Predicting Startup Success: Creating a Classification Model Using Crunchbase Investment Data

Greta Kichelmacher, Dimitrios Georgilis, Ahnaf Mohsin and Mohamed Fouad

1. BUSINESS UNDERSTANDING

a) How would investors benefit from a model that predicts startup success?

As mentioned in Arroyo et al. (2019), larger Venture Capital firms tend to invest in “bigger tickets in faster-scaling companies and to double on winners”. This presents an opportunity for venture capital firms to invest in early-stage companies which would “reduce” investment risk and manage uncertainty in long term outcomes (Arroyo et al., 2019). Leveraging machine learning models can help investors make “better and more informed” investment decisions (Arroyo et al., 2019).

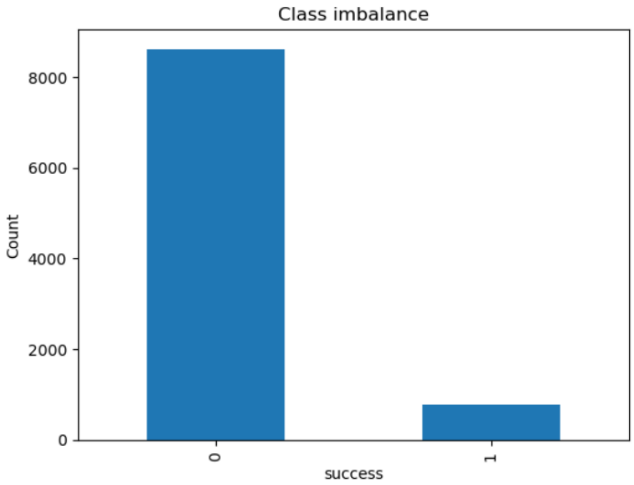
b) Why does it make sense to consider the lack of outcomes recorded in the Crunchbase database to be a sign of failure?

The CrunchBase database tends to favor successful companies, often showcasing those with significant achievements or milestones. Additionally, the timing and completeness of data updates can vary, meaning that companies with more pronounced successes may have their details updated or added more promptly. Given these characteristics, if a company lacks recorded outcomes or specific updates in CrunchBase, it suggests a potential relative lack of success or notable developments in its journey, making it a potential marker for determining a company's standing in the industry.

2. DATA UNDERSTANDING AND PREPARATION

a) Create a new target variable (success) with two values: 1 for success and 0 for failure. Use the definition of startup success provided above to determine the value of the target variable.

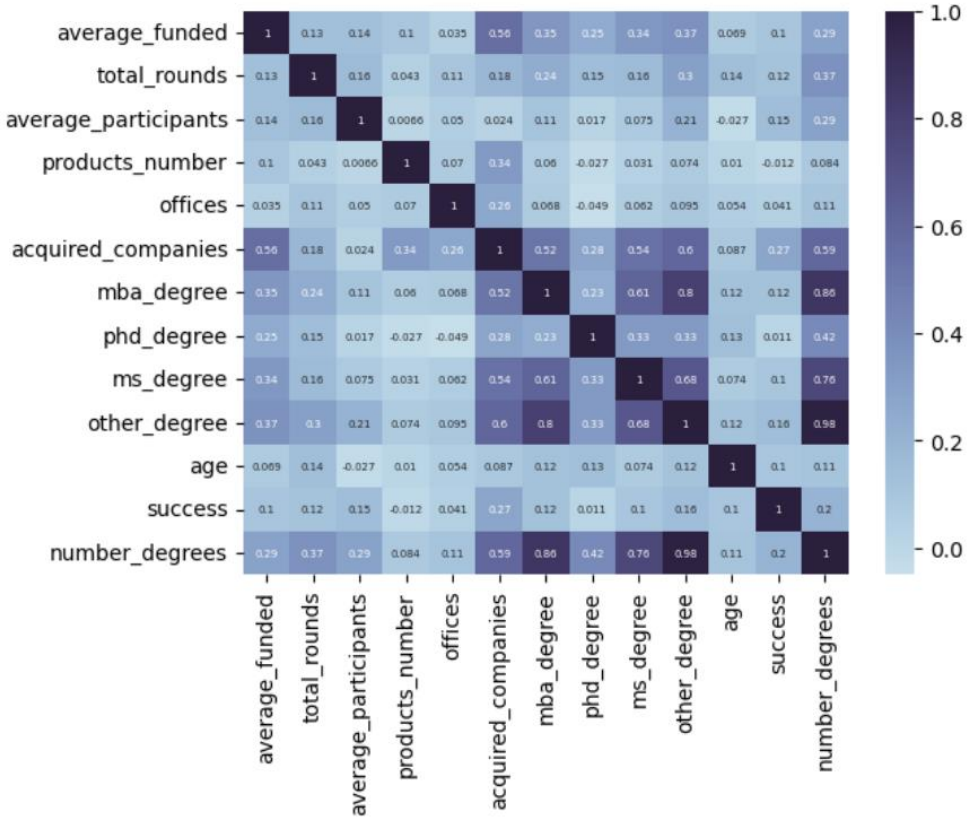
When creating the target variable ‘success’ it is possible to notice that the dataset is very unbalanced toward the ‘unsuccess’ class. More precisely, the total number of successful companies is 772 while the number of unsuccessful ones is 8617.



