EE 510: Machine Learning Theory & Algorithms **Spring 2022**

Mini Project 3

Due: June 3 (official) /June 10, 2022, 11:59PM PT

Student Name: Martín Rodriguez *Instructor Name: John Lipor*

Contents

1 Problem description

The goal for this final mini project is to take past financial data from the Japanese stock market and build a model which can accurately predict which trading decisions are likely to be profitable on incoming data [1]. A successful model will build a portfolio out of a subset of eligible stocks, ranked from highest to lowest expected returns for a given date. The top 200 ranked stocks are assumed to be purchased and the bottom 200 ranked stocks are assumed to be shorted. A spread return metric known as the Sharpe ratio is used to score the model's portfolio, and a model performs better if it more successful at producing consistent, more stable returns rather than big returns on a few days. The model was trained using the `CatBoost` algorithm which performs gradient boosting and uses decision trees as its weak learner [2]. The final model was then submitted through the competition's API and scored against a private set of data.

2 Exploratory data analysis (EDA)

2000 stocks are eligible for ranking in this competition. Because of the amount of data provided and the complexity of the API used to evaluate and submit the model's results, careful consideration of the data during the EDA phase is necessary. The `train_files` directory contains the main files to be used for training the model. The most important file for training is `stock_prices.csv`. It contains stock information for nearly all of the 2000 stocks beginning on 2017-01-04 and ending on 2021-12-03, amounting to 2,332,531 total data points. This training data includes the target variable, which for a given day t is the change in closing price for each stock, calculated from the closing price of day $t+1$ to day $t+2$. Some stocks were added in December 2020, however, so the data is not complete in the sense that some stocks are missing data from before this time. The `supplemental_files` folder also contains a `stock_prices.csv` file which contains the most recent stock data and is updated as the competition progresses.

Figure 1 shows the features with the most missing values. `ExpectedDividend` is missing over 99% of its values, so it is likely that this feature can be dropped when training.

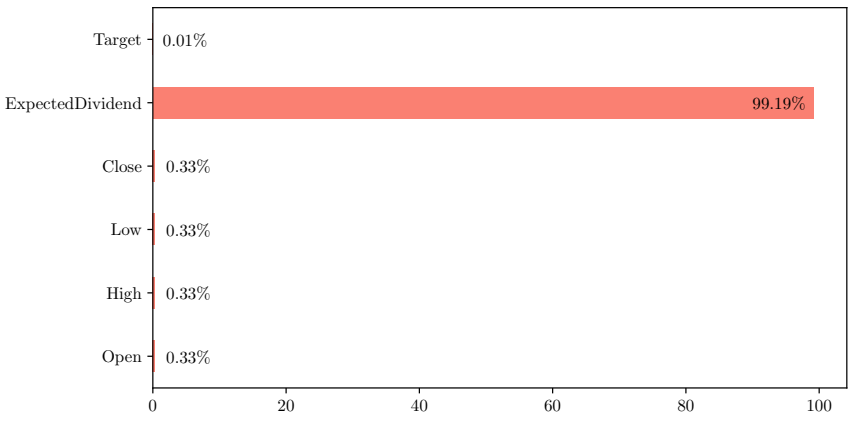

Figure 1: Features containing missing values

Figure 2 is an initial correlation matrix created from the raw data. As expected, `Close` is directly correlated with `Target` since the target variable is calculated from values of the closing price on successive days. What's interesting, though, is that all of the price-related features appear to be directly correlated with each other, which isn't as easy to explain.

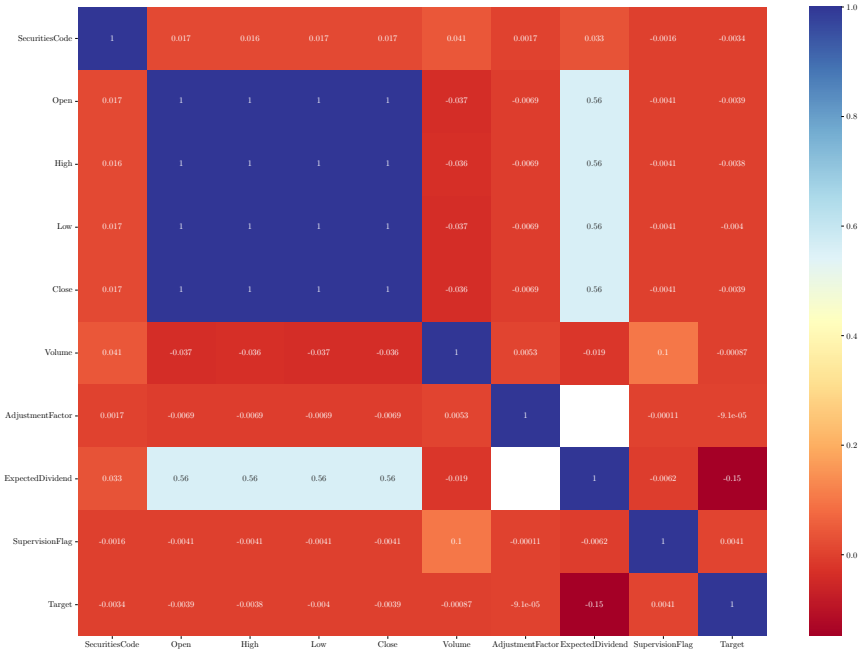

Figure 2: Feature correlation for raw data

2.1 Relationships between data features

2.1.1 Target variable vs. number of stocks

Again, looking at the raw data, Figures 3 and 4 show the distributions of the target variable mean and standard deviation over all the stocks. The mean is slightly positive indicating that, on average, the market had a net positive return. The kurtosis is fairly high, which means that the distribution deviates quite a bit from a Normal distribution in the length of its tails.

