

COMS4030A - Adaptive Computation and Machine Learning Project

School of Computer Science & Applied Mathematics
University of the Witwatersrand



Lisa Godwin - 2437980
Nihal Ranchod - 2427378

May 21, 2024

Table of Contents

1	Dataset Source	3
2	Dataset Contents and Objectives	3
3	Preprocessing and Data Splitting	4
3.1	Preprocessing Steps	4
3.1.1	Date Handling	4
3.1.2	Data Cleaning	4
3.1.3	Normalising the Data	4
3.1.4	Creating Sequences for LSTM input	4
3.2	Data Splitting Steps	5
3.2.1	Splitting the Dataset into Train and Test Sets	5
3.2.2	Train/Validation Split	6
3.2.3	Overall Train/Validation/Test Split	6
3.2.4	Reshaping for LSTM Input	6
3.2.5	Preparing the Test Set	6
4	Choice of Model and Machine Learning Algorithm	7
4.1	Choice of Model: Long Short-Term Memory (LSTM)	7
4.2	Machine Learning Algorithm: LSTM Network	7
4.3	Implementation of the Model	7
4.3.1	Hyper-parameters	7
4.3.2	Model Architecture	7

4.3.3	Model Compilation	7
4.4	Training the Model	8
4.4.1	Evaluation and Predictions	8
5	Experimentation and Model Fine-Tuning	9
5.1	Hyper-parameter Tuning	9
5.2	Choice of Hyper-parameters	9
5.3	Results	10
6	Graphs Generated During Training	11
6.1	Graphs Generated	11
6.2	Analysis of Graphs	12
7	Results of the Trained Model	13
7.1	Tesla Close Stock Price Prediction Plot	13
7.2	Prediction Errors Histogram	14
7.3	Error Metrics Analysis	14

All python scripts created for this project can be found at our [GitHub Repository](#)

1 Dataset Source

We selected a dataset from [Kaggle.com](https://www.kaggle.com), specifically the ‘Historical Stock Prices of Tesla’, which contains daily stock price data for Tesla. This dataset is publicly available and does not require any ethics clearance for use.

2 Dataset Contents and Objectives

The dataset includes historical daily Tesla stock price information from January 2nd, 2015 to January 16th, 2024, with the following columns:

1. **Date:** The date of the stock price entry.
2. **Open:** The opening price of the stock on the given date.
3. **High:** The highest price of the stock on the given date.
4. **Low:** The lowest price of the stock on the given date.
5. **Close:** The closing price of the stock on the given date.
6. **Volume:** The number of shares traded in the given data.

The main objective with respect to the dataset is aimed at developing and evaluating a Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) based model using machine learning algorithms for predicting daily closing prices of Tesla stock. The model leverages historical closing price data to learn patterns and predict ‘Close’ values.

The following is a break down of the specific objectives achieved through the code:

1. Stock Price Prediction Modelling:
 - Develop an LSTM model to predict Tesla’s stock closing prices.
2. Data Normalisation and Preprocessing:
 - Monitor training and validation loss to ensure model generalisation.
3. Model Training and Validation:
 - Normalise stock price data using MinMaxScaler.
 - Create sequences from time series data for LSTM input.
4. Model Evaluation and Performance Metrics:
 - Asses model accuracy using R^2 score, Mean Squared Error (MSE), and Mean Absolute Error (MAE).
5. Visualisation of Results:
 - Visualise training and validation loss over epochs.
 - Visualise actual vs. predicted stock prices.

3 Preprocessing and Data Splitting

In this section, we outline the preprocessing steps taken and the split of the dataset for training, validation, and testing purposes.

3.1 Preprocessing Steps

3.1.1 Date Handling

To facilitate time series analysis, the 'Date' column was converted to a date-time format and set as the index of the DataFrame. This ensured that dates are treated correctly and can be used for time-based analysis. We then dropped the 'Date' column itself since the index now holds the date information.

3.1.2 Data Cleaning

The data cleaning process ensured that the dataset was prepared for analysis and modelling. This involved identifying and addressing extra columns, missing values, and duplicate rows, thereby preventing any potential issues with the predictive model.

