Machine Learning Engineer Nanodegree

Capstone Proposal

Sandeep Paulraj December 25th, 2017

Proposal

Proposal for stock price estimator.

This proposal also has an associated ipython notebook stock__price__estimator__proposal.ipynb that has the initial data exploration and setup.

`stock_price_estimator_proposal.ipynb`

Domain Background

Fundamentally, owning stocks is a way to grow wealth. Usually investors buy stocks after some due dilligence and research. If the stock price goes up, the investor has the option of selling the stocks and making a profit. At the same time, several companies shell out dividends to shareholders if they own stock. This too is a way to accumulate wealth. However, traditional buy and hold techniques are no longer in vogue these days. Retail investors find it very tough to enter a particular stock since stock prices do vary and large institutional investors hold sway. High Frequency trading results in significant stock price variation within a trading day. Also fundamental research though still very important is kind of taking a back seat to algorithm based trading.

Machine learning technques are being used to make investment decisions. Machine learning techniques have carved out a niche for themselves in various domains and especially those domains where there is a plethora of data. Stock Prices are a very appealing domain since they provide several stocks and large amounts of data. Over the last few years we have also started hearing more about several hedge funds and investment banks beginning to use these techniques. I intend to investigate some of these techniques to predict stock prices.

To be precise, i would like to be able to predict Broadcom(AVGO) stock price. This company has been in the news lately for its takeover attempt of Qualcomm. Broadcom is also a very large Apple Supplier. Also owning Broadcom stock myself, i want to gauge if i can come up with a model to better gauge future price movement based on available data.

Problem Statement

Stock Prices fluctuate from day to day and to be precise, fluctuate by the second. Using publicly available data stock price data, i will attempt to predict the adjusted close price of the stock for the next seven trading days. If we are able to gauge the closing stock price, we might be able to make smart trading decisions based on the predictions. By giving a start date and a finite set of following trading days(in other words a range of trading days), it should be possible to predict the following 7 days adjusted closing stock price. Basically we have a range of trading days and we have to predict the closing adjusted stock prices of the 7 following days immediatley after this range.

Datasets and Inputs

It is possible to obtain stock price financial data from various sources. It is also possible to use python api and the yahoo finance library to obtain this data. For some reason, i am having trouble installing the yahoo finance library in python 3.6. So i have decided, to obtain the csv data from the yahoo finance website and read in the dataframe using pandas. For the initial exploratory analysis, I read in the data and realize that the dates increase, i.e in the csv file and data frame February 1 will come before February 2. The first thing that needs to be done is to reverse this order.

Next, i remove the first 7 rows. The reasn I do this is as follows. We have to predict the ensuing 7 trading days stock price. Hence the first 7 rows which happen to be the last 7 trading days will definitely have atleast 1 piece of data that we will not be aware of. To be precise let us take an example. As we are setting up and enhancing data frame, yesterday's data can be appended with today's closing stock price; however we don't know the next 6 trading days stock price. This is something which will be good to predict.

After saving of the first seven rows, i append the remaining data frame with seven columns that have the seven following trading days adjusted closing stock price. These seven columns will become the "prediction" columns. I also have another column where I store the difference between the highest and lowest daily stock price.

It is important to use standard pandas routines to set up the dataframe. This essentially will result in a more elegant and cleaner final solution. Please take a look at the accompanying notebook to look at all the exploratory analysis

Solution Statement

We are dealing with time series data. Also we fundamentally have a regression problem. This is not a classification problem. We have to predict an actual adjusted stock price; not whether the stock goes up or down. We have to predict seven outputs instead of one that I have been accustommed to do. So let us take an example. Say we need to predict 7 trading days closing stock price. We will have various inputs that are available to use such as trading volume, opening price, high price, low price. With this we can use regression techniques to predict the following seven days closing stock price. Now, we have already setup our data to know the following 7 days closing stock price. Thus we will have both actual closing stock price and predicted stock price based on our model. With this we can gauge how well our model is behaving. It is intended that the model will predict stock price withing a +- 5% range. Though this project, i will attempt to predict 7 trading days worth of stock price, it is possible to predict more stock prices. For the sake of this project, i will be predicting the next 7 days stock price only.

Benchmark Model

As mentioned to some extent above, we will have actual adjusted closing stock prices. It is these same stock prices that we will be predicting. So we have both actual and predicted prices.This is essentially our benchmark. We can use this information to gauge how good our model is. In a way, we can consider actual adjusted closing stock prices as our benchmark.

Evaluation Metrics

I will be leveraging sklearn in this project. From sklearn metrics we will have access to an array of metrics from our model. The main evaluation metric I will be using is the root mean squared error. The root mean squared is simple to calculate and can be calulated as shown below. Based on previous projects and experience, i don't think a metric exists to calculate this for us. This has to be derived and is simple to derive as can be seen below.

```
from sklearn.metrics import mean_squared_error
from math import sqrt

rms = sqrt(mean_squared_error(y_actual, y_predicted))
```

Now the difference between actual and prediction prices can be positive or negative. Hence it is important to take the square of the difference. We then have to take the mean of these squared values and finally take the square root.

Project Design

As mentioned previously my project proposal is associated with an ipython notebook.

`stock_price_estimator_proposal.ipynb`

The above notebook will be a starting point for my final project as well.

After obtaining my data, I enhance my data with extra columns. Seven of these columns will be for the 7 following trading days stock price. How I do this can be seen in the notebook. I explain various code cells in the notebook. I also have another column to store the difference between the highest and lowest daily stock price. As i continue this amazing journey, I might add other data columns that I derive. At this point of time, I don't plan to append my data from other sources. On many occassions a simple model is a good model. We don't want to over fit the data. Practically speaking a lot of factors go into a stock price. On any particular day, stock price will be somewhat based or related to the index that it is listed on. Then the index itself may and in all likelihood depend on the closing price of other indices. Let us take an example. Say the US has a good trading day and the markets are positive in Asia starting in Japan. All is well until say some bad data originates in Europe. Europe turns negative and this results in a negative start for US the next day. This feeds into a company stock price. For the sake of this project, I don't want to complicate things and will keep things simple. I will make predictions based only on that tickers available data.

As mentioned previously, this is a regression problem. But there are a few salient featuers of this project that I have not previously dealt with previously. The project will have to predict 7 different values. Based on some reserach that i have done, I have a problem that is essentially a problem of multi output regression. Multi output regression can be used with any regressor using MultiOutputRegressor that is available in sklearn. To start, I will be using Linear Regression. I will gauge results after using Linear Regression with MultiOutputRegressor. However, it is very likely that i will be using Support Vector Regression as well. I may also experiment with Decision Tree Regressor and Random Forest Regressor. But it is important for the model to be simple so I don't want to end up with something fancy. Having done several udacity projects in both machine learning and self driving car, I have learned that simple is good and sometimes better.

After running regression scenarios, I will look at the metrics to gauge how well my model is doing and whether machine learning is a viable approach to predicting stock prices.
