Machine learning
Evaluation

Master AIDN: Applications Interactives et Données Numériques

Sylvie Gibet

1

1

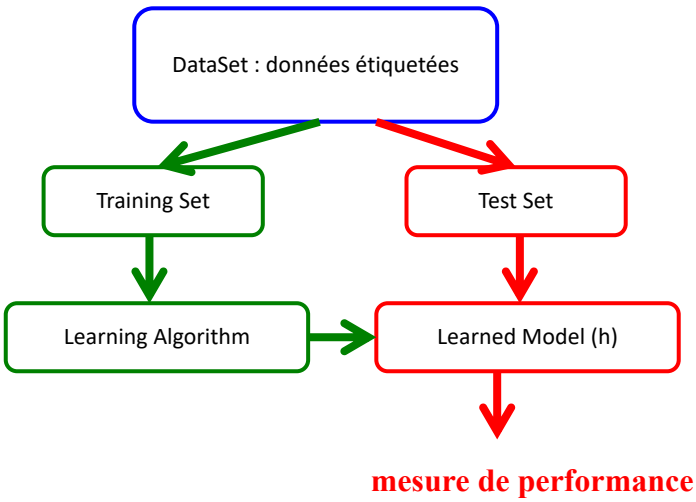
Évaluation - Objectifs

- Évaluer les performances d'un classifieur : mesures de performance
- Choisir les méta-paramètres d'un classifieur
- Évaluer et comparer chaque classe du classifieur

2

2

Évaluation – Principe



3

3

Mesures d'évaluation

Matrice de confusion

Classe prédite \ Classe réelle	+	-
+	True Positive (TP)	False Positive (FP)
-	False Negative (FN)	True Negative (TN)

Exemple

Classe prédite \ Classe réelle	Covid-19 = oui	Covid-19 = non	Total
Covid-19 = oui	6655	436	7091
Covid-19 = non	345	2564	2909
Total	7000	3000	10000

4

4

Métriques d'évaluation

	Positifs	Négatifs	
P\R	+	-	Total
+	TP	FP	P'
-	FN	TN	N'
	P	N	All

- Accuracy, ou taux de reconnaissance :

  Accuracy = (TP + TN)/All

- Taux d'erreur :

  Taux_erreur = 1 – Accuracy = (FP + FN)/All

5

5

Métriques d'évaluation

	Positifs	Négatifs	
P\R	+	-	Total
+	TP	FP	P'
-	FN	TN	N'
	P	N	All

- Précision : exactitude : % des objets que le classifieur a étiquété dans la classe positif qui sont effectivement positifs
  Précision = TP /(TP + FP)
- Rappel (ou sensibilité) (Recall) : complétude ou sensibilité : % des objets positifs que le classifieur a bien classé dans la classe positif :
  Rappel = TP / (TP + FN)
- F1-score = **2 * precision * rappel / (precision + rappel)**

Mesure une sorte de moyenne arithmétique de la précision et du rappel

6

6

Métriques d'évaluation - exemple

Classe prédite \ Classe réelle	Covid-19 = oui	Covid-19 = non	Total
Covid-19 = oui	6655	436	7091
Covid-19 = non	345	2564	2909
Total	7000	3000	10000

- Accuracy = 9219/ 10000 = 92.2%
- Précision = 6655/ 7091 = 93.85 %
- Rappel = 6655 / 7000 = 95.1%

	Positifs	Négatifs	
P\R	+	-	Total
+	TP	FP	P'
-	FN	TN	N'
	P	N	All

7

7

Métriques d'évaluation - exemple

Classe prédite \ Classe réelle	diabète = oui	diabète = non	Total
diabète = oui	267	455	722
diabète = non	733	8545	9278
Total	1000	9000	10000

- Accuracy = 8811 / 10000 = 88.1%
- Précision = 267/ 722 = 36.9%
- Rappel = 267/ 1000 = 26.7%

	Positifs	Négatifs	
P\R	+	-	Total
+	TP	FP	P'
-	FN	TN	N'
	P	N	All

8

8

Train/Test Split and Cross Validation

- Quand on utilise des méthodes de machine learning, on sépare les données en deux sous-ensembles : le training set et le test set, parfois en trois : training, validation et test sets.
- Puis on fait correspondre (fit) le modèle de ML aux données d'entraînement afin de faire des prédictions sur les données de test.
- Enfin, on évalue le modèle de prédiction appliqué aux données par les métriques d'évaluation.

