

# EE2211 Tutorial 10

Dr Feng LIN



# Q1

We have two classifiers showing the same accuracy with the same cross-validation. The more complex model (such as a 9th-order polynomial model) is preferred over the simpler one (such as a 2nd-order polynomial model).

- a) True
- b) False

# Q1

We have two classifiers showing the same accuracy with the same cross-validation. The more complex model (such as a 9th-order polynomial model) is preferred over the simpler one (such as a 2nd-order polynomial model).

- a) True
- b) False



## Q2

We have 3 parameter candidates for a classification model, and we would like to choose the optimal one for deployment. As such, we run 5-fold cross-validation.

Once we have completed the 5-fold cross-validation, in total, we have trained \_\_\_\_\_ classifiers. Note that, we treat models with different parameters as different classifiers.

A) 10

B) 20

C) 25

D) 15

## Q2

We have 3 parameter candidates for a classification model, and we would like to choose the optimal one for deployment. As such, we run 5-fold cross-validation.

Once we have completed the 5-fold cross-validation, in total, we have trained \_\_\_\_\_ classifiers. Note that, we treat models with different parameters as different classifiers.

A) 10

B) 20

C) 25

D) 15

In each fold, we train 3 classifiers, so 5 folds give 15 classifiers.

## Q3

Suppose the **binary classification** problem, which you are dealing with, has highly imbalanced classes. The majority class has **99 hundred samples** and the minority class has **1 hundred samples**. Which of the following metric(s) would you choose for assessing the classification performance?

- a) Classification Accuracy
- b) Cost sensitive accuracy
- c) Precision and recall
- d) None of these

# Q3

Suppose the binary classification problem, which you are dealing with, has highly imbalanced classes. The majority class has 99 hundred samples and the minority class has 1 hundred samples. Which of the following metric(s) would you choose for assessing the classification performance?

- a) Classification Accuracy
- b) Cost sensitive accuracy
- c) Precision and recall
- d) None of these

	$\hat{P}$ (predicted)	$\hat{N}$ (predicted)	
$P$ (actual)	TP	FN	Recall $TP/(TP+FN)$
$N$ (actual)	FP	TN	
	Precision $TP/(TP+FP)$	Accuracy $(TP+TN)/(TP+TN+FP+FN)$	

Page 28, Lec 10

The goal is to highlight the problems of the results!

In this case, we shall

- 1) Use cost matrix, assign different costs for each entry
- 2) Use Precision and Recall! Precision = 0.5 and Recall = 0.5

# Q4

Given below is a scenario for Training error rate  $Tr$ , and Validation error rate  $Va$  for a machine learning algorithm. You want to choose a hyperparameter ( $P$ ) based on  $Tr$  and  $Va$ . Which value of  $P$  will you choose based on the above table?

- a) 10
- b) 9
- c) 8
- d) 7
- e) 6

P	Tr	Va
10	0.10	0.25
9	0.30	0.35
8	0.22	0.15
7	0.15	0.25
6	0.18	0.15



# Q4

Given below is a scenario for Training error rate  $Tr$ , and Validation error rate  $Va$  for a machine learning algorithm. You want to choose a hyperparameter ( $P$ ) based on  $Tr$  and  $Va$ . Which value of  $P$  will you choose based on the above table?

- a) 10
- b) 9
- c) 8
- d) 7
- e) 6

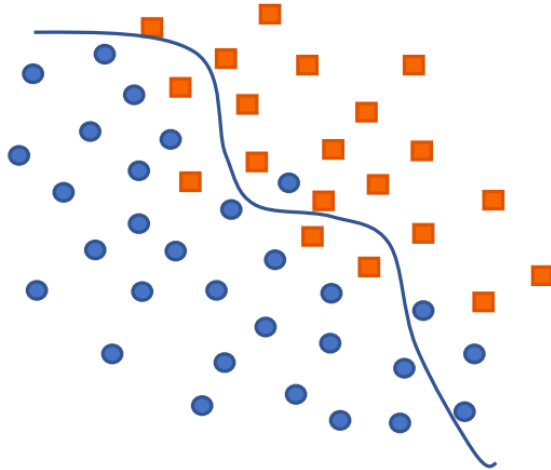
P	Tr	Va
10	0.10	0.25
9	0.30	0.35
8	0.22	0.15
7	0.15	0.25
6	0.18	0.15

# Q5

(Binary and Multicategory Confusion Matrices)

Tabulate the confusion matrices for the following classification problems.

