Following on from logistic regression, decision tree models were another avenue that was explored. Firstly, owing to the tabular nature of the data, they were determined as a particularly amenable approach. Decision trees have demonstrated their efficacy in this domain, even beating the likes of more advanced models such as neural networks [1]. They are also suitable because they can cope with the presence of superfluous and unhelpful features [2], which are a concern due to the substantial number of features present. There are also mechanisms such as weighting that can be used to address the class imbalance that is present in our data [3]. These considerations led to the choice of the following 3 decision tree models: random forests [4], XGBoost [5] and AdaBoost [6]. These are all decision tree ensemble models that should lead to more robust outcomes [7].

Random forests operate on the premise of creating a forest of multiple decision trees that are trained on randomly selected data subsets and feature subsets at each splitting rule.

get\_ranger\_grid = *function*(feat\_sel) {

*if* (feat\_sel) {

*return* (

expand.grid(*mtry* = c(3, 12, 29, 51),

*min.node.size*=c(1, 3, 5, 7, 9),

*splitrule*=c("gini", "extratrees", "hellinger")

)

)

}

*else* {

*return* (

expand.grid(*mtry* = c(3, 12, 29, 47, 73, 91),

*min.node.size*=c(1, 3, 5, 7, 9),

*splitrule*=c("gini", "extratrees", "hellinger")

)

)

}

}

fit\_ranger = *function*(data, *feat\_sel*=FALSE) {

*rangerGrid* = get\_ranger\_grid(feat\_sel)

*return* (

train(

*x*=data[, names(data) != 'Bankrupt'],

*y*=data[["Bankrupt"]], *method*="ranger",

*metric*="ROC", *trControl*=cv5foldCtrl, *tuneGrid*=rangerGrid,

*num.trees*=50, *max.depth*=3, *regularization.usedepth* = TRUE,

*weights*=get\_sample\_bankrupt\_weights(data), *class.weights*=get\_bankrupt\_class\_weights(data)

)

)

}

*# assumes balanced data input*

fit\_ranger\_on\_balanced = *function*(data, *feat\_sel*=FALSE) {

*rangerGrid* = get\_ranger\_grid(feat\_sel)

*return* (

train(

*x*=data[, names(data) != 'Bankrupt'],

*y*=data[["Bankrupt"]], *method*="ranger",

*metric*="ROC", *trControl*=cv5foldCtrl, *tuneGrid*=rangerGrid,

*num.trees*=50, *max.depth*=3, *regularization.usedepth* = TRUE

)

)

}

XGBoost operates by fitting successive decision trees that correct the residuals of the previously fitted trees.

xgbmGrid <- expand.grid(*max\_depth* = c(3, 6, 9),

*nrounds* = 100, *# number of trees*

*eta* = 0.3,

*gamma* = c(5, 15),

*subsample* = 0.65,

*colsample\_bytree* = 0.7,

*min\_child\_weight* = c(1, 3)

)

*# training a XGboost tree model while tuning parameters*

fit\_xgbm = *function*(data) {

*return* (

train(

*x*=data[, names(data) != 'Bankrupt'],

*y*=data[["Bankrupt"]],

*method* = "xgbTree", *metric*="ROC",

*trControl* = cv5foldCtrl, *tuneGrid* = xgbmGrid,

*scale\_pos\_weight*=get\_bankrupt\_weight(data)

)

)

}

fit\_xgbm\_on\_balanced = *function*(data) {

*return* (

train(

*x*=data[, names(data) != 'Bankrupt'],

*y*=data[["Bankrupt"]],

*method* = "xgbTree", *metric*="ROC",

*trControl* = cv5foldCtrl, *tuneGrid* = xgbmGrid

)

)

}

AdaBoost operates by fitting successive decision trees that focus on the previous misclassified or hard-to-classify examples.

adbGrid <- expand.grid(

*mfinal*= c(50, 100),

*maxdepth*= c(20, 30)

)

*# training an AdaBoost tree model while tuning parameters*

fit\_adb = *function*(data) {

*return* (

train(

*x*=data[, names(data) != 'Bankrupt'],

*y*=data[["Bankrupt"]],

*method* = "AdaBag", *metric*="ROC",

*trControl* = cv5foldCtrl, *tuneGrid* = adbGrid,

*weights*=get\_sample\_bankrupt\_weights(data)

)

)

}

fit\_adb\_on\_balanced = *function*(data) {

*return* (

train(

*x*=data[, names(data) != 'Bankrupt'],

*y*=data[["Bankrupt"]],

*method* = "AdaBag", *metric*="ROC",

*trControl* = cv5foldCtrl, *tuneGrid* = adbGrid

)

)

}

The training was completed with 5-fold cross-validation that focused on optimising ROC, following from the same motivations of logistic regression. Hyperparameters were set and tuned with the aid of this cross-validation.