b) Identify the numerical features in the dataset (including the new feature created in e) and show their correlations with one another and the target in a heatmap. Are any features highly correlated with one another? What does this tell you?

Heatmap interpretation:

- The target variable "success" seems to have the highest correlation with acquired_companies and other_degree, though the correlation is moderate and not very strong. This suggests that having acquired companies and having a degree might be associated with the "success" of a Startup.
- Features like mba_degree, other_degree and ms_degree are highly correlated with number_degrees, since we have manually created this feature. Including all these features in a predictive model might lead to multicollinearity, which can impact the interpretability of the model. Therefore, we would consider only the combined variable for the final model.
- Most of the other features show low correlation with the target variable "success", suggesting that individually, they might not have a strong linear relationship with the target.



c) Identify the categorical features in the dataset. What will you need to do with the categorical features before you can use them in the model? No code, just a description of what you need to do.

when dealing with categorical features, before we can use them in the model, we have to preprocess them since machine learning models require numerical inputs. Based on the nature of the categorical features in this dataset:

- company_id: this can be dropped from the dataset as it merely serves as a unique identifier for each entry, without any predictive value for the model.
- country_code and state_code: since they have few unique categories, we can proceed with One-Hot Encoder. It is crucial to note that One-Hot Encoding creates a new column for each unique category, which is why understanding the number of unique values is essential to avoid excessive dimensionality.
- category_code: since this feature has a lot of unique categories, we can encode it with LeaveOneOut encoder which is particularly useful when dealing with high-cardinality categorical features, where the number of unique categories is large. This encoder works by assigning a value to each category based on the target variable's mean or probability, calculated by leaving out the current observation being encoded (the target variable is 'success').

d) How do you avoid look-ahead bias? Can you use all the features in the dataset to predict startup success?

Look-ahead bias can be avoided by avoiding features from the dataset that “would not be available at the time of prediction” (Weiss, 2023).
This type of bias occurs when a model or analysis uses information that would not have been available or known during the period being analyzed. By considering how we have created the target variable 'success':

- the startup issued an IPO (ipo), or it got acquired (is_acquired)
- a startup would not be considered successful, if it closed (is_closed) or none of the previous outcomes (ipo, is_acquired, or is_closed) occur.

To avoid look-ahead bias, when predicting the 'success' target variable, we should exclude these features from the set of predictor variables, as these were directly used to derive the target.

