

Sous apprentissage — Sur apprentissage

Pascal Poncelet
LIRMM

Pascal.Poncelet@lirmm.fr
<http://www.lirmm.fr/~poncelet>



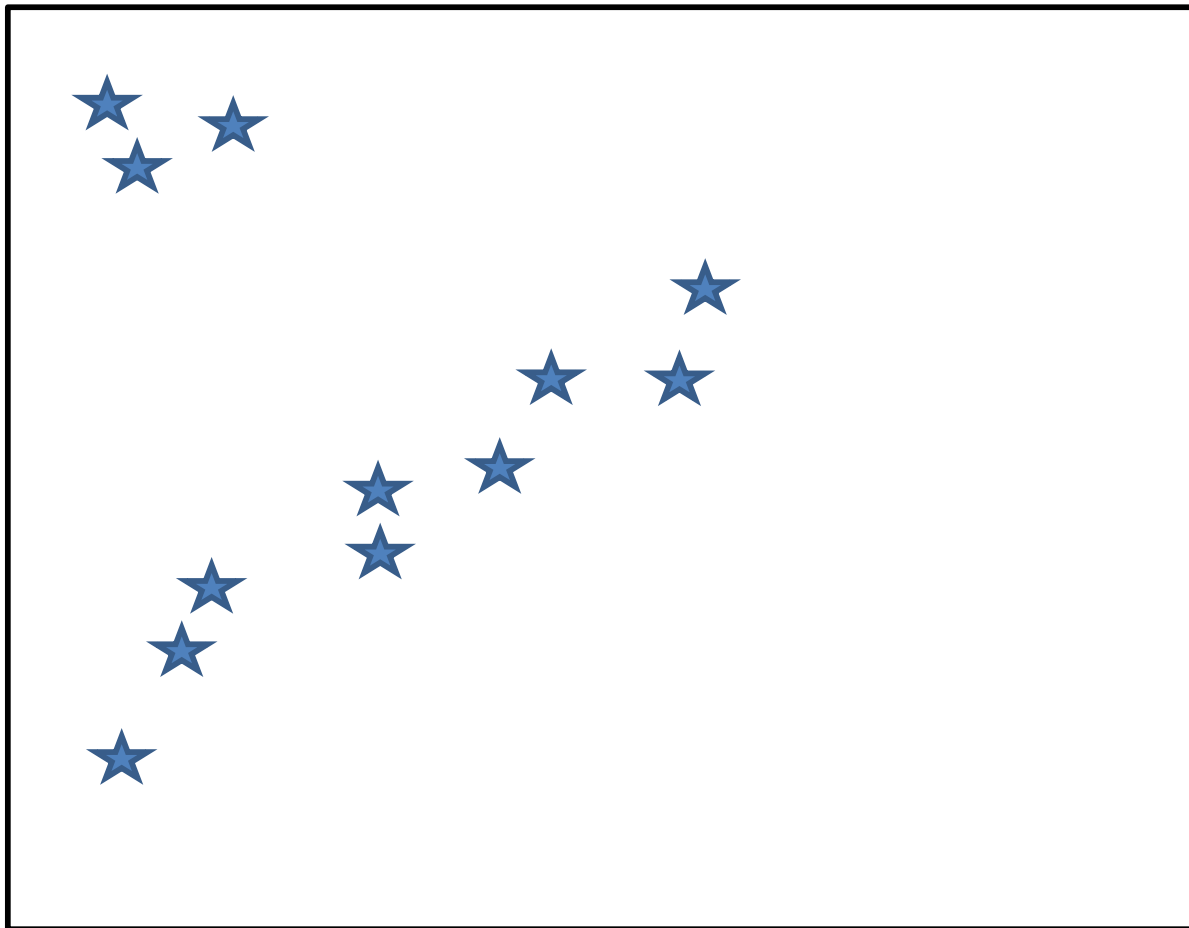
Quel est l'objectif d'un modèle ?



Quel est l'objectif d'un modèle ?

- Minimiser une fonction de coût
- Prendre en compte les relations qui peuvent exister dans les données d'apprentissage
- Avoir un bon pouvoir de généralisation

