Moods on Twitter can predict stock price movements on Nasdaq

Greed and fear are the two driving forces on the stock market. Turns out, that positive and negative sentiments in social media messages, like tweets, can be used to predict the daily movements of stock prices.

Although news most certainly influence stock market prices, public mood states or sentiment may play an equally important role. We know from psychological research that emotions, in addition to information, play a significant role in human decision-making. Behavioral finance has provided further proof that financial decisions are significantly driven by emotion and mood. It is therefore reasonable to assume that the public mood and sentiment can drive stock market values as much as news. As far as tweets go, some interesting discoveries have already been made, e.g. [1], [2], [3]and [4].

I also wanted to investigate this and made some interesting discoveries! The entire analysis was written in Python.

Hypothesis

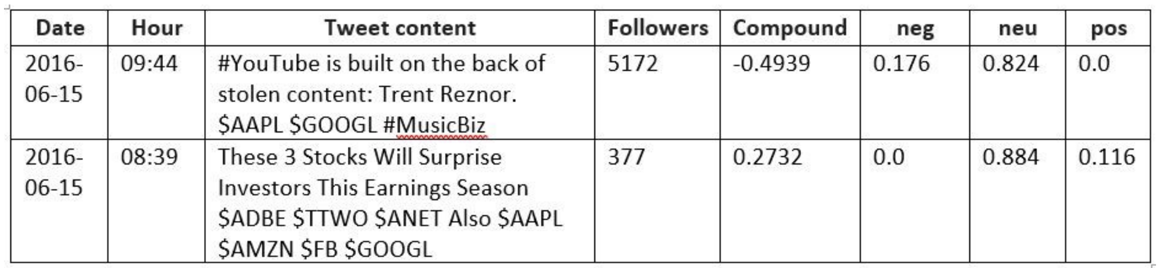
Tweets *today* with a positive or negative sentiment and containing one or several cashtags can affect the way a stock moves *tomorrow*. If negative sentiments dominate today, the stock price is expected to fall tomorrow, and rise, if the average sentiments are positive. The number of followers a Twitter account has is also a dominating factor. The more followers an account has, the more influential the tweets, and the more impacts their sentiments on the stock price.

What is a cashtag?

A feature in Twitter allowing users to click on stock symbols and see what the “Twitterverse” is saying about, say, $GOOG, $AAPL or $FB. The system works the same way as Twitter’s well-known #hashtags. Cashtags require a “$” followed by the ticker symbol.

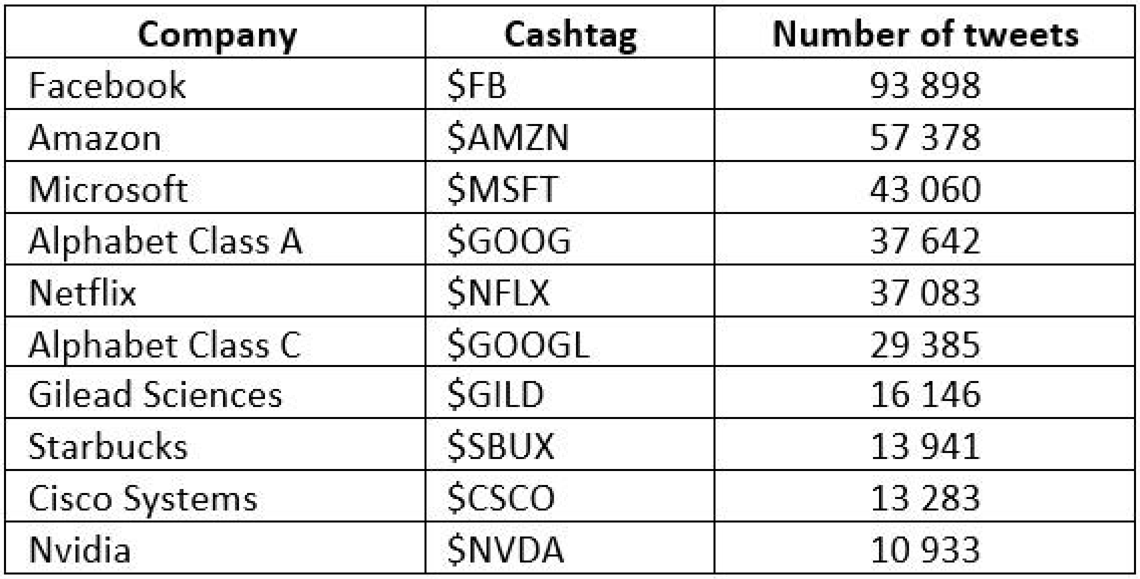
The dataset

A collection of about one million tweets gathered during 79 days, from 2016 March 28th to 2016 June 15th mentioning any NASDAQ 100 company cashtag. The data is provided by followthehashtag.com, a Twitter search analytics and business intelligence tool.
Here two examples of a negative and a positive tweets with cashtags for Apple, Google and few others.



Date	Hour	Tweet content	Followers	Compound	neg	neu	pos
2016-06-15	09:44	#YouTube is built on the back of stolen content: Trent Reznor. $AAPL $GOOGL #MusicBiz	5172	-0.4939	0.176	0.824	0.0
2016-06-15	08:39	These 3 Stocks Will Surprise Investors This Earnings Season $ADBE $TTWO $ANET Also $AAPL $AMZN $FB $GOOGL	377	0.2732	0.0	0.884	0.116

Examples of two tweets; first one with a negative sentiment, second tweet with a positive sentiment
Out of the 100 hundred original stocks in the data I had to drop 15 for various, data-specific reasons, such as inconsistencies in the dates, or simply because there were too few tweets about the cashtags, i.e. not even daily tweets. Among the excluded were, for example, Apple, Tesla and Yahoo.
The cashtags with the most tweets included in the final analysis were (Top 12):



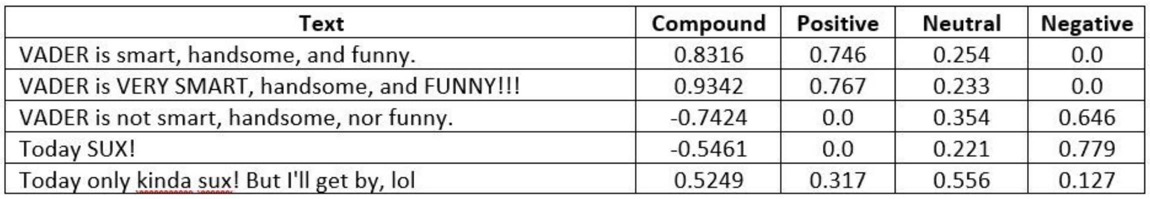
Company	Cashtag	Number of tweets
Facebook	$FB	93 898
Amazon	$AMZN	57 378
Microsoft	$MSFT	43 060
Alphabet Class A	$GOOG	37 642
Netflix	$NFLX	37 083
Alphabet Class C	$GOOGL	29 385
Gilead Sciences	$GILD	16 146
Starbucks	$SBUX	13 941
Cisco Systems	$CSCO	13 283
Nvidia	$NVDA	10 933

Cashtags with most tweets in the dataset

The average tweets per cashtag among all 100 stocks was 6 446 during the 79-day period, i.e. 81 tweets per day for each stock/cashtag.

Measuring the sentiment in tweets

To extract the sentiment of each tweet I used VADER, an off-the-shelf Python machine learning library for natural language processing, especially tailored for reading the sentiments of tweets. VADER is capable of giving more emphasis on capitals and also recognizes slang expressions, exclamation marks and the most common emojis. The sentiments are given scores from extremely negative (-1) to extremely positive (+1), neutral being zero. Examples



Text	Compound	Positive	Neutral	Negative
VADER is smart, handsome, and funny.	0.8316	0.746	0.254	0.0
VADER is VERY SMART, handsome, and FUNNY!!!	0.9342	0.767	0.233	0.0
VADER is not smart, handsome, nor funny.	-0.7424	0.0	0.354	0.646
Today SUX!	-0.5461	0.0	0.221	0.779
Today only kinda sux! But I'll get by, lol	0.5249	0.317	0.556	0.127

Examples of sentiments to texts given by VADER

Creating daily average values for the tweet data

After Combining each tweet with its sentiment, it was multiplied by the number of followers of the account. This way the sentiments of tweets by more "influential" accounts were given more emphasis in the final model. After this, the tweets, on average almost 6500 per cashtag, were narrowed down to 75 rows, consisting of a daily average for each sentiment, which would then be compared to the daily price changes of the related stock.

