Predicting Stock Movement with Various Machine Learning Models

Yi Wang

2/6/2022

1 Summary

In this project, the historical APPLE stock daily `Close` price data is used to build various machine learning models to predict the stock price movement. The historical data includes all data since APPLE stock's launch in 1980-12-12 until 2022-02-04. The models built include

- baseline model using mean value
- smoothing model (single, double and triple exponential smoothing models)
- SARIMA models of various orders using `SARIMAX` python library
- SARIMA models of various orders using `pmdarima` python library
- FB Prophet model
- RNN model
- LSTM model.

The models are trained using the historical data since its launch in 1980-12-12 until 2022-01-21, and the last ten day's data (from 2022-01-24 to 2022-02-04) were hold out for out-of-sample testing. Then the models' performances are compared using the metric MAPE (mean absolute percentage error).

The best model we found is a deep learning RNN model which outperforms all other models in making out-of-sample forecasts in the metric of MAPE. The RNN model we found slightly outperforms the LSTM model we built.

A double exponential smoothing model and a SARIMA model provide slightly worse performance but is much inexpensive to train.

A FB Prophet model and a SARIMA model built by the python library `pmdarima` perform significantly worse, which is somewhat surprising.

Meanwhile we note that there is a significant amount of randomness in training a deep learning model (either RNN or LSTM), even with the same set of hyper-parameters. Of course, training a deep learning model is much more expensive in terms of computing time.

2 Description of the data set and exploratory data analysis

The underlying data set is fetched by using the `python` library `yfinance`. The format of the returned data is a `Pandas` data frame, with `Date` as the `index`, and columns of `Open`, `High`, `Low`, `Close`, `Volume`, and `Stock Splits` for each business day. We can view the head and tail of this data frame below.

```
display(apple_share_price_data.head()), display(apple_share_price_data.tail())
```

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
1980-12-12	0.100453	0.100890	0.100453	0.100453	469033600	0.0	0.0
1980-12-15	0.095649	0.095649	0.095213	0.095213	175884800	0.0	0.0
1980-12-16	0.088661	0.088661	0.088224	0.088224	105728000	0.0	0.0
1980-12-17	0.090408	0.090845	0.090408	0.090408	86441600	0.0	0.0
1980-12-18	0.093029	0.093466	0.093029	0.093029	73449600	0.0	0.0

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2022-01-31	170.160004	175.000000	169.509995	174.779999	115541600	0.0	0.0
2022-02-01	174.009995	174.839996	172.309998	174.610001	86213900	0.0	0.0
2022-02-02	174.527647	175.656214	173.109456	175.616257	84914300	0.00	0.0
2022-02-03	174.257984	176.015754	171.900986	172.679993	89418100	0.00	0.0
2022-02-04	171.679993	174.100006	170.679993	172.389999	82391400	0.22	0.0

The code
`apple_share_price_data.info()` returns the summary information of the data frame:

```
DatetimeIndex: 10376 entries, 1980-12-12 to 2022-02-04
```

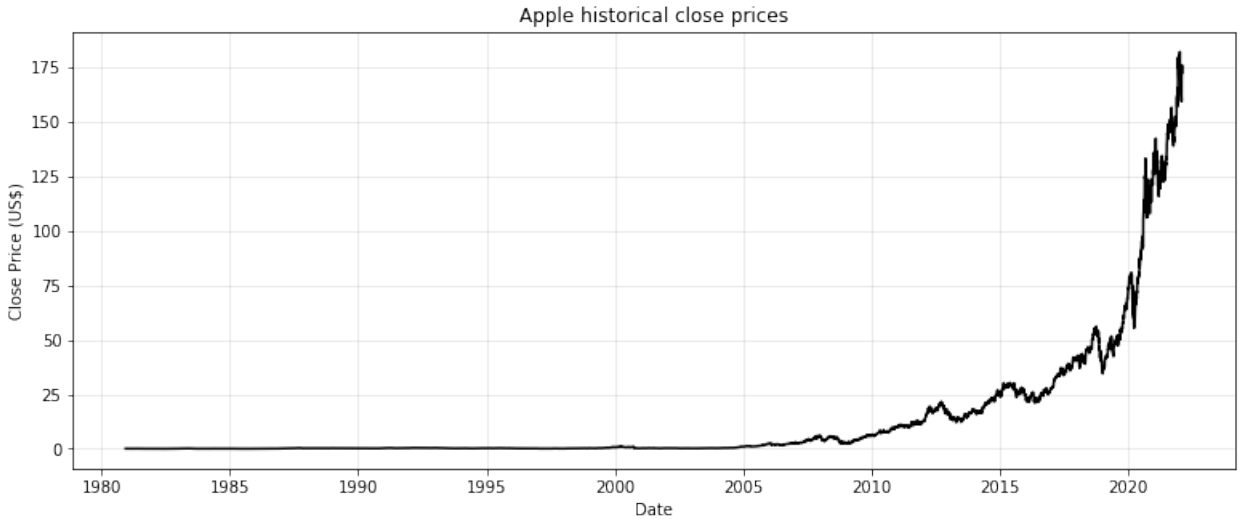
#	Column	Non-Null Count	Dtype
0	Open	10376 non-null	float64
1	High	10376 non-null	float64
2	Low	10376 non-null	float64
3	Close	10376 non-null	float64
4	Volume	10376 non-null	int64
5	Dividends	10376 non-null	float64
6	Stock Splits	10376 non-null	float64

