Business Success Prediction Via Supervised Learning

Justin Guo[1], **Matan Gans**[2]

Received June 03, 2025
Accepted September 21, 2025
Electronic access September 30, 2025

The ability to accurately predict successful businesses is crucial for investors. In this paper, we aim to build effective and accurate machine learning models to predict successful businesses, allowing for more informed investment decisions, especially for early-stage startups. Many people, including experts in the field of finance like CFOs, are often overconfident in their ability to predict the future of a given company and its potential success. Using a machine learning model could provide a more data driven prediction. We used various company features, including total funding, date founded, and location from a dataset of over sixty thousand companies on five models: logistic regression, decision tree, random forest, k-nearest neighbors, and a neural network, to predict whether or not companies will be acquired through a merger and acquisition, which is our metric of success for this problem. We used the AUC score primarily to evaluate the models, since AUC score is independent of class imbalance. The random forest performed best in AUC score, with 79.56%, but the more complex neural network fell short of even some basic models, with only 70.49%. Given the results, a random forest model is probably best for the practical application of finding good potential investments.

Keywords: Machine learning, businesses, investment, finance, CFO, logistic regression, decision tree, random forest, k-nearest neighbors, neural network, merger and acquisition, AUC score

Introduction

The world of investing is built upon the difficult task of identifying successful companies before they achieve widespread recognition. Traditionally, investors, venture capitalists, and financial professionals usually rely on intuition, experience, and basic financial metrics to make decisions about where to invest. However, research has shown that even seasoned experts such as CFOs can be overconfident in their ability to foresee a company's future performance[1]. This overreliance on subjective judgment underscores the need for more objective, data-driven approaches to evaluating company potential. In recent years, machine learning has emerged as a promising tool for tackling complex prediction problems across a range of industries. With the ability to analyze large datasets and detect subtle patterns, machine learning models offer a systematic way to predict company success. These models have the potential to outperform human judgment, especially when trained on diverse and relevant company-level features.

Thus, from a practical perspective, machine learning models for predicting company success can act as objective tools for identifying promising companies, especially in early or growth stages where traditional financial metrics may be limited or unavailable, therefore helping investors gain an edge on the competition and invest in small companies with huge projected success.

[1] *Brunswick School, Greenwich, Connecticut, United States*
[2] *Inspirit AI, Palo Alto, California, United States*

So, can machine learning models predict whether a company will be acquired based on publicly available company-level features? And if so, what specific machine learning models are best at predicting company success using such features? The primary objective of this study is to answer these questions, developing and evaluating machine learning models that can predict company success, comparing the success of each model with others to determine the most accurate and efficient models, and determining where each model falls short. We predicted that the neural network would perform the best, due to its ability to process large amounts of data and analyze complex patterns.

The prediction of business outcomes has long been a subject of interest in corporate finance and machine learning. Altman (1968) introduced the Z-score formula, which uses financial ratios to determine the probability that a company will go bankrupt within two years[2]. Shumway (2001) proposed a hazard model for company survival analysis, incorporating time-varying company features, as opposed to the static features used in models prior[3]. More recent works have seen the integration of machine learning into financial modeling. Heaton, Polson, and Witte (2016) discussed the broad potential for machine learning in financial prediction, concluding that machine learning models can potentially outperform traditional models due to their ability to detect complex data interactions[4]. Fischer and Krauss (2018) used long short-term memory (LSTM) networks for stock prediction, and were able to consistently outperform the general market[5]. Prior works have also incorporated Crunchbase data to predict startup acquisition like we are attempting. Xiang et al.

© The National High School Journal of Science 2025

(2012) combined basic financial features on Crunchbase with TechCrunch news articles to predict mergers[6]. Pan, Gao, and Luo (2018) compared the K-nearest neighbors model to other basic models using the same Crunchbase dataset as we are using for this problem[7]. Zbikowski and Antosiuk (2021) compared logistic regression, support vector machines, and gradient boosting classifiers for predicting business success using Crunchbase features, concluding that gradient boosting was best[8]. We noticed that none of these studies incorporated deep learning methods like neural networks or engineered new features from the basic Crunchbase features, so we decided to compare neural networks to other basic models, using some new engineered features.

