Does Meta-Labeling Add to Signal Efficacy?

A. Singh, J. Joubert

Abstract

Successful and long-lasting quantitative research programs require a solid foundation that includes procurement and curation of data, creation of building blocks for feature engineering, state of the art methodologies, and backtesting. In this project we create a open-source python package (mlfinlab) that is based on the work of Dr. Marcos Lopez de Prado in his book Advances in Financial Machine Learning. Dr. de Prados book provides a guideline for creating a successful platform. We also implement a Trend Following and Mean-reverting Bollinger band based trading strategies. Our results confirm the fact that a combination of event-based sampling, triple-barrier method and meta-labeling improves the performance of the strategies.

Keywords: Machine Learning, Meta-Labeling, CUSUM Filter, Triple Barrier Labeling, Trend-following, Mean-Reversion

1. Introduction

Inspired by 2019 Quant of the year Dr. Marcos Lopez de Prado we proposed an implementation and further research into the novel ideas and best practices published in his book Advances in Financial Machine Learning. Our project is split into two capstone sessions, the first six weeks create the foundation (of codes) by publishing an open-source python package which will enable further research into the field of quantitative investing. We also test a couple of trading strategies that leverage the foundation. The second 16 weeks would focus on further implementation of de Prados work and deeper research that culminates in a research article or a paper.

The key contribution in part one are the following:

1. An open-source python package.
2. Transformed data sets to promote further research.
3. Empirical proof that meta-labeling benefits signal generation and thereby performance of the said strategy.

Rest of the report focuses on SWOT analysis, methodology, results, and conclusions. We also discuss next steps and areas of further research. Many of these ideas / next steps are already being formulated and worked on. At the end of the paper is the appendix containing all the information regarding the data and how it was sourced.

2. SWOT Analysis

The following provides a traditional SWOT analysis on our project.

2.1. Strengths

This project reflects the idea of meta-strategies as discussed in López de Prado 2018. It is open-source and allows interested quantitative analysts like us to build on and contribute to it. We consider this to be the starting point with much work to be done. Second, we show that by using tick data and converting into event based sampling methods such as volume, dollar or tick leads to better statistical properties of the data and that in turn helps machine learning algorithms learn and predict. Third, our results corroborate (although with only two strategies) that meta-labeling has a propitious effect on the performance of the strategies.

2.2. Weaknesses

In order to build viable strategies one needs good quality tick data. This is costly and not readily available for research. However, as we show in this project that an expense of $1,000 can help build and test strategies that can generate interest. Second, as much as we show that meta-labeling works, it also needs a good primary algorithm that should have good performance in in-sample tests. One then needs to combine that algorithm with a rich set of features that are contextual, relevant and intuitive. If the algorithm is bad then meta-labeling would likely only reduce the downside.

2.3. Opportunities

This framework offers considerable upside opportunities. For instance,

- We have seen considerable interest from analysts wanting to expand on our work by building (for instance) imbalance bars, test our strategies on Euro STOXX tick data (that one analysts volunteered to purchase) etc.
- Build a feature zoo - a library of functions or function objects (functors) that would create technical and statistical features from the supplied data.
- Incorporate fractional differentiation in the feature set.
- Expand on the research and perhaps write a paper.

2.4. Threats

When we started this project we had noticed several efforts to address concepts outlined in López de Prado 2018. We think as the quantitative finance community becomes familiar with these concepts (meta-labeling, robust back-testing, use of machine learning in crafting signals and strategies, etc.) their use will expand and will become democratized. This is likely to have a downward pressure on alpha. Our belief is that these ideas can be applied to other asset classes and strategies.

3. Methodology

In López de Prado 2018 Dr. de Prado discusses the key success factors underlying successful algorithmic or quantitative investment strategies. One of the success factors is the concept of meta-strategies. First presented in López de Prado and Foreman 2014, it calls for creating a factory like platform for a sustainable long-term success. In this paradigm there are technologies and, roles and responsibilities for data acquisition and curation, high-performance computer infrastructure, feature engineering and analysis, execution simulation, and back-testing. Our methodology therefore starts with creation of the building-blocks for such a platform. For instance,

- For software development and continuous integration we built an open-source framework that would allow other practitioners to add to our work. Hence we are using Github and Travis CI.
- Coded packages to convert tick data into dollar, volume and tick bars; compute fractionally differenced series etc. In most cases we have reused the code from López de Prado 2018 or other sources with attribution. These codes are in a package called mlfinlab.
- Tested two commonly used strategies - trend-following and mean-reversion - to validate the concepts and ideas.
- Employed techniques like filtering to prevent signal whipsaws and improve the efficacy of the signal generation process; up-sampling when there were unbalanced classes; meta-labeling to improve the performance of the machine-learning process;
- Segregated data into training, validation and out-of-sample data sets. We ensured that out-of-sample data set was never used in the training and validation steps. As a best practice, we first trained and validated the model in an iterative process. Only when we felt comfortable with the parameters then we used out-of-sample data. This ensures the sanctity of the strategy design and testing process.
- Lagged the features to ensure that there was no look-ahead bias.
- Used cross-validation and grid search to train Random Forest machine-learning algorithm. The choice was driven by questions at the back of the chapters (2 and 3) of the book López de Prado 2018

In the subsections below we delve deeper into specific aspects of the methodology of this project.

3.1. Financial Data Structures

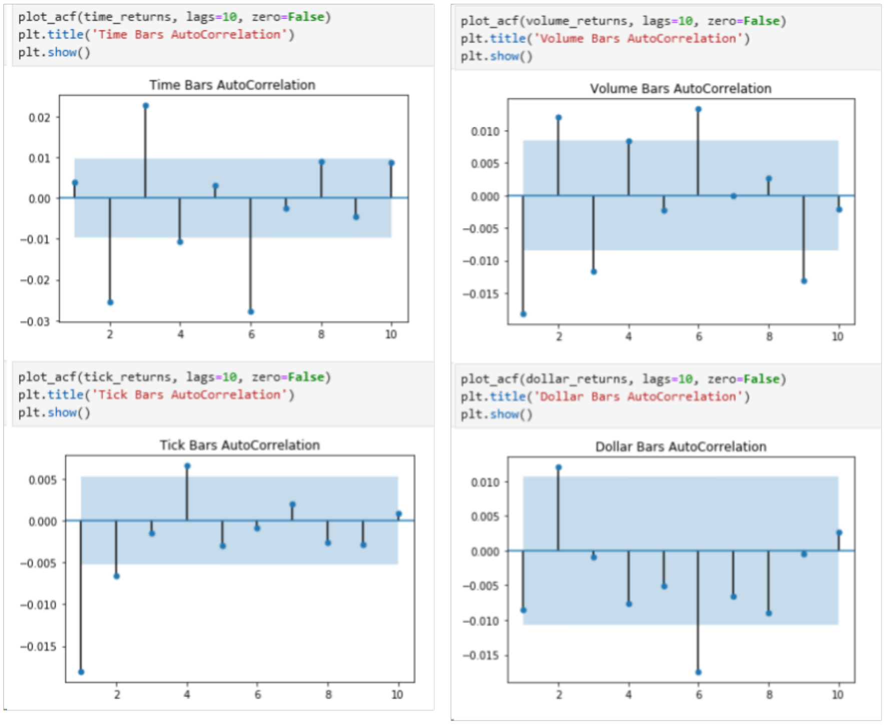
Machine learning in finance focuses on forecasting stock price movements using stock market data such as price and volume. According to the literature Thierry and Hélyette 2000, the stock price movements are nonlinear and stochastic in nature. Specifically, trading activity is rarely uniform during a day or a week or a month. It varies with the information flow in the form of macro-economic data releases, news about political leaders or company specific announcements. Fama and Blume 1966 showed that daily returns are more long tailed than the normal density. It is therefore necessary to sample data using a paradigm called event-based time as discussed in Easley, PRADO, and O'Hara 2011. These techniques involve sampling a session into equal volume chunks or bars (for instance, 100,000 contracts or shares) or dollar bars ($1 million) etc. Empirical analysis shows that these methods have better statistical properties. In addition to dollar and volume bars, there are also tick bars (100,000 ticks).

We computed above stated bars and performed various tests for statistical properties on the returns from these bars. A notebook 2019-03-03_JJ_Sample-Techniques.ipynb, in the Chapter 2 directory has the details. Below we show the Jarque-Bera tests for these bars which show that dollar bars are the closest to normality compared to all other bars (because its test statistic is the smallest).

Test Statistics:

- Time: 1782853
- Tick: 2898186
- Volume: 337591
- Dollar: 143045

The ACF of the bars show that dollar bars have the lowest auto-correlation among all others.



```
plot_acf(time_returns, lags=10, zero=False)
plt.title('Time Bars AutoCorrelation')
plt.show()
```

```
plot_acf(volume_returns, lags=10, zero=False)
plt.title('Volume Bars AutoCorrelation')
plt.show()
```

```
plot_acf(tick_returns, lags=10, zero=False)
plt.title('Tick Bars AutoCorrelation')
plt.show()
```

```
plot_acf(dollar_returns, lags=10, zero=False)
plt.title('Dollar Bars AutoCorrelation')
plt.show()
```

Figure 1: ACF on the various bars

The following figure illustrates how using event based sampling leads to a partial recovery of normality. This chart is inspired by Easley, PRADO, and O'Hara 2011.

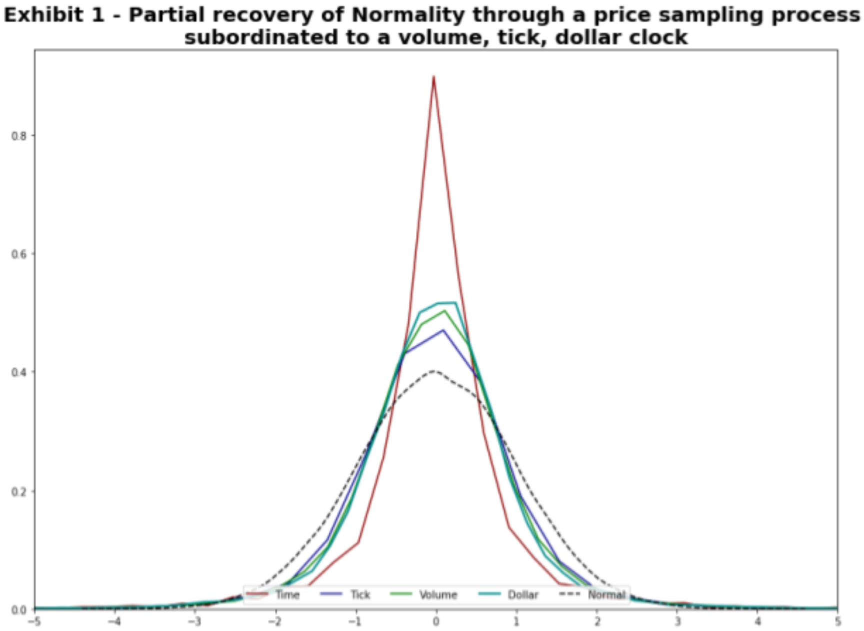


Figure 2: Partial Recovery of Normality

3.2. Issues with Machine Learning in Finance

Academic researchers and practitioners have found that prediction of stock price movements is more effective (compared to linear models) with algorithms that are themselves nonlinear, adaptive, and dont assume a fixed functional form. According to literature, machine learning methods such as Random Forests and ANN are better at forecasting stock prices partially because they are better at capturing the non-linearity in the asset prices. Wang and Chan 2006 indicate that efficacy of the forecasts tend to improve when multiple classifiers are organized in serial, conditional, hybrid or parallel combinations.

