Scikit-Learn API for classification and regression

- API: application programming interface

from some_package **import** **ClassifierA**, **RegressorB**

modelA=**ClassifierA**(set some **hyper**-parameters)

modelA.**fit**(X_train, Y_train)
find the optimal (internal) parameters of the model, using the training set

Y_pred = modelA.**predict**(X_new) # test on a new (validation/test) set
modelA.score(X, Y) # different meaning for classifiers and regressors
We may use other evaluation metrics (confusing matrix, MSE, MAE, etc)

Model Selection:
Find the Best Model via Hyper-parameter Optimization

- The performance of a model is determined by its hyper-parameter(s)

```
class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform',
algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)
```

```
class sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best',
max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,
max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0,
class_weight=None, ccp_alpha=0.0, monotonic_cst=None)
```

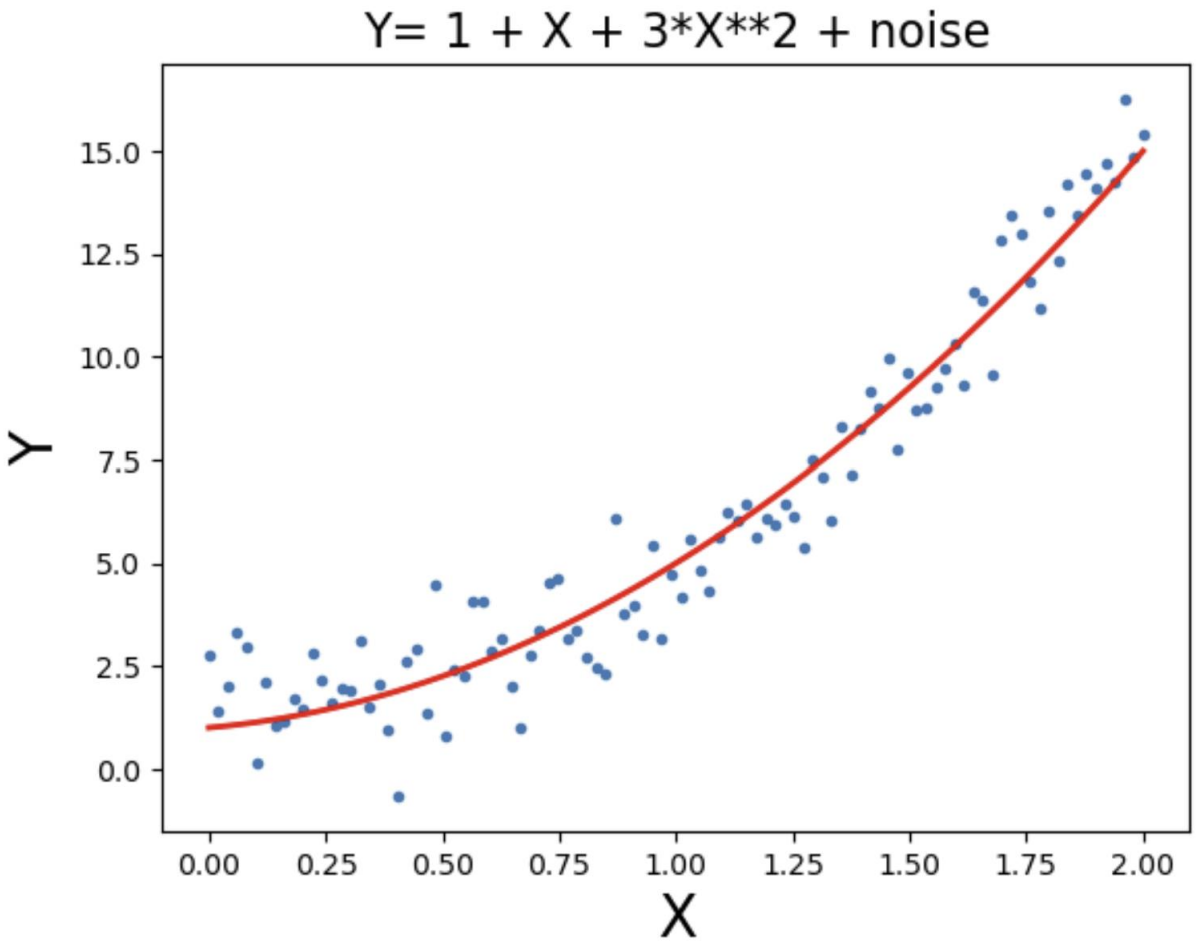
[source]

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=100, *, criterion='gini',
