Prepared for Prediction with Machine Learning course

This notebook uses code from AI and Gabor's case studies.

NOTE

This technical report notebook does not contain the full executable code. Some cells have been omitted for clarity and brevity. Running this notebook as-is will result in errors. We have retained only the most meaningful outputs and visualizations relevant for interpretation.

The full working code is available in our GitHub repository here.

Assignment Overview

The goal of this assignment is to predict which firms are likely to become high-growth using firm-level data from the Bisnode panel (2010–2015). We use a 2012 cross-section to train machine learning models and evaluate their performance in identifying high-growth firms in the following year.

Target Definition & Economic Intuition

Corporate Finance Motivation for Target Choice

We define fast growth based on a one-year percentage change in sales, aligning with the OECD standard for high-growth enterprises. From a corporate finance perspective, short-term revenue expansion reflects a firm's ability to generate and scale operations efficiently, often signaling strong demand, effective reinvestment of earnings, and operational leverage. Using a 1-year window captures more immediate performance dynamics and supports timely decision-making for investors or lenders who seek early indicators of growth potential. Longer horizons risk introducing noise from market cycles or external shocks, reducing predictive clarity.

Designing target variable

Exploration

We aim to compare three types of sales growth measures:

1-year growth: The percentage change in sales over a single year (e.g., 2011 to 2012).

2-year growth: The percentage change in sales over a two-year period (e.g., 2012 compared to 2010).

Average 1-year growth over 2 years: The average of two consecutive 1-year growth rates (e.g., average of growth from 2010–2011 and 2011–2012).

Method 1: **year-over-year percentage change in sales**

Growth rate is calculated as the **forward-looking change in sales**, i.e., how much a firm's sales increase (or decrease) in the **next year** relative to the current year

This threshold is based on OECD definition and it reflects a substantial year-over-year expansion.

```
# Sort by firm and year
data = data.sort_values(["comp_id", "year"])

# Calculate forward-looking growth rate (future sales vs. this year)
data["growth_rate"] = data.groupby("comp_id")["sales"].shift(-1) /
data["sales"] - 1

# Replace inf, -inf with NaN to avoid polluting calculations
data['growth_rate'] = data['growth_rate'].replace([np.inf, -np.inf],
np.nan)

# Step 1: Filter out the 99th percentile of growth rates
growth_rate_99th_percentile = data['growth_rate'].quantile(0.99)
data_filtered = data[data['growth_rate'] <=
growth_rate_99th_percentile]

# Step 2: Define high-growth firms (growth > 20%)
data_filtered['HG_turnover'] = (data_filtered['growth_rate'] >
0.2).astype(int)

# Step 2.1: Create a shifted HG column to track previous year's status
data_filtered["HG_turnover_prev"] = data_filtered.groupby("comp_id")
["HG_turnover"].shift(1)

# Step 2.2: Define transitions
data_filtered["HG_entry"] = np.where(
    (data_filtered["HG_turnover_prev"] == 0) &
(data_filtered["HG_turnover"] == 1), 1, 0)

data_filtered["HG_exit"] = np.where(
    (data_filtered["HG_turnover_prev"] == 1) &
(data_filtered["HG_turnover"] == 0), 1, 0)

# Step 3: Summarize transitions and growth metrics by year
hg_summary = data_filtered.groupby('year').apply(
    lambda x: pd.Series({
        'num_firms': x['comp_id'].nunique(),
        'hg_firms_share_percentage_next_period':
(x['HG_turnover'].sum() / x['comp_id'].nunique()) * 100,
        'avg_growth_rate_hg': x.loc[x['HG_turnover'] == 1,
'growth_rate'].mean(),
        'avg_growth_rate_non_hg': x.loc[x['HG_turnover'] == 0,
'growth_rate'].mean(),
```

```
        'hg_entry_count': x['HG_entry'].sum(),
        'hg_exit_count': x['HG_exit'].sum(),
        'hg_entry_share_percentage': (x['HG_entry'].sum() /
x['comp_id'].nunique()) * 100,
        'hg_exit_share_percentage': (x['HG_exit'].sum() /
x['comp_id'].nunique()) * 100
    })
).reset_index()

# Step 4: Clean up the table
pretty_hg_summary = (
    hg_summary
    .rename(columns={
        'year': 'Year',
        'num_firms': 'Firms',
        'hg_firms_share_percentage_next_period': 'HG Share (%)',
        'avg_growth_rate_hg': 'Avg Growth (HG)',
        'avg_growth_rate_non_hg': 'Avg Growth (Non-HG)',
        'hg_entry_count': 'HG Entries',
        'hg_exit_count': 'HG Exits',
        'hg_entry_share_percentage': 'Entry Share (%)',
        'hg_exit_share_percentage': 'Exit Share (%)'
    })
    .round({
        'HG Share (%)': 2,
        'Avg Growth (HG)': 2,
        'Avg Growth (Non-HG)': 2,
        'Entry Share (%)': 2,
        'Exit Share (%)': 2
    })
)

# Convert count columns to integers
for col in ['Firms', 'HG Entries', 'HG Exits']:
    pretty_hg_summary[col] = pretty_hg_summary[col].astype(int)

# Display nicely
from IPython.display import display
display(pretty_hg_summary)
```

```
   Year  Firms  HG Share (%)  Avg Growth (HG)  Avg Growth (Non-HG)  \
0  2009  18709         28.26             2.23                -0.20
1  2010  19100         32.10             2.05                -0.18
2  2011  19640         29.89             2.08                -0.20
3  2012  19703         32.08             1.98                -0.18
4  2013  19728         36.97             1.78                -0.16
5  2014  19711         33.80             1.65                -0.14

   HG Entries  HG Exits  Entry Share (%)  Exit Share (%)
0           0         0             0.00            0.00
```

```
1        2859      3239             14.97            16.96
2        2553      4060             13.00            20.67
3        3158      3708             16.03            18.82
4        3684      3707             18.67            18.79
5        3210      4626             16.29            23.47
```

For comparison: Methods 2 and 3

Method 2: percentage change in sales over a two-year period

This method measures a firm's sales growth from year *t* to *t+2* as a percentage change.

A firm is classified as high-growth if this two-year growth exceeds 20%.

```
display(result)
```

```
   year  num_firms  hg_firms_share_percentage  avg_growth_rate_hg  \
0  2009    16775.0                  33.698957            2.952986
1  2010    17152.0                  35.680970            2.870223
2  2011    17553.0                  36.859796            2.899975
3  2012    17740.0                  43.263811            2.541637
4  2013    18001.0                  45.908561            2.445419

   avg_growth_rate_non_hg    approach
0               -0.257597  per_2_year
1               -0.263967  per_2_year
2               -0.257321  per_2_year
3               -0.228713  per_2_year
4               -0.206262  per_2_year
```

! 2-Year growth of High-Growth firms is higher than their 1-Year growth

While the 2-year growth measure can highlight firms with more **sustained growth**, it also introduces certain challenges:

- **Lagged signal**: It may delay identifying high-growth firms until after two full years of data are available.
- **Data sensitivity**: Requires complete data across a longer time frame, increasing vulnerability to missing or inconsistent entries.
- **Inflated values**: The higher average growth may affect model calibration or threshold selection, especially in classification tasks.

Method 3 (additional): Average growth over two 1-year periods

In this approach, we define high-growth firms based on their **average sales growth across the next two years**. Specifically, for each firm-year `t`, we calculate:

- Growth from year `t` to `t+1`
- Growth from year `t+1` to `t+2`

We then take the **average** of these two growth rates to get a smoothed, forward-looking indicator of performance:

A firm is labeled **high-growth** if this average exceeds **20%**.

```
display(result_2y_avg)
```

```
   year  num_firms  hg_firms_share_percentage  avg_growth_rate_hg  \
0  2009    16440.0                  28.418491            1.649868
1  2010    16852.0                  29.883693            1.676212
2  2011    17278.0                  30.877416            1.656146
3  2012    17504.0                  35.186243            1.467276
4  2013    17761.0                  35.943922            1.398965

   avg_growth_rate_non_hg                    approach
0               -0.095491  avg_growth_2x_1yr_forward
1               -0.096110  avg_growth_2x_1yr_forward
2               -0.092175  avg_growth_2x_1yr_forward
3               -0.066894  avg_growth_2x_1yr_forward
4               -0.046852  avg_growth_2x_1yr_forward
```

Advantages:

- **Reduces noise** from short-term spikes or drops in sales
- **Captures sustained growth** over multiple years

Disadvantages:

- Requires data for **three consecutive years**, reducing usable sample size
- May **delay signal detection** — firms with sudden breakout growth may be missed

This method is especially useful when we care about **stable medium-term growth**, rather than one-off performance.

