Master of Science in Informatics at Grenoble
Master Informatique
Specialization Data Science

Plastic Parts Cost Prediction

Masoud Akhgar
Masters professional project performed at Schneider Electric

Supervised By
Slimane Merabti

Defended before a jury composed of:
Renaud Blanch
Emilie Devijver
Massih-Reza Amini

September, 2023

Abstract

This paper is trying to help to develop a tool to estimate the cost of the plastics as a regression problem. The estimation of the part's cost is currently done by the industrialization experts. With a detailed breakdown, in order to support the design to cost method with this project we want to use ML-oriented methods to estimate the cost of the parts based on the data records in hands to allow the non expert like a Designer, Cost Officer to estimate the cost during earlier phase of the project. In this research, we provide a machine learning-based methodology for accurate and efficient price forecasting. In order to extract features from the past produced data, feature engineering is first carried out. Furthermore, we made use of these attributes to predict the price as the target variable using eXtreme Gradient Boosting (XGBoost), after justifying and comparing it to the other regression methods. Finally, we tested also using artificial neural structures and the combination of both. In the context of price estimate scoring, different methods such as ensemble models, regression models and neural networks were experimented. Depending on the location of data, we have different error rates for countries, reaching the required error rate in one case, according to the lack of number of data which is the main problem of the whole project.

Cet article tente de contribuer au développement d'un outil permettant d'estimer le coût des plastiques en tant que problème de régression. L'estimation du coût de la pièce est actuellement réalisée par les experts en industrialisation. Avec une ventilation détaillée, afin de prendre en charge la méthode de conception au coût avec ce projet, nous souhaitons utiliser des méthodes orientées ML pour estimer le coût des pièces en fonction des enregistrements de données en main afin de permettre au non expert comme un concepteur, un responsable des coûts pour estimer le coût au cours de la phase antérieure du projet. Dans cette recherche, nous proposons une méthodologie basée sur l'apprentissage automatique pour des prévisions de prix précises et efficaces. Afin d'extraire des fonctionnalités des données produites antérieurement, une ingénierie des fonctionnalités est d'abord effectuée. De plus, nous avons utilisé ces attributs pour prédire le prix comme variable cible à l'aide d'eXtreme Gradient Boosting (XGBoost), après l'avoir justifié et comparé aux autres méthodes de régression. Enfin, nous avons également testé l'utilisation de structures neuronales artificielles et la combinaison des deux. Dans le cadre de la notation des estimations de prix, différentes méthodes telles que les modèles d'ensemble, les modèles de régression et les réseaux de neurones ont été expérimentées. En fonction de l'emplacement des données, nous avons des taux d'erreur différents pour les pays, atteignant le taux d'erreur requis dans un cas, en fonction du manque de données, qui constitue le principal problème de l'ensemble du projet.

Table of Contents

Chapter 1

Introduction

Schneider Electric is an international company, in North America, Western Europe, Asia, India and China, that provides energy solutions and digital automation for energy efficiency and sustainable development. Thanks to its technologies for energy management, real-time automation, software and services, Schneider-Electric offers integrated solutions for residential housing, tertiary buildings, data centers, infrastructures and industries. Schneider makes processes and energy safe and reliable, efficient and sustainable, open and connected.

Schneider-Electric is organized into two business units:

- Energy management
- Industrial automation

Schneider-Electric is present in more than 115 countries with more than 135,000 employees.

Our presence is balanced across four geographic regions and four diverse end markets, allowing us to seize growth opportunities where they are and balance the volatility of the environment.

Schneider Electric has built a global leadership position in each of its activities, and is leading the digital transformation of Energy Management and Industrial Automation for Digital Solutions with greater sustainability and efficiency.

1.1 Description of Project

This project consists of finding a regression model that allows us to obtain the manufacturing cost of a new plastic part using the simple visible parameters (directly taken from the plastic part sketch and quantity project data, etc.). This cost obtained must be close to reality in order to help estimate the investment envelope and the cost of the products.

At SE, the industrialization department must quantify the best technical and economic solution for projects. A profitability study is carried out before launching a project in order to acknowledge the designer and reduce the cost of products. The estimation of the cost of the part is done analytically with a breakdown, which requires a good knowledge of the injection process. This calculation takes engineers time, hence the idea in these upstream phases of the project to create a model that can be used by non-specialists (the product cost department).

In summary, the benefits of the project can be mentioned as follow:

- Save time of parts industrialization expert

- allow the design and the cost officer to make a simulation of plastic part cost without asking to industrialization experts

- Save time and support the agile transformation

- Support the design to cost Approach

We aim to employ ML-oriented approaches to estimate the cost of the parts based on the data records in hand in order to assist the design to cost process with this project. This will enable non-experts like a designer or cost officer to estimate the cost during an earlier stage of the project. Injection molding is the process of forcing heated thermoplastics into a mold cavity to create a desired component. In the terms of cost, to produce a special type of plastics part, there are the factors like number of cavities and the cost of production of the part that would change over the year, concerning price changes and the depreciation effect on the tooling injection machine.

having a deeper look at parts costs per injection, we would have:

- The Cost of the mold affected by depreciation: It would be calculated by the cost of the injection machine and the tool life-span.

- section Material cost: the most obvious and simple part, which can be calculated by the weight of the part and the price of the material per kilogram. When considering

the cost of a raw material, attention should be paid to the density of the material. For materials with the same unit price, the lower the density, the lighter the product is and the lower the cost.

- Added value: the cost of labor and machinery during part forming, which depends on the cycle time, the type of press, the hourly rate and the location of production. The shorter the cycle, the lower the energy consumption and the costs are. For example, the addition of a nucleating agent can reduce the molding cycle.

- Transport and logistics: the cost of the different phases of packaging, transport from the place of production to the place of assembly and the different possible taxes. The final desired cost of plastics parts to estimate is a price that all these factors containing three mentioned costs, after process cost and the share of tool cost that does the injection process.

The project was carried out on the "Electropole" site based in Eybens, Schneider Electric's first international research and development center to ensure the company its position as world leader. All of my work for this internship took place in the molded parts industrialization department, supervised by Mr. MERABTI from February 15 to August 15, 2023. The project was launched and statically studied by analytic about part's attributes and the possibilities of the cost prediction in 2010 by Mr. Tianjia Wan. Then, it was studied and coded by Mrs. Katerina Vinciguerraand in 2020, studying about the best model to use, first-level feature selection and platform preparation. I worked independently, we had a weekly meeting to discuss progress and the directions to take to solve the problems that I met during this project. I also had a lot of exchanges with Mrs. Vinciguerraand which allowed me to improve my work. The input data was obtained from several manufacturers, each of which is responsible for part of the parts. The analysis approach was advised to me by Mr. Merabti who followed the previous interaction between trainees and specialists working on this subject.

1.2 Project challenges/constraints

- Low number of data to reach the accuracy of 15% in the system of Mean Absolute Percentage Error (MAPE).

- The precision of the calculated input data, which is made by different manufacturers, causing a small variability from one manufacturer to another.

- Price inflation during last years that makes it hard to estimate without last years data.

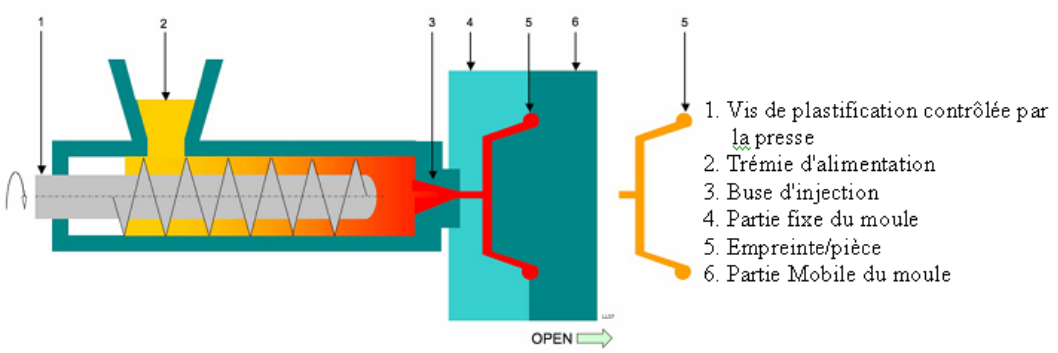

FIGURE 1.1: The injection process using injection machine

- We did not have any access to experiments recorded by Katerina Vinciguerra as the last data scientist intern working on this subject.

