Polish Bankruptcy Prediction Through Machine Learning

Project members: Yaşar Düzgün (39599212730), Mert Koçoğlu (14533043564)

Problem Statement

The goal of this project is to predict the bankruptcy of Polish companies based on their financial data. Financial statements from businesses from 2000 to 2012 (for bankrupt firms) and 2007 to 2013 (for non-bankrupt enterprises) make up the dataset. Our goal is to develop a prediction model that evaluates the likelihood of bankruptcy over a range of forecasting periods using these financial rates. Given the challenges inherent in predicting bankruptcy, the task is to classify companies into two categories:

- **Bankrupt**: Companies that went bankrupt within the given time frame.
- **Non-bankrupt**: Companies that continued to operate without bankruptcy within the given time frame.

Our goal is to utilize the dataset to create a model that, given the financial information from a particular year, can forecast the probability of a firm going bankrupt in subsequent years.

Dataset Overview

- Brief Description of the Dataset: The dataset aims to predict the insolvency of Polish firms based on their financial data. It includes financial metrics from companies across several years mainly 2000-2012 for bankrupt firms and 2007-2013 for non-bankrupt ones. The goal is to forecast which firms are likely to go bankrupt within one to five years based on their financial condition in a given year.
- Number of Instances (Rows): 1st Year 7027 instances, 2nd Year 10173 instances, 3rd Year 10503 instances, 4th Year 9792 instances, 5th Year 5910 instances.
- Number of Features (Columns): 64
- Types of Features: All features in the dataset are Numerical
- Target Variable (if applicable): The target variable is the bankruptcy status of the company
- Duplicate Entries: There are no duplicated entries

Data Processing

We used **Stratified K-Fold Cross-Validation** to evaluate our models. Instead of setting aside a fixed test set, we used stratified 5-fold cross-validation to ensure that each fold maintains the same proportion of bankrupt and non-bankrupt companies. This is important because the dataset is imbalanced, and stratification helps ensure that each fold is representative of the overall class distribution. Stratified splitting leads to more stable and reliable performance metrics, especially when dealing with imbalanced classification problems like bankruptcy prediction.

Missing values were handled using **mean imputation**. Specifically, for each feature with missing values, the mean of the available values in that column was calculated and used to fill in the missing entries. This approach preserves all instances in the dataset and avoids unnecessary data loss, which could harm model performance, especially with limited data. Mean imputation is a simple yet effective technique when the missing data is not heavily skewed and is missing at random.

The dataset used in this project consists entirely of **numerical financial ratios**, and thus **does not include any categorical features**. Therefore, no special encoding or transformation of categorical variables was necessary. If the dataset had included features like sector, region, or firm size category, appropriate encoding methods such as One-Hot Encoding or Label Encoding would have been applied.

Yes, normalization is required since many of the models used (especially distance-based or gradient-based algorithms like SVM, Logistic Regression, and boosting methods) perform better when the features are on a similar scale. To achieve this, **standardization (Z-score normalization)** was applied using StandardScaler, which transforms the data so that each feature has a mean of 0 and a standard deviation of 1. This scaling was done **inside a pipeline**, ensuring that it is applied only to the training folds during cross-validation, thus avoiding data leakage.

Model Selection and Training

Using **Stratified K-Fold Cross-Validation (k=5)**, we evaluated various classifiers including Random Forest, Gradient Boosting, AdaBoost, XGBoost, CatBoost, Logistic Regression, and others. Evaluation metrics included **accuracy, precision, recall, and F1-score**.

As you can see in the table below, we trained more models than requested. Because the f1-score in the models we trained did not satisfy us.

Based on the results, **CatBoostClassifier** achieved the best performance with:

- **Accuracy**: 97.83%
- **Precision**: 83.09%
- **Recall**: 69.15%
- **F1-score**: **75.45%** (highest among all)

Therefore, **CatBoostClassifier** was selected as the best model due to its superior balance between precision and recall, as reflected in the highest F1-score.

