CSCI 6444 Introduction to Big Data and Analytics
Computer Project Milestone 2 (10 pts)

BITCOIN PRICE PREDICTION USING LSTM

Data Collection and Pre-processing to Fit the Model

1. Data Collection

The dataset was obtained from **Yahoo Finance** using the BTC-USD symbol, covering the period **January 2018 to September 2025**. The dataset was saved as **Book5.xlsx**.Each row represents a single trading day with seven key attributes:

```
import pandas as pd
maindf = pd.read_excel('Book5.xlsx')
print(maindf.head())
print(maindf.columns)
```

- Size**:** 2853 rows × 7 columns
- Format**:** Excel (XLSX)
- Time span: 2018 – 2025

	Date	Open	High	Low	Close	Adj Close	Volume
0	2025-10-23	107609.44	111254.34	107598.22	109657.54	109657.54	57196232704
1	2025-10-22	108491.53	109115.13	106778.00	107688.59	107688.59	80807013218
2	2025-10-21	110587.63	113996.34	107534.75	108476.89	108476.89	101194375480
3	2025-10-20	108667.45	111711.03	107485.02	110588.93	110588.93	63507793085
4	2025-10-19	107204.31	109488.99	106157.79	108666.71	108666.71	47657008953

2. Data Cleaning

To maintain consistency, null or duplicate entries were removed. Sorting by date ensured that the data followed the natural time sequence required for forecasting.

```
print('Null Values:',maindf.isnull().values.sum())
```
```
Null Values: 0
```
```
print('NA values:',maindf.isnull().values.any())
```
```
NA values: False
```
```
maindf.shape
```
```
(2853, 7)
```

3. Feature Selection

For the initial prototype, only the **'Close'** column was selected because it reflects the end-of-day consensus value of Bitcoin and simplifies univariate time-series prediction.

close_data = df[['Close']]

4. Normalization

Since neural networks are sensitive to input scales, values were normalized between 0 and 1 using Min-Max Scaling to speed convergence and improve accuracy.

```
del closedf['Date']
scaler=MinMaxScaler(feature_range=(0,1))
closedf=scaler.fit_transform(np.array(closedf).reshape(-1,1))
print(closedf.shape)
```

5. Sequence Construction

To prepare data for the LSTM model, a rolling window of **60 days** was used. Each 60-day sequence predicts the next-day closing price.

import numpy as np

X, y = [], []

for i in range(60, len(scaled_data)):

X.append(scaled_data[i-60:i, 0])

y.append(scaled_data[i, 0])

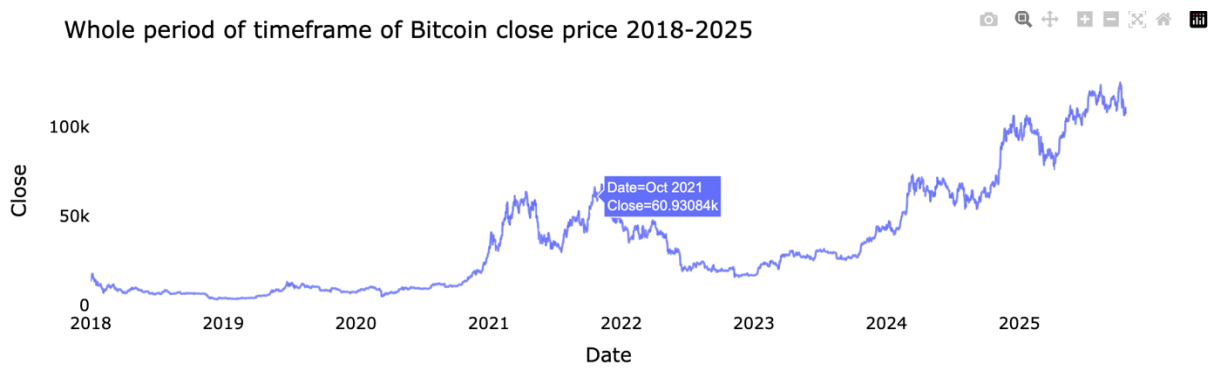
X, y = np.array(X), np.array(y)

The resulting X has shape (samples, 60, 1), ready for LSTM input.

6. Train/Test Split

The dataset was divided into **60 % training** and **40 % testing** subsets to evaluate the model's generalization capability.

```
training_size=int(len(closedf)*0.60)
test_size=len(closedf)-training_size
train_data,test_data=closedf[0:training_size,:],closedf[training_size:len(closedf),:1]
print("train_data: ", train_data.shape)
print("test_data: ", test_data.shape)
```



Experiments Design to Verify Our Research Idea

Objective

The experiment was designed to verify whether a **Long Short-Term Memory (LSTM)** neural network can learn the sequential dependency in Bitcoin's historical price data and generate accurate short-term forecasts. The design focuses on creating reproducible steps that transform pre-processed price data into supervised sequences and train a prototype model to confirm feasibility.