1.3 Definition of parameters

The injection of polymers makes it possible to obtain, in a single operation, finished parts of complex shapes in a weight range from a few grams to several kilograms. The plastic parts that make up a contactor or a circuit breaker (Schneider Electric's main products) are generally small and complex in shape. So the injection is a very appropriate technology for this type of manufacturing. Generally, the cost of manufacturing includes the price of the part and the depreciation of the mold, it means the final price that we are going to predict includes an unknown percentage of the compressor tool with unknown price for specific plastics part. The part cost also includes hidden cost as mentioned below. An injection cycle is mainly made up of four steps: mold closing, filling, pressure maintenance, cooling, mold opening and ejection. Of these different factors the most is the cooling time which is related to the geometry of the part, and the material used.

The structure of this paper is as follows:

Chapters 2 named related tasks describing the papers related to the whole project and the methods employed addressing the upcoming challenges and novel prediction approaches. Chapter 3 contains Information on the data preprocessing and data visualization using Tableau software. We would explain the reason behind proposing any algorithm and the advantages and drawbacks of each method in chapter 4, reporting and comparing methods in chapter 5.

Chapter 2

Related Tasks

To propose novel architectures according to previous observations on the project, more than 20 related papers studied, so some of which can be observed in this part.

According to the later explanation, the first chosen model to solve the non linear regression problem was XGBoost. [1] which studies predicting gene expression values, deduces that the D-GEX method is outperformed by the XGBoost-based gene expression value prediction algorithm, which also performs better than more established machine learning algorithms like Linear Regression and KNN.

Although XGBoost perfectly supports non-linearity, it tends to be overfitted, and there are many parameters to be tunned that affect directly its performance. [2] aimes to solve the non-linearity regression using neural network which also solves the imbalance data problem that appeared in our dataset. It presents a simple MLP architecture to solve linear and nonlinear modeling, using Receiver Operating Characteristic curves (ROC) and the Area Under each ROC Curve (AUC) as an indicator in the classification and prediction task.

In a real-world social media popularity prediction, a CNN model is exploited to learn high-level representations from the social cues of the data. These high-level representations are used in XGBoost to predict the popularity of the social posts [3]. The idea of using a hybrid model also helps when we would like to generate more features alongside using neural network architecture, tackling the non-linearity problem.

The data frequently shows skewed distributions with a lengthy tail rather than the desired uniform distribution over each category. Considering data imbalance problem, [4] uses label distribution smoothing (LDS) estimating the effective label density distribution by convolving a symmetric kernel with the empirical label density.

Algorithm 1: Training

Input: Training data $D = \{(x_1, l_1), (x_2, l_2), ..., (x_T, l_T)\}$.
Episode number K.
Initialize experience replay memory M
Randomly initialize parameters θ
Initialize simulation environments ε
for *episode* $k = 1$ *to* K **do**
  Shuffle the training data D
  Initialize state $s_1 = x_1$
  for $t = 1$ *to* T **do**
    Choose an action based ϵ-greedy policy:
    $a_t = \pi_\theta(s_t)$
    $r_t, terminal_t = STEP(a_t, l_t)$
    Set $s_{t+1} = x_{t+1}$
    Store $(s_t, a_t, r_t, s_{t+1}, terminal_t)$ to M
    Randomly sample $(s_j, a_j, r_j, s_{j+1}, terminal_j)$ from M
    Set $y_j =$
    $\begin{cases} r_j, & terminal_j\text{=True} \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta), & terminal_j\text{=False} \end{cases}$
    Perform a gradient descent step on $L(\theta)$ w.r.t. θ:
    $L(\theta) = (y_j - Q(s_j, a_j; \theta))^2$
    if $terminal_t$=*True* **then**
      break

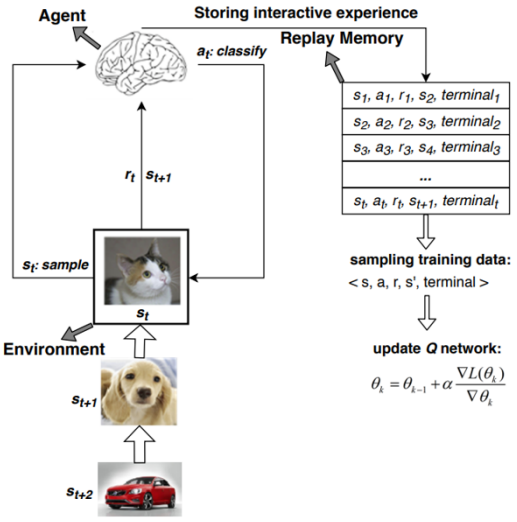

Fig. 1. Overall process of ICMDP.

Algorithm and schematic of Markov decision process

In addition, the above method that was studied used RL to find out the best weight given to data group as sample weights. In the paper[5], as a schematic is shown, known as Markov Decision Process, a RL approach has been used while it is considering an image dataset and it classified it into three classes. To find the optimized weights for the imbalanced dataset, it used a RL approach in which a positive reward is given to the agent by the environment when the agent correctly guesses the category of the sample considered as a state, otherwise a negative reward is given to the agent. As it is mentioned, each image, the action of the agent is picking up a category for that state, labeling the data. Labeling the data might be seen as assigning weights from a certain array to each sample, tackling with data imbalance problem.

In [6] a method for weighting samples is proposed. The goal is to weight individual data points based on the rarity of their target values. Thus, it wants to calculate a weight for each sample inversely proportional to the probability of the target value's occurrence, based directly on the target distribution's density function. It uses the Guassion function as the kernel function to define a density function, named DenseWeight, and processing a cost-sensitive learning approach using MLP defined to obtain a weight for each sample.

Chapter 3

Dataset and Pre-processing

3.1 analysis data

As it is mentioned, **Tableau** is mostly used for the illustration and comparisons and visualization of data. We have a highly distributed data in the database of project, in the terms of all feature parts and the target price. As it is shown in the chart below, mostly the features samples are following a logarithmic pattern, so all models are tested on the logarithm of data, although we would analyze all transformation functions to compare different effects on the final result.

TABLE 3.1: Number of records per country in the dataset.

Country	China	Europe	Asia	Mexico	India
Number of records	120	109	37	31	91

As it is obvious, data augmentation can be regarded as an appropriate technique to be used. Data augmentation is a technique of artificially increasing the training set by creating modified copies of a dataset using existing data. As [7] used augmented SGD with additive noise to measure effect of this Optimization for Linear Regression. However it is not appliable to our case because according to the few number of data, the pattern induced by augmented data can manipulate the real existing pattern.

In the chart below related to China dataset, we can compare how much the distribution of target prices is different when using a logaritmic transformation function or not using that. The vertical axis is the real value of price while the horizental axis is the price after transforming into the logarithm space.

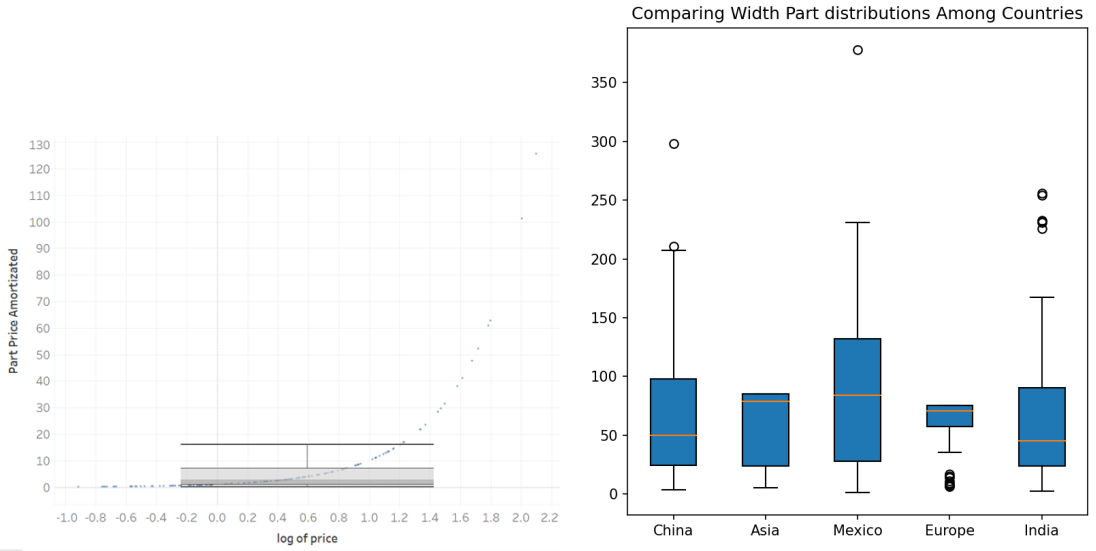

FIGURE 3.1: the importance of using logarithm transformation to have a homogeneous distribution

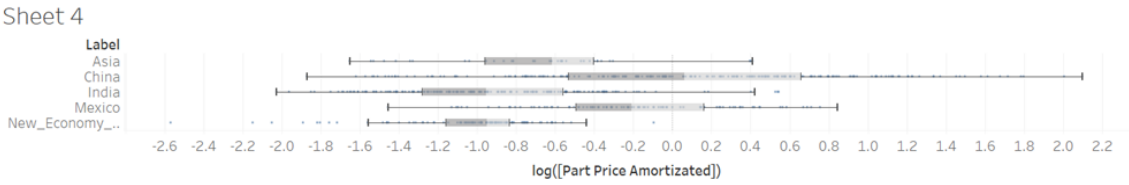

FIGURE 3.2: comparing the range of price with the range of width for parts, revising different ranges depend on the country, showing that we need to break database down into separated countries

To compare the prices of all countries, the chart below is illustrated comparing the prices of all countries while the next picture shows the distribution of "Width" of parts among all countries. We chose width of parts to plot while this feature has the most variance after quantity and weight feature.

