Stock Market Prediction

Frederick Fan, Oscar Ortega, Benson Yuan, Zhanyuan Zhang

Abstract

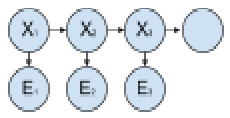
This report shows three approaches of analyzing stock price of a given company: Hidden Markov Model, Time Series analysis in time domain, and Logistic Regression/Feature Engineering. The report consists of three portions corresponding to these three approaches. Each portion includes the technical details of analysis, the resulting model, and the evaluation of it.

Dataset

The datasets applied here come from Quandl. This report focuses on analyzing the stocks of technology companies including Google, Facebook, and Amazon.

Model 1: Hidden Markov Model [1]

In a regular Markov Model, all the states are visible and the only modelling uncertainty lies in the transition probability, while in a Hidden Markov Model(HMM), the states are not directly observable, and we can only infer each state X_t at a particular time t by the evidence variable E_t, which is dependent on X_t. The HMM can be specified by the initial distribution π and transition probabilities P($X_t | X_{t-1,} E_{t-1}$) among the states. Since we do not have these parameters, we will let the hmmlearn package to fit our data specifically. We can then play around with the fitted model to find interesting pattern for stock prediction.



HMM assumes that transition rates satisfy the Markov property where the future is conditionally independent of the past via the current state. However, this may or may not be advantageous to us, given the high volatility and seasonal dependency in financial data (we will find ways to tackle this later). Regardless, it is still an intuitive choice to use HMM to model financial data since it has historically gained its fame in successful analysis of sequenced phenomena and it is easy to see that the real life behavior of a day to day market closely resembles that of a vanilla HMM. In the market, the values of stocks are driven by company decisions, earnings reports, the overall economy trend, psychology of the investors, all of which are too difficult to ascertain or quantify. However, there are market indices such as the S&P 500, considered by many to be the most representative indicator of the U.S. stock market, which are available for investors to have a sense of the health of the U.S. economy. Similarly, in our HMM, we can use the states X_t to capture the state of a stock on day t and predict the next state of the stock X_{t+1} based on E_t.

To simulate the HMM, we will use the hmmlearn package. Given a HMM, we can immediately answer some interesting questions such as:

- Given the model parameters and observed data, estimate the optimal sequence of hidden states. (Solved by Viterbi, a DP algorithm).
- Given the model parameters and observed data, calculate the likelihood of the data. (Forward-Backward algorithm)
- Given just the observed data, estimate the model parameters. (Iterative Expectation-Maximization (EM) algorithm)

We will make use of most of these algorithms in our prediction.

1.1 .1 Methodology (model initialization):

Before we start predicting daily stock values into the future, we need to first develop a model and then evaluate the accuracy of it by splitting our available data into training set and test set. The training data will be all data we have of our particular stock so far, and the test set will be the remaining 365 days of stock values which can help us evaluate the validity of our model. We will first start with the Google stock since it has about 10 years of training data and it is relatively linear, a more grounded task as we develop our initial model. Instead of setting a fixed number of states, we will experiment with different initialization to fit the chain and select the one with the highest score since EM is a non-convex, gradient based method, which is prone to getting stuck in a local optima. We will use the Bayesian Information Criterion(BIC), which takes the score of the model outputted by the hmmlearn package as a parameter, to help us decide which model setting is the most optimal. We chose BIC over AIC, another popular criterion, because BIC has a heavier regularization term to prevent overfitting which almost always prefers more complex model. We then pick the number of state corresponding to the lowest BIC index (lower is better).

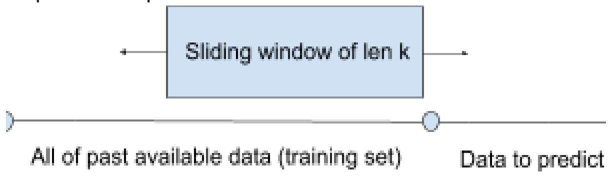
1.1.2 Model parameters:

- num_days: 365, this is how many days we are predicting.
- k : 50, this define the number of days of our sliding window.
- E_t : evidence or observation variables will be a vector of adjusted open, high, low and close stock value for Google.

Parameters are chosen with considerations of computational efficiency and tuned through cross validation.

1.1.3 Methodology (prediction)

After fitting as above, we will start the prediction phase. To predict the stock value of the next day, we start with the first day to predict and calculate the log-likelihood of the sequence ending at that day starting from k days back. We then shift this window backward in time until we find one which is most similar with the current sequence in question.



