Link to the Github repo.

Introduction

In this project we are going to predict corporate exit. We are going to predict the probability of firm exit and classify firms into prospective exiting firms and prospective staying in business firms. We are going to use `bisnode-firms` dataset. The data was collected and cleaned by Bisnode, a major European business information company.

Our study will focus on companies in the electronic product manufacturing industry. We will utilize data from before 2015 to train and test a model to predict whether companies remained in business in 2015.

We do not have an explicit y variable to indicate if a company exited the market. Instead, we will define an indicator variable where a value of 1 denotes companies with sales greater than 0 in 2014 but did not exist in 2015 or had zero or missing sales.

Data Cleaning and Feature Engineering

We imported libraries as pandas, numpy, sys, patsy, statsmodels, sklearn, plotnine, seaborn, matplotlib, py_helper_functions(it is locate in python file).

We have started with dropping the columns with more than 70% missing values, because it wouldn't be very informative and decisive to keep them. Subsequently, we created all possible combinations of firms and years, intending to create a new column named "status_alive." This column would indicate whether a company is still in operation based on non-missing and positive sales values.

```
# add all missing year and comp_id combinations -
# originally missing combinations will have NAs in all other columns
df = (
    df.set_index(["year", "comp_id"])
    .unstack(fill_value="toReplace")
    .stack()
    .reset_index()
)
df = df.replace("toReplace", np.nan)
```

To assess the financial indicators, we created flag variables to indicate potential issues. These flags identify negative values, missing data, and values that are too high or too low. In cases like sales, where the distribution has a long right tail, we also calculated logarithmic measures. To address the issue of negative values in the sales data, we replaced them with a value of 1 and created a corresponding flag variable. We generated new variables as sales ratios for profit and loss, and balance sheet indicators.

```
# default if there are sales in this year but no sales in the next year
df["default"] = (
    (df["status_alive"] == 1)
    & (df.groupby("comp_id")["status_alive"].shift(-1) == 0)
).astype(int)
```

For excessively high and low values, we employed the winsorization technique. For excessively low values, we established a threshold and substituted the same values for all excessively low values. We used the same technique in reverse for excessively high values. Additionally, we imputed missing values in numerical data with means and categorical data with mode. Flagged data that had been imputed.

```
# Variables that represent accounting items that cannot be negative (e.g. materials)
zero = [
    "extra_exp_pl",
    "extra_inc_pl",
    "inventories_pl",
    "material_exp_pl",
    "personnel_exp_pl",
    "curr_liab_bs",
    "fixed_assets_bs",
    "liq_assets_bs",
    "curr_assets_bs",
    "subscribed_cap_bs",
    "intang_assets_bs",
]

# Create flags for high values: 1 if > 1, NaN stays NaN
df[[col + "_flag_high" for col in zero]] = np.where(
    df[zero].isna(), np.nan, (df[zero] > 1).astype(int)
)

# Winsorize high values: cap at 1, preserving NaN
df[[col for col in zero]] = np.where(
    df[zero].isna(), np.nan, np.where(df[zero] > 1, 1, df[zero])
)

# Create flags for error values: 1 if < 0, NaN stays NaN
df[[col + "_flag_error" for col in zero]] = np.where(
    df[zero].isna(), np.nan, (df[zero] < 0).astype(int)
)

# Correct error values: set < 0 to 0, preserving NaN
df[[col for col in zero]] = np.where(
    df[zero].isna(), np.nan, np.where(df[zero] < 0, 0, df[zero])
)
```

