# Predicting Startup Outcomes: Failure, Acquisition or IPO

Krishn Ramesh
Department of
Systems Design Engineering
University of Waterloo
Waterloo, Canada
Student Number: 20521942
Email: kramesh@uwaterloo.ca

Siddhanth Unnithan
Department of
Systems Design Engineering
University of Waterloo
Waterloo, Canada
Student Number: 20523022
Email: suunnith@uwaterloo.ca

Ramandeep Farmaha
Department of
Systems Design Engineering
University of Waterloo
Waterloo, Canada
Student Number: 20516974
Email: rfarmaha@uwaterloo.ca

***Abstract*—Predicting if a company will succeed or fail is of great interest to startup founders, employees, investors, venture capitalists, and governments. This paper's primary objective is predicting whether a startup will be acquired, file for an Initial Public Offering (IPO), or fail by not reaching the aforementioned milestones. The 2013 Crunchbase dataset is used to aggregate a list of basic and financial features for thousands of companies. This dataset is augmented with company mentions in TechCrunch articles since 2012. The resulting dataset and features are used to train three classifiers: Random Forest, Support Vector Machine (SVM), and Naive Bayes. The Random Forest classifier performs the best, with an overall precision of 0.81 and overall recall of 0.82. All three classifiers perform relatively poorly in predicting if a company will fail or file for an IPO, largely due to the smaller sample size in the Crunchbase dataset for those classes relative to acquired companies. Improvements to the limitations of this study include: aggregating company data from multiple sources to reduce the sparsity of the data, and running more sophisticated versions of the classifiers to better improve the training and prediction processes.**

***Keywords*—*startups, companies, VC, investments, acquisition, failure, success, prediction.***

## I. Introduction & Background

Entrepreneurship and capital are two of the fundamental factors of economic production, and they come together constructively in the form of new companies, more commonly referred to as startups. Startups are inherently risky for everyone involved; founders dedicate their lives to realizing a vision, investors and venture capitalists make speculative bets, early employees make career shifts, and governments redirect public money with the hopes of fostering economic growth. The statistics too paint a dismal picture: more than 90% of startups ultimately fail [1] for a myriad of different reasons including scaling prematurely, running out of cash or lacking market need. Nevertheless, startups are being formed at record levels, inspired by the successes of billion-dollar companies such as Google and Amazon. High-tech firm births were 69% higher in 2011 compared with 1980, and were 210% higher for the Information Technology sector specifically during the same time period [2]. New and young businesses are now the primary source of net new jobs [2]. Evidently, high-tech startups are essential for a healthy economy.

There are only two outcomes for a startup: failure or an exit. This paper is primarily focused on exits, which are the successful scenario. The two main exit strategies are an acquisition or an initial public offering (IPO). An acquisition (or merger) is a situation in which a larger company buys or merges with a smaller one, usually for the product/market share or for the talent (this is colloquially known as an acqui-hire). An IPO is when a company offers the public its shares on the stock market to raise large amounts of capital. This is the path that the largest and most powerful companies have followed. Both these exit scenarios allow someone with equity in the company (founders, investors, employees) to sell their stake for a profit since the public or private valuation of the company is usually larger at the time of exit than when the equity was acquired.

Founders, investors, employees and governments all have a vested interest in predicting if a startup will have a successful exit. This is the primary objective of this paper: classifying whether a startup will successfully get acquired, go public by filing for an IPO, or fail by achieving neither of the exits. To achieve this goal, a dataset of startup data is compiled that includes details on the company, investments and founders. This data is augmented with public finance data, technology news and trends. Machine Learning classification techniques are used to find patterns in the underlying data and ultimately predict the success of a startup.

## II. Related Work

The explanatory factors of corporate success and failure are a heavily studied topic in business, management, economics and finance departments in universities around the world. Most of the literature is either heuristic-based (analyzing case studies and experimental samples) or model-based (building statistical models with quantitative financial data).

The first theme of related work explores the heuristic factors of a company that lead to success. Gelderen, Thurik & Bosma [3] followed 517 entrepreneurs for 3 years to conclude that high ambition, experience and perceived risk of the market were the greatest predictors of startup success. A model developed by Roure and Keeley [4] finds that completeness of the founding team, technical superiority of the product, expected time for product development, and buyer concentration explain the variance for success in their data set.

Secondly, bankruptcy prediction in particular is a deeply researched topic with lots of papers. Beaver's highly cited 1966 paper [5] posits financial ratios (e.g. cash flow/total debt, net income/total assets etc.) as a predictor of failure after a thorough analysis of failed companies. Beaver accurately predicted the failure of 78% of his sample companies five years early. Altman [6] and Deakin [7] built on Beaver's work in financial ratios, using discriminant analysis to achieve even higher prediction rate of 95% of failure within one year and 90% of failure within 3 years respectively. A number of modern papers in this area use machine learning techniques such as backpropogation neural networks, support vector machines (SVM) and genetic algorithms. These methods predicted bankruptcy in their own datasets with an accuracy of 92-97% [8], 67-75% [9] and 78.5% [10] respectively.

