Project Report

Predicting Sector of a firm

Overview

This problem comes from the world of finance. For financial analysis of a company, it is useful to assign the company to appropriate sector (i.e. its peers). This assignment helps the financial analysts better understand various metrics of the company and put them in perspective. E.g. Cost of Goods Sold for a "Consumer Goods" company would be a very high percentage of revenue as compared to, say, a "Technology" company. If a financial analyst sees very high cost of goods sold, it would be useful to know whether this firm is a technology firm or a consumer goods firm.

The question then arises, how does the analyst go about assigning this sector? In United States and other financially developed countries, this assignment is done by the company itself (or by other analysts) if the company is sufficiently large. However, for very small companies (e.g. unlisted) or for other non-public firms (e.g. funds), this assignment is not readily available.

An analyst looking at a universe of small non-public companies would struggle to identify the ones relevant to their own sector. In this project, we will develop a model to assign sector to a company based on available information.

Problem Statement

Identify the correct sector of a firm based on its returns, brief description and some other available factors.

Evaluation Metric

We will use accuracy score[1] to evaluate the goodness of our model. This score reflects the proportion of accurate labeled firms (to the total number of available firms).

Benchmark

Currently, there isn't a machine learning solution to this problem. Based on interviews with portfolio managers and analysts, we estimated that it would take a portfolio manager (PM) approximately 10 minutes to correctly identify the appropriate sector/industry of a firm. In contrast, it would take around 2 minutes to determine whether the industry/sector allocation is correct or not. This gives us an approximation of current manual model which is presumed to be 100% accurate (since there is no objectively true assignment).

Consider a universe of 100 firms. Under current methodology, it would take a PM 1,000 minutes to correctly assign the industry/sector to all firms.

Now, consider a new model which assigns a sector/industry correctly (as judged by a PM) 50% of the times. To review this, we expect the PM to spend around 200 minutes (2 minutes per firm) and without

[1] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

further effort, have 50 correctly identified firms (a process that otherwise would have taken around 500 minutes).

Based on this, a model that is accurate 20% of times would breakeven in terms of marginal time spent in review. A higher accuracy would be an improvement over current methodology.

There may be additional benefits to even an “incorrect” assignment as, depending on the assignment, it may provide the analyst important insights into the firm.

Project Design

Obtaining Data

We first identified the list of tickers of all firms listed on NYSE. We downloaded this list from NASDAQ website[2]. This was a universe of 3,170 firms/tickers. From our universe, we removed all firms which didn’t have a sector identified. This reduced our universe to 2,197 firms. We further removed all firms which had a caret (^) or a dot (.) in their tickers. Tickers with caret represent non-tradable indices while those with dots represent classes of shares.

We had a list of 2,155 firms/tickers after this initial cleanup. For each of these firms/tickers, we downloaded following two types of data:

1) Fundamental data: Specifically, we used the following data points for each of the firm:
    a. Summary
    b. Stock Type
    c. Market Cap
    d. Net Income
    e. Sales
    f. Employees

    All the fundamental data was downloaded from Morningstar website[3] using Selenium with Python[4] and Firefox browser using geckodriver[5].

    From the initial list, we ignored all companies for which no data was available on Morningstar website.

