Structure

1 - Overview

This project aims to collect text data related to certain stocks from social media and official news and then perform sentimental analysis and other kinds of data analysis on the collected data.
Such information is then further filtered and leveraged (probably together with the history stock data), to try to predict the stock index movement of the selected stocks using machine learning techniques.

1.1 - Stocks of Interest

As required, we only consider the current FTSE China A50 index constituents. We use python lists to store the information like codes, names and industries of these fifty stocks.

	序号	证券代码	公司名称	Company Name	行业
0	1	000002	万科	China Vanke A	房地产
1	2	601360	三六零	SJEC Corp	信息技术
2	3	600104	上汽集团	SAIC Motor Corp	汽车
3	4	600018	上港集团	Shanghai International Port	交通运输
4	5	600030	中信证券	CITIC Securities	金融业
5	6	601998	中信银行	China Citic Bank A	银行业
6	7	601766	中国中车	CRRC A	交通运输
7	8	601390	中国中铁	China Railway A	重工
8	9	601800	中国交建	China Communications Construction	重工
9	10	601628	中国人寿	China Life Insurance A	保险业

1.2 - Previous Work

After some research, I find that the majority of work already done on this topic focuses solely on collecting text data without the search of a particular keyword (just collect all twitters / weibos from the Internet and perform some rudimentary filtering).
Then, the text data is used mainly to predict the movement of some **index** as a whole, rather than the prices of some individual stock. In other words, previous work is mostly done on the graininess of the entire market, never so specific as to study any individual stock.

Representative:

- Stanford: Using Twitter to predict Dow Jones Industrial Average
  http://cs229.stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis.pdf
- Shanxi University: Using Weibo to predict average of stock prices
  http://cdmd.cnki.com.cn/Article/CDMD-10108-1018312077.html

2 - An Unsuccessful Attempt to Scrape Text Data Directly

My first experiment to obtain data is to manually collect text data from two sources. This experiment turns out to be unfruitful.

2.1 - Sina Weibo Scrawling

The first source I choose is Sina Weibo, the major social media platform where the general public in Mainland China express their thoughts and views casually.

Although Sina Weibo has its own API, it is heavily restricted and does not support the function of "search by keyword" very well. Therefore, I wrote my own scrawler to go through pages one by one and collect Weibo live.

For example, We can collect one page of Weibo related to the first stock 万科 and sort them into one pandas dataframe to have a casual look.

mid	comments	likes	reposts	source	text	time	user_gender	user_verified
4413883316134334	0	0	0	微博 weibo.com	<a href="https://m.weibo.cn/search?containeri...	刚刚	m	True
4413881541356044	0	0	0	iPhone客户端	记昨天。图二万科后面的骨汤麻辣烫。	7分钟前	f	False
4413880648347677	17	15	27	石家庄超话	#正定文化村#这个中秋你想怎么过？不如来万科&园博园中秋嘉年华！这里有奇幻的水幕电影、震撼的...	11分钟前	m	True
4413880136434605	4	4	4		【深圳回迁房"灰色"生意：违建变身拿"红本" 旧改3年房价翻5倍】短短5个月，凭借一套位于深...	13分钟前	m	True
4413879897164478	0	0	0	iPhone客户端	🎉🎉🎉良渚万科未来城二期小高层稀缺三房两卫带车位月租只要3900 只要3900 👏👏 <...	14分钟前	f	False
4413876831816569	0	0	0	iPhone客户端	合景退房成功 林给我买万科大都会滨江	26分钟前	m	False
4413874516763746	0	3	1	微博 weibo.com	上市企业员工都坚决不买自家上市企业股票，包括贵州茅台，格力电器，万科，其中原理实际也不难分析，	35分钟前	m	True
4413871097596495	0	0	0	荣耀V20 4800万3D相机	万科翡翠云图市集——红孩儿也来出集啦	48分钟前	m	True
4413870988330221	0	0	0	HUAWEI Mate 20	这是包手啦继续添加<img alt=[good...	49分钟前	m	True
4413870850223413	0	0	0	iPhone客户端	万科可以啊，给业主转基因大豆油！怎么想的？！ <...	49分钟前	f	False

As one can see, the above information, if in large scale, has quite a lot potential to exploit.

Unfortunately, further collection presents a somewhat insuperable obstacle.

Due to the limits imposed by Weibo, the best I can achieve is collecting around 300 weibos for each of the 50 stocks. (Demo).


```
"mid": "4412748823707166",
"time": "4小时前",
"source": "微博 weibo.com",
"user_verified": true,
"user_gender": "m",
"reposts": 0,
"comments": 0,
"likes": 0,
"text": "中国股市：周三持有以下票的朋友，有拿不准或被套的下方评论或留言，盘中精选强势股，尽在蔚
    <br />(600018)上港集团(600019)宝钢股份(600020)中原高速(600021)上海电力<br
    />(600022)山东钢铁(600023)浙能电力(600025)华能水电(600026)中远海能 "
```


300 Weibos cover a period of roughly 4-7 days, which is far from enough for extracting features, combining with daily stock prices and then training for models.

2.2 - Twitter Scrawling

Adapted from my own previous work at Berkeley, simply using Tweepy API (instead of manually scraping) to collect tweets related to certain keyword. Demo Below.

Note: Here, we search by the English name of the corresponding stock rather than the Chinese name.


```
{"created_at": "Tue Sep 03 05:33:34 +0000 2019", "id": 1168758916760555520, "id_str": "1168758916760555520",
   "full_text": "All of a sudden the Yen-basis widening is making sense. First GPIF &amp;amp; now
   this:\n\nDai-ichi Life Insurance —~$3.7t in assets- \"cut holdings of stocks and increased currency
   hedging on foreign bonds as U.S.-China trade frictions have escalated\" \n\nhttps://t.co/E6RXicWB9y",
   "truncated": false, "display_text_range": [0, 270], "entities": {"hashtags": [], "symbols": [],
   "user_mentions": [], "urls": [{"url": "https://t.co/E6RXicWB9y", "expanded_url":
   "https://www.theguardian.pe.ca/business/reuters/
```


Similar to the situation with Weibo, the best I can achieve is collecting around 500 tweets for each of the 50 stocks, which again, covers a period that is way too short to conduct any meaningful research.

3 - News Data Collection and Processing

3.1 News Collection and Aggregation - Using Tushare

After some research, I decided to use the API of **Tushare Pro**, an open source platform designed for financial big data focused on Mainland China.

Using this API, I can directly download news within a given period, with little to no constraints on the request frequencies. This platform allows me to access news from five sources, namely, 新浪财经，华尔街日报，同花顺，东方财富，云财经.

After some efforts and using some tricks, we can successfully collect all news from all of these five sources during the period **2019.01.01-2019.8.31** and save them locally for future accesses.

We can combine all news we get from these five different sources to form one single giant pandas Dataframe with 380k news inside.

```
news_aggregated = pd.concat(news_by_source, sort=True).fillna(value=0).iloc[:,1:]
news_aggregated.head()
```

	content	datetime	source	title
0	市场消息：Uber的估值促使零工经济面临审查。	2019-01-01 23:39:43	sina	0
1	【ST慧球：上海高院一审宣判 公司无需对顾国平债务承担担保责任】ST慧球(600556)1月...	2019-01-01 23:25:29	sina	0
2	【FAANG股分析之奈飞：2018年完胜FAANG股同行 但继续烧钱或使其2019年的股价承...	2019-01-01 23:06:07	sina	0
3	【FAANG股分析之苹果：苹果股价走低可能持续到2019年】苹果在2018年全年累跌近7%，...	2019-01-01 22:52:53	sina	0
4	【陕西省"民参军"企业已达589家】陕西省近日出台了一揽子优惠政策扶助民营企业参加军工生产，...	2019-01-01 22:42:33	sina	0

```
news_aggregated.shape
```

```
(379598, 4)
```

3.2 - Data Processing

Though at first it may appear natural to process these 380k news altogether in several batches, it turns out impossible to carry out certain operations like sentimental analysis in such a large scale.

Therefore, I decided to perform data analysis for each stock individually.

Basically, a pipeline to process these text data from news is established and then later applied to each of the fifty stocks of interest.

For illustrative purpose, I choose a specific stock, 万科 000001, and build the pipeline step by step using this particular stock as an example.

3.2.1 - Filtering (Only Keeping News related to This Stock)

What comes first is a somewhat bold filtering. We only keep those news that are related to our stock.

The word "related" is defined by specifying the following two rules:

1) The name of this stock appears in this piece of news, in this case, 万科

2) The industry of this stock appears in this piece of news, in this case, real estate, or, 房地产

```
name_appeared = news_aggregated[news_aggregated['content'].str.contains(stock_names_chn[0]).fillna(value=False)]

print(name_appeared.shape)
name_appeared.head()
```

```
(624, 4)
```

	content	datetime	source	title
177	【2018年千亿房企达30家 恒大夺销售权益榜首位】克而瑞发布的《2018年度中国房地产企业...	2019-01-01 09:02:56	sina	0
1758	【在港上市地产股集体大涨】长实集团在香港一度升5.4%，创下2016年3月来最大上涨。恒基地...	2019-01-04 15:24:33	sina	0
2002	【万科：2018年实现合同销售金额6069.5亿元】万科早间公告称，2018年12月份公司实...	2019-01-04 07:45:01	sina	0
2003	【万科：2018年实现合同销售金额6069.5亿元】 万科A早间公告：2018年12月份公司...	2019-01-04 07:43:54	sina	0
2131	【杭州万科回应作家投诉：两年来进行十余次沟通 已诉诸法律途径】针对作家张艳华投诉项目漏水一事...	2019-01-05 15:04:03	sina	0

```
industry_appeared = news_aggregated[news_aggregated['content'].str.contains(stock_industries[0]).fillna(value=False)]

print(industry_appeared.shape)
industry_appeared.head()
```

```
(4440, 4)
```

	content	datetime	source	title
10	【重庆调整房产税起征点 专家：这是例行调整】1月1日起，重庆市主城个人新购高档住房房产税起征...	2019-01-01 22:23:09	sina	0
69	【方正固收提出10个论断：经济增长拐点不会在2019出现】1、2019年的信用扩张很难起来。...	2019-01-01 17:09:56	sina	0
105	【王府井：长春赛特奥莱MALL对外营业】王府井公告，2018年12月30日，公司旗下长春赛特...	2019-01-01 15:38:20	sina	0
112	【天津：深化房地产市场调控 2018年住房交易量价稳定】去年以来，我市认真贯彻党中央、国务院...	2019-01-01 15:23:31	sina	0
120	【中国银行与浦发银行展开海南FT账户体系下多个品种业务办理】海南自由贸易账户(FT账户)体系...	2019-01-01 14:49:16	sina	0

Now, we can see that the major problem, which is the difficulty of scraping the Internet with particular keyword is solved by simply retrieving a huge amount of news with no keyword and bring the filtering offline to our own local machines.