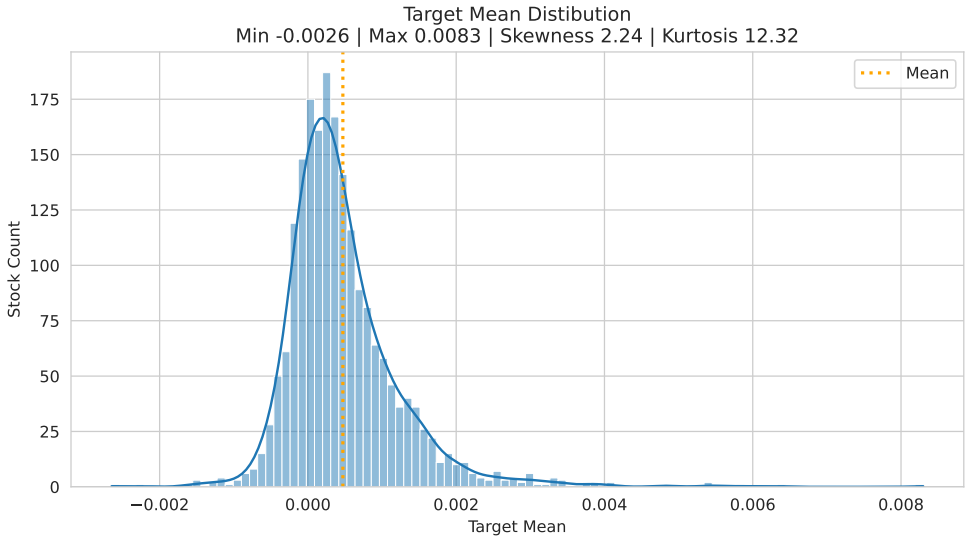

Figure 3: Distribution of target mean over all stocks

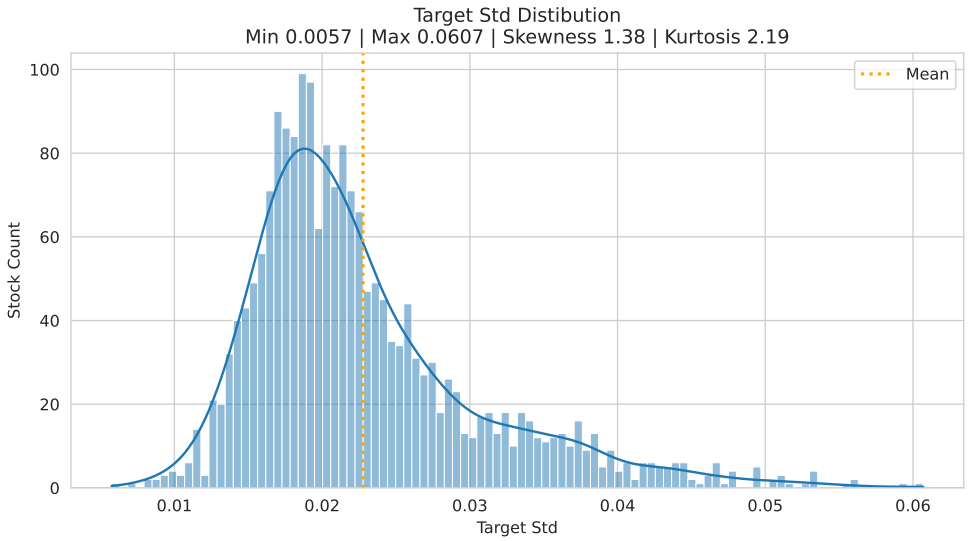

Figure 4: Distribution of target standard deviation over all stocks

2.1.2 Target variable vs. number of records per stock

Figures 5 and 6 show the overall joint plot of the target variable vs the number of records available for each plot. The majority of stocks have records for all 1202 dates, but we can see there are still a few that do not.

