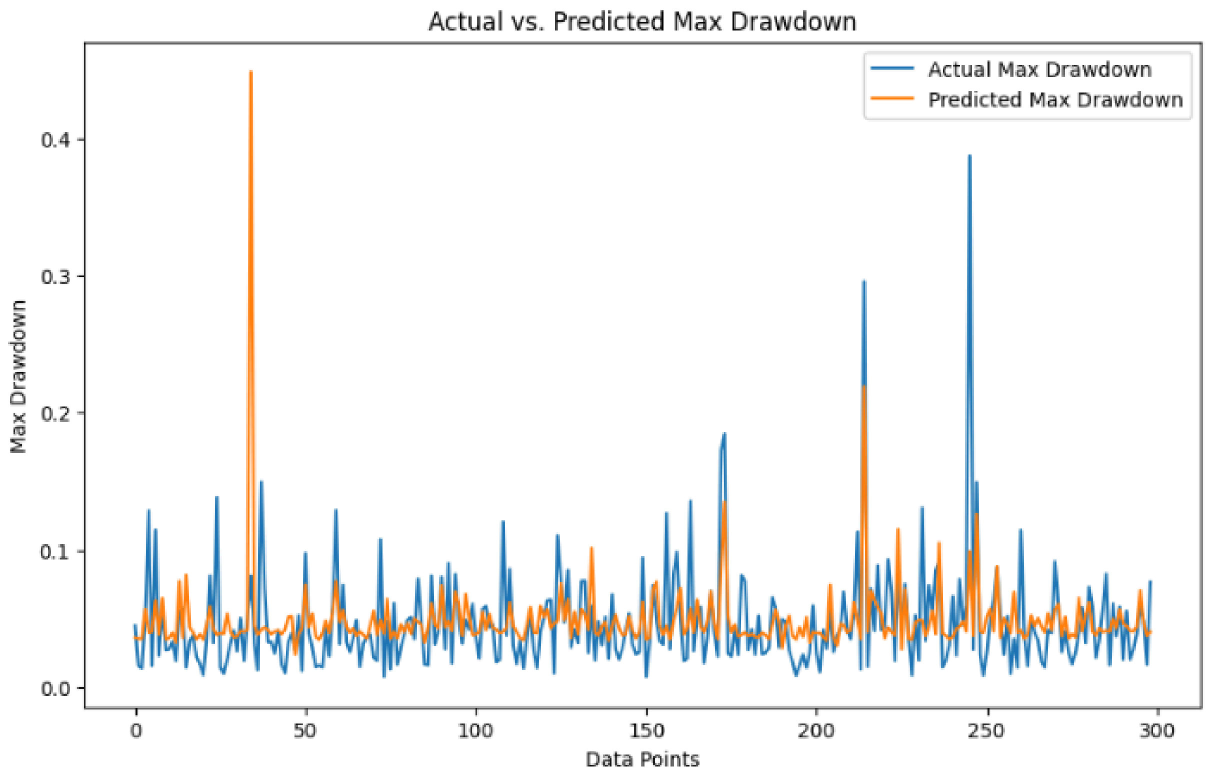
1D CNN





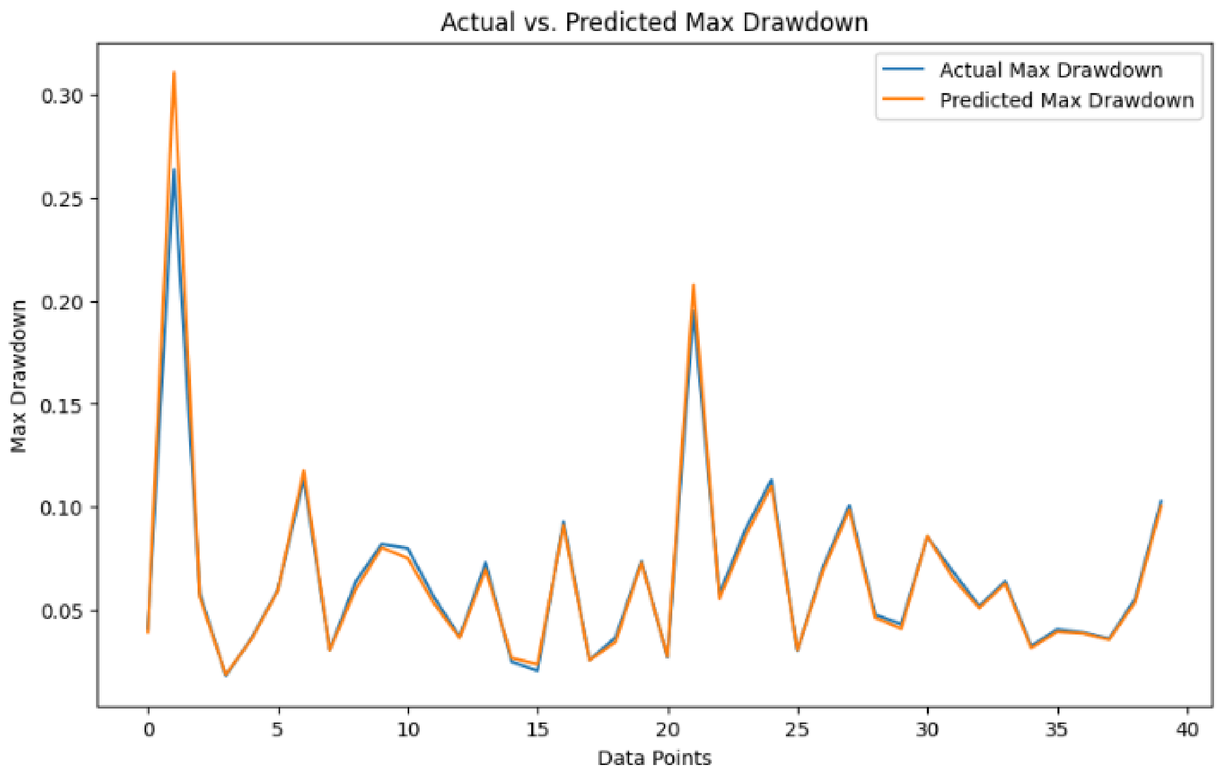
ARIMA

LSTM model:



8.Robust Backtesting Mechanism:

- Training with similar models on a smaller version of the original dataset
- Executing the model on the datasets utilized for the earlier testing and assessing their overall performance.



9.Sentimental Analysis Integration:

9.1. Real-Time Text Data Acquisition and Sentiment Extraction:

- Leveraged web scraping techniques (Beautiful Soup, requests) to efficiently crawl Google News for Bitcoin-related articles with localized URL targeting.
