Politecnico di Torino

Mathematics in Machine Learning

Gabriele Rosi s291082

A.Y. 2021/2022

01

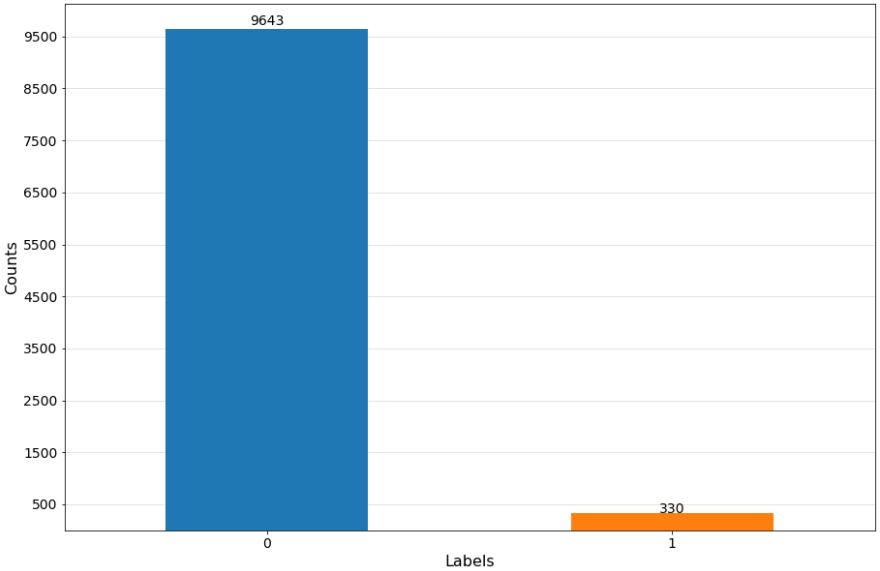
Data analysis

Data overview

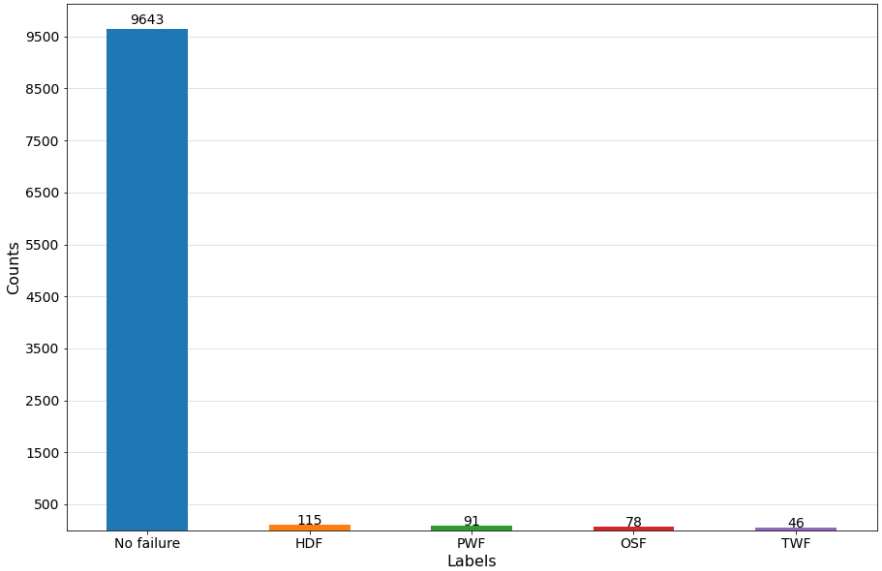
Our dataset is composed by 10000 records and 14 columns.