9

9

Train/Test Split

On sépare les données en deux sous-ensembles

- **Training set** : Un jeu de données d'apprentissage est un ensemble de données d'exemples utilisé pendant le processus d'apprentissage et est utilisé pour ajuster les paramètres (par exemple, les poids) d'un réseaux de neureones
- les entrées/sorties sont connues et le modèle apprend sur ces données

Nombre total d'exemples

10

Train/Test Split

On sépare les données en deux sous-ensembles

- **Test set** : Un jeu de données de test est un jeu de données indépendant du jeu de données d'apprentissage, mais qui suit la même distribution de probabilité que le jeu de données d'apprentissage
- Jeu d'exemples utilisés uniquement pour évaluer les performances d'un classificateur
- pour tester la prédiction du modèle de machine learning sur ce sous-ensemble

.

Nombre total d'exemples

11

Train/Test Split

En python

```
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
```

12

Exemple - load data

En python

```
# Load the Diabetes Housing dataset
    columns = “age sex bmi map tc ldl hdl tch ltg glu”.split()
# Declare the columns names
    diabetes = datasets.load_diabetes()
# Call the diabetes dataset from sklearn
    df = pd.DataFrame(diabetes.data, columns=columns)
# load the dataset as a pandas data frame
    y = diabetes.target
# define the target variable (dependent variable) as y
```

13

Exemple – split

En python

```
#create training and testing vars

X_train, X_test, y_train, y_test = train_test_split(df, y,
test_size=0.2)
```

14

X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.8, random_state=42)
taille du training set : 80% ; test set : 20%

- **Why Random_state=42?**
- If you don't set random_state to 42, every time you run your code again, it will generate a different test set.
- One solution is to save the test set on the first run, and then load it on subsequent runs. Another option is to set the start value of the random number generator (seed);
  - for example, with np.random.seed(42), it always generates the same clues mixed up.

15

Exemple - fit

En python

```
# fit le modèle (apprentissage)
lm = linear_model.LinearRegression()
model = lm.fit(X_train, y_train)
predictions = lm.predict(X_test) # prédiction sur les données test
# visualiser les premiers résultats
predictions[0:5]
array([ 205.68012533, 64.58785513, 175.12880278,
169.95993301,
128.92035866])
```

16

Exemple – plot

En python

```
# The line / model
plt.scatter(y_test, predictions)
plt.xlabel("True Values")
plt.ylabel("Predictions")
```

17

Séparation Train set / Test set : des limitations

- Il est important de vérifier la qualité de l'ensemble d'apprentissage (Training set) :
  - Y-a-t-il assez de données ?
  - Est-ce que le modèle d'apprentissage généralise bien ? C'est-à- dire peut s'adapter à de nouvelles données ?
  - Est-ce que les exemples sont répartis de manière équilibrée sur les différentes classes ?
  - Que se passe-t-il si le "split" n'est pas aléatoire ? (par exemple une classe n'est pas représentée dans cet ensemble)

18

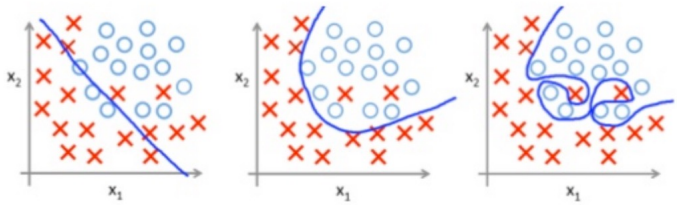
Séparation Train set / Test set : des limitations

- On veut éviter en particulier
  - **L'overfitting** et
  - **L'underfitting**

19

Underfitting (sous-apprentissage)

- Underfitting : le modèle ne s'adapte pas bien aux données d'apprentissage, il ne capte pas la tendance dans les données.
- Dans ce cas, le modèle, souvent trop simple, n'a pas une bonne capacité de prédire ; il ne généralise pas bien à de nouvelles données d'apprentissage ou de test.



