Loan Default Prediction

Aysan Jarvand

Problem Definition

CONTEXT

- Retail banks offer home loans to obtain profits.
  - Loans are borrowed by bank customers.
  - Banks are rigorous while approving loans.

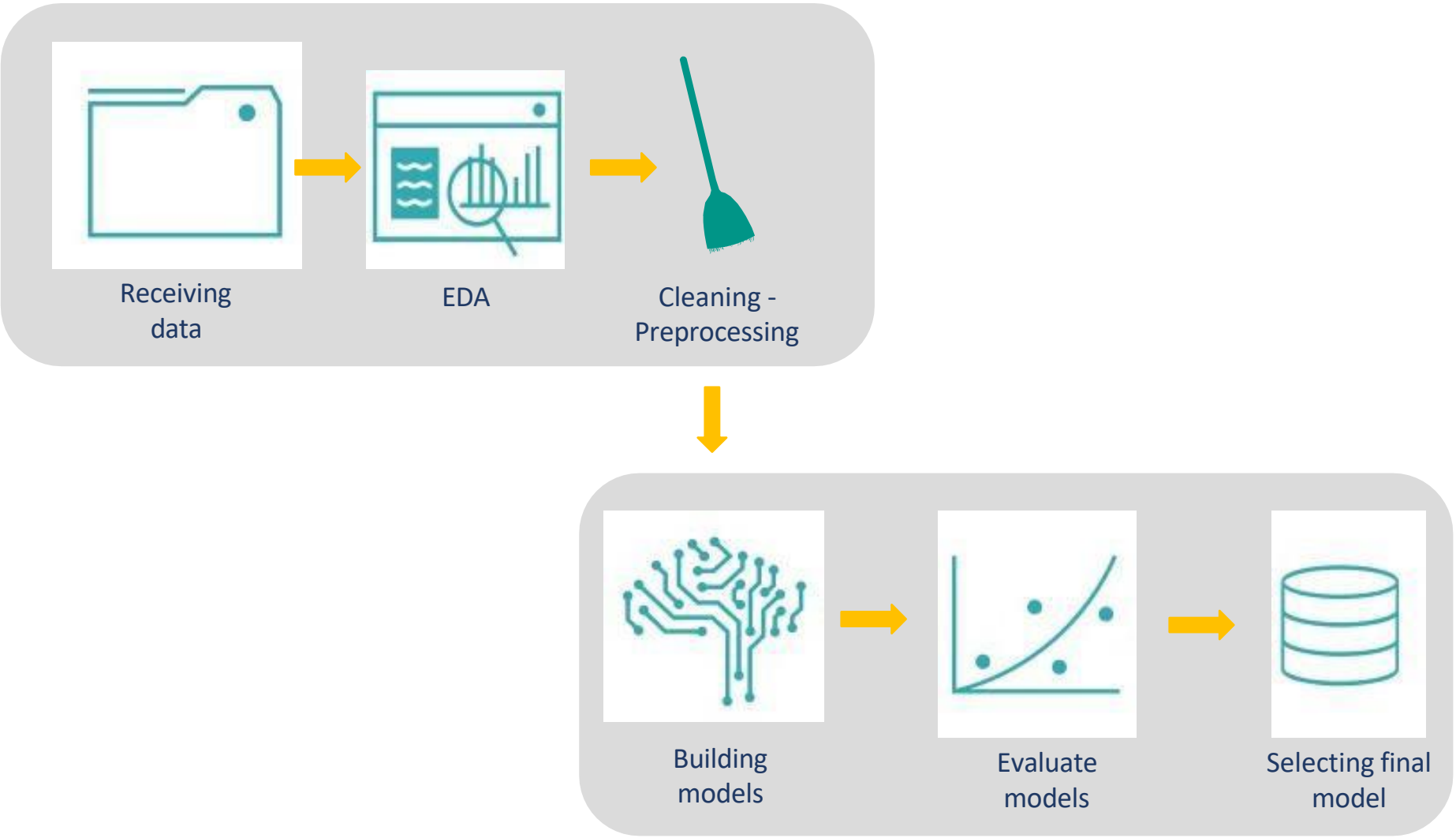
PROBLEM

- Banks need an effective approval process.
- This process is effort-intensive and sensitive to human error and biases.

