University of Bonn

Master of Science in Economics

Identifying Arbitrage in Cryptomarkets with Algorithmic Trading:
A Machine Learning Approach

Submitted by

Raphael Redmer

Supervisor: Prof. Dr. Joachim Freyberger

June 14, 2020

Abstract

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Contents

1 Introduction

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

2 Data and Software

2.1 Data

In this setup, we use minute-binned OHLC data of crypto/USD-pairs obtained from the cryptocurrency exchange Bitfinex via its API ranging from 01.01.2019 to 31.12.2019. For each minute-bin, we collect *Open*, *High*, *Low*, *Close*, *Volume* and *Timestamp* data. *High* and *Low* denote the highest and lowest price respectively that was traded within this timeframe. *Open* and *Close* denote the first and last traded price. *Volume* denotes the total volume traded within the respective minute-bin. *Timestamp* denotes the point in time for each minute-bin as a UNIX-Timestamp, i.e., is the number of seconds that have passed since 01.01.1970 (Cite: ¡IEEE and The Open Group 2018¿).

Even though Bitfinex is the largest exchange for cryptocurrency with a ¡???¿ market capitalization and ¡???¿ different tradable coins, for most coins, the trading frequency is so low such that many crypto/USD-trading pairs have a considerable amount of minute-bins in which no volume was traded. In case of a crypto-pair having no volume for a particular minute, the API leaves out this bin when requesting its data resulting in missing bins. We resolved this issue by propagating price values from the last active minute-bin and setting the volume to zero. Further, we restricted the number of crypto-USD-pairs to the top ten pairs by market capitalization (see ¡plot¿). In addition, we decided to only take data from 01.01.2019 to 31.12.2019, since for most coins 2019 was the most active year in terms of trading frequency. Thus, the resulting data set contains roughly $10 \times 365 \times 24 \times 60 = 5.256.000$ rows.

2.2 Software

The programming language used for conducting this study is Python 3.7 (¡cite¿). For data preparation and feature engineering, we used Pandas and numpy (¡cite¿). Data Visualization was done via Maplotllib (¡cite¿). For the training of the models Logistic Regression, Random Forest and Support Vector Machine, we used the respective Scikit-learn implementation (¡cite¿). The Artificial Neural Network was trained using the Keras framework with Tensorflow backend to enable GPU calculation.

3 Methodology

Similarly to ¡Krauss et al. (2017)¿ and ¡cite arbitrage¿, the methodology of this paper consists of the following steps:

1. The entire data set is split into a training, a validation and a trading set.
2. The respective features (explanatory variables) and targets (dependent variables) are created
3. Each model is trained on the training set
4. Conduct out-of-sample predictions on the trading set for each model
5. Evaluate its accuracy and trading-performance on the trading set respectively
6. Go to Step 2, and repeat the same steps for a different feature- and target-specification

3.1 Training and Trading Set

In our application to minute-binned data, the test set, i.e. trading set, contains all observations from 01.11.2019 to 31.12.2019. The training set ranges from 01.01.2019 to 14.09.2019, and the remaining 15.09.2019 to 31.10.2019 is reserved for the valdation set. We decided against the usual k-fold cross-validation approach in order to emphasize the importance of future observation for the model, since its performance only gets evaluated on the future trading set.

3.2 Feature and Target Generation

Broadly following ¡cite¿, we generate the feature space as follows:

Let $P^c = (P_t^c)_{t \in T}$ denote the price process of coin-USD-pair c, with $c \in \{1, ..., n\}$. The price itself is the average between *Open* and *Close*.

Features: From the data set we obtain the following features:

Returns: Let $R^c_{t,t-m}$ be the simple return for coin c over m periods defined as

$$R^c_{t,t-m} = \frac{P^c_t}{P^c_{t-m}} - 1 \tag{1}$$

Volumes: Let V_t^c be the traded volume for coin c in minute-bin t scaled by Quantile-Transformer fitted separately for each coin only on bins with $V_t^c > 0$.