Why We Focus on 2012 Transitions

We chose 2012 as our reference year to ensure a large enough sample while maintaining a balanced rate of high-growth entries and exits.

Rather than only looking at the share of high-growth firms, we analyze **transitions** — how many firms **enter or exit** the high-growth category. This helps capture the **dynamics and mobility** in firm performance over time.

Data Preparation & Feature Engineering

We use the Bisnode firm-level panel dataset covering the years 2010 to 2015. For modeling, we extract a cross-section of firms active in 2012, as it provides the most complete coverage across key financial and organizational variables. The prediction target is based on turnover growth in

the following year (2013). Basic filtering is applied to ensure non-missing outcomes and features.

Feature engineering includes grouping variables into financials, HR, and firm characteristics. We apply log and squared transformations, create dummies and one-hot encode categorical variables, and use flag variables to handle missing values and data quality issues.

All steps are documented in the full Jupyter notebook, available on our GitHub repository folder here.

Sample design

```
# look at cross section
data = data.query("year==2012 & status_alive == 1")

growth_rate_99th_percentile = data['growth_rate'].quantile(0.99)
data = data[data['growth_rate'] <= growth_rate_99th_percentile]

data['growth_rate'].describe()
```

```
count    19701.000000
mean         0.507666
std          2.419315
min         -0.999920
25%         -0.196189
50%          0.034247
75%          0.327473
max         30.431515
Name: growth_rate, dtype: float64
```

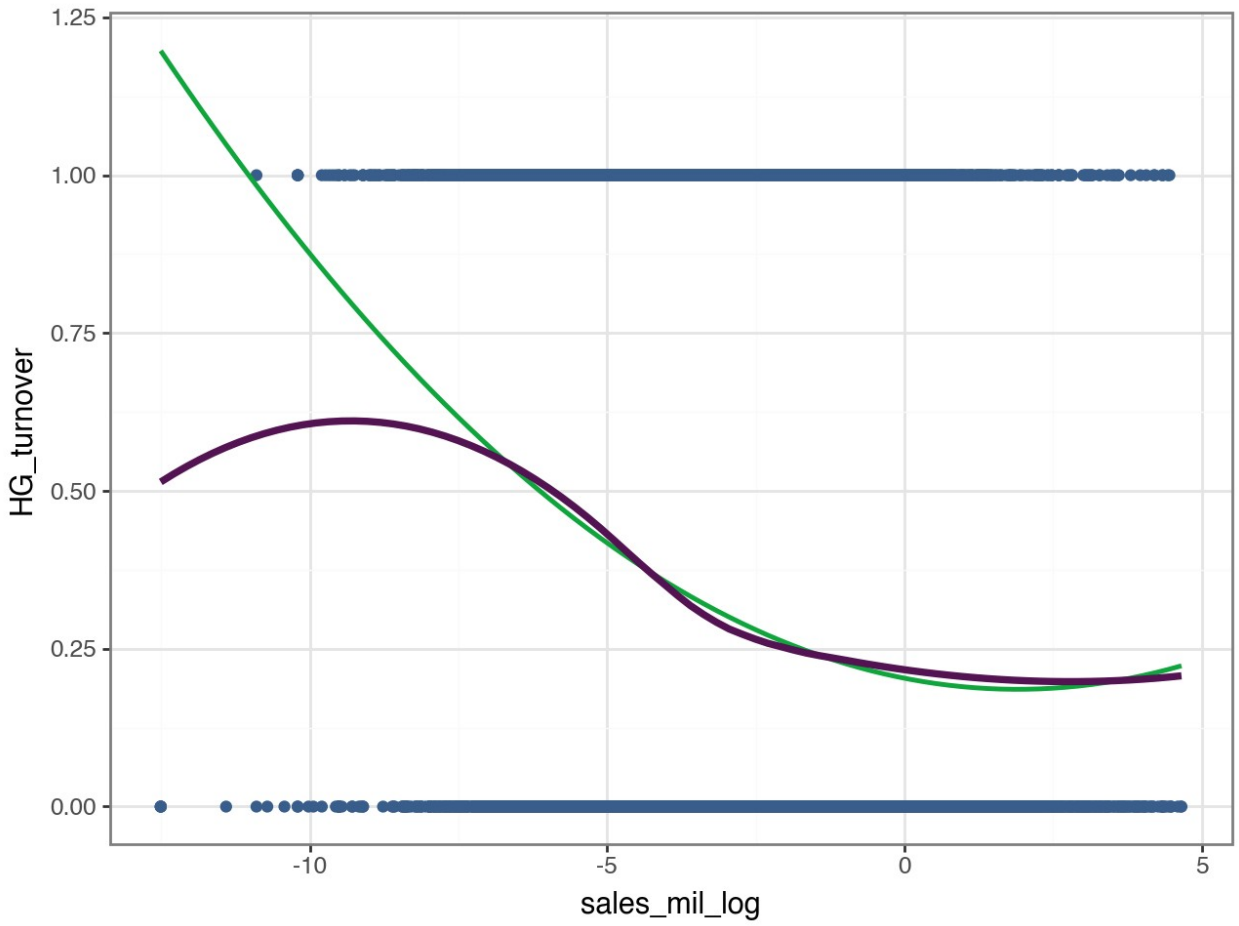
Feature engineering

Exploratory Analysis

Exploratory Visualization 1: Firm Size and High-Growth Probability

```
(
    ggplot(data, aes(x="sales_mil_log", y="HG_turnover"))
    + geom_point(color=color[0])
    + geom_smooth(method="lm", formula="y ~ x + I(x**2)",
color=color[1], se=False)
    + geom_smooth(method="loess", color=color[2], se=False, size=1.5,
span=0.9)
    + labs(x="sales_mil_log", y="HG_turnover")
```

```
    + theme_bw()
)
```



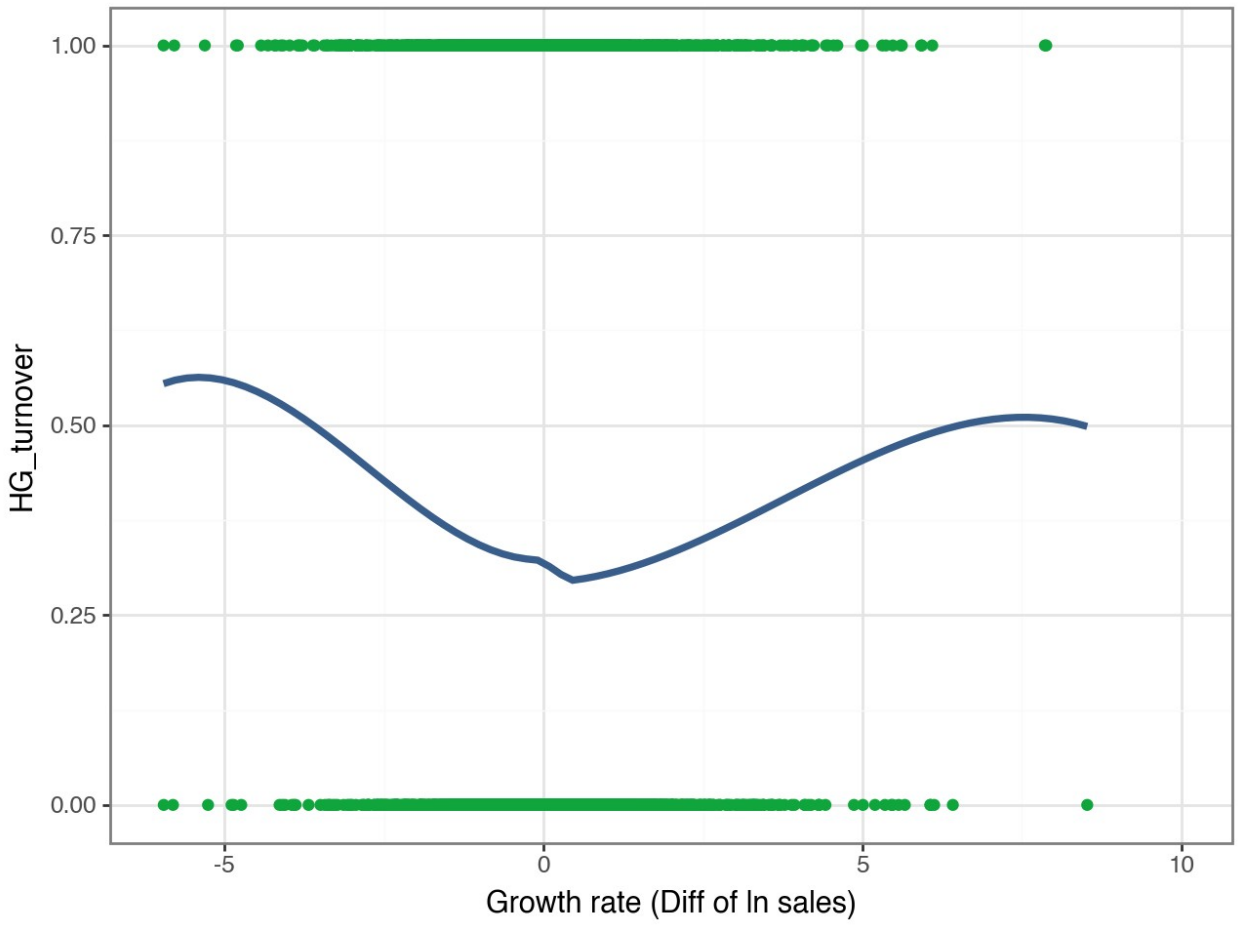
The plot shows the relationship between firm size (sales_mil_log) and the likelihood of being high-growth (HG_turnover). The variable sales_mil_log represents the natural logarithm of firm sales in millions of euros, allowing for better handling of skewed distributions and relative differences between firm sizes.