We then sum the change from that day to its next day with our current observation to arrive at our prediction for the next day.

$$E_{t+1} = (E_{t-j+1} - E_{t-j}) + E_t$$

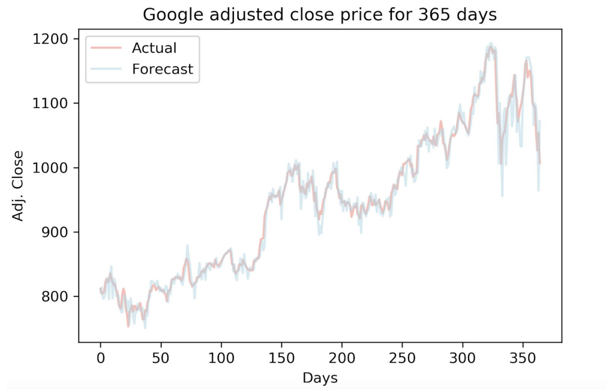
This is essentially maximum likelihood estimation(MLE) applied on the HMM setting. We first let time pass for a day (time lapse update) then make our prediction, then finally perform the observation update which helps us retune our models. The way we iteratively predict, perform the time lapse update, observation update and repeat is essentially what the forward algorithm entails. Interestingly enough, this sounds like what a stock exchange broker does on a daily basis as well!

1.2 Outcome

We will use the mean absolute percentage error(MAPE), a popular measure in statistics for forecasting, to evaluate our forecasted values against the actual data.

$$\mathrm{M} = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

MAPE for Google	Adj. Open	Adj. High	Adj. Low	Adj. Close
	0.013765	0.0113438	0.0136803	0.013060



Seeing that the errors are reasonably low, we can now utilize this model to predict into the future.

1.3 Conclusion

It was quite difficult to get started on this modeling process since training and predicting the hmmlearn package is not only a time consuming task but also a nebulous one. It was also not easy to wrap my mind around the state space and what exactly does each of them represent. Even though it is possible to extend this to predict more than one day in advance potentially by scaling each time index t from a day to say a week, we think it is still most natural to only have the HMM predict a day at a time since we can take more advantage of our iterative re-tuning method. I believe that this method is good at capturing fluctuations and seasonal pattern of the data since it works by mirroring the behavior of past data, which includes some quite unpredictable movements as well. A shortcoming of this prediction method is that since this result depends on fitting the data, convergence of the model, and greedily exhausting all past data to find the most similar match, it will not work well against all 500 stocks in the S&P 500 indices unless we throw it on cloud computing instances. Another interesting idea to try with multiple stocks' predictions would be to encode more information in our observation states such as some combination of ingenious features selected from feature engineering.

Acknowledgement/reference:
Nguyet Nguyen, Hidden Markov Model for Stock Trading, International Journal of Financial Studies.
https://github.com/ayushjain1594/Stock-Forecasting

Model 2: SARIMA Model

2.1 Introduction

This SARIMA model gives a long-term (next month) prediction of the adjusted closing price of a stock .
A process is a SARIMA $(p,\ d,\ q)\mathrm{x}(P,\ D,\ Q)_s$ process if it satisfies the following difference equation:

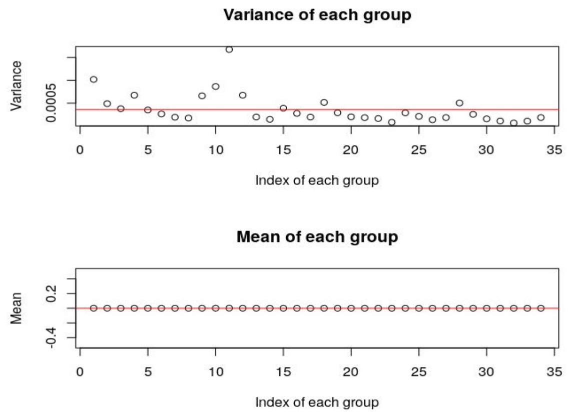
$$\Phi(B^s)\varphi(B)\nabla_s^{\,D}\nabla^d X_t = \delta + \Theta(B^s)\theta(B)Z_t$$

Where B is backshift notation, and $BX_t = X_{t-1}$. For more explanation of backshift notation, please check [http://ani.stat.fsu.edu/~huffer/mordor/timeseries/test1_material/13_backshift.pdf].
Based on time series analysis, this model is trying to capture the pattern and dependency structure over time. Patterns may include trending, seasonality, and dependency among noise and data itself. As for dependency structure, in SARIMA models, dependency moving average (MA) process models the dependency among noise, while auto-regression (AR) process models among data itself. Seasonality is modeled by the the seasonal terms in the model. To keep the model simple, here we restrict the maximal order of all type of processes (AR, MA, and the corresponding seasonal processes of these two, so totally 4 processes) to be 3. More parameters can be added if necessary. A model is good when its residual is closed to white noise.

