Introduction to Intelligent Computing
Final Project Report

The Diagnosis System of Stock Price

Group 21

102012072 林永昕

102030015 廖穎毅

101060020 劉冠佑

Abstract

This final project report has documented the whole process of how we build our final project. In the first chapter, we point out the reason why the chosen of stock price diagnosis as our domain, where the data is from and how we deal with them. Then, we make clear explanation and solid analysis on the code we implement. Finally, evaluation method is introduced and the simulation result is displayed in the last chapter.

The final project of our group is a “Diagnosis System of Stock Price”. What we aim to achieve is to determine the probability of each ungiven factor that is correlated to stock price whenever a certain stock goes up and other factors are given. This system can facilitate our forecast on stock price, since we are able to have a keen sense of which factors have larger influence on stock price than other factors, and thus provide us insight to make strong investment according to real situation.

As for implementation, first we construct a Bayesian Belief Network based on our domain knowledge on finance and stock investment. Then, Conditional Probability Table of each node is assigned by subjective judgement and objective historical statistic. The data source is obtained from “Yahoo! Finance”, this website allows us download the historical statistic of all the listed companies in the US. In this project, we focus on the stock of Apple Inc. in order to simplify the problem as well as to observe the relation between stock price and operating effectiveness of a certain company.

Finish building the Bayesian Belief Network and Conditional Probability Table, we can start to evaluate the correctness and accuracy of them. Markov Chain Monte Carlo is the algorithm we choose to do simulation, and take advantage of the convenience of the python library “PyMC” to write codes. Finally, the evaluation result is satisfying, proving that our diagnosis system makes sense and is trustworthy.

Content

Chapter 1 Domain and Data Selection

1. Domain

We chose target problem domain mainly based on common interests of all the team members. After discussion, we found out that all of us share a common interest, that is finance and stock investment. Besides, we all have had bad memories on losing money during investment. As a result, in order to increase the probability of making a profit when investing on stocks, we decided to choose "stock price" as our target problem domain. We expected that we could grasp this opportunity to build an AI system to help us perform better on stock investment.

In addition to interest reason, how the domain target problem is well - suited for Bayesian Belief Network is also another concern, after all BBN is the regulated model used in this project. Fortunately, stock price is affected by lots of factors and most of them are causal relation so it is appropriate to use BBN as its model.

To establish a reasonable Bayesian Belief Network, we had to cultivate strong domain knowledge first. Because our own knowledge is not enough, we had done some research about the factors that influence stock price and at the same time went ask the students and advisor of Investment Club in our school. Finally, we were able to choose 10 key factors as our causal state variables in BBN, moreover, endowing causal links to connect the nodes that have causal relation is also achieved.

2. Data Selection

After building a Bayesian Belief Network by domain knowledge, next thing we need to do is to set Conditional Probability Table for each node. In order to ensure the accuracy of the probability in CPT, we need some data source that was generated in real life, instead of assigning all the values by subjective opinion. Hence, we use the data on "Yahoo! Finance" as our data source. We can find the historical statistic of all listed companies on the website, and download it as a csv file, which is easy for us to calculate advanced data we need by Excel or program we wrote. From thousands of stocks, we chose Apple Inc. as our target to analyze, because it is a company that is in the leader status of IT industry, and the other reason is that its stock price has experienced significant rise and fall. Thus, we thought it is suitable for us to diagnose and furthermore learn the relation between industry performance and stock market performance from Apple Inc. There are 6 terms in the downloaded csv file, which are date, open, high, low, close and volume, all of them are representative statistic and crucial to advanced analysis.

With those elementary data, we calculated and got 7 advanced data that all of them become the nodes in our BBN. Those are “today’s price change”, “yesterday’s price change”, “yesterday’s volume change”, “S&P 500”, “Nasdaq”, ”month average price” and “month average volume”. The financial meaning of them and how we calculate to get them will be discussed later. In addition to the above 7 nodes which are all statistical, we also selected 3 key factors which is non-statistical as our causal variables, which are “negative news”, “sales”, and “new product release”. Below is the explanation of our 10 nodes in BBN:

① **new product release:**
We define 10 days before the release day and 20 days after the release day as true, and all the others are false.
People are often attracted by new released products because they are fancy, so whenever new products release, its sale is highly possible to increase.
② **negative news:**
We define 30 days after severe negative news is spread out as true, and all the others are false. “Severe” negative news we select including iCloud is hacked, power button broken, etc.
Whenever some negative news about Apple is spread out, sales usually decline.
③ **sales:**
If the sum of sales of Mac, iPad, iPhone of this quarter is larger than last quarter, then we define it as true, others are false.
It is affected by new product release and negative news, and it further influences the monthly average price and monthly average volume. Usually sales and monthly average price, monthly average volume has positive relation.
④ **monthly average price:**
If monthly average price of this month is higher than yearly average price, then we define it as true, otherwise false.
It is affected by sales and new product release, and it further influences yesterday’s price change and yesterday’s volume change. This data stands for the circulation of the stock price during the year. We often find that some months are always better on sales and thus better on average price than other months.

