G2JN - Baseline Model Report

Extreme Gradient Boosting (XGBoost) is an implementation of gradient boosting algorithm, which is an ensemble learning method. Due to its high performance and scalability, it is a popular choice for large-scale machine learning tasks.
XGBoost builds an ensemble of decision trees, where each tree is trained to correct the mistakes of the previous one. A simple model is trained, and then iteratively new trees are added to the ensemble, each of which is trained to correct the errors of the previous trees. The final ensemble consists of all the individual trees, and it is this ensemble that makes the final predictions.

Analytic Approach

- XGBoost requires feature vectors and labels as input. It can handle both numerical and categorical features, as well as missing data.
- There are a number of prediction functions in XGBoost with various parameters. We used the default which simply returns the predicion for each row in the input

Model Description

- We use Implementation of the scikit-learn API for XGBoost regression.
- XGBoost regression has several hyperparameters and We used all the default values. Some of the most important hyperparameters include:
    - n_estimators – Number of gradient boosted trees. Equivalent to the number of boosting rounds.
    - learning rate: Controls the size of the gradient steps taken by the algorithm.
    - max_depth: This controls the maximum depth of the individual decision trees in the ensemble.
    - subsample: This controls the fraction of the training data that is used to train each individual tree.
    - lambda & alpha : L1 and L2 regularization term for each node.
    - early_stopping_rounds : stopping training when the validation metric stops improving for certain rounds.

Results (Model Performance)

- To evaluate the results on two data-sets which we trained the model on we used RMSE (Root Mean Squared Error) which is a measure of the difference between predicted and actual values.
- On the French Motor dataset RMSE = 5.59 and on the Boston House dataset RMSE = 3.48

Conclusion and Discussions for Next Steps

The results at this stage on the two data sets which we trained the basic model on using his default parameters did show good results in the RMSE dimension but there are still things that can be done to improve them and we will work towards that.

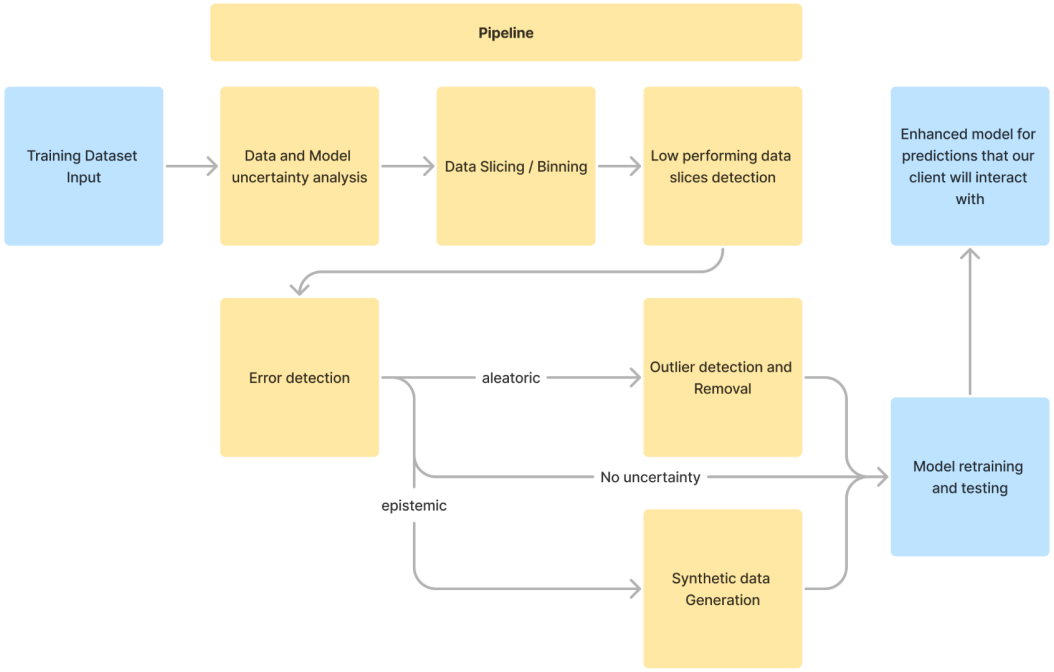
G2JN - Final Model Report

Analytic Approach

The model built was a pipeline that receives a dataset as input and modifies it and its samples in order to achieve better prediction results.
The input is any set of tabular data with the rows as samples and the columns as features, having one of the columns as the prediction labels in order to train the model. The task and dataset should be suited for regression purposes.
The output of the pipeline is a ML model trained on the enhanced dataset taking into considerations modifications within its samples which yield a better performance.