This problem is a classification, supervised learning problem as we are seeking to predict one of two classes from a labeled dataset that trains our models[9]. The two classes to be predicted are: 0, which represents a not-acquired company, and 1, which represents an acquired, successful company. The dataset, taken from Crunchbase, contains metrics of more than 60 thousand companies. The data consist of both numerical and categorical features. Some examples of numerical data are total company funding, date at which the company was founded, and number of funding rounds. Categorical data examples are industry of company (technology, healthcare, etc.) and location of headquarters. We used four basic models, logistic regression, decision tree, random forest, and k-nearest neighbors, as well as one deeper model, a neural network. To evaluate our models, we used accuracy score, precision score, F1 score, recall score, and AUC score, with a focus specifically on AUC score[10].

While we acknowledge that the metric of acquisition may not capture the full complexity of business performance, it should be noted that acquisition is most often viewed as a positive liquidity event, indicating that a company has created enough market value or strategic value to be purchased. Some acquisitions are indeed made to simply rescue struggling companies, which would not be an indicator of success, but the majority of acquisitions imply potential or realized financial return, characteristic of a business' success. For startups in particular, Ferrati and Muffato note that acquisition is often the ultimate objective of both founders and investors in their cross-industry analysis of tech-startups[11].

Nevertheless, since we define success for a company by this single metric, the scope of this study is limited. Other metrics of success, such as revenue growth, profitability, and market share are not considered in this study. Additionally, data availability can be limited, especially with smaller or private companies where some data points are missing. Economic factors that impact acquisition activity such as changing market conditions, industry cycles, and geopolitical conditions are not able to be captured by the Crunchbase data, thus limiting predictive capacity. Since relatively few companies in the dataset have already been acquired, there may be a class imbalance (much more not acquired than acquired companies), however this can be remedied with scaling.

Methods

The dataset we used was taken from Crunchbase Data Export with more than 60 thousand companies' data in both numerical and text form. Four data files, named company, investments, rounds, and acquisitions, were provided. We took the important features of the company file as a baseline and added important features from the other three files to it, resulting in a combined dataset of 13 columns:

- **Name:** company's name.
- **Category_list:** industry of which the company belongs.
- **Funding_total_usd:** total amount of funding from all investment rounds.
- **Country_code:** country of company's headquarters.
- **City_code:** city of company's headquarters.
- **Funding_rounds:** total number of funding rounds.
- **Founded_at:** date of company founding (string form, year-month-day).
- **First_funding_at:** date of first funding (string form, year-month-day).
- **Last_funding_at:** date of most recent funding (string form, year-month-day).
- **Acquired:** status of company (0: not acquired/IPO, 1: acquired/IPO); new column created by checking if company in "acquisitions" file was also in "company" file (assigning 1 if so, 0 if not).
- **Funding_round_type:** name of type of funding round.
- **Raised_amount:** amount raised from most recent investor funding.
- **Funded_at:** date of most recent investor funding (string form, year-month-day).

We used one-hot vector encoding to transfer the categorical data of funding_round_type, country_code, and city_code to numerical data, making a new column for each type, country, and city. We also one-hot encoded funding_rounds to reduce bias[12]. We used the to_datetime feature of the Pandas library, parsing all date fields into datetime objects, so that we could take differences in dates without compromising temporal relationships. We created the years_since_founding feature by taking the datetime of today and subtracting the datetime founded_at, then

© The National High School Journal of Science 2025

dividing by 365. We calculated the time between first and last funding (funding_duration_days) then normalized it against the company's age at last funding to derive a relative funding intensity feature (funding_intensity). This feature allows the model to distinguish between, for example, a long funding gap in a startup and a similar sized gap in an established company, thus accounting for differences in company maturity. There were about five thousand companies for which funding_total_usd was a null value. We created a has_funding' column, labeling companies with a null value in the funding_total_usd column as 0 and with a non-null value as 1. We then cleaned the funding_total_usd and raised_amount column by removing all null values. We noticed that there were significantly more smaller values than larger values for funding_total_usd and raised_amount, so we took the log function of all values in these columns resulting in a more normal distribution [13]. Figure 1 below shows the funding_total_usd data distribution before and after taking the log function.