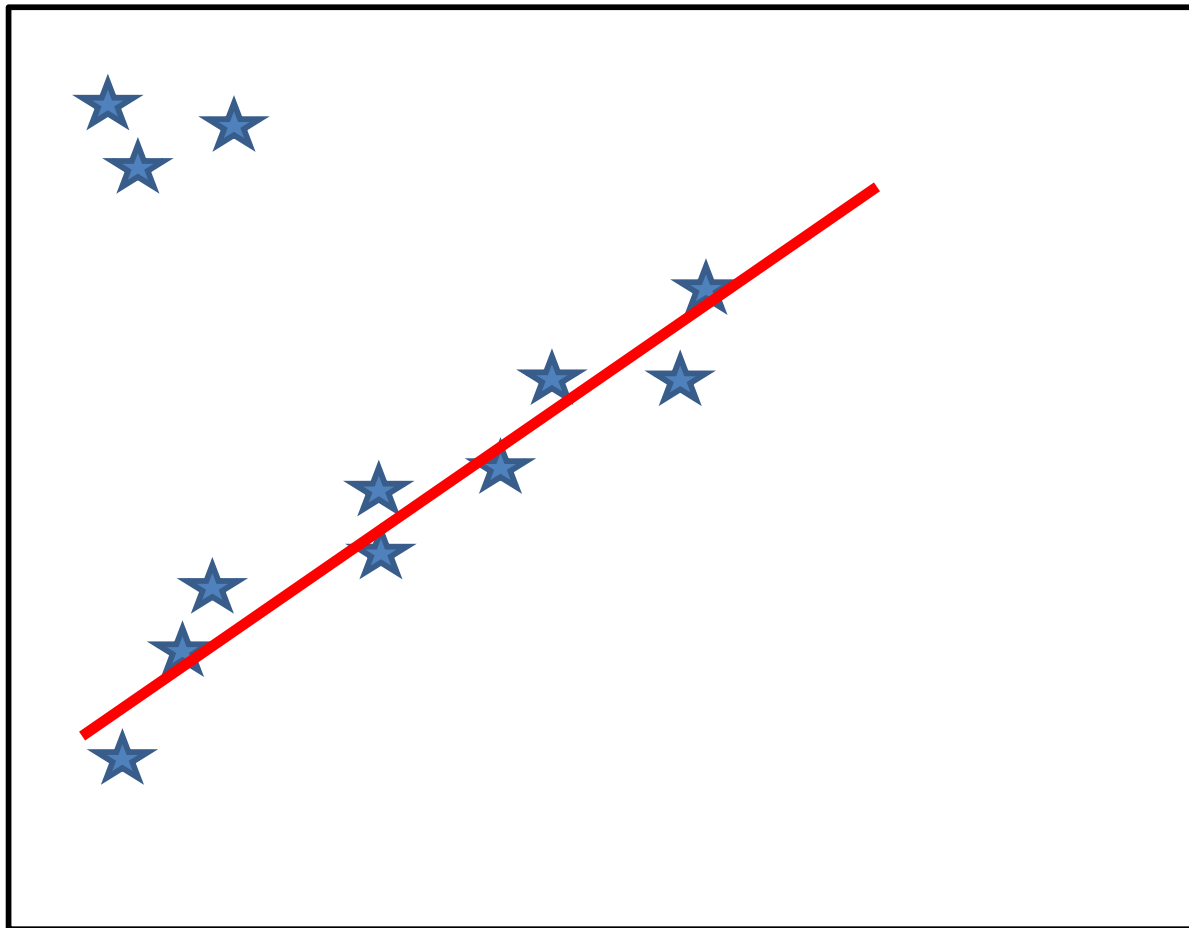
Importance de la fonction de coût



$$\sum |y_{\text{predict}} - y|$$

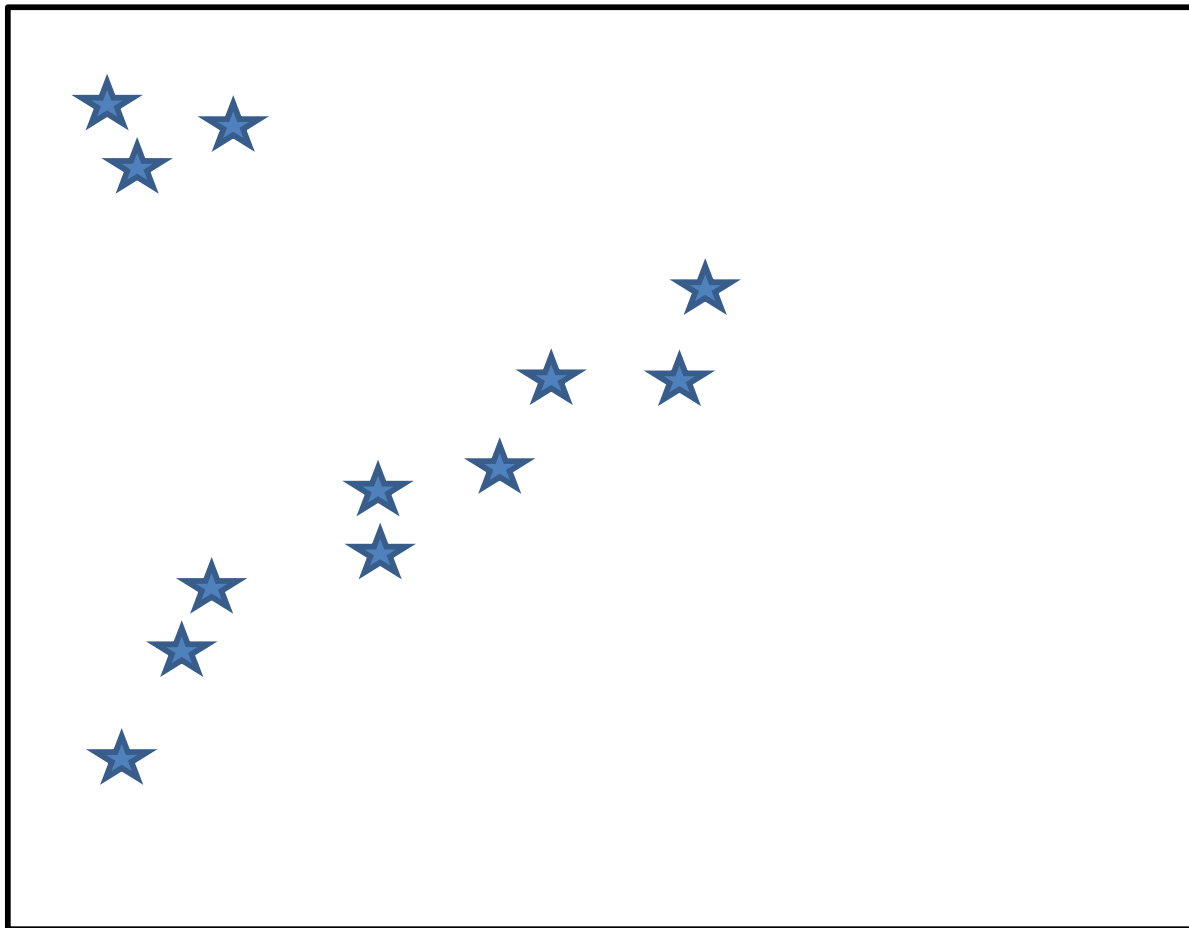
Droite de régression ?

