SBA LOAN APPROVAL ANALYSIS

Timotea Tetakova (3079127)

Aryan Mishra (3072150)

Alex Kubbinga (3075300)

Rossana Tatulli (3074750)

Luca Giglioli (3067159)

BACKGROUND INFORMATION

The dataset is a real dataset from the U.S. Small Business Administration (SBA), containing historical data of small business loans approved from 1987 to 2014 with 27 variables and 899,164 observations. The outcome of each loan in the dataset is known, meaning it is known whether the loan defaulted or whether it was paid in full.

- SBA is a government agency found in 1953 with the goal of promoting and providing support for small businesses in the US credit market.

- Through a loan guarantee program, designed to incentivize banks to grant loans to small enterprises, the SBA acts like an insurance provider. It takes part of the risk, so that if the loan goes into default, SBA covers the amount they guaranteed.

GENERAL OVERVIEW:
The research question

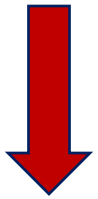
As the SBA guarantees only a portion of the entire loan, banks still incur losses in case of default and, consequently, are still faced with a difficult choice of whether or not to grant the loan.

Therefore, our analysis aims to address the following question through an algorithmic approach

"Based on the various risk factors entailed, should an SBA loan be approved or not?"

GENERAL OVERVIEW: Motivation

We believe that conducting this analysis, which leverages historical data to make informed decisions, can be a significant contribution especially given the ongoing pandemic. With small businesses currently needing more support/loans, a model to accurately predict loan default would be very beneficial for a bank in order to reduce potential charged off loans.



Why small businesses matter?

Fostering small business formation and growth has several benefits, as they are a vital component of the economy. Aside from contributions to the general economic well-being, small businesses also contribute to growth and vitality in specific areas of economic and socio-economic development. They reduce unemployment by creating job opportunities and spark innovation by fostering environments that appeal to individuals with the talent to invent new products. In addition, they usually complement the economic activity of large organizations by providing them with components, services, and distribution of their products.

Purpose of the Analysis & Model

- The model developed will be able to predict whether a small business will default on its loan.
- It learns by analyzing and training on the dataset of prior loans administered by the SBA from 2000 onwards.
- It will be able to classify future loans as defaulting or non-defaulting based on the firm's characteristics.

STEPS INVOLVED

1. UNDERSTANDING OF THE DATASET

2. DATA PREPARATION, CLEANING AND EXPLORATION:
- Identification of variables that may be good predictors or risk indicators of the level of risk associated with a loan
- Data exploration : statistics and visualization
- Missing data management
- Outlier treatment
- Creation of new relevant variables
- Variable transformation

3. MODEL BUILDING AND EVALUATION
- Logistic Regression Implementation & Evaluation
- Artifical Neural Network Implementation & Evaluation
- Gradient Boosted Trees Implementation & Evaluation

4. MODEL COMPARISON:
- Comparison of the three models' performance based on relevant metrics to choose the best model

5. MANAGERIAL IMPLICATIONS AND FURTHER INSIGHTS

Variable Name	Data Type	Description
LoanNr_ChkDgt	Text	Identifier- primary key
Name	Text	Borrower name
City	Text	Borrower City
State	Text	Borrower state
Zip	Text	Borrower zip code
Bank	Text	Bank name
BankState	Text	Bank state
NAICS	Text	North American industry classification system code
ApprovalDate	Date/Time	Date SBA commitment issued
ApprovalFY	Text	Fiscal year of commitment
Term	Number	Loan term in months
NoEmp	Number	Number of business employees
NewExist	Binary	1= Existing business, 0 = New business
CreateJob	Number	Number of jobs created
RetainedJob	Number	Number of jobs retained
FranchiseCode	Text	Franchise code, (00000 or 00001)= No franchise
UrbanRural	Text	1=Urban,2=rural,0=undefined
RevLineCr	Text	Revolving line of credit: Y=yes,N=no
LowDoc	Text	LowDoc loan program Y/N
ChgOffDate	Date/Time	Date the loan is declared to be in default
DisbursementDate	Date/Time	Disbursement date
DisbursementGross	Currency	Amount disbursed
BalanceGross	Currency	Gross amount outstanding
MIS_Status	Text	Loan charged off= CHGOFF, Paid in full=PIF
ChgOffPrinGr	Currency	Charged off amount
GrAppv	Currency	Gross amount of loan approved by bank
SBA_Appv	Currency	SBA's guaranteed amount of approved loan

Identifying Explanatory Variables

To the left is a table of all the original variables in the SBA dataset.

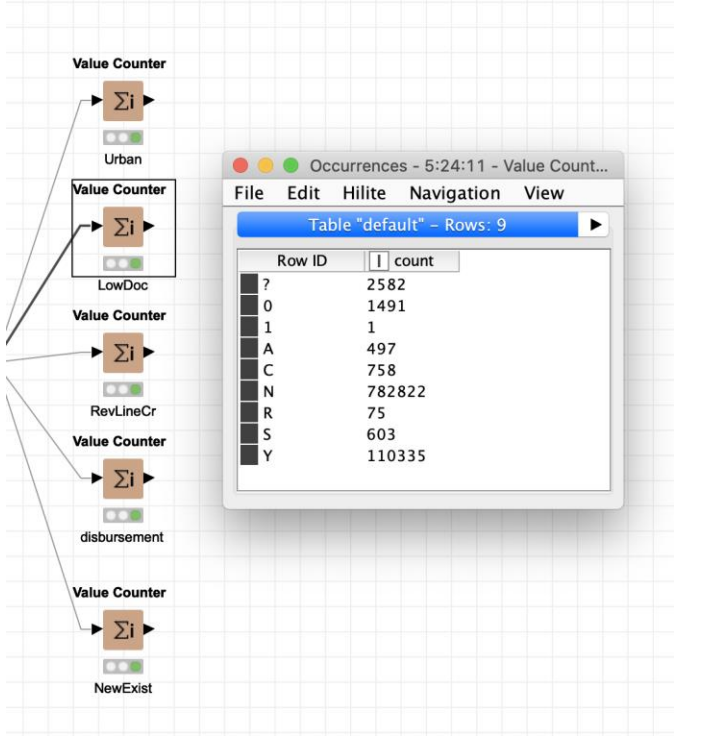
The variables in the yellow rows are the explanatory variables that can be good indicators of the level of risk associated with a small business loan. Therefore, the columns were manually filtered to include these variables in the analysis.

Additionally, our analysis included only observations starting from the year 2000, as small businesses have greatly evolved since the previous decades and have been increasingly successful. This was to obtain a better prediction for loans issued to small businesses under more recent circumstances.

The variables that had a currency type were converted to strings, and the DisbursementDate variable was used to create another variable containing only the Disbursement year.

Data Errors & Missing Values

Figure: Data entry errors

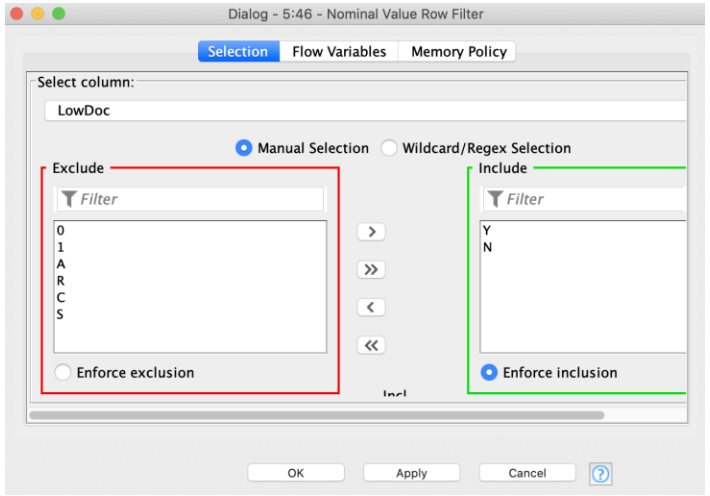


The variables were also investigated using the Value Counter node to see all the values that the variables take. This is particularly useful with categorical variables. Any instances with missing values or Data entry errors as shown in the figure were removed (Y,N remained).

Missing values occurred for certain variables. RevLineCr (a variable that is Y if the business had a revolving line of credit) had around 300,000 rows of missing data. Since it is not possible to identify ex-post whether a firm with a missing RevLineCr had a revolving line of credit, we decided to remove the variable entirely. This is because we did not want to replace it with Y or N and risk compromising the integrity of the data.

A similar approach was taken with the UrbanRural variable which also had 300k+ undefined or missing values.

With other variables (MIS_Status, LowDoc, NewExist) which only had 1-5 thousand missing values, we decided to simply remove those rows.



Additional Variables Generated for Our Analysis

Variable Name	Data Type	Description
Default (TARGET VARIABLE)	Number	0 = Did not default, 1= Defaulted
RealEstateB	Number	0 = Loan not backed by real estate, 1 = Loan backed by real estate
Recession	Number	1 = Loan active during Great Recession, 0= not active
Industry	String	Industry Name (Ex: "Retail and Trade")
SBA/Gross (% covered)	Number	SBA's Guaranteed Portion of Approved Loan = SBA_Appv/GrAappv

Default: This is the dependent variable and was created as a dummy variable from MIS_Status (1 is a defaulted loan).

RealEstateB: An additional risk indicator to observe whether the loan is backed by real estate. This is because oftentimes, the value of land in possession can be large enough to cover an amount of the loan, thus the probability of default may decrease. Loans backed by real estate have terms equal to or greater than 240 months, while those not backed by real estate will have terms of less than 240 months.

