Group 12 | MScFE 690 Capstone Course (C18-S3) | Final Project | 1 June 2020

Darko Šišak
darkosisak@gmail.com

An Event-driven Machine Learning Approach
in Designing Short-term Trading Models

Abstract

Trading models are complex systems composed of many interconnected and interacting events making it both difficult and time-consuming to derive mathematical quantities with classical statistical methods and standard time-series analysis. Applying machine learning in the design of short-term trading models is by now already a proven method. Still, the event-driven nature of financial markets is often overlooked. The goal of the project is to use machine learning in developing a robust short-term trading model based on parsimonious agent-based behaviour, accurately mirroring the dynamics of financial markets.

Keywords: Machine Learning, Event-Driven, Intrinsic time, Labeling, Meta-Labeling, Alpha Design

---

The complete code representing an integral part of this project can be found in the following GitHub repository:
https://github.com/darkosisak/WQU-Group12-Capstone

1. Introduction

Due to the complexity of trading systems, we often see trading models fail, mainly because quants and systematic traders run into common pitfalls on all levels of the trading model development workflow: data processing, classification, evaluation and epistemological. Today, the primary question is not how to design a profitable trading model, but how to deal with the resources required to design a profitable trading model.

Methods circumventing the trading systems' complexity, namely simple mathematical tools like econometrics and macroscopic alpha discovery techniques, fail to address the fact that time-series in the financial markets are typically non-IID. Consequently, such methods introduce enormous amounts of noise and fail to recognize the complexity of the data, resulting in useless theories due to oversimplification (López de Prado, 2018).

The goal of this project is to design an event-driven short-term trading model which balances between simplicity (of the workflow) and complexity (of systems). In other words – keeping the workflow simple while securing a high signal to noise ratio by maintaining the system's integrity.

In this project, a host of machine learning techniques described elsewhere in more detail by Halls-Moore (2015) and López de Prado (2018) will be deployed. However, to successfully address the aforementioned problems, it is of crucial importance to extend these techniques into local rules of interaction, based on the observation that macroscopic complexity is the result of simple interactions of "agents" at the micro-level. That way short-term patterns and behaviours seen in the complex system of real-world trading are preserved and can be precisely captured by machine learning techniques and translated into features and signals relevant for the development of an event-driven short-term trading model.

Therefore, in this paper the primary focus will be directed at the following three notions:

1. Event-drivenness and agent-based modelling;
2. Intrinsic time and data sampling methods;
3. Identification of most suitable classifier algorithms.

2. Theoretical Framework

This section will begin with a review and discussion of published work on event-based methodologies and agent-based modelling in the financial realm, followed by an analysis of different approaches to data sampling and data frequency and completed with the review of papers analysing machine learning algorithms.

2.1. Agent-based modelling and the notion of event-drivenness

In his study of complex systems, Glattfelder (2013) concluded that complexity on a higher level is the result of many simple interactions on a lower micro-level. According to Golub, Glattfelder and Olsen (2017), by focusing on local rules of interaction in a complex system the complex system can be represented as a network of agents, where structure determines function. Consequently, agent networks are the ideal formal representation of a system.

Such a perspective introduces a paradigm shift where mathematical analysis of complex systems is abandoned in favour of algorithmic models describing the interactions of agents through computation and simulation. Agent-based models (ABM) are capable of accurately replicating patterns and behaviour seen in financial markets.

In their *Agent-based models in financial market studies,* Wang et al (2018) pointed out how ABMs demonstrate the artificial intelligence of agents' behaviour in a bottom-up approach, as opposed to the top-down approach under the general equilibrium theory with its inherent mathematical inaccuracies.

Holland, Lebaron, Palmer and Tayler (1997) and Wolfram (1994) also observed that ABMs link the micro-actions of investors' behaviour with real price-action in the financial markets.

On the other hand, ABMs can be dauntingly challenging, especially in finding relevant parameters for capturing the unique behaviours of agents (Wang et al, 2018).

The concept of event-drivenness

Therefore, Golub, Glattfelder and Olsen (2017) proposed an event-based solution where every long or short position in the market can be understood as an agent, where p_i is the position comprised of a set of the $\{\overline{x_i}, \pm g_i\}$ entry price $\overline{x_i}$ and the position size and direction $\pm g_i$. Generally, this approach seems much more viable for our short-term trading model than a classical ABM method, although it provides only one level of events.

Halls-Moore (2015) extended this approach by dissecting a position into particular events, like market events, signal events, order events and fill events. The advantages of the event-driven trading engine implementation over a vectorised approach are especially evident in backtesting:

1. Code reusability – vectorised back-testers need all data to be available at once for analysis, while event-driven systems are used for both live trading and simulation testing;
2. The lack of lookahead bias because data is a set of events that must be acted upon. Therefore, event-driven back-testers are, as Halls-Moore puts it, "drip-fed" with realistic market data replicating the behaviour of order management and portfolio systems;
3. Realistic simulations – event-driven back-testers allow the handling of orders and consequently the exact simulation of order execution and transaction costs.

Halls-Moore's event-driven implementation addresses the position part of the trading engine, but it still lacks more sophisticated event labelling on the modelling level.

2.2. Data frequency and intrinsic time

In systematic trading, and in particular algorithmic trading, the frequency of data is one of the most important considerations. A general rule of thumb is the greater the number of data points the more trades and hence a more statistically robust analysis. Those data are typically organised either as tick-data or time-bars.

López de Prado (2018) recognises the need to avoid time-bars for two reasons:

1. Information is not processed at a constant time interval, in discrete-time. Time-bars, therefore, undersample information during high-activity periods and oversample at times of low activity;
2. As shown by Easley, López de Prado and O'Hara (2012), poor statistical properties, like serial correlation, heteroscedasticity, and non-normality of returns, are inherent to time-sampled series.

As an alternative, López de Prado suggests using one of the following bar construction methods:

- Tick bars, where bars will be formed after a predefined number of ticks (transactions) occurs, effectively solving the under/over-sampling issue seen with standard time-bars. However, outliers are a possible problem on exchanges that accumulate bids and offers in order books without matching them.
- Volume bars, which circumvent another issue seen with tick-bars, namely arbitrariness introduced from order fragmentation, by sampling after a predefined amount of asset's units have been exchanged.
- Dollar bars, as a means to sample bars in terms of the value of exchanged assets. This makes sense with significant price fluctuations and with equities since the number of outstanding shares often changes over the course of the analysis period.
- Information-driven bars, designed to capture market microstructural information, like tick and volume/dollar imbalances and tick and volume/dollar runs.

Müller et al (1993) approached the sampling problems of standard time-bars differently. By suspending time between events where physical time is substituted with interactions or events to make the system's clock tick, they introduced a novel time ontology: **intrinsic time**. Golub, Glattfelder and Olsen (2017) argue that this event-based approach forms the base for a self-referencing modelling framework, unrestrained by static building blocks and featuring a dynamic frame of reference. As a consequence, intrinsic time increases the signal to noise ratio in a time series by filtering out irrelevant information between relevant events.

These events dissect a time series $x(t)$ into a discrete set of events $\Omega[x(t), \delta]$ where δ is a given directional change threshold and the event dissecting the price curve.

López de Prado's bar construction methods are cutting edge, especially the information-driven bars and for forecasting models, but for a trading model, intrinsic time seems a better fit.