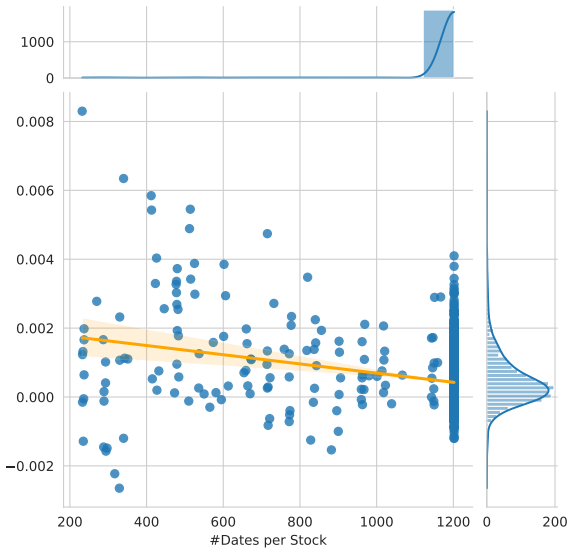

Figure 5: Joint plot of records per stock vs. target mean

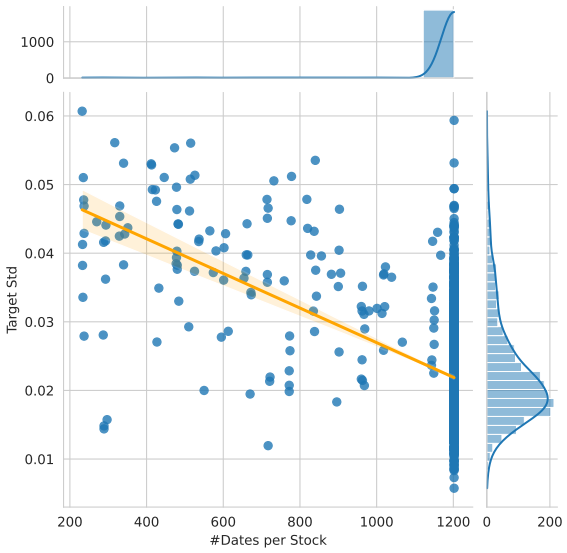

Figure 6: Joint plot of records per stock vs. target std

To get an idea of what the distribution looks like for stocks with fewer than the maximum 1202 records, the joint plots in Figures 7 and 8 show that these stocks have more dispersion in the target variable distribution.

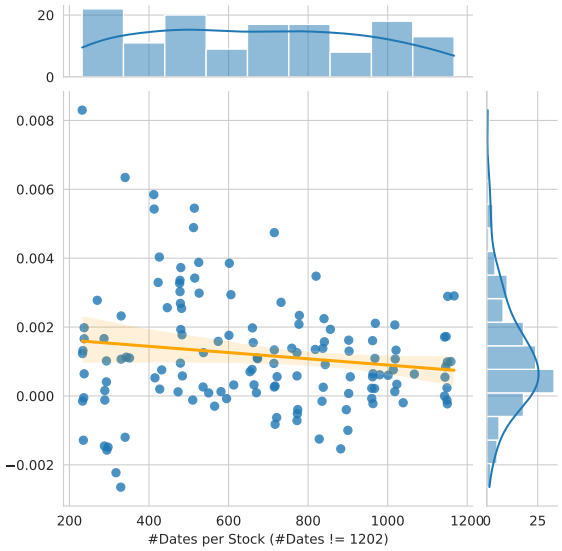

Figure 7: Joint plot of records per stock vs. target mean (less than 1202 records)

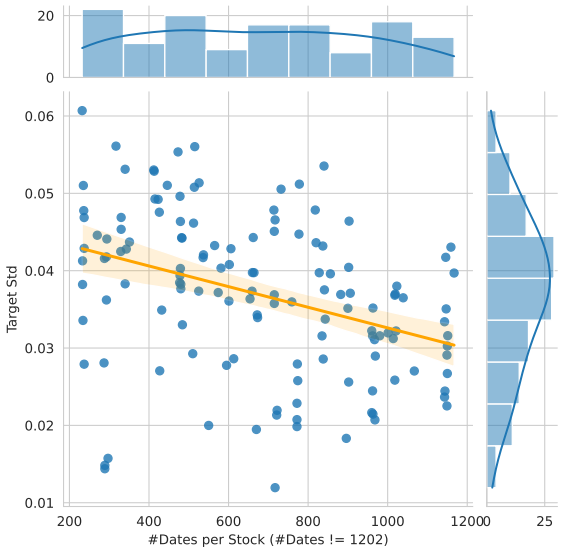

Figure 8: Joint plot of records per stock vs. target std (less than 1202 records)

2.1.3 Removing dates without all stock data available

Because 2020-12-24 is the first day when data is included for all 2000 stocks, dates before this were removed in order to train the model on the complete 231-day stretch of the 2000 stocks within the scope of the competition. Figures 9 and 10 show the target mean and standard deviation after data from before 2020-12-24 has been removed. We can see that the removal of this data lowered the skewness and kurtosis of the target mean distribution, making it appear more Gaussian. The distribution of the standard deviation also agrees with what we saw above, in that the removal of this data slightly increased the data's dispersion.

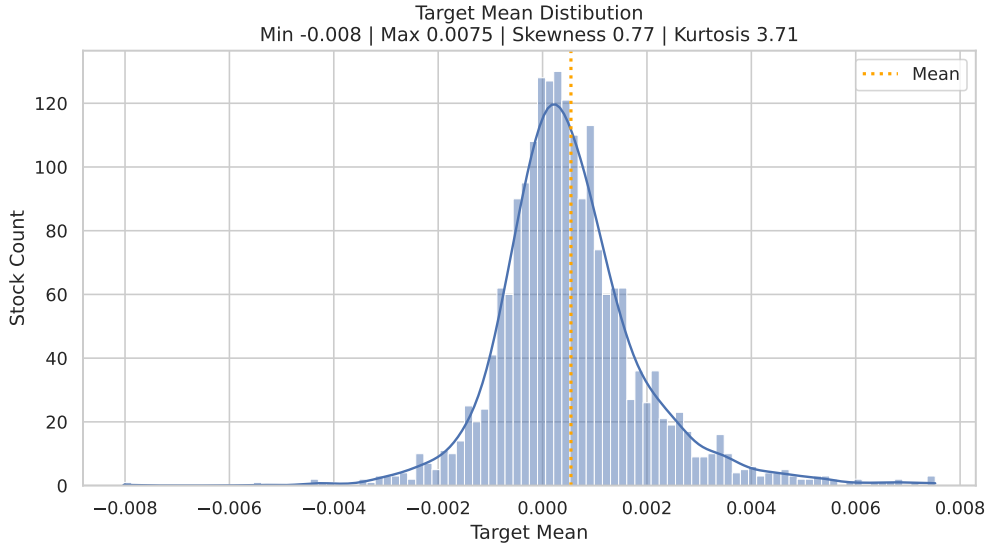

Figure 9: Distribution of target mean per stock (data after 2020-12-23)

