Company bankruptcy prediction

Group 3 - Jon, Giffin, Canxiu, Charles

Business Objective
Analytical Objective
Data Understanding
Data Exploration
Setup/Data preparation
Modeling
Evaluation
Deployment

Business Objective

Background: Due to the significant increase of Bankruptcy claims after 2008 financial crisis, investment portfolio managers want to proactively manage portfolio risk by reducing stock shares of the higher-risk companies from their investment portfolio

Goal: To identify higher-risk companies that could go bankrupt in the near term based on their quarterly financial statements

Task: To develop a bankruptcy prediction model that can help to inform whether a company should be removed from an investment portfolio

Analytical Objective

Business problem: Financial portfolio managers need to mitigate portfolio risk for their clients in a higher-risk investing environment, post financial crisis.

Analytical solution: Apply a binary classification model to predict whether a given company will go bankrupt in order to provide a tool for financial portfolio managers manage portfolio risk.

Machine learning method: We will use a Random Forest binary classification model with 13 financial input features in order to predict our output label ‘Bankrupt?’, where 0= ‘not bankrupt’ and 1=‘bankrupt’.

...more to come

Data Understanding

Context:

The data were collected from the Taiwan Economic Journal for the years 1999 to 2009. Company bankruptcy was defined based on the business regulations of the Taiwan Stock Exchange.

Dataset (6819 instances):

Target: Bankrupt? 1: company is bankrupt 0: company is not bankrupt

Features:

- 95 financial features of each company
- These features cover each company's financial evaluation, including solvency, capital structure ratios, profitability, cash flow rights, ownership structures, turnover ratios, cash flow ratios, growth, retention of key personnel, others

Data Exploration

Categorical (target): 1

Numerical (features): 95

Data quality assessment:

- missing value - NA
- inconsistent value - NA
- duplicate value - NA
- outlier - NA
- imbalanced: Bankrupt? - 1: 3.2%(220), 0: 96.8%(6599)
- zeros: Liability-Assets Flag - 1: 0.01%(8), 0: 99.9%(6811)
- constant values: Net Income Flag (value=1)

Setup/Data preparation

1. Training and Testing split 90/10
2. Hyperparameters
    a. Imbalance target variable
        i. SMOTE
        ii. Undersampling
        iii. SMOTETomek
    b. Multicollinearity
    c. Feature selection (95->13, 7 features from solvency category, 6 features from profitability category)
    d. Normalize
    e. Data split stratify

Modeling

Classification models

- Logistic Regression: accuracy was lower with higher FN's when features were reduced below 17 features in the final LR model. Final tuned LR model with 17 features showed 90% TP and 85% TN. LR model worked best with a combination of over/under sampling using SMOTETomek. LR can use probability threshold parameter available under pycaret's create_model function.

- Light Gradient Boosting Machine:
    - Gradient Boosting Machine (GBM) combines each decision tree in sequence, and each tree focuses on the errors from the previous one
    - Since trees are added sequentially, boosting algorithms learn slowly. In statistical learning, models that learn slowly perform better.
    - Hyperparameters like learning rate and n_estimator need to be carefully tuned to avoid overfitting

Modeling

Classification models

- Naive Bayes Classifier was tested due to its' ease of implementation and scalability. Unfortunately we discovered that it does not work well with unbalanced data. As a result the confusion matrix was significantly worse than the other models listed. (52% true positive and 47 % true negative approximately)

- K-Nearest Neighbours Classifier is a strong algorithm because it is minimally affected by unbalanced data. In fact it largely outperformed the Naive Bayes Classifier. Similar to Naive Bayes it also is very simple and easy to implement. In the end the confusion matrix resulted in approximately 72% true positive and 70% true negative.

Winning Model

Random Forest Classifier

- Higher precision comparing to the other models
- Corrects for the overfitting tendency of decision trees
- Run fast and can handle large number of inputs
- Can deal with missing data and outliers
- Easy to understand and explain to business

Develop the model by follow the below steps in Pycaret to discover the model with the highest performance

- Original model - **Tuning** - boosting - bagging - calibrating - blendering

Evaluation

Our random forest model strives to balance the tradeoff between False Positives (the model predicts bankruptcy when no bankruptcy happens) vs. False Negatives (the model predicts no bankruptcy, but bankruptcy happens). Cross-validation of model performance resulted in approximately 9% False Positive error and approximately 14% False Negative error in the output prediction classes.

The model also produces a probability score that gives a percent probability that the predicted label (0 or 1) will be true given the feature data entered at the time of prediction. The probability output column is shown under the predict_model function in pycaret.

Accuracy measures on Unseen data (Recall = 0.875; Precision =0.2333 , F1 =0.3684)

Deployment

- Styling
  - Bootstrap was used to make the app scalable as well as to ensure it was user friendly across devices/browsers
- Model
  - Random Forest Model was selected for model deployment with the highest overall F1 score between models and the lowest number of features for ease of web app use (13 features reduced from 95).
- Deployment
  - requirements.txt
  - Procfile
- Hosting
  - Uploaded the files to github
  - Deployed via Heroku

https://bankruptcy-prediction.herokuapp.com