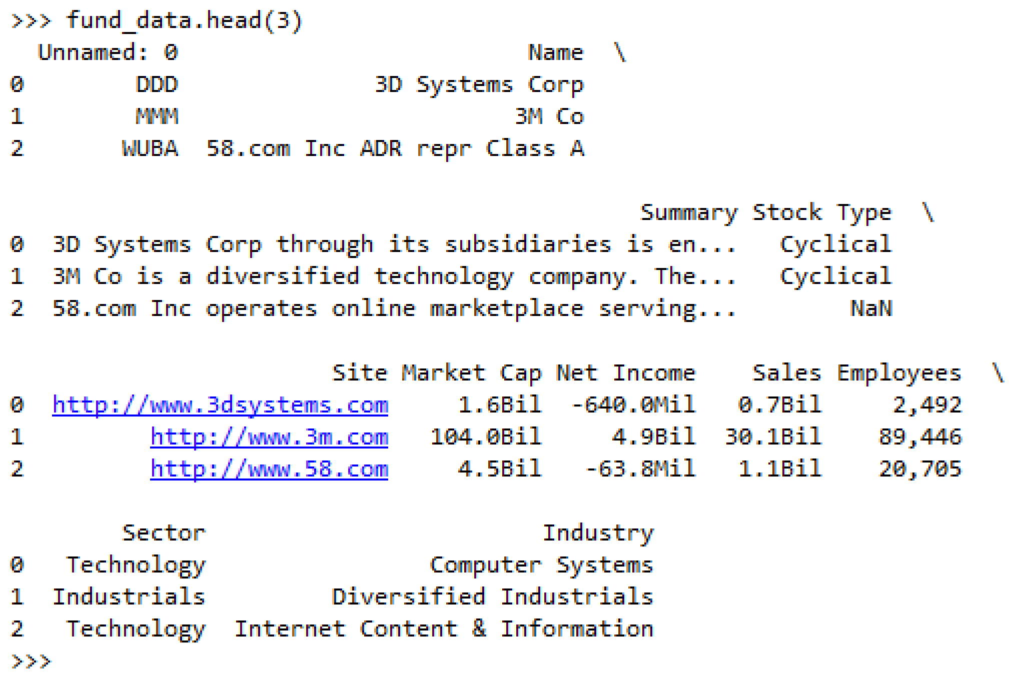
    This left us with a list of 2,065 tickers/firms which have 11 unique sectors.

2) Returns data: For each of the 2,065 tickers, we downloaded the daily price data using yahoo_finance python package[6]. We converted the price data into daily returns to use in our calculations.

[2] http://www.nasdaq.com/screening/companies-by-industry.aspx?exchange=NYSE
[3] http://financials.morningstar.com/
[4] http://selenium-python.readthedocs.io/index.html
[5] https://github.com/mozilla/geckodriver/releases/tag/v0.11.1
[6] https://pypi.python.org/pypi/yahoo-finance

Cleaning data

Let’s see what the raw data looks like:



```
>>> fund_data.head(3)
  Unnamed: 0                         Name  \
0        DDD              3D Systems Corp
1        MMM                        3M Co
2       WUBA  58.com Inc ADR repr Class A

                                             Summary Stock Type  \
0  3D Systems Corp through its subsidiaries is en...   Cyclical
1  3M Co is a diversified technology company. The...   Cyclical
2  58.com Inc operates online marketplace serving...        NaN

                       Site Market Cap Net Income    Sales Employees  \
0  http://www.3dsystems.com     1.6Bil  -640.0Mil   0.7Bil     2,492
1         http://www.3m.com   104.0Bil     4.9Bil  30.1Bil    89,446
2         http://www.58.com     4.5Bil   -63.8Mil   1.1Bil    20,705

        Sector                        Industry
0   Technology                Computer Systems
1  Industrials         Diversified Industrials
2   Technology  Internet Content & Information
>>>
```

The raw fundamental data had unusable format for some of the data which we fixed as follows:

1) Use of ‘Mil’, ‘Bil’ instead of numerical notation: This issue affected Net Income, Market Cap, and Sales data. E.g. instead of number 1,000,000, string 1Mil was used. These issues were fixed by replacing ‘Mil’ and ‘Bil’ with appropriate number of 0’s.
2) Use of strings: This issue impacted Employees data. E.g. instead of number 1000, string ‘1,000’ was used. This was fixed by removing the comma and converting the number to float.

