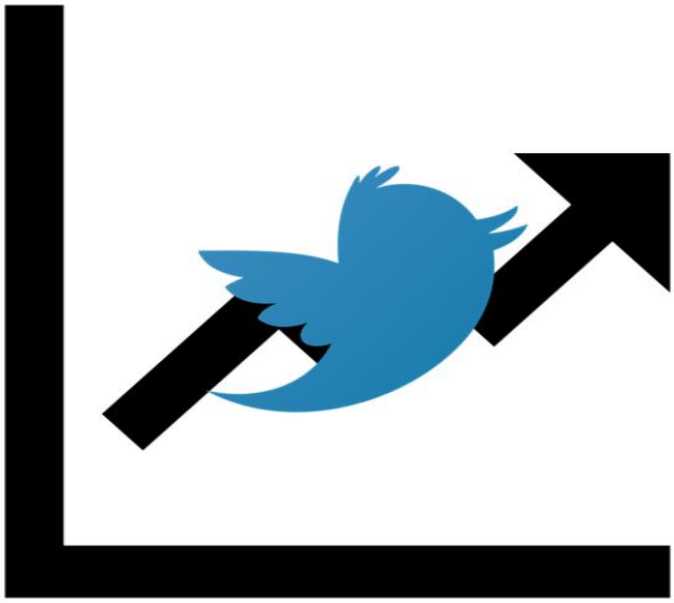


# FORECASTING STOCK MARKETS USING SOCIAL MEDIA

## Forecasting med sociale medier


### Resumé

The paper is inspired by the recent interest in forecasting using social media. The intentions of this paper is to test whether simple models can be applied and achieve similar results, as more advanced approaches already used in the literature. The paper is presented as a point of departure for further research and should not be read as a conclusive study on the subject. It furthermore provides basic insight into the world of sentiment analysis and social media-based forecasting.



Forfatter: Rasmus Lillebo Rudbæk Olsen, 201510489, au531212@uni.au.dk

Vejleder: E. Christian Montes Schutte


Tegn med mellemrum, eks. Billeder: 26.113

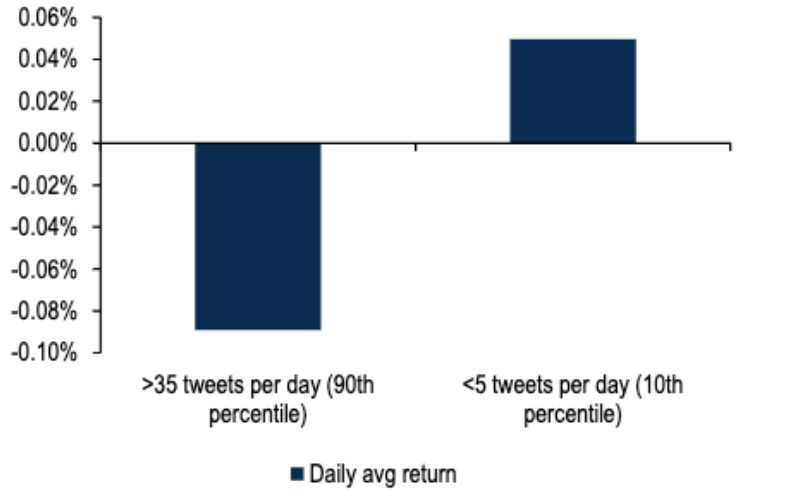
# Indhold

## Introduction:

Since the election of US president Trump, it has become increasingly evident how social media, and in particular twitter, has become a factor financial markets has to take into account. This correlation has not only shifted the models on Wall Street to also incorporate social media but is also evidence of an increasingly integrated and convoluted environment navigate when forecasting. For example; Merrill Lynch started modeling their investments after President Donald Trump's twitter account activity. as seen below. And Bloomberg.com now has a tracker for mentions of the financial markets on the same account[1].

**Chart 5: The market tends to fall, on average by 9bp, on days Trumps is more active on Twitter**

Average daily returns when Trump writes more than 35 Tweets a day (90th percentile) versus less than five Tweets a day (10th percentile), since 2016




Source: BofA Merrill Lynch US Equity & Quant Strategy, Twitter, S&P

Such initiatives shifted more focus onto the existing development in text mining and sentiment analysis, that has seen exponential growth in recent years. Topics like these is where machine learning is said to outperform, and with prominent figures in the financial sector already at it, it is reasonable to believe the research is well founded.

[1] Bloomberg.com mentions tracker: https://www.bloomberg.com/features/trump-tweets-market/

Merrill Lynch found that the financial markets would on average be positive if the current president posted less than five times a day, and negative for more than 35 tweets.

In theory, that is an exploitable correlation that could potentially be used as regressors when modeling forecasting tools for the financial markets.

In that line of thought, my intention with this paper is to:

1. Find whether twitter can be used to forecast S&P500
2. Find what type of data is most effective when forecasting S&P500 using twitter.
3. Find if the literature[2] can be repeated on a smaller scale and with less convoluted models.

In 2010, Bollen et. Al. proved correlation between public sentiment and the stock market. The study was conducted using twitter as data source, and a Self-Organizing Fuzzy Neural Network (SOFNN). Which is basically a neural network with a range of logic, instead of the binary true or false.

Such a model is not only advanced, its is also beyond the scope of what is currently available to me in Rstudio and Matlab. I will therefore attempt to recreate similar results using basic machine learning.

I will be using simple ordinary least squares (OLS) as my benchmark model, and then test whether I can find similar results using simple neural networks. The data I will be using is President Donald Trump's twitter history from 1-11-2016, the month he was elected president, up until 01-04-2020. This data will thereafter be split into text- and activity-based data, and then tested separately. This is to further argue whether sentiment analysis or regular analysis of activity show proof of higher correlation.