We identified and removed any unnecessary columns which were not required for the analysis, such as unnamed index columns. This column was dropped to streamline the dataset ensuring the model focuses solely on relevant features, such as the closing price in this case.

To handle missing values we utilised the `isnull()` method combined with the `sum()` to check for missing values in each column of the dataset by providing a count for each. If any missing values were detected, forward filling was used to maintain the temporal sequence integrity.

Furthermore, we utilised the `duplicated()` method to check for duplicate rows in the dataset. If duplicate rows were found, we removed them using the `drop_duplicates()` method.

By thoroughly cleaning the data, we ensured that the dataset is free from inconsistencies that could negatively impact the analysis and predictive modelling.

3.1.3 Normalising the Data

The 'Close' price column was normalised using `MinMaxScaler` to scale the data between 0 and 1. This normalisation step is critical for optimising the performance of the LSTM model, as neural networks generally perform better with normalised inputs.

3.1.4 Creating Sequences for LSTM input

Sequences of a specified length (80) were generated to serve as input for the LSTM model. Each sequence included the 'Close' prices for the past 80 days, providing the model with the necessary temporal context.

3.2 Data Splitting Steps

3.2.1 Splitting the Dataset into Train and Test Sets

The dataset was split into training and testing sets based on a specific date, ensuring chronological order was maintained. This method preserves the temporal structure of the data, which is essential for time-series forecasting. Specifically, 75% of the data was allocated to the training set and 25% to the test set. The training-test split of the data can be seen in figure 1. This split ratio is advantageous for the following reasons:

1. **Adequate Training Data:** Using 75% of the data for training ensured that the model had enough historical information to learn from. This is particularly important for LSTM models, which rely on recognising patterns and trends over time. A larger training set helps the model capture the complex temporal dependencies in the stock price data.
2. **Sufficient Test Data:** Allocating 25% of the data to the test set provided a substantial amount of data for evaluating the model's performance. This helped in assessing the model's ability to generalise to new, unseen data and ensured that the evaluation metrics are reliable.
3. **Temporal Integrity:** By splitting the data chronologically, we maintained the natural order of events. This is crucial for time series forecasting, as it prevented data leakage and ensured that the model is tested on future data relative to the training data, mimicking real-world scenarios.

