**Improving Stock Price Forecasting by Feature Engineering**

[[Automata](https://medium.com/@autognosi?source=post_page-----8a5d0be2be96--------------------------------)](https://medium.com/@autognosi?source=post_page-----8a5d0be2be96--------------------------------)

[[𝐀𝐈 𝐦𝐨𝐧𝐤𝐬.𝐢𝐨](https://medium.com/aimonks?source=post_page-----8a5d0be2be96--------------------------------)](https://medium.com/aimonks?source=post_page-----8a5d0be2be96--------------------------------)

[Automata](https://medium.com/@autognosi?source=post_page-----8a5d0be2be96--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2Fa543da0e3424&operation=register&redirect=https%3A%2F%2Fmedium.com%2Faimonks%2Fimproving-stock-price-forecasting-by-feature-engineering-8a5d0be2be96&user=Automata&userId=a543da0e3424&source=post_page-a543da0e3424----8a5d0be2be96---------------------post_header-----------)

Published in

[𝐀𝐈 𝐦𝐨𝐧𝐤𝐬.𝐢𝐨](https://medium.com/aimonks?source=post_page-----8a5d0be2be96--------------------------------)

·

10 min read

·

Jul 18

89

2



Stock Forecasting by Author with DreamShaperV6

In this article, I want to share with you how I tackled the problem of predicting the value of the stock at the next day’s close, using daily price data for indexes tracking stock exchanges from all over the world. This is a very difficult and ambitious task, as stock prices are influenced by many factors, such as supply and demand, news, sentiment, expectations, trends, and more. Moreover, the financial market is highly dynamic and volatile, making it hard to capture and model its behavior.

To solve this problem, I used a combination of **data engineering**, **feature engineering**, **machine learning**, and **hyperparameter optimization** techniques. I will explain in detail each step of my process, from data collection and preprocessing, to feature augmentation and selection, to model comparison and selection, to performance evaluation and improvement. I hope you will find this article interesting and useful for your own projects.

**Data Collection and Preprocessing**

The first step of any data science project is to collect and preprocess the data. For this project, I used daily price data for indexes tracking stock exchanges from all over the world, such as the United States, China, Canada, Germany, Japan, and more. The data was all collected from **Yahoo Finance**, which had several decades of data available for most exchanges.

The data consisted of seven columns: **Index**, **Date**, **Open**, **High**, **Low**, **Close**, and **Adj Close**. The Index column contained the ticker symbol for each index, such as ^GSPC for the S&P 500, ^DJI for the Dow Jones Industrial Average, ^IXIC for the Nasdaq Composite, and so on. The Date column contained the full date of observation, in the format YYYY-MM-DD. The Open column contained the opening price of the index on that day, while the High and Low columns contained the highest and lowest prices during the trading day. The Close column contained the closing price of the index on that day, while the Adj Close column contained the closing price adjusted for dividends and stock splits.

The first thing I did was to check the quality and consistency of the data. I looked for missing values, outliers, duplicates, and errors. I found that some indexes had missing values for some days, due to holidays or other reasons. I decided to fill these missing values using linear interpolation, which is a simple and reasonable method for time series data. I also checked the distribution and range of each column, and found that they were mostly normal and reasonable, except for some extreme values that could be attributed to market crashes or rallies.

The next thing I did was to convert the prices to a common currency, since they were quoted in terms of the national currency of where each exchange is located. For example, the S&P 500 was quoted in US dollars, while the Nikkei 225 was quoted in Japanese yen. To make them comparable and consistent, I used the exchange rates from Yahoo Finance to convert them to euros, which is the currency I’m most familiar with. This step also helped to reduce the effect of currency fluctuations on the prices.

The last thing I did was to split the data into three sets: **training**, **validation**, and **test**. The training set contained 80% of the data, from January 1st, 1970 to December 31st, 2018. The validation set contained 10% of the data, from January 1st, 2019 to December 31st, 2019. The test set contained 10% of the data, from January 1st, 2020 to December 31st, 2020. I used these sets to train, tune, and evaluate my models.

**Feature Engineering**

The next step of my project was to engineer new features from the existing data, to capture more information and patterns that could help my models to make better predictions. Feature engineering is one of the most important and creative aspects of data science, as it requires domain knowledge, intuition, and experimentation.

