DL3

FYP Final Report

A System for Predicting and Explaining Stock Price Prediction

by

Chen, Liang-Yu, Fan, Ziqian, and Shao, Yuming

DL3

Advised by

Prof. Dik-Lun LEE

Submitted in partial fulfillment

of the requirements for COMP 4900, CPEG 4901

in the

Department of Computer Science

and

Department of Computer Engineering Program

The Hong Kong University of Science and Technology

2019-2020

Date of submission: February 28, 2020

Table of Contents

1 Introduction

1.1 Overview

The performance of the stock market has been studied by economists since its birth in the 17th century [1]. After years of effort, researchers have found lots of factors and formulas that may help with the prediction of the stock market, like Technical Factor and Fundamental Factors [2]. But, in practice, the stock price prediction needs analysts to collect and analyze lots of information that covers a large number of aspects from historical price to the public mood. Because of the complexity of the stock market and large quantity of data, making accurate prediction in the traditional way is still an extremely difficult work that asks for rich experience and sharp instincts. Every year, companies spend a considerable amount of money hiring experts to analyze the information. However, retail investors who do not have enough ability to do that have to take great risk if they invest in stock market. Nowadays, with the help of advanced computing technology, complex relationship and large amount of data are no longer a problem but a feature that we can make use of. The task of stock price prediction perfectly accords with the advantage of big-data technologies. Therefore, a computer-based stock price prediction system is obviously needed to develop the prediction of stock market.

1.2 Objective

This project is to develop a program that can automatically collect information from internet, analyze the complex information, and predict the movement of stock market. We hope that it can help improving the working efficiency of financial practitioner and provide retail investors inexpensive reference that helps them with their decision and financing.

This project, tentatively named SI (Smart Investment) would be made up of 4 modules:

1. Web Crawler
2. Web Backend/Frontend
3. Sentiment Analysis Module
4. Prediction Module

The crawler is used to collect two types of information from Internet. One is the historical price of each stock. The other is textual data like financial news that may have the influence on stock price. As the amount of data collected by the crawler is large, it would be stored in a JSON file for further maintainence. After that, we will then analyze the sentiment scores for the financial news, including the score of its title and its description. Later we will combine them with numerical data scrawled from the financial API as our complete training dataset. After preprocessing our data, we will go through a systematic approach to evaluate the model we trained, and after obtaining the best model with best performance, we will then store the model in the backend server, which their prediction results will be transmitted whenever the API are called.

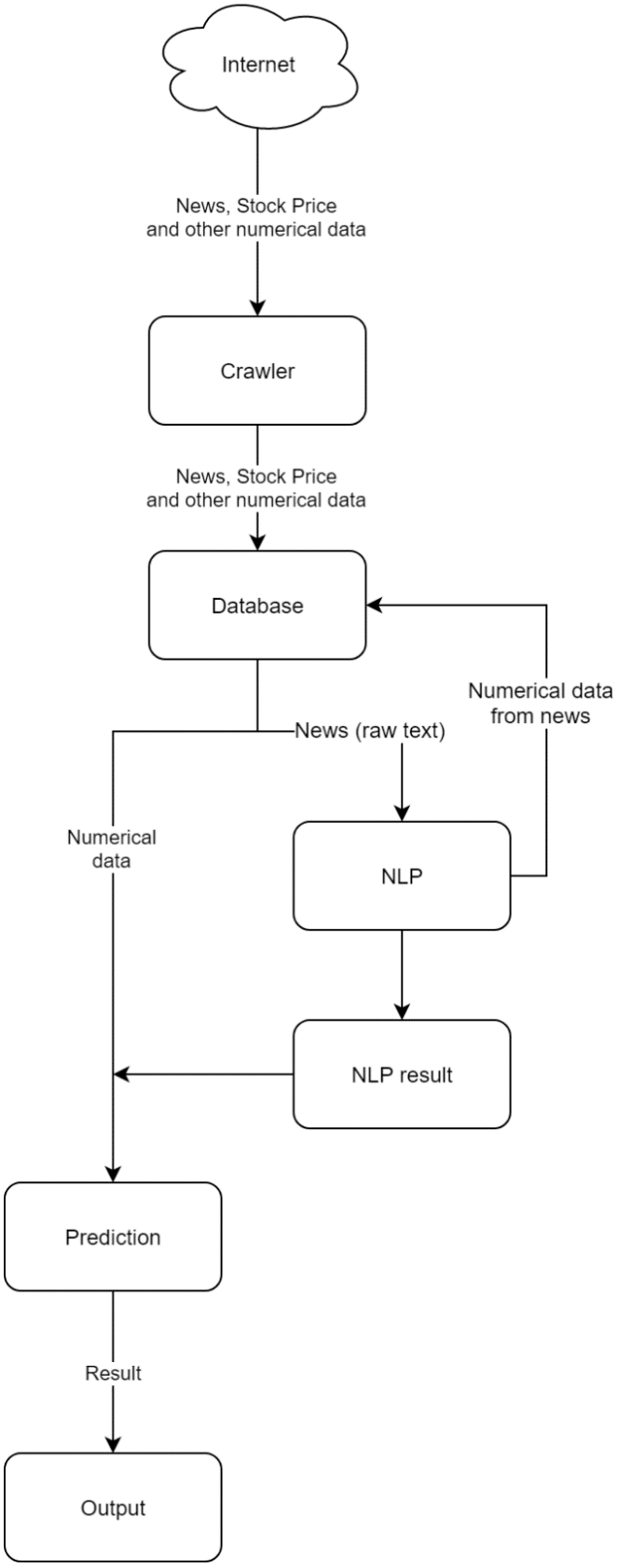


Figure 1. Flow Diagram

1.3 Literature Survey

There have been many prior works to address the prediction of stock market. Some of them provide valuable inspiration about the design of this project.

1.3.1 Random Forest model

Random Forest is a simple but powerful ensemble learning method that generates a large number of random decision trees and trains the weight of each tree. As there are relatively fewer parameters to train in random forest, it gets a considerably high training efficiency. It has been proved that with the same input (open, close, high and low price), random outperforms masny other prediction models with relatively short training time (Neural Network, Naïve Bayes and Support Vector Machine) [3]. Therefore, Random Forest could be a suitable model to start with. The detailed plan would be explained in the Methodology section.

1.3.2 LSTM model

Long short-term memory (LSTM) is a more complicated and accurate model widely used in time series prediction like stock market prediction. It uses memory cell instead of traditional neurons in the hidden layer. For which it is able to associate memories and inputs in time, hence suitable for the analysis of data over time. Some successful projects are found in our survey that achieve great accuracy using LSTM [4]. But their inputs are limited to 5 stock price records (high, low, open, close and volume). If more factors are considered, it is high possible that a better accuracy can be achieved.

1.3.3 Reinforcement learning

Reinforcement learning is a paradigm of machine learning that doesn't need labeled data. Instead, it focuses on the balance between the exploration of uncharted territory and exploitation of current knowledge. Instead of using label, reinforcement learning judges the performance of each behavior by giving positive or negative reinforcement. In this project, we can train the model by judging the behavior of predicting a rise or fall of stock price and give a positive reinforcement if the prediction is correct.

1.3.4 Public mood analysis

The mood of the public is an important influence factor in the stock market. The public mood can be reflected in many methods like news, consumption, and social media. It has been proven that mood factors derived from Twitter tweets do improves the performance of a DJIA prediction [5].

In that project Twitter tweet, two software packages, OpinonFinder and GPMOS are used to model the public mood in many dimensions. OpinonFinder contains several words that are labeled either positive or negative. By counting the appearance of those words, a score can be given to each tweet. GPMOS is a more complex tool that measures human mood states in terms of 6 different mood dimensions by counting 4- or 5-grams. After that, the dimensions that relates most to DJIA are selected and inputted to the model. As a result, there is a significant rise in the accuracy of the model. However, a relatively simple prediction model that takes few financial data is applied in this projection. It values a lot to try to combine the public mood measurement with a more complex and accurate model.

1.3.5 Breaking Financial News

Social event, especially financial events, also significantly influences the performance of the stock market. So, breaking financial news deserves our attention. There has been a prior work that collects and analyzes breaking financial news to predict the motion of stock price [6]. The raw text of financial news is analyzed by the method of Bag of Words, Noun Phrasing, and Named Entities. Bag of Words is a way to make raw text trainable by counting the appearance of each word. Although it loses some information as the order of words is ignored, it can still macroscopically represent the attitude of the author. Noun phrasing is accomplished by extracting the meaning of a noun or noun phrase throuth the lexicon and syntactic rules from the context. And entities are phrases that contain significant information like name, location or time point. Although the prior work does not build a whole stock price prediction system, it inspires us about the analysis of breaking news.

Those previous studies raises valuable potential method to improve the performance of stock marked prediction. But they only focus on one aspect of the problem. In contrast, our work is to integrate the achieve of all those works and add in our own innovate so as to create a practical stock price prediction system.

2 Methodology

2.1 Data Collection

As shown in Figure 1, we planned to build a stock prediction system, which starts from crawling the news from the web. After analyzing the news, we will than concatenate it's score with the stock price data and all other financial indicators derived from it. In the end, we will collect external features such as Gold price, S&P 500 index, 5/10 year treasury bond price to strengthen our training dataset

2.1.1 Stock Selection

Since the U.S. is the most liquid and biggest stock market in the world, and it is more sensitive to the world wide financial news, we decided to choose stocks from the U.S. as our historical stock data. Among all the U.S. stocks, S&P 500 are one of the most commonly followed equity indices, and many consider it to be one of the best representations of the U.S. stock market. Therefore, all of the stock we predict will be on the S&P500 list. However, since the list is always changing, we only target those stocks that exist on the list since 2009 in order to have enough training samples. One example is AAPL (Apple Inc.), and throughout all this project, we will show the prediction results of AAPL stock price as a demonstration.

```python
import pandas as pd
import numpy as np

import bs4 as bs
import pickle
import requests
import time
import lxml
from datetime import datetime

def get_sp500_tickers(date_added):
    resp = requests.get('http://en.wikipedia.org/wiki/List_of_S%26P_500_companies')
    soup = bs.BeautifulSoup(resp.text, 'html.parser')
    table = soup.find('table', {'class': 'wikitable sortable'})
    tickers = []
    for row in table.findAll('tr')[1:]:
        if row.findAll('td')[6].text == '':
            ticker = [row.findAll('td')[0].text.strip(), row.findAll('td')[6].text]
        else:
            if datetime.strptime(row.findAll('td')[6].text[0:10], '%Y-%m-%d') < date_added:
                ticker = [row.findAll('td')[0].text.strip(), row.findAll('td')[6].text]
            else:
                continue
        tickers.append(ticker)
    return tickers
```

```
307

[['A', '2000-06-05'],
 ['AAPL', '1982-11-30'],
 ['ABC', '2001-08-30'],
 ['ABT', '1964-03-31'],
 ['ADBE', '1997-05-05'],
 ['ADI', '1999-10-12'],
 ['ADM', '1981-07-29'],
 ['ADP', '1981-03-31'],
 ['ADSK', '1989-12-01'],
 ['AEE', '1991-09-19'],
 ['AEP', ''],
 ['AES', '1998-10-02'],
 ['AFL', '1999-05-28'],
 ['AGN', '1999-04-12'],
 ['AIG', '1980-03-31'],
 ['AIV', '2003-03-14'],
 ['AIZ', '2007-04-10'],
```

Figure 2. Acquire the Stocks that Exists Since 2009 and a Sample of the Stock List (Symbol, Add Date)

2.1.2 Crawling Financial News

By examining Google Finance, we've figured out that financial news are obtained by searching its market following by its symbol (i.e. NASDAQ: AAPL). Therefore, we've constructed a Google news crawlers to crawl daily top 5 news and store them in JSON for later processing.

```
USER_AGENT = "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:65.0) Gecko/20100101 Firefox/65.0"
headers={"user-agent" : USER_AGENT}
target='AAPL' # target symbol
keyword = quote(target.encode('utf8'))

total_search = {}
traind_date_list = pd.date_range(start="2009-12-30",end="2019-12-31").to_list()
start_time = time.time()

for i in traind_date_list:
    time.sleep(0.5) # prevent from anti crawling
    target_date = i.strftime("%m/%d/%Y")
    # google query parameters
    target_param = {
        "tbm": "nws", # search news
        "hl": "en",
        "lr": "lang_en", # in English
        "q": keyword, # search keyword
        "oq": keyword, # search keyword
        "dcr": "0",
        "source": "lnt",
        "num": 5, # top 5 news
        "tbs": "cdr:1,cd_min:"+target_date+",cd_max:"+target_date, # specific date
    }
    url = "https://www.google.com.tw/search?" + urlencode(target_param)
    res = requests.get(url, headers=headers)
    if res.status_code == 200:
        content = res.content
        soup = BeautifulSoup(content, "html.parser")
        search_list = []
        items = soup.findAll("div", {"class": "g"})
        if items:
            for index, item in enumerate(items):
                    # title
                    news_title = item.find("h3", {"class": "r"}).find("a").text
                    # url
                    href = item.find("h3", {"class": "r"}).find("a").get("href")
                    news_link = href
                    # content
                    news_text = item.find("div", {"class": "st"}).text
                    # source
                    news_source = item.find("h3", {"class": "r"}).findNext('div').text.split('-')
                    news_from = news_source[0]
                    time_created = str(news_source[1])
                    # add item into dict, and later store them in JSON
                    search_list.append({
                        "news_title": news_title,
                        "news_link": news_link,
                        "news_text": news_text,
                        "news_from": news_from,
                        "time_created": time_created
                    })
            total_search[target_date] = search_list
    else:
        print('error at '+str(i))
```

```
'12/30/2008': [
  {'news_title': 'Netflix, Adobe, and Google Rated Among Top Places to Work',
   'news_link': 'https://seekingalpha.com/article/112644-netflix-adobe-and-google-rated-among-top-places-to-work',
   'news_text': "Apple (NASDAQ:AAPL) is No. 19. Amazon (NASDAQ:AMZN) and Microsoft (NASDAQ:MSFT) didn't make
   the list. Neither did Yahoo (YHOO), but it escaped the...",
   'news_from': 'Seeking Alpha (blog)',
   'time_created': '30 Dec 2008'}
],
```

Figure 3. Acquire Financial News from Google and a Sample of the News Collected.

2.1.3 Stock Price Data Collection

Orginally, we thought maybe we should use trading data in minutes as our training dataset. However, since our goal is to provide predictions to common people, after meeting with the professor we decided to only focus on daily trading data. We retrieved the daily stock price from yfinance api, which includes Open, High, Low, Close, Volume, and Adj Close of the stock.

2.2 Data Preprocess

After acquiring the available stock list, financial news and stock price, next step we will focus on calculating fincial indicators, sentiment scores, and add more external data to form our final traing dataset. We also normalized and built up our validation dataset as well as testing dataset. In addition, we've created two kinds of training data set. The first type only contains those features that are derived after comparison (i.e. Ratio, Difference…), while the second type contains not only camparison features but also ***"absolute"*** features, such as real close price, real volume, real open price …etc.,

2.2.1 Financial Indicators

Here we introduce all financial indicators we calculated in a systematic way.

Change:

Percentage Change of close price of the interval.