Importance de la fonction de coût



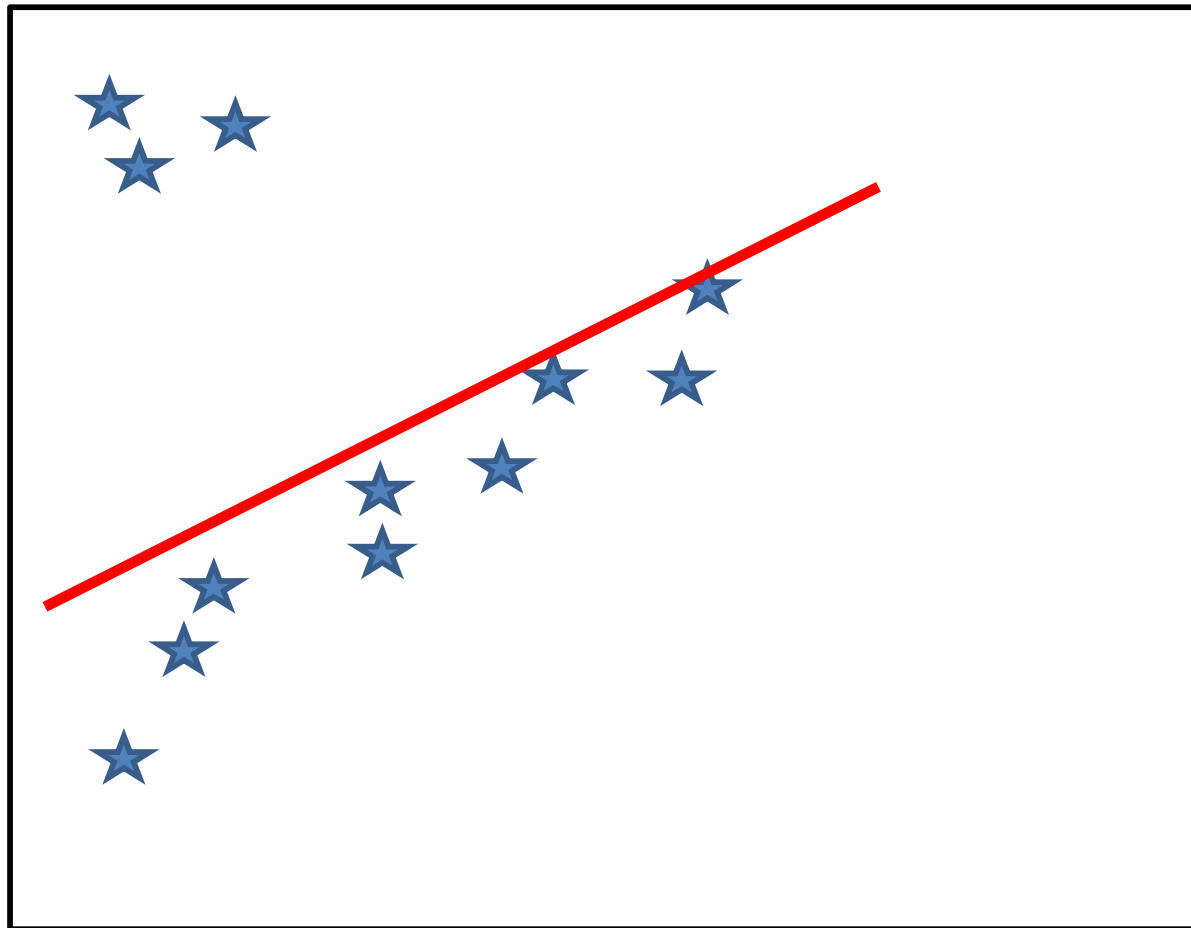
$$\sum |y_{\text{predict}} - y|$$

Importance de la fonction de coût



$$\sum |y_{\text{predict}} - y|^2$$

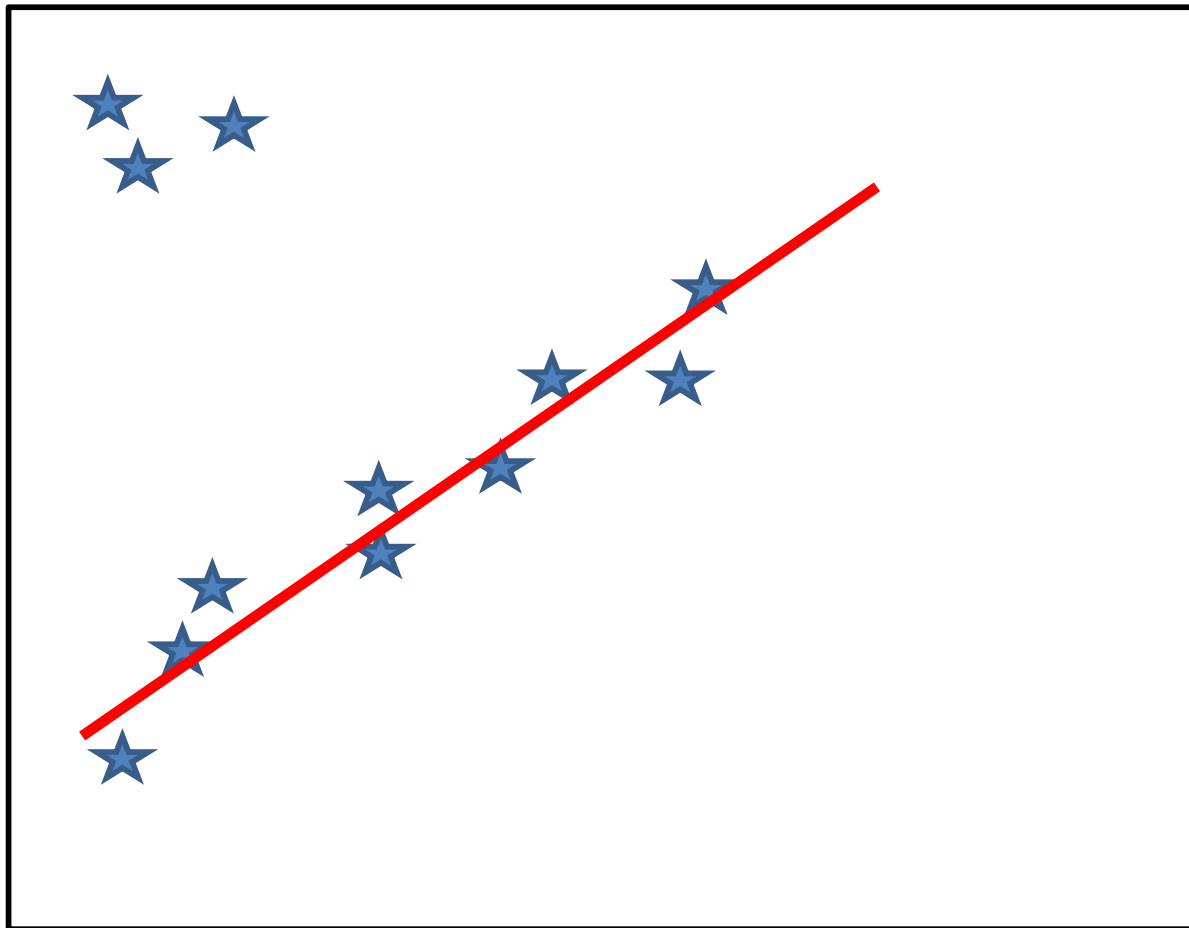
Importance de la fonction de coût



$$\sum |y_{\text{predict}} - y|^2$$

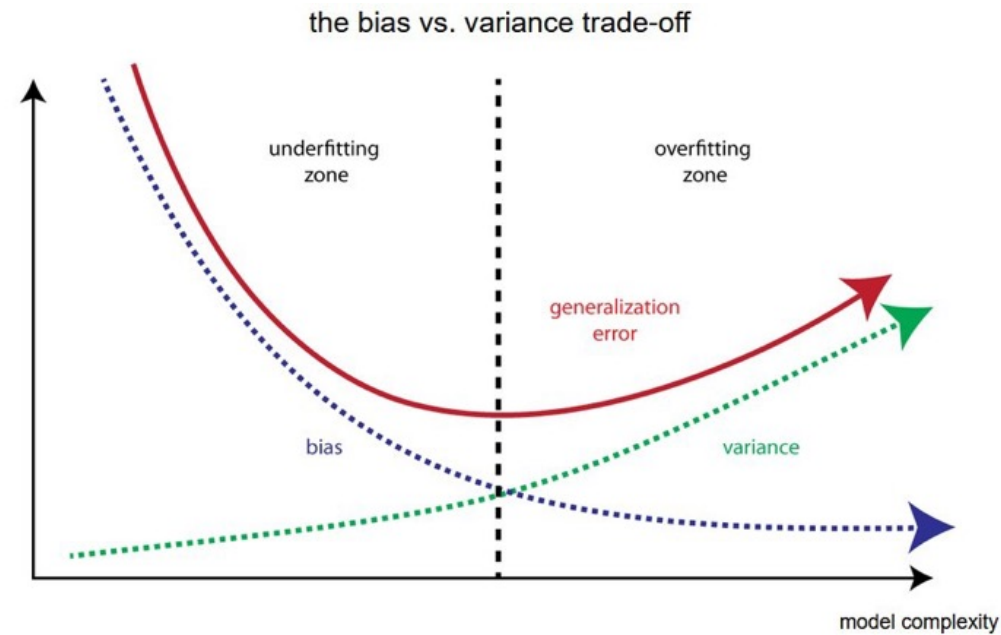
Plus de poids sur les outliers

Importance de la fonction de coût



$$\sum |y_{\text{predict}} - y|$$

Bien se situer

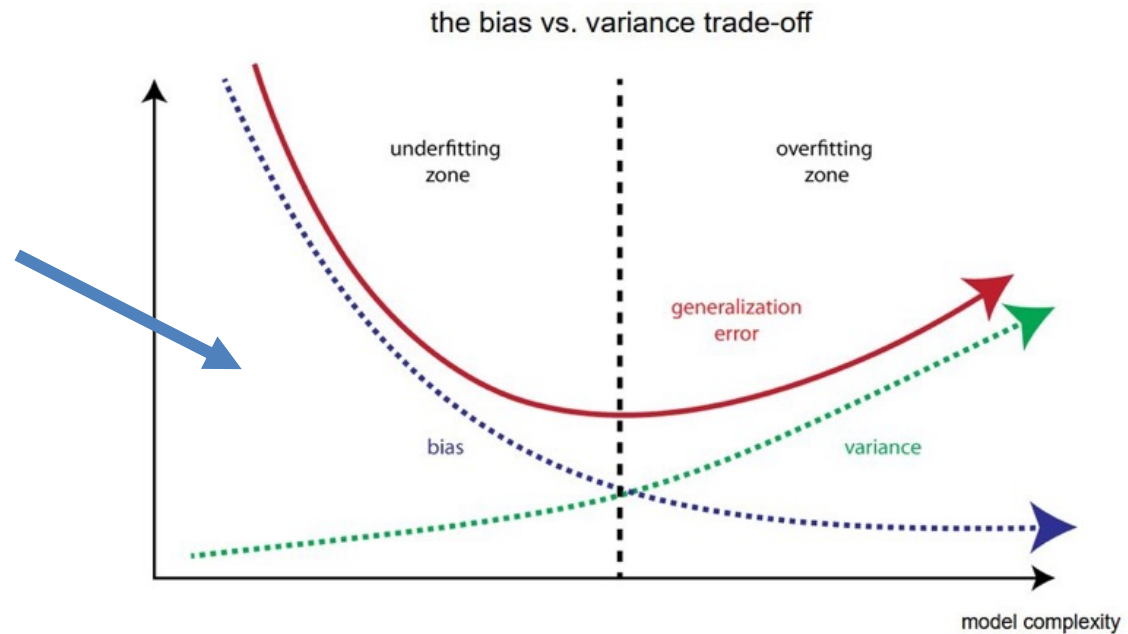


Bien se situer

Modèle trop simple

pas de prise en compte des relations qui peuvent exister entre les données d'apprentissage afin de pouvoir améliorer les prédictions.

BIAIS ELEVE



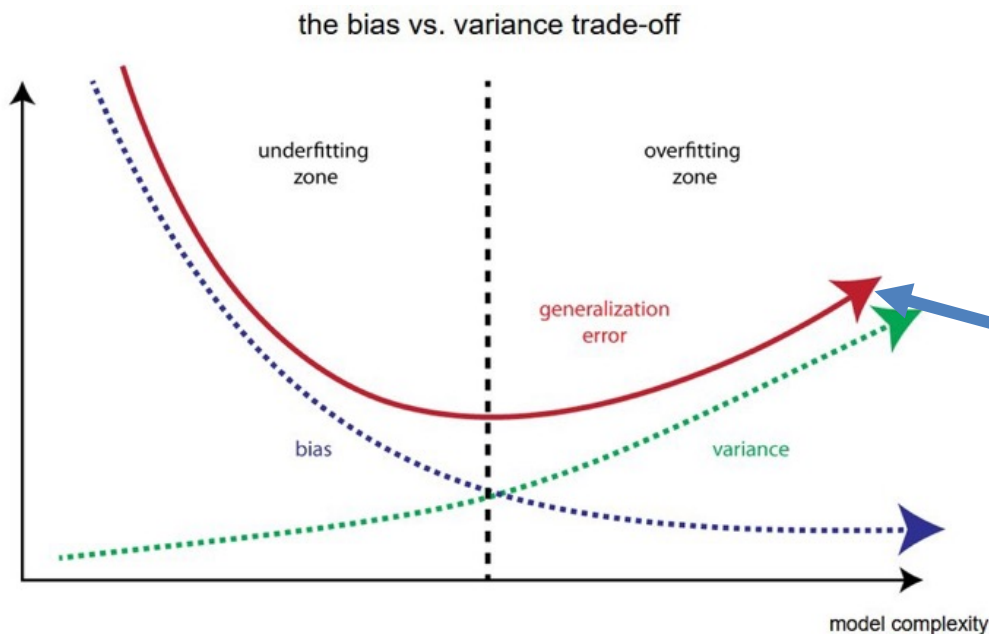
Bien se situer

Modèle trop compliqué

en cherchant des relations dans les variables : ajout du bruit.

en modifiant un peu les données, prédictions très différentes car le modèle est trop sensible et réagit de manière excessive au changement de données

VARIANCE ELEVEE

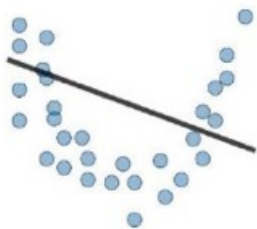


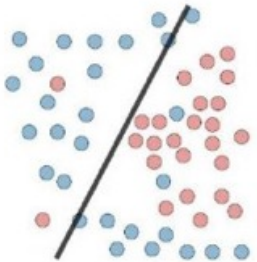
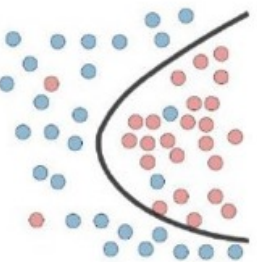
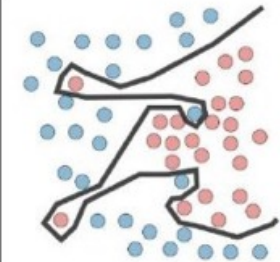