2.3. Machine learning algorithms in finance

López de Prado (2018) argues that since every empirical science must build theories based on observation, a researcher using classical statistical methods like linear regression will fail to observe the complexity of data, and as a result create overly simplistic and ultimately useless theories. Machine learning algorithms can learn patterns in a high-dimensional space, and once the features that are predictive of a phenomenon are identified by those algorithms, theoretical explanations based on realistic observations can be built and tested.

Halls-Moore (2015), also a proponent of implementing machine learning in finance, discusses several machine learning algorithms. He dismisses the **Naive Bayes** Classifier (specifically the Multinomial Naive Bayes, or MNB) for its inability to discern interactions between individual features which are not specified as extra features. He deems **Logistic regression** to be more useful because of its probabilistic interpretative nature and less concern about the correlation among features than in a Naive Bayes model.

Support Vector Machines (SVMs), according to Halls-Moore, possess a complicated fitting procedure but are nevertheless easy to work with. Types of data which are linearly separable work very well. However, a lot of data is not linearly separable so Halls-Moore proposes a modification to the kernel used by the SVM as a solution, which would allow non-linear decision boundaries. Still, significant disadvantages remain in their computational and tuning complexity and their fitted model's interpretation difficultness.

Also, López de Prado states that SVMs do not scale well with sample size. He proposes the building of a bagging algorithm for scalability of poorly scalable algorithms like SVM.

López de Prado (2018) and Halls-Moore (2015) agree that the Decision Tree's proneness to overfitting is solved using a random forest (RF). Halls-Moore points out that random forests are some of the best classifiers used in machine learning competitions and should always be considered. López de Prado draws similarities between RF and bagging like training individual estimates over bootstrapped data subsets independently. However, there is a key difference to bagging: RFs feature a second level of randomness by evaluating only a random subsample when optimising each node split. This action furtherly decorrelates the estimators.

Moreover, López de Prado (2018) observes the reduction of RF forecasts' variance without overfitting, as well as the RF's ability to evaluate feature importance. Pereda, Santos, and Galán (2015) recognised, citing Breiman (2001), that RFs as "an ensemble learning method that employs trees as weak learners, has become one of the most popular and widely used techniques in many scientific disciplines". Besides their solid predictive performance, random forests provide a natural way of gauging the importance of individual predictors (Criminisi et al. 2011).

A potential RF's disadvantage though, if left unaddressed, is that with a large number of non-IID samples overfitting still takes place. Decision trees are notorious for building essentially identical trees when sampling randomly with replacement, resulting in each decision tree being overfitted. López de Prado (2018), however, provides a fix in *sklearn* for that problem.

3. Methodology

As it is evident from the works described in the previous section, agent-based models are very capable of accurately replicating patterns and behaviour seen in financial markets, but are difficult to work with on many levels. On the other hand, the proposed event-driven solutions by Golub, Glattfelder and Olsen and Halls-Moore fail to completely capture events.

To fill this gap between complete but difficult-to-implement coverage of agents and relatively easy but incomplete alternative event-driven solutions, the rationale presented in this paper is to identify each agent by a proxy, extending López de Prado's work through the introduction of multi-meta signalling, e.g. multi-labelling on the signal generation level.

Python with Pandas, NumPy, MLFinLab and Sklearn libraries are used for the model development and for extending select techniques and principles found in the book "*Advances in Financial Machine Learning*" by López de Prado into an event-driven approach.

Upon setting up the basic data-frame, in-sample model fitting is performed. This includes the fitting of the first model iteration with signalling, preparation of features for the meta-model, labelling and in-sample training and testing. This is followed by the selection of the most appropriate algorithms with meta-model fitting. Finally, out-of-sample performance testing including validation of the model against a more realistic market environment by taking into account slippage and commission completes the workflow.

3.1. Data selection and preparation

In setting the basic data-frame, the focus is on the GBP/USD foreign exchange (FX) instrument. The data is sourced from Dukascopy Bank with unlimited access to various forms of high-quality data, including raw tick-data. The liquidity and the property of long/short symmetry inherent to currency pairs as instruments make the FX market the ideal environment for the quantitative development of automated trading systems. Moreover, directional symmetry makes currencies extremely difficult to trade profitably. Therefore, in theory, a trading model profitable in the foreign exchange market should also be applicable to other markets.

The GBP/USD, or colloquially called the Cable, is one of the three most liquid majors on the FX market (Ganti, 2019). Furthermore, Cable's historical "responsiveness" to fundamentals like the Great Recession, or the more recent Brexit vote, makes it an ideal instrument to encompass regime shifts while providing a sufficient number of signals for analysis.

Consequently, in-sample analysis is conducted on data from the time period from start to end of the year 2019 while out-of-sample model simulation spans over the first four months of 2020. Given the COVID-19 pandemic circumstances and its profound effects on virtually every single financial market, this approach should also provide for a realistic stress-test.

Sampling frequency and bar construction

To ensure a sufficiently-high sampling resolution for a short-term trading model, data is organised into time-series with approximately 300 observations daily. The sampling method will be chosen upon a comparison of serial correlation and normality properties of different bar construction methods, namely time bars, tick bars, volume bars and dollar bars. To answer what bar method has the lowest serial correlation, autocorrelation tests for all bar types are performed; normality for the different distributions will be evaluated by the Jarque–Bera test with the *stats.jarque_bera* function from the *scipy* package.

3.2. Multi-labelling

The principle of multi-labelling is the concept presented in this paper of capturing market events organically, exactly as they occur, to implement event-drivenness in machine learning methods. There are two methods investigated:

1. An extended version of López de Prado's meta-labelling technique, basically applying meta-labelling on more than one level;
2. A simpler multi-levelled approach during the signal generation phase, which can be described by the following pseudo-code example of a signal:

```
if close.shift(2) <= MA && close.shift(1) > close.shift(1)
```

Each method implements labelling on the event level in simulated real-time, hence shifting the system's clock effectively from sampled time, ticks or volume to intrinsic time determined by sampled market events through one or more of the previously mentioned signals.

In this vein, data is sampled for labelling from microstructure events translated into signals and the following signals are created:

Table 1. *Signals used during the trade-model design*

Name	**Description**	**Code**
Retracement Bounce	Signalling long or short after trend-positive momentum during price retracements	*rbo*
Micro-breakout	Signalling long or short when the price exceeds the previous high or falls below the previous low respectively	*mbo*
ATR Direction	Signalling long or short when the price crosses a dynamic support/resistance line determined by a multiple of Welles Wilder's Average True Range	*atr_dir*
BB Sides	Mean reverting long/short signals at Bollinger Bands price break-outs	*BB_sides*

Here the Retracement Bounce (*rbo*) signal implements the multi-levelled approach by capturing events where mean reversion occurs (price retracement) followed by another micro-event (the trend-positive price momentum).

For the creation of the primary model, the Triple-barrier method is deployed, as described by López de Prado (2018, 2020), where stop-loss, take-profit and time-horizon barriers are set. Instead of the original static standard deviation mean, the Welles Wilder's Average True Range is used to set the widths of the stop-loss and take-profit barriers dynamically, according to intra-day observed price volatility. The Triple-barrier method is "fed" with one of the event-based signals from Table 1.

Since a fairly high number of signals is expected, micro-trends to filter out false positives are captured. For that purpose the *DecisiveML* package is deployed, implementing the Trend-scanning method (López de Prado, 2020) on one of the event-based signals from Table 1 not processed by the Triple-barrier method.

