Bankruptcy Prediction of Polish Company

Kris Ghimire, Rajesh Satluri, Suchismita Moharana

June 21, 2021

Abstract

One of the biggest losses to any company is when it goes bankrupt due to various investment strategies. Company would spend any kind of resources to stop the bankruptcy if they can find out before hand when and how they could go bankrupt. In this case study, Polish companies' historical data will be used to predict bankruptcy using Logistic Regression, Random Forest and Xgboost algorithms and we will show that XGBoost is the best preforming model based on ROC, accuracy and F1 score.

1. Introduction:

Bankruptcy is defined as a situation in which company is no longer financially stable and goes through legal process to get help in eliminating or repaying its debt under the counsel and protection of federal court. It is an unfortunate fact that many small companies sometime run through financial troubles, so being able to predict if a company will suffer through bankruptcies in the future is essential for analyzing the financial and operational condition to make better business decisions. Accurate prediction of bankruptcy is of important concern not only to the company but also to other stakeholders such as investors, financial institutes, policymakers, employees, and consumers. Prediction helps stakeholders prepare for and protect themselves against any potential major financial shocks. Figure 1 below shows the basic approach of how we will make a bankruptcy prediction.

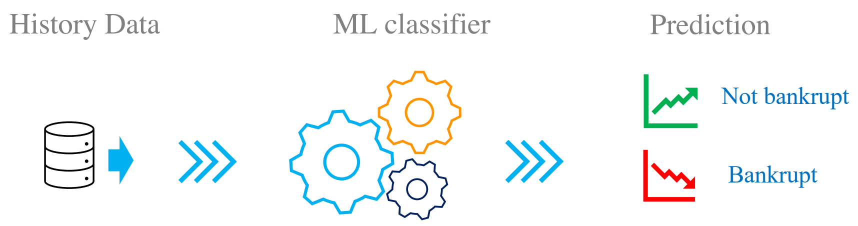
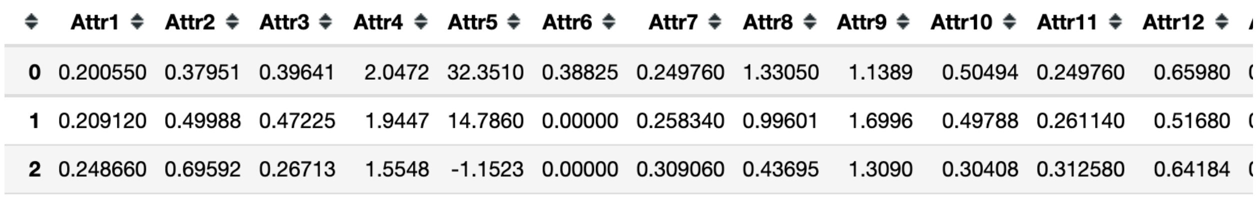


Figure 1: Basic ML schema

2. Methods

Dataset:



	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10	Attr11	Attr12
0	0.200550	0.37951	0.39641	2.0472	32.3510	0.38825	0.249760	1.33050	1.1389	0.50494	0.249760	0.65980
1	0.209120	0.49988	0.47225	1.9447	14.7860	0.00000	0.258340	0.99601	1.6996	0.49788	0.261140	0.51680
2	0.248660	0.69592	0.26713	1.5548	-1.1523	0.00000	0.309060	0.43695	1.3090	0.30408	0.312580	0.64184

Table 1

The data set is Polish companies' bankruptcy data, which contains 43,405 observations and 65 va riables as shown in table 1 above. The first 64 variables are the predictors, including all sorts of companies' current financial status and properties such as net profit, total liabilities and working capital against sales, remaining earnings as well as total assets. The 65th variable is the response, with values b'1' and b'0' reflecting whether the company will be bankrupt the next year. Upon visualizing the target features, (figure 2) we noticed unequal distribution of the bankrupt and not bankrupt classes. Only 2,091out of 41,314 observations are bankrupt, indicating the data is highly imbalanced which is quite common and always a challenge in classification problems. As with most machine learning algorithms, uneven distribution of class ratio in classifiers could lead to an inaccurate estimate of class prior which could then potentially decrease the predictive performance. Moreover, there are 5835 missing values in the predictors, so an imputation techni que should be performed.

