Forecasting Liquidity Using a Sentiment Index

Research Project

Duisenberg Honours Program in Finance and Technology

School of Business and Economics - Vrije Universiteit, Amsterdam

Members:
Levi Kamsteeg (25584230)
Kilian Wessels (2579173)
Matteo Guzzetti (2681586)

Supervisor: Dr. S.A. Borovkova

Date: 31/01/2020

Abstract

In this paper we describe the application of machine learning models for liquidity forecasting. Were we use aggregated high frequency sentiment in news as our main explanatory feature to forecast liquidity. As our liquidity proxy we chose the change in volume traded. Comparing two machine learning models, namely RandomForest and Multilayer Perceptron we come to the conclusion that: 1. Liquidity can be forecasted and 2. Sentiment is a relatively imported feature in training the models. By classifying our target output an out-of-sample AUROC score of 93% for the most liquid class and 81% for the most illiquid class have been obtained.

Keywords: Liquidity, Sentiment, High-Frequency, Forecasting, Machine Learning, RandomForest, Neural Networks

Contents

I. Introduction

Market liquidity is one of the most important risk factors in the financial markets. Many participants in the financial markets had to face liquidity crisis over the last few years, the main example is the Great financial crisis of 2008. Therefore, nowadays it has become increasingly important to understand and forecast the characteristics of market liquidity.

This naturally is not the only valuable application of liquidity forecasting. In the field of portfolio management and trading support it is of great importance for asset managers to understand liquidity and forecast this into the future as they preferably choose liquid stocks for their clients that can be bought and sold without any major problems. In risk management the Value at Risk (VaR) is an often-used measure to assess the risk a certain company might be facing in the future. If this measure is not corrected for liquidity the usefulness of the model can decline quite rapidly. Take for example a company knowing that next month it might be in financial distress and therefore certain positions need to be unwound to account for that. If these positions are illiquid and cannot be sold directly, the VaR measure is useless. It can be concluded that forecasting liquidity is important to the financial market as to individual asset manager.

The next issue lays in the question of how to model something like liquidity?
Liquidity is multi-dimensional and is often characterized by highly non-linear patterns and very sparse data which makes it difficult to model. Therefore, this type of model suits the use of machine learning (ML) which doesn't require many assumptions and can help decipher the complexity of liquidity. Moreover, in the context of high frequency trading, there is the possibility to work with high frequency data that allows us to study changes in financial markets during very shorts period, often even up to milliseconds.

An important factor influencing financial markets is the flow of information (Borovkova & Dijkstra, 2018). Next to trades, news items are also increasing in availability on a very high frequent level. This in parallel with the uprising of text mining techniques, that can read those news events and quantify them makes it possible to retrieve sentiment level, which then can represent the possible behavior of an investor.

This paper tries to combine both developments and study the relationship between investor sentiment and market liquidity. Using machine learning techniques, we try to decipher the complexity of high frequency data and by extracting this information we will forecast the movements in market liquidity for a selected heterogeneous group of stocks from the S&P 500.

This research adds to the existing papers on the relationship between market liquidity and sentiment and on previous research on liquidity forecasting via machine learning. More specifically we will follow the main findings of the work of Stefano Pasquali (head of liquidity

at Blackrock), who has been working with machine learning to create adequate liquidity models for quite some time. During his seminar at the Google Cloud convention, he presents his results stating that machine learning models can predict liquidity, even when using relatively simple models.

For a liquidity model to work properly mathematical assumptions must made, such as the assumption of linearity or normality. For a multi-dimensional beast as liquidity this is very difficult (Pasquali, 2019). Therefore, machine learning is a very plausible solution, as it handles multidimensionality and non-linearly quite well. Moreover, using high frequency data, we have a very large dataset which will help the models extract the information better.

When considering machine learning there are two ends of the spectrum in levels of explainability and learning performance. With quite high explainability but a medium learning performance we have ML models such as random forest and decision trees, whereas on the other side of the spectrum we have deep learning models such as neural networks. For interpretability reasons the fewer complex models are preferred as economic intuition can be demanded by investors.

Having introduced the subject, the paper is organized as follows. In section II we consider literature surrounding liquidity and sentiment to have a clearer idea of the relationship between the two. In section III we describe the data and the engineering that comes along. Section IV describes the methodology including the Random Forest and Multilayer Perceptron (MLP) model. Also, the findings of the hyperparameters optimization are discussed. Section V consists of the results, where firstly the outcomes of an OLS regression are presented and continued by the outcomes and out-of-sample performances of the Random Forest and MLP. And finally we conclude our paper in section VI.

II. Literature review

Liquidity has always been a phenomenon of vital importance for understanding the functioning of financial markets. Starting from the first attempt to define it with (Keynes, 1930) who affirms that an asset is more liquid when it is possible to convert it immediately in cash without a loss. Keynes states that in a dealer market, the ask price incorporates a premium for immediate buying while the bid price incorporates a concession for immediate sale. Therefore, the difference between the bid and ask prices reflects the sum of buying premium and selling concession which is used as a measure of illiquidity and is called the bid-ask spread. This spread is the dealer profit for facing the risk to hold the asset, adverse selection costs and inventory holding cost (Krinsky, 1996). Amihud and Mendelson (1986) introduced the concept of stock liquidity focusing on the link between liquidity and cost of capital. The most important finding in this work is that high liquid markets are more attractive to investors. The reason is that they can easily exit from their investments if the market is more liquid.

Subsequently, Fischer Black (1970) is the first who explored whether a relationship between liquidity and return exists, but only after Jack Treynor's work (1971), the intuition became that liquidity is not about value but about price. The main idea is that market liquidity depends on the ability of dealers to absorb imbalances in the flow of supply and demand by using inventories on their own balance sheets as a buffer. The main effect of this intuition is that if the price of an asset moves away from its value, this will attract buyers and sellers. The distance between price and value is the liquidity factor which is affected by a wide range of elements, such as asset characteristics, market conditions, and market imperfections. This is also the reason that Pasquali and Sommer (2015) talk about a multi-dimensional beast. Harris (1990) proposed that liquidity can be defined through four interrelated dimensions:

- Width, distance between bid and ask price
- Depth, volume traded at the bid-ask spread
- Immediacy, the time needed by a large trade to be executed
- Resiliency, the time that the price needs to return to the equilibrium price after a large trade is executed

This is known about dealer markets, but what about limit orders markets? The main difference between these two trading mechanisms is that in a limit order markets the final investors interacts directly with the other investors while in a dealer market the final investors interact with a specialized intermediary who provides liquidity to the liquidity suppliers (investors). Pasquali and Sommer (2015) show how liquidity is affected in these two different types of markets and how they basically react in the same way. That is an increase in the bid-ask price will cause higher transaction costs, when the size of the order goes up. The reason is that in a limit order market trading has a positive cost that increases with size, because the buyer/seller has to walk up/down the schedule of buy/sell limit orders to meet the desired transaction volume, while in a dealer market the dealers' quotes are typically valid for only a limited volume and a short period of time. A large order can be executed by splitting it among several dealers and in that case, effectively, a seller/buyer is walking down/up the demand/supply curve resulting from the aggregation of dealers' bid–ask quotes.