A third set of papers take a different approach and investigate successful investments and investors. A Stanford paper predicted investments using supervised random walks on a network of companies, people and investors up to an accuracy of 40% [11]. A Swiss paper investigates the business models of 181 startups from USA and Germany and uses SVM to classify their survival with 83.6% accuracy [12]. Another paper by Krishna et. al. [13] predicts startup success and failure using various machine learning methods and achieves a precision ranging from 73.3% to 96.3%.

Lastly, a few papers aim to predict mergers and acquisitions. A paper predicting corporate acquisitions using a rule inducing decision tree to make buy/sell decisions had a true positive rate of 81.3% and a false positive rate of 34.4% [14]. A closely related work to this paper comes from CMU, where Xiang et al. [15] predicted company acquisition rates using a Bayesian Network to achieve true positive rates of 79.8% and false positive rates of 8.3%.

Previous works had a problem with gaining access to large datasets to test the generalizability of their models. The sample sizes used in almost all the papers were fewer than a thousand, which were then further split into test and training sets. The samples were also often very limited by focusing only on specific industries, time periods or geographies.

This paper is novel in that it is a multi-class classification problem predicting not only failure, but also acquisition or a public exit. It also uses a larger dataset than has been typically used in all the related works mentioned. This paper aims to meet this objective by using modern ML algorithms such as Random Forest, SVM, and Naives Bayes, and compares their predictive ability.

## III. Material & Methods

### A. Datasets and Sources

Following the explosion of popular trends such as startups, crowd-sourcing and big data in the 2010s, a number of business intelligence data aggregators formed. One of the most comprehensive is Crunchbase, a graph database consisting of startups and investors, which has around 500,000 data points connecting companies, people (founders, employees etc.), funds, investments and major milestones. It is very popular, drawing 1 million user sessions per week and almost 20 million unique visitors to the site in 2014 [16]. Any registered user can make edits to the database, however a moderator reviews all changes before being accepted into the database. To ensure quality and validity, editors routinely review the data. As of 2014, Crunchbase had an annual unique contributor count of 80,000 [17].

Crunchbase provides the "Crunchbase 2013 Snapshot" ©2013 [18], which contains a replica of the database from December 2013. That includes all organization, people, and product profiles along with funding rounds, acquisitions, and IPOs. The data is provided as a `mysqldump`. A total of eleven tables are provided, listed in Table I.

TABLE I. Crunchbase Dataset

| Table Name | Count | Relevant Columns |
|---|---|---|
| `cb_objects` | 450,469 | `entity_type` (company or financial org.) `category_code` `status`, `region` |
| `cb_acquisitions` | 9562 | `acquired_object_id` `acquiring_object_id` `price_amount` |
| `cb_people` | 225,495 | `affiliation_name` `birthplace` |
| `cb_degrees` | 108,394 | `degree_type` `subject` `institution` |
| `cb_funding_rounds` | 51,819 | `funded_at` `funding_round_type` `raised_amount_usd` |
| `cb_funds` | 1,564 | `name` `raised_amount` |
| `cb_investments` | 80,822 | `funded_object_id` `investor_object_id` |
| `cb_ipos` | 1,259 | `valuation_amount` `public_at` `raised_amount_usd` |
| `cb_relationships` | 51,819 | `person_object_id` `relationship_object_id` `title` (CEO etc.) |
| `cb_milestones` | 39,340 | `description` |
| `cb_offices` | 51,819 | `region`, `city` |

There are a total of 196,553 companies, 11,652 financial organisations, 226,708 people and 27,738 products in the dataset. In terms of outcomes, 4.9% of companies have been acquired, 0.6% of companies have gone public in an IPO and 1.3% have closed down, with the remaining still operating. 16% of all companies have raised at least one round of funding.

The company data is augmented with news data from the reputable technology publication TechCrunch, which specializes in news about the startup ecosystem including all its companies, people, products and investors. The dataset, hosted on the popular data science competition website Kaggle, contains a list of 40,000 unique TechCrunch posts published before October 2016 [19] in `csv` format. The dataset contains the author(s), title, content, date, topics and more for each post.

### B. Feature Selection & Extraction

To begin with, the massive `cb_objects` table is broken up into its constituent companies, financial organizations and people. Next, the list of companies are filtered to only include

the ones whose status are closed, acquired or ipo; this reduces the list of relevant companies to 13,216. All the other tables are trimmed to only contain rows that pertain to the relevant 13,216 companies. In addition, irrelevant columns such as `created_at` and `updated_at` are dropped from all tables, leaving only relevant columns for feature selection. The code that does this is provided in Appendix A.

