Technical Report of Default Prediction Project

Student: Mai Pham
Github link: https://github.com/thimaipham/DA3_Assignment3

This report summarizes the development and evaluation of predictive models to identify potential defaults among SMEs in the 'Manufacture of computer, electronic and optical products' industry and how I handle with the data, model and prediction.

The data includes 2 parts:

- A portion is kept as a hold-out sample filtered according to the following conditions: belongs to the industry 'Manufacture of computer, electronic and optical products' industry, existed in 2014, but did not exist in 2015, sales in 2014 was between 1000 EUR and 10 million EUR
- The remaining Design data sample for training and testing. Filter Data for Years Before 2014: Select data from 2013 and earlier. This ensures that we do not use any information from the hold-out sample for training the model. The industry focuses on ind2 == 26, similar to what we did with the hold-out sample. Just like with the hold-out sample, we will identify SMEs based on revenue in 2013.

1. Data Exploration & Data Featuring

1.1. Constructing hold-out sample:

The methodology for constructing the hold-out sample comprised several key steps:

- Initial Data Filtering: I narrowed our dataset to include only firms within the specified industry for the year 2014
- SMEs Identification: From this filtered set, I identified SMEs with annual sales between 1,000 EUR and 10 million EUR in 2014,
- Analysis of Continuity and Financial Stability: I then identified which SMEs from 2014 were no longer present in 2015 or reported zero sales, indicating a cessation of operations or inactivity.
- Identification of Defaulted Firms
- Summary Statistics: This step compares my results with the expected result given in the assignment

1.2. **Design data sample for training and testing**

- Filter Data for Years Before 2014: Select data from 2013 and earlier. This ensures that we do not use any information from the hold-out sample for training the model.
- Select Industry: Focus on industry ind2 == 26, similar to what we did with the hold-out sample.
- Identifying SMEs: Just like with the hold-out sample, I will identify SMEs based on revenue in 2013.

1.3. Data Featuring and Missing values

In the technical report on our project analyzing the resilience and financial health of SMEs in the manufacturing sector, the feature engineering process plays a crucial role in enhancing the dataset for more insightful analysis. This section of the report explains the rationale behind the selection of specific features and the approach taken to handle missing values and zeroes in the dataset.

Selection of Features

- **Firm Age:** This feature is derived from the founded_date column by calculating the difference between the year 2013 and the firm's founding year (due to use this feature on training and testing data samples). Older firms might have more established customer bases and financial stability, making this feature valuable for assessing a firm's potential for longevity and success.
- **Current Ratio, Net Profit Margin, and Debt to Equity Ratio:**
    - The Current Ratio assesses a firm's ability to pay off its short-term liabilities with its short-term assets, indicating liquidity.
    - The Net Profit Margin reveals how effectively a firm converts revenue into actual profit, indicating profitability.
    - The Debt to Equity Ratio measures a firm's financial leverage, indicating the proportion of debt used to finance its assets relative to equity.

Handling Missing Values and Zeroes

The handling of missing values and zeroes is particularly important in financial data analysis, as zeroes can significantly impact the calculation of ratios, leading to distorted conclusions.

- Conversion to NaN for Financial Features: For columns like curr_assets, curr_liab, profit_loss_year, and share_eq, zeroes are replaced with NaN. This approach is taken because a zero value in these fields often does not indicate the absence of such assets or liabilities but rather missing or unreported data. By converting zeroes to NaN, we avoid skewing the calculations of financial ratios.
- Indicator Variables for Zero Values: Before converting zeroes to NaN, indicator variables are created for the aforementioned columns to flag original zero values. This step ensures that the analysis can still account for the presence of zero values in the original data, which might be relevant for certain types of financial analysis or modeling.

- Avoiding Division by Zero in Ratio Calculations: By replacing zero values with NaN in the denominator of ratios, we prevent division by zero errors.

```python
sme_pre_2014_filled = sme_pre_2014.copy()
# Change type of `founded_date` column to datetime
sme_pre_2014_filled['founded_date'] = pd.to_datetime(sme_pre_2014_filled['founded_date'], errors='coerce')

# Calculate `Firm Age` features
sme_pre_2014_filled['Firm Age'] = 2013 - sme_pre_2014_filled['founded_date'].dt.year

# Create indicator variables for columns with value 0
columns_with_zeros = ['curr_assets', 'curr_liab', 'profit_loss_year', 'share_eq']
for col in columns_with_zeros:
    flag_col = col + '_flag'
    sme_pre_2014_filled[flag_col] = (sme_pre_2014_filled[col] == 0).astype(int)
    sme_pre_2014_filled[col].replace(0, np.nan, inplace=True)

# Calculate new features, using columns that have been edited to avoid dividing by zero
sme_pre_2014_filled['Current Ratio'] = sme_pre_2014_filled['curr_assets'] / sme_pre_2014_filled['curr_liab']
sme_pre_2014_filled['Net Profit Margin'] = sme_pre_2014_filled['profit_loss_year'] / sme_pre_2014_filled['sales']
sme_pre_2014_filled['Debt to Equity Ratio'] = sme_pre_2014_filled['curr_liab'] / sme_pre_2014_filled['share_eq']

# Show data after adding new features
sme_pre_2014_filled[['Current Ratio', 'Net Profit Margin', 'Debt to Equity Ratio', 'Firm Age']].head()