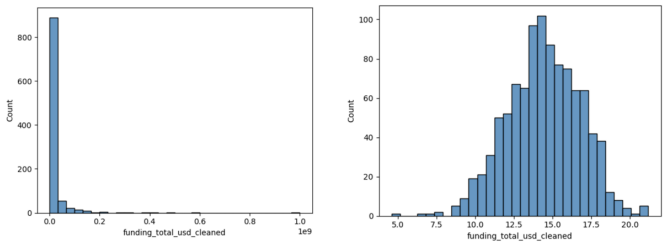

Figure 1: Total funding data before and after taking log function. Before, data has many more small values. After transformation, the distribution is more normal.

The dataset contained many more companies that were not acquired than companies that were acquired so we balanced the data, taking the majority class 0 and downscaling it to match the size of minority class 1.

We experimented with alternative dataset balancing methods, such as the Synthetic Minority Oversampling Technique (SMOTE), in order to try preserving more data. The SMOTE method balances the dataset by generating synthetic examples of the minority class, acquired, allowing the full majority class to be retained and thus reducing potential information loss. However, while this method improved class balance during training, it caused the models to overpredict the minority class on the imbalanced test set, resulting in a large number of false positives. The models, when trained on the artificially balanced data, developed an unrealistic sense of class distribution, so when trying to generalize their predictions in the testing set, which was quite imbalanced, they vastly over-predicted the acquired class and tanked in scoring metrics. Thus, we ultimately decided to use simple downscaling to deal with class imbalance.

To reduce variance from any single data split, we used stratified 5-fold cross-validation, dividing the dataset into five equal folds while preserving original class distribution in each fold. We used four basic models, logistic regression, decision tree, random forest, and k-nearest neighbors, and one complex model, a neural network. Each model was trained and evaluated across the five folds, and we took the average of metrics across the five folds.

The logistic regression model takes in the input features and forms a linear equation by summing the respective products of the features and their weights. The logistic function is then applied to the linear equation, giving the equation a value between 0 and 1 from which a prediction can be made. We used the limited-memory BFGS optimizer to minimize the loss function in training.

We used a decision tree as another basic model. Decision trees are singular trees where the nodes represent features of the dataset. The model splits data into subsets based on feature values, with the aim of maximizing learning. The branches on the tree are the model's decisions and the leaves are the final predictions.

Since the decision tree is prone to overfitting, we also used a random forest. Random forests use multiple decision trees, where each tree is trained on random subsets of features selected with replacement. This is bootstrap aggregation, reducing variance in the training data and hence preventing overfitting.

We also used k-nearest neighbors. This classification model tries to classify a desired point into a category by calculating the distance between the point and its neighboring points. The distances are sorted and the point is assigned to the class most common among its k-nearest neighbors. We determined that 13 was the best value for k, the amount of nearest neighbors, so used that as the hyperparameter.

We used a complex model of a neural network to see if deep learning could get better results. Neural networks receive raw data through the input layer, perform intermediary computations using interconnected nodes in the dense layers, and produce a result in the output layer. Each connection between nodes is given a weight based on the feature's importance [14]. To determine the optimal architecture for the neural network, we used RandomizedSearchCV to test different combinations of dense layers and neurons. RandomizedSearchCV randomly samples combinations from an input range, evaluates the combination by cross-validation, and selects the best performing combination of parameters by our chosen F1 score. We chose the input ranges to be between 1 and 5 layers, and between 10 and 100 neurons. Ultimately, two dense layers of 20 neurons each was the most optimal. We used the activation function rectified linear unit (relu) for each dense layer, which returns the maximum value between 0 and the input value. We attempted to mitigate potential overfitting by incorporating L2 regularization, which adds a penalty term to the loss function that discourages the model from assigning overly large weights to any particular feature. We applied an L2 regularization factor of 0.0005 to each dense layer. Additionally, we added dropout layers after each dense

layer with factors of 0.2. These layers randomly dropped 20% of neurons during training, lessening overfitting by ensuring the model did not rely too heavily on any one neuron. We used the sigmoid' activation function for the output layer of one node as it converts the model's outputs into probabilities in classification problems. We also used the binary cross-entropy loss function to train the model. This function measures the difference between the predicted probability from the sigmoid output and the actual class label. It penalizes incorrect predictions, so the model can learn more effectively. We used the Adaptive Moment Estimation (Adam) optimizer, which makes updates to model weights during training using the error from the loss function, resulting in better predictions as training progresses. We used early stopping to prevent unnecessary training time. Early stopping detects when the model is no longer improving in accuracy and terminates training. We chose a patience value of 7, meaning if no improvement was made over 7 epochs, the model would stop. This value allows certainty that the model is no longer improving while not running too many unnecessary epochs. Figure 2 is a visual of a neural network, with 5 input nodes, 3 dense layers of 9 nodes each and an output layer of 2 nodes [15].

