A dissertation submitted to the **University of Greenwich**
in partial fulfilment of the requirements for the Degree of

Master of Science
in
Data Science

Predicting Insights of Company Using Machine Learning & Deep Learning Techniques

Name: Sai Sandeep Chittiboina

Student ID: 001203994

Supervisor: Mohammad Al-Antary
Submission Date: 17-Jan-2023
Word count: 10386

Predicting Insights of Company Using Machine Learning & Deep Learning Techniques and Visualizing These Outcomes in Power BI

Computing & Mathematical Sciences, University of Greenwich, 30 Park Row, Greenwich, UK.

Abstract: Stock market prediction is the act of trying to determine the future value of a company's stock or the overall stock market. There are various methods and techniques that analysts, investors, and traders use to try and make predictions about the stock market, including technical analysis, fundamental analysis, and statistical modelling. However, it is important to note that stock market prediction is inherently uncertain and there is no surefire way to accurately predict the future performance of the stock market or individual stocks.

There are many factors that can influence the stock market, including economic conditions, market trends, company news, and global events. Analysts and investors often use a combination of these factors to try and make informed predictions about the direction of the stock market. However, the stock market is highly complex and dynamic, and it is often difficult to accurately predict its future movements.

Technical analysis involves studying the past price and volume data of a stock in order to identify patterns and trends that can be used to make predictions about future movements. Fundamental analysis involves evaluating a company's financial health and industry conditions to determine its intrinsic value and predict future stock price movements. Statistical modelling involves the use of statistical techniques and algorithms to analyse data and make predictions about future outcomes.

This article investigates the use of LSTM networks in that scenario to forecast future stock price trends based on price history, in conjunction with technical analysis indicators. To achieve this goal, a prediction model was created, and a series of experiments were run, with the results analysed against a variety of metrics to determine whether this type of algorithm outperforms other Machine Learning methods and investment strategies.

Keywords

Deep learning, Machine learning, Long short-term memory (LSTM), Recurrent neural network(RNN), and Stocks.

Acknowledgements:

I'd like to thank Mohammad Al-Antary for agreeing to be my supervisor and for his consistent advice, feedback, guidance, and support throughout the lifecycle of this MSc data science project.

Table of Contents

Introduction

Overview

The act of attempting to predict the future value of company stock or other financial instruments traded on a financial exchange is known as the stock market prediction. Machine learning is a subfield of artificial intelligence that entails using statistical models and algorithms to make data-driven predictions or decisions. Machine learning algorithms can be trained on historical data to predict the future performance of a stock in the context of stock market prediction.

There are numerous approaches to using machine learning for stock market prediction, and the approach chosen will be determined by the type of data available and the specific prediction task. Using supervised learning algorithms to predict stock price movements based on historical data, unsupervised learning algorithms to identify patterns in data, and reinforcement learning algorithms to learn trading strategies based on rewards and penalties are some common approaches.

Overall, the goal of using machine learning for stock market prediction is to create models that can analyse and interpret complex data and predict future performance accurately. Machine learning can be used to create models that can learn and adapt to changing market conditions, allowing them to make more accurate predictions than human analysts. It is

important to note, however, that stock market prediction is a difficult task, and no model can guarantee perfect accuracy.

Research aims and objectives

Aim of the research

To predict the future closing price of a company using the LSTM Model and give some insights into similar companies which are in the same industry.

Objectives of Research

Data collection and pre-processing: I need a dataset of historical stock prices as well as other relevant information like economic indicators and news articles. I also need to preprocess the data in order to train the model.

To collect and pre-process stock data using yfinance, you can use the following steps:

1. Install yfinance by running !pip install yfinance in your command line.
2. Import the yfinance library in your code.
3. Use the yfinance.download() function to download the historical data for a particular stock. You can specify the start and end date for the data and also the interval (daily, weekly, etc.).
4. Use the pandas library to preprocess the data. You can use functions such as .head() and .tail() to view the data, .describe() to get summary statistics, and .drop() to remove unnecessary columns.
5. we may also want to handle missing values by using functions such as .fillna() or .interpolate().
6. Save the preprocessed data to a file for future use.

It's important to note that yfinance is not an official API of Yahoo Finance; it's a third-party package that isn't guaranteed to work indefinitely, so you should be prepared to switch to another data source if necessary.

Choose a machine learning model: Choosing the best machine learning model for the stock prediction can be difficult, but there are some general guidelines to follow:

1. Understand the problem and the data: Before selecting a model, it is critical to understand the problem you are attempting to solve, the data you have at your disposal, and the performance metrics you wish to optimise.

2. Determine the type of issue: A stock prediction problem can be either a regression or a classification problem. The goal of regression is to predict a continuous value, such as the stock price, whereas the goal of classification is to predict a categorical value, such as whether the stock will rise or fall.

3. Consider the model's complexity: Simple models, such as linear regression, are easier to interpret and less computationally expensive, but they may be incapable of capturing complex relationships in data. Complex models, on the other hand, such as

deep neural networks, can be more powerful but are more difficult to interpret and require more data to train.

4. Consider the data's time-series nature: Stock prices are time-series data, which means they change over time. As a result, models such as ARIMA, LSTM, and others may be worth considering.

There are numerous machine learning models available for predicting stock prices. In this report, we are using LSTM.

Train the model: Several steps are involved in training an LSTM (Long Short-Term Memory) model for stock prediction:

1. Collect and preprocess the data: Gather historical stock data and preprocess it to make it ready for training. This could entail cleaning the data, dealing with missing values, and normalising the values.

2. Divide the data into two sets: training and testing: Separate the data into two sets: training and testing. The training set is used to train the model, and the testing set is used to assess its performance.

3. Prepare the data for the LSTM model as follows: The data for LSTM models must be in a specific format, typically a 3D array with the shape (samples, time steps, features). This can be accomplished by utilising the Keras function. preprocessing.sequence. TimeseriesGenerator

4. Define the model architecture: Create a model with the Keras library's LSTM layer. You can control the number of LSTM units, the activation function, the input shape, and whether or not the full sequence is returned. To prevent overfitting, you can also add additional layers such as a dropout layer.

5. Compile the model: Define the model's optimizer and loss function. The optimizer regulates the learning rate and the type of optimization algorithm used during training, whereas the loss function calculates the difference between predicted and actual values.

6. Fit the model: Call the fit() method to train the model on the training data. You can control the batch size, the number of epochs, and whether or not validation data is used.

7. Evaluate the model: Once the model has been trained, use the evaluate() method to assess its performance on the testing data. This will give you an idea of how well the model can predict data that has not yet been seen.

8. Fine-tune the model: Based on the evaluation results, you can improve the model's performance by adjusting the architecture, optimizer, or other parameters.

Examine model: A critical step in the model development process is evaluating the performance of a trained machine learning model. The following are some common methods for assessing the performance of a trained LSTM model for stock prediction:

- Mean Absolute Error (MAE): This is a measurement of how far predictions differ from actual values. The lower the MAE, the better the model's ability to predict correct values.

- Mean Squared Error (MSE): Like MAE, MSE penalises large errors more severely. The lower the MSE, the better the model's ability to predict correct values.

- R-squared: This is a metric that indicates how well the model fits the data. A value of 1 indicates that the model perfectly fits the data, while a value of 0 indicates that the model does not explain any of the variations in the data.

- The correlation coefficient (CC) measures the strength and direction of the relationship between predicted and actual values. A value of 1 represents a perfect positive correlation, while a value of -1 represents a perfect negative correlation.

- A confusion matrix is a table that is used to define a classification algorithm's performance. The number of true positives, true negatives, false positives, and false negatives is frequently used to predict a classification algorithm's performance.

- Precision, recall, and F1-score are metrics used to assess a classification model's performance. Precision is the proportion of correct positive predictions out of all positive observations, while recall is the proportion of correct positive predictions out of all positive observations, and F1-score is the harmonic mean of precision and recall.

Fine-tune the model: If the performance of a trained LSTM model for stock prediction is not satisfactory, the model can be fine-tuned in several ways:

1. Experiment with different architectures: Experimenting with different LSTM model architectures is one way to improve performance. Changing the number of LSTM units, adding or removing layers, and changing the activation functions are all examples of this.

2. Adjust the hyperparameters: The learning rate, batch size, and a number of epochs are all important parameters to consider. Experimenting with different values for these hyperparameters can aid in performance improvement.

3. Add more data: Having more data can often help to improve a model's performance. More historical stock data collected and used to train the model can help to improve performance.

4. Using ensemble methods entails combining the predictions of multiple models to make a final prediction. By reducing overfitting or bias, ensemble methods such as bagging or boosting can help to improve performance.

5. Feature engineering is the process of using domain knowledge of data to create features that allow machine learning algorithms to work. You can create new features or choose a more informative subset of the original features.

Use the model to make predictions: The following are the general steps for using a trained LSTM model to predict stock prices:

1. Prepare the new data: The new data should be in the same format as the training data. This may include data preprocessing, sequence creation, and value normalisation.

2. Make predictions: Call the trained model's predict() method and pass in the new data as an argument to make predictions on the new data. This will return the predicted values as an array.

3. Post-processing: Depending on the problem at hand, you might need to post-process the predicted values. For example, if the model was trained on normalised values, the predicted values may need to be denormalized before they can be interpreted.

4. After making predictions, it is critical to assess the model's performance on the new data. This can be accomplished by comparing predicted and actual values and calculating performance metrics such as MAE, MSE, or correlation coefficient.

5. Use the predictions: Once you're happy with the model's performance, you can use the predictions to make decisions. The forecasts can be used to inform buy or sell decisions, as well as to identify stock market trends.

Literature Review

Predicting the future value of a stock is difficult because it is affected by many factors, including the company's financial performance, the state of the economy, and market conditions. That being said, there are a few approaches you can use to try to predict a stock's future value:

Types of analysis

Fundamental analysis:

Fundamental analysis is a method of evaluating the intrinsic value of a company's stock by examining its financial and economic fundamentals. It is a way of attempting to determine the true value of a stock, independent of market fluctuations.

Some of the key factors that are considered in the fundamental analysis include:

- Financial statements: Analysts will review a company's balance sheet, income statement, and cash flow statement to get a sense of its financial health. This includes looking at metrics such as revenue, earnings, and debt levels.

- Management and leadership: Analysts will assess the quality of a company's management team and their track record of success.

- Competitive advantage: Analysts will look for factors that give a company an edge over its competitors, such as proprietary technology or a strong brand.

- Industry trends: Analysts will consider the overall health of the industry in which the company operates, as well as any trends that may impact its performance.

- Valuation metrics: Analysts will use various metrics to determine whether a stock is overvalued or undervalued based on its financial and economic fundamentals.

Analysts can make informed decisions about the intrinsic value of a stock and whether it is a good investment by taking these and other factors into account. It is important to note, however, that fundamental analysis is only one method of evaluating stocks, and there are no guarantees of accuracy.

Technical analysis:

Technical analysis is a method of evaluating securities that involve analysing market activity statistics such as past prices and volume. Market trends, as depicted by charts and other technical indicators, are thought to be predictive of future activity, according to technical analysts.

There are many different technical indicators that analysts may use in their analysis, including:

- Moving averages: A moving average is an average of a security's price over a certain number of periods. Moving averages can be used to identify trends and potential buy and sell signals.

