Predictive Analysis Using The Bitcoin Minute Data

Aamodini Gupta,[1] Bill Zheng,[1] Marc Lee,[1] and Naveen Sathiyanathan[1]
STAT 430 (Fall 2018) - Group 4

(Dated: 30 October 2018)

Classical machine learning algorithms including Logistic Regression, Random Forest, and Gradient Boosting Machine were applied as well as the additional algorithms such as Extremely Randomized Trees and Stacked Ensemble methods. Purged k-fold cross validation and sequential bootstrap sampling were adopted for hyperparameter tuning and testing the generalization of the model. It was found that the Stacked Ensemble method performed the best among the algorithms tested in terms of predicting the direction of the bitcoin price on returns.

I. INTRODUCTION

Bitcoin is a cryptocurrency - a form of decentralized digital currency that can be sent directly from users to users on a peer-to-peer bitcoin network without going through any intermediate financial institutions. The central idea of Bitcoin was implemented by a group of pseudonymous people called Satoshi Nakamoto in 2009.

Transactions of bitcoins from account to account can be recognized globally in a matter of seconds and securely settled within an hour, regardless of users' geographical location or jurisdiction. Bitcoin also has its own supply and demand market and the price vibrates substantially throughout the time, with fluctuations sometimes caused by external factors.

Bitcoin's growing trade volume and its fluctuating nature have sparked our interest in its study. In this project, the effect of volume and close price on returns was tested. The suggested model was trained to predict the direction of bitcoin price for a specified window (i.e test if the price will hit the upper barrier or not) that is calculated by the target returns.

II. DATA DESCRIPTION

The dataset was collected from Kaggle. It records bitcoin prices from January 2012 to July 2018 with 1-minute intervals, comprising of 1,922,030 observations and 8 variables. The observations contain minute-to-minute updates of Open, High, Low, Close prices, Volume in BTC and indicated currency, and weighted bitcoin price. Favorably, there was no missing data, so that any imputation method was not implemented. The dataset was provided in the structured time bars format. A summary of the variables was shown as below:

- Timestamp: Timestamp in Unix time format (in minutes).
- Open: Bitcoin price in currency units at time period open.
- High: Highest bitcoin price in currency units during time period.
- Low: Lowest bitcoin price in currency units during time period.
- Close: Bitcoin price in currency units at time period close.
- Volume_(BTC): Volume of BTC transacted in time period.
- Volume_(Currency): Volume of currency transacted in time period.
- Weighted_Price (VWAP): Volume-weighted average price.

III. EXPLORATORY ANALYSIS

At the first glance of the dataset, it was evident that there was not much variability in the early parts of the data.

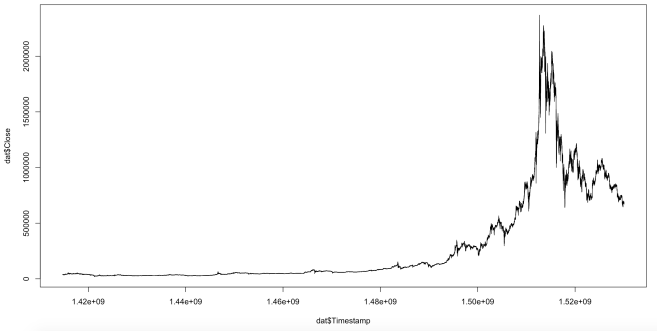

FIG. 1. Close Price vsTime.

In order to conduct analysis on the direction of price, it was important for the dataset to be highly volatile. Therefore, the focus of the project was shifted to the latter part of the dataset. The data was subsetted by taking the lowest close price, and considering all the timestamps after that drop. The updated dataset exhibited a lot more variability as evident in Figure 2.

One of the consequences of subsetting was that the data reduced from 1922030 observations to 511572 observations - this is still a substantial amount of information, so the analysis was conducted on this dataset. Note that there were no missing values in the dataset. It is also evident in Figure 2 that the dataset can be considered to be a non-stationary data because it does not tend to converge towards a certain level.