Two trend lines are included: a quadratic regression line and a LOESS smoother. Both suggest a negative relationship — smaller firms are more likely to be classified as high-growth. This aligns with economic intuition: smaller firms typically have more room to scale rapidly, while larger firms tend to grow more incrementally.

Including both firm size (defined by sales volume) and its square as features in the model captures this nonlinearity effectively. Since sales_mil_log is based on 2012 values and the target measures growth in 2013, this relationship reflects a valid predictive signal and does not introduce target leakage.

Exploratory Visualization 2: Sales Growth and High-Growth Status

```
(
    ggplot(data, aes(x="d1_sales_mil_log", y="HG_turnover"))
    + geom_point(color=color[1])
    + geom_smooth(method="loess", color=color[0], se=False, size=1.5,
span=0.9)
    + labs(x="Growth rate (Diff of ln sales)", y="HG_turnover")
    + scale_x_continuous(limits=(-6, 10), breaks=seq(-5, 10, by=5))
    + theme_bw()
)
```



The plot shows the relationship between recent sales growth (d1_sales_mil_log) and the probability of being classified as high-growth (HG_turnover). Each green dot represents a firm, while the blue line is a loess smoother capturing the trend.

Interestingly, the curve shows a nonlinear, U-shaped relationship. Firms with very low or very high past sales growth have a higher probability of being high-growth in the next year. In contrast, firms with moderate or near-zero growth show a lower probability. This suggests that

both strong positive momentum and recovering firms (from very negative growth) may be more likely to scale.

This pattern is not driven by the 90th percentile cutoff used to define the high-growth label, as d1_sales_mil_log reflects past growth, while HG_turnover captures future outcomes. The relationship supports the inclusion of sales growth as a meaningful predictive feature, while also motivating the use of nonlinear models like Random Forest or Gradient Boosting.

PART 1

Part 1: Full sample

Model building

Choice of models

We compare seven models: five logistic regressions of increasing complexity (M1–M5), a LASSO-regularized logistic model, a Random Forest (RF), and a Gradient Boosting Model (GBM).

1. Logit - A standard baseline model for binary outcomes. It is interpretable and widely used. LASSO adds automatic variable selection and helps avoid overfitting when there are many predictors.
2. Random Forest (RF) – A flexible, nonparametric method that captures complex patterns in the data. It generally performs better than Logit in prediction tasks and doesn't require model-building assumptions.
3. Gradient Boosting Model (GBM) – Builds trees in sequence, improving predictions step by step. It is accurate and effective for both predicting probabilities and classifying outcomes.

Define variable sets

Main firm variables

```
rawvars = [
    "curr_assets",
    "curr_liab",
    "extra_exp",
    "extra_inc",
    "extra_profit_loss",
    "fixed_assets",
    "inc_bef_tax",
    "intang_assets",
    "inventories",
```

```
    "liq_assets",
    "material_exp",
    "personnel_exp",
    "profit_loss_year",
    "sales",
    "share_eq",
    "subscribed_cap",
]
```

Further financial variables

```
qualityvars = ["balsheet_flag", "balsheet_length",
"balsheet_notfullyear"]
engvar = [
    "total_assets_bs",
    "fixed_assets_bs",
    "liq_assets_bs",
    "curr_assets_bs",
    "share_eq_bs",
    "subscribed_cap_bs",
    "intang_assets_bs",
    "extra_exp_pl",
    "extra_inc_pl",
    "extra_profit_loss_pl",
    "inc_bef_tax_pl",
    "inventories_pl",
    "material_exp_pl",
    "profit_loss_year_pl",
    "personnel_exp_pl",
]
engvar2 = [
    "extra_profit_loss_pl_quad",
    "inc_bef_tax_pl_quad",
    "profit_loss_year_pl_quad",
    "share_eq_bs_quad",
]
```

Growth variables

```
d1 = [
    "d1_sales_mil_log_mod",
    "d1_sales_mil_log_mod_sq",
    "flag_low_d1_sales_mil_log",
    "flag_high_d1_sales_mil_log",
]
```

Human capital related variables

```
hr = [
    "female",
    "ceo_age",
    "flag_high_ceo_age",
    "flag_low_ceo_age",
    "flag_miss_ceo_age",
    "ceo_count",
    "labor_avg_mod",
    "flag_miss_labor_avg",
    "foreign_management",
]
```

Firms history related variables

```
firm = ["age", "age2", "new", "C(ind2_cat)", "C(m_region_loc)",
"C(urban_m)"]
```

interactions for logit, LASSO

```
interactions1 = [
    "C(ind2_cat)*age",
    "C(ind2_cat)*age2",
    "C(ind2_cat)*d1_sales_mil_log_mod",
    "C(ind2_cat)*sales_mil_log",
    "C(ind2_cat)*ceo_age",
    "C(ind2_cat)*foreign_management",
    "C(ind2_cat)*female",
    "C(ind2_cat)*C(urban_m)",
    "C(ind2_cat)*labor_avg_mod",
]
interactions2 = [
    "sales_mil_log*age",
    "sales_mil_log*female",
    "sales_mil_log*profit_loss_year_pl",
    "sales_mil_log*foreign_management",
]
```

We constructed five sets of predictors (M1 to M5) with increasing complexity.

- M1 includes only basic firm-level characteristics. Whereas each subsequent model adds more variables, such as financial ratios, sector dummies, or interaction terms.

- This setup allows us to assess how model performance improves as we add more information.

Random Forest (RF) and Gradient Boosting Model (GBM) are trained on the full set of variables (equivalent to M5). These models can automatically capture interactions and nonlinear relationships, so no manual feature selection or transformation is needed.

Model setups

1. Simple logit models

```
M1 = [
    "sales_mil_log",
    "sales_mil_log_sq",
    "d1_sales_mil_log_mod",
    "profit_loss_year_pl",
    "C(ind2_cat)",
]
M2 = [
    "sales_mil_log",
    "sales_mil_log_sq",
    "d1_sales_mil_log_mod",
    "profit_loss_year_pl",
    "fixed_assets_bs",
    "share_eq_bs",
    "curr_liab_bs ",
    "curr_liab_bs_flag_high ",
    "curr_liab_bs_flag_error",
    "age",
    "foreign_management",
    "C(ind2_cat)",
]
M3 = [    "sales_mil_log",   "sales_mil_log_sq"] + firm + engvar + d1
M4 = (
    [    "sales_mil_log",  "sales_mil_log_sq" ]
    + firm
    + engvar
    + engvar2
    + engvar3
    + d1
    + hr
    + qualityvars
)
M5 = (
    ["sales_mil_log",
    "sales_mil_log_sq"]
    + firm
    + engvar
    + engvar2
    + engvar3
    + d1
    + hr
    + qualityvars
    + interactions1
    + interactions2
)
```

1. logit+LASSO

```
logit_lasso_vars = (
    ["sales_mil_log",
    "sales_mil_log_sq"]
    + engvar
    + engvar2
    + engvar3
    + d1
    + hr
    + firm
    + qualityvars
    + interactions1
    + interactions2
)
```

1. CART and RF (no interactions, no modified features)

```
rfvars = ["sales_mil_log", "sales_mil_log_sq"] + rawvars + hr + firm +
qualityvars
```

0. Separate train and holdout data

```
data_train, data_holdout = train_test_split(data, train_size=0.8,
random_state=42)

print("Total")
print(data["HG_turnover"].value_counts(normalize=True))
print("Train")
print(data_train["HG_turnover"].value_counts(normalize=True))
print("Holdout")
print(data_holdout["HG_turnover"].value_counts(normalize=True))
```

```
Total
HG_turnover
0    0.688737
1    0.311263
Name: proportion, dtype: float64
Train
HG_turnover
0    0.688696
1    0.311304
Name: proportion, dtype: float64
Holdout
HG_turnover
0    0.688901
1    0.311099
Name: proportion, dtype: float64
```