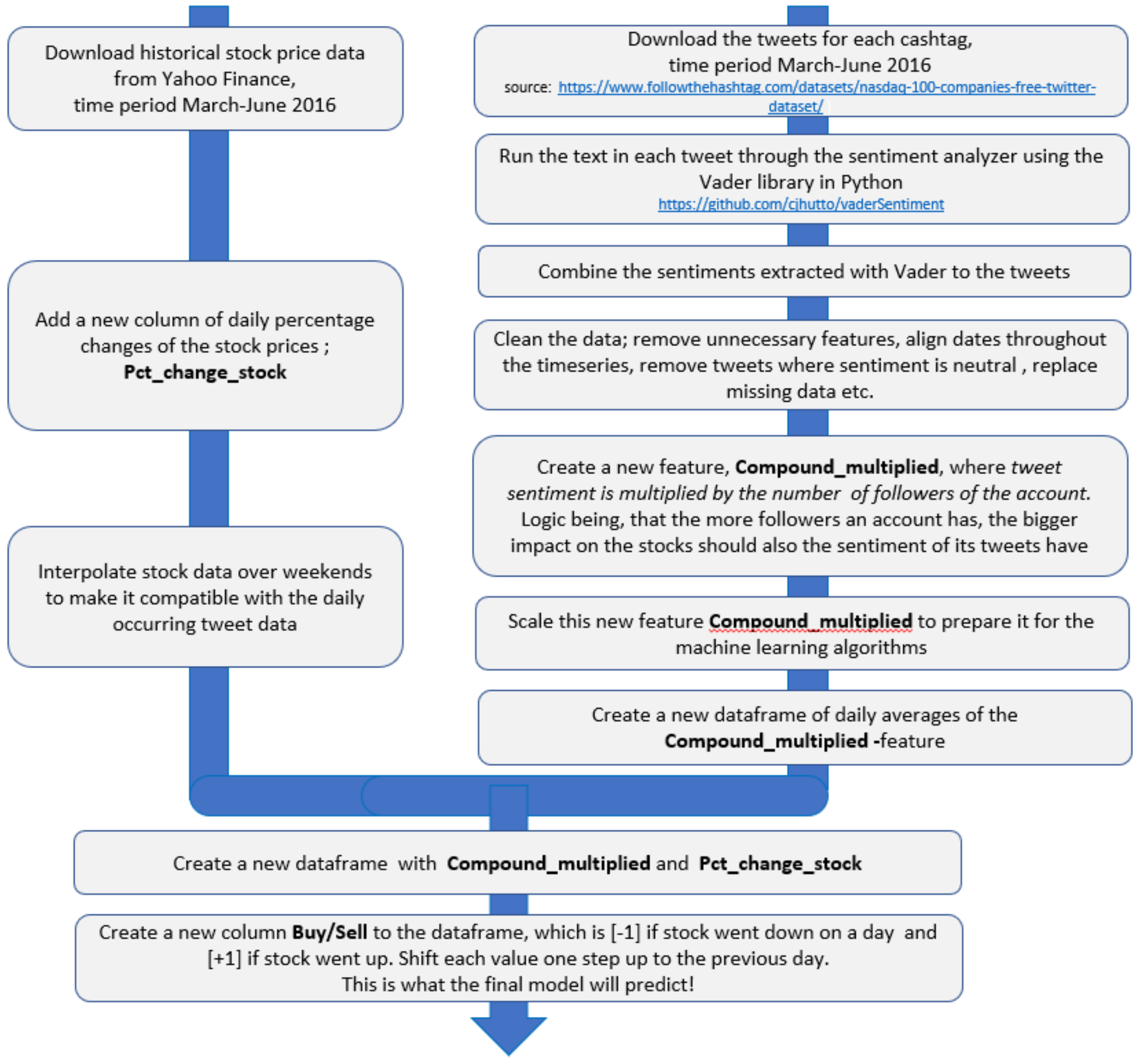
Gathering the stock data

Using Python's pandas-datareader library the daily data for the stocks was downloaded from Yahoo Finance. After adding a *daily percentage change* - column to the stock data and interpolating for the missing data for weekends, the two datasets, *tweets' sentiments* and *stocks' daily changes*, could now be combined.

A new dataframe, with two features, "Pct_change_stock" and "Compound_multiplied", and an added label data column "Buy/Sell", was now ready to be used in training.

Flow chart for the first part of the analysis

Stock data (left arrow flow) Twitter data (right arrow flow)


Download historical stock price data from Yahoo Finance, time period March-June 2016
Add a new column of daily percentage changes of the stock prices ; **Pct_change_stock**
Interpolate stock data over weekends to make it compatible with the daily occurring tweet data
Download the tweets for each cashtag, time period March-June 2016 source: https://www.followthehashtag.com/datasets/nasdaq-100-companies-free-twitter-dataset/
Run the text in each tweet through the sentiment analyzer using the Vader library in Python https://github.com/cjhutto/vaderSentiment
Combine the sentiments extracted with Vader to the tweets
Clean the data; remove unnecessary features, align dates throughout the timeseries, remove tweets where sentiment is neutral , replace missing data etc.
Create a new feature, **Compound_multiplied**, where *tweet sentiment is multiplied by the number of followers of the account.* Logic being, that the more followers an account has, the bigger impact on the stocks should also the sentiment of its tweets have
Scale this new feature **Compound_multiplied** to prepare it for the machine learning algorithms
Create a new dataframe of daily averages of the **Compound_multiplied** -feature
Create a new dataframe with **Compound_multiplied** and **Pct_change_stock**
Create a new column **Buy/Sell** to the dataframe, which is [-1] if stock went down on a day and [+1] if stock went up. Shift each value one step up to the previous day. This is what the final model will predict!

The steps made in the first part of the analysis; stock data vs. tweets

Training the machine learning classifiers with the data

Since this was a binary classification task, i.e. the outcome is either "Buy" or "Sell", I used 6 various algorithms suited for the cause.

· K-nearest Neighbors (KNN)

· Logistic Regression (LogReg)

· Support Vector Machine (SVM)

· Naive Bayes

· Decision Tree

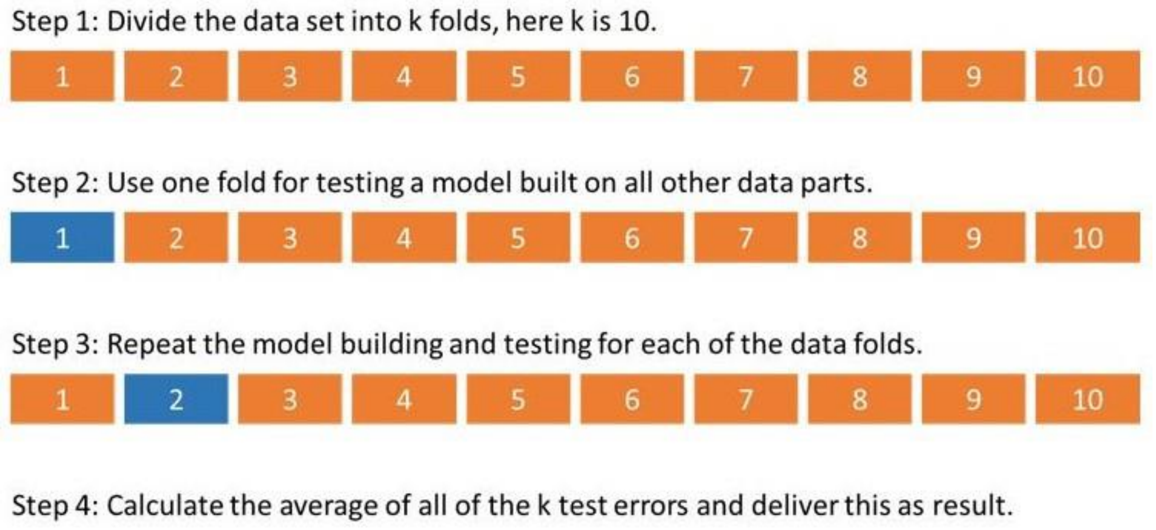
· Random Forest

Making the Training/Testing data split

Of the 74 days available, data from 59 days (80%) for each stock was used for training and 15 days (20%) for testing the accuracy of each algorithm.

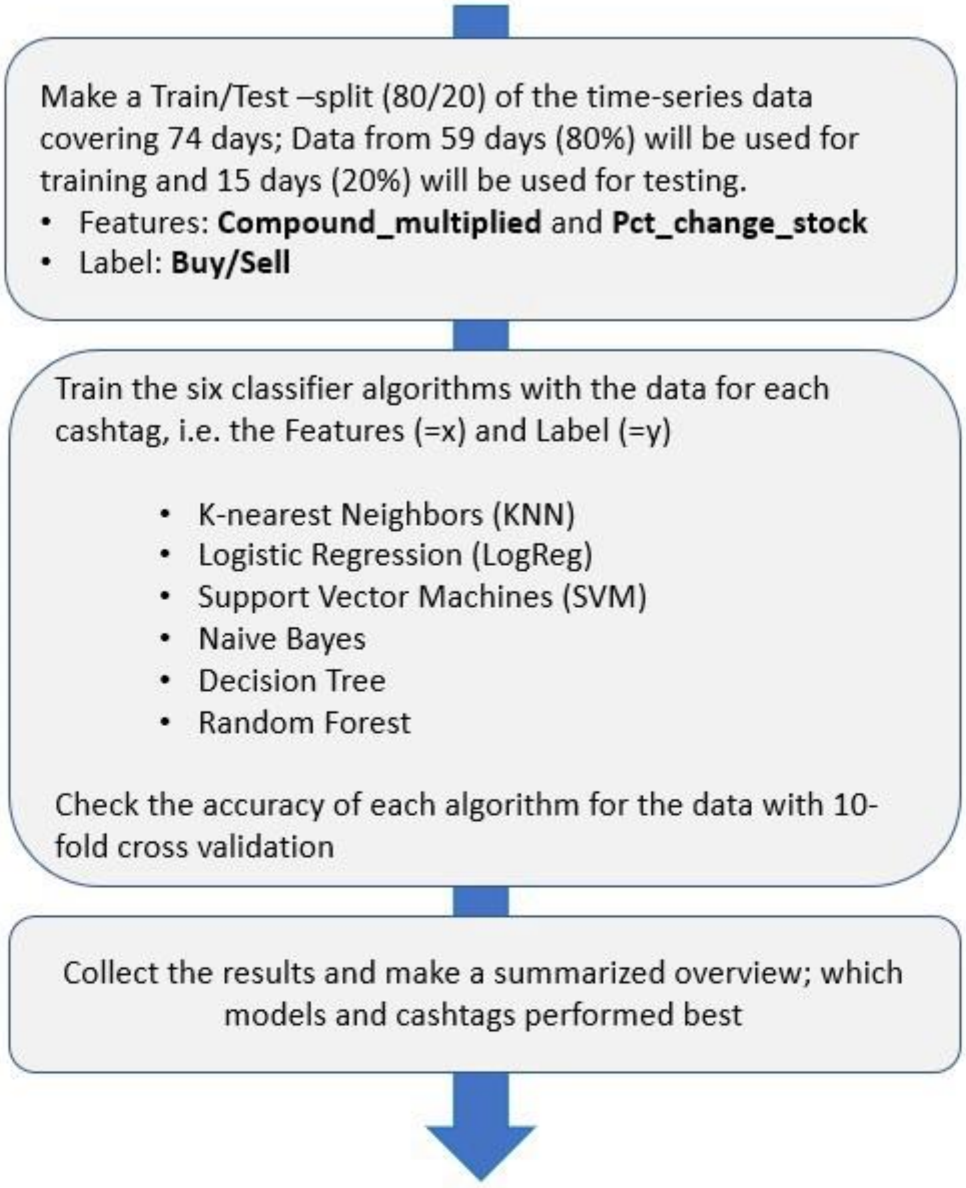
Cross-validation

Due to the limited amount of data, testing the models' with only 20% of data (15 days) against 80% of training data (59 days) may not be representative enough. To avoid the possibility of the Train/Test -split not being fully random, cross-validation of the data was performed, to get a more representative result of each algorithm's accuracy. Training data was split further into 10 subsets, where each was tested against the 9 others.


