

# EPISEN – ING3. SI

## Machine Learning



**Abdallah EL HIDALI**

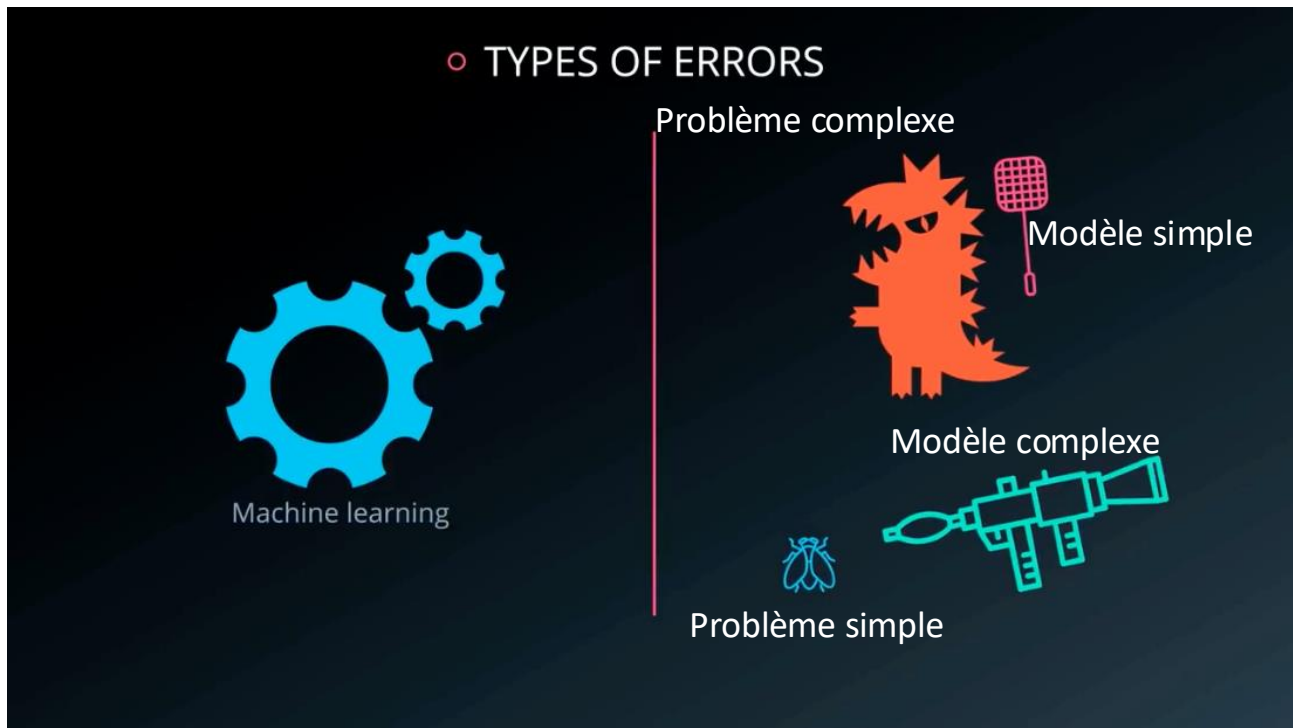
Tech Lead Sita For Aircraft  
abdallah.el-hidali@sit.aero

**EPISEN**

2024/2025

# **IX. Training & Tuning**

# Les types d'erreurs



# Les types d'erreurs

## ○ TYPES OF ERRORS

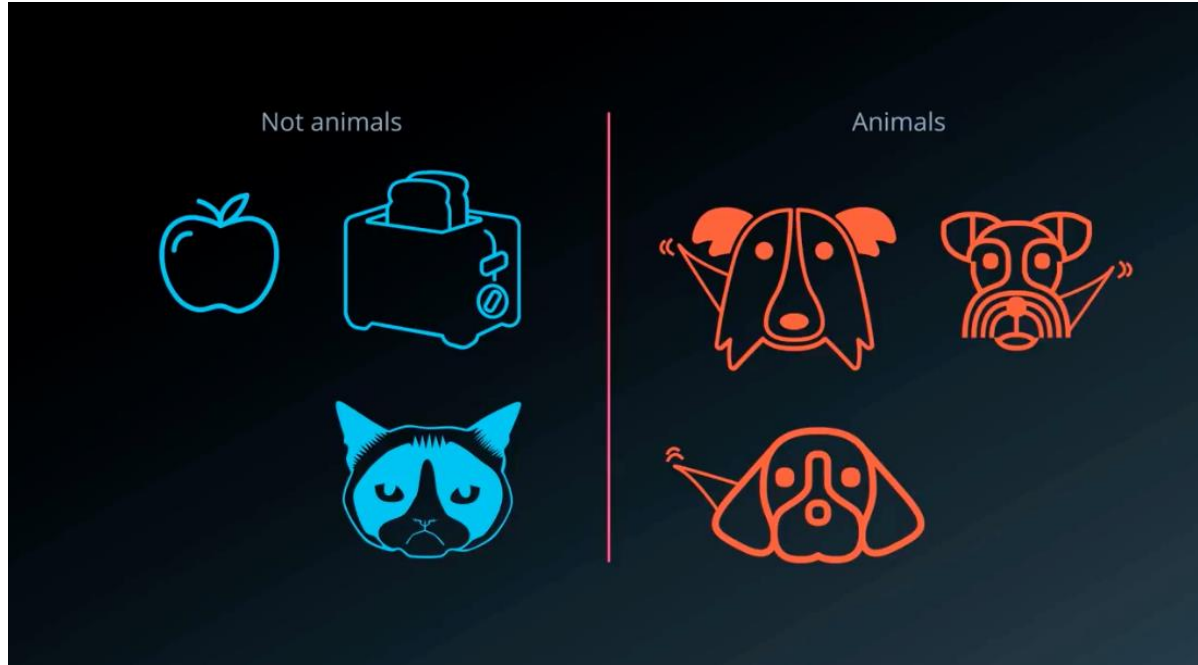
UNDERFITTING  
Sous-apprentissage



OVERFITTING  
Sur-apprentissage

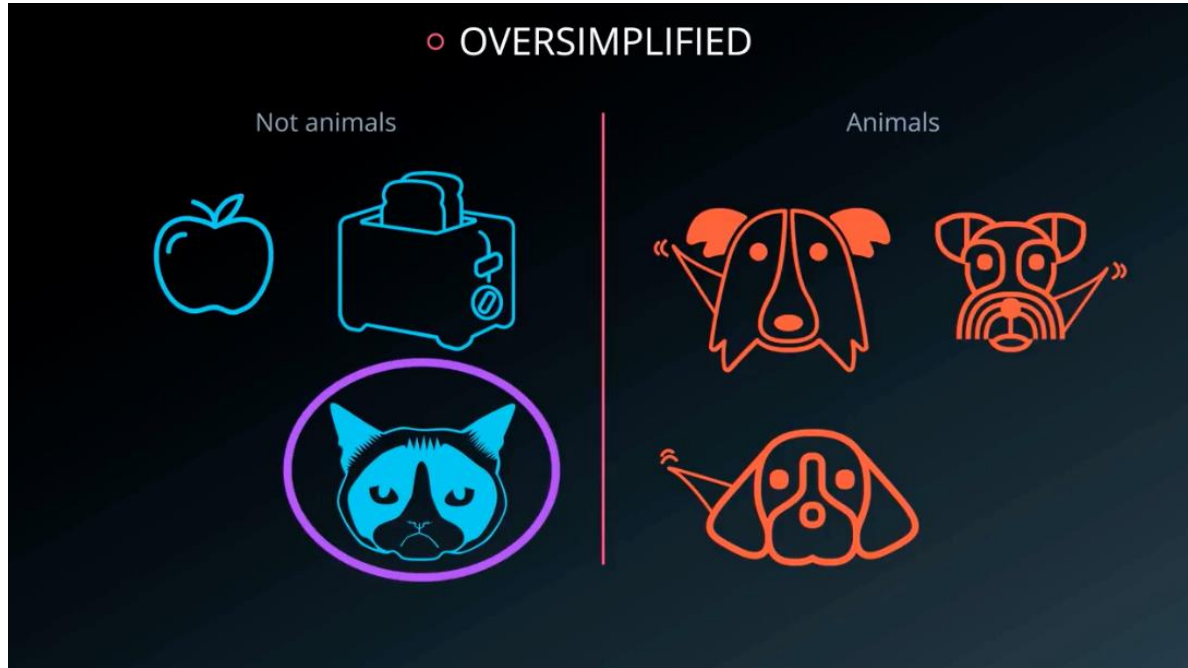


# Les types d'erreurs



Il s'agit d'un problème de classification binaire, où nous divisons notre ensemble de données en deux catégories : la première classe regroupe les animaux et la seconde classe inclut tout ce qui n'est pas un animal.

