Statistical learning methods 22/23

Assignment

Quentin FEVER – S383387

Department: Applied Artificial Intelligence

Date: 31/10/2022

Information categorisation:
Open

Enhance the maintenance of aircraft engine

Project Overview

The objective of the analysis is to improve the maintenance operations and preventive maintenance planning of the aircraft engine at a given time. This is supposed to be done by applying statistical learning methods: regression and classification.

The aim of this project is the study of aircraft engine failure based on sensor data provided by Springboard DS Career Track Bootcamp. This study is divided into two parts. First the Time-To-Failure (TTF) prediction for the engine, then the binary classification of which engine will fail in the analysed time period.

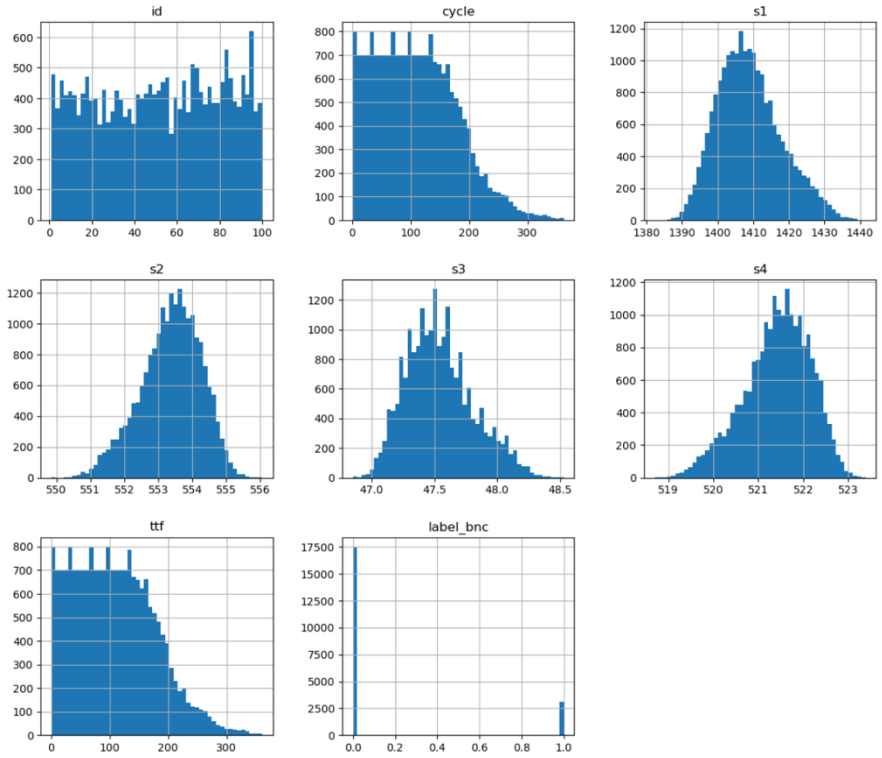
The complete code is available on my github at the following link:

https://github.com/quent1fvr/TimeToFail_prediction

Index table

Data exploration

First and foremost, the dataset contains no null or missing data.

