Financial Ratios as Predictors of Company Bankruptcy: A Predictive Model Approach

Technical Report, Logbook and Reflective Discussion

Name: Muhammad Muneeb Ullah Ansari

Student ID: 40426685

Word Count: 2,748

Technical Report submitted in part fulfilment of the degree of Master of Science in Business Analytics

September 2024

Queen's Business School

Table of Contents

1- Flowchart 3

2- Background 3

3- Outlier Analysis 4

4- Feature Selection 5

5- Outlier Removal 11

6- Sampling Methods 21

6.1- SMOTE (Synthetic Minority Oversampling Technique) 21

6.2- ADASYN (Adaptive Synthetic Sampling) 21

6.3- Random Over Sampling 22

6.4- Random Under Sampling 22

7- Predictive Modeling 22

7.1- MDA (Multiple Discriminant Analysis) 22

7.2- SVM (Support Vector Machines) 23

7.3- KNN (K-Nearest Neighbor) 23

7.4- LR (Logistic Regression) 24

7.5- RF (Random Forest) 24

8- Evaluation Metrics 24

9- Interpretable Machine Learning 26

9.1- Partial Dependency Plots (PDP) 26

9.2- Accumulated Local Effects (ALE) 26

10- Logbook 27

11- Reflective Discussion 29

References 30

Appendix 34

Variables renamed to letters 34

R Code 40

1- Flowchart

The following is a flowchart that shows the major technical tasks carried out. This was in accordance with CRISP-DM, refer “Methodology” in main report.

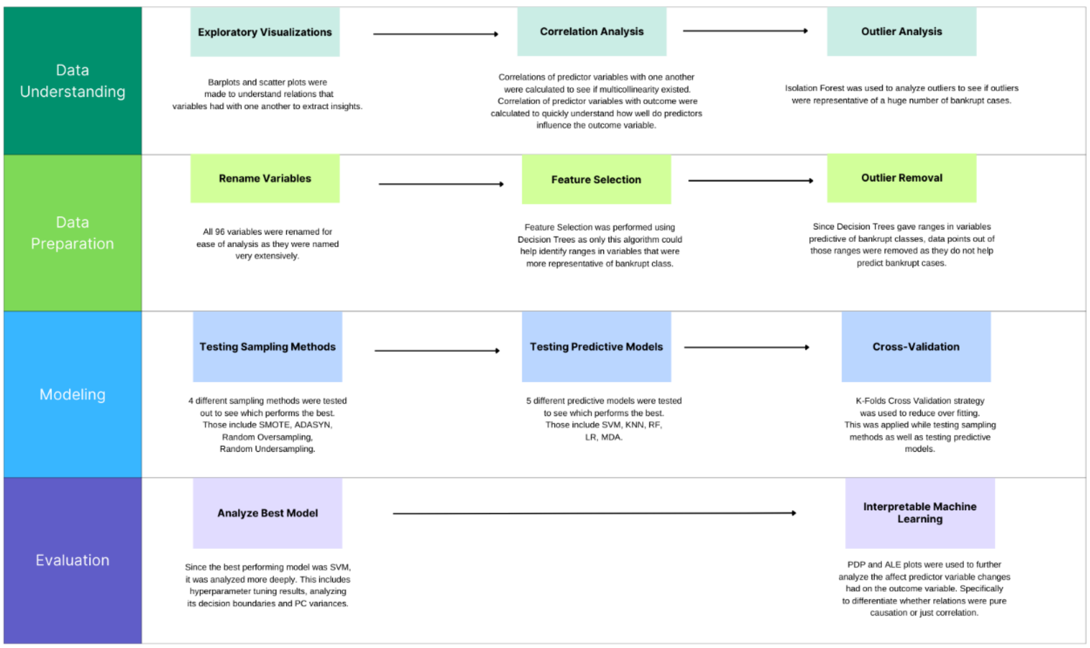


Figure 1: Technical Flowchart

2- Background

We have used RStudio throughout the project and used KNIME at one instance to perform feature selection. RStudio aids syntax highlighting, code completion, and the ability to manage multiple projects simultaneously, which significantly improves workflow management (Xie, 2014). The drag and drop feature and ease of interpretation for KNIME (Dwivedi, Kasliwal and Soni, 2016) makes it ideal to use for a quick output.

The CRAN packages that we utilized were as follows.

1- dplyr: Data manipulation
2- ggplot2: Data visualization
3- tidyr: Data manipulation
4- writexl: Save cleaned dataset
5- readxl: Read cleaned dataset
6- caret: Create predictive models
7- ROSE: Sampling methods (Random over and under sampling)
8- pROC: Creating ROC curves
9- themis: Sampling methods (SMOTE and ADASYN)
10- recipes: Cross validation for sampling methods
11- solitude: Anomaly detection using Isolation Forest
12- mda: MDA (Multivariate Discriminant Analysis) Model
13- e1071: SVM (Support Vector Machines) Model
14- iml: PDPs (Partial Dependency Plots) and ALE (Accumulated Local Effects)

3- Outlier Analysis

We performed Isolation Forest to analyze present outliers. We use Isolation Forest particularly because it is robust to noise in the data and works better in scenarios involving large, high-dimensional datasets, or where feature scaling is a concern (Al Farizi, Hidayah and Rizal, 2021).

4- Feature Selection

We decide to use the Decision Tree Algorithm as a feature selection technique and for dealing with some outliers in the data. In a study done by Gayatri et al. (2010), the decision tree algorithm was also used as a feature selection technique to detect defects in software. In another study done by Wang and Li (2008), this technique was used to test feature selection on hyperspectral data. For more details, refer to the studies by Li (2008) and Gayatri et al. (2010). The decision tree's use fit our study as follows.

1. Performance of Decision Trees is robust to outliers and our data contains outliers at this stage.
2. Algorithm automatically removes unimportant variables or variables with high multicollinearity which we have in our dataset.
3. Splits in the Decision Trees are indicative of ranges of predictor variables that will affect the target in certain ways. This is critical to identifying some outliers to be removed.

For Decision Trees, we decided to utilize KNIME Analytics platform as Decision Trees made on KNIME are much better visualized and interactive. At first the algorithm selected 18 of the total 95 predictor variables in the dataset. A model with an 81% predictive accuracy is considered good and is implementable.

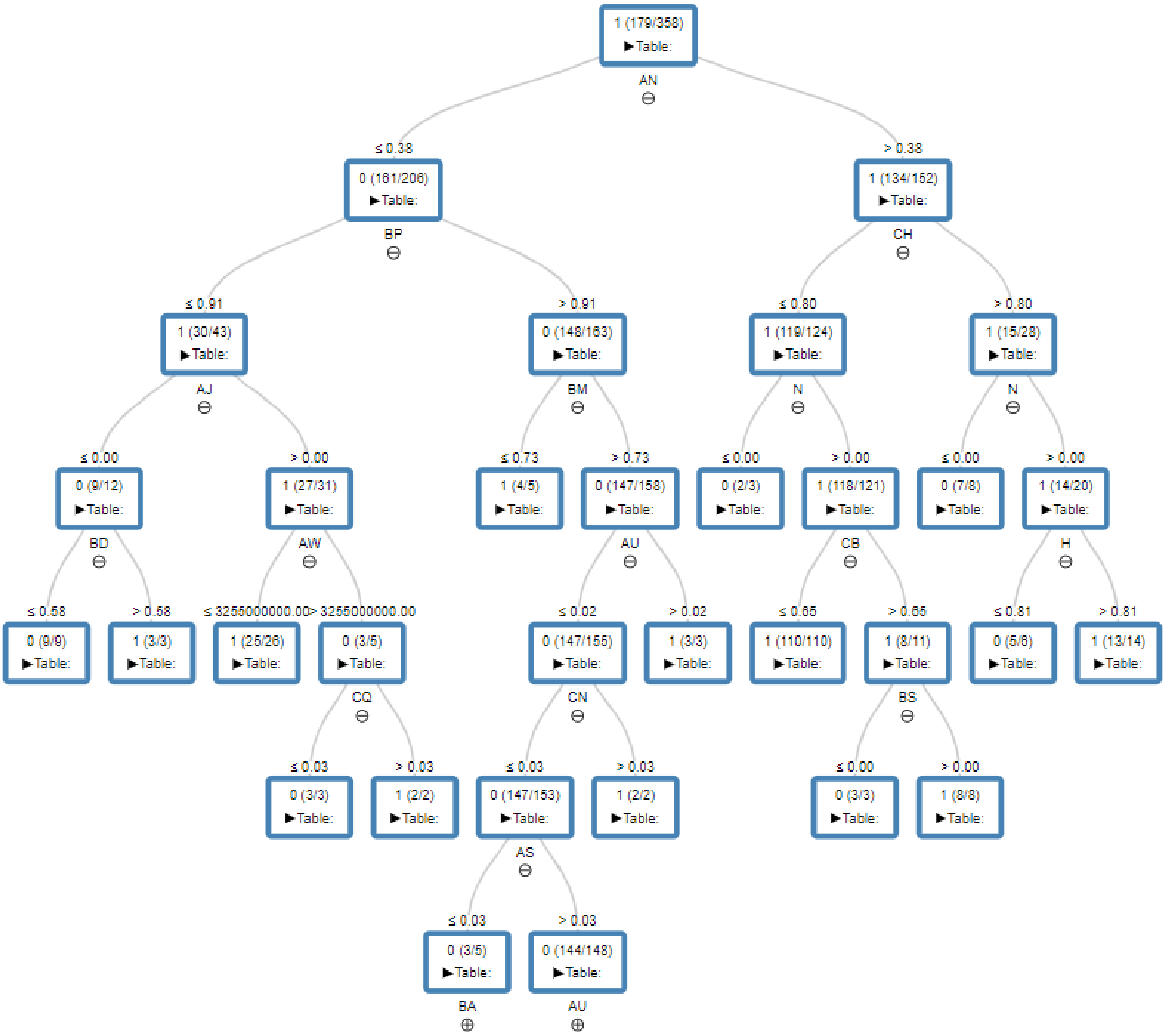


Figure 2: Decision Tree

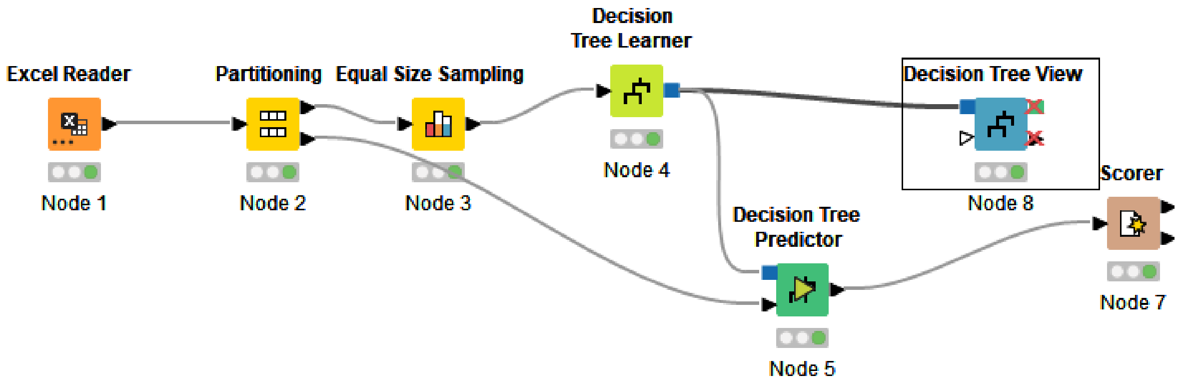


Figure 3: KNIME Workflow

Prediction (bankrupt) \ bankrupt	1	0
1	34	253
0	7	1070
Correct classified: 1,104	Wrong classified: 260	
Accuracy: 80.938%	Error: 19.062%	
Cohen's kappa (κ): 0.163%		

Figure 4: Accuracy Statistics for the Decision Tree

After the 18 variables were selected, we manually observed the decision tree to further remove certain variables. We aimed to keep only variables that were predictive of the bankrupt class or "1". We take a novel approach for this task: For each variable, we looked at the following 3 measures.

1. The **<u>Range</u>** of values with respect to the split point in the Decision Tree Node that predicts the bankrupt class. (For example: In a node, variable "Borrowing Dependency" is split by values >0.38 and <0.38 which are the two ranges. Out of these two ranges, >0.38 has majority predictions of bankrupt. Hence >0.38 will be the selected range.)

2. The **<u>Percentage Purity</u>** from the Decision Tree Node of the bankrupt class within the range that predicts it. (For example: Variable “Borrowing Dependency” has a selected range of >0.38. In this range, the percentage purity of the bankrupt class is 88.2% which is a large chunk.)
3. The **<u>Count of Observations</u>** in the original dataset (out of the total 6819 observations) for the selected range. (For example: For variable “Borrowing Dependency”, there are 788 observations in the original dataset that have a value of >0.38.)

Hence as an example, for the variable “Borrowing Dependency”, we can say that there are 788 observations in the original dataset that have 88.2% values bankrupt. Those observations can be identified by having a “Borrowing Dependency” greater than 0.38.

We repeated this process for each of the 18 selected variables and kept only the variables that had higher percentage purities with considerable count of observations. The following is a table showing variables selected and the relevant measures for them.