3.2.2 - Sort According to Date

Currently, the news data is aggregated from different sources. We want to sort it according to date so that it can be in a "time-series" form.

content	source	title	date
【2018年千亿房企达30家 恒大夺销售权益榜首位】克而瑞发布的《2018年度中国房地产企业...	sina	0	2019-01-01
【2018年千亿房企达30家，恒大夺销售权益榜首位】据2018年12月31日克而瑞发布的《2...	wallstreetcn	0	2019-01-01
【2018年千亿房企达30家 恒大夺销售权益榜首位】克而瑞发布的《2018年度中国房地产企业...	wallstreetcn	0	2019-01-01
云财经讯，万科A公告：2018年12月份公司实现合同销售面积438.7万平方米，合同销售金额...	yuncaijing	万科A公告：2018年12月份公司实现合同销售面积438.7万平方米	2019-01-04
云财经讯，万科A：2018年1-12月累计合同销售额6069.5亿元，合同销售面积4037....	yuncaijing	万科A：2018年1-12月累计合同销售额6069.5亿元	2019-01-04

3.2.3 - Sentimental Analysis

The step is where a sentimental score is assigned to each and every piece of news we collected, indicating how positive / negative this new is.

We use the API from Baidu AI Lab to accomplish this task, the result we get from Baidu indicates the probability that this news is positive.

	content	source	title	date	sentiment_score
177	【2018年千亿房企达30家 恒大夺销售权益榜首位】克而瑞发布的《2018年度中国房地产企业...	sina	0	2019-01-01	0.839714
166	【2018年千亿房企达30家，恒大夺销售权益榜首位】据2018年12月31日克而瑞发布的《2...	wallstreetcn	0	2019-01-01	0.914582
167	【2018年千亿房企达30家 恒大夺销售权益榜首位】克而瑞发布的《2018年度中国房地产企业...	wallstreetcn	0	2019-01-01	0.839714
1257	云财经讯，万科A公告：2018年12月份公司实现合同销售面积438.7万平方米，合同销售金额...	yuncaijing	万科A公告：2018年12月份公司实现合同销售面积438.7万平方米	2019-01-04	0.163503
1254	云财经讯，万科A：2018年1-12月累计合同销售额6069.5亿元，合同销售面积4037....	yuncaijing	万科A：2018年1-12月累计合同销售额6069.5亿元	2019-01-04	0.366212

After obtaining the sentiment scores from the collected and filtered news, we can actually dump the news itself, only keeping the useful information and form a new pandas Dataframe.
We save the dataframe locally with proper naming to save us from pinging Baidu AI each time we want such data in the future.

	date	sentiment_score	source
177	2019-01-01	0.839714	sina
166	2019-01-01	0.914582	wallstreetcn
167	2019-01-01	0.839714	wallstreetcn
1257	2019-01-04	0.163503	yuncaijing
1254	2019-01-04	0.366212	yuncaijing

We do exactly the same thing with the news related to industry that we filtered in the last step.
One thing to note is that local saving is actually of much more importance here since we can avoid the extra work for future stocks in the same industry, in this case, real estate.

3.2.4 - Kernel Smooth Method to Get Score in Each Day

Now that we have obtained the sentimental scores for all news that are directly or indirectly related to the stock that we are studying, we further process these data.
First, we group by date, and then for each day, simply take the average of sentiment scores for all news that are published on that day, regardless of its source.

date	sentiment_score
2019-01-01	0.864670
2019-01-04	0.311315
2019-01-05	0.332797
2019-01-07	0.596884
2019-01-08	0.584160

Next, we create a dictionary mapping date to its sentiment score.

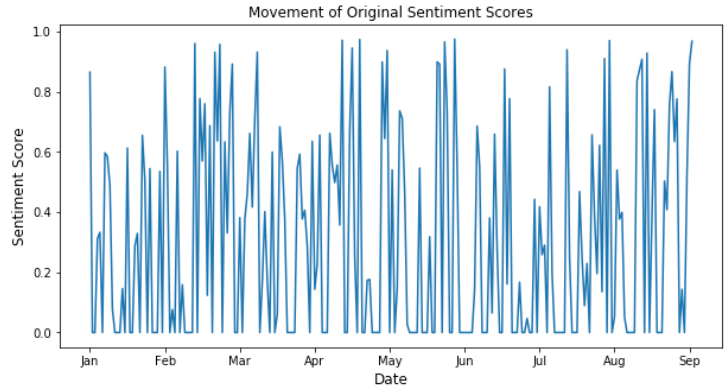
```
date_to_score_name = list(grouped_name.to_dict().values())[0]
date_to_score_name
```

```
{'2019-01-01': 0.8646699999999999,
 '2019-01-04': 0.31131512727272725,
 '2019-01-05': 0.33279713999999994,
 '2019-01-07': 0.596884,
```

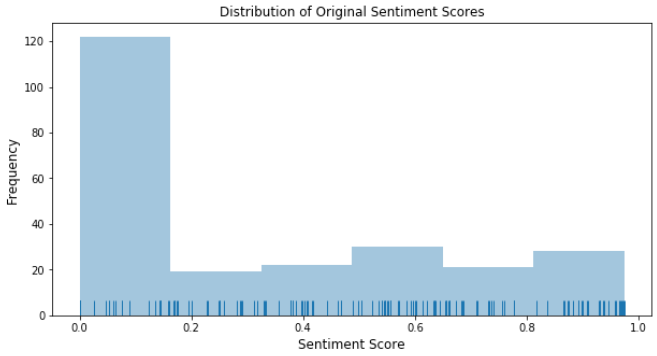
At this stage, not every single day has a sentimental score. Only those days when at least one piece of news is published has a score, which is not very sensible.

We can plot the scores for each and every day, and it is easy to observe that there are many days with score 0 and the curve is extremely volatile, not something desirable.

We can also take a look at the distribution at all the scores. There are roughly half of days that do not have a sentiment score.



We can also take a look at the distribution at all the scores. There are roughly half of days that do not have a sentiment score.



Here, I use the “one-sided” kernel smooth method to work around this issue and to comply with the actual situation.

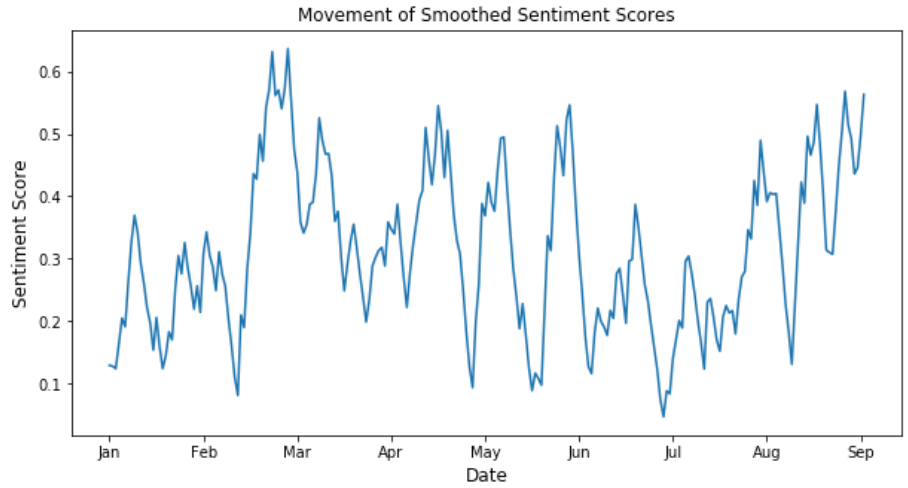
In real life, it is not the case that a news only has effect to the performance of the stock on the single particular day that this news is published. Instead, its influence should last for a couple of days and gradually decay to zero.

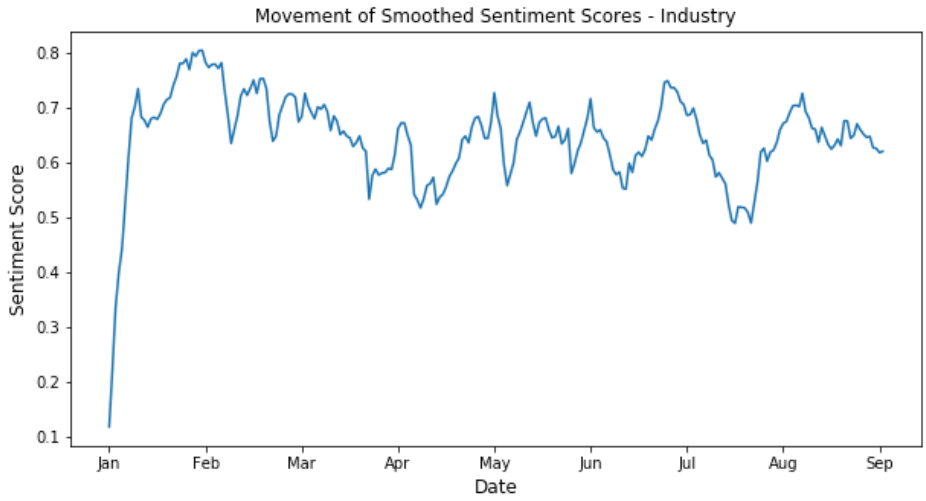
Three different kinds of kernels are implemented and are ready to be further tested upon.

