# Bankruptcy Prediction for Polish Companies

**Author: Kamile Yagci**

**Blog URL:** https://kamileyagci.github.io/

## Overview

In this study, I will analyze the financial standings of the Polish companies to identify whether the company will banckrupt in 1-5 years. I will use the Ensemble Method 'XGBoost', eXtreme Gradient Boosting, for classification.

## Repository Content

- data: directory containing the data files
- deliverables: directory containing the files to be submitted for project review
- figures: directory containing figures/images
- notUsed: directory containing some prelimenary analysis, which is not part of the latest version of the project
- saved_model_history: directory containing the saved models
- .gitignore: text file that contains the list of files/directories that should not be tracked by git repository
- README.md: markdown file that describes the git repository and the project
- analysis_1_explore.ipynb: jupyter notebook for analysis part 1, data exploring
- analysis_2_imbalance.ipynb: jupyter notebook for analysis part 2, class imbalance study for Data 3
- analysis_3_data3.ipynb: jupyter notebook for analysis part 3, XGBoost Classification for Data 3
- analysis_4_dataAll.ipynb: jupyter notebook for analysis part 4, XGBoost Classification for all data sets AND Interpretation of Results
- functions.py: python file containing the functions used in the analysis

## Project Outline

- Business Problem
- Data
- Methods
- Analysis and Results

    - Class Imbalance
    - Model Optimization for Data 3
    - Model Performance on All Datasest
    - Final Model
- Interpretation of Results
- Conclusion
- Future Work

## Business Problem

KPMG, international corporate financial consulting firm, hired me to analyze the financial standing of the Polish companies. The goal of the analysis is identifying whether the business will go to bankruptcy in 1-5 years or not. KMPG will use the results of this study to provide an early warning to Polish business clients on their financial standings, so they can take preventive actions.

## Data

The data contains the financial information and bankruptcy status of Polish companies. It is collected from Emerging Markets Information Service (EMIS, [Web Link]).

The data was collected in the period of

- 2000-2012 for the bankrupt companies
- 2007-2013 for the still operating companies

Depending on the forecasting period, the data is classified in five categories/datasets.:

- 1st Year: the data contains financial rates from 1st year of the forecasting period and corresponding class label that indicates bankruptcy status after 5 years (1year.arff).
- 2nd Year: the data contains financial rates from 2nd year of the forecasting period and corresponding class label that indicates bankruptcy status after 4 years (2year.arff).
- 3rd Year: the data contains financial rates from 3rd year of the forecasting period and corresponding class label that indicates bankruptcy status after 3 years (3year.arff).
- 4th Year: the data contains financial rates from 4th year of the forecasting period and corresponding class label that indicates bankruptcy status after 2 years (4year.arff).

- 5th Year: the data contains financial rates from 5th year of the forecasting period and corresponding class label that indicates bankruptcy status after 1 year (1year.arff).

There are 64 attributes for each company. The list of the attributes and their descriptions can be found on UIC page:

https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data

In my analysis, I name the five dataset files as:

- Data 1: '1year.arff'
- Data 2: '2year.arff'
- Data 3: '3year.arff'
- Data 4: '4year.arff'
- Data 5: '5year.arff'

The number of companies in each dataset and class distributions:

| Data # | Total | Still Operating (class=0) | Bankrupt (class=1) |
|---|---|---|---|
| Data 1 | 7027 | 6756 | 271 |
| Data 2 | 10173 | 9773 | 400 |
| Data 3 | 10503 | 10008 | 495 |
| Data 4 | 9792 | 9277 | 515 |
| Data 5 | 5910 | 5500 | 410 |

## Method

I will use Ensemble Method 'XGBoost', eXtreme Gradient Boosting, for classification.

This is a binary classification problem, since my goal is to identify whether the company will bankrupt or not.

Evaluation metric for data training = 'logloss'

I will focus on the performance of 'recall' metric in order to minimize false negatives. Besides, I will also keep an eye on 'precision', 'f1', 'accuracy, and 'AUC' metrics.

## Analysis and Results

Initially, I explored the XGBoost Classifier models on Data 3. After determing the best model designs, I applied them on other datasets and compare the results.

In each dataset, I have used 80% of data for training and 20% for testing.

Note: No cleaning applied to data. XGBoost Classifier can handle the missing values and outliers.

## Class Imbalance

Data 3 is used for Class Imbalance Study.

The default XGBClassifier init parameters are used (except max_depth and scale_pos_weight).

The class imbalance is one of the main issues in this data.

Imbalance Ratio = (# of class 0 companies) / (# of class 1 companies)

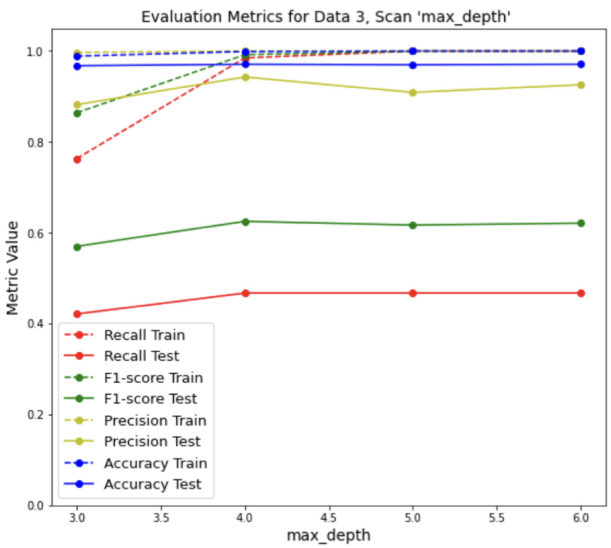
| Data # | Imbalance Ratio | Sqrt of Imbalance Ratio |
|---|---|---|
| Data 1 | 24.93 | 4.99 |
| Data 2 | 24.43 | 4.94 |
| Data 3 | 20.22 | 4.50 |
| Data 4 | 18.01 | 4.24 |
| Data 5 | 13.41 | 3.66 |

There are two approaches to deal with the class imbalance.

