

# Technical Report: Investment Classification & Risk Analysis

**Prepared by:** Ambrus Fodor, Katharina Burtscher  
**Date:** February 10, 2026  
**Subject:** Evaluation of Predictive Models for Capital Allocation Optimization  
**GitHub:** [Project Repository](#)

## 1. Executive Summary

This report details the development and validation of a machine learning framework designed to optimize the screening of high-potential investments. Our primary objective was to navigate the financial asymmetry between the risk of capital loss and the opportunity cost of missed growth.

**Key Results:**

- **Selected Model:** Random Forest Classifier
- **Optimal Decision Threshold:** 0.60
- **Financial Performance:** The model achieved a minimum average business cost of **\$63,559** on the holdout set, significantly outperforming Gradient Boosting and Logistic Regression baselines.
- **Risk Profile:** With a high **Specificity (98.04%)**, the model effectively filters out nearly all non-performing candidates while maintaining a **Precision of 76.57%** on its investment recommendations.

## 2. Problem Statement & Financial Constraints

### 2.1 Objective

The goal is to predict the likelihood of a company being "Fast Growing" (defined as achieving 30% growth between 2012 and 2013). We treat this as a binary classification task:

- **1 (Positive):** Fast-growing firm (Target).
- **0 (Negative):** Non-performing or average firm.

### 2.2 Asymmetric Cost Matrix

Unlike standard academic metrics, our optimization is driven by real-world financial penalties defined by the investment committee. A "Bad Investment" (losing principal) is considered **twice as damaging** as a "Missed Opportunity."

For a one-million-dollar investment, the loss function is structured as follows:

Prediction Error	Business Context	Cost (USD)
False Negative (FN)	Missed Opportunity: Rejecting a future high-performer.	\$300,000
False Positive (FP)	Bad Investment: Committing capital to a failing firm.	\$600,000

**Loss Function:**  $\text{Total Cost} = (\text{FN}/N) \times \$300,000 + (\text{FP}/N) \times \$600,000$

---

### 3. Data Source

The analysis utilizes a [dataset](#) containing 287,829 records of firm-level data from a mid-sized EU country. The data covers companies registered between 2005 and 2016 within three key industries: auto manufacturing, equipment manufacturing, and hospitality (hotels and restaurants).

#### 3.1 Sample Design

To align with our growth metric, we restricted the dataset to companies active in both 2012 and 2013. We also filtered for firms with annual sales between \$10,000 and \$10,000,000 to eliminate outliers and focus on our core target demographic.

```
# Get the sets of unique company IDs for each year
ids_2012 = set(data.loc[data["year"] == 2012, "comp_id"])
ids_2013 = set(data.loc[data["year"] == 2013, "comp_id"])

# Find the intersection (companies present in both years)
common_ids = ids_2012.intersection(ids_2013)

data = data[data["comp_id"].isin(common_ids)]
```

Variables with a high proportion of missing values were excluded to maintain data quality.

```
data = data.drop(
    columns=["COGS", "finished_prod", "net_dom_sales", "net_exp_sales",
"ages", "D", "exit_year", "exit_date"]
)
```

#### 3.2 Label Engineering

Our primary "Fast Growth" label was constructed using a 30% growth cutoff, determined through a combination of domain expertise and visual analysis of growth distribution.

**Decision Process:** We evaluated several metrics for growth and ultimately selected *Sales*. It provides a transparent, universal measure of traction; even venture-backed startups often prioritize sales growth before reaching profitability.

- **Functional Form:** We calculated the percentage growth in sales between 2012 and 2013. These years were chosen due to their high data density and availability.
- **Cut-off Selection:** While some definitions use 20% over three years, we applied a stricter 30% threshold for a single-year snapshot to capture truly high-velocity firms.