Variable	Range	Percentage Purity	Count of Observations
Borrowing Dependency	>0.38	88.2%	788
Net Income / Total Assets	<= 0.8 and >0.8	96% and 53.6%	2116 and 4703
Interest Bearing Debt Interest Rate	>0.00 and <=0	97.5% and 70%	5928 and 891
Total Debt / Total Net Worth	>0.00	87.1%	6818
Cash Flow / Total Assets	<=0.65 and >0.65	100% and 72.7%	4089 and 2730
Fixed Asset Turnover Frequency	<=3255e6	96.2%	6819
Current Assets / Total Assets	>0.58	100%	2703
Current Asset Turnover Rate	>0.00	100%	5999
Equity / Liability	>0.03	100%	4030
Net Income / Stockholder Equity	>0.84	100%	5297

Figure 5: Selection Criteria for Variables

The following are the formulas of the variables not already implied.

$$Borrowing\ Dependency = \frac{Total\ Debt}{Total\ Assets}$$

$$Interest\ Bearing\ Debt\ Interest\ Rate = \frac{Total\ Interest\ Expense}{Total\ Interest\ Bearing\ Dent}$$

$$Fixed\ Asset\ Turnover\ Frequency = \frac{Net\ Sales}{Average\ Fixed\ Assets}$$

$$Current\ Asset\ Turnover\ Frequency = \frac{Net\ Sales}{Average\ Current\ Assets}$$

Variables higher up in the table imply that they are more significant at predicting the output variable. This is due to them being split higher in the decision tree than other variables. However with this technique, we cannot numerically quantify how significant the variable is at predicting the outcome.

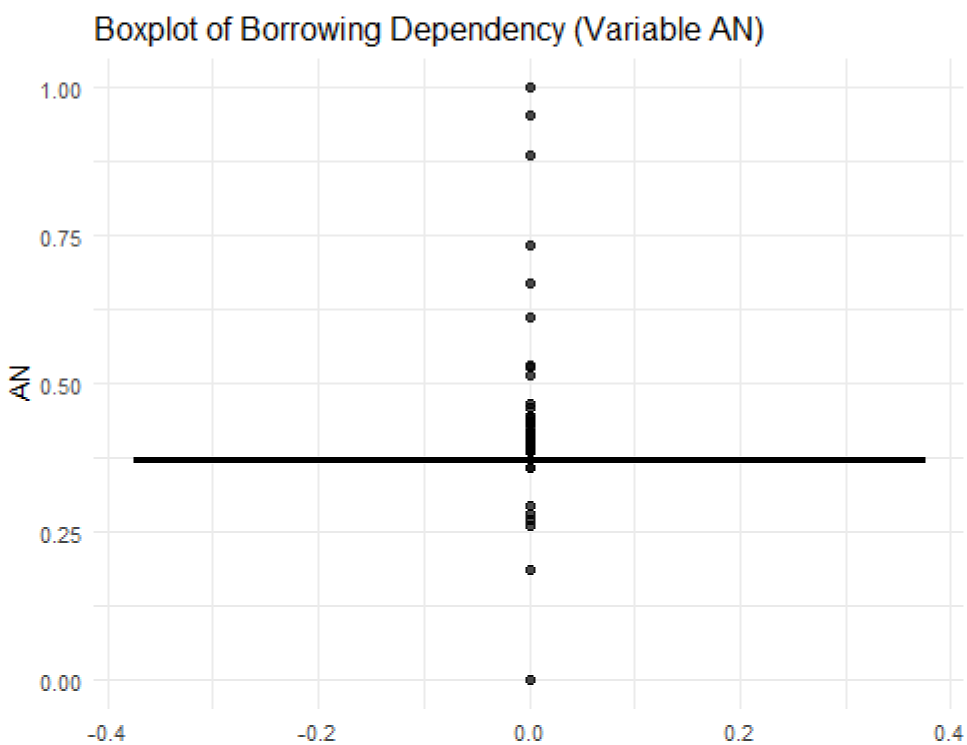
Some variables have two ranges, purities and counts with them. This is because at both sides of the split point of the node, the percentage purity of the bankrupt class was in majority.

Splits were made at 0.00 by the algorithm for 3 variables. Split was made at 3255e6 for 1 variable. This is because of very extreme outliers present in those variables. Even though such outliers effected the point of splits, they did not adversely affect the variable's importance for predicting the positive class.

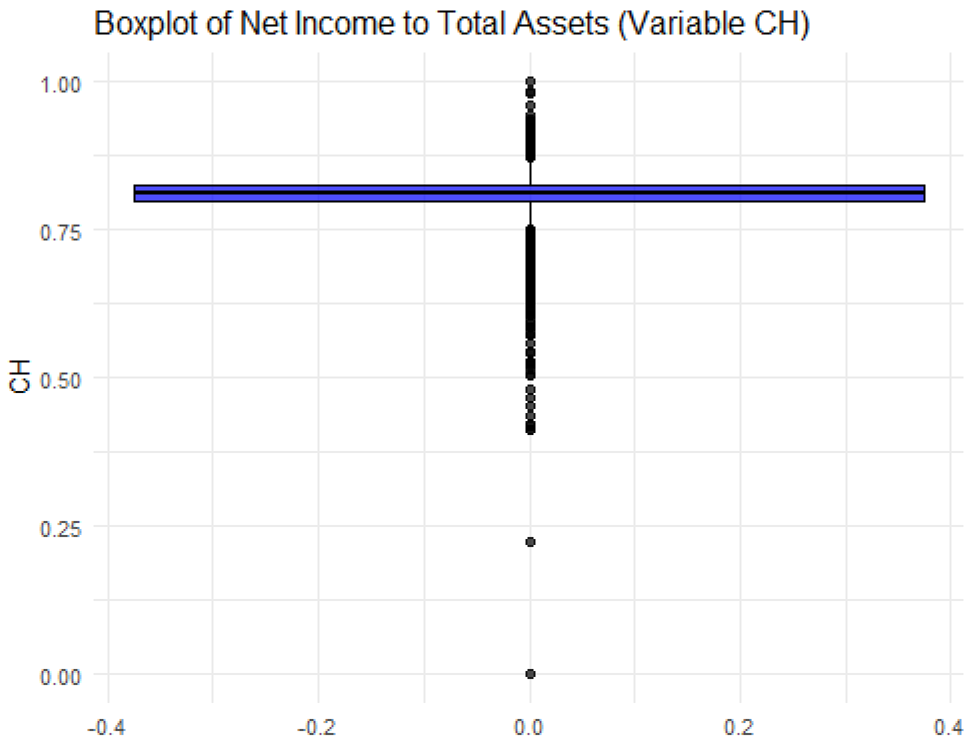
5- Outlier Removal

The second step was to keep outliers that are predictive of the bankrupt class. The use of decision tree as a method for removing outlier is also explored in the study by John (1995). He mentions that a lot of advanced machine learning algorithms address the issue of removing outliers but not fully, other statistical methods (like decision trees) can help to address the problem more directly. Hence he prunes the data using decision tree nodes as reference to remove outliers.

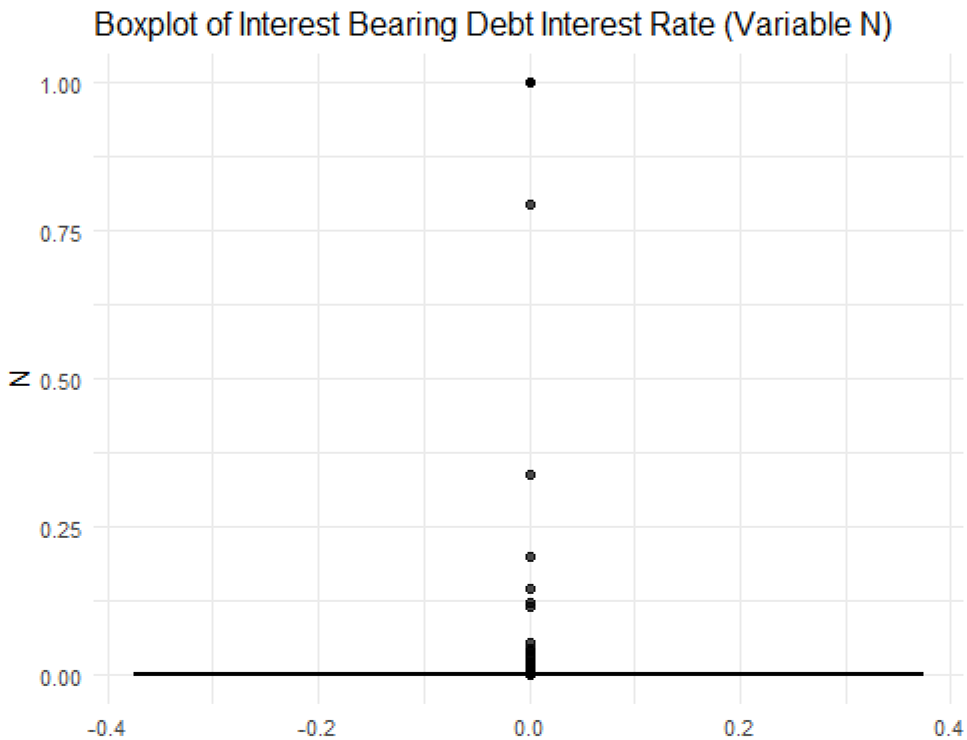
For that we made a boxplot and observed the number of observations in the range that have a higher percentage purity of bankruptcy. (For example: For “Borrowing Dependency”, bankrupt cases are more likely to be where values >0.38. The boxplot shows that values <0.38 are not many so they can be removed and we will not be losing critical information about bankruptcy.)



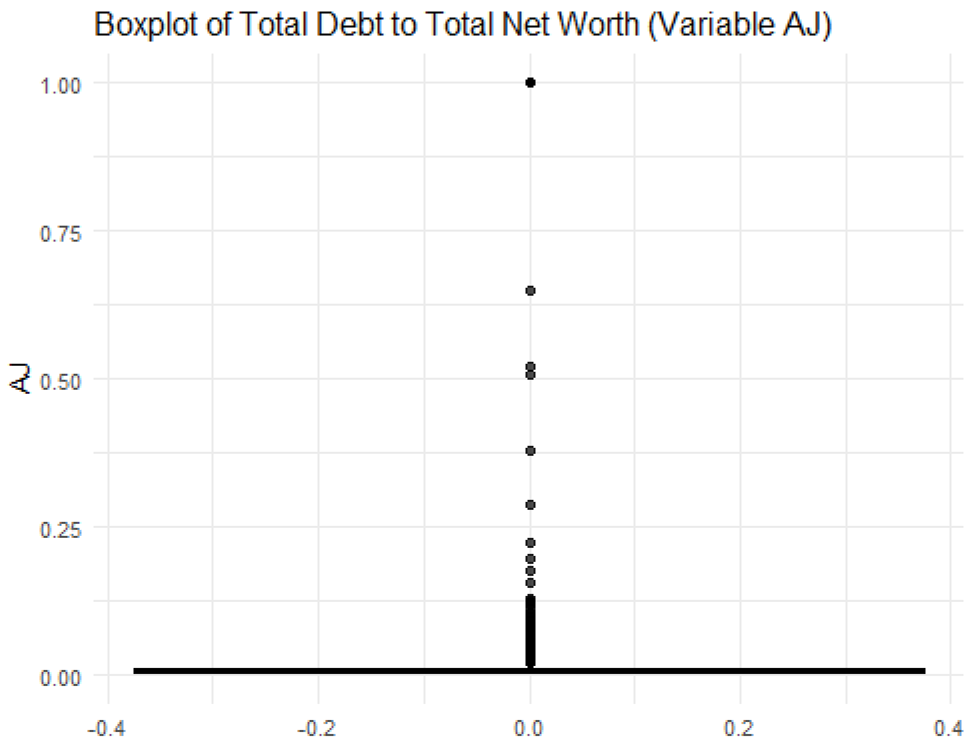
The following process was repeated for each variable.



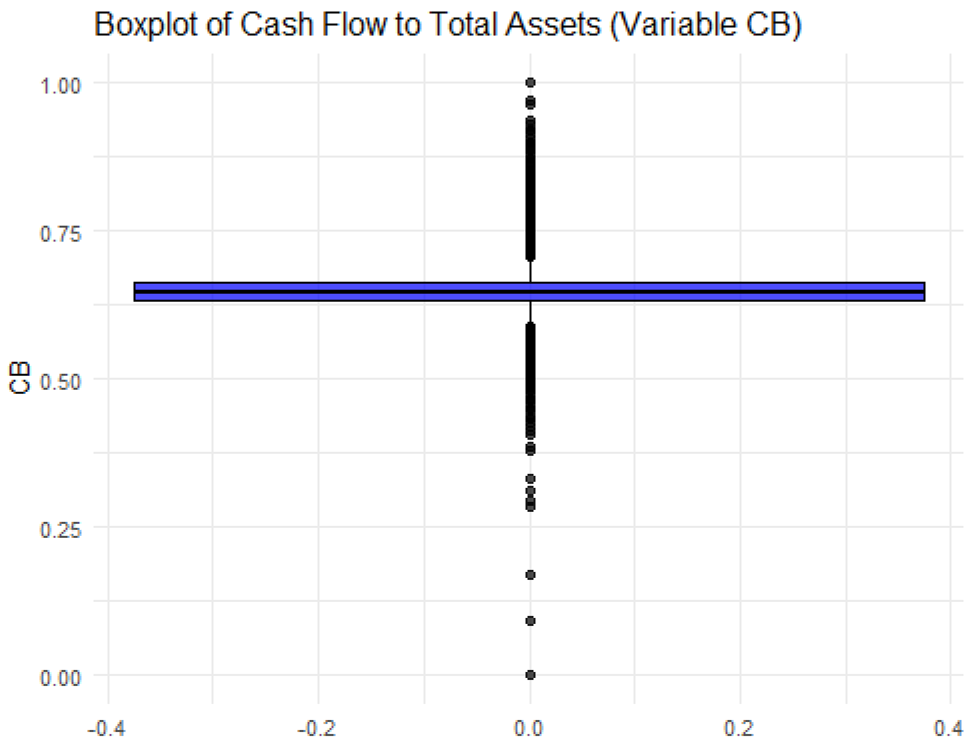
For Net Income to Total Assets, values above 0.08 were removed.



