Udacity Machine Learning Engineer Nanodegree

Capstone Project Report

Yingping Yang

1. Introduction

Investment firms and hedge funds have been using financial models to better understand market behavior and make profitable investments and trades. A wealth of information is available in the form of historical stock prices data, suitable for machine learning algorithms to process. The purpose of this project is using neural network to predict future stock price. We will predict the rate of change (ROC) of the close price and the actual close price can be easily calculated. More specifically, we will train the model using data of 2014, 2015, 2016 and use the trained model to predict ROC in 2017. This project is inspired by a time-series prediction project from the Udacity Machine Learning Engineer Nanodegree, Machine Learning Case Study course.

In this capstone project, model will be trained and deployed using AWS Sagemaker. The data used is the Apple Stock Price data from 2013 to 2018, which is recorded in the "Apple 2013-2018.csv" file. The model used is the DeepAR Forecasting Algorithm built in AWS Sagemaker. After that, the prediction results will be compared with the actual price data and evaluated by the Rooted Mean Square Error(RMSE). The workflow of this project is:

(1) Loading and exploring the data

(2) Creating training and test sets of time series

(3) Formatting data as JSON files and uploading to S3

(4) Instantiating and training a DeepAR estimator

(5) Deploying a model and creating a predictor

(6) Evaluating the predictor

2. Data Exploration and Preparation

For a certain stock, the raw daily trading data include Open, High, Low, Close, Volume. In this project, Close Price will be used as the input data as well as the prediction output. We first check if there is any missing value. The next step is extract the Close Price column from the whole Data Frame. After that, close prices from 2013 to 2018 are visualized In Figure 1.

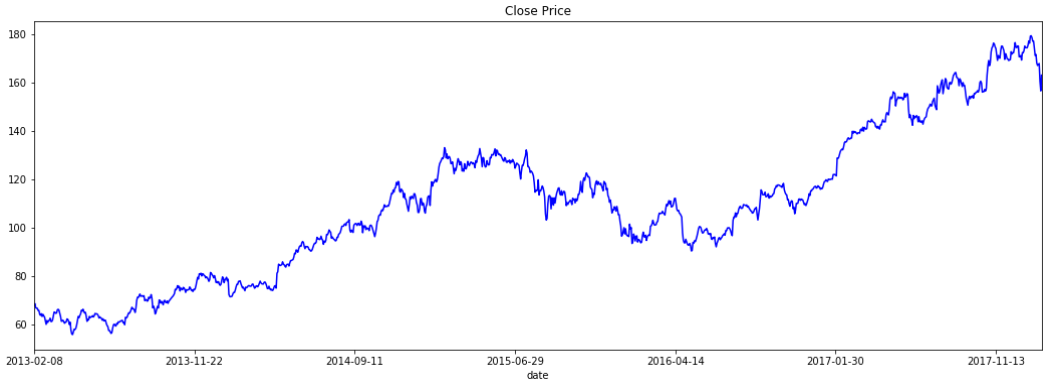

Figure 1: Close Price

We can see there is an increasing trend over five years despite of a slight downward trend from July 2015 to May 2016. Next step is calculating the rate of change (ROC) of the close price. ROC is calculated as Equation 1, where r_t is ROC at time t, p_t is close price at time t and p_{t-1} is close price at time t-1.

$$r_t = \frac{p_t - p_{t-1}}{p_{t-1}} \qquad \textbf{(Equation 1)}$$

We will use ROC as input data and the output prediction will also be ROC. Once we have ROC for every points, the actual close price can be recovered using Equation 2:

$$p_t = (1 + r_t)p_{t-1} \qquad \textbf{(Equation 2)}$$

ROC over the five years is visualized in Figure 2.