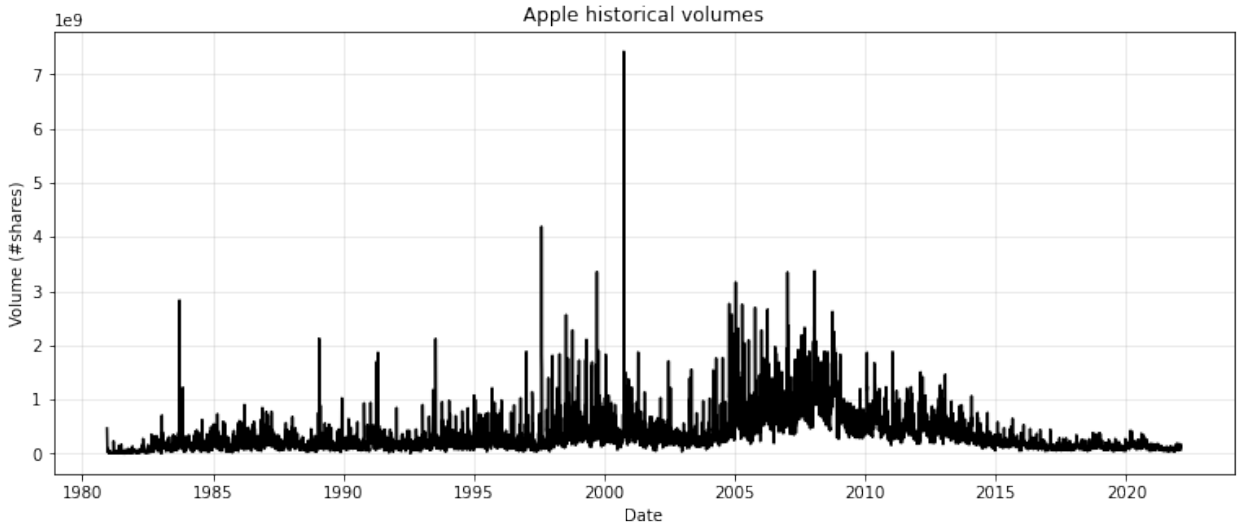
The following code returns the summary statistics of the columns:

```
apple_share_price_data.describe()
```

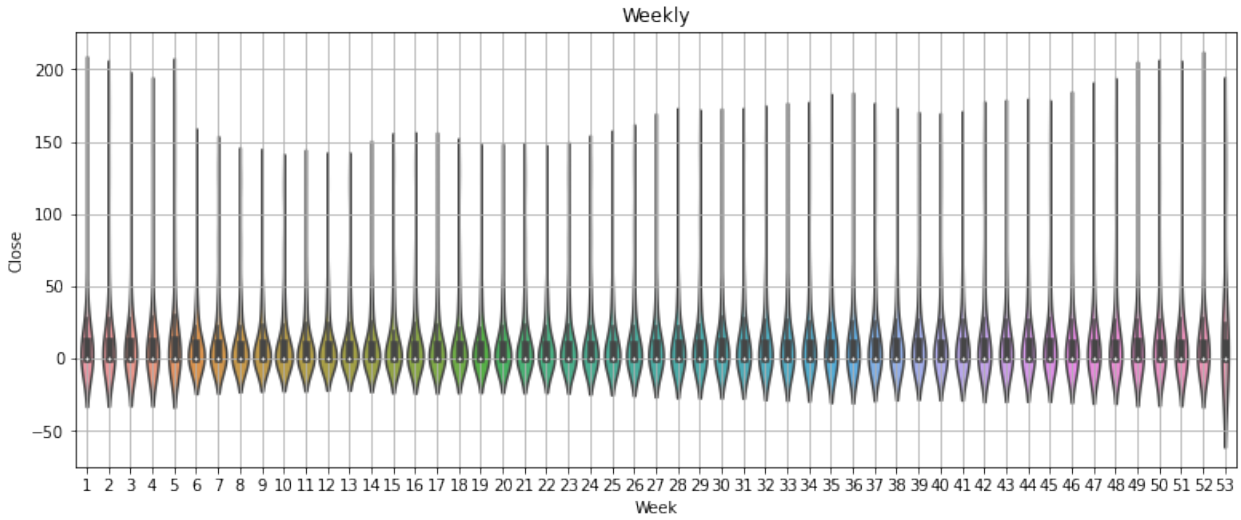
	Open	High	Low	Close	Volume	Dividends	Stock Splits
count	10376	10376	10376	10376	10376	10376	10376
mean	12.847074	12.985554	12.709491	12.852744	3.329877e+08	0.000571	0.001639
std	28.637493	28.959756	28.323289	28.654589	3.396185e+08	0.009665	0.086142
min	0.038871	0.038871	0.038434	0.038434	0.000000e+00	0.000000	0.000000
25%	0.234332	0.239407	0.228642	0.234135	1.254941e+08	0.000000	0.000000
50%	0.383714	0.390331	0.377110	0.383977	2.211366e+08	0.000000	0.000000
75%	11.957641	12.064507	11.850471	11.941411	4.143447e+08	0.000000	0.000000
max	182.630005	182.940002	179.119995	182.009995	7.421641e+09	0.220000	7.000000

