Capstone Project

Data Scientist Nanodegree Program

Zax Chan

20th Aug., 2020

Project Definition

Project Overview

Stock price prediction is a quite popular application for data science, which is trying to determine the future value of a company's stock on trading. Successful and consecutive predictions of a stock's future price could yield massive profit. Stock market allows us to buy and sell units of stocks (ownership) of a company. If the company's profits go up, then we own some of the profits and if they go down, then we lose profits. Prediction provides price trend and value information regarding the current status of the stock price movement.

This project will help the Intra-day Traders to get insights of the next day's stock market and will help them to prepare their buying and selling strategies. In this project, Stock market forecasting has gained more attention recently, maybe because of the fact that if the direction of the market is successfully predicted the investors may be better guided. The profitability of investing and trading in the stock market to a large extent depends on the predictability. If any system developed which can consistently predict the trends of the dynamic stock market, would make the owner of the system wealthy.

Here in this project, cmb China with the full name China Merchants Bank is a typical choice with the characteristic mechant bank features for data source, I fetch the historical data which is retrieved from Tushare[1]. Cmb China provides full mechant financial and bank services in China. It operates through Banking, Financial Services and Insurance, Consumer Business and others segments.

[1] http://tushare.org/trading.html

Problem Statement

The aim of this project is to predict the next day's closing prices of cmb China stock using the deep learning model Long Short Term Memory[2], and also to optimize the hyperparameters of the network and do feature selection on the dataset. The final model is useful to get the next day's closing price.

Evaluation Metrics

R2 Score: explained as variance score, and mean squared error, as well as explained variance score and median absolute error are all going to be used to validate the result. By using all the metrics, the risk of being biased will be reduced. Expected result is the best regression model have the highest score in all the scoring metrics. Best possible score is 1.0 and it can be negative because the model can be arbitrarily worse. A constant model that always predicts the expected value of y, disregarding the input features, would get a R2 score of 0.0[3].

$$R^2 = 1 - \frac{Sum\_of\_squared\_residuals/n}{variance_{y\_actual}}$$

Mean Squared Error (MSE) : It is the mean squared error regression loss which is defined as a non-negative floating point value on which the best value is 0.0[4].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(predicted\_value_i - actual\_value_i)^2}{n}}$$

[2] C. Olah, "Understanding lstm networks, https://colah.github.io/posts/2015-08-Understanding-LSTMs/
[3] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html
[4] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html?highlight=mean%20squared%20error#sklearn.metrics.mean_squared_error

Analysis

Data Exploration

The data used in this project is of the cmb China taken from Jan 4, 2010 to Aug 21,2020, a period of about 10 years. This is a series of data points indexed in time order. Some terminologies related to stock price are given below:

date : the date of the day on which the trading occurs.
open : price of the first trade for any listed stock is its daily opening price.
close : the last price at which a stock trades during a regular trading session.
high : the highest price at which a stock is traded during the course of the day.
low : the lowest price at which a stock is traded during the course of the day.
vol : the number of shares or contracts traded in an entire market during a given period of time.
amount: the amount money traded in an entire market during the course of the day.
The dataset looks like **Fig**.1 :