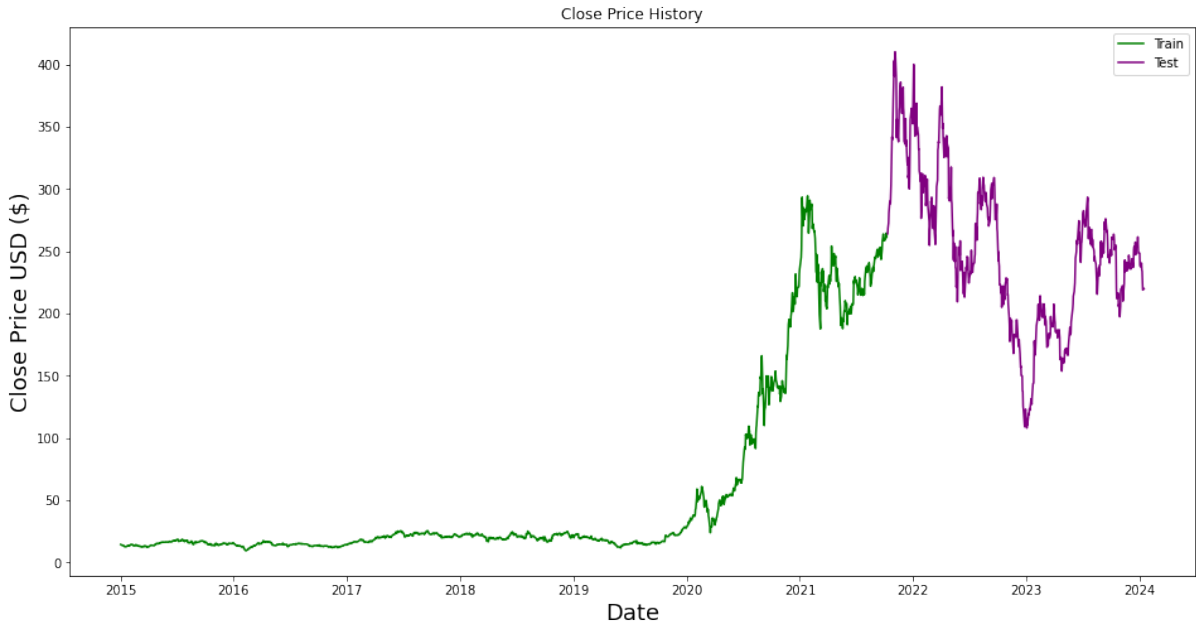


Figure 1: Closing Price History showing the training-test split of the dataset

3.2.2 Train/Validation Split

To monitor the model's performance on unseen data during training, the training data was further split into training and validation sets with a 70-30 split ratio. Specifically, 70% of the training data was allocated to the actual training process, while 30% was reserved for validation. This split is beneficial for several reasons:

1. **Model Fine-Tuning:** By setting aside 30% of the training data for validation, we evaluated the model's performance on data it hasn't seen during training. This helped in fine-tuning the model's hyper-parameters, ensuring that the model is optimised for better performance.
2. **Mitigating Over-fitting:** A separate validation set helped in monitoring the model's ability to generalise. If the model performs well on the training set but poorly on the validation set, it indicates over-fitting. The 70-30 split provided a balanced approach to detect and mitigate over-fitting.
3. **Balanced Data Utilisation:** Allocating 70% of the data to training ensured that the model had enough data to learn effectively, while the 30% validation set is substantial enough to provide meaningful feedback on the model's generalisation capabilities.

3.2.3 Overall Train/Validation/Test Split

- Training: 52.5%
- Validation: 22.5%
- Test: 25%

3.2.4 Reshaping for LSTM Input

The input data was reshaped to match the format expected by LSTM layers, which is (samples, time steps, features). This ensures compatibility with the LSTM network.

3.2.5 Preparing the Test Set

The test set was normalised and prepared in a manner similar to the training set. Sequences were created for the test data to maintain consistency in input structure and scaling

By following these preprocessing steps, the dataset was properly formatted and scaled, allowing the LSTM model to learn effectively from the training data and make accurate predictions on the test data. The train/validation split helped in tuning the model and preventing over-fitting, while the final test set provided an unbiased evaluation of the model's performance.

4 Choice of Model and Machine Learning Algorithm

4.1 Choice of Model: Long Short-Term Memory (LSTM)

For this closing stock price prediction task, we selected the Long Short-Term Memory (LSTM) model, a type of Recurrent Neural Network (RNN). The choice was driven by the following factors:

1. **Handling Sequential Data:** LSTMs are well-suited for time series data as they can learn and remember over long sequences, capturing temporal dependencies and trends effectively.
2. **Stock Price Prediction:** The ability of LSTMs to model complex patterns over the time makes them ideal for stock price prediction, where past prices influence future prices.

4.2 Machine Learning Algorithm: LSTM Network

The LSTM model was implemented using the Keras library with TensorFlow as the backend. Key components of the model include multiple LSTM layers, dropout layers for regularisation, and a dense output layer for prediction.

4.3 Implementation of the Model

4.3.1 Hyper-parameters

- Sequence Length: 80 (number of days to consider for each prediction)
- Batch Size: 32
- Epochs: 100
- Units: 50 (number of LSTM units per layer)
- Dropout Rate: 0.2 (20% dropout for regularisation)

4.3.2 Model Architecture

- LSTM Layers: Four LSTM layers were used, each followed by a dropout layer to prevent over-fitting.
- Dense Layer: A single dense layer was used as the output layer to predict the ‘Close’ price.

4.3.3 Model Compilation

- Optimiser: ‘Adam’ optimiser was used for its efficiency and adaptive learning rate capabilities.
- Loss Function: Mean Squared Error (MSE) was used as the loss function.