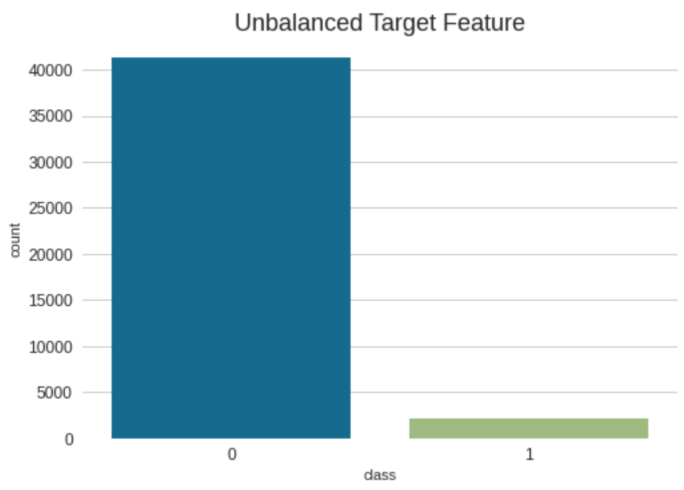


Figure 2

Data Exploration: One of the observations we had is that there was significant bankruptcy in the 5th year data. We are not sure if the collected dataset was random or if it included almost all companies from a particular category. Since we are not sure about how the dataset was collected, we decided not to consider the year factor as one of the reasons of bankruptcy. Ex: in 2020, a lot of small companies filed bankruptcy because they went through heavy loss because of coronavirus. Now, we know that it happened because of a total unrelated external agent. For handling null values, we did impute the missing data with median.

Model Assumption: Here in this case study we are dealing with mostly ensemble algorithms. Ensemble algorithms are like meta-algorithm that combine many other small ML techniques such as decision trees into one big predictive model to decrease variance and bias and improve prediction. Hence, unlike the linear models, ensemble model do not have any separate assumption of their own. One key focus of these algorithm is to make sure we have tuned parameters to fit the model.

Train Test Split: The dataset is split into a training and test set (80/20 split) to keep test data separate and to perform modeling only on the training set. Later, we will use the test set to test the model's general performance.

ML model pipelines: We will follow ML pipeline to build the classification model. Figure 3.

