Big Data Analytics Final Report
Trading Using Nonparametric Time Series Classification Models

Authors Yuan Cai, Bowen Wang, Junchao Zhang
Electrical Engineering and Computer Science
Columbia University
yc2870@columiba.edu,
bw2459@columbia.edu,
jz2607@columbia.edu

***Abstract*—Using classification models to give a prediction on whether the asset price will go up or down by incorporating unstructured data stream. In this project, Logistic Regression and SVM were implemented on Bitcoin price prediction. Bitcoin price and social network data such as Twitter were used as predict variables. The result was acceptable and the project was scalable.**

***Keywords*-Logistic Regression, SVM, Bitcoin, Twitter, price prediction**

I. Introduction

Using classification models to give a prediction on whether the asset price will go up or down by incorporating unstructured data stream. Bitcoin was chosen as the subject in this project since its constantly good liquidity and high price volatility. In this case, trained classification models can easily capture the trend. The Bitcoin price at December 10, 2014 is 342.02 US Dollar per Bitcoin. The trend showed that the Bitcoin price was flat and low until the 2013. Then the price goes all the way up to almost 1200 US Dollar per Bitcoin then started dropping. The trend here has dramatic fluctuation, which met the needs of this project. Except the Bitcoin price, the social network, Twitter, was also taken into account for prediction since Twitter's hash tag feature and the great number of active users and tweets.

The major problems should be solved in this project including dataset collection for both Bitcoin price and Twitter, data processing and structuring and classification algorithms implementation. First, the project required a large amount of data for models training and testing. Then those data should be organized into format, which can be used as predict variables for classification algorithm function. What is more, since the raw data didn't have attribute that can indicate the rise and fall of price, the project needed to generate the target value for each record and used the target value to evaluate the model. After the data was ready, the project requires appropriate implementation of classification algorithms and useful output. There were several machine learning open source libraries that were suitable for this project.

II. Related Works

The previous homework assignment of EECSE6893 Big Data Analytics had covered some classification problem related to 20 newsgroup data and Wikipedia. The previous assignment was using Mahout, a machine learning open source library, to implement classification. But both dataset were text data. In this project, the data sets were all numeric data saved in CSV file. Even thought the previous assignment did not perform the logistic regression, the general workflow was similar in a typical classification project [1]:

1. Training the model. In this project, the target variable and predictor variables should be defined. The historical data should be collected. Then select a learning algorithm and use the learning algorithm to train the model.

2. Evaluate the model. It requires to run test data and adjust the input (use different predictor variables, different learning algorithms, or both)

3. Using the model in production. Input the new examples to estimate unknown target values and retrain the model as needed.

This project also followed the above workflow. The more specific details will be illustrated in the following sections.

III. System Overview

The project composed by two parts: Data processor and price predictor. Pleaser refer to flow chart Figure 1.

The Bitcoin price dataset was from Bitfnex, an Internet trading platform for Bitcoin (URL: https://www.bitfinex.com/). The dataset had Bitcoin?s price and volume per second form 3/31/2013, 6:07:48 PM EST to 3/31/2014, 6:07:48 PM EST. In this dataset, there were 3,100,790 entries. Each entry contains 3 fields: timestamp, price at this second and the amount of

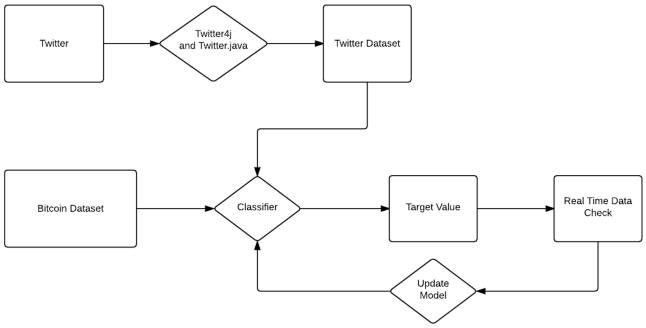

Figure 1. Flow Chart

```
Twitter twitter = new TwitterFactory().getInstance();
try {
    Query query = new Query(args[0]).since("2013-3-30");
    QueryResult result;
    int count = 0;
    do {
        result = twitter.search(query);
        List<Status> tweets = result.getTweets();
        for (Status tweet : tweets) {
            String name = tweet.getUser().getScreenName();
            String content = tweet.getText();
            Date createDate = tweet.getCreatedAt();
            int number = tweet.getUser().getFriendsCount();
            //System.out.println("@" + tweet.getUser().getScreenName() + " - " + tweet.getText());
            returnList.add(new TweetPack(name,content,number,createDate));
            //System.out.println("The friend count is "+tweet.getUser().getFriendsCount());
        }
        count++;
    } while (((query = result.nextQuery()) != null)&&(count < 100));
    System.out.println("The number of records is "+Integer.toString(returnList.size()));
} catch (TwitterException te) {
    te.printStackTrace();
    System.out.println("Failed to search tweets: " + te.getMessage());
    System.exit(-1);
}
```

