Machine Learning Engineer Nanodegree
Capstone Project

I. Definition

Project Overview

Investors and traders use a myriad of approaches to profit off the buying and selling of stocks and derivatives in public markets. These approaches include: (1) fundamental analysis, which relies upon the examination of fundamental business factors such as financial statements, (2) technical analysis, which focuses upon price trends and momentum, and (3) quantitative analysis, which may uses a combination of both. Traders constantly attempt to discover new strategies that exploit previously unknown patterns in an attempt to predict how a stock's price will move.

Traders generally rely on information available to the public to develop their strategies. This information can include historical stock prices, financial numbers, quarterly statements, and news. However, public information seems to provide less value every day for the average trader, as sophisticated algorithms and high-frequency traders can react to new information in fractions of a second, moving prices faster than most can even digest the news.

An approach of recent interest is to attempt to infer information from trades by company insiders. People that run a company can reasonably be expected to best know how their company will perform in the future. In the U.S, insiders (i.e., directors, officers, or owners for 10% of equity) must report any personal trades of their company's stock with the Securities and Exchange Commission (SEC) on a Form 4 filing. While insider-trading laws restrict how insiders can trade their company's stock, recent research has concluded that insider-trading patterns can provide clues as to future corporate events that may trigger stock price fluctuations [1]. Also, company insiders may have better information not only about their own company, but also about companies in the same industry, particularly those in the same supply chain [2].

In this project, I attempted to use data extracted from SEC forms (Form 4) to predict significant stock price increases or drops. The input dataset describes how company insiders bought and sold stock over a period of time, and how the stock price fluctuated during the same period. I then trained machine learning models to predict stock prices changes based on new insider trades input.

Problem Statement

While corporate insider trades may provide clues as to how insiders think their company stock will fare, it is challenging to weed out relevant trading events from noisy trading data. For example, insiders may sell stock on a regular basis simply out of a need for cash and not with any particular strategy in mind. Therefore, a machine learning approach may be useful in analyzing insider trading data to learn what types of trading activities may predict a significant corporate event and an associated sharp stock price change. In short, the problem to be solved is how to use Form 4 data to predict significant stock price increases or drops.

To solve this problem, I used a supervised learner that uses inputs from SEC forms (Form 4) and trained the model to classify stocks into class labels comprising ranges of stock price returns At a high level, the goal of this classifier was to accurately predict stock price changes given new Form 4 data.

Datasets were constructed solely from publicly available data, in particular, Form 4 data from SEC's Edgar database and historical stock prices from Tiingo:

- EDGAR Database: https://www.sec.gov/edgar/searchedgar/webusers.htm

- Free Tiingo API: https://api.tiingo.com/

Data was pulled from these sources and parsed using Python SOAP libraries. I then constructed datasets for a supervised learner, with data extracted from Form 4 as input features and stock price percentage change as predicted output. To simplify the input and reduce the dimensionality, certain pre-processing was done. Below are the input features and output:

Input features

- Insider **Net Buy Count** of company stock per month for 12 months (12-tuple of continuous variables from -1 to 1, can be discretized in 0.05 intervals)
- Insider **Net Buy Volume** of company stock per month for 12 months (12-tuple of continuous variables, can be discretized in 0.05 intervals)
- Company Industry **Sector** (Healthcare, Industrial Goods, Services, Technology, Utilities, Basic Materials, Conglomerates, Consumer Goods, Financial)
- **Market Cap** (Small, Medium, Large)

Output feature

- **Stock return over S&P500** for the 6 months following the inputted 12 months, discretized into 3 categories, for example: (-∞,-10%], (-10%,10%), [10%,∞).

Each datapoint will thus correspond to a single company and a range of time.

The solution trains various supervised learning classifiers to classify stocks into one of the three stock return categories. I used Support Vector Machines, Naïve Bayes, and ensemble learners, using grid searches to obtain optimal parameters, as discussed below.

Metrics

The model was evaluated on its ability to correctly classify stocks. The data was divided into training and testing sets and cross-validation was performed. The performance of the model was then scored using various evaluation metrics: accuracy, F1-score, precision, and recall.

