//============================================

How to become machine learining expert!

https://www.kaggle.com/discussions/general/262128

//==================================================================

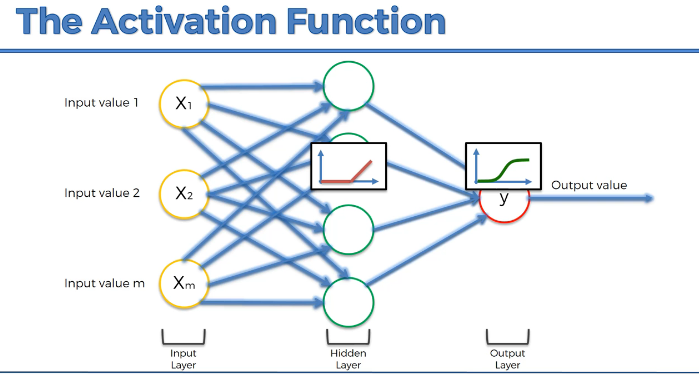
recall simple blogs for linear regression, multiple linear regression and logistic regression

<https://www.superdatascience.com/blogs/the-ultimate-guide-to-regression-classification>

Deep learning Notes:

Here we 4 types functions that are used in neural networks architectures

1. Thershold function
2. Sigmoid function
3. Rectify function
4. Hyperparabolic tangent function



=============================================

curse of dimensionality example ?

-> The curse of dimensionality is like adding more and more ingredients to a recipe.

-> In low dimensions, you can easily manage and understand the flavor. However, as you add more and more ingredients (dimensions),

-> the overall taste becomes more complex, and it becomes challenging to maintain a balanced and reliable recipe.

-> Similarly, in high-dimensional data, maintaining a clear understanding and making accurate predictions becomes more challenging

gradient descent with example ?

-> In a real-world scenario, you might be trying to find the best settings for a recipe.

-> You close your eyes, take a small taste, and adjust the ingredients based on whether it tastes better or worse.

-> You repeat this process until you have the perfect recipe.

Stochastic gradient descent with example ?

-> In a real-world scenario, think of each cookie as a customer review for a product.

-> Instead of waiting for all reviews to come in, you continuously read and adjust your product based on

-individual reviews until you have a product that satisfies most customers

* The first thing we can observe is that the data augmentation was performed succesfully,
* as the feature distribution for faulty instancies have not been significantly distorted(mishappend).
* It should also be noted that in Rotational Speed, Torque and Tool Wear the observations relating to failures have a density peak in extreme zones of the distribution.
* This implies that the outliers we discussed in Section 2.3 are not to be imputed(assign) to mistakes in the dataset building but rather to the natural variance of the same.
* This becomes even clearer when observing the distributions relative to the single causes of failure: in particular, an almost symmetrical behavior is recognized in Rotational Speed and Torque while in Tool Wear a clear separation is observed between PWF and HDF failures on lower values, and the peaks that are found at higher values relative to TWF and OSF. This is perfectly consistent with the description of the targets reported in the "Task and dataset description" section.

## Principal component analysis ( PCA ):

Simplify the complex scenes, identifying the underline patterns, reducing dimensionality of reduction

How it works PCA:

1. We have to ensure the scaling all features.
2. Covariance of matrix. It shows the relationships features in all variables
3. Find out eigen values and eigen vectors. With help of eigen values to represent the principal component and eigen vectors are represent the direction of principle component.
4. Select the most data falls in which category ( component)
5. Project the original data into selected components to reduce the dimensionality.

The idea of PCA is simple — **reduce the number of variables of a data set, while preserving as much information as possible.**

For better understand PCA read this article

<https://builtin.com/data-science/step-step-explanation-principal-component-analysis>

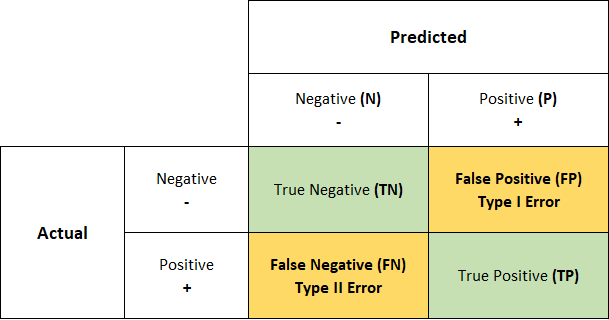
* we observe that the features related to temperature, as well as those related to power, are widely correlated.
* Furthermore, Tool Wear correlates well with both of our targets, confirming what we have observed by studying PCA.
* Finally, a less strong correlation is also observed between the torsion and the two targets.