Figure 2. twitter

transaction at this second.

There was no available free dataset for Twitter online. There were some site could provided the dataset with access fee 499 per month which was beyond the project?s budget. Instead using the dataset, the project used a Twitter API Twitter4j[1] to query tweets with specific hashtag and keywords, Bitcoin. By using the Twitter4j, it would able to query tweets in real time. Each returned data entry contains several available attributes. In this project, it used 4 fields: the username of the user who tweets this tweet, the tweet?s content, the number of friends of this user and timestamp. One query could return all tweets in a specific time. This project used the sum of number of tweets in 1 minute as additional one predictor variable. All these retrieved data could be stored into a CSV file as the input for classifier. But there was one deficiency for Twitter APIs. Twitter had rate limit and per-user limit, which means each individual can only access a fixed number of tweets per rate limit window and per day. This deficiency could not be fixed from the developer side and it would need the help from Twitter.

IV. Using Logistic Regression

The Twitter input will be used as predictor variable for classifier. The Twitter has an asymmetric relationship, which means that a follower of the user does not need to be followed by this user. If it just simply used the number of follower would the result would not be accurate. There are a lot of inactive accounts, which should be eliminated. In this case, the project used the number of friends instead. Friends mean that both users are following each other. To show the popularity of Bitcoin on Twitter, it would use the sum of number of tweets in 1 minute as additional one predictor variable. To weight different tweets so the result could be more representative, each tweet will be multiplied by the user?s friend number which has unit as 1000. The code to query and collect Twitter is shown in Figure 2

V. Using Logistic Regression

In statistics, logistic regression, or logit regression, or logit model,[2] is a type of probabilistic statistical classification model.It is also used to predict a binary response from a binary predictor, used for predicting the outcome of a categorical dependent variable (i.e., a class label) based on one or more predictor variables (features). That is, it is used in estimating the parameters of a qualitative response model. The probabilities describing the possible outcomes of a single trial are modeled, as a function of the explanatory (predictor) variables, using a logistic function. Frequently (and subsequently in this article) ”logistic regression” is used to refer specifically to the problem in which the dependent variable is binary, that is, the number of available categories is two, while problems with more than two categories are referred to as multinomial logistic regression or, if the multiple categories are ordered, as ordered logistic regression.

In order to classify whether asset price is going up or down, the first natural choice is using logistic regression. Logistic regression comes with the following definition. We want to use n features to predict dependent binary variable y, we can build a logistic regression model as

$$F(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}} \tag{1}$$

, where t is a linear function of predictor variables x. For example, one can write

$$t = \beta^T x = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n \tag{2}$$

The output variable $F(t) = P(y = 1|X)$ and thus $P(y = 0|X) = 1 - F(t)$.

In this project, we let x be log return of the asset of previous days and let $y = 1$ if it has a upward moving trend and $y = 0$ otherwise. Then we select a few latest log returns of current time as input features. We can train the models using the historical return data for the past one year or more and use the trained the model to make prediction. The code segment used to generate predictor variables and target values are shown in Figure 3.

```
N = n_training_sample_size
training_data = np.zeros((N,11))
print training_data.shape
for i in xrange(N):
    for j in xrange(10):
        training_data[i,j] = my_data_cut[i-j+10,1]/my_data_cut[i-j+9,1]
    if (my_data_cut[i+11,1] > my_data_cut[i+10, 1]):
        training_data[i,10] = 1

N = n_testing_sample_size
testing_target = np.zeros((N,1))-1
testing_data = np.zeros((N,10))
reset = n_training_sample_size
print reset
for i in xrange(N):
    for j in xrange(10):
        testing_data[i,j] = my_data_cut[reset+i-j+10,1]/my_data_cut[reset+i-j+9,1]
    if (my_data_cut[i+reset+11,1] > my_data_cut[i+reset+10, 1]):
        testing_target[i,0] = 1
    else:
        testing_target[i,0] = 0
```

Figure 3. Code Segment of Logistic Regression

VI. Using Support Vector Machine

In machine learning, support vector machines (SVMs, also support vector networks[3]) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

In our application, we have already applied logistic regression to use the probability to predict the up and down of the price of the bitcoin. In order to increase the accuracy of the model, we tried to use SVM model to do the improvement. By using SVM, we can find the optimal hyperplane for linearly separable examples. For non-linearly separable data, we can transform the original data by using a kernel function. See Figure 4.

