Predicting Tesla's Stock Price

Shayan Ghafoori
ghafoorishayan@gmail.com

June 1, 2024

1 Introduction

Predicting stock prices is a task that has long fascinated investors, analysts, and researchers. The ability to forecast the movement of stock prices accurately can lead to significant financial gains and is a topic of interest in both academia and industry. In the past two decades – with the explosion of available data and advancements in machine learning techniques – the movement of stock prices has been largely influenced by algorithmic training. Concurrently, over the past decade, there is one stock that has particularly stood out for its volatility, market disruption, and eccentric CEO: Tesla.

Moving average, linear regression, KNN (k-nearest neighbor), Auto ARIMA, and LSTM (Long Short Term Memory) are the most common methods of predicting stock prices [1]. Random forest decision trees are also used to predict options prices [2]. Except for linear regression, many of these methods are used for short-term analysis of stocks and fail to generalize the long-term trends (LSTM is an example of this). In this paper, I aim to create a model that successfully predicts the price of the Tesla stock by leveraging a variety of data sources and employing a linear regression model with a maximum likelihood estimation (MLE) approach. Drawing inspiration from the rich history of financial analysis and statistical modeling, the goal is to build a robust predictive model that can provide valuable insights into Tesla's future stock price movements.

In the following sections, I will detail the methodology, data preprocessing steps, model training procedure, and evaluation metrics. Additionally, I discuss the implications of my findings and avenues for future research in the domain of stock price prediction. Through this, I hope to provide a comprehensive framework for predicting Tesla's stock price and contribute to the broader discourse on financial forecasting methodologies and the evolving field of quantitative finance, ultimately enhancing our understanding of the intricate mechanisms driving Tesla's stock market behavior.

Datasets

- Tesla Stock Price Data: (https://www.kaggle.com/datasets/guillemservera/tsla-stock-data?select=tsla_raw_data.csv)
- Tesla Earnings Data: https://www.alphaquery.com/stock/TSLA/earnings-history
- Tesla Options Data: (https://fred.stlouisfed.org/series/FEDFUNDS)
- Interest Rates Data: https://www.kaggle.com/datasets/federalreserve/interest-rates

2 Methods

The process for predicting Tesla's stock price involves the construction of a linear regression model utilizing a maximum likelihood estimation (MLE) approach, with a Gaussian distribution serving as the likelihood function. We will use Recursive Feature Elimination with k-fold cross-validation (RFECV) using mean-squared error as our evaluation metric to choose the optimal features for our model. This methodology incorporates various sources of data, including time series data of the $TSLA stock spanning from 2016 to 2023, alongside options data, earnings data, and interest rates observed during this period. Given the inherent volatility in stock prices, employing an MLE approach to linear regression allows for a confidence

interval in our predictions rather than limiting it to a single value, enhancing the robustness of our forecasts. To determine the most suitable model, Recursive Feature Elimination (RFE) with K-fold cross-validation (using 10 folds) will be leveraged to evaluate model performance across different orders of polynomials. After selecting the optimal model, we will conduct a final evaluation of its performance by testing it against unseen Tesla stock data from 2024.

2.1 Data Collection and Setup

As stated before, we will utilize time series price and options data, earnings data, and interest rates observed from 2016 to 2023 to make our predictions. Since the options data and pricing data are both time series data, incorporating the datasets into one is simply a matter of joining the two datasets by their dates. The interest rates and earnings data will be imputed to match the time series data. For instance, if the interest rates and/or earnings data were updated 01/01/2020, 04/01/2020, and 07/01/2020, we will impute the respective interest rates and/or earnings data to each observation from 01/01/2020 - 04/01/2020 and 04/01/2020 - 07/01/2020. In addition to this, we will be adding two binary dummy variables to each observation representing a 7-day span for interest rates and earnings data. The observations whose dates were within 7-days of interest/earnings data being released will take a value of 1, otherwise they will take a value of 0. For instance, if earnings data was released on 01/07/2020, observations from 01/01/2020 - 01/13/2020 will be marked with the value 1 (please note that there will be two independent variables for interest rates and earnings data within the past 7-days). The reason behind this is to account for sudden volatility in options data and stock price – as they tend to be most volatile around earnings/interest dates. This is particularly true of industries that rely on lower interest like housing, or in this case, the auto industry – which relies heavily on auto loans for sales.

