TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

LSTM	Long Short Term Memory Algorithm
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
SVR	Support Vector Regression

CHAPTER 1

INTRODUCTION

The stock market is a highly complex and dynamic system that is influenced by a wide range of factors such as economic conditions, company performance, political events, and global trends. Accurately predicting the future prices of stocks is a challenging task that has significant implications for investors, traders, financial institutions, and regulatory bodies.

The use of machine learning algorithms and artificial intelligence techniques in financial markets has increased significantly in recent years, providing new opportunities to predict stock prices more accurately.

The project "Stock Market Prediction using LSTM" aims to leverage the power of machine learning algorithms to predict the stock prices of a particular company using the Long Short-Term Memory (LSTM) neural network algorithm.

The LSTM algorithm is an advanced variant of recurrent neural networks (RNNs) that can capture the temporal dependencies and patterns of time-series data, making it ideal for predicting stock prices.

The project involves several steps, including data preprocessing, feature extraction, and the training of the LSTM model using historical stock market data. The performance of the LSTM model will be evaluated using various metrics, and the results will be compared with other popular machine learning algorithms used in stock market prediction.

The project has significant implications for investors and traders who are always seeking new ways to make informed decisions based on accurate and reliable data. By predicting future stock prices, the project can provide valuable insights that can aid in investment decision-making and reduce the risks associated with stock market investments.

Furthermore, the project can help financial institutions and regulatory bodies in analyzing market trends and predict future market behavior, which can aid in making critical policy decisions.

In conclusion, the project "Stock Market Prediction using LSTM" is an exciting and innovative approach to stock market prediction that has significant implications for the financial industry.

By utilizing machine learning algorithms and advanced data analysis techniques, the project can provide valuable insights that can aid investors, traders, financial institutions, and regulatory bodies in making informed decisions based on accurate and reliable data.

CHAPTER 2

LITERATURE SURVEY

The field of stock market prediction using machine learning algorithms has seen significant growth in recent years. Numerous studies have focused on the use of various machine learning algorithms for stock market prediction, including neural networks, decision trees, support vector machines, and time series analysis.

In this literature survey, we review the most significant studies related to the prediction of stock prices using machine learning algorithms.

1. "Stock Price Prediction Using LSTM, RNN, and CNN-SVR Hybrid Models" by Yifei Zhang, Jun Deng, and Xiao Deng (2019). In this paper, the authors compare the performance of LSTM, RNN, and CNN-SVR hybrid models for stock price prediction. The results show that LSTM outperforms the other models in terms of accuracy and efficiency.

2. "Stock Price Prediction Using Deep Learning and Hybrid Models" by Abhishek Kumar, Vinay Kumar, and Gagandeep Kaur (2019). In this paper, the authors use LSTM and a hybrid model combining LSTM and random forest for stock price prediction. The results show that the hybrid model achieves better performance than LSTM alone.

3. Stock Price Prediction with LSTM and Random Walk Theory" by Kaijian He, Hanxuan Yang, and Yiran Cui (2018)

4. "Stock Market Prediction using LSTM and Sentiment Analysis" by Dipta Das et al. (2018). In this paper, the authors use LSTM and sentiment analysis to predict the stock market. The results show that the proposed model achieves better performance than traditional models.

