Predicting next day's Bitcoin movement based on keyword-usage in user-specific Twitter messages

Johannes M. van Hulst
Radboud University, Nijmegen, Gelderland
mick.vanhulst@gmail.com

ABSTRACT

This paper introduces a method that can be used for trading Bitcoin. It is based on specific word usage within Twitter messages (also known as 'Tweets') by Twitter users. The goal of this research is to show if predictions regarding the Bitcoin price can be made by utilizing textual relationships of Tweets for specific users and if user-specific information carries importance when making such predictions. We instantiate matrices that are specific to a user's Tweets over a set amount of time. These matrices are used to train a Convolutional Neural Net (CNN), which predicts whether or not a profitable trade can be made. Keywords are found by using TF-IDF, which is a general technique that iss used for finding words of importance in a set of documents (i.e. Tweets in our case). The user-specific results are aggregated by day, resulting in a single prediction per day by utilizing a Logistic Regression (LR) and arithmetic mean based method. Success is measured by calculating the Precision and Recall scores. These scores are related to the, assumed to be goal of the trader, which is to increase its overall capital. The results show that it is preliminary to say whether or not the method is of significance to a trader, but the work aims to serve as a baseline for future research.

1 INTRODUCTION

In 2008 the peer-to-peer electronic cash system called Bitcoin was introduced by Satoshi Nakamoto [10]. In the following years, Bitcoin increased in value and popularity. With the increasing popularity of social media, one would assume that the popularity of Bitcoin would be expressed on such platforms. Kaminski [8] suggests that the Twitter platform may be interpreted as a virtual trading floor that emotionally reflects Bitcoin's market movement, but his results show that Twitter sentiment is not predictive for Bitcoin market movement. Rather than focusing on the sentiment of Twitter, this work focuses on specific keyword usage of Twitter users overtime, such that models can be trained to predict whether or not a profitable trade can be made.

The model that this work focuses on is called a Convolutional Neural Net (CNN). CNNs have previously been used for problems where the data consisted of spatial relationships such as AlexNet, which is a CNN that predicted class labels of (ImageNet) images [9]. The structure of a CNN, however, can also be changed to handle one-dimensional sequences. This allows for dealing with other types of data that have spatial relationships. The problem that is tackled consists of a time series (i.e. a one-dimensional temporal relation). This allows for the possibility of finding patterns in the usage of words for specific users at certain time intervals.

The objective of this work is to find if it is possible to develop a model for predicting profitable trades based on specific Twitter users their messages. Furthermore, focusing on a specific set of Twitter users makes room for the usage of user specific meta-data, such as the amount of followers, which might hold meaning when predicting profitable trades based on user-specific Tweets. This work aimed to answer the following questions:

(1) Can a set of Tweets from specific Twitter users be used to predict profitable trades for Bitcoin?
(2) Does the predictive value of the aforementioned model increase when utilizing meta-data of this specific set of users?

2 RELATED WORK

As mentioned in the introduction, Kaminski [8] concluded that the Twitter platform may be interpreted as a virtual trading floor that emotionally reflects Bitcoin's market movement. However, Kaminski performed a dynamic Granger causality analysis and reports that it does not confirm a causal effect of emotional Tweets on Bitcoin market values, meaning that Twitter sentiment does not precede changes in Bitcoin market values. Hernandez et al. [3], in their work, explained that Twitter users that are interested in Bitcoin are less likely to use emotional related words in their Tweets. Sul et al. [13], when predicting stock returns, found that sentiment of Tweets coming from users with many followers had no significant impact on the return of a stock and they observed the opposite when users had fewer followers. Although their work focused on the stock market, this could be an indication that user meta-data (e.g. amount of followers), is of importance when measuring price changes. The combination of this previous work raised the idea of focusing this work on predicting profitable trades for Bitcoin by focusing on a fixed set of Twitter users. This would allow for using user's meta-data, and instead of relying on Twitter sentiment, we would rely on the user's their word-usage over a fixed time interval such that predictions can be made based on word usage patterns.

Previous work regarding the development of models for trading include Hoseinzade et al. [5] their work, which focused on using CNNs to predict the next day's movement of certain indices of S&P 500, NASDAQ, DJI, NYSE, and RUSSELL markets by using various features such as the RSI (Relative Strength Index).

For Bitcoin specific trading, Hotz-Behofsits et al. [6] developed non-Gaussian state space models and focused on the daily change in the log price of Bitcoin, Ethereum and Litecoin. They developed a simple trading strategy for optimizing portfolio allocation. For Bitcoin specific trading there is one interesting mention that, according to their results, achieved great success. This paper was

TxMM'18-'19, Radboud University, Nijmegen, Netherlands

released by Devavrat et al. [11], and used data coming from the order book of the exchange 'Okcoin.com' to perform Bayesian regression such that the model gave indications as to whether or not they should buy, sell or hold Bitcoin. After 60 days, they claim to have doubled their investment.

