Final Paper - Financial Modeling

Introduction

When tasked with predicting stock prices, we decided it would be important to consider current methods being used by professionals in the field of algorithmic trading as well as research being done in this area. After our preliminary research was complete, we converged on specific methods to featurize our data and specific models we would try to implement in order to predict the stock prices of the top 200 technology companies on Nasdaq for a variable amount of days into the future. We mainly consider extending ML techniques: first moving from simple linear regression to SVR's, and then exploring Ridge Regression, Lasso Regression, Elastic Net Regression,and Random Forests, all of which are supported in literature.

We believe that when predicting next-day prices, using broad market and industry indicator features (in addition to single-stock features) will improve upon the performance achieved using single-stock features. Intuitively, this makes sense; in the short run, such as next-day performance, general market outlook may weigh heavily on investors' appetite for new investments. We do not expect the variance of these results to differ significantly from single stock prediction models due to the stock market being heavily dependent on industry performance as well as individual company performance. In terms of noise, industry indicators such as competitors' performance or other sector-level indicators could lead to more noise in the results because of the additional, possibly insignificant data.

Dataset

For this project, we chose to use data provided by Quandl[1] for stock price data such as the adjusted prices of specific stocks. We first gathered a list of stock tickers from Nasdaq's official dataset[2] of stocks split by industry. We decided to focus on stocks in the Technology industry due to their reliable trends and present-day relevance. We picked the top 200 from this list and used the tickers to get the stock price data from Quandl. We cleaned the data in a similar fashion to the starter file provided for this project in order to only keep the Adjusted open, close, high and low prices of a one day period in each stock's data frame. We also calculate the percent change in price from the open and close prices per day and store in a separate column of data. We then store each clean data frame in a dictionary mapped by stock ticker.

Featurization - Single Stock

Initially, our dataset only had data on the prices on each day over the common time frame we worked with. In order to make meaningful decisions from this data, we decided to featurize the data with specific, industry-recognized stock and industry level indicators of future stock price, most of which are outlined in our initial project proposal. We looked at three main types of indicators: volatility, momentum, and trend. We chose these based on their prevalent usage in firms and research

[1] https://www.quandl.com/

[2] https://www.nasdaq.com/screening/companies-by-industry.aspx?industry=Technology

around the world. After selecting features important to us, we used the Python library ta (technical analysis)[3] to assist us in calculating these features.

For volatility and trend indicators, we chose to use, respectively, the average true range (ATR) and average directional movement index (ADX) because of their common occurrence in industry-level single stock predictions. ATR was also designed to support daily prices and measures volatility over 14 day periods, which makes it even more applicable to our data. ATR additionally only measures an absolute value of distance between the previous day close price and the current high and low prices, not considering direction, which is why we also need ADX to provide insight on the direction of the stock's price movement. ADX uses the ATR to calculate a scaled and smoothed moving average value that indicates the strength of a stock's trend. In this sense, it is a lagging indicator, which requires the data to have an established trend before ADX can provide a reliable signal. This is why we also use ATR as a feature in the scenario that we cannot rely on ADX. In addition to signalling the strength of a trend, ADX is renowned for assisting in market timing methods such as a buy signal when the ADX peaks and starts to decline. With this strategy you would sell when the ADX stops falling and goes flat.

Other volatility indicators include Bollinger bands, of which we used the high band indicator to signal when a stock is performing above its expected range of volatility. The Bollinger band is also useful to us in the calculation of the relative strength index (RSI), a momentum indicator. RSI is useful in indicating whether a stock, index, or other investment is overbought or oversold. It is calculated as the inverse of the average gain over average loss during a defined period of time, scaled by a constant to result in a score from 0 to 100. The RSI is most helpful in a non-trending environment in which the the value of a stock or other financial instrument fluctuates between a range of two prices. RSI is particularly useful when it reaches the extremes of its 0 to 100 range. The last single stock indicator we looked at is Williams %R, which is the inverse of another stochastic oscillator. Essentially this indicator gives the relationship between the close price and the high-low range for a given period in the past (default period is usually 14 days for daily stock price predictions). This indicator also, on the extremes of its range from 0 to -100, is useful to tell whether a stock is oversold or overbought.

Featurization - Industry Level

Since the purpose of our project was to try and improve single stock predictions by adding industry level features, we explored a number of industry and market-level features. We added these features to the set of single-stock, for each stock (matching by date and normalizing). We quickly realized that there is little access to index-level features for free on Quandl. We were able to find the following industry level features (from Quandl provider "NASDAQOMX"), which track general tech-sector stock performance.