Midway through the analysis, I will pause and ponder by relating my results to a third machine learning model; Random forest.

To check whether my neural network holds up as the preferred model for this type of regression, or the deeper neural network is a requirement to get sufficient results.

[2] Bollen et. Al. (2010)

Inspired by Merrill Lynch, I will forecast S&P500 with all 4 models: 2 with activity data, and 2 with data created from text analysis. The random forest will not be forecasted, as it does not contribute to my intentions with this paper.

## Data:

Twitter is a social media platform that allows an individual or company to share 140 characters or less to all that follows the account. It also allows users on the platform to share posts on their own account (Retweet) and highlight posts in their interest (Favorite).

The platform has 330 million active users a month and 145 million daily users (2019). 63% of the users are between 35-65 years old (2018), and 40% of users has reportedly made a purchase after seeing ads on twitter[3]. These numbers show that the data is unfiltered and created by people, and therefore optimal for sentiment analysis. It also shows that people are willing to react based on what they see on twitter.

The data in its raw form is stated below:

**Text (Character):** Consists of the 140 characters or less Donald Trump chooses to share with his followers.

**Created_at (Datetime):** Shows the exact time the 'tweet' was posted online on twitter.com by President Donald Trump or his social media team.

**Retweet_count (numeric):** Shows the count of reposts the single 'tweet' has.

**Favorite_count (numeric):** Shows the count of 'Favorites' the single 'tweet' has.

Furthermore, I use the 'Created_at' column to count the number of 'tweets' Donald Trump has on any specific day and call this **'Frequency'**.

The 'Text' variable will go through multiple cleaning and sorting steps, resulting in 4 different sentiment scores:

[3] Digital Marketing Institute, 2019.

**MeanSentiment (numeric):** The mean sentiment score of the day

**Positive:** The positive sentiment score of the day

**Negative:** The negative sentiment score of the day

**Neutral:** The neutral sentiment score of the day.

I will use S&P500 adjusted prices in both datasets, which therefore limits the days in the processed data to open market days.

# Methodology:

## Method:

I use machine learning due to its ability to better navigate in high dimensional environments, while OLS will be provided as a benchmark.

The methods used in this paper have been selected by their ability to make sense of noisy data.

While the two machine learning techniques have a higher chance to overfit, I believe the structure of the data will provide better results with these, than models with lower chance of overfitting.

The models used are OLS, neural networks and random forest. The neural networks will be cross validated to provide a better estimate of the mean squared error for both networks.

## OLS

OLS or 'Ordinary Least Squares' is perhaps the simplest model available to econometricians. I will therefore not spend a lot of pages writing about this model, as it should already be familiar to the reader.

However, the OLS does provide a few constraints due to its assumptions.

Because of these assumptions we not only normalize and scale the data, but I also omit the variable "MeanSentiment" from the analysis, as this model is computed by taking the mean of "Positive" and "Negative", and therefore violate the assumption of no multicollinearity.

## Neural Networks

Machine learning has proved to be especially capable in environments with large amounts of noisy data. Neural networks or "single hidden layer back-propagation" networks are non-linear multi-stage regression models, that can be used for both classification and regression problems.

I will only be using it for regression in this paper.

Neural networks recognize patterns from inputs and weights, which it runs through two functions called the *net input function* and the *activation function*. Much of the neural network's parameter specification is black box-based computations. However, we can specify certain inputs in the model, and we know how the network works. I've chosen neural networks due to literature having favored them in sentiment analysis, and because they are easily applicable. They are also prone to overfit; We therefore bypass this issue by cross validating the results.

The network itself consists of two parts; namely the neuron and the weights. Neurons are just functions that take some input, and output a number between 0 and 1, called the 'Activation'. Neurons appear in rows and in layers; the layers correspond to steps in the algorithm, whereas the neurons are stacked in rows in each layer. The last layer is the output coefficient, which includes another step, that decides how good the fit of the network is.

Between the starting layer and the last layer, we usually find some hidden layers, which are just the name for the layers in between.

The number of neurons in each layer is arbitrary, but there are some rules of thumb: According to Heaton[4], the number of hidden layers should be between the number of inputs and output. Both the networks presented in this paper have 4 inputs and one output, which is why I experimented in the range from 2 to 4 hidden layers.

[4] Heaton, Jeff. Ph.D in comuter science. https://www.heatonresearch.com/2017/06/01/hidden-layers.html

The activation in a single neuron in a single layer, determines the activation in the next layer.

In order to create a fit, we assign a weight to each neuron in each layer, and then assign that weight to the activation of the neuron, which is then cumulated.

As previously stated, we typically want our activation to be between zero and one, which means we must include a function that scales the weight into the desired range, this function is called the activation function.

The activation function used in this paper is the sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Also knows as the logistic curve. The sigmoid function scales the highly negative values, into something close to zero, and the highly positive values into something close to one. The function then has a slight increase from values larger than 1. More recent papers frequently use Rectified Linear Unit (ReLU) as activation function, which is closer to the biological interpretation of the model, and easier to train when diving into deep neural networks. More formally; have sigmoid be $\sigma$, the activation $a$ and the weights $w$.

$$\sigma(w_1 a_1 + w_2 a_2 + w_3 a_3 + \cdots + w_n a_n - bias)$$

By this logic, the layer is just a measure of how positive a weighted sum is. However, in practice neural networks consist of more layers, which can be represented as follow:

$$\sigma\left(\begin{bmatrix} w_{0,0} & \cdots & w_{0,n} \\ \vdots & \ddots & \vdots \\ w_{k,0} & \cdots & w_{k.n} \end{bmatrix}\begin{bmatrix} a_0^0 \\ \vdots \\ a_n^0 \end{bmatrix} + \begin{bmatrix} b_0 \\ \vdots \\ b_n \end{bmatrix}\right)$$

$$or$$

$$a^1 = \sigma(Wa^0 + b)$$

The bias in neural networks can be thought of as the intercepts in OLS. It allows the output of the activation function to increase or decrease.

