MULTIMODAL APPROACH FOR PORTFOLIO OPTIMISATION

A PROJECT REPORT

submitted by

ANUPAMA AJITH

Reg. No. MAC19CS007

JAIMY SIMON

Reg. No. MAC19CS029

SOHAIL PM

Reg. No. MAC19CS052

to

A P J Abdul Kalam Technological University

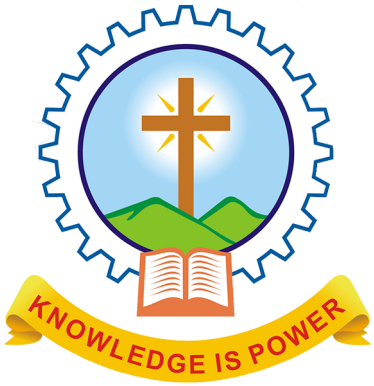
in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology

in

Computer Science and Engineering



Department of Computer Science & Engineering

Mar Athanasius College of Engineering

Kothamangalam, Kerala, India 686 666

DECEMBER 2022

MULTIMODAL APPROACH FOR PORTFOLIO OPTIMISATION

A PROJECT REPORT

submitted by

ANUPAMA AJITH

Reg. No. MAC19CS007

JAIMY SIMON

Reg. No. MAC19CS029

SOHAIL PM

Reg. No. MAC19CS052

to

A P J Abdul Kalam Technological University

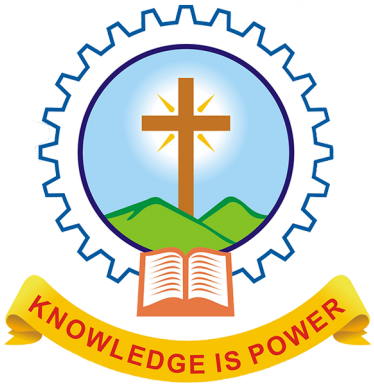
in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology

in

Computer Science and Engineering



Department of Computer Science & Engineering

Mar Athanasius College of Engineering

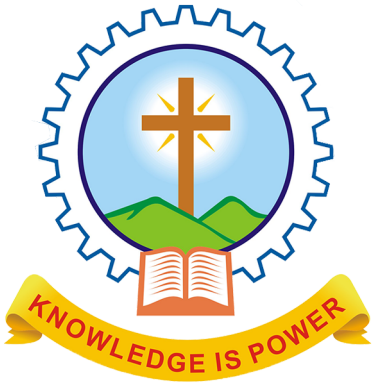
Kothamangalam, Kerala, India 686 666

DECEMBER 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAR ATHANASIUS COLLEGE OF ENGINEERING

KOTHAMANGALAM



CERTIFICATE

This is to certify that the report entitled **Multimodal approach for Portfolio Optimisation** *submitted by* **Anupama Ajith (MAC19CS007), Jaimy Simon (MAC19CS029), Sohail PM (MAC19CS052)**, *towards partial fulfillment of the requirement for the award of Degree of Bachelor of Technology in Computer Science and Engineering from APJ Abdul Kalam Technological University for December 2022 is a bonafide record of the project carried out by them under our supervision and guidance.*

.....	.....	.....
Dr.Elizabeth Issac	**Dr.Surekha Varghese**	**Prof.Joby George**
Coordinator	***Project Guide***	***Head of the Department***

Date: Dept. Seal

ACKNOWLEDGEMENT

We express our sincere gratitude and thanks to Dr. Bos Mathew Jos, Principal and Prof. Joby George, Head of the Department for providing the necessary facilities and their encouragement and support.

We owe special thanks to the project guide Dr. Surekha Mariam Varghese and project coordinator Dr. Elizabeth Issac for their corrections, suggestions and sincere efforts to co-ordinate the project under a tight schedule.

We express our sincere thanks to staff members in the Department of Computer Science and Engineering who have taken sincere efforts in guiding and correcting us in conducting this project.

Finally, we would like to acknowledge the heartfelt efforts, comments, criticisms, co-operation and tremendous support given by our dear friends during the preparation of the project and also during the presentation without which this work would have been all the more difficult to accomplish.