Step 1: Divide the data set into k folds, here k is 10.
1 2 3 4 5 6 7 8 9 10
Step 2: Use one fold for testing a model built on all other data parts.
1 2 3 4 5 6 7 8 9 10
Step 3: Repeat the model building and testing for each of the data folds.
1 2 3 4 5 6 7 8 9 10
Step 4: Calculate the average of all of the k test errors and deliver this as result.

Visual image of a 10-fold cross-validation

Flow chart for the second part of the analysis

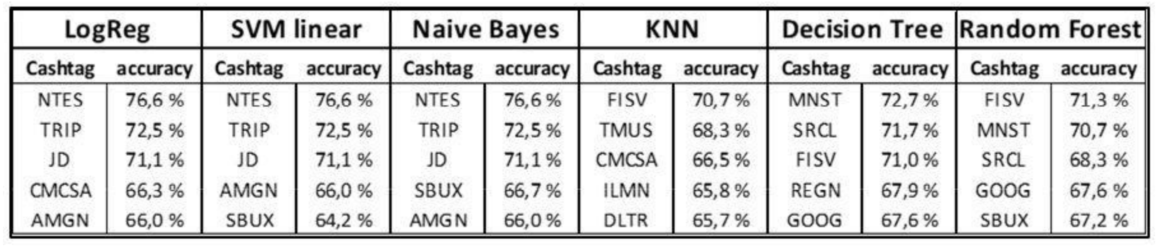

Make a Train/Test –split (80/20) of the time-series data covering 74 days; Data from 59 days (80%) will be used for training and 15 days (20%) will be used for testing.
• Features: **Compound_multiplied** and **Pct_change_stock**
• Label: **Buy/Sell**
Train the six classifier algorithms with the data for each cashtag, i.e. the Features (=x) and Label (=y)
• K-nearest Neighbors (KNN)
• Logistic Regression (LogReg)
• Support Vector Machines (SVM)
• Naive Bayes
• Decision Tree
• Random Forest
Check the accuracy of each algorithm for the data with 10-fold cross validation
Collect the results and make a summarized overview; which models and cashtags performed best

The steps made in the second part of the analysis; training the six classifiers

Results

The results, after running each of the 85 stocks through each of the 6 binary classifiers and a 10-fold cross-validation, were as follows. On average, the accuracy of each classifier was above 50%. This means, that there is predictive power in the tweets' sentiments, which beats at least tossing a coin. A coin toss would on average result in an accuracy of 50%, so an accuracy of above 50% is proof of the models' ability to achieve "extraordinary" gains to some extent. What’ s more, is that for many stocks the models’ accuracies/predictive powers were between 65–75%!

Accuracy of each classifier per cashtag — Top 5

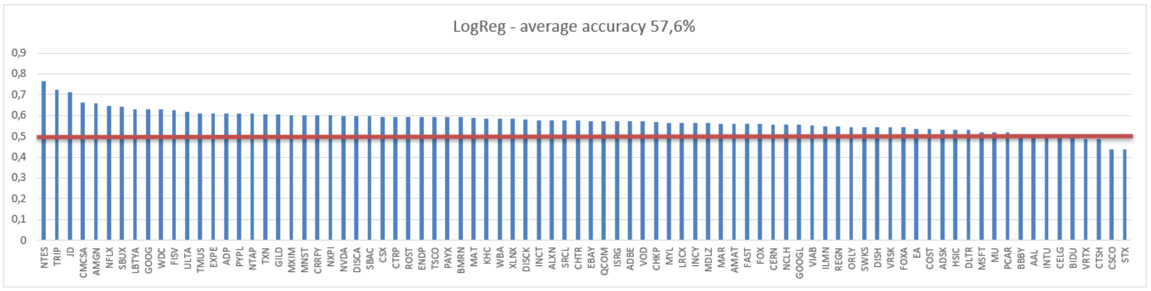


LogReg		SVM linear		Naive Bayes		KNN		Decision Tree		Random Forest	
Cashtag	accuracy	Cashtag	accuracy	Cashtag	accuracy	Cashtag	accuracy	Cashtag	accuracy	Cashtag	accuracy
NTES	76,6 %	NTES	76,6 %	NTES	76,6 %	FISV	70,7 %	MNST	72,7 %	FISV	71,3 %
TRIP	72,5 %	TRIP	72,5 %	TRIP	72,5 %	TMUS	68,3 %	SRCL	71,7 %	MNST	70,7 %
JD	71,1 %	JD	71,1 %	JD	71,1 %	CMCSA	66,5 %	FISV	71,0 %	SRCL	68,3 %
CMCSA	66,3 %	AMGN	66,0 %	SBUX	66,7 %	ILMN	65,8 %	REGN	67,9 %	GOOG	67,6 %
AMGN	66,0 %	SBUX	64,2 %	AMGN	66,0 %	DLTR	65,7 %	GOOG	67,6 %	SBUX	67,2 %

What share of the stocks’ next-day movements did the classifiers predict

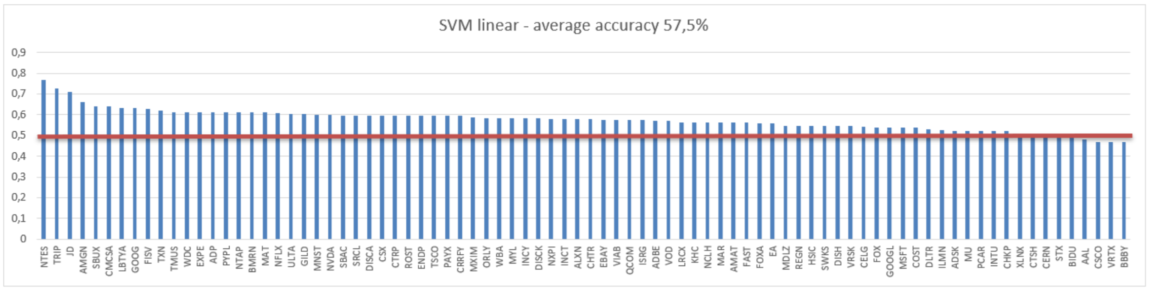
In the charts below the red line represents the 50% accuracy limit.

Logistic Regression: average accuracy 57.6% with 76 cashtags out of 85 above 50%



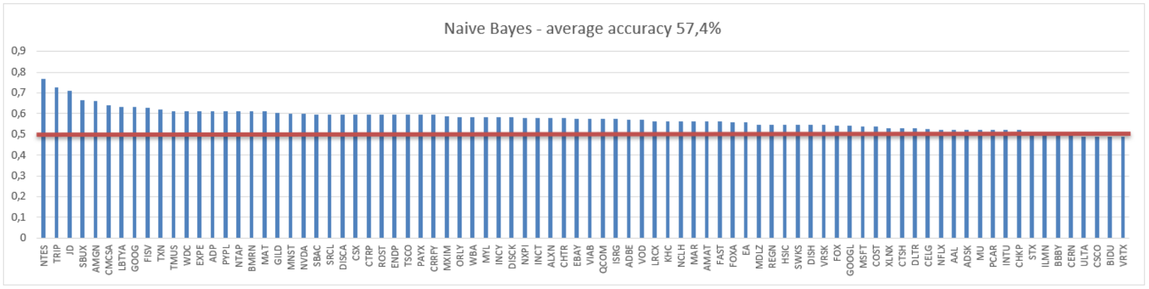
Accuracy per cashtag with LogReg

Support Vector Machine: average accuracy 57.5% with 77 cashtags out of 85 above 50%



