A Project report on

" STOCK PRICE PREDICTOR"

submitted in partial fulfillment of the Academic requirements for the award of the degree of

Bachelor of Technology

Submitted by

K.GAYATHRI	**- (23H51A05FR)**
K.SHALINI	**- (23H51A05G1)**
K.AKHILA	**- (23H51A05GC)**

UNDER THE COURSE
REAL TIME RESEARCH PROJECT

CMR COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous)
(NAAC Accredited with 'A+' Grade & NBA Accredited)
(Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)
KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501401

2024-25

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMUS)

(NAAC Accredited with 'A+' Grade & NBA Accredited) (Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)

KANDLAKOYA, MEDCHAL ROAD,

HYDERABAD-501401 2024-25

CERTIFICATE

This is to certify that a Project entitled with " Stock Price Predictor" is being Submitted By

K.GAYATHRI	**23H51A05FR**
K.SHALINI	**23H51A05G1**
K. AKHILA	**23H51A05GC**

In partial fulfillment of the requirement for completion of the **" REAL TIME RESEARCH PROJECT" -** of II-B. Tech IV- Semester is a record of a bonafide work carried out under guidance and supervision.

Signature of Faculty **Signature of HOD**

ACKNOWLEDGEMENT

We are obliged and grateful to thank, CMRCET, for his cooperation in all respects during the course.

We would like to thank the Principal of CMRCET, Major. Dr. V. A. Narayana, for his support in the course of this project work.

Finally, we thank all our faculty members and Lab Programmer for their valid support.

We own all our success to our beloved parents, whose vision, love and inspiration has made us reach out for these glories.

K.GAYATHRI **23H51A05FR**
K.SHALINI **23H51A05G1**
K. AKHILA **23H51A05GC**

TABLE OF CONTENTS

ABSTRACT

This project focuses on building a Stock Price Predictor using machine learning models to forecast future prices based on historical market data. The system aims to help traders, investors, and financial analysts by providing data-driven insights into potential price trends. Key techniques used include Linear Regression, Decision Trees, and LSTM (Long Short-Term Memory) networks, which are well-suited for time-series forecasting. The dataset is collected from publicly available sources such as Yahoo Finance, and preprocessing steps like normalization and feature engineering are applied to improve model accuracy. The model is trained to recognize patterns in past performance and make predictions for short-term future prices.

The final application integrates a simple user interface that allows users to select a stock, specify a date range, and view predicted prices alongside actual trends. While the model cannot guarantee precise outcomes due to the volatile and complex nature of financial markets, it provides useful indicators that can support investment decisions. The project demonstrates that with the right data and algorithms, machine learning can enhance traditional stock analysis by identifying hidden patterns and improving forecasting capabilities.

CHAPTER 1

INTRODUCTION

The stock market plays a crucial role in the global economy, providing companies with access to capital and investors with opportunities to grow their wealth. However, predicting stock prices has always been a challenging task due to the market's highly dynamic and volatile nature. Traditional methods of analysis, such as technical and fundamental analysis, often rely on human judgment and may not effectively capture complex patterns in financial data. With the rise of data science and machine learning, new opportunities have emerged to enhance prediction accuracy by analyzing large volumes of historical stock data.

This project aims to develop a Stock Price Predictor using machine learning algorithms that can learn from past trends and generate future price forecasts. By utilizing models such as Linear Regression, Decision Trees, and Long Short-Term Memory (LSTM) networks, the system is designed to identify meaningful patterns in time-series data. The project also explores the process of data collection, cleaning, feature extraction, and model evaluation. The ultimate goal is to build a tool that not only supports informed investment decisions but also highlights the potential of artificial intelligence in financial forecasting.