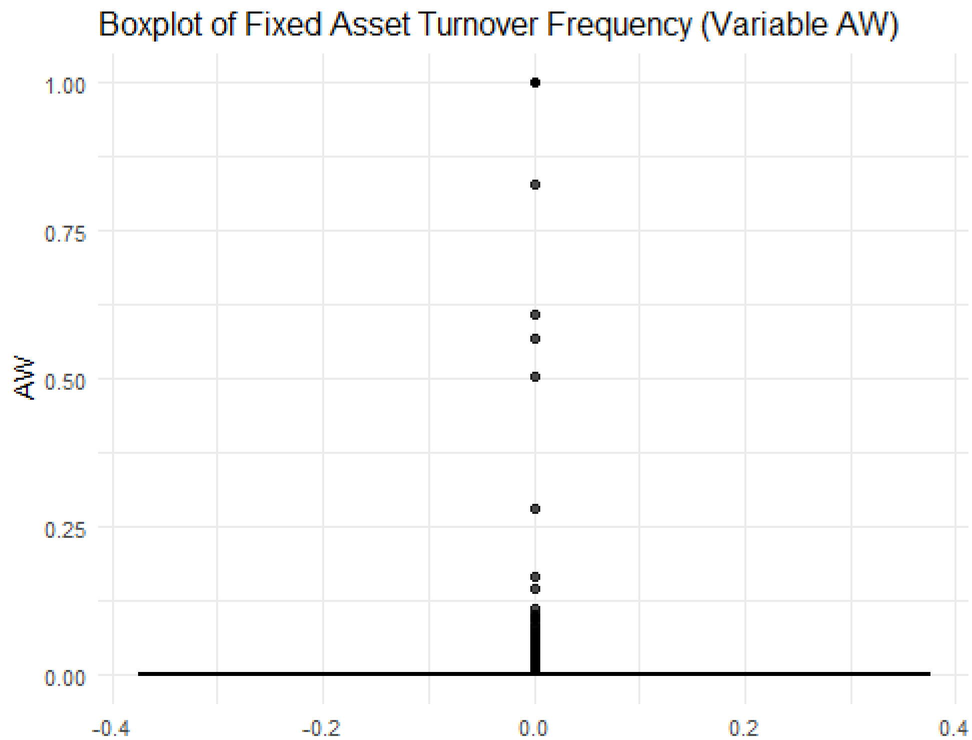
For Interest Bearing Debt Interest Rate, 6 values that be seen as distinct outliers were removed above 0.00.



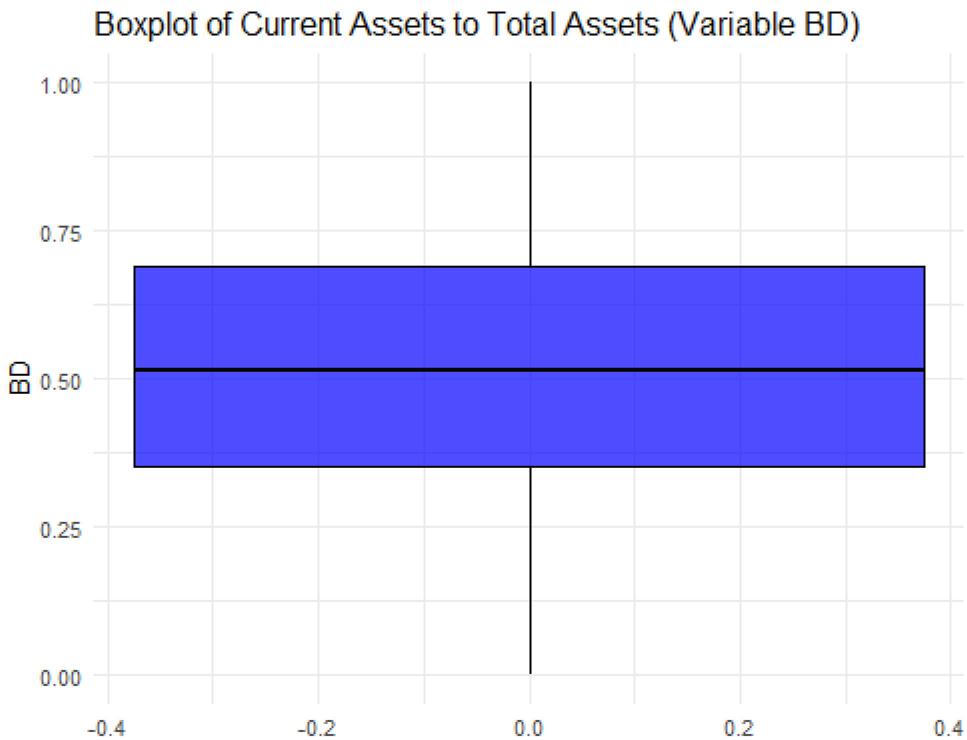
For Total Debt to Total Net Worth, distinctly visible observations above 0.2 were removed.



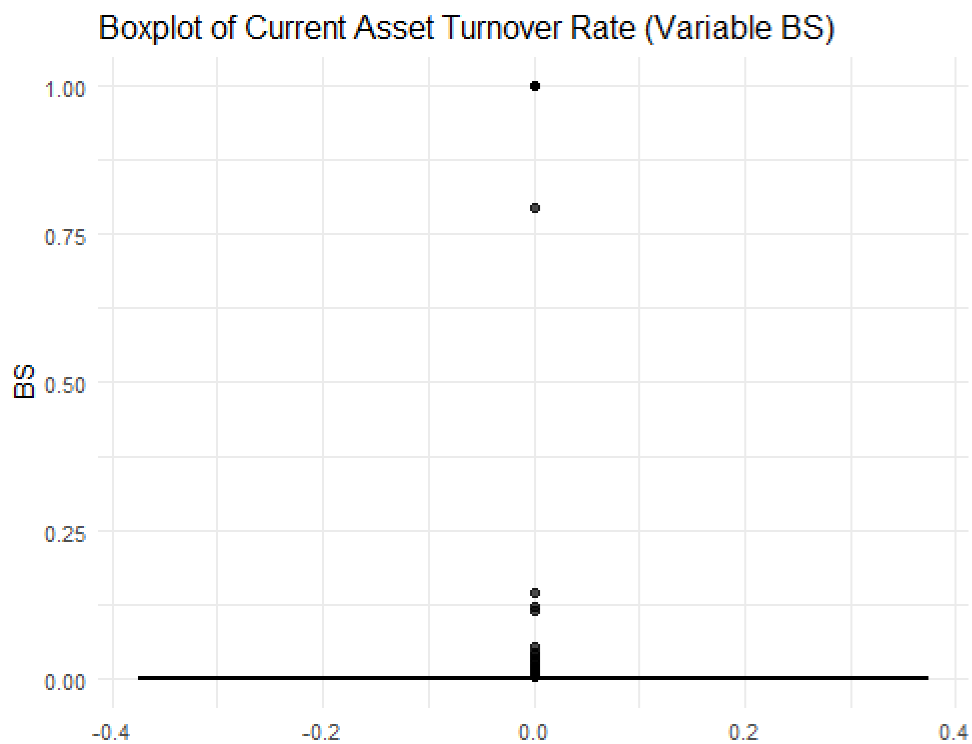
For Cash Flow to Total Assets, only distinctly visible points were removed.



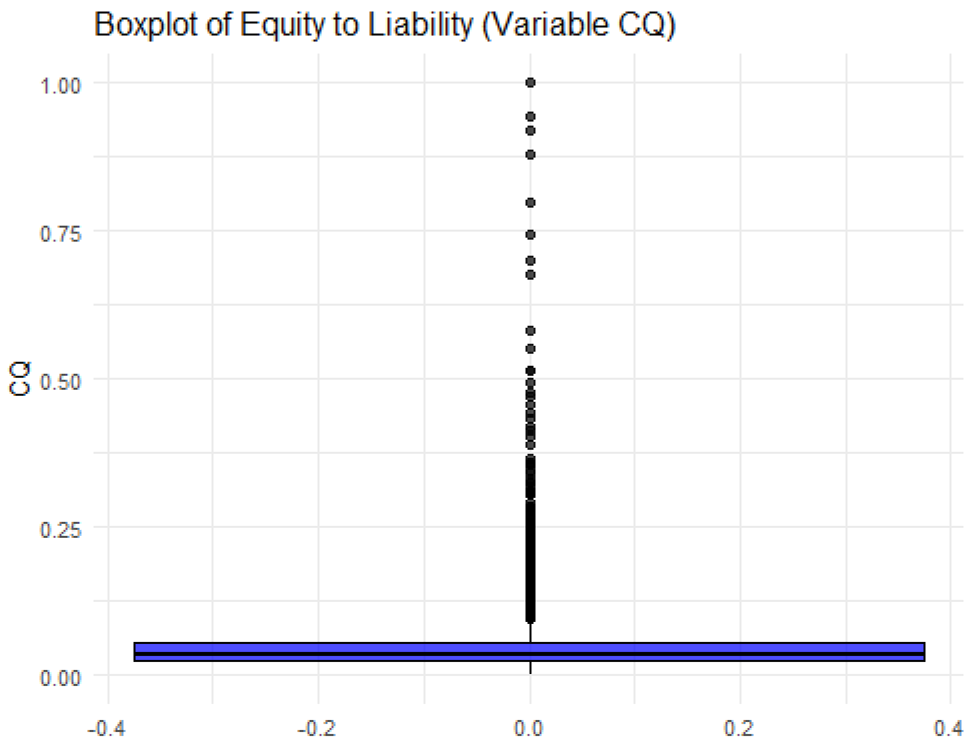
For Fixed Asset Turnover Frequency, distinctly visible points were removed.



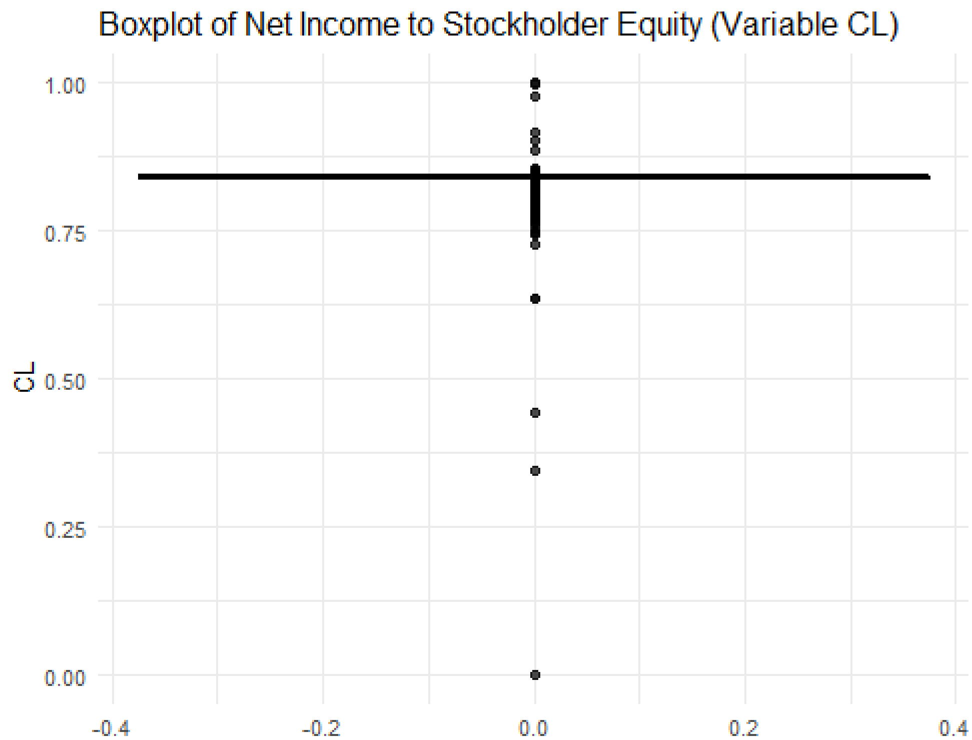
For Current Assets to Total Assets, there were no visible outliers.



For Current Asset Turnover Rate, distinctly visible points were removed.



For Equity to Liability, Distinctly visible points above 0.5 will be cluttered at 0.5.



For Net Income to Stockholder Equity, values below 0.75 will be removed.

6- Sampling Methods

We will go forward with determining what sampling method might be best for our dataset to deal with the class imbalance.

To test these sampling techniques, we need to choose an algorithm that is most robust to feature selection techniques. That way changes in accuracy will mostly be attributed to the sampling technique changing and not being dependent on which features are selected in the model. A study done by Liang et al. (2016) selected a few algorithms to determine changes in accuracy as feature selection techniques change. It was found that SVM **(Support Vector Machines)** was the algorithm that was most robust to feature selection techniques giving consistent results for every technique.

6.1- SMOTE (Synthetic Minority Oversampling Technique)

In a study done by Mahembe (2024), a few oversampling techniques were tested on highly imbalanced datasets which included SMOTE, ROWR, OBPS and WNN. Results proved that SMOTE performed the best. In another study done by Alam et al. (2021), SMOTE gave the best results when tested on financial data to predict corporate bankruptcy.

6.2- ADASYN (Adaptive Synthetic Sampling)

We used this technique because it does a good job at highlighting the boundary between the outcome classes (Barboza, Kimura and Altman, 2017). In our dataset that is particularly useful because applying PCA earlier revealed that classes are highly overlapping. Hence it might be a good idea to use ADASYN to highlight the decision boundary a bit more.

6.3- Random Over Sampling

We are more inclined towards using oversampling techniques as under sampling might cause loss of important data. In a study done by Mohammed, Rawashdeh and Abdullah (2020), it was proven that randomly oversampling minority class gives better results than under sampling it. Random Over Sampling is chosen because it is the simplest oversampling technique which could act as a benchmark.

