Stock Price Prediction

Beatrice Insalata Matteo Pancini Samuele Peri
insalata pancini samuelep | @kth.se

October 23, 2023

1 Problem Description

Predicting stock market movements is a challenging task that has intrigued researchers for decades. Stock prices are dependent on a complex interplay of variables related to company movements and macroeconomic conditions, trends, and global geopolitical events. These exhibited nonlinear, chaotic dynamics are difficult to model using traditional statistical methods. For investors and traders, a dependable model forecasting stock price movements can be a pivotal asset, facilitating well-informed decision-making. Advanced algorithms can discern complex patterns across heterogeneous variables at a scale that was previously unreachable. In this study, we attempt to predict Tesla (TSLA) shares' closing prices using extensive historical data and pertinent financial indices. Our objective extends beyond merely evaluating the efficacy of various algorithms; we also aim to delve into the underlying decision-making processes of these models by employing explainability techniques.

2 Tools

This project will leverage a relevant set of tools to enable powerful analytics on large-scale stock market data.

- **PySpark**: Used for distributed data processing and streaming analytics.
- **Spark SQL**: Used to query the pre-processed dataset obtained from various sources and perform initial exploration.

- **PySpark MLlib**: This library provided the tools necessary to build scalable machine learning models, with optimized algorithms.
- **Databricks**: To harness the scalability of Data Lakes. By connecting to various APIs for data retrieval, Databricks allowed us to process large datasets seamlessly, optimizing flexibility and performance.
- **yahoo_fin**: A module to scrape stock-related data from Yahoo finance.
- **pandas_datareader**: Used to fetch data from various internet sources, including Economic indicators from the Federal Reserve Economic Data (FRED).
- **Keras and Tensorflow**: For deep learning modelling.
- **Matplotlib**: For visualization and plotting of results.
- **LIME**: A tool for explaining predictions made by machine learning models.
- **Streamlit**: Framework used to create a web dashboard based on our code and deploy it for everyone.

Our project is mainly composed of 2 modules:

- **Jupyter Notebook**: Adaptable and versatile, this can be executed both on Databricks and locally. As a demonstrative example, we've employed Tesla stocks, though the project can be swiftly tailored for other stocks.
- **Web Application**: Built on the framework of Streamlit, this platform provides real-time analysis and insights.

3 Data

We used the yahoo_fin library to get **historical stock data** for Tesla (TSLA). Daily resolution stock price data including opening, closing, highest, lowest quotes along with trading volume will be retrieved from the Yahoo Finance API for the period between 2010 to present.

At the same time, a variety of technical and macroeconomic indicators has been sourced to capture contextual market trends and conditions. The

Federal Reserve Economic Data (FRED) database provided **macroeconomic indicators** like the S&P 500, DJIA, and others, contributing to significant additional metrics. These helped us understand the bigger financial picture and gave us more insights into TSLA's stock movements.

4 Method & Algorithms

4.1 Data Pre-Processing

Spark was employed to organize and clean the data. First, the historical stock data and the financial indicators were turned from FRED into Spark DataFrame for distributed processing. Missing value imputation was conducted by forward-filling where needed, as well as renaming features, and creating a combined date column. After that, stock data was combined with FRED's data by joining both DataFrames on their dates. The finalized set was sorted by date and cleaned from any missing information. Lastly, important details, like dates and closing stock prices, were extracted for later use.

4.2 Data Exploration

Leveraging Spark SQL, summary statistics are calculated across key variables in the distributed DataFrame, to get a sense of the statistics around Tesla's closing stock prices. Firstly, closing prices' lowest, highest, average, and standard deviation were investigated. Then, histograms sliced into bins offer descriptive views using Matplotlib, with each bar representing a range of closing prices and their height, showing how often those prices occurred.

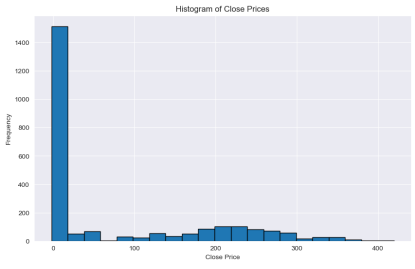

Figure 4.1: Closes Price Ranges of Tesla Stock

4.3 Models

Machine learning (ML) and deep techniques were leveraged to effectively capture patterns and produce accurate stock movement forecasts.

4.3.1 Machine Learning

MLlib, a Spark machine learning library, provides valuable tools to train models and predict Tesla's stock closing prices. Initial data preparation was conducted to prepare the data, and ensure all relevant features were included.The data was then split, reserving 80% for training and 20% for testing. Three machine learning models were employed to achieve predictions.

- **Linear Regression** fits a single-degree-of-freedom model to quantify the general trend. While simple, it provides a coherent benchmark for analysis.
- **Random Forests**, an ensemble of decorrelated decision trees, can model multivariate non-linear relationships through its adaptive construction of an immense number of weak learners. Spark natively parallelizes training across a cluster to scale efficiently.
- **Gradient Boosted Trees** sequentially combine weak prediction rules via steepest descent, minimizing residuals on difficult regions of the input space.

4.3.2 Deep Learning

An LSTM neural network offers distinct advantages through its unique architectural design. Before building the model, extensive preprocessing of the dataset was needed, including data vectorization, Min-Max scaling for input normalization, and a different dataset split, more unbalanced towards training (90% vs 10%). The reason behind this choice is that while tools like linear regression are able to capture linear correlations in stock price movements, LSTM needs more data to identify long-term linear patterns. The model includes two LSTM layers (with 32 and 16 units respectively), and a Dropout layer for regularization, to prevent the high degree of overfitting on the training set. Finally, a Dense layer produces the predicted stock price value.
For each model, the dataset was leveraged for training, making predictions,

and evaluating accuracy based on the discrepancy between predicted and actual values, measured by the 'Root Mean Square Error' (RMSE).