The neural network is not only a network of functions, but also has the ability to improve through training. When the network has computed the last layer, it takes the sum of the squares of each difference between the prediction and the goal for each component, and that becomes the average cost of its inaccuracy. This is called the cost function:

$$C_0(\dots) = (a^L - y)^2$$

Where $y$ is our desired output.

The cost function is a single number of the network's inaccuracy. We therefore want to minimize the cost function to optimize our model. Some input spaces have local minima, which underline the importance of accurate steps on the surface and step size.

We know from multivariable calculus that the negative gradient gives the direction of steepest slope on a surface, that is the cost function. The weights and activations therefore change themselves in order to find this minimum.

This entire step of minimizing the cost function is called the 'back propagation', and the action of repeating steps of minimization is called 'gradient decent'. I will not go far into the calculus behind this minimization problem, as the mathematics should already be familiar to the reader, but the formal notation has been included in **Appendix 4.**

However, to understand the intuition behind the gradient decent, it is important to note how the weights and biases that connects the neurons, have different effects on the cost function. This is due to the cost function being more sensitive to some components than others[5].

Back propagation is the algorithm that re-specify the weight and bias of a single training sample, thereby determining the most optimal gradient decent.

Typically, we don't want the global minimizer of the cost function, as this is likely to be an overfit solution. Instead some regularization is needed: this is achieved directly through a penalty term, or indirectly by stopping early.

[5] Further reading on the learning aspect in neural networks is called 'Hebbian theory' and is beyond the scope of this paper.

It is common for neural networks to show overfitting due to too many weights, on top of that, they also perform better when scaled. This makes it valuable to preprocess the data, which we can do by standardizing the mean and standard deviation into the range [0, 1]. This process ensures equal computations for all inputs in the regularization process.

## Random Forest:

The following model appears in material that is prerequisite to the reader. I will therefore not go into what a decision tree or bootstrapping is.

Random forest is a simple tree-based machine learning technique, which predicts using bagged, independent decision trees. Bagging is a method of lowering variance error. The random forest algorithm creates a bunch of decorrelated decision trees on bootstrapped training samples. But when a split is considered in the model, it creates a random sample of predictors as candidates from the full set of predictors and chose one for the split. This is due to strong predictors in the set making predicted trees highly correlated.

Random forest recognizes that repeated applications of the same learning algorithm on different subsets, can lead to highly correlated predictors, which furthermore limits the amount of variance reduction.

Therefore, in order to properly decorrelate the predictors, each is based on a randomized subset of predictors.

Note that Random forest, unlike other tree-based models, does not prune its individual trees. However, random forest presents other favorable properties, as it not only scales well, but is also easy to apply, due to its versatility.

If we follow the notation of 'The Elements of Statistical Learning '(2009), page 588, we find the following:

Have $Z^*$be the bootstrapped sample of size $N$ from training data, and $T_b$ be the random-forest tree from the bootstrapped sample. Then, for each terminal node of the tree, go through the following steps repeatedly:

1. Randomly select $m$ variables from the $p$ variables available.
2. Choose the optimal split-point among the $m$ variables.
3. Split the node into two daughter nodes.

This ensample of trees is then denoted as: $\{T_b\}_1^B$.

The predictive regression looks as follows:

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

When running the random forest algorithm, it is important to note that the subsets can be extracted both with and without replacements.

In order to choose the number of $m$ variables that must be picked randomly; we would usually use the following rule of thumb for regression:

$$m = \left[\frac{p}{3}\right]$$

With the minimum node size being five. The default setting in the programming of this model follow the same intuition and will therefore not be specifically stated in the code.

Note that Cutler et al. (2007) reported that different values of $m$ did not affect the correct classification rates of the model and that other performance metrics were stable under different values of $m$. On the other hand, Strobl et al. (2008) reported that $m$ had a strong influence on predictor variable importance estimates.

# Empirical Analysis:

## Application:

First part of the analysis is to establish whether correlation is present between the twitter data and the financial markets. To do this, we clean the data and plot the correlation (See **Appendix 1**). It is clear to see that correlation is present. Especially for the 'Retweet_count' variable. This could possibly be because the 'retweet' shares the post to a larger group and therefore has a higher market impact than the other variables.

Next we scale the data and prepare for the neural networks by splitting the data into test and training set[6]. In the activity-based network, we get an accuracy of 0.99 on the training data, which is extremely high, as well as an MSE of 489.651. We therefore continue by cross validating the model, where we find an MSE of 635.390. The same is the case for the sentiment-based network, where we also get an accuracy of 0.99 and a MSE of 510.197 when predicting on the training set.

We cross validate this model in similar fashion as previously, to get an MSE of 755.431. See boxplot in **Appendix 3** for range of MSE in the various cross validation steps.