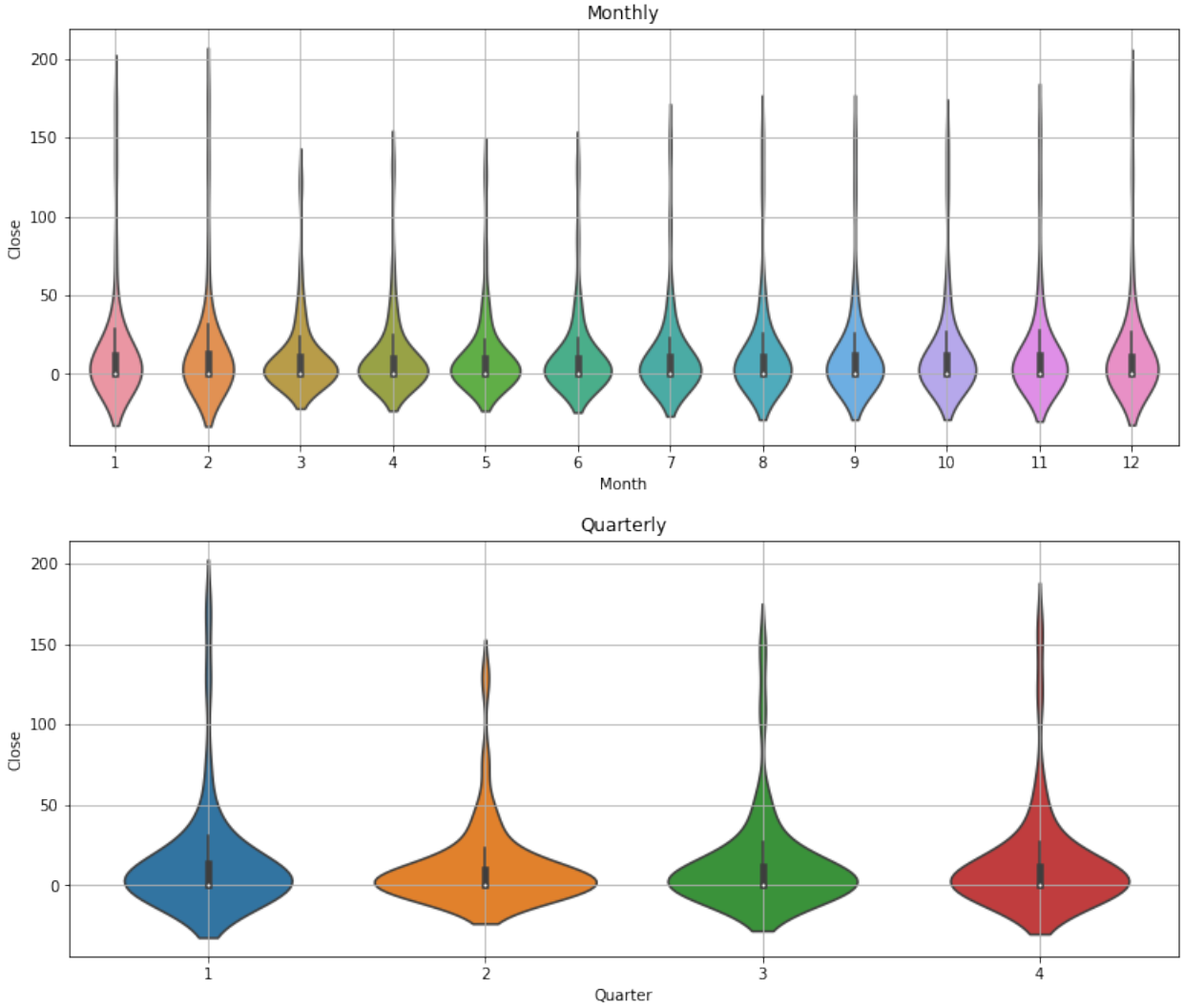
We can visualize the historical `Close` price and `Volume` in the next two plots.