⑤ **monthly average volume:**

If monthly average volume of this month is larger than yearly average volume, then we define it as true, otherwise false.

It is affected by sales and new product release, and it further influences yesterday’s price change and yesterday’s volume change. This data stands for the circulation of the stock price during the year. We often find that some months are always better on sales and thus larger on average volume than other months.

⑥ **S&P 500 index:**

If the S&P 500 index’s price rises, then we define it as true, otherwise false.

It is a representative cumulated stock index, and highly correlated to individual stock price.

⑦ **Nasdaq index:**

If the Nasdaq index’s price rises, then we define it as true, otherwise false.

It is a representative cumulated stock index, and highly correlated to individual stock price.

⑧ **yesterday’s price change:**

If yesterday’s price rises, then we define it as true, otherwise false.

It is affected by monthly average price, monthly average volume, S&P 500 index and Nasdaq index, it further influences today’s price change.

⑨ **yesterday’s volume change**

If yesterday’s volume rises, then we define it as true, otherwise false.

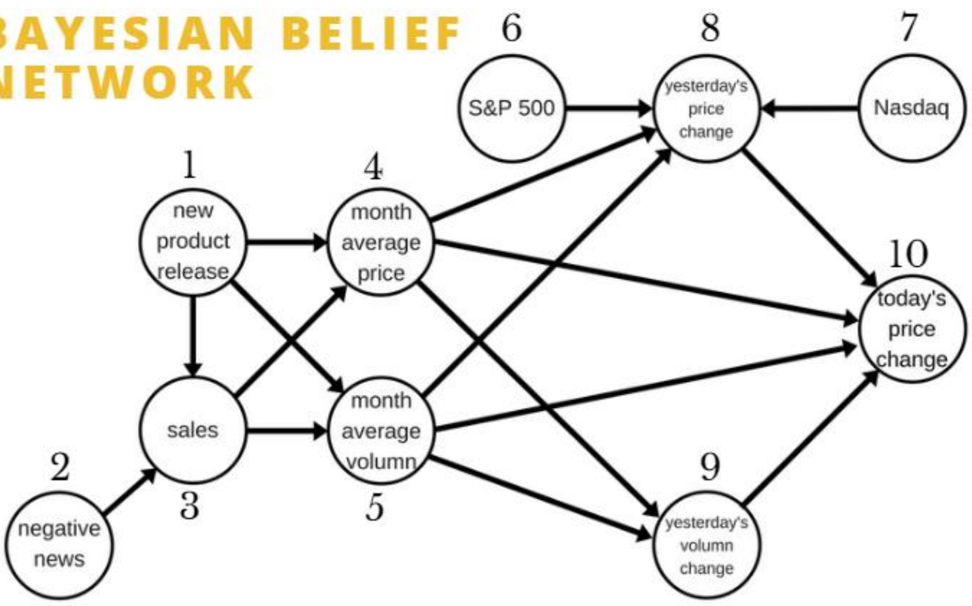
It is affected by monthly average price and monthly average volume, it further influences today’s price change.

⑩ **today’s price change**

If today’s price rises, then we define it as true, otherwise false.

It is our mainly target node, in other words, the problem that we would like to diagnose. All we want to do in this project is to obtain the ungiven nodes’ probability under the condition that today’s price rises or falls and some nodes given. It is affected by monthly average price, monthly average volume, yesterday’s price change and yesterday’s volume change, it further influences today’s price change.