max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,
max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True,
oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False,
class_weight=None, ccp_alpha=0.0, max_samples=None, monotonic_cst=None)
```

[source]

Model Selection:
Find the Best Model via Hyper-parameter Optimization

- The performance of a model is determined by its hyper-parameter(s)



100 data points (blue)

the red curve (the best/true model):
a polynomial model with degree 2

Model Selection:
Find the Best Model via Hyper-parameter Optimization

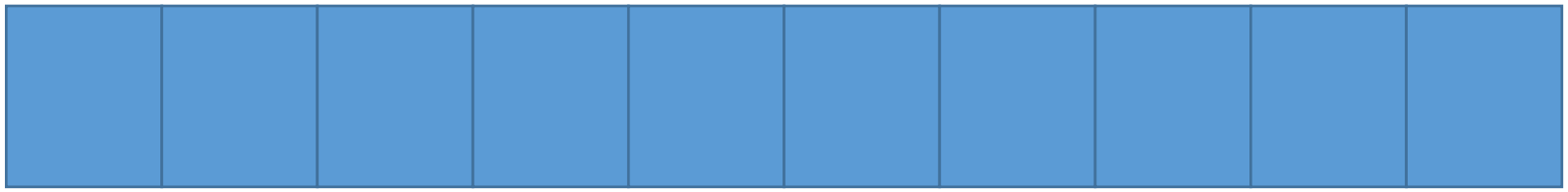
- We choose a polynomial model of degree K

$$\hat{y} = w_{(1)}x + w_{(2)}x^2 \ldots + w_{(K)}x^K + b$$

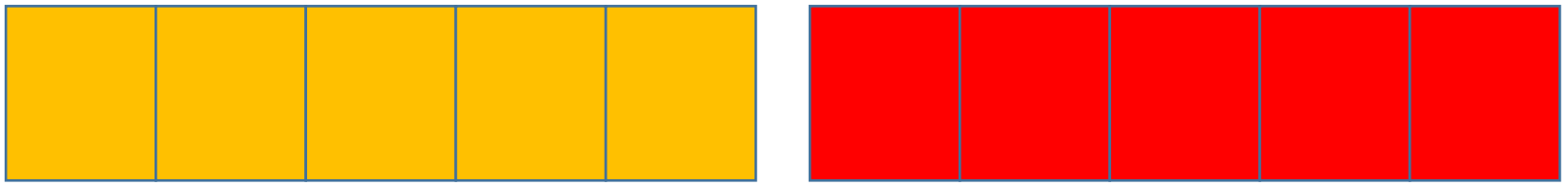
- This polynomial model has two sets of parameters:
  - Hyper-parameter: K
  - Learnable parameters: $\{w_{(1)}, w_{(2)}, \ldots, w_{(K)}, b\}$
- The optimal values of the learnable parameters can be obtained by using the gradient descent method with the MSE loss.
- The optimal value of the hyper-parameter can be obtained via cross-validation.

