x1(features) | x2 | ...| xn | y (outputs) | y2 (outputed with the model using the first parameters) | y-y2= Error function

inputs//variables

On veut diminuer l'Error en changeant les paramter

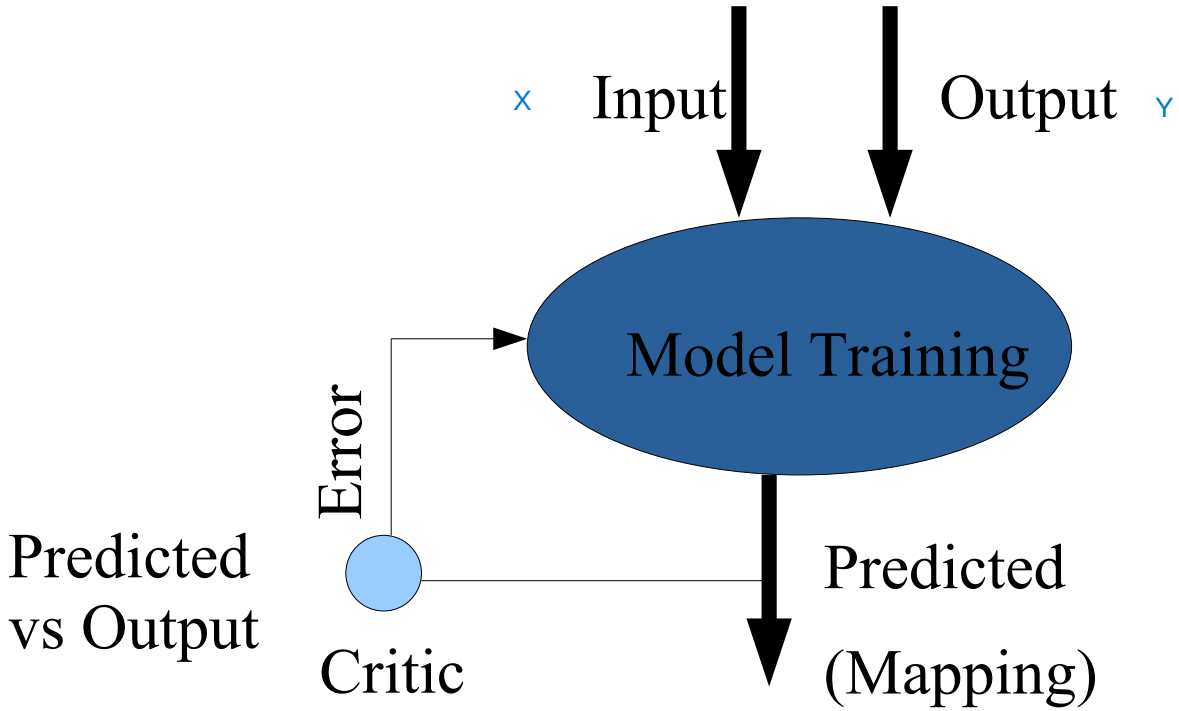
CM: 07/12/23

Introduction

- You have **input variables** (X) **and an output** variable (Y) and you use model to learn the **mapping function from the input to the output**.

$$Y = f(X)$$

- When you have new input data (X) that you can **predict** the output variables (Y) for that data.



Model : Given algorithm (the form of the function)

Train : Search the optimal parameters of the function

y-y2= Error function

Introduction

Supervised Learning

Regression

y 's are real values

The output variable is a real value such as “dollars” or “weight”

- Linear regression
- Polynomial Regression
- Ridge Regression
- Lasso Regression
- Support vector Regression
- Trees Regression
- k-Nearest Neighbors
- Random Forest Regression
- **Neural Networks**

Classification

The output variable is an integer or a category, such as “red” or “blue”

- Logistic Regression
- Support vector Machines
- Decision Trees
- k-Nearest Neighbors
- Random Forest
- **Neural Networks**

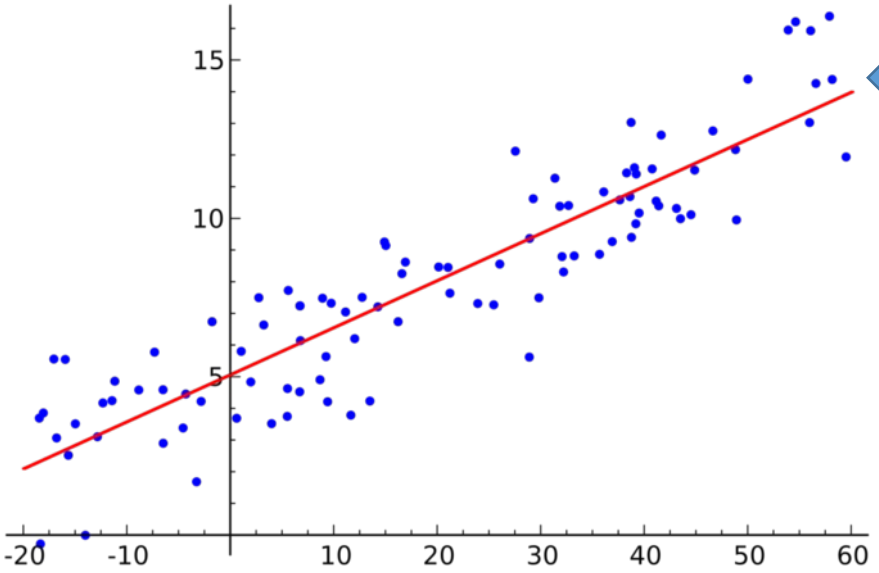
sklearn

(library containing those methods)

Deep Learning (*Tensorflow*)

Linear Regression

Linear regression shows the **linear relationship** between the independent variable **(X-axis)** and the dependent variable **(Y-axis)**



Best fit straight line to the data points

y = target, x = variables

$y = a\,x + b$ (Model)

Choix du model dont on va faire du training pour determiner les meilleurs parameter qui sont a et b.
Pour ce probleme on va definir une fonction d'error,

How to find the best values for a and b ?

X('feature') | Y('target')

Linear Regression

Training :

Cost Function : $$J=\frac{1}{n}\sum_{i=1}^{n}\left(y_{predted}^{(i)}-y_{data}^{(i)}\right)^2$$

$$;\ y_{predted}^{(i)}=a\,x^{(i)}+b$$

ligne no i

predite par le model

Mean Squared Error(MSE) function

Minimize J (training) $$J(a,b)=\frac{1}{n}\sum_{i=1}^{n}\left(a\,x^{(i)}+b-y^{(i)}\right)^2$$

$$\frac{\partial J}{\partial a}=0 \text{ and } \frac{\partial J}{\partial b}=0$$

$$a=\frac{\sum_{i=1}^{n}\left(x^{(i)}-\bar{x}\right)\left(y^{(i)}-\bar{y}\right)}{\sum_{i=1}^{n}\left(x^{(i)}-\bar{x}\right)^2}$$

$$b=\bar{y}-a\,\bar{x}$$

Linear Regression

Matrix representation :

$$y^{(i)}_{predted} = a\,x^{(i)} + b$$

$$\begin{pmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \dots \\ y^{(n)} \end{pmatrix} = \begin{pmatrix} x^{(1)} & 1 \\ x^{(2)} & 1 \\ x^{(3)} & 1 \\ \dots & 1 \\ x^{(n)} & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

$$\boxed{Y = X\,\Theta}$$

Generalization to m variables: $y = a_1 x_1 + a_2 x_2 + \dots + a_m x_m + b$

$$\begin{pmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \dots \\ y^{(n)} \end{pmatrix} = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \dots & 1 \\ x_1^{(2)} & x_2^{(2)} & \dots & 1 \\ x_1^{(3)} & x_2^{(3)} & \dots & 1 \\ \dots & \dots & \dots & 1 \\ x_1^{(n)} & x_n^{(n)} & \dots & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \\ b \end{pmatrix}$$

$$\boxed{Y = X\,\Theta}$$

$\Theta = column\,;\ X = \text{matrix}\ n.(m+1)$

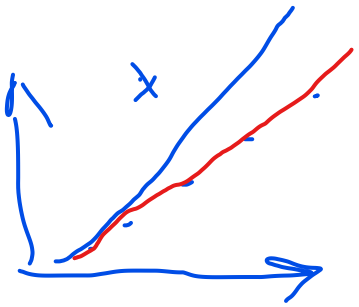
$\boldsymbol{x}^{(i)} = ligne\ \#\,i\,;\ y = colomn$

$$J(\Theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \sum_{j=1}^{m}\Theta_j\,x_j^{(i)}\right)^2$$

$$J(\Theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{x}^{(i)}\Theta\right)^2$$

$$a_{(k)} = \frac{\sum_{i=1}^{n}\left(x_k^{(i)} - \overline{x_k}\right)\left(y^{(i)} - \bar{y}\right)}{\sum_{i=1}^{n}\left(x_k^{(i)} - \overline{x_k}\right)^2}$$

$$b = \bar{y} - \left(a_1\overline{x_1} + a_2\overline{x_2} + \dots + a_n\overline{x_n}\right)$$



Avantages: pas de iteration, facile a implementer | Inconveniences: J donne bcp de poid a des valeurs qui ne sont tres loin des autres

Theta(i+1) = Theta(i)+ Alpha*grad(J)