OBJECTIVES

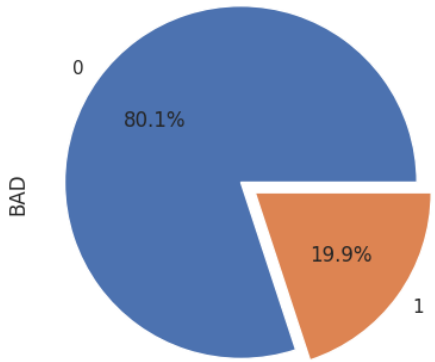
- Predict clients who are likely to default on their loan
- Avoid the risk of misclassification of default loans predicted as non-default loans.

Solution approach

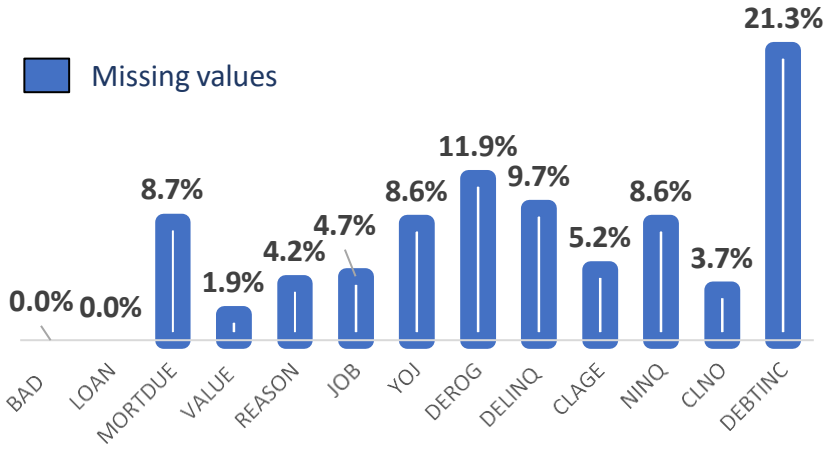


* Images taken from databricks.com

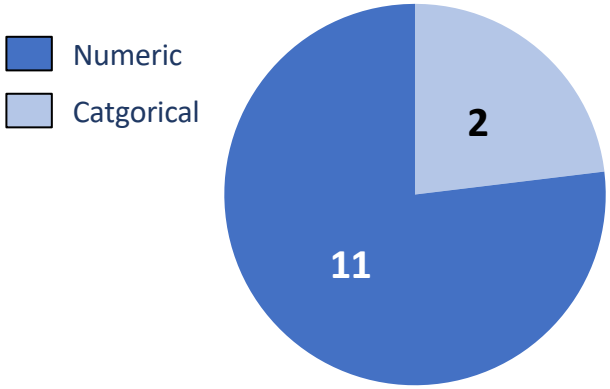
Data Insights



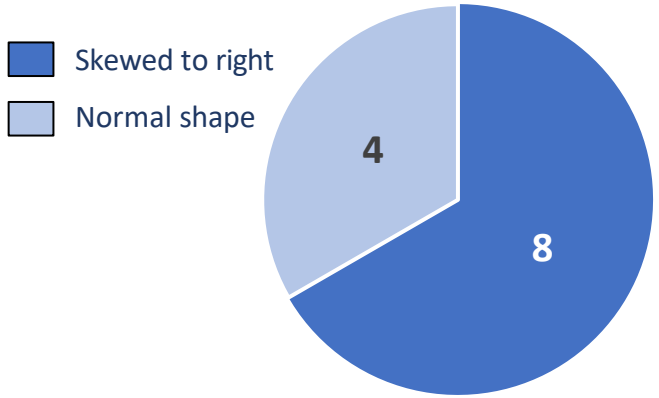
- Data contains 5,960 registers and 13 features.
- Data is unbalanced in a 80%-20% proportion.
- A balancing process of the data is needed to modeling.