- UID (*numerical*)
- Product ID (*string*)
- Type (*categorical*)
- Air temperature [K] (*numerical*)
- Process temperature [K] (*numerical*)
- Rotational speed [rpm] (*numerical*)
- Torque [Nm] (*numerical*)
- Tool wear [min] (*numerical*)
- Machine failure (*binary*)
- TWF, HDF, PWF, OSF, RNF (*binary*)

Target variable distribution

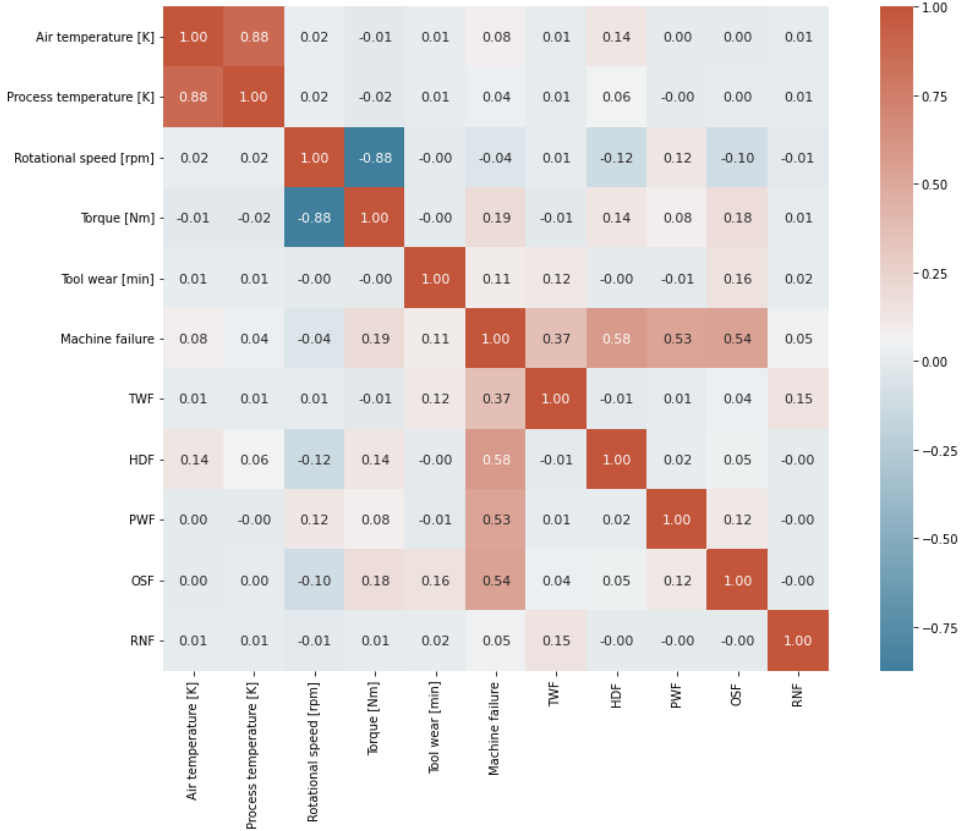


Machine failure distribution



Failure type distribution

Correlation



Positive correlation

- *Machine failure* and *Failure Types*
- *Air temperature* and *Process Temperature*

Negative correlation

- *Torque* and *Rotational speed*

02

Data preprocessing

General preprocessing

Normalization

Air Temperature, *Process temperature*, *Rotational speed*, *Torque* and *Tool wear a*re on different scales and not normally distributed.

$$Z = \frac{X - \mu}{\sigma}$$

Categorical features encoding

Type can assume {H,M,L}. We encode it using one-hot encoding.

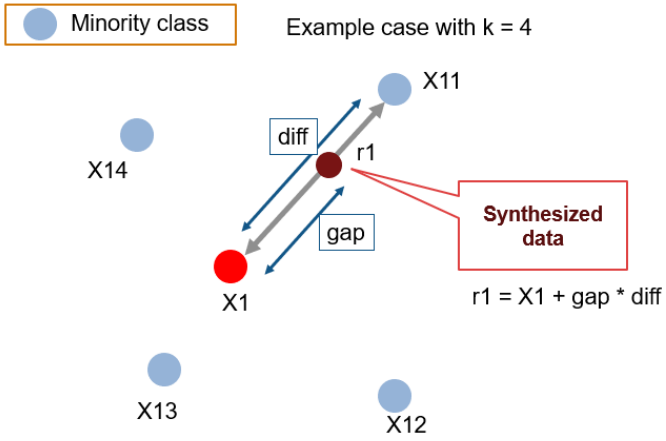
Failure type decoding-encoding

We reverse the one-hot encoding and then we encode each class with an ordinal number from 1 to 5.

Dataset balancing

The dataset used is very unbalanced. To resolve this issue, we apply **SMOTE** (*Synthetic Minority Over-sampling Technique*). In the current situation, undersampling will lead to a huge loss of data.