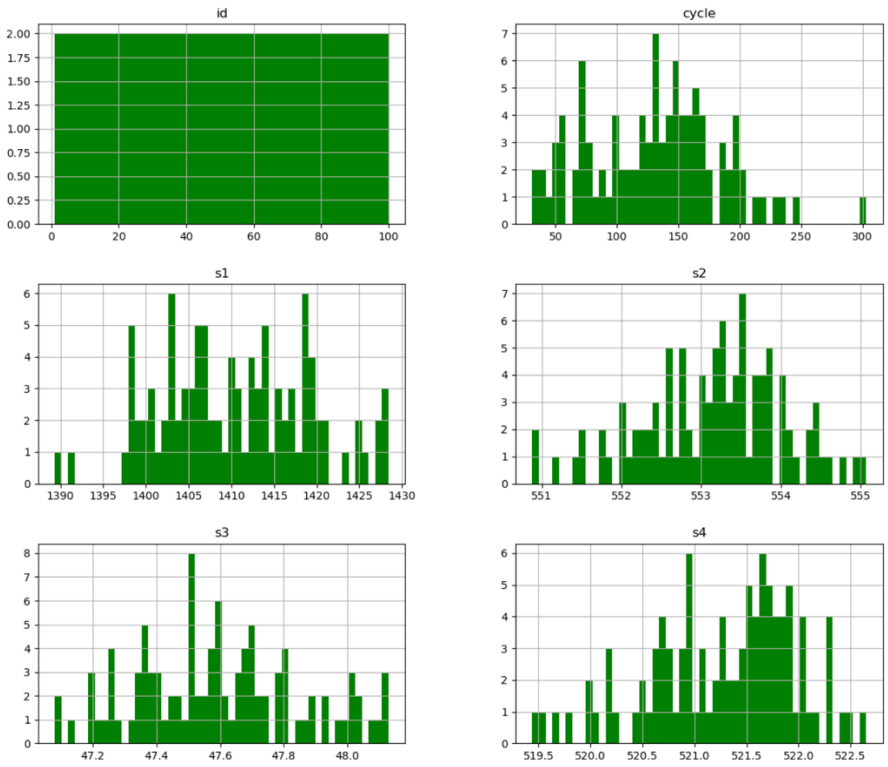


Features distribution of the training set

This drive data set is composed of:

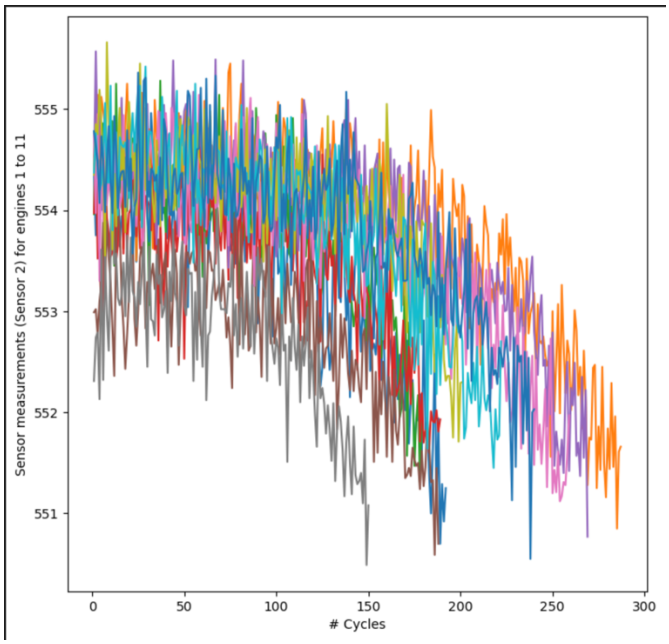
1. An id corresponding to each motor
2. The cycle, which corresponds to a temporal state data of the motor
3. Four sensors from S1 to S4 which quantitatively characterize the temporal state of the cycle of a motor
4. The Time-To-Failure (TTF) which is in the regression problem the label and what we will try to estimate
5. The label_bnc which is in the classification problem the label, 0 representing an operational engine and 1 a failed engine

It is noticeable that the distribution of sensor data can be like a Gaussian curve.