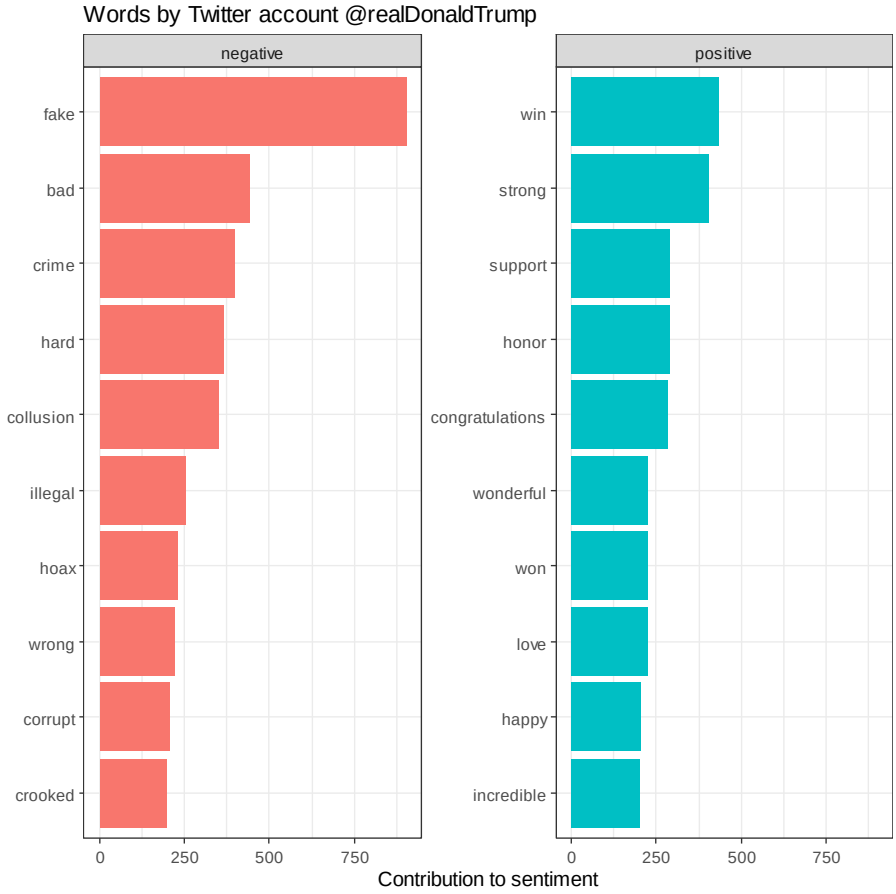
**Table 1. MSE of Neural Networks & Cross Validated Neural Networks**

| | **CV NN Quant** | **CV NN Sentiment** | **NN Quant** | **NN Sentiment** |
|---|---|---|---|---|
| **MSE** | 635.390 | 755.431 | 489.651 | 510.197 |

The text data is prepared by using the third-party package 'SentimentAnalysis' and is cleaned from frequently appearing stop words. One of these words is 'trump', which adds to the positive score, due to it being an adjective as well as a name. As expected, this word appears approximately 1500 times in the data, and therefore has a significant impact on the following computation when omitted.

[6] See both Neural Networks in **Appendix 6; Neural Networks**

The sentiments are quantified into two groups depending on their wording being negative or positive, and furthermore computed into a score dependent on the gravity of that word. The wording is scored by third party library 'bing' which is one of the most frequently used stop word libraries in sentiment analysis in R. Both the positive and negative top 10 words and count are presented below:




From the above plot alone, we see that there is a slight overweight in negative words, which explain the decreasing trend in the omitted variable 'MeanSentiment'.

To check whether these scores are valid regressors, we once again plot the correlation with the S&P500 in **Appendix 2.**

There is once again evident correlation between the independent and dependent variables.

In the neural networks we normalize and scale the data and specify a train and a test set for both models. The training data is split at 75%, which correspond to 409 observations, leaving 25% in the training set, which corresponds to 139 observations. This scaled and normalized data is the same I use for both the linear models as well as the random forest. This is primarily to avoid reading performance measures based on differently scaled data.

Furthermore, I use cross validation with ten loops, to specify a more accurate MSE for the neural networks.

When the model has computed we receive the following performance measures as output:

**Table 2. Model performance measures**

| | **LM Quant** | **LM Sentiment** | **NN Quant** | **NN Sentiment** |
|---|---|---|---|---|
| **BIC** | 5077.663 | 5135.466 | 179.458 | 180.501 |
| **AIC** | 5056.466 | 5109.650 | 67.074 | 68.117 |
| **MSE** | 604.35 | 666.61 | 635.39 | 622.63 |
| **$R^2$** | 0.3663 | 0.3024 | | |

Both linear models show highly significant results. Especially the 'Retweet_count' in the linear activity model has p-value below the 1% significance level, and both 'Favorite' and 'Frequency' are significant on the 5% level. This is not surprising as we already established the high correlation of the 'Retweet_count' variable in the beginning of the 'Application' chapter.

After cross validating the neural network models, I find that the activity-based network has a lower mean MSE than the neural network based on sentiment.

In **Table 2.** We also see how the linear activity model is doing better than the linear sentiment-based model, this might be because the simple OLS find it easier to extrapolate the correlation from the simple activity-based data, than it is to comprehend the sentiment-based data.

We know that the OLS is simply a measure of distance from the data to the dependent variable, it would therefore also be logical to infer that the better performance measures are visible due to the high correlation from the 'Retweet_count' variable.
In contradiction, we see that the neural network based on activity is doing worse than the neural network based on sentiments. This could potentially be a display of the strengths of the neural networks themselves, or simply a case of machine learning outperforming linear models.
It is important to note that the larger MSE of the activity-based network also means that it is less trusted than the sentiment -based neural network.

To figure whether these results are due to the models being good, or simply a case of luck, I extend my application by including random forest. This is foremost to help determining the strength of the neural network, but also to add a depth to the intuition behind the performance measures presented.

**Table 3. Random Forest performance measures**

| | **RF Quant** | **RF Sentiment** |
|---|---|---|
| **MSE** | 596.7436 | 614.7076 |
| **% Var Explained** | 36.33 | 34.13 |
| **$R^2$** | 0.1655254 | 0.1442621 |

We can see from the performance measures in **Table 3.**, that random forest does remarkably well in minimizing the error, relative to the other models previously presented. This adds to the notion that machine learning methods thrive in the designated data environment. The MSE for both models are well below the linear models and the networks[7], and a decent amount of the variance can be explained by the models.