4.4 Training the Model

- The model was trained for 100 epochs with a batch size of 32. The training process was monitored using the validation set to track over-fitting and performance.

4.4.1 Evaluation and Predictions

- The trained model was evaluated using the test set. Predictions were made and then inverse transformed to the original scale.
- Performance metrics such as R^2 score, Mean Absolute Error (MAE), and Mean Squared Error (MSE) were calculated to assess the model's accuracy.
- Plots were generated to visualise the training/validation loss over epochs and the comparison between actual and predicted stock prices. These plots can be seen in [section 6](#) and [section 7](#).

5 Experimentation and Model Fine-Tuning

To find a suitable model and fine-tune its performance, a series of experiments were conducted, primarily focusing on optimising the hyper-parameters. These experiments were crucial for improving the model’s predictive accuracy and generalisation capabilities.

5.1 Hyper-parameter Tuning

A separate python script was created to perform hyper-parameter tuning using the `RandomizedSearchCV` technique, which can be found [here](#). The objective was to identify the optimal combination of hyper-parameters for the LSTM model.

1. Choice of Hyper-parameters:

- **Epochs:** The number of training epochs was selected as a hyper-parameter to determine the number of times the entire dataset is passed forward and backward through the neural network during training.
- **Batch Size:** This hyper-parameter specifies the number of samples processed before the model is updated. It affects the speed and stability of the training process.

2. Fixed Hyper-parameters:

- **Sequence Length:** The length of the input sequence (80 days) was kept fixed to maintain consistency in capturing historical data for each prediction.
- **Number of Units:** The number of LSTM units in each layer was kept fixed at 50. This choice was based on empirical evidence suggesting that a moderate number of units is sufficient for capturing temporal patterns in the data without over-fitting.
- **Dropout Rate:** The dropout rate was fixed at 0.2 for regularisation to prevent over-fitting.

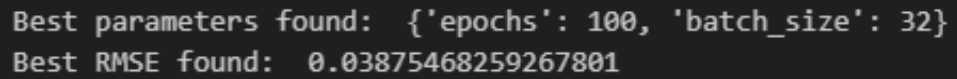
5.2 Choice of Hyper-parameters

1. **Epochs:** A range of values (75, 100, 125) was considered for the number of epochs to allow sufficient iterations for the model to converge without over-fitting.
2. **Batch Size:** Different batch sizes (16, 24, 32) were explored to balance computational efficiency and model stability during training.

The `RandomizedSearchCV` technique was employed with a cross-validation strategy using `TimeSeriesSplit`, which preserved the temporal order of the data during training and validation. This approach ensured that the model is evaluated on realistic future data, enhancing its ability to generalise.

5.3 Results

The best combination of hyper-parameters was identified based on the mean squared error (MSE) score obtained during cross-validation. By systematically exploring various hyper-parameter configurations, we aimed to improve the model's performance and achieve more accurate stock price predictions. The results from running the hyper-parameter tuning script, show that the best parameters for the LSTM model were to train the model for 100 epochs with a batch size of 32. The output of this can be seen in figure 2.



```
Best parameters found: {'epochs': 100, 'batch_size': 32}  
Best RMSE found: 0.03875468259267801
```

Figure 2: Output of the best parameters from hyper-parameter tuning

6 Graphs Generated During Training

During the training of the LSTM model, several graphs were generated to monitor the performance and convergence of the model. These graphs include the Mean Squared Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and the Training Loss against the Validation Loss over the epochs. The figures below illustrate these metrics and provide insight into the model's learning behaviour.