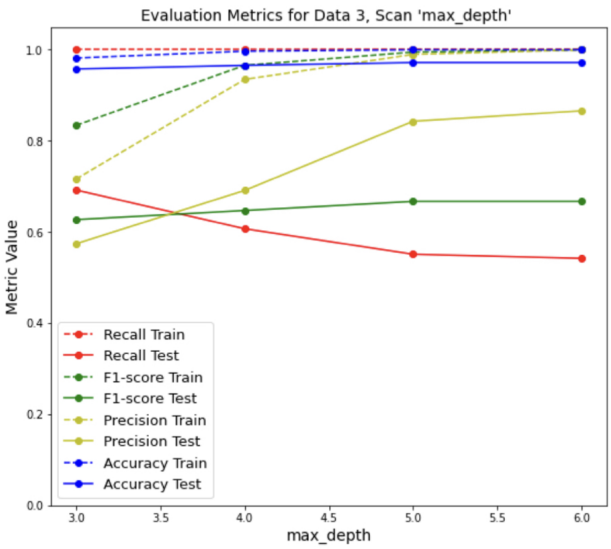
1. **sample_weight:** class weight parameter when training the data. The class weights are calculated for each dataset seperately during training.
2. **scale_pos_weight:** class weight parameter when initiating the classifier. I provide certain constant values to initiate the classifier.

The graphs below show the effect of sample_weight on model performance at several max_depth values for Data 3. Left: No sample_weight applied. Right: sample_weight method applied.

No Class Weight

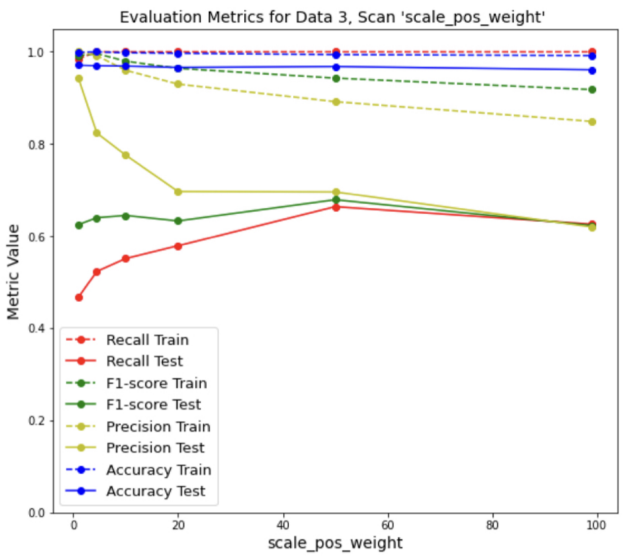



With sample_weight

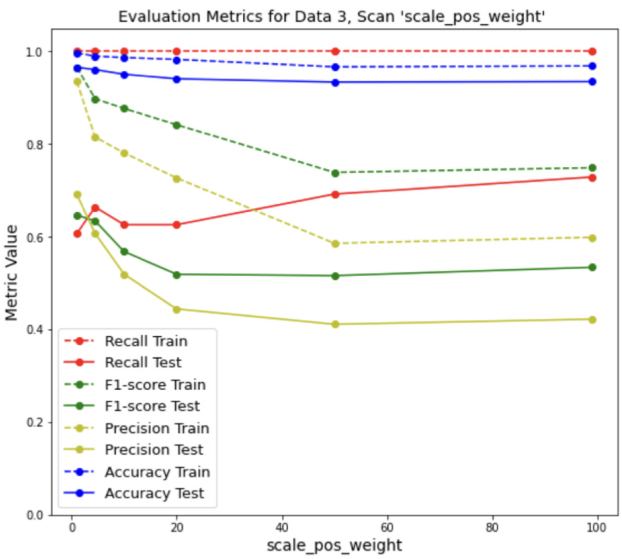



The graphs below shows the performance of model as sample_pos_weight varies, at max_depth=4 for Data 3. Left: only scale_pos_weight applied. Right: both sample_weight + scale_pos_weight applied.

scale_pos_weight




sample weight + scale_pos_weight




The table below shows the comparison of imbalance treatment approaches at max_depth=4 for Data 3:

| Params | precision | recall | f1 | accuracy | auc |
|---|---|---|---|---|---|
| **sample_weight Train** | 0.935 | 1.000 | 0.966 | 0.997 | 1.000 |
| **Test** | 0.691 | 0.607 | 0.647 | 0.966 | 0.920 |
| **scale_pos_weight=50 Train** | 0.892 | 1.000 | 0.943 | 0.994 | 1.000 |
| **Test** | 0.696 | 0.664 | 0.679 | 0.968 | 0.932 |
| **sample_weight + scale_pos_weight=4.5 Train** | 0.815 | 1.000 | 0.898 | 0.990 | 1.000 |
| **Test** | 0.607 | 0.664 | 0.634 | 0.961 | 0.922 |
| **sample_weight + scale_pos_weight=50 Train** | 0.586 | 1.000 | 0.739 | 0.967 | 1.000 |
| **Test** | 0.411 | 0.692 | 0.516 | 0.934 | 0.911 |

