data and code available at: https://github.com/icecodesred/Data-Analysis-3/tree/main/Assignment2

Technical Report: Predicting High-Growth Firms

1. Introduction and Target Definition The objective of this project is to identify “high-growth” firms for investment by predicting which companies fall into the top 25% fastest growing segment based on revenue growth. We started with the bisnode_firms_clean2 file—a panel dataset containing firm-level data from 2012 to 2014. The target variable, high_growth, was created by computing sales growth between 2012 and 2014 and flagging firms that rank in the top 25% as high-growth (True). To ensure our model uses only forward-looking data, observations from years other than 2012 were dropped. This design allows us to capture sustainable, multi-year growth rather than transient spikes that might occur in a single year.

From a corporate finance perspective, using a multi-year growth measure helps smooth out short-term volatility and better reflects a firm’s underlying performance and expansion capability. Investors typically assess the medium-term outlook of a firm (2–3 years) to gauge whether it has the financial stability, capital structure, and operational efficiency to sustain growth. Although alternative definitions could use one-year growth or even non-revenue metrics like asset or employee growth, revenue growth is more directly tied to the firm’s operating performance and profitability, making it the preferred measure for our analysis.

2. Data and Feature Engineering Data was created from the same file as the case study, except including years between 2012-2014 to calculate the target boolean variable: high growth or not. A total of 43 predictors were selected based on their relevance in capturing the financial, operational, and managerial aspects of the firm. The predictors include financial statement variables such as sales, profit_loss_year, and curr_assets, balance sheet quality indicators like balsheet_flag, and management/HR features such as ceo_age and female. The final predictor list is constructed as follows:

```
all_predictors = [
    "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", "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",
    "female", "ceo_age", "flag_high_ceo_age", "flag_low_ceo_age", "flag_miss_ceo_age",
    "ceo_count", "foreign_management", "age", "age2", "new"
]
```

3. Modeling Approach

Data Splitting and Preprocessing

The dataset was divided into an 80% training set and a 20% holdout set. The holdout set remains unseen during the model tuning and cross-validation phase to provide an unbiased evaluation of the final model performance. For Lasso Logistic Regression, the training data was scaled using StandardScaler, as this model benefits from features on

similar scales. In contrast, tree-based models like Random Forest and XGBoost operate well on the raw data. 3.2 Models and Cost-Sensitive Framework

Three models were evaluated:

```
Lasso Logistic Regression (with L1 penalty using scaled features)
Random Forest
XGBoost
```

Given our business context—where missing a high-growth firm (a false negative) is 2.5 times more costly than a false positive—we defined a custom loss function as follows: Loss=cost_fp×(False Positives)+cost_fn×(False Negatives)

Cross-Validation and Threshold Optimization A 5-fold stratified cross-validation was conducted on the training set to ensure that each fold maintained the class distribution of the target variable. For each fold, the following metrics were computed for each model:

```
RMSE on predicted probabilities,
ROC AUC,
Expected Loss (using the defined cost function),
Optimal Threshold (the threshold that minimized expected loss).
```

The cross-validation results were aggregated into tables for RMSE, AUC, expected loss, and optimal thresholds. Random Forest emerged as the best model with the lowest average expected loss (~2166) and competitive RMSE and AUC values.

```
for train_index, test_index in skf.split(X_train, y_train):
    fold_names.append(f'Fold {fold}')
    fold += 1

    for model_name, (model, X_model) in models.items():
        if isinstance(X_model, pd.DataFrame):
            X_fold_train = X_model.iloc[train_index]
            X_fold_test = X_model.iloc[test_index]
        else:
            X_fold_train = X_model[train_index]
            X_fold_test = X_model[test_index]

        y_fold_train, y_fold_test = y_train.iloc[train_index],
y_train.iloc[test_index]

        model.fit(X_fold_train, y_fold_train)

        y_prob = model.predict_proba(X_fold_test)[:, 1]

        rmse = np.sqrt(mean_squared_error(y_fold_test, y_prob))
        rmse_results[model_name].append(rmse)

        auc_val = roc_auc_score(y_fold_test, y_prob)
        auc_results[model_name].append(auc_val)

        losses = [expected_loss(y_fold_test, y_prob, t, cost_fp, cost_fn) for t in
```

```
thresholds]
        best_idx = np.argmin(losses)
        best_loss = losses[best_idx]
        best_thresh = thresholds[best_idx]

        loss_results[model_name].append(best_loss)
        threshold_results[model_name].append(best_thresh)
```

4. Final Evaluation on Holdout Set After selecting Random Forest as the optimal model, it was retrained on the full training set and then applied to the 20% holdout set. The holdout evaluation included generating a confusion matrix, classification report, and ROC curve. The holdout set evaluation confirmed similar performance to the cross-validation results. The confusion matrix revealed a trade-off that minimizes false negatives (the costlier error) even if it increases false positives, which aligns with our cost-sensitive framework.

Variable Importance A key benefit of using Random Forest is the ability to assess variable importance, which revealed that predictors such as sales, material_exp, and personnel_exp were among the most influential.

```
if isinstance(X_train, pd.DataFrame):
    feature_names = X_train.columns
else:
    feature_names = all_predictors
        threshold_results[model_name].append(best_thresh)

importance_df = pd.DataFrame({
    'feature': feature_names,
    'importance': importances
}).sort_values(by='importance', ascending=False)

filtered_importance_df = importance_df[importance_df['importance'] >= 0.02]
```

5. Discussion and Conclusions

Our analysis demonstrates that defining high growth based on a two-year revenue growth measure (2012–2014) effectively captures firms with sustained expansion, rather than transient spikes. This aligns with typical corporate finance approaches, where medium-term trends are more indicative of a firm’s underlying performance and potential.

Among the evaluated models, Random Forest performed best under our cost-sensitive framework, achieving an average expected loss of approximately 2166 across cross-validation folds. This cost function was tailored to penalize false negatives 2.5 times more than false positives, reflecting the high opportunity cost of missing a truly high-growth firm. The holdout set evaluation further confirmed Random Forest's predictive value, with an AUC of around 0.70 and a confusion matrix that favors capturing high-growth firms.

Final Recommendation: Use the Random Forest model (with an optimal threshold around 0.28) as the primary screening tool for identifying high-growth investment opportunities. Future work could include expanding the feature set and periodically retraining the model as new data becomes available.