Define Holdout and Work sets

To study firm default, we created a holdout dataset using specific filtering criteria. The dataset includes firms in the "Manufacture of computer, electronic, and optical products" industry (ind2 == 26) with 2014 sales figures between 1,000 EUR and 10 million EUR. This selection ensures that we focus on small or medium enterprises (SMEs) that existed in 2014, had positive sales, and did not exist in 2015 (either due to zero sales or missing data), meeting the definition of default. Our successful sample design resulted in a dataset of 1037 firms, of which 56 defaulted, and 981 remained active. The dataset shows an average sales figure of 0.4902 million EUR, with minimum and maximum sales recorded at 0.00107 million EUR and 9.57648 million EUR, respectively, adhering to the assignment's specified criteria.

```
# define work set
# we will use all data from 2014 except for the firms in the holdout set
work = df[(df['year'] == 2014) & (df['ind2'] != 26) & (df['sales'] >= 1000) & (df['sales'] <= 10000000)]
work.shape
```

```
(20253, 109)
```

For our work dataset, we selected all small and medium-sized enterprises (SMEs) from 2014 with sales ranging between 1000 EUR and 10 million EUR, excluding companies in the manufacturing electronic products field. Additionally, we constructed categorical variable matrices for various categories in the dataset. To establish reference categories for comparisons, we removed the first level of each category.

Before building models we defined our variable groups as it shown below:

```
# Define raw financial 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"]

# Define variables for quality assessment of balance sheets
qualityvars = ["balsheet_flag", "balsheet_length", "balsheet_notfullyear"]

# Define engineered variables from balance sheet and profit & loss statement for deeper analysis
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"]

# Define advanced engineered variables for more complex financial analysis
engvar2 = ["extra_profit_loss_pl_quad", "inc_bef_tax_pl_quad",
           "profit_loss_year_pl_quad", "share_eq_bs_quad"]

# Collect all flag variables dynamically based on their name patterns
flagvar = []
for col in df.columns:
    if col.endswith('flag_low') or col.endswith('flag_high') or col.endswith('flag_error') or col.endswith('flag_zero') or col.endswith('flag_missing'):
        flagvar.append(col)

# Define variables related to the first difference of log-transformed sales and their squared values along with flags for low and high values
d1 = ["d1_sales_mil_log_mod", "d1_sales_mil_log_mod_sq",
      "flag_low_d1_sales_mil_log", "flag_high_d1_sales_mil_log"]

# Define human resource-related variables including demographics and company management indicators
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"]
```

We created categorical variable matrices for different categories in the dataset and dropped the first level of each to establish reference categories for comparisons.

```
# we are dropping the first level of each categorical variable to have reference categories

ind2_catmat = patsy.dmatrix("0 + C(ind2)",dataset, return_type="dataframe")
ind2_catmat = ind2_catmat.drop(['C(ind2)[1.0]'], axis=1)

ind_catmat = patsy.dmatrix("0 + C(ind)",dataset, return_type="dataframe")
ind_catmat = ind_catmat.drop(['C(ind)[1.0]'], axis=1)

m_region_locmat = patsy.dmatrix("0 + C(region_m)",dataset, return_type="dataframe")
m_region_locmat = m_region_locmat.drop(['C(region_m)[Central]'], axis=1)

urban_mmat = patsy.dmatrix("0 + C(urban_m)",dataset, return_type="dataframe")
urban_mmat = urban_mmat.drop(['C(urban_m)[1.0]'], axis=1)

gender_catmat = patsy.dmatrix("0 + C(gender)",dataset, return_type="dataframe")
gender_catmat = gender_catmat.drop(['C(gender)[male]'], axis=1)

origin_catmat = patsy.dmatrix("0 + C(origin)",dataset, return_type="dataframe")
origin_catmat = origin_catmat.drop(['C(origin)[Domestic]'], axis=1)
```

Modeling

Using these groups of variables we built 4 logistic regression models.