F1 score was the most useful metric, since it combines both precision and recall, which are better metrics for this scenario where the output is not uniformly distributed. Precision of a class will measure the number of stocks that were correctly classified into that class over the total number of stocks classified into that class. Recall of a class measures the number of stocks that were correctly classified into that class over the number of stocks that should have been classified into that class.

II. Analysis

Data Exploration

Dataset Description

The dataset consists of data from Jan-2010 through Dec-2016. Each datapoint is associated with a single stock and its price return from a first date to six (6) months thereafter (e.g., AAPL, January 1, 2011 through June 30, 2011). The target variable is the stock price return relative to the S&P500 index for that 6-month period, computed as:

$$Return_{rel} = \frac{\frac{Price_{last}}{SP500_{last}} - \frac{Price_{first}}{SP500_{first}}}{\frac{Price_{first}}{SP500_{first}}}$$

Adjusted prices were used for stock prices, and adjusted prices of SPY fund (trust ETF that tracks the S&P500). In this manner, the feature data is being used to predict the future return of the stock (i.e., the following 6 months).

The feature set includes the monthly Net Buy Count (NBC1-12) and Net Buy Volume (NBV1-12) for each of the twelve months prior to the 6-month period of stock price return data (e.g., January 1, 2010 through December 31, 2010 for the above example). Net Buy Count reflects ratio of the number of stock purchases vs. stock sales in a month:

$$NBC = \frac{Num_{purchases} - Num_{sales}}{Num_{purchases} + Num_{sales}}$$

Net Buy Count reflects ratio of the volume of stock purchased vs. stock sold in a month:

$$NBV = \frac{Vol_{purchases} - Vol_{sales}}{Vol_{purchases} + Vol_{sales}}$$

The feature set also includes the Industry Sector associated with the stock (out of 10 categories) and the market capitalization category (small, medium, or large). The target column is the return for the stock from the first day of the 6-month period to the last day of the period, computed as (Price Last Day – Price First Day) / Price First Day. These last two columns are categorical variables that will be divided into binary columns, as discussed below in the Data Preprocessing section. Below is a sample of 5 datapoints:

NBC1	NBC2	NBC3	...	NBV11	NBV12	SECTOR	MKT CAP	RETURN
1.000000	0.066667	-0.428571	...	-0.190212	-0.183784	Information Technology	small	-0.038247887838
0.066667	0.000000	-0.217391	...	-0.183784	-0.139860	Information Technology	small	0.0707736976746
0.000000	0.000000	-0.578947	...	-0.139860	-0.007563	Information Technology	small	-0.0917734937395
0.000000	0.000000	-0.428571	...	-0.007563	0.006516	Information Technology	small	0.126127808822
0.000000	0.000000	0.555556	...	0.006516	-0.072261	Information Technology	small	-0.131451077943

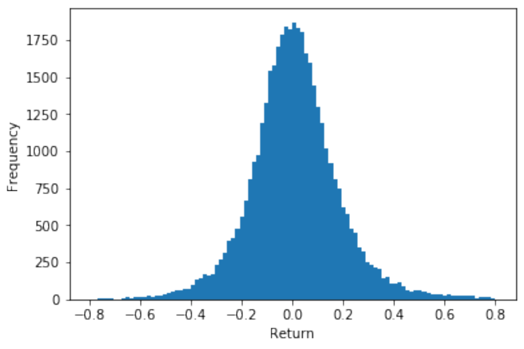
Dataset Statistics

In order to balance the target categories, I calculated some simple statistics for the return prices, shown below.

