Feature Engineering and Predictive Modeling for Financial Time Series Data

September 10, 2019

Contents

Abstract

This paper presents a series of results achieved by attempting to predict the closing price of a financial market time series. The data source was Yahoo Finance, containing 19 years worth of daily stock information, and an ETF tracking the Nasdaq 100 index (QQQ) was chosen as the specific dataset to predict. A set of features for financial time series data was engineered and five (5) models were explored: Random Forest, LeNet, custom LeNet, ResNet, and a LSTM. The goal was to use to Deep Neural Network models, while engineering features that allow reliable prediction with only a single day worth of information. The best performing models managed to achieve accuracy ranges between 83-85%, showing that the features engineered have predictive value.

1 Introduction

Financial Time Series forecasting is a problem that has been studied for as long as there has been financial markets. In this specific work, Time Series Classification is being tackled. In contrast to the traditional regression problem, a classification about the future outcome in binary format is attempted. The specific target was the closing price change between current day's close, and previous day's close.

$$\text{Closing value change} = \Delta(C, t) = \frac{C_t - C_{t-1}}{C_{t-1}} \tag{1}$$

t: time index
C: Closing Price

In addition to attempting an approach which is different than the traditional regression problem for time series forecasting, an alternative to a strict Deep Learning approach [1] was also pursued. This alternate solution involved engineering features for a specific day worth of information and feeding these features to a Deep Neural Network.

In order to evaluate the features and restrict the scope of this project, feature engineering was applied to only one day worth of information. Ideally, the same method can be applied to multiple days, and models can be explored accordingly.

2 Background

In this section, further background information about the problem, data and tools used will be provided. The goal is to allow the reader to have a more refined picture of the work and its challenge.

2.1 General Problem

As mentioned, the problem tackled in this project is a Time Series Classification. It is a challenging problem that has been explored in the areas of Machine Learning and Data Mining [6].

For this specific project, the attempt is to produce a classifier that is able to determine whether the closing value described by equation 1 will be greater than 0.025. If the closing value is > 0.025 than label 1, else, label 0.

An additional note, is that in order to build this classifier, the only information that can be used from the day t is the opening price of the stock (or index). While the opening price is on day t, it is a constant value that traders cannot change over the course of the trading day. More information about this can be found in section 2.2 and figure 2. Consequently, it is a value that can be used for trades that are placed at the end of the trading day (or in the course of the day for that matter).

2.2 Data Description

The data used for this work was extracted from Yahoo Finance website. There, it is possible to search and download historical financial data for free, where one is able to stipulate the start and end dates for the data desired. Figure 1 shows five days worth of information that can be drawn from the dataset. The specific stock data used here was for Invesco's Exchange Traded

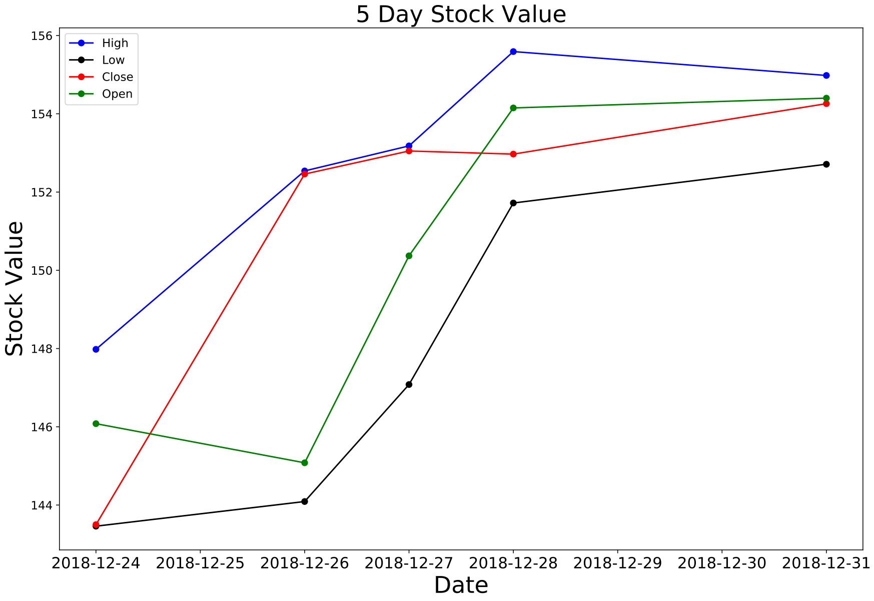


Figure 1: Five days worth of information, where High, Low, Close, and Open prices are displayed. For any prediction related to Close price on day t, only the Open price on day t can be used.