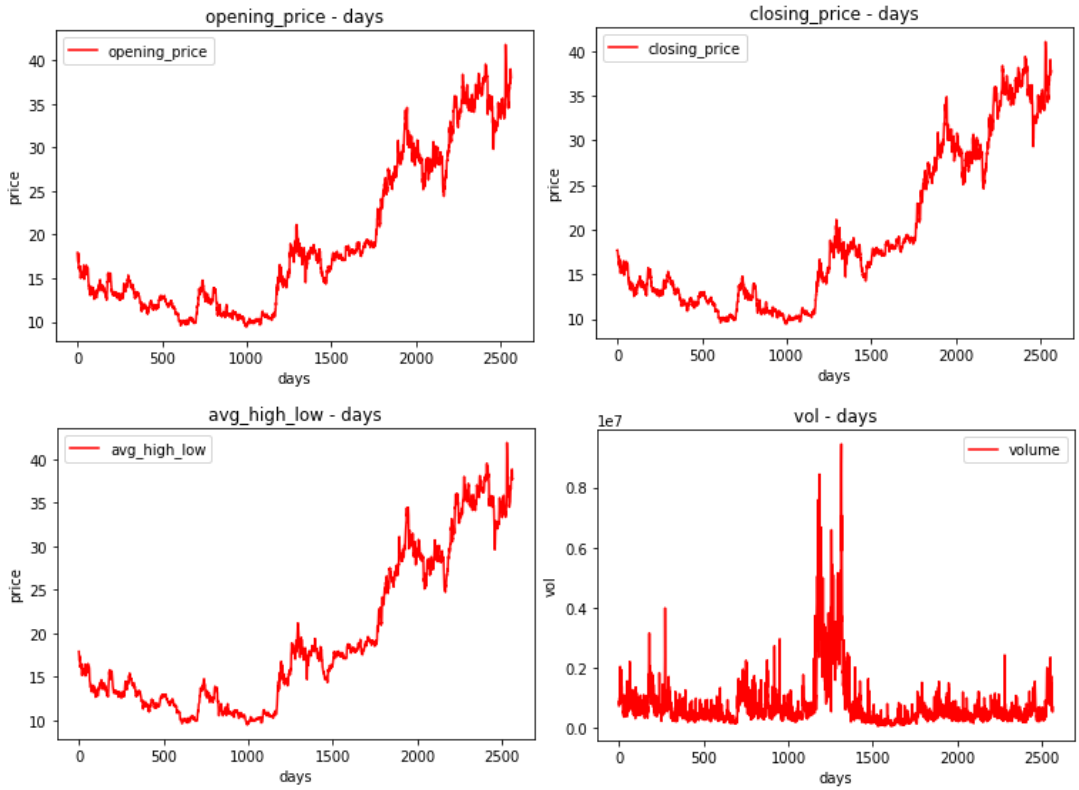
Fig.1: cmb-China Dataset from Jan 4,2010 to Aug 21,2020

date	open	high	low	close	vol	amount
20100104	17.93	18.11	17.66	17.71	746135.6	1332370
20100105	17.71	18	17.34	17.73	914013.7	1615350
20100106	17.69	17.69	17.31	17.36	774277.1	1348703
20100107	17.33	17.43	16.78	16.91	1019432	1742282
20100108	16.88	16.98	16.68	16.91	803598.6	1351767
20100111	17.88	17.91	16.8	16.94	1465693	2530625
20100112	16.88	17.17	16.45	17.07	1121898	1879278
20100113	16.52	16.7	15.97	16.15	2052233	3344313
20100114	16.2	16.35	15.95	16.33	1247176	2011436
20100115	16.27	16.46	16.03	16.31	873022	1417366

Data Visualization

Below are the plots of all the features which are taken into account to train the model. The X axis represents the number of days from Jan 4, 2010 onwards and Y axis represents price or volume. From the plots, the fluctuating growth about the prices and volume of trading about the stock of the company can be seen as **Fig**.2.

Fig.2: Four plots showing the opening price, closing price, average high and low price, and volume along the days.



Methodology

Data Preprocessing

The preprocessing is done in the following step:

- First, the null values are dropped from the dataset
- Second, the normalization of the column values is performed using MinMaxScaler
- Third, creating 3D input tensor

We performed normalization of the data with the help of the following:

$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

After Normalizing, I split the data into 4:1 ratio. So, after splitting the shape of the set are as follows:

- Training set : (2052, 4)

- Testing set : (513, 4)

After Splitting, I reshaped the data to form the 3D input tensor where the shape of the set are as follows:

- Training set : (2052, 60, 4)
- Testing set : (513, 60, 4)

Implementation

The implementation process are as follows:

- Loading the dataset and splitting into training and testing set by 4:1 ratio
- Creating the 3D input tensor
- Training the model
- Validating the model
- Plotting the results

