無料電子ブック

学習

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をいめる

`scikit-learn`は、Pythonでかれたデータのためののオープンソースライブラリです。これはのPythonライブラリにづいていますNumPy、SciPy、matplotlib

`scikit-learn`は、さまざまなアルゴリズムのがまれています。

Examples

scikit-learnのインストール

scikit-learnののにはのものがです

- Python> = 2.6または> = 3.3、
- NumPy> = 1.6.1、
- SciPy> = 0.9。

ほとんどのインストールのは`pip`のpythonパッケージマネージャのpythonとそののすべてをインストールすることができます。

```
pip install scikit-learn
```

しかし、Linuxシステムの、なビルドプロセスをするために`conda`パッケージマネージャをすることがされています

```
conda install scikit-learn
```

`scikit-learn`あることをするには、シェルでします

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

WindowsとMac OSXのインストール

CanopyとAnacondaは、WindowsのなPythonライブラリ、Mac OSXLinuxにもしているのほかに、の*scikit-learn*のバージョンをしています。

クロスバリデーションをしてクラシファイアをトレーニングする

アイリスデータセットの

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

データはとテストセットにされます。これをうには、 `train_test_split`ユーティリティをして、 `train_test_split` `train_size=0.75`オプショントレーニングセットにはデータの75がまれていますをして`X`と`y` データとターゲットベクトルをランダムにします。

トレーニングデータセットは、 kのにされる 。クラシファイアのメソッド`fit`は、モデルをデータにさせます。

```
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)
```

に、テストサンプルのをする

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

1のトレインとテストセットをすることによって、データスプリットののによるののりをることができる。 クロスバリデーションをすることにより、データのなる/サブセットにをさせることができ、すべてのにしてをとることができます。 `cross_val_score` 、をしてデータに`cross_val_score`させます。するフォールドをすることができますのでは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)
```

パイプラインの

データのパターンをつけることは、くの、、、などののデータステップでします。 `sklearn`では、これのためにステージのパイプラインがされています。

たとえば、のコードは2つのステージからなるパイプラインをしています。のものはフィーチャをスケールし、2のフィーチャはのデータセットにをします。

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

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

パイプラインがされると、そののにじて、のステージのようにできます。ここでは、えば、パイプラインはクラシファイアのようにるいます。したがって、のようにできます。

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

```
pipeline.predict_proba(X_test)
```

インタフェースと

なクラスをして、データのがなります。

クラスのほとんどは、のグループのいずれかにします。

- のをするためのアルゴリズム `sklearn.base.ClassifierMixin`から`sklearn.base.ClassifierMixin`
- をするをするためのアルゴリズム `sklearn.base.RegressorMixin`からした `sklearn.base.RegressorMixin`
- データをするデータ `sklearn.base.TransformerMixin`から`sklearn.base.TransformerMixin`

データはにされ`numpy.array` Sしかしのようなのアレイオブジェクト`pandas.DataFrame`ものにであるにsがけれられる`numpy.array`

データのオブジェクトは、のによってされます。なりめでは、データサンプルはでされ、のはデータサンプルID、2のはフィーチャIDです。

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

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

の`sklearn`では、2つのフィーチャでそれぞれされる5つのオブジェクトがまれています。

サンプルデータセット

テストをにするために、 `sklearn`は`sklearn.datasets`モジュールにいくつかのビルトインデータセットを`sklearn.datasets`ます。たとえば、Fisherのデータセットをみみます。

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

な、の、およびクラスの `target_names` をむことができます。それらはとしてされます。

たちは、 `data`と`target`フィールドにされているデータとクラスにがあります。により、それらは`X`および`y`