- Oscillators: Oscillators are technical indicators that oscillate between two points, such as 0 and 100. They can be used to identify overbought and oversold conditions in the market.

- Trend lines: Trend lines are straight lines that can be drawn on a chart to connect a series of highs or lows. They can be used to identify uptrends and downtrends in a security's price.

- Candlestick patterns: Candlestick patterns are graphical representations of price data over a certain period of time. They can be used to identify potential reversal patterns in a security's price.

Overall, technical analysis is predicated on the idea that market trends, as depicted by charts and other technical indicators, can predict future activity. Technical analysts employ a variety of tools and techniques to identify patterns and make sound decisions about purchasing and selling securities. It is important to note, however, that technical analysis is only one method of evaluating stocks, and there are no guarantees of accuracy.

Economic analysis:
Economic analysis is a method of evaluating the performance of a company's stock by taking into account macroeconomic factors that may have an impact on the company's operations. Interest rates, inflation, unemployment, and economic growth are examples of such variables.

Some of the most important aspects of economic analysis are:

- Economic indicators: To assess the overall health of the economy, analysts will examine various economic indicators such as GDP, inflation, and employment data.

- Analysts will look at the overall health of the industry in which the company operates, as well as any trends that may have an impact on its performance.

- Company fundamentals: To determine a company's overall financial health, analysts will examine its financial statements, management team, and competitive advantage.

Analysts can make informed decisions about the economic factors that may impact a company's performance, as well as the potential risks and opportunities, by taking these and other factors into account. It is important to note, however, that economic analysis is only one method of evaluating stocks, and there are no guarantees of accuracy.

Time series data is data that is collected and recorded at regular intervals across time. It differs from standard data in several ways:

- Time-Dependency: Time series data is time-dependent, and the observations are typically ordered by time. This distinguishes time series data from other types of data, such as cross-sectional data, in which observations are collected at a single point in time.

- Time series data frequently exhibit patterns and trends that change over time. Seasonality, in which the data follows a consistent pattern over time, and trends, in which the data moves in a consistent direction over time, are examples of these patterns. Identifying and comprehending these patterns is a critical first step.

- Stationarity: When the mean and variance of a time series are constant over time, the data is said to be stationary. Most time series data is non-stationary, which means that the mean and variance change over time, making analysis and forecasting more difficult.

- Forecasting: Time series data is frequently used to predict future values based on previous observations. This is an important aspect of time series analysis, and forecasting techniques like ARIMA (AutoRegressive Integrated Moving Average) and Exponential Smoothing are widely used.

- Missing data points are common in time series data, and how to handle missing data points is an important aspect of time series analysis. Interpolation and forward or backward filling are two methods for dealing with missing data.

- Handling outliers: Because outliers can have a significant impact on time series data analysis, they must be handled with care. Outliers can be handled in a variety of ways, including Z-score, winsorizing, and so on.

- Time-based indexing: Time series data is typically indexed by time, allowing for simple manipulation and analysis of the data. This is distinct from other types of data that may be indexed by different variables.

For many years, machine learning has been used to predict stock market movements because it is an excellent tool for analysing and interpreting large and complex datasets. By analysing historical data with machine learning algorithms, it is possible to create models that can identify patterns and relationships in the data and predict future performance.

History of LSTM

The origins of LSTM can be traced back to the late 1980s and early 1990s when artificial intelligence researchers began to investigate the use of recurrent neural networks (RNNs) for time series prediction.

Elman introduced the first RNN model in 1990, which was a simple RNN architecture capable of storing a small amount of information over time. It was, however, incapable of dealing with long-term dependencies, which is a common issue in time series data.

In their paper "Long Short-Term Memory," Sepp Hochreiter and Jürgen Schmidhuber introduced the LSTM architecture in 1997. They proposed a new RNN architecture that could store information over longer time periods, making it more effective for dealing with long-term dependencies in time series data. The use of a "memory cell" that can retain information for an extended period of time is the key innovation of LSTM (Yanhui, 2021).

Neural network applications

LSTM has since been widely used in a variety of applications, including natural language processing, speech recognition, and time series forecasting. To improve the performance of time series analysis and prediction tasks, LSTM has been combined with other neural network architectures, such as convolutional neural networks (CNNs), in recent years.

Variants of LSTM, such as Bi-directional LSTM, Attention-based LSTM, and Gated Recurrent Unit (GRU), have been introduced in recent years and are used in applications such as speech recognition, natural language processing, and computer vision.

The development of expert systems, which were designed to replicate the decision-making process of human analysts, was one of the early applications of machine learning in stock market prediction. To analyse data and make predictions, these systems used rules-based systems and decision trees.

To make more sophisticated predictions, machine learning algorithms such as artificial neural networks and support vector machines have recently been used. In many cases, these algorithms can learn and adapt to changing market conditions, and they can make more accurate predictions than expert systems or human analysts.

The increasing availability of data and the development of more advanced algorithms capable of analysing and interpreting that data have fuelled the use of machine learning in stock market prediction. Machine learning can be used to create models that can learn and adapt to changing market conditions, allowing them to make more accurate predictions than human analysts.

A recurrent neural network (RNN) that is especially good at predicting time series data. Time series data is a collection of data points over time that is commonly used in fields such as finance, economics, and weather forecasting.

The ability of LSTMs to remember information for long periods of time is one of their key characteristics, making them well-suited to modelling long-term dependencies in data. This means that an LSTM can use information from earlier in the time series to make more accurate predictions about future values in the context of time series prediction.

For example, if a time series represents a company's stock price, an LSTM could use information about the company's financial performance, industry trends, and other factors to predict future stock prices more accurately.

Because they can remember information for long periods of time and use that information to make informed predictions about future values, LSTMs are a powerful tool for predicting time series data. They have been used successfully in a variety of time series prediction tasks, including stock price prediction, sales data prediction, and weather pattern prediction.

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) that is able to capture long-term dependencies in data. It does this by using memory cells, which are able to retain information for long periods of time, and gates, which control the flow of information into and out of the memory cells (Ackerson, 2021).

Types of gates in an LSTM network

There are three types of gates in an LSTM network: input, output, and forget gates. The input gate controls data flow into the memory cell, the forget gate controls data flow out of the memory cell, and the output gate controls data flow from the memory cell to the LSTM output. The gates are activated by sigmoid activation functions, and their output is multiplied element-wise with the memory cell's input or output to produce the final output.

LSTM networks have been used in a variety of applications, including natural language processing, speech recognition, and time series forecasting. They are able to handle input sequences of variable length and are able to learn long-term dependencies in the data, making them well-suited for tasks that require modelling long-term dependencies.

Here is a brief overview of how an LSTM model works:

1. The input sequence is fed into the model one element at a time.

2. At each time step, the LSTM model processes the current input element and the previous internal state and updates the internal state.

3. The updated internal state is then used to predict the next element in the sequence.

4. This process is repeated for each element in the input sequence.

LSTMs control the flow of information through a network by using a series of gates. These gates are intended to allow some data to pass through while blocking or forgetting other data. The input gate, forget gate, and output gate are the three main types of gates in an LSTM.

1. The input gate controls which parts of the LSTM's input are passed on to the internal state.

2. The forget gate specifies which parts of the previous internal state should be forgotten and not carried forward to the next time step.

3. The output gate specifies which parts of the internal state should be used to generate the LSTM output (Y. Wang, 2019).

Sequential LSTM model

A sequential LSTM model processes input data in a linear fashion, one-time step at a time. The LSTM model processes the input data and updates the hidden state, which contains information from previous time steps, at each time step. After that, the hidden state is used to predict the output at the current time step.

The input gate controls what information gets passed to the internal state of the LSTM cell. The forget gate controls what information gets discarded from the internal state. And the output gate controls what information gets output from the LSTM cell and passed on to the next time step.

Advantages & Disadvantages of LSTM

Advantages of LSTM (Long Short-Term Memory) model:

- LSTMs can handle and remember information in sequential data for longer periods of time, making them well-suited for tasks like speech recognition and natural language processing.

- Because LSTMs can selectively remember and forget information, they can deal with the problem of vanishing gradients that can occur in traditional RNNs.

Disadvantages of LSTM:

- LSTMs are computationally expensive, which can make them less practical for certain applications where resources are limited.

- LSTMs can be difficult to train and fine-tune, which can make them more challenging to use for some tasks.

- LSTMs are also complex in nature and may require a lot of data to train.

Anyhow there are multiple neural networks Artificial Neural Networks (ANNs), Recurrent Neural Networks (RNNs), and Convolutional Neural Networks (CNNs) are all types of neural networks, but they are designed for different types of tasks and have different architectures.

- ANNs are a type of feedforward neural network, where information flows in only one direction, from input to output. They are commonly used for tasks such as image classification and pattern recognition.

- RNNs are a type of neural network that can process sequential data, such as time series data or natural language text. They have feedback connections, which allow them to maintain a hidden state that can remember information from previous time steps. RNNs are typically used for tasks such as speech recognition and natural language processing.

- CNNs are a type of neural network that is specifically designed to process data with a grid-like topology, such as an image. They are composed of multiple layers of filters that are used to extract features from the input data. CNNs are commonly used for tasks such as image classification, object detection, and image segmentation.

In summary, ANNs are feedforward and are good for simple pattern recognition, RNNs are designed for sequential data, and CNNs are designed for image data, they are good for image classification, object detection and image segmentation task (Tondak, 2022).

Difference between (RNN) and (LSTM)

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) that can detect long-term dependencies in data. RNNs are neural networks that process time series or natural language. Traditional RNNs, on the other hand, have difficulty dealing with long-term dependencies because the gradients propagated through the network can either vanish or explode. This problem is solved by LSTMs, which include a memory cell, gates that control the flow of information into and out of the cell, and a hidden state that is passed from one-time step to the next. These mechanisms enable LSTMs to selectively retain or discard information from the past, allowing them to handle long-term dependencies more effectively.

This is designed to handle and remember information in sequential data for longer periods of time. RNNs are neural networks with feedback connections that enable them to maintain a hidden state capable of remembering data from previous time steps. As a result, they are well-suited for tasks like speech recognition and natural language processing.

An LSTM network is made up of a series of LSTM cells connected in a chain-like structure. There are three gates in each LSTM cell: the input gate, the forget gate, and the output gate. These gates regulate the flow of information into and out of the cell, as well as which information should be remembered and which should be forgotten.

The input gate, which is determined by the current input and the previous hidden state, controls the flow of new information into the cell. The forget gate, which is determined by the current input and the previous hidden state, controls the flow of information out of the cell. The output gate, which is determined by the current input, the previous hidden state, and

the current cell state, controls the flow of information out of the cell (Recurrent Neural Networks (RNN) and LSTM: Overview and Uses, n.d.).

Each LSTM cell has a cell state, which is a vector that holds the information that the cell is currently "remembering" in addition to the gates. At each time step, the cell state is updated based on the input, forget, and output gates.

In summary, LSTMs are a type of RNN that is designed to handle and remember information in sequential data for longer periods of time. The cell, which contains the gates that control the flow of information into and out of the cell, and the cell state, which holds the information that the cell is currently "remembering," are the main features of LSTMs. The gates and cell state in LSTM allow it to selectively remember and forget information, which aids in overcoming the vanishing gradients problem that can occur in traditional RNNs.

