Algorithmic Venture Capital

Predicting valuation step-up multiple in venture backed companies through deep learning techniques

Casey Caruso
Google Corporation
caseycaruso@google.com

Francisco Enriquez
Stanford Business School
fwe@stanford.edu

Abraham Oshotse
Stanford University
aoshotse@stanford.edu

Gautam Pradeep
Stanford University
gpradeep@stanford.edu

Abstract—**Venture capital plays an instrumental role in the modern American economy. Corporations such as Google, Apple, Facebook, Instagram, PayPal, Tesla, SpaceX, Airbnb, FedEx and Intel all received venture funding to enduringly grow into the iconic companies they are today. As of 2015, venture backed companies made up 17% of U.S. public companies, accounted for 44% of R&D spending and employed 11% of U.S. citizens [3]. Despite the unequivocal impact venture capital has on the United States, the process behind venture investment decisions remains manual, subjective and unsystematic.**

This paper sets out to explore whether venture capital can benefit from machine learning, particularly deep learning, to make investment decisions in a more scientific way. Specifically, we aim to predict the valuation step-up multiple in the subsequent financing round of venture backed U.S. companies. The primary dataset for our model comes from Pitchbook, a popular commercial dataset of company and funding information. We produced a regression-based model to utilize a fully-connected 10-layer neural network to encode features and predict the valuation step-up multiple. Results show that for the regression task of predicting company valuation step-up, deep-learning techniques meaningfully outperform statistical inference models such as linear regression in. However, non-deep learning models, such as random forest, appear to be better suited for such regression analysis.

Keywords: fully-connected neural network, venture capital, regression, company valuation prediction

I. Introduction

Venture capital is a type of financing where investors provide capital to startups to finance growth in return for equity. After interviewing 20+ investors from top venture funds such as Bessemer Venture Partners, Andreessen, Sequoia, Redpoint and GV, we have defined the typical early-stage venture capital process as follows: investors conduct research on the team, market, product, competitive landscape and financials. The investing team amalgamates research and subjectively weighs data points to arrive at a binary decision of whether to invest or pass on the opportunity. A successful investment is one where the company's valuation appreciates and the investor is able to liquidate their stake at the heightened valuation, generating a profit. Given this, the ideal task is predicting valuation multiple at time of exit. With this said, potential exit and liquidation events are unpredictable, vary significantly in time horizon and are often not documented. Therefore, we defined our goal as predicting the valuation multiple in the subsequent round of funding. While we recognize gains cannot necessarily be realized in successive rounds, valuation step-up multiple is the cardinal indicator of future exit value.

We limited our dataset to venture-backed companies in the U.S. who have raised a seed round of financing and were founded between the years 1990 and 2020. Through leveraging Pitchbook data, a popular commercial dataset of company and funding information, we were able to yield over 160 features ranging from founder name, which we can deduce gender from, to industry_code, from which we can extract market-wide growth rates. After extensive cleaning, pre-processing and transformations, we honed in on 30 core features. These features were fed into a custom-trained 10-layer dense, fully-connected neural network which encodes the features and predicts a numerical valuation step-up multiple.

II. Related Work

Some previous work has attempted to predict the success of a startup based on information regarding the company and its founders. A particular project named the *Holy Grail of Venture Capital* [5] was conducted by a technical and VC-experienced team from the University of California at Berkeley who utilized traits of a founder including fear of failure, persistence, perusation, reliability, competitiveness, network strength, and trust to determine likelihood of success. These traits were scaled as a quantitative measure from 1 to 5. Using data from Crunchbase and a founder survey, the team ended up using eight features and one target output determined from their cycles of pre-processing. When building their model, they compared algorithms built on logistic regression, SVM, perceptron, Naive Bayes, XGBoost, as well as random forest. While many algorithms were compared, they did not explore using deep learning techniques.