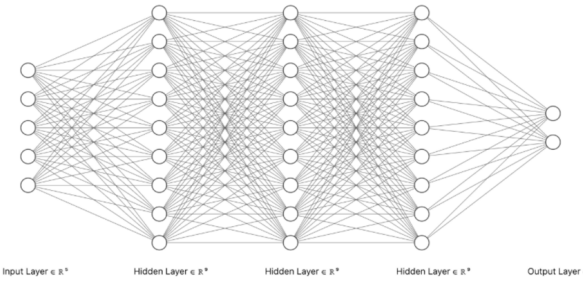

Figure 2: Example neural network consisting of 5 input layers, 3 dense hidden layers with 9 nodes each, and 2 output layers.

We tuned hyperparameters for all models. The hyperparameters were: inverse regularization strength (c-value) for logistic regression, maximum depth for decision tree, number of trees for random forest, number of neighbors for k-nearest neighbors, and learning rate for neural network optimizer. We used GridSearchCV for ranges of values for each hyperparameter. This method performs cross-validation on a single train-test split, training a given model on different subsets of the training data and selecting the hyperparameter value that yields the best average accuracy across the folds. To improve tuning reliability, we implemented nested cross-validation. Nested cross-validation repeats the tuning and evaluation process over several different train-test splits. For each split, the model is tuned on a training set using GridSearchCV, and evaluated on an unseen test set. Final performance is averaged across all test sets. For c-value, we chose to test the values 0.01, 0.1, 1, and 10. Low values have strong regularization, preventing overfitting, while high values allow the model to capture more complex patterns. For maximum depth, we chose to test the values 5, 10, 15, and 20. Lower values test whether a simple tree works well, while higher values work better when data is complex. We capped the tested values at 20 because higher values could cause overfitting. For random forest trees, we tested the values 25, 50, 75, and 100. Smaller values risk underfitting, while larger values increase computation time for diminishing returns. For k-neighbors, we tested the values 3 through 15, incrementing by 2. We determined this range since smaller values could overfit noise while larger values could miss patterns by averaging over too many neighbors. We manually tuned the learning rate for the neural network optimizer, testing values between 0.0001 and 0.1. The value 0.001, which is the default learning rate for the Adam optimizer resulted in the best scores. Table 1 shows the ranges of the values tested and the values which yielded the best results.

Table 1 Range of tested hyperparameter values and most optimal value for all model hyperparameters.

Model Hyperparameter	Range of Tested Values	Most Optimal Value
Logistic Regression C-Value	0.01–100	1
Decision Tree Max Depth	5–20	10
Random Forest Trees	25–100	50
K-Neighbors	3–15	13
Optimizer Learning Rate	0.0001–0.1	0.001

To establish a baseline for evaluating model performance, we implemented a dummy classifier using the most-frequent strategy. This classifier always predicts the majority class, not acquired, from the training set. This approach uses no input features or learning, but serves as a benchmark to compare with our other models, to determine if they actually provide value.

After running the basic models, we will evaluate them with average accuracy score, precision score, F1 score, and ROC-AUC score across the five folds. We will evaluate the neural network by those same scoring metrics, as well as by tracking the accuracy, validation accuracy, and loss over all the epochs. We also tracked the training times for all models.

Accuracy score is the proportion of number of correct predictions to total number of predictions, hence calculated by dividing the former by the latter and returning a percentage. Precision score measures precision, the proportion of true positives against the sum of true positives and false positives. F1 score combines precision and recall scores by taking the harmonic mean between the two. Precision score was described above and recall score is the proportion of true positives over the sum of true positives and false negatives. AUC score evaluates the ability of the model to distinguish between the two classes by taking the area under the ROC curve. This curve is found by plotting the true positive rate on the y-axis against the false positive rate on the x-axis. The area under the curve is found, with

© The National High School Journal of Science 2025

Table 2 Comparison of all scores between all models. Focus is primarily on AUC score. Random Forest performs best in all scoring metrics.

