STOCK MARKET PREDICTOR
A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

PRASHAM JAIN[RA2111026010396]
PRIYANSHI MAHESHWARI [RA2111026010409]

Under the guidance of
Dr. Karpagam M

Assistant Professor, Department of CINTEL

in partial fulfillment for the award of the

degreeof

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"STOCK MARKET PREDICTION "** is the bona fide work of **PRASHAM JAIN[RA2111026010396], PRIYANSHI MAHESWARI [RA2111026010409]** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Faculty In-Charge
Dr. Karpagam M
Assistant Professor
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur

SIGNATURE
Head of Department
Dr. R. Annie Uthra
Professor and Head
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur

ABSTRACT

Stock market prediction using artificial intelligence (AI) has gained significant attention due to its potential to provide valuable insights forinvestors and traders. This project aims to develop a predictive model leveraging AI techniques to forecast stock price movements accurately. The methodology involves collecting and preprocessing historical stock data, including prices, volumes, financial statements, and external factors such as economic indicators and news sentiment. Feature selection techniques are employed to identify relevant predictors influencing stock prices. Various machine learning algorithms, including regression models, support vector machines, decision trees, random forests, and neural networks, are evaluated and compared for their predictiveperformance.

INDEX

LIST OF FIGURES

ABBREVIATIONS

LSTM: Long Short-Term Memory

AI: Artificial Intelligence

ML: Machine Learning

DL: Deep Learning

RNN: Recurrent Neural Network

ANN: Artificial Neural Network

GRU: Gated Recurrent Unit

INTRODUCTION

In the ever-evolving landscape of financial markets, the ability to predict stock movements accurately has long been the Holy Grail for investors and analysts alike. With the advent of Artificial Intelligence (AI) and machine learning technologies, the dream of developing a reliablestock market predictor has come closer to reality than ever before.

This project aims to harness the power of AI to build a robust and efficient stock market predictor that not only analyzes historical data but also adapts to real-time market dynamics. Byleveraging advanced algorithms and data analytics techniques, our endeavor seeks to unlock valuable insights from vast volumes of financial data, enabling investors to make informed decisions and capitalize on market opportunities with greater precision.

In this introduction, we will delve into the significance of predicting stock market trends, explorethe challenges inherent in traditional methods, and outline the objectives and methodology of ourAI-driven approach. Additionally, we will highlight the potential impact of such a predictive toolon financial markets and the broader economy, paving the way for a new era of augmented intelligence in finance.

LITERATURE REVIEW

Author	Title	Methods
Troy J. Strader, Drake University, John J. Rozycki, Drake Univ,	Machine Learning Stock Market Prediction Studies: Review and Research Directions	A systematic literature review methodology is used to identify relevant peer-reviewed journal articles from the past twenty years and categorize studies that have similar methods and contexts
Yixin Guo, Södertörn University	Stock Price Prediction Using Machine Learning	This article introduces the theoretical knowledgeof time series model and LSTM neural network,, and then use the root mean square error to compare the prediction results of several models

SYSTEM ARCHITECTURE AND DESIGN

Designing a system architecture for a stock market predictor project using LSTM (Long Short-TermMemory) involves several key components and considerations. Here's an outline of the system architecture and design:

1. Data Preparation:

- **Gather Data:** Collect historical stock market data from reliable sources (APIs, databases, scraping).
- **Clean and Preprocess:** Address missing values, outliers, and inconsistencies. Normalize data for consistent scaling.
- **Feature Engineering:** Create sequences/windows from the time-series data for LSTM training.

2. Building the LSTM Model:

- **Design the Architecture:** Define the number of LSTM layers, hidden units, activation functions, and input/output structure.
- **Train, Split, and Evaluate:** Train the model on historical data. Split data into training, validation (for hyperparameter tuning), and testing sets. Evaluate performance with metrics like MAE, MSE, or RMSE.

3. Refining and Improvement:

- **Optimize Hyperparameters:** Adjust settings like learning rate, batch size, and dropout rate to improve accuracy.
- **Prevent Overfitting:** Implement techniques like early stopping or regularization to avoid overfitting the training data.
- **Sensitivity Analysis:** Assess how the model reacts to changes in features, hyperparameters, or market conditions.