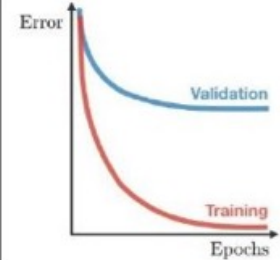


Underfit - Overfit

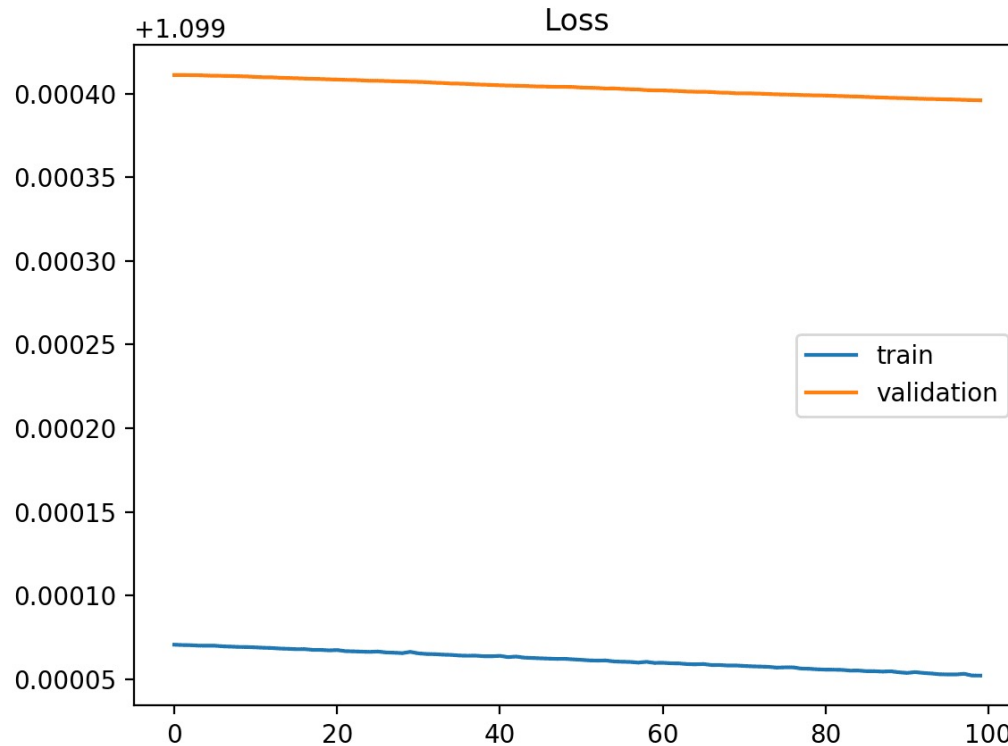
- **Biais élevé** : modèle *sous-adapté* (**Underfit – sous apprentissage**) [aux données d'apprentissage]
 - erreur de prédiction élevée à la fois sur les données d'entraînement et les données de test
- **Forte variance** : modèle *sur ajusté* (**Overfit – sur apprentissage**) [aux données d'apprentissage]
 - erreur de prédiction généralement faible sur les données d'entraînement mais élevée sur les données de test (manque de généralisation)



Des exemples

	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none"> - High training error - Training error close to test error - High bias 	<ul style="list-style-type: none"> - Training error slightly lower than test error 	<ul style="list-style-type: none"> - Low training error - Training error much lower than test error - High variance
Regression			
Classification			
Deep learning			
Remedies	<ul style="list-style-type: none"> - Complexify model - Add more features - Train longer 		<ul style="list-style-type: none"> - Regularize - Get more data

Exemples d'underfitting

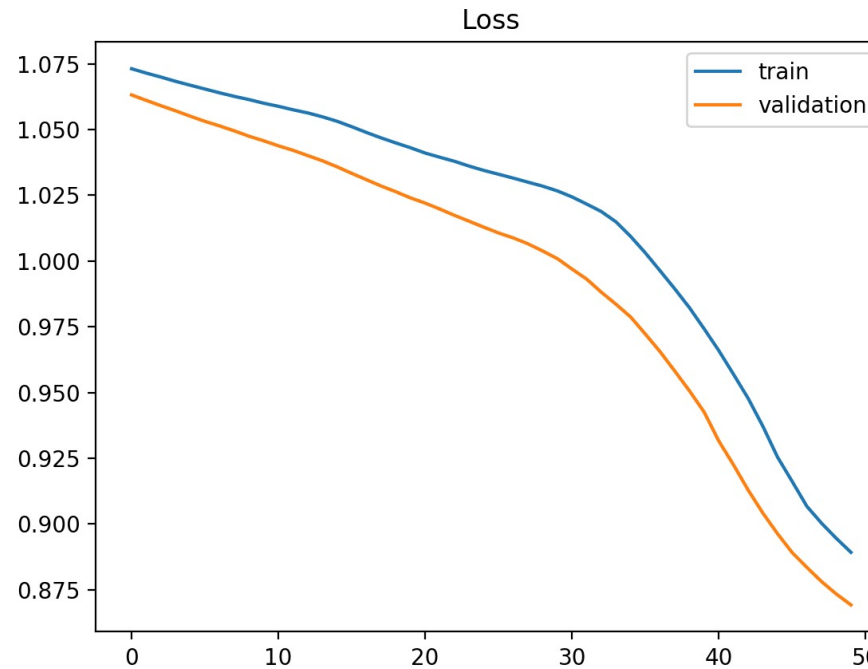


modèle trop simple

- Beaucoup d'erreurs sur la validation (orange)
- Incapable de prendre en compte la complexité du jeu de données



Exemples d'underfitting

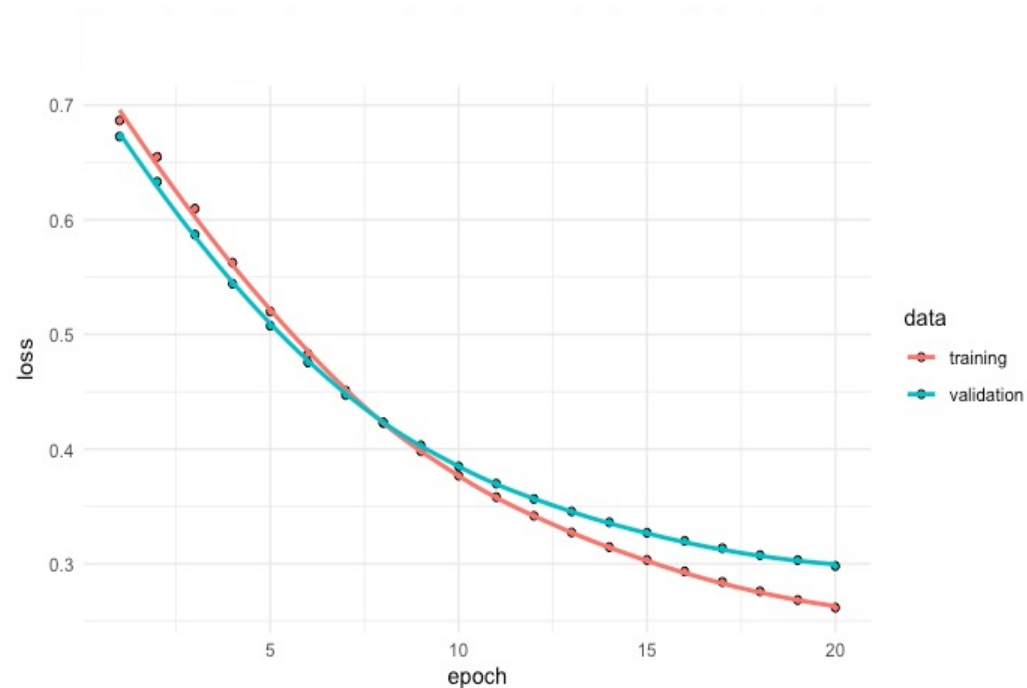


modèle arrêté prématurément

- La *loss* diminue et continue à diminuer jusqu'en bas.
- Le modèle est encore capable d'apprendre



Exemples d'underfitting



modèle aussi arrêté prématurément

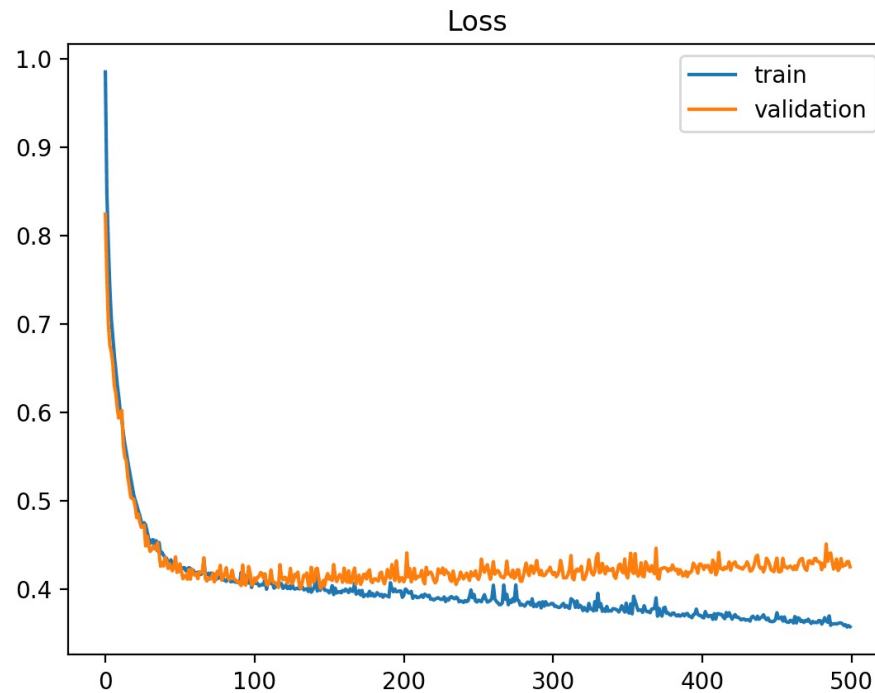
- La *loss* diminue et continue à diminuer jusqu'en bas.
- Le modèle est encore capable d'apprendre

Condition d'underfitting

- Symptômes :
 - La courbe *loss* reste stable quel que soit l'entraînement
 - La courbe *loss* continue de diminuer jusqu'à la fin de l'entraînement
- Solutions :
 - Modèle trop simple : augmenter la complexité du modèle (ajout de features, augmenter le nombre de paramètres en deep learning, etc.). Attention pas d'erreur d'entraînement à 0 (overfitting)
 - Augmentation du nombre d'epochs si le modèle montre qu'il peut apprendre plus. Peut être voir le *learning rate* pour accélérer le processus.
 - Bien vérifier que les données soient bien mélangées à chaque epoch (*cross validation*).