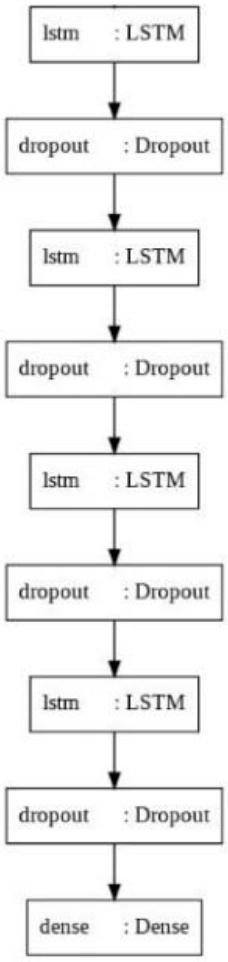
In this project, I have incorporated a stacked LSTM network to work on the problem of stock price prediction which is a type of sequence-to-sequence time series problem. A stacked LSTM architecture can be defined as an LSTM model comprising of multiple LSTM layers. Stacking LSTM layers makes the model deeper which helps the model to store more complex information and trends. The additional hidden layers are understood to recombine the learned representation from prior layers and create new representations at high levels of abstraction for next layer.

Here there are 4 LSTM layers, 4 dropout layers and 1 dense layer at the end as **Fig**.3. In this case, the input LSTM network changes according to the number of features taken into account. In order to feed the input features to the neural network, first it was created with a 3-dimensional numpy tensor of input features.

There are 60 input nodes in the first LSTM layer which should be equal to the number of timesteps.

Then, there are 3 LSTM hidden layers where first two hidden layers have 100 hidden nodes and last LSTM hidden layer has 40 hidden nodes. At the end, there is a Dense layer having only 1 node which outputs the result. After every LSTM layer, I have introduced a Dropout layer to avoid overfitting of the model with a dropout rate of 20% at each layer. In every LSTM layer, I have used a Linear activation function and at the Dense layer, used tanh activation function.

Fig.3: The illustration below shows the model architecture, which includes networks, and shapes of every layer, and parameter quantity for corresponding layer.



Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 60, 60)	14880
dropout_1 (Dropout)	(None, 60, 60)	0
lstm_2 (LSTM)	(None, 60, 100)	64400
dropout_2 (Dropout)	(None, 60, 100)	0
lstm_3 (LSTM)	(None, 60, 100)	80400
dropout_3 (Dropout)	(None, 60, 100)	0
lstm_4 (LSTM)	(None, 40)	22560
dropout_4 (Dropout)	(None, 40)	0
dense_1 (Dense)	(None, 1)	41

Total params: 182,281
Trainable params: 182,281
Non-trainable params: 0

Sometimes after completing the training, I may not get the perfect model because of local minima of cost function. So, I try to incorporate some optimizer in our training topology like Adam or Rmsprop to avoid this type of problems. Here, I have used Adam optimizer which will help to attain the global minima of cost function. To train the model, I have sent the input samples with a batch size of 32. In very first run, the first input LSTM layer will fetch very first timestep from the input tensor containing stock trend of starting 60 days of the dataset. In case of single attribute model, the timestep will be a (60, 1) matrix whereas in case of multiple attribute model eg. 4 feature attributes, thetimestep will be a (60, 4) matrix. After this, a feed forward pass is done and I get an output which I use to calculate the cost function. Here, I have used Mean Squared Error as the cost function to calculate the error.

