CM1

L'année dernière vous avez fait les modèles traditionnel et vous avez vu les données et les chaines de traitements.
En M2 vous verrez du Deep Learning, la chaine de traitement de change pas avec celle de M1.
La seule grosse différence ça va être la complexité des modèles.
Il existe des modèles pour tout mais toujours spécialisé dans un domaine.

Important

Le deep learning consomme énormément de ressource donc éviter d'utiliser ça à tout va. Pour cette raison vous ne lancerez pas deux fois la même expérimentation, stocker les résultats de vos modèles vos données etc.. C'est important aussi sur un principe de travail d'équipe, pour que quand un nouveau collaborateur arrives il voit direct toutes les anciennes expérimentation, les différents types de données qui ont été utilisé etc...

Stage

Chercher le stage très vite

Pour la lettre de motivation ne la faites pas générer, elle doit être bien faites et adapté à chaque entreprise / labo. Si possible essayer de rendre votre github attrayant.
Les boites actuellement sont très frileuse et ont assez peur à cause de la situation politique et économique.

Stage de recherche : état d'art, proposition, évaluation, étude bibliographique
Stage de pro : idem + présentation de l'entreprise

Pour les stages académique ils seront proposé, sinon contacter directement les profs dans des labos.

Ne pas hésiter à demander à Todorov ou PROF DE ML pour être représentat de stage en IA.

Thèse:

Obligatoirement financé, soit grâce à

- un financement minestériel (intéressant car vous avez juste à travaillez sur votre thèse) mais très concurrentiel.
- financement par Projet: Européen, ANR (nationaux), Feder, Université

- Cifre : ATTENTION car beaucoup d'entreprise prennes ça pour un ingénieur à bas prix, faire attention aux horaires, salaires etc... Faire attention à qui paie les missions histoire de pas avoir de mauvaise surprise. Lors de la demande deux experts sont contactés, un expert en economie qui s'assure que vous soyez payé et un expert scientifique qui s'assure que vous faisiez du bon travail.

Pour ce qui est de la recherche d'un docteur aujourd'hui sur le plan du travail, ça dépend de l'entreprise. Aussi les entreprises ont un bonus si elle prennent un docteur, elle paies pas leur charges pendant un an (CIR).

Programme

- Descente de gradient : très important à comprendre, pas très compliqué mais fondamental
- Modèles complexes avec des Réseaux de neurones, CNN, AE, VAE, GAN, GNN, Transformers
- Traitement des images : classification, colorisation, génération
- Traitement de texte : classer, générer, traduire
- Faire notre propre ChatGPT

Pour nous aider nous aurons droits à des guides (pas de notebooks).

- Guide pratique des réseaux de neurones : descente de gradient, réseaux de neurones, et queras (dans le monde réelle les gens choisissent PyTorch). On apprend queras car il est un peu plus simple, mais si on arrives à utiliser queras on s'en sortiras avec PyTorch. Ce guide explique en grandes partie comment utiliser queras.
- Guide pratique de ... : VAE, AE, Encoder etc... Et voir ce que l'on peut faire avec. On va voir aussi les GAN et les Transformer dans ce Guide pratique.
- Guide pratique de l'apprentissage profond de données textuelle : ChatGPT, on ne parle pas de comment adapter mon modèle à mes données, car c'est trouvable sur internet extrêmement facilement, donc faites le vous même, pas d'intérêt de vous l'expliquer.
  Les guides ont été un peu épurés pour les rendre plus facile à lire. Ils sont pensé pour données des astuces, des remarques pertinentes et des conseils, pas pour nous noyer dans de la doc.

Evaluation

Un projet en groupe + un test individuel

Pour le projet : un document court + code + vidéo

Pour le projet on hésite entre la classification d'images, et des Transformers. Potentiellement ce sera un modèle CLIP. Vu que c'est un peu compliqué il hésite encore, mais ce type de projet

multimodale est très intéressant et ça serait bien. 8 - 10 pages max pour le document.
Taille des groupes non défini.

Test individuel sous forme de QCM ou de petites question pour vérifier que tous le monde ait bien travaillé.

