Case Study 4: Five Business Over 5 Years - Bankruptcy Files

Tim Cabaza

June 24, 2024

**Abstract**

The case study is an exercise in use RandomizedSearchCV to compare and beat the results of a tuned-random forest model versus a tuned XGboost Model. The goal of the classifier is to predict a binary target (1/0) of whether a business will go bankrupt to inform business decisions on divesting from said companies. The data provided is a 5 year period from 5 different business financial data and the stakeholder believes that a classifier model can accurately provide insight into future bankruptcies. Note this is not Time Series data. The following is a list of steps that are covered in the case study: Exploratory Data Analysis, Data Cleansing, Label Encoding, Normalizing, RandomForestClassifer, XGBOOST, Randomized Classifier Cross validation.

**1 Introduction**

The financial data provided is over a 5 year period from 5 different businesses. All the features are numeric in nature except for the target variable (bankruptcy) represented as a binary 1 or 0. In reviewing the data provide (5 separate files), the features having missing values but the missing values for most features represent less than <1% of the data per feature and there is no missing data in the target variable. The missing data required imputation and the datatypes required transformation to numeric datatypes (all columns were numeric in nature but represented as objects) while the target variable was transformed to categorical (originally an int64, the values already binary (0,1) no label encoding required).

**Table 1: Data**

|  |  |
| --- | --- |
| **Rows** | 43,405 |
| **Columns** | 65 |

**Table 2:** Missing Data

Abbreviated to show the only the first 10 columns

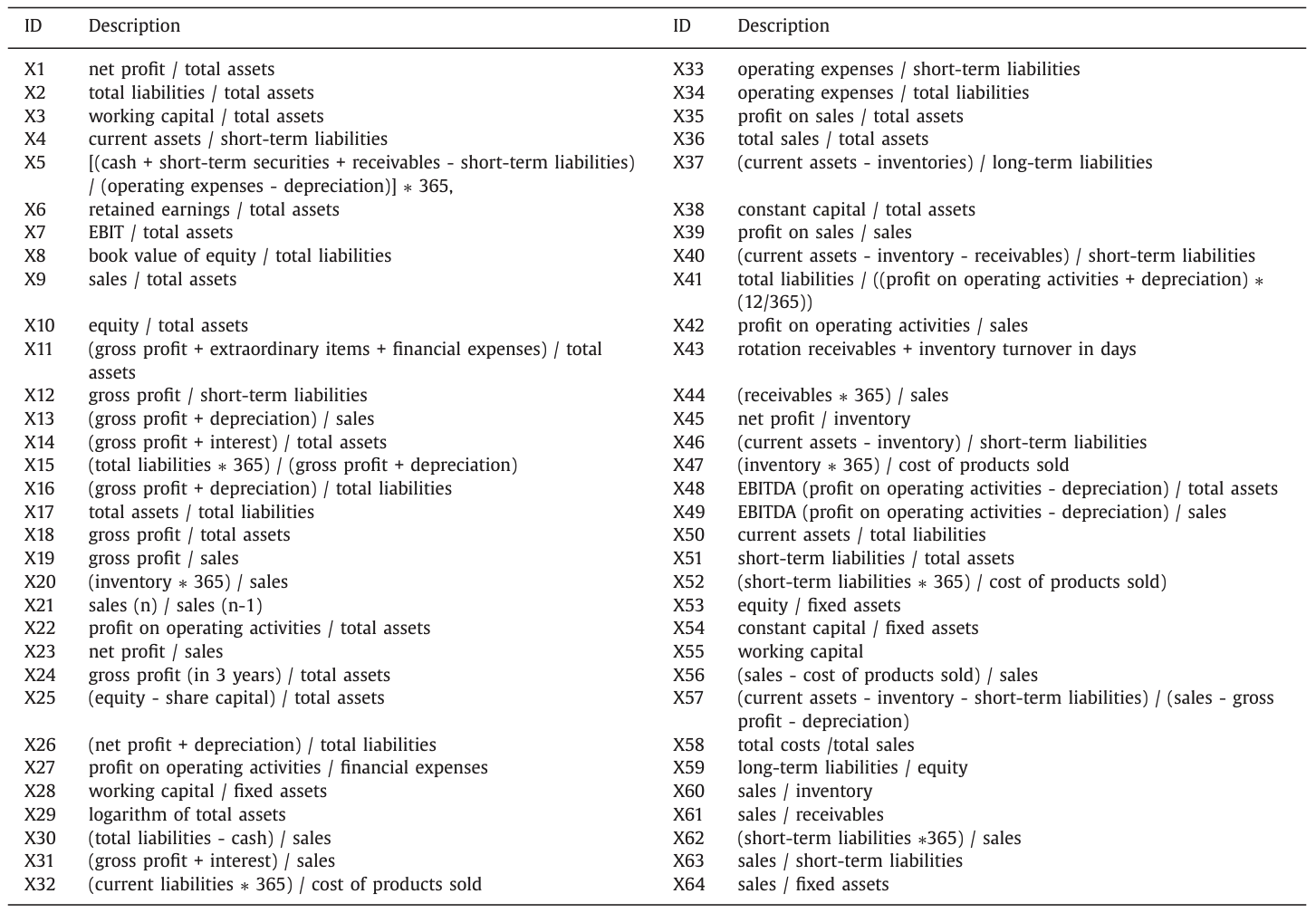
|  |  |
| --- | --- |
| **Column #** | **Number of Missing Values** |
| **0** | **8** |
| **1** | **8** |
| **2** | **8** |
| **3** | **134** |
| **4** | **89** |
| **5** | **8** |
| **6** | **8** |
| **7** | **94** |
| **8** | **9** |
| **9** | **8** |
| **10** | **44** |