3 APPROACH

3.1 Data

3.1.1 Data Gathering. The method for gathering data that is used in this work is based on prior work done by Sul et al. [13]. Their work mentions that Twitter users use a dollar sign and a shortcut of a certain stock as an indication that they are Tweeting about a particular stock. In the case of this work, this means that Tweets are collected that contain '$btc' (not capital letter sensitive). Collecting Twitter data consisted of using Tweepy (a Python package) [14], and after collecting data between the period of 26th of September (2018) to the 2nd of January (2019), the Tweets were processed and a list of unique usernames was generated. Iterating over each user and gathering all their Tweets that contained the aforementioned keyword resulted in the final dataset[1]. The resulting Tweets were preprocessed by removing Retweets, Tweets with more than one ticker symbol sign[2], URLs, mentions, excessive characters (e.g. 'goood' is transformed to 'good'), unicode characters, stopwords, digits that indicated a price and punctuations. Lastly, all characters were converted to lower-case characters and all Tweets prior to June 1st 2017 were removed. Figure 1 shows the final distribution of the entire dataset before splitting the dataset into a training and test set. Before splitting, the data was ordered such that a clear temporal cut could be made. The test data consisted of the last two months of data (1st of November 2018 to the 2nd of January 2019). This allowed for making predictions in the 'future' as the data in the data coming from dates in the test set lied ahead in time with respect to the training data. The total amount of Tweets for the training set equaled 190.038 and the total amount of Tweets for the test set equaled 62.644, where all users that were not both in training and test data were removed.

[1]This approach was used due to Twitter limiting access to historical data when using specific keywords (i.e. scraping can be done up to 14 days back in time)

[2]According to Sul et al. [13] multiple ticker symbols makes it hard to differentiate between which subject a message is about. Meaning a combination of e.g. $ETH for Ethereum and $BTC for Bitcoin in one message makes it unclear as to which one of the tickers the Tweet is about.

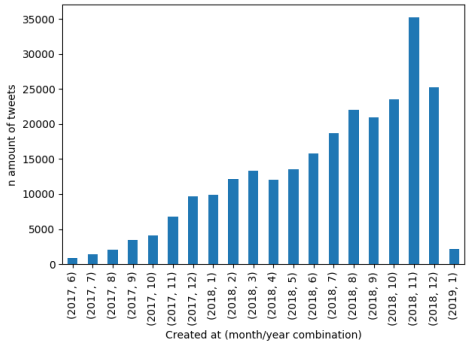


Figure 1: Distribution of final set of Tweets per month and year combination.

Besides Tweets, users their meta-data was also retrieved using the aforementioned package Tweepy. As a user's meta-data changes overtime (e.g. people gain more followers), only the most recent meta-data for a particular user was used for the experiments. The set of meta-data consisted of: If the user's account was verified or not, the amount of followers the user had, amount of accounts the user followed, the amount of public lists the user was in, total amount of Tweets the user liked, number of Tweets created by the user, the time the account user has existed for expressed in months and the user's twitter ID.

To determine whether or not a trade would have been profitable for a particular day, the Bitcoin closing price of a day prior to that day was subtracted from the Bitcoin closing price of that particular day. A fee was then applied to the resulting value that has to be paid when making a trade. If the resulting calculated value was positive, then the trade would have been profitable, else it would not have been profitable. Bitcoin price information was retrieved from CryptoDataDownload [2], which is a platform that aims to provide free historical data for several cryptocurrencies. As Bitcoin prices can differ per platform, this work focused on Coinbase's price fluctuations and thus their fees were applied in the aforementioned calculation [1].

3.1.2 Matrices. CNNs require a matrix as input, where the goal is that the matrix captures some form of a pattern that can be used as an indicator for the class that it represents. The matrices existed of the amount of features as rows and the amount of timesteps as columns. These features were derived from Tweets, where Tweets contained words and/or specific characteristics that were informative for prediction profitable trades. This way, a time series was created that captured the change in features (i.e. word usage) for a specific amount of timesteps.