- Except Loan and BAD all features have missing values.
- A filling missing value is needed using median and mode.



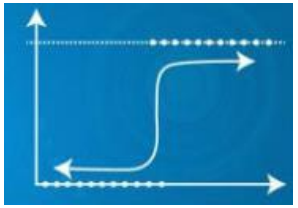
- Data types are distributed in 11 numeric features and 2 categorical.
- Categorical data needs to be treated in order to use it.



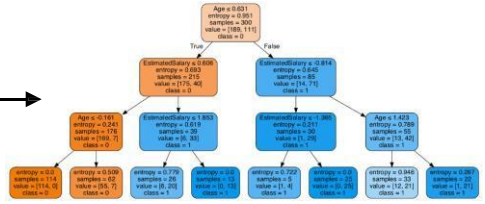
- 8 features are skewed to the right and 4 are normal shaped.
- All features have a big amount of outliers.
- Outliers need to be treated for Logistic Regression modeling

Proposed model solutions

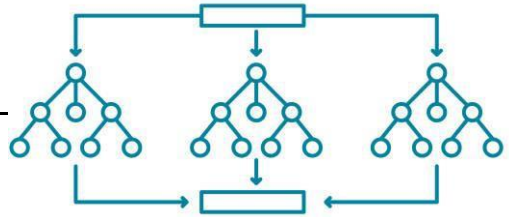
Logistic regression



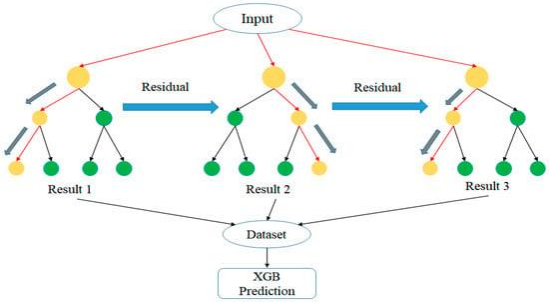
Decision Trees



Random Forest



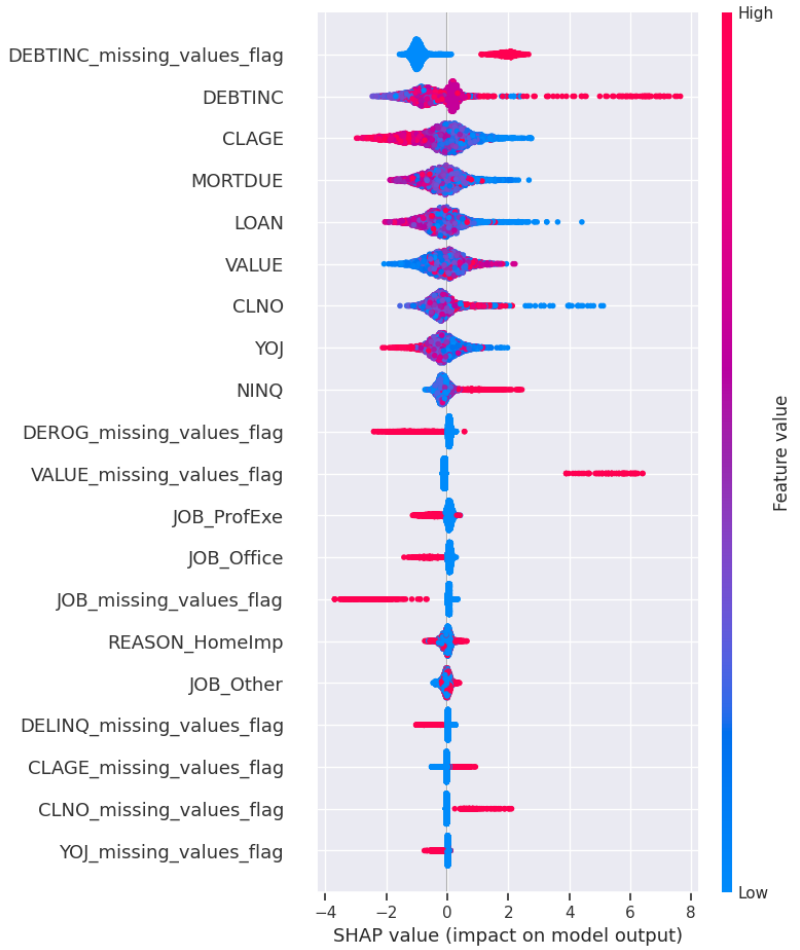
XGBoost Classifier



Comparison of techniques and performances

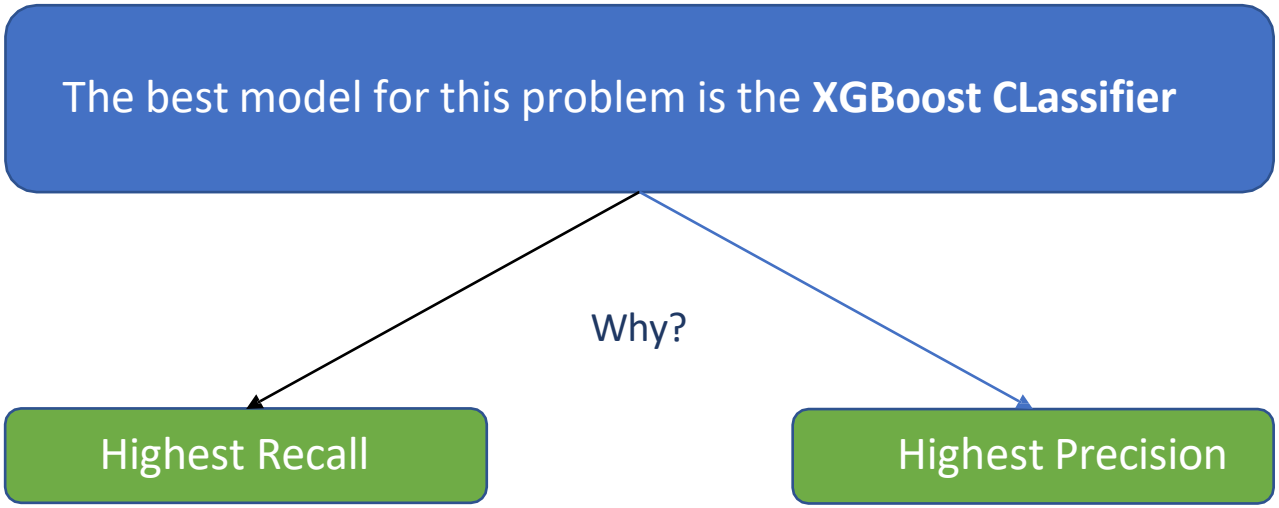
Models	Accuracy	Recall	Precision	Pros	Cons
XGBoost Classifier	.91	.82	.87	-High precision, recall & accuracy	-Non interpretable
Tuned Random Forest	0.88	0.78	0.83	-Higher precision than Decision Trees	-Non Interpretable
Random Forest	0.90	0.82	0.85	-Higher precision than Decision Trees	-Non interpretable
Tuned Decision Tree	0.86	0.76	0.78	-Highest recall -Interpretable	-Lower precision than Random Forest
Decision Tree	0.86	0.76	0.78	-Similar recall than Random Forests. -Interpretable	-Lower precision than Random Forest
Logistic Regression	.81	.54	.75	-Higher precision than Decision Trees -Interpretable	-Very low recall

Model Interpretability check with Shap values



•High DEBTINC values are on the right side, contribute positively to identify defaulter

Proposal for the Final Solution Design



- The XGBoost classifier is giving highest 0.82 recall , 0.91 accuracy and a precision of .87

Executive summary

- A XGBoost Classifier model can predict loan defaulters 82% of the time they come to ask for a home loan.
- This model has highest 82% recall for class 1 on the test data, 87% precision score, and 91% accuracy.
- The most important features to make a proper prediction are DEBINC, DELINQ and CLAGE.
- The Debt-to-Income ratio is the most important feature but also the one with the most missing data (21.3%) which is similar to the proportion of defaulted customers (20%).