Findings:

- Applying sample_weight or sample_pos_weight improves the model performance:
  - increases recall
  - decreases overfitting
- sample_weight is more effective at lower max_depth (3 or 4)
- Using both approaches together provides a better result.
- Optimum configurations:
  - max_depth=4: sample_weight + scale_pos_weight=4.5 (~square root of imbalance ratio)
  - max_depth=5: sample_weight + scale_pos_weight=20 (~imbalance ratio)
  - max_depth=6: sample_weight + scale_pos_weight=20 (~imbalance ratio)

## Model Optimization for Data 3

I performed a detailed analysis on Data 3.

The ovefitting looked a like a main issue for this study. I tried to control overfitting by tuning the XGBoost Classifier Parameters, while trying to improve the model at the same time.

I firstly used GridSearchCV on few parameters to obtain best performance parameters. Then, I looked in parameters that effect overfitting.

The paramaters that can affect overfitting are grouped in four categories:

1. Boosting Rounds
    - n_estimators
2. Pruning:
    - max_depth
    - min_child_weight
    - gamma
3. Regularization:
    - learning_rate
    - max_delta_step
    - reg_lambda
    - reg_alpha
4. Sampling
    - subsample
    - colsample_bytree

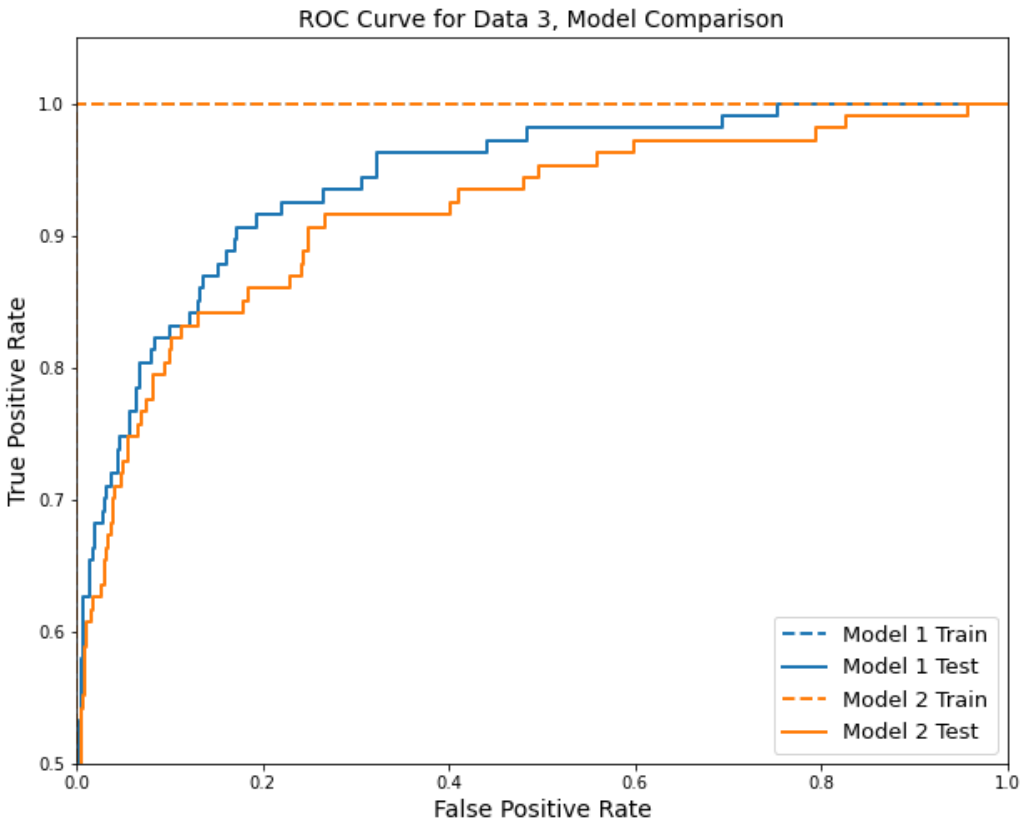
I started with baseline model and did improve the performance step by step.

**Model 1: Baseline Model**

- Default XGBClassifier parameters:
    - base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1, eval_metric='logloss', gamma=0, gpu_id=-1, importance_type='gain', interaction_constraints='', learning_rate=0.300000012, max_delta_step=0, **max_depth=6**, min_child_weight=1, missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=4, num_parallel_tree=1, random_state=42, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact', validate_parameters=1, verbosity=None

**Model 2: Model with Class Imbalance treated**

- Optimized class balancing method used, in addition to the XGBClassifier default parameters.
- max_depth=6: sample_weight + scale_pos_weight=20 (~imbalance ratio)
- Model performance increased compared to Model 1.
- See the ROC curve graph below and summary metrics table at the end of 'Model Optimization for Data 3' section.




