Prediction of Stock Prices Using Markov Chain Monte Carlo

Mochammad Hariadi
Department of Computer Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
mochar@te.its.ac.id

Alfin Alim Muhammad
Department of Computer Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
alfin.alim14@mhs.te.its.ac.id

Supeno Mardi Susiki Nugroho
Department of Computer Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
mardi@te.its.ac.id

***Abstract*—Financial sector investment is an activity that attract a lot of public interest. One of them is investing funds in purchase company's shares. Stocks are proof of ownership of a company or business entity. Stocks are an attractive investment and quite challenging because they can provide large profits for investors if they predict correctly. Basically peoples buy stocks for long-term investment to get profit from dividend, but there are also investors who want to get benefit from buying and selling stock prices in the short-term periods. Predicting of stock prices become an attractive and challenges because it can be easy and hard based on fundamental and technical analysis. Apply Bayesian inference to create prediction model and Markov Chain Monte Carlo (MCMC) to generate predicted data. By mathematically predicting stock prices become more challenges, and spending much time to compute. But with parallel computing on Apache Spark requires less time.**

***Index Terms*—Stock, Bayesian Inference, Bayesian Statistics, MCMC, Apache Spark**

I. INTORDUCTION

The definition Stock is the participan of person or bussiness entity in company or limited company. Stock become an attractive and much chosen by investors, because its provides an attractive and challenges level of profit gain. There are two way analysis of stock i.e. fundamental analysis and technical analysis. Fundamental analysis is analysis by whole company, both product analysis, financial analysis and marketing analysis. Most people use fundamental analysis to choose which stock they want to invest. Technical analysis can be interpreted as an analysis of patterns of price movement in the past to predict the future prices using historical data.

Stock prices prediction has attracted extentive attention from scholars in the field of deep learning. Some of them applying complex neural network [1] base on their model analysis. Some also use sentimental analysis through social media data [2]. Basically applying machine learning to predict stock prices need complex matematics analysis to make model computation. Huge computation spending much time to compute on single node computer. Also applying machine learning to analyze stock prices. Divide computation to few computers by apache spark make it faster.

Markov Chain Monte Carlo or MCMC is methods for sampling from posterior ditribution. MCMC methods make it simple by making a model base on probability distribution, then applying bayesisan inference. Using MCMC we want to analize stock prices by technical analysis to predict the probability of future stock prices running on apache spark multinode cluster.

The purpose of this research is to implement Markov Chain Monte Carlo on Apache Spark to predict future stock prices. Parallel computing in Apache Spark multinode cluster reduces running time than using single node computer. This research is expected to accelerate the computing proccess and assist investor in making decision when investing in a company.

II. BASIC THEORY

A. Data

Data is an information, especially facts or numbers, collected to be examined and considered and used to help decision-making, or information in an electronic form that can be stored and used by a computer [3]. Data which is not processed will not be useful because of the tendency of data that is stored has no structure and relations between one another [4], the processes by which data are collected, transmitted, and stored are various, storage methods diverse, and diverse data types, so it must be processed such that people can draw conclusions based on that data. Data is classified into two, structured data and unstructured data. Structured data is data that depends on data model and occupies a fixed field in record. Structured data usually stored in a database or spreadsheet. Structured data needs to be a data model first to define the data type(number or text), and other restriction(minimum or maximul number value, allowed characters). Unstructured data is data that does not have model data. This type of data usually contains a lot of text, or multimedia content that cause irregularity and ambiguity in the dataset. Examples of unstructured data is word processor, videos and images.