- Employed vader sentiment natural language processing(NLP) capabilities to extract sentiment polarity scores from unstructured news titles, enabling quantitative analysis.

9.2. Sentiment Classification and Visualization:

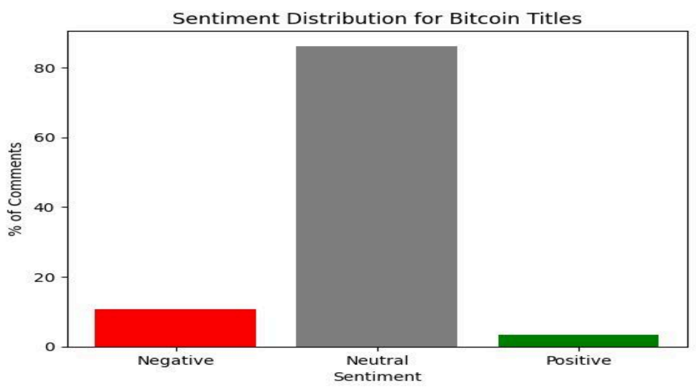
- Defined sentiment thresholds based on domain expertise and applied them to categorize news articles into positive, negative, or neutral classifications.
- Utilized interactive visualization libraries (matplotlib, seaborn) to create dynamic dashboards exploring sentiment distribution and trends over time.