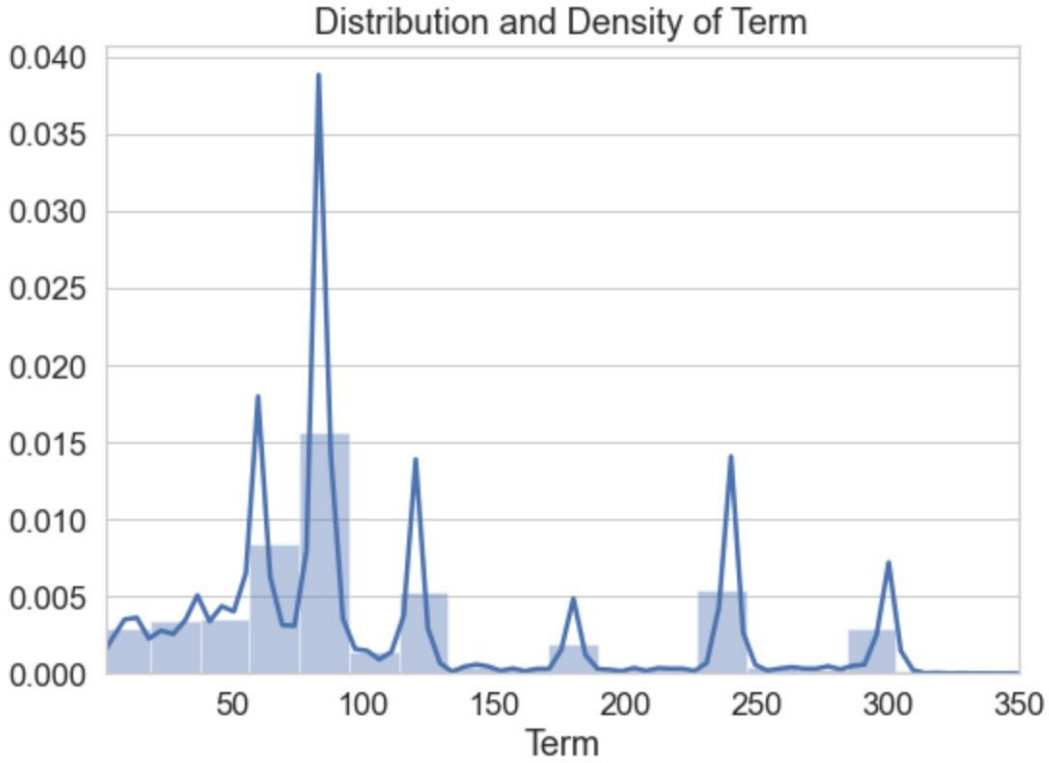
Recession: Small business loans can have a higher chance of defaulting during a recession since they are generally affected by the economy. Hence the Recession variable is equal to 1 only if the loans were active from December 2007 to June 2009. (Great Recession)

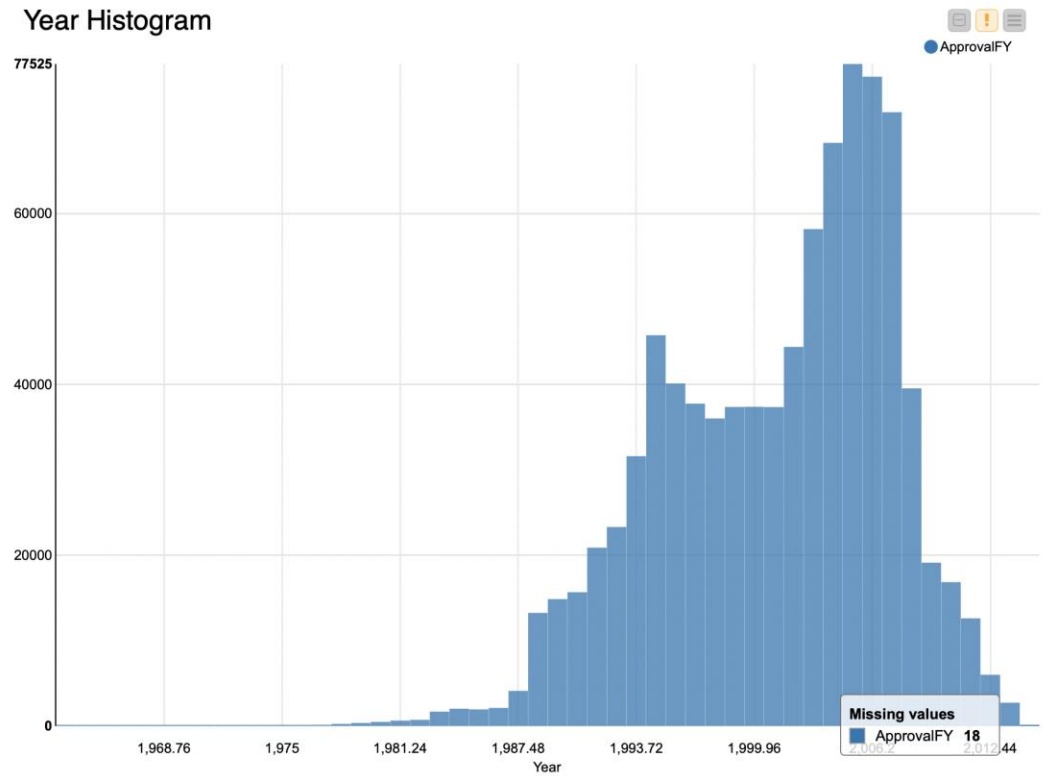
Industry: The first two digits of the NAICS code were used to show the industry, as the probability of default could be affected by the nature of certain industries. This code was then transformed into a string variable with the industry name to improve interpretability.

SBA/Gross: Shows the percentage of the loan that SBA covered. Created to observe whether a loan is more likely to be default based on the amount of it that the SBA covers.

Data visualization & further exploration

- We also decided to visualize the data in order to better understand the dataset and identify missing values or outliers in the numerical variables.





Data Visualization Continued

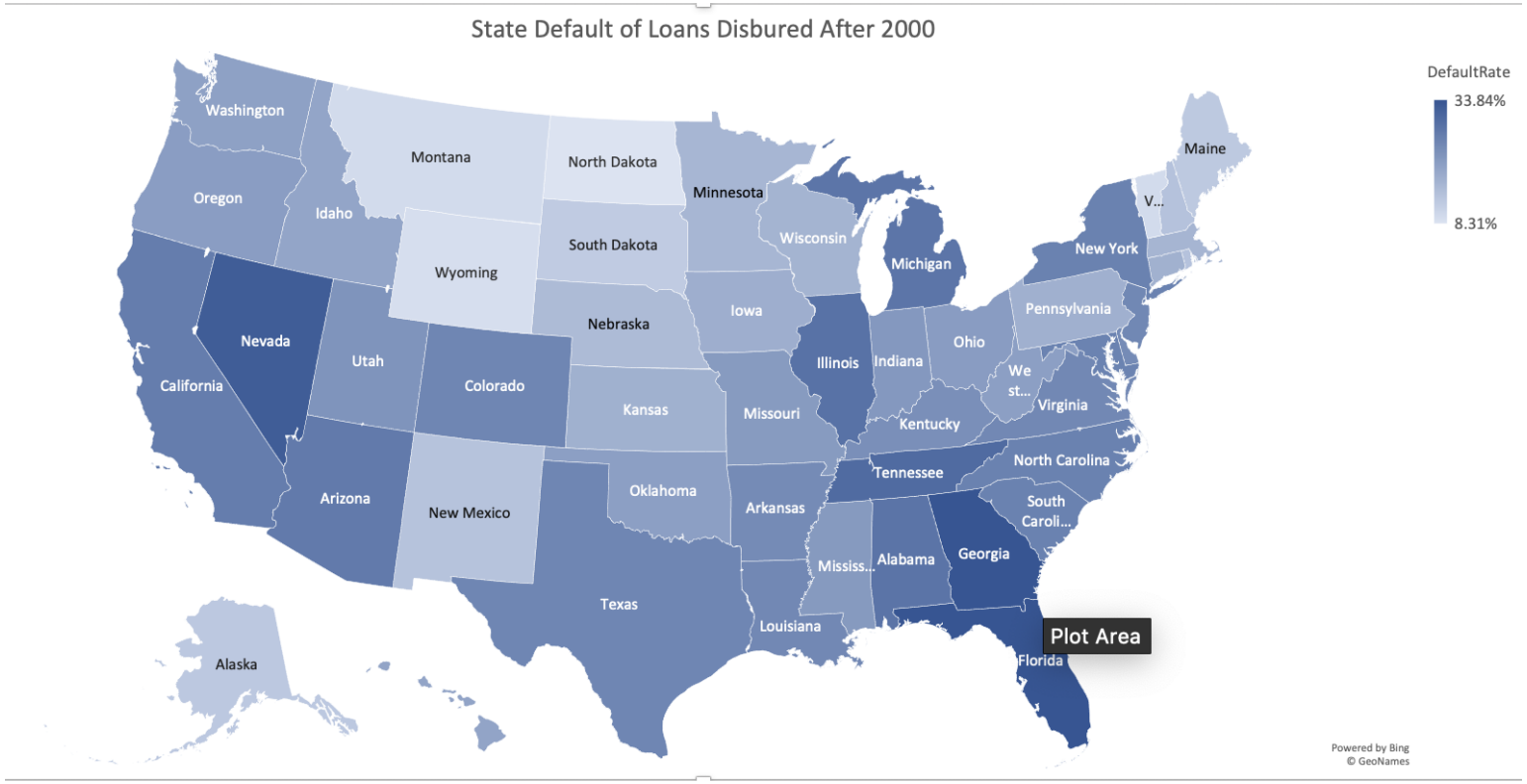
- The figure on the right shows the true default rate by industry as seen from the data. As we can see the industries of Public Administration, Finance and Insurance and Real Estate and Leasing had the highest default rates. This can be useful to know as loan officers could pay more attention when deciding to approve loans for these industries.

