Data Challenge - Report

Marc KASPAR (M2 IASD) Caio Rocha (M2 IASD)

March 2025

1 Introduction

The objective of the Crédit Agricole challenge is to develop predictive models for the Multi-Risk Agricultural Contract, requiring the development of two separate models: one to predict the frequency of claims (FREQ) and another to predict the average cost of claims (CM). The final target variable, CHARGE, is derived by multiplying the predicted frequency, average cost, and the number of years since the contract was taken out (ANNEE_ASSURANCE).

The goal is to outperform a baseline model based on Generalized Linear Models (GLMs) using Poisson and Tweedie distributions for frequency and average cost, respectively. To address this challenge, we experimented with various models, including Random Forest (RF), XGBoost (XGB), Linear Regression, and Multi-Layer Perceptrons (MLPs). Each model was evaluated based on its ability to predict the frequency and average cost of claims.

2 Task Division

We divided the tasks according to the following:

- Caio Rocha: Related Work, **Preprocessing** – Encoding of Binary and Categorical variables; **Models** – Random Forest, XGBoost, Neural Network; **Feature Importance**.
- Marc Kaspar: **Preprocessing** – Encoding of Mixed and Numerical variables; **Models** – Linear Regression, MLP.

3 Related Work

We conducted a small review of the algorithms used for this kind of problem in the literature, finding that neural networks and GLMs are the most used baselines [5] [1]. The Poisson distribution is used for modeling claim frequency because it is suitable for count data, assuming independence of claim frequency [6] [3]. Gamma distributions are also often used for modeling claim severity due to their ability to capture skewed data.

[1] advocate for robust preprocessing techniques, such as feature encoding and normalization, as a crucial part of the prediction process. [4] present a framework to forecast insurance claims, recommending RFs as satisfactory predictive models. Finally, [2] explored XGBoost to predict insurance claims, obtaining better accuracies in term of normalized Gini than other methods.

4 Data analysis

Figure 1 shows the distribution of a random sample of the training data. We observe that most columns are categorical, including binary columns in this case. We need to convert these columns to a numerical representation in order to feed the regression models. This is explained in section 5.