There are several quantitative and qualitative factors that determine and affect the success of startups. Similar to the paper by Xiang et al. [15], the features used in this paper to develop the predictive models are separated into key categories: basic company features, financial features, and finally article features derived from the TechCrunch database.

Basic company features include the foundational metrics of a company, namely: the age of a company in months, the number of employees, the country of origin and the category codes that the company falls within.

The financial features of each company are derived from the Crunchbase database, and thus are limited to publicly available data. A similar paper done by Krishna et. al. [13] focuses primarily on the financial data within the Crunchbase data dump, namely the total number of funding rounds, the total amount of funding, as well as the delta valuation, which is the difference in valuation from the first round to the most recent round.

This paper extends the financial feature set by taking a closer examination of the relationships and prior knowledge of investors and key members of the companies. These features include: the number of investors, the number of successful companies in investors' portfolios, and the number of successful companies associated with board members or c-level employees. A "successful company" in this context is defined as a company that has been acquired, has had an IPO, and/or is still active within the Crunchbase database. The key hypothesis being made here is that successful investors and higher level employees have ability to scout out companies with higher potential, and by virtue of their experiences are able to propel these companies to success.

Since TechCrunch is widely considered in the startup community as the most comprehensive news website covering the tech industry, one potentially valid indicator for a company's success could be the number of times it is mentioned in the TechCrunch article dataset. In order to achieve this, the `nltk` Python library was used to first extract all named proper nouns from each article in the dataset. Given a set of all pronouns for each article and a set of unique companies, an intersection operation was performed to obtain the overlapping set of company mentions per article. A detailed look into this procedure is shown in Algorithm 1.

To select only the minimal set of features, an inter-feature correlation matrix is calculated and visualized in Figure 1. The values in the Figure are representative of the Pearson Correlation Coefficient which is evaluated as the ratio of covariance to the standard deviation product for a feature pair.

The final list of all features is summarized in Table II. The implementation of derived features, such as the average time difference between a company's milestones and the average raised amount across all funding rounds, can be found in Appendix B. Redundant features, denoted with a correlation coefficient of 1 in Figure 1 were removed from the final feature set.

**Algorithm 1:** Extract Company Mention Frequency

**Input:** TechCrunch Articles $TC$, Companies $C$
**Output:** Dictionary, $DC$ of Company ID to Number of Mentions

```
for article ∈ TC do
    for sentence ∈ sentence_tokenize(article) do
        words = word_tokenize(sentence);
        tagged_words = get_POS_tags(words);
        for tagged_word ∈ tagged_words do
            if tagged_word.position is NNP then
                tagged_word_list += tagged_word;
    articles_tagged_words += tagged_word_list;
for company ∈ C do
    count = 0;
    for article ∈ articles_tagged_words do
        if company ∈ article then
            count++;
    DC += {company.id : count};
    return DC;
```

TABLE II. FULL LIST OF FEATURES

| # | Feature Name | Description |
|---|---|---|
| | **Basic** | |
| 1 | Category Code | The industry the company is in e.g. health. |
| 2 | Country Code | The country the company is in e.g. Canada. |
| 3 | # of Relationships | The number of recorded relationships between the company and individuals. |
| 4 | # of Offices | Total number of company offices. |
| 5 | # of Milestones | Total count of company milestones. |
| | **Financial** | |
| 6 | # of Rounds | Total number of funding rounds. |
| 7 | Total Amount Invested | Amount invested in other companies. |
| 8 | # of Investors | Sum of company, financial and individual investors. |
| 9 | Total Funding | Total funds raised normalized over active days |
| 10 | # of Investor Portfolio Successes | Sum of IPOs and acquisitions in investor portfolios. |
| 11 | # of Board Members | Total count of board members in the company |
| 12 | # of Invested Companies | Number of other companies this company has invested in. |
| | **Derived** | |
| 13 | Average Milestone Delta | Average length of time between milestones. |
| 14 | # of TechCrunch Mentions | Count of mentions of company names in TechCrunch article database |
| 15 | Average Raised Amount Delta | Average raised amount between funding rounds. |

### C. Data Processing

Prior to training each of the estimators, compatibility transformations are applied to the feature set. Zeros are imputed in place of missing values for continuous features, such as total