Another prior work was conducted by Will Gornell and Ilya Strebulaev to build a valuation model of venture capital-backed companies with multiple rounds of financing [4]. They built a model using data from Pitchbook and Genesis and limited their scope to U.S. companies from 2004 onwards. They analyzed about 19,000 companies over 37,000 financing rounds. The paper explores regression models to determine how current value, value change, and prior contractual terms impact the terms of a new round. Their results indicated an overestimation of post-valuation of companies but found results in line with the prices reported from the VC industry's finance intermediaries.

III. Dataset and features

Our research utilized data from Pitchbook, one the premier commercial datasets of startups and venture capitalists. We began our data collection by scraping every company and fundraising round from the Pitchbook database. We restricted our dataset to U.S. companies that had raised a seed round of venture financing and were founded post-1990 and pre-2020.

While the raw dataset included over 10,000 rows, a large percentage of data entries were notably sparse and included duplicates. To solve for such, we subsequently mapped every company and fundraising round to a unique company identifier and deal identifier, respectively. We then removed duplicate entries by their unique deal identifier and excluded companies for which we could not find consistent, round-to-round information. After data cleaning, a dataset of 34,265 unique deals across 22,067 unique companies remained.

Given we are predicting the valuation step-up multiple in subsequent rounds, we wanted to ensure our data was well distributed among various initial fundraising rounds. Below is the distribution of rounds for the five predominant fundraising stages.

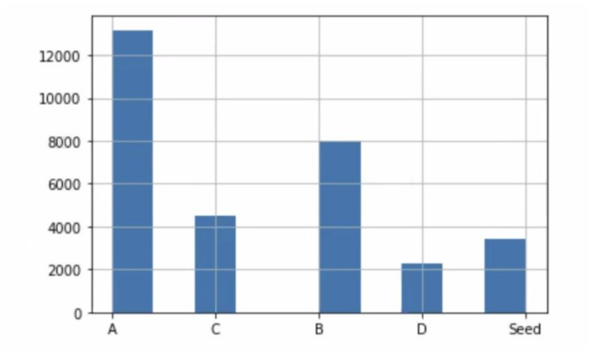


Fig. 1. Distribution by fundraising round (Seed, A, B, C, D) in dataset

An important factor to correct for within our dataset was survivorship bias given the preponderance of companies that fail, especially early on in their development. Inherently our dataset is biased towards favoring successful vs unsuccessful companies as we are limited to a dataset of companies which PitchBook tracks. According to The National Venture Capital Association, roughly 25% of venture-backed businesses fail. In our dataset, only 7% of companies fail, which is nearly 1/4th of the national average [3]. We attempted to solve for survivorship bias by taking a longitudinal approach to our data collection and factoring company status into our predictive model. More concretely, if the business_status was "Out of Business", we assigned a valuation multiple of 0 to the last round of funding. Despite our efforts, we are aware survivorship bias is a weakness of our model.

A. Data Processing and Transformations

The data natively included 160+ columns, including: founder demographics, company information (e.g. year founded, location, employee count), customer count, number of deals, and features of the fundraise (e.g. deal size and liquidation preference as a proxy for perceived risk), and more. Please see the full set of raw available columns in the appendix.

We performed extensive pre-processing and transformations to convert columns into meaningful features. Below is a table of the pre-processing and transformation techniques deployed.

	Pre-processing techniques
As-is	`["employee_count","year_founded","deal_number","percent_owned","percent_acquired","pre_valuation","raised_to_date","deal_size","price_per_share_x","post_valuation"]`
Hot encodings	`["business_status","ownership_status","financing_status","universe","sic_codes","naics_codes","state","stock_type_x","state","deal_status","deal_class","deal_type_2"]`
Binary /Tertiary Conversions	`["website","parent_company","tech_hotspot","country","name", "board_voting_rights"]`
Count Conversions	`['sister_companies_count','subsidiary_companies_count','customers_count','market_count','competition','products']`
Other	`'elapsed_announced_deal'` - Contains # of days elapsed between the announced date and the date the deal got done. Calculated from fields `'deal_date'` and `'announced_date'` `'founder_gender'` - Using gender_guesser.detector to predict the gender of the founder

In addition to the techniques mentioned above, we integrated a GDP-deflator as an effort to standardize values over time.

