MACHINE LEARNING PROJECT: ASSESSING SUCCESS OF STARTUP COMPANIES

Authors:
Eugen Mojsović (SNR: 2067563)
Max Malinov (SNR: 2078507)
Nikodem Baehr (SNR: 2076515)
Rana Yıldırım (SNR: 2063666)

300362-B-6: Real-life Machine Learning

Contents

1 Introduction to the problem

A startup company is by definition a company that is just beginning to operate. It is a product of one or several entrepreneurs and at the beginning is faced with high cost and limited revenues. Startups are facing very high uncertainty and plenty of competitors within a market from the get-go. As a result, company's owners are looking for a number of investors, most likely through venture capital and private equity, who could help with the financial matters. A venture capital fund is an investment fund that focuses on high-risk investments in innovative and/or fast-growing companies (Netherlands Chamber of Commerce Financing Desk, KVK; n.d.). Private equity is investment of equity capital in private companies (Snow, n.d.).

Startups benefit the economy through creating employment, leading to innovation and putting money into circulation. According to the Small Business Administration, small businesses, including startups, account for 44% of U.S. economic activity (Hale, 2023).

Some startups become the so called unicorns; that is, when a startup reaches a value of over a billion US dollars. For example, Instagram was launched in autumn of 2010, but the startup itself was created in 2009. In early 2010, the founder of Instagram, Kevin Systrom, raised over $500,000 from 2 venture capital funds. In 2011, the startup received Series A funding of 7 million USD. In spring 2012, Facebook bought Instagram for a billion dollars making Instagram a unicorn (Blystone, 2022).

Series A funding was mentioned in the paragraph above - it referes to a certain type of a founding round. Each funding round is designed to give a company sufficient capital (from the investors) in order to achieve its milestone(s). Pre-seed funding and seed funding precede the 'alphabet' series (A, B, C, D) funding (Cremades, 2018). Series D is usually the last round of funding. The funding increases in further rounds.

2 Descritption of Machine Learning Techniques

2.1 Random Forest

One method we implemented in our research is the Random Forest ensemble technique. It is tangent to our topic as overall it is a good classification method, whilst it is also robust to overfitting the data. The number of trees we iterate over is 100 and, adhering to the course contents, the split criterion we consider is the entropy value. As for the hyperparameter, we tune the minimal sample split size. We tune the model so that we prevent overfitting.

2.2 Logistic Regression

Another element of our study is the Logistic Regression. Due to it being rather simple, we expect that it could underperform compared to the rest of the models. Specifying the parameters for this method, we consider solvers and penalties when it comes to the hyperparameter tuning. 'Solver' refers to the optimization algorithm that the logistic regression involves to fit the model, whereas the 'penalty' pertains to the notion of regularization as means to penalize outliers (and avoid overfitting). We established that due to the format of our data, only the *liblinear* and *lbfgs* solvers are feasible, and for the penalties we quantified the effects of *l1*, alluding to Lasso regression, and *l2*, for Ridge regression, penalties.

2.3 Support Vector Machines

Given that our research revolves around a classification problem (startup being successful or not), we can consider Support Vector Machines as another prediction alternative. In particular, we pay special attention to nonlinear SVMs. Consequently, we explore several different kernel types to map our data into higher-dimensional space, where the data would eventually be linearly separable, aiding our decision-making algorithms. To be more specific, the four kernal types embedded in our project are the *Linear, Polyonomial, Radial* and *Sigmoid*. Importantly, so that our data is compatible with the compuations to follow, we had to scale it on the range (-1,1) using the *MinMaxScaler* function on our training and test sets.

1. **Linear kernel**
   This is the simplest kernel we consider as it establishes a linear decision boundary within the input space. Regarding hyperparameter tuning, we collate the method values for various regularization parameter (abbreviated as c), considering both very low ones, in the magnitudes of 0.1, up to big values such as 100. Since *sklearn* only takes an array of values for the aforementioned hyperparameter and we cannot specify a whole range, we provide a number of parameters between 0.1 and 100 and compare the F1 scores.