Experimental analysis

I have used "`-q yfinance`" to import the historical data of the stocks and here I have taken four motor companies for analysis and predictions namely

1. Tesla
2. Ford Motor
3. General Motors
4. Toyota Motor

yfinance is a Python library that allows us to get stock market data using the Yahoo Finance API. It allows us to download data for a variety of financial instruments, including stocks, options, currencies, and commodities, and to retrieve information about financial events such as earnings and dividends. We can use yfinance to access real-time and historical data, and to perform various types of analysis on the data (Aroussi, 2019). For example, we can use it to get the current price of a stock, to retrieve a time series of stock prices

Fig 1 is a data frame which contains all four companies' historical stock data, in comma-separated values indicating their open, high, low, close, adj close and volume.

Date	Open	High	Low	Close	Adj Close	Volume	company_name
2022-12-20	138.889999	140.240005	138.770004	138.830002	138.830002	178300	Toyota Motor
2022-12-21	137.110001	137.809998	136.690002	137.460007	137.460007	238600	Toyota Motor
2022-12-22	138.279999	138.429993	135.880005	137.000000	137.000000	211600	Toyota Motor
2022-12-23	137.039993	137.759995	136.190002	137.139999	137.139999	149400	Toyota Motor
2022-12-27	136.119995	136.970001	135.839996	136.160004	136.160004	137600	Toyota Motor
2022-12-28	136.000000	136.770004	133.899994	134.130005	134.130005	236900	Toyota Motor
2022-12-29	135.940002	137.490005	135.910004	137.410004	137.410004	249400	Toyota Motor
2022-12-30	136.880005	137.399994	136.000000	136.580002	136.580002	168700	Toyota Motor
2023-01-03	137.960007	138.470001	137.020004	138.279999	138.279999	309800	Toyota Motor
2023-01-04	138.210007	138.399994	136.649994	137.190002	137.190002	467100	Toyota Motor

Figure 1:Data frame of four automobile companies

The describe() function can help us quickly understand the distribution of a dataset and identify outliers or anomalies in the data. It can also be used to generate plots or charts that visualise the distribution of the data, and it will return a new data frame with various statistical measures for each numeric column, such as the count, mean, standard deviation, minimum, maximum, and quartiles.

Here we are performing this function on tesla stocks to see the insights of the company

```
TSLA.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	251.000000	251.000000	251.000000	251.000000	251.000000	2.510000e+02
mean	261.937941	268.298261	254.352497	260.856494	260.856494	8.775980e+07
std	58.279055	59.316863	57.105009	58.226460	58.226460	3.138969e+07
min	109.110001	114.589996	104.639999	108.099998	108.099998	4.186470e+07
25%	223.969994	229.638336	217.514999	223.771667	223.771667	6.685095e+07
50%	272.216675	278.433319	264.003326	269.956665	269.956665	8.203590e+07
75%	301.449997	308.964996	294.361664	301.841660	301.841660	9.778950e+07
max	382.216675	390.113342	362.433319	381.816681	381.816681	2.314028e+08

Figure 2:describe of tesla stocks

The info() function can be useful for quickly understanding the structure and content of a dataset, as well as identifying potential issues such as missing or misformatted data. It can

also be useful for debugging, as it can help you understand why certain operations or functions are not working as expected.

As we can clearly say that the data here don't have any null values and all the data in the row are in float format and the column which is in the object is the one in which we have added the company names.

```
TSLA.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 251 entries, 2022-01-05 to 2023-01-04
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Open          251 non-null    float64
 1   High          251 non-null    float64
 2   Low           251 non-null    float64
 3   Close         251 non-null    float64
 4   Adj Close     251 non-null    float64
 5   Volume        251 non-null    int64
 6   company_name  251 non-null    object
dtypes: float64(5), int64(1), object(1)
memory usage: 15.7+ KB
```

Figure 3:Info of tesla stock

A stock's closing price is its price at the end of the trading day after the market has closed. It is the final price at which the stock trades on that specific day. The closing price is significant because it reflects the market's overall sentiment toward the stock at the end of the trading day. It is used in conjunction with the opening price to calculate a stock's daily price range and daily price change. This is the plot for the closing prices of the four motor companies which have taken looking at the graph we can say that all the stocks have significantly low by the end of the year when compared to their price at the beginning of the year.

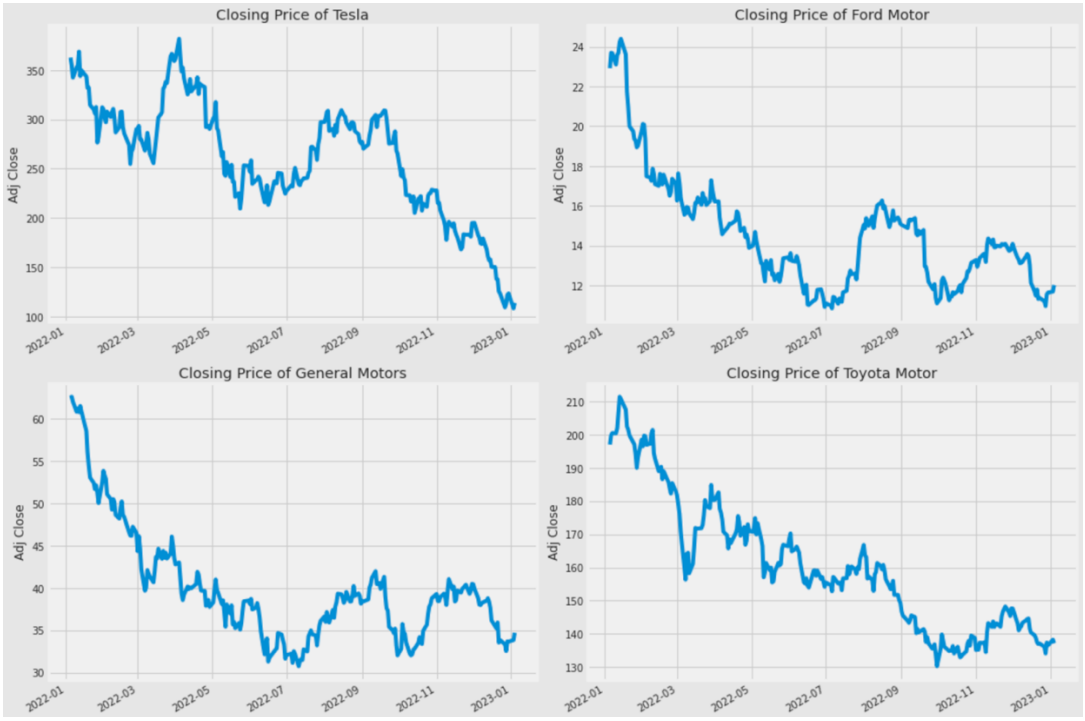

Figure 4:Closing price plot

Volume is the number of shares of a particular stock that are traded during a particular time period, usually a day. It is an important indicator of the level of activity in stock and can be used to identify trends and patterns in the market. High volume typically indicates a strong interest in a stock, while low volume may suggest a lack of interest.

Volume can be used in conjunction with price to identify potential trading opportunities. For example, if a stock has a high volume and the price is also rising, it may be a good time to buy the stock because there is strong demand for it. On the other hand, if a stock has a low volume and the price is falling, it may be a good time to sell the stock because there is weak demand for it.

Volume is also used to confirm trends and patterns in the market. For example, if a stock is in an uptrend and the volume is increasing, it may be a sign that the trend is likely to continue. On the other hand, if a stock is in a downtrend and the volume is decreasing, it may be a sign that the trend is starting to weaken.

Looking at fig 5 we can clearly say that the volume of the stocks in tesla company is significantly high when compares to the other stocks, so we can say that the stock has a huge demand in the market when compared to others

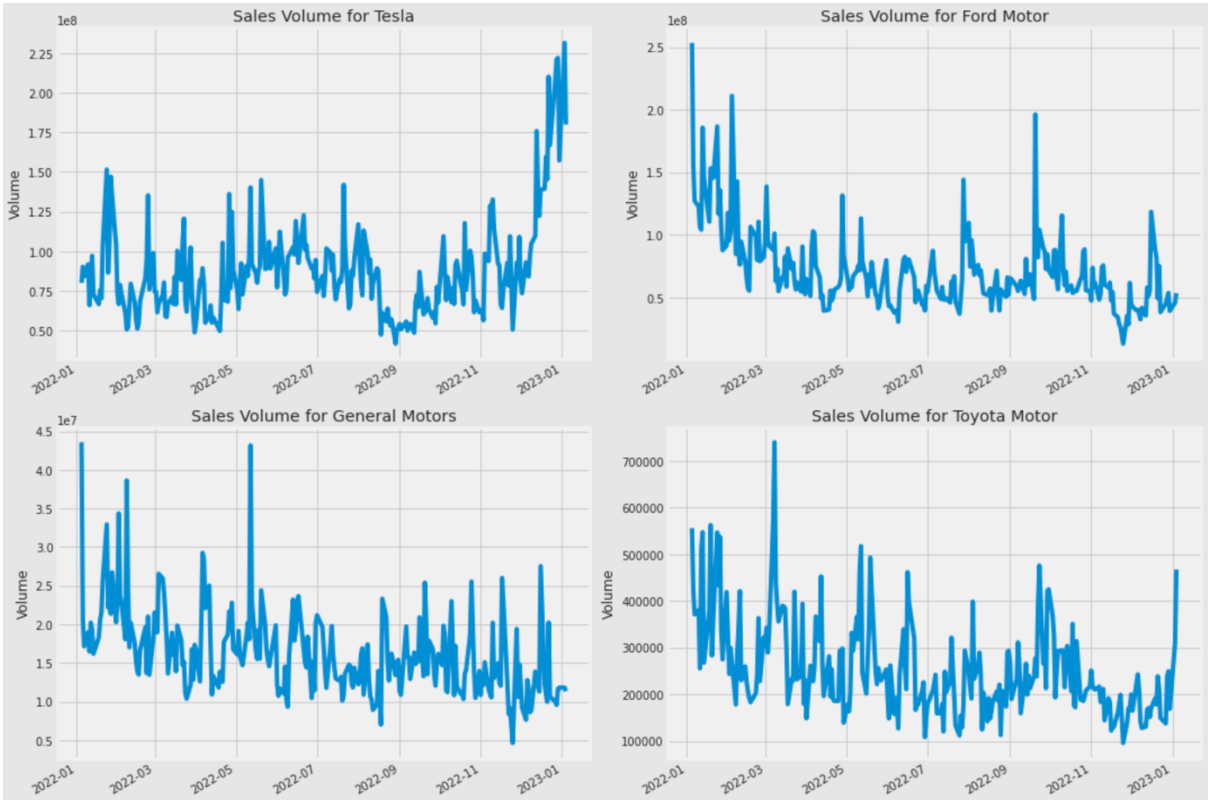

Figure 5:Scales volume Plot

A moving average is a technical analysis tool that helps identify trends by smoothing out price data. It is calculated by taking the average of a set number of price periods (days, weeks, months, etc.) and plotting it on a chart.

Moving averages are classified into three types: simple moving averages (SMAs), exponential moving averages (EMAs), and weighted moving averages (WMAs). Each type is calculated differently and may be better suited to various types of analysis.

Moving averages can be useful for identifying price trends as well as potential support and resistance levels in stock. For example, if the price of a stock is higher than its moving average, it may be in an uptrend; if the price is lower than its moving average, it may be in a downtrend. Similarly, if the stock's price is approaching a moving average from below, it may serve as a support level, while it may serve as a resistance level if the price is approaching it from above.

Moving averages can be used alone or in conjunction with other technical analysis tools to assist traders in making informed decisions about whether to buy or sell a stock.

$$\text{Simple moving average} = \frac{a_1+a_2+a_3+\cdots+a_n}{N}$$

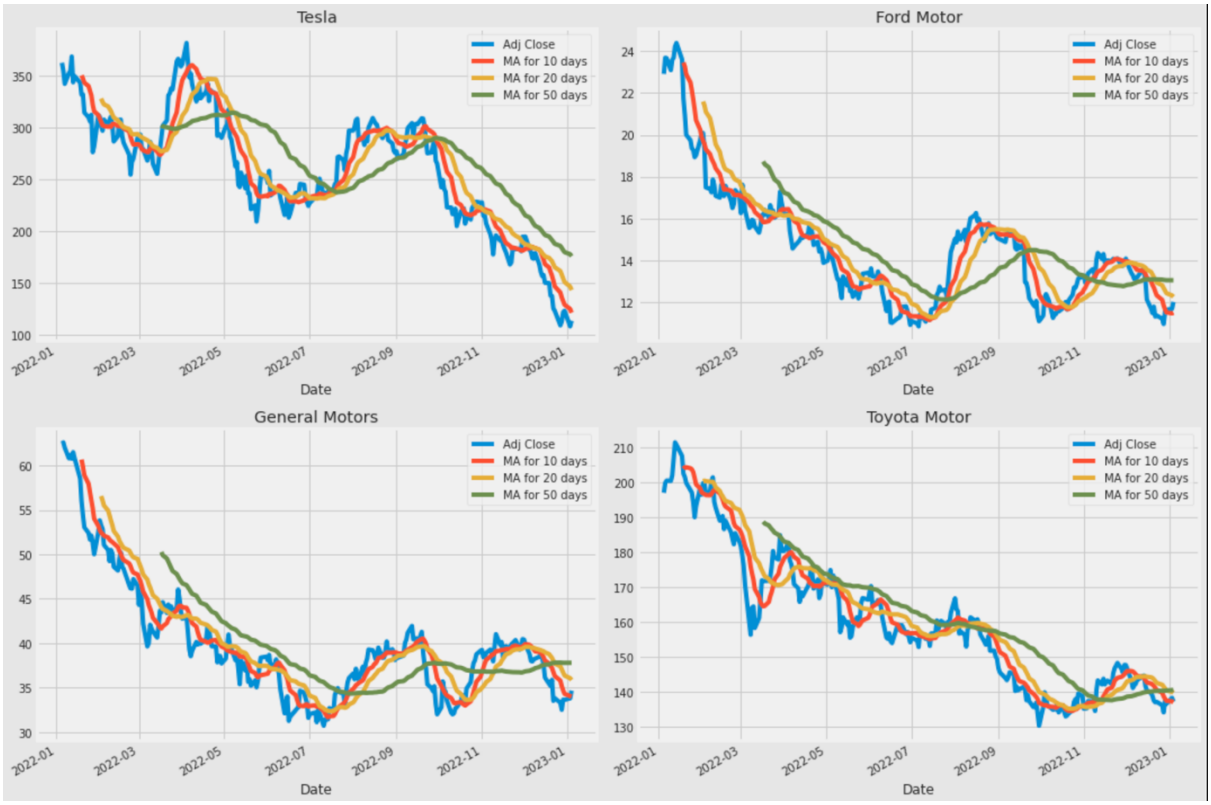

Figure 6:Moving Average of ADJ Close for 10,20,30 days

The graph(fig:6) shows that the best values for measuring the moving average are 10 and 20 days because we can still capture trends in the data without the time this can lead.

Now that we've completed some preliminary research, let's delve a little deeper. We're now going to look at the stock's risk. To do so, we'll need to look at the stock's daily movements rather than just its absolute value. Let's go ahead and use pandas to get the stock's daily returns. To calculate the daily return of the stock we need to use the pct_change() function on the adjusted close price to see the returns.