Open and Close Price:

The following features derived from open and close price are used in the models:

- The open and close price
- The change of open and close price against previous day
- The ratio and difference of the open and close price during the interval
- The change of open price (in percentage) between the current day and 2 and 6 days before
- The ratio of the difference of open and close price to close price

High and Low Price:

The following features derived from the highest and lowest price during the interval are used in the models:

- The highest and lowest price
- The difference of the highest and lowest price
- The ratio of the difference of the highest and lowest price to close price

Volume of Trading:

The volume of stocks traded

- The change of the volume of stocks traded against today is used in the models

Adjusted Closing Price

Adjusted closing price is the close price after accounting dividend, stock split and so on which can reflect the true price of that stock.

KDJ Index

KDJ Index consist of 3 values: %K, %D and %J. They can be calculated by the foll owing:

$$RSV(n) = \frac{C_t - L_n}{H_n - L_n} * 100$$

Where n is 9 by default, C_t is the close price of the last of the n-day period, H_n and L_n are the highest and lowest price during the n-day period. And then,

$$\%K = \frac{2}{3} * \%K \text{ of previous day} * RSV$$

$$\%D = \frac{2}{3} * \%D \text{ of previous day} * \%K$$

$$\%J = 3 * \%D - 2 * \%K$$

Moving Average Convergence / Divergence (MACD)

MACD is derived from exponential moving average (EMA). The EMA of a n-day period can be calculated by the following:

The EMA of the first n-day period is the average price of the period. Then the EMA of close of the n-day period ended on day x can be calculated recursively by:

$$EMA_{(\text{close},n)} = \frac{2 * \text{close price on day x} + (n-1) * EMA_{(\text{close},n)} \text{ of day x-1}}{n+1}$$

Then, MACD can be derived from EMA by:

$$DIF = EMA_{(\text{close},12)} - EMA_{(close,26)}$$

$$MACD = EMA_{DIF,9}$$

$$OSC = DIF - MAD$$

DIF, MACD and OSC are all used as a feature in the models.

Relative Strength Index (RSI)

RSI can be calculated by the following:

If $close > close_{[-1]}$:

$$U = close - close_{[-1]}$$

$$D = 0$$

Else:

$$U = 0$$
$$D = close_{[-1]} - close$$

Using the formula of EMA introduced before:

$$RS = \frac{EMA_{U,n}}{EMA_{D,n}}$$

$$RSI = \left(1 - \frac{1}{1 + RS}\right) * 100\%$$

Where $n = 6$ and $n = 12$ are used respectively, which is corresponding to one and two weeks.

Williams %R

The purpose of Williams %R is to tell whether a stock or commodity market is trading near the high or the low, or somewhere in between, of its recent trading range. The %R of an n-day period can be calculated by:

$$\%R = \frac{H_n - C}{H_n - L_n} * 100\%$$

Where C is the close price at the end of the period, H_n and L_n are the highest and lowest price during the period. In the models, n = 6 and n = 12 are used respectively.

Commodity Channel Index (CCI)

CCI is used to identify overbought and oversold levels by measuring an instrument's variations away from its statistical mean. The CCI of a n-day period can be calculated by the following:

$$TP = \frac{H_n + L_n + C}{3}$$

$$MA = \frac{\sum_i^n \text{close price of the i-th day in the period}}{n}$$

$$MD = \frac{\sum_i^n (MA - \text{close price of the i-th day in the period})}{n}$$

$$CCI_n = \frac{TP - MA}{0.015 * MD}$$

Where C is the close price at the end of the period, H_n and L_n are the highest and lowest price during the period. And n = 14 is used in the models

Average True Range (ATR)

ATR is derived from true range (TR) which is used to indicate degree of price volatility. The ATR of day t can be calculated by:

$$TR = \max(H_t, C_{t-1}) - \min(L_t, C_{t-1})$$

$$ATR = EMA_{(TR,n)}$$

Where C_{t-1} is the close price of the last trading day, H_t and L_t is the highest and lowest price of day t.

Different of Moving Average (DMA)

DMA is the difference between short-term and long-term moving average. In this project, DMA is the difference between the average price of 10-day and 50-day period.

Volatility Ratio (VR)

VR is used to identify price patterns and breakouts from volume of trading. The VR of a n-day period can be calculated by:

$$VR = \frac{H_t - L_t}{ATR}$$

Where H_t and L_t is the highest and lowest price of the last day of the period and n = 26 is used in the models. The formula of ATR has been introduced before.

Bolinger Bands (BB)

Bolinger Bands contains an upper band (BOLU) and a lower band (BOLD). The Bolinger Bands during a n-day period can be calculated by:

$$BOLU = MA(TP, n) + m * \sigma[TP, n]$$

$$BOLD = MA(TP, n) - m * \sigma[TP, n]$$

Where $TP = \frac{H+L+C}{3}$, $MA(TP, n)$ is the mean of TP over the n-day period, C is the close price at the end of the period, H and L are the highest and lowest price during the period and $\sigma[TP, n]$ is the standard deviation over last n periods of TP. In the models, the value of n and m is the typical value which is $n = 20$ and $m = 2$. $BOLU$, $BOLD$, $MA(TP, n)$ and the difference between $MA(TP, n)$ and close price of the previous trading day are used as four features in the models.

Here is a summary of the indicators we used.

Indicator	Variable	Features
Price	open	Open price of the stock of the tarding day
	close	Close price of the stock of the tarding day
	high	Highest price of the stock on the tarding day
	low	Lowest price of the stock on the tarding day
	adj close	Adjusted closing price
Volume	volume	The volume of stocks traded
Price change	change	Change of price of the interval
	open_close_diff	The difference of open and close price
	high_low_diff	The difference of high and low price
	open_delta	The change of open and close price against previous day
	close_delta	The change of open and close price against previous day
Price ratio	close_-2_r, close_-6_r	The change of open price (in percent) between the current day and 2 days or one week before
	open_close_diff_ratio	The ratio of the difference of the open and close price to close price
	high_low_diff_ratio	The ratio of the difference of the highest and lowest price to close price
Trend	dma	Different of Moving Average on 10-day and 50-day timeframe
	kdjk	Fast stochastic oscillator (%K of KDJ index)
	kdjd	Slow stochastic oscillator (%D of KDJ index)
	kdjj	%J of KDJ index
	macd	MACD value
	macds	Value of signal line in MACD
	macdh	Value of divergence in MACD
	cci	Commodity channel index
Momentum	rsi_6, rsi_12	Relative strength index (RSI) on a 1- or 2-week timeframe
	wr_6 ,wr_12	Williams %R on a 1- or 2-week timeframe
Volatility	atr	Average true range (ATR)
	vr	Volatility Ratio
	boll_-1_d, boll_ub_-1_d, boll_lb_-1_d	Middle, upper and lower band of Bollinger Bands (BB) on the previous trading day

2.2.2 External Features

The following features are respectively calculated from the S&P 500 Index, gold price, and the price of 5-year and 10-year treasury bonds, using the same techniques mentioned earlier.

Indicator	Variable	Features
Price	open	Open price of the stock of the tarding day
	close	Close price of the stock of the tarding day
	high	Highest price of the stock on the tarding day
	low	Lowest price of the stock on the tarding day
	adj close	Adjusted closing price
Volume	volume	The volume of stocks traded
Price change	change	Change of price of the interval
	open_close_diff	The difference of open and close price
	high_low_diff	The difference of high and low price
	close_delta	The change of open and close price against previous day

*Gold, S&P 500, 5-year, 10-year bond will add prefix to variable with gold_, sp500_, y5_bond_, y10_bond_ respectively.

2.2.3 Sentiment Scores

Originally, we've planned to analyze the financial news using Google NLP api. However, due to the Internet firewall, two of our members had connection issues. Most importantly, it is charged according to band-width consumption. Therefore, we decided to use TextBlob to conduct the general sentiment analysis. We will obtain two sentiment scores, one for news title, and one for news description. We will then calculate the mean of the score for all the news of the day and add them as our new features, namely news_title_score and news_des_score.

```
with open('AAPL_news_09-19.json') as json_file:
    news_data = json.load(json_file)

time_index = list(stock_with_absolute.index)
score = {}
des_score = {}
for time in time_index:
    yesterday = (time - timedelta(days=1)).strftime("%m/%d/%Y")
    if(yesterday in news_data):
        num_news = len(news_data[yesterday])
        sentiment = 0
        des_sentiment = 0
        for news in news_data[yesterday]:
            news_title = news['news_title'].replace('...', '')
            news_des = news['news_text'].encode("ascii", "ignore").decode("ascii").replace('...', '')
            blob = TextBlob(news_title)
            des_blob = TextBlob(news_des)
            sentiment += blob.sentiment.polarity
            des_sentiment += des_blob.sentiment.polarity
        score[time] = sentiment / num_news
        des_score[time] = des_sentiment / num_news
    else:
        score[time] = 0.0
        des_score[time] = 0.0
des_score_series = pd.Series(des_score)
des_score_series.name = "news_des_score"
title_score_series = pd.Series(score)
title_score_series.name = "news_title_score"
stock_without_absolute = stock_without_absolute.join([title_score_series, des_score_series])
stock_with_absolute = stock_with_absolute.join([title_score_series, des_score_series])
```

Figure 4. Compute Sentiment Scores for News Title and Description

2.2.4 Label Construction

After finishing creating training data set, we then focused on creating label. There are two types of labels. The first type is the trend label. If the stock rises tomorrow, we will label it as 1 else 0; The second type we call it value label, which is the true rise/fall of the stock price. Each type contains three different periods, which are 1day, 7 days, and 30 days respectively. We use stockstats to calculate the value.

```
# Generate absolute label (namely 0,1) and non-absolute label (real price)
label_abs_1d = stock_data['close_1_d'].apply(lambda x: 1 if x > 0 else 0)
label_abs_7d = stock_data['close_7_d'].apply(lambda x: 1 if x > 0 else 0)
label_abs_30d = stock_data['close_30_d'].apply(lambda x: 1 if x > 0 else 0)

label_value_1d = stock_data['close_1_d']
label_value_7d = stock_data['close_7_d']
label_value_30d = stock_data['close_30_d']
```

Figure 5. Create Labels

2.2.5 Normalization

We use MinMaxScaler to normalize the data. We first fit_transform the training data, and transform the test data accordingly. Besides, we also randomly split the data into train/test/validation dataset using train_test_split()

```
X_train, X_test, y_train, y_test = train_test_split(train_data, label, test_size=0.1, random_state=42)
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Figure 6. Normalizing the data

2.2.6 Feature Summarization and Exploration

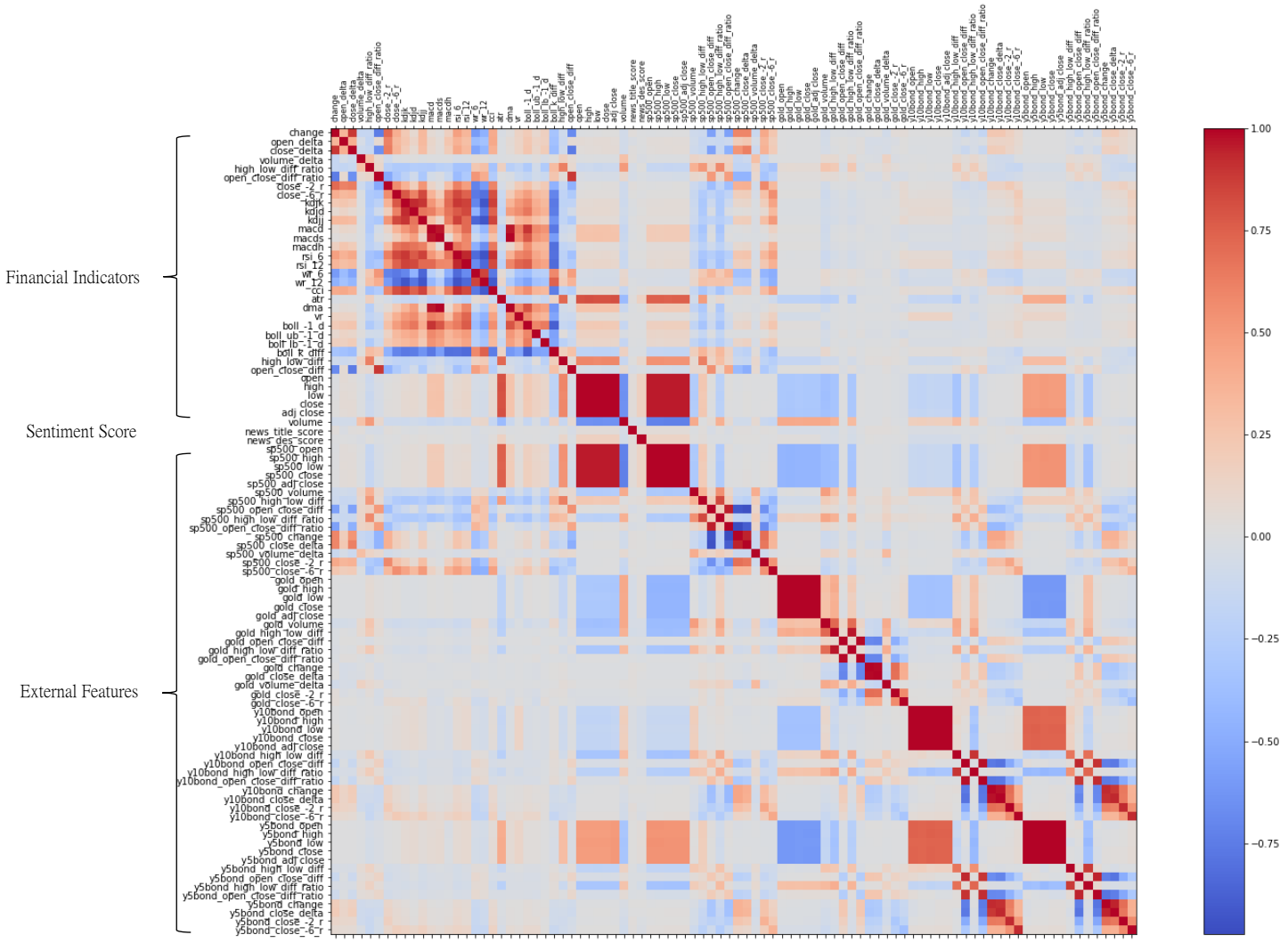


Figure 7. Correlation between Features

1. Financial Indicators have more correlation in between compare to its correlation with others
2. Description sentiment score has little correlation, while title sentiment has almost no correlation to other features.
3. Gold assets have negative correlation, which proves that gold is a great hedge asset and we believe the negative correlation could benefit the prediction.
4. Those “absolute ” features have a correlation near 1, meaning that they are not differentiable from each other, which may be useless when predicting. This observation may need further testing.

2.3 Model Implementation

2.3.1 Evaluate and Search Hyper Parameters

Before implementing the model, we need to first build up functions to systematically evaluate and search for hyper-parameters of these models. Here is our approach.