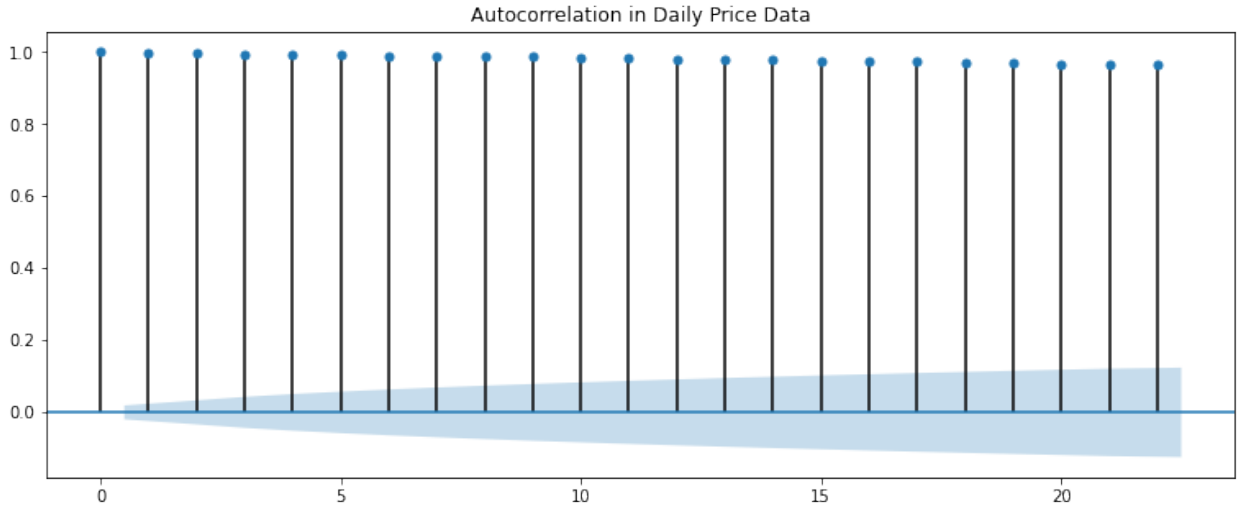


To example if there is any seasonality in the daily `Close` price data, we plot the **violin** plots of the data by week, month and quarter.



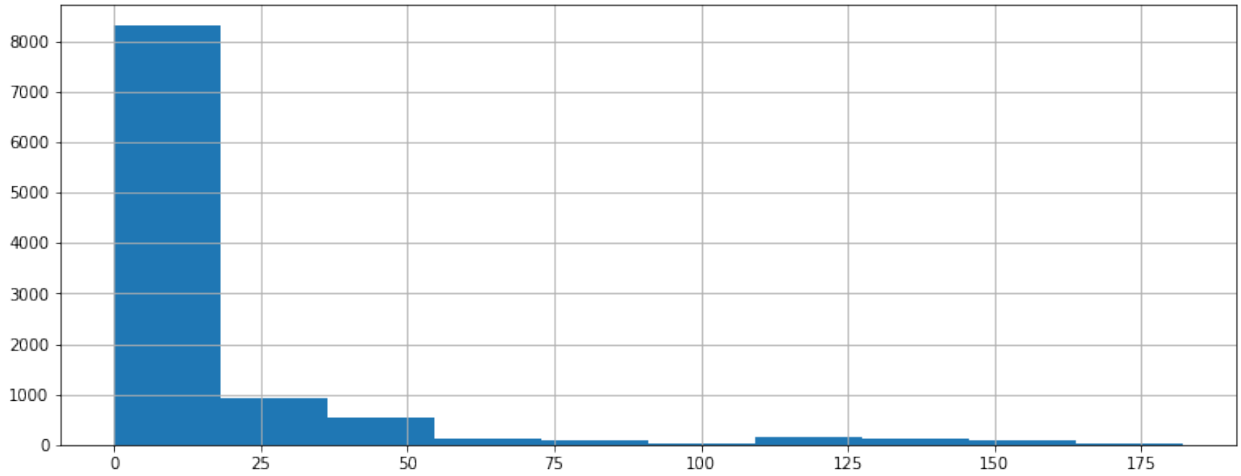


The violin plots reveals that there is no significant difference in the data by week, month or by quarter. Next we examine the autocorrelation in the data by looking at the ACF and PACF.

