Capstone Proposal
Collin Hazlet
July 25, 2017

Predicting Future Stock Prices

Domain Background

Ever since the Stock Market was created, people have been trying to make money by predicting the future price of a stock. Therefore, it is imperative that stock traders utilize accurate predictive models when placing trades.

The Efficient Market Hypothesis is an investment theory that states it is impossible to “beat the market” because stock market efficiency causes existing share prices to always incorporate and reflect all relevant information. There are multiple forms of this theory which assign tiers of efficiency and allow for some deviation but generally it asserts that a person attempting to beat the market would do no better than a person holding a diversified portfolio of stocks.

However, this hypothesis is contradicted by the profits of investment funds, banks, and traders who make a living buying and selling stocks according to different financial models. Some of these models are based on technical analysis of stock charts while others adhere to fundamental principles of company valuation.

Udacity currently has a course on Machine Learning for Trading which outlines how to build datasets and apply models. These two sites provide historical and future projects in finance using machine learning:

1. www.stat.wharton.upenn.edu/~steele/Courses/9xx/Resources/MLFinancialApplications/MLFinance.html
2. www.techemergence.com/machine-learning-in-finance-applications

The website, www.pythonprogramming.net, also contains videos and code for pulling financial data and creating dynamic charts.

For this project, I will build a stock price predictor which can predict a stock’s future price to profit from trends in the data. The problem will not be approached with timeseries models but with standard regression techniques because I feel more comfortable with these models than timeseries ones such as ARIMA.

Problem Statement

The goal is to build a stock price predictor that can take the daily adj. close prices from all stocks in the S&P500 over a certain date range as inputs, and outputs projected estimates of a selected stock for given dates. Although there are many stock prices in a day, the model will only need to predict the adjusted closing price.

To generate future stock prices the model will complete the following tasks:

I. Build a data set or query data from a financial API
II. Implement a user interface which allows the user to choose which stock to predict and how far into the future to predict

III. Select relevant features for the estimation model
IV. Train a prediction model over the data set
V. Plot the predictions for the user to see and gain information for stock selection
VI. Give the user information concerning the reliability of the model by scoring the model and comparing it to a benchmark model

Datasets and Inputs

The dataset will be built by querying from the Quandl financial API to select price information. According to the Quandl site, the API is easy to use and is fully documented (https://blog.quandl.com/getting-started-with-the-quandl-api). After watching a series of videos from pythonprogramming.net on financial data analysis, I will be able to re-purpose an existing querying script to pull in data over a list of stock tickers and a date range specified. For this project, I will be pulling historical price data for all stocks currently in the S&P500. The time frame will be from 2000-01-01 to 2016-12-31 which amounts to 4,032 trading days so the data set will contain 4,032 rows. The columns of the dataset will be the Adj. Close of each stock so that there will be a total of 505 columns. (update: two stocks were omitted due to incorrect formatting of ticker from Wikipedia into the Quandl API so there will only be 503 total columns)

After going through the Machine Learning for Finance Udacity videos, 3 calculated fields of moving averages will also be added into the data set. These will be the 10, 20, and 30 day moving averages for the stock being predicted.

The [Date] field will be transformed from a date variable into a [Day_N] field which will essentially be an index variable which can be used to track time as a discrete numerical variable.

The inputs of the script will be the {ticker} which will tell the program which stock to predict, the number of days into the future to predict, and if possible, the date range over which to train the model.

The target variable will be the stock price column designated by the user from the {ticker} variable and will be shifted up by the number of days the user wants to predict.

Solution Statement

Before any models are trained, I will be performing PCA and Recursive Feature Elimination for feature selection to narrow down the number of variables. I believe that I can automate these procedures so that the user can choose any stock within the set and the model will choose the correct features to use.

Because this course did not cover time-series data forecasting, and because I feel more familiar with standard estimation techniques, this problem will be approached using linear models such as LASSO and Stochastic Gradient Decent Regression and other estimators covered in this course such as the SVM Regressor, KNR, and Random Forest Regressor. The models will be individually optimized through cross-validation and grid search methods.

I will be using many different types of models for two reasons: The first is to better understand the models through personal usage. The second is to enable the creation of an ensemble model at the end of the process. Ensemble modeling is very helpful when considering prediction problems because it can help counteract any biases within each individual model. All models will be judged according to their score and against the benchmark model to give the user useful information.

Benchmark Model

The benchmark model for this problem will be a univariate linear regression with the current price of the stock as the single variable to predict the future price of the stock. This was chosen because it is the simplest model I could think of that could be useful. This baseline model would be able to find the general trend of the stock's current price with the future price.