From pictures above, we can perceive that although the range of all features is the same, we have completely different prices in all countries that proves we probably must develop an independent model for each country.

3.2 Feature selection

3.2.1 correlation

After study performed to find out the importance of feature selections and dimensionality reduction in [8] for a complex disease risk prediction, we understand the importance of

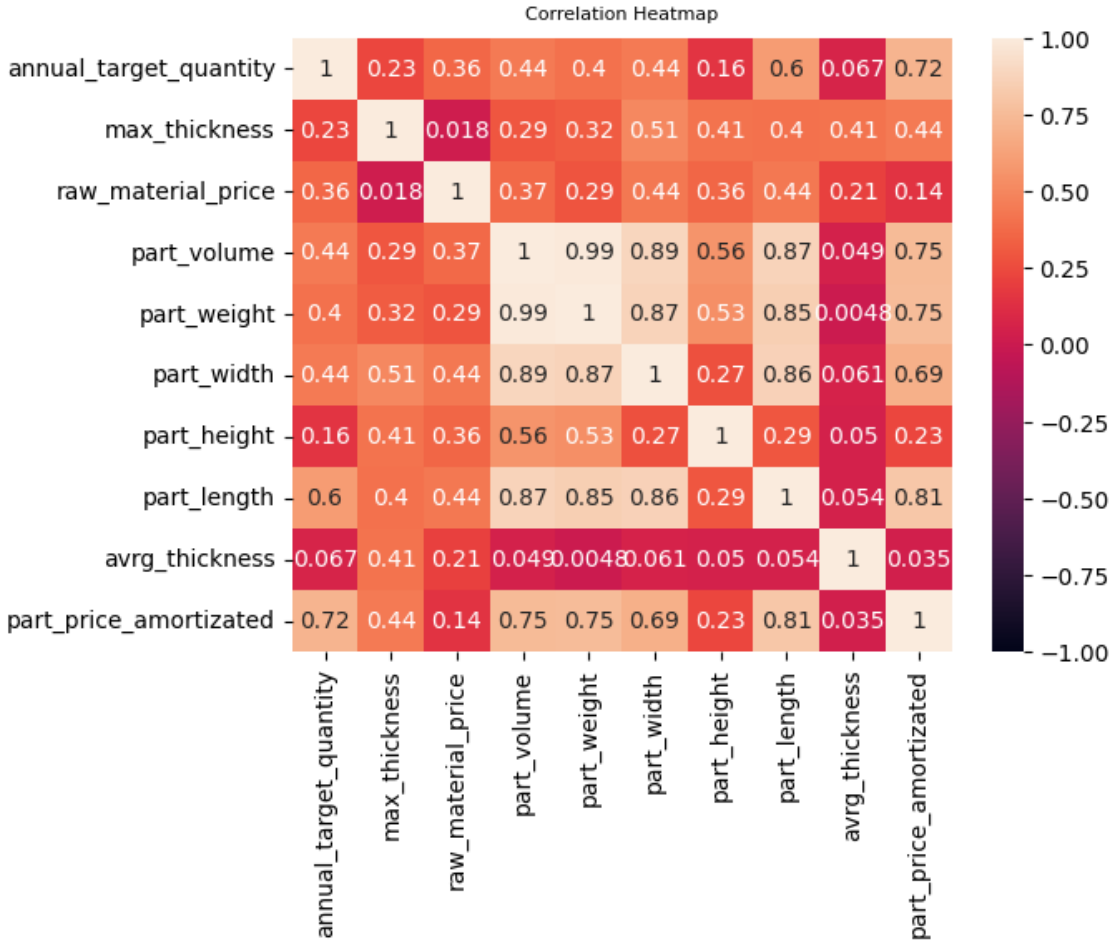

FIGURE 3.3: Heatmap of the feature correlation for Europe dataset

removing features having highly correlated in machine learning considering redundant. This part is done by statics student worked on this project, so finally we would test on each country for occasionally correlated features, but we can create features with high correlation with the target price. This means having a high correlation with the target is not always ideal, since to estimate accurately, the model needs to have many features with different correlation with the target and we must be sure that this correlation is not due to data leakage.

In the figure above, considering correlation threshold, if we select 0.85 as the threshold, we would eliminate two of width, length, weight and volume, while it would be different for higher threshold. As a result, correlation function is considered as an option in the tunning phase.

Also, we can remove the average thickness feature, observing the low possibility of contributing of this feature on the final target price. Although the correlation between

the target and raw material price seems low, and occasionally quantity feature, we won't remove these features as we know they have a direct effect on the price.

3.2.2 P Value

Null hypothesis is a general statement that there is no relationship between two measured phenomena. p-value is a probability value for a given statistical model that, if the null hypothesis is true. Removal of different features from the dataset will have different effects on the p-value for the dataset. We can remove different features and measure the p-value in each case. These measured p-values can be used to decide whether to keep a feature or not.

3.2.3 Feature creation

Here it is decided to try some simple and mathematical combinations of basic features to create new features. After testing features created, we have reached to a feature that improves noticeably the accuracy on almost all countries, significantly on China dataset. I naming this feature as 'QRV', the formula deduced is brought below:

$$QRV = \frac{log(log(quantity))}{raw\_material\_price * volume\_of\_part}$$

3.2.4 Data transformation

Data transformation is a critical preprocessing technique used in data analysis and machine learning to modify the scale or distribution of the data. The purpose of data transformation is to improve the performance of statistical and machine learning models, address issues like non-linearity, and handle data that does not meet the assumptions of certain algorithms. Logarithmic transformation is particularly useful when dealing with data that exhibits a skewed distribution, such as exponential growth or decay. It can compress large values and spread out small values, helping to normalize the distribution and reduce the influence of extreme values. As we have such skewed data, this is the best transformation to be used on data. It's essential to be cautious when applying logarithmic transformations to data that contains zero or negative values, as the logarithm of these values is undefined. In such cases, a data shift or adding a constant can help make the data suitable for transformation.

Categorical feature encoding - after investigating the results, we understood that a part of price changes is because of the **color** feature that surprisingly affects the price. For example, in the ANN model, it improves the result in the China dataset just by adding

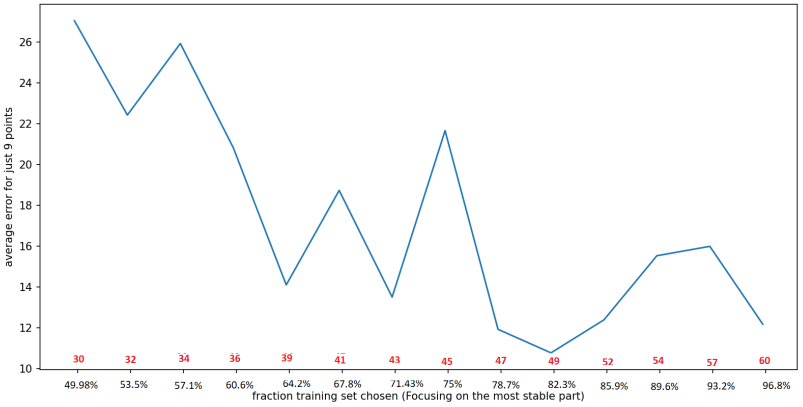

FIGURE 3.4: the MAPE percentage of predicted values Xgb algorithm would decrease by raising the number of data in trainset, predicting the number of homogeneous data needed for the desired result

color into the calculations. Mean Target Encoding (also known as Target Encoding or Probability Encoding) is a feature engineering technique, calculating the mean of the target variable for each unique category in the categorical feature, replacing the original categorical values with the corresponding mean target value. As it has been seen, this method of encoding has a higher impact on the result rather than hot-vector encoding when we have many possibilities for color feature.

3.3 The Lack of data

To estimate the number of data points required to have an error less than 15%, the concept of statistical power analysis can be used. By doing the following steps, we can estimate that how many more data we need to reach our ideal accuracy.