[7] See **Appendix 5** for boxplots over the range of MSE provided by the Random Forest models.

Between the two random forests, we see that the activity-based data had better results than the sentiment-based, however, both models show relatively low fit when considering the linear models. This creates a conundrum, as there now exist a conflict in the performance measures. Two of the models have the activity-based data in higher regard than the sentiment-based. However, both models that favor the activity-based data have MSE's well below the contrasting models and is therefore in higher regard than the networks.

Moreover, it is interesting that the low MSE's and largely explained variance in the random forest, is accompanied by a small fit. In this situation, it is important to note that random forest with few regressors are prone to overfitting. A low $R^2$ is not inherently bad or unexpected when the data is human and noisy, but it would have proved difficult to properly predict using the random forest.

Despite both machine learning methods being prone to overfitting, the neural networks and the random forests has treated the two data types differently.

As noted above, we see how the activity-based data has displayed the better performance measures for the random forest and linear models. This can either be a sign of good backpropagation of the networks, or the strength of the random node-splitting.

I infer that the random forest provided valuable insights to the previously shown performance measures in **Table 2.**

I will continue to comment on random forest in the following extend of this paper if contributing to the overall understanding.

## Forecast:

We predict 5 days ahead to correspond a week of open markets for the S&P500.

The forecast shows great difference in results:

**Table 4. Forecast performance measures:**

| | **LM Quant** | **LM Sentiment** | **NN Quant** | **NN Sentiment** |
|---|---|---|---|---|
| **BIC** | 1405.900 | 1440.502 | 44.24 | 79.59 |
| **AIC** | 1391.300 | 1425.902 | 29.64 | 70.81 |
| **RMSE** | 13.9460 | 15.9304 | 0.0928 | 0.1178 |
| **Sigma** | 0.0501 | 0.0567 | 0.1838 | 0.2292 |

Both the BIC and the AIC scores imply that the activity-based forecast is the optimal model of the 4 presented. This is also implied by the 'root mean squared error' and the sigma.

Between the linear models and the neural networks, we see that the networks are better at predicting the S&P500. This is not surprising considering the literature on the subject, as well as the data available to us.

## Discussion:

What is perhaps most eye catching is the difference in the performance measures in the predicting state and the forecasting state. We saw that the models had similar MSE's in the prediction state, but when forecasting, we see that the RMSE are completely different from the linear models and the networks respectively.

This is not unusual, but it shows the cruciality of the type of model used, and not only the data, as well as the difficulty in making good forecasting models.

It also shows that the method and technique provide larger impact on the performance measures, than the data itself.

We see that the activity based neural network came in as the most optimal model of the ones presented, closely followed by the sentiment-based network. This is not surprising, considering the literature on the subject, but it does conflict the importance of sentiments. However, despite the model not outperforming, we have still confirmed that sentiments are valid as regressors and inputs, while also validating the idea of Bloomberg and Merrill Lynch initiatives. It shows that text-based data can be used in forecasting the financial markets, which adds to the toolbox of econometricians.

Both the adjusted price of the S&P500 and the frequency of the tweets are increasing over time, this could to some extend explain the model performance. If the frequency increases over time, there is more data available to create sentiments scores from, possibly skewing the cumulate scores of the day. This should however not be a big problem, considering all data was normalized, scaled and manually cleaned, but it does infer reason to conduct similar computation with different scaling.

Despite the various precautions in the methodology chapter, there might still be evidence of the neural network's overfitting. This can either be due to the rule of thumb, or the data was simply not noisy enough for the neural network to generalize from training data. We did use the test on a test set and cross validated, but there is still room for improvement and more comprehensive testing of the models.

Of the two linear models, we saw that the activity based OLS had a better fit according to the adjusted R^2 in the predictive state. This was backed up further by the BIC, AIC and MSE. In the forecasting state, that relationship has continued.

In contradiction, we see the opposite in the neural networks. In the predictive step, the sentiment-based network has higher BIC and AIC, but with a smaller MSE. This then changes in the forecast, where the activity-based network is performing better on all measures. This indicates the random forest was indeed a better model, as it followed the same dynamics in the predictive state as the linear models.

## Conclusion:

In this paper I found that twitter.com is a decent data source when forecasting the S&P500, and both the activity- and- sentiment based data is significant. However, I also found that the activity-based data perform better when forecasting using OLS, neural networks and Random forest.

In summary I found that Bollen et. Al.'s conclusion can be repeated on a smaller scale with less convoluted models, while also presenting lesser results. This argues in favor of more advanced neural network's strength in text-based forecasting environments, as well as the strength of machine learning in general.

I therefore conclude that the intentions of this paper have been answered to a satisfying degree.

To further this study, I would have chosen more twitter accounts to analyze, and created a portfolio of indices. I would also had included more methods, for example ARIMA or LASSO.

With many twitter accounts, it would furthermore be interesting to apply PLC to determine useful independent variables. One could also use multiple types of neural networks chosen in the literature and summarize the strengths of each approach.

If I had not summarized the frequency of twitter posts in a day, it would also have been possible to delve into the high frequency environment and used some version of the HAR model against a high frequency-based sentiment model.

## Sources:

### Papers & Articles:

Javier E. David (2019), *Trump tweets and market volatility have a 'statistically significant' relationship, BofA finds,* Yahoo Finance.

Michael A. Nielsen (2015) *Neural Networks and Deep learning,* Determination Press.

Arratia, A.: Arias, M. (2013) *Forecasting with Twitter Data*, ACM Transactions on Intelligent Systems and Technology ,5(1)

Mittal, A: Goel, A. (2011). *Stock prediction using twitter sentiment analysis*. Stanford CS229

Pang, B.: Lee, L. (2008) *Opinion mining and sentiment analysis,*Trends Inf. Retr., vol. 2, no. 1-2, pp. 1

Sipra, V. (2019). *Twitter Sentiment Analysis and Visualization using R*.