As discussed in the introduction we want to forecast market liquidity using investor sentiment that, as proposed by Baker and Wurgler (2006), is defined as the investor optimism or pessimism about the future of the stock market. The theoretical idea between market liquidity and investor sentiment is that if investors are more optimistic the liquidity of the market increases due to direct effects and indirect effects. Regarding the direct effects, higher investor sentiment generates two different results. The first is that it produces more noise traders that increases market liquidity as shown in DeLong (1990) and Kyle (1985). The second result of higher investor sentiment is that generates more irrational market makers. Since these dealers are assumed to underreact to the information contained in order flows, the price impact caused by order flows is lower, and thus liquidity increases. Regarding the indirect effect,

higher investor sentiment shows that the overconfidence in the market is higher and this causes an increase in stock liquidity.

The empirical relationship between investor sentiment and market liquidity is shown in Liu's (2015) paper, where he tests this relationship in three steps. The first one is to investigate the relationship between investor sentiment and market liquidity. Here the results show that the market is more liquid when investor sentiment is higher. The second step is to check the effects of investor sentiment on trading volume where it is shown that trading volume increases in both cases of noise traders and irrational market makers. Since sentiment is a proxy of overconfidence, it is also shown that, for the indirect effects, we expect an increase in trading volume because overconfident investors trade more (Odean, 1998). The last step is to check whether short sale constraints are important in the relation between sentiment and market liquidity and it is shown that they play an important role in the relation between individual sentiment and market liquidity suggesting the existence of the direct effect that individual sentiment have on market liquidity. However, there is no similar evidence for institutional sentiment.

Having established that there is a link between investor sentiment and market liquidity we will take it a step further by trying to forecast market liquidity using an adequate liquidity model. Nowadays, in the high frequency context, machine learning techniques and in particular artificial neural networks (ANN) have been used in several works in developing applications to help traders in financial decision making. In Putra and Kosal (2011) ANN is used to predict trading signals using technical indicators. In Kablan and Ng (2010) ANN is used to build an algorithm which proposed order placement that considers the market's intraday volatility to reduce transaction costs. In Silva et al. (2014) ANN are used to support the market making strategies using intraday technical indicators as input and they find that ANN is more effective in short term oscillations (5 min) compared to the results obtained in longer periods, this allows to insert higher amount of limit orders that has the result to improve the market liquidity. In Heuver and Triepels (2019) ANN is used to build a probabilistic classifier that provides the probability that a bank faces liquidity stress, using banks payment data and several known stress events as features. This research has two main challenges to make this supervised method work in practice. The first one is that stress events are quite rare and usually last for only few days. Secondly, there is not so much data about stress events at banks. These problems can be addressed using several probabilistic classifiers trained on a dataset that illustrates the payments behavior of several European banks in the last 10 years. Training the classifiers based on a weighted loss function, they can deal with the imbalances between stress and non-stress examples. Furthermore, they labeled the dataset searching for news article regarding stress events at banks, using them as output. Following the last research from DNB (Heuver and Triepels), in our work we will use machine learning techniques, and in particular random forest and MLP. The main difference between this research and the previously discussed one is that we use sentiment news and others features to classify the changes in liquidity, while they use bank's payment data as features to classify whether the banks face liquidity stress.

III. Data

In this section we start with an introduction to our dataset. We go over how we retrieved and constructed our data. We explain the liquidity proxies and sentiment features that we used. Then we show the descriptive statistics of all features. After this we describe how we engineered our features.

Sample Data: The data has been obtained in collaboration with Probability and Partners and Refinitiv. As mentioned in the introduction, Refitniv obtained a sentiment index using text mining techniques. More specifically this is done using Natural Language Processing (NLP) algorithms that can understand and convert the text into usable numerical data. Using both companies' expertise regarding the Sentiment index and high frequency data we set up a scope for our research.

After merging multiple datasets containing high frequency sentiment and high frequency quote data, we created a single clean dataset that could be used for further research. The dataset contains trades aggregated over 5 minutes. This dataset contained 498069 observations, which are all trades observations of all S&P 500 companies starting 2nd January 2014 until 27th September 2018. Table 1 gives an overview of the data which will be further discussed in this section.

Trade and Quote: The merged dataset consisted of data on traded volume and returns, but also the lagged returns and volume up to 6 time periods. This gives a clearer overview of the dynamics in both variables, which will be discussed later on. The Buy Sell Indicator is an indicator which ranges between -2 and 2, to indicate if it's better to buy (2) or sell (-2) the asset. This variable is constructed using the return distribution and devising it into 5 quantiles.

Sentiment: As sentiment is what we want to use to forecast liquidity, it was important to construct all the needed features. In the dataset we received we had an absolute value of sentiment, a lag value and a buzz index.

The absolute sentiment variable is the average sentiment over the last half hour. If an event and thus a sentiment change took place in the first 5 minutes of this half hour it influences the absolute sentiment value a lot more than if the event would happen in the last 5 minutes of the time window. To take this into account we also have a variable called 'Lag', this variable ranges from zero to six. If the variable is zero, the event happened in the last 5 minutes, if the variable is six it happened in the first 5 minutes of the time window. Thus this 'Lag' variable can be interpreted as how long a certain event has influenced the absolute sentiment value.

The absolute sentiment value ranges from minus 0.78 to positive 0.83, to check whether there is different in liquidity for positive and negative news we constructed a dummy, this dummy is one if the sentiment is positive and zero if the sentiment is negative. We also constructed a sentiment change variable, which is the change between the previous sentiment value and the actual sentiment value for each corresponding asset. We also have a buzz index; this is the amount of news generated about the specific asset in the last twelve hours.

One thing we noticed about our dataset is that the sentiment value of negative 0.050568 occurs 83726 times, which is about 16 percentage of the total observations. For comparison the second most recurring observation only occurred 5430 times, which is only one percentage of our observations. After a closer look at our dataset it appears that the negative sentiment value of 0.050568 often occurs at the end of the day, often for a couple of hundreds of stocks at the same time. We looked at some of the days that this occurred. And in some cases, something big happened on macroeconomic level. Therefore, after some discussion we decided to not delete this value from our dataset as we think it might proxy a correction and in order to not lose valuable information.