Features distribution of the test set

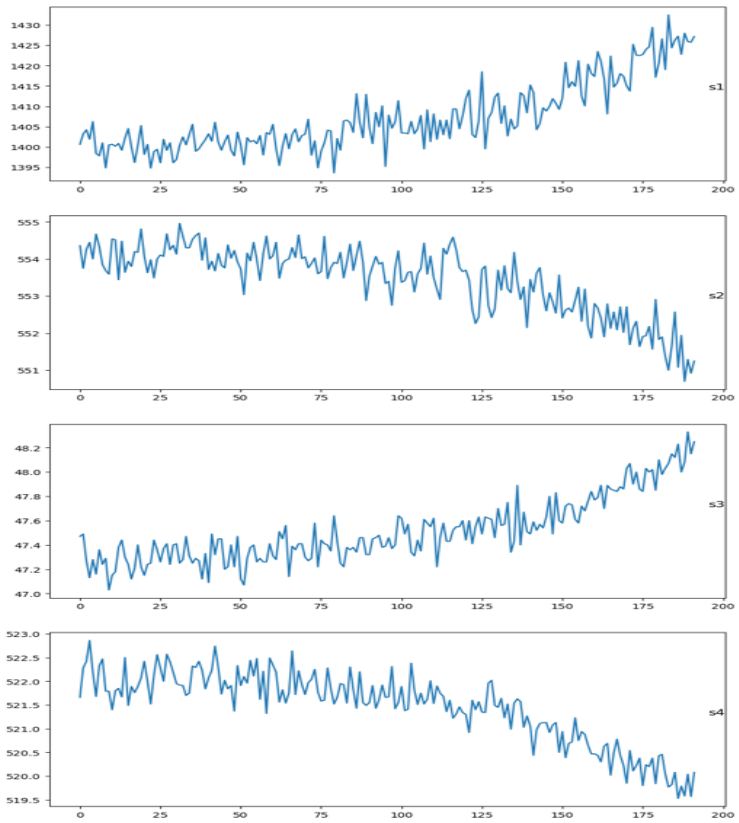
The test dataset has the same features as the training set, without the labels.
It is noticeable that the dataset looks less like a Gaussian distribution than the training set because of the size of the test set.
To get a concrete idea of the data present in the dataset, it is interesting to visualize them in 2D by arbitrarily choosing engines.



Sensor 2 measurements for engines 1 to 11.

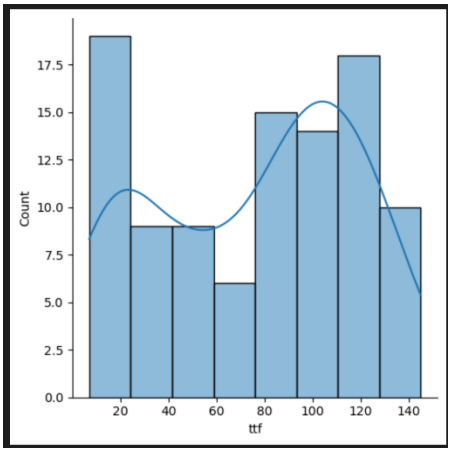
The graph on the left shows the evolution over the cycles of sensor 2 on motors 1 to 11.
They follow a similar pattern. As a first approximation, it seems possible to try a polynomial regression of degree 2.

Moreover, it is notable that the curves are noisy and that smoothing these curves may be an interesting path to pursue.



Evolution of sensors 1 to 4 of engine 1

Once again, the impression of noise noted on the previous graph with the engine is generalized to the other sensors.



Distribution of Time To Failure labels

the distribution of all label values is quite satisfactory. No label is overrepresented, and none is really underrepresented.

Regression

Problem Statement

The objective of this regression problem is to predict the Time To Failure (TTF) of aircraft engines based on a dataset composed of temporal data from sensors at different times.

Resolution axis

To answer this problem, the main idea is to train different statistical machine learning models, to draw results from them with a critical mind and to try to see improvement paths. For this, we will focus on 3 regression models, Random Forest (based on Decision Trees), Gradient Boosting and the K Nearest Neighbours algorithm.

Pre-processing

In our dataset, no value is missing, and none is null. The standardization step consists in normalizing data between 0 and 1.

To generalize the problem and give it more real meaning, it makes sense to remove the "id" column from our dataset to perform the learning. Indeed, in the industry world, the goal of the model is that it can predict the time to failure for new engines and not only those of our dataset. The "id" column is therefore removed for learning and for prediction.