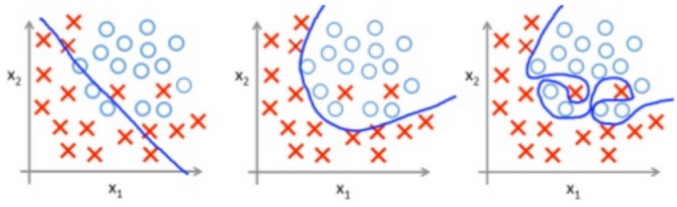
underfitting correct overfitting

20

20

Overfitting (sur-apprentissage)

- Overfitting : le modèle est trop bien entraîné, il s'adapte trop aux données
- On apprend les données, pas le modèle
- Par conséquent, le modèle d'apprentissage marche bien sur les données utilisées pour l'apprentissage, mais marche mal pour de nouvelles données (il ne généralise pas bien)



underfitting correct overfitting

21

21

Train/Validation/Test

Pour éviter ces problèmes, on considère 3 sous-ensmbles :

- **Training set**
- **Test set**.
- **Validation set**

22

Train/Validation/Test

Pour éviter ces problèmes, on considère 3 sous-ensmbles :

- **Validation set** : Un jeu de données de validation est un jeu de données d'exemples utilisés pour régler les hyperparamètres du modèle de machine learning
- Ce jeu de données de validation, comme le jeu de test, doit suivre la même loi de probabilité que le jeu de données d'apprentissage.
  - Exemple : pour un réseau de neuronnes, nombre d'unités cachées dans chaque couche
  - Pour un kNN : nombre de plus proches voisins

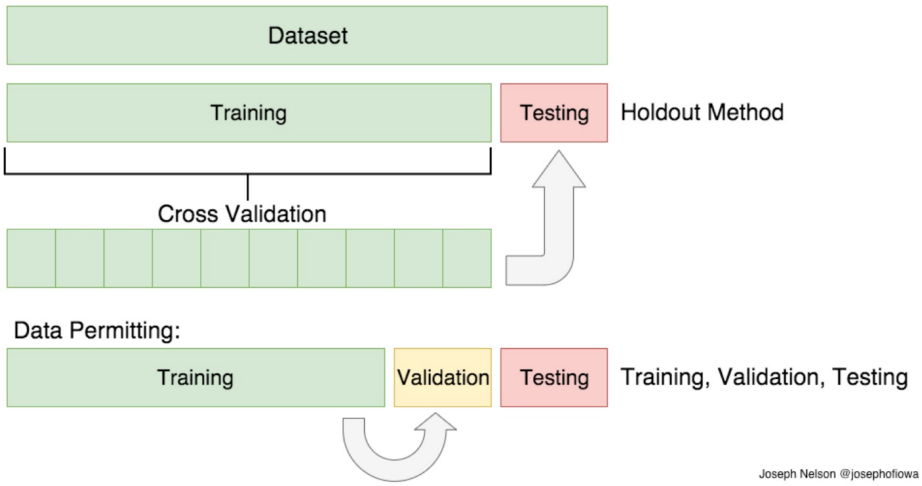
23

Train/Validation/Test

- Afin d'obtenir des résultats plus stables et d'utiliser toutes les données précieuses pour l'entraînement, un jeu de données peut être divisé à plusieurs reprises en plusieurs jeu de données d'entraînement et de validation.
- C'est ce qu'on appelle la validation croisée (cross validation).
- Pour valider les performances du modèle, un jeu de données de test supplémentaire, qui n'est pas soumis à la validation croisée, est normalement utilisé.

