ACQUIRING BUSINESS INSIGHT THROUGH REAL-TIME KUALA LUMPUR STOCK EXCHANGE DATA USING DATA MINING

NUR AMIRA BINTI MOHAMAD MANSOR

WQD180094

FACULTY OF COMPUTER SCIENCE AND

INFORMATION TECHNOLOGY

UNIVERSITY OF MALAYA

2018 / 2019

WQD7005

DATA MINING AND WAREHOUSING PROJECT REPORT

ACQUIRING BUSINESS INSIGHT THROUGH REAL-TIME KUALA LUMPUR STOCK EXCHANGE DATA USING DATA MINING

LECTURER:

ASSOC. PROF. DR. TEH YING WAH

PREPARED BY:

NUR AMIRA BINTI MOHAMAD MANSOR

WQD180094

02 JUNE 2019

CONTENTS OF REPORT

ABSTRACT

This report will firstly cover the introduction of data mining along with the data retrieval from the web method. Data mining is the act of extracting valuable information from large databases or better known as data warehouse. Data mining implementations use artificial intelligence algorithms such as the clustering, neural network, decision tree, and rule induction to identify interesting patterns inside databases and from it, useful information can be gained. Neural network is a loose model, which resembles on how the human brain works. Association rule and rule induction are algorithms, which utilizes rules in making a prediction. The difference between them is that association rule algorithm extract raw models which need further transformation into a classification rule-set.

In this project, we are going to acquire on the business insight through real-time Kuala Lumpur Stock Exchange (KLSE) data using data mining techniques where we crawl the stock exchange data in order to experience predictive analysing and prescriptive analysing data with the main objective of solving real-world problems.

ACKNOWLEDGEMENT

First and foremost, I would like to extend my gratitude to my family for their love and care which gave me the light and strength during obscurity.

Not forgotten, I would also like to express my sincere appreciation to my supervisor, Assoc. Prof. Dr Teh Ying Wah for his advices, guidance and efforts in helping me to complete this project.

I would also like to thank my fellow teammates and friends who had given the critique and tremendous support in ensuring that the codes will be extensive and also made a substantial contribution through brilliant ideas in producing this project.

Nur Amira binti Mohamad Mansor
Department of Information System
Faculty of Computer Science and Information Technology
University of Malaya
Kuala Lumpur

INTRODUCTION

The first process that was done in this project was to acquire the data. About **1854 companies** were retrieved in total. Below are the sources of data which was crawled using Python:

1) https://www.thestar.com.my/business/marketwatch/stocks/?qcounter=
2) https://www.malaysiastock.biz/Market-Watch.aspx?type=A&value=
3) Malaysian Indices
4) PDF extraction of company's quarterly report
5) Twitter data on the companies

All data retrieved are in the folders attached together with this report.

After pulling and gathering the large volumes of data and merging multiple sources [such as web information (stock indices), Twitter, news and company's quarterly reports], the data are analysed for analytics. This is a repetitive process until a useful set of data is produced.

Then, the co-variance of the stocks were investigated; whether each company's data affect another, and what kind of industries are related to each other during the processing of the data.

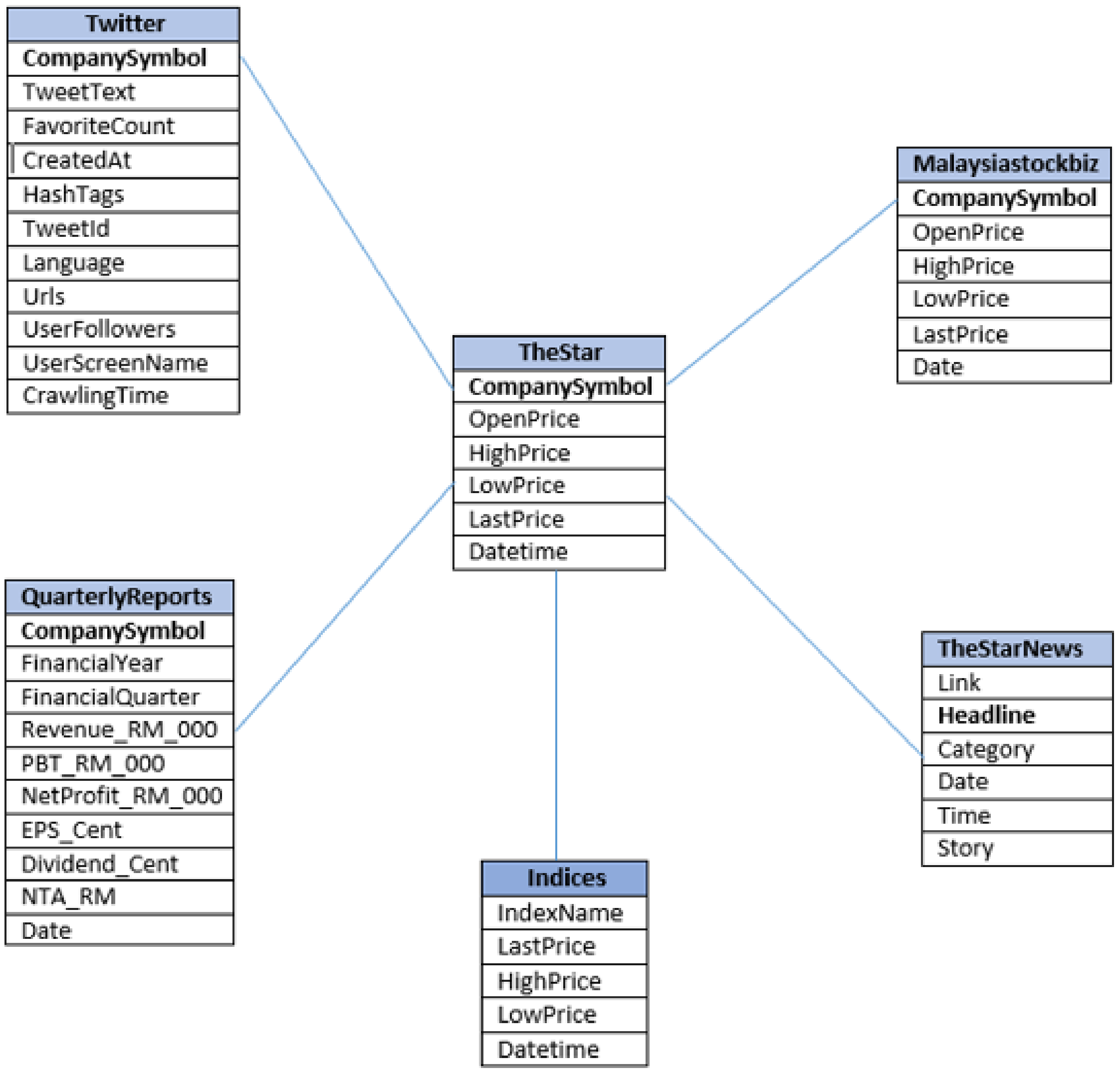
Interpretation of data is the next step where in this step, the data gathered are interpreted to layman language to provide business insights of the data which is the main objective of this project.