6.4- Random Under Sampling

We did not want our study to be showing results only from the perspective of oversampling techniques even though they give better results (Mohammed, Rawashdeh and Abdullah, 2020). That is why for comparison purposes, the simplest under sampling technique was chosen.

7- Predictive Modeling

We decided on 5 models to test. Only KNN and RF are non-parametric models in our study. All models were cross validated 10 folds 3 times. All models except Logistic Regression and Multivariate Discriminant Analysis were tuned for hyperparameters because they do not specifically have hyperparameter tuning.

7.1- MDA (Multiple Discriminant Analysis)

One of the earliest works on predicting corporate bankruptcy was done by Altman (1968). He developed the Z-score model using MDA, which remains a benchmark in bankruptcy prediction studies where financial ratios are predictor variables. MDA excels in scenarios where linear relationships among predictors and normally distributed data within each group are present, allowing for clear classification boundaries. In another study done by Mvula Chijoriga (2011) where credit defaults were predicted by MDA, it was found that this model is robust enough to predict failure two years before the incident. Mahembe (2024) also proved that MDA was the least sensitive to sampling methods. Hence we chose this algorithm for its proven success throughout the years.

7.2- SVM (Support Vector Machines)

SVM is one of the most widely used prediction techniques for bankruptcy (Lin, Hu and Tsai, 2011). It is also one of the top 5 machine learning algorithms used in data mining (Wu et al., 2008). We chose SVM because by maximizing the margin between classes, SVM aims to improve the model's generalizability to unseen data, which is crucial for reliable bankruptcy prediction (Pisner and Schnyer, 2020).

For SVM we used Linear Kernel as decision boundaries created are straight lines which clearly indicate model’s performance (Wang, 2005). We controlled the regularization parameter which controls the trade-off between achieving a low training error and a low testing error, which is achieved by controlling the margin that separates the classes. We set its values to 0.1, 1 and 10.

7.3- KNN (K-Nearest Neighbor)

KNN is again one of the most widely used techniques for bankruptcy prediction (Lin, Hu and Tsai, 2011) and also one of the top 5 algorithms for data mining (Wu et al., 2008). Our primary reason for choosing KNN was the study done by Alam et al. (2021) where different countries proved to have different best performing algorithms. Our dataset revolves around Taiwanese companies and for that region KNN proved to be the best performing. If KNN performs best in our study as well, we can say that geographic differences on algorithm performance are proven further.

For KNN, the value of k represents the number of neighbors considered by the KNN algorithm when classifying a data point. In this tuning grid, four values of k are tested: 3, 5, 7, 9.

7.4- LR (Logistic Regression)

LR is present to act as a control model where linearity and normality is assumed in the dataset. Since in our dataset the distribution of variables and outcome is such that it is difficult to observe linear and normal relationships, a high performance for LR in our case would reflect relations with the target variable for predictor variables being linear and not complex (Healy, 2006).

7.5- RF (Random Forest)

We saw that our findings of PCA (highly overlapping classes) and Sampling Techniques (poor testing accuracies for NPV) were being heavily influenced by noise in the data. Hence we selected RF as it has a significant advantage of being very robust to noise in the data (Genuer, Poggi and Tuleau-Malot, 2010).

For Random Forest, mtry is the number of variables (features) randomly sampled as candidates at each split in the Random Forest model. In this tuning grid, four values of mtry are tested: 2, 4, 6, 8. We set ntree at 5 which specifies the number of trees to be grown in the forest.

8- Evaluation Metrics

Bankruptcy is our negative class represented by X1 and non-bankruptcy is our positive class represented by X0. The metrics that we will use to evaluate the performance of each type of sampling technique will be as follows.

$$TNR = \frac{True\ Negatives\ (TN)}{True\ Negatives\ (TN) + False\ Positives\ (FP)}$$

$$NPV = \frac{True\ Negatives\ (TN)}{True\ Negatives\ (TN) + False\ Negatives\ (FN)}$$

$$F2 = (1 + 2^2) * \frac{Precision * Recall}{4 * Precision + Recall}$$

$$AUC = Area\ under\ the\ curve\ for\ ROC$$

This is because Specificity will tell us the percentage of bankrupt cases correctly identified while Negative Predicted Value will tell how often the model raised a false alarm by calling a non-ban krupt company as bankrupt. The F2 score will tell us how well the model is identifying bankrupt companies with an emphasis on reducing missed cases, while AUC will tell us how effectively th e model can distinguish between bankrupt and non-bankrupt companies overall.

For our case, F2 is more important than F1 score. The F2 score is particularly useful when you w ant to prioritize minimizing false negatives over false positives, which is often the case in scenari os were missing a critical event (like bankruptcy) is more costly than raising a false alarm. F2 sc ore of 0.82 suggests that the model is performing well in terms of recall while maintaining a bala nce with precision, but with a clear emphasis on avoiding false negatives.

9- Interpretable Machine Learning

9.1- Partial Dependency Plots (PDP)

According to Friedman (2001), Partial dependency plots can show much more complex marginal dependence of each predictor on the outcome. This is done by marginalizing the predicted outcome over the features that we are not interested in, so that marginal dependence of the features of interest are visible (Son et al., 2019)

9.2- Accumulated Local Effects (ALE)

ALE works by explaining the average impact of features on machine learning model predictions (Apley and Zhu, 2020). We are accompanying PDP with ALE because ALE works even if predictor variables are highly dependent on each other while PDP assumes feature independence which might not always be the case (Danesh et al., 2022).

10- Logbook

Figure 6: Proposed Gantt Chart

Date	Entry
1-7 July	Searched literature for best outlier analysis method and applied it. Searched literature for best sampling method and applied them.
8-14 July	Searched literature for best performing algorithm for modeling and applied it. Analyzed all models and then deeply analyzed the best performing model further.
15-21 July	Searched literature for best performing IML methods and applied them. Compiled all findings to this point.
22-28 July	Completed the “Findings” heading of the main report.

29 July - 4 Aug	Completed the “Discussions” heading of the main report.
5-11 Aug	Completed the “Literature Review” heading in the main report.
12-18 Aug	Completed rest of the headings for the main report.
19-25 Aug	Completed the “Feature Selection” and “Outlier Removal” heading in the technical report.
26 Aug – 1 Sept	Completed rest of the headings for the technical report.
2-13 Sept	Final adjustments to complete both reports.

Figure 7: Entries for Each Week

11- Reflective Discussion

It goes without saying that this project has been an immense learning opportunity for me. It has made me confident about my own ability to handle data and extract insights from it. Apart from technicalities, the topic itself was something of my own interest as I aspire to work in a relevant domain in my career. Studying and going through literature about bankruptcy also helped me develop a better understanding of it.

The most difficult thing for me about the whole project was feature selection, outlier removal and sampling methods. That is because each of this things required me to go into extreme technical details to perform correctly because that was the technical foundation my research stood on. I had to perform a lot of rigorous manual work on these things too which caused them to take the most time, equal to the time taken by modeling. Some ideas given to students during our dissertation classes were also used by me. Those included using IML methods and Isolation Forest.

Ultimately this project was sufficient at building a postgraduate level understanding and expertise of business analytics. I eagerly wait to apply this newfound prowess down in my career.

References

Altman, E.I., 1968. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. The journal of finance, 23(4), pp.589-609.

John, G.H., 1995, August. Robust Decision Trees: Removing Outliers from Databases. In KDD (Vol. 95, pp. 174-179).

Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Annals of statistics, pp.1189-1232.

Wang, L. ed., 2005. Support vector machines: theory and applications (Vol. 177). Springer Science & Business Media.

Healy, L.M., 2006. Logistic regression: An overview. Eastern Michighan College of Technology.

Wang, Y.Y. and Li, J., 2008. Feature-selection ability of the decision-tree algorithm and the impact of feature-selection/extraction on decision-tree results based on hyperspectral data. International Journal of Remote Sensing, 29(10), pp.2993-3010.

Wu, X., Kumar, V., Quinlan, J.R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., and Philip, S.Y. (2008) Top 10 algorithms in data mining. Knowledge and Information Systems, vol. 14, pp. 1-37.

Gayatri, N., Nickolas, S., Reddy, A.V., Reddy, S. and Nickolas, A., 2010, October. Feature selection using decision tree induction in class level metrics dataset for software defect predictions. In Proceedings of the world congress on engineering and computer science (Vol. 1, pp. 124-129).

Genuer, R., Poggi, J.M. and Tuleau-Malot, C., 2010. Variable selection using random forests. Pattern recognition letters, 31(14), pp.2225-2236.

Liu, J., Cai, W. and Shao, X., 2011. Cancer classification based on microarray gene expression data using a principal component accumulation method. Science China Chemistry, 54, pp.802-811.

Lin, W.Y., Hu, Y.H. and Tsai, C.F., 2011. Machine learning in financial crisis prediction: a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(4), pp.421-436.

Mvula Chijoriga, M., 2011. Application of multiple discriminant analysis (MDA) as a credit scoring and risk assessment model. International Journal of Emerging Markets, 6(2), pp.132-147.

Xie, Y. (2014). Dynamic Documents with R and knitr. 2nd edn. Boca Raton, FL: Chapman & Hall/CRC.

Dwivedi, S., Kasliwal, P. and Soni, S., 2016, March. Comprehensive study of data analytics tools (RapidMiner, Weka, R tool, Knime). In 2016 Symposium on Colossal Data Analysis and Networking (CDAN) (pp. 1-8). IEEE.

Jolliffe, I.T. and Cadima, J., 2016. Principal component analysis: a review and recent developments. Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences, 374(2065), p.20150202.

Liang, D., Lu, C.C., Tsai, C.F. and Shih, G.A., 2016. Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study. European journal of operational research, 252(2), pp.561-572.

Barboza, F., Kimura, H. and Altman, E., 2017. Machine learning models and bankruptcy prediction. Expert Systems with Applications, 83, pp.405-417.

Son, H., Hyun, C., Phan, D. and Hwang, H.J., 2019. Data analytic approach for bankruptcy prediction. Expert Systems with Applications, 138, p.112816.

Mohammed, R., Rawashdeh, J. and Abdullah, M., 2020, April. Machine learning with oversampling and undersampling techniques: overview study and experimental results. In 2020 11th international conference on information and communication systems (ICICS) (pp. 243-248). IEEE.

UCI Machine Learning Repository, 2020. Taiwanese Bankruptcy Prediction. Available at: https://doi.org/10.24432/C5004D

Pisner, D.A. and Schnyer, D.M., 2020. Support vector machine. In Machine learning (pp. 101-121). Academic Press.

Apley, D.W. and Zhu, J., 2020. Visualizing the effects of predictor variables in black box supervised learning models. Journal of the Royal Statistical Society Series B: Statistical Methodology, 82(4), pp.1059-1086.

Al Farizi, W.S., Hidayah, I. and Rizal, M.N., 2021, September. Isolation forest based anomaly detection: A systematic literature review. In 2021 8th International Conference on Information Technology, Computer and Electrical Engineering (ICITACEE) (pp. 118-122). IEEE.

Alam, T.M., Shaukat, K., Mushtaq, M., Ali, Y., Khushi, M., Luo, S. and Wahab, A., 2021. Corporate bankruptcy prediction: An approach towards better corporate world. The Computer Journal, 64(11), pp.1731-1746.

Danesh, T., Ouaret, R., Floquet, P. and Negny, S., 2022. Interpretability of neural networks predictions using Accumulated Local Effects as a model-agnostic method. In Computer aided chemical engineering (Vol. 51, pp. 1501-1506). Elsevier.

Mahembe, W., 2024. Use of machine learning in bankruptcy prediction with highly imbalanced datasets: The impact of sampling methods.