```
def create_grid_model(classifier, param_grid):
    cv = StratifiedShuffleSplit(n_splits=5, test_size=0.1)
    grid_model = GridSearchCV(classifier, param_grid=param_grid, cv=cv, n_jobs=-1,verbose=1, scoring='accuracy
    return grid_model
```

Figure 8. GridSearchCV model

```
def result(grid_model, train_data, predicted_test, test_label, predicted_train, train_label, file_name,
           decision_function, clf_name="Classifier"):
    print("Results for ", clf_name, ": ")
    print()
    print("The best parameters are %s" % (grid_model.best_params_))
    acc_train = accuracy_score(train_label, predicted_train)
    acc_test = accuracy_score(test_label, predicted_test)
    print("The Train Accuracy  %0.3f" % (acc_train))
    print("The Validation Accuracy   %0.3f" % (grid_model.best_score_))
    print("The Test Accuracy   %0.3f" % (acc_test ))

    if (clf_name[:6] == 'Random') | (clf_name == 'DecisionTree'):
        test_label_roc = np.zeros((len(test_label),2) )
        for i,v in enumerate(test_label):
            if v > 0.5:
                test_label_roc[i,1] = 1
            else:
                test_label_roc[i,0] = 1


    if (clf_name[:6] == 'Random') | (clf_name == 'DecisionTree'):
        print("AUC ROC : %0.3f" %( roc_auc_score(test_label_roc, decision_function) ))
    else:
        print("AUC ROC : %0.3f" %( roc_auc_score(test_label, decision_function) ))

    print("The mean training time of %f" % (np.mean(grid_model.cv_results_['mean_fit_time'], axis=0)) )
    print("The mean test time of %f" % (np.mean(grid_model.cv_results_['mean_score_time'], axis=0)) )
    # confusion matrix
    print("confusion matrix / precision recall scores")
    print ( confusion_matrix(test_label, predicted_test) )
    print ( classification_report(test_label, predicted_test) )

    #feature importance
    feats = {} # a dict to hold feature_name: feature_importance
    for feature, importance in zip(train_data.columns, grid_model.best_estimator_.feature_importances_):
        feats[feature] = importance #add the name/value pair
    importances = pd.DataFrame.from_dict(feats, orient='index').rename(columns={0: 'Gini-importance'})
    importances = importances.sort_values(by='Gini-importance', ascending=False)
    print(importances.head(7)) # print the top 7 features with greater importance
```

Figure 9. First Half of Our Model Evaluation Function

Figure 7. shows how we create the grid search model. Noticing that we apply StratifiedShuffleSplit as our cross-validator, it takes 10% of our training dataset and cross validaties for 5 folds.

Figure 8. shows how we evaluate the model performance. Noticing that this is only the first half of the function, the second half is about writing these data in .txt file as a backup according to the file_name input. By calling this function, we can have a clear

view of the model performance, including train/test/validation accuracy, AUC ROC score, confusion matrix, traing/test time, its recall, f-1 score … etc., If the classifier could produce feature importance, we would list it as well (top 7 features). For the following model implementation, all these detailed evaluations will be provided as reference.

2.3.2 Logistic Regression

Instruction

Logistic regression is a classification algorithm derived from linear regression by using Sigmoid function as the cost function. Logistic regression is very sensitive to linear relations. Therefore, it can be used to check for linear relations in stock prediction.

Hyperparameters

solver

Algorithm to use in the optimization problem.

In this project, 'newton-cg', 'lbfgs', 'liblinear', 'sag' and 'saga' are checked in grid search CV.

General Summarize

Grid search CV is done on this model for the 6 different tasks. The result is summarized in the following tables. Where open, close, high, low, adj close, and volume are defined as ‘absolute features’.

	With absolute features	Without absolute features
Predict 1-day trend	A1	B1
Predict 7-day trend	A2	B2
Predict 30-day trend	A3	B3

Table 1:Table of tasks

Task	A1	A2	A3
Train accuracy	0.57	0.615	0.67
Test Accuracy	0.559	0.583	0.729
AUC	0.5210	0.623	0.706
Mean Training Time	0.06265	0.17885	0.18992
Mean Test Time	0.00099	0.00063	0.00058

Table 2: Performance of the model with Absolute Features

Task	B1	B2	B3
Train accuracy	0.564	0.602	0.643
Test Accuracy	0.563	0.583	0.696
AUC	0.532	0.592	0.647
Mean Training Time	0.13621	0.10096	0.13521
Mean Test Time	0.00083	0.00063	0.00069

Table 3:Performance of the model without Absolute Features

Detailed Evaluation

A1

```
The best parameters are {'solver': 'newton-cg'}
The Train Accuracy 0.570
AUC ROC : 0.521
The Validation Accuracy 0.538
The Test Accuracy 0.559
[[91 50]
 [59 47]]
              precision    recall  f1-score   support

           0       0.61      0.65      0.63       141
           1       0.48      0.44      0.46       106

    accuracy                           0.56       247
   macro avg       0.55      0.54      0.54       247
weighted avg       0.55      0.56      0.56       247
```

A2

```
The best parameters are {'solver': 'newton-cg'}
The Train Accuracy 0.615
AUC ROC : 0.623
The Validation Accuracy 0.590
The Test Accuracy 0.583
[[125  12]
 [ 91  19]]
              precision    recall  f1-score   support

           0       0.58      0.91      0.71       137
           1       0.61      0.17      0.27       110

    accuracy                           0.58       247
   macro avg       0.60      0.54      0.49       247
weighted avg       0.59      0.58      0.51       247
```

B1

```
The best parameters are {'solver': 'lbfgs'}
The Train Accuracy 0.564
AUC ROC : 0.532
The Validation Accuracy 0.512
The Test Accuracy 0.563
[[93 48]
 [60 46]]
              precision    recall  f1-score   support

           0       0.61      0.66      0.63       141
           1       0.49      0.43      0.46       106

    accuracy                           0.56       247
   macro avg       0.55      0.55      0.55       247
weighted avg       0.56      0.56      0.56       247
```

B2

```
The best parameters are {'solver': 'lbfgs'}
The Train Accuracy 0.602
AUC ROC : 0.592
The Validation Accuracy 0.586
The Test Accuracy 0.583
[[129   8]
 [ 95  15]]
              precision    recall  f1-score   support

           0       0.58      0.94      0.71       137
           1       0.65      0.14      0.23       110

    accuracy                           0.58       247
   macro avg       0.61      0.54      0.47       247
weighted avg       0.61      0.58      0.50       247
```

A3

```
The best parameters are {'solver': 'liblinear'}
The Train Accuracy 0.670
AUC ROC : 0.706
The Validation Accuracy 0.644
The Test Accuracy 0.729
[[156  10]
 [ 57  24]]
              precision    recall  f1-score   support

           0       0.73      0.94      0.82       166
           1       0.71      0.30      0.42        81

    accuracy                           0.73       247
   macro avg       0.72      0.62      0.62       247
weighted avg       0.72      0.73      0.69       247
```

B3

```
The best parameters are {'solver': 'liblinear'}
The Train Accuracy 0.643
AUC ROC : 0.647
The Validation Accuracy 0.628
The Test Accuracy 0.696
[[165   1]
 [ 74   7]]
              precision    recall  f1-score   support

           0       0.69      0.99      0.81       166
           1       0.88      0.09      0.16        81

    accuracy                           0.70       247
   macro avg       0.78      0.54      0.49       247
weighted avg       0.75      0.70      0.60       247
```

Remark

According to the above results, we can find that:

1. The test accuracy and AUC are obviously lower than other models.

Conclusion

1. There are few linear relations in stock prediction tasks.
2. Logistic regression is not suitable for stock prediction

2.3.3 Decision Tree Classifier

Instruction

Decision Tree is a non-parametric supervised learning algorithm widely used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Because of the simple principle of the decision tree algorithm, it is fast to train a decision tree model. Therefore, decision tree model is used not only as a prediction model but also for the early stage coding and testing of this project.

Hyperparameters

max_depth

The maximum depth of the tree. If max_depth is None, then nodes would expand until all leaves are pure or all leaves contain less than min_samples_split samples.

max_features

The number of features to consider when looking for the best split. The value of max_features can be selected from the following:

- An int, then consider max_features features at each split.
- "sqrt", then max_features = sqrt(number of features).
- None, then max_features = number of features.

min_samples_split

The minimum number of samples required to be at a leaf node. A split point at any depth will be considered only if it leaves at least min_samples_leaf training samples in each branch. Turning this hyperparameter may smooth the model, especially for regression.

General Summarize

Grid search CV is done on this model for the 6 different tasks. The result is summarized in the following tables. Where open, close, high, low, adj close and volume are defined as 'absolute features'.

	With absolute features	Without absolute features
Predict 1-day trend	A1	B1
Predict 7-day trend	A2	B2
Predict 30-day trend	A3	B3

Table 4:Table of tasks

Task	A1	A2	A3
Train accuracy	0.748	0.964	0.95
Test Accuracy	0.457	0.66	0.818
AUC	0.407	0.678	0.879
MeanTraining Time	0.06265	0.06568	0.05885
Mean Test Time	0.00076	0.00074	0.00072
Maximum Gini-importance	gold_open 0.053594	sp500_high 0.061133	gold_adj_close 0.124139

Table 5: Performance of Decision Tree Model with Absolute Features

Task	B1	B2	B3
Train accuracy	0.735	0.86	0.968
Test Accuracy	0.49	0.591	0.729
AUC	0.474	0.619	0.716
Mean Training Time	0.04713	0.04887	0.04447
Mean Test Time	0.00074	0.00072	0.0007
Maximum Gini-importance	y5bond_change 0.067923	atr 0.140071	atr 0.118414

Table 6:Performance of Decision Tree Model without Absolute Feature

Detailed Evaluation

A1

```
The best parameters are
{'max_depth': 17, 'max_features': 2, 'min_samples_split': 30}
The Train Accuracy 0.748
AUC ROC : 0.407
The Validation Accuracy 0.550
The Test Accuracy 0.457
[[73 68]
 [66 40]]
              precision    recall  f1-score   support

           0       0.53      0.52      0.52       141
           1       0.37      0.38      0.37       106

    accuracy                           0.46       247
   macro avg       0.45      0.45      0.45       247
weighted avg       0.46      0.46      0.46       247
```

B1

```
The best parameters are
{'max_depth': 17, 'max_features': 2, 'min_samples_split': 40}
The Train Accuracy 0.735
AUC ROC : 0.474
The Validation Accuracy 0.551
The Test Accuracy 0.490
[[73 68]
 [58 48]]
              precision    recall  f1-score   support

           0       0.56      0.52      0.54       141
           1       0.41      0.45      0.43       106

    accuracy                           0.49       247
   macro avg       0.49      0.49      0.48       247
weighted avg       0.50      0.49      0.49       247
```

A2

```
The best parameters are
{'max_depth': 17, 'max_features': None, 'min_samples_split': 10}
The Train Accuracy 0.964
AUC ROC : 0.678
The Validation Accuracy 0.715
The Test Accuracy 0.660
[[96 41]
 [43 67]]
              precision    recall  f1-score   support

           0       0.69      0.70      0.70       137
           1       0.62      0.61      0.61       110

    accuracy                           0.66       247
   macro avg       0.66      0.65      0.66       247
weighted avg       0.66      0.66      0.66       247
```

B2

```
The best parameters are
{'max_depth': 17, 'max_features': None, 'min_samples_split': 40}
The Train Accuracy 0.860
AUC ROC : 0.619
The Validation Accuracy 0.649
The Test Accuracy 0.591
[[89 48]
 [53 57]]
              precision    recall  f1-score   support

           0       0.63      0.65      0.64       137
           1       0.54      0.52      0.53       110

    accuracy                           0.59       247
   macro avg       0.58      0.58      0.58       247
weighted avg       0.59      0.59      0.59       247
```

A3

```
The best parameters are
{'max_depth': 11, 'max_features': None, 'min_samples_split': 20}
The Train Accuracy 0.950
AUC ROC : 0.879
The Validation Accuracy 0.839
The Test Accuracy 0.818
[[148  18]
 [ 27  54]]
              precision    recall  f1-score   support

           0       0.85      0.89      0.87       166
           1       0.75      0.67      0.71        81

    accuracy                           0.82       247
   macro avg       0.80      0.78      0.79       247
weighted avg       0.81      0.82      0.81       247
```

B3

```
The best parameters are
{'max_depth': 19, 'max_features': None, 'min_samples_split': 10}
The Train Accuracy 0.968
AUC ROC : 0.716
The Validation Accuracy 0.729
The Test Accuracy 0.729
[[133  33]
 [ 34  47]]
              precision    recall  f1-score   support

           0       0.80      0.80      0.80       166
           1       0.59      0.58      0.58        81

    accuracy                           0.73       247
   macro avg       0.69      0.69      0.69       247
weighted avg       0.73      0.73      0.73       247
```

Remark

According to the above results, we can find that:

1. The test accuracy and AUC are remarkably higher for long-term prediction.
2. In 1-day trend prediction, AUC < 0.5.
3. There is an obvious improvement in the accuracy of 7-day and 30-day trend prediction when absolute features are used in the model while the accuracy reduction of 1-day trend prediction is relatively small.
4. The training time is very low.

Conclusion

1. The performance of this model is better for the prediction of long-term trend.
2. In this model absolute features contribute to the accuracy of a long-term prediction.

2.3.4 Random Forest Classifier

Instruction

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the accuracy and control over-fitting. Random forest is also able to provide reliable data about the importance of features, which would help with the evaluation of features.

Hyperparameters

n_estimators

The number of trees in the forest.

max_depth

The maximum depth of the tree. If max_depth is None, then nodes would expand until all leaves are pure or all leaves contain less than min_samples_split samples.

max_features

The number of features to consider when looking for the best split. The value of max_features can be selected from the following:

- An int, then consider max_features features at each split.
- “sqrt”, then max_features = sqrt(number of features).
- None, then max_features = number of features.

min_samples_split

The minimum number of samples required to be at a leaf node. A split point at any depth will be considered only if it leaves at least min_samples_leaf training samples

in each branch. Turning this hyperparameter may smooth the model, especially for regression. Unlike decision tree model, min_samples_split is not in the grid search CV and instead fixed at 2.

General Summarize

Grid search CV is done on this model for the 6 different tasks. The result is summarized in the following tables. Where open, close, high, low, adj close and volume are defined as 'absolute features'.

	With absolute features	Without absolute features
Predict 1-day trend	A1	B1
Predict 7-day trend	A2	B2
Predict 30-day trend	A3	B3

Table 7:Table of tasks

Task	A1	A2	A3
Train accuracy	1.000	1.000	1.000
Test Accuracy	0.534	0.789	0.935
AUC	0.518	0.892	0.974
Mean Training Time	4.704044	4.361641	4.216090
Mean Test Time	0.097991	0.086335	0.089626
Maximum Gini-importance	wr_6 0.012832	atr 0.025842	atr 0.034419

Table 8: Performance of the model with Absolute Features

Task	B1	B2	B3
Train accuracy	0.720	1.000	1.000
Test Accuracy	0.575	0.636	0.794
AUC	0.553	0.732	0.851
Mean Training Time	4.165487	4.002384	3.888232
Mean Test Time	0.091939	0.083970	0.086812

Maximum Gini-importance	boll_-1_d 0.032865	atr 0.037563	atr 0.060794

Table 9:Performance of the model without Absolute Features

Detailed Evaluation

A1