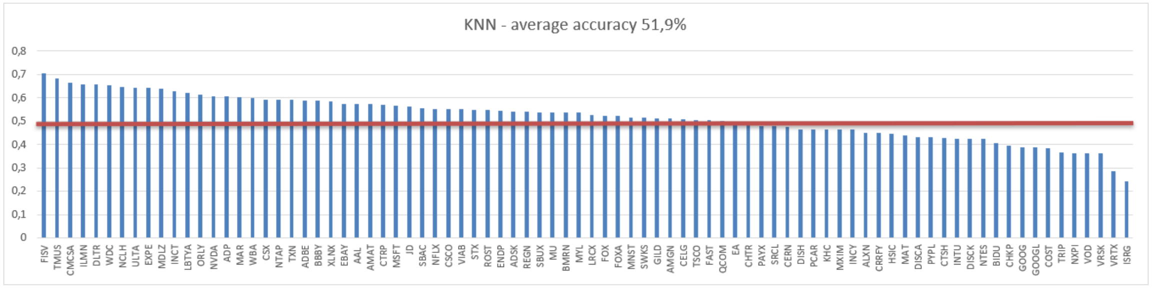
Accuracy per cashtag with SVM

Naïve Bayes: average accuracy 57.4% with 77 cashtags out of 85 above 50%



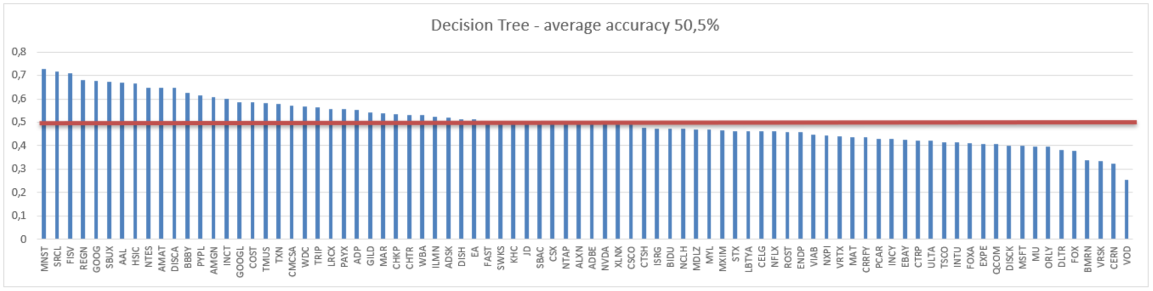
Accuracy per cashtag with Naive Bayes

K-Nearest Neighbors: average accuracy 51.9% with 53 cashtags out of 85 above 50%



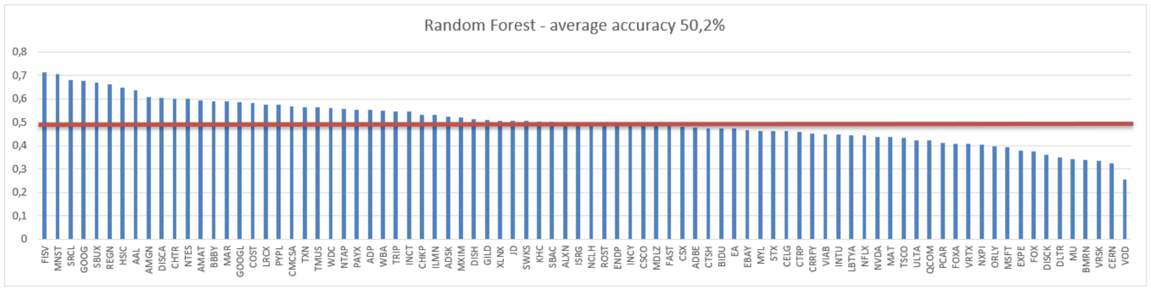
Accuracy per cashtag with KNN

Decision Tree: average accuracy 50.5% with 40 cashtags out of 85 above 50%



Accuracy per cashtag with Decision Tree

Random Forest: average accuracy 50.2% with 53 cashtags out of 85 above 50%



Accuracy per cashtag with Random Forest

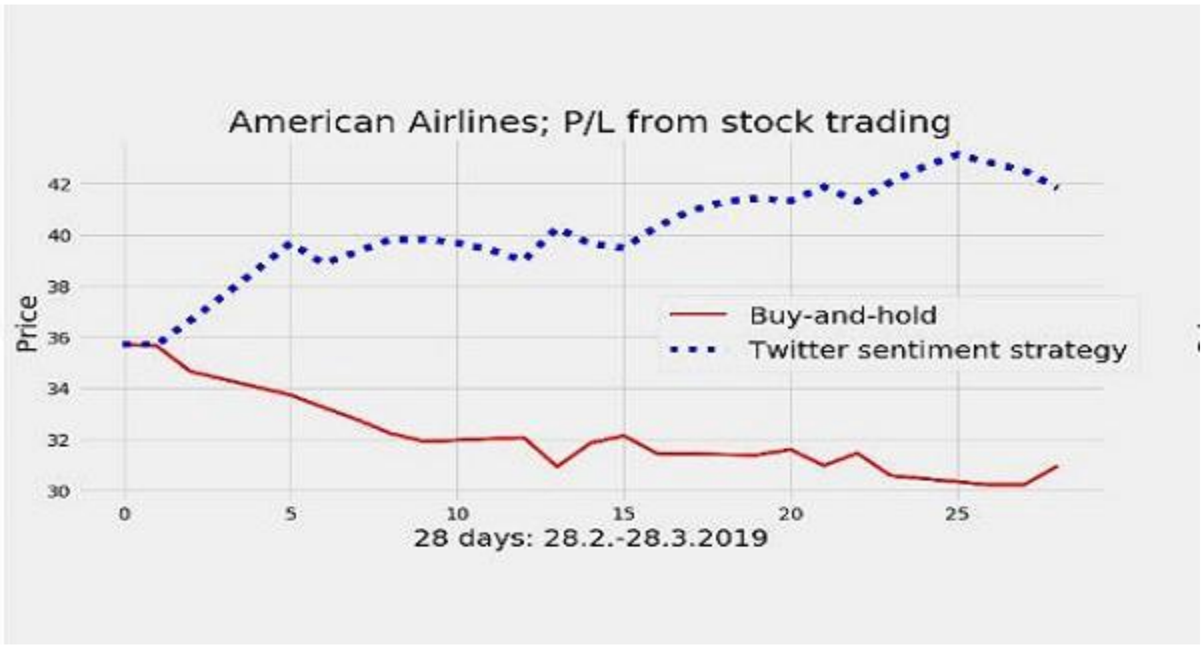
Summary

It appears there is clear predictive power in tweets, if you know how to squeeze the needed data out of them. Positive average sentiment the day before seems to indicate a rise in the stock’ s price the following day, while a negative average sentiment is followed by a drop. Out of the 6 classifier algorithms employed here, both Decision Trees and Random Forests appear to be inferior compared to the four others and may not prove that useful during the next steps (see below).

Code and samples of the data available on my Github.

Next steps

I intend to make trading simulations with some of the stocks showing most promising results from the models. The tweet data for the simulations covering stocks during March 2019 will be freshly scraped from Twitter using its development API.



Comparison of simulated P/L from two strategies in March 2019

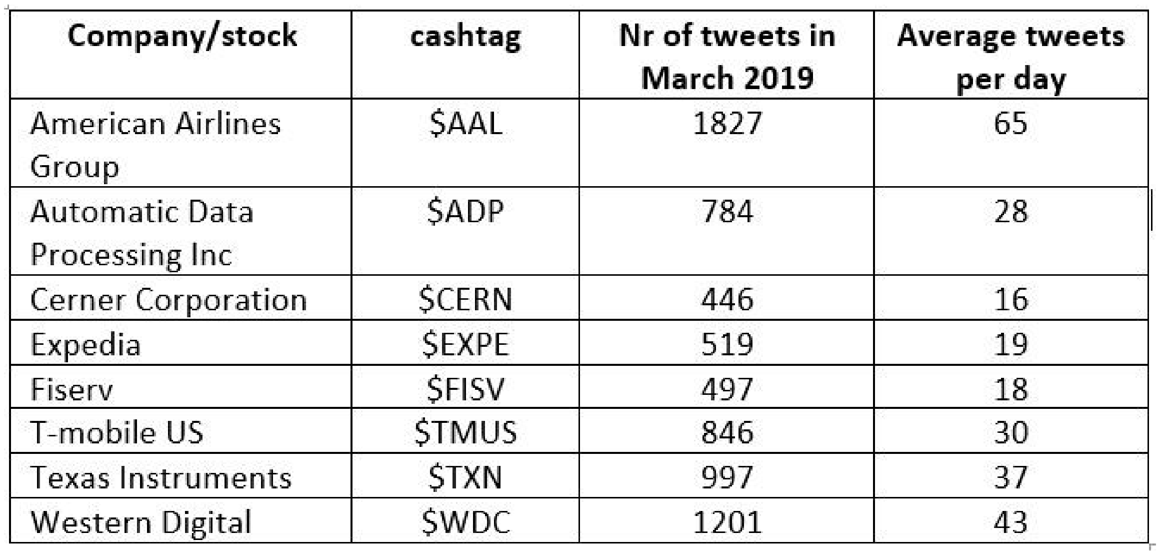
In the beginning part, I used Twitter data containing cashtags from 100 Nasdaq companies from 2016 and used it to train 6 different classifier models. The aim was to see if the sentiments in tweets could predict whether a stock is expected to rise or fall the following day. Astonishingly, on average there was clear predictive power in the sentiments, although large variations between different stocks were also present.

In the next step, the results of which are presented below, I compared the P/L from simple buy-and-hold strategies with the P/L that would' ve been achieved using the models. To my surprise during the 4-week period of simulated trades, most of the models turned out to produce profits far above what I had anticipated!

The entire analysis was written in Python.

Downloading the tweets

The 8 stocks on Nasdaq I chose for the simulations were some of the ones showing most promising results from the training of the models with the data from 2016. Total number of tweets used for the simulated trades in March was almost 7200, on average roughly 800 tweets per each stock.



Company/stock	cashtag	Nr of tweets in March 2019	Average tweets per day
American Airlines Group	$AAL	1827	65
Automatic Data Processing Inc	$ADP	784	28
Cerner Corporation	$CERN	446	16
Expedia	$EXPE	519	19
Fiserv	$FISV	497	18
T-mobile US	$TMUS	846	30
Texas Instruments	$TXN	997	37
Western Digital	$WDC	1201	43

Summary of the tweets used for the simulated trades

The tweet data was gathered by “scraping” Twitter using its Developer API. I simply downloaded all tweets during March 2016 containing the cashtags $AAL, $ADP, $CERN, $EXPE, $FISV, $TMUS, $TXN and $WDC.