Appendix

Variables renamed to letters

A = "ROA.C..before.interest.and.depreciation.before.interest",

B = "ROA.A..before.interest.and...after.tax",

C = "ROA.B..before.interest.and.depreciation.after.tax",

D = "Operating.Gross.Margin",

E = "Realized.Sales.Gross.Margin",

F = "Operating.Profit.Rate",

G = "Pre.tax.net.Interest.Rate",

H = "After.tax.net.Interest.Rate",

I = "Non.industry.income.and.expenditure.revenue",

J = "Continuous.interest.rate..after.tax.",

K = "Operating.Expense.Rate",

L = "Research.and.development.expense.rate",

M = "Cash.flow.rate",

N = "Interest.bearing.debt.interest.rate",

O = "Tax.rate..A.",

P = "Net.Value.Per.Share..B.",

Q = "Net.Value.Per.Share..A.",

R = "Net.Value.Per.Share..C.",

S = "Persistent.EPS.in.the.Last.Four.Seasons",

T = "Cash.Flow.Per.Share",

U = "Revenue.Per.Share..Yuan...",

V = "Operating.Profit.Per.Share..Yuan...",

W = "Per.Share.Net.profit.before.tax..Yuan...",

X = "Realized.Sales.Gross.Profit.Growth.Rate",

Y = "Operating.Profit.Growth.Rate",

Z = "After.tax.Net.Profit.Growth.Rate",

AA = "Regular.Net.Profit.Growth.Rate",

AB = "Continuous.Net.Profit.Growth.Rate",

AC = "Total.Asset.Growth.Rate",

AD = "Net.Value.Growth.Rate",

AE = "Total.Asset.Return.Growth.Rate.Ratio",

AF = "Cash.Reinvestment..",

AG = "Current.Ratio",

AH = "Quick.Ratio",

AI = "Interest.Expense.Ratio",

AJ = "Total.debt.Total.net.worth",

AK = "Debt.ratio..",

AL = "Net.worth.Assets",

AM = "Long.term.fund.suitability.ratio..A.",

AN = "Borrowing.dependency",

AO = "Contingent.liabilities.Net.worth",

AP = "Operating.profit.Paid.in.capital",

AQ = "Net.profit.before.tax.Paid.in.capital",

AR = "Inventory.and.accounts.receivable.Net.value",

AS = "Total.Asset.Turnover",

AT = "Accounts.Receivable.Turnover",

AU = "Average.Collection.Days",

AV = "Inventory.Turnover.Rate..times.",

AW = "Fixed.Assets.Turnover.Frequency",

AX = "Net.Worth.Turnover.Rate..times.",

AY = "Revenue.per.person",

AZ = "Operating.profit.per.person",

BA = "Allocation.rate.per.person",

BB = "Working.Capital.to.Total.Assets",

BC = "Quick.Assets.Total.Assets",

BD = "Current.Assets.Total.Assets",

BE = "Cash.Total.Assets",

BF = "Quick.Assets.Current.Liability",

BG = "Cash.Current.Liability",

BH = "Current.Liability.to.Assets",

BI = "Operating.Funds.to.Liability",

BJ = "Inventory.Working.Capital",

BK = "Inventory.Current.Liability",

BL = "Current.Liabilities.Liability",

BM = "Working.Capital.Equity",

BN = "Current.Liabilities.Equity",

BO = "Long.term.Liability.to.Current.Assets",

BP = "Retained.Earnings.to.Total.Assets",

BQ = "Total.income.Total.expense",

BR = "Total.expense.Assets",

BS = "Current.Asset.Turnover.Rate",

BT = "Quick.Asset.Turnover.Rate",

BU = "Working.capitcal.Turnover.Rate",

BV = "Cash.Turnover.Rate",

BW = "Cash.Flow.to.Sales",

BX = "Fixed.Assets.to.Assets",

BY = "Current.Liability.to.Liability",

BZ = "Current.Liability.to.Equity",

CA = "Equity.to.Long.term.Liability",

CB = "Cash.Flow.to.Total.Assets",

CC = "Cash.Flow.to.Liability",

CD = "CFO.to.Assets",

CE = "Cash.Flow.to.Equity",

CF = "Current.Liability.to.Current.Assets",

CG = "Liability.Assets.Flag",

CH = "Net.Income.to.Total.Assets",

CI = "Total.assets.to.GNP.price",

CJ = "No.credit.Interval",

CK = "Gross.Profit.to.Sales",

CL = "Net.Income.to.Stockholder.s.Equity",

CM = "Liability.to.Equity",

CN = "Degree.of.Financial.Leverage..DFL.",

CO = "Interest.Coverage.Ratio..Interest.expense.to.EBIT.",

CP = "Net.Income.Flag",

CQ = "Equity.to.Liability",

bankrupt = "Bankrupt."