- Model 1: only using rawvars
- Model 2: adding qualityvars and categorical variables
- Model 3: adding engvar1 and engvar2
- Model 4: adding d1, hr and flags

```
# model 1
raw = dataset[rawvars]
X1 = raw

# model 2
quality = dataset[qualityvars]
X2 = pd.concat([X1, quality, ind2_catmat, ind_catmat, m_region_locmat, urban_mmat, gender_catmat, origin_catmat], axis = 1)

# model 3
eng = dataset[engvar]
eng2 = dataset[engvar2]
X3 = pd.concat([X2, eng, eng2], axis = 1)

# model 4
flags = dataset[flagvar]
hrdata = dataset[hr]
d1data = dataset[d1]
X4 = pd.concat([X3, flags, hrdata, d1data], axis = 1)

# define vars for RF
rfvars = pd.concat([X3, flags, hrdata, d1data], axis = 1)

# define y
y = dataset['default']
```

In this code block below, we defined indices for training and holdout data based on predefined work and holdout datasets. Using these indices, we separated the target variable 'y' into corresponding training and holdout sets. We then established a cross-validation scheme using KFold with 5 splits, enabling shuffling and setting a specific random state for reproducibility. Additionally, we prepared for logistic regression without regularization by setting the regularization strength parameter (Cs) to a very high value, indicating no regularization is needed for our model training process.

```
# Define training and holdout split indices based on work and holdout_data setup
index_train = work.index
index_holdout = holdout.index

# Use the indices to split the target variable 'y' into training and holdout sets
y_train = y.loc[index_train]
y_holdout = y.loc[index_holdout]

# cross-validation setup
k = KFold(n_splits = 5, shuffle = True, random_state = 20240303)

# no regularisation needed so setting the paremeter to very high value
Cs_value_logit = [1e20]
```

Using our cross-validation settings we calculated RMSE for each model within each fold. We can see from the table below that Model 4 outperformed all other 3 models in all folds.

```
# Convert CV_RMSE_folds to a DataFrame
df_cv_rmse = pd.DataFrame(CV_RMSE_folds)

# Calculate the average of each column and append it as a new row
df_cv_rmse.loc['Average'] = df_cv_rmse.mean()

df_cv_rmse
```

	X1	X2	X3	X4
0	0.292228	0.290873	0.287043	0.281215
1	0.296953	0.296145	0.290290	0.285687
2	0.302767	0.301821	0.298020	0.291982
3	0.297870	0.297314	0.292492	0.290998
4	0.309067	0.308152	0.303956	0.299604
Average	0.299777	0.298861	0.294360	0.289897

We continued the same and got similar results for AUC.

```
# Convert AUC to a DataFrame
df_cv_auc = pd.DataFrame(CV_AUC_folds)

# Calculate the average of each column and append it as a new row
df_cv_auc.loc['Average'] = df_cv_auc.mean()

df_cv_auc
```

	X1	X2	X3	X4
0	0.760618	0.705920	0.760104	0.785092
1	0.717943	0.717291	0.729746	0.763348
2	0.730290	0.705731	0.738296	0.765110
3	0.735635	0.713524	0.748583	0.768290
4	0.712669	0.706188	0.730172	0.756346
Average	0.731431	0.709731	0.741380	0.767637

The results presented indicate the performance of four logistic regression models, each progressively incorporating more variables from different categories, evaluated using 5-fold cross-validation. Model 1 (X1), which uses only basic financial variables (rawvars), achieved an average Cross-Validation Root Mean Square Error (CV RMSE) of 0.299777 and a Cross-Validation Area Under the Curve (CV AUC) of 0.731431, serving as a baseline for comparison.

As additional variables are introduced in subsequent models—quality and categorical variables in Model 2 (X2), engineered variables in Model 3 (X3), and a combination of first differences, human resources-related variables, and flags in Model 4 (X4)—there is a noticeable improvement in both CV RMSE

	CV RMSE	CV AUC
X1	0.299777	0.731431
X2	0.298861	0.709731
X3	0.294360	0.741380
X4	0.289897	0.767637

and CV AUC. Notably, Model 4 shows the most significant enhancement, achieving the lowest RMSE (0.289897) and the highest AUC (0.767637), indicating its superior predictive accuracy and ability to distinguish between classes compared to the simpler models. This suggests that incorporating a broader range of variables and capturing more complexity in the data can lead to better model performance.