2.2 Algorithm

Let $\mathbf{x}_n$ be our feature vector representing a bias term and our predictor variables.

$$\mathbf{x}_n = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix},$$

We can explore higher orders of $\mathbf{x}_n$ and different combinations of that order. For instance, for order k, and 3 predictor variables, the number of rows in the feature vector is given by

$$\mathbf{x}_n = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \text{ for } k = 1, \quad \mathbf{x}_n = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ x_1^2 \\ x_1x_2 \\ x_1x_3 \\ x_2^2 \\ x_2x_3 \\ x_3^2 \end{bmatrix} \text{ for } k = 2$$

An important observation to highlight is the hierarchical relationship between different orders. Each increase in order encompasses its preceding order. For instance, it's evident that the vector representing order 1 includes all elements of the vector representing order 2, this will come up later.

Let the matrix $\mathbf{X}$ be an $N \times D$ matrix (where N is the number of observations) defined as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix}$$

Let $\hat{\mathbf{w}}$ be a vector of size D that represents the estimated parameters in our model, and let $\mathbf{p}$ be a vector of size N that represents the stock prices.

$$\hat{\mathbf{w}} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_N \end{bmatrix}$$

Since we are using Gaussian likelihood, our density function for a single observation is $p(p_n|\mathbf{x}_n, \mathbf{w}, \sigma^2) = \mathcal{N}(\mathbf{w}^T\mathbf{x}_n, \sigma^2)$.

The likelihood function becomes

$$\mathcal{L} = p(\mathbf{p}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^{N} p(p_n|\mathbf{x}_n, \mathbf{w}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{w}^T\mathbf{x}_n, \sigma^2).$$

When maximizing this likelihood, we obtain

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{p}, \quad \hat{\sigma}^2 = \frac{1}{N}(\mathbf{p}^T\mathbf{p} - \mathbf{p}^T\mathbf{X}\hat{\mathbf{w}}).$$

Our predicted stock price and interval can then be computed as:

$$\hat{p}_{\text{new}} = \mathbf{x}_{\text{new}}^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{p} = \mathbf{x}_{\text{new}}^T\hat{\mathbf{w}}$$

$$\sigma_{\text{new}}^2 = \sigma^2\mathbf{x}_{\text{new}}^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_{\text{new}} = \sigma^2\mathbf{x}_{\text{new}}^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_{\text{new}}$$

where σ^2 is the true variance of the dataset noise. In its place, we can use our estimate, $\hat{\sigma}^2$.

Now that we understand the mathematical underpinnings of our model, the subsequent step involves generating various ordered $\mathbf{X}$ matrices through different ordered feature vectors. We will then employ Recursive Feature Elimination coupled with k-fold cross-validation (RFECV) using 10 folds to discern the optimal model by iteratively discarding features that contribute minimally to predicting our stock price. Our RFECV metric is the average mean-squared error (MSE) across folds to recursively eliminate any extraneous features (it uses least square error linear regression), which is the same metric we use to determine our $\hat{\mathbf{w}}$ vector.

Earlier, we acknowledged the hierarchical relationship among increasing orders of the feature vector. While it might seem logical to construct a feature vector with the highest possible order and utilize RFECV to select optimal features, this approach carries a risk. As the order increases, the feature vector expands exponentially, leading to the introduction of numerous correlated features, many of which may be unnecessary. This heightened complexity escalates the potential for RFECV overfitting, even with cross-validation.

Hence, our strategy is to generate a feature vector and conduct RFECV for each order up to order 8. This approach curtails the number of correlated factors and enables the selection of optimal features at each order. Conceptually, we aim to determine the optimal features within a maximum order, thereby mitigating model complexity.

The pseudocode outlining this process is presented below:

Algorithm 1 RFECV with 10 Folds and MSE

Require: $\mathbf{X}$: Design matrix, $\mathbf{y}$: Target vector
Ensure: Selected features $\mathbf{X}_{\text{selected}}$
  function RFECV($\mathbf{X}$, $\mathbf{y}$)
    Initialize list of selected features $\mathbf{X}_{\text{selected}} \leftarrow \mathbf{X}$
    Initialize best mean squared error (MSE) best_mse $\leftarrow \infty$
    for $k \leftarrow 1$ to n_{features} **do**
      Perform k-fold cross-validation with $k = 10$
      Split $\mathbf{X}_{\text{selected}}$ and $\mathbf{y}$ into k folds
      Initialize list of mean squared errors mse_list $\leftarrow$ []
      for each feature f in $\mathbf{X}_{\text{selected}}$ **do**
        Train model with all features except f
        Compute mean squared error (MSE) using cross-validation
        Append MSE to mse_list
      end for
      Compute average MSE from mse_list
      if Average MSE is lower than best_mse **then**
        Update best_mse to average MSE
        Select the features corresponding to the lowest MSE
      end if
    end for
    return $\mathbf{X}_{\text{selected}}$
  end function