1. Predict probabilities with logit and Lasso with CV

Specify 5 fold cross-validation method

```
k = KFold(n_splits=5, shuffle=True, random_state=42)
```

a) cross validate logit models

Set up X-vars

```
logit_model_vars = [M1, M2, M3, M4, M5]

pd.DataFrame(CV_RMSE_folds)
```

	M1	M2	M3	M4	M5
0	0.453674	0.448324	0.441399	0.439223	0.439766
1	0.453006	0.447705	0.443118	0.440257	0.440562
2	0.456544	0.449611	0.447823	0.444704	0.446635
3	0.451503	0.445823	0.439853	0.437845	0.438358
4	0.450207	0.444982	0.437950	0.435802	0.435067

b) Logit + LASSO

```
model_equation = "HG_turnover~" + "+".join(logit_lasso_vars)
y_train, X_train = patsy.dmatrices(model_equation, data_train)
```

See CV-fold RMSE-s (negative brier score)

```
cv_summary_lasso = cv_summary(lambdas, C_values,
logit_models["LASSO"])
cv_summary_lasso["mean_cv_score"] =
np.sqrt(cv_summary_lasso["mean_cv_score"] * -1)
cv_summary_lasso
```

	lambdas	C_values	mean_cv_score
0	0.100000	0.000876	0.473713
1	0.046416	0.001887	0.453217
2	0.021544	0.004066	0.444684
3	0.010000	0.008760	0.441233
4	0.004642	0.018873	0.439632
5	0.002154	0.040661	0.439533
6	0.001000	0.087602	0.439746
7	0.000464	0.188734	0.440259
8	0.000215	0.406615	0.440731
9	0.000100	0.876025	0.441169

Save best lambda's index for later use

```
best_lambda_i
```

```
5
```

2. AUC, Calibration Curve, Confusion Matrix, ROC

1. Calculate AUC for folds

First, for logits, then for Lasso

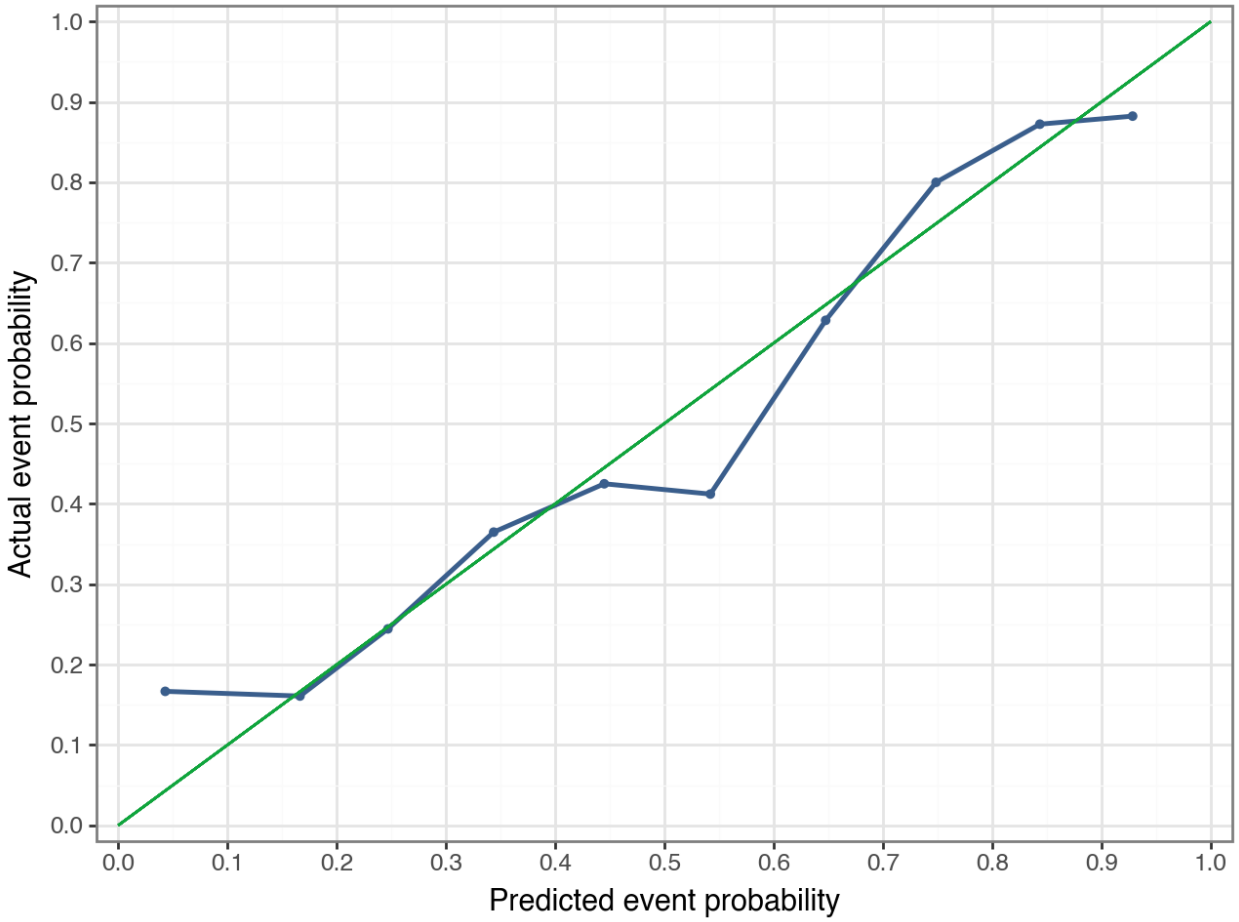
```
summary
```

	Number of Coefficients	CV RMSE	CV AUC
M1	16	0.452987	0.618541
M2	23	0.447289	0.651345
M3	40	0.442029	0.672425
M4	83	0.439566	0.676471
M5	197	0.440077	0.676408
LASSO	101	0.439519	0.677835

a) Calibration curve

how well do estimated vs actual event probabilities relate to each other?

```
create_calibration_plot(
    data_holdout,
    file_name="da3-a3-logit-m4-calibration",
    prob_var="best_logit_pred_proba",
    actual_var="HG_turnover",
    y_lab="Actual event probability",
    n_bins=10,
    breaks=None,
)
```



This plot shows how well the predicted probabilities match actual outcomes. The model is reasonably well calibrated, especially in the mid-to-high probability range. It is important to point out that it tends to be slightly underconfident when it predicts a high chance of success, the real success rate is often even higher.

This is a good sign: the model is cautious rather than overconfident. It suggests that in high-probability cases, we can trust the predictions even more than the model does.

A sensible choice: mean of predicted probabilities

```
mean_predicted_default_prob =
np.mean(data_holdout["best_logit_pred_proba"])
round(mean_predicted_default_prob, 3)
```

```
0.316
```

```
holdout_prediction = np.where(
    data_holdout["best_logit_pred_proba"] <
mean_predicted_default_prob, 0, 1
)
cm_object2 = confusion_matrix(
    data_holdout["HG_turnover"], holdout_prediction, labels=[0, 1]
```

