第9章　scikit-learn

Jupyter金融应用从入门到实践

scikit-learn是基于Python语言的简单高效的数据分析开源工具和机器学习库。它同时还与科学计算的Python包（NumPy、SciPy、Matplotlib）很好地集成在一起。在实际的使用场景中，我们可以直接使用scikit-learn工具提供的方法来完成数据预处理、分类、回归、降维、模型选择等常用操作。

线性回归

在 scikit-learn 中线性回归的算法模块是最基础的模块，其函数初始化方法如下所示。

```
# 引入线性回归
from sklearn.linear_model import LinearRegression
clf = LinearRegression()
# init 方法
def __init__(self, fit_intercept=True, normalize=False, copy_X=True, n_jobs=1)
```

在初始化函数中有几个参数可以自行设置，详细内容见表 9-1。

表 9-1　线性回归模型的具体参数

参数	取值	说明
copy_X	bool 类型	是否对 X 复制，如果选择 False，则直接对原数据进行覆盖
fit_intercept	bool 类型	是否存在截距，默认为 True
n_jobs	‘None’、数值类型	计算时使用的核数，默认为 1
normalize	bool 类型	是否将数据归一化，默认为 True

逻辑回归

在 scikit-learn 中，逻辑回归函数的初始化方法如下所示。

初始化

```
1.  # 引入逻辑回归模型
2.  from sklearn.linear_model import LogisticRegression
3.  clf = LogisticRegression()
4.  ## init 函数
5.  def __init__(self,penalty='l2',dual=False,tol=1e-4,C=1.0,fit_intercept=True,
6.                intercept_scaling=1,class_weight=None,random_state=None,
7.                solver='liblinear',max_iter=100,multi_class='ovr',
8.                verbose=0,warm_start=False,n_jobs=1):
```

下面的示例选择 sklearn 自带的鸢尾花卉数据集来演示，首先是加载训练集，然后是训练模型，最后预测新的样本数据。

训练和预测

[In]

```
1.  from sklearn.datasets import load_iris
2.  from sklearn.linear_model import LogisticRegression
3.  #加载数据
4.  X, y = load_iris(return_X_y=True)
5.  #训练模型
6.  clf = LogisticRegression(random_state=0).fit(X, y)
7.  #预测
8.  clf.predict(X[:2, :])
```

[Out]

```
array([0, 0])
```

scikit-learn常用函数

支持向量机

初始化

```
# 引入模型
from sklearn.svm import SVC
svc = SVC()
# init 函数
def __init__(self, C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0,shrinking=True,
             probability=False, tol=1e-3, cache_size=200, class_weight=None,verbose=False,
             max_iter=-1, decision_function_shape='ovr', random_state=None):
```

训练

```
import numpy as np
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
y = np.array([1, 1, 2, 2])
from sklearn.svm import SVC
clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
clf.fit(X, y)
```

通过训练完成相应模型之后，就可以对新的样本值进行预测。

预测

```
[In]  clf.predict([[-0.8, -1]])
[Out] array([1])
```

决策树

决策树模型初始化方式如以下例子所示。

初始化

```
import json  # 引入决策树模型
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
# init函数
def __init__(self, criterion="gini",splitter="best",max_depth=None,min_samples_split=2,
             min_samples_leaf=1, min_weight_fraction_leaf=0., max_features=None,
             random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.,
             min_impurity_split=None, class_weight=None, presort=False):
```

验证

cross_val_score()函数表示对数据集进行指定次数的交叉验证并为每次验证效果评测

```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0)
iris = load_iris()
cross_val_score(clf, iris.data, iris.target, cv=10)
```

```
array([1. , 0.93333333, 1. , 0.93333333, 0.93333333, 0.86666667, 0.93333333, 1. , 1. ,
1. ])
```

scikit-learn常用函数

随机森林

初始化

```
# 引入随机森林
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
# init 函数
def __init__(self,n_estimators=10,criterion="gini",max_depth=None,
             min_samples_split=2,min_samples_leaf=1,
             min_weight_fraction_leaf=0.,max_features="auto",
             max_leaf_nodes=None,min_impurity_decrease=0.,
             min_impurity_split=None,bootstrap=True,oob_score=False,
             n_jobs=1,random_state=None,verbose=0,
             warm_start=False,class_weight=None):
```

训练

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=4,n_informative=2,
                           n_redundant=0, random_state=0, shuffle=False)
clf = RandomForestClassifier(max_depth=2, random_state=0)
clf.fit(X, y)
```

预测

```
[In]  clf.predict([[0, 0, 0, 0]])
[Out] array([1])
```

K均值聚类

例子列举了 6 个样本，将其聚合成两种类型。从最终结果可以看出，前 3 个样本被聚成一类，后 3 个样本被聚成另外一类。

[In]
```
1.  from sklearn.cluster import KMeans
2.  import numpy as np
3.  X = np.array([[1, 2], [1, 4], [1, 0],[10, 2], [10, 4], [10, 0]])
4.  kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
5.  kmeans.labels_
```

[Out]
```
array([1, 1, 1, 0, 0, 0])
```

使用训练好的聚类模型进行预测，可以发现样本 1 被归为类型 1，样本 2 被归为类型 0。

[In]
```
kmeans.predict([[0, 0], [12, 3]])
```

[Out]
```
array([1, 0])
```

通过“cluster_centers_”属性值输出聚类质心。

[In]
```
kmeans.cluster_centers_
```

[Out]
```
array([[10., 2.], [ 1., 2.]])
```

谢谢！