2. **Polynomial kernel**
   This kernel type operates on the basis of dot-product computations. For its execution, we supply two parameters: the regularization parameter c we mentioned in the linear kernel discussion, as well as *degree*. In the tuning process, we consider an identical range of values for c, whereas for the *degree* we compare the model performance for the integers between 2 and 5 inclusive. The choice of range here again dawns from the relatively small dataset.

3. **Radial kernel**
   For this kernel type we had to fine-tune the regularization parameter c only. To maintain consistency, we considered the same range of possible values as for the previous two kernels.

4. **Sigmoid kernel**
   Similarly to the radial kernel, the sigmoid one also requires a single hyperparameter to be tuned. Namely, we observe the impact of the same regularization parameter c. We define the same feasible value range as before.

2.4 Perceptron

As penultimate strategy to tackle the problem, we decided to implement an artificial neural network to our research. We utilised the default linear activation function, since the sole purpose of our project is to be able to distinguish between successful startups (indicated by y=1) and unsuccessful ones (respectively *y=0*), so it is a simplified binary decision rule. The penalty type, choosing between *l1, l2, elasticnet* or *None*, since only these ones would fit our dataset, and learning rate, η. As for the learning rate, we consider the values 0.1, 0.01 and 0.001 to assess whether larger or smaller values fetch improved performance in our model.

2.5 Stacking

The final method we've decided to focus our attention to is the Stacking ensemble technique. In this phase, we establish a meta-model which features several base models and combines their predictions rather than computing averages/carrying out voting. The models we feed to the Stacking method are the Random Forest, Radial SVM, Sigmoid SVM and Perceptron. Furthermore, our final estimator that we use to combine the base estimators is the default Logistic Regression.
Before we initiate the stacking algorithm, in order to alleviate part of the computational time of the model, given that we implement a number of basic methods simultaneously and that could result in a significant running time increase, we create pipelines for the models we suspect would benefit from premature scaling. We carry out such a procedure for the Perceptron output as well as for the Radial and Sigmoid Support Vector Machines.
Having computed the aforementioned pipelines, we determine the optimal hyperparameters for the final logistic regression that will capture the meta-model performance. Finally, we carry out the K-fold validation procedure on the stacking model to assess its performance.

3 Experimental Evaluation

3.1 Goal of the project

The goal of project is to truthfully predict whether a startup is successful or not. The dataset deems a company successful if the owners/founders of the company will significantly benefit, in financial terms, through the merger and acquisition process (M&A) or through Initial Public Offering (IPO). An IPO is when a company lists its stocks on a stock exchange. A startup will be considered a failed one if they have shut down.

Our project will highlight certain areas which are significant for a company to be acclaimed a successful one. That information will benefit the founders of the startups, as it will show them the features, which need considered with extra attention. The information will also benefit the established companies and investors (venture capital funds, angel investors or private equity) knowing where, when and how to act - whether to drive the growth of a startup or to refuse taking part in investing into it. An angel investor is a high net-worth individual who provides financial backing and initial seed money for startup businesses, usually in exchange for ownership equity in the company (Ganti, 2023)

We are putting aside entrepreneurial and creative aspects of the company and are only focused on the data.

CRISP-DM model stands for cross-industry standard process for data mining. The phases of the model are: business understanding, data understanding, data preparation, modelling, evaluation and deployment.

Figure 1: CRISP-DM Model

3.2 Data understanding and data preparation

In this section, we will cover two phases of the CRISP-DM model (Figure 1) - data understanding and data preparation. For analysing whether a startup is successful or not, we were using data available at Kaggle (Startup Success Prediction).

In the provided Jupyter notebook, an URL link was created in order to fetch the data. Data covers a selection of startups from the United States. Our data consists of 923 startups out of which 597 were considered successful (64.9%). The available data offers 49 features. Many features in the data were found useless in order to solve the problem and were therefore dropped.