Creating Features from Fundamental data

Instead of using raw data, we determined (based on domain knowledge) that better predictors would be ratios of various fundamental data points. Specifically we created the following ratios which we used in our model:

1) Margin: Ratio of Net Income to Sales
2) Price/Sales ratio: Ratio of Market Cap to Sales
3) Price/Earnings ratio: Ratio of Market Cap to Net Income
4) Sales/Employees ratio: Ratio of Sales to Employees
5) Earnings/Employees ratio: Ratio of Net Income to Employees

In addition, we also created dummy variables from Stock Type.

In addition to features from fundamental data, we also converted our returns data into features. Specifically, we used correlation of returns to sector returns as a feature. It is intuitive that a firm’s return would have higher correlation to returns of its own sector. However, we face the challenge of figuring out ‘sector returns’. We could consider all firms in a sector to calculate sector returns but this would cause us to mix training and testing data. Hence, this is done after we split the data as discussed in the next section.

Creating Training and Test returns (and correlations)

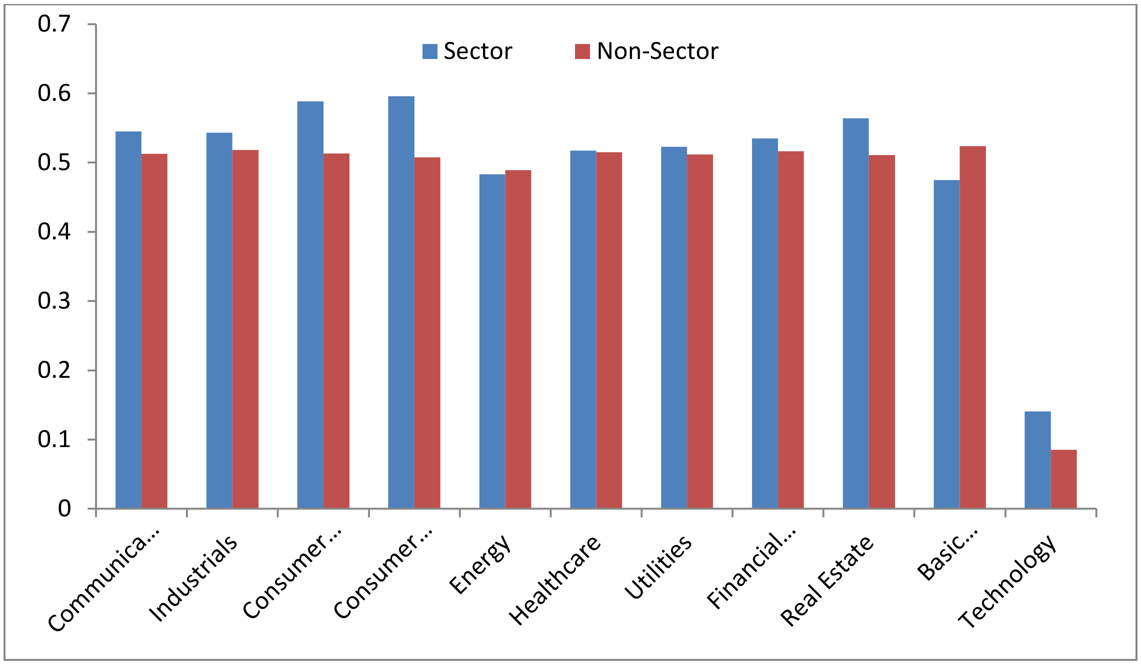
We split the full set of tickers into training and test data (75-25 split). There was a miniscule chance that our complete universe of sectors may not be available in training data but this was not observed. For each firm in training data, we calculated the correlation to all the sectors using firms in training data only. Specifically, the sector returns were calculated as simple arithmetic mean of returns of all constituent firms. To calculate correlation to its own sector, we removed the firm from the calculation (to eliminate bias).

Training data sector returns are defined as mean of returns of all firms in that sector in training data. For test data, 11 correlations, one for each sector, to training data sector returns were also calculated and added as features.

This analysis assumes that we have labeled training data available but test data is unlabeled. Hence, using correlation of test data returns to training data returns does not introduce bias in our model.