Attention :

- Non respect des consignes = -4 pts
- Pas de LLM, sinon = 0

Cours

L'attention c'est quel est le poids que le modèle à mis sur tel ou tel phrase.
Pour commencer nous allons faire un petit rappel :

Exemple

ID	Age	IMC	cholesterol	poids	malade
1	25	22.5	180	68	NULL
2	30	24.0	190	72	NULL
3	28	23.5	175	70	NULL
4	45	27	220	82	1
5	50	28.5	242300	85	1
6	47	29.0	250	84	1
7	52	30.0	210	88	1
8	49	26.5	185	80	1
9	33	23.0	185	73	NULL
10	120	60	800	150	1

Lecture du parseur

```
import panda as pd
df = pd.read_csv("sante.csv")
= pd.read_json("sante.json")
= pd.read_excel("sante.xls")
= pd.read_sql("select * from table")
```

Première Lecture

Première chose à faire c'est essaye de comprendre ce que l'on a dans les données.
Regarder aussi le retour des données renvoyer par le parseur

```
df.head()
df.info() # Donne combien on a de ligne, colonne, type de données etc...
df.describe() #Donne des stats (ecart type, moyenne, min, max, quantites)
# Le decribe c'est très pratique pour voir les données incohérentes, qui sont
affolantes ou trop différentes
df['Malade'].value.counts() # donnes la distribution des classes,permet de
voir l'équilibrages
```

Par exemple on remarque que la données à la ligne 10 à un age de 120 ans, ce qui est pas forcément pertinent.

Visualiser les données

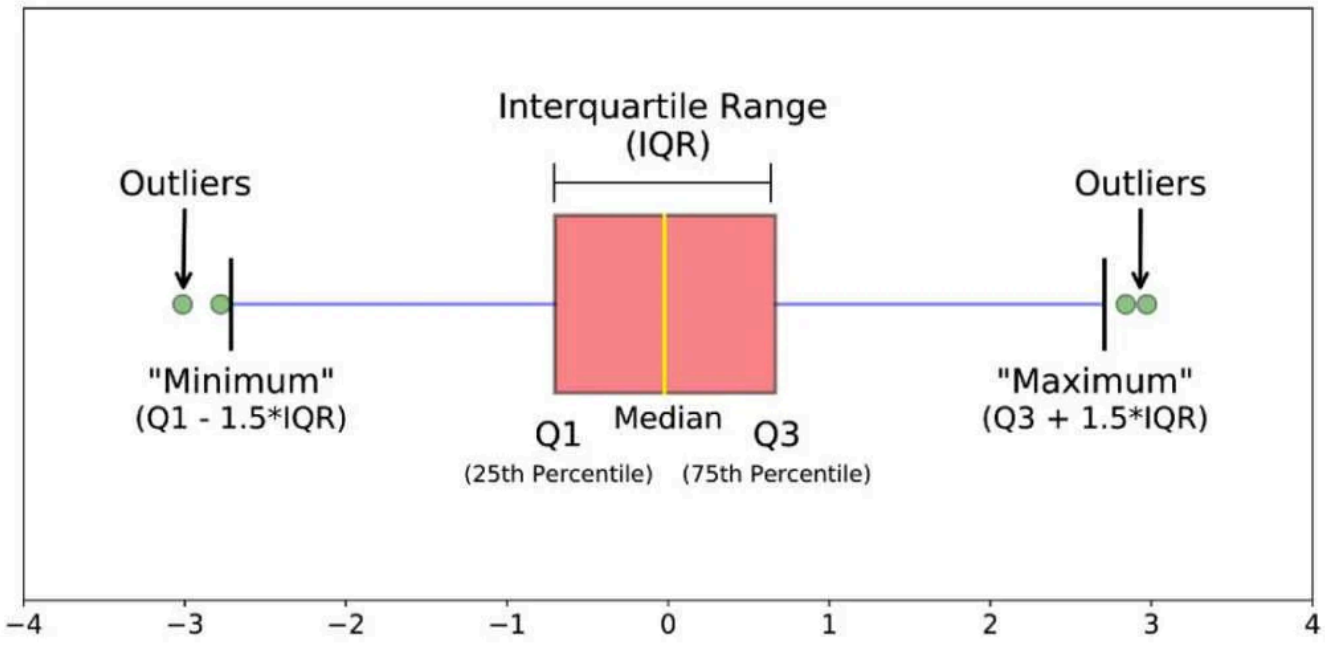
Pour visualiser les données on va utiliser Matplotlib, Seaborn (SNS). Hésiter pas à demander à chatGPT pour ce genre de chose, c'est de la programmation bête, faites ça vite.