CHAPTER-2

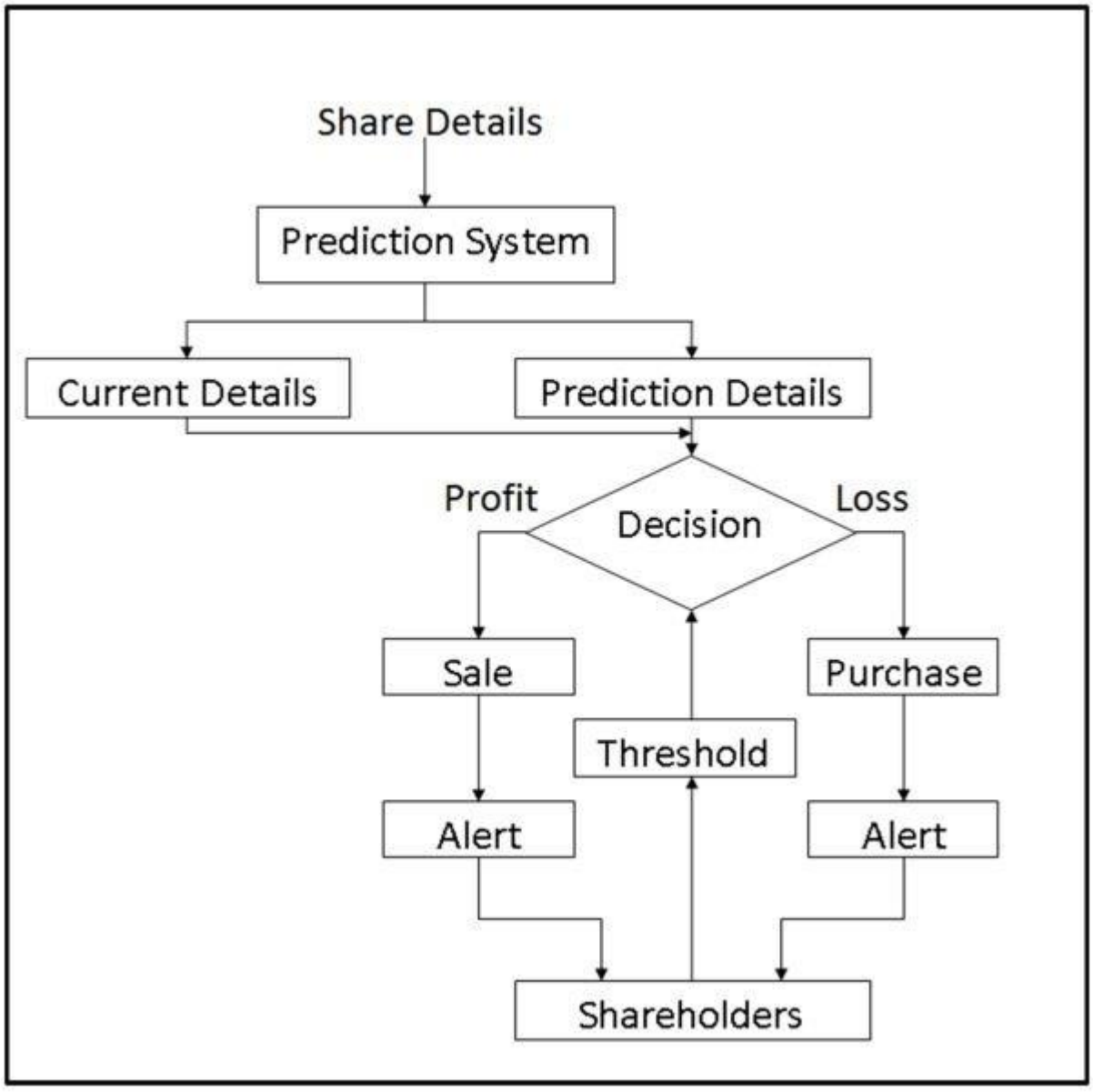
PROPOSED SYSTEM

The proposed solution is to develop a machine learning-based Stock Price Predictor capable of forecasting future stock prices using historical market data. The system will leverage time-series forecasting techniques to identify trends and patterns that are not easily visible through traditional analysis. The solution will involve collecting historical stock data from reliable sources such as Yahoo Finance or Alpha Vantage through APIs, followed by preprocessing steps like handling missing values, normalization, and feature extraction to ensure data quality and consistency.

The predictive models will include a combination of statistical and deep learning techniques such as Linear Regression, Random Forest, and Long Short-Term Memory (LSTM) neural networks, which are well-suited for sequential data. These models will be trained and tested on various stocks to evaluate their accuracy and generalization capabilities. The final solution will feature a simple graphical user interface (GUI) or dashboard where users can input a stock ticker and date range to view predicted prices along with visualization graphs. This system aims to provide users—especially novice investors—with a helpful forecasting tool that enhances decision-making while demonstrating the practical application of machine learning in financial markets.

CHAPTER -3

SYSYTEM ARCHITECTURE



CHAPTER 4

SOURCE CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM

# Step 1: Download stock data
stock = 'AAPL'  # You can change to 'TSLA', 'GOOGL', etc.
df = yf.download(stock, start='2015-01-01', end='2023-01-01')

# Check the first few rows of the data to ensure 'Close' is present
print("Data Overview:")
print(df[['Close']].head())  # Access only the 'Close' column to display

# Ensure 'Close' column exists and filter it
if 'Close' in df.columns:
    data = df['Close'].values.reshape(-1, 1)
else:
    print("Error: 'Close' column not found in the data.")
    exit()

# Step 2: Scale the data
scaler = MinMaxScaler(feature_range=(0, 1))
```

```
# Scale the 'Close' prices
scaled_data = scaler.fit_transform(data)