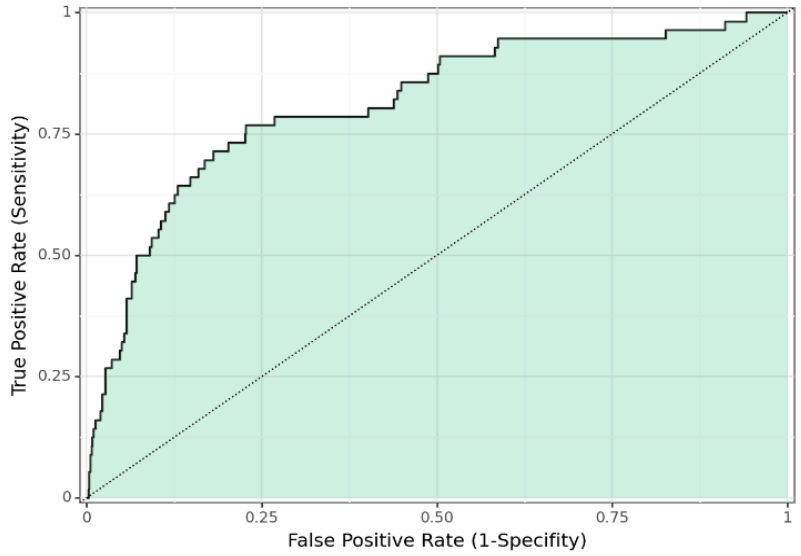
We continued with model 4 applying it on the holdout set:

```
# Select the best logistic regression model previously identified as 'X4'
best_model = logit_models['X4']

# Prepare the features from the holdout set that correspond to the best model's requirements
best_model_X_holdout = X4.loc[index_holdout]

# Predict the probabilities of the positive class for the holdout set using the best model
logit_predicted_probabilities_holdout = best_model.predict_proba(best_model_X_holdout)[:,1]

# Calculate the Root Mean Square Error (RMSE) between the predicted probabilities and the actual target values in the holdout set
best_rmse_holdout = np.sqrt(metrics.mean_squared_error(y_holdout, logit_predicted_probabilities_holdout))

# Round the RMSE to three decimal places for reporting
rounded_rmse_holdout = round(best_rmse_holdout, 3)
print('RMSE of holdout logistic model is: ',rounded_rmse_holdout)
```

```
0.214
```

Comparing this RMSE (0.214) on the holdout set with the Cross-Validation RMSE values obtained during model selection, we observe that the holdout RMSE is significantly lower than any of the CV RMSEs recorded for the models (X1 through X4). This improvement suggests that the model 'X4' not only performed well during cross-validation (with the lowest CV RMSE of 0.289897 and highest CV AUC of 0.767637) but also generalized effectively to unseen data, as evidenced by its performance on the holdout set. The results affirm the model's robustness and its potential for reliable predictions in practical applications.

AUC of holdout logistic model is: 0.813, which is also a good indicator of our chosen logistic model.
We continued with plotting ROC.



The ROC curve shows that the model is quite good, because it is well above 45 degree line. From confusion matrix we can see that the threshold is not the optimal one. Becaue all firms are classified as not default

Further to classify default firms we started to look for optimal thresholds

```
logit_summary2 = pd.DataFrame(best_thresholds_cv.items(),columns=['Model', 'Avg of optimal thresholds'])
logit_summary2['Threshold for Fold5'] = fold5_threshold.values()
logit_summary2['Avg expected loss'] = expected_loss_cv.values()
logit_summary2['Expected loss for Fold5'] = fold5_expected_loss.values()
logit_summary2
```

