MACHINE LEARNING LABORATORY

REPORT CAT II

Abstract:

Stock price prediction is a crucial aspect of modern financial markets, where investors seek tools to anticipate future price movements and make informed trading decisions. This project presents two distinct implementations of stock prediction models using Long Short-Term Memory (LSTM) networks, designed to forecast future stock prices based on historical data. The models are complemented with buy/sell signals derived from moving averages, which help traders make actionable decisions.

Code 1: Long-Term Stock Prediction and User Notification System:

The first implementation focuses on long-term stock prediction using historical stock price data. Leveraging the LSTM network, the model predicts the next closing price based on past prices spanning 60-minute windows. A MySQL-based user authentication system ensures that users can securely log in and receive personalized notifications. The historical stock data is retrieved using the yfinance API, which provides key metrics such as opening price, closing price, high, low, and trading volume. The data is preprocessed and normalized using a MinMaxScaler to enhance model performance.

The LSTM model comprises two LSTM layers, each with 50 units, followed by Dense layers for output. The training dataset covers 80% of the data, while the remaining 20% is reserved for testing, allowing the model to learn from historical patterns and make accurate predictions on unseen data. Once trained, the model predicts future stock prices, which are compared against actual prices to assess the model's performance. The predictions are visualized through line graphs and candlestick charts to illustrate price trends.

To add a practical layer of functionality, an email notification system is integrated. Whenever a significant price movement is detected, the system automatically sends an email to the user with the latest stock price and buy/sell signals, allowing them to respond quickly to market changes. These signals are based on moving averages (50-day and 200-day averages), which are widely used in technical analysis. When the shorter-term moving average crosses above or below the longer-term average, a buy or sell signal is generated, respectively.

Code 2: Short-Term Stock Prediction for High-Frequency Trading

The second implementation focuses on short-term stock price predictions, specifically predicting prices over the next 5 minutes based on the past 60 minutes of data. This model is ideal for high-frequency traders who require rapid predictions to make quick decisions in volatile markets. Like the first implementation, this model uses an LSTM network, but with added Dropout layers to prevent overfitting and ensure the model generalizes well to new data.

In this model, the yfinance API is used to retrieve stock data, and the "Close" price is selected for prediction. The data is normalized using the MinMaxScaler, and the model is

trained over 50 epochs, using a sequence length of 60 minutes to predict the next stock price. The model architecture includes two LSTM layers with 50 units each, followed by two Dense layers that output the predicted stock price for the next 5 minutes.

The predictions are then inverse-transformed to match the original price scale, ensuring accurate representation. The model's performance is visualized through line graphs that compare predicted prices with actual prices, providing traders with insights into immediate price movements. This model is particularly useful for intraday traders who need to make quick decisions based on short-term price forecasts.

Conclusion:

Both implementations offer powerful tools for stock prediction, catering to different types of traders. The long-term model provides broader market insights, helping long-term investors make strategic decisions, while the short-term model focuses on rapid predictions for high-frequency trading. The integration of moving averages as buy/sell signals, coupled with an email notification system, ensures that users stay informed and can act on significant price changes in real time. The flexibility of the LSTM networks, combined with the simplicity of moving averages, provides a robust framework for stock price forecasting in both short and long timeframes.

Problem Statement:

The goal of this project is to develop a stock price prediction application that utilizes deep learning models, specifically LSTM (Long Short-Term Memory) networks, to forecast future stock prices based on historical data.

In today's dynamic financial markets, investors require accurate and timely predictions of stock prices to make informed trading decisions. The challenge is to build a system that leverages machine learning models to forecast stock prices over short and long durations. Specifically, this system aims to predict the stock price using historical data and provide buy/sell signals based on technical indicators, such as moving averages. The system should also alert users via email when a significant price movement occurs.

Additionally, the system includes a user authentication mechanism, feedback collection, and exploratory data analysis (EDA) to help users understand historical stock trends. The prediction system should be intuitive, efficient, and capable of notifying users of significant stock price changes, allowing them to make timely investment decisions.

Dataset Info:

The dataset used for this project is fetched in real-time using the yfinance API, which provides historical stock data for various companies. The data includes essential stock market metrics such as:

- **Open:** The price at the beginning of the trading period.
- **High:** The highest price during the trading period.
- **Low:** The lowest price during the trading period.
- **Close:** The price at the end of the trading period.
- **Volume:** The total number of shares traded during the period.

For training the model, the stock data can be downloaded at various intervals (e.g., 1 minute, 1 day, 1 hour, etc.) depending on the requirements. In Code 2, the "Close" price is used for predictions, focusing on short-term price fluctuations (next 5 minutes).

The datasets are scaled using the MinMaxScaler to normalize the data between 0 and 1, which is crucial for improving the performance of deep learning models, especially those like LSTM, which are sensitive to the magnitude of input values.