We also implemented normalization, a technique to constrain values to a common scale, for features which were based on multiple ranges. Normalization brought variables to the same range which shortened the time to model convergence. Note that without normalization the gradients took a much longer time to find a local minimum.

B. Generating the output variable

The dataset did not natively include our desired output variable: valuation step-up multiple. To generate this label, we grouped all rows by company_id, iterated over each deal and calculated the valuation step-up multiple which is equal to:

$$\textit{valuation step up multiple} = \frac{\textit{current round post money valuation}}{\textit{previous round post money valuation}}$$

Recall this formula yields the multiple compared to the previous round of fundraising. For instance, if the post money valuation of the Seed round was \$10M and the post money valuation of the current Series A round was \$20M, the output label of valuation step-up multiple would be 2.0. Note that the output of the algorithm is a numeric float.

Below is the distribution of our cleaned dataset based on our output label of valuation step-up multiple. As the figure displays, the majority of companies less than double round to round and the 99th percentile hovers at a 10.25x multiple.

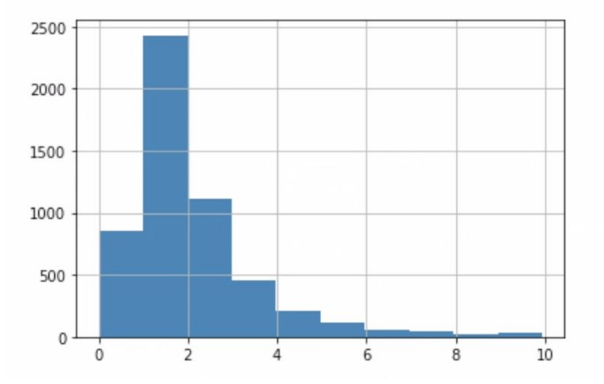


Fig. 2. Distribution of output label (valuation multiple) in training set

C. *Test, Development and Training Set*

The dataset was split into training, development and test sets. The final division was:

- Training set: 70% (16424 samples)
- Development set: 10% (2348 samples)
- Validation set: 10% (2347 samples)
- Test set: 10% (2344 samples)

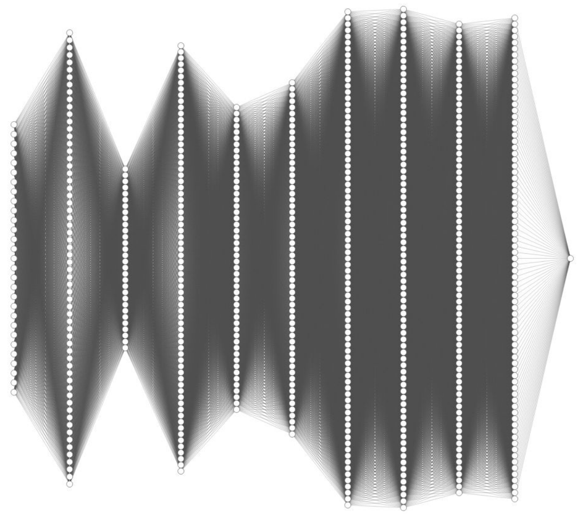
We arrived at this breakdown through considering the goals of the development set, which is to evaluate different algorithms and hyperparameters, and the training set, which is to learn the optimal model parameters. We aimed to utilize the training and dev set while fitting the model so we could obtain and plot how both the losses vary over the epochs of training. In order to have a separate dataset for hyperparameter tuning and experimentation, we crafted a validation dataset that was evaluated over only after training. To construct the subsets we randomly chose samples and confirmed the distribution of successful and unsuccessful companies were similar in all sets.

IV. Methods

Data manipulation and models were built using Keras [2], Tensorflow [1], Numpy [6], and Scikit-learn [7]. The architecture was designed from scratch, albeit motivated by previous research in this domain.