As described earlier, these observations were within even 1-minute intervals, therefore unit bars were used in order to

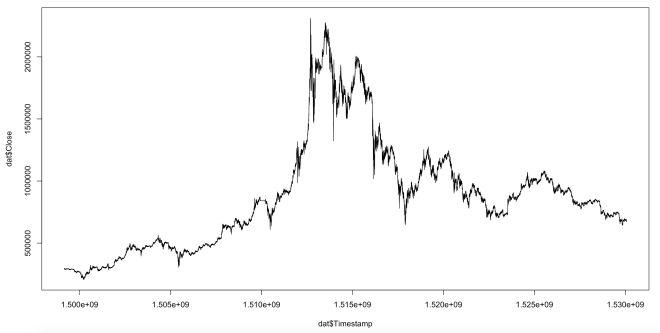

FIG. 2. Close Price vsTime.

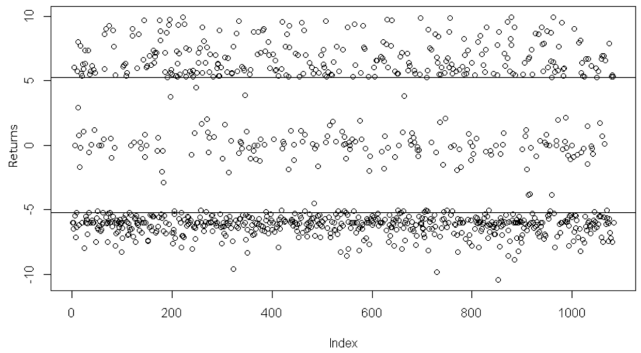

FIG. 3. All Target Returns on Fractionally Differentiated Series

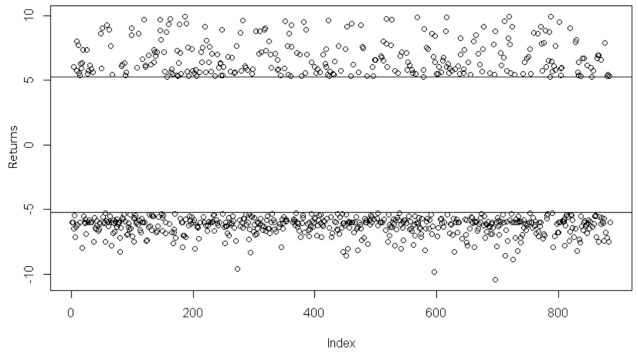

FIG. 4. Hits Barrier: Target Returns on Fractionally Differentiated Series

find more interesting patterns. In order to construct the unit bars, the bar_unit( ) function was created that takes in 2 inputs - the data and the unit - which are both specified by the user. For this report, the unit threshold was set to $7 * 10^7$ and the size of the unit bars came out to be 81,169

IV. FEATURE ENGINEERING

A. Fractionally Differentiated Features

In order to get meaningful information from the data, it needed to be transformed in a way that makes the data series stationary while preserving as much of the memory as possible. To do so, an input for the amount of differentiation that was required (the coefficient d).

The fixed-window method was implemented to get the optimal value for the coefficient d for which the fractionally differentiated series is stationary. The window was fixed at a value of 10. The smallest value of d that passed both the Efficient Unit Root Test and the KPSS Test (test for stationarity) was used. The amount of differentiation that was used for the purpose of this analysis was 0.96.

B. Labeling

Since the aim of this project is to predict direction of the prices, the triple barrier method of labelling was used. This path-dependent labelling technique included profit-taking and stop-loss of $[1, 0]$ that was applied to only consider the upper barrier. The label "1" was assigned if the upper bar is touched first, while "0" was assigned otherwise.