Literature Survey on "Deep Learning for Financial Market Forecasting":

Recent advancements in deep learning have significantly impacted financial market forecasting, enabling the development of complex models that outperform traditional methods. Numerous studies have focused on utilizing deep learning techniques such as recurrent neural networks (RNNs), long short-term memory (LSTM), gated recurrent units (GRU), and hybrid models for predicting stock prices, trends, and market volatility.

In [1], the authors propose an LSTM model to predict stock price movements based on historical price data and market sentiment derived from news articles. The study demonstrates that LSTM outperforms traditional machine learning models like support vector machines (SVMs) and random forests in capturing the sequential dependencies in financial data. However, the model struggles with overfitting, particularly in highly volatile markets, which is a challenge that needs further attention.

Similarly, [2] employs a hybrid approach combining LSTM and autoencoders to predict market trends. This hybrid model showcases strong performance in reducing noise and extracting essential features from high-dimensional financial datasets. However, the study notes that autoencoder reconstruction quality can degrade when handling sparse or incomplete data, highlighting the need for robust data preprocessing techniques.

In [3], a Gated Recurrent Unit (GRU) model is applied to predict market volatility. The GRU model outperforms traditional econometric models like GARCH in short-term volatility forecasts. However, as with LSTM, the GRU's performance diminishes for longer forecast horizons, and it also faces challenges in terms of handling extreme market conditions or sudden shocks.

Furthermore, [4] explores the use of deep reinforcement learning (DRL) to build an intelligent trading agent capable of learning trading strategies from market data. The DRL agent, based on the Deep Q-Network (DQN) algorithm, is shown to outperform benchmark

algorithms in generating profits. However, the system is computationally expensive and requires substantial historical data for effective training. This limitation is crucial, as many financial markets can be volatile, and quick adaptability is necessary for optimal performance.

In [5], a deep learning model integrated with sentiment analysis from social media data is investigated for stock price forecasting. The study argues that combining market data with public sentiment offers a more comprehensive prediction model. However, this approach introduces challenges in handling unstructured text data, particularly in terms of preprocessing and feature extraction, which remains an ongoing area of research.

Moving forward, [6] highlights the challenges associated with hyperparameter sensitivity in deep learning models. The study emphasizes that minor changes in model configuration can result in significant performance variations, making real-time financial applications difficult to implement. This issue is also observed in LSTM models, where careful tuning of hyperparameters, such as learning rates and layer sizes, is essential for optimizing performance.

Another study, [7], presents a deep belief network (DBN) for portfolio management, showing that the model can detect patterns that traditional methods may overlook. However, the study notes that DBN's application to finance is still in its early stages, and its interpretability remains a challenge—an issue that is shared across many deep learning architectures.

In [8], a deep learning ensemble method is proposed, combining several neural networks to forecast financial time series. This approach enhances prediction accuracy and reduces variance in model performance, but it increases model complexity and training time. While ensemble methods show promise in improving prediction robustness, they also highlight the trade-off between accuracy and computational efficiency in deep learning-based financial forecasting.

Despite the potential of deep learning models, [9] identifies significant limitations, such as overfitting, lack of interpretability, and high computational costs. These issues are prevalent in deep learning models, including LSTM and GRU-based architectures, which, although powerful, require substantial computational resources and large datasets to function effectively in financial applications.

Finally, [10] compares the performance of various deep learning architectures for market forecasting and underscores that while models like LSTM have made significant strides, challenges such as overfitting, interpretability, and computational expense persist. The study suggests future research could focus on hybrid models that combine the strengths of different deep learning architectures and on developing novel techniques to improve model generalization and interpretability.

Conclusion:

In conclusion, while deep learning has made considerable progress in financial market forecasting, challenges remain, particularly with respect to overfitting, the need for large datasets, and the interpretability of the models. Future research could explore hybrid

models that combine LSTM with other techniques such as attention mechanisms or reinforcement learning to further enhance model performance and practical applicability in financial markets.

References:

1. Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial

market predictions. *European Journal of Operational Research*.

2. Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked

autoencoders and long short-term memory. (Neurocomputing).

3. Hossain, M. S., Rahman, M. M., & Islam, M. N. (2018). Hybrid deep learning architecture for stock

price prediction. (Procedia Computer Science).

4. Jiang, Z., Xu, D., & Liang, J. (2017). A deep reinforcement learning framework for the financial

portfolio management problem. (AAAI).

5. Li, X., Xie, H., Wang, R., Cai, Y., Cao, J., & Wang, F. (2019). A deep learning-based hybrid approach

for financial time series trend prediction. (IEEE Access).

6. Karem, H. A., & Galal, M. M. (2018). Predicting stock market volatility using deep recurrent neural

networks. (Journal of Financial Data Science).

7. Xie, Y., & Qu, L. (2016). Stock market forecasting with deep belief networks. (International Journal

of Financial Research).