Data Visualization

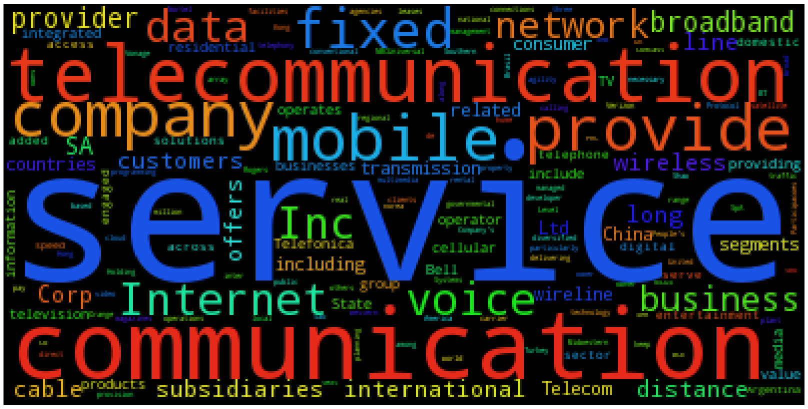
At this stage, let’s take a look at some of the relationships in our data (which is nearly ready to be modeled). Firstly, let’s take a look at how good is our assumption about correlation of firms’ returns to sectors. Below we see average of correlations of firms to their own sector as compared to other sectors using training data only.



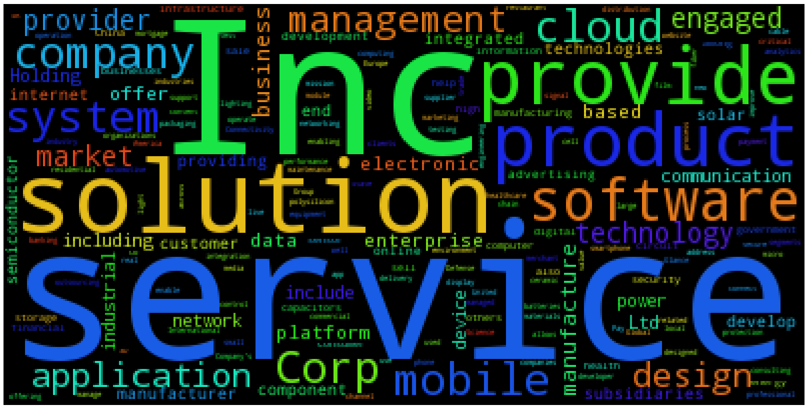
This data visualization is encouraging: On an average we do see that correlations of returns of a firm to its own sector are higher than its correlations to returns of other sectors.

Next, let’s see what the Summary looks for each of the sector. Here are word clouds of summary section for all firms in some of the sectors:

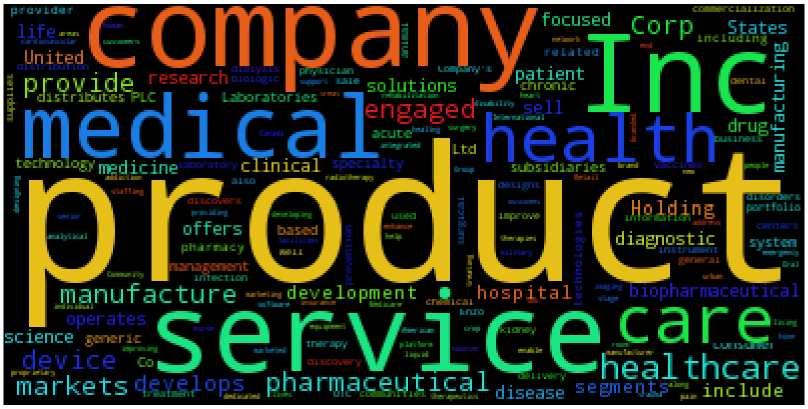
Sector: Communication Services:



Sector: Technology:



Sector: Healthcare:

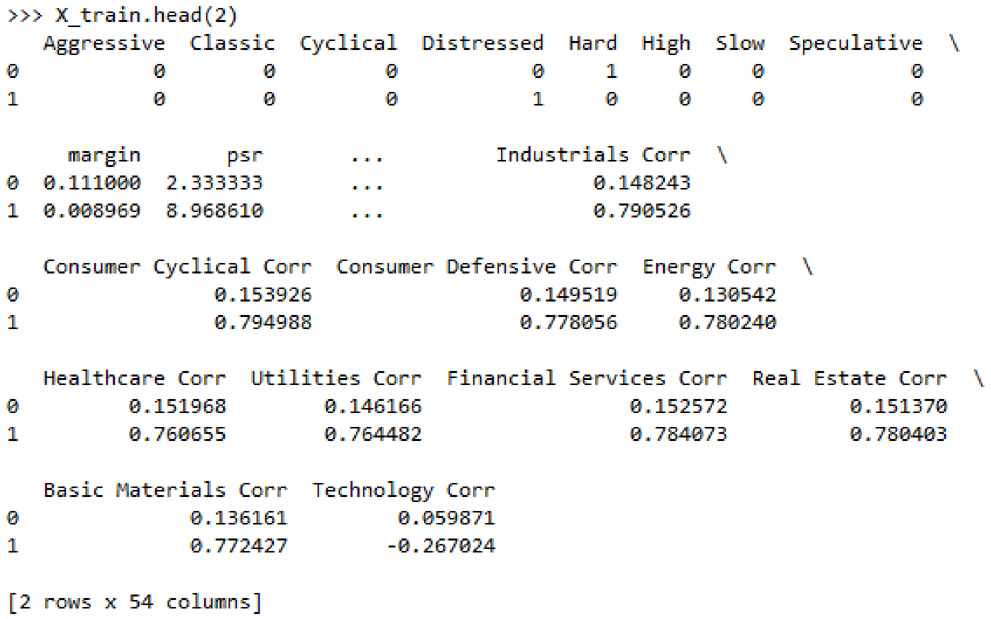


Create keyword counts

At this stage, we face a couple of options: we could manually decide some of the key words and then look for their occurrence in the Summary or we could simply pick most commonly used words across summaries. To ensure robustness and validity of model in similar domains, we chose to pick top 30 most frequent words across all summaries in training data. It is likely that we could have generated much better discriminatory power in our model by using some domain knowledge here but that would have reduced the cross-applications of this model.

Preprocessing clean data

At this stage, we have clean, usable training and test data with appropriate features. Let's take a quick look at how our data looks like now:



```
>>> X_train.head(2)
   Aggressive  Classic  Cyclical  Distressed  Hard  High  Slow  Speculative  \
0           0        0         0           0     1     0     0            0
1           0        0         0           1     0     0     0            0

     margin       psr       ...         Industrials Corr  \
0  0.111000  2.333333       ...                 0.148243
1  0.008969  8.968610       ...                 0.790526

   Consumer Cyclical Corr  Consumer Defensive Corr  Energy Corr  \
0                0.153926                 0.149519     0.130542
1                0.794988                 0.778056     0.780240

   Healthcare Corr  Utilities Corr  Financial Services Corr  Real Estate Corr  \
0         0.151968        0.146166                 0.152572          0.151370
1         0.760655        0.764482                 0.784073          0.780403

   Basic Materials Corr  Technology Corr
0              0.136161         0.059871
1              0.772427        -0.267024

[2 rows x 54 columns]
```

We face two challenges at this stage. Some of our data is missing. Furthermore, our data is not scaled which could cause an issue with potential classifiers (such as SVCs).

We replaced missing values for each feature by median values of that feature using training data for both training and test data. We used the Imputer functionality available in preprocessing module of sklearn package.

Finally, we scaled both the training and test data using scale functionality of preprocessing module, i.e. for each feature we centered it to zero mean and unit variance.