5 Results

The effectiveness of the four predictive models - Linear Regression, Random Forest, Gradient Boosted Tree, and LSTM - can be inspected via RMSE. It provides a snapshot of the prediction accuracy, with lower values indicating better performance.

- Linear Regression RMSE: 3.812
- Random Forest RMSE: 108.762
- Gradient Boosted Tree RMSE: 50.452
- Long-Short-Term Memory RMSE: 0.057

Figure 5.1 illustrates the actual close prices and the predictions from each ML model over time, offering a visual comparison to easily spot where predictions align or diverge from real values. As seen in the graph, random forests struggle with the highly nonlinear temporal patterns in stock data, while individual tree learners don't explicitly handle time series nature of data. The linear regression model was able to capture the general trend in stock prices.

Figure 5.1: Performances of Models

After training for 50 epochs, the LSTM achieved optimal performance on the test set, demonstrating its ability to accurately forecast stock prices based on patterns learned over long sequences of historical data. This represented a sizable improvement over other models and confirmed the value of recurrent networks for time series forecasting problems.

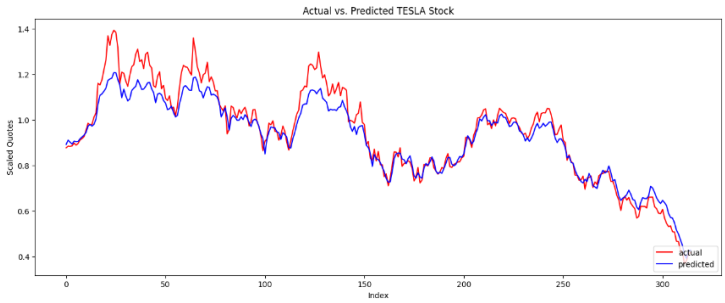

Figure 5.2: Close-Up LSTM Performance on Scaled Test Split

5.1 Explainability

Two crucial aspects of model evaluation are understanding feature importance and model explanations. Firstly, Feature Importance was tackled. By analyzing this,insight can be gained into which factors had the strongest sway over Random Forest and Gradient Boosted Trees' decisions.

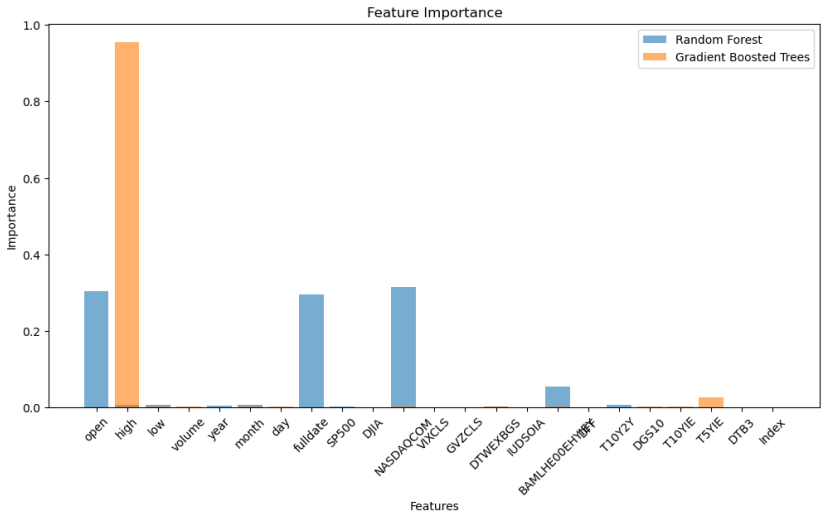

Figure 5.3: Feature Importance Graph

Next, LIME (Local Interpretable Model-Agnostic Explanations) was exploited to gain insight into how individual predictions were made. It's like peeking behind the scenes to see why a model predicts a certain way. LIME revealed which features most influenced predictions from ML models. Positive weights indicated features that pushed the prediction higher, while negative weights lowered it.

While the numbers and accuracy of models are valuable, a deeper understanding felt needed to tackle the reasons behind scores. By embracing both these angles, the aim was not only measure our models' success but also demystify the thought process behind their predictions, given how trustworthy indicators are even more valuable for real-world applications.

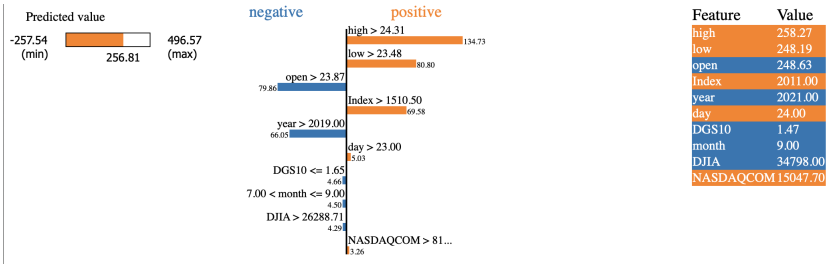

Figure 5.4: Lime Explaination

6 How To Run?

To ensure flexibility and ease of access for users, we've made provisions for the code to be executed in multiple environments. Here's a step-by-step guide on how to run it:

- Locally via Jupyter Notebook: Use the Jupyter Notebook (attached) to execute the code cell by cell (it should be already executed).
- On Databricks: Databricks cloud notebook
- As a Streamlit Web Application: Clone the repository to your local machine. Ensure you have Streamlit installed; if not, install it using pip. Navigate to the directory via the terminal or command prompt and run the command: streamlit run app.py. After executing the above, a browser window should open with the web application interface, ready for interaction.