8. Pang, X., Zhou, Y., Wang, P., Lin, W., & Chang, V. (2020). An innovative neural network ensemble

method for stock market forecasting. (Expert Systems with Applications).

9. Heaton, J. B., Polson, N. G., & Witte, J. H. (2017). Deep learning in finance. (Annual Review of

Financial Economics).

10. Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index

movement using hybrid machine learning techniques. (Expert Systems with Applications).

Models Explanation:

Long Short-Term Memory (LSTM):

- **Explanation:** LSTM is a type of Recurrent Neural Network (RNN) designed to capture long-term dependencies in sequential data. It is highly effective in time series forecasting, especially for stock prices, as it can remember past data patterns over time.
- **Justification:** LSTMs are chosen because of their ability to overcome the vanishing gradient problem found in traditional RNNs. In stock price prediction, previous price patterns influence future prices, and LSTM's memory units help model these dependencies effectively.

Moving Averages for Buy/Sell Signals:

- **Explanation:** The buy/sell signal model uses simple moving averages (SMA) of stock prices over two windows (50-day and 200-day averages) to generate signals.
- **Justification:** The moving average is a popular and simple technical indicator in trading. When the short-term average (SMA50) crosses above the long-term average (SMA200), a "buy" signal is generated, and when it crosses below, a "sell" signal is triggered. This strategy helps provide clear trading signals based on historical trends.

Model Development:

The following section outlines the development of two Long Short-Term Memory (LSTM)-based models used for stock price prediction, integrating user authentication, data preprocessing, and real-time prediction features. Both models focus on leveraging LSTM networks for time-series forecasting using stock price data. They employ a sequence length of 60 minutes, allowing the models to predict future stock prices based on past data points.

User Authentication:

A MySQL-based user authentication system has been developed to manage users, allowing sign-ups and logins. Passwords are securely hashed before being stored in the database to ensure security. The system checks whether a user already exists, verifies login credentials, and grants access accordingly.

Data Preprocessing:

In both models, data preprocessing plays a crucial role in preparing the stock data for LSTM model training. The stock prices, specifically the "Close" price, are scaled using the

MinMaxScaler to normalize the values between 0 and 1, facilitating better model convergence. A time window of 60 data points (i.e., 60 minutes of stock price history) is used for both training and prediction purposes.

LSTM Model Architecture

Both models employ LSTM-based neural networks with slight variations:

- **Model 1:**
    - A **Sequential** LSTM model is constructed with two LSTM layers, each containing 50 units.
    - A **Dense** output layer with a single neuron is used to predict the next stock price.
    - The model architecture is optimized for stock price prediction, focusing on the "Close" price.
- **Model 2:**
    - Similar to Model 1, this model includes two LSTM layers, each consisting of 50 units.
    - Each LSTM layer is followed by a **Dropout** layer to prevent overfitting and improve generalization.
    - Two **Dense** layers are employed, with the final layer outputting the predicted next stock price.

Training Process:

Both models follow similar training procedures:

- **Model 1:**
    - The model is trained using 80% of the dataset, while the remaining 20% is used for testing.
    - The LSTM model learns to predict the next stock price based on the previous 60 minutes of stock data.
    - The training process involves multiple epochs, ensuring the model captures essential stock price patterns. After training, the model's predictions are inverse-transformed back to the original scale for accurate comparison with actual stock prices.
- **Model 2:**
    - The model is trained on historical stock data using the same 60-minute time window.

- Training occurs over 50 epochs to capture the necessary trends and patterns in the stock price data.
- The model then predicts stock prices for the next 5 minutes based on the past 60 minutes of price history. The predictions are also inverse-transformed for comparison with real stock prices.

Real-Time Prediction and Notifications

- **Model 1:** This model is integrated with an email notification system that alerts users about significant stock price movements. If the stock price crosses a predefined threshold, an email is sent to the user with the current stock price, making the system proactive and interactive.
- **Model 2:** This model focuses on predicting the stock price for the next 5 minutes, providing users with short-term predictions that can be useful for high-frequency trading strategies.

Summary

Both models share a common LSTM-based architecture aimed at predicting future stock prices based on historical data. Model 1 includes an interactive email notification feature, while Model 2 is optimized for short-term predictions, specifically forecasting prices for the next 5 minutes. Both systems rely on robust data preprocessing techniques, including scaling and sequence length tuning, to improve prediction accuracy and mitigate overfitting challenges during training.