Recommendations and next steps

- Check the possibility to create an alternative business process to manage and take decisions on those clients with no Debt/Income ratio available.
- Explore other machine learning techniques such as engineering features, dropping columns, different method for handling missing value with new algorithms like other Bosting algorithms, Support Vector Machine, KNN, neural networks.
- Create a pilot test with the new model and compare the results with the current manual process before completing the transition to the new model.
- Consider using balancing techniques, such as ROSE (Random Over Sampling Examples) and SMOTE (Synthetic Minority Oversampling Technique) or under sampling methods, to generate synthetic data. These methods have achieved good results on unbalanced datasets.

Risks and challenges

- The major risk associated with developing a product that meets the needs of both consumer and business.
- A big challenge is to increase the current incomes with the new model.
- There will be a emotional challenge to launch new technology while supporting old ones.
- Cost will be related to train the user with new system.

Thank You

Appendix

Why is recall important to this project?

The model can make two types of wrong predictions:

1. Predicting a client will pay his loan when the client can't pay.
2. Predicting a client won't pay his loan when the client can pay.

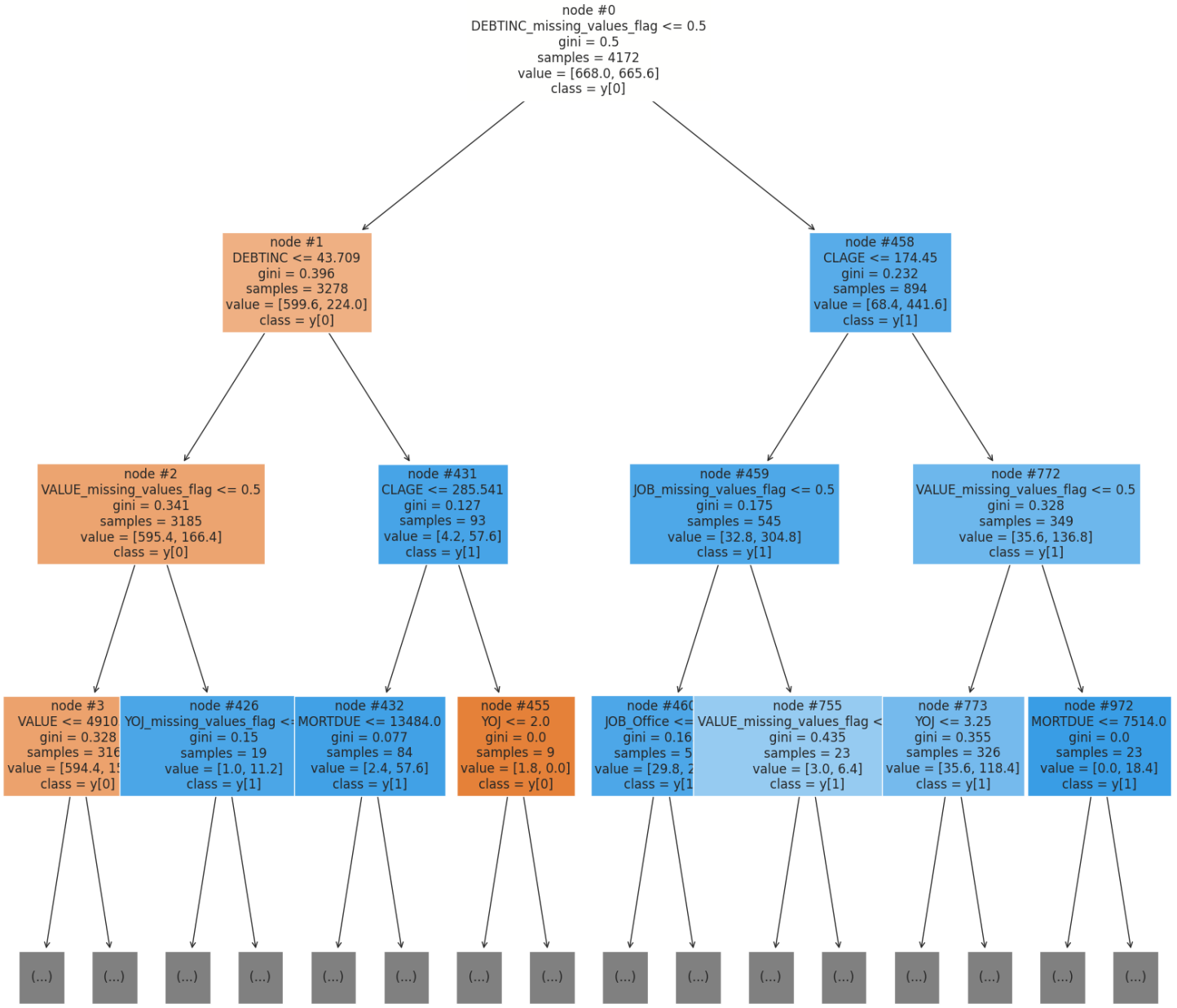
* high recall + high precision : the class is perfectly handled by the model
* low recall + high precision : the model can't detect the class well but is highly trustable when it does
* high recall + low precision : the class is well detected but the model also include points of other classes in it
* low recall + low precision : the class is poorly handled by the model