From this data, it is possible to attain information about daily High, Low, Open, Close, Adj Close, and Volume. The reader may refer to table 2.2 in order to find a definition for each of column names mentioned.

Open:	Price in which the stock started the trading day
Low:	Lower price achieved in that specific trading day
High:	Highest price achieved in that specific trading day
Close:	Price at which the stock closed for day t
Adj Close:	Closing price adjusted to stock splits
Volume:	Number of shares traded

Table 1: Description of Yahoo data columns

In addition to a definition of the various columns involved in the data, the specific variables that are used to compose the features that are eventually used for prediction are also highlighted in figure 2 (with the exception of Volume and Adj Close). The only information used from the day t of prediction is the Open price O_t. All other "raw" variables (meaning variables that are taking directly from the original dataset), are from the previous day.

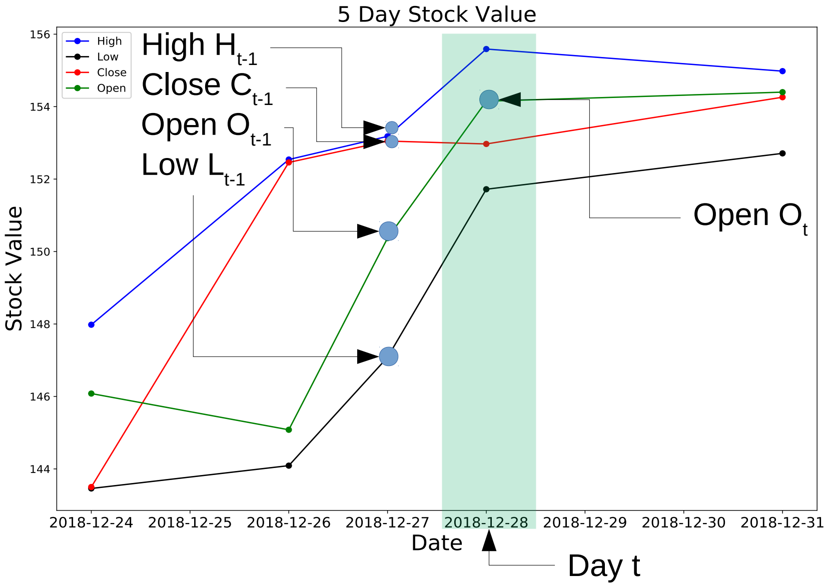
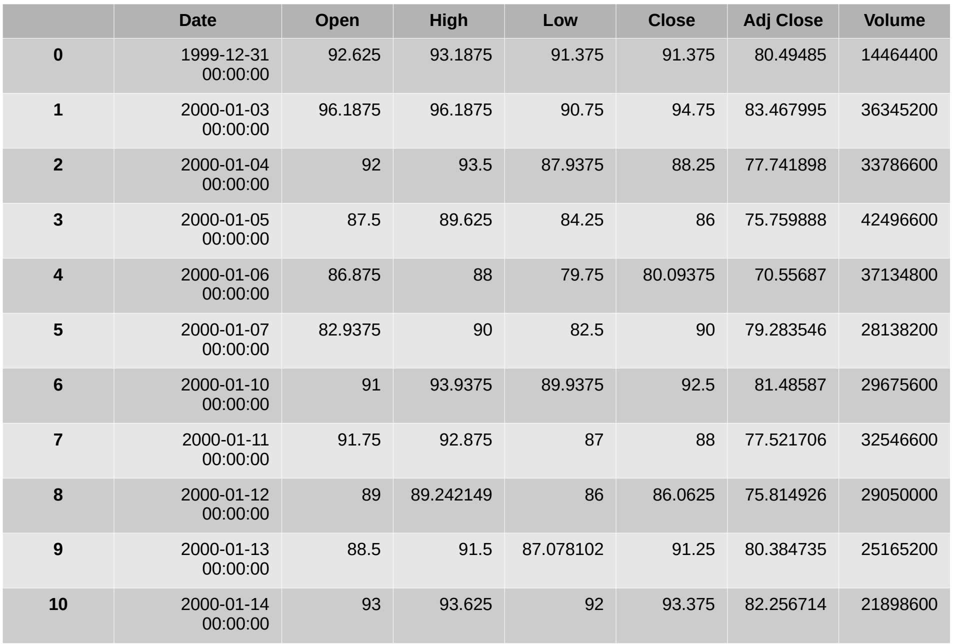


Figure 2: Five days worth of information, where the variables used for prediction are highlighted in the graph.

In order to further illustrate what the table that was initially acquired contains, please see figure 3.