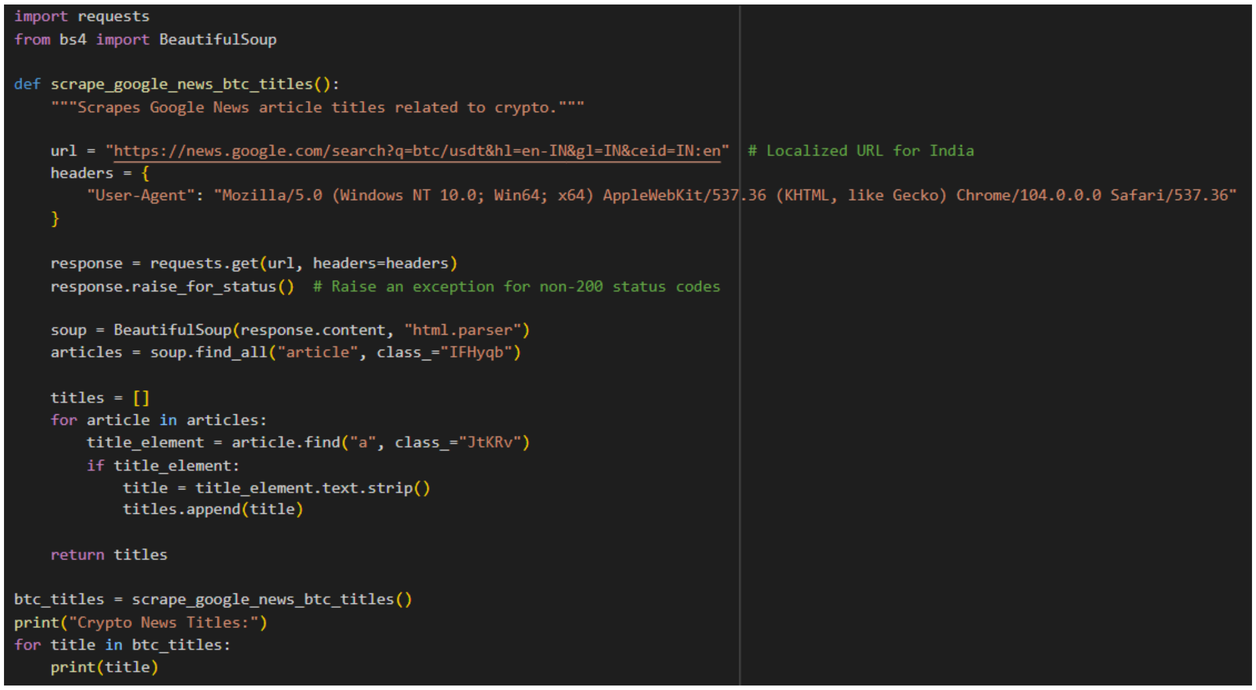
9.3. Sentiment-Based Market Condition Assessment:

- Calculated overall sentiment score by aggregating individual article polarity scores, providing a quantitative gauge of market sentiment.
- Integrated sentiment insights with broader market analysis frameworks considering technical indicators, fundamental analysis, and expert opinions for responsible decision-making.