A. *Deep Neural Network Architecture*

Our final architecture was a deep fully-connected neural network consisting of a total of 10 fully-connected layers (4 hidden, 1 input, and 1 output layer). Given that our data was structured, quantitative data in tabular form, we postulated that a fully-connected neural network would be the optimal model compared to other neural network implementations such as CNNs or RNNs. We initialized the weights with He Normalization and implemented L1 Regularization for each layer except the output. Between each layer we passed the vectorized outputs into a Leaky ReLU activation function for all layers except the output where we used a linear activation. We used a linear activation in our final layer as our desired output prediction is a real number (float type) which is the valuation step-up multiple of the company. We settled on the following distribution after extensive testing and analysis detailed in the section VI (Experiments).



Input Layer ∈ $\mathbb{R}^{29}$ Hidden $\mathbb{R}^{62}$ Hidden $\mathbb{R}^{30}$ Hidden $\mathbb{R}^{70}$ Hidden $\mathbb{R}^{50}$ Hidden $\mathbb{R}^{58}$ Hidden $\mathbb{R}^{81}$ Hidden $\mathbb{R}^{78}$ Hidden $\mathbb{R}^{77}$ Hidden $\mathbb{R}^{79}$ Output $\mathbb{R}^{1}$

Fig. 3. Net Architecture

B. *Hyperparameters and Loss Function*

Relevant hyperparameters (outside of number of layers) which we tuned resulted in the following:

1. Learning rate: 0.005
2. Gradient Clipping Parameter: 0.7
3. Regularization: L1
4. Dropout Rate: 0.25
5. Epochs: 100
6. Batch Size: 64

For our loss function, we decided to prioritize mean-squared error (MSE) as our primary loss metric given our model is a regression problem, opposed to classification. MSE was selected over options such as MAE due to having harsher penalties for more distant errors compared to MAE (given that errors are squared).

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

C. *Baseline Models*

A linear regression model was implemented as a baseline to compare performance. The linear regression model yielded 6.6 * 10^{11} for MSE on the dev set which was significantly worse than our deep learning model.

We also implemented a random forest model as an alternative baseline to compare performance. We used the RandomForestRegressor model from sklearn and trained the model with n_estimators = 100 and random_state = 42. This random forest regression model yielded a .97 MSE on the dev set.

V. Experiments, results and discussion

A. Experiments

We ran our model and iterated our experiments using Google Colab and implemented the Google TPU hardware accelerator which significantly reduced our runtime.

Learning rate was tuned manually, using *keras.callbacks.ReduceLROnPlateau,* a function for learning rate step decay. This function reduces the learning rate when no improvement is seen within a certain number of epochs. Adjustments to the learning rate decay and scheduling were made by monitoring the loss curve, with preeminent results achieved using an initial learning rate of 0.005, a minimum rate of 0.001, and a 0.2 factor drop per reduction.

After running a few initial iterations of our model, we noticed a high bias given the substantial discrepancy between our training loss and that of other models such as random forest, as previously reported. To combat this underfitting, we implemented gradient clipping to prevent exploding gradients, which significantly reduced our training loss.

We ran several experiments in order to optimize our model.

The number of epochs was manually selected through various trial runs. We settled on 100 epochs because the training and dev set losses were plateauing at that point.

Another improvement came from changing the type of regularization to mitigate overfitting. When regularization was changed from L2 to L1 the MSE was reduced for both the training and dev set. We hypothesize this improvement was due to L1 shrinking the non-relevant weights to 0, serving as feature selection. After observing superior performance when utilizing L1 regularization, we implemented all of our subsequent experimental models using L1.

	Training MSE	Dev MSE
L1	1.77	1.55
L2	1.91	2.53

Fig. 4. Results from comparing L1 and L2 regularization

To optimize most other hyperparameters, we used a random search tuning process. Instead of using the Keras Tuner library, we decided to build the optimizer in-house, from scratch.

We ran fifteen different experiments, each utilizing a different seed (going sequentially from 0 to 14) to ensure each of our experiments yielded a different network architecture and utilized different hyperparameters so we could effectively compare. Seeds were used to ensure our experiments were replicable.

