Group assignment ECON4170 Data science for economists

- By candidate numbers 171800, 171837 and 171849

Contents

1.Introduction

"Even apart from the instability due to speculation, there is the instability due to the characteristic of human nature that a large proportion of our positive activities depend on spontaneous optimism rather than on a mathematical expectation, whether moral or hedonistic or economic. Most, probably, of our decisions to do something positive, the full consequences of which will be drawn out over many days to come, can only be taken as a result of animal spirits—of a spontaneous urge to action rather than inaction, and not as the outcome of a weighted average of quantitative benefits multiplied by quantitative probabilities." (Keynes 1936).

The famous remark by John Maynard Keynes about the "*animal spirits*" speaks to the periodically irrational actions by investors in financial markets. The statement seems to resonate the belief from the field of behavioral finance - that the movements of the financial market reflect more than the impartial mathematical calculation of the master economist. The emotions of investors and the aggregate sentiment in a market can result in a difference between the observed and the intrinsic value of a security.

Nobel laureate Richard Thaler provides anecdotal evidence of such irrational market behavior by discussing the price development of The Herzfeld Caribbean Basin Fund (ticker name CUBA). On 18 December 2014, the Obama administration announced that they would be lifting sanctions on Cuba. This led to a massive increase in the price of a share in The Herzfeld Caribbean Basin Fund. A share in the fund grew from being worth 90% of the value of the underlying assets, to being worth 170% in a single day. This change happened despite the fund not owning any securities in Cuba (Thaler, 2016). Another piece of anecdotal evidence is the bitcoin bubble of late 2017. It appears that interest and optimism increased the price of bitcoin beyond the intrinsic value of the commodity. Figure 1 shows the trends of Normalized searches (hits) for bitcoin, and normalized price of bitcoin[1].

[1] Normalized by dividing the entire series of hits and prices, by the maximum in the series and multiplying by 100.

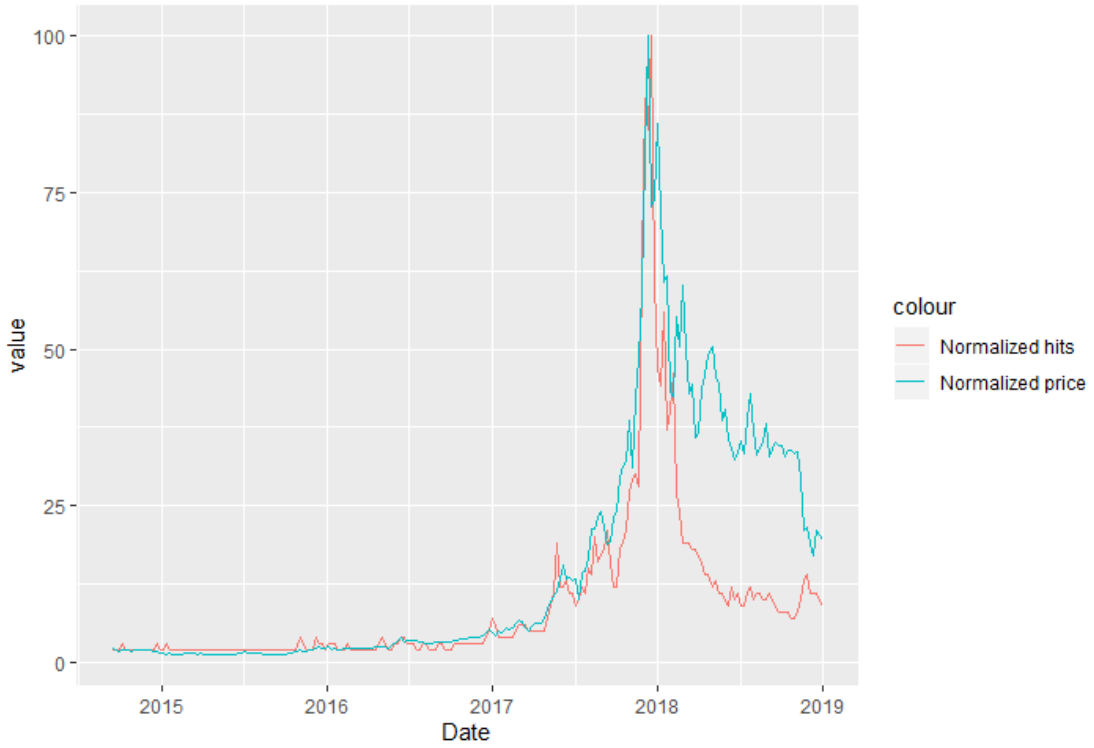

Figure 1. Bitcoin price and search trends

It appears that the animal spirits of Keynes, the optimism, the interest and the overall sentiment of investors can misprice assets. Knowing the possibility of market sentiment to affect market prices leads to the obvious question, how do you measure market sentiment and investors' emotions? This paper attempts to do so in two different ways. One compares Google trends and stock prices for a selection of companies. The other uses data on headlines of the most discussed news articles on reddit.com/r/worldnews and a measure of how positive or negative the headlines are to see how it is correlated with the Dow Jones Industrial Average.

2. Background

Preis et al. (2010) study Google trends data on stock prices and transaction volumes in the period 2004-2010, and they find that there is a significant correlation between volumes and Google trends, but no significant relationship between stock prices and Google trends. These findings are similar to what we find, even though there is a ten-year difference to our dataset. They do this by calculating correlation coefficients, slightly different to our regressions and methods of machine learning. Their conclusion is that the market is moved by news, but in either direction. Bilj et al. (2016) find that high Google trends lead to negative excess returns,

but the findings are too weak to use in a trading strategy when including transaction costs. They estimate a panel data regression where they include stock betas to calculate excess returns of a stock. Our method is more similar to Preis et al (2010) in terms of outcome variables, and results.

The last lecture of ECON4170 was about using textual data to perform sentiment analysis in financial markets, among other things. Vegard Høghaug Larsen, the guest lecturer, and his co-author Leif Anders Thorsrud discuss the relationship between news data and the stock market in their 2017 paper. They argue simple zero-cost news-based investment strategies yield significant annualized risk-adjusted returns of up to 20 percent. The method used by Thorsrud and Larsen is more complex than the sentiment analysis in this paper, which is closer to the method used in Cortis et al (2017) since we are only using headline data and not full articles.

3. Summary statistics and descriptive data

We are using data from three different sources, separated into two different datasets. The first dataset consists of two components: data about stock prices and volumes and Google search trends for nine companies[2]. The Google trends work in a very specific way: if the desired dataset is for a time interval larger than two months, it will only provide weekly data[3]. As the data is only available in a normalized way for free, we ended up using weekly data for 5 years (2014.01.01-2019.01.01). Stock prices and volumes were pulled from Yahoo! Finance. Financial markets are not open on Sundays, so we opted for the adjusted closing prices on Mondays for the same interval as Google trends. Table 1. and Table 2. show the summary statistics for normalized Google trends, and normalized price paths for each company. Figure 2 shows how these trends are developing over time for each company.

ticker	*Mean(hits)*	*Sd(hits)*	*Min(hits)*	*Max(hits)*
AAPL	8.282443	10.833307	0	100
AMZN	48.312977	13.351473	26	100
FB	51.236641	22.154398	22	100
GOOGL	58.706107	17.529217	36	100
MSFT	15.198473	11.622029	4	100
NHY	26.599237	9.902769	9	100
SALM	66.087786	13.852394	32	100
TEL	74.374046	7.960954	58	100
TSLA	20.103053	11.379927	6	100

[2] Microsoft, Apple, Google (Alphabet Inc.), Amazon, Tesla, Facebook, Salmar, Norsk Hydro and Telenor

[3] Derived as aggregate searches during Monday until Sunday, sampled at the end of the week).

Table 1. Summary statistics of the hits variable

ticker	Mean(Normalized)	Sd(Normalized)	Min(Normalized)	Max(Normalized)
AAPL	55.80255	17.25458	28.79633	100
AMZN	40.98534	23.68988	14.26236	100
FB	57.31554	19.67260	25.93598	100
GOOGL	64.06749	17.17062	39.96520	100
MSFT	53.53786	20.51927	28.06380	100
NHY	62.15557	15.93504	38.09724	100
SALM	37.80578	21.50570	10.41156	100
TEL	74.74365	11.48211	53.84996	100
TSLA	67.37999	14.47697	38.00235	100