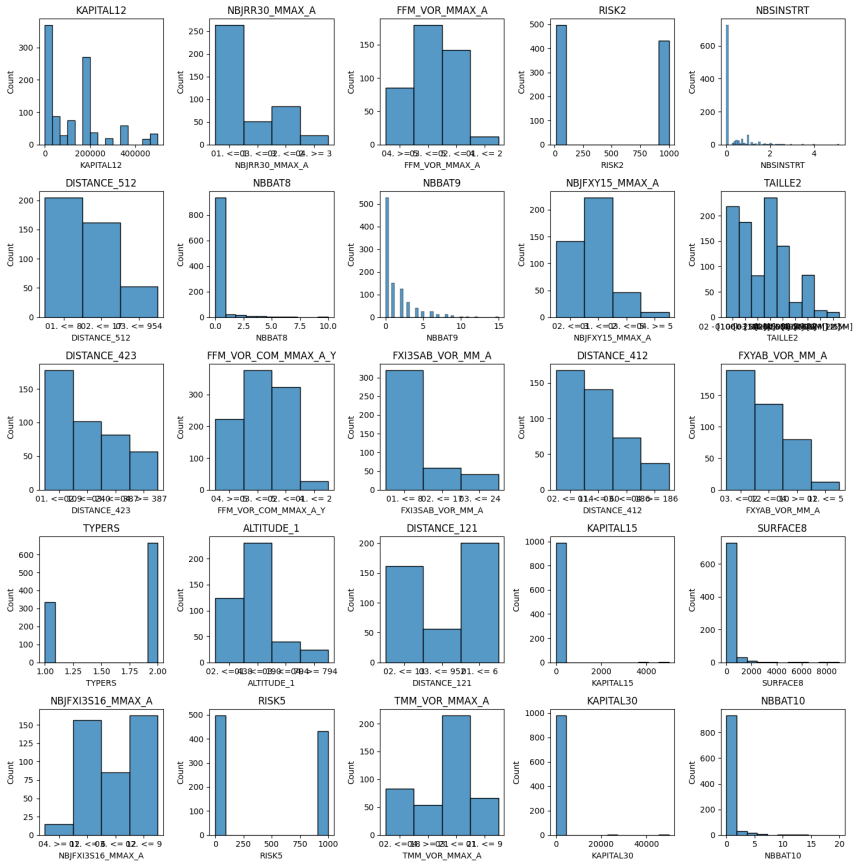


Figure 1: Illustration of distribution of the data for a random subset of 25 attributes and 1000 observations.

Figure 2 displays the distribution of the variables in y_train. The variable "CHARGE" is the target variable that is composed of "FREQ" and "CM". It is highly correlated to "CM" ($\rho = 0.995$), while it is only poorly correlated to "FREQ" ($\rho = 0.141$).

4.0.1 Feature Importance

We aimed to investigate whether there was any particular variable that was most important to the trained models in order to think of smarter training strategies, so we plotted the SHAP values and the internal feature importances of the trained RF model, shown in Figure 3. The SHAP feature importance for RF estimates the contribution of each feature in the final prediction by exploring the paths through the trees where the feature affects the outcome, evaluating how the prediction changes when the feature is included versus when it is not. The Gini importance is computed during

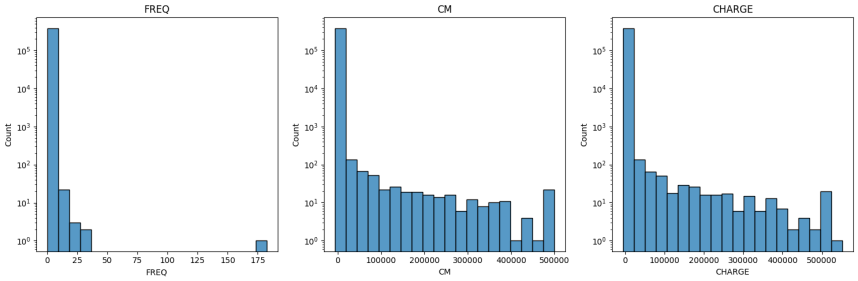


Figure 2: Histograms of the target variables in log scale.

training based on how much each feature reduces the impurity in the decision-making process of the trees.

In the plot, we can't identify any significantly more important feature, which indicates that the information of the "FREQ" target variable is distributed across all features.

5 Preprocesssing

We focused a great deal of time ($\sim$ 40%) on the preprocessing and encoding steps, since we observed that they significantly impacted the results. We first removed the columns "ID" and "ANNEE_ASSURANCE", which represents an attribute of the target data y_train. The variables are on different scales, containing a mix of categorical and numerical features and several NaNs.