Target: Let $Y^c_{t+1,t}$ be a binary response variable for each coin c and $d = 120$ the size of the future time interval. It assumes value 1 (class *up*) if its future 120 min return $R^c_{t+d,t+1}$ is greater than its cross-sectional median across all pairs $(R^c_{t+d,t+1})^n_{c=1}$, else -1 (class *down*). Instead of just using the simple return $R^c_{t+d,t+1}$ as in (¡cite¿), we included an additional condition, which demands that $V^c_{t+d} > 0$ for realizing the

feature return. If not skip bins until you reach a bin $t^* = t + d + \delta$ in which $V^c_{t^*} > 0$, then realize return as in equation 1.

The reason for this further restriction is to make the training of the model more similar to the trading decisions in the backtest, since we only allow trades to be executed in a bin, if any volume was traded for the respective coin. We decided for the inclusion of volume such that the model has a measure for taking trading activity into account without breaking vital assumptions needed for testing the 1. Efficient Market Hypothesis (¡cite¿). In addition, the volume got scaled for each coin in order to make the measure more comparable across coins, since we are training a single universal model for each of the selected coins. Further, the Quantile-Transformation handles outliers (¡cite¿) and restricts the feature to an intervall ranging from 0 to 1.

3.3 Model Training

As explained in chapter 3.1, we construct a training set ranging 01.01.2019 to 14.09.2019, a validation set ranging from 15.09.2019 to 31.10.201, and a trading set ranging from 01.11.2019 to 31.12.2019. Further, we restrict the training and validation set by excluding bins for which no volume was traded in in the following bin or lagged values are not available.

We cross-validate the respective parameter space by first training the model on the test set an evaluating on the chronologically following validation set for each parameter combination. After obtaining an accuracy evaluation of each combination, we fit the model with the best validation performance on the combined training and valdation set.

3.3.1 Logistic Regression

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

3.3.2 Random Forest

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

3.3.3 Support Vector Machine

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

3.3.4 Artificial Neural Network

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

3.4 Trading Algorithm

For trading phase, we proceed similarly to (¡cite¿). After having trained each model for different specifications, we generate probability estimates for each class based on the trading sets feature's. These probabilities for each bin are then used as signals in bin t for each coin in order for the algorithm to decide whether to enter or close a position invested in a coin in the next bin in $t + 1$. We refer to these position as active. More specifically, the algorithm compares the probability estimate for each coin and decides whether to close the current position and enter a new one. The coin with the highest probability estimate for a down movement is considered as a candidate for the short position, if its probability is also above a certain threshold, since it is most likely to go down. The algorithm proceeds analogously for the long position. If a coin gets chosen this way and the position for this movement is active, then it proceeds to close this position. Therefore, we enter a long and a short position at most per bin t. In order get a better estimate of the return, we enter 60 initial short and long positions at different points in time, thus the resulting portfolio has 120 active positions at most. Then, we proceed to calculate aggregate values for each of these positions. To render the backtest more realistic, we incorporate the following constraints:

Minimum duration: Any active position has a minimum duration of $d = 120$, before considering closing it.

Execution gap: To account for the time it takes to generate a probability signal and submit the order accordingly, we introduce a execution gap of one minute. This means that when generating the signal from the bin in t, the order gets executed in $t + 1$ the earliest.

Volume constraint (orders): Orders for opening or closing a position only get executed in bin t, if any volume was traded in the respective bin.

Volume constraint (opening a position): If after submitting the order according to the sufficient probability signal the traded volume in bin t is zero, the order gets canceled and a new probability signal gets generated for bin t.

Transaction cost: For every order execution a transaction cost of ϵ-bps gets subtracted. Thus, the opening and closing of a position costs $2 \times \epsilon$-bps.

Keep active position: If the probability signal yields the same coin as the one from the respective open position, then keep the same position open for another $d = 120$ minutes before again generating another probability signal.