Refinement

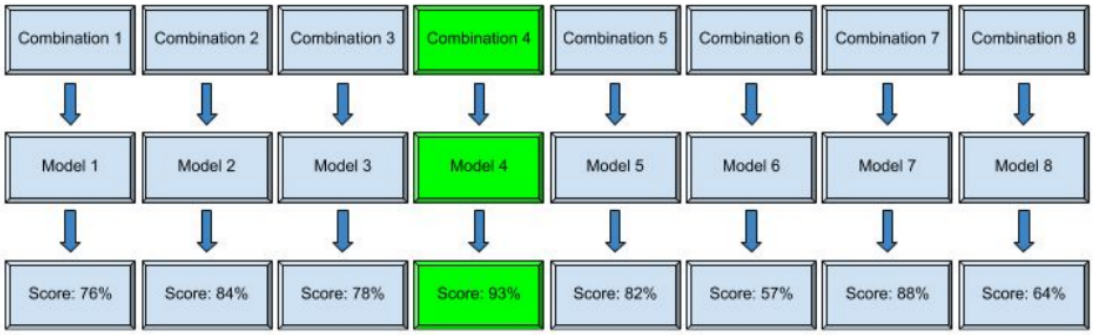
Hyperparameter Optimization

To perform the refinement process, I have incorporated Hyperparameter Optimization and

Feature selection using trial and error method.

To get the best possible hyperparameters, trial and error method is used. In this method, I have created a list of all possible combinations of the hyperparameters and trained the model on each of the combinations and validated the model using the test set. The parameters with the highest validation score will be chosen. **Fig**.4 is an example image.

Fig.4: Scoring for all possible combinations of the hyperparameters



Feature Selection

Feature Selection is the process where we automatically or manually select those features which contribute most to our prediction variable or output in which we are interested in. Here, we have again used trial and error method to select the best features. We have created a list of all possible combinations of feature attributes and trained the model on each combination and validated the model with the testing set. The combinations with the highest validation score will be selected.

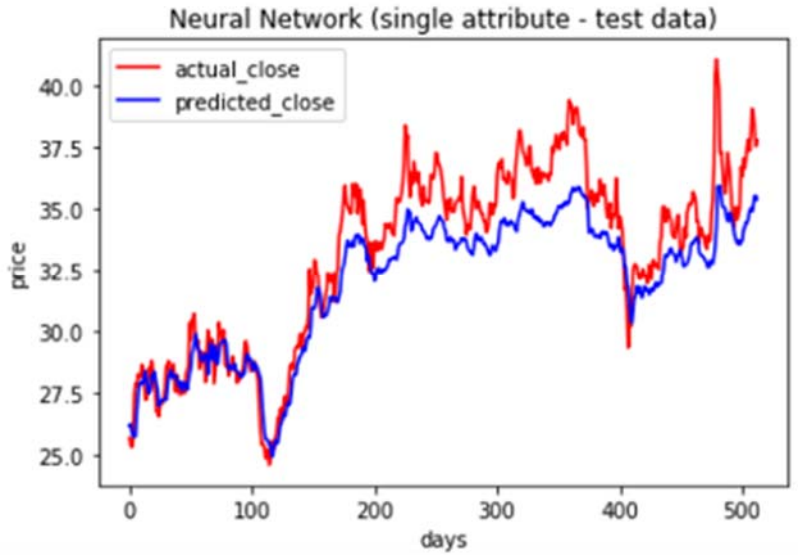
- Reduces Overfitting : Less redundant data means less opportunity to make decisions based on noise.
- Improves Accuracy : Less misleading data means modeling accuracy improves.
- Reduces Training Time : Fewer data points reduce algorithm complexity and algorithms train faster.