Exemples d'overfitting

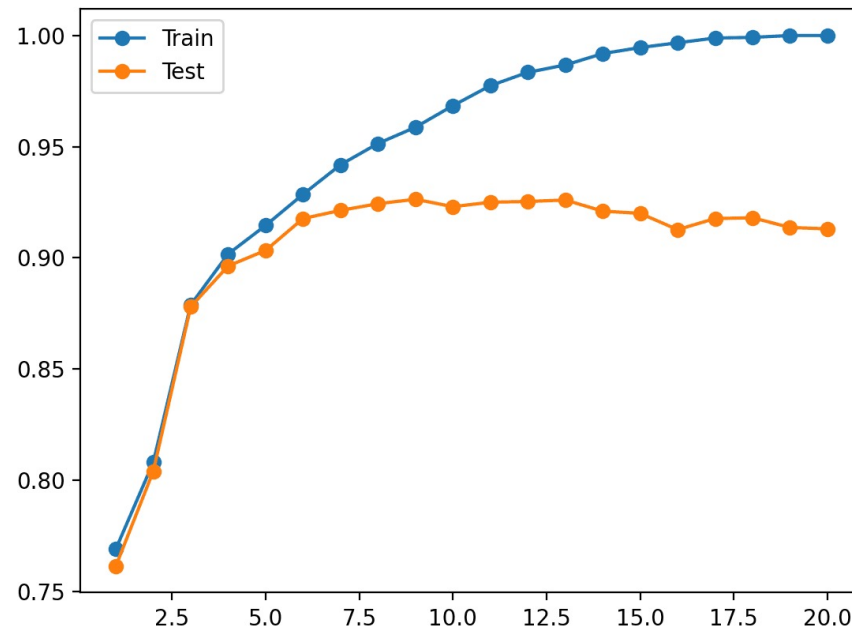


modèle de « données d'entraînement »

- La *loss* pour la validation ne diminue pas
- Le modèle apprend les données d'entraînement et les bruits.



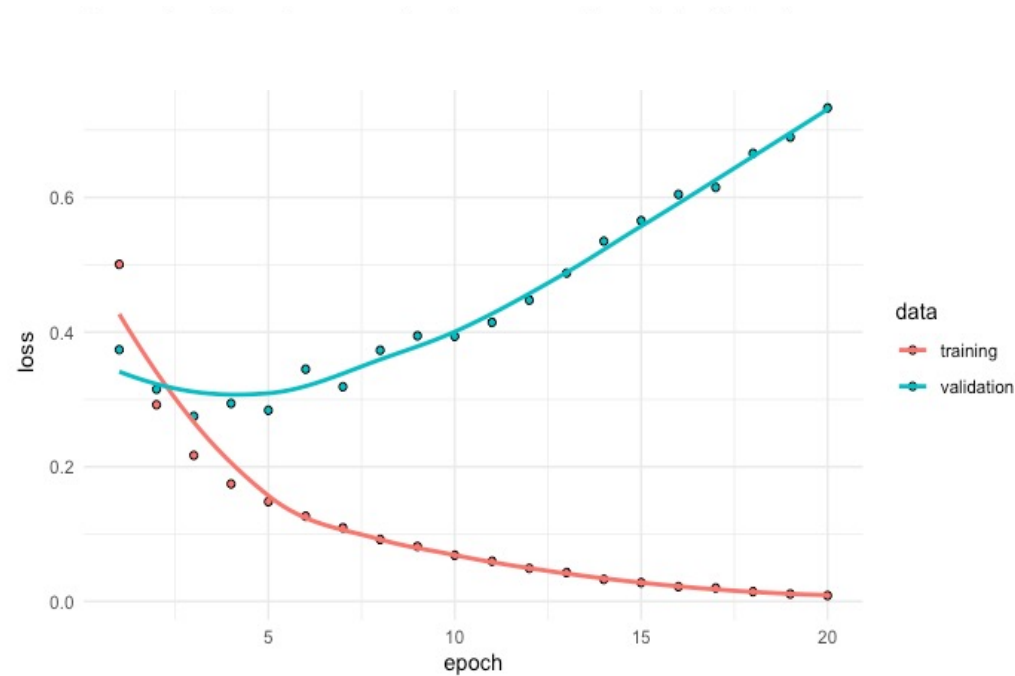
Exemples d'overfitting



modèle de « données d'entrainement »

- La courbe d'*accuracy* pour la validation n'augmente plus – plateau
- Idem que précédent mais avec l'*accuracy*

Exemples d'overfitting



Courbe loss en U

- L'overfitting démarre très tôt (courbe en forme de U)
- Le modèle apprend trop vite
- Généralement le learning rate est trop élevé



Condition d'overfitting

Symptômes :

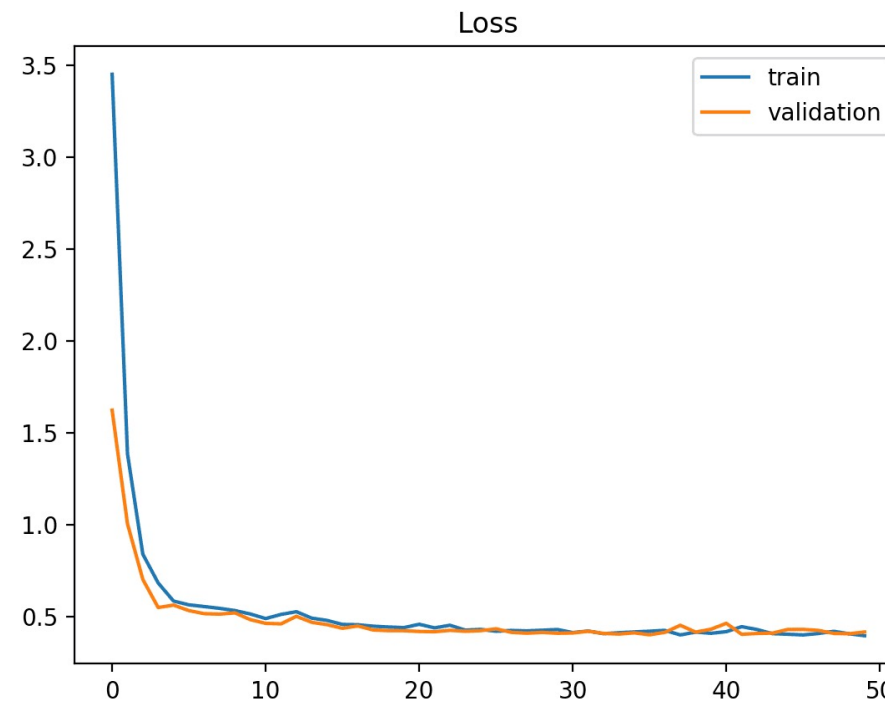
- La courbe *loss* pour l'entraînement continue de diminuer
- La courbe *loss* pour la validation diminue jusqu'à un certain point (pallier) et peut même recommencer à augmenter.

• Solutions :

- Ajout de données d'entraînement. Des données plus importantes et diversifiées aide à améliorer les performances du modèle (meilleure généralisation)
- Augmentation des données : génération de données, transformation des données existantes, etc.
- Early Stopping. Keras propose par exemple de pouvoir arrêter l'apprentissage lorsque des métriques (*loss*, *accuracy*, etc.) n'évoluent pas pendant un certain temps.
- Ajout de dropout : "bloquer" des neurones différents à chaque étape pendant l'apprentissage
 - (attention dropout utilisés lors de l'apprentissage et non pas avec le test de validation -> courbes un peu différentes et l'*accuracy* peut être moins bonne)



Exemples bon modèle (good fitting)

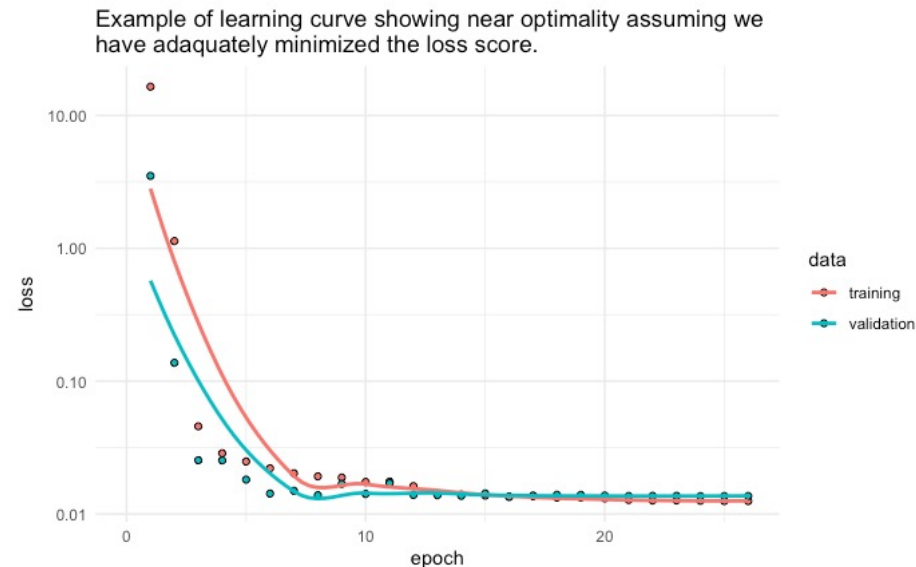


Un bon modèle

- La *loss* décroît vers un point de stabilité aussi bien pour le test que pour l'apprentissage
- L'écart entre les deux s'appelle « **écart de généralisation** »