With this, potential user recommendations around optimal actions to achieve investment objectives such as profits and return on investment are further discussed and laid down.

THE STUDY

In this study, Python programming language was applied to crawl the data. The crawler was ran using crontab (a Linux scheduler utility) in an Amazon server in order to let it running automatically in real time. For Twitter data, the crawling was done to include the popularity of the tweets (counted the Favourites and Retweets) in order to showcase the popularity and the authenticity of the tweets. In this case, the more the number of retweets and likes the tweets get, it is believed that the tweets are popular among Twitter users, which are then considered to be included into this project.

Star Schema of data collected



TheStar is the fact table and there are five dimension tables; Malaysianstockbiz, TheStarNews, Indices, QuarterlyReports and Twitter.

Each dimension table has a primary key on its CompanySymbol, relating to one of the column (viewed as rows in the example schema) of the fact TheStar table's six-column (compound) primary key (CompanySymbol). The non-primary key OpenPrice, HighPrice, LowPrice, LastPrice, Datetime columns of the fact table represent the measure of metric that can be used in calculation and analysis. The non-primary key columns of the dimension tables represent additional attributes of the dimensions (such as TweetText of the Twitter dimension).

From combination of different dimension tables different results are obtained. By roll up by CompanySymbol, the result able to show the prices of the companies. And drill down from the above result by Malaysianstockbiz able to show the prices of the companies as well.

Another example is that, select CompanySymbol from Twitter will produce the TweetText on the companies. When drilling down QuarterlyReports, the net profits are able to see which are the profitable ones that worth investigating on. If drill down to TheStarNews, the related news headlines are able to be viewed. Lastly, drilling down to Indices can showcase all the company's indices on the particular date and time as well as the prices.

After data gathering method, the data were cleaned and all null values and noises are cleared from the data to reduce the useless data using data reduction technique. Data reduction is the process of reducing the amount of capacity required to store data. Data reduction can increase storage efficiency and reduce costs. In this process, all the retrieved data are sorted by its time and date after removing the null values and noises. Then, the data were pumped into MySQL database in order to store them. However, due to MySQL basic license limitation, only limited number of data were imported into the database. However, the amount was sufficient to be worked on and analysed.

Next, the co-variance of the stocks are investigated. In this step, two companies were chosen as examples; Maybank and CIMB, to find out if their prices somehow affects one another or affected by other foreign factors such as controversial tweets or the companies' current business health. In this part, it is observed that these 2 companies are in no way relate to each other; when Maybank stock prices recorded an increase or decrease, it does not in any way affect CIMB stock prices. Apart from that, for a particular scenario, the world's economy current status at the moment retrieved through some tweets do affect Maybank and CIMB stock prices independently. This indicates that the world's economy status definitely affects the progress of these 2 companies.

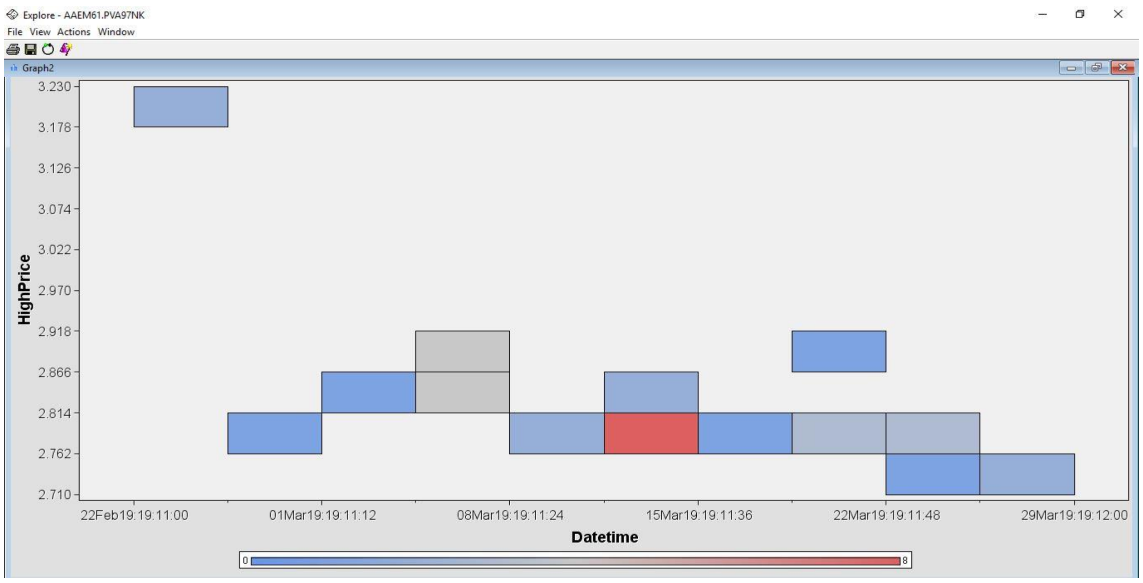
ANALYSIS AND FINDINGS (BUSINESS INSIGHT)

