10. The Big Data Applications Project : Stock Price Prediction

Abstract

A stock, or equity, signifies ownership in a corporation, providing the shareholder with a proportional claim to the company's assets and profits based on the amount of stock they possess. Shares are the individual units of stock. Typically, the term "stock" denotes ownership in any company. Stock prices vary daily due to market forces, primarily supply and demand. When demand for a stock rise, with more buyers than sellers, the price increases. Conversely, if there are more sellers than buyers, creating an excess supply, the price falls. While the basic concept of supply and demand is straightforward, the reasons behind stock price changes are complex and uncertain. Some argue that predicting stock price movements is impossible, while others believe that analyzing charts and past price trends can provide insights on when to buy or sell. The only certainty is that stock prices are highly volatile and can change very quickly.

Objective

The stock market is vital to the economy of a country. Stock market trading requires investors to have access to have accurate information to make informed decisions. Everyday lots of stocks being traded on exchanges, with various factors influence of these decisions. Additionally, the behavior of stock prices is unpredictable and complex. Predicting stock prices has become both important and challenging. The research aims to identify the most effective prediction model that produces the most accurate forecasts with the lowest error percentage. The main objective of this mini project is to explore and determine the best methods for predicting the closing prices of stocks by through deep learning and machine learning methods.

Dataset

The dataset is obtained from Kaggle and includes historical prices and fundamental data for 500 companies listed on the AMEX, NASDAQ, and NYSE stock markets. It covers a period of 10 years, from January 2010 to April 2020, giving a detailed view of stock performance over that time.
The dataset contains 97,648 rows and 8 columns.

Name: This column lists the name of the stock or company being tracked.

Date: This column shows the date for each stock market entry, indicating when the data was recorded.

Open: This column notes the price of the stock at the start of the trading day.

Closing_Price: This column provides the price of the stock at the close of the trading day.

Daily_High: This column displays the highest price reached by the stock during the day.

Daily_Low: This column records the lowest price of the stock during the trading day.

Volume: This column indicates the total number of shares that were traded throughout the day.

```
[8]  df.head()
```

	Unnamed: 0	Name	Date	Open	Closing_Price	Daily_High	Daily_Low	Volume
0	0	Accor	2020-04-03	22.99	23.40	23.40	22.99	67
1	1	Accor	2020-04-02	23.91	22.99	23.91	22.99	250
2	2	Accor	2020-04-01	24.10	23.83	24.10	23.83	37
3	3	Accor	2020-03-31	25.04	25.00	25.24	24.99	336
4	4	Accor	2020-03-30	26.50	25.02	26.50	24.99	415

Data Pre-processing

1. Dropping Unnecessary Columns

The column Unnamed: 0 was identified as unnecessary and is dropped.

2. Handling Missing Values

The ‘Open’ column's missing values were filled with the column's mean.
It contains continuous numerical data, making mean imputation a suitable choice. The mean is a balanced approach that maintains stocks dataset integrity and simplicity.

The 'Closing_Price' column's missing values is filled with the column's median.
Stock prices can have significant fluctuations, and using the median ensures that extreme high or low prices do not skew the imputation, providing a more robust and representative central value.

The 'Daily_High' column's missing values is filled using the forward fill method. Forward filling helps in preserving peaks in the high prices of stocks, where high volatility and rapid price changes are common.

The 'Daily_Low' column's missing values were filled using the backward fill method.
Daily low prices are often influenced by intraday market fluctuations. Using backward fill leverages the latest observed low price.

The 'Volume' column's missing values and non-numeric entries were converted to numeric, and missing values were filled with 0. While filling missing values with 0 might not accurately reflect actual trading volume, it preserves the information that there was no reported trading activity for those instances. This can still be valuable information for analysis and modeling.

Splitting the Dataset

The dataset was split into a training set (60%), validation set (10%) and a testing set (30%) to evaluate the performance of the models.
The sequence length, also known as the number of past time steps (n_past), was set to 70. This means that the model will use the last 70-time steps to predict the next time step.

I utilized the 'specific_data' function, specifying 'Peugeot' as the company name, along with designated start and end dates, to procure a dataset for analysis. Employing this function, my objective is to forecast the price movement of Peugeot stock.

Visualisation

The plot below showcases the daily closing prices of Peugeot's stock alongside a 30-day moving average. The moving average aids in smoothing out short-term fluctuations, providing a clearer trend analysis.
Closing Price: The solid line represents the actual closing prices of Peugeot's stock on each trading day.
30-Day Moving Average: The dashed line depicts the rolling average of the closing prices over a 30-day period, highlighting the underlying trend.