VARIABLES	N	mean	sd	min	max
Volume	498.069	45.208,179	93.388,68	1	4.630.430
Winsorized volume	498.069	37.085,255	48.970,87	1014	181.800
Volume delta 1	498.069	1,030	13,158	-1	4.029
Winsorized volume delta 1	498.069	0,668	1,41	-0,612	4,656
Log return 5 minutes	498.069	0,000	0,002	-0,129	0,138
Log Return Lag 1	498.069	0,000	0,003	-0,175	0,153
Log Return Lag 2	498.069	0,000	0,003	-0,252	0,153
Buy Sell Indicator 5 min	498.069	-0,001	1,491	-2	2
Volume Lag 1	498.069	29.600,572	61.366,85	1	5.112.796
Volume Lag 2	498.069	30.052,064	63.062,01	1	3.375.144
Volume delta 2	498.069	0,416	14,709	-0,999	4.632
Sentiment Absolute	498.069	0,002	0,471	-0,785	0,829
Sentiment Change	498.069	0,000	0,374	-1,609	1,604
Buzz	498.069	9,052	15,042	0	411

Table 1: Descriptive statistics of firstly the target variables (Volume traded) followed by the explanatory features: log return, lagged volume and sentiment features. For all features the number of observations, mean, standard deviation, minimum and maximum are shown.

Feature Engineering: As the returns and volume showed some outliers, we decided to winsorize the top and bottom one percentage of these features.

As can be seen in table 1, there is a large gap between the minimum absolute volume and the maximum, which is why the differences were taken for the current and lagged volume to have a better understanding of their relationship. The same has been done for the returns, which means that this

variable is the second difference of the price of the stock. As we have returns and volume up to six lags, we were able to construct five 'delta' variables. Because we have data of all the S&P 500 companies, these companies differ a lot in market capitalization. Which leads to a natural difference in traded volume, as a big company gets traded more often than a small company. By taking the difference between between the traded volumes we rule out this characteristic's bias.

As we only have volume traded as a liquidity measure and to create a more robust understanding of liquidity, a proxy has been constructed. There are multiple established liquidity measures, as we have discussed before in our literature review. One of them is the Amihud illiquidity measure (Illiquidity and stock returns: cross-section and time-series effects, Y. Amihud, 2002). This measures the sensitivity of price to trading volume and is formulated as shown in equation 1.1:

$$Illiq = = \frac{1}{N}\sum_{t=1}^{T}\frac{|r_t|}{\$V_t}$$

With $|r_t|$ as the absolute stock return within a certain time window t and $\$V_{t}$ as the respective volume in that time window and N the number of minutes in the time window. The intuition behind the Amihud Illiquidity measure is that if a stock is illiquid the stock price moves a lot in responses to little volume. Unfortunately, after constructing this measure we noticed that because the returns are often zero, due to the five-minute interval, the information provided by this variable was not reliable anymore. Therefore, we chose to not continue using this measure as a proxy for liquidity.

After winsorizing and creating delta variables we also standardized our features, as this was needed for the Multilayer Perceptron that we will explain later. We used the Scikit-learn StandardScaler module to scale the features. It is important to note that in order to prevent data leakage from the training into the test set, the scaler needs to be fitted on trading data only. Thereafter we standardized both the training and test data with the scaler.

As mentioned earlier indicators for liquidity have been constructed as our target variable. Classifiers need a binary or multiclass target value. Just like with the Buy/Sell indicator, the traded volume and the difference in traded volume have been distributed over five quantiles which will be referred to as buckets. This means that the low volume gets placed in bucket one and high volume in bucket five. For the volume difference a high decrease in volume gets placed in bucket one and a high increase gets placed in bucket five. Ultimately, we get the following buckets shown in Table 2.

	min	max
Volume Change Indicator		
Bucket 1	-0.1	-0.327
Bucket 2	-0.327	-0.04
Bucket 3	-0.04	0.34
Bucket 4	0.34	1,413
Bucket 5	1,413	4029,0
Volume		
Bucket 1	1,0	3794,0
Bucket 2	3795,0	9770,0
Bucket 3	9771,0	23539,0
Bucket 4	23540,0	61304,0
Bucket 5	61305,0	4630430,0

Table 2: Descriptive statistics of the 5 liquidity buckets that have been constructed from the volume traded and differenced volume traded. The buckets are the five quantiles taken from the distribution of both variables

IV. Methodology

As mentioned in the introduction, the use of Machine Learning algorithms will help us tackle some issues that liquidity modelling is raising. A lot of the models can work with non-linear data they will not assume most mathematical assumptions. A helpful definition of how Machine learning should be interpreted is given by Mitchell (1997) : 'A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.'. Using this approach, we will discuss the models used in this research.

Random Forrest: The random forest ensemble model was firstly introduced by Ho (1995) and then further developed by Breiman (2001). This model consists of an ensemble (collection) of decision trees. Before we continue with random forest it is important to understand the working of a decision tree. Decision trees are mainly used to predict classes, where each node is used to narrow the classification problem. A more concrete example of how a tree would look like is given at the end of this subsection.

Le (2018) describes the working of decision trees in mathematical terms as follows: For a feature space of size p, which is a subset of $\mathbb{R}_p$ the space can be divided into M regions called R_m.

These regions are p-dimensional blocks separating the different classes from each other. To start growing the tree a first split is made at the root node where a partition of two spaces is created. This split is carried out multiple times, where with each split an impurity measure is used to decide on which split performs best. In our paper we will make use of the Gini index, which measures how pure a certain region is. This is done by:

$$Gini = \sum_{c=1}^{C} \hat{\pi}_{mc}(1 - \hat{\pi}_{mc})$$

where, $\hat{\pi}_{mc}$ is the fraction of training data n region R_m that belongs to a certain class.

As our main goal is to classify liquidity into different buckets we know that decision trees might be very helpful as they are often easy to visualize and to interpret, however it is good to note that they are less robust than other measures due to the fact that the tree might look different if the data is changed, even only slightly. This also means that when you would split the data in two and model the same tree, the results could be quite different. This is where random forest become very helpful. Using a bootstrapped dataset (random sampling of the data with replacement), new data can be created and used to train the decision trees. Lastly an average value of the prediction of the classifiers can be taken which results in a more robust outcome than that of one tree.

The prediction of the random forest is then given by:

$$\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T(x; \Theta_b),$$

where, $T(x; \Theta_b)$ is the decision tree, b is is the b-th tree in the forest and B is the total number of decision trees in the random forest.