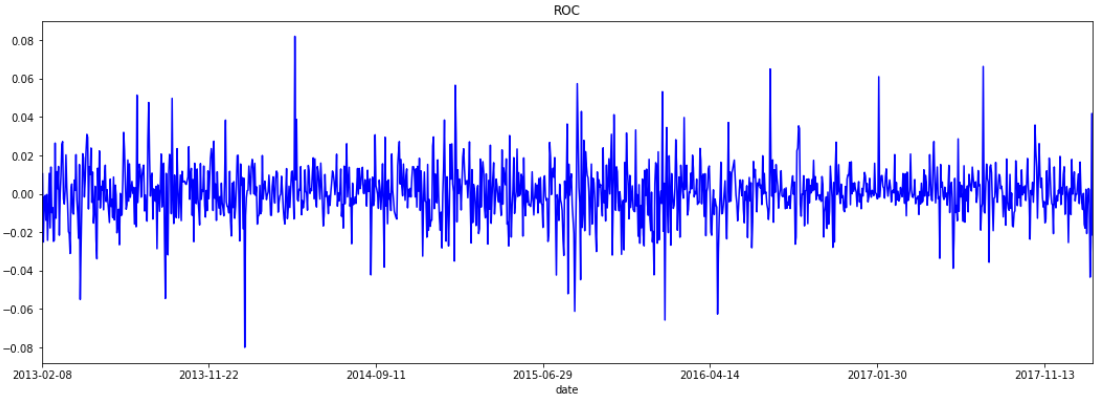

Figure 2: ROC

We will also explore the distribution of ROC in Figure 3. From the plot, we can see it is normally distributed with mean 0.

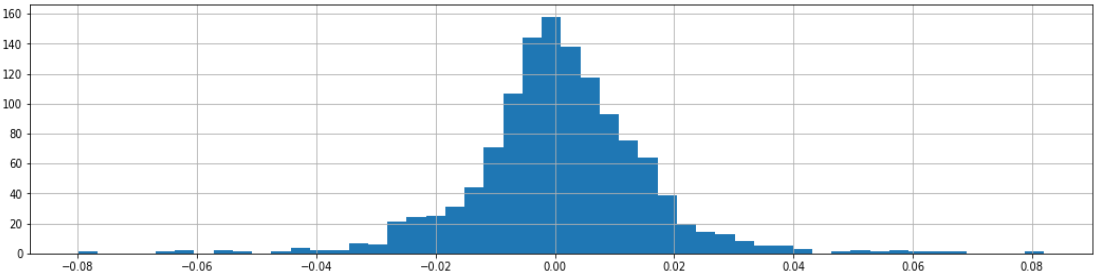

Figure 3: Distribution of ROC

After some data explorations, we will create time series from the given data. There are 252 trading days each year and the entire time series is used as the training set. The test set is the last 20 trading days in every year. Figure 4 shows the train set and test set of 2014 data.

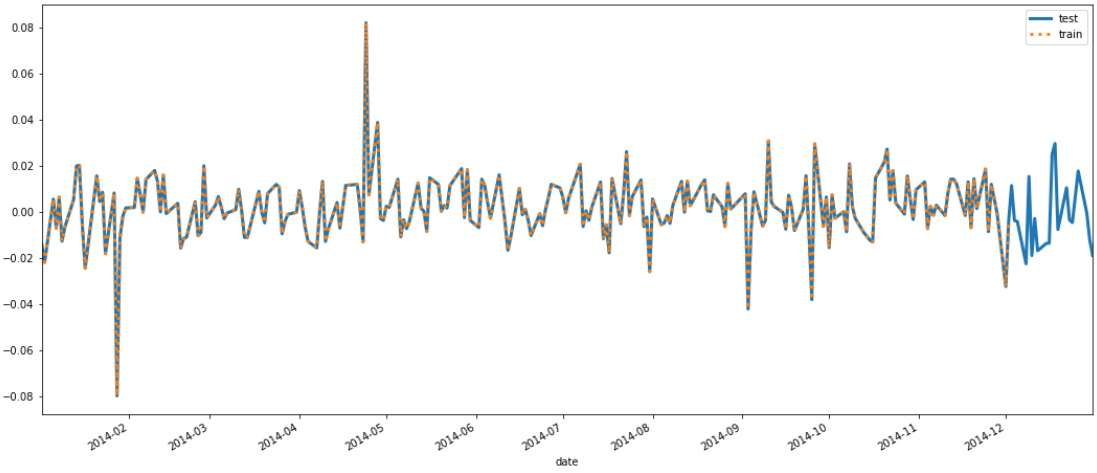

Figure 4: Train Set and Test Set

According to Sagemaker DeepAR documentation, DeepAR expects to see input training data in a specific JSON format. Hence, we transform the original data into JSON format and save it locally. After that, the preprocessed data are uploaded to S3 and ready to be used to train and test our model.