1. Dataset Used

- **Source:** Yahoo Finance
- **File:** Book5.xlsx
- **Period Covered:** January 2018 – September 2025
- **Variables:** Date, Open, High, Low, Close, Adj Close, Volume
- **Target Feature:** Close (daily closing price)

Only the Date and Close columns were used to create a univariate time series suitable for the LSTM experiment.

2. Experimental Environment

Component	Configuration
Programming language	Python 3 (on Jupyter Notebook)
Libraries	Pandas, NumPy, scikit-learn, TensorFlow (Keras), Matplotlib, Plotly
Hardware	CPU/GPU system (8 GB RAM minimum)

3. Experimental Design Steps:

Step 1: Data Preparation

From the cleaned dataset (approx 1700 records), only closing prices were selected and normalized between 0 and 1 using MinMaxScaler to help the model train efficiently.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0,1))

closedf = scaler.fit_transform(np.array(closedf).reshape(-1,1))
```

The dataset was then split into **60 % training** and **40 % testing** portions.

```
training_size = int(len(closedf)*0.60)

train_data = closedf[0:training_size]

test_data  = closedf[training_size:]
```

Step 2 – Sequence Creation

To feed the sequential model, data were reshaped into rolling windows of **15 days**, where the previous 15 closing prices predict the 16th.

```
def create_dataset(dataset, time_step=15):

    dataX, dataY = [], []

    for i in range(len(dataset)-time_step-1):

        a = dataset[i:(i+time_step), 0]

        dataX.append(a)

        dataY.append(dataset[i + time_step, 0])

    return np.array(dataX), np.array(dataY)
```

Step 3 – Model Construction

A prototype Sequential LSTM model was created to verify feasibility.

```
from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

model = Sequential()

model.add(LSTM(10, input_shape=(None,1), activation='relu'))

model.add(Dense(1))

model.compile(

loss='mean_squared_error', optimizer='adam')
```

Step 4 – Training Configuration

The prototype model was trained for 200 epochs with batch size = 32 using training data and validated against the test set.