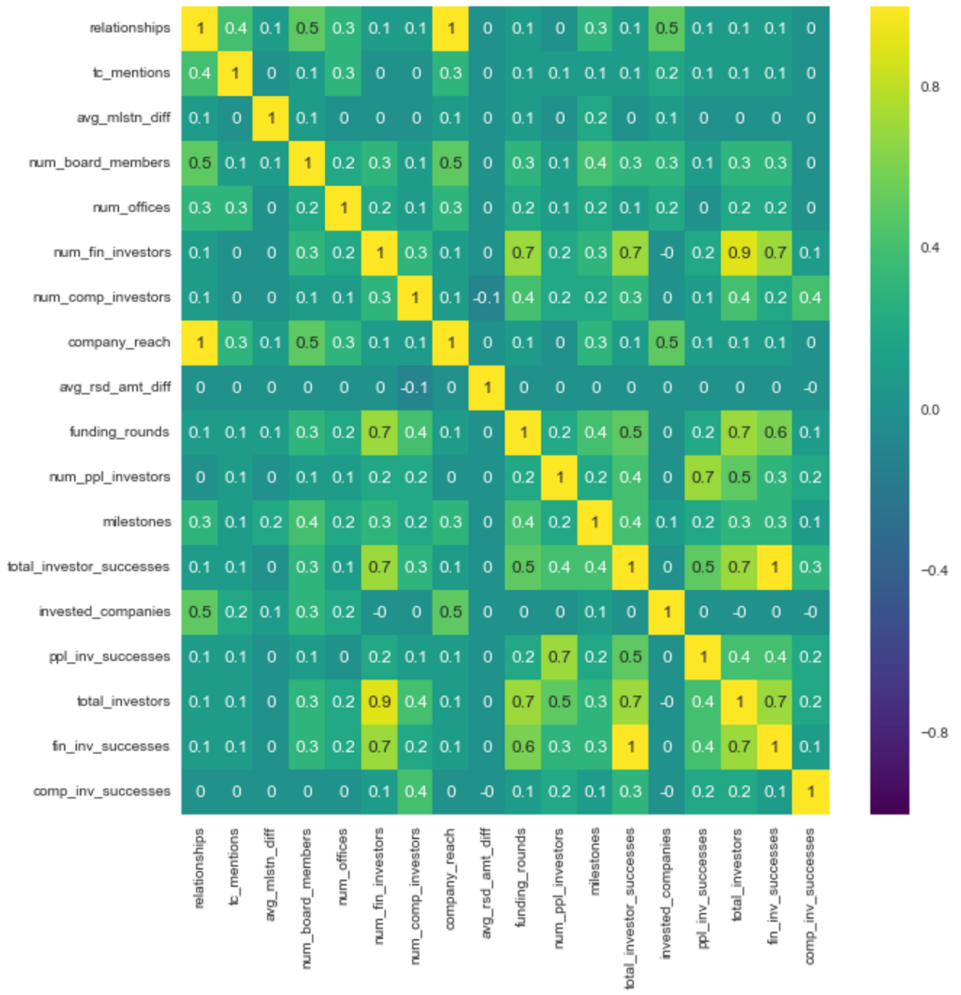


Fig. 1. Feature Correlation Plot

investment and number of funding rounds, with the assumption that the missing value is representative of a company that has not undertaken said actions. Importance is given to features in which a zero-value imputation is inappropriate; imputing a feature such as company valuation with zero is incorrect as a missing value is not indicative of a company with no valuation. Records with null values are filtered out of the data set for features in which imputation was inappropriate.

Categorical features are transformed, as the classifier implementations require strictly numeric values in the data set. Each discrete value for a categorical variable is transformed into a separate feature, with the values of the newly created feature being strictly binary digits. All records with a value of $x$ in the original feature, *cat_feat*, are assigned a value of 1 for the newly created feature denoted *cat_feat_x*. The described method of creating new binary features for each of the categorical variables is known as one-hot-encoding. The caveat of using the one-hot-encoding method for categorical features is the resulting volume of binary features. For a data set with $m$ categorical features, each with $n$ discrete categories, there requires an addition of $n * m$ features to the original set.

To ensure that the estimator is trained such that the observed performance propagates to newly obtained data points, the k-fold cross validation method is used. A helper function is written to execute the cross validation, allowing for dynamic specification of the estimator and the number of folds. Each fold is representative of $\frac{100}{k}\%$ of all records in the feature set; the k-fold method iterates $k$ times over the entire set, using the $k$th segment for testing and *k-1* segments for training. The passed estimator's default scoring method is used upon every iteration, with the final output being the average across all scores. An advantage of using the k-fold cross validation method is the irrelevance of the test-training split of the feature set as all records are used as test and training points across the entire pool of iterations. A drawback of using the k-fold method is the computational complexity associated with training the estimator $k$ times. Within the study, a single fold ($k = 1$) was used with a 70-30 train-test split of the feature set, as presented in Appendix C.

