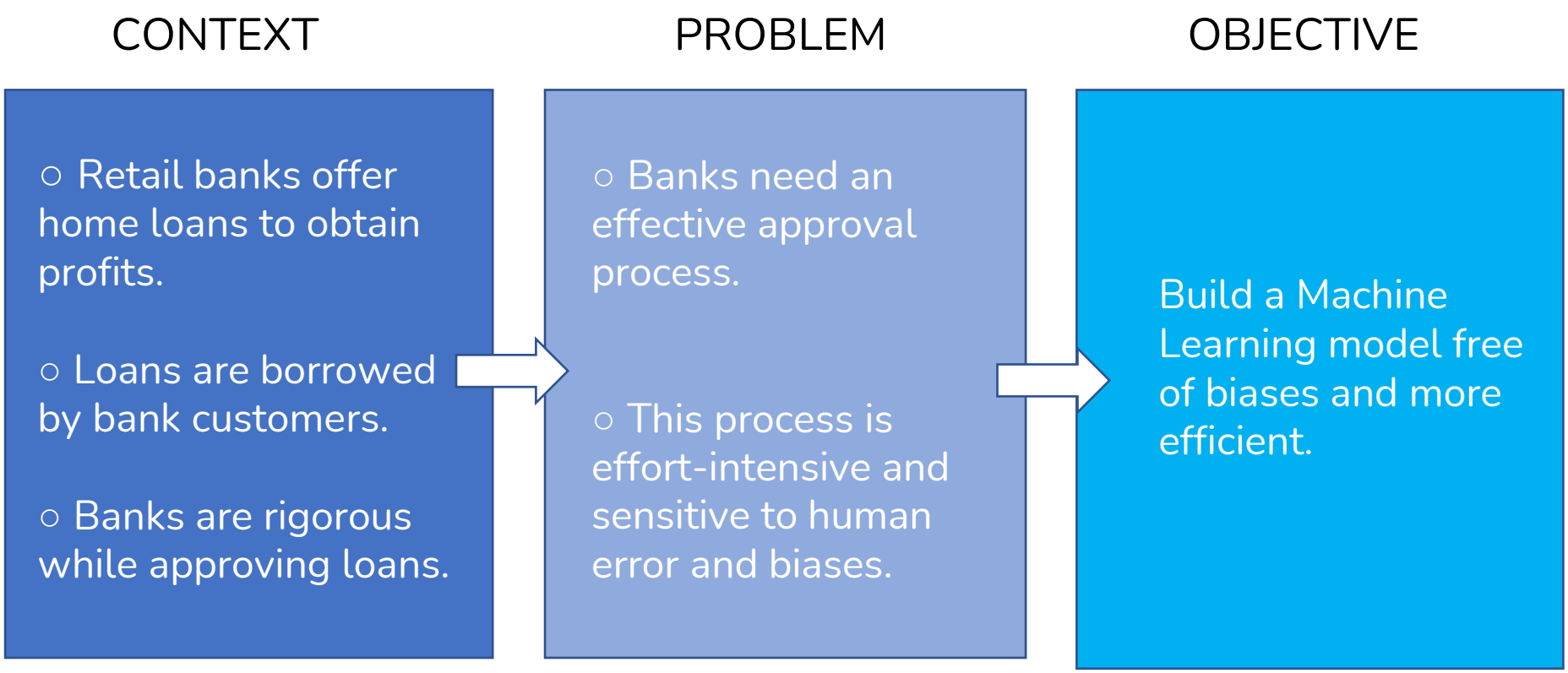
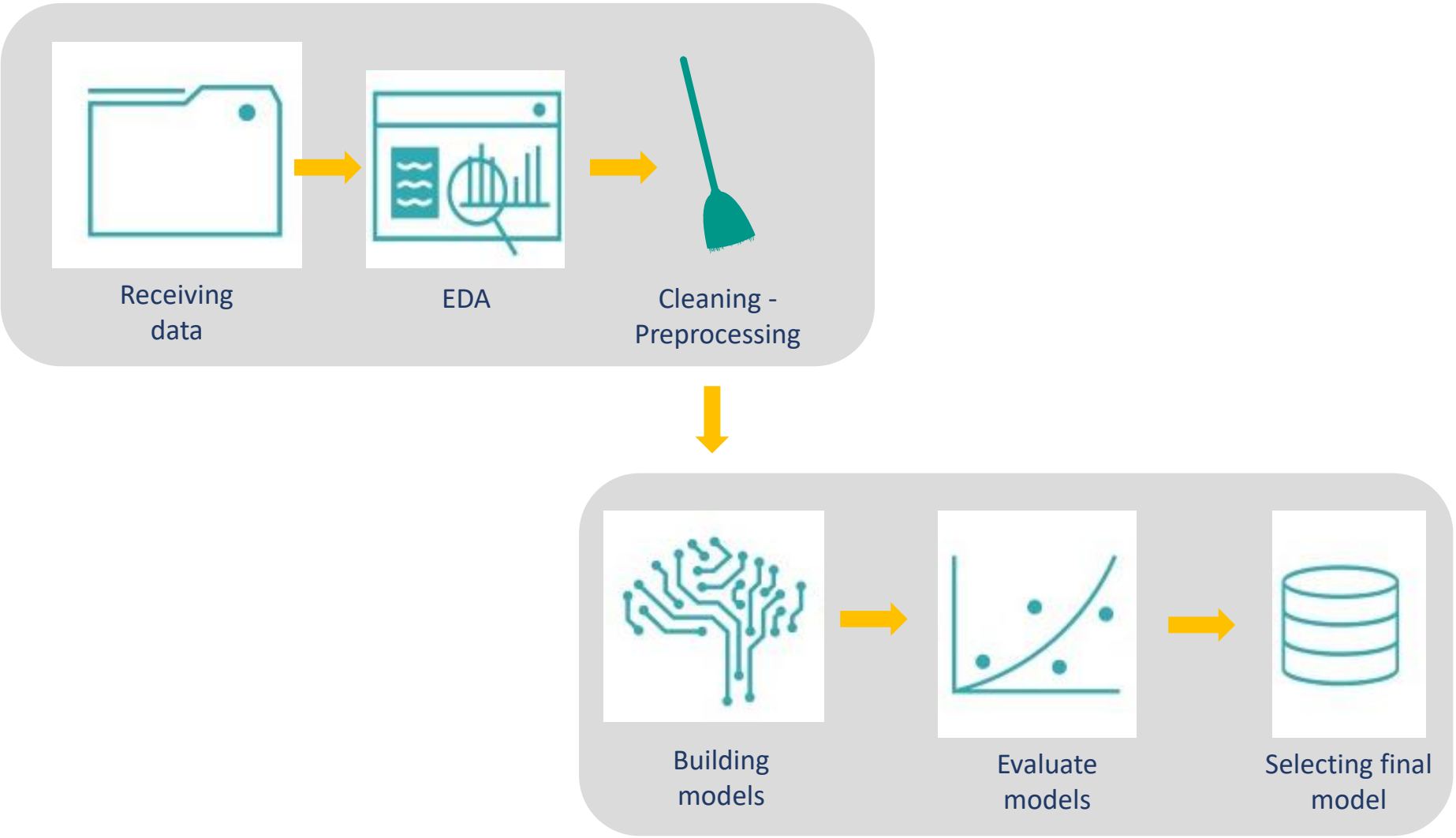
Loan default prediction using Machine Learning techniques