Model Selection (Hyper-parameter Optimization) using Cross Validation

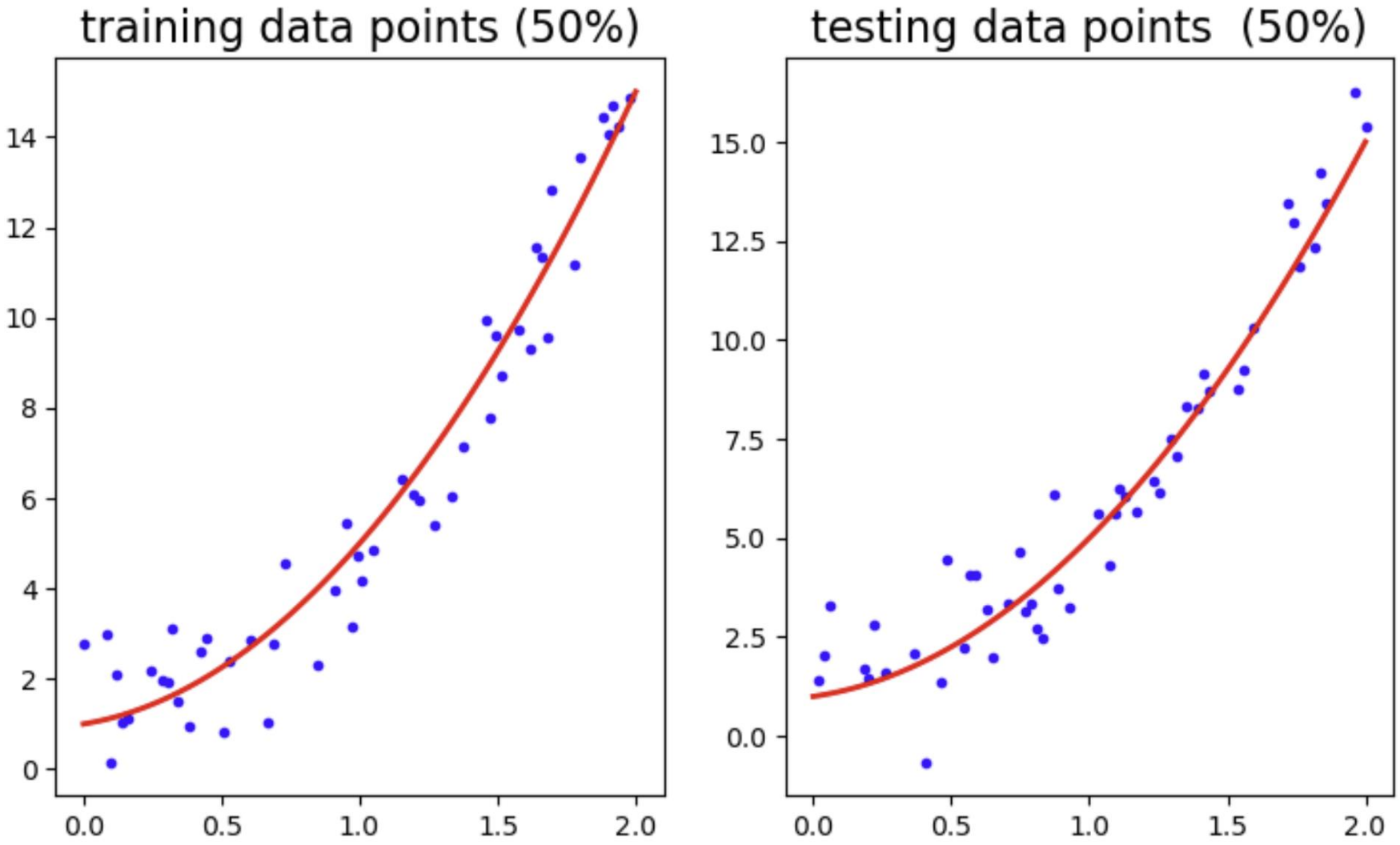
- We have a dataset



- Divide the whole dataset into a training dataset and a testing dataset



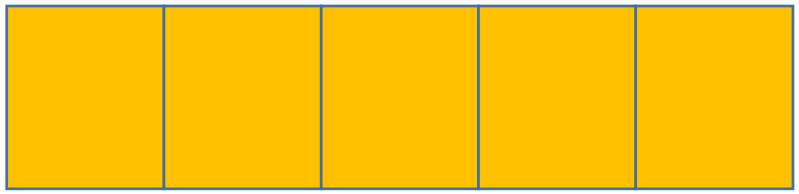
- Perform model selection on the training dataset
- Do not use the testing dataset for model selection
  the testing dataset is used only for model evaluation (the final judgment)
  It should not be used to find the best model (otherwise, it is cheating)



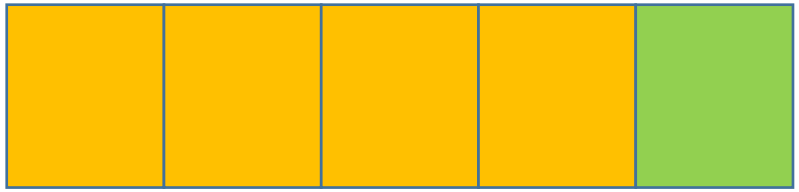
We will fit a polynomial model with degree = K to the training data.
