Project Milestone (6.862, Fall 2020)
Predicting Returns in the US Equity Markets

Michele Marinucci Raphael Bakobza Harry Donnelly

Abstract

This paper builds upon a Machine Learning contest proposed by Qube Research Technologies (QRT), a quantitative hedge fund. Using historical data in the US equity market spanning a period of 20 days across different dates, we attempted to predict the one-day horizon stock return. More specifically, we built a classification model for the sign of such stock returns. We evaluated the performance of our models through a score based on accuracy (i.e. percentage of correct predictions). QRT provides a testing data base as well as a benchmark performance with a score of 51.31%. After engineering the right features, fitting a wealth of machine learning models and tuning their hyperparameter following state-of-the-art techniques, our final model beats QRT's benchmark though a random forest yielding an accuracy of 51.43%. As a note to the reader, both QRT's benchmark model and our model may seem to be inaccurate predictors, nevertheless these results are fairly remarkable if one considers the random walk nature of stock markets. In fact, through sheer repetition and volume, a model that's right 51-52% of the times may translate into significant profits on the financial markets over time, though not necessarily beating similar trading strategies in terms of Sharpe ratio or return-risk ratio.

1. Precedent Literature

Predicting stock returns is a topic that has fascinated several scholars throughout history, and a wealth of approaches has been utilized to forecast or describe the future price path of equities as well as their corresponding returns.

Some of the earlier methodologies harnessed the power of econometrics to pinpoint factors that could be identified as determinants of future stock performance. Such early attempts led to the financial field's most infamous models, including the CAPM, the Fama-French three-factor model, and the Carhart four factor model; notably, these models included factors such as the market factor, the value factor, the size factor, and the momentum factor to attempt to explain and forecast future returns.

As this econometric research broadened and deepened by adding ever more complex and numerous factors, the technologic capacities as well as the analytics techniques expanded in parallel, and an array of novel machine learning approaches began to be applied in an endeavor to maximize predictive power. This paper will continue on that path.

Here below is a non-exhaustive list of previous papers that have likewise focused on applying machine learning techniques to predict stock returns:

- *"Predicting stock returns by classifier ensembles"*, by Tsai, Lin, Yen and Chen. This study uses machine learning techniques such as decision trees and neural networks to predict stock returns. In particular, classifier ensembles (i.e. combinations of several classifiers) have been shown to be a superior method, and this study harnesses them to analyze stock returns.
- *"Predicting stock prices using data mining techniques"*, by Al-Radaideh et Al. This study aims to help stock investors decide the best timing to buy or sell stocks based based upon knowledge arising from past prices of such stocks. Each decision taken is based upon a decision tree classifier.
- *"Deep Learning for Forecasting Stock Returns in the Cross-Section"*, by Abe and Nakayama. This paper utilizes deep learning to forecast stock returns across companies in the Japanese market and assesses the performance of the method. Results show that deep neural networks outperform shallow neural networks, and that the best networks outperform representative machine learning models.
- *"Stock market's price movement prediction with LSTM neural networks"*, by Nelson, Pereira, and Oliveira. This study investigates the usage of LSTM networks to predict future stock price trends based upon historical prices together with technical analysis indicators.
- *"Predicting global stock returns"*, by Hjalmarsson. This study delves into stock return predictability using

four common forecasting variables: the dividend-price (DP) and earnings-price (EP) ratios, the short-term interest rate, and the term spread.

We incorporate some of these ideas in our model, focusing in particular on the more modern machine learning approaches.

2. Database and Benchmark Overview

2.1. Data Description

The data upon which we built our model refers to realized, historical financial data that have been made anonymous by Qube Research. Each row of the data set corresponds to a stock, indexed between 0 and 5716. For each of these, the historical data at our disposal consist of 20-lags of residual returns and relative volume. The one-day return of a stock j on day t with price P_j^t is defined by:

$$R_j^t = \frac{P_j^t}{P_j^{t-1}} - 1$$

Residual returns correspond to the stock return adjusted for the market return. Volume at time t refers to the total amount of stocks bought or sold at t. Relative volume, noted $\mathcal{V}_j^t$ for stock j at time t, adjusts for past volume and the mean volume of other stocks. Formally:

$$\bar{V}_j^t = \frac{V^t}{median(\{V_j^{t-1}, ..., V_j^{t-20}\})}$$

$$\mathcal{V}_j^t = \bar{V}_j^t - \frac{1}{n}\sum_{i=1}^{n} \bar{V}_i^t$$

where V_j^t is the volume of stock j at time t.