- NASDAQ-100 Ex-Tech Sector (NDXX)
- NASDAQ-100 Technology Sector (NDXT)
- NASDAQ-100 Technology Sector Total Return (NTTR)

At the market level, we were able to find the following (from Quandl provider "NASDAQOMX"). These indices track general NASDAQ performance, in America, across all stocks, and across smaller sets of companies (by market cap):

- NASDAQ N America Index (NQNA)
- NASDAQ US All Market Index (NQUSA)

[3] https://technical-analysis-library-in-python.readthedocs.io/en/latest/index.html

- NASDAQ US 1500 Index (NQUSS1500)
- NASDAQ US 450 Index (NQUSM450)
- NASDAQ US 300 Index (NQUSL300)
- NASDAQ US Small Cap Index (NQUSS)
- NASDAQ US Large Cap Index (NQUSL)
- NASDAQ US Mid Cap Index (NQUSM)

At the market-level, we were also able to find the following indicators (from Quandl provider "URC"). These indicators track more sophisticated measures of market performance, including the measures of advancing, stagnating and declining markets:

- NASDAQ: Number of Stocks with Prices Unchanged
- NASDAQ: Number of Stocks Making 52-Week Lows
- NASDAQ: Number of Stocks Making 52-Week Highs
- NASDAQ: Volume of Stocks with Prices Unchanged
- NASDAQ: Volume of Stocks with Prices Declining
- NASDAQ: Volume of Stocks with Prices Advancing
- NASDAQ: Number of Stocks with Prices Unchanged
- NASDAQ: Number of Stocks with Prices Declining

Modeling

Linear Regression

We began by adding stock-specific and market-specific explanatory variables, and ran an ordinary least squares linear regression on this new data set. This was a natural starting point for our project because of its relative simplicity. In linear regression, we estimate the model parameters under the assumption that the relationship between the dependent variable, 'closing price', and the explanatory variables ('volatility_avg, volume, etc) is linear. Therefore our prediction, the expected value of

$$y \mid x \text{ is: } E[y \mid X] = \beta X + \varepsilon ,$$

where ε represents noise. Using OLS, we estimate the β parameters by minimizing the sum of the squared difference between the observed dependent variables and predicted values. From this, we use the r^2 value to determine the strength of the model, where r^2 represents the proportion of variance in the dependent variable that is predictable from the explanatory variables. As mentioned earlier, this model is only meant to be a starting point for our project, and for many stocks, will not produce a good estimate. The model assumes a linear relationship and independence of errors of the response variables, both of which may not be true.

Ridge Regression

Ridge regression, as well as Lasso regression (discussed later) both use regularization techniques to penalize estimates in order to correct some of the overfitting fallacies found in the ordinary least-squares method of estimating coefficients used in linear regression. Ridge regression uses L2 regularization, a way of assigning a penalty to coefficients that is dependent on the tuning parameter (λ) and the sum of the squares of the coefficients. Where ridge regression differs from linear regression, ultimately, is in the ridge parameter (k) when estimating the coefficients from regression:

$$\overline{\beta} = (X^T X + kI)^{-1} X^T Y$$

The value of k, scales the identity matrix, creating a "ridge" on the diagonal and adjusts the values of the estimated coefficients. Adding this parameter times the identity matrix also solves the problem of $X'X$ having dependent columns, guaranteeing that it is invertible. This also means that the results of ridge regression are non-sparse, which means that there are more useful, non-zero values in the resulting matrix. Choosing this parameter is a complex task and can drastically change the result of ridge regression.

Lasso Regression

Lasso regression is based on the same idea of L2 regularization from ridge regression; it just penalizes the estimates of the the coefficients using the sum of the absolute values of the coefficients which creates a tighter bound by the sum of squares from ridge regression. Thus, it qualifies as an L1 regularization. This naturally induces sparsity in the coefficients (can be understood as intersecting the corner of the polytope-shaped constraint region).

Where lasso also stands out from ridge and linear regression is in terms of accuracy of prediction. Because the minimum squared error and R-square values for a model using lasso regression are higher, lasso regression tends to be more accurate. Lasso regression also allows feature selection, where the coefficients of concern are used and the others are reduced to 0. This is absent in ridge regression and adds another degree of general accuracy to lasso regression.

