

Machine Learning

Defect Prediction on Production Line

Team:

AIT BACHIR Romuald

ALLEMAND Fabien

FLORET Arthur

JARDOT Charles



Contents

1	Introduction	1
2	Data Analysis	1
2.1	Data Description	1
2.2	In-Depth Analysis	1
3	Data Preparation	3
3.1	Outliers	3
3.2	Balance Classes	3
4	Model Selection	5
5	Model Fine Tuning	5
6	Conclusion	5

List of Figures

1	Distributions of the input features	2
2	Box-plot of the <i>snap_ring_final_stroke</i> feature	2
3	Confusion matrix of a k-Nearest Neighbors classifier with the base dataset	3
4	Confusion matrix of a k-Nearest Neighbors classifier with the dataset balanced by removing individuals	4
5	Confusion matrix of a k-Nearest Neighbors classifier with the dataset balanced by duplicating individuals	4

1 Introduction

The goal of this project is to develop an AI based solution for a real world industry problem faced by Valeo [1].

Valeo [2] is an industry leading French automotive supplier. In order to stay competitive, the company wants to develop a system that is able to identify defects on products before testing.

Four files containing relevant data are available. The data are mainly values measured during production on different mounting stations as well as additional measures performed on test benches.

The target is to find the best prediction: $\text{Output} = f(\text{inputs})$

2 Data Analysis

The goal of this section is to take a first look at the data in order to understand what it represents, then proceed to an in-depth analysis of said data.

2.1 Data Description

Data is contained in four different comma-separated values (csv) files: inputs and output for both training and testing. The very first line of each file corresponds to the headers. The following lines contain informations about a given product where each product is identified by a unique code:

ID = PROC_TRACEINFO = It's a unique code given to the product.

Example: I-B-XA1207672-190701-00494.

- XA1207672 is the reference.
- 190701 is the date: here 01st of July of year 2019.
- 00494 is the unique code given to the product, whatever it happens, the product will have this id number frozen forever.

This number is increased by 1 each time we process a new product, every 12 seconds. So for example: I-B-XA1207672-190701-00495 is the next product.

Input features are measures (numerical values) collected on different assembly stations with the sensors or devices connected to Programmable Logic Controllers which are storing all of them to keep the full quality traceability. Examples : OP070_V_1.angle_value, OP120_Rodage_L.value...

This is the result value of OP130 (test bench). Value 0 is assigned to OK samples (passed) and value 1 is assigned to KO samples (failed). This is the combined result of multiple electrical, acoustic and vibro-acoustic tests.

2.2 In-Depth Analysis

Let us put the testing data aside for now and take a closer look at the training data.

The training data consists of 13 measures (inputs features) made during production line on just shy of 35000 products. These measures correspond to various forces, angles, screwing torques... By taking a look at the distribution for each feature (Figure 1) we can clearly see that every one of them has a different shape.

An in-depth look at the box-plots (Figure 2) reveals that there might be some outliers in the dataset.

