Model Training Process for Stock Price Prediction

Jayasooriya J.A.N.S

March 10, 2025

1 Model Selection

For predicting stock prices, we will explore several machine learning models. Below are the models selected for evaluation:

- **Linear Regression:** A simple baseline model that assumes a linear relationship between features and the target variable.
- **Random Forest Regressor:** An ensemble learning method that uses multiple decision trees to make predictions.
- **Gradient Boosting Regressor:** A powerful ensemble technique that builds trees sequentially to correct errors of prior trees.

2 Data Preprocessing

Before training the models, the dataset undergoes preprocessing:

- **Handling Missing Values:**

  The dataset contains missing values in multiple columns, including the Date column. I identified missing dates by comparing adjacent entries. For example, there is a missing value between 1980-01-15 and 1980-01-17. These missing dates were filled with their respective correct values.

  The table below summarizes the missing values and unique counts for each column:

  For numerical columns such as High, Low, and Adj Close, I observed that the missing values occurred in places where adjacent values were not significantly different. Therefore, I applied linear interpolation to fill these missing values smoothly.

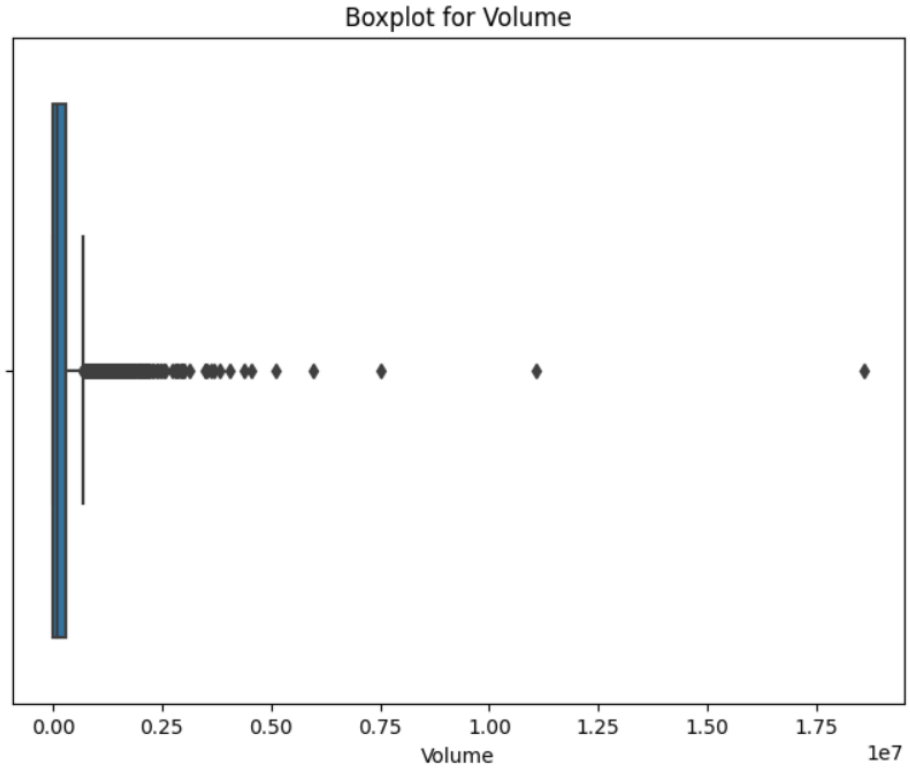
  Additionally, I noticed that some entries in the Open column had a value of 0, which is unrealistic in stock market data. These erroneous rows were removed to ensure data integrity.

- Extract temporal features such as the day of the week, month, and year from the date column.

	NaN count	Unique Count
Unnamed: 0	0	11291
Date	110	11181
Adj Close	93	7449
High	95	5669
Low	127	5644
Open	103	4769
Volume	145	5718

Table 1: NaN Count and Unique Count of Columns

- Normalize numerical features like the stock's closing price and volume.
- Volume feature:

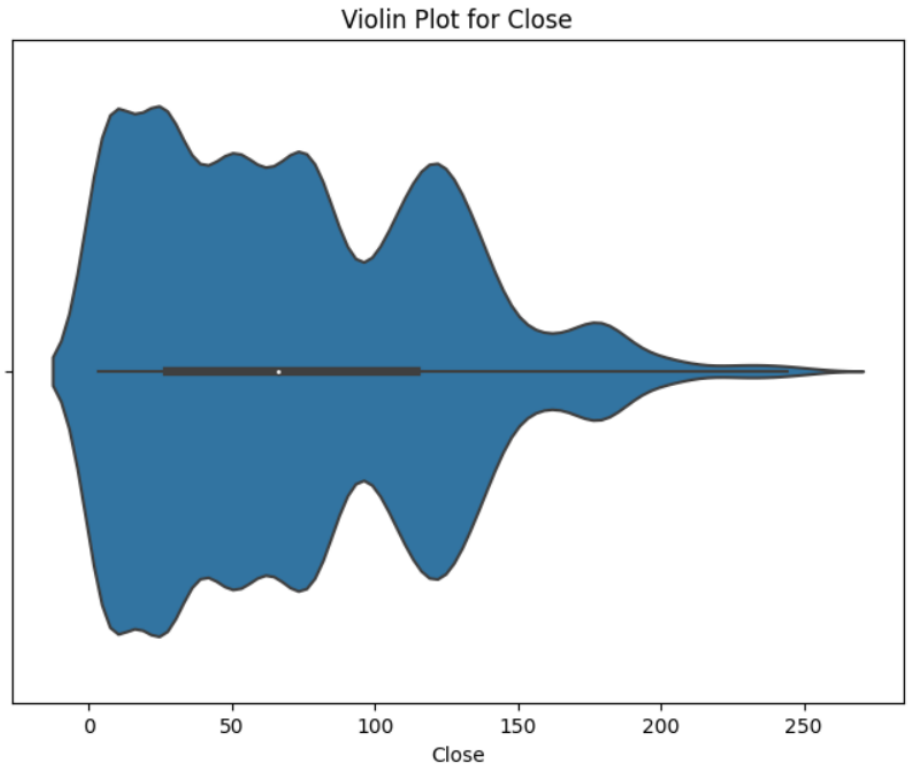


figureVolume values violin plot Here it is identified that, there are outliers

in this. Let

3 EDA

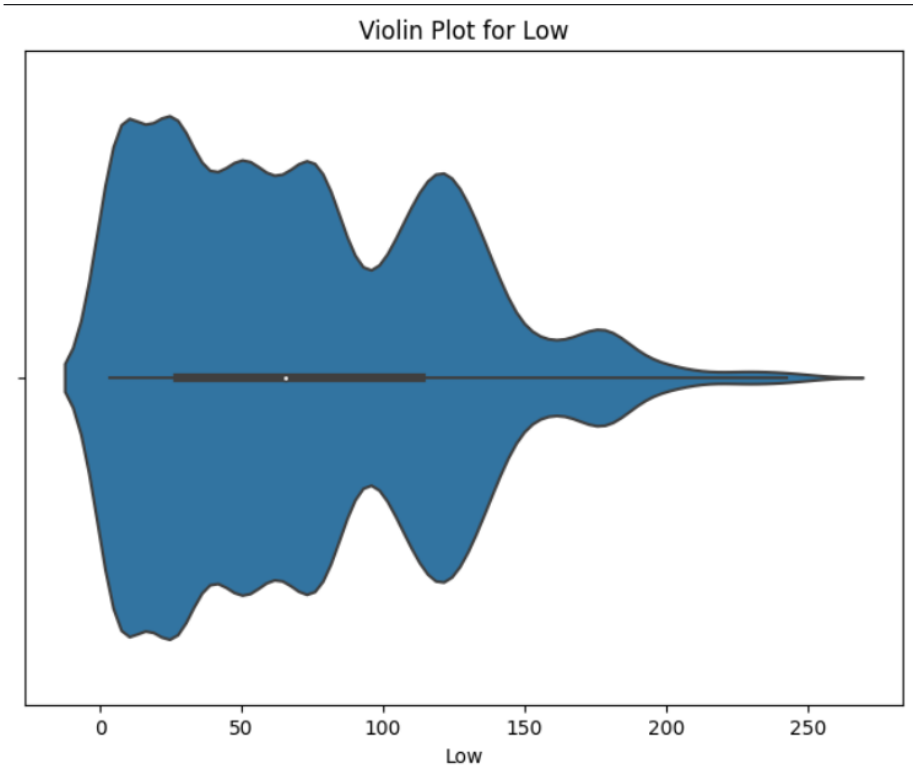
- **Analyze violin plots:**
  - I identified certain patterns between close values and other features. this is close feature :



figureClose values violin plot

Lets compare others with this.