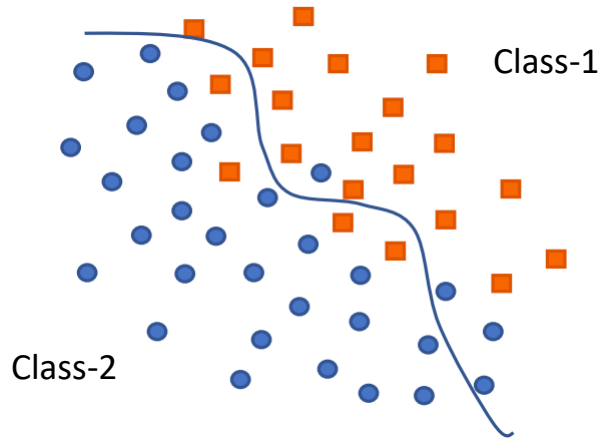
- a) Binary problem (the class-1 and class-2 data points are respectively indicated by squares and circles)



# Q5

Tabulate the confusion matrices for the following classification problems.

- a) Binary problem (the class-1 and class-2 data points are respectively indicated by squares and circles)

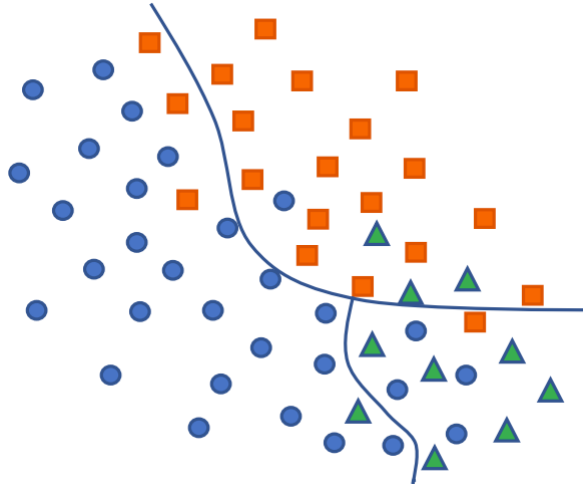


	$P_{\hat{1}}$	$P_{\hat{2}}$
$P_1$	16	4
$P_2$	4	26

# Q5

Tabulate the confusion matrices for the following classification problems.

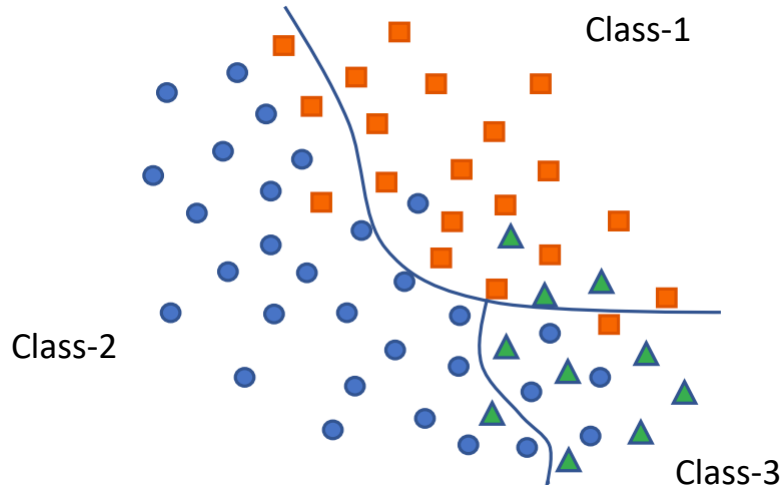
- b) Three-category problem (the **class-1**, **class-2** and **class-3** data points are respectively indicated by **squares**, **circles** and **triangles**)



# Q5

Tabulate the confusion matrices for the following classification problems.

- b) Three-category problem (the class-1, class-2 and class-3 data points are respectively indicated by squares, circles and triangles)



	$P_{\hat{1}}$	$P_{\hat{2}}$	$P_{\hat{3}}$
$P_1$	16	3	1
$P_2$	1	25	4
$P_3$	3	1	6

# Q6 (python)





## (5-fold Cross-Validation)

Get the data set “from sklearn.datasets import load\_iris”. Perform a 5-fold Cross-validation to observe the best polynomial order (among orders 1 to 10 and without regularization) for validation prediction. Note that, you will have to partition the whole dataset for training/validation/test parts, where the size of validation set is the same as that of test. Provide a plot of the average 5-fold training and validation error rates over the polynomial orders. The randomly partitioned data sets of the 5-fold shall be maintained for reuse in evaluation of future algorithms