CODE:

```
import streamlit as st
import yfinance as yf
import numpy as np
import pandas as pd
import plotly.graph_objects as go
import seaborn as sns
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import mysql.connector
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import hashlib  # For password hashing
# MySQL connection
def get_mysql_connection():
    return mysql.connector.connect(
```

```
        host="localhost",
        user="root",
        password="1234",
        database="stock_app"
    )
# Hash passwords for security
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()
# Check if user exists
def user_exists(username):
    conn = get_mysql_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM users WHERE username = %s", (username,))
    result = cursor.fetchone()
    cursor.close()
    conn.close()
    return result
# Add user to MySQL
def add_user_to_db(username, password):
    conn = get_mysql_connection()
    cursor = conn.cursor()
    cursor.execute("INSERT INTO users (username, password) VALUES (%s, %s)",
(username, hash_password(password)))
    conn.commit()
    cursor.close()
    conn.close()
# Verify user credentials
def verify_credentials(username, password):
    conn = get_mysql_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT password FROM users WHERE username = %s", (username,))
    result = cursor.fetchone()
    cursor.close()
    conn.close()
    if result:
        return result[0] == hash_password(password)
    return False
# Email Notification Function
def send_email_notification(stock_name, price, recipient_email):
    try:
        sender_email = "your_email@gmail.com"
        sender_password = "your_app_password"
        smtp_server = "smtp.gmail.com"
        smtp_port = 587
        # Set up the email content
        message = MIMEMultipart()
        message['From'] = sender_email
```