```
X, y = iris_dataset['data'], iris_dataset['target']
X.shape, y.shape
```

```
((150, 4), (150,))
```

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

`X`と`y`は、4つのをつ150のサンプルがあるとう。サンプルは、0,1または2のいずれかのクラスにします。

`X`と`y`は、の`fit()`メソッドをびすことで、のにできるようになりました。

`sklearn.datasets`モジュールでされるデータセットのサイズとのなリストはの`sklearn.datasets`です。

		サイズ	
`load_boston()`	ボストンのデータセット	506	
`load_breast_cancer()`	がんウィスコンシンのデータセット	569	バイナリ
`load_diabetes()`	のデータセット	442	
`load_digits(n_class)`	データセット	1797	
`load_iris()`	アイリスデータセット	150	マルチクラス
`load_linnerud()`	Linnerudデータセット	20	

してください ソース http : //scikit-learn.org/stable/datasets/

> これらのデータセットは、scikitでされたさまざまなアルゴリズムのをすばやくするのにです。しかし、のタスクをするにはさすぎることがあります。

これらのみみのサンプルデータセットに`sklearn.datasets`、 `sklearn.datasets`はデータセットをロードするユーティリティもします

- `load_mlcomp`リポジトリからサンプルデータセットをみむためのload_mlcomp データセットをにダウンロードするがあることにしてください。 ここではです。
- フェイスにされるhttp://vis-www.cs.umass.edu/lfw/のワイルドLFWペアデータセットのLabeled Facesをみむための`fetch_lfw_pairs`および`fetch_lfw_people`。 このデータセットは200 MBをえています。 ここではです。

オンラインでscikit-learnをいめるをむ https://riptutorial.com/ja/scikit-learn/topic/1035/scikit-learnをいめる

2: モデル

Examples

のパラメータをしてじデータでテストするのはなりです。たサンプルのラベルをなスコアでりしますが、えないデータ。このをオーバーフィットといいます 。それをけるために、なデータのをテストセット `X_test, y_test` としてするされたをするのが `X_test, y_test` 。 「」というは、なでさえ、がにまるため、なのみをするものではないことにしてください。

scikit-learnでは、 training_test_splitヘルパーをして、トレーニングセットとテストセットのランダムなをにできます。データセットをサポートベクトルマシンにわせてロードしましょう

```
>>> 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,))
```

をテストするためにデータの40をしながら、すぐにトレーニングセットをサンプリングすることができます。

```
>>> 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,))
```

さて、たちがとテストセットをっていれば、それをうことができます

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

K-Foldクロス

Kクロスバリデーションは、/テストスプリットののにするをするために、/テストスプリットをりすためのなプロセスです。には、データセットをKのしいサイズの「りみ」にし、りみはモデルをテストするために1、モデルをトレーニングするためにK-1されます。

Scikitライブラリではのりたたみができます。それらのは、データにします。いくつかのがあります

Kフォールド

には、データセットをKのしいサイズの「りみ」にし、りみはモデルをテストするために1、モデルをトレーニングするためにK-1されます。

```
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`は、りみをすkのバリエーションです。セットは、なセットとターゲットクラスのサンプルのほぼじパーセンテージをみます