```

2. Modelling:

The preparation for training predictive models involves labeling, feature selection, and data splitting:

- **Labeling:** A 'Default' column is added to mark businesses as defaulted (1) or non-defaulted (0) based on predefined identifiers. This step creates the target variable for the prediction model.

```python
# Generate output labels based on default information
# Mark defaulted businesses as 1 and non-defaulted businesses as 0
# Create a new column 'Default' in the data, default value is 0 (no default)
sme_pre_2014_filled['Default'] = 0

# Flag defaulted firms based on all_defaulted_firms_ids
sme_pre_2014_filled.loc[sme_pre_2014_filled['comp_id'].isin(all_defaulted_firms_ids), 'Default'] = 1

# Check the corporate default rate in the data
default_rate = sme_pre_2014_filled['Default'].mean()

# Perform data division
# Select input characteristics, remove 'comp_id', 'founded_date', and irrelevant columns or target data
X_columns = ['Current Ratio', 'Net Profit Margin', 'Debt to Equity Ratio', 'Firm Age']
Y_column = 'Default'

X = sme_pre_2014_filled[X_columns].fillna(0)  # Handle missing values by replacing them with 0
Y = sme_pre_2014_filled[Y_column]

# Divide data into training set and test set with ratio 80:20
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Check the size of the training set and test set
(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape, default_rate)

```

- **Default Rate:** The default rate is calculated to understand the dataset's balance between defaulted and non-defaulted firms, guiding model evaluation.
- **Feature Selection and Preprocessing:** Key financial ratios and firm age are selected as predictive features. Missing values in these features are filled with 0 to ensure model compatibility.
- **Data Splitting:** The dataset is divided into training and test sets with an 80:20 ratio, facilitating model training and subsequent evaluation on unseen data.

2.1. Logistic regression

As a requirement from the assignment, I started with the Logistic regression model.

- **High Accuracy but Low AUC**: The logistic regression model demonstrated high accuracy levels on both the training set (96.58%) and the test set (96.65%). However,

the Area Under the Curve (AUC) scores were significantly lower, with 0.525 on the training set and 0.475 on the test set. This discrepancy indicates that while the model is accurate in predicting the majority class (non-defaulting firms), it struggles to distinguish effectively between the defaulting and non-defaulting firms.

- **Impact of Class Imbalance**: The observed high accuracy alongside low AUC can largely be attributed to the class imbalance present in our dataset, where only about 3.41% of the firms defaulted. In such scenarios, accuracy can be misleading as a performance metric, since simply predicting the majority class for all observations would also yield high accuracy but poor model utility.

2.2. Random Forest model

Based on the current situation and the results from the logistic regression model, a reasonable option is to experiment with a more complex model to see if it can improve the prediction performance, especially the AUC, in the context of imbalanced data. The Random Forest model is a good choice because it is not only powerful in handling imbalanced data but also can handle features with non-linear relationships and complex interactions between features well. display.

Why Choose Random Forest?

- **Ability to Handle Imbalanced Data**: Random Forest can improve performance on imbalanced data through its bagging mechanism and underlying decision trees.
- **Nonlinear Feature Processing and Interaction**: This model is capable of automatically detecting and using nonlinear relationships and interactions between features without the need for complex feature engineering.
- **Generalization Ability**: Random Forest generally shows good generalization ability and has a lower risk of overfitting than single decision tree models due to its ensemble mechanism.

The results:

- **Significant Improvement in AUC**: The Random Forest model showed a significant improvement in the Area Under the Curve (AUC) on the test set, increasing from 0.475 (as observed with the logistic regression model) to 0.7075. This indicates a much better ability to distinguish between defaulting and non-defaulting firms.
- **High Accuracy Maintained**: Despite the complexities introduced by the Random Forest model, it maintained a high level of accuracy on the test set (96.75%), similar to the logistic regression model. This suggests that the model is robust in predicting outcomes even with the imbalanced nature of the dataset.
- **Exceptional Training Performance**: The model achieved near-perfect performance on the training set, with an accuracy of 99.94% and an AUC of 1.0. While this showcases the model's capability to fit the training data closely, it also prompts consideration for potential overfitting.

