Machine Learning on Cryptocurrency Perpetual Swaps

Abstract

We use BitMEX perpetual swap tick data for Bitcoin, Bitcoin cash, and Ethereum and build machine learning models (Linear and Logistic Regression, XGBoost) for predicting 8 quantities of interest (e.g., change in best bid price). We use empirical model selection methods and improve the chosen models using feature engineering and hyperparameter tuning. Our results show that model performance varies both in terms of the predicted quantity and the asset. Namely, prices are expectedly more difficult to predict and models for Bitcoin cash underperforms in comparison. Our study provides valuable insights and avenues for future research on machine learning in cryptocurrency markets.

Introduction

Research Objectives and Motivation

In this study, we use BitMEX data on financial instruments in the high frequency (HF) setting to investigate whether quantities, such as price changes, probabilities of up- or downticks in bid/ask prices, bid/ask sizes, and the number of buys/sells, can be predicted. By discretizing, transforming, and analyzing the data, and training machine learning (ML) models on the processed data, our aim is to arrive at a set of predictive models that could be used in real financial markets.

Our research objective is well motivated – most of stock trading these days is conducted by algorithms in the HF setting, so developing good models can lead to significant gains. At least, the study will raise new important questions for future work on the topic.

The remainder of the report will be structured as follows. First, we describe the data and methodology. Second, we lay out all results. Lastly, we continue by discussing the results and learnings from the study, and finish by providing avenues for future work on the topic.

Data and Methodology

Data

In terms of data, we use the publicly available BitMEX perpetual swap tick data for both top-of-book and trade information (BitMEX, 2022). As per the assignment, we focus on three swap contracts: XBTUSD (Bitcoin), ETHUSD (Ethereum), and BCHUSD (Bitcoin Cash). We use data from two separate days: March 18th for training and validation purposes, and March 20th for final testing. Having a one-day lag in between is used to prevent effects of serial correlation. Notably, the amount of data varies quite significantly between the assets with BCHUSD only having 6% of total observations.

Methodology

Preprocessing

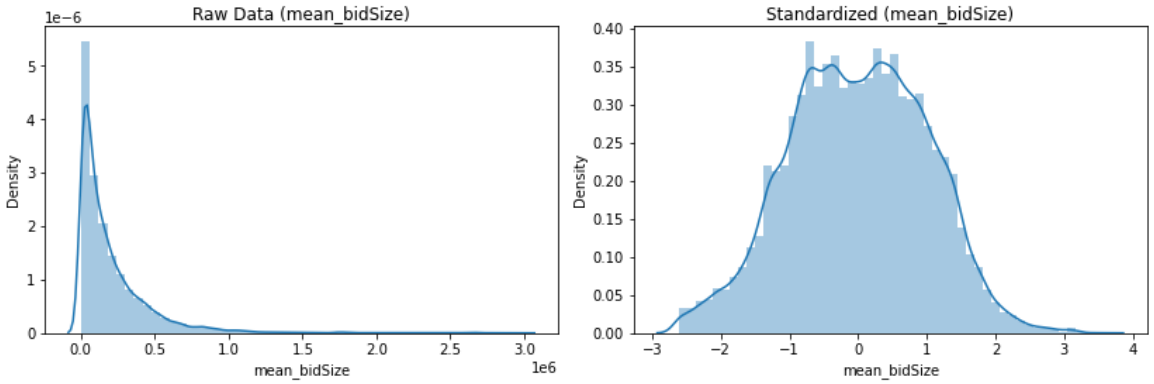
We first separate each asset in their own datasets. This is done due to the potentially differing dynamics between the assets and due to dataset size differences. As later discussed, we develop models and measure the performance of each asset separately. These datasets are then resampled to one second intervals, which corresponds to the given forecasting horizon. As the raw data is, depending on the update frequency, sampled at millisecond intervals, we calculate aggregate statistics like the mean, last value, and standard deviation for each 1 second interval. In case of no observations during the 1 second interval, we increase the interval until an update is found.

Positively, the data is already quite clean, but we also do some basic data cleaning and normalization. We for instance transform all absolute variables into relative measures, appropriately fill NaN values, and calculate and match respective target variables. In addition, we also create some calculated variables such as MicroPriceAdjustment and RelativeSpread. A full list of the predictive variables is included in Appendix A.