Daniel Jordan
04/08/2022

Problem definition

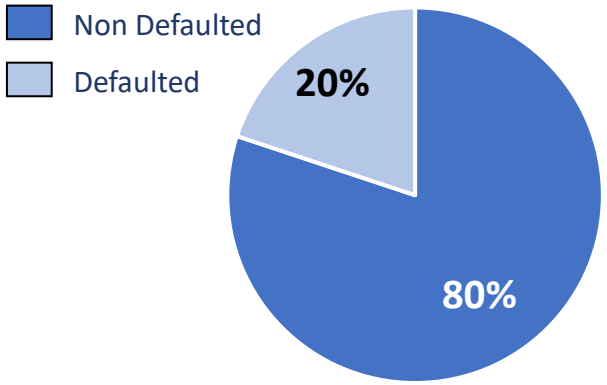


Solution approach

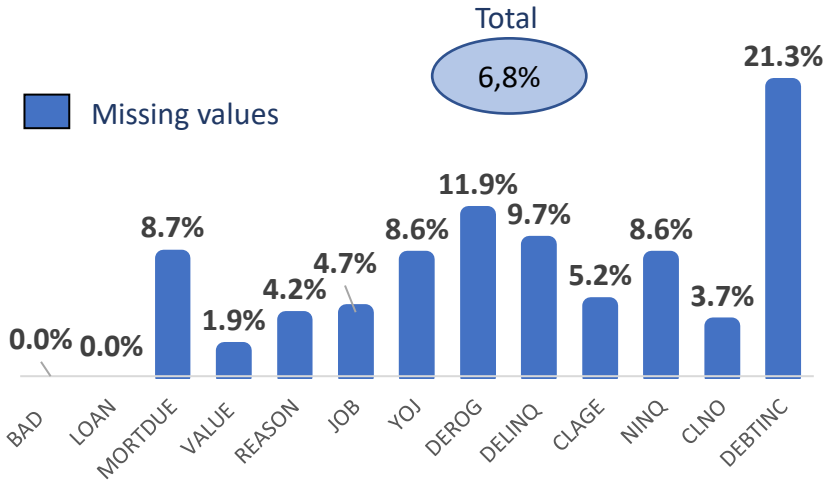


* Images taken from databricks.com

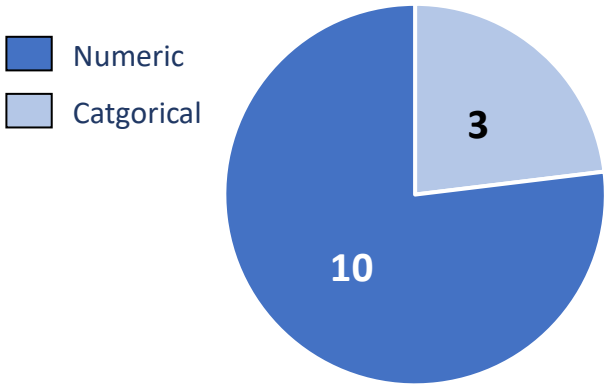
Data Insights



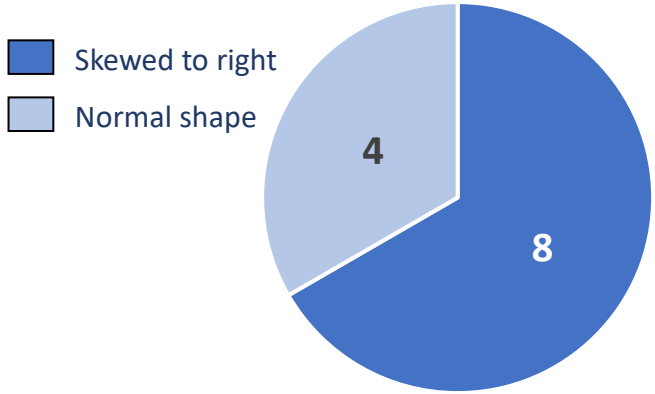
- Data contains 5,960 registers and 13 features.
- Data is unbalanced in a 80%-20% proportion.
- A balancing process of the data is needed to modeling.



- 6,8% of the data has missing values.
- All features have missing values.
- A filling data process is needed using median and mode.