The root node (given at the top of the tree) often contains the feature that can distinguish the difference in classes the best. To further distinguish the classes, more splits are made at every node to narrow the region the observation belongs to. In figure 1 for the first two layers the path is shown on how decision in the tree are made. This path consists of five parts and we will discuss them for each row they have in the node:

1. Here a condition that can be answered with True or False is given. In the root node the condition is if the sentiment value is below -0.05 or above, after which it will either go through the left or the right path.
2. The Gini purity score discussed above is calculated here and as we can see it decreases when moving down the tree.
3. Number of observations in the node.
4. The number of samples in each class respectively.

5. Here the majority classification of all the points in the node are translated into the class that has the highest chance of being the desired target output.

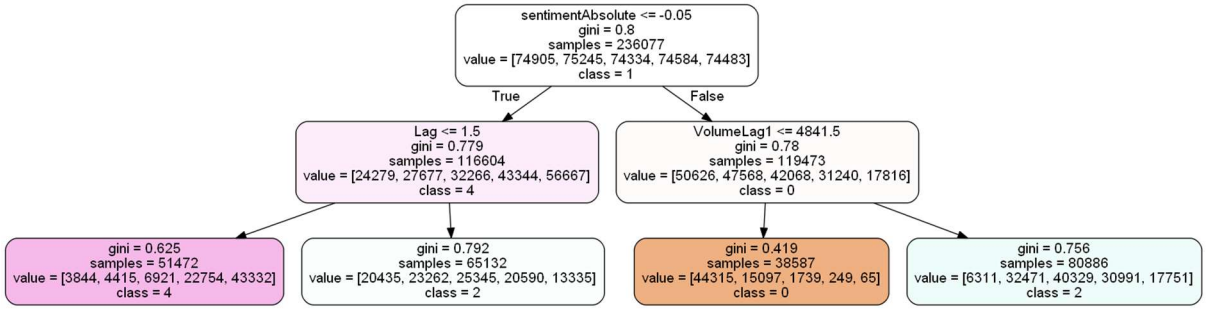


Figure 1: The decision path for one of the trees in our random forest. The tree depth has been limited at 2 and each node shows the condition it has been split on, the Gini score, number of observations, the number of samples in each class and the final decision of the node.

Multilayer Perceptron Network: We also used a Neural Network model, namely the Multilayer Perceptron network (MLP). This is a form of a deep neural network and consists of three components: an input layer, hidden layers and an output layer. In every layer there are multiple nodes, the nodes in the input layer receives input and sends information to all hidden layers where the calculations are made and finally connects the nodes to the output layer where a decision is made or a probability is estimated. Given that we have i number of inputs, h_n the nth hidden layer and o the number of outputs, one can visualize such a model as below:

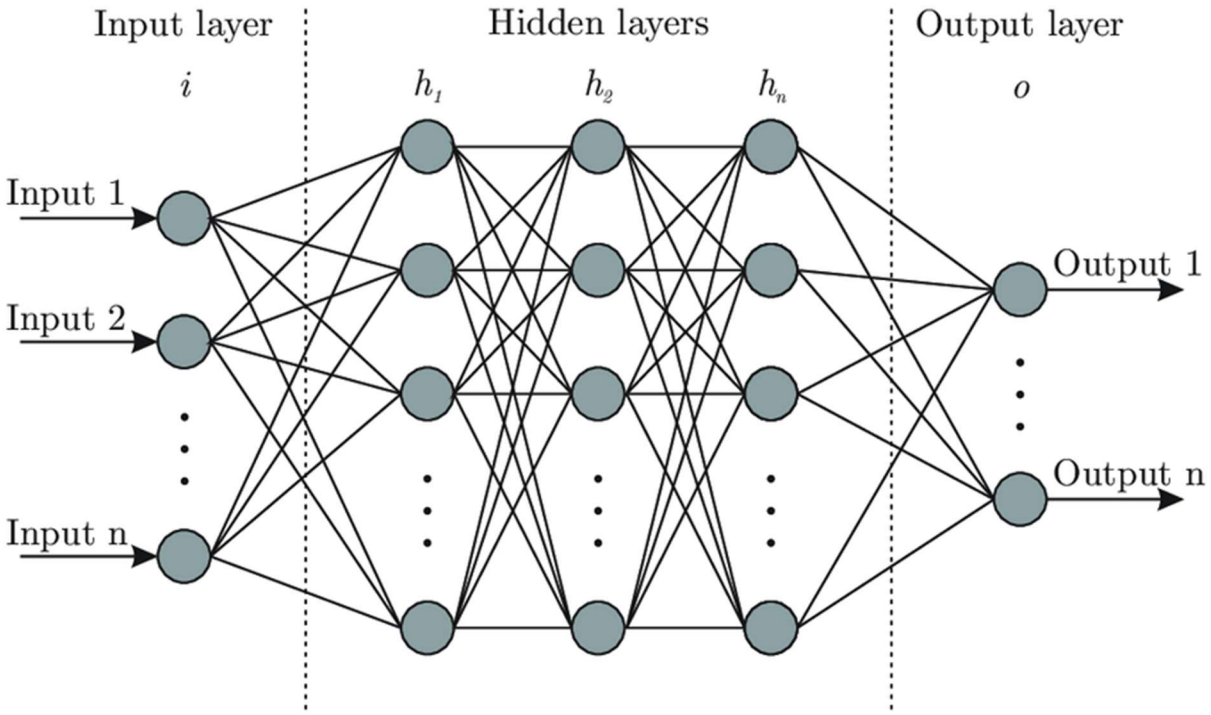


Figure 2: An example of how a Multilayer Perceptron network looks like. The first layer is an input layer and consists of i number of inputs, the last layer is the output layer and consist of o number of outputs and in between we have the hidden layers denoted by h_n that are used to activate the inputs using non-linear mapping.

The figure above shows the relations between the layers. As an MLP network is a feedforward network the nodes only sent information forward and thus the artificial neural network does not form a cycle. Another restriction is that there is no connection between nodes in the same layer. Each node in the hidden layer is a neuron, these neurons use a nonlinear activation function and are the true engine of the model. It means that these hidden layers enable the model to learn non-linear mapping from its input to the outputs.

Mathematically we can describe the process as follows (Heuver and Triepels, 2019): Let ∂, be the number of layers in total, $s_i$the size of the i-th layer and the output of each layer be denoted by ρ_i. The output of the input layer will be a vector containing all features, $x_i^{(k)}$. Then this vector is passed to the hidden layers for which the output will be:

$$\rho_i = \psi^{s_i}(W_i\rho_{i-1} + b_i), \text{ for } 1 < i < \partial.$$

Here W_i is a s_i by s_{i-1} matrix of weights, $b_i$the bias term and ψ^{s_i} a set of non-linear activation functions applied to the feature vector $x_i^{(k)}$. In this paper we use the tan activation function given by $f(x) = \tanh(x)$, as proposed in (LeCun et al., 1998) due to the outcomes of hyperparameters optimization which will be discussed later.

