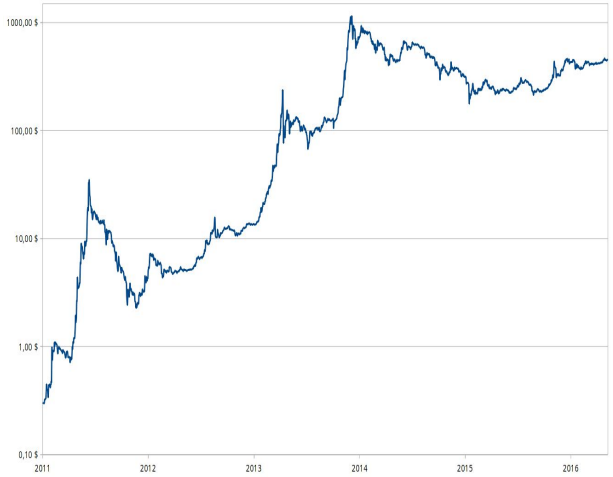
Accuracy of flipping a (Bit)coin

Prediction of Bitcoin prices using Machine Learning techniques



Katja Bouman & Tom Lotze

Machine Learning Fall 2017

Gosia Migut

Final project

22-12-2017

Problem description

In 2008 the Bitcoin (BTC) was introduced, a revolutionary cryptocurrency that uses the Blockchain technology to enable a peer-to-peer, decentralized financial system. Transactions and value in the network are based on cryptography, rather than on a bank or government, hence the name. The popularity of not only Bitcoin, but all kinds of cryptocurrencies has skyrocketed over the past 5 years. The price of 1 BTC was $1000 in the beginning of 2017, and recently the price reached $18,000, and shows an exponential growth rather than linear, as can be seen in Figure 1.

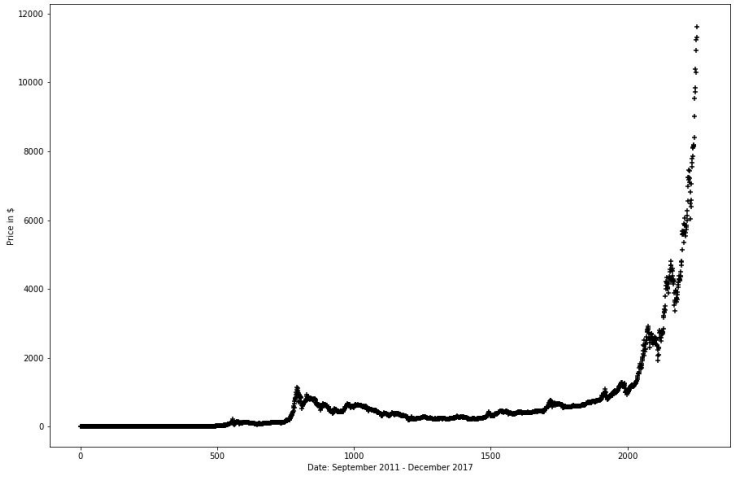


Figure 1: Price development of Bitcoin from 2011-2017

For now, there is one main obstacle in using Bitcoin and related cryptocurrencies as actual payment method, namely the very fluctuating value, and the astronomical increase in value over the past few years. As a result of this deflation, people are less likely to actually use their cryptocurrencies to pay for goods. This phenomenon is known as the deflationary spiral, in which consumers don't spend their money because the value is increasing [Barber et al., 2012].

Currently, Bitcoin is used as an investment rather than a functional currency. It is estimated that only a million people possess Bitcoin right now. The volatile price developments make it possible to make huge profits, but also huge losses when investing in cryptocurrencies. Crucial for investing is optimal timing in terms of buying and selling. Hence, in this project, we will analyze the historical price data of Bitcoin and with that, we hope to be able to train different classifiers and predict the future Bitcoin price as accurately as possible.

Comparable machine learning problems are the prediction of stock prices, which was already done in the 1990's using neural networks [Schöneburg, 1990]. This research performed day to day prediction of German stock prices and achieved a accuracy of 90% using only previous price information. However, the Bitcoin price is significantly more volatile and we expect it to hold on to a pattern to a lesser extent, which makes it harder to predict. Moreover, given the age of BTC, there is not so much training data available to find patterns.

Approach & Justification

For our analysis, we used a dataset that consisted of 2255 instances (days), from September 13, 2011 until December 5, 2017 and per instance it contained the following features about the bitcoin price and its trading volume: 'Date', 'Open', 'High', 'Low', 'Close', 'Volume (BTC)', 'Volume (Dollar)' and 'Weighted Price'. Using these data it is possible to construct more complex features, the first one being the relative change per day (in percents), which we computed in terms of the opening and closing price.

