Project Report

Predicting Sector of a Firm

1 Overview

This problem comes from the world of finance. For financial analysis of a company, it is useful to assign the company to an appropriate sector (i.e. its peers). This assignment helps financial analysts better understand various metrics of the company; 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.

1.1 Problem Statement

Identify the correct sector of a firm based on its returns, a brief description and other information about the firm.

Let's define what 'correct sector' means. There isn't an objectively true sector assignment. For the purpose of this exercise, we will assume sector classifications available in labeled data are correct. From business perspective, we assume sector classifications made by analysts/companies are correct.

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. For a PM/analyst working in developing countries or with non-public firms, there are thousands of assignments to be made. Even assuming 100% accuracy, this is a time consuming process.

A good model is expected to save valuable time of PMs and analysts (by automating part of their work). A good model is also expected to serve another purpose: it provides an insight into the business model of the firm even if the PM/analyst eventually disagrees with sector chosen.

To build this model, we will require some data about the firm. Specifically, we will use data that is easily available to an analyst: some financial metrics about the firm and the returns of the firm. From this data, we will build a multi-class classification model which will assign sector labels to each of the firms. The model will learn from a labeled training data (presumably from work already done). The final model

should have the ability to predict the sector of a firm based on the data and should be robust enough to handle partially missing data.

1.2 Evaluation Metric

Requisite properties of evaluation metric:

1) Must be simple to explain to laypersons: We need to convince PMs and analysts to use our model and it would be helpful if they can fully understand the benefit of our model
2) Doesn't distinguish between labels: i.e. doesn't have a concept for False Positives/False Negatives (only right/wrong) or have a separate score based on label or a specific weight for a label.
3) Doesn't penalize/rewards closeness of match: Based on discussion in the previous segment, the punishment for wrong label is the equal in all cases, even if the wrong label is close to the correct label in any sense.

We choose a metric that satisfies all these requirements to evaluate the goodness of our model: Accuracy score[1]. This score reflects the proportion of accurately labeled firms to the total number of available firms (with labels).

1.3 Benchmark

As discussed, there isn't a machine learning benchmark for this problem. Let's benchmark against the present situation. Currently it takes around 10 minutes per classification. In contrast, based on interviews, 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 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.

2 Design and Analysis

2.1 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

[1] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
[2] http://www.nasdaq.com/screening/companies-by-industry.aspx?exchange=NYSE

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.

2.2 Data Exploration

Let's see what the raw fundamental data looks like. We look at 3 out of total 2,065 firms.

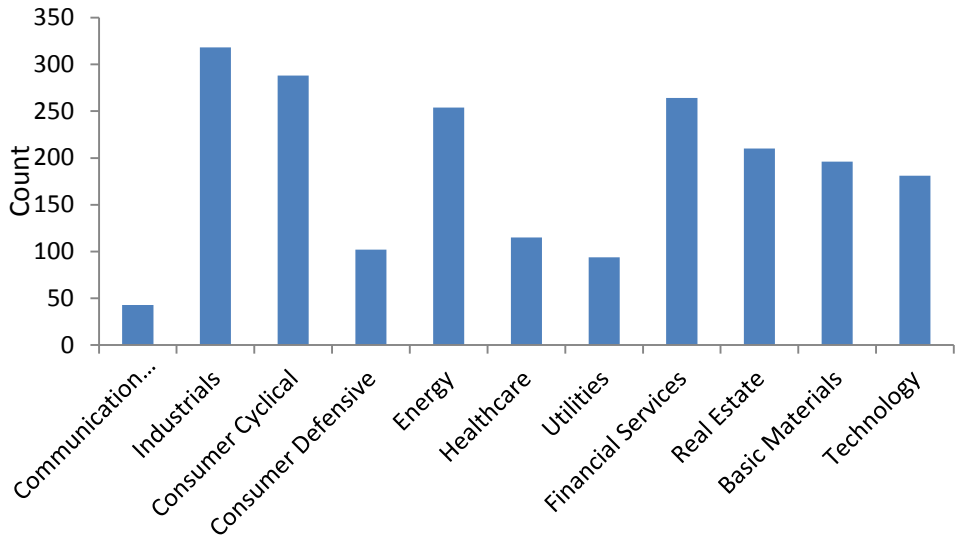
```
>>> fund_data.head(3)
  Unnamed: 0                          Name  \
0        DDD              3D Systems Corp
1        MMM                       3M Co
2       WUBA  58.com Inc ADR repr Class A

                                      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
>>>
```

[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

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.