	0	1	2	3	4
0	0.000000	0.309757	0.726248	0.369048	0.633262
1	0.002770	0.352633	0.628019	0.380952	0.765458
2	0.005540	0.370527	0.710145	0.250000	0.795309
3	0.008310	0.331195	0.740741	0.166667	0.889126
4	0.011080	0.404625	0.668277	0.255952	0.746269
...	...	...	...	...	...
20626	0.540166	0.782917	0.254428	0.726190	0.170576
20627	0.542936	0.866475	0.162641	0.708333	0.211087
20628	0.545706	0.775321	0.175523	0.738095	0.281450
20629	0.548476	0.747468	0.133655	0.916667	0.208955
20630	0.551247	0.842167	0.151369	0.803571	0.130064

20631 rows × 5 columns

Training game at the end of the Pre-Processing with the cycle feature and the four sensors

Model

As mentioned before, we will compare 3 regression models based on different approaches.

- The K-Nearest Neighbour algorithm
- The Polynomial regression
- The GradientBoosting algorithm based on decision trees and the Boosting technique.

Why focus on these models?

The K-Nearest Neighbour algorithm: Without doubt the most intuitive and easy to implement model, it will allow us to have a first reliable estimate. The only drawback is that the algorithm can become quite long in complexity in case of a large dataset, which is our case (>20 000 lines).

The Polynomial regression: Following the preliminary analysis of the dataset, it is clear that the evolution of the sensors is non-linear, so in this regression problem, it is preferable to consider a polynomial regression rather than a linear regression. For the choice of the degree of the pôlynome, the strategy will be to test different values of degree and to keep the one which allows to obtain the best values on the different metrics.

The GradientBoosting algorithm: This algorithm is less naive than the two previous ones, essentially to see if it allows to obtain better results on the different metrics.

Why did we choose these 3 models? Because they are based on different statistical concepts and do not belong to the same algorithm family. They are therefore likely to give different results.

We use the K-Fold cross validation technique to avoid overfitting on our data.

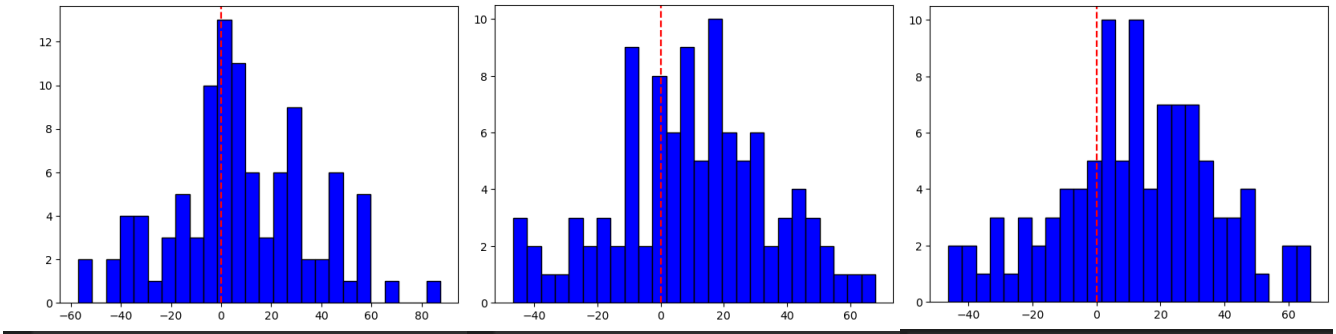
Evaluation

First, before discussing model evaluation, a look at the importance of the features of our training set, taken from the KNN model

Feature	Importance
cycle	75.52%
s3	11.75%
s1	6.09%
s4	4.08%
s2	2.55%

Feature Importance based on KNN model

It is normal that the cycle is the first feature in terms of importance, it is logical that as the cycle increases, the TTF decreases, there is a direct correlation between TTF and the cycle. The low importance of sensors and 2 is surprising, indeed the cycle and the sensor alone explain more than 87% of the model.