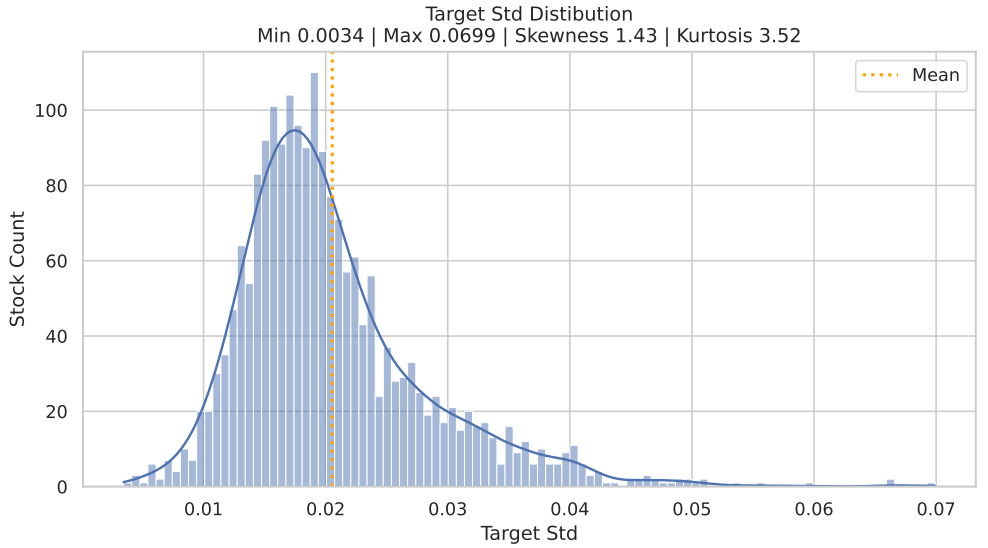

Figure 10: Distribution of target std per stock (data after 2020-12-23)

2.2 Looking at information included in stock list

The `stock_list.csv` file contains more information on the stocks included in the training file. Because information such as what market sector a stock belongs to could be helpful, the `SectorName` column was added to the training data. Figures 11 and 12 show the ratios of market sections/products and sector names of the 2000 target stocks.

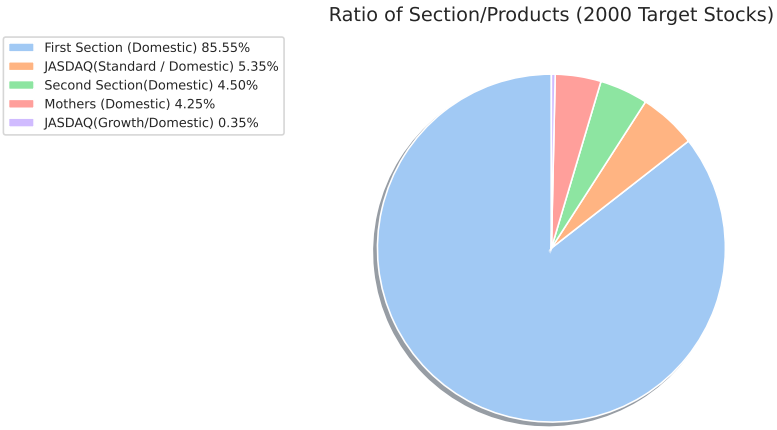

Figure 11: Ratio of Sections/Products in 2000 target stocks

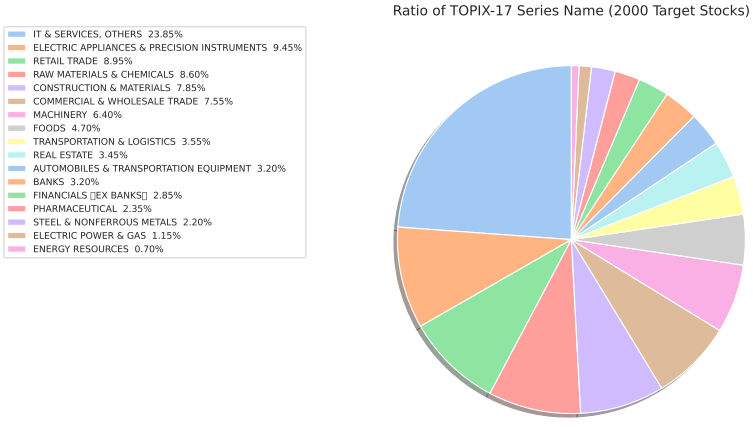

Figure 12: Ratio of 17 Sector Name in 2000 target stocks

Figure 13 is helpful for seeing the trends of each sector as well as the trends of the market as a whole for each year. Figure 14 lets us view which sectors have outliers in the target variable. Notably, the commercial & wholesale trade sector achieved almost 62% returns in at least one stock and at least one stock in the IT & services sector had a return of almost -40%.