	Date	Open	High	Low	Close	Adj Close	Volume
0	1999-12-31 00:00:00	92.625	93.1875	91.375	91.375	80.49485	14464400
1	2000-01-03 00:00:00	96.1875	96.1875	90.75	94.75	83.467995	36345200
2	2000-01-04 00:00:00	92	93.5	87.9375	88.25	77.741898	33786600
3	2000-01-05 00:00:00	87.5	89.625	84.25	86	75.759888	42496600
4	2000-01-06 00:00:00	86.875	88	79.75	80.09375	70.55687	37134800
5	2000-01-07 00:00:00	82.9375	90	82.5	90	79.283546	28138200
6	2000-01-10 00:00:00	91	93.9375	89.9375	92.5	81.48587	29675600
7	2000-01-11 00:00:00	91.75	92.875	87	88	77.521706	32546600
8	2000-01-12 00:00:00	89	89.242149	86	86.0625	75.814926	29050000
9	2000-01-13 00:00:00	88.5	91.5	87.078102	91.25	80.384735	25165200
10	2000-01-14 00:00:00	93	93.625	92	93.375	82.256714	21898600

Figure 3: Data retrieved from Yahoo Finance for Invesco's QQQ

2.3 Tools Used

The tools used to execute this project fall in two different categories: Programming libraries, and Interactive Development Environments (IDEs).

The language of choice was Python, and the main libraries include: Pandas, Numpy, Pytorch, Skorch, and Matplotlib. The IDE of choice was Pycharm. A secondary development environment that was also used was Jupyter Notebooks, to display and prototype algorithms.

3 Methods

In this section the algorithms that were used to develop the features will be explained. In addition, the machine learning models that will process those features will also be exposed, along with a justification for each of these models.

3.1 Feature Engineering

The main idea was to create features that help describe a specific day for the machine learning model. *The features engineered can be separated in two* *classes: Analytical features, and Transform features.*

3.1.1 Analytical Features

Analytical features were features that were created from applying functions that produce insight into the relation of between different components in the dataset. For example: $\frac{O_t - O_{t-1}}{O_{t-1}}$ is illustrate the change between day t open price O_t and the previous day's open price.

In order to be more general, a general function that evaluates equation 2 was created to produce these features, herein called *change features*:

$$\Delta(D, R) = \frac{D - R}{R} \tag{2}$$

D:	Input data to evaluate
R:	Reference data to compare against

Table 2: Variables composing equation 2

The different features produced by applying equation 2 create ratios of D with respect to (wrt) R. The result is described in table 3:

Open day change:	$\Delta(O_t, O_{t-1})$
Next day open change	$\Delta(O_{t+1}, O_t)$
Open change wrt Close	$\Delta(O_t, C_{t-1})$
Next day open change wrt Close	$\Delta(O_{t+1}, C_t)$
Open change wrt High	$\Delta(O_t, H_{t-1})$
Next day open change wrt High	$\Delta(O_{t+1}, H_t)$
Open change wrt Low	$\Delta(O_t, L_{t-1})$
Next day open change wrt Low	$\Delta(O_{t+1}, L_t)$
Open change wrt Volume	$\Delta(O_t, V_{t-1})$
Next day open change wrt Volume	$\Delta(O_{t+1}, V_t)$
Close change	$\Delta(C_t, C_{t-1})$
High change	$\Delta(H_t, H_{t-1})$
Low change	$\Delta(L_t, L_{t-1})$
Volume change	$\Delta(V_t, V_{t-1})$
High Low range	$H_t - L_t$

Table 3: Features created by applying equation 2. Note that features are mentioned in terms of the next day, denoting the prediction day to be $t + 1$

The set of features described in table 3 highlights the main analytical features developed for the experiments realized in this project. these features are general features that also seen in what is called ”Technical Stock Analysis”. The main idea behind this industry standard is to create information solely based on the movement of the stock price over time.

3.1.2 Transform Features

The transform features build on the ideas developed by Joseph Fourier and the Discrete Fourier Transform (DFT) [5]. DFTs, as defined in equation 3, and inverse DFT as defined in equation 4, are used extensively in the areas of signal and image processing. The main idea is to translate the information, either a signal (or time related sequence) or image (spatial based data) to an alternative format for analysis. This alternative format highlights changes and patterns that emerge over the domain of reference (time for signals or time series data, spatial for images).

$$X[k] = \sum_{n=0}^{N-1} \left\{ x[n] \cdot e^{\frac{j2\pi}{N} \cdot kn} \right\} = \sum_{n=0}^{N-1} \left\{ x[n] \cdot (\cos(\frac{j2\pi}{N} \cdot kn) + j \sin(\frac{j2\pi}{N} \cdot kn)) \right\} \tag{3}$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} \left\{ X[k] \cdot e^{\frac{j2\pi}{N} \cdot kn} \right\} = \frac{1}{N} \sum_{k=0}^{N-1} \left\{ X[k] \cdot (\cos(\frac{j2\pi}{N} \cdot kn) + j \sin(\frac{j2\pi}{N} \cdot kn)) \right\} \tag{4}$$

For most practical purposes, what is actually used is the Fast Fourier Transform (FFT) [5], which is an algorithm that applies the DFT in a more efficient manner [7]. See Dagupta’s book on Algorithms for further details [7].