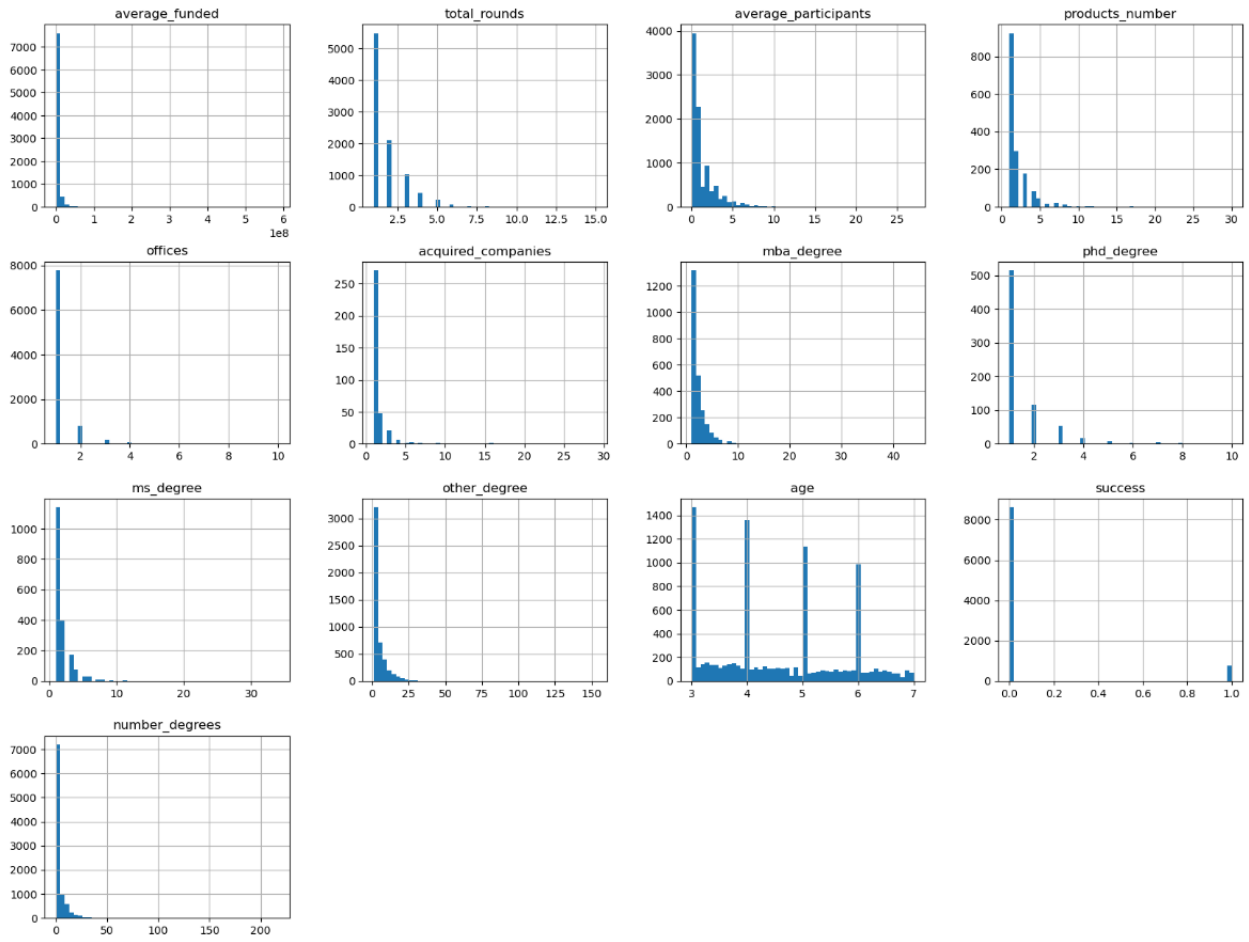
e) Can we choose reasonable default values for the missing values or should any of these features be removed?

The features with missing values are:

- Category_code
- Average_funded
- Products_number
- Offices
- Acquired_companies
- Mba_degree
- Phd_degree
- Ms_degree
- Other_degree

To better understand which imputation to do for the numerical features we looked at their distribution. More specifically:

- if the distribution of the feature has already many 0, we can consider imputing with 0;
- if the distribution of the feature suggests that typically that specific feature has some value it will be better to impute with the median value or others.



In conclusion:

- CATEGORICAL VARIABLE:
  - category_code: Since this is the company category, a missing value might be imputed as 'other' since also other features (such as country_code and state_code) already has this value.

- NUMERICAL VARIABLES:
  - average_funded: considering this is the average funding per round, a missing value might indicate that the company did not get funding, or the data was not recorded. Since the distribution already has 0, meaning that is a possible value, we will impute with 0 values.
  - products_number: has no 0 in the distribution, therefore we will impute with the median value. We decided to impute with the median (instead of the mean) since is more robust to outliers and is more appropriate to skewed data (i.e., the case of the distribution of products_number that is right skewed).
  - offices: given the context, it is logical to fill in missing values with 1 since each company is expected to have at least one office.
  - acquired_companies: has no 0 in the distribution. However, considering the context of the data, if a company's acquisition data is missing, one might assume they have not acquired any companies. Therefore, we will impute with 0.

- mba_degree, phd_degree, ms_degree, other_degree: As stated before all these features are highly correlated between each other and with the new variable created 'number_degrees'. Therefore, we will remove those variable from the model and use only 'number_degrees' which has zero missing values.

3. MODELLING

a) How do you choose training and test dataset from the full dataset?