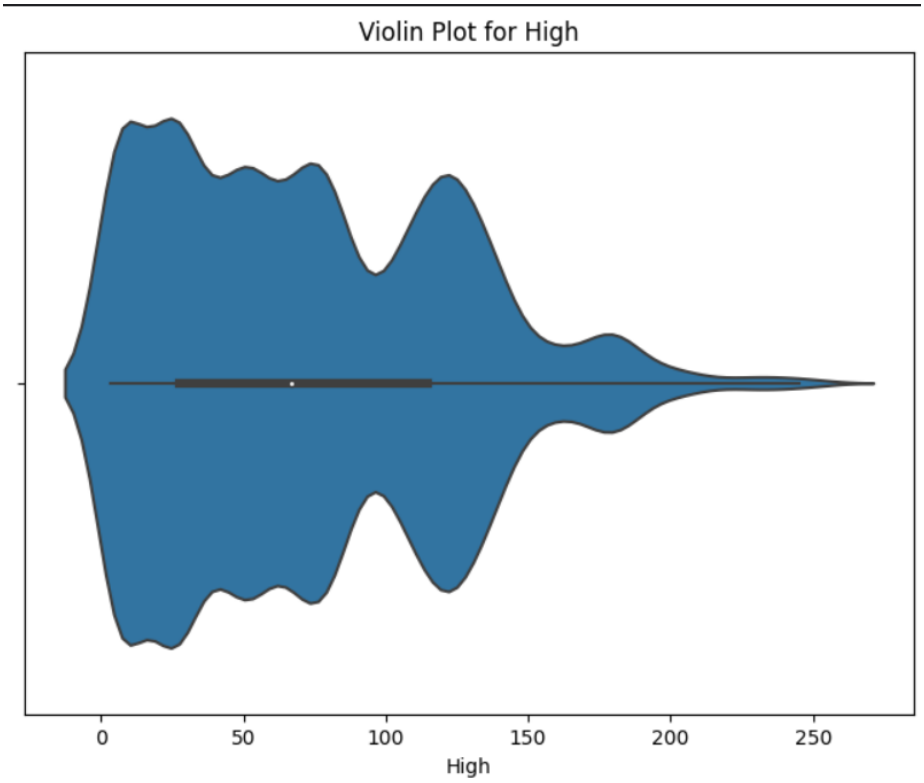
  - Low feature:



figureLow values violin plot

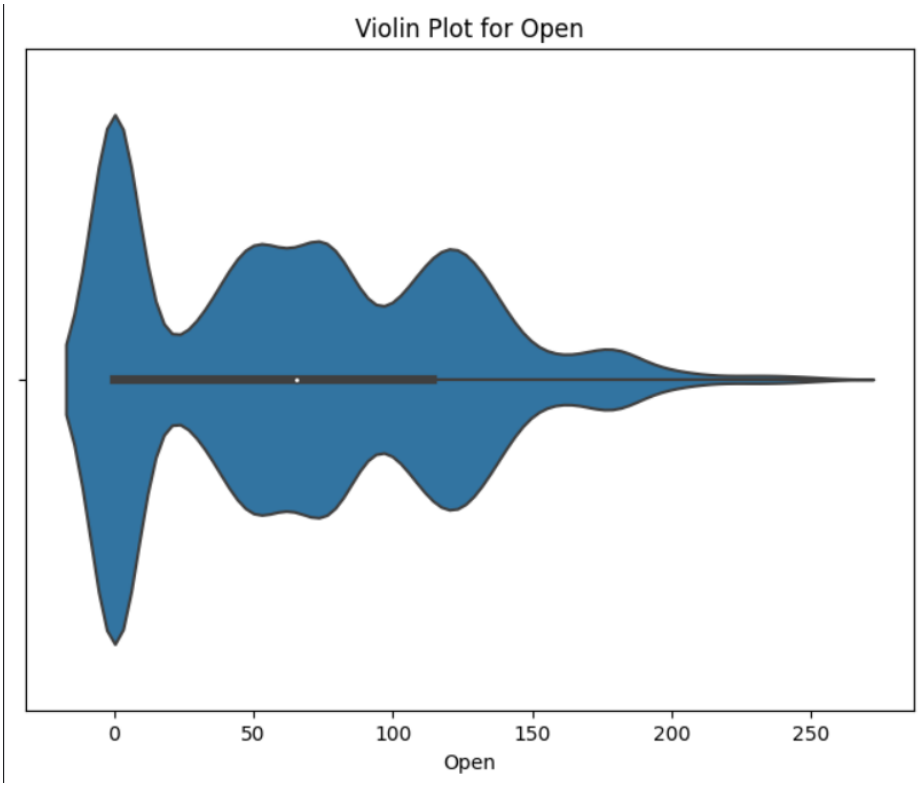
We can see there is same pattern between Low values and Close values. So this feature can be useful.

