Constructing trading strategy ensembles by classifying market states

Michal Balcerak[*][1] and Thomas Schmelzer[†][2]

[1]Institute for Theoretical Physics,
Georg-August-Universität Göttingen, Germany
[2]Faculty of Business and Economics (HEC Lausanne),
University of Lausanne, Switzerland

Rather than directly predicting future prices or returns, we follow a more recent trend in asset management and classify the state of a market based on labels. We use numerous standard labels and even construct our own ones.

The labels rely on future data to be calculated, and can be used a target for training a market state classifier using an appropriate set of market features, e.g. moving averages. The construction of those features relies on their *label separation power*. Only a set of reasonable distinct features can approximate the labels.

For each label we use a specific neural network to classify the state using the market features from our *feature space*. Each classifier gives a probability to buy or to sell and combining all their recommendations (here only done in a linear way) results in what we call a *trading strategy*. There are many such strategies and some of them are somewhat dubious and misleading. We construct our own metric based on past returns but penalizing for a low number of transactions or small capital involvement. Only top score-performance-wise trading strategies end up in final ensembles.

Using the Bitcoin market we show that the strategy ensembles outperform both in returns and risk-adjusted returns in the out-of-sample period. Even more so we demonstrate that there is a clear correlation between the success achieved in the *past* (if measured in our custom metric) and the *future*.

Contents

[*]m1balcerak@gmail.com, balcerak.michal@stud.uni-goettingen.de
[†]thomas.schmelzer@gmail.com, thomas.schmelzer@unil.ch

1   Introduction

Using neural networks to predict financial time series data is today widely regarded as the old unfulfilled dream of quantitative finance. An idea would be to apply supervised learning and train a neural network with sub-windows of a time series to predict the next data point(s). So instead of using images of dogs and cats we use at some time t the last n points of a time series to predict a point following at some time $t' > t$. Given the non-stationary nature of time series market data and low signal-to-noise ratios, this is a rather ambitious problem.

For instance, rather than using n prices (or returns), we reduce the dimensionality of the problem by using $m << n$ features based on the very same n points, i.e. an optimal combination of m moving averages. Such questions have typically been addressed by linear regression. However, linear regression fails to exploit any non-linear effects between the features.

We do not stop by only modifying the input - we also alter the goals of our predictions. Rather than aiming for a (noisy) price trajectory we ask simpler questions more suitable for the machinery of machine learning. Our goal is to

quantify the probability p of a market being in a class or category c or moving into one within the next hours or minutes. This could be the probability for a trend reversion or a spike in volatility or volume. We rely on labels as recently made popular by López de Prado [2] but also create some on our own. The flexibility of labels allows us to design a strategy by emphasizing effects we try to cover.

For each label we ask for an optimal set of m features to approximate them. These features, through a classifier, induce a probability for the market to be in a particular label-class. We then ask for an optimal linear combination of those probabilities to execute trades. Rather than looking at a Sharpe ratio in an out-of-sample period we construct robust variations of this concept and penalize for a lack of trading activity, etc. Although we don't aim directly for it we observe high Sharpe ratios and attractive returns as an unavoidable collateral side effect.

2   Labeling

We describe a market by a time series of datapoints $p_{t_0}, p_{t_1}, \ldots$. Predicting unseen price data is a hard problem often resulting in the notorious estimate that the next price is just the last observed price.

Rather than aiming for the next price, we argue that the market is currently in a particular label-class which we ultimately want to identify without using any unseen future data.

Throughout this work we distinguish three such label-classes and identify them with the actions we intend to take:

- Buy. The market may start or continue to rise over the next few periods

- Sell. The market may drop over the next few periods, the volume may drop significantly or there is a spike in volatility.

- Neutral. We do nothing.

Obviously identifying buy opportunities is trivial with a good level of hindsight. Looking at a historic time series we can identify numerous buy opportunities. This process is subject to some constraints we set, e.g. we may argue that the price at time t was a buy opportunity if we see a significant rise over the next minutes following t.

We call the process of classifying over time the *labeling* of a time series. So the particular *label* is a time series mapping p_t to one of the three classes.

Numerous such labels can and should be used. We use the popular *threshold label* to discuss the concept. We define the return over the period t_i to t_{i+1} as

$$r_{i,i+1} = \frac{p_{t_{i+1}}}{p_{t_i}} - 1.$$

We introduce a threshold τ and use

$$y_i = \begin{cases} sgn(r_{i,i+1}) & |r_{i,i+1}| > \tau \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Note that $y_i = 0$ if $|r_{i,i+1}| \leq \tau$ otherwise $y_i = 1$ or $y_i = -1$.

We also use a continuous companion of this label function. We use a continuous interpolation between the two labels 1 and -1. So if $|r_{i,i+1}| <= \tau$ we use instead of $y_i = 0$ the function

$$y_i = \left(\frac{r_{i,i+1}}{\tau} \right)^3 .$$

We could use a simpler linear term. However, in our experiments we have made better experiences with this particular choice.

So given a historic time series with all its price jumps and chaotic behaviour we reduce it to a time series just oscillating between three label-classes. Obviously we loose some information in this process but one could also argue we emphasize the information we really care about. And we can always combine multiple labels.

Identifying the moments we have missed to make a profit can help to evaluate the quality of a strategy, however, its inherent delay renders it of limited use in a live trading setup.

The idea is to approximate the labels with market features (i.e. technical trading indicators) that do not use any future data. Once in live trading, we can live update the indicators and therefore talk about label-classes predictions. The threshold is often made dynamic using estimates for the current volatility.

We use a variation of this idea where rather than $p_{t_{i+1}}$ in the definition of the return we use a moving average of prices following t_i.

The construction of such labels is an exercise only done during the training phase of the strategies. Running a backtest based on the actions induced by the labels over this training period would be a severe mistake.

Although it would be possible to have labels based on all sorts of financial data, e.g. volume, we use here exclusively labels based on price data.

3 Market representation

The central idea of this paper is to approximate the labels with a set of functions, referred to also as market features. The functions we use are standard technical indicators. The art is to resolve the labels in a small set of such parametrized functions. Those parameters are chosen in a way to maximize the *label separation power* of those functions.