BAYESIAN BELIEF NETWORK



Chapter 2 Code Overview

```
In [1]:
%matplotlib inline
import daft
import pymc
import matplotlib.pyplot as plt
import matplotlib as mlp
import numpy as np
import ggplot
import pandas as pd

# Initialization
observed_values = [1.]
```

In [1] is importing necessary libraries.

```
In [2]:
# Tier 1 node
new_product_release = pymc.Bernoulli('new_product_release', 0.197933, value=np.ones(len(observed_values)))

negative_news = pymc.Bernoulli('negative_news', 0.0389507, value=np.ones(len(observed_values)))

S_and_P_500 = pymc.Bernoulli('S_and_P_500', 0.534976, value=np.ones(len(observed_values)))

Nasdaq = pymc.Bernoulli('Nasdaq', 0.547695, value=np.ones(len(observed_values)))
```

In [2] is initializing the probability of each tier 1 node. A tier 1 node means it's a node which doesn't have any parent, so we can assign the probability to each Tier 1 node by using Bernoulli distribution directly.

```
In [3]:
1  # Tier 2 node
2  p_sales = pymc.Lambda('p_sales', lambda new_product_release=new_product_release, negative_news=negative_news: \
3                        np.where(new_product_release, np.where(negative_news, 1-10e-5, 0.628571), \
4                                 np.where(negative_news, 0.1, 0.414414)))
5
6  sales = pymc.Bernoulli('sales', p_sales, value=np.ones(len(observed_values)))
```

In[3] is initializing the conditional probability of the tier 2 node with 2 parents. A tier 2 ~ tier 6 node is a node that it has at least one parent node in Bayesian network, and the CPT is done in line 3 by setting corresponding value to each condition.
First we have to make a Lambda distribution which includes the CPT information, because in pyMC we can't assign CPT to Bernoulli distribution directly. Later, we pass the lambda distribution to Bernoulli distribution. For example, this node has 2 parent nodes, which means there are 4 conditions (TT/TF/FT/FF) in its CPT.
And the value assigning process is done by following code.
np.where(Event1, np.where(Event2, true, false), np.where(Event2, true, false))
                    ^ Event1 is true   ^Event2 is true

```
In [4]:
# Tier 3 node
p_month_average_price = pymc.Lambda('p_month_average_price', lambda sales=sales, new_product_release=new_product_release: \
                                    np.where(sales, np.where(new_product_release, 0.660819, 0.498795), \
                                             np.where(new_product_release, 0.551282, 0.429293)))

month_average_price = pymc.Bernoulli('month_average_price', p_month_average_price, value=np.ones(len(observed_values)))

p_month_average_volumn = pymc.Lambda('p_month_average_volumn', lambda sales=sales, new_product_release=new_product_release:\
                                    np.where(sales, np.where(new_product_release, 0.438596, 0.448193), \
                                             np.where(new_product_release, 0.769231, 0.358586)))

month_average_volumn = pymc.Bernoulli('month_average_volumn', p_month_average_volumn, value=np.ones(len(observed_values)))
```

```
In [5]: # Tier 4 node
        p_yesterday_price_change = pymc.Lambda('p_yesterday_price_change', lambda month_average_price=month_average_price, \
                                        month_average_volumn=month_average_volumn, S_and_P_500=S_and_P_500, Nasdaq=Nasdaq: \
                                        np.where(month_average_price, \
                                                 np.where(month_average_volumn, \
                                                          np.where(S_and_P_500, \
                                                                   np.where(Nasdaq, 0.695652, 0.125), \
                                                                   np.where(Nasdaq, 0.647059, 0.290323)), \
                                                          np.where(S_and_P_500, \
                                                                   np.where(Nasdaq, 0.711628, 0.428571), \
                                                                   np.where(Nasdaq, 0.589744, 0.27933))), \
                                                 np.where(month_average_volumn, \
                                                          np.where(S_and_P_500, \
                                                                   np.where(Nasdaq, 0.724719, 0.304348), \
                                                                   np.where(Nasdaq, 0.529412, 0.342105)), \
                                                          np.where(S_and_P_500, \
                                                                   np.where(Nasdaq, 0.689394, 0.210526), \
                                                                   np.where(Nasdaq, 0.5, 0.268041)))))

        yesterday_price_change = pymc.Bernoulli('yesterday_price_change', p_yesterday_price_change, value=np.ones(len(observed_values)))

        p_yesterday_price_volumn = pymc.Lambda('p_yesterday_price_volumn', lambda \
                                         p_month_average_price=p_month_average_price, month_average_volumn=month_average_volumn: \
                                         np.where(p_month_average_price, np.where(month_average_volumn, 0.469512, 0.475771), \
                                                                         np.where(month_average_volumn, 0.489189, 0.448148)))

        yesterday_price_volumn = pymc.Bernoulli('yesterday_price_volumn', p_yesterday_price_volumn,  value=np.ones(len(observed_values)))
```

```
In [6]: # Tier 5 node
        p_today_price_change = pymc.Lambda('p_today_price_change', lambda month_average_price=month_average_price,\
                                    month_average_volumn=month_average_volumn, yesterday_price_change=yesterday_price_change, \
                                    yesterday_price_volumn=yesterday_price_volumn:\
                                    np.where(month_average_price, \
                                             np.where(month_average_volumn, \
                                                      np.where(yesterday_price_change, \
                                                               np.where(yesterday_price_volumn, 0.552632, 0.439024), \
                                                               np.where(yesterday_price_volumn, 0.282051, 0.608696)), \
                                                      np.where(yesterday_price_change, \
                                                               np.where(yesterday_price_volumn, 0.62037, 0.417323), \
                                                               np.where(yesterday_price_volumn, 0.425926, 0.621622))), \
                                             np.where(month_average_volumn, \
                                                      np.where(yesterday_price_change, \
                                                               np.where(yesterday_price_volumn, 0.561798, 0.5), \
                                                               np.where(yesterday_price_volumn, 0.456522, 0.62963)), \
                                                      np.where(yesterday_price_change, \
                                                               np.where(yesterday_price_volumn, 0.462963, 0.461538), \
                                                               np.where(yesterday_price_volumn, 0.492537, 0.56338)))))

        today_price_change = pymc.Bernoulli('today_price_change', p_today_price_change, value=np.ones(len(observed_values)))
```