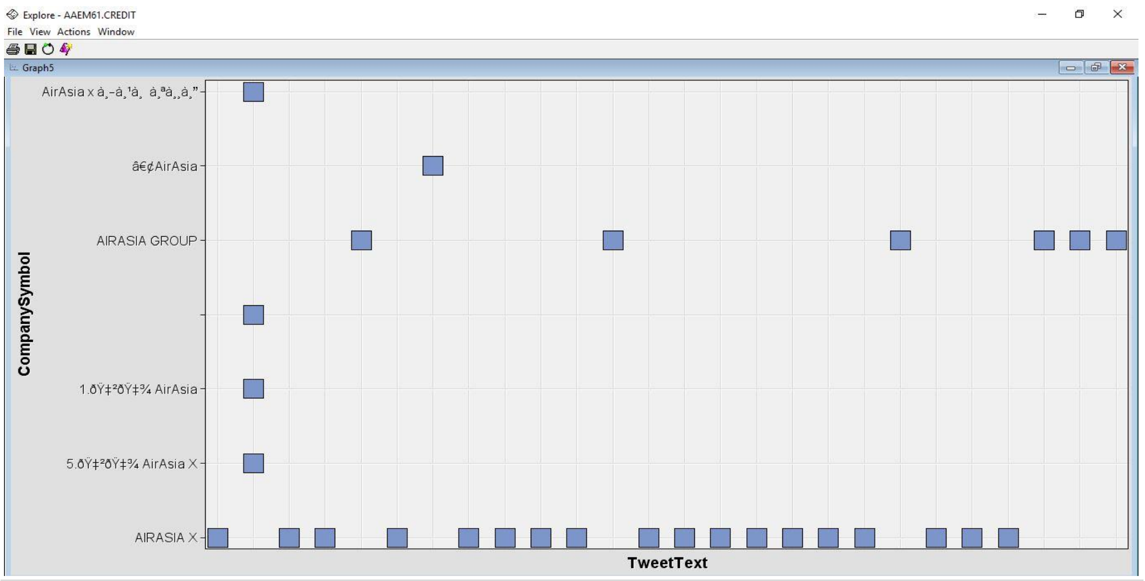
The analysis goal is to determine the effect and relevance of company's popularity on its market price using Twitter data (tweets) along with the relevant companies' stock market price. This is to investigate if the company is being tweeted or retweeted for multiple times positively or negatively, will these affect its stock price or not.

In this analysis, categorical and historical data extracted from Twitter with companies' names as keywords, as well as the type of tweets that are being tweeted by user. For example, if it is positive tweets or negative tweets. The numerical data extracted from TheStar, MalaysianBiz, and Indices are considered as quantitative data of this study.

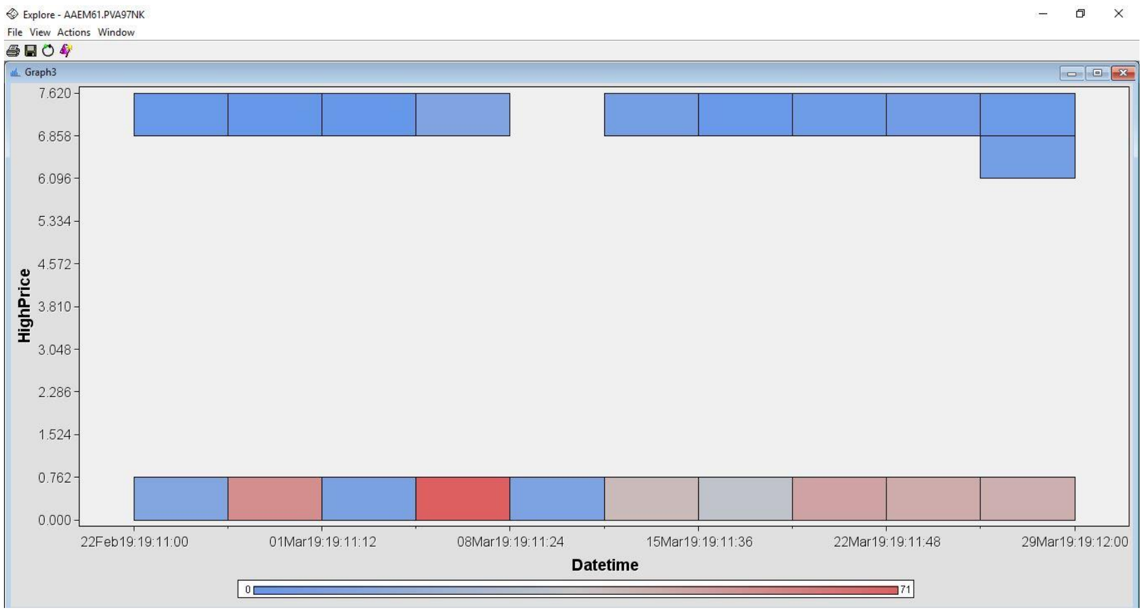
The measurement of the retrieved data are categorical and historical which the category is done by the company name and its stock market price, and the history is looked on the tweets based on the tweet dates and the mapped stock price date.