# Q6

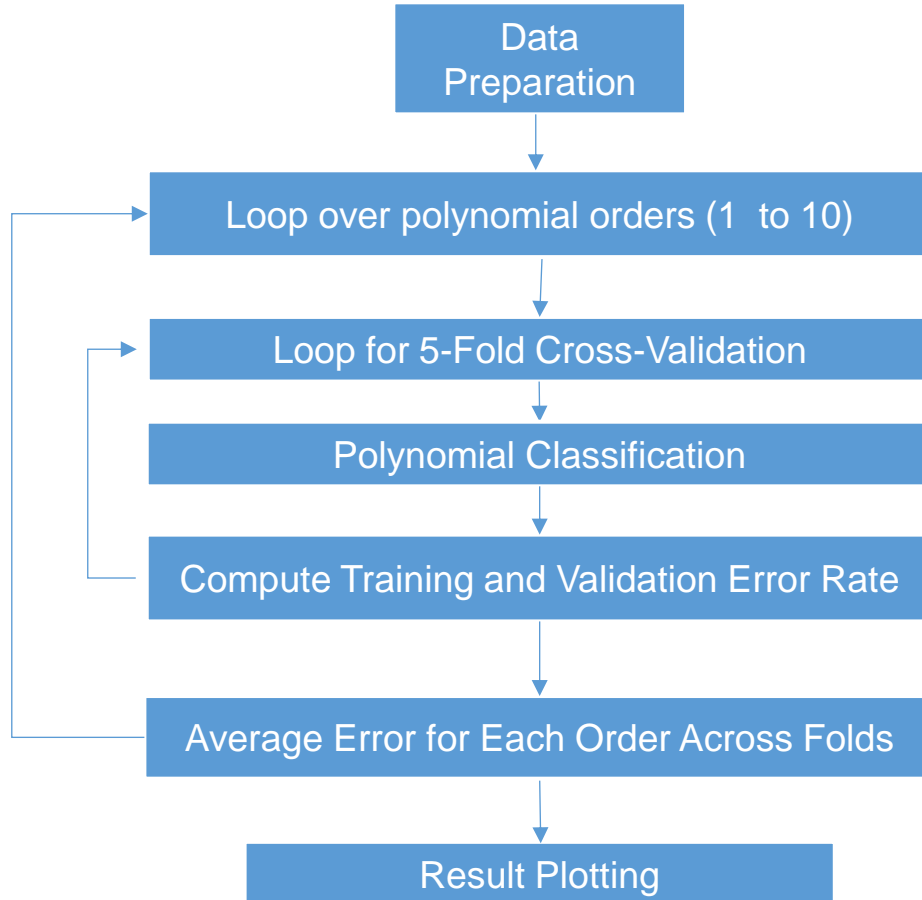
## About this file

The dataset is a CSV file which contains a set of 150 records under 5 attributes - Petal Length, Petal Width, Sepal Length, Sepal width and Class(Species)

# sepal_length	# sepal_width	# petal_length	# petal_width	Δ species
 4.3 7.9	 2 4.4	 1 6.9	 0.1 2.5	<b>3</b> unique values
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
4.8	3	1.4	0.1	Iris-setosa