First, TF-IDF was used to generate the importance of specific words with respect to the entire set of training tweets. Second, the top 500 words with the highest TF-IDF scores were used as individual rows for the matrices. Third, the amount of columns for a matrix was fixed to 7, meaning that a matrix's shape consisted of 500 rows by 7 columns. Fourth, Tweets were represented in the aforementioned matrices by processing Tweets for specific users of specific dates and seeing whether or not the top 500 TF-IDF generated words occurred in a Tweet for a specific date. To create a

single matrix, this last step was repeated 7 times for each date for a specific user. The final matrix captured the change in specific word usage for a user over a period of 7 days. Using the aforementioned method, four types of matrices were created. The first matrix was a binary matrix consisting of zeroes and ones, where ones represented a TF-IDF word occurring in a Tweet for a specific user and date combination, and where a one meant that there was a full match between a word in the Tweet and a word in the TF-IDF generated list (i.e. meaning that the word 'fear' only matches 'fear' and not 'fearful). The second type of matrix was similar to the previous matrix, but instead of being binary, it counted the occurrences of a word in a tweet, meaning that matrices did not solely consist of zeroes and ones. The third and fourth matrix were similar to the aforementioned matrices, but used partial matching, meaning that a word such as 'fear' also matched 'fearful' as 'fear' is a subword of 'fearful'. Figure 2 shows an example of a binary matrix for partial matching, where the x-axis represents the columns (i.e. 7 days) and the y-axis represents the number of features (i.e. TF-IDF words), and where the occurrences of a word are represented by a black stripe. Throughout this work partial matching will be referred to as 'Partial TF-IDF' and full matching will be referred to as 'TF-IDF'.

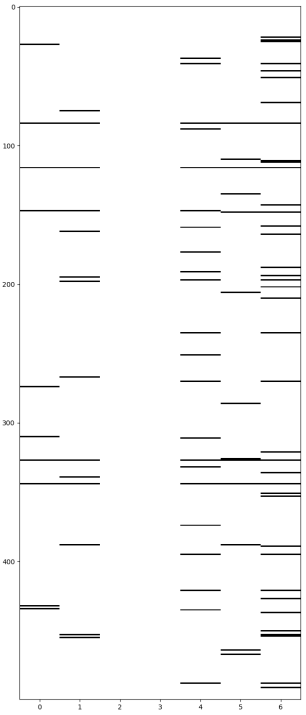


Figure 2: Example binary matrix for partial matching.

User meta-data and classes were not processed in matrices as this did not include a spatial relationship. User meta-data was concatenated in the network, which will be discussed in the next section. The labels the models were trained on were created by storing whether or not a trade would have been profitable the day after the last day of the matrix[3]. As a last step, all matrices and users their meta-data were normalized.

Table 1 shows the total amount of matrices that were gathered per type of matrix representation. One can observe that the amount of matrices per representation differs, which is due to matrices only being non-zero if the TF-IDF words occurred in a Tweet.

Table 1: Size of training and test set for each model.

	Train	Test
Partial TF-IDF	126.949	35.750
TF-IDF	24.278	7.003

3.2 Models

This section introduces the methods that were used for predicting profitable trades for Bitcoin. Making a prediction for a particular date consisted of two steps. The first step consisted of making a prediction per user and date combination. The second step consisted of combining these prediction into a single final prediction for a particular date.

3.2.1 User and date specific predictions. The per user and date combination predictions were made by using a CNN. This CNN used the matrices explained in Section 3.1.2 as an input and resulted in a probability, which indicated whether or not the model thought a trade should be made. The model itself consisted of four convolutional layers followed by three fully connected layers and a Sigmoid layer. All layers made use of batch-normalization layer and the fully connected layers also made use of Dropout (with a keep probability of 0.5) to prevent overfitting [12] [7]. The user meta-data was concatenated to the last fully connected layer. For completeness, an overview of the architecture was added to Appendix A.

3.2.2 Date specific predictions. Two methods were used for combining the user and date specific predictions.

The first method consisted of using the arithmetic mean over all predictions, which resulted in a probability per day. The final score then consisted of converting the resulting probability in a binary score (with the decision boundary being 0.5, meaning $p<0.5$ represented no trade, else do trade).

The second method that was used was LR. Using LR required a training set and thus some preprocessing of the data was performed. As the focus lied on a fixed set of users, the data was represented as an array of length k. This array was instantiated as an array of all zeroes, where k was the total amount of users (which was equal in both training and test data as was mentioned in Section 3.1.1) and each individual row was filled with probabilities, where each cell represented a prediction for a specific user and date combination. For both the training and testing data, the labels consisted of the same labels as the CNN used (i.e. if a trade should be made such that a profit is made the day after). The task of the LR model was to attempt to predict these labels based on the given data. The LR model was trained on the probabilities that the CNN model generated by predicting on the training set. Afterwards, the LR model

[3]Meaning that if a matrix consisted of columns between t-7 and t, where t was an arbitrary date, the class consisted of if the trade would have been profitable at t+1.

was tasked to make a final prediction by using the probabilities of the test set that were generated by the CNN.

