KFold Cross Validation Python Tutorial

In [83]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
import numpy as np
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt
digits = load_digits()
```

In [84]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(digits.data,digits.
target,test_size=0.3)
```

Logistic Regression

In [85]:

```
lr = LogisticRegression(solver='liblinear',multi_class='ovr')
lr.fit(X_train, y_train)
lr.score(X_test, y_test)
```

Out[85]:

```
0.9666666666666667
```

SVM

In [86]:

```
svm = SVC(gamma='auto')
svm.fit(X_train, y_train)
svm.score(X_test, y_test)
```

Out[86]:

```
0.3685185185185185
```

Random Forest

```
In [87]: rf = RandomForestClassifier(n_estimators=40)
         rf.fit(X_train, y_train)
         rf.score(X_test, y_test)

Out[87]: 0.9703703703703703
```

KFold cross validation

Basic example

```
In [88]: from sklearn.model_selection import KFold
         kf = KFold(n_splits = 3)
         kf

Out[88]: KFold(n_splits=3, random_state=None, shuffle=False)
```

```
In [89]: for train_index, test_index in kf.split([1,2,3,4,5,6,7,8,9]):
             print(train_index,test_index)

         [3 4 5 6 7 8] [0 1 2]
         [0 1 2 6 7 8] [3 4 5]
         [0 1 2 3 4 5] [6 7 8]
```

Use KFold for our digits example

```
In [90]: def get_score(model,X_train,X_test,y_train,y_test):
             model.fit(X_train,y_train)
             return model.score(X_test,y_test)
```

```
In [91]: from sklearn.model_selection import StratifiedKFold
         folds = StratifiedKFold(n_splits=3)

         scores_logistic = []
         scores_svm = []
         scores_rf = []
```

```
for train_index, test_index in folds.split(digits.data,digits.target):
    X_train, X_test, y_train, y_test = digits.data[train_index], digits
.data[test_index], \
                                       digits.target[train_index], digi
ts.target[test_index]
    scores_logistic.append(get_score(LogisticRegression(solver='libline
ar',multi_class='ovr'), X_train, X_test, y_train, y_test))
    scores_svm.append(get_score(SVC(gamma='auto'), X_train, X_test, y_t
rain, y_test))
    scores_rf.append(get_score(RandomForestClassifier(n_estimators=40),
 X_train, X_test, y_train, y_test))
```

In [92]:
```
scores_logistic
```

Out[92]:
```
[0.8948247078464107, 0.9532554257095158, 0.9098497495826378]
```

In [93]:
```
scores_svm
```

Out[93]:
```
[0.3806343906510851, 0.41068447412353926, 0.5125208681135225]
```

In [94]:
```
scores_rf
```

Out[94]:
```
[0.9248747913188647, 0.9465776293823038, 0.9248747913188647]
```

cross_val_score function

In [95]:
```
from sklearn.model_selection import cross_val_score
```

Logistic regression model performance using cross_val_score

In [97]:
```
cross_val_score(LogisticRegression(solver='liblinear',multi_class='ovr'
), digits.data, digits.target,cv=3)
```

Out[97]:
```
array([0.89482471, 0.95325543, 0.90984975])
```

svm model performance using cross_val_score

In [98]:

```
cross_val_score(SVC(gamma='auto'), digits.data, digits.target,cv=3)
```

Out[98]:

```
array([0.38063439, 0.41068447, 0.51252087])
```

random forest performance using cross_val_score

In [99]:

```
cross_val_score(RandomForestClassifier(n_estimators=40),digits.data, digits.target,cv=3)
```

Out[99]:

```
array([0.92988314, 0.93656093, 0.92320534])
```

cross_val_score uses stratifield kfold by default

Parameter tunning using k fold cross validation

In [100]:

```
scores1 = cross_val_score(RandomForestClassifier(n_estimators=5),digits.data, digits.target, cv=10)
np.average(scores1)
```

Out[100]:

```
0.892029795158287
```

In [101]:

```
scores2 = cross_val_score(RandomForestClassifier(n_estimators=20),digits.data, digits.target, cv=10)
np.average(scores2)
```

Out[101]:

```
0.9332091868404717
```

In [102]:

```
scores3 = cross_val_score(RandomForestClassifier(n_estimators=30),digits.data, digits.target, cv=10)
np.average(scores3)
```

```
Out[102]: 0.9409931719428926
```

```
In [103]: scores4 = cross_val_score(RandomForestClassifier(n_estimators=40),digit
          s.data, digits.target, cv=10)
          np.average(scores4)
```

```
Out[103]: 0.9443358162631904
```

Here we used cross_val_score to fine tune our random forest classifier and figured that having around 40 trees in random forest gives best result.

Exercise

Use iris flower dataset from sklearn library and use cross_val_score against following models to measure the performance of each. In the end figure out the model with best performance,

1. Logistic Regression
2. SVM
3. Decision Tree
4. Random Forest