In In [4]~In[6] is similar to In[3], initializing CPTs from tier 3 to tier 5 nodes.

```
In [7]: model = pymc.Model([today_price_change, p_today_price_change, \
                            yesterday_price_volumn, p_yesterday_price_volumn, yesterday_price_change, p_yesterday_price_change, \
                            month_average_volumn, p_month_average_volumn, month_average_price, p_month_average_price, \
                            sales, p_sales, \
                            new_product_release, negative_news, S_and_P_500, Nasdaq])
```

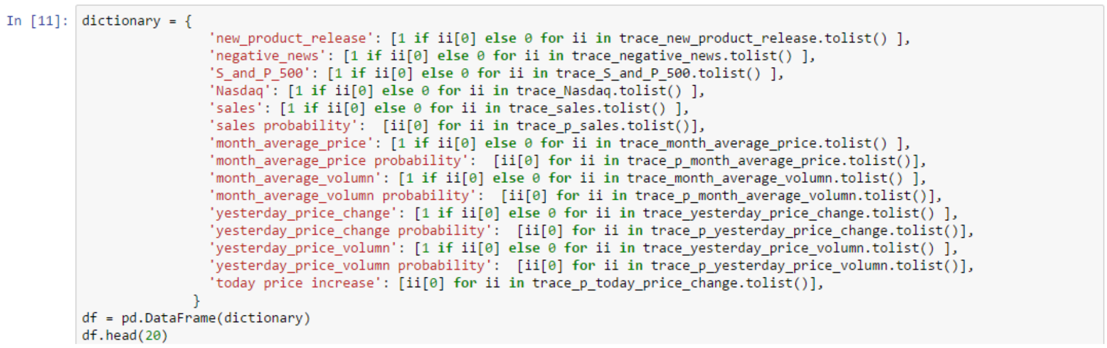
In [7] is to make a model of the Bayesian belief network in this project by passing a set of all distributions above to the pymc.Model.

```
In [8]: mcmc = pymc.MCMC(model)
        mcmc.sample(60000, 20000)
```

In [8] starts the Markov-Chain Monte-Carlo algorithm to sample the model we built in In [7] 60000 times, and we discard the first 20000 samples, because we only want to see the convergent results of this model.