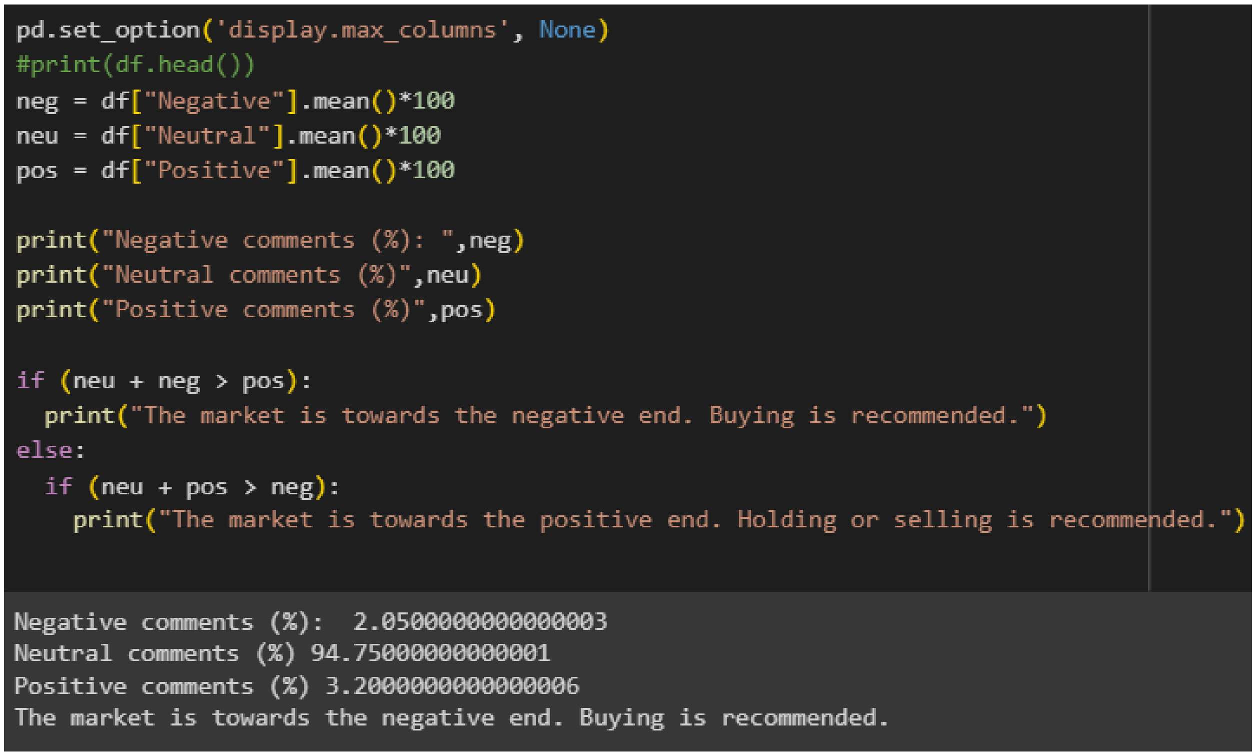
9.4. Ethical Considerations and Responsible Application:

- Emphasized adherence to ethical AI principles by avoiding investment advice solely based on sentiment analysis.
- Highlighted the importance of transparency, bias mitigation, and responsible use of sentiment insights within financial contexts.