The end goal is then to apply the FFT in the set of analytical features, so to produce an alternative perspective on that information. In essence, an array of features $\mathbf{x}_A$ containing only analytical features is seen as a one dimensional image.

As one can infer from equation 4, what will be achieved is a sequence that will be the same length, say N, as the length of the input sequence. Therefore $|\mathbf{x}_A| = |\mathbf{x}_{FFT}| = N$.

In terms of equations 3 and 4, $|\mathbf{X}| = |\mathbf{x}| = N$, where

$$\mathbf{X} = [X[0], ..., X[k], ..., X[N-1]], k \in \mathbb{Z}_{\geq 0}$$

$$\mathbf{x} = [x[0], ..., X[n], ..., X[N-1]], n \in \mathbb{Z}_{\geq 0}$$

Now, as it is possible to inspect in equation 3, the array created by the transform contains imaginary numbers. In order to simplify calculations and the optimization task task for the machine learning models [8] [1], we apply another operation which returns the absolute value of every element in $X[k]$. This allows a feature set that is within the $\mathbb{R}_{\geq 0}$ domain.

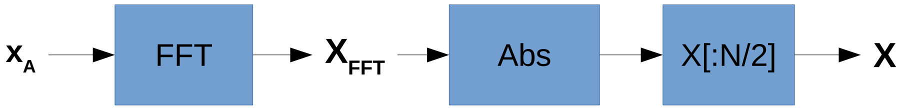


Figure 4: Block diagram illustrating the computational flow from analytical feature vector $\mathbf{x}_A$ to transform feature vector $\mathbf{X}$.

As illustrated per figure 4, first the FFT is applied, then the absolute value function is applied at every element of $\mathbf{X}_{FFT}$, finally providing a preliminary $\mathbf{X}_{abs}$. The final action is to take only the first half of the sequence $N/2$, since the values produced by the FFT repeat itself starting at π (remember the sum of complex sinusoids with π within their arguments). With that, we achieved the transform feature set $\mathbf{X}$.

3.2 Models

During the exploration of the Time Series Classification problem described in the previous sections five models were developed, in an attempt to not only reach a solution that produces relevant predictive results, but also to evaluate the performance of different models. These models were: one Random Forest (RF), three Convolutional Neural Networks (CNN), and one Long Term Short Term Memory Neural Networks (LSTMs). Over the course this section, each model will be described.

3.2.1 Input Data

The format of the input data to each model will be described here. Throughout the main experiment, the data was kept the same. This means, same features and sample size. The only different is that for th RF model, the Transform data is concatenated to the Analytical data. This is in contrast with the other models, where the input data is added to a second channel. Please see table 3.2.1.

Model Name	Training Shape	Test Shape
RF:	(522, 19)	(522, 19)
CNNs:	(522, 2, 13)	(177, 2, 13)
LSTM:	(522, 2, 13)	(177, 2, 13)

Table 4: Approximately the same data format, along with the same data entries were used for every model tested in this project.

3.2.2 Random Forest (RF)

Random Forests is a machine learning method that stems from Bagging [4]. It is a sub category of Ensemble Learning, where classification trees are built and averaged so to reduce noisy decisions. An ensemble of trees is then picked to make decisions for any input $\mathbf{x}$ to the system.

The resulting model developed in this proeject used 13421 estimators and the gini criterion to produce an estimate. For further background on RF models, please see chapter 15 in [4].

3.2.3 LeNet5 (CNN)

LetNet5 is one of the reference CNN models. It is more explicitly described in [9]. The LeNet5 architecture used here follows the one proposed in [9], along with small changes such to adapt for the specific input set at hands. Namely, instead of using two dimensional layers, one dimensional layers are used. In addition, we have dropout layers with different activation probabilities and 39 middle layers. More details can be found in the code supporting this paper.