Models	Accuracy Score	Precision Score	F1 Score	AUC Score	Recall Score	Training Time
Dummy Classifier	90.59%	0.00%	0.00%	50.00%	0.00%	0.002 s
Logistic Regression	67.81% ± 1.2%	66.93% ± 1.5%	68.65% ± 0.8%	71.39% ± 1.2%	72.07% ± 0.5%	1.09 s ± 0.14 s
Decision Tree	66.39% ± 0.6%	65.75% ± 0.7%	66.72% ± 1.1%	77.39% ± 0.6%	68.06% ± 2.7%	0.25 s ± 0.02 s
Random Forest	72.77% ± 1.1%	70.87% ± 1.5%	73.39% ± 0.9%	79.56% ± 1.6%	79.26% ± 1.7%	3.56 s ± 0.17 s
K-Neighbors	64.89% ± 1.5%	60.49% ± 1.1%	68.55% ± 0.8%	69.97% ± 1.7%	79.09% ± 0.6%	0.01 s ± 0.0009 s
Neural Network	68.47% ± 0.9%	66.22% ± 1.2%	70.23% ± 0.8%	70.49% ± 1.5%	74.75% ± 0.5%	6.51 s ± 0.62 s

resulting values ranging from 0.5 (model classifies randomly) to 1.0 (model classifies perfectly).

Results

The average across 5 cross-validation folds of scores (accuracy, precision, F1, AUC, recall) and the 95% confidence interval range are shown side-by-side in Table 2 for all the models. The ROC curves for all models are plotted with a chance line (curve if class labels were guessed completely at random) in Figure 3. Confusion matrices for the best two models by F1 score, random forest and neural network, are shown in Figures 4 and 5. Figure 6 is a plot of the training and testing accuracies for each epoch of the neural network.

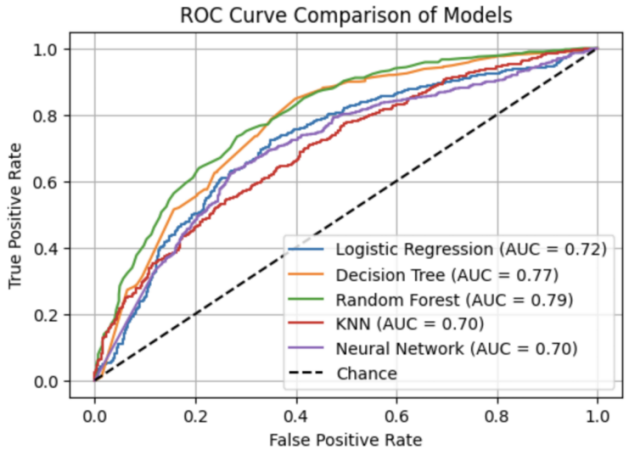

Figure 3: ROC curves for all models compared to the diagonal chance line representing random guessing. Random forest achieves the highest AUC (80%).

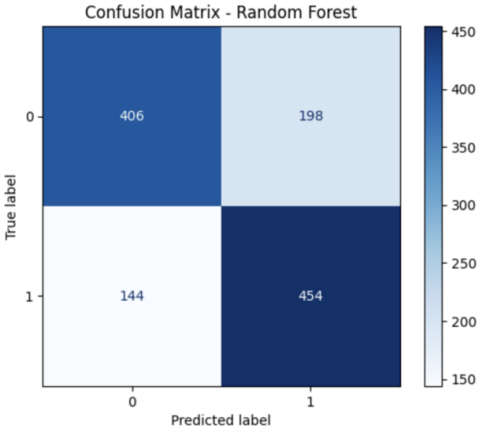

Figure 4: Confusion matrix for the random forest model, showing the distribution of true vs. predicted labels. The model performs well on true positives and true negatives, but produces many false positives.

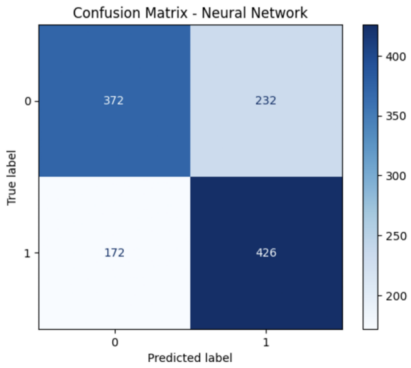

Figure 5: Confusion matrix for the neural network, showing the distribution of true vs. predicted labels. The model does not perform as well as the random forest predicting true positives and negatives, and has quite high false positive rate.