Downloading and preparing the rest of the data

The procedures for the next steps were explained in detail before, so I won’t cover them here again. Here a short recap of the steps, though

1. Tweets are ran through the sentiment analyzer algorithm and each is given a sentiment; positive, neutral or negative.
2. Each tweet is multiplied by the number of followers of the account. This way the sentiments of tweets by more “influential” accounts are given more emphasis in the final model.
3. Tweet data is narrowed down to 28 rows, consisting of a daily average for each sentiment, which will be compared to the daily price changes of the related stock during the same period.
4. Stock data is downloaded and added with a ‘daily % change’ -column
5. Tweet and stock data are combined, and a label column is added, i.e. “Buy or sell”. This is what the models are trying to predict. In other words, should a stock be bought or sold tomorrow, based on the predicted values from the tweet sentiments today?

These datasets were then used by comparing a buy-and-hold strategy against six different models were each daily expected daily stock price movement was predicted using the models. Maybe the pictures below make the logic clearer!

Making the simulated trades — March 2019

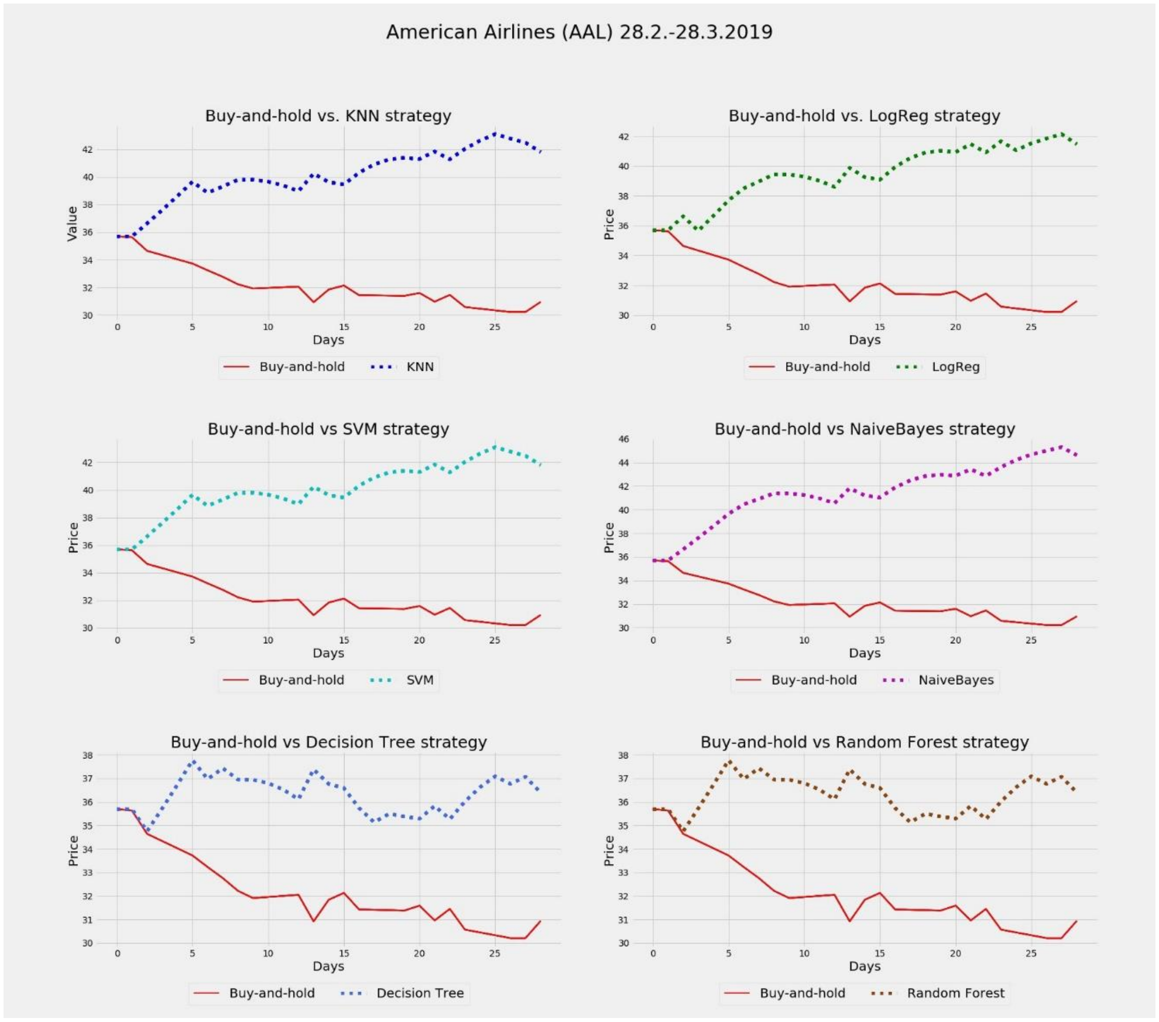
For each of the 8 stocks a buy-and-hold strategy was compared against the six other strategies based on the binary classifier algorithms;

· K-nearest Neighbors (KNN)
· Logistic Regression (LogReg)
· Support Vector Machine (SVM)
· Naive Bayes
· Decision Tree
· Random Forest

Each model having been trained with the original tweets from 2016 (again, details in the first article, Part 1) gave a daily suggestion — buy or sell tomorrow at open and sell or buy tomorrow at close. The daily trade simulations then followed blindly these suggestions.

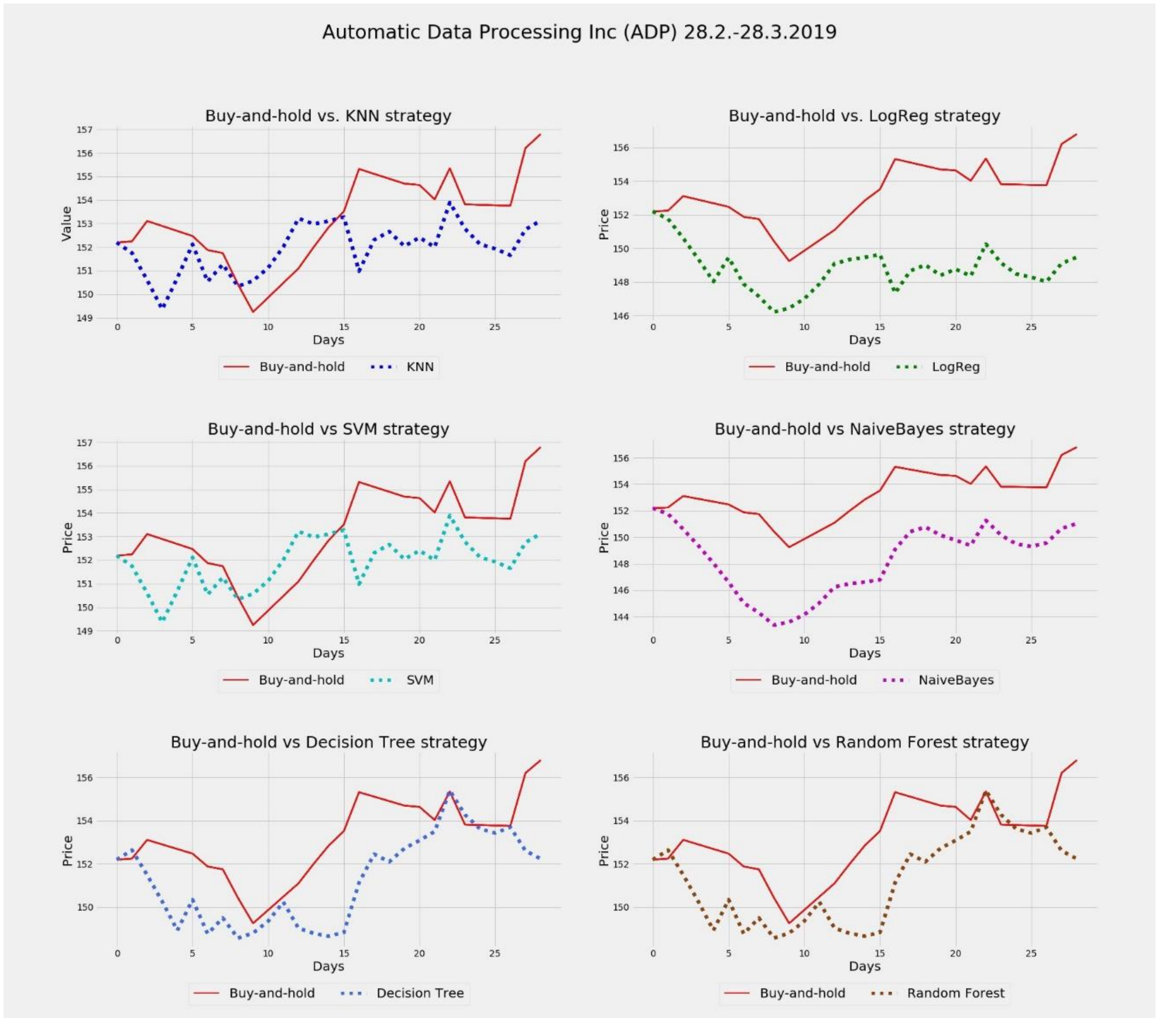
American Airlines — $AAL

All six models produced surprisingly good results beating the buy-and-hold -strategy with a clear margin