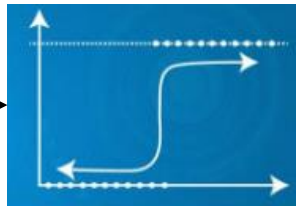
- Data types are distributed in 10 numeric features and 3 categorical.
- Categorical data needs to be treated in order to use it.



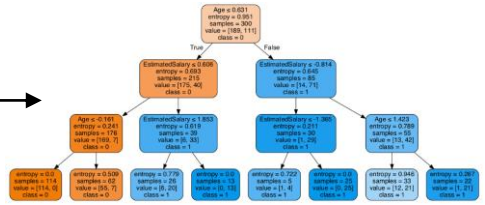
- 8 features are skewed to the right and 4 are normal shaped.
- All features have a big amount of outliers.
- Outliers need to be treated for Logistic Regression modeling

Proposed model solutions

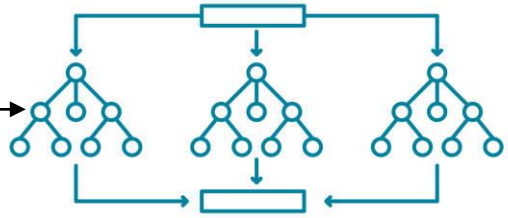
Logistic regression



Decision Trees



Random Forest



Model performances

Baseline model

Logistic Regression

AC	RC	PR	F1
.81	.08	.68	.15

AC \ PR		
	1417	14
	327	30

Decision tree

AC	RC	PR	F1
.88	.62	.73	.67

(Overfitting)

AC \ PR		
	1349	82
	134	223

Random Forest

AC	RC	PR	F1
.91	.68	.84	.75

(Overfitting)

AC \ PR		
	1345	46
	115	242

Tuned model

Logistic Regression

AC	RC	PR	F1
.76	.41	.39	.40

AC \ PR		
	1208	223
	212	145

Decision tree

AC	RC	PR	F1
.86	.74	.62	.68

AC \ PR		
	1270	161
	93	264

Random Forest

AC	RC	PR	F1
.91	.68	.82	.75

AC \ PR		
	1379	52
	113	244

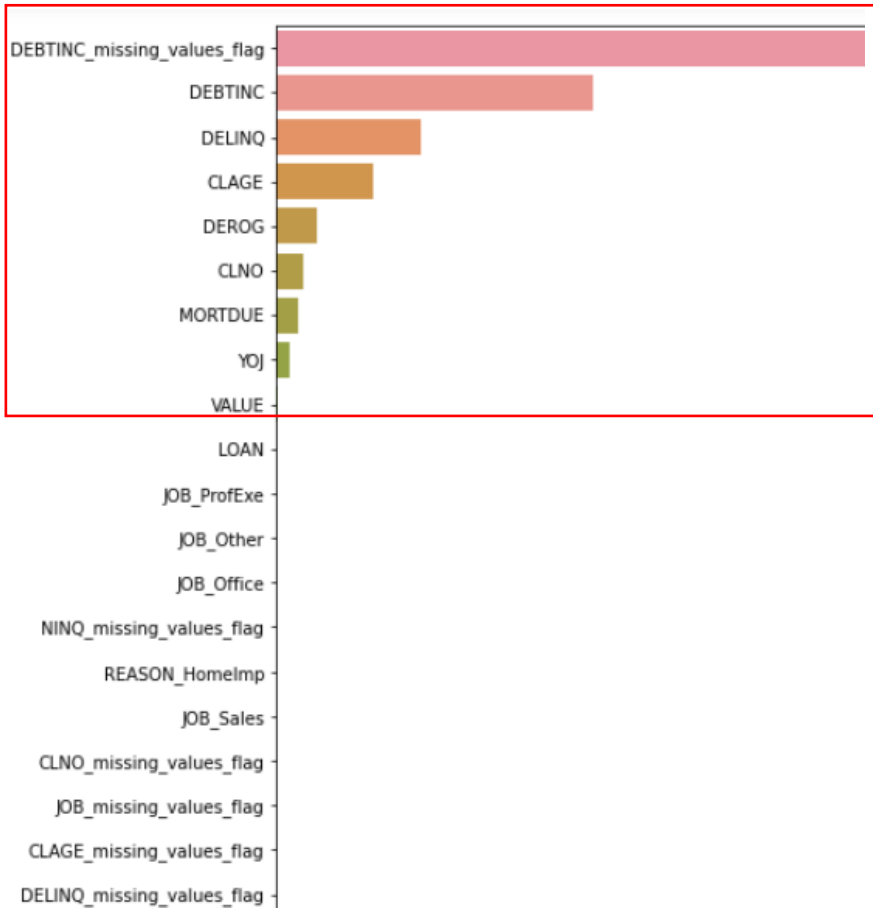
Comparison of techniques and performances