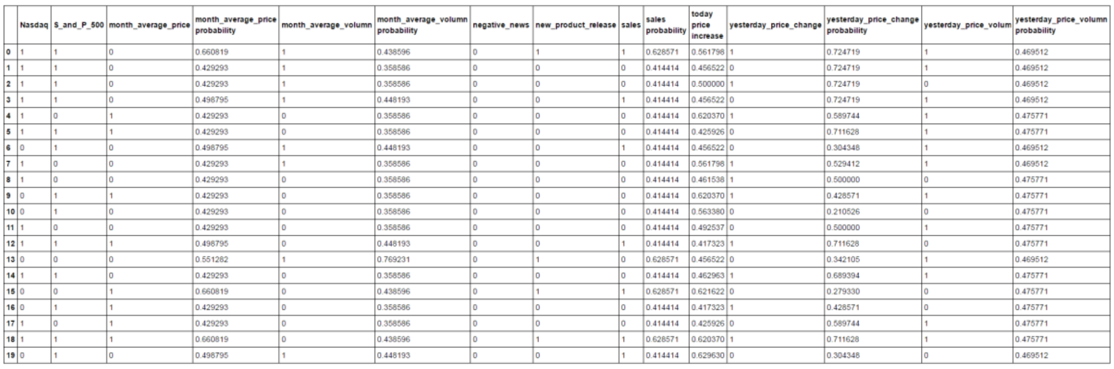
```
In [9]: trace_new_product_release = mcmc.trace('new_product_release')[:]
        trace_negative_news = mcmc.trace('negative_news')[:]
        trace_S_and_P_500 = mcmc.trace('S_and_P_500')[:]
        trace_Nasdaq = mcmc.trace('Nasdaq')[:]
        trace_sales = mcmc.trace('sales')[:]
        trace_p_sales = mcmc.trace('p_sales')[:]
        trace_month_average_price = mcmc.trace('month_average_price')[:]
        trace_p_month_average_price = mcmc.trace('p_month_average_price')[:]
        trace_month_average_volumn = mcmc.trace('month_average_volumn')[:]
        trace_p_month_average_volumn = mcmc.trace('p_month_average_volumn')[:]
        trace_yesterday_price_change = mcmc.trace('yesterday_price_change')[:]
        trace_p_yesterday_price_change = mcmc.trace('p_yesterday_price_change')[:]
        trace_yesterday_price_volumn = mcmc.trace('yesterday_price_volumn')[:]
        trace_p_yesterday_price_volumn = mcmc.trace('p_yesterday_price_volumn')[:]
        trace_p_today_price_change = mcmc.trace('p_today_price_change')[:]
```

In [9] traces all the distribution declared in previous cells to help us analyzing desired conditional probability later.



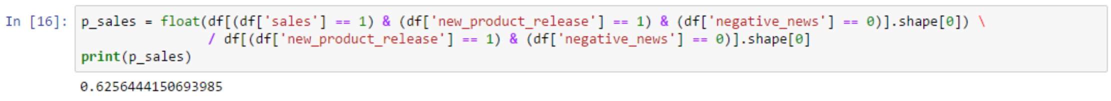
```
In [11]: dictionary = {
                     'new_product_release': [1 if ii[0] else 0 for ii in trace_new_product_release.tolist() ],
                     'negative_news': [1 if ii[0] else 0 for ii in trace_negative_news.tolist() ],
                     'S_and_P_500': [1 if ii[0] else 0 for ii in trace_S_and_P_500.tolist() ],
                     'Nasdaq': [1 if ii[0] else 0 for ii in trace_Nasdaq.tolist() ],
                     'sales': [1 if ii[0] else 0 for ii in trace_sales.tolist() ],
                     'sales probability':  [ii[0] for ii in trace_p_sales.tolist()],
                     'month_average_price': [1 if ii[0] else 0 for ii in trace_month_average_price.tolist() ],
                     'month_average_price probability':  [ii[0] for ii in trace_p_month_average_price.tolist()],
                     'month_average_volumn': [1 if ii[0] else 0 for ii in trace_month_average_volumn.tolist() ],
                     'month_average_volumn probability':  [ii[0] for ii in trace_p_month_average_volumn.tolist()],
                     'yesterday_price_change': [1 if ii[0] else 0 for ii in trace_yesterday_price_change.tolist() ],
                     'yesterday_price_change probability':  [ii[0] for ii in trace_p_yesterday_price_change.tolist()],
                     'yesterday_price_volumn': [1 if ii[0] else 0 for ii in trace_yesterday_price_volumn.tolist() ],
                     'yesterday_price_volumn probability':  [ii[0] for ii in trace_p_yesterday_price_volumn.tolist()],
                     'today price increase': [ii[0] for ii in trace_p_today_price_change.tolist()],
                   }
         df = pd.DataFrame(dictionary)
         df.head(20)
```

In [11] makes a dictionary of all samples with some keywords, and show the first 20 answer on screen after discarding first 20000 samples.
The result is also shown below.



