免费电子书

学习

scikit-learn

Free unaffiliated eBook created from
Stack Overflow contributors.

#scikit-learn

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: scikit-learn

It is an unofficial and free scikit-learn ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official scikit-learn.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: scikit-learn

`scikit-learn`python。 pythonNumPySciPymatplotlib

`scikit-learn`。

Examples

scikit-learn

scikit-learn

- Python> = 2.6> = 3.3
- NumPy> = 1.6.1
- SciPy> = 0.9。

`pip` pythonpython

```
pip install scikit-learn
```

Linux`conda`

```
conda install scikit-learn
```

`scikit-learn` shell

```
python -c 'import sklearn; print(sklearn.__version__)'
```

WindowsMac OSX

WindowspythonMac OSXLinux CanopyAnaconda*scikit-learn* 。

iris

```
import sklearn.datasets
iris_dataset = sklearn.datasets.load_iris()
X, y = iris_dataset['data'], iris_dataset['target']
```

。 `train_test_split`Xy `train_size=0.75` 75。

k- 。 `fit`。

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.75)
from sklearn.neighbors import KNeighborsClassifier
```

```
clf = KNeighborsClassifier(n_neighbors=3)
clf.fit(X_train, y_train)
```

```
clf.score(X_test, y_test) # Output: 0.94736842105263153
```

◦ /◦ cross_val_scorecross_val_score◦ 5◦

```
from sklearn.cross_validation import cross_val_score
scores = cross_val_score(clf, X, y, cv=5)
print(scores)
# Output: array([ 0.96666667,  0.96666667,  0.93333333,  0.96666667,  1.        ])
print "Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() / 2)
# Output: Accuracy: 0.97 (+/- 0.03)
```

◦ sklearn ◦

◦

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier

pipeline = make_pipeline(StandardScaler(), KNeighborsClassifier(n_neighbors=4))
```

◦ ◦

```
# fitting a classifier
pipeline.fit(X_train, y_train)
# getting predictions for the new data sample
pipeline.predict_proba(X_test)
```

◦

- sklearn.base.ClassifierMixin
- sklearn.base.RegressorMixin
- sklearn.base.TransformerMixin

numpy.arraypandas.DataFrameSnumpy.array S

idid◦

```
import numpy
data = numpy.arange(10).reshape(5, 2)
print(data)

Output:
[[0 1]
 [2 3]
 [4 5]
 [6 7]
 [8 9]]
```

sklearn52。

sklearnsklearn.datasets。 Fisher

```
import sklearn.datasets
iris_dataset = sklearn.datasets.load_iris()
iris_dataset.keys()
['target_names', 'data', 'target', 'DESCR', 'feature_names']
```

target_names 。 。

datatarget。 Xy

```
X, y = iris_dataset['data'], iris_dataset['target']
X.shape, y.shape
((150, 4), (150,))
```

```
numpy.unique(y)
array([0, 1, 2])
```

Xy1504。 0,12。

fit()Xy。

---

sklearn.datasets

load_boston()		506	
load_breast_cancer()		569	
load_diabetes()		442	
load_digits(n_class)		1797	
load_iris()		150	
load_linnerud()	Linnerud	20	

http //scikit-learn.org/stable/datasets/

scikit。 。

sklearn.datasets

- load_mlcompmlcomp.org。 。
- fetch_lfw_pairsfetch_lfw_peoplehttp://vis-www.cs.umass.edu/lfw/Labeled FacesLFW。 200 MB。 。

scikit-learn https://riptutorial.com/zh-CN/scikit-learn/topic/1035/scikit-learn

2:

Examples

◦ SVM`sklearn.svm.SVC``sklearn.svm.SVR` ◦

2D◦

```
import numpy as np
```

xy

```
x0, x1 = np.random.randn(10, 2), np.random.randn(10, 2) + (1, 1)
x = np.vstack((x0, x1))

y = [0] * 10 + [1] * 10
```

x0,01,1◦

```
from sklearn import svm

svm.SVC(kernel='linear').fit(x, y)
```

0,0

```
>>> svm.SVC(kernel='linear').fit(x, y).predict([[0, 0]])
array([0])
```

0◦

```
svm.SVR(kernel='linear').fit(x, y)
```

RandomForestClassifier

◦

```
from sklearn.ensemble import RandomForestClassifier
```

```
train = [[1,2,3],[2,5,1],[2,1,7]]
target = [0,1,0]
```

`target`◦

RandomForestlearnfit

```
rf = RandomForestClassifier(n_estimators=100)
```

```
rf.fit(train, target)
```

```
test = [2,2,3]
predicted = rf.predict(test)
```

f1- 。

sklearn docs

```
from sklearn.metrics import classification_report
y_true = [0, 1, 2, 2, 2]
y_pred = [0, 0, 2, 2, 1]
target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y_true, y_pred, target_names=target_names))
```

-

```
             precision    recall  f1-score   support

    class 0       0.50      1.00      0.67         1
    class 1       0.00      0.00      0.00         1
    class 2       1.00      0.67      0.80         3

avg / total       0.70      0.60      0.61         5
```