Although the arsenal of orthogonal functions, i.e. a set of sin waves, is generally a great choice for approximations, we believe it is not suitable to capture market dynamics. A Fourier transform of the label would learn everything about the seasonality of this label but is of very limited generalization in an out-of-sample period.

We present our ideas using a toy example of only two functions with one free parameter each. In Appendix: *feature space* we give a complete list of features we have used.

The set of feature functions we identify as *feature space*. The parameters are not completely free. They are integer numbers from intervals we define. Hence we can pick for each label from a finite set of such features.

3.1   Feature space

To illustrate a *feature space* on an relatively simple example, let us define it as 2 indicators with some possible parameters:
Example feature space:

- feature 1: $A[X]$, VWAP - SMA(X) where $X \in [2, 10]$ [minutes]

- feature 2: $B[Y]$, VWAP - SMA(Y) where $Y \in [30, 60]$ [minutes]

where VWAP stands for volume weighted average price of a givin minute and SMA(Z) stands for a moving average of the last Z minutes.

This set of two features has one feature that looks at relatively short term time horizon and one feature with relatively long term. We normalise their values to $[-1, 1]$ using local scaling by standard deviation and arctangent function.

Let us define a label as one of the threshold labels: 1.5 % price change in 5 minute window. We now face a dilemma - which features from the feature space should we use? There are 279 candidates (9 different feature 1 and 31 different feature 2).

Let us fix parameters to acquire 2 possible feature sets from the feature space and solve the dilemma there:

- feature set 1:
$$A[5], B[50] \tag{2}$$

- feature set 2:
$$A[10], B[30] \tag{3}$$

An approximator sees market states only through their market representation. It is essential that features used in the market representation will differ in values if they encounter different classes of our label of choice. To measure these differences we use the L1 distance between corresponding vectors of features values. The market representations are illustrated on Figures 1 and 2
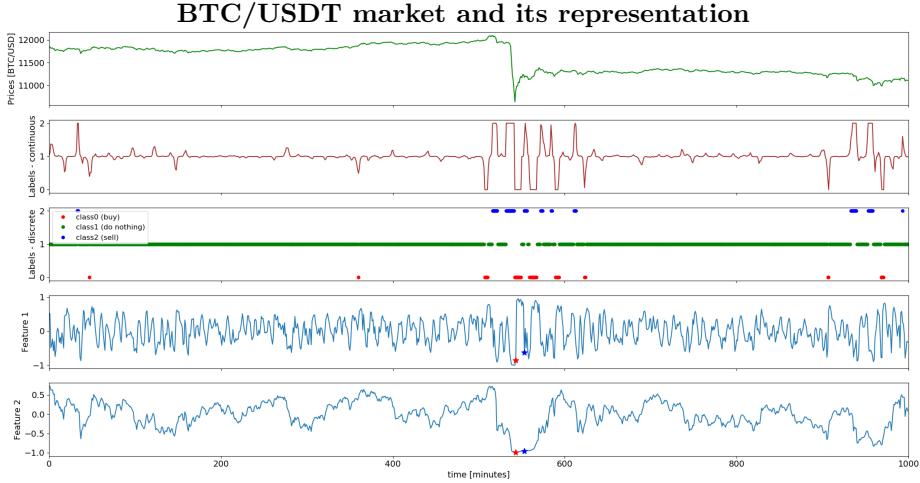
BTC/USDT market and its representation

Figure 1: Market VWMP, threshold label with 1.2% price change, 5 minutes time horizon and its Eq. 2 market representation. Zoomed out Fig. 2.

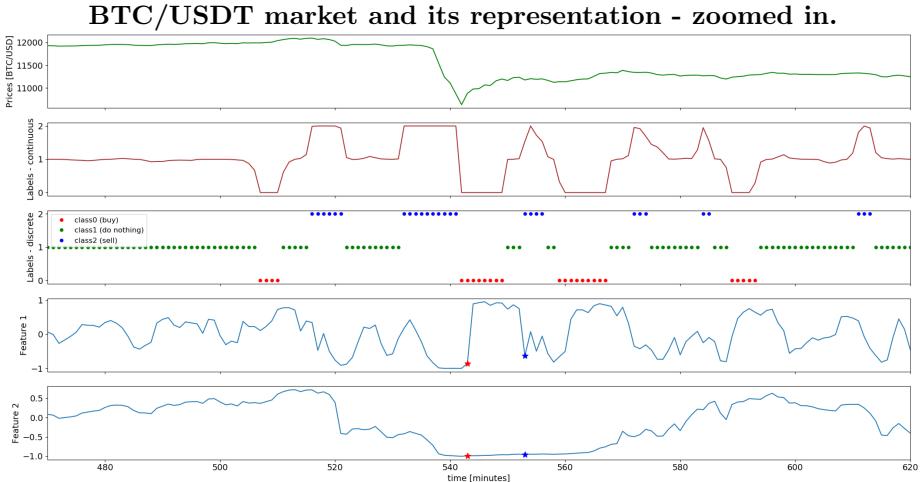**BTC/USDT market and its representation - zoomed in.**

Figure 2: Fig. 1 - zoomed in. The red and the blue star indicate two different market states label-class-wise but with similar feature values in Eq. 2 market representation. The two features fail to resolve the cross-label-classes which is the central problem of the market representation through feature selection.