Algorithm 2 Maximum Likelihood Estimation (MLE) Linear Regression

Require: $\mathbf{X}$: Design matrix, $\mathbf{p}$: Target vector
Ensure: $\hat{\mathbf{w}}$: Estimated coefficients, $\hat{\sigma}^2$: Residual variance
  function MLE($\mathbf{X}$, $\mathbf{p}$)
    Compute estimated coefficients $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{p}$
    Compute residual vector $\mathbf{r} = \mathbf{p} - \mathbf{X}\hat{\mathbf{w}}$
    Compute residual variance $\hat{\sigma}^2 = \frac{1}{N}(\mathbf{r}^T\mathbf{r})$ ▷ Where N is the number of observations
    return $\hat{\mathbf{w}}$, $\hat{\sigma}^2$
  end function

Algorithm 3 Model Selection with RFECV and MLE

Require: $\mathbf{X}$: Initial Design Matrix of feature vectors, $\mathbf{p}$: Output vector
Ensure: Best model and predictions
  function ModelSelection($\mathbf{X}$, $\mathbf{p}$)
    Scale feature vectors in $\mathbf{X}$ using z-scaling
    Scale output vector $\mathbf{p}$ using z-scaling
    Initialize best model variables best_model $\leftarrow$ None, best_mse $\leftarrow \infty$
    for each order n from 1 to N **do**
      Run RFECV on $\mathbf{X}$ to obtain selected features
      Compute $\hat{\mathbf{w}}$ and $\hat{\sigma}^2$ using MLE on selected features
      Compute $\hat{p}_{\text{new}}$ and σ^2_{new} using $\mathbf{x}_{\text{new}}$ and $\hat{\mathbf{w}}$, $\hat{\sigma}^2$
      Store model information
    end for
    return best_model
  end function

3 Results

The options data proved to be too much information and very computationally expensive to incorporate into our model, thus we omitted it. The variables we will use to predict our data are

- **Date**: Represents the date of the data point.
- **Volume**: Refers to the trading volume for the corresponding date.
- **20-Day-Vol**: Represents the average 20-day trading volume, indicating the average volume over the past 20 days.
- **Interest**: Indicates the federal interest rate, representing the interest rate set by the Federal Reserve.
- **I-Bool**: A binary variable derived from 'Interest', representing whether there was an interest event within the past 7 days. It is coded as 1 if an interest event occurred and 0 otherwise.
- **Earnings**: Indicates the quarterly earnings per share (EPS), representing the company's earnings per share for the quarter.
- **E-Bool**: A binary variable derived from 'Earnings', representing whether there was an earnings event within the past 7 days. It is coded as 1 if an earnings event occurred and 0 otherwise.

The final model is of order 2 and is defined as:

$$\begin{aligned}\hat{z}_{\text{scaled}}(\text{Stock Price}) = {} & 1.11992 \times z_{\text{scaled}}(\text{Date}) \\ & + 0.48305 \times z_{\text{scaled}}(\text{Date}^2) \\ & - 0.24540 \times z_{\text{scaled}}(\text{Date * Interest}) \\ & - 0.16302 \times z_{\text{scaled}}(\text{Volume * Earnings}) \\ & - 0.11550 \times z_{\text{scaled}}(\text{Interest}^2)\end{aligned}$$

Not only did the second order model have the lowest mean score after peforming RFECV, it also performed the best on the new 2024 data (unseen testing set)