**Table 3:** Columns with >1% Missing Values

|  |  |  |
| --- | --- | --- |
| **Column #** | **# of Missing Values** | **% Missing** |
| **20** | **5854** | **13%** |
| **23** | **922** | **2%** |
| **26** | **2764** | **6%** |
| **27** | **812** | **2%** |
| **36** | **18984** | **44%** |
| **40** | **754** | **2%** |
| **44** | **2147** | **5%** |
| **53** | **812** | **2%** |
| **59** | **2,152** | **5%** |
| **63** | **812** | **2%** |

**Table 4: Column Information**

Source: https://www.sciencedirect.com/science/article/pii/S0957417416301592

****

**2 Methods**

**Exploratory Data Analysis:** The data has missing values that need to be addressed via imputation. The imputation values while important should not theoretically effect any resulting model greatly, thus it was decided the best course of action (since these are numeric/discrete values) was to use the average of each feature to fill in each missing value. Overall, this is a conservative approach that avoids adding any unnecessary bias into the data. Table 3 highlights the amount missing, the only concerning column was column 36 with 44% missing but rather than drop the data or delve deeper into the features meaning ‘total sales / total assets’, it too was imputed with its mean value.

**Table 5:** Imputation

Highlighting Columns with >1% Missing Values

|  |  |
| --- | --- |
| **Imputed Column Number** | **Mean Value** |
| **20** | **3.88** |
| **23** | **0.27** |
| **26** | **1107.90** |
| **27** | **6.00** |
| **36** | **105.09** |
| **40** | **7.12** |
| **44** | **14.83** |
| **53** | **24.65** |
| **59** | **448.09** |
| **63** | **72.79** |

The target variable was dropped and the features were normalized using Scikit Learns Standard Scaler.

**Table 6:** Data Shape

|  |  |
| --- | --- |
| **Data** | **Shape** |
| Target (y) | 43405, 1 |
| Features (X) | 43405, 64 |

Two models were built and tuned to predict the target variable (bankruptcy/no bankruptcy) represented by a binary 1/0. SciKitLearn’s Random Forest Classifier and xgboost (no from SciKit Learn) were utilized to create these models. The basic process was to search over a reasonable and logical space over each models hyperparameters using RandomizedSearchCV (also from Scikit Learn). The 5 fold cross validation returned scores for Accuracy, Precision, Recall, F1 Score, and an ROC-AUC. The models were optimized for ROC-AUC to get the smoothest curve and accuracy was used to judge the overall performance.

**Table 6:** Random Forest Classifier Model

|  |  |
| --- | --- |
| **Hyperparameters Tuned** | **Values/Selection** |
| n\_estimators | Testing [50], Final Model [500] |
| criterion | [‘gini’, ‘entropy’] |
| max\_depth | [None, 3, 5, 8, 10] |
| min\_samples\_split | [2, 4, 6, 8, 10] |
| min\_weight\_fraction\_leaf | [0.0, 0.1, 0.2, 0.3, 0.4] |
| min\_samples\_leaf | [2, 4, 5] |
| max\_features | [ 1, 2, 'sqrt', None] |
| min\_impurity\_decrease | 0.0 |

**Resutlts from RF random grid search at 50 n\_estimators**

Best Parameters per Metric:

- accuracy: {'n\_estimators': 50, 'min\_weight\_fraction\_leaf': 0.0, 'min\_samples\_split': 4, 'min\_samples\_leaf': 2, 'max\_features': None, 'max\_depth': 8, 'criterion': 'gini'}