In the attached Jupyter notebooks we create trend-following and Bollinger band mean-reversion strategies. These use the concepts and best practices discussed above. The steps in these notebooks have the following flow:

1. Compute long short signals for the strategy. For instance, in the mean-reverting strategy, generate a long signal when the close price is below the lower Bollinger band and create a sell signal if the close price is higher than the upper Bollinger band. We call this the Primary model.
2. Get time stamps of the events using CUSUM (or cumulative sum control chart) filter and point estimate of the volatility. See section 4.2.

3. Determine events when one of the three exit points (profit taking, stop-loss and vertical barrier) occur. López de Prado 2018 discusses this in Chapter 3. The result of this step is a trade decision long or short, or 1 or -1.
4. Determine the bet size. The prior step tell us the direction of the trade. This step says if we should trade or not a one or zero decision.
5. Tune the hyper-parameters (max_depth and n_estimators) of Random Forest using grid search and cross-validation. We keep the random state constant for reproducibility of the results.
6. Train a machine-learning algorithm (we use Random Forest for illustration) with new features like one to five day serial correlations, one to five-day returns, 50-day volatility, and 14-day RSI. We iterate over this step number of times until we see in-sample results that are acceptable. In other words, we only exit this step when we consider the model to be ready and there is no turning back.
7. Evaluate the performance of in sample and out-of-sample or this meta-model model.
8. Evaluate the performance of the Primary model
9. Compare the performance of the meta-model and the primary model

3.2.1. Training Random Forest

We found that at the completion of step 4 above, the number of observations tagged 1 (to trade) were considerably smaller than 0 (not to trade). To provide balanced classes and thereby get better trained classifier we up-sampled (sampled with replacement) the training data to balance the classes. To gauge the performance of the model we employed the Classification Report, Confusion Matrix and Receiver Operating Characteristic (ROC) curve.

3.3. Filtering

Alexander 1961; Alexander 1964 showed the belief among the investment professionals that the asset prices gradually adjust to new information. This creates trends as opposed to instantaneous jumps as market participants become aware of new information. Alexander 1961 says that this meant that if the prices have moved up (or down) by x percent then they are likely to move more than x percent further before moving down x percent.

Lam and Yam 1997 use the CUSUM filter to detect an upward or downward shift in the prices and use that to generate trading signals. CUSUM or Cumulative Sum Control Chart is a technique used to detect shift in the mean of a process away from a target value. Consider a locally stationary process $\{y_t\}_{t=1,\ldots,T}$. Define a cumulative sum S_t such that:

$$S_t = max\{0, S_{t-1} + y_t - E_{t-1}[y_t]\}$$

A symmetric CUSUM filter can be defined (as done by López de Prado 2018 that will detect any shift on the up and down side.

$$\begin{aligned} S_t^+ &= max\{0, S_{t-1}^+ + y_t - E_{t-1}[y_t]\}, S_0^+ = 0 \\ S_t^- &= min\{0, S_{t-1}^- + y_t - E_{t-1}[y_t]\}, S_0^- = 0 \\ S_t &= max\{S_t^+, -S_t^-\} \end{aligned}$$

López de Prado 2018 pg 38 employs the CUSUM filter to detect events that would trigger a trade. These events could be a structural break, an extracted signal or micro-structural phenomenon López de Prado 2018. There are two advantages to using a filter such as CUSUM: first, it samples key events in the data. Second, the filter prevents multiple events from getting generated when the price series hovers around a threshold value, thereby preventing whipsaws in trading.

We employ CUSUM filter as suggested by López de Prado 2018 with the threshold of point-in-time volatility.

3.4. Triple Barrier Labeling

In the majority of the literature, authors will make use of a labeling scheme where they classify the next periods directional move as either a 1 for a positive move, a -1 for a negative move, and some authors may add a threshold level that if the return is not above or below it, then a 0 label is provided.

This technique has a few flaws. First the threshold level is usually static and stock returns are known to be heteroskedastic, the volatility changes over time and a fixed threshold value fails to account for this. Second, using this -1, 0, 1 scheme fails to account for positions that would have been closed by stop loss or profit taking orders.

A more advanced technique such as the Triple Barrier method López de Prado 2018, addresses these concerns and I am sure that many of you will agree - it makes more sense.

In derivatives pricing, a series of stock prices can be modeled using Geometric Brownian Motion. Similarly in the Triple Barrier method, we assume that stock prices follow a random walk with some drift and variance, we then label this path.

At a given time stamp, 3 barriers are set. An upper and lower horizontal barrier to represent a take profit and stop loss levels. A third and vertical barrier is placed to represent the end of the duration of the trade.

Should the path of a stock reach the upper barrier before the vertical then a value of 1 is returned, conversely if it reaches the bottom barrier then a -1, however should the stock price reach the vertical barrier first then a 0 is returned. This is still a -1. 0, 1 scheme, however we are labeling a path of returns rather than the next directional move.

The horizontal barriers are determined by calculating the daily standard deviation of the log returns multiplied by a user defined multiple. For example a [1, 1] tuple will set both barriers to be equal to 1 standard deviation.

The following figure provides an example:

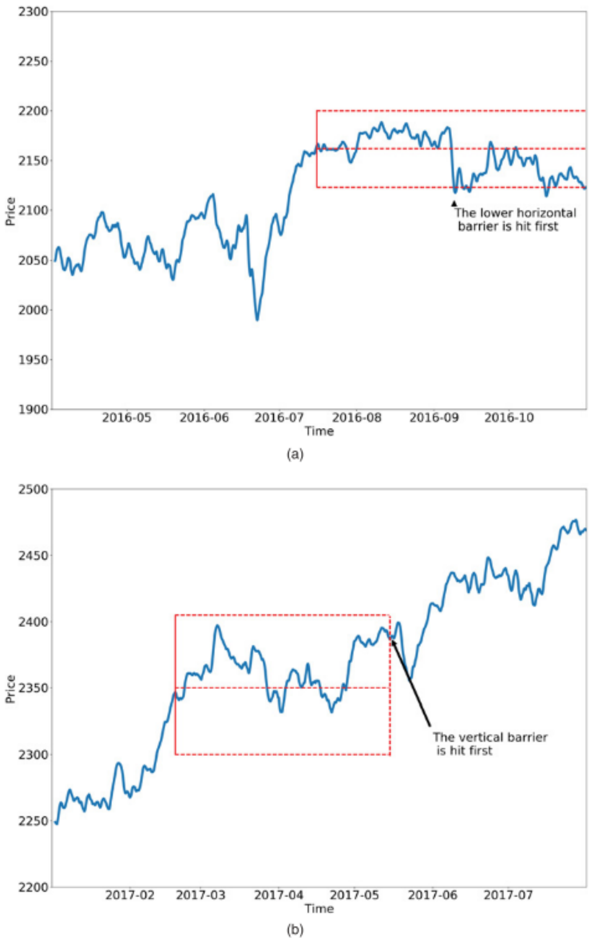


Figure 3: Triple Barrier Labeling
López de Prado 2018

In chart (a) we can see that the lower horizontal barrier is first reached, a -1 value is returned. In chart (b) the path never reaches the horizontal barriers and triggers a 0 label when the vertical barrier is reached.

3.5. Fitting a Primary Model

The primary model is the component that determines which side of the trade to take. It generates a signal -1, 0, 1. Where -1 is a short position, 1 is a long position, and 0 means to close all positions.

This model could be but not limited to:

- Statistical arbitrage model based on the spread between two assets.
- Machine learning model such as an SVM or Neural Network.
- Fundamental value or events based strategy where the portfolio manager generates the signal.
- Rules based, technical trading strategy such as moving average crossovers.

The only requirement is that a signal is generated which is used to determine the side of the position. We look to meta labeling and bet sizing to determine the size of the position.

The following two sections discuss the technical analysis inspired strategies we used.

3.5.1. Trend Following

A simple moving average crossover strategy is employed. The idea behind this strategy is to make use of two moving averages to help smooth out the noise in the data and then determine when a trend is in affect.

Traditionally a slow 200 day and a fast 50 day moving average are used. When the fast moving average crosses above the slow, a buy signal (1) is generated. Conversely when the fast crosses below the slow then a sell signal (-1) is generated. Under this scheme, there is always a long or a short side active, i.e. no 0 signals. The figure below shows an example of this.

The green upward arrows indicate when a long (buy) signal is in affect and a red downward arrow a short (sell) signal.

For the primary trend following model we implemented a 20 and 50 bar SMA crossover strategy. Remember that we reduced the number of events by making use of the CUSUM filter, because of this we need much shorter SMA periods to capture the short term trends that may be in affect, and provide more current information to the secondary model since the vertical barrier is set to a single day.

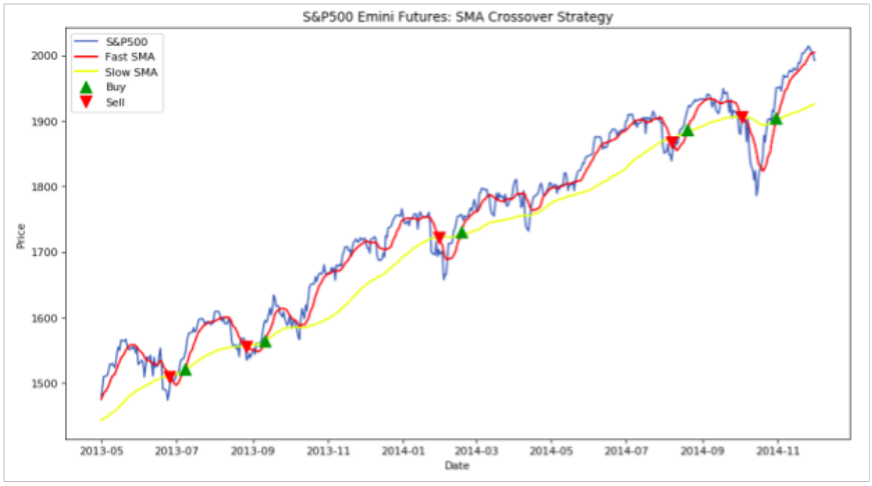


Figure 4: SMA Crossover Strategy

3.5.2. Mean Reversion

The second primary model is based on mean reversion and makes use of Bollinger Bands. Bollinger Bands are a technical analysis indicator which creates bands around the price level which are more than x standard deviations away, where x is a user defined multiple.

The principal is that stock prices are log normally distributed and thus we can make use of the Empirical rule which states that 99.7% of the data lies within 3 standard deviations, 95% within 2 and 68% within 1 standard deviation. Should the closing price be above say 2 standard deviations then we generate short signal (-1) on the premise that prices should mean revert in the near term. The reverse is also true, if prices are below 2 standard deviations a buy signal is generated (1).

The figure below shows an example of a traditional Bollinger band strategy.

The green upward arrows indicate when a long (buy) signal is in affect and a red downward arrow a short (sell) signal.