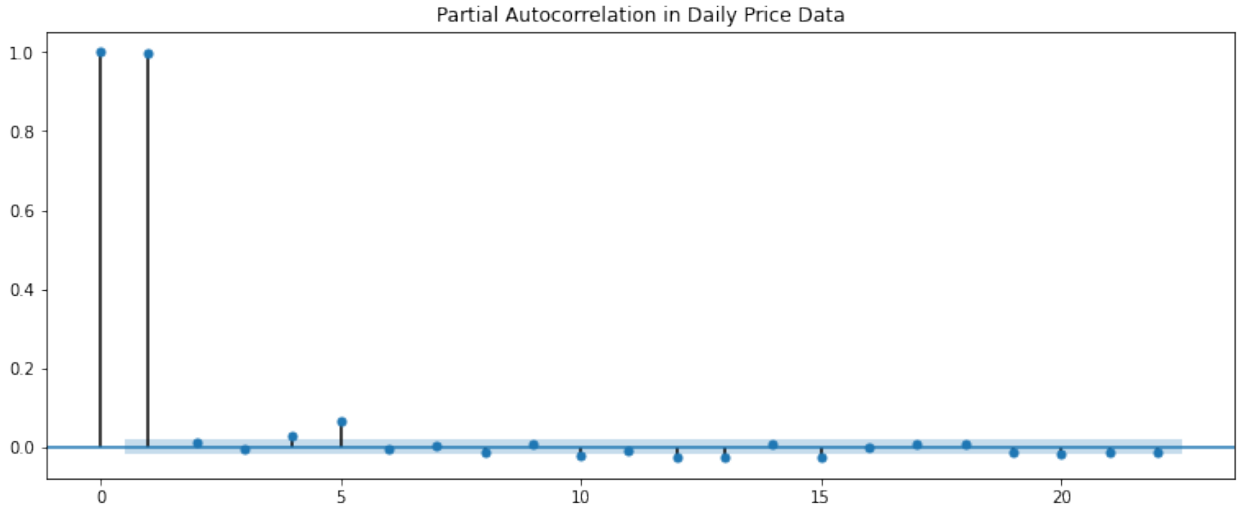


The ACF does not decay at all, indicates *the data is non-stationary.* This is confirmed by the histogram of

the data as shown below.



A look at the PACF function indicates that it has a large value at lag 1 and almost zero elsewhere. This indicates that at least there is a **strong order 1 (p=1) autocorrelation** for the daily close price data.



In summary, the daily `Close` price data has the following attributes:

- non-stationary
- no significant seasonality
- a strong order 1 ($p = 1$) autocorrelation.

Since the returned data frame is already in the format that the analysis needs, therefore, there is no need to clean the data. For the sake of this project, as we are going to *only focus on the time series of the daily* `Close` *price data*, we will not engineer any additional features.

3 Objectives

The objective of this project is to use the historical APPLE stock daily `Close` price data to build various machine learning models to forecast APPLE's stock price movement. The models built include

- baseline model using mean value

- smoothing models (single, double and triple exponential smoothing models)
- SARIMA models of various orders using `SARIMAX` python library
- SARIMA models of various orders using `pmdarima` python library
- FB Prophet model
- RNN model
- LSTM model.

The models are trained using the historical data since its launch in 1980-12-12 until 2022-01-21, and the last ten day's data (from 2022-01-24 to 2022-02-04) were hold out for testing. Then the models' performances are compared using MAPE (mean absolute percentage error). A **potential business benefit** is to find a best model that can bring meaningful forecasting of APPLE's stock price movement for investors.

4 Modelling

In this section, we will explore various machine models. Each model will be trained by using the APPLE historical daily `Close` price data from its launch in 1980-12-12 until 2022-01-21, and the last ten day's data (from 2022-01-24 to 2022-02-04) were hold out for out-of-sample testing and assessing each model with the metric MAPE.

4.1 Smoothing models

We tapped various smoothing models including

- Simple averaging smoothing
- Single exponential smoothing
- Double exponential smoothing
- Triple exponential smoothing

The hyper-parameters of the triple exponential smoothing were chosen by a grid search from the following grid:

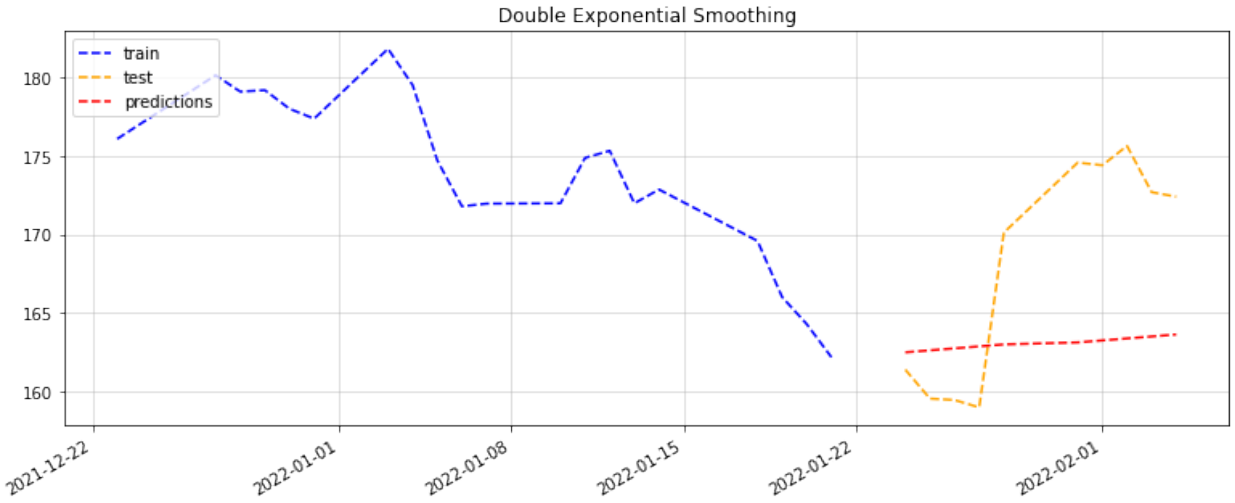
- trends = [None, additive, multipicative]
- seasonal = [None, additive, multiplicative]
- seasonal period = [2:10:1]

The best hyper-parameters we found for a trip exponential smoothing model is: trend=`None`, seasonal=`multiplicative`, seasonal_periods=`2`.