	Model	Avg of optimal thresholds	Threshold for Fold5	Avg expected loss	Expected loss for Fold5
0	X1	0.155653	0.157603	1.236561	1.329630
1	X2	0.122835	0.120193	1.295067	1.351852
2	X3	0.147568	0.155680	1.222786	1.302963
3	X4	0.147531	0.167311	1.147094	1.233333

The table presents the results of evaluating four logistic regression models (X1, X2, X3, X4) on their ability to predict outcomes, focusing on the determination of the optimal threshold for classification and the associated expected loss. The "Avg of optimal thresholds" column shows the average optimal threshold value for classifying observations across all folds, with X1 having the highest average threshold (0.155653) and X2 the lowest (0.122835). The "Threshold for Fold5" column reveals the optimal threshold specifically for the fifth fold of cross-validation, indicating slight variations from the overall average, which suggests model sensitivity to different data splits. "Avg expected loss" and "Expected loss for Fold5" columns quantify the average and fold-specific expected losses, respectively, with model X4 exhibiting the lowest average expected loss (1.147094) and, consequently, the lowest expected loss in the fifth fold (1.233333), highlighting its superior performance in minimizing cost-adjusted misclassifications. This suggests that model X4, despite having a higher threshold for the fifth fold compared to its overall average, manages to balance the trade-off between false positives and false negatives more effectively than the other models, underlining its potential as the most cost-efficient choice for prediction within this specific setup.

Following that we got our confusion matrix:

```
# Apply the optimal threshold to the holdout set to classify observations
holdout_treshold = np.where(logit_predicted_probabilities_holdout < best_logit_optimal_treshold, 0, 1)

# Calculate the confusion matrix components: true negatives (tn), false positives (fp), false negatives (fn), and true positives (tp)
tn, fp, fn, tp = confusion_matrix(y_holdout, holdout_treshold, labels=[0,1]).ravel()

# Compute the expected loss on the holdout set using the calculated tn, fp, fn, tp, and predefined FP and FN costs
expected_loss_holdout_log1 = (fp*FP + fn*FN)/len(y_holdout)

# print the expected loss
print('Exptected loss on holdout set is:',round(expected_loss_holdout_log1, 3))
```

```
Exptected loss on holdout set is: 0.613
```

```
# Create a DataFrame from the confusion matrix for better visualization and understanding of the model's performance
cm_object3 = confusion_matrix(y_holdout, holdout_treshold, labels=[0,1])
cm3 = pd.DataFrame(cm_object3,
    index=['Actul no defaul', 'Actual default'],
    columns=['Predicted no default', 'Predicted default'])
cm3
```

	Predicted no default	Predicted default
Actul no defaul	909	72
Actual default	28	28

The confusion matrix provides a detailed breakdown of the model's predictions against the actual outcomes for the holdout dataset, revealing its predictive performance. With 981 firms that stayed alive and 56 that defaulted in the actual data:

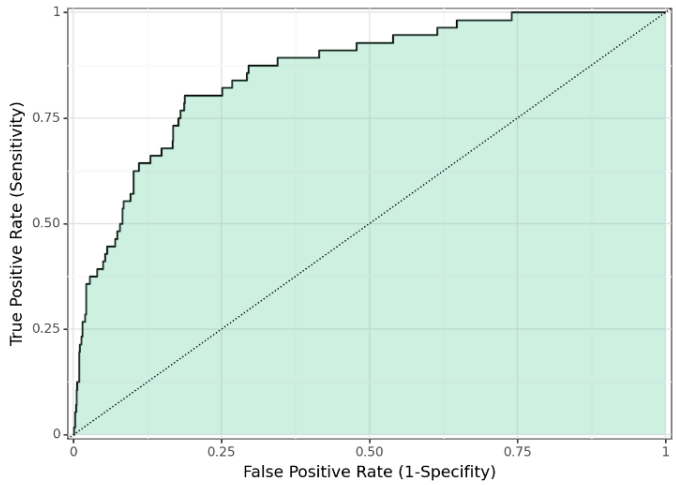
- True Negatives (TN): 909 firms were correctly predicted as "no default" (stayed alive), indicating the model's ability to identify firms that would continue operating.
- False Positives (FP): 72 firms were incorrectly predicted to default ("Predicted default") when they actually stayed alive. This represents Type I error, where the model mistakenly flags firms at risk.
- False Negatives (FN): 28 firms were incorrectly predicted as "no default" when they actually defaulted. This represents Type II error, indicating missed opportunities to identify at-risk firms.
- True Positives (TP): 28 firms were correctly predicted to default, showcasing the model's capability to identify actual defaults accurately.