R Code

```
# Load the necessary libraries
library(dplyr)        # Data manipulation
library(ggplot2)      # Data visualization
library(tidyr)        # Data manipulation
library(writexl)      # Save cleaned dataset
library(readxl)       # Read cleaned dataset
library(caret)        # Create predictive models
library(ROSE)         # Sampling methods (Random over and under sampling)
library(pROC)         # Creating ROC curves
library(themis)       # Sampling methods (SMOTE and ADASYN)
library(recipes)      # Cross validation for sampling methods
library(solitude)     # Anomaly detection using Isolation Forest
library(mda)          # MDA (Multivariate Discriminant Analysis) Model
library(e1071)        # SVM (Support Vector Machines) Model
library(iml)          # PDPs (Partial Dependency Plots) and ALE (Accumulated Local Effects)


#Loading Data and Summary Statistics
df_original <- read.csv(file.choose())


#Renaming Variables
df_original <- df_original %>%
  rename(
    A = "ROA.C..before.interest.and.depreciation.before.interest",
    B = "ROA.A..before.interest.and...after.tax",
    C = "ROA.B..before.interest.and.depreciation.after.tax",
    D = "Operating.Gross.Margin",
    E = "Realized.Sales.Gross.Margin",
    F = "Operating.Profit.Rate",
    G = "Pre.tax.net.Interest.Rate",
    H = "After.tax.net.Interest.Rate",
    I = "Non.industry.income.and.expenditure.revenue",
    J = "Continuous.interest.rate..after.tax.",
    K = "Operating.Expense.Rate",
    L = "Research.and.development.expense.rate",
    M = "Cash.flow.rate",
    N = "Interest.bearing.debt.interest.rate",
    O = "Tax.rate..A.",
    P = "Net.Value.Per.Share..B.",
    Q = "Net.Value.Per.Share..A.",
    R = "Net.Value.Per.Share..C.",
    S = "Persistent.EPS.in.the.Last.Four.Seasons",
    T = "Cash.Flow.Per.Share",
    U = "Revenue.Per.Share..Yuan...",
    V = "Operating.Profit.Per.Share..Yuan...",
    W = "Per.Share.Net.profit.before.tax..Yuan...",
    X = "Realized.Sales.Gross.Profit.Growth.Rate",
    Y = "Operating.Profit.Growth.Rate",
    Z = "After.tax.Net.Profit.Growth.Rate",
    AA = "Regular.Net.Profit.Growth.Rate",
    AB = "Continuous.Net.Profit.Growth.Rate",
    AC = "Total.Asset.Growth.Rate",
    AD = "Net.Value.Growth.Rate",
    AE = "Total.Asset.Return.Growth.Rate.Ratio",
```

```
    AF = "Cash.Reinvestment..",
    AG = "Current.Ratio",
    AH = "Quick.Ratio",
    AI = "Interest.Expense.Ratio",
    AJ = "Total.debt.Total.net.worth",
    AK = "Debt.ratio..",
    AL = "Net.worth.Assets",
    AM = "Long.term.fund.suitability.ratio..A.",
    AN = "Borrowing.dependency",
    AO = "Contingent.liabilities.Net.worth",
    AP = "Operating.profit.Paid.in.capital",
    AQ = "Net.profit.before.tax.Paid.in.capital",
    AR = "Inventory.and.accounts.receivable.Net.value",
    AS = "Total.Asset.Turnover",
    AT = "Accounts.Receivable.Turnover",
    AU = "Average.Collection.Days",
    AV = "Inventory.Turnover.Rate..times.",
    AW = "Fixed.Assets.Turnover.Frequency",
    AX = "Net.Worth.Turnover.Rate..times.",
    AY = "Revenue.per.person",
    AZ = "Operating.profit.per.person",
    BA = "Allocation.rate.per.person",
    BB = "Working.Capital.to.Total.Assets",
    BC = "Quick.Assets.Total.Assets",
    BD = "Current.Assets.Total.Assets",
    BE = "Cash.Total.Assets",
    BF = "Quick.Assets.Current.Liability",
    BG = "Cash.Current.Liability",
    BH = "Current.Liability.to.Assets",
    BI = "Operating.Funds.to.Liability",
    BJ = "Inventory.Working.Capital",
    BK = "Inventory.Current.Liability",
    BL = "Current.Liabilities.Liability",
    BM = "Working.Capital.Equity",
    BN = "Current.Liabilities.Equity",
    BO = "Long.term.Liability.to.Current.Assets",
    BP = "Retained.Earnings.to.Total.Assets",
    BQ = "Total.income.Total.expense",
    BR = "Total.expense.Assets",
    BS = "Current.Asset.Turnover.Rate",
    BT = "Quick.Asset.Turnover.Rate",
    BU = "Working.capitcal.Turnover.Rate",
    BV = "Cash.Turnover.Rate",
    BW = "Cash.Flow.to.Sales",
    BX = "Fixed.Assets.to.Assets",
    BY = "Current.Liability.to.Liability",
    BZ = "Current.Liability.to.Equity",
    CA = "Equity.to.Long.term.Liability",
    CB = "Cash.Flow.to.Total.Assets",
    CC = "Cash.Flow.to.Liability",
    CD = "CFO.to.Assets",
    CE = "Cash.Flow.to.Equity",
    CF = "Current.Liability.to.Current.Assets",
    CG = "Liability.Assets.Flag",
    CH = "Net.Income.to.Total.Assets",
    CI = "Total.assets.to.GNP.price",
    CJ = "No.credit.Interval",
    CK = "Gross.Profit.to.Sales",
    CL = "Net.Income.to.Stockholder.s.Equity",
    CM = "Liability.to.Equity",
    CN = "Degree.of.Financial.Leverage..DFL.",
```

```
    CO = "Interest.Coverage.Ratio..Interest.expense.to.EBIT.",
    CP = "Net.Income.Flag",
    CQ = "Equity.to.Liability",
    bankrupt = "Bankrupt."
  )

#Remove variables with 0 Standard Deviation
non_zero_sd_columns <- sapply(df_original, function(x) sd(x, na.rm = TRUE) != 0)
df_original <- df_original[, non_zero_sd_columns]


#Converting outcome to a factor
df_original$bankrupt <- factor(make.names(df_original$bankrupt))




# Calculating Correlations within Predictor Variables

df <- df_original
df_correlation <- df[, -which(names(df) == "bankrupt")]


correlation_matrix <- cor(df_correlation, use = "complete.obs")
correlation_pairs <- as.data.frame(as.table(correlation_matrix))
correlation_pairs <- correlation_pairs %>%
  filter(Var1 != Var2)
correlation_pairs <- correlation_pairs %>%
  arrange(desc(abs(Freq)))
print(correlation_pairs)

summary(correlation_pairs$Freq)

boxplot(correlation_pairs$Freq)

ggplot(correlation_pairs, aes(y = Freq)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(title = "Boxplot of Correlation Coefficients",
       y = "Correlation Coefficient (Freq)",
       x = "") +
  theme_minimal()

# Calculating Correlations with Outcome Variable

df_correlation <- df

df_correlation <- df_correlation %>%
  mutate(across(everything(), as.numeric))

correlations2 <- sapply(df_correlation, function(x) cor(x, df_correlation$bankrupt, use = "com
plete.obs"))
correlation_df2 <- data.frame(Variable = names(correlations2), Correlation = correlations2)
correlation_df2 <- correlation_df2[correlation_df2$Variable != "bankrupt", ]
bankruptcy_correlations <- correlation_df2 %>%
  arrange(desc(abs(Correlation)))
print(bankruptcy_correlations)

summary(bankruptcy_correlations$Correlation)
```

```r
ggplot(bankruptcy_correlations, aes(y = Correlation)) +
  geom_boxplot(fill = "lightgreen", color = "black") +
  labs(title = "Boxplot of Correlations with Bankruptcy",
       y = "Correlation Coefficient",
       x = "") +
  theme_minimal()

#Load data as df
df <- df_original


#### Outlier Analysis using Isolation Forest

# Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(df$bankrupt, p = 0.8, list = FALSE)
trainData <- df[trainIndex, ]
testData <- df[-trainIndex, ]

# Fit Isolation Forest model using solitude on the training dataset
iso_model <- isolationForest$new()
iso_model$fit(trainData %>% select(-bankrupt))

# Predict anomaly scores for the training and testing datasets
trainData$anomaly_score <- iso_model$predict(trainData %>% select(-bankrupt))$anomaly_score
testData$anomaly_score <- iso_model$predict(testData %>% select(-bankrupt))$anomaly_score

# Count the number of outliers
num_outliers <- sum(testData$anomaly_score > 0.61)
print(paste("Number of outliers: ", num_outliers))

print(paste("Portion of the total dataset: ", num_outliers/1363))

# Classify as outlier based on the threshold
trainData$outlier <- ifelse(trainData$anomaly_score > 0.61, 1, 0)
testData$outlier <- ifelse(testData$anomaly_score > 0.61, 1, 0)

# Compute ROC Curve and AUC-ROC for test data
roc_curve <- roc(testData$bankrupt, testData$anomaly_score)
auc_roc <- auc(roc_curve)

# Plot ROC Curve
plot(roc_curve, col = "blue", main = paste("ROC Curve (AUC = ", round(auc_roc, 2), ")", sep =
""))
abline(a = 0, b = 1, col = "red", lty = 2)

# Print AUC-ROC value
print(paste("AUC-ROC:", round(auc_roc, 2)))

# Visualize the anomaly scores for training data
ggplot(trainData, aes(x = anomaly_score)) +
  geom_histogram(bins = 50, alpha = 0.7, fill = "blue", color = "black") +
  labs(title = "Anomaly Scores from Isolation Forest (Training Data)",
       x = "Anomaly Score",
       y = "Frequency") +
  theme_minimal()

# Visualize the anomaly scores for test data
ggplot(testData, aes(x = anomaly_score)) +
  geom_histogram(bins = 50, alpha = 0.7, fill = "blue", color = "black") +
  labs(title = "Anomaly Scores from Isolation Forest (Test Data)",
```

```
       x = "Anomaly Score",
       y = "Frequency") +
  theme_minimal()

#Filtering Variables after KNIME Decision Tree

df_knime <- df %>% select(bankrupt, N, AJ, BS, AW,
                          AU, BA,
                          AN, BP, CH, BM, BD, CB, H, CQ, CN, AS, CO, CL)
df_knime$bankrupt <- factor(make.names(df_knime$bankrupt))

#Focusing on removing further variables

#### N

summary(df$N)

#Table of Values to identify extreme outliers
value_counts_N <- table(df$N)
df_value_counts_N <- as.data.frame(value_counts_N)
colnames(df_value_counts_N) <- c("Value", "Count")
df_value_counts_N$Value <- as.numeric(as.character(df_value_counts_N$Value))
df_value_counts_N <- df_value_counts_N[order(-df_value_counts_N$Value), ]
print(df_value_counts_N)

print(sum(df$N > 1))

print(sum(df$N > 0.00)) #effecting 1 of bankruptcy

print(sum(df$N <= 0.00))

# Modify N such that values greater than 1 are replaced by 1
df <- df %>%
  mutate(N = ifelse(N > 1, 1, N))



##### AJ

summary(df$AJ)

#Table of Values to identify extreme outliers
value_counts_AJ <- table(df$AJ)
df_value_counts_AJ <- as.data.frame(value_counts_AJ)
colnames(df_value_counts_AJ) <- c("Value", "Count")
df_value_counts_AJ$Value <- as.numeric(as.character(df_value_counts_AJ$Value))
df_value_counts_AJ <- df_value_counts_AJ[order(-df_value_counts_AJ$Value), ]
print(df_value_counts_AJ)

print(sum(df$AJ > 1))

print(sum(df$AJ > 0.00)) #effecting 1 of bankruptcy

print(sum(df$AJ <= 0.00))

# Modify AJ such that values greater than 1 are replaced by 1
df <- df %>%
  mutate(AJ = ifelse(AJ > 1, 1, AJ))



#### BS
```

```r
summary(df$BS)

#Table of Values to identify extreme outliers
value_counts_BS <- table(df$BS)
df_value_counts_BS <- as.data.frame(value_counts_BS)
colnames(df_value_counts_BS) <- c("Value", "Count")
df_value_counts_BS$Value <- as.numeric(as.character(df_value_counts_BS$Value))
df_value_counts_BS <- df_value_counts_BS[order(-df_value_counts_BS$Value), ]
print(df_value_counts_BS)

print(sum(df$BS > 1))

print(sum(df$BS > 0.00)) #effecting 1 of bankruptcy

# Modify BS such that values greater than 1 are replaced by 1
df <- df %>%
  mutate(BS = ifelse(BS > 1, 1, BS))

print(sum(df$BS > 0.00))

#### AW

summary(df$AW)

#Table of Values to identify extreme outliers
value_counts_AW <- table(df$AW)
df_value_counts_AW <- as.data.frame(value_counts_AW)
colnames(df_value_counts_AW) <- c("Value", "Count")
df_value_counts_AW$Value <- as.numeric(as.character(df_value_counts_AW$Value))
df_value_counts_AW <- df_value_counts_AW[order(-df_value_counts_AW$Value), ]
print(df_value_counts_AW)

print(sum(df$AW > 1)) #very strong outliers

print(sum(df$AW > 0.00))

print(sum(df$AW <= 3255000000)) #effecting 1 of bankruptcy (3.255e9)

print(sum(df$AW > 3255000000))

print(sum(df$AW <= 3225000000))

df <- df %>%
  mutate(AW = ifelse(AW > 1, 1, AW))



#### AU

print(sum(df$AU <= 0.02)) #affecting 1 of bankruptcy

#Table to Identify outliers
value_counts_AU <- table(df$AU)
df_value_counts_AU <- as.data.frame(value_counts_AU)
colnames(df_value_counts_AU) <- c("Value", "Count")
df_value_counts_AU$Value <- as.numeric(as.character(df_value_counts_AU$Value))
df_value_counts_AU <- df_value_counts_AU[order(-df_value_counts_AU$Value), ]
print(df_value_counts_AU)
```

```r
df_knime <- df_knime %>%
  mutate(AU = ifelse(AU > 1, 1, AU))
boxplot(df_knime$AU)

#### BA

print(sum(df$BA <= 0.02))

print(sum(df$BA <= 0.05))

#Table to identify outliers
value_counts_BA <- table(df$BA)
df_value_counts_BA <- as.data.frame(value_counts_BA)
colnames(df_value_counts_BA) <- c("Value", "Count")
df_value_counts_BA$Value <- as.numeric(as.character(df_value_counts_BA$Value))
df_value_counts_BA <- df_value_counts_BA[order(-df_value_counts_BA$Value), ]
print(df_value_counts_BA)

df_knime <- df_knime %>%
  mutate(BA = ifelse(BA > 1, 1, BA))
boxplot(df_knime$BA)

#### AN

print(sum(df$AN > 0.38)) #values effecting bankruptcy

summary(df$AN)

#### BP

print(sum(df$BP <= 0.91)) #values predicting bankruptcy

summary(df$BP)

#### CH

print(sum(df$CH <= 0.8)) #Values predicting bankruptcy

summary(df$CH)

##### BM


print(sum(df$BM <= 0.73)) #Values predicting bankruptcy

#### BD

print(sum(df$BD > 0.58)) #Values predicting bankruptcy

####CB

print(sum(df$CB <= 0.65)) #Values predicting bankruptcy

##### H

print(sum(df$H > 0.81)) #Values predicting bankruptcy

#### CQ

print(sum(df$CQ > 0.03)) #Values predicting bankruptcy
```

```r
####CN

print(sum(df$CN > 0.03)) #Values predicting bankruptcy

#### AS

print(sum(df$AS <= 0.03)) #Values predicting bankruptcy

#### CO

print(sum(df$CO <= 0.56)) #Values predicting bankruptcy

#### CL

print(sum(df$CL > 0.84)) #Values predicting bankruptcy

#Variables selected: N, AJ, BS, AW, AN, CH, BD, CB, CQ, CL

#Box plot for observing outliers

### N
boxplot(df$N)

ggplot(df, aes(y = N)) +
  geom_boxplot(fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Boxplot of Interest Bearing Debt Interest Rate (Variable N)",
       y = "N",
       x = "") +
  theme_minimal()

### AJ
boxplot(df$AJ)

ggplot(df, aes(y = AJ)) +
  geom_boxplot(fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Boxplot of Total Debt to Total Net Worth (Variable AJ)",
       y = "AJ",
       x = "") +
  theme_minimal()

### BS
boxplot(df$BS)

ggplot(df, aes(y = BS)) +
  geom_boxplot(fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Boxplot of Current Asset Turnover Rate (Variable BS)",
       y = "BS",
       x = "") +
  theme_minimal()

### AW
boxplot(df$AW)

summary(df$AW)

print(sum(df$AW <= 1)) #count of observations predicting bankruptcy

ggplot(df, aes(y = AW)) +
  geom_boxplot(fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Boxplot of Fixed Asset Turnover Frequency (Variable AW)",
```

```r
       y = "AW",
       x = "") +
  theme_minimal()

### AN
boxplot(df$AN)

ggplot(df, aes(y = AN)) +
  geom_boxplot(fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Boxplot of Borrowing Dependency (Variable AN)",
       y = "AN",
       x = "") +
  theme_minimal()

### CH
boxplot(df$CH)

ggplot(df, aes(y = CH)) +
  geom_boxplot(fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Boxplot of Net Income to Total Assets (Variable CH)",
       y = "CH",
       x = "") +
  theme_minimal()

### BD
boxplot(df$BD) #No outliers

ggplot(df, aes(y = BD)) +
  geom_boxplot(fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Boxplot of Current Assets to Total Assets (Variable BD)",
       y = "BD",
       x = "") +
  theme_minimal()

### CB
boxplot(df$CB)

ggplot(df, aes(y = CB)) +
  geom_boxplot(fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Boxplot of Cash Flow to Total Assets (Variable CB)",
       y = "CB",
       x = "") +
  theme_minimal()