For each point in the minority class, k nearest neighbours are found. Then by selecting one random neighbour, we compute the difference between the two points and then we multiply it by a random number $\in [0, 1]$. This gives us a synthetic example along the line between the two points.



Dimensionality reduction

PCA is an unsupervised algorithm whose goal is to project data to a lower dimensional space. PCA tries to find a linear transformation that minimize the difference between the original vector x and recovered vector $\tilde{x} = UWx$:

$$\underset{W \in \mathbb{R}^{n,d}, U \in \mathbb{R}^{d,n}}{\operatorname{argmin}} \sum_{i=1}^{m} \|x_i - UWx_i\|_2^2$$

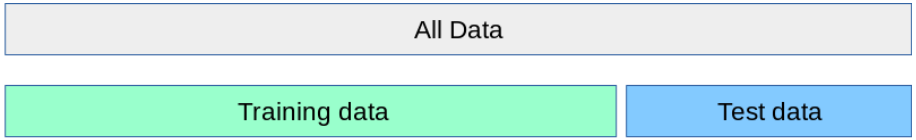
Regarding our dataset, we decide to **not apply** a dimensionality reduction technique since the features are not so many and the number of instances is way bigger than the number of features.

03

Validation and metrics

Data splitting

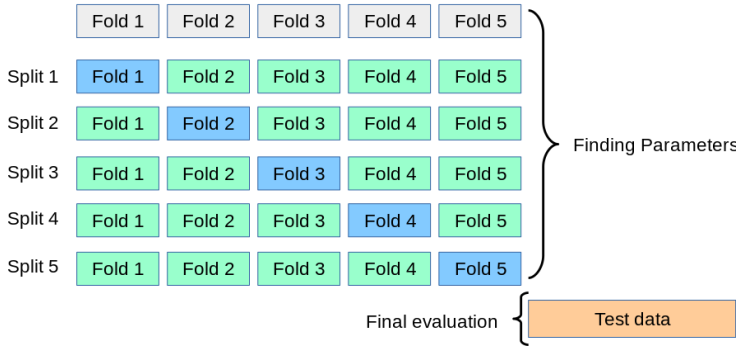
The dataset is **split into a training and a test** set with a ratio of 80/20.
In order to approximately preserve the same distribution of the target variable, a stratified split is performed.



Cross validation

While performing hyperparameters tuning, we have a high risk of overfitting on the test set. To mitigate this risk, we apply a cross validation technique, in particular **Stratified K-Folds cross validation**.

1. The training set is divided in k folds with the same class ratio as in the original dataset
2. For each fold, the model is trained on $k-1$ folds and validated on the remaining one.
3. The process is repeated k times, with each of the k folds used exactly once as the validation data.

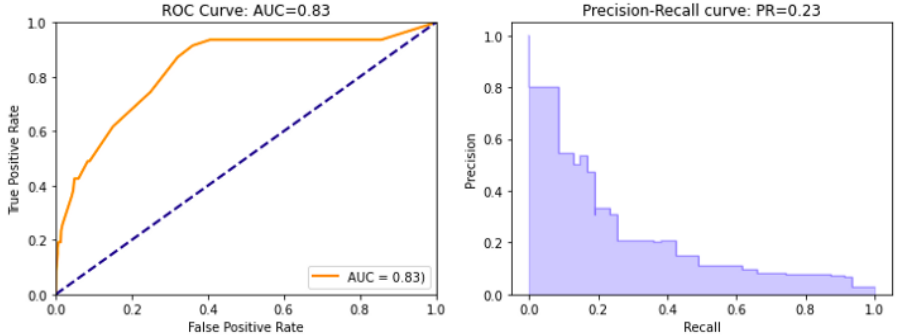


Metrics (1)

- Accuracy: measures the number of correct predictions over the total number of predictions.
- Precision: measures the number of true predicted positives over the total number of positives.
- Recall: measure the number of correctly positive predicted values over the total actual positives.
- F1-score: harmonic mean between precision and recall.
- Confusion matrix: matrix in which each row is an instance of the true class, and each column are the instances of the predicted classes.