# Les types d'erreurs



Ce modèle démontre un cas typique de sous-apprentissage (underfitting). Sa simplicité excessive par rapport à la complexité de la tâche de classification se manifeste par des erreurs flagrantes, comme l'incapacité à reconnaître un chat comme un animal.

# Le sous-apprentissage (underfitting)



# Les types d'erreurs

## ○ OVERCOMPLICATED

Aucun chien qui remue la queue



Les chiens qui remuent la queue



Ce modèle est très spécifique



# Les types d'erreurs

## ○ OVERCOMPLICATED

Aucun chien qui remue la queue



Les chiens qui remuent la queue



Un chien statique sera mal étiqueté par notre modèle qui se base uniquement sur le mouvement de la queue comme critère de classification → on parle de sur-apprentissage (overfitting)

# Le sur-apprentissage (overfitting)

## ○ OVERFITTING

Il performe bien dans l'ensemble d'entraînement, mais il a tendance à mémoriser plutôt qu'à apprendre les caractéristiques.

Erreur due à la variance

Aucun chien qui remue la queue



Les chiens qui remuent la queue



# Recap

## TRADEOFF

### High bias (underfitting)



Oversimplify the problem

Bad on training set

Bad on testing set

### Good Model



Good model

Good on training set

Good on testing set

### High variance (overfitting)

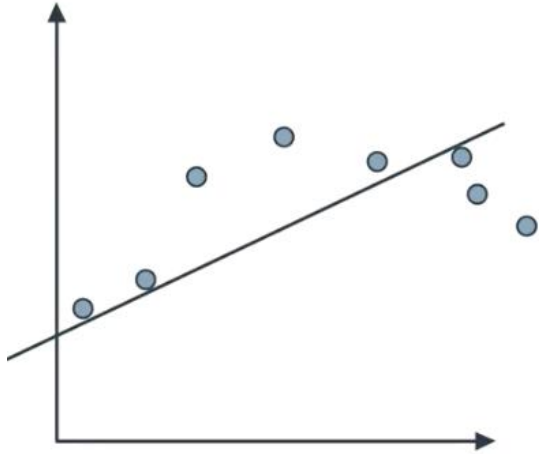


Overcomplicate the problem

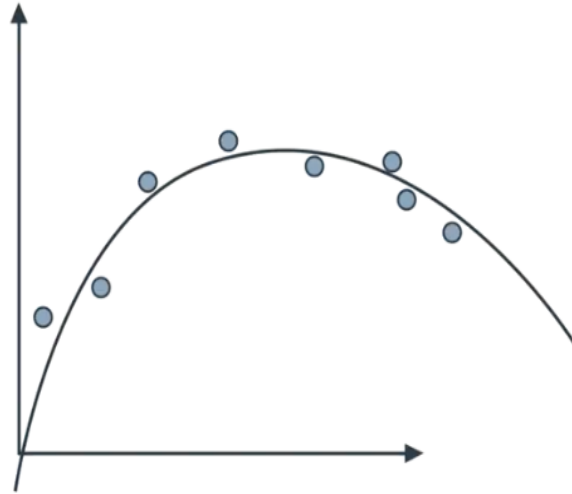
Great on training set

Bad on testing set

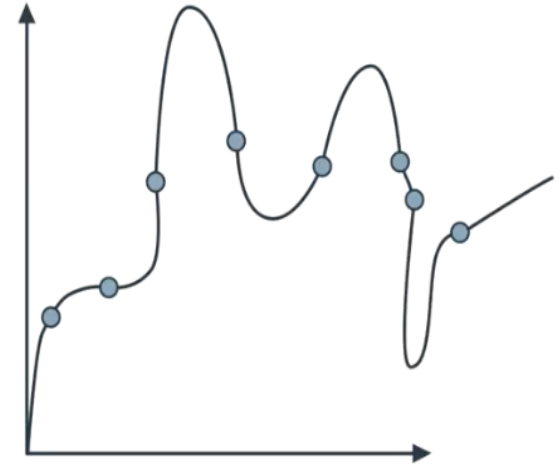
# Recap



Underfitting  
Haut biais



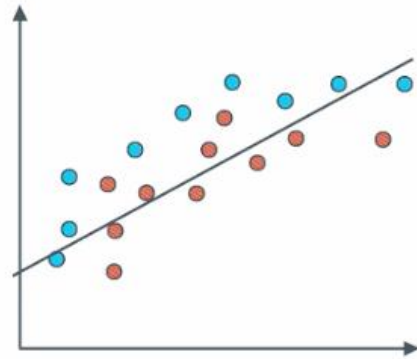
Bon modèle



Overfitting  
Haute variance

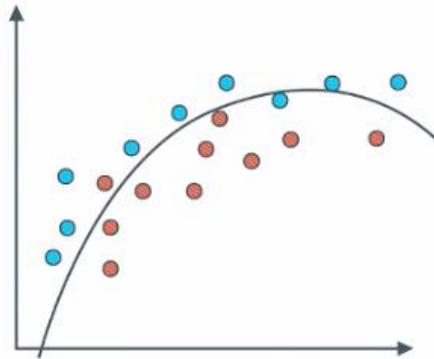
# Graphique de Complexité du Modèle

## ○ MODEL COMPLEXITY GRAPH



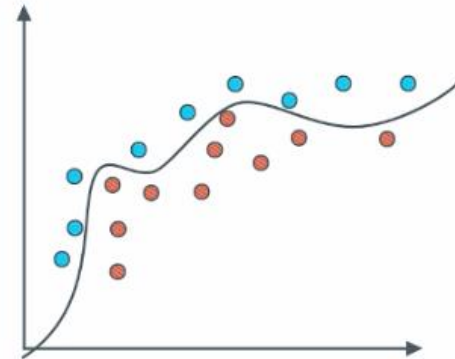
HIGH BIAS

Degree = 1



POLYNOMIAL

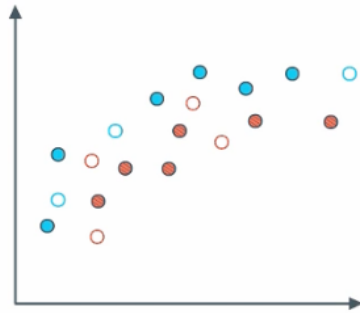
Degree = 2



HIGH VARIANCE

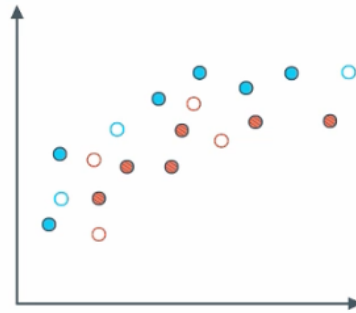
Degree = 6

# Graphique de Complexité du Modèle



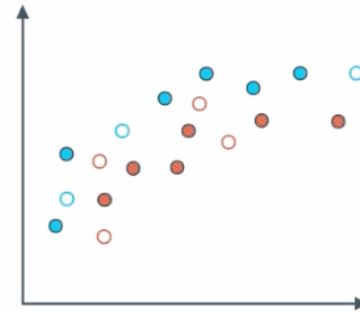
LINEAR MODEL

Degree = 1



QUADRATIC MODEL

Degree = 2

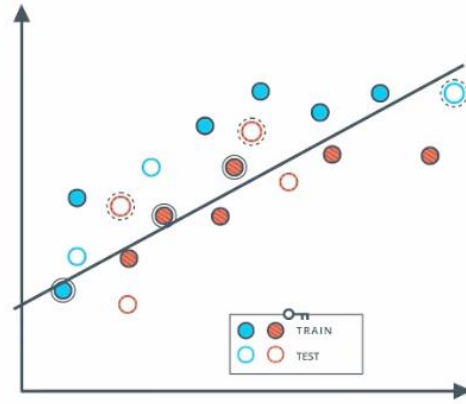


POLYNOMIAL MODEL

Degree = 6



# Graphique de Complexité du Modèle



LINEAR MODEL

Degree = 1

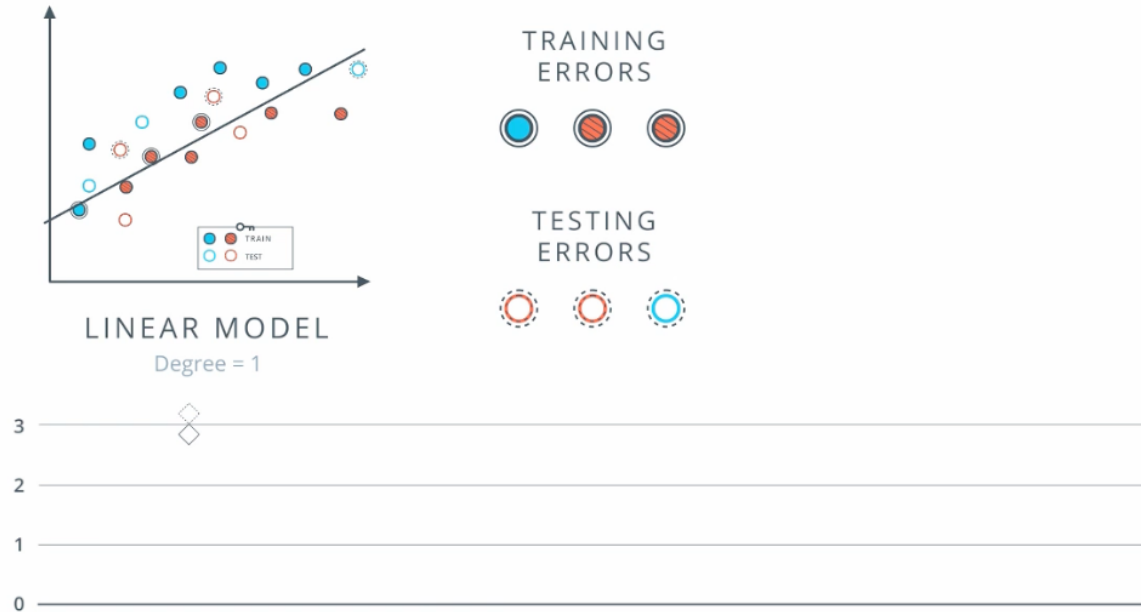
TRAINING  
ERRORS



TESTING  
ERRORS



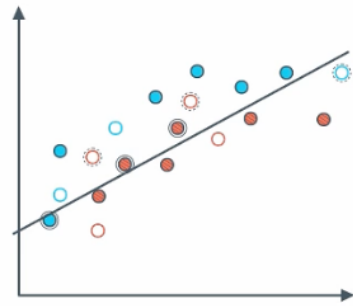
# Graphique de Complexité du Modèle



Nous calculons l'erreur du modèle à la fois sur l'ensemble d'entraînement et l'ensemble de test, puis nous les représentons sur un graphique.