Training Models

We trained 3 types of classifiers. A brief overview and their accuracy scores are as follows:

1) Decision Trees[7]: A decision tree classifier aims to create simple rules combination of which is used to describe the data. These rules are typically of if-then-else form, with each condition comparing value of a feature with a fixed value. Decision trees would be highly preferred (with reasonable scores) for our model since they are very easy to explain, even to laypersons. Unfortunately, our initial decision tree doesn't perform sufficiently well to warrant further investigation; the accuracy score our decision tree was: 0.42. While this is above our breakeven score of 0.2, as we will see, other models perform better.
2) Gaussian Naïve Bayes[8]: A Naïve Bayes classifier tries to calculate bayesian probability of each possible classification based on the simplifying assumption that each feature is pairwise independent of all others. The model then selects the classification with highest probability as the prediction. This model retains the advantage of being easy to explain. Most finance professionals are reasonably familiar with concept of Bayesian probability and concept of independent variables. Gaussian Naïve Bayes assumes that our features have a Gaussian distribution. This is not really a very good assumption as most of our features are dummies or word counts). An alternate would have been to consider Multinomial Naïve Bayes which assumes the features to be multinomially distributed, i.e. features need to be non-negative which is again not a very good description for correlation data (which ranges from -1 to 1). Again, Gaussian NB doesn't perform sufficiently well to warrant further investigation. The accuracy score of our Gaussian Naïve Bayes was: 0.17.
3) Linear Support Vector Machine[9]: An SVM models tries to construct a set of theoretical hyperplanes in which the pair-wise distance between nearest training data point of each class to each of others is maximized. Specifically, a linear SVM model only constructs the hyperplane using a linear combination of features. We face some obvious issues with SVMs: They are a black box model and not very intuitive without a concept of n-feature data existing in n-dimensions and the idea of data-transformation into theoretical hyperplanes. However, they perform very well in our case, with an accuracy score of: 0.57.

Due to the high score, we chose Linear SVC as our model of interest and explored further refinements.

Refinements

Firstly, let's see if we can improve our features in some way. We selected correlation to each sector as one of our features. This choice faces an obvious business logic issue: it is unfair to simply claim that higher correlation to a sector's returns implies higher possibility of belonging to that sector; this claim doesn't consider the *a priori* correlations (and probabilities). E.g. some firms may have high correlations to all sectors (and vice versa).

Instead, consider the alternate claim: A firm is more likely to belong to the sector to whose returns it has a higher correlation as compared to its (firm's) correlation to other sector's returns. This claim likely has sounder business logic as it takes into account *a priori* correlations.

[7] http://scikit-learn.org/stable/modules/tree.html
[8] http://scikit-learn.org/stable/modules/naive_bayes.html
[9] http://scikit-learn.org/stable/modules/svm.html#svm-classification

To accommodate this domain knowledge, we ran the SVM model by replacing the actual correlations with rank of correlations (among sectors). i.e. we removed the 11 features related to correlation of returns with sector and instead added 11 features, denoting the rank of the correlation (the sector with highest correlation got rank 1). With this data, our accuracy score actually went down to 0.52. This is slightly unexpected until we realize that by replacing correlations with ranks, we may be reducing information in our model: the actual value of the correlation.

We then added back that information and evaluated the model. We got an accuracy score of 0.49. The lower value is due to the regularization parameter, C. We could fine tune C to improve our score but at this stage, it seems likely that this approach may not improve the model performance easily.

Instead, we take the original data set (with correlations) and search for best regularization parameter. We tried 20 parameters, spaced evenly in the log space between log-1 and log1 (i.e. between 0.1 and 10). This helps us determine if an alternate choice of 'C' may have been better. We need to careful about over fitting the data. Specifically, if improvement in score is small, it may be best to ignore the discovered value of 'C' to prevent over fitting.

In this case, we actually don't discover a much better value of 'C'. The best fit model gives an accuracy score of 0.58 as well, with 'C' value of 0.43. This seems to suggest that our model is remarkably robust: changing 'C' value doesn't significantly impact the accuracy of the model.