From the full dataset (filt_df) we have first created 2 different datasets:

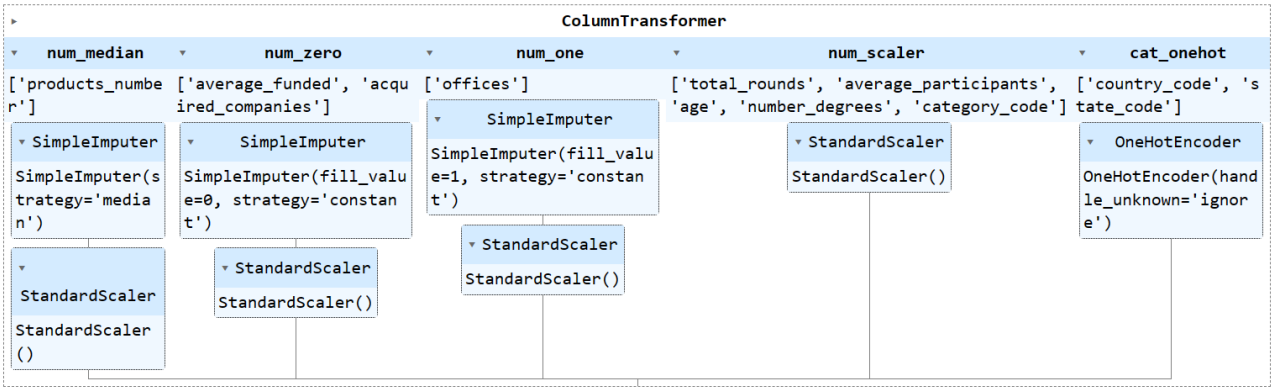
- X: that will contain all the predictors, except the ones that we decided to remove for this specified reasons:
  - company_id: since is just a feature that uniquely identify each row but does not have any predicting power.
  - 'ipo', 'is_acquired', 'is_closed': to avoid look-ahead bias.
  - "mba_degree", "phd_degree", "ms_degree", 'other_degree': to avoid multicollinearity and bias we used only the total number of degree ("number_degree"). This is also in line with what has been done in the paper (Arroyo et.al. 2019)
- y: that will contain only the target variable ('success').

To construct the training and test datasets, given the dataset's size, we have opted to allocate 80% of the data to the training set and reserve the remaining 20% for the test set. Furthermore, considering that the target variable is very unbalanced on the 0 values (“unsuccess”) we will also use the stratification technique. Stratification ensures that both training and test sets have a similar distribution of the target class, preserving the original imbalance proportion and improving the model's generalization on the minority class.

b) Create a pipeline for pre-processing numerical and categorical features. This should also take care of missing values.

Pipeline:

- impute the null values of numerical and categorical features with the specifications from above.
- Scaling with standard scaler all the numerical features.
- encoding categorical features: state_code and country_code with One-Hot Encoder since they have few unique categories, and category_code with LeaveOneOut Encoder (done before the pipeline).



c) Create two models: one using logistic regression, the other using random forests. As in the example discussed in class, allow the user to choose which model should be created and evaluated.

For both logistic regression and random forest a GridSearchCV has been implemented to find the best parameters to use in the final model.

The best results for the **logistic regression** on the test set are:

- Accuracy: 0.7523961661341853
  This is the ratio of correctly predicted instances to the total number of instances in the dataset. It is a pretty high accuracy, which sounds promising (an accuracy of 0.7524 means that the model correctly predicted the outcome for approximately 75.24% of the test samples), but accuracy alone can be misleading in imbalanced datasets (i.e., when one class is much more frequent than the other) as it is in our case.

- Precision: 0.1908548707753479
  This is the ratio of correctly predicted positive observations to the total predicted positives. A precision of 0.1909 means that when the model predicts an observation as positive, it's correct only about 19.09% of the time. This indicates a high number of false positives.

- Recall: 0.6233766233766234
  This is the ratio of correctly predicted positive observations to all the observations in the actual class. A recall of 0.6234 means that the model correctly identifies about 62.34% of the actual positive cases.

- F1-Score: 0.2922374429223744
  This is the weighted average of precision and recall. It takes both false positives and false negatives into account. A high F1-score suggests better robustness and reliability in terms of both precision and recall. F1-Score closer to 0 indicates poor predictive power. Therefore, as we can see in our case, the F1-score of 0.2922 is quite low, suggesting that the balance between precision and recall is not ideal.

In summary, the model has good accuracy and recall, indicating its ability to classify positive instances, but its precision is low, implying that it tends to produce false positives.

The best results for the **Random Forest** on the test set are:

- Accuracy: 0.9217252396166135
  The model achieved an accuracy of approximately 92.17%, which means it correctly predicted the class labels for roughly 92.17% of the instances in the test set.

