机器学习

实验一：决策树、随机森林、线性模型

报告提交时间：下下周五(5.23)晚上12点前

提交方式：将压缩包发送至3120245649@bit.edu.cn

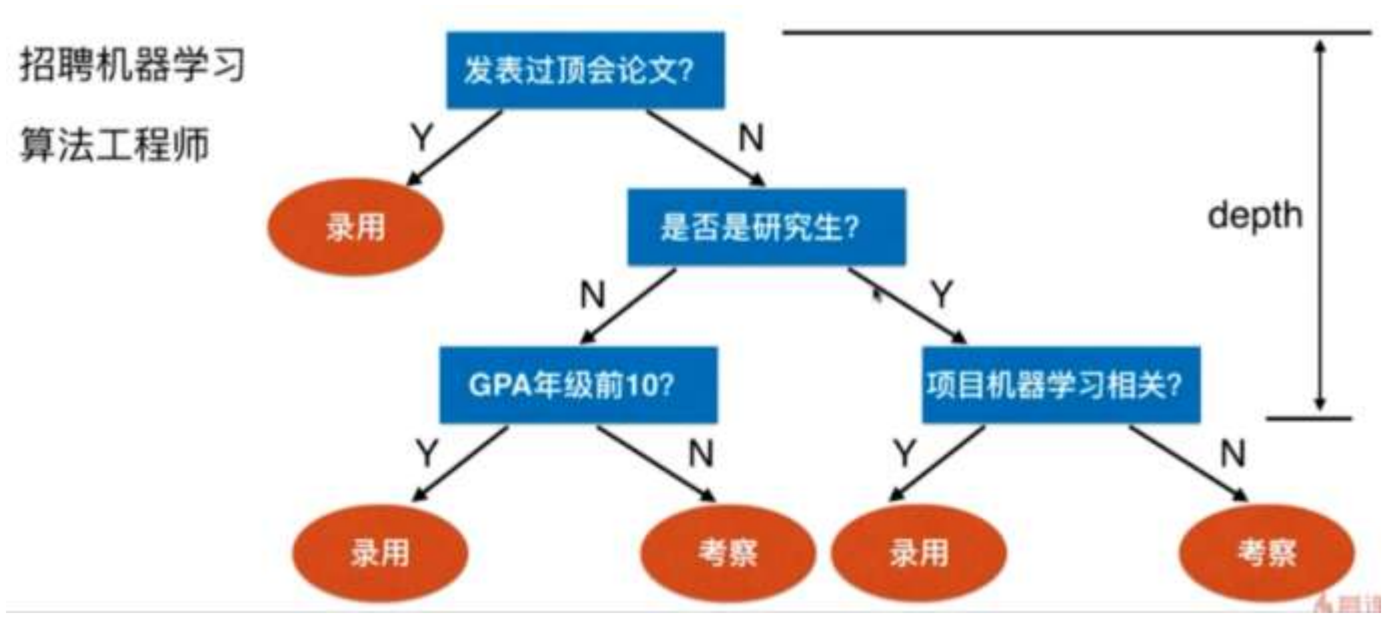
评分：

- 总分100分
- 任务一、任务二各50分
- 每个任务基础分30分，加分项20分

实验任务一（决策树、随机森林）：

- 1.使用决策树模型完成对红酒数据集(datasets.load_wine())和糖尿病数据集(datasets.load_diabetes())的分类、回归任务
- 2.使用随机森林模型完成对红酒数据集和糖尿病数据集的分类、回归任务

决策树



- 一组规则
- split策略
- criterion
- 剪枝

实验环境：

- Python 3.0以上版本，开发工具任选
- 使用scikit-learn包中的机器学习模型
- 安装说明：https://scikit-learn.org/stable/install.html#installation- instructions

实验环境：

- Anaconda(https://www.anaconda.com/distribution/)
- VSCode

Anaconda+VSCode（+jupyter）安装步骤参考：
https://www.jianshu.com/p/ef1ae10ba950

实验要求和评分细则

- 使用决策树模型完成在红酒数据集和糖尿病数据集上的分类任务和回归任务（15分）
- 使用随机森林模型完成在红酒数据集和糖尿病数据集上的分类任务和回归任务（15分）