B. Apache Spark

Apache Spark is a software specifically to procees large-scale of data with distributed cluster technology. Using apache spark to combine multiple computer make computing ability faster. Data processed in memory on apache spark, thats make it faster. Apache spark has two main abstraction,

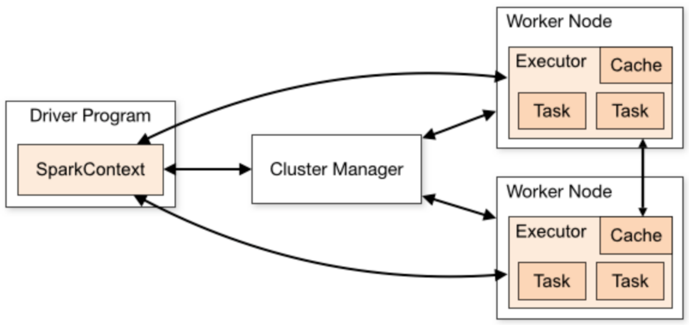


Fig. 1. Apache Spark Architechtures

- Resiliant Distributed Dataset or RDD is data element which cannot to be changed but can duplicated to other forms.
- Directed Acrylic Graph or DAG is part of scheduling in apache spark that contain data flow.

C. Stock

Stock can be defined as a sign of capital participation of a person or party (business entity) in a company or limited liability company [5]. Distribute shares is one of the company's choices in funding the company. Stock gains can be obtained from dividends and capital gains. Dividends are profit sharing from a company. To pay a dividend, a person must hold shares for a relatively long time until ownership of the shares is recognized to get dividends at the General Meeting of Shareholders (GMS). Capital Gain is the difference between the purchase price and the selling price due to stock trading activities in the secondary market. Stock returns calculated by

$$R_t = \ln\left(\frac{S_t}{S_{t-1}}\right) \tag{1}$$

where S_t is stock price at t time.

D. Bayessian Inference

In Bayesian Inference, probabilities are interpreted as subjective degrees of belief [6]. The goals is to state and analyze your beliefs. Model of Bayesian Inference is derived from Bayes' Theorem below :

$$p(\theta|x) = \frac{p(\theta|x)p(\theta)}{p(x)} \tag{2}$$

by removing $p(x)$ the equation become

$$p(\theta|x) \propto p(\theta|x)p(\theta) \tag{3}$$

Components of Bayesian inference are :

1) $p(\theta)$ is the set of prior distribution for parameter set θ, and uses as a means of quantifying uncertainty about θ before taking the data.
2) $p(x|\theta)$ is the likelihood function, which all variables are related in a full probability model.
3) $p(\theta|x)$ is joint posterior distribution that express uncertaintty about parameter set θ after taking tboth prior and data.

Bayesian Inference procedure carried out in the following way :

1) First we choose probability density $p(\theta)$, that we called prior distribution for our beliefs about a parameter θ before we see any data.
2) Second we choose a statistical model $p(x|\theta)$ that reflects our beliefs abaout x given θ, that we called likelihood.
3) After observing data $D_n = X_1, ..., X_n$, we update our beliefs and calculate the posterior distribution $p(\theta|D_n)$

The parameters of a prior distribution are called hyperparameter, to distinguish them from the parameter θ of the model. Prior is multiplied by the likelihood funtion and then normalized to estimate the posterior probability distribution, which is conditional distribution of θ given the data x.

E. Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is an algorithm for sampling from a distribution by constructing a Markov chain in a stationary particular distribution. The generation is done randomly, but there is a mechanism of acceptance or rejection of that number. Based on Bayes' Theorem, Posterior is the product of priors and likelihoods. Posterior is the hypothesis probability that we want to find. Prior is a function that provides data before it is proven. Likelihood is a function of the available evidence. Generally there is two ways used to sampling i.e. The Metropolis-Hastings Algorithm and Gibbs Sampling

- The Metropolis-Hasting Algorithm
  Metropolis-Hastings using the accept and reject mechanism to generate a sample sequence from a distribution, which is difficult for sampling [7]. The first is to generate random samples in the desired posterior distribution with a normal distribution, then calculate the probability ratio θ_{new} to θ_{t-1} as in the equation 4

$$r(\theta_{new}, \theta_{t-1}) = \frac{Posterior(\theta_{new})}{Posterior(\theta_{t-1})} \tag{4}$$

  If $r > 1$, value of θ_{new} accepted and if $r < 1$, value of θ_{new} rejected, then create a new random number from normal distribution, if the means $< r(\theta_{new}, \theta_{t-1})$ then value accepted to be θ_{new}. This process runs as many as N iterations.