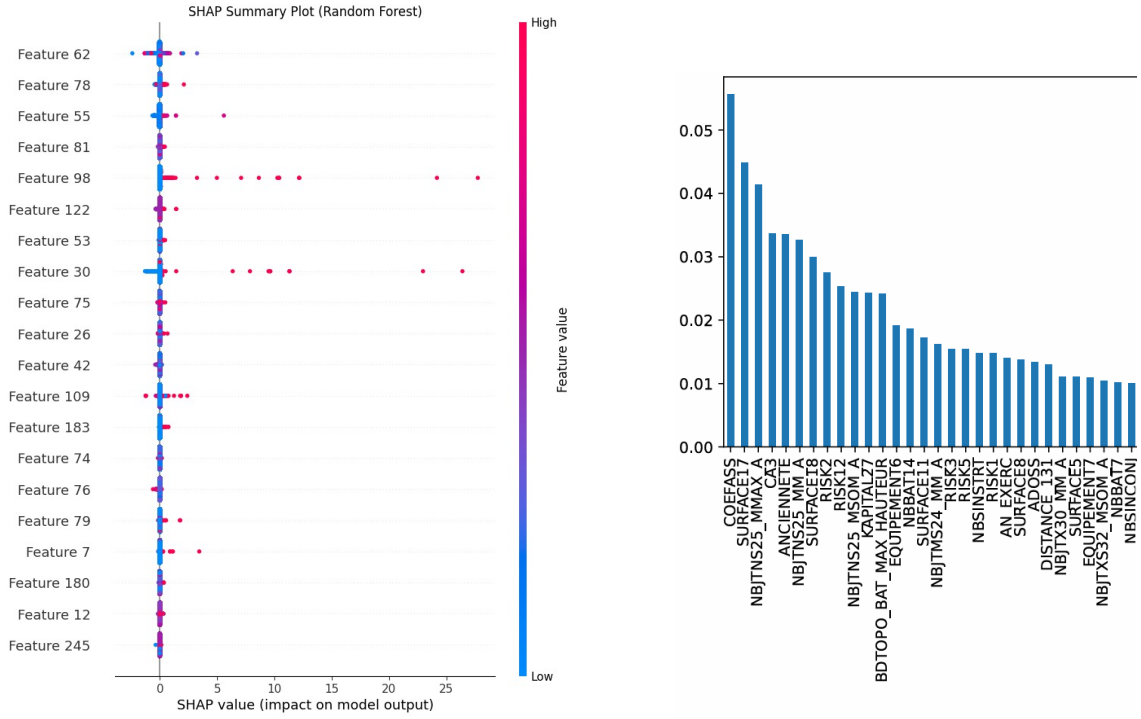
5.1 Categorical columns

Binary columns and categorical columns were encoded with an ordinal encoder, which represents the categories as a sequence of numbers from 0 to $n_{categories} - 1$. There were also columns such as "ADOSS", "CARACT" and "INDEM" which assume the values N (Non), O (Oui) and R. In this case, N became 0, O became 1 and R became 2. Other columns included a scale of values represented in inequalities, e.g. "PROPORTION" and "DISTANCE", having inequalities on the form $XX. <= YY.$. Others included more straightforward order information, such as "TAILLE" and "COEFASS", which included information in the form of ranges of values (e.g. $01-[0-250k]$ $02-[250k-500k]$ ... or 0 $01-10$ $11-20$ ... respectively).

We applied one hot encoding to the remaining 7 categorical columns (['TYPBAT1', 'DEROG12', 'DEROG13', 'DEROG14', 'DEROG16', 'EQUIPEMENT5', 'ESPINSEE']), dropping the dummy column created to represent NaN data, since it is not relevant.

5.2 Mixed columns

Mixed columns contained to type of data. There were two : columns 'FRCH1' and 'FRCH2'. For example,'FRCH1' contained integers from 0 to 3 as well as string versions as these numbers and the string 'i'. Figure 4 shows this distribution. To solve this issue, we decide to first transform all string values of numbers into integers to get similar values. Figure 5 shows this transformation. We then decided to separate these columns in two to take care of the remaining string values. A string column would contain all the string values and a numerical column would contain the numerical



(a) SHAP Feature Importance.

(b) Gini Feature Importance.

Figure 3: Random Forest.

values. The missing values where replaced with NaNs. Then for the string column, we performed a one-hot encoding, while for the numerical column, we imputed using the mean and then normalized.

5.3 Numerical columns

Numerical columns containing NaNs were filled with the mean values, and next were normalized according to the z-score (i.e. columns were assigned zero mean and unit standard deviation). The test data were normalized according to the train data mean and variance. Figure 6 shows that most features have a mean of order of magnitude of up to 10^2, however a select few have means with order of magnitude 10^5! This causes the regression models to attribute very high importance to them. The normalization process is crucial since it normalizes all features to a mean of 1, avoiding biases.

The columns $['SURFACE4','SURFACE6','DISTANCE\_244','IND\_Y1\_Y2','IND\_INC']$ contained only NaN after preprocessing, so they were removed from the experiment.