C. Feature Matrix

To create this feature matrix, a function was defined: get_feaMat( ). The function output the following columns:

- Target variable (binary)
- Volume
- Close Price
- tFeat
- tLabel

The CUSUM filter was used to filter and track the unit bars based on the events that occur at a defined threshold. The matrix takes into consideration the volume traded in the defined time windows(volume with CUSUM filters), fractionally differentiated (close) price, and other features extracted from time such as trend, seasonality, day of the week, hour of the day. Having these features in one place as a matrix provided a convenient way of storing important information that was needed to conduct further analysis in predicting the target returns.

V. METHODOLOGY AND ANALYSIS

Using the feature matrix and the labels created, the data was then sampled using the following two techniques:

A. Sequential Bootstrap

This technique was used to avoid the redundancies caused by sampling with replacement. The sequential bootstrap essentially makes draws according to a changing probability that accounts for this redundancy. This method takes into consideration the average uniqueness of an observation at a particular time and updates the probabilities so that the likelihood of picking a repeated value decreases after every draw.

B. Purged k-fold Cross Validation

Cross validation was employed to determine the accuracy and errors of the machine learning algorithm. The k-fold cross validation generally fails in financial analysis due to the fact that such observations cannot be assumed to be independent and identically distributed, and data leakage could take place if there was overlapping information between the training and testing datasets. One of the approaches to avoid this was to conduct a purged k-fold cross validation, which would remove any observations whose labels were overlapped in time between the training set and testing set. Some data from the training set that follows immediately after (embargo) was also excluded because of its high correlation with the testing set.

After the initial setup, different techniques were introduced to predict the target.

C. Ensemble Methods

Several ensemble methods were modeled in this project in order to obtain a better predictive performance. Random Forest was one of the ensemble methods tested, making it easier for predicting the target variable using all the features and indicate which features were the most important for the prediction. The parameter tuning to increase the accuracy was done in the later section.

Another method included in the analysis was the Stacked Ensemble. According to the h2o documentation about the algorithms, it stated that "Stacking, also called Super Learning or Stacked Regression, is a class of algorithms that involves training a second-level "meta-learner" to find the optimal combination of the base learners. Unlike bagging and boosting, the goal in stacking is to ensemble strong, diverse sets of learners together.".

Other methods tested were Gradient Boosting Machine (GBM), Logistic Regression, and Extremely Randomized Trees. Amongst these, the Logistic Regression was implemented for the sake of comparing the traditional machine learning algorithm to the ensemble methods.

D. Feature Importance Analysis

The feature importance analysis was conducted for the purpose of verifying which features played the significant role in predicting the target. Although the model and the function in the analysis did not require dropping insignificant features, this importance analysis was implemented to understand the effect of the Volume and the Closed Price on returns as indicated in the introduction.

Random Forest and Extremely Randomized Trees methods were used to produce the scaled variable importances. In addition to the scaled variable importances, the coefficient of model outputs of the Random Forest and the Extremely Randomized Trees was generated from the Stacked Ensemble method.

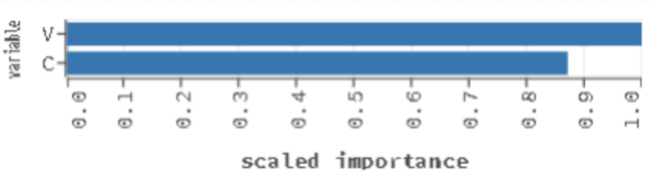

FIG. 5. Scaled feature importance based on Random Forest

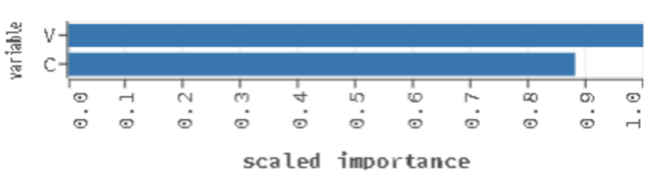

FIG. 6. Scaled feature importance based on Extremely Randomized Trees

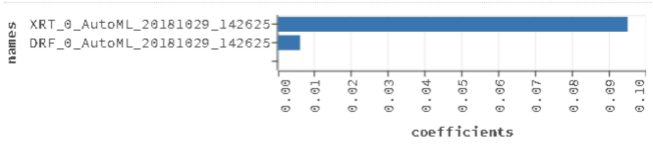

FIG. 7. Coefficient of model outputs in Stacked Ensemble

E. Hyper-Parameter Tuning

Randomized grid search method was applied on values h and trgt for fracDiff and meta labelling respectively for tuning the parameters to get optimal results, where h was a threshold for the CUSUM filter and trgt was a threshold used to return "0" or "1" in meta labeling. Models were built on folds obtained from sequential bootstrap and purged k-fold cross-validation. The average performance of the models on these

