Mechanical & Industrial Engineering
UNIVERSITY OF TORONTO

MIE 1628   Big Data Science
Project Report

Apple Stock Price Prediction: Time Series Analysis

Team 9

Bowen Xu   1006411786

Acknowledgement

I take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project. I would like to show our greatest appreciation to our mentor Professor Yanina Shevchenko. We thank her for her support and help. The guidance and support received from all the team members was vital for the success of the project.

Problem Statement

In the past decades, investors in the stock market do not properly understand which stocks to buy or which stocks to sell in order to gain more profits due to lack of awareness of the stock market behaviors. In today's world, intelligent investors use historical data or time series analysis to build machine learning models to predict the stock market behaviors. This will allow investors to foresee the behavior of the stock that they are interested in and thus act accordingly.

The main input to this project will be historical data from Apple Stock Prices downloaded from Yahoo Finance. Appropriate external data resources and features engineering would be applied to help the prediction models be more accurate. Measure of performance will be based on varied horizons such as one day, one week, two weeks, one month and 4 months. The entire system would be built and implemented by PySpark API in Databricks clusters.

Literature Review

Recently, many interesting work has been done in the area of applying machine learnings for analyzing price patterns and predicting stock prices among most of technology companies. Most stock traders nowadays depend on Intelligent Trading Systems which help them automatically make trading decisions, submit orders and manage orders after submission. It is consisted of four conceptual components which are data, model , execution and monitor. It is worth to mention that, in the model section, models are constructed using different methodologies and techniques but the ultimate goal is to make inference or predict about the world (Reid 2015).

Furthermore, some systems use mathematical models and historical data to make predictions while some use financial articles and expert reviews as input to conduct sentiment analysis and predict the stock market. A wide range of machine learning algorithms are implemented in the both systems. However, because of the volatility of the stock market, no system has a perfect or accurate prediction. In this project, we will be using historical data as input to machine learning models to predict Apple's stock prices.

Data Preparation

<u>Datasets</u>

- *2014-2018 Apple Stock Price Datasets*: Original dataset starts from year 1980 which was the time where Apple just started to launch their company and the stock prices were very low compared to their today's prices. In this project, we will focus on the time period in which Apple is in their mature stage, i.e. from year 2014 to 2018 rather than start-up or growth stages.
- *S&P 500 Index*: A useful stock market index that measures the stock performance (market-capitalization-weighted) of 500 large companies in the U.S. and to be considered as the best representation of U.S. stock market. Apple, as the top company in U.S., S&P 500 index may represent the trend of their stock prices by some extent.
- *GBP to USD Exchange Rate*: According to Figure 1, apart from U.S. itself, Europe accounted almost 22% of Apple's sales is the largest market (Mattera 2018). We think the exchange rate in GBP may have effect on the Apple's stock prices as European people may facing different pricing from Apple due to the change in their purchasing power by some way.

- *RMB to USD Exchange Rate*: As the second largest market for Apple, China is still a rapid-growing market for them. In addition, most of Apple's factories are located in China, therefore, the exchange rate between these two countries may affect the costs of production and finally the stock prices.
- *Federal Rates*: The interest rates may also have effect on Apple's stock prices to some extent since it is directly related to the financial planning of Apple. They will have to decide when to borrow money from banks to minimize the interests paid and maximize their profit in long term.

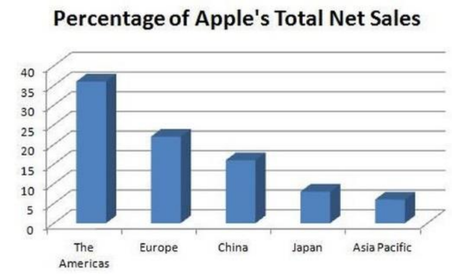


Figure 1. Percentage of Apple's Total Net Sales

Feature Engineering

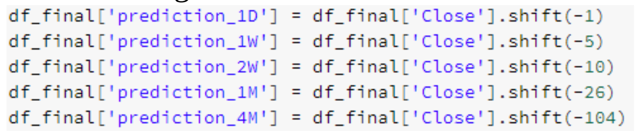
Apart from some external resources and useful datasets which may indicate some relationships with Apple's stock prices, more focus were put on the features engineering of the original dataset from Apple. Original dataset only has six features which are Close, High, Low, Volume, Open and Adj Close; therefore, some technical indicators related to time series analysis are researched from the internet and summarized below:

- *Simple Moving Average*: It is the average closing prices of a stock over a specified time period from the historical data. It can easily identify the current price trend and potential change in an established trend (Hayes 2019).
- *Exponential Moving Average*: It is very similar to the simple moving average, except that more weight is given to the latest data. It is more preferred since it is more reactive to the latest price changes than SMA and can make the results more timely.
- *Momentums*: It is a function of features of High, Low, Close and Volume that can indicate the ability of a market to sustain an increase or decrease in prices, in technical analysis, this indicator gives a signal in change of prices.
- *Moving Average Convergence Divergence (MACD)*: MACD is a trend-following momentum indicator that shows the relationship between two moving averages of prices. It is calculated by subtracting the 26-day EMA from 12-day EMA.
- *Z-score*: It is a numerical measurement used in statistics of a value's relationship to the mean of a group of values by using mean and standard deviations from the mean. It can measure the variability in prices and can be used to determine market volatility.
- *Lag*: Lag features over different horizons by 1, 5, 10, 26 and 104 days.
- *Return*: Percentage gains over different horizons, take one day as an example, lag Close for one day and take the difference between Close and Lag_Close to get the gain for one day period, then divide the gain by Lag_Close to get the percentage return.

- *Return Mean, std and Volume Mean, std*: Means and standard deviations of returns and volumes over different horizons.
- *Loss*: Differences of Close prices over different horizons, take one day as an example, subtract tomorrow's price from today's price.
- *Time Periods Indicator*: Day of week, quarter, month, year, days of year, days of month, week of year are time indicators that may help prediction models to be more accurate.

Target Variable

We used supervised machine learning method in this project, therefore, target variables or labels will be very important. Closing prices are the labels we are trying to focus on, however, for different time horizons, the column "Close" should be shifted by certain time periods in order to predict the prices in each period. Detailed transformation was implemented by using a simple method shift() in PySpark as shown in Figure 2 below.



```
df_final['prediction_1D'] = df_final['Close'].shift(-1)
df_final['prediction_1W'] = df_final['Close'].shift(-5)
df_final['prediction_2W'] = df_final['Close'].shift(-10)
df_final['prediction_1M'] = df_final['Close'].shift(-26)
df_final['prediction_4M'] = df_final['Close'].shift(-104)
```

Figure 2. Target Variable Transformation

Statistical Models

Statistical modelling uses mathematical equations or relationships between variables to perform predictions, the only feature used in statistical modelling is the "Close" column in the original Apple's dataset.

Moving Averages

Moving average is the simplest statistical model for forecasting the stock prices. There are two different types of moving average: simple moving averages and exponential weighted moving averages, in this project, we used the former one. For simple moving average forecasting, the value of y at time $t-1$ is calculated at time t by averaging the most recent m observations with equally weights, the equation is shown below. In this project, we took 2, 5, 10, 20 and 80 days as the time span to calculate different predictions. As shown in Figure 3, we can see the longer the time span, the less sensitive the predictions are, therefore, the prediction curve becomes more smoothed.

$$\hat{y}_t = \frac{y_{t-1} + y_{t-2} + \cdots + y_{t-m}}{m}$$

The main advantage of moving average technique is its ability to smooth out the predictions, i.e. it can filter out the noises in the random short-term price fluctuations. However, it is very slow to respond to rapid price changes which always occur at the market reversal points.