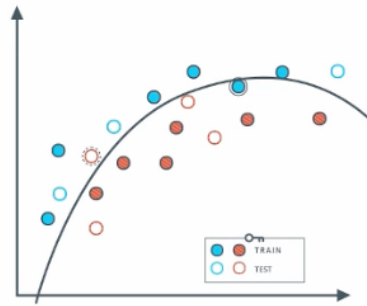


# Graphique de Complexité du Modèle



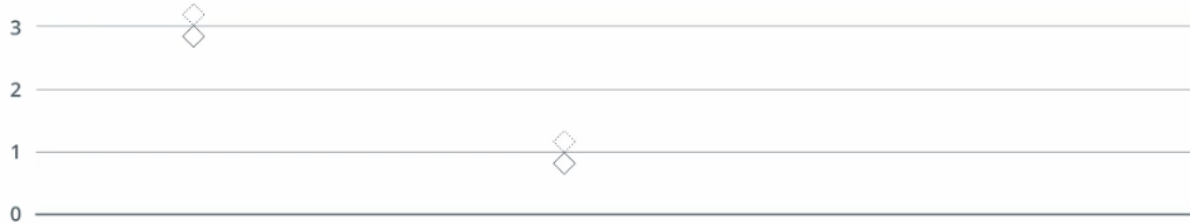
LINEAR MODEL

Degree = 1

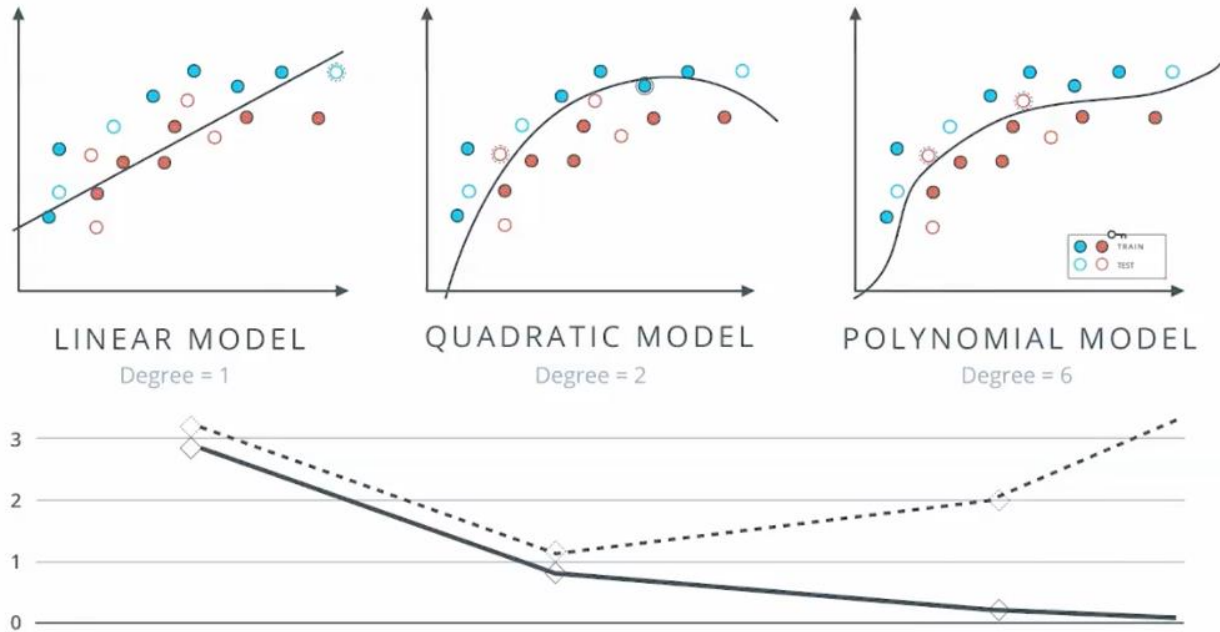


QUADRATIC MODEL

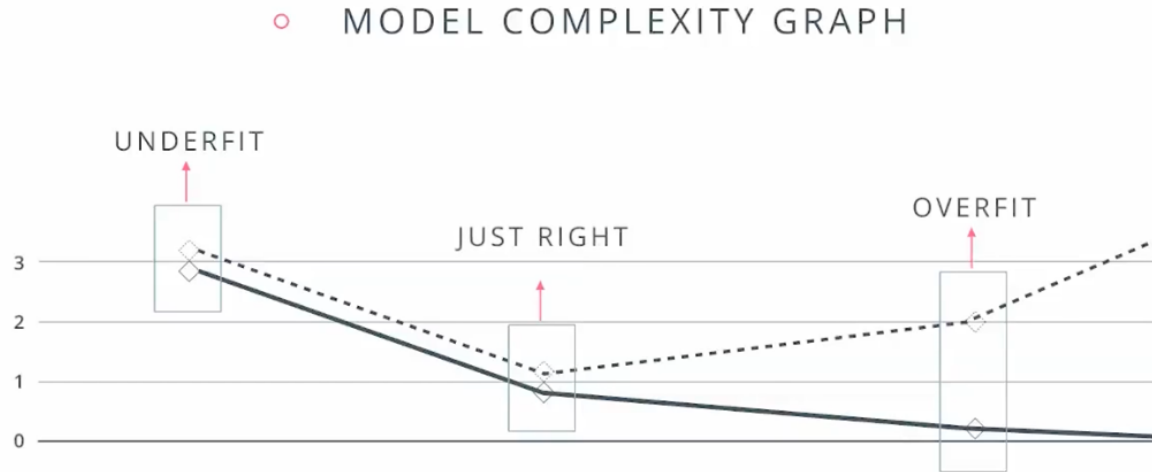
Degree = 2



# Graphique de Complexité du Modèle

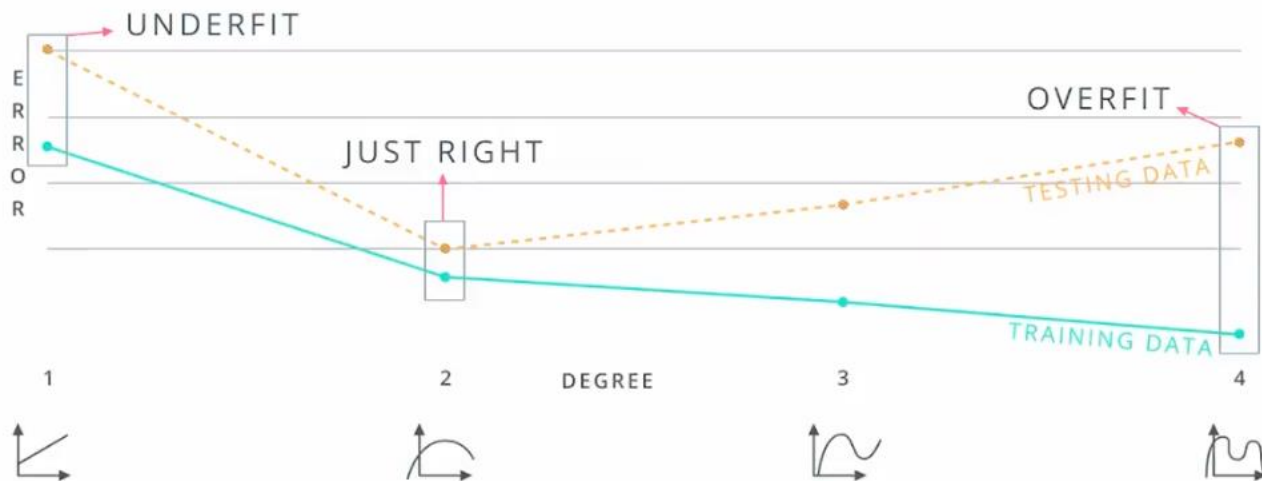


# Graphique de Complexité du Modèle



# Graphique de Complexité du Modèle

○ MODEL COMPLEXITY GRAPH



# Graphique de Complexité du Modèle

## ○ MODEL COMPLEXITY GRAPH



On ne doit jamais utiliser tes données de test pour l'entraînement.

# Graphique de Complexité du Modèle

## ○ MODEL COMPLEXITY GRAPH



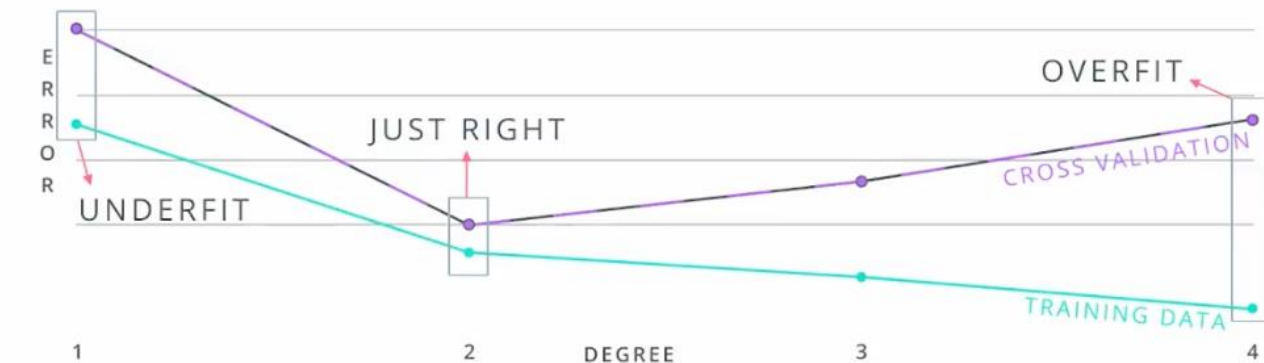
Nous avons utilisé les données de test pour évaluer si notre modèle souffre de sous-apprentissage, de sur-apprentissage ou s'il est correctement ajusté. Cependant, il est important de noter que les données de test devraient être réservées pour la phase finale, afin d'évaluer comment le modèle se comportera dans des conditions réelles.