ABSTRACT

Data has always been a key component of trading, and traders have long sought to edge themselves by having access to greater information. Traditionally, The majority of the time, investors used publicly accessible market and fundamental data. In order to estimate profits per share and stock prices, traditional tactics concentrate on equity fundamentals and create financial models from published financials, sometimes in combination with industry or macro data. Conversely, they use indicators calculated from price and volume data to derive signals from market data using technical analysis. But essentially, a variety of information sources have an impact on stock markets., broadly, fundamental and financial-related news. Our goal is to devise a tool that leverages ML models, specifically implementing stacked models, wherein a bigger stacking ensemble model for training and prediction incorporates neural network sub-models by combining media sentiments, technical fundamental analysis. To accomplish this task, relevant data has to be collected and informative features must be extracted. Models and trading strategies are to be designed and tuned to do the predictive task. Then, its performance is evaluated on employing a backtesting engine and historical data and finally, it would be deployed on live stock data.

Table of Contents

ACKNOWLEDGEMENT i

ABSTRACT ii

List of Abbreviations v

1 Introduction 1

1.1 Project Outline 2

2 Background 3

2.1 Raw Data 3

2.1.1 Data Types 3

2.2 Feature Extraction 4

2.2.1 FinBert 4

2.3 Prediction Model 4

2.3.1 Long Short Term Memory (LSTM) 5

2.3.2 LSTM as baseline model 5

2.4 Backtesting 6

2.4.1 Backtesting.py 6

2.5 Model Evaluation 7

2.5.1 Sharpe Ratio 7

3 Related works 8

3.1 Based on Target Output and Frequency 8

3.2 Based on Surveyed Markets 8

3.3 Based on Combination of Data 9

3.4 Based on Prediction Models 9

4 Implementation **11**

4.1 Data Collection 11

4.1.1 Historical Stock Data 11

4.1.2 News Data 12

4.2 LSTM Model 13

4.3 Sentiment Analysis 15

4.3.1 Data Preprocessing 15

4.3.2 Analysis 15

4.4 Data Split 16

4.5 Backtesting 17

5 Results **19**

5.1 Effectiveness of Multimodal approach 19

5.2 Effectiveness of LSTM as baseline model 20

6 Conclusion **21**

REFERENCES **22**

List of Abbreviations

LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
EMH	Efficient Market Hypothesis
ARMA	Autoregressive–moving-average model
BERT	Bidirectional Encoder Representations from Transformers

Chapter 1

Introduction

Stock market forecasting has drawn interest from both economists and computer scientists as a classic yet difficult topic. The well-known efficient market hypothesis (EMH), which holds that technical analysis or fundamental analysis (or any analysis) would not deliver any persistent above-average profit to investors, provides a pessimistic stance and implies that the financial market is efficient for this problem. Several scientists, however, disagree with EMH. While some studies are attempting to construct successful stock market prediction models, some studies are attempting to quantify the various efficiency levels for mature and emerging markets.

The stories of fundamental analysis and technical analysis serve as the effort's starting point. Technical analysis solely considers charts and trends, but fundamental analysis examines the stock price based on its inherent worth, or fair value. Experience-based technical indicators can also be used as custom input features for deep learning and machine learning models.

Moreover, with the availability of online news or twitter data has provided new sources for predicting the stock market. Apart from these, professional investors also map the interconnectedness of firms and events using their extensive understanding of inter-market and inter-company relationships, and use this map to improve market predictions. So another scope for enhancing the predictions is incorporating company relations using graphs.

1.1 Project Outline

The objective is to devise a tool that leverages ML models, specifically implementing stacked models, which integrate smaller neural network models into a bigger stacking ensemble model for prediction and training by combining media sentiments, technical fundamental analysis. To accomplish this task, relevant data has to be collected and informative features must be extracted. Models and trading strategies are to be designed and tuned to do the predictive task. Then, its performance is evaluated on employing a backtesting engine and historical data and finally, it would be deployed on live stock data.

Chapter 2

Background

The general workflow of stock market prediction can be summarized in high level in the following sections:

2.1 Raw Data

The first stage in forecasting is to gather reliable data to use as a foundation. With the understanding that history repeats itself, it might be extrinsic data sources or intrinsic past pricing that have an impact on the stock market.

2.1.1 Data Types

The raw data that are frequently utilised to forecast the stock market are categorized into the following:

- Market data: This refers to all stock market trading activity, such as open, high, low, and close prices, trading volume, etc.
- Text data: This is the text that people have provided, such as from news articles, social media posts, web searches, etc. This text data can be subjected to sentiment analysis, which will yield a sentiment component (such as positive, negative, or neural) that can then be utilised for prediction.
- Fundamental data: Accounting data, such as assets, liabilities, and other items that are reported on a quarterly basis, is the most typical kind of fundamental data.