GradientBoostingClassifier

Gradient Boosting。 。

```
from sklearn.ensemble import GradientBoostingClassifier
```

```
from sklearn.datasets import load_iris

iris_dataset = load_iris()

X, y = iris_dataset.data, iris_dataset.target
```

。

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=0)
```

GradientBoostingClassifier。

```
gbc = GradientBoostingClassifier()
gbc.fit(X_train, y_train)
```

```
# We are using the default classification accuracy score
>>> gbc.score(X_test, y_test)
```

```
1
```

100

```
>>> gbc.n_estimators
100
```

n_estimators。

a> 7。

3。

```
from sklearn.tree import DecisionTreeClassifier

# Define training and target set for the classifier
train = [[1,2,3],[2,5,1],[2,1,7]]
target = [10,20,30]

# Initialize Classifier.
# Random values are initialized with always the same random seed of value 0
# (allows reproducible results)
dectree = DecisionTreeClassifier(random_state=0)
dectree.fit(train, target)

# Test classifier with other, unknown feature vector
test = [2,2,3]
predicted = dectree.predict(test)

print predicted
```

```
import pydot
import StringIO

dotfile = StringIO.StringIO()
tree.export_graphviz(dectree, out_file=dotfile)
(graph,)=pydot.graph_from_dot_data(dotfile.getvalue())
graph.write_png("dtree.png")
graph.write_pdf("dtree.pdf")
```

Logistic

LR。 `linear_model`

```
from sklearn.linear_model import LogisticRegression
```

sklearn LRL2L1。sklearn

```
from sklearn.datasets import make_hastie_10_2

X,y = make_hastie_10_2(n_samples=1000)
```

X`n_samples X 10`y-1+1。

70-30

```
from sklearn.model_selection import train_test_split
#sklearn.cross_validation in older scikit versions

data_train, data_test, labels_train, labels_test = train_test_split(X,y, test_size=0.3)
```

LR

```
# Initialize Classifier.
LRC = LogisticRegression()
LRC.fit(data_train, labels_train)

# Test classifier with the test data
predicted = LRC.predict(data_test)
```

Confusion

```
from sklearn.metrics import confusion_matrix

confusion_matrix(predicted, labels_test)
```

https://riptutorial.com/zh-CN/scikit-learn/topic/2468/

3:

Examples

。

0。

。

```
X = [[0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 1, 1], [0, 1, 0], [0, 1, 1]]
```

36。80。*0.8 *1 - 0.8*。

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

。

“”。*3*。

```
from sklearn.feature_selection import VarianceThreshold
import numpy as np

# generate dataset
np.random.seed(0)

feat1 = np.random.normal(loc=0, scale=.1, size=100) # normal dist. with mean=0 and std=.1
feat2 = np.random.normal(loc=0, scale=10, size=100) # normal dist. with mean=0 and std=10
feat3 = np.random.uniform(low=0, high=10, size=100) # uniform dist. in the interval [0,10)
data = np.column_stack((feat1,feat2,feat3))

data[:5]
# Output:
# array([[  0.17640523,  18.83150697,   9.61936379],
#        [  0.04001572, -13.47759061,   2.92147527],
#        [  0.0978738 , -12.70484998,   2.4082878 ],
#        [  0.22408932,   9.69396708,   1.00293942],
#        [  0.1867558 , -11.73123405,   0.1642963 ]])

np.var(data, axis=0)
# Output: array([  1.01582662e-02,   1.07053580e+02,   9.07187722e+00])

sel = VarianceThreshold(threshold=0.1)
sel.fit_transform(data)[:5]
# Output:
```

```
# array([[ 18.83150697,   9.61936379],
#        [-13.47759061,   2.92147527],
#        [-12.70484998,   2.4082878 ],
#        [  9.69396708,   1.00293942],
#        [-11.73123405,   0.1642963 ]])
```

。 。

https://riptutorial.com/zh-CN/scikit-learn/topic/4909/

4:

Examples

。

y X OLS*β*

minβ | *y* ^ - *y* | *2* *2*

y ^ =*Xβ*。

sklearn`sklearn.linear_model.LinearRegression`。

OLS

- Classfication
- upvotes

`LinearRegression`。

`X`

```
import numpy as np

X = np.random.randn(100, 3)
```

`yX`

```
beta = np.array([[1, 1, 0]])
y = (np.dot(x, beta.T) + 0.01 * np.random.randn(100, 1))[:, 0]
```

`y`β。

`Xy`