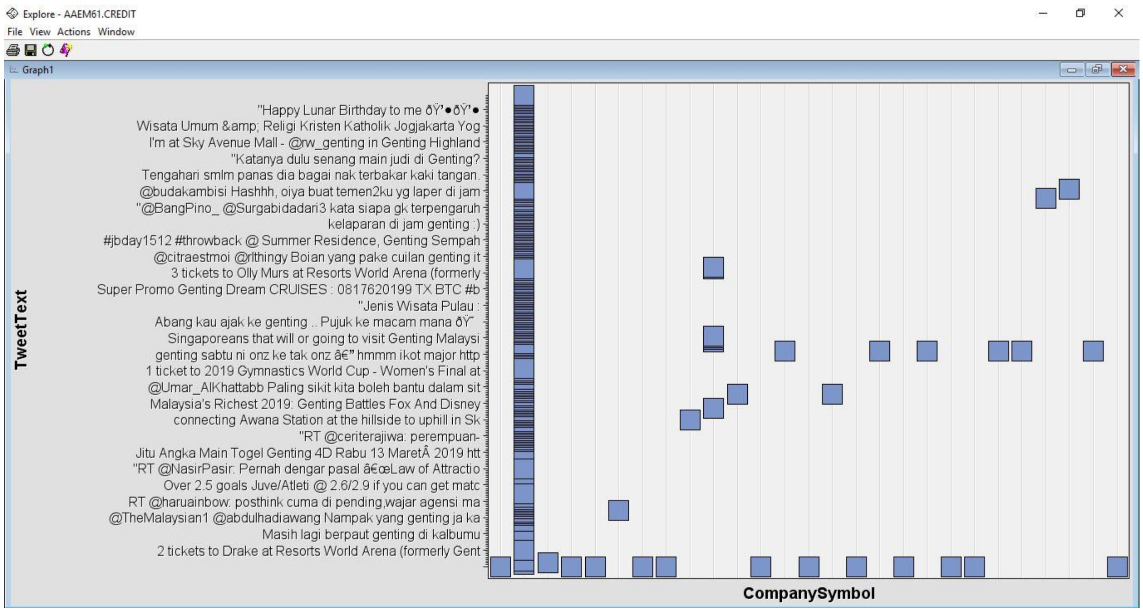
The diagrams used in this studies are histogram for quantitative data and scatter plot for qualitative data which were drawn using SAS Enterprise Miner. The first chosen company is Air Asia. In the diagram below, it is clear that the prices are unstable; dropping and increasing without a direct pattern. Comparing the possibility of this being affected by number of relevant tweets, the company was not tweeted about much by Twitter users during the mapped date and time.





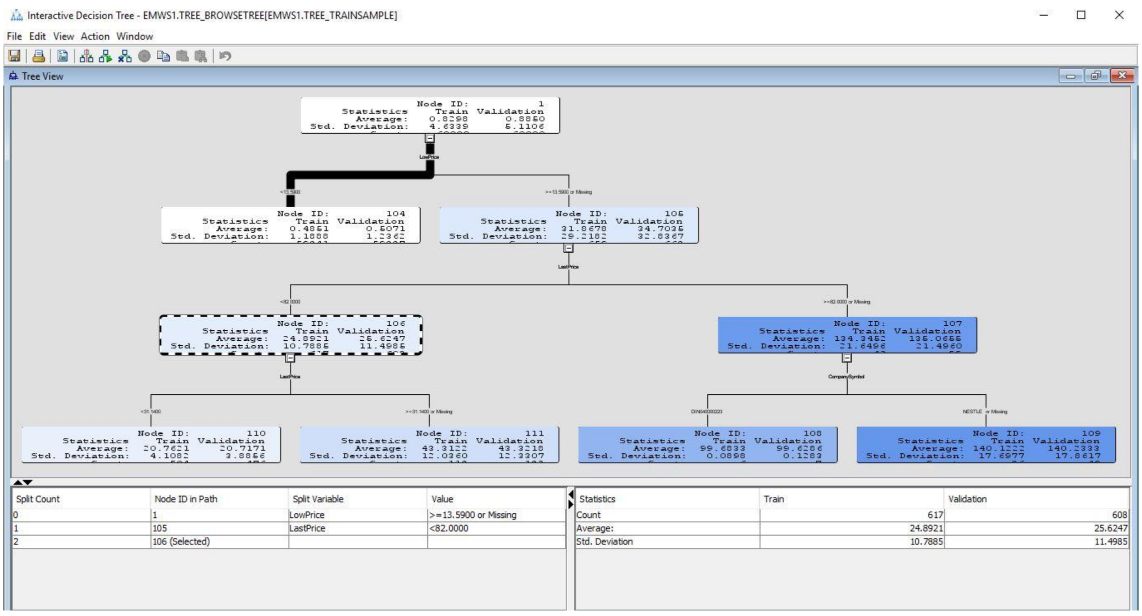
However, when comparing with Genting, it is clear that Genting shows higher stock prices than Air Asia. This is supported by the more number of tweets being tweeted, retweeted and liked by Twitter users about Genting. Besides that, Genting's stock price is relatively stable at above the value 6 when compared to Air Asia's stock price (only at 3.23 maximum level) which showcase a higher value to Genting than AirAsia. In short, Genting is more popular and favourable to the public than Air Asia which could be the reason of it having much higher stock prices throughout the same duration than Air Asia.





"Happy Lunar Birthday to me ðŸ'•ðŸ'•
Wisata Umum &amp; Religi Kristen Katholik Jogjakarta Yog
I'm at Sky Avenue Mall - @rw_genting in Genting Highland
"Katanya dulu senang main judi di Genting?
Tengahari smlm panas dia bagai nak terbakar kaki tangan.
@budakambisi Hashhh, oiya buat temen2ku yg laper di jam
"@BangPino_ @Surgabidadari3 kata siapa gk terpengaruh
kelaparan di jam genting :)
#jbday1512 #throwback @ Summer Residence, Genting Sempah
@citraestmoi @rlthingy Boian yang pake cuilan genting it
3 tickets to Olly Murs at Resorts World Arena (formerly
Super Promo Genting Dream CRUISES : 0817620199 TX BTC #b
"Jenis Wisata Pulau :
Abang kau ajak ke genting .. Pujuk ke macam mana ðŸ˜
Singaporeans that will or going to visit Genting Malaysi
genting sabtu ni onz ke tak onz â€" hmmm ikot major http
1 ticket to 2019 Gymnastics World Cup - Women's Final at
@Umar_AlKhattabb Paling sikit kita boleh bantu dalam sit
Malaysia's Richest 2019: Genting Battles Fox And Disney
connecting Awana Station at the hillside to uphill in Sk
"RT @ceriterajiwa: perempuan-
Jitu Angka Main Togel Genting 4D Rabu 13 MaretÂ 2019 htt
"RT @NasirPasir: Pernah dengar pasal â€œLaw of Attractio
Over 2.5 goals Juve/Atleti @ 2.6/2.9 if you can get matc
RT @haruainbow: posthink cuma di pending,wajar agensi ma
@TheMalaysian1 @abdulhadiawang Nampak yang genting ja ka
Masih lagi berpaut genting di kalbumu
2 tickets to Drake at Resorts World Arena (formerly Gent

Next, the focus is on the communication of insights of data which is represented by decision tree that was drawn using SAS Enterprise Miner. For this process, the data are split to training and validation sets. The parent node drawn was the LowPrice which is interpreted as the most significant attribute for the analysis of this project.