For each stock, we are also given categorical variables on its sector, industry-group, industry, and sub-industry, each of them being a subset of another. We are also given dates spanning from 0 to 223. Dates have been randomized and are not meant to be understood chronologically. In addition, although we have historical information, there is no overlap in the data. The training set contains 418,595 rows for 47 columns, whereas the test set contains 198,429 rows for 47 columns. Both in the training and in the testing data base, each row contains a unique ID identifier.

Missing data occurs both in the volume and and in the return columns. They comprise roughly one forth of the rows for a total of approximately 1.5 million entries.

2.2. From Continuous to Binary

The feature we are attempting to predict is stock return, which is a continuous numerical variable. In our data set, returns have been encoded as a binary variable, as follows:

$$\mathcal{R}_j^t = \mathbb{I}[R_j^t > median(\{R_1^t, ..., R_n^t\})]$$

where $\mathbb{I}[.]$ denotes the indicator function and $\mathcal{R}_j^t$ stands for the binary encoded return of stock j at time t. In other words, conditional on a date, the top half of best performing stocks are labelled as 1's while the others as 0's. Therefore, by construction, the target output is said to be balanced: it contains 50% of 1's and 50% of 0's. This enables us to use accuracy, a fairly simple measure of performance that is defined by the number of correct predictions divided by the number of predictions.

2.3. Caveats for Estimating Out of Sample Accuracy

What differentiates the testing set is that it contains separate dates to the training data set. In other words, we do not have access to data from the dates of the stocks' return we are trying to predict. We must acknowledge this deficiency in the way we perform cross validation. Therefore, we perform a cross validation, not by splitting the training data in k folds but by splitting by dates. Since each date contains approximately the same number of data, we will be able to split our data as 75% train and 25% test.

2.4. Benchmark Overview

The model we build in the next section will be compared to the benchmark model provided by Qube Research. This model had a performance of 51.31% on the out-of-sample testing data. The benchmark constructed a feature, $Z_{t-1,j}^s$ for stock j, defined as the mean of $\mathcal{R}_j^{t-1}$ conditional on sector s. Formally,

$$Z_{t-1,j}^s = \sum_{j \in s} \mathcal{R}_j^{t-1}$$

To reduce noise and to minimise stationarity, only the latest 5-day returns and volumes are used along with the newly constructed features. For simplicity, missing values are filled with zeroes. Finally, a Random Forest model with 500 estimators and a maximum depth of 8 is fitted. A large number of trees and minimum depth have been used to prevent over-fitting. Accuracy obtained through four fold cross validation is 51.43%, whereas out-of-sample accuracy is 51.31%.

3. Preliminary Model Construction and Preliminary Results

The problem we aim to solve is classifying US equity market returns using past returns, volume measures and some company characteristics (e.g. Industry). Feeding a classification machine learning model with the raw data yields an accuracy of less than 50%, a worse than random result. The

reasons are two-fold. First, the data has a significant noise to signal ratio; and second, we are dealing with a time series at the edge of stationarity. To overcome these challenging issues, we need to carefully make use of feature selection and feature engineering. We will aim to appropriately aggregate information to extract signals and weight time lags and consequently account for the changing statistical nature of the series.

3.1. Preliminary Handling of Missing Data

There is a substantial amount of missing data in the volume and return columns, and roughly one forth of our data set contains NaN values. Filling these values with 0s as done by the benchmark may be costly and add further noise to our dataset, hence we propose an alternative method. For the rows that contain more than 15 missing values in the past returns or past volumes columns, we fill in those values by taking an average of these features over their corresponding SECTOR and DATES. The choice of 15 as well as the choice of features are arbitrary at the moment, but we come back to these choices in later sections. For the rows where we have less than 15 missing data for returns or volume, we fill NaN by taking an average of the row for the relevant feature.