One thing that is obviously different from the traditional kernel smooth method is that the kernel I implemented here is single sided rather than two sided.

```python
def kernel_density(kind, width, distance):
    if distance > width:
        return 0
    if kind == 'uniform':
        return 1 / width
    if kind == 'square':
        return 1.5 * (1 - math.pow((distance/width) ,2)) / width
    if kind == 'triangular':
        return 2 * (1 - (distance/width))/ width
```

Width of the kernel is set to 10 currently, subject to future fine tuning.





We can plot the smoothed scores and the result is not only much more desirable but also more realistic when considering real life situation.

One observation from the above plot is that the curve experiences a somewhat peculiar sharp increase at the beginning. This can be explained since the for first 10 (or whatever the width we set) days, their scores should come from the 10 days prior to the starting date of our entire news data set, which is of course, missing. Therefore, this sudden increase is actually quite reasonable and suggest that we should exclude the first 10 days from our training.

4 - Market Data Collection and Input Preparation

Still using the first stock in our list, 000001, Vanke, as an example, we retrieve the daily market data (open prices, close prices, high, low, volume...) for the concerned period.

	ts_code	trade_date	open	high	low	close	pre_close	change	pct_chg	vol	amount
0	000001.SZ	20190830	14.29	14.39	14.10	14.16	14.13	0.03	0.2123	798500.57	1136119.659
1	000001.SZ	20190829	14.22	14.24	14.08	14.13	14.27	-0.14	-0.9811	609034.34	861177.482
2	000001.SZ	20190828	14.26	14.28	14.05	14.27	14.31	-0.04	-0.2795	829657.68	1176329.095
3	000001.SZ	20190827	14.36	14.48	14.24	14.31	14.25	0.06	0.4211	1356713.37	1948127.123
4	000001.SZ	20190826	14.42	14.50	14.15	14.25	14.65	-0.40	-2.7304	1415122.54	2018498.039

At this stage, for this particular stock, we have actually acquired all input data. All there left to do is to sort them through and build a dataframe for further training and testing.
However, before doing that, several important **simplifications and assumptions** need to be made:

1) What we are trying to predict here is only the percentage change of some particular stock within one particular trading day. We consider this as a key measurement for how well the stock is performing and as an indicator for our trading strategy in later sections. Everything else including open price, close price and trading volume is regarded as side information and not predicted. (Some of them might be used for prediction, though.)

2) We assume that the performance of a stock is related to only the recent news and recent stock market data. For now, we define the word "recent" as "last three trading days" for market data and "last three days" for news. Of course, such assumption and definition are extremely arbitrary and is subject to possible great future changes to improve the performance of models.

3) We further simplify the market data at each day to three numbers: open price, percentage change and trading volume. Close prices are omitted here since it can be derived directly from open prices and percent changes.

To begin with, we start off simple and easy, we only use recent news sentiment scores (and not include recent market data) to predict the percent change of trading days.

Nevertheless, as other market data might be used for future model complication, we prepare 16 numbers (namely the open prices, percent changes and trading volumes of the previous three trading days and sentiment scores of the news in the past three days from both name-specific news and industry-specific news, plus 1 for output, i.e., percentage change on that day) for each and every trading day from the beginning of January to the end of August.

```
print(final_dataframe.shape)
final_dataframe.head()
```

```
(157, 16)
```

	open-1	open-2	open-3	pctchg-1	pctchg-2	pctchg-3	vol-1	vol-2	vol-3	name-1	name-2	name-3	industry-1	indutrys-2	industry-3
20190110	9.74	9.73	9.84	2.8986	-0.8214	-0.1026	1233486.36	402388.11	865687.66	0.369590	0.327061	0.262958	0.701542	0.680709	0.607741
20190111	9.87	9.74	9.73	1.6097	2.8986	-0.8214	1071817.66	1233486.36	402388.11	0.341526	0.369590	0.327061	0.734110	0.701542	0.680709
20190114	10.11	9.87	9.74	0.9901	1.6097	2.8986	696364.55	1071817.66	1233486.36	0.221808	0.260669	0.292360	0.664182	0.676530	0.682409
20190115	10.22	10.11	9.87	-0.8824	0.9901	1.6097	500443.59	696364.55	1071817.66	0.197657	0.221808	0.260669	0.679139	0.664182	0.676530
20190116	10.11	10.22	10.11	1.2859	-0.8824	0.9901	542160.55	500443.59	696364.55	0.154045	0.197657	0.221808	0.681913	0.679139	0.664182

5 - Building and Applying the Data Preparation Pipeline

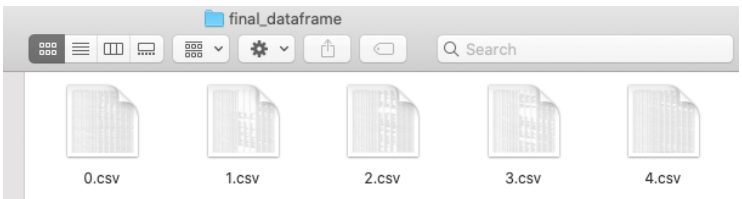
Now, we have gone through the entire process of preparing the input and output data for one stock.

Since eventually we will need to prepare such data for each and every stock from the fifty stocks of interest, it would greatly ease our life if we define several functions to modularize the whole process and build a pipeline for this data preparation process.

After all functions been defined for each step along the way, we can simply define another function to summarize the whole process, within which we call those functions defined above and meanwhile deal with some tiny details that may cause error.

```
def build_dataframe(i):

    # Filter the 380k news and extract those related to either this particular stock or industry
    name_appeared = filter_news(stock_names_chn[i])
    industry_appeared = filter_news(stock_industries[i])

    # Sort According to Date
    name_appeared = sort_by_date(name_appeared)
    industry_appeared = sort_by_date(industry_appeared)

    # Get sentiment scores for name appeared news and industry appeared news
    name_sentiment_result = get_sentiment_score(name_appeared, stock_names_chn[i])
    industry_sentiment_result = get_sentiment_score(industry_appeared, stock_industries[i])

    # Get smoothed scores
    smoothed_scores_name = get_smoothed_score(name_sentiment_result)
    smoothed_scores_industry = get_smoothed_score(industry_sentiment_result)

    # Get Five Dicts and Build Dataframe
    market_data = pro.daily(ts_code=stock_codes[i]+'.SZ', start_date='20190101', end_date='20190831')
    if market_data.shape[0] == 0:
        market_data = pro.daily(ts_code=stock_codes[i]+'.SH', start_date='20190101', end_date='20190831')
    final_dataframe = build_final_dataframe(*build_five_dicts(market_data, smoothed_scores_name, smoothed_scores_ind

    final_dataframe.to_csv('final_dataframe/' + str(i) + '.csv')

    return final_dataframe
```

Finally, we call the above function for each stock and get a data matrix for each stock that we save locally for the model training and testing in the next section.



6 - Training and Testing Linear Regression Models

As mentioned in section 4, the very first thing we try is using recent news sentiment scores (6 numbers only) to try to predict the percent change of trading days.

To start with, we use the most simple and fundamental model: linear regression.

6.1 - Universal Model

Based on the thought that the underlying logic of using recent news and market data to predict current stock performance should be homogenous rather than heterogenous for different stocks, the first attempt is aggregating all data from all stocks first and then training a universal model for prediction.

```
l = []
for i in range(50):
    l.append(pd.read_csv('final_dataframe/'+str(i)+'.csv'))
all_data = pd.concat(l)
print(all_data.shape)
all_data.head()
```

```
(7828, 17)
```

	Unnamed: 0	open-1	open-2	open-3	pctchg-1	pctchg-2	pctchg-3	vol-1	vol-2	vol-3	name-1	name-2	name-3	industry-1	industry-2	industry-3
0	20190110	25.40	25.05	25.29	1.3200	-0.1996	0.4813	340140.62	214382.02	427154.85	0.369590	0.327061	0.262958	0.701542	0.680709	0.607741
1	20190111	25.22	25.40	25.05	-0.8685	1.3200	-0.1996	224649.20	340140.62	214382.02	0.341526	0.369590	0.327061	0.734110	0.701542	0.680709
2	20190114	25.10	25.22	25.40	0.7567	-0.8685	1.3200	240694.90	224649.20	340140.62	0.221808	0.260669	0.292360	0.664182	0.676530	0.682409
3	20190115	25.13	25.10	25.22	-1.1067	0.7567	-0.8685	183281.34	240694.90	224649.20	0.197657	0.221808	0.260669	0.679139	0.664182	0.676530
4	20190116	25.00	25.13	25.10	0.5596	-1.1067	0.7567	368758.85	183281.34	240694.90	0.154045	0.197657	0.221808	0.681913	0.679139	0.664182

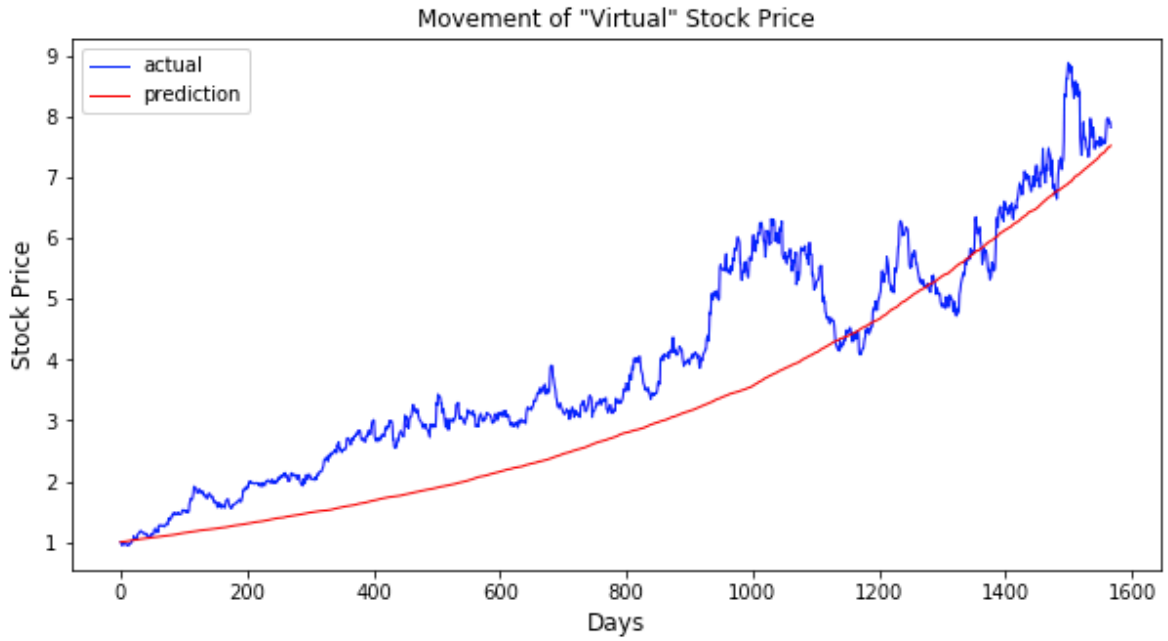
There are 7828 data points in total, we shuffle the data and randomly split into training set and testing set with the ratio of 4:1.