**Model 3: Model after GridSearchCV Parameter Tuning**

- I have applied GridSearcCV on the parameters: n_estimators, max_depth, min_child_weight, learning_rate, subsample
- Scoring = 'f1'. Even though, my focus is on 'recall', I use 'f1; in GridSearchCV to obtain overall good performance.
- I only used sample_weight method for parameter tuning. I chose not to use 'scale_pos_weight' during grid search, since the optimum value varies depending on the max_depth. After parameter tuning, I applied the optimum scale_pos_weight on data training.
- The best parameters:
    - n_estimators = 150
    - max_depth = 5
    - min_child_weight = 1
    - learning_rate = 0.20
    - subsample = 1

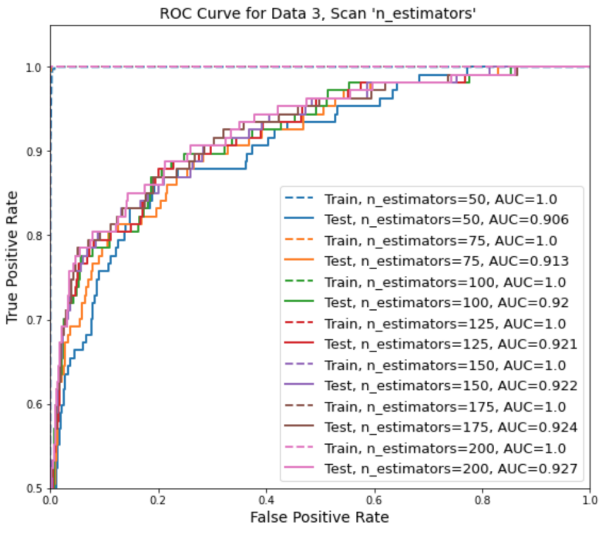
- The GridSearch Model improved the model performance.
- However, the large overfitting still exists.
- See summary metrics table at the end of 'Model Optimization for Data 3' section.

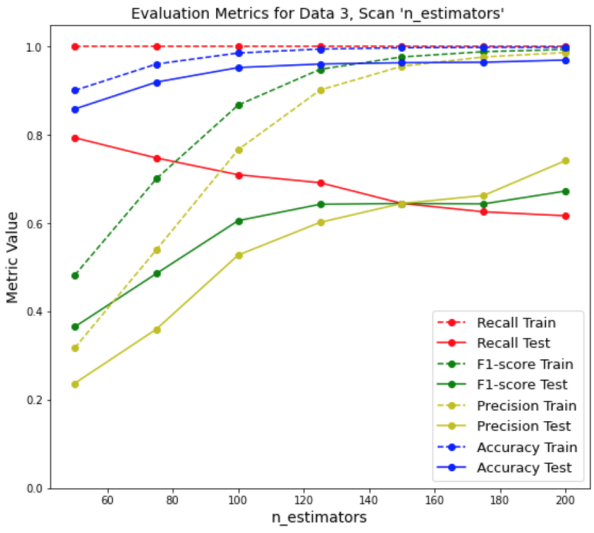
**Model 7: Optimized Model at max_depth=5**

- From Model 4 to Model 7, I have tried to tune the parameters, at max_depth=5, in order to
    - decrease overfitting
    - increase 'recall' metric
    - avoid large decrease in 'precision' as much as possible
- I have scanned the parameters, which affects the overfitting, step by step in an order as listed at the beginning of 'Model Optimization for Data 3' section.
- Model 7 has the optimum design at max_depth=5 for Data 3.

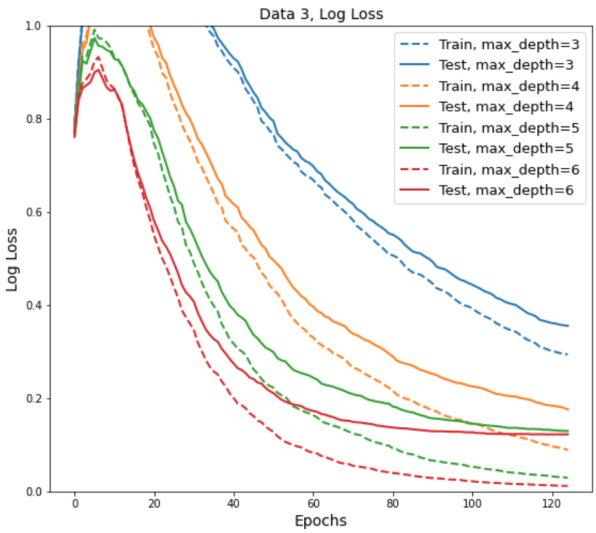
The scan results of the selected parameters are shown below:

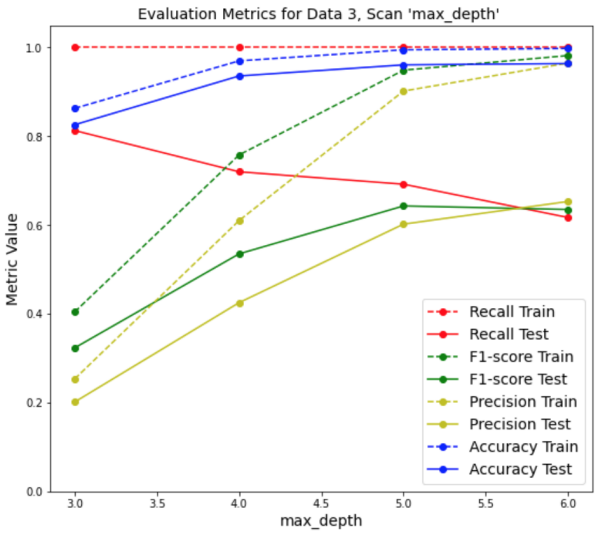
- n_estimators: ROC curve and metrics



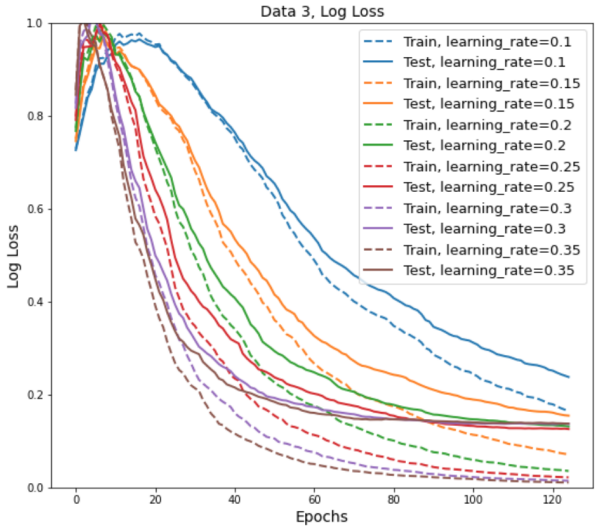



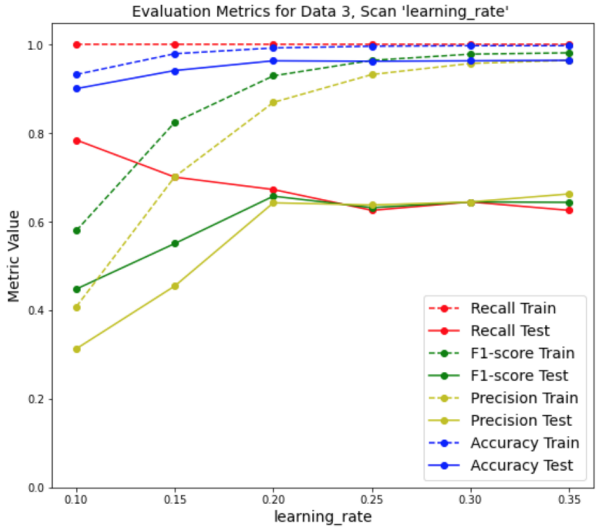

- max_depth: Log loss and metrics



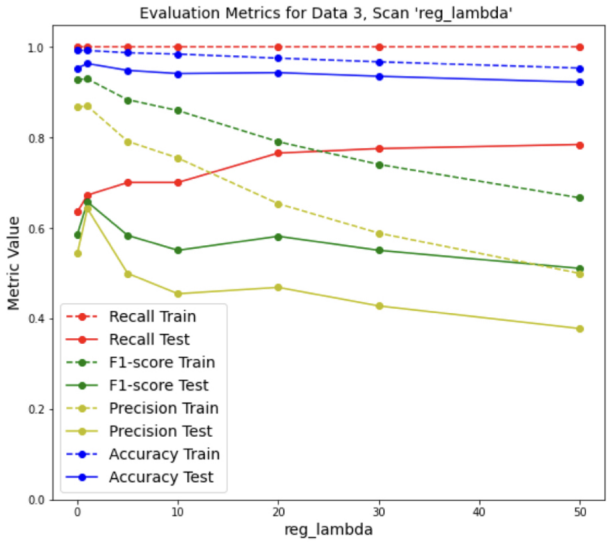

- learning_rate: Log loss and metrics



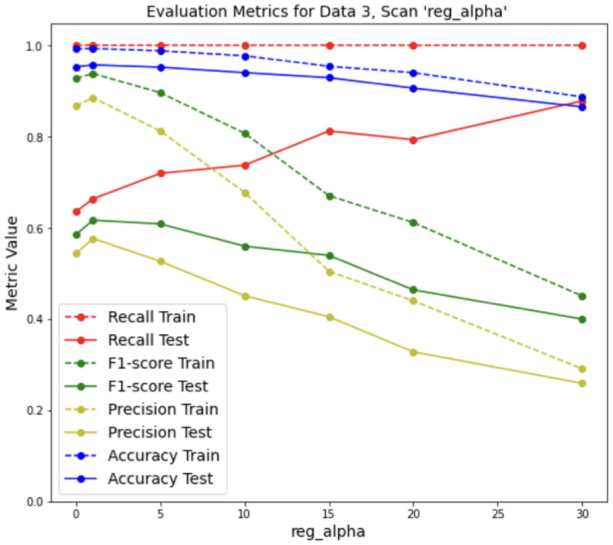




- L1 and L2 Regularization metrics

L2 Regularization




L1 Regularization




- Optimum paratemters for max_depth=5:
  - scale_pos_weight = 20,
  - n_estimators = 125,
  - max_depth = 5,
  - min_child_weight = 3,
  - gamma = 0,
  - learning_rate = 0.20,
  - max_delta_step = 0,
  - reg_lambda = 0,
  - reg_alpha = 5,
  - subsample = 1,
  - colsample_bytree = 0.7