Table 2. Summary statistics of the Normalized variable

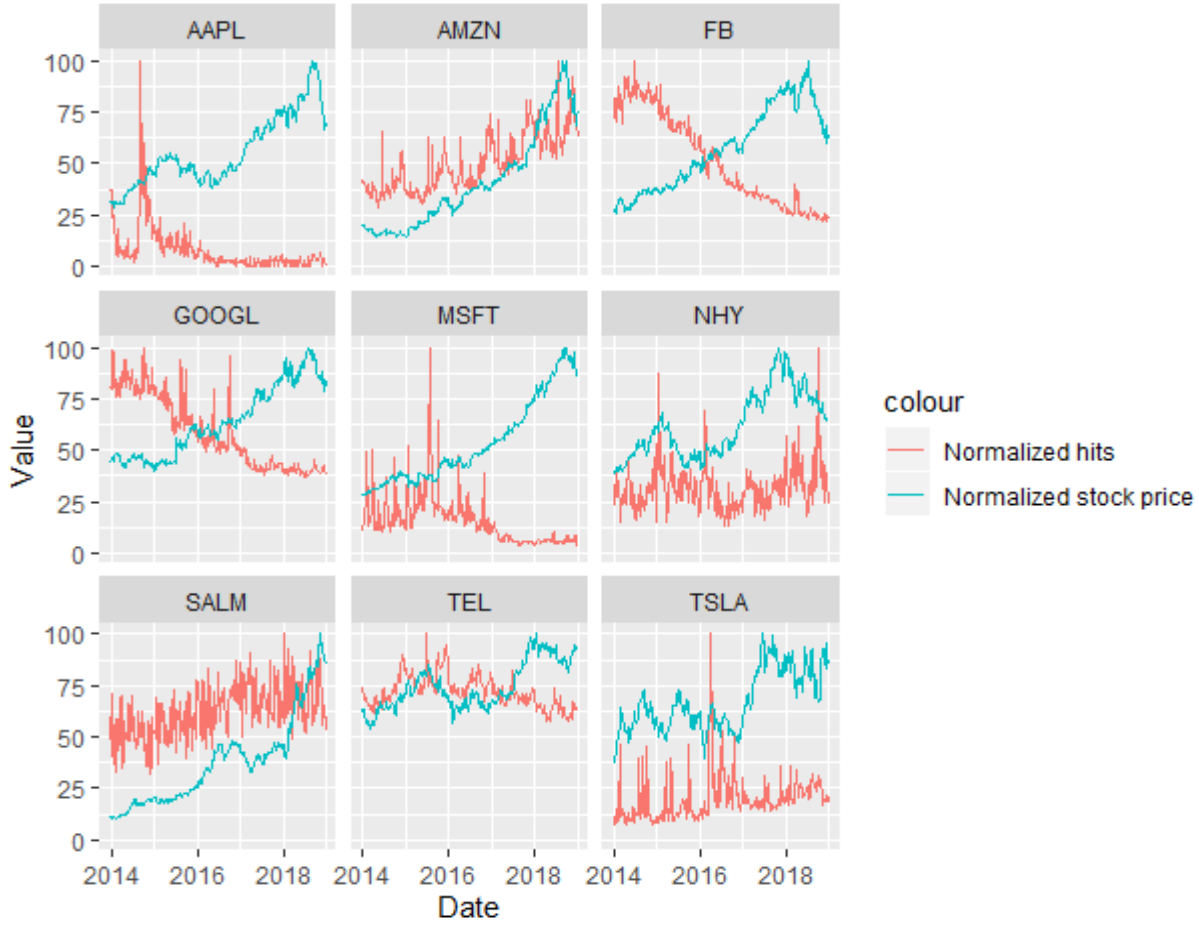

Figure 2. Trends of normalized hits, and normalized price for all companies

The second dataset contains headlines of the top 25 most read news articles pulled from reddit /r/worldnews in an 8-year-long interval (from mid-2008 to mid-2016). We downloaded a

dictionary from Harvard that gives numerical values to the different aspects of words. We changed this to contain only a dummy variable (where the value is 1, if the words has a positive meaning and 0 for negative) and got rid of all the duplicates to reduce the size of the dictionary. The news data was the trickiest to get in an analyzable shape, as we needed to stem the document, get rid of punctuations, numbers, abbreviations, etc.

4. LASSO and specification

Before testing any relationships between Google hits and stock price movements or stock price transaction, we do a graphical analysis. The scatterplot in Figure 3a gives an indication of no correlation between hits and stock price movements. The positions of the dots seem arbitrary and points to an almost non-existent correlation between the variables, not promising for our analysis to follow. When checking the same graphs on individual companies in Figure 3b, the picture looks slightly better for some companies. Especially the Facebook and Google stock prices seem to have a significant negative correlation with the Hits. Doing the same graphical analysis on Volume, in Figure 4a the scatterplot shows a slight upward trend of Volume relative to hits. It is possible there is a weak correlation between Volume and hits. This weak correlation is reflected in the individual company scatterplots in Figure 4b, which seem to paint the same picture.

Our paper is a data science paper, and our focus is to predict. Therefore, we have skipped the econometric misspecification tests. The main goal is instead to use OLS regressions as a tool to compare with our chosen machine learning methods. To test whether there are any relationships between Google trends and stock price movements or stock price transaction volumes we split our sample into a training sample and test sample, where the test sample is 1/5 of our full sample. The training sample is the one we will do our main analysis on, both for regular OLS regressions and LASSO.

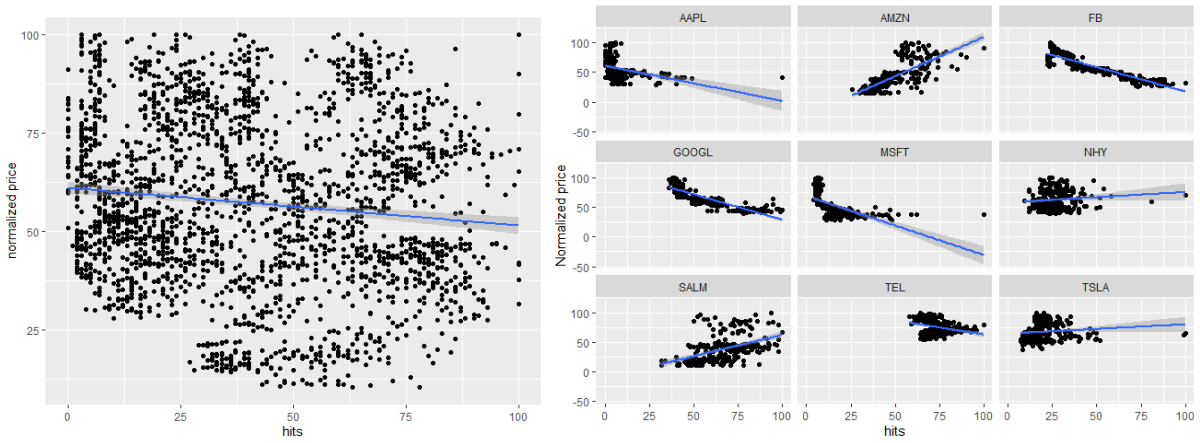

Figure 3a (left): Google hits and normalized price, aggregate.
Figure 3b (right): Google hits and normalized price, individual companies.

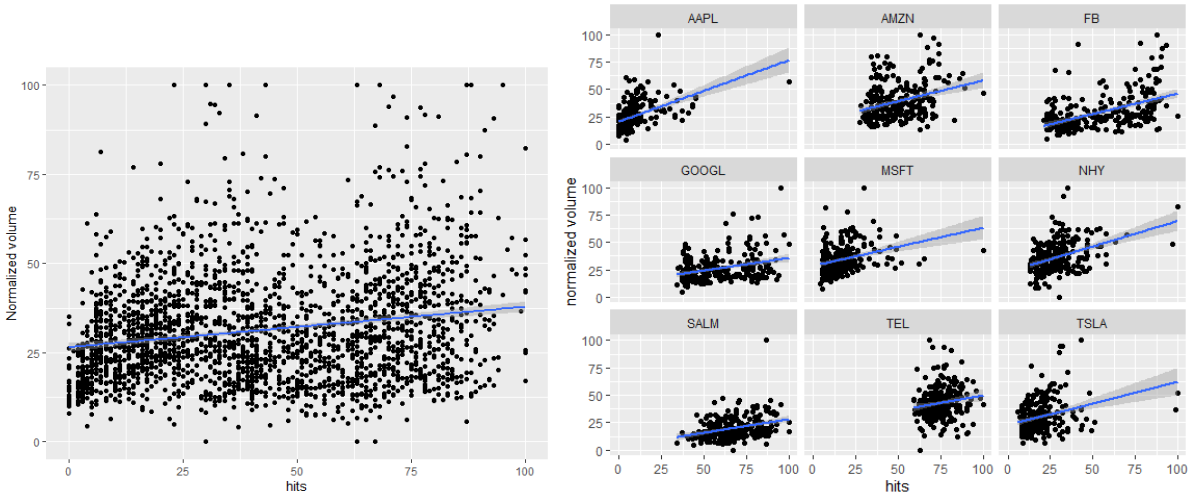

Figure 4a (left): Google hits and normalized transaction volume, aggregate.
Figure 4b (right): Google hits and normalized transaction volume, individual companies.

In the OLS regressions we have the adjusted closing prices of the stocks and transaction volumes of the stocks as our outcome variables. Our main goal is to see whether the Google hits or Google hits in the weeks before can help predicting movements in the stock price in either direction or in the transaction volumes. Because we have time series data, we will also include lags of the outcome variables as explanatory variables, in addition to the other outcome variable. Our main data is normalised as explained in the section on summary statistics and how we collected the data. We perform the following regressions:

$$Price_t = \alpha + \beta_1 Price_{t-1} + \beta_2 Price_{t-2} + \gamma_1 Hits_t + \gamma_2 Hits_{t-1} + \gamma_3 Hits_{t-2} + \delta Volume_t + \varepsilon_t$$

$$Volume_t = \alpha + \beta_1 Volume_{t-1} + \gamma_1 Hits_t + \gamma_2 Hits_{t-1} + \gamma_3 Hits_{t-2} + \delta Price_t + \varepsilon_t$$

The regressions are based on the variables we consider to be economically relevant to the outcome variables. Based on inference we will conclude whether the regressors perform well in explaining the outcomes in-sample. Table A1 and Table A2 in the appendix display the regression results on the individual companies, with beta coefficients and the column "sign" indicating whether the coefficient is significant on the 5% level. In the regression on price in Table A1 we find that for most companies neither Hits nor its own lags are significant in explaining changes in stock prices. In fact, the only coefficients that are significant are mostly the lag of price, as well as the volume in some instances. Doing the regression on volume in Table A2, we find that this regression performs better. Now Hits is significant for almost all the companies. Additionally, either the first or second lags of Hits are also significant for several of the companies. This is after controlling for price and lags of volume. The fact that Hits can help explain Volume, but not the change in stock price is the same conclusion that Preis et al. (2010) reaches in their paper.