Solution Description

The following is a schema of the architecture built.



The architecture is based on performing different steps on a pipeline in order to analyze data and affect it (remove or generate) based on the results (type of uncertainty). In the end we get a ML model as output, already trained for further predictions.

Data and Features

The data used was derived from two datasets provided by client companies.
Real Estate company uses the 506 sample boston housing dataset from Boston Standard Metropolitan Statistical Area. It included 13 features. This is an example of the first 3 samples.

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03

All features were used for the model. The label and target value to be predicted was attribute MEDV that stands for the price of the house.

The **French Insurance company** uses its historic data of 678013 samples of claims of its customers related to their vehicle malfunctioning. It included 12 features. This is an example of the first 3 samples.

	IDpol	ClaimNb	Exposure	Area	VehPower	VehAge	DrivAge	BonusMalus	VehBrand	VehGas	Density	Region
0	1.0	1	0.10	D	5	0	55	50	B12	Regular	1217	R82
1	3.0	1	0.77	D	5	0	55	50	B12	Regular	1217	R82
2	5.0	1	0.75	B	6	2	52	50	B12	Diesel	54	R22

The target prediction value was *Frequency* and is derived from normalizing the number of claims of a vehicle (*ClaimNb*) with respect to its exposure (*Exposure*). After engineering *Frequency* attribute, those to features were dropped as well as *IDpol* as it only indicates a unique value for the sample, thus not providing relevant information to the model.
Non numerical features such as *VehBrand*, *Area* and *VehGas* were treated with one-hot-encoding technique so that they were suitable for the regression model.

The challenge on this dataset was related to how imbalanced it is. Having frequencies larger that the relative value of 730, almost 83% of the samples were concentrated with a Frequency value below 1.5.

Algorithm

As described in the architecture, the algorithm had several steps as follow.

1. **Confidence estimation**
   This step was performed based on MACEst which provides an agnostic estimator of confidence, based on the function we created called `get_conf_interval()`. We initially split the data on train (and sub splits as required by algorithm: train, calibration, confidence) and test. The standard and recommended parameters for MACEst were used. With the train set we

trained a baseline RandomRegressorForest from sklearn and used confidence set to predict from it. We then used the calibration set to initiate macest with the new samples. Finally, the test set was used to assess the confidence estimator for the model, considering 95% confidence interval.

The output of this step consists of a confidence interval (lower and upper bound) around each sample.

	lower_bound_pred	upper_bound_pred	diff_prop	y_pred
0	16.668435	33.156565	16.488129	25.492981
1	20.042229	39.553771	19.511543	29.003119

2. **Data Slicing and bin assignment**
   The whole dataset was then sliced into small chunks to be analyzed. The proposed technique was to split the samples into equal size (width) bins based on their prediction label.

   There is a fixed parametrized amount of samples per bin (we decided 20). According to the amount of samples in the total dataset, it will fall from it how many equally spaced bins we need to create. There is a cap of number of bins (1000).
   Then, each sample would be assigned to the bin that suits its prediction value.

	count_samples	start_bin_value	end_bin_value
0	0	-inf	1.227150
1	0	1.227150	1.484950
2	3	1.484950	6.029745
3	7	6.029745	7.437000

   This is responsibility is taken by a function called `assign_bins()`.

3. **Bin analysis and error/uncertainty detection**
   The same function also helps to determine if the bin is suspected low in data. This is done by calculating the mean number of samples that the bins have. If a bin is below the mean number of samples by a threshold, then it is labeled as “suspected low in data”.