The output of the last hidden layer ρ_{i-1} is processed through a single sigmoid unit:

$\rho_\partial = \sigma(w_\partial \rho_{\partial-1} + b_\partial)$, where w_∂ is now a row of weights instead of a matrix and b_∂ denotes the bias for the last layer. Output ρ_∂ is now an estimate of $f(x_i^{(k)})$ which can be either a decision outcome or in this case an estimation of the probability for a certain liquidity bucket.

Model Optimization: For both models the hyperparameters need to be set before deploying it. Finding these hyperparameters can be a computationally costly exercise and is also one of the limitations in this research. An often-used tool is the GridSearch deployed by ScikitLearn, which exhaustively searches for optima from a grid of parameter values that can be specified beforehand. As this is an exhaustive search and to deal with limited computational power the parameter values given to the optimizer were given in a small range. The following parameters were optimized per model:

Random forest:

- Max depth: This parameter is used to decide on the depth of each tree. On default this parameter will continue to grow the tree until a leaf is pure, which means that all the datapoints on the leaf will contain the same class. From theory we know that the deeper the tree, the more prone it is to overfitting as we are narrowing the pureness of a leaf. The GridSearch showed that a depth of 5 was enough to create robust decision tree.

The rest of the parameters have been left on their default value as they did not contribute that much in the performance of the trees keeping in mind the computational power required for extra parameters.

Multilayer Perceptron:

- Activation function for the hidden layer: We discussed the importance of this function in the previous subsection and found that hidden layers translated into output the best when activated by a hyperbolic tan function.
- The L2 regularization term: This parameter regulates the penalty that is added to the error function. This is an important parameter as a large penalty term will cause overfitting on the given train set, whereas a very small penalty can cause under-fitting (Bullarina, 2004). Given our dataset a small penalty of 0.0001 was optimal to train the model with.
- Number of hidden layers: Here the depth of the neural network is determined. The number of layers determines how complex input information can be retrieved. Often 1 layer will suffice for simple problems, whereas in deep learning multiple layers will suit the model better but will be computationally costly. Jeff Heaton (2008), proposes a rule of thumb that the number of layers should be between the input and output size. Taking this into account we test for 3 possible layers, namely (10,8,6,4,2) (100,) and (100,100,100,100,100). The results show that extending the number of neurons did not increase the performance of the model, but also 1 layer is not enough to capture all information. Therefore, we settled on the first option, were a decreasing number of neurons are chosen to cut down in computation time.

- To update that weight at each point we use the 'adam' solver as proposed by Kingma & Ba (2014), because of its efficiency with large data frames.

V. Results

In this section we will analyze the results obtained in this research. Firstly, we will show that no linear relationship can be found, which is why machine learning models are considered. Then we will introduce the results of those models with their performance scores. We consider the confusion matrix with the recall, precision and f1 score and end the result section with the ROC curve including the area under the curve.

Linear Relationship: Considering the paper of Liu (2015), where the link between sentiment and liquidity has been researched, we started of this research with the same idea. The only big difference between the researches is that Liu uses a continuous time series containing both a sentiment index and liquidity measures. In our research we use high frequency data of sentiment and liquidity that only is registered when a trade has taken place. Therefore, we are using a cross-sectional analysis that looks at the broader question, namely are sentiment and liquidity related?

The results are somewhat counterintuitive. We know from the literature that we should expect a positive relation between liquidity and positive sentiment. Using our dataset however this doesn't seem to be the case. In table 3 we see a negative relationship that is quite significant between the variable positive sentiment and change in liquidity. When performing this analysis for all the companies separately, we see that in 28,9% of the companies the coefficient for positive sentiment is positive but not significant at all cases. Although the relationship between sentiment change and liquidity change is positive the coefficient of positive sentiment is too negative to make it economically interpretable. To verify its inefficiency to address the research question, we test the model for its out of sample performance. With the coefficient of determination being equal to 0, we come to our first conclusion, namely that we cannot find an economically plausible linear relationship between liquidity and sentiment value in the cross-section.

VARIABLES	OLS
Cons	1,9103
	(0.003)
Lag T2	-1,2769
	(0.006)
Lag T3	-1,2785
	(0.006)
Lag T4	-1,2758
	(0.006)
Lag T5	-1,2748
	(0.006)
Lag T6	-1,2850
	(0.006)
Positive Sentiment	-0,5168
	(0.004)
Sentiment Change	0,1485
	(0.005)
log Return 5min	-1,9543
	(0.745)
Buzz	-0,0102
	(0)
Volume delta 2	-0,0016
	(0)
Volume delta 3	-5,97E-05
	(7.48E-05)
Volume delta 4	0,0001
	(7.87E-05)

Table 3: The results of the OLS regression are presented here. The depended variable is the change in liquidity, whereas the independent variables are the lagged values thereof, log Return and sentiment variables.

Having that said this still does not mean that our search in forecasting liquidity is over. We still have a quite powerful tool to use, namely machine learning. Because a plausible linear relationship could not be found, it is encouraged to try on machine learning techniques. As discussed in the Data section, we

choose the liquidity buckets as our target value. We consider both the absolute traded volume as the differenced traded volume buckets.