Univariés : on va regardé qu'un seul attribut, dans ce cas on va s'intéresser aux histogrammes, boxplot

histogramme:

```
df['Age'].hist(bins:6) #bins c'est en combien je vais découper mes valeurs,
par exemple ici les données seront découpés en 6 zones.
```

boxplot:

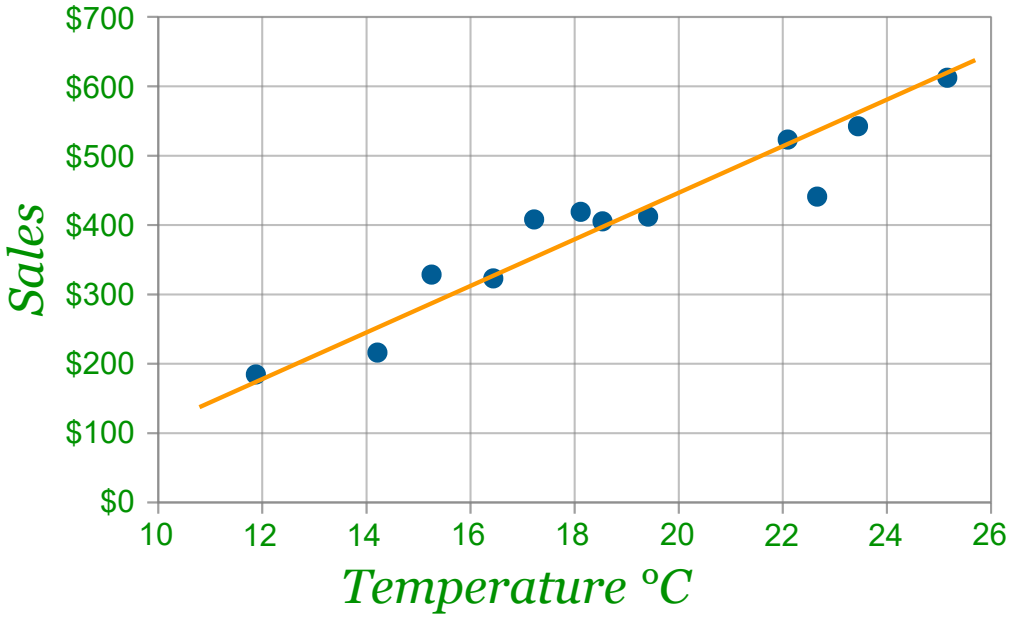


C'est uniquement une fois que les données ont été visualiser et analyser que l'on appelle l'expert des données pour lui faire un rapport sur qu'est ce qui va ou non, pas après.
Il existe des librairies qui font tout, où une fois qu'on lui a donné les données vous font tout automatiquement etc... Mais pour le moment c'est mieux si on le fait à la main.

Bivariés : on va s'intéressé à deux attributs, et ici on va voir le scatter plot

ScatterPlot :