   With the same reasoning, we calculate -within each bin- the mean value of the difference between lower and upper bound of the confidence interval. With the average confidence interval for each bin, we consider the bins with lowest confidence (the largest difference in the interval) to be the most uncertain. The threshold to consider a bin with uncertainty is if it is included in the top (25 parametrized by our algorithm) percentile of the bins according to its average confidence.

Finally, we were able to analyze each bin uncertainty and label it accordingly:
- Low in data and uncertain bins are considered to have Epistemic Error.
- Non low in data and uncertain bins are considered to have Aleatoric Error.
- If not uncertain, then no error is considered for that bin.

This step outputs the following data arrangement as an example:

	count_samples	start_bin_value	end_bin_value	suspected_low_in_data_bins	uncertainty_type
3	7	6.029745	7.437000	True	Epistemic
4	1	7.437000	7.865000	True	NaN
5	74	7.865000	14.915000	False	Aleatoric

4. **Aleatoric error handling with outlier removal**

   Each bin with aleatoric error was handled by a process of outlier detection and removal. PyOD library was used with its default settings of considering outlier the 5th neighbor to each sample and assessing euclidean distance. All samples considered as outliers by the library (used in the funcion `remove_outliers(df)`) where dropped from each bin.

5. **Epistemic error handling with data generation**

   Each bin with epistemic error was treated separately to generate data. The SDV library was used to generate tabular data. The generation model was set up as a variational encoder, using TVAE from the library, with its default settings.

   The function we created to deal with this responsibility is `generate_data()` and doubles the data in the bin, creating as much as one sample per existing sample.

6. **Model training**

   The last step, once the full dataset was modified by data generation and outlier removal, was training a simple model. This step used XGBoost Regressor with default settings. As output, we get a trained model on the enhanced dataset that can be used by the client to perform further predictions.

7. **Hyperparameters tuning**

   There are also some tuning that can be considered when applying our pipeline on a new dataset:

	Samples per bin	Max bins	Binning threshold method	Threshold	Percentile threshold	Min amount samples in bin
Min / default	20 (default = 30)	100 (default = 30)	“mean”	0.3	25	5
Max	50	1000	“median”	0.8	75	15

We provided a generalized parameters tuning function:

```
def parameters_tuning(X,y,name)
```

which can automatically tune those parameters by getting a dataset as an input.

The full code repository can be found in github.

Results

RMSE - Root mean squared error (RMSE) is the square root of the mean of the square of all of the error. The use of RMSE is very common, and it is considered an excellent general purpose error metric for numerical predictions.

	Boston housing dataset	French motor dataset
Baseline / RMSE	3.48	5.59
Our model / RMSE	3.27	5.59
Improvement	6%	0%

On the Boston housing dataset our pipeline managed to improve the RMSE by 6% while on the french motor dataset we were struggling to improve the RMSE.
The main reasons for challenges we were facing during the French motor dataset are:

1. Macest upper and lower bound were extremely high and low while *Frequency attribute* is mostly around zero. By using another model for example CQR we might be able to address this challenge and produce a tighter interval (lower uncertainty).

2. The binning strategy - around 94% of the data points were in this range 0 -0.732 (*Frequency*). By spreading the data into more bins that are not too evenly spread and not too sparse we would be able to classify more bins as aleatoric and epistemic uncertainty type (more data points in more bins) and therefore we would be able to improve the performance of our pipeline in such a high concentrated data set.
   This can be to Improve by clustering, supervised binning or even add other features value to the binning decisions.

G2JN - Exit Report of Project for Customer

Customers: French Insurance Company & Real Estate Startup
Team Members and involved stakeholders:

- Data Science team: Gil Zeevi, Gil Ayache, Nadav Elyakim and Joel Liurner
- From French Insurance Co: James Smith (data scientist), Mary Jones (Head of Risk)
- From Real Estate Stup: Robert Williams (data scientist), Patricia Brown (Head of Growth and Expansion)

Overview

We created G2JN, an agnostic data solution that deals with datasets to reduce errors in predictions. The solution is an automatic pipeline that analyzes the input dataset and decides whether to remove data and/or generate data to improve predictions. It is applicable for tabular datasets suitable for regression tasks.