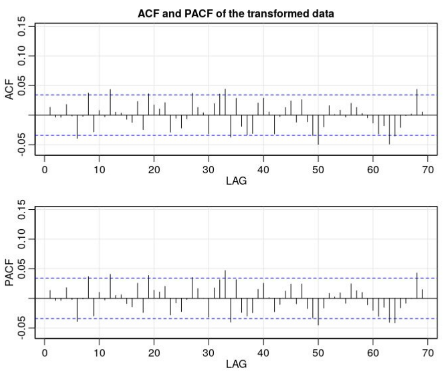
2.2 Exploratory Data Analysis

Before finding a SARIMA model for a time series, one needs to make sure that the dependency structures will not change as time progresses (a.k.a., the time series should be stationary); otherwise, it won't make sense to fit a model. Furthermore, SARIMA is designed for a stationary process. In practise, weak stationary is

enough, which can be approximate by a process with zero-mean and constant variance.
For this data, looks like taking a log-transformation and then taking a first order differencing with lag 1 will stabilize the time series. The following plot displays the variance and mean of the each part of the time series, where each part is one of the even 35 partitions. One can see that the transformed data has almost zero-mean and constant variance.



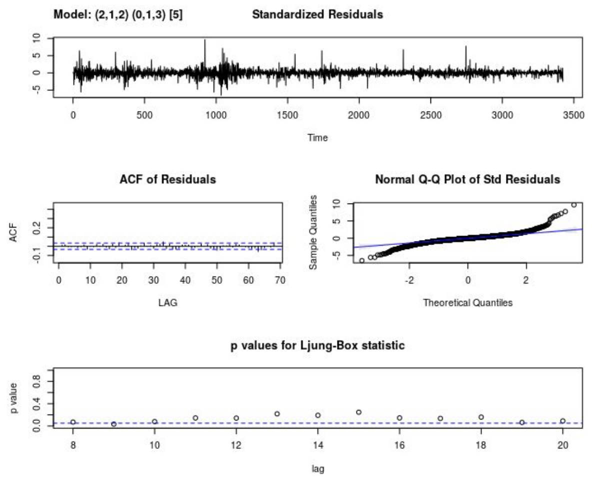
The sample autocorrelation plot (ACF) and sample partial autocorrelation plot (PACF) give us an idea of what kinds of dependency structures behind the process. It can be shown that under some general conditions (AR is causal and MA is invertible), all the coefficients go to zero exponentially fast. Also, an AR process can convert to a MA process with infinity parameters, while MA can also convert to an infinity AR process. Since white noise has zero-correlation, it makes sense that an ACF of a MA process will cut off after some lag. Partial autocorrelation at some lag defined as the coefficient of the best linear predictor of a data point at that lag. Hence, it makes sense that the PACF of an AR process cuts off after some lag. These phenomena can be generalized to a process with seasonality, except that ACF of a seasonal MA (SMA) cuts off at every seasonal lag while PACF of a seasonal AR (SAR) cuts off at every seasonal lag. However, if a process has both AR and MA, then both of its ACF and PACF taper off instead of cut off since it can convert to either infinity AR or infinity MA. The following plot shows the ACF and PACF of the transformed data:



We can see that both ACF and PACF have spike roughly every 5 lags. None of these plot has clear cutting-off pattern. Therefore, we can conclude that the time series consists of AR, MA, SAR, and SMA processes with period 5.

2.3 Model selection

Based on the restrictions and conclusions above we have potentially 256 models.The strategy of model selection applied here is that first we use Criteria Based Selection. The criteria we consider here is AIC. The reason is that although BIC and AICc are good candidate, we've already restrict the number of parameters of our model. So we do not need to give priority to simple model, thus BIC is not desirable here. Also, we have great amount of data, so AICc is closed to AIC. Next, perform cross-validation on these 4 models to pick the best-fitted one. To do cross-validation, here we first start from training the models on the data of year 2007, and use the model to predict year 2008's data. This is considered as one iteration. The next iteration will include all the data from 2007 to 2008 then make prediction on 2009. Doing so for 11 iterations. The cross-validation score of a model is calculated by taking the mean of its residual sum of square of all the 11 iterations.
The AIC gives us the following model in the format of (order of AR, order of differencing, order of MA)(order of SAR, order of seasonal differencing, order of SMA)[Period]: (2, 1, 2)(0, 1, 3)[5], (0, 1, 0)(1, 1, 5)[5], (0, 1, 1)(0, 1, 1)[5], and (1, 1, 0)(0, 1, 1)[5].
The cross-validation scores of these four model is: 2091516, 2106714, 2100623, and 2100715. Since the first model has the least cross-validation score, we pick (2, 1, 2)(0, 1, 3)[5] to do prediction.
To further make sure that we find a good model, check some diagnostic plots of the model:

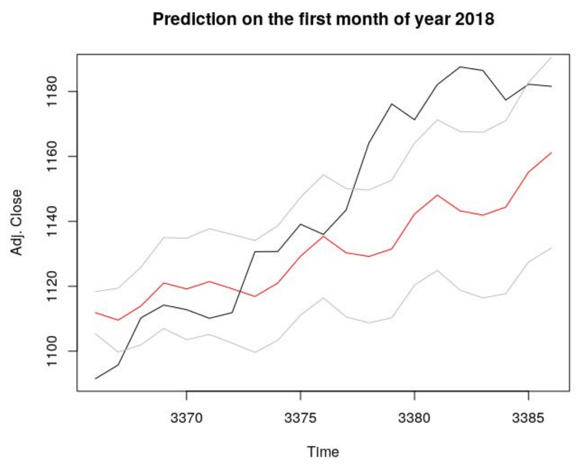


The plot shows that the ACF of residuals are all within the blue bound, which is 95% bound for the hypothesis test of whether the residual is white noise. Also, the Ljung-Box tests from lag 0 to lag 20 shows that most of the the p-value are above 0.05 except one of them at lag 5 (this is fine since we expect 20*0.05 = 1 out of 20 is out of the bound), which gives us confidence that the residuals are independent. The Normal Q-Q plot tells us that compared with normal distribution, the residual has more extreme values. This may due to that the variance of the time series is not strictly constant.

2.4 Model Fitting and Prediction

Now that we know what kind of SARIMA model should we use, the next step is to estimate the parameters and do prediction. Estimating the parameters by Yule-Walker method gives ar1 = -0.0315, ar2 = -0.9583, ma1 = 0.0368, ma2 = 0.9390, sma1 = -10.0053, sma3 = -0.0272.

We use this model to perform prediction on the first month of 2018 and the result is as following, where the black line is the true value, the red line is the prediction, and the gray lines are the standard error of our prediction.



As the plot shows, the prediction can capture the pattern of the adjusted closing price for the first half of the month, but its standard error gets larger and it becomes unreliable as time goes.

2.5 Conclusion

This SARIMA model turns out to be not accurate enough to make a long-term prediction. There are some potential sources for the inaccuracy of this model. First or all, the data transformation does not stabilize the time series. Even though we see that the transformed data has almost zero-mean and constant variance, whether it is stational still requires further investigation. After all, stationary implies zero-mean and constant variance, but not vice versa. Also, more parameters may increase the accuracy of the model. Furthermore, the period we choose may not be accurate. Fitting a time series model and doing relevant hypothesis test involves making several important assumptions about the time series (stationarity process, Gaussian noise ect.). To improve the model we needs to carefully make sure that these assumptions are not violated by some unknown factors even before model selection. Necessary data cleaning and modeling the correlation between adjusted closing price and other variables will also increase the accuracy of the model.

Model 3: Logistic Regression / Feature Engineering:

3.1 Introduction:

Typically, in a model where one uses logistic regression, the objectives are to first perform feature engineering, the process of creating and adding new features to add complexity to our models, in this case based on the data provided using popular economic metrics of market activity, and then perform maximization of the softmax loss by performing logistic regression, which allows one to then perform classification on a dataset by choosing the class which generates the maximal probability of being part of the class given the features.

In other words, we choose $argmax_x P_x = P(x \mid f_1, ..., f_n)$ where x is the class and $f_1, ..., f_n$ are the features of the data.

3.2 Methodology:

This feature engineering/logistic model rather than try to predict closing price for each day, seeks to predict whether a given day for a particular company is either a ‘gain’ or a ‘loss’, where we define a ‘gain’ as a day where the closing price of the current day has greater value than the closing price of the previous day, and the ‘loss’ as a day where the closing price is less than closing price of the previous day.