- Spliting the dataset into training and testing data, and using the training data to train the xgboost regressor model.

- Evaluate the performance of the model on the testing data.

- Vary the sample size by randomly selecting a subset of the original dataset (e.g., starting with 10% of the data and increasing to 100%). For each sample size, we repeat step 1 and step 2 to evaluate the performance of the model.

To see the result we plot the relationship between the sample size and the error rate on a graph Fig 3.4.

This strategy can also be used to find out if having more data solves and optimize a solution or not.

3.4 Imbalanced Regression

At the first, having a huge gap among the error percentages after prediction gives this message that the model is actually overfitted or there are several noises in the dataset, leading to appearing high errors but low in numbers. as an acknowledge to this, we could use either the box plot noise reduction method or to generalize the model to improve the result.

The first approach enhanced the model and it is approximately justified since we have a few data that affect badly the whole prediction system, so they are considered as noises knowing that they are reported from the same source. However, the problem of mentioned approach is that we must develop the most generalized model that not only predicts the data being in the dataset, but also the different patterned data which we have less from.

The next approach was the usage of hyper parameters to make the model more smooth on our dataset, and it seems crucial since the Xgboost tends to be overefitted. It sounds confusing when we had less accuracy following less fitted model on the dataset.

According to the alternative mentioned approaches problems, we realized the imbalance solutions should be the ones to be followed.

Data imbalance is ubiquitous and inherent in the real world. Rather than preserving an ideal uniform distribution over each category, the data often exhibit skewed distributions with a long tail. [4] investigates Deep Imbalanced Regression (DIR) arising in real-world settings, dividing imbalance regression solutions into two categories:

- Data-based solutions either over-sample the minority class or under-sample the majority[9]. Using this paper, an approach known as SMOGN is implemented that is used in our experiements to compare with other methods proposed.

- Model-based solutions include re-weighting or adjusting the loss function to compensate for class imbalance [10]. Instead, I inspired from the paper, using the absolute error as the weight for each data, assigning as sample weights in the Xgb algorithm known as Weighted-XGB.

In some classification tasks, the target classes are imbalanced, meaning that one class is significantly more common than the others. In these cases, the default loss functions in scikit-learn (such as $accuracy\_score$ or $f1\_score$) may not be the most appropriate measure of performance. For example, if a classification task has 90% negative examples and 10% positive examples, a model that always predicts negative would achieve an accuracy of 90%. In this case, a custom loss function that gives more weight to the

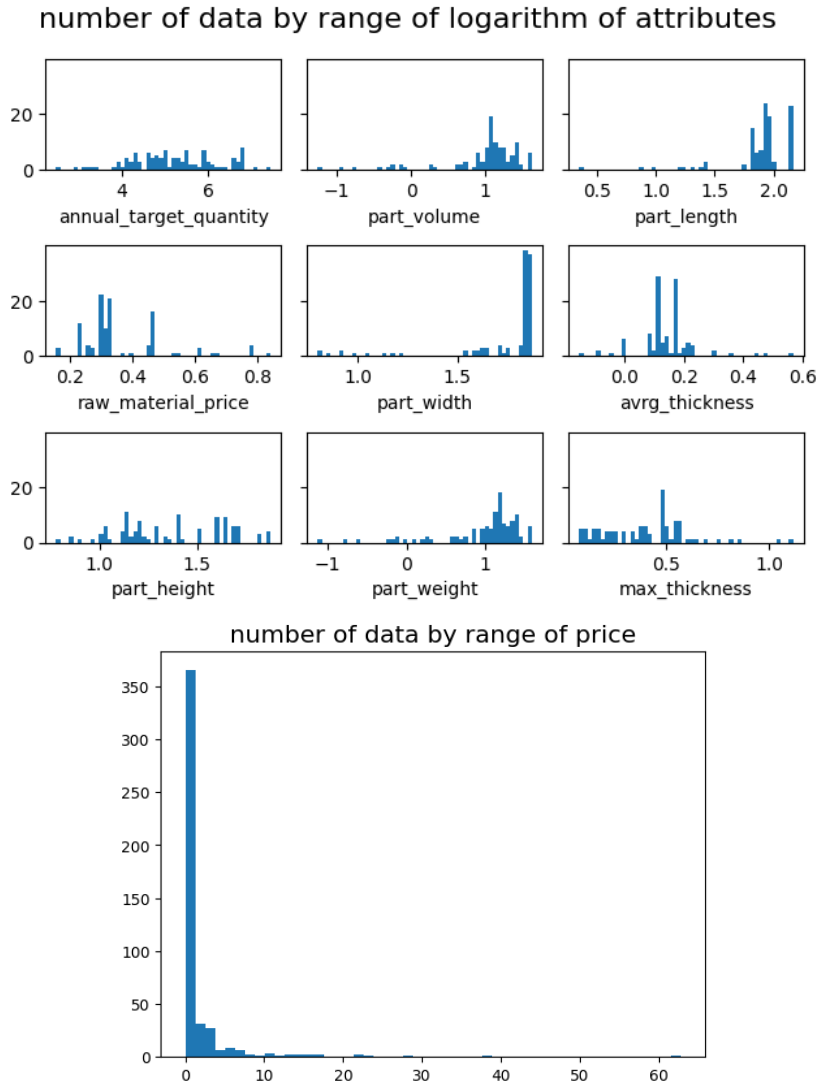

FIGURE 3.5: analyzing the multi classes data having different grouped distribution

minority class might be a better measure of performance. Because of this reason, the inspired XGB-weighted was proposed by giving higher priority to data having higher error.

As it is presented in 3.5, we have a big gap in the number of data considering the range price. This imbalance data problem also appears among features distribution. This will give us a visual representation of how the values are distributed across different ranges. Because the histogram is skewed or concentrated in a particular range, it could indicate an imbalance.

3.5 Error definition

In the Xgboost algorithm, we used mean absolute percentage error to track and measure the performance and in ANN to optimize on trainset.

The significant point about choosing between MAPE and MSE depends on the specific context of your problem and your priorities regarding error measurement. MAPE might be preferred when you want to focus on percentage accuracy and relative performance, while MSE might be more suitable when you want to minimize squared errors and prioritize accurate predictions overall.

Lets define the error function such that:

$$E_{mean\text{-}absolute\text{-}percentage\text{-}error} = \sum \frac{(y_{predicted} - y)}{y}$$

When using MSE, it was observed that the error percentage is much less than we calculate MAPE. In general, the acceptable range for MAPE according to [11], is less than 25%.

[2] compares a linear and non-linearity MLP model by adding multi-layer perceptrons, and it is measuring the performance of two proposed model by metrics like R-square and std error for the classification task. Inpiring [6], presenting our results by other metrics might come close to understanding of how much the models coded are good enough. R-squared values range from 0 to 1 and are commonly stated as percentages from 0% to 100%. In social science research, an R-squared between 0.10 and 0.50 (or between 10% and 50% when expressed in percentage) is only acceptable when some or most of the explanatory variables are statistically significant. R-squared is most frequently used to determine how well a regression model explains observed data. For instance, an r-squared of 20% indicates that the regression model accounts for 20% of the variability seen in the target variable.

3.6 Noises

According to the box plots 3.2, we can understand that probably the data is so much distributed with high variance. For example, we can see a figure around 53 as the variance of "width" of parts that is relatively high while the maximum width is 86 among 120 records. Therefore, we would have two approaches to tackle the problem.

- We can consider the data outside of the box as noises, not because they are wrong data, just because they are few in numbers so the model won't be generalized in this situation and they affect the results for the data that we have them a lot in certain

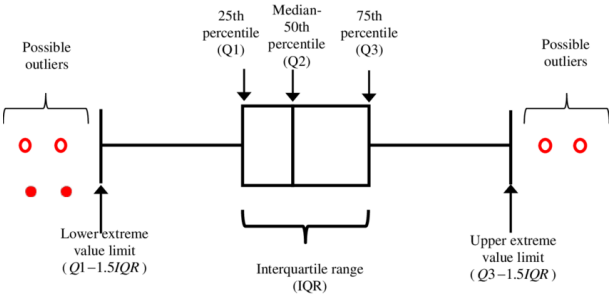

FIGURE 3.6: removing noises after a certain range

ranges. We used this approach at the first months as we see using this approach, we would have an improvement on the error, however, as it would be mentioned in following discussions, I conclude that the second approach also can be used in which we don't need to delete data, considering lack of data that we encounter with, and actually having better performance for the future parts. Of course that we need to select a threshold to remove data coming farer from a certain range from most other data, considering them as noises.

For example, in Picture 3.6, we would consider points in the red square as noises since they are outliers. We would do the same thing for each feature and the price.