Dropped features are: (Table 1)

Feature	**Reason for dropping**
Unnamed: 0	identification variable
object_id	identification variable
id	identification variable
state_code	unnecessary location information for the problem
Unnamed: 6	unnecessary location information missing in a majority of instances
zip_code	location information unnecessary
state_code.1	location information unnecessary
city	location information unnecessary
first_funding_at	information about 'when' is unnecessary; important if received funding (also, date data)
last_funding_at	information about 'when' is unnecessary; important if received funding (also, date data)
age_first_funding_year	information about 'when' is unnecessary; important if received the funding
age_last_funding_year	information about 'when' is unnecessary; important if received the funding
age_first_milestone_year	information about 'when' is unnecessary; important if reached the milestones
age_last_milestone_year	information about 'when' is unnecessary; important if reached the milestones
category_code	word descriptive information which is presented in categorical Boolean variables
name	does not contain predictive information
status & id	outcome variables (comparing the predictions to the variables)
founded_at & closed_at	**replaced with a new variable age**

As a result, we ended up with 29 features. Some features are self-explanatory like for example funding_total_usd, or founding_rounds. For others, more explanation is provided:

- milestones - number of reached milestones
- relationships - number of relationships a startup has whether it's accounts, vendors, mentors etc
- is_CA,is_NY,is_MA,is_TX - state of origin (CA is California, NY is New York, MA is Massachusetts and TX is Texas) (Boolean)
- is_software, is_web etc. - type of industry the company is in (Boolean)
- has_VC - company with venture capital (Boolean)
- has_angel - company with angel investor(s) (Boolean)
- has_roundA, has_roundB, has_roundC, has_roundD - received different rounds of funding (Boolean)
- avg_participants - average number of investors
- age - age of the company in days (if failed, age when shut down; if successful, age as of Nov 11^{th}, 2023). Defined as difference in days (closed_at - founded_at)

latitude	longitude	relationships	founding_rounds
funding_total_usd	milestones	is_CA	is_NY
is_MA	is_TX	is_otherstate	is_software
is_web	is_mobile	is_enterprise	is_advertising
is_gamesvideo	is_ecommerce	is_biotech	is_consulting
is_othercategory	has_VC	has_angel	has_roundA
has_roundB	has_roundC	has_roundD	avg_participants
age	is_top500	still open	

Table 2: Features

Also, a new feature was created - a variable called 'still open'. It is a Boolean type variable telling whether a company is still open or whether it shut down. After creating variables 'age' and 'still open', the number of features totaled to 31.

After creating a variable 'age', entries were checked to see if any had negative values. The data points which returned negative variable age were dropped due to a false information.
Later, the data set was split into the training and test sets (90% and 10%) such that different methods of prediction could be carried out.

3.3 Methods

To provide a comprehensive way to classify startups, we've decided to assess the applicability of various machine learning methods in the context of our topic. This section serves as the 'Modeling' stage of the CRISP-DM cycle. We are comparing the performance of the models across the precision, accuracy, recall and F1 scores, but in the sequel we predominantly focus on the F1 scores associated with the models in question. The grounds for this choice reside on the nature of this measure, which encapsulates both the precision and recall figures within its formula. Thus, we deem this measure applicable to our case as it showcases the trade-off between true/false positives and negatives across different iterations.

3.3.1 Some Methodological Clarifications

As preliminary work before initializing each model to monitor its performance, we need to clarify a few technical conventions we make during the study. Firstly, we split our dataset into two. The resulting sets are *xt*, our training set, and *xv*, our test set (xv is used for notational convenience). Furthermore, the way we split our data is by allocating 90% to the training set and, respectively, 10% to the test set. Secondly, to generate consistent results that can be reproduced, we initially set a seed that captures the data separation and every time the code is executed, the training and test sets feature the same entries. In the last section of our study, we drop this seed. Then, we run a simulation across 100 iterations and on each run we set a new seed, so that we obtain a benchmark for comparing the models. In this way, we shield our results against potential influence of outliers or stray data points from the initial split. This adds to the robustness of our conclusions.
Thirdly, as the algorithm validates each model using K-fold cross-validation. For each method, we carry out this method for 10 folds by allocating 10% of the training set for validation. By convention in the field, this is widely regarded as being a sensible choice as it negates outliers being grouped together when data is alloted to each fold. We investigate the performance on both the training and the validation set as we seek to establish whether the models over-/underfit the data.

