COMP3009/ COMP4139 Machine Learning

Lab 5 – Method Evaluation

Dr Xin Chen, Autumn Semester, 2024

1. Introduction

Once we build the ML models in labs 3 and 4, it is very important to evaluate the performance of these models. In this lab, we will introduce different evaluation metrics and typically used evaluation experimental designs. After finishing all the lab sessions 1-5, you should be able to complete assignment 1 now. Note that these lab sessions only help you to get started with Python and some basic ML packages, please take your time to learn more useful functions, and more importantly, know how to get help from the official documentation. You may also use this lab session to work on assignment 1 in your work.

2. Tutorial

2.1 K-fold cross-validation:

https://scikit-learn.org/stable/modules/cross_validation.html

- Previously we used 70% training and 30% testing split to evaluate our ML models. However, if we have a small dataset, we want to iterate the training and testing examples to generate a test result for every example. K-fold cross-validation is a widely used experimental design to achieve this. K could be set to 3, 5,…, 10, etc, depending on the size of the available data and computational resources. Now let's try 5-fold cross-validation using the SVM classifier as an example. Then print out the mean and standard deviation classification accuracy of the 5 folds. Try to evaluate the performance of the other models you built previously using k-fold cross validation (decision tree, MLP, logistic regression, etc.).

```
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC

n_folds = 5
clf_cv = SVC(C=1.0, kernel='rbf', degree=3, gamma='auto')
scores = cross_val_score(clf_cv, Xs, y, cv=5)

print(scores)
avg = (100 * np.mean(scores), 100 * np.std(scores)/np.sqrt(scores.shape[0]))
print("Average score and standard deviation: (%.2f +- %.3f)%%"  %avg)
```

- We can also use a cross-validation iterator to split the data. Then we have more flexibility to play with the model in each fold and use different metrics to evaluate the model.

```
import numpy as np
from sklearn.model_selection import KFold

kf = KFold(n_splits=5)
for train, test in kf.split(Xs, y):
    SVM_clf = SVC(C=1.0, kernel='rbf', degree=3, gamma='auto',
probability=True)
    SVM_clf.fit(Xs[train], y[train])
    classifier_score = SVM_clf.score(Xs[test], y[test])
    print('The classifier accuracy is {:03.2f}'.format(classifier_score))
```

2.2 Evaluation metrics

- We use a confusion matrix (TP, TN, FP, FN) to visualise the performance. Here we use 70% training and 30% testing to split the data as an example. Understand the meaning of each evaluation metric (e.g. precision, recall, etc.). You should be able to compute the confusion matrix using k-fold cross-validation. Understand the code below by referring to some online documentation.

```
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.svm import SVC

Xs_train, Xs_test, y_train, y_test = train_test_split(Xs, y, test_size=0.3,
random_state=1, stratify=y)
SVM_clf = SVC(C=1.0, kernel='rbf', degree=3, gamma='auto',
probability=True)
y_pred = SVM_clf.fit(Xs_train, y_train).predict(Xs_test)
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix,
fig, ax = plt.subplots(figsize=(3, 3))
ax.matshow(cm, cmap=plt.cm.Blues, alpha=0.3)

for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(x=j, y=i,
                s=cm[i, j],
                va='center', ha='center')

classes=["Benign","Malignant"]
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=45)
plt.yticks(tick_marks, classes)
plt.xlabel('Predicted Values', )
plt.ylabel('Actual Values');
print(classification_report(y_test, y_pred ))
```

- Plot the receiver operating characteristic curve (ROC). Can you plot the ROC curves of different models on the same figure and compare their performance?

```
from sklearn.metrics import roc_curve, auc

plt.figure(figsize=(10,8))
probas_ = SVM_clf.predict_proba(Xs_test)
fpr, tpr, thresholds = roc_curve(y_test, probas_[:, 1])
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, lw=1, label='ROC fold (area = %0.2f)' % (roc_auc))
plt.plot([0, 1], [0, 1], '--', color=(0.6, 0.6, 0.6), label='Random')
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate | 1 - specificity (1 - Benign recall)')
plt.ylabel('True Positive Rate | Sensitivity (Malignant recall)')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
```

3. Additional Questions

- Now you can complete assignment 1, which asks you to compare the four specified methods using k-fold cross-validation.

- Assignment 1 also asks you to implement the four methods for a regression task.