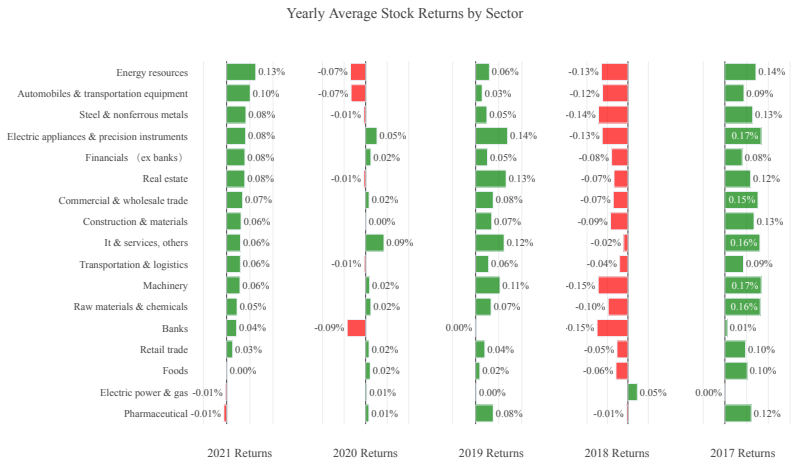

Figure 13: Yearly average returns by sector

Figure 14: Target distribution by sector

Figure 15 focuses on the best and worst individual stock returns over all the sectors. Figure 16 shows which stocks are most correlated with the target variable through their closing price. Riken keiki co., ltd. has the highest correlation between its closing price and the target variable at 0.12.

Figure 15: Highest and lowest returns by sector

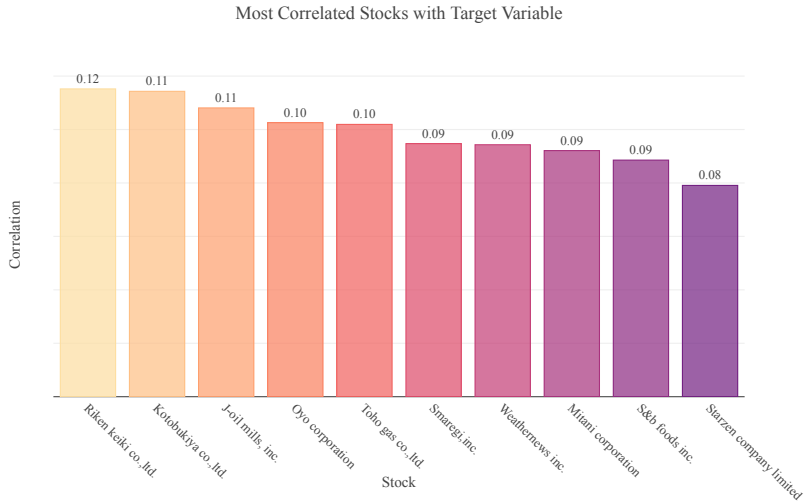

Figure 16: Stocks most correlated with target variable

Figure 17 shows the correlations of closing price between different sectors. Notably, the transporation and logistics sector is closely paired with the commercial and wholesale trade sector. The same goes for the retail trade and foods sectors. A few other sectors are also slightly less correlated, but are also worth looking at.

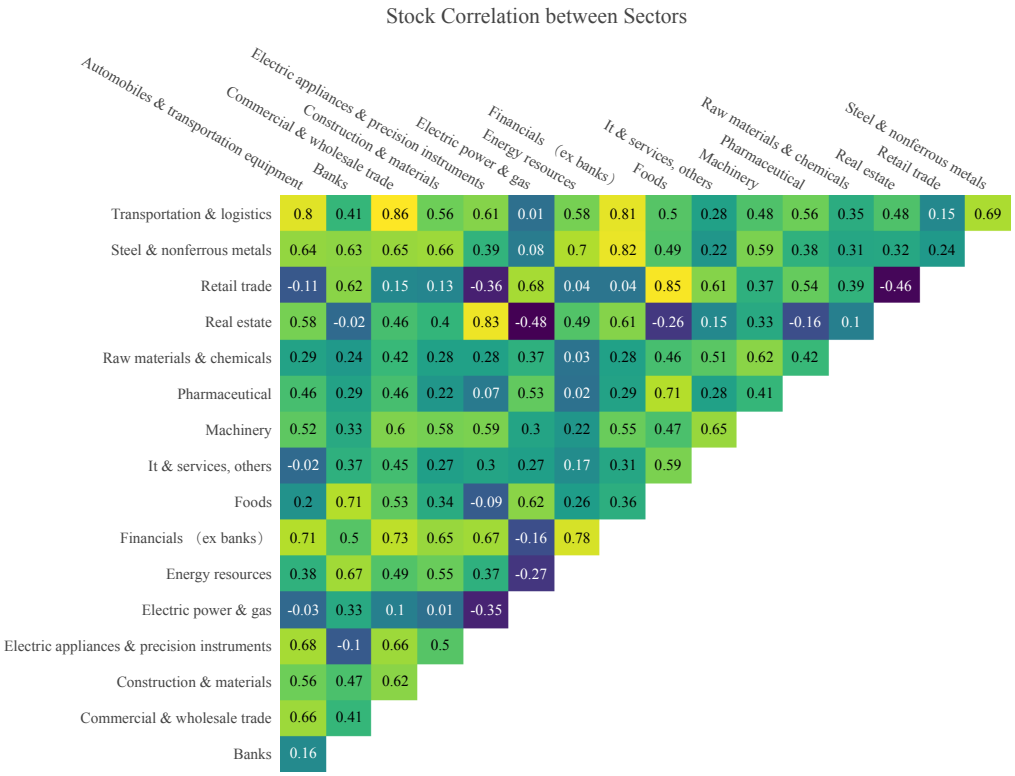

Figure 17: Stock correlation between sectors

3 Challenges

This project was quite challenging due to the sheer number of datapoints and supplementary files to digest, new techniques and definitions to learn, as well as the more complex metric and API needed to make a successful submission. The necessity of transforming the data from a time-series problem into a supervised learning problem also took some time to understand. Some of the implicit problems with the data were also not easy to discover initially. I had no idea that the competition hosts had written a notebook to aid in calculating the adjusted closing price or that it was even a problem until I stumbled upon the extremely informative notebook here: [3]. It was not clear at all how one should be able to find that notebook on their own. Another challenge was trying not to get lost in the potential mountain of EDA one could perform on the data. At times it felt like a book could be written on just this topic. It also was not entirely clear

what would be considered "cheating" in the competition. To play it safe, I only trained the model on the data included in the `train_data` folder for my initial submission. After some more reading, I wanted to see how my model would do after including the supplemental data, so I trained it on this data as well. I saw a slight increase in performance on the private data after using the supplemental data, but I am still not sure whether or not that was in the spirit of the competition.