3.3.2 Hyperparameter Tuning

To offset any irregularities in our research and improve each model's performance, we introduce optimal hyperparameters to all implemented methods. Due to the variable nature of our methods, we assess each case separately. That is, for each different classification technique, provided what the feasible hyperparameters are, we pick the optimal one in each case. The procedure for hyperparameter tuning is identical for each model:

1. Given our test set, we run the model for all relevant hyperparameter values/types.
2. On each iteration mentioned above, we extract the corresponding F1 scores of the prediction on the test set and store them in a list.
3. Observing which the highest F1 score is, we take the corresponding parameter value as the optimal one and use it in the code execution thereon.

3.3.3 Simulation

Because we originally set a single seed for the entire model evaluation procedure, we would try to offset the possible discrepancies that could arise given the initial training-test set split. To accomplish such data-split independence, we run 100 simulations across different seeds (different allocation of training and tests sets), following the exact same rules and sequence as before. We conduct this experiment to objectively verify the goodness of our models and to monitor the performance of each method across a high number of iterations. Such an approach should allow us to draw more coherent and comprehensive conclusions regarding which techniques work best when it comes to startup success prediction.

3.4 Hyperparameter Optimization Results

An essential element of our computations is which hyperparameters fetched the optimal values for each model. This is the first step of our 'Evaluation' CRSIP DM stage. Below, we provide a brief discussion indicating which parameter values showcased the best performance.

3.4.1 Random Forest

For the Random Forest approach, as explained before we want to achieve optimal split size. We observe that in the range between 2 and 10, the optimal model performance is fetched for the hyperparameter 2. The reason why we focus on this range is the relatively small size of the dataset.

3.4.2 Logistic Regression

In this model, we have to decipher the best solver and penalization pair. After the hyperparameter tuning cycle, we deduced that the *liblinear* solver performs best, alongside an *l2* penalty.

3.4.3 Support Vector Machines

1. **Linear Kernel**
   Here, we compare different regularization parameter values between 0.1 and 100 based on the respective F1 scores. The best score is ascribed to *c=100*, so that's the parameter we supply to the model.

2. **Polynomial Kernel**
   With respect to this kernel type, we seek optimal regularization parameter and degree. The values we find to produce the highest F1 scores are *c=3* and *degree=3*.

3. **Radial Kernel**
   Similarly to the linear kernel, the optimal magnitude here appeared to be *c=100*.

4. **Sigmoid Kernel**
   Extending the discussion for the other kernel types above, we obtain that the optimal model performance is achieved at regularization *c=0.1*.

3.4.4 Perceptron

In case of the Perceptron model, we collate different solver-and-penalty combinations to see which one gives the highest F1 score. After computing the scores for each pair, we obtain that if we take η=0.1 together with no penalty, we yield the best results.

3.4.5 Stacking

With respect to the optimal hyperparameters here, we arrive at an identical result as compared to the logistic regresison above. The solver we will use is *liblinear* and the penalty we take will be *l1* (Lasso regression).

3.5 Results and Discussion

In the following section, we present some of our findings alongside our interpretation of what they imply with respect to our models to complete the 'Evaluation' stage in the CRISP DM cycle. In order to better illustrate our results, lets revisit the F1 score formula:

$$\text{F1 Score} = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}} \tag{1}$$

Furthermore, we have that Recall is determined by taking the ratio of the true positives generated by the model and the sum of true positives and false negatives. On the other hand, Precision estimates the ratio of true positives and all positives (both true and false).
Therefore, it is sensible to conclude that if we observe values of the F1 score close to 1, it implies that both the recall and precision measures must both be close to 1, too. Now, let's consider each model separately to establish its applicability to the topic at hand.

3.5.1 Random Forest

Let's start with the discussion with respect to the Random Forest method. In the figure below, we see that for all 10 folds the training set exhibits very high F1 scores. This implies that the model generates a low number of false positives as well as false negatives. Furthermore, the estimation results are strong across all but the 7th fold, so we could conclude that Random Forest performs consistently well across the validation process.
Looking at the test set performance measures below, we see that supplied with unfamiliar data, the precision of the model is immaculate, there are no false positives generated. Precision is also very high, which suggests that this method produces few false negatives too. These two measures contribute to high F1 score as well. Finally, the accuracy measure confirms that in general the total count of false positives and false negatives is low when we run the model on the test set.