# La Validation Croisée (Cross Validation )

## ○ SOLUTION: CROSS VALIDATION

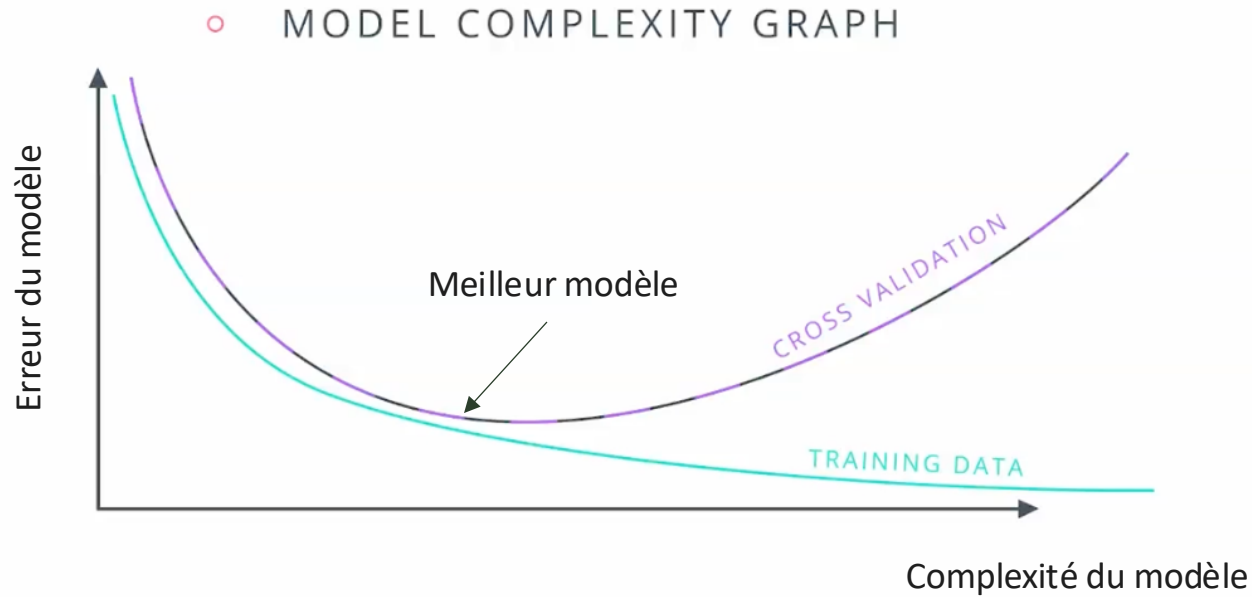


# La Validation Croisée (Cross Validation )



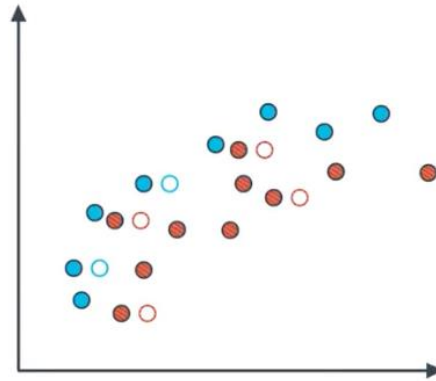


# Graphique de Complexité du Modèle



# La Validation Croisée K-Fold (K-Fold Cross Validation)

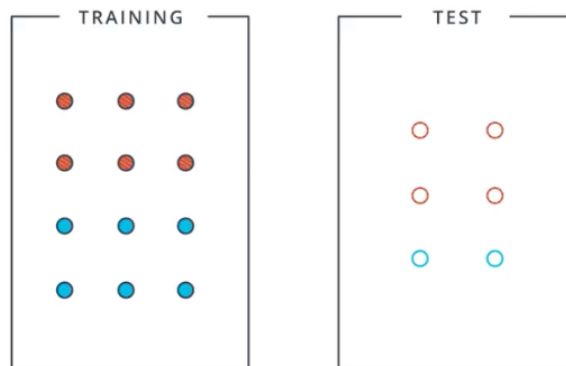
○ K-FOLD CROSS VALIDATION



Nous avons un ensemble de données que nous divisons en deux parties : un ensemble d'entraînement et un ensemble de test.

# La Validation Croisée K-Fold (K-Fold Cross Validation)

## ○ K-FOLD CROSS VALIDATION

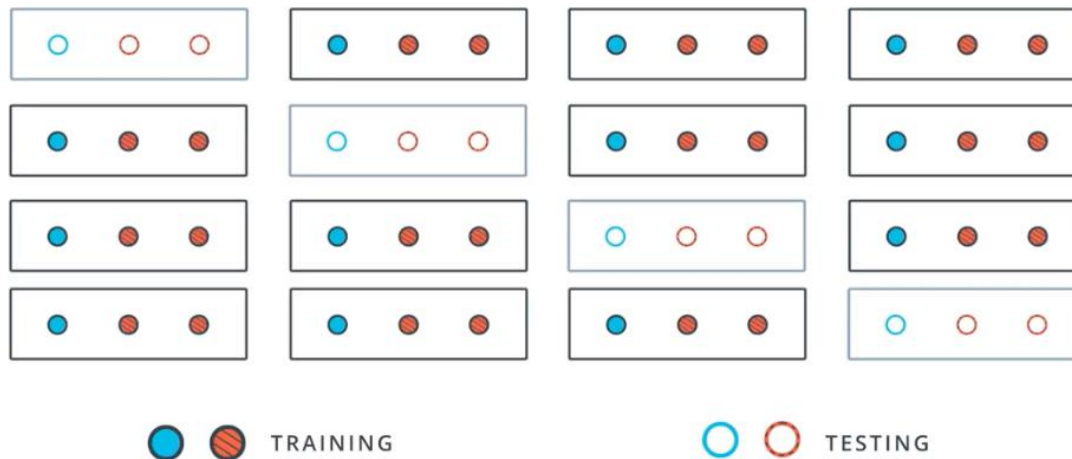


How do we not 'lose' the training data?

Lorsqu'on divise les données en ensemble d'entraînement et ensemble de test, un risque est de ne pas utiliser une partie des données lors de l'apprentissage. La question qui se pose est de savoir s'il existe un moyen de séparer les données sans introduire de biais.

# La Validation Croisée K-Fold (K-Fold Cross Validation)

○ K-FOLD CROSS VALIDATION



L'idée est de diviser les données en K sous-ensembles (par exemple, K=4). Le modèle est ensuite entraîné sur K-1 sous-ensembles et évalué sur le sous-ensemble restant. Ce processus est répété pour chaque sous-ensemble, et la performance du modèle est calculée en moyenne sur tous les sous-ensembles.

# La Validation Croisée K-Fold (K-Fold Cross Validation)

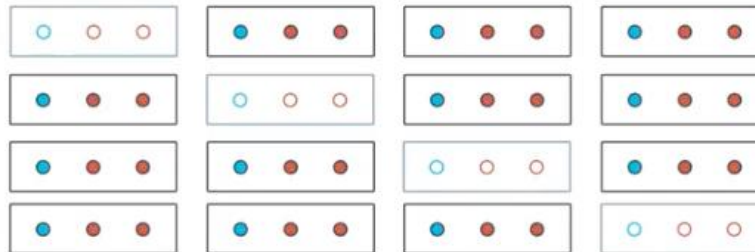
