Machine Learning Engineer Nanodegree

Capstone Project

Tony Ng
Nov 5, 2019

1. Definition

1.1 Project Overview

Bitcoin is a digital currency established in 2009 by Satoshi Nakamoto. It is decentralized and transactions are verified by network nodes through cryptography and recorded in a public distributed ledger called blockchain [1]. Bitcoin becomes popular in recent years, and more and more products and services can now be paid by bitcoin. Exchange trading volumes of bitcoin continue to increase. Poloniex, a digital asset trading service, saw an increase of more than 600% active bitcoin traders online and regularly processed 640% more transactions between January and May 2017 [2].

Bitcoin is adopted worldwide; however, its price volatility has also widely been criticized. One bitcoin was worth less than 3000 US dollars at the start of 2017, but it jumped to around 20000 US dollar at the end of that year. A year later, the price fell to around 3000 US dollars again. According to Mark T. Williams, bitcoin has volatility seven times greater than gold, eight times greater than the S&P 500, and 18 times greater than the US dollar [3].

As bitcoin price is hard to predict, people have been trying many different methods to tackle it. One of the methods is to use machine learning techniques to predict the bitcoin price. In Kaggle which is an online community for data scientists and machine learning enthusiasts to post their datasets, ideas, and works, hundreds of posts have been made to try to predict the bitcoin price. In this project, we use a dataset provided by Kaggle [4] to do the same thing.

1.2 Problem Statement

We wish to buy bitcoin for trading and investment; however, we do not want the bitcoin price drop immediately after we have bought it. We need to make sure that when we buy bitcoin, the bitcoin price is going up on that day. To tackle this problem, we train up

a machine learning model using the historical bitcoin price data from Kaggle to predict whether the bitcoin price will go up or down on the next day. We will acquire bitcoins if the model predicts that the price will go up tomorrow. Otherwise, we will wait until one day the model says the price will go up again.

1.3 Metrics

Since it is a supervised learning problem, we use accuracy to evaluate the performance of the model because it is simple and easy to calculate. Furthermore, accuracy is widely and frequently used to measure the performance of machine learning models. We can easily compare the accuracy score of our model with that of other people's models.

We use the prediction of our model to make the bitcoin buying decision. We do not care whether the model can detect all the days on which the bitcoin price is going up. The most important thing that we are concerned is to make sure the price is rising when I buy the bitcoin. In this case, precision is more important that recall. As a result, we choose F-0.5 score as the evaluation metric instead of F-1 score.

2. Analysis

2.1 Data Exploration

The dataset that we acquire from Kaggle includes one CSV file that contains minute-to-minute bitcoin exchange data recorded by Bitstamp from January 1, 2012 to August 12, 2019 [4]. The file contains 3997697 records and have eight columns:

- Timestamp - Start time of time window (1 minute) in UNIX time
- Open - Open price at start time window
- High - High price within time window
- Low - Low price within time window
- Close - Close price at the end of time window
- Volume_(BTC) - Amount of BTC transacted in time window
- Volume_(Currency) - Amount of Currency transacted in time window
- Weighted_Price - Volume-weighted average price

Table 1 is a sample of the raw data:

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
0	2011-12-31 07:52:00	4.39	4.39	4.39	4.39	0.455581	2.0	4.39
1	2011-12-31 07:53:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2011-12-31 07:54:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2011-12-31 07:55:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2011-12-31 07:56:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	2011-12-31 07:57:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	2011-12-31 07:58:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	2011-12-31 07:59:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	2011-12-31 08:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	2011-12-31 08:01:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Table 1

Time windows without any trade have their data fields (except timestamp) filled with NaNs. The records with ‘NaN’ are deleted and the dataset are grouped by daily basis. Then we calculate the following information on each day between January 1, 2012 to August 12, 2019:

- Open - the open price of bitcoin on a day
- High - the highest bitcoin price on a day
- Low - the lowest bitcoin price on a day
- Volume - the total amount of bitcoin transacted on a day
- Close - the close price of bitcoin on a day
- Exponential Moving Average (EMA) - weighted moving average that gives more weighting, or importance, to recent price data than the simple moving average (SMA) does
- Relative Strength Index (RSI) - a momentum indicator that measures the magnitude of recent price changes to evaluate overbought or oversold conditions in the price of bitcoin [5]. It is a good indicator of whether bitcoin price is in an uptrend or downtrend
- Stochastic oscillator (KDJ) - a momentum indicator comparing the bitcoin closing price on a day to a range of its prices over a certain period of time [6]. It is a good indicator of overbought and oversold conditions to predict bitcoin price turning point. K value (KDJK), D value (KDJD), and J value (KDJJ) are calculated in the project
- Moving Average Convergence Divergence (MACD) - a trend-following momentum indicator that shows the relationship between two moving averages

of bitcoin price [7]. It indicates whether the bullish or bearish movement in the bitcoin price is strengthening or weakening.