### CQ
boxplot(df$CQ)

ggplot(df, aes(y = CQ)) +
  geom_boxplot(fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Boxplot of Equity to Liability (Variable CQ)",
       y = "CQ",
       x = "") +
  theme_minimal()

### CL
boxplot(df$CL)

ggplot(df, aes(y = CL)) +
  geom_boxplot(fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Boxplot of Net Income to Stockholder Equity (Variable CL)",
       y = "CL",
```

```
      x = "") +
  theme_minimal()

# Removing observed outliers

df_knime <- df_knime %>%
  mutate(N = ifelse(N >= 0.2, 0.2 , N))

df_knime <- df_knime %>%
  mutate(AJ = ifelse(AJ >= 0.2, 0.2 , AJ))

df_knime <- df_knime %>%
  mutate(BS = ifelse(BS >= 0.06, 0.06 , BS))


df_knime <- df_knime %>%
  mutate(AW = ifelse(AW > 1, 1, AW))
df_knime <- df_knime %>%
  mutate(AW = ifelse(AW > 0.125, 0.125, AW))

df_knime <- df_knime %>%
  mutate(AN = ifelse(AN <= 0.38, 0.38 , AN))

df_knime <- df_knime %>%
  mutate(CH = ifelse(CH <= 0.5, 0.5 , CH))

df_knime <- df_knime %>%
  mutate(CB = ifelse(CB <= 0.375, 0.375 , CB))

df_knime <- df_knime %>%
  mutate(CQ = ifelse(CQ >= 0.5, 0.5 , CQ))

df_knime <- df_knime %>%
  mutate(CL = ifelse(CL <= 0.75, 0.75 , CL))


#Creating a clean new dataset
df_clean <- df_knime %>% select(bankrupt, AN, CH, N, AJ, CB, AW, BD, BS, CQ, CL)
summary(df_knime)

#Save cleaned dataset
library(writexl)
write_xlsx(df_clean, path = "Cleaned Data.xlsx")


#New dataset for visualizations
df_viz <- df_clean

# Function to remove outliers based on 1.5*IQR criterion from the visualization dataset
remove_outliers <- function(df, variable) {
  Q1 <- quantile(df[[variable]], 0.25, na.rm = TRUE)
  Q3 <- quantile(df[[variable]], 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR
  df <- df %>% filter(df[[variable]] >= lower_bound & df[[variable]] <= upper_bound)
  return(df)
}
```

```r
# List of variables to remove outliers from
variables <- c("AN", "CH", "N", "AJ", "CB", "AW", "BD", "BS", "CQ", "CL")

# Apply the outlier removal function to each variable
df_viz <- df_viz
for (variable in variables) {
  df_viz <- remove_outliers(df_viz, variable)
}

#Visualizations

# Calculate the mean of the AN and CH variables
mean_AN <- mean(df_viz$AN, na.rm = TRUE)
mean_CH <- mean(df_viz$CH, na.rm = TRUE)

# Create the scatter plot for variables AN and CH with vertical and horizontal lines at the me
ans
ggplot(df_viz, aes(x = AN, y = CH)) +
  geom_point(color = "blue", alpha = 0.6) +
  geom_vline(xintercept = mean_AN, color = "red", linetype = "solid") +
  geom_hline(yintercept = mean_CH, color = "red", linetype = "solid") +
  labs(title = "Borrowing Dependency vs. Net Income / Total Assets",
       x = "Borrowing Dependency",
       y = "Net Income / Total Assets") +
  theme_minimal()

ggplot(df_viz, aes(x = AN, y = AW)) +
  geom_point(color = "blue", alpha = 0.6) +
  labs(title = "Borrowing Dependency vs. Fixed Asset Turnover Frequency",
       x = "Borrowing Dependency",
       y = "Fixed Asset Turnover Frequency") +
  theme_minimal()

ggplot(df_viz, aes(x = CQ, y = N)) +
  geom_point(color = "blue", alpha = 0.6) +
  labs(title = "Equity / Liability vs. Interest Bearing Debt Interest Rate",
       x = "Equity / Liability",
       y = "Interest Bearing Debt Interest Rate") +
  theme_minimal()

ggplot(df_viz, aes(x = AN, y = CB)) +
  geom_point(color = "blue", alpha = 0.6) +
  labs(title = "Borrowing Dependency vs. Cash Flow",
       x = "Borrowing Dependency",
       y = "Cash Flow") +
  theme_minimal()

df_avg <- df_viz %>%
  group_by(bankrupt) %>%
  summarise(Current.Asset.Turnover = mean(BS, na.rm = TRUE), Fixed.Asset.Turnover = mean(AW, n
a.rm = TRUE)) %>%
  pivot_longer(cols = c(Current.Asset.Turnover, Fixed.Asset.Turnover), names_to = "Turnovers",
values_to = "Average")
print(df_avg)

combined_plot <- ggplot(df_avg, aes(x = Turnovers, y = Average, fill = factor(bankrupt))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Average Turnovers by Bankrupt Status",
       x = "Turnover",
```

```r
      y = "Average",
      fill = "Bankrupt Status") +
  theme_minimal()
print(combined_plot)

### Testing Sampling Methods

# Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(df$bankrupt, p = 0.8, list = FALSE)
trainData <- df[trainIndex, ]
testData <- df[-trainIndex, ]


# Function to calculate NPV
calculate_npv <- function(cm) {
  tn <- cm$table[2,2]
  fn <- cm$table[2,1]
  npv <- tn / (tn + fn)
  return(npv)
}

# Function to train and evaluate model
train_and_evaluate <- function(sampling_method, trainData, testData) {
  # Apply sampling within the train function using a recipe
  if (sampling_method == "smote") {
    recipe <- recipe(bankrupt ~ ., data = trainData) %>%
      step_smote(bankrupt, over_ratio = 1)
  } else if (sampling_method == "adasyn") {
    recipe <- recipe(bankrupt ~ ., data = trainData) %>%
      step_adasyn(bankrupt, over_ratio = 1)
  } else if (sampling_method == "under") {
    recipe <- recipe(bankrupt ~ ., data = trainData) %>%
      step_downsample(bankrupt)
  } else if (sampling_method == "over") {
    recipe <- recipe(bankrupt ~ ., data = trainData) %>%
      step_upsample(bankrupt)
  }

  # Prepare the recipe and apply it to the training data
  prep_recipe <- prep(recipe)
  trainData_sampled <- bake(prep_recipe, new_data = NULL)

  # Define cross-validation with resampling
  cvControl <- trainControl(
    method = "repeatedcv",
    number = 10,
    repeats = 3,
    summaryFunction = twoClassSummary,
    classProbs = TRUE,
    savePredictions = "all"
  )

  # Train the model using cross-validation
  set.seed(123)
  model <- train(
    bankrupt ~ .,
    data = trainData_sampled,
    method = "svmLinear",  # Random Forest, you can choose any other method
```

```
    trControl = cvControl,
    metric = "ROC",
    preProcess = NULL,
    tuneLength = 5
  )

  # Predict on test data
  predictions <- predict(model, newdata = testData)
  test_cm <- confusionMatrix(predictions, testData$bankrupt)
  test_specificity <- test_cm$byClass["Specificity"]
  test_npv <- calculate_npv(test_cm)

  # Extract resampling results
  resampling_results <- model$resample
  resampling_results$TestSpecificity <- test_specificity
  resampling_results$TestNPV <- test_npv
  resampling_results$SamplingMethod <- sampling_method

  # Calculate training NPV for each resample
  npv_values <- numeric()
  for (resample in unique(model$pred$Resample)) {
    pred_subset <- model$pred[model$pred$Resample == resample, ]
    cm <- confusionMatrix(pred_subset$pred, pred_subset$obs)
    npv <- calculate_npv(cm)
    npv_values <- c(npv_values, npv)
  }

  resampling_results$TrainNPV <- npv_values

  return(resampling_results)
}

# Apply the function to SMOTE, ADASYN, under-sampling, and over-sampling
sampling_methods <- c("smote", "adasyn", "under", "over")
results_list <- lapply(sampling_methods, function(method) train_and_evaluate(method, trainData
, testData))

# Combine and compare results
combined_results <- bind_rows(results_list)

# Plot the distribution of Test Specificity
ggplot(combined_results, aes(x = TestSpecificity, fill = SamplingMethod)) +
  geom_histogram(binwidth = 0.01, color = "black", alpha = 0.7, position = "dodge") +
  labs(title = "Comparison of Test Specificity for Different Sampling Methods",
       x = "Test Specificity",
       y = "Frequency") +
  theme_minimal()

# Plot the distribution of Test NPV
ggplot(combined_results, aes(x = TestNPV, fill = SamplingMethod)) +
  geom_histogram(binwidth = 0.01, color = "black", alpha = 0.7, position = "dodge") +
  labs(title = "Comparison of Test NPV for Different Sampling Methods",
       x = "Test NPV",
       y = "Frequency") +
  theme_minimal()

# Plot the distribution of Training Specificity
ggplot(combined_results, aes(x = Spec, fill = SamplingMethod)) +
  geom_histogram(binwidth = 0.01, color = "black", alpha = 0.7, position = "dodge") +
  labs(title = "Comparison of Training Specificity for Different Sampling Methods",
```

```r
       x = "Training Specificity",
       y = "Frequency") +
  theme_minimal()

# Plot the distribution of Training NPV
ggplot(combined_results, aes(x = TrainNPV, fill = SamplingMethod)) +
  geom_histogram(binwidth = 0.01, color = "black", alpha = 0.7, position = "dodge") +
  labs(title = "Comparison of Training NPV for Different Sampling Methods",
       x = "Training NPV",
       y = "Frequency") +
  theme_minimal()

print(combined_results)

# Plot the distribution of Training NPV for "over" and "under" sampling methods
ggplot(combined_results, aes(x = TrainNPV, fill = SamplingMethod)) +
  geom_histogram(data = subset(combined_results, SamplingMethod %in% c("over", "under")),
                 binwidth = 0.01, color = "black", alpha = 0.7, position = "dodge") +
  labs(title = "Comparison of Training NPV for Over and Under Sampling Methods",
       x = "Training NPV",
       y = "Frequency") +
  theme_minimal()

# Plot the distribution of Training NPV for "smote" and "adasyn" sampling methods
ggplot(combined_results, aes(x = TrainNPV, fill = SamplingMethod)) +
  geom_histogram(data = subset(combined_results, SamplingMethod %in% c("smote", "adasyn")),
                 binwidth = 0.01, color = "black", alpha = 0.7, position = "dodge") +
  labs(title = "Comparison of Training NPV for SMOTE and ADASYN Sampling Methods",
       x = "Training NPV",
       y = "Frequency") +
  theme_minimal()

# Plot the distribution of Test NPV for "over" and "under" sampling methods
ggplot(combined_results, aes(x = TestNPV, fill = SamplingMethod)) +
  geom_histogram(data = subset(combined_results, SamplingMethod %in% c("over", "under")),
                 binwidth = 0.01, color = "black", alpha = 0.7, position = "dodge") +
  labs(title = "Comparison of Test NPV for Over and Under Sampling Methods",
       x = "Test NPV",
       y = "Frequency") +
  theme_minimal()

# Plot the distribution of Test NPV for "smote" and "adasyn" sampling methods
ggplot(combined_results, aes(x = TestNPV, fill = SamplingMethod)) +
  geom_histogram(data = subset(combined_results, SamplingMethod %in% c("smote", "adasyn")),
                 binwidth = 0.01, color = "black", alpha = 0.7, position = "dodge") +
  labs(title = "Comparison of Test NPV for SMOTE and ADASYN Sampling Methods",
       x = "Test NPV",
       y = "Frequency") +
  theme_minimal()

# Plot the distribution of Training Specificity for "over" and "under" sampling methods
ggplot(combined_results, aes(x = Spec, fill = SamplingMethod)) +
  geom_histogram(data = subset(combined_results, SamplingMethod %in% c("over", "under")),
                 binwidth = 0.01, color = "black", alpha = 0.7, position = "dodge") +
  labs(title = "Comparison of Training Specificity for Over and Under Sampling Methods",
       x = "Training Specificity",
       y = "Frequency") +
  theme_minimal()

# Plot the distribution of Training Specificity for "smote" and "adasyn" sampling methods
ggplot(combined_results, aes(x = Spec, fill = SamplingMethod)) +
```

```r
  geom_histogram(data = subset(combined_results, SamplingMethod %in% c("smote", "adasyn")),
                 binwidth = 0.01, color = "black", alpha = 0.7, position = "dodge") +
  labs(title = "Comparison of Training Specificity for SMOTE and ADASYN Sampling Methods",
       x = "Training Specificity",
       y = "Frequency") +
  theme_minimal()

# Plot the distribution of Test Specificity for "over" and "under" sampling methods
ggplot(combined_results, aes(x = TestSpecificity, fill = SamplingMethod)) +
  geom_histogram(data = subset(combined_results, SamplingMethod %in% c("over", "under")),
                 binwidth = 0.01, color = "black", alpha = 0.7, position = "dodge") +
  labs(title = "Comparison of Test Specificity for Over and Under Sampling Methods",
       x = "Test Specificity",
       y = "Frequency") +
  theme_minimal()

# Plot the distribution of Test Specificity for "smote" and "adasyn" sampling methods
ggplot(combined_results, aes(x = TestSpecificity, fill = SamplingMethod)) +
  geom_histogram(data = subset(combined_results, SamplingMethod %in% c("smote", "adasyn")),
                 binwidth = 0.01, color = "black", alpha = 0.7, position = "dodge") +
  labs(title = "Comparison of Test Specificity for SMOTE and ADASYN Sampling Methods",
       x = "Test Specificity",
       y = "Frequency") +
  theme_minimal()

### Testing Predictive Models

df <- df_clean

df$bankrupt <- factor(make.names(df$bankrupt))

# Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(df$bankrupt, p = 0.8, list = FALSE)
trainData <- df[trainIndex, ]
testData <- df[-trainIndex, ]

# Apply SMOTE sampling using a recipe
recipe_smote <- recipe(bankrupt ~ ., data = trainData) %>%
  step_smote(bankrupt, over_ratio = 1)

# Prepare the recipe and apply it to the training data
prep_recipe_smote <- prep(recipe_smote)
trainData_smote <- bake(prep_recipe_smote, new_data = NULL)

# Create trainControl to include specificity, sensitivity, and other metrics
cv_control <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 3,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = "final"  # Save the predictions for each resample
)


# Train MDA model #doesnt have hyperparameters
mda_model <- train(
  bankrupt ~ .,
```

```r
  data = trainData_smote,
  method = "mda",
  trControl = cv_control,
  metric = "ROC"
)

# Predict and evaluate the model
mda_predictions <- predict(mda_model, newdata = testData)
confusionMatrix(mda_predictions, testData$bankrupt)

confusionMatrix(predict(mda_model, newdata = trainData), trainData$bankrupt)

# Train SVM model with hyperparameter tuning
svm_grid <- expand.grid(C = c(0.1, 1, 10))

svm_model <- train(
  bankrupt ~ .,
  data = trainData_smote,
  method = "svmLinear",
  trControl = cv_control,
  metric = "ROC",  # You can use "ROC" as a metric for binary classification
  tuneGrid = svm_grid
)

# Predict and evaluate the model
svm_predictions <- predict(svm_model, newdata = testData)
confusionMatrix(svm_predictions, testData$bankrupt)

confusionMatrix(predict(svm_model, newdata = trainData), trainData$bankrupt)

# Train KNN model with hyperparameter tuning
knn_grid <- expand.grid(k = c(3, 5, 7, 9))

knn_model <- train(
  bankrupt ~ .,
  data = trainData_smote,
  method = "knn",
  trControl = cv_control,
  metric = "ROC",  # You can use "ROC" as a metric for binary classification
  tuneGrid = knn_grid
)

# Predict and evaluate the model
knn_predictions <- predict(knn_model, newdata = testData)
confusionMatrix(knn_predictions, testData$bankrupt)

confusionMatrix(predict(knn_model, newdata = trainData), trainData$bankrupt)

# Train Logistic Regression model
lr_model <- train(
  bankrupt ~ .,
  data = trainData_smote,
  method = "glm",
  family = binomial,
  trControl = cv_control,
  metric = "ROC"  # You can use "ROC" as a metric for binary classification
)

# Predict and evaluate the model
```

```r
lr_predictions <- predict(lr_model, newdata = testData)
confusionMatrix(lr_predictions, testData$bankrupt)

confusionMatrix(predict(lr_model, newdata = trainData), trainData$bankrupt)

# Train Random Forest model with hyperparameter tuning
rf_grid <- expand.grid(mtry = c(2, 4, 6, 8))

rf_model <- train(
  bankrupt ~ .,
  data = trainData_smote,
  method = "rf",
  trControl = cv_control,