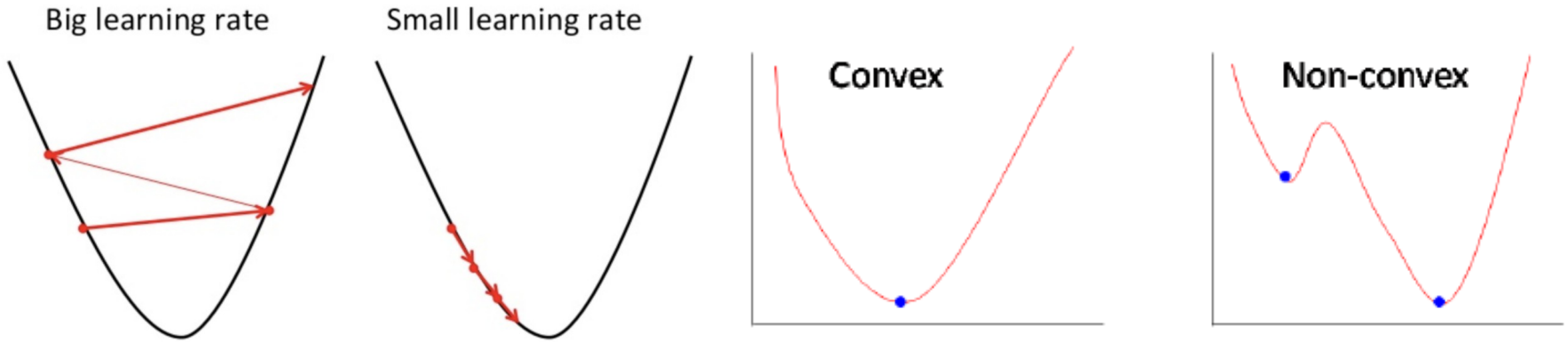
Linear Regression

Gradient Descent : If cost fonction is not a MSE one.

$$\text{Example} : J = \frac{1}{n}\sum_{i=1}^{n} \left| y_{predted}^{(i)} - y_{data}^{(i)} \right|$$

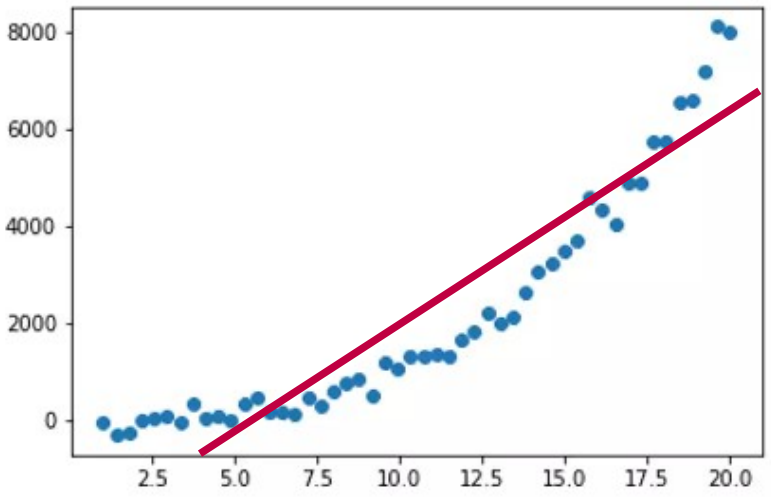
• Starts with random values. An iterative process then goes in the direction towards minimizing the error.

• Uses a **learning rate** (**alpha**) parameter that determines the size of the improvement step to take on each iteration of the procedure.



The learning rate is often varied during training, in accordance to a learning rate schedule or by using an adaptive learning rate

Polynomial Regression



$y = a + b\,x$ Linear, bad Model!

$$y = A\,x^2 + B\,x + C$$

Model : polynomial

a different model function has been choosen

Hyperparameter : Polynomial degree

Training : fit A, B and C

parameter : a, b,c
hyperparameter: choisi avec le model, elle est fixer et ne varie plus lors du training.

New variables : $x_1 = x^2$ and $x_2 = x$

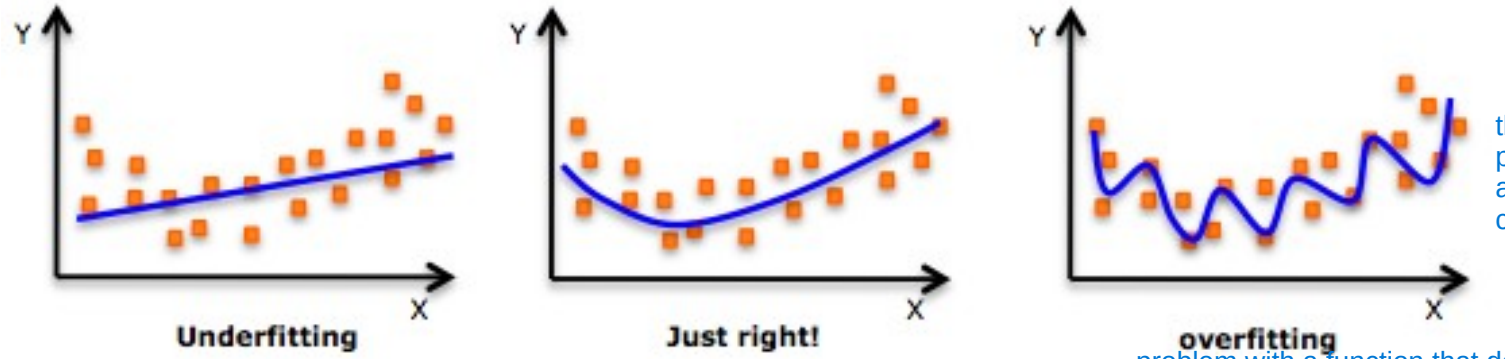
$$\Rightarrow y = A\,x_1 + B\,x_2 + C \qquad \Rightarrow y = a_1\,x_1 + a_2\,x_2 + b$$

x | x2 | y : on a ajouté un feature en plus, nouvelle variable, la dimension du matrice a augmenter

A polynomial of degree 2 with one variable is a **Linear** function with 2 variables.

A polynomial Model is a Linear model with more variables.

Regularization : Ridge and Lasso models



- Regularization methods seek to both minimize the sum of the squared error of the model on the training data but also to reduce the complexity of the model.
- These methods are effective to use when there is **collinearity in your input values** and ordinary **least squares would overfit the training data**.
- Two popular examples of regularization procedures for linear regression are:
  - Lasso Regression
  - Ridge Regression

Regularization : Ridge and Lasso models

- Lasso Regression (called **L1** regularization) :

$$J(\Theta)=\frac{1}{n}\left[\sum_{i=1}^{n}\left(y^{(i)}-\sum_{j=1}^{m}\Theta_j x_j^{(i)}\right)^2+\lambda\sum_{i=1}^{m}|\Theta_i|\right]$$

Minimum de terme de polynome non-nul le plus petit possible

ajout de contrainte

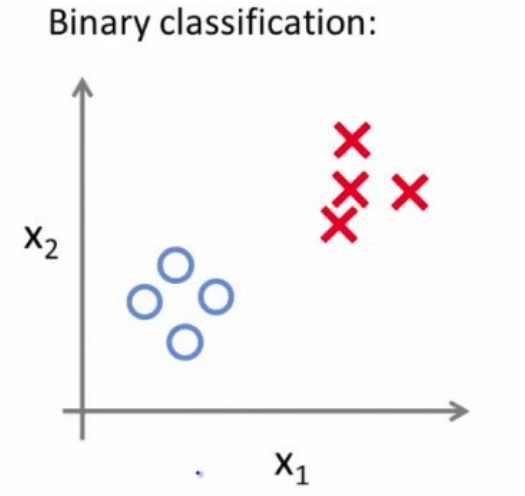
Lambda est un hyperparameter

- Ridge Regression (called **L2** regularization):

$$J(\Theta)=\frac{1}{n}\left[\sum_{i=1}^{n}\left(y^{(i)}-\sum_{j=1}^{m}\Theta_j x_j^{(i)}\right)^2+\lambda\sum_{i=1}^{m}(\Theta_i)^2\right]$$

x1
x2
y={0,1}

Logistic regression



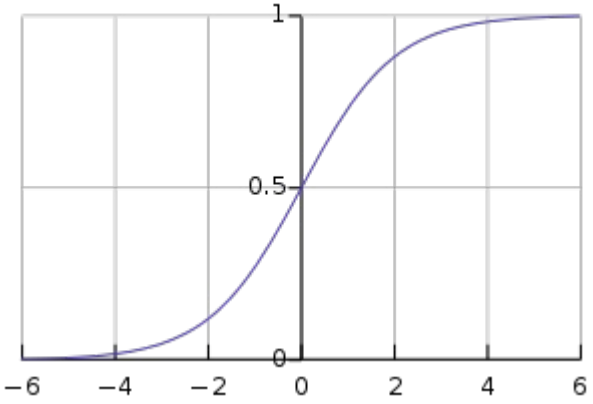
x_1, x_2 : variables

y : integer target (0 for red or 1 for blue)