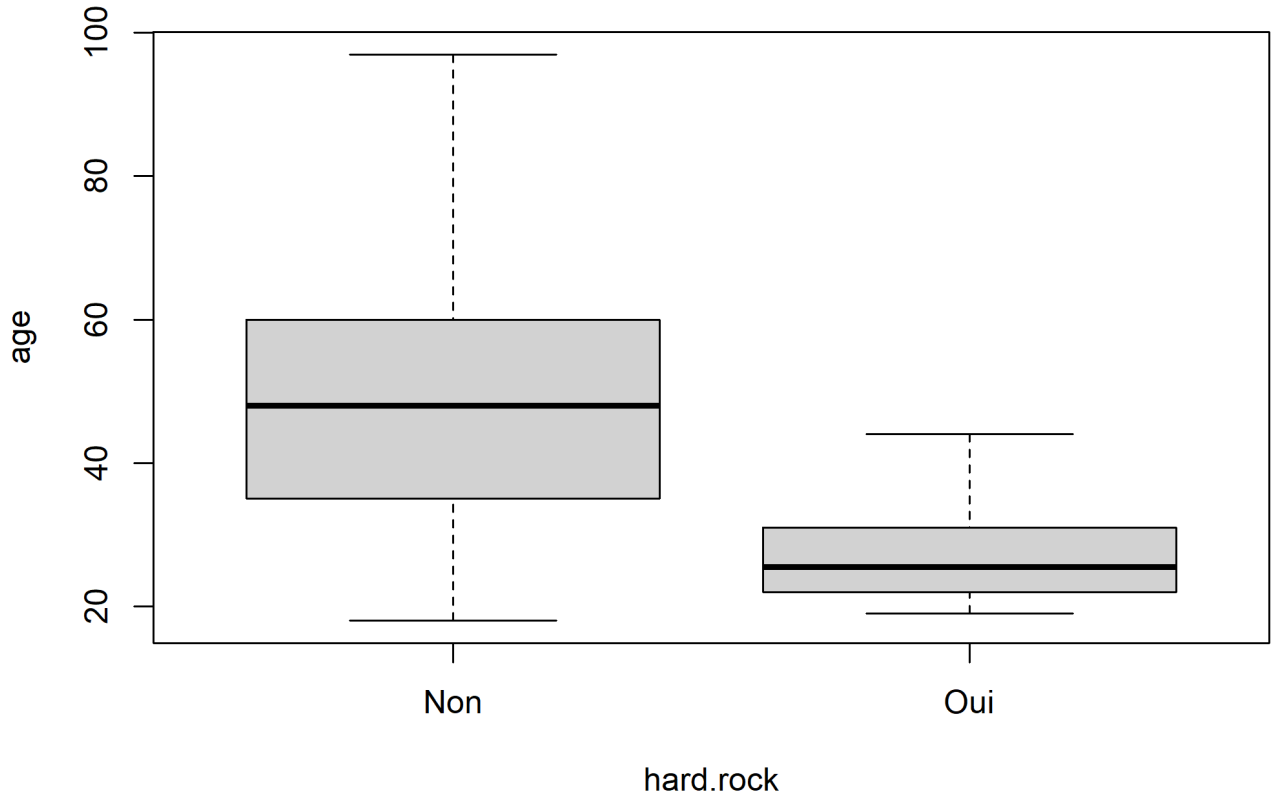
```
sns.scatterplot(data=df, x='IMC', y = 'Cholesterole', hue='Malade') # Permet de voir si il y a une corrélation entre des données.
```