6.1 Graphs Generated

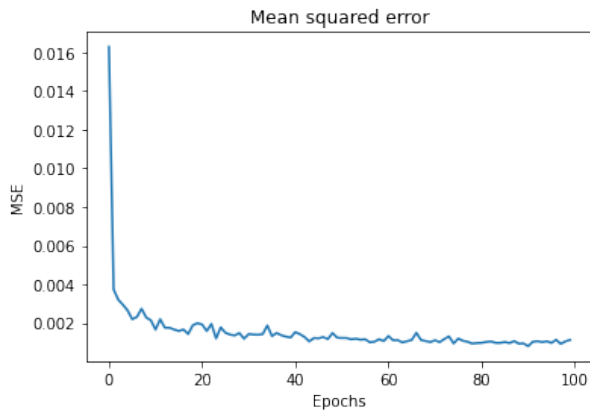


Figure 3: Mean Squared Error (MSE)

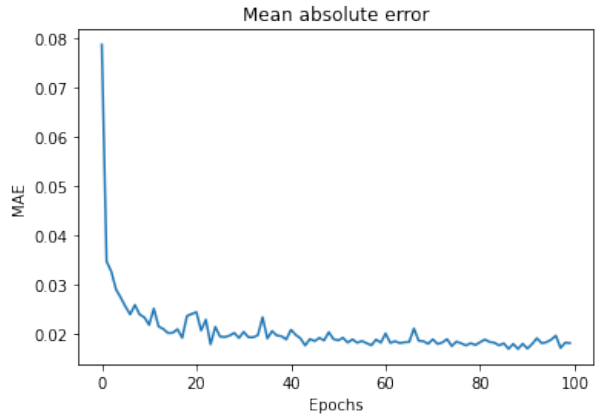


Figure 4: Mean Absolute Error (MAE)

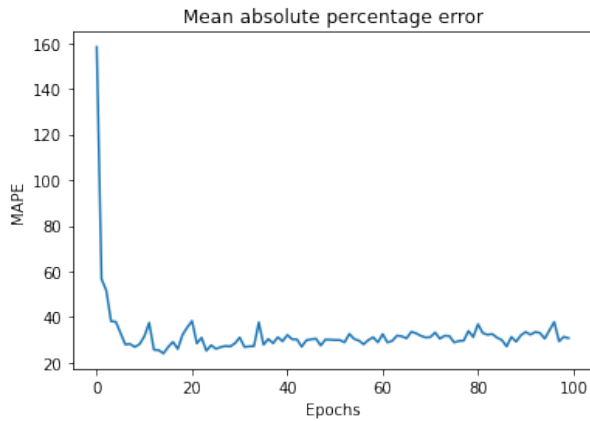


Figure 5: Mean Absolute Percentage Error (MAPE)

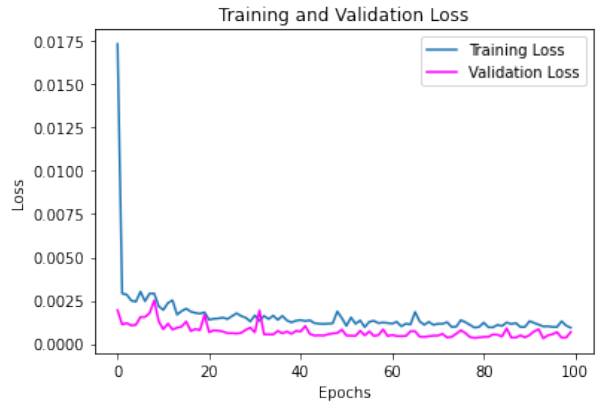


Figure 6: Training Loss vs. Validation Loss

6.2 Analysis of Graphs

The MSE plot in figure 3 shows a significant decrease in error within the first few epochs, indicating that the model quickly learns from the data. As the epochs progress, the MSE stabilises, suggesting that the model is converging. Lower MSE values imply that the model predictions are closely matching the actual values.

The MAE plot in figure 4 also exhibits a rapid decline in the initial epochs, similar to the MSE plot. This behaviour demonstrates the model's ability to reduce the absolute differences between predicted and actual values effectively. The error stabilises towards the latter epochs, confirming model convergence. The stabilisation towards the end suggests that the model has learned the underlying patterns in the data.