3.2.4 AvgLeNet5 (CNN)

The AvgLenet5 CNN was a custom version of the original LeNet5 designed in this project to experiment with Average Pooling the Convolutional layers. Consequently, instead of having a number of Fully Connected layers directly after the Convolution Layers, an Average Pooling is done, along with only one Fully Connected Layer which reduces to the number of classes desired for decision making.

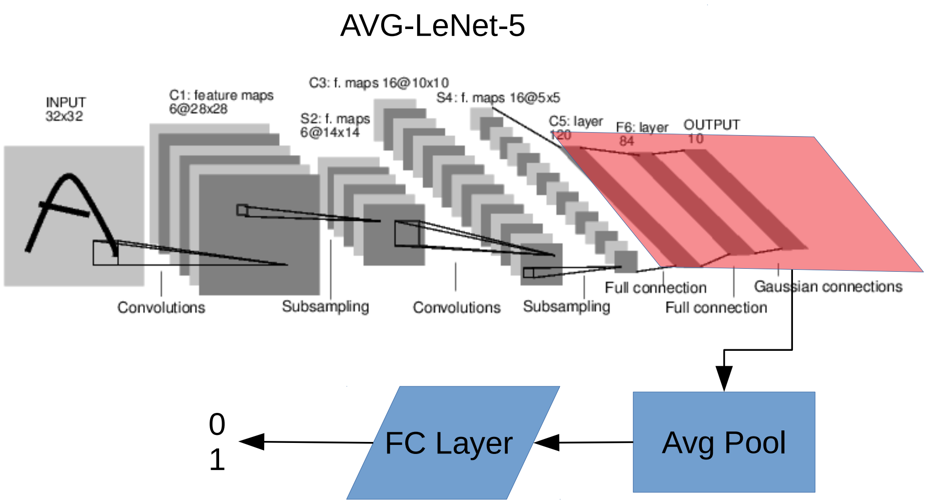


Figure 5: Architecture of the Avg-LeNet-5 designed for this project

Figure 5 attempts to clarify the architecture used. The logic behind producing this type of averaging layer, and then a decision was to attempt to further generalize the solution.

3.2.5 ResNet (CNN)

The ResNet model [1] [10] used for this work was very similar to the model developed during the course. However, it contains the dimensional differences presented by the LeNet5 model. Please see code attached to better understand the architecture. Final parameters included 64 mid channels, along with three groups of two blocks each.

3.2.6 LSTM

The LSTM model used in this project was composed a single LSTM layer which reduced the input dimension to 5. This output was then fed into a fully connected layer, which finally translated the information to two classes.

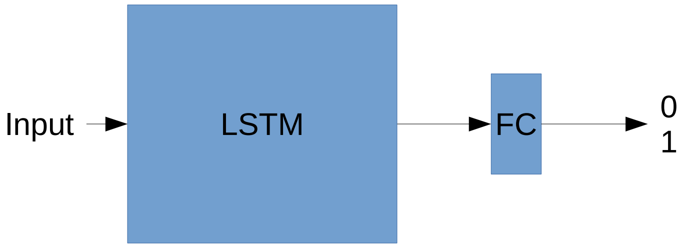
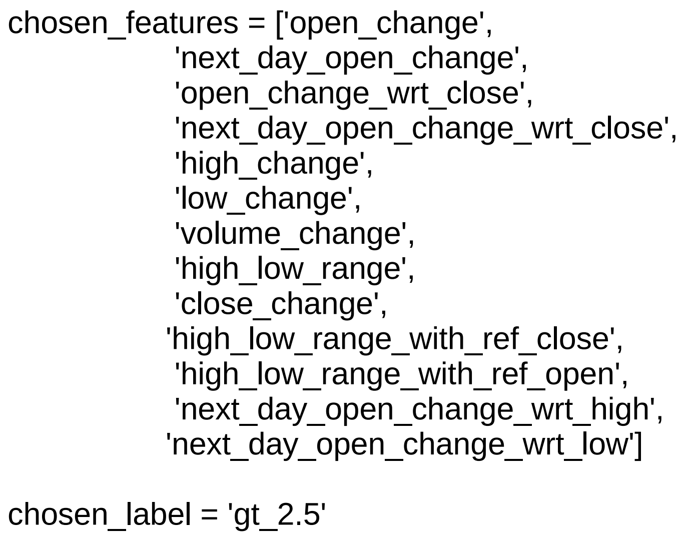


Figure 6: Simple LSTM architecture used to test features. On the final architecture, the FC layer was composed of 5 neurons.