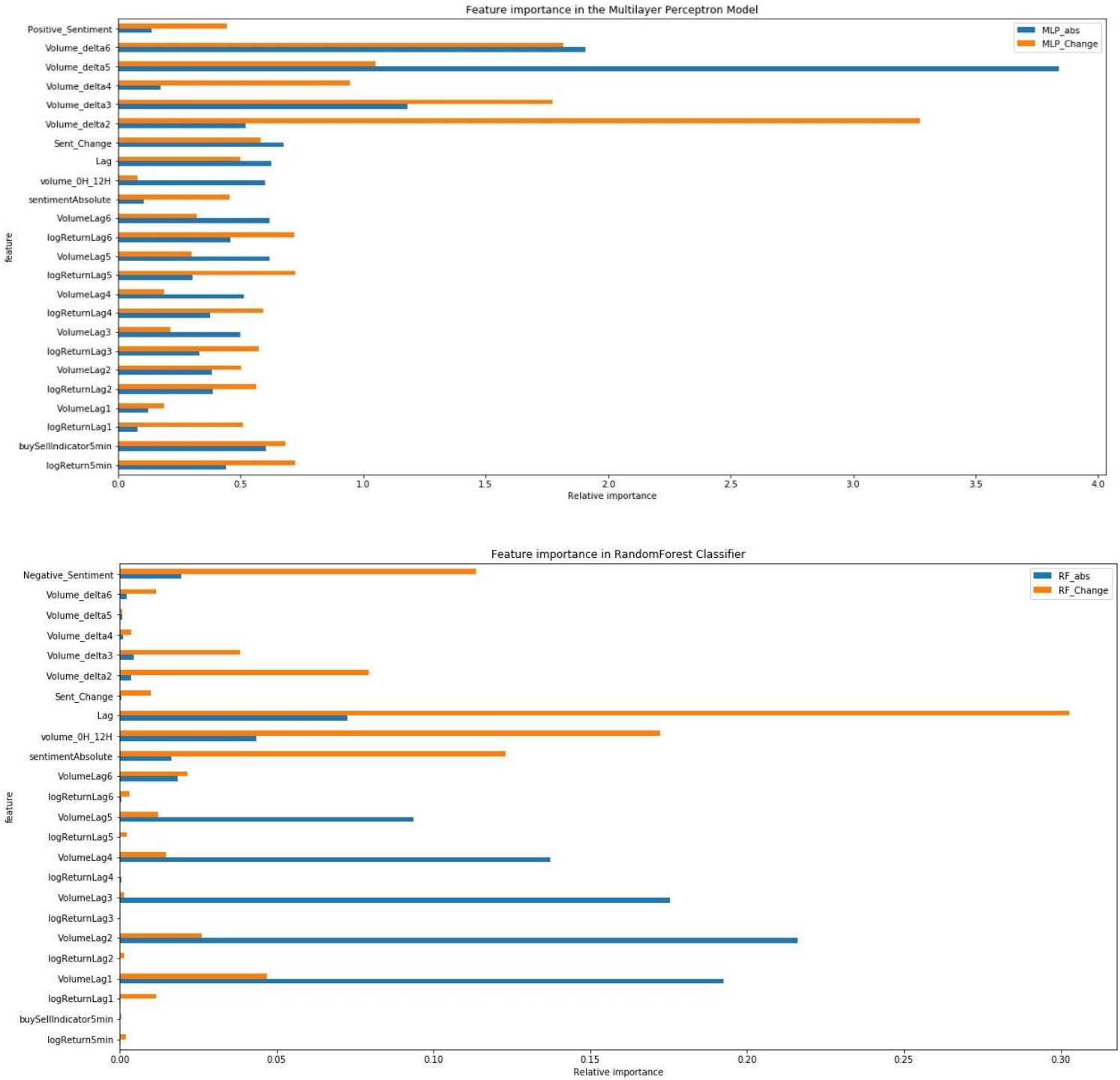


Figure 3: The feature importance of the Multilayer Perceptron model (top) and Random Forest (bottom). In blue the absolute indicator of liquidity was chosen as output target and in orange the change indicator in liquidity. On the y-axis al explanatory features are depicted, whereas the x-axis shows the relative importance in training the model of the features.

For both ML models the relative feature importance has been calculated. The feature importance of the Random Forest and Multilayer Perceptron are presented in figure 3. These graphs tell us the relative level of information contribution of the features when training the model. For the absolute traded volume model, we see that the lags in traded volume are informative, as explained earlier this can be explained as volume is highly correlated with company specific characteristics rather than sentiment. The absolute volume contains information about the size of the firm, a large firm will by default have a constant higher volume than a small firm. We can omit this bias using the difference in traded volume.

When using random forest to predict the classification of the change in volume indicator, we see that the features: lag, buzz and sentiment index obtain quite some importance in the model. This is a quite striking result as it shows that the sentiment features contribute to the performance of the model.

For the MLP feature importance we only looked at the first hidden layer. This layer consists of 10 hyperbolic tan (the result of the grid search) combinations for each feature that we input. By calculating the absolute weight coefficient for each feature, we can take a look at the relative importance for the MLP model. Again, we compare both the absolute as the differenced target. Although this graph gives us some indication as to which features contribute the most, there are still 5 other layers, which can slightly differ in feature relevance, which could explain the difference in feature relevance for the random forest and the MLP model.

	Random Forrest	
	Differences	*Absolute*
Recall	0,415	*0,672*
Precision	0,398	0,671
F1_score	0,398	0,671
	Multilayer Perceptron	
	Difference	*Absolute*
Recall	0,460	0,704
Precision	0,456	0,703
F1_score	0,455	0,703

Table 4: The results obtained after training and cross validation for the testing part. The table contains scores for Recall, precision and the harmonic mean between the two, namely the f1 score. Both the MLP and Random forest are considered using the absolute and differenced liquidity indicators as target values.

The end results are quite promising. For both outputs, the Multilayer Perceptron outperforms the Random Forest model. This is probably due to the fact that the multilayer perceptron is considered to be more complex especially when more than one hidden layer is added to the model. The differences

are not that big, meaning that a relatively simple model as random forest can make quite accurate predictions. However, as there is not any literature yet on forecasting this kind of measure for liquidity it is hard to make a relative comparison.

Seeing that the Multilayer Perceptron outperforms the Random Forest model, we wanted to further investigate the Multilayer Perceptron results. One of the first things we looked at was the confusion matrix. A confusion matrix, also known as an error matrix, is a visualization of the correct and incorrect predictions of each bucket. The horizontal axis shows the predicted bucket of the machine learning algorithm and the vertical axis shows the actual bucket the volume is in. As can be seen in figure 4, the model was more right than wrong. As each bucket has the exact same amount of observations, it is clear that the algorithm performed best when it had to classify bucket one or five. One other very important conclusion that can be drawn from the graph is that if the classifier picked the wrong bucket, it was most of the time the bucket closest to the correct bucket. It very rarely happened that it picked bucket one or five when it had to pick respectively five or one. As in our classification problem the incorrectness of prediction one bucket lower or higher than the true bucket is lower than if it would have predicted two buckets lower or higher. Please note that the F1-score does not differentiate between almost correct or not correct at all. This means that the obtained scores are very conservative, and the predictions are in reality even better.