	Nasdaq	S_and_P_500	month_average_price	month_average_price probability	month_average_volumn	month_average_volumn probability	negative_news	new_product_release	sales	sales probability	today price increase	yesterday_price_change	yesterday_price_change probability	yesterday_price_volum	yesterday_price_volumn probability
0	1	1	0	0.660819	1	0.438596	0	1	1	0.628571	0.561798	1	0.724719	1	0.469512
1	1	1	0	0.429293	1	0.358586	0	0	0	0.414414	0.456522	0	0.724719	1	0.469512
2	1	1	0	0.429293	1	0.358586	0	0	0	0.414414	0.500000	1	0.724719	0	0.469512
3	1	1	0	0.498795	1	0.448193	0	0	1	0.414414	0.456522	0	0.724719	1	0.469512
4	1	0	1	0.429293	0	0.358586	0	0	0	0.414414	0.620370	1	0.589744	1	0.475771
5	1	1	1	0.429293	0	0.358586	0	0	0	0.414414	0.425926	0	0.711628	1	0.475771
6	0	1	0	0.498795	1	0.448193	0	0	1	0.414414	0.456522	0	0.304348	1	0.469512
7	1	0	0	0.429293	1	0.358586	0	0	0	0.414414	0.561798	1	0.529412	1	0.469512
8	1	0	0	0.429293	0	0.358586	0	0	0	0.414414	0.461538	1	0.500000	0	0.475771
9	0	1	1	0.429293	0	0.358586	0	0	0	0.414414	0.620370	1	0.428571	1	0.475771
10	0	1	0	0.429293	0	0.358586	0	0	0	0.414414	0.563380	0	0.210526	0	0.475771
11	1	0	0	0.429293	0	0.358586	0	0	0	0.414414	0.492537	0	0.500000	1	0.475771
12	1	1	1	0.498795	0	0.448193	0	0	1	0.414414	0.417323	1	0.711628	0	0.475771
13	0	0	0	0.551282	1	0.769231	0	1	0	0.628571	0.456522	0	0.342105	1	0.469512
14	1	1	0	0.429293	0	0.358586	0	0	0	0.414414	0.462963	1	0.689394	1	0.475771
15	0	0	1	0.660819	0	0.438596	0	1	1	0.628571	0.621622	0	0.279330	0	0.475771
16	0	1	1	0.429293	0	0.358586	0	0	0	0.414414	0.417323	1	0.428571	0	0.475771
17	1	0	1	0.429293	0	0.358586	0	0	0	0.414414	0.425926	0	0.589744	1	0.475771
18	1	1	1	0.660819	0	0.438596	0	1	1	0.628571	0.620370	1	0.711628	1	0.475771
19	0	1	0	0.498795	1	0.448193	0	0	1	0.414414	0.629630	0	0.304348	0	0.469512

After all, we can search desired conditional probability by giving conditions, for example, P (sales| new_product_release=True, negative_news=False)
  = P (sales=True, new_product_release=True, negative_news=False)
    / P (new_product_release=True, negative_news=False)
We search the dictionary to get the data fitting our conditions by:



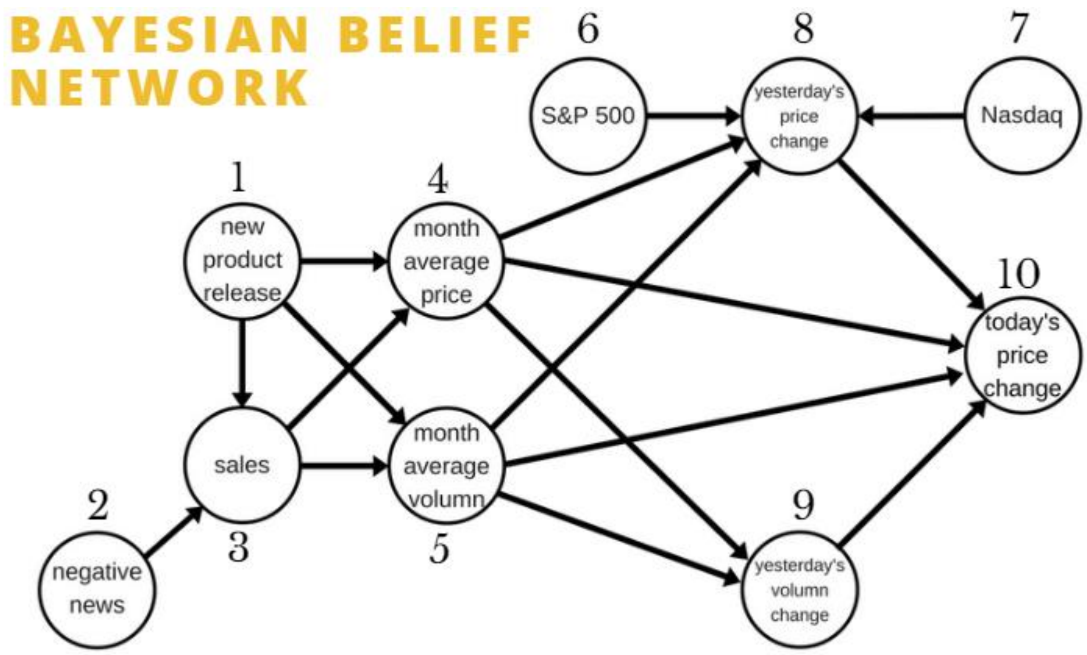
```
In [16]: p_sales = float(df[(df['sales'] == 1) & (df['new_product_release'] == 1) & (df['negative_news'] == 0)].shape[0]) \
                 / df[(df['new_product_release'] == 1) & (df['negative_news'] == 0)].shape[0]
         print(p_sales)

         0.6256444150693985
```

