Classifiers

Report 7

Information leakage avoidance

Authors:
Piotr Pawełko
Piotr Wojsa

Laboratory date:
06.05.2022

1. Choosing feature selection and classification methods

For feature selection we choose available in *sklearn* library *SelectKBest* function, which can select *k* best features based on highest score.

```
from sklearn.feature_selection import SelectKBest, chi2
def Select(dataset, labels, k_features):
    selection = SelectKBest(score_func=chi2, k=k_features).fit(dataset, 
labels)
    scores = pd.DataFrame(selection.scores_)
    scores.columns = ['Score']
    features = scores.nlargest(k_features, 'Score').index.values
    new_dataset = dataset[features]
    return new_dataset, features, scores
```

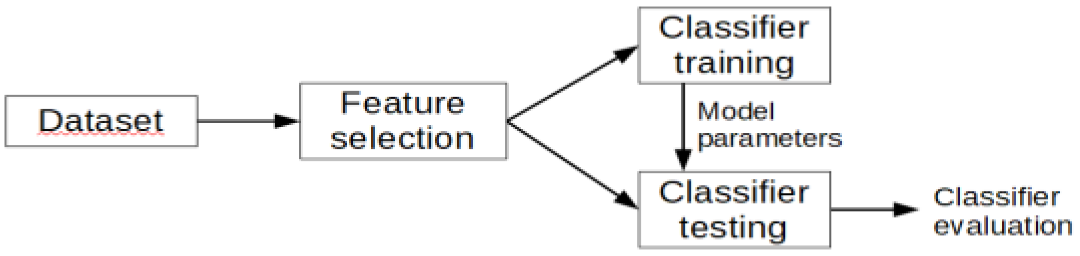
For classification we choose *SVM* method that is also available in *sklearn* library.

```
from sklearn import svm
from sklearn.metrics import confusion_matrix
def classifier(train, test, train_label, test_label):
    SVM = svm.SVC(kernel="rbf").fit(train, train_label)
    SVM.fit(train, train_label)
    prediction = SVM.predict(test)

    TN, FP, FN, TP = confusion_matrix(test_label, prediction).ravel()
    Accuracy = (TN + TP) / (TN + FP + FN + TP)
    Error = 1 - Accuracy
    return Accuracy, Error
```