For this project, I went from the previous seven columns to these features:

* **Year**: The year of the observation, extracted from the Date column.
* **Month**: The month of the year, extracted from the Date column.
* **Day**: The day of the month, extracted from the Date column.
* **Daily Variation**: The difference between the High and Low columns, divided by the Open column. This feature represents the volatility of the index on that day.
* **TimeStamp**: The number of seconds elapsed since January 1st, 1970 00:00:00 UTC, calculated from the Date column. This feature represents the temporal order of the observations.
* **Index Hash**: A numerical representation of the Index column, obtained by applying a hash function. This feature encodes the identity of each index in a compact and unique way.
* **Daily Return**: The percentage change in the Close column from the previous day’s Close column. This feature represents the performance of the index on that day.
* **7-Day SMA**: The 7-day simple moving average of the Close column. This feature represents the short-term trend of the index.
* **7-Day STD**: The 7-day standard deviation of the Close column. This feature represents the short-term variability of the index.
* **SMA + 2 STD**: The 7-day SMA plus two times the 7-day STD. This feature represents the upper bound of a confidence interval for the index.
* **SMA — 2 STD**: The 7-day SMA minus two times the 7-day STD. This feature represents the lower bound of a confidence interval for the index.
* **High — Close**: The difference between the High and Close columns, divided by the Open column. This feature represents the downward pressure on the index on that day.
* **Low — Open**: The difference between the Low and Open columns, divided by the Open column. This feature represents the upward pressure on the index on that day.
* **Cumulative Return**: The cumulative percentage change in the Close column from the first observation in the training set. This feature represents the long-term performance of the index.
* **14-Day EMA**: The 14-day exponential moving average of the Close column. This feature represents a smoother and more responsive version of the SMA.
* **Close % Change**: The percentage change in the Close column from the previous day’s Close column. This feature is similar to Daily Return, but without scaling by 100.
* **Close Change**: The difference between the Close and previous day’s Close columns. This feature is similar to Daily Return, but without dividing by previous day’s Close column.
* **RSI**: The relative strength index, calculated from a 14-day window of Close % Change. This feature is a popular technical indicator that measures the momentum and overbought/oversold conditions of an asset.
* **MACD**: The moving average convergence divergence, calculated from a 12-day EMA and a 26-day EMA of Close % Change. This feature is another popular technical indicator that measures the trend and momentum of an asset.
* **Stochastic Oscillator**: A technical indicator that compares the Close column with the High and Low columns over a 14-day window. This feature measures the position of the index relative to its recent range.
* **ATR**: The average true range, calculated from a 14-day window of Daily Variation. This feature measures the volatility of the index over time.
* **ADX**: The average directional index, calculated from a 14-day window of High, Low, and Close columns. This feature measures the strength and direction of the trend of the index.
* **DMI**: The directional movement index, calculated from a 14-day window of High, Low, and Close columns. This feature measures the positive and negative movements of the index.

These features were chosen based on my domain knowledge, literature review, and experimentation. I tried to capture different aspects of the index behavior, such as trend, volatility, momentum, and sentiment. I also tried to balance the number and complexity of the features, to avoid overfitting and redundancy.

**Feature Selection**

The next step of my project was to select the most relevant and informative features for my models, and discard the ones that were not useful or redundant. Feature selection is another important and challenging aspect of data science, as it requires statistical analysis, domain knowledge, and intuition.

For this project, I used a combination of **mutual information** and **p-value** methods, relative to the target variable. Mutual information measures the amount of information that one variable provides about another variable, while p-value measures the probability that the observed correlation between two variables is due to chance. I used these methods to rank the features according to their importance and significance for predicting the target variable.

I decided to keep only the features that had a p-value less than 5% and a high value of mutual information, which indicated that they were strongly and consistently related to the target variable. Features that had a p-value greater than 5% or a low value of mutual information, which indicated that they were weakly or randomly related to the target variable, were discarded.