5.4 Size and type of the data

A summary of the sizes of the training and test data is presented in Table 1 and the types of the attributes are presented in Table 2. The reason why after preprocessing we have more features is the one hot encoding step which added new columns. We also converted the data to tensor which is a better representation for the neural network models.

The percentage of NaN data before preprocessing is 30.57 % and after preprocessing it is 0 %.

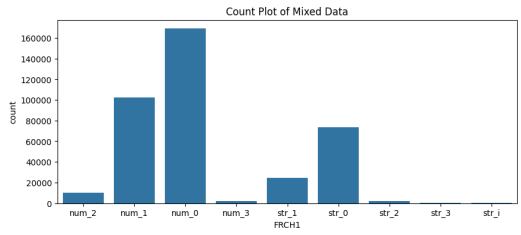


Figure 4: Distribution of data of column 'FRCH1'. Numerical and string values have been separated for plotting

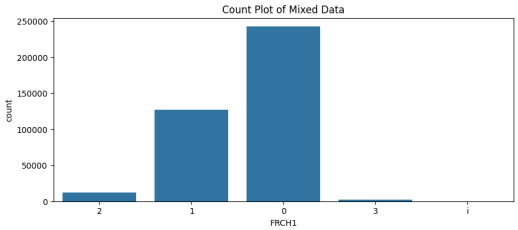


Figure 5: Distribution of data of column 'FRCH1' post processing

	X_train	y_train	X_test
# observations	383610	383610	95852
# features before preprocessing	372	4	372
# features after preprocessing	399	4	399

Table 1: Dimensions of the data, where X is the input and y is the output.

5.5 Train / validation split

Since training the models with all the data took a lot of time, we decided to make smaller tests with only 50,000 observations and then retrain the best models with all observations. We used 80% of the data as training data and 20% as validation data.

6 Models

We carried out several experiments in order to choose the models, described in the subsections below.

6.1 Random Forest and XGBoost

Since the example notebook featured training a random forest (RF) model to predict the "FREQ" and a XGBoost model (XGB) to predict the average cost "CM", we first experimented with these models aiming to select the best preprocessing and encoding strategy. This was useful because it provided a baseline to compare to other models. Table 3 displays the results obtained with different encoding strategies.

We also noticed that when the "FREQ" is 0, the "CM" is also 0, however that value is not representative of the real cost in case of an incident, and thus we also tried training "CM" using only inputs with a non-zero "FREQ" however, we end up having worst results. We think the reasons are that the number of elements with non-zeros is quite small and that the zeros elements helped the model to be a bit more conservative with its prediction and thus lowered overall error.

6.2 Neural Networks

The description of the challenge mentions that the baseline consists of a regression model based on the Poisson distribution to predict the "FREQ" and one based on the Tweedie distribution to predict the "CM".

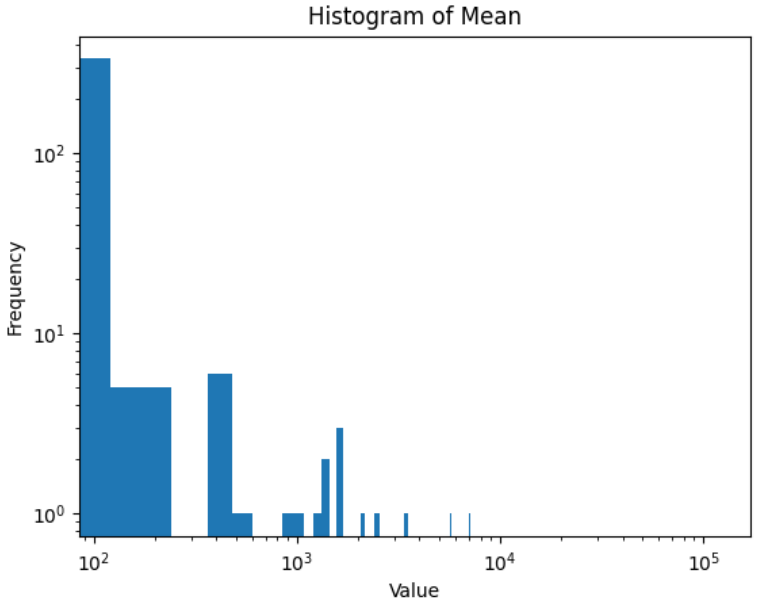


Figure 6: Histogram of means of each feature.