- We can decide to give weight to each sample. In this part we tried to increase the weight of outlying data, and at the other side, when we decided to decrease their weights it means that we consider them as noises, not contributing in the trainset.

3.7 Train Test division

Since our dataset is very small, we have decided to consider using cross-validation instead of a single train/test split, except the neural network model. Cross-validation involves dividing the dataset into multiple subsets (folds) and then training and testing your model on different combinations of these subsets. For instance, in a 5-fold cross-validation, we would divide our dataset into 5 subsets and then perform training and testing 5 times, using a different subset as the test set in each iteration while using the others for training. When selecting the proportion for dividing our database into training and test sets using cross-validation, we're essentially deciding how many folds (subsets) our data should be split into. Cross-validation aims to assess our model's performance more robustly by repeatedly training and testing on different subsets of data.

To do this, we run the cross-validation iteratively on the dataset which is k-folded. The average error of the outcome is plotted in the chart. We ran it simply having the XGB

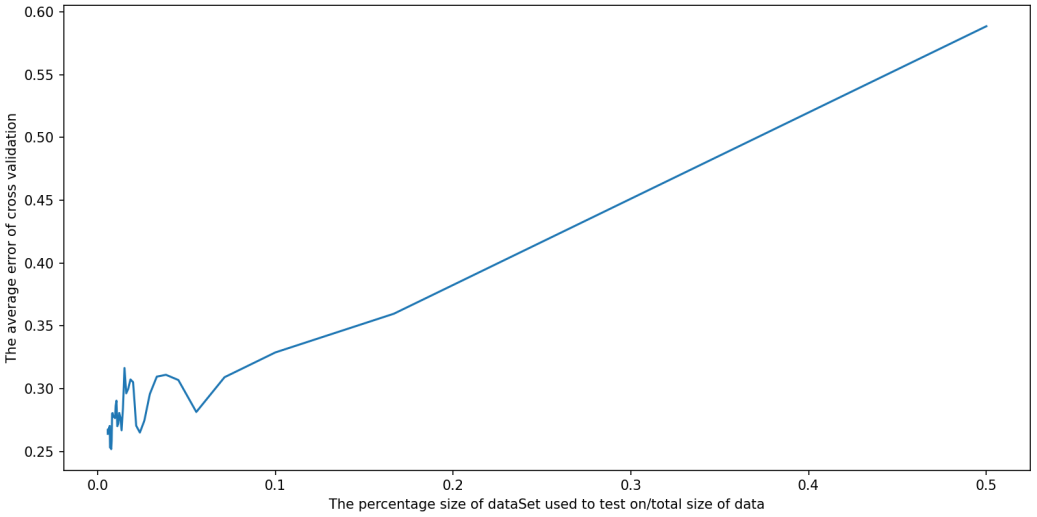

FIGURE 3.7: Cross-validation Error Dependence on Train/Test Proportion

model, and as it is visible there is a linear ascending trend that shows the effect of noises/non-homogeneous records won't affect on the final result.

Considering the figure above, we considered 10 percentage as a figure that the dataset would be divided into.

Chapter 4

Model proposition

4.1 Model proposition

4.1.1 linear regression

The first candidate to be used as a model is Linear regression. After a meeting with Slimane Merabti, the final price of part production comes from three sets about the part: 1) raw material price used 2) cycle time that relates to dimensions 3) tool cost effect that relates to quantity. It means that the final production cost is the linear combination of these parts that any of which is a linear combination of part features, so linear regression can be a good candidate although the target might relate to a combination of part attributes so that we cannot expect a perfect accuracy there. In addition, since an unbalanced data is distributed, it might not be a desired data distribution for the linear regression.

Decision trees also are a perfect option for business problems. One of most advantages that is a key for our prediction problem, is supporting non-linearity. We can find our problem is more complex than a linear regression as we know the process. However, DT needs a lot of features to predict perfectly while we don't have relatively high number of features. Although I will prove that non-categorical features don't matter in our solution, DT works better when having categorical features. Because of the reasons above, I search through most famous DT algorithms.

4.1.2 XGBoost

Therefore, I decided to choose a model that combines regression with trees, so I chose the XGBoost algorithm, and also some other algorithms to compare the results with them

as competitors. The XGBoost algorithm is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment. It is applicable in many health-related issues and it is known as a fast prediction method. It is used in [12] to designed an algorithm for predicting gene expression values.

An overview about parameters:

1. *n_estimators*: *n_estimators* is the number of iterations in training. A too small *n_estimators* can lead to under-fitting, which makes the model not fully perform its learning ability. However, a too large *n_estimators* is usually not good either, because it will cause over-fitting.

2. *min_child_weight*: As we mentioned earlier, *min_child_weight* defines the sum of sample weight of the smallest leaf nodes to prevent over-fitting.

3. *max_depth*: It is the maximum depth of the tree. The greater the depth of the tree, the more complex the tree model is, and the stronger the fitting ability is, but at the same time, the model is much easier to overfit.

4. *learning_rate*(α): In most algorithms, learning rate is a very important parameter that needs to adjust, as well as in XGBoost. It greatly affects the performance of the model. We can reduce the weight of each step to make the model more robust.

5. *Regularization_parameter*(λ) : Regularization is used to avoid over-fitting on the data. Higher the λ, higher will be regularization and the solution will be highly biased. Lower the λ, solution will be of high variance. An intermediate value is preferable.

Here is a record of some parameters' effect on the error percentage (MAPE). 4.1

As it is visible, the learning rate and the tree booster are the most effective parameters. As we describe later, we use also the *sample_weight* parameter to balance multi-class sizes by applying class weight to each data instance, calculating a weight for each sample. Because XGB tends to overfit on the dataset, we note the amount that the model fit on trainset comparing to the accuracy on the testset.

In the XGB-tunned we tried to select the best parameters except dedicated XGB model hyper parameters in the Sklearn module. we also consider the possible usage of transformation functions, feature engineering functions, booster learner types- gbtree, gblinear-

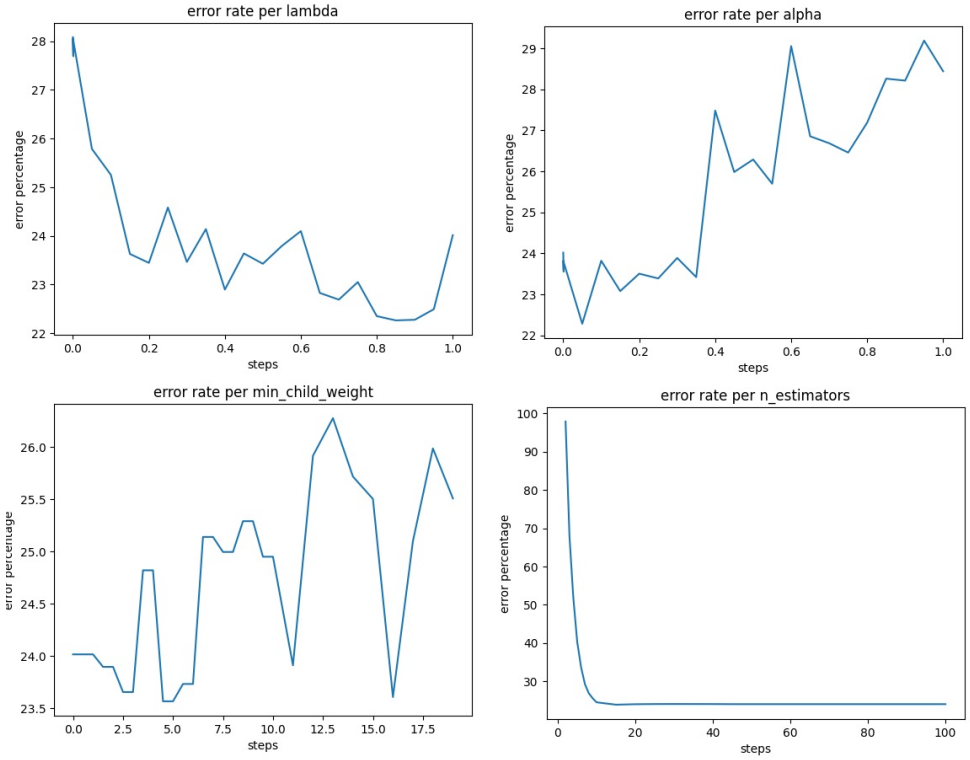

FIGURE 4.1: the effect of hyperparameters in XGBoost algorithm

and possible usage of categorical encoded features and QRV feature for specific countries. One another idea was using a combination of more than two datasets having similar range of price to tackle the lack of data problem.