Models	Accuracy	Recall	Precision	Pros	Cons
Tuned Random Forest	0.91	0.68	0.82	-Higher precision than Decision Trees	-Non interpretable
Random Forest	0.90	0.63	0.85	-Higher precision than Decision Trees	-Non interpretable
Tuned Decision Tree	0.86	0.74	0.62	-Highest recall -Interpretable	-Lower precision than Random Forest
Decision Tree	0.88	0.62	0.73	-Similar recall than Random Forests. -Interpretable	-Lower precision than Random Forest

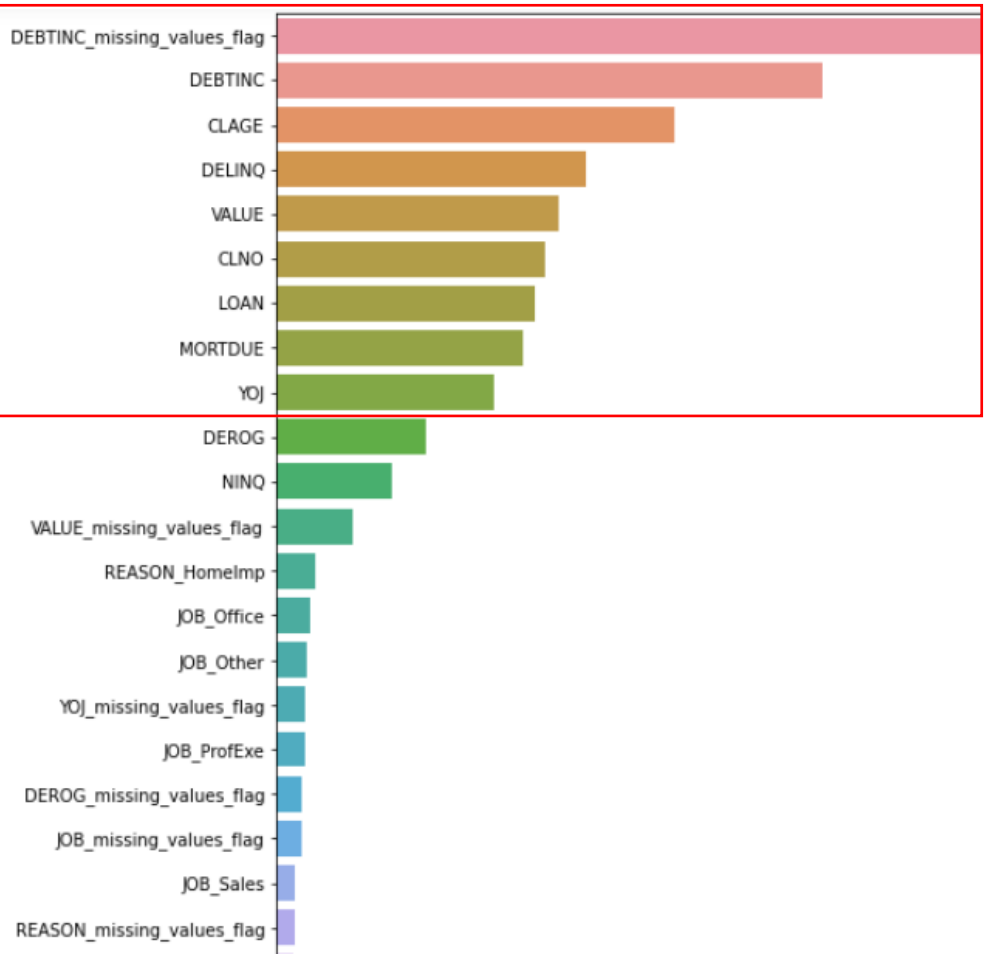
- * Data observed from test sets
- ** Logistic Regression wasn't included due to its very low recall