4 Approach

The general approach taken in working with the data was as follows:

1. Exploratory data analysis (EDA)
   (a) Look for trends in the data
2. Preprocessing
   (a) Remove features with the most missing values
   (b) Fill other missing values
3. Feature engineering
   (a) Draw from prior knowledge about stock markets and financial metrics
4. Training
   (a) Create training and validation sets
   (b) Perform model selection
   (c) Train the final model
5. Model submission

4.1 Preprocessing

The feature `ExpectedDividend` was dropped completely because it was missing over 90% of its values. Other missing values were simply replaced with zeros. Many more preprocessing techniques could be explored on this data such as normalization, outlier removal, other methods of missing value replacement, etc. For lack of time, these were not considered, but they could be a fruitful area for future study. `CatBoost` can handle categorical data, so no encoding of categorical features was necessary [4].

4.2 Feature engineering

Historically, stocks have the potential of being split or reverse-split [5]. When a stock splits it issues more stocks to the current shareholders and the price of the stock per share decreases. In a reverse-split, exactly the opposite occurs. Current shareholders now earn fewer shares, but each share is worth more. In both cases the company is worth the same amount of money on the market. Stock splits and reverse-splits affect the closing price of a company, so these should be taken into account with what is known as the adjusted close price [3]. The competition hosts shared a function to calculate the adjusted close price, so this was incorporated before creating any new features which might rely on the close price [6].

Based on the guidance in [3], twenty total new features were added to the price data: the Moving Average (MA), the Exponential Moving Average (EMA), stock return, and stock volatility, each calculated over a period of 5, 10, 20, 30, and 50 days. The EMA can be thought of as a weighted average, since it places more emphasis on more recent data [7]. The volatility measure used was the standard deviation over each period.

Figure 18 shows the features that were added and their utility in understanding the price movements in a particular sector. If we look closely at the MA and EMA plots, whenever the 10-day average crosses the 50-day average from above, the closing price is likely to fall. Whenever the 10-day average crosses the 50-day average from below, the closing price is likely to increase. Comparing the stock return and volatility plots, the 50-day period exhibits more fluctuation than the 10-day period in the returns, whereas the reverse is true in the volatility.

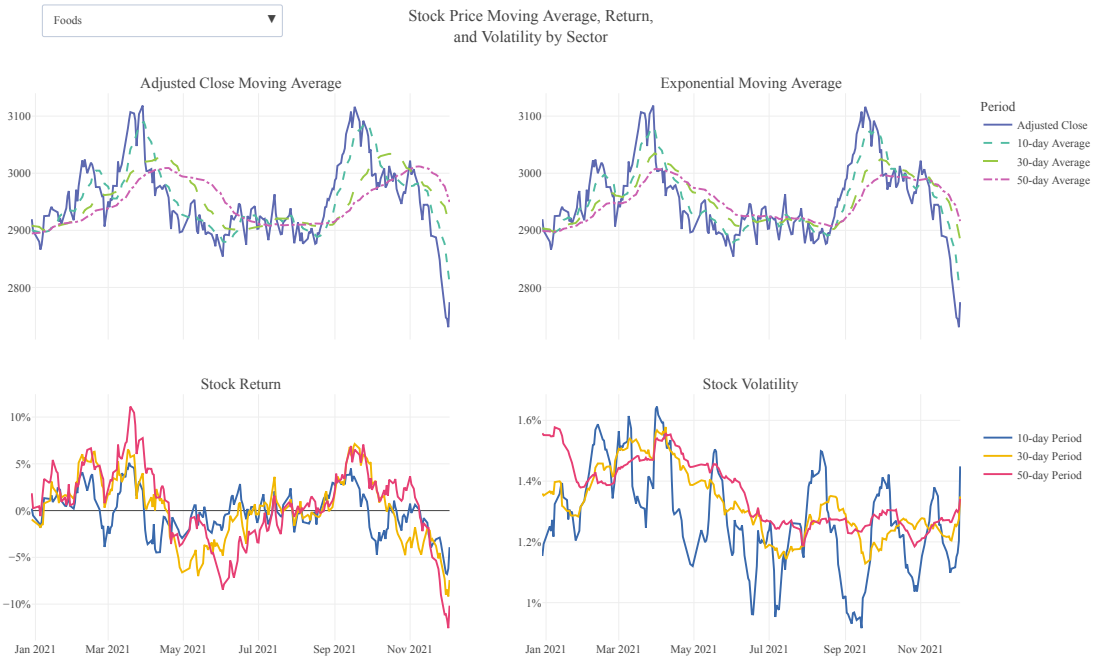

Figure 18: Moving Average (MA), Exponential Moving Average (EMA), returns, and volatility for one sector

4.3 Training

4.3.1 Creating training and validation sets

This problem essentially consists of a time-series regression, so typical train-test splits and k-Fold cross validation schemes will not work for validating the model because these methods shuffle the data and ignore its time-series nature. It is possible, however, to adapt train-test split methods to respect the temporal aspect of the observations. Another method is to use what is called "walk-forward validation." With walk-forward validation, the model is updated each new time step that new data is given to the model. Walk-forward validation may be helpful for the stock prediction problem, since an aspect of the competition is that new data is generated each week or so. For the purposes of this project, `scikit-learn`'s implementation of `TimeSeriesSplit` was chosen for model selection [8]. It allows for multiple time-series-based train-test splits to be easily created from a set of data, and uses a variation of the k-Fold algorithm for cross-validation. The number of folds used was 10, with a gap between the train sets and test sets of 10,000 to help prevent leakage.

