EE2211 Tutorial 10

Question 1:
We have two classifiers showing the same accuracy with the same cross-validation. The more complex model (such as a 9^{th}-order polynomial model) is preferred over the simpler one (such as a 2^{nd}-order polynomial model).
a) True
b) False

Answer: b).

Question 2:

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

Answer: D)

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

Question 3:
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? (Select all relevant metric(s) to get full credit)
a) Classification Accuracy
b) Cost sensitive accuracy
c) Precision and recall
d) None of these

Answer: (b, c)

Question 4:
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.

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

Which value of P will you choose based on the above table?

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

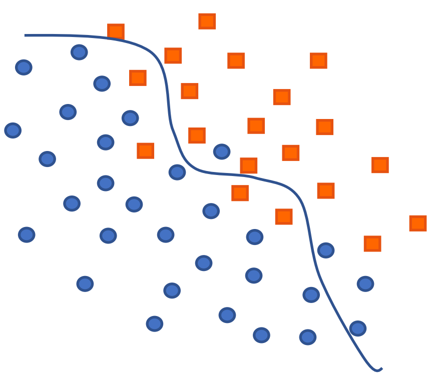
Answer: e).

(Binary and Multicategory Confusion Matrices)

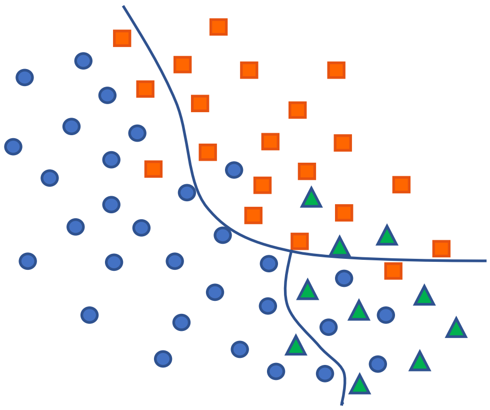
Question 5:

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)



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



Answer:

(a)

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

(b)

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

(5-fold Cross-validation)

Question 6:

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.

Answer:

```
##--- 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:]]

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]
        ##--- 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
            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)
        ##--- 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)]
##--- plotting ---##
import matplotlib.pyplot as plt
order=[x for x in range(1,11)]
plt.plot(order, error_rate_train_array, color='blue', marker='o', linewidth=3,
label='Training')
plt.plot(order, error_rate_val_array, color='orange', marker='x', linewidth=3,
label='Validation')
plt.xlabel('Order')
plt.ylabel('Error Rates')
plt.title('Training and Validation Error Rates')
plt.legend()
plt.show()
```


Training and Validation Error Rates
Training
Validation
Error Rates
Order
0.0
0.1
0.2
0.3
0.4
0.5
2
4
6
8
10