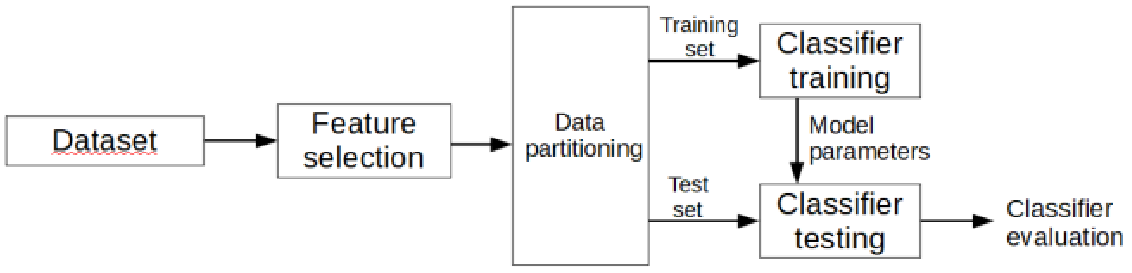
2. Applying scenarios

First scenario



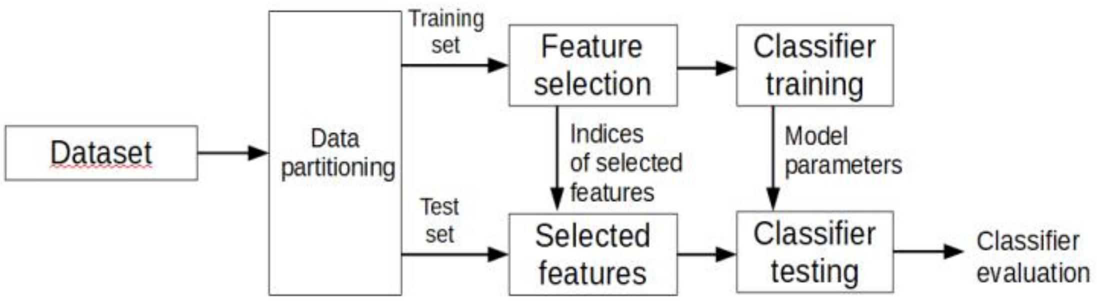
```
[dataset1_S1, features1_S1, scores1_S1] = Select(dataset1, labels, 10)
[dataset2_S1, features2_S1, scores2_S1] = Select(dataset2, labels, 10)
[S1_Acc1, S1_Err1] = classifier(dataset1_S1, dataset1_S1, 
labels.values.ravel(), labels.values.ravel())
[S1_Acc2, S1_Err2] = classifier(dataset2_S1, dataset2_S1, 
labels.values.ravel(), labels.values.ravel())
```

Second Scenario



```
dataset1_S2, features1_S2, scores1_S2 = Select(dataset1, labels, 10)
dataset2_S2, features2_S2, scores2_S2 = Select(dataset2, labels, 10)
x1_train_S2, x1_test_S2, y1_train_S2, y1_test_S2 =
train_test_split(dataset1_S2,labels,test_size=0.5)
x2_train_S2, x2_test_S2, y2_train_S2, y2_test_S2 =
train_test_split(dataset2_S2,labels,test_size=0.5)
S2_Acc1, S2_Err1 = classifier(x1_train_S2, x1_test_S2,
y1_train_S2.values.ravel(), y1_test_S2.values.ravel())
S2_Acc2, S2_Err2 = classifier(x2_train_S2, x2_test_S2,
y2_train_S2.values.ravel(), y2_test_S2.values.ravel())
```

