Using Machine Learning Methods to Predict Price Movements of Index-Tracking ETFs

Zechang (Charlie) Yang
Professor Jonathan Hanke
SML 312 Fall 2024

December 6, 2024

This paper represents my own work in accordance with University regulations. /s/ Zechang Yang 12/6/2024

1 Overview

Predicting the price of financial assets is a challenging task that is of great interest to investors, traders, and researchers for reasons such as risk management, portfolio optimization, and profit maximization. Market participants are constantly seeking new ways to predict the price of financial assets, in which randomness and uncertainty play a significant role especially when it comes to the short-term (intra-day) price or price movement. In this effort, machine learning methods have become increasingly popular in this regard. At the same time, it is well-known that the price of financial assets is influenced by (or at least hypothesized to being influenced by) a variety of factors, including macroeconomic data, technical indicators, and market sentiment.

In this project, I am interested in predicting the price of index-tracking ETFs (primarily the Invesco QQQ Trust that tracks the NASDAQ index) using macroeconomic data, technical indicators, volatility indices etc. of previous 30-trading-day period (for simplicity, the term "trading-day" will be referred to as "day"). The main research questions are as follows:

1) What factor(s) play the most prominent role in accurately predicting the ETF price or price movement for the next day?

2) What is the model/method most suited for this purpose?

3) Can we develop a (intra-day) trading strategy based on the optimal model/method and outperform the market?

To answer these questions, two more questions will also be asked:

1) What is the most suitable output variable for the model?

2) What is the most suitable metric to evaluate the model?

The main types of analysis would be feature engineering and selection, model evaluation using cross validation, and backtesting.

2 Related Work

Here I will provide an overview to the three major related work that inspired me to pursue this project as well as guided me in the beginning process of this project. The rest with be referenced throughout the paper as I build and analyze the models.

[1]: this is the article that originally sparked my interest in this topic. The author uses an array of different methods/models including Generative Adversarial Networks (GAN) with LSTM, CNN, Bayesian optimization, and Reinforcement learning (RL) to process the data and train the model (to be honest, I only have limited understanding to his article). In the end, he was able to accurately predict the price of the Goldman Sachs stock despite the fact that the price of a single stock is more volatile than an index fund. I plan to use a simplified model with fewer possible factors to predict the price of an ETF, which hopefully is easier to predict due to the lowered volatility and subjectivity to financial events (e.g., earnings calls).

[2]: the authors build a model that identifies the investment signals and design an algorithmic trading system, which goes beyond the scope of my focus but does include many financial indices that I may also be able to use as predictive factors. For example, I will include the Cboe volatility indicies (will be referred to as "volatility indicies") as part of my model.

[3]: the authors use past returns, past volume, dummies for days/months, and a combination of all three to predict the direction (not the price/price change) of ETFs and evaluate the performance of three models: (1) deep neural networks, (2) random forests, and (3) support vector machines. It inspires me to use some other ETFs that are more indicative of the overall market condition as features in my model, which I will also include in my model, such as TLT (iShares 20 + Years U.S. Treasury bonds) and GLD (SPDR Gold Shares) to complement the limited macroeconomic data I have.

3 Data

I used the following data from [6]:

1) (*Market Sentiment*) The open, high, low, and close of the following volatility indices:
   (a) VIX (CBOE S&P500 Volatility Index),
   (b) VIN (CBOE NASDAQ Volatility Index), and
   (c) VID (CBOE DJIA Volatility Index).
2) The open, high, low, close prices and volumes of the following ETFs:
   (a) QQQ (Invesco QQQ Trust),
   (b) SPY (SPDR S&P 500 ETF Trust),
   (c) OIH (VanEck Oil Services ETF),
   (d) IYR (iShares US Real Estate ETF),
   (e) LQD (iShares iBoxx $ Investment Grade Corporate Bond ETF),
   (f) TLT (iShares 20+ Year Treasury Bond ETF),
   (g) TIP (iShares TIPS Bond ETF), and
   (h) GLD (SPDR Gold Trust).
3) (*Macroeconomic*) The daily exchange rates of USD to and from the following major currencies:
   (a) EUR (Euro),
   (b) JPY (Japanese Yen),
   (c) GBP (British Pound),
   (d) CHF (Swiss Franc), and
   (e) CNY (Chinese Yuan).