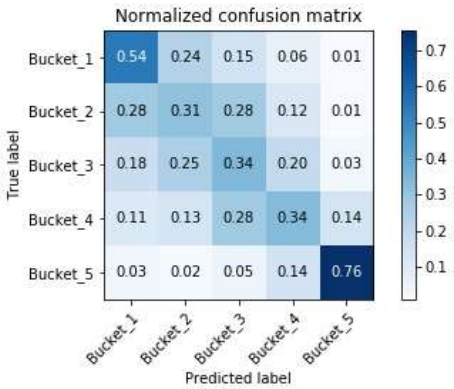
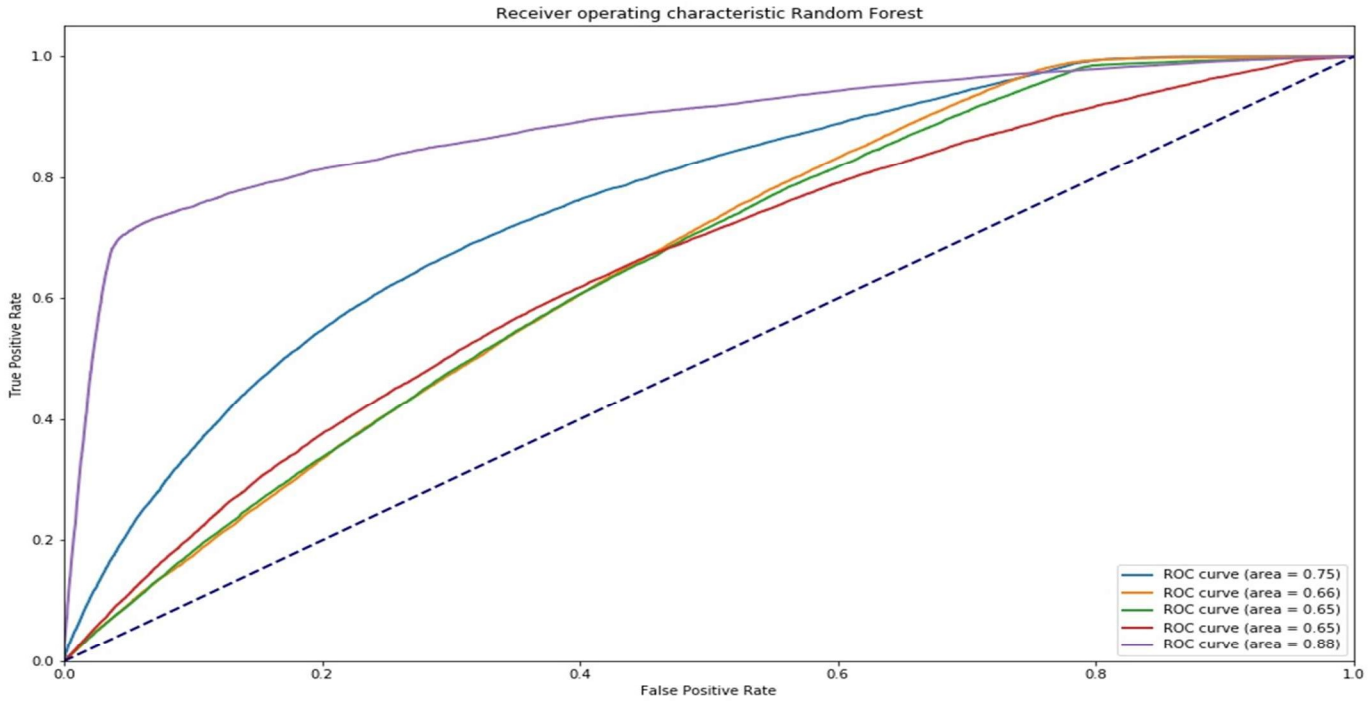


Figure 5: The normalized confusion matrix, where the predictions for all buckets are depicted. On the diagonal we have the true positives, which measures the percentage of correct predictions per bucket.

To further illustrate the performance of both models we illustrate the Receiver Operating Characteristic. Here a plot is made between the true positive rate and false positive rate at different classification thresholds. The ROC uses the same predicted probabilities as calculated for each bucket. Normally the ROC curve is used to assess binary targets. As we have a multiple classes, it is important that we binarize our outputs. This means that for every bucket we assess if the output target is equal to a certain bucket and then assign the value one to that specific bucket, which is also known as the one versus all methodology. Now we have 5 output targets which all represent a bucket and contain the

value 1 if the outcome of the multi classifier equals to the bucket number. Then the observed target variable is compared against the predicted probability and plotted on the graph in terms of its sensitivity (true positive) and specificity (true negative). In the ROC figures the The Area under the ROC (AUROC) is depicted as well and it provides an aggregate measure of performance for all threshold provided in the ROC curve.



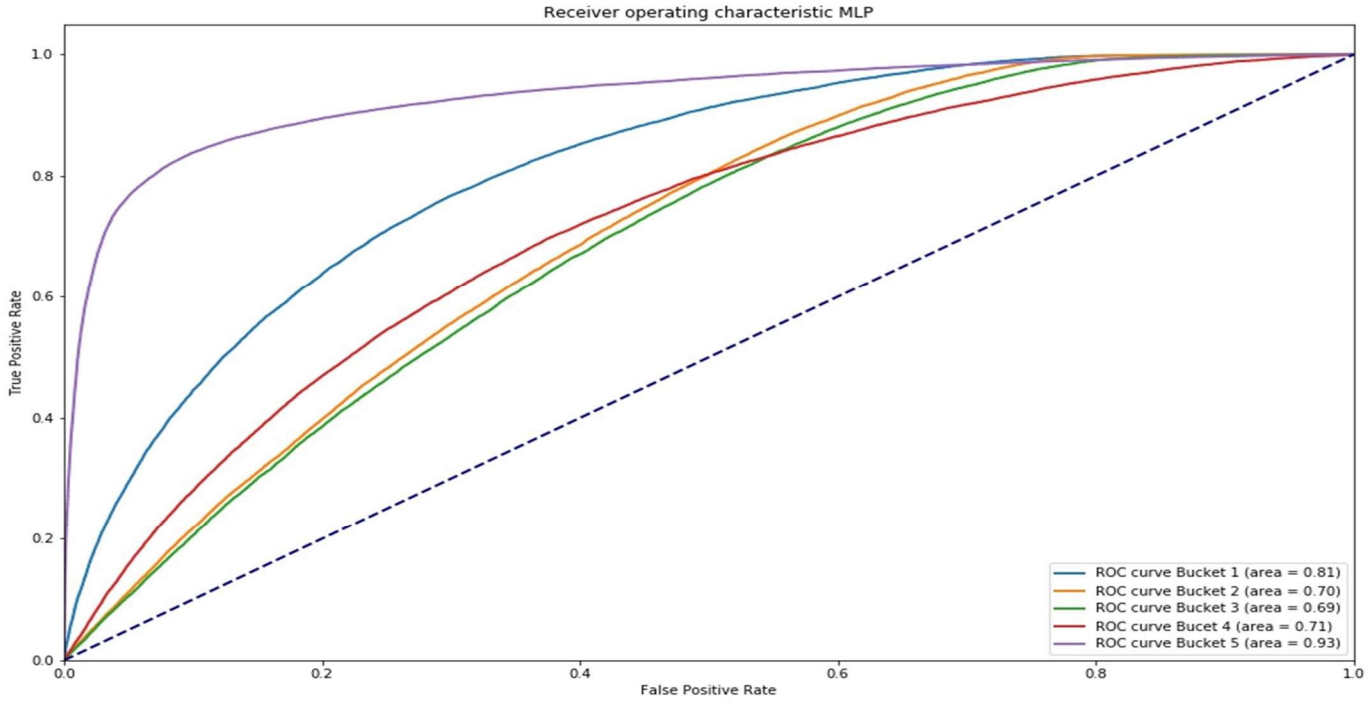


Figure 6: The Receiver Operating Characteristics curve for all 5 buckets. The buckets are assessed using the one versus all methodology. The area under the curve representing an aggregate performance measure for all threshold in the ROC curve is depicted in the legend. Note that an AUROC value of 0.5 mean that the model does not perform at all.