As many of the variables are not standardized, we use sklearn's StandardScaler and PowerTransformer to make the distributions more Gaussian-like when appropriate. In addition,

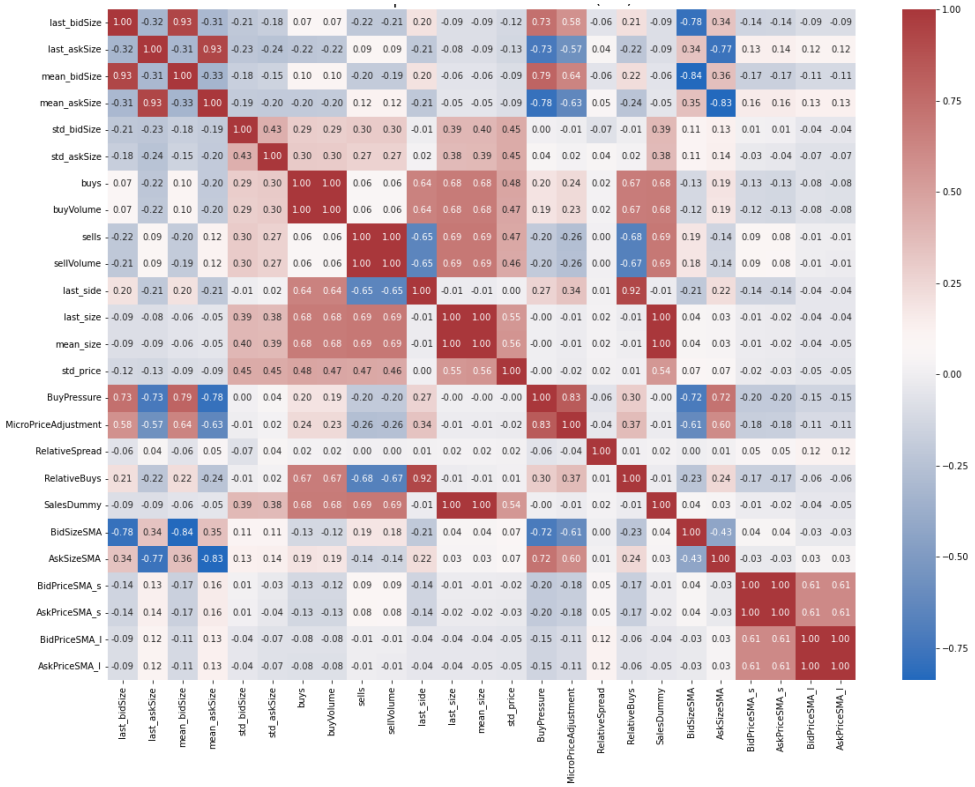
RelativeSpread is transformed into bins. Figure 1 below is a demonstrative example of mean_bidSize before and after standardization. A full list of the transformations is available in Appendix A. Standardization is important as many ML estimators behave poorly if the features do not follow a standard normal distribution. Therefore, if a feature has a variance larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly. (scikit-learn, 2022)

Figure 1: Mean bid size before and after standardization



We can also explore the variable relationships with a heatmap of feature correlations. Figure 2 below shows the feature correlations for XTBUSD. We can see that some of the aggregate statistics like the mean and last values are commonly strongly correlated. In addition, the number of buys/sells and their respective volumes are almost perfectly correlated, indicating the trade size does not vary significantly.

Figure 2: Heatmap of XBTUSD feature correlations



Model selection

To choose the optimal ML technique for each target variable, we use the training and validation sets to measure the performance of these techniques using all available predictors. The techniques we choose to test are linear regression, lasso regression, random forest, and XGBoost. All of these methods are very common for regression tasks, and particularly XGBoost has recently gained popularity in financial applications (Jidong & Ran, 2018). In addition, we use logistic regression for the two probability target variables, as it gives out natural probabilistic predictions.

To compare the techniques, we measure the R2 and mean squared error (MSE) of the models in each dataset. As the target variables have "pairs" (both bid/ask sides), we also consider their performance pairwise. We then choose the methods with the highest R2 and/or the lowest MSE. The average performance for each model is shown below in Figure 3 and Figure 4, with the chosen model bolded. We decide to use linear regression for changes in the bid/ask price, logistic regression for the probabilities of bid/ask ticking down/up, and XGBoost for the bid/ask size and number of sells/buys.

