Predicting Startup Success

Milestone 4 - Conclusion, Results & Project Report

Michael Van Vuuren
Data Science
University of Colorado, Boulder
miva6981@colorado.com

William Hinkley
Data Science
University of Colorado, Boulder
wihi1131@colorado.edu

Melina Kopischkie
Data Science
University of Colorado, Boulder
meko4807@colorado.edu

ABSTRACT

We offer insights for investors and startups by leveraging data science to forecast startup success. Data from OurCrowd and CB Insights was collected and processed to train machine learning models, including k-means clustering, decision trees, random forests, logistic regressions, and support vector and k-nearest neighbor regressions. The decision tree and neural network models excelled in classifying startups into clusters found with k-means. The SVR achieved high precision with an R-squared value of 0.997. The other models had lower performance, because features were dropped to avoid look-ahead bias, but still did a decent job at classifying startups using limited information. The final dataset used in training integrated financial, sentimental, funding, and industry information.

INTRODUCTION

Venture capitalists and other investors have typically relied on their expertise and experience to determine the startups with the greatest return on investment. Data science is already widespread in the financial world, with applications including market trend forecasting, customer segmentation, fraud detection, risk analysis, predictive analytics, and customer sentiment analysis. We leveraged data science to forecast the potential of success for startups. We followed a multistep approach. First, we found websites with information about startups and scrape relevant data. Then, we preprocessed this data to prepare it as input to machine learning models. We used text sources such as news articles to integrate sentiment into our models. After training the models, we isolated the most relevant factors for startup success. Using that information, we provided both investors and startup owners with key metrics for success. Prospective investors can use our models to determine the best investments, while startups gain insights into key factors that should be prioritized for growth.

In this report, we answer some key questions, including which measurements best predict startup success potential, how much sentiment impacts startup success, whether the industry of a startup affects its success potential, how industry information can be harnessed to predict startup performance, and more.

RELATED WORK

Several research papers with differing data sources and methods have used machine learning to predict startup success.

One paper examined time-independent factors associated with startup success using an *XGBoost* model on publicly available information. The study identified factors like entrepreneurial experience and educational background as significant predictors of IPOs, mergers, or acquisitions [1].

Another study focused on predicting both post-money valuation and the likelihood of success (e.g., acquisition or IPO) for startups by combining structured data from Crunchbase with natural language processing. The authors achieved high accuracy in predicting valuations. Additionally, they used a neural network to estimate success [2].

A third study explored using web-based open sources alongside structured data from startup ecosystems, such as databases of investors and incubators, to predict whether a startup would secure additional funding. This approach highlighted the value of web signals, particularly companies' mentions on the web, which significantly enhanced model performance when combined with structured data. The authors made a gradient boosting pipeline and conducted an analysis of feature importance [4].

Lastly, a paper addressed bias in success prediction models, emphasizing the importance of preventing look-ahead bias by excluding data unavailable at the decision moment. Using Crunchbase data, the authors compared logistic regression, support vector machines, and gradient boosting classifiers. Despite limiting predictors, they achieved promising precision, recall, and F-1 scores with the gradient boosting classifier. The study revealed that region, country, and industry were the most influential features [6].

We hope to add to these studies by using new data sources besides Crunchbase, factoring in sentiment, and current data. We also will avoid look-ahead bias as mentioned in the

last paper by removing circular features, so that our model has real-world applicability.

METHODS

1 Overview

To gather the required data, we utilized web scraping techniques on sources such as OurCrowd and CB Insights. Using Python's *BeautifulSoup* library, we accessed financial data for startup companies, including metrics like total revenue, valuation, mosaic change, and investor count. Sentiment scores for articles about the startups were included. The scraping process involved sending requests to webpages to retrieve HTML/JavaScript documents, parsing them with Beautiful Soup, extracting relevant information using regular expressions, and storing the results in CSV files. Once scraping was completed, we compiled the data into three separate CSV files for preprocessing and cleaning, which is discussed in the next section.

After scraping and cleaning, we had three datasets. While scraping each website, we collected the homepage URL of the startup being scraped, so the integration process of the cleaned datasets was straightforward. We performed an inner join on the profile dataset with the financial dataset on the column website. Then, we performed another inner join using the column website on that combined dataset and the overview dataset. In the end, we created a primary dataset called 'primary.csv' under the 'clean-datasets/' directory. After integrating our cleaned datasets to create a primary dataset, we decided to create two mini-datasets also under 'clean-datasets/'. One contains the startups from primary with complete revenue information, and the other contains startups with complete valuation information. Then, the valuation and revenue information were dropped from the primary dataset.

After cleaning and integrating the data, we conducted Exploratory Data Analysis (EDA) to construct visualizations that revealed correlations between key variables. These insights informed our model selection process. We used the processed data to train machine learning models, including k-means clustering, decision trees, and support vector regression, to classify high-earning startup companies and predict total valuations. During training, we fine-tuned hyperparameters using validation sets to optimize performance. Model accuracy was evaluated using metrics such as the F-1 score, the area under the ROC curve, confusion matrices, and the R^2 value, depending on the specific model.

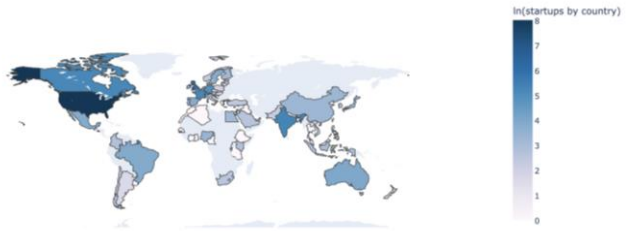


Figure 1: Company locations by country

2 Exploratory Data Analysis

Before building machine learning models, data obtained via webscraping was cleaned, integrated, and visualized. This exploratory process revealed key correlations between variables and identified the direction the models should follow. One of the most interesting visualizations illustrated the relationship between a companies' last funding date and the mosaic change. Mosaic change is a metric designed by CB Insights, one of our data sources, to evaluate long-term success of startups. From the violin plot below, startups who recently received funding had a positive mosaic change, which intuitively makes sense.