In order to use the model, we have to do some operation on the data we have collected. The data we collected about the price of the bitcoin is selected per second. The data we collected from Twitter about the heat rate of bitcoin is

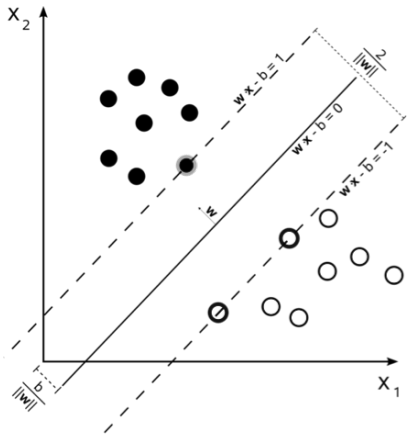

Figure 4. Support Vector Machine

collected per minute. So, we have to preprocess the data of the price. Also, after finishing the step, we still have to compare the price minute by minute in order to shorten the array and match the data of twitter. The code segment for using different kernel for SVM is shown in Figure 5.

```
#training
xx, yy = np.meshgrid(np.linspace(0, 4, 500),
                     np.linspace(0, 4, 500))
np.random.seed(0)

#clf=svm.SVC(kernel='linear',C=1000)
#clf=svm.SVC(kernel='poly',degree=7,C=5)
clf=svm.SVC(kernel='rbf',gamma=10,C=2)
#clf=svm.LinearSVC()
sss = cross_validation.StratifiedShuffleSplit(label, n_iter=3, test_size=0.4, random_state=0)
for train_index, test_index in sss:
    X_train, X_test = x[train_index], x[test_index]
    y_train, y_test = label[train_index], label[test_index]
print clf.fit(X_train,y_train)
scores = cross_validation.cross_val_score(clf, X_test, y_test, cv=5, score_func=None)
print scores
#svcsvm.SVC(kernel='linear')
#print svc.fit(x,label)
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.imshow(Z, interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()), aspect='auto',
           origin='lower', cmap=plt.cm.PuOr_r)
contours = plt.contour(xx, yy, Z, levels=[0], linewidths=2,
                       linetypes='--')
plt.scatter(a,b, s=30, c=label, cmap=plt.cm.Paired)
plt.xticks(())
plt.yticks(())
plt.axis([0, 4, 0, 4])
plt.show()
```

Figure 5. Using different kernel for SVM

VII. Software Package Description

Everything in this project will be open sourced. Several open source libraries and API were used in this project. Due to the developing consideration and the advantage of open source, the team decided that the whole project would be open sourced. scikit-learn (formerly scikits.learn) is an open source machine learning library for the Python programming language.[4] It features various classification, regression and clustering algorithms including support vector machines, logistic regression, naive Bayes, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Since this project?s emphasis was on algorithm and result, the UI is extremely simple. The program does not require any input or command from user. So the user can simply start running the given scripts and the result will be updated constantly based on real time data. There were three scripts required during the execution. The TwitterTry.java was used to query Twitter data. The logistic.py was used to process the input predictor variables and run logistic regression. The SVM.py was used to process the input data and run SVM algorithm.

VIII. Experiment Results

Result For Logistic Regression

In this project, we apply this approach to bitcoin prices with duration as frequent as 1 second. The training data set contains data of a full year of 2013. Here, we simply make $n = 10$. After training, we have interception and coefficient as $\beta_0 = -0.54051457$ and $\beta = [0.00058511, -0.00606645, -0.0007854,$
$0.00573074, 0.00079103, -0.00056447,$
$-0.00066931, 0.005007, 0.00090641, 0.00383245]$.
The in training error is 32.38%, and out of sample error is 34.58.

