451 Feature Engineering: Programming Assignment 1

Prepared by Tom Miller

June 18, 2025

Financial Feature Engineering

We use machine learing classifiers, including tree-based ensemble boosting methods, to predict the direction of oil futures (up or down) using a number of lagged price features. In particular, we look at daily closing spot prices for West Texas Intermediate (ticker WTI) with lags of one to three days, as well as features based on opening, closing, high, and low prices, and daily trading volume.

Feature selection is a balancing act. We prefer parsimonious models—models with few parameters or features. We prefer models that fit the data well—models with high goodness of fit.

Consider three categories of feature selection methods: filtering, embedding, and wrapper methods. As the following discussion shows, we adopted a comprehensive wrapper approach in favor of filter and embedded methods. Our approach is well-supported by leading thinkers in statistics [7] [9].

Filtering methods assess the relevance of each explanatory variable or feature individually, working with one feature at a time and noting that feature's relationship with the target or response variable being predicted. Analysts sometimes employ filtering with a selection criterion relating to the strength of the relationsihp between the explanatory feature and the response or target. Testing for statistical significance is often employed in filtering. Even when correcting for inflated Type I errors (a la Bonferonni) we reject this method because evaluating one feature at at a time ignores the potential explanatory value of combinations of features.

Embedded methods incorporate feature selection into the model's training process, which for the method employed in this project would be feature selection within a time series cross-validation design and using a preferred boosting algorithm, XGBoost. We rejected this method in favor of all possible subsets. We have other considerations with regard to XGBoost. In particular, we want to thoroughly test hyperparameter settings.

Wrapper methods evaluate subsets of features based on model performance. Forward and backward stepwise selection fall within this class of feature selection methods. Forward methods start with the feature of highest predictive value, adding one feature at a time based—the feature that adds the most to the model's predictive power. Backward stepwise selection starts will all features and drops one feature at a time—the feature that detracts the most from the model's predictive power. We view stepwise procedures as less trustworthy than evaluating all possible subsets.

We employed the ultimate wrapper method in this project, considering all possible subsets of 15 features. We selected the "best" subset by referring to the Akaike information criterion (AIC) [1]. Evaluations of this type are common in traditional time series modeling.

Let k be the number of parameters in a candidate model and $\hat{L}$ be the maximum likelihood of that model. Then

$$AIC = 2k - 2ln(\hat{L})$$

We select the model with the smallest AIC. The formula clearly reflects the model selection balancing act. Higher k is avoided because we prefer models with fewer parameters. Higher $\hat{L}$ is preferred because we want models that fit the data well (have high goodness of fit).

Across all possible subset models and using traditional logistic regression, we select the "best" subset of features for predicting the binary target on the entire training set.

Subset selecion with AIC does not require splitting the data into training and test.

As with many time series data sets, we expect a high lag-one autocorrelation and possible positive autocorrelations at lag-two and lag-three. We did not expect a need to go beyond lag-three, but we did utilize exponential moving averages across as many as eight time periods.

Knowing that technical analysts (including chartists with their infamous candlestick plots) often draw on patterns detected in high versus low prices and opening versus closing prices, we defined HML (high minus low prices) and OMC (opening minus closing prices) as daily features, computing lag-one, lag-two, and lag-three values. We also considered trading volume up to three lags.

We drew daily trading data on West Texas Intermediate (ticker WTI) from Yahoo! Finance. We defined an intial set of fifteen features (DataFrame columns or series) using Python Polars. Here are the index numbers, feature names, and feature descriptions to begin the subset selection process:

(0) CloseLag1 Lag-one daily closing price

(1) CloseLag2 Lag-two daily closing price

(2) CloseLag3 Lag-three daily closing price

(3) HMLLag1 Lag-one high minus low daily prices

(4) HMLLag2 Lag-two high minus low daily prices

(5) HMLLag3 Lag-three high minus low daily prices

(6) OMCLag1 Lag-one open minus closing daily prices

(7) OMCLag2 Lag-two open minus closing daily prices

(8) OMCLag3 Lag-three open minus closing daily prices

(9) VolumeLag1 Lag-one daily volume

(10) VolumeLag2 Lag-two daily volume

(11) VolumeLag3 Lag-three daily volume

(12) CloseEMA2 Exponential moving average across two days

(13) CloseEMA4 Exponential moving average across four days

(14) CloseEMA8 Exponential moving average across eight days

In preparing data for modeling (both in this initial work with logistic regression and in subsequent gradient boosting), we employed standard scaling across the full set of features. Reviewing the definition of the target variable, we observed a nearly balanced set of binary responses: 2442 (47.8 percent) 1s for upward movement and 2668 (52.2 percent) 0s for even or downward movement. We saw no need for oversampling to acheive additional balance. Also, our final evaluation of classification performance will draw on the area under the Receiver Operating Characteristic Curve (AUC), which provides an effective metric for classification performance with unbalanced as well as balanced data.