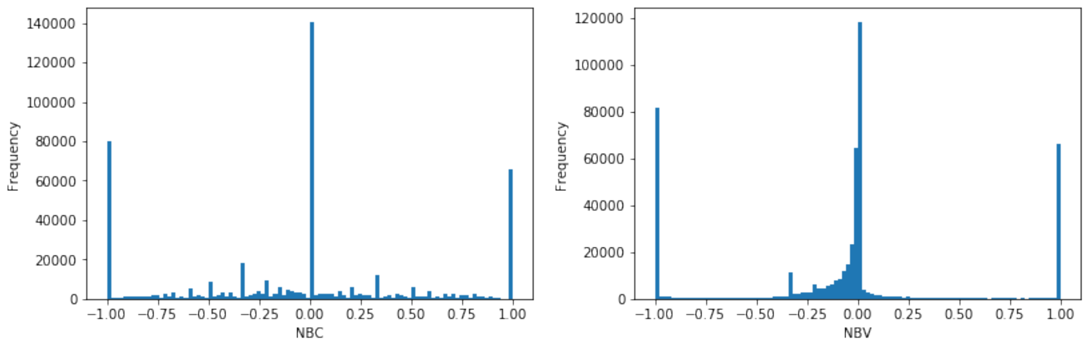
Min return	-0.867926289208
Max return	6.66432449583
Mean return	0.0138842243279
Median return	0.00513197463559
Std Dev of return	0.197127084811
First quartile	-0.0916451313919
Third quartile	0.105930831069
First third	-0.0581371249263
Second third	0.0647762819449

Exploratory Visualization

The following histogram shows the frequency of relative price returns over the entire dataset. Visual inspection shows that the returns are normally distributed around the mean.



The histograms below show the frequency of values for Net Buy Count and Net Buy Volumes over all months in the dataset. The graph reveals that for a large majority of months the NBC and NBV were 0.0, indicated that for a large number of months there were no insider purchases or sales of stock. The charts also show that a disproportionate number of months have NBC and NBV values of 1.0 and -1.0, indicating that for a large number of months insiders either exclusively bought (1.0) or exclusively sold (-1.0) stocks.



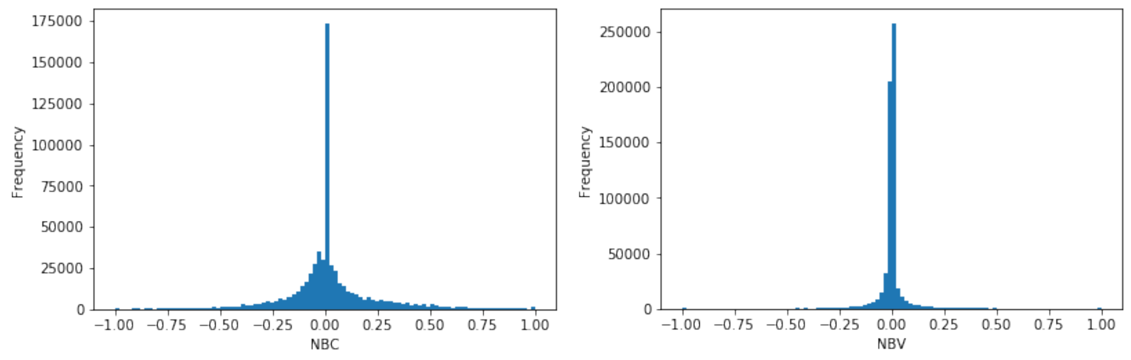
This indicates a problem with the dataset where the metrics don't yield as much information as they could. For example, a month where 1 stock was bought and a month were 1,000 stocks were bought may both have a NBV value of 1.0. As such, I decided to re-normalize the NBC and NBV values as a fraction of the maximum NBC and NBV for a given company:

$$NBC_{norm} = \frac{Num_{purchases}}{Max(Num_{purchases})} - \frac{Num_{sales}}{Max(Num_{sales})}$$

$$NBV_{norm} = \frac{Vol_{purchases}}{Max(Vol_{purchases})} - \frac{Vol_{sales}}{Max(Vol_{sales})}$$

$Max(Num_{purchases})$ represents number of stock purchases for the month with largest number of stock purchases for that company in the entire dataset. Likewise, $Max(Num_{sales})$ represents number of stock sales for the month with largest number of stock sales for that company in the entire dataset. $Max(Vol_{purchases})$ and $Max(Vol_{sales})$ represents the same quantities for the number of stocks purchased and sold, respectively.

As such, NBC_{norm} and NBV_{norm} will also be between -1.0 and 1.0. The graph below shows the values are better distributed than with the previous normalization:



Algorithms and Techniques

I trained two classifiers to predict return categories: Naïve Bayes and an Ensemble of weak decision tree classifiers. Both classifiers are adequate to solve the problem here. While I tried Support Vector Machines as well, SVM's running time was too long to perform adequate grid search and cross validation, and performance was similar to the other models with the few parameters I tested.