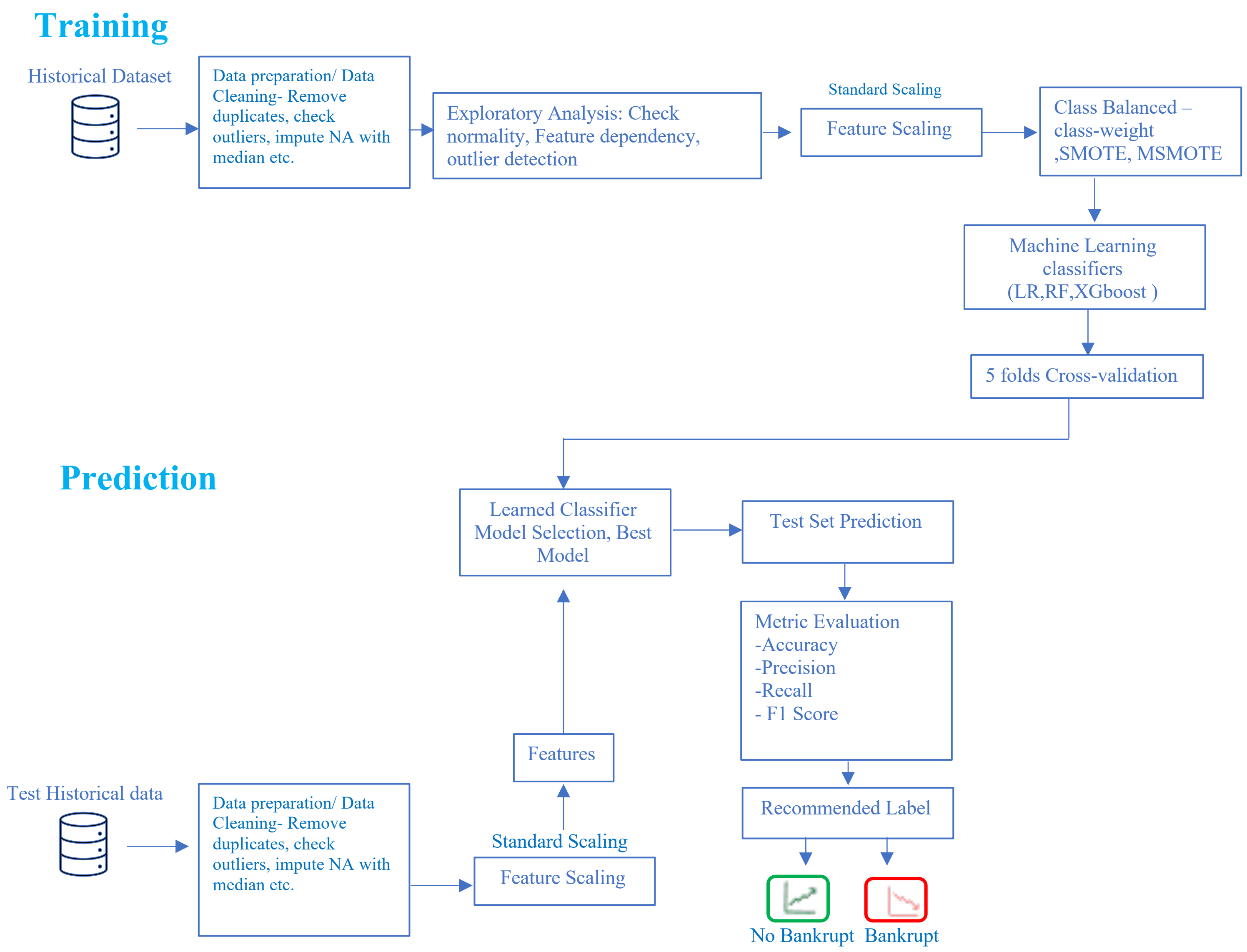

Training
Historical Dataset
Data preparation/ Data Cleaning- Remove duplicates, check outliers, impute NA with median etc.
Exploratory Analysis: Check normality, Feature dependency, outlier detection
Standard Scaling
Feature Scaling
Class Balanced – class-weight ,SMOTE, MSMOTE
Machine Learning classifiers (LR,RF,XGboost)
5 folds Cross-validation
Prediction
Learned Classifier Model Selection, Best Model
Test Set Prediction
Metric Evaluation -Accuracy -Precision -Recall - F1 Score
Features
Recommended Label
Test Historical data
Data preparation/ Data Cleaning- Remove duplicates, check outliers, impute NA with median etc.
Standard Scaling
Feature Scaling
No Bankrupt
Bankrupt

Figure 3: ML Flowchart

Imbalance target features: The dataset has imbalanced target features. Modeling on imbalanced target features would give evaluation metrics that are not accurate. The key concept to getting an accurate predictive model is to balance the data before fitting the predictive model. Hence, we will implement two approaches, first we will use an algorithm with parameters, class-weight = 'balanced'. Next, we will also build another models using the SMOTE technique to balance data.

Cross-Validation: We will use StratifiedKFold cross-validation with k=5 because with StratifiedKFold, the class distribution in the dataset is preserved in the training and test splits. Shuffle is set to true so that the splitting will be random.