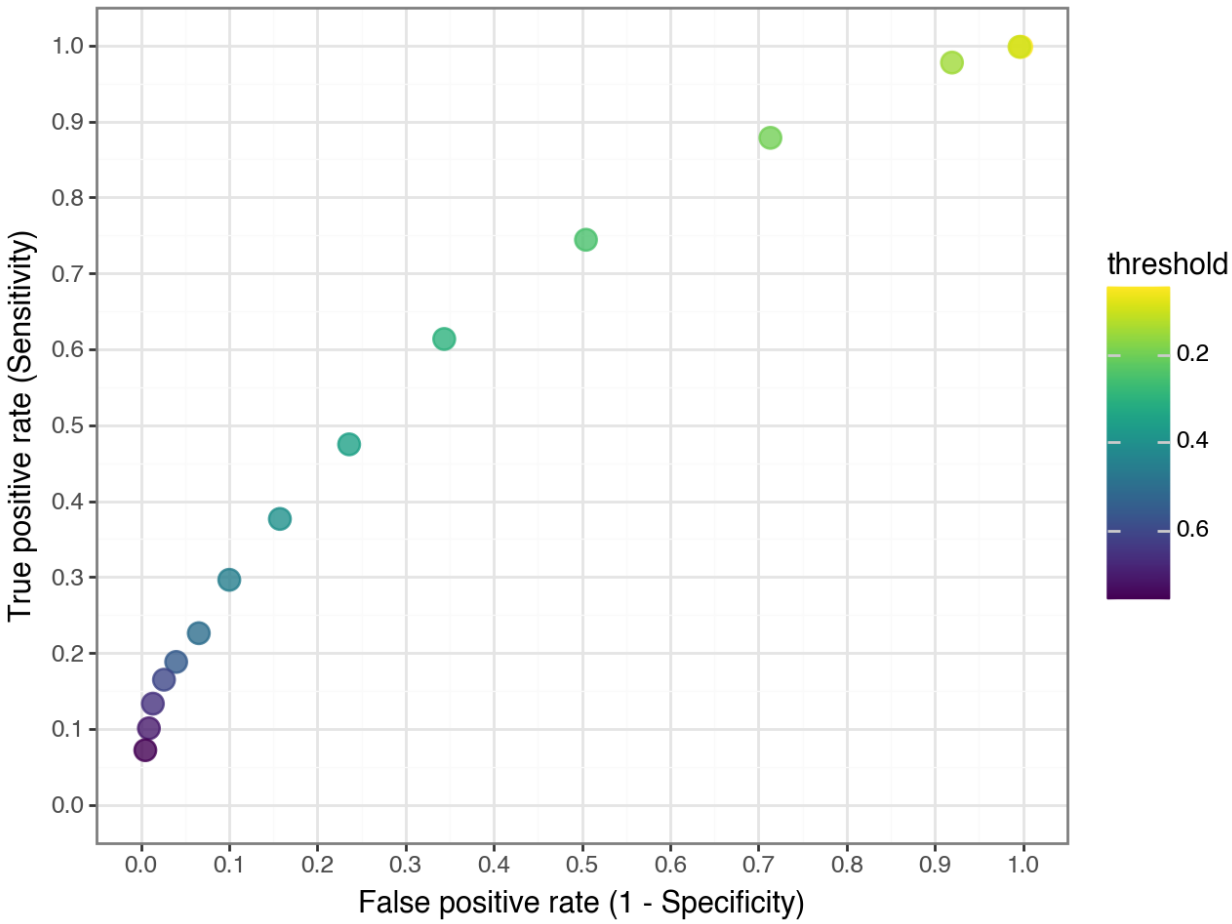
```
)
cm2 = pd.DataFrame(
    cm_object2,
    index=["Actul non-HG", "Actual HG"],
    columns=["Predicted non-HG", "Predicted HG"],
)
cm2
```

	Predicted non-HG	Predicted HG
Actul non-HG	1720	738
Actual HG	487	623

c) Visualize ROC (with thresholds in steps) on holdout

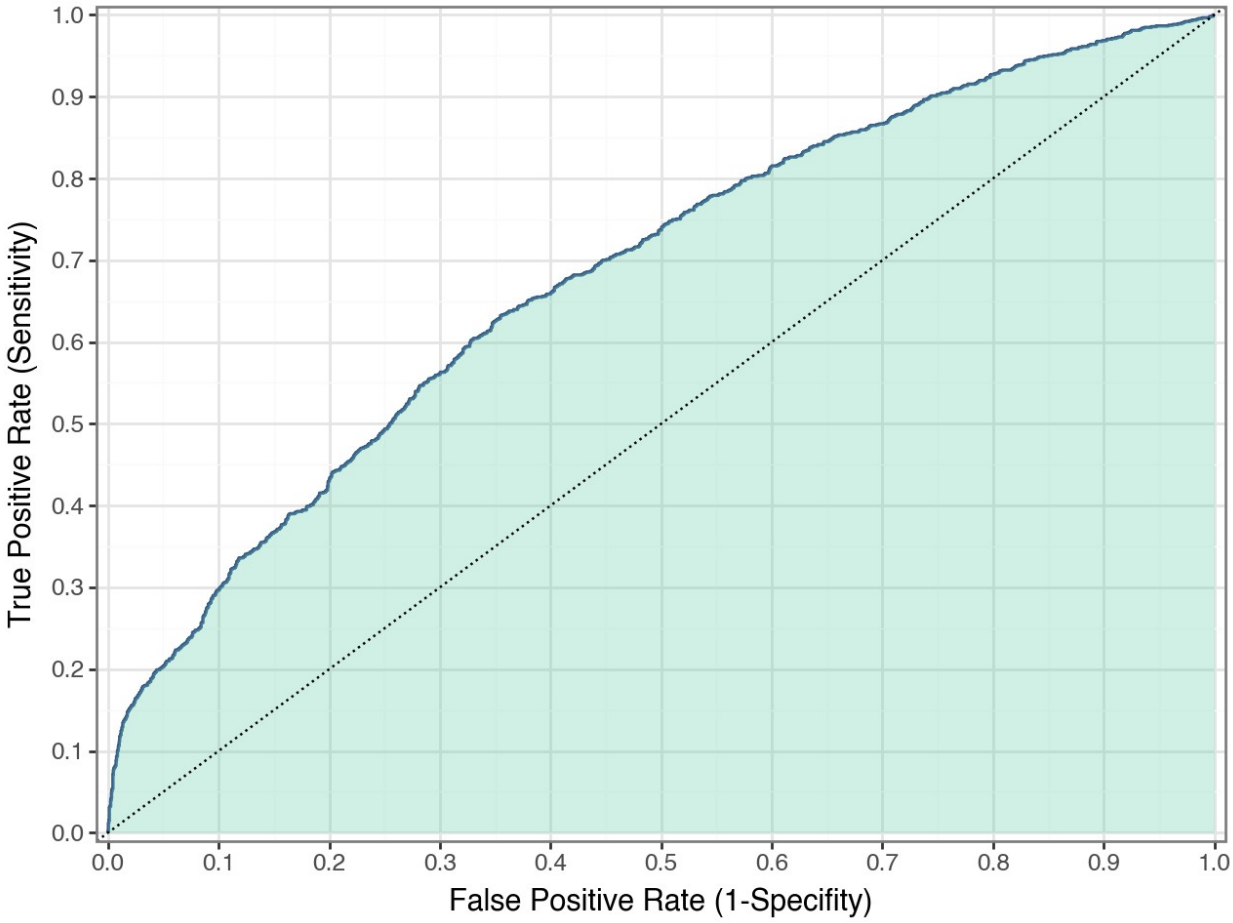
The **discrete ROC plot** highlights how different classification thresholds impact the trade-off between true and false positives, while the **continuous ROC curve** confirms that the model consistently performs better than random guessing across all thresholds. This supports the reliability for classification.

```
(
    ggplot(
        tpr_fpr_for_thresholds,
        aes(x="false_positive_rates", y="true_positive_rates",
color="thresholds"),
    )
    + labs(
        x="False positive rate (1 - Specificity)", y="True positive
rate (Sensitivity)"
    )
    + geom_point(size=4, alpha=0.8)
    + scale_color_continuous(trans="reverse", name="threshold")
    + scale_x_continuous(limits=(0, 1), breaks=np.arange(0, 1.01,
0.1))
    + scale_y_continuous(limits=(0, 1), breaks=np.arange(0, 1.01,
0.1))
    + theme_bw()
    + theme(legend_position="right")
)
```



Continuous ROC on holdout with best model

```
create_roc_plot(data_holdout["HG_turnover"],
data_holdout["best_logit_pred_proba"])
```



Classification & Loss Function

In this analysis, we adopt the perspective of a risk-tolerant investor who values capturing high-growth opportunities, even at the risk of occasional misclassifications.

Finding the optimal classification threshold

For each model, we first predict probabilities, then search for the best classification threshold by minimizing expected loss.

Instead of using a fixed cutoff like 0.5, we adjust the threshold to reflect the cost of mistakes. In our case, missing a high-growth firm (false negative) is much costlier than backing a low-growth one (false positive). This approach helps align model decisions with investor priorities, where the opportunity cost of missing potential winners is high.

```
FP = 1
FN = 4
cost = FN / FP
```

Intuition: From a corporate finance perspective, fast growth typically reflects the presence of strong reinvestment capabilities, increasing returns to scale, or strategic expansion

opportunities. For example for investors, the opportunity cost of missing a high-growth firm often outweighs the loss from investing in a low-growth one, thus motivating our asymmetric loss function in the classification task.

The prevalence, or the proportion of cases in the population (n.cases/(n.controls+n.cases))

```
prevelance = y_train.sum() / len(y_train)
```

Draw ROC Curve and find optimal threshold with loss function

The optimal cut-off is the threshold that maximizes the distance to the identity (diagonal) line

Iterate through:

1. models
2. Folds

In the end, each model will be evaluated using four metrics: **RMSE** (overall prediction error), **AUC** (ranking quality of predicted probabilities), **Optimal threshold** (chosen to minimize expected loss), **Expected loss** (based on asymmetric cost of false negatives and false positives).

