To the moon: finding fast-growing firms for investment opportunities[1]

1. Introduction

Correctly identifying fast-growing companies is a key task at many investment firms. Let's assume that we are working for a venture-capital or private equity fund, and we are tasked with building a model that can help us pick out companies that are expected to have considerable growth in the near future. To do so, we shall use company-level financial data from the past. We want to have such a model that is good for predicting the probability of a firm being fast-growth, but our end goal is to correctly classify companies as fast-growing or non-fast-growing. The key steps of building such a model is deciding on our target variable (that is, how do we define fast growth), engineering potential features for prediction in our dataset, finding the best-performing model through cross-validation and performance metrics, finding the most suitable classification threshold, and analyzing the expected loss of our best classification model. This paper will guide you through these steps.

2. Data preparation

As we were working with the already cleaned Bisnode panel dataset (39375 firms observed between 2010 and 2015, not balanced), preparing the data mainly consisted of label and feature engineering, with paying special attention to missing values. Importantly, I filtered my sample to have sales between 1000 and 10 million EUR in 2012, as I deemed smaller or larger companies would be inappropriate for my business case. After the data preparation, I ended up with a 2012 cross-section of 19036 firms.

2.1 Label engineering

One of the key questions of this exercise was how to define fast growing firms. Given the assumption that we are building this model for an investment fund, we would want to classify such firms as fast growing whose growth can be translated into a profit-making exit opportunity. Of course, the most appropriate measure for this would have been looking at growth in company valuation over time. However, as we do not have market valuation data in our database, we could have only calculated the valuation based on financial reports, which is known to be a very imperfect measure of actual market value. Thus, I have resorted to a simpler approach: simply looking at growth in sales, as that most often translates to market value growth as well.

More specifically, I have looked at the compound annual growth rate (CAGR) of every company over a two-year period between 2012 and 2014. I believe looking at the 2-year CAGR is more appropriate than simply calculating it for one year, as this way we can filter out companies who could not sustain their growth, thus would be inappropriate to invest in. The next question that arises is what level of 2-year CAGR means fast growth. I believe what an investment fund should be looking at is which firms could beat the market in the given period – therefore categorizing such firms as fast growth that had a higher 2-year CAGR than the sample mean[2]. In addition, those firms that defaulted (or simply exited the market) by 2014 were categorized as non-fast growing. As the CAGR values have a very skewed distribution, this meant categorizing only around 12% of firms alive in 2012 as fast growing.

2.2. Feature engineering

2.2.1. Dropped features

Some features had to be dropped from the sample either because they had too many missing values or because there was simply another variable already capturing the information content of them. For the first type, I dropped all initial features that had missing values for more than 90% of the firm-year observations. For the second issue, I ended up not using the foreign and female manager share variables, as the CEO origin and gender categorical variables already contained this information. In addition, I did not use the length of the balance sheet, but only a flag to indicate whether the balance sheet relates to a whole year.

[1] Codes and data files available at my public GitHub repository.

[2] Of course, the sample mean is a very imperfect measure in this case for the market average, as we only have a few industries in our database. But if we assume that the fund is focusing only on these industries, then it should be fine.

2.2.2. Modifying and adding features

I have calculated both the firms' and the CEOs' age from the dataset (with the addition of dummies for new firms and young CEOs) as new features. For the CEO age and the average number employees, I imputed missing values with the sample mean and added a flag for these. I have corrected apparent errors[3] in key balance sheet and P&L variables and included flags for the changes. For the most important financial metrics, I have also calculated their financial ratio versions by dividing them either with total assets or sales. I have taken logs of strictly positive financial features, and I have added differences for the previous two years (for some level financial variables as well). Then I winsorized most of my variables below the 5th and above the 95th percentiles[4], and I added dummies for high and low values. As suggested by LOWESS estimates, I added squared and cubed terms for my continuous features[5]. Lastly, I have dropped observations which had missing values in some key features which could not be imputed, like industry code, firm age, origin of CEOs, region of the firm, liquid assets over total assets and material expenditures over sales.

2.2.3. Interactions

For finding the optimal LASSO logit model, I also defined a very vast set of interactions. I interacted the 2-digit industry code with age, CEO age, origin and gender of the CEOs and the urban category of the firm. I have also interacted the above features with log sales, log total assets and profit/loss (and their two differences). Lastly, I interacted the above continuous features (and their two differences) with each other.