F. Geometric Brownian Motion

The portofolio cumulative return of stock prices are modeled by Geometric Brownian Motion [8]. The equation of for geometric Brownian motion is a stochastic differential equation :

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{5}$$

where S_t is stock prices at t period, W_t is Brownian Motion, and μ, σ is drift and volatility.

G. Student-T Distribution

Student-T distribution is family of normal distribution with $v > 0$ degrees of freedom. The equation given by

$$f(x|\mu,\lambda,\nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})}\left(\frac{\lambda}{\pi\nu}\right)^{\frac{1}{2}}\left[1+\frac{\lambda(x-\mu)^2}{\nu}\right]^{-\frac{\nu+1}{2}} \quad (6)$$

where $\mu \in (-\infty, \infty)$ is the location, $\lambda > 0$ determines the scales, and $v > 0$ represent the degrees of freedom. The Student-T distribution used to test hypotheses to our beliefs. In statistics t-distribution first derived as a posterior distribution in 1878 by Helmert and Luroth [9]

H. PyMC3 Python

PyMC3 is a new open source Probabilistic Programming framework written in Python that uses Theano to compute gradients via automatic differentiation as well as compile probabilistic programs on-the-fly to C for increased speed. Contrary to other Probabilistic Programming languages, PyMC3 allows model specification directly in Python code. The lack of a domain specific language allows for great flexibility and direct interaction with the model. It features next-generation Markov chain Monte Carlo (MCMC) sampling algorithms such as the No-U-Turn Sampler [10], a self-tuning variant of Hamiltonian Monte Carlo. This class of samplers works well on high dimensional and complex posterior distributions and allows many complex models to be fit without specialized knowledge about fitting algorithms.

III. SYSTEM DESIGN

This study implements apache spark multinode cluster in computing MCMC to predict daily stock prices. The development of the system in this study is based on Apache Spark software with a data processing program created using the python language. Before conducting an analysis of shares, infrastructure and systems from Apache Spark must be built first.

A. Apache Spark Multinode-CLuster

The system is made clustering with spark manager as its cluster management, consisting of one master node and nine worker nodes with identical specifications. In cluster deployment mode on Apache Spark, it needs to be configured so that textit node master and textit slave can interact with each other.

Check whether the Spark is working can see through the spark web page by entering the IP and master port in the URL in the browser as shown in Fig. 5. Even though the slave node's memory is 1 GB but the one given to the master node is less than 1 GB because assuming the others are used for the system. If forced to allocate 2GB to be given to the master node, the slave node will be error and must be restarted. We have to make sure every node can communicate each other with ssh protocol, so we have to setup passwordless ssh on every node.

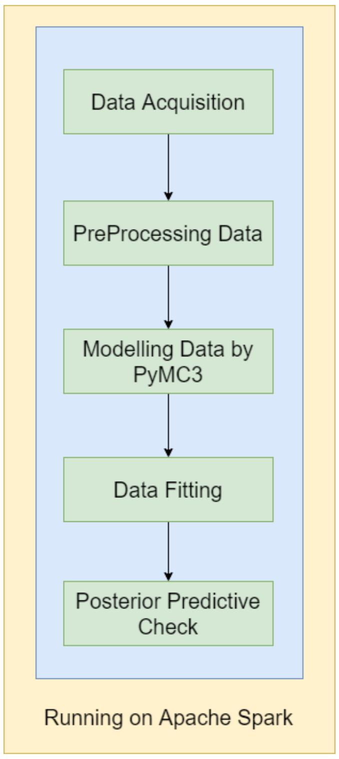
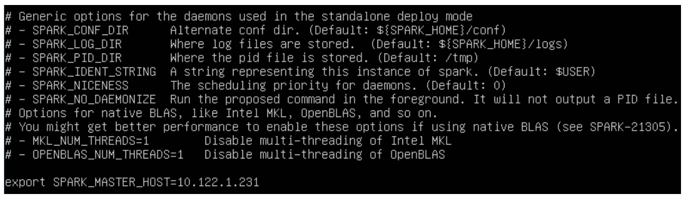