## ○ CROSS VALIDATION IN SKLEARN

```
from sklearn.model_selection import KFold
```

`kf = KFold(12, 3)` 12 correspond à la taille totale des données, tandis que 3 représente la taille de l'ensemble de test.

```
for train_indices, test_indices in kf:  
    print train_indices, test_indices
```

```
[3 4 4 6 7 8 9 10 11] [0 1 2]  
[0 1 2 6 7 8 9 10 11] [3 4 5]  
[0 1 2 3 4 5 9 10 11] [6 7 8]  
[0 1 2 3 4 5 6 4 8 ] [9 10 11]
```



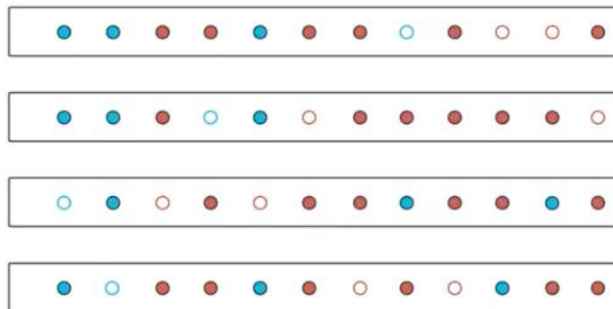
# La Validation Croisée K-Fold (K-Fold Cross Validation)

## ○ CROSS VALIDATION IN SKLEARN

```
from sklearn.model_selection import KFold  
kf = KFold(12, 3, shuffle = True)
```

```
for train_indices, test_indices in kf:  
    print train_indices, test_indices
```

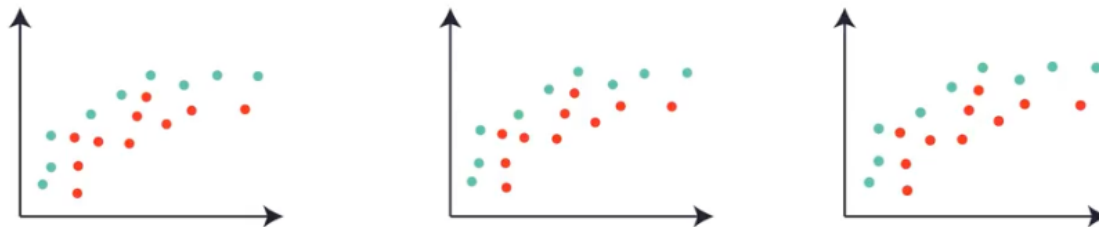
```
[0 1 2 3 4 5 6 8 11] [7 9 10]  
[0 1 2 4 6 7 8 9 10] [3 5 11]  
[1 3 5 6 7 8 9 10 11] [0 2 4]  
[0 2 3 4 5 7 9 10 11] [1 6 8]
```



Il est également conseillé de mélanger le jeu de données de manière aléatoire afin de supprimer tout type de biais.

# Les Courbes d'Apprentissage (Learning Curves)

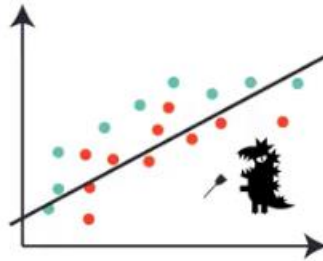
## ◦ LEARNING CURVES



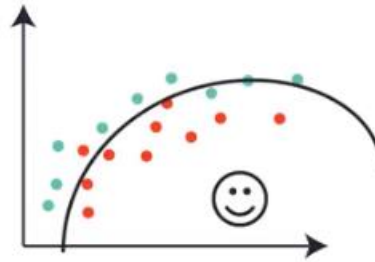
- L'objectif est de trouver une méthode pour identifier le sous-apprentissage ou le sur-apprentissage.
- Ici, nous avons trois copies du même ensemble de données, et l'idée est d'entraîner trois modèles différents.