We will now use LASSO to see if we can either draw any new conclusions or strengthen our OLS results. One advantage with LASSO is that it works well with a lot of variables. Unfortunately, we do not have too many variables in our dataset to test on, but we can include a mix of lags of price, lags of volume, and lags of hits. LASSO penalises an overfitted model, and our hope is that it will exclude variables without any explanatory power. We do the cross-validation method to find optimal lambdas for LASSO for each company. By optimal lambda we mean to find the value of the shrinkage parameter that minimises the mean squared error of the model. The results of each company are displayed in Table A3 and Table A4 in the appendix. The "sign" column indicates whether a variable was included in the final model or not. Lasso on price in Table A3 gives us a result with only Facebook and Google to have Hits included in the final model, corresponding with our graphical analysis in Figure 3b. For volume on the other hand, we get that Hits or lag of Hits are included in Apple, Amazon, Facebook, Norsk Hydro and Tesla, but not for Google, Microsoft, Salmar or Telenor. Hence, it works only in about half of the companies we have data on, and the two groups do not necessarily have different company characteristics.

How well can our analyses from the training sample predict the outcomes of the test sample? We check this by comparing our predicted outcome variables of the test sample with the actual outcome variables of the test sample. The mean squared error gives us the average of

the squared differences between predicted and actual values. It gives us a method of comparing our models of predicting price, and our models of predicting volume. Table 3 below displays the MSE for our models of the individual companies. For our price models, there is an indication that OLS performs better than LASSO, having a lower MSE for all companies except Norsk Hydro and Telenor. On the other hand, OLS performs more poorly than LASSO when predicting volume, having a lower MSE for only Amazon and Norsk Hydro.

Now we will discuss some of the weaknesses to our models. Firstly, we do not have a large sample. Google Trends only has weekly data, and for our analysis it would have been optimal with daily data to get the daily changes in stock prices. The data is from the period 2014-2018 because we think that Google search behaviour was different before, hence there is a trade-off between getting either more reliable data or more observations. In the 2014-2018 sample for a company there are only 262 observations. Splitting our already small sample to do cross-validation can make our analysis sensitive to outliers in the subsamples. Secondly, it would have been more optimal if we had more variables to choose from. Variables that could have been helpful to predict stock prices or volumes are for example country-specific, industry-specific or company-specific controls. Without relevant variables included, OLS will give us biased coefficients. Lasso would potentially have performed better as well, by ending up with more relevant variables in the final result. Thirdly, our results are extremely dependent on which method we use or which company we analyse. It would have been interesting to do similar analyses on all companies in S&P500 or on Oslo Stock Exchange. Then we could have tried to find similarities between the companies that have volumes or stock price correlating with the amount of Google searches. It would also have been interesting to find out for how many companies this is the case.

ticker	*mse_price_ols*	*mse_price_lasso*	*mse_vol_ols*	*mse_vol_lasso*
AAPL	3.537	3.595	75.531	66.099
AMZN	4.624	4.748	193.079	199.784
FB	3.785	4.006	99.055	87.575
GOOGL	3.861	3.983	78.401	75.224
MSFT	2.993	3.186	108.780	107.028
NHY	5.596	5.492	94.801	94.084
SALM	4.387	3.667	50.914	51.012
TEL	5.338	5.671	196.325	189.436
TSLA	18.673	18.792	164.726	163.629

Table 3. Mean squared error for OLS and LASSO when predicting normalized stock price and normalized volume on the individual companies

5. A tree and a forest

It could be that high amounts of interest in a given company might not result in monotonic effects on the stock price. This is because high amounts of interest can arise from both positive and negative reasons. Let us now rather look at the relationship between google trends and volatility. That public interest in any given company might lead to large changes in price, and if it is possible to use information on google searches to predict if any given week is highly volatile.

We will use a classification tree from R package "rpart", and the random forest method from the package "randomForest". The classification tree works by splitting the data along the explanatory variables in a way that maximizes the probability of finding a particular value of the outcome variable on one side of the split. The random forest method does something similar to the classification tree, but with multiple parallel trees, where the trees are randomly assigned explanatory variables, and the result from each tree is weighted differently.

The first step of the analysis is to decide what constitutes a highly volatile week. We define a highly volatile week as being a member of the top 10% volatile weeks for a company. The most volatile weeks are assigned the value 1, and the rest are assigned zeros. The second step is to divide the dataset into a training dataset, and a test dataset. The third is to train both the classification tree and the random forest on the training dataset and use the trained models to predict the outcome variable in the test dataset. Tables 4.a and 4.b show the resulting confusion matrices after applying the classification tree and random forest methods to all firms at once.

	Predicted (Classification tree)		
Actual		0	1
	0	387	0
	1	90	0

	Predicted (random forest)		
Actual		0	1
	0	387	0
	1	90	0

Tables 4.a(left) & 4.b(right): Confusion matrices from predicting the test data, analyzing all firms at once

As is evident from tables 4.a and 4.b, both methods just guess zeros for all data points. Which is a strong indication that there is overarching relationships between google searches and volatility. These results are similar to the once in tables 5.a and 5.b, that uses variation of

google searches within each company. Both methods seem more inclined to predict high volatility weeks, but neither do a good job of predicting overall.

	Predicted (Classification tree)		
Actual		0	1
	0	384	3
	1	89	1

	Predicted (random forest)		
Actual		0	1
	0	380	7
	1	87	3

Tables 5.a(left) & 5.b(right): Confusion matrices from predicting the test data, analyzing the firms separately.

The results will vary depending on what seed one are using, since it dictates both what observations are included in the training and test samples, as well as some parts the randomForest function. The main trends is that the random forest was more inclined to guess high volatility weeks, but suffered due to limited sample sizes. As a final test, we attempted to use the same method on bitcoin. Table 6.a and 6.b seem to indicate that the methods did a lot better when analysing bitcoin, even though the sample size is quite small.

	Predicted (Classification tree)		
Actual		0	1
	0	26	0
	1	1	1

	Predicted (random forest)		
Actual		0	1
	0	26	0
	1	0	2

Tables 6.a(left) & 6.b(right): Confusion matrices from predicting the test data, analyzing only bitcoin

6. Text Analysis

In this part of the project, our goal was to find a relation between world news and stock movements. As the data would be too large to handle and due to our difficulties getting them properly, we choose to analyze only the news headlines and the Dow Jones Industrial Average index, which consists of the 30 largest US traded companies by market capitalization. The dictionary acquired from Harvard specifies words in a much larger detail then we ended up using: our code takes creates a dummy variable for every word and assigns the value of 1 if it

has a positive meaning, 0 if negative, but leaving out every word that does not have any of those values (for example freeze is considered a neither positive nor negative word). Some words had multiple meanings, in these cases we got rid of the duplicates. The news data was modified greatly as well. For example, the following sentence: "Georgia 'downs two Russian warplanes' as countries move to brink of war" becomes “war war” in the vector containing all words and disappears from the vector completely when we only include positive words. This is something we can work with. The inclusion of all the step-by-step changes of the news data might not be efficient, we could have filtered through the dictionary right away (which step by the way is probably the largest one in terms of computational needs in the project as it takes several minutes to run), but we wanted to show how the features work and that we can use them. We set all the characters to lowercase, remove punctuations and numbers and use the built-in function in the SnowballC package to stem them (we tried different stemming functions from other packages but this seemed to be the best), and finally replace abbreviations, contractions and symbols with words. The document-term matrices include for each day, how many times the certain words appeared. We add the rows for both the base matrix and the one only including the positive terms. Interestingly, despite 44% of the words being positive in the original dictionary, the mean of positive words per day is less than 31%. The naïve interpretation of this result is that the media puts too much weight on covering bad news, but as we discussed in the lecture, the analysis on the level of words might be insufficient (high crime rates regards high as positive for instance). In comparison, the Dow Jones index increased by an average of 0.03% daily. After plotting the graphs, it was clear, we will not get any correlation between our score of the ratio between positivity of the daily words and the relevant Dow Jones price change, as we can see values everywhere without a visible pattern. We setup a linear regression model and our fears were proven right: no correlation can be shown from the data.

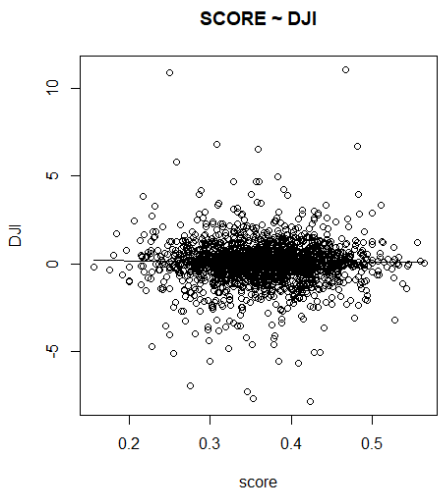

Figure 5. Plot of our score model and Dow Jones

7. Conclusion

The fundamental goal of this paper was to investigate the relationship between market sentiment and price changes in financial markets. We gathered data on google trends for companies and global headlines over multiple years, as well as financial data. We divided the data into two different datasets. We used regression, LASSO and classification methods to look at the relationship between google trends and prices of stocks. Afterwards we used textual sentiment analysis to look at the relationship between headline data and changes to a large stock index. We little to no evidence of google trends and headlines being indicators of movements in stock prices, regardless of the methods that were used.