(c) Provide the final list of selected features. We searched across all possible subsets, more than 30 thousand, identifying 10 subsets with the smallest *AIC* values. Then we looked for features that appeard most often in these best subsets. We decided to move ahead with these five features:

(2) CloseLag3 Lag-three daily closing price

(3) HMLLag1 Lag-one high minus low daily prices

(7) OMCLag2 Lag-two open minus closing daily prices

(8) OMCLag3 Lag-three open minus closing daily prices

(14) CloseEMA8 Exponential moving average across eight days

Our reading of financial time series literature ([8] [15] [18] [17] [19]) pointed to the importance of autoregressive-lag-one models. So, we were surprised that **CloseLag1** was not included in the set of five. Reviewing the correlation heat map, however, we saw that much of the autoregressive-lag-one information had been captured by **CloseLag3** and **CloseEMA8**. Curiously, each of the five selected features has a near-zero correlation with **LogReturn**, suggesting that we may face challenges in developing an accurate classification model for the **Target**.

Financial Machine Learning

Model Building Overview. Let's build a model to predict positive market moves (uptrend) using the feature subset derived above. As we are asked to employ gradient boosting, XGBoost (eXtreme Gradient Boosting) seemed a good choice. It is popular and well-documented. Our initial model building efforts employed XGBoost with all hyperparameters set to default values.

We are not using AutoML or AutoML-like methods to select a "best" modeling algorithm among a set of algorithms. Our feeling is that XGBoost has proven to be an effective boosting method for a wide range of data analysis problems, including time series and classification problems like we have here. Were we to use an AutoML-like technology, it would likely be the method proposed by Feurer et al. [5], which integrates with the Scikit-Learn environment.

We used a five-fold time series cross-validation design for model building, a design that has been fully implemented within Scikit-Learn.

We asked for one thousand XGBoost classifiers to provide a thorough test. The results of this initial cross-validation exercise were not especially encouraging: **The average percentage of correct classifications (up or down predictions in test sets) was only 50.3.** Can we set hyperparameters to obtain more accurate classification with XGBoost?

Hyperparameter Tuning. Tune the hyperparameters of the estimator to obtain an optimal model. Key hyperparameters within XGBoost include **max_depth**, **min_child_weight**, **subsample**, **learning_rate**, and **n_estimators** [3]. Rather than focusing on a predefined set of hyperparameter values in a grid search, we explored a wide range of possible values with randomized search. Advantages of randomized search have been noted by Geron [6].

The first two hyperparameters, **max_depth** and **min_child_weight**, relate to the complexity of models, with more depth and lower child weights associated with more complex models. Calling for more randomness with higher **subsample** values and lower **learning_rate** values protects against overfitting.

The results of randomized search implied that we may indeed be able to improve on classification performance through judicious hyperparameter settings. Here are the suggested "best" hyperparameter settings:

max_depth: 9

min_child_weight: 9

subsample: 0.50

learning_rate: 0.09

n_estimators: 273

Model Evaluation. Evaluate the model's prediction quality using the area under the receiver operating characteristic (ROC) curve, a confusion matrix, and classification report. Using the suggested "best" hyperparameter settings, we fit an XGBoost model to the full data set using the subset of five features.

Having arived at a fully specified model, with the feature subset selected and hyperparameters set, it is common to evaluate model performance on the full training set. This we do, obtaining an ROC curve with area under the curve (AUC) of 0.82.