1. The goal of this section is to find the best model for binary classification of the dataset to predict whether or not there will be Machine Failure.
2. Classification algorithms are part of data mining and use supervised machine learning methods to make predictions about data. In particular, a set of data already divided (”labeled”) into two or more classes of belonging is provided as input thanks to which a classification model is created, which will than be used on new (”unlabeled”) data to assign them to the appropriate class.
3. The starting dataset is usually divided into three groups: the training dataset, i.e. the sample of data used to fit the model, the validation dataset, i.e. the sample of data used to provide an evaluation of a model fit on the training dataset while tuning model hyperparameters and the test dataset, which has the purpose of testing the model. At the beginning of a project a data scientist must make this division and the common ratios used are:

Metrics types from sklearn.

1. Confusion Matrix
2. Accuracy Score.
3. F1 Score
4. ROC score.

**Confusion Matrix,**



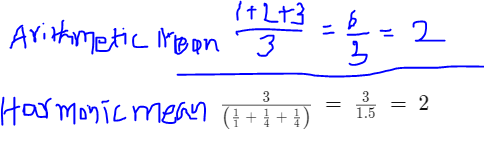
========================================

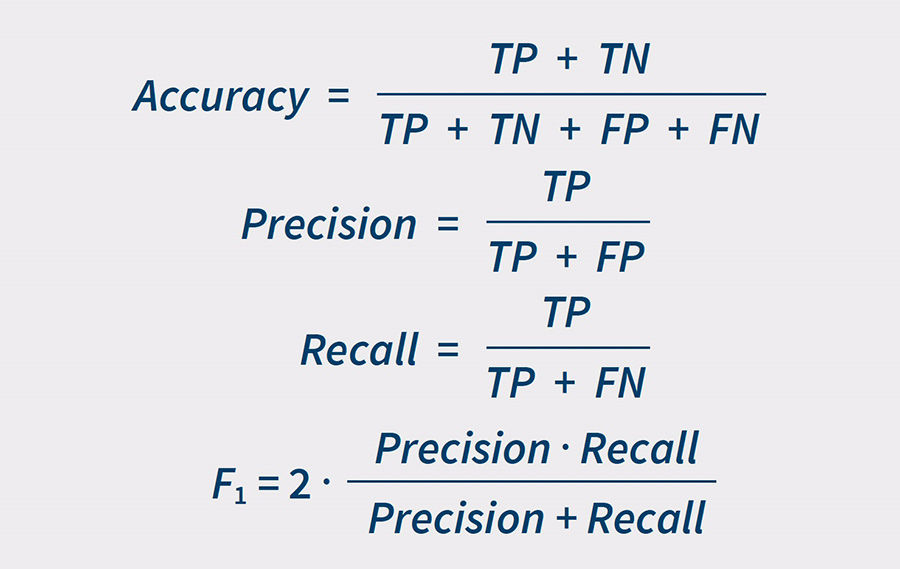
**Accuracy Score and F1 Score**

Accuracy : where the classes are balanced we can use accuracy score.

F1 Score : Especially useful when there is an imbalance between classes or when both precision and recall need to be considered.

The reason the harmonic mean is commonly used in the F1 score is because it naturally balances precision and recall. This is particularly valuable in situations where both false positives and false negatives have significant consequences, and you want to avoid favoring one at the expense of the other.



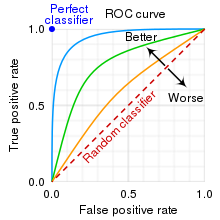


**ROC** **(Receiver Operating Characteristic Curve)**

Particularly relevant in industries where distinguishing between true positives and false positives is critical, such as in medical diagnostics or fraud detection..we can use the ROC when binary classification problems.

Refer this youtube video link for better understand

<https://www.youtube.com/watch?v=4jRBRDbJemM&pp=ygUDcm9j>



=========================================

OVR (One Vs Rest):

* In roc\_auc\_score, multi - class level we have multi-class parameters. It's a default behavior. For each class , it treats that class as the positive and rest as the negative class.
* Then it calculates ROC AUC for each class independently.

Average:

* In fbeta we have an average parameter that is used to specify how the scores should be aggregated across different classes in multi class classification problems.
* why are we using beta parameters in Fbeta\_score ?

The beta parameter determines the weight of recall in the combined score. beta < 1 lends more weight to precision, while beta > 1 favors recall (beta -> 0 considers only precision, beta -> +inf only recall).

Why we using stratify parameter in train\_test\_split()?

X\_train\_val, X\_test, y\_train\_val, y\_test = train\_test\_split(X,y, test\_size=0.1,

stratify=df\_pre['Failure Type'],

random\_state= 0)

* By using stratify we can split the data and ensure that both the training and testing sets have a similar proportion of passing and failing students. This helps your model learn from both types of students, making it better at predicting whether any student, passing or failing, will do well.

Below code is Stratify example.