Typically a position is held until the price reaches the moving average but in our case, because we are using the triple barrier method, a position is held until one of the three barriers are touched.

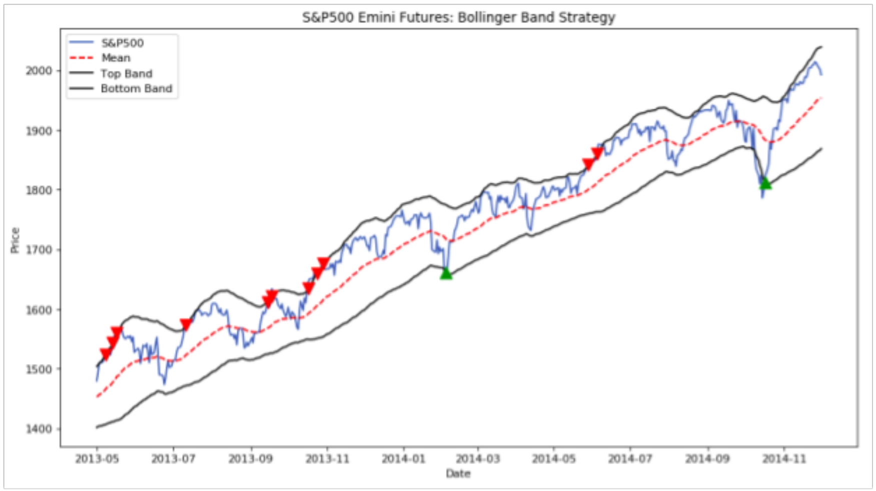


Figure 5: Bollinger Band Mean Reversion Strategy

3.6. Meta-Labeling

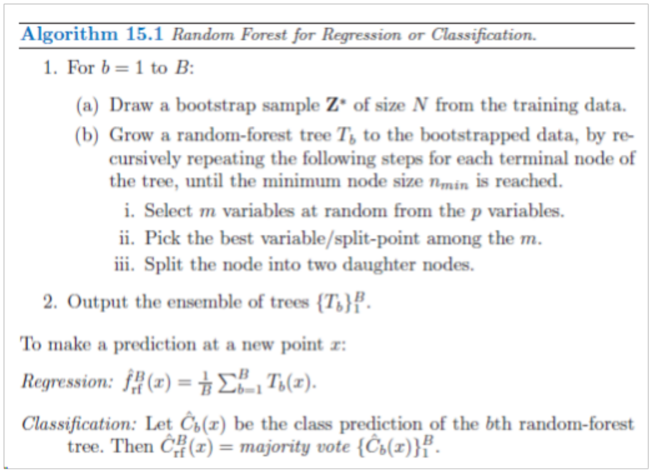
The central idea is to create a secondary machine learning (ML) model that learns how to use the primary exogenous model. This leads to improved performance metrics, including: Accuracy, Precision, Recall, and F1-Score. For those readers who are interested in building up a deeper intuition around meta-labeling, we have included a thorough section in Appendix B.

Use in Financial Machine Learning

Meta-labeling in finance follows the same principles as we outlined in Appendix B. First we make use of a primary model, in this case a simple trend following and mean reverting strategy, to determine the position of the trade. Then we fit a Random Forest meta-label model to the primary model to determine when to trade or not.

3.7. Random Forest Model

Random forests (RF) (Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome. 2009) is a leap forward from the bagging algorithm that build an ensemble of de-correlated trees, and then averages them. The idea of bagging and RF methods is that an average of a collection of (uncorrelated/unbiased and noisy) learning methods lead to a better result (lower variance). Trees themselves are very good at capturing the complex interaction or nonlinear structures in the data, they are also un-biased when grown to a sufficient depth. The performance of RF is similar to that of boosting method, and they are simpler to train and tune. RF can be used for classification and regression problems. The algorithm is defined as follows (Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome. 2009, page 588):



```
Algorithm 15.1 Random Forest for Regression or Classification.
1. For $b = 1$ to $B$:
    (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size $N$ from the training data.
    (b) Grow a random-forest tree $T_b$ to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{min}$ is reached.
        i. Select $m$ variables at random from the $p$ variables.
        ii. Pick the best variable/split-point among the $m$.
        iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.
To make a prediction at a new point $x$:
Regression: $\hat{f}_{\mathrm{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.
Classification: Let $\hat{C}_b(x)$ be the class prediction of the $b$th random-forest tree. Then $\hat{C}_{\mathrm{rf}}^B(x) = \mathit{majority\ vote}\ \{\hat{C}_b(x)\}_1^B$.
```

Figure 6: Random Forest Algorithm
Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome. 2009

3.7.1. Important hyperparameters

Hyperparameters are used to improve the performance of a learning algorithm. In Sklearns RF class, the following hyperparameters play a significant role:

- n_estimators: This is the number of trees that the algorithm builds before it takes the average of predictions. Higher the number of trees, better the predictions but it also negatively affects the system performance.
- max_features: It is the maximum number of features that RF should consider when splitting a node.
- min_sample_leaf: the minimum number of leaves that are required to split an internal node.

3.7.2. Grid Search

Grid search does an exhaustive search over specified parameter values for an estimator. Like any other classifier, it implements a fit and a predict method except that the parameters of the classifier used to predict is optimized by cross-validation. scikit learn 2019. The schematic below provides greater detail:

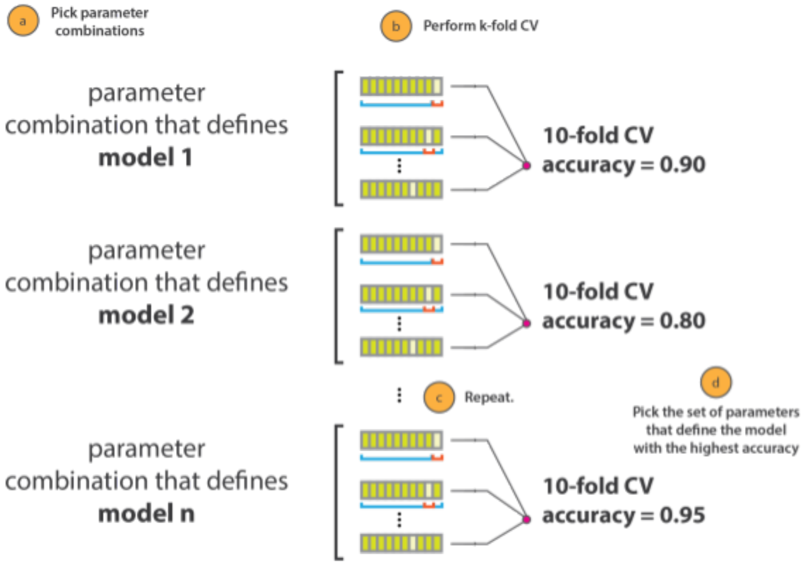

a
Pick parameter combinations
b
Perform k-fold CV
parameter combination that defines model 1
10-fold CV accuracy = 0.90
parameter combination that defines model 2
10-fold CV accuracy = 0.80
c
Repeat.
d
Pick the set of parameters that define the model with the highest accuracy
parameter combination that defines model n
10-fold CV accuracy = 0.95

Figure 7: Random Forest Grid Search
NATLAT 2016

3.7.3. Cross-Validation

Cross-validation (CV) is used to estimate the expected prediction error or an algorithm [Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome. 2009, page 241]. It is easy to understand and implement, and often results in lower bias than other methods. K-fold CV tries to overcome the issue of limited data by taking a portion of the data to fit the model, and a different part to test it. In a 5 part CV, one of the five parts will be set aside for testing and others will be used to fit the model. (Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome. 2009) provides guidelines for how to use CV:

1. Divide the samples into K CV folds or groups at random.
2. For each fold $k = 1...K$:
    (a) Find a subset of good predictors that show strong correlation with the class labels using all the samples except those in fold k.
    (b) Using just this subset of predictors, build a multivariate classifier, using all the samples except the fold k.
    (c) Use the classifier to predict the class labels for the samples in fold k.

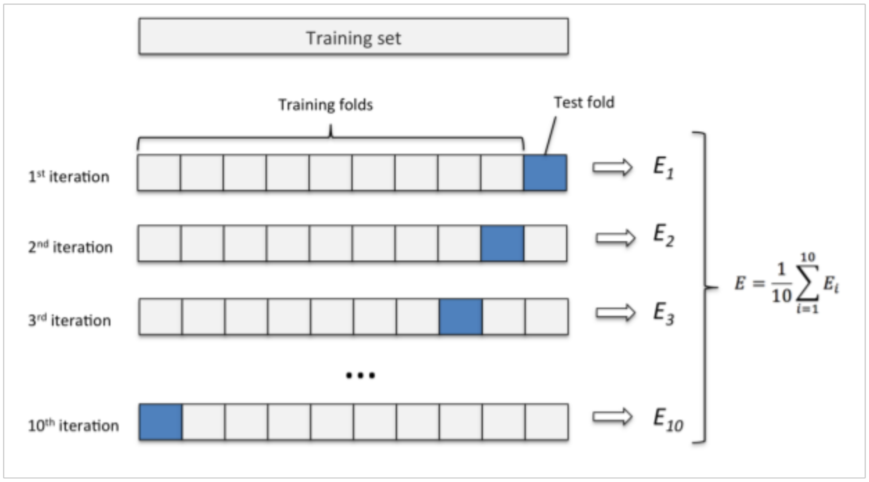


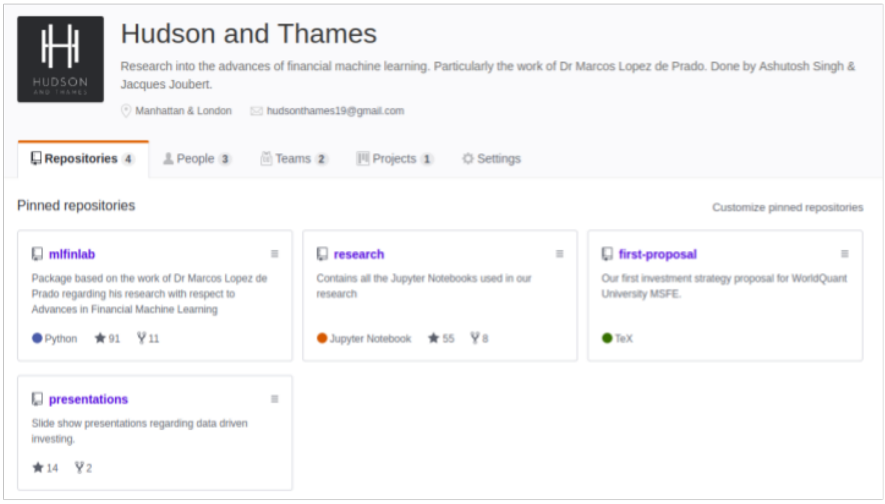
Figure 8: K Fold Cross Validation
Norena, Sebastian 2018

4. Results

We developed the packages and Jupyter Notebooks and shared them on Github (Singh, Ashutosh. Joubert, Jacques. 2019). The core functionality is under the package name mlfinlab. As we stated above (in the section, Methodology) that our goal was to build a platform where practitioners can use our codes and also contribute to this research. We are happy to report that this library has received considerable interest from the quantitative finance community and several have volunteered to add to the code base. A few have forked from the repository to extend the work we have done so far.