4.3.2 Performing model selection

Interestingly, the best cross-validation performance was observed when using a Poisson loss function. Other loss functions were used, including the Root Mean Squared Error (RMSE) and Quantile loss functions, but none of these achieved better performance than the model trained using the Poisson loss. The Poisson loss is defined as:

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N} (\hat{y}_i - y_i \log \hat{y}_i)$$

where $\hat{y}$ is the prediction for target variable y. Normally this loss is used when the nature of the problem closely resembles a Poisson distribution or can be modeled as a counting problem. According to the `CatBoost` docs, a weighted version of the Poisson is used, where if the bootstrap type is also set to Poisson, the weights for each example are drawn i.i.d. from the Poisson distribution [9]. Initially the number of iterations was set to 1000. `CatBoost` automatically selects the best iteration to use for training, but it was observed that for a few folds the best iteration was taken to be 999, so the number of iterations was increased to 2000 to observe the results. The results for the average Sharpe ratio dropped with increased iterations. Tables 1 and 2 summarize the final parameters used and performance metrics achieved through model selection.

Parameter	Value
`n_estimators`	1000
`objective`	Poisson
`learning_rate`	0.01
`boostrap_type`	Poisson
`reg_lambda`	3

Table 1: Summary of parameters used in final model

Metric	Value
Submission score	0.1476
Std. deviation	0.09

Table 2: Summary of metrics on all validation sets

We can see from Figure 19 that around training round 180, the training and validation error diverge quite a bit. While the round on which this occurred was different for each cross-validation fold, similar diverging behavior was observed on each fold.

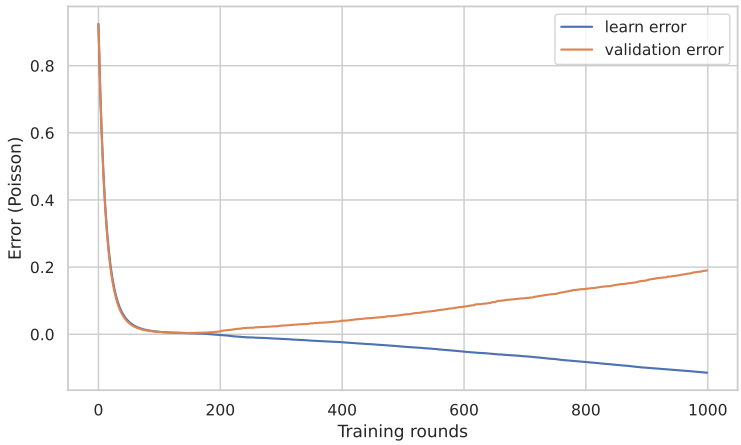

Figure 19: Training and validation error per training round for one cross-validation fold

Figure 20 shows what are defined as the most important features when training this model. These include the trading volume and several of the added features, such as the 20- and 30-day volatilities.

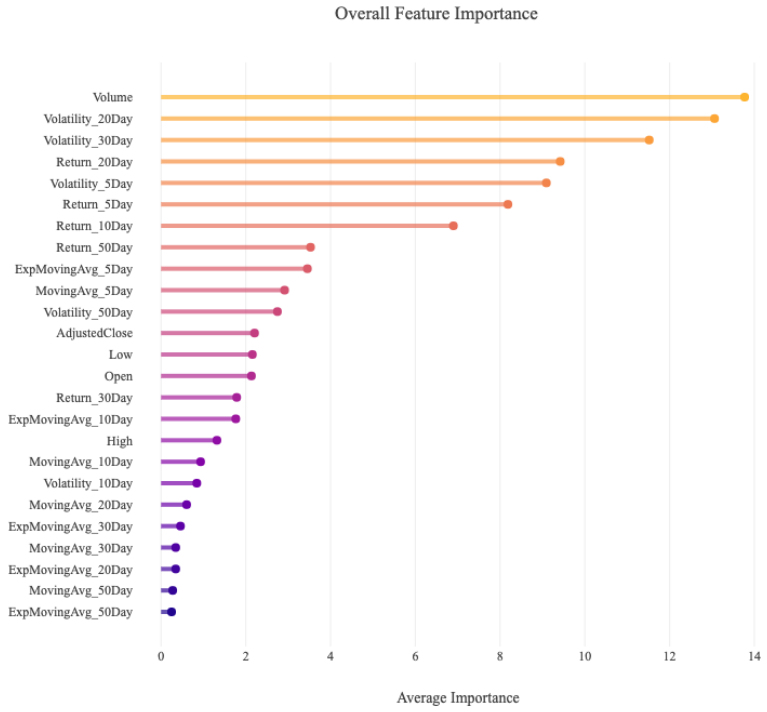

Figure 20: Feature importance averaged across each cross-validation fold

5 Evaluation and summary

5.1 Evaluation metric

The metric used for this competition is what is known as the Sharpe ratio. It is a method of calculating the "return per unit of risk" involved in an investment [10]. The formula for calculating this metric is taken from [11].