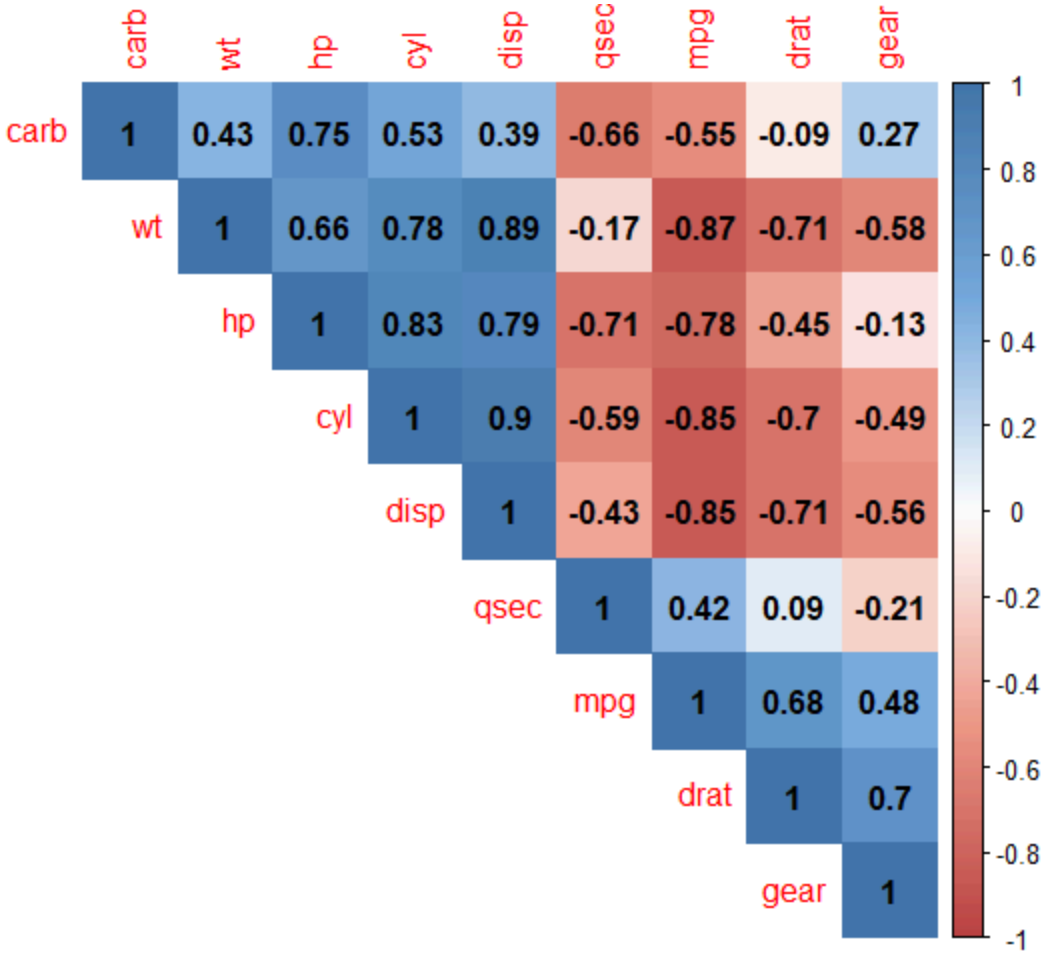
Une fois qu'on a déterminé les attributs qui sont corrélés on peut les supprimer car ces attributs sont similaire et ne sont donc pas nécessaire, aussi cela permet d'éviter de rendre confus les modèles car ils ont tendance à donner plus de poids à des attributs corrélés, c'est le cas de SVM par exemple.

Pour choisir lequel des deux attributs à supprimer cela dépends des autres attributs et du contexte.

On devrait aussi utiliser les boxplot bivariés :



Enfin pour avoir le plus d'info sur tous les attributs, il faut utiliser une matrice de corrélations :



Permet d'avoir une vision des corrélations générale

Une fois les données visualisé, il faut les projeter en utilisant :

- t-SNE 2D
- umap 3D
- ACP

On va obtenir un graphique avec des objets de deux classes, séparé par une frontière. Si des objets se trouvent du mauvais côté de la frontière, alors elles sont considéré comme perdu et elle vont réduire l'efficacité du modèle. Dans ce cas il existe trois chose à faire :

1. Continuer d'entraîner le modèle pour que la frontière de décision soit plus précise, mais il faut faire attention à ce que le modèle ne fassent pas du surapprentissage
2. Vérifier et modifier les données des classes concerné
3. Rien faire

Le prof ne fait pas confiance à un outil en particulier, il utilise les trois, par exemple t-SNE va relancer le placement des points à chaque fois, ce qui rend complexe l'interprétation des données.

Classer les données

ID	Age	cholesterol	poids	malade
1	25	180	68	NULL
2	30	190	72	NULL
3	28	175	70	NULL
4	45	220	82	1
5	50	242300	85	1
6	47	250	84	1
7	52	210	88	1
8	49	185	80	1
9	33	185	73	NULL

Suppression de la colonne IMC qui concorde avec cholesterol et poids
Supression de la ligne avec un patient dont l'âge est trop différents des autres

Normaliser les données

```
scaler.standardScaler()
x.scalerd = scaler.fit_transformer(df['Age','Poids'])
RobutScaler()
```

Découpage en train/test

Durant les premiers tests je vais devoir découper les données en données d'entrainement et données de test.

```
x_train, x_test, y_train, y_test = train.test_split(X,Y,test_size=0.3,
random.state=seed)
```