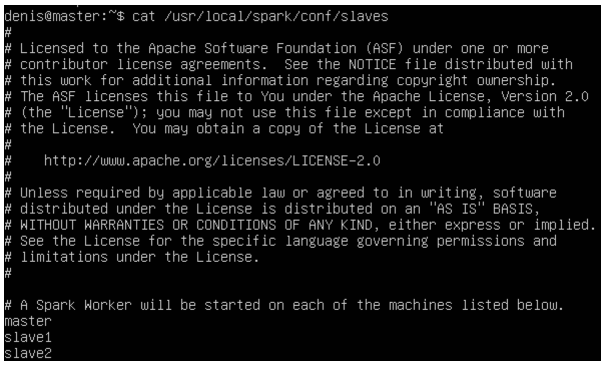
Fig. 2. Data Flow System Design



```
# Generic options for the daemons used in the standalone deploy mode
# - SPARK_CONF_DIR      Alternate conf dir. (Default: ${SPARK_HOME}/conf)
# - SPARK_LOG_DIR       Where log files are stored.  (Default: ${SPARK_HOME}/logs)
# - SPARK_PID_DIR       Where the pid file is stored. (Default: /tmp)
# - SPARK_IDENT_STRING  A string representing this instance of spark. (Default: $USER)
# - SPARK_NICENESS      The scheduling priority for daemons. (Default: 0)
# - SPARK_NO_DAEMONIZE  Run the proposed command in the foreground. It will not output a PID file.
# Options for native BLAS, like Intel MKL, OpenBLAS, and so on.
# You might get better performance to enable these options if using native BLAS (see SPARK-21305).
# - MKL_NUM_THREADS=1        Disable multi-threading of Intel MKL
# - OPENBLAS_NUM_THREADS=1   Disable multi-threading of OpenBLAS

export SPARK_MASTER_HOST=10.122.1.231
```

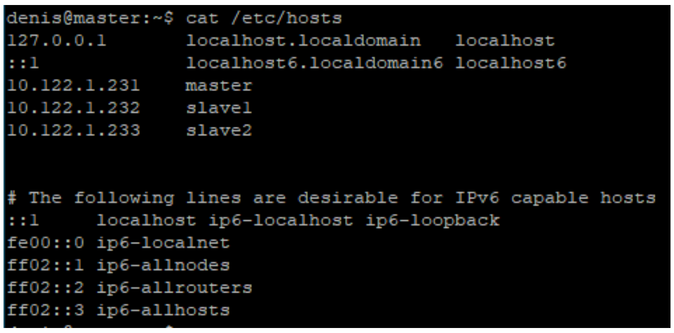
Fig. 3. Configuration of spark-env.sh in every nodes



```
denis@master:~$ cat /usr/local/spark/conf/slaves
#
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# A Spark Worker will be started on each of the machines listed below.
master
slave1
slave2
```