Frequent low values of cross-label-class distances in a given market representation may cause severe problems for an approximator to correctly classify different market states as different label classes. Let us look at feature-wise market representation distances across a time period through a histogram of cross-label-class distances - Fig. 3

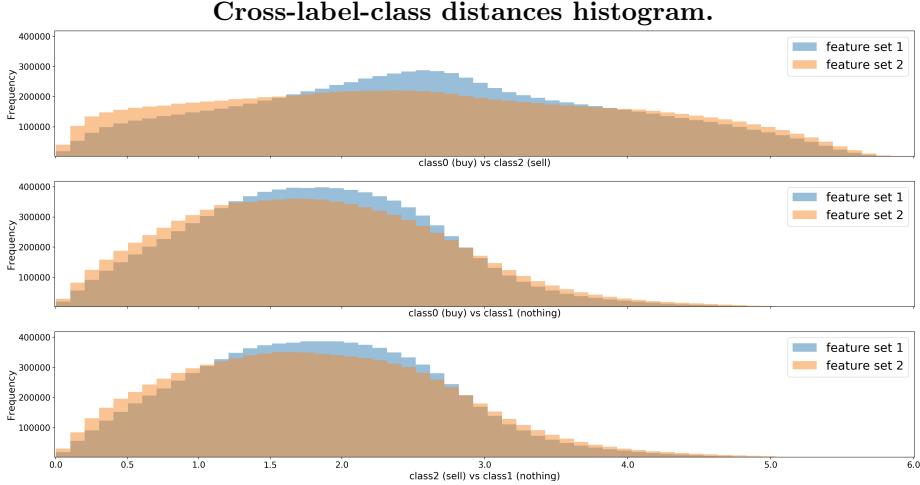**Cross-label-class distances histogram.**

Figure 3: Histogram of distances in Eq. 2 (feature set 1) and Eq. 3 (feature set 2) market representations. Both representations contain a lot of cross-label-class pairs with distances close to zero, however, the feature set 1 should be slightly better for an approximator than the feature set 2 because the feature set 1 distances histogram is skewed to the right. 3000 points from each class so 9 million points per histogram

Based on Fig. 3 we conclude this subsection saying that the Eq. 2 market representation is better for the threshold label with 1.2% price change, 5 minutes time horizon, than the Eq. 3 market representation. In it important to point out that both representations contain a lot of cross-label-class pairs with distances close to zero, so one should either search for a different feature set from the feature space or change the feature set altogether.

3.2 Fixing feature space parameters

We need a way to quantify goodness of a particular features set to represent a market for a particular label across a time period. Let us define a following metric for it:

Label separation power of a feature set: inverse of an area under a cross-label-class distances histogram (like in Fig. 3) weighted by a function to only select values relatively close to zero. Choice of the weighting function depends on the label and the numbers of feature in the feature space.

Choosing a particular feature set from a feature space for a given label is done through maximising their label separation power with Bayesian Optimization [5] and HyperBand [1].

3.3 Chosen market representation.

Feature space used in later parts of the paper contains 28 standard price and volume indicators and is formally defined in Appendix: *feature space*.

Threshold label with 1.2% price change and 5 minutes time horizon is one of the labels which we used for the analysis. The selected feature set has the following cross-label-class distance histogram:
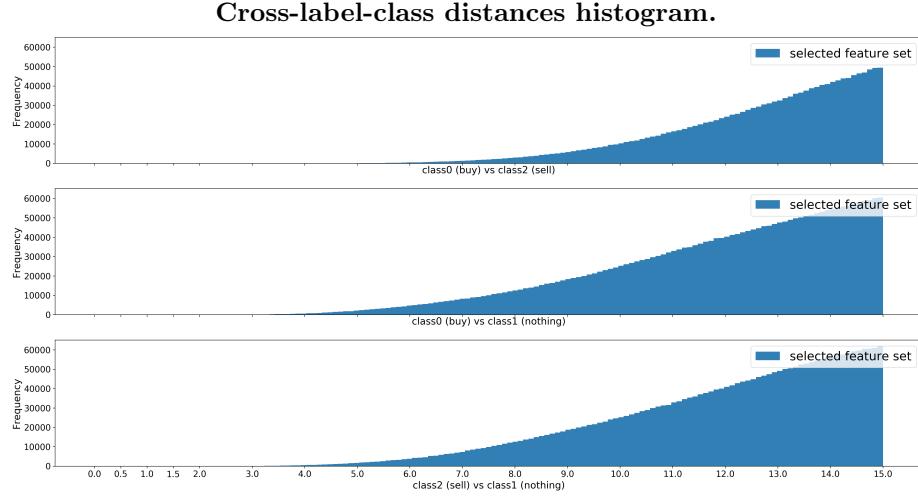
Cross-label-class distances histogram.

Figure 4: Histogram of distances of the selected features for the threshold label with 1.2% price change and 5 minutes time horizon. 3000 points from each class so 9 million points per histogram. Note fundamental differences between Fig. 3 and this one. Only **0.005%** of cross-label-class distances is below 3 on the same dataset as Fig. 3 histogram.
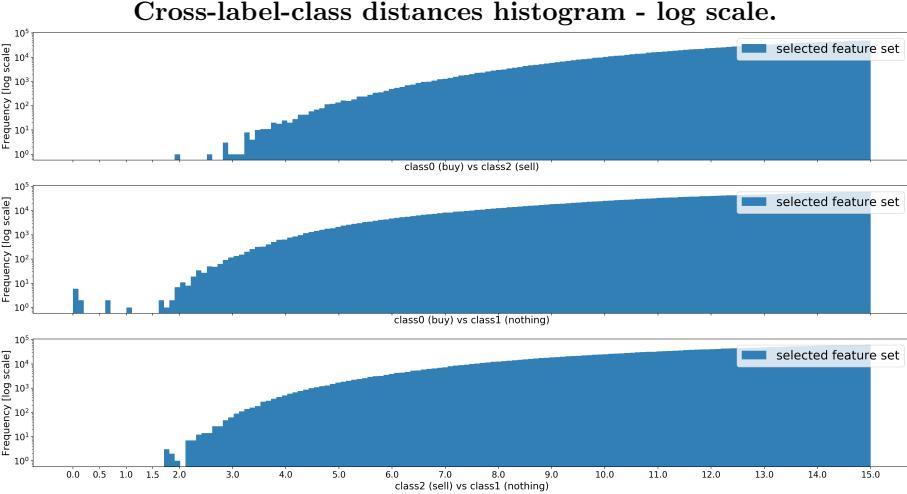
Cross-label-class distances histogram - log scale.

Figure 5: Log scaled Fig. 4 Y-axis-wise. class0 (buy) and class2 (sell) are the easiest to separate - this is in agreement with our intuition.