The real probability is 0.628571. It's very close. And we can use this method to get further information later in the evaluation part.

Chapter 3 Evaluation Method and Result

1. Conditional Probability Table



P(Node1 True)
0.197933

P(Node2 True)
0.0389507

P(Node6 True)
0.534976

P(Node7 True)
0.547695

Node1	Node2	P(Node3 True)
F	F	0.414414
F	T	0.1
T	F	0.628571
T	T	1

Node1	Node3	P(Node4 True)
F	F	0.429293
F	T	0.498795
T	F	0.551282
T	T	0.660819

Node1	Node3	P(Node5 True)
F	F	0.358586
F	T	0.448193
T	F	0.769231
T	T	0.438596

Node4	Node5	P(Node9 True)
F	F	0.448148
F	T	0.489189
T	F	0.475771
T	T	0.469512

Node4	**Node5**	**Node6**	**Node7**	**P(Node8 True)**
F	F	F	F	0.268041
F	F	F	T	0.5
F	F	T	F	0.210526
F	F	T	T	0.689394
F	T	F	F	0.342105
F	T	F	T	0.529412
F	T	T	F	0.304348
F	T	T	T	0.724719
T	F	F	F	0.27933
T	F	F	T	0.589744
T	F	T	F	0.428571
T	F	T	T	0.711628
T	T	F	F	0.290323
T	T	F	T	0.647059
T	T	T	F	0.125
T	T	T	T	0.695652

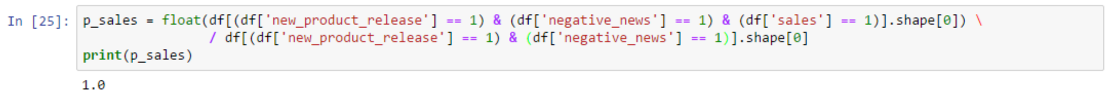
Node4	**Node5**	**Node8**	**Node9**	**P(Node10 True)**
F	F	F	F	0.56338
F	F	F	T	0.492537
F	F	T	F	0.461538
F	F	T	T	0.462963
F	T	F	F	0.62963
F	T	F	T	0.456522
F	T	T	F	0.5
F	T	T	T	0.561798
T	F	F	F	0.621622
T	F	F	T	0.425926
T	F	T	F	0.417323
T	F	T	T	0.62037
T	T	F	F	0.608696
T	T	F	T	0.282051
T	T	T	F	0.439024
T	T	T	T	0.552632

2. Result:

We use two methods to evaluate the accuracy of our Bayesian Belief Model. One is setting conditions fitting CPT and test the accuracy, while the other one is choosing several data from data set to calculate the accuracy of prediction. Note that we predict true when the conditional probability is greater than 0.5.