```
The best parameters are
{'max_depth': 20, 'max_features': 2, 'n_estimators': 100}
The Train Accuracy 1.000
AUC ROC : 0.518
The Validation Accuracy 0.540
The Test Accuracy 0.534
[[84 57]
 [58 48]]
              precision    recall  f1-score   support

           0       0.59      0.60      0.59       141
           1       0.46      0.45      0.45       106

    accuracy                           0.53       247
   macro avg       0.52      0.52      0.52       247
weighted avg       0.53      0.53      0.53       247
```

B1

```
The best parameters are
{'max_depth': 4, 'max_features': 'sqrt', 'n_estimators': 100}
The Train Accuracy 0.720
AUC ROC : 0.553
The Validation Accuracy 0.546
The Test Accuracy 0.575
[[99 42]
 [63 43]]
              precision    recall  f1-score   support

           0       0.61      0.70      0.65       141
           1       0.51      0.41      0.45       106

    accuracy                           0.57       247
   macro avg       0.56      0.55      0.55       247
weighted avg       0.57      0.57      0.57       247
```

A2

```
The best parameters are
{'max_depth': 20, 'max_features': 'sqrt', 'n_estimators': 1000}
The Train Accuracy 1.000
AUC ROC : 0.892
The Validation Accuracy 0.792
The Test Accuracy 0.789
[[127  10]
 [ 42  68]]
              precision    recall  f1-score   support

           0       0.75      0.93      0.83       137
           1       0.87      0.62      0.72       110

    accuracy                           0.79       247
   macro avg       0.81      0.77      0.78       247
weighted avg       0.81      0.79      0.78       247
```

B2

```
The best parameters are
{'max_depth': 20, 'max_features': 'sqrt', 'n_estimators': 500}
The Train Accuracy 1.000
AUC ROC : 0.732
The Validation Accuracy 0.670
The Test Accuracy 0.636
[[126  11]
 [ 79  31]]
              precision    recall  f1-score   support

           0       0.61      0.92      0.74       137
           1       0.74      0.28      0.41       110

    accuracy                           0.64       247
   macro avg       0.68      0.60      0.57       247
weighted avg       0.67      0.64      0.59       247
```

A3

```
The best parameters are
{'max_depth': 20, 'max_features': 'sqrt', 'n_estimators': 100}
The Train Accuracy 1.000
AUC ROC : 0.974
The Validation Accuracy 0.902
The Test Accuracy 0.935
[[162   4]
 [ 12  69]]
              precision    recall  f1-score   support

           0       0.93      0.98      0.95       166
           1       0.95      0.85      0.90        81

    accuracy                           0.94       247
   macro avg       0.94      0.91      0.92       247
weighted avg       0.94      0.94      0.93       247
```

B3

```
The best parameters are
{'max_depth': 20, 'max_features': 'sqrt', 'n_estimators': 1000}
The Train Accuracy 1.000
AUC ROC : 0.851
The Validation Accuracy 0.745
The Test Accuracy 0.794
[[161   5]
 [ 46  35]]
              precision    recall  f1-score   support

           0       0.78      0.97      0.86       166
           1       0.88      0.43      0.58        81

    accuracy                           0.79       247
   macro avg       0.83      0.70      0.72       247
weighted avg       0.81      0.79      0.77       247
```

Remark

According to the above results, we can find that:

1. The test accuracy and AUC are remarkably higher for long-term prediction.

2. There is an obvious improvement in the accuracy of 7-day and 30-day trend prediction when absolute features are used in the model while the accuracy reduction of 1-day trend prediction is relatively small.
3. The train accuracy of A1, A2, A3, B2 and B3 are 1.0.
4. Average True Range (ATR) gets the highest Gini-importance in A2, A3, B2 and B3.

Conclusion

1. The performance of this model is better for the prediction of a long-term trend.
2. In this model, absolute features contribute to the accuracy of long-term prediction.
3. It is possible that overfitting occurred in the random forest model.
4. Average True Range may have great importance in the prediction of the stock trend.

2.3.5 XGBoost Classifier

Instruction

XGBoost (eXtreme Gradient Boosting) is a supervised learning algorithm based on GBDT (gradient boosting decision tree). This algorithm is dominatingly applied in many recent machine learning projects or competitions. Because of the great potential of XGBoost, it is applied in this project so as to gain high accuracy.

Hyperparameters

max_depth

Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit.

min_child_weight

Minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning.

gamma

Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be.

General Summarize

Grid search CV is done on this model for the 6 different tasks. The result is summarized in the following tables. Where open, close, high, low, adj close and volume are defined as 'absolute features'.

	With absolute features	Without absolute features
Predict 1-day trend	A1	B1
Predict 7-day trend	A2	B2
Predict 30-day trend	A3	B3

Table 10:Table of tasks

Task	A1	A2	A3
Train accuracy	0.733	1.000	1.000
Test Accuracy	0.551	0.765	0.927
AUC	0.562	0.849	0.964
Mean Training Time	23.831478	20.873437	18.967230
Mean Test Time	0.005965	0.005388	0.004888
Maximum Gini-importance	y5bond_high_low_diff 0.052265	atr 0.046401	atr 0.057297

Table 11: Performance of the model with Absolute Features

Task	B1	B2	B3
Train accuracy	1.000	1.000	1.000
Test Accuracy	0.551	0.664	0.781
AUC	0.544	0.706	0.868
Mean Training Time	15.991266	15.514667	4.462023
Mean Test Time	0.005566	0.005433	0.005341
Maximum Gini-importance	sp500_volume_delta 0.028931	atr 0.067559	atr 0.071785

Table 12:Performance of the model without Absolute Features

Detailed Evaluation

A1

```
The best parameters are
{'gamma': 0.8, 'max_depth': 2, 'min_child_weight': 7, 'n_estimators': 100}
The Train Accuracy 0.733
AUC ROC : 0.562
The Validation Accuracy 0.555
The Test Accuracy 0.551
[[81 60]
 [51 55]]
              precision    recall  f1-score   support

           0       0.61      0.57      0.59       141
           1       0.48      0.52      0.50       106

    accuracy                           0.55       247
   macro avg       0.55      0.55      0.55       247
weighted avg       0.56      0.55      0.55       247
```

B1

```
The best parameters are
{'gamma': 0.8, 'max_depth': 20, 'min_child_weight': 7, 'n_estimators': 100}
The Train Accuracy 1.000
AUC ROC : 0.544
The Validation Accuracy 0.529
The Test Accuracy 0.551
[[81 60]
 [51 55]]
              precision    recall  f1-score   support

           0       0.61      0.57      0.59       141
           1       0.48      0.52      0.50       106

    accuracy                           0.55       247
   macro avg       0.55      0.55      0.55       247
weighted avg       0.56      0.55      0.55       247
```

A2

```
The best parameters are
{'gamma': 0.8, 'max_depth': 20, 'min_child_weight': 3, 'n_estimators': 100}
The Train Accuracy 1.000
AUC ROC : 0.849
The Validation Accuracy 0.799
The Test Accuracy 0.765
[[122  15]
 [ 43  67]]
              precision    recall  f1-score   support

           0       0.74      0.89      0.81       137
           1       0.82      0.61      0.70       110

    accuracy                           0.77       247
   macro avg       0.78      0.75      0.75       247
weighted avg       0.77      0.77      0.76       247
```

B2

```
The best parameters are
{'gamma': 0.8, 'max_depth': 10, 'min_child_weight': 5, 'n_estimators': 100}
The Train Accuracy 1.000
AUC ROC : 0.706
The Validation Accuracy 0.722
The Test Accuracy 0.664
[[113  24]
 [ 59  51]]
              precision    recall  f1-score   support

           0       0.66      0.82      0.73       137
           1       0.68      0.46      0.55       110

    accuracy                           0.66       247
   macro avg       0.67      0.64      0.64       247
weighted avg       0.67      0.66      0.65       247
```

A3

```
The best parameters are
{'gamma': 0.8, 'max_depth': 20, 'min_child_weight': 3, 'n_estimators': 100}
The Train Accuracy 1.000
AUC ROC : 0.964
The Validation Accuracy 0.913
The Test Accuracy 0.927
[[160   6]
 [ 12  69]]
              precision    recall  f1-score   support

           0       0.93      0.96      0.95       166
           1       0.92      0.85      0.88        81

    accuracy                           0.93       247
   macro avg       0.93      0.91      0.92       247
weighted avg       0.93      0.93      0.93       247
```

B3

```
The best parameters are
{'gamma': 0.8, 'max_depth': 10, 'min_child_weight': 3, 'n_estimators': 100}
The Train Accuracy 1.000
AUC ROC : 0.868
The Validation Accuracy 0.810
The Test Accuracy 0.781
[[155  11]
 [ 43  38]]
              precision    recall  f1-score   support

           0       0.78      0.93      0.85       166
           1       0.78      0.47      0.58        81

    accuracy                           0.78       247
   macro avg       0.78      0.70      0.72       247
weighted avg       0.78      0.78      0.76       247
```

Remark

According to the above results, we can find that:

1. The test accuracy and AUC are remarkably higher for long-term prediction.
2. There is an obvious improvement in the accuracy of 7-day and 30-day trend prediction when absolute features are used in the model while the accuracy reduction of 1-day trend prediction is relatively small.
3. The train accuracy of A1, A2, A3, B2 and B3 are 1.0.
4. Average True Range (ATR) gets the highest Gini-importance in A2, A3, B2 and B3.
5. The training time is long.

Conclusion

1. The performance of this model is better for the prediction of a long-term trend.
2. In this model, absolute features contribute to the accuracy of long-term prediction.
3. It is possible that there is overfitting in the XGBoost model.
4. Average True Range may have great importance in the prediction of stock trend.
5. Although XGBoost provides high accuracy, the training time of a XGBoost model is long. A trade between accuracy and model efficiency is required.

2.3.6 Long Short Term Memory

Instruction

LSTM (Long short-term memory) is a machine learning algorithm that uses memory cells instead of traditional neurons in the hidden layer. LSTM is designed specially for tasks in long time series, so it has a great potential in stock prediction. However, LSTM requires large computing power. So, it is also worthy to test the performance of LSTM with low computing power.

Hyperparameters

hidden_layer_size: Size of hidden layers

stacks: How many layers of LSTM are staked in the model

sequence_day: How many data in batch (ex. 3 day sequence → 3 days data in a batch)

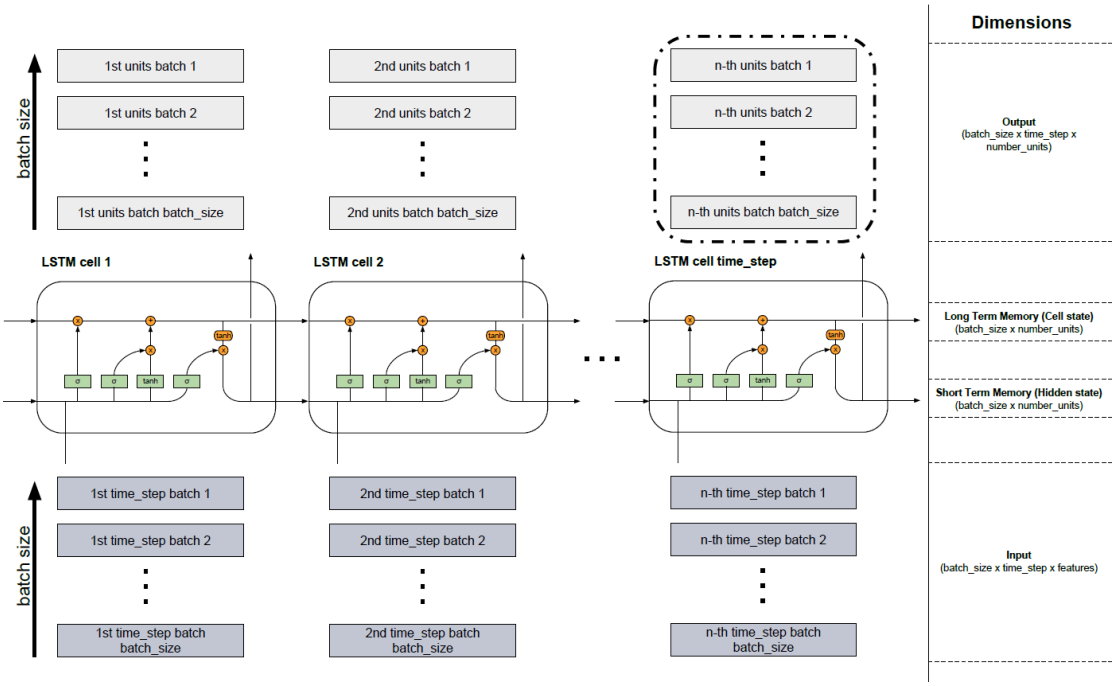


Figure 10. LSTM Model Diagram and Batch Illustration

General Summarize

Since there is no Grid Search CV for the LSTM model, the only thing we can do is to manually conduct the search.

1. Experiments on Stacks (How many LSTM layers) and Early Stop Monitor

```
def buildTrendModel_1stacks(shape, hidden_layer_size, batch_size):
    print(shape)
    model = Sequential()
    model.add(LSTM(hidden_layer_size,  batch_input_shape=(batch_size, shape[1], shape[2]),
                   stateful=True, init='glorot_uniform'))
    model.add(Dense(20, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss="binary_crossentropy", optimizer='adam', metrics=['accuracy'])
#     model.summary()
    return model
```

1 Stack LSTM Design

```
def buildTrendModel_2stacks(shape, hidden_layer_size, batch_size):
    model = Sequential()
    model.add(LSTM(hidden_layer_size, return_sequences=True,
    batch_input_shape=(batch_size, shape[1], shape[2]), stateful=True, init='glorot_uniform'))
    model.add(LSTM(hidden_layer_size, stateful=True, init='glorot_uniform'))
    model.add(Dropout(0.5))
    model.add(Dense(20, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss="binary_crossentropy", optimizer='adam', metrics=['accuracy'])
#     model.summary()
    return model
```

2 Stack LSTM Design

```
def buildTrendModel_3stacks(shape, hidden_layer_size, batch_size):
    model = Sequential()
    model.add(LSTM(hidden_layer_size, return_sequences=True,
    batch_input_shape=(batch_size, shape[1], shape[2]), stateful=True, init='glorot_uniform'))
    model.add(LSTM(hidden_layer_size, return_sequences=True, stateful=True, init='glorot_uniform'))
    model.add(Dropout(0.5))
    model.add(LSTM(hidden_layer_size, stateful=True, init='glorot_uniform'))
    model.add(Dropout(0.2))
    model.add(Dense(20, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss="binary_crossentropy", optimizer='adam', metrics=['accuracy']) # used for Binary classify
#     model.summary()
    return model
```

3 Stack LSTM Design