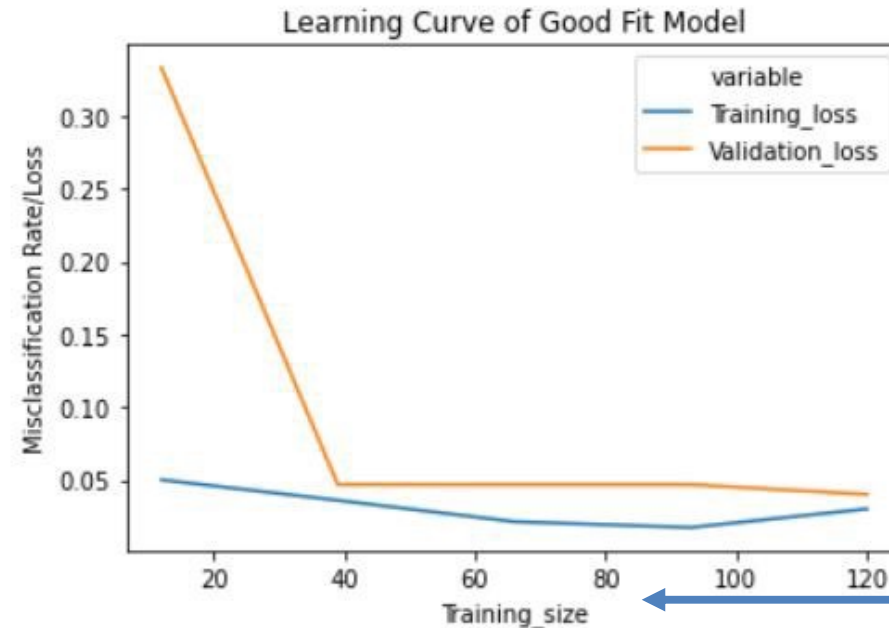
Exemples bon modèle (good fitting)



Un bon modèle

- La loss pour l'apprentissage et le test décroît vers un point stable avec un écart de généralisation très faible

Exemples bon modèle (good fitting)



Attention taille du jeu d'entraînement

Un bon modèle

- Au début la *loss* est assez faible pour l'entraînement
- La *loss* pour la validation diminue rapidement puis très progressivement lors de l'ajout d'exemples d'apprentissage.
- Par la suite elle s'aplatit (diminution de l'écart de généralisation)



Condition de good fitting

Symptômes :

- La courbe *loss* pour l'entraînement diminue jusqu'à un point de stabilité
- La courbe *loss* pour la validation diminue jusqu'à un point de stabilité et petit écart avec la *loss* pour l'entraînement

• Solutions :

- Tout va bien ! Bien penser à regarder la *loss* !

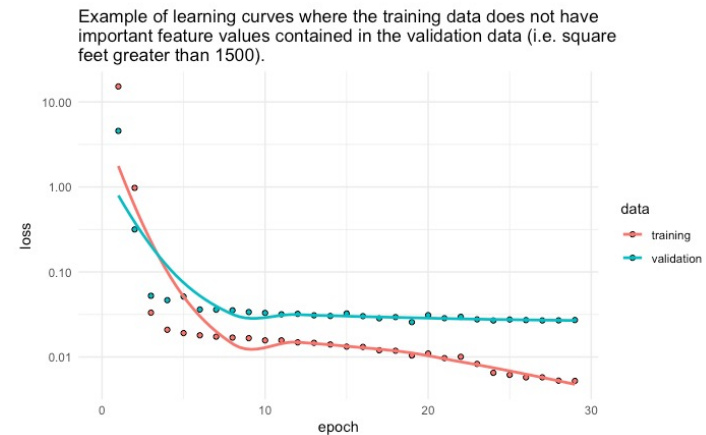
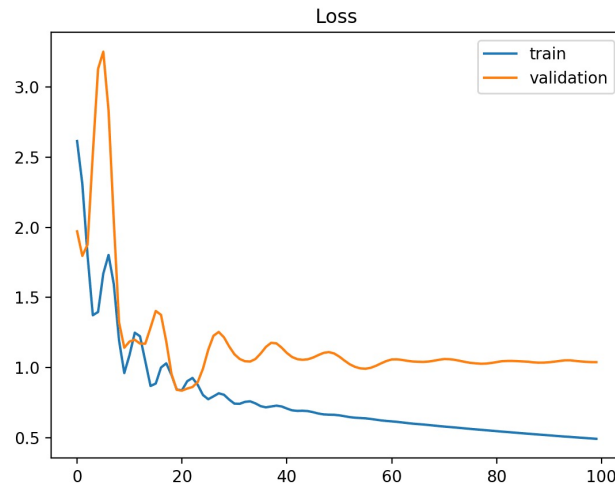


Jeux de données non représentatifs

- En apprentissage :
 - Les données d'apprentissage ne fournissent pas suffisamment d'informations pour apprendre par rapport à l'ensemble des données de validation utilisées pour évaluer
- En évaluation :
 - Les données d'évaluation ne fournissent pas suffisamment d'informations pour évaluer la capacité du modèle à généraliser



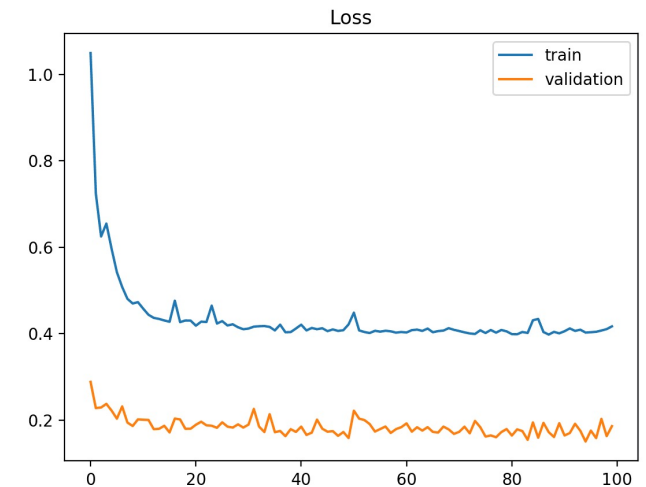
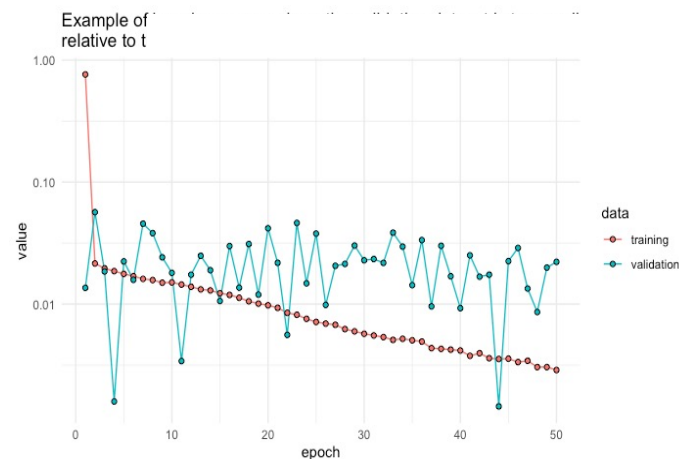
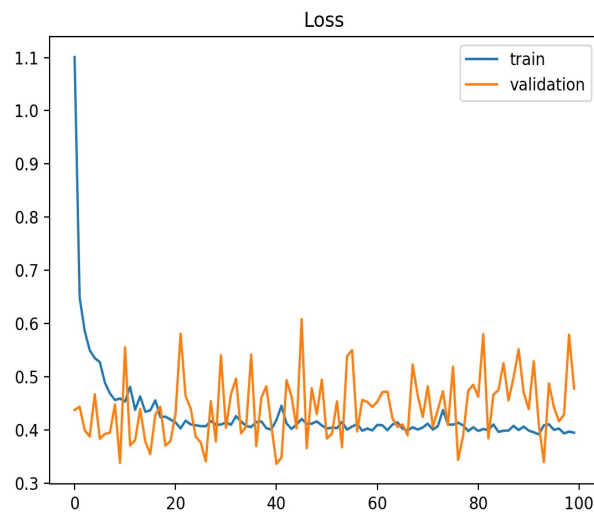
Jeux d'apprentissage non représentatifs



Amélioration mais écart de généralisation fort

- Les *loss* pour l'entraînement et la validation montrent une amélioration mais il reste un écart important entre les deux courbes