We iterated over three options for the activation function. Options included ReLU, Leaky ReLU, and the Swish function. The Swish function is similar to the ReLU function but it's a smooth function, meaning it smoothly blends past zero and upwards, and has been displaying promising results in academia and industry [8]. The equation and graph for the swish activation function is as follows:

$$f(x) = x * sigmoid(x)$$
$$= x * (1 + e^{-x})^{-1}$$

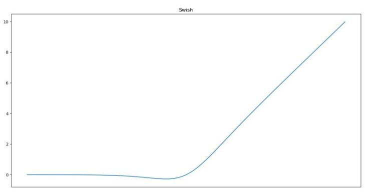


Fig. 5. Swish activation function formula and plot

Throughout each experiment we calculated three values of mean-squared errors - training, dev, and validation MSE. In our implementation, the dev dataset was used to calculate losses per epoch along with the training so we could generate a plot of how the loss changed over the training of the model. The training and dev MSE reported below are the final values after running through all the epochs. Validation MSE was determined by evaluating the model on a separate subset (different than the train, dev, and test) after all the epochs of training were completed. The table belows shows a subset of our experiments and results from our random search function.

	#0	#3	#4	#10	#13
Regularization	.00637	.00626	.000135	.00822	.00775
Clipnorm	0.6	0.2	0.8	0.5	0.3
Dropout	0.05	0.2	0.3	0.05	0.05
# of layers	7	4	5	5	10
Hidden units per layer	[71, 13,87, 25, 40, 91,74]	[76, 4,25, 23]	[91,76,54,13,62]	[32,93,97,33, 12]	[29, 62, 30, 70, 50, 58, 81, 78, 77, 79]
Activation	Leaky Relu	Leaky Relu	Leaky Relu	Relu	Leaky Relu
Training MSE	1.910	2.067	1.930	1.546	1.488
Dev MSE	1.814	1.778	1.670	1.593	1.543
Validation MSE	1.585	1.483	1.381	1.255	1.260

Fig. 6. Hyperparameter tuning experiments

B. Evaluation and Results

Below is a table of our best deep learning (DL) model alongside our two non-deep learning baseline models.

	Results				
	Description	***Training MSE***	***Validation MSE***	***Test MSE***	***Standard Deviation***
Best DL Model	10 layer leaky ReLu from hyperparameter tuning with L1	1.488	1.260	1.58	Training: 6.5 Validation: 0.61 Overall:4.6
Linear Regression	Ordinary least squares (OLS) linear regression with L1	2.452	2.339	2.441	N/A
Random Forest	RandomForest Regressor model with n_estimators = 100	0.153	1.101	1.105	N/A

Fig. 7. Results of NN arch improvements and hyperparameter tuning.

Mean-squared errors for our best model are also shown in the plot below over both training and development datasets.

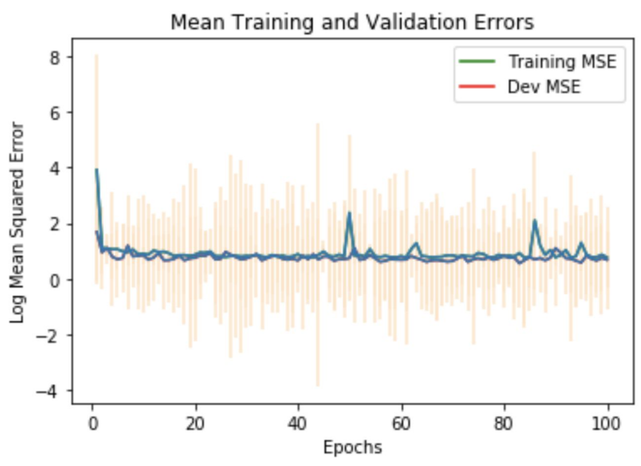


Fig. 8. Mean Log MSE among 5 runs of our best deep learning model plotted with vertical error bars for Training and Dev MSE

Our best model utilizing a deep neural network resulted in a 1.488 training loss, 1.260 validation loss, and a 1.58 test loss. After running our best model 5 times, we had a standard deviation in the MSE of 6.5 for just the training, 0.61 for the validation, and a combined 4.6 over both training and validation. Comparing our model with non-deep learning approaches, we see that the deep network performs better than the OLS Lasso linear regression model. However, the random forest regressor outperforms both linear regression and our deep learning model by a substantial margin. We hypothesize that this discrepancy is due to the robustness of random forest models for structured, tabular data.