Prenez toujours un jeu de test, c'est toujours nécessaire. Ce jeu de test sera gardé pour toujours et il ne doit pas être touché ou modifier une fois fini. Le modèle ne devra jamais s'entrainer dessus, l'objectif c'est vraiment qu'il s'entraine dessus une fois tous les entraînement terminé.

```
x_train, x_val, y_train, y_val = train.test_split(x_train,y_train)
#x_val correspond aux valeurs qui vont être utilisé pour la validation, il
```

```
x_val correspond à un jeu de validation.
```

Il y à trois jeux de données en DeepLearning:

- train : données d'entraînement utilisé par le modèle
- test : à tester une seule fois et en dernier avec des données que le modèle n'a jamais vu
- validation : permet de savoir pendant l'entrainement si le modèle apprend bien, à suivre son évolution etc...

Il ne faut pas faire de up-sampling sur le jeu de données initial, on ne peut le faire que sur le jeu de données d'entrainement. On doit rajouter des exemples au sein des données d'entraînement pour qu'il soit capable de s'améliorer à identifier les cas.

Premier apprentissage

Permet de savoir si certains classifieur valent plus le coup que d'autres, et avec l'expérience on apprend ceux qui sont un peu meilleurs.

```
clf=LogisticRegression()
clf.fit('x_train','y_train')
x_prod = clf.prodict(x_val)
```

Evaluation

Une fois le premier entrainement effectué on doit vérifier les résultats obtenu et pour cela on utilise une matrice de confusion :

```
cm = confusion.matrix(y_val, y_pred)
```

Réels / Produits	**0**	**1**
0	True Negatif	False Positif
1	False Negatif	True Positif

Une fois la matrice de confusion obtenu on peut utiliser des formules pour mieux comprendre les résultats de notre modèle

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$f1 = \frac{Precision * Recall}{Accuracy + Recall}$$

Le modèle lui il sort des probas qu'un objet se trouve dans tel ou tel classe, et il place l'objet dans une classe si sa probabilité d'être dans une classe X est supérieur à 0.5. Il faut cependant faire attention, car ça reste des probas.
On peut ce poser la question de pourquoi c'est 0.5, c'est assez arbitraire. Pour régler ce problème de choix arbitraire, on va mettre en place un système d'air sous la courbe.

```
predict = clf.predict_proba(x_val) # Renvoie 1 ou 0 !
y = [0,0,0,1,1,1,1,1,0]
probabilite = [0.4,0.35,0.3,0.9,0.85,0.7,0.6,0.55,0.75]
#trier les probas
```

Le seuil a été choisis à 0.9 on a donc

rang	ID	clan	proba	TP/FP/TN/FN
1	4	1	0.9	TP
2	5	1	0.85	FN
3	9	0	0.75	TN
4	6	1	0.7	FN
5	7	1	0.6	FN
6	8	1	0.55	FN
7	1	0	0.4	TN
8	2	0	0.35	TN
9	3	0	0.3	TN

On a donc :