```
def buildTrendModel_4stacks(shape, hidden_layer_size, batch_size):
    model = Sequential()
    model.add(LSTM(hidden_layer_size, return_sequences=True,
    batch_input_shape=(batch_size, shape[1], shape[2]), stateful=True, init='glorot_uniform'))
    model.add(LSTM(hidden_layer_size, return_sequences=True, stateful=True, init='glorot_uniform'))
    model.add(Dropout(0.5))
    model.add(LSTM(hidden_layer_size, return_sequences=True, stateful=True, init='glorot_uniform'))
    model.add(LSTM(hidden_layer_size, stateful=True, init='glorot_uniform'))
    model.add(Dropout(0.2))
    model.add(Dense(20, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss="binary_crossentropy", optimizer='adam', metrics=['accuracy']) # used for Binary classify
#     model.summary()
    return model
```

4 Stack LSTM Design

Generally, more stacks would have better performance compare to single stacks. However, we still need to test it in order to get the optimal model. We've test all the model twice with monitoring val_accuracy and val_loss accordingly. Here is the result

(sequence 7; hidden layer: 64; batch size: 7; predict one day trend)

	1-Stack	2-Stack	3-Stack	4-Stack
val_accuracy	0.539	0.555	0.551	0.551
val_loss	0.551	0.551	0.551	0.551

Test Accuracy of Different Stacks o Model Under Different Monitoring Target

2-Stack is better. Noticing the results with accuracy 0.551. They actually predicts all value to 0, which we believe should be failed cases.

2. Different Hidden Layer Size and Sequence day (Batch Size).

We've also done experiments on different Layer Size and Sequence day. The result is summarized in the following tables. Where open, close, high, low, adj close, and volume are defined as 'absolute features'.

	With absolute features	Without absolute features
Predict 1-day trend	A1	B1
Predict 7-day trend	A2	B2
Predict 30-day trend	A3	B3

Table 13:Table of tasks

Task	A1	A2	A3
Train accuracy	0.526	0.579	0.630
Test Accuracy	0.551	0.694	0.792
AUC	0.500	0.494	0.500

Table 14: Performance of the model with Absolute Features

Task	B1	B2	B3
Train accuracy	0.964	0.904	0.630
Test Accuracy	0.559	0.637	0.792
AUC	0.543	0.540	0.500

Table 15:Performance of the model without Absolute Features

For training efficiency, all of the model training time are longer than XGBoost classifier. Moreover, the larger the layer size and the sequence day, the longer the training time. It took 2 complete days to run through all the search. With the longest training time around 2 hours, and shortest training time 191.84 s.

Detailed Evaluation

A1

```
Best Parameter: {hidden_layer: 64, sequence_day: 3}
The Train Accuracy 0.526
The Validation Accuracy 0.535
The Test Accuracy 0.551
AUC ROC : 0.500
[[135   0]
 [110   0]]
              precision    recall  f1-score   support

           0       0.55      1.00      0.71       135
           1       0.00      0.00      0.00       110

    accuracy                           0.55       245
   macro avg       0.28      0.50      0.36       245
weighted avg       0.30      0.55      0.39       245
```

B1

```
Best Parameter: {hidden_layer: 32, sequence_day: 15}
The Train Accuracy 0.964
The Validation Accuracy 0.576
The Test Accuracy 0.559
AUC ROC : 0.543
[[95 40]
 [68 42]]
              precision    recall  f1-score   support

           0       0.58      0.70      0.64       135
           1       0.51      0.38      0.44       110

    accuracy                           0.56       245
   macro avg       0.55      0.54      0.54       245
weighted avg       0.55      0.56      0.55       245
```

A2

```
Best Parameter: {hidden_layer: 128, sequence_day: 1}
The Train Accuracy 0.579
The Validation Accuracy 0.567
The Test Accuracy 0.694
AUC ROC : 0.494
[[170   2]
 [ 73   0]]
              precision    recall  f1-score   support

           0       0.70      0.99      0.82       172
           1       0.00      0.00      0.00        73

    accuracy                           0.69       245
   macro avg       0.35      0.49      0.41       245
weighted avg       0.49      0.69      0.58       245
```

B2

```
Best Parameter: {hidden_layer: 128, sequence_day: 1}
The Train Accuracy 0.904
The Validation Accuracy 0.539
The Test Accuracy 0.637
AUC ROC : 0.540
[[134  38]
 [ 51  22]]
              precision    recall  f1-score   support

           0       0.72      0.78      0.75       172
           1       0.37      0.30      0.33        73

    accuracy                           0.64       245
   macro avg       0.55      0.54      0.54       245
weighted avg       0.62      0.64      0.63       245
```

A3

```
Best Parameter: {hidden_layer: 128, sequence_day: 15}
The Train Accuracy 0.630
The Validation Accuracy 0.562
The Test Accuracy 0.792
AUC ROC : 0.500
[[194   0]
 [ 51   0]]
              precision    recall  f1-score   support

           0       0.79      1.00      0.88       194
           1       0.00      0.00      0.00        51

    accuracy                           0.79       245
   macro avg       0.40      0.50      0.44       245
weighted avg       0.63      0.79      0.70       245
```

B3

```
Best Parameter: {hidden_layer: 128, sequence_day: 15}
The Train Accuracy 0.630
The Validation Accuracy 0.562
The Test Accuracy 0.792
AUC ROC : 0.500
[[194   0]
 [ 51   0]]
              precision    recall  f1-score   support

           0       0.79      1.00      0.88       194
           1       0.00      0.00      0.00        51

    accuracy                           0.79       245
   macro avg       0.40      0.50      0.44       245
weighted avg       0.63      0.79      0.70       245
```

Remark

According to the above results, we can find that:

1. The accuracy of LSTM on task B3 is the second among all models (0.792, while the random forest is 0.794).
2. The accuracy of LSTM on task B3 and A3 are the same.
3. Train accuracy on task B1 is high.
4. Train accuracy on task B's is obviously higher than task A's while test accuracy is similar.

Conclusion

1. Absolute features do not contribute to the performance of LSTM.
2. But absolute features may keep the model from overfitting.
3. Overfitting is highly possible on task B1.

2.3.7 Reinforcement Learning

Instruction

Reinforcement learning is another area of machine learning. In a reinforcement model, an agent interacts with its environment by receiving observations which typically includes the reward and taking actions to the environment. In this project, the environment is established by the chronological historical data of stock price. The agent is a virtual stock trader that interacts with the environment by selling, buying or holding stocks. The reword is the amount of money the agent earns or losses.

Hyperparameters

total_timesteps

The number of steps (i.e. trading days) used in the training. As the amount of stock data is limited, data may be reused if total_timesteps is too high and therefore cause overfitting.

test_timesteps

The number of steps used in the testing. To get a fair test result, test_timesteps should be equivalent when two models are compared.

General Summarize

As the agent interacts with the environment dynamically, the performance of this model is evaluated by the amount of money the agent earns in 3000 trading days with a beginning balance of 10,000 dollars.

Hyperparameters	total_timesteps=20,000 test_timesteps=3000	total_timesteps=20,000 test_timesteps=20000
Profit	13649.609295255003	-7577.014401069046
Max net worth	31312.777715979355	38994.4381245817

Detailed Evaluation

total_timesteps=20,000, test_timesteps=3000:

```
Balance: 23649.609295255003
Shares held: 0 (Total sold: 12559)
Avg cost for held shares: 0 (Total sales value: 455195.94704648684)
Net worth: 23649.609295255003 (Max net worth: 31312.777715979355)
Profit: 13649.609295255003
```

total_timesteps=20,000, test_timesteps=20000:

```
Balance: 2422.9855989309544
Shares held: 0 (Total sold: 99865)
Avg cost for held shares: 0 (Total sales value: 4179274.3536050334)
Net worth: 2422.9855989309544 (Max net worth: 38994.4381245817)
Profit: -7577.014401069046
```

Remark

According to the above results, we can find that:

1. When test_timesteps = 20,000, the agent gets an abnormal loss while the max net worth is similar to test_timesteps = 3,000.
2. The number of rows in test data is between 3000 and 20,000.

Conclusion

1. The Reinforcement Model is able to generate an agent that earns money in the stock market.
2. Number of time steps in test should not exceed the number of rows in test data as the amount of stock data is limited, data may be reused if total_timesteps is too high and therefore cause an unreliable test result.

2.3.8 Attempt for Model Explanation

In order to determine which features result in better accuaracy, we constructed the following experiment. Since XGBoost Classifier perform relatively well compare to other models, we chose XGBoost Classifier to train different combination of categories and see what would be the influence to the accuracy

```
finacial_list = ['change', 'open_delta', 'close_delta', 'volume_delta',
'high_low_diff_ratio', 'open_close_diff_ratio', 'close_-2_r',
'close_-6_r', 'kdjk', 'kdjd', 'kdjj', 'macd', 'macds', 'macdh', 'rsi_6',
'rsi_12', 'wr_6', 'wr_12', 'cci', 'atr', 'dma', 'vr', 'boll_-1_d',
'boll_ub_-1_d', 'boll_lb_-1_d', 'boll_k_diff', 'high_low_diff',
'open_close_diff', 'open', 'high', 'low', 'close', 'adj close', 'volume']

news_list = ['news_des_score', 'sp500_open']

external_list = ['sp500_open',
       'sp500_high', 'sp500_low', 'sp500_close', 'sp500_adj close',
       'sp500_volume', 'sp500_high_low_diff', 'sp500_open_close_diff',
       'sp500_high_low_diff_ratio', 'sp500_open_close_diff_ratio',
       'sp500_change', 'sp500_close_delta', 'sp500_volume_delta',
       'sp500_close_-2_r', 'sp500_close_-6_r', 'gold_open', 'gold_high',
       'gold_low', 'gold_close', 'gold_adj close', 'gold_volume',
       'gold_high_low_diff', 'gold_open_close_diff',
       'gold_high_low_diff_ratio', 'gold_open_close_diff_ratio', 'gold_change',
       'gold_close_delta', 'gold_volume_delta', 'gold_close_-2_r',
       'gold_close_-6_r', 'y10bond_open', 'y10bond_high', 'y10bond_low',
       'y10bond_close', 'y10bond_adj close', 'y10bond_high_low_diff',
       'y10bond_open_close_diff', 'y10bond_high_low_diff_ratio',
       'y10bond_open_close_diff_ratio', 'y10bond_change',
       'y10bond_close_delta', 'y10bond_close_-2_r', 'y10bond_close_-6_r',
       'y5bond_open', 'y5bond_high', 'y5bond_low', 'y5bond_close',
       'y5bond_adj close', 'y5bond_high_low_diff', 'y5bond_open_close_diff',
       'y5bond_high_low_diff_ratio', 'y5bond_open_close_diff_ratio',
       'y5bond_change', 'y5bond_close_delta', 'y5bond_close_-2_r','y5bond_close_-6_r']
```

Figure 11. Extracting Feature Name for Different Categories

We've test and train the model with the best parameters found previously, and here is the result.

Features	*1 Day*	*7 Day*	*30 Day*
Only Financial	0.567	0.721	0.891
Only Sentiment	0.571	0.607	0.725
Only External	0.514	0.652	0.883
Without Financial	0.522	0.664	0.891
Without Sentiment	0.559	0.761	0.923
Without External	0.543	0.749	0.911
All	0.551	0.749	0.927

Table 1. Test Accuracy of Different Combination of Categories

Remark

For predicting 1 day trend, sentiment features have the best performance, following by financial indicators and external features.

However, for predicting 7 day and 30 day trend, financial indicators dominate the performance, following by external features, and sentiment features results in the lowest accuracy. We believe this demonstrates the market efficiency, saying that news will only have large effect to latest stock price, and its influence to stock price would decrease as the time goes by.

2.3.9 LSTM Real Stock Price Prediction

Followeing the best hyper parameters found previously, we've constructed our LSTM model to predict real stock price.

```
def buildTrendModel_2stacks_true_value(shape, hidden_layer_size, batch_size):
    model = Sequential()
    model.add(LSTM(hidden_layer_size, return_sequences=True, batch_input_shape=(batch_size, shape[1], shape[2]),
              stateful=True))
    model.add(LSTM(hidden_layer_size, stateful=True))
    model.add(Dropout(0.5))
    model.add(Dense(20))
    model.add(Dense(1))
    model.compile(loss="mean_squared_error", optimizer='adam', metrics=['mae']) # or sgd
    return model
```

```
batch_size = 7
model, X_train, y_train, X_valid, y_valid, X_test, y_test  = model_train(buildTrendModel_2stacks_true_value, 3,
                                                    stock_with_abs_norm, label_value_1d, 64, batch_size, "loss")

predicted_test = np.array(model.predict(X_test, batch_size=batch_size))
predicted_train = np.array(model.predict(X_train, batch_size=batch_size))
predicted_valid = np.array(model.predict(X_valid, batch_size=batch_size))
print()
print('\n# Evaluate on test data')
results = model.evaluate(X_test, y_test, batch_size=7)
print('test loss, test acc:', results)

# Generate predictions (probabilities -- the output of the last layer)
# on new data using `predict`
print('\n# Generate predictions for 3 samples')
predictions = model.predict(X_test, batch_size=7)
print('predictions shape:', predictions.shape)