4 Experiment and Results

The main experiment developed with this project attempted to evaluate the performance of different models with a chosen feature set. The feature set was kept constant, and was chosen after experimentation with the RF and CNN models. The final analytical feature set used can be found in the PipelineResource.py file with the variable name: "chosen_features". A copy of its definition, along with the label chosen, is presented in figure 7 :



```
chosen_features = ['open_change',
                   'next_day_open_change',
                   'open_change_wrt_close',
                   'next_day_open_change_wrt_close',
                   'high_change',
                   'low_change',
                   'volume_change',
                   'high_low_range',
                   'close_change',
                  'high_low_range_with_ref_close',
                   'high_low_range_with_ref_open',
                   'next_day_open_change_wrt_high',
                  'next_day_open_change_wrt_low']

chosen_label = 'gt_2.5'
```

Figure 7: Definition of the final analytical features used in this experiment, along with the definition of the label. Namely, we are looking for changes that are above 2.5%.

With Analytical and Transform features, the five different models were evaluated. The final training performances can be found in figure 8, while the final test performances may be found in figure 9.

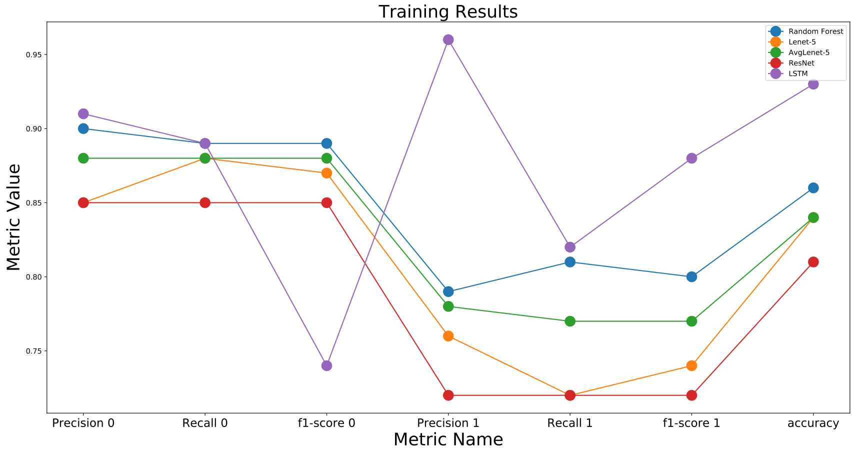


Figure 8: Results for all of the models trained. It is possible to see that the LSTM model outperforms in nearly all metrics and classes in training.

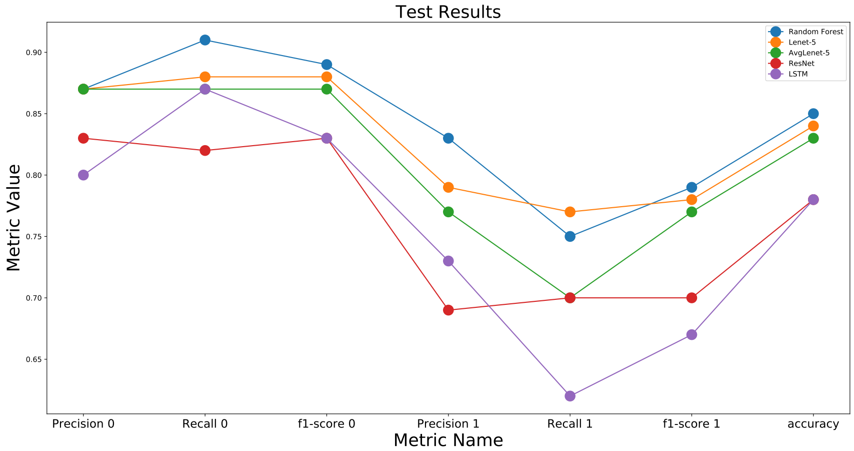


Figure 9: Test data results for all models explored. It is possible to note that the RF model outperforms in the Test data.

Figures 9 and 8 show the performance of the various models tested. The different line colors illustrate the different models tested. Different metrics were exposed so to more thoroughly evaluate the models performance.

Each model underwent its own hyperparameter optimization process, as it will be evident in the supporting jupyter notebooks.