The second one is a widely used indicator for price analysis, the x-day simple moving average (SMA), which is the average over the previous x days. These moving averages show the more global trends in price development by filtering out noise. The SMA follows the trends since it is based on previous values, but intersections between SMA's can indicate price change. We arbitrarily chose to construct the 5-day-, and the 15-day moving average by taking the averages over the weighted prices. However, we did not add both moving averages as features, but rather added the ratio between the two as a feature, since the ratio between these two moving averages is especially interesting for price analysis.

After adding the 'Change' and 'Moving average ratio' as features and testing with the complete dataset (excluding 'Date'), we decided to remove all the features except for the 'Weighted Price', 'Change' and 'Moving average ratio', to prevent overfitting on the training data. We reasoned that the opening and closing price of the previous day was not very indicative of price change, whereas the difference between the two (the 'Change') is. We were aware of the fact that three features might not be enough to predict the price very accurately, but we wanted to check if there was any predictive power in the features that we had. We could have added more complex features, or find a more extensive dataset that allowed us to use more features, but decided to check first whether or not it was possible to predict the price in some way.

As for the labels, we expected that predicting the exact price would be too optimistic and we decided to instead predict a discrete output, i.e. classes. We tried 2 different label sets, one

that was binary and contained only the sign of the change (1 for increase and 0 for decrease), and a more extensive label set that had 5 classes: ‘stable’ (-1% < x < 1%), ‘small increase’ (1% < x < 10%), ‘small decrease’ (-10% < x < -1%), ’large increase’ (x > 10%) or ‘large decrease’ (x < -10%), where x is the change per day in percent. In the end we chose for the binary classification, because after testing both methods, the binary classification performed better and was more simplistic. The distribution of the labels can be seen in Table 1 below.

	Decrease (0)	Increase (1)	**Total**
Training set	782	1018	1800
Test set	180	275	455
Total	962	1293	2255

Tabel 1: Distribution of the labels across (non-shuffled) training- and test set

Based on Table 1 and given the small number of instances in our dataset, we decided to take a test set size of a little bit over 20%. A test set of 30% for example would reduce the training phase significantly, which we did not want. Furthermore we chose to not shuffle the data before splitting the data, which made the test set also the last 20% of the data in time. We also tested all the classifiers on a shuffled training- and test set, the results of which can seen found in the results section. Since the non-shuffled method returned the best results on the test set (classifiers that predicted a single value not included), we decided to continue with the non-shuffled training- and test set.

Results

We have tried a great number of different classifiers, both classification and regression models, which are specified below. For all the classifiers we used the same train and test set, which was retrieved by splitting the unshuffled data and selecting the last ~20% for the test set and the rest for the training set. Also the features were identical for all classifiers.
To evaluate the performance of our models we used a few different methods. First we looked at the accuracy of the predictions of the test instances. Also, we used cross validation to determine whether our model was suffering of overfitting. Lastly, we used a confusion matrix and a classification report to visualize the type of mistakes the model was making.

We started with the following classification models: Logistic regression, Neural Networks, Decision Tree, Random forest, and Support Vector Machine. As mentioned before we have tried multiple different implementation approaches, namely random test instances or an unshuffled test set and we tried a different number of output classes. Table 2 below shows the corresponding accuracy scores. Overall the results were worse than we expected, and hoped. The majority of the classifiers had a accuracy of less than 50% on the testset. This means the classifiers performed worse than a random prediction would do, as we only used the two classes increase and decrease.

Shuffling had some effect on the accuracies, as can be seen in Table 2 below. Logistic regression seems to score higher when the data is shuffled, but in fact it still predicts only one output class, only the one that occurs most often now, hence the higher accuracy. As explained in the approach section, the highest (non-biased) accuracy in the non-shuffled variant (SVM, 0.563), is higher than the one in the shuffled section (Neural Networks, 0.510)

As shown in Table 2 below, the Logistic Regression classifier was the worst performing model with an accuracy of less than 40%. The Neural Network has the highest accuracy. However, further investigation shows that this is due to the fact that the model classified all instances as increasing, and 60% of the examples in the test set have the label 'increasing', as can be seen in Table 1.

	Logistic Regression	Neural Networks	Decision Tree	Random Forest	SVM
Not shuffled	0.396 *	0.6 *	0.514	0.459	0.563
Shuffled	0.609 *	0.510	0.479	0.481	0.611 *

*Table 2: Accuracy scores of the classifiers on the test set, values marked with * means that the classifier predicted only a single outcome.*

The figures 2 to 6 below show the confusion matrices of all the classification models using a heat map, with the fractions predicted in the two labels indicated. It is clear that both Logistic regression and Neural Network were strongly biased towards one class. This is very undesirable as it is a sign of a bad performing classifier. The predictions of other three classifiers are a bit more equally distributed, however Random forest and SVM also tend to be biased towards classifying one class. Such imbalanced classification problems often occur when a dataset is unequally distributed and one class is over represented. However, this imbalance is not exceptional in our data set. Note that not all classifiers have a bias in

favor of the same class, despite being trained on the same set. An explanation for this can probably be found by inspecting the exact training mechanism per class, which we did not focus on.