4.1.3 XGB-weighted

Inspiring [6] that proposed a density function using kernel function, replaced by Guassion distribution, and bandwidth meaning that a predicted value with low error multiplies by Guassion distribution, optimizing weights by MLP, we propose a simple iterative learning process.

Weighting samples according to their error range is a technique used in regression models to improve the model's accuracy and predictive performance. In this approach, each data point in the training dataset is assigned a weight based on the magnitude of its prediction error. The idea is to give more importance to data points with larger errors, allowing the model to focus on learning from those instances that it struggles to predict accurately. It was proven that most data follows a linearly pattern, so to give a continues weight to samples, we decided to use the absolute error of linear regression model.

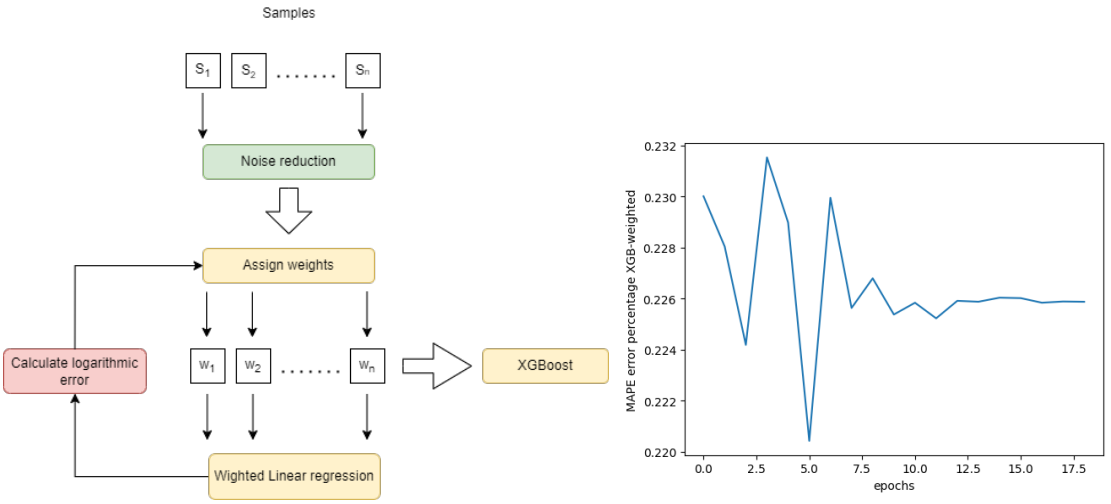

FIGURE 4.2: the schematics of an iterative XGB-weighted model proposed

Here's a detailed description of weighting samples based on their error range in the regression model:

- Error Range Calculation: To implement sample weighting, the first step involves training the linear regression model on the given dataset and obtaining predictions for each data point. The error range of each sample is then calculated by comparing its predicted value with the actual ground truth value.

- Assigning Weights: Once the error ranges are computed, the next step is to assign weights to each data point based on its error magnitude. Samples with larger errors are assigned higher weights, indicating that the model should pay more attention to them during subsequent training iterations.

- Model training: As a result, the model is encouraged to focus on reducing errors for those instances that were originally challenging to predict accurately.

- Iterative Process: Weighted sample training can be an iterative process, where the model is trained multiple times, and the weights are updated at each iteration. The point here is that after doing iteration, the error for linear regression would be more, but it improves XGBoost model's performance. we also don't transform back the target variable when calculating the error.

4.1.4 XGB-C

The next model that can be used is a combination of K-means classification and XGBoost algorithm, named C-XGB. In this way, we divide the trainset and testset, using just the one unit class to train or test on. Since we use XGBoost, this approach helps

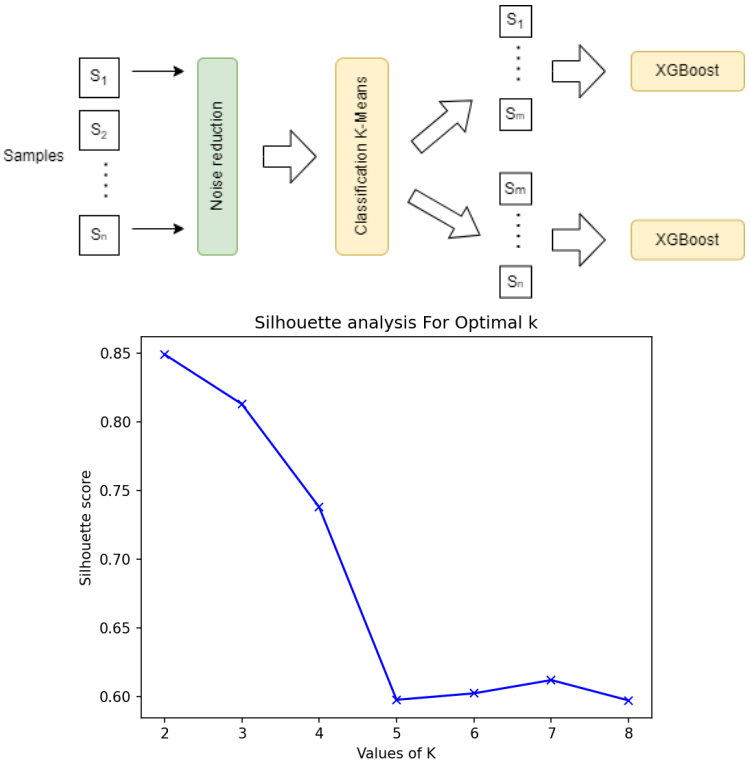

FIGURE 4.3: XGB-C Model architecture proposed

to understand if there are two or more different patterns among the data or not. By classifying data, we would create two XGBoost models and feed them by two different mutually exclusive dataset.

According to the design, we classify the weights into two classes, assigning the same weight to the same-classified data. We decided to break down the dataset into two subsets because of silhouettte experiments that shows the best option is selecting two parts as k in k-means model.

4.1.5 XGB-Smote

The next model that have been tested is a combination of Xgboost algorithm and Smote algorithm [7]. Through the creation of synthetic samples for the minority class, SMOTE helps to address the issue of class imbalance. In order to ensure that the machine learning models are trained on a more representative set of data, this results in a balanced dataset. This leads to significant improvements in performance metrics like precision, recall, F1-score, and area under the ROC curve (AUC-ROC).

Considering sensitivity to Noise, we would see the effect of noises in future parts, in the presence of noisy data points within the minority class, SMOTE may generate synthetic

samples that amplify the impact of these noise points, leading to erroneous predictions and reduced model accuracy.

we should consider that before using Smote algorithm, we should divide our dataset into at least two groups. To do so, we used Linear Regression grouping high error records as a separated second group. For dividing into two groups, we need to distinguish a threshold in which we almost divide dataset into two groups so we increase the number of data as least as possible, avoiding creating a artificial pattern in trainset. According to the process mentioned, this method is not acceptable in the industry at all, and we just consider it as a way of tackling with data imbalance.

4.1.6 Auto-encoder

Auto-encoders are of a type of neural network architecture used for unsupervised learning, particularly in the field of feature learning and dimensionality reduction. The primary purpose of auto-encoders is to reconstruct the input data as closely as possible, acting as self-supervised models. It also would help us in two matters:

- Auto-encoders can learn meaningful representations of the input data, capturing essential features and patterns in an unsupervised manner. These learned representations can be used for various downstream tasks, such as classification and clustering.

- Auto-encoders can be used as denoising models by training them to reconstruct clean data from noisy inputs. This helps improve the model's robustness and generalization to noisy environments.

This idea was proposed to see if there is a non-linearity problem in the data or not. In the case of having a non-linearity in the data, auto-encoders can learn non-linear transformations, allowing them to capture complex patterns and representations in the data. However, using neural networks can have some undesirable effects like overfitting and having high cost in computation.