Interactive Decision Tree - EMWS1.TREE_BROWSETREE[EMWS1.TREE_TRAINSAMPLE]
File Edit View Action Window
Tree View
Node ID: 1 Statistics Train Validation Average: 0.8298 0.8850 Std. Deviation: 4.6339 5.1106
LowPrice
<13.5900
>=13.5900 or Missing
Node ID: 104 Statistics Train Validation Average: 0.4851 0.5071 Std. Deviation: 1.1888 1.2362
Node ID: 105 Statistics Train Validation Average: 31.8678 34.7035 Std. Deviation: 29.2182 32.8367
LastPrice
<82.0000
>=82.0000 or Missing
Node ID: 106 Statistics Train Validation Average: 24.8921 25.6247 Std. Deviation: 10.7885 11.4985
Node ID: 107 Statistics Train Validation Average: 134.3452 135.0666 Std. Deviation: 21.6496 21.4960
LastPrice
CompanySymbol
<31.1400
>=31.1400 or Missing
DIN040000223
NESTLE or Missing
Node ID: 110 Statistics Train Validation Average: 20.7621 20.7171 Std. Deviation: 4.1082 3.8856
Node ID: 111 Statistics Train Validation Average: 43.3122 43.3218 Std. Deviation: 12.0360 12.3307
Node ID: 108 Statistics Train Validation Average: 99.6833 99.6286 Std. Deviation: 0.0898 0.1283
Node ID: 109 Statistics Train Validation Average: 140.1222 140.2333 Std. Deviation: 17.6977 17.8617

Split Count	Node ID in Path	Split Variable	Value
0	1	LowPrice	>=13.5900 or Missing
1	105	LastPrice	<82.0000
2	106 (Selected)		

Statistics	Train	Validation
Count	617	608
Average:	24.8921	25.6247
Std. Deviation	10.7885	11.4985

CONCLUSION (USER RECOMMENDATIONS)

Based on the analysis and the findings, it is safe to recommend Genting to potential investors who look forward to invest in a good and stable company as it showcased a rather stable performance during the project timeline. Genting has potential to provide excellent profits and return of investments to investors. This is proved in the analysis section where it is popular and more favourable among the public with its stable stock price.

PROJECT LINKS

Presentation Video:

Disclaimer: This video is only shared with people with the link.

https://drive.google.com/open?id=1BzkrTthp4C9wuanNDlO1z1A40GBywGTK

Github:

https://github.com/nuramiramansor/wqd180094_datamining

APPENDIX – SOURCE CODE

TheStar Company Names extractor

```
rows = document.querySelectorAll('#marketwatchtable tr.linedlist');
companies = []
last = []
for (var i=0; i<rows.length; i++) {
    companies.push(rows[i].childNodes[0].innerText);
}

str = ""
for (var j=0; j<companies.length; j++) {
    str = str + "'" + companies[j] + "', ";
}
```

Please check on the folder ”thestar” and filename company_names.txt for the output of this script; complete company name list

TheStar Scraper Code

```
import requests
from bs4 import BeautifulSoup
import urllib
import time
import os

script_dir = os.path.dirname(os.path.realpath(__file__))
os.chdir(script_dir)

base_url =
'https://www.thestar.com.my/business/marketwatch/stocks/?qcounter={}'

# reading the company names into a list
company_names = []
with open("company_names.txt") as f:
    for line in f:
        company_names.append(line.strip())

company_open_prices = []
company_high_prices = []
company_low_prices = []
company_last_prices = []
update_datetimes = []

error_log_path = script_dir + '/error_log.txt'
error_log = open(error_log_path, 'a')

# we will scrape the last price and the update data and time
# for each company
for c in company_names:
    # create the url for the company after encoding the company
    # name to be valid for the url
    url = base_url.format(urllib.parse.quote(c))
    print('Scraping: ' + url)
    # download the web page of the company and get its HTML contents
    headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X '
               '10_14_3) AppleWebKit/537.36 (KHTML, like Gecko) '
               'Chrome/72.0.3626.109 Safari/537.36'}
    try:
        html_doc = requests.get(url, headers=headers).text
        # parse the HTML contents using BeautifulSoup parser
        soup = BeautifulSoup(html_doc, 'html.parser')
        # get the open price of the company
        open_price =
soup.select_one('#slcontent_0_ileft_0_opentext').text
        # get the high price of the company
        high_price =
soup.select_one('#slcontent_0_ileft_0_hightext').text
        # get the low price of the company
        low_price =
soup.select_one('#slcontent_0_ileft_0_lowtext').text
        # get the last price of the company
        last_price =
soup.select_one('#slcontent_0_ileft_0_lastdonetext').text
        # get the update date of the data
        update_date =
soup.select_one('#slcontent_0_ileft_0_datetxt').text
        # get the update time of the data
        update_time =
soup.select_one('#slcontent_0_ileft_0_timetxt').text
    except:
        error_log.write("{}, {}\n".format(c, time.strftime('%d-%b-
%Y_%H-%M')))

    # add the values to the corresponding lists
    company_open_prices.append(open_price)
    company_high_prices.append(high_price)
    company_low_prices.append(low_price)
    company_last_prices.append(last_price)
    update_datetimes.append(update_date + ' ' + update_time)

error_log.close()

# save the scraped prices to a file whose name contains the
# current datetime
file_name = 'prices_' + time.strftime('%d-%b-%Y_%H-%M') + '.txt'
with open(file_name, 'w') as f:
    for c, opp, hip, lop, lap, u in zip(company_names,
company_open_prices,
    company_high_prices, company_low_prices, company_last_prices,
update_datetimes):
        f.write(c + ' : ' + opp + ' : ' + hip + ' : ' + lop + ' : ' +
lap + ' : ' + ' [' + u + ']' + '\n')
```