Industries and their default rates

	Industry	DefaultRate
0	Public Administration	32.61%
1	Real Estate and Rental and Leasing	31.87%
2	Finance and Insurance	29.25%
3	Information	28.32%
4	Transportation and Warehousing	27.94%
5	Construction	27.87%
6	Retail Trade	27.37%
7	Administrative and Support and Waste Managemen...	25.53%
8	Wholesale Trade	25.18%
9	Arts, Entertainment, and Recreation	24.95%
10	Educational Services	24.81%
11	Other Services (except Public Administration)	24.56%
12	AverageIndustry	23.67%
13	Accommodation and Food Services	22.97%
14	Professional, Scientific, and Technical Services	22.65%
15	Manufacturing	19.19%
16	Utilities	18.74%
17	Management of Companies and Enterprises	16.35%
18	Agriculture, Forestry, Fishing and Hunting	13.47%
19	Health Care and Social Assistance	13.17%
20	Mining, Quarrying, and Oil and Gas Extraction	10.85%
21	Unkown Industry	10.73%

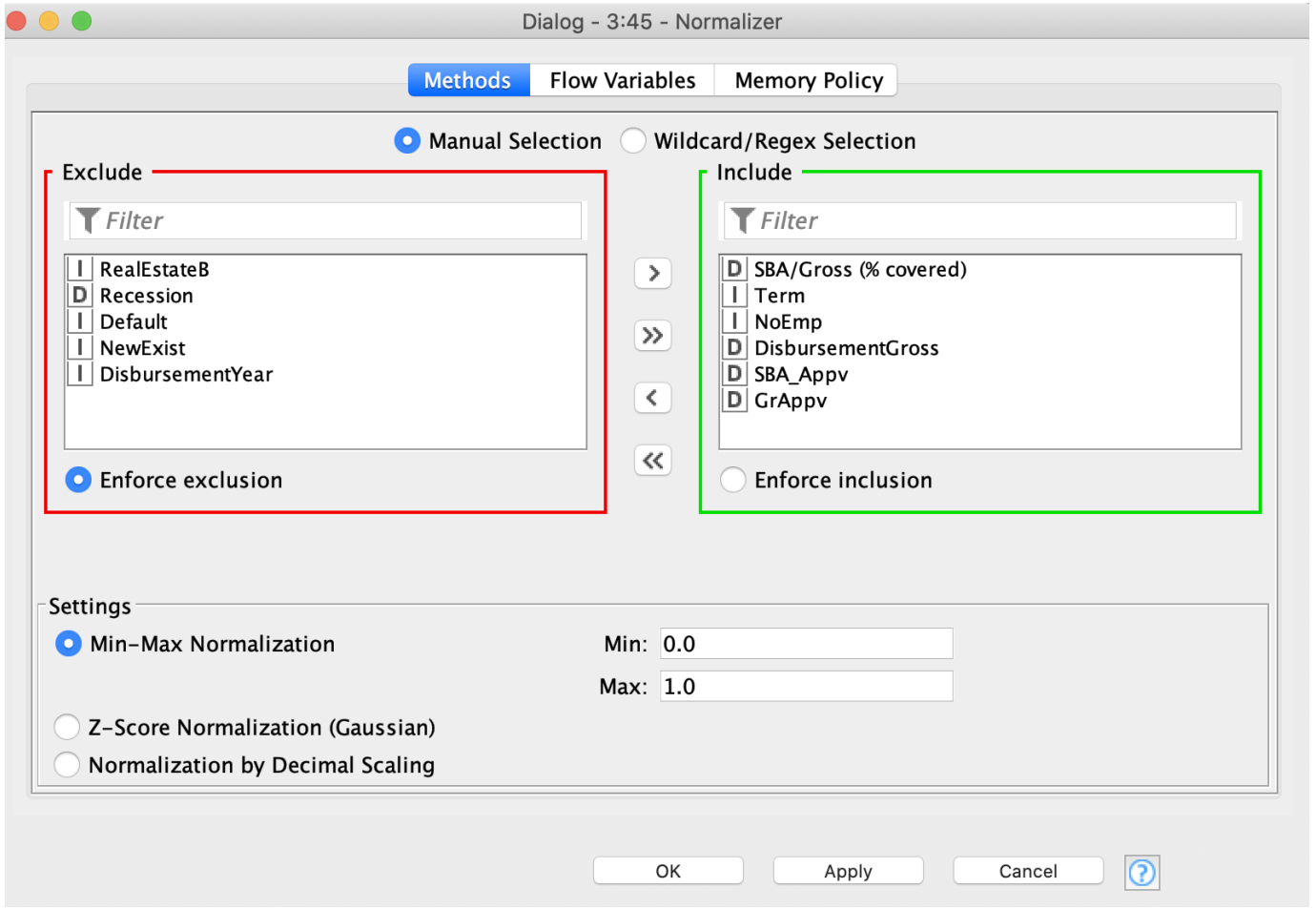
State Map of Default Rates Visualization

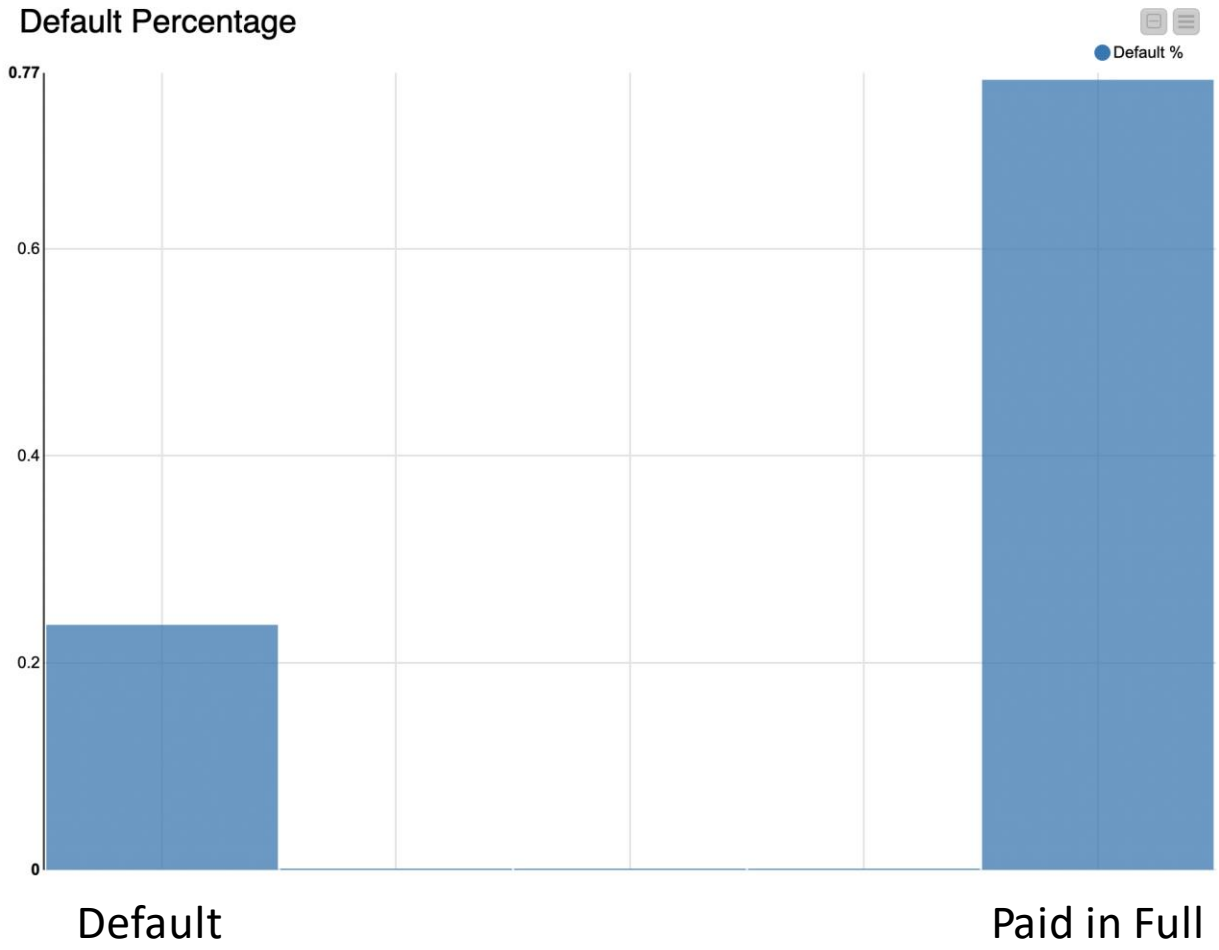
The visualization shows the states colored by their default rates with the darker shaded states having higher default rates. As we can see from the visualization the states of Florida, Georgia and Nevada had some of the highest default rates. North Dakota, Montana, Wyoming had the lowest. Some states may have higher defaults due to their industry composition and the economic situation during the period of 2000-2014. Although not entirely transferable to today, due to different industry and economic situations in each state, it is advisable that loan officers in the states with the highest default rates be more cautious when distributing loans.



Normalization

- The relevant numerical variables were normalized in order to create a common range for the variables. This is important as some of the variables had different ranges. For example, disbursement gross has a range between 50,000 to millions and SBA/Gross between 0 and 100.
- This helps improve model performance and reduce the time taken for gradient descent to converge.