Metrics (2)

- ROC curve: plot illustrating the classification performance. Basically, on the x-axis we have the false positive rate and, on the y-axis, the true positive rate.
- Precision-Recall curve: plot like ROC curve but is a useful measure of success of prediction when the classes are very imbalanced. A high area under the curve represents both high recall (low false negative rate) and high precision (low false positive rate).



04

Models

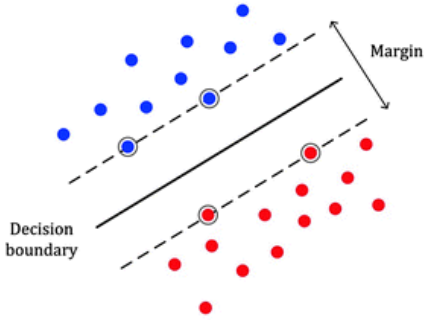
SVM (1)

Support Vector Machine (SVM) is a supervised method used both for classification and regression, whose goal is to find a hyperplane that maximize the distance between two classes in a feature space.

Hard Margin: given any separable training set, we take the hyperplane that maximize the margin, i.e. the minimal distance between a point of the training set and the hyperplane itself.

$$\underset{(w,b):\|w\|=1}{\operatorname{argmax}} \min_{i\in[m]} |\langle w, x_i\rangle + b| \quad \text{s.t.} \quad y_i\left(\langle w, x_i\rangle + b\right) > 0 \quad \forall i$$

Solution: $\widehat{\mathrm{w}} = \frac{\mathrm{w}_0}{||w_0||} \quad \hat{b} = \frac{\mathrm{b}_0}{||w_0||}$



SVM (2)

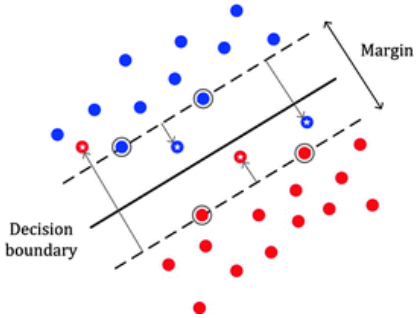
Linearly separable data is a quite hard assumption. If we relax this constraint, we end up with the Soft Margin.

Soft Margin: we can allow the violation of the constraint introducing slack variables ξ_i. The parameter C control the number of violation we allow.

- C big → margin is smaller → rare to make mistakes
- C small → margin is bigger → more likely to make mistakes

$$\min_{\mathrm{w},\mathrm{b},\xi} \left(\lambda ||\mathrm{w}||^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i \right) \quad \text{s.t.} \quad y_i \left(\langle w, x_i \rangle + b \right) \geq 1 - \xi_{\mathrm{i}}$$

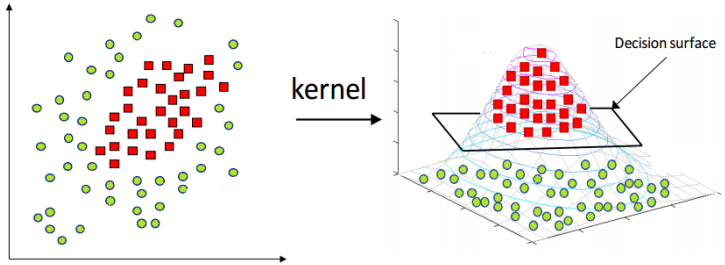
$$\sum_{i=1}^{m} \xi_i \leq C$$



SVM (3)

If data are not linearly separable, we need to map them from the original space into a higher dimensional feature space. However, computing linear separators in very high dimensional space can be very computation expensive. The **kernel "trick"** provides a solution to this problem. The kernel function is a function that takes as input the vectors in the original space and return the dot product of the vector in the features space.

$$K(\mathrm{x}, \mathrm{x}') = \langle \psi(\mathrm{x}), \psi(\mathrm{x}') \rangle$$



Decision trees

Decision tree is a supervised classification algorithm whose goal is to create a model by **learning simple decision rules** inferred from the data features.

Finding every possible partition of the feature space is unfeasible at training time. So, we start at the top of the tree and at each step, the **local best split** is performed, rather than the one that would lead to the best global result. The most common criteria used to obtain the best split are *Gini Index* and *Cross entropy*.