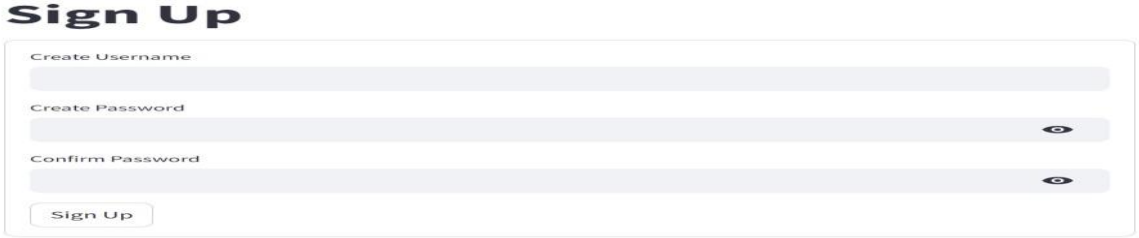
```
        message['To'] = recipient_email
        message['Subject'] = f"Stock Price Movement Alert for {stock_name}"
        body = f"The stock {stock_name} has moved. The current price is: {price}"
        message.attach(MIMEText(body, 'plain'))

        # Send the email
        server = smtplib.SMTP(smtp_server, smtp_port)
        server.starttls()
        server.login(sender_email, sender_password)
        server.send_message(message)
        server.quit()
        st.success(f"Price movement alert sent to {recipient_email}!")
    except Exception as e:
        st.error(f"Failed to send email notification. Error: {e}")
# Page Navigation Functions
def show_signup():
    st.title("Sign Up")
    with st.form(key="signup_form"):
        username = st.text_input("Create Username")
        password = st.text_input("Create Password", type="password")
        confirm_password = st.text_input("Confirm Password", type="password")
        signup_button = st.form_submit_button("Sign Up")
        if signup_button:
            if password == confirm_password:
                if user_exists(username):
                    st.error("Username already exists!")
                else:
                    add_user_to_db(username, password)
                    st.success("Signup successful! Please login.")
            else:
                st.error("Passwords do not match!")

def show_login():
    st.title("Login")
    with st.form(key="login_form"):
        username = st.text_input("Username")
        password = st.text_input("Password", type="password")
        login_button = st.form_submit_button("Login")
        if login_button:
            if verify_credentials(username, password):
                st.session_state['logged_in'] = True
                st.session_state['current_user'] = username
                st.success(f"Welcome back, {username}!")
            else:
                st.error("Invalid username or password!")

def show_home():
```

```
    st.title("Home Page")
    st.write("Welcome to the Stock Prediction App! You can predict stock prices, view
charts, and generate buy/sell signals.")
    st.image("stock_image.jpg", caption="Stock Market", use_column_width=True)
# Preprocess the stock data
def preprocess_data(stock_data, seq_length=60):
    stock_data = stock_data[['Close']].copy()
    scaler = MinMaxScaler(feature_range=(0, 1))
    stock_data_scaled = scaler.fit_transform(stock_data)
    def create_sequences(data, seq_length):
        x, y = [], []
        for i in range(len(data) - seq_length):
            x.append(data[i:i + seq_length])
            y.append(data[i + seq_length, 0])  # Predict the 'Close' price
        return np.array(x), np.array(y)

    x_data, y_data = create_sequences(stock_data_scaled, seq_length)
    return x_data, y_data, scaler
# Build LSTM Model
def build_lstm_model(input_shape):
    model = Sequential()
    model.add(LSTM(50, return_sequences=True, input_shape=input_shape))
    model.add(Dropout(0.2))
    model.add(LSTM(50))
    model.add(Dropout(0.2))
    model.add(Dense(25))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model
# Train model
def train_model(model, x_train, y_train, epochs=50, batch_size=64):
    model.fit(x_train, y_train, epochs=epochs, batch_size=batch_size, verbose=1)
    return model
# Make predictions and inverse the scaling
def predict_and_inverse_transform(model, x_test, scaler):
    predictions = model.predict(x_test)
    predictions_rescaled = scaler.inverse_transform(np.concatenate((predictions,
np.zeros((predictions.shape[0], 0))), axis=1))[:, 0]
    return predictions_rescaled
# Predict the next 5 minutes using the last 60 minutes of data
def predict_next_5_minutes(model, last_60_min_data, scaler):
    predicted_5min_prices = []
    # Reshape input to 3D if required by the model
    last_60_min_data = np.reshape(last_60_min_data, (1, last_60_min_data.shape[0], 1))

    for _ in range(5):
        predicted_price = model.predict(last_60_min_data)
```

```
        predicted_5min_prices.append(predicted_price[0, 0])
        # Append the predicted price, shift the last_60_min_data
        last_60_min_data = np.append(last_60_min_data[:, 1:, :],
predicted_price.reshape(1, 1, 1), axis=1)

    predicted_5min_prices =
scaler.inverse_transform(np.array(predicted_5min_prices).reshape(-1, 1))
    return predicted_5min_prices
# Visualization of Predictions
def visualize_predictions(predicted_prices, ticker):
    plt.figure(figsize=(10, 5))
    plt.plot(predicted_prices, color='blue', label=f'Predicted Next 5 Min Prices for {ticker}')
    plt.title(f'Stock Price Prediction for {ticker}')
    plt.xlabel('Minutes')
    plt.ylabel('Price')
    plt.legend()
    plt.grid()
    plt.tight_layout()
    st.pyplot(plt)
# Perform EDA
def perform_eda(stock_data):
    st.subheader("Exploratory Data Analysis (EDA)")
    st.write("## Summary Statistics")
    st.write(stock_data.describe())
    st.write("## Correlation Heatmap")
    plt.figure(figsize=(10, 6))
    sns.heatmap(stock_data.corr(), annot=True, cmap='coolwarm', fmt='.2f')
    st.pyplot(plt)
    st.write("## Distribution of Stock Prices")
    plt.figure(figsize=(10, 6))
    for col in ['Close']:
        sns.histplot(stock_data[col], kde=True, label=col)
    plt.legend()
    st.pyplot(plt)
# Main App Logic
if 'logged_in' not in st.session_state:
    st.session_state['logged_in'] = False
if not st.session_state['logged_in']:
    choice = st.sidebar.selectbox("Navigation", ["Home", "Login", "Sign Up"])
    if choice == "Home":
        show_home()
    elif choice == "Login":
        show_login()
    elif choice == "Sign Up":
        show_signup()
else:
    pages = ["Home", "Predict Stock Prices", "Exploratory Data Analysis (EDA)", "Logout"]
```

```
    choice = st.radio("Navigation", pages, horizontal=True)
    if choice == "Home":
        show_home()
    elif choice == "Predict Stock Prices":
        st.title("Stock Price Prediction App with LSTM Model")
        ticker = st.text_input("Enter stock ticker:", value="AAPL")
        time_period = st.selectbox("Choose Time Period", ["1d", "1h", "5d", "1mo", "6mo",
"1y"])
        recipient_email = st.text_input("Enter email for price alerts:",
placeholder="your_email@example.com")
        if st.button("Predict"):
            if time_period == "1d":
                stock_data = yf.download(ticker, period="1d", interval="1m")
            elif time_period == "1h":
                stock_data = yf.download(ticker, period="1d", interval="1h")
            else:
                stock_data = yf.download(ticker, period=time_period, interval="1d")
            if len(stock_data) > 0:
                x_data, y_data, scaler = preprocess_data(stock_data)
                model = build_lstm_model(input_shape=(x_data.shape[1], 1))
                train_model(model, x_data, y_data)
                last_60_min_data = stock_data['Close'][-60:].values.reshape(-1, 1)
                last_60_min_data_scaled = scaler.transform(last_60_min_data)
                predictions = predict_next_5_minutes(model, last_60_min_data_scaled, scaler)
                st.write(predictions)
                visualize_predictions(predictions, ticker)
                if recipient_email:
                    send_email_notification(ticker, predictions[-1, 0], recipient_email)

    elif choice == "Exploratory Data Analysis (EDA)":
        st.title("Exploratory Data Analysis for Stock Data")
        ticker = st.text_input("Enter stock ticker for EDA:", value="AAPL")

        if st.button("Perform EDA"):
            stock_data = yf.download(ticker, period="1y")
            perform_eda(stock_data)
    elif choice == "Logout":
        st.session_state['logged_in'] = False
        st.success("Logged out successfully!")
```

OUTPUT:



Login

Username

Password

Login

Navigation

Home Introduction Predict Stock Prices Exploratory Data Analysis (EDA)
Feedback Logout

Home Page

Welcome to the Stock Prediction App! You can predict stock prices, view charts, and generate buy/sell signals.