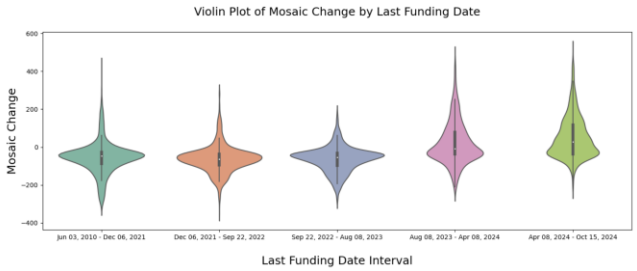


Figure 2: Mosaic change by last funding date bin

Another visualization shows that the majority of startups receive Series and Seed funding.

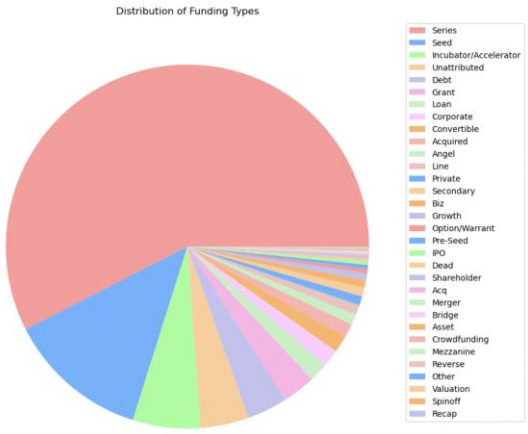


Figure 3: Pie chart of funding types

Lastly, the plot below shows that China and the United States lead in maximum funding awarded for a single round. The

chart uses a log scale, so the true differences in maximum funding are larger than they seem.

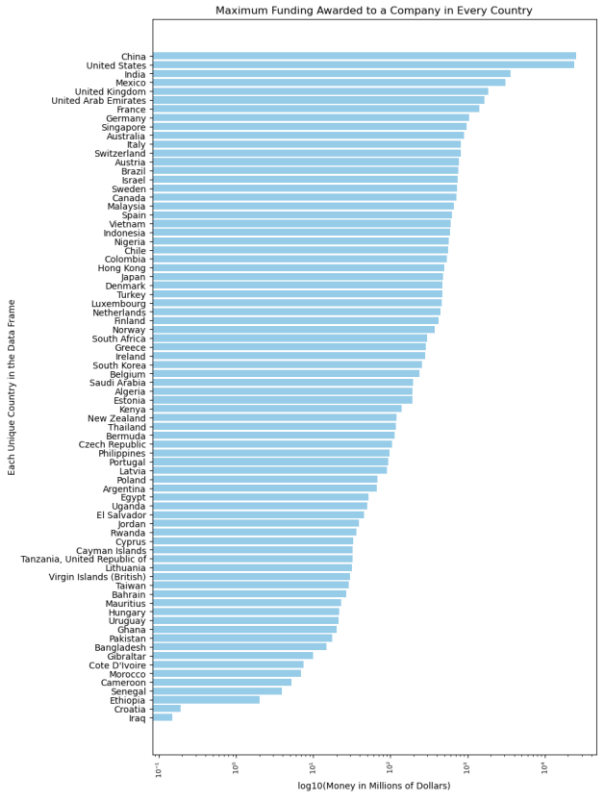


Figure 4: Maximum funding by country

PREPROCESSING

Prior to model training, data was preprocessed. Data was sent through a pipeline to be cleaned, encoded, augmented, balanced, and scaled. Preprocessing was performed on the training set and test set. For encoding and scaling, encoders and scalers were fitted on training data, then the training and test sets were transformed accordingly. This prevented data leakage. Steps 2-6 cover the preprocessing steps performed for the classification models (Sections 3 & 4); however, the same process was used prior to training the K-Means model, the Neural Network, and the regression models.

1 Cleaning

Three datasets 'financial.csv', 'profile.csv', and 'overview.csv' were obtained using webscraping. Cleaning the datasets was a two-step process. First, missing values were imputed or dropped, then default data types were converted to their appropriate types.

Imputation was performed on numerical features based on the number of null values present. If the null values made up less than 25% of the total rows, the missing values were imputed. Each dataset had 5000-7000 rows, so many values were imputed. If the distribution of a missing value's feature column was moderately-to-strongly skewed, the median was used to negate outliers and skewness. Otherwise, the mean or mode were used. In most cases the median was used.

Type conversion was done for efficiency and ease of use. First, object types were narrowed. For example, categorical types such as type of funding received were converted into category types to improve space efficiency. Then, integer types such as investor count were changed to Int64 types from float64 types. Finally, date and time information was converted to datetime types and then stored as formatted string types.

A step-by-step explanation of the cleaning process can be found in 'cleaning.ipynb' under the 'cleaning/' directory: https://github.com/WiHi1131/Data-Mining- Project/

2 Loading

The clean dataset was loaded and split into a feature matrix and a target vector, each of which was split into training and test sets.

```
X_train.head()
```

	name	tagline	summary	description	year_founded	website	city	region	country	postal_code
3463	Kinetic	Protecting Your ...	Kinetic is a com...	Kinetic speciali...	2014	https://wearkine...	New York	New York	United States	10001
4561	Grow Credit	Elevate Your Cre...	Grow Credit is a...	Grow Credit offe...	2018	https://growcred...	Santa Monica	California	United States	90401
3968	Retirable	Design Your Drea...	Retirable is a c...	Retirable is a c...	2019	https://retirabl...	New York	New York	United States	10003
5350	Flow Neuroscience	Elevate Your Min...	Flow Neuroscienc...	Flow Neuroscienc...	2016	https://flowneur...	Malmo	NaN	Sweden	211 40
2052	thirdweb	Unleashing the F...	Thirdweb is a Ca...	Thirdweb is a fu...	2020	https://thirdweb...	San Francisco	California	United States	94123

Figure 5: Incomplete snippet of initial training dataset

```
y_train.value_counts()
```

```
2    2379
1    1355
0     120
Name: cluster, dtype: int64
```

Figure 6: Incomplete snippet of initial test dataset

3 Encoding

Next, categorical features were encoded into numerical features. The cardinality of each categorical feature (count of unique values) determined how each feature was encoded.

```
X_train.select_dtypes(exclude=['number']).nunique()
```

```
name                3799
tagline             3757
summary             3799
description         3799
website             3799
city                 659
region               153
country               70
postal_code         1856
concepts            3732
keywords            3744
last_funding_type    131
last_funding_date   1168
sentiment              3796
articles               3799
dtype: int64
```