This visualization enables a comprehensive understanding of Peugeot's stock performance over time, aiding in trend identification and decision-making processes for investors and analyst.

Regression Algorithms

Three regression algorithms were selected for this task:

a. Simple RNN (Recurrent Neural Networks)

b. Simple Stacked RNN (Recurrent Neural Networks)

c. LSTM RNN (Long short-term memory Recurrent Neural Networks)

Model Evaluation

Each algorithm was trained on the training data and evaluated on the testing data using the following evaluation metrics:

• Mean Absolute Error
• Mean Squared Error
• Root Mean Square Error

Results

a. Simple RNN:

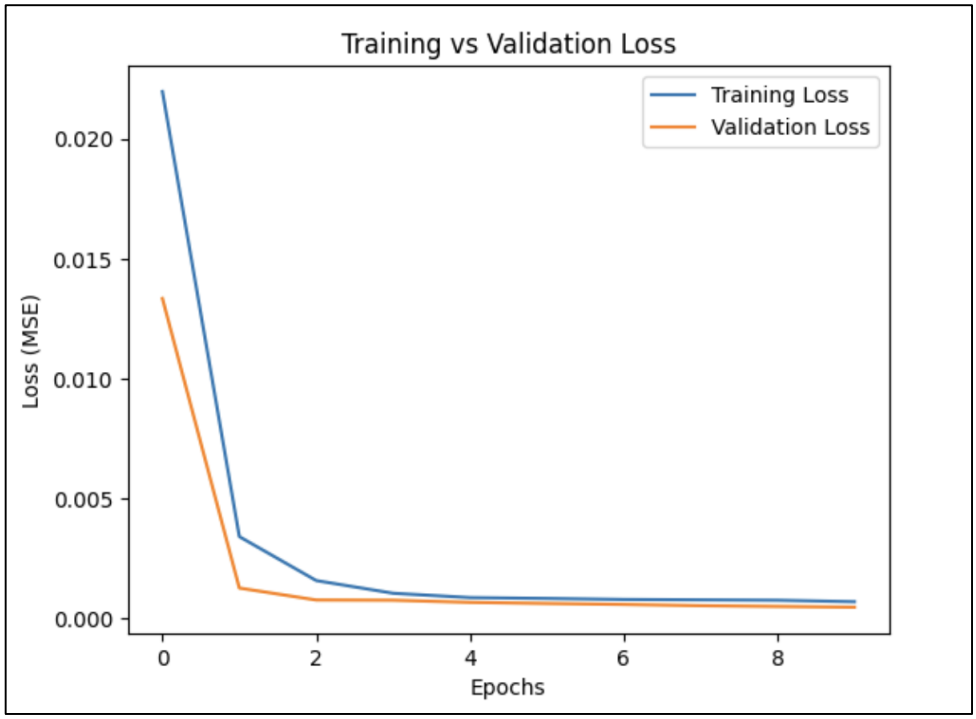
This model is simple Recurrent Neural Network (RNN) designed for a sequence prediction task. It consists of a single Simple RNN layer followed by a Dense layer to produce the final output.
The total number of 2651 trainable parameters in the model.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn (SimpleRNN)       (None, 50)                2600