The MAPE plot in figure 5 shows a decrease in error in the early epochs, with more variability observed compared to MSE and MAE. This metric is sensitive to the scale of the data, which may explain the fluctuations. However, the overall trend indicates a reduction in percentage error, pointing to the model's improving accuracy.

The training and validation loss of the LSTM model seen in figure 6 shows the training loss decrease sharply within the first few epochs. This indicates that the model is quickly learning and adjusting its parameters to minimise the loss. After the initial drop, the training loss gradually decreases and stabilises. This suggests that the model continues to improve but at a slower rate as it converges. The validation loss generally follows the trend of the training loss but is slightly higher on average. This indicates that the model's performance on unseen data is slightly worse than on training data, which is typical. Both training and validation losses remain very low and stable from around epoch 20 onwards, showing that the model generalises well without significant over-fitting. Minor fluctuations in both training and validation loss are visible, which is normal due to the stochastic nature of the training process.

7 Results of the Trained Model

7.1 Tesla Close Stock Price Prediction Plot

In the Actual vs. Predicted Values plot seen in figure 7:

Actual vs. Predicted Values: The blue line represents the actual closing stock prices of Tesla, while the red line shows the predicted values from the model.

Fit and Trends: The predicted line closely follows the actual line, capturing the general trends and patterns. There are some deviations, especially in the short-term fluctuations, but overall, the model tracks the long-term trends well.

Performance Over Time: The prediction accuracy seems consistent over the entire time period, with no significant divergence between actual and predicted values at any specific time-frame.