3.3 Evaluation

The experiments were performed by making use of 5-fold cross validation. Success was measured by evaluating the Precision and Recall.

Precision, in the case of this work, represented the amount of trades that were succesfully made divided by the amount of trades that were succesfully made and the trades that were made, but should not have been made (i.e. resulting in a loss). This means that the Precision signified the amount of times a profit was made with respect to the total amount of trades made.

Recall, in the case of this work, represented the amount of trades that a trader has succesfully made divided by the amount of trades that were succesfully made plus the amount of trades that were not made, but should have been made. This means that the Recall signifies the amount of times a profit was made with respect to the total amount of profitable trades (including the ones that were not made).

Both measures are important, but this work assumed that the Precision is of the highest importance as a low Precision resulted in the loss of overall capital, while a low Recall meant the loss of trading opportunities. As such, the goal of the trader was translated to finding a model that had the highest Precision, which in turn had the capacity to improve the overall capital of the trader while taking the least amount of risk.

As a last measure, confusion matrices were used for further analysis as will be shown in the following sections.

4 RESULTS

During training time, the learning rate was initialized at 0.001 and was reduced overtime when the validation loss did not improve. Furthermore, training was stopped if overfitting seemed to occur by applying early stoppage [4]. The max amount of epochs was set to 35, but all models stopped prior to that due to early stoppage.

In total, 40 experiments were performed, where each experiment consisted of a unique combination of batch size (32, 64, 128, 256 or 512), usage of user meta-data, whether to represent occurrences of words in binary or count format and usage of partial or full matching. Reporting the results limits itself to the best performing batch size for a particular experiment combination. In this case, 'best performing' meant what served the trader's goal best, which was a score which would have allowed a trader to make a profitable trade. In this case, as mentioned in Section 3.3, the assumption was made that Precision was the most important score. Furthermore, the score that would have allowed a trader to make a trade was based on the aggregated daily score, meaning that the Precision for either one of the aggregation methods had to be the highest. Table 2 shows the final results, where the scores are represented as arithmetic means for 5-fold cross validation. As the Precision and Recall scores did not include True Negatives, the results are extended by also reporting confusion matrices for all the aggregated models their results, which can be found in Appendix B. These confusion matrices are the result of 5-fold cross validation and thus the tables represent the arithmetic means over all folds.

Table 2: Precision/Recall scores on the user-specific train, validation and test set. Remaining two columns represent the Precision/Recall scores for daily aggregated predictions using LR and the arithmetic Mean

				Train		Val		Test		LR		Mean	
Model	**Type**	**User Meta-data**	**Batch size**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
TF-IDF	**Binary**	No	512	0.93	0.15	0.55	0.08	0.16	0.08	**0.46**	**0.34**	0	0
		Yes	512	0.70	0.25	0.43	0.15	0.41	0.13	**0.46**	**0.62**	0	0
	Count	No	512	0.82	0.27	0.38	0.15	0.44	0.16	**0.49**	**0.54**	0	0
		Yes	512	0.76	0.5	0.44	0.3	0.42	0.29	**0.43**	**0.71**	0.13	0.02
P. TF-IDF	**Binary**	No	512	0.68	0.35	0.48	0.25	0.43	0.22	**0.48**	**0.16**	0	0
		Yes	64	0.66	0.33	0.48	0.24	0.43	0.21	**0.49**	**0.17**	0	0
	Count	No	256	0.71	0.35	0.48	0.23	0.44	0.21	**0.56**	**0.18**	0	0
		Yes	512	0.67	0.33	0.47	0.24	0.44	0.22	**0.50**	**0.13**	0	0

5 DISCUSSION

This section discusses the results and reflects on the research questions that were stated in Section 1.

5.1 Predicting profitable trades

In terms of the first research question, one can observe that within this experimental setup, using Partial TF-IDF rather than TF-IDF resulted in higher scores. This could be due to the example given in Section 3.1.2, but it could also be due to the fact that Partial TF-IDF had more than five times as much training data (see Table 1). Especially for Partial TF-IDF, representing occurrences as counts rather than binary values seems to improve the overall score, a reason for this could be that there are more candidates for Partial TF-IDF to match and as such, it could carry more meaning. When evaluating the scores, one could blindly look at the scores and conclude that Partial TF-IDF gave a Precision of 0.56 when predicting using LR, counting occurrences and without the usage of user meta-data. However, further inspection of the confusion matrix (see Appendix B.7 and the corresponding Precision, indicate that there is a low amount of profitable trades that were recognized by the model. Overall, when evaluating the scores, it seems that a higher Precision comes paired with a lower Recall, meaning that many potential trades were not recognized by the model. Furthermore, the Mean score is mostly zero, which is due to the mean mostly predicting only false and true negatives, and as such, the Precision and Recall is also zero (see Appendix B.7 for an example). In Section 3.3, the goal of the trader is defined and thus it is of importance to reflect on that with respect to the current research question. In terms of being able to predict profitable trades, it is too preliminary to conclude whether or not the model can be used in real world scenarios. As such, further research is required, such as experimenting with actual trading such that variables as order delays can be experimented with.