```
for company in company_list:
    company['Daily Return'] = company['Adj Close'].pct_change()
```

pct_change() is a function that can be used to calculate the percentage change between the current element and the previous element in a Pandas series or Data Frame. Because there is no previous element to compare it to, the first element in the output series is NaN. The remaining elements are the percentage differences between each element and the one before it. The pct_change() function can be used to analyse data trends and identify significant changes. It is important to note, however, that the percentage change only reflects the difference between two adjacent elements and does not account for the overall trend or pattern in the data.

The price change of a stock over a single day is measured by daily returns on stocks. They can be useful for a variety of things, including:

Analysis of stock performance: Daily returns can be used to determine how well a stock performs on a daily basis. A stock, for example, that has consistently high daily returns over time may be considered a good investment.

Identifying trends: Daily returns can aid in the identification of stock market trends. For example, if a stock's daily returns are consistently positive over a period of time, it may indicate an upward trend.

Risk management: Daily returns can be used to assess portfolio risk. A portfolio with high daily returns, for example, may be considered riskier than a portfolio with low daily returns.

Calculating total return: A stock's or portfolio's total return is the sum of price appreciation and dividends received over a given time period. The total return of a stock or portfolio can be calculated using daily returns.

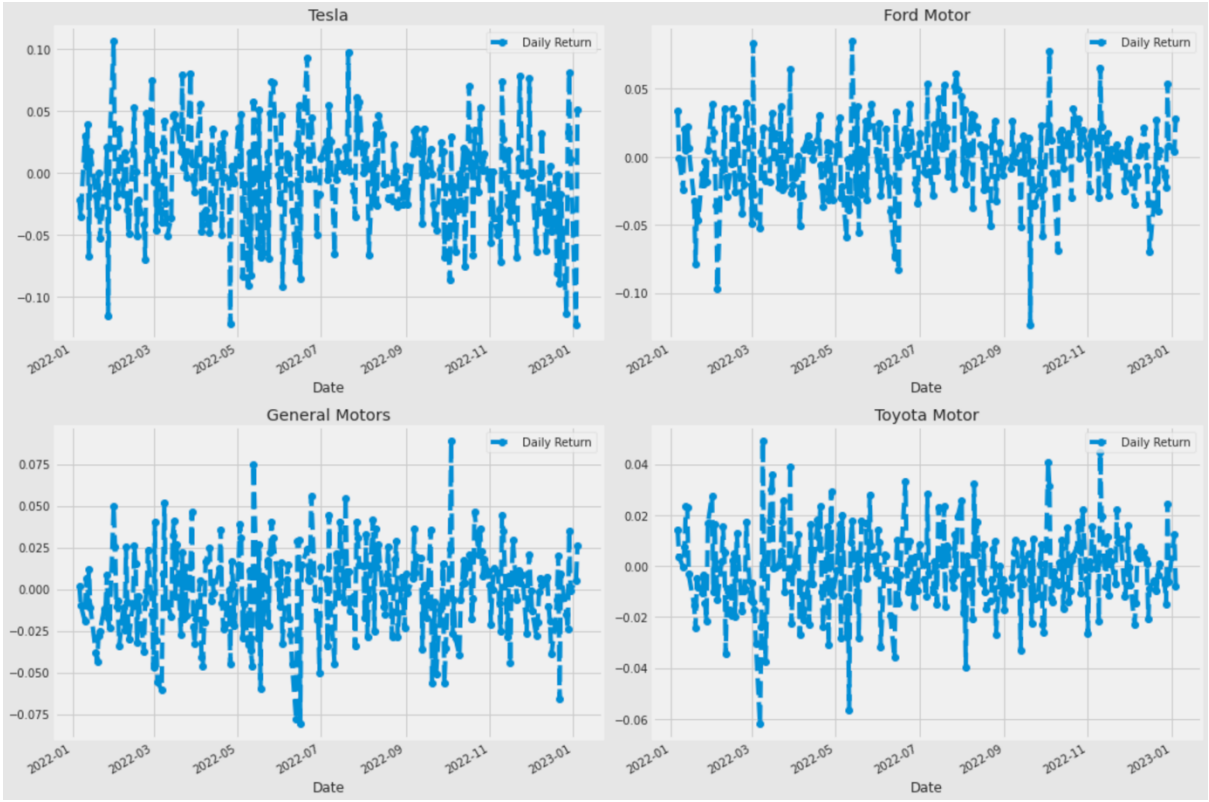

Figure 7:Daily returns the plot

It's so evident that even in the daily returns plot the stocks of tesla stocks outperformed when compared to Ford Motor and Toyota Motor

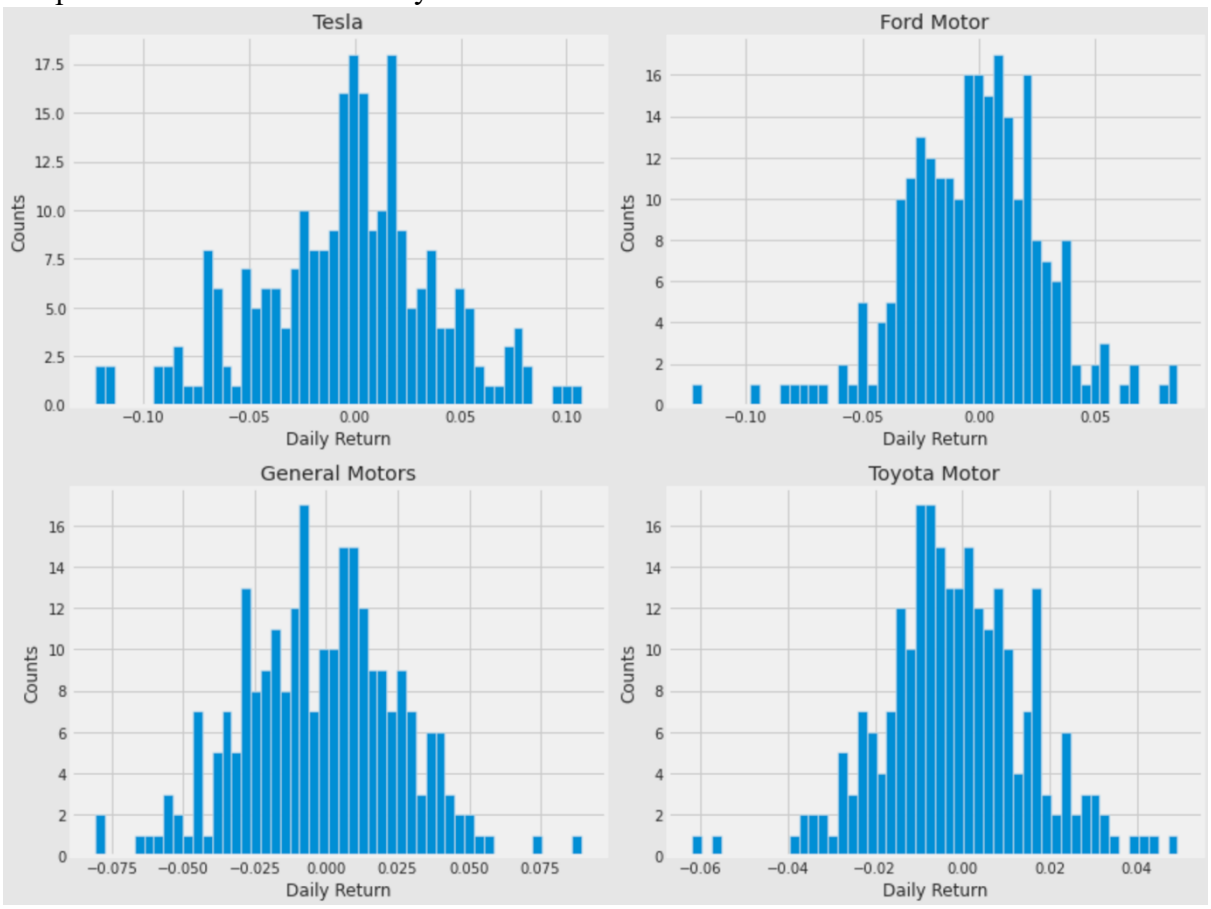

Figure 8: Average Daily returns histogram

In stock analysis, histograms can be used to visualise the distribution of certain financial metrics for a specific stock or portfolio. A histogram, for example, could be used to show the frequency of returns for a specific stock over time. This could assist an investor in understanding the stock's risk profile and making informed decisions about whether to buy, sell, or hold the stock.