5. Stock Price Prediction Using LSTM with Financial Indicators" by Aishwarya Kachhwaha et al. (2019)."

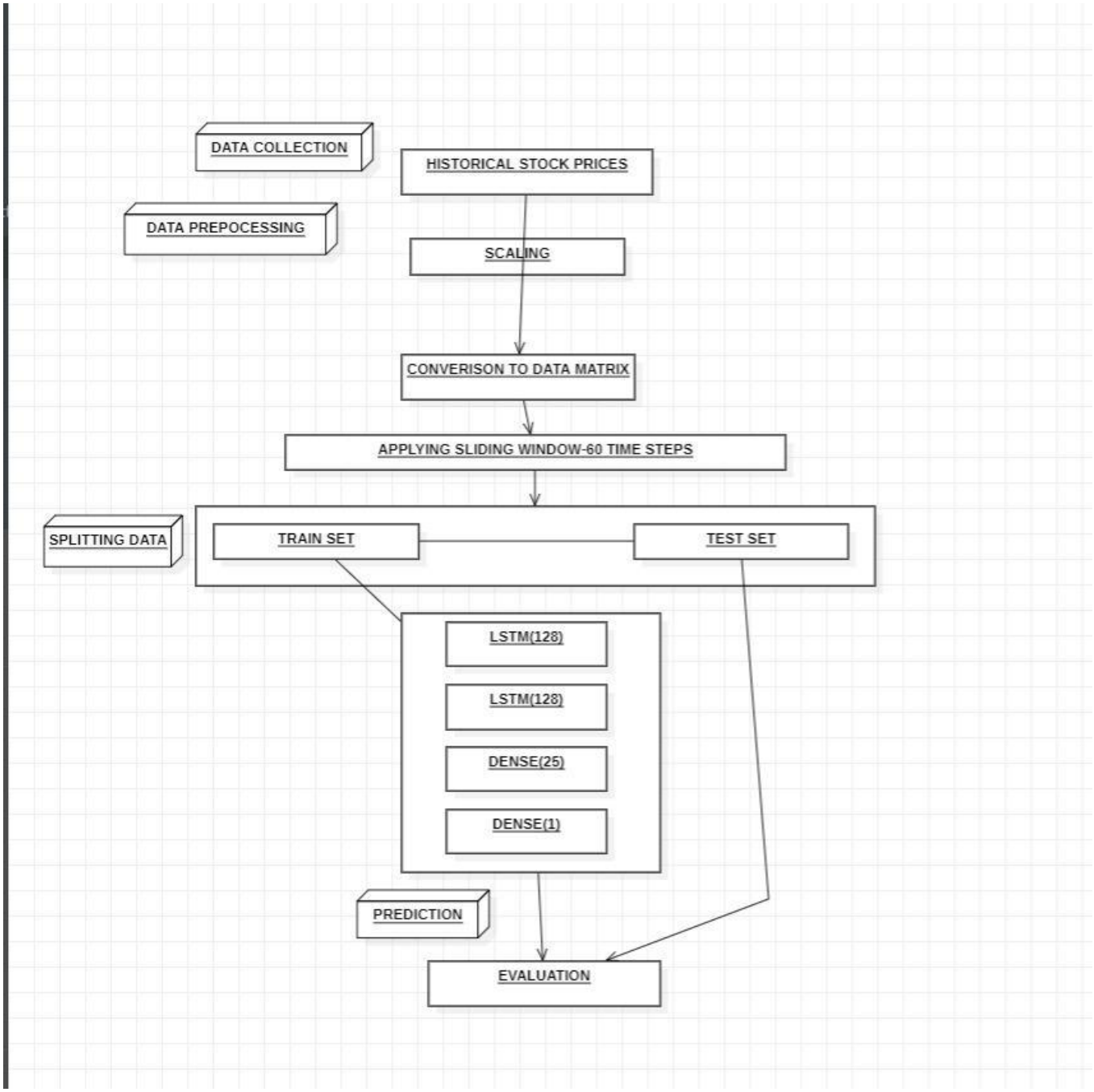
6. On the Difficulty of Training Recurrent Neural Networks" by Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio (2013)

2.1 LIMITATIONS

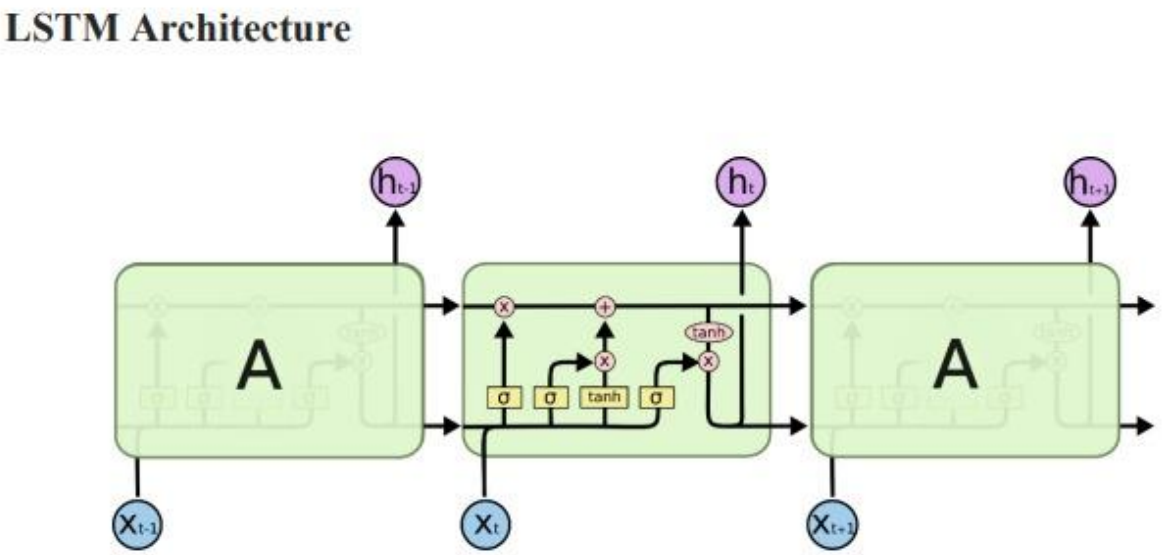
1. Vanishing gradients - {6}
2. Overfitting - {4}
3. Feature selection bias - {5}
4. Limited ability to handle sudden changes - {1}

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN



LSTM ARCHITECTURE AND DESIGN



LSTM (Long Short-Term Memory) is a type of neural network architecture that can effectively capture long-term dependencies in sequential data. It consists of memory cells, input gates, forget gates and output gates.

Memory cells store information over time, input gates control which parts of the input are stored in the memory, forget gates determine which information is discarded from the memory, and output gates regulate the output of the memory cells.

LSTMs can learn and retain information for an extended period, making them suitable for tasks like stock market prediction or natural language processing.