folds was compared to find the combination of parameters that maximized the CV performance in accuracy. The idea was to randomize the search with purged k-fold CV and to sample parameters from a distribution. With the correct input and commands, it came up with the optimal returns and the window size using the Grid search. The final parameters chosen by the randomized grid search method were $h = 228$ and $trgt = 5.25$.

VI. RESULTS AND CONCLUSIONS

Train set predictions were calculated using the best model chosen on the validation set using AUC. F1 was calculated on the train set predictions and the threshold with largest F1 score on train set was chosen for predicting classes on the train, validation and test sets. Using this threshold, the following results were observed:

Dataset	Model	Metric	Value
Test	Logistic Regression	AUC	0.504
Test	Logistic Regression	Accuracy	0.535
Test	Random Forest	AUC	0.797
Test	Random Forest	Accuracy	0.73
Test	Stacked Ensemble	AUC	0.881
Test	Stacked Ensemble	Accuracy	0.779

Compared to the Logistic Regression (traditional ML methods), the Stacked Ensemble method performed a lot better in predicting the direction of the price. It was observed that the no-information rate in test set was 0.544. Logistic regression performed close to random guess with an accuracy of 0.535 and AUC of 0.504. Random forest performed better than logistic regression with an accuracy of 0.73 and AUC of 0.797. However, the stacked ensemble of extremely randomized trees and random forest (excluding gradient boosting machine, because its prediction was correlated with that of random forest) outperformed these models with an accuracy of 0.779. Refer to the figures 8, 9, 10, 11 to see the lift and ROC curves for the training and validation sets.

VII. TEXT ANALYTICS

It is believed that news related to bitcoins drastically affects the price of bitcoins. Also, there are multiple sources of news articles about bitcoin prices because of the hype and myths around bitcoin. During 2017-18 there were significant bitcoin related events that led to positive and (mostly) negative news about bitcoins. We performed an independent analysis to study the effect of words used in news articles on the returns of bitcoins on the next day. Text data was collected manually from 50 different news articles. We assume that short term effects of news are significant and long term effects are negligible.