In addition to training and test performance graphs, the confusion matrices for are also included:

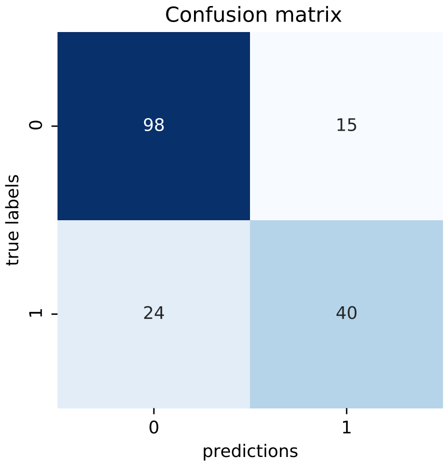


Figure 10: Confusion matrix for LSTM test results

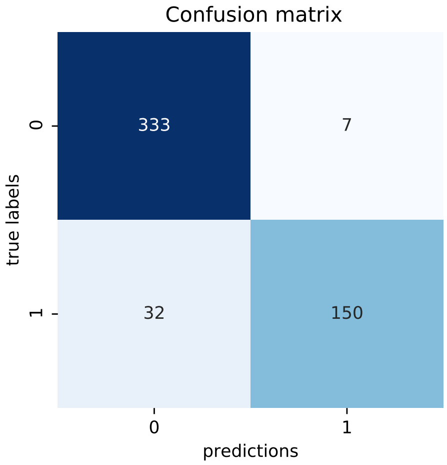


Figure 11: Confusion matrix for LSTM training results

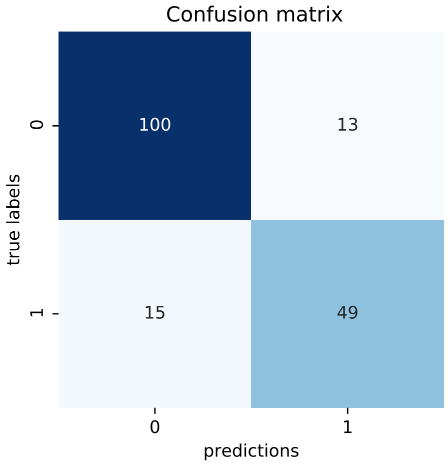


Figure 12: Confusion matrix for Lenet-5 test results

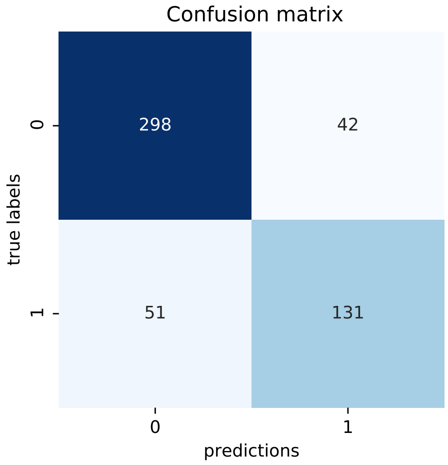


Figure 13: Confusion matrix for Lenet-5 training results

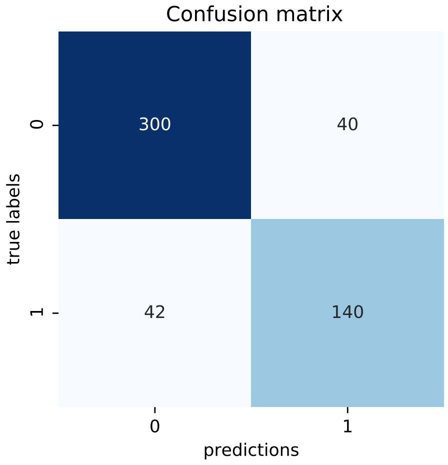


Figure 14: Confusion Matrix for AvgLenet Training Results

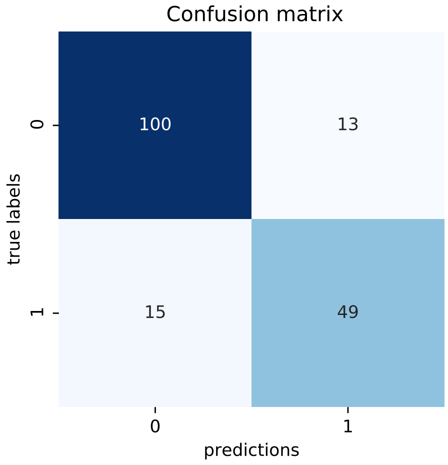


Figure 15: Confusion Matrix for AvgLenet Test Results

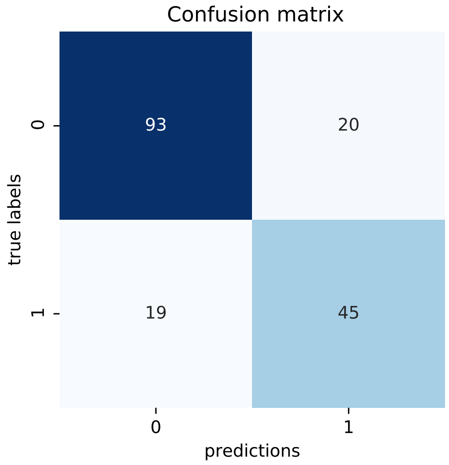


Figure 16: Confusion matrix ResNet test results

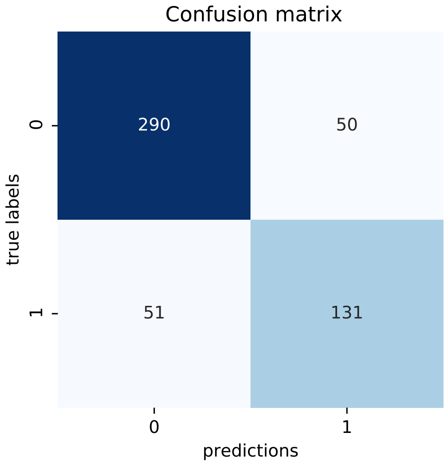


Figure 17: Confusion matrix ResNet training results

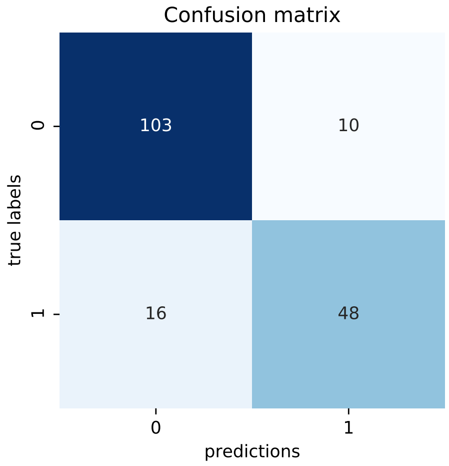


Figure 18: Confusion matrix for RF test results

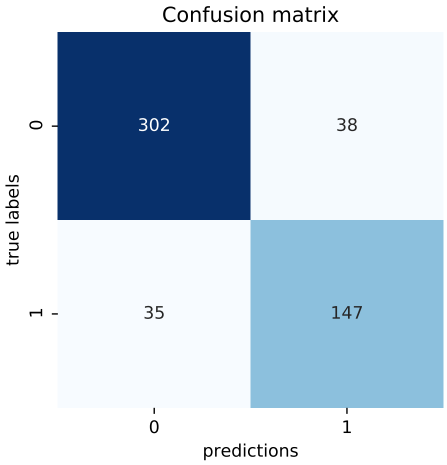


Figure 19: Confusion matrix for RF training results

Figures 10-19 show the confusion matrix of the various models tested.

5 Conclusion

The general conclusion reached for this project is that the set of features engineered contain sufficient value such to allow different models to predict the closing price of day t. Considering the scenario designed for this experiment, where we have the opening price of day t, and the opportunity to calculate the features, we are able to make predictions that range between 78% to 85% accuracy. Namely, the best performing models optimized for this problem, RF, Lenet-5 and AvgLenet-5, can reach accuracies of 85%, 84% and 83% respectively. This is significantly better than the flip of a coin or a momentum based approach.