Logistic regression estimate a probability

$$p = h_\Theta(X) = \sigma(X\Theta)$$

$$\sigma(t) = \frac{1}{1+\exp(-t)}$$: Logistic function



$$y = \begin{cases} 0 \; if \; p < 0.5 \\ 1 \; if \; p \geq 0.5 \end{cases}$$

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{n}\left[y^{(i)}\log p^{(i)} + (1-y^{(i)})\log(1-p^{(i)})\right]$$

entropy

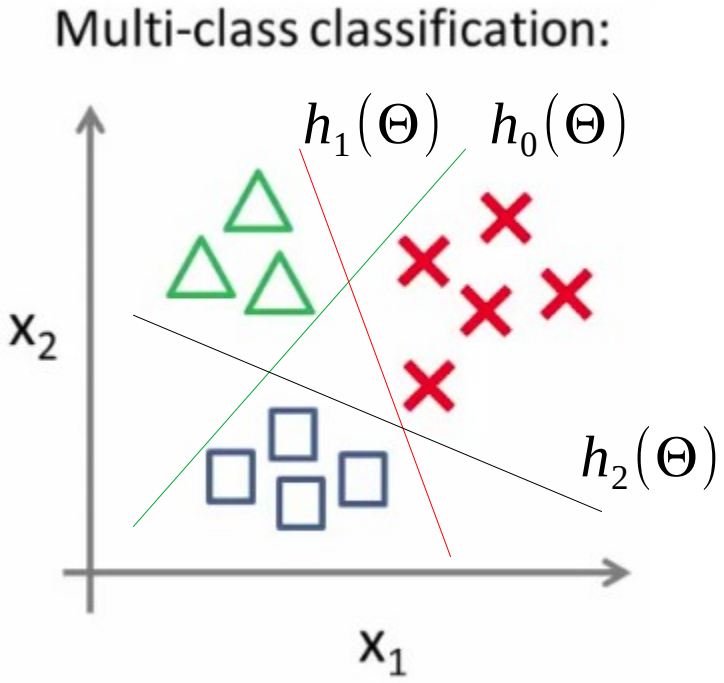
Logistic function = sigmoid function

Cross-Entropy cost

Multi-Class Classification

X : variables

y : integer target : 0, 1, ….N



One-Vs-Rest

One-Vs-Rest

1) Divide the problem into N+1 binary classification problems

2)For each class i : class i=>0 , others =>1 (binary classification)

3) ***Prediction*** for a X : Compute all $h_i(\Theta)$. Take : $max(h_i)$

$$h_0(\Theta), h_1(\Theta), ..., h_N(\Theta)$$

probability to be in class 1
// 2
...

Multi-Class Classification

One-Vs-One :

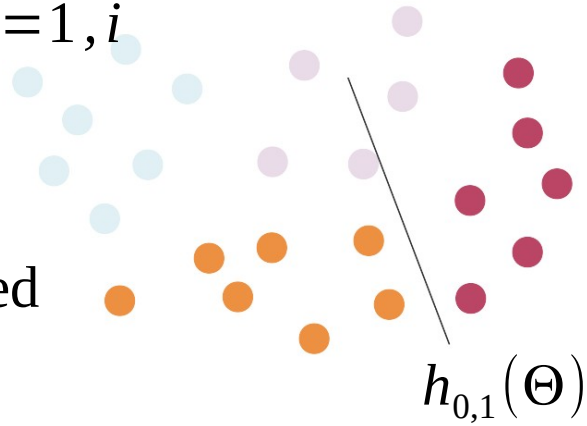
1) Divide the problem into N(N+1)/2 binary classification problems

2) For each (i,j) binary problem : $h_{i,j}(\Theta); i=1,N; j=1,i$

3) ***Prediction*** for a X: Compute all $h_{i,j}(\Theta)$.

Each (i,j) predict a class.

The class which received the most votes is selected

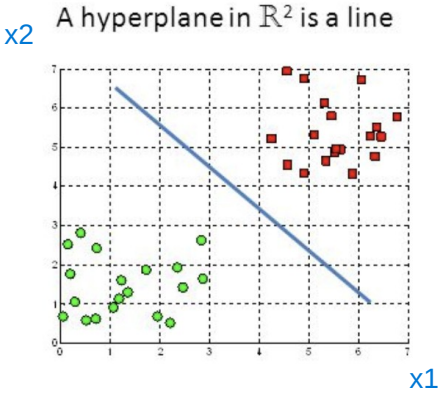


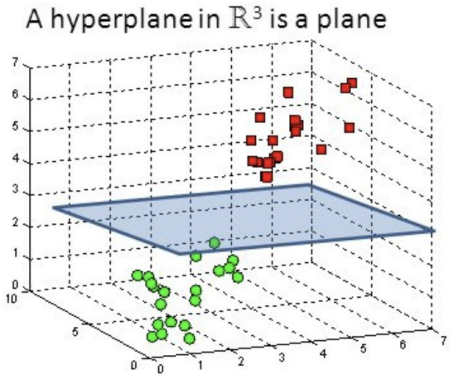
This One-Vs-Rest and One-Vs-One can be used with **any binary classifier** like Logistic Regression, SVM, ... for multi-class classification.

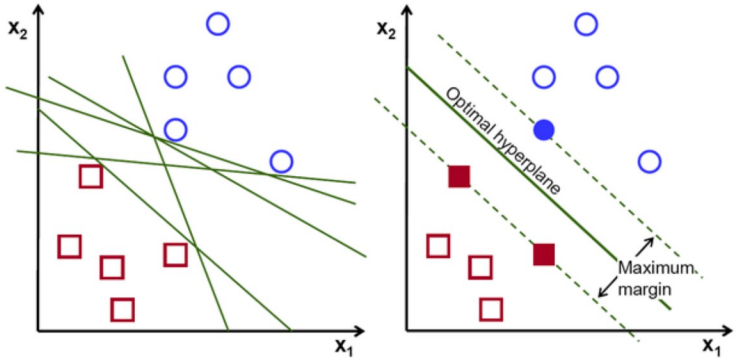
Inherently multi-class : Some classifiers have inherent support for multi-class problems (e.g. k-NN, DecisionTree, RandomForest)

Support Vector Machines (SVM)

trouver une droite/plan qui permet de diviser les deux classes





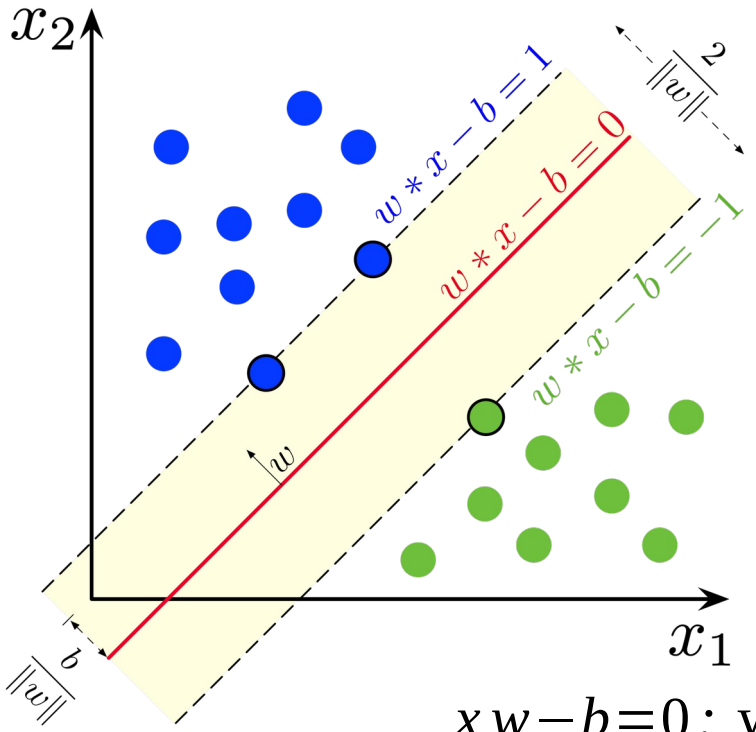


- Objective:
  - find a hyperplane in an m-dimensional space that distinctly classifies the data points
  - find a plane that has the maximum margin, i.e the maximum distance between data points of both classes

- Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence

y={-1,+1} vert//bleu

Support Vector Machines (SVM)



y : integer target ; Binary
-1 for green and 1 for blue

$(x^{(i)}, y^{(i)}); i = 1, n$

We want "maximum-margin hyperplane" that divides the group of points $x^{(i)}$, $y^{(i)}=1$ from the group of points for which $y^{(i)}=-1$

Any hyperline can be written as the set of points x satisfying : $x\,w - b = 0$

$x\,w - b = 0$; where w is a vector normal to the hyperplane

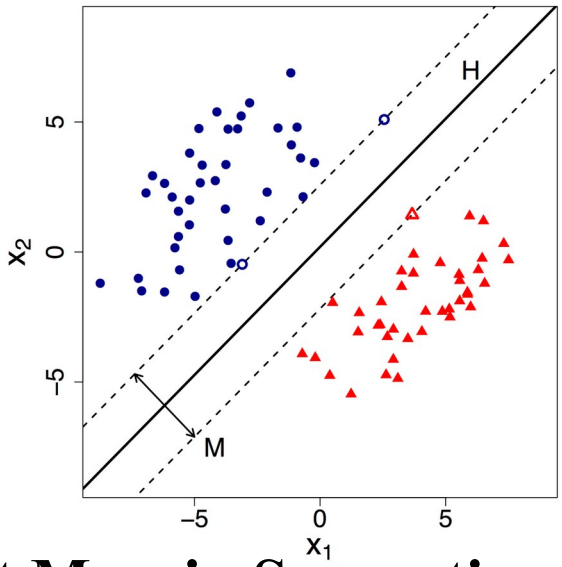
The distance between these two hyperplanes is 2/‖w‖. So to maximize the distance between the planes we want to minimize ‖w‖

Minimize $\|w\|$ subject to $y^{(i)}(x^{(i)} w - b) \geq 1$ for i = 1 , … , n .