3.2. Feature Engineering

We engineer features that will attempt to reduce noise and capture information that might be too sparse in the raw data. When useful, we make use of the literature to find indicators that have been known to drive returns.

3.2.1. Conditional Mean Volume

Building upon the benchmark's idea of aggregating returns conditional on the sector, we construct a similar feature on conditional sector volume $\mathcal{V}_j^{t-1}$:

$$X_{t-1,j}^{s} = \sum_{j \in s} \mathcal{V}_j^{t-1}$$

This feature was constructed with the aim to combine information on a sector level. Stocks in the same industry are usually exposed to the same industry-specific risks and often move in tandem when responding to these risks. By aggregating sector-based volume, we attempt to use this information to help classify individual stocks within specific industries.

3.2.2. 20-day Accumulated Returns

We aim to engineer a feature that represents the aggregate returns of the stocks over the past 20 days. We create this feature as follows:

$$M_j^t = \sum_{i=1}^{20} Ln(1 + \mathcal{R}_j^{t-i})$$

where $Ln(.)$ is the natural logarithm. Using this feature we aim to aggregate into a single figure some of the information contained in the previous 20 days of return, removing the need to include all the previous returns as features.

3.2.3. 20-day Absolute Returns

We construct the volume factor as follows:

$$Y_j^t = \sum_{i=1}^{20} |\mathcal{R}_j^{t-i}|$$

The motivation behind this feature was to incorporate a measure of volatility. By combining information from the previous 20 days of returns we gain an insight into the magnitude of historical returns for each particular stock.

3.2.4. Market Beta

Market Beta is an important theoretical finance construct and aims to quantify an individual stocks exposure to market risk. A stock with a higher Beta is said to be exposed to more market risk and would be expected to have greater absolute moves in comparison to the broader market index. Further details on Beta can be found in the paper by Fama and French (2004).

We aimed to construct a synthetic Beta measure by aggregating stock returns by date, taking the mean return based on this condition and then regressing each stock's return against this result. For our purposes and considering one-day returns, we assume the risk free rate is zero and as such all returns can be described as excess returns.

Hence, we construct the Market Beta factor as follows:

$$\mathcal{R}_j^t = \alpha + \beta \bar{\mathcal{R}}^t$$

where $\bar{\mathcal{R}}^t$ corresponds to the average of returns for a given date at time t.

3.3. Feature and Model Selection

We experimented with including more lags of returns and volume, however none of these approaches increased accuracy. Hence, we use only five lags for both returns and volume, as in the benchmark.

3.4. Preliminary Results

We fit a random forest with maximum depth of 8 and with 500 estimators on a data set composed of the new features along with the 5-previous day returns and volume. We also fit a Discriminant Analysis classifier, a Logistic Regression and ensemble of all these models. The ensemble is made using the stacking function of sklearn. As shown in Table 1, we manage to get an accuracy of 51.61% with the ensemble

Table 1. Preliminary results for average cross-validated accuracy across several methods

METHOD	AVERAGE ACCURACY
RANDOM FOREST	51.56%
DISCRIMINANT ANALYSIS	51.23%
LOGISTIC REGRESSION	51.17%
ENSEMBLE	51.61%

based on four fold cross validation, an increase of 30 basis points from the benchmark where a basis point is defined as a hundredth of a percent.

Given we now know that our feature engineering and selection had a positive impact on our evaluation criterion we can now attempt to tweak the hyper-parameters of our model and of our methods to deal with missing values.

4. Model Tuning and Final Results

We now use more systematic methods to increase our accuracy score. First, we perform grid search on methods of handling missing values, and then we tune hyperparameters.

4.1. NaN Value Filling

We define threshold k as the number of elements over which to take the average in order to fill values. If the filling occurs horizontally, we take an average over rows; if it occurs vertically, we take an average over features, such as SECTOR and DATES. We arbitrarily set the threshold of k to 15 and examine cross-validation accuracy when varying k. We use a Random Forest with a maximum number of estimators of 300 and a maximum depth of 8 to assess accuracy.

