Model Assessment and Selection

YoungWoong Cho

July 2020

In this project, a right way and a wrong way of applying a K-fold cross validation(CV) as a method of model assessment is analyzed. In the wrong way of applying K-fold CV, feature selection is implemented before the dataset is split into K folds; on the other hand, in the right way of applying K-fold CV, splitting dataset into K folds is followed by the feature selection of each fold.

Dataset : Randomly generated classification dataset

Source : sklearn.datasets/make_classification( )

Features : 5000 features

Label : 0 or 1

0.1 Preliminary works

1. Import libraries

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsClassifier
```

2. Create dataset

[ ]:
```
# Create a classification dataset
X = np.random.rand(50, 5000)
Y = np.random.randint(2, size=50)
```

[ ]:
```
def X_feature_selected(X, Y, n):
  X_new = SelectKBest(chi2, k=n).fit_transform(X, Y)
  return X_new
```

[ ]:
```
def percent_error(X_predicted, Y):
  return (Y - X_predicted).T.dot(Y - X_predicted)/len(X_predicted)*100
```

0.2 Wrong way of applying K-folding Cross Validation

This is an intentional misapplication of K-folding CV in which feature selection is implemented before the cross validation.

[ ]:
```
def wrong_CV(X, Y, n):
  # First take 100 features
  X_reduced = X_feature_selected(X, Y, n)

  # K-fold CV
  # For each fold, a KNN classifier is generated using the training dataset
  # Then the classifier is tested on the validation dataset
  percent_wrng = []

  kf = KFold(n_splits=5, shuffle=True)
  for train_index, val_index in kf.split(X_reduced):
    # Split training and validation sets
    X_train = X_reduced[train_index, :]
    X_val = X_reduced[val_index, :]
    Y_train = Y[train_index]
    Y_val = Y[val_index]

    # Fit KNN with training dataset
    KNN = KNeighborsClassifier(n_neighbors=1)
    KNN.fit(X_train, Y_train)

    # Calculates MSE using validation set
    percent_wrng.append( percent_error( KNN.predict(X_val), Y_val ))

  return (np.mean(percent_wrng))
```

[ ]:
```
X_wrong = X_feature_selected(X, Y, 100)
```

[ ]:
```
# Computer average CV error rate repeated 50 times
error = []
for i in range(50):
  error.append(wrong_CV(X, Y, 100))
print("Error rate using wrong way of CV: %.2f%%" %np.mean(error))
```

```
Error rate using wrong way of CV: 4.12%
```

0.3 Correct way of applying K-fold Cross Validation

This is a correct way of implementing K-fold CV with feature selection in which the dataset is first split into K-folds and then the feature selection is implemented on each fold.

[ ]:
```
def correct_CV(X, Y, n):
  percent_wrng = []
```

```
  kf = KFold(n_splits=5, shuffle=True)
  for train_index, val_index in kf.split(X):
    # Construct training and validation dataset for this fold
    X_train = X[train_index, :]
    Y_train = Y[train_index]
    X_val = X[val_index, :]
    Y_val = Y[val_index]

    # Select top 100 features for this fold
    X_train_reduced = X_feature_selected(X_train, Y_train, n)
    X_val_reduced = X_feature_selected(X_val, Y_val, n)

    # Fit KNN with training dataset with selected features
    KNN = KNeighborsClassifier(n_neighbors=1)
    KNN.fit(X_train_reduced, Y_train)

    # Calculates MSE using validation set
    percent_wrng.append( percent_error( KNN.predict(X_val_reduced), Y_val ) )

  return np.mean(percent_wrng)
```

[ ]:
```
# Computer average CV error rate repeated 50 times
error = []
for i in range(50):
  error.append(correct_CV(X, Y, 100))
print("Error rate using correct way of CV: %.2f%%" %np.mean(error))
```

```
Error rate using correct way of CV: 50.56%
```