Decision Trees:
Decision trees (DT) are a method to classify data into segments using a yes/no logic. They are formed from partition trees which, even though not used directly, form the basis of all other algorithms such as Random Forest and XGBoost, which we will implement to solve the bankruptcy problems. Contrary to linear algorithms which uses a line to separate targets, partition trees are used to solve the nonlinear problems and can be decreased or increased in any direction. Decision trees work by partitioning data into bins. For instance, as shown in figure 4 below, if a value in total sales > 5M place it in left bin else place in right bin (figure 4) What happens is, we go through the data and make decisions based on the data. The decision is made based on the side that has the maximum information gain, which is calculated using two different techniques, Gini or entropy. Either Gini or entropy can be used, however, based on the type of data and problem we are trying to solve, we can pick the one that gives better performance metrics. We repeat the process of splitting until we reach the stopping criteria, which can be as simple or complex as we want. We can have a set number of decisions such as continuing until five decisions or 10 decisions which can be set using parameter max depth.

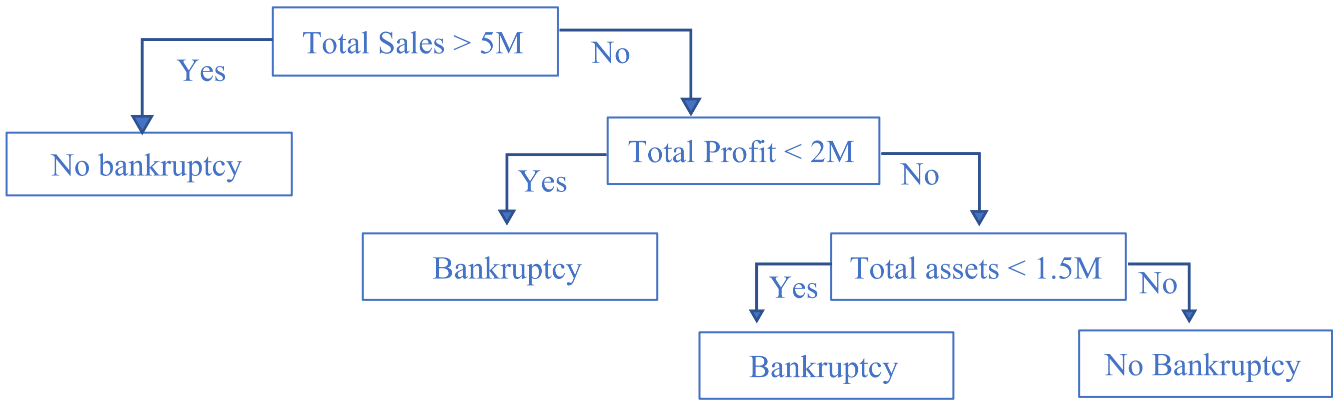


Figure 4: Decision Tree

Random Forest: One major drawback of DT is that, as the number of data and depth of the trees increase, they are more prone to overfit. This means that the model might work really well with training data but will perform poorly on the test data. To overcome this issue, we use Random Forest (RF), which is an ensemble model that fits a number of DT classifiers on various sub-samples of the dataset and uses an average of all DT to improve the model performance and checks the over-fitting issues (Figure 5). Subsamples of datasets in RF are done with replacement called bootstrapping. Also with RF classifiers, all the trees are trained in parallel (bagging model / bootstrap aggregation). The key concept in RF algorithms is that by randomly picking the data and selecting features at each split, we will have each tree be uncorrelated with other trees. Hence, due to this randomness, the bias of the forest slightly increases. However, due to the averaging of all DT, the overall variance of the RF model decreases significantly as compared to increase in bias, hence RF give us an overall better model. Random forest, while on the surface, seeming to be a simple algorithm, is really one of the best initial guesses that we can find in modern data science.