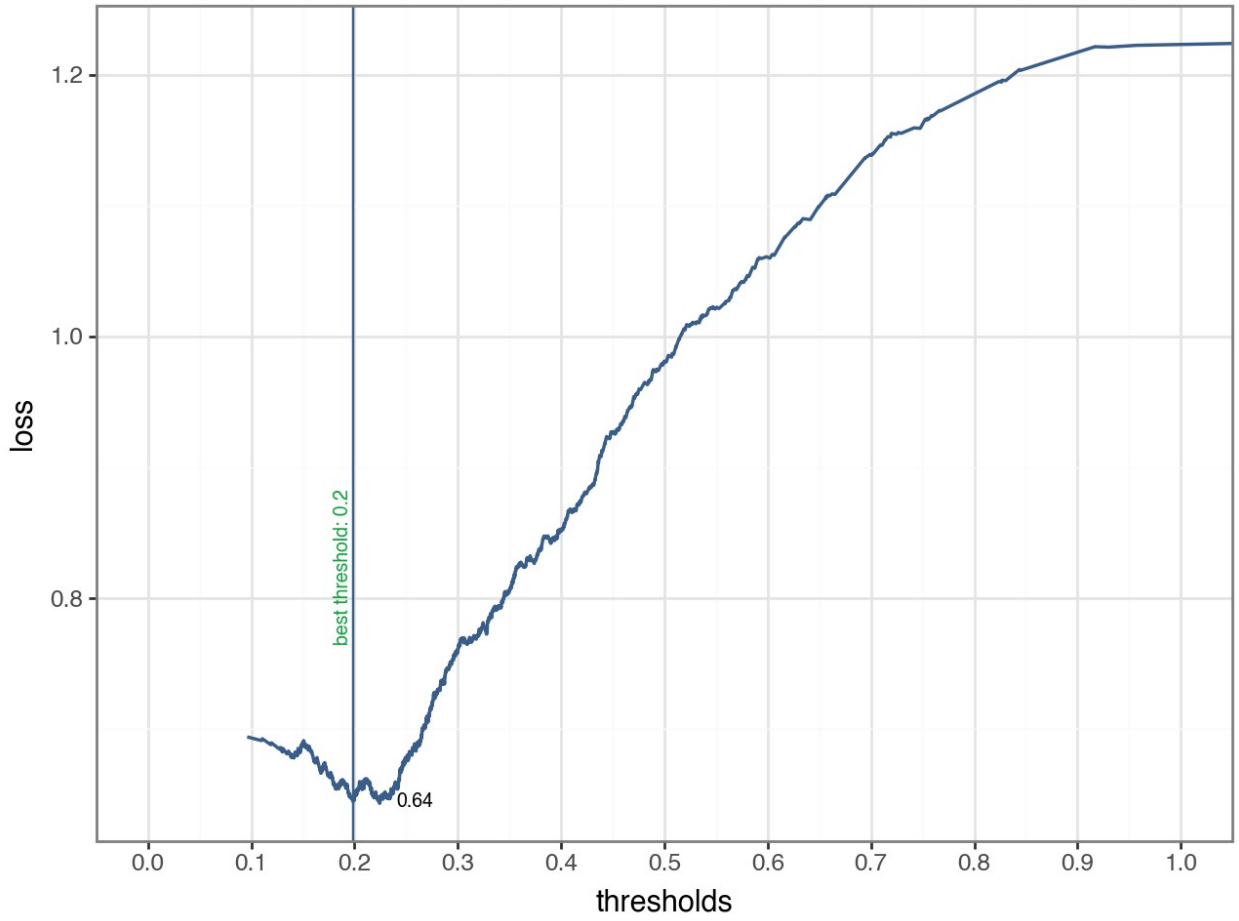
Expected loss is the most decision-relevant metric because it incorporates the asymmetric costs of false positives and false negatives. Unlike RMSE or AUC, it links model performance directly to real-world consequences, enabling cost-sensitive model selection.

```
summary_with_lossfnc
```

```
   Model  Avg of optimal thresholds  Threshold for Fold5  Avg expected
loss  \
0     M1                   0.200771             0.201370
0.672017
1     M2                   0.213057             0.172817
0.663537
2     M3                   0.194024             0.211113
0.645736
3     M4                   0.212004             0.198559
0.642862
4     M5                   0.193520             0.198828
0.640690
5  LASSO                   0.210698             0.203597
0.640759

   Expected loss for Fold5
0                 0.677883
1                 0.669120
2                 0.651945
3                 0.644585
4                 0.640379
5                 0.633368
```

Create loss plot based on Fold5 in CV

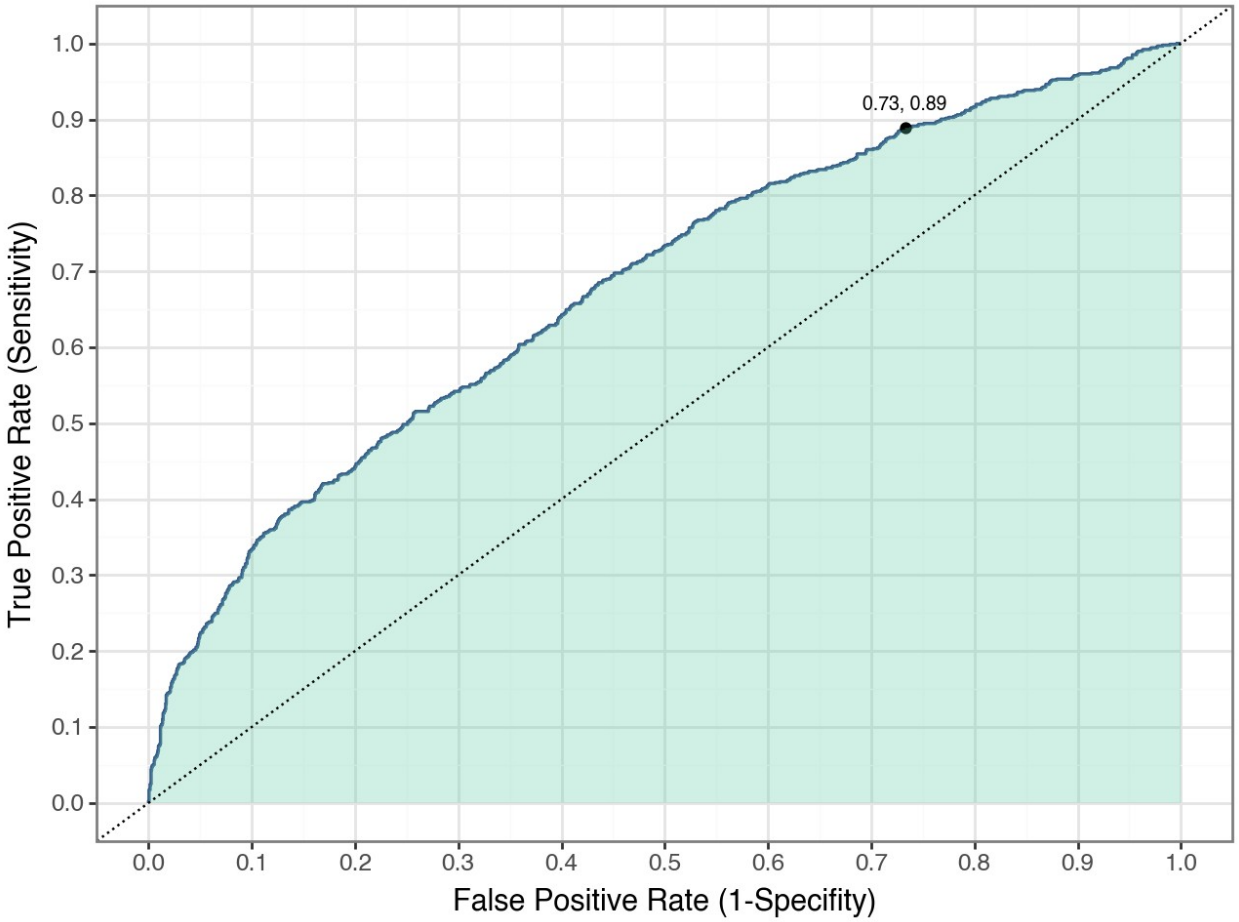
```
model_to_plot = "M4"  # select model to plot
create_loss_plot(
    fold5_all_coords[model_to_plot],
    fold5_threshold[model_to_plot],
    fold5_expected_loss[model_to_plot],
)
```



This plot supports our decision to move away from the default 0.5 threshold and instead use a lower, loss-minimizing threshold, which better reflects the higher cost of false negatives in our setting.