### D. Machine Learning Approaches

This is a multi-class classification problem in predicting one of three startup outcomes: failure, acquisition or IPO. As such, a number of different machine learning classifiers were considered. Neural networks or Multi-Layer Perceptrons would do the job quite well, but they are quite sophisticated to work

with and would have a tendency to overfit the data, since the labelled data does not number in at least the hundreds of thousands. Other classifiers considered include Random Forests, SVM and Naive Bayes. Rather than implementing the classifiers, the popular Python library `scikit-learn` is used for training and testing. Classifier instantiation and training is presented in Appendix C.

1) **Random forests** are an ensemble method for classification that involve training a multitude of decision trees. The final output class is obtained by taking the mode of all the outcomes of all the individual trees. Random forests have the advantage of correcting the tendency of decision trees to overfit their training set. They are also capable of handling a very large number of input features and highly imbalanced class distributions, while still providing accurate classifications.
2) **SVM** is a discriminative classifier that finds the *optimal* hyperplane to separate all the classes. If the classes are not linearly separable, a kernel can be used to map the features into a higher dimension where they might be better separable. SVM is only directly applicable to binary classification problems, but can be extended to multiple classes by training multiple *one-vs-one* or *one-vs-all* classifiers.
3) **Naive Bayes** classifiers are very simple probabilistic classifiers relying on the assumption of independence between features. They are very simple to implement and are surprisingly effective given their simplistic assumptions. For multi-class problems, the Naive Bayes classifier outputs the class with the highest posterior distribution.

### *E. Error Metrics*

Error metrics used for each of the classifiers include precision, recall, and f1-scores. The precision score measures the true positives against the sum of true positives and false positives for a given class (Equation 1). A precision score close to 1 is indicative of a classifier that has incorrectly labelled a small number of test points as belonging to the class of interest.

$$precision = \frac{TP}{TP + FP} \tag{1}$$

On the other hand, a recall score measures the true positives of a predicted class against the sum of true positives and false negatives (Equation 2). Incorrect labelling of a large number of test points as belonging to the negative class would result in a recall score close to 0.

$$recall = \frac{TP}{TP + FN} \tag{2}$$

The f1-score is a harmonic mean of the precision and recall.

$$f_1 score = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} \tag{3}$$

## IV. Results

The results for each of the 3 classifiers are summarized in tables III, IV and V. The first of the two tables presents an $3x3$ confusion matrix, in which index $ij$ is representative of the number of test points predicted to be in class *i* but belonging to class *j*. The gray cells along the diagonal are the number of startups classified correctly. The second table presents precision, recall, and f1-scores for each of the classes, where the class of interest is labelled positive and all other classes are grouped together and labelled negative. The support column refers to the number of occurrences of each class in the testing set. With the k-fold cross validation method, the support distributions are consistent across the training and test sets.

TABLE III. Error Metrics for Naive Bayes

| | **Classified** | | |
|---|---|---|---|
| **Truth** | Acquired | IPO | Closed |
| Acquired | 66 | 5 | 2761 |
| IPO | 44 | 9 | 234 |
| Closed | 26 | 2 | 787 |

| **Class** | **Precision** | **Recall** | **f1-score** | **support** |
|---|---|---|---|---|
| Acquired | 0.49 | 0.02 | 0.04 | 2832 |
| IPO | 0.56 | 0.03 | 0.06 | 287 |
| Closed | 0.21 | 0.97 | 0.34 | 815 |

TABLE IV. Error Metrics for SVM

| | **Classified** | | |
|---|---|---|---|
| **Truth** | Acquired | IPO | Closed |
| Acquired | 2597 | 54 | 181 |
| IPO | 207 | 71 | 9 |
| Closed | 479 | 11 | 325 |

| **Class** | **Precision** | **Recall** | **f1-score** | **support** |
|---|---|---|---|---|
| Acquired | 0.79 | 0.92 | 0.85 | 2832 |
| IPO | 0.52 | 0.25 | 0.34 | 287 |
| Closed | 0.63 | 0.40 | 0.49 | 815 |

TABLE V. Error Metrics for Random Forest

| | **Classified** | | |
|---|---|---|---|
| **Truth** | Acquired | IPO | Closed |
| Acquired | 2689 | 40 | 103 |
| IPO | 192 | 87 | 8 |
| Closed | 360 | 6 | 449 |

| **Class** | **Precision** | **Recall** | **f1-score** | **support** |
|---|---|---|---|---|
| Acquired | 0.83 | 0.95 | 0.89 | 2832 |
| IPO | 0.65 | 0.30 | 0.41 | 287 |
| Closed | 0.80 | 0.55 | 0.65 | 815 |

## V. Discussion & Conclusions

Based on the results displayed in the previous section, it can be concluded that the best to worst performing classifiers are Random Forest, SVM, and Naive Bayes. The poor performance of the Naive Bayes classifier can be attributed to the preliminary assumption that all features are independent of one another. From Figure 1 it can be seen that most features are positive correlated with one another, thus reinforcing that the

independence assumption is inappropriate. Similarly, the SVM classifier makes the assumption that pairs of classes are linearly separable. Different kernels were not used in this study due to the amount of time required for the underlying optimization to converge. On the other hand, the Random Forest classifier was successful in capturing most of the intrinsic, non-linear relationships between the features, without over-fitting the training set.