Figure 7: Cardinality of each categorical feature

Feature	Encoding	Reasoning
`name`, `tagline`, `summary`, `description`, `website`, `concepts`, `keywords`, `sentiment`, `articles`	Drop	Not useful for model
`city`, `postal_code`	Frequency encoding	High cardinality
`region`, `country`, `last_funding_type`	Label encoding	Moderate cardinality
`last_funding_date`	Split into day, month, year	Easier to deal with

Figure 8: Encodings

4 Augmenting

Augmenting the feature matrix was tested but augmentations did not significantly improve model performance.

```
X_train_augmented = X_train_encoded.copy()
X_test_augmented = X_test_encoded.copy()

augment = False

if augment:
    for df in [X_train_augmented, X_test_augmented]:
        df['funding_per_investor'] = df['funding_total_millions'] / df['investor_count']
        df['funding_per_round'] = df['funding_total_millions'] / df['funding_count']
        df['last_funding_since_founded'] = df['last_funding_year'] - df['year_founded']
```

Figure 9: Example of augmentations that were explored

5 Balancing

The data was balanced such that each class in the target vector had roughly the same number of samples. Oversampling was performed using SMOTE. See

https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

```
X_train_balanced, y_train_balanced = preprocessor.balance_data(X_train_augmented, y_train, X_test_augmented, ratio=1/3)

Class distribution before SMOTE:
Class 2: 2379
Class 1: 1355
Class 0: 120

Class distribution after SMOTE:
Class 2: 2379
Class 1: 1355
Class 0: 793
Balanced data!
```

Figure 10: Data balancing

6 Scaling

Each feature was scaled appropriately. For distance-based models such as K-Nearest Neighbors and Support Vector Machines, scaling improves model performance and efficiency. Using a scale previewer, a feature and its scaling methods were selected.

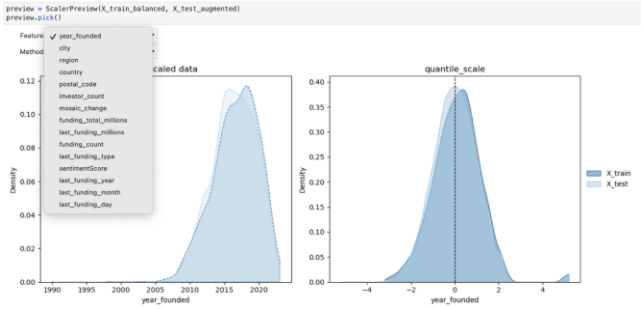


Figure 11: *year_founded* is normalized using quantile scaling

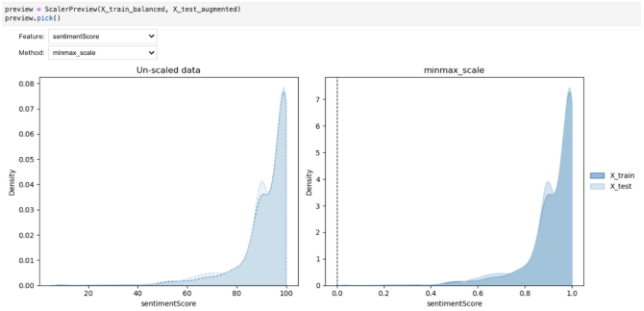


Figure 12: *sentimentScore* is min-max scaled between 0 and 1

Feature	Scaling	Reasoning
funding_total_millions, last_funding_millions, funding_count	Log Scale	Heavy right skew
year_founded	Standard Scale	Nearly normal
mosaic_change	Robust Scale	Nearly normal with outliers
region, postal_code, investor_count, last_funding_type, sentimentScore	Yeo-Johnson Scale	Brings skewed and multimodal distributions closer to normal
city, country, last_funding_year, last_funding_month, last_funding_day	Min-Max Scale	Brings distributions between 0 and 1

Figure 13: Scalings

```
X_train.head()
```

	year_founded	city	region	country	postal_code	investor_count	mosaic_change
0	-0.738319	0.868263	0.507421	0.959459	1.706064	0.414914	-0.634146
1	0.508137	0.035928	-1.151447	0.959459	0.476535	1.529194	0.000000
2	0.819751	0.868263	0.507421	0.959459	1.893561	-0.479262	1.146341
3	-0.115091	0.005988	1.193066	0.837838	-0.627624	-0.479262	0.000000
4	1.131365	1.000000	-1.151447	0.959459	0.476535	0.750441	-0.341463

Figure 14: Incomplete snippet of preprocessed training dataset

SECTION 1: K-Means Clustering

1 MODELING

Much of the scraped data lacked valuation information. It was also worthwhile for our analysis to explore other potential metrics for evaluation and prediction. To accomplish this, it was decided to perform a K-means clustering of the data. From this clustering, we hoped to identify groups of startups that had a combination of favorable characteristics that could substitute for a valuation and still indicate through classification which startups could be more favorable to investors.

Only numerical columns were used for clustering. The numerical sentiment score for each startup was extracted and outliers were dropped before clustering was performed; standard scaling was also performed as a best practice.

Several values of K were tested. The Elbow Method was used to try and find an optimal value for K. Distortion values were plotted at different values of K in Figure 15.

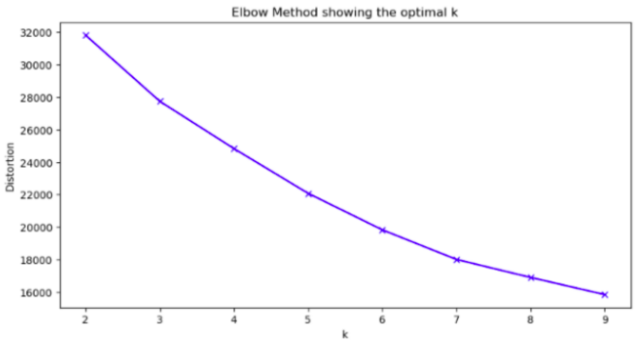


Figure 15: K-means elbow plot

Figure 15 does not show a definitive value of K for which the slope of the curve of distortion values drastically decreases from one value of K to the next. Instead, the decrease in distortion is fairly gradual. The largest decrease in the slope of the curve appears to occur between values of 2 and 3 for K, and between 6 and 7 for K. Since the goal of our clustering was to find classes for prediction, a smaller value of K was

decided to be more useful for subsequent predictive models, so a K value of 3 was chosen.