Naïve Bayes

NB assumes each pair of features is independent of each other, which is not the case here. However, NB often still works even when this assumption is not true. NB is also very efficient, which is advantageous for a large dataset like the one used here.

Ensemble Methods

Ensemble methods work well with a wide array of data. They can represent very complex hypotheses while being less prone than other models to over-fitting, and are more efficient than SVMs. Ensemble methods can achieve very high accuracy by combining multiple hypotheses by weak learners into a stronger hypothesis. This also enables them to represent more complex hypotheses that may not be contained in the hypothesis space of the models from which it is built. Therefore, they may work well when other models cannot achieve the complexity necessary to represent the target concept. They are also less prone to overfitting. Bagging algorithms perform well when used with strong models, such as NB in this case.

Bagging may be a good candidate for this problem because it can make use of multiple strong models, to come up with a robust hypothesis while using less data than a boosting algorithm. It may also be able to represent more complex hypotheses that are non-linear, which may be necessary with this dataset.

Benchmark

Since I divided the target data into three classes of equal size, the benchmark for classification using random prediction is 1/3 (33%) precision and (33%) recall, which would yield an F1 score of 0.33. That is, randomly selecting a category for each stock would yield a precision and recall of 33%. If our model performs better than 33%, it is doing a better job than randomly categorizing stocks, indicating the classifier obtained useful insights from the data.

III. Methodology

Data Preprocessing

First, I eliminated missing values in the RET columns, which were labeled either "Fail", "None", or "NaN". "None" points were data that was not available and "Fail" points were data that failed to download for some reason. The rest of the columns did not have any data anomalies.

The dataset contains COMPANY and MONTH columns (not shown above), which I dropped since these don't add any meaningful information for our purposes and would dramatically increase the dimensionality of the problem. While the dataset contained return data for 1, 6, and 12 months, this report focuses on 12 months since this provided the best results. Thus, I dropped RET1 and RET6 columns, and renamed RET12 to RET. Finally, I separated the categorical variables of SECTOR and MKTCAP into separate binary variables using the get_dummies() function.

In order to create balanced classes, I used the first third and second third of the return data (as computed above in the Data Exploration section) as the cutoff point for the 3 categories. Therefore, based on the observed statistics, the target classes were defined as follows:

- Class 1 ("DOWN"): Under -0.05813 return
- Class 2 ("STABLE"): Between -0.05813 and 0.06477
- Class 3 ("UP"): Above 0.06477

Implementation

To find the best model, I first tried both models with their default parameters using K-fold cross-validation to assess their performance. The models were input the entire dataset for this step, since the entire dataset can be used for both training and testing using cross-validation. The results are shown below:

Model	**F1 Score**
Naïve Bayes	0.4012
Ensemble (Bagging)	0.4003

I then separated the dataset into a training set including 80% of the data, and a testing set using 20%. I trained both models and the results of classification using the testing set are shown below.

<u>Classification Report for Naïve Bayes</u>

Label	Precision	Recall	F1-score	Support
UP	0.39	0.42	0.40	6572
DOWN	0.43	0.37	0.40	6341
STABLE	0.41	0.43	0.42	6377
avg / total	0.41	0.41	0.40	19290

<u>Classification Report for Ensemble</u>

Label	Precision	Recall	F1-score	Support
UP	0.39	0.43	0.41	6572
DOWN	0.43	0.36	0.39	6341
STABLE	0.40	0.43	0.42	6377
avg / total	0.41	0.41	0.41	19290

Refinement

Parameter Optimization

For NB there are no parameters so there was no need to perform parameter search. The model does take prior probabilities as a parameter, but in this case there are no prior probabilities for each class.

In order to determine the best parameters for the Bagging classifier, I used grid search cross-validation using Naïve Bayes estimators. The parameters tested were:

```
params = {      'n_estimators': [1,10,100,500,1000],
                'max_samples': [0.3,0.5,1.0],
                'max_features': [0.3,0.5,1.0]}
```

n_estimators corresponded to the number of estimators used. Max_samples and max_features corresponded to the fraction of the total samples and features, respectively, to draw from the dataset for training. Grid search of the above parameters yielded the following optimal parameters:

```
best_params_ = {      'max_features': 0.3,
                      'max_samples': 0.3,
                      'n_estimators': 100}.
```

The results of the ensemble model trained with these parameters on the testing set are shown below.