	Numerical	Categorical	Mixed
Before preprocessing	92	278	2
After preprocessing	399	0	0

Table 2: Input data attribute types. The mixed columns contained a mix of categorical and numerical values.

We followed this advice and implemented neural networks which had these loss functions, however these models failed to converge. We tried different numbers of layers, hidden layer sizes and dropout, but we couldn't make it work. Also, quite possibly, there was an error in the implemented of the Gamma loss, since it is not natively implemented in Tensorflow and often times would diverge to NaN. The results obtained are shown in Figure 7 and Figure 8.

6.3 Linear Regression

We decided to test a linear regression model as the problem seems simple enough and well adapted to this kind of model. We also tested using regularization, and we reported these values in the Table 4. Lasso regularization seems to give the best values for the validation data, however, this is not consistent with the platforms results as Ridge regression has the best results. The scale of the numbers is also very different which we are unable to explain. Overall, Linear Regression still gave us the best results. We are currently sitting at fifth place with almost 80 participants.

Train set size	Encoding	Normalization	Public score
383,610	Ordinal + Binary + One hot	Yes	5605.205184
383,610	Count + One hot	Yes	5606.166742
383,610	Ordinal + Binary + One hot	No	5607.787574
50,000	Ordinal + Binary + One hot	Yes	5725.741728
50,000	Count + One hot	Yes	5735.751156
50,000	Ordinal + Binary + One hot	No	5754.734910

Table 3: Results obtained with the Random Forest model to predict "FREQ" and XGBoost to predict "CM". The best public score was achieved using ordinal encoding, binary encoding, one hot encoding, and normalization of numerical features.

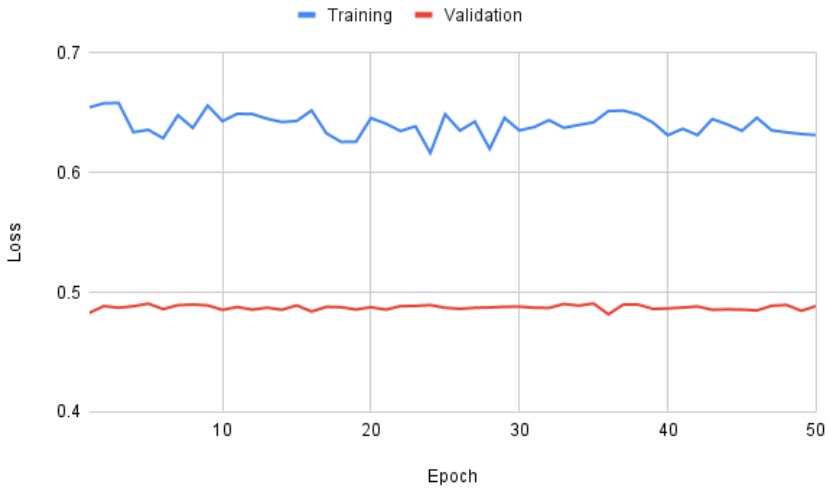


Figure 7: Poisson loss to predict "FREQ" with 0.0001 learning rate, batch size 1024, and 3 hidden layers.

7 Future Work

Despite encountering challenges with neural network convergence, our linear regression model achieved competitive results, placing us in the top five of the competition. While our current model predicts decently well, we believe there are still improvements to be made. Firstly, our pre-processing can still be further improved. Also, we believe that we haven't experimented enough with Random Forest and XGBoost and tuning the hyperparameter might give us better results. We would also want to give neural networks a second try as we believe they might give decent results if trained well.

