Machine Learning Engineer Nanodegree

Capstone Project

Pengfei Gao

December 26, 2017

I. Definition

Project Overview

Predicting the stock price from chaotic market data has always been attractive for both investors and researchers, technical analysis and fundamental analysis are the two main schools of thought when it comes to analyzing the financial market. Technical analysis looks at the price movement of a security and uses this data to predict future price movements while fundamental analysis instead looks at economic and financial factors that influence a business. E.g. technical analysis is trying to interpret the stock indicator like 'sample moving average' which is the average stock price over a period, 'moving average convergence divergence' which reveal changes in strength, direction, momentum, and duration of a trend in a stock price in order to predict the price trend, fundamental analysis trying to analysis the company financial report like balance sheet, incoming statement, cashflow statement in order to calculate company's value and to see whether the company's stock price is overvalued or underestimated. Machine learning technology has already applied to stock prediction (*https://www.researchgate.net/publication/259240183_A_Machine_Learning_Model_for_Stock_Market_Prediction*).

Problem Statement

This project is focusing on the technical analysis and is a binary classification problem, I will use the stock indicator such as SMA, MACD, Bollinger Bands... to predict the stock price trend in next few days, i.e. the output for this project has 2 classes(+1, and -1), I will classify the output as +1 if next 10 day's average adjusted close price is greater than today's adjusted close price and -1 if next 10 day's average adjusted close price is less than today's adjusted close price. It is very hard to predict the stock price trend by just one indicator nowadays, but I believer by combining some the indicators and use machine learning methods(e.g. SVM, Adaboosting) to create a model, we may get a better prediction than just one or few indicators.

Metrics

I will use accuracy and f-1 score to compare the performance for benchmark model since the classes for stock data will not be balanced (stock values increase more than they decrease overall) and F1 metric will not be affected by class imbalances.