Row ID	count	Total	Default %
0	430387	563,856	0.763
1	133469	563,856	0.237

Default Imbalance Visualization

As we can see from the histogram on the left, we have a significant imbalance in our target class. Specifically, the percentage of businesses that paid the loan in full is much higher at 76.3% compared to the ones that defaulted at 23.7%. Therefore, before proceeding to the modelling stage, we must be more careful when evaluating the performance of our predictive models. Accuracy will no longer be a relevant metric to evaluate model performance due to the "accuracy paradox" - cases when accuracy values are artificially high, but the accuracy is only reflecting the underlying class distribution. Therefore, to combat class imbalance, we decided to change our performance metric to the F-Score (or F-Measure on KNIME) - a weighted average of precision and recall.

Motivation behind the use of F-Measure

Class imbalance is not the sole reason for which we chose F-Score as our main performance metric. If we decide to place ourselves in the shoes of a loan officer, we would be highly focused on avoiding too many false negatives – instances where we incorrectly predict that the company will not default when it ends up defaulting. Therefore, it would be better to have a high recall (Sensitivity) as the bank wouldn't want to lose money.

Furthermore, it would be a good idea to alarm the bank even if there is a slight doubt about a defaulter. Avoiding too many false positives (when the bank incorrectly predicts a default when in fact the client would pay back in full) is also important since they represent opportunity costs for the bank, which is why we should take precision into account too. Thus, the F-Measure is used to evaluate model performance as it takes both precision and recall into account

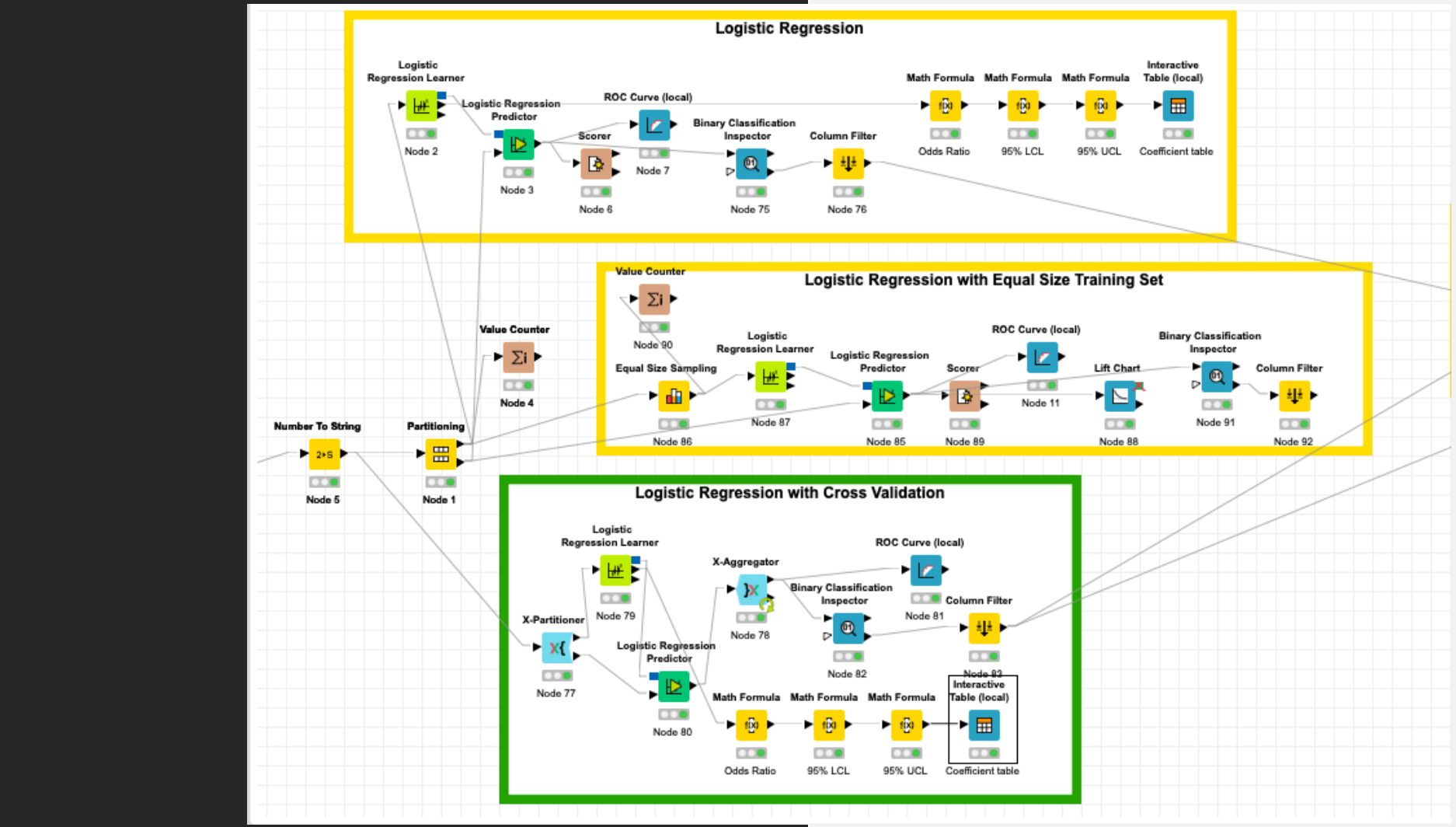
Now that we have our performance metric, we proceed to the modelling stage.

Model 1 – Logistic Regression

Logistic Regression – Model Overview

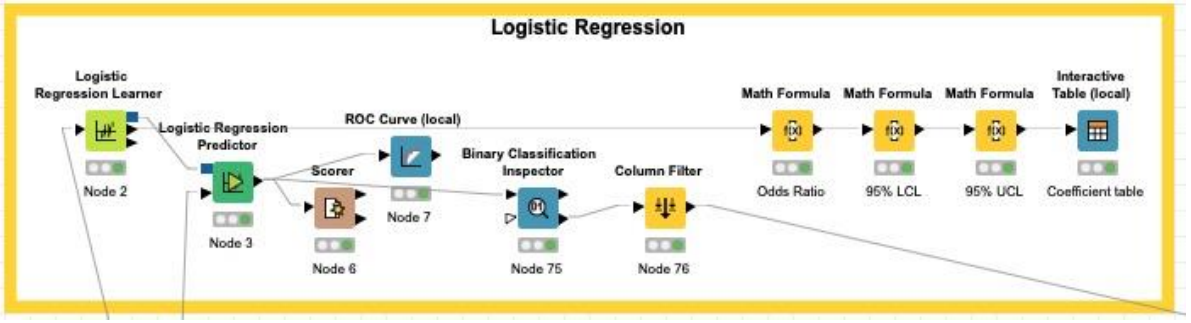
- Logistic Regression is a statistical model that uses, in its most basic form, a logistic function to model a binary dependent variable.

- Logistic Regression can either be binomial, ordinal or multinomial. In our case, we have a binomial logistic regression since our dependent variable is "DEFAULT" which can only be encoded by two values – 0 and 1, where 1 is the "positive" class that implies the occurrence of a default on an SBA-backed loan and 0 implies the loan was paid in full.



Model 1 – Logistic Regression workflow

Logistic Regression



After the dataset was partitioned into 70% training and 30% testing sets, we compared two different variations of the Logistic Regression model to choose a candidate model that can then be compared to our other two main models – Gradient Boosting and Multilayer Neural Networks. Specifically, our first variation of the Logistic Regression model (as shown in the picture above) is a regular logistic regression model without any cross-validation. However, it has a few slight modifications. In particular, the most notable ones are as follows:

- Stochastic Gradient Descent was used as the iterative optimization method rather than the Newton-Raphson Algorithm (iteratively reweighted least squares on KNIME) since the latter takes a long time per iteration and is memory intensive. To be more precise, Newton's Method requires computing the second derivative, H, which is $O(N\text{^}2)$, where N is the number of features, while computing the gradient, g, is only $O(N)$. Therefore, the large number of features in our dataset further supported the motivation to choose SGD over Newton-Raphson.

- L2 (Gauss) Regularization was applied to prevent overfitting. From a practical point of view, L2 tends to shrink coefficients evenly and is useful when we have collinear/co-dependent features. Co-dependency tends to increase coefficient variance, making coefficients unreliable/unstable, which hurts model generality. L2 reduces the variance of these estimates, which counteracts the effect of co - dependencies. This is crucial since our dataset includes co-dependent variables, specifically SBA_Appv & Gross_Appv, as the SBA backed loans depend on the Gross loans.

Logistic Regression - Model Performance

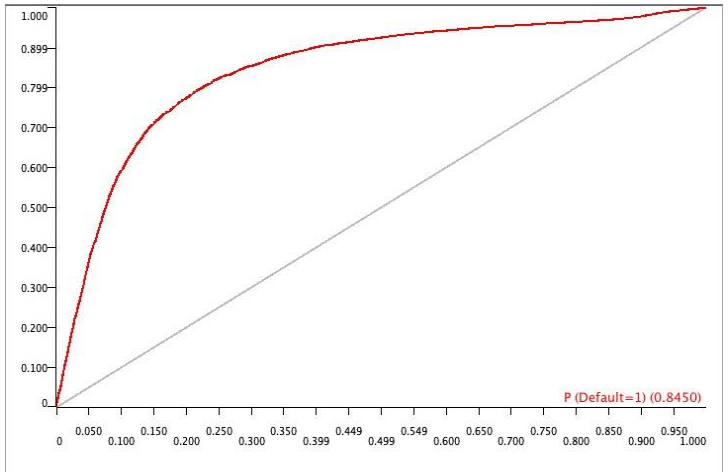
Default \ Prediction (Default)	0	1
0	121894	7300
1	24044	15919

Correct classified: 137,813 | Wrong classified: 31,344

Accuracy: 81.47 % | Error: 18.53 %

Cohen's kappa (κ) 0.4

The first variation of the model yielded interesting results. Specifically, the model yielded an accuracy of 81.47%, wrongly classifying 31,344 observations against 137,813 that were correctly classified.