- Analytics data: This is information that may be gleaned from reports (such as a recommendation to buy or sell a stock) that are given by investment banks and research businesses after thorough examination of the business models, operations, rivalries, and other factors of various companies.

2.2 Feature Extraction

Technical analysis produces many indicators for predicting the direction of prices based on previous prices and volumes, such as moving averages or moving average convergence/divergence (MACD), Relative strength index (RSI), etc.

Articles and headlines were gathered from Google News between July 2010 and July 2020 for sentiment research. Using the NLTK (Natural Language Toolkit) library, the scraped articles were cleaned and pre-processed before being sent into the finBERT transformer pipeline for sentiment analysis.

2.2.1 FinBert

FinBERT is a BERT model pre-trained on financial communication text. The objective is to advance financial NLP theory and application. Huggingface's transformers library has been uploaded and merged with the improved FinBERT model for categorising financial sentiment. 10,000 manually annotated (positive, negative, or neutral) phrases from analyst reports are used to fine-tune this model. This model performs better than average when it comes to financial tone analysis.

2.3 Prediction Model

Existing standard models perform well at early stages of research. But on further research their variants are further developed to improve the prediction performance.

The approach in our proposed work is to use stacked models, where neural network sub-models are embedded in a larger stacking ensemble model for training and prediction. he baseline model chosen is Long Short Term Memory (LSTM) model for its ability to handle long dependencies by means of gate mechanism.

2.3.1 Long Short Term Memory (LSTM)

The flaw of regular RNNs having vanishing and exploding gradients has been improved using LSTM models. When creating sequences from time series data, LSTMs perform remarkably well. Forget, Input, and Output are the three gates that make up LSTM's internal structure. The requirement to keep or discard information is governed by the forget gate. The input gate based on cell states learns the conditions and dependencies, aiding in the recognition and retention of sequences. Whatever information will be transmitted forward as an input to the following cell is decided by the output gate.

2.3.2 LSTM as baseline model

Here are some examples of stock prediction models that use LSTM as a baseline:

- LSTM with exogenous variables: This model uses additional input features, known as exogenous variables, to make predictions. These variables can include economic indicators, news articles, and other data that may influence the stock price.

- Multivariate LSTM: This model uses multiple time series as input, rather than just a single series. This can be useful if there are multiple stocks or other assets that you want to predict.

- LSTM with attention: This model uses an attention mechanism to allow the model to focus on specific parts of the input sequence when making predictions. This can be useful if certain parts of the sequence are more important than others.

- LSTM with variable-length sequences: This model can handle input sequences of variable length, which can be useful if the length of the input sequences varies over time.

- Hybrid LSTM-ARIMA model: This model combines an LSTM with an autoregressive integrated moving average (ARIMA) model to make predic-

tions. The LSTM is used to capture long-term dependencies in the data, while the ARIMA model is used to capture short-term dependencies.

2.4 Backtesting

A backtesting engine, to put it simply, iterates over past prices (and other data), sends the most recent values to your algorithm, gets orders in return, and records the positions that result and their value.
Typically, a backtest is developed by a programmer simulating the trading technique. Stock, bond, and other historical financial instrument data are used to perform the simulation. The person conducting the backtest will evaluate the model's returns across several datasets. To judge performance objectively, the model must also be tested in a variety of market scenarios. The model's variables are then adjusted for optimum performance against various backtesting metrics.

2.4.1 Backtesting.py

backtesting.py is a Python library that allows you to backtest trading strategies in an easy and flexible way. It provides a simple and intuitive API for defining strategies and running backtests, as well as a variety of tools for analyzing and visualizing the results of the backtest.

With backtesting.py, you can define a strategy as a class that inherits from the Strategy class and implements a few simple methods. The init() method is called when the strategy is initialized, and the next() method is called at each time step of the backtest. In the next() method, you can use the buy() and sell() methods to simulate trades.

The Backtest class is used to run the backtest and evaluate the performance of the strategy. You can specify the data to use for the backtest, the strategy to use, and various other parameters such as the initial cash and the commission rate. The run() method will run the backtest and return a BacktestResult object with various metrics on the performance of the strategy, such as the profit-and-loss curve and the Sharpe ratio.

Overall, backtesting.py is a powerful and easy-to-use library that can help

you quickly prototype and test trading strategies. It is a useful tool for both quantitative traders and researchers.

2.5 Model Evaluation

If the predicted-based trading strategy may generate a profit or not will be determined by the model. Typically, it is assessed from two perspectives: return and risk. The average return over the risk-free rate per unit of volatility, or the Sharpe Ratio, is a comprehensive indicator that takes both return and risk into account.