The main disadvantage of decision trees is that they tend to **overfit the training set**, especially if we build very complex trees.

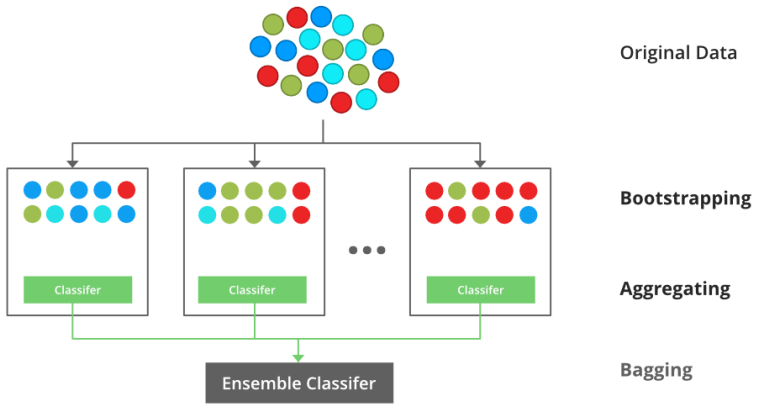
Random forest

Random forest has been created to resolve the overfitting issue of decision trees. The main idea behind is **bagging** (***b****ootstrap-****agg****regat****ing***): multiple prediction functions learned from different dataset is combined to reduce the variance.

1. Bootstrapping: uniform sampling at random with replacement from the training data.

2. *N* decision trees is built.

3. Aggregating: the class is chosen via majority voting among the different predictions.



Logistic regression

Logistic regression estimates the probability of an event occurring. Since the outcome of the logistic function is a probability, the output is bounded between 0 and 1.

To obtain the class (instead of the probability), we need to define a decision boundary. In the binary case, by default, if the probability is above 0.5, then the class 1 will be assigned and the class 0 otherwise.

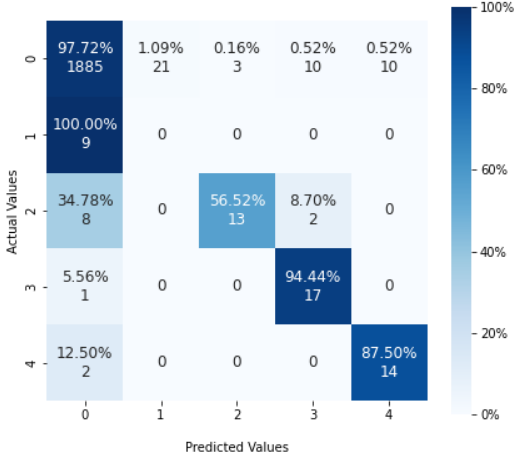
Binary case → sigmoid function $\mathbb{P}(y_i = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$

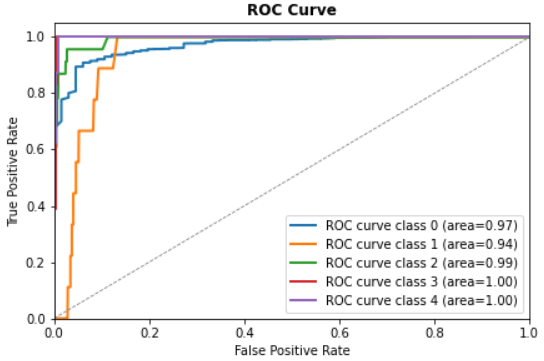
Multi-label case → softmax function $\mathbb{P}(y_i = k) = \frac{e^{\beta_{0k} + \beta_{ik} X_i}}{\sum_{l=0}^{K-1} e^{\beta_{0l} + \beta_{il} X_i}}$

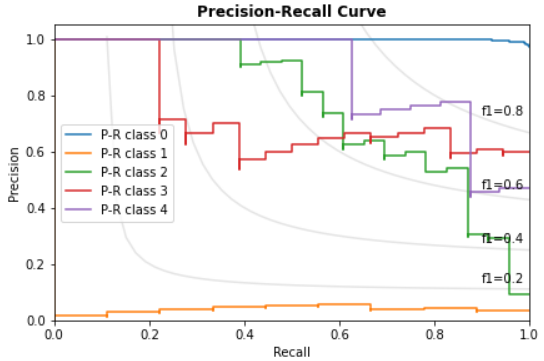
05

Results

Results (1)