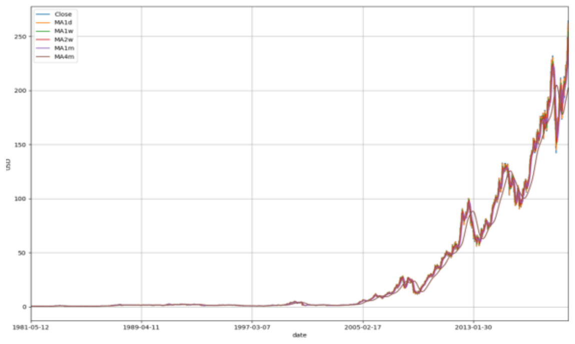


Figure 3. Predictions Curve for Different Time Spans

Autoregression

Autoregression model uses observations from previous time steps as input to predict future values with multiple lag variables. It models the output values based on a linear combination of previous prices, a p^{th} order autoregressive model is shown below:

$$y_t = c + \emptyset_1 y_{t-1} + \emptyset_2 y_{t-2} + \cdots + \emptyset_p y_{t-p} + \varepsilon_t = c + \sum_{i=1}^{p} \emptyset_i y_{t-i} + \varepsilon_t$$

where ε_t is white noise with normal distribution that can only change the scale of the time series but not the patterns. $\emptyset$ is the most important factor in the autoregressive model which measures how the observations in a time series are related to each other, firstly, the autocovariance at lag k and the autocorrelation coefficient $\emptyset$ at lag k are defined, respectively as follows:

$$\gamma_k = Cov(y_t, y_{t+k}) = (y_t - \mu)(y_{t+k} - \mu) \qquad \emptyset_k = \frac{\gamma_k}{\gamma_0}$$

where μ and $\gamma_0$is the mean and variance of y_t.

ARIMA

ARIMA is the combination of both autoregressive models and moving average models plus a "integrated" term. There are three components of ARIMA model: p indicates the number of autoregressive terms which can determine the pattern of growth or decline in the data, d is the number of non-seasonal differences which corresponds to the "integrated" part that can indicate the rate of change of growth or decline and q is the number of moving-average terms which is the noise between consecutive time points.

ARIMA is really good at time series forecasting since it can make the series stationary by differencing it. That is, subtract the previous value from the current value, sometimes more than one differencing will be needed when the series is complex. This job is done the "integrated" part.

Machine Learning Models

Machine learning models take multivariate features as input and use different algorithms to perform predictions. In this project linear regression used a different set of features from other two machine learning models (decision tree and XGBoost) which used all of the features as the input to the model as shown in Figure 5. All three models used 90% of original data as training data and 10% for test data.

Linear Regression

Linear regression is to predict a scalar value y from variables x when there is assumes to be a linear relationship between x and y. Mathematically,

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \sum_{j=0}^{n} \theta_j x_j$$

where θ is the model coefficients which will be learned in the training process.

$$objective\ function = \frac{1}{2}\sum_{i=1}^{n}(\theta_i x_i - y_i)^2 + \frac{1}{2}\sum_{j=1}^{k} w_j^2$$

The objective function for linear regression is to minimize the squared error losses between targets and predictions. It is worth to mention that linear regression is very easy to overfit when there are too many features in the input variable, careful selection of features is needed in order to get a good results or small generalization error in the test data. Furthermore, regularization needs to be added to the model to reduce overfitting problem.

Decision Trees

Regression trees can take multiple variables as input to predict the continuous targets, in this project, we used all of the features as input to the model. For a dataset has a lot of features, regression trees can divide the data space to many non-overlapping sub-space recursively, i.e. it uses tree to represent the recursive partitions over the entire data space which can make predictions really fast. For observations in the same sub-space, the predictions are the same. Regression trees are very easy to overfit when the depth of the tree is very deep, in this project, we carefully tuned the hyperparameters to reduce the overfitting to let it generalize in the test dataset.

There are four criteria to decide where to do the splitting: gini coefficient, information gain, Chi-square and variance. In this project, we used variance as our criteria since variance is highly related to numerical data and we want lower variance in the predictions to reduce overfitting.

Regression trees are very easy to interpret and understand by people since its representation is very intuitive. They also required less data cleaning compared to other modelling techniques which means they are not influenced by outliers and missing values by a fair extent.