Fig. 4. Configuration of slaves.sh in master node



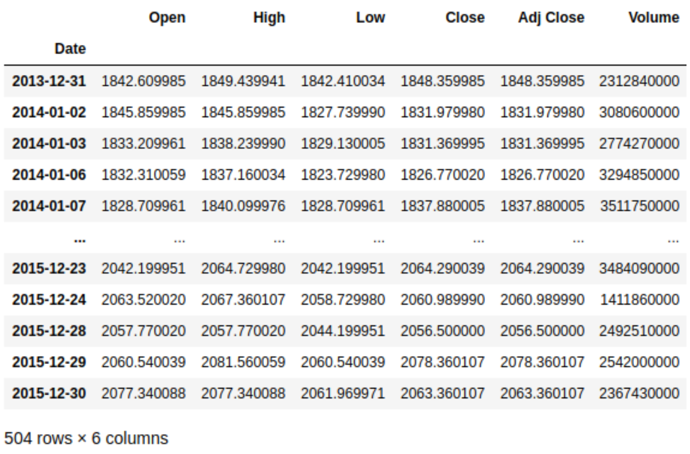
```
denis@master:~$ cat /etc/hosts
127.0.0.1       localhost.localdomain   localhost
::1             localhost6.localdomain6 localhost6
10.122.1.231    master
10.122.1.232    slave1
10.122.1.233    slave2


# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Fig. 5. Configuration of hosts in every nodes

B. Pre-Processing Data

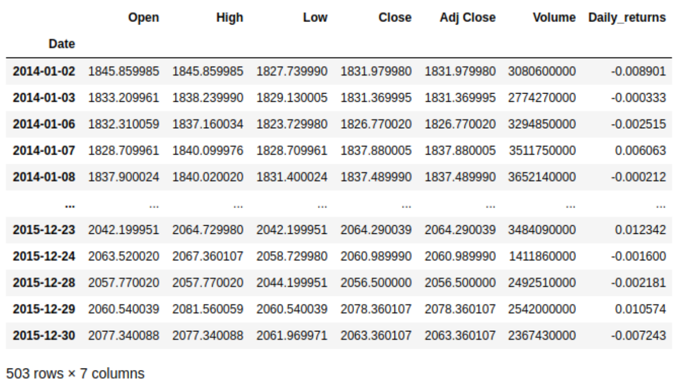
Data that we gonna procces is hitorical raw data from Yahoo finance API. We can get any company data from there. We gonna get SP500 stock with company code GSPC from 2014 to 2015 . The data Fig. 6. is raw data from Yahoo API. After we get data, we calculate daily returns values as log distribution, its means and variance.



	Open	High	Low	Close	Adj Close	Volume
Date						
2013-12-31	1842.609985	1849.439941	1842.410034	1848.359985	1848.359985	2312840000
2014-01-02	1845.859985	1845.859985	1827.739990	1831.979980	1831.979980	3080600000
2014-01-03	1833.209961	1838.239990	1829.130005	1831.369995	1831.369995	2774270000
2014-01-06	1832.310059	1837.160034	1823.729980	1826.770020	1826.770020	3294850000
2014-01-07	1828.709961	1840.099976	1828.709961	1837.880005	1837.880005	3511750000
...	...	...	...	...	...	...
2015-12-23	2042.199951	2064.729980	2042.199951	2064.290039	2064.290039	3484090000
2015-12-24	2063.520020	2067.360107	2058.729980	2060.989990	2060.989990	1411860000
2015-12-28	2057.770020	2057.770020	2044.199951	2056.500000	2056.500000	2492510000
2015-12-29	2060.540039	2081.560059	2060.540039	2078.360107	2078.360107	2542000000
2015-12-30	2077.340088	2077.340088	2061.969971	2063.360107	2063.360107	2367430000

504 rows × 6 columns

Fig. 6. Data from Yahoo API



	Open	High	Low	Close	Adj Close	Volume	Daily_returns
Date							
2014-01-02	1845.859985	1845.859985	1827.739990	1831.979980	1831.979980	3080600000	-0.008901
2014-01-03	1833.209961	1838.239990	1829.130005	1831.369995	1831.369995	2774270000	-0.000333