2 EVALUATION

With our chosen K-value of 3, our clustering divided the data according to the results from the elbow plot.

```
#Printing the number of values in each cluster
print(df['cluster'].value_counts())
```

```
2    3414
1    1940
0     153
Name: cluster, dtype: int64
```

Figure 16: Three groups of clusters

Using principal component analysis, the dimensions of the data were reduced to two for visualization purposes, and a plot of the data colored by the clustered groups was created in Figure 17.

Figure 17: K-Means clustering result

With a goal of identifying interesting clusters, there were no accuracy metrics generated by this clustering. We can see from Figure 17 that Cluster 0 appears to contain data points high in both values of the principal components, as opposed to the other two clusters.

With three clusters identified by the model, it was at our discretion to select any interesting 'target' clusters worth predicting with other models.

3 RESULTS

Based on the visualization of the clusters in Figure 17, Cluster 0 seemed the most interesting in terms of prediction. Figure 18 shows the mean values of numerical attributes in each cluster.

```
#Visualizing means of values within each cluster
cluster_profiles = df.groupby('cluster')[numerical_cols].mean()
print(cluster_profiles)
```

```
         year_founded  investor_count  mosaic_change  funding_total_millions  \
cluster
0         2015.209150       19.686275     -17.058824             1160.205882
1         2014.214433       15.649485     -37.505670              130.425923
2         2018.084359        8.507616     -13.135325               43.507496

         last_funding_millions  funding_count  sentimentScore
cluster
0                   342.752026       8.607843       85.032680
1                    33.632320       8.446392       92.418557
2                    21.976216       3.888694       92.707674
```

Figure 18: Average feature values within each cluster

From examination of the means, we identified exactly what makes Cluster 0 so interesting as a 'target' class of startup. We can see it has a similar year_founded value as Cluster 1, but has higher financial metrics across the board, containing the highest average funding, the highest average last funding amount, the highest funding count, and the most investors. It contains a smaller absolute mosaic change value than cluster 0 as well. Interestingly, Cluster 0 also contains the lowest average sentiment scores among the three clusters. These mean values identify Cluster 0 as having remarkable attributes that both distinguish it from the other two clusters and distinguish it as a worth 'target' cluster for prediction. Startups in this cluster will likely contain attributes that investors would find worthy of their investment dollars.

SECTION 2: Cluster Classification via Neural Network

1 MODELING

After performing clustering, our goal was to create models that would be able to predict which clusters startups would belong to, since clusters contained information relevant for investment decisions. Our first approach was to create a Neural Network using TensorFlow and Keras for this predictive task. Only numerical attributes were used, and data was scaled and preprocessed.

```
#neural network model using keras. Note the very basic model with only three layers and fairly standard
#activations for each
model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    layers.Dense(32, activation='relu'),
    layers.Dense(3, activation='softmax')  #since there are 3 classes
])

#compiling the model with a typical optimizer and loss function
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

#training the model on the data
history = model.fit(
    X_train_scaled, y_train,
    epochs=9,
    batch_size=32,
    validation_split=0.1,          #Note a validation set is used
    verbose=1
)
```

Figure 19: Neural network implementation

The model contained two hidden layers and an output layer for use as a classifier to predict the three classes corresponding to the K-means clusters we found previously. A

standard ReLU activation function for the hidden layers was used, along with a softmax function for the output layer, was used as simple baseline functions suitable for classification tasks. We omitted more complex techniques such as regularization, dropout, and batch normalization, since this was meant to be a baseline model that could potentially be optimized later.

The architecture of the model is simple. The first ReLU layer contains 64 neurons, the second hidden layer contains 32 neurons, and the output layer contains 3 neurons. The Adam optimizer and sparse categorical cross-entropy loss functions were chosen, again, as baseline options since these are both commonly used. The sparse categorical cross entropy function was able to be used here since the labels for the clusters were provided to the model as integers.

2 EVALUATION

```
#model evaluation for each epoch
test_loss, test_accuracy = model.evaluate(X_test_scaled, y_test, verbose=0)
print(f'Test Accuracy: {test_accuracy:.4f}')

/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/9
124/124 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 0.6785 - loss: 0.7923 - val_accuracy: 0.9524 - val_loss: 0.1884
Epoch 2/9
124/124 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9791 - loss: 0.1314 - val_accuracy: 0.9615 - val_loss: 0.1111
Epoch 3/9
124/124 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9824 - loss: 0.0793 - val_accuracy: 0.9728 - val_loss: 0.0761
Epoch 4/9
124/124 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9866 - loss: 0.0625 - val_accuracy: 0.9887 - val_loss: 0.0515
Epoch 5/9
124/124 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9881 - loss: 0.0497 - val_accuracy: 0.9909 - val_loss: 0.0479
Epoch 6/9
124/124 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9911 - loss: 0.0406 - val_accuracy: 0.9819 - val_loss: 0.0462
Epoch 7/9
124/124 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9904 - loss: 0.0425 - val_accuracy: 0.9887 - val_loss: 0.0385
Epoch 8/9
124/124 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9925 - loss: 0.0290 - val_accuracy: 0.9909 - val_loss: 0.0355
Epoch 9/9
124/124 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9929 - loss: 0.0294 - val_accuracy: 0.9955 - val_loss: 0.0333
Test Accuracy: 0.9873
```

Figure 20: Neural network training

Figure 20 shows our monitoring of accuracy of the neural network during training and evaluation. Nine epochs were chosen after initially trying 30 and noticing an area where the loss function ceased to decrease, which was around nine epochs. Note that a validation set of size one-tenth of the dataset was implemented to help prevent overfitting.

Figure 21 shows the confusion matrix for the model's performance on the test dataset.

