## Project Report

### Predicting Sector of a firm

### Domain Background

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 to 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.

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]](#footnote-1) to evaluate the goodness of our model. This score reflects the proportion of accurate labeled firms (to the total number of available firms).

### 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]](#footnote-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:
   1. Summary
   2. Stock Type
   3. Market Cap
   4. Net Income
   5. Sales
   6. Employees

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

1. Returns data: For each of the 2,065 tickers, we downloaded the daily price data using yahoo\_finance python package[[6]](#footnote-6). We converted the price data into daily returns to use in our calculations.

#### Cleaning data

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, correlation to training data sector returns was also calculated and added as features.

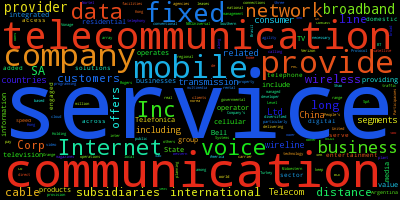
This analysis assumes that we have labeled training data available but test data is unlabeled. Hence, using correlation of test data returns to training data returns does not introduce bias in our model.

#### Data Visualization

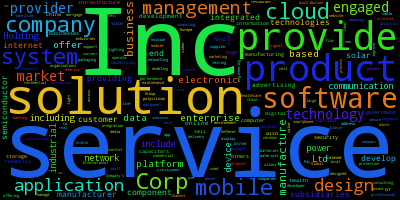
At this stage, let’s take a look at some of the relationships in our data (which is nearly ready to be modeled).

Firstly, let’s 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:

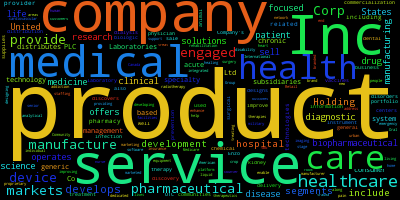
**Sector: Communication Services**:



**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. We face two challenges at this stage. Some of our data is missing. Furthermore, our data is not scaled (this is not a big problem, but fixing it will make the effect of features comparable).

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 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]](#footnote-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 reasonably 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:
2. Naïve Bayes[[8]](#footnote-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. We tried two forms of Naïve Bayes classifier due to nature of our dataset: Gaussian Naïve Bayes and Multinomial Naïve Bayes
3. Linear Support Vector Machine[[9]](#footnote-9):

***Refinements***

Using rank instead of correlations.

Using GridSearch

### Results

***Model evaluation and Validation***

Blah blah blah

***Justification***

Given 11 potential sectors, an absolutely random model would achieve an accuracy of approx. 9%. In our proposal, 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. 35%. 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***

***Reflection***

***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:
   1. 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.
   2. 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.

1. http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy\_score.html [↑](#footnote-ref-1)
2. http://www.nasdaq.com/screening/companies-by-industry.aspx?exchange=NYSE [↑](#footnote-ref-2)
3. http://financials.morningstar.com/ [↑](#footnote-ref-3)
4. http://selenium-python.readthedocs.io/index.html [↑](#footnote-ref-4)
5. https://github.com/mozilla/geckodriver/releases/tag/v0.11.1 [↑](#footnote-ref-5)
6. https://pypi.python.org/pypi/yahoo-finance [↑](#footnote-ref-6)
7. http://scikit-learn.org/stable/modules/tree.html [↑](#footnote-ref-7)
8. http://scikit-learn.org/stable/modules/naive\_bayes.html [↑](#footnote-ref-8)
9. http://scikit-learn.org/stable/modules/svm.html#svm-classification [↑](#footnote-ref-9)