24

Train / Validation / Test



25

Séparation Train set / Test set : des limitations

- Plusieurs méthodes pour palier ces problèmes
  - **leave-one-out** : 1 exemple comme test, tout le reste en apprentissage (lorsqu'il y a très peu de données)
  - **Cross-validation (validation croisée)** : on partitionne le Training set en sous-ensembles de taille équivalente
  - **Ré-échantillonnage aléatoire** : on ré-échantillonne de manière aléatoire l'ensemble des données en ensembles de Training / Test
  - **Ré-échantillonnage aléatoire stratifié** : on conserve des proportions d'exemples dans les classes semblables

26

Cross-validation

- Principe : on divise le training set en partitions de tailles égales (par exemple division en 10-fold (10%) ou 20-fold (20%))
- Exemple : 5 sous-ensembles S1, S2, S3, S4, S5

Training set	Test set	Taux d'erreur
S1, S2, S3, S4	S5	
S2, S3, S4, S5	S1	
S3, S4, S5, S1	S2	
S4, S5, S1, S2	S3	
S5, S1, S2, S3	S4	

- Résultat : moyenne des taux d'erreurs sur les différentes partitions

27

Cross-validation

- Utilisation de la cross-validation pour la sélection de modèle
- Exemple du k-NN (k-nearest neighbors)

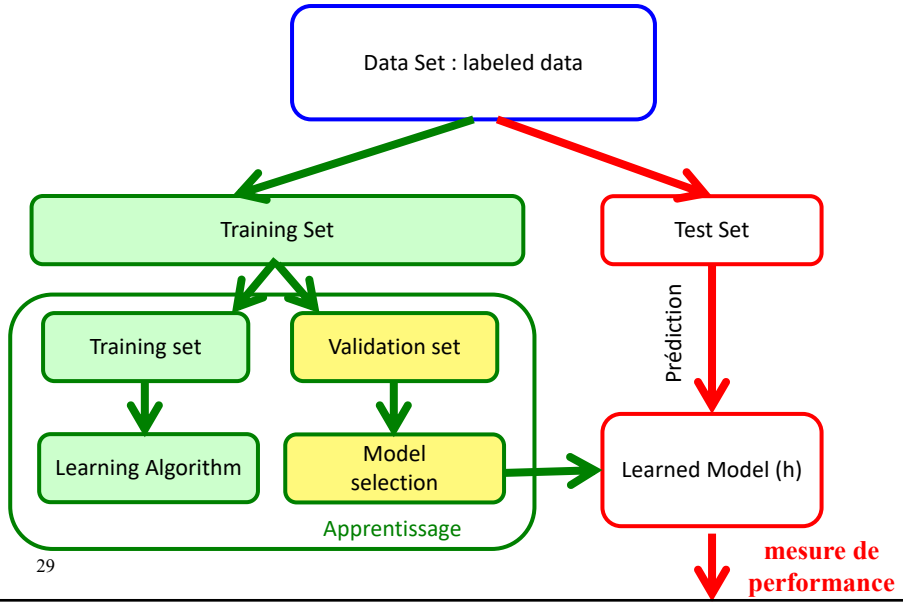
Pour chaque k (k=1, k=3, k=5)