# Les Courbes d'Apprentissage (Learning Curves)

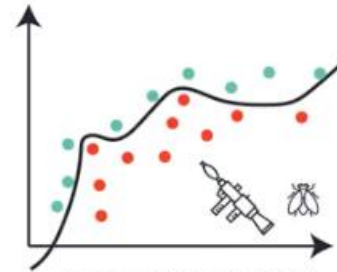
## ◦ LEARNING CURVES



HIGH BIAS  
DEGREE = 1



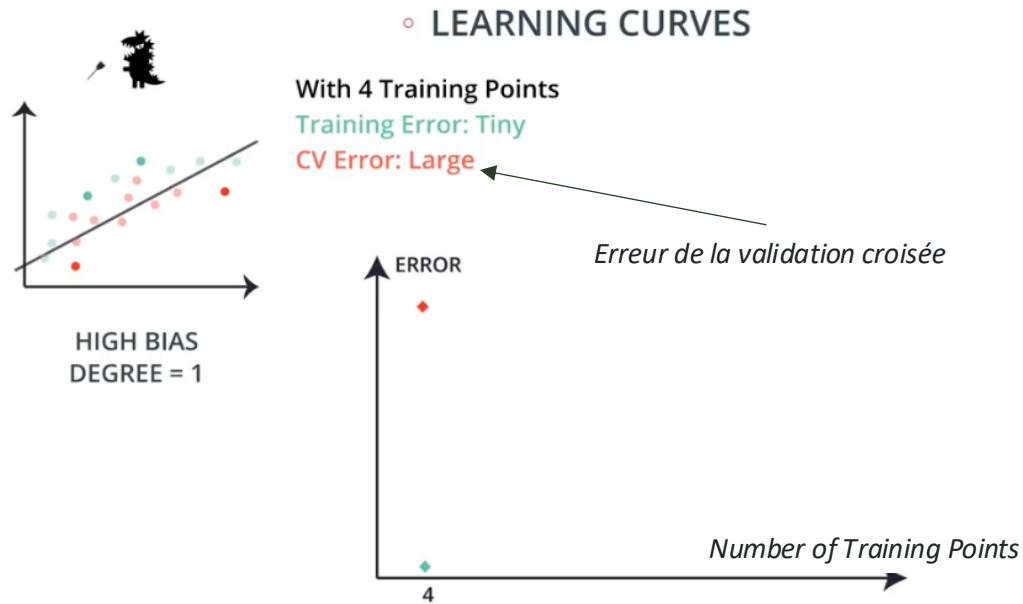
JUST RIGHT  
DEGREE = 2



HIGH VARIANCE  
DEGREE = 6

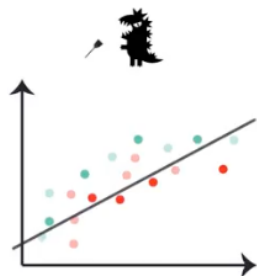


# Les Courbes d'Apprentissage (Learning Curves)



# Les Courbes d'Apprentissage (Learning Curves)

## ◦ LEARNING CURVES



HIGH BIAS  
DEGREE = 1

With 4 Training Points

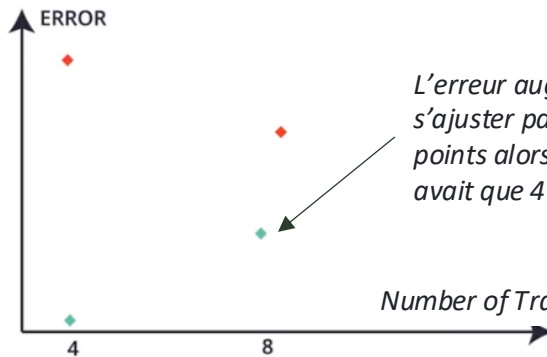
Training Error: Tiny

CV Error: Large

With 8 Training Points

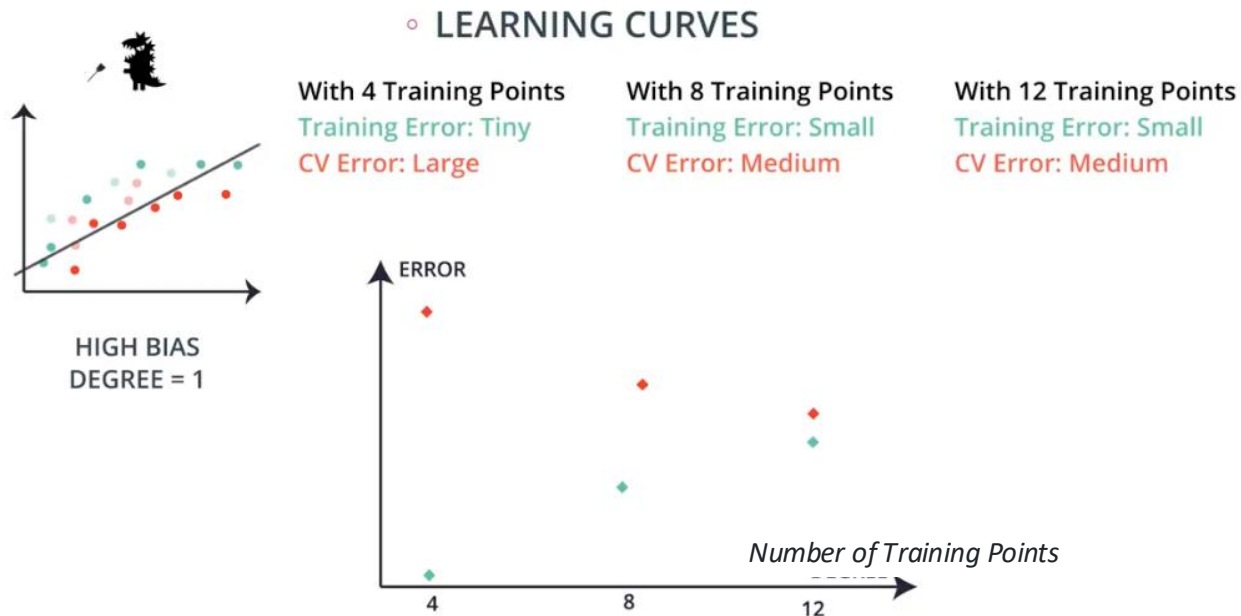
Training Error: Small

CV Error: Medium

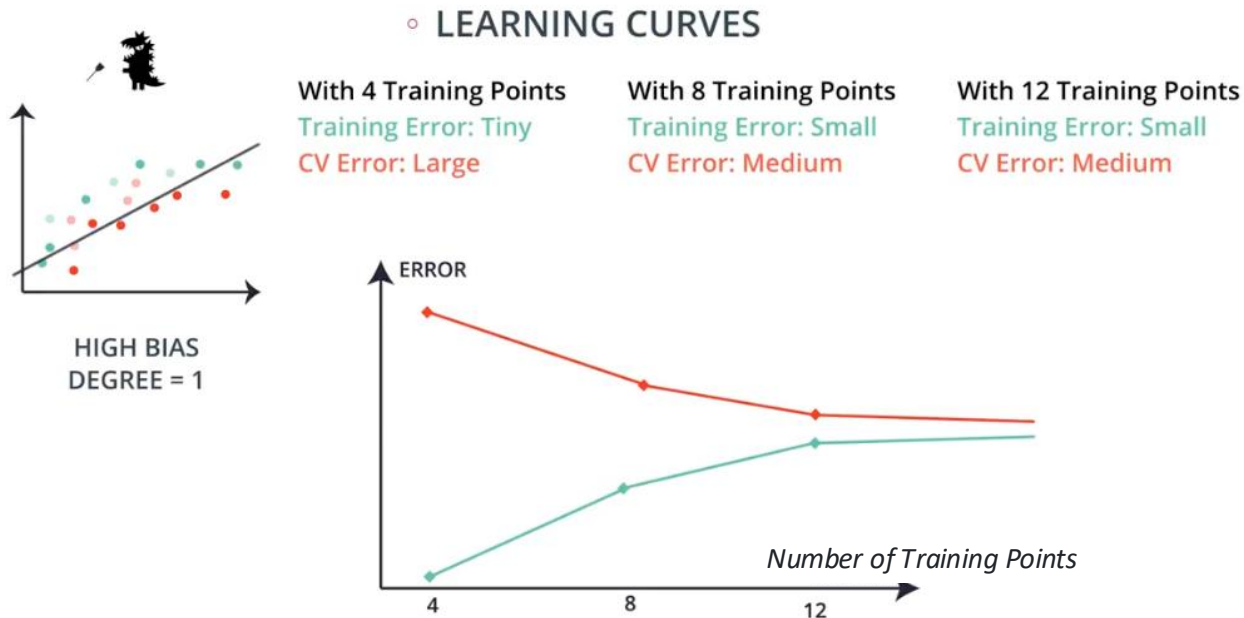


*L'erreur augmente car on doit  
s'ajuster par rapport à 8  
points alors que avant on  
avait que 4 points*

