COMP3009/ COMP4139 Machine Learning

Lab 3 – Machine Learning Models (1)

Dr Xin Chen, Autumn Semester, 2024

1. Introduction

This lab session continues the work of lab 2. If you haven't done lab 2, don't start this one. In lab 2, we have loaded the WDBC data and performed feature analysis. We will continue the implementation to build and compare some machine learning models to achieve prediction (i.e. breast cancer diagnosis in this case).

2. Tutorial

2.1 Construct training and testing datasets

- **Transform the class labels from their original string representation (M and B) into integers 1: M; 0: B. You may have done this in lab 2.**

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
all_df['Diagnosis'] = le.fit_transform(all_df['Diagnosis'])
all_df.head()

# assign numerical label to y
y = all_df['Diagnosis']
```

- **Divide data into training (70%) and testing sets (30%). Pay attention that we are using the normalised data value Xs rather than X. You may try using X.**

```
Xs_train, Xs_test, y_train, y_test = train_test_split(Xs, y, test_size=0.3,
random_state=1, stratify=y)
```

2.2 Logistic Regression

- **It is simple to use scikit-learn built-in function, but you need to understand the working principle of logistic regression (check lecture notes)**

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
log_reg.fit(Xs_train, y_train)
```

- **Classify the test dataset and output the accuracy. Any issues to use classification accuracy as the only evaluation metric?**

```
classifier_score = log_reg.score(Xs_test, y_test)
print('The classifier accuracy score is {:03.2f}'.format(classifier_score))
```

2.3 Gaussian Naive Bayes

- **You need to understand the working principle of Naïve Bayes (check lecture notes)**

```
from sklearn.naive_bayes import GaussianNB

gnb_clf = GaussianNB()
gnb_clf.fit(Xs_train, y_train)

classifier_score = gnb_clf.score(Xs_test, y_test)
print('The classifier accuracy score is {:03.2f}'.format(classifier_score))
```

2.4 K-Nearest Neighbour

- **You may try different number of neighbours and observe the result. Could you implement KNN without Scikit-learn library?**

```
from sklearn.neighbors import KNeighborsClassifier

knn_clf = KNeighborsClassifier(n_neighbors=3)
knn_clf.fit(Xs_train, y_train)

classifier_score = knn_clf.score(Xs_test, y_test)
print('The classifier accuracy score is {:03.2f}'.format(classifier_score))
```

2.5 Support Vector Machine (SVM)

- **Take your time to understand the function parameters and tune the parameters to see the impact to the result. What does parameter C control?**

```
from sklearn.svm import SVC

svm_clf = SVC(C=1.0, kernel='rbf', degree=3, gamma='auto',
probability=True)
svm_clf.fit(Xs_train, y_train)
```

2.6 Modelling based on a subset of good features

- **The above models were based on all 30 features. We may have “curse of dimensionality” problem. Now Let's try classification with some selected good features, not all the features. Re-run all the above models with the best few features using a feature selection method (lab 2) or the first few principle components of PCA (lab 2). Can we still achieve good classification accuracy?**

  Reference: https://scikit-learn.org/stable/modules/feature_selection.html

3 Additional questions to think about

- Check which method can handle multiple class classification and which ones can only work for a binary-class task. What are one-versus-all and one-versus-one strategies?

- Observe the results of each method if we don't normalise the feature value range.

- Instead of using the default parameter settings for the modelling methods (i.e. SVM, KNN, LG, etc), check the related documents and try different settings to see if the performance can be improved.