3.3 Model Parameters:

N := Number of days we try to predict ahead into the future

3.4 Metrics/Features Chosen:

30 day moving variance:

For a given company X day on day t, we compute the 30 day moving variance X_t, by computing the variance of the closing price for days $t, t-1, ..., t-30$

N day percent change:

We compute the percent change $X_t = (C_t - C_{t-30}) / C_t$ where C_t is the closing price for a company X on day t

N day Chaikin Money Flow:

The CMF is an economic indicator that measures the Money Flow volume over a set period of time. We first compute The money flow volume for a company X on a day t, $X_t = (C_t - L_t) - (H_t - C_t) / (H_t - L_t) * V_t$ where the variables $C_t\ H_t\ L_t\ V_t$ are the Close, High, Low and volume for the company on a given day. We then add the money flow volume over the previous N days and divide the volume over the previous N days to compute the value CMF_t. The CMF can be interpreted as a random variable on support [-1,1], and a basic interpretation of the CMF is that when the CMF on a given day is closer to 1, there is more pressure for the company to buy, and that when the CMF for the company is closer to -1, the pressure to sell.

N day Relative Strength Index:
The RSI is a indicator that measures the size of recent price changes to estimate whether a stock is being over or undersold based on the current stock price. For a company, the relative strength index on day t X_t = 100 - (100 / (GLR_n)) where GLR_n is the n day gain loss ratio which is computed by first adding all the gains of days where there are gains, diving that quantity by the number of days, and then dividing that quantity by the n day loss, which is computed by adding all the losses of days where there are losses and dividing by the number of days. A traditional interpretation of the relative strength index is that when the RSI is above 70 that the stock is currently overvalued, and that when the RSI is below 30, that the stock is currently undervalued.

N day Williams R indicator:
The Williams R Indicator is a momentum indicator, that like the RSI, attempts to measure whether a stock is either being currently overbought or oversold. We can compute a company's Williams R indicator value for a day t X_t by first finding the highest high of the stock price over the previous N days, finding the difference between the highest high and lowest low of the stock price over the previous N days and dividing that quantity Q_t, by $(-100 * Q_t)$. A traditional view of the indicator is that when its value is above -20 the company's stock is currently being overbought, and that when the indicator is below -80, the stock is currently being oversold.

Aggregation based methods:
In this model, we then computed the mean percent change of companies grouped by their corresponding business sector for 3, 7, and 30 days respectively and performed the same computation for the top three weighted and unweighted companies in the S&P500.

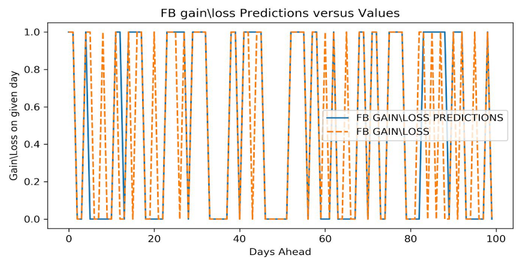
3.5 Results:
We will measure the accuracy of our model as the percentage of correct classifications over the number of days we lookahead in our model. Recall that our classes are 'gain' and 'loss'.

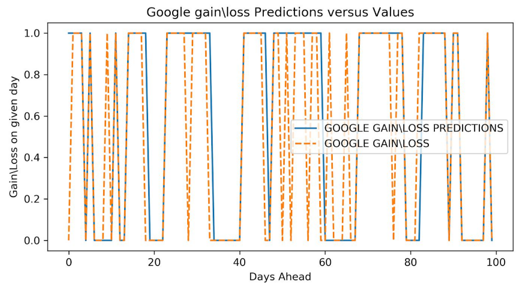
In this model, we selected, Google, Amazon, and Facebook as indicators for the models performance and we set the number of days of lookahead N = 100 .

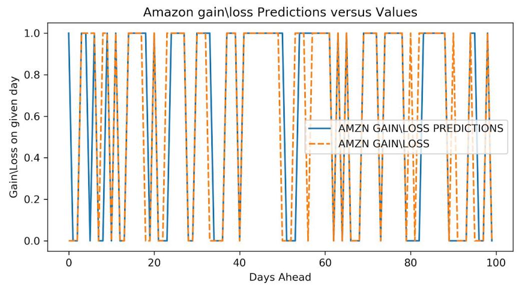
Accuracy:

Amazon	Facebook	Google
.87	.85	.91

3.5.1 Gain Loss Predictions versus Gain/Loss of Company:







3.6 Conclusions:
With this data in mind, one can see that with the use of the given features, one can approximate gain loss patterns for a given stock with relatively high accuracy. Although this model doesn't predict the values of a given stock, the conditional probabilities generated based on the set of features, coupled with other techniques such as a hidden markov model we first discussed can be used to better predict stock prices.

3.7 Acknowledgements/References:

https://www.investopedia.com/terms

https://arxiv.org/ftp/arxiv/papers/1603/1603.00751.pdf