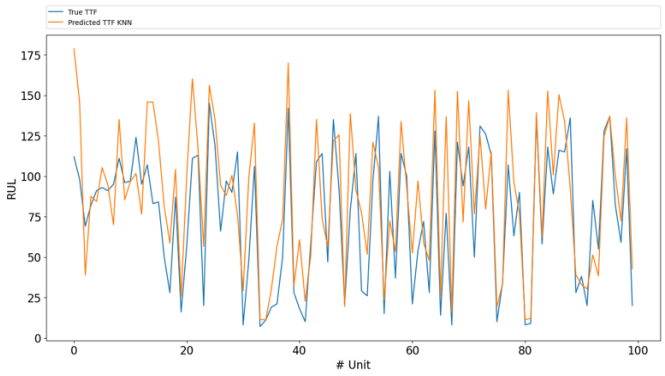


	Count
Smaller	40
Zero	0
Larger	60

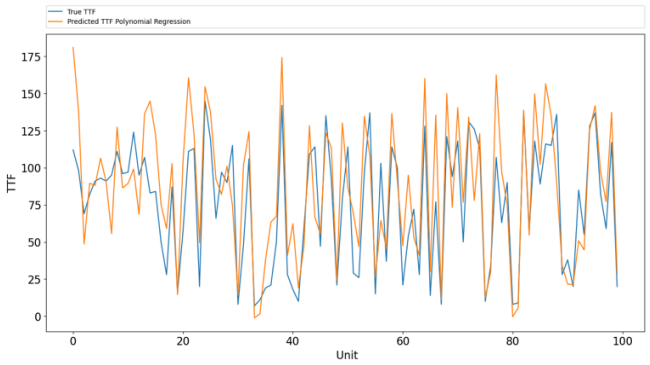
	Count
Smaller	33
Zero	0
Larger	67

	Count
Smaller	29
Zero	0
Larger	71

KNN model | Polynomial Regression | Gradient Boosting



Comparison True/Predicted value KNN model



Comparison True/Predicted value Polynomial regression

The graphs comparing the true value of TTF, and the estimated value are reassuring with respect to the learning of the three models. Although the RMSE is modest, the models globally follow the trend of the true value and in this sense, they can be qualified as reliable.

Model/Metrics	R squared	RMSE	MAE
KNN	50.20	29.33	22.41
Polynomial Regression	58.13	26.89	21.76
Gradient Boosting - Grid Search	56.74	27.33	22.22

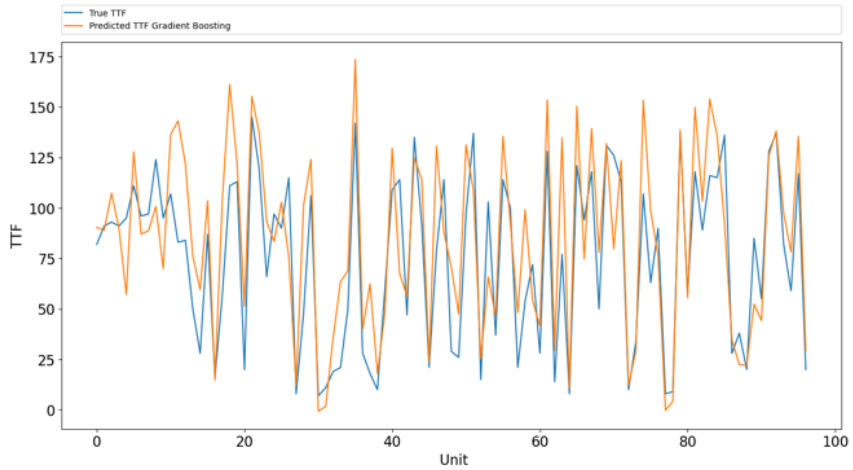
Models Performances

We can see that the most intuitive model in view of the distribution of the data and the shape of the sensor curves allow us to obtain the best results. Following this study, we can say that we can predict with an average error of 27 hours the Time to Failure knowing that the model tends to overestimate the Time To Failure.

It is also interesting to notice the prediction outliers on the test of the first engines for each of the developed models. To improve the model, it may be wise to remove these outliers from the test model, which will significantly improve the model.