Model	F1 Score
Naïve Bayes	0.4012
Ensemble (Bagging)	0.4003
Ensemble (optimal)	0.4022

Classification Report for Ensemble (with optimal Bagging parameters)

Label	Precision	Recall	F1-score	Support
UP	0.40	0.44	0.42	6572
DOWN	0.43	0.37	0.40	6341
STABLE	0.41	0.42	0.42	6377
avg / total	0.41	0.41	0.41	19290

As is readily apparent, the results were marginally improved using optimal parameters, but the original models perform similarly.

IV. Results

Model Evaluation and Validation

While the results of the final model shown above are somewhat underwhelming, the model can still be useful. For stock trading, we only need one stock pick. Thus, the classifier can be useful if it can confidently classify a single stock. To assess this, I used the predict_prob() function to determine the classifier's confidence in its predictions. I then sorted the results in descending order to determine the highest confidence predictions.

The table below shows the results for the predictions with 0.60 probability or higher. A first thing to note is that all the highest confidence predictions were in the DOWN category, signaling the model is better at predicting which stock are going down in price. The precision and accuracy for predictions with over 0.60 probability is much higher than for the rest of the dataset.

Company	First month	Predicted Return Category	Actual Return Category	Actual Return Value	Prediction Confidence
UNT	2011-06	DOWN	DOWN	-0.104260989149	0.701770
SLCA	2013-06	DOWN	DOWN	-0.458631326212	0.678251
UNT	2011-03	DOWN	DOWN	-0.160898545378	0.673817
EEQ	2013-11	DOWN	DOWN	-0.232851191059	0.642364
WLL	2012-02	DOWN	STABLE	0.0100388284056	0.622348
SLCA	2012-07	DOWN	UP	1.19795096909	0.621403
NGL	2012-07	DOWN	UP	0.223497806037	0.614668
ENLC	2014-03	DOWN	DOWN	-0.734449961872	0.611460

MDR	2013-09	DOWN	DOWN	-0.283808352244	0.610698
WGP	2014-06	DOWN	DOWN	-0.320706536349	0.608916
BWP	2013-11	DOWN	DOWN	-0.254832641306	0.608861
ENBL	2014-03	DOWN	DOWN	-0.636109502496	0.606810
CVRR	2012-12	DOWN	STABLE	-0.0175873114779	0.605505
UNT	2011-08	DOWN	DOWN	-0.128695085235	0.604216
GEL	2012-08	DOWN	STABLE	-0.0726908388025	0.603567
GEL	2011-04	DOWN	UP	0.46960215203	0.601394
KOS	2011-04	DOWN	DOWN	-0.256496072168	0.600888

Metrics for DOWN predictions with 0.60 or higher probability

Metric	**Score**
Precision	0.647
Recall	1.0
F1 Score	0.8235

As such, the final model may be useful to earn money trading by using the predictions with the highest probability. For example, an investor could place "short sells" (which produce gains when the stock price goes down) in the 10 stocks with a probability of 0.60 or higher and expect 6 or 7 of them to obtain gains. More realistically, an investor may use this as a screening tool to obtain a list of stocks that could go down, and then use other investment methods and strategies to select from among the list. As such, the model may be a viable tool to complement an investment strategy.

Justification

The model's final solution obtains around 40% precision in classifying the direction any stock will move, and around 67% precision in classifying the direction that a few stocks will move. These results are an improvement over the 33% benchmark from randomly picking stocks. Depending on an investor's risk aversion and experience with the stock market, the model may prove useful as a screening tool to find a list of stocks likely to move down.