The algorithm described above can also be expressed in a pseudo-algorithmic way:

Algorithm 1: Pseudo-Algorithm for a single position (short and long are analogous)

```
Data: Minute-binned Test-OHLC-Data and its estimated probability signals
Result: Trading decisions and its realized returns
initiate position;
while row_number < (max_row_number − delta) do
    load row data given by row_number;
    obtain maximum probability pair;
    determine whether the pair fulfills prob-conditions;
    if sufficient probability signal then
        if no currently active position then
            open new position under constraint;
        else if maximum probability pair != active position pair then
            close current position;
    else
        if currently active position then
            close current position;
    if currently active position then
        row_number ← row_number + delta
    else
        row_number ← row_number + 1
```

4 Results

4.1 General Results

After training the models, we evaluated their accuracies on the remaining trading data ranging from 01.11.2019 to 31.12.2019. Before applying the model on the test data, we excluded bins for which no volume was traded in the following bin or lagged values are not available, as was already done to the training set.

	With Volume		No Volume	
Model	**Validation Score**	**Test Score**	**Validation Score**	**Test Score**
Logistic	52.35	53.02	52.18	53.22
Forest	52.71	53.27	52.30	53.53

Figure 1: This table illustrates the accuracy both for the validation and test set for different feature selection.

For the Logistic Regression, we achieved an accuracy of 53.23% and 53.02% with and without volume respectively. The Random Forest achieved an accuracy of 53.54% and

53.28% with and without volume respectively (see figure 1). As one can see for most models, the incorporation of volumes makes no discernable difference. These accuracies are not out of the ordinary when comparing their performance to the results of other methods (see ¡cite Predicting price¿). Due to this and reduced amount of computational effort, we decided to continue with models that were trained without volume.