S. B. Hasan, K. (2019). *Stock Prediction Using Twitter.* towardsdatascience.com

Friedman, Tibshirani, Hastie (2009). *The Elements of Statistical Learning,* Springer-verlag New York Inc., 2. Ed.

Cutler et al. (2007), *Random Forest for Classification in Ecology*, Ecology, vol. 88, no. 11, pp. 2783-2792

Strobl et al. (2008), *Conditional variable importance for random forests,* BMC Bioinformatics, vol. 9, no. 307

Heaton, J. (2017) *The Number of Hidden Layers.* Heaton Research, Inc

Packages:

Wickham, H: François, R.: Henry, L.: Müller, K. (2020-03-07) “dplyr”, 0.8.5, https://cran.r-project.org/web/packages/dplyr/index.html

Wickham, H.: Hester, J.: Francois, R.: Jylänki, J.: Jørgensen, M. (2018-12-21) “readr”, 1.3.1, https://cran.r-project.org/web/packages/readr/index.html

A. Ryan, J.: M. Ulrich, J.: Thielen, W.: Teetor, P.: Bronder ,S. (2020-03-31) ”quantmod”, 0.4.17, https://cran.r-project.org/web/packages/quantmod/index.html

Hyndman, R.: Athanasopoulos, G.: Bergmeir, C.: Caceres, G.: Chhay, L.: O'Hara-Wild, M.: Petropoulos, F.: Razbash, S.: Wang, E.: Yasmeen, F.: Ihaka, R.: Reid , D.: Shaub, D.: Tang, Y.: Zhou, Z. (2020-03-31) “forecast”, 8.12, https://cran.r-project.org/web/packages/forecast/index.html

R Core Team and contributors worldwide, “stats”, 4.1.0, https://stat.ethz.ch/R-manual/R-devel/library/stats/html/00Index.html

Zeileis, A.:Grothendieck, G.:A. Ryan, J.:M. Ulrich, J.:Andrews, F.:(2020-01-10), “zoo”, 1.8-7, https://cran.r-project.org/web/packages/zoo/index.html

Feuerriegel, S.:Proellochs, N. (2019-03-26), “SentimentAnalysis”, 1.3-3, https://cran.r-project.org/web/packages/SentimentAnalysis/index.html

Auguie, B.:Antonov, A. (2017-09-09) “gridExtra”, 2.3, https://cran.r-project.org/web/packages/gridExtra/index.html

De Queiroz, D.:Fay, C.: Hvitfeldt, E.: Misra, K.:Mastny, T.: Erickson, J.:Robinson, D.: Silge, J.:(2020-04-17) ”tidytext”, 0.2.4, https://cran.r-project.org/web/packages/tidytext/index.html

Wickham, H.:Henry, L. (2020-01-24) “tidyr”, 1.0.2, https://cran.r-project.org/web/packages/tidyr/index.html

Wickham, H.: Chang, W.: Henry, L.: L. Pedersen, T.: Takahashi, K.: Wilke, C.: Woo, K.: Yutani, K.: Dunnington, D. (2020-03-05), “ggplot2”, 3.3.0, https://cran.r-project.org/web/packages/ggplot2/index.html

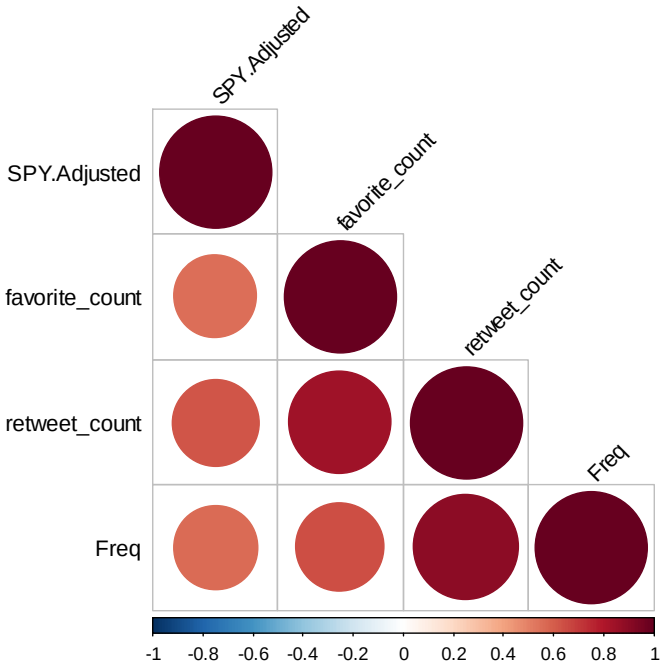
Naimi, B. (2019-02-03), “rts”, 1.0-49, https://cran.r-project.org/web/packages/rts/index.html

Wickham, H. (2020-03-03), “plyr”, 1.8.6, https://cran.r-project.org/web/packages/plyr/index.html