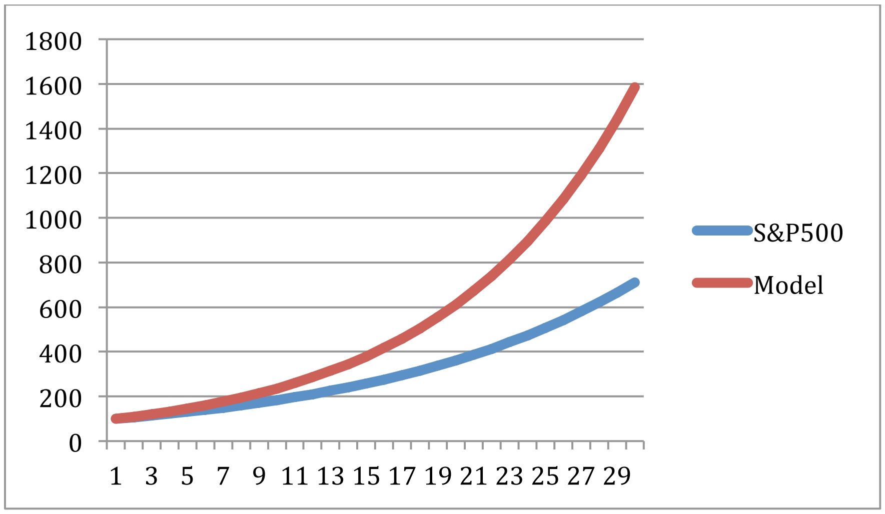
V. Conclusion

Free-Form Visualization

The model for this project attempts to predict—given data on insider trades for a year-long period—the general direction a stock price will move. A user of the trained model would input insider trades data for the past 12 months for a list of companies, and obtain a prediction for how the stock price will have moved 12 months into the future.

Based on the results, I concluded that the model may be useful to obtain a list of companies that will likely go down in price 12 months into the future. This is obtained by using the companies with a 0.60 or higher prediction probability. If a person had acquired equal-sized short positions in these companies according to the data above, the person would have obtained a 10% annual return (computed by averaging the actual return values column of the table above, which yields -10.36% return, thus yielding 10.36% for a short position).

For comparison purposes, investing in the S&P500 has historically yielded a 7% return (adjusted for inflation) [See http://www.investopedia.com/ask/answers/042415/what-average-annual-return-sp-500.asp]. Thus, the model by itself has the potential to yield a better return than the S&P500. The graph below illustrates the difference in two portfolios using each strategy over 30 years (with reinvestment of gains).



As the graph illustrates, the value of the portfolio would be an order of magnitude larger after 30 years using the model by itself. Complementing the model with additional strategies could presumably yield even better results.

Reflection

To summarize, this project attempted to predict stock price movements from insider trades data. I downloaded insider trades data for over 2000 companies over a 7 year period. I then created a dataset that reflected the number of stock transactions and the volume of shares transacted, for both stock purchases and sales by insiders. Each datapoint reflects the monthly transaction history for a company during a 12-month period, and the price return for the stock for the following 12-month period. Each datapoint also contained the company's industry sector and market capitalization. The return values were discretized intro three categories of equal sample size (DOWN,STABLE,UP). Finally, machine learning models were trained using the dataset that learned to predict which category of price return a stock would be 12-months into the future.

In retrospect, the problem I set out to solve was very challenging, and it is not surprising the solution was less than ideal. A point of reflection is how important handling of the data is before starting to train models. First, collection of the data can take a very long time when a dataset is not given. In this case, data collection involved parsing webpage results on the SEC Edgar Database, downloading over 7GB of txt files, and handling HTTP connection errors and limits. Second, putting the data in the right format is important and challenging.

I learned it is important to analyze and visualize the data thoroughly. As explained in the Data Exploration section above, I did not realize so many values were -1.0, 0, and 1.0 using the initial normalization until I visualized the data for this report. This then inspired me to re-arrange the data to reflect more information per datapoint.

Finally, it is also important to thoroughly analyze the results to obtain insights about the predictions made. In this case, although the overall precision of the model was low, the precision for a few points was good enough to make the model useful in particular scenarios.

Improvement

Given the nature of the stock prediction problem, there are many ways in which this model can be improved. The main insight that drives the insider trading solution is that insiders inherently have the best information about their company's future outlook. But the crux of the problem is that insiders will trade stock for many reasons, not just performance outlook, and are further restricted by insider trading laws in the way they can buy and sell their company's stock.

Therefore, the model may benefit from more information about the companies that may help the model distinguish between meaningful and routine insider trades. For example, historic or seasonal prices for each stock, news headlines, or other technical information may all be useful.