4.1. Performance of the Strategies

We tested two trading strategies - trend-following and Bollinger band mean-reverting to use our framework and test their performance. During the training and validation phases of the strategy build-out, we manually tuned a few parameters to ensure that we have sufficient data points. For instance, a function called get_t_events filters the data (using CUSUM filter) for events when there has been a structure shift. We changed the threshold parameter manually to get sufficiently large data set. Second, we found that meta-labeling often resulted in unbalanced classes - to trade (=1) or not to trade (=0) with many more instances of not to trade. We used up-sampling to balance these classes prior to training the machine-learning algorithm (Random Forest).



Hudson and Thames

Research into the advances of financial machine learning. Particularly the work of Dr Marcos Lopez de Prado. Done by Ashutosh Singh & Jacques Joubert.

Manhattan & London hudsonthames19@gmail.com

Pinned repositories

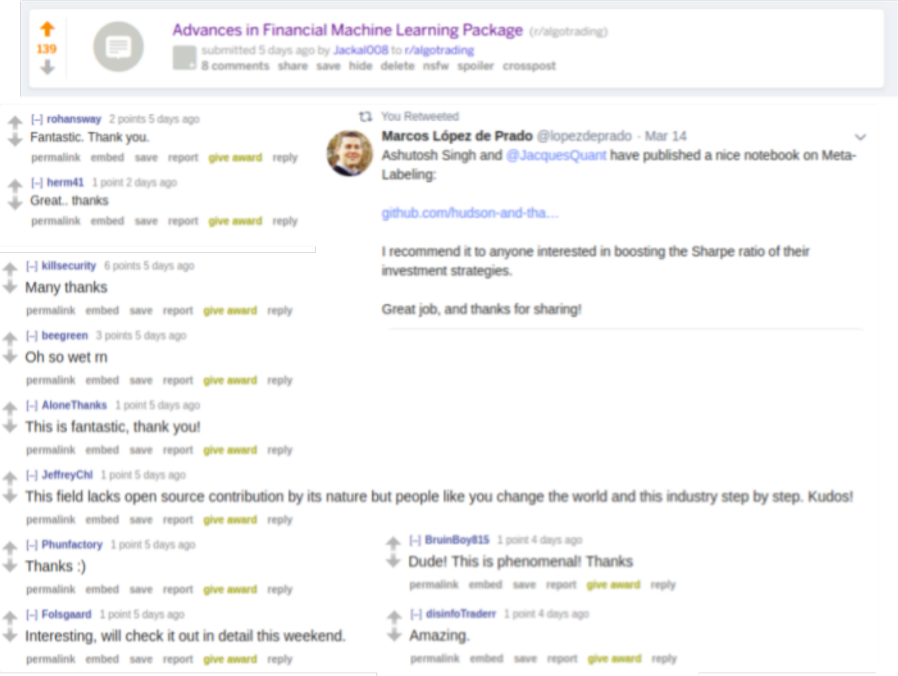
mlfinlab
Package based on the work of Dr Marcos Lopez de Prado regarding his research with respect to Advances in Financial Machine Learning
Python ★ 91 Forks 11

research
Contains all the Jupyter Notebooks used in our research
Jupyter Notebook ★ 55 Forks 8

first-proposal
Our first investment strategy proposal for WorldQuant University MSFE.
TeX

presentations
Slide show presentations regarding data driven investing.
★ 14 Forks 2

Figure 9: Project Dashboard


139
8 comments share save hide delete nsfw spoiler crosspost
permalink embed save report give award reply
You Retweeted

Advances in Financial Machine Learning Package (r/algotrading)
submitted 5 days ago by Jackal008 to r/algotrading

[–] rohansway 2 points 5 days ago
Fantastic. Thank you.

[–] herm41 1 point 2 days ago
Great.. thanks

[–] killsecurity 6 points 5 days ago
Many thanks

[–] beegreen 3 points 5 days ago
Oh so wet rn

[–] AloneThanks 1 point 5 days ago
This is fantastic, thank you!

[–] JeffreyChl 1 point 5 days ago
This field lacks open source contribution by its nature but people like you change the world and this industry step by step. Kudos!

[–] Phunfactory 1 point 5 days ago
Thanks :)

[–] Folsgaard 1 point 5 days ago
Interesting, will check it out in detail this weekend.

Marcos López de Prado @lopezdeprado · Mar 14
Ashutosh Singh and @JacquesQuant have published a nice notebook on Meta-Labeling:

github.com/hudson-and-tha…

I recommend it to anyone interested in boosting the Sharpe ratio of their investment strategies.

Great job, and thanks for sharing!

[–] BruinBoy815 1 point 4 days ago
Dude! This is phenomenal! Thanks

[–] disinfoTraderr 1 point 4 days ago
Amazing.

Figure 10: Community Feedback

4.1.1. Performance Metrics

To evaluate the efficacy of meta-labeling we look at a models performance metrics between the validation set and the out-of-sample test set. This allows us to draw conclusions about the models ability to generalize. In particular we need to look at the recall, precision, F1 score, and accuracy. The reason why we dont compare the strategies performance metrics (annualized returns, sharpe ratio, and drawdowns) is because the two data sets are from very different time periods. For example, if the validation set has a much higher volatility than the test set, then the validation returns will be larger. This will prevent like for like comparison.

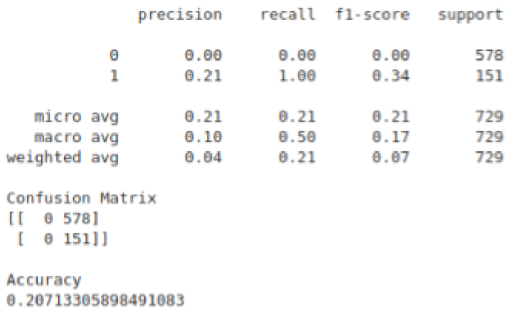
We can however compare strategy metrics if they are both from the same time period. We do provide performance metrics on the test data. Additionally we add a performance tear sheet, and see that meta-labeling results in better strategy metrics but it should be noted that we have yet to add a bet sizing component to the strategy. Additionally the two strategies we test are based on technical analysis and they dont provide the best signals. A primary model with better predictive power would provide further insights.

4.1.2. Bollinger Band Mean-Reversion Strategy.

We construct 1.5 standard deviation upper and lower bands around the average closing price of the S&P500 e-mini futures. The strategy buys when the close price falls at or below the lower band and sells when the close price rises at or above the upper band. These generate the buy/sell signals also called the side. The meta-labeling function decides on the size (to trade or not to trade). This information along with features such as 14-day RSI, volatility, 7 and 15-day moving averages, one to five day auto-correlation, and one to five day momentum is used to train Random Forest algorithm. The trained algorithm is used to validate the signal. Finally, after finalizing the algorithm we use the trained model to test out-of-sample.

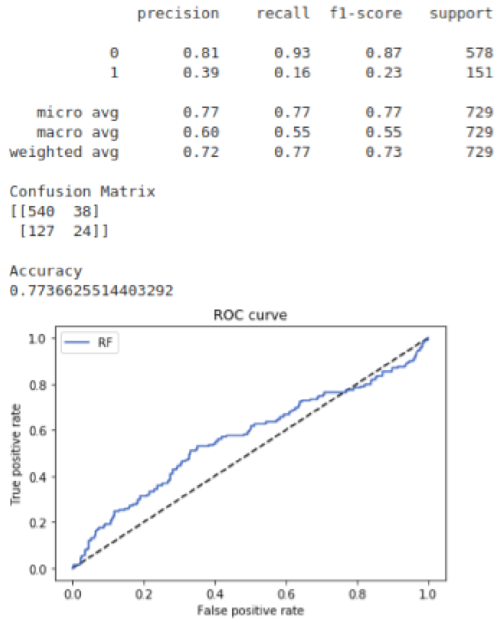
The results are as follows:

Validation Data



```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       578
           1       0.21      1.00      0.34       151

   micro avg       0.21      0.21      0.21       729
   macro avg       0.10      0.50      0.17       729
weighted avg       0.04      0.21      0.07       729

Confusion Matrix
[[  0 578]
 [  0 151]]

Accuracy
0.20713305898491083
```

Figure 11: Primary Model on Validation Set (Mean Reverting)

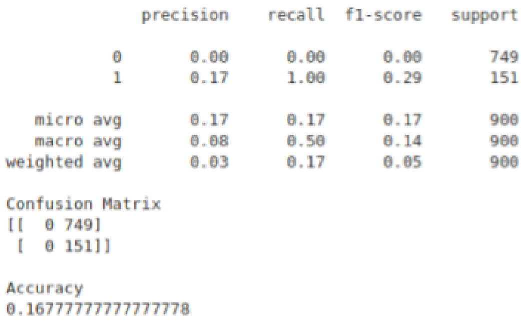


```
              precision    recall  f1-score   support

           0       0.81      0.93      0.87       578
           1       0.39      0.16      0.23       151

   micro avg       0.77      0.77      0.77       729
   macro avg       0.60      0.55      0.55       729
weighted avg       0.72      0.77      0.73       729

Confusion Matrix
[[540  38]
 [127  24]]

Accuracy
0.7736625514403292
```

Figure 12: Meta Model on Validation Set (Mean Reverting)

In the validation data we can see that the performance metrics increase. The accuracy jumps from 20% to 77%. The precision of correct trades also jumps from 0.21 to 0.39, this will correlate to greater profits and lower drawdowns.

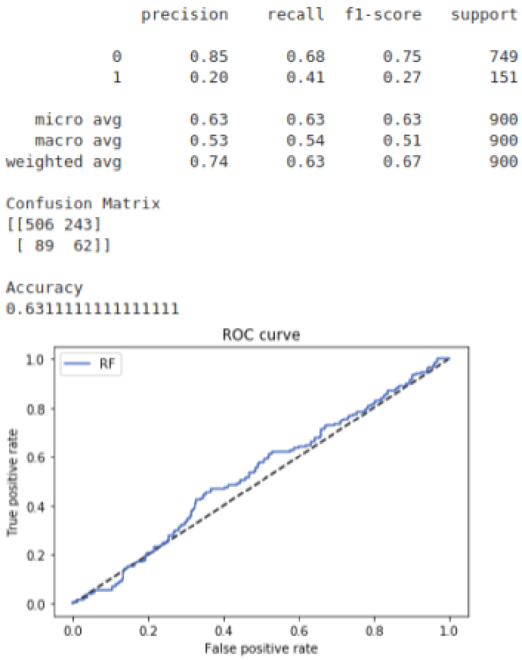
Out-of-Sample Data



```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       749
           1       0.17      1.00      0.29       151

   micro avg       0.17      0.17      0.17       900
   macro avg       0.08      0.50      0.14       900
weighted avg       0.03      0.17      0.05       900

Confusion Matrix
[[  0 749]
 [  0 151]]

Accuracy
0.16777777777777778
```

Figure 13: Primary Model on Out-of-Sample Set (Mean Reverting)

model.png model.png model.png



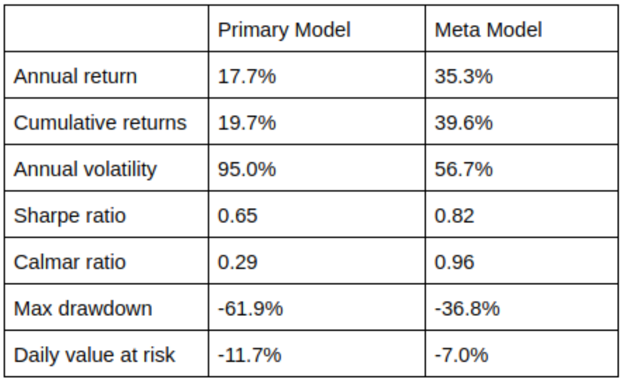
```
              precision    recall  f1-score   support

           0       0.85      0.68      0.75       749
           1       0.20      0.41      0.27       151

   micro avg       0.63      0.63      0.63       900
   macro avg       0.53      0.54      0.51       900
weighted avg       0.74      0.63      0.67       900

Confusion Matrix
[[506 243]
 [ 89  62]]

Accuracy
0.6311111111111111
```

Figure 14: Meta Model on Out-of-Sample Set (Mean Reverting)

This test data is completely out-of-sample. The precision jumps from 0.17 to 0.20 and the accuracy from 17% to 63%. This should translate to improved strategy performance metrics as well.