**Model 9: Optimized Model at max_depth=4**

- Same optimization approach is followed for max_depth=4.
- Optimum paratemters for max_depth=4:
  - scale_pos_weight = 4.5,
  - n_estimators = 125,
  - max_depth = 4,
  - min_child_weight = 3,
  - gamma = 0,
  - learning_rate = 0.20,

- max_delta_step = 0,
- reg_lambda = 1,
- reg_alpha = 0,
- subsample = 1,
- colsample_bytree = 1

**Model 10: Optimized Model at max_depth=6**

- Same optimizationapproach is followed for max_depth=6.
- Optimum paratemters for max_depth=6:
  - scale_pos_weight = 20,
  - n_estimators = 80,
  - max_depth = 6,
  - min_child_weight = 3,
  - gamma = 0,
  - learning_rate = 0.25,
  - max_delta_step = 4,
  - reg_lambda = 1,
  - reg_alpha = 0,
  - subsample = 1,
  - colsample_bytree = 1

**Best Model for Data 3**

- Model 9, optimized at max_depth=4, is the best model for Data 3.
  - Maximum recall
  - Moderate precision
  - Minumum overfitting

The table below compares the performance of models for Data 3:

| Params | precision | recall | f1 | accuracy | auc |
|---|---|---|---|---|---|
| **Model 1 Train** | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| **Test** | 0.926 | 0.467 | 0.621 | 0.971 | 0.941 |
| **Model 2 Train** | 0.990 | 1.000 | 0.995 | 1.000 | 1.000 |
| **Test** | 0.677 | 0.607 | 0.640 | 0.965 | 0.916 |
| **Model 3 Train** | 0.956 | 1.000 | 0.977 | 0.998 | 1.000 |
| **Test** | 0.645 | 0.645 | 0.645 | 0.964 | 0.922 |
| **Model 7 Train** | 0.771 | 1.000 | 0.871 | 0.986 | 1.000 |
| **Test** | 0.535 | 0.720 | 0.614 | 0.954 | 0.923 |
| **Model 9 Train** | 0.719 | 1.000 | 0.836 | 0.982 | 1.000 |
| **Test** | 0.554 | 0.720 | 0.626 | 0.956 | 0.916 |
| **Model 10 Train** | 0.900 | 1.000 | 0.947 | 0.995 | 1.000 |
| **Test** | 0.636 | 0.701 | 0.667 | 0.964 | 0.931 |