Let's also look at how our firms are divided among our sectors:



This visualization provides an intuition about sector-wise performance of our model (which is one of the minor goals of the model). A model may be very good without specifically being good in sectors with few firms such as 'Communication Services' and 'Utilities'.

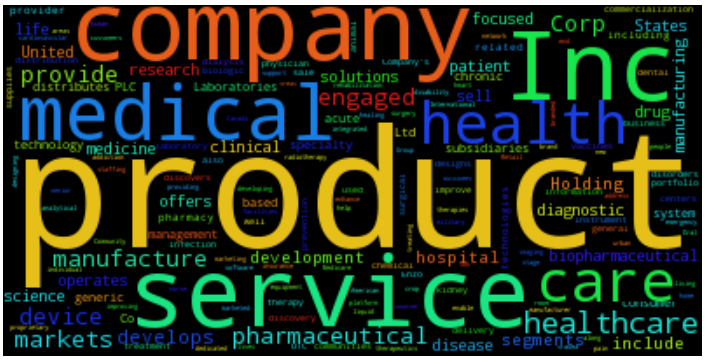
2.3 Exploratory Visualization

First, let's see what the 'Summary' section 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:

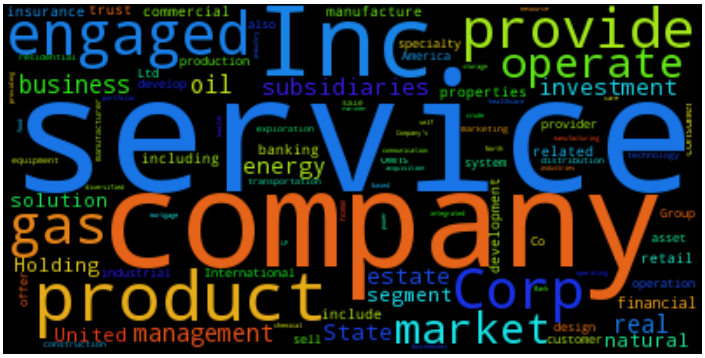
Sector: Healthcare:



Let's consider some potential features we can generate from this data:

1) We could take top 10 (or so) keywords from each sector's summary and add their word counts as features. This solution is tempting but not very scalable. In datasets where number of sectors is a significant proportion, say 5% or so, of the total number of firms, the number of features would become huge and require feature selection/reduction implementation.
2) As we notice some overlap between top keywords from different sectors, we could simply take top 30 (or so) keywords from all sectors' summary and add their word counts as features.