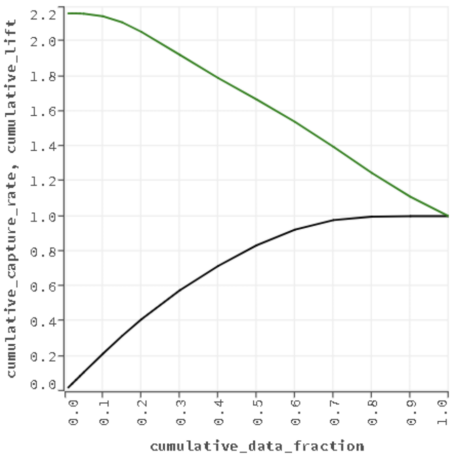

FIG. 8. Training Set Lift Curve

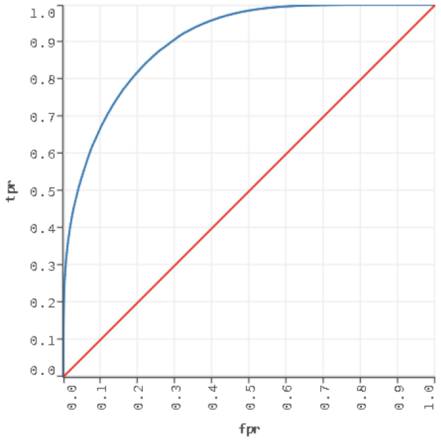

FIG. 9. Training Set ROC Curve

A. Text processing

50 randomly chosen news (38 distinct days between 15th July 2017 and 26th June 2018) articles related to bitcoins were considered and their dates of publication were recorded along with the text. Standard text processing steps (removal of stop words, punctuations and numbers, performing stemming) were applied and the term document matrix was aggregated to date level. Words that occurred in less than 3 documents were dropped. Term-frequency inverse-document-frequency (TF-IDF) transformation with document length normalization was applied on the data set. This led to 38 observations of 1152 features. This data set was merged with the returns of the next day. Finally, all variables were centered and scaled.

B. Modeling

Random forest with large number of trees was applied to estimate the relative importance of variables. Simultaneously,

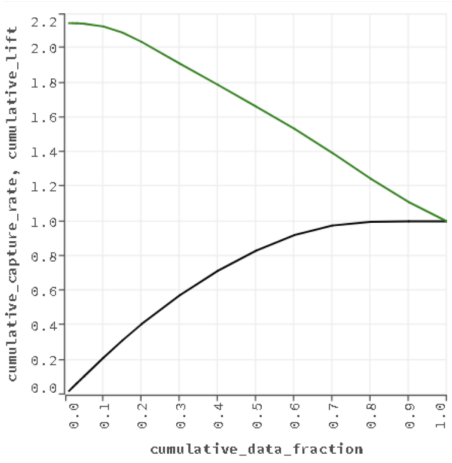

FIG. 10. Validation Set Lift Curve

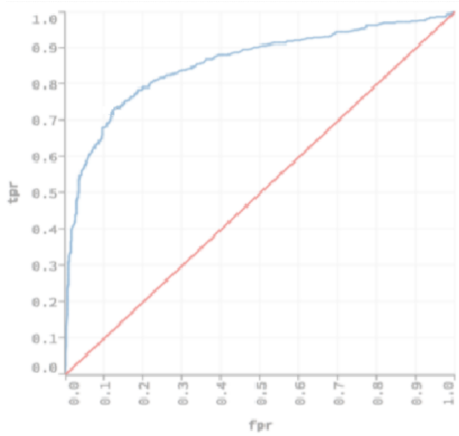

FIG. 11. Cross Validation ROC Curve

a l_1 regularized (lasso) linear regression model was built. This forces a sparse coefficient matrix at higher penalties. Results of both models were combined to understand the magnitude and direction of effect of words on bitcoin returns. Summary of important words identified by the model:

Positive		Negative	
Word	**Lasso Coefficient**	**Word**	**Lasso Coefficient**
danger	-0.0155	check	0.0150
ultim	-0.0154	stabil	0.0118
blow	-0.0086	less	0.0098
stupid	-0.0078	japanes	0.0034
live	-0.0072	solut	0.0027

VIII. SCOPE FOR IMPROVEMENT

The data set provided was already summarized in the form of time bars. It did not contain tick data. Therefore, an approximation was used to compute unit bars, which were used in the downstream analysis.

Text predictors were proved to have significant effect on the returns of next day. We were unable to add these features to predict the binary label because of the high model run-time with the reduced feature set.

Text analysis was performed without considering the context of the words used in articles. Only bag-of-words features were used. This can be improved by using better text representation models such as word2vec.

Larger number of uncorrelated regression models such as naive Bayes, linear discriminant analysis, etc. could be fit to the data. This will enrich the stacked ensemble model, leading to further increase in classification accuracy.