# Les Courbes d'Apprentissage (Learning Curves)

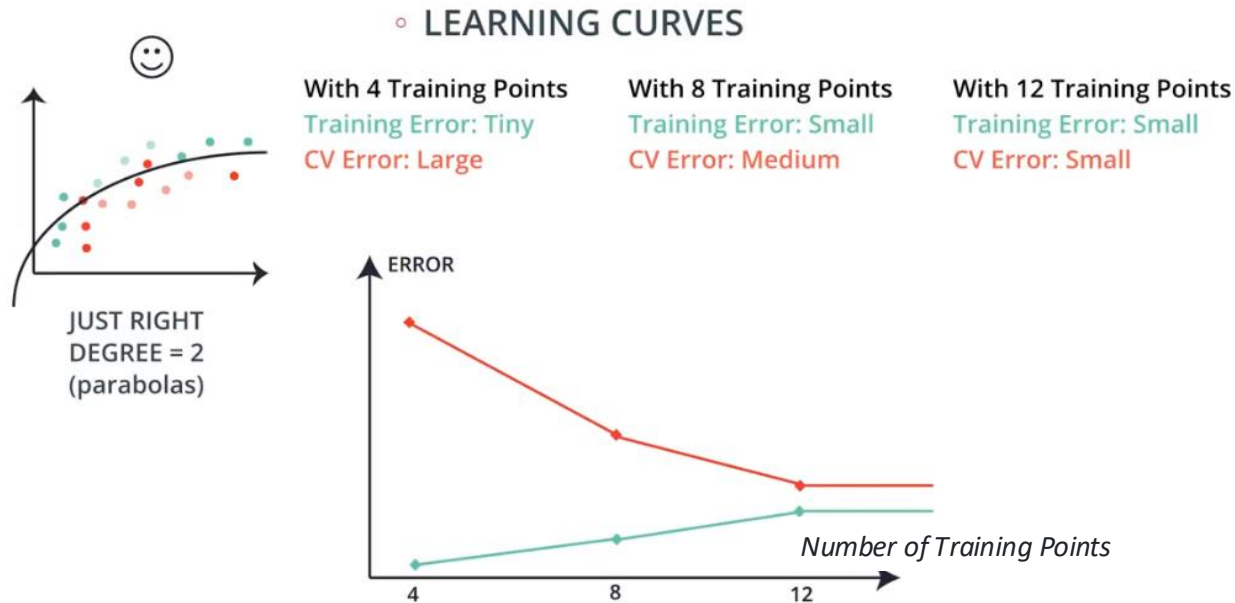


# Les Courbes d'Apprentissage (Learning Curves)



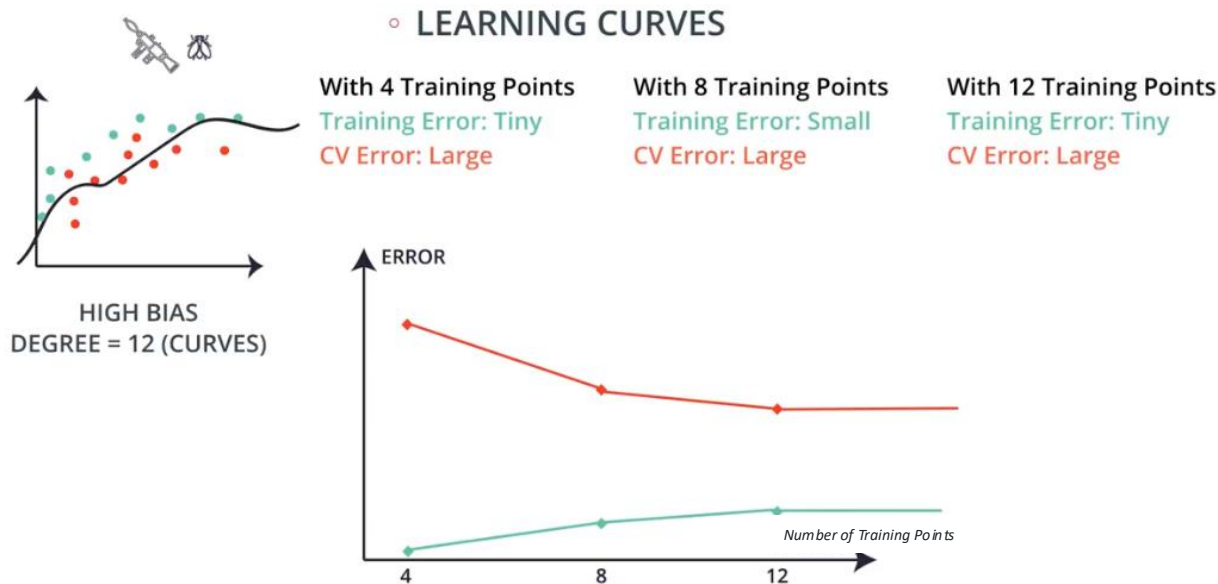
- À mesure que le nombre de points augmente, l'erreur d'entraînement s'accroît. Ce modèle ne performe pas bien sur les données d'apprentissage, ce qui indique **un sous-apprentissage**.
- L'erreur sur l'ensemble d'entraînement demeure significativement plus élevée que celle de l'ensemble de test.

# Les Courbes d'Apprentissage (Learning Curves)



À mesure que le nombre de points augmente, l'erreur sur l'ensemble d'entraînement et l'ensemble de test converge vers une valeur faible.

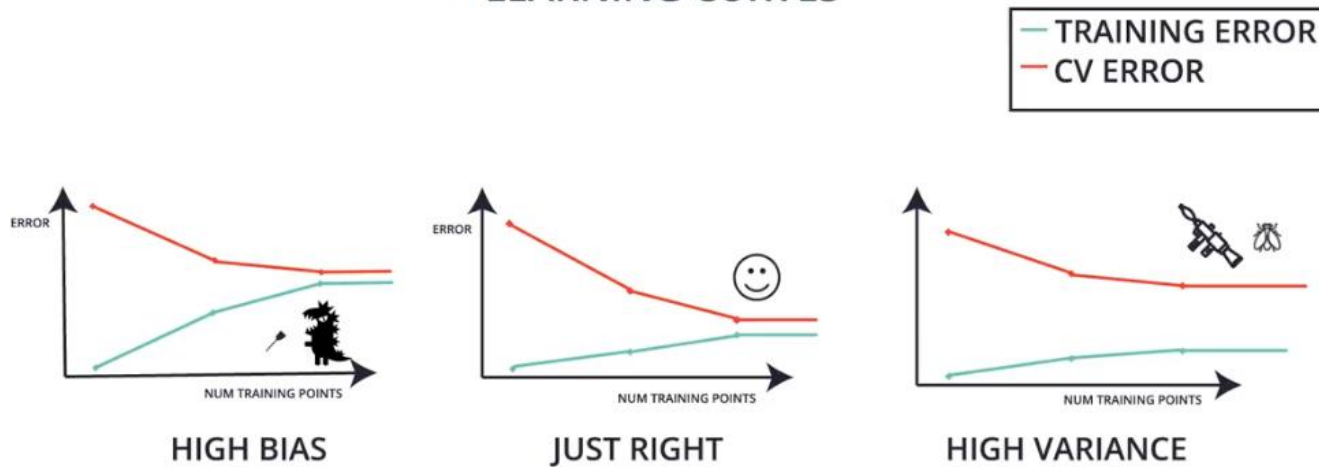
# Les Courbes d'Apprentissage (Learning Curves)



L'erreur sur l'ensemble d'entraînement et l'erreur sur l'ensemble de test ne convergent pas vers le même point, et il existe toujours un écart significatif entre les deux. On peut donc dire que le modèle souffre **de sur-apprentissage**.