4 Approximation of labels

4.1 Goal of the approximation

There are eight labels to approximate (defined in Appendix: selected labels) using eight different market representations (from a feature space defined in Appendix: *feature space*). Instead of approximating the continuous labels, we choose to classify discrete-label-classes. This way we can focus on identification of the most important three regimes of the labels. In addition, this approach gives us a probabilistic way to determine confidences of our predictions. Each discrete-label-classes will be assigned a probability of occurring at a given time. The process of training a label classifier using historical market data is illustrated on the Fig. 6, whereas getting the label approximation is illustrated on the Fig. 7:

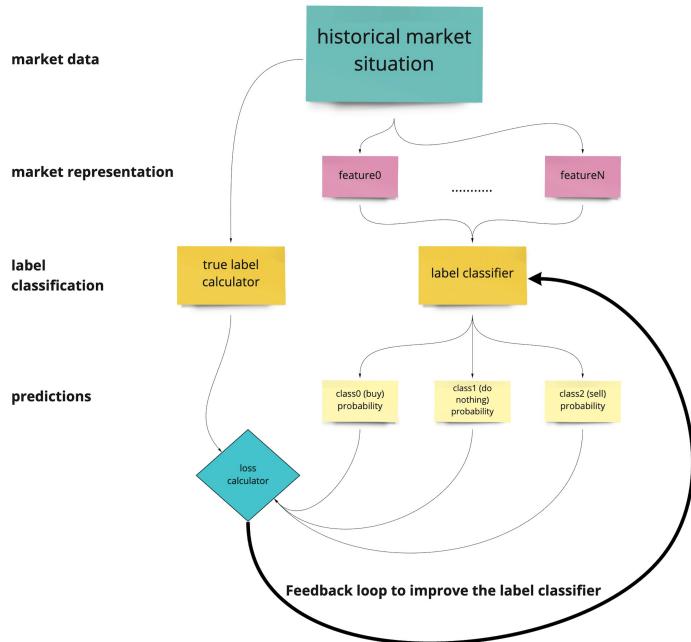**Process of label classifier preparation (training)**

market data

historical market situation

market representation

feature0 .... featureN

label classification

true label calculator

label classifier

predictions

class0 (buy) probability

class1 (do nothing) probability

class2 (sell) probability

loss calculator

Feedback loop to improve the label classifier

Figure 6: Illustration of training a label classifier using historical market data. Calculating the features requires knowledge of what happened in recent past, however, calculating true labels requires also knowledge of the near future.

Process of live label classification

market data

current market situation

market representation

feature0 .... featureN

label classification

label classifier

predictions

class0 (buy) probability

class1 (do nothing) probability
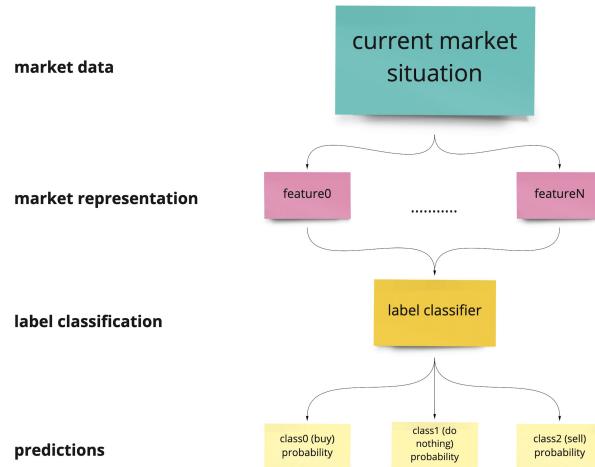
class2 (sell) probability

Figure 7: Illustration of label classification using current market data. There is no need to know near future in this process, which makes it possible to perform live.

4.2 Chosen algorithm

Based on Fig. 4 we see that an accurate approximator for this label and this feature set is possible to built but has to be non-linear. We conclude the same for the other seven labels. Because of a high number of datapoints in our training dataset (exact numbers in Experimental setup) and the requested non-linear behaviour we have decided to use a neural network classifier and a supervised learning algorithm. For hyper-parameter optimisation we used previously mentioned Bayesian Optimization [5] and HyperBand [1]. The loss calculator, which appears on Fig. 6, is built based on a concept called *loss scaling* which scales loss based on continuous labels. The central idea is to make class0 (buy) and class2 (sell) prediction accuracy more significant than class1 (do nothing) in the feedback loop to the label classifier during the training. This is an essential step because of heavy class-unbalance in the labels we have chosen. We construct the scaling in such a way that the sum of loss scale factors associated with class0 (buy) and class2 (sell) is equal to the sum of loss scale factors for class1 (do nothing). In addition, we reduce the loss scaling in-between 0-1 and 1-2 continuous label to make the training focus on clearer buy/do nothing/sell signals {0,1,2}.
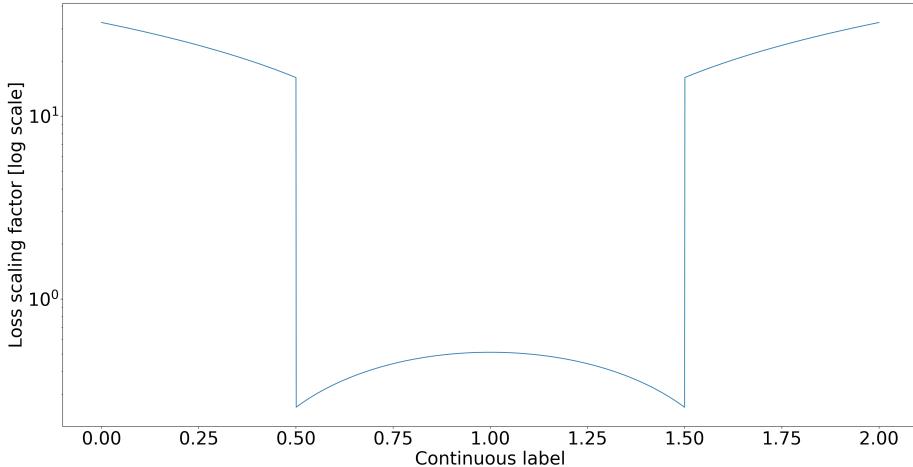

Figure 8: Continuous label based loss scaling factors used during the training process (Fig. 6). The sum of loss scale factors associated with class0 (buy) and class2 (sell) is equal to the sum of loss scale factors for class1 (do nothing). We reduce the loss scaling in-between 0-1 and 1-2 continuous label to make the training focus on clearer buy/do nothing/sell signals (0,1,2).