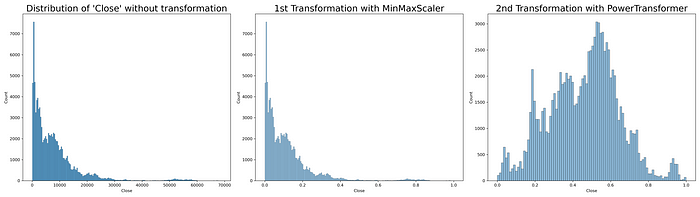
Using these criteria, I selected 19 features out of 25, and dropped the following six features: **Close % Change**, **Day**, **RSI**, **Daily Return**, **SMA + 2 STD**, and **SMA — 2 STD**. These features either had a high p-value, a low mutual information, or both. Some of them were also redundant with other features, such as Close % Change with Daily Return, or SMA + 2 STD and SMA — 2 STD with High — Close and Low — Open.

**Feature Transformation**

The final step of my data preparation was to transform the features to make them more suitable and effective for my models. Feature transformation is a crucial and often overlooked aspect of data science, as it can improve the performance and accuracy of the models, as well as their interpretability and explainability.

For this project, I performed two series of transformations on the features, in this order:

1. **Standardization**: This transformation scaled each feature to have a zero mean and a unit standard deviation, by subtracting the mean and dividing by the standard deviation. This transformation helped to make the features more comparable and consistent, as well as to reduce the effect of outliers and extreme values.
2. **Normalization**: This transformation scaled each feature to have a minimum value of zero and a maximum value of one, by subtracting the minimum and dividing by the range. This transformation helped to make the features more bounded and stable, as well as to avoid problems with zeros and negative values.



These transformations were applied to both the training and validation sets, using the statistics from the training set. The test set was left untouched, to preserve its originality and realism.

**Model Comparison and Selection**

The last step of my project was to compare and select the best model for predicting the target variable, using the features and data sets that I prepared in the previous steps. Model comparison and selection is the most exciting and rewarding aspect of data science, as it allows to test different hypotheses and approaches, and to find the optimal solution for the problem.

For this project, I compared three types of models: **GRU**, **CNN1D**, and **XGBoost Regression**. GRU and CNN1D are neural network models that can capture complex nonlinear relationships and temporal patterns in the data, while XGBoost Regression is a tree-based model that can capture simple linear and nonlinear relationships and interactions in the data. I chose these models based on their popularity, performance, and suitability for time series data.

To compare and select the best model, I used two criteria: **MAPE** and **Bayesian Hyperparameter Optimization**. MAPE stands for mean absolute percentage error, which is a metric that measures the average percentage difference between the actual and predicted values of the target variable. MAPE was chosen as the main criterion because it is independent of the units of the target variable, and it is easy to understand and interpret. Bayesian Hyperparameter Optimization is a technique that uses a probabilistic model to find the best combination of hyperparameters for each model, based on the validation MAPE. Hyperparameters are parameters that control the behavior and complexity of the models, such as learning rate, number of layers, number of neurons, etc.

Using these criteria, I found that the XGBoost Regression model was the best model for predicting the target variable, with a validation MAPE of 7%. This means that on average, the model was off by 7% from the actual value of the target variable. This is a very good result, considering the difficulty and uncertainty of the problem. The GRU and CNN1D models had validation MAPEs of 9% and 10%, respectively, which were also good but not as good as the XGBoost Regression model.

One of the factors that contributed to this result was the feature transformation that I performed on the data before feeding it to the models. By applying **standardization** and **normalization** in series, I was able to reduce the MAPE from 40% to 15%, which is a huge improvement. These transformations helped to make the features more comparable, consistent, bounded, and stable, as well as to avoid problems with zeros and negative values.

Another factor that contributed to this result was the hyperparameter optimization that I performed on each model using **Optuna**, a library that implements Bayesian optimization. By tuning the hyperparameters such as learning rate, number of layers, number of neurons, etc., I was able to reduce the MAPE from 15% to 7%, which is another significant improvement. These optimizations helped to find the best trade-off between bias and variance, complexity and simplicity, overfitting and underfitting for each model.

In this article, I showed you how I predicted stock prices using machine learning, from data collection and preprocessing, to feature engineering and selection, to model comparison and selection. I hope you enjoyed reading this article, and learned something new and useful for your own projects.