Figure 3: Model R2 for validation set

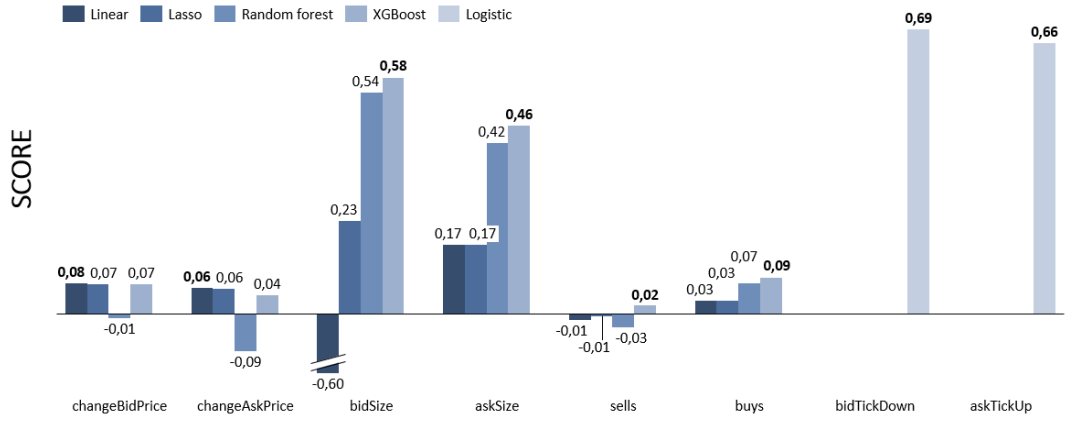
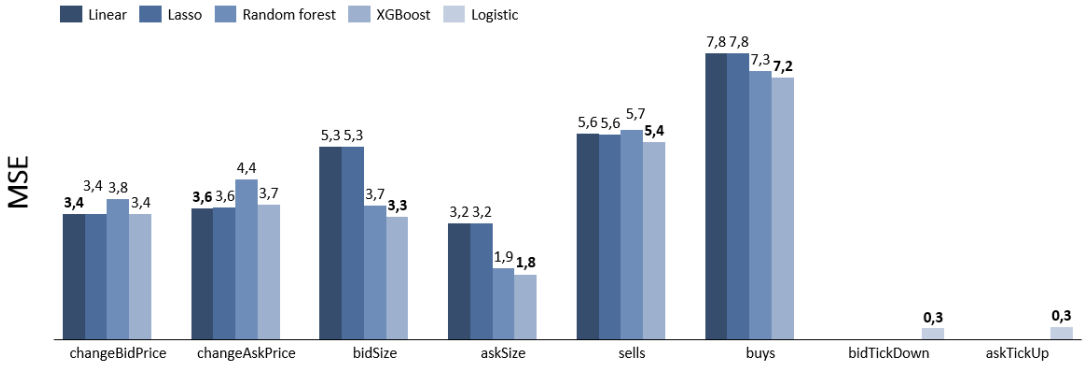