Given the context, the model demonstrates a balanced ability to detect both defaults and non-defaults, with an equal number of true positives and false negatives. However, the presence of false positives and false negatives highlights the model's limitations in perfectly classifying firms. The goal in predictive modeling, especially in financial contexts like default prediction, is to maximize true positives (correctly identifying defaults) while minimizing false negatives (missing defaults) due to the typically higher cost associated with failing to identify a defaulting firm. The model's performance suggests room for improvement, particularly in reducing false negatives and false positives, to enhance its predictive accuracy and reliability in identifying default risk among firms.

We did the same process with Random Forest model:

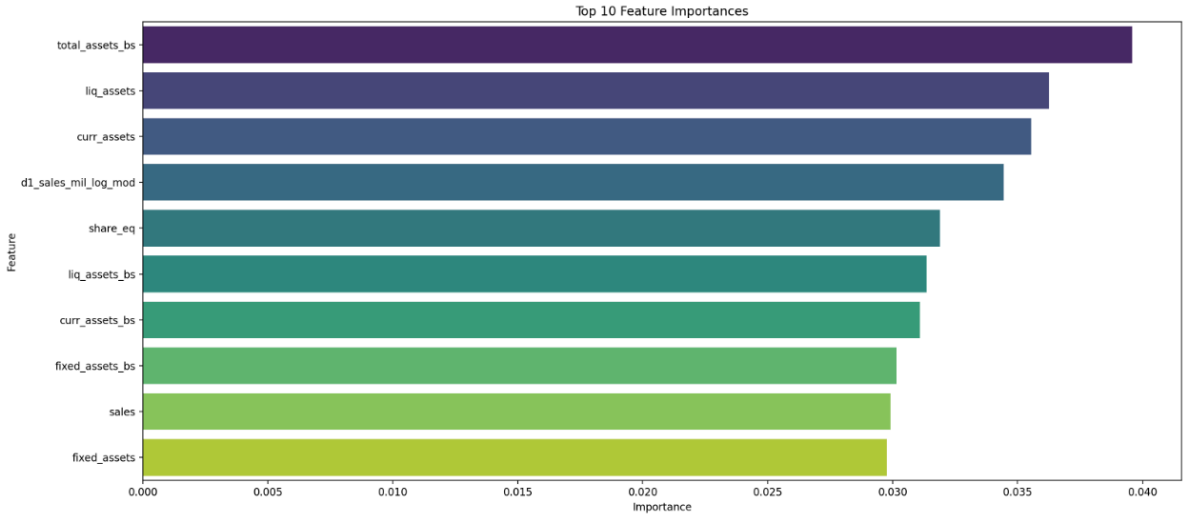
	Predicted no default	Predicted default
Actul no defaul	898	83
Actual default	26	30

The confusion matrix for the Random Forest model shows that out of the firms predicted to not default, 898 were correctly identified (true negatives), but 26 defaults were missed (false negatives). On the other hand, of those predicted to default, 30 were accurately predicted (true positives), while 83 were incorrectly classified as defaults (false positives). This indicates a relatively strong ability of the model to identify non-defaulting firms, albeit with some room for improvement in correctly identifying actual defaults and reducing false positives.



Visually we can confirm that AUC value is also higher in RF than in logistic regression model(0.86 vs 0.813)

Then we built variable importance plot



The importance score quantifies how valuable each variable is for predicting the target outcome in the model.