2014-01-06	1832.310059	1837.160034	1823.729980	1826.770020	1826.770020	3294850000	-0.002515
2014-01-07	1828.709961	1840.099976	1828.709961	1837.880005	1837.880005	3511750000	0.006063
2014-01-08	1837.900024	1840.020020	1831.400024	1837.489990	1837.489990	3652140000	-0.000212
...	...	...	...	...	...	...	...
2015-12-23	2042.199951	2064.729980	2042.199951	2064.290039	2064.290039	3484090000	0.012342
2015-12-24	2063.520020	2067.360107	2058.729980	2060.989990	2060.989990	1411860000	-0.001600
2015-12-28	2057.770020	2057.770020	2044.199951	2056.500000	2056.500000	2492510000	-0.002181
2015-12-29	2060.540039	2081.560059	2060.540039	2078.360107	2078.360107	2542000000	0.010574
2015-12-30	2077.340088	2077.340088	2061.969971	2063.360107	2063.360107	2367430000	-0.007243

503 rows × 7 columns

Fig. 7. Data processed with daily return

C. Modelling and Fitting

The data used is divided into training and testing. Data from sp500 on 2014 to 2015 consist of 504 days. Training data is the first 450 days, and testing is 451 to last day.

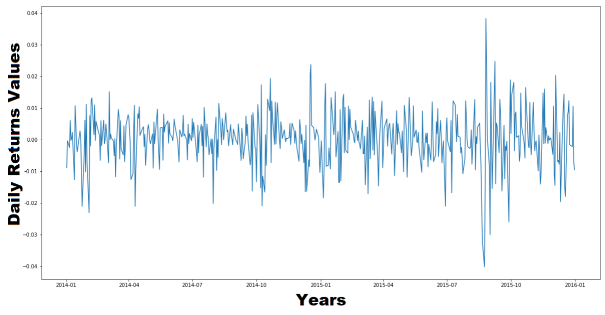


Fig. 8. Daily returns of data 2014-2015

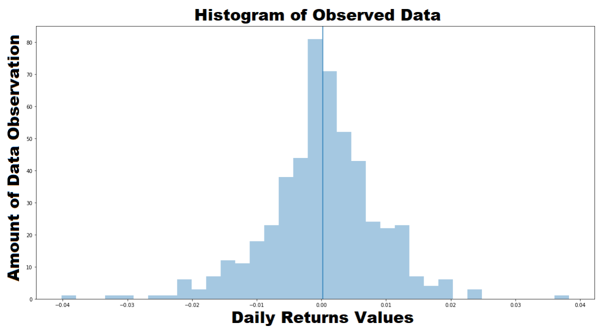


Fig. 9. Histogram of daily returns 2014-2015

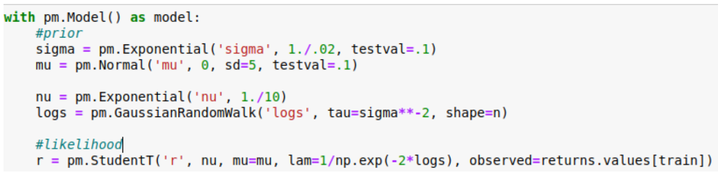
Assume that the data are normally distributed, but the data modelled as StudentT distribution [11].Then create the model with PyMC3 to define prior, likelihood and posterior distribution. Create a data distribution model then determine the parameter values according to what we believe. The prior model is given by

$$\sigma \sim Exp(50) \tag{7}$$

$$\mu \sim Normal(0, 5) \tag{8}$$

$$v \sim Exp(.1) \tag{9}$$

and the likelihood model would be Student T distribution as in Fig. 10 with prior distribution as parameters. Tunning will



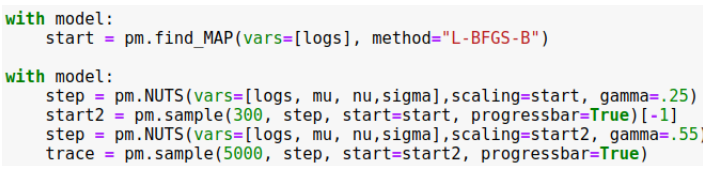
```
with pm.Model() as model:
    #prior
    sigma = pm.Exponential('sigma', 1./.02, testval=.1)
    mu = pm.Normal('mu', 0, sd=5, testval=.1)