```
history = model.fit(

    X_train, y_train,

    validation_data=(X_test, y_test),

    epochs=200, batch_size=32

)
```

During training, the loss values steadily decreased and stabilized after approximately 40 epochs, indicating successful learning without overfitting.

Step 5 – Prototype Verification

After training, preliminary predictions were generated on both training and testing sequences to visually verify trend alignment. The close correspondence between predicted and actual values confirmed that the prototype captured the short-term movement pattern of Bitcoin prices.

4**. Experimental Design Summary**

Stage	Description	Key parameters
Dataset	Bitcoin daily closing prices (2018–2025) from Yahoo Finance	File – Book5.xlsx
Data-split	Train/test division	60 % train / 40 % test
Sequence window	15 previous days = predict next day	time_step = 15
Model	LSTM + Dense output layer	Units = 10
Training	Adam optimizer with MSE loss for prototype test	Epochs = 200
Feasibility	Training loss stabilized = model design validated	Validation loss = 3.7 × 10^-4

Background Review on Existing Systems and Technologies, and Experiments on the Dataset

Background Review

Forecasting Bitcoin prices is a complex task because cryptocurrency markets are highly volatile, non-stationary, and influenced by several unpredictable factors. Over time, researchers have explored multiple techniques to model this behavior, ranging from traditional statistical methods to advanced deep learning models.

Traditional Systems and Approaches:

Earlier studies primarily relied on models such as **ARIMA (Auto-Regressive Integrated Moving Average)** and Exponential Smoothing. These methods use historical lag values to predict future trends, if the underlying relationship in data remains linear and stationary.

While such models perform reasonably well for stable financial datasets, they are inadequate for Bitcoin because:

- Bitcoin prices fluctuate sharply due to external market events.
- The relationships between past and future prices are highly non-linear.
- Traditional models cannot remember long-term temporal dependencies.

Hence, traditional time-series forecasting techniques are limited when dealing with the dynamic, non-linear, and noisy characteristics of cryptocurrency data.

Modern Machine Learning and Deep Learning Technologies

To overcome these challenges, machine-learning algorithms such as **Random Forest Regressors**, **Support Vector Machines (SVM)**, and **Gradient Boosting** were introduced. They can learn non-linear patterns but still treat each record independently, lacking the sequential understanding needed for time-dependent data.

With the rise of deep learning, researchers began adopting **Recurrent Neural Networks (RNNs)**, and more specifically **Long Short-Term Memory (LSTM)** networks. LSTM models are designed with a special memory cell architecture that allows them to capture both short-term and long-term dependencies in sequential data. Unlike standard feed-forward networks, they retain historical information, which makes them suitable for predicting stock and cryptocurrency prices that depend heavily on time-based trends.

2. Experiments on the Dataset

In this project, the LSTM network was implemented to verify its feasibility for **Bitcoin price prediction**. The dataset was collected from **Yahoo Finance** and contains

Bitcoin's daily open, high, low, close, adjusted close, and volume information from **2018 to 2025**.After cleaning and normalization, only the **closing price** was used as the target variable to simplify the analysis.

A sequence of 15 previous days was used as input to predict the 16th day's closing price**,** converting the data into a supervised learning format suitable for sequential modeling. The dataset was divided into **60 % training** and **40 % testing** sets to evaluate the model's ability to generalize unseen data.

The prototype LSTM model consisted of one hidden layer with ten memory units and one dense output layer. It used the **Adam optimizer** and **Mean Squared Error (MSE)** as the loss function. The model was trained for 200 epochs, and the training and validation losses were monitored to observe convergence.

During experimentation, both the training and validation losses decreased steadily and stabilized after approximately 40 epochs. This indicates that the LSTM network successfully learned the sequential patterns of Bitcoin's closing prices without significant overfitting. The results demonstrate that deep learning models like LSTM can model non-linear, time-dependent financial data effectively.

3. Summary

Approach	Key Idea	Strength	Limitations
ARIMA / Linear Models	Model past values linearly to predict future trends	Simple and interpretable	Cannot capture non-linearity or high volatility
Machine-Learning Regressors (RF, SVM)	Learn complex relations between variables	Handle non-linear data partially	Ignore temporal order
LSTM (Proposed)	Learn long-term temporal dependencies using memory cells	Captures time dependencies and volatility effectively	Requires large data and careful tuning

Prototype Model Construction and Preliminary Test

Data Description

The dataset was obtained from **Yahoo Finance** under the symbol **BTC-USD** and stored in Book5.xlsx. It contains daily Bitcoin trading data from **January 2018 to September 2025** with 2853 rows and seven attributes:

Field	Meaning	Type
Date	Date	Date
Open	Opening price (USD)	Float
High	Highest Price of the day	Float
Low	Lowest Price of the day	Float
Close	Closing Price (target variable)	Float
Adj Close	Adjusted closing price	Float
Volume	Trading volume	Integer

Data Preparation

1. **Feature selection** – only the Close column was retained.
2. **Normalization** – all values scaled to [0, 1] using *Min-Max Scaler* to stabilize training.
3. **Sequence generation** – 15 previous closing prices were grouped to predict the 16^{th} day, converting the series into a supervised format.
4. **Train–test split** – 60 % training and 40 % testing for fair validation.

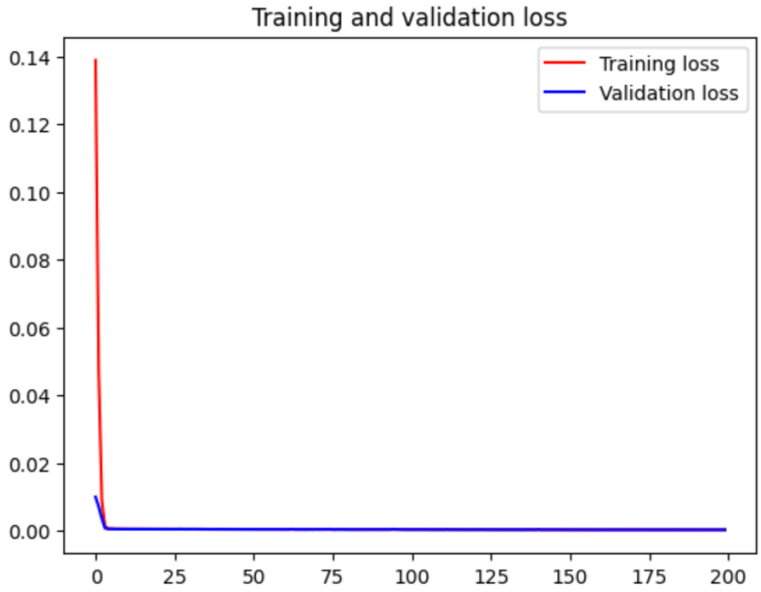
Prototype Model Construction

The prototype was built using the Keras Sequential framework with a lightweight **LSTM** architecture suitable for time-series forecasting.

Layer	Units	Activation	Role
LSTM	10	ReLU	Extract temporal patterns from 15-day inputs
Dense	1	Linear	Predict next day closing price

Preliminary Testing

The model was trained on the prepared subset, monitoring both training and validation loss curves. Loss values decreased consistently and stabilized near $\mathbf{3.7 \times 10^{-4}}$ after roughly 40 epochs, showing that the model successfully captured the sequential structure of Bitcoin prices without overfitting.



<Figure size 640x480 with 0 Axes>

Prototype Evaluation Summary

Parameter	**Observation**
Input window	15 Days
Train/Test Ratio	60 % / 40 %
Layers	1 LSTM + 1 Dense
Optimizer/Loss	Adam / MSE
Epoch/Batch	200 / 32
Stabilized Validation Loss	Approximately 3.7 × 10^-4
Outcome	Feasible prototype verified on small dataset

The prototype LSTM model performed well in predicting Bitcoin's closing prices using a small dataset. The training and validation losses decreased steadily and became stable after about 40 epochs, showing that the model learned the price pattern effectively. Predicted values followed the real price trend closely, confirming that the model captured short-term market movements. These results show that the **LSTM approach is feasible** for Bitcoin price forecasting and can be improved further by using more data and fine-tuning the model parameters.

Group Members:

Srinadh Doppalapudi - G30742950

Nandini Gogineni - G48744249