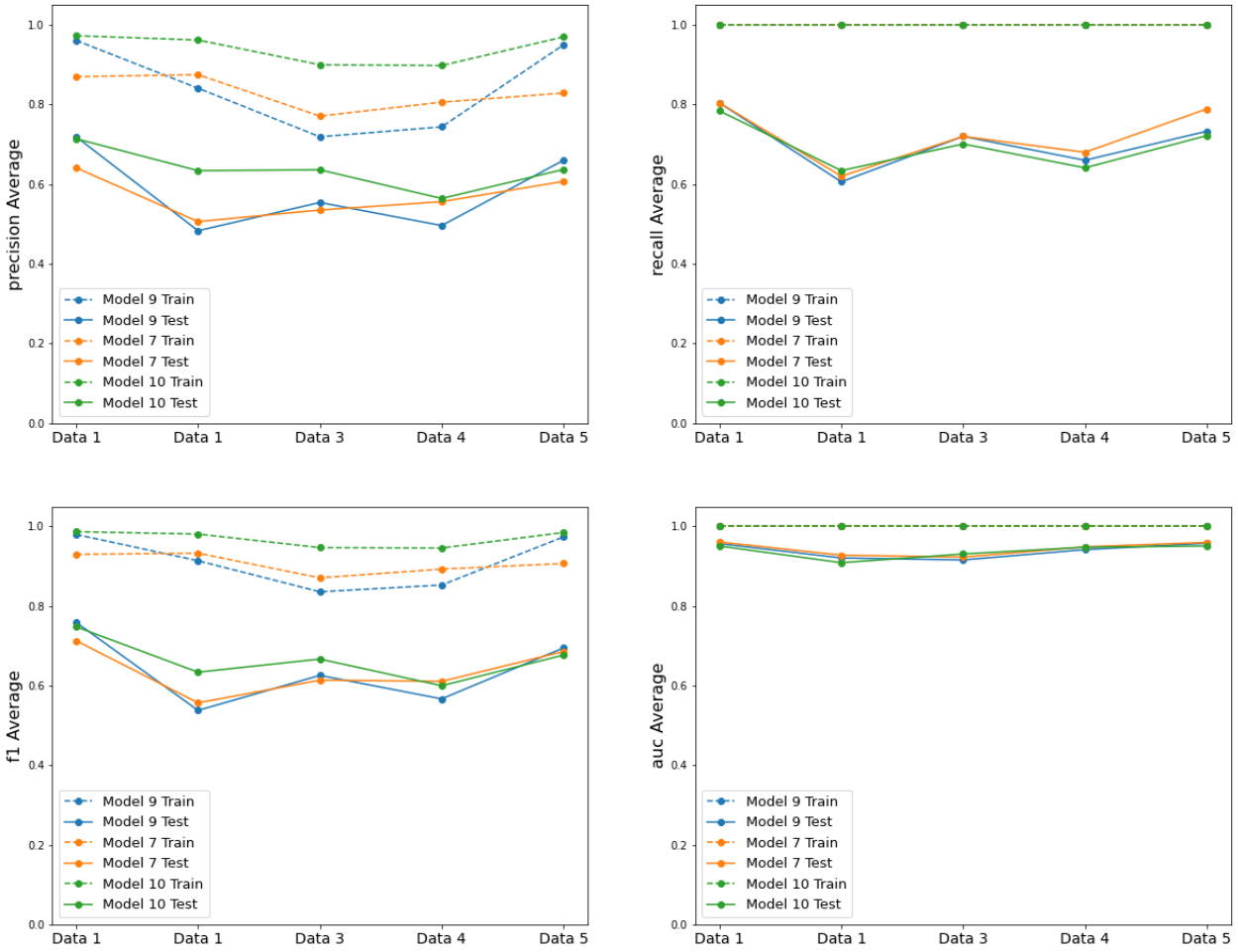
## Model Performance on All Datasest

I have applied the optimumum models on all five datasets.

- Model 7 at max_depth=5
- Model 9 at max_depth=4
- Model 10 at max_depth=6

Note: I also tried to tune the parameters to see if I can get a model performs equally good, better, on all data files. But no success.

Metrics graphs for comparison:




Model 9 (max_depth=4): * Previously Selected performance for data 3 * Overfitting is least on Data 3

Model 7 (max_depth=5): * Performance is similar to Model 9 * More smooth performance on all datasets

Model 10 (max_depth=6): * Highest metrics * But overfitting is sligtly larger

Selected model: Model 7

## Final Model

Model 7 is the best performing, when all datasets are considered.

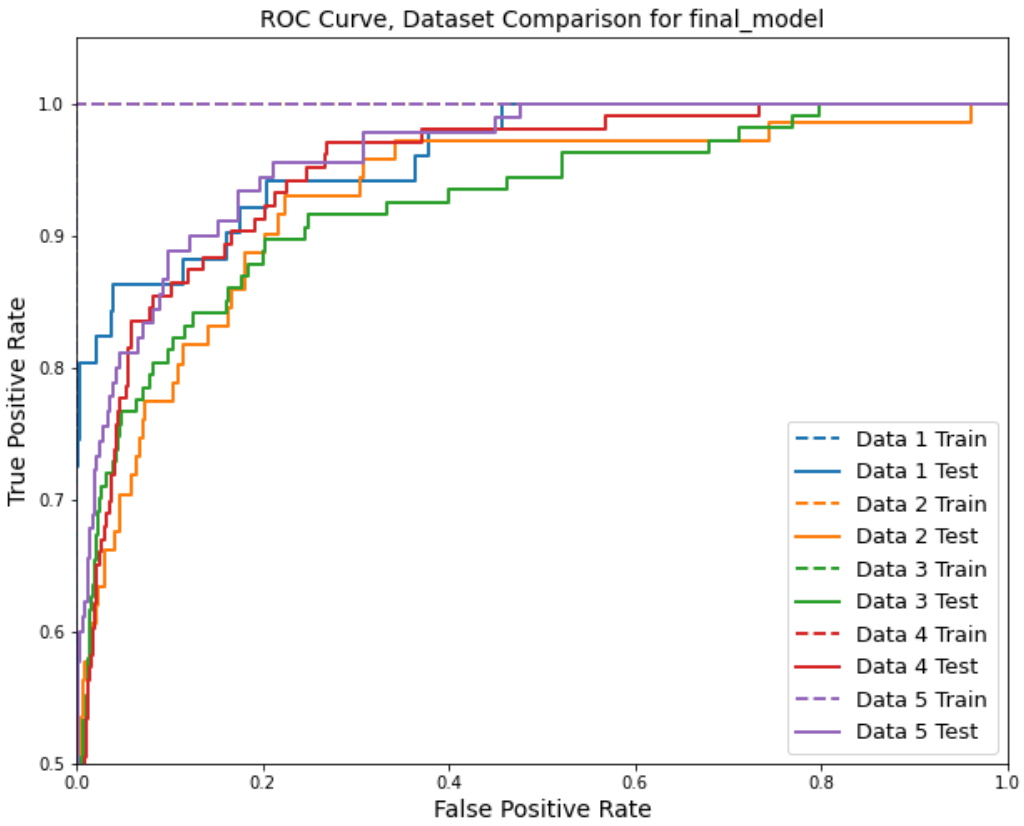
- Final Model parameters:
    - scale_pos_weight = 20,
    - n_estimators = 125,
    - max_depth = 5,

- min_child_weight = 3,
- gamma = 0,
- learning_rate = 0.20,
- max_delta_step = 0,
- reg_lambda = 0,
- reg_alpha = 5,
- subsample = 1,
- colsample_bytree = 0.7

- Metrics table:

| Data | Sample | precision | recall | f1 | accuracy | auc |
|---|---|---|---|---|---|---|
| **Data 2** | Train | 0.8750 | 1.0000 | 0.9330 | 0.9940 | 1.0000 |
| **Data 2** | Test | 0.5060 | 0.6200 | 0.5570 | 0.9660 | 0.9280 |
| **Data 3** | Train | 0.7710 | 1.0000 | 0.8710 | 0.9860 | 1.0000 |
| **Data 3** | Test | 0.5350 | 0.7200 | 0.6140 | 0.9540 | 0.9230 |
| **Data 4** | Train | 0.8060 | 1.0000 | 0.8930 | 0.9870 | 1.0000 |
| **Data 4** | Test | 0.5560 | 0.6800 | 0.6110 | 0.9550 | 0.9490 |
| **Data 5** | Train | 0.8290 | 1.0000 | 0.9070 | 0.9860 | 1.0000 |
| **Data 5** | Test | 0.6070 | 0.7890 | 0.6860 | 0.9450 | 0.9600 |
| **Average** | Train | 0.8302 | 1.0000 | 0.9068 | 0.9894 | 1.0000 |
| **Average** | Test | 0.5690 | 0.7226 | 0.6362 | 0.9594 | 0.9442 |