Volatility, trading volume, and various financial ratios such as price-to-earnings ratio or price-to-book ratio could also be plotted on a histogram for stock analysis. By examining the distribution of these metrics, investors can gain a better understanding of a stock's or portfolio's underlying financial health and risk profile.

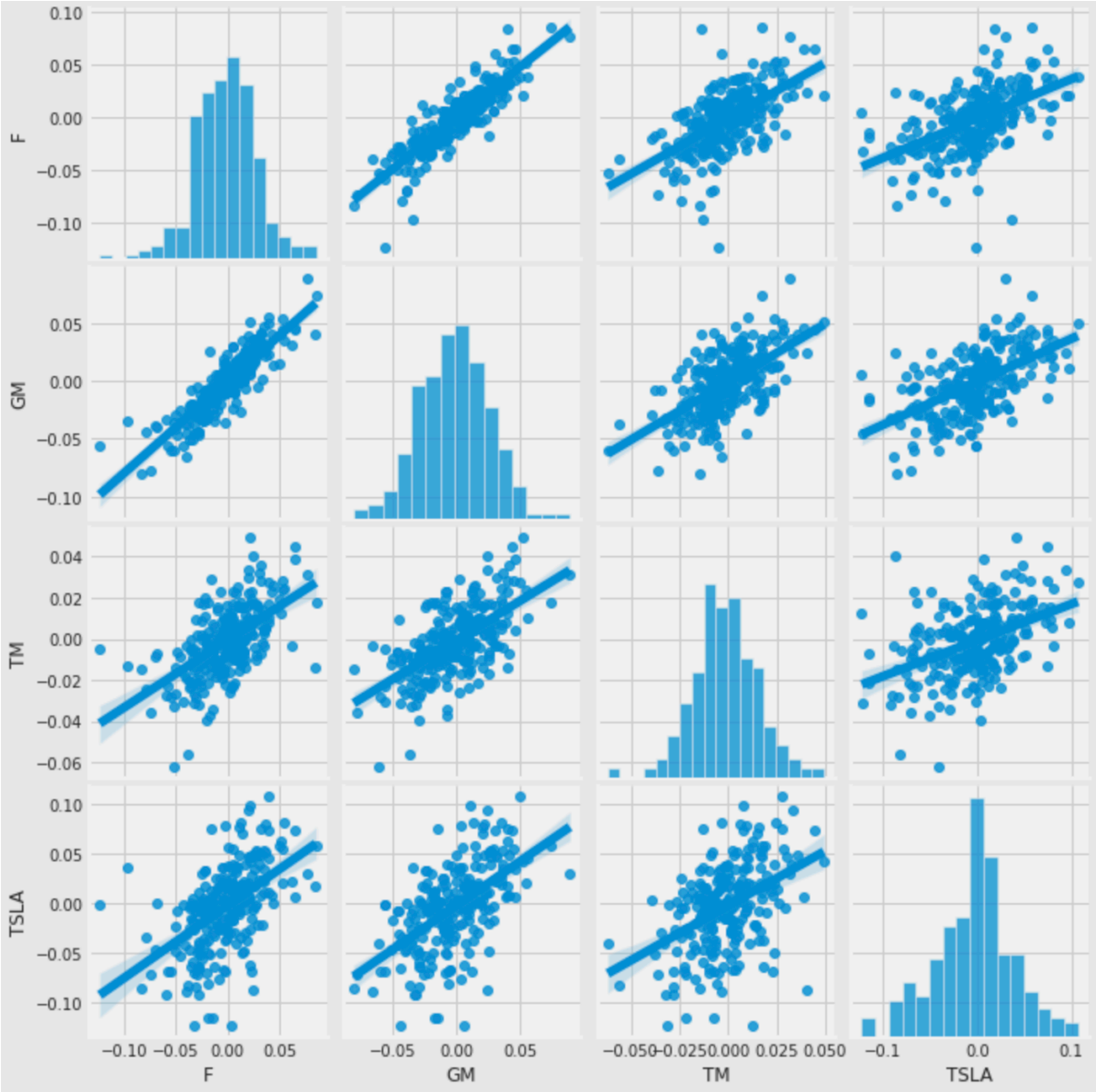

Figure 9:Pair plot of our Data frame(1)

This is a plot that allows you to visualise the relationships between variables in a dataset. It is especially useful for datasets with multiple variables because it allows you to quickly see how the variables are related to one another.

To create a pair plot in seaborn, you need to import the seaborn library and use the sns.pairplot() function, passing in a Data Frame as the data argument. You can also specify which variables you want to include in the plot by passing them in as the x_vars and y_vars arguments.

Seaborn will generate scatterplots for each pair of variables and histograms for each individual variable by default. The plot's appearance can be customised by adjusting parameters such as the colour palette, marker size, and transparency.

Pair plots are a useful tool for investigating variable relationships and identifying patterns in a dataset. They can also be used to create visualisations for reports or presentations.

Above, we can see all of the daily return relationships between all of the stocks. A quick glance reveals an intriguing correlation between Ford and GM daily returns. It could be interesting to look into that individual comparison.

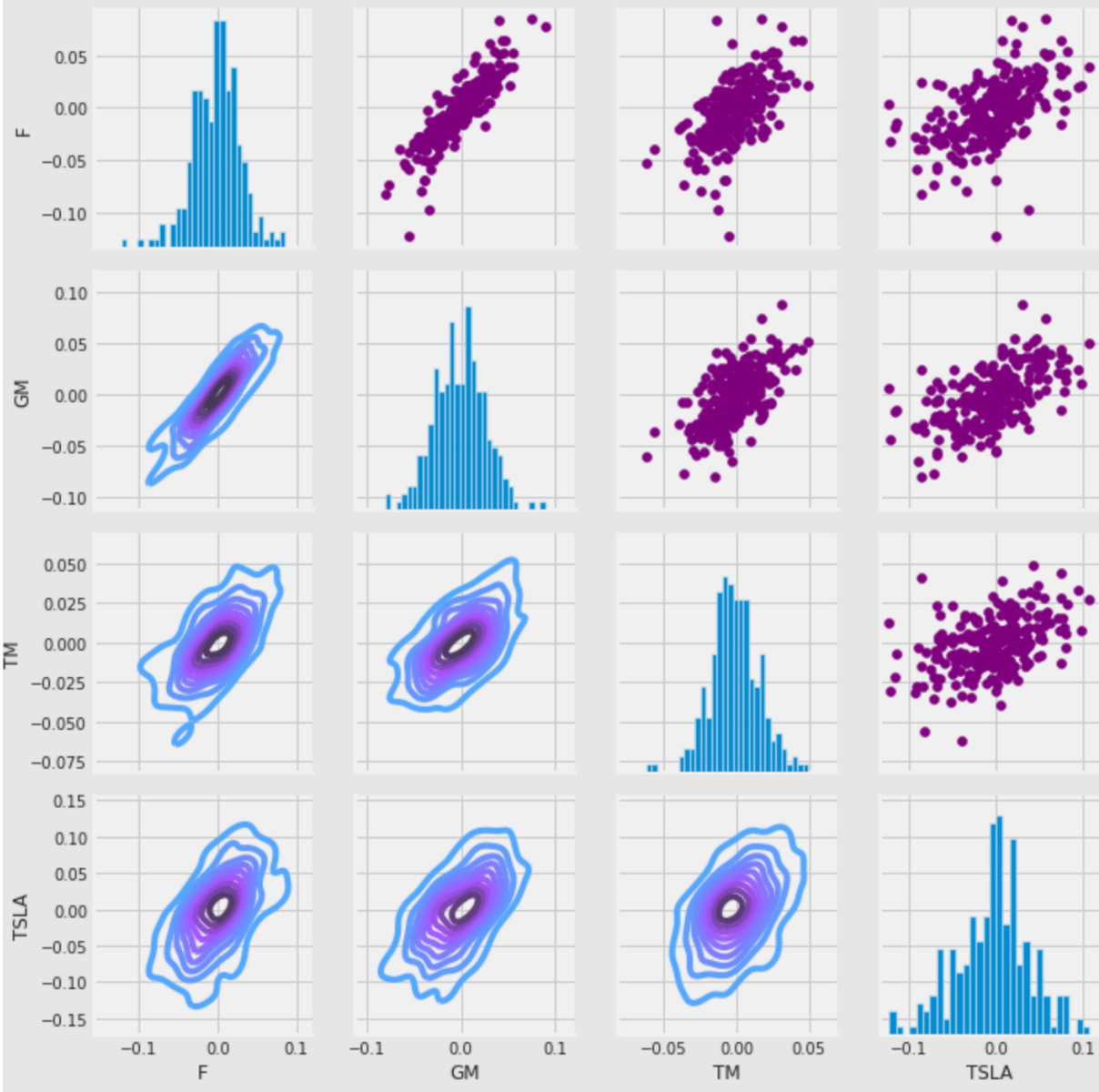

Figure 10:Pair plot of our Data frame(2)

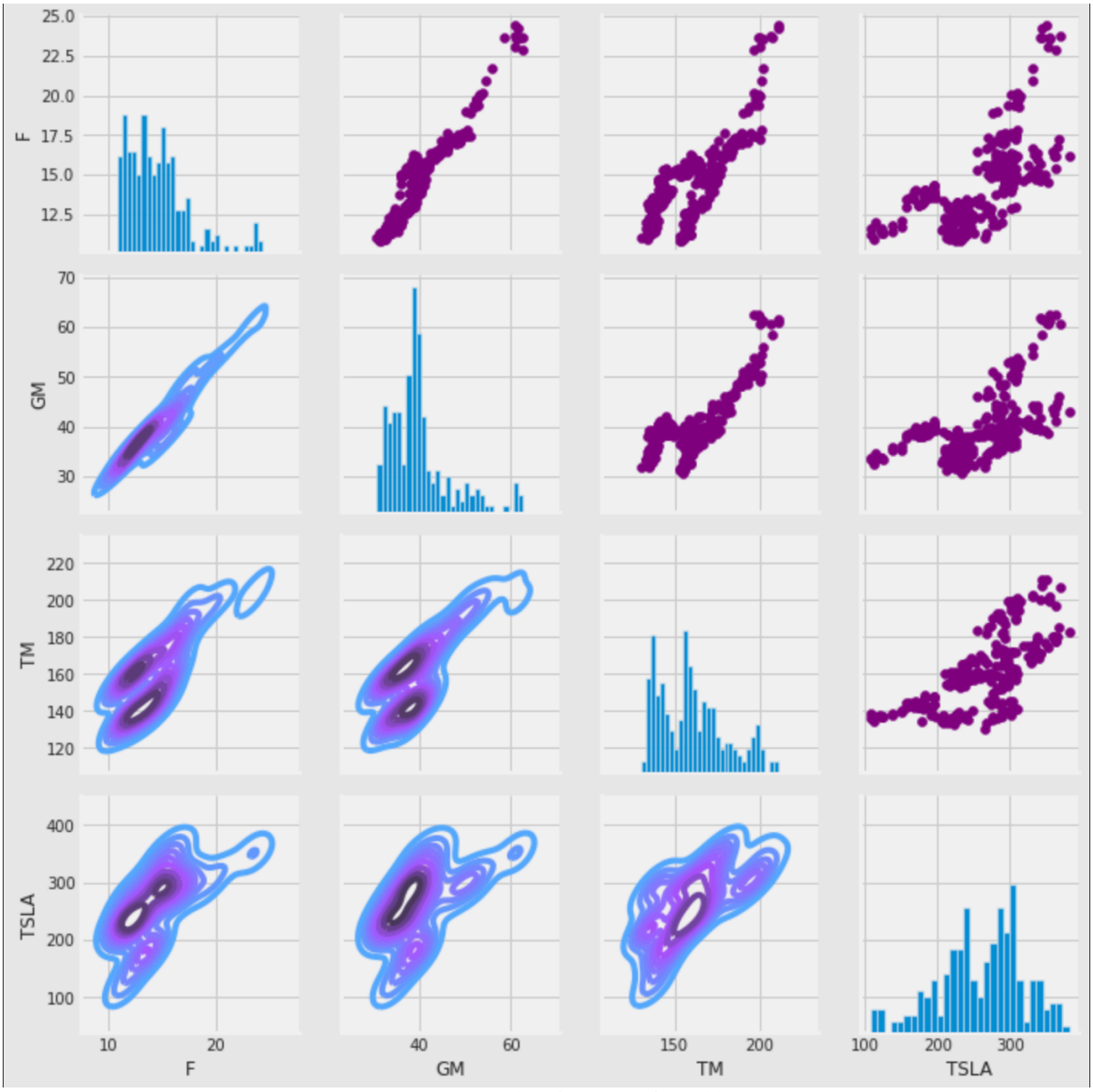

Figure 11:Pair plot of our Data frame(3)

Finally, we could perform a correlation plot to obtain actual numerical values for the correlation between the daily return values of the stocks. We can see an interesting relationship between Ford and General Motors by comparing closing prices.

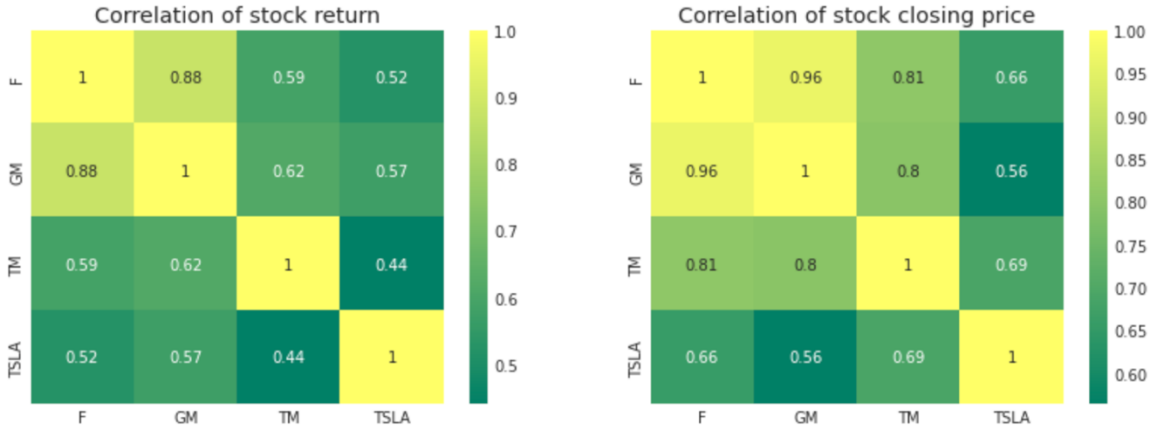

Figure 12:correlation between companies' returns and closing price

Even in fig 12, the correlation between ford and general motors seems so interesting with a 96% of a positive relation between these two companies in their closing prices and 88% in returns to the companies, It's also worth noting that all of the automobile companies are positively correlated.