Create roc plot plot based on Fold5 in CV

```
create_roc_plot_with_optimal(
    fold5_all_coords[model_to_plot], fold5_threshold[model_to_plot]
)
```



This ROC curve illustrates the trade-off between sensitivity and false positive rate for the model across all thresholds. More specifically, the highlighted point shows that at this point the model captures 89% of true positives, but at the cost of a 73% false positive rate. This may be acceptable in high-risk-tolerant settings, where capturing high-growth firms is more important than avoiding false alarms. This coincides with the assumption we have made.

CM on holdout

```
cm_object3 = confusion_matrix(data_holdout["HG_turnover"],
holdout_threshold, labels=[0, 1])
cm3 = pd.DataFrame(
    cm_object3,
    index=["Actual non-HG", "Actual HG"],
    columns=["Predicted non-HG", "Predicted HG"],
)
cm3
```

	Predicted non-HG	Predicted HG
Actual non-HG	839	1619
Actual HG	165	945

PREDICTION WITH RANDOM FOREST

```
y_train, rfvars_train = patsy.dmatrices("HG_turnover~" +
"+".join(rfvars), data_train)
y_holdout, rfvars_holdout = patsy.dmatrices("HG_turnover~" +
"+".join(rfvars), data_holdout)
```

Probability forest

Create CV summary table for Random Forest

```
prob_forest_cv_results
```

```
   max_features  min_samples_split    cv_auc   cv_rmse
0             5                 11  0.676994  0.438757
1             5                 16  0.676853  0.438557
2             6                 11  0.677191  0.438710
3             6                 16  0.675727  0.438826
4             7                 11  0.673880  0.439405
5             7                 16  0.675949  0.438886
```

```
summary
```

```
      Number of Coefficients   CV RMSE    CV AUC  CV treshold  \
M1                      16.0  0.452987  0.618541     0.200771
M2                      23.0  0.447289  0.651345     0.213057
M3                      40.0  0.442029  0.672425     0.194024
M4                      83.0  0.439566  0.676471     0.212004
M5                     197.0  0.440077  0.676408     0.193520
LASSO                  101.0  0.439519  0.677835     0.210698
RF                      n.a.  0.438710  0.677191     0.200581

       CV expected Loss
M1             0.672017
M2             0.663537
M3             0.645736
M4             0.642862
M5             0.640690
LASSO          0.640759
RF             0.642791
```

Gradient Boosting

Discussion of Results

Model Performance: RMSE, AUC, Threshold, and Expected Loss

While AUC values were broadly similar—indicating that all models ranked firms reasonably well—the expected loss was the most important criterion, as it directly captures the business cost of classification errors under the asymmetric loss function.

Logit M5 emerged as the most effective model overall, striking a strong balance between predictive performance and interpretability. Random Forest also performed well, especially in terms of probability accuracy, reflected in its low RMSE, but it lacked the interpretability of Logit models. Gradient Boosting had a comparable AUC but suffered from poorer RMSE performance, likely due to limited tuning, which affected its calibration.

The optimal classification thresholds for all models were notably lower than the default, reinforcing the idea that a lower cutoff is appropriate when the cost of false negatives outweighs that of false positives.

```
summary

      Number of Coefficients   CV RMSE    CV AUC  CV treshold  \
M1                      16.0  0.452987  0.618541     0.200771
M2                      23.0  0.447289  0.651345     0.213057
M3                      40.0  0.442029  0.672425     0.194024
M4                      83.0  0.439566  0.676471     0.212004
M5                     197.0  0.440077  0.676408     0.193520
LASSO                  101.0  0.439519  0.677835     0.210698
RF                       n.a.  0.438710  0.677191     0.200581
GBM                      NaN  0.528000  0.676000     0.192000

       CV expected Loss
M1             0.672017
M2             0.663537
M3             0.645736
M4             0.642862
M5             0.640690
LASSO          0.640759
RF             0.642791
GBM            0.644000
```

Confusion matrix

We use a confusion matrix on the holdout set to evaluate the Logit M5 model at the optimal threshold from Fold 5. This threshold minimizes the expected loss based on our custom investor-oriented loss function, which penalizes false negatives more heavily.

The confusion matrix clearly illustrates how the model balances false positives and false negatives, offering insight into real-world performance beyond metrics like AUC.

In this context, a false negative means failing to identify a firm that will grow rapidly — a costly mistake for an investor who seeks high-return opportunities. We assume that missing a high-growth firm is more harmful than backing one that does not grow. For this reason, our loss function places greater weight on false negatives.

```
cm_m5
```

	Predicted non-HG	Predicted HG
Actual non-HG	691	1767
Actual HG	134	976

Model Results and Usefulness

Based on the summary table and the confusion matrix, Logit M5 emerged as the best model overall, offering the lowest expected loss and a strong balance between prediction accuracy and interpretability.
Random Forest and Gradient Boosting also performed well, though RF showed slightly better results than GBM, likely due to limited hyperparameter tuning in the latter.
The chosen models are particularly useful in high-risk-tolerant environments, where missing a high-growth firm is costlier than mistakenly investing in a low-growth one.

Overall, the models provide valuable decision support for investors or policymakers aiming to identify high-growth firms. Their usefulness depends on the user's risk preferences. In our case, we prioritize capturing high-growth opportunities even at the cost of some false positives, acknowledging the trade-off between potential gains and occasional misclassification.

Part 2: Sector Comparison

To assess model performance across sectors, we applied it separately to manufacturing and services firms, using a consistent loss function and evaluation framework. Random Forest was selected as the final model for its strong predictive accuracy, robustness to overfitting, and relatively low computational demands during tuning.

```
data_train, data_holdout = train_test_split(data, train_size=0.8,
random_state=42)
```

```
Total
HG_turnover
0    0.685175
1    0.314825
Name: proportion, dtype: float64
Train
HG_turnover
0    0.685175
1    0.314825
```

```
Name: proportion, dtype: float64
Holdout
HG_turnover
0    0.685175
1    0.314825
Name: proportion, dtype: float64
```

```
y_train, rfvars_train = patsy.dmatrices("HG_turnover~" +
"+".join(rfvars), data_train)
y_holdout, rfvars_holdout = patsy.dmatrices("HG_turnover~" +
"+".join(rfvars), data_holdout)
```

The loss function

We follow the same intuition as in Part 1.

```
FP = 1
FN = 4
cost = FN / FP
```

Probability forest

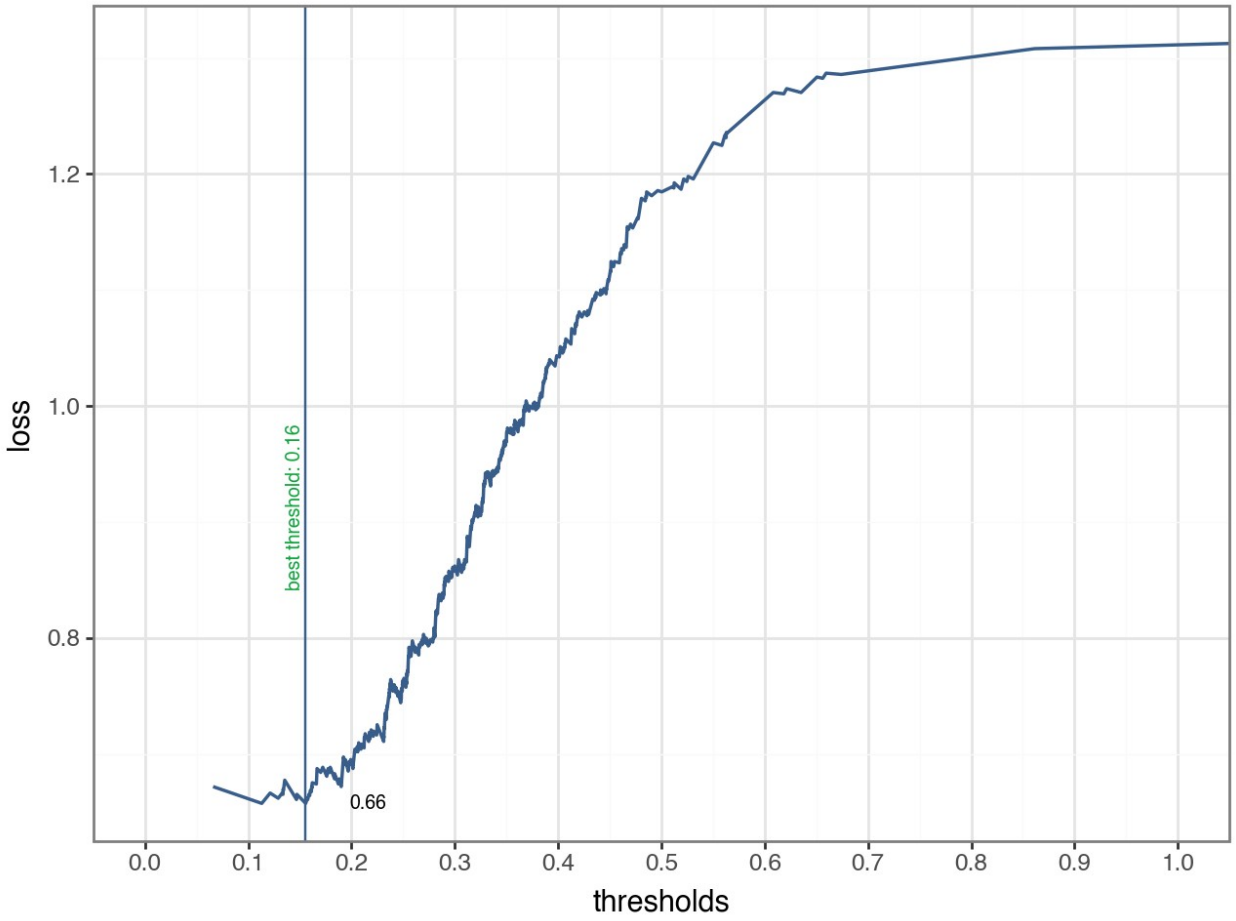
5 fold cross validation

```
prob_forest = RandomForestClassifier(random_state=42,
n_estimators=500, oob_score=True)
prob_forest_grid = GridSearchCV(
    prob_forest,
    grid,
    cv=k,
    refit="roc_auc",
    scoring=["roc_auc", "neg_brier_score"],
)

prob_forest_fit = prob_forest_grid.fit(rfvars_train, y_train)

prob_forest_cv_results
```

```
   max_features  min_samples_split    cv_auc   cv_rmse
0             5                 11  0.613108  0.455124
1             5                 16  0.616903  0.454470
2             6                 11  0.611186  0.455933
3             6                 16  0.614566  0.455164
4             7                 11  0.611617  0.455835
5             7                 16  0.612785  0.455536
```