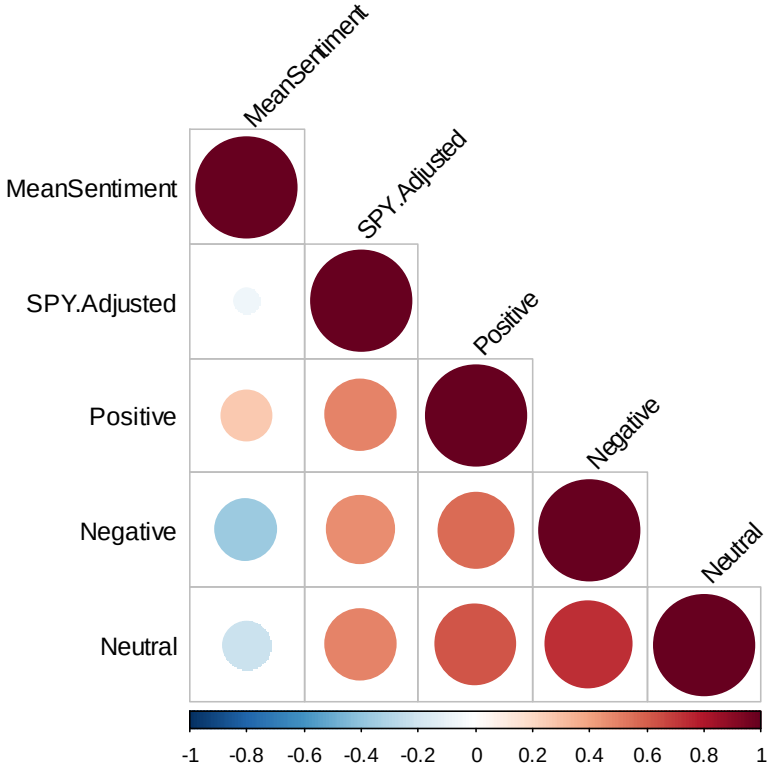
Fritsch, S.: Guenther, F.: N. Wright, M.: Suling, M.: M. Mueller, S. (20109-02-07), “neuralnet”, 1.44.2, https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf

## Appendix:

### Appendix 1: Correlation plot

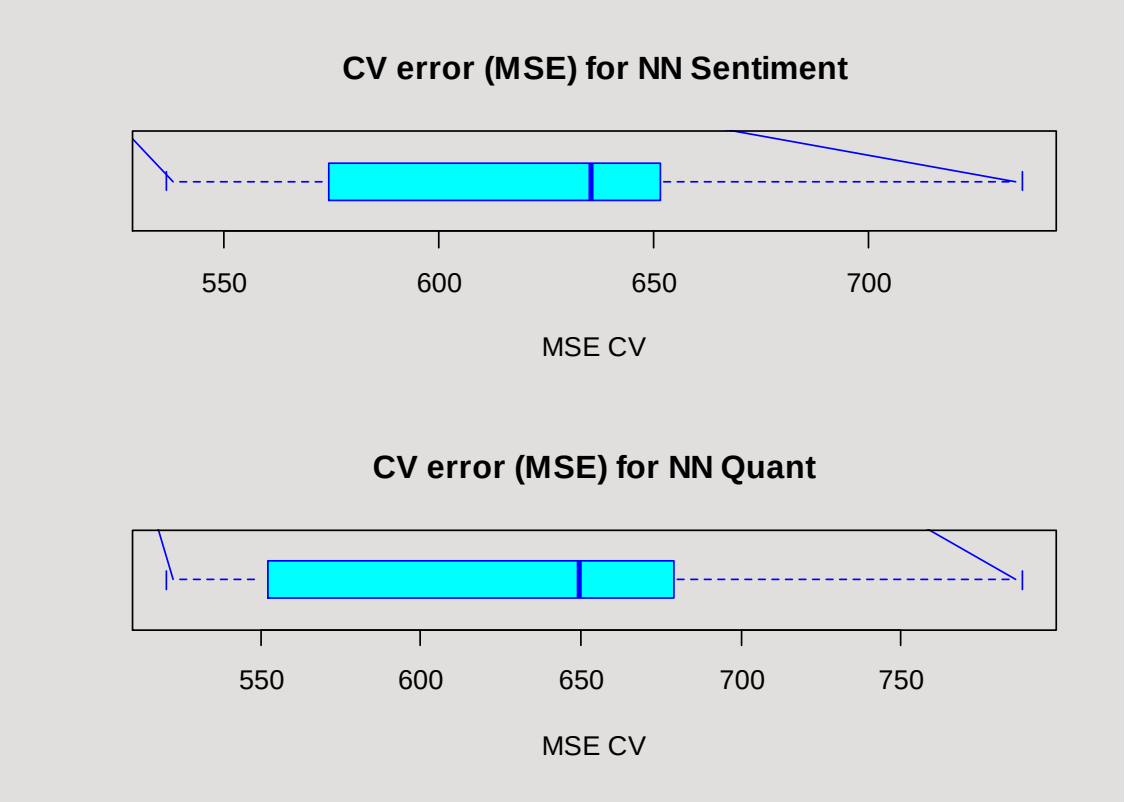



### Appendix 2: Correlation plot, Sentiment scores.



## Appendix 3: Boxplot of MSE cross validation

Below we see the cross validated mean squared error of the two neural network.´




## Appendix 4: Back propagation

From: A. Nielsen, M. (2015): *Neural Networks and Deep Learning*, Determination Press.

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \quad or \quad \delta^L = \sum'(z^L)\nabla_a C$$

$$\delta^l = \left((w^{l+1})^T \delta^{l+1}\right) \odot \sigma'(z^l) or \;\; \delta^l = \sum'(z^l)(w^{l+1})^T \delta^{l+1}$$

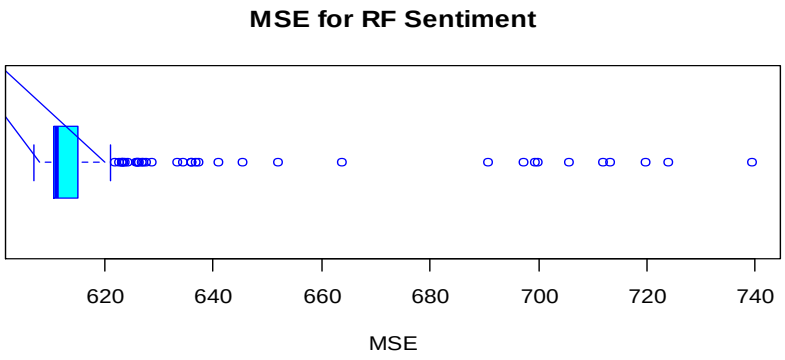
$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