dense (Dense)                (None, 1)                 51

=================================================================
```

The RMSE is close to the MAE , indicating that the errors are relatively consistent without extremely large deviations. If there were large outliers, the RMSE would be significantly higher than the MAE.

Train MAE	**14.73**
Train MSE	**243.54**
Train RMSE	**15.60**

When you see the below graph of training vs validation loss it is trained with 10 epochs, the model initially trains well. But after epoch 2 the learning rate slows down. Eventually, the loss becomes almost a straight line close to zero (MSE = 0.000), which indicates that the model fitting the training data very closely. This suggests that the model is likely overfitting.



b. Simple Stacked RNN:

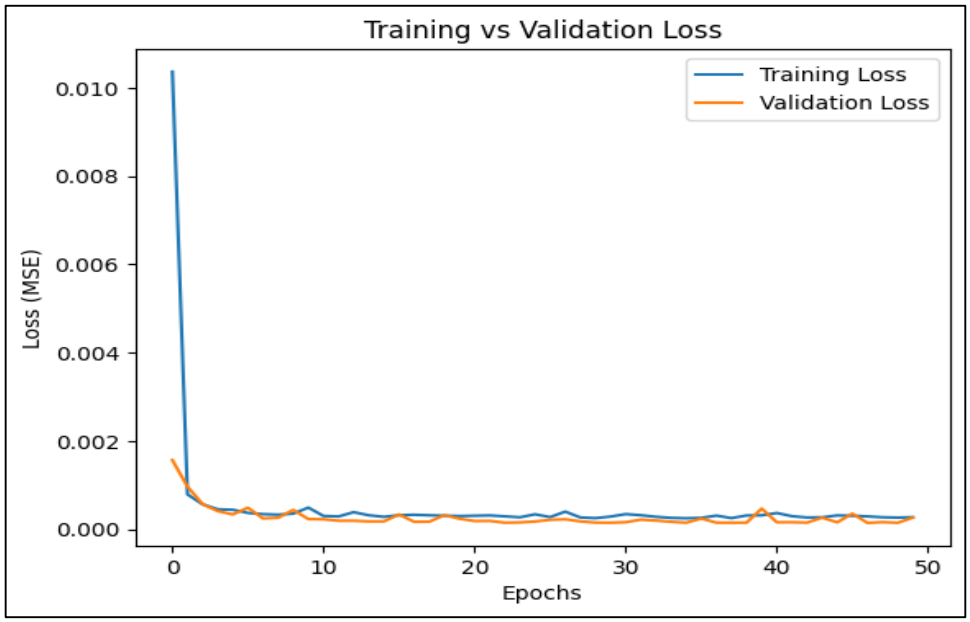
By stacking two Simple RNN layers, the model captures more complex temporal patterns. The first layer processes the input sequence and outputs 50-dimensional vectors, which the second layer condenses into a single 50-dimensional vector. The final Dense layer maps this output to a single value, making it suitable for predicting continuous values in regression tasks.

```
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 70, 50)            2600