II. Analysis

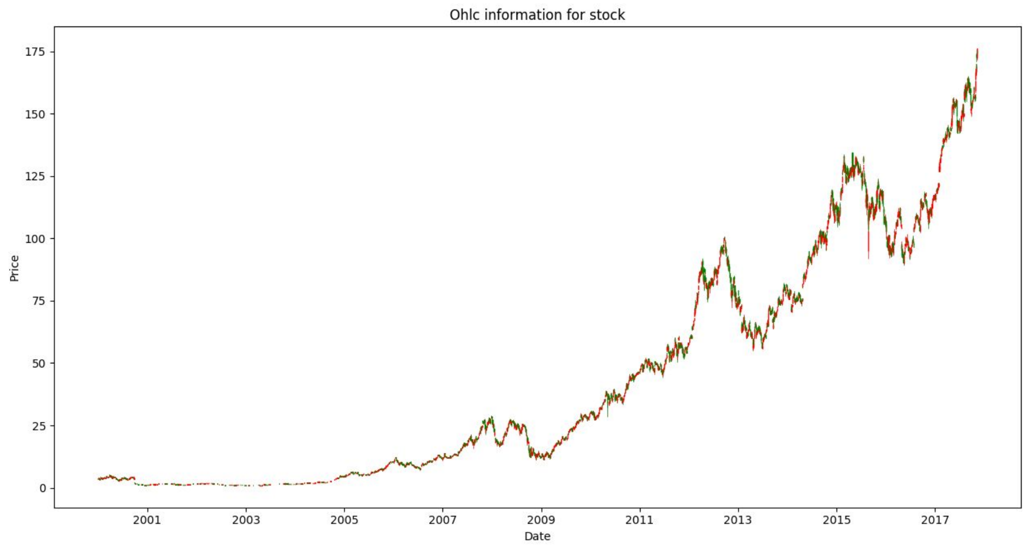
Data Exploration

Yahoo finance(*https://finance.yahoo.com/*) provide the stock information(i.e. open, high, low, close, adjusted close, volume) for any given days, and pandas_datareader.data provide the interface to accessing these data, e.g. if we want to get Apple's stock information from 2010-1-1 to 2016-12-31, we just need to call pandas_datareader.data.DataReader('AAPL', datetime.date(2010,1,1), datetime.date(2016,12,31)), the function will return a pandas dataframe which contains the columns of date, open, high, low, close, adjusted close, volume. Then I will use these raw data to compute the indicators(e.g. 6 days SMA(sample moving average) = df['Adj Close'].rolling(window = 6).mean()), and treat all these indicators as features.
The label is the average price in next 10 days compare with today's price, +1 if greater than current price and -1 if less than current price, e.g. if the given date is 2010-1-1, and the price is 100, we want to predict 10 days price trend, the average price for price from 2010-1-2 to 2010-1-11 is 110, we say the label of 2010-1-1 for the given stock is +1 assuming all these days are trading days.
I will use these 5 stocks ['MMM', 'MSFT', 'AAPL', 'XOM', 'BAC'] , and since stocks are heavily influenced by the market, i.e. if market moves up, the stock price is more likely to move up, and vice versa, I will also include the market index(i.e. NASDAQ, SP&500)
I will use following 29 features for the stocks :
['Adj close' 'SMA3', 'EMA6', 'EMA12', 'EMA26',
'MACD12_26_9', 'MACD_signal12_26_9', 'SMA14', 'Upper_band14', 'Lower band14', 'RSI6EMA', 'RSI12EMA',
'Momentum1', 'Momentum3', 'RateOfChange3', 'RateOfChange12', 'CCI12', 'CCI20', 'WILLR14', 'ATR14',
'TripleEMA6', 'OBV', 'MFI14', 'PDI14', 'NDI14', 'ADX14', 'PDI20', 'NDI20', 'ADX20']
These Indicators are computed against different period, e.g. SMA3 is the 3days simple moving average, the formula for these indicators can be find at *http://stockcharts.com/*.
I will include 2 market index NASDAQ and SP&500, so the total features for the stocks are 29*3 = 87.
I plan to use the data from 2010-1-1 to 2015-12-31 as training data and data from 2016-1-1 to 2016-12-31 as testing data.

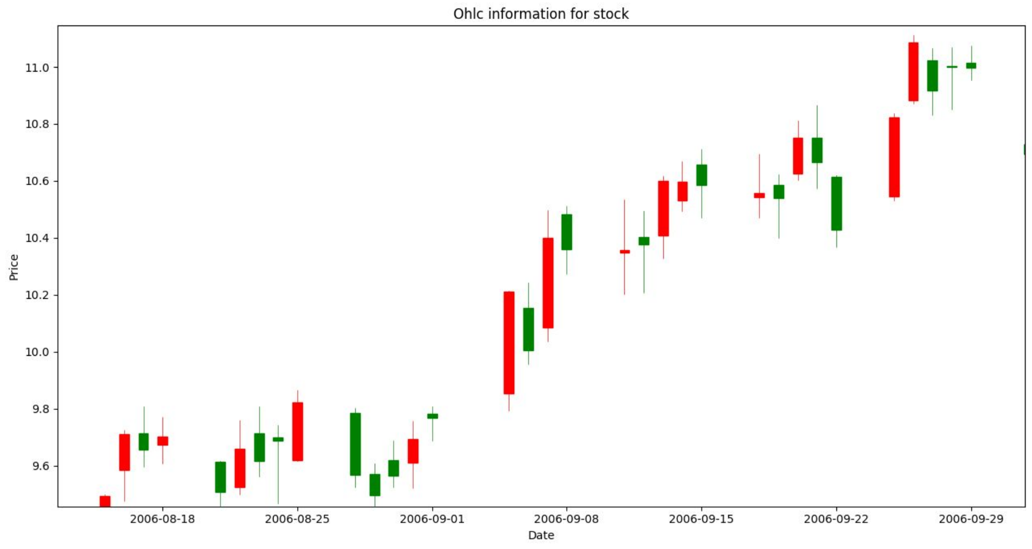
Exploratory Visualization

The following figure is the candlestick plot which include open, high, low close information for stock AAPL.

The x-axis is the date information, and y-axis contains open, high, low, close information for each day.