Figure 4: Model MSE for validation set

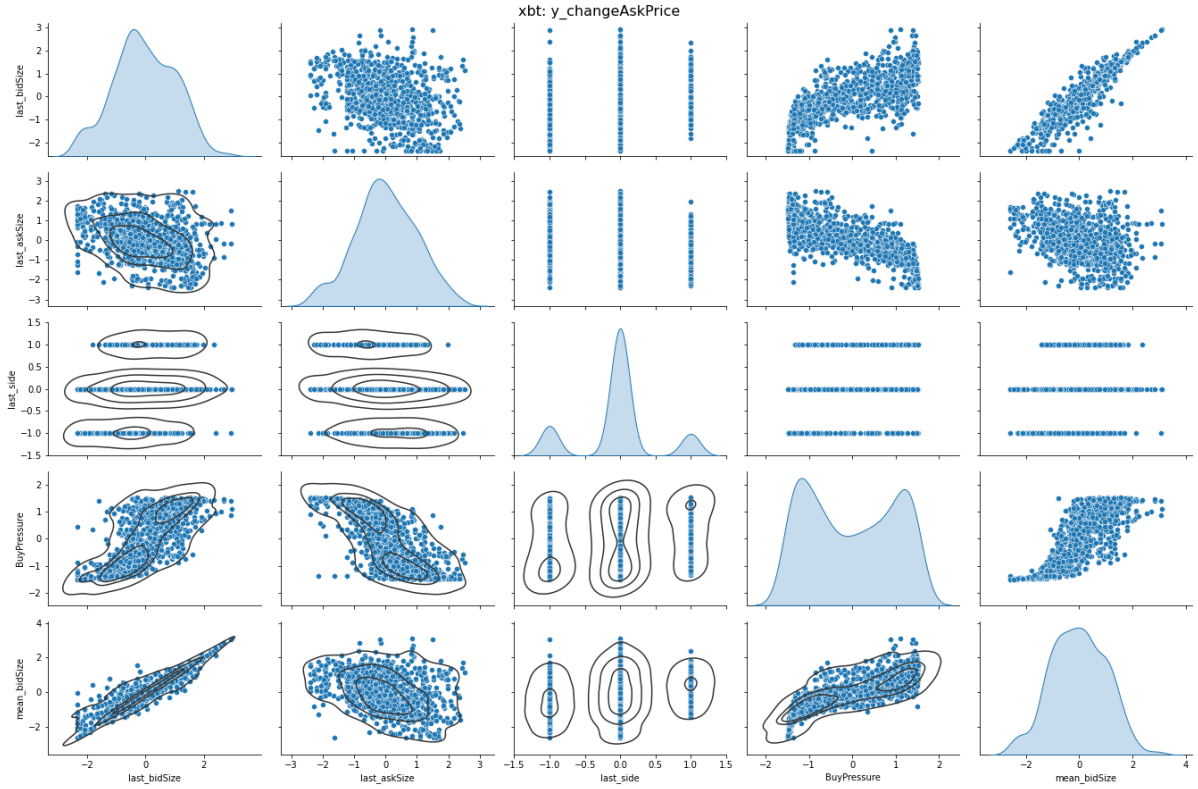


Note: bid/ask size in 1E9, changes in bid/ask price in 1E1. Lower is better.

The model choices are justified by the empirical findings and our preference towards simpler models. However, we can also further explore the feature relationships by creating a pair plot of selected features. Figure 5 below shows these relationships for XTBUSD. We can clearly see that many relationships are non-linear (e.g., relationship between mean_bidSize and

BuyPressure), and thus indicating linear or lasso regression might not be the optimal choices for these predictors. Instead, tree-based models, as shown by our empirical findings, might outperform in these situations.

Figure 5: Pair plot of selected features for dependent variable changeAskPrice from XBTUSD with randomly sampled 1000 observations. The diagonal shows kernel density estimates of the univariate distributions, above-diagonal entries show traditional scatter plots, and below-diagonal elements show scatter plots with kernel density contours for the joint distributions.

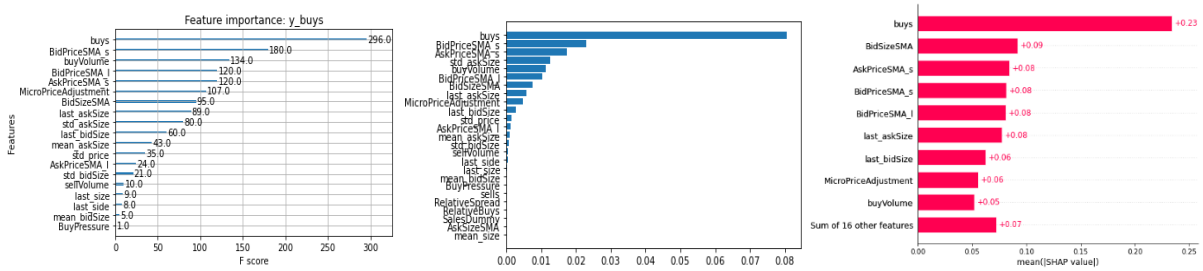


Feature selection

To minimize the complexity of the models and to improve their interpretability, we use feature selection methods to choose five features for each model. In case of over 90% correlation between predictors, we ignore the one with less significance and include the next most important predictor in the model. Due to differences in the methods, we consider XGBoost as well as linear and logistic regression separately.

For XGBoost we use three feature importance calculations: standard feature importance, permutation importance, and SHAP importance. The standard feature importance is calculated using the mean decrease in impurity within each tree (scikit-learn, 2022). The permutation importance is calculated as the decrease in model score when features are shuffled and the model refitted to estimate the importances (scikit-learn, 2022). The SHAP (SHapley Additive exPlanations) importance is based on Shapley values from a game theoretic approach to explain the output of ML models. (Lundberg, 2022) As these are calculated separately for each asset, we use an approximate weighted average of the importances for each predictor. Below is a demonstration of each of the importances for the number of buys.