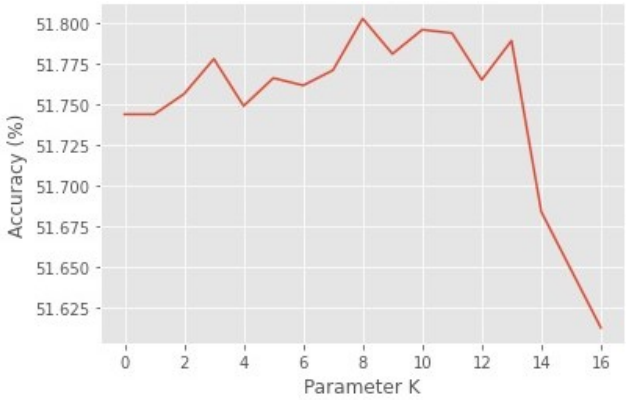

Figure 1. Grid Search on NaN Values. We look for the threshold k that will give the highest accuracy. By setting parameter to 8 we manage to achieve an accuracy of 51.80%, an increase of roughly 20 basis points from previous results.

As shown in Figure 1, by setting depth parameter to 8 we manage to achieve an accuracy of 51.80%, an increase of roughly 20 basis points. While we have used SECTOR and DATES to take an average so far, we then evaluate how other indicators such as INDUSTRY_GROUP, SUB_INDUSTRY and INDUSTRY perform while still conditioning on DATES.

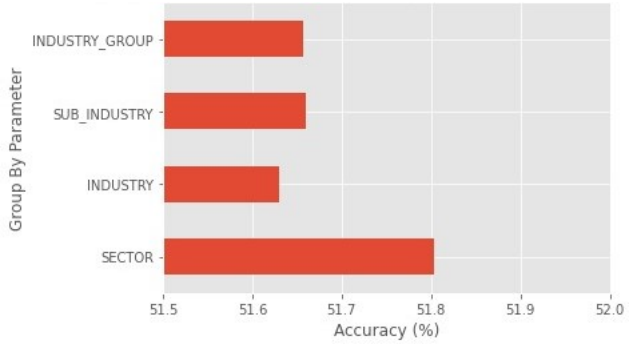

Figure 2. GridSearch on NaN Values. We change the way we take average over columns. Grouping by parameters using one feature and DATES.

As shown in figure 2, SECTOR yields the optimal result. We then take an average over a single feature and we drop the conditioning on DATE. The results are shown in Figure 3.

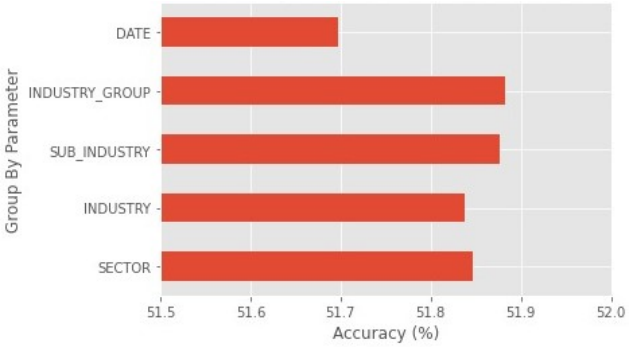

Figure 3. Grid Search on NaN Values. We change the way take we take average over columns. Grouping is made according to a single parameter. Optimal grouping appears to be INDUSTRY_GROUP, which pushes accuracy to 51.88%.

As shown in figure 3, filling NaN values vertically by INDUSTRY_GROUP gives us the best result, increasing accuracy by 10 basis points. Using this new method for filling NaN, we re-run a test on the optimal threshold k.

Setting k to 8 optimises accuracy with a score of 51.88%, as shown in Figure 4. We can now turn to hyperparameter tuning of our classification models.

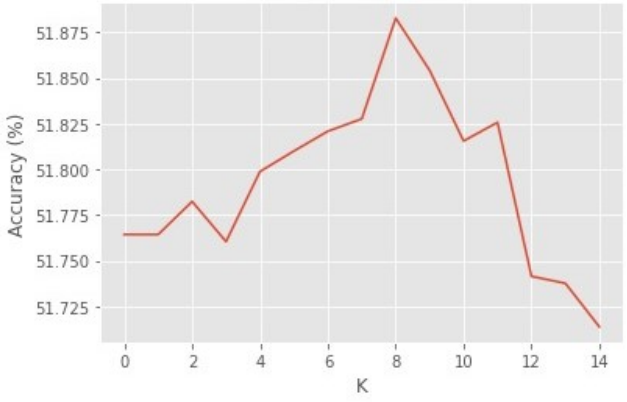

Figure 4. New K GridSearch. Filling NA values vertically by INDUSTRY_GROUP gives us the best result. Accuracy jumped by 10 basis points. Using this new method for filling NA, we redo a test on the optimal threshold K.