Support Vector Machines (SVM)

Minimize $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} max\left(0, 1 - y^{(i)}\left(x^{(i)}w - b\right)\right)$ L1 regularization

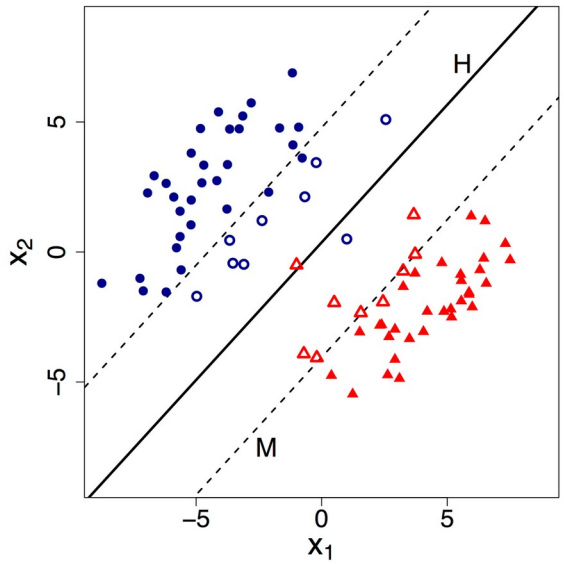
Minimize $\frac{1}{2}\|w\|^2 + C\sum_{i-1}^{n} max\left(0, 1 - y^{(i)}\left(x^{(i)}w - b\right)\right)^2$ L2 regularization



Largest Margin Separating

large values of C

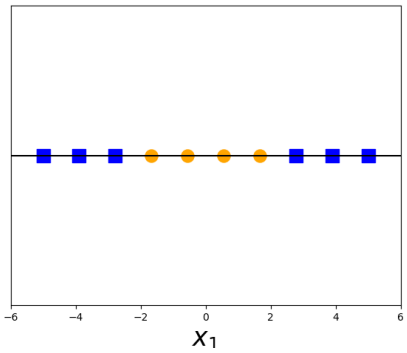
overfitting

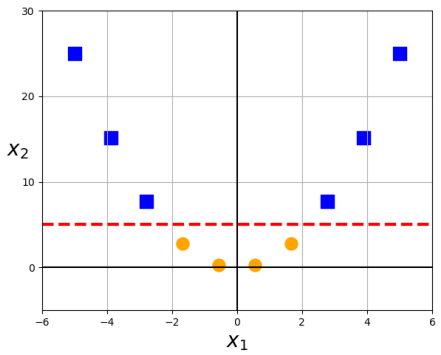


Soft Margin Separating

small values of C

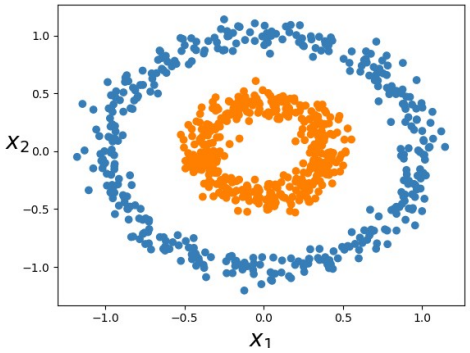
Support Vector Machines (SVM)

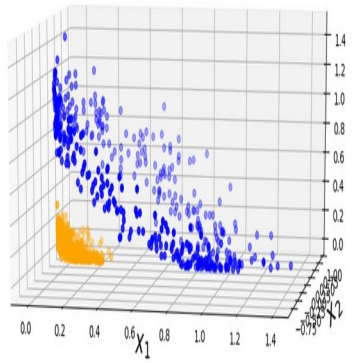




$$\phi(x)=\begin{pmatrix} x \\ x^2 \end{pmatrix}$$

Point in 1st space Point in 2nd space





$$\phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}=\begin{pmatrix} x_1^2 \\ \sqrt{2}\,x_1 x_2 \\ x_2^2 \end{pmatrix}$$

Transformation from a system with m dimensions to a system with m+1 dimensions. However, for many variables, the transformations will be impractical with high computational cost

Support Vector Machines (SVM)

Kernel trick :

The dot product is replaced by a function, called kernel.

The kernel function accepts input in the original lower dimensional space and returns the dot product of the transformer vectors in the higher dimensional space.

Example : $A\begin{pmatrix}a_1\\a_2\end{pmatrix}; B\begin{pmatrix}b_1\\b_2\end{pmatrix}$ $\qquad \phi\begin{pmatrix}x_1\\x_2\end{pmatrix}=\begin{pmatrix}x_1^2\\\sqrt{2}\,x_1x_2\\x_2^2\end{pmatrix}$

$$\phi(A^T).\phi(B)=\begin{pmatrix}a_1^2 & \sqrt{2}a_1a_2 & a_2^2\end{pmatrix}\begin{pmatrix}b_1^2\\\sqrt{2}b_1b_2\\b_2^2\end{pmatrix}=(a_1b_1+a_2b_2)^2=(A^T.B)^2$$

$$(A,B)=>K(A,B)=(A^T.B)^2$$

The transformation not needed.

The dot product in lower dimensional space is replaced by a the Kernel function in the lower dimensional space

Support Vector Machines (SVM)

Non linear SVM :

Kernel trick : $A^T.B \rightarrow K(A,B)$

Kernel: fonction qui permet de calculer les fonctions sans passer au dimension supperieur,(pas de creation de nouvelle colonne) calcule directe, gagner dumemoire et du temps de calcul

Polynomial : $K(A,B)=(\gamma A^T.B+r)^d$

Radial(RBF) : $K(A,B)=e^{-\gamma(\vec{AB})^2}$

Sigmoid : $K(A,A)=\tanh(\kappa A^T.B+r)$

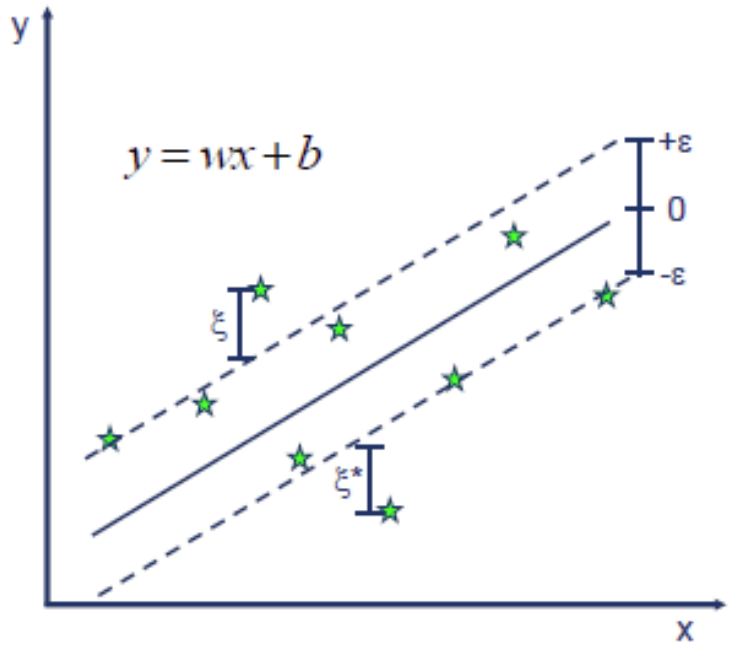
Linear : $K(A,B)=A^T.B$

SVC Hyperparameters

- Type of kernel (linear, polynomial, RBF; etc.)
- Parameters of the kernel
- Type of regularization
- C: regularization strength

crée 2 droites pour avoir le plus grand nombre de points a l'interieur

Support Vector Regression (SVR)



- Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

- Constraints:

$$y_i - wx_i - b \le \varepsilon + \xi_i$$

$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \ge 0$$

Support Vector Regression tries to fit the best line within a predefined error value

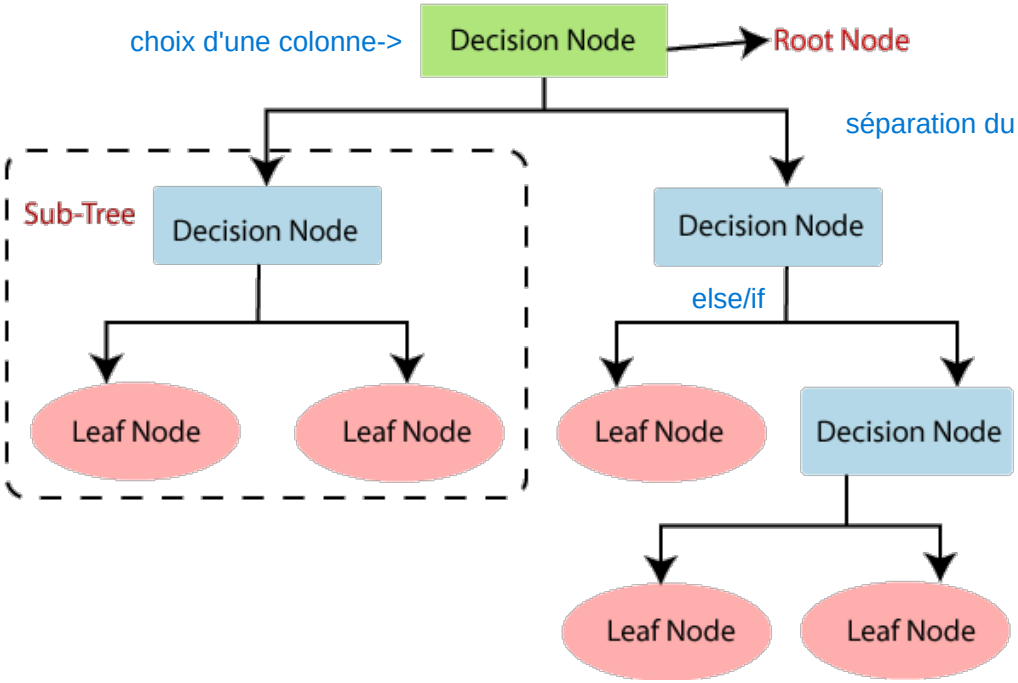
Minimize error, maximizes the margin, keeping in mind that part of the error is tolerated.