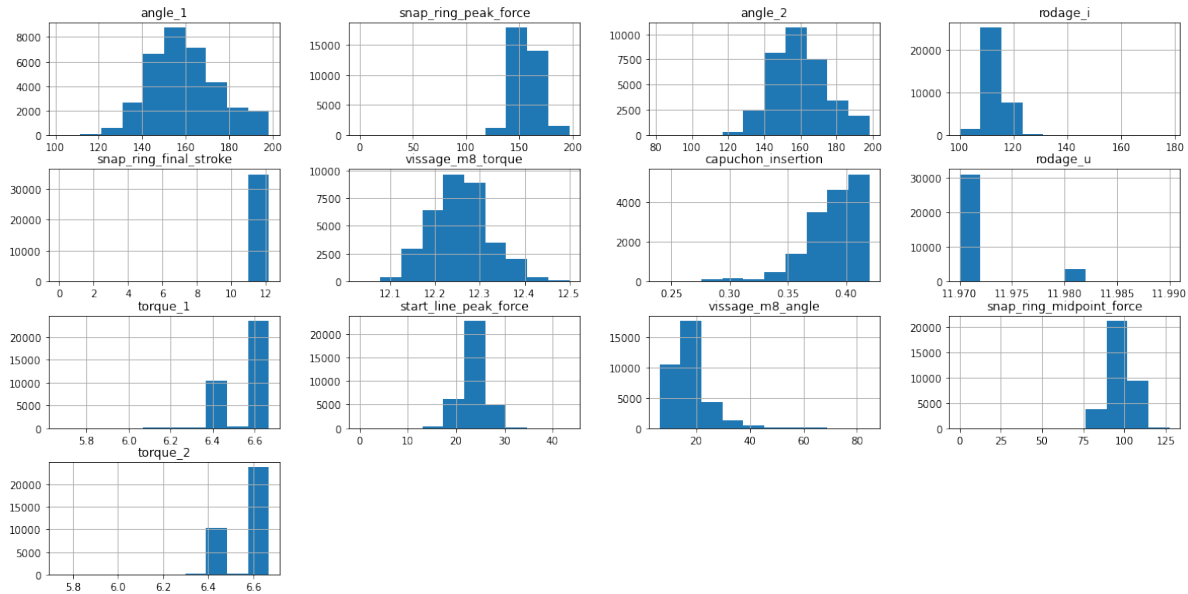


Figure 1: Distributions of the input features

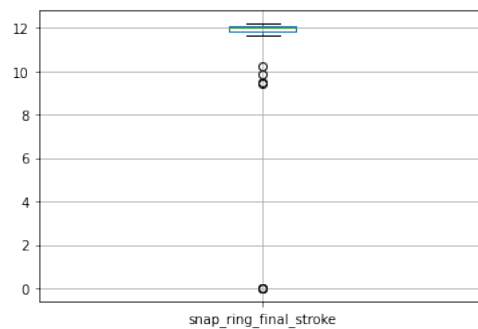


Figure 2: Box-plot of the *snap_ring_final_stroke* feature

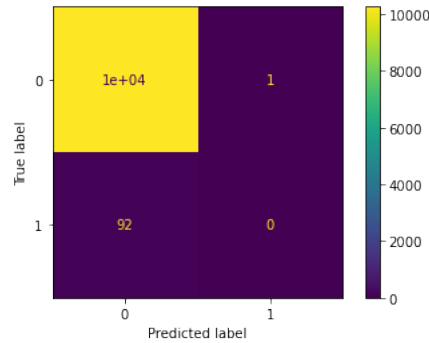


Figure 3: Confusion matrix of a k-Nearest Neighbors classifier with the base dataset

One feature requires more attention. `capuchon_insertion` is the only feature containing missing values. In fact, more than 50% of the sample population are missing this value.

The training data also contains the result value of the test bench for each product tested on the production line. As expected, there are only a few of defective items. Although this is good for the company, it means that less than 1% of the training population are defective items. The effectives of the two classes are very unbalanced which could really impact our models.

3 Data Preparation

The data analysis performed in the previous section revealed some flaws in the dataset. The two most noticeable are the presence of outliers and the population imbalance between the two classes. These two flaws are known for harming the learning of most ML models. Different attempts at solving these issues are presented in this section.

3.1 Outliers

3.2 Balance Classes

The following paragraphs deal with class balancing, explaining the importance of balanced classes and tree different approaches to rectify the imbalance.

As said in the previous section, a mere 0.008% of the population is defective individuals. In other words, the class of individuals which needs to be detected is truly under-represented in the dataset. Without any pre-processing such a dataset is very unlikely to give good results with most of the ML models. As shown in Figure 3, there are not enough defective individuals for the classifier to properly learn how to distinguish a defective individual from a valid individual. As a result, the classifier simply over-fit the valid class.

Remarque Over-fitting the over-represented class is very hazardous as the accuracy of the classifier will remain really high even though the classifier is not classifying anything. Hopefully the confusion matrix shows how the classification in more details.

The easiest method to deal with unbalanced classes is to remove valid individuals from the training dataset. This process is fairly straight forward as valid individuals can be randomly selected in the dataset and does not require a lot of processing as selected individuals are simply removed.

Figure 4 clearly shows that the classifier is not over-fitting the valid class anymore. In that sense, the results are better but they are still far from what is expected from a performant classifier.

This method is very arguable for two reasons.

Deleting individuals obviously implies a loss of data. Even though the individuals are randomly selected, some important features can disappear. In other words, the remaining data is not necessarily representative of the global population.

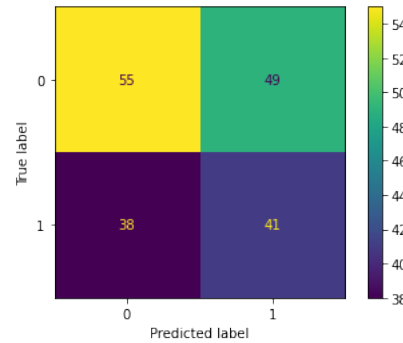


Figure 4: Confusion matrix of a k-Nearest Neighbors classifier with the dataset balanced by removing individuals

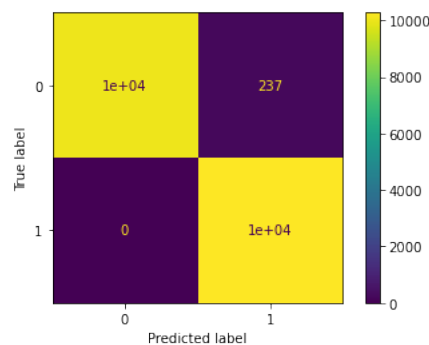


Figure 5: Confusion matrix of a k-Nearest Neighbors classifier with the dataset balanced by duplicating individuals

On top of that, the resulting dataset is smaller meaning the model will have less instances to learn on. In this problem, this will dramatically impact the results because there is already few data.

Of course it is also possible to work the other way around and artificially create individuals instead of deleting others. The easiest method is actually to duplicate defective items until there are as many as the valid ones. This compensate the imbalance between the two classes without knowing anything about the features values. Figure 5 shows the effect on the classifier is significant.

Another way that has been implemented during this project is to slightly modify individuals during the duplication. This way new individuals are actually new individuals (not only duplicates) adding information to the overall dataset. This method assumes individuals from the same class are close to one another on every feature and changing a feature's value of a small amount will not induce a change of class. The way it has been implemented is surely not ideal but rather intuitive. A random amount of features are selected during each "duplication" and the value is modified by a very small percentage (typically 0.005).

The final method in order to balance classes consists in leaving the classes imbalanced in the dataset but during computation attributing a more significant weight to the individuals from the under-represented class. Technically speaking this is equivalent to duplicating individuals but it requires less memory, it is easier for computation and is already implemented for some algorithms from well known ML libraries.

4 Model Selection

5 Model Fine Tuning

6 Conclusion

References

- [1] Defect prediction on production line by valeo. <https://challengedata.ens.fr/challenges/36>. Accessed: 2023.
- [2] Valeo. <https://www.valeo.com/fr/>. Accessed: 2023.