We decided to use the original SVM model as our final model, i.e. no refinements were accepted.

Results

Model Validation

Firstly, let's consider the selected classifier: Linear SVM. Our model has many different kinds of features: correlations (which range from -1 to 1), dummies (which are either 0 or 1), financial ratios (which can range from –inf to inf) etc. By pre-processing our data, we reduced scaled our features to have zero mean and unit variance, however, we didn't change the underlying distribution of these features. SVMs do not make any assumption about the underlying distributions of the features and hence are prima facie, more robust than, e.g. Naïve Bayes models.

However, let's consider how sensitive our model is to regularization parameter. For 'C' value from 0.1 to 10, the accuracy score ranged from 0.54 to 0.58. The narrow range of accuracy score strongly indicates that underlying model is a good fit for the data. Our final selection of default 'C' value of 1 further validates the robustness of the model. It significantly reduces the probability that our model has been over fitted on the data.

Let's also try some cross validation; we changed the test-train split by changing the random state from 42 to 99. By fitting the same model, we obtained a score of 0.54. This further seems to validate the robustness of our model to changes in data.

Based on this analysis, we accept this model for predicting sector of a firm.

Evaluation and Justification

Given 11 potential sectors, an absolutely random model would achieve an accuracy of approx. 9%. In the benchmark section, we discussed how a model with 20% accuracy would breakeven in terms of marginal time spent on review and higher accuracy would be an improvement. Our model shows an accuracy of approx. 58%. This is a significant improvement over current methodology. In addition, even for firms that are mislabeled by the model, an analyst learns something potentially useful particularly if the firm shows high correlation to other sectors.

Conclusion

Important qualities of model

By selecting a model (SVC) which doesn't make assumptions about the distribution of features, we have made it relatively easy to add features to the framework in the future. There are many potential financial ratios and metrics available for each of the firms which may be of potential interest when expanding the scope of the model.

Reflections

The initial problem statement: 'predicting the sector of a firm' doesn't present itself with obvious choice of features. One of the main challenges of this model was 'feature engineering', i.e. creating features which may be useful in making the predictions.

As is common in financial world, the data itself required significant effort to obtain. While this implementation chose Selenium library to scrape the data from a website, in some cases, paid solutions may be available which may significantly reduce this effort.

Also, as is common in financial world, the data obtained required significant cleaning to use in a model. Specifically, there were missing values, incorrect formats (strings instead of numbers) and incompatible data values (such as inf). This problem is likely not as severe in other areas where data generation may be from automated processes, reducing the likelihood of dirty data.

Finally, our data set didn't allow us to use an out-of-the-box solution to fit an appropriate Naïve Bayes model. Our feature data was neither all Gaussian nor all multinomial. An appropriate solution would be to construct a customized Bayesian model which makes appropriate assumptions (Gaussian or multinomial) for each of our features. While construction of such a model is interesting, it is beyond the scope of this project.

Further improvements

There were many assumptions made during the development of this model that may be improved with further analysis.

1) Choice of only using top 30 words across summaries: Improving this choice is possibly the easiest way to improve the model. The current choice faces the following issues:

   a. Most of the top 30 words have no discriminatory power, they occur frequently in firm summaries across sectors. An easy way to solve this problem would be to use more words and then use feature reduction. Another way would be to use domain knowledge to eliminate words which are not expected discriminate among sectors.
   b. Ignores actual keywords: In some cases, we can see sector name occurring in the summary. This model ignore such occurrences as they are not likely available in actual financial data (and hence, including them would reduce future applications of the model)

2) Choice of using correlation rank (or) actual correlation: This model uses rank of correlation among all sector correlations as metric. We also tried using actual correlation. Both these choices face issues. In case of using rank, we are implicitly telling a linear model that, in absence of any other information, a sector with rank of 4 is 1/2 as likely as a sector with rank 2 but this is likely false. Further analysis on assigning correct feature value based on correlation would also help in significantly improving this model.