Strategy Performance Metrics

Table 1: Out-of-sample (2018-01-04 : 2019-01-28)



	Primary Model	Meta Model
Annual return	17.7%	35.3%
Cumulative returns	19.7%	39.6%
Annual volatility	95.0%	56.7%
Sharpe ratio	0.65	0.82
Calmar ratio	0.29	0.96
Max drawdown	-61.9%	-36.8%
Daily value at risk	-11.7%	-7.0%

This shows that the meta-model adds a lot of value to the out-of-sample performance. All the metrics have improved across the board.

Performance Tear Sheet

The following charts are added for sake of completeness and to illustrate the risk return profile of the mean reverting strategy.

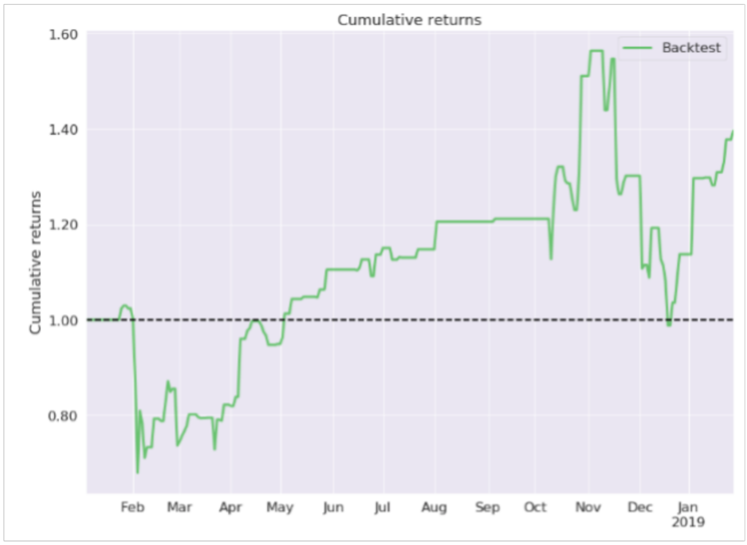


Figure 15: Cumulative Returns (Mean Reverting)

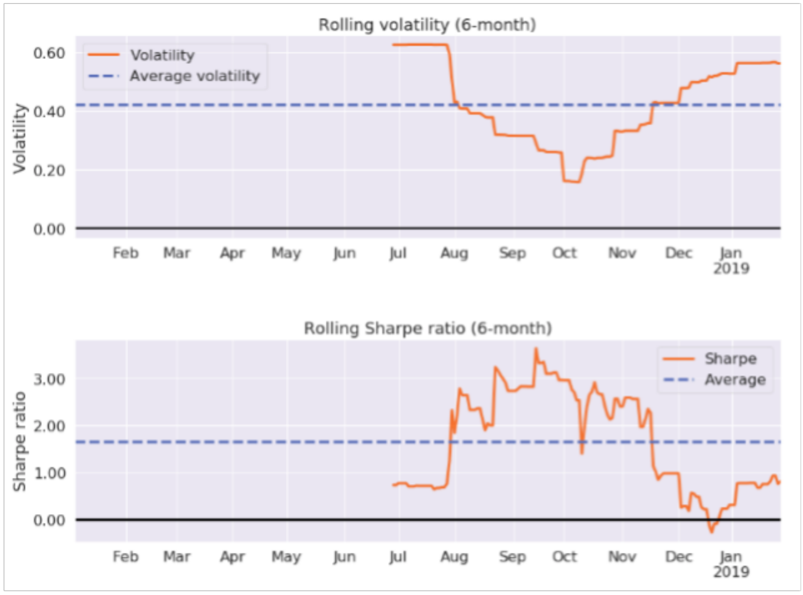


Figure 16: 6 Month Volatility and Sharpe Ratio (Mean Reverting)

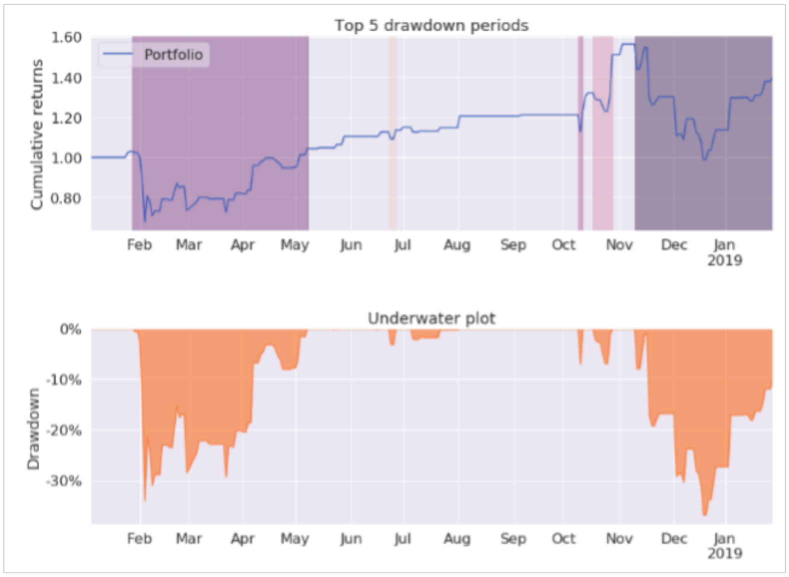


Figure 17: Drawdowns and Underwater Plot (Mean Reverting)

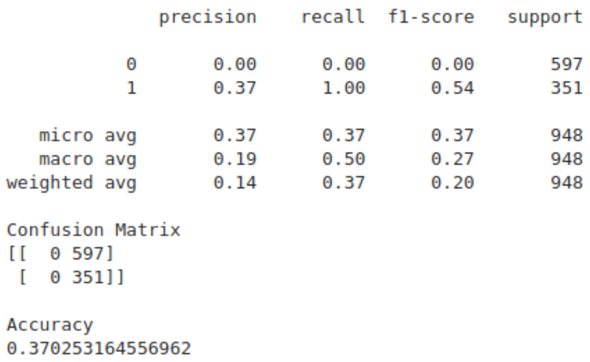
4.1.3. Simple Moving Average (SMA) Crossover - Trend Following Strategy

We construct two moving averages. A fast 20 bar SMA and a slow 50 bar SMA around the closing price of the S&P500 e-mini futures.

The strategy buys when the fast SMA is above the slow SMA and sells when the fast SMA is below the slow SMA. These generate the buy/sell signals also called the side. The meta-labeling function decides on the size (to trade or not to trade). This information along with features such as fifty, thirty one, and fifteen bar rolling volatility, one to five day auto-correlation, and one to five day momentum is used to train Random Forest algorithm. The trained algorithm is used to validate the signal. Finally, after finalizing the algorithm we use the trained model to test out-of-sample.

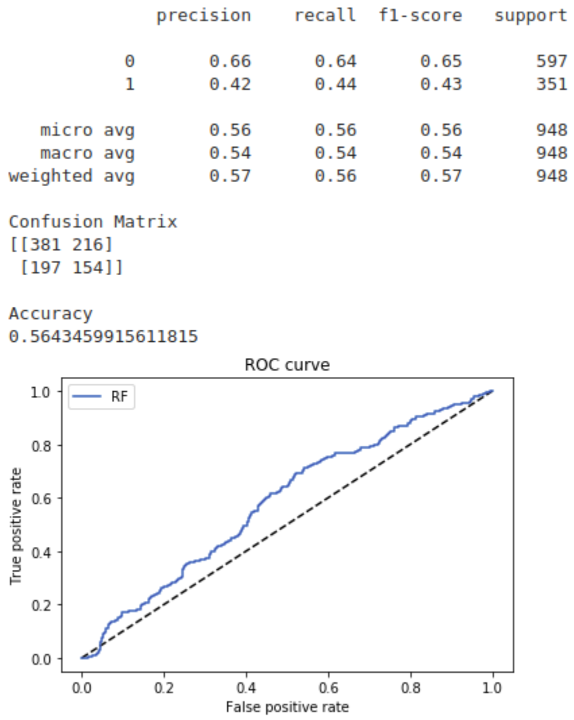
The results are as follows:

Validation Data



```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       597
           1       0.37      1.00      0.54       351

   micro avg       0.37      0.37      0.37       948
   macro avg       0.19      0.50      0.27       948
weighted avg       0.14      0.37      0.20       948

Confusion Matrix
[[  0 597]
 [  0 351]]

Accuracy
0.370253164556962
```

Figure 18: Primary Model on Validation Set (Trend Following)

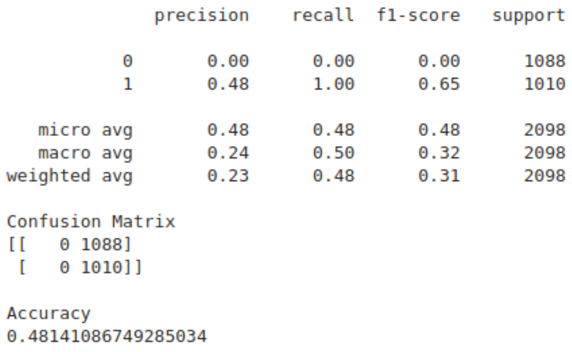


```
              precision    recall  f1-score   support

           0       0.66      0.64      0.65       597
           1       0.42      0.44      0.43       351

   micro avg       0.56      0.56      0.56       948
   macro avg       0.54      0.54      0.54       948
weighted avg       0.57      0.56      0.57       948

Confusion Matrix
[[381 216]
 [197 154]]

Accuracy
0.5643459915611815
```

Figure 19: Meta Model on Validation Set (Trend Following)

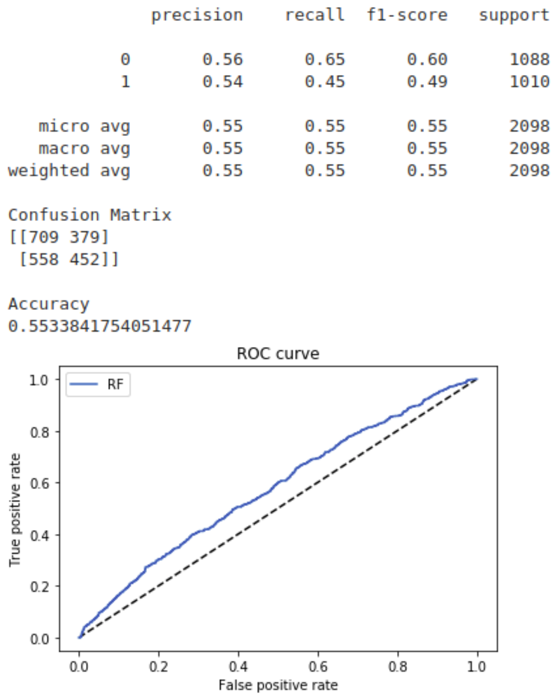
In the validation data we can see that the performance metrics increase. The accuracy jumps from 37% to 56%. The precision of correct trades also increases from 0.37 to 0.42, this will correlate to greater profits and lower drawdowns in the long run.