6 Future Work

For future work in this area, it would be interesting to validate these results against other ETFs. Meaning, use same features and test the models, so to evaluate the predictive potential of the features. In addition, a logical step forward would be to start evaluating the value of using different number of days in the feature input, along with the creation of features that consider previous information, as well as other economic related information.

References

[1] Ian Goodfellow, Yoshua Bengio, A. C. (2017). The Deep Learning Book. MIT Press, 521(7553), 785. https://doi.org/10.1016/B978-0-12-391420-0.09987-X

[2] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. https://doi.org/10.1109/CVPR.2016.90

[3] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition, 114. https://doi.org/10.2146/ajhp170251

[4] Hastie, T., Tibshirani, R., & Friedman, J. (2017). The Elements of Statistical Learning The Elements of Statistical Learning. Retrieved from https://web.stanford.edu/ hastie/Papers/ESLII.pdf

[5] Oppenheim, A. V., Schafer, R. W., & Buck, J. R. (1998). Discrete-Time Signal Processing. Prentice Hall.

[6] Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., & Muller, P. A. (2019). Deep learning for time series classification: a review. Data Mining and Knowledge Discovery, 144. https://doi.org/10.1007/s10618-019-00619-1

[7] Dasgupta, S., Papadimitriou, C. H., & Vazirani, U. V. (2006). Algorithms.

[8] Boyd, S., & Vandenberghe, L. (2004). Convex Optimization. https://doi.org/10.1017/CBO9780511804441

[9] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. Proc. OF THE IEEE. Retrieved from http://ieeexplore.ieee.org/document/726791/

[10] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. https://doi.org/10.1109/CVPR.2016.90