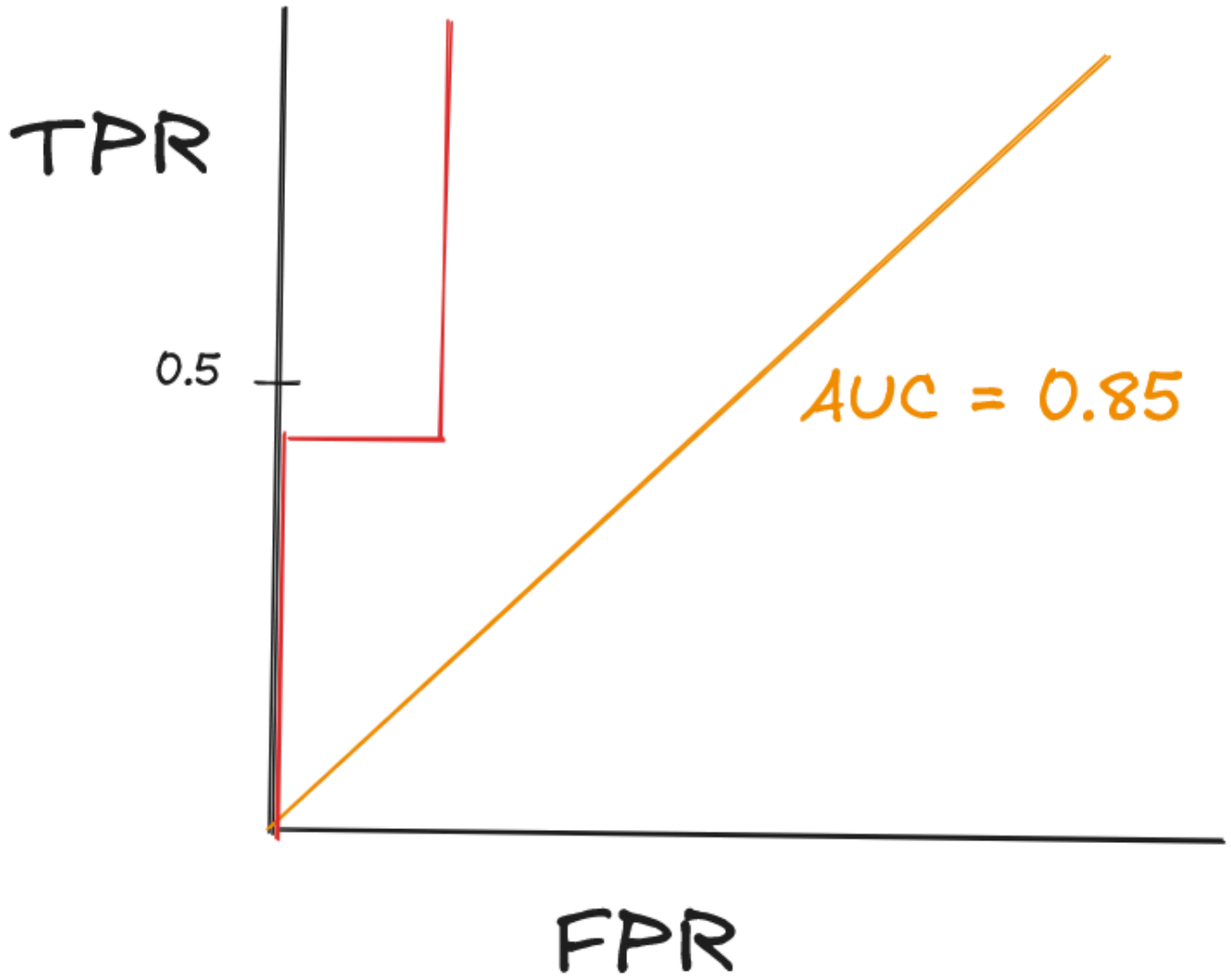
- TP = 1
- FP = 0
- TN = 4
- FN = 4

Une fois cela fais on peut faire les calculs d'accuracy, de recall, de précision puis le f1
On calcul ensuite le ratio de TP :

TPR = Recall

FPR = $\frac{FP}{FP + TN}$

On fait ensuite un graphique avec TPR en ordonné et FPR abscisse:



AUC = Area Under the Curve

Cross validation

```
crol_val_score(clf, x_train, y_train, cv=5, scoring='accuracy')
```

Vous ne pouvez pas distribuez votre modèle sans avoir fait de cross validation. Lors de la présentation d'un modèle on présente toujours la moyenne du modèle et l'écart type.
Par exemple avec deux modèles :

- 0.7 +/- 0.3 : très mauvais modèle, une bonne moyenne mais l'écart type est énorme

- 0.7 +/- 0.01 : plutôt bon modèle, très consistant

Lorsque l'on va essayer de générer des images, ou de faire des interpollation d'images, le temps d'apprentissage devient très très long, et dans ces cas là on ne fait plus la cross validation, même si il faudrait.

Pour ce qui est de la recherche d'hyper paramètre, utilisé Optuna, Random search ou autre outil. Mais ça rajoute énormément de temps de calculs.