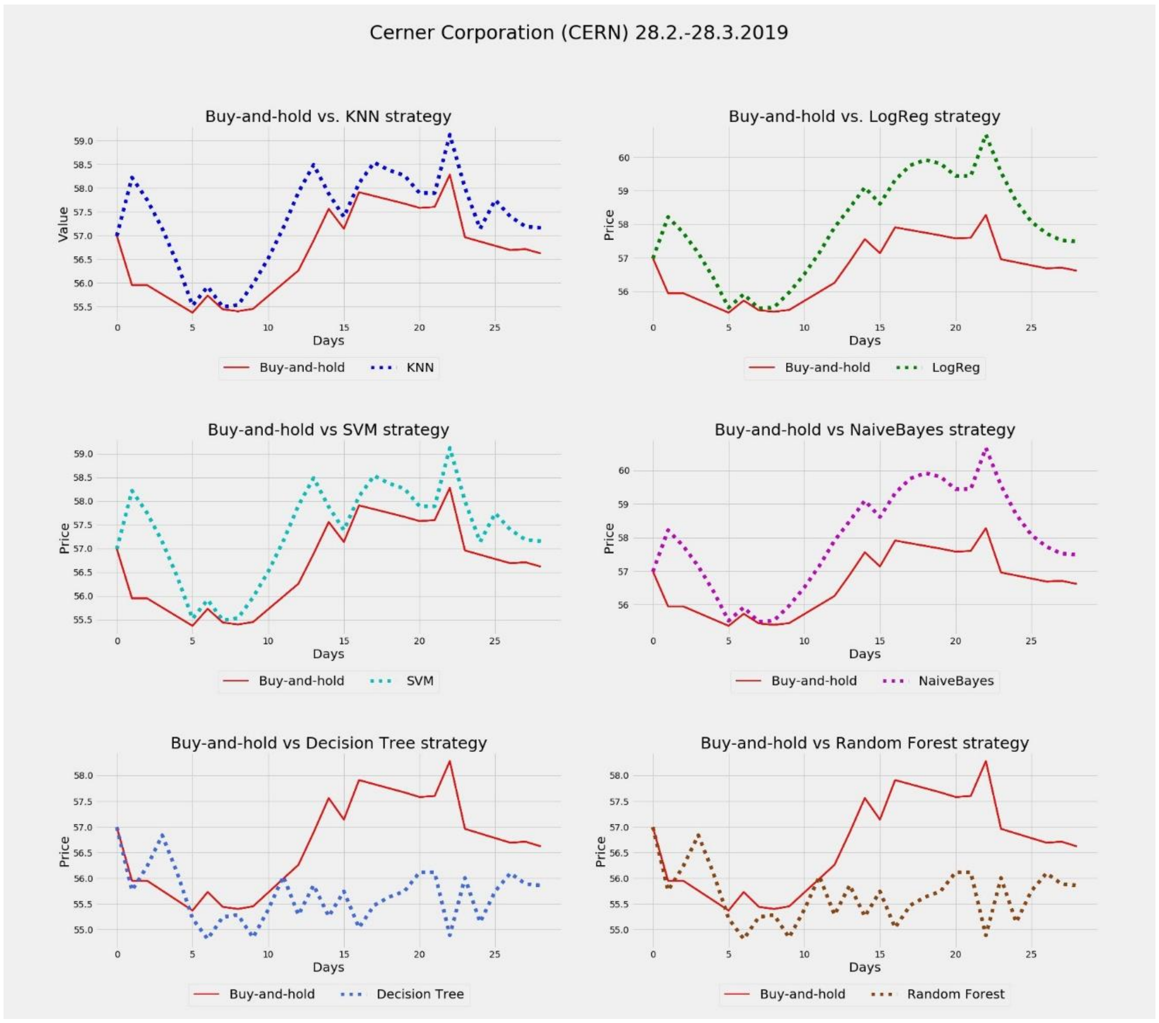
Automatic Data Processing — $ADP

In this case, all six models would’ ve made losses compared to a simple buy-and-hold.



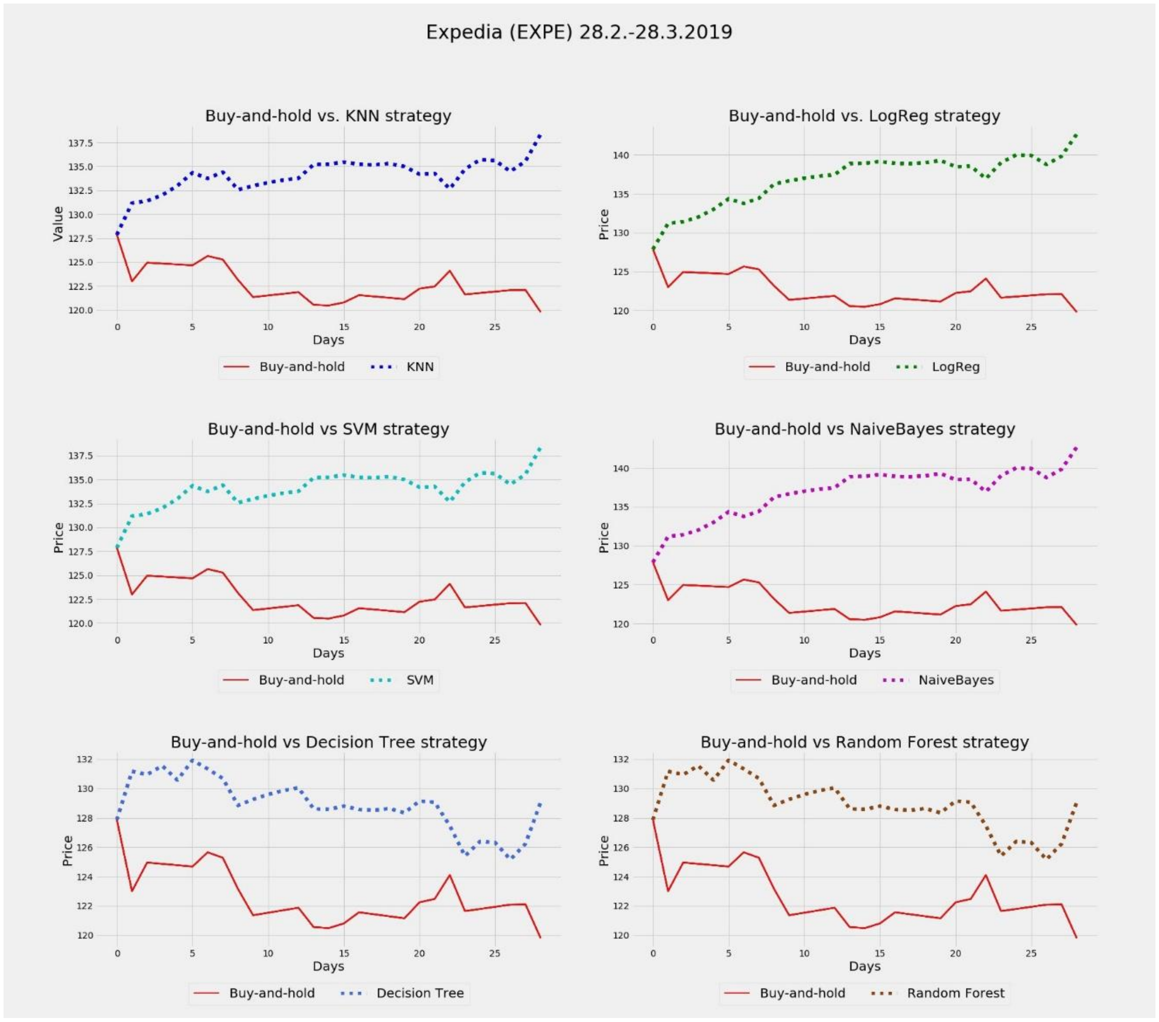
Cerner Corporation — $CERN

Four out of six models would’ ve beaten the buy-and-hold, the weakest models here being *Decision Tree* and *Random Forest*, as also indicated by tests presented in the first article.