Then, a linear regression is fitted on the training data and tested on the testing data.

```
x = 1

p_actual = [1]
p_pred = [1]

for y_t in y_test:
    p_actual.append(p_actual[-1] * (1+(0.01*y_t)))
for y_p in y_pred:
    p_pred.append(p_pred[-1] * ((0.01*y_p)+1))

plt.figure(figsize=(10,5))
plt.title("Movement of \"Virtual\" Stock Price")
plt.xlabel('Days', fontsize=12)
plt.ylabel('Stock Price', fontsize=12)
plt.plot(p_actual, linewidth=1, color='blue', label='actual')
plt.plot(p_pred, linewidth=1, color='red', label='prediction')
plt.legend();
```

Here, to visualize how accurate our prediction after the entire dataset being randomly shuffled, we create a "virtual stock" by simulating its actual price using actual percentage change data (which comes from different stocks and is not sorted in chronological order) and simulating its predicted price using corresponding predicted percentage changes.

The result looks like the following:



We can see that although the prediction can somewhat capture the main trend (going up) of the actual movement, clearly it appears way too stable compared to the actual fluctuation of stock prices.
The reason is probably that we have used 6k~7k data points to train a linear regression with only 6 features. It is thus reasonable that we end up with the prediction algorithm being inordinately "conservative", in the sense that it will always predict a gentle, steady, slow increase no matter what input is given.

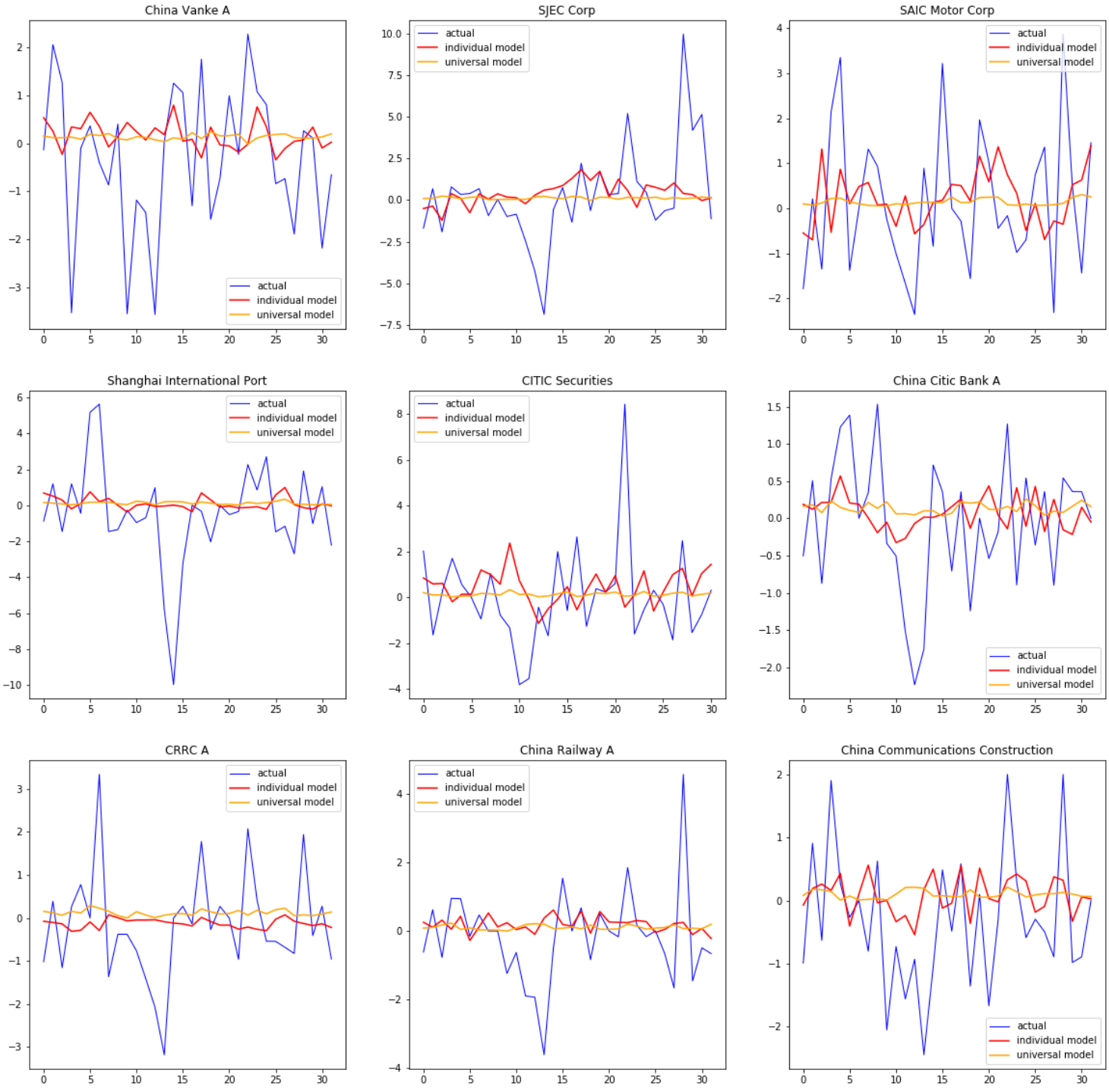
6.2 - Individual Models

6.2.1 – Percentage Change Prediction Results

Seeing that the result of training one universal model after aggregating data from all stocks is not that desirable, we try a different approach and train individual models for each stock.
Again, we use the simplest linear regression model, but instead of training one model over 7828 data points, we train 50 models, each over around 130 data points.
Here, since the testing set contains only around 30 data points, we can actually plot the predicted percentage changes and compare to the actual numbers. (Note that this is not possible in the previous section since 1000 data points would be too versatile can no meaningful conclusion can be drawn).

So, we compare the actual percentage changes, the predicted percentage changes using universal model and the percentage changes using the individual model.

The result looks like the following (only the results for the first 9 stocks are displayed):
We can observe that the individual model is clearly better than the universal model generally speaking. It is indeed capable of capturing some of those fluctuations.



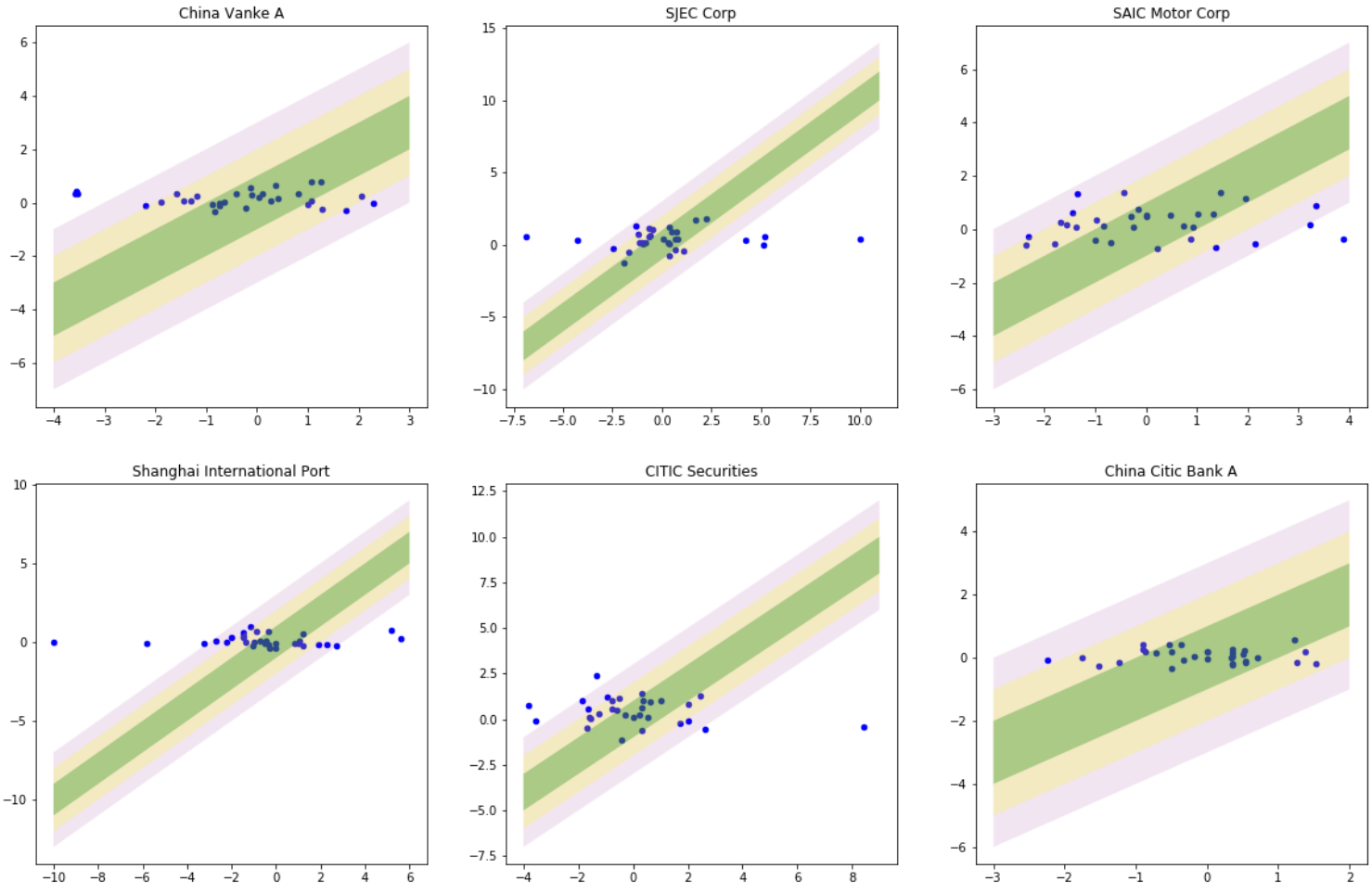
However, both predicted results are too “stable”, there are way less fluctuations and variance in the predicted percentage changes compared to the actual situation.