The performance of the classifiers showcases that it is incorrect to assume that features are independent and thus uncorrelated with one another. Through a Random Forest implementation with a sufficient number of decision trees, strong predictors are obtained for the multiple classes in the form of individual and combined features. Furthermore, the best performing class across all 3 classifiers was the Acquired class. This is due to how imbalanced the dataset is, given that the Acquired class has more than 2 times as many support points than the other two classes. The number of support points is found to be correlated to class performance. Further study is required to determine how sampling methods, such as undersampling the over-represented class, would affect classifier performance.

Two other related works mentioned in section 2 use the same Crunchbase dataset and so their results can be directly compared. Firstly, the acquisition prediction of the Random Forest had a true positive (TP) rate of 95% and a false positive (FP) rate of 0.19%, which outperforms Xiang et al.'s Bayesian Network [15], which achieved a TP of 79.8% and a FP of 8.3%. Secondly, the paper by Krishna et al [13] predicted a binary outcome, either failure or success (which was defined as not failing). The paper trained 6 classifiers and 10 models depending on the round of funding raised by the company (seed, series A etc). The accuracy ranged from 0.793 for companies that have not raised any funding to 0.895 for companies that have raised a series B round. To compare, the trace of the Random Forest confusion matrix in Table V was divided by the total number of points to get an accuracy of 0.82. The achieved result indicates comparable performance as this paper's dataset included companies at various different stages of funding and the calculated accuracy lies in the range achieved by Krishna et al.

The other related works did not use the Crunchbase dataset and so cannot be directly compared, but it is still worth noting their achievements. The failure prediction papers [5-10] achieved accuracies ranging from 67% to 97% using a variety of different classification techniques. A Stanford paper [11] predicting investments in companies achieved an accuracy of 40% and a Swiss paper [12] predicting business model success achieved an accuracy of 83.6%. Lastly, a paper predicting acquisitions [14] achieved a TP of 81.3% and a FP or 34.4%. The performance achieved by the Random Forest classifier in this paper meets or exceeds the related work in acquisition/success prediction, but falls short in failure prediction due to the limited number of closed companies in the dataset.

One major limitation in this study was the sparsity and incompleteness of the Crunchbase dataset. Although the total dataset encapsulated more than 407,000 companies and people, many of these entries were missing key features required for building and running the models. The pruned dataset containing only companies that included all of the required features was limited to less than 14,000 entries. One suggestion would be to aggregate company data from other sources, such as Google Finance and Finimize, in order to supplement the sparse Crunchbase data.

Another limitation of this study was the limited available computational power. An increase in the size of the training and test sets would prompt the use of more powerful computers, which in turn would enable the specification of different parameters for each estimator. More sophisticated kernels such as polynomial or sigmoid may be used for the SVM classifier, thus avoiding the linearly separable class assumption. In the case of a Random Forest classifier, a more powerful machine would enable the training of an increased number of decision trees to handle a larger feature space while obtaining improved classification results.

## REFERENCES


[1] M. Marmer and E. Dogrultan, *Startup Genome Report Extra on Premature Scaling*, Startup Genome, 2011 [Online]. Available: https://s3.amazonaws.com/startupcompass-public/StartupGenomeReport2_Why_Startups_Fail_v2.pdf. [Accessed: 19- Apr- 2017]

[2] I. Hathaway, *Tech Starts: High-Technology Business Formation and Job Creation in the United States*, SSRN Electronic Journal, 2013.

[3] M. van Gelderen, R. Thurik and N. Bosma, "Success and Risk Factors in the Pre-Startup Phase", Small Business Economics, vol. 26, no. 4, pp. 319-335, 2006.

[4] J. Roure and R. Keeley, "Predictors of success in new technology based ventures", Journal of Business Venturing, vol. 5, no. 4, pp. 201-220, 1990.

[5] W. Beaver, "Financial Ratios As Predictors of Failure", Journal of Accounting Research, vol. 4, p. 71, 1966.

[6] E. Altman, "Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy", The Journal of Finance, vol. 23, no. 4, p. 589, 1968.

[7] E. Deakin, "A Discriminant Analysis of Predictors of Business Failure", Journal of Accounting Research, vol. 10, no. 1, p. 167, 1972.

[8] R. Wilson and R. Sharda, "Bankruptcy prediction using neural networks", Decision Support Systems, vol. 11, no. 5, pp. 545-557, 1994.