Please check on the folder ”thestar” > “output” for the output of this scraping script; stock exchange data

MalaysiaStock Scraper Code

```
import requests
from bs4 import BeautifulSoup
import urllib
import time
import os
import re

os.chdir(os.path.dirname(os.path.realpath(__file__)))

base_url = 'https://www.malaysiastock.biz/Market-
Watch.aspx?type=A&value={}'

company_names = []
company_open_prices = []
company_high_prices = []
company_low_prices = []
company_last_prices = []
update_dates = []

# we will scrape the last price and the update data and time
# for each company
for c in "ABCDEFGHIJKLMNOPQRSTUVWXYZ0":
    # create the url for the company after encoding the company
    # name to be valid for the url
    url = base_url.format(c)
    # download the web page of the company and get its HTML contents
    headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X '
               '10_14_3) AppleWebKit/537.36 (KHTML, like Gecko) '
               'Chrome/72.0.3626.109 Safari/537.36'}
    html_doc = requests.get(url, headers=headers).text
    # parse the HTML contents using BeautifulSoup parser
    soup = BeautifulSoup(html_doc, 'html.parser')
    company_links = soup.select("tr > td.alignLeft > span > a")
    for co in company_links:
        co_url = "https://www.malaysiastock.biz/" + co.get('href')
        print('Scraping', co_url)
        co_name = co.text
        co_html_doc =  requests.get(co_url, headers=headers).text
        co_soup = BeautifulSoup(co_html_doc, 'html.parser')
        last_price =
co_soup.select_one('#MainContent_lbQuoteLast').text
        open_price =
co_soup.select_one('#MainContent_lbQuoteOpen').text
        price_range =
co_soup.select_one('#MainContent_lbDayRange').text
        low_price, high_price = price_range.split('-')
        update_date = co_soup.find("div", string=re.compile("Market
Date")).text
        company_names.append(co_name)
        company_open_prices.append(open_price)
        company_high_prices.append(high_price)
        company_low_prices.append(low_price)
        company_last_prices.append(last_price)
        update_dates.append(update_date)
```

```
print(len(company_high_prices), len(company_names))

# save the scraped prices to a file whose name contains the
# current datetime
file_name = 'malaysiastock_prices_' + time.strftime('%d-%b-%Y_%H-%M')
+ '.txt'
with open(file_name, 'w') as f:
    for c, opp, hip, lop, lap, u in zip(company_names,
company_open_prices,
    company_high_prices, company_low_prices, company_last_prices,
update_dates):
        f.write(c + ' : ' + opp + ' : ' + hip + ' : ' + lop + ' : ' +
lap + ' : ' + ' [' + u + ']' + '\n')
```

Please check on the folder ”msiastock” > “output” for the output of this scraping script; stock exchange data

<u>MalaysianIndices Scraper Code</u>

```
import requests
from bs4 import BeautifulSoup
import time

#Website to get the indices
base_url = 'https://www.investing.com/indices/malaysia-indices?'

print('Scraping: ' + base_url)
headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X '
           '10_14_3) AppleWebKit/537.36 (KHTML, like Gecko) '
           'Chrome/72.0.3626.109 Safari/537.36'}
html_doc = requests.get(base_url, headers=headers).text
        # parse the HTML contents using BeautifulSoup parser
soup = BeautifulSoup(html_doc, 'html.parser')

#KLCI
indiceKLCI = soup.select_one('#pair_29078 >
td.bold.left.noWrap.elp.plusIconTd > a').text
LastA = soup.select_one('#pair_29078 > td.pid-29078-last').text
LastA = LastA.replace(",","")
HighA = soup.select_one('#pair_29078 > td.pid-29078-high').text
HighA = HighA.replace(",","")
LowA = soup.select_one('#pair_29078 > td.pid-29078-low').text
LowA = LowA.replace(",","")

#Malaysia ACE
indiceMalaysiaACE = soup.select_one('#pair_29075 >
td.bold.left.noWrap.elp.plusIconTd > a').text
LastB = soup.select_one('#pair_29075 > td.pid-29075-last').text
LastB = LastB.replace(",","")
HighB = soup.select_one('#pair_29075 > td.pid-29075-high').text
HighB = HighB.replace(",","")
LowB =  soup.select_one('#pair_29075 > td.pid-29075-low').text
LowB = LowB.replace(",","")
```

```
#FTSE BM Mid 70
indiceFTSEBMMid70 = soup.select_one('#pair_29076 >
td.bold.left.noWrap.elp.plusIconTd > a').text
LastC = soup.select_one('#pair_29076 > td.pid-29076-last').text
LastC = LastC.replace(",","")
HighC = soup.select_one('#pair_29076 > td.pid-29076-high').text
HighC = HighC.replace(",","")
LowC = soup.select_one('#pair_29076 > td.pid-29076-low').text
LowC = LowC.replace(",","")

#Malaysia Top 100
indiceMalaysiaTop100 = soup.select_one('#pair_29077 >
td.bold.left.noWrap.elp.plusIconTd > a').text
LastD = soup.select_one('#pair_29077 > td.pid-29077-last').text
LastD = LastD.replace(",","")
HighD = soup.select_one('#pair_29077 > td.pid-29077-high').text
HighD = HighD.replace(",","")
LowD = soup.select_one('#pair_29077 > td.pid-29077-low').text
LowD = LowD.replace(",","")

indice_name = [indiceKLCI, indiceMalaysiaACE,
                   indiceFTSEBMMid70, indiceMalaysiaTop100]
Last = [LastA, LastB, LastC, LastD]
High = [HighA, HighB, HighC, HighD]
Low = [LowA, LowB, LowC, LowD]
Time = [time.strftime('%H:%M'),time.strftime('%H:%M') ,
        time.strftime('%H:%M'), time.strftime('%H:%M')]
Date = [time.strftime('%d-%b-%Y'),time.strftime('%d-%b-%Y'),
        time.strftime('%d-%b-%Y'),time.strftime('%d-%b-%Y')]

# save the scraped prices to a file whose name contains the
# current datetime
file_name = 'indices_' + time.strftime('%d-%b-%Y_%H-%M') + '.csv'
with open(file_name, 'w') as f:
    for A, B, C, D, G, H in zip(indice_name, Last, High,
    Low, Date, Time):
        f.write(A + ',' + B + ',' + C + ',' + D + ',' +  '[' + G +
'|' + H + ']' + '\n')
```