4.2. Hyperparameter Tuning

Given the high performance of the Random Forest compared to other models, we first perform a grid search on this model. Figures 5 and 6 detail how cross-validation accuracy changes when varying maximum depth and the number of estimators.

To identify the optimal model parameters for the other models to be included in our ensemble, we then perform a grid search across multiple different parameters for each model. Table 2 shows the number of possible combinations that each model originally had, whereas Table 3 shows the final, optimal results for each model's hyperparameters.

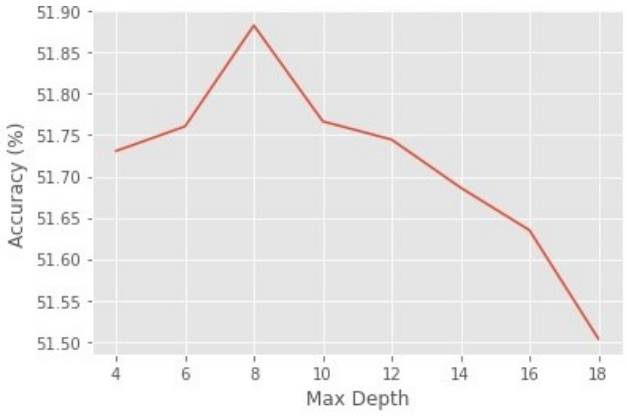

Figure 5. Random Forest Depth Grid Search. This figure clearly shows that the optimal depth of our random forest is 8.

The optimal parameters identified for each model using grid search are detailed in Table 3. Interestingly, the grid search on the Random Forest did not help us improve accuracy as we were already using optimal parameters.

Table 2. In the context of hyperparameter tuning, this table shows the total number of possible combinations for each model's hyperparameters.

MODEL	PARAMETERS	COMBOS
DECISION TREE	MIN SAMPLES MAX FEATURES MAX DEPTH CRITERION	1000
LOGISTIC REG.	PENALTY MAX ITERATIONS C	240
K-NN	WEIGHTS # NEIGHBORS LEAF SIZE	520
RANDOM FOREST	# ESTIMATORS MAX DEPTH MAX FEATURES	720

4.3. Ensemble

Prior to creating the ensemble, we performed hyperparameter optimisation on our models features using a grid search based approach as detailed above. Once we identified the best performing parameters for each model, we then created an ensemble which consisted of a combination of optimised k-Nearest Neighbours, Decision Trees, Random Forest, Naïve Bayes, and Logistic Regression models based upon argmax voting. Argmax voting sums the predicted probability of each classification from the different models, as such it assigns more weight to predictions which have a higher certainty. Using 4-fold cross validation we then fitted and tested this model to the entire feature-engineered dataset.

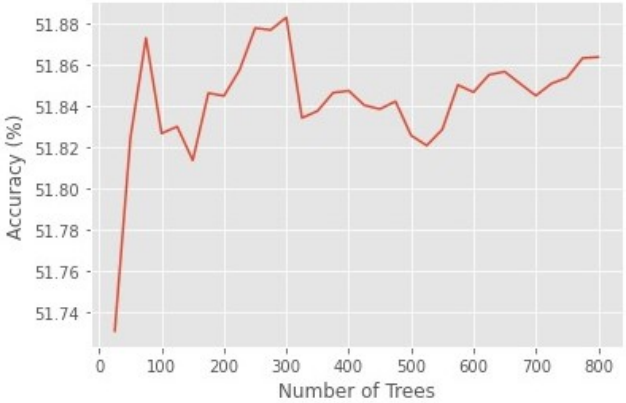

Figure 6. Random Forest Estimator Grid Search. This graph shows that the optimal number of trees in out random forest may be roughly 300.

Table 3. The optimal value for each model's hyperparameters.