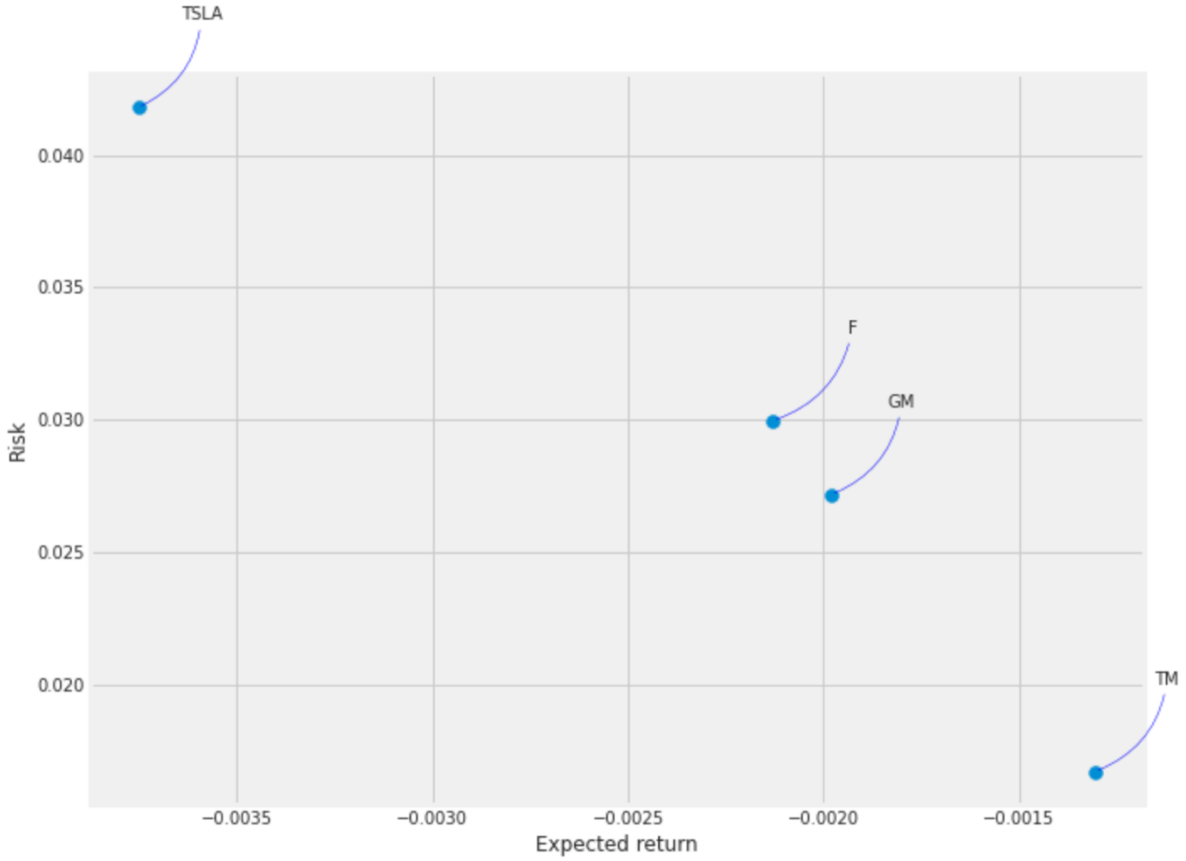

Figure 13:Risk & Expected returns plot

In fig 13 the x-axis indicates the expected returns and the y-axis indicates the risk involved in investing in those companies, after observing the plot it's so clear that Toyota motors have the least amount of risk with the high amount in returns and the least recommended stock according to the plot was tesla with very high risk and very low return price.

Figure 14:Closing price plot of Toyota motor

As fig 13 clearly states that the investment in the stock at Toyota motors has less risk, so in this report, we are choosing, the same stock to train our LSTM model for our future closing price prediction. Toyota Motor Corporation is a Japanese multinational corporation that

designs, manufactures, and sells vehicles and vehicle parts. Toyota is well-known for making popular automobiles such as the Toyota Camry, Toyota Corolla, and Toyota RAV4. Toyota also manufactures luxury automobiles under the Lexus brand. Aside from automobiles, the company also manufactures commercial vehicles, buses, and engines. Toyota's stock is symbol TM on the Tokyo Stock Exchange and the New York Stock Exchange.

There are several ways to split time series stock data into training and testing sets, some of the common ways are:

Random Split: This method randomly splits the data into training and testing sets. This method works well when the data is not time-dependent.

Time-Based Split: This method splits the data based on a specific time period, such as the last year's data for testing and the rest for training. This method is useful when the data is time-dependent and you want the test set to include more recent and relevant data.

Rolling Window: This method splits the data into chunks of a fixed-size window, such as the last 60 days for testing and the rest for training. This method is useful when the data is time-dependent and you want to test the model's ability to predict future values based on recent history.

Expanding Window: This method starts with a small window of data for testing and gradually increases the size of the window as the training set grows. This method is useful for testing the model's ability to adapt to different time periods.

It is important to note that the most appropriate method to split the data depends on the specific characteristics of the dataset and the goal of the analysis dependencies.