Snippet 1: Getting labels from Trend-scanning with DecisiveML

```
import decisiveml as dml
# ----------------------------------------
trend = dml.getBinsFromTrend(
   molecule=r["rbo"].dropna().index,
   close=r.Close,
   span=[22, 44, 11],)
trend
```

The Trend-scanning method assigns to every observation x_t a label $y_t \in \{-1, 0, 1\}$based on whether x_t is on a downtrend, no-trend or uptrend respectively, where $\{x_t\}_{t=1,\ldots,T}$. The method computes values $\hat{t}_{\hat{\beta}_1}$ associated with the estimated regressor coefficient in a linear time-trend model,

$$x_{t+l} = \beta_0 + \beta_1 l + \varepsilon_{t+l}$$

$$\hat{t}_{\hat{\beta}_1} = \frac{\hat{\beta}_1}{\hat{\sigma}_{\hat{\beta}_1}},$$

where $\hat{\sigma}$ is the standard error of the estimated regressor coefficient $\hat{\beta}_1$, and $l = 0, \dots, L-1$ with L setting the look-ahead period.

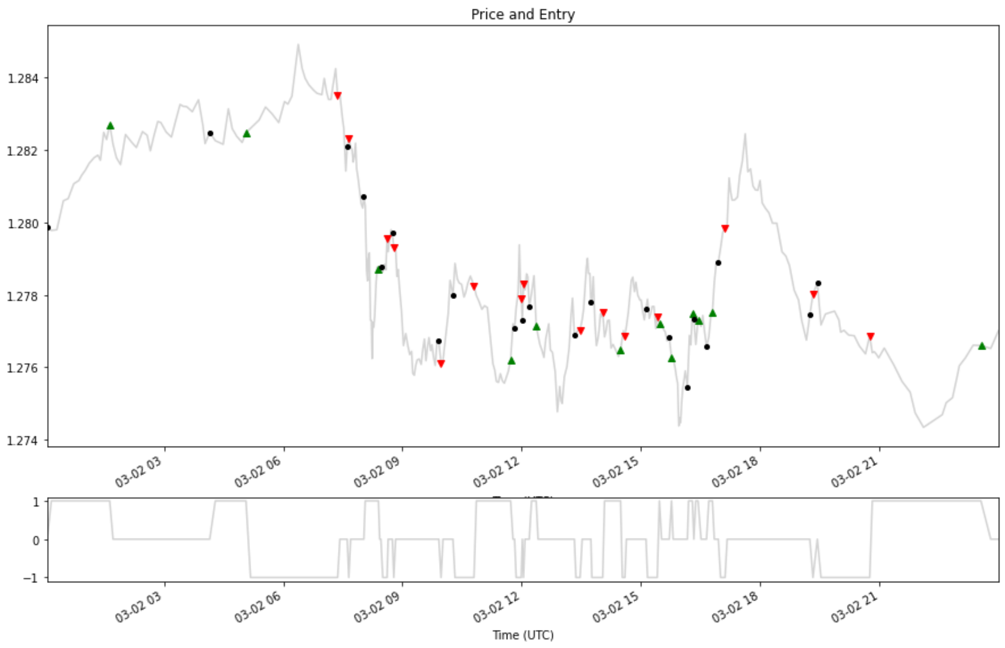


Figure 1: Trend Scanning labels applied to the GBP/USD time-series

Features for the final meta-model

Finally, another meta-label layer is added to improve the model's F1- score, i.e. its harmonic average of precision and recall, by running a set of features through a classifier algorithm. Those features should capture a host of events and metrics, but without correlation. Therefore a Principal Component Analysis is performed to check for correlation between the features along with a PCA transformation to decorrelate the features if needed

The set of features used in this project includes technical indicators, micro-events and statistical metrics. PCA and feature importance analysis are used to filter out correlated and weak features respectively to prevent overfitting in the final modelling phase.

3.3. ML-algorithm selection and optimisation

The machine-learning algorithms taken into consideration in this project are either a Support Vector Classifier (SVC), a logistic regression (LR) classifier or random forest (RF) with a *sklearn* fix.

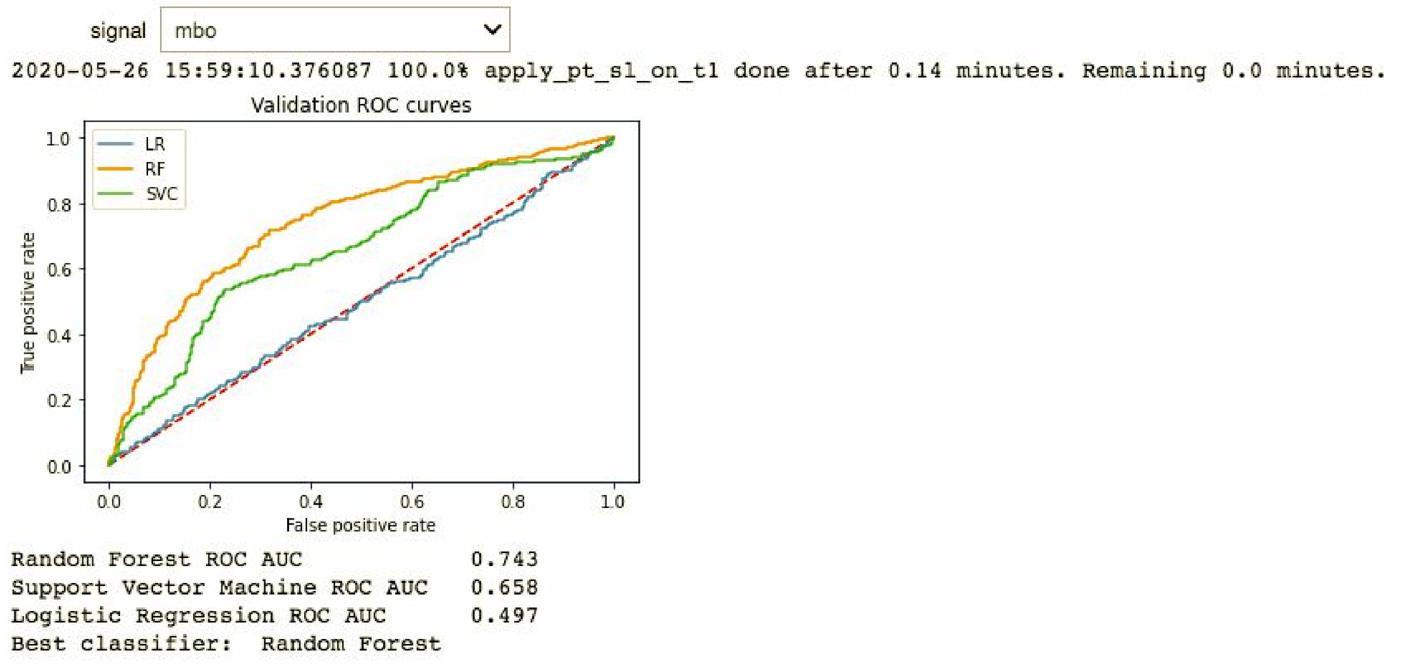
Since it has recently been published that sometimes simple algorithms can be surprisingly efficient (Malato, 2020) and one of the goals of this project is to keep the resource footprint of the modelling process as small as possible, accuracy and speed of the three algorithms are validated. This is done by performing ROC-AUC for each algorithm and comparing the returned values with execution speed, after which the algorithm with the best ROC-AUC to speed ratio for the final meta-labelling step is chosen.

