Q2

May 15, 2024

In HW3 you did an exercise on performing a classification on the HeartData.csv data file. We recently learned how to apply the LOOCV and K-Fold CV to regression problems. In this homework we would like to apply the LOOCV and K-Fold CV to the logistic regression, LDA and QDA models used in HW3. Using 10-fold CV and LOOCV fit the models and report the classification accuracy for the 3 models (logistic regression, LDA and QDA).For this question use num as the response variable and all the other variables as features.

[1]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from ISLP.models import sklearn_sm
from ISLP.models import (ModelSpec as MS,summarize)
from sklearn.discriminant_analysis import \
     (LinearDiscriminantAnalysis as LDA,
      QuadraticDiscriminantAnalysis as QDA)
from sklearn.model_selection import KFold
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error
```

[2]:
```
Data = pd.read_csv('HeartData-1.csv')
y = Data['num']
X = Data.drop(['num'], axis = 1) # without intercept
X_LR = MS(X).fit_transform(Data) # with intercept
```

0.1 = = = = = Running 10-Fold CV= = = = =

[3]:
```
def split_into_10_Fold_chunks(X, y, train_index, test_index):
    X_train = X.iloc[train_index]
    y_train = y.iloc[train_index]
    X_test = X.iloc[test_index]
    y_test = y.iloc[test_index]
    return X_train, y_train, X_test, y_test
```

[4]:
```
def get_accuracy(probs, y_test):
    labels = (probs > 0.5)
    return np.mean(labels == y_test)
```

[5]:
```python
# define LDA/QDA model
lda = LDA(store_covariance=True)
qda = QDA(store_covariance=True)
```

[6]:
```python
K = 10
kf = KFold(n_splits=K, shuffle=True, random_state=0)
PROB_LR = np.zeros(K)
PROB_LDA = np.zeros(K)
PROB_QDA = np.zeros(K)

for i, (train_index, test_index) in enumerate(kf.split(Data)):
    X_train_LR, y_train, X_test_LR, y_test = split_into_10_Fold_chunks(X_LR, y,␣
 ↪train_index, test_index)
    X_train = X_train_LR.drop(['intercept'], axis = 1)
    X_test = X_test_LR.drop(['intercept'], axis = 1)

    # fit model
    lrm = sm.GLM(y_train, X_train_LR, family=sm.families.Binomial()).fit()
    lda.fit(X_train, y_train)
    qda.fit(X_train, y_train)

    # predict the values
    probs_LR = lrm.predict(exog=X_test_LR)
    probs_LDA = lda.predict(X_test)
    probs_QDA = qda.predict(X_test)

    # save accuracy
    labels = (probs_LR > 0.5)
    PROB_LR[i] = np.mean(labels == y_test)
    PROB_LDA[i] = np.mean(probs_LDA == y_test)
    PROB_QDA[i] = np.mean(probs_QDA == y_test)

print('~ ~ ~ ~ ~ Accuracy ~ ~ ~ ~ ~')
print("Logistic Regression:", np.mean(PROB_LR))
print("LDA:", np.mean(PROB_LDA))
print("QDA:", np.mean(PROB_QDA))
```

```
~ ~ ~ ~ ~ Accuracy ~ ~ ~ ~ ~
Logistic Regression: 0.8282758620689655
LDA: 0.8383908045977012
QDA: 0.8185057471264369
```

0.2 = = = = = Running LOOCV = = = = =

[7]:
```python
def split_data_leave_one_out(X, y, i):
    # training data size 296
    X_train = X.drop(i)
```

```
    y_train = y.drop(i)
    # test data size 1
    X_test = X.iloc[i:i+1]
    y_test = y.iloc[i]
    return X_train, y_train, X_test, y_test
```

[8]:
```
PROB_LR = np.zeros(len(Data))
PROB_LDA = np.zeros(len(Data))
PROB_QDA = np.zeros(len(Data))

for i in range(len(Data)):
    X_train_LR, y_train, X_test_LR, y_test = split_data_leave_one_out(X_LR, y,␣
 ↪i)
    X_train = X_train_LR.drop(['intercept'], axis = 1)
    X_test = X_test_LR.drop(['intercept'], axis = 1)

    # fit model
    lrm = sm.GLM(y_train, X_train_LR, family=sm.families.Binomial()).fit()
    lda.fit(X_train, y_train)
    qda.fit(X_train, y_train)

    # predict the values
    probs_LR = lrm.predict(exog=X_test_LR)
    probs_LDA = lda.predict(X_test)
    probs_QDA = qda.predict(X_test)

    # save accuracy
    labels_loocv = (probs_LR > 0.5)
    PROB_LR[i] = np.mean(labels_loocv == y_test)
    PROB_LDA[i] = np.mean(probs_LDA == y_test)
    PROB_QDA[i] = np.mean(probs_QDA == y_test)

print('~ ~ ~ ~ ~ Accuracy ~ ~ ~ ~ ~')
print("Logistic Regression:", np.mean(PROB_LR))
print("LDA:", np.mean(PROB_LDA))
print("QDA:", np.mean(PROB_QDA))
```

```
~ ~ ~ ~ ~ Accuracy ~ ~ ~ ~ ~
Logistic Regression: 0.8249158249158249
LDA: 0.835016835016835
QDA: 0.8249158249158249
```

For 10-Fold CV, LDA has the best accuracy, 83.84%, among the three models.

For LOOCV, LDA still has the best accuracy, 83.5%, among the three models.