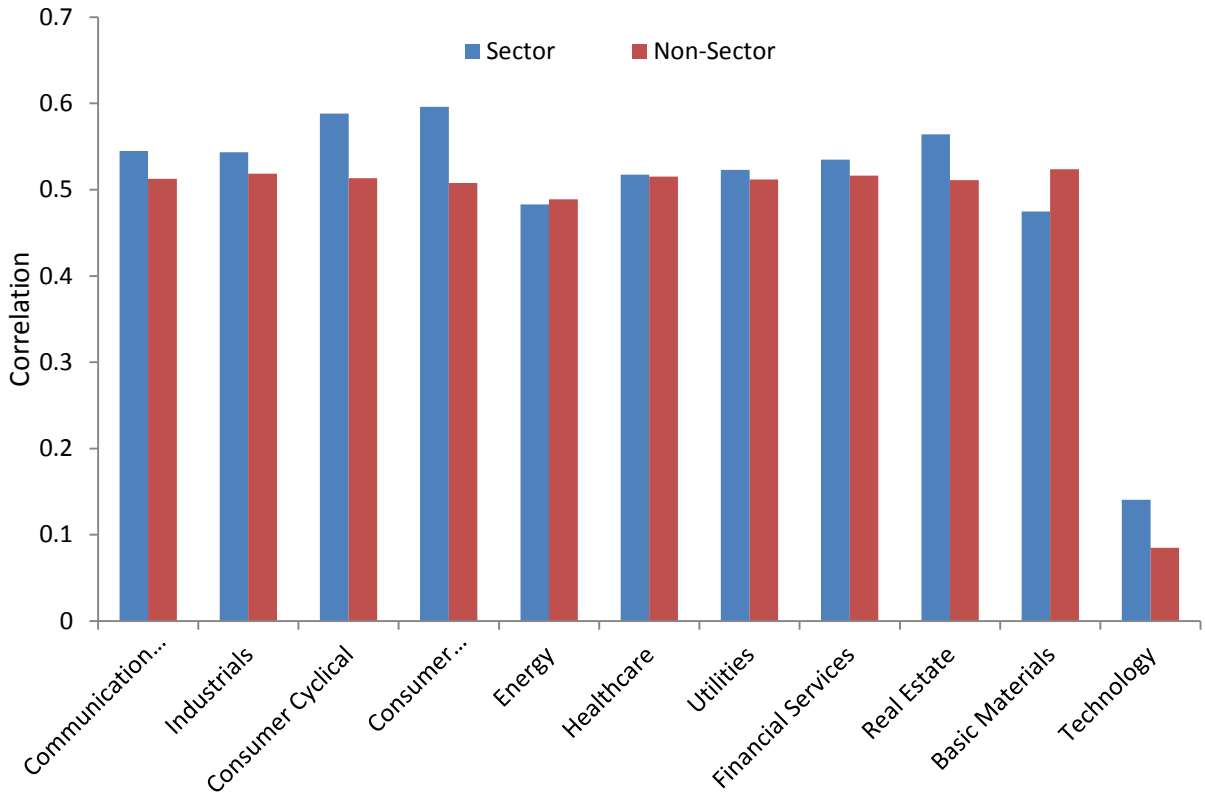
Let's first look at how the wordcloud for all Summary sections looks like:



At first glance it seems like this choice would significantly reduce the discriminatory power of any potential model: most common words (such as Inc., company, corp, market) likely do not occur any less/more frequently in any specific sector. However, beyond some of the most common words, we can see words which we expect to have significantly different frequencies in different sector (such as service, gas, oil, energy, real etc.)

An option may be to simply discard the top 5 (or so) common keywords and keep the next 30 (or so) keywords in our model. For this version of our model, we chose to go with the simpler option: word count of top 30 keywords in all Summary sections.

Next, let's consider another business concept: a firm belonging to a sector would perform similar to the performance of the sector in the market. We use this concept as 'correlation of returns' in our model. Specifically, let's look at correlation of returns of a firm with the returns of the sector. Sector returns are calculated as arithmetic average of returns (for simplicity; a better solution may be to consider market cap weighted returns but it is not expected to add much value) of all firms in the sector. Specifically we excluded the firm from its own sector when calculating correlation to sector to eliminate potential bias. Below we see average of correlations of firms to their own sector as compared to other sectors.



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.

2.4 Techniques: Creating features from data

In last section, we explored potentially interesting ideas about our data. In this section, we will discuss various features we created from those relationships.

2.4.1 Features from fundamental data

Instead of using raw data about, e.g. Net Income, 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. Specifically, there were 8 unique stock type values in our data, each of which became a feature (note that in some cases, no stock types were available).

Before we create features from correlations and word counts (as discussed in previous section), we need to be careful about not using our 'test' data in any way for the model. Hence, at this stage, we divide our data into test and training sets.

2.4.2 Techniques: Test-Train Split

We split the full set of tickers into training and test data (75-25 split) using out-of-the-box randomizer (train_test_split) in cross_validation module. There was a miniscule chance that our complete universe of sectors may not be available in training data but this was not observed.

2.4.3 Creating sector correlations features

For each firm in training data, we calculated the correlation to each of 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 of sector returns (to eliminate bias).

For test data, 11 correlations, one for each sector, to training data sector returns were also calculated and added as features.

2.4.4 Creating features from keyword counts

As discussed previously, we constructed 30 new features, one each for top 30 most common keywords in the Summary section of training data. Each feature represents the number of the times that keyword occurs in the Summary section of the firm.

3 Methodology

3.1 Data Preprocessing: Filling in the gaps

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.

3.2 Implementation

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.