– High feature:



figureHigh values violin plot

– Open feature:



figureOpen values violin plot Considering this Open value violin plot,

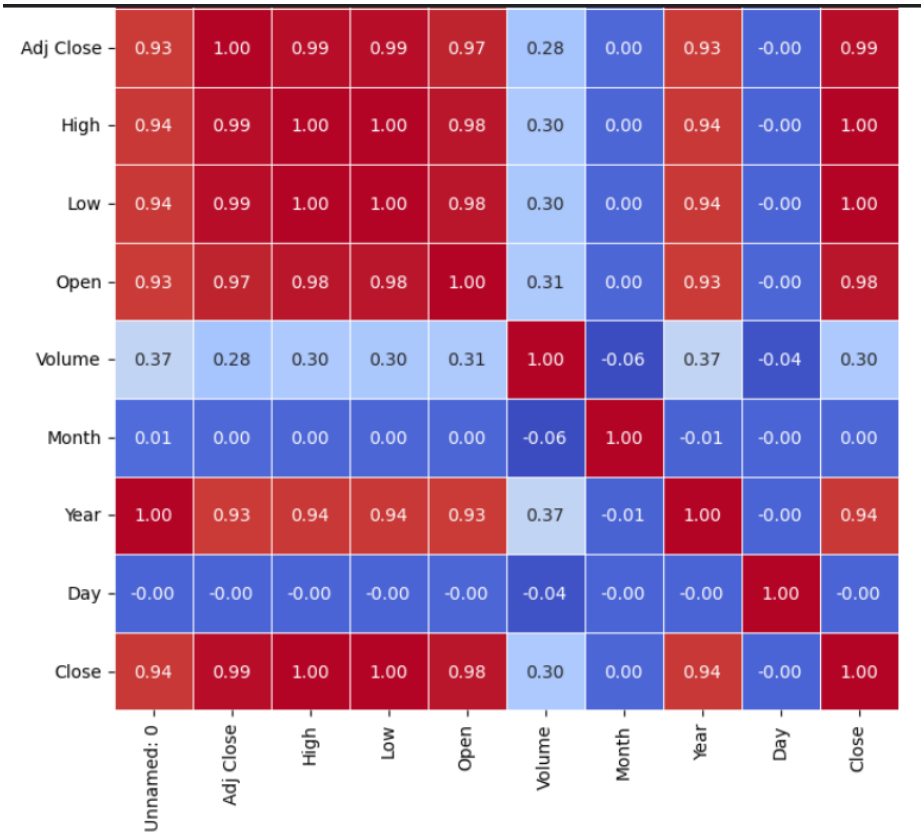
we can not identify strong relation ship. But we cannot decide it is useful or not.

– let's Anlyse Mitual infomations.

	MI Score
Adj Close	5.721763
Low	5.503116
High	5.201009
Open	3.504797
Unnamed: 0	3.481040
Year	2.465296
Volume	0.662591
Month	0.399250
Day	0.000000

Table 2: Mutual Information (MI) Scores for Features

– let's Anlyse Pearson Correlation Heatmap of Features and Close Price.



figureCorrelation Heat map Based on correlation values lot of useful

features can be identified.

4 Feature Engineering

- **Create lag features to incorporate historical stock prices:** To capture temporal dependencies in stock prices, I created lag features that included past closing prices. These features were intended to help the model learn patterns from previous days' prices. However, even after applying PCA for dimensionality reduction, I observed that the RMSE value increased instead of improving. This indicated that the lag features did not contribute positively to the model's predictive accuracy. As a result, I decided to remove them to maintain a more effective feature set.
- The above information show that Some low MI score features. We can train model by removing some of them .

	MI Score
PCA1	3.076907
Year	2.465296
Volume	0.662591
Month	0.399250
PCA2	0.056093
Day	0.000000

Table 3: Mutual Information (MI) Scores for Features

5 Model Training and Hyperparameter Tuning

Each model will be trained using the preprocessed data. Below is the general procedure followed for each model:

1. **Train-Test Split:** The data will be split into training and testing sets, ensuring that the data is split chronologically, with the training data used for training and the later data used for testing.

2. **Cross-Validation:** For each model, we will use cross-validation (e.g., K-fold cross-validation) to estimate the model's performance and tune hyperparameters to avoid overfitting.

3. **Hyperparameter Tuning:** We will perform a grid search or random search over a predefined hyperparameter space to find the optimal settings for each model. For example:

   - For `RandomForestRegressor`: The number of trees (`n_estimators`), maximum depth of the trees (`max_depth`), and minimum samples split (`min_samples_split`) will be tuned.
   - For `GradientBoostingRegressor`: The learning rate, number of boosting stages (`n_estimators`), and maximum depth of the individual trees will be optimized.

     (I got this as my best result after crossvalidation and hyper parameter tuning (Linear Regression - Mean RMSE: 0.3983, Std Dev: 0.0132))

6 Model Evaluation

To evaluate the performance of the models, the following metrics will be used:

- **Root Mean Squared Error (RMSE):** This metric will evaluate the model's accuracy in predicting stock prices by measuring the differences between the predicted and actual values.

7 Model Comparison and Final Selection

The models will be compared based on their performance in terms of RMSE, directional accuracy, and simulated trading performance. The model that delivers the best balance between prediction accuracy and practical trading value will be selected for deployment.

8 Conclusion

This section concludes the model training process, highlighting the chosen model and its expected performance in predicting future stock prices. Additional model improvements and strategies for further development will be discussed in future sections.