 simple_rnn_1 (SimpleRNN)    (None, 50)                5050

 dense (Dense)               (None, 1)                 51
```

The Simple Stacked RNN model shows improvement over the Simple RNN, with lower MAE/MSE on training datasets.

Train MAE	**14.77**
Train MSE	**244.71**
Train RMSE	**15.64**

Simple Stacked RNN potentially offers better performance by capturing hierarchical features through multiple layers. However when you see the graph of training loss vs validation loss, it is evident that the validation loss is significantly diverging from the training loss. This indicates that the model is overfitting the training data.



c. Long short-term memory RNN:

The Long short-term memory RNN architecture comprises three LSTM layers, each housing 50 units. The initial LSTM layer yields sequences with dimensions (None, 70, 50), featuring 10,400 parameters, while the subsequent layer produces similar-shaped sequences with 20,200 parameters. The third LSTM layer outputs a vector of size (None, 50), by 20,200 parameters. Every LSTM tier incorporates a dropout layer set at a 0.2 rate to forestall overfitting. Following this, a dense layer,

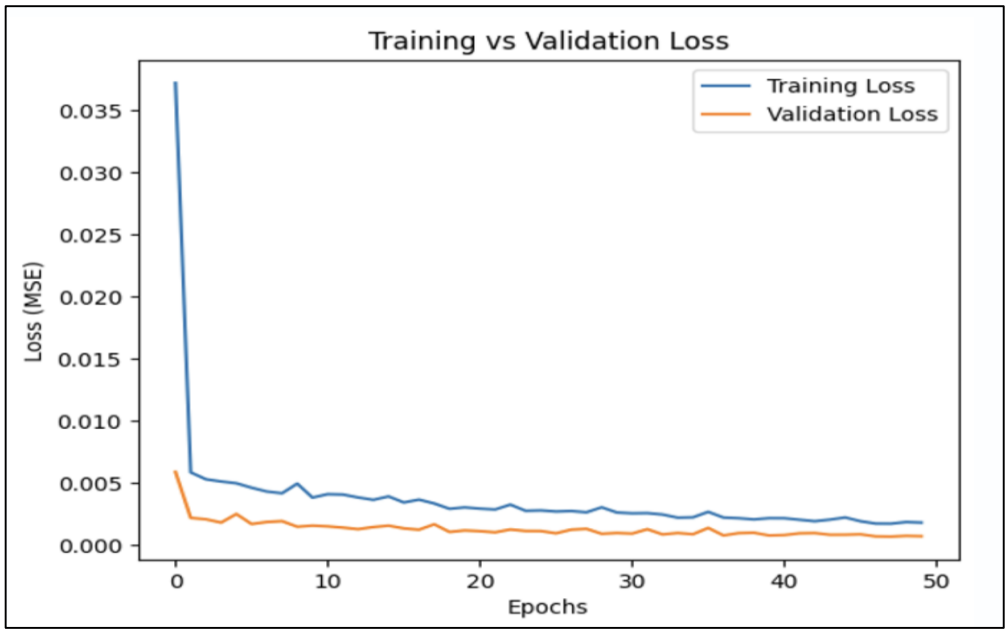
generates the model's output, characterized by 51 parameters. These dropout layers, devoid of parameters, serve the purpose of regularization. This model adeptly captures temporal intricacies in data, offering utility in tasks such as time series forecasting and sequence prediction.

Layer (type)	Output Shape	Param #
lstm_28 (LSTM)	(None, 70, 50)	10400
dropout_26 (Dropout)	(None, 70, 50)	0
lstm_29 (LSTM)	(None, 70, 50)	20200
dropout_27 (Dropout)	(None, 70, 50)	0
lstm_30 (LSTM)	(None, 50)	20200
dropout_28 (Dropout)	(None, 50)	0
dense_9 (Dense)	(None, 1)	51

The LSTM model performs the best among the three models, with the lowest MAE/MSE/RMSE on training datasets. The inclusion of LSTM layers allows the model to capture long-term dependencies better, leading to improved performance.

Train MAE	**14.36**
Train MSE	**229.31**
Train RMSE	**15.14**

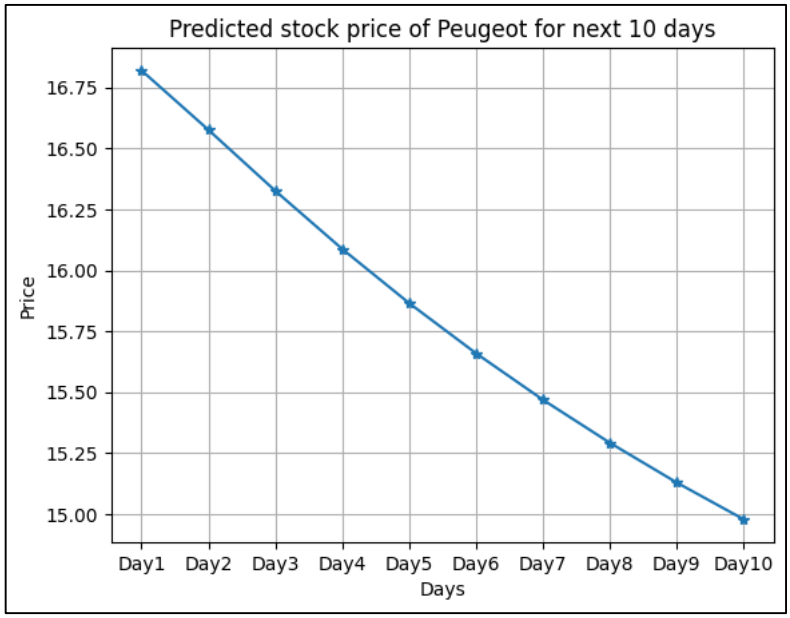
In this LSTM model trained over 50 epochs with 50 neurons, the training dynamics reveal a compelling pattern. After just 2 epochs, the model exhibits proficient learning, as evidenced by the plotted graph. Notably, as training progresses, the learning rate slows down, indicative of the model's assimilation of intricate patterns within the data. The minimal gap observed between the training and validation losses signifies , suggesting the model is learning well.



Conclusion

The LSTM model outperforms both Simple RNN and Simple Stacked RNN in terms of predictive accuracy and generalization.
The LSTM model is preferred due to its ability to handle long-term dependencies effectively, resulting in better performance on time series data.
Although Simple RNN shows improvement over Simple Stacked RNN, it still lags behind LSTM in terms of performance, indicating the importance of more advanced architectures like LSTM for sequential data modeling.

The LSTM model trained on historical stock price data, we have successfully predicted the next 10 days' stock prices for Peugeot by utilizing the last sequential length(n_past) days of data. The below visualisation representation of the predictions was provided through a line plot, showcasing the projected stock prices over the next 10 days for Peugeot.



Reference :

kaggle.com. (n.d.). Stock Market Analysis + Prediction using LSTM. [online] Available at: https://www.kaggle.com/code/faressayah/stock-market-analysis-prediction-using-lstm.

www.kaggle.com. (n.d.). New York Stock Exchange. [online] Available at: https://www.kaggle.com/datasets/dgawlik/nyse.

venky14 (2019). venky14/Stock-Market-Analysis-and-Prediction. [online] GitHub. Available at: https://github.com/venky14/Stock-Market-Analysis-and-Prediction [Accessed 28 Dec. 2019].