(voir: pdf en plus pour demo et explication)

Decision trees

• Can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.

• Tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.



lorsque nous ne pouvons plus séparer les colonnes(tout les val sont identique), ou bien on avait fixer une limite de séparation. On s'arrete.

Decision trees

Hyperparameters :

- Criterion: function to measure the quality of a split

$$GiniIndex = 1 - \sum_j p_j^2$$

$$Entropy = -\sum_j p_j \cdot log_2 \cdot p_j$$

Regression : STD reduction

décision de séparation en ragardant la probabilité

déviation-standrd

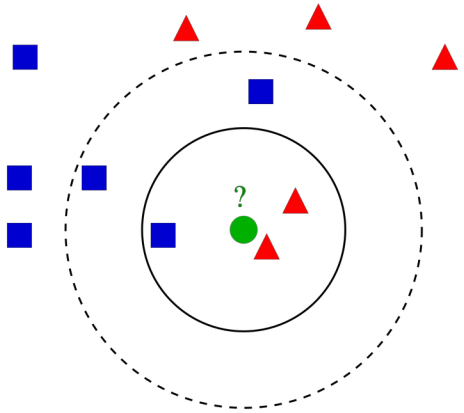
- Max Tree depth: maximum depth of your model. Maximum depth refers to the length of the longest path from a root to a leaf

- Min Splitting of nodes: minimum number of samples (i.e. training inputs) required to split an internal node

- Min Splitting of leaves: minimum number of samples (i.e. training inputs) required to be at a leaf node

K-Nearest Neighbors (KNN)

- The k-Nearest neighbors (k-NN) classifier is probably the simplest non-trivial classifier. Given a training dataset we predict the class of an instance by taking the nearest instances according to some distance metric and predict class.
- The number of neighbors we will determine with cross validation

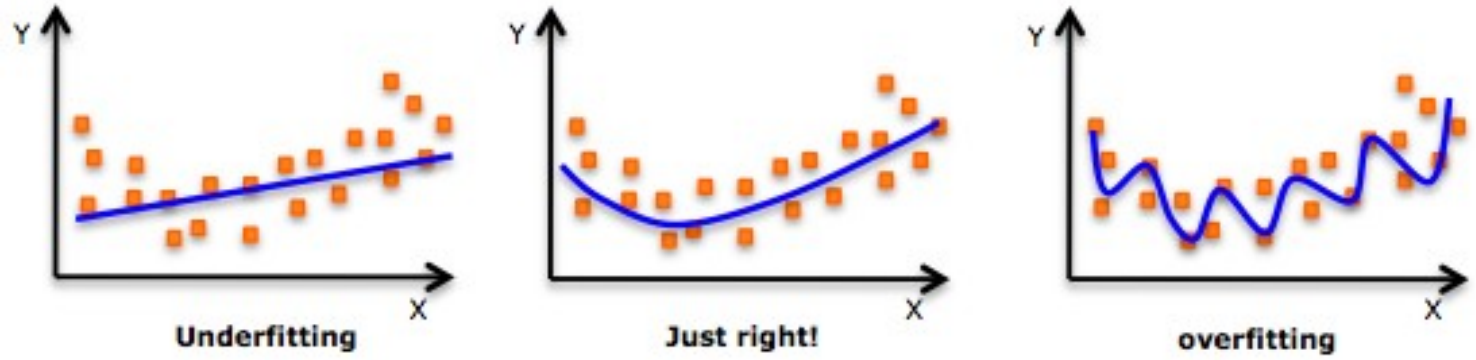


soit 2 class: rouge/bleu
on regarde ce qu'il y a autour du point vert pour le classifier
->le + grd # de voisin d'1 class qu'il y a autour> le point est de ce class

- Pros
  - **Training phase** much **faster** compared to other classification algorithms
  - **Can be used for regression**: the output value for the object is computed by the average of k closest neighbors value.
- Cons
  - It requires **large memory** for storing the entire training dataset for prediction
  - It is **not suitable for large dimensional data**.

method tres rapide
pas de training ici
pas de parameter
on a besoin de tout les données

Over-fitting



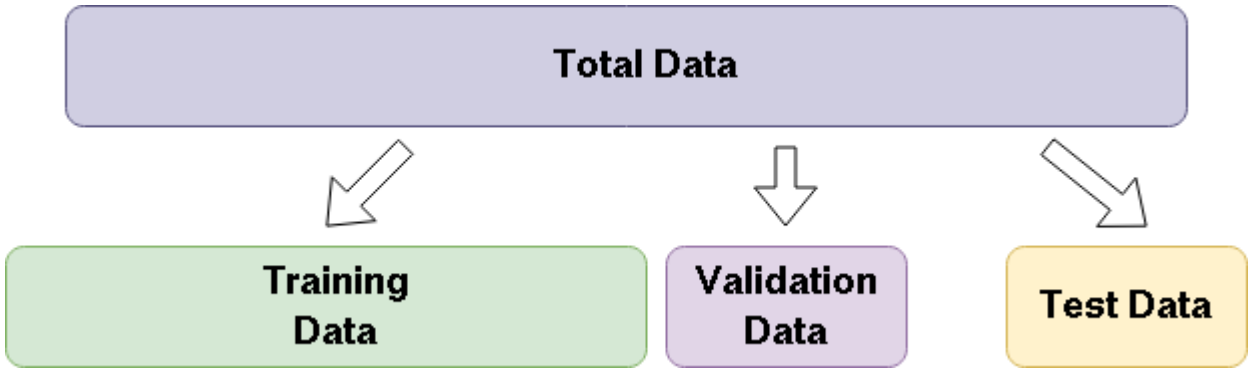
Methods to avoid Over-fitting, Commonly used methodologies :

- **Regularization (**see regression with Ridge & Lasso**)**
- **Early Stopping**
- **Cross-Validation**
- **Pruning** : Reduce overfitting by creating smaller trees.

Over-fitting

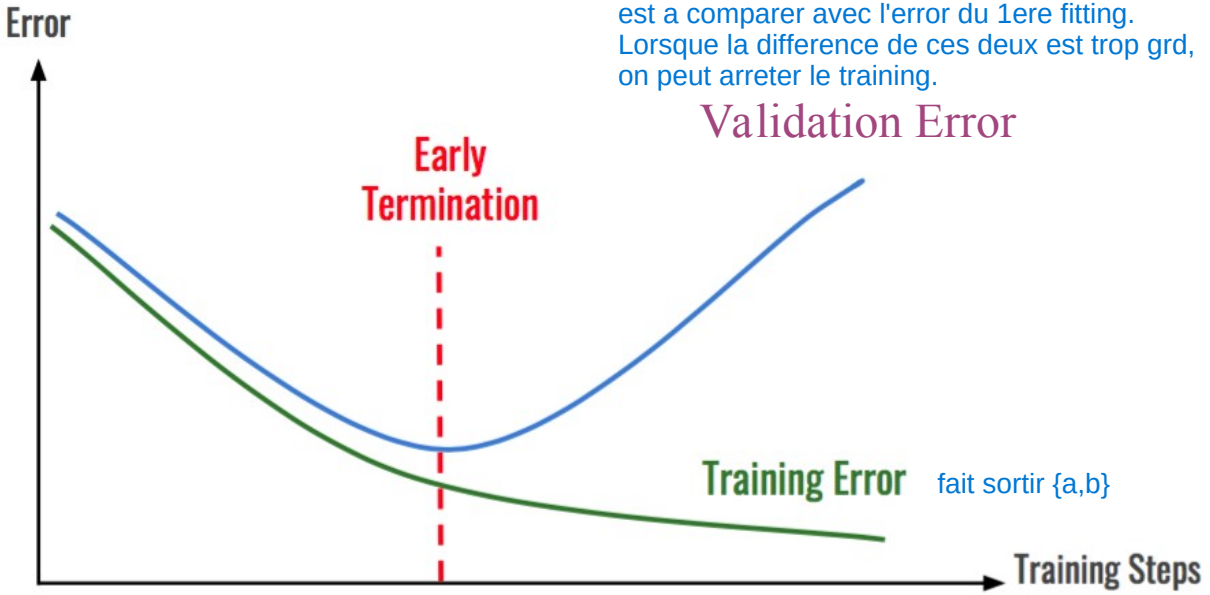
Early Stopping

séparation des datas en 3 groupes



test du training sur une autre parti
on obtient une autre(2ieme) fitting dont l'erreur est a comparer avec l'error du 1ere fitting.
Lorsque la difference de ces deux est trop grd, on peut arreter le training.