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 \right\}$$

Elastic Net Regression

Elastic net regression linearly combines the L1 and L2 penalties of lasso and ridge regression respectively. In typical lasso regression, the coefficients that are being reduced to zero might be correlated to the coefficients that are actually being used. This gives less-than-optimal results. To overcome this, elastic net forms groups of correlated coefficients. Consider a group of n such clusters that suitably describe the data. If we find any one coefficient that is a strong predictor of the model, we include the entire cluster that coefficient belongs to so that no correlated coefficients are lost.

$$\hat{\beta} \equiv \underset{\beta}{\operatorname{argmin}}(\|y - X\beta\|^2 + \lambda_2 \|\beta\|^2 + \lambda_1 \|\beta\|_1)$$

Random Forests

Some relevant background:

Decision Trees: Decision trees are hierarchical models that learn "heuristics" or decision rules (at the feature level) in order to classify or regress in inputs. Each leaf node in a tree represents a class in classification and a range of target values in regression. Each node in the tree itself is a decision rule that "carves up" feature space. To train, we pass a data point down from the root, seeking the combination of feature and value of that feature which minimizes the entropy remaining in the dataset after the split. This is equivalent to maximizing the information gain (mutual information) at the split:

$$maximize\ I(X_{j,v};\ Y) := H(Y) - H(Y - X_{j,v})$$

With training points $\{(x_i, y_i)\}^n_{i=1}$, indicator variable $X_{j,v} = \{1 \text{ iff } x_j < v \text{ else } 0\}$ and random variable Y is the class of the datapoint in question.

Ensembles: Ensembling assumes that if we take many "simple" classifiers / regressors and average their predictions, we will get a "smarter" prediction than if we had taken the prediction of any single one of them. Intuitively, an average of random variables will have the same mean as a single model, yet reduced variance!

Random Forests:
Random Forests are ensembles of decision trees. We seek to select and train the trees in the forest in such a way (mostly by tweaking hyperparameters) as so reduce the correlation between them (and thus reduce the variance!).

Support Vector Regression

SVRs are the regression analog to the more common Support Vector Machines (SVMs).

SVMs: SVMs are improvements on the simple linear hyperplane-finding algorithm: perceptions. SVMs seek to maximize the smallest orthogonal distance ("margin") between the decision-boundary hyperplane and the nearest point ("support vectors") in each class. We can give this margin "slack" (with slack variables) and allow it to be violated (at some penalty) or leave the margin "hard" with no slack. Training SVMs is solving a constrained optimization problem. "Kernels" (e.g. RBF, polynomial) can be used to "lift" the data and thus provide greater separability. SVMs call points on one side of the hyperplane class A and those on the other class B.

SVRs:
SVRs simply gives a margin on possible error in the regression epsilon and finds a separating hyperplane maximizing the margin.

Conclusions

In conclusion, our intuition and hypothesis, that investors do not solely consider the specific characteristics of a company, are supported by our data. For each of the models that we experimented with, adding industry and market-level features increased the r^2 value by at least 0.05, and as much as 0.19 in the case of Ridge Regression.

The modeling techniques build on each other. Lasso regression and Ridge regression are improvements on the simple linear regression model since they use the L1 and L2 regularization techniques respectively. As we can see in the notebook, Lasso regression dramatically outperforms Ridge Regression, with an r^2 value of 0.86 versus 0.64. Elastic Net, which uses strategies from both Lasso and Ridge regression, falls in between these values, with an r^2 value of 0.7. In our case, it makes sense that Lasso regression performs best, because we used a wide variety of features. Some features may be more correlated with stock A, whereas a different set of features may be more correlated with stock B. In Lasso Regression, coefficients can shrink to exactly zero, which would also allow it to cull the list of features in each particular case. If we were to further extend this

project, our first step would be to experiment more with feature engineering and improve the performances of Ridge Regression and Elastic Net.

As for SVR's and Random Forests, we can see from their descriptions why Random Forests would perform better. Support Vector Machines is intrinsically two-class, whereas Random Forests are intrinsically suited for multi-class problems, like our stock forecasting project. As such, it makes sense that Random Forests would dramatically outperform the other models, with an r^2 value of 0.96.

TODO:

A. RILEY : Code up / Find package for market/industry indicators
B. Pull all functions into a util file so the notebook is clean
    a.
    b. Function for each model
C. Write a plotting notebook to validate some assumptions about the data
    a. Ask some questions about the data
    b. Answer them through plots

-----FRIDAY--------

D. Move onto Modeling

E. Paper
    a. Abstract
    b. Intro (problem setup, background, related work)
    c. Dataset
        i. Where we got the data, etc… How we chose stocks
    d. Featurization
    e. Modeling
        i. Justify the models, give bounds, theory behind them
    f. Conclusions

Financial Modeling Project Proposal
Team Members: Riley Edmunds, Aman Sidhant, Ronald Zhang, Pransu Dash

We have one main question we would like to tackle (#1). If this question is too broad, we have prepared two alternatives (#2 and #3).