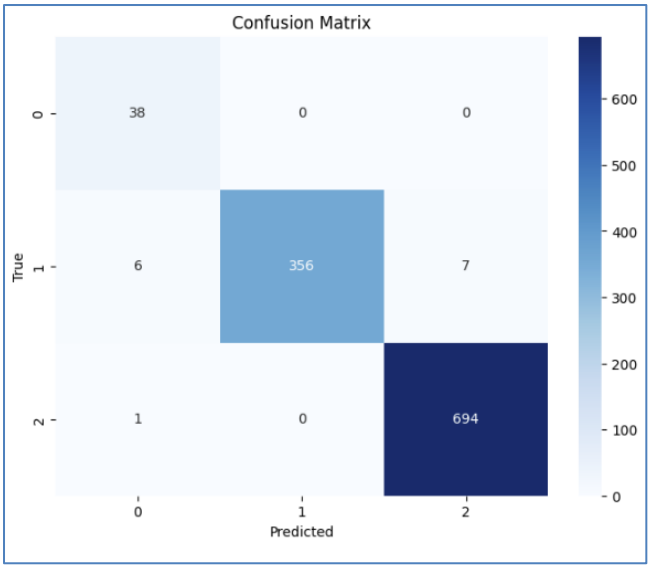


Figure 21: Neural network confusion matrix

3 RESULTS

Surprisingly, we saw an extremely high rate of accuracy of prediction even though no class balancing was performed on the data and despite the simplistic nature of the neural network. Note the final accuracy on the test set was 0.9873. We can also examine the confusion matrix in Figure 21. We can see the only errors are 6 predictions that an item would be in class 0, but was in fact in class 1, 7 items predicted to be in class 2, but were actually in class 1, and only 1 item predicted to be in class 0, but was actually in class 2. Since Class 0 is our interesting class and the one that would likely yield startups most likely to receive large amounts of funding, we likely don't care much about incorrect predictions of class 2 items that were in actuality in class 1. We do see that this neural network tends to place some items in class 0 that do not belong there – considering the rarity of a startup actually belonging to class 0, this may be a bit concerning should this model be introduced to new data (7 wrong predictions out of 45 predicted to be in Class 0 would give us a 15% likelihood that our model might tell us to invest in a Class 0 company that is not actually classified as one).

Of course, since we are predicting clusters and not actual performance of a company, it is entirely possible that a Class 1 or 2 company might contain some metrics like funding amount that would be favorable, even if they are not within our 'interesting' cluster. And, regardless, an 85% accurate prediction for Class 0 is still very good considering how imbalanced the classes are in this case.

Overall, our model performs extremely well without the need for class balancing or adjustment of a very simple neural network. With such an excellent accuracy score, we should be wary of possible overfitting. The best way to uncover

overfitting would be to find out how well this model performs on brand-new, unseen data.

SECTION 3: Cluster Classification via Decision Tree Classifier

1 MODELING

The goal was to predict clusters found with K-Means using classification on 'primary.csv'. Given an unseen sample, its cluster can be predicted based on its features. This section was not necessarily applicable to real world problems, because 1) the sample could have simply been added to the dataset then clustered with K-Means, and 2) too many features were used in training making the model less applicable to unseen companies without those features. Nonetheless, this section tested the processing pipeline and model outputs.

The model employed was a decision tree; it was interpretable and tested data viability. Initially, the maximum depth was set to three. There were three cluster classes, so One-vs-Rest was used when making the ROC Curves.

2 EVALUATION

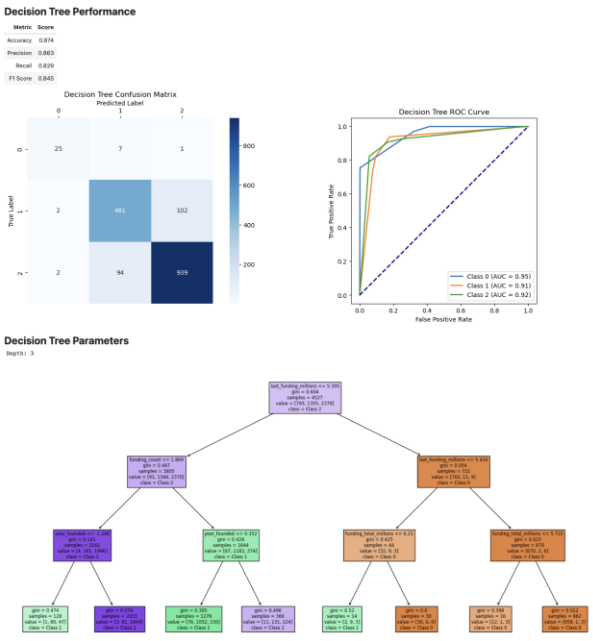


Figure 22: (a) Decision tree with depth of 3

The F-1 Score for the decision tree with a depth of three was already quite high. The feature that resulted in the greatest information gain when split on was *last_funding_millions*. Information about funding and company founding were the features the decision tree used to predict the cluster a company belonged to. After hyperparameter tuning, the F-1 Score increased further.

Hyperparameter tuning

```
dt_param_grid = {
    'max_depth': [3, 5, 10, 15, None],
    'min_samples_split': [2, 50, 100],
    'min_samples_leaf': [1, 5, 10]
}

dt_best_params = dt_trainer.tune_hyperparameters(dt_param_grid, cv=5)
```

```
Best parameters found: {'max_depth': 10, 'min_samples_leaf': 5, 'min_samples_split': 2}
```

Figure 22: (b) Hyperparameter tuning parameters

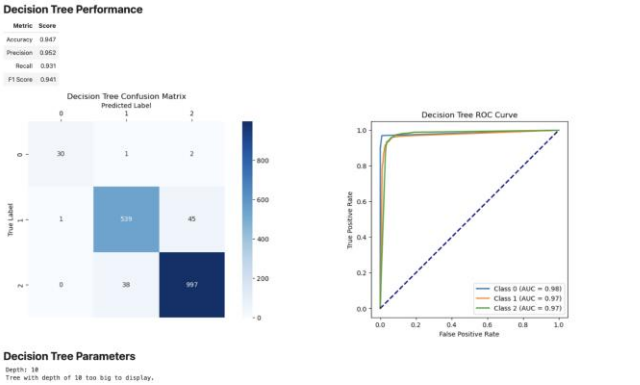


Figure 22: (c) Tuned decision tree with depth of 10

3 RESULTS

Using a decision tree on the primary dataset to predict the cluster a given company belonged to was highly effective. An F-1 Score of 0.941 is excellent for a multiclass classifier, and an AUC near 1 for all the ROC curves shows that this model is well suited to predicting clusters.