- precision: {'n\_estimators': 50, 'min\_weight\_fraction\_leaf': 0.0, 'min\_samples\_split': 10, 'min\_samples\_leaf': 4, 'max\_features': 'sqrt', 'max\_depth': 8, 'criterion': 'gini'}

- recall: {'n\_estimators': 50, 'min\_weight\_fraction\_leaf': 0.0, 'min\_samples\_split': 4, 'min\_samples\_leaf': 2, 'max\_features': None, 'max\_depth': 8, 'criterion': 'gini'}

- roc\_auc: {'n\_estimators': 50, 'min\_weight\_fraction\_leaf': 0.0, 'min\_samples\_split': 4, 'min\_samples\_leaf': 2, 'max\_features': None, 'max\_depth': 8, 'criterion': 'gini'}

- average\_precision: {'n\_estimators': 50, 'min\_weight\_fraction\_leaf': 0.0, 'min\_samples\_split': 4, 'min\_samples\_leaf': 2, 'max\_features': None, 'max\_depth': 8, 'criterion': 'gini'}

accuracy: 0.9560 +/- 0.0005

precision: 0.2850 +/- 0.0197

recall: 0.0941 +/- 0.0071

roc\_auc: 0.5469 +/- 0.0035

average\_precision: 0.1324 +/- 0.0085

Classification Report:

precision recall f1-score support

0 0.97 1.00 0.98 41314

1 0.91 0.38 0.54 2091

accuracy 0.97 43405

macro avg 0.94 0.69 0.76 43405

weighted avg 0.97 0.97 0.96 43405

Confusion Matrix:

[[41230 84]

[ 1286 805]]

A line graph with orange and blue lines

Description automatically generated

A green and white chart with numbers and a green square

Description automatically generated

REDO with 500 esimators with RFClassifier and record scores, best hyperparameter per Accuracy Scores. show AUC/Curve

**Table 6:** XGBoost

|  |  |
| --- | --- |
| **Hyperparameters Tuned** | **Values/Selection** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Best parameters: {'booster': 'gbtree', 'objective': 'binary:logistic', 'eta': 0.1, 'subsample': 0.8917730007820798, 'colsample\_bytree': 0.832619845547938, 'reg\_lambda': 2, 'max\_depth': 15}

Best accuracy: 0.9677 beat the best

**Table 7:** Randomized Grid Search Configuration

|  |  |
| --- | --- |
| **Parameter** | **Value/Selection** |
| Estimator | Model (rf, xg) |
| Param\_distribution | Params (relative to model) |
| Scoring | [accuracy, precision, recall, roc\_auc, average\_precision] |
| Refit | roc\_auc |
| n\_jobs | -1 (all processors) |
| cv | 5 |
| random\_state | 42 |

Next Modeling.

Two Models (Random Forest tuning the parameters) RandomizedGRIDCV search

get AUC/Accuracy – note the hyperparameters tuned Use Random Forest and XGBoost to accurately predict bankruptcy. Tune your models for maximum accuracy, but include precision and recall as summary metrics.

**3 Results**

The dataset once readied to be modeled, sklearns Logistic RegressionCV with OVR (one-versus-rest) was done over a logspace (adjusted and tested had to use smaller values) to find the optimal regularization value (c). The best score (f1\_macro) was collected as well as the following Classification Report and Confusion Matrix Report were generated.

**Table 3**

**The Top 5 Features**

|  |  |
| --- | --- |
| Best Regurlarization Parameter : | 0.8858667904100823 |
| Top 5 Features: | * 'number\_emergency' * 'number\_diagnoses' * 'patient\_nbr' * 'encounter\_id', * 'number\_inpatient' |
| Best cross-validation F1 macro score: | 0.486207403452041 |
| Classification Report: |  |
| Confusion Matrix Report |  |

**4 Conclusions**

The case study assessed a medical dataset. There were a number of missing values and None type values that had to be either, dropped or imputed on a case by case basis. Once the data was cleansed, data types set, label-encoded, and normalized the Logisitc Regression CV from Sklearn was utilized. All data other than weight, was kept despite possible ethical questions regarding race, age, and gender. In this case, the data is used for medical reasons and to gain insights and the customer has expressed that they believe the data is important enough due to possible links, therefore it was cautiously included. However, outside the scope of the medical field given the sensitivity of the data, it would be good practice to reassess the use of these particular columns of data.

The model was then tuned for the best regularization parameter 0.8858667904100823returned an F1 score of 0.486207403452041. The top 5 most important features were: ‘number\_emergency’, ‘number\_diagnoses’, ‘patient\_nbr’, ‘ecnounter\_id’, ‘number\_inpatient’.

**Appendix: Code**