3. Model Training

The model used in this project is DeepAR, which is a built-in deep learning algorithm in AWS Sagemaer. DeepAR forecasting algorithm is a supervised learning algorithm for forecasting scalar (one-dimensional) time series using recurrent neural networks (RNN). A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. During training, DeepAR accepts a training dataset and an optional test dataset. It uses the test dataset to evaluate the trained model.

Our next step is to instantiate an estimator. Some estimators have specific, SageMaker constructors, but not all. Instead we need to create a base estimator constructor and pass in the specific image (or container) that holds a specific model. For DeepAR model, we need to specify a image "forecasting-deepar" and pass it to the base estimator constructor. After estimator instantiating, we need to define some DeepAR hyperparameters that define the model size and training behavior. Values for the epochs, frequency, prediction length, and context length are required parameters. Apart from those, we also set number of cells, number of layers, minibatch size, learning rate and early stopping patience.

After setting the hyperparameters, we can start the training work by calling the estimator fit method. The training takes 6min 43s. After training the model, the next step is model deployment, which takes 8min 31s to finish. From the output window, we can see the performance of our model evaluated by test:RMSE and test:mean_wQuantileLoss. Test:RMSE is defined as the root mean square error between the forecast and the actual target computed on the test set. Test:mean_wQuantileLoss is defined as The average overall quantile losses computed on the test set. For our model, test:RMSE = 0.0149326097809; test:mean_wQuantileLoss = 1.05884942. The test results of last 20 trading days in 2014, 2015 and 2016 are visualized in Figure 5, Figure 6 and Figure 7 respectively. From the plots we can see our model capture the trends of the ROC movements.