2.5.1 Sharpe Ratio

The Sharpe ratio evaluates the relationship between an investment's return and risk. The idea that excess returns over time may indicate greater volatility and risk rather than investment expertise is expressed mathematically in this way. The numerator of the Sharpe ratio is the difference over time between realised or predicted returns and a benchmark, such as the performance of a certain investment category or the risk-free rate of return. The standard deviation of returns over the same time period, which is a gauge of volatility and risk, serves as the denominator.

Chapter 3

Related works

Long-term research has been conducted on stock market forecasting, and the emergence of deep learning techniques in this area has been accompanied by various review studies. An overview of some of the latest progress in the past couple of years, primarily deep learning techniques are summarized below:

3.1 Based on Target Output and Frequency

Regression problems are those where the goal is to predict a specific price value; classification problems are those where the goal is to predict the direction of price movement, such as whether it will go up or down. Just a small number of research take intraday predictions, such 5- or hourly predictions, into account; the majority of studies exclusively take daily predictions into account. The difficulty of gathering the necessary data could partially justify the cause of this. While intraday data is fairly scarce in academia, daily historical prices and news titles are simpler to gather and process for research..

3.2 Based on Surveyed Markets

Most studies would concentrate on a single market, while others would test their models across several markets. In research, both established markets (such as the US) and developing markets (such as China) are receiving a lot of attention.

3.3 Based on Combination of Data

There are various degrees of acquiring and processing difficulty for various types of raw data. A significant amount of input data is required for deep learning models in order to train a sophisticated neural network model. In this situation, market data is the greatest option and is most frequently employed since it offers the highest data sample size, whereas the other data kinds typically have a lesser size. Due to the popularity of social media and online news websites as well as the simplicity of using web crawlers to collect text data, text data is utilised for the second-most applications. An extreme example is the analytics data, which is never employed in the studies that were surveyed due to the sparse data and high access costs. Also, there is a trend towards more varied data kinds, which shows that it is challenging to obtain improved prediction outcomes using simply market data.

3.4 Based on Prediction Models

The majority of prediction models use supervised learning, where test sets are used for evaluation and training sets are used for training. When the labels are unavailable during the feature extraction step, only a small number of research employ semi-supervised learning. Moreover, different prediction models are divided into three categories: hybrid models, alternative models, and standard models and their variations. Three kinds of deep learning models—feedforward neural network, convolutional neural network, and recurrent neural network—are frequently employed for standard models.

The performance of standard models is good in the early stages of research, but their variations are further refined to enhance forecast accuracy. Using stacked models, which integrate neural network sub-models in a larger stacking ensemble model for training and prediction, is one strategy. The attention mechanism, which is a generalised pooling method with bias alignment over inputs, has also

been incorporated into recurrent neural network models.

LSTM-based DNN, which consists of three components, namely, LSTM, VADER model, and differential privacy mechanism that integrates various news sources, and AdaBoost algorithm, which generates both training samples and ensemble weights for each LSTM predictor, are some of the popular hybrid models. The sentiment-ARMA model, for example, incorporates the news articles as hidden information.

Generative adversarial networks, transfer learning, and reinforcement learning are other models that have only recently been developed and are still in their early stages of application for stock market prediction..

Chapter 4

Implementation

4.1 Data Collection

4.1.1 Historical Stock Data

The opening price (Open), highest and lowest price the stock ever traded at (High, Low), closing price (Close), volume of stocks traded (Volume), and adjusted close will all be downloaded using the yfinance API.

```
import requests
import json

# Enter the ticker symbol for the company
whose stock data you want to retrieve
ticker = "AAPL"

# Enter the start and end dates for the range
of data you want to retrieve

start_date = "2022-01-01"
end_date = "2022-12-31"

# Set the frequency of data points
frequency = "1d"  # daily data
```

```
# Construct the URL for the API request

url = f"https://query1.finance.yahoo.com/v8/
finance/chart/{ticker}?formatted=true&crumb=
swg2ljKDY0m&lang=en-US&region=US&interval=
{frequency}&period1={start_date}&period2={end_date}"

# Make the API request
response = requests.get(url)

# Convert the response to a JSON object
data = response.json()

# Extract the stock data from the JSON object
stock_data = data["chart"]["result"][0]
["indicators"]["quote"][0]

# Print the stock data
print(stock_data)
```