    nu = pm.Exponential('nu', 1./10)
    logs = pm.GaussianRandomWalk('logs', tau=sigma**-2, shape=n)

    #likelihood
    r = pm.StudentT('r', nu, mu=mu, lam=1/np.exp(-2*logs), observed=returns.values[train])
```

Fig. 10. PyMC3 data model

do a thousand iteration to generate sample from given data. Sampler will find a maximum posterior from training iteration.



```
with model:
    start = pm.find_MAP(vars=[logs], method="L-BFGS-B")

with model:
    step = pm.NUTS(vars=[logs, mu, nu,sigma],scaling=start, gamma=.25)
    start2 = pm.sample(300, step, start=start, progressbar=True)[-1]
    step = pm.NUTS(vars=[logs, mu, nu,sigma],scaling=start2, gamma=.55)
    trace = pm.sample(5000, step, start=start2, progressbar=True)
```

Fig. 11. Data tunning model

After training data processed, sample will generate data for each parameters $(\mu, logs, \sigma, \nu)$ from posterior distribution as in "Fig. 12.

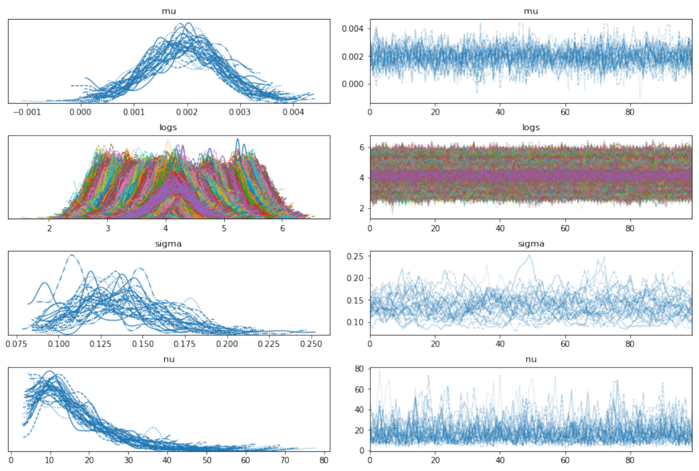


Fig. 12. Tunning Data into model

D. *Posterior Predictive Check*

Validate model with posterior predictive check using PyMC3, this is also generate data from the model using parameters from draws from the posterior. After doing this, we can find out how our model fits and calculate the probability of stock returns up or down for next day."Fig. 13 shows that predicted daily returns r lies 94% HPD between -0.015 and 0.017 with increase probability 52.9% and decrease 47.1%.

IV. Testing and Result

The tests are carried to get best model prediction and check running time of MCMC on Apache Spark Cluster. Computers was used in this experiment has the following spesification :

TABLE I
Hardware Spesification

Node	CPU	Cores	RAM	Cache
Master	FX 6300	6	24GB	2MB
Slave1	E5-2603v3	6	8GB	15MB
Slave2	i3-7100	4	8GB	3MB

The tests is done consider on size of data and number of sampling. Several test done to see which model get the best prediction.

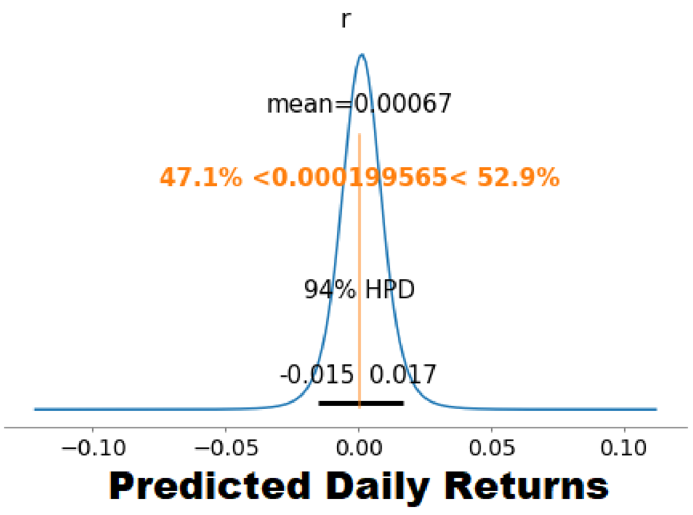


Fig. 13. Posterior predictive check of predicted data

TABLE II
Simulation Scenario

Case	Action	Data Size	Sample Size
1	Large data - Small sampling	2 years data (500)	2000
2	Large data - Large sampling	2 years data (500)	5000
3	Small data - Small sampling	1 years data (250)	2000
4	Small data - Large sampling	1 years data (250)	5000

The first case with two years of data from 2014-2015 and 2000 samples size which generally raising of stock prices. The results get probability of stock prices for next day (2 Jan 2016) raising by 54.16% and going down 45.84%. The running time on Apache Spark is 24 minutes.

The second case with two years of data from 2014-2015 and 5000 samples size which generally raising of stock prices. The results get probability of stock prices for next day (2 Jan 2016) raising by 54.14% and going down 45.86%. The running time on Apache Spark is 43 minutes.

The third case with one years data on 2015 and 2000 samples size which generally raising of stock prices. The results get probability of stock prices for next day (2 Jan 2016) raising by 51.04% and going down 48.96%. The running time on Apache Spark is 23 minutes.