The table below lists the out-of-sample testing error in terms of MAPE:

Methods	MAPE
Simple Avg.	0.9243
single exponential	0.0450
double exponential	0.0423
triple exponential	0.0449

It appears among the smoothing methods, **double exponential smoothing** outperforms other smoothing methods. We plot the train data (only showing partially), test data and the forecast data by the double exponential smoothing method below.



4.2 SARIMA models of various orders using python library SARIMAX

We next explore SARIMA models with various orders using the last ten day's stock `Close` price for out-of-sample testing to select a best SARIMA model. The hyper-parameters of the SARIMA model was chosen by a grid search from the grid:

ARMA order =[(1,0,0), (1,1,0),(1,0,1),(1,1,1)]

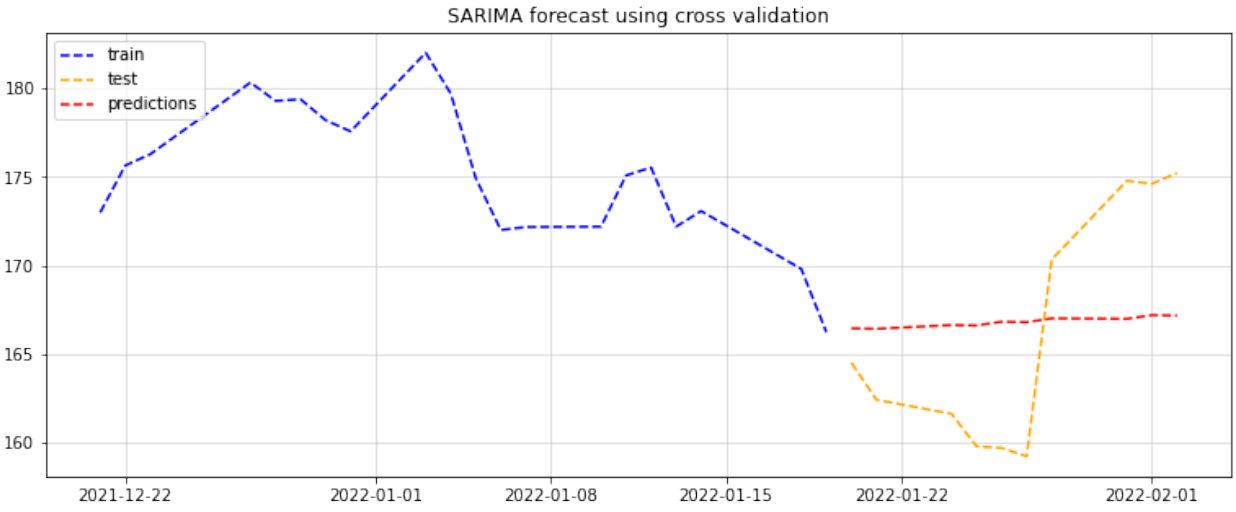
Seasonal order = [(0,0,0,0),(1,0,1,2), (1,1,1,2), (1,1,0,2)]

trend =[None, c]

The best set of hyper-parameters we obtained is a SARIMA model with no trend, orders (1,1,1)(1,1,1,2) with a

- MAPE = 0.0416.

An illustration of the predicted data versus out-of-sample test data is shown below.



4.3 SARIMA models Using python library pmdarima

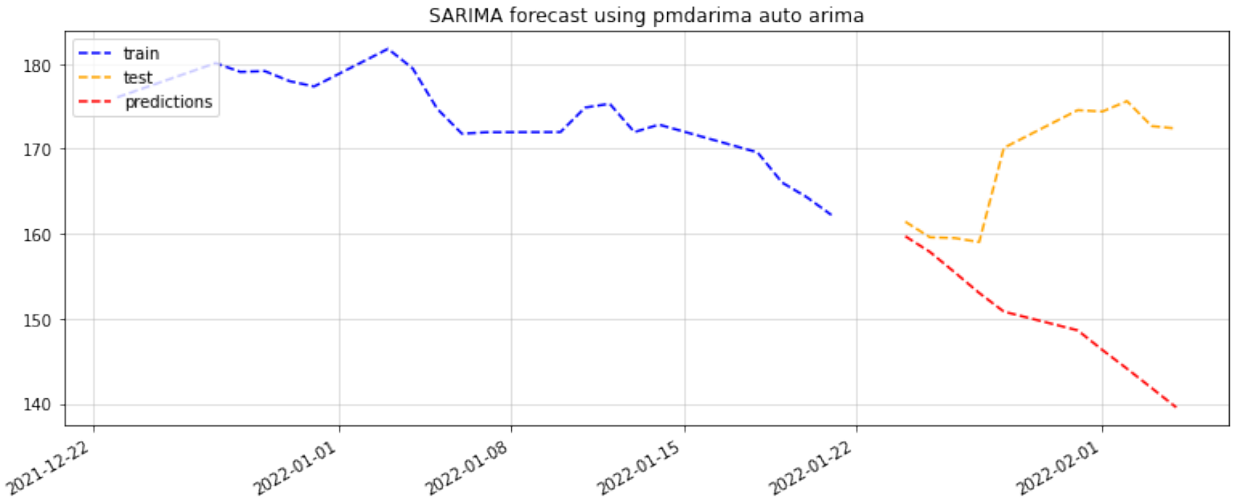
We used the python auto SARIMA library `pmdarima` to conduct grid search with the following parameters:

- $max_p = 3$, $max_d = 2$, $max_q = 3$, $m(period) = 2$, $max_P = 2$, $max_D = 1$, $max_Q = 2$.