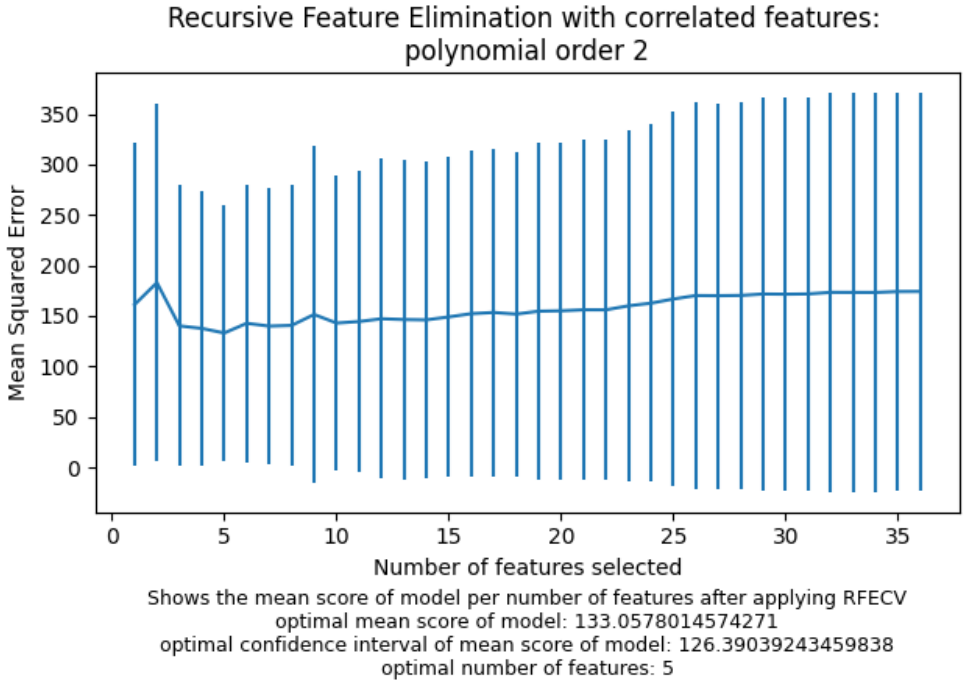
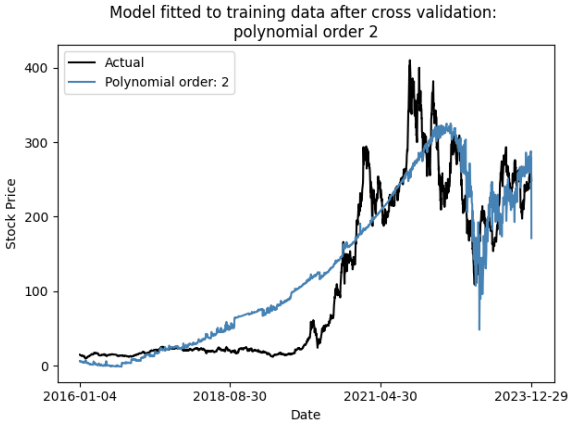
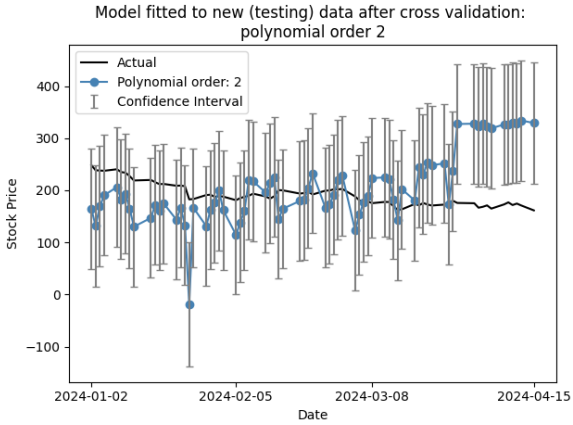


Figure 1: RFECV mean squared error results and their confidence intervals on each feature

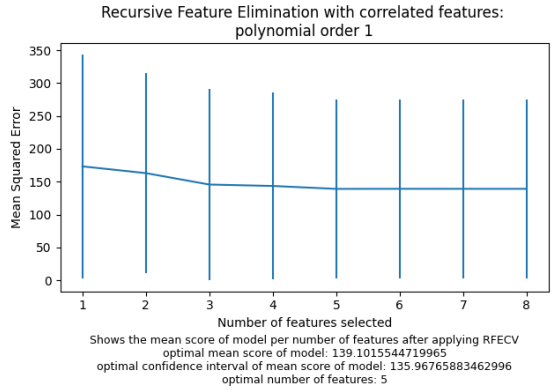


(a) How the model performed on the training data (it was trained on 10-folds of this data, this is how it fits on the training data as a whole)