2.3. Gradient Boosting Model

- **High Accuracy on Training and Test Sets**: The Gradient Boosting model achieved an accuracy of 96.93% on the training set and 96.49% on the test set. These results

suggest that the model is able to predict with high reliability whether a firm will default or not, considering the entire dataset.

- **AUC Performance**: The model's AUC on the training set is 0.870, indicating a strong capability to distinguish between the defaulting and non-defaulting firms during training. The AUC on the test set is 0.699, which, while lower than the training AUC, still represents a reasonable ability to generalize and differentiate between the two outcomes in unseen data.
- **Comparison with Other Models**: When compared to the logistic regression and Random Forest models previously trained, the Gradient Boosting model presents a competitive AUC on the test set, signifying its potential as a robust predictive model for firm defaults.
- **Consideration of Overfitting**: There is a noticeable difference between the AUC scores on the training and test sets, which may suggest some overfitting. This could warrant further investigation and potential model tuning to ensure that the model generalizes well to new data.

	Model	Train Accuracy	Test Accuracy	Train AUC	Test AUC	Precision	Recall	F1 Score
0	Logistic Regression	0.9658	0.9665	0.5250	0.4751	0.0000	0.0000	0.0000
1	Random Forest	0.9994	0.9675	1.0000	0.7075	0.6667	0.0312	0.0597
2	Gradient Boosting	0.9693	0.9649	0.8703	0.6985	0.0000	0.0000	0.0000

INCONCLUSION for this part: In this specific context, even though Random Forest is overfitting, it still appears to be the best choice based on its ability to differentiate between classes and some success in detecting default cases.

3. Applying the best-performed training model to hold-out sample

This section describes applying a trained model to a hold-out sample to evaluate its performance in predicting business defaults. The process involves preparing the hold-out data, making predictions, and comparing these predictions to the actual outcomes. Here's a concise explanation:

- **Preparation of Hold-Out Sample:** The sme_2014 dataset is copied to X_holdout and processed to align with the model's input requirements. This includes calculating 'Firm Age' as of 2014, handling zero values by replacing them with NaN (and subsequently filling these NaNs with 0 for model compatibility), and creating financial ratio features ('Current Ratio', 'Net Profit Margin', 'Debt to Equity Ratio') that the model uses for predictions.
- **Feature Finalization:** Indicator variables are recalculated post-zero-value processing to ensure consistency with the model's training data. The final set of features is then selected for the model.
- **Labeling the Hold-Out Sample:** A new column, defaulted, is added to sme_2014 to indicate the actual default status based on all_defaulted_firms_ids, with defaulted businesses marked as 1 and non-defaulted as 0. This creates the y_true array, representing the true default statuses in the hold-out sample.

```python
X_holdout = sme_2014.copy()

# Calculate 'Firm Age' cho until 2014
X_holdout['founded_date'] = pd.to_datetime(X_holdout['founded_date'], errors='coerce')
X_holdout['Firm Age'] = 2014 - X_holdout['founded_date'].dt.year

# Handle with 0 values and create feature corresponding to with the model
X_holdout['Current Ratio'] = X_holdout['curr_assets'].replace(0, np.nan) / X_holdout['curr_liab'].replace(0, np.nan)
X_holdout['Net Profit Margin'] = X_holdout['profit_loss_year'].replace(0, np.nan) / X_holdout['sales'].replace(0, np.nan)
X_holdout['Debt to Equity Ratio'] = X_holdout['curr_liab'].replace(0, np.nan) / X_holdout['share_eq'].replace(0, np.nan)

# Fill in the missing value after replacing 0 with NaN
X_holdout.fillna(0, inplace=True)

# Make sure that indicator variables are calculated after the value 0 has been processed
for col in columns_with_zeros:
    flag_col = col + '_flag'
    X_holdout[flag_col] = (X_holdout[col] == 0).astype(int)

# Select the features prepared for the model
features_final = ['Current Ratio', 'Net Profit Margin', 'Debt to Equity Ratio', 'Firm Age']
X_holdout_final = X_holdout[features_final]