The best model we obtained was SARIMA (1,2,0)(2,0,0)[2] with

- MAPE= 1.085

The prediction again is illustrated below.



Unfortunately, the auto.arima of the library pmdarima actually returned a worse model.

4.4 Facebook Prohphet model

We next explore the Facebook Prophet tool to build a model. We conducted grid search. The grid searched is:

changepoint_prior_scales = [0.001, 0.01, 0.1, 0.5,1]

seasonality_prior_scales = [0.01, 0.1, 1.0, 10.0]

seasonality_modes=['additive','multiplicative']

The best model we obtained from the grid search is a model with the hyper-parameters:

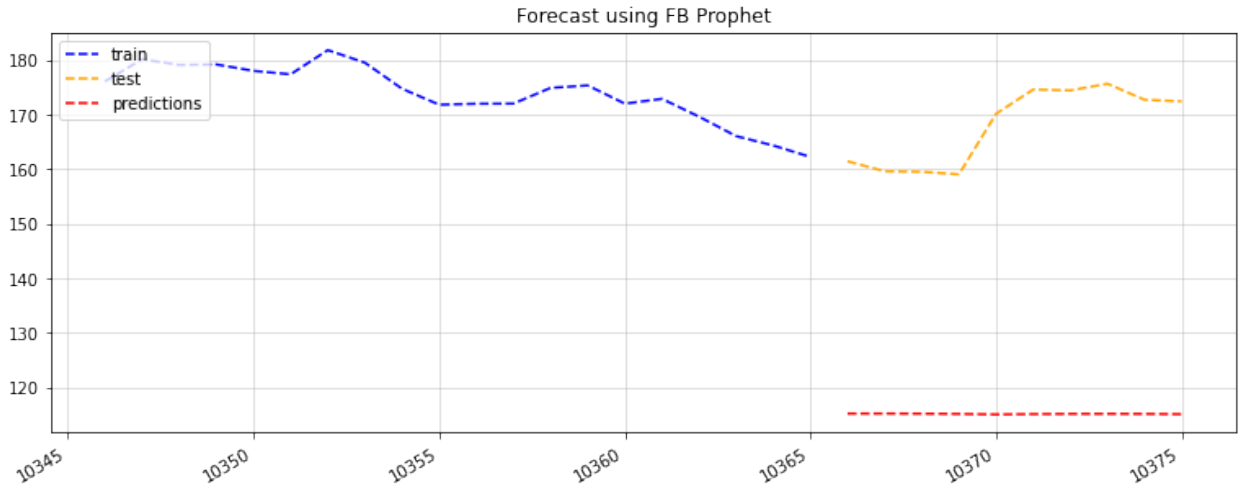
changepoint_prior_scale = 1.0

seasonality_prior_scale = 1.0

seasonality_mode='additive'

The MAPE of the best Prophet model is

- MAPE = 0.3144



Unfortunately, the model did not demonstrate a good performance.

4.5 Simple RNN model

We now explore deep learning models. The grid search conducted is:

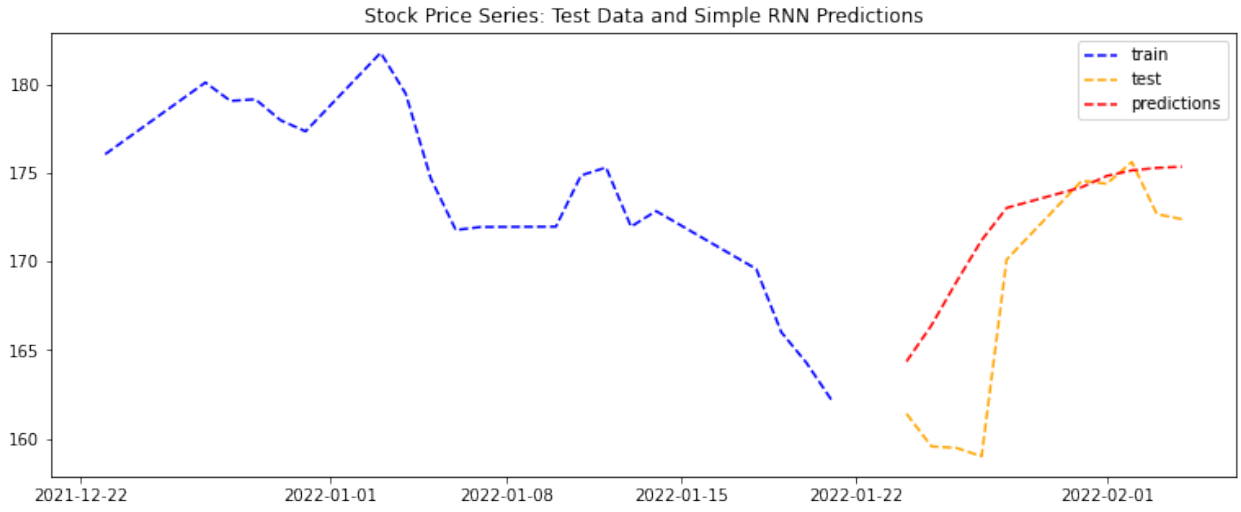
input_days=[10, 20, 30]

cell units =[10, 20, 30, 40]

epochs=[2000, 3000]