We do not know the optimal value of K: the hyper-parameter.

Choose a possible value of the hyperparameter (e.g., degree K), then perform Five-fold cross validation

- We have a training dataset

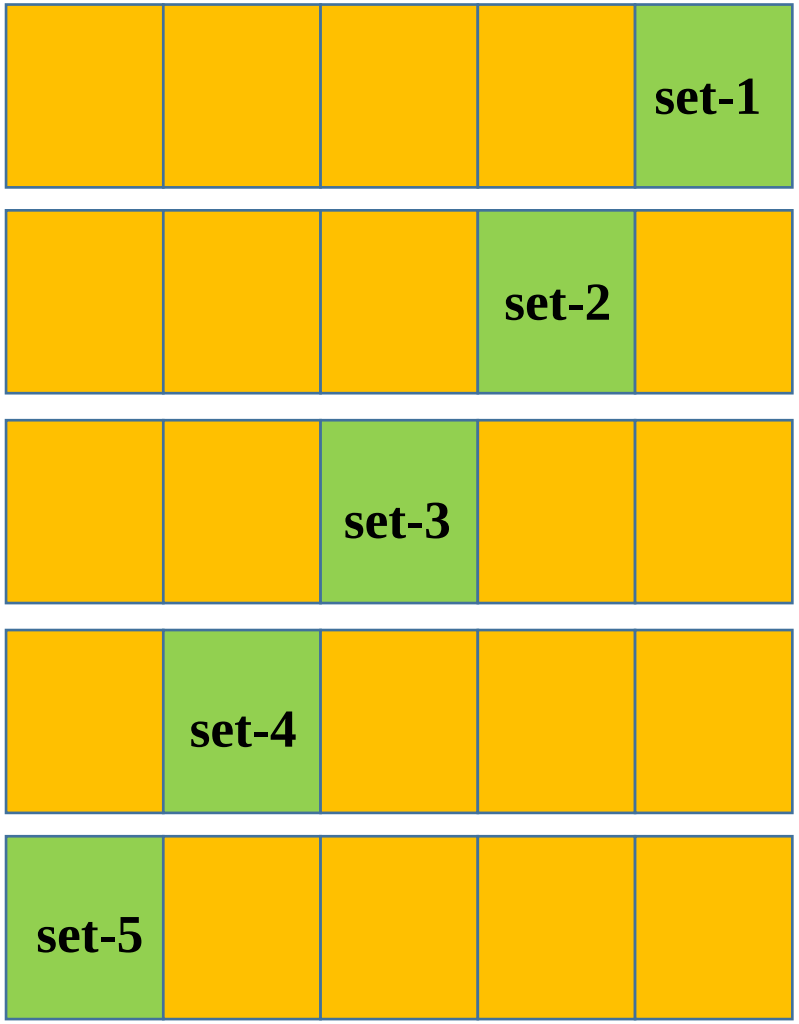


- Divide it into a 'pure' training dataset (80%) and a validation dataset (20%)



- Fit the model to the 'pure' training dataset (yellow)
- Measure performance of the fitted model on the validation set (green)

Choose a possible value of the hyperparameter (e.g., degree K), then perform Five-fold cross validation



MSE1: MSE on the validation set-1

MSE2 : MSE on the validation set-2

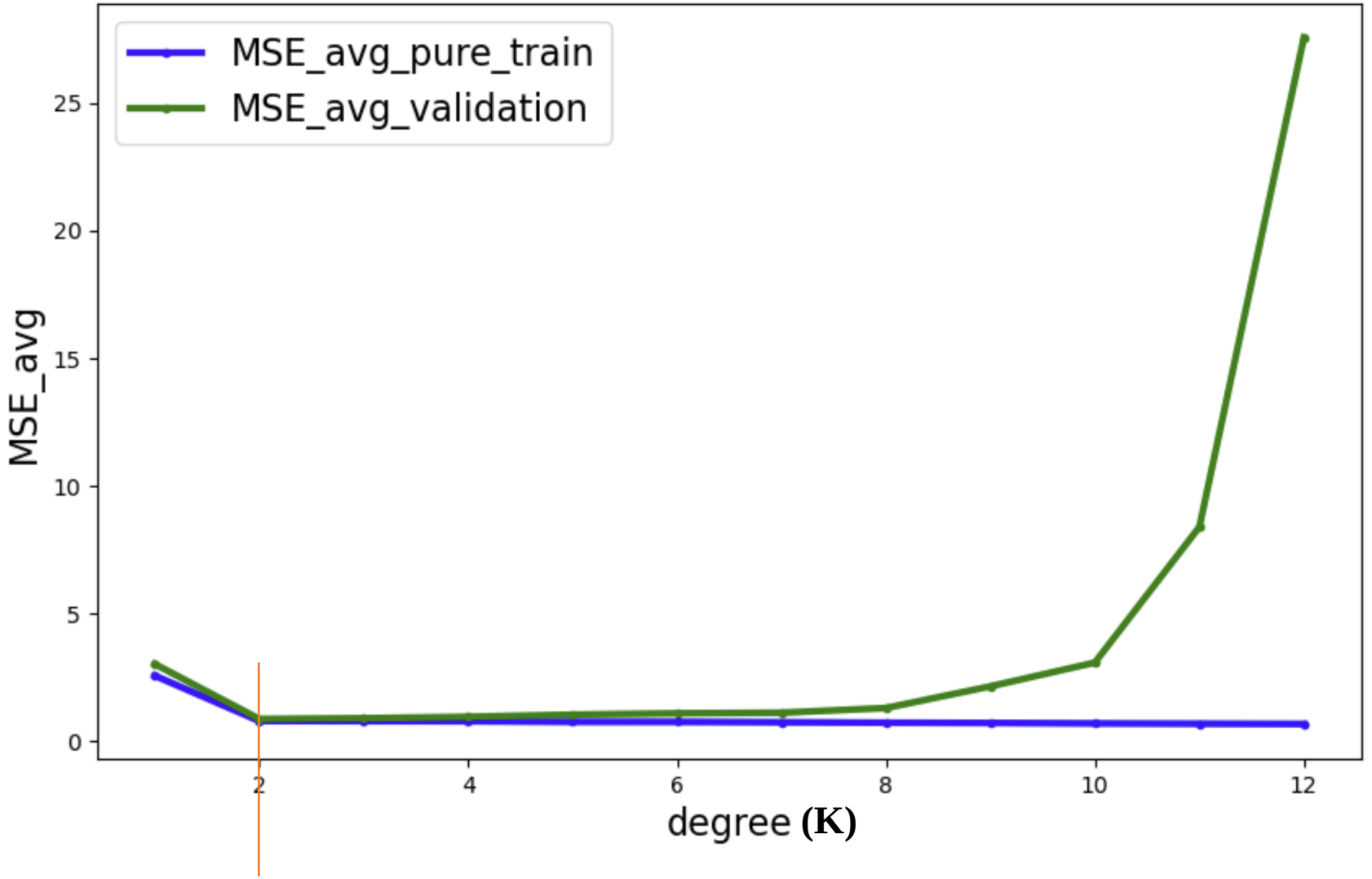