# Q6

## Block diagram



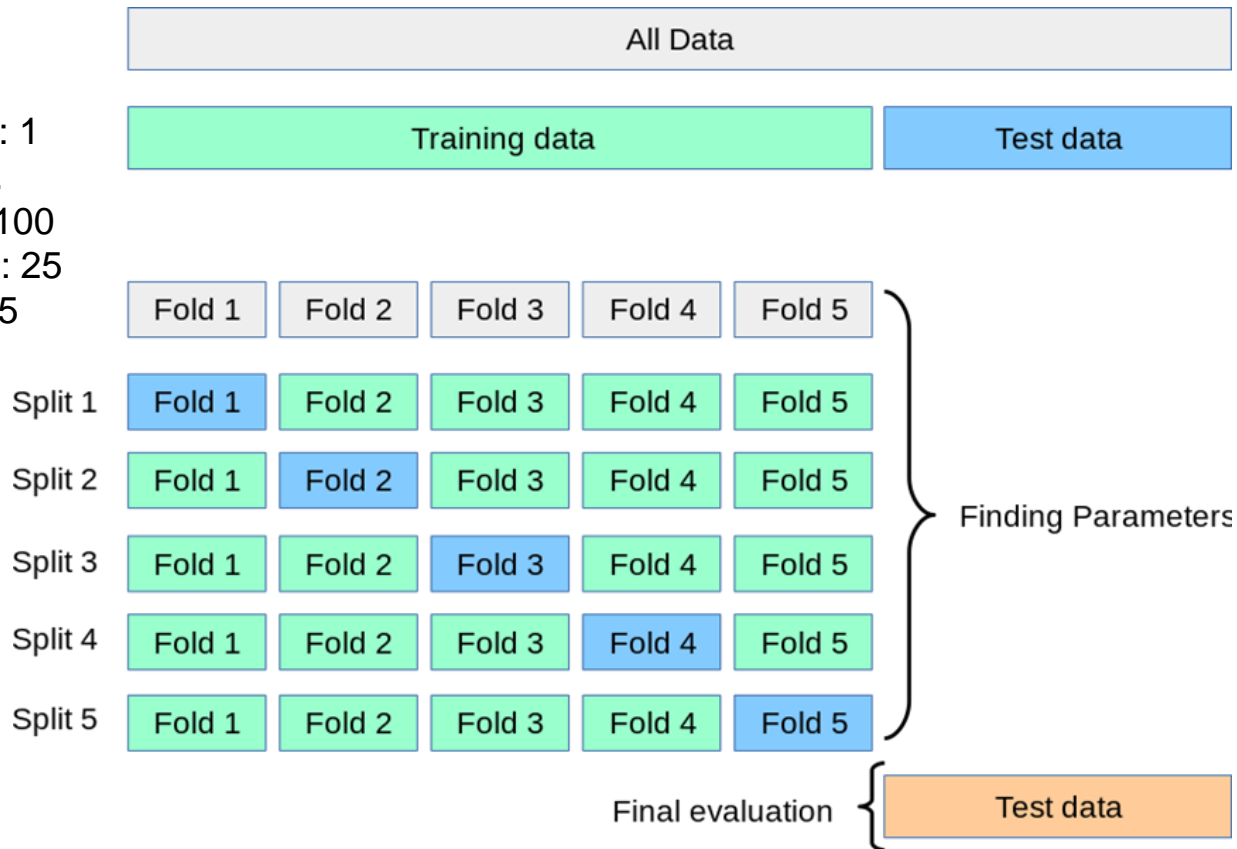
- One-Hot Encoding
- Data Splitting

- 5-Fold Splitting
- Feature Expansion
- Least-Squares Solution



# Q6

- Train : Validation : Test = 4 : 1 : 1
- In total, we have 150 samples.
- Number of Training Samples: 100
- Number of Validation Samples: 25
- Number of Testing Samples: 25



[https://scikit-learn.org/stable/modules/cross\\_validation.html](https://scikit-learn.org/stable/modules/cross_validation.html)

# Q6

```
##--- load data from scikit ---##
import numpy as np
import pandas as pd
print("pandas version: {}".format(pd.__version__))
import sklearn
print("scikit-learn version: {}".format(sklearn.__version__))
from sklearn.datasets import load_iris
iris_dataset = load_iris()
X = np.array(iris_dataset['data'])
y = np.array(iris_dataset['target'])
## one-hot encoding
Y = list()
for i in y:
    letter = [0, 0, 0]
    letter[i] = 1
    Y.append(letter)
Y = np.array(Y)
test_idx = np.random.RandomState(seed=2).permutation(Y.shape[0])
X_test = X[test_idx[:25]]
Y_test = Y[test_idx[:25]]
X = X[test_idx[25:]]
Y = Y[test_idx[25:]]
```

- Loads the Iris dataset, a classic dataset with three classes of flowers (Setosa, Versicolor, and Virginica).
- X: A NumPy array of shape (150, 4), containing four features for each sample.
- y: A NumPy array of shape (150,), representing class labels (0, 1, or 2).
- One-Hot Encoding: The target variable, y, is converted to a one-hot encoding format (Y) for each class (e.g., [1,0,0] for class 0, [0,1,0] for class 1, etc.).
- Y is now a NumPy array of shape (150, 3)
- Uses a fixed seed (2) to create a reproducible random permutation of indices for splitting data.
- Selects the first 25 samples for X\_test and Y\_test, leaving the remaining 125 samples in X and Y for training and validation.

