BNP Paribas Cardif Claims Management

Elsa Mattson, Johnny Shapiro, Hui Wang, Tyler Williamson

Agenda

1. Problem Statement & Data
2. Models and Algorithms
3. Critique
4. Our Final Model

Problem Statement

Problem

BNP Paribas Cardif is global specialist in personal insurance. They want to find out how to predict the category of a claim based on features available early in the process, helping them accelerate their claims process and therefore provide a better service to their customers.

In this challenge, BNP Paribas Cardif is providing an anonymized database with two categories of claims - 1 for claims suitable for accelerated approval and 0 for claims unsuitable for accelerated approval.

DATA

Anonymized Dataset

114,321 Observations

133 Columns:

- **1** column of observation ID
- **1** binary target variable
- **4** integer variables
- **19** factor variables
- **108** numeric variables
- **Variables** are unnamed and ambiguous

Models and Algorithms

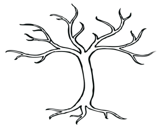
“Bare-Bones” Random Forest - developed from publicly available Kaggle Submission

Simple xgboost Model - developed from publicly available Kaggle submission

Random Forest - constructed using the h2o package in R

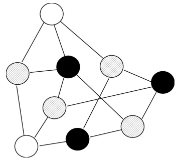
Final GBM Model - constructed using the h2o package in R

Models and Algorithms Results



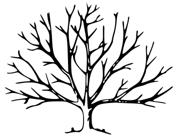
Bare Random

Log-loss: 0.50600



xgboost gbm

Log-loss: 0.46824



h2o Random Forest

Log-Loss: 0.515319

h2o GBM

Log-loss: 0.4684028

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right)$$

Critique One - “Bare Bones” Random Forest

Algorithm / Model - simple random forest that only utilizes numeric variables

Time Complexity - takes slightly less than 10 minutes to run

Data Processing - minor explanations provided, variables not defined (though they are self-explanatory). Time taken to run the variables is the longest out of all the code.

Model Issues:

- Too much reliance on roughfix to impute missing values
- Assumes that using only numeric columns for training/testing is the most efficient method
- While this example does its job, it oversimplifies the problem for the sake of inducing results

Reproducibility - Bare Bones Random Forest

```r
library(randomForest)
set.seed(35)

train <- read.csv("../input/train.csv")
test <- read.csv("../input/test.csv")

# Use only numeric cols
train <- train[, sapply(train, is.numeric)]
test <- test[, sapply(test, is.numeric)]

train <- na.roughfix(train)
test <- na.roughfix(test)

rf <- randomForest(as.factor(target) ~ ., data = train, ntree = 200)
yhat <- predict(rf, test, type="prob")[,2]

write.csv(data.frame(ID = test$ID, PredictedProb = yhat), "random_forest_benchmark.csv", row.names = F)
```

Critique Two - GBM with xgboost

Algorithm / Model - Gradient Boosting Machine with xgboost package

Time Complexity - 1 minute 15 seconds

Data Processing

- Identified predictors with > 15,000 NA's
- For predictors with > 15,000 NA's replaced with negative of maximum column value
- Identified predictors with <= 15,000 NA's and replace with *medianImpute* function

Biggest Issue: Variance reduction

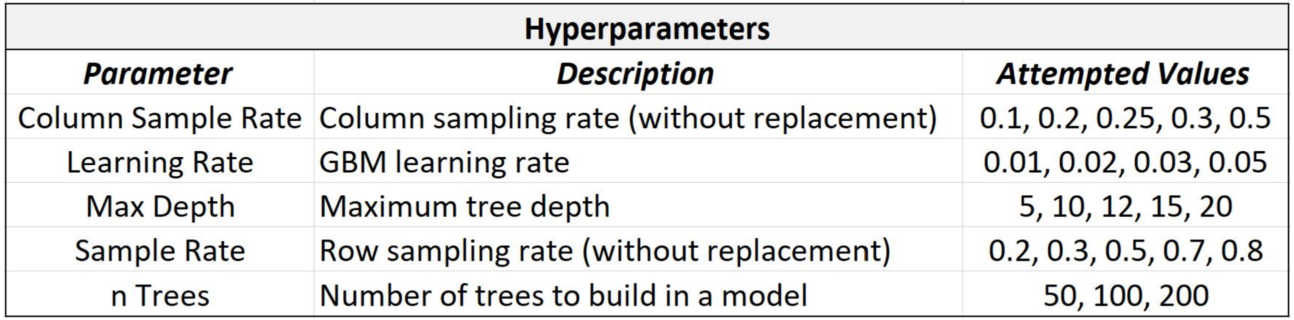
Reproducibility - xgboost

Main Issues

- Vague code comments, no description of why the methods the team used were chosen.
- No hyperparameter optimization or justification for the chosen parameters.
- Training and validation done on a specified number of observations, not a percentage of the total, with no explanation of why that number was chosen.