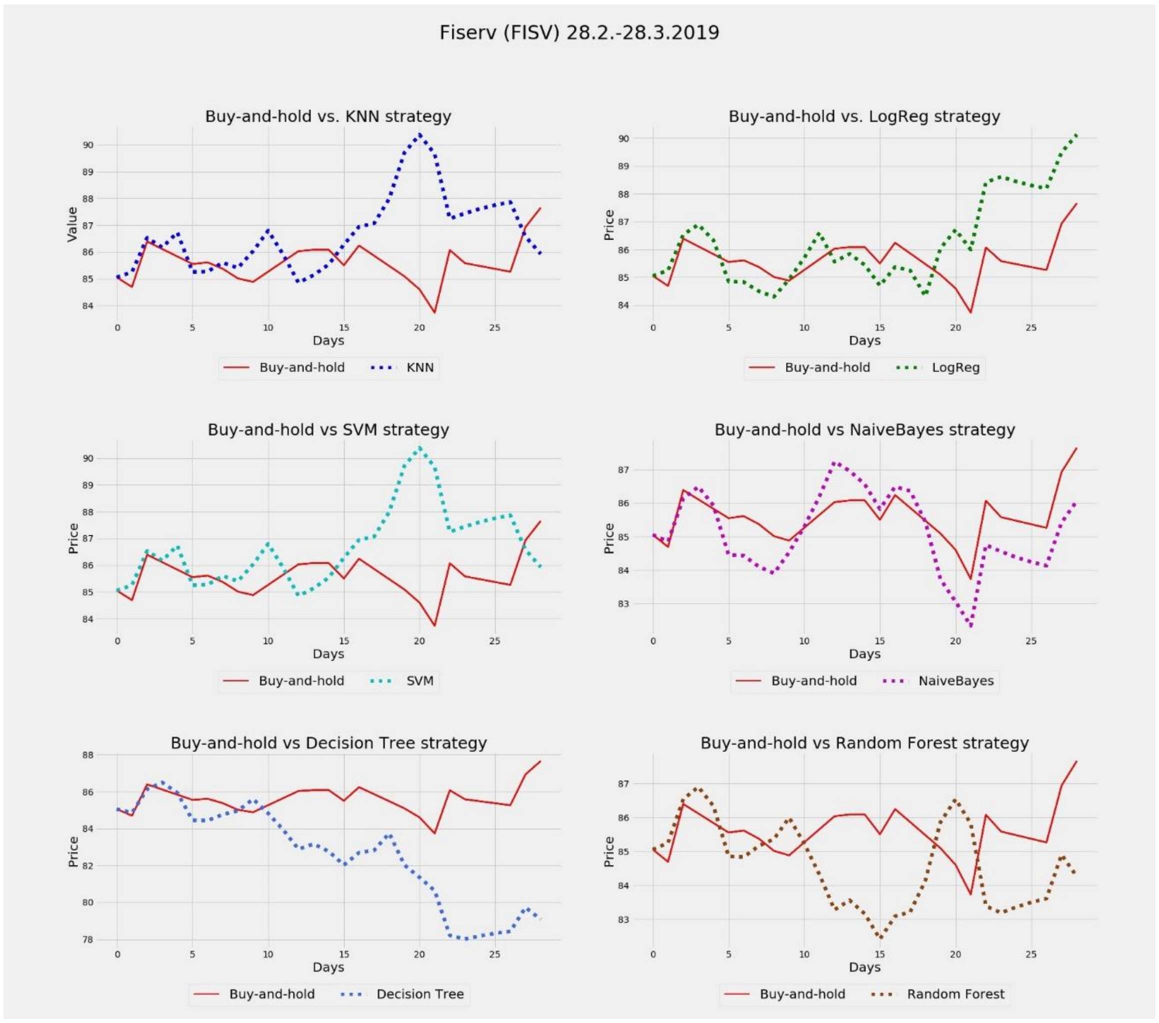
Expedia — $EXPE

Another strike for the six models, all clearly beating the buy-and-hold.



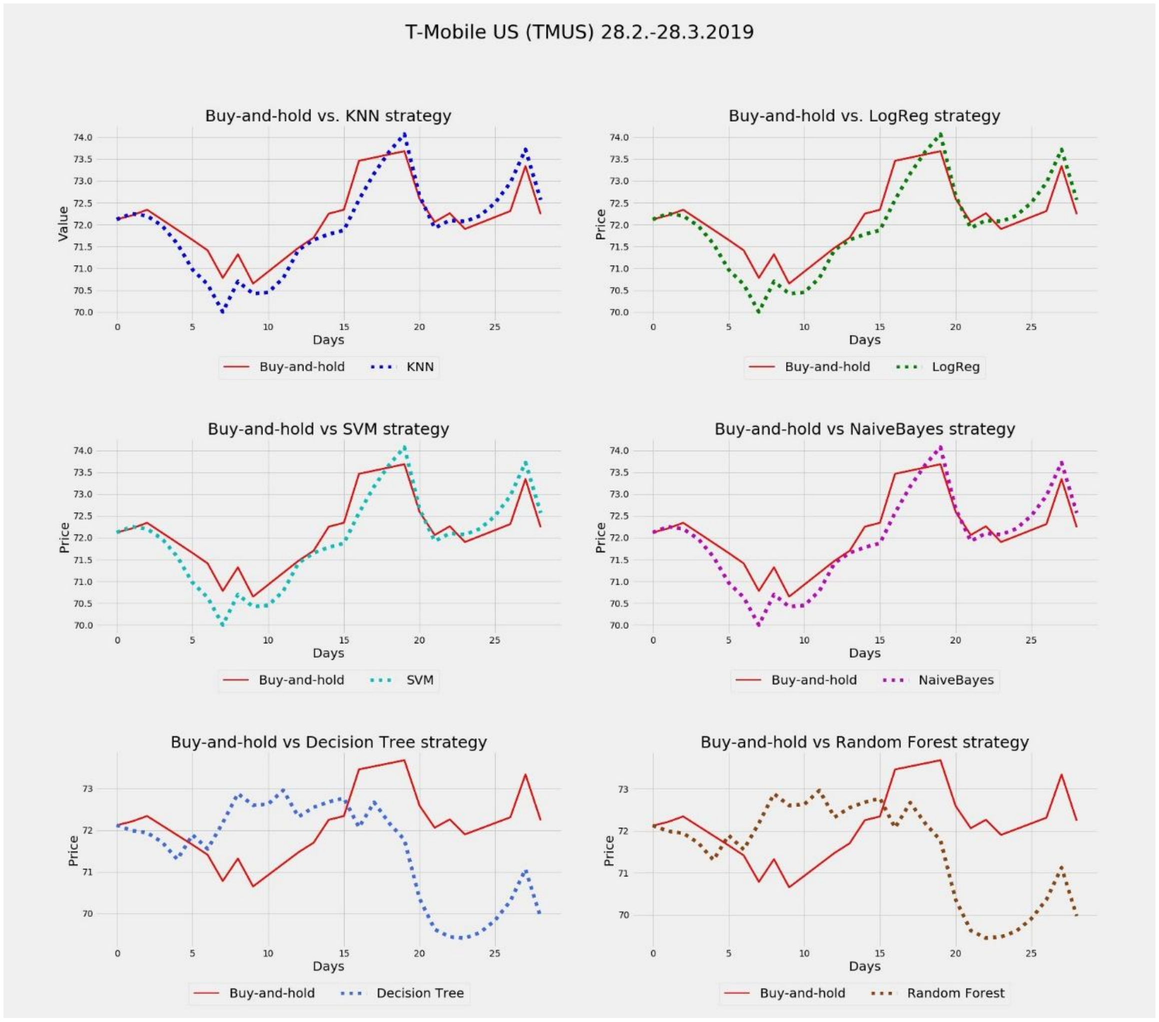
Fiserv — $FISV

More mixed results. Only one model, the one based on Logistic Regression, would’ ve beaten buy-and-hold. Interesting to note, that during a shorter 3-week period, 4 out of 6 models would’ ve been superior again.



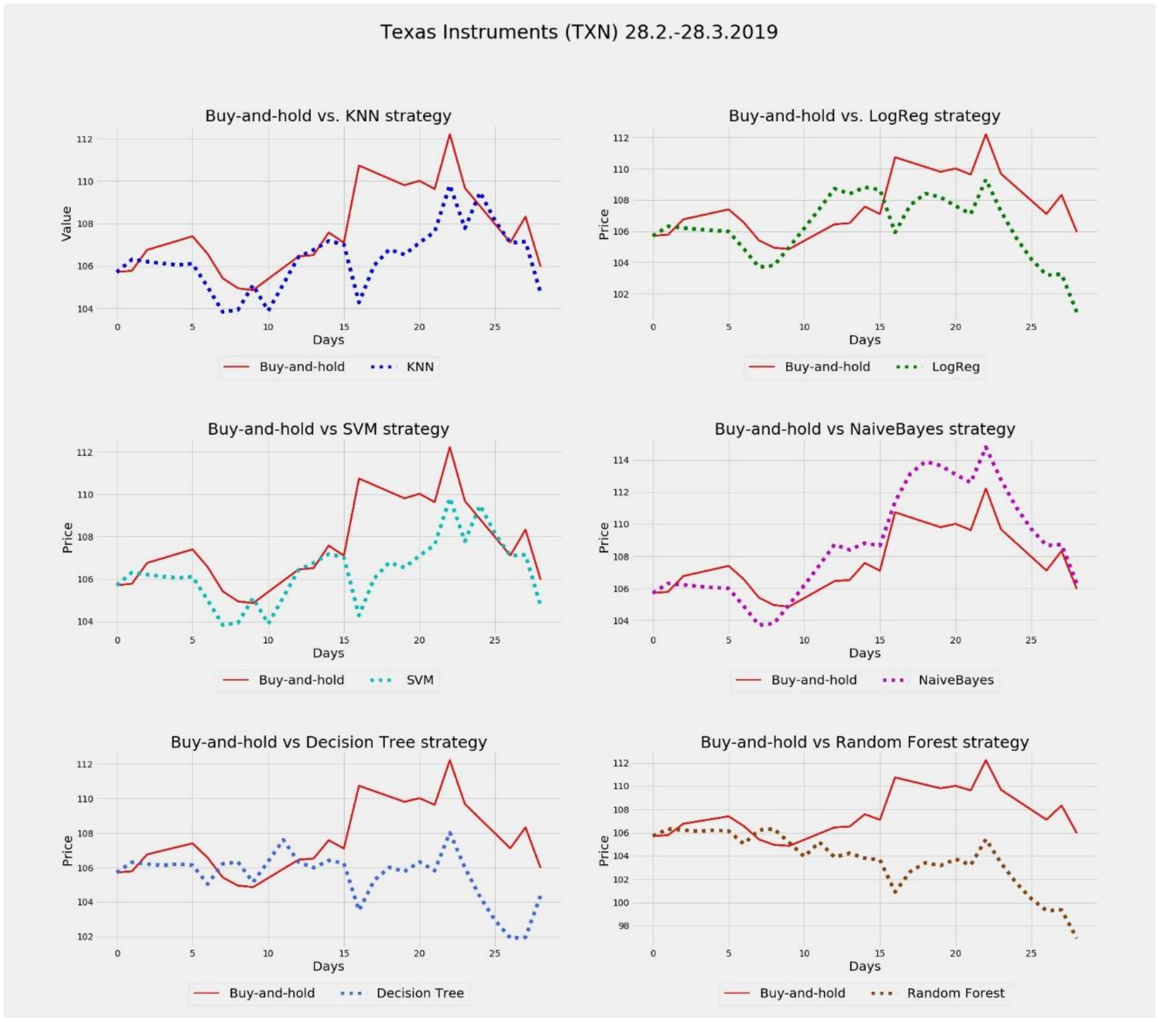
T-mobile US — $TMUS

Again, 4 out the 6 models would’ ve beaten buy-and-hold, but just barely!