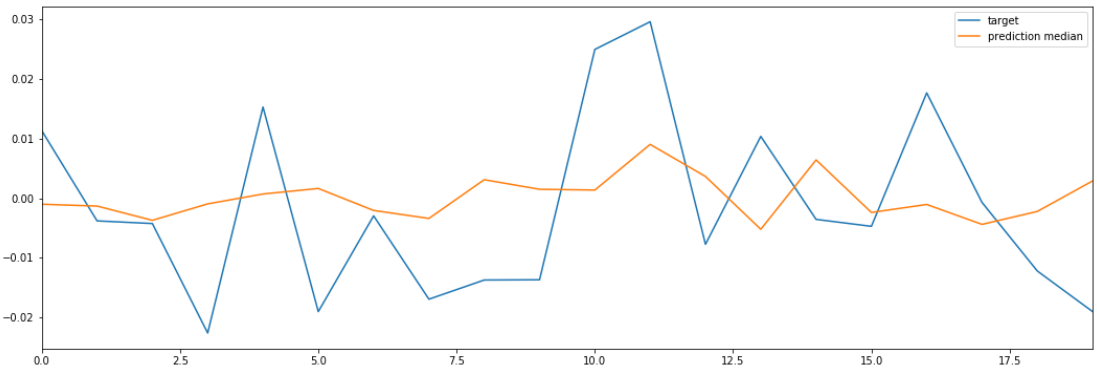

Figure 5: Test Result of the Last 20 Trading Days in 2014

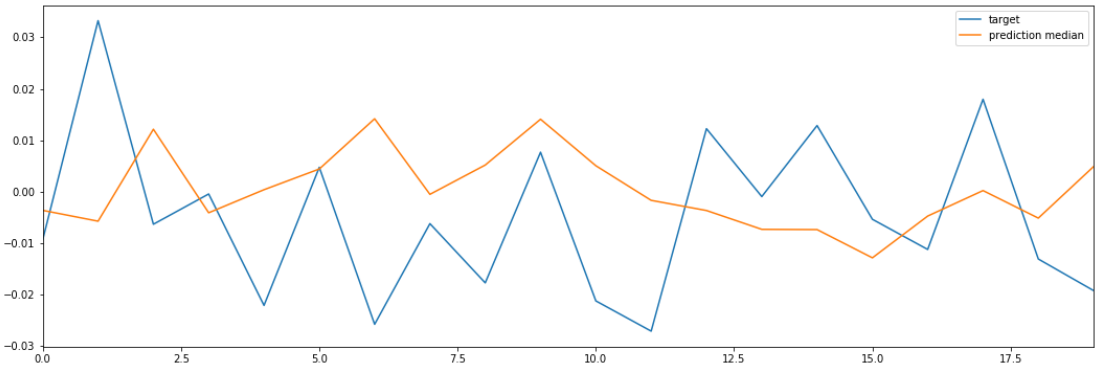

Figure 6: Test Result of the Last 20 Trading Days in 2015

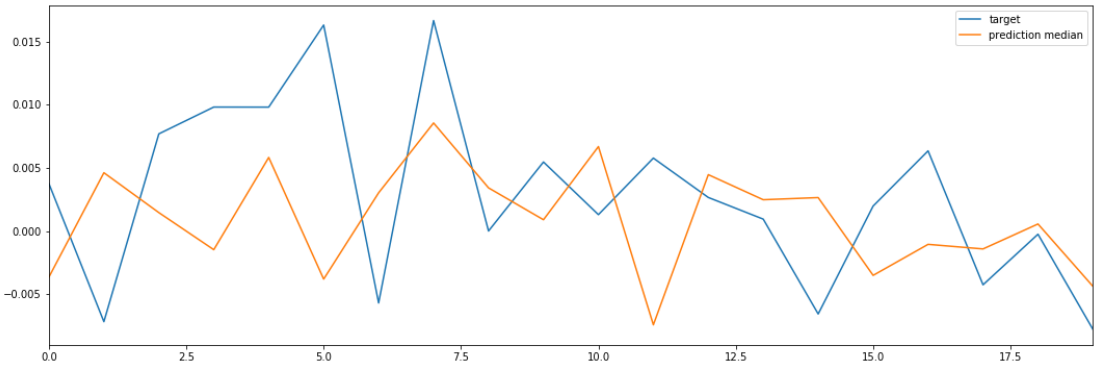

Figure 7: Test Result of the Last 20 Trading Days in 2016

4. Prediction and Evaluation

Remember that the goal of this project is to predict the ROC in the future. We use the data in 2014, 2015 and 2016 to train so the data in 2017 are never seen by the model. The output starting from 2017 is regarded as the future prediction. To obtain the prediction result, we first format the request data. We would like to predict the first 60 trading days in 2017, which is separated into 3 requests, 20 days each. The prediction results for three requests are shown in Figure 8, Figure 9 and Figure 10 respectively. In the plot, the 80% confidence intervals are also shown.