Final Model - Theoretical Claim

We use Gradient Boosting Machine (**GBM**) to perform **hyperparameter optimization** and identify the best available model metrics for this classification task using grid search with the **h2o** package in R



Hyperparameters		
Parameter	***Description***	***Attempted Values***
Column Sample Rate	Column sampling rate (without replacement)	0.1, 0.2, 0.25, 0.3, 0.5
Learning Rate	GBM learning rate	0.01, 0.02, 0.03, 0.05
Max Depth	Maximum tree depth	5, 10, 12, 15, 20
Sample Rate	Row sampling rate (without replacement)	0.2, 0.3, 0.5, 0.7, 0.8
n Trees	Number of trees to build in a model	50, 100, 200

Final Model - Key Definitions

Gradient Boosting Machine Algorithm – training an initial decision tree with equal weights, evaluate performance and adjust weights, grow subsequent trees on adjusted weights to improve prediction

Log-Loss Function - indicates how close prediction probabilities are to corresponding observed values - The more divergence (error), the higher the log-loss, and vice versa

ROC Curve - probability curve where AUC represents the degree of separability - the higher the AUC the better the model is at predicting 0 classes as 0 and 1 classes as 1

Final Model - Data

Data - 114,321 Observations, 1 binary target, 129 Variables

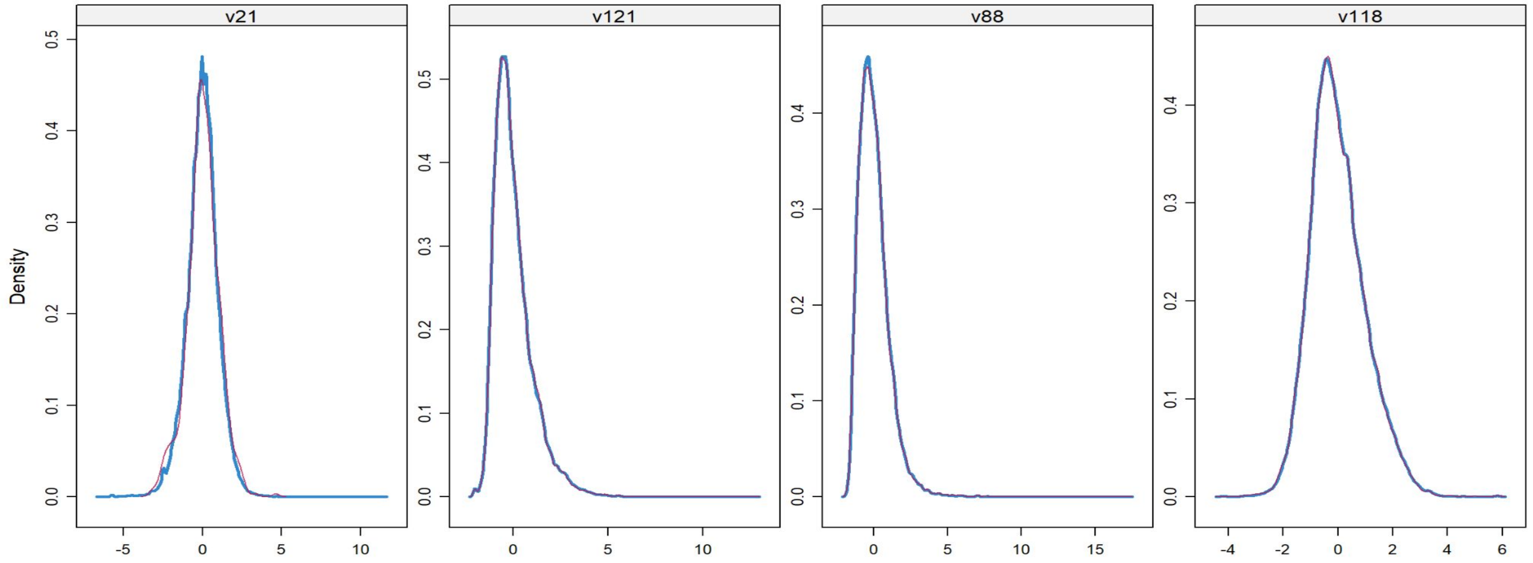
Excluded Data - v30 (factor variable with **60,110 NA's**) & **v113** (factor variable with **55,304 NA's**)

Train / Validation Splits - 80% Training and 20% Validation

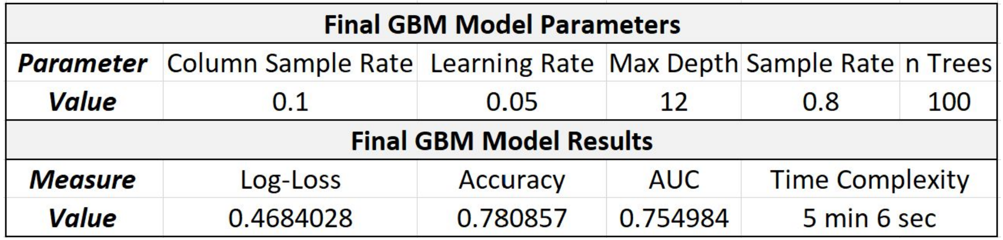
MICE package: assumes that the missing data are Missing at Random (MAR), which means that the probability that a value is missing depends only on observed value and can be predicted using them.

PMM(Predictive mean matching) - imputing method that involves selecting a datapoint from the nonmissing data which has a predicted value close to the predicted value of the missing sample.

PMM Density Plots



Final Results & Conclusions



Final GBM Model Parameters					
Parameter	Column Sample Rate	Learning Rate	Max Depth	Sample Rate	n Trees
Value	0.1	0.05	12	0.8	100

Final GBM Model Results				
Measure	Log-Loss	Accuracy	AUC	Time Complexity
Value	0.4684028	0.780857	0.754984	5 min 6 sec

- **Data preprocessing (*always look at your data!*)**
    - Missing value treatment for categorical and numeric variables
    - NA values and null values
- **User-friendly Packages like h2o**
    - Efficient for large datasets
    - Ease of use when testing model parameters