4.3 Classifier evaluation

A trained classifier acting on unseen data is illustrated on Fig. 9. Apart from industry-standard metrics like generalisation and confusion matrix coefficients, we also study our classifiers through Shapley Values [3][4][6]. This approach enables to see impact of a particular feature on the model output. If at this point, the data would not comply with our intuitions, we would not have chosen this

particular feature space and the algorithm for label approximation for further experiments.
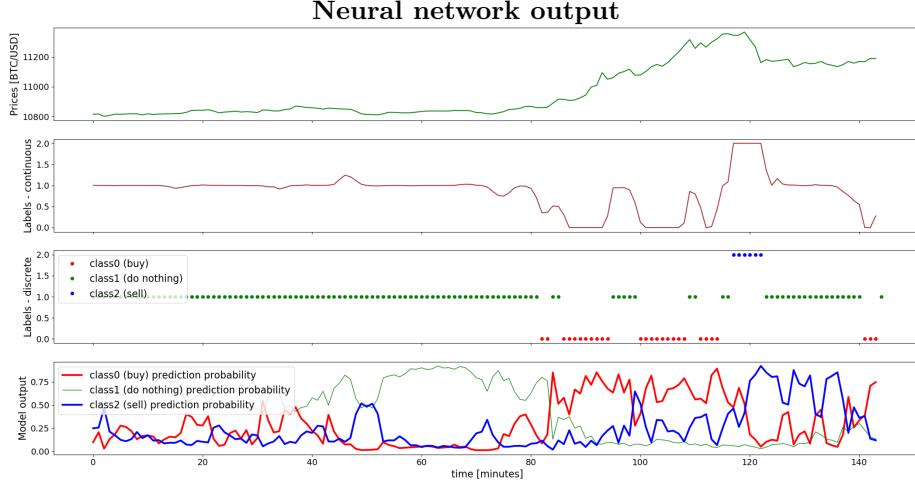

Figure 9: Prices, corresponding true labels and neural network model output predictions. Class0 (buy) in red, class1 (do nothing) in green, and class2(sell) in blue for the threshold label with 1.2% price change and 5 minutes time horizon. Out-of-sample data fragment.

5 Trading strategy

Our strategies are based on linear combinations of 16 model outputs - class0(buy) and class2(sell) of each 8 labels. We map it to $[0, 1]$ using a map:

$$\phi(x) = \begin{cases} (x+1)/2 & |x| < 1 \\ sgn(x) & \text{otherwise} \end{cases} \tag{4}$$

to acquire a trading signal y:

$$y = \phi(W^T X) \tag{5}$$

where y is a trading signal, W is a column of weights, X is a column of 16 model outputs - class0(buy) and class2(sell) of each 8 labels. We interpret the output value y as **a desired long position** on an asset. To execute trades, we use three thresholds: y_{buy} to buy, y_{sell} to sell and y_{width} to prevent execution of relatively small transactions. 19 constrained parameters in total. The parameter space are formally defined in Appendix: *strategy space*. Fig. 10 illustrates threshold-based trading execution. We increase (decrease) the long position if the trading signal y is above (below) y_{buy} (y_{sell}) and a distance between our previous position and the desired position is greater than y_{width}.
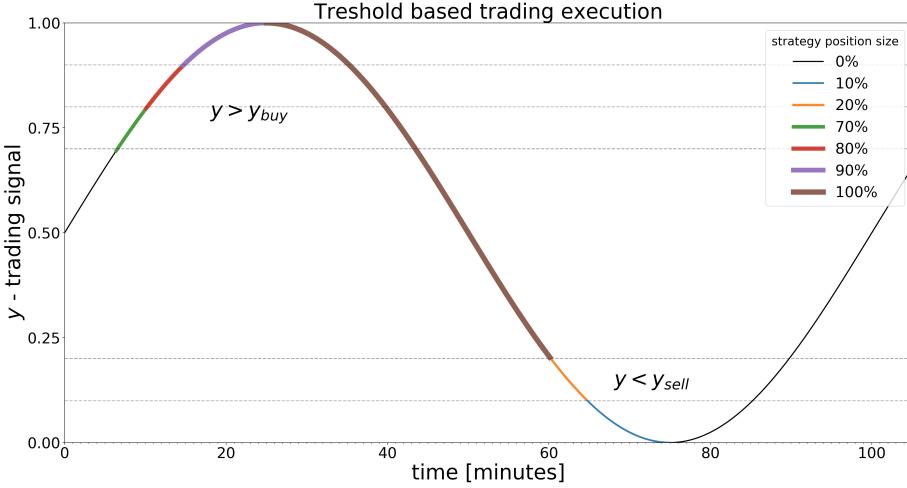
Figure 10: Illustration of the way our trading strategy changes the long position size based on a trading signal y. Thresholds for the illustration: $y_{buy} = 0.75$, $y_{sell} = 0.25$, $y_{width} = 0.10$.

6 Experiment

We ran an experiment of backtesting Trading strategy with 20 thousand different weights columns. The goal was to check performances in the *past* dataset and check their generalisation on the *future* dataset. Exact definitions of the used datasets are in the Experimental setup section. To acquire the weights we ran Bayesian Optimization [5] and HyperBand [1] with a task of producing weights with high performance in the *past* dataset. Because of random nature of the HyperBand algorithm we have acquired a full spectrum of strategies - from bad to good performance-wise.