5.2 Relevance user meta-data

In terms of the second research question, one can observe that within this experimental setup, including user meta-data does for some experiments result in an improved Recall, but in general results in a lower Precision. The total amount of users that were used was 7351, meaning that the average amount of Tweets per user was around 17 for Partial TF-IDF and around 3 for TF-IDF. This could be an indication that the sheer amount of data per user resulted in the user meta-data not being as important as it could have been. Further research is required with a higher amount of

data per user to observe whether or not the user meta-data is of significance or not.

6 CONCLUSION & OUTLOOK

In general, this work can be concluded by stating that more research is required to draw conclusions as to whether or not user-specific Twitter data is of significance when predicting Bitcoin prices. As such, the added value of this work is that it may serve as a baseline for future work. This work is not sufficient for traders to work with as this would require additional extensive experimenting and statistic testing. Lastly, it is important to emphasize that the total amount of data that was gathered was relatively small, especially when taking into account the average amount of Tweets per user. As such, future work includes the continuous gathering of Tweets and testing whether or not this improves the scores overtime. It may well be that the scores increase as the amount of data increases. To extend this research, the following points of action are currently being considered:

- Extending the problem to a multi-label problem. When trading, one can also 'short', meaning that the trader bets that the price of a certain valuable will go down. In such a case, a three-label problem could be interesting where a -1 represents a short, a +1 represents a long[4] and a 0 represents that the trader should not trade at this point-in-time.
- Extending this work by adding more features such features that were mentioned by Hoseinzade et al. [5], where instead of deriving features from indexes, one would derive features from Bitcoin meta-data (e.g. price data).
- Experimenting with longer time intervals. This work focused solely on time frames of 7 days, but perhaps patterns in the data become more clear when increasing the time frame to a different time frame.

REFERENCES

[1] [n. d.]. Digital Asset Exchange. ([n. d.]). https://pro.coinbase.com/fees

[2] [n. d.]. Free CryptoCurrency Time Series Data. ([n. d.]). https://www.cryptodatadownload.com/index.html#features

[3] Ivan Hernandez, Masooda Bashir, Gahyun Jeon, and Jeremiah Bohr. 2014. Are Bitcoin Users Less Sociable? An Analysis of UsersâĂŹ Language and Social Connections on Twitter. 26–31. https://doi.org/10.1007/978-3-319-07854-0_5

[4] Thomas Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine learning* 42, 1-2 (2001), 177–196.

[5] Ehsan Hoseinzade and Saman Haratizadeh. 2018. CNNPred: CNN-based stock market prediction using several data sources. (2018). arXiv:arXiv:1810.08923

[6] Christian Hotz-Behofsits, Florian Huber, and Thomas O. ZÃűrner. 2018. Predicting crypto-currencies using sparse non-Gaussian state space models. (2018). arXiv:arXiv:1801.06373

[7] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[8] Jermain Kaminski. 2014. Nowcasting the Bitcoin Market with Twitter Signals. (2014). arXiv:arXiv:1406.7577

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1097–1105. http://papers.nips.cc/paper/4824-imagenet-classification-with\-deep-convolutional-neural-networks.pdf

[10] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[11] Devavrat Shah and Kang Zhang. 2014. Bayesian regression and Bitcoin. (2014). arXiv:arXiv:1410.1231

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[13] Hong Kee Sul, Alan R Dennis, and Lingyao Yuan. 2017. Trading on twitter: Using social media sentiment to predict stock returns. *Decision Sciences* 48, 3 (2017), 454–488.