Importance features

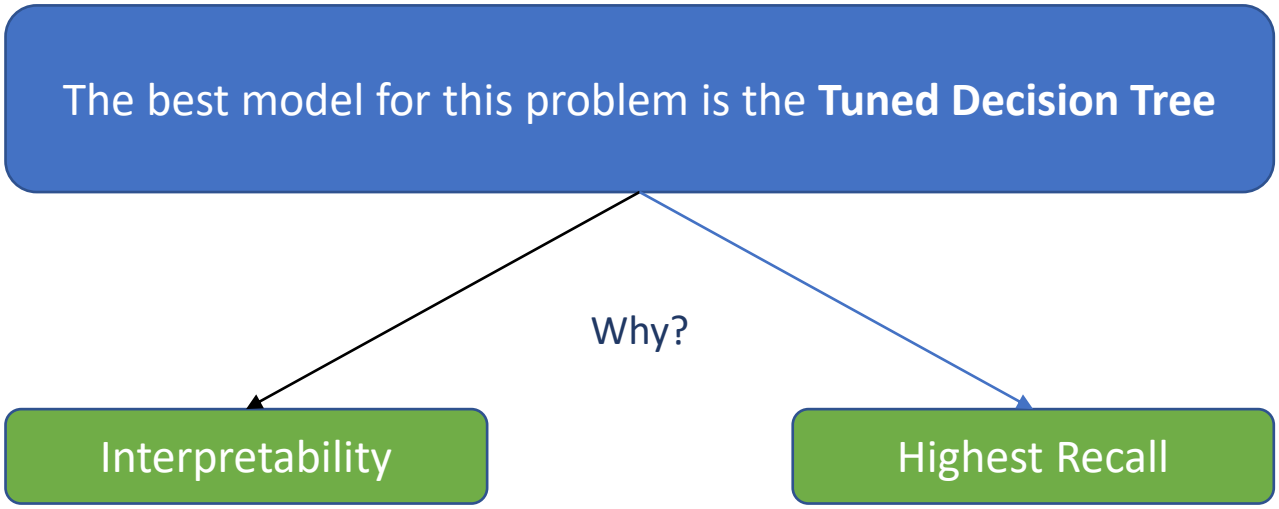
Tuned decisión tree



Tuned random forest



Proposal for the Final Solution Design



- The Tuned Decision Tree is giving 0.74 recall (highest), 0.86 accuracy and a precision of .62
- A decision tree is by nature very interpretable
- The most important features used by it are the same than the tuned random forest.

Executive summary

- A decision tree model can predict loan defaulters 74% of the time they come to ask for a home loan.
- This model is highly interpretable.
- The most important features to make a proper prediction are DEBINC, DELINQ and CLAGE.
- The Debt/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%).
- It is recommendable to explore the possibility to create an alternative business process to manage and take decisions on those clients with no Debt/Income ratio available.

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, support vector machine, neural networks, among others.
- 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.
- Check if there is a way to complete the missing values in the data.

Risks and challenges

- The major risk associated with this project is to underperform versus the current and manual process.
- A big challenge will be changing the internal culture of the bank to adapt the new model to it.
- Another challenge is to exceed the current incomes with the new model.