Jeux de validation non représentatifs



Grande variation ou loss validation inférieure

- La loss d'entraînement se comporte bien mais la *loss* en validation montre de grandes variations et pas ou peu d'amélioration
- La *loss* validation est inférieure à la *loss* d'apprentissage. L'ensemble de données de validation peut être plus facile à prédire pour le modèle que l'ensemble de données d'apprentissage



Jeux de données non représentatifs

Symptômes :

- Apprentissage : *loss* pour entraînement et validation avec une amélioration mais reste un écart important entre les deux courbes
- Validation : *loss* validation avec grande variation ou en dessous de la *loss* entraînement

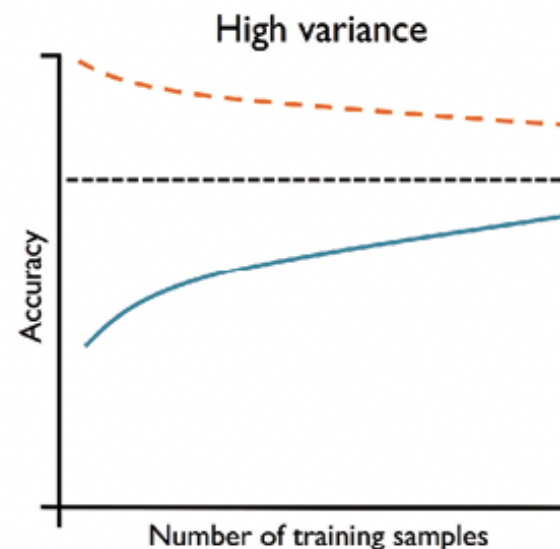
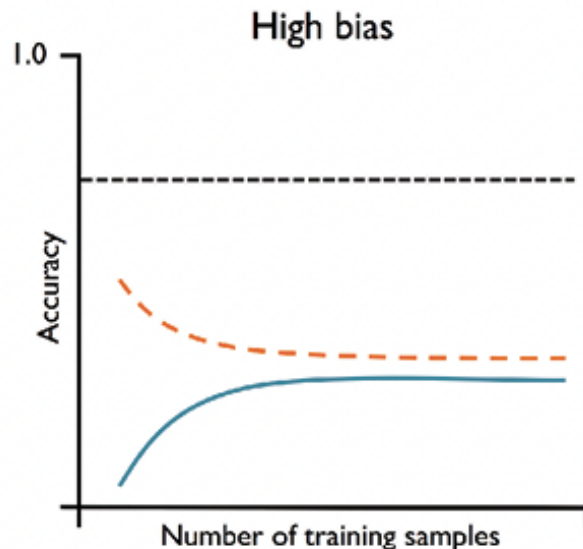
Solutions :

- Ajouter plus de données. Augmenter les données
- Vérifier que l'échantillonnage est bien fait. Si classes déséquilibrées vérifier que les échantillons sont représentatifs (*stratify=y* dans *train_test_split*)
- Faire de la cross validation pour être certain que toutes les données puissent être dans le jeu d'apprentissage et de validation



Les courbes d'apprentissage

- Scikit learn propose une méthode *learning_curve*
 - *Un estimateur (classifieur)*
 - *Un jeu de données*
 - *Une cross validation*
 - *Les tailles du jeu d'apprentissage*



Exemple d'appel

```
dataset = datasets.load_digits()

# X contient les variables prédictives et y la variable à prédire
X, y = dataset.data, dataset.target

# normalisation du jeu de données
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
```

```
cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=0)
estimator=LogisticRegression(solver='lbfgs', random_state=1, max_iter=1000)
train_sizes, train_scores, test_scores = learning_curve(estimator=estimator, X=X, y=y,
                                                         cv=cv, train_sizes=np.linspace(0.1, 1.0, 20),
                                                         n_jobs=1)

plot_learningcurve(train_scores, test_scores)
```

Estimateur = LogisticRegression

Jeu de données : X – y

Cross validation : 10

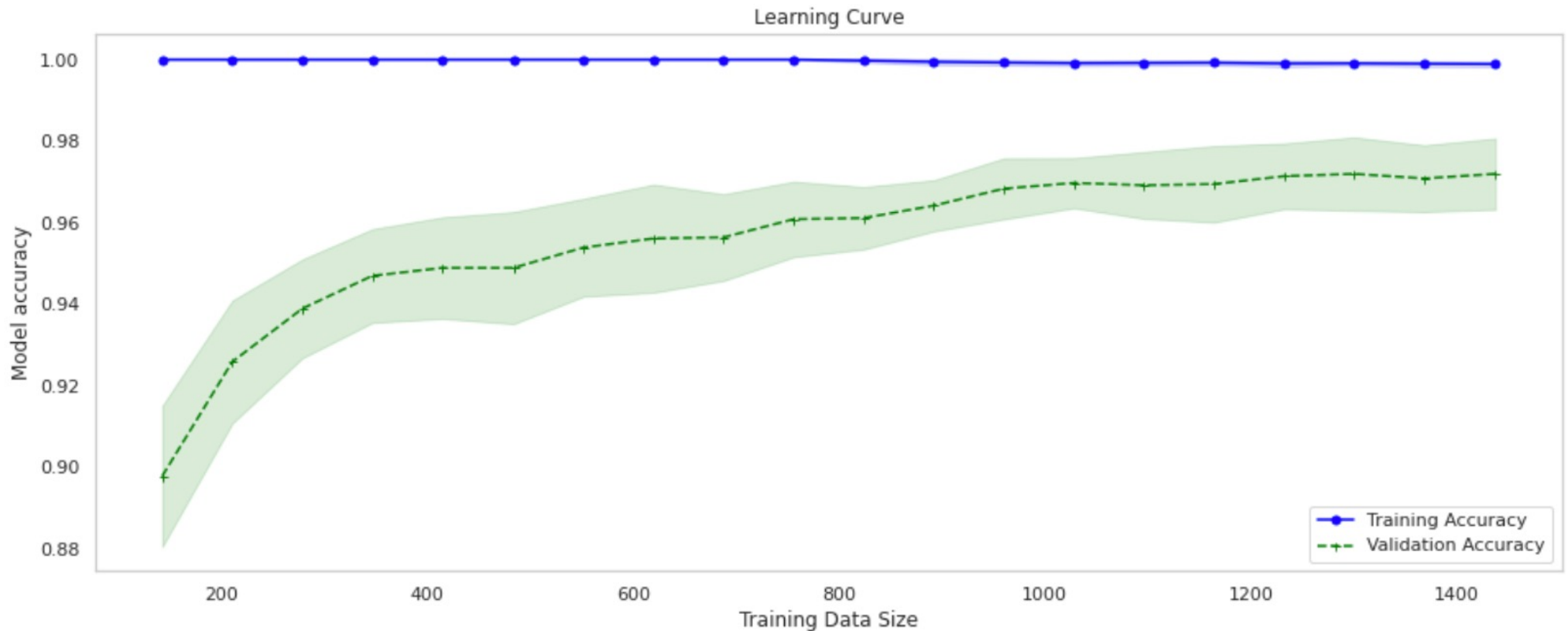
Taille du jeu d'apprentissage à faire varier :

np.linspace(start=0.1, end=1.0, num=20)

génération de 20 échantillons



Avec Logistic Regression

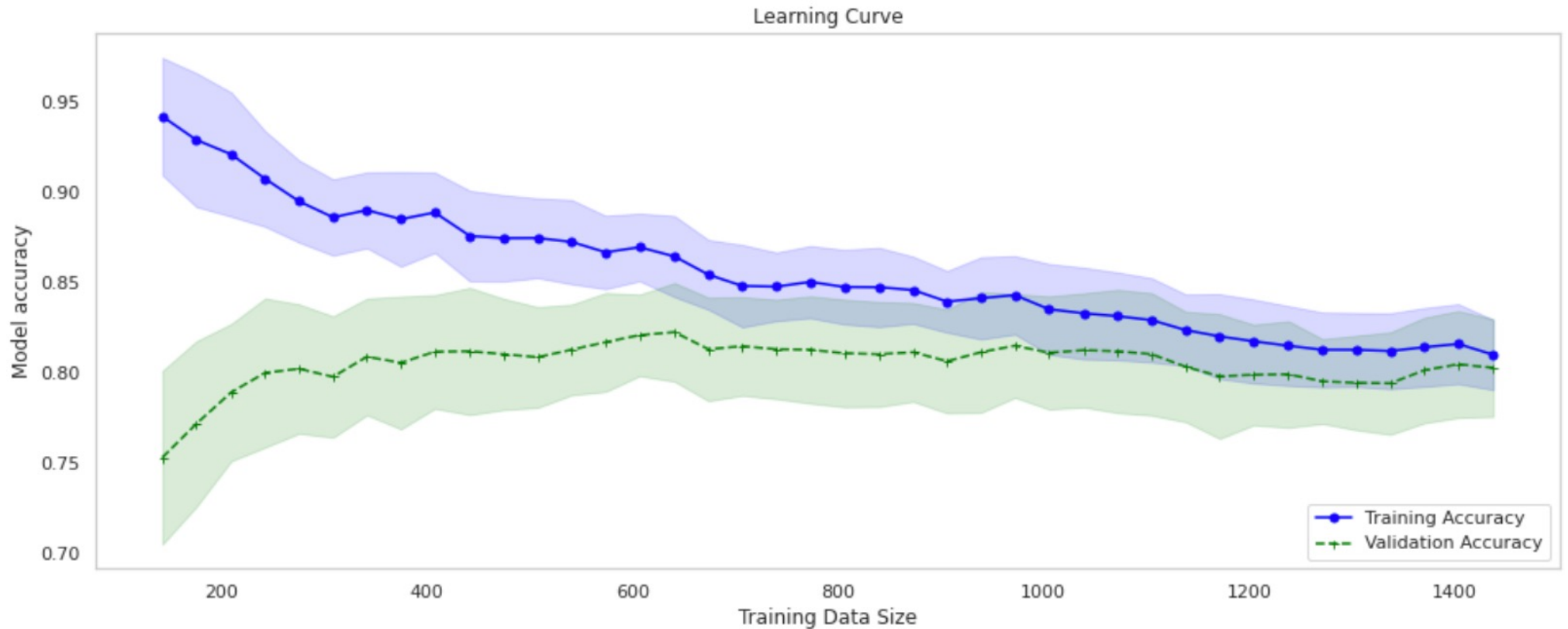


Underfitting

- Quelque soit la taille du jeu de données d'apprentissage le score sur l'apprentissage ne change pas
- Pour le jeu de validation un palier est obtenu