Cross-validation : mesures de performance

On choisit le k* qui maximise les mesures de performance

28

Méthodologie de validation pour la sélection de modèle



29

29

Courbe ROC

Receiving Operating Characteristics

Autre manière d'évaluer les modèles de prédiction

30

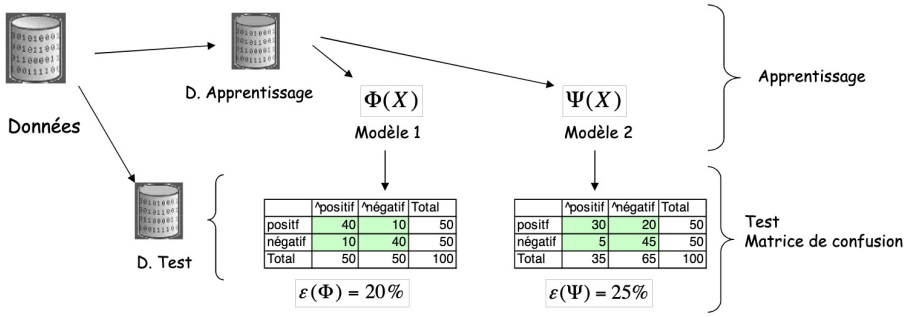
Courbe ROC : pourquoi ?

- Nécessité d'évaluer les modèles de prédiction
  - Comparer plusieurs modèles
  - Évaluer la performance du modèle en déploiement (fiabilité)
- Taux d'erreur (estimation de la probabilité de mal classer un individu), mesures d'accuracy, de rappel et de précision : oui, mais… possibilité de mauvaise interprétation (dépend du problème posé)
- Courbe ROC : la portée va au-delà des indicateurs issus de la matrice de confusion
  - Cas des classes très déséquilibrées

31

Taux d'erreur : attention !

Schéma habituel d'évaluation des modèles



	^positif	^négatif	Total
positf	40	10	50
négatif	10	40	50
Total	50	50	100

$\varepsilon(\Phi) = 20\%$

	^positif	^négatif	Total
positf	30	20	50
négatif	5	45	50
Total	35	65	100

$\varepsilon(\Psi) = 25\%$

Conclusion : Modèle 1 serait meilleur que Modèle 2

- Suppose que la matrice de coût de mauvais classements est symétrique et unitaire

32

Taux d'erreur : attention !

Coût de mauvaise affectation non-symétrique

	^positif	^négatif
positf	0	1
négatif	10	0

	^positif	^négatif	Total
positf	40	10	50
négatif	10	40	50
Total	50	50	100

$c(\Phi) = 1.1$

	^positif	^négatif	Total
positf	30	20	50
négatif	5	45	50
Total	35	65	100

$c(\Psi) = 0.7$

Conclusion : Modèle 2 serait meilleur que Modèle 1 dans ce cas ???

- Matrices de coût : dépendent de la tâche
- Question : comment comparer les modèles, indépendamment de la matrice de coût de mauvaise affectation ?

33

Courbe ROC : intérêt

- Courbe ROC : la portée va au-delà des indicateurs issus de la matrice de confusion (taux d'erreur, indicateurs précision/rappel) :
  - Cas des classes très déséquilibrées
  - Cas où le coût de mauvaise affectation peut être sujet à modifications
- Attention : s'applique au problème à deux classes dans lequel la classe positive (cible) est bien identifiée
- Remarque : Les courbes ROC furent inventées pendant la Seconde Guerre mondiale pour montrer la séparation entre les signaux radar et le bruit de fond.

34

Sensibilité et spécificité

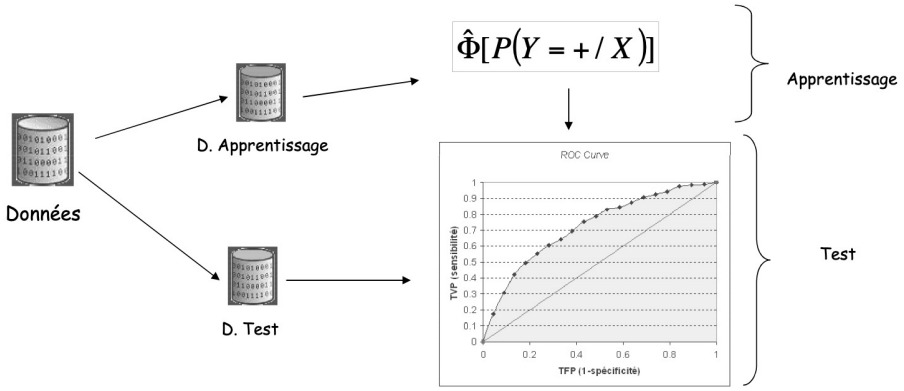
- Importance majeure en épidémiologie et en théorie de la détection :
  - La sensibilité (ou Recall) d'un test disgnostic en médecine est la capacité à détecter le maximum de malades (il faut avoir le moins de faux négatifs : personnes malades détectées négatives) : TP/(TP+FN)
    -> éviter les faux négatifs !!!
  - La spécificité est la capacité à ne détecter que les malades (il faut avoir le moins de faux positifs, c'est à dire des personnes non malades testées positives)