In below the diagram of the feature learning paradigm, which can be applied to either raw data such as images or text, or to an initial set of features for the data, for the application to downstream tasks is shown. Feature learning is intended to result in faster training or better performance in task-specific settings than if the data was inputted directly. [13]

We have total control over the architecture of the auto-encoder. We can make it very powerful by increasing the number of layers, nodes per layer and most importantly the code size. Increasing these hyper-parameters will let the auto-encoder to learn more

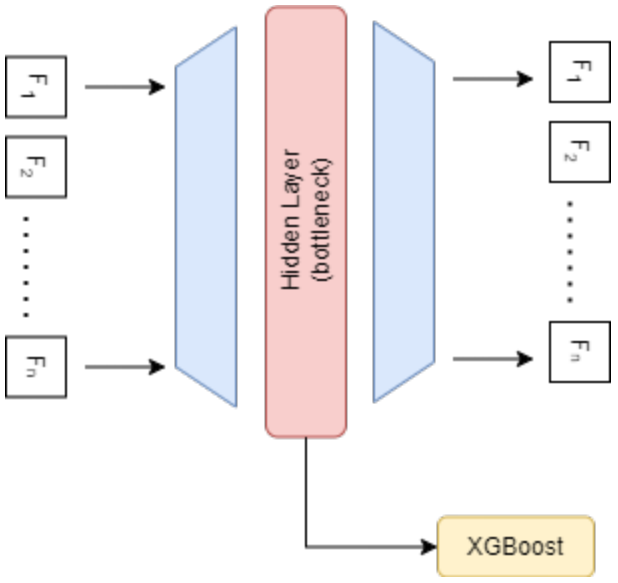

FIGURE 4.4: the schematics of the XGB-autoencoder model proposed

complex codings. But we should be careful to not make it too powerful. Otherwise the auto-encoder will simply learn to copy its inputs to the output, without learning any meaningful representation. It will just mimic the identity function. The auto-encoder will reconstruct the training data perfectly, but it will be overfitting without being able to generalize to new instances, which is not what we want. This is why we prefer a "sandwitch" architecture, and deliberately keep the code size small. Since the coding layer has a lower dimensionality than the input data, the auto-encoder is said to be undercomplete. It won't be able to directly copy its inputs to the output, and will be forced to learn intelligent features. If the input data has a pattern, for example the digit "1" usually contains a somewhat straight line and the digit "0" is circular, it will learn this fact and encode it in a more compact form. If the input data was completely random without any internal correlation or dependency, then an undercomplete auto-encoder won't be able to recover it perfectly. But luckily in the real-world there is a lot of dependency.

At the end we would have a unique representation encoded of our features. As I will discuss about its results, I deduced that we don't have lack of feature numbers and probably the problem is not about non-linearity of data which helps me to reach to the next model.

4.1.7 Artificial Neural Network

The purpose of using Artificial Neural Networks for Regression is learning the complex non-linear relationship and deal with, although we have few data and using neural

network does not sound as being the best solution. Having a better result in the final experiments does mean that we encounter a complex problem rather than a regression problem having the lack of data.

It was tested and reported the error percentage per each layer added to the network. As we increase the number of layer, network would learn more complex patterns, however, it might overfit on the data, so it was attempted to have the minimum number of layers as least as possible. In addition, activation functions are a crucial component of artificial neural networks. They introduce non-linearity to the model, enabling it to learn complex relationships within the data. As we studied, the best activation function for neural network regression is ReLU and linear function for the last layer.

Refering to the optimization method, optimization algorithms are used to adjust the weights and biases of the neural network during training in order to minimize the loss function. Variants of gradient descent, such as Stochastic Gradient Descent (SGD), Adam, and RMSprop, have been developed to improve convergence speed and handling of different types of data.

The batch size refers to the number of training examples utilized in a single iteration of the optimization algorithm. Larger batch sizes can lead to more stable updates, as they average out noise in the gradient estimation, tending more to be overfitted on trainset. An epoch refers to one complete pass through the entire training dataset. Training is typically performed over multiple epochs to allow the model to learn complex patterns in the data.

Table 4.1: Selected hyperparameters in ANN approach

Activation Functions	Loss Function	Optimization Algorithm	Batch Size	Number of Hidden Layers	Epochs
ReLu,linear	MAPE	adam	50	2	300

Table 4.2: Selected hyperparameters in ANN approach

Errors per hidden layers number	0 hidden layers	using batch normalization, ReLU	1 hidden layer	1 hidden layer, batch normalization, ReLU	Epochs
Europe dataset	33 %	31%	31%	22%	50

Chapter 5

Model Comparison

5.1 Model Comparison

In this section, all our experiments are carried out on a computer with a 2.30 GHz CPU, an Intel(R) Core(TM) i5 6200U CPU 2 Cores with 8 GB of RAM. We have implemented the proposed models and other methods in Python.

In the table 5.1, considering tuuned XGB there is a possibility that the model tends to be overfitted on small dataset specially in Asia as we see it presents a better result than others while we encounter a smaller dataset rather than other countries.

Considering XGB-C, Since it improves the error percentage as it is shown in the table, we conclude that we have an imbalanced data problem, it means that if we consider the whole data as N class that follows N different patterns among features, we don't have an equal number of data for each group of data, affecting our model performance. This conclusion can be driven by these observations below that have been seen during the project.

- As it is mentioned, the first reason is the performance improvement on the data assigned to the same groups by unsupervised classifications.

- Appearing high errors after optimizing the XGboost model while most of error ranges are less than 40 percentage. 5.1

- Receiving the same accuracy while using randomForest, XGboost or gradient descent algorithm after optimizing level, so it emphasizes that the problem is not about the model used. After using the feature learning method, we infer the problem of the data can be the high balance disordered.

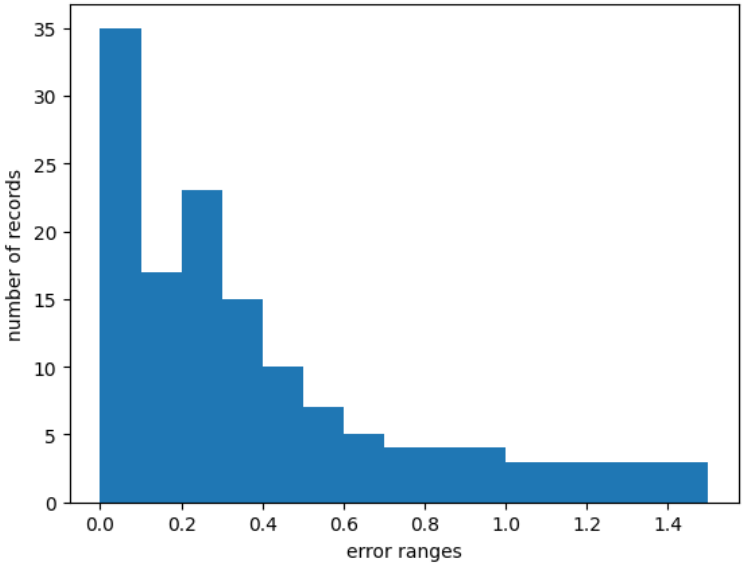

FIGURE 5.1: The histogram of error on China dataset after applying XGB

TABLE 5.1: Results of Model Comparison

Errors per method	Europe	China	India	Mexico	Asia
number of records	109	120	91	31	37
LR	29.5 %	33.4%	35.6%	71%	168%
Gradient Regressor	28.5 %	27.0%	20.3%	46%	88%
XGB	28.0 %	26.8%	22.7%	39.9%	62%
XGB-Smote	20 %	22.15%	20.4%	30%	52%
Tunned XGB	29.18 %	27.84%	19.92%	44.0%	67.9%
DenseWeight-XGB	23.8 %	25.04%	19.12%	36.5%	32.4%
XGB-C	24.6 %	25.7%	23.8%	101%	43.9%
Weighted-XGB	23.4 %	20.7%	19.1%	38.6%	47.5%
auto-encoder	28.3 %	29.5%	22.6%	61.5%	69.3%
NN	18.0%±2.6	16.2%±3.1	13.8%±2.1	62%	33.9%±4.8

In this table, it is sensible to say that there should be other factor(s) affecting the accuracy of the model because there is no strong correlation between number of data

and the performance. Almost the highest errors belong to Asia and China for all implemented models that emphasizes the role of data count on the result. From other aspects, although neural network has the best percentages for the three first countries, those percentages fall in the smaller datasets.

As we have the best results in neural network, it shows that the existing complex pattern and non-linearity is solved by higher number of neurons in neural network. As it is obvious, weighted-XGB has increased the accuracy and as an ensemble model, it might be a opportunity to focus on in the future works.

The auto-encoder coinciding with XGB does not improve the accuracy, and this presents that having more features or added encoded existing features do not give any more information about the records. Because the figures are an average of iterations, in neural network they are mentioned as a range.

After testing on different structures of ANN, the choice of appropriate values for the last two parameters can significantly impact the model's performance and convergence, so we plot the result 5.2. As it is clear, having an epoch less than hundred, the model would converge with batch-size equal with 50. However, even when batch-size equals hundred, it does not seem that the model is converged and also it is not overfitted, knowing that the higher batch-size, the higher possibility of overfitting. Since we have almost hundred data in the database, this number is the maximum elected for batch-size and it is a sign that these models can be optimized by increasing data number in the future.

the results in the table 5.2 are recorded from the Europe dataset.

TABLE 5.2: Results of Model Comparison

Model	MAPE	MAE	R-Square
LR	29.5 %	4.6%	50.9%
xgb	28.0 %	3.8%	42.2%
Tunned xgb	23.8 %	5.0%	38%
Weighted-XGB	23.4 %	4.3%	39.1%

The MAE metric states around 5% while it does not have the correlation with MAPE for Tunned XGB and weighted-XGB. It is justified because probably in these two methods we are minimizing the MAPE error as the goal, however MAE measures the error in total. In addition in R-Square we have observed an acceptable figures in all models as they almost all represent less than 50%.