This proved the viability of the data for modeling and developed a preprocessing pipeline. The next two sections use companies from the primary dataset that had valuation information. The models were created to be more useful for real-world inference, because circular features, such as total funding, were dropped during training. Instead, the models were trained using information that newer companies would have access to, such as industry, location, year founded, and latest funding.

SECTION 4: Binned Valuation Classification via Random Forest Classifier and Logistic Regression

1 MODELING

While the decision tree was successful, it required too much information to infer company success for unseen, real-world companies. This is because the features used were either unavailable or circular in nature for most companies. In this section, each sample from 'primary-with-valuation.csv' was binned according to the valuation amount feature. The first model, which was a random forest, used two bins for

classification, while the second model, which was a logistic regression, used three bins. In each case, the goal was to predict whether a company had potential for a lower or higher valuation. The latter case simply increased the granularity of the classification.

Since many features were dropped, the dataset became too minimal. To augment the dataset, the previously dropped keywords feature, which contains industry information for each company, was instead encoded such that the model could use industry area to aid in its predictions.

First, all the keywords in each list for each company were one-hot encoded using Scikit-learn's *MultiLabelBinarizer*. Then Scikit-learn's *TfidfTransformer* reduced the effects of this sparse one-hot encoded matrix. Next, K-Means clustering grouped similar points in the one-hot encoded vector space. Finally, keyword bins formed by K-Means were one-hot encoded.

	keywords	keyword_bins
432	[your list of company industry categories to c...	1
582	[edtech, fintech, application software, ai as ...	1
442	[fintech, platform as a service (paas), e-comm...	6
675	[software, procurement, saas management, it se...	4
334	[data analytics, business intelligence, saas, ...	7

Figure 23: Industry keyword clusters

Using the top 3 most common keywords in each bin, a *networkx* graph showed which keyword bins connect to which keywords. With this clustering algorithm, clusters were created that share conceptually related keywords, ie. Bin 0 has Cybersecurity, Software Development, and Information Technology, while Bin 6 has E-commerce, Retail, and Technology.

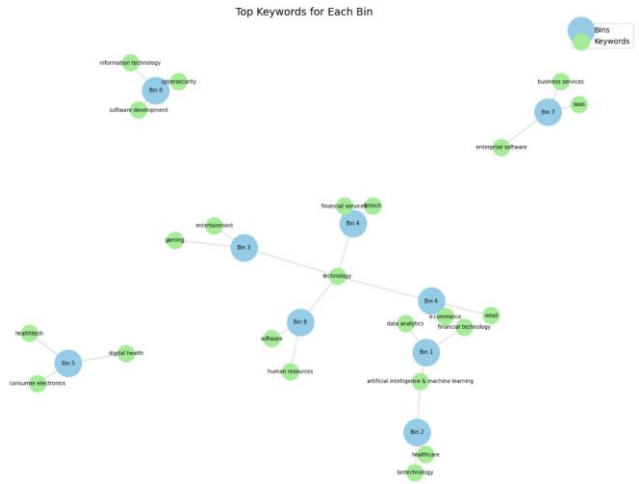


Figure 24: Graph of common keywords in each cluster

The model was trained using these one-hot encoded keyword bins in addition to other parameters. A random forest model was used because it avoided overfitting and had the best performance.

As mentioned earlier, for the random forest model, the valuation amounts were divided into two bins, low and high.

For the logistic regression model, the valuation amounts were divided into three bins, low, medium, and high.

2 EVALUATION

First, the random forest model on two bins:

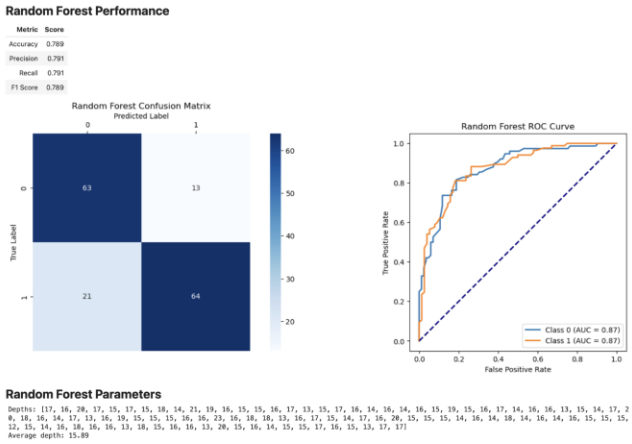


Figure 25: (a) Randon forest with 100 trees

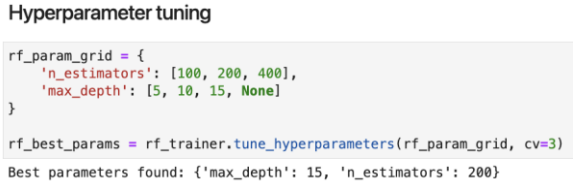


Figure 25: (b) Hyperparameter tuning parameters

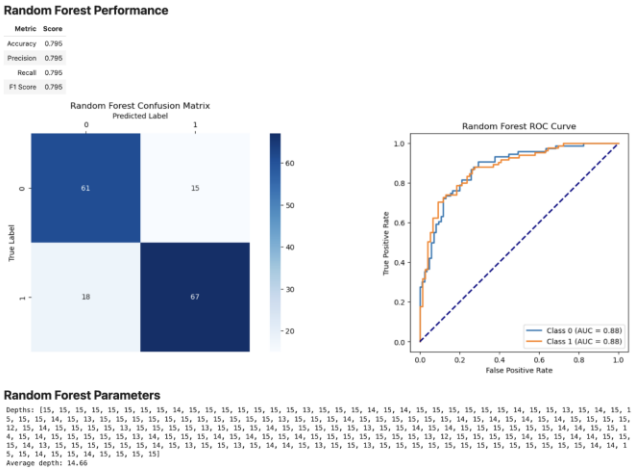


Figure 26: (c) Tuned random forest with 200 trees

Second, the tuned logistic regression model on three bins:

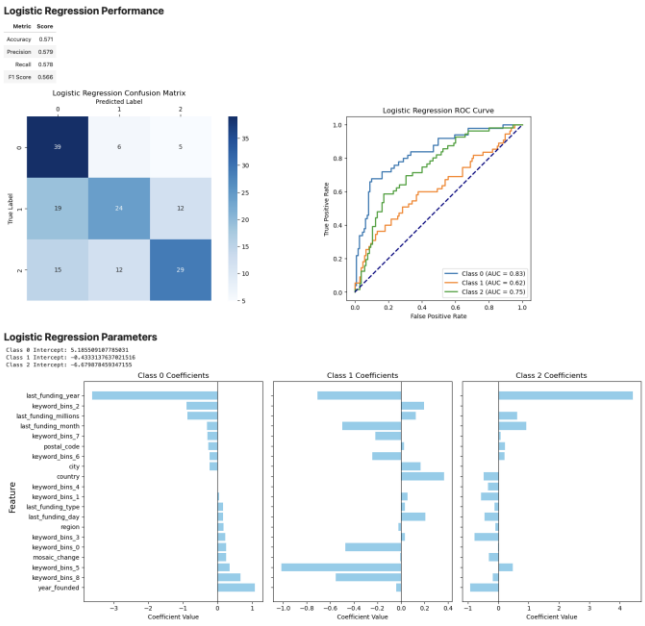


Figure 27: Tuned logistic regression model

Class 0, Class 1, and Class 2 correspond to low, medium, and high valuations. The tuned logistic regression model is more likely to classify companies in Bin 8 as Class 0 (lower valuation). Bin 8 includes software, technology, and human resources. Compared to other bins, it makes sense that human resources would have lower valuations. Additionally, the model is more likely to classify companies in Bin 5 as either Class 0 or Class 2. It is much less likely to classify them as Class 1. In other words, the model sees Bin 5, which includes digital health, consumer electronics, and healthtech as more likely to either have low or high valuations, with fewer in the middle.

3 RESULTS

The random forest model was good at deciding whether a company would have low or high valuation based on the modified feature set. Given an unseen company, with moderate confidence it could be determined whether it had potential for a low or high valuation.

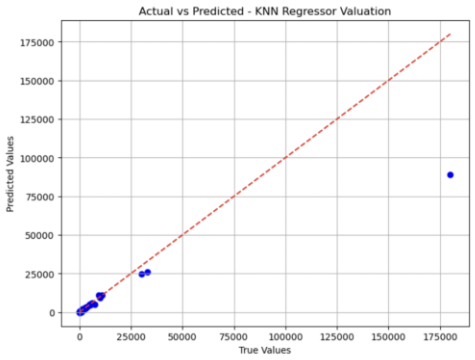
The logistic regression model did an okay job of deciding whether a company would have low, medium, or high valuation. Given an unseen company, with some confidence it could be determined whether it had potential for a low, medium, or high valuation.

SECTION 5: Valuation Prediction via Support Vector and K-Nearest Neighbor Regressions

1 MODELING

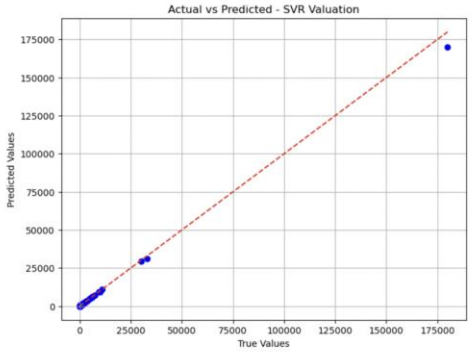
The goal of implementing regression machine learning techniques was to predict a startup companies' total valuation. The valuation refers to the estimated market value of a company at any given time. This information and the predictive technology could be beneficial to investors. Some of the features contributing to the prediction include the city, mosaic change, and the sentiment score.

A K-Nearest Neighbors (KNN) and Support Vector Regression (SVR) were built to predict the total valuation. KNN is a lazy learner and doesn't train itself, it simply looks at the nearest data points to make its predictions. Whereas the SVR maps the data into a higher-dimensional space to find the hyperplanes that separate the data from one another. SVR handles non-linearity and outliers more effectively, which is useful for this particular dataset.




```
Mean Absolute Error (MAE): 751.0012829743388
Mean Squared Error (MSE): 51766817.745064884
Root Mean Squared Error (RMSE): 7194.9161041019015
R-squared (R^2): 0.7552680231304131
```

Figure 28: KNN Regression




```
Mean Absolute Error (MAE): 111.03196638107684
Mean Squared Error (MSE): 637862.359483656
Root Mean Squared Error (RMSE): 798.6628572080061
R-squared (R^2): 0.9969844521450806
```

Figure 29: Support Vector Regression

2 EVALUATION

The graphs reaffirm that KNN models are less suitable for outliers than SVR models. To tune the hyperparameters for the KNN, an elbow plot was constructed and determined that two neighbors yielded the highest R2 value. However, we used

three neighbors in the final KNN model so that it wouldn't be as sensitive to noise or outliers.

As this model predicts the valuation of startup companies with data from the past five years, we didn't want to overtrain the model for future use. The following code was added to tune the parameters for the SVR model:

```
param_grid = {
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'C': [0.1, 1, 10, 100],
    'epsilon': [0.01, 0.1, 0.5, 1]
}

grid_search = GridSearchCV(SVR(), param_grid, cv=3, scoring='neg_mean_squared_error', verbose=1)
grid_search.fit(X1_train_scaled, y1_train)

print("Best Parameters:", grid_search.best_params_)
best_svr = grid_search.best_estimator_
```

```
Fitting 3 folds for each of 64 candidates, totalling 192 fits
Best Parameters: {'C': 100, 'epsilon': 1, 'kernel': 'linear'}
```

Figure 30: SVR Hyperparameter Tuning

Various kernels were tested with different C and Epsilon values to determine the best fit for the given dataset. The C value controls the trade-off between a low error on the training data and the complexity of the model. Epsilon defines a margin of tolerance for the prediction. A small Epsilon has a lower margin of tolerance to predict more precisely but is more prone to overfitting.

3 RESULTS

The KNN performed decently well with an R value of 0.755, but didn't handle the outlier as well. Omitting the outlier would have resulted in a much higher R^2 value, but we would have lost vital information as we want to predict the trends that result in a high startup valuation.