4. Deployment and Monitoring:

- **Production Environment:** Deploy the trained model for real-world use (standalone app, API, trading platform).
- **Performance Tracking:** Monitor the model's performance in real-time, identifying anomalies and potential issues.

5. Continuous Learning:

- **Feedback Loop:** Gather user and stakeholder feedback to identify areas for improvement.
- **Iterative Enhancement:** Based on new data, insights, and feedback, refine the system architecture and model design for better accuracy and usability over time.

METHODOLOGY

Methodology outline for developing a stock market predictor project using LSTM:

This guide presents a step-by-step approach to creating a stock market predictor utilizing LSTM networks. Here's a breakdown of the crucial phases:

1. Define Your Goals and Gather Data (Phase 1):

- **Project Objectives:** Clearly state your aim (e.g., predict prices, identify trends, generate trading signals).
- **Target Market:** Specify the financial instruments you'll focus on (stocks, commodities, cryptocurrencies).
- **Measurable Goals:** Set performance targets for your model (accuracy, precision, profitability).
- **Data Collection:** Acquire historical financial data from reliable sources (daily/intraday prices, trading volumes, etc.).
- **Data Quality:** Ensure completeness, consistency, and accuracy of the data. Address missing values, outliers, and errors.

2. Prepare Your Data (Phase 2):

- **Cleaning:** Remove duplicates, outliers, and irrelevant features from the raw data.
- **Normalization/Scaling:** Ensure all variables have similar scales for better model performance.
- **Feature Engineering:** Create input-output sequences or windows from the time-series data suitable for LSTM training.

3. Build and Train the LSTM Model (Phase 3):

- **Model Design:** Define the LSTM architecture (number of layers, hidden units, activation functions, input/output structure).
- **Data Splitting:** Divide the preprocessed data into training, validation, and testing sets. Use most data for training, a smaller portion for validation (hyperparameter tuning), and a separate set for final evaluation.
- **Training and Optimization:** Train the LSTM model on the training data and optimize hyperparameters (learning rate, batch size) for improved accuracy. Techniques like grid search or Bayesian optimization can be used.

- **Regularization:** Implement techniques (dropout, L2 regularization, early stopping) to prevent overfitting on training data.

4. Evaluate and Refine the Model (Phase 4):

- **Evaluation Metrics:** Use metrics like MAE, MSE, RMSE, or classification accuracy (for predicting price movements) to assess performance.
- **Visualization:** Compare model predictions against actual market data using time series plots, scatter plots, or confusion matrices.
- **Sensitivity Analysis:** Evaluate how the model responds to changes in input features, hyperparameters, or market conditions.

5. Deploy and Monitor the Model (Phase 5):

- **Deployment:** Integrate the trained model into a production environment (standalone app, API, trading platform).
- **System Integration:** Combine the model with other components (data pipelines, user interfaces, backtesting frameworks).
- **Performance Monitoring:** Track the model's performance in real-time, identifying anomalies and potential issues.

6. Continuous Improvement (Phase 6):

- **Feedback Loop:** Gather feedback from users and stakeholders to identify areas for improvement.
- **Iterative Enhancement:** Based on new data, insights, and feedback, refine the system architecture and model design for better accuracy and usability over time.

CODING AND SCREENSHOT

1. Import all the Required Libraries

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM, Bidirectional
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Reading the Google Stock Price Train Dataset

```
gstock = pd.read_csv(r"Google_Stock_Price_Train.csv")
gstock.head(5)
```

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800

3.Using opening values for our experimentation of time series with LSTM.