```
training_data_len
```

```
241
```

Figure 15 : Length of training data

As the length of our training model is very low (241) our model may experience low accuracy because In general, the more data the model has been trained on, the better it is able to generalise to new situations and predict accurately. However, this does not preclude an LSTM model from performing well with fewer data.

A few factors can affect the performance of an LSTM model with fewer data:

1. The model's complexity: A more complex LSTM model with a larger number of parameters may necessitate more data to avoid overfitting.

2. The data quality: If the data is of high quality and relevant to the task at hand, a smaller amount of data may be enough to train a good model.

3. Strong patterns in the data: If the data contains strong patterns, a smaller amount of data may be enough to train a good model.

Overall, the performance of an LSTM model with fewer data will be determined by the characteristics of the dataset and the task at hand. It may be possible to obtain good results with fewer data, but this will be dependent on the aforementioned factors and may necessitate some experimentation to determine the optimal model configuration.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 128)           66560

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 25)                1625

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 117,619
Trainable params: 117,619
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 16:Summary of the model

A Sequential LSTM model is a type of LSTM (Long Short-Term Memory) model created in Python with the Keras library. The Sequential model is a linear stack of layers that can be added one at a time. The layer added in the case of LSTM is the LSTM layer.

Sequential LSTM models can be applied to a variety of tasks, including time series forecasting, natural language processing, and speech recognition. The sequential LSTM model in time series forecasting takes in a sequence of historical data and attempts to predict the next value of the time series. The sequential LSTM model can be fine-tuned further by adding more layers, changing the number of neurons in each layer, and adjusting the model hyperparameters.

To build a Sequential LSTM model in Keras, begin by importing the necessary libraries and creating a new Sequential model. Then, using the LSTM() function, you would add an LSTM layer to the model, specifying the number of neurons and other parameters as needed. Finally, any additional layers, such as a dense layer, would be added, and the model would be compiled with a loss function and optimizer before being trained on your data.

A dense layer in a Sequential LSTM model is a fully connected layer of neurons in a neural network. That is, each neuron in the dense layer is linked to every neuron in the previous layer. The dense layer uses the output of the LSTM layer to perform the final classification or regression task.

In the case of LSTM, a dense layer takes the previous layer's output, which is a 3D tensor (batch size, timesteps, features), and applies a linear transformation to it using a set of weights and biases. The dense layer can have a single output node and be used for binary classification, or it can have multiple output nodes and be used for multi-class classification.

The dense layer can be added to the LSTM model by using the Keras library's Dense() function and specifying the number of neurons and activation function. For instance, suppose you want to build a dense layer with 32 neurons and a ReLU activation function.

In summary, a dense layer in an LSTM Sequential model is a fully connected layer added at the end of the model to perform the final classification or regression task on the LSTM layer's output. It uses a set of weights and biases to apply a linear transformation to the output of the LSTM layer.

In our model, we are using the optimizer Adam (Adaptive Moment Estimation) is a training optimization algorithm that is used to update the parameters of a neural network. It is a stochastic gradient descent (SGD) algorithm that combines the benefits of two other SGD algorithms, AdaGrad and RMSProp.

Adam employs a combination of AdaGrad and RMSProp's learning rate and adaptive learning rate methods. AdaGrad employs a different learning rate for each weight and adapts the learning rate to the parameters, resulting in a good performance on problems involving sparse gradients. RMSProp employs the moving average of the squared gradient rather than the gradient itself, which aids in the prevention of oscillations.
Adam also keeps an exponentially decaying average of previously squared gradients (similar to RMSProp) and an exponentially decaying average of previous gradients (similar to momentum).

The main advantage of using Adam is that it can automatically adjust the learning rate, which can help to speed up and stabilise the training process. It is widely regarded as a robust optimization algorithm capable of performing a variety of deep learning tasks such as image classification, natural language processing, and time series forecasting.

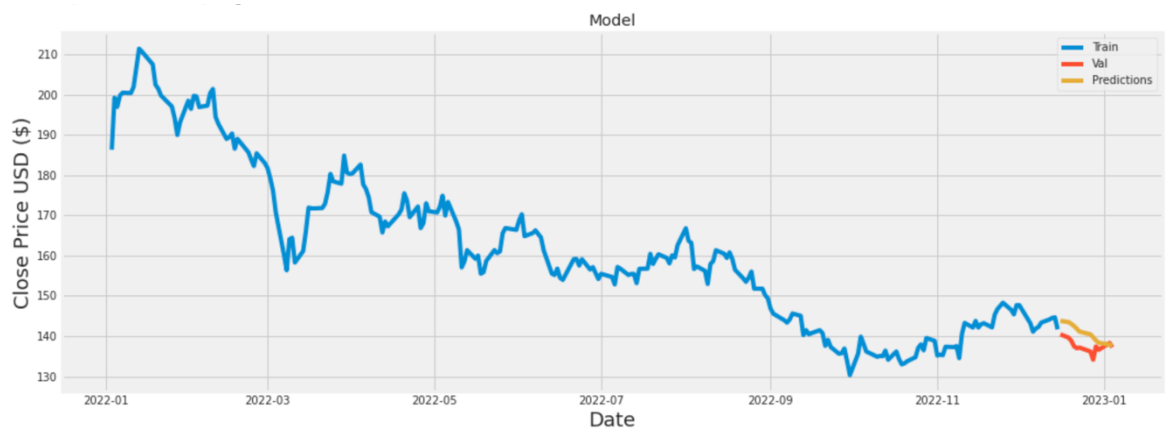

Figure 17:Closing price prediction of Toyota motors using LSTM-Model

The Root Mean Squared Error (RMSE) is a commonly used metric in LSTM (Long Short-Term Memory) models to evaluate model performance. It computes the difference between predicted and actual values.

The square root of the mean of the squared differences between the predicted and true values is used to calculate the RMSE. The lower the RMSE value, the better the model's performance. An RMSE of zero indicates that the model can predict the values perfectly.

In time series forecasting, the RMSE is used to assess the accuracy of the LSTM model's predictions. It is a measure of how well the model predicts future values based on previous values. A lower RMSE score indicates that the predictions are more accurate, and thus the model is more accurate.

The RMSE score is sensitive to outliers and does not indicate the direction of the error, for example, if the model predicts a value that is too high or too low. As a result, other metrics such as Mean Absolute Error (MAE) or Mean Absolute Percentage Error (MAPE) are frequently used in conjunction with RMSE.

For our model, the Root Mean Squared Error (RMSE) was 3.1629635065246684

Date	Close	Predictions
2022-12-16	140.429993	143.776321
2022-12-19	139.610001	143.462799
2022-12-20	138.830002	143.024475
2022-12-21	137.460007	142.484283
2022-12-22	137.000000	141.828735
2022-12-23	137.139999	141.112305
2022-12-27	136.160004	140.411591
2022-12-28	134.130005	139.707962
2022-12-29	137.410004	138.924820
2022-12-30	136.580002	138.337814
2023-01-03	138.279999	137.869278
2023-01-04	137.190002	137.611755

Figure 18:Expected closing price and actual closing price

Analysis of system

Legal, Social, Ethical and Professional issues

A variety of legal, social, ethical, and professional issues can arise in the context of stock forecasting. Here are a couple of examples:

Legal issues:

Insider trading: Insider trading is the illegal purchase or sale of securities by someone with non-public information about the security. This information is not available to the general public and would almost certainly affect the stock price if it were. Company officers,

directors, and employees are examples of insiders. Insider trading is punishable by fines and imprisonment.

False or misleading statements: False or misleading stock statements refer to the practise of making false or inaccurate statements about a company or its securities in order to influence the price of the stock. Exaggerating a company's financial performance, concealing negative information, or providing false information about the company's products or services are examples of this. This type of behaviour is illegal, and those found guilty face fines and imprisonment. The Securities and Exchange Commission (SEC) is in charge of enforcing securities laws as well as investigating potential cases of false or misleading stock statements..

Social issues:

Conflicts of interest: In the stock market, conflicts of interest refer to situations in which individuals or companies have competing interests that may influence their decision-making in ways that are not in the best interests of their clients or investors. Insider trading, in which a person with privileged information about a company uses that information to make trades for their own benefit, and analysts promoting stocks in which their firm has a financial interest, are two examples of stock market conflicts of interest. Investors must be aware of potential conflicts of interest and thoroughly research any investment before making a decision.

Undue influence: In the context of stock prediction using machine learning, undue influence refers to the use of biassed or manipulated data, algorithms, or parameters to achieve unfair or undesirable results. Machine learning models are trained on historical data, which can result in a biassed model that makes incorrect predictions if the data is not diverse or manipulated. For example, if a model is trained on historical data that only includes successful company stock prices, it may be unable to accurately predict the stock prices of unsuccessful companies.

Furthermore, if the model's parameters are set to achieve a specific outcome, it can lead to undue influence. For example, if a model is trained to accurately predict stock prices, it may overlook important factors that may affect stock prices in the future, resulting in inaccurate predictions.

Furthermore, if data and parameters are manipulated to benefit specific individuals, it can lead to insider trading, fraud, and harm investors' interests.

Ethical concerns:

Misuse of information: In the stock market, misusing information can raise serious ethical concerns. Insider trading is a prime example of this, in which a person with privileged information about a company uses that information to make trades for their own benefit. Insider trading is prohibited because it gives the person with insider information an unfair advantage over other investors and can lead to market manipulation and price distortion. Furthermore, when analysts or investment firms publish research or make stock recommendations, they must disclose any potential conflicts of interest, such as if the firm has a financial interest in the stock being recommended. Failure to disclose conflicts of

interest can be seen as unethical and can lead to investor distrust. Misuse of information can also lead to market manipulation, which can have a negative impact on the economy as a whole.

Misleading the public: In stock prediction, misleading the public can raise a number of ethical concerns. One source of concern is that it may result in financial losses for individual investors who base their decisions on false or inaccurate information. It can also contribute to market manipulation and cause stock prices to be distorted. Misleading the public about stock predictions can erode trust in financial markets and make it more difficult for people to make informed investment decisions.

Another concern is that this type of behaviour may perpetuate the perception that the stock market is speculative and untrustworthy, discouraging long-term investments and ultimately harming the economy as a whole. Furthermore, it can lead to a lack of transparency and accountability, further undermining trust in the financial system.

Furthermore, providing misleading information by a professional, such as a financial advisor or analyst, may be a violation of their professional code of conduct, which could result in disciplinary action or even legal repercussions.

Overall, misleading the public about stock predictions is unethical and can have serious consequences for both individuals and the financial system as a whole.

Professional concerns:

Reputation: Professional reputation concerns in stock prediction include the potential damage to one's reputation if their predictions are consistently inaccurate or if they are found to be engaging in unethical or illegal behaviour, such as insider trading or misleading the public.

Clients and investors are more likely to trust and invest with individuals and firms that have a good reputation for accuracy and integrity, which is important for financial analysts and advisors. A poor reputation can result in client and revenue loss, as well as damage to the firm's overall reputation.

Stock market commentators, analysts, and financial professionals are expected to have a high level of expertise and knowledge in their field, so making predictions that do not come true can call their credibility and expertise into question.