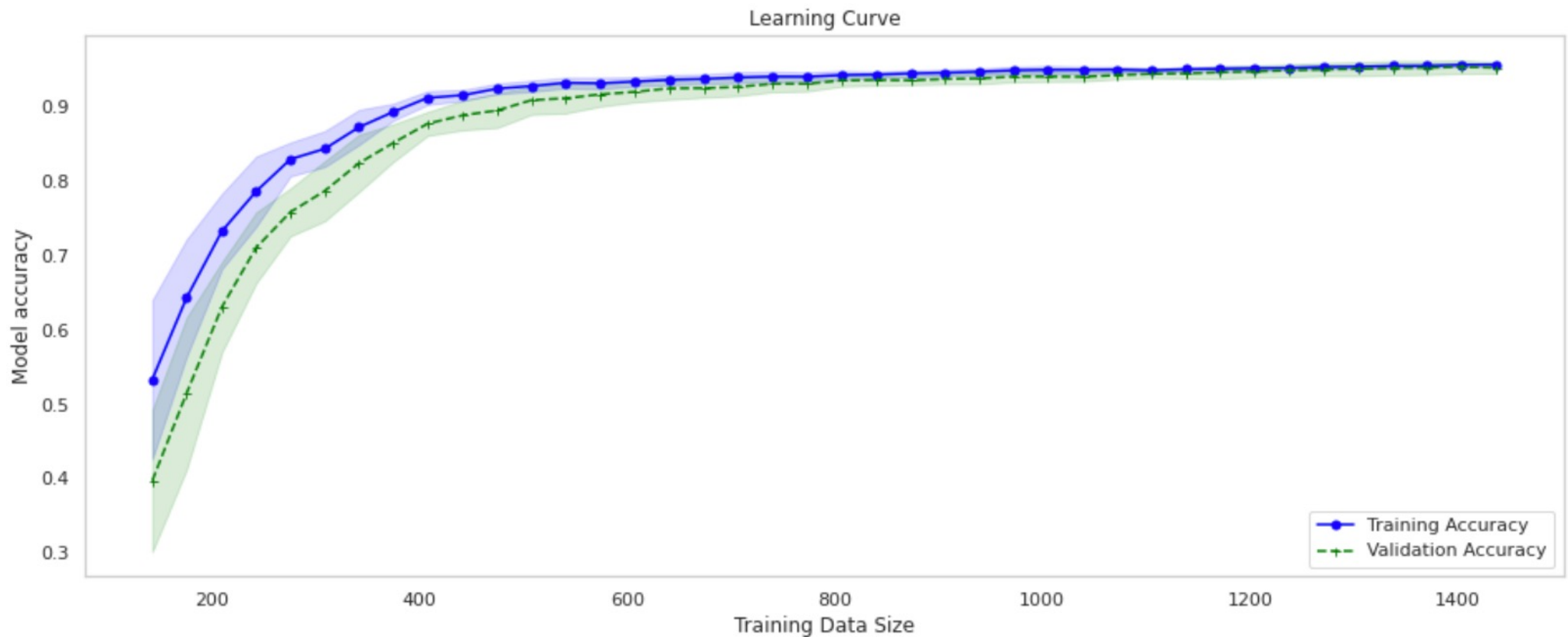


Avec Naïve Bayes



- Le score de validation et le score d'apprentissage convergent vers une valeur assez faible avec l'augmentation de la taille de l'ensemble d'apprentissage
- Avoir plus de données d'entrainement ne serait pas utile

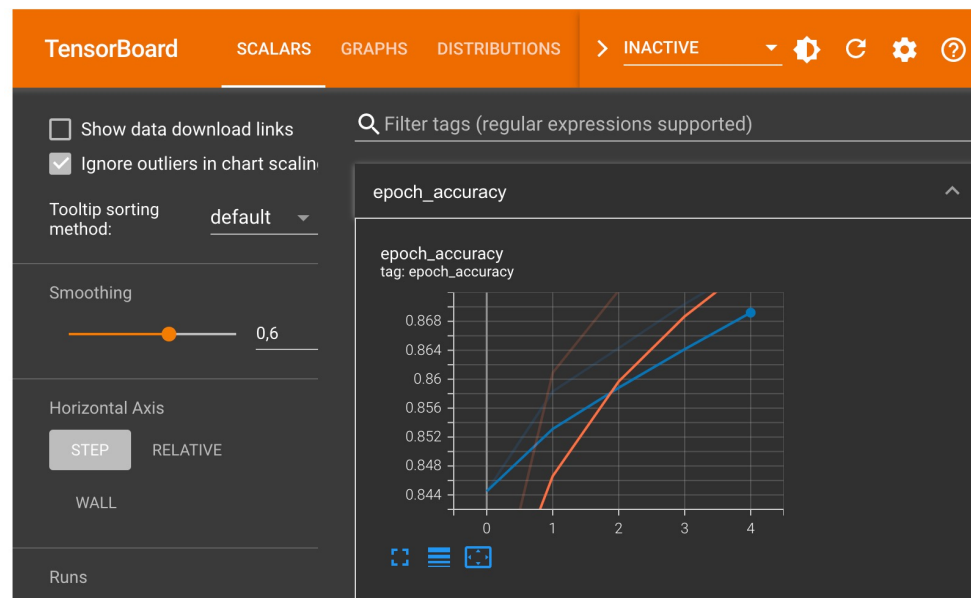
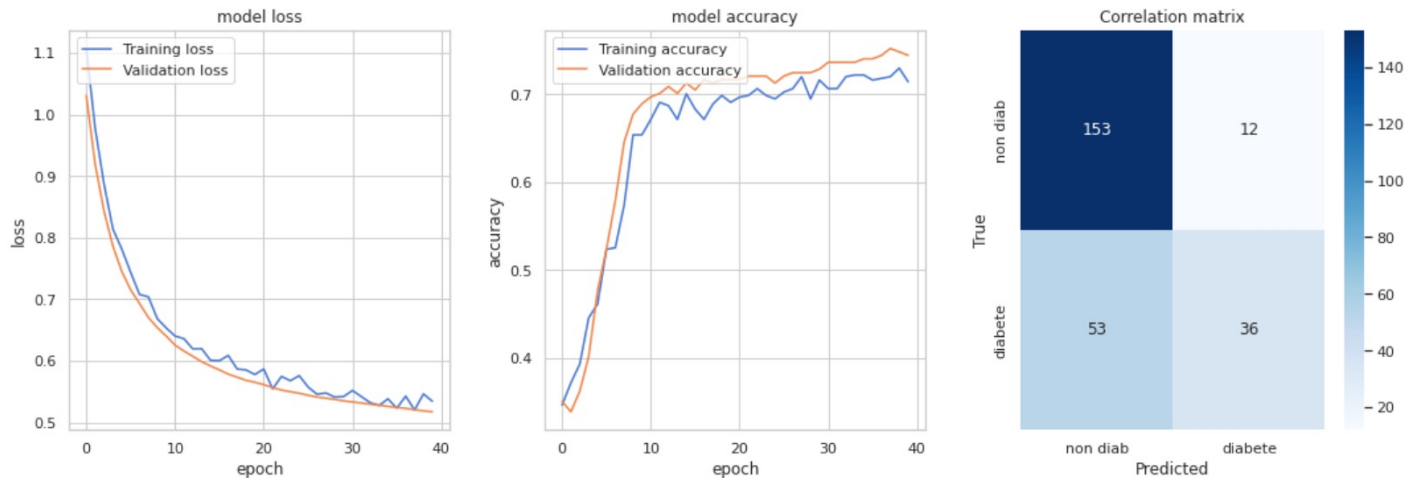
Avec SVM



- Plus la taille de l'ensemble d'apprentissage augmente, plus la courbe de score d'apprentissage et la courbe de score de validation croisée convergent
- L'ajout de plus de données d'apprentissage augmentera sans doute la généralisation

En apprentissage profond

- Voir le notebook `overfitting_underfitting`



-
- Des questions ?