4) US Treasury and Inflation Indices: 30 Year Bond (B30), 20 Year Bond (B20), 10 Year Bond (B10), 7 Year Bond (B7), 5 Year Bond (B5), 2 Year Bond (B2), 1 Year Bond (B1), 90 Day Bill (T90), 30 Day Bill (T30), and Inflation (CPI).

I cleaned the raw data by removing rows with invalid numbers and retaining only the dates that were common across all datasets (see Figures 2 and 3 for visualizations). Then, for each day (day 0), I made the y value the output variable I chose (more on this in the next section) and the X value the set of values of the previous 30-days of the features chosen (days -1 through -30) (see Figure 4). Lastly, I splitted the data in three: 80% training data, 10% cross validation, and 10% test. Figure 1 shows the price of QQQ over time, which is closely related to any output variables we may choose.

1) Training set from 2005-01-03 to 2020-02-24 (size 3799)

2) Cross-validation set from 2020-02-24 to 2022-01-11 (size 475)

3) Test set from 2022-01-11 to 2023-12-01 (size 475).

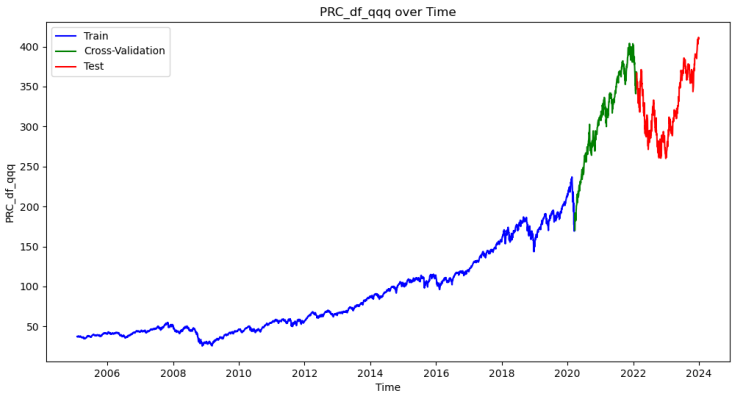

Figure 1. Price of QQQ over Time, Split into Training, Cross Validation, and Testing Sets

4 Modeling & Analysis

1. Choosing the Output Variable

After obtaining a clean dataset, I proceeded to build a machine learning model. Firstly, I needed to decide what is the output variable that best suits the purpose: it is equally or even more important for me to build a trading strategy based on the model hence it makes to select from the following three output variables:

1) the (close) price of the ETF of the next day

2) the price movement (1 for up and 0 for down) of the ETF within the next day, as given by the formula:

$$\text{price movement} = \begin{cases} 1 & \text{if (close) price}_{\text{next day}} > \text{open price}_{\text{next day}} \\ 0 & \text{otherwise} \end{cases}$$

3) the percent change of the ETF within the next day, as given by the formula:

$$\text{percent change} = \frac{\text{(close) price}_{\text{next day}} - \text{open price}_{\text{next day}}}{\text{open price}_{\text{next day}}} \times 100$$

which all gives me a way to predict whether it goes up or down the next day that implicitly suggests a trading strategy (will be discussed in the results section).

For the sake of simplicity, I limited myself mostly to the first two output variables, which is standard in existing literature, and built models on a subset of the entire dataset (to save time) to decide which one to use. (Note: I did also run a linear regression model using the third output variable, and it gave a similar result as the second output variable after converting a positive

percent change to 1 and a negative to 0. It could be interesting to explore this further in the future as it would give an alternative perhaps better way to build the trading strategy, as will be discussed in the results section.)