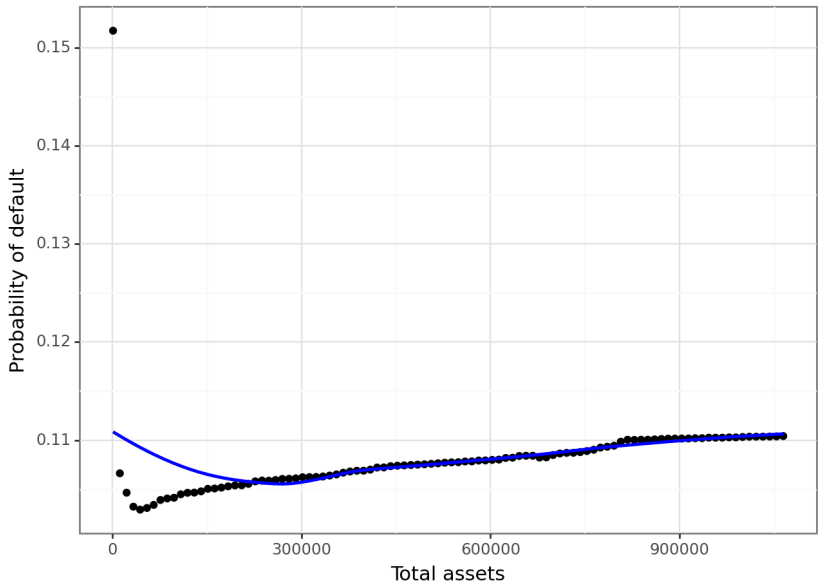
Balance Sheet and Liquidity Variables: The most significant variables include `total_assets_bs`, `liq_assets`, and `curr_assets`, indicating that overall asset volume and liquidity measures are critical for prediction. This suggests the model heavily relies on the firm's balance sheet strength and liquidity status to make accurate predictions.

Sales and Adjusted Sales Metrics: `d1_sales_mil_log_mod` and `sales` also appear as crucial predictors, underscoring the importance of a firm's revenue and its growth or decline rate in the model's decision-making process.

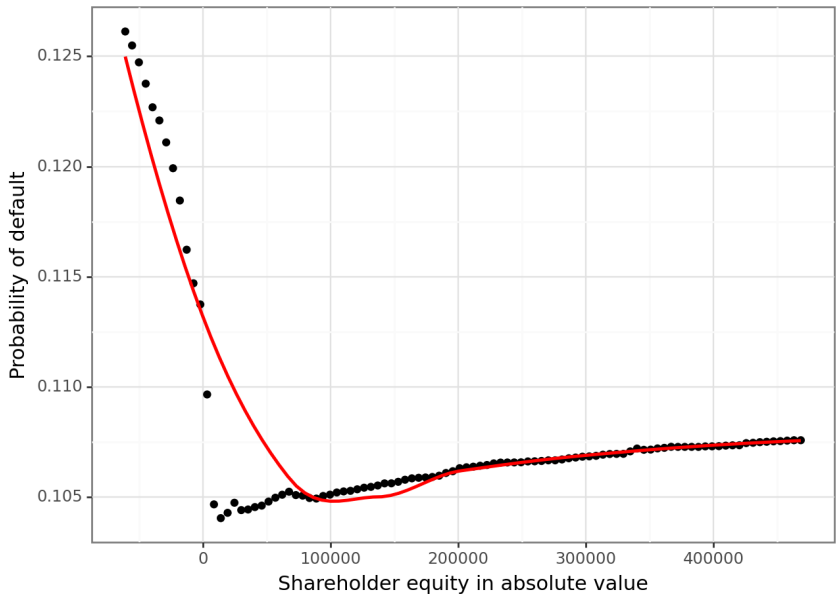
Equity and Fixed Assets: `share_eq` (shareholders' equity) and variables related to fixed assets (`fixed_assets_bs` and `fixed_assets`) are also highlighted as important, suggesting that the model considers the firm's equity base and the value of its long-term assets significant in predicting the outcome.