The best model we obtained is given by input_days=20, cell_units=30, epochs =2000, and the MAPE is - MAPE =0.0245

Again the prediction is illustrated below.



4.6 LSTM model

Lastly, we explore LSTM deep learning models. The grid searched is:

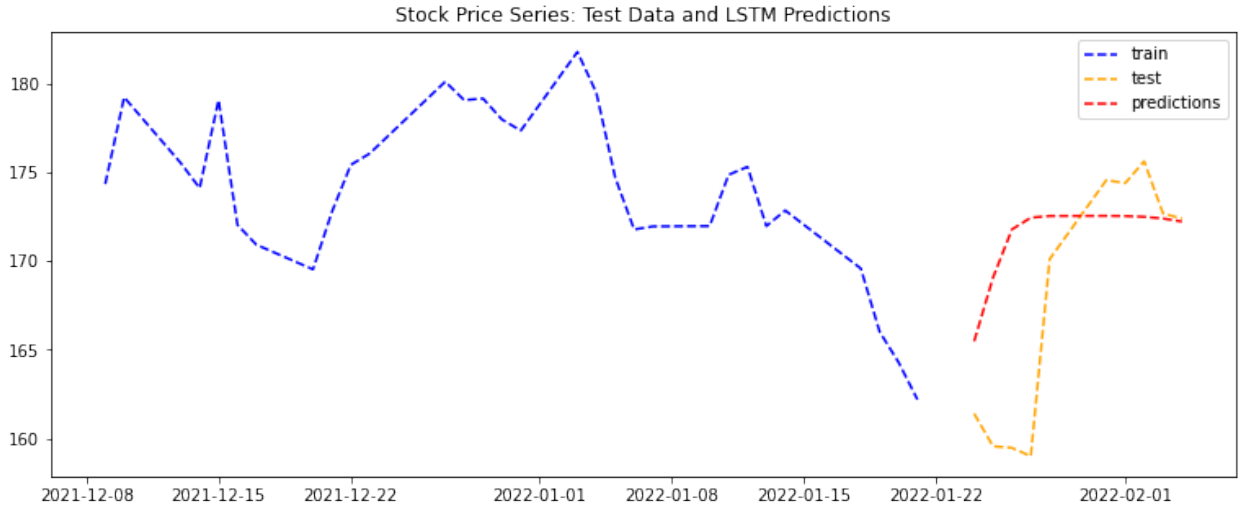
input_days=[10, 20, 30]

cell units =[10, 20, 30, 40, 70, 130]

epochs=[2000, 3000]

The best model we obtained is given by input_days=30, cell_units=130, epochs =2000, and the MAPE is - MAPE =0.0293.

The prediction is illustrated below.



5 Key findings

We list below the models we built and their out-of-sample(using the hold-out last ten day's prices) testing MAPE values:

Methods	MAPE
Simple Avg.	0.9243
Single exponential	0.0450
Double exponential	**0.0423**
Triple exponential	0.0449
SARIMA	**0.0416**
pmdarima	1.085
FB Prophet	0.3144
RNN deep learning	**0.0245**
LSTM deep learning	**0.0293**

- Without much surprise, the deep learning models LSTM and RNN outperform all other methods. For some reason, the trained simple RNN model outperforms the LSTM model slightly. Due to a large number of parameters to be determined for a deep learning model, the training is very expensive in terms of computing time.
- Surprisingly, the FB Prophet and the pmdarima auto arima model perform significantly worse.
- There is a significant amount of randomness in training a deep learning model (either RNN or LSTM), even with the same set of hyper-parameters.
- In training a deep learning model for time series, one does not need to worry about the SARIMA order which is quite handy, however, at the expense of explainability.

6 Discussions and future work

In this project, our models were tested using 10 out-of-sample time series values. In reality, making a forecast on a shorter horizon (for example, shorter than 10 days) and then re-training a model with newly available data is quite doable and also seems very reasonable. Therefore, future work may involve evaluating a model by forecasting a shorter horizon (such as a day or two) and then retrain a model with newly available data.

Another future work may involve using more features, such as volume data instead of just the 'Close' price in training a deep learning model. Using more features might be more appropriate to train a deep learning model.

Of course, another future work is to involve explore more hyper-parameters which we could not complete in this project due to the limited computing resources.

7 Appendix-Computing environment

The experiment was conducted using google colab. A copy of the Jupyter notebook is available at the GitHub address: Time_series_stock_data_analysis_ver2.ipynb.