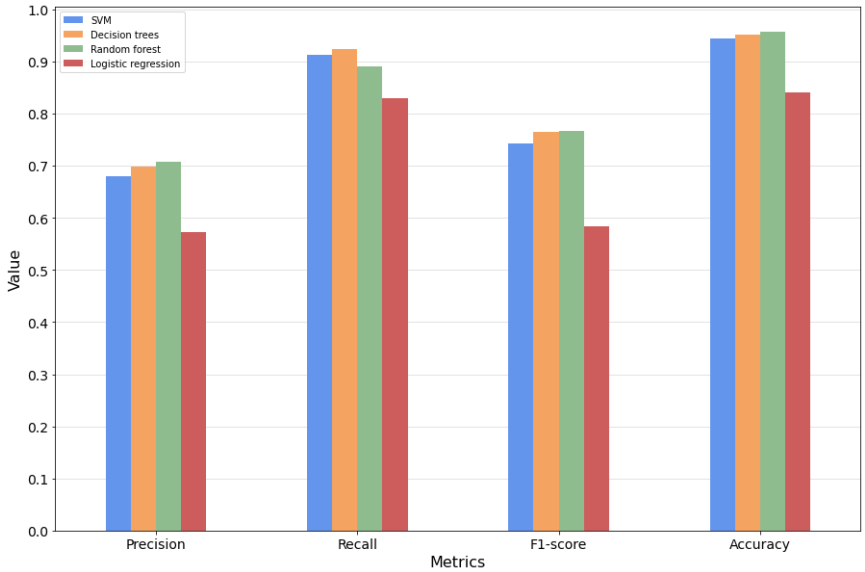


	precision	recall	f1-score
0.0	0.990	0.977	0.983
1.0	0.000	0.000	0.000
2.0	0.812	0.565	0.667
3.0	0.586	0.944	0.723
4.0	0.583	0.875	0.700
accuracy			0.967
macro avg	0.594	0.672	0.615

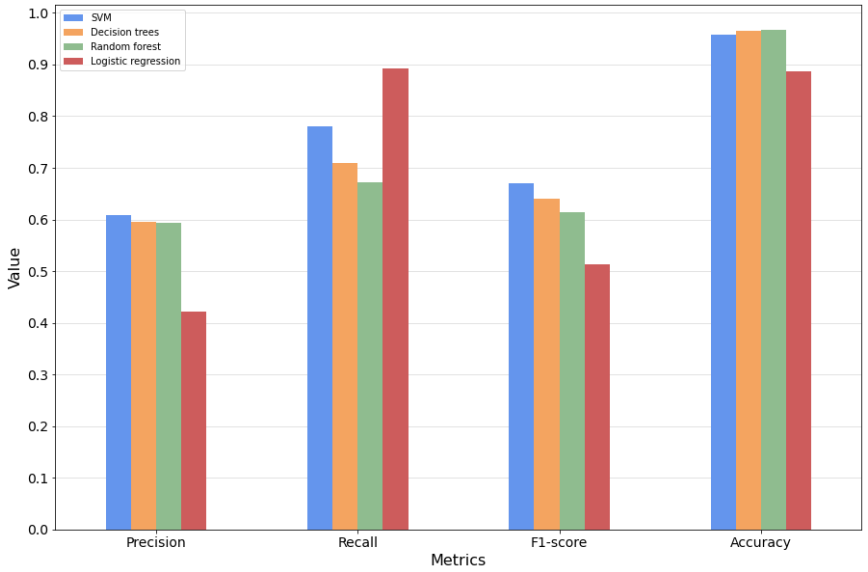
Failure type random forest classifier results

Results (2)

In both the classification task, we obtain a high accuracy with all the model tested. Precision instead is not so good probably because of data imbalance.



Machine failure classification



Failure type classification

Results (3)

Which model to choose depends on the result that we need.

For the sake of simplicity just consider the machine failure classification task. If we choose:

- Model with high recall → we allow to predict a machine failure also if it will not and we want to be sure that almost all the real failure will be detected
- Model with high precision → we do not want any false positive failure alarm and we can allow that some machine failure will be classified as non failure

Thanks for your attention!