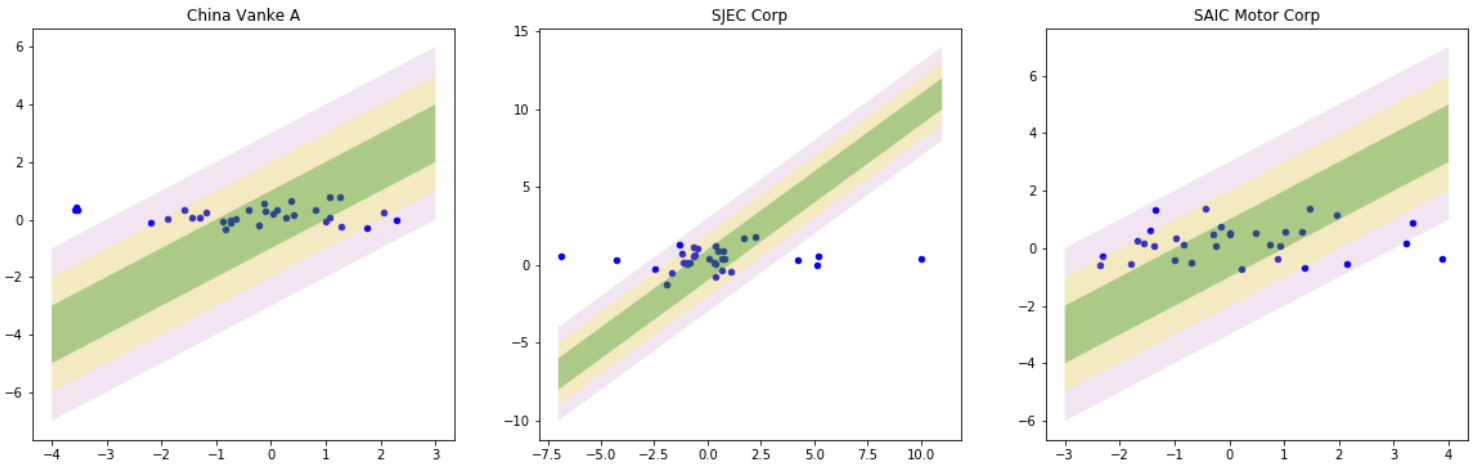
Another thing is that the changes predicted using individual model appears to be one or two days “later” than actual changes, which is also reasonable due to the untimeliness nature of nature.

The stock may respond immediately to events, market emotions and public opinions, but it would take time for the corresponding news to be created, edited, published and then used to predict prices.
Given that our model only uses news in the previous three days (not inclusive of today) to predict the percentage change today (which is sensible since you cannot know the news today in advance), this result is anticipated.

We can also plot the results using scatter plots to get a sense of how accurate our results are compared to the actual situation.

Below:
X-axis is the actual percentage changes; Y-axis is the predicted percentage changes individual models).
Green: +/- 1%, Yellow: +/- 2%, Purple: +/- 3%





We can see that while almost all points fall into the purple region (except some outliers), there are not so many points in the green region, meaning that the predicted percentage changes and the actual values indeed have some differences.

6.2.2 – Price Movement Prediction Results

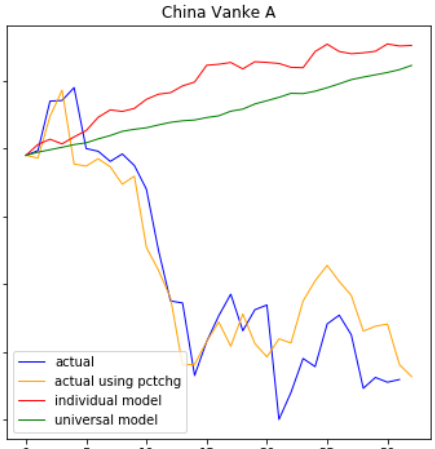
Here, because during the training of individual model, no random shuffling is performed on the dataset and chronological order is reserved, the testing data is actually the set of some consecutive trading days. Therefore, we can actually compare the "calculated price using predicted percentage change" to the actual price movements of any particular stock.

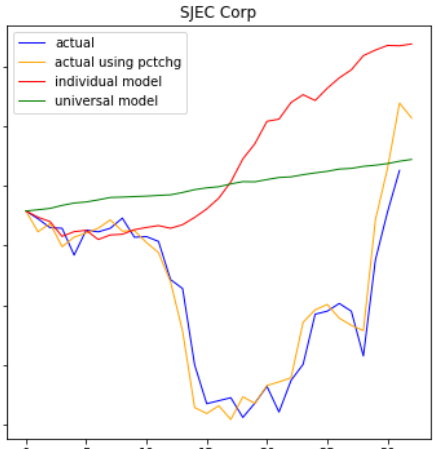
Result is shown below.

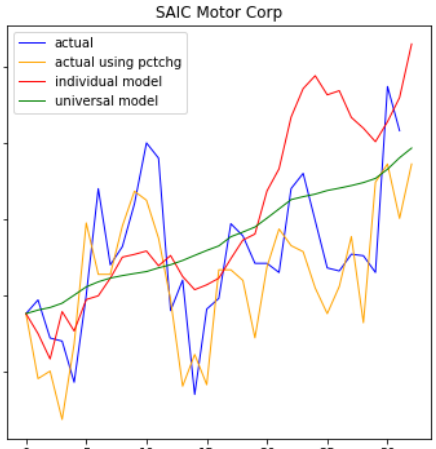
Note that while the blue line is the actual price movement, there is also an orange line, which is the price movement calculated using percentage changes in each trading day (in other words, changes from close price day one to open price day two are all considered noise and removed).

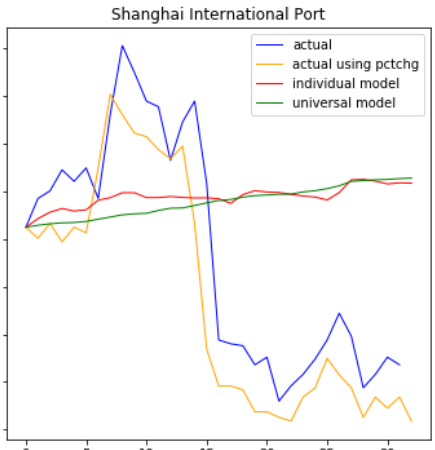
We can see that though the individual model may be slightly better, both models actually perform quite badly when we draw out the price movement curve.

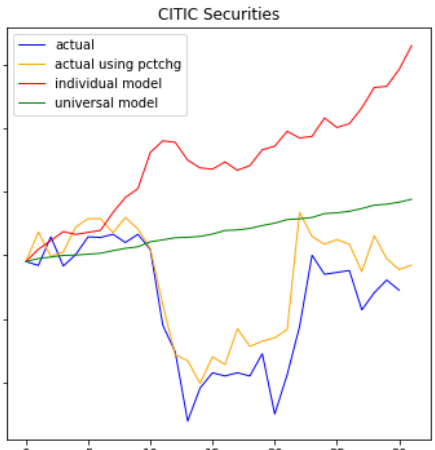
One explanation of this would be that our prediction solely focuses on percentage changes. And any error, any deviation from the actual changes would be accumulated and then "amplified" in a sort of exponential way.

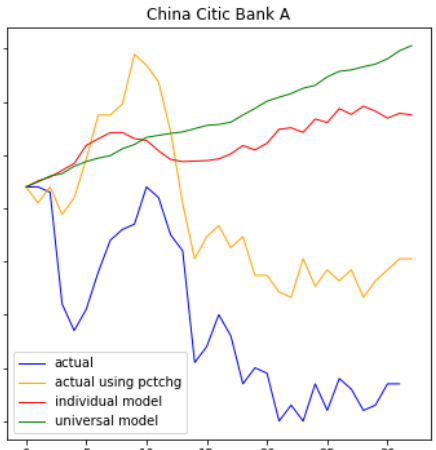

China Vanke A
actual
actual using pctchg
individual model
universal model

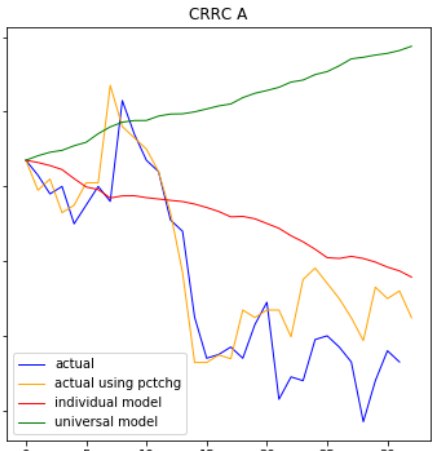

SJEC Corp
actual
actual using pctchg
individual model
universal model

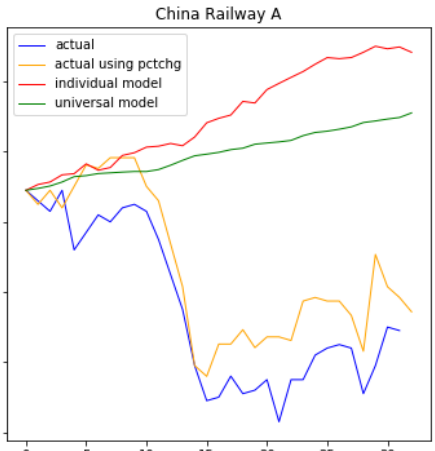

SAIC Motor Corp
actual
actual using pctchg
individual model
universal model

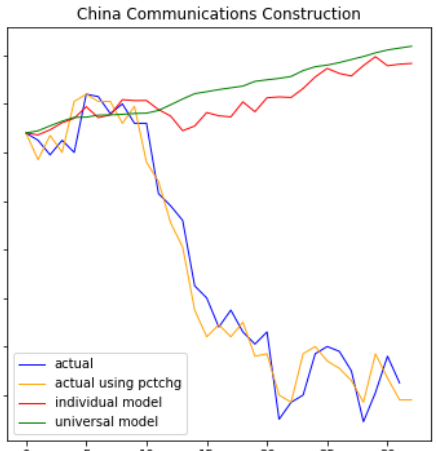

Shanghai International Port
actual
actual using pctchg
individual model
universal model


CITIC Securities
actual
actual using pctchg
individual model
universal model


China Citic Bank A
actual
actual using pctchg
individual model
universal model


CRRC A
actual
actual using pctchg
individual model
universal model


China Railway A
actual
actual using pctchg
individual model
universal model


China Communications Construction
actual
actual using pctchg
individual model
universal model

7 – Using Model to Actually Trade

Although our model is very naïve, simple and inaccurate, we implement the trading simulation and see its performance anyways.