1. **Main Hypothesis: we believe that when predicting next-day prices, using broad market and industry indicator features (in addition to single-stock features) will improve upon the performance achieved using single-stock features.**
    a. Intuitively, this makes sense. In the short run, such as next-day performance, general market outlook may weigh heavily on investors' appetite for new investments. We do not expect the variance of these results to differ significantly from single stock prediction models due to the stock market being heavily dependent on industry performance as well as individual company performance. In terms of noise, industry indicators such as competitors' performance or other sector-level indicators could lead to more noise in the results because of the additional, possibly insignificant data.
2. How much data do we look at and predict how much of the future?
    a. Eg. look at data from around 2007 and run a model on it to predict 2008 prices, compare it to actual, recession type 2008 prices. Do something similar now to see if the vague talk of an incumbent financial crisis is true.
3. Are percentage drops in individual stock prices (A) anomalies that will quickly correct or (B) beginning of a significant fall in price?
    a. I.e. Given a price drop for stock X on day d, can we predict within some confidence interval, the stock price at day d+7 (next week), at which point, if the drop was (A), we know it was an anomaly, else (B).

1. Modeling techniques.
    a. There are many established modeling techniques developed, and we plan on focusing on two categories.
        i. We will consider extending machine learning techniques, such as using Random Forests and SVM[4]
        ii. We will also consider statistical methods such as time series model, where the stock price movement is treated as a function of time series and solved as a regression problem. Listed below are models supported in literature[5]
            1. Autoregressive Models (AR)
            2. Moving Average Models (MA)
            3. Seasonal Regression Models
            4. Distributed Lags Models

[4] https://arxiv.org/pdf/1605.00003.pdf
[5] https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/

b. Ultimately, we hope to find which combination of these techniques will result in the most accurate forecasts.

2. The main part of a good model is how the features are engineered. In other words, how do we take raw data and transform it into something that is easier to model (Eg. Raw stock price to percentage change.)
    a. Some common representations of stock price data in literature:
        i. Percent change daily (normalized like in the Jupyter notebook)
        ii. Frequency domain (FFT)
        iii. Dimensionality reduction - i.e. PCA.
    b. Normalization:
        i. We want to have consistent normalization between the train, validation, and test sets - thus we will save out the normalizer used in training to apply at validation/test time.
    c. Features:
        i. Training:
            1. Single stock: previous {3,5,10,30,50,100,365, 2yr, 3yr} days prices and moving averages or various granularity.
            2. **Industry: adjusted closing prices for all companies in the industry**
            3. Volatility: inter-period (day, multi-day, week, etc) variance in price
            4. **Market Beta: beta between stock and general market.**
            5. **Industry Beta: beta between stock and industry/sector.**
            6. Other common technical indicators include RSI, MACD, William R%, Bollinger Bands, Stochastic Oscillator (momentum of price over time), Price Rate of Change, On Balance Volume, and others listed here.
                a. Moving Average (Indicator) - Useful for short term trading. General rule, if the 50 day moving average crosses above the 200 moving average is a buy signal, and when the 50 day moving average crosses below the 200 moving average is a sell signal
                b. MACD (Oscillator) - General rule if it is above 0, is consider an uptrend and below 0 it is a downtrend.
                c. Chaikin Oscillator - Follows from MACD. Needs at least a 10 day period to work. Uses the Exponential Moving Average, which recent values more heavily.
                d. RSI (Oscillator) - General rule, when it is over 70 (can be extended to anything less that 100) is overbought and may be an indicator of a correction, if it is under

30 (can be extended to anything greater than 0) is oversold and may be an indicator of a bounce.

e. Stochastic (Oscillator) - Works on the principle that momentum always changes direction before price. One strategy is to only make moves when the SO is within a particular range. The stochastic oscillator is plotted within a range of zero and 100. Readings above 80 are considered overbought while readings below 20 are considered oversold. The oscillator has two lines, the %K and %D, where the former measures momentum and the latter measures the moving average of the former. The %D line is more important of the two indicators and tends to produce better trading signals.

f. On Balance Volume (Indicator) - Useful for working with trends

g. Exponential Moving Average - works well for short term stuff

7. In general, indicators are good for a trending stock, and oscillators are good for sideways stocks. A good strategy would be to use Moving Average to see if the stock is trending or sideways, and then use oscillators to give better results.
8. **We would also look at Broad Market Indicators on the following levels:**
    a. **Market Level such as the DJIA, SP500, NASDAQ, NDX.**
    b. **Sector Level such as BKX, CRX, RLX, XOI.**
    c. **TICK, TRIN, and others listed here.**

ii. The target variable ("label") is just the adjusted daily closing price for a particular stock - or DOW Jones

3. After finalizing a set of features, we need to split the data set into two to simulate how well the various models perform. We will need to test multiple models to see how they perform on different combinations of the features. We will rely on models in scikit-learn to do this analysis. Additionally, we would like to try having sub-models to the main model: i.e. model A predicts future volatility, model B predicts future Beta, and the master model uses predictions of volatility and beta as features.