Model	Accuracy	Precision	Recall	F1-Score
RandomForestClassifier	0.9615	0.6156	0.5399	0.5749
GradientBoostingClassifier	0.9194	0.3370	0.6934	0.4533
AdaBoostClassifier	0.8347	0.1732	0.6437	0.2728
ExtraTreesClassifier	0.9544	0.5365	0.3984	0.4570
HistGradientBoostingClassifier	0.9654	0.6237	0.7064	0.6622
BaggingClassifier	0.9823	0.4823	0.5524	0.5147
XGBClassifier	0.9765	0.8029	0.6781	0.7349
LGBMClassifier	0.9646	0.6180	0.6949	0.6540
CatBoostClassifier	**0.9783**	**0.8309**	**0.6915**	**0.7545**
LogisticRegression	0.7092	0.1032	0.6552	0.1784
SVC	0.6880	0.0985	0.6719	0.1718

Fine-tune Your Model

In this section, the model selected in the previous step, which was the CatBoostClassifier, is fine-tuned using a hyperparameter optimization approach. Hyperparameter tuning is a crucial step to improve the performance of the model by selecting the best combination of hyperparameters that result in the most accurate predictions. For this task, we utilized **Optuna**, an automatic hyperparameter optimization framework.

Hyperparameter Tuning Process

To fine-tune the CatBoostClassifier, we performed the following steps:

1. **Objective Function Definition**: The objective function defines the parameters to optimize. In this case, we used Optuna to minimize the cross-validation loss, specifically focusing on the **F1-score** to balance precision and recall, especially for the minority class (class 1 – bankrupt).

2. **Search Space**: The hyperparameters chosen for tuning include:

    - **Learning rate** (learning_rate): Controls how much the model learns with each iteration.
    - **Depth** (depth): The maximum depth of the tree.
    - **Number of trees** (iterations): The number of boosting rounds.
    - **L2 Regularization** (l2_leaf_reg): Regularization parameter to control overfitting.
    - **Random State(**random_state**):** 42

3. **Optuna Optimization**: The **Optuna** framework was used to search for the best combination of hyperparameters. It uses **trial-based optimization**, where each trial corresponds to a unique set of hyperparameters, and the optimization process automatically adjusts the hyperparameters to maximize performance. We fine-tuned the model twice due to a low F1 score. Initially, the training was constrained to 30 minutes with 15 trials, but after observing a low F1 score, we extended the search to 2 hours with 40 trials. This adjustment helped us find more optimal hyperparameters and improve the model's performance.
4. **Cross-Validation**: To ensure the robustness of the tuned model, we used **k-fold cross-validation** with 5 splits. This helps in evaluating the model’s performance on different subsets of the dataset and prevents overfitting to a single split.

Best Hyperparameters

The best hyperparameters found by Optuna for the CatBoostClassifier in first try were:

- **Learning rate**: 0.16206510949819358
- **Depth**: 7
- **Iterations**: 484
- **L2 Regularization**: 7.0
- **Best F1-Score:** 0.7508350662668773

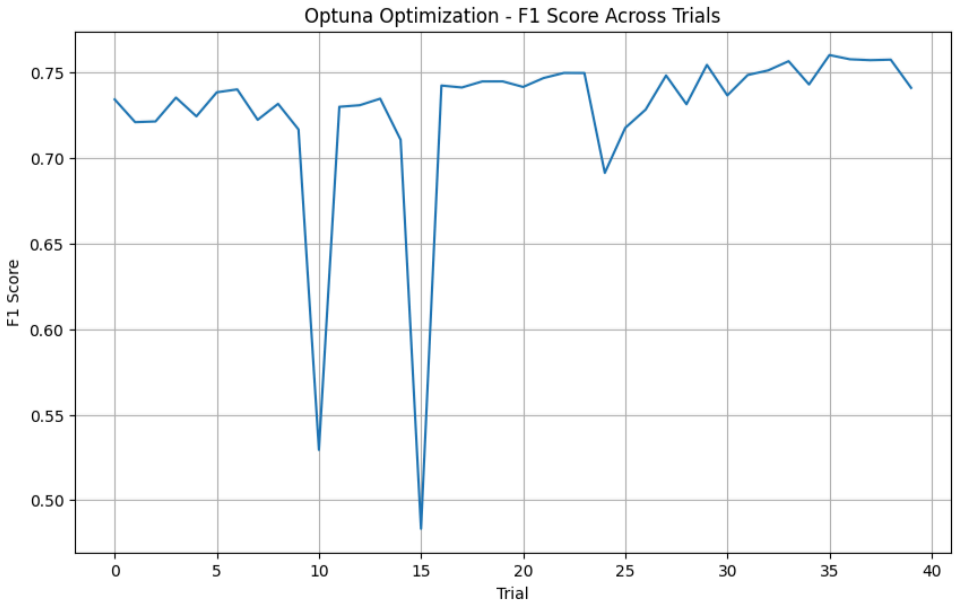
The best hyperparameters found by Optuna for the CatBoostClassifier in second try were:

- **Learning rate**: 0.21686651662049206
- **Depth**: 7
- **Iterations**: 412
- **L2 Regularization**: 2.0
- **Best F1-Score:**0.7602785318012055

These hyperparameters were selected to balance both **precision** and **recall**, especially focusing on reducing the false negatives for the minority class (bankrupt companies).

Model Evaluation

After fine-tuning the hyperparameters, the final model was trained using the best parameters identified by Optuna. The performance of the fine-tuned model was evaluated using **cross-validation** and several performance metrics including **precision**, **recall**, **F1-score**, and the **confusion matrix**.



(**Figure 1.** F1 Score Progression Across Trials During Optuna Hyperparameter Optimization)

Testing

We evaluated the final model using cross-validation on the full dataset after applying SMOTE and scaling. The model was tested with the best hyperparameters obtained through Optuna. The evaluation metrics show that the model achieved an overall **accuracy of 97.91%**. The performance on the minority class (class 1) was also satisfactory, with a **precision of 0.8509**, **recall of 0.6877**, and an **F1-score of 0.7606**. These values indicate that the model is relatively good at identifying positive cases (bankrupt companies), although there is still room for improvement in recall. The majority class (class 0) showed a very strong performance with an **F1-score of 0.9891**. The macro average F1-score was **0.8749**, showing a balanced performance across both classes, and the weighted average F1-score was **0.9781**, confirming high overall performance.

We evaluated the performance of our model on the test dataset by retraining it with the best hyperparameters obtained in the previous section. The model's performance on the test set was as follows:

- **Accuracy:** 0.9791
- **Precision (Class 0)**: 0.9843
- **Precision (Class 1):** 0.8509
- **Recall (Class 0):** 0.9939
- **Recall (Class 1):** 0.6877
- **F1-Score (Class 0):** 0.9891
- **F1-Score (Class 1):** 0.7606

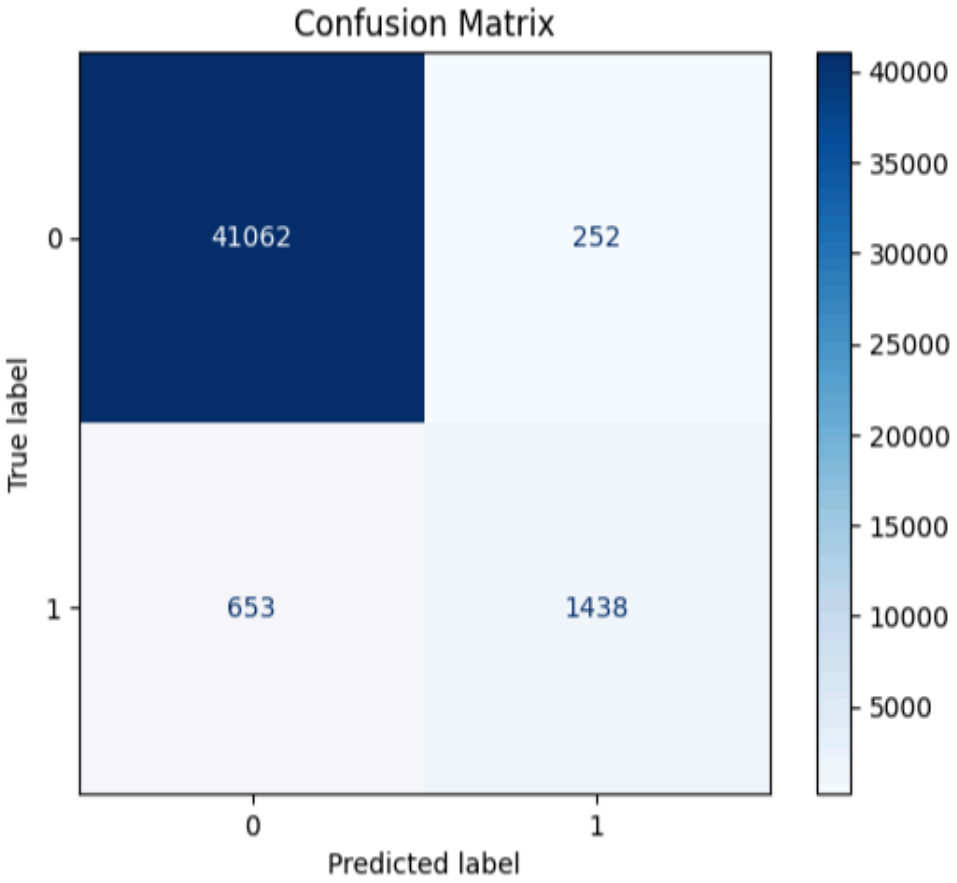
Classification Report:

	precision	recall	f1-score	support
0	0.9843	0.9939	0.9891	41314
1	0.8509	0.6877	0.7606	2091
accuracy			0.9791	43405
macro avg	0.9176	0.8408	0.8749	43405
weighted avg	0.9779	0.9791	0.9781	43405

(**Figure 2:**test results after best hyperparameters)

From the results presented above, the following conclusion can be inferred:

- **Accuracy**: The model achieved a high accuracy of 97.91%, demonstrating overall good performance across the dataset.
- **Precision**: Precision is high for Class 0 (0.9843), but lower for Class 1 (0.8509), indicating that the model performs better at identifying Class 0 but misclassifies Class 1 more often.
- **Recall**: The recall for Class 0 is very high (0.9939), meaning the model almost perfectly identifies all instances of Class 0. However, recall for Class 1 (0.6877) is lower, suggesting that the model fails to recognize some of the Class 1 instances.
- **F1-Score**: The F1 score for Class 0 is excellent (0.9891), while for Class 1, it is lower (0.7606). This shows that the model performs well for Class 0, but there is room for improvement in Class 1.



(Figure 3: Confusion Matrix)

Summary

The model's performance shows good results, with an overall F1-score of 0.7602 after fine-tuning. Initially, the F1-score for Class 1 was relatively low, which led us to train the model for a longer duration and with more trials. This adjustment helped improve the F1-score from 0.7508 to 0.7602. The likely cause for the lower performance, particularly in Class 1, is the imbalanced dataset.

There were no major unexpected results, but the imbalance between the two classes led to some expected challenges, particularly with Class 1, which is underrepresented. Despite fine-tuning the hyperparameters and using technicus like SMOTE for oversampling the minority class, the recall for Class 1 is still relatively low. This indicates that the model might not be sensitive enough to correctly identify Class 1 instances.

To further improve the model's performance, especially on the minority class, several strategies can be considered. First, more advanced ensemble techniques such as stacking multiple classifiers or combining CatBoost with other tree-based models might help capture different aspects of the data. Additionally, using more sophisticated resampling methods like SMOTEENN or ADASYN could provide better results than plain SMOTE by both generating synthetic samples and cleaning ambiguous ones. Another effective approach is to incorporate cost-sensitive learning by adjusting class weights or modifying the loss function to penalize the misclassification of bankrupt companies more heavily. Beyond these, applying proper feature engineering or selection techniques may enhance class separability by removing irrelevant or noisy features. Lastly, collecting additional data for the minority class or exploring anomaly detection techniques may also help improve the model's ability to identify rare but important cases such as bankruptcies.

References

For dataset: https://archive.ics.uci.edu/dataset/365/polish+companies+bankruptcy+data

For Figure 1 :Generated in jupyter notebook.

For Figure 2 :Generated in jupyter notebook.

For Figure 3 :Generated in jupyter notebook.