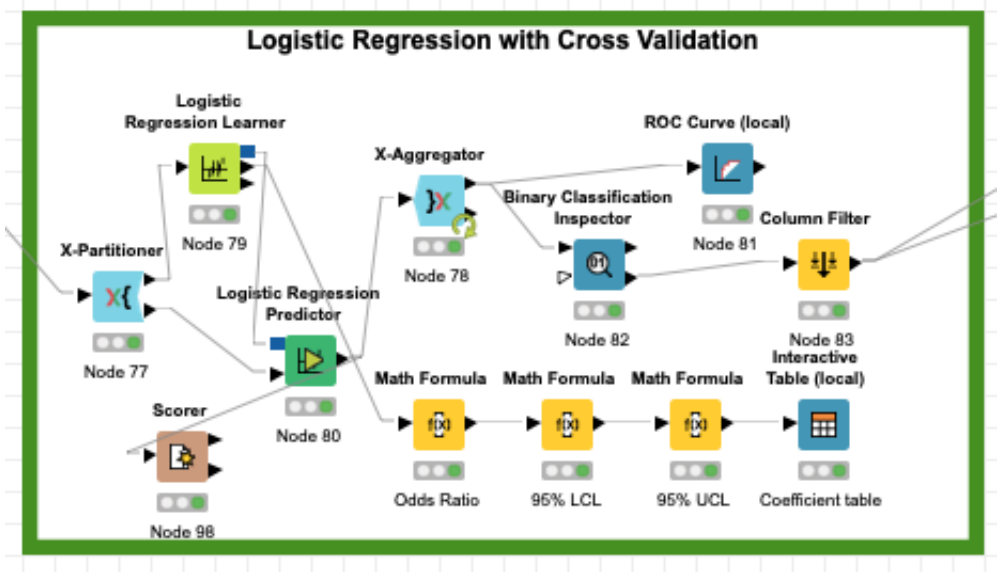


The ROC Chart implies an Area Under the Curve of 0.8450, which considering the complexity of the dataset, is not a bad value at all and provides a solid benchmark for comparison against other models.

S Model Name	D Recall	D Precision	D Sensitivity	D Specificity	D F-Measure	D AUC
P (Default=1)	0.714	0.592	0.714	0.848	0.647	0.845

The model recorded a recall score of 0.714, precision of 0.592, specificity of 0.848 and an F-Measure of 0.647. The low precision would be an issue of this model was to be implemented as it would incorrectly predict more defaults than would occur.

Logistic Regression with Cross-Validation



- The second variation of the model uses the exact same specifications as the first variation (L2 regularization & SGD as the iterative optimization method) except that we now cross-validated the model using the "X-Partitioner" node as shown above. We decided to cross-validate in order to further prevent overfitting and prevent biased results since the model is also trained on a validation set.

    - Specifically, the X-Partitioner node was configured such that the number of validations would equal 5, instead of the default value of 10, for computational efficiency reasons.

Logistic Regression with Cross-Validation - Model Performance

File Hilite

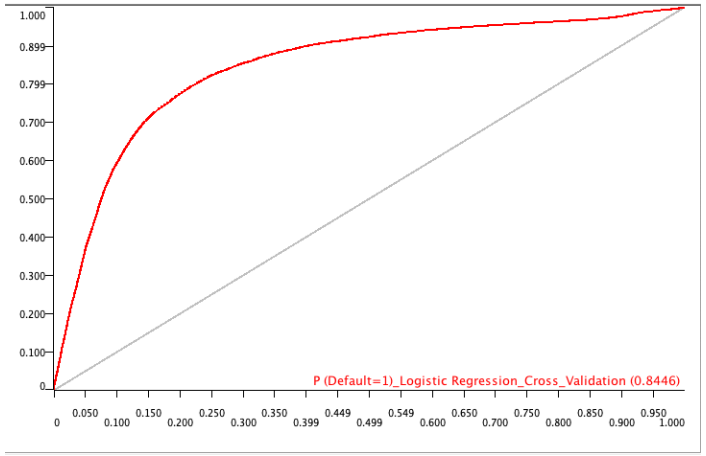
Default \ P...	0	1
0	82289	5173
1	14875	10434

Correct classified: 92,723 Wrong classified: 20,048

Accuracy: 82.222 % Error: 17.778 %

Cohen's kappa (κ) 0.409

The second variation of the model yielded an accuracy of 82.222%, wrongly classifying 20,048 observations against 130,931 that were correctly classified. This is slightly higher than the first LR model.



The ROC Chart implies an Area Under the Curve of 0.8446, which is slightly below the previous model.

S Model Name	D Recall	D Precision	D Sensitivity	D Specificity	D F-Measure	D AUC
P (Default=1)_Logistic Regression	0.707	0.598	0.707	0.853	0.648	0.845

This model recorded a recall of 0.704, precision of 0.599, specificity of 0.853 and an F-Measure of 0.648.

Logistic Regression Comparison

	D Recall	D Precision	D Sensitivity	D Specificity	D F-Measure	D AUC
Logistic Regression	0.714	0.592	0.714	0.848	0.647	0.845
Logistic Regression + CV	0.707	0.598	0.707	0.853	0.648	0.845

We noticed that both variations of the logistic regression model yield very similar results, with the first Logistic Regression yielding a higher recall of 0.714 and the same AUC of 0.845. The cross-validated variation yielded a higher precision, specificity and F-Measure, albeit by a very small margin.

Since the F-Measure is our main key metric for model performance, we will proceed with the second variation of the Logistic Regression model since it has a higher F-Measure. However, it is important to note that such small margins are rarely significant in a business context. In fact, we could continue with the first variation and use it as our candidate model without any major repercussions.
However, for the sake of consistency, we will stick to the cross-validated Logistic Regression as our candidate Logistic Regression model.

Model 2 – Multilayer Perceptron

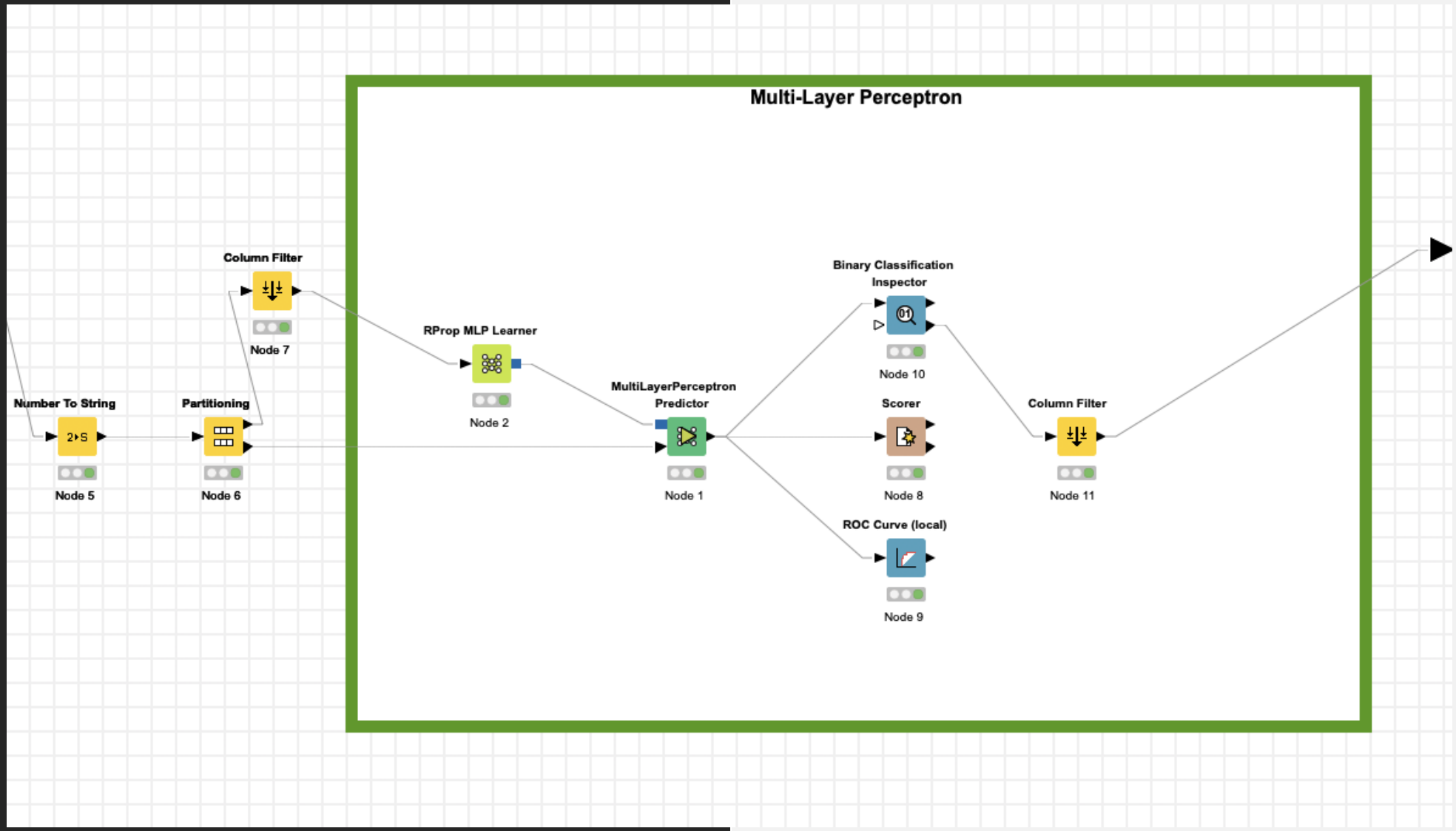
Multi-Layer Perceptron – Model Overview

A Multi-Layer Perceptron (MLP) is a class of feedforwards artificial neural networks (ANN). ANN are often known to yield better performances than more traditional models such as Logistic Regression, especially on high dimensional data.