This in effect means that there are three working sets: a training set, a validation set (for choosing the algorithm) and separate testing set to prevent test set overfitting; once the modelling on the first two sets is concluded the performance backtest on the third set is run only once, without going back to tweak parameters.

To facilitate the whole process, including the validation of different multi-labelling signals, all of the machine learning and the classifier metrics are deployed in the *TBL_ML* function, which is then called by an interactive drop-down menu in an *ipywidgets* implementation. This enables the deployment of different signals and finding the best classifier/signal combination without the need to rerun the code.

Snippet 2: Widget implementation for the *TBL_ML* function

```
import ipywidgets as widgets
from ipywidgets import interact, interact_manual, interactive
from IPython.display import display
# ----------------------------------------
w = interactive(TBL_ML, signal = ['mbo','rbo','atr_dir','BB_sides','MACD_mom','over','dir'])
display(w)
```



```
2020-05-26 15:59:10.376087 100.0% apply_pt_sl_on_tl done after 0.14 minutes. Remaining 0.0 minutes.
```

```
Random Forest ROC AUC              0.743
Support Vector Machine ROC AUC     0.658
Logistic Regression ROC AUC        0.497
Best classifier:  Random Forest
```

Figure 2: The widget's interface with the first part of the output

4. Results and outcomes

4.1. Data structure and event sampling

4.1.1. Calculation of serial correlations for the bar types

To find what bar method has the lowest serial correlation, each of the bar types underwent autocorrelation tests, returning the following results:

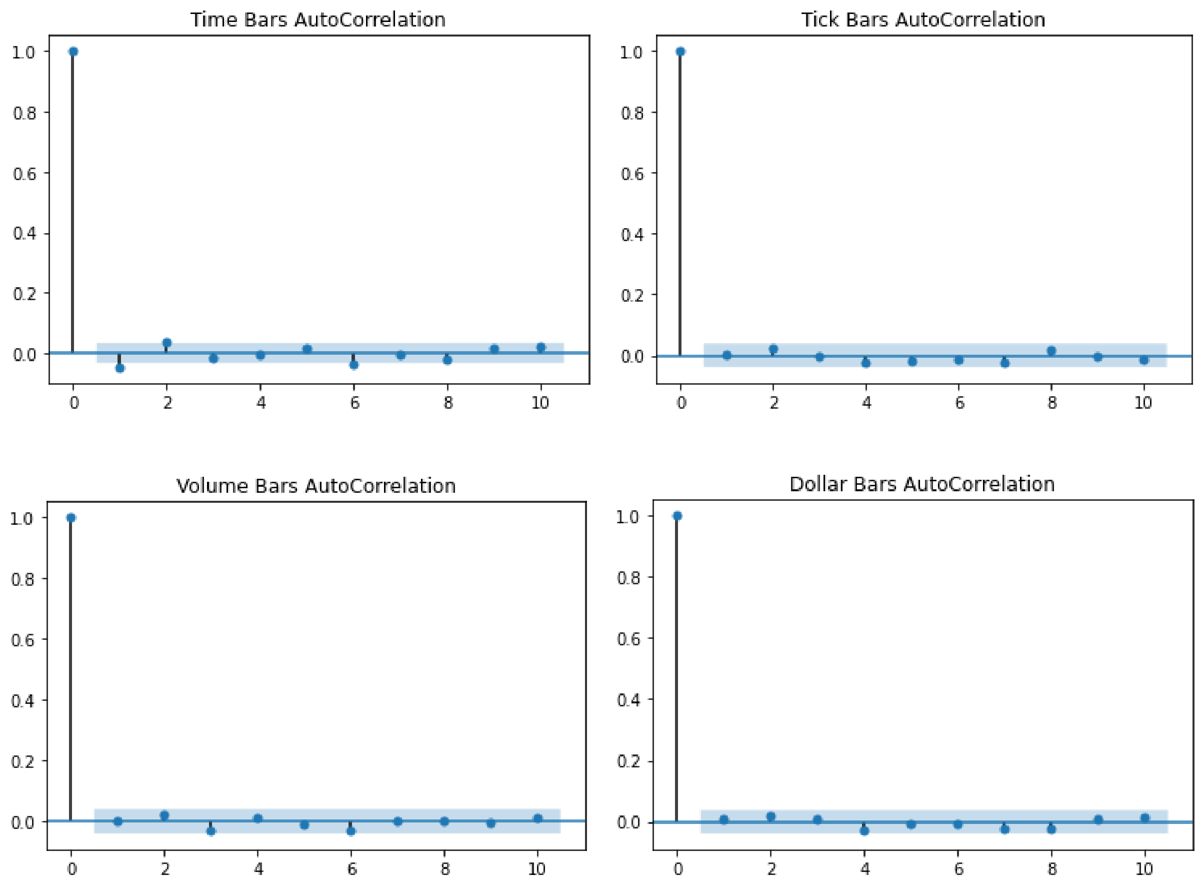


Figure 3: Autocorrelation charts for different types of bars

All bar methods exhibit autocorrelation levels near zero, which is clear evidence against autocorrelation. However, the time bar method's first, second and sixth lags protrude slightly outside significance levels.

4.1.2. Jarque-Bera normality tests for the bar types

With the Jarque–Bera test the bar types' distributions are tested for normality. The test statistics are always positive numbers; normality numbers far from zero signal that the data do not have a normal distribution.

Table 1: *Jarque-Bera test results*

	normality	**χ^2**
time bars	4011	0.0
tick bars	65	8.4×10^{-15}
volume bars	125	0.0
dollar bars	71	5.6×10^{-16}

All bar methods have a *Chi*-squared value smaller than p = 0.05. Consequently, the null hypothesis is rejected in every method, meaning that all bar methods' returns have a non-normal distribution. The tick bars method achieves the lowest normality test value.

4.1.3. Plotting of standardised distributions

It is already shown by the Jarque-Bera test that none of the bar methods achieves normally distributed returns. Since many statistics inferences require if not a normal, then at least a nearly normal distribution, it is interesting to see if the non-time clocked methods achieve distributions nearer to normal than time bars.