Discussion

The table of models and metrics shows that the random forest performs best overall, with the highest scores for all metrics. AUC score, which represents the probability that the model scores an acquired company higher than a not acquired one across all thresholds, is not affected by the class imbalance inherent to the dataset in this problem like other scores are, so we focus primarily on that scoring metric.

The dummy classifier, which always predicted the majority class, performs well in accuracy score because of the imbalanced data favoring the majority class. Precision score, recall score, and F1 score all are calculated with true positive rate in the numerator, and since the dummy never predicted the minority class, it scores 0% in those metrics. Its AUC score is 50%

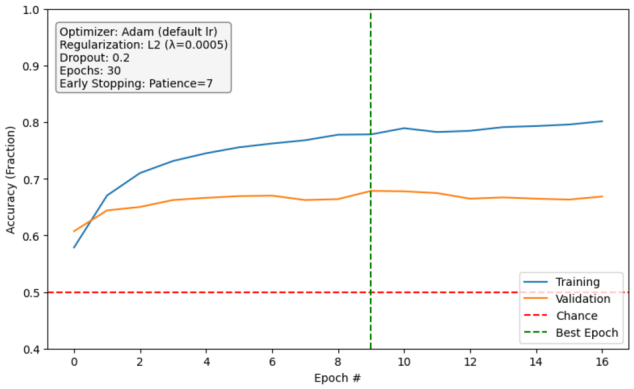

Figure 6: Neural network overfitting curve showing training and validation accuracies across epochs. Early stopping detects when improvements aren't being made and halts training at 16 epochs instead of 30. Training accuracy improves while validation accuracy plateaus, indicating overfitting.

as expected, since it is essentially randomly guessing. These results reinforce that accuracy alone is a misleading metric in imbalanced classification problems, and more importantly that our trained models do add predictive value.

As said, the random forest performs the best, with an AUC score of 79.56%. Though the neural network performs second best in F1 score, with a score of 70.23%, it falls short of other models like logistic regression and decision tree in AUC score, with only 70.49%.

Figure 3 shows the ROC curves for all models. Each point on the curves represents a different classification threshold. At higher thresholds, only predictions where models are very confident are labeled as positive, while at low thresholds, less model confidence is required and more predictions are labeled as positive. The random forest and decision tree models have curves that are consistently above the other models' curves, indicating that over all thresholds, these models produce fewer false positives and more true positives than others. Both the logistic regression and neural network curves stayed relatively close to the better performing models at middling thresholds, but fell off near the extremes. This indicates that these models make predictions that are not confidently separated, allowing fairly good performance at middle thresholds but poor performance when needing high confidence. In contrast, the k-nearest neighbors model curve was close to the tree-based models at the extremes, but fell closest to the chance line in the middle. This indicates that the model struggles when very subtle class distinctions must be made.

Random forests do well handling large datasets with many input features and are able to reduce overfitting effectively by training individual trees on different subsets of data and features. Given the large amount of features included in this particular dataset, and the fact that the data varied in type (categorical, numerical, and binary), the random forest excelled in making predictions. This model did have longer training time, at 3.56 seconds, than all the other basic models, due to the many trees. For this problem, the random forest only had a small increase in training time, and given its better results, the trade-off was worth it. However, training time increases even more with more data. Thus, when scaled to larger datasets, random forests could be inefficient despite their performance gains.