Model/Metrics	R squared	RMSE	MAE
KNN	54.10	28.43	21.59
Polynomial Regression	61.69	26.09	21.10
Gradient Boosting + Grid search	60.72	26.30	21.10

Evaluation in % after elimination of outliers (the first three engines)

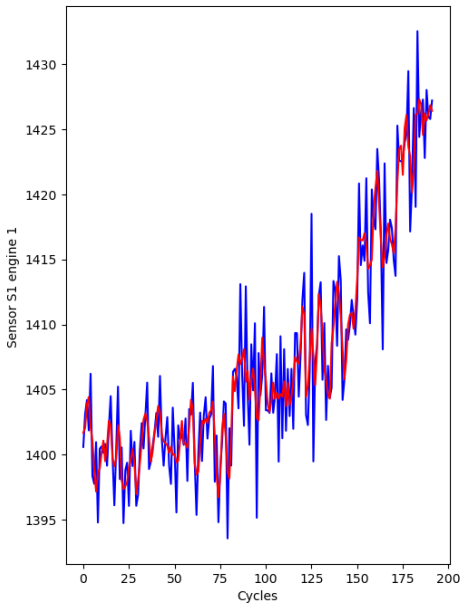


Evaluation after elimination of outliers on Gradient Boosting model

Admittedly, the models tend to overestimate the real TTF, but it is better from an industrial maintenance point of view to have an overestimated TTF than an underestimated one. By overestimating, the risks are reduced contrary to an under estimation.

The results can be considered acceptable although the R^2 is not closer than 1.
Indeed, with a mean square error of 26H, the model correctly predicts the time To Failure. Knowing that the model tends to overestimate, the maintenance operators can have a good estimate of decision support

As previously discussed, a prior smoothing of the curves may be judicious to eliminate noise. This track has been studied on the three models. The Savitzky-Golay algorithm was applied to the training set before learning.



Comparison true/smoothed curve example

The graph shows a satisfactory smoothing, the noise is clearly reduced while keeping a certain level of detail. Based on this algorithm, a new smoothed dataset has been created for training.

However, the results obtained after smoothing are worse than the previous ones, whatever the level of smoothing and the hyperparameters of the algorithm for an unknown reason. Perhaps the loss of information after smoothing is too great and the model fails to generalize as well.

Another possible reason is that only the training set is smoothed and not the test set. Perhaps the test data should be smoothed by inserting it in its respective place in the training set, but this is probably considered cheating and would not be applicable in industry, so this idea was not developed.

Improvement axis :

- It is obvious that having more data available remains the best axis for improvement.
- Try a Deep Learning model for time series with LSTM / GRU cells, because the LTSM and GRU cells learn directly from the previous moment, which can be useful when working with time series. This idea was not pursued because it is not the focus of this module.
- It may be possible to create a more realistic evaluation metric that would penalize underestimation more than overestimation.
- It may also be possible to apply a correction coefficient to compensate for the overestimation of the FTT.

Conclusion of the Regression problem:

This regression problem has allowed us to develop several models and several avenues of improvement, giving fairly satisfactory results that could be exploited in production.
It would be a question of putting it into production and disseminating the results on a Dashboard in an industrial maintenance department with a solution like Tableau Software for example.

Classification

Problem Statement

The challenge of the classification problem is to classify in a binary way whether the engine is failing at a given state.

Resolution axis

To conduct this study, two methods are presented below:

- K Nearest Neighbour
- Logistic regression
- The Random Forest Classifier algorithm based on ensemble methods and decision trees

The dataset is the same as the regression problem, it is just provided with an additional column "label_gnc" which takes a binary value 0 or 1 depending on whether the engine is considered failed or not.
Thus, unlike the regression problem where the goal was to predict a quantitative value (the TTF), here it is to predict a binary categorical value (0 or 1).

Notation: 0 corresponds to an operational engine and 1 corresponds to a failed engine.

Pre-processing

The dataset is similar to the previous one, so the conclusions are the same, the pre-processing phase is limited to the standardization of the data between 0 and 1.

Model

As mentioned before, we will compare three classifiers models based on different approaches.

- K-Nearest Neighbour algorithm
- The Logistic Regression
- The Random Forest Classifier

Why did we choose these 3 models? Because they are based on different statistical concepts and do not belong to the same algorithm family. They are therefore likely to give different results.