MSE3 : MSE on the validation set-3

MSE4 : MSE on the validation set-4

MSE5 : MSE on the validation set-5

$$MSE\_avg\ (K) = (MSE1+MSE2+MSE3+MSE4+MSE5)/5$$

cross_validation_for_model_selection.ipynb



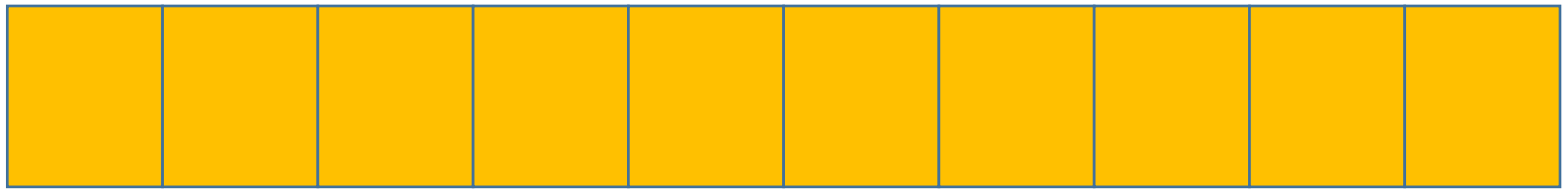
the best value of K is 2, leading to the minimum MSE on the validation sets

Five-fold cross validation to find the optimal hyper-parameter K

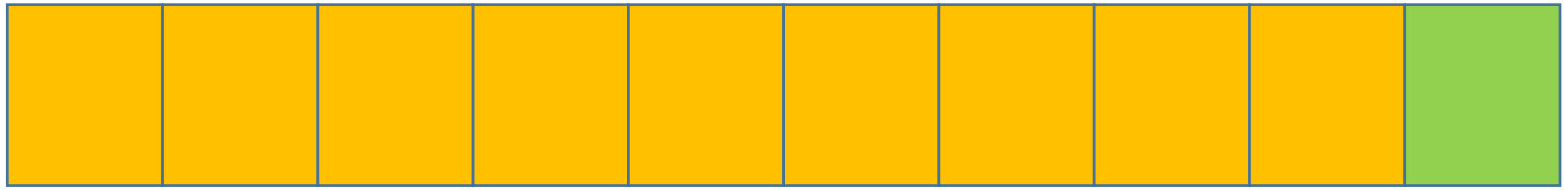
- We want to fit a regression model to the training data
- Split the training data into pure training sets and validation sets
- The validation sets are disjoint (they do not share data points)
