机器学习算法系列（30）：Scikit-Learn总结

Scikit-learn是一个很受欢迎的机器学习方面的python工具包，它定义的一些范式和处理流程影响深远，所以，了解这个工具包对于机器学习算法的整个流程会有一个整体的了解。它已经实现了很多方法帮助我们便捷的处理数据，例如，划分数据集为训练集和验证集，交叉验证，数据预处理，归一化等等。

一、性能评价指标

```
# 计算均方误差
from sklearn import metrics
import numpy as np
rmse = np.sqrt(metrics.mean_squared_error(y_test,y_pred))

# 计算准确率
acc = metrics.accuracy_score(y_test,y_pred)

#混淆矩阵
cm = metrics.confusion_matrix(y_test,y_pred)

# classification_report
cr = metrics.classification_report(y_true,y_pred)

# ROC AUC 曲线
from sklearn.metrics import roc_curve,auc
```

二、数据集划分

```
from sklearn import cross_validation
X_train,X_test,y_train,y_test = cross_validation.train_test_split(X,y,test_size=0.3
,random_state=0)

# K折
from sklearn.cross_validation import KFold
kf = KFold(n_samples, n_folds=2)
for train, test in kf:
    print("%s %s" % (train, test))
```

```python
# 保证不同的类别之间的均衡，这里需要用到标签labels
from sklearn.cross_validation import StratifiedKFold
labels = [0, 0, 0, 0, 1, 1, 1, 1, 1, 1]
skf = StratifiedKFold(labels, 3)
for train, test in skf:
    print("%s %s" % (train, test))

# 留一交叉验证
from sklearn.cross_validation import LeaveOneOut
loo = LeaveOneOut(n_samples)
for train, test in loo:
    print("%s %s" % (train, test))

# 留P交叉验证
from sklearn.cross_validation import LeavePOut
lpo = LeavePOut(n_samples, p=2)
for train, test in lpo:
    print("%s %s" % (train, test))


# 按照额外提供的标签留一交叉验证,常用的情况是按照时间序列
from sklearn.cross_validation import LeaveOneLabelOut
labels = [1, 1,1, 2, 2]
lolo = LeaveOneLabelOut(labels)
for train, test in lolo:
    print("%s %s" % (train, test))

# 按照额外提供的标签留P交叉验证
from sklearn.cross_validation import LeavePLabelOut
labels = [1, 1, 2, 2, 3, 3,3]
lplo = LeavePLabelOut(labels, p=2)
for train, test in lplo:
    print("%s %s" % (train, test))

# 随机分组
from sklearn.cross_validation import ShuffleSplit
ss = ShuffleSplit(16, n_iter=3, test_size=0.25,random_state=0)
for train_index, test_index in ss:
    print("%s %s" % (train_index, test_index))

# 考虑类别均衡的随机分组
from sklearn.cross_validation import StratifiedShuffleSplit
import numpy as np
X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
y = np.array([0, 0, 1, 1])
sss = StratifiedShuffleSplit(y, 3, test_size=0.5, random_state=0)
for train, test in sss:
    print("%s %s" % (train, test))
```

三、特征选择

```
# 基于方差的特征选择
from sklearn import feature_selection
vt = feature_selection.VarianceThreshold(threshold='')
vt.fit(X_train)
X_train_transformed = vt.transform(X_train)
X_test_transformed = vt.transform(X_test)

# 按照某种排序规则 选择前K个特征
# 除了使用系统定义好的函数f_classif，还可以自己定义函数
sk = SelectKBest(feature_selection.f_classif,k=100)
sk.fit(X_train,y_train)
X_train_transformed = sk.transform(X_train)
X_test_transformed = sk.transform(X_test)

# 递归特征消除
rfecv = RFECV(estimator=svc, step=step, cv=StratifiedKFold(y, n_folds = n_folds),sc
oring='accuracy')
rfecv.fit(X_train, y_train)
X_train_transformed = rfecv.transform(X_train)
X_test_transformed = rfecv.transform(y_train)

# 使用L1做特征选择
from sklearn.svm import LinearSVC
lsvc = LinearSVC(C=1, penalty="l1", dual=False)
lsvc.fit(X_train,y_train)
X_train_transformed = lsvc.transform(X_train)
X_test_transformed = lsvc.transform(y_train)

# 基于树的特征选择
from sklearn.ensemble import ExtraTreesClassifier
etc = ExtraTreesClassifier()
etc.fit(X_train, y_train)
X_train_transformed = etc.transform(X_train)
X_test_transformed = etc.transform(X_test)

# 基于线性判别分析做特征选择
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis(solver='lsqr',shrinkage='auto')
lda.fit(X_train, y_train)
X_train_transformed = lda.transform(X_train)
X_test_transformed = lda.transform(X_test)
```

> 基于方差的特征选择

```
from sklearn.feature_selection import VarianceThreshold
X = [[0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 1, 1], [0, 1, 0], [0, 1, 1]]
sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
X2 = sel.fit_transform(X)
X2
Out[5]:
array([[0, 1],
       [1, 0],
       [0, 0],
       [1, 1],
       [1, 0],
       [1, 1]])
```

可以看到，方差小于0.16的只有第一维特征，所以X2保留下来的是原来的第二维和第三维特征。这应该是最简单的特征选择方法了：假设某特征的特征值只有0和1，并且在所有输入样本中，95%的实例的该特征取值都是1，那就可以认为这个特征作用不大。如果100%都是1，那这个特征就没意义了。当特征值都是离散型变量的时候这种方法才能用，如果是连续型变量，就需要将连续变量离散化之后才能用，而且实际当中，一般不太会有95%以上都取某个值的特征存在，所以这种方法虽然简单但是不太好用。可以把它作为特征选择的预处理，先去掉那些取值变化小的特征，然后再从接下来提到的的特征选择方法中选择合适的进行进一步的特征选择。

> Univariate feature selection （单变量特征选择）

主要使用统计的方法计算各个统计值，再根据一定的阈值筛选出符合要求的特征，去掉不符合要求的特征。

主要的统计方法有

- F值分类： f_classif
- 值回归：f_regression
- 卡方统计：chi2 (适用于非负特征值和稀疏特征值)

主要的选择策略

- 选择排名前K的特征：SelectKbest
- 选择前百分之几的特征：SelectPercentile
- SelectFpr：Select features based on a false positive rate test.
- SelectFdr：Select features based on an estimated false discovery rate.
- SelectFwe：Select features based on family-wise error rate.
- GenericUnivariateSelect：Univariate feature selector with configurable mode.

其中

- false positive rate：FP / (FP + TP) 假设类别为0，1；记0为negative,1为positive, FPR就是实际的类别是0，但是分类器错误的预测为1的个数 与 分类器预测的类别为1的样本的总数（包括正确的预测为1和错误的预测为1） 的比值。
- estimated false discovery rate: 错误的拒绝原假设的概率；
- family-wise error rate: 至少有一个检验犯第一类错误的概率；假设检验的两类错误： > - 第一类错误：原假设是正确的，但是却被拒绝了。(用α表示） > - 第二类错误：原假设是错误的，但是却被接受了。(用β表示)

http://ff120.github.io/2017/05/14/机器学习专题/机器学习_Scikit-Learn使用技巧/