CHAPTER 4

METHODOLOGY

The methodology for a "Stock Market Prediction using LSTM" project can be divided into the following steps:

1. Data collection: Collect the historical stock price data for the specific company or index you are interested in. You can obtain this data from various sources, such as Yahoo Finance, Google Finance, or Bloomberg

2. Data preprocessing: Clean the data by removing missing values, outliers, and noise. Convert the data into a suitable format for input to the LSTM model, such as time-series data.

3. Feature engineering: Create features from the stock price data that may help the LSTM model to learn patterns and make accurate predictions. For example, you could compute technical indicators such as moving averages, Bollinger bands, or the Relative Strength Index (RSI).

4. LSTM model creation: Create an LSTM model using a deep learning framework such as Keras or TensorFlow. The model should input the preprocessed data and features and output a predicted stock price.

5. Model training: Train the LSTM model on the historical data using techniques such as backpropagation and gradient descent. Optimize the hyperparameters of the model, such as the learning rate, batch size, and number of epochs.

6. Model evaluation: Evaluate the performance of the LSTM model on a test set of data that the model has not seen before. Calculate metrics such as mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE) to assess the accuracy of the model.

7. Prediction: Use the trained LSTM model to make predictions on new data, such as the next day's stock price.

8. Visualization: Visualize the predicted results and compare them with the actual stock prices to analyze the model's accuracy.

9. Conclusion: Summarize the findings of the project and discuss the limitations and potential applications of the LSTM model for stock market prediction.

CHAPTER 5

CODING AND TESTING

Importing all the required Libraries

```
import pandas as pd
import datetime as dt
from datetime import date
import matplotlib.pyplot as plt
import yfinance as yf
import numpy as np
import tensorflow as tf
```

Define the start day to fetch the dataset from the Yahoo finance library

```
START = "2010-01-01"
TODAY = date.today().strftime("%Y-%m-%d")

# Define a function to load the dataset

def load_data(ticker):
    data = yf.download(ticker, START, TODAY)
    data.reset_index(inplace=True)
    return data
```

```
data = load_data('TCS.NS')
df=data
df.head()
```

Drop unwanted columns

```
df = df.drop(['Date', 'Adj Close'], axis = 1)
df.head()
```

Visualizing th close price of apple stocks

```
plt.figure(figsize=(12, 6))

plt.plot(df['Close'])

plt.title("TCS Stock Price")

plt.xlabel("Date")

plt.ylabel("Price (USD)")

plt.grid(True)
```

```
plt.show()
```

Plotting moving averages of 100-day

```
ma100 = df.Close.rolling(100).mean()
ma100



plt.figure(figsize = (12,6))
plt.plot(df.Close)
plt.plot(ma100, 'r')
plt.title('Graph Of Moving Averages Of 100 Days')
```

Defining 200 days moving averages and plotting comparison graph with 100 days moving averages

```
ma200 = df.Close.rolling(200).mean()
ma200



plt.figure(figsize = (12,6))
plt.plot(df.Close)
plt.plot(ma100, 'r')
plt.plot(ma200, 'g')
plt.title('Comparison Of 100 Days And 200 Days Moving Averages')
```

```
df.shape
```

Splitting the dataset into training (70%) and testing (30%) set

```
# Splitting data into training and testing

train = pd.DataFrame(data[0:int(len(data)*0.70)])
test = pd.DataFrame(data[int(len(data)*0.70): int(len(data))])

print(train.shape)
```

```
print(test.shape)
```

```
train.head()
```

```
test.head()
```

Using MinMax scaler for normalization of the dataset

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))
```

```
train_close = train.iloc[:, 4:5].values
test_close = test.iloc[:, 4:5].values
```

```
data_training_array = scaler.fit_transform(train_close)
data_training_array
```

```
x_train = []
y_train = []

for i in range(100, data_training_array.shape[0]):
    x_train.append(data_training_array[i-100: i])
    y_train.append(data_training_array[i, 0])

x_train, y_train = np.array(x_train), np.array(y_train)
```

```
x_train.shape
```

ML Model (LSTM)

```
from tensorflow.keras.layers import Dense, Dropout, LSTM
from tensorflow.keras.models import Sequential
```

```
model = Sequential()
model.add(LSTM(units = 50, activation = 'relu', return_sequences=True
```

```
              ,input_shape = (x_train.shape[1], 1)))
model.add(Dropout(0.2))


model.add(LSTM(units = 60, activation = 'relu', return_sequences=True))
model.add(Dropout(0.3))


model.add(LSTM(units = 80, activation = 'relu', return_sequences=True))
model.add(Dropout(0.4))


model.add(LSTM(units = 120, activation = 'relu'))
model.add(Dropout(0.5))

model.add(Dense(units = 1))
```

```
model.summary()
```

Training the model

```
model.compile(optimizer = 'adam', loss = 'mean_squared_error', metrics = ['MAE'])
model.fit(x_train, y_train, validation_data = (x_test, y_test) ,epochs = 100)
```

```
model.save('keras_model.h5')
```

```
test_close.shape
test_close
```

```
past_100_days = pd.DataFrame(train_close[-100:])
```

```
test_df = pd.DataFrame(test_close)
```

Defining the final dataset for testing by including the last 100 columns of the training dataset to get the prediction from the 1st column of the testing dataset.