Results

Model Evaluation and Validation

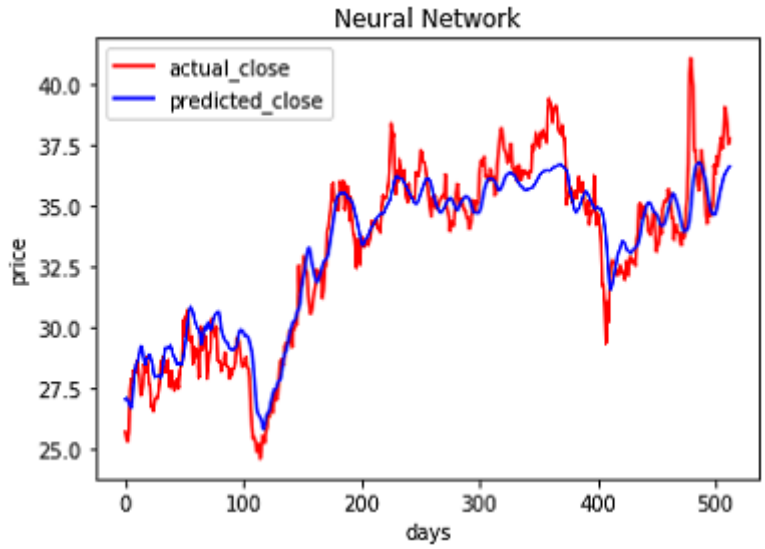
Single Feature Model

This model achieved an r2 score of 0.793 and mse value of 2.799 on the test set.
Fig.5: plot for Single Feature Model on test set.



Hyperparameter Optimized Model

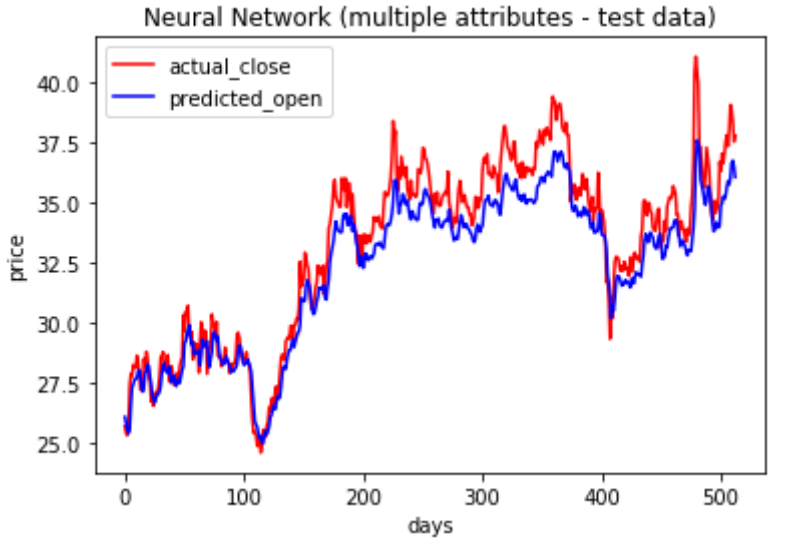
In the process of hyperparameter optimization, the best model achieved an r2 score of 0.89 and mse value of 1.45 on the test set.
Fig.6: plot for Hyperparameter Optimized Model on test set.



Final Model

After feature selection process, the best model achieved an r2 score of 0.91 and mse value of 1.09 on the test set.

Fig.7: plot for the Best Model on test set.



As the r2 score value is nearly close to 1, so we can say that the model is pretty good in predicting the next day's closing price.

Justification

We trained the model using the data of cmb China stock and first predicted the closing prices using single attribute only yielding an r2 score of 0.793 and mse value of 2.799 on the test set. Then after doing hyperparameter optimization and feature selection, we have

seen the results yielding an r2 score of 0.91 and mse value of 1.09 on the test set in predicting the closing price.And came to the conclusion that our LSTM model is good at predicting time dependent data analysis problem when proper optimization techniques are used.

Conclusion

Reflection

Using neural networks to forecast stock market prices will be a continuing area of research as researchers and investors strive to outperform the market, with the ultimate goal of bettering their returns. Since there are many indeterminate parameters that directly or indirectly affect stock market, each one of them cannot be taken into account. So our model only depends on the relationship of our selected parameters of the stock market.

Improvement

Interesting results and validation of theories occured as neural networks are applied to more complicated problems. To improve it further, we can also apply some other method or model:

- Sentiment analysis using news headline and can club together the results of the two models to get the more concrete predictions.
- Take more number of hyperparameters into account to optimize the model further, but note that training the model may take plenty of time.
- Use new method, such as Transformer[5], to reconstruct the model may get better performance.

[5] Transformer, Google(2017), Attention Is All You Need, https://arxiv.org/abs/1706.03762