Appendix A, Tables

Table A1: OLS regression with normalized stock price as outcome variable. One "l" in front of variable name indicates one lag, two "l"s indicates two lags, and so on.

term	*estimate*	*std.error*	*statistic*	*p.value*	*ticker*	*sign*
(Intercept)	2.229	0.810	2.752	0.006	AAPL	1
lNormalized	0.959	0.075	12.826	0.000	AAPL	1
llNormalized	0.017	0.074	0.231	0.818	AAPL	0
norm_volume	-0.030	0.015	-1.966	0.051	AAPL	1
hits	0.003	0.035	0.074	0.941	AAPL	0
lhits	-0.009	0.028	-0.313	0.754	AAPL	0
llhits	0.003	0.032	0.102	0.919	AAPL	0
(Intercept)	0.898	0.594	1.512	0.132	AMZN	0
lNormalized	0.901	0.065	13.906	0.000	AMZN	1
llNormalized	0.098	0.065	1.490	0.138	AMZN	0
norm_volume	-0.029	0.008	-3.608	0.000	AMZN	1
hits	-0.006	0.016	-0.384	0.702	AMZN	0
lhits	0.006	0.017	0.325	0.745	AMZN	0
llhits	0.010	0.015	0.623	0.534	AMZN	0
(Intercept)	5.448	2.307	2.361	0.019	FB	1
lNormalized	0.740	0.068	10.836	0.000	FB	1
llNormalized	0.205	0.068	3.022	0.003	FB	1
norm_volume	-0.039	0.012	-3.348	0.001	FB	1
hits	-0.062	0.048	-1.300	0.195	FB	0
lhits	-0.004	0.052	-0.071	0.944	FB	0
llhits	0.044	0.048	0.936	0.350	FB	0
(Intercept)	6.046	2.377	2.543	0.012	GOOGL	1
lNormalized	0.816	0.068	12.072	0.000	GOOGL	1
llNormalized	0.135	0.066	2.039	0.043	GOOGL	1
norm_volume	-0.018	0.014	-1.342	0.181	GOOGL	0
hits	-0.015	0.024	-0.634	0.527	GOOGL	0
lhits	-0.025	0.024	-1.073	0.284	GOOGL	0
llhits	0.001	0.023	0.034	0.973	GOOGL	0
(Intercept)	1.813	0.559	3.245	0.001	MSFT	1
lNormalized	0.813	0.065	12.486	0.000	MSFT	1
llNormalized	0.178	0.065	2.718	0.007	MSFT	1
norm_volume	-0.030	0.008	-3.663	0.000	MSFT	1
hits	0.020	0.012	1.665	0.097	MSFT	0
lhits	-0.036	0.013	-2.768	0.006	MSFT	1
llhits	0.010	0.013	0.818	0.414	MSFT	0
(Intercept)	1.935	1.078	1.794	0.074	NHY	0
lNormalized	1.011	0.072	14.130	0.000	NHY	1
llNormalized	-0.028	0.072	-0.386	0.700	NHY	0
norm_volume	0.005	0.015	0.294	0.769	NHY	0
hits	-0.033	0.018	-1.833	0.068	NHY	0
lhits	-0.005	0.020	-0.254	0.800	NHY	0
llhits	0.009	0.017	0.542	0.589	NHY	0
(Intercept)	0.432	0.798	0.541	0.589	SALM	0
lNormalized	1.160	0.077	14.978	0.000	SALM	1
llNormalized	-0.157	0.078	-2.029	0.044	SALM	1
norm_volume	-0.011	0.014	-0.794	0.428	SALM	0
hits	0.012	0.012	1.026	0.306	SALM	0
lhits	-0.007	0.012	-0.588	0.557	SALM	0
llhits	-0.006	0.012	-0.549	0.584	SALM	0
(Intercept)	2.061	2.259	0.912	0.363	TEL	0
lNormalized	0.934	0.067	13.906	0.000	TEL	1
llNormalized	0.049	0.067	0.735	0.463	TEL	0
norm_volume	-0.021	0.011	-1.904	0.058	TEL	0
hits	-0.005	0.037	-0.137	0.891	TEL	0
lhits	-0.066	0.045	-1.463	0.145	TEL	0
llhits	0.073	0.038	1.903	0.058	TEL	0
(Intercept)	3.030	1.476	2.052	0.041	TSLA	1
lNormalized	0.985	0.069	14.198	0.000	TSLA	1
llNormalized	-0.052	0.069	-0.750	0.454	TSLA	0
norm_volume	0.039	0.022	1.753	0.081	TSLA	0
hits	0.006	0.030	0.191	0.849	TSLA	0
lhits	0.020	0.035	0.581	0.562	TSLA	0
llhits	-0.009	0.031	-0.308	0.758	TSLA	0

Table A2: OLS regression with normalized stock transaction volume as outcome variable. One "l" in front of variable name indicates one lag, two "l"s indicates two lags, and so on.

term	*estimate*	*std.error*	*statistic*	*p.value*	*ticker*	*sign*
(Intercept)	18.642	3.607	5.168	0.000	AAPL	1
lVolume	0.478	0.071	6.736	0.000	AAPL	1
Normalized	-0.135	0.044	-3.084	0.002	AAPL	1
hits	0.234	0.146	1.595	0.112	AAPL	0
lhits	-0.201	0.120	-1.679	0.095	AAPL	0
llhits	0.286	0.136	2.100	0.037	AAPL	1
(Intercept)	8.837	4.537	1.948	0.053	AMZN	0
lVolume	0.515	0.062	8.315	0.000	AMZN	1
Normalized	-0.022	0.068	-0.330	0.742	AMZN	0
hits	0.341	0.122	2.798	0.006	AMZN	1
lhits	0.157	0.131	1.200	0.232	AMZN	0
llhits	-0.295	0.120	-2.458	0.015	AMZN	1
(Intercept)	5.726	11.474	0.499	0.618	FB	0
lVolume	0.600	0.057	10.490	0.000	FB	1
Normalized	-0.014	0.112	-0.121	0.904	FB	0
hits	1.035	0.231	4.472	0.000	FB	1
lhits	-0.645	0.260	-2.482	0.014	FB	1
llhits	-0.262	0.233	-1.124	0.262	FB	0
(Intercept)	-2.486	11.033	-0.225	0.822	GOOGL	0
lVolume	0.487	0.065	7.538	0.000	GOOGL	1
Normalized	0.079	0.092	0.861	0.390	GOOGL	0
hits	0.344	0.106	3.244	0.001	GOOGL	1
lhits	-0.295	0.109	-2.692	0.008	GOOGL	1
llhits	0.133	0.104	1.282	0.201	GOOGL	0
(Intercept)	16.363	4.473	3.658	0.000	MSFT	1
lVolume	0.455	0.072	6.342	0.000	MSFT	1
Normalized	-0.013	0.052	-0.245	0.806	MSFT	0
hits	0.173	0.095	1.824	0.070	MSFT	0
lhits	-0.027	0.104	-0.255	0.799	MSFT	0
llhits	0.051	0.098	0.519	0.604	MSFT	0
(Intercept)	15.995	4.469	3.579	0.000	NHY	1
lVolume	0.516	0.063	8.197	0.000	NHY	1
Normalized	-0.147	0.051	-2.889	0.004	NHY	1
hits	0.402	0.068	5.886	0.000	NHY	1
lhits	-0.066	0.084	-0.779	0.437	NHY	0
llhits	0.022	0.070	0.315	0.753	NHY	0
(Intercept)	0.509	3.748	0.136	0.892	SALM	0
lVolume	0.359	0.067	5.383	0.000	SALM	1
Normalized	0.054	0.036	1.490	0.138	SALM	0
hits	0.134	0.055	2.441	0.016	SALM	1
lhits	0.039	0.057	0.692	0.490	SALM	0
llhits	0.003	0.056	0.053	0.958	SALM	0
(Intercept)	37.336	13.450	2.776	0.006	TEL	1
lVolume	0.373	0.073	5.091	0.000	TEL	1
Normalized	-0.151	0.084	-1.788	0.075	TEL	0
hits	0.448	0.221	2.028	0.044	TEL	1
lhits	-0.530	0.275	-1.930	0.055	TEL	0
llhits	0.096	0.231	0.416	0.678	TEL	0
(Intercept)	4.423	4.071	1.087	0.279	TSLA	0
lVolume	0.451	0.059	7.591	0.000	TSLA	1
Normalized	0.145	0.059	2.442	0.015	TSLA	1
hits	0.375	0.084	4.480	0.000	TSLA	1
lhits	-0.145	0.101	-1.434	0.153	TSLA	0
llhits	-0.127	0.085	-1.491	0.138	TSLA	0

Table A3: Lasso of with normalized stock price as outcome variable. One "l" in front of variable name indicates one lag, two "l"s indicates two lags, and so on.