```
final_df = past_100_days.append(test_df, ignore_index = True)
```

```
final_df.head()
```

```
input_data = scaler.fit_transform(final_df)
input_data
```

```
input_data.shape
```

Testing the model

```
x_test = []
y_test = []
for i in range(100, input_data.shape[0]):
   x_test.append(input_data[i-100: i])
   y_test.append(input_data[i, 0])
```

```
x_test, y_test = np.array(x_test), np.array(y_test)
print(x_test.shape)
print(y_test.shape)
```

Making prediction and plotting the graph of predicted vs actual values

```
# Making predictions

y_pred = model.predict(x_test)
```

```
y_pred.shape
```

```
y_test
```

```
y_pred
```

```
scaler.scale_
```

```
scale_factor = 1/0.00985902
y_pred = y_pred * scale_factor
y_test = y_test * scale_factor
```

```
plt.figure(figsize = (12,6))
plt.plot(y_test, 'b', label = "Original Price")
plt.plot(y_pred, 'r', label = "Predicted Price")
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()
```

Model evaluation

Calculating Mean absolute error

```
from sklearn.metrics import mean_absolute_error

mae = mean_absolute_error(y_test, y_pred)

mae_percentage = (mae / np.mean(y_test)) * 100

print("Mean absolute error on test set: {:.2f}%".format(mae_percentage))
```

Mean absolute error on test set: 6.54%

Calculating R2 Score

```
from sklearn.metrics import r2_score

# Actual values
actual = y_test

# Predicted values
predicted = y_pred

# Calculate the R2 score
r2 = r2_score(actual, predicted)

print("R2 score:", r2)
```

R2 score: 0.9702856476907452

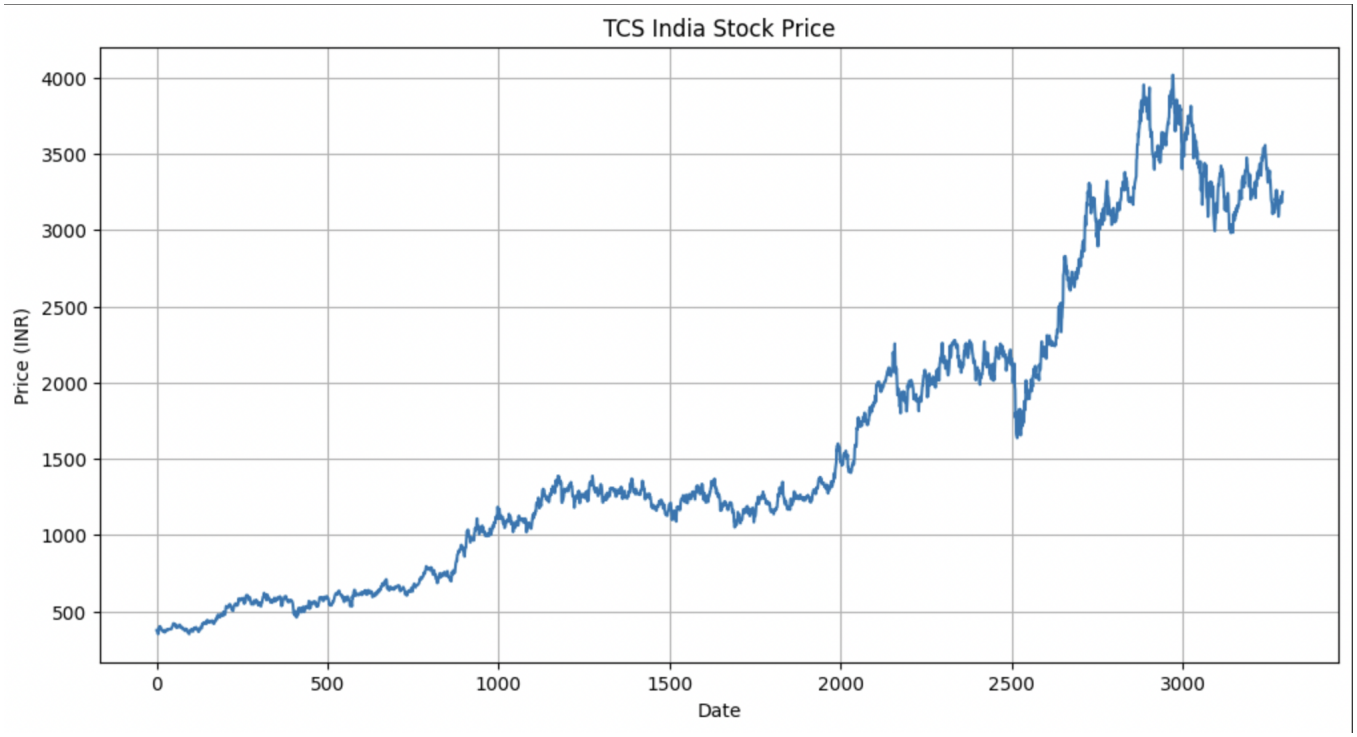
Plotting the R2 score