To visualize the result, the Figure 6 shows an example of log returns of a time period of 250 seconds in the testing data set. We use yellow dots to mark all the model predictions of the asset price going up. You can see most of the correct predictions give us the maximum return, because most of the yellow dots are above the zero line.

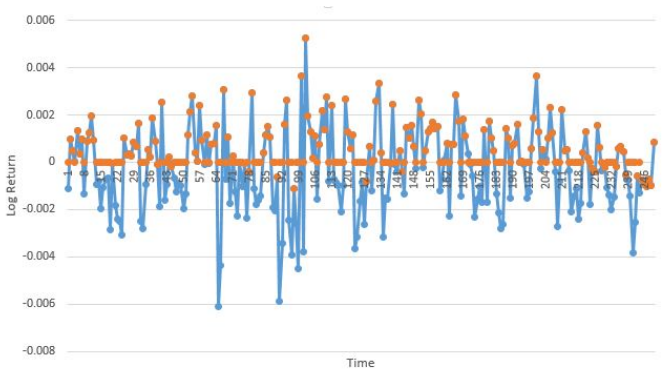

Figure 6. Sample prediction of asset return

Result for SVM

After processing the data, we can insert the data of the price and the twitter into the SVM model. We can use the model of SVM in SKlearn with convenience.

We have tried several models in order to fit the data. The optimal model should be linear because the price should be proportional to the twitter data. Presumably, if the search heat is high, it means the popularity of bitcoins is high. However, after training using linear model, we found that we cannot draw a good line between up and down of the price. The graph is shown as Figure 7.

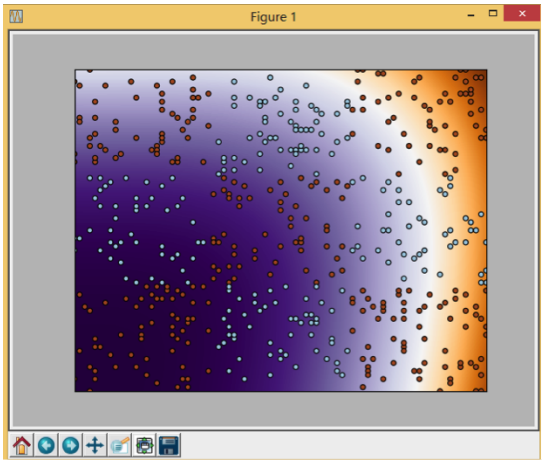

Figure 7. Data Distribution

For the reason that the number of up and down of the price are different, the predication accuracy is not 50%. If the number of up is larger, the probability of up should be larger. So, the model cannot be used accurately. Then we tried other two models which is polynomial and RBF, whose kernels are more complicated than linear. RBF is one of the best kernel for SVM. We adjust the value of C and gamma in order to make the model fit the data better. After separating the data into training set and testing set (4:1), we run the code and get the result by using cross_validation.

$$P(\text{Correct Predictions}) = [0.585, 0.585, 0.6, 0.589, 0.589]$$

If the value of gamma is not suitable for the data, the accuracy of the testing set is low. After adjusting the parameters, the result is a little better.

However, we still cannot ensure the relationship between the search heat of bitcoins and the price. What we want to do next is to use ensemble method to combine some useful models in order to get an optimal model for our data and to find some other data besides twitter data.

IX. Conclusion

This project provides an interface that utilizes machine learning models to predict asset price changes. Our experiments on logistic regression model and SVM are successful in giving correct prediction on whether the

price will go up or down in the next iteration. We also incorporated unstructured social media data, such as Twitter as one feature in our classification model. Each team member contributed on data collection and classification models to this framework. The job were divided evenly and every team member did a great job.

In future, we can expand our research on other machine learning algorithms and measure the performance of each and try to improve on it.

ACKNOWLEDGMENT

The team would like to thanks Professor Ching-Yung Lin and all TAs of EECS E6893: Big Data Analytics.

APPENDIX

REFERENCES

[1] CY. Lin, *Big Data Analytics Algorithms Clustering & Classification* Columbia University, 2014.

[2] Yusuke Yamamoto, *Twitter4j* http://twitter4j.org.

[3] D. A. Freedman, *Statistical Models: Theory and Practice* Cambridge University Press, 2005.

[4] C. C. Vapnik, *Support-vector networks* Machine Learning 20, 1995.

[5] A. Mller, *scikit-learn 0.15.1* Python Package Index, 2009.