[14] Tweepy. 2018. tweepy/tweepy. (Dec 2018). https://github.com/tweepy/tweepy

APPENDIX

A CNN ARCHITECTURE

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv1d-1              [-1, 256, 7]         640,256
       BatchNorm1d-2              [-1, 256, 7]             512
            Conv1d-3              [-1, 128, 5]         163,968
       BatchNorm1d-4              [-1, 128, 5]             256
            Conv1d-5               [-1, 64, 5]          24,640
       BatchNorm1d-6               [-1, 64, 5]             128
            Conv1d-7               [-1, 64, 5]          12,352
       BatchNorm1d-8               [-1, 64, 5]             128
            Linear-9                 [-1, 128]          41,088
      BatchNorm1d-10                 [-1, 128]             256
          Dropout-11                 [-1, 128]               0
           Linear-12                  [-1, 32]           4,128
      BatchNorm1d-13                  [-1, 32]              64
          Dropout-14                  [-1, 32]               0
           Linear-15                   [-1, 1]              33
      BatchNorm1d-16                   [-1, 1]               2
          Dropout-17                   [-1, 1]               0
          Sigmoid-18                   [-1, 1]               0
================================================================
Total params: 887,811
Trainable params: 887,811
Non-trainable params: 0
```

Figure 3: Model summary of CNN

B CONFUSION MATRICES

Due to the underlying code that was used for this project, there is an error when calculating the Precision/Recall scores and comparing them to Table 2. As a result, it is advised to look at the aforementioned table for reference as these scores were rounded manually.

Note: *The following subsection s consist of confusion matrices for each row in Table 2, where the first subfigure is the confusion matrix for Logistic Regression and the second subfigure is the result for the Mean prediction (both scores were calculated using the test set). Furthermore, due to the training set not exactly fitting the amount of*

[4]Meaning that the trader things the price will go up, which is equal to making a profitable trade in this work.

batches (meaning $n \mod b \neq 0$, with n being the size of the dataset and b being the batch size), the amount of values in the cells in the confusion matrices can differ between batch sizes.

B.1 TF-IDF, Binary, without user meta-data and batch size equal to 512

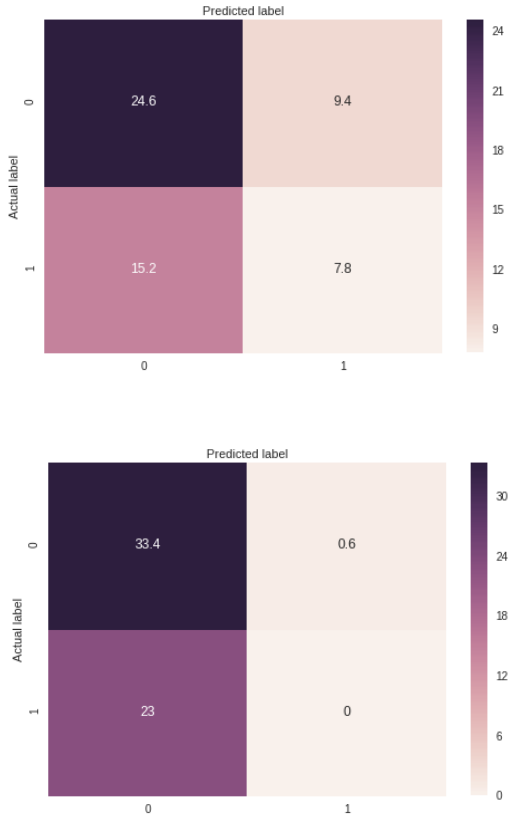


Figure 4: Confusion matrices LR and Mean (see settings in Section title)

B.2 TF-IDF, Binary, with user meta-data and batch size equal to 512