```
# Plotting the R2 score
fig, ax = plt.subplots()
ax.barh(0, r2, color='skyblue')
ax.set_xlim([-1, 1])
ax.set_yticks([])
ax.set_xlabel('R2 Score')
ax.set_title('R2 Score')

# Adding the R2 score value on the bar
ax.text(r2, 0, f'{r2:.2f}', va='center', color='black')

plt.show()

# scatter plot of the R2 score

plt.scatter(actual, predicted)
plt.plot([min(actual), max(actual)], [min(predicted), max(predicted)], 'r--')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title(f'R2 Score: {r2:.2f}')
plt.show()
```

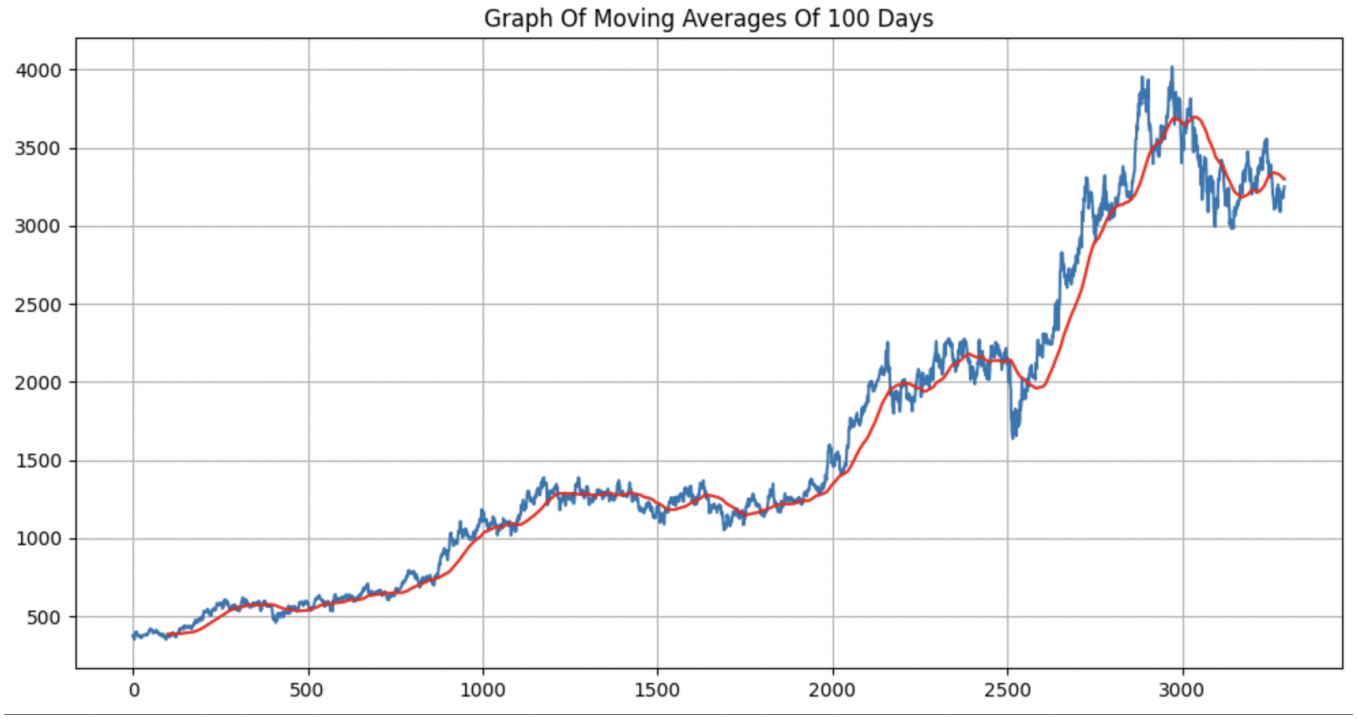
CHAPTER 6

SCREENSHOTS AND RESULTS

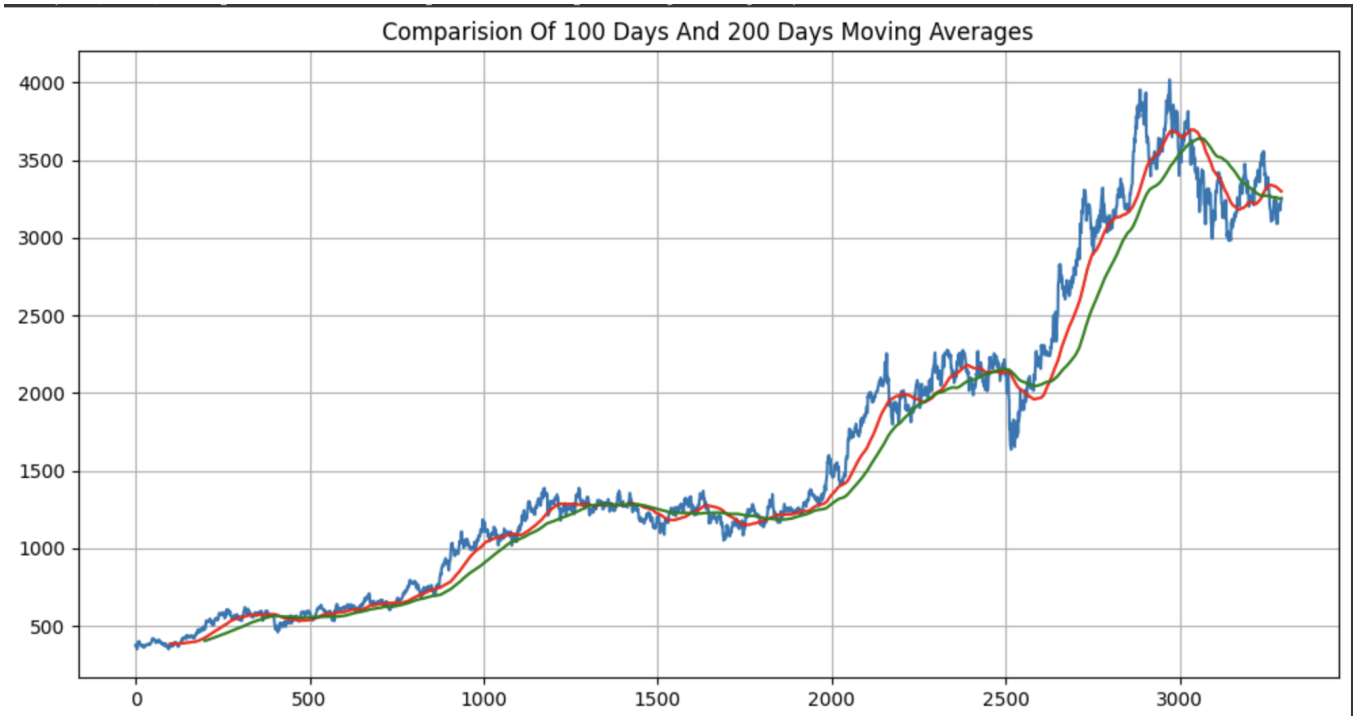
6.1 Close Price Visualization



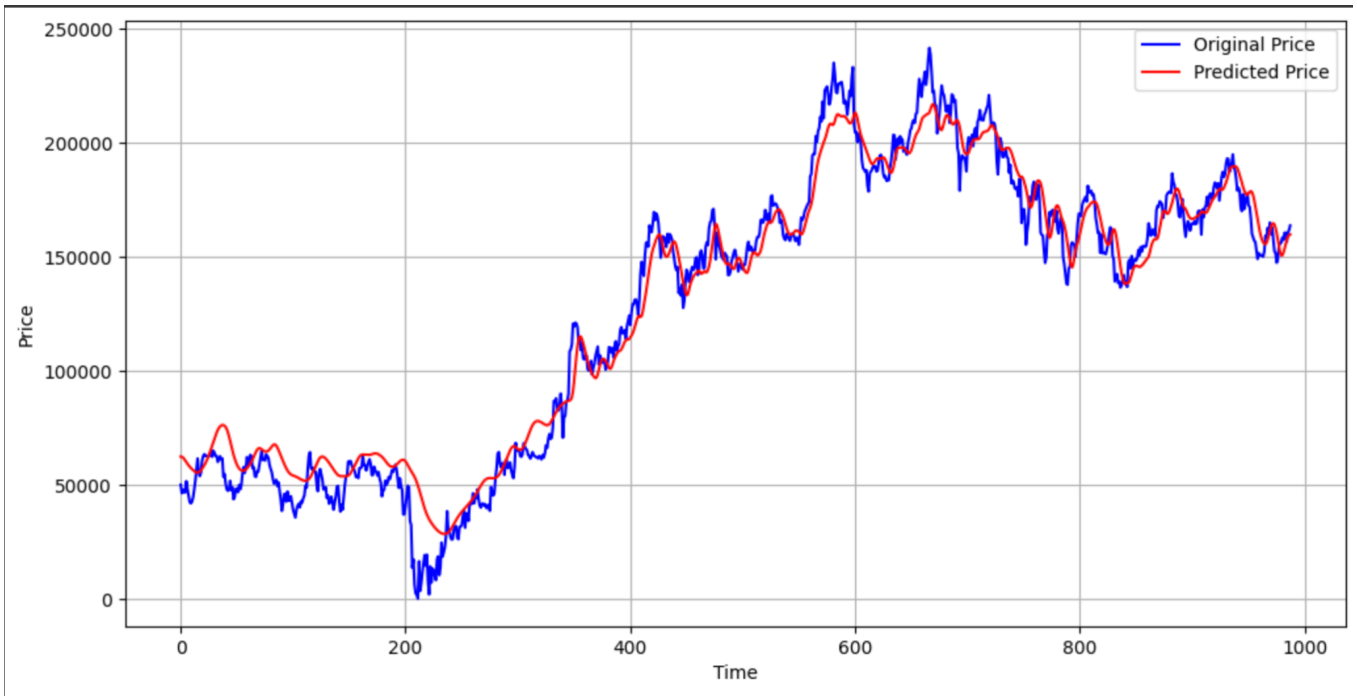
6.2 Moving averages of 100 days



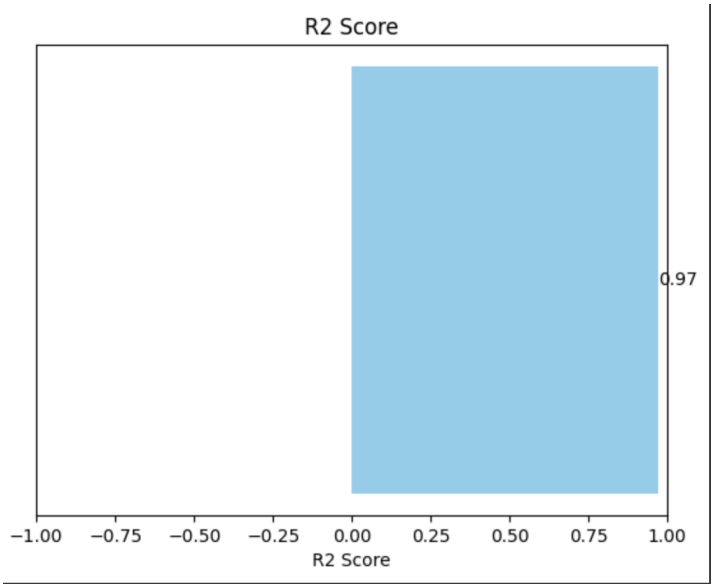
6.3 Moving averages of 200 days

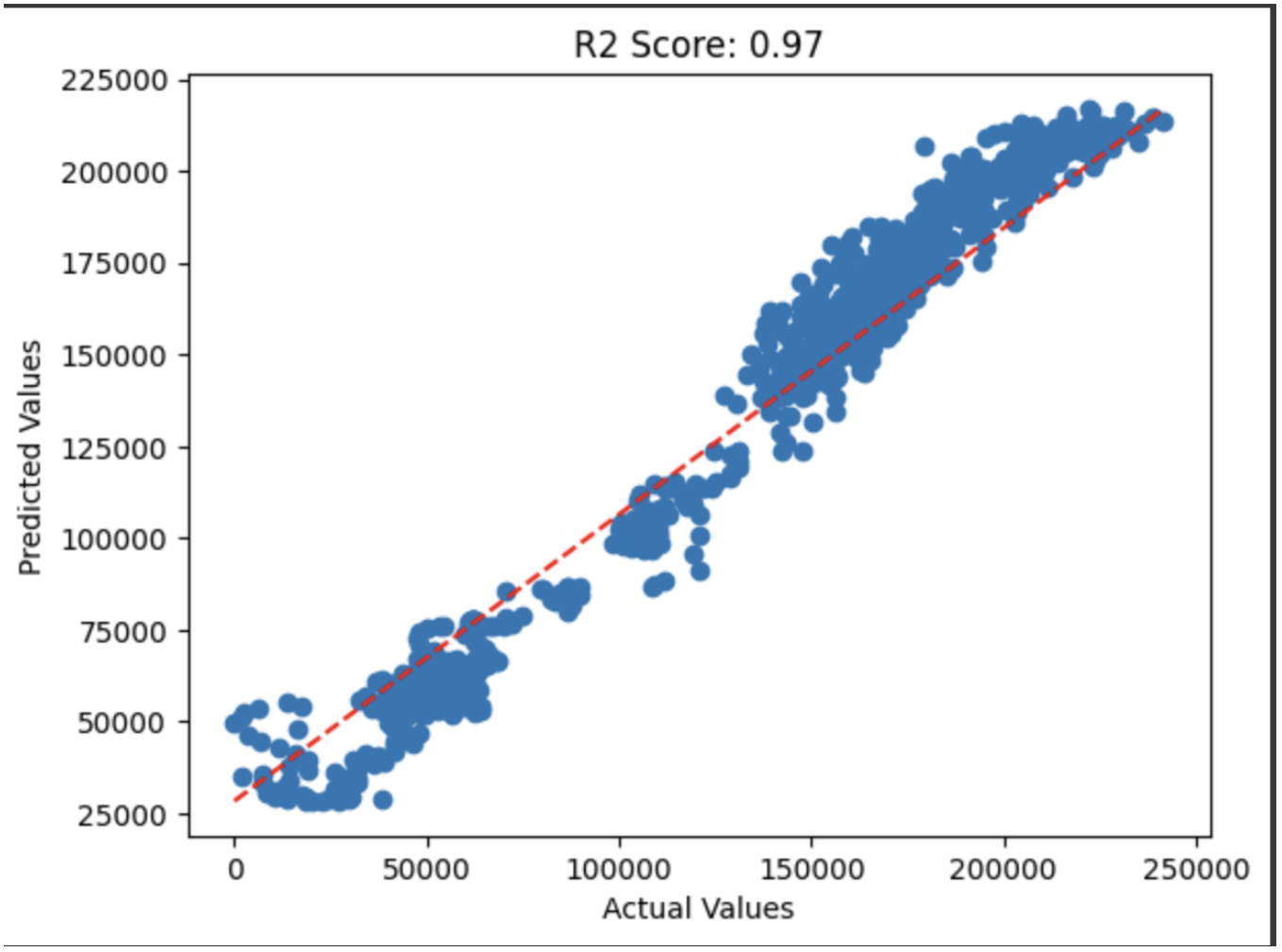


6.4 Final Result



6.5 Evaluation Matrix Graphs





6.6 METRICS COMPARISON OF MODELS

6.6.1 ACCURACY

Accuracy is a highly intuitive metric, so you should not experience any challenges in understanding it. The Accuracy score is calculated by dividing the number of correct predictions by the total prediction number.

$$Accuracy = \frac{TrueNegatives + TruePositive}{TruePositive + FalsePositive + TrueNegative + FalseNegative}$$

6.6.2 F-1 SCORE

The F-1 score(also known as F-measure) is a metric used to evaluate performance of a Machine Learning model. It combines precision and recall into a single score.

F-measure formula:

F-score = 2 * (precision * recall) / (precision + recall)

6.6.3 PRECISION