[9] K. Shin, T. Lee and H. Kim, "An application of support vector machines in bankruptcy prediction model", Expert Systems with Applications, vol. 28, no. 1, pp. 127-135, 2005.

[10] K. Shin and Y. Lee, "A genetic algorithm application in bankruptcy prediction modeling", Expert Systems with Applications, vol. 23, no. 3, pp. 321-328, 2002.

[11] A. Raghuvanshi, T. Balakrishnan and M. Balakrishnan, "Predicting Investments in Startups using Network Features and Supervised Random Walks", Stanford, 2015.

[12] M. Bhm, J. Weking, F. Fortunat, S. Mller, I. Welpe and H. Krcmar, "The Business Model DNA: Towards an Approach for Predicting Business Model Success", in 13th International Conference on Wirtschaftsinformatik, St. Gallen, Switzerland, 2017.

[13] A. Krishna, A. Agrawal and A. Choudhary, "Predicting the Outcome of Startups: Less Failure, More Success", 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), 2016.

[14] S. Ragothaman, B. Naik and K. Ramakrishnan, "Predicting Corporate Acquisitions: An Application of Uncertain Reasoning Using Rule Induction", Information Systems Frontiers, vol. 5, no. 4, pp. 401-412, 2003.

[15] G. Xiang, Z. Zheng, M. Wen, J. Hong, C. Rose and C. Liu, "A Supervised Approach to Predict Company Acquisition With Factual and Topic Features Using Profiles and News Articles on TechCrunch", Carnegie Mellon University, School of Computer Science.

[16] M. Kaufman, "Birthday 1.0 for CrunchBase 2.0", Crunchbase, 2015. [Online]. Available: https://about.crunchbase.com/2015/04/happy-1-0-birthday/. [Accessed: 20- Apr- 2017].

[17] K. Freytag, "Showcasing our Contributors", Crunchbase, 2014. [Online]. Available: https://about.crunchbase.com/2014/11/showcasing-our-contributors/. [Accessed: 20- Apr- 2017].

[18] "2013 Snapshot", Crunchbase Data. [Online]. Available: https://data.crunchbase.com/docs/2013-snapshot. [Accessed: 20- Apr- 2017].

[19] T. Balbo, "TechCrunch Posts Compilation", Kaggle. [Online]. Available: https://www.kaggle.com/thibalbo/techcrunch-posts-compilation. [Accessed: 20- Apr- 2017].

## Appendix A
## Data Transformation and Filtering

```
# segmenting objects table
df_all_companies = df_objects.loc[df_objects['entity_type']=='Company']
df_closed_companies = df_all_companies.loc[df_all_companies['status']=='closed']

df_acquired_companies = df_all_companies.loc[df_all_companies['status']=='acquired']
df_ipo_companies = df_all_companies.loc[df_all_companies['status']=='ipo']

df_all_fin_orgs = df_objects.loc[df_objects['entity_type']=='FinancialOrg']
df_all_people = df_objects.loc[df_objects['entity_type']=='Person']

# compiling a list of all relevant companies
frames = [df_closed_companies, df_acquired_companies, df_ipo_companies]
df_relevant_companies = pd.concat(frames)
relevant_company_ids = df_relevant_companies.id.values

# all remaining relevant tables
df_relevant_funrnds = df_funrnds[df_funrnds.object_id.isin(relevant_company_ids)]
df_relevant_investments = \
df_investments[df_investments.funded_object_id.isin(relevant_company_ids)]
df_relevant_milestones = df_milestones[df_milestones.object_id.isin(relevant_company_ids)]
df_relevant_offices = df_offices[df_offices.object_id.isin(relevant_company_ids)]

relevant_fund_ids = df_relevant_investments.investor_object_id.values
df_relevant_fin_orgs = df_fin_orgs[df_fin_orgs.id.isin(relevant_fund_ids)]
df_relevant_funds = df_funds[df_funds.object_id.isin(relevant_fund_ids)]

relevant_affiliation_ids = np.concatenate([relevant_company_ids, relevant_fund_ids])
df_relevant_relationships = \
    df_relationships[df_relationships.relationship_object_id.isin(relevant_affiliation_ids)]
relevant_people_ids = df_relevant_relationships.person_object_id.values
df_relevant_people = df_people[df_people.object_id.isin(relevant_people_ids)]
df_relevant_degrees = df_degrees[df_degrees.object_id.isin(relevant_people_ids)]
```