- Given a value of *K*, we fit the model to the pure training sets and measure performance on the validation sets: MSE1, MSE2, MSE3, MSE4, MSE5
- MSE_avg(*K*) = (MSE1 + MSE2 + MSE3 + MSE4 + MSE5)/5
- repeat the process with different a different value of *K*
- The optimal *K* is the one associated with the minimum MSE_avg

Ten-fold cross validation

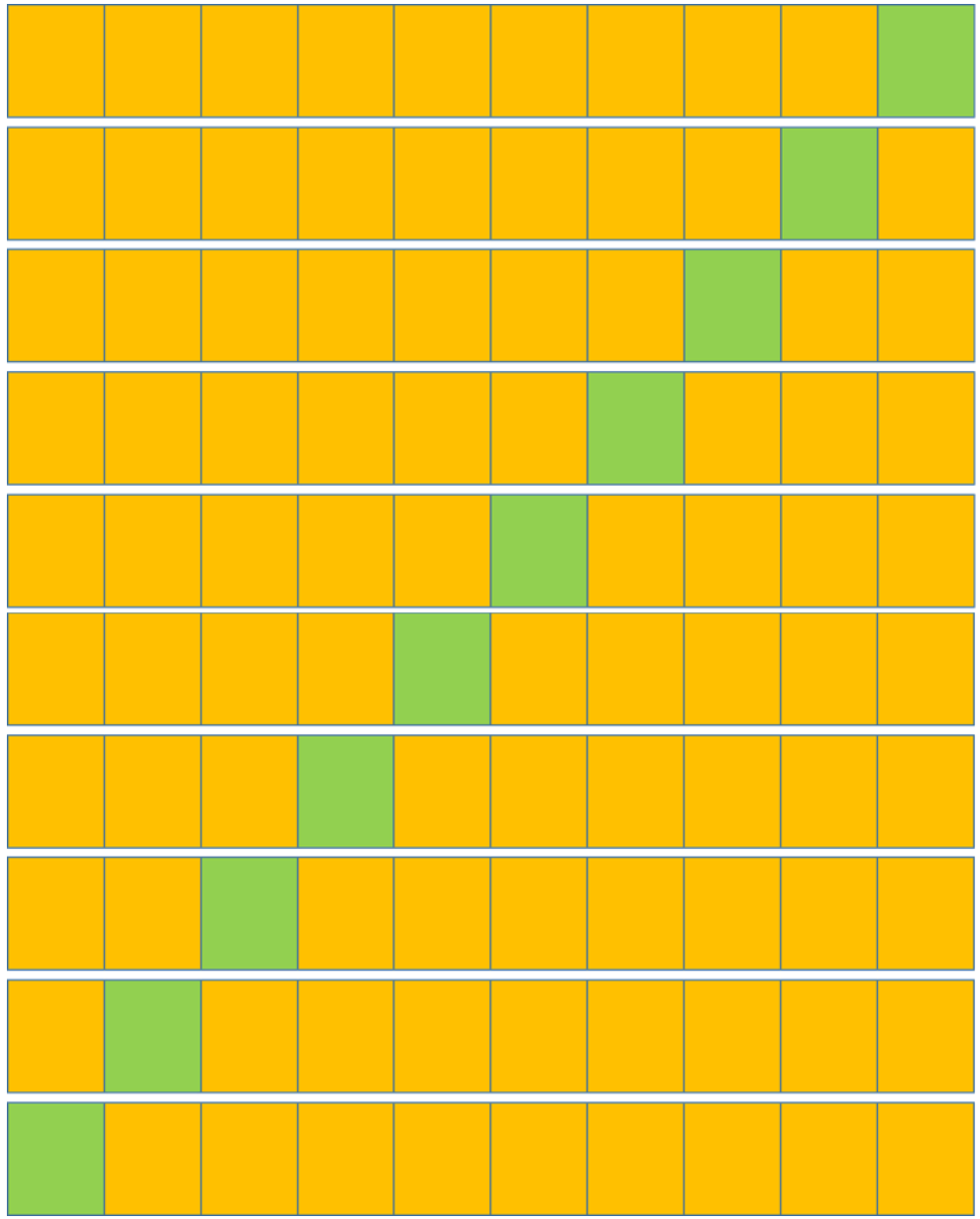
- We have a training dataset



- Divide it into a pure training dataset (90%) and a validation dataset (10%)



- Fit the model to the pure training dataset (yellow)
- Measure performance of the fitted model on the validation set (green)



MSE1: MSE on the validation set-1