- Precision: 0.5614035087719298
  The precision is approximately 56.14%, indicating that when the model predicted a positive outcome, it was correct about 56.14% of the time.

- Recall: 0.2077922077922078
  The recall is approximately 20.78%, which means the model correctly identified about 20.78% of all actual positive instances.

- F1-Score: 0.30331753554502366
  The F1-Score in this case is approximately 30.33%, indicating a trade-off between precision and recall.

Overall, these metrics suggest that the random forest model has a relatively high accuracy, but it may not perform as well in terms of precision and recall, especially for positive instances. The low recall suggests that the model might miss many positive cases.

4. EVALUATION AND INTERPRETATION

a) Which performance metrics are most suitable for evaluating your model to predict startup success? It helps to think about what performance metrics would matter most to an investor. Hint: It helps to read the article carefully.

The performance metrics most suitable for the model (in accordance with Arroyo et.al. 2019) are as follows: Note: not all data used in the article was present in our Crunchbase database. Only information available to us has been listed.

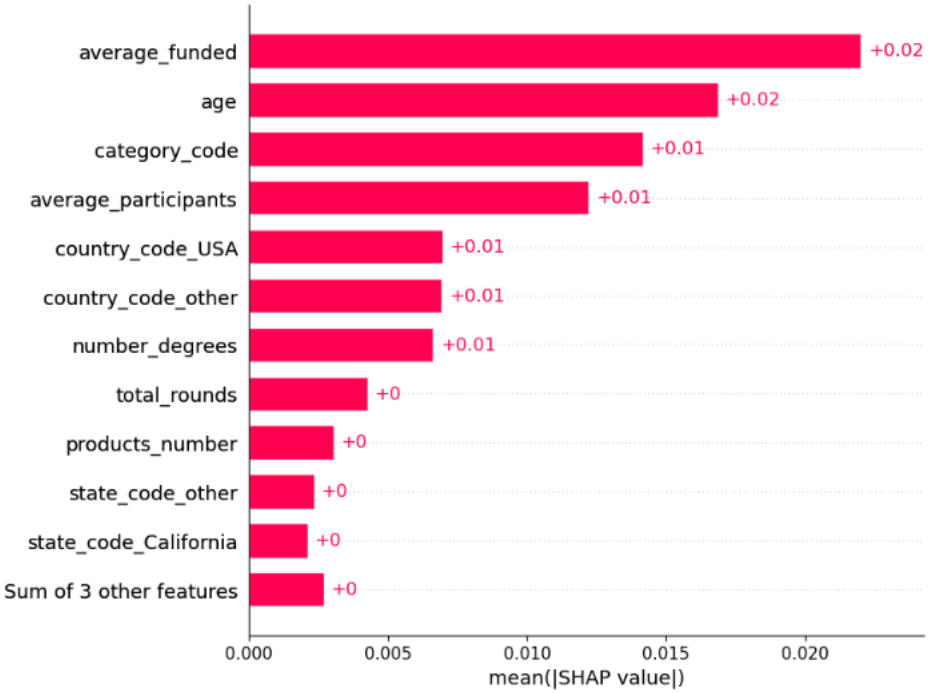
- Funding Information:
  - Total funding amount raised (raised_amount_usd)
  - Number of funding rounds (round_count)
- Company Information:
  - Age of the company (age_months)
- Founders Information:
  - Educational background of founders, such as total degrees obtained (founders_degree_count_total)

These were the primary indicators in startup success prediction model.

For assessing our model, the relevant metrics include precision, recall, and the F1 score for each category. While recall is not a pivotal metric, as venture capitalists (VCs) do not need to identify every promising company, the F1 score is incorporated merely for comparative purposes and will

not guide our final choices. The primary concern for a VC investor is to enhance their success rate when deciding on potential investments. Consequently, our emphasis should be on the precision of lucrative classes like FR, AC, and IP. Precision will form the foundation of our classifier comparisons, and recall will provide additional insights (Arroyo et.al. 2019).

Furthermore, we have used the SHAP library to show the feature importance.
SHAP is a powerful library that makes use of a mathematical method to explain the predictions of machine learning models. It is based on the concepts of game theory and can be used to explain the predictions of any machine learning model by calculating the contribution of each feature to the prediction. Here we used shap on our best model to let it be more interpretable and to understand better how this model makes its predictions. We wanted to use this library to get the insights on how our best model (Random Forest) works.