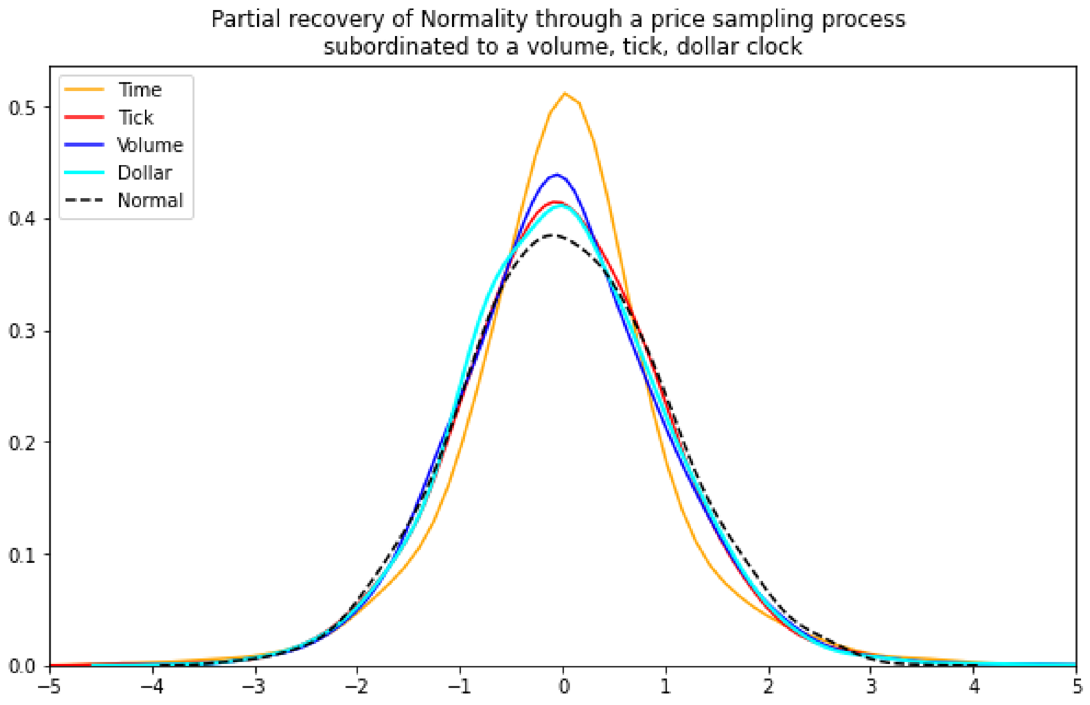


Figure 4: Distributions of different types of bars compared to the normal distribution

By looking at the distributions of the bar methods plotted on the chart above, it is evident that the time bars have a higher excess kurtosis and a moderate right skewness. When comparing the distributions on the same chart it is obvious that subordination of the price sampling process to non-time clocks—namely tick, volume and dollar clocks—partially recovers normality. The tick bar's distribution is closest to normal, with low excess kurtosis and virtually without skewness.

4.1.4. Initial sampling

Therefore the most appropriate bar construction method is the tick-bar method with a sampling frequency of 500 ticks, which provided us with the following number of observations in the time-series for the period from 2 Jan 2019 to 30 Apr 2020.

Table 2: *Number of observations in the time-series*

	Observations
Total	87026
Training set	48955
Validation set	12238
Performance test set	25832

4.1.5. Sampling events for intrinsic time

The *rbo* (Retracement Bounce) signal was chosen for creating 25,447 intrinsic time-based observations in total and processes with the trend scanning method into labels using the *getBinsFromTrend* function.

4.2. Features analysis and algorithm evaluation

Since the features have been carefully chosen before deployment, the Principal Component Analysis showed a very low correlation between the features, as shown in the left, raw data heatmap in Figure 5.

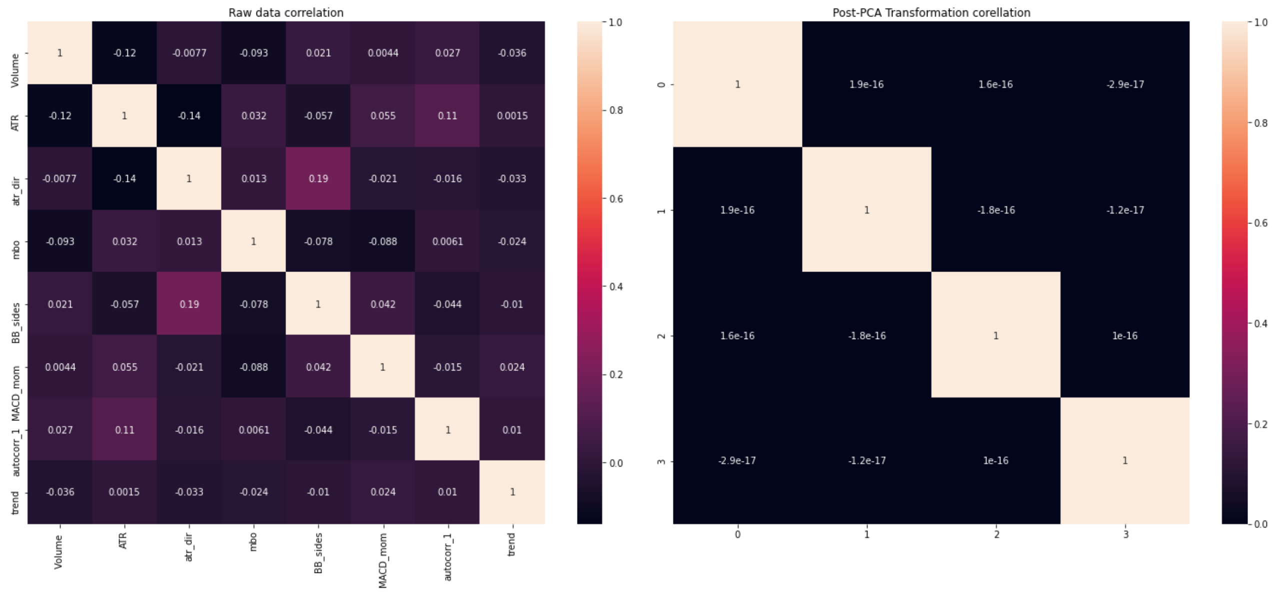


Figure 5: PCA heatmaps for features

Therefore, the original set of features for the evaluation of the three machine-learning algorithms was used. Otherwise (the raw PCA heatmap showing highly correlated features), the transformed data should have been taken instead.

As shown in Figure 6, upon a comparison of three algorithm metrics between the training and validation sets, the Random Forest algorithm returned the best ROC values.

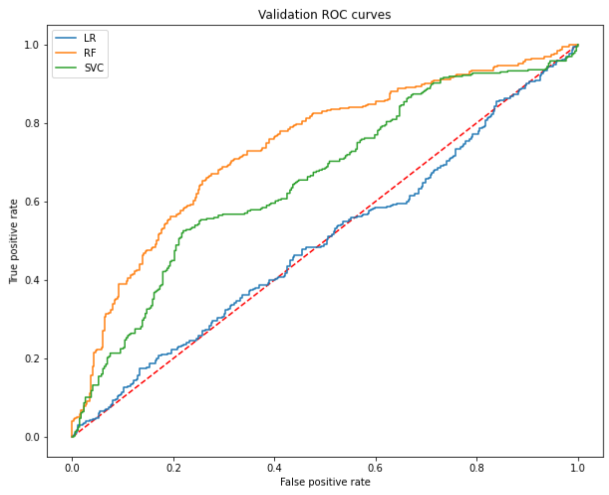


Figure 6: ROC curves from analyzed classifiers

The exact ROC values returned from the validation sets are as follows:

Random Forest ROC AUC:	0.738
Support Vector Machine ROC AUC:	0.658
Logistic Regression ROC AUC:	0.497

Consequently, the *TBL_ML* function selected the RF algorithm. Since on separate test-runs the RF algorithm required only negligible more time for data-processing than the LR classifier, and virtually the same as SVC, the choice is clear.

By comparing the training and validation ROC metrics from the plot in Figure 7, it can be seen if it is indicative of overfittedness of the training set.