3. Modeling for probability prediction

3.1. Model building

To find the most appropriate model for this task, I trained three different models (a LASSO logit[6], a Random Forest classifier, and an XGBoost classifier). The models were trained on a randomly selected 80% of my sample, while I reserved the other 20% as a hold-out set. The most suitable hyperparameters were found through 5-fold cross-validation for all the models, scoring by AUC[7]. Because of computational constraints, the XGBoost model was tuned in two steps: first, a larger training grid was evaluated with lower number of boosting rounds by a 3-fold cross-validated halving grid search[8]. Then, for the most prospective parameters a regular, 5-fold cross-validated grid search was run with increased boosting rounds.

The input features for LASSO logit were practically all features created during data preparation (with only log transformed and winsorized variables where applicable, with appropriate flags). This meant 602 features (with categorical features encoded as binaries, excluding one category for each) in total. For these, LASSO logit ended up shrinking 422 coefficients to zero, thus selecting 180 features. For the tree-based models, the input features consisted of raw vars only (that is, without logs and without winsorization), with the inclusion of appropriate flags created during data preparation. This meant 122 input features (with categorical features one-hot-encoded).

3.2. Model performance

Table 1 presents key performance metrics of the three models both during cross-validation and on the hold-out set. First, let's focus on the cross-validated mean test metrics. These suggest that our LASSO logit and probability forest models perform practically the same, with a very slight edge towards the former in terms of AUC. The best model seems to be the probability XGBoost, which outperforms both the

[3] E.g. negative values for theoretically non-negative features.

[4] I used this more basic, less aware approach as I had more than 60 features to winsorize. Some variables were already capped and floored by hand during the creation of e.g. CEO age or financial ratios – I did not winsorize these further.

[5] Logged values for 2012, winsorized features and polynomial terms will only be used for LASSO logit models.

[6] As I had a couple hundred features to select from and too little domain knowledge to manually construct promising models.

[7] This yielded identical results for the LASSO logit and the probability forest models as scoring by the RMSE. For the probability XGBoost, the RMSE pointed to another hyperparameter set then AUC (though the best model by AUC had only a marginally worse RMSE than the best RMSE, and vice versa). I decided to use the best model by AUC as the end goal was classification.

[8] This starts cross-validation by only a fraction of the data and eliminates the worse-performing hyperparameters in each iteration. Then, the smaller and smaller hyperparameter grids are evaluated on more and more data. This approach speeds up training significantly.

previous models, both in terms of RMSE and AUC – though this performance increase is rather miniscule.

Next, looking at the hold-out set results, we can see that there is an insignificant change in performance for all three models in both metrics. This suggests that the models are rather properly tuned and they do not overfit the training data. In addition, we can see the same pattern between the models as we have noted above, with the probability XGBoost having a slight edge over the other two models in both hold-out metrics. Lastly, looking at the training times, we can observe that the slightly higher performance of the XGBoost model comes at the cost of significantly higher training times.

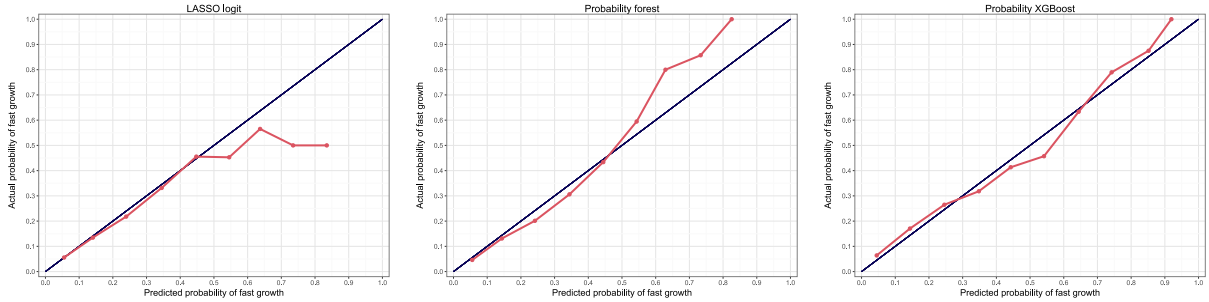
Table 1: Model performance for probability prediction

	CV mean test RMSE	Hold-out RMSE	CV mean test AUC	Hold-out AUC	Training time (minutes)
LASSO logit	0.3121	0.3118	0.7483	0.7444	2.42
Probability forest	0.3111	0.3097	0.7468	0.7456	17.93
Probability XGBoost	0.3094	0.3079	0.7647	0.7555	24.50

Notes: All models trained using all available CPU cores.

The metrics presented in Table 1 do not tell everything about the performance of our models. In addition, we shall examine the calibration plots as well to see how well the predicted probabilities compare to the actual ones. Figure 1 presents these calibration plots. These reveal that both the LASSO logit and the probability forest models are rather badly calibrated: the former overpredicts relatively high probabilities, while the latter usually underpredicts them. The best-calibrated model seems to be the probability XGBoost, as it stays very close to the diagonal even in the case of high probabilities. However, we should not worry too much about miscalibration for the higher range: just as fast-growing firms were rather rare in our sample; high predicted probabilities are also quite rare for all the models.