Classification Threshold Optimization – Manufacturing Sector

```
create_loss_plot(all_coords_rf, fold5_threshold_rf,
fold5_expected_loss_rf)
```

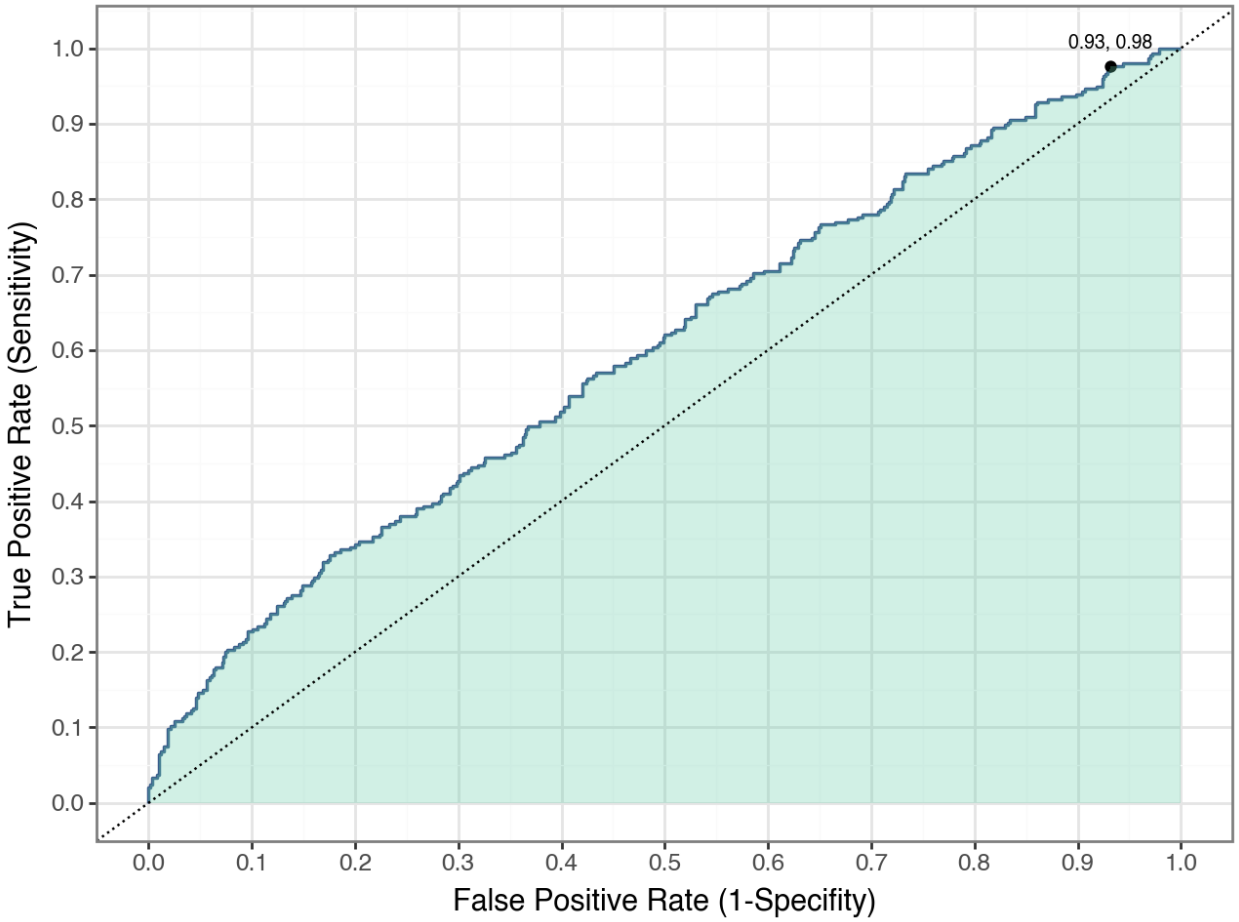


The plot above shows the relationship between classification thresholds and expected loss for the Random Forest model applied to the manufacturing subsample. The x-axis represents the probability threshold used to classify firms as high-growth, and the y-axis shows the corresponding expected loss based on our custom loss function (False Positive = 1, False Negative = 4).

The curve displays a shallow U-shape, with the lowest expected loss occurring around a threshold of 0.16 (highlighted by the vertical line). Beyond this point, the loss increases more steeply, particularly at higher thresholds. This indicates that misclassification costs grow rapidly when the model becomes too conservative. The selected threshold provides the best trade-off between capturing high-growth firms and minimizing false positives, in line with our investor-driven loss function where false negatives are more costly.

ROC Curve – Fold 5 (Random Forest, Manufacturing)

```
create_roc_plot_with_optimal(all_coords_rf, fold5_threshold_rf)
```



The ROC curve lies above the diagonal baseline, confirming that the model performs better than random guessing.

While the AUC is moderate, the steep initial slope indicates that the model correctly identifies a significant share of high-growth firms at low false positive rates. This is desirable in our context, where identifying high-growth firms (true positives) is economically more valuable than avoiding false positives. The marked point shows the selected optimal threshold for Fold 5, which minimizes expected loss under the investor-weighted loss function.

Final results

To compare the results of the RF modeling on `2012_hg_workfile_S.csv` with the previous results, we need to repeat the RF modeling steps for the new dataset (`2012_hg_workfile_S.csv`) and then compare the key metrics such as CV RMSE, CV AUC, thresholds, and expected loss.

Comparison of the results for 2 samples (manufacturing vs. services)

```
comparison_summary
```

	Metric	2012_hg_workfile_M	2012_hg_workfile_S
0	CV RMSE	0.454	0.433

```
1            CV AUC                0.617                0.700
2       Avg Threshold              0.176                0.204
3  Avg Expected Loss               0.658                0.622
```

The table above compares model performance between manufacturing (_M) and services (_S) sectors using the same prediction model (Random Forest) and consistent evaluation criteria.

The services sector shows better predictive performance, with a lower RMSE (0.433 vs. 0.454) and a higher AUC (0.700 vs. 0.617). This suggests that high-growth firms in services are more consistently identified by the general model, which was also run in Part 1.

The expected loss is also lower in services (0.622 vs. 0.658), indicating fewer costly misclassifications under our investor-weighted loss function.

The optimal threshold for classification is higher in services (0.204 vs. 0.176), reflecting different score distributions.

These results may partly reflect differences in sample sizes—the services group includes more than twice as many firms, which likely supports better model learning and generalization. Additionally, services firms tend to exhibit more stable growth patterns compared to the higher variability seen in manufacturing due to investment cycles and capital intensity.