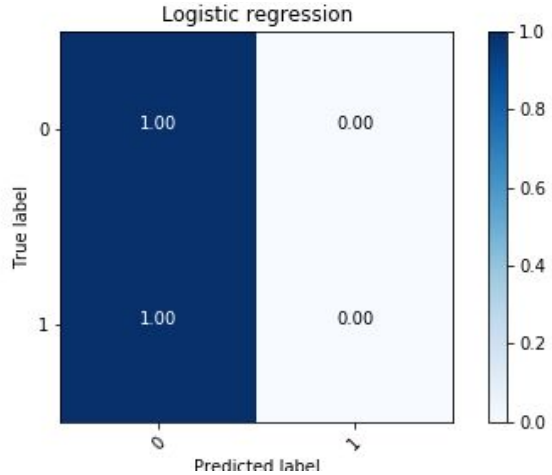


Figure 2: Confusion Matrix Logistic Regression

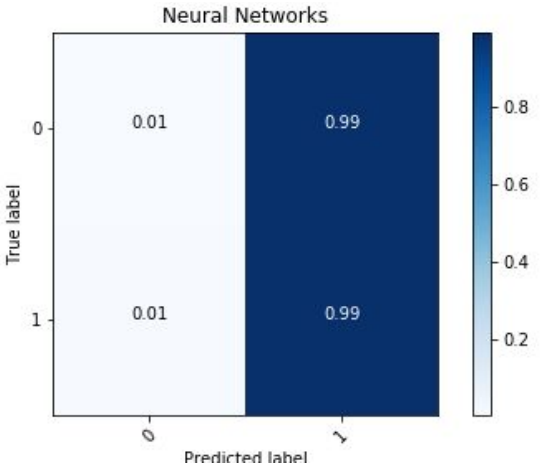


Figure 3: Confusion Matrix Neural Networks

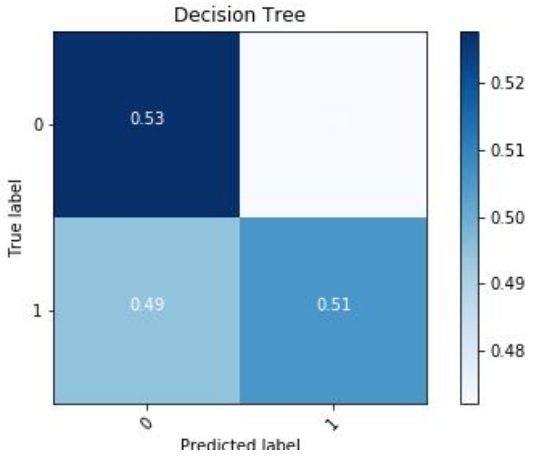


Figure 4: Confusion Matrix Decision Tree

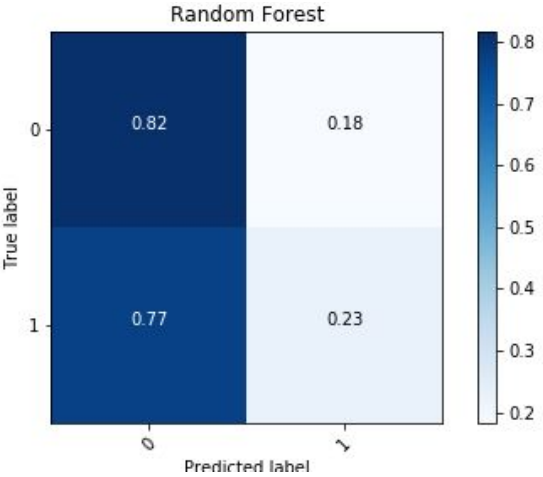


Figure 5: Confusion Matrix Random Forest

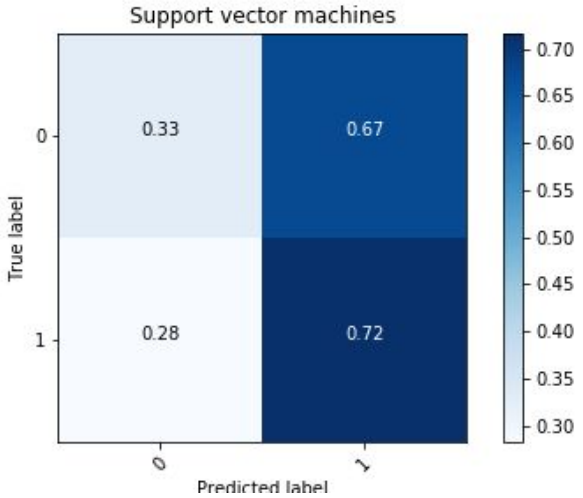


Figure 6: Confusion Matrix SVM

As SVM was the best performing classifier, and hence we tried to optimize this classifier. This suggested a sigmoid kernel with parameters C equal to 1000 and Gamma equal to 0.1. Though this greatly improved the performance on the cross validation set, with an average accuracy of about 60% and on some cross-validation sets accuracies of over 75%, it still performed rather bad on the test set, which implies overfitting on the training set.