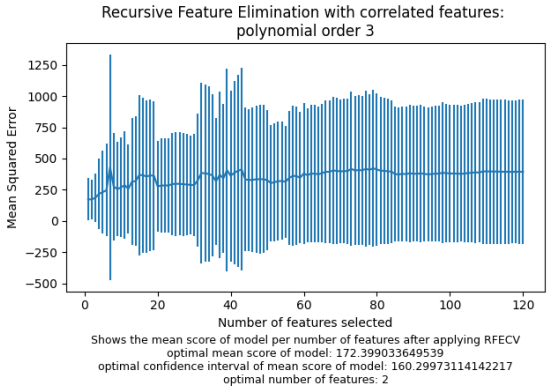


(b) How the model performed on new 2024 Data

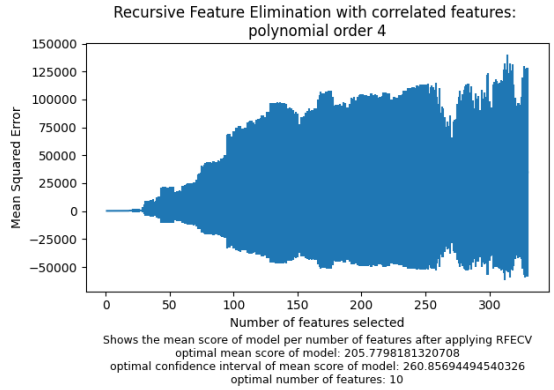
Figure 2: Model performance on training data and testing data



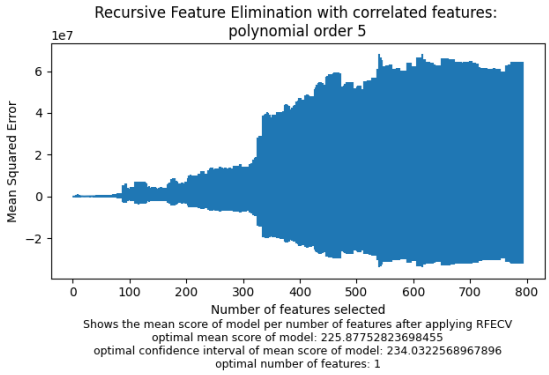
(a) Optimal features: **Date**, **Volume**, **20-Day-Vol**, **Interest**, **Earnings**



(b) Optimal features: **Date**, **Date^2**



(c) Optimal features: **Date**, **Date^2**, **Date * Interest**, **Earnings^2**, **Date^2 * E-Bool**, **Date * Interest * Earnings**, **Interest * Earnings^2**, **Date^2 * Earnings^2**, **Date^2 * E-Bool^2**, **Date * Interest * Earnings^2**