Appendix

Why is recall important to this project?

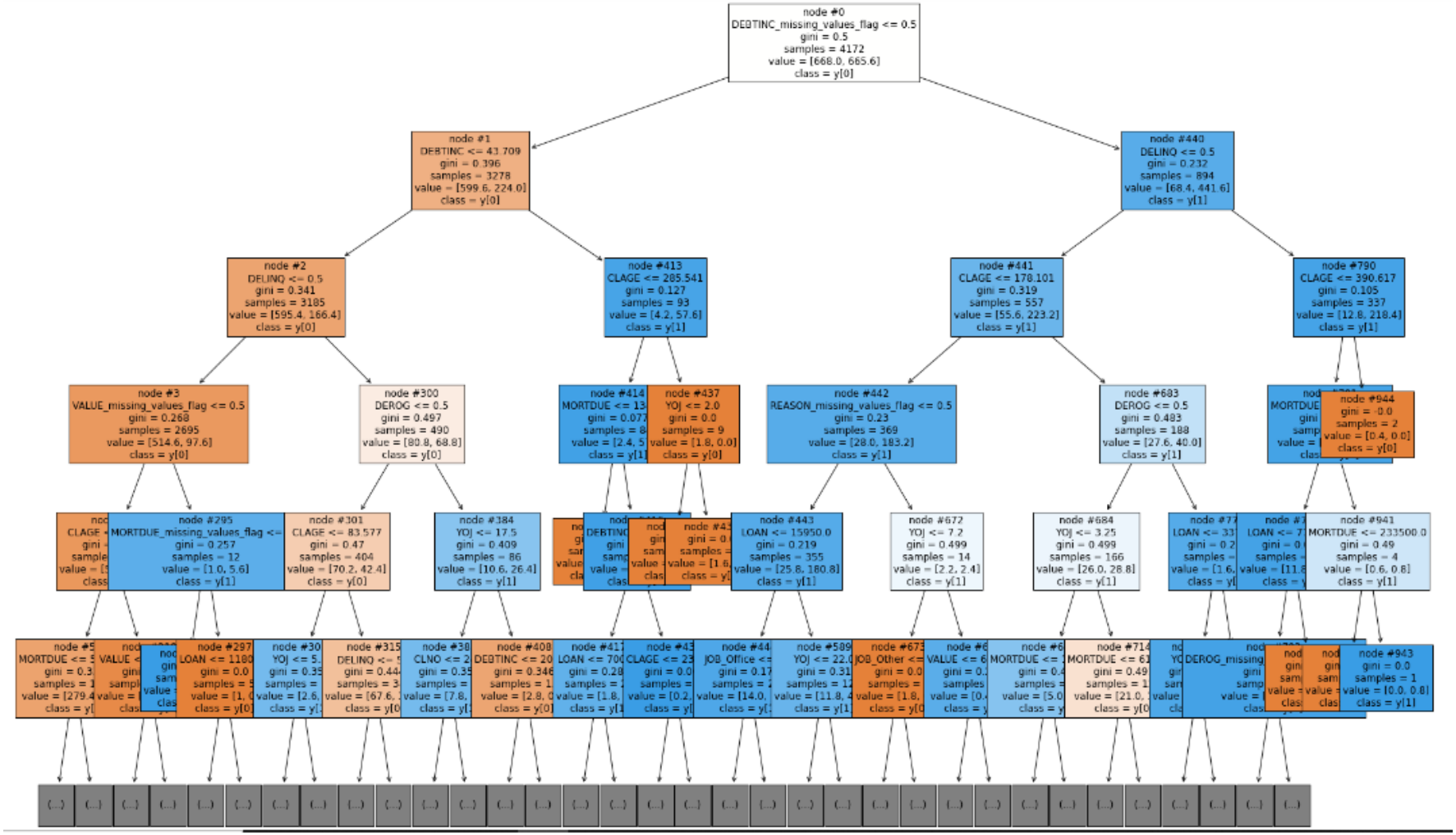
The model can make two types of wrong predictions:
1.Predicting a client will pay his loan when the client can't pay.
2.Predicting a client won't pay his loan when the client can pay.