For the first output variable, I tried linear regression and random forest. Although linear regression gives an MSE of 25.04, which is relatively small, it mostly uses the previous day's price to predict the next day's (as evident in the heavy weight on the previous day's price and Figure 5, a plot of predicted price against actual price). Furthermore, I checked its performance against a simple strategy of predicting the next day's price to be the same as today's price, which is something [1] didn't consider, and noticed it outperforms the linear regression model in terms of MSE (23.58). Indeed, the day-to-day change of an index-tracking ETF (in this case QQQ) is relatively small given it's a combination of many stocks hence it makes sense that this simple model would perform well. However, it doesn't give rise to a meaningful trading strategy as it also thinks the price would stay the same. When I tried linear regression with more features, it performs even worse with an MSE of 26.68. On the other hand, random forest (so is KNN) performs the worst: it practically predicts the price of the last day in the training set, which makes sense because QQQ is mostly going up in the last few years, so the future is more similar to the latest present. One interesting phenomenon to observe is that it did (kind of) predict the right trend during the bear market in 2022, which provides a confidence booster that the predictor variables are somewhat useful (for example, the volatility indices and the exchange rates with major foreign currencies are somewhat indicative of the overall market condition). It doesn't seem to be a good idea to use it as the output variable for the model.

For the second output variable, I tried logistic regression, random forest, XG boost, SVM, neural network, and QDA, and obtained an accuracy rate around 50% in all cases, which is roughly consistent with Dai and Zhang's findings ([5]). I will then proceed with this output variable.

Let's then examine the distribution of this output variable in the training, cross validation, and testing sets:

Dataset	Size	Class 0 (%)	Class 1 (%)
Training Set	3799	46.88	53.12
Cross Validation Set	475	43.79	56.21
Testing Set	475	45.89	54.11

Table 1. Distribution of Output Variable in Training, Cross Validation, and Testing Sets

We see that they are roughly balanced, which is better than in the case where we use the actual price where the overall market condition plays a more significant role. Even though the market is mostly going up in the long run, the price movement of the ETF is more or less random within the day (with the upward movement, class 1, consistently maintaining a slight majority).

2. Metrics to Evaluate the Model

I chose to use accuracy to evaluate the model, which is the most common metric used in the literature (with discretion, though: I need to take into account the specific confusion matrix as well since I don't want the model to be a trivial model that always predicts one output). However, it is arguably more important to use return as the metric to evaluate the model, as the purpose of the model is to build a trading strategy, but then in this case, the models are trained to maximize one thing but evaluated on another. It would be interesting for future to develop a model that is trained on the return directly for a given strategy, which can be further complicated by varying the strategies and select the best model with the best strategy suited to that model.

Additionally, the precision and recall scores also plays a prominent role

when we assume that we will adopt a trading strategy that does one thing when the ETF is predicted to go up and another thing when the ETF is predicted to go down:

1) Precision: the proportion of true positive predictions (the ETF is predicted to go up and does go up) among all positive predictions (the ETF is predicted to go up), which is important because I don't want to make a trade when the model predicts the price to go up, but it actually goes down. This is akin to a measure of risk-level of the strategy.

2) Recall: the proportion of true positive predictions (the ETF is predicted to go up and does go up) among all actual positive outcomes (the ETF goes up), which is important because I don't want to miss out on a trade when the model predicts the price to go down, but it actually goes up. This is akin to a measure of the profit-level of the strategy.

Whether precision or recall is more important is likely to be dependent on the specific market condition, which will be analyzed after backtesting the trading strategy.

One can also tailor trading strategies based on specific properties of the models, such as the ROC curve, precision rate, and recall rate, which, again for the sake of simplicity, is not considered in this project.

3. Feature Engineering

As suggested in the literature, the 7- and 21- day moving averages seem the most important ([1]), which I will use. Additionally, I calculated the percent changes for all ETFs.

4. Modeling and Backtesting Methodology

The following models are models I primarily trained and evaluated:

1) K Nearest Neighbors (knn)
2) Naive Bayes (nb)
3) Logistic Regression (log_reg)
4) Support Vector Machine (svm)
5) Decision Tree (dt)
6) Random Forest (rf)
7) Linear Discriminant Analysis (lda)
8) Quadratic Discriminant Analysis (qda), which is the model that [4] and [5] found to be the best performing model.

Firstly, I trained these models on the entire training set without feature selection.

Then, I selected features based on feature importance in the random forest model, PCA (which [1] uses), and logistic regression with Lasso regularization. I trained the models on the selected features and evaluated them on the cross validation set again. (I also trained neural network models, but it always converged to the trivial model that predicts 1 all the time hence I didn't train it with selected features.)

Lastly, I backtested on the cross validation and testing set with the best performing models (in terms of cross validation accuracy) using the following trading strategies.