MSE2 : MSE on the validation set-2

MSE3 : MSE on the validation set-3

MSE4 : MSE on the validation set-4

MSE5 : MSE on the validation set-5

MSE6: MSE on the validation set-6

MSE7 : MSE on the validation set-7

MSE8 : MSE on the validation set-8

MSE9 : MSE on the validation set-9

MSE10 : MSE on the validation set-10

$$MSE\_avg = (MSE1+MSE2+MSE3+MSE4+MSE5+MSE6+MSE7+MSE8+MSE9+MSE10)/10$$

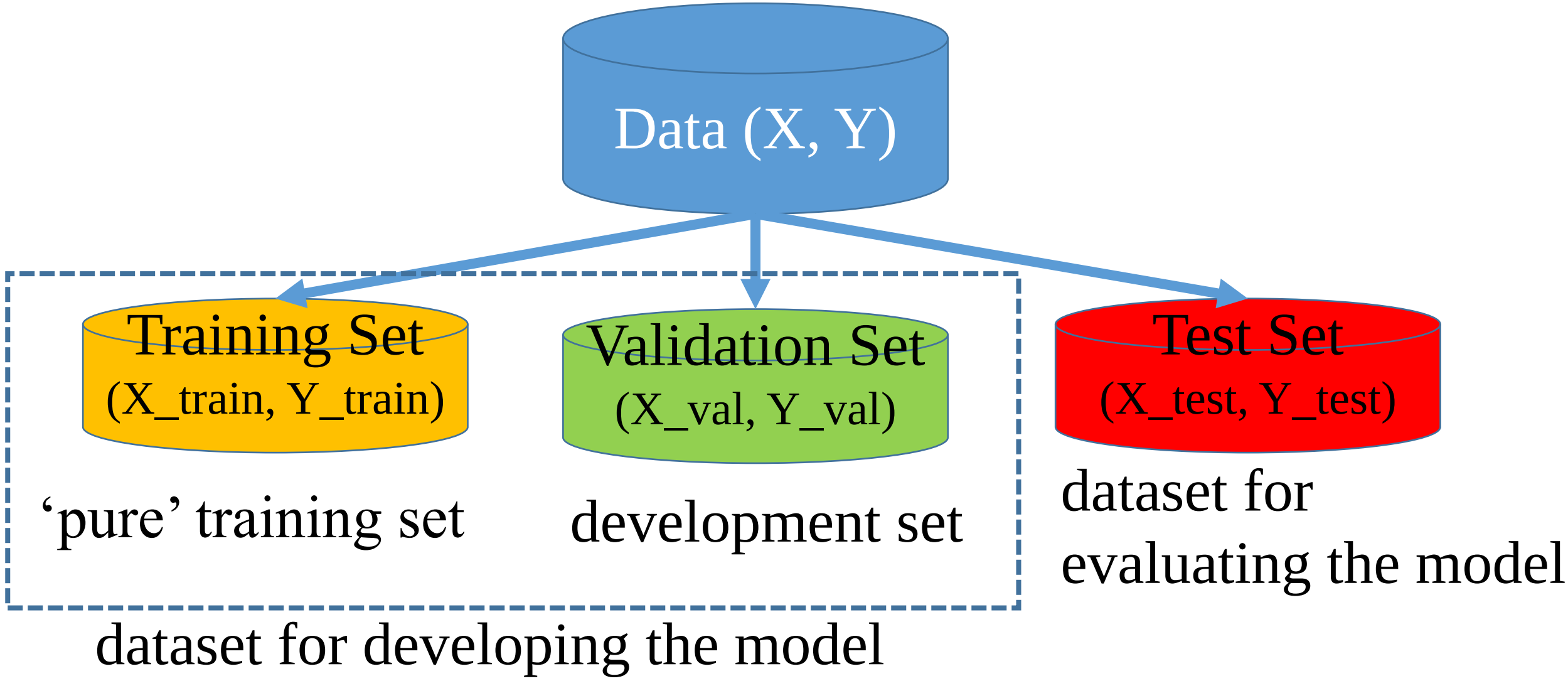
M-fold Cross Validation
Leave one out (LOO) Cross Validation

- In M-fold Cross Validation, the training data is divided into M-groups and the validation set only contains one group of data points.

- LOO is an extreme case of M-fold cross validation, in which each validation set contains only one data point.

- Usually, people choose 10-fold or 5-fold

For a large dataset:
Cross-validation has high computational cost
Training-Validation-Test is enough



The Train-Validation-Test Approach

We choose a KNN model for a classification task.
The hyper-parameter is K: the number of neighbors

Hyper-parameter K	Training Accuracy	Validation Accuracy	Testing Accuracy
1	0.80	0.84	0.82
2	0.81	0.83	**0.90**
3	0.88	**0.85**	0.87
4	0.90	0.80	0.85
5	**1.00** (100%)	0.81	0.80

The optimal value of the parameter K is 3:
it is optimal on the validation set