6.1 Experimental setup

We based our experiment on BTC/USDT tick-by-tick transaction data recorded on the Binance exchange. We divided our dataset into three parts (dd-mm-yyyy):

- 01-01-2018 : 31-12-2019: Classifier *train&evaluate* dataset

- 04-01-2020 : 05-05-2020: Strategy *past* dataset

- 09-05-2020 : 19-09-2020: Strategy *future* dataset (out-of-sample)

Gaps in-between datasets were designed to prevent the look ahead bias. We aggregate the data into 1 minute datapoints. Transactions in our backtesting use *the next open price* to execute orders and carry a flat transaction fee equal to **0.05%**. This is a realistic estimation of a transaction cost which can be achieved on an exchange.

6.2   Results - single strategy performances

We have to find a selection metric for our strategies. First, let us talk about cross-datasets returns, where we put *past* dataset strategy returns against corresponding *future* dataset strategy returns. The comparisons are illustrated on the Fig. 11.
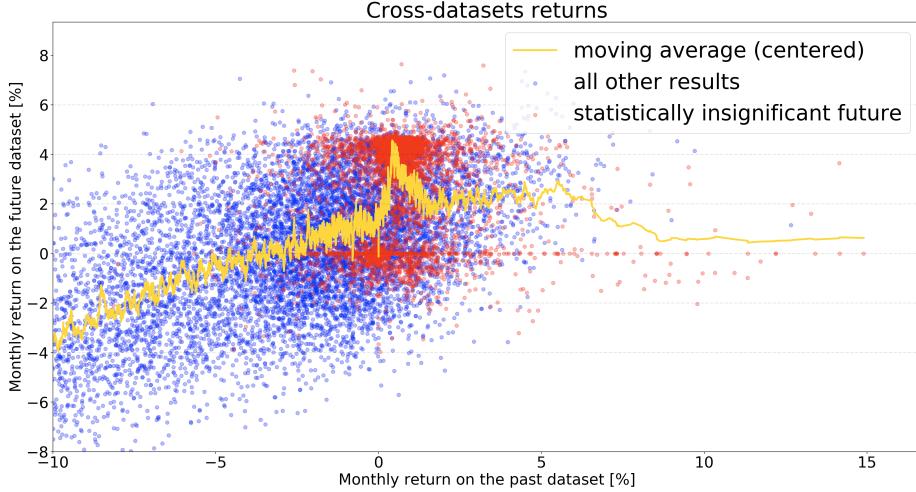

Figure 11: Cross-dataset returns of strategies. Returns on the *past* and *future* datasets are on the X and Y axis respectively. Statistically insignificant future red label corresponds to strategies that resulted in less than 5 trades per month on the *future* dataset

As illustrated on Fig. 11, a *past* dataset return cannot be used as a metric to select strategies with promising results in the future. The positive correlation brakes down around 5% montly return on the *past* dataset and the region with highest *future* returns is filled with statistically insignificant *future* performances. We need a better strategy selection method.

Let us introduce *score function* as follows:

$$S = MP - TP - MCIP \tag{6}$$

where:

- S: strategy score

- MR: montly return

- TP: transaction penalty - if number of transaction per month on the *past* dataset is lower than 30, then it is equal to the missing transactions per month

- MCIP: mean capital involvement penalty - if a mean capital involvement (mean long position) is lower than 25%, then it is equal to a half of missing percentages

The goal of the *score function* is to map the problematic low-return or statistically insignificant regions (illustrated on the Fig. 11) to low scores but preserving the positive performance correlation structure. Parameter values of the *score* function were chosen intuitively, before the cross-datasets studies. The number 30 in the transaction penalty was chosen simply because for Bitcoin it is the number of trading days in a month, and all of the used features can drastically change intra-day. An idea how to modify the score function, to make it less accidental is presented in Future work.

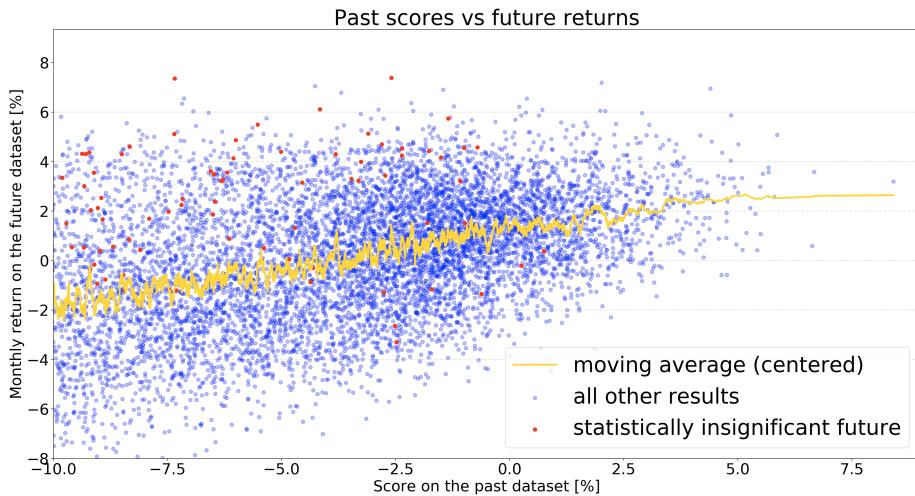Cross-score performances are illustrated on the Fig. 12 down below:

Figure 12: Cross-dataset performances of strategies. Scores on the *past* dataset and returns on the *future* dataset are on the X and Y axis respectively. Statistically insignificant future red label corresponds to strategies that resulted in less than 5 trades per month on the *future* dataset.

The higher the *past* dataset score the higher (on average) the *future* dataset monthly returns. The statistically insignificant regions were mapped out of the illustrated on Fig. 12 region. The *score* can now be used as a selection metric for our trading strategies.