<u>PDF Extraction of Company's Quarterly Report Code</u>

```
import PyPDF2
import re

pdfName = 'C:\\Users\\Amira\\Dropbox\\_Master of Data Science -
UM\\WQD7005 - Data Mining\\Assignment\\Milestone 2\\wip\\KESMI.pdf'
reader = PyPDF2.PdfFileReader(pdfName)
```

```
search_items = []
for i in range(reader.getNumPages()):
    page = reader.getPage(i)
    page_content = page.extractText()

    pattern = 'Revenue([\s\S]*?)Other items of income'
    result = re.findall(pattern, page_content, re.MULTILINE)
    search_items.append(result)

clean_data = []
data = search_items[0][0].split('\n')
for i in data:
    if i.strip() != '':
        clean_data.append(i.strip().strip('\(%\)').replace(',',''))

names = ['Current Year Quarter','Preceding Year Quarter','%
Change','Current Year to Date','Preceding Year to Date','% Change']

file = open("pdf-demo-output.txt","a+")
file.write(','.join(names)+'\n')
file.write(','.join(clean_data))
file.close()



import requests
from bs4 import BeautifulSoup
import urllib
import time
import os
import re

base_url = 'https://www.malaysiastock.biz/Market-
Watch.aspx?type=A&value={}'


for letter in "ABCDEFGHIJKLMNOPQRSTUVWXYZ0":
    url = base_url.format(letter)
    headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X '
               '10_14_3) AppleWebKit/537.36 (KHTML, like Gecko) '
               'Chrome/72.0.3626.109 Safari/537.36'}
    html_doc = requests.get(url, headers=headers).text

    main_soup = BeautifulSoup(html_doc, 'html.parser')
    company_links = main_soup.select("tr > td.alignLeft > span > a")
    for co in company_links:
        co_url = "https://www.malaysiastock.biz/" + co.get('href')
        print('Scraping', co_url)
        co_name = co.text
        co_html_doc =  requests.get(co_url, headers=headers).text
        soup = BeautifulSoup(co_html_doc, 'html.parser')

        #find table of financial reports
        test = soup.find_all('table', id="MainContent_gvReport")
        test_string = '>(.+?)</td'
        try:
            result = re.findall(test_string, str(test[0]))

            #extract variables
            try:
                for i in range(0,500,10):
                    date = result[i]
                    financial_year = 
re.findall('>(.+?)$',result[i+1])[0]
                    financial_quarter = 
re.findall('>(.+?)$',result[i+3])[0]
                    revenue = 
re.findall('>(.+?)$',result[i+4])[0].replace(',','')
                    pbt = 
re.findall('>(.+?)$',result[i+5])[0].replace(',','')
                    net_profit = 
re.findall('>(.+?)$',result[i+6])[0].replace(',','')
                    eps = re.findall('>(.+?)$',result[i+7])[0]
                    divident = re.findall('>(.+?)$',result[i+8])[0]
                    nta = re.findall('>(.+?)$',result[i+9])[0]

                    row = co_name + ',' + date + ',' + financial_year 
+ ',' + financial_quarter + ',' + revenue + ',' + pbt + ',' + 
net_profit + ',' + eps + ',' + divident + ',' + nta
                    file = open("output.txt","a+")
                    file.write(row+'\n')
                    file.close()
            except:
                print('End of table reached!')
        except:
            print('Financial report not available for this company!')
```