Figure 6: Feature importance metrics for y_buys. Feature importance (left), permutation importance (middle), SHAP (right)



For linear and logistic regression, we use forward selection, which is a stepwise regression starting with an empty model and adding variables one by one that most improve the model (Statistics How To, 2017). Again, these are calculated separately for each asset. Therefore, the final predictors are chosen using a weighted average of the datasets and thereby giving less importance to BCHUSD as it is a significantly smaller dataset.

Parameter tuning

Of the chosen models, only XGBoost has tunable hyperparameters. As its performance is dependent on the chosen hyperparameters, we tune them with the training data separately for each target variable. We utilize RandomizedSearchCV, which is a randomized cross-validation method to find the best performing parameters. The parameters to be tuned are n_estimators, max_depth, eta, subsample, and colsample_bytree. Given the choices, we have 144 potential combinations of which we try 40 for each target variable. The chosen parameters are shown below.

Table 1: Tuned parameters for XGBoost models

	n_estimators	*max_depth*	*eta*	*subsample*	*colsample_bytree*
bidSize	250	2	0.05	0.8	0.8
askSize	250	3	0.01	1	1
sells	100	3	0.05	0.8	1
buys	500	2	0.01	1	1

We can clearly see some commonalities among the models. Particularly, all the models seem to favor low-depth trees with a small learning rate eta. Other differences may simply be due to the random sampling. One interesting property is the apparent link between the number of trees, their depth, and the learning rate from them. For instance, ‘buys’ prefers many short trees with a low learning rate, whereas ‘sells’ prefers relatively less but longer tress with a higher learning rate.

Results

R^2-scores and Mean Squared Errors

Based on the results from model and feature selection methods presented in the previous chapters, we fit the final models on data from March 18th, 2022, and use the data from March 20th, 2022, as a test set. We compute corresponding R^2-scores and mean squared errors (MSE) for both the train and test set for all assets. We then compute the average for both measures between the three assets. Figure 7 shows the results for R^2-scores and Figure 8 shows the results for MSEs.

Figure 7 shows that R^2-scores tend to be higher for the train set, which is expected. There is large variation in the deviation from the train set. For example, the R^2 for bidSize dropped from 0.77 (train) to 0.02 (test) while the R^2s for variables askTickUp and bidTickDown both increased three percentage units from 0.68-0.70 to 0.71-0.73. R^2s for buys and sells also weakened significantly and became negative. askSize, changeAskPrice and changeBidPrice all saw moderate negative drops in R^2.

Figure 8 helps to explain the changes in R^2s: test MSEs are higher for variables that experienced a decrease in R^2 in the test set (e.g., askSize, bidSize, buys, and sells). However, the test MSEs are lower for changeAskPrice and changeBidPrice despite having experienced a decrease in R^2 in the test set. This means that while the models were more accurate in the test set, they still explained less variation in the dependent variables than in the train set.

Figure 7: Average R2 of train and test set

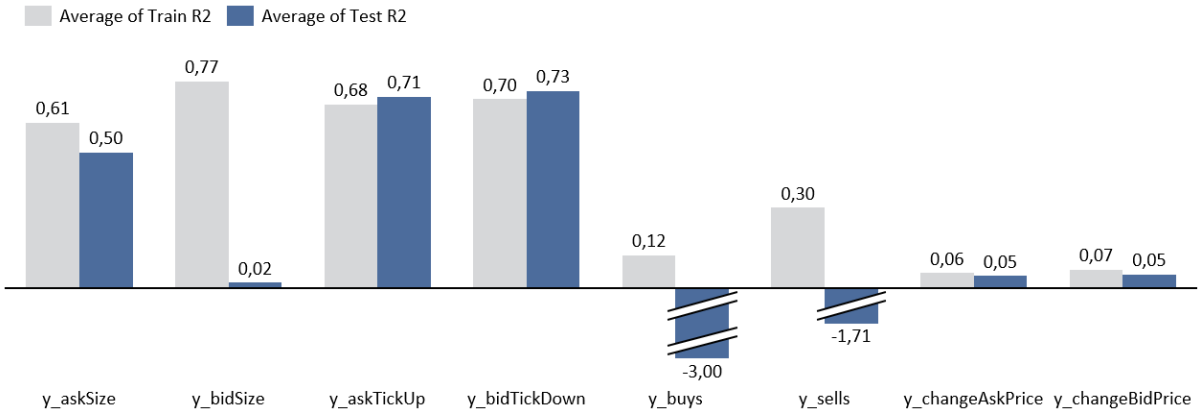
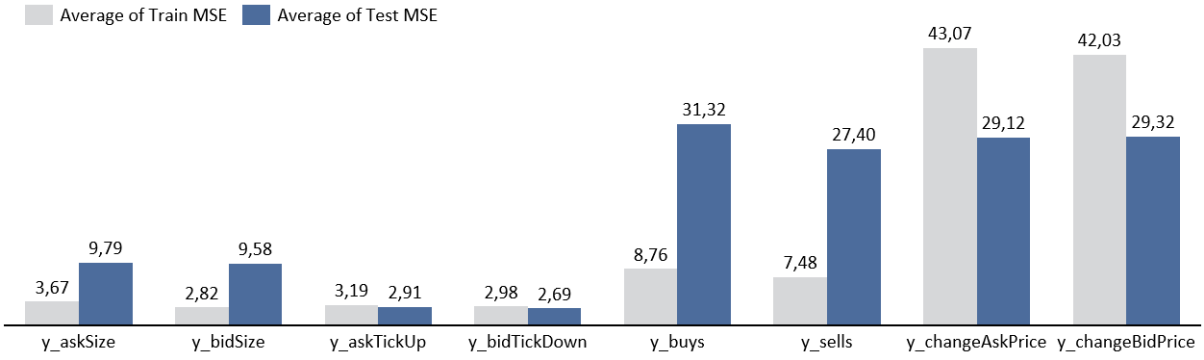