Before I proceed, I will provide a visual representation of the modeling and backtesting methodology in Figure 6.

1) Buy the ETF at the open price and sell at the close price if the model predicts the price to go up, do nothing otherwise.

2) Buy the ETF at the open price and sell at the close price if the model predicts the price to go up, do nothing otherwise (this, combined with analysis of the ROC curve, is to provide a sanity check that the model does learn something about the labels as presented not as the reverse of it). It takes into account the awkward situation where the model somehow thinks 0 is the case where the ETF goes up, which does also provide a meaningful trading strategy.

3) Buy and hold the ETF for the entire period and sell it at the close price of the last day in the period (which is the simplest thing an investor could do). This is to provide a baseline for the trading strategies developed according to the models. However, we note that it is not a fair comparison because we are ignoring the out-of-trading-day movements of the ETF in the other strategies, which could be significant.

4) Buy the ETF at the open and sell at the close price, which provides the most comparable strategy to the first two strategies.

We note that there are many other possible trading strategies can be developed based on the models, such as shorting the ETF when the model predicts the price to go down/up instead of doing nothing. We can also buy an inverse ETF, a leveraged ETF, or call/put options, which introduces additional risks (and costs in the real world as inverse and leveraged ETFs have higher expense ratios and options have trading fees). This could be further explored in the future. Additionally, with developing trading strategies in mind and assuming we don't necessarily want an intra-day trading strategy which this project focuses on, one can also define price movement by using the open price of the next day instead of the close price of today, which adds yet another possible output variable to consider in the future.

5 Results & Discussion

1. Model Performance without Feature Selection

Model	Abbreviation	Accuracy
K Nearest Neighbors	knn	0.4737
Naive Bayes	nb	**0.5537**
Logistic Regression	log_reg	0.5179
Support Vector Machine	svm	0.4400
Decision Tree	dt	0.4779
Random Forest	rf	0.4526
Linear Discriminant Analysis	lda	0.4442
Quadratic Discriminant Analysis	qda	0.5221

Table 2. Model Performance without Feature Selection on the Cross Validation Set

Table 2 summarizes the accuracy of the models on the cross validation set without feature selection. We see that Naive Bayes performs the best, followed by Logistic Regression and Quadratic Discriminant Analysis. However, many model's performance is below 50%, which means they are effectively guessing (or worse than guessing) the price movement of the ETF if we only consider accuracy. Moreover, as observed in the case where we used the actual price as return, we see the overall market trend (which is going up in the long run) plays a role, so it is worthwhile to investigate the analogous concept in the price movement of the ETF: the distributions of the true labels. It would be disappointing if the model is just predicting the most common label. Recalling that the true distribution is 56.21% for class 1, we see that none of the models outperforms the trivial model that always perdicts 1 in terms of accuracy and that Naive Bayes comes pretty close to it, which despite the disappointing performance, is still consistent with the literature ([5]). This is *probably* not useful for trading when we want to outperform the market via the simplest trading strategies (we say "probably" here because

it is still possible for a model with a lower accuracy to outperform a trivial trading strategy because it is possible that the model more accurately identifies the bigger price movements in either direction thus harnessing bigger growth opportunities while avoiding bigger losses, which again brings up the potential loss of information when we limit ourselves to the signs the actual percent changes).

2. Model Performance with Feature Selection

We then proceed to select features based on several metrics: feature importance in the random forest model, PCA, and logistic regression with Lasso regularization. We notice in the random forest model that if a feature is important, it often "comes in a group," i.e., if df_tip_pct_change_-17 (the percent change of TIP 17 days ago) is important, then df_tip_pct_change_-xx (the percent change of TIP xx days ago) is also important. This makes sense because with the assumption that two assets are related, the percent change of one asset on each day for a certain period should be related to the percent change of the other asset. Thus, we shall consider the feature importance disregarding the number of days ago the value of the feature is from.

I then selected the top 19 group features, which corresponds with $19*30 = 570$ actual features (I was going to use 20 but noticed the 20th is the low price of another index which is already in the list, so I decided to drop it as it is highly correlated with that index and doesn't add too much new information). The following seemed important:

the percent change and volume of other ETFs and the volatility indices.