1. Diagnosis

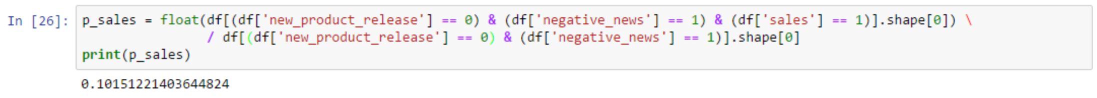
   Data 1: P(sales=True|new_product_release=True, negative_news=True) = 1



```
In [25]: p_sales = float(df[(df['new_product_release'] == 1) & (df['negative_news'] == 1) & (df['sales'] == 1)].shape[0]) \
                   / df[(df['new_product_release'] == 1) & (df['negative_news'] == 1)].shape[0]
         print(p_sales)

         1.0
```

   Data 2: P(sales=True|new_product_release=False, negative_news=True) = 0.1



```
In [26]: p_sales = float(df[(df['new_product_release'] == 0) & (df['negative_news'] == 1) & (df['sales'] == 1)].shape[0]) \
                   / df[(df['new_product_release'] == 0) & (df['negative_news'] == 1)].shape[0]
         print(p_sales)

         0.10151221403644824
```

   By these simple checks, the result is very close to our CPT in node 'sales'. We can conclude that our model is fitting to the dataset. Let's check which condition dominates whether today' price change higher.

   P(any one condition=True|today_price_change=True)

new_product_release	negative_news	sales	month_average_price	month_average_volumn
0.196593	0.039087	0.449663	0.481484	0.425271
S_and_P_500	Nasdaq	yesterday_price_change	yesterday_price_volumn	today_price_change
0.533633	0.544015	0.468687	0.450765	TRUE

   By the list above, we can conclude that 'Nasdaq' dominates whether today's price increase or not the most.

2. Prediction

   Use DataSet {new_product_release, negative_news, sales, month_average_price, month_average_volumn, S&P500, Nasdaq, yesterday_price_change, yesterday_volumn_change} to get the probability of today's price increase.

   If the probability is greater than 0.5, predict true. Let's check prediction accuracy. (0:false, 1:true)

Fact # Data set => result P(true|dataset)	prediction	result
Fact 01: {001110001} => 1 Prediction: 0.297 =>	false	wrong
Fact 02: {000100110} => 0 Prediction: 0.405 =>	false	correct
Fact 03: {101001111} => 1 Prediction: 0.445 =>	false	wrong
Fact 04: {001101110} => 1 Prediction: 0.404 =>	false	wrong
Fact 05: {101000010} => 1 Prediction: 0.452 =>	false	wrong
Fact 06: {101001110} => 0 Prediction: 0.453 =>	false	correct
Fact 07: {101010000} => 0 Prediction: 0.350 =>	false	correct
Fact 08: {000010001} => 0 Prediction: 0.465 =>	false	correct

```
Fact 09: {000100101} => 0 Prediction: 0.435 =>    false        correct
Fact 10: {001110110} => 0 Prediction: 0.438 =>    false        correct
```

Accuracy=(# correct) / (# predictions) = 0.6.

We do a simple prediction among ten days data, and the accuracy is acceptable, which means it’s greater 0.5. But there are two issue we can’t solve: A. How to make Prediction more correct, maybe we can’t directly pick 0.5 as the boundary; B. the predictions we make is too small, maybe we should test more data.

Conclusion

We have done and learned a lot of things in this final project. First of all, we spent about two weeks understanding all the theory that has to do with this project, including Bayesian Belief Network, Conditional Probability Table and Markov Chain Monte Carlo algorithm. The content of these theories are not easy at all, however, once we completely understand how they work, we can implement it to code with fewer effort. Thus, we thought that it is worth spending much time on comprehension.

After fully understanding BBN, we started to choose an appropriate domain. In fact, we had much fun when we were choosing topic, because we could brainstorm and brought our creativity into the project. Whenever we came up with a new idea that might solve some problems in daily life, it made us really happy and felt sense of fulfillment. Then, we decided to do the diagnosis system of stock price. Having an opportunity to solve financial problems by Artificial Intelligence made us feel even more excited about the results, and hoping that the system we were going to build is able to assist us in stock investment.

However, things were not going so well as we had expected. Although we had no problem retrieving data from "Yahoo! Finance" and analyze it, neither did we face obstacles building BBN and implementing MCMC, we found it difficult to make the result accurate. At the beginning, we were frustrated about the low performance, and had no idea where we should modify to improve our system. As a result, we re-examined from the very beginning, checked if the domain knowledge we set on the BBN causal relation is reasonable. By doing more research on stock investment, we changed about half of our BBN connection and CPT value, and finally could we obtain accurate simulation result.

In summary, this is a tough project, but it brings us a lot. We all felt satisfied after we finished this task. The end of final project indicates the end of the course "Introduction to Intelligent Computing", we have learnt a lot of new knowledge in class, conquered many difficulties in project, most importantly, have cultivated the ability of solving problems in various domains from the perspective of Artificial Intelligence.