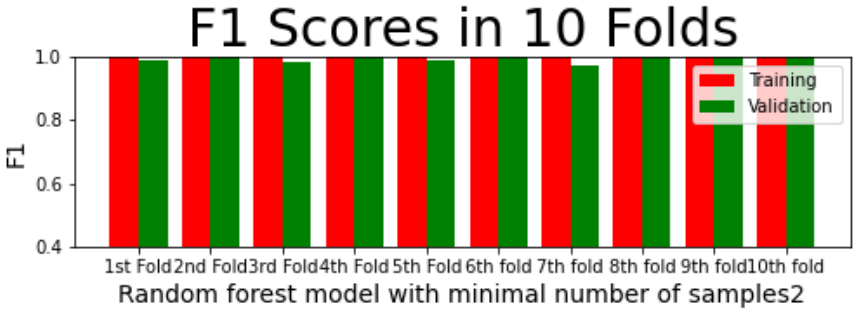

Figure 2: Random Forest Cross-Validation Results

	Accuracy	Precision	Recall	F1 Scores
Random Forest	0.978	1.0	0.967	0.983

Table 3: Random Forest Test Set Results

3.5.2 Logistic Regression

The second model to be discussed is the Logistic Regression. Under k-fold validation conditions, the method ostensibly performs a tad worse than its Random Forest counterpart when it comes to estimating the validation set. The F1 score on the training set appear to be approximately perfect, whilst the scores on the validation set for each fold also exhibit large magnitudes, despite being slightly off when compared to the Random Forest output.

When we feed in the test set, we observe a surprising result. The figures for all accuracy, precision, recall and F1 scores showcase the same magnitudes as the corresponding statistics in the Random Forest test phase, if we round to 3 decimal places. This implies that we can draw the same conclusions regarding the distribution of true/false positives/negatives as for Random Forest. Although counterintuitive at first, using tuned hyperparameters in the models can result in irregularities in the data being put the dampener on and results being drawn closer together.

Figure 3: Logistic Regression Cross-Validation Results

	Accuracy	Precision	Recall	F1 Scores
Logistic Regression	0.978	1.0	0.967	0.983

Table 4: Logistic Regression Test Set Results

3.5.3 Support Vector Machines

1. **Linear Kernel** Compared to the previous two models, the SVM with Linear Kernel proves a little bit worse at predicting startup success. Considering the K-fold validation results below, one notices that unlike the other methods, this SVM produces lower F1 scores when initialized on the validation set. It still has great training set prediction power, but performance on new observations appears to be more imperfect.
   Both accuracy and precision revolve around the 95% mark, whereas recall is over 98%. This suggests that this model generates more false positives than false negatives. Nevertheless, the F1 score remains high, amounting to almost 97%

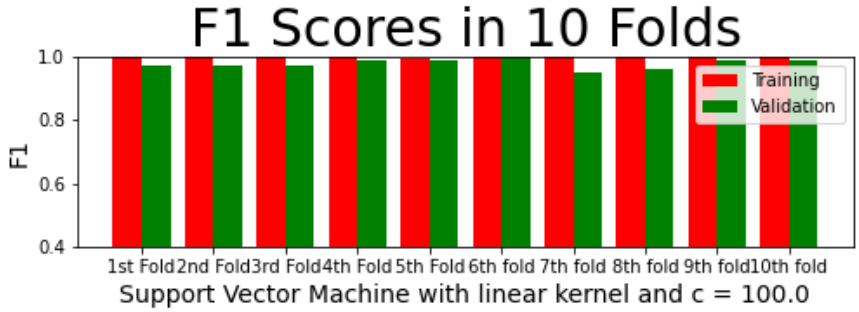

Figure 4: Linear SVM Cross-Validation Results

	Accuracy	Precision	Recall	F1 Scores
Linear SVM	0.957	0.952	0.983	0.968

Table 5: Linear SVM Test Set Results

2. **Polynomial Kernel** The Polynomial SVM proves to possess strong predicting power in our research. The results on the validation set are mostly accurate apart from folds 3 and 7. More importantly, one notices that when implemented with the test set, the accuracy is very high (almost 99%) and precision is 100%. Recall is also big in magnitude, so this method also produces very few false positives and false negatives overall. This contributes to the high F1 Score value we observed.