C. Conclusion and Future Work

Among all our deep learning models, a 10 layer network with a Leaky ReLU activation and L1 regularization had the best performance, achieving a 1.488 training MSE, 1.260 validation MSE.

While we were not able to outperform the random forest model, we believe more analysis would have to be performed to understand the chief differences between the approaches that result in these discrepancies.

The primary limitations of this project were time and data collection. Not only did the data take a significant amount of time to collect but once collected, it was sparse and required ample pre-preprocessing and transformations. With additional time we would like to explore implementing a feed-forward neural network given it has shown promising results specifically with tabular data. Additionally, we would like to explore additional loss metrics and alternative ways of evaluating model performance.

Github: https://github.com/gautam0831/CS230_Project

VI. Member Contributions

All team members contributed equally to the direction of the project through weekly meetings. This included work for loading data, cleaning and pre-processing our features, experimenting with neural network architectures, and analyzing our results together in discussion.

Acknowledgment

Thank you to Andrew Ng for teaching CS230 at Stanford and helping us grasp these deep learning concepts. We also appreciate the guidance and support from our project mentor Jonathan Li. We would also like to thank Prof. Ilya Strebulaev, Professor of Finance at the Stanford Graduate School of Business, for providing us with access to a comprehensive database of venture capital investments that formed the foundation of our research.

References

[1] Abadi, Martín, Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … others. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (pp. 265–283).

[2] Chollet, F. & others, 2015. Keras. Available at: https://github.com/fchollet/keras.

[3] Gage, Deborah. “The Venture Capital Secret: 3 Out of 4 Start-Ups Fail.”*The Wall Street Journal*, Dow Jones & Company, 20 Sept. 2012, www.wsj.com/articles/SB10000872396390443720204578004980476429190.

[4] Gornall, Will, and Ilya A. Strebulaev. “A Valuation Model of Venture Capital-Backed Companies with Multiple Financing Rounds.” *SRN*,5Nov.2020,papers.ssrn.com/sol3/papers.cfm?abstract_id=3725240.

[5] Nakagawa, Alex, et al. “vc_holy_grail-1.” *GitHub*, 7 Dec. 2017, github.com/Annyou/vc_holy_grail-1.

[6] Oliphant, T. E. (2006). *A guide to NumPy* (Vol. 1). Trelgol Publishing USA.

[7] Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830

[8] Mary Ann Azevedo, news.crunchbase.com/news /austin-reaches-top-10-in-us-venture-markets-with-record-funding-in-2019/, *13*(Jan), 2020

APPENDIX

Available columns provided by dataset:

company_id, company_name_x, familiar_name, previous_name, exchange, ticker, employee_count, year_founded, business_status, ownership_status, financing_status, universe, website, financing_note, full_description, total_raised_to_date, valuation_revenue, primary_contact_first_name, primary_contact_last_name, parent_company, sister_companies, subsidiary_companies, sic_codes, morningstar_codes, naics_codes, cik_code, customers, market, competition, products, performance_as_of_date, stock_price, average_volume, shares_outstanding, previous_close, price_percent_change_1_week, price_percent_change_4_weeks, beta, X52_week_range_low, X52_week_range_hi, market_cap_tso, eps, p_e, last_updated_x_x, company_name_y, location_id, location_name, address_1, address_2, city, state, zip, country, location_type, location_status, office_phone, office_fax, last_updated_y_x, company_name, deal_id, deal_number, announced_date, deal_date, deal_size, pre_valuation, post_valuation, post_valuation_status, ceo_first_name, ceo_last_name, deal_status, deal_class, deal_type_1, deal_type_2, deal_type_3, stock_type_x, stock_series, conversion_ratio_x, stock_split_x, percent_acquired, price_per_share_x, raised_to_date, add_on, total_debt, assumed_liabilities, debt_type_1, debt_type_2, debt_type_3, debt_amount_1, debt_amount_2, debt_amount_3, deal_synopsis, financials_period, financials_ending, debt_raised_in_round_ebitda, debt_raised_in_round_equity, deal_size_ebitda, valuation_ebitda, deal_size_ebit, valuation_ebit, deal_size_net_income, valuation_net_income, deal_size_revenue, deal_size_cash_flow, valuation_cash_flow, implied_ev_ebitda, last_updated_y_y implied_ev_ebit, implied_ev_net_income, implied_ev_revenue, implied_ev_cash_flow, total_revenue, gross_profit, net_income, ebitda, ebit, diluted_eps_net_income, total_current_assets, total_long_term_assets, cumulative, total_assets, total_current_liabilities, total_long_term_liabilities, total_liabilities, total_shareholders_equity, total_liabilities_and_equity, Ebitda_margin, book_value, lt_debt_lt_capital, basic_weighted_average_shares, diluted_weighted_average_shares, lt_debt_total_capital, implied_ev, cash_from_operating_activities, participating, cash_from_investing_activities, cash_from_financing_activities, change_in_cash, cf_net_income, debt_ebitda, debt_equity, liquidation, revenue_percent_growth, ebitda_percent_growth, ebit_percent_growth, net_income_percent_growth, last_updated_x_y, captable_id, series, stock_type_y, price_per_share_y, shares_sought, shares_acquired, conversion_ratio_y, stock_split_y, liquidation_preferences, dividend_rights, anti_dilution_provisions, board_voting_rights, general_voting.rights, shares_authorized, par_value, dividend_rate, original_price, liquidation_pref_mutliple, conversion_price, percent_owned

Final featured used within the model:

1. `employee_count`: number of employees
2. `year_founded`: year company was founded
3. `deal_number`: unique deal identifier
4. `percent_owned`: percent acquired through round
5. `percent_aquired`: percent acquired by investors in the round
6. `pre_valuation`: pre-money valuation in round
7. `raised_to_date`: capital previously raised by company
8. `deal_size`: amount raised, in Millions
9. `price_per_share`: price per share in round
10. `post_valuation`: post money valuation in round
11. `business_status`: stage of company, Options include: Clinical Trials, Product Development, Generating Revenue, Profitable, Out of Business.
12. `ownership_status`: who owns the company. Options include: Publicly Held, Privately Held (backing), Acquired/Merged, Out of Business
13. `financing_status`: universe: sic_codes:
14. `naics_codes`: industry code dictated by the North American Industry Classification System
15. `state`: country of company headquarters
16. `stock_type_x`: type of stock. Options include: Preferred, Participating Preferred, Combination or Common.
17. `deal_status`: status of deal. Some options include: Completed and Announced.
18. `deal_class`: type of deal. Some options include: Venture Capital and Individual.
19. `deal_type_2`: type of round. Options include: Series A, Series B, Series C, etc.
20. `website`: company URL
21. `parent_company`: parent company of current company
22. `tech_hotspot`: yes/no if company in a array of tech hotspots, sourced from Crunchbase's list of top recipient cities of venture capital funding [7] and a subjectively sourced list of cities that may serve as ancillary cities to the top ten cities identified ("New York", "San Francisco", "San Mateo", "San Jose", "Menlo Park", "Mountain View", "Boston", "Cambridge", "Seattle", "Berkeley", "Palo Alto", "Stanford", "Chicago", "Sunnyvale", "Redwood City", "South San Francisco", "Millbrae", "Austin","Evanston", "Raleigh", "Durham", "Cupertino")
23. `country`: country of company headquarters
24. `name`: company name
25. `board_voting_rights`: yes/no if board can vote
26. `sister_companies_count`: number of sister companies
27. `subsidiary_companies_count`: number of subsidiaries
28. `customers_count`: number of customers
29. `market_count`: market the company operates within
30. `competition`: major competitors of the company
31. `products`: What products the company sells
32. `business_status`: Business status. Options include Generating Revenue, Out of Business, Startup, and Profitable.
33. `Elapsed_announced_deal:` Time elapsed between announcement date and deal date.