Out-of-Sample Data



```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1088
           1       0.48      1.00      0.65      1010

   micro avg       0.48      0.48      0.48      2098
   macro avg       0.24      0.50      0.32      2098
weighted avg       0.23      0.48      0.31      2098

Confusion Matrix
[[   0 1088]
 [   0 1010]]

Accuracy
0.48141086749285034
```

Figure 20: Primary Model on Out-of-Sample Set (Trend Following)



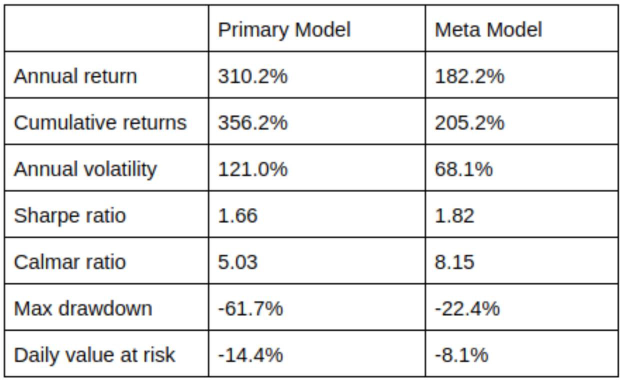
```
              precision    recall  f1-score   support

           0       0.56      0.65      0.60      1088
           1       0.54      0.45      0.49      1010

   micro avg       0.55      0.55      0.55      2098
   macro avg       0.55      0.55      0.55      2098
weighted avg       0.55      0.55      0.55      2098

Confusion Matrix
[[709 379]
 [558 452]]

Accuracy
0.5533841754051477
```

Figure 21: Meta Model on Out-of-Sample Set (Trend Following)

This test data is completely out-of-sample. The precision increases from 0.48 to 0.54 and the accuracy from 48% to 55%. This should translate to improved strategy performance metrics as well.

Strategy Performance Metrics

Table 2: Out-of-sample (2018-01-18 : 2019-01-31)



	Primary Model	Meta Model
Annual return	310.2%	182.2%
Cumulative returns	356.2%	205.2%
Annual volatility	121.0%	68.1%
Sharpe ratio	1.66	1.82
Calmar ratio	5.03	8.15
Max drawdown	-61.7%	-22.4%
Daily value at risk	-14.4%	-8.1%

The above is slightly different to the mean reverting strategy as it doesnt out perform on all the metrics however it does outperform on a risk adjusted basis. This is exactly what meta-labeling sets out to do!

Performance Tear Sheet

The following charts are added for sake of completeness and to illustrate the risk return profile of the trend following strategy.

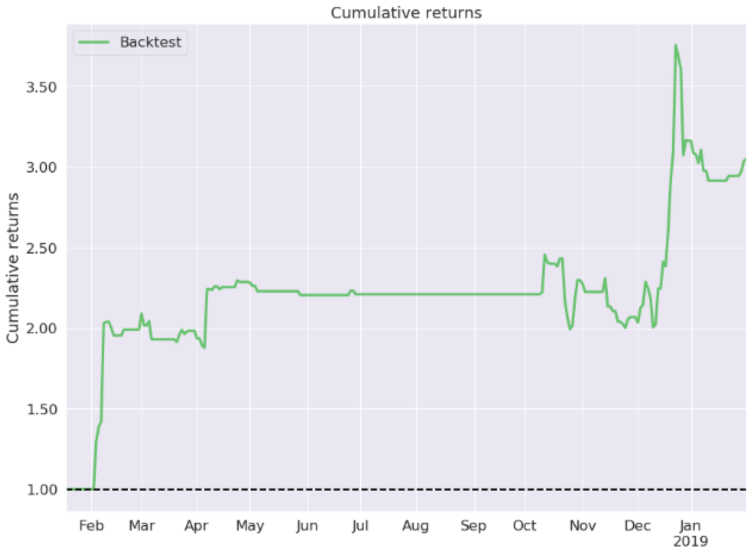


Figure 22: Cumulative Returns (Trend Following)

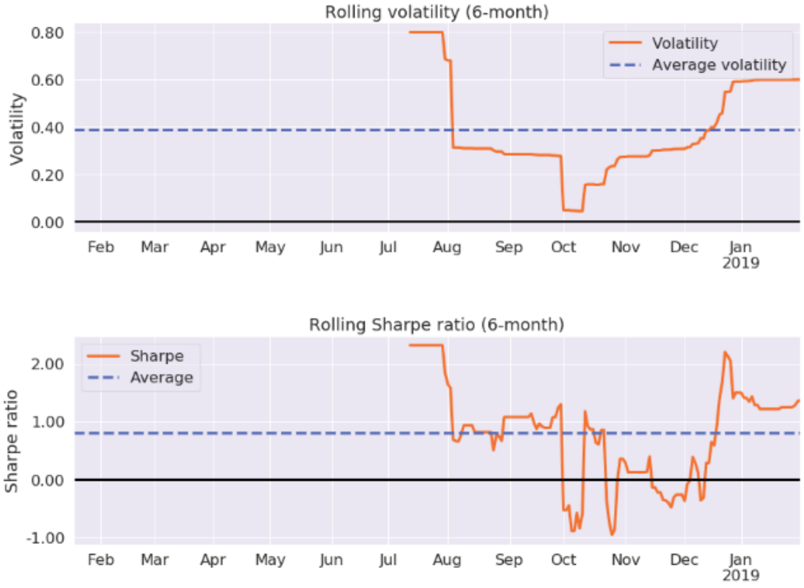


Figure 23: 6 Month Volatility and Sharpe Ratio (Trend Following)

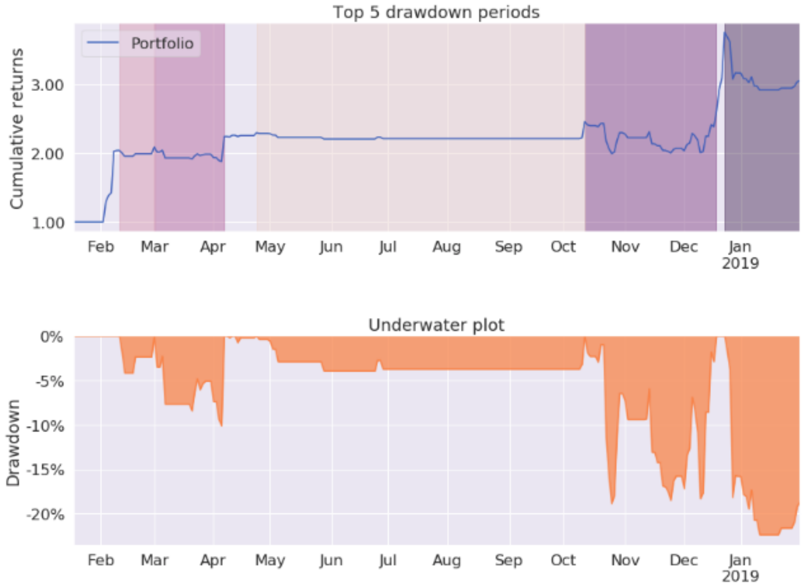


Figure 24: Drawdowns and Underwater Plot (Trend Following)

5. Next Steps

We plan to continue to enhance and expand the platform and the mlfinlab package. Specifically, in the short-term:

- Use the best-practices of cross-validation (see section on Random Forests, Cross- Validation and Grid Search).
- Add position sizing (bet sizing [de Prado 2018, Chapter 10] and risk management to the strategies. This will provide a much more realistic picture of a strategys performance.
- Build unit-tests for each of the library functions.
- Build a feature zoo.
- Use new features and a better model to redesign the current trend-following strategy.
- Test the strategies with other data such as Euro STOXX index.
- Write a paper.

6. Conclusion

This capstone project was conceived as a step toward a larger goal of creating a platform for ongoing quantitative research that [de Prado 2018] speaks about in the form of meta-strategies. Our goal in this phase of the larger endeavor was to create an open-source package that serves as a foundation and then leverage that to test a couple of trading strategies. We also wanted to use concepts, ideas and theories learnt from courses, projects and papers during the MSFE at WorldQuant University.

Given the interest shown by various quant practitioners and Dr. de Prado, the author of the book Advances in Financial Machine Learning, we feel that we are on the right track. We also did not want this to be a purely pedagogical but examine the efficacy of the key concepts like meta-labeling and triple-barrier. Our results on the two strategies - trend-following and mean-reversion - bear that out (Please see Results section).

But as we stated above, this is only the first step and much work needs to be done. We have discussed in the section Next Steps many of the immediate to dos. In the long-term we hope to learn more via the discussion and contribution from others as we continue to contribute.

References

[Ale61] Sidney S. Alexander. “Price Movements in Speculative Markets: Trends or Random Walks”. In: *Industrial Management Review* 2 (1961), pp. 7–26.

[Ale64] Sidney S. Alexander. “Price Movements in Speculative Markets: Trends or Random Walks, No. 2”. In: *Industrial Management Review* 5 (1964), pp. 25–46.

[EPO11] David Easley, MARCOS LÓPEZ DE PRADO, and Maureen O’Hara. “The Volume Clock: Insights into the High-Frequency Paradigm”. In: *Journal of Portfolio Management* Cl (2011), pp. 901–921. URL: `https://c.mql5.com/forextsd/forum/173/the%7B%5C_%7Dvolume%7B%5C_%7Dclock%7B%5C_%7D-%7B%5C_%7Dinsights%7B%5C_%7Dinto%7B%5C_%7Dthe%7B%5C_%7Dhigh%7B%5C_%7Dfrequency%7B%5C_%7Dparadigm.pdf`.

[FB66] Eugene F. Fama and Marshall E. Blume. “Filter rules and stock-market trading”. In: *Journal of Business* 39.1 (1966), pp. 226–241.

[Has09] Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome. *The Elements of Statistical Learning.* 2009. URL: `http://web.stanford.edu/~hastie/ElemStatLearn/`.

[LF14] Marcos López de Prado and Matthew D. Foreman. “A mixture of Gaussians approach to mathematical portfolio oversight: the EF3M algorithm”. In: *Quantitative Finance* 14.5 (2014), pp. 913–930. ISSN: 14697696. DOI: 10.1080/14697688.2013.861075.

[LLC18] Tick Data LLC. *Global Futures Trade and Quote Data File Format Document, Version 1.6.* 2018. URL: `https://s3-us-west-2.amazonaws.com/tick-data-s3/pdf/Futures_File_Format_Guide.pdf`.