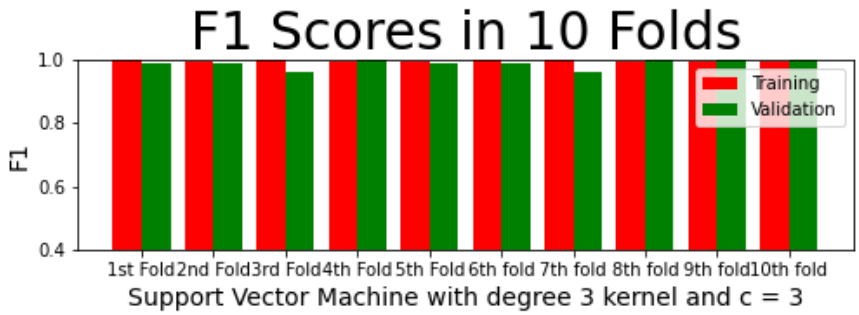

Figure 5: Polynomial SVM Cross-Validation Results

	Accuracy	Precision	Recall	F1 Scores
Linear SVM	0.989	1.0	0.983	0.992

Table 6: Polynomial SVM Test Set Results

3. **Radial Kernel** The Radial Kernel SVM performed better than the Linear counterpart, but a touch worse than the Polynomial SVM. Looking at the K-fold validaiton graph below, we postulate that the results are very much alike. Furthermore, the only two folds where the model underperforms compared to the other iterations are the same as with the Polynomial SVM (folds 3 and 7). Therefore, we surmise that the validation set in these folds possibly contained outliers that the model couldn't predict based on the training set it used for learning.
   As for the performance measures, recall is the same as in the polynomial kernel case, and the rest of the measures are up to a tenth down on the polynomial output. This again signifies good prediction ability, but a few additional false positives and false negatives are conceded when applying this kernel.

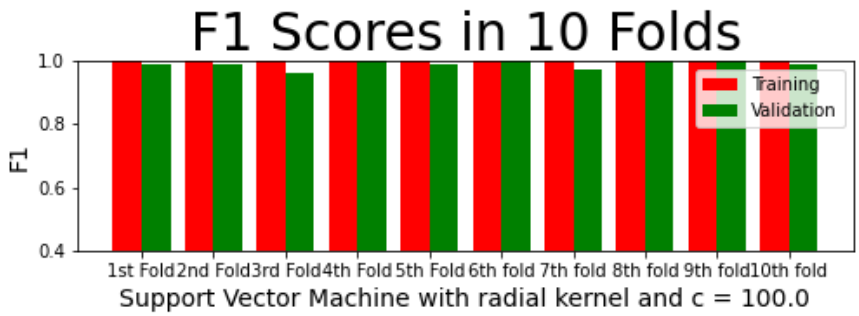

Figure 6: Radial SVM Test Set Cross-Validation Results

	Accuracy	Precision	Recall	F1 Scores
Radial SVM	0.978	0.984	0.984	0.984

Table 7: Radial SVM Test Set Results

4. **Sigmoid Kernel** The last SVM model, namely the Sigmoid one, was no exception to the generally strong performance of the Support Vector Machine algorithms featured in our study. It performed arguably the best out of all four kernels in the validation stage (consult the figure below). Furthermore, the method didn't produce any false positives (captured by 100% precision. Strong accuracy and recall scores, as well as over 98% F1, suggest that we can deem this model a strong predictor.