Figure 8: Average MSE of train and test set

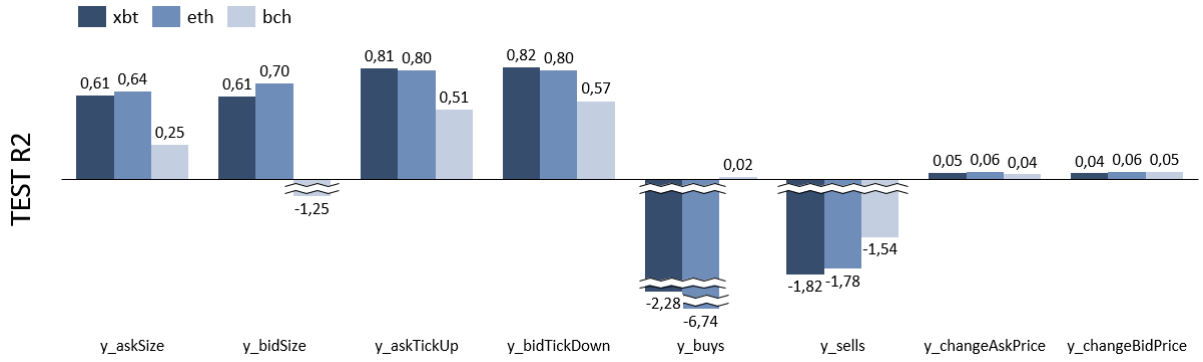


Note: bid/ask size in 1E9, ask/bid TickDown in 1E-1. Lower is better.

To assess the differences in model performance between the assets, we plot the test R^2 separately for the three assets. Figure 9 shows the resulting plot. We can see that there are some clear differences between the assets. For example, the R^2 for BCH is significantly worse for variables askSize, bidSize, askTickUp, and bidTickDown. In bidSize, the R^2 is -1.25 as opposed to 0.61-0.7 for XBT and ETH. This clearly explains why the decrease in test R^2 was so severe in Figure 7. However, for variables buys and sells, we see quite the contrary: test

R^2s are lower for XBT and ETH. For buys, the test R^2s for XBT and ETH are as low as -2.28 and -6.74, respectively, while it is 0.02 for BCH. The differences in test R^2 for sells in not as significant: values are -1.82 and -1.78 for XBT and ETH, respectively, while it is -1.54 for BCH. For variables changeAskPrice and changeBidPrice, the test R^2 is similar for all assets.

Figure 9: Test R2 for different assets



Overall, the models show some non-robustness, which is most likely caused by the BCH data. The model for variables buys and sells appear to perform worst while results are decent for variables askSize, bidSize, askTickUp, and bidTickDown excluding models for BCH.