A model may perform poorly in terms of giving inaccurate predictions of the absolute values it intend to predict (in this case, percentage changes), nevertheless, it may be a different story how it performs in terms of the differences, or relativity across different inputs (in this case, how predictions of different stocks compare to each other). In other words, though the absolute value prediction may be poor, it is still likely for the model to be powerful in term of distinguishing those stocks with potential and high probability to increase in price.

The trading simulation is pretty straight-forward and goes like the following:

Since the entire August is well within the testing dataset of individual models for all stocks, we can use the model to trade daily in August.

We start with 1 million RMB, for each day, we predict the percentage changes for all 50 stocks, choose the top **N** stocks, buy them at open prices and sell them close prices each day.

We always all-in with whatever we and further assume that our fund is split equally on the **N** stocks that we buy each buy.

In the end, we check how much money we have left in our fund.

The following is the result obtained by setting different N.

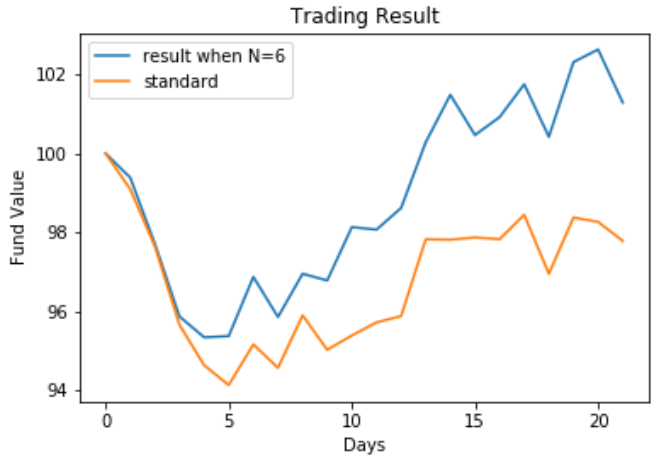
Note that N=50 means we basically buy all stocks with equal funding every day and is set as a standard for comparison.

We also add another strategy where we randomly select N to compare the noise coming from selection.

The result appears quite satisfactory at first sight. Specifically, when randomly selected, the ending position should fluctuate up and down around the standard (like the green line) but the blue line appears to be always above the standard line.

Secondly, in most cases, the blue line performs better than the green line, meaning that our percentage change prediction model indeed has some effect.

Also, when N=4-7, the ending result is 101, much higher that then standard, which is less than 98.
We can dig deeper and draw out the position movement when we set N=6.



We can see that the movement trend of N=6 and standard is very similar, but it is a bit superior to the standard curve in terms of performance almost across the entire timespan.

In conclusion, our prediction does seem to have a little bit edge over the dummy trading and "no prediction at all" strategy.

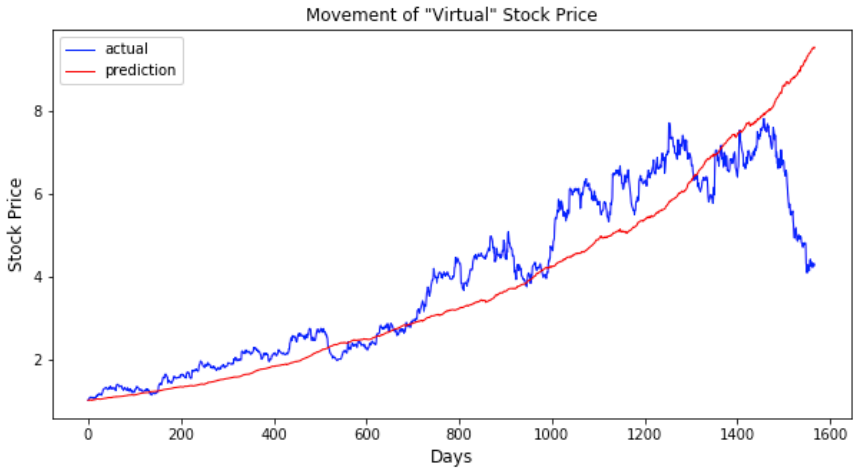
8 – Include History Market Data

Sticking with the simplest linear regression model, we try to include more features from the history market data for prediction this time.

This time, instead of only 6 sentiment scores, we use 15 numerical numbers for prediction. Namely, we use the open prices, percentage changes and trading volumes of previous three *trading days* and the six numbers already in use (news sentiment scores) to predict the percentage change today.

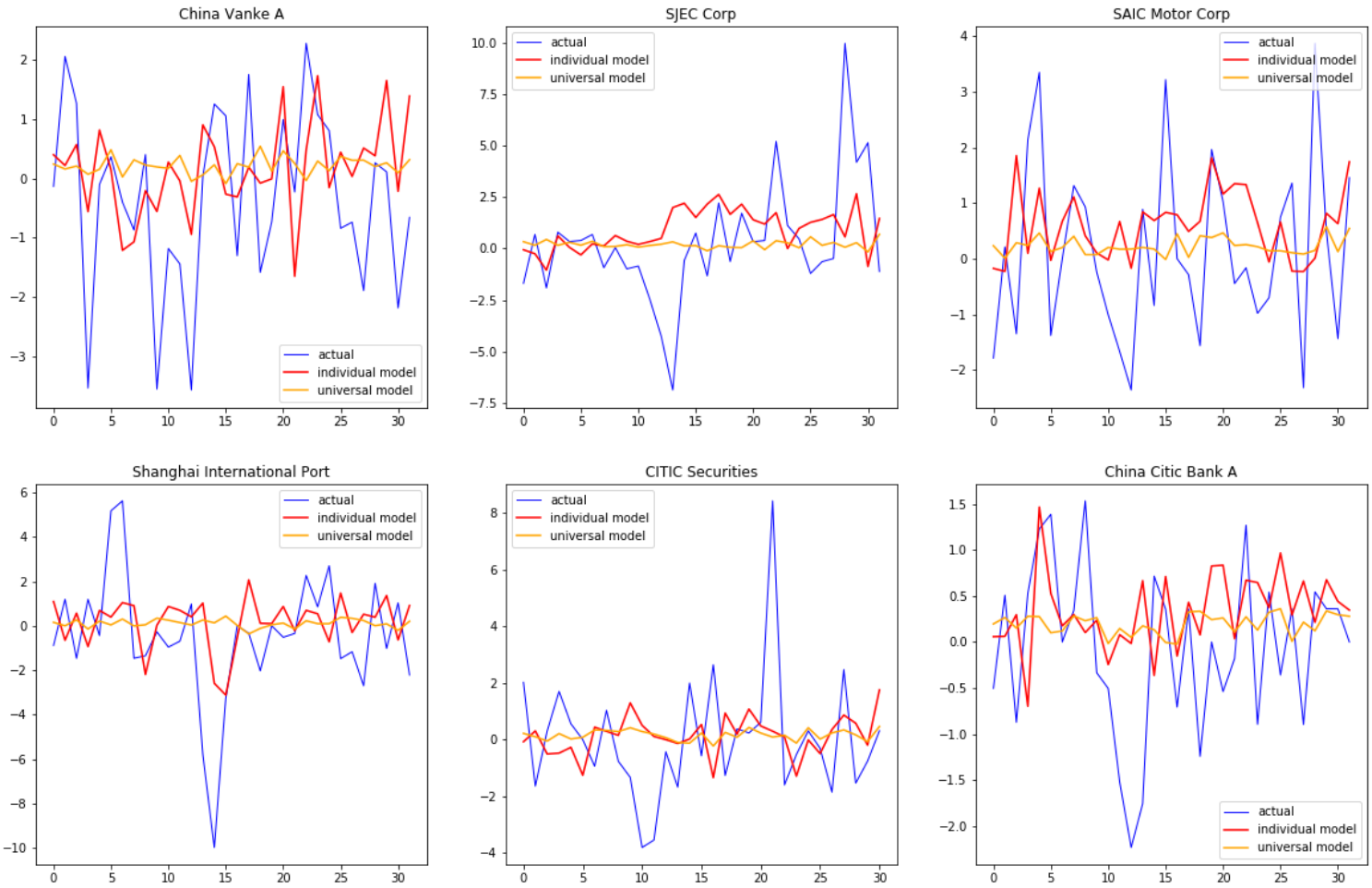
Basically, we use everything we prepared in the final dataframe (previously in section 5) to train and test a new linear regression model.

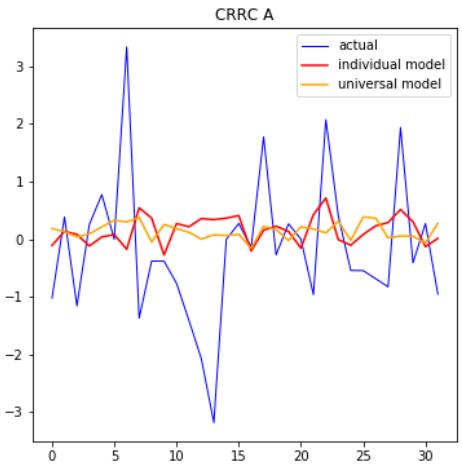
We display the results like section 6.