However, improvement is minimal or non-existent for models trained on selected features.

The results are similarly disappointing with PCA: many models become a trivial model, which makes sense because PCA dramatically reduces the dimensionality of the data and hence the model is less likely to learn anything useful. This also serves as a warning that we need more features to obtain meaningful results (this correlates with the result obtained from pruned decision trees, which is another failed attempt to improve the accuracy).

Lastly, I used logistic regression with LASSO to select features. Learning from the previous failed attempts, I selected more features: 301 features with 36 distinct groups of features. It is consistent with the findings from the random forest model that the percent change and volume of other ETFs and the volatility indices are important. In addition,

the exchange rates with GBP and CHF, the CPI, and the return on some treasury bonds/bills and the CPI

are also important, which suggests that LASSO picks out more macroeconomic related features. In this case, decision tree and quadratic discriminant analysis improve the most, with decision tree achieving the same performance as the trivial model:

Model	**Metric**	**Before Feature Selection**	**After**
dt	Accuracy	0.4779	**0.5621**
	Precision	0.5333	0.6130
	Recall	0.5693	0.5993
qda	Accuracy	0.5116	**0.5368**
	Precision	0.5665	0.5781
	Recall	0.5581	0.6517

Table 3. Model Performance for dt and qda with and without Feature Selection

I included the QDA model here even though it didn't outperform the trivial model because it has a remarkably high recall rate, which might be important in backtesting.

3. Backtesting & Analysis

I then backtested the trading strategies based on the naive bayes model trained on the entire training set, decision tree model trained on the selected features (with LASSO), and the QDA model trained on the selected features (with LASSO). The results are as follows:

Model	**Strategy 1**	**Strategy 2**	**Strategy 3**	**Strategy 4**
Naive Bayes	21.39%	-2.63%	110.11%	18.20 %
Decision Tree	**27.39%**	-7.22%	110.11%	18.20 %
QDA	11.83%	5.69%	110.11%	18.20 %

Table 4. Return of Strategies on Cross Validation Set for Different Models

Firstly, the returns on strategy 2 is always less than that of strategy 1, which suggests that the model did learn something about the labels as presented, which is validated by the area under the ROC curve (which is greater than 0.5). While the strategies didn't beat the buy and hold strategy (Strategy 3), two of them did beat the comparable strategy that trades within the day (Strategy 4).

Next, I backtested on the test set (for the best performing model but also for the other models as well since it is computational trivial to do so). The results are as follows:

Model	**Strategy 1**	**Strategy 2**	**Strategy 3**	**Strategy 4**
Naive Bayes	16.41%	9.63%	12.58%	27.62 %
Decision Tree	14.24%	11.72%	12.58%	27.62 %
QDA	**40.94%**	-9.45%	12.58%	27.62 %

Table 5. Return of Strategies on Test Set for Different Models

This time, the QDA model outperforms both of the trivial strategies by a great amount, which suggests a trading strategy based on the QDA model is promising. It seems that the QDA model is able to better capture the alpha

in the test set, even though it doesn't have an extrodinarily high accuracy rate for the test set. It does, however, have a high recall rate of 0.7276, which suggests that it might be more important that the model is able to capture the upward movements of the ETF than to avoid adverse conditions. At the same time, we lose some alpha's with the naive bayes and decision tree models in this case, probably for analogous reasons. Therefore, as alluded to before, the actual percent change might be important for trading strategy, which is something to consider in the future.

However, the best performing model for the cross validation set performs the worst for the test set while the worse performin model for the cross validation set performs the best for the test set, which validates the efficient market hypothesis to some extent. This calls my attention to investigate the difference in market conditions between the cross validation set and the test set. Indeed, a buy and hold strategy that performs well for the cross validation set does do as well for the test set, probably because the cross-validation set starts after the COVID dip and consists of only a strong bull market while the test set contains the majority of the bear market in 2022 and the bull starting late that year. Due to the short time frame for both sets, neither of them accurately captures the diverse market condition that the models were exposed to in the training set, with the test set arguably more representative of the training set. This suggests that it might be helpful to select models and built strategies based on market conditions, which is a topic for future research. At the same time, one should also be careful about making sure the training, cross validation, and testing sets are representative of the overall market condition.