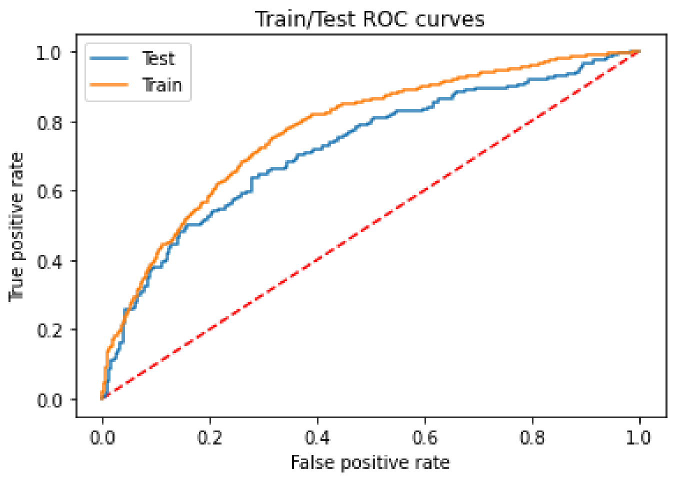


Figure 7: ROC curves training and validation sets

The ROC_AUC values are 0.776 and 0.738 for the training and validation sets respectively, which does not indicate overfittedness.

4.3. Approach validation

To empirically validate the core elements of the approach presented in this paper, the ROC properties of two modified copies of our original model were measured—the first without the *mbo* event, effectively removing micro-event-baseness from the model, and the other one without the *trend* feature, dropping intrinsic time as a consequence.

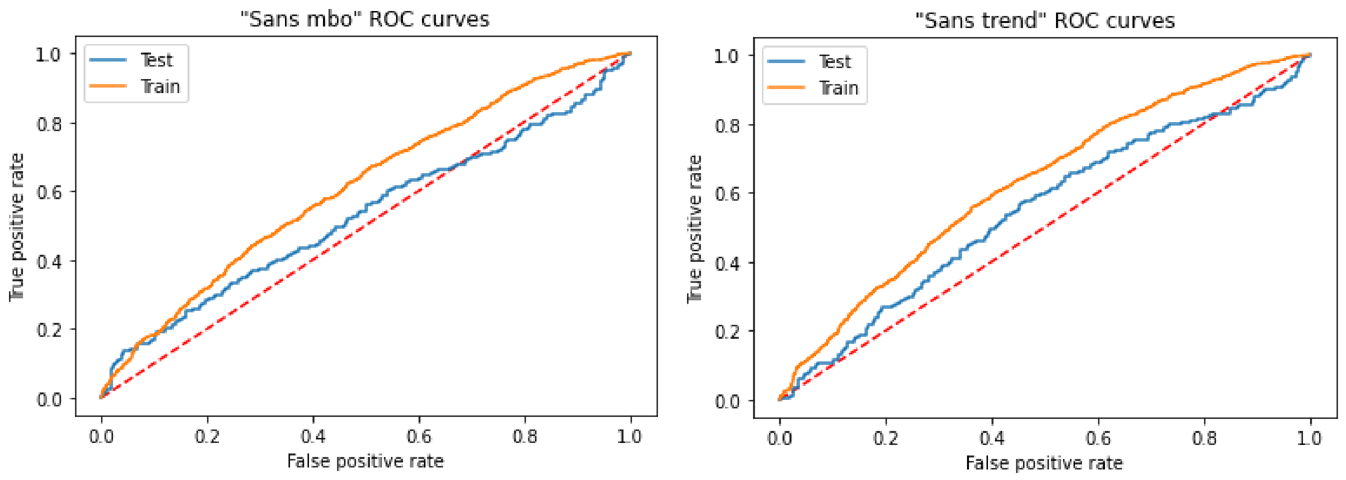


Figure 8: ROC curves of models without 'mbo' and 'trend'

Table 3: *ROC_AUC metrics and accuracies for the original and ordinary models*

	Original model	**Model sans *mbo***	**Model sans *trend***
Train ROC AUC	**0.776**	0.611	0.629
Validation ROC AUC	**0.738**	0.530	0.548
Δ ROC_AUC	**0.038**	0.081	0.081
Test Accuracy	**0.692**	0.519	0.537

Best values in the table above are in bold.

4.3. Model performance and trading simulation

The out-of-sample performance testing of the model was performed over the period from 3 Jan 2020 to 30 Apr 2020, encompassing 25.832 observations and returning 1721 executed round-turns of trades.

To validate the meta-label concept, it was compared to a performance test of a model version without meta-labelling.

Consequently, two different sets of performance statistics were generated using the *show_perf_stats* (*create_full_tear_sheet*) function from the *pyfolio* package, shown in a side-by-side comparison in the following table:

Table 4: *GBP/USD performance statistics comparison*

Out-of-sample trading performance		
From–to date	2020-01-03–2020-04-30	
Total months	4	
	no meta-labelling	**with meta-labelling**
Annual return	-12.4%	**783.6%**
Cumulative returns	-4.4%	**108.5%**
Annual volatility	19.3%	**22.3%**
Sharpe ratio	-0.59	**9.94**
Calmar ratio	-0.89	**575.61**
Stability	0.01	**0.87**
Max drawdown	-13.9%	**-1.4%**
Omega ratio	0.89	**12.89**
Sortino ratio	-0.89	**57.42**
Skew	0.72	**1.92**
Kurtosis	2.65	**3.88**
Tail ratio	1.47	**6.53**
Daily value at risk	-2.5%	**-1.9%**
Alpha	-0.08	**8.77**
Beta	0.00	**0.00**

Table 5: *Drawdown periods for the complete model (with meta-labelling)*

Worst dd periods	Net drawdown in %	Peak date	Valley date	Recovery date	Duration
0	1.36	2020-01-29	2020-01-30	2020-02-05	6
1	1.05	2020-04-16	2020-04-17	2020-04-21	4
2	0.77	2020-04-08	2020-04-09	2020-04-13	4
3	0.70	2020-01-21	2020-01-22	2020-01-24	4
4	0.58	2020-04-23	2020-04-24	2020-04-27	3

Table 6: *Stress events for the complete model*

Stress Events	mean	min	max
New Normal	0.88%	-1.36%	5.92%

The complete performance tear sheets are provided in Appendices 1 and 2.

5. Discussion

During the development of the short-term trading model based on the event-driven machine learning principles presented in this paper, it perspired that the random forest classifier is most suitable due to its overall good performance, the built-in feature importance functionality and because train set overfitting can be controlled by applying various techniques. By adjusting relevant *RandomForestClassifier* parameters, features with importance below 5% were filtered out against overfitting and a *balance_subsample* method was applied to deal with unbalanced classes. Furthermore, parsimony was enforced by keeping the depth of the decision tree rather small in tandem with a high number of estimators.

From the ROC_AUC metrics, it is evident that feature selection also plays an important role in the accuracy of the model: the project has empirically proven that the RF algorithm performs significantly worse when any of the core elements of the original approach is missing. With intrinsic time (the *rbo* signal contained in the *trend* feature) removed, metrics deteriorate to barely acceptable levels, and without the event-based *mbo* feature component even further. Rather unsurprisingly, since the feature importance analysis pointed those two out to be the most important features.

The out-of-sample trading performance data confirms the success of event-based meta-labelling in filtering out false positives. While both the complete and primary models outperform the S&P 500 benchmark significantly, the former shows much stronger robustness to market regime shifts, as evident from the performance charts provided in the Appendices 1 and 2.

The complete model also withstands the reality-check in regards to commission and spread. The following table shows the performance metrics from returns adjusted to include an average spread of 0.8 pips and a commission of 6 USD per round-turn per standard lot (100K) traded.

Table 7: *Adjusted performance metrics*

Annual return	512.4%	**Omega ratio**	8.71
Cumulative returns	84.3%	**Sortino ratio**	41.35
Annual volatility	21.2%	**Skew**	1.89
Sharpe ratio	8.69	**Kurtosis**	3.93
Calmar ratio	358.18	**Tail ratio**	5.12
Stability	0.85	**Daily value at risk**	-1.9%
Max drawdown	-1.4%	**Alpha**	5.69

Of course, a more detailed dynamic liquidity analysis should be performed to gauge slippage and execution risks before trading larger volumes (above 20 standard lots), but this is beyond the scope of this project.