Evaluation

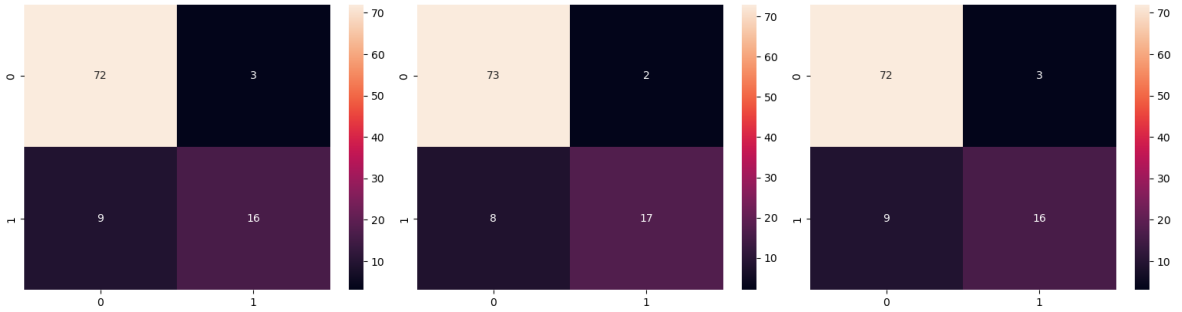
Without optimization, the results are quite good.
Both models offer similar performance. They make the same mistakes especially the false negatives (at least 8 per model, cf confusion matrix).
In agreement with the confusion matrices, the classifiers classify the non-failing engines without problems and make more errors to classify the failing engines.

Model/Metrics	Precision	Recall
Random Forest	0.87	0.80
Logistic Regression	0.90	0.83
KNN	0.87	0.80

Classification results

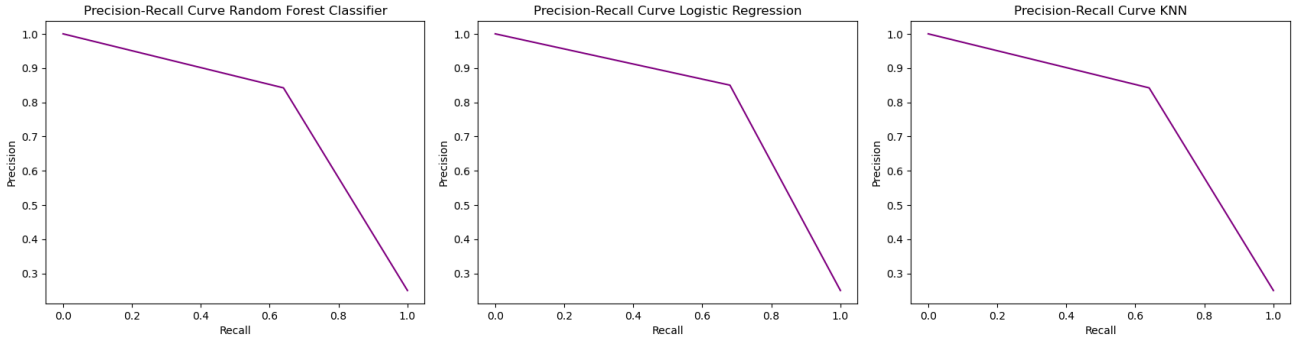
The macro-averaged Precision and Recall score are calculated using the mean of all the per-class Precision/Recall scores.

Confusion Matrix:



Random Forest *Logistic Regression* *KNN*

Precision-Recall Curve:



Random Forest *Logistic Regression* *KNN*

These curves are similar for each model. They show the compromise between precision and recall. It is impossible to have these two metrics equal to 1, we try to get closer by maximizing the 2. It depends on the strategy we want to develop and the question we want to answer.

Basically, the following interpretation can be given to Recall and Accuracy:

Accuracy: Among the predicted failed engines, how many are really failed?

Recall: Among the failed engines, how many have been correctly predicted?