  metric = "ROC",  # You can use "ROC" as a metric for binary classification
  tuneGrid = rf_grid,
  ntree = 500  # Number of trees
)

# Predict and evaluate the model
rf_predictions <- predict(rf_model, newdata = testData)
confusionMatrix(rf_predictions, testData$bankrupt)

confusionMatrix(predict(rf_model, newdata = trainData), trainData$bankrupt)

#Check Cross-validation results

# Extract specificity from the resample object
specificity_values_lr <- lr_model$resample[, "Spec"]
print("Logistic Regression - Specificity for each fold:")

print(specificity_values_lr)

# Extract specificity from the resample object
print("Random Forest - Specificity for each fold:")

print(specificity_values_rf)

# Extract specificity from the resample object
specificity_values_mda <- mda_model$resample[, "Spec"]
print("Multiple Discriminant Analysis - Specificity for each fold:")

print(specificity_values_mda)

# Extract specificity from the resample object
specificity_values_svm <- svm_model$resample[, "Spec"]
print("Support Vector Machines - Specificity for each fold:")

print(specificity_values_svm)

# Extract specificity from the resample object
specificity_values_knn <- knn_model$resample[, "Spec"]
print("K-Nearest Neighbor - Specificity for each fold:")

## [1] "K-Nearest Neighbor - Specificity for each fold:"

print(specificity_values_knn)

#Checking distribution of specificty for Cross-validation

# Create a data frame from the specificity values
specificity_df_knn <- data.frame(Fold = 1:length(specificity_values_knn), Specificity = specif
```

```
icity_values_knn)

# Create the barplot using ggplot2
ggplot(specificity_df_knn, aes(x = factor(Fold), y = Specificity)) +
  geom_bar(stat = "identity", fill = "skyblue", color = "black") +
  labs(title = "Specificity for Each Cross-Validation Fold (KNN Model)",
       x = "Cross-Validation Fold",
       y = "Specificity") +
  coord_cartesian(ylim = c(0.9, 0.96)) +
  theme_minimal()

# Create a data frame from the specificity values
specificity_df_mda <- data.frame(Fold = 1:length(specificity_values_mda), Specificity = specif
icity_values_mda)

# Create the barplot using ggplot2
ggplot(specificity_df_mda, aes(x = factor(Fold), y = Specificity)) +
  geom_bar(stat = "identity", fill = "skyblue", color = "black") +
  labs(title = "Specificity for Each Cross-Validation Fold (MDA Model)",
       x = "Cross-Validation Fold",
       y = "Specificity") +
  coord_cartesian(ylim = c(0.7, 0.81))

theme_minimal()

# Create a data frame from the specificity values
specificity_df_svm <- data.frame(Fold = 1:length(specificity_values_svm), Specificity = specif
icity_values_svm)

# Create the barplot using ggplot2
ggplot(specificity_df_svm, aes(x = factor(Fold), y = Specificity)) +
  geom_bar(stat = "identity", fill = "skyblue", color = "black") +
  labs(title = "Specificity for Each Cross-Validation Fold (SVM Model)",
       x = "Cross-Validation Fold",
       y = "Specificity") +
  coord_cartesian(ylim = c(0.825, 0.9125))+
  theme_minimal()

# Create a data frame from the specificity values
specificity_df_rf <- data.frame(Fold = 1:length(specificity_values_rf), Specificity = specific
ity_values_rf)

# Create the barplot using ggplot2
ggplot(specificity_df_rf, aes(x = factor(Fold), y = Specificity)) +
  geom_bar(stat = "identity", fill = "skyblue", color = "black") +
  labs(title = "Specificity for Each Cross-Validation Fold (RF Model)",
       x = "Cross-Validation Fold",
       y = "Specificity") +
  coord_cartesian(ylim = c(0.975, 0.9975))+
  theme_minimal()

# Create a data frame from the specificity values
specificity_df_lr <- data.frame(Fold = 1:length(specificity_values_lr), Specificity = specific
ity_values_lr)

# Create the barplot using ggplot2
ggplot(specificity_df_lr, aes(x = factor(Fold), y = Specificity)) +
  geom_bar(stat = "identity", fill = "skyblue", color = "black") +
  labs(title = "Specificity for Each Cross-Validation Fold (LR Model)",
       x = "Cross-Validation Fold",
```

```r
       y = "Specificity") +
  coord_cartesian(ylim = c(0.85, 0.91))+
  theme_minimal()

knitr::opts_chunk$set(fig.show = "hide", echo = TRUE)

### Calculating evaluation metrics

#Function to calculate F2 Score
calculate_f2_score <- function(conf_matrix) {
  precision <- conf_matrix$byClass["Pos Pred Value"]
  recall <- conf_matrix$byClass["Sensitivity"]
  f2_score <- 5 * ((precision * recall) / ((4 * precision) + recall))
  return(f2_score)
}

# MDA Model - F2 Score and AUC
mda_probabilities <- predict(mda_model, newdata = testData, type = "prob")
mda_conf_matrix <- confusionMatrix(mda_predictions, testData$bankrupt)
f2_score_mda <- calculate_f2_score(mda_conf_matrix)
roc_curve_mda <- roc(testData$bankrupt, mda_probabilities[, 2])
auc_mda <- auc(roc_curve_mda)

cat("MDA Model F2 Score:", f2_score_mda, "\n")

cat("MDA Model AUC:", auc_mda, "\n")

# SVM Model - F2 Score and AUC
svm_probabilities <- predict(svm_model, newdata = testData, type = "prob")
svm_conf_matrix <- confusionMatrix(svm_predictions, testData$bankrupt)
f2_score_svm <- calculate_f2_score(svm_conf_matrix)
roc_curve_svm <- roc(testData$bankrupt, svm_probabilities[, 2])
auc_svm <- auc(roc_curve_svm)

cat("SVM Model F2 Score:", f2_score_svm, "\n")

cat("SVM Model AUC:", auc_svm, "\n")

# KNN Model - F2 Score and AUC
knn_probabilities <- predict(knn_model, newdata = testData, type = "prob")
knn_conf_matrix <- confusionMatrix(knn_predictions, testData$bankrupt)
f2_score_knn <- calculate_f2_score(knn_conf_matrix)
roc_curve_knn <- roc(testData$bankrupt, knn_probabilities[, 2])
auc_knn <- auc(roc_curve_knn)

cat("KNN Model F2 Score:", f2_score_knn, "\n")

cat("KNN Model AUC:", auc_knn, "\n")

# Logistic Regression Model - F2 Score and AUC
lr_probabilities <- predict(lr_model, newdata = testData, type = "prob")
lr_conf_matrix <- confusionMatrix(lr_predictions, testData$bankrupt)
f2_score_lr <- calculate_f2_score(lr_conf_matrix)
roc_curve_lr <- roc(testData$bankrupt, lr_probabilities[, 2])
auc_lr <- auc(roc_curve_lr)

cat("Logistic Regression Model F2 Score:", f2_score_lr, "\n")

cat("Logistic Regression Model AUC:", auc_lr, "\n")
```

```r
# Random Forest Model - F2 Score and AUC
rf_probabilities <- predict(rf_model, newdata = testData, type = "prob")
rf_conf_matrix <- confusionMatrix(rf_predictions, testData$bankrupt)
f2_score_rf <- calculate_f2_score(rf_conf_matrix)
roc_curve_rf <- roc(testData$bankrupt, rf_probabilities[, 2])
auc_rf <- auc(roc_curve_rf)

cat("Random Forest Model F2 Score:", f2_score_rf, "\n")

cat("Random Forest Model AUC:", auc_rf, "\n")

### Evaluating SVM (Best Model)

# Define a range of values for the cost parameter (C) to match the caret example
cost_values <- c(0.1, 1, 10)

# Perform hyperparameter tuning using cross-validation
set.seed(123)

svm_model_2 <- tune.svm(
  bankrupt ~ .,
  data = trainData_smote,
  kernel = "linear",
  scale = TRUE,  # Scale the data
  probability = TRUE,
  cost = cost_values,  # Grid of C values to test
  tunecontrol = tune.control(sampling = "cross", cross = 10)  # 10-fold cross-validation
)

# View the results
print(svm_model_2)

# Best model found
best_svm_model <- svm_model_2$best.model

# Summary of the best model
summary(best_svm_model)

# Use the best model to make predictions on the test set
test_predictions <- predict(best_svm_model, newdata = testData, probability = TRUE)

# Evaluate the performance on the test set
conf_matrix <- confusionMatrix(test_predictions, testData$bankrupt)
print(conf_matrix)

# Coefficients (weights) for the support vectors
svm_coefficients <- t(best_svm_model$coefs) %*% best_svm_model$SV

# Print the coefficients
print(svm_coefficients)

# Intercept (bias term)
svm_intercept <- -best_svm_model$rho
print(svm_intercept)

levels(trainData_smote$bankrupt)

#X0 is positive class, not bankrupt
#X1 is negative class, bankrupt
```

```r
# Print the summary of the tuning process
summary(svm_model_2)

# Extract cross-validation results
cv_results <- svm_model_2$performances

# View the structure of the cross-validation results
str(cv_results)

# Visualize cross-validation results (e.g., Accuracy or ROC across different values of C)
ggplot(cv_results, aes(x = factor(cost), y = error, group = cost)) +
  geom_point() +
  geom_line() +
  labs(title = "Cross-Validation Results (Error Rate)",
       x = "Cost (C)",
       y = "Error Rate") +
  theme_minimal()

# Predict class probabilities on the test set
test_probabilities <- attr(test_predictions, "probabilities")

str(test_probabilities)

print(test_probabilities)

#Create ROC curve

roc_curve <- roc(testData$bankrupt, test_probabilities[,2], levels = rev(levels(testData$bankr
upt)))

# Calculate False Negative Rate (FNR) and True Negative Rate (TNR)
fnr <- 1 - roc_curve$sensitivities
tnr <- roc_curve$specificities

# Plot TNR vs FNR
plot(fnr, tnr, type = "l", col = "blue",
     xlab = "False Negative Rate (FNR)",
     ylab = "True Negative Rate (TNR)",
     main = "ROC Curve with TNR vs FNR")
abline(a = 0, b = 1, col = "red", lty = 2)  # Add diagonal line for referenced diagonal line f
or reference

auc_value <- auc(roc_curve)
cat("AUC:", auc_value, "\n")

###Creating Decision Boundary plot for SVM

#Perform PCA on the training data
pca <- prcomp(trainData_smote[, -which(names(trainData_smote) == "bankrupt")], center = TRUE,
scale. = TRUE)

#Create a new dataset with the first two principal components
trainData_smote_pca <- data.frame(
  PC1 = pca$x[, 1],
  PC2 = pca$x[, 2],
  bankrupt = trainData_smote$bankrupt
)
```

```r
#Use the best hyperparameters found during tunings
best_C <- svm_model_2$best.parameters$cost  # Extracting the best cost (C) from your tuned mod
el

#Train the SVM model using the first two principal components with the best C value
best_svm_model_pca <- svm(
  bankrupt ~ .,
  data = trainData_smote_pca,
  kernel = "linear",
  cost = best_C,  # Use the best C from tuning
  scale = TRUE,
  probability = TRUE
)

#Create a grid of values covering the feature space
x_min <- min(trainData_smote_pca$PC1) - 1
x_max <- max(trainData_smote_pca$PC1) + 1
y_min <- min(trainData_smote_pca$PC2) - 1
y_max <- max(trainData_smote_pca$PC2) + 1

grid_values <- expand.grid(
  PC1 = seq(x_min, x_max, length.out = 100),
  PC2 = seq(y_min, y_max, length.out = 100)
)

# Predict over the grid using the best SVM model
grid_predictions <- predict(best_svm_model_pca, grid_values, probability = TRUE)

# Add predictions to the grid
grid_values$Prediction <- grid_predictions

#Plot the decision boundary
ggplot() +
  geom_tile(data = grid_values, aes(x = PC1, y = PC2, fill = Prediction), alpha = 0.3) +
  geom_point(data = trainData_smote_pca, aes(x = PC1, y = PC2, color = bankrupt), size = 1, al
pha = 2) +
  labs(title = "SVM Decision Boundary (PCA-Reduced Features with Tuned Model)",
       x = "Principal Component 1 (PC1)",
       y = "Principal Component 2 (PC2)") +
  scale_fill_manual(values = c("X0" = "lightblue", "X1" = "yellow")) +  # Customize colors if
necessary
  theme_minimal()

pca_loadings <- pca$rotation

# Bar plot of loadings for the first principal component (PC1)
loading_plot_1 <- ggplot(as.data.frame(pca_loadings), aes(x = rownames(pca_loadings), y = PC1)
) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "PCA Loadings for PC1", x = "Variables", y = "Loadings") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(loading_plot_1)

# Bar plot of loadings for the second principal component (PC2)
loading_plot_2 <- ggplot(as.data.frame(pca_loadings), aes(x = rownames(pca_loadings), y = PC2)
) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "PCA Loadings for PC2", x = "Variables", y = "Loadings") +
  theme_minimal() +
```

```
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(loading_plot_2)

# Scatter plot of loadings for PC1 vs PC2
loading_scatter <- ggplot(as.data.frame(pca_loadings), aes(x = PC1, y = PC2, label = rownames(
pca_loadings))) +
  geom_point() +
  geom_text(vjust = 1.5, hjust = 1.5) +
  labs(title = "PCA Loadings for PC1 and PC2", x = "PC1 Loadings", y = "PC2 Loadings") +
  theme_minimal()
print(loading_scatter)

# Plot the proportion of variance explained by each principal component
explained_variance <- pca$sdev^2 / sum(pca$sdev^2)

variance_plot <- qplot(1:length(explained_variance), explained_variance, geom = "line") +
  geom_point() +
  labs(title = "Scree Plot", x = "Principal Component", y = "Proportion of Variance Explained"
) +
  scale_x_continuous(breaks = 1:length(explained_variance)) +  # Set x-axis labels as 1, 2, 3,
etc.
  theme_minimal()
print(variance_plot)

# Calculate the cumulative variance explained for principal components
cumulative_variance <- cumsum(explained_variance)

# Create a data frame with the PCA components, explained variance, and cumulative variance
pca_variance_table <- data.frame(
  PCA = paste0("PC", 1:length(explained_variance)),
  Variance_Explained = explained_variance,
  Cumulative_Variance_Explained = cumulative_variance
)

# Display the cumulative variance explained table
print(pca_variance_table)

### IML

#PDP and ALE

#Creating a predictor for IML
predictor <- Predictor$new(
  model = best_svm_model,
  data = trainData_smote[, !colnames(trainData_smote) %in% "bankrupt"],
  y = trainData_smote$bankrupt
)

str(predictor)

## Create and plot Partial Dependency Plot (PDP) for each variable

# For variable 'AN'
pdp_AN <- FeatureEffect$new(predictor, feature = "AN", method = "pdp")
plot(pdp_AN)

# For variable 'CH'
pdp_CH <- FeatureEffect$new(predictor, feature = "CH", method = "pdp")
plot(pdp_CH)
```

```r
# For variable 'N'
pdp_N <- FeatureEffect$new(predictor, feature = "N", method = "pdp")
plot(pdp_N)

# For variable 'AJ'
pdp_AJ <- FeatureEffect$new(predictor, feature = "AJ", method = "pdp")
plot(pdp_AJ)

# For variable 'CB'
pdp_CB <- FeatureEffect$new(predictor, feature = "CB", method = "pdp")
plot(pdp_CB)

# For variable 'AW'
pdp_AW <- FeatureEffect$new(predictor, feature = "AW", method = "pdp")
plot(pdp_AW)

# For variable 'BD'
pdp_BD <- FeatureEffect$new(predictor, feature = "BD", method = "pdp")
plot(pdp_BD)

# For variable 'BS'
pdp_BS <- FeatureEffect$new(predictor, feature = "BS", method = "pdp")
plot(pdp_BS)

# For variable 'CQ'
pdp_CQ <- FeatureEffect$new(predictor, feature = "CQ", method = "pdp")
plot(pdp_CQ)

# For variable 'CL'
pdp_CL <- FeatureEffect$new(predictor, feature = "CL", method = "pdp")
plot(pdp_CL)

##  Create and Plot Accumulated Local Effects (ALE) for each variable

# Create the ALE for 'AN'
ale_AN <- FeatureEffect$new(predictor, feature = "AN")
plot(ale_AN)

# Create the ALE for 'CH'
ale_CH <- FeatureEffect$new(predictor, feature = "CH")
plot(ale_CH)

# Create the ALE for 'N'
ale_N <- FeatureEffect$new(predictor, feature = "N")
plot(ale_N)

# Create the ALE for other variables
ale_AJ <- FeatureEffect$new(predictor, feature = "AJ")
ale_CB <- FeatureEffect$new(predictor, feature = "CB")
ale_AW <- FeatureEffect$new(predictor, feature = "AW")
ale_BD <- FeatureEffect$new(predictor, feature = "BD")
ale_BS <- FeatureEffect$new(predictor, feature = "BS")
ale_CQ <- FeatureEffect$new(predictor, feature = "CQ")
ale_CL <- FeatureEffect$new(predictor, feature = "CL")

# Plot the ALE Plots
plot(ale_AJ)

plot(ale_CB)

plot(ale_AW)
```

```r
plot(ale_BD)

plot(ale_BS)

plot(ale_CQ)

plot(ale_CL)
```