[7] http://scikit-learn.org/stable/modules/tree.html

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]: 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, a soft-margin Linear SVM classifier performs very well right out-of-the-box in our case, with an accuracy score of 0.58. In next section, we will discuss this algorithm in details.

For each of these algorithms, we used out-of-the-box classifiers available in the scikit-learn package using default parameters.

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

3.3 Implemented Algorithm: Soft-Margin Linear Support Vector Classifier

An SVM model allows for many flavors each with its specific implementation and use cases. In this section, we reserve our discussion to the model implemented.

A Linear SVC model considers each data point (row in our dataset or values of features for 1 firm) in a p-dimensional space (where p is the number of features) and then tries to construct a set of theoretical hyperplanes (of p - 1 dimensions) in which the pairwise distance between nearest training data point of each class to nearest training data point of all other classes (together) is maximized. This results in training of n models (number of classes, in our case the 11 unique sectors). Note that this is different from training a model for each pair of classes which would have constructed n x (n-1)/2 models. The chosen classification for each data point is the one from the model which had the best class-score for that data point (i.e. highest probability of belonging to that class). The 'linear' in the name refers to the fact that in the p-dimensional space, the separating margins are linear.

We could have also chosen a non-linear SVM classifier which would have considered the data in a higher dimensional space (thus allowing for linear separation using hyperplanes in the higher dimension space which would be a non-linear separation in p-dimensional space). However, without stronger data

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

exploration fitting a more complex model doesn't necessarily provide better insights. Given the large number of features in our model (54), it is not clear if an even higher dimension space would necessarily provide a better result. Nor would such a model be easily explainable to end-users, reducing its utility.

Soft-Margin: Since our training data is not necessarily completely separable linearly, we use a soft-margin, i.e. we penalize incorrect classification while maintaining as large a distance as possible between pairwise nearest training points (called the margin). This is a trade-off: if we choose larger margins, we will have more misclassified points. If we choose fewer misclassifications, we will have a smaller margin. We balance this trade-off with 'C'[10] parameter in our implementation. A small value of C reduces the penalty for misclassification, thus increasing our margin (and hence, making our model less prone to over-fitting) while a larger value of 'C' assigns a higher penalty for misclassification, thus making our margins thinner.

Penalty for misclassification: Generally called the 'loss-function', SVM implementations allow for different loss functions. A popular loss function is hinge-loss which penalizes misclassified data points linearly proportional to their distance from the margin. The default version of Linear SVC implementation uses square of the hinge-loss function as the penalty for misclassification which is also used in our model.

3.4 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.

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.

[10] https://en.wikipedia.org/wiki/Support_vector_machine#Soft-margin: 'C' value is 1/λ as discussed on this page

Instead, we take the original data set (with correlations) and search for best regularization parameter. We tried 20 parameters, spaced 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. As discussed above, choice of 'C' can make the model more (or less) likely to be over fitted, so if changing 'C' doesn't significantly improve our model, it may be best to accept the default values.

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.

There are other potential refinements to consider in this model; e.g. in previous section we discussed an alternate loss function: hinge-loss. Another potential refinement may be to discover the optimization of the bias term in the final model, e.g. by exploring the bias scaling parameter. However, none of these refinements are *prima facie* expected to generate significantly improved results.

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

4 Results

4.1 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.

Due to the time-consuming nature of our feature creation process (around 30 minutes per run) which is unique to each test-train split, we are unable to do meaningful cross-validation. However, we did try 1-fold 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.

4.2 Evaluation and Justification

Given 11 potential sectors, an absolutely random model would achieve an accuracy of approx. 9%. A slightly better random model which assigns all firms to most common sector ('Industrials') would achieve an accuracy of approx. 15%. 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.