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import make\_classification

# Generate a synthetic dataset with imbalanced classes

X, y = make\_classification(n\_samples=1000, n\_features=5, n\_classes=2, weights=[0.9, 0.1], random\_state=42)

# Without stratify

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# With stratify

X\_train\_stratify, X\_test\_stratify, y\_train\_stratify, y\_test\_stratify = train\_test\_split(X, y, test\_size=0.2, stratify=y, random\_state=42)

# Print class distribution in the original dataset

print("Original class distribution:")

print("Class 0:", sum(y == 0))

print("Class 1:", sum(y == 1))

print()

# Print class distribution in the datasets without stratify

print("Class distribution without stratify:")

print("Training Set - Class 0:", sum(y\_train == 0))

print("Training Set - Class 1:", sum(y\_train == 1))

print("Testing Set - Class 0:", sum(y\_test == 0))

print("Testing Set - Class 1:", sum(y\_test == 1))

print()

# Print class distribution in the datasets with stratify

print("Class distribution with stratify:")

print("Training Set (Stratified) - Class 0:", sum(y\_train\_stratify == 0))

print("Training Set (Stratified) - Class 1:", sum(y\_train\_stratify == 1))

print("Testing Set (Stratified) - Class 0:", sum(y\_test\_stratify == 0))

print("Testing Set (Stratified) - Class 1:", sum(y\_test\_stratify == 1))

Output for above code

Original class distribution:

Class 0: 895

Class 1: 105

Class distribution without stratify:

Training Set - Class 0: 710

Training Set - Class 1: 90

Testing Set - Class 0: 185

Testing Set - Class 1: 15

Class distribution with stratify:

Training Set (Stratified) - Class 0: 716

Training Set (Stratified) - Class 1: 84

Testing Set (Stratified) - Class 0: 179

Testing Set (Stratified) - Class 1: 21

Classification models

1. Logistic regression
2. KNN(K-Nearest Neibors)
3. Support vector machines
4. Random Forest
5. XGBoost

XGBoost:

XGBoost is a Gradient boosted decision tree (GBDT)is machine learning library.

GBDT is decision tree ensemble learning algorithm, similar like random forest.

Difference is boosting technique, it trains the ensemble of shallow decision trees

With each iteration error residuals of the previous model to fit into the next model.

The final prediction is sum of all the tree predictions.

Purpose of Permutation feature importance:

For good accuracy depends on selecting features. Which feature is best for accuracy to get this we using permutation feature importance.

How its works:

* Original dataset features compared to shuffled the values in single feature. Observe the model performance
* Main purpose why we using permutation is how model is depends on which feature is important to get good accuracy.

Multi - class task description

* We now proceed to the second task of this project, that is predict not only if there will be a failure, but also the type of failure that will occur.
* So we are in the case of multiclass classification problems that make the assumption that each sample is assigned to one and only one label.
* This hypothesis is verified because in data preprocessing we removed all the ambiguous observations that belonged to more than one class.

==========================

* For multiclass targets, when we calculate the values of AUC, F1 and F2 scores, we need to set the parameter "average". We choose "average=weighted", in order to account for class imbalance: in fact, at the end of data preprocessing, we have 80% WORKING machine and 20% that fail.
* As for binary classification task, we choose Logistic Regression as baseline model
* and we look for models that get higher values for the chosen metrics.
* In particular, we adapt to the multiclass case the models developed in the previous section.
* While many classification algorithms (such as K-nearest neighbor, Random Forest and XGBoost) naturally permit the use of more than two classes,
* some (like Logistic Regression and Support Vector Machines) are by nature binary algorithms;
* these can, however, be turned into multiclass classifiers by a variety of strategies.
* For our project, we decide to use "OnevsRest" approach, who involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives. We choose it because it is computationally more efficient than other types of approach.

Conclusion about the Predictive Maintenance Project:

According to the analyses carried out and the results obtained, it is possible to make some conclusive considerations related to this project.

We decided to tackle two tasks:

1. predict whether a machine will fail or not and
2. predict the type of failure that will occur.
3. Before developing the models we did data preprocessing to ensure the validity of the assumptions of applicability of the models and ensure the best performances.
4. Briefly, in the preprocessing phase we have deleted some ambiguous samples,
5. We applied a label encoding to the categorical columns and then we performed the scaling of the columns with StandardScaler. We also noticed the presence of some data points which at first we referred to as outliers but later turned out to be part of the natural variance of the data and played an important role in the classification task.
6. Then we ran PCA and found that most of the variance is explained by the first three components, that can be represented as the following features:

* combination of the two Temperatures,
* Machine Power (product of Rotational Speed and Torque) and
* Tool Wear.

1. In according to this, we found that these are the features that contribute the most in the predictions when apply the models.
2. Contrary to logical predictions, we demonstrated that the machine’s type does not affect the presence of failure.

* At the end, we can conclude that for both task the chosen models perform very well
* For both tasks the best model is XGBoost and the worst is KNN;
* However, the response time of KNN is instant while XGBoost takes more time and this further increases when we proceed with the multi-class classification task.
* The choice of the model depends on the needs of the company:
* For faster applications one can use KNN while if one cares more about accuracy one can use XGBoost.

// =======================================