Additionally, as [5] suggests, I don't have to limit myself to predicting the price movement of the next day: they've found that lengthening the time-horizon of the prediction could improve the performances with 44 days being the optimal. This is something to consider in the future.

6 Acknowledgements

I would like to thank Professor Jonathan Hanke and TA Chenyu Wang for their guidance and support throughout the semester. I would also like to thank the librarians Bobray Bordelon and Mary Carter for their help with getting the relevant financial data. Lastly, I would like to thank the rest of the SML 312 class for their support.

References

[1] Boris Banushev, "Using the latest advancements in AI to predict stock market movements," GitHub, 2018. https://github.com/borisbanushev/stockpredictionai/blob/master/readme.md

[2] S. Baek, K. Y. Lee, M. Uctum, and S. H. Oh, "Robo-Advisors: Machine Learning in Trend-Following ETF Investments," Sustainability, vol. 12, no. 16, p. 6399, 2020, doi: 10.3390/su12166399.

[3] J. K.-S. Liew and B. Mayster, "Forecasting ETFs with Machine Learning Algorithms," The Journal of Alternative Investments, vol. 20, no. 3, pp. 58–78, Winter 2018, doi: 10.3905/jai.2018.20.3.058.

[4] C. Hua, "Trading Strategies Based on Machine Learning Predictions of Long-Term Stock Price Movement in the Healthcare Industry," Apr. 2017. https://dataspace.princeton.edu/handle/88435/dsp01mp48sg39g

[5] Y. Dai and Y. Zhang, "Machine learning in stock price trend forecasting," 2013. https://cs229.stanford.edu/proj2013/DaiZhang-MachineLearningInStockPriceTrendForecasting.pdf

[6] Wharton Research Data Services. "WRDS" https://wrds.wharton.upenn.edu, accessed 2024-12-02.