Classification Metrics

To assess the performance of the classifiers for variables askTickUp and bidTickDown, we compute classification reports for all three assets evaluated on the test set. The results are shown in Tables 2, 3, and 4. They show that classifiers for XBT and ETH are considerably more accurate than the classifier for BCH in the case of variable askTickUp: the weighted average precisions are 0.91 and 0.83 for XBT and ETH, respectively, and 0.70 for BCH. Moreover, the weighted average recalls are 0.81 and 0.80 for XBT and ETH, respectively, and 0.51 for BCH. The results are analogous for variable bidTickDown. We can see that the mass of the observations is in zero for all assets and variables, meaning that most of the time, the ask and bid prices do not move up or down, respectively.

Analyzing precision more closely shows that only ~20% of the time we are correct when predicting that the ask ticks up. Same holds for predicting that the bid ticks down. On the other hand, the precision for no movement is very high with XBT and ETH, and quite high for BCH, meaning that we are correct > 80% of the time when predicting no movement.

Analyzing recall in more depth shows that for XBT, we can predict only ~40% of upticks in the ask and downticks in the bid. The proportion is even worse for ETH (~25%). However, for BCH, we can recall >50% of true positives, i.e., upticks in the ask price and downticks in the

bid price. On the other hand, the recall for true negatives is also worse in comparison to XBT and ETH (~50-60% as opposed to ~85%).

All in all, these results further support the fact that BCH is quite a different dataset compared to XBT and ETH. Precision and recall are generally high for true negatives (no up-/downticks in ask/bid) and low for true positives (up-/downticks in ask/bid). Considering these results, the performance of classifiers for XBT and ETH is better than the performance of the BCH classifier. Confusion matrices for all classifiers in the test set are also shown in Appendix B.

Table 2: Classification report for XBTUSD

	y_askTickUp (xbt)				y_bidTickDown (xbt)			
	precision	recall	f1-score	support	precision	recall	f1-score	support
0.0	0,96	0,84	0,89	62782	0,96	0,85	0,90	62692
1.0	0,14	0,44	0,21	3847	0,15	0,42	0,22	3937
accuracy	0,81	0,81	0,81	0,81	0,82	0,82	0,82	0,82
macro avg	0,55	0,64	0,55	66629	0,55	0,64	0,56	66629
weighted avg	0,91	0,81	0,86	66629	0,91	0,82	0,86	66629

Table 3: Classification report for ETHUSD

	y_askTickUp (eth)				y_bidTickDown (eth)			
	precision	recall	f1-score	support	precision	recall	f1-score	support
0.0	0,90	0,87	0,88	43028	0,90	0,87	0,88	42902
1.0	0,20	0,26	0,22	5377	0,21	0,28	0,24	5503
accuracy	0,80	0,80	0,80	0,80	0,80	0,80	0,80	0,80
macro avg	0,55	0,56	0,55	48405	0,56	0,57	0,56	48405
weighted avg	0,83	0,80	0,81	48405	0,82	0,80	0,81	48405

Table 4: Classification report for BCHUSD

	y_askTickUp (bch)				y_bidTickDown (bch)			
	precision	recall	f1-score	support	precision	recall	f1-score	support
0.0	0,82	0,50	0,62	19667	0,82	0,59	0,69	19538
1.0	0,22	0,56	0,32	4895	0,24	0,50	0,32	5024
accuracy	0,51	0,51	0,51	0,51	0,57	0,57	0,57	0,57
macro avg	0,52	0,53	0,47	24562	0,53	0,54	0,51	24562
weighted avg	0,70	0,51	0,56	24562	0,70	0,57	0,61	24562

Discussion and conclusions

Summary

In this study, we retrieved publicly available BitMEX perpetual swap tick data for both top-of-book and trade information for XBTUSD (Bitcoin), ETHUSD (Ethereum), and BCHUSD

(Bitcoin Cash). We performed feature engineering, exploratory data analysis, and feature importance methods along with model selection to arrive at a set of models to predict 8 different dependent variables in the data. We used linear, logistic, and XGBoost regression. Results showed that some of the models have decent predictive power in the test set, as model diagnostics in some cases even improved. Moreover, there appears to be inconsistencies in model performance between the assets: models performed considerably worse for the smallest dataset, BCHUSD. Lastly, some variables appear to be easier to predict than others. In particular, price changes are hard to predict, as one can expect due to market efficiency and random walk.

Learnings and Implications

The study taught us that it is important to assess the distributions of the variables in the raw data. Often, variables have strongly skewed distributions, and simple transformations can increase the predictive power of the models. Furthermore, in many cases, variables are strongly correlated, so bivariate analysis should be done to avoid using unnecessary features – especially in the case of models that do not handle multicollinearity well.