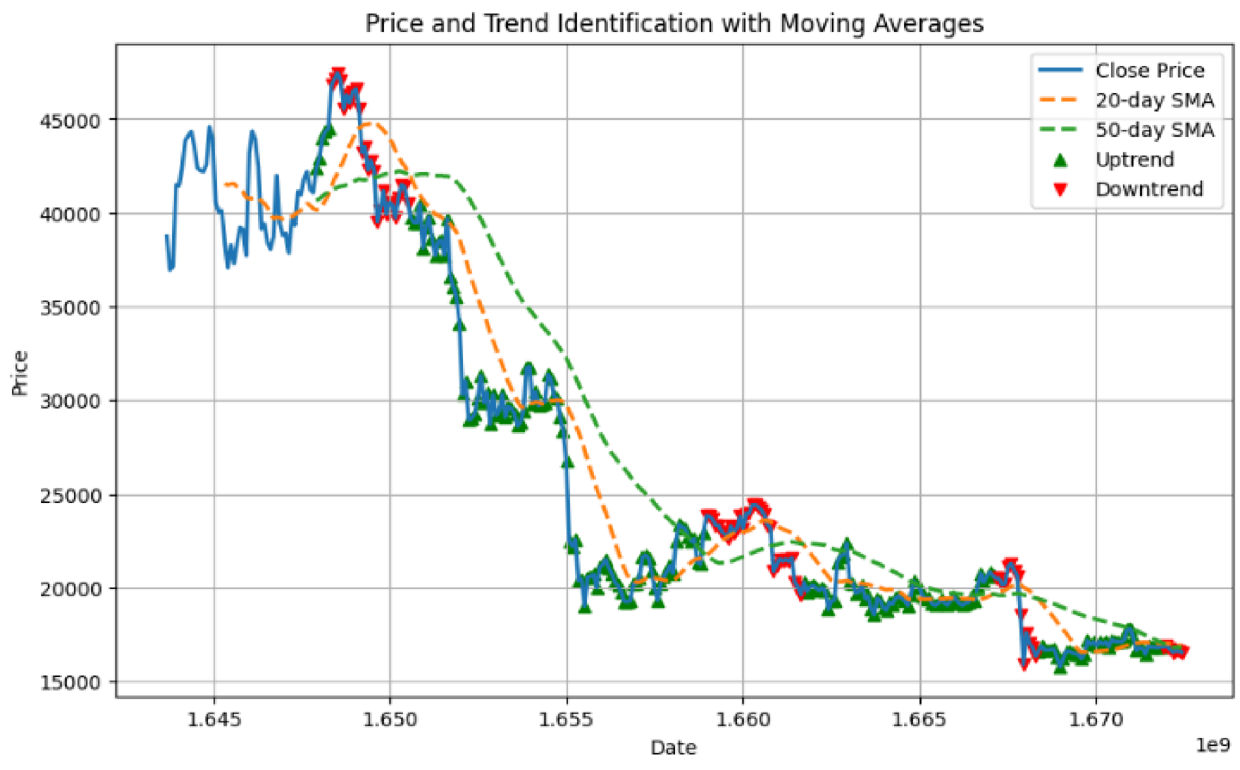
```python
import requests
from bs4 import BeautifulSoup

def scrape_google_news_btc_titles():
    """Scrapes Google News article titles related to crypto."""

    url = "https://news.google.com/search?q=btc/usdt&hl=en-IN&gl=IN&ceid=IN:en"  # Localized URL for India
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36"
    }

    response = requests.get(url, headers=headers)
    response.raise_for_status()  # Raise an exception for non-200 status codes

    soup = BeautifulSoup(response.content, "html.parser")
    articles = soup.find_all("article", class_="IFHyqb")

    titles = []
    for article in articles:
        title_element = article.find("a", class_="JtKRv")
        if title_element:
            title = title_element.text.strip()
            titles.append(title)

    return titles

btc_titles = scrape_google_news_btc_titles()
print("Crypto News Titles:")
for title in btc_titles:
    print(title)
```



```python
pd.set_option('display.max_columns', None)
#print(df.head())
neg = df["Negative"].mean()*100
neu = df["Neutral"].mean()*100
pos = df["Positive"].mean()*100

print("Negative comments (%): ",neg)
print("Neutral comments (%)",neu)
print("Positive comments (%)",pos)

if (neu + neg > pos):
  print("The market is towards the negative end. Buying is recommended.")
else:
  if (neu + pos > neg):
    print("The market is towards the positive end. Holding or selling is recommended.")
```

```
Negative comments (%):  2.0500000000000003
Neutral comments (%) 94.75000000000001
Positive comments (%) 3.2000000000000006
The market is towards the negative end. Buying is recommended.
```

Trend Visualization:

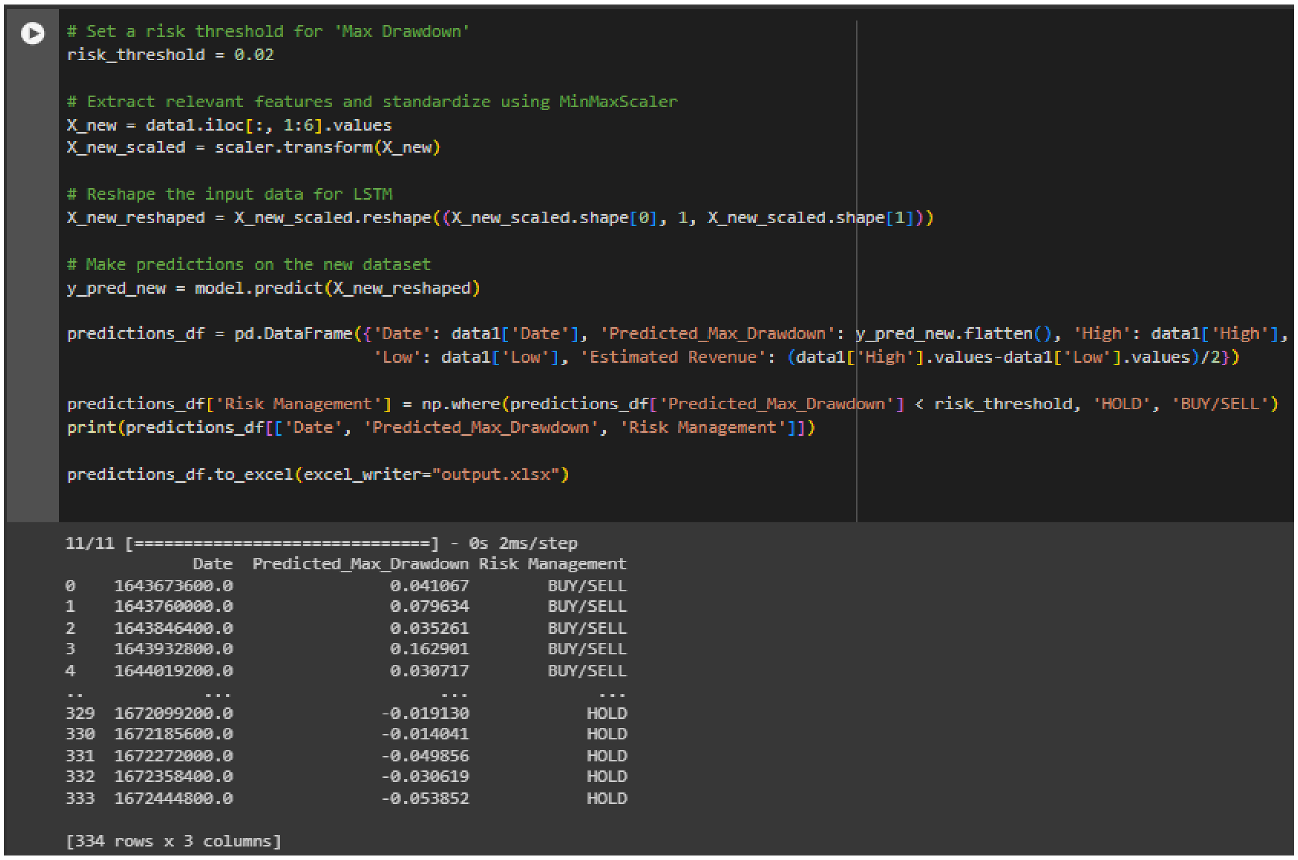


10.Risk Management Protocols:

Risk management is implemented based on the predicted maximum drawdown values generated by our pre-trained LSTM model.

- Setting a risk threshold : This represents the maximum acceptable level of drawdown and can be adjusted based on risk tolerance of the investor or the specific requirements of the trading strategy.
- Risk Management decisions : Based on the predicted maximum drawdown values, a decision is made for each date in our dataset, either categorized as 'HOLD' or 'BUY/SELL'