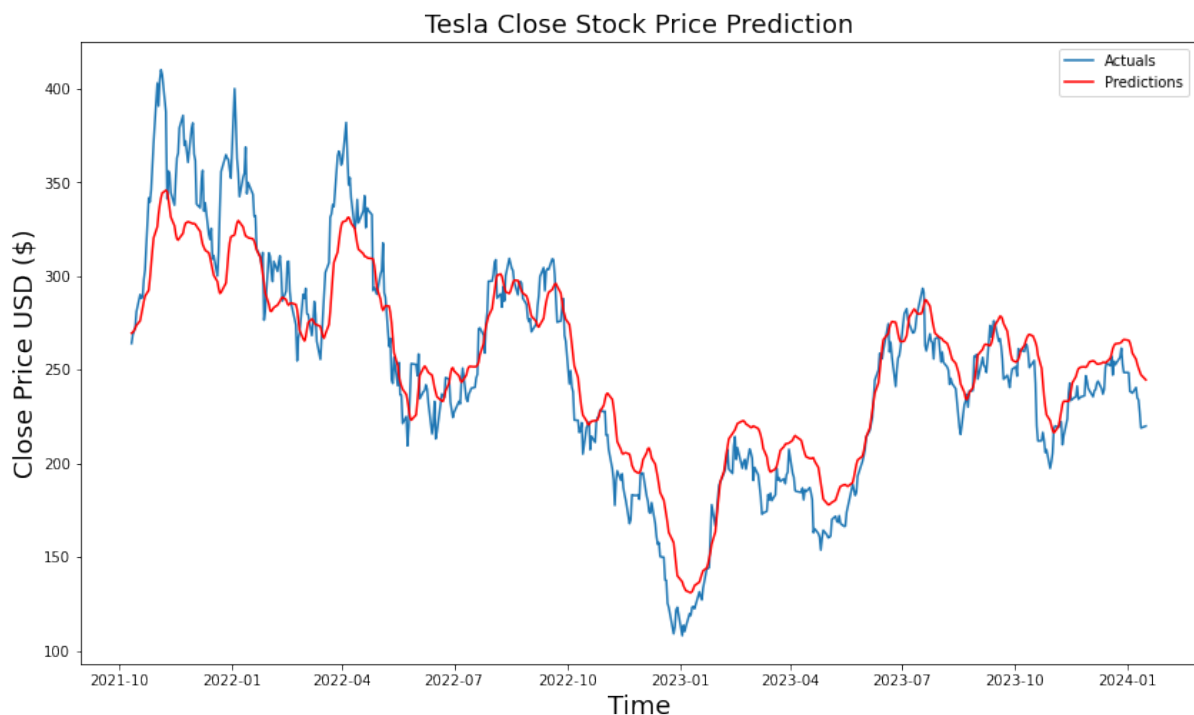


Figure 7: Tesla 'Close' Stock Price Predictions against Actuals Price

7.2 Prediction Errors Histogram

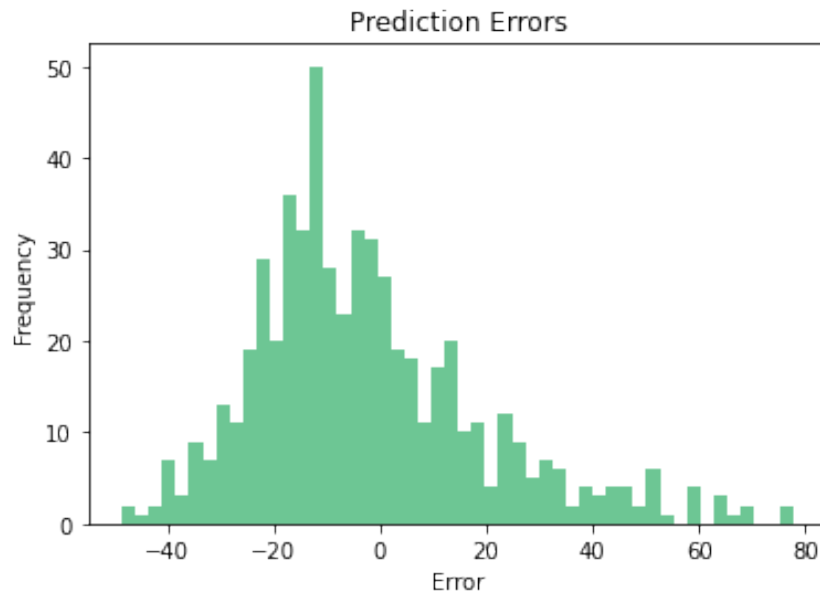


Figure 8: Histogram to show the frequency of the prediction errors

Distribution of Errors: The histogram in figure 8 shows the frequency of prediction errors (the difference between actual and predicted values).

Central Tendency: The errors are centred around -20, indicating a slight bias where the model tends to under predict the stock price by about \$20 on average.

Spread and Outliers: Most errors are within the range of -40 to 40, with fewer occurrences of larger errors. The distribution is somewhat skewed to the right, indicating some instances where the model over predicted the stock price by larger margins.

7.3 Error Metrics Analysis

Mean Squared Error (MSE):

- Value: 481.69
- Interpretation: MSE measures the average squared difference between actual and predicted values. A lower MSE indicates better model performance. Here, the MSE indicates some level of error, but considering the scale of stock prices, it is reasonable.

Mean Absolute Error (MAE):

- Value: 17.22
- Interpretation: MAE represents the average absolute difference between actual and predicted values. An MAE of 17.22 means that, on average, the model's predictions are off by about \$17.22. This is relatively small compared to the overall range of stock prices, suggesting good performance.

R-squared (R^2) Score:

- Value: 0.8733
- Interpretation: The R^2 score indicates the proportion of the variance in the dependent variable that is predictable from the independent variable(s). An R^2 score of 0.8733 means that approximately 87.33% of the variance in Tesla's closing stock price is explained by the model. This is a high R^2 value, indicating a strong fit and that the model explains most of the variability in the data.

Model Performance: The model performs well, capturing the overall trends in Tesla's stock price with a high R^2 score and relatively low MAE and MSE. The prediction errors are mostly small and normally distributed with a slight bias towards under prediction.

Practical Implications: The strong fit and low errors suggest that this model can be useful for predicting Tesla's stock price trends and making informed decisions based on these predictions. However, the slight under prediction bias and occasional larger errors should be considered, especially for short-term predictions or trading decisions.

In summary, the model shows strong predictive capability with a good balance between accuracy and reliability, making it a valuable tool for stock price prediction while still warranting cautious interpretation of its predictions.