```

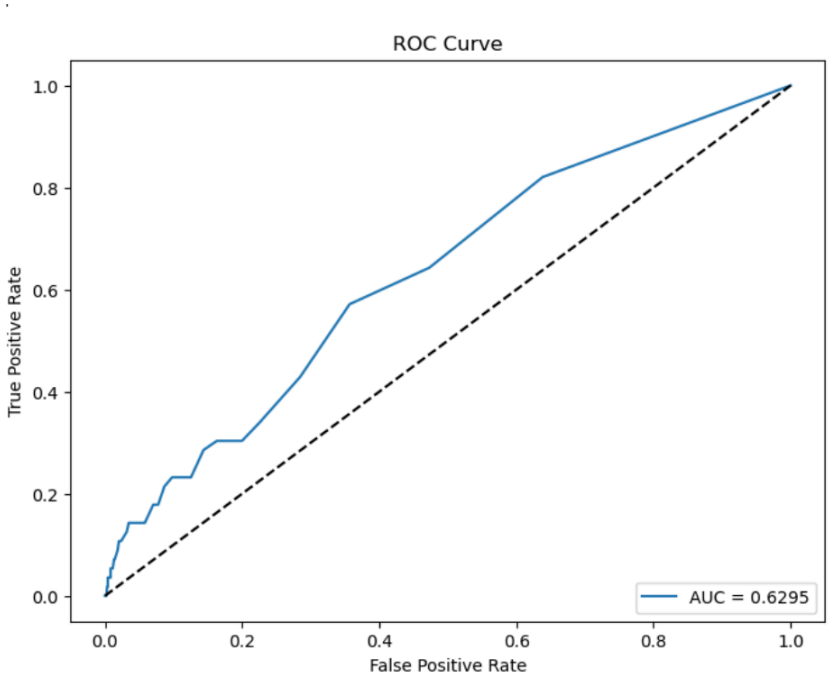
- **Making Predictions:** The trained random_forest_model is used to predict the default status (y_pred_holdout) for each company in the hold-out sample. Additionally, probabilities of default (y_proba_holdout) are calculated, which are particularly useful for evaluating the model's performance through metrics like the Area Under the Receiver Operating Characteristic curve (AUC).

```python
# Start by creating a new column in sme_2014 to store the default status
sme_2014['defaulted'] = 0  # Initialize with zeros

# Mark the companies that have defaulted based on your logic
sme_2014.loc[sme_2014['comp_id'].isin(all_defaulted_firms_ids), 'defaulted'] = 1

# Now that you have a 'defaulted' column indicating the default status, you can create y_true
y_true = sme_2014['defaulted'].values

# Predict the class for the hold-out pattern
y_pred_holdout = random_forest_model.predict(X_holdout_final)

# Probability prediction for hold-out samples (mainly used for calculating AUC)
y_proba_holdout = random_forest_model.predict_proba(X_holdout_final)[:, 1]
```

4. Result Discussion:

- **AUC:** The AUC provided by the Random Forest model on the test set (0.7075) suggests that the model has a moderate ability to distinguish between the default and non-default classes.
- **Brier Score:** The Brier score of 0.0524 on the hold-out sample is relatively low, suggesting that the model's predicted probabilities are, on average, not far from the true outcomes. However, given the imbalanced nature of the dataset, Brier score may not fully capture the model's performance in distinguishing between the two classes.
- **Accuracy (63.93%):** At the optimal threshold, the accuracy is significantly lower than the default threshold. This drop suggests that while attempting to capture more true positives (defaults), the model also increases the number of false positives, thus reducing overall accuracy.
- **Precision (8.38%):** The precision is considerably low, indicating that of all the cases the model predicts as default, only a small fraction are actually default. This low precision can be problematic, as it may lead to unnecessary actions being taken against firms incorrectly identified as likely to default.

- **Recall/Sensitivity (57.14%):** A recall rate of over 57% is a marked improvement and indicates that the model can now correctly identify more than half of the actual default cases. In the context of default prediction, a higher recall is usually more desirable than high precision, as failing to detect a default can have significant consequences.
- **Specificity (64.32%):** The specificity at the optimal threshold suggests that about 64% of non-default cases are correctly identified. However, this also means that around 36% of non-defaulting firms are being incorrectly flagged as defaulting, which could potentially lead to unwarranted credit decisions.
- **Expected Loss:** The increase in expected loss from 0.8158 to 1.3597 upon adjusting the threshold indicates a higher cost associated with misclassification, underscoring the importance of choosing an appropriate threshold that balances the cost of false positives and false negatives.
- **Area Under the Curve (AUC):** The AUC value is 0.6295, which is average, indicating that the model is able to distinguish between default and non-default classes slightly better than random prediction.
- **Classification Performance:** The ROC curve is above the dashed diagonal line, which shows that the model has better classification performance than random prediction. However, this performance is not great, because the curve is not close to the top left corner of the graph, where a perfect model would lie.



Given the current model's limitations in accurately predicting defaults, we recommend further exploration of feature selection, data rebalancing techniques, and advanced machine learning algorithms that might offer improved predictive power for imbalanced datasets.

V. Conclusion

Although the Random Forest model has shown potential, the results on the hold-out sample indicate that significant improvements are needed, particularly in correctly classifying default cases. The imbalance in the dataset and the dependence on threshold selection suggest a need for continued optimization of the model and evaluation methods.