Business Domain & Business Problem

French Insurance Co underwrites motor insurance policies. They have a current model for predictions, however, their specific request is “how can they reduce the error on their current prediction of claims” so that they can underwrite better their customers?. In the end, they need to perform better so that their bottom line profit increases by reducing the amount of wrong (underperforming) insurance policies given and being able to price higher the most high risk cases. Also, they want not to overprice so that they do not lose potential customers.

Real Estate Startup that is expanding its operations to Boston, USA. They are performing its research in order to be able to price the properties they sell. Their need is to price fairly each house so that (1) they continue building reputation in the market that will drive winning market share against its traditional competitors, (2) not underprice to leave profit on the table for them and the house owners and (3) not to overprice so that they do not close or lose a deal. They requested from us to improve their current model by being more accurate in predictions.

Data Processing.

For the french motor dataset, the original features were: *IDpol* (IDpol The policy ID), *ClaimNb* (Number of claims during the exposure period), *Exposure* (The exposure period), *Area* (The area code -C-), *VehPower* (The power of the car), *VehAge* (The vehicle age, in years), *DrivAge* (The driver age, in years), *BonusMalus* (Bonus/malus), *VehBrand* (The car brand -C-), *VehGas* (The car gas -C-), *Density* (The density of inhabitants in the city the driver of the car lives in), *Region* (The policy regions in France -C-).

Categorical features (labeled as -C-) were turned into numerical by one-hot-encoding strategy. The target or prediction variable was calculated as a normalization of *ClaimNb* relative to the *Exposure* period. This was an engineered feature that we named ***Frequency***.
Then *ClaimNb* and *Exposure* features (together with *IDpol* as it is a unique value per policy) were dropped from the set. Finally, 678013 claims were used as samples for the model.

For the Boston housing dataset, the original features were: *CRIM* (per capita crime in town), *ZN* (proportion of residential land zoned for lots over 25,000 sq.ft.) , *INDUS* (proportion of non-retail business acres per town), *CHAS* (boolean true if bounds Charles River), *NOX* (nitric oxides concentration) , *RM* (average number of rooms per dwelling), *AGE* (proportion of owner-occupied units built prior to 1940), *DIS* (weighted distances to five Boston employment centres), *RAD* (index of accessibility to radial highways), *TAX* (full-value property-tax rate per $10,000), *PTRATIO* (pupil-teacher ratio by town), B (calularion proportional to blacks by town), *LSTAT* (% lower status of the population). In addition, *MEDV* was the price of the house and our target variable.
All features were numerical and kept with no pre-processing for the model. In total, 506 samples were used.

Modeling, Validation

Our model was based on training the well known model XGBRegressor. We had to train it twice (per dataset), first time to imitate the baseline results that our clients previously had. Second time was after applying the pipeline we created. Both datasets were split into train (85% of the samples) and test sets. We used RMSE as a measure of the performance of the models as by measuring the change in error of predictions was very suitable to clients demands and needs. The proposed pipeline intends the RMSE value from the baseline model.