# Q6

```
from sklearn.preprocessing import PolynomialFeatures
error_rate_train_array = []
error_rate_val_array = []
##--- Loop for Polynomial orders 1 to 10 ---##
for order in range(1,11):
    error_rate_train_array_fold = []
    error_rate_val_array_fold = []
    # Random permutation of data
    Idx = np.random.RandomState(seed=8).permutation(Y.shape[0])
    # Loop 5 times for 5-fold
    for k in range(0,5):
        ##--- Prepare training, validation, and test data for the 5-fold ---#
        # Prepare indexing for each fold
        X_val = X[Idx[k*25:(k+1)*25]]
        Y_val = Y[Idx[k*25:(k+1)*25]]
        Idxtrn = np.setdiff1d(Idx, Idx[k*25:(k+1)*25])
        X_train = X[Idxtrn]
        Y_train = Y[Idxtrn]
```

The code performs polynomial classification by expanding features to polynomial forms of varying degrees (1 to 10). For each polynomial order, it uses 5-fold cross-validation:

Creates a new random permutation of indices to assign data for 5-fold cross-validation.

Divides data into training and validation sets for each fold:

- X\_val and Y\_val: Select the next 25 samples for validation.
- X\_train and Y\_train: Exclude the validation indices, using the remaining samples for training.

# Q6

```
##--- Polynomial Classification ---##  
poly = PolynomialFeatures(order)  
P = poly.fit_transform(X_train)  
Pval = poly.fit_transform(X_val)  
if P.shape[0] > P.shape[1]: # over-/under-determined cases
```

- Creates polynomial features of the specified order for X\_train and X\_val.
- P and Pval are the transformed features for training and validation, respectively.

```
    reg_L = 0.00*np.identity(P.shape[1])  
    inv_PTP = np.linalg.inv(P.transpose().dot(P)+reg_L)  
    pinv_L = inv_PTP.dot(P.transpose())  
    wp = pinv_L.dot(Y_train)
```

```
else:  
    reg_R = 0.00*np.identity(P.shape[0])  
    inv_PPT = np.linalg.inv(P.dot(P.transpose())+reg_R)  
    pinv_R = P.transpose().dot(inv_PPT)  
    wp = pinv_R.dot(Y_train)
```

- Checks if the system is over- or under-determined (more rows than columns).
- Adds a small regularization term (reg\_L or reg\_R) for numerical stability in the pseudoinverse calculation.
- Least-Squares Solution: It calculates weights wp to fit the model by solving a system of equations based on whether the system is overdetermined or underdetermined.

$$\hat{\mathbf{w}} = (\mathbf{P}^T \mathbf{P} + \lambda \mathbf{I})^{-1} \mathbf{P}^T \mathbf{y}$$

$$\hat{\mathbf{w}} = \mathbf{P}^T (\mathbf{P} \mathbf{P}^T + \lambda \mathbf{I})^{-1} \mathbf{y}$$

# Q6

```
##--- trained output ---##
y_est_p = P.dot(wp);
y_cls_p = [[1 if y == max(x) else 0 for y in x] for x in y_est_p ]
m1tr = np.matrix(Y_train)
m2tr = np.matrix(y_cls_p)
# training classification error count and rate computation
difference = np.abs(m1tr - m2tr)
error_train = np.where(difference.any(axis=1))[0]
error_rate_train = len(error_train)/len(difference)
error_rate_train_array_fold += [error_rate_train]
##--- validation output ---##
yval_est_p = Pval.dot(wp);
yval_cls_p = [[1 if y == max(x) else 0 for y in x] for x in yval_est_p ]
m1 = np.matrix(Y_val)
m2 = np.matrix(yval_cls_p)
# validation classification error count and rate computation
difference = np.abs(m1 - m2)
error_val = np.where(difference.any(axis=1))[0]
error_rate_val = len(error_val)/len(difference)
error_rate_val_array_fold += [error_rate_val]

# store results for each polynomial order
error_rate_train_array += [np.mean(error_rate_train_array_fold)]
error_rate_val_array += [np.mean(error_rate_val_array_fold)]
```

$$\hat{f}_w(P(X_{new})) = P_{new}\hat{W}$$

- `y_est_p`: Predicts continuous outputs by applying `wp` to the training data.
  - `y_cls_p`: Converts `y_est_p` to a binary one-hot format for classification (1 for the maximum value, 0 elsewhere).
  - `m1tr` and `m2tr` represent the true and predicted one-hot encoded labels as matrices for easy comparison.
  - Computes the training error rate by comparing `y_cls_p` to `Y_train`, identifying misclassified samples in each fold.
- 
- Applies the same classification process to the validation set.
  - Appends the validation error rate for this fold to `error_rate_val_array_fold`.
- 
- Stores the average training and validation error rates across the 5 folds for each polynomial order.

# Q6





**THANK YOU**