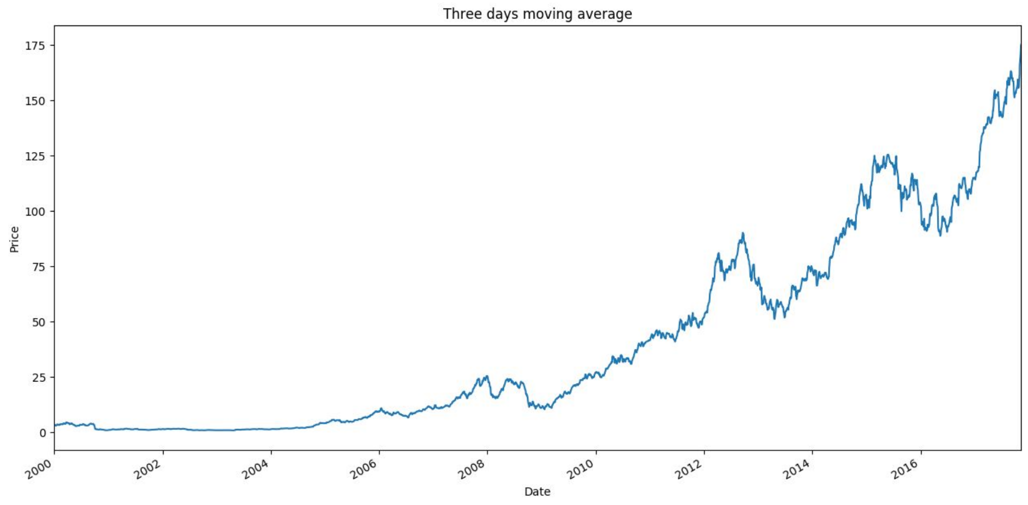


The detail information for candlestick plot can be found below:



The red stick means open price is less than the close price i.e. the stock price is moving up in that day, and the green stick means open price is more than the close price i.e. the stock price is moving down in that day.

All indicators are derived from Open, High, Low, Close, Adj Close and Volume, e.g. three days moving average is the the average price for last three days, see plot below:



Algorithms and Techniques

Support Vector Machine which is one of the best binary classifiers will be applied in this project. The goal of the SVM is to train a model that assigns new unseen objects into a particular category. It achieves this by creating a partition of feature space into two categories.

Consider an n-dimensional feature vector x = (X1,...,Xn) . We can define a linear boundary (hyperplane) as

$a0 + a1X1 + ... + anXn = a0 + \sum aiXi = 0$

Notice that it can be rewritten as dot product:

a0+**a***X = 0 where * stands for dot product

Then elements in one category will be such that the sum is greater than 0, while elements in the other category will have the sum be less than 0. With labeled examples, a0+**a***X = y, where y is the label. In our classification, $y \in \{-1,1\}$. The optimal hyperplane is such that we maximize the distance from the plane to any point. This is known as the margin. The maximum margin hyperplane (MMH) best splits the data. These points are known as the support vectors, and the hyperplane is known as a Support Vector Classifier (SVC) since it places each support vector in one class or the other.

Specifically, I will use sklearn.svm.SVC as classifier, The following parameters will be tuned to optimize the classifier**:**

C : float, optional (default=1.0)

Penalty parameter C of the error term.

kernel : string, optional (default='rbf')

Specifies the kernel type to be used in the algorithm. 'linear', and 'rbf', will be applied in this project.

gamma : float, optional (default='auto')

Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. If gamma is 'auto' then 1/n_features will be used instead.

Grid search and cross-validation, will be applied to auto tuning the parameters.

Grid search technique is a way to automatic working through multiple combinations of parameter tunes, it tries all the parameter combinations and return a parameter combination which gives the best performance which may measured by cross validation.

Since stock data is the time series data, time series splitting will also be applied in grid search cross validation.

Benchmark

The benchmark is a simple trading strategy for stock market, i.e. if current stock price is greater than the 10 days simple moving average price, we predict the average price for next 10 days will greater than current price, and vice versa. The result of accuracy and f1 score for the benchmark can be find below:

stock: MMM accuracy: 0.5515873015873016 f1score: 0.34682080924855496

stock: MSFT accuracy: 0.4246031746031746 f1score: 0.2995169082125604

stock: AAPL accuracy: 0.5753968253968254 f1score: 0.4928909952606635

stock: XOM accuracy: 0.5277777777777778 f1score: 0.37696335078534027

stock: BAC accuracy: 0.6071428571428571 f1score: 0.547945205479452

III. Methodology

Data Preprocessing

Assuming we already get all raw data, (Open, High, Low, Close, Volume, Adjusted close) and calculate all the features/indicators and labels for stocks and index(SP500, NASDAQ) and save them as csv file in local machine. The step for Data Preprocessing as as follows:

1. Use pandas's dataframe to load the individual stock and index, and combine their features, then we can have a big dataframe which include indicators for the individual stock as well as index.
2. Drop the labels from the big dataframe. The label for this project is simple moving average price for next 10 days compare to current price, +1 if greater than current price, and -1 if less than current price, the column name for this label is 'Updown10'
3. Get training data and testing data from dataframe, the training data is from 2010-1-1 to 2015-12-31, and the testing data is from 2016-1-1 to 2016-12-31.
4. Normalise all training and testing data via MinMaxScaler since some features is extremely larger than others.

Implementation

During the implementation, the first step is to define the classify, specifically, svm.svc() has been applied, then I use f1 score as evaluation matrix, specifically, I use scorer = make_scorer(f1_score, pos_label = -1, average = 'binary') from sklearn.metrics, I also applied the grid search, parameters for grid search is as follows:

```
SVMParm = [

   {'C': [1, 10, 100, 1000], 'kernel': ['linear']},

 {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},

 {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['poly'], 'degree':[1,2,3,4,5]},

 {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['sigmoid']}

  ]
```

Since stock data is the time series data, the default cross validation for grid search will use future data as training data and past data as cross validation data which is not applicable for time series data, I applied time series split technique for the cross validation, specifically I applied the TimeSeriesSplit from sklearn.model_selection. And split training data into 5 parts:

```
my_cv = TimeSeriesSplit(n_splits=5).split(X_train)
```

And then use this data split cross validation for grid search:

```
grid_obj = GridSearchCV(clf, parameters, scoring = scorer, cv = my_cv)
```

The training and cross validation data set during grid search will then be separate as follows:

```
TRAIN: [2010] cross validation: [2011]
TRAIN: [2010, 2011] cross validation: [2012]
TRAIN: [2010, 2011, 2012] cross validation: [2013]
TRAIN: [2010, 2011, 2012, 2013] cross validation: [2014]
TRAIN: [2010, 2011, 2012, 2013, 2014] cross validation: [2015]
```

Last, I use grid search technical to tuning the parameters and find the best classifier and use the classify to test the accuracy and f1 score on testing data.

Refinement

For the initial solution for the project, I use the default pos_label parameter which is 1 for the f1_score as evaluation matrix for grid search and final evaluation , i.e. performance of precision and recall for the price up trend and it turns out the performance for f1_score is very good:

```
stock: MMM accuracy: 0.6547619047619048 f1score: 0.7913669064748202

stock: MSFT accuracy: 0.5833333333333334 f1score: 0.7368421052631579

stock: AAPL accuracy: 0.5952380952380952 f1score: 0.7462686567164178

stock: XOM accuracy: 0.6111111111111112 f1score: 0.7586206896551725

stock: BAC accuracy: 0.5674603174603174 f1score: 0.7240506329113924
```

However I find the classifier will always predict 1 i.e. it will always predict the price moving up.

So I changed the pos_label parameter to -1 to find the performance of precision and recall for price down trend, and use this as evaluation matrix for grid search and final evaluation.

IV. Results

Model Evaluation and Validation

```
  Stock  Accuracy   F1score                           bestTrainParm

0   MMM  0.654762  0.539683              {'C': 1000, 'kernel': 'linear'}

1  MSFT  0.547619  0.583942  {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
```

```
2  AAPL  0.603175  0.431818               {'C': 10, 'kernel': 'linear'}

3   XOM  0.539683  0.504274              {'C': 100, 'kernel': 'linear'}

4   BAC  0.587302  0.365854  {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}

mean accuracy: 0.5865079365079365, mean f1score: 0.4851138980300454
```

The table above is the performance and best train parameter find in grid search for the SVM model, we can find all accuray are above 50%, and mean of the accuracy is 58.7%, and accuracy for stock MMM and AAPL are above 60%, for f1 score, 3 of the 5 are above 50%, the highest f1 score is stock MSFT which is 58%, the lowest is stock BAC which is 36.6%

Justification

Result for benchmark model:

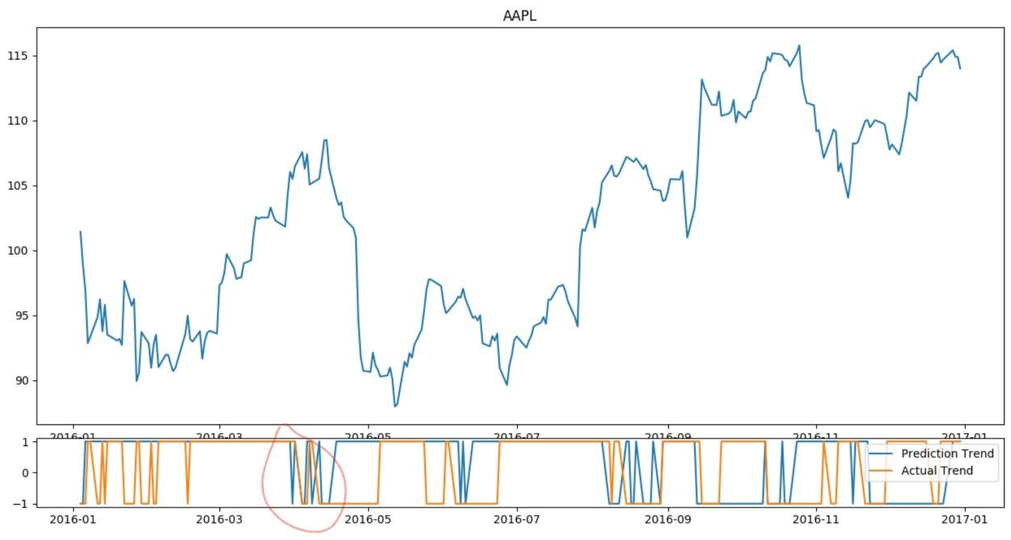
```
stock: MMM accuracy: 0.5515873015873016 f1score: 0.34682080924855496

stock: MSFT accuracy: 0.5952380952380952 f1score: 0.5048543689320388

stock: AAPL accuracy: 0.5753968253968254 f1score: 0.4928909952606635

stock: XOM accuracy: 0.5277777777777778 f1score: 0.37696335078534027

stock: BAC accuracy: 0.6071428571428571 f1score: 0.547945205479452

mean accuracy: 0.5373015873015874, mean f1score: 0.4128274537973143
```

Notice that the mean accuracy of the SVM model is 58.7%, and the mean f1_score is 48.5%, the improvement of mean accuracy is about 4%, and the improvement of f1_score is about 7%, so I think the SVM model have a high performance for most stocks than this trading strategy, but for some other stocks, the benchmark strategy is working better.

V. Conclusion

Free-Form Visualization

The figure below is the price, prediction trend and actual trend for stock ‘AAPL’, I think this can help us as a reference to do the investment, e.g. it predict the price will went down on April 2016 before the price actually went down.



Reflection

The process used for this project can be summarized in following steps:

1. Collect raw information from these 5 stocks ['MMM', 'MSFT', 'AAPL', 'XOM', 'BAC'] and these 2 index [S&P500, NASDAQ] from yahoo finance via pandas dataframe, and store them in local machine.
2. Create features/indicators and labels for each stock and market index through the raw data via pandas dataframe and store them in local machine.
3. Combining the stock dataframe and index dataframe, normalize the dataframe and get the training data and testing data from dataframe.
4. Applying SVM, grid search cross validation, time series split technique to training the model.
5. Testing and evaluating the model.

It turns out the most time consuming step for this project is step2, i.e. creating features/indicators from the raw data. Some indicators is not very straightforward(e.g. ADX,http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:average_directional_index_adx), first I need to understand the indicator, then I need to calculate the indicator via pandas dataframe and store them as spreadsheet file, last I need to do some manual calculation in spreadsheet, and to verify whether the indicator calculate via pandas dataframe is correct.

Improvement

1. For different stocks try some different indicators, e.g. currently I use 3 days and 14 days simple moving average as features for all stocks, may be 5 days and 20 days moving average have better performance some of the stocks.

2. Currently project is focusing on technical analysis, I think fundamental information can also be used to do the prediction, e.g. Price earning Ratios, information on balance sheet, incoming statements and cash flow statements.