XGBoost

XGBoost is one of ensemble algorithms called “boosting” that can incrementally build models and each new model will focus on the training instances that previous models get wrong. Predictions of the final ensemble model is therefore the weighted sum of the predictions made by the previous tree models. Gradient boosting combines gradient descent and boosting algorithms together to minimize errors in sequential models and XGBoost makes gradient boosting even much stronger.

XGBoost is an optimized version of gradient boosting algorithm through parallel processing, tree pruning, handling missing values and regularization to avoid overfitting. The objective function for XGBoost has two terms: first one is the loss function (always mean squared error) and other one is the regularization term that penalizes complex models through both L1 and L2 regularization to prevent overfitting.

$$objective\ function = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \sum_{j=1}^{K} \Omega(f_j)$$

Metrics

To evaluate the performance of all six models, we used symmetric mean absolute percentage error. SMAPE is the preferred metric for time series analysis because it is scale-independent and the results are normalized. Furthermore, root mean squared error is used during the hyperparameter tuning process to evaluate the performance for different sets of hyperparameter combinations, however, it depends on the scales of the data which means it is not a suitable metric for evaluating the final result for time series analysis.

$$SMAPE = \frac{100\%}{n} \sum_{t=1}^{n} \frac{|F_t - A_t|}{(|A_t| + |F_t|)/2} \qquad RMSE = \sqrt{\frac{1}{n} \sum_{t=1}^{n} (F_t - A_t)^2}$$

Results

Model	1 Day sMAPE	1 Week sMAPE	2 Weeks sMAPE	1 Month sMAPE	4 Months sMAPE
Moving Averages	2.22	4.91	5.82	9.35	20.41
Autoregression	2.16	4.66	6.70	11.41	24.81
ARIMA	1.22	3.75	10.68	13.15	17.20
Linear Regression	2.22	3.46	5.21	13.27	22.20
Decision Tree	4.72	5.86	7.65	10.79	15.31
XGBoost	1.95	5.23	7.27	10.69	15.45

Figure 4. SMAPE Results for All Models in Different Time Horizons

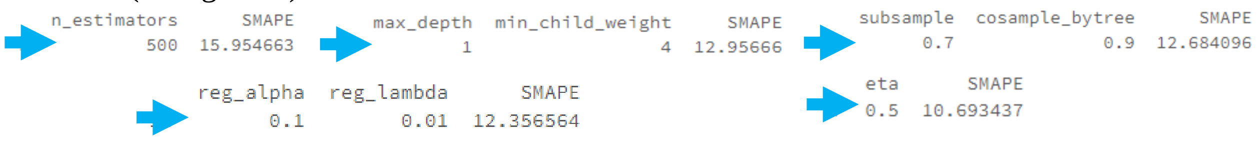
Model	Used Data	Feature Transformation	Train/Test Split	Optimzation
Moving Averages	Close	None	None	None
Autoregression	Close	None	None	None
ARIMA	Close	None	None	None
Linear Regression	High,Low,Open Returns, return_mean, std volume_mean, std	Volume,Adj Close	90% Train, 10% Test	elasticNetParam egularization,fitIntercept
				maxBins, maxDepth
Decision Tree	All		90% Train, 10% Test	minInfoGain
XGBoost	All		90% Train, 10% Test	n_estimator, max_depth, n_child_weight, subsample cosample_bytree, reg_alpha reg_lambda, learning_rate

Figure 5. Summary for All Models

As we can see from the SMAPE results for all models, all results are getting worse when the time horizons are longer. ARIMA, decision tree and XGBoost achieved better results among all other models, however, classical statistical models such as moving averages and autoregression did a good work in short-term predictions compared to machine learning models. When the time horizons increase to one month and 4 months, we can see machine learning models are getting much better than statistical models.