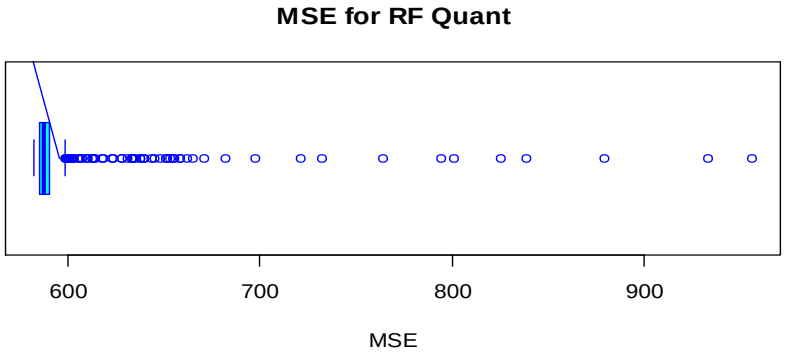
$$\frac{\partial C}{\partial w_{jk}^l} a_k^{l+1} \delta_j^l$$

By combining the top two equations, we can write $\delta^l$ as:

$$\delta^l = \sum'(z^l)(w^{l+1})^T \; \ldots \sum'(z^{L-1})(w^L)^T \sum'(z^L)\nabla_a C$$

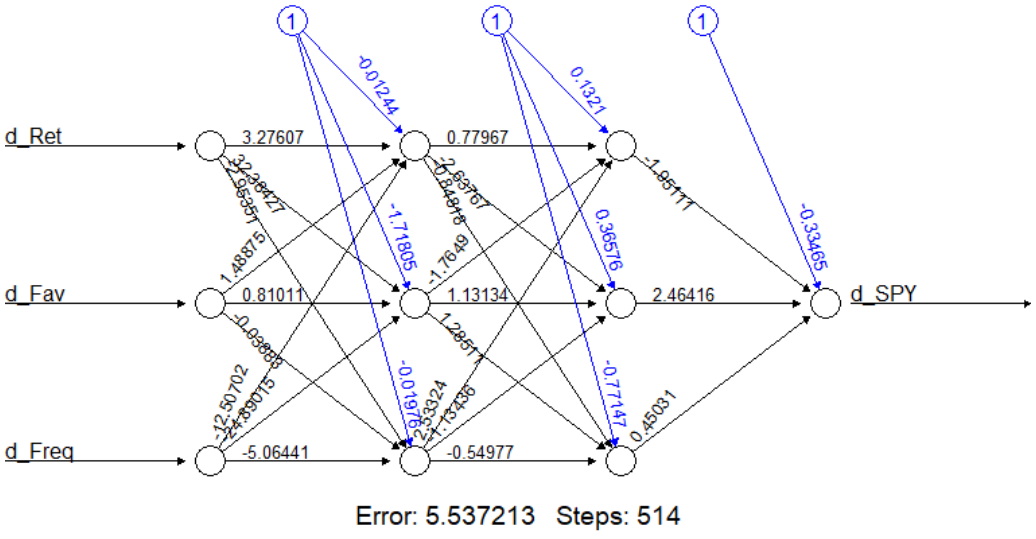
## Appendix 5: Boxplots over Random Forest MSE's



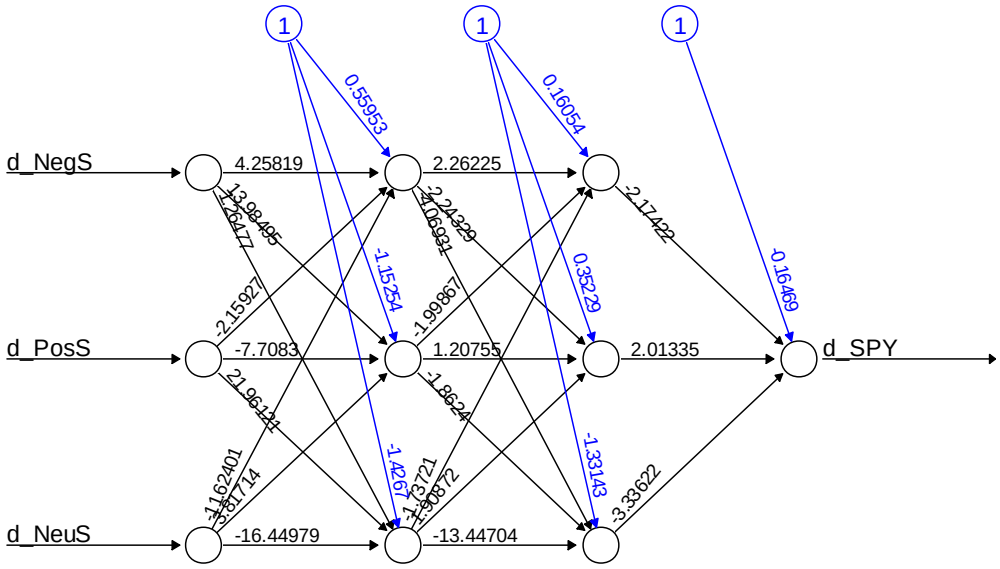




## Appendix 6: Neural Networks

1. The data is as follows: d_Ret = Retweet_count, d_Fav = Favorite_count, d_Freq=Frequency, d_SPY = S&P500



2. The data is as follows: d_NegS = Negative, d_PosS = Positive, d_NeuS = Neutral, d_SPY = S&P500




Error: 5.810556 Steps: 678