MODEL	PARAMETERS	OPTIMUM
DECISION TREE	MIN SAMPLES	7
	MAX FEATURES	5
	MAX DEPTH	11
	CRITERION	ENTROPY
LOGISTIC REG.	PENALTY	L2
	MAX ITERATIONS	45
	C	0.1
K-NN	WEIGHTS	UNIFORM
	# NEIGHBORS	2
	LEAF SIZE	10
RANDOM FOREST	# ESTIMATORS	300
	MAX DEPTH	8
	MAX FEATURES	8

4.4. Auto-sklearn

In parallel to the hyperparameter tuning and the ensemble just discussed, we also utilized Auto-sklearn. Auto-sklearn is an automated machine learning package which attempts to learn the meta features of the dataset before using Bayesian Optimisation in order to decide on the most promising hyperparameters for a variety of models. It then automatically builds an ensemble using all the classifiers constructed during Bayesian Optimisation. We trained and fitted an Auto-sklearn model on both the original and feature engineered datasets, however as reported in our results in Table 4, it underperformed every other model on both datasets with the exception of K-Nearest Neighbors.

5. Final Results and Conclusion

The average cross-validated accuracy for each of the hyperpameter optimised models is shown in Table 4. The Random Forest model outperformed all other models, including both our ensemble and the Auto-sklearn ensemble.

In reference to the QRT Competition, we uploaded our predictions on the online server, and we ultimately get a final out of sample accuracy of 51.43%. This score is quite disappointing, especially when compared to our estimates. While it is difficult to assess why and how the testing accuracy differs so drastically from our cross-validation accuracy, we believe there may be two possible explanations as to what exactly has brought down our testing accuracy.

First and foremost, over-fitting of the training set may be the most reasonable explanation. While we carefully designed a cross validation system to assess our accuracy, we may have fallen prey to over-fitting. Our approach relied (perhaps excessively) on trying different new features and missing-data handling methods to evaluate what works best. By repeating this process iteratively, we may have over-fitted the particular 4 Fold validation. One hint of this is that as our accuracy increased, so did the standard deviation of our folds.

Table 4. Final results across all models.

METHOD	AVERAGE ACCURACY
RANDOM FOREST	51.88%
ENSEMBLE	51.75%
DECISION TREES	51.65%
GRADIENT BOOSTING	51.62%
DISCRIMINANT ANALYSIS	51.23%
LOGISTIC REGRESSION	51.19%
AUTO-SKLEARN	50.83%
KNN	50.07%

Secondly, missing value frequency may be another possible explanation. We spent a significant amount of time handling our missing data, primarily because they made up an important proportion of the training dataset. However, missing data on returns occurs less often in the testing set. We have perhaps overstated the effect of handling NaN values.

On a more positive note, this final submission still beat the benchmark by 10 basis points, meaning that we at least met our minimum objective as set out at the start of the project.

References

To see the code and the data we utilized, please refer to this GitHub url: https://github.mit.edu/mike-mar/ml_final_project. Please see below for an exhaustive list of all the research papers we referenced.

Tsai, Chih-Fong, et al. "Predicting Stock Returns by Classifier Ensembles." Applied Soft Computing, Elsevier, 10 Oct. 2010, www.bit.ly/3jec5PG494610002516

Al-Radaideh, Q. Et Al. (2013). Predicting stock prices using data mining techniques. Retrieved from www.bit.ly/2Gb2mLB

Abe, M. (2018, June 3). Deep Learning for Forecasting Stock Returns in the Cross-Section. SpringerLink. www.bit.ly/3cHQfkS

Nelson, D., et Al. (2017, May 1). Stock market's price movement prediction with LSTM neural networks - IEEE Conference Publication. https://www.bit.ly/2GaEAPT

Hjalmarsson, E. (2010). Predicting Global Stock Returns. The Journal of Financial and Quantitative Analysis, 45(1), 49-80. Retrieved December 1, 2020, from

www.bit.ly/34b0jiz

Fama, E. F. (2004). The Capital Asset Pricing Model: Theory and Evidence. American Economic Association. https://bit.ly/33vPJmU

Feurer, M. (2020). Auto-sklearn. ResearchGate. https://www.researchgate.net/publication/342801746_Auto-Sklearn_20_The_Next_Generation