We built 2 partial dependence plots.

From the plot, we observe that as total assets increase from 0, there is an initial sharp decline in the probability of default. This suggests that firms with very few assets are much more likely to default. However, as assets continue to increase, the probability of default begins to level out. The trend becomes relatively flat for higher asset values, indicating that increases in total assets beyond a certain point have a negligible impact on reducing the probability of default. This could imply that once a firm has reached a threshold of asset accumulation, other factors may play a more significant role in the default risk, or that the firm's asset size is sufficient to not significantly affect the default risk further.



From the plot, we can observe that as shareholder equity increases from zero, the probability of default sharply decreases initially, suggesting that firms with very low equity are at a higher risk of default. However, beyond a certain point, increases in shareholder equity have a diminishing effect on reducing default probability. For higher values of shareholder equity, the probability of default seems to level off, indicating that additional equity above this point does not significantly alter the default risk as assessed by the model. This plateau effect suggests that other factors might become more influential in predicting default risk once a firm has an adequate equity buffer.



Conclusion

	Logistic regression	Random forest
Accuracy	0.904	0.895
Sensitivity	0.500	0.536
Specificity	0.927	0.915
Precision	0.280	0.265
AUC	0.813	0.860
Expected loss	0.613	0.616
RMSE	0.214	0.209
Optimal threshold	0.148	0.192
Brier score	0.176	0.165

Comparing the performance metrics of Logistic Regression and Random Forest models:

- Accuracy: Both models show high accuracy, with logistic regression slightly outperforming the random forest.
- Sensitivity (Recall): The random forest has a higher sensitivity, meaning it is better at correctly identifying positive cases.
- Specificity: Logistic regression exhibits higher specificity, indicating it is more adept at correctly identifying negative cases.
- Precision: Logistic regression has a marginally higher precision than the random forest, suggesting it has fewer false positives relative to true positives.
- AUC: The Area Under the Curve is higher for the random forest, suggesting it is better at distinguishing between the classes across all thresholds.
- Expected Loss: The expected loss, which factors in the cost of false negatives and positives, is very similar between the two models, despite the higher cost associated with false negatives.
- RMSE: The Root Mean Square Error is slightly lower for the random forest, indicating better performance in terms of predicted probability accuracy.
- Optimal Threshold: The optimal threshold for classification is higher for the random forest, which may be contributing to its lower specificity.
- Brier Score: The Brier score, which measures the accuracy of probabilistic predictions, is lower for the random forest, indicating better performance in this aspect.

Between Logistic Regression and Random Forest, the latter emerges as the better model when considering the balance between all the metrics provided, particularly in the context where false negatives carry a higher cost than false positives. The Random Forest model exhibits a higher Sensitivity, which is crucial in minimizing false negatives—this means it is more effective at identifying cases that are truly defaulting, a vital characteristic in financial risk assessment where the consequences of missing a default can be very costly. Additionally, the Random Forest has a higher AUC, indicating a stronger ability to differentiate between defaulting and non-defaulting cases over various threshold settings, a key advantage in creating a more robust and discriminative model.

Furthermore, despite a slightly higher Expected Loss and lower Precision, the Random Forest model has a better Brier score, which shows it provides more accurate probability estimates, and a lower RMSE, suggesting its predictions are closer to the actual outcomes. In practice, the Random Forest model's strengths in Sensitivity, AUC, Brier score, and RMSE would likely translate to better performance in real-world default prediction scenarios, where accurately identifying default risk and estimating the

probabilities of default are of great importance. Hence, the Random Forest model is preferred for its comprehensive performance and suitability for complex risk prediction tasks.

However in this particular case, when the goal is to get the model with the smallest expected loss the logistic regression could be a better choice!