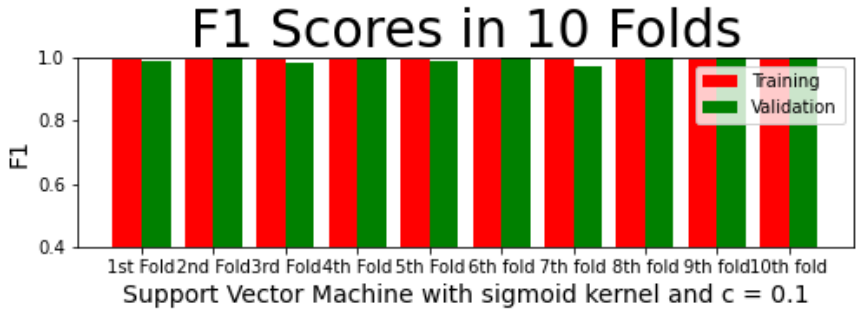

Figure 7: Sigmoid SVM Cross-Validation Results

	Accuracy	Precision	Recall	F1 Scores
Sigmoid SVM	0.978	1.0	0.967	0.983

Table 8: Sigmoid SVM Test Set Results

3.5.4 Perceptron

Interestingly enough, the Perceptron method fetched fairly identical results to the Sigmoid SVM. K-fold validation output appears to be extremely similar, whereas the performance measures obtained on the test set have the exact same magnitude when rounding to 3 decimal places. This could result from the efficient choice of hyperparameters as outliers are punished accordingly and the model is less susceptible to overfitting.

Figure 8: Perceptron Cross-Validation Results

	Accuracy	Precision	Recall	F1 Scores
Stacking	0.978	1.0	0.967	0.983

Table 9: Perceptron Test Set Results

3.5.5 Stacking

We expect the stating algorithm to possess stern predicting power. After all, it captures a number of models which exhibited robust and accurate performance per se. Surprisingly, the outcome of this method is not very different from out observation on the Perceptron model. K-fold validation appears to be comparable, and the assessment measures calculated on the training set have the same numerical values. Therefore, stacking doesn't present an improvement to the perceptron neural network, but nevertheless the model's versatility should not be underestimated.

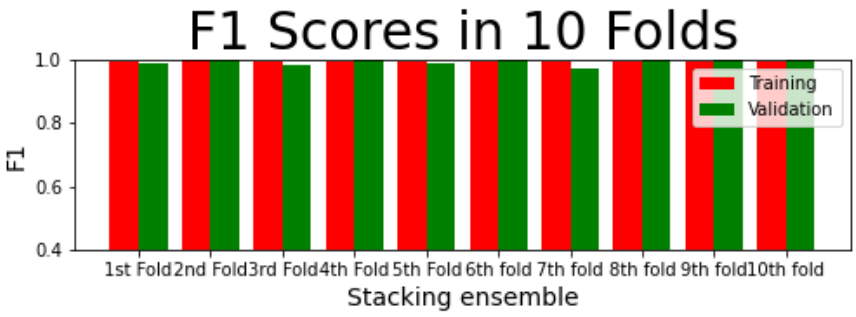

Figure 9: Stacking Cross-Validation Results

	Accuracy	Precision	Recall	F1 Scores
Stacking	0.978	1.0	0.967	0.983

Table 10: Stacking Test Set Results

3.5.6 Simulation

As mentioned in the preface, when we originally split the test and training sets in the beginning of our research, we set a random seed so as to store the split and enable others to reproduce identical results to ours. However, it isn't sensible to rely on the output of just one seed, for it opens the door for many irregularities if the results happen to be contingent on this specific seed. To tackle this issue, we run 100 simulations on various seeds to extract comprehensive insight into our model performances that could be used to draw more general conclusions about how to curb the startup success prediction topic.
In the procedure we adapt to our research, during each of the 100 cycles we carry out we make note of which model generates the highest F1 score (sometimes more than one model achieve the same result and we give a point to both in such cases). As previously discussed, F1 scores are a great indicator for the sake of our study in the sense that they are proficient in capturing the false positives/false negatives production vulnerability of the models.
As displayed below in the table, the two models that are most consistent in out case are the Random Forrest and Sigmoid SVM. Across all simulations, the former attained the highest F1 in 83 cases, and the latter-in 84. Those numericals imply that in our framework the two models mentioned above would serve as the best predictors for startup success. Nevertheless, Logistic regression and Stacking also exhibited promising performance, albeit they weren't on a par with the top two. Perceptron and the remaining 3 SVMs,though, proved to be less efficient than their competitors, so we shouldn't opt for them in the long run.