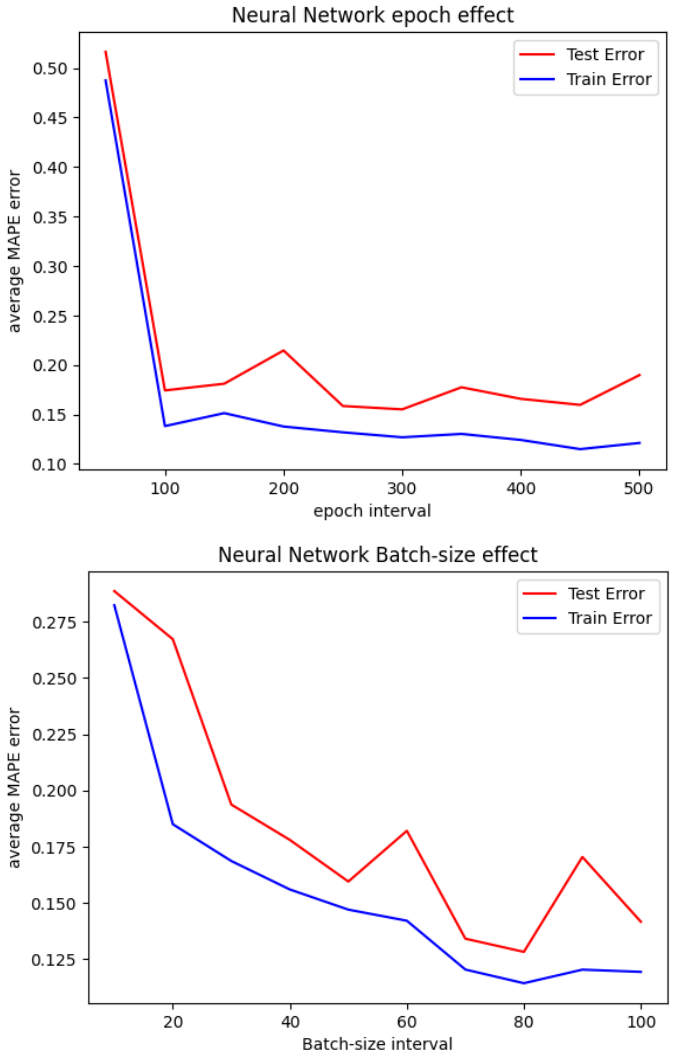

FIGURE 5.2: Neural Network parameters effect

5.2 Future works

Referring to [5], it turns a classification task to a weighting process for an agent to be taught using reinforcement learning. In the context of reinforcement learning, the focus is on training agents to make sequential decisions to maximize cumulative rewards, each state can be regarded as a sample in our problem. After assigning a weight to each class of five countries, we can convert the problem into a more homogeneous regression problem having more data, feeding into the ANN system to reach the desired accuracy.

Chapter 6

Conclusion

6.1 Conclusion

During the project, there were contacts with engineering teams to update or request new data as well as editing the data because there have been many mistakes in the dataset link data from India. The process of understanding of data and the basic perception of the data was done alongside with contacts with the teams. Although the target was not archived except in one country, there were justified reasons about the errors and the future progress. This reason can be regarded mostly because of the lack of data as we have seen in the neural network accuracy based on the batch-size, and the imbalance data which are explained in the report.

We make the decision to create a tool to estimate the cost of plastics as a regression problem. Currently, industrialization experts estimate the cost of the component. Finding a regression model that enables us to determine the manufacturing cost of a new plastic part using the straightforward visible parameters is what it entails. To estimate the investment envelope and the cost of the products, the cost that was obtained had to be fairly accurate.

The difference between the definitions of errors brings a challenge when it comes to having a small dataset. Noticing this point, to have a slight error on each sample we must be assured to own a homogeneous dataset. Otherwise, we can rely on noise reduction approaches or performing a classification task before the regression-based approach.

The ability to automatically calculate estimates of feature importance from a trained predictive model is a benefit of ensembles of decision tree methods, such as gradient boosting. The gradient boosting ensemble algorithm is implemented with strength and efficiency by XGBoost. The hyperparameters of XGBoost models can be difficult to

configure, which frequently necessitates the use of time-consuming large grid search experiments.

The negative side of XGB algorithm is where it tends to overfit and the error rate is also high that it can be understood that some part of the lack of accuracy is because of complexity of the problem. However there are few data representing poor estimations so we tried firstly to give weight to all samples, and second, propose some ensemble methods, improving the accuracy y focusing on the way that we measure the error. As a result of testing on all methods and ensemble methods, some improvement in the performance was observed.

Using neural networks to ensure discovering the deep relationship among features, requires more data than other methods while we should consider to avoid from overfitting on the dataset containing few number of samples. by observing comparision among models, the number of data strongly affects the performance of neural network.

References

[1] Xiongwen Quan Wei Li, Yanbin Yin and Han Zhang. Gene expression value prediction based on xgboost algorithm. *Frontiers in Genetics*, 10, November 2019. URL https://www.frontiersin.org/articles/10.3389/fgene.2019.01077/full.

[2] Alberto Landi; Paolo Piaggi; Marco Laurino; Danilo Menicucci. Artificial neural networks for nonlinear regression and classification. *10th International Conference on Intelligent Systems Design and Applications*, 11792539, December 2010. URL https://ieeexplore.ieee.org/document/5687280.

[3] Junyan Gao Zhenguo Yang Wenyin Liu Liuwu Li, Runwei Situ. A hybrid model combining convolutional neural network with xgboost for predicting social media popularity. *ACM Digital library*, page 1912–1917, October 2017. URL https://dl.acm.org/doi/10.1145/3123266.3127902.

[4] Ying-Cong Chen Hao Wang Dina Katabi Yuzhe Yang, Kaiwen Zha. Delving into deep imbalanced regression. *38 th International Conference on Machine Learning*, February 2021. URL http://proceedings.mlr.press/v139/yang21m/yang21m.pdf.

[5] Xiaoming Qi Enlu Lin, Qiong Chen. Deep reinforcement learning for imbalanced classification. *Applied Intelligence*, March 2020. URL https://link.springer.com/article/10.1007/s10489-020-01637-z.

[6] Padraig Davidson Anna Krause Andreas Hotho Michael Steininger, Konstantin Kobs. Density-based weighting for imbalanced regression. *springer link*, November 2021. URL https://link.springer.com/article/10.1007/s10994-021-06023-5.

[7] Lawrence O. Hall W. Philip Kegelmeyer Nitesh V. Chawla, Kevin W. Bowyer. Smote: Synthetic minority over-sampling technique. *artificial intelligence research*, June 2002. URL https://www.jair.org/index.php/jair/article/view/10302.

[8] Andreas W. Kempa-Liehr Justin M. O'Sullivan Nicholas Pudjihartono, Tayaza Fadason. A review of feature selection methods for machine learning-based disease risk prediction. *Frontiers in Bioinformatics*, June 2022. URL https://www.frontiersin.org/articles/10.3389/fbinf.2022.927312/full.

[9] Rita P. Ribeiro Paula Branco, Luís Torgo. Smogn: a pre-processing approach for imbalanced regression. *Theory and Applications*, September 2017. URL https://www.semanticscholar.org/paper/SMOGN%3A-a-Pre-processing-Approach-for-Imbalanced-Branco-Torgo/5839b2b19bf85a7b02b5bdabb752dae2993131ca.

[10] Padraig Davidson Anna Krause Andreas Hotho Michael Steininger, Konstantin Kobs. Smogn: a pre-processing approach for imbalanced regression. *Machine Learning*, July 2021. URL https://www.semanticscholar.org/paper/SMOGN%3A-a-Pre-processing-Approach-for-Imbalanced-Branco-Torgo/5839b2b19bf85a7b02b5bdabb752dae2993131ca.

[11] Berta Cajal Alfonso Palmer Juan José Montaño, Albert Sesé. Using the r-mape index as a resistant measure of forecast accuracy. *Psicothema*, November 2013. URL https://www.psicothema.com/pi?pii=4144.

[12] Xiongwen Quan Han Zhang Wei Li, Yanbin Yin. Gene expression value prediction based on xgboost algorithm. *ORIGINAL RESEARCH*, November 2019. URL https://www.frontiersin.org/articles/10.3389/fgene.2019.01077/full.

[13] Massachusetts Yoshua Bengio, Aaron Courville. Cambridge. Deep learning.