Another lesson is that hyperparameter tuning should be done to improve methods that are sensitive to changes in the parameters, such as XGBoost. There are significant improvements to be made by altering the default choices of parameters. We also learned that using a more advanced model like XGBoost does not always clearly outperform simpler models such as Linear Regression. Thus, simpler models should also be considered when appropriate and can in fact be the better choice over more complex models in cases where there is no clear distinction between the performances of different models.

Lastly, we also noticed that the performance of models varies between different assets albeit they are similar in nature. For example, there was a great difference between XBT and BCH, and the reason is arguably not fully explained by the difference in the number of observations between the datasets.

Suggestions for Future Work

We suggest that future work on this topic should consider whether altering the number of features significantly affects model performance. For example, it would be interesting to see whether only two or three variables could predict dependent variables as well as five features. While we considered only traditional R^2-scores, perhaps adjusted R^2 or AIC could be used to penalize additional features that do not enhance the predictive power of the model enough.

Furthermore, we suggest that more features could be engineered. We only came up with some features, but there is clearly more opportunities and features that could enhance our models. Previous literature could be examined more carefully to get ideas for features with proven predictive ability in the HF setting. Models could also be trained beyond just one day. Perhaps models would perform better if they were trained over a sustained period instead. Also, other instruments could be used to train and test the models.

One could also try other ML techniques than those considered in this study – perhaps methods like ridge regression or elastic net could perform well. More extensive hyperparameter tuning could also be done with more time and computing resources. Concurrently, one could use a smaller time interval than 1s. It could well be that the higher the frequency, the more the

predictive power increases. On the same note, subsampling could be considered: can we predict large or smaller values better? Can we predict more accurately during a certain time of the day? Questions like these are left unanswered in our study but answering them would be of interest.

Lastly, a natural next step would also be to use the models for some trading strategy and assess whether the models are good enough for making profits considering all transaction costs, which were out of scope in this study.

References

BitMEX. (2022). *Data*. Retrieved from https://public.bitmex.com/

Jidong, L., & Ran, Z. (2018). Dynamic Weighting Multi Factor Stock Selection Strategy Based on XGboost Machine Learning Algorithm. *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*, 868-872.

Lundberg, S. (2022). *SHAP*. Retrieved from Github: https://github.com/slundberg/shap

scikit-learn. (2022). *4.2 Permutation feature importance*. Retrieved from Documentation: https://scikit-learn.org/stable/modules/permutation_importance.html

scikit-learn. (2022). *6.3. Preprocessing data*. Retrieved from Documentation: https://scikit-learn.org/stable/modules/preprocessing.html

scikit-learn. (2022). *Feature importances with a forest of trees*. Retrieved from Documentation: https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

Statistics How To. (2017). *Forward Selection*. Retrieved from Statistics How To: https://www.statisticshowto.com/forward-selection/

Appendix

Appendix A: Predictive variables and transformations

Predictor	Description	Transformation
last_bidSize	Last bid size	Power
last_askSize	Last ask size	Power
mean_bidSize	Aggregate mean of bid size	Power
mean_askSize	Aggregate mean of ask size	Power
std_bidSize	Aggregate standard deviation of bid size	Power
std_askSize	Aggregate standard deviation of ask size	Power
buys	Buys during interval	Power
buyVolume	Total contracts bought during interval	Power
sells	Sells during interval	Power
sellVolume	Total contracts sold during interval	Power
last_side	The side of last trade	Power
last_size	The size of last trade	Power
mean_size	Aggregate mean of trade size during interval	Power
std_price	Aggregate standard deviation of trade prices during interval	Power
BuyPressure	Calculated measure for buy pressure	Standard
MicroPriceAdjustment	Calculated measure for expected price movements	Standard

RelativeSpread	Relative spread between bid and ask sides	Bins
RelativeBuys	Relative share of buys during interval	Standard
SalesDummy	Dummy variable if there has been a trade during interval	Power
BidSizeSMA	60 second simple moving average relative to current bid size	Power
AskSizeSMA	60 second simple moving average relative to current ask size	Power
BidPriceSMA_s	60 second simple moving average relative to current bid price	Standard
AskPriceSMA_s	60 second simple moving average relative to current ask price	Standard
BidPriceSMA_l	300 second simple moving average relative to current bid price	Standard
AskPriceSMA_l	300 second simple moving average relative to current ask price	Standard

Appendix B: Confusion Matrices for classifiers in the test set