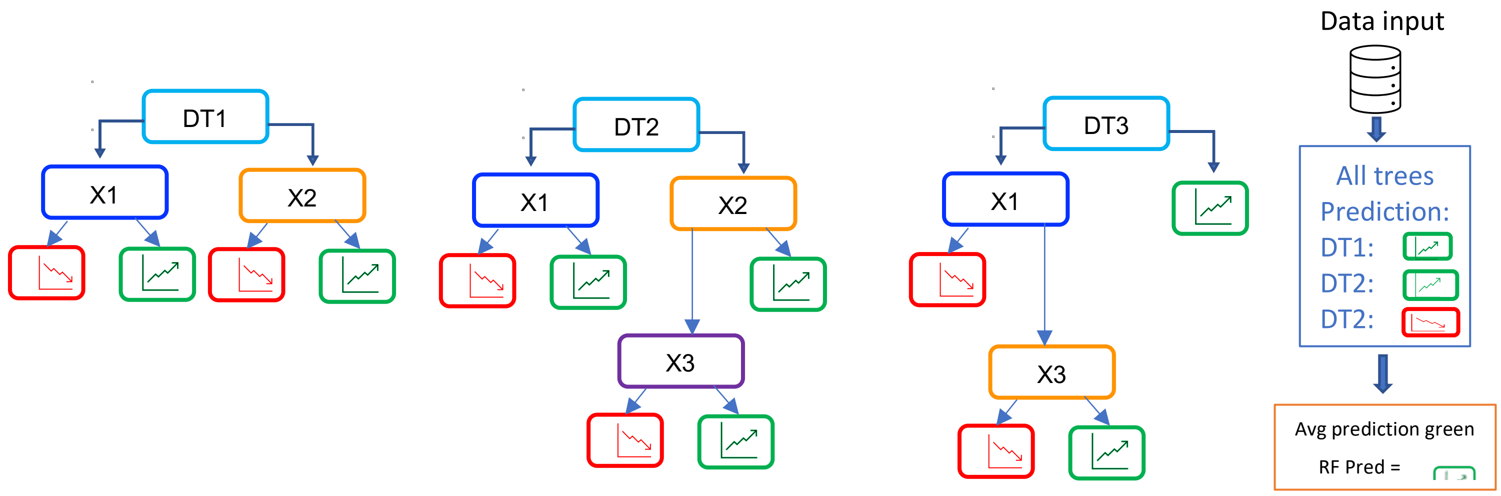


Figure 5: Random Forest

XGBoost Classification: XGBoost (extreme Gradient Boosting) relies on the concept of boosting. Boosting is a way to make weak learners produce a strong learner by iteratively fitting on the residual (sequential manner) from the previous round instead of the target until it reaches the point where the residual is normally distributed. The key advantage of XGBoost is its high-performance speed compared to other models and its regularization parameter (figure 6) that successfully reduces the variance. Besides speeds and regularization, XGBoost also leverages a learning rate (shrinkage) and subsamples from features like RF, which increases its ability to generalize even further. Another concept that makes XGBoost so powerful is that it uses an approximate loss (use any loss function), so instead of actually going through the calculus and finding specific updates rule, XGBoost approximate that loss.

$$J = \sum_i \overset{\text{General loss function}}{\boxed{l(p_i, y_i)}} + \overset{\text{Penalties}}{\boxed{\sum_k \Omega(f_k)}}$$

$$\Omega(f) = \gamma T + \tfrac{1}{2}\lambda||w||^2$$

T= Nodes and leaves and w = Slope

Figure 6: XGBoost loss function

With XGBoost instead of penalizing slopes we penalize trees. The above penalty term makes the tree to be smaller as the number of trees increases. Also, we can notice that not only is slope w score involved in the equation, it's also the second order of the score, hence making it analogous to L2 regularization.