4.1.2 News Data

To perform sentiment analysis news and headlines were scraped from Google News using FeedParser from July 2010 to July 2020.

```
import feedparser

# Make a request to the news website's RSS feed
d = feedparser.parse('https://www.example.com/news/rss')

# Print the news headlines
for entry in d.entries:
    print(entry.title)
```

```
for entry in d.entries:
    print(f"{entry.title} ({entry.published})")
    print(entry.link)
```

4.2 LSTM Model

Once the data is obtained, it has to be preprocessed. This may involve cleaning the data, normalizing it, and creating sequences of data that the LSTM can use to learn patterns. Then LSTM network is trained on the preprocessed data. The appropriate architecture has to be decided for the LSTM model, such as the number of layers and the number of hidden units in each layer. The following is a general outline of prediction model:

```
import numpy as np
import pandas as pd

# Load the data
df = pd.read_csv("stock_data.csv")

# Preprocess the data
df = df.dropna()
df = df.reset_index(drop=True)

# Split the data into training and test sets
train_data = df[:int(len(df) * 0.8)]
test_data = df[int(len(df) * 0.8):]

# Extract the stock prices and normalize them
stock_prices = df["Close"].values.astype(float)
stock_prices = (stock_prices - stock_prices.mean()) / stock_prices.std()

# Create sequences of data for the LSTM
sequence_length = 50
```

```
X_train = []
y_train = []
for i in range(sequence_length, len(stock_prices)):
    X_train.append(stock_prices[i-sequence_length:i])
    y_train.append(stock_prices[i])
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))

# Build the LSTM model
from keras.models import Sequential
from keras.layers import LSTM, Dense
model = Sequential()
model.add(LSTM(units=50, return_sequences=True,
input_shape=(X_train.shape[1], 1)))
model.add(LSTM(units=50))
model.add(Dense(1))

# Compile and fit the model
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(X_train, y_train, epochs=1, batch_size=32)

# Make predictions on the test set
X_test = []
y_test = []
for i in range(sequence_length, len(stock_prices)):
    X_test.append(stock_prices[i-sequence_length:i])
    y_test.append(stock_prices[i])
X_test, y_test = np.array(X_test), np.array(y_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
predictions = model.predict(X_test)

# Evaluate the model
```

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, predictions)
print("Mean Squared Error: ", mse)
```

4.3 Sentiment Analysis

Sentiment analysis will then be performed using FinBert to find sentiment scores before combining the results with historical stock price data to determine whether news sentiment influences stock price direction.

FinBERT is a pre-trained language model developed by FINRA (Financial Industry Regulatory Authority) that can be fine-tuned for a variety of natural language processing tasks, including sentiment analysis.

To do sentiment analysis using FinBERT, the necessary libraries and dependencies has to be installed. The transformers library can be used to download a pre-trained FinBERT model, or can fine-tune a FinBERT model on our own dataset.

4.3.1 Data Preprocessing

This approach has a flaw in that humans are unable to process lengthy texts. So, we will divide our text into chunks of no more than 512 tokens each in order to calculate the sentiment for larger texts. We send our chunks through our model to obtain the sentiment scores for each chunk after transforming them to make them suitable for FinBERT consumption. Lastly, an average is calculated for each sentiment category, giving us a general estimate of the sentiment for the entire text.

4.3.2 Analysis

Once the data is preprocessed, we load into the pre-trained FinBERT model and set it to evaluation mode. The tokenized input is passed through the model and applying a linear layer and a softmax function to the output to obtain the predicted class probabilities.

```
import transformers

# Load the pre-trained FinBERT model
model = transformers.BertModel.from_pretrained('finbert-base-uncased')

# Set the model to evaluation mode
model.eval()

import torch

# Classify the sentiment of a single text sample
input_ids = torch.tensor(tokenized_input).unsqueeze(0)  # batch size 1
output = model(input_ids)[0]  # take the output of the first batch
output = torch.softmax(output, dim=1)
predicted_class = torch.argmax(output).item()