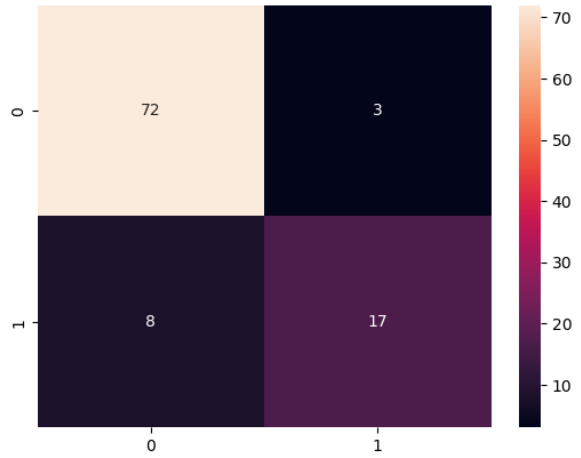
If we want to maximize the accuracy, we can set the classification threshold to 90%. The precision is then 100%:
The predicted failing engines are indeed failing. But what about recall? Among the real failed engines, the model will have detected very few, so the recall will be very low (<0.2)

If we want to maximize recall, we can set the classification threshold to 20%. The recall is then 100%:
All the engines that really fail have been detected by the model.
But what about accuracy?
Among the predicted failed motors, how many are really failed, the precision would be very low (<0.3 according to the graphs)

In this case of study, it is perhaps more judicious to try to maximize the recall, we try to identify all the failed engines even if some in the batch are not, from a safety point of view, it is surely the wisest choice.

Optimization and Tuning

To try to better classify the failing engines, the K Fold Cross Validation is used, it is coupled with the search of the best parameters of the models thanks to Grid Search CV.

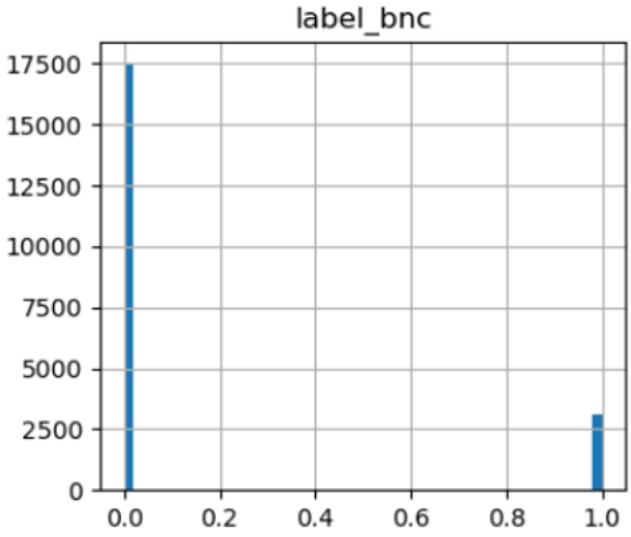


Results after optimization on logistic regression

The results do not vary after this optimization test on the logistic regression. The same is true for the Random Forest model.

Improvement axis:

The most interesting thing would be to have more training data from failed motors.



Label_bnc distribution

Indeed, on this graph, we observe that the dataset is unbalanced in favor of the healthy engine cases. This explains why the model has more difficulty to classify the failing engines than the healthy ones.

It might be possible to exploit a Data Augmentation technique to generate data close to failed engines as it is possible to do in Deep Learning by generating images artificially. This would make it possible to compensate for this lack of data.

As with the regression problem, trying a neural network Deep Learning model can be interesting because we have a fairly large volume of training data.

Conclusion of the Classification problem:

This classification problem has brought to light different models with quite similar results. Rather satisfying, the improvement tracks are probably based more on the data collection than on the processing that is done afterwards.

Bibliography

[1] *scikit-learn: machine learning in Python — scikit-learn 1.1.3 documentation*. (n.d.). https://scikit-learn.org/stable/

[2] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics)* (1st ed. 2013, Corr. 7th printing 2017). Springer.

[3] Murphy, Kevin P. *Machine learning: a probabilistic perspective*. MIT Press, 2012.