They offer several advantages such as the ability to implicitly detect complex nonlinear relationships between dependent and independent variables, ability to detect all possible interactions between predictor variables, and the availability of multiple training algorithms.

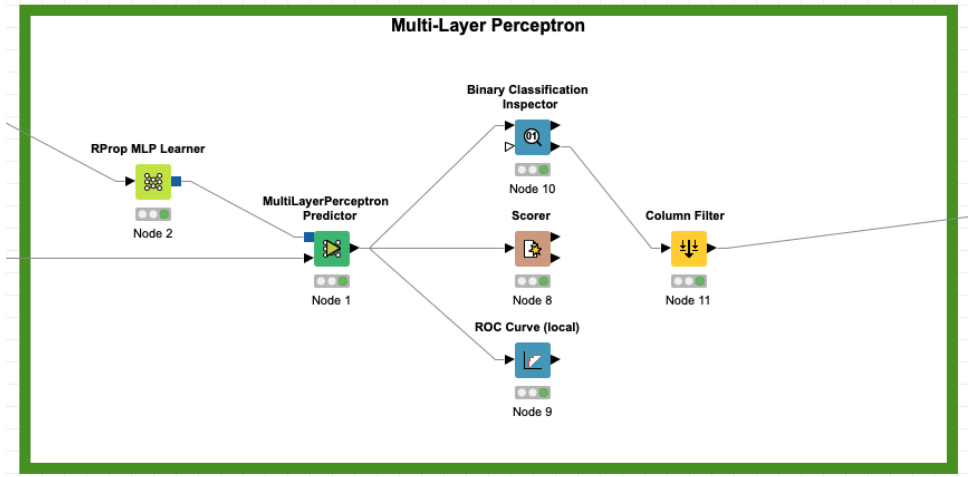
However, the black box nature of these models can often lead to interpretability issues along with a high overfitting risk and computationally intensive training time. These implications can often dissuade managers to implement this model.

Nevertheless, due to the high dimensional nature of our dataset and its complexity, we decided to include this model in our overall analysis.



Model 2 – Multi-Layer Perceptron workflow

Multi-Layer Perceptron



Like before, the dataset was partitioned into a 70/30 train/test split. However, a key issue we encountered with the model is that the "RProp MLP Learner" node expects numeric (referred to as "Double columns" on KNIME) and needs in addition one non-numeric (string) column as a target variable ("Default" in our case) in the input dataset. Since the variables "State", "LowDoc" and "Industry" are all string columns, they would all have to be converted to dummy variables which would result in over 80 features in total. Therefore, to manage computational cost, we decided to drop those three features. We must keep this in mind when we evaluate the results from this model.

We set the maximum number of iterations to 100 with 2 hidden layers and 10 hidden neurons per layer. These values were carefully selected to balance model performance with computational efficiency. As mentioned in the introductory slide, ANN's are known for excessive computational times which is why we stuck to only using 2 hidden layers.

Multi-Layer Perceptron - Model Performance

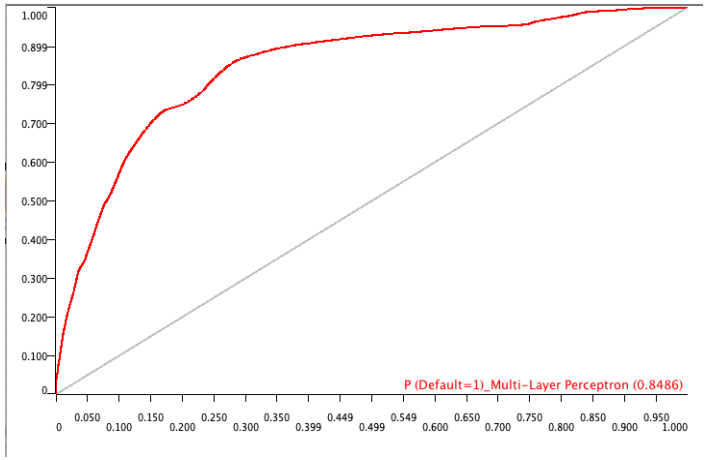
Default \ Prediction (Default)	0	1
0	118025	11169
1	19176	20787

Correct classified: 138,812 Wrong classified: 30,345

Accuracy: 82.061 % Error: 17.939 %

Cohen's kappa (κ) 0.466

The MLP model yielded an accuracy of 82.061%, which is slightly higher than the one yielded by the candidate Logistic Regression model.



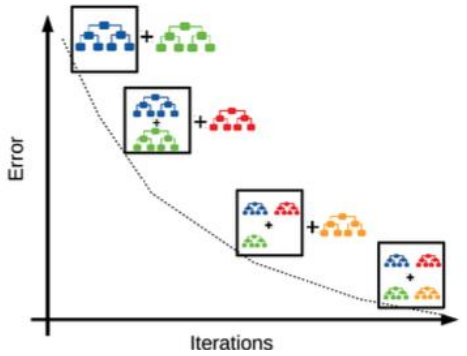
The ROC Chart implies an Area Under the Curve of 0.849, which is again slightly higher than the candidate Logistic Regression model.

S Model Name	D Recall	D Precision	D Sensitivity	D Specificity	D F-Measure	D AUC
P (Default=1)_Multi-Layer Perceptron	0.72	0.58	0.72	0.839	0.642	0.849

The F-Measure is slightly lower than the one yielded by the candidate logistic regression model.
A possible explanation can be attributed to the exclusion of the features "State", "LowDoc" and "Industry", which may have had an adverse impact on model performance.

Model 3 – Gradient Boosting

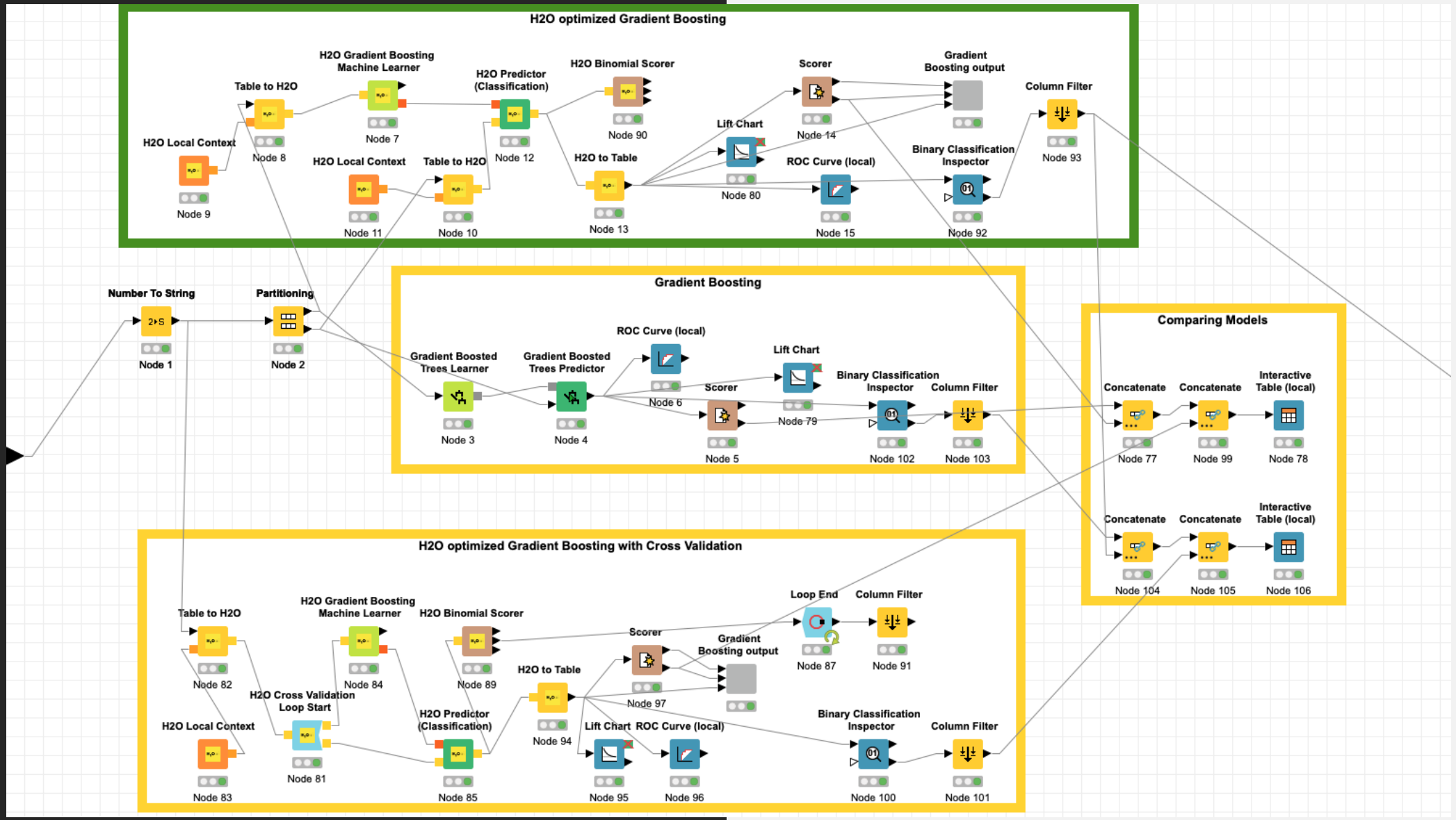
Gradient Boosting - Model Overview



- **Gradient Boosted Trees** are ensemble tree models, based on the boosting technique, combining multiple sequential simple decision trees into a stronger model: the final model is "improved" focusing its complexity growth on the correction of errors obtained phase by phase.