Twitter Scraper Code

```
import twitter
import pandas as pd
import numpy as np
import os
from datetime import datetime, timedelta
import time
import re
import sys

script_dir = os.path.dirname(os.path.realpath(__file__))
os.chdir(script_dir)

api = twitter.Api(consumer_key='bg47VgnvB83qypdhR2d7w4bsP',

consumer_secret='LGOaENfzRsUEhuESTOVysDt6I7iKK6Lub4pu7eXxT748lSvlh0',
                  access_token_key='329665949-
LTYBPi2PJFNQ9BbxwzECbeSSnlD8fxxb3ys8Afoi',

access_token_secret='5E4EK3xapVTQfCVMfQuug0cXyFDJJfgVeVvHBIvC3mPrF',
                  tweet_mode='extended')
```

```
if not (os.path.exists('symbol_tweets.csv')
        and os.path.exists('fullname_tweets.csv')):
    symbol_tweets_df = pd.DataFrame(columns=['company_symbol',
'tweet_text', 'favorite_count', 'created_at', 'hashtags',
                                             'tweet_id', 'lang',
'urls', 'user_followers', 'user_screen_name', 'crawling_time'])
    fullname_tweets_df = pd.DataFrame(columns=['company_fullname',
'tweet_text', 'favorite_count', 'created_at', 'hashtags',
                                               'tweet_id', 'lang',
'urls', 'user_followers', 'user_screen_name', 'crawling_time'])
else:
    symbol_tweets_df = pd.read_csv('symbol_tweets.csv',
engine='python')
    fullname_tweets_df = pd.read_csv('fullname_tweets.csv')

company_symbols = []
with open("company_symbols.txt") as f:
    for line in f:
        company_symbols.append(line.strip())

company_fullnames = []
with open("company_fullnames.txt") as f:
    for line in f:
        company_fullnames.append(line.strip())

# search for tweets since 2 days before today until yesterday
# search_start = (datetime.today() - timedelta(days=2)).strftime('%Y-
%m-%d')
# search_end = (datetime.today() - timedelta(days=1)).strftime('%Y-%m-
%d')
# sym missing (2019-03-10 -> 2019-03-11, )
# search_start = '2019-03-10' #first
# search_end = '2019-03-11'
search_start = '2019-03-16'
search_end = '2019-03-17'



if len(sys.argv) == 2:
    sym_or_fn = sys.argv[1]
else:
    sys.exit()


def crawl_tweets(sym_or_fn, df):
    """
    sym_or_fn should be equal to 'sym' to crawl tweets that contain
company
    symbols or equal to 'fn' to crawl tweets that contain company full
names
    """
    print('Crawling tweets containing company',
          'symbols' if sym_or_fn == 'sym' else 'full names',
          time.strftime('%H:%M'))
    if sym_or_fn == 'sym':
        company_labels = company_symbols
    else:
        company_labels = company_fullnames
        # Since company full formal names like 'AIRASIAC59: CW AIRASIA 
GROUP BERHAD (MACQ)'
        # for example will not be used usually in tweets, we will 
apply some
        # operations to them so the name above becomes 'CW AIRASIA 
GROUP'
        company_labels = 
[re.sub(r'(.*:\s*)|(\s*\(.*\))|(\s*BERHAD)|(\s*BHD)', '', com_fn)
                          for com_fn in company_labels]
    num_of_tweets_added = 0
    for com_lb in company_labels:
        print(com_lb, end=', ', flush=True)
        tweets = api.GetSearch(
            term=com_lb,
            since=search_start,
            until=search_end,
            count=100)
        time.sleep(9)
        for twt in tweets:
            twt = twt.AsDict()
            tweet_text = twt.get('full_text', '')
            favorite_count = twt.get('favorite_count', 0)
            created_at = twt.get('created_at', '')
            hashtag_list = twt.get('hashtags', [])
            hashtags = []
            for d in hashtag_list:
                h = d.get('text', '')
                if h:
                    hashtags.append(h)
            hashtags = ','.join(hashtags)
            tweet_id = twt.get('id_str', '')
            lang = twt.get('lang', '')
            url_list = twt.get('urls', [])
            urls = []
            for d in url_list:
                u = d.get('expanded_url', '')
                if u:
                    urls.append(u)
            urls = ','.join(urls)
            user = twt.get('user', {})
            user_followers = user.get('followers_count', 0)
            user_screen_name = user.get('screen_name', '')
            crawling_time = time.strftime('%Y-%m-%d|%H:%M')
            if re.search(pattern=r'\b' + com_lb.lower() +
                         r'\b', string=tweet_text.lower()):
                num_of_tweets_added += 1
                df.loc[df.shape[0], :] = \
                    (com_lb, tweet_text, favorite_count, created_at, 
hashtags,
                     tweet_id, lang, urls, user_followers, 
user_screen_name, crawling_time)
    print('Finished crawling...', time.strftime('%H:%M'))
    print('Added', num_of_tweets_added, 'to the dataset')
    if sym_or_fn == 'sym':
        df.to_csv('symbol_tweets.csv', index=False)
    else:
        df.to_csv('fullname_tweets.csv', index=False)


if sym_or_fn == 'sym':
    crawl_tweets(sym_or_fn, symbol_tweets_df)
else:
    crawl_tweets(sym_or_fn, fullname_tweets_df)
```

<u>Extra Effort: Cryptocurrency Scraper Code</u>

Please check on the folder ”cryptocurrencty” > “crypto-list.txt” for the cryptocurrencies

```
from lxml import html
import requests
import re
import time
import sys
from datetime import datetime

class Worker:
    def __init__(self, code):
        self.code = code

    def execute(self):
        link="https://www.coinbase.com/price/" + self.code
        start_page = requests.get(link)
        tree = html.fromstring(start_page.text)

        # implemented in a retry loop since some websites have an
initial loading screen
        for x in range(3):
            try:
                usd_price =
tree.xpath('/html/body/div[1]/div/div/main/div/div/div[3]/div[1]/div[1
]/div[1]/div[1]/span/text()')[0].strip()
                crypto_name =
tree.xpath('/html/body/div[1]/div/div/main/div/div/div[2]/div[1]/div/d
iv/h1[1]/text()')[0].replace(' Price','')
                symbol =
tree.xpath('/html/body/div[1]/div/div/main/div/div/div[2]/div[1]/div/d
iv/h1[2]/text()')[0]
                break;
            except:
                print("An error occurred while crawling.")
                return

        usd_price = usd_price.replace(',','').replace('$','')
        symbol = re.sub('[^a-zA-Z]+', '', symbol)
        ts = datetime.now()

        row = usd_price + ',' + crypto_name + ',' + symbol + ',' +
ts.strftime("%Y-%m-%d %H:%M:%S.%f")

        file =
open("C:\\Users\\Amira\\Documents\\temp\\output.txt","a+")
        file.write(row+'\n')
        file.close()

        print("Inserted -> " + row)

        return


class Handler:
    def __init__(self, path):
        self.path = path

    def execute(self):
        file = open(self.path,'r')
        lines = file.readlines()
        for line in lines:
            worker = Worker(line.strip())
            worker.execute()

        return

handler = Handler('C:\\Users\\Amira\\Documents\\temp\\coinbase-
list.txt');
handler.execute()
```

Please check on the folder ”cryptocurrencty” > “output_sample.csv” for the output of this scraping script; cryptocurrency data