1. Calculate change of closing price for stock k at time t:

$$r_{(k,t)} = \frac{C_{(k,t+2)} - C_{(k,t+1)}}{C_{(k,t+1)}}$$

2. Calculate daily return spread ("linear function" is just a mapping of 200 weights in the interval $[2, 1]$, equivalent to `numpy.linspace(start=2,stop=1,num=200)`:

$$S_{\text{up}} = \frac{\sum_{i=1}^{200}(r_{(\text{up, t})} \times \text{linear function}(2,1)_i)}{\text{Average}(\text{linear function}(2,1))}$$

$$S_{\text{down}} = \frac{\sum_{i=1}^{200}(r_{(\text{down, t})} \times \text{linear function}(2,1)_i)}{\text{Average}(\text{linear function}(2,1))}$$

$$R_{\text{day}} = S_{\text{up}} - S_{\text{down}}$$

3. Calculate score (Sharpe ratio):

$$\text{Score} = \frac{\text{Average}\left(R_{\text{day}_1 - \text{day}_x}\right)}{\text{STD}\left(R_{\text{day}_1 - \text{day}_x}\right)}$$

5.2 Results

Metric	Value
Submission score	-0.103
Ranking	1488

Table 3: Summary of initial competition submission (trained only on data in `train_files`)

Metric	Value
Submission score	-0.018
Ranking	1439

Table 4: Summary of final competition submission after also training on data in `supplemental_files`

Unfortunately, based on the results seen in Tables 3 and 4 it looks like the model does not generalize well to new data. Because of the complexity of this problem, and the many possible avenues for improvement, it remains an open-ended question on where to go next. Probably the biggest mystery about the current model is why the Poisson loss achieved the best results, which may be linked to the fact the the Poisson bootstrap was also used to pick the example weights. There is also room for improvement through more rigorous model selection based on grid search or other methods, as well as a number of preprocessing steps that could be taken to shape the data's distribution.

6 What I learned

Several Kaggle notebooks were invaluable for digesting the competition API and the nature of the data. [12] and [13] were probably the most helpful in understanding the competition guidelines and metrics. [14], [15], [16], [17], [18], and [19] were great for learning how to frame a set of time-series data as a supervised learning problem, as well as the issues associated with cross-validating a model on such data. [3] was extremely helpful in identifying trends in the data, for some guidance in creating new features, as well as for help with implementing the cross-validation scheme used. [20] and [2] allowed me to grapple with the huge number of potential parameters and configurations for the `CatBoost` algorithm. Although many of these parameters are similar to those in `XGBoost`, `CatBoost` does have some different behaviors, particularly when dealing with accessing the GPU. Overall, this was a challenging project and many lessons were learned in dealing with such a complex problem.

References

[1] (2022) Jpx tokyo stock exchange prediction. [Online]. Available: https://www.kaggle.com/competitions/jpx-tokyo-stock-exchange-prediction

[2] (2022) Catboost. [Online]. Available: https://catboost.ai/en/docs/

[3] (2022) Jpx stock market analysis & prediction with lgbm. [Online]. Available: https://www.kaggle.com/code/kellibelcher/jpx-stock-market-analysis-prediction-with-lgbm/notebook

[4] (2022) Categorical features. [Online]. Available: https://catboost.ai/en/docs/features/categorical-features

[5] B. Beers. (2022) Understanding stock splits. [Online]. Available: https://www.investopedia.com/ask/answers/what-stock-split-why-do-stocks-split/

[6] (2022) Train demo - generating adjustedclose price. [Online]. Available: https://www.kaggle.com/code/smeitoma/train-demo#Generating-AdjustedClose-price

[7] J. Fernando. (2021) Moving average (ma). [Online]. Available: https://www.investopedia.com/terms/m/movingaverage.asp

[8] (2022) sklearn.model_selection.timeseriessplit. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html

[9] (2022) Regression: objectives and metrics. [Online]. Available: https://catboost.ai/en/docs/concepts/loss-functions-regression#objectives-and-metrics

[10] (2022) Sharpe ratio. [Online]. Available: https://en.wikipedia.org/wiki/Sharpe_ratio

[11] (2022) Jpx competition metric definition. [Online]. Available: https://www.kaggle.com/code/smeitoma/jpx-competition-metric-definition/notebook

[12] (2022) Easy to understand the competition. [Online]. Available: https://www.kaggle.com/code/chumajin/easy-to-understand-the-competition/notebook

[13] (2022) Jpx - detailed eda. [Online]. Available: https://www.kaggle.com/code/abaojiang/jpx-detailed-eda/notebook

[14] (2016) Time series forecasting as supervised learning. [Online]. Available: https://machinelearningmastery.com/time-series-forecasting-supervised-learning/

[15] (2017) How to convert a time series to a supervised learning problem in python. [Online]. Available: https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/

[16] (2019) How to backtest machine learning models for time series forecasting. [Online]. Available: https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/

[17] J. Brownlee, *Introduction to Time Series Forecasting with Python*, 2020.

[18] T. G. Dietterich, "Machine learning for sequential data: A review," *Lecture Notes in Computer Science*, vol. 2396, 2002.

[19] G. Bontempi, S. B. Taieb, and Y.-A. L. Borgne, "Machine learning strategies for time series forecasting," *Lecture Notes in Business Information Processing*, 2013.

[20] (2020) Catboost - an in-depth guide [python]. [Online]. Available: https://coderzcolumn.com/tutorials/machine-learning/catboost-an-in-depth-guide-python