term	*coefficient*	*ticker*	*sign*
(Intercept)	3.963	AAPL	1
lNormalized	0.931	AAPL	1
llNormalized	0.000	AAPL	0
norm_volume	0.000	AAPL	0
lVolume	0.000	AAPL	0
hits	0.000	AAPL	0
lhits	0.000	AAPL	0
llhits	0.000	AAPL	0
(Intercept)	1.752	AMZN	1
lNormalized	0.953	AMZN	1
llNormalized	0.009	AMZN	1
norm_volume	0.000	AMZN	0
lVolume	0.000	AMZN	0
Hits	0.000	AMZN	0
Lhits	0.000	AMZN	0
Llhits	0.000	AMZN	0
(Intercept)	6.853	FB	1
lNormalized	0.774	FB	1
llNormalized	0.131	FB	1
norm_volume	0.000	FB	0
lVolume	0.000	FB	0
Hits	-0.025	FB	1
Lhits	0.000	FB	0
Llhits	0.000	FB	0
(Intercept)	5.439	GOOGL	1
lNormalized	0.834	GOOGL	1
llNormalized	0.092	GOOGL	1
norm_volume	0.000	GOOGL	0
lVolume	0.000	GOOGL	0
Hits	-0.005	GOOGL	1
Lhits	-0.004	GOOGL	1
Llhits	0.000	GOOGL	0
(Intercept)	1.942	MSFT	1
lNormalized	0.869	MSFT	1
llNormalized	0.098	MSFT	1
norm_volume	0.000	MSFT	0
lVolume	0.000	MSFT	0
Hits	0.000	MSFT	0
Lhits	0.000	MSFT	0
Llhits	0.000	MSFT	0
(Intercept)	5.227	NHY	1
lNormalized	0.919	NHY	1
llNormalized	0.000	NHY	0
norm_volume	0.000	NHY	0
lVolume	0.000	NHY	0
Hits	0.000	NHY	0
Lhits	0.000	NHY	0
Llhits	0.000	NHY	0
(Intercept)	1.981	SALM	1
lNormalized	0.955	SALM	1
llNormalized	0.000	SALM	0
norm_volume	0.000	SALM	0
lVolume	0.000	SALM	0
Hits	0.000	SALM	0
Lhits	0.000	SALM	0
Llhits	0.000	SALM	0
(Intercept)	5.652	TEL	1
lNormalized	0.911	TEL	1
llNormalized	0.015	TEL	1
norm_volume	0.000	TEL	0
lVolume	0.000	TEL	0
Hits	0.000	TEL	0
Lhits	0.000	TEL	0
Llhits	0.000	TEL	0
(Intercept)	11.961	TSLA	1
lNormalized	0.824	TSLA	1
llNormalized	0.000	TSLA	0
norm_volume	0.000	TSLA	0
lVolume	0.000	TSLA	0
Hits	0.000	TSLA	0
Lhits	0.000	TSLA	0
Llhits	0.000	TSLA	0

Table A4: LASSO with normalized stock transaction volume as outcome variable. One "l" in front of variable name indicates one lag, two "l"s indicates two lags, and so on.

term	*coefficient*	*ticker*	*sign*
(Intercept)	17.164	AAPL	1
lVolume	0.309	AAPL	1
llVolume	0.070	AAPL	1
Normalized	-0.037	AAPL	1
lNormalized	0.000	AAPL	0
hits	0.000	AAPL	0
lhits	0.000	AAPL	0
llhits	0.067	AAPL	1
(Intercept)	26.952	AMZN	1
lVolume	0.291	AMZN	1
llVolume	0.000	AMZN	0
Normalized	0.000	AMZN	0
lNormalized	0.000	AMZN	0
Hits	0.005	AMZN	1
Lhits	0.000	AMZN	0
Llhits	0.000	AMZN	0
(Intercept)	13.336	FB	1
lVolume	0.377	FB	1
llVolume	0.088	FB	1
Normalized	0.000	FB	0
lNormalized	0.000	FB	0
Hits	0.032	FB	1
Lhits	0.000	FB	0
Llhits	0.000	FB	0
(Intercept)	22.177	GOOGL	1
lVolume	0.129	GOOGL	1
llVolume	0.000	GOOGL	0
Normalized	0.000	GOOGL	0
lNormalized	0.000	GOOGL	0
Hits	0.000	GOOGL	0
Lhits	0.000	GOOGL	0
Llhits	0.000	GOOGL	0
(Intercept)	28.678	MSFT	1
lVolume	0.157	MSFT	1
llVolume	0.000	MSFT	0
Normalized	0.000	MSFT	0
lNormalized	0.000	MSFT	0
Hits	0.000	MSFT	0
Lhits	0.000	MSFT	0
Llhits	0.000	MSFT	0
(Intercept)	18.585	NHY	1
lVolume	0.361	NHY	1
llVolume	0.017	NHY	1
Normalized	0.000	NHY	0
lNormalized	-0.005	NHY	1
Hits	0.152	NHY	1
Lhits	0.000	NHY	0
Llhits	0.000	NHY	0
(Intercept)	19.671	SALM	1
lVolume	0.000	SALM	0
llVolume	0.000	SALM	0
Normalized	0.000	SALM	0
lNormalized	0.000	SALM	0
Hits	0.000	SALM	0
Lhits	0.000	SALM	0
Llhits	0.000	SALM	0
(Intercept)	42.695	TEL	1
lVolume	0.000	TEL	0
llVolume	0.000	TEL	0
Normalized	0.000	TEL	0
lNormalized	0.000	TEL	0
Hits	0.000	TEL	0
Lhits	0.000	TEL	0
Llhits	0.000	TEL	0
(Intercept)	19.555	TSLA	1
lVolume	0.263	TSLA	1
llVolume	0.000	TSLA	0
Normalized	0.016	TSLA	1
lNormalized	0.000	TSLA	0
Hits	0.071	TSLA	1
Lhits	0.000	TSLA	0
Llhits	0.000	TSLA	0

Appendix B, References

Keynes, J. M. (1936). *The general theory of employment, interest and money (1936)*. Kessinger Publishing.161-162.

Thaler, R. H. (2016). Behavioral economics: Past, present, and future. *American Economic Review*, *106*(7), 1577-1600.

Preis, T., Reith, D., & Stanley, H. E. (2010). Complex dynamics of our economic life on different scales: insights from search engine query data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *368*(1933), 5707-5719.

Bijl, L., Kringhaug, G., Molnár, P., & Sandvik, E. (2016). Google searches and stock returns. *International Review of Financial Analysis*, *45*, 150-156.

Cortis, K., Freitas, A., Daudert, T., Huerlimann, M., Zarrouk, M., Handschuh, S., & Davis, B. (2017, August). Semeval-2017 task 5: Fine-grained sentiment analysis on financial microblogs and news. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (pp. 519-535).

Larsen, V., & Thorsrud, L. A. (2017). Asset returns, news topics, and media effects.

Cortis, K., Freitas, A., Daudert, T., Huerlimann, M., Zarrouk, M., Handschuh, S., & Davis, B. (2017, August). Semeval-2017 task 5: Fine-grained sentiment analysis on financial microblogs and news. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (pp. 519-535).