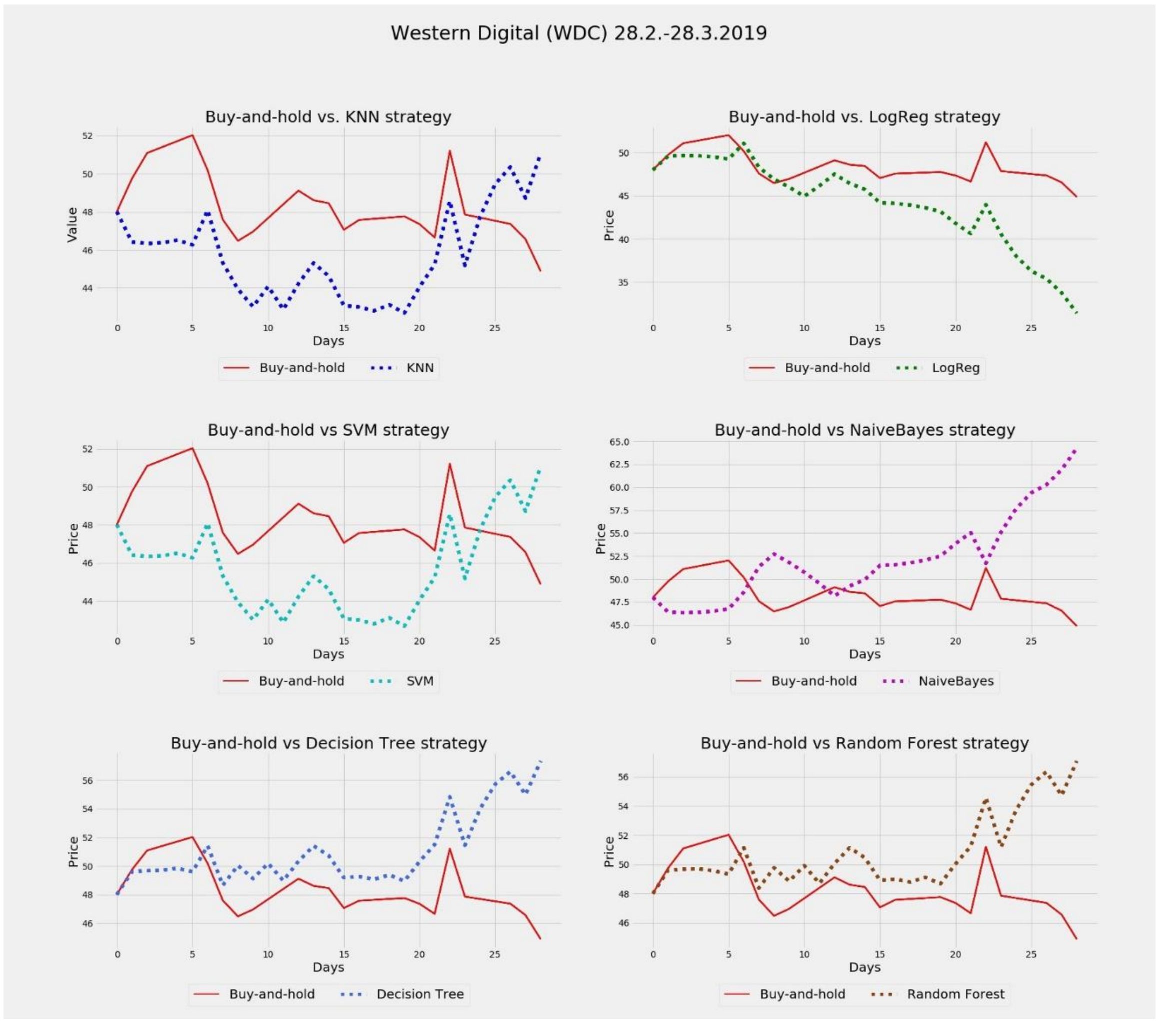
Texas Instruments — $TXN

Not so good for the models; buy-and-hold wins by 6 to 0.



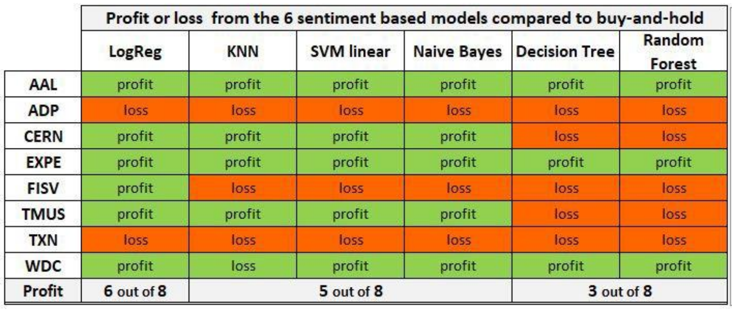
Western Digital

In the final simulation, models would’ ve beaten buy-and-hold once again, now 5 to 1.

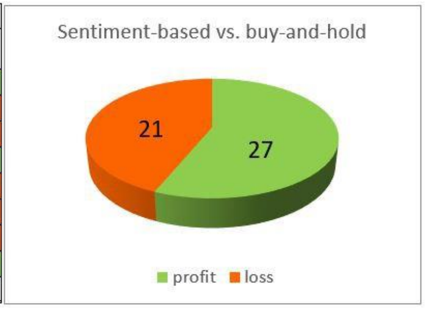


Summary

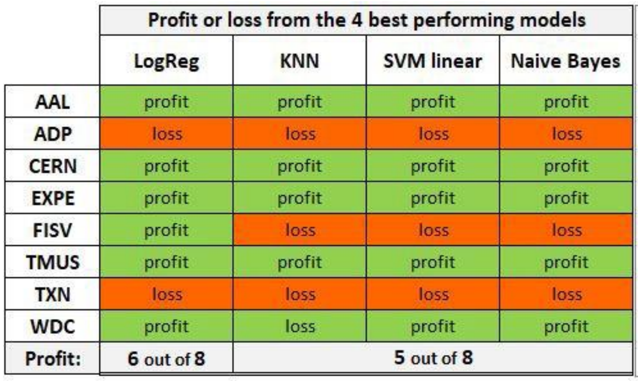
Overall, on average, the twitter sentiment-based strategies beat the buy-and-holds in closer to 60% of the simulation cases.



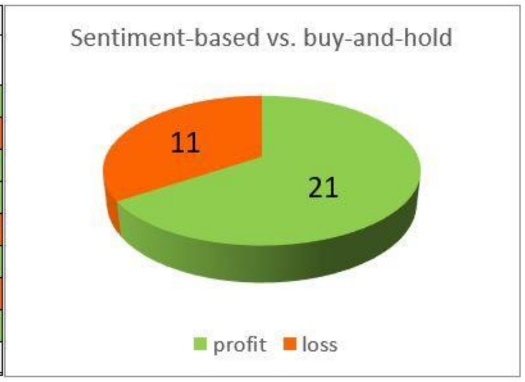
	Profit or loss from the 6 sentiment based models compared to buy-and-hold					
	LogReg	KNN	SVM linear	Naive Bayes	Decision Tree	Random Forest
AAL	profit	profit	profit	profit	profit	profit
ADP	loss	loss	loss	loss	loss	loss
CERN	profit	profit	profit	profit	loss	loss
EXPE	profit	profit	profit	profit	profit	profit
FISV	profit	loss	loss	loss	loss	loss
TMUS	profit	profit	profit	profit	loss	loss
TXN	loss	loss	loss	loss	loss	loss
WDC	profit	loss	profit	profit	profit	profit
Profit	6 out of 8	5 out of 8			3 out of 8	



Leaving out the two worst performing models, *Decision Trees* and *Random Forests*, the results were improved even further. Buy-and-holds were beaten in two thirds of the cases.



	Profit or loss from the 4 best performing models			
	LogReg	KNN	SVM linear	Naive Bayes
AAL	profit	profit	profit	profit
ADP	loss	loss	loss	loss
CERN	profit	profit	profit	profit
EXPE	profit	profit	profit	profit
FISV	profit	loss	loss	loss
TMUS	profit	profit	profit	profit
TXN	loss	loss	loss	loss
WDC	profit	loss	profit	profit
Profit:	6 out of 8	5 out of 8		



Should only the best performing model, *Logistic Regression*, have been followed, a profit would' ve been made in every 3 stocks out of 4!

Concluding remarks, ideas for further improving the models

Time frame too short?

· The models had only data from 75 days both for training and testing. Adding more data from a longer and perhaps even a more recent period could improve the results considerably, if there really is predictive power in the sentiments.

Adjustment for weekends

· To make the stock data, which covers only 5 days a week, fit with the twitter data, which covers 7 days a week, the *Adjusted Closing* prices needed to be interpolated for the weekends. Although considered feature engineering, the created stock prices for the weekends are artificial and might distort the results. Perhaps the tweets from Fridays to Sundays should somehow be grouped together when considering their impact on the stock movements on Mondays?

A neural network instead?

· One might consider using the results from tweet sentiments in combination with other techniques, such as a LSTM neural network for time-series analysis, making predictions always one day ahead.

Try another sentiment analyzer?

· Try extracting the tweet sentiments with some other off-the-shelf model than VADER, for example TextBlob. Or better yet, train your on sentiment classifier by building a neural network and train it with your own data, available here, for example; 1.6mio tweets each row marked as 0 = negative, 2 = neutral, 4 = positive.

Timing is everything?

· How much would timing have an impact on the final results? In the simulations the final P/L was dependent on the length of the period. In some cases, a profit turned into a loss and vice versa the longer the trading period.

Transaction costs

· No transaction costs were taken into consideration in the simulations. At least in the case where the final profit was rather slim, transaction costs could' ve turned the profit into a loss.

Are some stocks more sensitive to tweets?

· Could patterns be discovered in certain stocks within specific business areas? In this analysis, the greatest profits were made with two stocks in the travel industry, American Airlines and Expedia. Is this just a coincidence, or could the movements of stocks in certain businesses be more prone to tweet sentiments?

Code and samples of the data available on my Github.