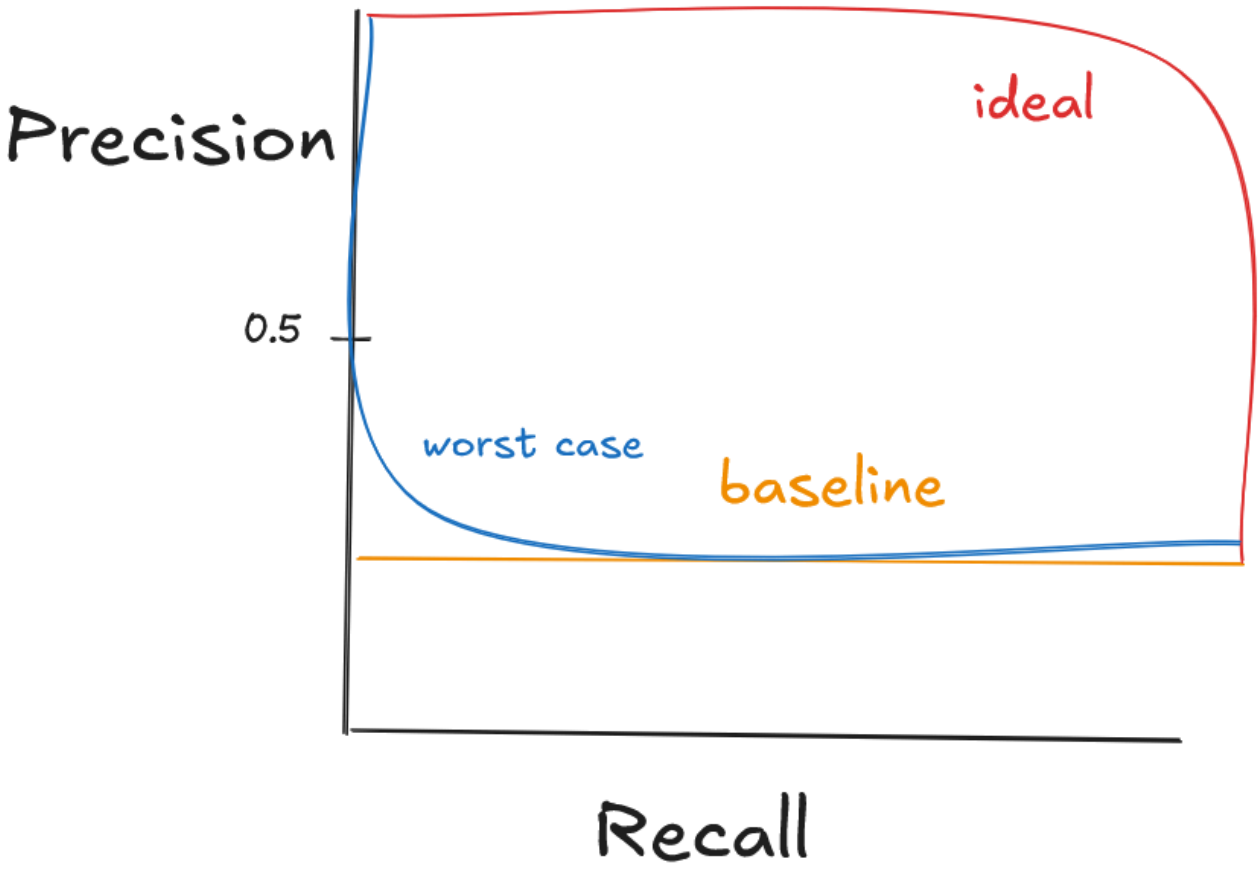
A la fin de la cross validation et du choix des hyper paramètres avec Optuna ou autre, alors on peut le tester sur x_test, y_test. Une fois les tests fait, on peut enfin mettre en production !

Mise en production

Pour ce faire il faut déjà quitter le notebook et ensuite créer un Pipeline :

```
pipeline(preparam = standardScaler(), clf = bestclass(----))
pipeline.fit(X,Y) # Bien prendre X et Y, pas x_train y_train, ça permet de ne pas perdre des données
```

Il faut garder la même proportion de classe entre train et test, même si on a globalement plus de données dans train !
Si on a des classes déséquilibré on ne peut pas simplement faire de l'AUC, vous devrez utiliser PRAUC.


Precision
0.5
ideal
worst case
baseline
Recall