シャッフルスプリット

/データセットのユーザをするためにされます。サンプルをにシャッフルしてから、トレインとテストセットのペアにします。

```
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`はShuffleSplitのバリエーションです。されたをします。つまり、なセットとじターゲットクラスごとにじパーセンテージをすることでをします。

Leave One / p Out、TimeSeriesSplitKのバリエーションなどののりたたみは、scikit model_selectionライブラリでできます。

オンラインでモデルをむ https://riptutorial.com/ja/scikit-learn/topic/4901/モデル

3: レシーバROC

Examples

ROCとAUCの

のをするReceiver Operating CharacteristicROCメトリックの

ROCは、、Yでを、Xでをとする。これは、プロットのが「な」であることをします。これは、0、1です。これはあまりではありませんが、AUCがはよりいことをします。

のをにし、をにすることがであるため、ROCの「さ」もです。

な

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

の$_y$ - には、これはされたです `model.predict(x_test)`

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

スコアは、えられたテストデータとラベルのです `model.score(X,Y)` 。

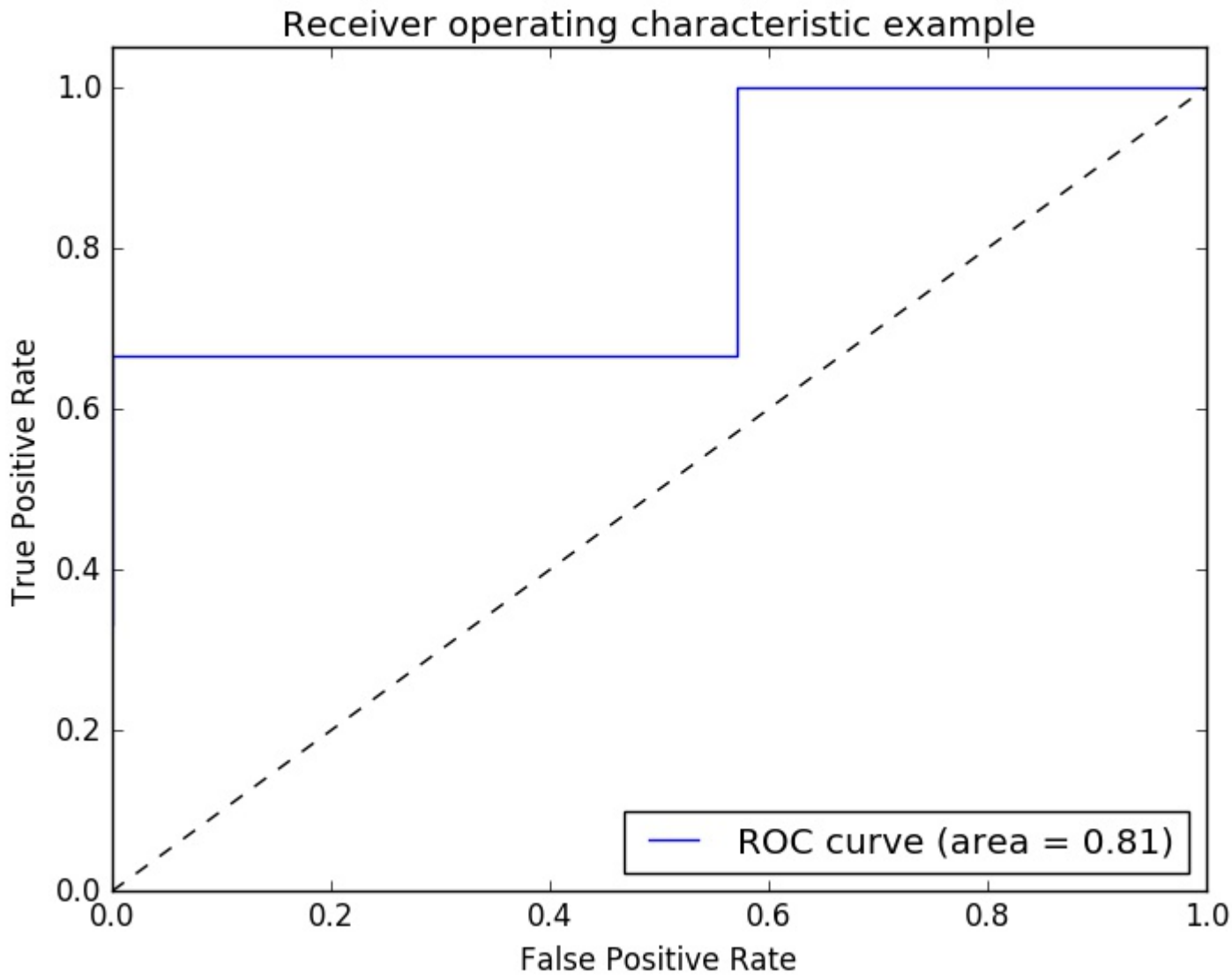
```
scores = np.array([0.3, 0.4, 0.95,0.78,0.8,0.64,0.86,0.81,0.9, 0.8])
```

ROCおよびAUCをする

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

プロット

```
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()
```



ソースは、これらからしたリンク1とリンク2

オーバーライドおよびクロスによるROC-AUCスコア

ROC-AUCスコアをするためにされるがである。 `cross_val_predict`は、の`predict`メソッドをします。 ROC-AUCスコアをすることができるようにするために、つはにきして、をサブクラスすることができます`predict` 、それはのようにするように、を`predict_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
```

オンラインでレシーバROCをむ https://riptutorial.com/ja/scikit-learn/topic/5945/レシーバ-roc-

4:

Examples

サポートベクターマシンの

サポートベクターマシンは、からのがあるでであるように、2つのラベルけされたのでによってはのをそうとするアルゴリズムのファミリです。 SVMはまたはにできますそれぞれsklearn.svm.SVCおよびsklearn.svm.SVRに。

2Dでするとします。まず、いくつかのデータをします。

```
import numpy as np
```

はxとyをします

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

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

xは2つのガウスでされています 1つは0、0をに、もう1つは1,1をにしています 。

クラシファイアをするには、をできます。

```
from sklearn import svm

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

0、0のをしましょう

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

はクラスが0であるということです。

については、にうことができます

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

RandomForestClassifier

ランダムフォレストは、データセットのさまざまなサブサンプルにのデシジョンツリーをさせ、をしてとオーバーフィットをするメタです。

な

インポート

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

データとデータをする

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

`target`のは、したいラベルをします。

RandomForestオブジェクトをし、フィットをします。

```
rf = RandomForestClassifier(n_estimators=100)
rf.fit(train, target)
```

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

レポートの

とリコール 、 f1スコア とリコールのハーモニック 、サポートトレーニングセットのそのクラスのをむなメトリックをすテキストレポートをします。

`sklearn` docsの

```
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

のためのグラジエントブースト 。 Gradient Boosting Classifierは、ののをするのによって、がまたはでされるモデルのです。

インポート

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

おもちゃのデ―タをする

```
from sklearn.datasets import load_iris

iris_dataset = load_iris()

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

このデ―タをトレ―ニングとテストのセットにしましょう。