Furthermore, reputation is easily harmed by negative news, social media, and word of mouth, and it is difficult to repair once harmed. To protect their reputation, stock market professionals must maintain a high level of integrity and be transparent and honest in their predictions and analysis.

Professional standards: Financial analysts and advisors should follow several professional standards when making stock predictions. These requirements include:

Independence: Financial analysts and advisors should be independent and not allow conflicts of interest, such as financial interests in the stocks they recommend, to influence their predictions.

Due Diligence: Before making predictions and recommendations on stocks, financial analysts and advisors should conduct extensive research and due diligence.

Transparency: Financial analysts and advisors should be open about their research methods and information sources, as well as any potential conflicts of interest.

Objectivity: Financial analysts and advisors must maintain objectivity and avoid allowing personal biases to influence their predictions.

Professionalism: Financial analysts and advisors should conduct themselves professionally and should not engage in behaviour that could harm their or their firm's reputation.

Compliance: Financial analysts and advisors must abide by all applicable laws, regulations, and industry standards.

Following these guidelines can help to build trust and credibility with clients and investors, as well as protect the reputation of the financial analyst or advisor and the firm for which they work.

Conclusion

Finally, LSTM (Long Short-Term Memory) models can be an effective tool for stock market forecasting. These models can handle sequential data and take into account historical information, making them ideal for forecasting future stock prices. LSTMs can also be used to analyse and identify patterns in financial data, such as identifying stock price trends or detecting anomalies in trading activity. While LSTM models have been shown to be effective in stock market prediction, it is important to note that these predictions are never completely accurate, and other factors such as the investor's risk tolerance, market conditions, and trading strategy should be taken into account.

There are several potential uses of stock market prediction by LSTM models, including:

Trading: LSTM models can be used to predict future stock prices, which can help traders make better buy and sell decisions.

Risk Management: LSTM models can be used to identify patterns and trends in stock prices, which can help investors and traders manage risk.

Portfolio Optimization: LSTM models can be used to predict the performance of different stocks, which can help investors and traders optimize their portfolios.

Market Analysis: LSTM models can be used to analyse large amounts of stock market data and identify patterns that may not be immediately apparent.

Algorithmic Trading: LSTM models can be used to build sophisticated trading algorithms that can automatically make buy and sell decisions based on predictions of stock prices.

It is important to note that LSTM models, like all predictive models, are not perfect and they are subject to errors, so they should be used as one of the many tools in making a decision. Also, it's important to keep in mind that there are many other factors that can influence stock prices such as news, economic indicators, and market sentiment.

Bibliography

1. Ariyo, A.A., Adewumi, A.O. and Ayo, C.K., 2014, March. Stock price prediction using the ARIMA model. In *2014 UKSim-AMSS 16th international conference on computer modelling and simulation* (pp. 106-112). IEEE.

2. Nelson, D.M., Pereira, A.C. and De Oliveira, R.A., 2017, May. Stock market's price movement prediction with LSTM neural networks. In *2017 International joint conference on neural networks (IJCNN)* (pp. 1419-1426). Ieee.

3. Sak, H., Senior, A. and Google, B. (n.d.). *Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling*. [online] Available at: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43905.pdf.

4. Ding, G. and Qin, L., 2020. Study on the prediction of stock price based on the associated network model of LSTM. *International Journal of Machine Learning and Cybernetics*, *11*(6), pp.1307-1317.

5. Sunny, M.A.I., Maswood, M.M.S. and Alharbi, A.G., 2020, October. Deep learning-based stock price prediction using LSTM and bi-directional LSTM model. In *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)* (pp. 87-92). IEEE.

6. Selvin, S., Vinayakumar, R., Gopalakrishnan, E.A., Menon, V.K. and Soman, K.P., 2017, September. Stock price prediction using LSTM, RNN and CNN-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)* (pp. 1643-1647). IEEE.

7. Liu, Y., Wang, Z. and Zheng, B., 2019, December. Application of regularized GRU-LSTM model in stock price prediction. In *2019 IEEE 5th International Conference on Computer and Communications (ICCC)* (pp. 1886-1890). IEEE.

8. Bontempi, G., Ben Taieb, S. and Borgne, Y.A.L., 2012, July. Machine learning strategies for time series forecasting. In *European business intelligence summer school* (pp. 62-77). Springer, Berlin, Heidelberg.

9. Zeroual, A., Harrou, F., Dairi, A. and Sun, Y., 2020. Deep learning methods for forecasting COVID-19 time-Series data: A Comparative study. *Chaos, Solitons & Fractals*, *140*, p.110121.

10. Siami-Namini, S., Tavakoli, N. and Namin, A.S., 2018, December. A comparison of ARIMA and LSTM in forecasting time series. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)* (pp. 1394-1401). IEEE.

11. Sezer, O.B., Gudelek, M.U. and Ozbayoglu, A.M., 2020. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, *90*, p.106181.

12. Parmar, I., Agarwal, N., Saxena, S., Arora, R., Gupta, S., Dhiman, H. and Chouhan, L., 2018, December. Stock market prediction using machine learning. In *2018 first international conference on secure cyber computing and communication (ICSCCC)* (pp. 574-576). IEEE.

13. Jiang, W., 2021. Applications of deep learning in stock market prediction: recent progress. *Expert Systems with Applications*, *184*, p.115537.

References

Y. Wang, S. Z. a. C. L., 2019. *Research on Multistep Time Series Prediction Based on LSTM.* [Online]
Available at: https://ieeexplore.ieee.org/document/9095044
[Accessed 2023].
Tondak, a., 2022. *k21academy.* [Online]
Available at: https://k21academy.com/datascience-blog/machine-learning/recurrent-neural-networks/
[Accessed 2023].
Ackerson, J. D. R. a. S. N., 2021. *mdpi.* [Online]
Available at: https://www.mdpi.com/2078-2489/12/7/272
Yanhui, C., 2021. *towards data science.* [Online]
Available at: https://towardsdatascience.com/a-battle-against-amnesia-a-brief-history-and-introduction-of-recurrent-neural-networks-50496aae6740
Aroussi, R., 2019. *aroussi.* [Online]
Available at: https://aroussi.com/post/python-yahoo-finance
[Accessed 2023].
Recurrent Neural Networks (RNN) and LSTM: Overview and Uses, n.d. *turing.* [Online]
Available at: https://www.turing.com/kb/recurrent-neural-networks-and-lstm
[Accessed 2023].

Appendices

Source code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
plt.style.use("fivethirtyeight")
%matplotlib inline
from pandas_datareader.data import DataReader
import yfinance as yf
from datetime import datetime



tech_list = ['TSLA', 'F', 'GM','TM']
tech_list = ['TSLA', 'F', 'GM','TM']

end = datetime.now()
start = datetime(end.year - 1, end.month, end.day)

for stock in tech_list:
    globals()[stock] = yf.download(stock, start, end)


company_list = [TSLA, F, GM,TM]
company_name = ["Tesla", "Ford Motor", "General Motors","Toyota Motor"]

for company, com_name in zip(company_list, company_name):
    company["company_name"] = com_name

df = pd.concat(company_list, axis=0)
df.tail(10)
```

```
TSLA.describe()
```

```
TSLA.info()
```

```
plt.figure(figsize=(15, 10))
plt.subplots_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(company_list, 1):
    plt.subplot(2, 2, i)
    company['Adj Close'].plot()
    plt.ylabel('Adj Close')
    plt.xlabel(None)
    plt.title(f"Closing Price of {company_name[i - 1]}")

plt.tight_layout()
```

```
plt.figure(figsize=(15, 10))
plt.subplots_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(company_list, 1):
    plt.subplot(2, 2, i)
    company['Volume'].plot()
    plt.ylabel('Volume')
    plt.xlabel(None)
    plt.title(f"Sales Volume for {company_name[i - 1]}")

plt.tight_layout()
```

```
ma_day = [10, 20, 50]

for ma in ma_day:
    for company in company_list:
        column_name = f"MA for {ma} days"
        company[column_name] = company['Adj Close'].rolling(ma).mean()


fig, axes = plt.subplots(nrows=2, ncols=2)
fig.set_figheight(10)
fig.set_figwidth(15)

TSLA[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,0])
axes[0,0].set_title('Tesla')

F[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,1])
axes[0,1].set_title('Ford Motor')

GM[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,0])
axes[1,0].set_title('General Motors')

TM[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,1])
axes[1,1].set_title('Toyota Motor')

fig.tight_layout()
```

```
for company in company_list:
    company['Daily Return'] = company['Adj Close'].pct_change()


fig, axes = plt.subplots(nrows=2, ncols=2)
fig.set_figheight(10)
fig.set_figwidth(15)

TSLA['Daily Return'].plot(ax=axes[0,0], legend=True, linestyle='--', marker='o')
axes[0,0].set_title('Tesla')

F['Daily Return'].plot(ax=axes[0,1], legend=True, linestyle='--', marker='o')
axes[0,1].set_title('Ford Motor')

GM['Daily Return'].plot(ax=axes[1,0], legend=True, linestyle='--', marker='o')
axes[1,0].set_title('General Motors')

TM['Daily Return'].plot(ax=axes[1,1], legend=True, linestyle='--', marker='o')
axes[1,1].set_title('Toyota Motor')

fig.tight_layout()
```

```
plt.figure(figsize=(12, 9))

for i, company in enumerate(company_list, 1):
    plt.subplot(2, 2, i)
    company['Daily Return'].hist(bins=50)
    plt.xlabel('Daily Return')
    plt.ylabel('Counts')
    plt.title(f'{company_name[i - 1]}')

plt.tight_layout()
```

```
import pandas
from pandas_datareader import data as pdr
import yfinance as yfin
yfin.pdr_override()

closing_df = pdr.get_data_yahoo(tech_list, start, end)['Adj Close']
tech_rets = closing_df.pct_change()
tech_rets.head()
```

```
sns.pairplot(tech_rets, kind='reg')
```

```
return_fig = sns.PairGrid(tech_rets.dropna())
return_fig.map_upper(plt.scatter, color='purple')
return_fig.map_lower(sns.kdeplot, cmap='cool_d')
return_fig.map_diag(plt.hist, bins=30)
```

```
returns_fig = sns.PairGrid(closing_df)
returns_fig.map_upper(plt.scatter,color='purple')
returns_fig.map_lower(sns.kdeplot,cmap='cool_d')
returns_fig.map_diag(plt.hist,bins=30)
```

```
plt.figure(figsize=(12, 10))

plt.subplot(2, 2, 1)
sns.heatmap(tech_rets.corr(), annot=True, cmap='summer')
plt.title('Correlation of stock return')

plt.subplot(2, 2, 2)
sns.heatmap(closing_df.corr(), annot=True, cmap='summer')
plt.title('Correlation of stock closing price')
```

```
rets = tech_rets.dropna()

area = np.pi * 20

plt.figure(figsize=(10, 8))
plt.scatter(rets.mean(), rets.std(), s=area)
plt.xlabel('Expected return')
plt.ylabel('Risk')

for label, x, y in zip(rets.columns, rets.mean(), rets.std()):
    plt.annotate(label, xy=(x, y), xytext=(50, 50), textcoords='offset points', ha='right', va='bottom',
                 arrowprops=dict(arrowstyle='-', color='blue', connectionstyle='arc3,rad=-0.3'))
```

```
df = pdr.get_data_yahoo("TM", start='2022-01-01', end=datetime.now())
df
```

```
plt.figure(figsize=(16,6))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
```

```
data = df.filter(['Close'])
dataset = data.values
training_data_len = int(np.ceil( len(dataset) * .95 ))

training_data_len
```

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)

scaled_data
```

```
train_data = scaled_data[0:int(training_data_len), :]

x_train = []
y_train = []

for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])
    if i<= 61:
        print(x_train)
        print(y_train)
        print()


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))
```

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

model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))

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

model.fit(x_train, y_train, batch_size=1, epochs=1)
```

```
test_data = scaled_data[training_data_len - 60: , :]

x_test = []
y_test = dataset[training_data_len:, :]
for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])


x_test = np.array(x_test)

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


predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)


rmse = np.sqrt(np.mean(((predictions - y_test) ** 2)))
rmse
```

```
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictions
plt.figure(figsize=(16,6))
plt.title('Model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
plt.legend(['Train', 'Val', 'Predictions'], loc='upper right')
plt.show()
```