A possible interpretation is that the model ranks a random positive observation more highly than a negative observation. In our case, a randomly recorded trade with a large increase in volume traded will have a higher probability to be assigned to the 'very liquid' bucket than a trade with a small increase or decrease in volume. The Area under the curve tells us for each bucket how well the model performed. Like the other performance measures in table 4, we see that the AUROC is higher for all cases in the MLP model. For the very liquid bucket it means that the probability that the model can distinguish between the positive (the right bucket) and negative (the wrong bucket) class is equal to 93%, which is a very good outcome. For the most illiquid bucket this is equal to 81%.

VI. Conclusion, Limitations and Further Research

Concluding this paper, we examine the relationship in the cross-section of investor sentiment on market liquidity for the S&P 500 stocks. Our data includes all trades for these stocks spanning a period from 2nd January 2014 to 27th September 2018. We used absolute volume traded and the change in volume traded as liquidity proxies and for sentiment we used an absolute sentiment value, the change in sentiment, a dummy for positive or negative sentiment and a buzz index which measured the volume of news generated for the corresponding stock.

We started this research with a simple OLS regression that turned out not to have any significant forecasting capabilities. And thus, the conclusion could be made that there is no economy plausible linear relation between investor sentiment and market liquidity. To further examine the possibility of a non-linear relationship different machine learning techniques called Random Forest and Multilayer Perceptron were considered. To use these classification models we constructed five buckets, ranking from low to high volume and negative to positive change in volume. With a high positive change in volume in bucket five and a high negative change in volume in bucket one.

These two ML algorithms perform quite well in this classification problem, providing us with a F1 score for the absolute values of volume using the Multilayer Perceptron of 0,703 against a 0,671 F1 score for the Random Forest. For the change in sentiment we got an F1 score of 0,455 and 0,398 for the Multilayer Perceptron and the Random Forest respectively. These results are in line with the research of Pasquali (Sommer, Liquidity - How to Capture a Multidimensional Beast, 2015) who gave a Google seminar in 2019 about forecasting liquidity, where he mentioned that neural network performs better in this type of forecasting.

As we are most interested in forecasting the change of liquidity the area under the ROC curve has been calculated as well for this target variable using the one vs all methodology. The results for the most liquid and most illiquid bucket were 93% and 81% respectively for the MLP model. And therefore, we conclude that it is possible to forecasting liquidity. As our research question is: does sentiment influence liquidity? We considered the feature importance's of our ML models with the striking result that the sentiment features have a relatively large impact in training the models. Thus, we can conclude that sentiment can be used to forecast liquidity.

In 1990 DeLong and Shleifer (De Long, 1990) already concluded that sentiment leads to noise traders and noise traders lead to an increase in market liquidity. With the evidence provided by this paper, we can confirm these findings using machine learning models. As mentioned in the introduction this result might prove to be helpful for agents in the financial markets in the fields of portfolio management, transaction support and risk management.

One limitation can be found in testing the forecasting abilities of our models. Here the one versus all method has been used. This makes it a multiclass classification problem, but the scoring measures we used did not differentiate between slightly incorrect and totally incorrect. Therefore,

arguably our results are better than our scoring indicates. For future research it is recommend to use a scoring measure that takes multiclass classifications into account.

Arguably our largest limitation in our research was the lack of computational power. In future research a larger set of parameters should be given to the GridSearch or another optimization program, which could result in better performance. Another limitation was the absence of another liquidity measures. The lack hereof makes the results less robust, and in future research it is recommended to also take other liquidity measures into account, specifically the bid-ask spread. It is good to note that only two machine learning models were used in this research and as we know there are a lot of alternative options that could be considered as for example a Recurrent Neural Network. Also a more applied research could be conducted where liquidity would be incorporated in Value at Risk models, to extend research on risk management.

VII. References

Amihud, Y. a. (1986). *Asset pricing and the bid-ask spread.* Journal of Financial Economics.

Bagehot, W. (. (1971). The Only Game in Town. *Financial Analysts Journal*.

BAKER, M. a. (2006). Investor Sentiment and the Cross-Section of Stock Returns. *The Journal of Finance, 61*.

Black, F. (1970). Fundamentals of Liquidity. *University of Chicago*.

Borovkova, S., & Dijkstra, M. (2018). Deep Learning Prediction of the EUROSTOXX 50 with News Sentiment. *SSRN*.

De Long, J. B. (1990). Noise Trader Risk in Financial Markets. *Journal of Political Economy, 98*.

Harris, L. (1990). Liquidity, Trading Rules, and Electronic Trading Systems. *New York University Salomon Center Monograph Series in Finance*.

Heuver, R. a. (2019). Liquidity Stress Detection in the European Banking Sector . *De Nederlandsche Bank Working Paper No. 642*.

Keynes, J. (1930). *A Treatise on Money* (Vol. 2). London: Macmillan.

Kosala, E. F. (2011). Application of artificial neural networks to predict intraday trading signals. *Recent Researches in E-Activities*.

Krinsky, I. &. (1996). *Earnings announcements and the components of the bid-ask spread.* The Journal of Finance.

Kyle, A. S. (1985). Continuous Auctions and Insider Trading. *Econometrica, 53*.

Liu, S. (2015). Investor Sentiment and Stock Market Liquidity. *Journal of Behavioral Finance*.

Ng, A. K. (2010). High frequency trading using fuzzy momentum analysis. *Proceedings of the World Congress on Engineering, vol. 1*.

Odean, T. (1998). Volume, Volatility, Price, and Profit When All Traders Are Above Average. *Journal of Finance, 53*.

Pasquali, S. (2019). Machine Learning Framework for Liquidity Risk Management . *Next.* Google Cloud.

Silva, E. &. (2014). A neural network based approach to support the Market Making strategies in High-Frequency Trading. *Proceedings of the International Joint Conference on Neural Networks*.

Sommer, S. P. (2015). Liquidity - How to Capture a Multidimensional Beast. *The Journal Of Trading*.