```
>>> linear_model.LinearRegression().fit(x, y).coef_
array([  9.97768469e-01,   9.98237634e-01,   7.55016533e-04])
```

`beta`。

https://riptutorial.com/zh-CN/scikit-learn/topic/5190/

5:

Examples

- ◦ ◦ X_test, y_test ◦ “”◦

scikit-learn train_test_split◦

```
>>> import numpy as np
>>> from sklearn import cross_validation
>>> from sklearn import datasets
>>> from sklearn import svm

>>> iris = datasets.load_iris()
>>> iris.data.shape, iris.target.shape
((150, 4), (150,))
```

40

```
>>> X_train, X_test, y_train, y_test = cross_validation.train_test_split(
...     iris.data, iris.target, test_size=0.4, random_state=0)

>>> X_train.shape, y_train.shape
((90, 4), (90,))
>>> X_test.shape, y_test.shape
((60, 4), (60,))
```

```
>>> clf = svm.SVC(kernel='linear', C=1).fit(X_train, y_train)
>>> clf.score(X_test, y_test)
```

K-Cross

K//◦ K“”K-1◦

scikit◦ ◦

K-

K“”K-1◦

```
from sklearn.model_selection import KFold
X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
y = np.array([1, 2, 1, 2])
cv = KFold(n_splits=3, random_state=0)

for train_index, test_index in cv.split(X):
...    print("TRAIN:", train_index, "TEST:", test_index)

TRAIN: [2 3] TEST: [0 1]
```

```
TRAIN: [0 1 3] TEST: [2]
TRAIN: [0 1 2] TEST: [3]
```

`StratifiedKFold`k-fold

ShuffleSplit

/。 。

```
from sklearn.model_selection import ShuffleSplit
X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
y = np.array([1, 2, 1, 2])
cv = ShuffleSplit(n_splits=3, test_size=.25, random_state=0)

for train_index, test_index in cv.split(X):
...     print("TRAIN:", train_index, "TEST:", test_index)

TRAIN: [3 1 0] TEST: [2]
TRAIN: [2 1 3] TEST: [0]
TRAIN: [0 2 1] TEST: [3]
```

`StratifiedShuffleSplit`ShuffleSplit。

Leave One / p OutTimeSeriesSplitK-foldscikit model_selection。

https://riptutorial.com/zh-CN/scikit-learn/topic/4901/

6: ROC

Examples

ROCAUC

ROC。

ROCYX。“” - 1。AUC。

ROC“”。

```
import numpy as np
from sklearn import metrics
import matplotlib.pyplot as plt
```

```
y - model.predict(x_test)
```

```
y = np.array([1,1,2,2,3,3,4,4,2,3])
```

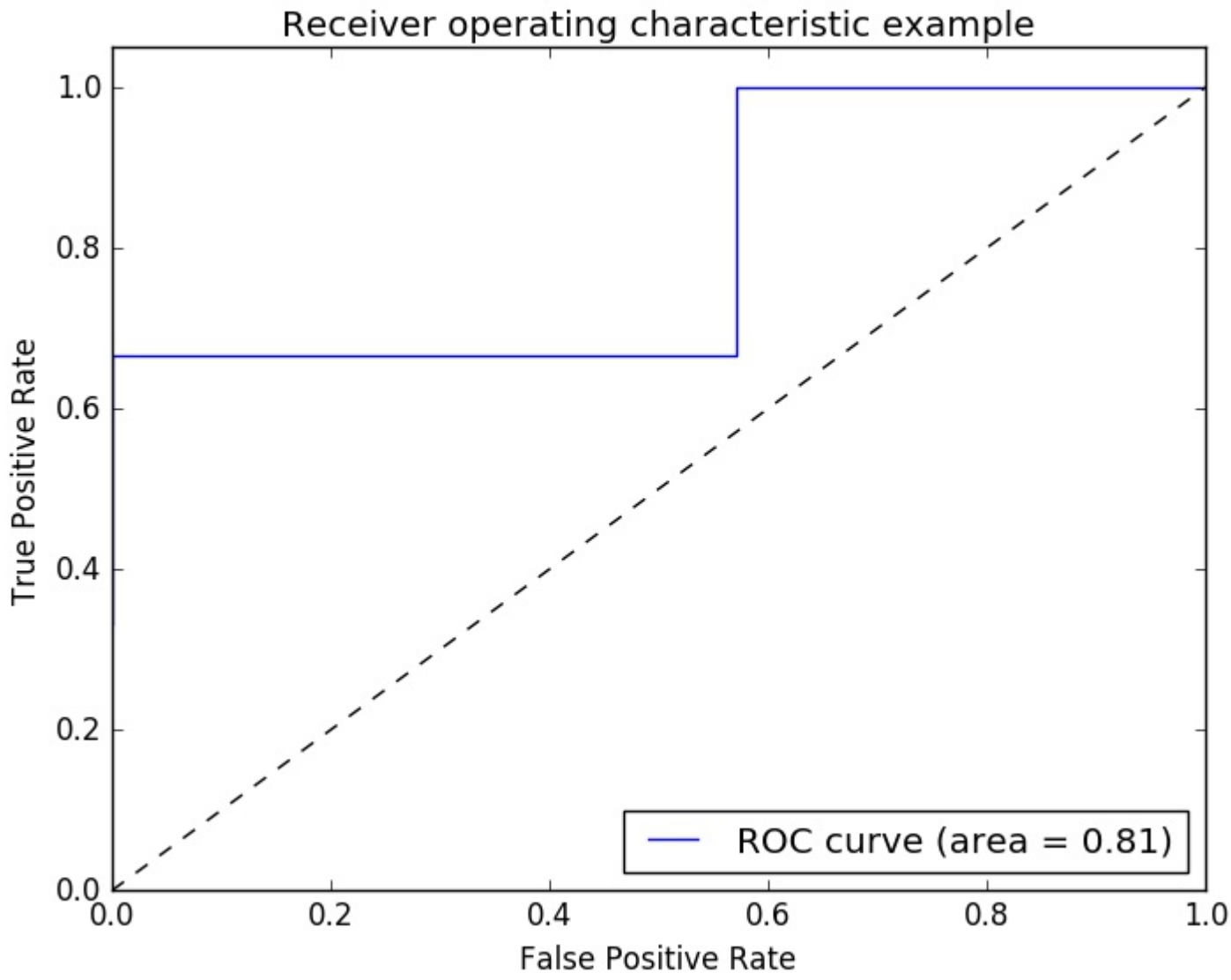
```
model.score(X,Y)
```