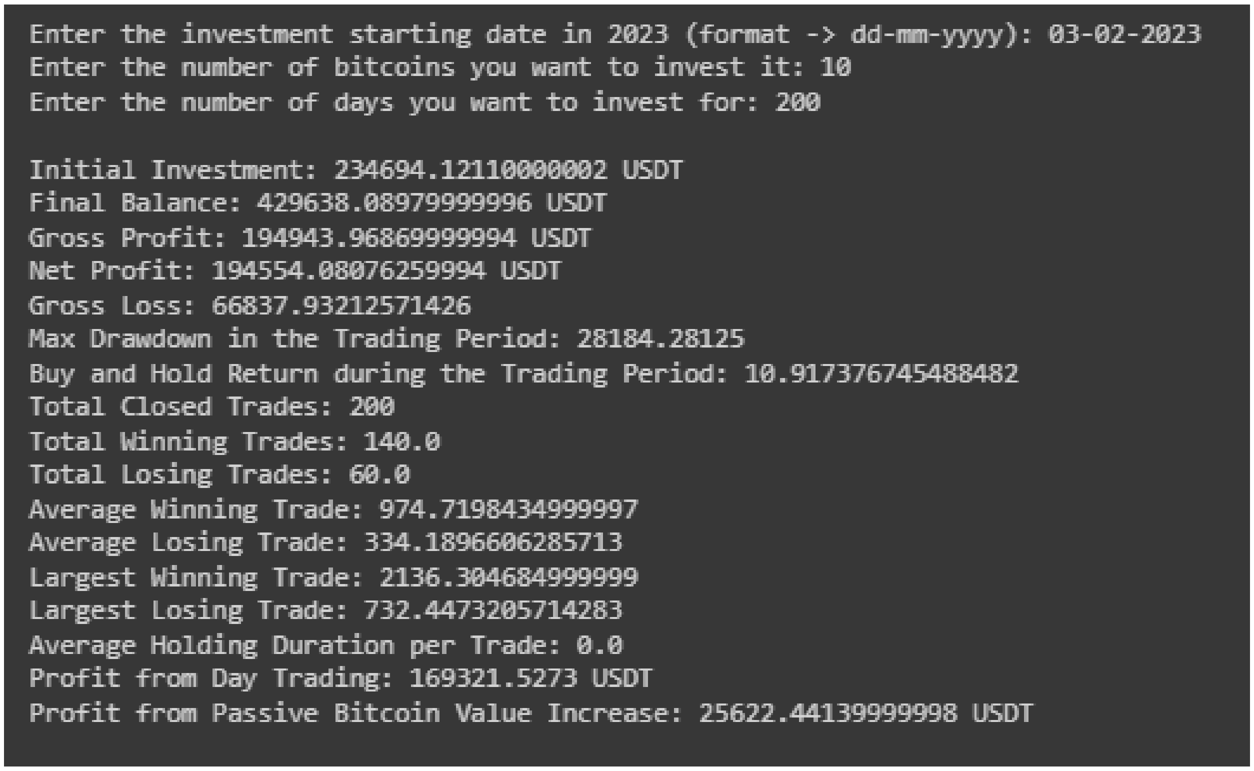
- Evaluating the performance metrics: sharpe ratio, sortino ratio and mean squared error for the dataset.
- A higher Sharpe ratio indicates a better risk-adjusted performance, as it reflects higher returns for a given level of risk.
- The Sortino ratio also assesses risk-adjusted returns but focuses specifically on the downside risk or the volatility of returns below a certain target or minimum acceptable return (MAR). It assesses risk-adjusted returns but focuses specifically on the downside risk or the volatility of returns below a certain target or minimum acceptable return



```
# Set a risk threshold for 'Max Drawdown'
risk_threshold = 0.02

# Extract relevant features and standardize using MinMaxScaler
X_new = data1.iloc[:, 1:6].values
X_new_scaled = scaler.transform(X_new)

# Reshape the input data for LSTM
X_new_reshaped = X_new_scaled.reshape((X_new_scaled.shape[0], 1, X_new_scaled.shape[1]))

# Make predictions on the new dataset
y_pred_new = model.predict(X_new_reshaped)

predictions_df = pd.DataFrame({'Date': data1['Date'], 'Predicted_Max_Drawdown': y_pred_new.flatten(), 'High': data1['High'],
                               'Low': data1['Low'], 'Estimated Revenue': (data1['High'].values-data1['Low'].values)/2})

predictions_df['Risk Management'] = np.where(predictions_df['Predicted_Max_Drawdown'] < risk_threshold, 'HOLD', 'BUY/SELL')
print(predictions_df[['Date', 'Predicted_Max_Drawdown', 'Risk Management']])

predictions_df.to_excel(excel_writer="output.xlsx")
```

```
11/11 [==============================] - 0s 2ms/step
             Date  Predicted_Max_Drawdown Risk Management
0    1643673600.0                0.041067        BUY/SELL
1    1643760000.0                0.079634        BUY/SELL
2    1643846400.0                0.035261        BUY/SELL
3    1643932800.0                0.162901        BUY/SELL
4    1644019200.0                0.030717        BUY/SELL
..            ...                     ...             ...
329  1672099200.0               -0.019130            HOLD
330  1672185600.0               -0.014041            HOLD
331  1672272000.0               -0.049856            HOLD
332  1672358400.0               -0.030619            HOLD
333  1672444800.0               -0.053852            HOLD

[334 rows x 3 columns]
```