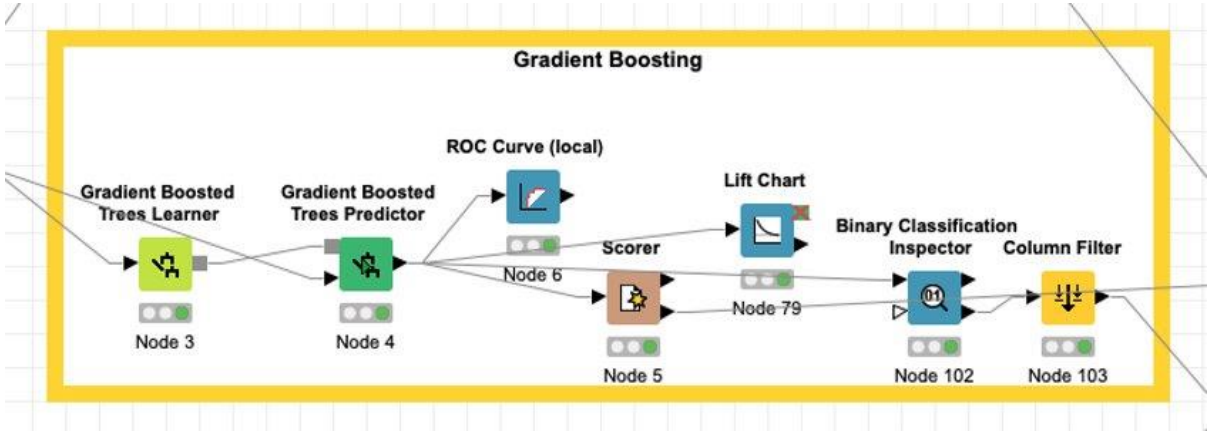
- The learners are **trained sequentially** each trying to correct its predecessor

- All the weak learners with their higher accuracy of error are combined in some way to get a strong classifier, with a higher accuracy.



Model 3 – Gradient Boosting workflow

Gradient Boosting



After partitioning the data into 70% training and 30% testing sets, we decided to first implement a regular Gradient Boosting algorithm without any optimization:

- All variables were included in the learner.
- The tree depth was limited at 4 in order not to unnecessarily increase the computational cost of the algorithm.
- The learning rate was set to 0.1.

Gradient Boosting - Model Performance

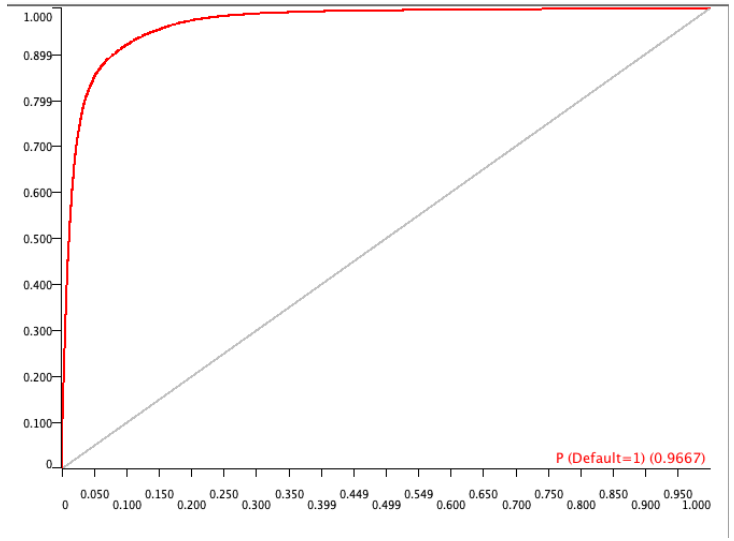
Default \ Prediction (Default)	0	1
0	123776	5593
1	6937	32851

Correct classified: 156,627 Wrong classified: 12,530

Accuracy: 92.593 % Error: 7.407 %

Cohen's kappa (κ) 0.792

The (non-optimized) model yielded interesting results, achieving an error rate of 7.407%, therefore wrongly classifying 12,530 defaults against 156,627 that were correctly classified



The ROC Chart shows significant improvements in performance compared to the Logistic Regression and Multi-Layer Perceptron model, with an Area Under the Curve of 0.967

S Model Name	D Recall	D Precision	D Sensitivity	D Specificity	D F-Measure	D AUC
P (Default=1)	0.857	0.832	0.857	0.947	0.844	0.967

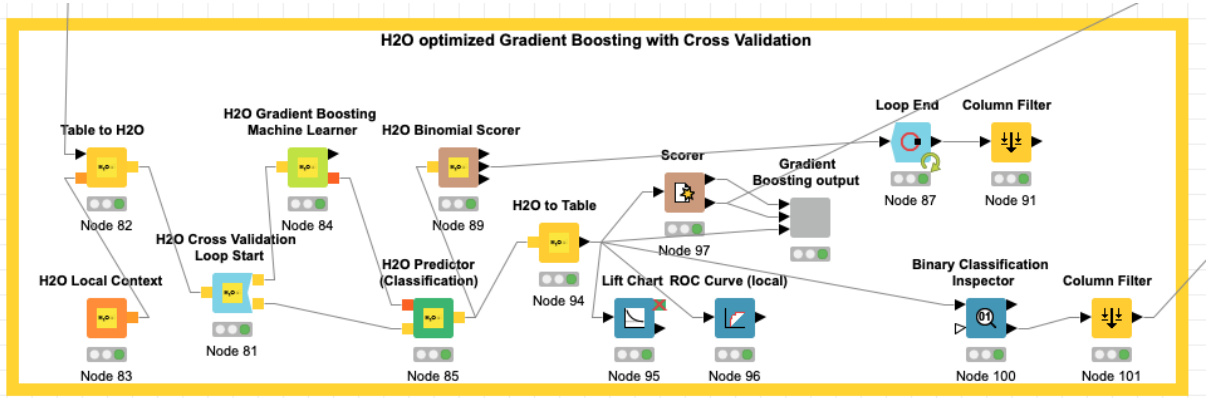
Finally, also the F-Measure confirms the increase in model performance suggested by both Accuracy and AUC values

We then decided to try further increase performance by using the H2O optimized library on KNIME for Machine Learning.

The integration of Driverless AI offers additional option to automate machine learning out of the box with a huge range of powerful algorithms. And therefore, we decided to run H2O Gradient Boosting Machine Learners both with and without Cross-Validation

Cross-Validation allowed us to potentially flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset

H2O optimized Gradient Boosting with Cross Validation



- We decided to use a (H2O) 5-fold Cross Validation loop with random fold assignment.
- We also increased the tree depth to 10 for comparison purposes.
- Other algorithm settings were not changed from default as we were able to reach better overall performance taking into consideration also the computational cost of the algorithm.

H2O optimized Gradient Boosting with Cross Validation – Model Performance

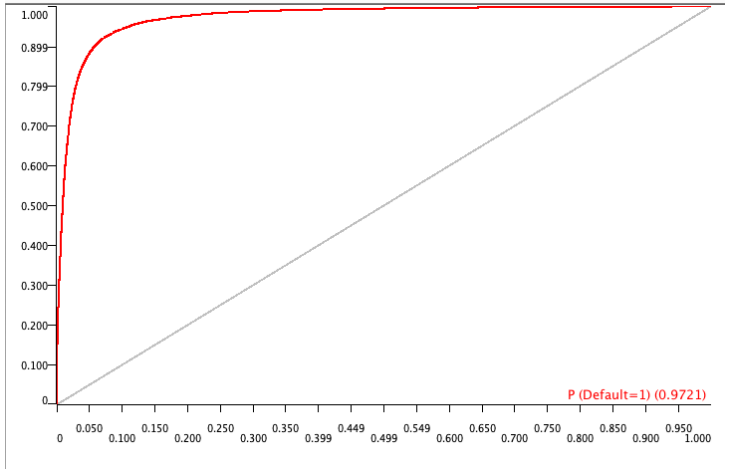
Default \ P ...	1	0
1	22780	3966
0	3450	82119

Correct classified: 104,899 Wrong classified: 7,416

Accuracy: 93.397 % Error: 6.603 %

Cohen's kappa (κ) 0.817

The H2O optimized Gradient Boosting algorithm with Cross-Validation increased performance with respect to the previous model: misclassification rate dropped to 6.603% (7,416 wrong classifications against 12,530).

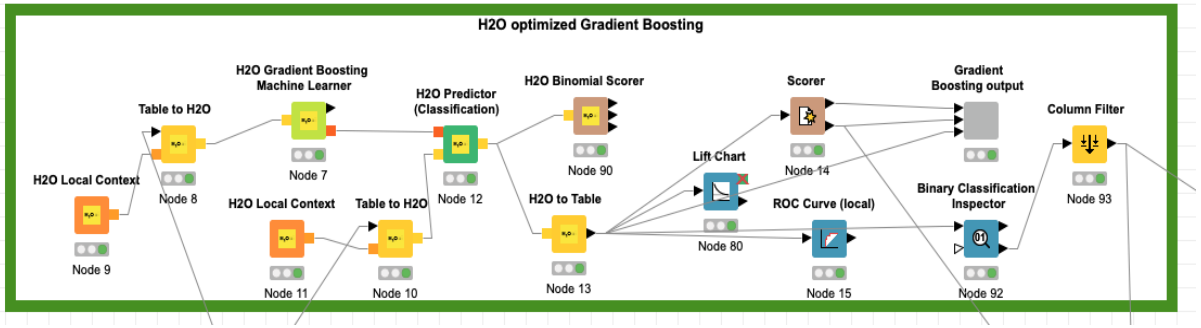


As expected, also the ROC curve is better and as a result the AUC value increased.