The SVR performed exceptionally well with the outlier and yielded an R^2 of 0.997. The increase in performance indicates that the data used may have complex correlations that are best defined in higher dimensions. Both models contribute to an accurate prediction of business valuation, and as more training data is added, the more accurate the models will be.

CONCLUSION

Predicts:	Model	Accuracy	Precision	Recall	F-1
Cluster (3 clusters)	Neural Network	0.987	N/A	N/A	N/A
Cluster (3 clusters)	Decision Tree	0.947	0.952	0.931	0.941
Valuation (2 bins)	Random Forest	0.789	0.791	0.791	0.789
Valuation (3 bins)	Logistic Regression	0.571	0.579	0.578	0.566
		MAE	**MSE**	**RMSE**	**R^2**
Value (continuous)	KNN Regression	751	5e7	7e3	0.755
Value (continuous)	SVR Regression	111	6e5	799	0.997

Table 1: Summary of model results

Predicting the K-means clustering-generated classes was done best by our simple neural network model, but the decision tree model did only slightly worse; both models were very accurate at their classification tasks with high accuracy rates. Notably, the neural network used less features than the decision tree but performed slightly better in accuracy. Without consideration of complexity, the decision tree shows more interpretable information, whereas the neural network's inner workings are more complex, and its better performance cannot necessarily be explained easily. Because the decision tree performs almost as well, this may have an advantage over the neural network, especially if we want to capture subtleties of more complex data with location information included. Potential for overfitting exists for both models, due to their extremely high scores; they may not perform as well on newly scraped data.

For predicting company valuation in terms of classification bins, the random forest model bested the best accuracy, precision, recall, and F-1 scores, all being just below 0.8. These were much better than the logistic regression scores, which hovered around 0.57 for all scores. This could be expected, since random forest models are known for being better suited at capturing complex patterns. We knew from data exploration that none of the numerical variables showed very linear relationships with one another, so it is within our expectations that the logistic regression performed as our poorest model.

Our SVR model was our most robust model for predicting company valuation numerically, with an extremely high R-squared value of 0.997. The KNN regression did not perform as well, with an R-squared value of 0.755. SVR is known to perform better on more complex datasets, so this difference in R-squared values could be expected. One of KNN regression's advantages is its computational efficiency, but in this case our dataset was very small (the startup data with valuation information only had approximately 800 rows), so using KNN over SVR has no merit in this case where performance differs so greatly. With the high R-squared value for the SVR, we should be wary of overfitting once more. The model may perform poorly on newly scraped data.

With the complex nature of the problem, we have provided investors with multiple options for models to choose from to best suit their goals. If valuation information is ambiguous or not particularly useful as a metric of future success, as it likely would not be for many industries and to certain investors, users of our models should feel secure that we have used K-means clustering to identify startups with characteristics indicative of success, and that either our neural network model or our decision tree model would predict whether a startup should be associated with these unique companies or not with high accuracy. If valuation information is useful to predict, we recommend our SVR to

find the value of a startup with extremely high precision, and our Random Forest model as a classifier if exact numerical outputs may not be useful or as a corroboration to the SVR results.

FUTURE WORK

The key next step to truly test the efficacy of our models would be to scrape more data from OurCrowd, CB Insights, or other sources, and test how well our models perform with these new sources of information. Several rounds of scraping and testing would likely be required to smooth out any errors in predictive ability. If models could continue to be fed new data and improved, after many rounds of testing these models might be ready to be tested with real investment capital on real companies.

Our K-means clustering could also be improved upon or tested with different values of K, to narrow in on truly successful startups as opposed to nearly successful ones.

Examining the work of others in this field yields areas for exploration as well as confirmation of the methods we applied. Bidgoli et al employed many of the same machine learning techniques we did, achieving a best accuracy of around 80%, and utilized K-means clustering to help distinguish successful startup characteristics [3]. Gautam and Wattanapongsakorn achieved accuracy scores of around 80% using techniques such as XGBoost and LightGBM, but with a larger dataset of 2 million companies [5].

Undoubtedly, the task of predicting startup success is a difficult one, and even high accuracy scores from machine learning models cannot guarantee whether money will be gained or lost by venture capitalists who actually invest in them. However, our methods for finding and creating usable data were thorough, and our models provide an excellent jumping-off point for further analysis. With more robust data and iterative training of our models, we may be able to create tools relied upon by savvy investors in time.

REFERENCES

[1] Abhinav Ndah Thirupathi, Tuka Alhanai, and Mohammad M. Ghassemi. 2022. A machine learning approach to detect early signs of startup success. In Proceedings of the Second ACM International Conference on AI in Finance (ICAIF '21). Association for Computing Machinery, New York, NY, USA, Article 9, 1–8. https://doi.org/10.1145/3490354.3494374.

[2] Ang, Y.Q., Chia, A., Saghafian, S. (2022). Using Machine Learning to Demystify Startups' Funding, Post-Money Valuation, and Success. In: Babich, V., Birge, J.R., Hilary, G. (eds) Innovative Technology at the Interface of Finance and Operations. Springer Series in Supply Chain Management, vol 11. Springer, Cham. https://doi.org/10.1007/978-3-030-75729-8_10.

[3] Bidgoli, M. R., Vanani, I. R., and Goodarzi, M. 2024. Predicting the success of startups using a machine learning approach. Journal of Innovation and Entrepreneurship 13, 1 (2024). DOI: https://doi.org/10.1186/s13731-024-00436-x.

[4] Boris Sharchilev, Michael Roizner, Andrey Rumyantsev, Denis Ozornin, Pavel Serdyukov, and Maarten de Rijke. 2018. Web-based Startup Success Prediction. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18). Association for Computing Machinery, New York, NY, USA, 2283–2291. https://doi.org/10.1145/3269206.3272011.

[5] Gautam, L. and Wattanapongsakorn, N. 2024. Machine Learning Models to Investigate Startup Success in Venture Capital Using Crunchbase Dataset. IEEE. DOI: https://doi.org/10.1109/jcsse61278.2024.10613650.

[6] Kamil Żbikowski and Piotr Antosiuk. 2021. A machine learning, bias-free approach for predicting business success using Crunchbase data. Information Processing & Management, 58, 4, Article 102555. https://doi.org/10.1016/j.ipm.2021.102555.