Feature Importance: Knowing which features could possibly lead to company bankruptcy might be of useful information for a company to make better decision as to where they need to focus more. Hence, using RF classifier in the figure below, we have shown the top 10 importance features (Figure 7). We can see that feature Attr37 which corresponds to its exact name (current assets – inventories) / long-term liabilities from data dictionary is the most importance feature followed by Attr27, which stands for profit on operating activities / financial expenses.

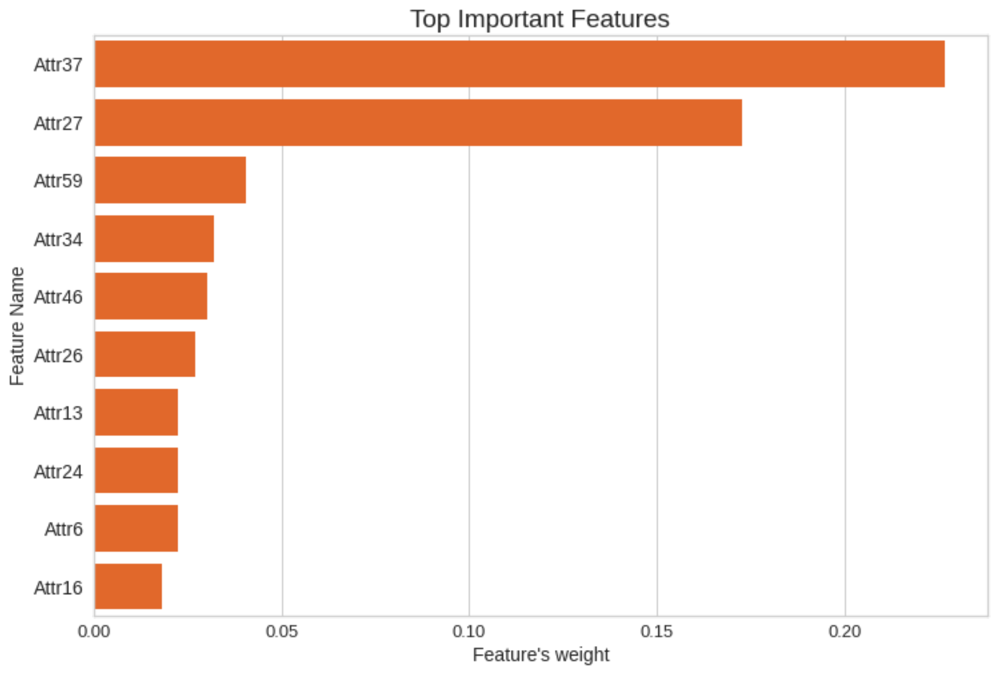


Figure 7: RF Top Features

3. Result:

We could use any classifiers to build models to classify bankruptcy. However, the main objective of this case study is to show that XGBoost performs better than other classifiers such as logistic regression, decision tree, and random forest. First, we begin with building a simple logistic regression (LR) model. The figure below (figure 8) shows that the LR model performs poorly for this problem with a mean accuracy of 67%. For the next few ensemble models, to get better and more generalized scores, we will perform parameter tuning using both the RandomSearch technique and GridSearch. We have noticed that RandomSearch tends to run much faster than GridSearch and the difference in metric between the two is not large. Hence, using the best parameters we built an RF model as our second classifier. The figure below (figure 8) shows that the RF model does much better than the LR model with a mean accuracy of 97.4%.

As our goal is to build a model that outperforms the RF model, next, we built the XGBoost model. We can see that based on the accuracy score the XGBoost model does the best classification.