6.3   Results - strategy ensemble performances

Strategy ensemble backtests using Top100, Top20, Top10, and Top5 *past* score-performance-wise models are illustrated on Fig. 13, Fig. 14, Fig. 15, and Fig. 16 respectively.
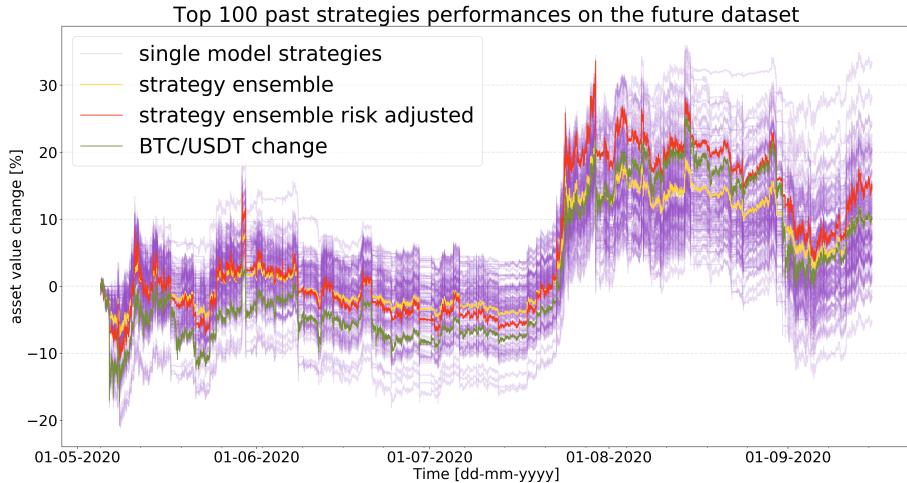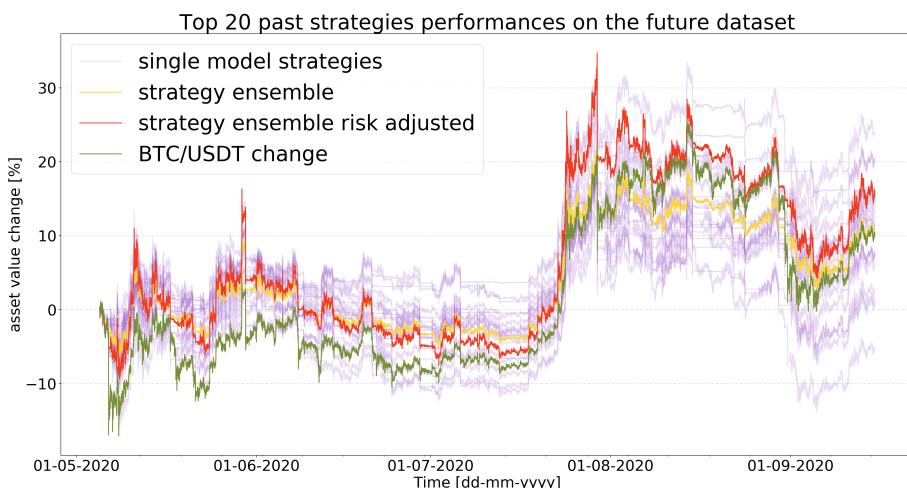
Figure 13: Performance of top 100 score-wise strategies and a corresponding strategy ensemble on the *future* dataset. Risk adjustment is performed through scaling the strategy ensemble returns by $\frac{\sigma_{BTC/USDT}}{\sigma_{ensemble}}$ where σ is the standard deviation of returns.

Figure 14: Performance of top 20 score-wise strategies and a corresponding strategy ensemble on the *future* dataset. Risk adjustment is performed through scaling the strategy ensemble returns by $\frac{\sigma_{BTC/USDT}}{\sigma_{ensemble}}$ where σ is the standard deviation of returns.
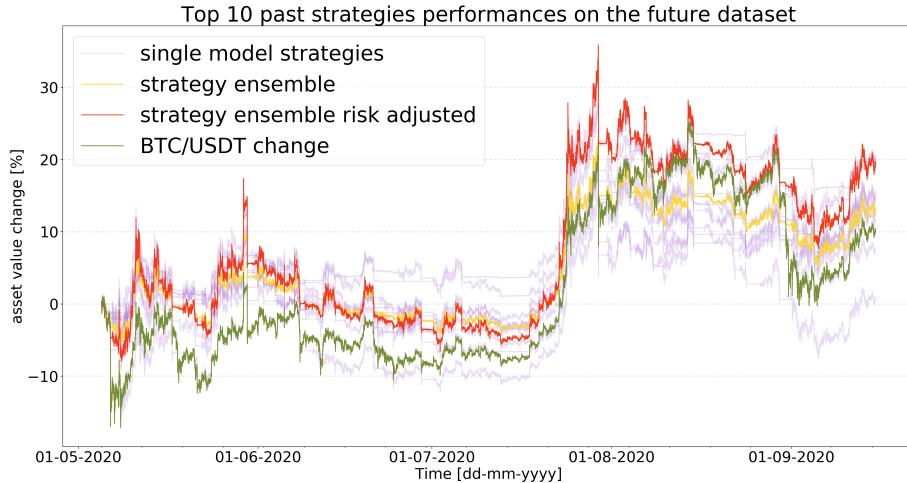
16

Figure 15: Performance of top 10 score-wise strategies and a corresponding strategy ensemble on the *future* dataset. Risk adjustment is performed through scaling the strategy ensemble returns by $\frac{\sigma_{BTC/USDT}}{\sigma_{ensemble}}$ where σ is the standard deviation of returns.
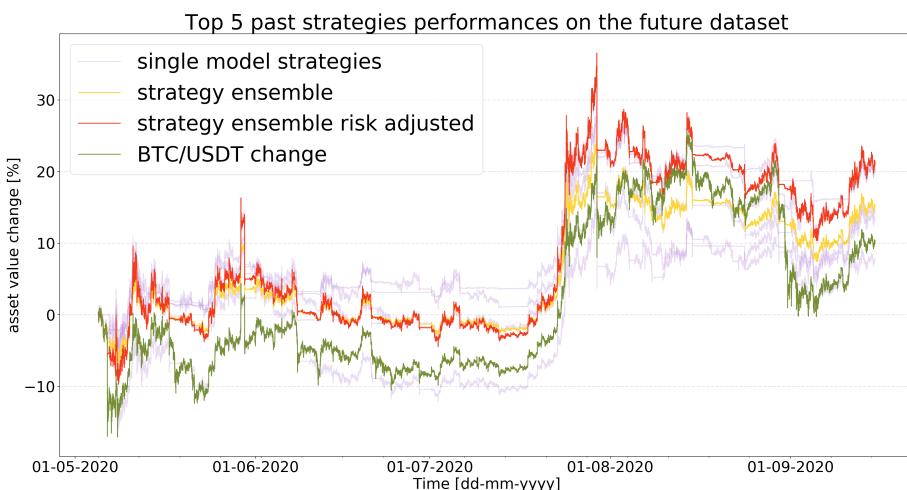

Figure 16: Performance of top 5 score-wise strategies and a corresponding strategy ensemble on the *future* dataset. Risk adjustment is performed through scaling the strategy ensemble returns by $\frac{\sigma_{BTC/USDT}}{\sigma_{ensemble}}$ where σ is the standard deviation of returns.