References

[1] Piet de Jong and Gillian Z. Heller. *Generalized Linear Models for Insurance Data*. International Series on Actuarial Science. Cambridge University Press, 2008.

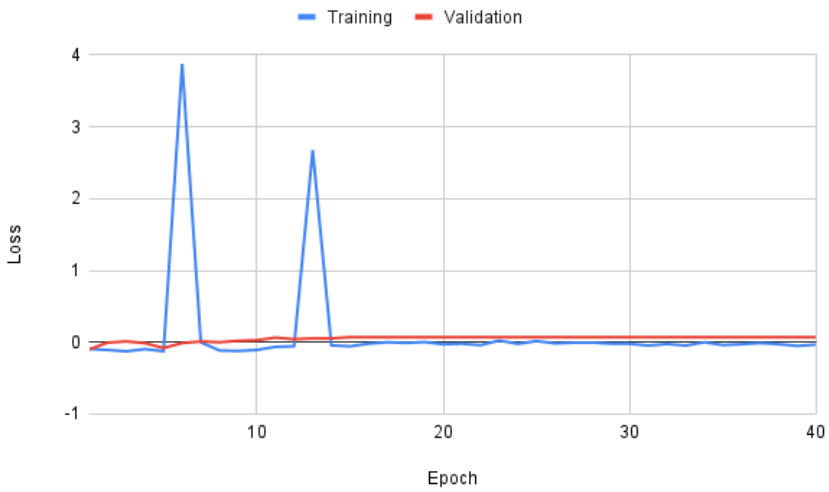


Figure 8: Gamma loss to predict "CM" with 0.0001 learning rate, batch size 1024, and 3 hidden layers. The negative values indicate an error in the implementation that we couldn't exactly find.

[2] Muhammad Arief Fauzan and Hendri Murfi. The accuracy of xgboost for insurance claim prediction. *International Journal of Advances in Soft Computing and its Applications*, 10(2):159–171, 2018. Publisher Copyright: © 2018, International Center for Scientific Research and Studies.

[3] Joseph M. Hilbe. *Poisson Regression*, page 35–73. Cambridge University Press, 2014.

[4] T. (Tim) Pijl. A framework to forecast insurance claims. Master's thesis, October 2017.

[5] Anil Fernando Umar Isa Abdulkadir. A deep learning model for insurance claims predictions. *Journal on Artificial Intelligence*, 6(1):71–83, 2024.

[6] Matthew D. Urban. *Regression Models*. Bookdown, 2023.

Model	Regul. (α)	Frequence		Mean Cost		Charge		Platform
		Train	Val	Train	Val	Train	Val	
Lin Reg	–	**0.2730**	0.3710	5945.9390	6855.7070	49.75	100.72	5603,669
Lasso	0.01	0.2754	**0.3679**	5977.8804	**6799.1216**	7.32	8.32	X
	0.1	0.2754	**0.3679**	5982.2603	6801.0586	**2.84**	**3.00**	X
	1.0	0.2754	**0.3679**	5982.2603	6801.0586	**2.84**	**3.00**	5604,807
	10.0	0.2754	**0.3679**	5982.2603	6801.0586	**2.84**	**3.00**	X
	100.0	0.2754	**0.3679**	5982.2603	6801.0586	**2.84**	**3.00**	X
Ridge	0.01	**0.2730**	0.8984	**5945.5728**	11762.1914	50.05	2195278.75	X
	0.1	**0.2730**	0.4205	5945.8203	7240.1064	49.79	129755.34	X
	1.0	**0.2730**	0.3716	5945.9053	6860.6704	49.71	1777.75	X
	10.0	**0.2730**	0.3710	5945.9277	6855.0103	49.21	101.52	X
	100.0	**0.2730**	0.3707	5946.1274	6849.9077	46.06	93.57	5603,664

Table 4: Comparison of linear regression models with different regularization values, 50,000 elements used for training