- ROC curve:




### Class 0 predictions for Final Model

The table below shows the prediction metrics of the final model for the still operating companies (class 0).

| Data # | sample | precision | recall | f1 |
|---|---|---|---|---|
| Data 1 | Train | 1.00 | 0.99 | 1.00 |
| Data 1 | Test | 0.99 | 0.98 | 0.99 |
| Data 2 | Train | 1.00 | 0.99 | 1.00 |
| Data 2 | Test | 0.99 | 0.98 | 0.98 |
| Data 3 | Train | 1.00 | 0.99 | 0.99 |
| Data 3 | Test | 0.98 | 0.97 | 0.98 |
| Data 4 | Train | 1.00 | 0.99 | 0.99 |
| Data 4 | Test | 0.98 | 0.97 | 0.98 |
| Data 5 | Train | 1.00 | 0.99 | 0.99 |

| Data # | sample | precision | recall | f1 |
|---|---|---|---|---|
| Data 5 | Test | 0.98 | 0.96 | 0.97 |

**Best common predictors**

These three attributes are good predictors for all five datasets.

- X27: profit on operating activities / financial expenses
- X34: operating expenses / total liabilities
- X5: [(cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] * 365

## Interpretation of Results

5-year Period (Data 1):

- Model correctly identifies the 80.4% of the true bankrupt companies, which will bankrupt 5 years later. (recall)
- Among the model predicted bankruptcy companies, 64.1% of them are true bankrupt companies, which will bankrupt 5 years later. (precision)
- The Harmonic Mean of Precision and Recall (f1-score) is 71.3%.

4-year Period (Data 2):

- Model correctly identifies the 62.0% of the true bankrupt companies, which will bankrupt 4 years later. (recall)
- Among the model predicted bankruptcy companies, 50.6% of them are true bankrupt companies, which will bankrupt 4 years later. (precision)
- The Harmonic Mean of Precision and Recall (f1-score) is 55.7%.

3-year Period (Data 3):

- Model correctly identifies the 72.0% of the true bankrupt companies, which will bankrupt 3 years later. (recall)
- Among the model predicted bankruptcy companies, 53.5% of them are true bankrupt companies, which will bankrupt 3 years later. * The Harmonic Mean of Precision and Recall (f1-score) is 61.4%.

2-year Period (Data 4):

- Model correctly identifies the 68.0% of the true bankrupt companies, which will bankrupt 2 years later. (recall)
- Among the model predicted bankruptcy companies, 55.6% of them are true bankrupt companies, which will bankrupt 2 years later. (precision)
- The Harmonic Mean of Precision and Recall (f1-score) is 61.1%.

1-year Period (Data 5):

- Model correctly identifies the 78.9% of the true bankrupt companies, which will bankrupt 1 years later. (recall)
- Among the model predicted bankruptcy companies, 60.7% of them are true bankrupt companies, which will bankrupt 1 years later. (precision)
- The Harmonic Mean of Precision and Recall (f1-score) is 68.6%.

On Average:

- Model correctly identifies the 72.3% of the true bankrupt companies. (recall)
- Among the model predicted bankruptcy companies, 56.9% of them are true bankrupt companies. (precision)
- The Harmonic Mean of Precision and Recall (f1-score) is 63.6%.

Class 0 predictions:

- Model correctly identifies the ~97% of the true still operating companies. (recall, class 0)
- Among the model predicted still operating companies, ~98% of them are true still operating companies. (precision, class 0)
- The Harmonic Mean of Precision and Recall (f1-score, class 0) is ~98%.

## Conclusion

I had three main challenges in this project:

1. **Class Imbalance:**
    - Resolved.
    - Balancing the sample increased the recall and so decreased the precision. The balance sample has moderate evaluation scores. See Model 2 for Data 3.
    - However overfitting didn't improve much.
2. **Low recall score:**

- I managed to improve recall significantly for all datasets. For instance, Data 3 recall score improved from 0.467 to 0.720.
- I couldn't go for higher recall value, because the precision was decreasing dramatically (below 0.5). The recall and precision are inversely proportional.

3. **Large overfitting:**

   - I did decrease the overfitting, but not on desired level.
   - I tuned parameters which are effecive on overfitting, and find the optimum designs that produces low overfitting, large recall and moderate precision.
   - However, I couldn't enforce larger reduction in overfitting, since it causes the precision go below 0.5. which is the random guess probability.

Overall, my model correctly identifies

- 72.3% of the true bankrupt companies
- 97% of the true still operating companies

## Future Work

- Create seperate final models for each dataset; not just one final model that applies on all.
- Search for alternative Classifier methods/tools.
- Simply/shorten functions that are created during the project. They have repeating codes.

### Releases

No releases published
Create a new release

### Packages

No packages published
Publish your first package

### Languages


Jupyter Notebook 99.7%
Python 0.3%