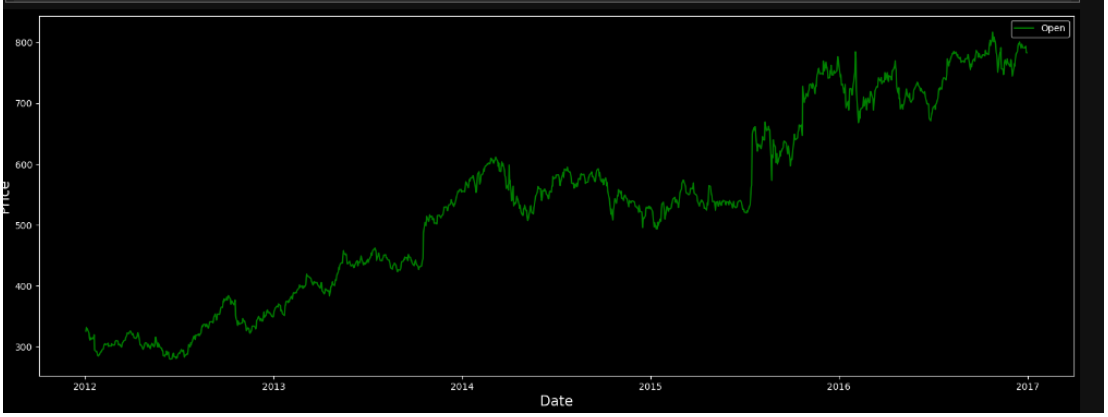
```
gstock_data = gstock[['Date', 'Open']]
gstock_data['Date'] = pd.to_datetime(gstock['Date'].apply(lambda x:x.split()[0]))
gstock_data.set_index('Date', drop = True, inplace = True)
gstock_data.head()
```

	Open
Date	
2012-01-03	325.25
2012-01-04	331.27
2012-01-05	329.83
2012-01-06	328.34
2012-01-09	322.04

4. *Plotting the graph*

```
plt.figure(figsize =(20,7))
plt.plot(gstock_data['Open'], label='Open',color='green')
plt.xlabel('Date',size=15)
plt.ylabel('Price',size=15)
plt.legend()

plt.show()
```



5. Data Pre-processing

```
gstock_data.info()
```

```
class 'pandas.core.frame.DataFrame'>
atetimeIndex: 1258 entries, 2012-01-03 to 2016-12-30
ata columns (total 1 columns):
#    Column  Non-Null Count  Dtype
--   ------  --------------  -----
0    Open    1258 non-null   float64
types: float64(1)
emory usage: 19.7 KB
```

```
gstock_data.shape
```

```
(1258, 1)
```

6.Normalizing the data

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range= (0,1))
scaled_training_set = scaler.fit_transform(gstock_data)
scaled_training_set
```

```
array([[0.08581368],
       [0.09701243],
       [0.09433366],
       ...,
       [0.95725128],
       [0.93796041],
       [0.93688146]])
```

7.CREATING X_TRAIN AND Y_TRAIN DATA STRUCTURES

```
X_train = []
y_train = []

for i in range(60,1258):
    X_train.append(scaled_training_set[i-60:i,0])
    y_train.append(scaled_training_set[i,0])

X_train = np.array(X_train)
y_train = np.array(y_train)

X_train.shape, y_train.shape
```

```
((1198, 60), (1198,))
```

Reshaping the data

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

```
(1198, 60, 1)
```

8. build the RNN model using LSTM and other Layers

```
model = Sequential()

model.add(LSTM(units=50, return_sequences = True, input_shape = (X_train.shape[1],1)))
model.add(Dropout(0.2))

model.add(LSTM(units=50, return_sequences = True))
model.add(Dropout(0.2))

model.add(LSTM(units=50, return_sequences = True))
model.add(Dropout(0.2))

model.add(LSTM(units=50))
model.add(Dropout(0.2))

model.add(Dense(units=1))

model.compile(optimizer='adam',loss='mean_squared_error')

model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_12 (LSTM)               (None, 60, 50)            10400

dropout_12 (Dropout)         (None, 60, 50)            0

lstm_13 (LSTM)               (None, 60, 50)            20200

dropout_13 (Dropout)         (None, 60, 50)            0

lstm_14 (LSTM)               (None, 60, 50)            20200

dropout_14 (Dropout)         (None, 60, 50)            0

lstm_15 (LSTM)               (None, 50)                20200

dropout_15 (Dropout)         (None, 50)                0
```

9. Fitting the model on X_train and y_train