rmse=np.sqrt(np.mean(((predictions- y_test)**2)))
print('rmse:', rmse)
```

Figure 12. 2 Stack LSTM to Predict Real Stock Price

Figure 12. shows the model implementation for real stock price prediction. Noticing that we removed activation function away, and also there would be no initial states for the model. The model will train to acheive the lowest loss, which is the mean squared error. We also print out the mean average error as our reference.

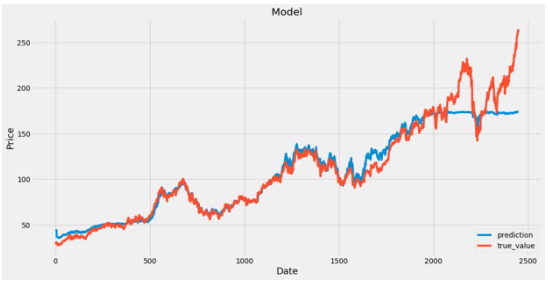
Theses are the basic hyper-parameters:

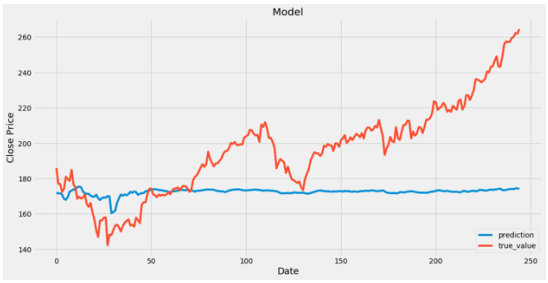
{2 stack, stateful, hidden layer 64, batch size 7},

There are 1981 train data, 217 validation data and 245 test data in total. Since the average trading days is around 253 days an year, and since our data are in between 2009 – 2019, we can say that we use the data before 2018 for training, and we validate the model using data in 2018, then we predict the price in 2019. The prediction results are shown below follow by its Root Mean Squared Error Score(RMSE).

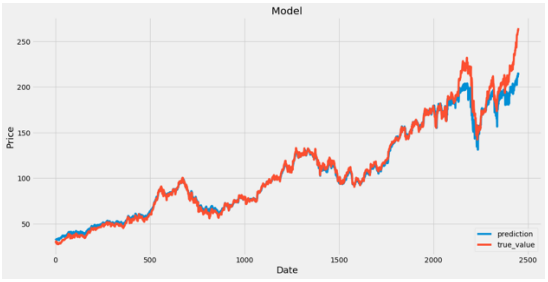
(Left: contain training /validation data prediction; Right: Only test data prediction)

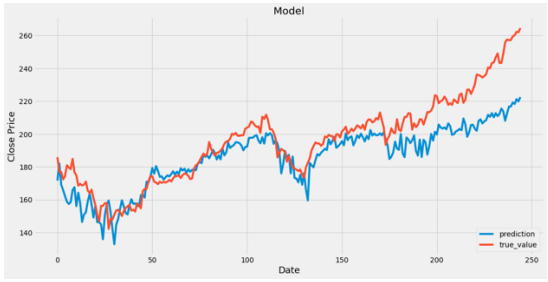
1 Day Sequence (#Batch=1) RMSE: 34.495



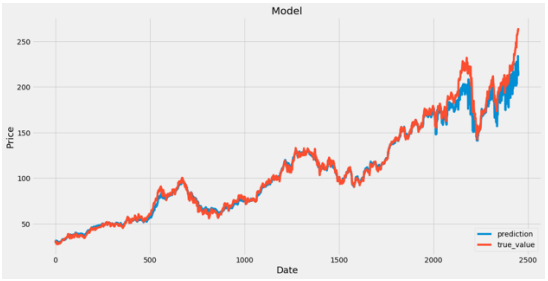


3 Day Sequence (#Batch=3) RMSE: 15.257

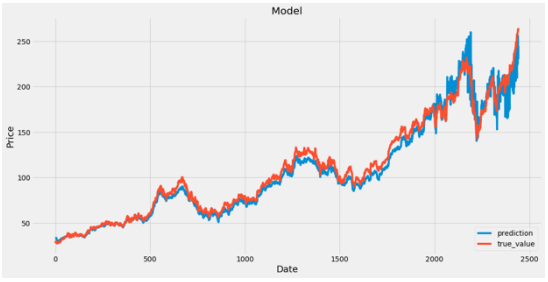


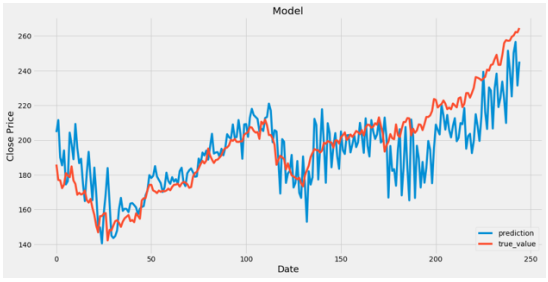


5 Day Sequence (#Batch=5) RMSE: 16.221



7 Day Sequence (#Batch=7) RMSE: 15.726





Observation

1. All models well fit the training data set. Only 1-day sequence model failed to fit the validation dataset.

2. In general, 1-day sequence of data is not sufficient for the LSTM model to make predictions. Starting from using 3-day sequence of data, the predictions can show the trend and are closer to the real stock price. This reveals that LSTM could observe the pattern for multiple-day sequence and apply the pattern to its prediction
3. 3-Day sequence model has lower root mean square error, and it did not oscillate severely, so it has the best performance compared to all other models.
4. Starting from using the 5-day sequence data, the predictions oscillate severely. We do not know the reason, but we believed if this issue is solved, it is obvious that both 5-day/ 7-day sequence model could perform better than the 3-day sequence model since they are closer to the real price.

2.4 Predicion System Implementation

After retrieving prediction results from the model evaluated earlier, we then focus on constructing a website to demonstrate and visualize the prediction. The main goal is to build a useful website that can help the user to easily understand the stock prediction.

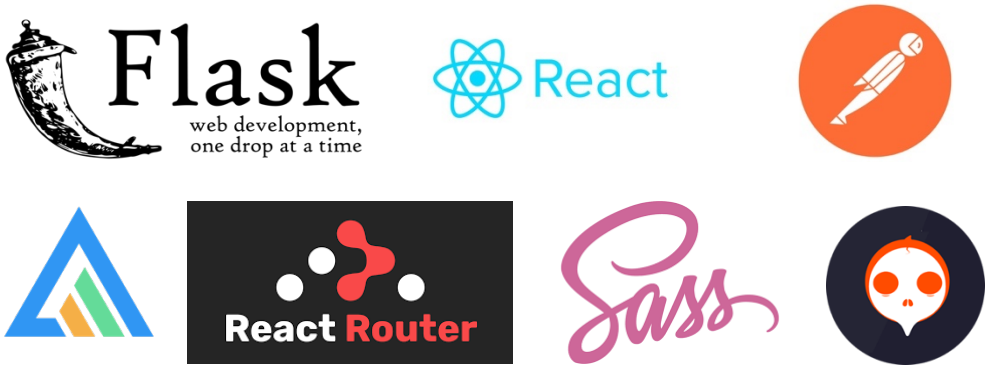


Figure 13. Tools used for Building our System

2.4.1 Back-end Implementation

Originally, we planned to use ExpressJS as our backend server. However, after training the machine learning models, we switch to use Flask for our backend server in order to have better integration with the models. The main function of the backend server is to transmit data to the clients. Therefore, in order to immediately generate prediction results after receiving POST request, the first thing we need to do is to store all the pre-trained models according to their best parameters (Figure 10.)

```
# stock_with_absolute, predict 1 day trend
best_parameters = {'max_depth': 4, 'max_features': 'sqrt', 'n_estimators': 100}
rfc = RandomForestClassifier(**best_parameters)

# train data and label
train_data = stock_with_absolute
label = label_abs_1d

X_train, X_test, y_train, y_test = train_test_split(train_data, label, test_size=0.1, random_state=42)
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

rfc.fit(X_train, y_train)

predicted_test = rfc.predict(X_test)
predicted_train = rfc.predict(X_train)
decision_function = rfc.predict_proba(X_test)

acc_train = accuracy_score(y_train, predicted_train)
acc_test = accuracy_score(y_test, predicted_test)
print("The Train Accuracy  %0.3f" % (acc_train))
print("The Test Accuracy   %0.3f" % (acc_test ))
pickle.dump(rfc, open('./backend/AAPL/RandomForest/RFC_1d.pkl','wb')) # store model to .pkl
```

Figure 14. Pre-train Model and Store it (i.e. RandomForest Model for this Figure)

After storing the model, we can then focus on API design. There are two API in total. The first API is to retrieve the stock price data (Figure 12.) It will simply return stock data in JSON format.

```
@app.route('/api/v0.1/get_stock_price',methods=['POST'])
def get_stock_price():
    data = request.get_json(force=True)
    symbol = data["symbol"]
    if(symbol in avail_symbols):
        stock_raw = yf.download(symbol, start=stock_date_start)
        return Response(stock_raw.transpose().to_json(), mimetype='application/json')
    else:
        error = {"warn": "Symbol not preprocessed"}
        return jsonify(error)
```

Figure 12. API which Return Stock Data According to Symbol Given

The second API is to preprocess all the data, load the machine learning models stored previously , and return the predictions accordingly (Figure 13.)

```
@app.route('/api/v0.1/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    symbol = data["symbol"]
    stock_date_start = "2020-03-01"
    stock_with_absolute = pd.read_pickle('./AAPL/data/stock_with_absolute.pkl')
    label_abs_1d = pd.read_pickle('./AAPL/data/label_abs_1d.pkl')


...

return_dict["LGR"] = [int(pickle.load(open('./AAPL/LogisticRegression/LR_1d.pkl','rb')).predict(target)[0]),
                      int(pickle.load(open('./AAPL/LogisticRegression/LR_7d.pkl','rb')).predict(target)[0]),
                      int(pickle.load(open('./AAPL/LogisticRegression/LR_30d.pkl','rb')).predict(target)[0])]
return_dict["DT"] = [int(pickle.load(open('./AAPL/DecisionTree/DT_1d.pkl','rb')).predict(target)[0]),
                      int(pickle.load(open('./AAPL/DecisionTree/DT_7d.pkl','rb')).predict(target)[0]),
                      int(pickle.load(open('./AAPL/DecisionTree/DT_30d.pkl','rb')).predict(target)[0])]
return_dict["RF"] = [int(pickle.load(open('./AAPL/RandomForest/RFC_1d.pkl','rb')).predict(target)[0]),
                      int(pickle.load(open('./AAPL/RandomForest/RFC_7d.pkl','rb')).predict(target)[0]),
                      int(pickle.load(open('./AAPL/RandomForest/RFC_30d.pkl','rb')).predict(target)[0])]
return_dict["XGB"] = [int(pickle.load(open('./AAPL/XGBoost/XGB_1d.pkl','rb')).predict(target)[0]),
                      int(pickle.load(open('./AAPL/XGBoost/XGB_7d.pkl','rb')).predict(target)[0]),
                      int(pickle.load(open('./AAPL/XGBoost/XGB_30d.pkl','rb')).predict(target)[0])]
return jsonify(return_dict)
```

Figure 15. Load data, Preprocess it, Predict it, Send it

The function would also return today's top 5 news and its sentiment score respectively. We've generally followed the RESTful API design, which we versioned our API, and we also used verb as our API name.

2.4.2 Front-end Implementaion

For the front end website, we created our website based on React. Starting from create-react-app, we integrated SCSS, CSS-module, react-router, IconFont, and some minor libraries like Lodash. Besides, we decided to use ApexChart as our interactive charting and visualization library. Following we will explain our implementation.

1. Structure

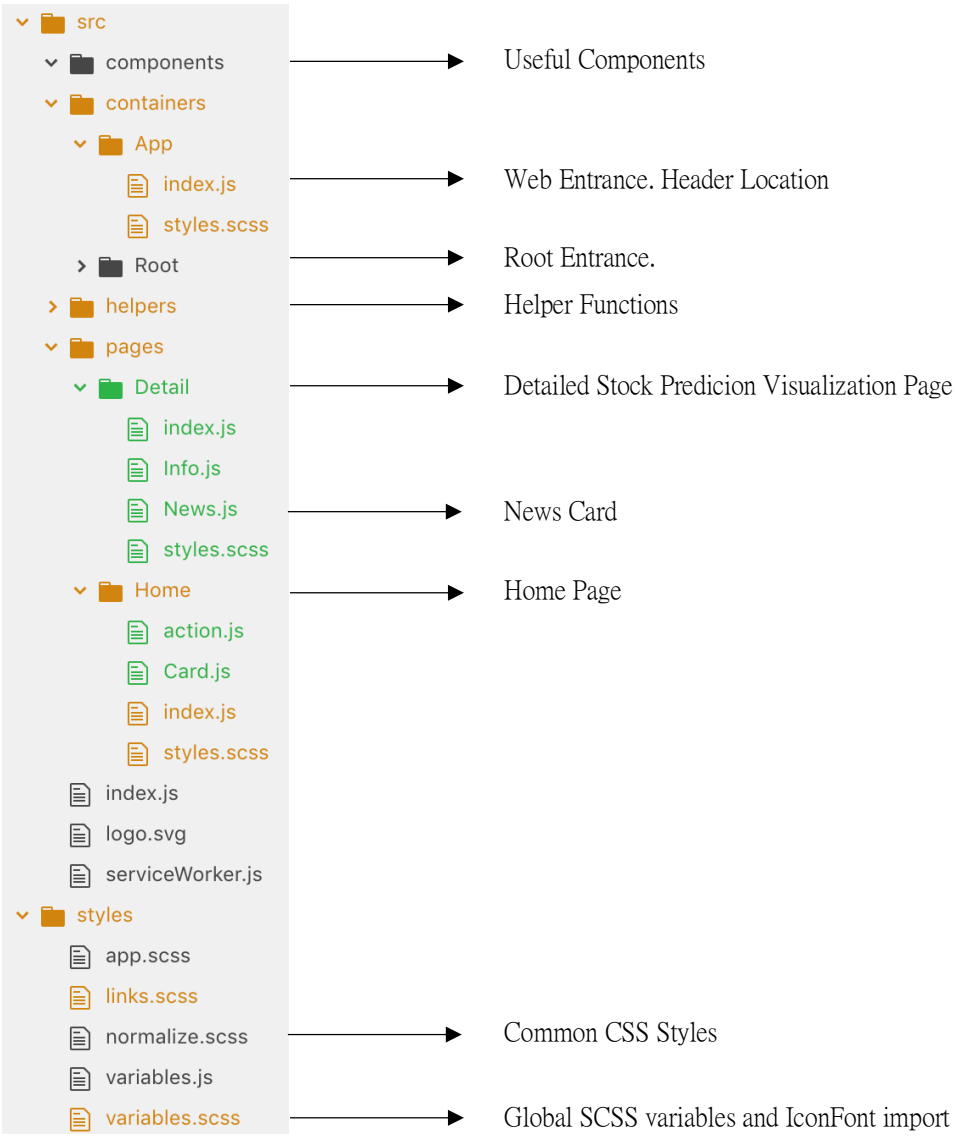


Figure 14. File Structure

2. Helper

Helper is the place for helper functions. In this project, the most important helper function is run, which is used for fetching data from the backend server.

```
import _ from 'lodash'

async function run(func, params) {

  let res = await fetch(`http://localhost:5001/api/v0.1/${func}`, {
    method: 'POST',
    headers: {
      Accept: 'application/json, text/plain, */*',
      'Content-Type': 'application/json',
      'Cache-Control': 'no-cache',
    },
    body: JSON.stringify(
      params,
    ),
  });
  res = await res.json();
  if (typeof res === 'string' && (res.startsWith('{') || res.startsWith('['))) res = JSON.parse(res);
  if (res && res.error) {
    console.error(res.error);
    throw res.error;
  }

  return res;
}

export default run;
```

Figure 16. Run Function to Fetch Data from Server

It is an asynchronous function, which other processes will wait until the data is fetched. It will fetch the "func" API by posting to the specific url with parameters wrap in JSON. After receiving the data, it will return the response if no error occurs.

3. Router

The Root of our web is wrapped by BrowserRouter. It is a router that uses the HTML5 history API (pushState, replaceState and the popstate event) to keep the UI in sync with the URL. The specific routes are specified in APP, so that the web will render different page according to the browser url. Notice that the id is the detailed symbol that user wish to predict.

```
render() {
  return (
    <div className={styles.home}>
      <div className={styles.header}>
        Smart Investment
        <div
          className={styles.returnIcon}
          onClick={()=>{
            this.props.history.push('/');
          }}
        >
          &#xe624;
        </div>
      </div>
      <Route exact path="/" component={Home} />
      <Route path="/detail/:id" component={Stock} />
    </div>
  );
}
```

Figure 17. Routes in APP

4. Home Page

```
render() {
  return (
    <div className={styles.body}>
      <div className={styles.title}>
        Please Input the Symbol(Ticker)
      </div>
      {/* {_.map(plugs, (item, index) => {
      //   return(<Card key={index} plugID={item} />);
      // })} */}
      <div className={styles.addPlug}>
        <input
          placeholder="Symbol (ex: AAPL)"
          onChange={(event)=>{
            this.setState({
              symbol: event.target.value,
            });
          }}
        />
        <div
          className={styles.addButton}
          onClick={() => {
              this.props.history.push("/detail/"+this.state.symbol);
          }}
        >Search</div>
      </div>
      <div style={{color: 'red'}}>{this.state.error}</div>
    </div>
  );
}
```

The home page is designed for users to input the symbol of the stock they wish to predict. The stock symbol is maintained in state, and whenever the input field is changed, it will update correspondingly. After Pressing the search button, the browser will then push users to the destined pages.

Figure 19. Home Page render function

5. Detail Page

The detail page is for detailed stock visualization. There will be a candle chart displaying stock price followed by the prediction result and the news. In componentDidMount, the webpage will fetch the stock price and the prediction results. If the stock is not trained yet, it will display a webpage to ask the auther to train for the particular symbol.

Remark: Each webpage is designed responsively so that mobile users could also have a user friendly experience (accomplish by using grid flex design). Iconfont is used for those small Icons.