class_labels = {0: "negative", 1: "neutral", 2: "positive"}

print(f"Sentiment: {class_labels[predicted_class]}")
```

4.4 Data Split

In the domains of machine learning and deep learning, the in-sample/out-of-sample split or train/validation/test split of data samples is frequently used for evaluating various prediction models. Models must, however, be able to accurately forecast the market over a wide time horizon in the financial sector.

To divide the entire dataset into the training and test sets, the rolling window analysis approach is used. Here, a fixed window of 2000 timesteps is utilised for training and 200 timesteps for testing, and that window is slid through the entire amount of timesteps. During training and testing, for instance, the first iteration will use the indexes 0 to 2000 and 2000 to 2200, respectively. The indices for the second iteration will be 200 to 2200, 2200 to 2400, and so on. This approach

represents the premise that recent stock prices have more predictive potential than older ones, as well as ensuring that the model can be generalised across large time horizons.

4.5 Backtesting

Once the trained model is used to generate trades on the validation and test data, the model should output a prediction for each time step, and these predictions can be used to decide when to buy or sell the assets.

The data is iterated and trades are done according to the model's predictions and keeping track of your portfolio value. This is started with a certain amount of cash and using the prices in the data to simulate buying and selling the assets.

Then the performance of your model is evaluated and this may include calculating metrics such as the Sharpe ratio, the maximum drawdown, and the overall return. The following is a general outline of the backtesting module:

```
import pandas as pd
from backtesting import Backtest, Strategy
from backtesting.lib import crossover

# Define a strategy that uses the ML model to make trades
class MLStrategy(Strategy):
    def init(self):
        # Set the model as a class attribute
        self.model = model

    def next(self):
        # Use the model to generate predictions for the next time step
        prediction = self.model.predict(self.data)[0][0]

        # Buy if the prediction is positive, sell if it is negative
        if prediction > 0:
            self.buy()
```

```
        elif prediction < 0:
            self.sell()

# Run the backtest
bt = Backtest(test_data, MLStrategy, cash=10000, commission=0.001)
results = bt.run()

# Print the results
print(results)
```

Chapter 5

Results

5.1 Effectiveness of Multimodal approach

Using a multimodal approach that includes news data can produce better results for stock prediction because news articles can contain valuable information about a company or the overall market. For example, a news article might discuss a company's earnings, new products or services, or changes to its management team. This information can be useful for predicting the stock price, as it can indicate whether a company is performing well or poorly.

In addition to providing useful information about a company, news articles can also contain sentiment information that can be useful for stock prediction. For example, if a news article has a positive tone, it might suggest that the company or the overall market is performing well. On the other hand, a negative tone might suggest that the company or market is performing poorly.

By using a multimodal approach that includes news data, it is possible to incorporate both the content and sentiment of news articles into the prediction model. This can lead to more accurate predictions, as it allows the model to consider a wider range of information.

It is important to note that using a multimodal approach is just one approach to stock prediction and it is not a guarantee of success. There are many factors that can affect the performance of a prediction model, including the quality of the data and the choice of model architecture.

5.2 Effectiveness of LSTM as baseline model

This section discusses the results of implementing LSTM as baseline model for stock prediction:

1. LSTM models are able to capture long-term dependencies in the data, which is important in the stock market as there can be long-term trends that affect the price of a stock.

2. They are able to handle a large amount of noise in the data, which is common in the stock market where prices can be affected by many unpredictable events.

3. These models are able to make predictions using data from multiple time steps, which is important in the stock market as past prices can be a good indicator of future prices.

4. LSTM models are able to handle missing data and are resistant to forgetting important information, which is important in the stock market where data can be incomplete or noisy.

Chapter 6

Conclusion

Stock markets are strongly affected by interrelated information. This can be exploited in order build better stock market prediction models. This project aims to devise a tool that leverages ML models, specifically implementing stacked models, where neural network sub-models are embedded in a larger stacking ensemble model for training and prediction by combining media sentiments, technical fundamental analysis. To accomplish this task, relevant data has to be collected and informative features must be extracted. Models and trading strategies are to be designed and tuned to do the predictive task. Then, its performance is evaluated on historical data using a backtesting engine and finally, it would be deployed on live stock data.

REFERENCES

[1] Q. Li, J. Tan, J. Wang and H. Chen, "A Multimodal Event-Driven LSTM Model for Stock Prediction Using Online News," in IEEE Transactions on Knowledge and Data Engineering, vol. 33, no. 10, pp. 3323-3337, 1 Oct. 2021, doi: 10.1109/TKDE.2020.2968894.

[2] Weiwei Jiang, Applications of deep learning in stock market prediction: Recent progress, Expert Systems with Applications, Volume 184, 2021, 115537, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2021.115537.

[3] Chong, E., Han, C., Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. Expert Systems with Applications, 83, 187 – 205. URL: http://www.sciencedirect.com/science/article/pii/S0957417417302750. doi:https://doi.org/10.1016/j.eswa.2017.04.030

[4] Araci, Dogu. "Finbert: Financial sentiment analysis with pre-trained language models." arXiv preprint arXiv:1908.10063 (2019).