DRAFT

Financial Modeling Project Proposal

Team Members: Riley Edmunds, Aman Sidhant, Ronald Zhang, Pransu Dash

2. There are many established modeling techniques developed, and we plan on focusing on two categories:
    a. ML
        i. Random forests
            1. CART / Bagging / Boosted Trees
        ii. SVM
        iii. Logistic Regression
        iv. Linear Regression
            1. If we choose a longer time frame for a single stock, the model could develop a linear trend
    b. Time-Series
        i. Code here and here
        ii. Autoregressive Models (AR)
        iii. Moving Average Models (MA)
        iv. Seasonal Regression Models
        v. Distributed Lags Models
3. Featurizing techniques
    a. The essential part of a good model is how the features are engineered. In other words, how do we take raw data and transform it into something that is easier to model (Eg. Raw stock price to percentage change.)
        i. Some common representations of stock price data in industry level models:
            1. Percent change daily (normalized like in the Jupyter notebook)
            2. Frequency domain (FFT)
            3. PCA
        ii. Normalization:
            1. We want to have consistent normalization between the train, validation, and test sets - thus we save out the normalizer from training to apply at validation/test time.
    b. Features: this includes what we use to train versus what we use to test our model
        i. Training:
            1. Single stock: previous {3,5,10,30,50,100,365, 2yr, 3yr} days price, moving averages
            2. Industry: historical data for all companies in the industry
            3. Volatility: essentially inter-day variance in price
            4. Beta: correlation between stock and general market
            5. Industry beta: correlation between stock and industry trends
            6. Common Technical Indicators:
                a. RSI (could be interpreted as a sentiment indicator)

b. MACD
c. William %R
d. Bollinger Bands
e. All here

7. Other Common Technical Indicators:
    a. Stochastic Oscillator
        i. Momentum of price over time, which requires choosing appropriate time frame to measure over
    b. Price Rate of Change
    c. On Balance Volume
8. Sentiment Indicators:
    a. Volatility Index: implied volatility due to market estimate
        i. Could be used to predict whether lull is temporary or an indicator of stock decline
    b. Put/Call Ratio: essentially sell to buy ratio

ii. Test:

1. Adjusted daily closing price for a particular stock - or DOW Jones

4. After converging on the features, we need to split the data set into two to simulate how well the various models perform. We will need to test multiple models to see how they perform on different combinations of the features. We will rely on the models in scikit-learn to do this analysis. Additionally, we could have multiple models predict multiple things: one model measures volatility, one measures future prices etc.

Questions to answer:

2. How much data do we look at and predict how much of the future?
    a. Eg. look at data from 2007ish and run a model on it to predict 2008 prices, compare it to actual, recession type 2008 prices. Do something similar now to see if the vague talk of an incumbent financial crisis is true.
3. Are X% drops in individual stock prices (A) anomalies that will quickly correct or (B) beginning of a significant fall in price?
    a. I.e. Given an X% drop on day d, can we predict within some confidence interval, the stock price at day d+7 (next week), at which point, if the drop was (A), we know it was an anomaly, else (B).
4. **Hypothesis: we believe that modeling broad market and industry indicator features (in addition to single-stock features) market-wide features will improve upon the performance achieved using single-stock features to predict next-day prices. We will try *all models* and see what does best lol**

Questions to answer for the proposal:

- What is the motivation behind your idea?
    - TODO: find some papers that back this up

- What do you expect to see from the results?
    - In terms of Variance? Noise?
    - How much better performance do we expect?
    - Do we expect it to predict better based on what metric? L1, L2, etc…?

Resources:
Time Series Book
Computational Investing Course

TODOs:
@RILEY:

- find and add single-stock cross-industry and cross-market indicators
    - BROAD MARKET INDICATORS
        - Market-level:
            - DJIA, SP500, NASDAQ, NDX
        - Sector-level:
            - See page
        - Other
            - TICK, TRIN
            - Everything here
- Skim time series book and look for modeling statistically sound techniques to try
    - See stuff added in modeling