(d) Optimal features: **Date^3**

Figure 3: RFECV results for orders 1, 3, 4, 5

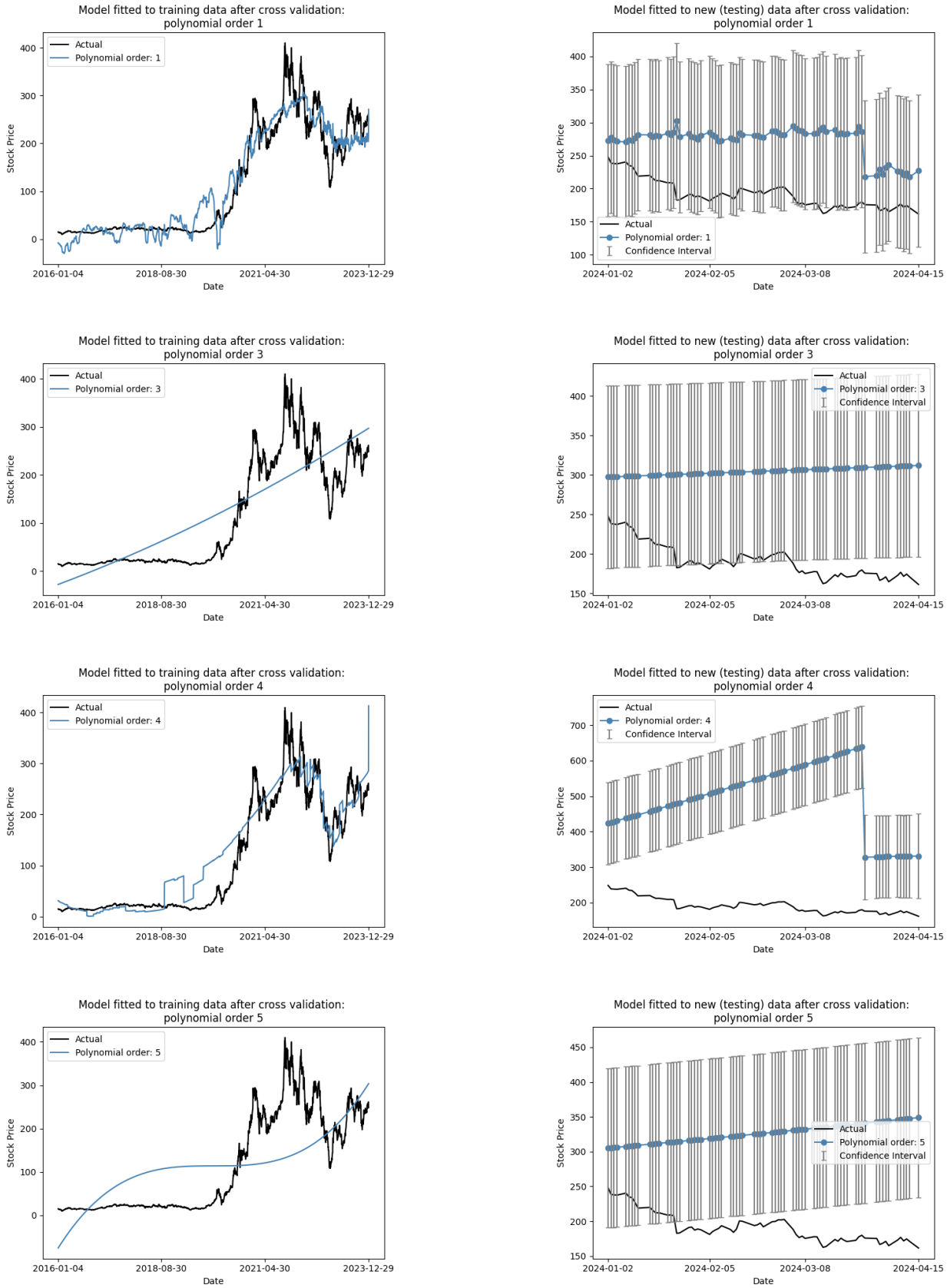


Figure 4: Model performance on training data and testing data for orders 1, 3, 4, 5

4 Discussion

While the results for the best model order and the number of features selected for each order were as expected (lower complexity models tend to perform better across multiple folds), it is somewhat disappointing that our binary variables did not significantly contribute to our model's performance.

Initially, we chose an MLE approach to linear regression with Gaussian noise with the intention of adding confidence intervals to our predictions. However, the resulting confidence intervals turned out to be quite large, rendering them not very useful. Notably, there wasn't much variation in the confidence intervals across different point estimates. The majority of confidence interval values fell between 110 and 120, which may appear uniform due to the larger scaling of the y-axis.

Unsurprisingly, 'Date' emerged as the most influential predictor across all models. While this correlation is expected in stock predictions, it's nonetheless disappointing that 'Earnings' did not exhibit a stronger presence. This might be attributed to the fact that TESLA's stock typically trades well above its price-to-earnings ratio and is considered a highly speculative stock. As such, much of the price movement is already accounted for both before and after earnings announcements, possibly diminishing the impact of earnings on stock price. It's possible that a stock like NVIDIA, with a different market profile, would exhibit a stronger relationship with its earnings.

Looking ahead, future analyses could consider factors such as earnings predictions versus actual earnings. Understanding the discrepancies between predicted and actual earnings could shed light on their impact on stock prices. For instance, if predictions were consistently low, the effect of poor earnings might be negligible, and vice versa.

In summary, while our model performed reasonably well in predicting long-term trends, it's important to note that this success is somewhat expected given the strongly positive relationship between date and price. Trying to build a model that reliably predicts stock prices is an incredibly difficult task. If it were as simple as crunching numbers and making predictions, we'd all be millionaires by now! But the reality is, the stock market is a complex beast, influenced by countless variables and unpredictable human behavior. So, while we've made strides in our modeling efforts, it's important to remember that there's no magic formula for guaranteed success in the stock market. It's a journey of learning and adaptation, where even the best models can only take us so far.

REFERENCES

[1] *Stock Prices Prediction Using Machine Learning and Deep Learning*. URL: https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/#:~:text=Moving%20average%2C%20linear%20regression%2C%20KNN,used%20to%20predict%20stock%20prices..

[2] *Applications of machine learning in options trading*. URL: https://medium.com/@fhuqtheta/applications-of-machine-learning-in-options-trading-b416c5a67831.