```
scores = np.array([0.3, 0.4, 0.95,0.78,0.8,0.64,0.86,0.81,0.9, 0.8])
```

ROCAUC

```
fpr, tpr, thresholds = metrics.roc_curve(y, scores, pos_label=2)
roc_auc = metrics.auc(fpr, tpr)
```

```
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



link1link2

ROC-AUC

ROC-AUC。 `cross_val_predictpredict`。 ROC-AUC`predictpredict_probapredict_proba`。

```
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import cross_val_predict
from sklearn.metrics import roc_auc_score

class LogisticRegressionWrapper(LogisticRegression):
    def predict(self, X):
        return super(LogisticRegressionWrapper, self).predict_proba(X)

X, y = make_classification(n_samples = 1000, n_features=10, n_classes = 2, flip_y = 0.5)

log_reg_clf = LogisticRegressionWrapper(C=0.1, class_weight=None, dual=False,
             fit_intercept=True)

y_hat = cross_val_predict(log_reg_clf, X, y)[:,1]
```

```
print("ROC-AUC score: {}".format(roc_auc_score(y, y_hat)))
```

```
ROC-AUC score: 0.724972396025
```

ROC https://riptutorial.com/zh-CN/scikit-learn/topic/5945/-roc-

7:

Examples

。 。 。

k。 *nmX.kmk*$_{\beta k}$。 *Xβnk*。 *βkmkX*。

scikit-learn sklearn.decomposition.PCAPCA。 100 X 75

```
import numpy as np
np.random.seed(123) # we'll set a random seed so that our results are reproducible
X = np.hstack((np.random.randn(100, 2) + (10, 10), 0.001 * np.random.randn(100, 5)))
```

2

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(X)
```

。

```
pca.components_
# array([[ -2.84271217e-01,  -9.58743893e-01,  -8.25412629e-05,
#           1.96237855e-05,  -1.25862328e-05,   8.27127496e-05,
#          -9.46906600e-05],
#        [ -9.58743890e-01,   2.84271223e-01,  -7.33055823e-05,
#          -1.23188872e-04,  -1.82458739e-05,   5.50383246e-05,
#           1.96503690e-05]])
```

PCA。

PCApca.explained_variance_ratio_

```
pca.explained_variance_ratio_
# array([ 0.57039059,  0.42960728])
```

https://riptutorial.com/zh-CN/scikit-learn/topic/4829/--

S. No		Contributors
1	scikit-learn	Alleo, Ami Tavory, Community, Gabe, Gal Dreiman, panty, Sean Easter, user2314737
2		Ami Tavory, Drew, Gal Dreiman, hashcode55, Mechanic, Raghav RV, Sean Easter, tfv, user6903745, Wayne Werner
3		Ami Tavory, user2314737
4		Ami Tavory, draco_alpine
5		Gal Dreiman, Mechanic
6	ROC	Gal Dreiman, Gorkem Ozkaya
7		Ami Tavory, DataSwede, Gal Dreiman, Sean Easter, user2314737