Precision is defined as the ratio of correctly classified positive samples (True Positive) to a total number of classified positive samples (either correctly or incorrectly).

Precision = True Positive/True Positive + False Positive

Precision = TP/TP+FP

6.6.4 RECALL

The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples. The *recall measures the model's ability to detect positive samples*. The higher the recall, the more positive samples detected.

1. Recall = True Positive/True Positive + False Negative

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

In conclusion, this project focused on utilizing LSTM (Long Short-Term Memory) algorithm for stock market prediction. The LSTM architecture proved to be effective in capturing long-term dependencies and temporal patterns within the sequential data.

By leveraging historical stock prices and relevant financial indicators, the LSTM-based model successfully generated accurate predictions, providing valuable insights to investors and traders.

Through comprehensive evaluation metrics and comparisons with baseline models, the superiority of the LSTM-based approach in capturing the complexities of the stock market was demonstrated.

The model showcased its ability to forecast stock prices over different time horizons, aiding both short-term and long-term investment decision-making processes.

Furthermore, the project explored the potential of incorporating additional features, such as social media sentiment analysis and macroeconomic indicators, to enhance prediction accuracy. This highlighted the importance of considering external factors that influence stock market dynamics.

7.2 Future Enhancements

There are several avenues for future enhancements and research in the field of stock market prediction using LSTM algorithms:

1. Feature Engineering: Further investigation can be done on identifying and incorporating additional relevant features that might impact stock prices, such as news sentiment, corporate events, or geopolitical factors. Enhancing feature engineering techniques could lead to more accurate predictions.

2. Model Architecture: Experimenting with different LSTM model architectures, such as stacked or bidirectional LSTMs, may improve the performance and prediction capabilities. Exploring other variants of RNNs, such as Gated Recurrent Units (GRUs), could also be beneficial.

3. Ensemble Methods: Employing ensemble methods, such as combining predictions from multiple LSTM models or integrating with other machine learning techniques like random forests or gradient boosting, could potentially enhance the overall predictive power and robustness of the model.

4. Hyperparameter Tuning: Conducting a systematic search for optimal hyperparameters, including the number of hidden layers, learning rate, batch size, and regularization techniques, can further optimize the LSTM model's performance.

5. Real-time Data: Expanding the model to incorporate real-time data streams, including intraday price movements and up-to-date financial news, can enable more timely and accurate predictions, catering to high-frequency trading strategies.

6. Interpretability: Exploring techniques to interpret and explain the predictions made by LSTM models can enhance trust and transparency, enabling users to understand the factors and patterns contributing to the forecasts.