Evaluation Metric

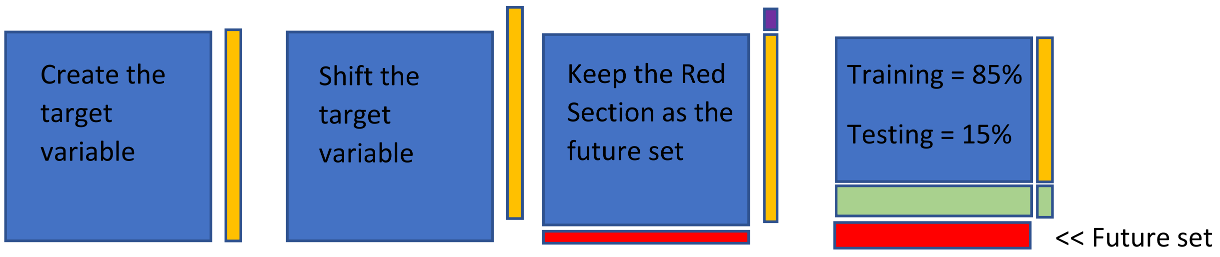
The learned model will be judged against this benchmark using the R-squared metric as this is the most common metric to score regression models. This metric is already the base metric in sklearn for scoring the models so the ".score(X,y)" method will be implemented.

In addition to R-squared, I will also look at the mean absolute error so the user can easily understand how to interpret the results of the model in terms of US dollars.

Because this program will have the capability to predict any stock price any number of days into the future, the score of the model will greatly depend on the inputs given by the user. For this reason, I cannot give an acceptable desired score.

Project Design

After the dataset is assembled, I will need to clean the data of any null values. The target variable will be created by selecting the stock price based upon the user's input and shifting the entire field up the number of days that the user wants to predict. For example, if the user selects 'MMM', a copy of this stock price will be designated as the target variable, then the entire target variable will be shifted up the number of days also designated by the user, say 7. This final set will be broken into training and testing sets based on an 85 – 15% split rate. Due to the shifting of the target variable, a third set will contain N number of rows depending on what the user defines as his future price to predict. These last rows will not allow for verification of our predictions so they will be kept by themselves as a "future" prediction set. Below is a visual representation of the process.


Create the target variable
Shift the target variable
Keep the Red Section as the future set
Training = 85%
Testing = 15%
<< Future set

The dataset will then be passed through multiple models to select the best features. The first model chosen for this process will be PCA because it can find new dimensions that a human could not find on their own. Second will be Recursive Feature Elimination, because it will be able to cycle through the given variables and find the best field while excluding unnecessary ones. These two datasets will double the number of predictions possible which will hopefully add to our end ensemble model.

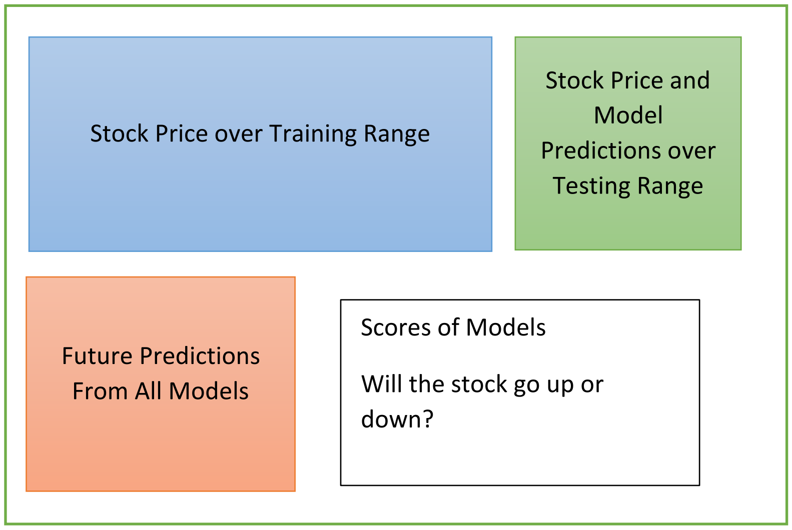
The general learning process will proceed as follows:

1. A function for training a specific model is defined
2. A dataset is passed to the training function
3. TimeSeriesSplit() splits the data into 10 different subsets on which to perform cross-validation
4. The learning algorithm is defined
5. A dictionary of relevant parameters and their values is defined
6. The regressor, parameters, and cross-validation set are passed to a GridSearchCV() function
7. The training and target sets are passed into the grid
8. The best estimator is returned

The above process will be applied to several regression estimators (Lasso, SGD, KNN, SVM, and Random Forest) thus creating multiple predictions over which to ensemble. Several different processes will be tried as ensemble models. The first ensemble method will be another linear regression that will take the predictions of the models as inputs and try to predict the target price. The next two models will be SVM and KNN in a similar manner and the last way will be a simple average of the models' predictions.

Once all models are built and optimized, scores will be taken for each one and compared to the benchmark. These scores will be presented to the user along with graphs of the training, testing, and future predictions.

Below is the planned outcome of the script which will be presented to the user upon completion of all training and testing of the models.


Stock Price over Training Range
Stock Price and Model Predictions over Testing Range
Future Predictions From All Models
Scores of Models
Will the stock go up or down?