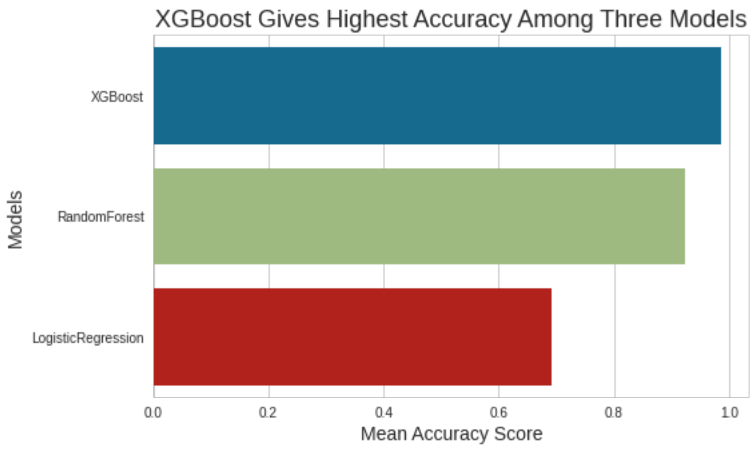


Figure 8: Model Comparison

The dataset we have used to build the model is not a balanced dataset. Relying solely on accuracy score can be misleading. To further compare XGBoost with the RF model we used other model validation metrics such as, precision, recall, ROC, and F1 score. The AUC-ROC shows how much the model is capable of distinguishing between classes. The higher the AUC, the better the model is at predicting no bankruptcy classes as 0 and bankruptcy classes as 1. Since the LR model is our base model, other validation metrics are not shown in this paper. From the confusion matrix and ROC curve in the figure below (figure 9,10) we can see that XGBoost gives a better score for ROC and F1-score compared to RF, while the AUC-ROC score for both RF and XGBoost are quite similar. In term of F1-score, which is a combination of precision and recall and conveys the balance between those two, shows that XGBoost outperforms RF.

Model 2:

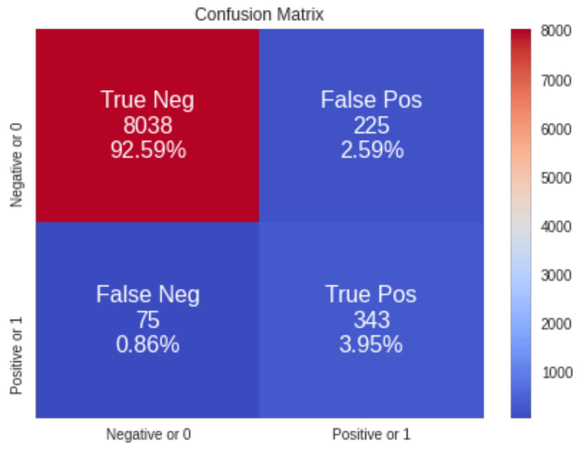


Figure 9: Random Forest confusion matrix

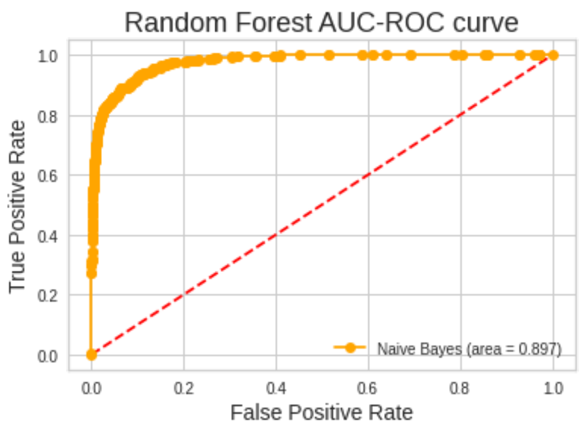
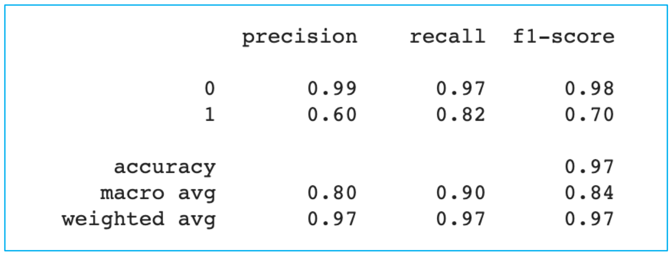