Figure 1: Calibration plots on the hold-out set



Notes: For all models, 10 equal-sized bins used over the 0–1 range.

To sum things up, we can clearly see that of the three trained models, the probability XGBoost's performance is the best in terms of probability predictions. This is true for the cross-validated performance metrics, for the hold-out set performance metrics, as well as for the calibration plots. Thus, if our goal was simply to predict the probability of a firm being fast-growing, then the best model for this would be the probability XGBoost.

4. Classification

4.1. Loss function

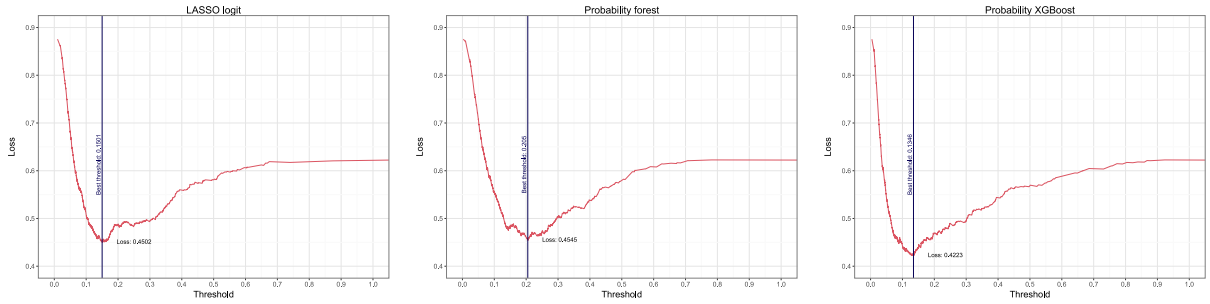
First, let's define our loss function. We know that there are relatively few fast-growth firms out there, so we would like to capture as many of them as possible - even if that means that we will put at least part of our money into bad-performing companies. This means that a false negative is more costly for us (as we miss capitalizing on potentially extreme growth) than a false positive (where we only lose our invested capital (or just a part of it), which is not a lot compared to the fund sizes in the industry). But how much is exactly the cost difference? Consider an investment of a thousand USD. As the mean 2-year CAGR in the sample was around 40% for 2012, we may think of fast-growing firms as those that have around a 50% CAGR. Over two years, this means 125% growth of the invested capital, or 1250 USD profit on the 1000 USD invested. We would lose the opportunity of around this much profit in the case of a false negative classification. Now let's think about the other case. We invest 1000 USD into a firm that turns out to be underperforming, or worse, it may even default. So, the most we can lose is 1000 USD (in the case of default), but it would be rather conservative to assume that we would always lose this much, as some

firms may be just below our fast-growth threshold. For simplicity, let's assume a loss of 250 USD for false negatives. Thus, our false negative / false positive loss rate comes out to be 5-to-1.

4.2. Finding the optimal classification threshold

To find the optimal classification threshold for the models described in Section 3 given the above loss function, I cross-validated the threshold over 5-folds using the models with the previously found best hyperparameters. For each model, the final optimal threshold was the average of the fold-wise optimal thresholds. To illustrate the idea of finding the optimal threshold, Figure 2 presents the expected loss associated with certain thresholds for the 5th fold[9], marking the loss-minimizing threshold.

Figure 2: Expected loss versus classification thresholds (CV 5th fold)



4.3. Classification performance

Table 2 presents the key results related to the optimal classification thresholds. As Figure 2 already suggested, the highest cross-validated expected loss is associated with the probability forest. The LASSO logit model has a marginally smaller cross-validated expected loss. However, the probability XGBoost seems to outperform both other models in terms of classification as well. In addition, note that the optimal thresholds for each model are quite different which is natural, given that the predicted probability distributions also differ[10].

As for the expected losses on the hold-out set it seems that the probability forest generalizes worse than the LASSO logit model, but XGBoost still outperforms both. Also note that for the former two models, the expected loss on the hold-out set is marginally higher than the cross-validated one – but for the XGBoost, these are virtually the same. All in all, it seems that the best model for classification is again the probability XGBoost, just as it was the best in terms of probability predictions.

Table 2: Classification performance – optimal thresholds and expected loss

	Avg. CV optimal thresholds	Avg. CV expected loss	Hold-out set expected loss
LASSO logit	0.1868	0.4486	0.4651
Probability forest	0.2076	0.4533	0.4669
Probability XGBoost	0.1363	0.4375	0.4370

Notes: optimal thresholds for false positive cost of 1, false negative cost of 5.