#### 3.3 Feature Engineering

We re-coded industry categories based on domain insights and introduced a quadratic term for firm age to account for non-linear life-cycle effects.

```
# Consolidating industry categories
data["ind2_cat"] = np.where(data["ind2"] > 56, 60, data["ind2"])
# Capturing non-linear age effects
data["age2"] = data["age"] ** 2
```

To handle illogical negative values in asset columns (physical and intangible), we flagged these errors and capped them at zero.

```
# Flagging problematic assets and flooring at zero
data["flag_asset_problem"] = np.where((data["intang_assets"] < 0) |
(data["curr_assets"] < 0), 1, 0)
data["intang_assets"] = np.where(data["intang_assets"] < 0, 0,
data["intang_assets"])
```

To normalize for firm size, absolute currency values were converted into financial ratios.

```
# Ratio generation for Balance Sheet items
data[[col + "_bs" for col in bs_names]] =
data[bs_names].div(data["total_assets_bs"], axis="index")
```

Financial ratios often contain extreme values; we used a "flag and cap" approach for variables that theoretically reside within specific bounds (e.g., 0 to 1).

```
# Flagging and capping values > 1 (e.g., for accounting ratios)
data[col + "_flag_high"] = np.where(data[zero].isna(), np.nan, (data[zero]
> 1).astype(int))
data[col] = np.where(data[zero] > 1, 1, data[zero])
```

Missing data was managed via a hybrid imputation strategy to preserve sample size without diluting signal quality.

```
# Mean imputation for CEO age
data["ceo_age"] = np.where(data["ceo_age"].isna(), data["ceo_age"].mean(),
data["ceo_age"])
# Final cleanup of critical missing values
data.dropna(subset=["liq_assets_bs", "foreign", "ind"], inplace=True)
```

---

## 4. Methodology & Implementation

We implemented a rigorous training pipeline to ensure model robustness and prevent common statistical pitfalls.

## 4.1 Preventing Data Leakage

A cornerstone of our architecture was the use of **Pipelines**. Preprocessing steps, specifically standardization (scaling), were fitted exclusively on the training folds within the Cross-Validation loop. This ensures that the test set remains strictly independent.

### Code Snippet: Pipeline Implementation

```
from sklearn.pipeline import Pipeline

# We use a Pipeline to scale data INSIDE the CV loop (prevents leakage)
lasso_rmse_pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('lasso', LogisticRegressionCV(
        Cs=Cs_values,
        penalty='l1',
        cv=k,
        scoring='neg_brier_score',
        solver='liblinear',
        random_state=42,
        n_jobs=-1
    ))
])
```

## 4.2 Hyperparameter Tuning

We utilized **GridSearchCV** to optimize the Random Forest and Gradient Boosting models. Our primary scoring metric was the Brier Score (mean squared error of probabilities), ensuring our models provide well-calibrated probability estimates rather than just binary labels.

### Code Snippet: Grid Search Configuration

```
from sklearn.model_selection import GridSearchCV

# Define RF Grid & Model
grid = {
    'max_features': [5, 6, 7, "sqrt"],
    'criterion': ['gini'],
    'min_samples_split': [11, 16],
    'n_estimators': [500]
}

prob_forest = RandomForestClassifier(
    random_state=42,
    n_estimators=100,
    oob_score=True
)
```

```
# Run Grid Search
prob_forest_grid = GridSearchCV(
    prob_forest,
    grid,
    cv=5,
    refit='neg_brier_score',
    scoring=['accuracy', 'roc_auc', 'neg_brier_score'],
    n_jobs=-1
)
```

4.3 Custom Business Cost Function

To finalize model selection, we moved beyond standard AUC metrics. We developed a custom loss function to evaluate models based on our specific \$300k / \$600k penalty structure.

Code Snippet: Financial Loss Calculation

```
def calculate_business_cost(y_true, y_probs, threshold):
    # Convert probabilities to binary predictions based on threshold
    y_pred = (y_probs >= threshold).astype(int)

    # Calculate False Positives and False Negatives
    fp = np.sum((y_pred == 1) & (y_true == 0))
    fn = np.sum((y_pred == 0) & (y_true == 1))

    # Apply Financial Costs
    cost_fn = 300_000
    cost_fp = 600_000

    total_loss = (fn * cost_fn) + (fp * cost_fp)
    return total_loss
```

5. Model Comparison & Selection

We benchmarked four candidate models using 5-fold cross-validation. While Gradient Boosting achieved the highest raw AUC, the Random Forest model generated the lowest expected financial loss when applied to our cost matrix.