We have also implemented different regression models, namely Linear Regression, Support Vector Regression (SVR) and Random Forest Regression. For the regression models we tried multiple different combinations of features, by excluding specific features. This let to the conclusion that solely using the dates and the opening price leads to the best results. Evaluation of regression models needs to be executed in a different way then the classification models as the predicted values are continues. We used graphs to visualize the similarity between the data and the predicted values of the models. Other evaluation methods are possible, but for us the graphs gave a sufficient implication of the performances. Linear Regression (Figure 7) performed clearly better than the other two classifiers (data not shown). However, the accuracy does not seem to be better than the classification models.

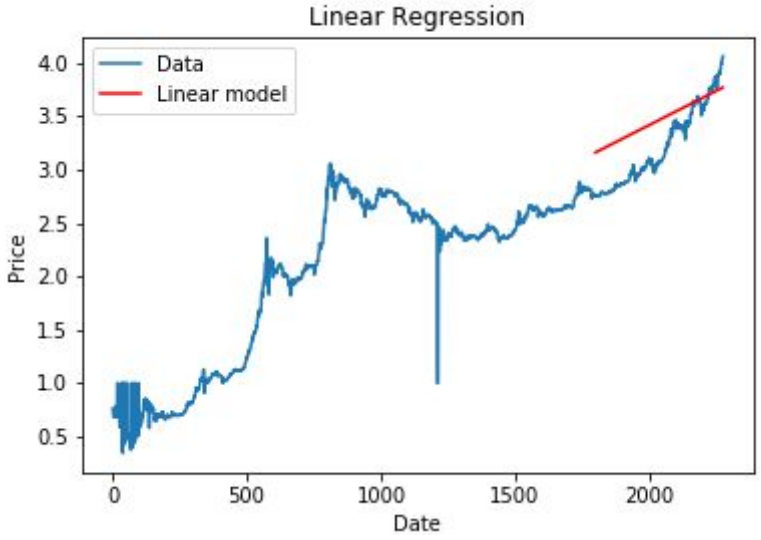


Figure 7: Linear regression model, prediction on the test set

Evaluation of approach and results

First of all, the Bitcoin price is very volatile, and the price changes can sometimes be described as random, which implies that prediction might be hard. Also because of this, the results of our approach and testing were not as good as we expected. There can be thought of a few reasons for that, which future research can try to solve.

The first and most obvious one is a shortage of features. Similar studies that tried to model the Bitcoin price specifically that showed better results (98.7% accuracy in predicting the sign of the change per day) [Madan et al. 2015], used many more features, in this case 25. Examples of features we could have added are the number of bitcoins available, most recent block size in the blockchain and average confirmation time (which is an indicator for the number of transactions).

Another problem was the classifiers that consistently predicted one outcome, which are marked with an asterix in Table 2. These classifiers were overfitted on the training set, where increasing was the most frequent label. By only predicting increase, the label that also made up the majority of our test set, these classifiers seemed to have a relative high accuracy on the test set, but in fact did not generalize at all. The basis for this problem lies again in the small number of features, which apparently did not contain enough information to accurately predict the Bitcoin price.

For future research we would first of all recommend using more features. This could be accomplished by using a more extensive dataset in the first place, and construct more features. Furthermore, an extension to this model could be analyzing the news and taking the interest of the public into Bitcoin into account. Example features could be the number of times ‘Bitcoin’ is in the headlines, possible using computational linguistics to determine whether the news is positive or negative. To measure public interest, search statistics could be used for example.

References

Barber S. Boyen X., Shi E., Uzun E. (2012). Bitter to Better — How to Make Bitcoin a Better Currency. Financial Cryptography and Data Security Lecture Notes in Computer Science, 2012, pp. 399–414., doi:10.1007/978-3-642-32946-3_29.

Madan, I., Saluja, S., & Zhao, A. (2015). Automated Bitcoin Trading via Machine Learning Algorithms.

Schöneburg, E. “Stock Price Prediction Using Neural Networks: A Project Report.” *Neurocomputing*, vol. 2, no. 1, 1990, pp. 17–27., doi:10.1016/0925-2312(90)90013-h.