Overall, the future enhancements in stock market prediction using LSTM algorithms hold the potential to refine and advance the accuracy and applicability of the models, supporting investors and traders in making informed decisions in the dynamic and ever-changing stock market landscape.

REFERENCES

[1] 1. "Stock Price Prediction Using LSTM, RNN and CNN-SVR Hybrid Models" by Yifei Zhang, Jun Deng, and Xiao Deng (2019). In this paper, the authors compare the performance of LSTM, RNN, and CNN-SVR hybrid models for stock price prediction. The results show that LSTM outperforms the other models in terms of accuracy and efficiency.

[2] "Stock Price Prediction Using Deep Learning and Hybrid Models" by Abhishek Kumar, Vinay Kumar, and Gagandeep Kaur (2019). In this paper, the authors use LSTM and a hybrid model combining LSTM and random forest for stock price prediction. The results show that the hybrid model achieves better performance than LSTM alone.

[3] Stock Price Prediction with LSTM and Random Walk Theory" by Kaijian He, Hanxuan Yang, and Yiran Cui (2018)

[4]"Stock Market Prediction using LSTM and Sentiment Analysis" by Dipta Das et al. (2018). In this paper, the authors use LSTM and sentiment analysis to predict the stock market. The results show that the proposed model achieves better performance than traditional models.

[5] Stock Price Prediction Using LSTM with Financial Indicators" by Aishwarya Kachhwaha et al. (2019)."

[6] On the Difficulty of Training Recurrent Neural Networks" by Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio (2013)