T2 - CV

February 28, 2021

1 K-fold cross-validation

1.1 Setting up

[ ]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline

# Breast cancer data
from sklearn.datasets import load_breast_cancer

# Load data
dataObj = load_breast_cancer()
X = dataObj.data
y = dataObj.target

# Splitting data
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.20,
    stratify=y,
    random_state=1)

# Constructing a pipeline object
pipe_lr = Pipeline([('scl', StandardScaler()),
                    ('pca', PCA(n_components=2)),
                    ('clf', LogisticRegression(random_state=1))])
```

1.2 K-Fold cross-validation

[ ]:
```
from sklearn.model_selection import KFold
kf = KFold(n_splits=10, random_state=1)
print(kf)
```

[ ]:
```
genSplit = kf.split(X_train,y_train)
print(genSplit)
```

[ ]:
```
#Note that generator needs to be created since it is exhausted after used.
genSplit = kf.split(X_train,y_train)

for idxTrain, idxVal in genSplit:
    print(idxTrain[0:10],"...", idxVal[0:10], "...")
```

[ ]:
```
genSplit = kf.split(X_train,y_train)

for idxTrain, idxVal in genSplit:
    print(idxTrain.shape, idxVal.shape, np.bincount(y_train[idxTrain]))
```

[ ]:
```
genSplit = kf.split(X_train,y_train)

trainScores = []
valScores = []
for count, (idxTrain, idxVal) in enumerate(genSplit):
    # Training
    pipe_lr.fit(X_train[idxTrain], y_train[idxTrain])
    # Traning score
    trainScore = pipe_lr.score(X_train[idxTrain], y_train[idxTrain])
    trainScores.append(trainScore)
    # Validation score
    valScore = pipe_lr.score(X_train[idxVal], y_train[idxVal])
    valScores.append(valScore)
    # Printing
    print(f"Fold:{count+1:2d}, Training accuracy:{trainScore:6.3f}, Validation␣
 ↪accuracy:{valScore:6.3f}")

print("\nTraining accuracy")
print(f"Mean:{np.mean(trainScores):6.3f}")
print(f"Std:{np.std(trainScores):6.3f}")

print("\nCrossed-validation accuracy")
print(f"Mean:{np.mean(valScores):6.3f}")
print(f"Std:{np.std(valScores):6.3f}")
```

1.3 Stratified K-Folds cross-validation

[ ]:
```
from sklearn.model_selection import StratifiedKFold
skf = StratifiedKFold(n_splits=10, random_state=1)
print(skf)
```

[ ]:
```
genSplit = skf.split(X_train,y_train)
```

```
for idxTrain, idxVal in genSplit:
    print(idxTrain.shape, idxVal.shape, np.bincount(y_train[idxTrain]))
```

[ ]:
```
genSplit = skf.split(X_train,y_train)

trainScores = []
valScores = []
for count, (idxTrain, idxVal) in enumerate(genSplit):
    # Training
    pipe_lr.fit(X_train[idxTrain], y_train[idxTrain])
    # Traning score
    trainScore = pipe_lr.score(X_train[idxTrain], y_train[idxTrain])
    trainScores.append(trainScore)
    # Validation score
    valScore = pipe_lr.score(X_train[idxVal], y_train[idxVal])
    valScores.append(valScore)
    # Printing
    print(f"Fold:{count+1:2d}, Training accuracy:{trainScore:6.3f}, Validation␣
 ↪accuracy:{valScore:6.3f}")

print("\nTraining accuracy")
print(f"Mean:{np.mean(trainScores):6.3f}")
print(f"Std:{np.std(trainScores):6.3f}")

print("\nCrossed-validation accuracy")
print(f"Mean:{np.mean(valScores):6.3f}")
print(f"Std:{np.std(valScores):6.3f}")
```

1.4 Stratified k-fold cross-validation (SKL)

[ ]:
```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(estimator=pipe_lr,
                         X=X_train,
                         y=y_train,
                         cv=10,
                         n_jobs=1)

print("\nCrossed-validation accuracy")
print(f"Mean:{np.mean(scores):6.3f}")
print(f"Std:{np.std(scores):6.3f}")
```