6. Conclusion

As mentioned in the introduction, the complex nature of trading systems makes it very difficult and time-consuming to derive mathematical quantities with classical statistical methods and standard time-series analysis.

The currently prevalent paradigm of simplifying complex systems into conformity to traditional econometric methods comes with the consequence of losing crucial information in the process, while classical event-driven approaches like agent-based modelling can be complicated and often require large resources.

This project managed to deliver a robust short-term trading modelling process with a small resource footprint, based on the principles and methods presented in this paper. This should eliminate many of the pitfalls traders and quants usually encounter while working on trading models. Also, the described model development workflow with code could, due to its comparably modest maintenance requirements, serve as a guideline to the professional retail trader and the large institutional trading firm alike, keeping running costs down.

This paper also provides strong evidence that event-driven signals coupled with intrinsic time work better than conventional, one-dimensional signals.

Since to date nothing similar has been found in literature capable of capturing the dynamics of real-life systematic trading action while being “machine learning ready”, the work with the subsequent trading model can be seen as a unique solution for getting the typical trading model-developing and maintaining job done quickly and in a parsimonious environment with reasonable human and computational resource requirements.

References

Arthur, W. B., Holland, J. H., LeBaron, B., Palmer, R. G. and Tayler, P., 1997. *The Economy as an Evolving Complex System*. MA: Addison-Wesley Reading. pp 15-44

Criminisi, A., Shotton, J., Konukoglu, E., 2011. *Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning*. Foundations and Trends in Computer Graphics and Computer Vision 7 (2-3), 81-227. DOI: 10.1561/0600000035

Glattfelder, J. B., 2013. *Decoding Complexity*. Springer, Berlin.

Golub, A., Glattfelder, J. B., and Olsen, R. B., 2017. *The Alpha Engine: Designing an Automated Trading Algorithm,* SSRN

Ganti, A., 2019. *Foreign Exchange Market.* [ONLINE] Available at: https://www.investopedia.com/terms/forex/f/foreign-exchange-markets.asp. [Accessed 20 April 2020]

Halls-Moore, M.J. 2015, *Successful Algorithmic Trading*. QuarkGluon Ltd

López de Prado, M., 2018. *Advances in Financial Machine Learning*. John Wiley & Sons. Hoboken, NJ

López de Prado, M., 2020. *Machine Learning for Asset Managers*. Cambridge University Press. Cambridge, UK. DOI: 10.1017/9781108883658

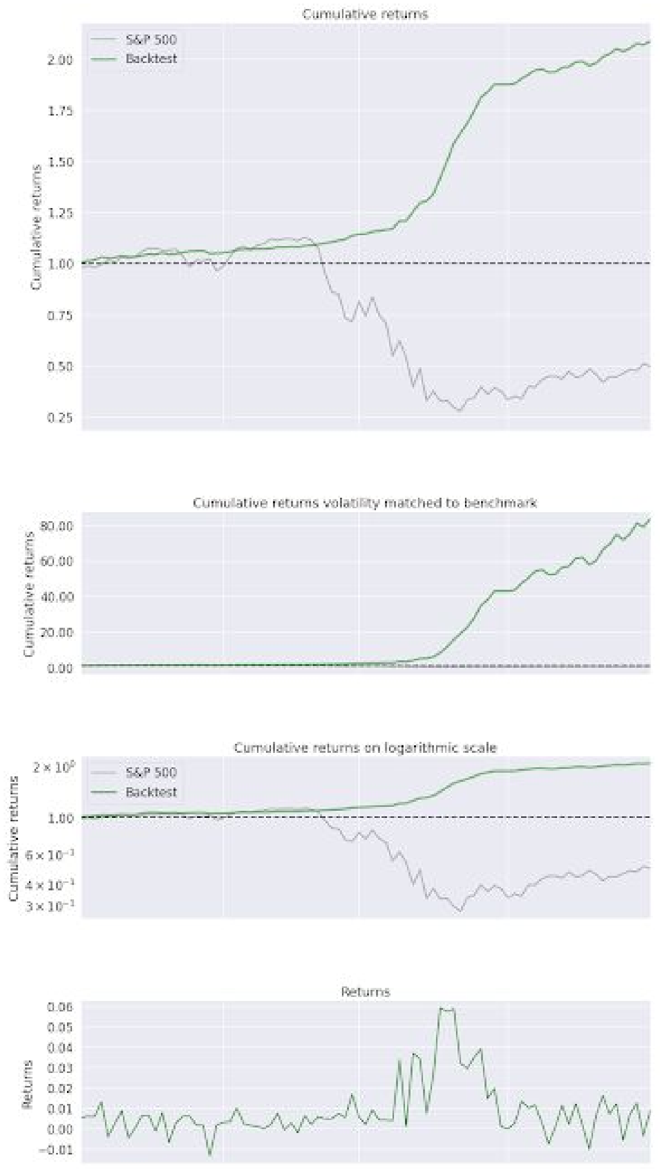
Malato, G., 2020. *Machine learning for stock prediction. A quantitative approach.* [ONLINE] Available at: https://towardsdatascience.com/machine-learning-for-stock-prediction-a-quantitative-approach-4ca98c0bfb8c. [Accessed 20 April 2020]

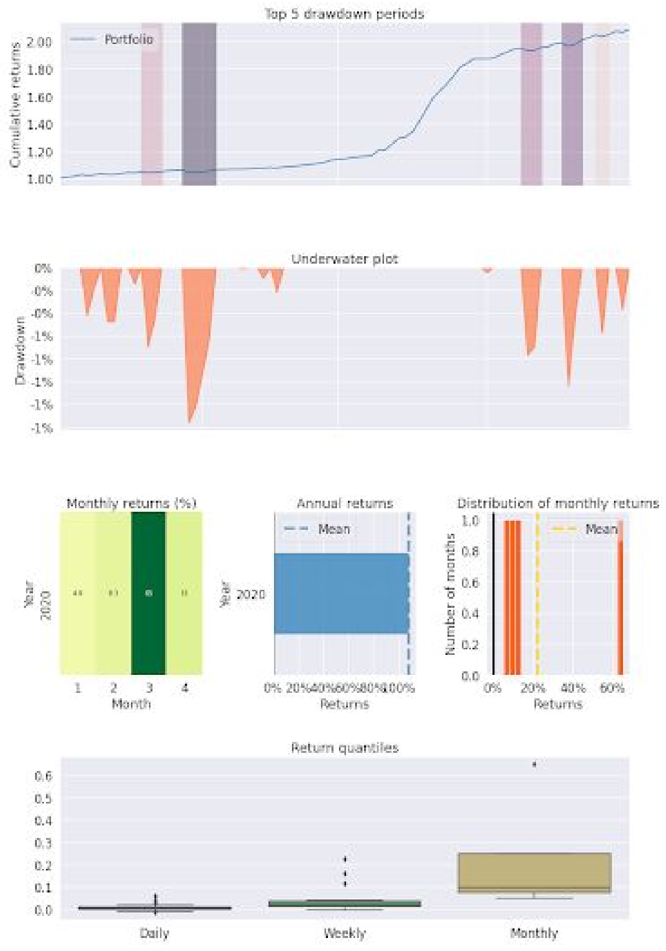
Müller, U. A., Dacorogna, M. M., Davé, R. D., Pictet, O. V., Olsen, R. B., and Ward, J. R., 1993. *Fractals and intrinsic time: A challenge to econometricians*. In: presentation at the XXXIX International AEA Conference on Real-Time Econometrics, 14-15 Oct 1993 in Luxembourg