```
async componentDidMount(){

  const param = {
    "symbol": this.props.match.params.id,
  };

  const result = await run('get_stock_price', param);
  if (result.warn) {
    this.setState({
      error: "Not Supported Yet",
    });
  }
  else {
    let stock_series = []
    _.each(result,(item, index)=>{
      stock_series.push({
        x: new Date(parseInt(index)),
        y: [item["Open"].toFixed(2), item["High"].toFixed(2), item["Low"].toFixed(2), item["Close"].toFixed(2)]
      })
    });
    var stock_data = [{data:stock_series}];
    this.setState({
      series: stock_data,
    });
    const predict = await run('predict', param);
    console.log(predict);
    this.setState({
      predict: predict,
    });
  }
}
```

Figure 20. Detail Page componentDidMount which process the data to state

2.4.3 Results Demonstration

After the user enters the webpage, it will first route to the Home page.

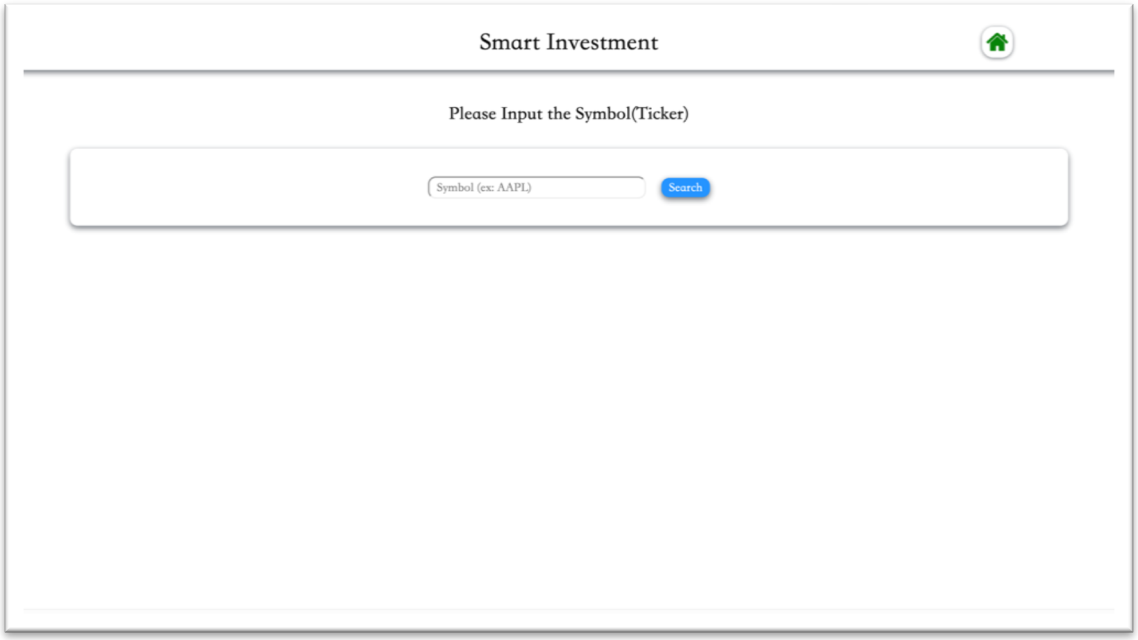


Figure 21. Home Page

After entering the target stock symbol, if the stock symbol is not trained yet, it will display an error message: (ex: Enter Trash)

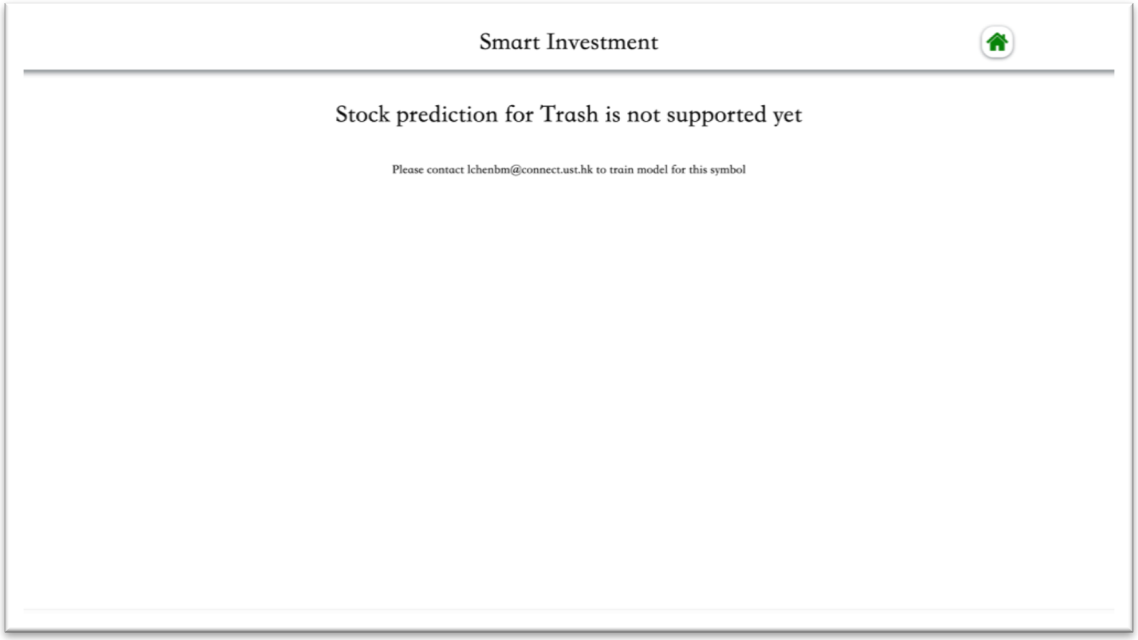


Figure 22. Unavailable Page

If available input, the website will render the detail stock page. (ex: Enter AAPL)

Figure 23. Detail Page with Stock Price in Candle Chart

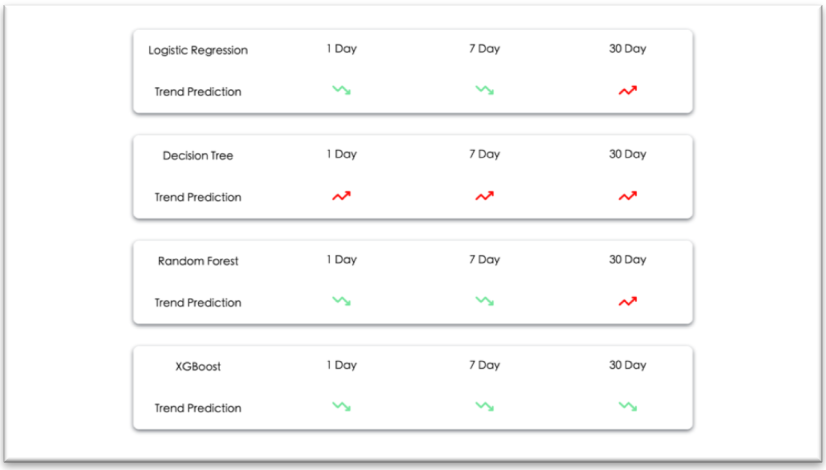


Figure 24. Detail Page with Stock Price Prediction Corresponing to Different Model

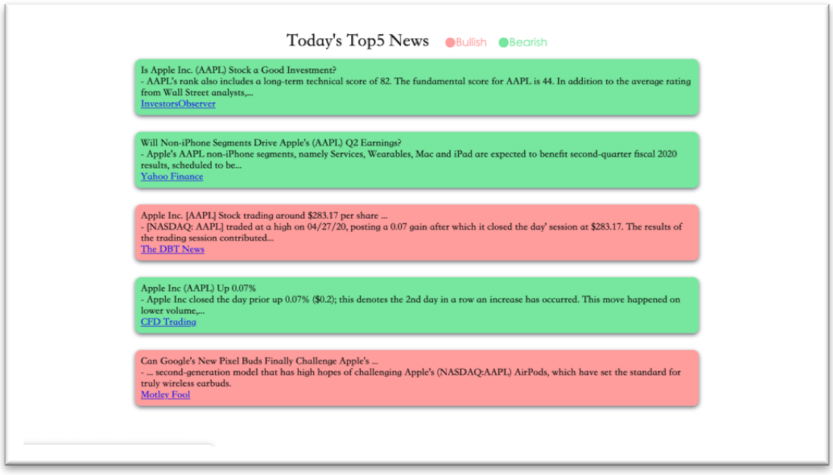


Figure 25. Crawled News With its Description Sentiment Score (Red > 0; Green ≤ 0)

In addition, the website pages are responsive.

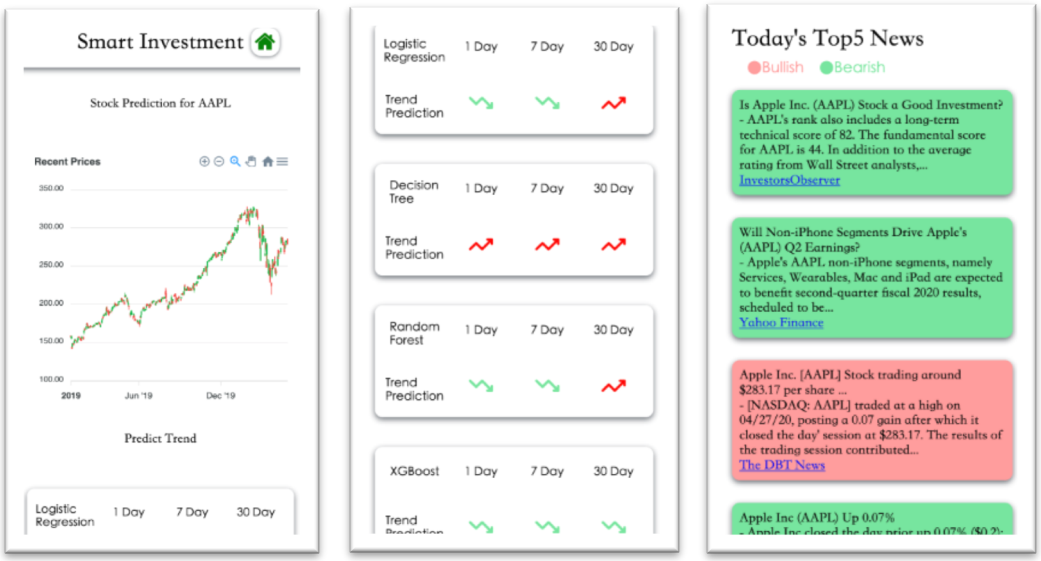


Figure 26. Responsive Design

2.5 Test

2.5.1 Test the News Scraper

To test the data scraper, we've compared the crawl result to the real google news search page.

Status: Valid

2.5.2 Test Stockstats Library and Label Correctness.

Although we use stock stats library to retrieve the financial indeicators, since it is not a popular library, we still tested it by calculating some of the financial indicators from scratch and compare the result. Moreover, since the label is extremely important, we also double-checked the trend label is following the criteria (if rise > 0, label it with 1; else label it with 0)

Status: Valid

2.5.3 Test the Backend Server

For each API, after construction, we will first use Postman to test if the data can be received. After that, we will then work on NodeJS fetch and console the response to see if the two data are identically the same.

Status: Valid

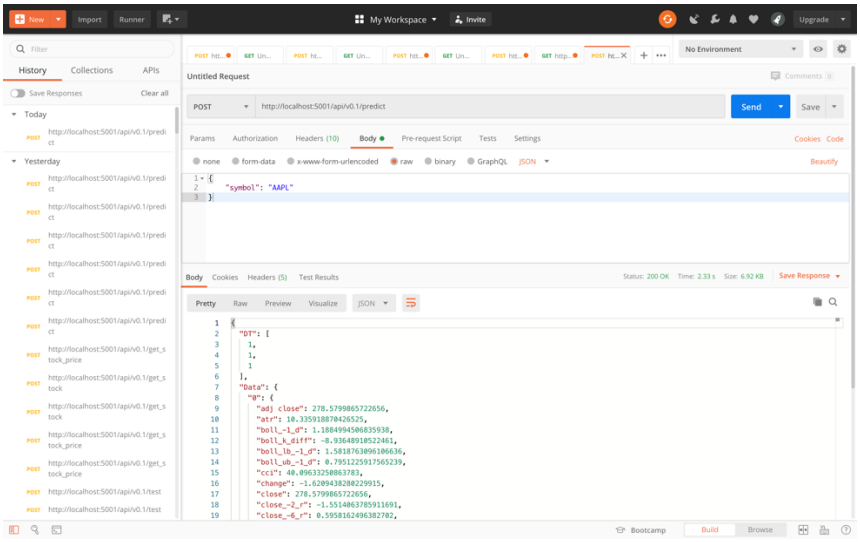


Figure 27. Postman Testing Result

2.5.4 Test the Front End User Interface

To test the front end user interface, we will go through the whole process and see if we can input different symbols, and the website can successfully route to the detail webpage or not.

Status: Valid

2.6 Evaluation

In this project, most of the evaluations could be found right below the training model explanation. Here we conclude some major findings.

In this project, most of the evaluations could be found right below the training model explanation. Here we conclude some major findings.

1. Difference between stock data with and without absolute price

   Generally, training stock data with absolute value had better performance compare to training stock data without it. However, stock data without "absolute" value can help prevent from overfitting for some models (ex: LSTM)

2. 1-Day Trend Prediction

   Except for the Reinforcement Learning, all models perform poorly in the prediction of 1-Day trend while train accuracy is disproportionately higher than test accuracy, which shows that there is serious overfiting in 1-Day trend prediction.

3. 7/30 Day trend prediction

   XGBoost and the Random Forest model get a high test accuracy in 7 and 30-day trend prediction. Those two models have great potential in stock prediction.

4. Reinforcement Learning

   The agent successfully triple the worth in the dynamic stock trade simulation environment where the profit depends on the accuracy of 1-day trend prediction while other models perform poorly in it. So, reinforcement learning is potentially powerful in 1-day trend prediction which other models are not suitable for.

5. ATR

   Average True Range (ATR) gets the highest Gini-importance in many models. It could have high reliability in the prediction of the stock price.

6. Feature influences

   For 1 Day trend prediction, news sentiment scores have more impact, following by the financial indicators, and the external features.

   However, when it comes to 7/30 days trend prediction, the financial indicators dominate the performance, following by external features and news sentiment scores.

7. LSTM Real Price Prediction

   Using 3 Day sequence of data could best predict the real stock price. The RMSE(root mean squared error) is only 15.3. We also observe some oscillation when the number of sequences of day increases, which may need further examination and explanation.

8. Backend/ Frontend Implementation

   The frontend and backend are well implemented, but we didn't ask for other's advice. Before final presentation we will seek for other's recommendations to improve our website as well as the user friendliness.

2.7 Conclusion

For this project, we've constructed a system to predict and explain the stock price behavior. It includes a scrawler to obtain financial news and stock price, several different classifiers to predict stock price trend on a 1 day/7 day/ 30 day basis, a LSTM model to conduct both real stock price prediction and trend prediction, a backend server to transmit prediction results, and a front end website for users to visualize the prediction. We are satisfied for the best prediction accuracy which could reach 90% when predicting 30 day trends. Moreover, we are glad to see that the LSTM real price prediction is close enough to the real stock price (with only 15 RMSE), since we've failed many times for constructing a well-performed model. We believe the reason behind is because we have not only collected enough data and indicators but also evaluate the models in a systematic and rigorous approach.

Here we respond to the goals we set at the beginning of the project:

1. Does the machine learning models we build perform well according to its accuracy, precision, and all other metrics?

   For 1 Day trend prediction, the answer is no. We can only get 55% of accuracy. However, for both 7 day trend and 30 day trend prediction, we could achieve more than 75% accuracy with good precision. Therefore we believe we partially achieved our goal.

2. Is the user interface user-friendly?

   Yes, the user interface is well designed. The webpage is responsively designed, and we also made use of icons to display our trend predictions.

3. Is the data crawler working stably?

   Yes, we can scrawl daily financial news from google news periodically.

4. Does the news and tweets have the influence to the stock price?

   Due to data privacy issue and lack of resources, we did not include tweets sentiment scores in our model. However, for news, following the observation we claimed in 2.3.8, it do have the influence to stock's 1-day trend prediction.

5. Can we explain our prediction according to the model we built?

   For Decision Tree and Logistic Regression results, since they are naturally explainable, we can explain our results. However, for LSTM, Random Forest, and XGBoost classifier, we cannot explain due to their black-box nature. However, we've tried some other attempts to explain how the feature influence the results, please refer to 2.2.6 and 2.3.8.

2.8 Future Work

1. We wish to explore longer forecast horizons, for example 1 year. However, we should also consider financial statements data such as earnings, assets …etc., to predict long term trends of the stock price.

2. We wish to explore other forecasting techniques such as autoregressive integrated moving average(ARIMA) and triple exponential smoothing(ie. Holt-Winters method)

4. The training period we chose skipped from the financial crisis. Therefore, for future work, we may need to include data in between the period of financial crisis. Since we are now in a pandemic and a financial crisis, if we wish to make our model more convincing, we should extend the training data period.

5. All classifier models are predicting trends, which we wish in the future we can find out a methond to make use of these classifiers to predict real stock price.

6. LSTM model needs more testing, especially for layer design, the experiments we've done still have limitations.

7. The accuracy for predicting 1-day trend is pretty low. According to the evaluation, train accuracy are extremely high, while validation accuracy and test accuracy are low, meaning that the feature we select may not be sufficient for one day trend prediction. Therfore, in the future, we wish to add more features to improve the result when predicting one day trend.

8. In the project, we used a general sentiment analysis method for obtaining sentiment score. However, the common practice is to train a specific NLP model for financial terms and scores. Therefore, in the future, we wish to build an NLP model focusing on financial terms to get the sentiment analysis score.

3 Project Planning

3.1 Distribution of Work

Table 16

Task	CHEN Liang-yu	SHAO Yuming	FAN Ziqian
Do the Literature Survey	○	○	●
Analyze Social Networks	○	●	○
Design Data Crawling Techniques	○	●	○
Design the Database	●	○	○
Design the User Interface	●	○	○
Design Data Mining Algorithms	●	○	○
Design NLP Module	○	○	●
Design Prediction Model	○	○	●
Develop the Data Crawler	○	●	○
Build the Database	●	○	○
Build the User Interface	●	○	○
Develop the Data mining Algorithms	●	○	○
Develop NLP Module	○	○	●
Develop Prediction Model	○	●	○
Test the Web Crawler	○	●	○
Test the Database	○	●	○
Test the User Interface	○	○	●
Test the Data Mining Algorithms	●	○	○

Test the NLP Model	●	○	○
Test the Prediction Model	○	○	●
Perform Integration Testing	○	○	●
Write the Proposal	○	●	○
Write the Monthly Reports	○	○	●
Write the Progress Report	●	○	○
Write the Final Report	○	○	●
Prepare for the Presentation	○	●	●
Design the Project Poster	○	●	○

3.2 GANTT Chart

Table 17

Task	July	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr
Do the Literature Survey										
Analyze Social Networks										
Design Data Crawling Techniques										
Design the Database										
Design the User Interface										
Design Data Mining Algorithms										
Design NLP Module										
Design Prediction Module										

Develop the Data Crawler										
Build the Database										
Build the User Interface										