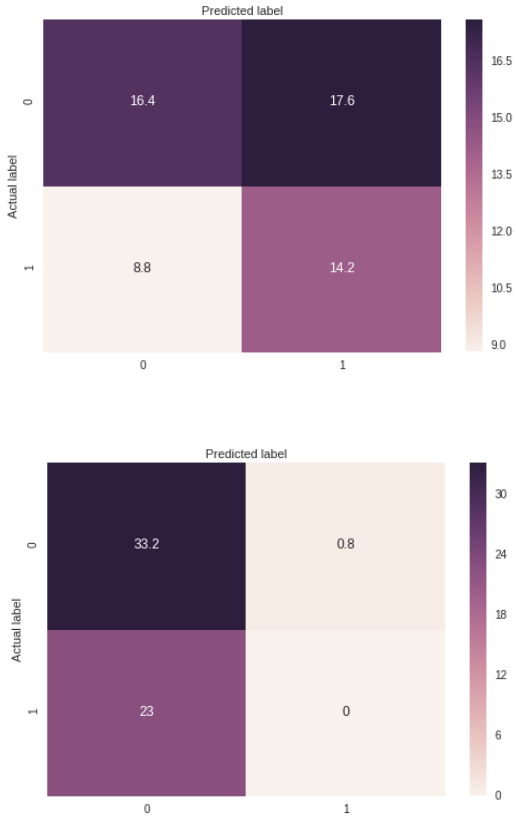


Figure 5: Confusion matrices LR and Mean (see settings in Section title)

B.3 TF-IDF, Count, without user meta-data and batch size equal to 512

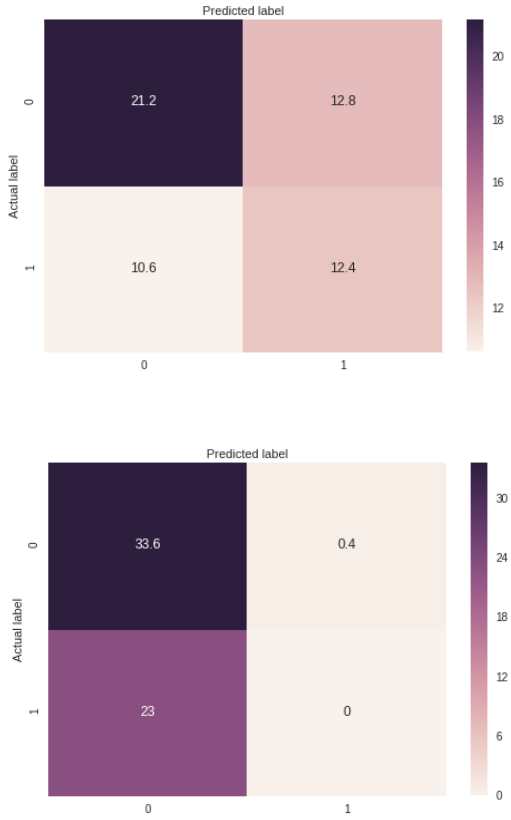


Figure 6: Confusion matrices LR and Mean (see settings in Section title)

B.4 TF-IDF, Count, with user meta-data and batch size equal to 512

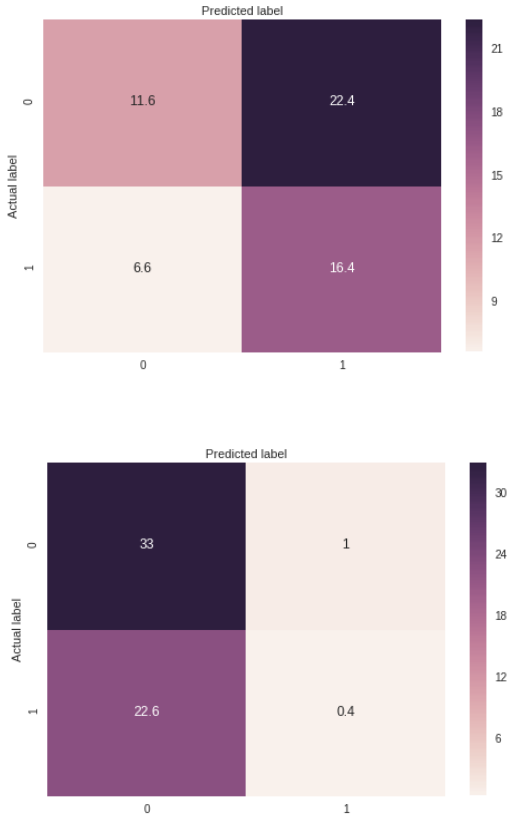
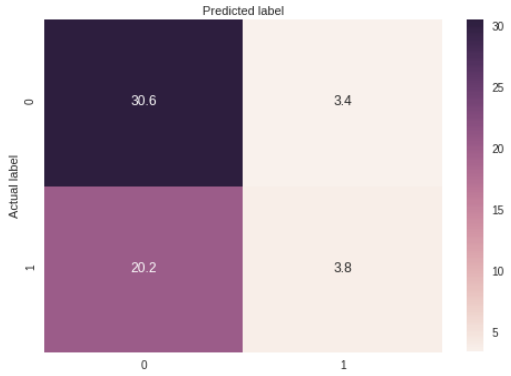


Figure 7: Confusion matrices LR and Mean (see settings in Section title)

B.5 Partial TF-IDF, Binary, without user meta-data and batch size equal to 512

B.6 Partial TF-IDF, Binary, with user meta-data and batch size equal to 64



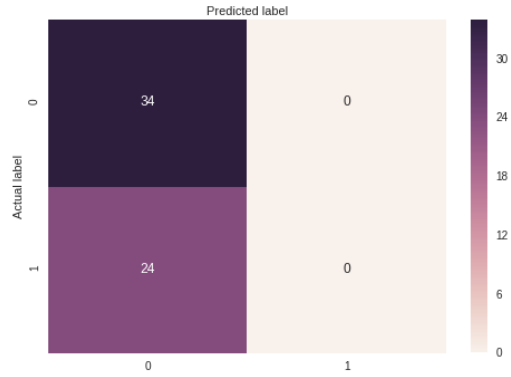
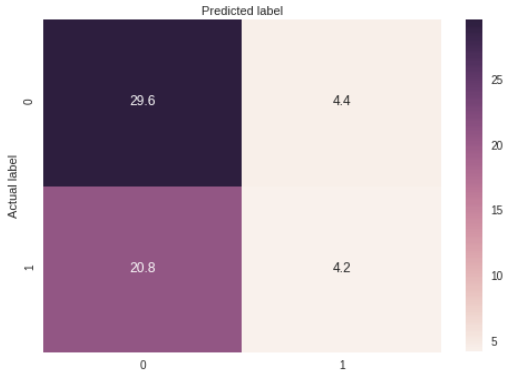


Figure 8: Confusion matrices LR and Mean (see settings in Section title)



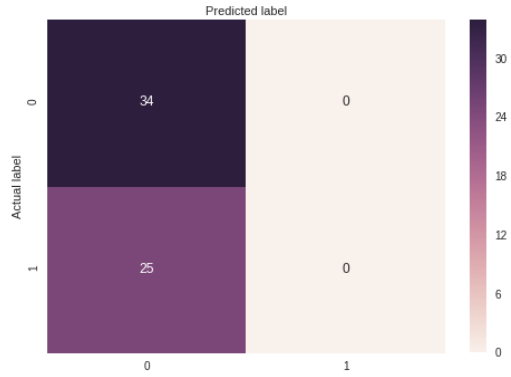


Figure 9: Confusion matrices LR and Mean (see settings in Section title)

B.7 Partial TF-IDF, Count, without user meta-data and batch size equal to 256

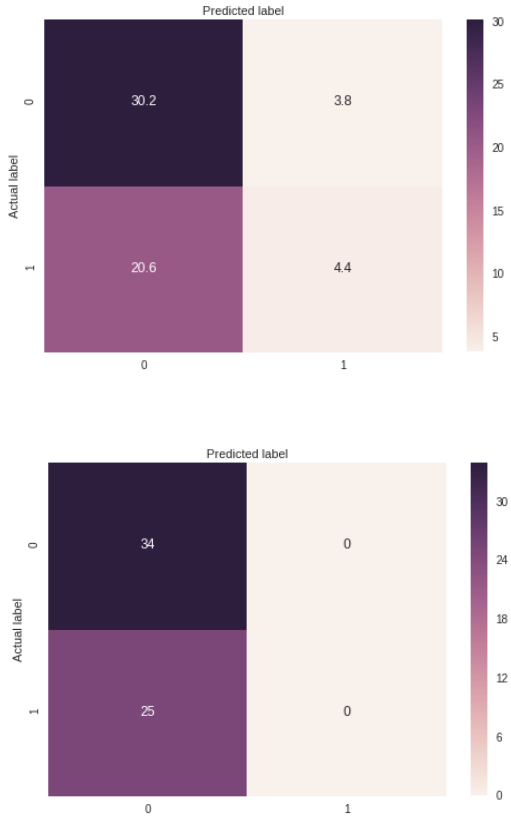


Figure 10: Confusion matrices LR and Mean (see settings in Section title)

B.8 Partial TF-IDF, Count, with user meta-data and batch size equal to 512

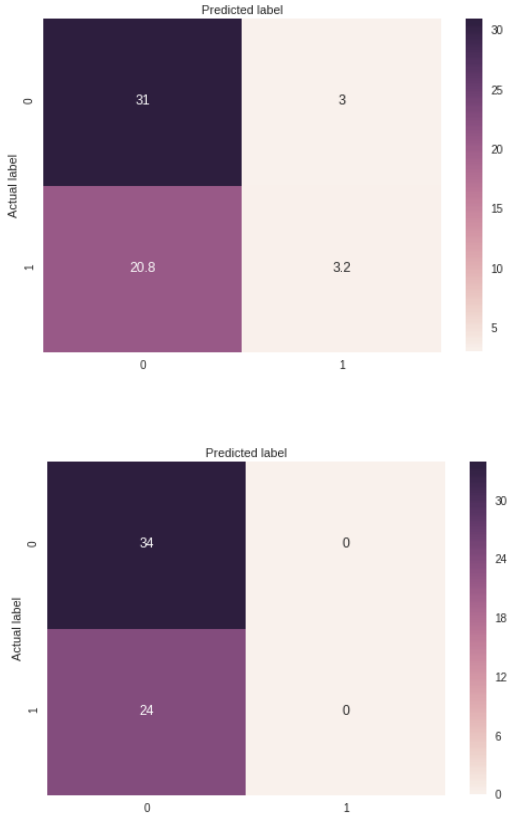


Figure 11: Confusion matrices LR and Mean (see settings in Section title)