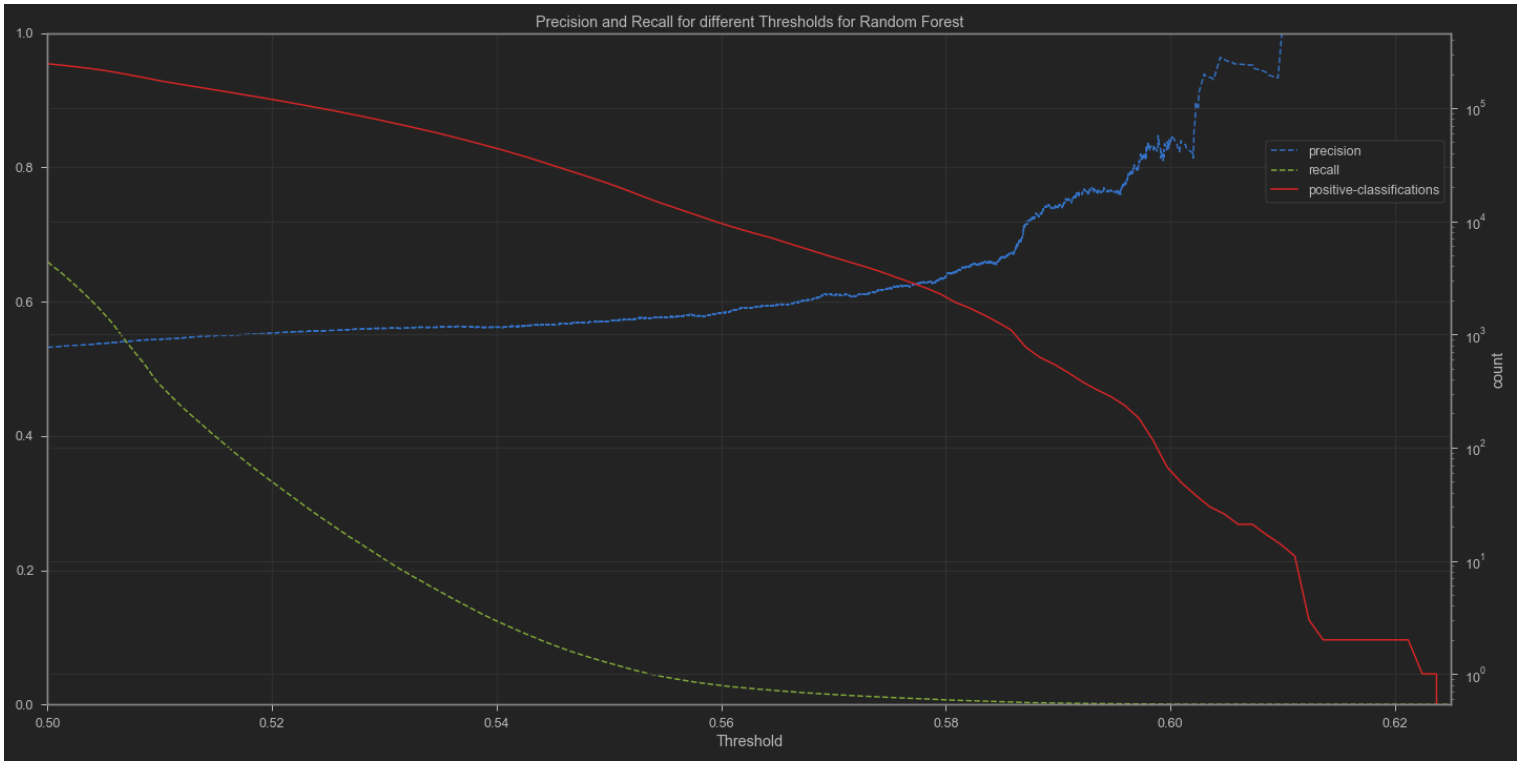

Figure 2: This figure illustrates the precision, recall and up-classifications of the Random Forest for different thresholds without taking volume into consideration.

Figure 2 illustrates the precision, recall and up-classifications of the Random Forest for different thresholds. The slope of the precision of remains relatively stable up until a threshold of 0.58 after which a strict increase takes place for increasing thresholds. On one hand, this comes with the tradeoff that drastically less up-classifications take place, which in turn leads to less trades for the trading algorithm, thus less potential return. On the other hand, strong probability signals indicate relatively high up-movements and low error probabilities, thus higher returns per trade and profitability after transaction cost. Therefore, it is paramount to fine tune the threshold in order to avoid costly missclassification. Only judging from the aforementioned figure, the highest returns will be achieved somewhere around a classification threshold of 0.6, since there are still a considerable amounts of trades in combination with high precision. In chapter 4.2, we demonstrate how changing thresholds leads to drastic differences in terms of return and profitability.

4.2 Strategy Performance

In this chapter, we outline the performance after employing the trading algorithm on the trading set as described in chapter 3.4. Figure 3 illustrates total returns using the Random Forest for probability signals when employing the trading algorithm for different classification threshold levels. For low probability thresholds, the returns are strongly

negative, since due to the low precision (see figure 2), a high amount of missclassifications take place. These missclassification then cancel off the positive returns in case of a correct classification.