The neural network, a more complex model, did well in F1 score but lagged behind simple models in AUC score. The high F1 score indicates that the neural network was able to predict better than other models at a certain probability threshold, yet its low AUC score shows that the model was not consistently ranking acquired companies over not acquired ones over all probability thresholds. Neural networks do have the advantage of being able to detect more complex nonlinear relationships of features. Logistic regression, for example, assumes a linear decision boundary, and may have been too simple a model for this problem. That said, given our results, it is clear that the neural network has many flaws. The complexity of neural networks causes much greater training time than other basic models. Here, the neural network had significantly greater training time, at 6.51 seconds. If neural networks do not provide significant performance increases, they might not be worth it given their inefficiency. Figure 6 displays the training and testing accuracy of the neural network across 30 epochs (early stopping ends the process after only 16 epochs). Training accuracy is a measure of how well the model performed on the data it was trained on, while testing accuracy evaluates the model based on new, unseen data. The training accuracy curve consistently increases to over 80%, while validation (testing) accuracy only increases during the first few epochs, then remains relatively constant around 65%. The growing gap between the training and testing accuracy curves is evidence of overfitting. The neural network has learned the training data very well, as indicated by the consistent increase of the training accuracy curve, but this learning is mostly memorization of the training set, and the model is unable to generalize and make more accurate predictions about unseen data, as evidenced by the plateauing testing curve.

Even with optimized neural network structure, overfitting still occurred, implying that for this specific problem and dataset, neural networks might be too complex.

The results of this study carry many practical and theoretical implications. We showed that machine learning models can effectively analyze large amounts of data and extract patterns to predict future outcomes of businesses. In practice, investors should, based on our results, choose random forest models to predict successful companies for investment, not only because they yielded the best accuracy numbers, but because they are generally easier to understand and take less computing power than more complex models such as neural networks. Even though the accuracy scores for these models are not perfect, investors can still use machine learning as a second opinion, rather

© The National High School Journal of Science 2025

than simply making the decision on their own. Consider a case where an investor wants to evaluate a large batch of early stage startups. Instead of relying solely on intuition and spending much time manually evaluating each startup, the investor could input company features, such as funding history, funding dates, and location, into a random forest, getting a quick, data-driven output. The model would output a probability score for each company's likelihood of being acquired, helping the investor efficiently decide which startups to look deeper into. While not a replacement for personal judgement, the model would save time and offer a quantitatively informed supplement to the decision.

That said, deploying these models in practice requires addressing of a few key issues. Constant maintenance is very important. Models trained on historical data can become outdated if funding patterns or company behaviors change over time. To maintain their relevance, models should be monitored, validated, and even retrained at times as new data becomes available. Model explainability is also important. For investors to have trust in using models and be able to use them effectively, they must first understand exactly how they work and what their potential flaws may be. This is another reason in support of random forests, since they are typically quite simple to understand and use. Data availability and quality are other possible concerns. The models depend solely on company-specific data, such as funding rounds and amounts. If these features are unavailable or inconsistently reported, predictions could become unreliable, especially if missing features are vital to prediction.

We originally set out to determine if machine learning was a viable way of predicting business success, and if so, what specific models were best at doing so. Each model performed fairly well, better than the random guessing of the dummy classifier, and we were able to successfully determine the best models for the job. Both those research objectives thus were fulfilled. However, while we predicted that the neural network would perform the best, it was ultimately outclassed by the random forest. Our original hypothesis was thus proven incorrect by the results, as we had overlooked the fact that neural networks are prone to overfitting and might be too complex for this dataset.

This study is ultimately limited by a few factors, including its focus on company-specific features over broader macroeconomic trends, potential for survivorship bias in the data, risk of data leakage, limited dataset scope, and a narrow definition of success.

The dataset spans many years, with the bulk of the companies having founding dates between the late 1990s and early 2010s. During which period, as is true over all relatively long periods of time, economic conditions fluctuated significantly. The models we used made acquisition predictions solely based on the individual companies' specific features in the dataset, and did not account for macroeconomic variables that could significantly impact acquisition activity. Different market conditions can lead to huge differences in acquisitions. For example, companies founded or funded during periods of economic expansion, such as the post-2009 recovery, generally have higher acquisition rates than those launched in downturns like the early 2000s recession or the 2008 crisis. Changing interest rates also greatly affect the general pattern of acquisitions, as lower interest rates tend to result in more acquisitions. Specific industry booms like the rise of social media in the 2000s could give companies in the surging industry significantly more funding and acquisition activity. However, our models, using only company-specific features, do not pick up on these general factors of market sentiment, temporal trends, specific industry growth, and other economic variables in their predictions and consequently, may not generalize well across different market conditions and different time periods.

The dataset may be biased in that it likely includes more companies that have survived long enough to be recorded with meaningful data, whereas startups that quickly failed or never received funding may be underrepresented or missing entirely. This is survivorship bias, and as a result, since the models may be learning patterns from companies that were relatively successful or stable by default, predictions may be skewed towards overestimating acquisition likelihood. The models thus might perform fairly well on this specific data, where all companies are already fairly well off, but may not generalize well to the full population of startups, including those that fail early.