We select which technical indicators to use based on their correlations. We remove EMA which is highly correlated with the OHLC prices. Also, we delete KDJK and KDJD which are more correlated with MACD than KDJJ. As a result, the final input data have eight features and Figure 1 shows a sample of the data.

	open	high	low	volume	close	rsi
0	5.32	5.32	5.14	88.037281	5.29	100.000000
1	4.93	5.57	4.93	107.233260	5.57	100.000000
2	5.72	6.65	5.72	94.801829	6.65	100.000000
3	6.65	6.90	6.00	33.882747	6.00	69.027732
4	6.80	6.80	6.80	0.295858	6.80	78.124634
...	...	...	...	...	...	...
2770	11471.58	12145.42	11388.01	15480.455242	11971.57	66.509764
2771	11973.46	12061.10	11450.93	10144.462109	11983.43	66.609543
2772	11985.60	12040.00	11650.00	7336.914849	11859.32	64.418529
2773	11859.32	11976.68	11200.00	7447.248341	11273.20	55.083811
2774	11273.03	11589.73	11080.37	4361.390692	11528.73	57.979772

	kdjj	macd
0	111.661662	0.008993
1	114.959404	0.022972
2	114.762911	0.073060
3	82.862844	0.074002
4	93.098743	0.109573
...	...	...
2770	92.961670	256.679625
2771	92.723656	325.133147
2772	86.901939	365.159090
2773	57.687198	345.601079
2774	50.362643	346.723537

Figure 1 Sample of final input data to train the model

From the scatter matrix that we have made in the notebook, the distributions of ‘open’, ‘high’, ‘low’, ‘volume’, and ‘close’ features are right-skewed. Supervised learning algorithms can be sensitive to such distributions of values and can underperform if the range is not properly normalized. So we will apply log transformation to those features in the ‘Data Preprocessing’ section in the project before using them to train up the models.

2.2 Exploratory Visualization

Figure 2 shows the closing bitcoin price from Jan 1, 2012 to Aug 12, 2019. Bitcoin price was not increasing in most of the time between 2012 to 2016. However, its price rocketed up from USD 1000 to around 20000 during 2017 and then dropped back to around USD 3000 at the end of 2018. This sudden rise and drop of bitcoin price shows extreme volatility and the bitcoin price is hard to predict.