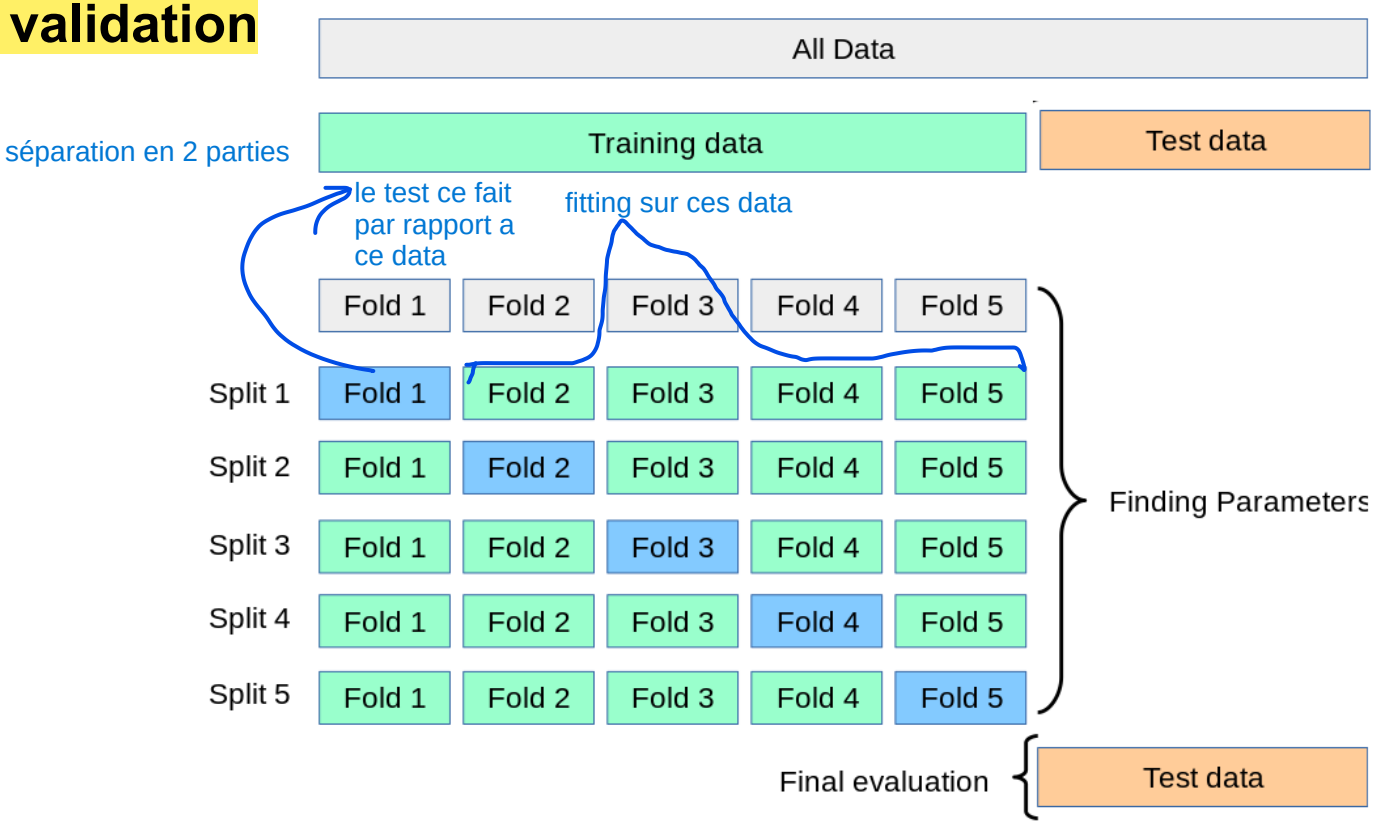
This technique does not generally work well for cases when we don't have a **large** datasets.



fait sortir {a,b}

Over-fitting

K-Fold cross validation



Every fold gets chance to appears in the training set (k-1) times.

Compute the average errors from those of k partitions.

Over-fitting

Pruning Decision Trees : Reduce over-fitting by creating smaller trees.

1. Pre pruning techniques

Pre pruning is the stopping the growth of decision tree on an early stage.
Search those parameters and choose the optimum values that gives better performance on **test data**.

max_depth: maximum depth of decision tree
min_samples_split: The minimum number of samples required to split an internal node
min_samples_leaf: The minimum number of samples required to be at a leaf node.

2. Post pruning techniques

Limit the growth of trees by setting constrains.

Minimize : $Cost + \alpha|T|$; where T is the number of leaves of the tree

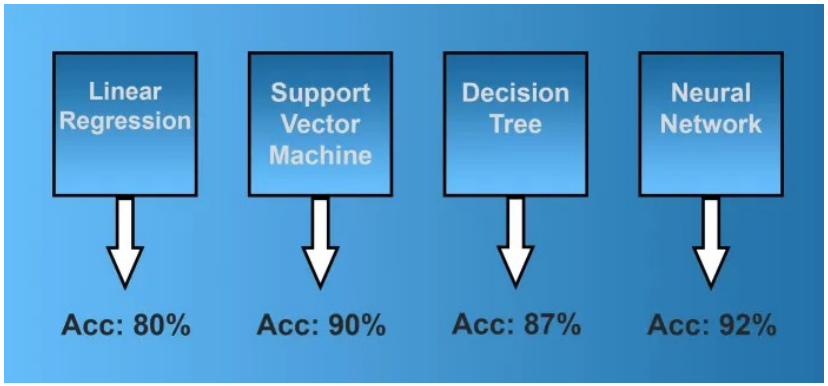
Find the alpha value for that gives better performance on **test data**..

automatique -> **Pruning should be used with cross-validation**

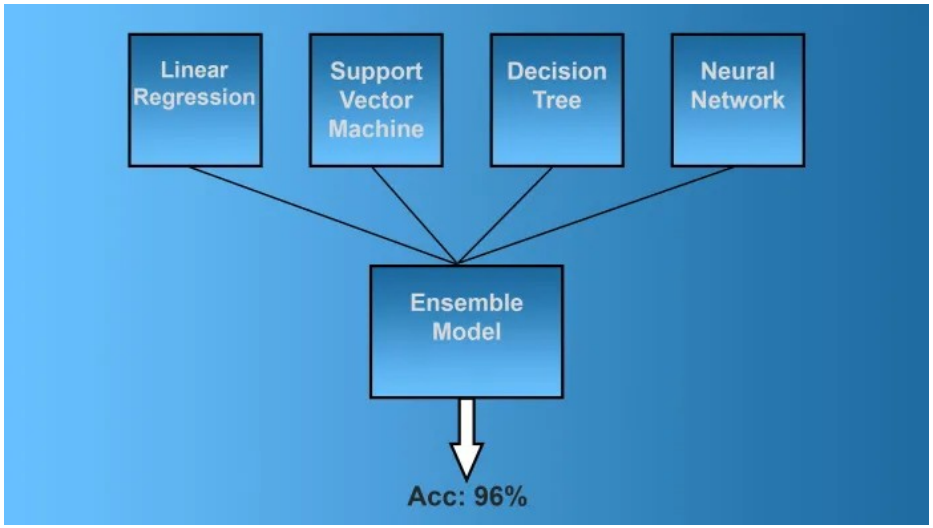
Ensemble learning

Ensemble learning is a general **meta approach** to machine learning that seeks **better predictive** performance by **combining** the predictions **from multiple models**.

faire un mélange de plusieurs method



accuracy



Each model might perform well on some data and less accurately on others.

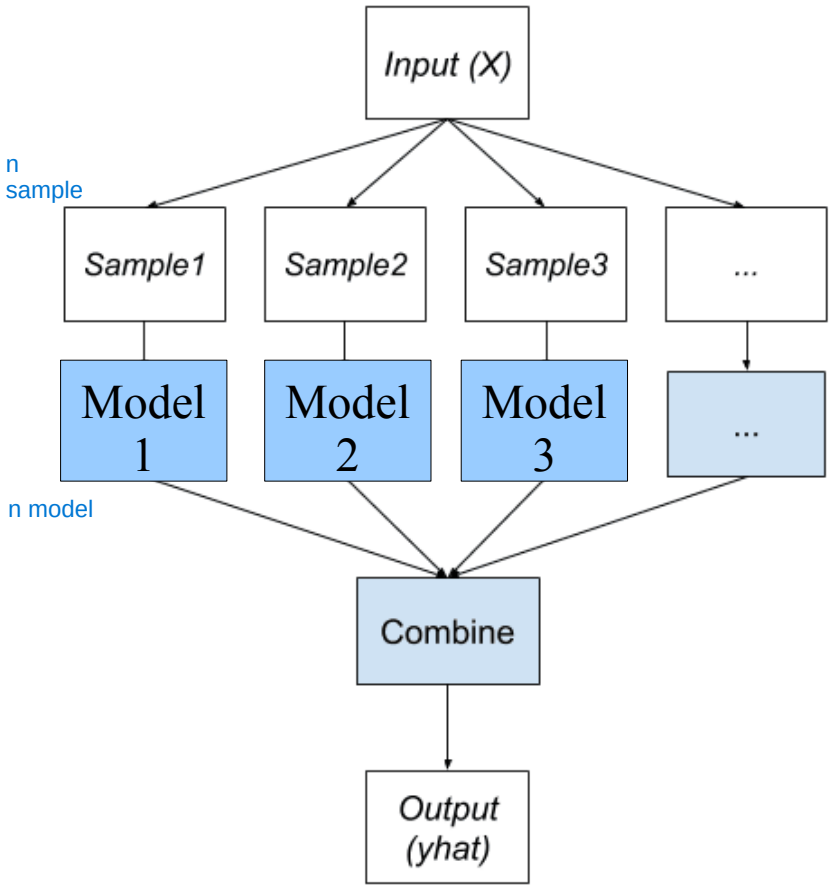
When you combine all them, they cancel out each other's weaknesses.