7 Appendix

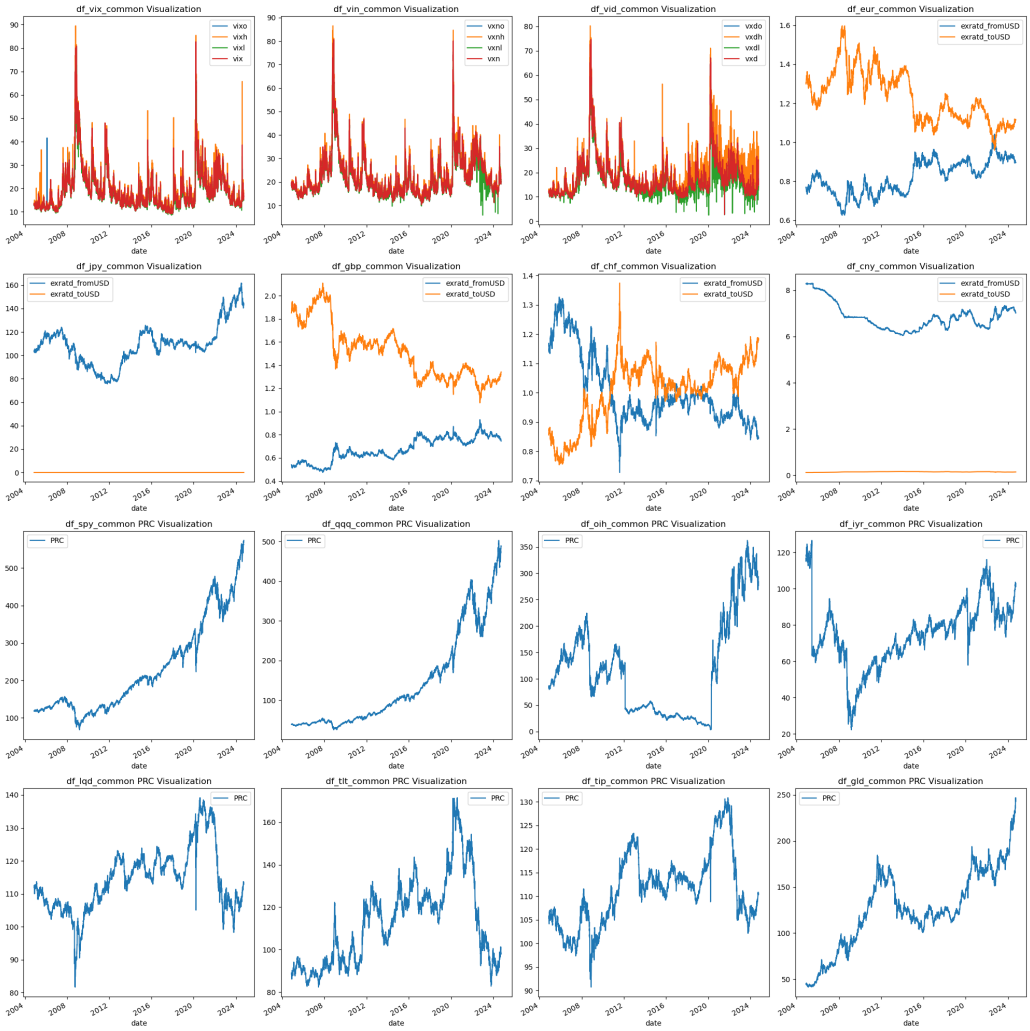

Figure 2. Visualization of the Data - Part 1 (Volatility Indices, Exchange Rates, and ETFs)

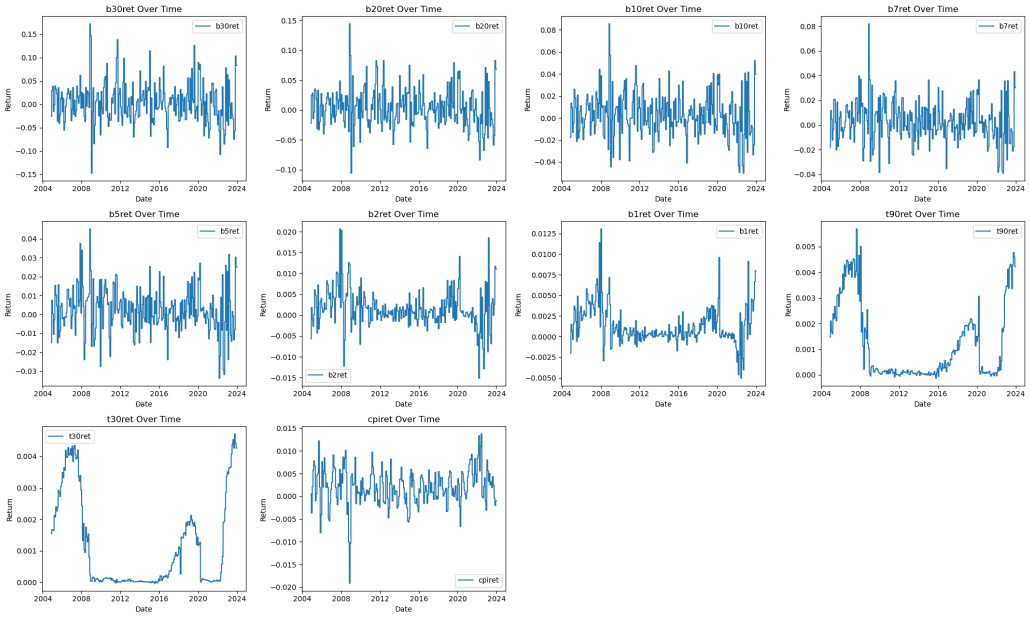

Figure 3. Visualization of the Data - Part 2 (Treasury and Inflation Indices)