[Lóp18] Marcos López de Prado. *Advances in Financial Machine Learning.* Wiley, 2018, p. 366. ISBN: 1119482089 9781119482086. URL: `https://dl.acm.org/citation.cfm?id=3217448`.

[LY97] Kin Lam and H.C. Yam. “CUSUM Techniques for Technical Trading in Financial Markets”. In: *Financial Engineering and Japanese Markets* 4 (1997), pp. 257–274.

[NAT16] NATLAT. *Scanning hyperspace: how to tune machine learning models.* [Online; accessed March 18, 2019]. 2016. URL: `https://cambridgecoding.files.wordpress.com/2016/03/gridsearch_cv.png`.

[Nor18] Norena, Sebastian. *Python Model Tuning Methods Using Cross Validation and Grid Search.* [Online; accessed March 18, 2019]. 2018. URL: `https://medium.com/@sebastiannorena/some-model-tuning-methods-bfef3e6544f0`.

[sci19] scikit learn. *Precision and recall.* [Online; accessed March 18, 2019]. 2019. URL: `https://scikit-learn.org/0.16/modules/generated/sklearn.grid_search.GridSearchCV.html`.

[Sin19] Singh, Ashutosh. Joubert, Jacques. *Capstone1.* [Online; accessed March 18, 2019]. 2019. URL: `https://github.com/hudson-and-thames`.

[TH00] Ané Thierry and Geman Hélyette. "Order Flow, Transaction Clock, and Normality of Asset Returns". In: 55.5 (2000), pp. 2259–2284.

[WC06] Jar-long Wang and Shu-hui Chan. "Stock market trading rule discovery using two-layer bias decision tree". In: *Expert Systems with Applications* 30.1 (2006), pp. 605–611. DOI: 10.1016/j.eswa.2005.07.006.

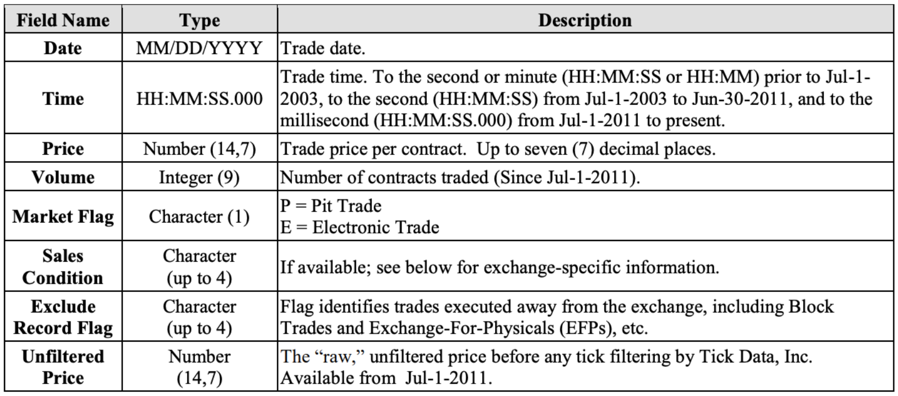
[Wik19] Wikipedia, the free encyclopedia. *Precision and recall.* [Online; accessed March 18, 2019]. 2019. URL: https://en.wikipedia.org/wiki/Precision_and_recall#/media/File:Precisionrecall.svg.

Appendix A. Data

High quality tick data has been sourced from Tick Data LLC at the cost of approximately 750 USD. The focus for our research will be on S&P 500 E-mini futures, for the period 01 July 2011 to 13 February 2019. The S&P 500 E-Mini futures data is the set which de Prado regularly references in his work and by using the same set we create a natural way to benchmark our implementations.

Tick Datas historical futures data contains a millisecond time stamp in the format (HH:MM:SS.000) and includes additional fields as shown in table 1.

Table A.3: Field Descriptions



Field Name	Type	Description
Date	MM/DD/YYYY	Trade date.
Time	HH:MM:SS.000	Trade time. To the second or minute (HH:MM:SS or HH:MM) prior to Jul-1-2003, to the second (HH:MM:SS) from Jul-1-2003 to Jun-30-2011, and to the millisecond (HH:MM:SS.000) from Jul-1-2011 to present.
Price	Number (14,7)	Trade price per contract. Up to seven (7) decimal places.
Volume	Integer (9)	Number of contracts traded (Since Jul-1-2011).

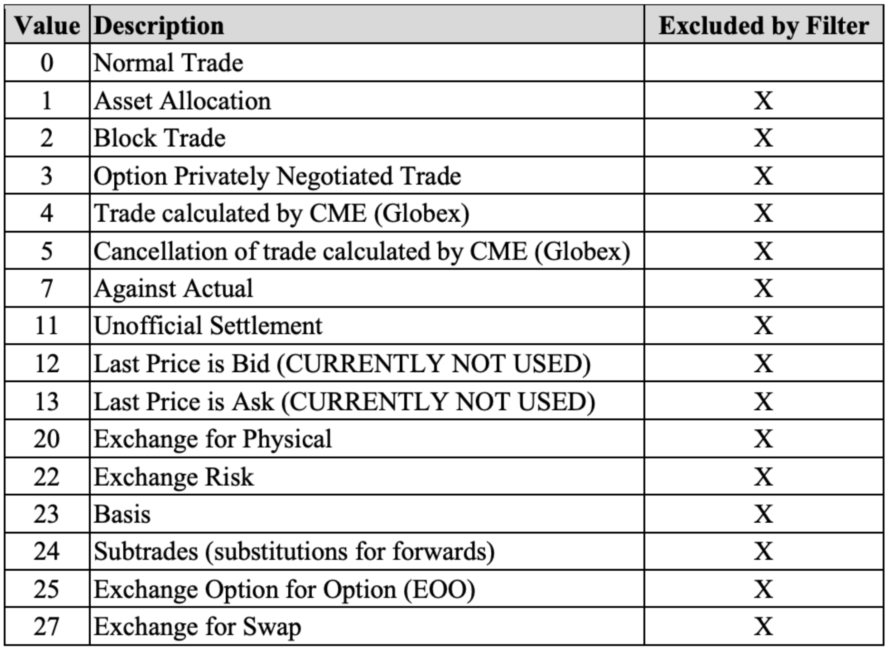
Market Flag	Character (1)	P = Pit Trade
		E = Electronic Trade

Sales Condition	Character (up to 4)	If available; see below for exchange-specific information.
Exclude Record Flag	Character (up to 4)	Flag identifies trades executed away from the exchange, including Block Trades and Exchange-For-Physicals (EFPs), etc.
Unfiltered Price	Number (14,7)	The "raw," unfiltered price before any tick filtering by Tick Data, Inc. Available from Jul-1-2011.

LLC 2018

The following futures sales condition codes are included with the data and we filter transactions to include only the zero flag for normal trades.

Table A.4: Condition Codes



Value	Description	Excluded by Filter
0	Normal Trade	
1	Asset Allocation	X
2	Block Trade	X
3	Option Privately Negotiated Trade	X
4	Trade calculated by CME (Globex)	X
5	Cancellation of trade calculated by CME (Globex)	X
7	Against Actual	X
11	Unofficial Settlement	X
12	Last Price is Bid (CURRENTLY NOT USED)	X
13	Last Price is Ask (CURRENTLY NOT USED)	X
20	Exchange for Physical	X
22	Exchange Risk	X
23	Basis	X
24	Subtrades (substitutions for forwards)	X
25	Exchange Option for Option (EOO)	X
27	Exchange for Swap	X

LLC 2018

Appendix A.1. TickWrite7 Settings

TickWrite is the software offered by TickData LLC to create files of tick data.

We made use of the following settings:

1. Include pit trades = false
2. Output contract = 0
3. Extract type = collector subtype fut trades
4. Requested data = 33—ES[20190213]
5. Start date = 08/01/2011
6. End date = 02/01/2019
7. Collection type = collection type.futures
8. Include header row = true
9. Selected fields = date and time, price, volume
10. Date format = yyyy/MM/dd
11. Output time zone = GMT
12. Roll method = Autoroll
13. Sessions = day, night, electronic

A job file is saved to the research repo (`https://github.com/hudson-and-thames/research/tree/master/Tick-Data-Notes`), which can be imported into TickWrite7 to generate the same results.

Appendix A.2. Additional Questions

Appendix A.2.1. Does the data include GLOBEX, electronic or the overnight sessions?

Data for electronic and overnight sessions is included for all applicable markets beginning Jul-1-2003. Prior to then, our futures data contains only day session data.

Appendix A.2.2. Is the E-mini data 24 hours?

The E-mini data, futures contracts that represent a fraction of the value of normal contracts, contains the full 24-hour session, with volume, beginning on Jul-1-2003. Prior to then, it begins each day at 12:00 a.m. and ends at 3:15 p.m. Central time.

We made sure to time stamp all of the data according to GMT time.

Appendix A.2.3. What is the roll method?

There are a variety of roll methods however the recommended method for S&P500 E-mini Futures is Most Active AutoRoll. At a later stage we would like to apply the ETF trick (pg 33, Advances in Financial Machine Learning).

Appendix A.2.4. What about trades that happened off exchange?

Trades that did not occur on the exchange, such as Exchange-For-Physicals (EFPs) and block trades are excluded by default.

From Jul-1-2011 on, these trades are in the data, though they are excluded from output by default.

Appendix B. Meta-Labeling Explained

Appendix B.0.1. Lopez de Prado's Original idea

The following section is taken directly from the de Prado and left in its original form as to make sure no errors are introduced by means of interpretation.

Advances in Financial Machine Learning, Chapter 3, page 50. Reads:

Suppose that you have a model for setting the side of the bet (long or short). You just need to learn the size of that bet, which includes the possibility of no bet at all (zero size). This is a situation that practitioners face regularly. We often know whether we want to buy or sell a product, and the only remaining question is how much money we should risk in such a bet. We do not want the ML algorithm to learn the side, just to tell us what is the appropriate size. At this point, it probably does not surprise you to hear that no book or paper has so far discussed this common problem. Thankfully, that misery ends here.

I call this problem meta-labeling because we want to build a secondary ML model that learns how to use a primary exogenous model.

The ML algorithm will be trained to decide whether to take the bet or pass, a purely binary prediction. When the predicted label is 1, we can use the probability of this secondary prediction to derive the size of the bet, where the side (sign) of the position has been set by the primary model.

How to use Meta-Labeling

Binary classification problems present a trade-off between type-I errors (false positives) and type-II errors (false negatives). In general, increasing the true positive rate of a binary classifier will tend to increase its false positive rate. The receiver operating characteristic (ROC) curve of a binary classifier measures the cost of increasing the true positive rate, in terms of accepting higher false positive rates.

The image illustrates the so-called confusion matrix. On a set of observations, there are items that exhibit a condition (positives, left rectangle), and items that do not exhibit a condition (negative, right rectangle). A binary classifier predicts that some items exhibit the condition (ellipse), where the TP area contains the true positives and the TN area contains the true negatives. This leads to two kinds of errors: false positives (FP) and false negatives (FN). Precision is the ratio between the TP area and the area in the ellipse. Recall is the ratio between the TP area and the area in the left rectangle. This notion of recall (aka true positive rate) is in the context of classification problems, the analogous to power in the context of hypothesis testing. Accuracy is the sum of the TP and TN areas divided by the overall set of items (square). In general, decreasing the FP area comes at a cost of increasing the FN area, because higher precision typically means fewer calls, hence lower recall. Still, there is some combination of precision and recall that maximizes the overall efficiency of the classifier. The F1-score measures the efficiency of a classifier as the harmonic average between precision and recall.