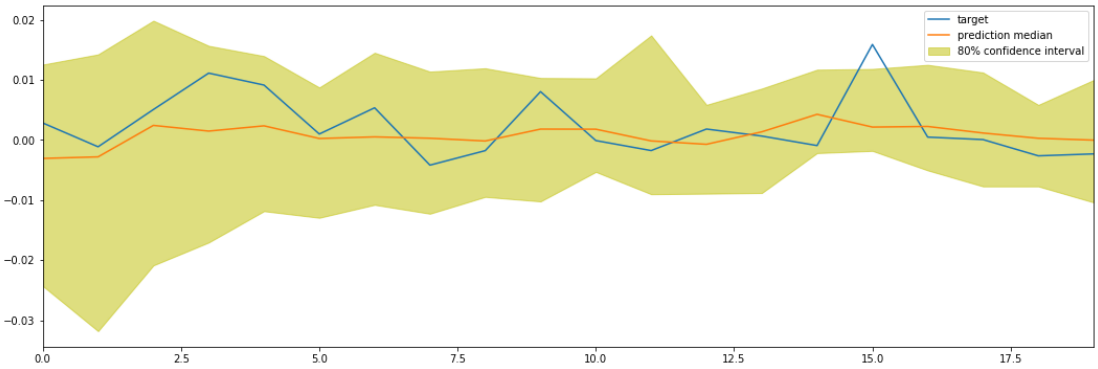

Figure 8: Predictions for the First 20 Trading Days

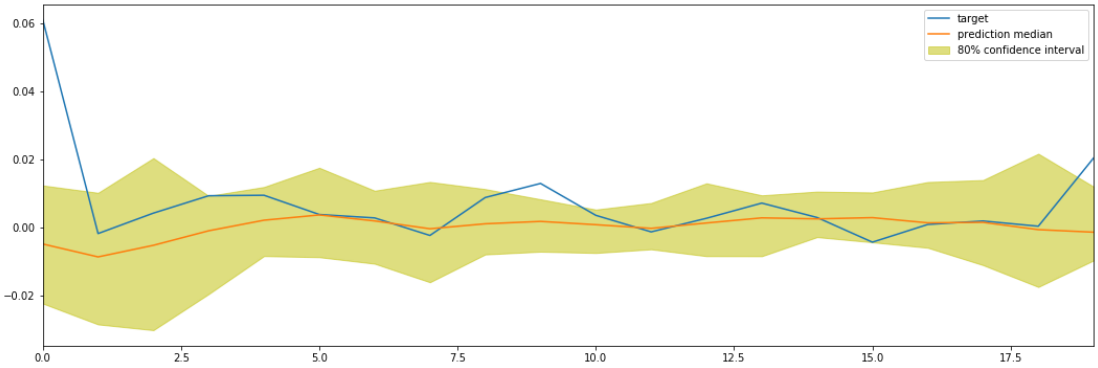

Figure 9: Predictions for the Second 20 Trading Days

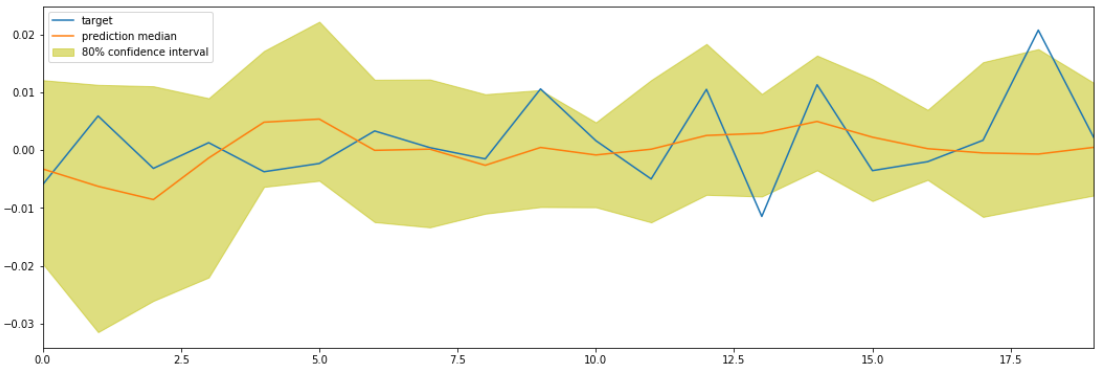

Figure 10: Predictions for the Third 20 Trading Days

From the result, we can see our prediction for next 60 trading days in 2017 is pretty good. The prediction median follows the trend of the actual close price. In addition, nearly all actual close price points are in the 80% confidence interval.

Next, we would like to measure our prediction quantitatively using RMSE, which is defined in Equation (3). $\widehat{y_t}$ is the predicted value at t, y_t is the actual value at t and T is total points in the time series.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(\widehat{y_t} - y_t)^2}{T}} \qquad \textbf{Equation(3)}$$

RMSE for our prediction results:

Day 1 to Day 20: ***RMSE_1*** = 0.0050848803184042534

Day 21 to Day 40: ***RMSE_2*** = 0.016372285316725375

Day 41 to Day 60: ***RMSE_3*** = 0.008013909721949828

5. Conclusion and Future Improvement

In this project we follow the Machine learning workflow to build a model that can predict future daily stock price. As we can see from the plots and RMSE values, the results of our prediction is pretty good. However, this project only predicts the daily close price of one particular stock. This can only be used as a very rough analysis of financial market. In the real market, trading data are collected every millisecond. In order to design AI trading strategies that can be applied in the real market, more detailed historical trading data should be used to train the model. Correspondingly, more sophisticated model may provide better results.