The confusion matrix has 2668 actual negative return days and 2442 actual positive return days. 2249 of the negative return days were correctly predicted, and 419 were incorrectly predicted (false positives). 1929 of the positive return days were correctly predicted, and 513 were incorrectly predicted (false negatives). So, 4178 of the 5110 days were correctly predicted, yielding an overall classification accuracy of 81.8 percent on the training data. Here is the confusion matrix:

	predicted negative return	predicted positive return	total totals
actual negative return	2249	419	2668
actual positive return	513	1929	2442
totals	2762	2348	5110

The first two lines of the classification report are reproduced here:

	precision	recall	f1-score	support
negative return	0.81	0.84	0.83	2668
positive return	0.82	0.79	0.81	2442

Conclusions and Next Steps

Predictions on the training data are, of course, much better than we would expect with new data. To see how well a model performs in practice, we put it into practice. One way to anticipate how it will perform is to evaluate it on a hold-out test set.

To assist in automated, algorithmic trading we prefer a multi-category prediction. Instead of predicting up or down returns, we could use a three-level predictive model with associated trading action or inaction:

Up Take a long position, buy call options

Stable Hold current position, no trading

Down Take a short position, buy put options

A model for predicting the direction of daily returns sets the stage for testing the predictive utility of additional features. The domain of potential features or leading indicators is wide, including those associated with other price series, economic indicators, international events, securities filings, analyst and news reports, and media measures. Baumohl [2] and Masters [12] point to the potential of adding features not defined directly from an asset's price time series.

Developing machine learning models for predicting future prices or returns is a challenging endeavor, as noted by numerous exceptional books in the area [13] [14]. To be successful in quantitative finance, a practioner needs to have more than a general understanding of machine learning. Practical experience with financial time series is essential.

References

[1] Hirotugu Akaike. "Information theory and an extension of the maximum likelihood principle". In: *2nd International Symposium on Information Theory*. Ed. by B.N. Petrov and F. Csaki. Budapest: Akademia Kiado, 1973, pp. 267–281.

[2] Bernard Baumohl. *The Secrets of Economic Indicators: Hidden Cues to Future Economic Trends and Investment Opportunities*. (third edition). Upper Saddle River, NJ: Pearson Education, 2012.

[3] Partha Pritam Deka and Joyce Weiner. *XGBoost for Regression Predictive Modeling and Time Series Analysis: Learn How to Build, Evaluate, and Deploy Predictive Models with Expert Guidance*. Birmingham, UK: Packt, 2024.

[4] Pablo Duboue. *The Art of Feature Engineering: Essentials for Machine Learning*. Cambridge, UK: Cambridge University Press, 2020.

[5] Matthias Feurer et al. "Efficient and Robust Automated Machine Learning". In: *Advances in Neural Information Processing Systems 28*. 2015, pp. 2962–2970.

[6] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. (third edition). Sebastopol, CA: O'Reilly, 2023.

[7] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. (second edition). New York: Springer, 2009.

[8] Rob J. Hyndman and George Athanasopoulos. *Forecasting Principles and Practice*. (third edition). OTexts: Online Open-Access Textbooks, 2021.

[9] Alan Julian Izenman. *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. Springer, 2008.

[10] Jeroen Janssens and Thijs Nieuwdorp. *Python Polars: The Definitive Guide*. Sebastopol, CA: O'Reilly, 2025.

[11] Max Kuhn and Kjell Johnson. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. Boca Raton, FL: CRC Press, 2020.

[12] Timothy Masters. *Statistically Sound Indicators for Financial Market Prediction*. (second edition). Self-published, 2020.

[13] Jiri Pik et al. *Hands-On AI Trading with Python, QuantConnect, and AWS*. New York: Wiley, 2025.

[14] Marcos López de Prado. *Advances in Financial Machine Learning*. New York: Wiley, 2018.

[15] Sheldon M. Ross. *An Elementary Introduction to Mathematical Finance*. (third edition). Cambridge, UK: Cambridge University Press, 2011.

[16] Robert E. Schapire and Yoav Freund. *Boosting Foundations and Algorithms*. Cambridge, MA: MIT Press, 2012.

[17] Stephen J. Taylor. *Asset Price Dynamics, Volatility, and Prediction*. Princeton, JJ: Princeton University Press, 2005.

[18] Stephen J. Taylor. *Modeling Financial Time Series*. New York: Wiley, 1986.

[19] Ruey S. Tsay. *An Introduction to Analysis of Financial Data with R*. New York: Wiley, 2013.

[20] Paul Wilmott. *Machine Learning: An Applied Mathematics Introduction*. Panda Ohana Publishing, 2019.

[21] Alice Zheng and Amanda Casari. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. Sebastopol, CA: O'Reilly, 2018.