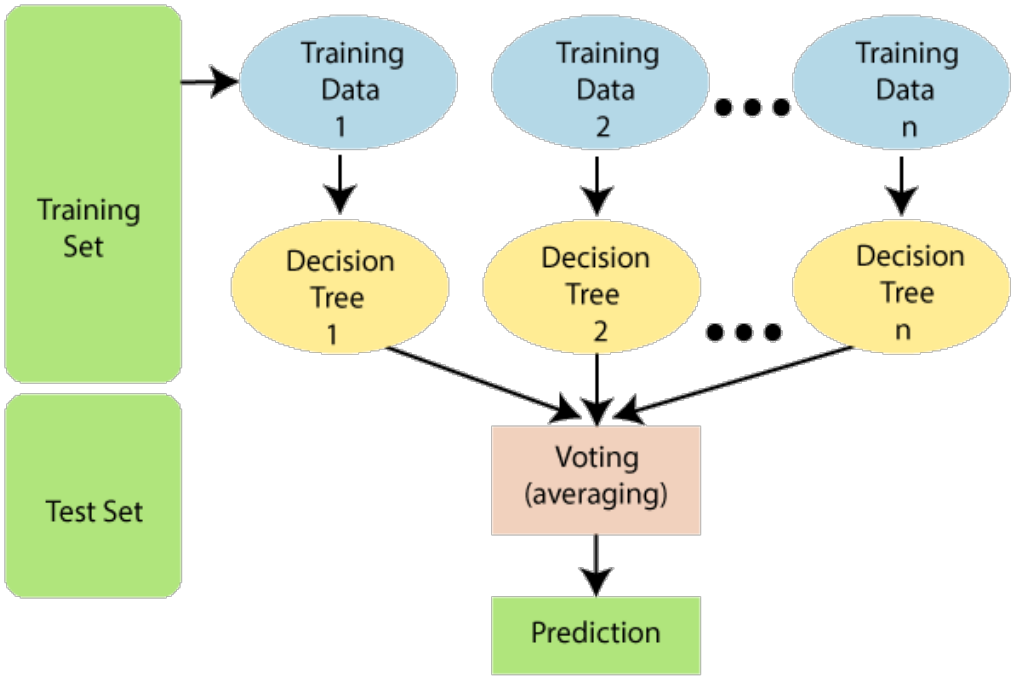
The three main classes :

- Bagging
- Stacking
- Boosting

Ensemble learning

Bagging





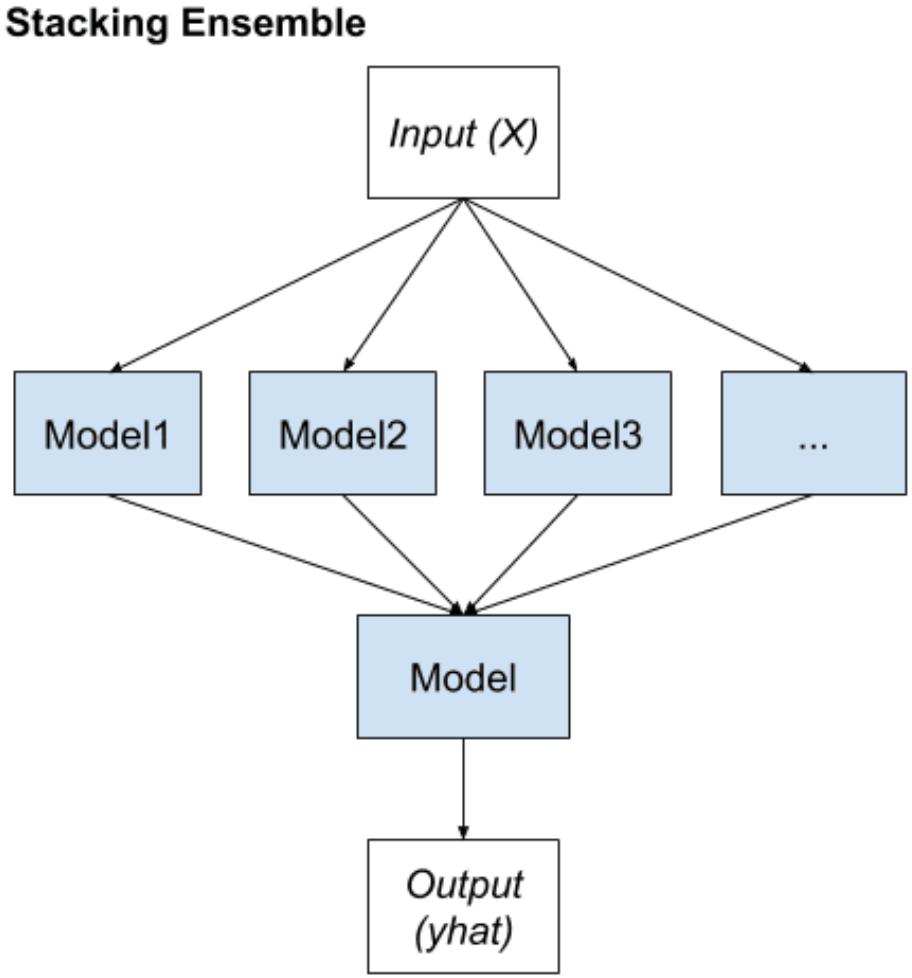
- random *n* training datasets
- Hyperparameters:
  - Same as for the decision trees
  - Number of trees in the forest

hardvoting:
softvoting: proba is taken

Random Forest

Ensemble learning

Stacking



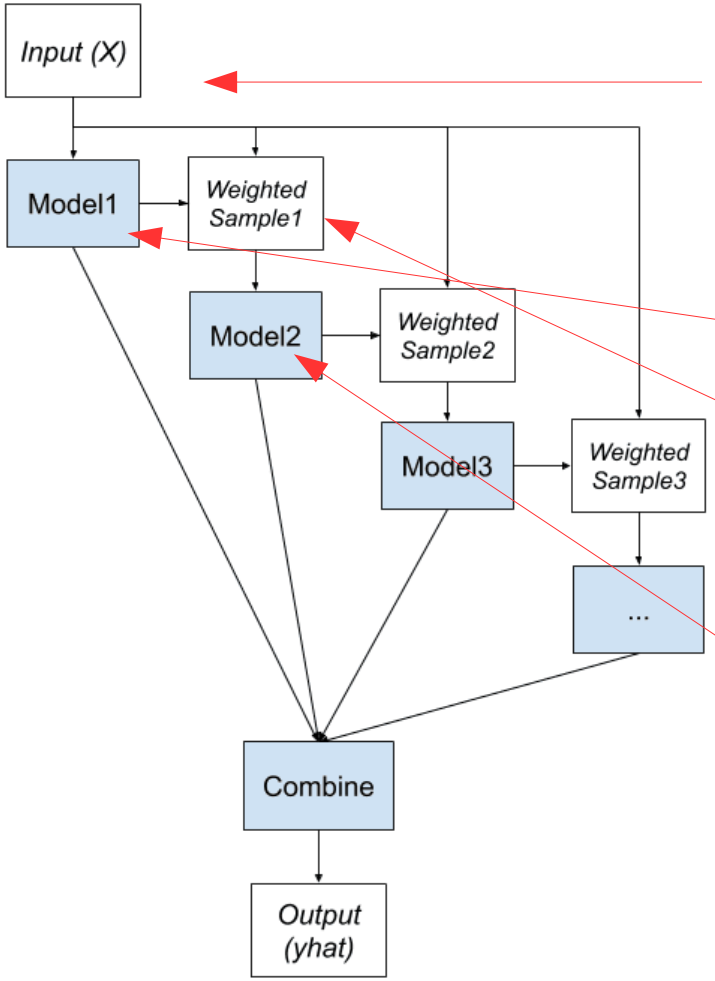
n different model is taken

Voting Ensembles
Weighted Average Ensemble
Super Ensemble

faire un réseau de neuronne

Ensemble learning

Adaptive Boosting (AdaBoost)



1. Assign to every observation X_i, an initial weights value. $\omega_i = 1/n$, where n is the total number of observations.

2. Train a "week" model (often a decision tree).

3. For each observation :
   - 3.1. If predicted incorrectly, ω_i is **in**creased
   - 3.2. If predicted correctly, ω_i is **de**creased