```
model.fit(X_train,y_train, epochs=100, batch_size = 32)
```

```
Epoch 1/100
38/38 [==============================] - 12s 107ms/step - loss: 0.0420
Epoch 2/100
38/38 [==============================] - 4s 116ms/step - loss: 0.0069
Epoch 3/100
38/38 [==============================] - 5s 143ms/step - loss: 0.0052
Epoch 4/100
38/38 [==============================] - 4s 107ms/step - loss: 0.0058
Epoch 5/100
38/38 [==============================] - 5s 134ms/step - loss: 0.0051
Epoch 6/100
38/38 [==============================] - 4s 111ms/step - loss: 0.0046
Epoch 7/100
38/38 [==============================] - 4s 107ms/step - loss: 0.0047
Epoch 8/100
38/38 [==============================] - 5s 138ms/step - loss: 0.0041
Epoch 9/100
38/38 [==============================] - 4s 107ms/step - loss: 0.0044
Epoch 10/100
38/38 [==============================] - 4s 107ms/step - loss: 0.0047
Epoch 11/100
38/38 [==============================] - 5s 137ms/step - loss: 0.0043
Epoch 12/100
38/38 [==============================] - 4s 108ms/step - loss: 0.0038
Epoch 13/100
38/38 [==============================] - 4s 107ms/step - loss: 0.0042
Epoch 14/100
38/38 [==============================] - 5s 139ms/step - loss: 0.0042
Epoch 15/100
38/38 [==============================] - 4s 108ms/step - loss: 0.0040
Epoch 16/100
38/38 [==============================] - 6s 149ms/step - loss: 0.0036
Epoch 17/100
38/38 [==============================] - 5s 127ms/step - loss: 0.0036
Epoch 18/100
38/38 [==============================] - 4s 106ms/step - loss: 0.0034
Epoch 19/100
38/38 [==============================] - 5s 128ms/step - loss: 0.0037
```

10. Extracting the actual Stock prices of Jan -2017.

```
test_data = pd.read_csv("Google_Stock_Price_Test.csv")
actual_stock_price = test_data.iloc[:,1:2].values
actual_stock_price
```

```
array([[778.81],
       [788.36],
       [786.08],
       [795.26],
       [806.4 ],
       [807.86],
       [805.  ],
       [807.14],
       [807.48],
       [807.08],
       [805.81],
       [805.12],
       [806.91],
       [807.25],
       [822.3 ],
       [829.62],
       [837.81],
       [834.71],
       [814.66],
       [796.86]])
```

11. Preparing the input for the Model

```
dataset_total = pd.concat((gstock_data["Open"],test_data["Open"]), axis= 0)
inputs = dataset_total[len(dataset_total) - len(test_data) - 60:].values

inputs = inputs.reshape(-1,1)
inputs = scaler.transform(inputs)