Figure 10: Random Forest ROC curve



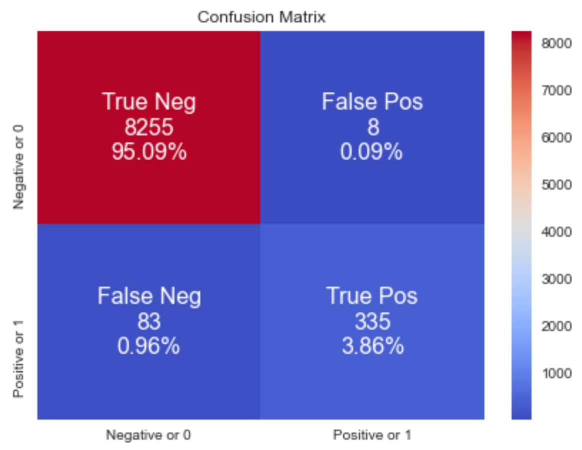
```
              precision    recall  f1-score

           0       0.99      0.97      0.98
           1       0.60      0.82      0.70

    accuracy                           0.97
   macro avg       0.80      0.90      0.84
weighted avg       0.97      0.97      0.97
```

Figure 11. Random Forest Classification

Model 3:



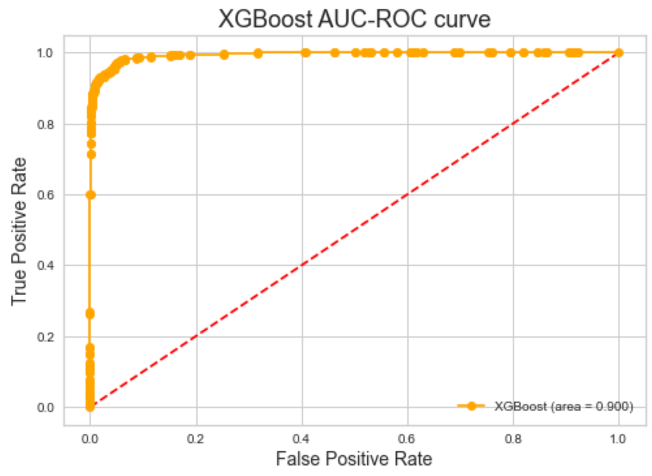
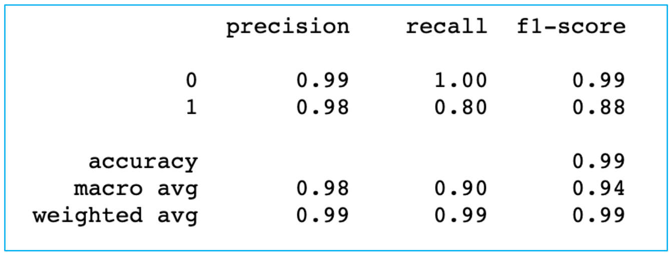


Figure 12: XGboost Confusion matrix

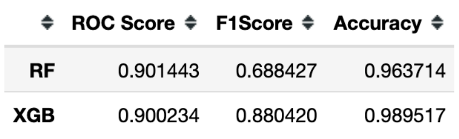
Figure 13. XGboost ROC Curve



```
              precision    recall  f1-score

           0       0.99      1.00      0.99
           1       0.98      0.80      0.88

    accuracy                           0.99
   macro avg       0.98      0.90      0.94
weighted avg       0.99      0.99      0.99
```



	ROC Score	F1Score	Accuracy
RF	0.901443	0.688427	0.963714
XGB	0.900234	0.880420	0.989517

Table 2.

Figure 14: XGboost classification Report

4. Conclusion:

We built three models in this case study: Logistic Regression, Random Forest and XGBoost. We used Logistic Regression for a benchmark comparison only but did extensive comparisons between RandomForest and XGBoost for performance. We observed that XGBoost is faster than RandomForest., but further analysis on this can be done with a higher volume of data to do the comparison more intensely.