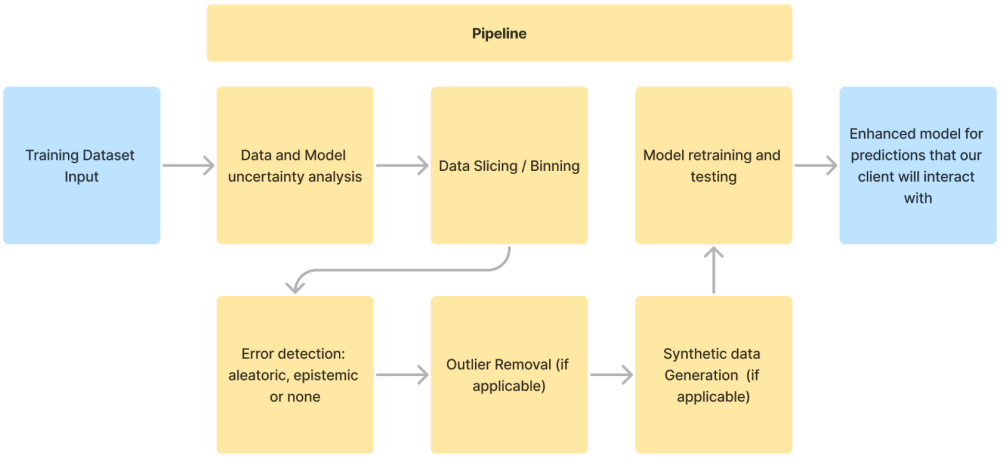
	French Motor Claims	Boston House Prices
Baseline	5.59	3.48
Model after pipeline	5.59	3.27
Improvement	0%	5.93%

Solution Architecture

The solution is a automatic stepped pipeline that consists of the following simple phases:

- Estimate confidence of the model and its predictions. We used MACEst for this step.
- Slice the data and extract the bins with high uncertainty

- Deal with the uncertainty in each bin with different techniques according to the type of error (aleatoric or epistemic). Main alternatives were data generation (using SDV library) or outlier removal based on PyOD library.
- Re-assemble the dataset with modified slices and train XGBoost Regressor.



Benefits

Company Benefit: thanks to our customers who acted as design partners, our company was able to build an internal tool to easily serve regression tasks. Also, it was our first project that enabled us to create and empower our Data Science team.
Customer Benefit:

- Real Estate Startup got a better model that helped them price better (by 6%) the properties they found. They are much more accurate and can strengthen their expansion strategy. They will be able to reduce part of the revenue they were not catching by underpricing houses, they are now able to loose less customers (against competitors) due to overpricing cases and continue to build their reputation in the market.
- French Insurance Company: the client did not get their expected results by improving the model. However, we were able to share with them insights on the data they had and potential weaknesses on the imbalance of their dataset. This will serve their team to continue working on the model and iterating into a more accurate solution.

Learnings

Data Science / Engineering: the main takeaways were related to learning about uncertainty in data. It was new to the team on taking a project with focus on “what the model does not know”. Being uncertainty centric opens a total alternative path to tackle solutions. Also, we specifically learned how to detect and deal with aleatoric and epistemic uncertainty separately.
Domain: the most relevant learning on this aspect was that certain specific features that might seem irrelevant or even not related, they take more sense once you

understand the business context. As an example, at a glance, it does not seem that the density of the town where a driver lives might affect the number of claims he does, however, it does take an important role. With this, we learned how relevant are the business meetings to discuss with the customer not only their needs but also their data, their own learnings and challenges.
Product: we worked with several libraries during the process and here some key points:

- MACEst is a great tool but very new and not extremely clear on how to use it in an extremely fit way, specially for regression tasks.
- We explored CQR (Conformalized Quantile Regression) as a substitute. It is a very useful package. However, it is not efficient and we could not run it with the complete french motor dataset. If we could, we would have chosen it instead of macest.
- Generating tabular data is tough, specially when dealing with small amounts of data (a slice with epistemic uncertainty). SMOGN was not stable, but SDV yielded great results.

Challenges: dealing with large datasets and unknown solutions is risky, it is important to manage time and expectations with internal and external stakeholders.

Links

- Main packages and libraries with its papers were referenced in the previous section.
- Link to the code of our pipeline general solution and the examples/results on the mentioned datasets.
- Link to the presentation of the project and its results.

Next Steps

We consider that there are several steps to be performed on the pipeline to continue improving it:

- Re-apply the pipeline again to the same data as by removing outliers or generating data we can be introducing new uncertainty that we ourselves can deal with it.
- Dedicate more efforts on CQR in order to replace macest. It looks promising for imbalance and very bounded datasets.
- Improve the data slicing technique by clustering, supervised binning or even add other features value to the binning decisions.
- Test the pipeline in other datasets to continue validating its performance and usability.