Table 1: Model Performance Summary (Holdout Set)

Model	AUC	Classification Threshold	False Positives (Bad Inv)	False Negatives (Missed Opp)	Total Expected Loss
Logit X4	0.758	0.65	42	501	\$175,500,000
Lasso Logit	0.761	0.62	48	493	\$176,700,000

Model	AUC	Classification Threshold	False Positives (Bad Inv)	False Negatives (Missed Opp)	Total Expected Loss
Gradient Boosting	0.781	0.68	45	491	\$174,300,000
Random Forest (Selected)	0.775	0.60	41	493	\$172,500,000

**Selection Logic:** Although Gradient Boosting showed a marginal edge in AUC, the Random Forest model's probability distribution allowed for a more efficient threshold (0.60). By minimizing expensive False Positives (41 vs 45), it provides a **\$1.8 Million cost saving** over the next best alternative.

## 6. Final Model Evaluation: Random Forest

The Random Forest model was evaluated on the complete holdout dataset (\$N \approx 2,714\$).

### 6.1 Confusion Matrix Analysis

At the selected probability threshold of **0.60**:

-	Actual Success (1)	Actual Failure (0)
Predicted Investment (1)	134 (TP)	41 (FP)
Predicted Rejection (0)	493 (FN)	2,046 (TN)

- **True Positives (134):** Correctly identified high-growth firms.
- **False Positives (41):** "Value Traps"—investments that failed to perform.
- **True Negatives (2,046):** Successfully avoided poor performers.

#### Classification Metrics:

Metric	Value
Accuracy	80.32%
Precision	76.57%
Recall	21.37%
Specificity	98.04%

### 6.2 Financial Impact Breakdown

**Cost of Missed Opportunities (FN):**  $\$493 \text{ (Missed Deals)} \times \frac{\$300,000}{2,714} = \text{\textbf{\$54,495}}$   
*Analysis:* This represents the largest cost component. The model is intentionally conservative, prioritizing capital preservation over deal volume.

**Cost of Bad Investments (FP):**  $\$41 \text{ (Failed Investments)} \times \frac{\$600,000}{2,714} = \text{\textbf{\$9,064}}$   
*Analysis:* Direct capital loss is strictly minimized. Only ~16% of the total expected cost

stems from actual cash loss; the remainder is theoretical "lost profit."

**Total Business Cost (per applicant):**  $\frac{\$172,500,000}{2,714} = \textbf{\$63,559}$

### 6.3 Sector Analysis

A post-hoc analysis revealed significant performance variance across industries:

- **Manufacturing Sector:** The model struggled, yielding a Recall of only 11.05%.
- **Services Sector:** Performance was significantly stronger here, with a Recall of 25.56%.

*Recommendation:* The model is best suited for Service-sector candidates. Manufacturing firms may require a lower threshold or qualitative human intervention.

---

## 7. Conclusion & Recommendations

The Random Forest model serves as a "Capital Shield." By achieving 98.04% Specificity, it protects the firm's principal investment at the expense of potential deal volume.

**Action Plan:**

1. **Automated Filtering:** Deploy the model to automatically filter out applicants with a probability score below 0.60. This safely removes ~80% of the deal pipeline (mostly True Negatives).
2. **Strategic Due Diligence:** Focus investment analysts on the remaining ~20% of applicants. With a Precision of 76.6%, 3 out of every 4 of these firms are likely high-growth winners.
3. **Sector-Specific Tuning:** Exercise caution in Manufacturing; consider lowering the decision threshold to 0.50 for this sector to improve opportunity capture.

---

## 8. Environment Details

- **Python Version:** 3.12.4
- **Core Packages:** numpy, pandas, matplotlib, sklearn, patsy