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 #440
DELINQ <= 0.5
gini = 0.232
samples = 894
value = [68.4, 441.6]
class = y[1]
node #2
DELINQ <= 0.5
gini = 0.341
samples = 3185
value = [595.4, 166.4]
class = y[0]
node #413
CLAGE <= 285.541
gini = 0.127
samples = 93
value = [4.2, 57.6]
class = y[1]
node #441
CLAGE <= 178.101
gini = 0.319
samples = 557
value = [55.6, 223.2]
class = y[1]
node #790
CLAGE <= 390.617
gini = 0.105
samples = 337
value = [12.8, 218.4]
class = y[1]
node #3
VALUE_missing_values_flag <= 0.5
gini = 0.268
samples = 2695
value = [514.6, 97.6]
class = y[0]
node #300
DEROG <= 0.5
gini = 0.497
samples = 490
value = [80.8, 68.8]
class = y[0]
node #414
MORTDUE <= 13
gini = 0.077
samples = 8
value = [2.4, 5
class = y[1]
node #437
YOJ <= 2.0
gini = 0.0
samples = 9
value = [1.8, 0.0]
class = y[0]
node #442
REASON_missing_values_flag <= 0.5
gini = 0.23
samples = 369
value = [28.0, 183.2]
class = y[1]
node #683
DEROG <= 0.5
gini = 0.483
samples = 188
value = [27.6, 40.0]
class = y[1]
node #944
gini = 0.0
samples = 2
value = [0.4, 0.0]
class = y[0]
node #295
MORTDUE_missing_values_flag <=
gini = 0.257
samples = 12
value = [1.0, 5.6]
class = y[1]
node #301
CLAGE <= 83.577
gini = 0.47
samples = 404
value = [70.2, 42.4]
class = y[0]
node #384
YOJ <= 17.5
gini = 0.409
samples = 86
value = [10.6, 26.4]
class = y[1]
node #443
LOAN <= 15950.0
gini = 0.219
samples = 355
value = [25.8, 180.8]
class = y[1]
node #672
YOJ <= 7.2
gini = 0.499
samples = 14
value = [2.2, 2.4]
class = y[1]
node #684
YOJ <= 3.25
gini = 0.499
samples = 166
value = [26.0, 28.8]
class = y[1]
node #941
MORTDUE <= 233500.0
gini = 0.49
samples = 4
value = [0.6, 0.8]
class = y[1]
node #297
LOAN <= 1100
gini = 0.0
samples = 5
value = [1,
class = y[0]
node #315
DELINQ <=
gini = 0.44
samples = 3
value = [67.6,
class = y[0]
node #408
DEBTINC <= 20
gini = 0.346
samples = 1
value = [2.8,
class = y[0]
node #411
LOAN <= 700
gini = 0.28
samples =
value = [1.8,
class = y[1]
node #43
CLAGE <= 23
gini = 0.0
samples =
value = [0.2,
class = y[1]
node #44
JOB_Office <=
gini = 0.17
samples =
value = [14.0,
class = y[1]
node #589
YOJ <= 22.0
gini = 0.31
samples = 12
value = [11.8, 4
class = y[1]
node #673
JOB_Other <=
gini = 0.0
samples =
value = [1.8,
class = y[0]
node #6
VALUE <= 6
gini = 0.2
samples
value = [0.
class = y
node #6
MORTDUE <=
gini = 0.4
samples =
value = [5.0
class = y
node #714
MORTDUE <= 61
gini = 0.49
samples = 1
value = [21.0,
class = y[0]
node #943
gini = 0.0
samples = 1
value = [0.0, 0.8]
class = y[1]