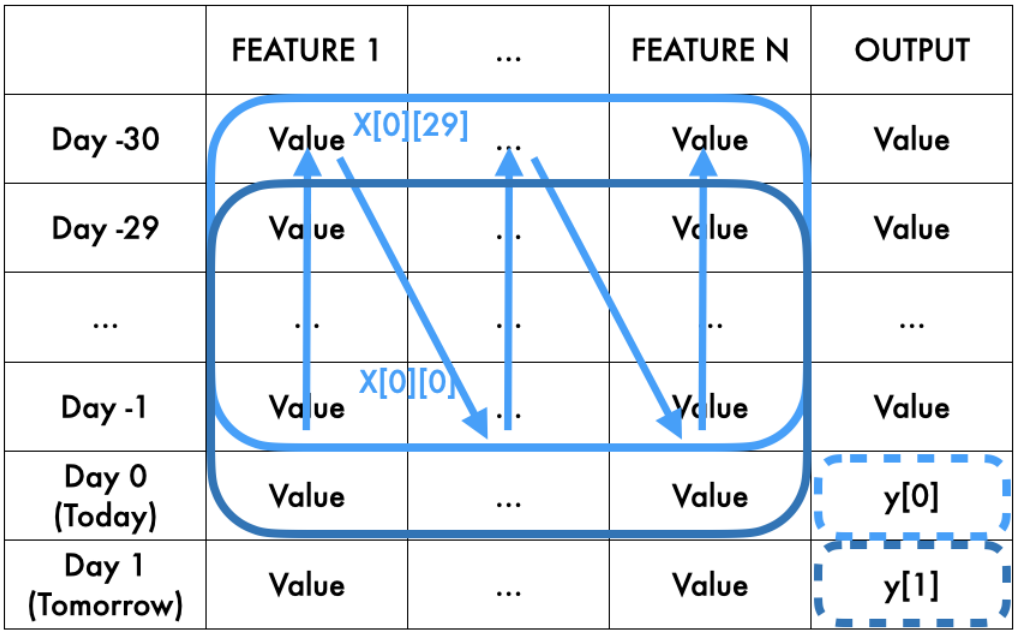

Figure 4. Making X and y

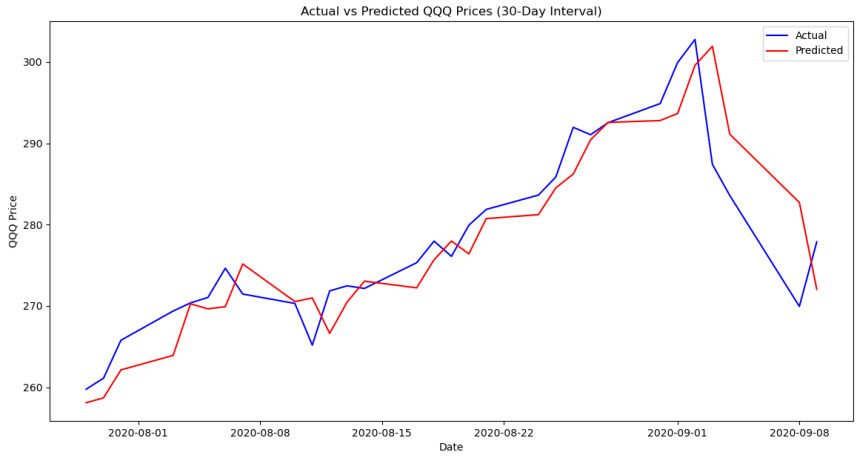

Figure 5. Actual vs. Predicted QQQ Prices for 30 Days for the Linear Regression Model

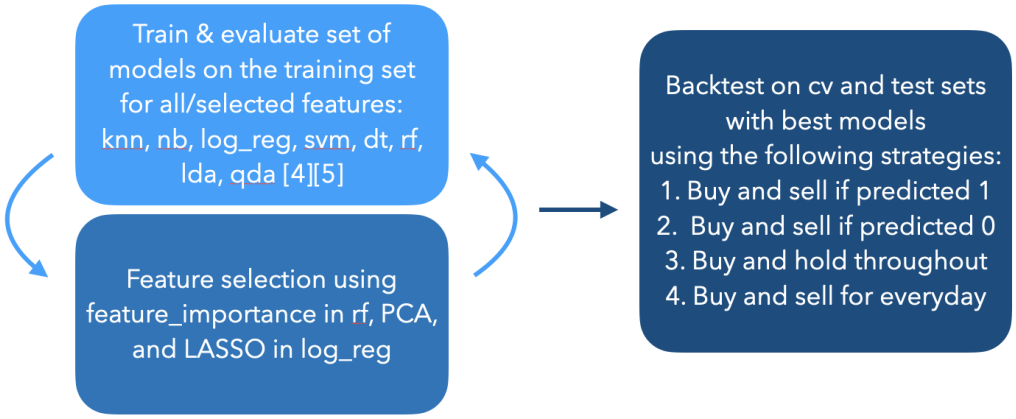

Figure 6. Modeling and Backtesting Methodology