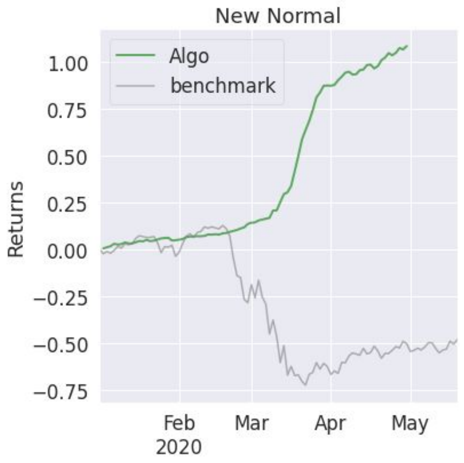
Wang, et al, 2018. *Agent-based models in financial market studies*. J. Phys.: Conf. Ser. 1039 012022

Wolfram, S., 1994. *Cellular Automata and Complexity: Collected Papers vol 1.* MA: Addison-Wesley Reading

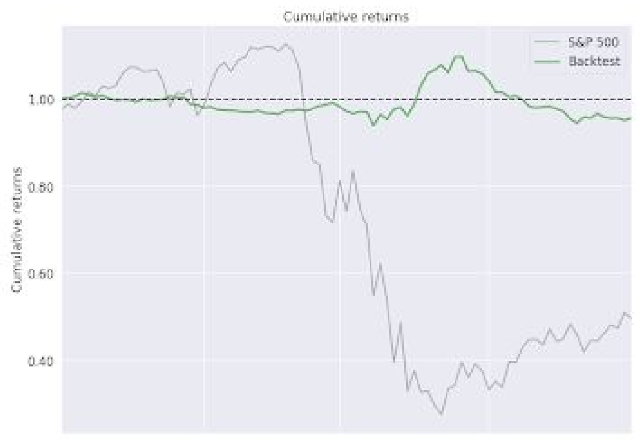
Appendix 1. Performance of the complete model

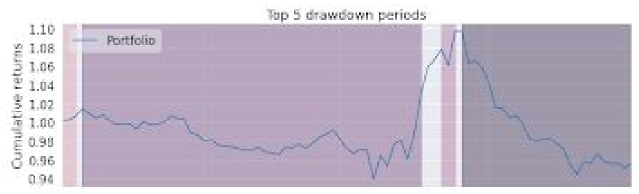


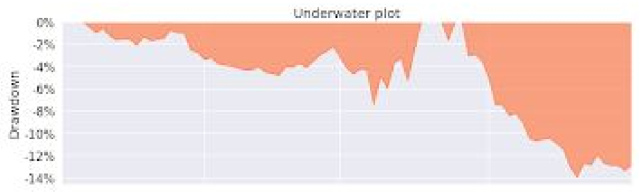


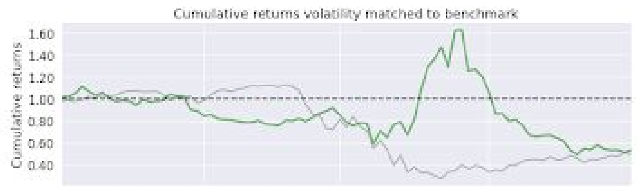


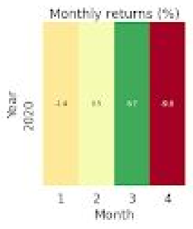
Appendix 2. Performance of the primary model (sans meta-labelling)

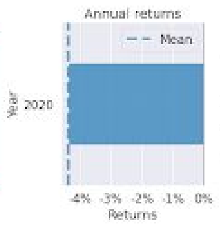


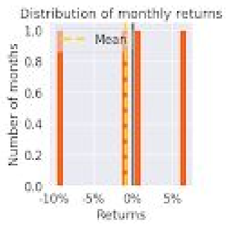


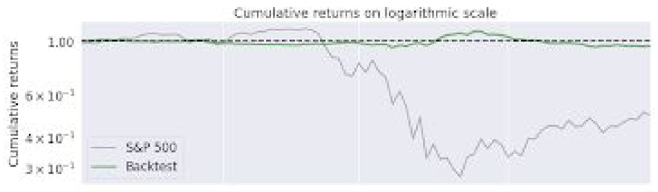


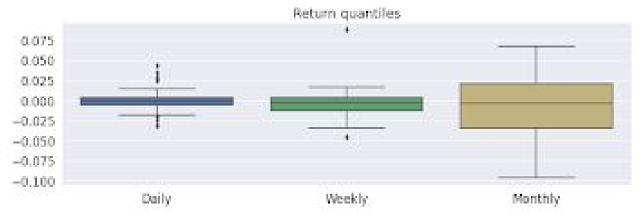


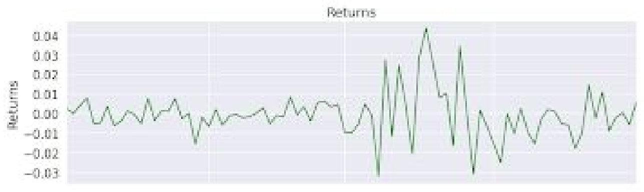


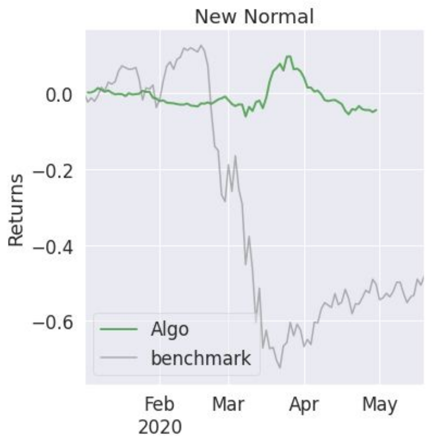












Appendix 3. Working plan, final report

The following time table, shown in Table 1, presents a report of the working plan, including anticipated potential obstacles and impediments. Completed tasks and materialised but successfully solved impediments are marked in green text colour. Impediments which did not occur are greyed out.

Table 1: Time table

	Tasks	**Potential impediments**
Project Draft		
w1	• collection of data and creating the basic dataset • creating preliminary features (indicators and signals) • deploying labelling methods and creating initial labels	
w2	• performing PCA feature analysis and other metrics • selecting features for the final set • selecting final labels • investigating and selecting classifier algorithms	Some combinations of data frequency with certain classifiers might require unfeasible amounts of computing resources.
w3	• devising a strategy based on the previous analysis • validating the model and simulation testing • comparing the event-driven model against the primary one	Depending on the final strategy, porting the model for simulation testing might prove to be too time-consuming.
Final Project		
w4	• reviewing and implementing feedback received • implementing interactive ipywidgets* • cleaning up the Python code • building the performance report	The Bokeh and free Dash visualisation solutions might both prove to be more limiting than anticipated, leaving us only with a paid Dash subscription as an alternative.
w5	• consolidating and completing the documentation • preparing and filming the presentation	

* Interactive *ipywidgets* proved to be the most suitable solution and have been used instead Bokeh or Dash