Predicting Stock Investment Strategies

Seminar: Machine Learning in Finance ST22

Franziska Heusser: 19-705-029

Matteo Gamba: 19-753-443

Timo Bischofberger: 18-701-466

Leon Frielingsdorf: 18-712-786

Philip Smidt Andersen: 21-734-637

Spring Term 2022

University of Zurich

1 Introduction

Predicting stock market prices and the strategies attached to investing in the stock market has been an intriguing challelnge for many researchers and firms. However, to this date, no golden model has been created to accurately predict stock price movement as it is nearly impossible to model and predict the stock market dynamics (Schumaker and Chen, 2009).

In this project, we thus do not aim to exactly classify stock price movements, but we rather try to build different models with different parameters and evaluate the up- and downsides of said models and properties.

The first part of the code consists of preprocessing our data, where we create a training and test split, treat missing and null values, handle outliers and select features. In the second part of the project, we train and test models using Logistic Regression, Random Forest and Support Vector Machines. This is followed by an evaluation of the algorithms as the final part.

2 Preprocessing

Before we can start to train our algorithms, we not only need to load, merge and split (train-test) the data, but we also need to look look at it to get a grip on what we are working with and handle missing values as algorithms can be sensitive to them (Pelckmans et al., 2005). We look at the data by plotting a summary statistics, drawing some boxplots and adding 2 histograms which provide information about the balancedness of our dataset. We see that our dataset is quite imbalanced regarding the response, which we will take into consideration when we train our algorithms. For the missing values, we first set infinity values to NaN, visualise the missing values using a histogram and remove the columns of the training and test set that contain more than 20% of missing values in the training set. We then treat null values the same way, but only remove the columns which contain more than 30% null values. For null values, we set the border higher because unlike NaN for most cases, assuming MCAR (missing completely at random), null values can contain information. We hypothesise that in financial data, there are substanitally more null values than NaN which contain information (e.g., no preferred dividends, no R&D expenses, etc.). We also see an indication for this in the histogram, as there are substantially less columns containing less than 2000 null values than there are columns containing less than 2000 NaN values and the distributions vary considerably.

In order to be able to handle the outliers, because Mahalonobis Distance is sensitive to NaN, we impute the remaining missing values using an iterative imputer which models each feature containing NaN as a function of the other features. To reduce overfitting and remove a potential bias on the model, we standardise the data to accurately use Mahalonobis D, whereby we remove 944 Outliers.

This step is followed by a second imputation of the (previous) NaN values, where we fit the imputer on the training data and transform it both on the training and test data set to not base any calculations on the test set and keep it as a "surprise" for our model. Now, we standardise our freshly imputed data again to remove the bias caused by the outliers on the mean and standard deviation, and we can move on to the feature selection process. We decided to choose Random Forest as our method for feature selection where we pick the best 20, 30, and 50 features in order to compare their impact on the models. The reasoning for this is to see whether adding features leads to overfitting or an actual model that performs better.

3 Machine Learning Algorithms

We choose three main machine learning algorithms on our preprocessed data: Logistic Regression, Random Forest, and Support Vector Machines. We then also train a small (one hidden-layer) neural network for comparison. For all the algorithms, we use a pipeline, where we do a cross-validation for hyperparameter-tuning using Grid Search. Hereby we try different parameters which we derive from the respective function's description (skicit-learn). We then print the F1-score, the accuracy and the confusion matrix for the evaluation of the models.

4 Results

We decided to measure the results using three different approaches: accuracy, F1-score, confusion matrix. The reasoning for this lies in the value of each of these methods. The accuracy measures how well the model performs on the test set, meaning, what percentage of responses are classified correctly. The F1-score weighs precision and recall evenly, and thus puts a weight on the false predictions, which are linked to high costs (especially in the case of a buying prediction which should have been classified as a selling prediction). The F1-score is also a good metric in our case as we are partly evaluating models that are trained and tested on imbalanced data. Lastly, the confusion matrix allows for not only a visualised result but also for a broader understanding of the learning process of our models and hence, of their up- and downsides.