Appendix C, R code for importing and tidying data

```
rm(list=ls())
# setwd("set your working directory")
library(gtrendsR)
library(tidyverse)

########## importing the stock data. Found through yahoo finance, form 30.12.2013(DMY), to 31.12.2018 (DMY), 262 weeks (except for bitcoin)
FB <- read_csv("FB.csv")
AAPL <- read_csv("AAPL.csv")
DJI <- read_csv("DJI.csv")
GOOGL <- read_csv("GOOGL.csv")
GSPC <- read_csv("GSPC.csv")
MSFT <- read_csv("MSFT.csv")
NHY <- read_csv("NHY.csv")
SALM <- read_csv("SALM.csv")
TEL <- read_csv("TEL.csv")
TSLA <- read_csv("TSLA.csv")
AMZN <- read_csv("AMZN.csv")
BTC <- read_csv("BTC.csv")

#getting weekly data for companies

## NOTE due to low(er) average volume of searches for Norwegian companies, we will use all kinds of google searches for the Norwegian companies (more specifically: gprop = "web")
salm_gt <- gtrends(keyword = 'salmar', time = "2013-12-29 2018-12-31", onlyInterest = T)
nhy_gt <- gtrends(keyword = 'norsk hydro', time = "2013-12-29 2018-12-31", onlyInterest = T)
tel_gt <- gtrends(keyword = 'telenor', time = "2013-12-29 2018-12-31", onlyInterest = T)

## Note for the American companies we will use only searches for news about the companies(i call them American even though they have HQ in other places, for totally not tax-related reasons)

msft_gt <- gtrends(keyword = 'microsoft', time = "2013-12-29 2018-12-31", gprop = "news", onlyInterest = T)
fb_gt <- gtrends(keyword = 'facebook', time = "2013-12-29 2018-12-31", gprop = "news", onlyInterest = T)
aapl_gt <- gtrends(keyword = 'apple inc', time = "2013-12-29 2018-12-31", gprop = "news", onlyInterest = T)
tsla_gt <- gtrends(keyword = 'tesla', time = "2013-12-29 2018-12-31", gprop = "news", onlyInterest = T)
amzn_gt <- gtrends(keyword = 'amazon', time = "2013-12-29 2018-12-31", gprop = "news", onlyInterest = T)
googl_gt <- gtrends(keyword = 'google', time = "2013-12-29 2018-12-31", gprop = "news", onlyInterest = T)


# and bitcoin. It's a shorter trend due to it having a shorter available price trend compared to the others.
btc_gt <- gtrends(keyword = 'bitcoin', time = "2014-09-13 2018-12-31", onlyInterest = T)


#now to gather all the data into a nice dataset, and make some new variables:
```

```
### 1) creating a new vector in each that are normalized in the same way as the google trends are normalized
### 2) Set google hits and stock info together. and add ticker names as a variable
### 3) making a column with the ticker name
AAPL <- AAPL %>%
  mutate(Normalized = `Adj Close`/max(`Adj Close`, na.rm = FALSE)*100)%>%
  mutate(hits = aapl_gt$interest_over_time$hits)%>%
  mutate(ticker = "AAPL")

DJI <- DJI %>%
  mutate(hits = NA)%>%
  mutate(Normalized = `Adj Close`/max(`Adj Close`, na.rm = FALSE)*100)%>%
  mutate(ticker = "DJI")

FB <- FB %>%
  mutate(Normalized = `Adj Close`/max(`Adj Close`, na.rm = FALSE)*100)%>%
  mutate(ticker = "FB")%>%
  mutate(hits = fb_gt$interest_over_time$hits)

GOOGL <- GOOGL %>%
  mutate(Normalized = `Adj Close`/max(`Adj Close`, na.rm = FALSE)*100)%>%
  mutate(ticker = "GOOGL")%>%
  mutate(hits = googl_gt$interest_over_time$hits)

GSPC <- GSPC %>%
  mutate(hits = NA)%>%
  mutate(Normalized = `Adj Close`/max(`Adj Close`, na.rm = FALSE)*100)%>%
  mutate(ticker = "GSPC")

MSFT <- MSFT %>%
  mutate(Normalized = `Adj Close`/max(`Adj Close`, na.rm = FALSE)*100)%>%
  mutate(ticker = "MSFT")%>%
  mutate(hits = msft_gt$interest_over_time$hits)

NHY <- NHY %>%
  mutate(Normalized = `Adj Close`/max(`Adj Close`, na.rm = FALSE)*100)%>%
  mutate(ticker = "NHY")%>%
  mutate(hits = nhy_gt$interest_over_time$hits)

SALM <- SALM %>%
  mutate(Normalized = `Adj Close`/max(`Adj Close`, na.rm = FALSE)*100)%>%
  mutate(ticker = "SALM")%>%
  mutate(hits = salm_gt$interest_over_time$hits)

TEL <- TEL %>%
  mutate(Normalized = `Adj Close`/max(`Adj Close`, na.rm = FALSE)*100)%>%
  mutate(ticker = "TEL")%>%
```

```
  mutate(hits = tel_gt$interest_over_time$hits)

TSLA <- TSLA %>%
  mutate(Normalized = `Adj Close`/max(`Adj Close`, na.rm = FALSE)*100)%>%
  mutate(ticker = "TSLA")%>%
  mutate(hits = tsla_gt$interest_over_time$hits)

AMZN <- AMZN %>%
  mutate(Normalized = `Adj Close`/max(`Adj Close`, na.rm = FALSE)*100)%>%
  mutate(ticker = "AMZN")%>%
  mutate(hits = amzn_gt$interest_over_time$hits)

BTC <- BTC %>%
  mutate(Normalized = `Adj Close`/max(`Adj Close`, na.rm = FALSE)*100)%>%
  mutate(ticker = "BTC")%>%
  mutate(hits = btc_gt$interest_over_time$hits)

#combine all the different datasets into a nice Tidy tibble

combinedd <- rbind(AMZN,TSLA,DJI,FB,GOOGL,GSPC,MSFT,NHY,SALM,TEL,AAPL,BTC)

#remove all the unused vectors and datasets.

lsss <- ls()

lsss <- lsss[lsss != "combinedd"]

rm(list = lsss)
```

Appendix D, R code for summary statistics

```
#Making summary statistics for the searches and normalized prices for the 9 companies


hits_sum <- combinedd %>% #summary statistics for the "hits" variable
  filter(! ticker %in% c("BTC","GSPC","DJI"))%>%
  select(Date, hits, ticker)%>%
  group_by(ticker) %>% #want to group by the companies in the summaries
  summarize(mean_hits = mean(hits),sd_hits = sd(hits),min_hits = min(hits),max_hits =      max(hits))%>% #selecting
what should be summarized
  as.matrix() #need to be converted to a matrix...


write.table(hits_sum, file = "hits_sum.txt", sep = ",", quote = FALSE, row.names = F) #... in order to write it to a
text file
#this .txt file can be copied into a word/latex file and converted to a table quite easily


norm_sum <- combinedd %>% #summary statistics for the "Normalized", but done the same way as for hits
  filter(! ticker %in% c("BTC","GSPC","DJI"))%>%
  select(Date, Normalized,ticker)%>%
  group_by(ticker) %>%
  summarize(mean_Normalized = mean(Normalized),sd_Normalized = sd(Normalized),
            min_Normalized = min(Normalized),max_Normalized = max(Normalized))%>%
  as.matrix()


write.table(norm_sum, file = "norms_sum.txt", sep = ",", quote = FALSE, row.names = F)
```

Appendix E, R code for all graphs

```
#need to run the tidy_data.R (initializing the data set) before running this code
library(tidyverse)
# adding some variables to the dataset:
combidata <- combinedd%>%
  filter(! ticker %in% c("DJI","GSPC"))%>%
  group_by(ticker)%>%
  mutate(delta =(Normalized - lag(Normalized)))%>% #makes a variable that shows the change in normalized price
  mutate(abs_delta = abs(delta))%>%
  mutate(lhits = lag(hits))%>%
  mutate(norm_volume = (Volume/max(Volume))*100)%>%
  drop_na()

#LETS MAKE SOME GRAPHS

btc_trends <- combidata%>% #bitcoin trend
  filter(ticker == "BTC")%>%
  ggplot()+
  geom_line(aes(x = Date, y = hits, color = "Normalized hits"))+
  geom_line(aes(x = Date, y = Normalized, color = "Normalized price"))+
  ylab("value")

facetin_trends <- combidata%>% #trends for the six companies
  group_by(ticker)%>%
  filter(ticker != "BTC")%>%
  ggplot(aes(x =  Date),theme())+
  geom_line(aes(y = hits, color = "Normalized hits"))+
  geom_line(aes(y = Normalized, color = "Normalized stock price"))+
  facet_wrap(~ticker)+
  ylab("Value")

facetin_normalized <- combidata%>% #Displaying normalized stock price and hits in a scatter plot. and a linear
regression line
  group_by(ticker)%>%
  filter(ticker != "BTC")%>%
  ggplot(aes(x = hits, y= Normalized))+
  geom_point()+
  geom_smooth(method = 'lm')+
  facet_wrap(~ticker)+
  ylab("Normalized price")

total_normalized <- combidata%>% #displaying all obeservations of normalized and hits in the same graph with a
regression line
  filter(ticker != "BTC")%>%
  ggplot(aes(x = hits, y= Normalized))+
  geom_point()+
```

```
  geom_smooth(method = 'lm')+
  ylab("normalized price")

facetin_volume_hits <- combidata%>% #volume and hits in the same scatter plot. with a regression line, for each
company
  group_by(ticker)%>%
  filter(ticker != "BTC")%>%
  ggplot(aes(x = hits, y= norm_volume))+
  geom_point()+
  facet_wrap(~ticker)+
  geom_smooth(method = 'lm')+
  ylab("normalized volume")

total_volume <- combidata%>% # all observations of volume and hits in the same graph with a regression line
  filter(ticker != "BTC")%>%
  ggplot(aes(x = hits, y= norm_volume))+
  geom_point()+
  geom_smooth(method = 'lm')+
  ylab("Normalized volume")
```