S Model Name	D Recall	D Precision	D Sensitivity	D Specificity	D F-Measure	D AUC
P (Default=1)	0.885	0.844	0.885	0.949	0.864	0.972

The F-Measure confirms the result that H2O optimized Gradient Boosting with Cross Validation performs better than regular GBM.

H2O optimized Gradient Boosting



Since the Cross-Validation is a computationally expensive procedure, we wanted to test whether an H2O optimized Gradient Boosting model could generate comparable results using the 70/30 partitioning mentioned previously instead of the CV loop.

Algorithm settings are the same used in the Cross-Validated model we just discussed.

H2O optimized Gradient Boosting – Model Performance

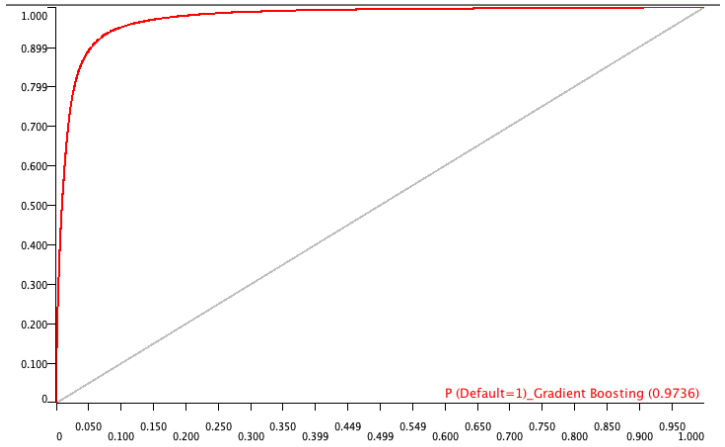
Default \ P ...	1	0
1	34330	5458
0	5293	124076

Correct classified: 158,406 Wrong classified: 10,751

Accuracy: 93.644 % Error: 6.356 %

Cohen's kappa (κ) 0.823

We observed a higher number of wrongly classified instances compared to the previous model, that is because of the kind of partitioning considered: in fact, CV provided a smaller test set (< 30%) and as a result, in this model we have higher accuracy and lower error rate even if we see that more total wrong classifications occurred.



AUC value and the ROC curve confirm our expectation, providing the best results experienced so far.

S Model Name	D Recall	D Precision	D Sensitivity	D Specificity	D F-Measure	D AUC
P (Default=1)_Gradient Boosting	0.885	0.849	0.885	0.952	0.867	0.974

The same happens when we observe the F-Measure, which compared to the Cross-Validated GB model slightly increased as well.

Gradient Boosting Comparison

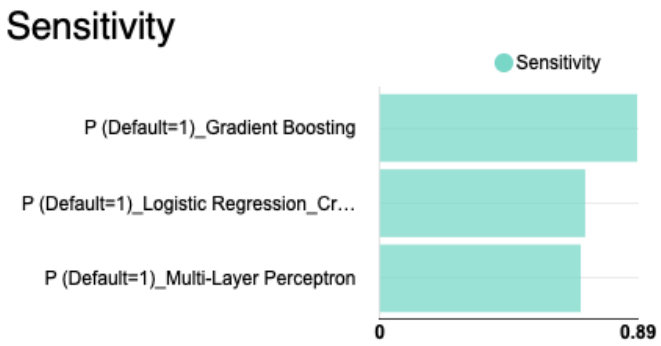
Row ID	I TruePositives	I FalsePositives	I TrueNegatives	I FalseNegatives	D Recall	D Precision	D Sensitivity	D Specificity	D F-measure	D Accuracy	D Cohen's kappa
0	123776	6937	32851	5593	0.957	0.947	0.957	0.826	0.952	?	?
1	32851	5593	123776	6937	0.826	0.855	0.826	0.957	0.84	?	?
Overall	?	?	?	?	?	?	?	?	?	0.926	0.792
0_H2O	123349	4786	35002	6020	0.953	0.963	0.953	0.88	0.958	?	?
1_H2O	35002	6020	123349	4786	0.88	0.853	0.88	0.953	0.866	?	?
Overall_H2O	?	?	?	?	?	?	?	?	?	0.936	0.824
0_H2O_CV	81633	3491	23255	3936	0.954	0.959	0.954	0.869	0.956	?	?
1_H2O_CV	23255	3936	81633	3491	0.869	0.855	0.869	0.954	0.862	?	?
Overall_H2O_CV	?	?	?	?	?	?	?	?	?	0.934	0.819

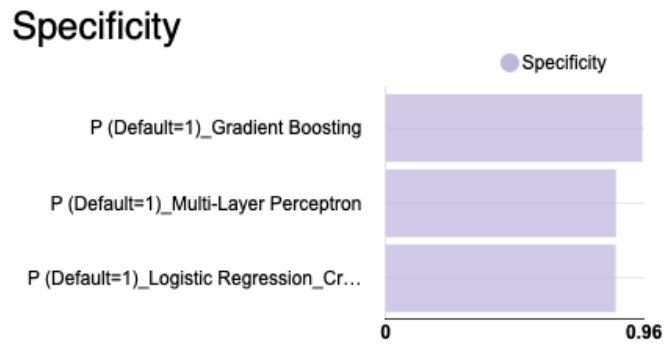
	D Recall	D Precision	D Sensitivity	D Specificity	D F-Measure	D AUC
Gradient Boosting	0.857	0.832	0.857	0.947	0.844	0.967
H2O Gradient Boosting	0.885	0.849	0.885	0.952	0.867	0.974
H2O Gradient Bosting + CV	0.885	0.844	0.885	0.949	0.864	0.972

By looking at overall models performance comparison, we finally concluded that, since we are able to obtain higher Accuracy, F-Measure and AUC score, and at the same time being able to cut the additional computational costs of the Cross-Validation procedure, our best Gradient Boosting model is the one with 70% train and 30% partitioning and H2O implementation.

Model Comparison

Model Name	Recall	Precision	Sensitivity	Specificity	F-Measure	AUC
P (Default=1)_Gradient Boosting	0.885	0.849	0.885	0.952	0.867	0.974
P (Default=1)_Multi-Layer Perceptron	0.691	0.593	0.691	0.854	0.638	0.851
P (Default=1)_Logistic Regression_Cross_Validation	0.706	0.599	0.706	0.853	0.648	0.845





Model Comparison

Model Name	Recall	Precision	Sensitivity	Specificity	F-Measure	AUC
P (Default=1)_Gradient Boosting	0.885	0.849	0.885	0.952	0.867	0.974
P (Default=1)_Multi-Layer Perceptron	0.691	0.593	0.691	0.854	0.638	0.851
P (Default=1)_Logistic Regression_Cross_Validation	0.706	0.599	0.706	0.853	0.648	0.845

As we can see from the chart above, Gradient Boosting outperforms all other models considered in the analysis in every metric of interest, while still being comparable to the others in terms of running time and computational complexity.

Therefore, the model of choice for our analysis is the
H2O optimized Gradient Bosting Machine algorithm

Managerial Implications and Conclusion

Through this project we have developed an *(algorithmic)* model that can predict if a loan will default or not and thus, we can answer the research question of whether a specific loan should be approved based on a firms' characteristics.

In addition, the results previously discussed show that, when classifying loans as defaulting or non-defaulting, especially with the primary choice of model selected - H20 optimised gradient boosting - a high accuracy, low error rate and more importantly, a high F-Score are achieved.

Therefore, we believe that the study conducted can be a significant contribution from a managerial perspective. First and foremost, while being able to more efficiently predict if a loan will default or not, banks can reduce the potential losses of issuing loans that most likely will default. By multiplying the estimated loss per defaulted loan by the total number of missed defaults (false negatives) produced by each model's confusion matrix, bank officers can also compare the overall losses with different models and choose the one that minimises the overall losses. In this regard, we posit that the H20 optimised gradient descent model would result in the lowest overall expected losses.

Limiting the false positives (loans that wouldn't default but predicted as default) is another important consideration for a bank officer as they would be foregoing potential interest profit on the loan by rejecting the incorrectly classified loan. This is an additional reason supporting the H20 optimised gradient boosting model which had the highest precision and F-score. Reducing false negatives (high recall) minimizes potential losses and reducing false positives (high precision) maximizes profits. Therefore, the model with the best F-score will provide the best performance for bank managers.

Second, combining the use of the model with their domain knowledge, loan officers at a bank are better supported in their decisions. They can better identify the features that indicate a potential default, significantly aiding due-diligence investigation. When unsure whether a small busines will default on a loan, bank managers could rely on the results of the model to support their decision making. Apart from the model, managers can also derive key insights from some of the additional variables that were generated for the purpose of this analysis. Examples of some of the insights derived from these additional variables are described in detail in the following slide.

Managerial Implications and Conclusion

Managers can utilize some of the additional generated variables to draw further insights. For example, they can explore whether Real Estate has any tangible impact when determining whether to approve loans or not that are backed by real estate. For instance, according to the H20 Optimized gradient boosting model, the average probability of a default on a real-estate backed loan is 2.1%, which is significantly lower than the average probability of a default on a loan not backed by real-estate (26.6%). Therefore, a real-estate backed loan can be seen as a positive indicator for loan officers when considering whether or not to grant the loan. Furthermore, managers can also explore industries which are more likely to be susceptible to default. For example, according to our model, a company in the real estate and rental and lending industry has an average probability of default of 29.7%. On the other hand, a company in the health care and social services industry has an average probability of a default that is much lower at 15.6%. Using these additional variables, managers can gain better insights on the probability of a loan defaulting and use these additional perspectives when deciding whether to grant a loan.

RealEstateB	Average Probability (Default = 1)
1	0.021
0	0.266

Industry	Average Probability (Default = 1)
Real Estate and Rental and Lending	0.297
Health Care and Social Services	0.156