From this graph it is possible to notice the contribution of each variable to the predictions. Higher is the shap value, higher is the importance of that predictor. Here we have only the total importance without saying if it is directly or inversely correlated with the response variable.

Moreover, it is evident that the variables that contribute the most in determining the predictions are:

- average_funded
- age
- category_code
- average_participants
- country_code_USA
- country_code_others
- number_degrees

Therefore, it is possible to say that this is in line with what has been shown also in the paper (Arroyo et.al. 2019)

b) The dataset is imbalanced. There are many more failed startups than successful startups. Suggest two ways to deal with imbalanced data and implement one of them.

There are two ways to deal with imbalanced data and those are:

- Oversampling: this technique involves increasing the number of instances in the minority class by duplicating instances or generating synthetic instances. This technique can improve model performance on the minority class. However, this technique can lead to overfitting.
- Undersampling: this technique involves reducing the number of instances from the majority class to balance the class distribution. This technique involves reducing the number of instances from the majority class to balance the class distribution. However, this technique can lead to underfitting due to the diminished dataset size.

We decided to implement the undersampling technique. Furthermore, considering what has been said in 4a we also removed the features with low prediction value in the previous random forest model.

The best results for the **logistic regression** on the test set are:

- Accuracy: 0.7678381256656017
- Precision: 0.19480519480519481
- Recall: 0.5844155844155844
- F1-Score: 0.2922077922077922

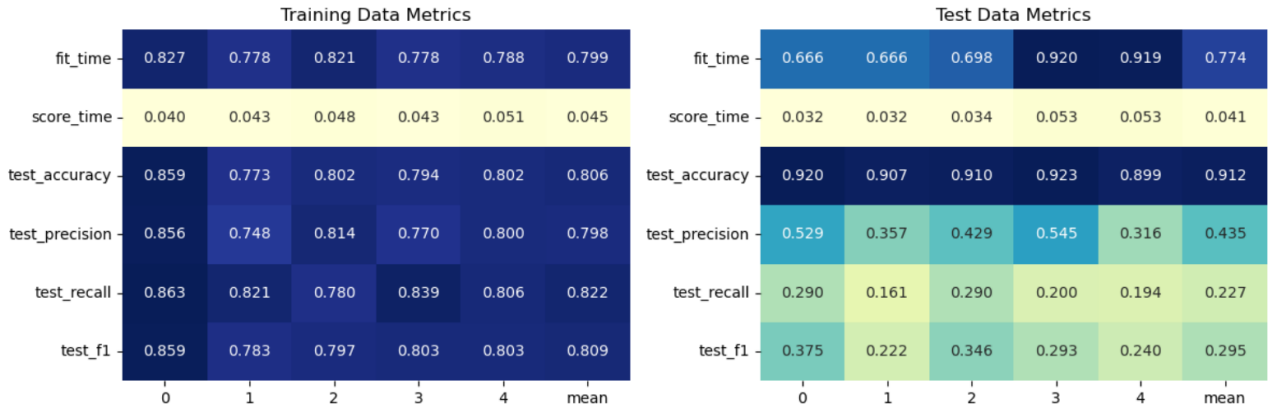
In this context, it is apparent that despite implementing undersampling and feature reduction, there are minimal alterations in the results. Nevertheless, adhering to Occam's Razor principle, this model is preferable as it yields comparable outcomes with a more concise feature set.

The best results for the **Random Forest** on the test set are:

- Accuracy: 0.7843450479233227
- Precision: 0.232409381663113
- Recall: 0.7077922077922078
- F1-Score: 0.34991974317817015

In this scenario, one can observe a decrease in both accuracy and recall. However, precision shows an improvement, and given that it is our primary performance metric, it is possible to say that this model performs better.

d) For comparison, evaluate the model against the test dataset. Show the performance metrics in a heatmap.



Analyzing the metrics between training and test datasets provides key insights into model performance:

- A disparity where training metrics significantly outperform test metrics might indicate overfitting, where the model has learned the training data too well and may not generalize effectively to unseen data.
- Conversely, when test metrics are on par with, or even exceed, training metrics, it suggests that the model is generalizing well to new, unseen data.

Upon evaluation, the test metrics here largely outperform the training metrics, particularly in terms of accuracy, precision, and f1-score. This is an encouraging sign, indicating that the model is likely well-tuned and adept at handling new data.

REFERENCES

Arroyo, J., Corea, F., Jimenez-Diaz, G., & Recio-Garcia, J. A. (2019). Assessment of machine learning performance for decision support in venture capital investments. IEEE Access, 7, 124233-124243