Model	Number of iterations it is best
Random Forest	83
Logistic Reg	62
Linear SVM	20
Polynomial SVM	28
Radial SVM	27
Sigmoid SVM	84
Perceptron	35
Stacking	74

Table 11: Simulation Results

4 Conclusion

In summary, the results point at a few phenomena that went against our inherent expectations.
During the simulation stage, the Perceptron model revealed significantly lower dependability than we expected. Despite it being rather simple by construction, our original suspicion was that in the long run the Perceptron would be more efficient as with the first seed it was amongst the top-performing models. However, across the domain of simulations the performance plummeted significantly.
Another interesting result for us was that stacking wasn't the optimal approach. Considering ensemble techniques take various models and combine their performance, we predicted that stacking would display a more accurate figure than the individual models. However, that statement was disconfirmed by our observations.
Eventually, the two models that provide the best predictions are the Random Forrest ensemble technique and the Sigmoid Support Vector Machine. Even though they could be computationally expensive for large datasets, they produce the least false positives and false negatives across all methods considered. Therefore, we recommend that, if the specific setup permits, the aforementioned two methods should be used in the startup success prediction algorithms.
For the takeaway of our topic of discussion, if one wants to predict the path of a startup (whether it will survive or will go out of business) based on the characteristics of the company (such as the investigated funding, investors, location, milestones etc.), they would obtain the best results if they deploy a classification algorithm residing on a Random Forest or Sigmoid SVM model. Given the modeling process we went through, these two models prove to be most feasible for the task.

5 Possible Improvements

Although we conducted an extensive research on the matter and did a great deal of modeling, we are aware that the final product of our research features areas that could be improved. Namely, we would like to point out three key components of our approach that could be rectified in the future:

1. On one hand, the number of data points in our experiment was arguably small. Although macroeconomic studies often feature medium-sized datasets, to ensure the robustness of our model we could screen its predicting power over larger information sets. This would enable us to see if the strong model performance we observed persists or setbacks such as overfitting or significantly higher computational costs arise.

2. Second, we would like to investigate the outcomes predicted on a dataset that has less variables than ours. A considerable amount of data was available to the models in the simulations we carried out, and we ought to experiment with the performance of the algorithm on less extensive data to see if the model continues to excel.

3. Finally, we are interested to compare the output of our best methods to a more sophisticated machine learning algorithm, such as a multiple-hidden-layer artificial neural network. Given the simplicity of some of the models embedded in our research, it is important to verify their applicability by comparing the predicting power with a more involving and less assumptions-dependent alternative.

6 Bibliography

- Manish KC, Startup Success Prediction
  https://www.kaggle.com/datasets/manishkc06/startup-success-prediction

- Cremades A (Dec 26, 2018), How Funding Rounds Work For Startups, *Forbes*
  https://www.forbes.com/sites/alejandrocremades/2018/12/26/how-funding-rounds-work-for-startups/?sh=57858cc73866

- Hale K (Sep 11, 2023), How Cultivating A Strong Startup And Accelerator Ecosystem Spurs Economic Growth, *Forbes*
  https://www.forbes.com/sites/korihale/2023/09/11/how-cultivating-a-strong-startup-and-accelerator-ecosystem-spurs-economic-growth/

- Ganti A (29 Sep, 2023), Angel Investor Definition and How It Works
  https://www.investopedia.com/terms/a/angelinvestor.asp

- Blystone D (22 Oct, 2022), Instagram: What It Is, Its History, and How the Popular App Works
  https://www.investopedia.com/articles/investing/102615/story-instagram-rise-1-photo0sharing-app.asp

- David Snow (n.d.), Private equity: a Brief Overview An introduction to the fundamentals of an expanding, global industry, *PeiMedia*, 2

- Netherlands Chamber of Commerce Financing Desk, KVK; Venture Capital Funds
  https://business.gov.nl/financing-your-business/funding-and-loans/funding-by-private-investors-or-banks/funding-from-venture-capital-companies/