*# Create 5 fold cv, sensitivity, specificity, ROC control*

cv5foldCtrl <- trainControl(*method*="cv", *number*=5, *classProbs* = TRUE, *summaryFunction* = twoClassSummary, *allowParallel* = TRUE)

Notably, this involved structural and regularisation hyperparameters to combat overfitting. For example, with the numbers of trees/rounds, tree maximum depth and depth regularisation. This can be seen in the hyperparameter tuning grids above.

The ROC plots of the trained models, based on the probability threshold for predicting bankruptcy, are depicted below. Thus, the desired model can be chosen along the ROC curve based on the trade-off between sensitivity and specificity. The AUC results from these ROC curves are depicted below. The confusion matrices of the models, at the optimal point on the ROC curve, are also depicted below.

A graph of a number and a line

Description automatically generated

A graph of a graph

Description automatically generated A graph showing the growth of a forest

Description automatically generatedA graph of a graph

Description automatically generated

A white background with black text

Description automatically generated

A white paper with black text

Description automatically generated

A white background with black text

Description automatically generated A white screen with black text

Description automatically generated

A graph of a curve

Description automatically generated

A graph of a number of data

Description automatically generated with medium confidence

A graph showing the number of data

Description automatically generated with medium confidence A graph of a curve

Description automatically generated

A white background with black text

Description automatically generated A white background with black text

Description automatically generated A white background with black text

Description automatically generated A screenshot of a white screen

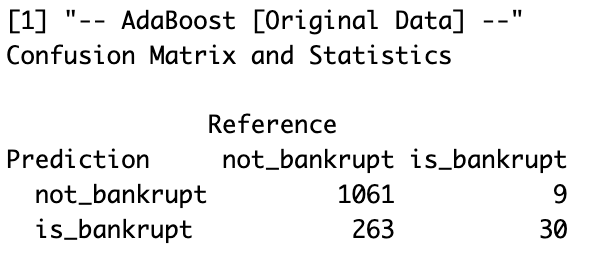
Description automatically generated

A graph showing the growth of a number

Description automatically generated A graph showing the growth of a stock market

Description automatically generated A graph showing the growth of a stock market

Description automatically generated A graph showing a line

Description automatically generated  A white background with black text

Description automatically generated A white background with black text

Description automatically generated A white background with black text

Description automatically generated

From the results, we can see that the random forest model performed the best with an AUC of 0.935, on the balanced dataset, which outperforms the prior logistic regression model and positions it as the optimal solution to the classification task that was developed. This is in line with expectations as decision trees introduce feature interactions and non-linearity. This is evident as feature-splitting rules along branches are conditioned on the ancestor feature-splitting rules. Thus, decision trees can capture greater complexity compared to linear models that assign coefficients to individual features and therefore don’t account for their interactions.

Another point of interest is that the usage of balanced datasets improved performance compared to the application of weighting on the original datasets. This highlights that the up-sampling methods used were indeed productive in this case.

# Bibliography

|  |  |
| --- | --- |
| [1] | L. Grinsztajn, E. Oyallon and G. Varoquaux, “Why do tree-based models still outperform deep learning on tabular data?,” 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2207.08815. [Accessed 20 04 2024]. |
| [2] | T. Ibrikci, E. M. Karabulut and S. A. Ozel, “A comparative study on the effect of feature selection on classification accuracy,” *Procedia Technology,* vol. 1, pp. 323-327, 2012. |
| [3] | P. Branco, L. Torgo and R. P. Ribeiro, “A Survey of Predictive Modeling on Imbalanced Domains,” *ACM Comput. Surv.,* vol. 49, no. 2, p. 50, 2016. |
| [4] | L. Breiman, “Random forests,” *Machine learning,* vol. 45, pp. 5-32, 2001. |
| [5] | T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016. |
| [6] | Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of computer and system sciences,* vol. 55, no. 1, pp. 119-139, 1997. |
| [7] | L. Breiman, “Bagging predictors,” *Machine learning,* vol. 24, pp. 123-140, 1996. |