There is potential for data leakage using time-based features, especially last_funding_at. For companies that were acquired, there is a very real possibility that this feature date occurred after the company was acquired, allowing the model to learn from data that would not have been available at prediction time, giving seemingly better results than models should have had. Since acquisition dates are not available in the dataset, there was no good way to fully prevent this data leakage issue. While we mitigated the leakage risk slightly by using features that described broader funding time trends like funding_intensity rather than specific times such as last_funding_at, the calculation for the derived temporal features still had to use specific dates.

While Crunchbase provided a healthy number of companies and features, it could be lacking in very specific financial data beyond the features we had access to which might lead models to different predictions. Moreover, data points for the features we did have were occasionally missing. Additionally, the definition of success is subjective. We choose the metric to be whether or not a company gets acquired through the process of a merger and acquisition to have a binary output. This may oversimplify the determination of a successful outcome for a company.

For future work, we suggest including macroeconomic indicators and data from failed startups along with the company-specific features already in use as data for predictions to enhance model generalizability. Improving the neural network should also be attempted, since its ability to analyze complex data makes it an excellent candidate to solve problems like this one.

A larger dataset that incorporates both more companies and more features could be used to potentially fix the overfitting problem. It would be interesting to look into feature importances using logistic regression coefficients, Shapley values, or the Gini index, to determine which specific factors contribute most to the models' prediction of business success. For a more comprehensive definition of success, future studies could try to predict multiple success criteria, including revenue growth, profitability, or customer growth.

Appendix

Code: https://github.com/justinguo18/Predicting-Company-Success-with-Machine-Learning/blob/main/M%26A_Prediction.ipynb
Dataset: https://drive.google.com/file/d/1sW1fDXLpnC1oSCVRB6FdzZtUykWGnyhD/view?usp=sharing

Acknowledgments

Thank you to Matan, Matthew, and the Inspirit AI team.

References

1 R. Thaler, *The Overconfidence Problem in Forecasting*, https://www.nytimes.com/2010/08/22/business/economy/22view.html.

2 E. Altman, *Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy*, https://www.jstor.org/stable/2978933.

3 T. Shumway, *Forecasting Bankruptcy More Accurately: A Simple Hazard Model*.

4 J. Heaton, N. Polson and J. Witte, *Deep Learning in Finance*, https://arxiv.org/abs/1602.06561.

5 T. Fischer and C. Krauss, *Deep learning with long short-term memory networks for financial market predictions*, https://www.sciencedirect.com/science/article/abs/pii/S0377221717310652.

6 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*, https://www.researchgate.net/publication/266224153.

7 C. Pan, Y. Gao and Y. Luo, *Machine Learning Prediction of Companies' Business Success*, https://cs229.stanford.edu/proj2018/report/88.pdf.

8 K. Zbikowski and P. Antosiuk, *A machine learning, bias-free approach for predicting business success using Crunchbase data*, https://www.sciencedirect.com/science/article/pii/S0306457321000595.

9 S. Brown, *Machine learning, explained*, mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained.

10 D. Steen, *Understanding the ROC Curve and AUC*, https://towardsdatascience.com/understanding-the-roc-curve-and-auc-dd4f9a192ecb/.

11 F. Ferrati and M. Muffatto, *Startup Exits by Acquisition: A Cross Industry Analysis of Speed and Funding*, https://www.researchgate.net/publication/352192927.

12 A. Thakur, *How One-Hot Encoding Improves Machine Learning Performance*, https://wandb.ai/ayush-thakur/dl-question-bank/reports/How-One-Hot-Encoding-Improves-Machine-Learning-Performance--VmlldzoxOTkzMDk.

13 A. Shrestha, *Understanding the Role of Logarithmic Functions in Machine Learning Models*, https://aishwaryashrestha232.medium.com/understanding-the-role-of-logarithmic-functions-in-machine-learning-models-5d7f6ee95a9c.

14 C. Hansen, *Neural networks from scratch*.

15 A. Lenail, *Publication-ready NN-architecture schematics*, alexlenail.me/NN-SVG/.

© The National High School Journal of Science 2025