```
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)
```

デフォルトのパラメ―タをして**GradientBoostingClassifier**モデルをインスタンスします。

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

たちはテストセットでそれをましょう

```
# We are using the default classification accuracy score
>>> gbc.score(X_test, y_test)
1
```

デフォルトでは、100のもりがされています

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

これは、に`n_estimators`をのにすることでできます。

ツリ―

デシジョンツリ―は、> 7のようなルールのシ―ケンスをしてにできるクラシファイアです。

のは、さ3の3つのベクトルをしてツリ―をし、に、の4のベクトル、いわゆるテストベクトルのをする。

```
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
```

はをしてできます。

```
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")
```

ロジスティックをいた

LRでは、ののなをするは、ロジスティックをしてモデルされる。 `linear_model`ライブラリにされて`linear_model`ます

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

Sklearn LRのは、バイナリ、One-vs-Rest、ロジスティック、オプションのL2またはL1をたすことができます。たとえば、サンプルsklearnデータセットのバイナリをえてみましょう

```
from sklearn.datasets import make_hastie_10_2

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

Xは`n_samples X 10`、yはターゲットラベル-1または+1です。

テストをして、データをトレーニングセットとテストセットにします7030

```
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クラシファイアのはのとです

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

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

マトリックスをしてをする

```
from sklearn.metrics import confusion_matrix

confusion_matrix(predicted, labels_test)
```

オンラインでをむ https://riptutorial.com/ja/scikit-learn/topic/2468/

5:

Examples

の

のは、のでにもよくするフィーチャのをつけるです。

されるのベクトルがyであり、がXをする、OLSはベクトルβをめる

$min_\beta \| y\hat{} - y \|_2^2$、

ここで、 $y\hat{} = X\beta$はである。

sklearnでは、これは`sklearn.linear_model.LinearRegression`をってわれます。

アプリケーションコンテキスト

OLSはにのみされるべきであり、ににはであるコントラスト

- メールはスパムですか
- upvotesのは、のさにしますか

のノイズをむモデルをし、 `LinearRegression`がモデルをするかどうかをてみましょう。

まず、 `X`をします。

```
import numpy as np

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

、たちは、します`y`のとして`X`いくつかのノイズをちます

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

`y`するのは、βによってえられることにされたい。

`X`と`y`だけからこれをしようとするには、のようにしましょう

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

このベクトルは`beta`ににています。

オンラインでをむ https://riptutorial.com/ja/scikit-learn/topic/5190/

6: の

Examples

の

これはになフィ―チャです。

そのなえは、フィ―チャがであるすなわち、が0である、いパタ―ンをつけるためにすることができず、デ―タセットからすることができるということです。

したがって、フィ―チャにするヒュ―リスティックなアプロ―チは、があるいをるすべてのフィ―チャをにすることである。

ドキュメントのをまえて、まずは

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

ここには6 つのブ―ルがあり、それぞれ6つのインスタンスがあります。インスタンスのなくとも80でであるインスタンスをしたいとします。いくつかのは、これらのが*0.8 *1 - 0.8*よりさいをするがあることをしている。その、

```
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]])
```

のがどのようにされたかにしてください。

このはにするがあります。なぜなら、がいということはずしもが「くない」というではないからです。のでは、 3つのフィ―チャをむデ―タセットをします。の2つは、ランダムにすると3つののでされています。

```
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 ]])
```

では、1のはがいためにされていますが、3のそれはもいがされています。この、スケーリングとはであるため、をすることがよりであろう。

オンラインでのをむ https://riptutorial.com/ja/scikit-learn/topic/4909/の

7: のの

Examples

コンポーネントによるディメンションの

は、ののシーケンスをつける。のは、フィーチャのをにしますのをけます。ののゞは、のにまたがるとするにおけるのをにする。

なは、 kのそのようなのみをすることである。がnmのXであるとする。のkのは、m kのβkをします。Xβは、 nとkとをする。したがって、のβkは、のXのをして、 mからkへのとえることができる。

scikit-learnでは、PCAはsklearn.decomposition.PCAでされsklearn.decomposition.PCA。 たとえば、がの2つのにのみまれるようにされた100×7のからめるとしますの5つのをする。

```
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にしましょう

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

さて、をしましょう。まず、があります

```
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]])
```

ベクトルのの2つのがのよりもきいことにしてください.PCAはがにの2つのカラムにまれていることをしています。

このPCAによってされるのをべるために、 pca.explained_variance_ratio_をべることができ

pca.explained_variance_ratio_。

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

オンラインでののをむ https://riptutorial.com/ja/scikit-learn/topic/4829/の-の-

クレジット

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