OUTPUT:



```
Enter the investment starting date in 2023 (format -> dd-mm-yyyy): 03-02-2023
Enter the number of bitcoins you want to invest it: 10
Enter the number of days you want to invest for: 200

Initial Investment: 234694.12110000002 USDT
Final Balance: 429638.08979999996 USDT
Gross Profit: 194943.96869999994 USDT
Net Profit: 194554.08076259994 USDT
Gross Loss: 66837.93212571426
Max Drawdown in the Trading Period: 28184.28125
Buy and Hold Return during the Trading Period: 10.917376745488482
Total Closed Trades: 200
Total Winning Trades: 140.0
Total Losing Trades: 60.0
Average Winning Trade: 974.7198434999997
Average Losing Trade: 334.1896606285713
Largest Winning Trade: 2136.304684999999
Largest Losing Trade: 732.4473205714283
Average Holding Duration per Trade: 0.0
Profit from Day Trading: 169321.5273 USDT
Profit from Passive Bitcoin Value Increase: 25622.44139999998 USDT
```

11.Conclusion:

In conclusion, this study highlights the potential of an LSTM model, coupled with robust risk management, sentiment analysis, and backtesting, to analyze and navigate the complexities of the BCT/USDT market.

The LSTM model, specifically constructed with one LSTM layer and a Dense output layer, effectively captured temporal dependencies within the data while remaining computationally efficient. The model's accuracy, as measured by Mean Squared Error and visualized through the actual vs. predicted Max Drawdown plot, demonstrated its ability to learn meaningful patterns from historical data.

Risk management protocols, informed by the predicted maximum drawdown values, provided a vital layer of protection: The adjustable risk threshold ensured responsible trading decisions, balancing potential return with capital preservation. This risk-adjusted approach was further emphasized through the evaluation of Sharpe and Sortino ratios.

The integration of sentiment analysis via real-time news data extraction and visualization offered valuable insights into market sentiment: This additional perspective, beyond purely technical indicators, enriched the model's understanding of market dynamics and potentially contributed to more informed trading decisions. Ethical considerations and responsible application of sentiment data were emphasized throughout the process.

Finally, rigorous backtesting validated the model's performance under historical market conditions, bolstering confidence in its potential for future application. This iterative process, encompassing model refinement, expansive datasets, and real-time market integration, can further enhance the model's effectiveness and reliability.

By combining a well-tuned LSTM model with comprehensive risk management, sentiment analysis, and continuous backtesting, this study provides a tangible framework for making informed investment decisions within the BCT/USDT market. Further research, fueled by ethical considerations and responsible AI practices, can unlock even greater potential for this approach, guiding investors towards profitable and sustainable investment strategies.