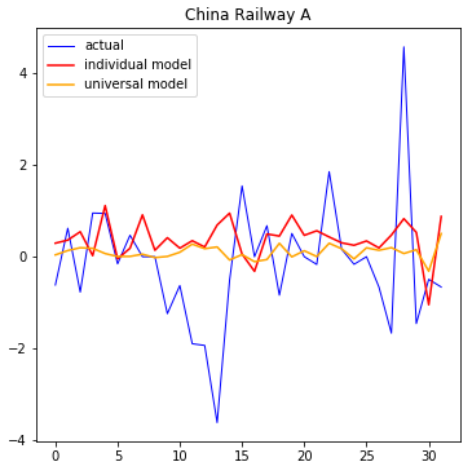


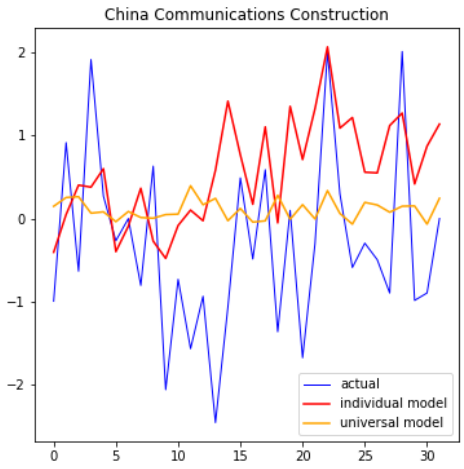
The result using universal model is largely the same as before. (6k-7k points is still way too much for 15 features).

Next, the prediction of percentage changes.



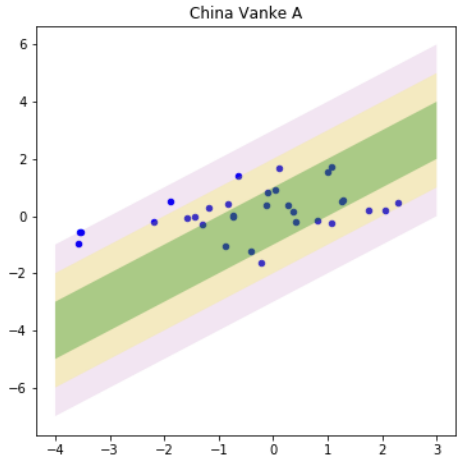


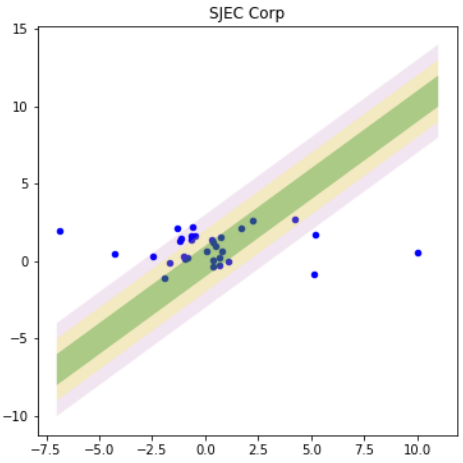


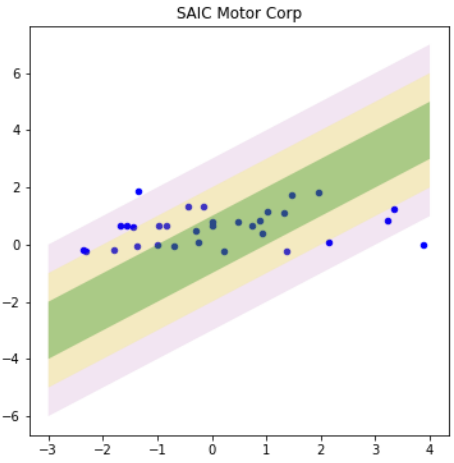


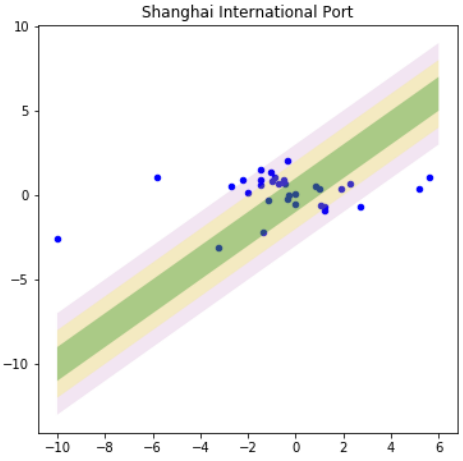
It seems with the addition of more features, the prediction of individual models become slightly more accurate and slightly more capable of capturing the ups and downs of the actual movements.

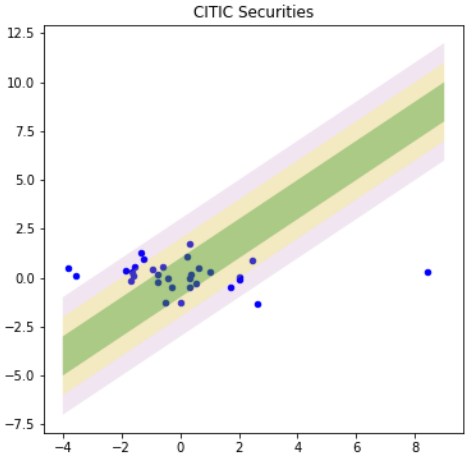
We then plot the scatter plot like previously.

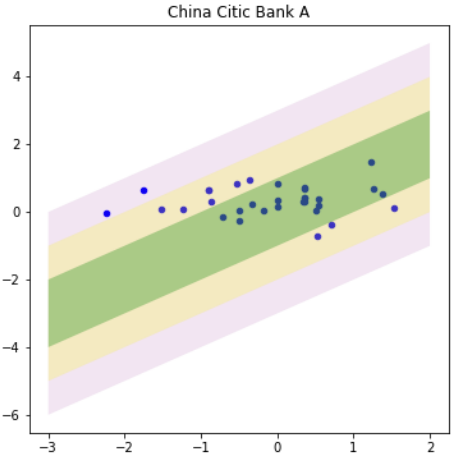


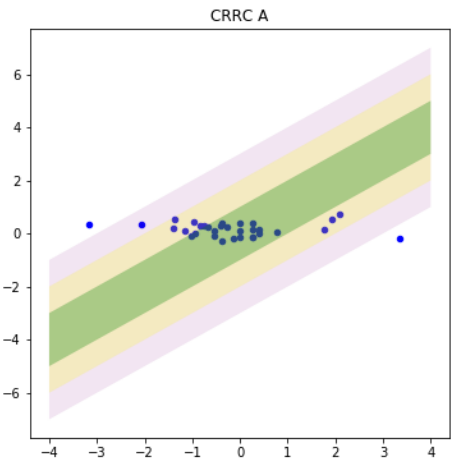


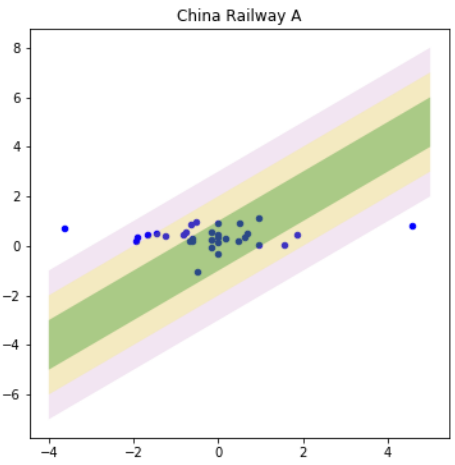


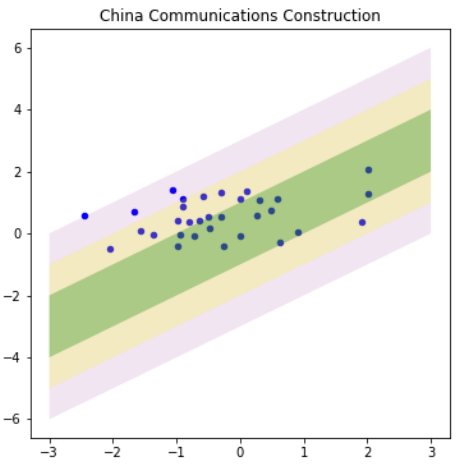






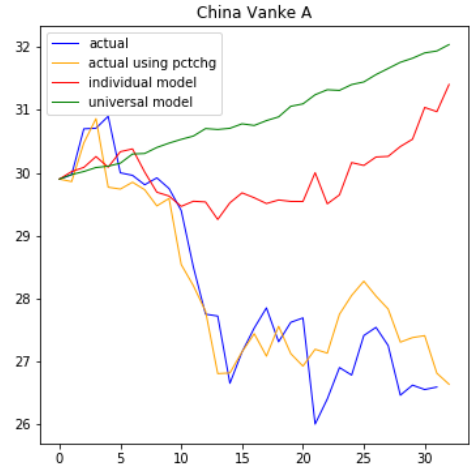


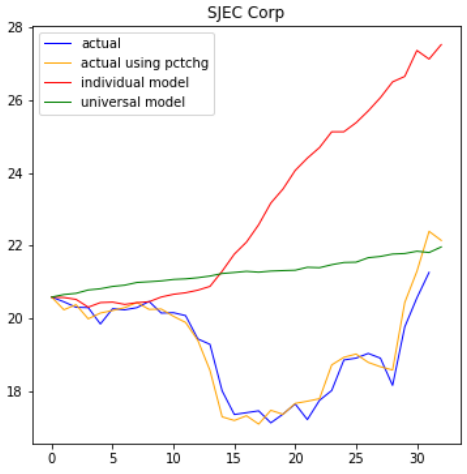


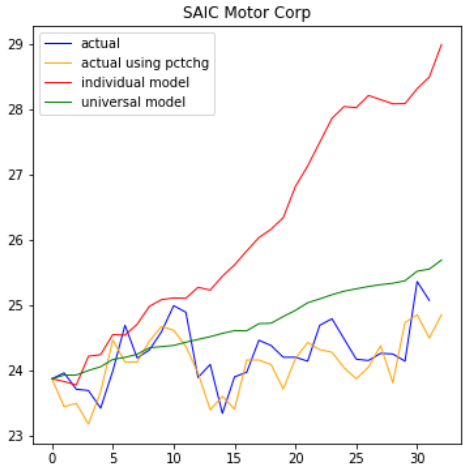


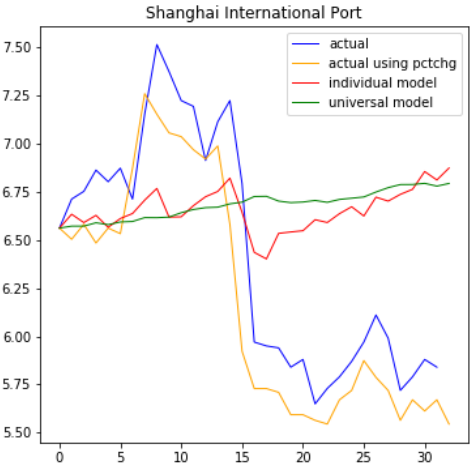
In general, there is not an obvious change that the dots converge more to the green region (which should be the case if the model is greatly improved by the addition of new features).