## Appendix B
## Feature Extraction

```
drop_cols = ["entity_type", "description", "first_milestone_at", "logo_url", "parent_id",
             "updated_at","created_at", "homepage_url", "investment_rounds",
             "last_milestone_at", "logo_width", "permalink", "short_description", "closed_at",
             "created_by", "entity_id", "logo_height", "twitter_username", "domain"]
df_company_mod = df_relevant_companies.drop(drop_cols, axis=1)

...

# feature - raised money between funding rounds
fnrnds_obj_dt_grp = df_funrnds.sort(columns="funded_at")
fnrnds_obj_dt_grp = df_funrnds.groupby(["object_id", "funded_at"]).count()["id"]
fnrnds_obj_dt_grp = fnrnds_obj_dt_grp.reset_index()
fnrnds_obj_dt_grp.drop("id", inplace=True, axis=1)
fnrnds_obj_dt_join = fnrnds_obj_dt_grp.merge(
    df_funrnds[["object_id", "funded_at", "raised_amount"]],
    how="inner", on=["object_id", "funded_at"])
# impute values of 0 for funding rounds with no raised amounts
fnrnds_obj_dt_join["raised_amount"] = \
    fnrnds_obj_dt_join["raised_amount"].apply(lambda x: 0 if x is None else x)

# create custom dictionary mapping company => [ra_1, .., ra_n]
fnrnds_dct = {}
combined = map(lambda x: (x[1], x[3]), fnrnds_obj_dt_join.to_records())
for obj_id, amt in combined:
```

```
    if obj_id not in fnrnds_dct:
        fnrnds_dct[obj_id] = [float(amt)]
        continue
    fnrnds_dct[obj_id].append(float(amt))

# compute deltas
fnrnds_delta = \
    {obj_id: np.average(np.diff(amts)) for obj_id, amts in fnrnds_dct.iteritems()}
fnrnds_delta = pd.DataFrame(zip(fnrnds_delta.keys(), fnrnds_delta.values()))
fnrnds_delta.columns = ["object_id", "avg_rsd_amt_diff"]

# single round - average is 0
fnrnds_delta.avg_rsd_amt_diff = \
    fnrnds_delta.avg_rsd_amt_diff.apply(lambda x: 0 if np.isnan(x) else x)
df_company_mod = \
    df_company_mod.merge(fnrnds_delta, left_on="id", right_on="object_id", how="left")

# feature represents 'growth' so companies with no funding have 0 'growth'
df_company_mod["avg_rsd_amt_diff"] = \
    df_company_mod["avg_rsd_amt_diff"].apply(lambda x: 0 if np.isnan(x) else x)
df_company_mod.drop("object_id", inplace=True, axis=1)


# feature - time between milestones (average time to achieve milestone)
# also want to compute time between founding and first milestone
mlstn_obj_dt_grp = df_milestones.sort(columns="milestone_at")
mlstn_obj_dt_grp = mlstn_obj_dt_grp.groupby(["object_id", "milestone_at"]).count()["id"]
mlstn_obj_dt_grp = mlstn_obj_dt_grp.reset_index()
mlstn_obj_dt_grp.drop("id", inplace=True, axis=1)

mlstn_dct = {}
for obj_id, dt in map(lambda x: (x[1], x[2]), mlstn_obj_dt_grp.to_records()):
    if obj_id not in mlstn_dct:
        mlstn_dct[obj_id] = [dt]
        continue
    mlstn_dct[obj_id].append(dt)

# compute average deltas
mlstn_delta = {obj_id: np.average(map(lambda x: x.days, np.diff(arr)))
               for obj_id, arr in mlstn_dct.iteritems()}
mlstn_delta = \
    pd.DataFrame(zip(mlstn_delta.keys(), mlstn_delta.values()),
                 columns=["object_id", "avg_mlstn_diff"])

# only one milestone - set to zero
mlstn_delta["avg_mlstn_diff"] = \
    mlstn_delta["avg_mlstn_diff"].apply(lambda x: 0 if np.isnan(x) else x)
df_company_mod = \
    df_company_mod.merge(mlstn_delta, left_on="id", right_on="object_id", how="left")

# zero milestone growth in place of null values
df_company_mod["avg_mlstn_diff"] = \
    df_company_mod["avg_mlstn_diff"].apply(lambda x: 0 if np.isnan(x) else x)
df_company_mod.drop("object_id", inplace=True, axis=1)

...
```

## APPENDIX C
## MODEL TRAINING

```
# building feature set
features = list(set(df_encode.columns) - set(omit_features))
```

```
X = df_encode[features].values
y = df_encode["class"].values
X_train = X[:int(len(X)*0.7)]
X_test = X[int(len(X)*0.7):]
y_train = y[:int(len(y)*0.7)]
y_test = y[int(len(y)*0.7):]

# naive bayes
nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred = nb.predict(X_test)
print "\n"
print classification_report(y_test, y_pred)

# linear svm
lin_svc = LinearSVC(verbose=2)
lin_svc.fit(X_train, y_train)
y_pred = lin_svc.predict(X_test)
print "\n"
print classification_report(y_test, y_pred)

# random forest
rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
print "\n"
print classification_report(y_test, y_pred)
```