Third Scenario

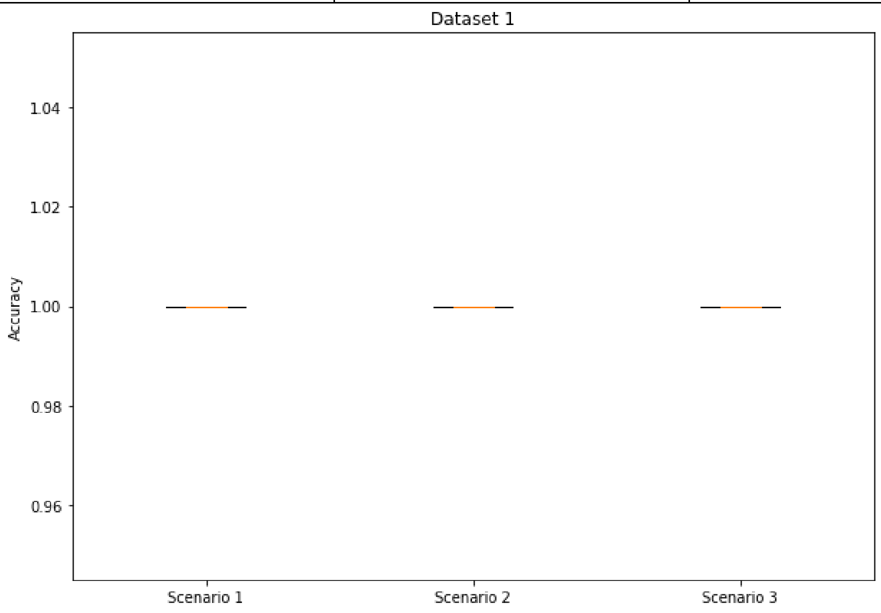


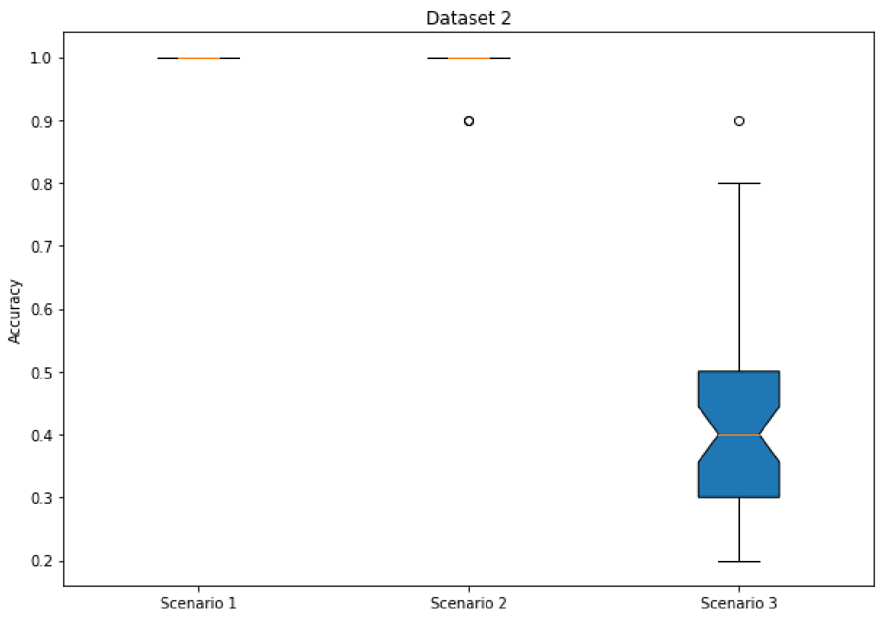
```
x1_train_S3, x1_test_S3, y1_train_S3, y1_test_S3 =
train_test_split(dataset1,labels,test_size=0.5)
x2_train_S3, x2_test_S3, y2_train_S3, y2_test_S3 =
train_test_split(dataset2,labels,test_size=0.5)
x1_train_S3_2,features1_S3_2,scores1_S3_2=Select(x1_train_S3, y1_train_S3, 10)
x2_train_S3_2,features2_S3_2,scores2_S3_2=Select(x2_train_S3, y2_train_S3, 10)
x1_test_S3 = x1_test_S3[features1_S3_2]
x2_test_S3 = x2_test_S3[features2_S3_2]
S3_Acc1, S3_Err1 = classifier(x1_train_S3_2, x1_test_S3,
y1_train_S3.values.ravel(), y1_test_S3.values.ravel())
S3_Acc2, S3_Err2 = classifier(x2_train_S3_2, x2_test_S3,
y2_train_S3.values.ravel(), y2_test_S3.values.ravel())
```

3. Tests

Datasets used for testing: S4_set1, S4_set2

	Classification Error	
	Set 1	**Set 2**
Validation Scenario 1	0	0
Validation Scenario 2	0	0.012
Validation Scenario 3	0.03	0.56





For first two validation scenarios classification error was the same or very similar for both sets. The difference starts to appear only for scenario 3, where error value for set 2 shows that classes A and B for dataset2 are the same.

Set 1 is a set where classes A and B are different. Did not find indices of discriminative features.

Tests on real data

	Classification Error
Validation Scenario 1	0.04
Validation Scenario 2	0.08
Validation Scenario 3	0.015

4. Result

In work on dataset it is unusually that test set is provided separately to a train set, because of that most of time partitioning data must be performed by data scientist in a way that prevents information leaks. Information leakage is when information from outside the training dataset is used to create the model. This additional information can allow the model to learn or know something that it otherwise would not know and in turn invalidate the estimated performance of the mode being constructed. The easiest way to discover if there is an information leakage is achieving performance that seems too good to be true. It happend during testing for Validation Scenarios 1 and 2. The best way to prevent data leakage is to perform partitioning before any other action as it is presented for Validation Scenario 3 called Holdout method.