4. Train a new "weak" model where observation with greater wights are given more priority.

5. Repeat steps 3 and 4 until observation perfectly predicted or a preset number of models are trained

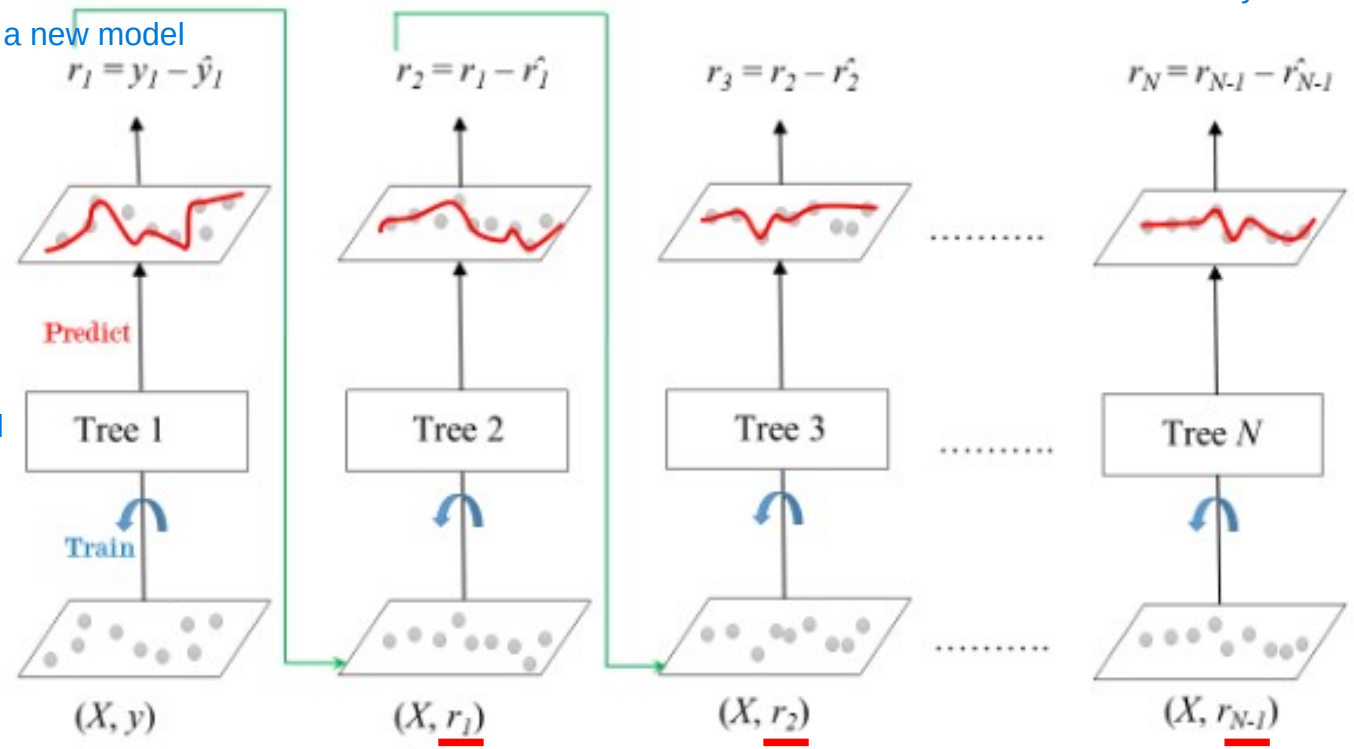
Ensemble learning

Gradient Boosting

$$y(pred) = y_1 + \eta r_1 + \eta r_2 .... + \eta r_N$$

"un sorte de dev de Taylor"

error is given to a new model



a model

Shrinkage : Each tree is shrunk after it is multiplied by the learning rate (η)

hyperparameter given at the start

Gradient boosting is a greedy algorithm and can over-fit a training dataset quickly

Libraries

NB : XGBoost (eXtreme Gradient Boosting) and sklearn's GradientBoost are fundamentally the same. However XGBoost is a lot faster.

seul l'erreur ne suffit pas pour dire que le model est bon! D'autre metrique sont necessaire!

Metrics

Regression metrics : MSE, RMSE, MAE, R^2 (R-Squared)

Mean Squared Error (**MSE**)

$$MSE = \frac{1}{N}\sum_{j=1}^{N}(y_j - \check{y}_j)^2$$

Root Mean Squared Error (**RMSE**)

$$RMSE = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(y_j - \check{y}_j)^2}$$

- It's differentiable, so it can be optimized better.
- Due to the squaring factor, it's fundamentally more sensitive to outliers than other metrics.

Mean Absolute Error (**MAE**)

$$MAE = \frac{1}{N}\sum_{j=1}^{N}|y_j - \check{y}_j|$$

- MAE is non-differentiable as opposed to MSE, which is differentiable.
- It's more robust towards outliers than MSE, since it doesn't exaggerate errors.

R^2 Coefficient of determination

$$= 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}.$$

- If MSE is small => $R^2 \sim 1$ (Ideal),
- If MSE is high => $R^2 \sim 0$
- The range of R^2 is not (0,1). It's actually $(-\infty,1)$

if R<0: inversed correlation

Metrics

Classification metrics

- Accuracy
- Log loss
- Confusion Matrix (not a metric but fundamental to others)
- Precision and Recall
- F1-score

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

$$LogarithmicLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * \log(p_{ij})$$

voir l'entropie

y_{ij} whether sample i belongs to class j or not

p_{ij} : the probability of sample i belonging to class j

Confusion Matrix

dim = (2x2)

		Predicted class: P	Predicted class: N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

une fit qui est bon suggere une matrice diagonal

Metrics

Confusion Matrix

n=165

Actual Class \ Predicted class	P	N
P	50	10
N	5	100

Actual Class \ Predicted class	P	N
P	True Positives (TP)	False Negatives (FN)
N	False Positives (FP)	True Negatives (TN)

True Positives : TP
True Negatives : TN
False Positives : FP
False Negatives : FN

$$Accuracy = \frac{TruePositive + TrueNegative}{TotalSample}$$

Precision : $P = \frac{TP}{TP+FP}$

true Positive/ all Positives

total positives predicted

Recall : $R = \frac{TP}{TP+FN}$

all the positives in ground truth

If you try to reduce FP, no direct effect will take place on FN

Metrics

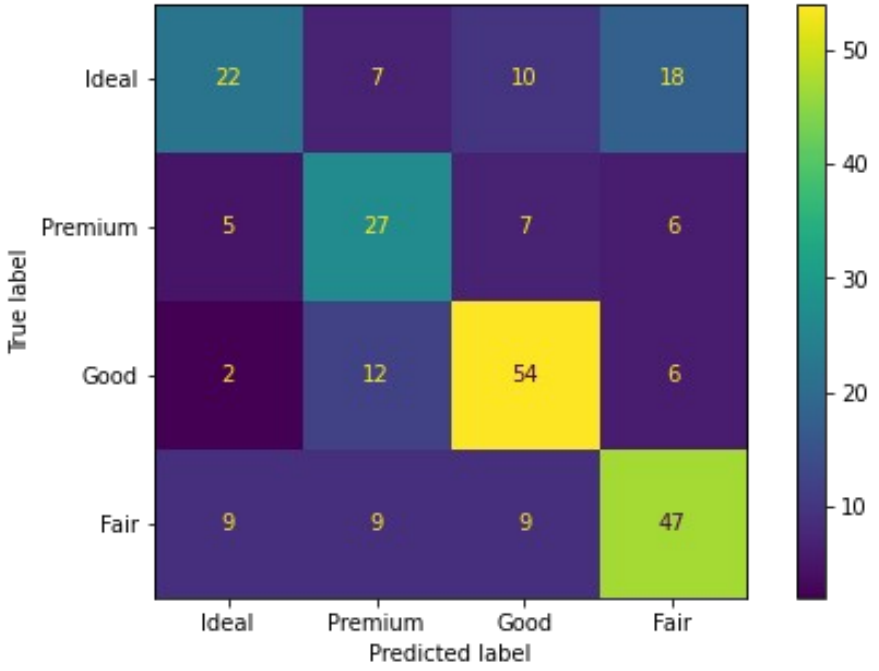
F1-score : score is the harmonic mean of the two :

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$

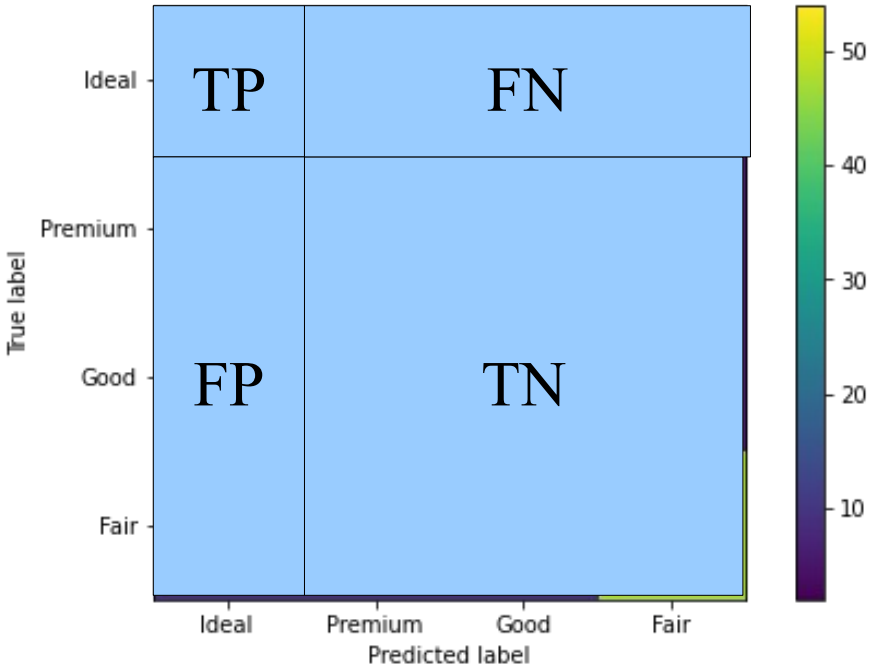
- A high F1 score symbolizes a **high precision** as well as **high recall**.
- It presents a good balance between precision and recall and gives good results on imbalanced classification problems.

Metrics

Multiclass Classification Metrics



For the ideal diamonds



Precision (ideal): 22 / (22 + 5+2+9) = 0.579

Metrics

Multiclass Classification Metrics

	precision	recall	f1-score	support
Ideal	0.58	0.39	0.46	57
Premium	0.49	0.60	0.54	45
Good	0.68	0.73	0.70	74
Fair	0.61	0.64	0.62	74
accuracy			0.60	250
macro avg	0.59	0.59	0.58	250
weighted avg	0.60	0.60	0.59	250