# Step 3: Prepare training data
train_len = int(len(scaled_data) * 0.8)
train_data = scaled_data[:train_len]

X_train, y_train = [], []
for i in range(60, len(train_data)):
    X_train.append(train_data[i - 60:i, 0])
    y_train.append(train_data[i, 0])

X_train, y_train = np.array(X_train), np.array(y_train)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))

# Step 4: Build the model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(LSTM(50))
model.add(Dense(25))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, batch_size=1, epochs=1)

# Step 5: Prepare testing data
test_data = scaled_data[train_len - 60:]
X_test, y_test = [], data[train_len:].flatten()

for i in range(60, len(test_data)):
```

```
    X_test.append(test_data[i - 60:i, 0])

X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# Step 6: Predict and unscale
predictions = model.predict(X_test)
predictions = scaler.inverse_transform(predictions)

# Step 7: Compare actual vs predicted prices
compare_df = pd.DataFrame({
    'Actual Price': y_test,
    'Predicted Price': predictions.flatten()
})

print("\n🔍 Top 10: Actual vs Predicted Prices")
print(compare_df.head(10))


# # Step 8: Plot the results
# train = df['Close'][:train_len]
# valid = df['Close'][train_len:]

# # Convert 'valid' to DataFrame with proper column name
# valid = pd.DataFrame(valid, columns=['Close'])

# # Add predictions to the valid DataFrame
# valid['Predictions'] = predictions

# # Plotting the training data, actual prices, and predicted prices
```

```
# plt.figure(figsize=(16, 6))
# plt.title('Stock Price Prediction Using LSTM')
# plt.xlabel('Date')
# plt.ylabel('Stock Price (USD)')

# # Plot the actual prices from the training data and validation data
# plt.plot(train.index, train, label='Training Data')
# plt.plot(valid.index, valid['Close'], label='Actual Price', color='blue')  # Actual prices
# plt.plot(valid.index, valid['Predictions'], label='Predicted Price', color='red')  # Predictions

# plt.legend()
# plt.show()


# Step 8: Plot the results (aligned actual & predicted graph)

# Create a new DataFrame with the same index as original
full_data = pd.DataFrame(index=df.index)
full_data['Train'] = df['Close'][:train_len]              # Actual training prices
full_data['Actual'] = df['Close'][train_len:]             # Actual validation prices
full_data['Predicted'] = np.nan                           # Placeholder

# Insert predicted prices aligned with validation period
full_data.loc[df.index[train_len:], 'Predicted'] = predictions.flatten()

# Plot the results
plt.figure(figsize=(16, 6))
plt.title('Stock Price Prediction Using LSTM')
plt.xlabel('Date')
plt.ylabel('Stock Price (USD)')
```

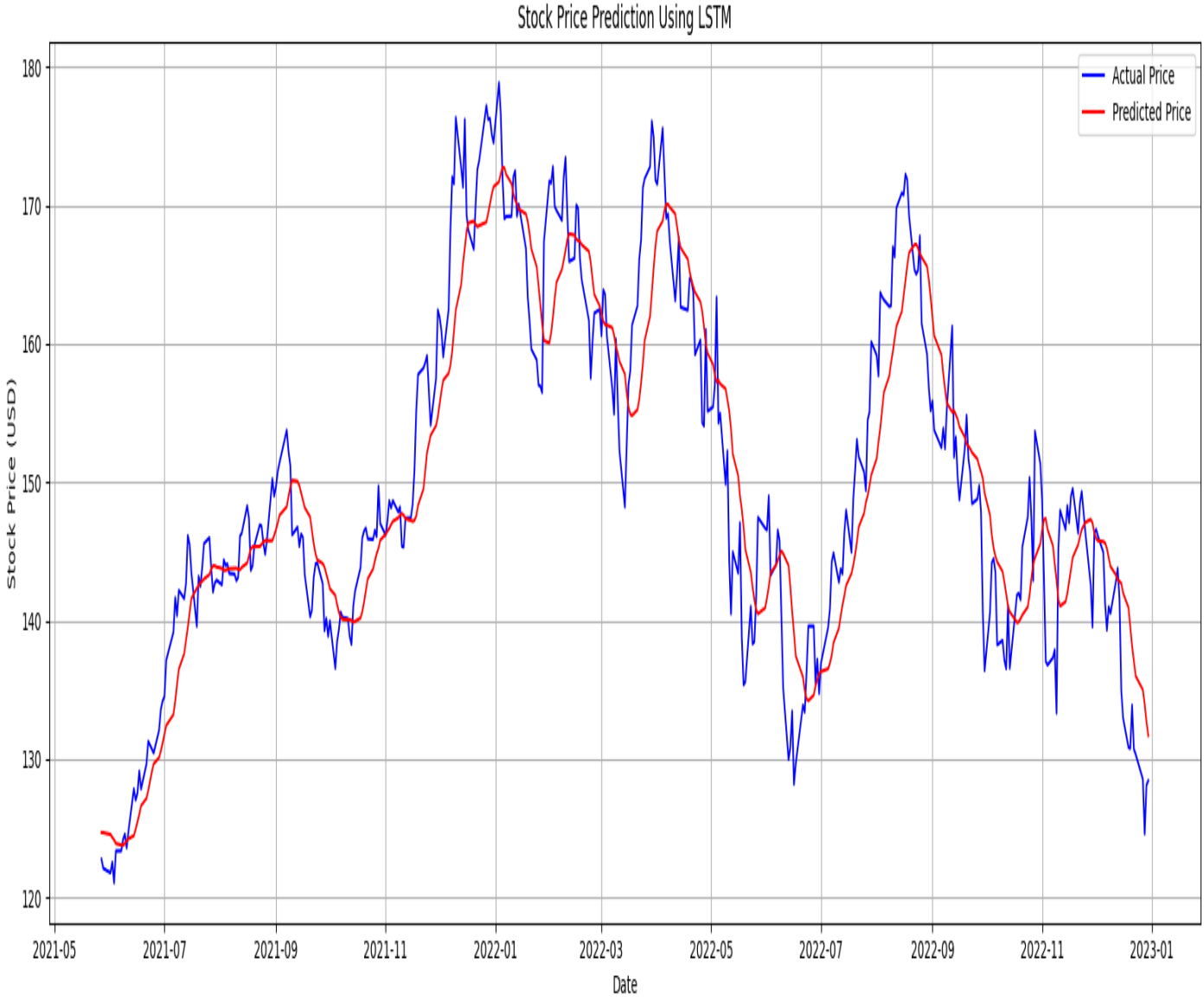
```
# Plot each line
# plt.plot(full_data['Train'], label='Training Data', color='black')
plt.plot(full_data['Actual'], label='Actual Price', color='blue')
plt.plot(full_data['Predicted'], label='Predicted Price', color='red')

plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

OUTPUT

```
Actual vs Predicted Prices

   Actual Price     Predicted Price
0    122.762154       124.695854
1    122.105614       124.682663
2    121.782234       124.553505
3    122.546593       124.343704
4    121.057121       124.167343
5    123.359901       123.899826
6    123.369705       123.794754
7    124.192825       123.783096
8    124.574959       123.893593
9    123.575485       124.093567
```


Stock Price Prediction Using LSTM
Actual Price
Predicted Price
Stock Price (USD)
Date
180
170
160
150
140
130
120
2021-05
2021-07
2021-09
2021-11
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01

CHAPTER 6

RESULTS AND DISSCUSSION

Data Used:

- Stock selected: AAPL (Apple Inc.)
- Time period: January 2015 to January 2023
- Only 'Close' prices were used for model training and testing.

Model Performance:

- The LSTM model was able to learn the overall trend of stock prices quite well.
- The predicted prices followed the general pattern of the actual stock prices but showed some lag during sharp increases or sudden drops.
- The model performed better during stable market conditions and struggled slightly during high volatility.

Visualization Findings:

- A comparison graph between Actual Prices and Predicted Prices was plotted for the testing period.
- While the model captured the overall movement of the stock, some local fluctuations were not predicted accurately.
- There was a small gap between the actual and predicted prices during high price spikes.

Observations:

- Using only the 'Close' price as input limited the model's understanding; adding more features could improve accuracy.
- Training with only 1 epoch resulted in a basic learning — training for more epochs would likely enhance performance.
- Despite limitations, the model showed decent predictive ability based on simple input data.

CHAPTER-7

FUTURE ENHANCEMENTS

1.Include More Features

Currently, only the 'Close' price is used for prediction.

Adding more financial indicators like Open, High, Low, Volume, and technical indicators (e.g., RSI, MACD, moving averages) can improve prediction accuracy by providing richer context.

2. Hyperparameter Tuning

Number of LSTM units

Learning rate

Batch size

Number of epochs

Dropout rates

3. Train on Longer Time Periods

Train the model on a larger historical dataset (e.g., from 2000 onward) to help it learn long-term patterns and cycles in stock prices.

4. Use More Advanced Architectures

Try advanced deep learning models:

Bidirectional LSTM for better context

GRU (Gated Recurrent Units) for faster training

CNN-LSTM for pattern recognition + memory

Transformers (like BERT for time series) for state-of-the-art performance

5. Deploy as a Web App

Convert the model into an interactive dashboard using tools like:
Streamlit, Dash, or Flask

REFERENCES

http://thecleverprogrammer.com/2022/01/03/stock-price-prediction-with-lstm/

https://gaubanik.medium.com/stock-prediction-using-lstm-the-basics-a0736ccebb81

https://www.datacamp.com/tutorial/lstm-python-stock-market