Develop the Data mining Algorithms										
Develop NLP Module										
Develop Prediction Module										
Test the Web Crawler										
Test the Database										
Test the User Interface										
Test the Data Mining Algorithms										
Test the NLP Module										
Test the Prediction Module										
Perform Integration Testing										
Write the Proposal										
Write the Monthly Reports										
Write the Progress Report										
Write the Final Report										
Prepare for the Presentation										
Design the Project Poster										

4 Required Hardware & Software

4.1 Hardware

Development PC:	PC with basic operation system
GPU:	NVIDIA 2080 Ti
Server PC:	Linux CentOS Server

4.2 Software

MongoDB	For our database
React, SCSS, NodeJS	Web programming languages
Bloomberg API	Historical Data Source
Selenium, BeautifulSoup	For web crawler
Flask	For backend server
Keras, Tensorflow	For deep learning model
Scikit-learn	For machine learning model
Stockstats	For calculate Financial indicators
Yfinance	Download stock data
Textblob	For general sentiment analysis

5 References

[1] "Stock market prediction," En.wikipedia.org, 6 6 2019. [Online]. Available: https://en.wikipedia.org/wiki/Stock_market. [Accessed 9 6 2019].

[2] D. R. HARPER, "Forces That Move Stock Prices," Investopedia, 8 6 2019. [Online]. Available: https://www.investopedia.com/articles/basics/04/100804.asp. [Accessed 9 6 2019].

[3] J. Patel, S. Shah, P. Thakkar and K. Kotecha , "Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques," *Expert Systems with Applications,* p. 259, 2015.

[4] K. Chen, Y. Zhou and D. Fangyan, "A LSTM-based method for stock returns prediction : A case study of China stock market," *2015 IEEE International Conference on Big Data (Big Data),* 2015.

[5] B. Johan, M. Huina and Z. Xiaojun, "Journal of Computational Science 2," 2011.

[6] R. P. Schumaker and H. Chen, "Textual Analysis of Stock Market Prediction Using Breaking Financial News: The," *ACM Transactions on Information Systems (TOIS),* vol. 27, no. 2, 2009.

[7] B. Vanschoenwinkel, "A discrete Kernel Approach to Support Vector Machine Learning in Language Independent Named Entity Recognition," 2003.

6 Appendix A: Meeting Minutes

6.1 Minutes of the 1st Project Meeting

Date: June 4, 2019

Time: 8:00 pm

Place: Online Video

Present: CHEN Liangyu, SHAO Yuming, FAN Ziqian

Absent: None

Recorder: FAN Ziqian

1. Approval of minutes

Since this was the first formal group meeting, there were no approval of minutes.

2. Report on progress

2.1 All team members have read the soft copy of instructions of the Final Year Project and proceeded to conduct a certain level of research for the prediction of stocks.

2.2 CHEN, SHAO and FAN placed their focus on the research of existing system of Stock prediction.

2.3 All team members have read the FYP reports in the past semesters which were provided by Prof. Dik-Lun LEE.

3. Discussion items

3.1 Knowing that this project is to build a system for the performance of stock prediction in helping people to make decisions for stock investments, our team

has made the following decisions. Initially, the team continued in designing the relevant functions for the system.

3.2 For the back-end of the system, the team will build the data crawler to get enough data from a specific stock. Therefore, SHAO will start to learn professional knowledge of stock and know the model of traditional stock forecasting.

3.3 For the interface, FAN needs to use Python and JS to realize the intersection of different data.

3.4 CHEN mentioned about the recent and commonly used finance model for stock prediction in nowadays and he decided to add more features to a fundamental model.

4. Goals for the coming week

4.1 All group members will get knowledge of traditional stock forecasting and other popular stock prediction system.

4.2 FAN will set up a server for front-end interface.

4.3 SHAO will design a web crawler for data collection.

4.4 CHEN will design a practice model for stock prediction.

5. Meeting adjournment and next meeting

The meeting was adjourned at 11:30 pm.

The next meeting will be at August 15. The exact time will depend on all the members.

6.2 Minutes of the 2nd Project Meeting

Date: August 20, 2019

Time: 8:00 pm

Place: Online Video

Present: CHEN Liangyu, SHAO Yuming, FAN Ziqian

Absent: None

Recorder: FAN Ziqian

1. Approval of minutes

   Because of the time conflict between SHAO and FAN, the minutes of second meeting were approved at August 20.

2. Report on progress

   2.1 SHAO had designed a news-crawler for data collection.

   2.2 FAN built a server using Node JS and MongoDB.

   2.3 CHEN came up with the finance model and computer algorithms for the system.

3. Discussion items

   3.1 The team decided to build this news-crawler for the system and set up other crawlers for different kinds of stocks.

   3.2 Because the team had set clear objects on interface and crawler, SHAO thought that the team need build the database which can realize data transfer through API.

   3.3 FAN considered that we should do a project with the model that is different from previous projects.

4. Goals for the coming week

4.1 All group members prepare for the proposal.

4.2 Each member should read at least three relative research essays.

4.3 SHAO will use API to realize the data transfer.

5. Meeting adjournment and next meeting

The meeting was adjourned at 10:00 pm.

The next meeting will be organized after meeting the professor.

6.3 Minutes of the 3rd Project Meeting

Date: September 2, 2019

Time: 4:00 pm – 5:30 pm

Place: Online Video

Present: CHEN Liangyu, SHAO Yuming, FAN Ziqian

Absent: None

Recorder: FAN Ziqian

1. Approval of minutes

   As FAN got injured and staied home, this meeting is online.

2. Report on progress

   2.1.CHEN read some papers about predicting stock price with neural network.

   2.2.CHEN and FAN read some papers about NLP used to analyze public mood and financial news.

3. Discussion items

   N/A

4. Goals for the coming week

   4.1.Continue woeking on collecting papers and design the system

5. Meeting adjournment and next meeting

   The meeting was adjourned at 5:30 pm.

6.4 Minutes of the 4th Project Meeting

Date: October 10, 2019

Time: 7:00 pm – 11:00 pm

Place: HKUST Library

Present: CHEN Liangyu, SHAO Yuming, FAN Ziqian

Absent: None

Recorder: FAN Ziqian

1. Approval of minutes
2. Report on progress

   2.1. We finished the draft of this project.

   2.2. Each team member finished his own part of the proposal.

3. Discussion items

3.1. Finalize the proposal.

4. Goals for the coming week

4.1. CHEN: building the model of price prediction.

4.2. FAN: designing the NLP part

4.3. SHAO: work on the web crawler

5. Meeting adjournment and next meeting

The meeting was adjourned at 11:00 pm. The proposal is submitted.

6.5 Minutes of the 5th Project Meeting

Date: February 17, 2020

Time: 4:30 pm – 6:00 pm

Place: HKUST Library

Present: Prof. LEE, Dik-Lun, CHEN Liangyu, FAN Ziqian, SHAO Yuming

Absent: None

Recorder: FAN Ziqian

1. Approval of minutes

After discussing with Prof. LEE, we decided this time to meet.

2. Report on progress

2.1. Finish of Random forest, XGboost, and LSTM model.

2.2. Future of NLP modular and more prediction models.

3. Discussion items

    3.1. Prof. LEE provided some suggestion about future plan of the project.

    3.2. Normalization of financial data.

4. Goals for the coming week

    4.1. CHEN: building the model of price prediction.

    4.2. FAN: shelve NLP, made a reinforcement learning model

    4.3. SHAO: work on the web crawler

5. Meeting adjournment and next meeting

    The meeting was adjourned at 6:00 pm.

6.6 Minutes of the 6th Project Meeting

Date: February 27, 2020

Time: 8:00 pm – 10:30 pm

Place: Zoom Meeting

Present: CHEN Liangyu, FAN Ziqian, SHAO Yuming

Absent: None

Recorder: FAN Ziqian

1. Approval of minutes

    After meeting with our FYP communication tutor on Zoom, we decided this time to meet.

2. Report on progress

2.1. Future of NLP modular and more prediction models.

2.2. Try a reinforcement learning model

3. Discussion items

3.1. Discuss the problem of big data and data set issue raised by Communication tutor.

3.2. Learn to justify the model through the pipline or experiment to get the conclusion of the model.

4. Goals for the coming week

4.1. CHEN: building the model of price prediction.

4.2. FAN: shelve NLP, made a reinforcement learning model

4.3. SHAO: work on the web crawler

5. Meeting adjournment and next meeting

The meeting was adjourned at 10:30 pm.

6.7 Minutes of the 7th Project Meeting

Date: April 1, 2020

Time: 7:00-9:00 pm

Place: Zoom Meeting

Present: CHEN Liangyu, FAN Ziqian, SHAO Yuming

Absent: None

Recorder: FAN Ziqian

1. Approval of minutes

   Because of the Covid-19, we chose the online meeting.

2. Report on progress

   2.1. Added more feature in our data processing to get a higher accuracy.

   2.2. Worked on NLP modular and more prediction models.

   2.3. Got a result of the reinforcement learning model, however the accuracy was not good enough.

3. Discussion items

   3.1. Discuss the existing problem of reinforcement learning model.

   3.2. Sort out the features we used and try to calculate their correlation.

   3.3. All the member get familiar with the features in data-processing.

4. Goals for the coming week

   4.1. CHEN: work on the correlation calculation.

   4.2. FAN: still working on the reinforcement learning model.

   4.3. SHAO: work on report and feature engineering.

5. Meeting adjournment and next meeting

   The meeting was adjourned at 9:00 pm.

6.8 Minutes of the 8th Project Meeting

Date: April 21, 2020

Time: 3:00 pm – 6:00, 7:00-9:00 pm

Place: Zoom Meeting

Present: CHEN Liangyu, FAN Ziqian, SHAO Yuming

Absent: None

Recorder: FAN Ziqian

1. Approval of minutes

   Before/After meeting with our FYP communication tutor on Zoom, we decided this time to meet.

2. Report on progress

   2.1. Still working on a reinforcement learning model.

   2.2. Calculated the correlation among Financial indicators, Gold and Sentiment.

3. Discussion items

   3.1. Discuss the existing problem of reinforcement learning model.

   3.2. Plan to do a PPT to explain our work to someone who don't know our project.(Suggested by communication tutor)

   3.3. All the member get familiar with the features in data-processing.

4. Goals for the coming week

   4.1. CHEN: work on the backend and front-end

   4.2. FAN: still working on the reinforcement learning model.

4.3. SHAO: work on report and help Chen with the backend.

5. Meeting adjournment and next meeting

The meeting was adjourned at 9:00 pm.