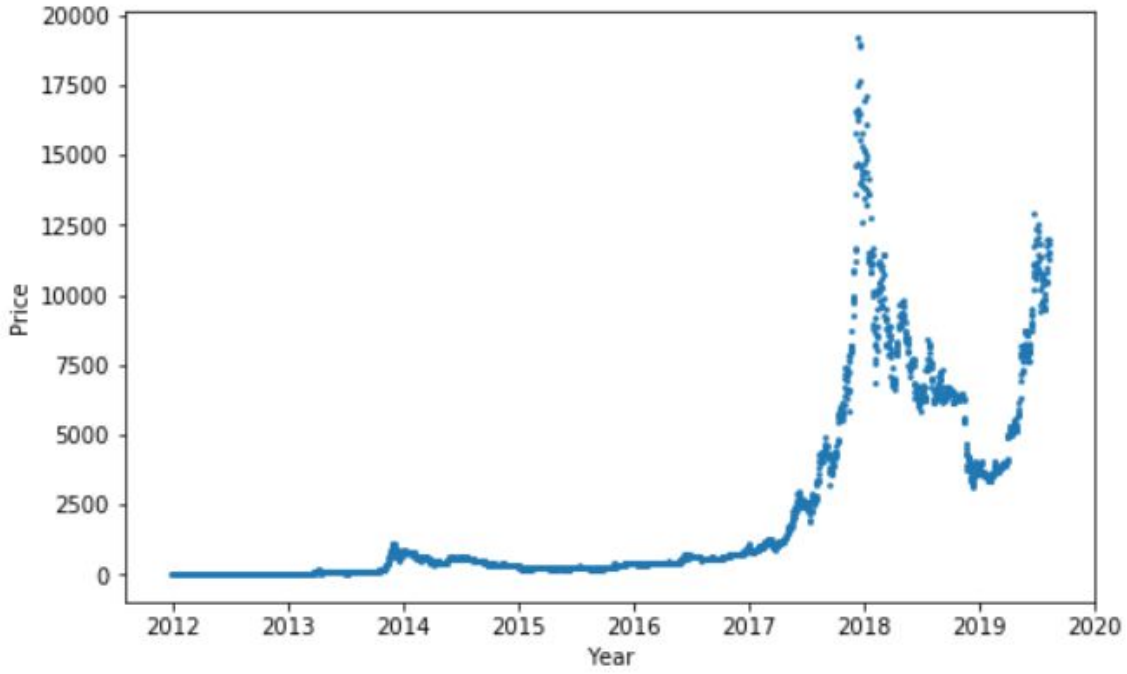

Figure 2 Bitcoin Price from Jan 2012 to Aug 2019

2.3 Algorithms and Techniques

We use three supervised learning algorithms to train up the models: Logistic Regression, Support Vector Machines (SVM), and Gradient Boosting. In Refinement, we adopt K-fold cross validation technique to fine-tune the model from each algorithm.

2.3.1 Logistic Regression

Logistic regression is typically seen as a robust baseline method for classification. This algorithm is fast in training large dataset and has linear classification boundary. Logistic regression is suitable in this project as our dataset is relatively large (over 2500 records) and our problem is a binary classification.

2.3.2 Support Vector Machines (SVM)

SVM is a discriminative classifier that generates a separating hyperplane. Error tolerance budget is included to make separating hyperplane robust in case of inseparable class data [8]. SVM can deal with high dimensional data and find the optimal separation hyperplane through custom kernels. This algorithm is suitable in this project as our dataset contains eight features.

2.3.3 Gradient Boosting

Gradient boosting is a machine learning algorithm which produces a strong prediction model in the form of an ensemble of weak prediction models, typically decision trees. The performance of gradient boosting model is very good on large dataset, so it is an ideal candidate for our algorithm choice.

2.3.4 K-Fold Cross Validation

We split the dataset into training, validation, and testing sets, and only the training set is used to train up the model. The model misses the information in the validation set and as a result, may lead to overfitting. K-fold cross validation allows the validation set being used to train up the model. In K-fold cross validation, training and validation data will be split into k buckets and the training process is reiterated k times. Each round, a bucket is used as a testing set and the others are used as the training set. The final model would be the average from each round. This technique is good to reduce overfitting and help to fine-tune our models.

2.4 Benchmark

From the dataset, we know that the number of days on which bitcoin price is going up is more than that of days on which bitcoin price is going down. Therefore, we set our benchmark model to be a naive guess which always predicts the bitcoin price will go up on the next day. Refer to the calculation in the notebook, the accuracy and F-0.5 score of the benchmark model is 0.5447 and 0.5993 respectively.

3. Methodology

3.1 Data Preprocessing

We label each bitcoin daily record as 1 if the closing price of bitcoin on the next day is higher than that of the daily record; otherwise label as 0. Furthermore, from the scatter matrix, the distribution of features ‘open’, ‘high’, ‘low’, ‘volume’, and ‘close’ are right-skewed. Since the very large or very small values do affect the performance of some learning algorithms, we apply logarithmic transformation to those features. Using a logarithmic transformation significantly reduces the range of values caused by outliers.

In addition to performing transformations on features that are highly skewed, we also use min-max scaling to normalize all features in the dataset. Applying a scaling to the data does not change the shape of each feature's distribution; however, normalization ensures that each feature is treated equally when applying supervised learners. Since all feature values are positive, we scale them between 0 and 1.

After the scaling is done, we split the dataset into training, validation, and testing sets. We set the training set including the first 84% of data records to make sure that the models can learn from the bitcoin price rising and falling trends in late 2017 and early 2018 respectively. Then, the validation set contains the next 8% of records and the testing set takes the remaining data.