Appendix F, R code for OLS and LASSO

```
library(glmnet)
library(broom)
library(data.table)

set.seed(911222)
companies <- c("AAPL","AMZN","FB", "GOOGL", "MSFT", "NHY", "SALM", "TEL", "TSLA")

combinedd <- combinedd %>%
  filter(! ticker %in% c("DJI","GSPC"))%>%
  group_by(ticker)%>%
  mutate(delta =(Normalized - lag(Normalized)))%>%
  mutate(abs_delta = abs(delta))%>%
  mutate(norm_volume = (Volume/max(Volume))*100)

## CREATE LAGGED VARIABLES
combinedd <- combinedd %>%
  group_by(ticker) %>%
  mutate(lNormalized = lag(Normalized), llNormalized = lag(Normalized, 2), lVolume = lag(norm_volume),
         llVolume = lag(norm_volume, 2), lhits = lag(hits), llhits = lag(hits, 2)) %>%
  filter(llhits!="NA") # removes rows with missing values from lagging.


## CREATE TRAINING AND TEST SAMPLE
train <- sample(1:260, 208) #all companies have the same number of rows = 262 (260 after removing missing values)
trainset <- combinedd %>%
  group_by(ticker) %>%
  slice(train)

testset <- combinedd %>%
  group_by(ticker) %>%
  slice(-train)

## CREATE OLS REGRESSIONS ON PRICE FOR ALL COMPANIES
ols_price <- tibble() #initialise a reg summary tibble
ols_price_pred <- tibble()#initialise a pred summary tibble
mse_table <- tibble(ticker = companies, mse_price_ols = 0, mse_price_lasso = 0, mse_vol_ols = 0,
                    mse_vol_lasso = 0) #initialise a summary tibble of mse to compare.



#LOOP OVER ALL THE COMPANIES
for (i in companies) {
  regset <- trainset %>% #create set filtered on company for doing regressions
    filter(ticker == i)
```

```
  reg_price <- lm(Normalized ~ lNormalized + llNormalized + norm_volume + hits + lhits + llhits, data=regset)
#regression

  tidyp <- tidy(reg_price) #tidies regression output into a tibble
  tidyp <- tidyp %>%
    mutate(ticker = i)

  ols_price <- rbind(ols_price, tidyp) #add results to the aggregate table

  predset <- testset %>% #filter testset on company to do predictions
    filter(ticker == i)

  pred_price <- predict(reg_price, newdata=predset) #predicted price

  actual_price <- testset %>% #actual price from the testing sample
    filter(ticker == i) %>%
    select("Normalized") %>%
    as.matrix()
  actual_price <- actual_price[,2]
  class(actual_price) <- "numeric" #make sure matrix can be used for mathematical operations

  mse_price <- mean((pred_price-actual_price)^2) #calculates mean squared error
  confusionp <- tibble(actual_price, pred_price, mse_price, Date = predset$Date) #combine
  confusionp <- confusionp %>%
    mutate(ticker = i)

  ols_price_pred <- rbind(ols_price_pred, confusionp) #add predictions to aggregate table

  mse_table <- mse_table %>% #add mse to aggregate table
    mutate(mse_price_ols = replace(mse_price_ols, ticker==i, mse_price))
}

ols_price_pred[,1:3] <- round(ols_price_pred[,1:3], 3)
ols_price[,2:5] <- round(ols_price[,2:5],3) #round decimals
ols_price <- ols_price %>%
  mutate(sign = ifelse(abs(statistic) > 1.96, 1, 0)) #create dummy to display if variable is significant at 5% level.



##CREATE REGRESSIONS FOR ALL COMPANIES ON VOLUME
ols_vol <- tibble() #initialise a reg summary tibble
ols_vol_pred <- tibble()#initialise a pred summary tibble

#LOOP OVER ALL THE COMPANIES
for (i in companies) {
```

```
  regset <- trainset %>% #create set filtered on company for doing regressions
    filter(ticker == i)
  reg_vol <- lm(norm_volume ~ lVolume + Normalized + hits + lhits + llhits, data=regset) #regression

  tidyv <- tidy(reg_vol) #tidies regression output into a tibble
  tidyv <- tidyv %>%
    mutate(ticker = i)

  ols_vol <- rbind(ols_vol, tidyv) #add results to the aggregate table

  predset <- testset %>% #filter testset on company to do predictions
    filter(ticker == i)
  pred_vol <- predict(reg_vol, newdata=predset) #predicted price

  actual_vol <- testset %>% #actual price from the testing sample
    filter(ticker == i) %>%
    select("norm_volume") %>%
    as.matrix()
  actual_vol <- actual_vol[,2]
  class(actual_vol) <- "numeric" #make sure matrix can be used for mathematical operations

  mse_vol <- mean((pred_vol-actual_vol)^2) #calculates mean squared error
  confusionv <- tibble(actual_vol, pred_vol, mse_vol, Date = predset$Date) #combine
  confusionv <- confusionv %>%
    mutate(ticker = i)

  ols_vol_pred <- rbind(ols_vol_pred, confusionv) #add predictions to aggregate table
  mse_table <- mse_table %>% #add mse to aggregate table
    mutate(mse_vol_ols = replace(mse_vol_ols, ticker==i, mse_vol))
}

ols_vol_pred[,1:3] <- round(ols_vol_pred[,1:3], 3)
ols_vol[,2:5] <- round(ols_vol[,2:5],3) #round decimals
ols_vol <- ols_vol %>%
  mutate(sign = ifelse(abs(statistic) > 1.96, 1, 0)) #create dummy to display if variable is significant at 5% level.




##LASSO PRICE
lasso_price <- tibble() #initialise tibbles similar to the ones in OLS
lasso_price_coef <- tibble() #initialise tibbles similar to the ones in OLS

##LOOP THROUGH COMPANIES
for (i in companies) {
  #Create matrices to be used in glmnet
```

```
  yprice <- trainset %>%
    filter(ticker == i) %>%
    select("Normalized") %>%
    as.matrix()
  yprice <- yprice[,-1]
  class(yprice) <- "numeric"
  yprice <- as.matrix(yprice)
  xprice <- trainset %>%
    filter(ticker == i) %>%
    select("lNormalized", "llNormalized","norm_volume", "lVolume", "hits", "lhits", "llhits") %>%
    as.matrix()
  xprice <- xprice[,-1]
  class(xprice) <- "numeric"
  #Create matrices from test sample
  yprice_out <- testset %>%
    filter(ticker == i) %>%
    select("Normalized") %>%
    as.matrix()
  yprice_out <- yprice_out[,-1]
  class(yprice_out) <- "numeric"
  yprice_out <- as.matrix(yprice_out)
  xprice_out <- testset %>%
    filter(ticker == i) %>%
    select("lNormalized", "llNormalized", "norm_volume", "lVolume", "hits", "lhits", "llhits") %>%
    as.matrix()
  xprice_out <- xprice_out[,-1]
  class(xprice_out) <- "numeric"
  #LASSO
  predset <- testset %>%
    filter(ticker == i)

  cv_price <- cv.glmnet(y=yprice, x=xprice)
  cv_price_pred <- predict(cv_price, newx=xprice_out, s='lambda.min') #How does lasso predict test data
  mse_cv_price <- mean((cv_price_pred-yprice_out)^2) #mse to compare with ols
  confusionp <- tibble(actual=yprice_out, pred = cv_price_pred, mse = mse_cv_price, Date=predset$Date, ticker=i)
#combine results

  lasso_price <- rbind(lasso_price, confusionp) #add results to aggregate lasso table

  coeffs <- cbind(as.matrix(coef(cv_price)), ticker=i) #create table with coefficients from lasso, which ones are
kept?
  lasso_price_coef <- rbind(lasso_price_coef, coeffs) #add coefficients to aggregate table
  mse_table <- mse_table %>% #add mse to aggregate table
    mutate(mse_price_lasso = replace(mse_price_lasso, ticker==i, mse_cv_price))
}
```

```
setDT(lasso_price_coef, keep.rownames = "term") #from package data.table. Converts the rownames into values in the
table
lasso_price_coef <- lasso_price_coef %>% #tidy
  rename(coefficient = 2) %>%
  mutate(sign = ifelse(coefficient != 0, 1, 0))%>%
  mutate(coefficient = as.numeric(as.character(coefficient)))
lasso_price_coef[,2] <- round(lasso_price_coef[,2],3)
lasso_price[,1:3] <- round(lasso_price[,1:3],3)

##LASSO VOLUME
lasso_vol <- tibble() #initialise tibbles similar to the ones in OLS
lasso_vol_coef <- tibble() #initialise tibbles similar to the ones in OLS

##LOOP THROUGH COMPANIES
for (i in companies) {
  #Create matrices to be used in glmnet
  yvol <- trainset %>%
    filter(ticker == i) %>%
    select("norm_volume") %>%
    as.matrix()
  yvol <- yvol[,-1]
  class(yvol) <- "numeric"
  yvol <- as.matrix(yvol)
  xvol <- trainset %>%
    filter(ticker == i) %>%
    select("lVolume", "llVolume","Normalized", "lNormalized", "hits", "lhits", "llhits") %>%
    as.matrix()
  xvol <- xvol[,-1]
  class(xvol) <- "numeric"
  #Create matrices from test sample
  yvol_out <- testset %>%
    filter(ticker == i) %>%
    select("norm_volume") %>%
    as.matrix()
  yvol_out <- yvol_out[,-1]
  class(yvol_out) <- "numeric"
  yvol_out <- as.matrix(yvol_out)
  xvol_out <- testset %>%
    filter(ticker == i) %>%
    select("lVolume", "llVolume", "Normalized", "lNormalized", "hits", "lhits", "llhits") %>%
    as.matrix()
  xvol_out <- xvol_out[,-1]
  class(xvol_out) <- "numeric"
  #LASSO
  predset <- testset %>%
    filter(ticker == i)
```

```
  cv_vol <- cv.glmnet(y=yvol, x=xvol)
  cv_vol_pred <- predict(cv_vol, newx=xvol_out, s='lambda.min') #How does lasso predict test data
  mse_cv_vol <- mean((cv_vol_pred-yvol_out)^2) #mse to compare with ols
  confusionv <- tibble(actual=yvol_out, pred = cv_vol_pred, mse = mse_cv_vol, Date=predset$Date, ticker=i)#combine
results

  lasso_vol <- rbind(lasso_vol, confusionv) #add results to aggregate lasso table

  coeffs <- cbind(as.matrix(coef(cv_vol)), ticker=i) #create table with coefficients from lasso, which ones are kept?
  lasso_vol_coef <- rbind(lasso_vol_coef, coeffs)#add coefficients to aggregate table
  mse_table <- mse_table %>% #add mse to aggregate table
    mutate(mse_vol_lasso = replace(mse_vol_lasso, ticker==i, mse_cv_vol))
}

setDT(lasso_vol_coef, keep.rownames = "term") #from package data.table. Converts the rownames into values in the
table
lasso_vol_coef <- lasso_vol_coef %>% #tidy
  rename(coefficient = 2) %>%
  mutate(sign = ifelse(coefficient != 0, 1, 0)) %>%
  mutate(coefficient = as.numeric(as.character(coefficient)))
lasso_vol_coef[,2] <- round(lasso_vol_coef[,2],3)
lasso_vol[,1:3] <- round(lasso_vol[,1:3],3)

mse_table[,-1] <- round(mse_table[,-1],3)

##Create tables to use in paper
ols_price <- ols_price %>%
  as.matrix()
write.table(ols_price, file = "ols_price.txt", sep = ",", quote = FALSE, row.names = F)

ols_vol <- ols_vol %>%
  as.matrix()
write.table(ols_vol, file = "ols_vol.txt", sep = ",", quote = FALSE, row.names = F)

lasso_price_coef <- lasso_price_coef %>%
  as.matrix()
write.table(lasso_price_coef, file = "lasso_price_coef.txt", sep = ",", quote = FALSE, row.names = F)

lasso_vol_coef <- lasso_vol_coef %>%
  as.matrix()
write.table(lasso_vol_coef, file = "lasso_vol_coef.txt", sep = ",", quote = FALSE, row.names = F)

mse_table <- mse_table %>%
  as.matrix()
write.table(mse_table, file = "mse_table.txt", sep = ",", quote = FALSE, row.names = F)
```