X_test = []
for i in range(60,80):
    X_test.append(inputs[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))
```

Predicting the values for Jan 2017 Stock Prices

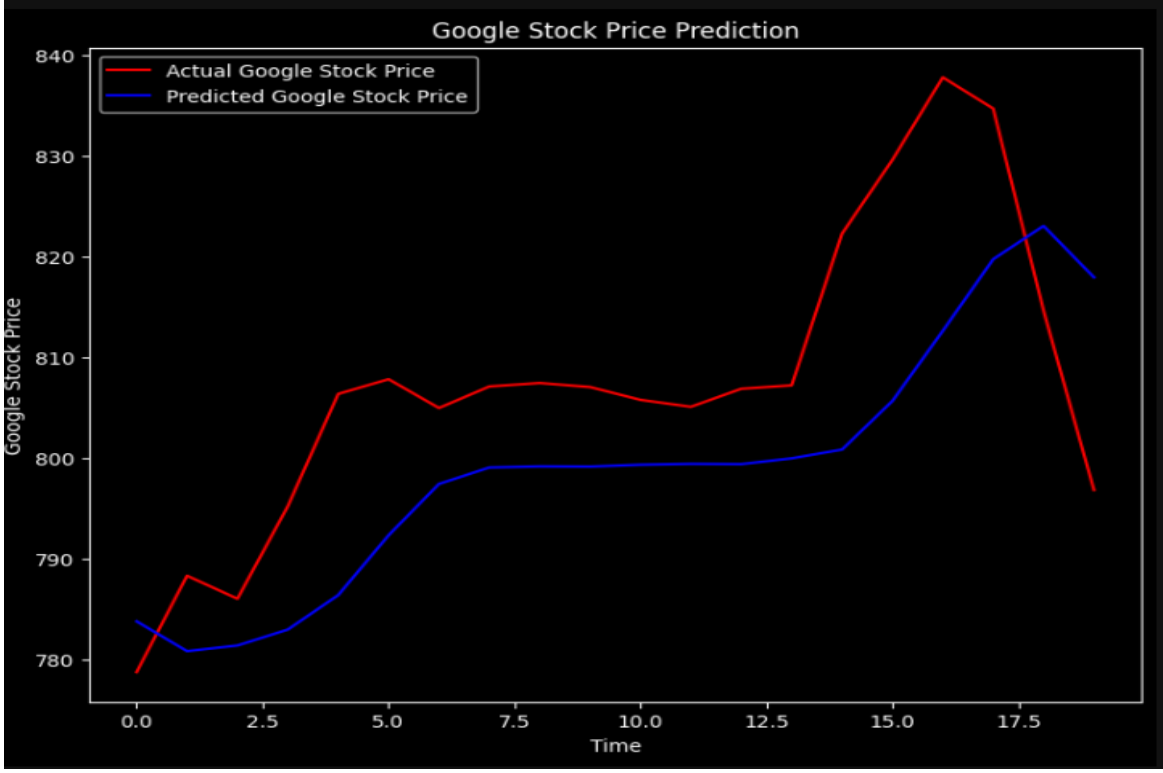
```
predicted_stock_prices = model.predict(X_test)
predicted_stock_prices = scaler.inverse_transform(predicted_stock_prices)
predicted_stock_prices
```

```
/1 [==============================] - 4s 4s/step
array([[783.8368 ],
       [780.8792 ],
       [781.442  ],
       [783.02277],
       [786.4381 ],
       [792.37494],
       [797.4675 ],
       [799.1022 ],
       [799.20807],
       [799.18713],
       [799.37427],
       [799.4602 ],
       [799.43365],
       [800.0086 ],
       [800.88007],
       [805.69867],
       [812.65717],
       [819.7622 ],
```

12. Plotting the Actual vs Predicted Open Prices for Google Stocks

```
plt.figure(figsize = (9,7))
plt.plot(actual_stock_price, color = 'red', label = 'Actual Google Stock Price')
plt.plot(predicted_stock_prices, color = "blue", label = "Predicted Google Stock Price")
plt.title("Google Stock Price Prediction")
plt.xlabel("Time")
plt.ylabel("Google Stock Price")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x79f021169fc0>
```



CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, overcoming the limitations of existing methodologies in stock prediction requires a multifaceted approach that leverages alternative data sources, advanced modeling techniques, and rigorous evaluation methods. By incorporating diverse data streams, such as social media sentiment and consumer behavior, researchers can gain deeper insights into market dynamics and investor sentiment. Enhanced feature engineering, ensemble modeling, and deep learning architectures enable the extraction of complex patterns from financial data, leading to more accurate predictions.

Future Enhancements

To further enhance the stock market predictor project and address its limitations, several future enhancements can be considered:

1. Feature Engineering: Explore additional input features such as sentiment analysis of news articles, social media sentiment, economic indicators, or technical indicators to capture more nuanced market dynamics.
2. Model Ensemble: Implement ensemble learning techniques to combine predictions from multiple models, including LSTM, convolutional neural networks (CNNs), or traditional statistical models, to improve predictive accuracy and robustness.

REFERENCES

1. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
2. https://www.sas.com/en_us/insights/analytics/machine-learning.html
3. https://groww.in/us-stocks/googl
4. https://www.investopedia.com/terms/d/deep-learning.asp
5. https://www.nasdaq.com/market-activity/stocks/google