# Les Courbes d'Apprentissage (Learning Curves)

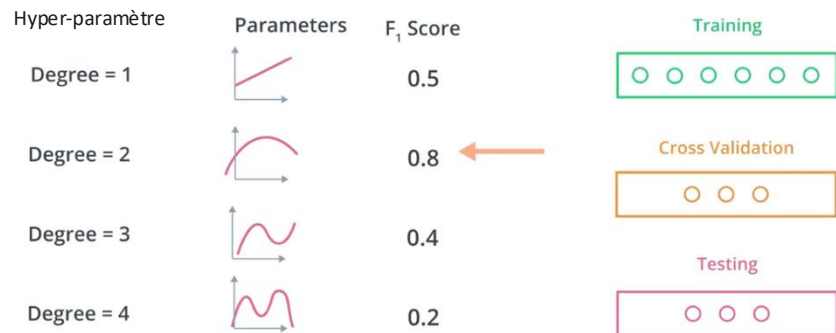
## ◦ LEARNING CURVES



On peut déterminer si un modèle souffre de sous-apprentissage ou de sur-apprentissage en analysant la forme des courbes d'apprentissage.

# Grid Search

## ◦ TRAINING A LOGISTIC REGRESSION MODEL



Dans le cadre d'un problème de classification, nous optons pour l'utilisation d'un modèle de régression logistique. Notre approche consiste à expérimenter avec différents degrés polynomiaux pour le modèle. Pour chaque degré testé, nous évaluons la performance du modèle en calculant son score F1. Le processus de sélection du modèle optimal s'effectue par validation croisée. Nous retenons le modèle qui présente le score F1 le plus élevé parmi toutes les configurations testées.

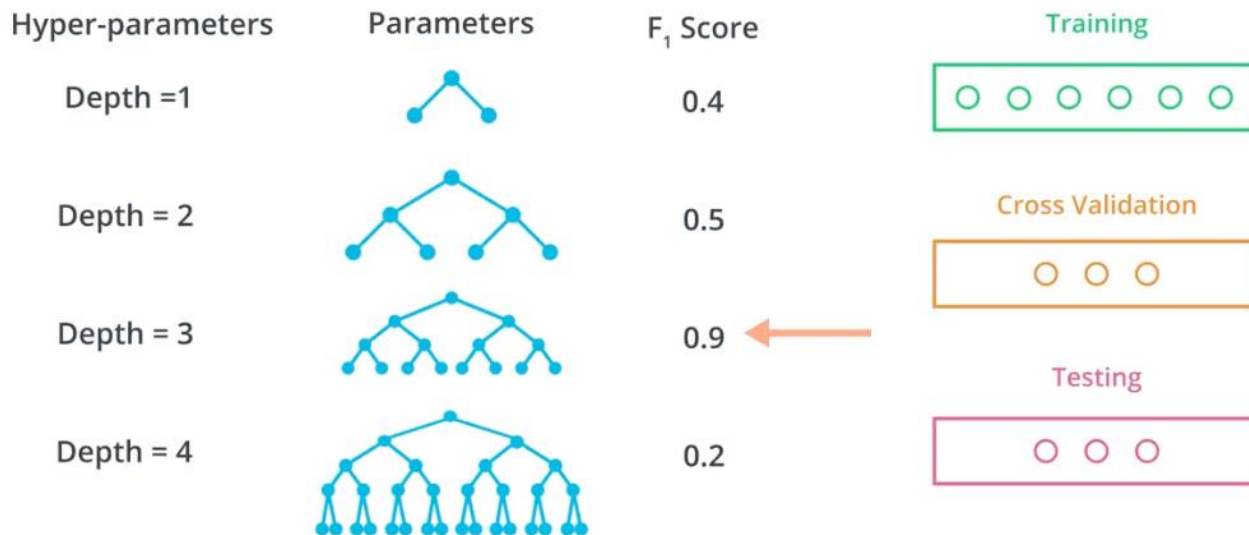
Il est important de distinguer deux aspects du modèle :

- **Les paramètres du modèle** : Il s'agit des coefficients de régression, qui sont ajustés pendant la phase d'apprentissage.
- **Les hyperparamètres du modèle** : Dans ce cas, le degré polynomial du modèle est considéré comme un hyperparamètre. Contrairement aux paramètres, les hyperparamètres sont définis avant l'entraînement et influencent la structure ou le comportement global du modèle.



# Grid Search

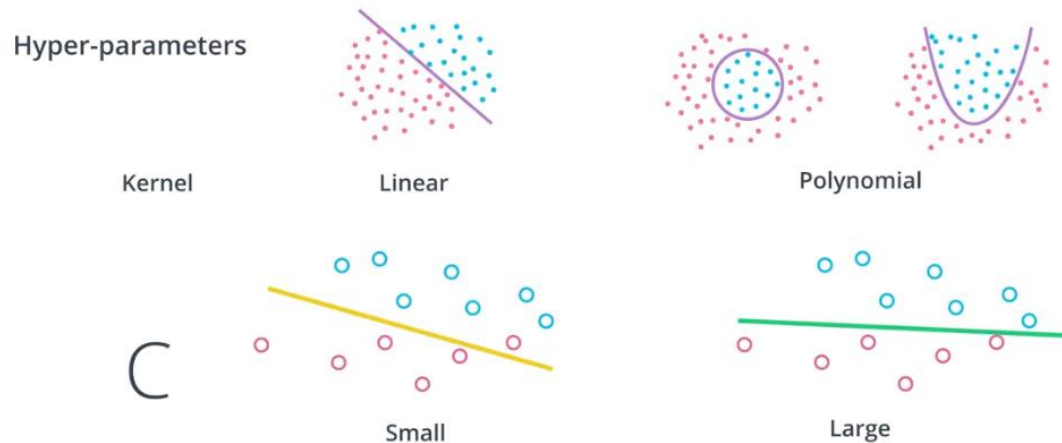
## ◦ TRAINING A DECISION TREE



- De la même manière, nous nous appuyons sur le score  $F_1$  obtenu lors de la validation croisée pour choisir la valeur de l'hyperparamètre qui convient le mieux à notre problème.
- La question qui se pose maintenant est : que se passe-t-il si nous devons ajuster plusieurs hyperparamètres ?

# Grid Search







## ◦ TRAINING A SUPPORT VECTOR MACHINE



Comment identifier la meilleure combinaison entre le noyau (kernel) et le paramètre C ?

# Grid Search

## ◦ GRID SEARCH CROSS VALIDATION

Kernel C	Linear	Polynomial
0.1	 F1 SCORE = 0.5	 F1 SCORE = 0.2
1	 F1 SCORE = 0.8	 F1 SCORE = 0.4
10	 F1 SCORE = 0.6	 F1 SCORE = 0.6

Training



Cross Validation



Testing



# Grid Search sur sklearn

```
from sklearn.model_selection import GridSearchCV
```

```
parameters = {'kernel':['poly', 'rbf'],'C':[0.1, 1, 10]}
```

```
from sklearn.metrics import make_scorer  
from sklearn.metrics import f1_score  
scorer = make_scorer(f1_score)
```

```
# Create the object.  
grid_obj = GridSearchCV(clf, parameters, scoring=scorer)  
# Fit the data  
grid_fit = grid_obj.fit(X, y)
```

```
best_clf = grid_fit.best_estimator_
```

Grid search Lab: <https://github.com/elhidali/EPISEN-2024>