ROC = sensibilité = f(1 - spécificité)

35

35

Courbe ROC : principe

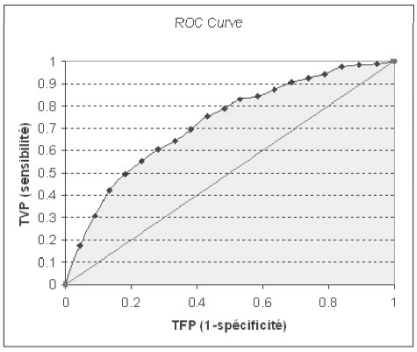


- Attention : Problème à deux classes dans lequel la classe positive (cible) est bien identifiée
- Intuitivement : on veut le plus haut niveau de détection sur le plus bas niveau de fausses alarmes

36

36

Courbe ROC : principe



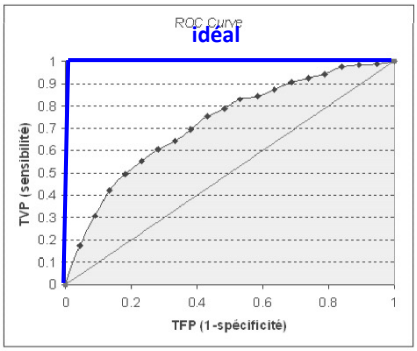
	Positifs	Négatifs	
P\R	+	-	Total
^ +	TP	FP	P’
^ -	FN	TN	N’
	P	N	All

- On trace la courbe sensibilité= f(1-spécificité)
  - sensibilité = Rappel = TP/Positifs = TP/(TP+FN)
  - 1 –spécificité = 1 – précision = FP/Négatifs = FP / (FP+TN)

37

37

Courbe ROC : principe



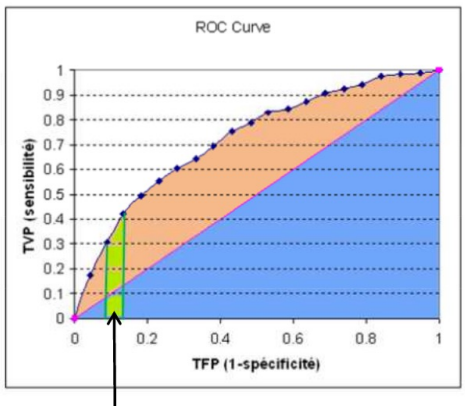
	Positifs	Négatifs	
P\R	+	-	Total
^ +	TP	FP	P’
^ -	FN	TN	N’
	P	N	All

- On trace la courbe sensibilité= f(1-spécificité)
  - sensibilité = Rappel = TP/Positifs = TP/(TP+FN)
  - 1 –spécificité = 1 – précision = FP/Négatifs = FP / (FP+TN)

38

38

Courbe ROC : intérêt



- AUC : aire sous la courbe
- L'AUC fournit une mesure agrégée des performances pour tous les seuils de classification possibles.
- On peut interpréter l'AUC comme une mesure de la probabilité pour que le modèle classe un exemple positif aléatoire au-dessus d'un exemple négatif aléatoire.

39

39