Both the returns and risk-adjusted returns are increasing as the average *past* score-performance increases.

7   Conclusions

Performances of created strategies increase in terms of return and risk-adjusted return on the out-of-sample *future* dataset as the *past score*-performance increases. Using top *score*-performance-wise strategies we achieved exceptional market-beating results. As of right now, using a framework which we have described can lead to further improvements of capital allocations of institutional investors with access to market data and computational power.

8   Future work

Making transaction rates in the *score function* definition dataset dependent, because static rates lead to ruling out potentially high performance strategies if they do not comply with dataset's dynamics. The optimal transaction rate should be based on characteristics of the *past* dataset - i.e. average volatility - and in general should not be hard-coded.

Changing strategies *on the fly* should further increase performance by swapping under-performing single model strategies with more promising substitutes. In this approach, the strategies are ranked for selection based on their up-to-date past performances. The effective *past* dataset would change periodically.

More sophisticated *feature space* would potentially lead to better classifiers and enable detection of sub-minute movements.

More sophisticated trading strategies - our linear combination was selected to reduce complexity. We now look into more complex solutions which are still relatively easy to interpret.

Running computations for longer to find higher *past* score-performance-wise strategies should further increase out-of-sample performances.

Acknowledgement

A   Appendix: selected labels

The 8 chosen labels can be categorised into 2 subcategories: threshold labels (see Labeling) and local extrema labels.

Threshold labels short descriptions:

18

- 1.2% price change in the next 5 minutes

- 1.2% price change in the next 60 minutes

- 2.2% price change in the next 2 minutes

- 3% price change in the next 5 minutes

- 3% price change in the next 60 minutes

The remaining 3 local extrema labels are a custom construct and are a material for a separate paper. Visualise and explore them all through our repository: GitHub Labels

B   Appendix: *feature space*

The *feature space* (see Market representation) we use consists of 28 functions and is illustrated on the Fig. 17 below:
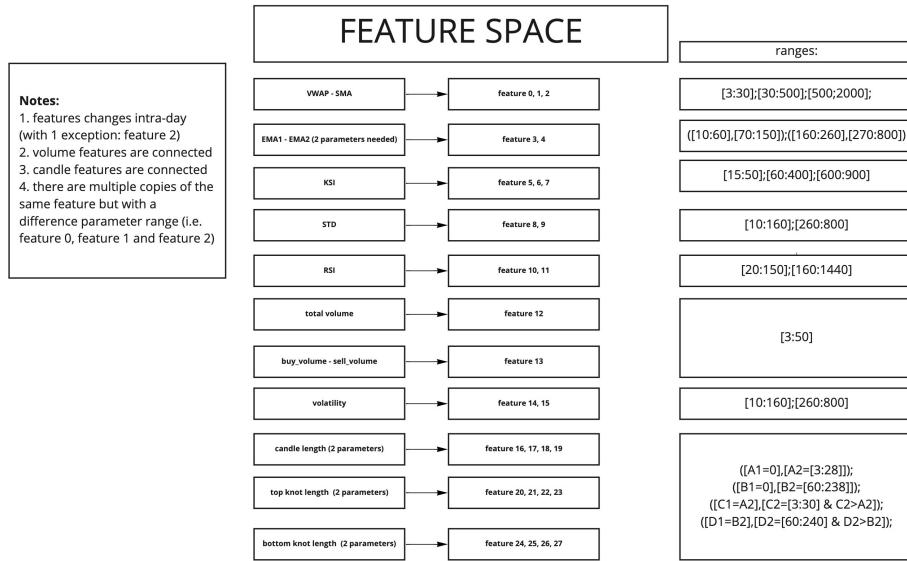
Definition of our *feature space* .

Figure 17: Definition of each of the 28 features and their corresponding parameter ranges that were used during our search for optimal market representations in Market representation. Parameters are integer only and represent minutes.

Ranges of possible parameters and types of indicators are based on our domain knowledge.

C   Appendix: *strategy space*

The *strategy space* (see Trading strategy) consists of 19 parameters: 3 thresholds and 16 weights.

Strategy thresholds: $y_{buy} \in [0.7, 1]$, $y_{sell} \in [0, 0.3]$, $y_{width} \in [0.02, 0.1]$.
Weights: $w_n \in [-1, 1]$.

References

[1] Lisha Li et al. "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization". In: *J. Mach. Learn. Res.* 18 (Jan. 2017), pp. 6765–6816. ISSN: 1532-4435.

[2] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. Wiley Publishing, 2018.

[3] Marcos López de Prado. *Interpretable Machine Learning: Shapley Values (Seminar Slides.* pt. Available at SSRN: June 27, 2020. DOI: 10.2139/ssrn.3637020. URL: http://dx.doi.org/10.2139/ssrn.3637020.

[4] L.S. Shapley. *A value for n-person games, vol II of Contributions to the theory of games.* en. Princeton: Princeton University Press, 1953.

[5] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems 25.* Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 2951–2959. URL: http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf.

[6] Erik Štrumbelj and Igor Kononenko. "Explaining prediction models and individual predictions with feature contributions". In: *Knowledge and Information Systems* 41.3 (Dec. 2014), pp. 647–665. ISSN: 0219-3116. DOI: 10.1007/s10115-013-0679-x. URL: https://doi.org/10.1007/s10115-013-0679-x.