Appendix G, R code for classification tree and random forest

```
#need to run the tidy_data.R (initializing the data set) before running this code
library(rpart)
library(rpart.plot)
library(randomForest)

set.seed(69745354)
companies <- c("AAPL","AMZN","FB", "GOOGL", "MSFT", "NHY", "SALM", "TEL", "TSLA")

combinedd_tree <- combinedd %>% #Making some new varaibles and storin the it as a new set
  filter(ticker %in% companies)%>%
  group_by(ticker)%>%
  mutate(delta =(Normalized - lag(Normalized)))%>%
  mutate(lhits = lag(hits))%>%
  mutate(abs_delta = abs(delta))%>%
  mutate(norm_volume = (Volume/max(Volume))*100)%>%
  drop_na()

#### making an outcom variable, it will be a boolean variable
zeros <- rep(0,235)
ones <- rep(1,26)
tot <- c(zeros,ones)

combinedd_tree <- combinedd_tree%>%
  group_by(ticker)%>%
  arrange(ticker,abs(delta))%>%
  drop_na()%>%
  mutate(big_delta = as.factor(tot)) ##assigning the outcome variable to the most volotile weeks, for each company


## CREATE TRAINING AND TEST SAMPLE
train <- sample(1:261, 208) #all companies have the same number of rows = 261
trainset <- combinedd_tree %>%
  group_by(ticker) %>%
  slice(train)

testset <- combinedd_tree %>%
  group_by(ticker) %>%
  slice(-train)

######## The analysis ########
##### first by looking at all the companies at once. ####

rocky <- rpart(big_delta ~ hits, data = trainset) #training the classefiation tree
predz <- predict(rocky, testset, type = "class") # predicting the outcome variables in the test dataset
```

```
testset$total_single <- predz  #adding the results to the test data set.

rocky_random <- randomForest(big_delta ~ hits, data = trainset) #training the random forest

predz_random <- predict(rocky_random, testset) # predicting the outcome variables in the test dataset

testset$total_single_random <- predz_random  #adding the results to the test data set.


table(testset$big_delta,testset$total_single) #creating a confusion matrix with the results from the classefication
tree

table(testset$big_delta,testset$total_single_random) #creating a confusion matrix with the results from the random
forest


####### now the analysis using the variation of google searches for each company ########

companies <- c("AAPL","AMZN","FB", "GOOGL", "MSFT", "NHY", "SALM", "TEL", "TSLA") #making a list of all the company
names

###classification tree#######

guesses <- c() #creating an empty vector to store the results

for (i in companies) {
  training <- trainset%>% #finding the data of company i, that is in the training dataset
    filter(ticker == i)
  testing <- testset%>%
    filter(ticker == i)%>% #finding the data of company i, that is in the test dataset
    arrange(big_delta)

  balboa <- rpart(big_delta~ hits, training) #training the classefiation tree model for company i

  pred <- predict(balboa,testing, type = "class")  # predicting the outcome variables in the test dataset for company
i

  guesses <- append(guesses,pred) #adding the results for company i to a list of all the results

}
guesses <- guesses -1 #for some reason, the predict function resulted in 1's and 2's.. not 0's and 1's...

testset$guess <- guesses # adding the results to the test dataset


#########random forest ########
```

```
guesses_random <- c() #creating an empty vector to store the results
for (i in companies) {
  training <- trainset%>% #finding the data of company i, that is in the training dataset
    filter(ticker == i)

  testing <- testset%>% #finding the data of company i, that is in the test dataset
    filter(ticker == i)%>%
    arrange(big_delta)

  balboa_random <- randomForest(big_delta ~ hits, training) #training the random forest model for company i

  pred_random <- predict(balboa_random,testing, type = "class") # predicting the outcome variables in the test
dataset for company i

  guesses_random <- append(guesses_random,pred_random) #adding the results for company i to a list of all the results

}

guesses_random <- guesses_random -1 #for some reason, the predict function resulted in 1's and 2's.. not 0's and
1's...

testset$guess_random <- guesses_random # adding the results to the test dataset


#### making confusion matrices for both the classification tree, and the random forest

table(testset$big_delta,testset$guess)

table(testset$big_delta,testset$guess_random)

##########for bitcoin################################## it follows the same procedure as the analysis above
zeros_b <- rep(0,202)
ones_b <- rep(1,22)
tot_b <- c(zeros_b,ones_b,NA)

bit_coin <- combinedd%>%
  filter(ticker == "BTC")%>%
  mutate(delta = Normalized - lag(Normalized))%>%
  arrange(ticker, abs(delta))
bit_coin$big_delta <- as.factor(tot_b)

train_b <- sample(1:226, 198)
train_set_b <- bit_coin%>%
  slice(train_b)%>%
  drop_na()
```

```r
test_set_b <- bit_coin%>%
  slice(-train_b)%>%
  drop_na()

#single tree
single_tull <- rpart(big_delta ~ hits, data = train_set_b)

rpart.plot(single_tull)

rocky <- predict(single_tull,test_set_b, type = "class")

test_set_b$guess <- rocky


#random forest

tull <- randomForest(big_delta ~ hits, data = train_set_b)

rocky_random <- predict(tull,test_set_b, type = "class")

test_set_b$guess_random <- rocky_random

table(test_set_b$big_delta,test_set_b$guess)

table(test_set_b$big_delta,test_set_b$guess_random)
```

Appendix H, R code for text analysis

```
rm(list = ls())


library(tidytext)
library(tidyverse)
library(readxl)
library(qdap)
library(tm) #all the three text analysis packages are used, as they all have an exlusive function
library(SnowballC)
library(textmineR)


dic <- read_excel("inquirerbasic.xls")


dic$Positiv[dic$Positiv == "Positiv"] <- 1 #create a dummy variable


dic$Positiv[dic$Negativ == "Negativ"] <- 0


words <- na.omit(dic[1:3])


words <- words[,-2] #we only will use these two columns


words$Entry <- gsub("#1","",words$Entry)


words <- words[- grep("#", words$Entry),]


words$Entry <- tolower(words$Entry) #this is the final dictionary, only containing a dummy variable for a specific
word being positive


pos <- filter(words, words$Positiv == 1) #list of only the positive words, used to create a document-text matrix with
only these


DJI <- read_csv("DJIA.csv")


DJI <- DJI %>%
  arrange(Date) %>%
  select(`Adj Close`) %>%
  mutate(diff = 100 * (`Adj Close` / lag(`Adj Close`) - 1))


DJI <- DJI[,2]
DJI[1,1] = 0 #normalize the adjusted close and take everything else out of the matrix




news <- read_csv("CombinedNews.csv")
dates <- news[, 1]
```

```
news <- news[,-2]
news <- apply(news, 1, paste, collapse=" ") #we want all the news of a given day to be in one cell of the matrix
news <- tolower(news) #the following commands make the news data easier to use
news <- removeNumbers(news)
news <- wordStem(news)
news <- replace_abbreviation(news)
news <- replace_contraction(news)
news <- replace_symbol(news)
news <- removePunctuation(news)

keep <- paste0(words$Entry, collapse = "|") #we create a single cell value that contains all the words we would like to keep (only the positive words in the next row)

keeppos <- paste0(pos$Entry, collapse = "|")

Text <- str_extract_all(string = news,pattern = keep) #this code check all the words in a day if they are included in the pattern

TextPos <- str_extract_all(string = news,pattern = keeppos)

dtm <- CreateDtm(Text) #we create the document-term matrix (documents=days are rows, while words are columns)

dtmpos <- CreateDtm(TextPos)

all <- rowSums(as.matrix(dtm)) #number of words from the dictionary per day (every row is one day, every column is one word)

posonly <- rowSums(as.matrix(dtmpos)) #number of positives only

score <- posonly/all #finally we assign a score to every day (row), stating the ratio between the positivity and all the words from the dictionary

scatter.smooth(x = score, y=DJI$diff, main="SCORE ~ DJI", ylab = "DJI") #visibly no correlation

linearmodel <- lm(DJI$diff ~ score)
summary(linearmodel)
```