Optimizations

All machine learning models were tuned carefully to reduce overfitting for the test dataset, especially for decision tree and XGBoost. For linear regression and decision tree, they used the same method to get the best hyperparameters: in PySpark API, we used ParamGridBuilder to construct parameter grid for different hyperparameters (all hyperparameters tuned are listed in Figure 5), then used trainValidationSplit to split inputs into training and validation data and iterate all grids for hyperparameter to find the best combination. For XGBoost, we used a different way to tune the hyperparameter since PySpark API does not have the built-in method for XGBoost and there are many of hyperparameters. Therefore, we used for-loops to tune them sequentially, as shown in Figure 6. For each hyperparameter, we were to find the best value to get the smallest SMAPE (Tseng 2018).



```
n_estimators      SMAPE
         500  15.954663

max_depth  min_child_weight      SMAPE
        1                 4   12.95666

subsample  cosample_bytree      SMAPE
      0.7              0.9  12.684096

reg_alpha  reg_lambda      SMAPE
      0.1        0.01  12.356564

eta       SMAPE
0.5   10.693437
```

Figure 6. Tuning Process for XGBoost (one month)

Feature Importance

Model	1 Day	1 Week	2 Weeks	1 Month	4 Months
Linear Regression	Return_1D	Return_1D	Return_1D	Return_1D	Return_1D
	Return_mean_4M	Return_mean_2W	Return_mean_2W	Return_mean_4M	Return_mean_4M
	Return_mean_2W	Return_mean_1M	Return_std_2W	Retun_std_4M	Return_std_4M
Decision Tree	Retun_mean_2W	SP500	SP500	Loss_1M	Volume_mean_4M
XGBoost	Loss_1D	Loss_1W	Loss 2W	Loss_1M	Loss_4M
	pct_change_4M	Loss_2W	sma_1D	Adj Close	Return_mean_4M
	Return_std_4M	Adj Close	pct_change_2W	Return_std_1M	Volume_mean_4M

Figure 7. Feature Importance for Machine Learning Models

We can see above table, all three machine learning models depend on the return_mean, return_std and loss features, this makes sense since mean, standard deviation and losses are all indicate the changes in prices at different time horizons. Time series data depends on time which makes it non-stationary, the trends can result in a varying mean over time and seasonality can result in a changing variance over time. These three features can provide the models stationary patterns about the data which mean they can actually help the model learn well in time series dataset.

Recommendations

In this project, we used external datasets and features engineering to make our predictions more accurate, however, these features are all numerical or quantitative. We thought we could have added some categorical features such as expert reviews or financial articles related to Apple's stock prices, this may help us enhance our models in a more qualitative way. However, not all models accept categorical data as input such as linear regression, therefore, more feature transformations are needed such as tokenizer to transform categorical data into numerical data. Decision tree and XGBoost can take advantage of taking any data types as input to save some work to do. Furthermore, some internal resources from Apple's company may also help us engineer our features such as debt and account receivable which closing prices are highly dependent on, however, it was very hard for us to find any information in this area.

Bibliography

Hayes, Adam. 2019. *Simple Moving Average (SMA).* 4 14. Accessed 12 1, 2019. https://www.investopedia.com/terms/s/sma.asp.

Mattera, Sam. 2018. *The 5 Largest Markets for Apple.* 10 8. Accessed 12 1, 2019. https://www.fool.com/investing/general/2015/04/01/5-largest-markets-for-apple.aspx.

Reid, Stuart. 2015. *Intelligent Algorithmic Trading Systems.* 9 26. Accessed 12 1, 2019. http://www.turingfinance.com/dissecting-algorithmic-trading/.

Tseng, Gabriel. 2018. *Gradient Boosting and XGBoost.* 4 13. Accessed 12 2, 2019. https://medium.com/@gabrieltseng/gradient-boosting-and-xgboost-c306c1bcfaf5.

Appendix

Moving Average
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/8809909644818878/2601936400720127/8042632490685997/latest.html

Autoregression
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3566472574954842/312731291627753/2530627590817800/latest.html

ARIMA
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3514124519708953/1150248169258661/1708052135813957/latest.html

Linear Regression
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/6701749513620212/613360282979738/8050944378397175/latest.html

Decision Tree
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3514124519708953/3299985615969072/1708052135813957/latest.html

XGBoost
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/6328523600086529/2287063822595265/804232932623426/latest.html