The fourth case with one years data on 2015 and 5000 samples size which generally raising of stock prices. The results get probability of stock prices for next day (2 Jan 2016) raising by 50.14% and going down 49.86%. The running time on Apache Spark is 41 minutes.

While tested with another stock company, model should be different, because of it's characteristic.

Conclusion

From the results of tests that have been carried out, some conclusions can be drawn as follows:

1) The model created for each company can be different. Creating model for one company still need intuition by trader to check their beliefs.
2) The selected data has an affect on the prediction result. We have to select the right range of data for prediction. Stock data up to past years is the best data for prediction.
3) The number of nodes affects the sample chains generation time. The more number of nodes, the less time it takes. Processor capability also has an effect, because the generation of the sample use cores of processor.
4) The amount of memory of executor affects the maximum sample chains could be generate. The more amount of memories(RAM), the more sample chains could be generated.
5) There are not many machine learning library that compatible with Apache Spark, so its need modification to the program to runs distributed on Apache Spark.

References

[1] Wei, D. (2019). Prediction of Stock Price Based on LSTM Neural Network. Proceedings - 2019 International Conference on Artificial Intelligence and Advanced Manufacturing, AIAM 2019, 544–547. https://doi.org/10.1109/AIAM48774.2019.00113

[2] Reddy, N. N., E, N., & Kumar B P, V. (2020). Predicting Stock Price Using Sentimental Analysis Through Twitter Data. 1–5. https://doi.org/10.1109/conecct50063.2020.9198494

[3] (2020) The Cambridge English Dictionary website. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/

[4] F. Nelli, Python Data Analytics. Berkeley, CA New York: Apress, Distributed to the Book trade worldwide by Springer, 2015.

[5] I. Gibranata, The Detective of Stock. Surabaya, Jawa Timur:YLT Publishing, 2011, pp. 15.

[6] Inference, B. (2014). Bayesian Inference Chapter 12. Statistical Machine Learning, 299–351.

[7] B. Steve, G. Andrew, J. Galin, M. Xiao-Li, Handbook of Markov Chain Monte Carlo, NY:Chapman and Hall/CRC, 2011, pp. 20-24.

[8] D. Brigo, A. Dalessandro, M. Neugebauer, F. Triki, Fares. (2009). A stochastic processes toolkit for risk management: mean reverting processes and jumps. Journal of Risk Management in Financial Institutions. 3. 65-83.

[9] (2020) Student's t-distribution. [Online]. Available: https://en.wikipedia.org/wiki/Student27st-distribution

[10] Hoffman, M. D., & Gelman, A. (2014). The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. Journal of Machine Learning Research, 15, 1593–1623.

[11] Rome, S. (2016). Eigen-vesting IV. Predicting Stock and Portfolio Returns With Bayesian Methods. [Online]. Available: https://srome.github.io/Eigenvesting-IV-Predicting-Stock-And-Portfolio-Returns-With-Bayesian-Statistics/