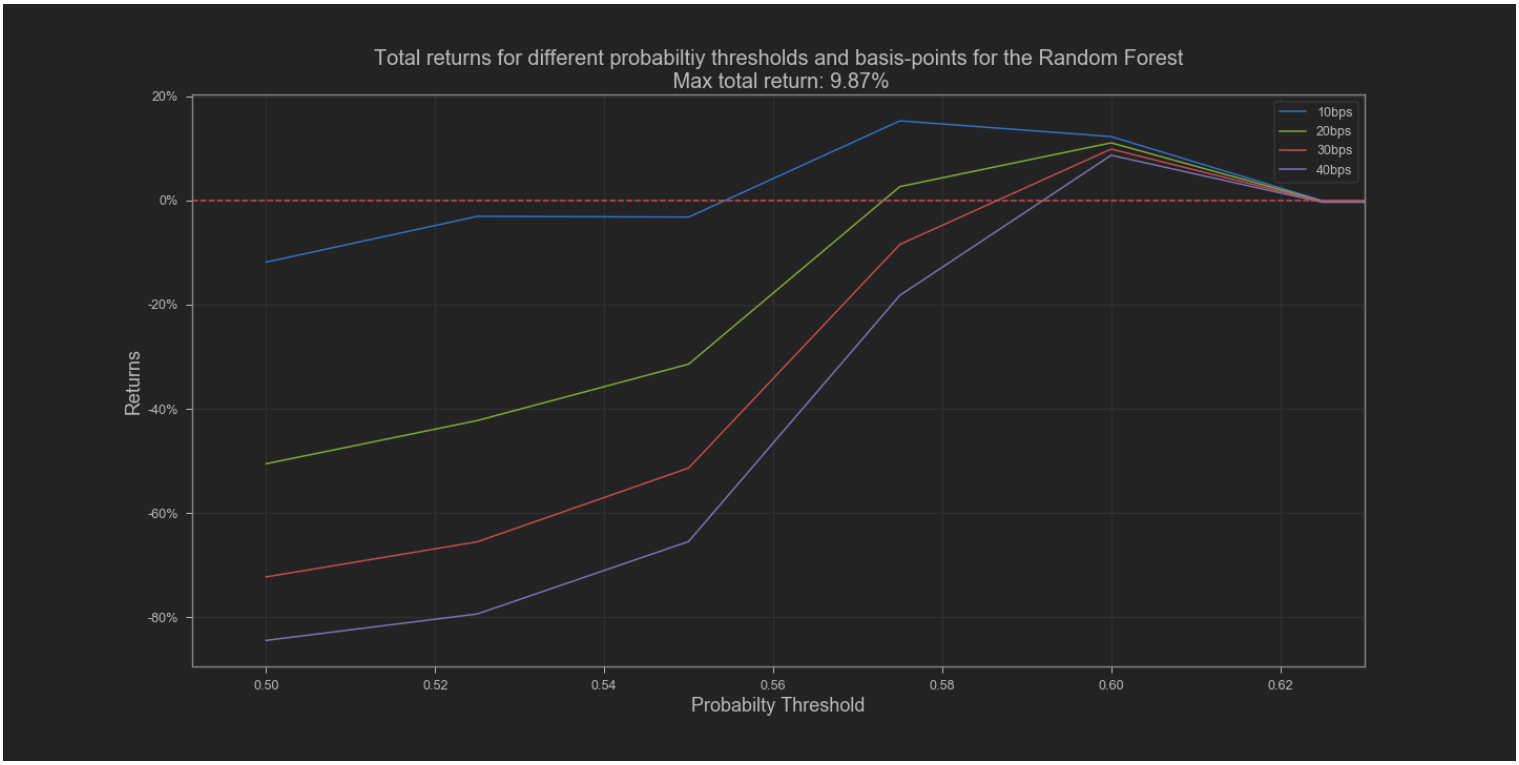

Figure 3: This figure illustrates total returns using the Random Forest for probability signals when employing the trading algorithm for different classification threshold levels. These returns are also further subjected to different levels of transaction cost (10, 20, 30 and 40 bps).

Further, because a weak probability signal is sufficient to surpass the threshold, the proportion of correct classification with insufficient returns is higher, as can be seen in figure 4. Even though the algorithm chose the correct position according to the price movement, the actual realized return can be below the transaction cost, which causes further decline in total return. Therefore, increasing the threshold also increases the proportion of classifications with sufficient return, since these observations with such returns tend to exhibit stronger probabilities.