3.2 Implementation

3.2.1 Regression Models with Rolling Forecast Validation

On the first day of this project implementation, we were to implement Vanilla LSTM to predict the bitcoin price on the next day using the historical price data as proposed in the Capstone Proposal submitted. And the ARIMA model acts as the benchmark model and R2 score as the evaluation metric. In the validation process during implementation, we adopted rolling forecast in which each time LSTM predicted only the price on the next day. Then the LSTM was re-trained with the truth price on that day and predicted the price of the day after it. Since LSTM needed to be re-trained in each iteration, it took three hours to finish the validation process though using only one neuron in the model run in GPU instance. The running time was too long for this project, so we changed from rolling forecast to predicting multi time steps as suggested by the mentor.

3.2.2 Regression Models with Multi TimeStep Validation

We changed to predict the prices in the next seven days during the validation process. The running time was a lot faster in this case; however, the performance of LSTM was

not satisfactory. I usually got a negative R2 score even though I tried different numbers of neurons, epochs, batches, and historical price data to train the model. The performance of LSTM was really unstable: it easily got overfitting in longer epochs and underfitting in shorter epochs. The best R2 score we got so far was around 0.37. As this problem was really challenging, the mentor suggested to change to binary classification problem.

3.2.3 Supervised Learning Models with Binary Classification

We switch from regression to classification problem in which a supervised learning model is trained up to predict whether the bitcoin price goes up or down on the next day. We decide to use Logistic Regression, SVM, and Gradient Boosting to train up the model and pick the one that performs the best. For each algorithm used, we record the accuracy and F-0.5 scores of validation set and the last 300 training records when different training set sizes are used: 10%, 50%, and 100% of the training set data. In addition, training and prediction times of those algorithms are marked down. Figure 3 shows the graphs of performance metrics on those three supervised learning models.

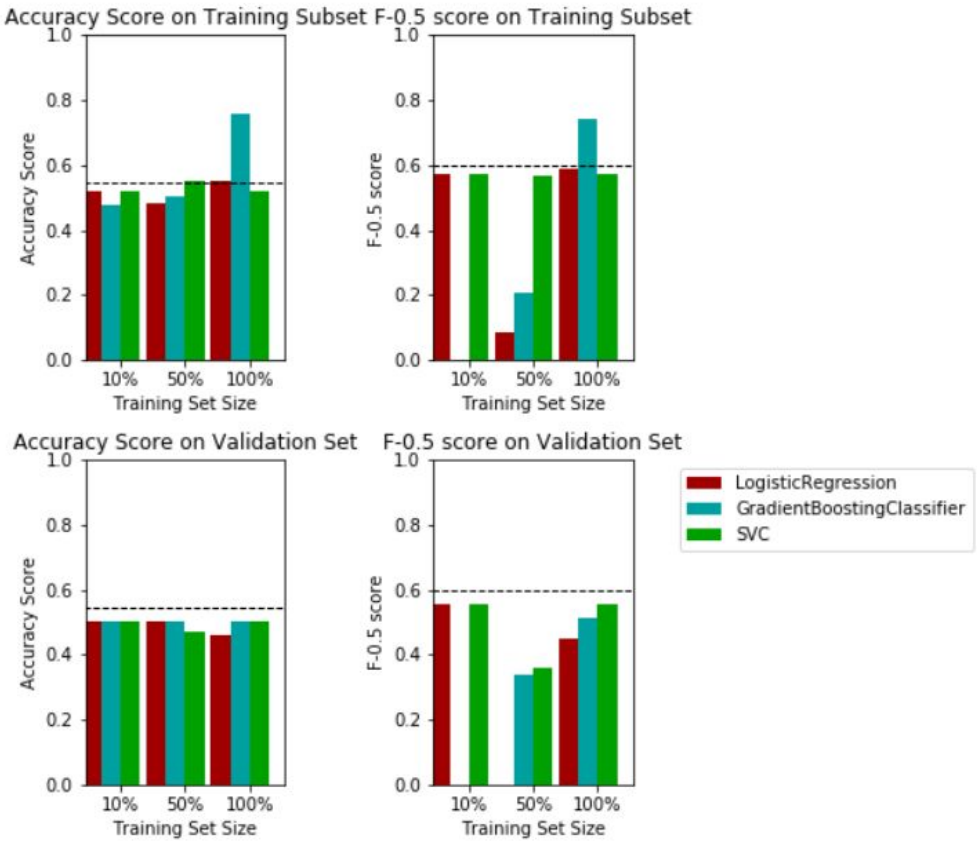

Figure 3 Performance Metrics on Three Supervised Learning Algorithms

None of the algorithm can beat the benchmark model on accuracy and F-0.5 score for validation set. Gradient Boosting surpasses the benchmark model for predicting 300 training set records when 100% of the training set data are used, which achieves around 0.8 on accuracy and F-0.5 score. However, it suffers overfitting a lot since those scores drop to 0.5 when predicting validation set. Since the performance of the models are not satisfactory, we fine-tune them to see if we can get a better performance.