Which case is more important?

- **Predicting that a client will pay but the client can't**, i.e., losing money immediately. This would be considered the most important case of wrong predictions because bad loans (NPA) usually eat up a major chunk of the bank profits.

How to reduce this loss i.e the need to reduce False Negatives?

- **The bank would want the RECALL to be maximized**, the greater the Recall, the higher the chances of minimizing false negatives. Hence, the focus should be on increasing the Recall (minimizing the false negatives) or, in other words, identifying the true positives (i.e. Class 1) This would help in increasing the bank profit that comes from interests in the form of home loans.


node #0
DEBTINC_missing_values_flag <= 0.5
gini = 0.5
samples = 4172
value = [668.0, 665.6]
class = y[0]
node #1
DEBTINC <= 43.709
gini = 0.396
samples = 3278
value = [599.6, 224.0]
class = y[0]
node #458
CLAGE <= 174.45
gini = 0.232
samples = 894
value = [68.4, 441.6]
class = y[1]
node #2
VALUE_missing_values_flag <= 0.5
gini = 0.341
samples = 3185
value = [595.4, 166.4]
class = y[0]
node #431
CLAGE <= 285.541
gini = 0.127
samples = 93
value = [4.2, 57.6]
class = y[1]
node #459
JOB_missing_values_flag <= 0.5
gini = 0.175
samples = 545
value = [32.8, 304.8]
class = y[1]
node #772
VALUE_missing_values_flag <= 0.5
gini = 0.328
samples = 349
value = [35.6, 136.8]
class = y[1]
node #3
VALUE <= 4910
gini = 0.328
samples = 316
value = [594.4, 1
class = y[0]
node #426
YOJ_missing_values_flag <=
gini = 0.15
samples = 19
value = [1.0, 11.2]
class = y[1]
node #432
MORTDUE <= 13484.0
gini = 0.077
samples = 84
value = [2.4, 57.6]
class = y[1]
node #455
YOJ <= 2.0
gini = 0.0
samples = 9
value = [1.8, 0.0]
class = y[0]
node #460
JOB_Office <=
gini = 0.16
samples = 5
value = [29.8, 2
class = y[1]
node #755
VALUE_missing_values_flag <
gini = 0.435
samples = 23
value = [3.0, 6.4]
class = y[1]
node #773
YOJ <= 3.25
gini = 0.355
samples = 326
value = [35.6, 118.4]
class = y[1]
node #972
MORTDUE <= 7514.0
gini = 0.0
samples = 23
value = [0.0, 18.4]
class = y[1]
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)