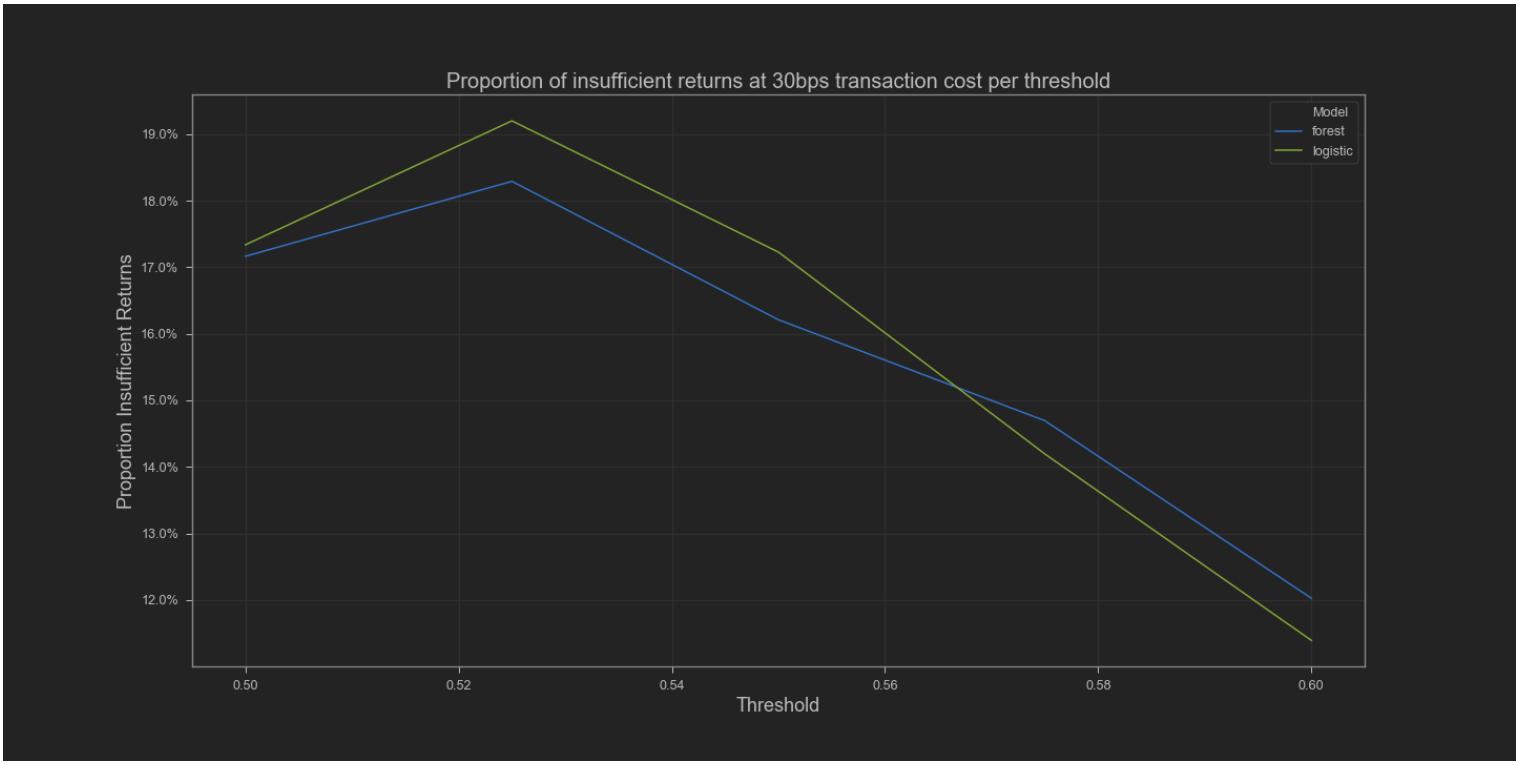

Figure 4: This figure illustrates the total proportion of correct trading decision according to the price movement with insufficient returns at 30bps transaction cost for different thresholds. These returns are insufficient, because they are below the 30bps which leads to effectively negative returns.

For increasing probability threshold, the returns also tend increase up until 9.87% at 30bps for the Random Forest, which can also be explained with the aforementioned implications from figure 2 and 3. Even when introducing different levels of transaction cost, the returns are consistently positive for a threshold of 0.6 converging to similar values. The convergence can be explained by the fact that considerably less trades take place for high threshold values as can be seen in figure 2. Since less trades are conducted, the implicit compound interest effect of transaction cost is drastically reduced, thus slight changes in costs do not affect overall profitability as much. Finally, after a certain threshold level, virtually no positive classifications take place according to figure 2, thus leading to zero returns eventually, which can also be seen for the Logistic Regression in figure 5.