Having established that the probability XGBoost model performs best for classification, we can now examine its confusion matrix on the hold-out set, as shown in Table 3. The model achieves an overall accuracy of approximately 78%, indicating that it correctly classifies a substantial majority of observations. The sensitivity, which measures the proportion of actual fast-growing firms that are correctly identified, is around 56%, suggesting that the model detects more than half of the fast-growing cases. Meanwhile, the specificity, which reflects the proportion of non-fast-growing firms correctly classified, stands at more than 81%, meaning the model effectively filters out non-fast-growing cases.

Table 3: Confusion matrix for probability XGBoost on the hold-out set

	Predicted no fast growth	Predicted fast growth	Total
Actual no fast growth	2706 (71.1%)	629 (16.5%)	3335 (87.6)
Actual fast growth	207 (5.4%)	266 (7.0%)	473 (12.4%)
Total	2913 (76.5%)	895 (23.5%)	3808 (100.0%)

Notes: Percentages represent share of grand total. Classification threshold is 13.63%.

[9] Thus, the thresholds and loss values presented are not the final optimal ones (but they are very close to them). The averaged out threshold and expected loss values will be presented in Table 2.

[10] For more details on the predicted probability distributions, please refer to the technical report.

Let's assess now how useful this model may be for our actual business case. From the perspective of an investment fund, the probability XGBoost model provides a useful but imperfect tool for identifying high-growth companies. With an accuracy of 78% and a specificity of more than 81%, the model is effective at filtering out firms that are unlikely to experience significant growth, allowing investors to focus their attention on a more promising subset of companies. However, its sensitivity of 56% suggests that it fails to identify a substantial share of actual high-growth firms, meaning that relying solely on the model could result in missed investment opportunities. While it cannot replace deeper qualitative analysis and industry expertise, it can serve as a valuable first-pass screening tool, helping investors allocate resources more efficiently and prioritize firms with the highest predicted potential for future growth.

5. Subsample classification

To carry out classification on the two sectors in the dataset, I used the previously introduced probability XGBoost model with the cross-validated hyperparameters. I then trained a separate model for both sectors, and cross-validated the optimal thresholds by sector. After, I carried out classification for both sectors on the respective hold-out sets to calculate the expected loss on unseen data. The results are summarized in Table 4.

Table 4: Sectoral classification performance – optimal thresholds and expected loss

	Avg. CV optimal thresholds	Avg. CV expected loss	Hold-out set expected loss
Service	0.1298	0.4287	0.4129
Manufacturing	0.1323	0.4698	0.5490

Notes: optimal thresholds for false positive cost of 1, false negative cost of 5. Using the probability XGBoost model with previously cross-validated hyperparameters on the whole training sample.

The model's performance varies notably between the service and manufacturing sectors, reflecting differences in classification effectiveness and expected loss. The expected loss metrics indicate that classification is more reliable in the service sector. The hold-out set expected loss for service firms is substantially lower than for manufacturing firms. This suggests that the model struggles more with accurately distinguishing fast-growth firms in manufacturing. Also note that the hold-out set expected loss is significantly higher for manufacturing firms than the cross-validated expected loss, signaling an overfitting issue. The difference in expected loss implies that while the model provides some value for both sectors, it is a more reliable screening tool for service firms than for manufacturing firms, where misclassification costs are expected to be higher.

6. Discussion

This paper highlights the complexity of identifying fast-growing firms and the trade-offs involved in model selection. While all three models demonstrated reasonable predictive power, the probability XGBoost classifier consistently outperformed the others. This superior performance, however, came at the cost of significantly higher training times, which may pose challenges for real-world implementation. Additionally, the results underline the challenges of defining fast growth using available financial data, as the choice of a growth measure – such as sales CAGR – introduces inherent limitations. The selection of an appropriate metric is crucial, as it directly influences model performance and the validity of predictions.

Beyond model performance, my analysis emphasizes the strategic implications of predictive modeling in an investment context. Missing out on a high-growth firm can have far greater financial consequences than mistakenly investing in a slow-growing one, making recall a higher priority than precision. This trade-off is particularly relevant for venture capitalists and private equity firms that seek to identify high-potential companies early. While optimizing recall improves the chances of capturing truly high-growth firms, it also increases the likelihood of false positives, which may require further qualitative screening to mitigate risks. The findings reinforce the importance of integrating machine learning models with domain expertise to refine decision-making processes and improve overall investment outcomes.

The key takeaway of this paper should be that predictive modeling should not operate in isolation but should be integrated into a broader investment strategy. The most effective approach combines model-driven insights with qualitative assessments and domain expertise to maximize returns and make well-informed investment choices.