Model	**Logistic Regression**			**Random Forest**			**Support Vector Machines**			**MLP Classifier**		
Evaluation Method	Accuracy	F_1	Confusion Matrix	Accuracy	F_1	Confusion Matrix	Accuracy	F_1	Confusion Matrix	Accuracy	F_1	Confusion Matrix
Imbalanced (n=30)	0.598	0.477	2575 11 88 223 5 20 1417 6 47	0.559	0.534	1959 3 712 158 1 89 974 0 496	0.475	0.470	1663 127 884 172 10 66 970 86 414	0.559	0.543	1852 8 814 135 1 112 863 3 604
Balanced (n=30)	0.507	0.491	1942 461 271 141 82 25 954 313 203	0.555	0.533	1910 13 751 154 1 93 945 0 525	0.497	0.468	1963 274 437 186 27 35 1097 181 192			
Balanced (n=20)	0.513	0.499	1961 451 262 134 84 30 932 328 210	0.557	0.537	1906 15 753 145 2 101 931 1 538	0.370	0.395	947 392 1335 112 51 85 549 292 629			
Balanced (n=40)	0.490	0.492	1814 510 350 135 76 37 873 335 262	0.536	0.518	1842 12 820 153 2.0 93 959 0 511	0.486	0.485	1686 235 753 166 7 75 943 86 441			

Figure 1: Evaluation of the Models

Figure 1 provides an overview of all the models and their performances. In total, there are 13 models which differentiate in the balancedness and the number of features used to predict the responses.

4.1 Accuracy

In terms of accuracy, the best performing model is clearly the Logistic Regression on the imbalanced data (0.598). The underlying reason is the imbalancedness of both the training and the test data and the fact that the logistic regression predicts almost exclusively a selling strategy. Here, we see that the structures are too difficult to learn for the model and the logistic regression thus learns to classify the strategy that appears the most often. For the Logistic Regression using the balanced data, we underline the previous sentence as the model becomes less accurate for a higher number of features and more accurate for a lower number of features. What is interesting to see, the accuracy of the model that uses Support Vector Machines is very low ($0.370 - 0.497$). Unlike the other models, the SVM model actually learns to include the holding strategy in its prediction even for the imbalanced data set. However, the performance in this regard is not too good, maybe because the test set is just as imbalanced as the training set and because 20 features are just too few for the SVM to successfully predict the response.

4.2 F_1-Score

The best F_1-score is achieved by the neural network (0.559) as this achieves the best precision and recall on average. Almost equally as successful in this manner scores the Random Forest model, which also achieves quite a high accuracy. The models using SVM and Logistic Regression both achieve lower scores. On the one hand, for the Logistic Regression, the F_1-score underlines that the high accuracy stems from the selling strategy which is most often predicted, but that the model fails to predict the other strategies. On the other hand, the SVM model to some extent predicts other strategies as well, but fails in doing so (especially for a low number of features).

4.3 Confusion Matrix

To some extent, the Confusion Matrix provides insight on how the models learn and which difficulties they might experience. In the case of the Logistic Regression, we see that the model almost exclusively predicts a selling strategy for the imbalanced set and that balancing the data actually impacts the model quite heavily. The Random Forest model tends to make a binary prediction (buy / sell) which scores well as there are also almost no holding responses in the test set. The Support Vector Machines model tries to make a ternary prediction and seems to learn that there are more selling strategies to be predicted, however, both the accuracy and the F_1-score are quite low as it fails in the prediction.

5 Discussion

In a nutshell, this project has showed us well that there is not the one best model that takes everything into consideration and will predict well on completely different results. There are also many more options to possibly implement in a further step, which would have gone beyond the scope, but some of which we will consider here. Firstly, for detecting outliers, a Principal Component Analysis or any other outlier detection method could be used. Also, one could zoom in to the NaN values to check if they are really MCAR or if they are potentially correlated with the data. For the feature selection, in a further step, one could create new features using interaction effects, polynomial features, logistic transformation or other techniques. Also, the models could be trained on all the available features and all the possible subsets and then the optimum of selected features could be chosen.

As for the models, there are many more options on which to train and test our data. We used a simple one-layered neural network. Using more complex networks, such as TensorFlow or PyTorch could potentially heavily impact our evaluation scores. Hereby, one could also directly select the features using a neural network instead of Random Forest, which would most probably yield better results. For the hyperparameter tuning, the Grid Search could also be narrowed down to yield better results.

Lastly, one could weigh different errors differently. E.g., for an investor, a predicted buying strategy when they should actually sell a share is connected to greater costs than a predicted selling strategy that should have been a holding strategy. Here, one could also implement different requirements of different investing strategies or investors and evaluate the models based on that. One last idea is not to do a classification but to do a regression and evaluate actual portfolios based on the prediction because we argue that a stock that performs 2.5% worse than the SP500 is still not as bad of a prediction if predicted as sell than a stock that underperforms the SP500 by 20%. One could then create other classes or evaluate on the different investors' needs.

Bibliography

Pelckmans, K., De Brabanter, J., Suykens, J. A., and De Moor, B. (2005). Handling missing values in support vector machine classifiers. *Neural Networks*, 18(5-6):684–692.

Schumaker, R. P. and Chen, H. (2009). Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):1–19.