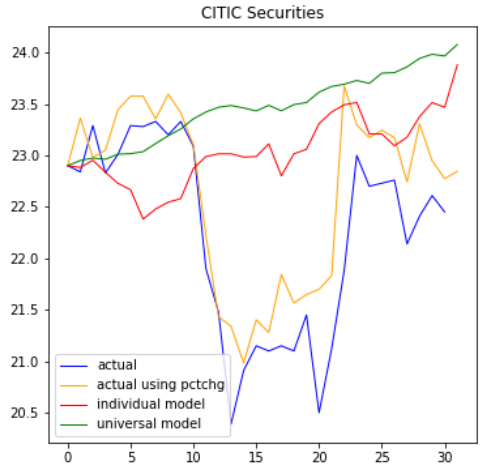
Next, we plot the prices calculated using the predicted percentage changes.

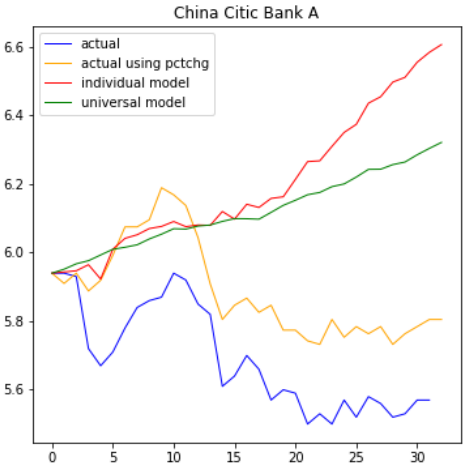


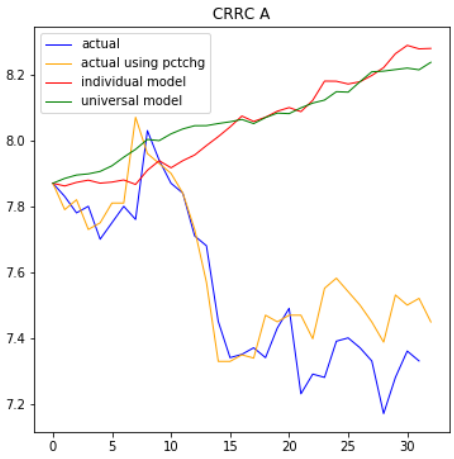


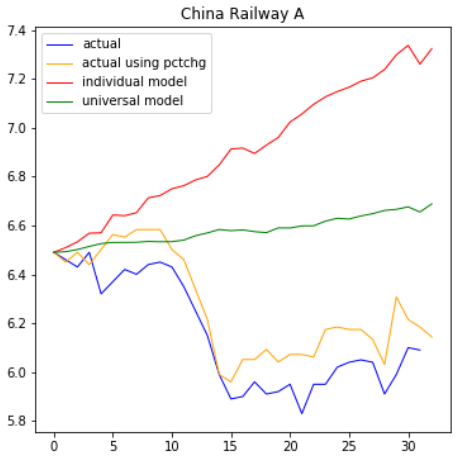


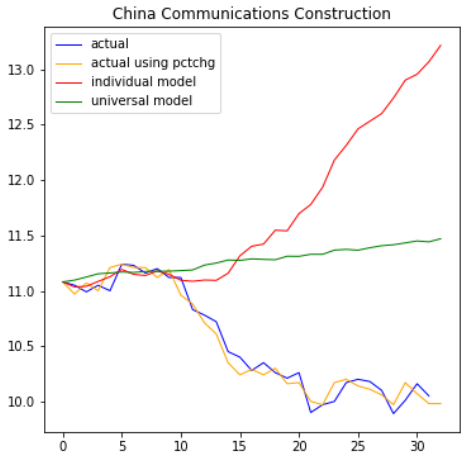










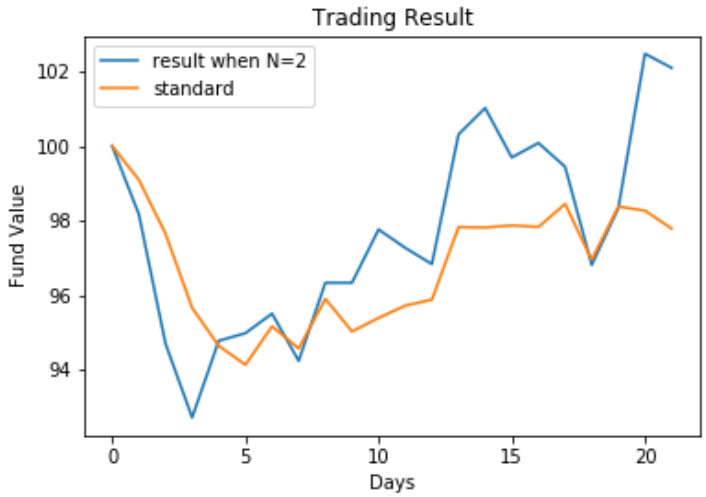


Again, there does not seem to be any discernable improvement. On the contrary, the individual models seem to perform even worse than previously if only look at these price movement graphs.

We put the models into trading and see our results like section 7,

Like before, when N<25, the blue line appears to be always superior than the green line. After the 25 point, though the green line sometimes performs better than the blue one, the blue one is way more stable than the green line.

In terms of the trading result, when N=2, the result outperforms the standard by the largest edge. We hence examine N=2 more carefully like before.



We see that the result is not always superior to the stock, it is actually worse than the standard during the first 3-4 days, which might worth a second look and a deeper investigation.

Overall, to sum up this section, we do see a slight (if any) improvement on the performance of models as more features (recent stock market data) are introduced. But such improvement is obviously not that significant.

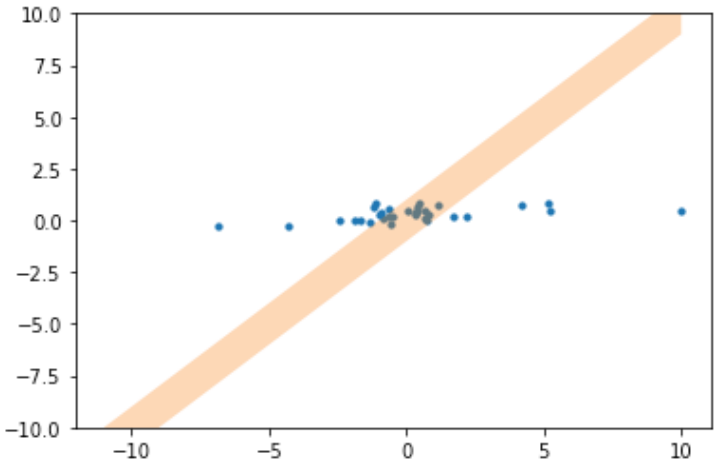
This suggests that the features that play the major role in our prediction model are still the news sentiment scores in the previous section.

9 – Training and Testing Neural Networks

Now that we have explored much using the simplest linear regression model, we try a basic neural network this time, which should be more sensible since the relationship between news sentiment scores and stock performance is probably non-linear.

Quite strangely, once I started to implement the neural network, the output converges to constant zero in an exponential speed. This is the case with both universal model and individual models.

Following is the scatter plot using the same logic as before.



The basic neural network structure (using RELU as activation function, one hidden layer with 5 neurons), after training, tends to always give a prediction of zero, regardless of the input given.

The reason of such peculiar behavior is yet to be investigated, but this is clearly not what we desired.

10 – Where More Studies can be Done

Basically, we have finished going through a shallow and experimental exploration of using text data to predict stock performance. Due to the time and efforts limitation, there are lots of places where we make casual, arbitrary assumptions and set some hyperparameters without any careful investigation.

This section summarizes those places where we skip a detailed look and where more studies can definitely be done to improve the performance of models or generate new ideas.

10.1 - Data Processing Stage

- When aggregating news on each single day, we simply discard the source information and take the average of all news published that days regardless of its source. We can distinguish news from different sources by one hot encoding since the reliability and the target readers of different websites can be quite different and this can have effect on the final prediction.

- We use the square kernel for the decay of the effect of news, there are various other kernels in the statistics field that may better model the real-life situation. I have implemented three simplest kernels but do not yet have time to test out all of them.

- In terms of the width (how long at maximum would the effect of a news last), we arbitrarily set this to be 10 days. Different settings can definitely be tested to see the changes.

10.2 – Input Preparation Stage

- For each piece of news, only its sentiment sore is used for prediction. Other information may have value as well, like the length of the text, appearance of names of other stocks, etc., may also be computed and used for prediction.

- When we say using “recent” news sentiment scores (and market data), we arbitrarily set “recent” to be three days. This can be adjusted to 5 days, one week, one fortnight or even one month to include more data and perhaps perform other sorts of data aggregation.

10.3 – Model Training Stage

- Given the nature of the data being “time-series”, recurrent neural network structures and Long Short Term Memory can be built and tested to exploit the underlying relationship of the news sentiment scores and stock market data between consecutive days.

- Given the nature of the stock market being similar to fuzzy logic, networks like SOFNN (Self Organizing Fuzzy Neural Networks) can be implemented and tested.