APPENDIX

- The Kaggle Data Source https://www.kaggle.com/mczielinski/bitcoin-historical-data
- R code for the overall flow.

```
hVec <- seq(10, 300,length = 5)
trgtVec <- seq(0.5, 10, length = 5)
all_test_auc <- c()
set.seed(1)
k <- 10
gam <- 0.01
for(ih in 1:5) {
  for(jtrgt in 1:5) {
    print(paste("Running h=", hVec[ih], ", trgt=", trgtVec[jtrgt]))
    i_CUSUM <- istar_CUSUM(C_fracD_all, h=hVec[ih])
    i_CUSUM_train <- i_CUSUM[i_CUSUM <= length(C_fracD)]
    ## Feature matrix ##
    fMat0 <- get_fMat0(C_fracD_all, h, C_fracD, V_fracD_all, i_CUSUM, i_CUSUM_train)

    #meta labeling based on
    #the fracDiffed closed price: C_fracD_all

    i_CUSUM_used <- i_CUSUM
    n_Event_used <- length(i_CUSUM_used)
    events <- data.frame(t0 = i_CUSUM_used + 1, t1 = i_CUSUM_used + 200, trgt = trgtVec[jtrgt], side = rep(1, n_Event_used))
    ptSl <- c(1, 1)
    out0 <- get_feaMat(C_fracD_all, events, ptSl, i_CUSUM_used)

    out <- subset(out0, ret >= events$trgt*ptSl[1])
    fMat <- fMat0
    Y_train <- rep(0, n_Event_used)
    Y_train[out0$ret >= events$trgt*ptSl[1]] <- 1
    # 1 for positive resturns >= trgt*ptSl[1]
    X_train <- data.frame(fMat)
    allSet <- data.frame(Y = as.factor(Y_train), C = as.numeric(X_train$C), V = as.numeric(X_train$V), tFea = out0$tFea, tLabel = out0$tLabel)
    all_tbl <- table(allSet$Y)
    print(paste0("The combination of ", hVec[ih], " ", trgtVec[jtrgt], " is decently balanced"))
    i_CUSUM_train <- intersect(i_CUSUM_train, i_CUSUM_used)
    trainSet <- allSet[1:length(i_CUSUM_train), ]
    testSet <- allSet[(length(i_CUSUM_train) + 1):nrow(allSet), ]
    train_tbl <- table(trainSet$Y)
    test_tbl <- table(testSet$Y)
    if(any(all_tbl/sum(all_tbl) > 0.6) | any(train_tbl/sum(train_tbl) > 0.6) | any(test_tbl/sum(test_tbl) > 0.6)) {
      next
    } else {
      i_j_auc <- return_auc(cc, C_fracD_all, hVec[ih], C_fracD, V_fracD_all, seqboot = F, purged_kfold = T, ih, jtrgt, k, gam, allSet, max_time = 120)
      names(i_j_auc) <- paste0(hVec[ih], "_", trgtVec[jtrgt])
      all_test_auc <- c(all_test_auc, i_j_auc)
    }
  }
}
```

- R code for obtaining the AUC value.

```
trainSet <- as.h2o(CVobj[[i]]$trainSet)
valSet <- as.h2o(CVobj[[i]]$testSet)
testSet1 <- as.h2o(testSet)
print(colnames(trainSet))
bag <- h2o.automl(x = 2:3, y = 1, training_frame = trainSet, validation_frame = valSet, seed = 1, max_runtime_secs = max_time, keep_cross_validation_models = F, exclude_algos = "DeepLearning")
print(bag)
roc_prob <- predict(bag, newdata=testSet1, type="prob")
pred <- prediction(as.vector(roc_prob[,3]),as.vector(testSet$Y))
auc <- performance(pred, measure = "auc")@y.values[[1]]
print(auc)
roc_prob1 <- predict(bag, newdata = trainSet, type = "prob")
pred1 <- prediction(as.vector(roc_prob1[,3]),as.vector(trainSet$Y))
f <- performance(pred1,"f")
thr <- f@x.values[[1]][which.max(f@y.values[[1]])]
print(thr)
output <- as.integer(as.vector(roc_prob[,3]) > thr)
tbl <- table(output, testSet$Y)
print(tbl)
acc <- (tbl[1, 1] + tbl[2, 2])/sum(tbl)
accs <- c(accs, acc)
aucs <- c(aucs, auc)
            if(i != k)
              h2o.removeAll()
          }, error = function(e) NULL)
        }
```