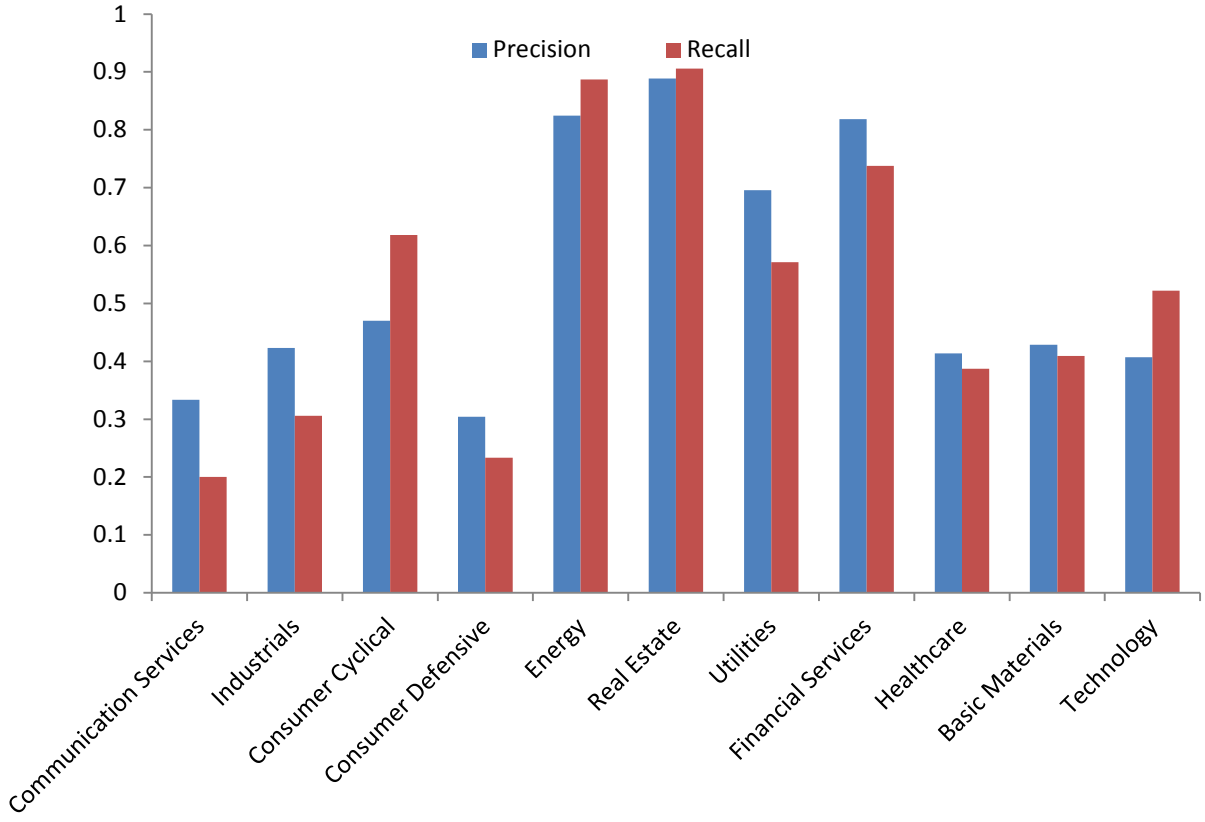
5 Conclusion

5.1 Model Quality/Free Form Visualization

One of the ideas we discussed previously was our indifference between different sector labels. Let's see how well our model does on this metric. We measure the label-wise precision and recall of our model on test data.

For each sector, precision is the ratio of number of firms correctly assigned that sector by the model to total number of firms assigned that sector by the model.

For each sector, recall is the ratio of number of firms correctly assigned that sector by the model to total number of firms belonging to that sector.



Firstly, in no sector does our model perform worse than the benchmark. Secondly, this visualization helps the analyst/PM identify the weaknesses of our model: only around 30% of firms classified by our model as 'Communication Services' or 'Consumer Defensive' actually belong to that sector. This allows the analyst to appropriately allocate manual effort in correcting the classifications. Finally, it helps the analyst make other subjective judgments: e.g. if the analyst's aim is to shortlist all firms belonging to a particular sector, the model doesn't do a very good job in identifying 'Communication Services' and

'Consumer Defensive' (only around 20% in both cases). However, neither of these issues is a major one; these two sectors also have among the fewest firms assigned to them (43 and 102 respectively).

5.2 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.

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.

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.

5.3 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. Some 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. Alternatively, we could top keywords from each sector.
    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 correlation of returns to each of the sectors as metric. We also tried using rank of correlation (among sectors). Both these choices face issues. Further analysis on assigning correct feature value based on correlation would also help in significantly improving this model.