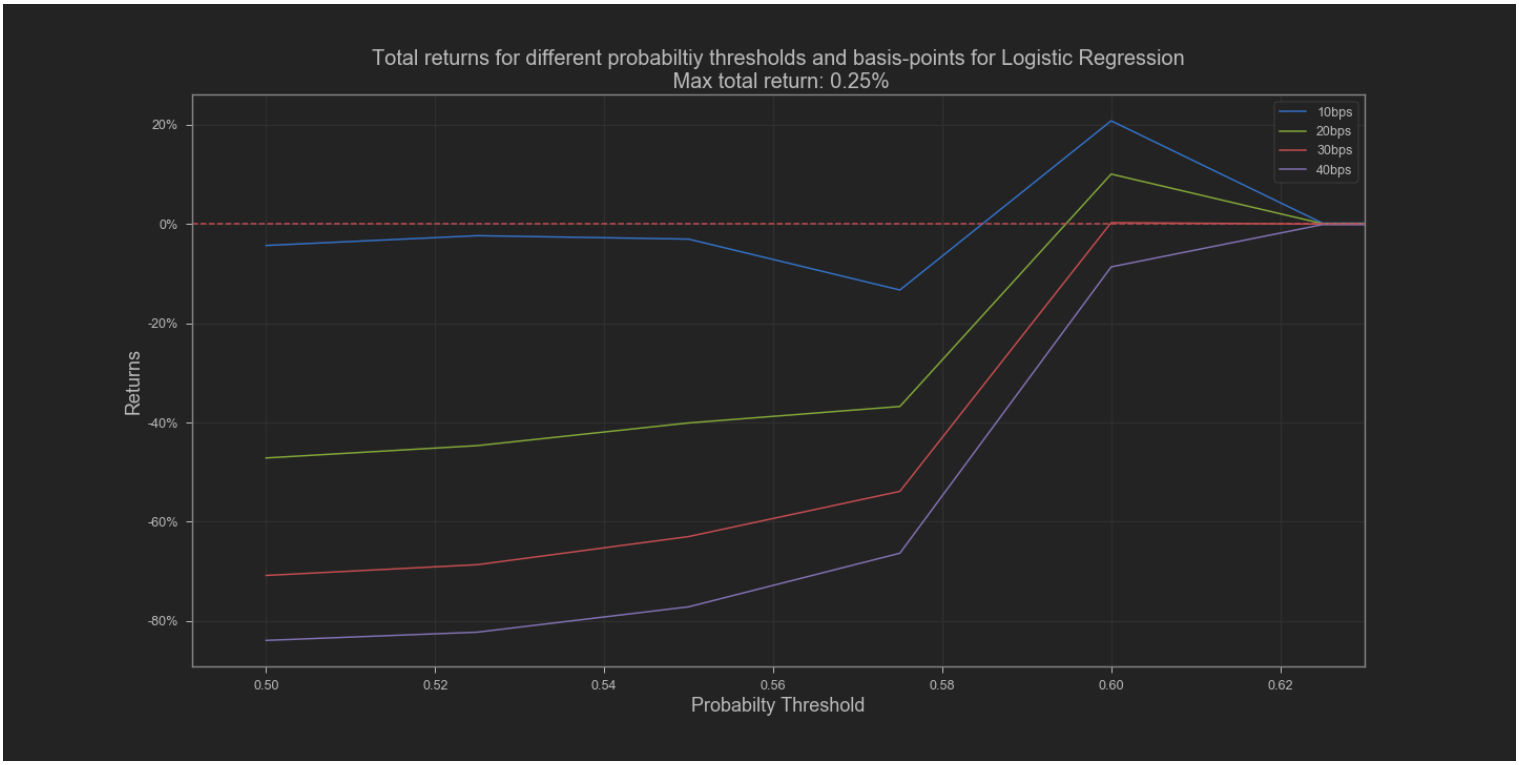

Figure 5: This figure illustrates total returns using the Logistic Regression for probability signals when employing the trading algorithm for different classification threshold levels. These returns are also further subjected to different levels of transaction cost (10, 20, 30 and 40 bps).

4.3 Further Analyses

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

5 Discussion

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.