Meta-labeling is particularly helpful when you want to achieve higher F1-scores. First, we build a model that achieves high recall, even if the precision is not particularly high. Second, we correct for the low precision by applying meta-labeling to the positives predicted by the primary model.

Meta-labeling will increase your F1-score by filtering out the false positives, where the majority of positives have already been identified by the primary model. Stated differently, the role of the secondary ML algorithm is to determine whether a positive from the primary (exogenous) model is true or false. It is not its purpose to come up with a betting opportunity. Its purpose is to determine whether we should act or pass on the opportunity that has been presented.

Additional uses of Meta-Labeling

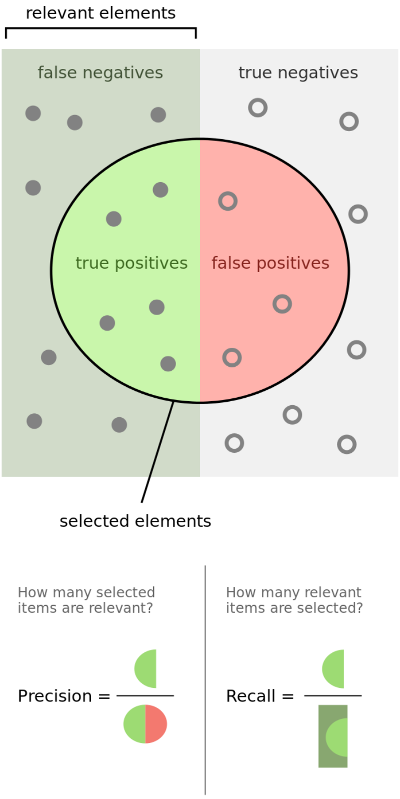


Figure B.25: Precision & Recall
Wikipedia, the free encyclopedia 2019

Meta-labeling is a very powerful tool to have in your arsenal, for four additional reasons. First, ML algorithms are often criticized as black boxes.

Meta-labeling allows you to build an ML system on top of a white box (like a fundamental model founded on economic theory). This ability to transform a fundamental model into an ML model should make meta-labeling particularly useful to quantamental firms. Second, the effects of overfitting are limited when you apply meta-labeling, because ML will not decide the side of your bet, only the size. Third, by decoupling the side prediction from the size prediction, meta-labeling enables sophisticated strategy structures. For instance, consider that the features driving a rally may differ from the features driving a sell-off. In that case, you may want to develop an ML strategy exclusively for long positions, based on the buy recommendations of a primary model, and an ML strategy exclusively for short positions, based on the sell recommendations of an entirely different primary model. Fourth, achieving high accuracy on small bets and low accuracy on large bets will ruin you. As important as identifying good opportunities is to size them properly, so it makes sense to develop an ML algorithm solely focused on getting that critical decision (sizing) right. In my experience, meta-labeling ML models can deliver more robust and reliable outcomes than standard labeling models.

Appendix B.0.2. Toy Example

To illustrate the concept we made use of the MNIST data set to train a binary classifier on identifying the number 3, from a set that only includes the digits 3 and 5. The reason for this is that the number 3 looks very similar to 5 and we expect there to be some overlap in the data, i.e. the data are not linearly separable. Another reason we chose the MNIST dataset to illustrate the concept, is that MNIST is a solved problem and we can witness improvements in performance metrics with ease.

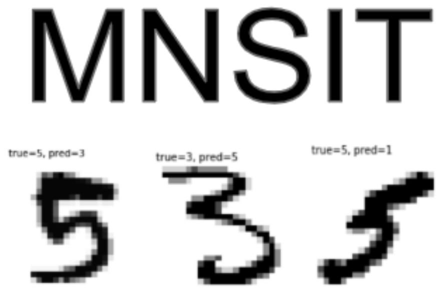


Figure B.26: Handwritten 5 and 3

Model Architecture

The following image explains the model architecture. The first step is to train a primary model (binary classification) with a high recall. Second a threshold level is determined at which the primary model has a high recall, ROC curves could be used to help determine a good level. Third the features from the first model are concatenated with the predictions from the first model, into a new feature set for the secondary model. Meta-labels are used as the target variable in the second model. Now fit the second model. Fourth the prediction from the secondary model is combined with the prediction from the primary model and only where both are true, is your final prediction true. I.e. if your primary model predicts a 3 and your secondary model says you have a high probability of the primary model being correct, is your final prediction a 3, else not 3.

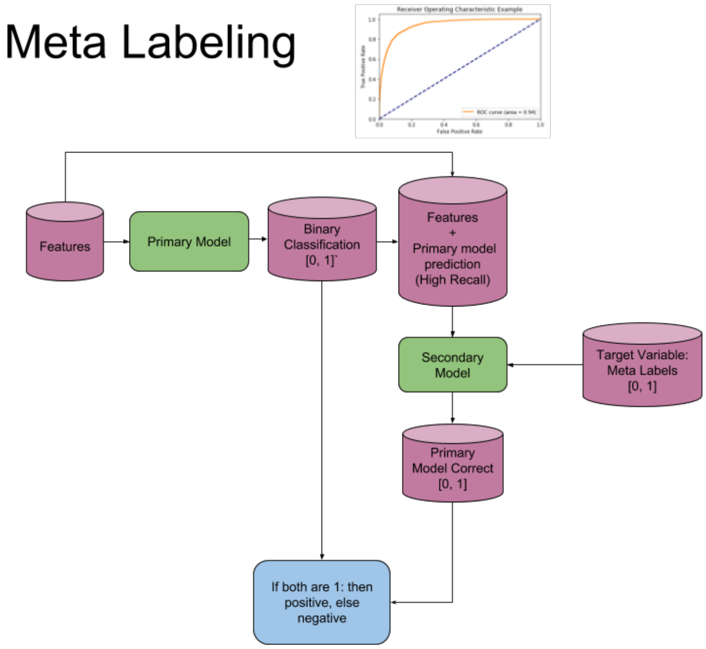

Meta Labeling
Receiver Operating Characteristic Example
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.94)
Features
Primary Model
Binary Classification [0, 1]
Features + Primary model prediction (High Recall)
Secondary Model
Target Variable: Meta Labels [0, 1]
Primary Model Correct [0, 1]
If both are 1: then positive, else negative

Figure B.27: Meta-Label Model Architecture

Build Primary Model with High Recall

The first step is to train a primary model (binary classification). For this we trained a logistic regression, using the keras package. The data are split into a 90% train, 10% validation. This allows us to see when we are over-fitting.

Second a threshold level is determined at which the primary model has a high recall, ROC curves could be used to help determine a good level. A high recall means that the primary model captures the majority of positive samples even if there are a large number of false positives. The meta-model will correct this by reducing the number of false positives and thus boosting all performance metrics.

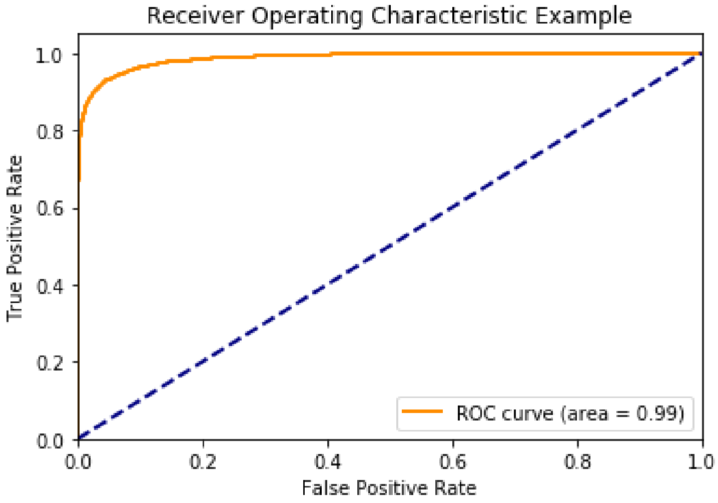


Figure B.28: Receiver Operating Characteristic (ROC) Curve

Build Meta-Model

Third the features from the first model are concatenated with the predictions from the first model, into a new feature set for the secondary model. Meta-labels are used as the target variable in the second model. Now fit the second model.

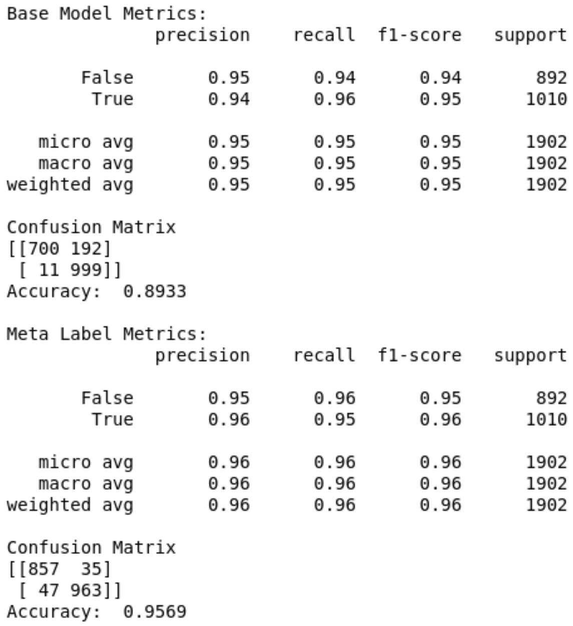
Meta-labels are defined as: If the primary model's predictions matches the actual values, then we label it as 1, else 0. In this example we said that if an observation was a true positive or true negative then label it as 1(i.e. the model is correct), else 0 (the model in incorrect). Note that because it is categorical, we have to add One Hot Encoding.

Evaluate Performance

Fourth the prediction from the secondary model is combined with the prediction from the primary model and only where both are true, is your final prediction true. e.g. if your primary model predicts a 3 and your secondary model says you have a high probability of the primary model being correct, is your final prediction a 3, else not a 3.

The section below shows the performance of the primary model vs the performance of using Meta-labeling, on out-of-sample data. Notice how the performance metrics improve.

We can see that in the confusion matrix, that the false positives from the primary model, are now being correctly identified as true negatives with the help of meta-labeling. This leads to a boost in performance metrics. Meta-labeling works as advertised!



```
Base Model Metrics:
              precision    recall  f1-score   support

       False       0.95      0.94      0.94       892
        True       0.94      0.96      0.95      1010

   micro avg       0.95      0.95      0.95      1902
   macro avg       0.95      0.95      0.95      1902
weighted avg       0.95      0.95      0.95      1902

Confusion Matrix
[[700 192]
 [ 11 999]]
Accuracy:  0.8933

Meta Label Metrics:
              precision    recall  f1-score   support

       False       0.95      0.96      0.95       892
        True       0.96      0.95      0.96      1010

   micro avg       0.96      0.96      0.96      1902
   macro avg       0.96      0.96      0.96      1902
weighted avg       0.96      0.96      0.96      1902

Confusion Matrix
[[857  35]
 [ 47 963]]
Accuracy:  0.9569
```

Figure B.29: Meta-Labeling Performance Metrics