3.3 Refinement

3.3.1 Refinement of SVM Classifier

We try to use different kernels, rbf and linear, with numbers of penalty parameter C to find out the best model. We use GridSearch and K-fold cross validation which we try K = 5 and 10. The optimal model for SVM has accuracy of 0.5856 and f-0.5 score of 0.6385 on the testing set.

3.3.2 Refinement of Logistic Regression Classifier

We use ‘saga’ as the solver instead of ‘liblinear’ which is the default setting since ‘saga’ can handle L1 and L2 regularization. We try to use different regularization strength to find out which setting is the optimal. The optimal model of Logistic Regression has accuracy of 0.5901 and f-0.5 score of 0.6415 on the testing set.

3.3.3 Refinement of Gradient Boosting Classifier

We use ‘exponential’ as the loss function and try different numbers of minimum sample leaf and split. We find out that a large number of boosting stage, which is 100 in default setting, leads to overfitting. We use smaller numbers and enhance performance. The optimal model of Gradient Boosting has accuracy of 0.6126 and f-0.5 score of 0.6598 on the testing set.

3.3.4 Comparison of Initial and Optimal Models

Gradient boosting produces the test model among three algorithms. We compare its initial and optimal models on the testing set. Figure 4 shows the comparison.

```
Accuracy score for the initial model: 0.5946
F-0.5 score for the initial model: 0.6489
Accuracy score for the optimal model: 0.6126
F-0.5 score for the optimal model: 0.6598
```

Figure 4 Comparison of Initial and Optimal Models

The optimal model surpasses the initial model on both the accuracy and F-0.5 scores.

4. Results

4.1 Model Evaluation and Validation

As shown in the notebook, the final model has the minimum sample leaf of 5 and the minimum sample split of 3. More samples in the leaf node tend to improve performance. And the optimal model only has 10 estimators which in this case, reduce overfitting. The optimal model has the accuracy of 0.6126 which is a good score in predicting bitcoin price. In the project of Alex Greaves, Benjamin Au, they built models to predict whether the bitcoin price one hour into the future would go up or down [8]. Their best model which was a 2-hidden-layer neural network had accuracy of 0.551. Our optimal model has better performance and it proves our model is robust.

4.2 Justification

```
Benchmark model on testing set: [Accuracy score: 0.5856, F-0.5 score: 0.6385]
Best Gradient Boosting model on testing set: [Accuracy score: 0.6126, F-0.5 score: 0.6598]
```

Figure 5 Comparison between Optimal and Benchmark Models

Figure 5 shows the comparison between the performance of the optimal and benchmark models. The optimal model has better scores on both accuracy and F-0.5 scores. We conclude that the final model performs better than the benchmark model. And F-0.5 score of 0.6598, which is roughly two-thirds, gives a certain message that when the model predicts the price will go up tomorrow, it likely will do. The final model has adequately solved the problem.

References

[1] Bitcoin -Wikipedia
https://en.wikipedia.org/wiki/Bitcoin

[2] Bitcoin History – Price since 2009 to 2019, BTC Charts – BitcoinWiki
https://en.bitcoinwiki.org/wiki/Bitcoin_history

[3] Virtual Currencies - Bitcoin Risk
Mark T. Williams
http://www.bu.edu/questrom/files/2014/10/Wlliams-World-Bank-10-21-2014.pdf

[4] Bitcoin Historical Data | Kaggle
https://www.kaggle.com/mczielinski/bitcoin-historical-data

[5] Relative Strength Index – RSI Definition & Calculation
https://www.investopedia.com/terms/r/rsi.asp

[6] Stochastic Oscillator Definition
https://www.investopedia.com/terms/s/stochasticoscillator.asp

[7] Moving Average Convergence Divergence – MACD Definition
https://www.investopedia.com/terms/m/macd.asp

[8] Using the bitcoin transaction graph to predict the price of bitcoin
Alex Greaves, Benjamin Au
https://pdfs.semanticscholar.org/a0ce/864663c100582805ffa88918910da89add47.pdf