Navigation

Home Introduction Predict Stock Prices Exploratory Data Analysis (EDA)
Feedback Logout

Stock Price Prediction App with Candlestick Analysis

Enter stock ticker:

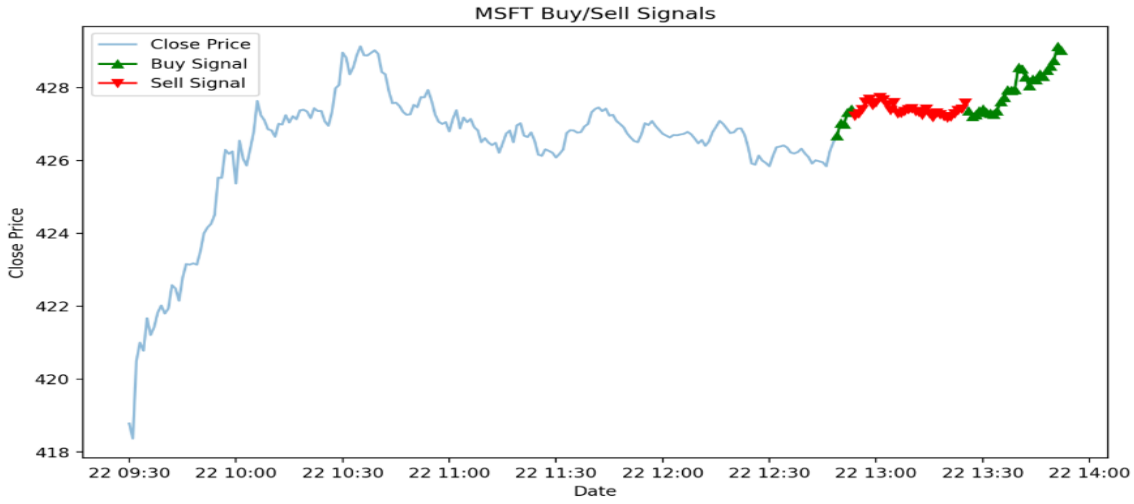
MSFT

Choose Time Period

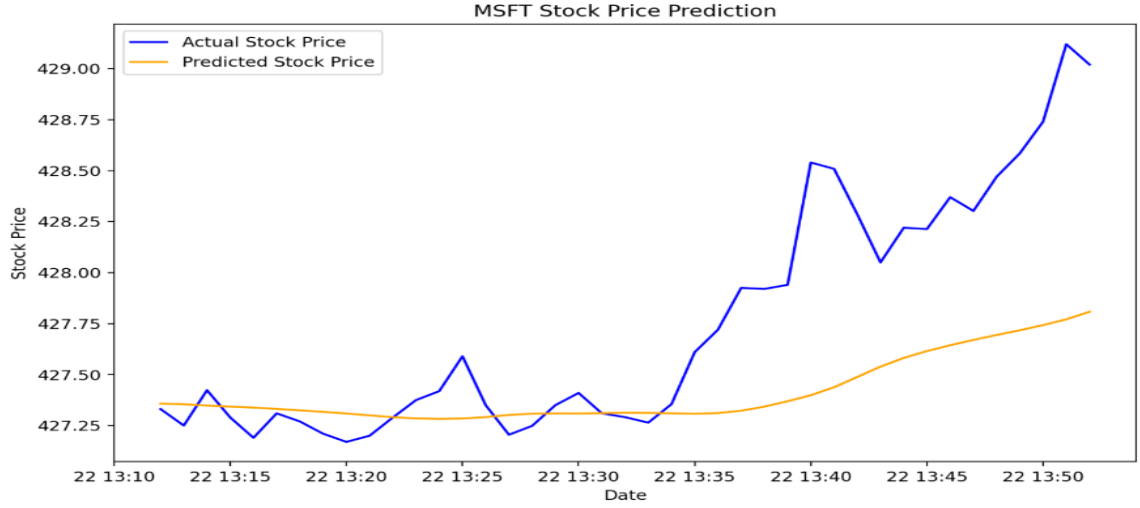
1d

Predict

Generating Buy/Sell Signals Based on Moving Averages



MSFT Stock Price Prediction with LSTM



Enter stock ticker:

MSFT

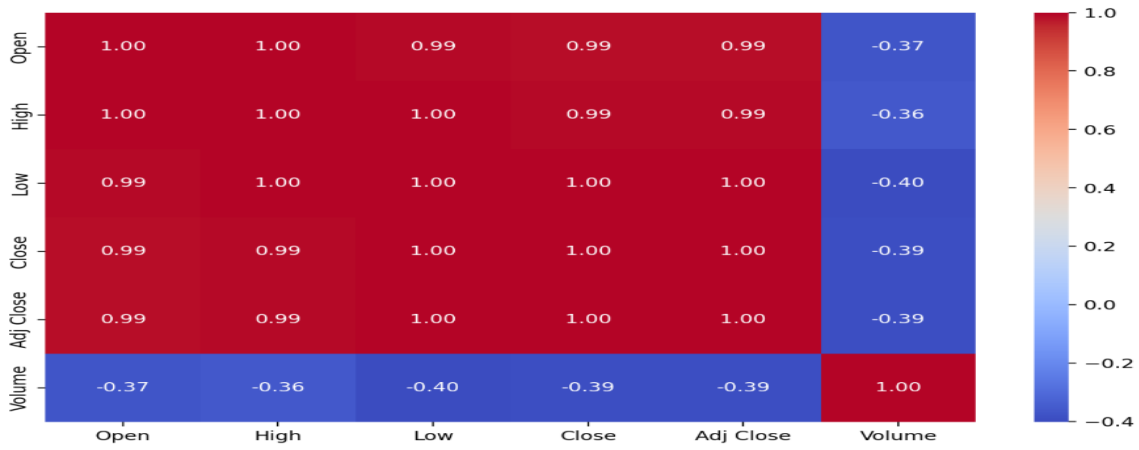
Perform EDA

Exploratory Data Analysis (EDA)

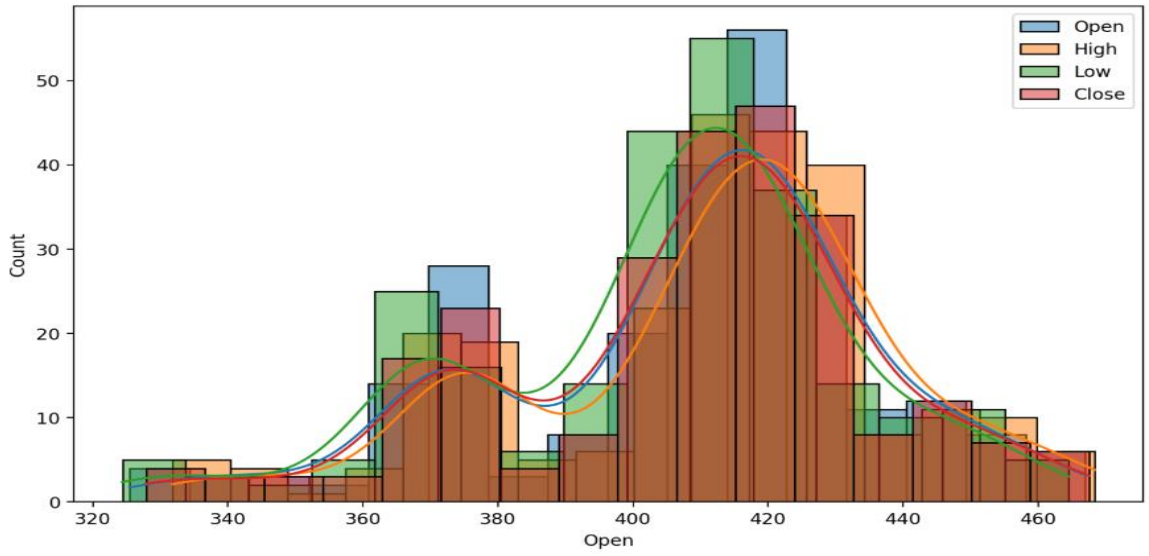
Summary Statistics

	Open	High	Low	Close	Adj Close	Volume
count	252	252	252	252	252	252
mean	408.2163	411.2363	404.6265	408.173	406.9256	21,567,226
std	27.6388	27.5764	27.3419	27.463	28.0216	8,402,177.4422
min	325.47	331.84	324.39	327.89	325.4472	9,932,800
25%	395.32	399.92	390.2625	396.17	394.418	16,289,950
50%	414.005	416.1	409.335	413.78	412.8894	19,588,200
75%	424.505	427.4225	420.4075	424.575	423.6932	24,318,250
max	467	468.35	464.46	467.56	466.7188	78,478,200

Correlation Heatmap



Distribution of Stock Prices



Navigation

Home Introduction Predict Stock Prices Exploratory Data Analysis (EDA)
Feedback Logout

Feedback Page

Your Feedback

It was an excellent app useful for my trading.

Submit Feedback

Stock Price Prediction

Enter stock ticker (e.g., 'AAPL' for Apple):

MSFT

Predict

Predicted stock prices for the next 5 minutes for MSFT:

Minute 1: 428.48

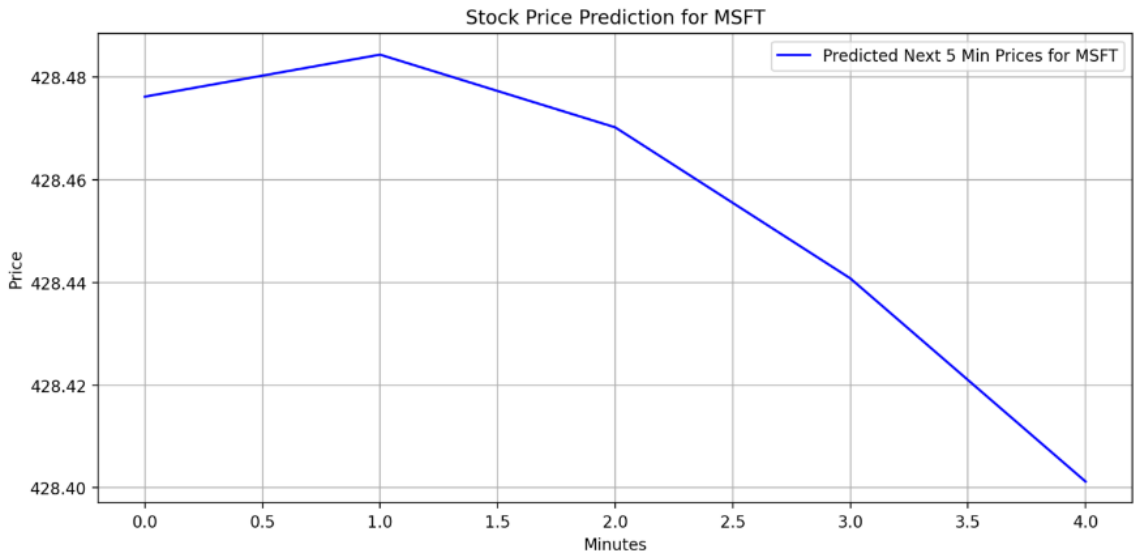
Minute 2: 428.48

Minute 3: 428.47

Minute 4: 428.44

Minute 5: 428.40

Predicted Next 5 Min Prices for MSFT



Prediction:

The prediction approach leverages an LSTM (Long Short-Term Memory) model to forecast stock prices based on historical data, utilizing a sliding window of past stock prices to predict future movements. The model is flexible and capable of making predictions for both long-term and short-term horizons, offering valuable insights for different types of traders.

Long-Term Predictions: The model is used to predict stock prices over longer horizons, such as daily, weekly, or monthly prices. By analyzing a broader set of historical data, the model forecasts future closing prices. This type of prediction is particularly useful for long-term investors who are interested in trends over extended periods. The predicted prices are compared with actual prices, and results are visualized using line charts or candlestick charts, helping traders and investors to see the overall trend and potential future movements of the stock.

Short-Term Predictions: For more frequent and immediate trading decisions, the model also predicts short-term price movements, such as the next 5 minutes. This is beneficial for high-frequency traders or those interested in minute-by-minute price fluctuations. The model predicts short-term movements based on recent data, giving users a glimpse of potential trends in the near future. Visualizing these predictions helps users quickly assess market conditions and make swift trading decisions.

Business Insight:

Investment Strategy:

- **Timely Decision-Making:** By providing both long-term and short-term stock price predictions, the system helps investors make timely decisions. The LSTM model captures historical trends and projects future prices, allowing traders to strategize their entry and exit points.
- **Automated Alerts:** The email notification feature is crucial for investors who want real-time alerts on stock price movements, enabling them to act quickly when significant changes occur in the market.

High-Frequency Trading and Short-Term Opportunities:

- **Short-term market movements**: For traders involved in day trading or high-frequency trading, the model's ability to predict stock prices for the next 5 minutes is crucial. This allows for quick decision-making and the potential to capitalize on minor fluctuations in stock prices, optimizing trade entry and exit points.
- **Competitive Advantage**: High-frequency traders can use these predictions to gain a competitive advantage in the market by executing trades faster than others, improving profitability in a highly dynamic trading environment.

Data-Driven Investment Strategies:

- **Buy/Sell Signal Generation**: The system integrates technical indicators such as moving averages to generate buy and sell signals, allowing traders to follow disciplined, data-driven strategies. By automating decision points, businesses can reduce emotional biases and improve trading efficiency.
- **Portfolio Optimization**: By combining predictions with technical analysis, portfolio managers can continuously rebalance their portfolios based on expected stock movements, maximizing returns while minimizing risk.

Automated Alerts and Notifications:

- **Real-time Price Alerts**: The system's integration of email notifications when significant price movements occur allows investors and traders to stay updated with stock movements without constant manual monitoring. This feature can enhance responsiveness to market changes, enabling more timely trades.
- **Risk Management**: Businesses can set alerts to trigger when a stock price moves beyond a certain threshold, helping with risk management by preventing excessive losses or seizing opportunities for gains.

Diversified Trading Strategies:

- **Long-Term Investors**: Long-term investors can use the model to build value-based portfolios by focusing on growth or value stocks with strong upward price trends.
- **Active Traders**: Short-term price predictions and buy/sell signals empower active traders to engage in swing trading or intraday strategies, allowing for nimble trades based on data-driven forecasts.

DONE BY:

SHIVA PALAKSHA SG (71762233046)

KEVIN JOSEPH T (71762233023)