实验要求和评分标准

加分项：

- 将结果可视化（10分）提示：可视化时选择2个特征即可
- 对模型参数进行优化（5分）
- 每类实验中包含2个以上数据集，并将不同数据集上的结果加以对比分析（5分）

作业提交内容：

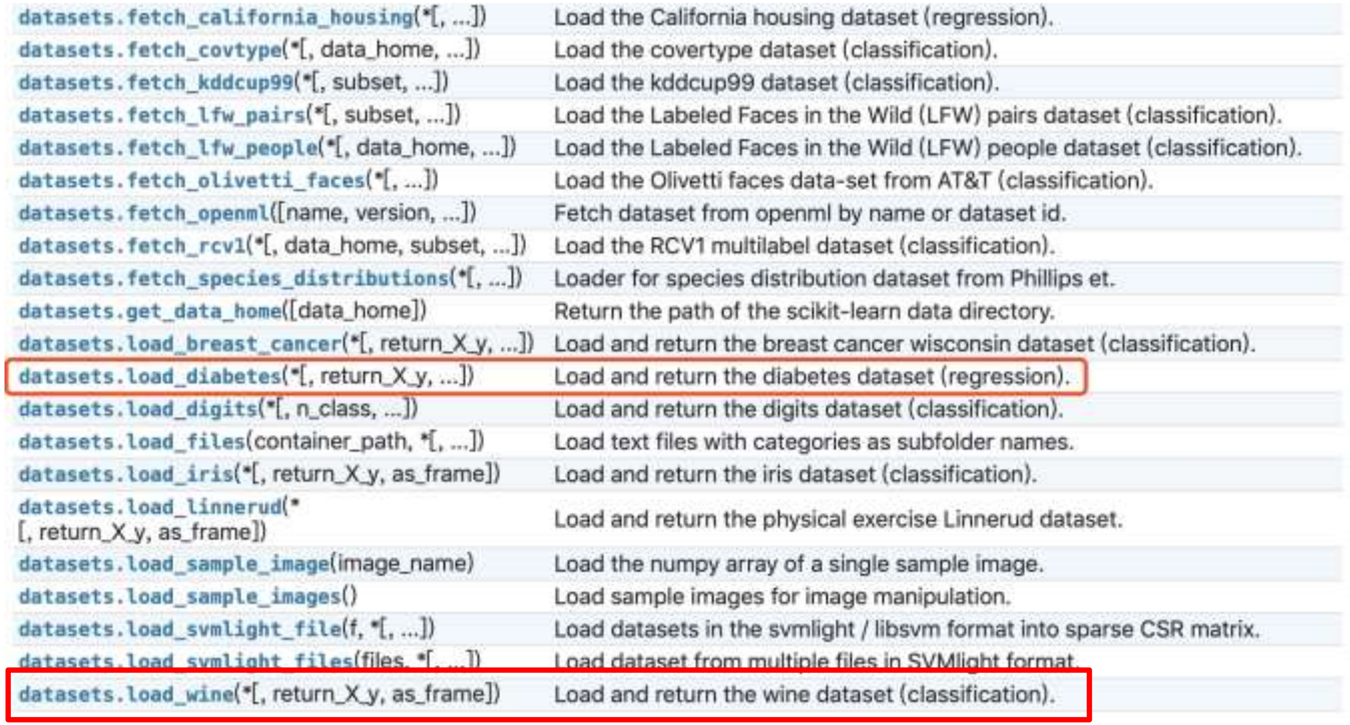
- 所有代码文件
- 实验report，包含每个模型运行后的结果截图（word文档的命名规则为：姓名-学号-实验1-任务1.docx）

实验步骤

https://scikit-learn.org/

- 加载数据集
- 拆分数据集
- 构建模型
- 获取在训练集中的模型
- 在测试集上预测结果
- 模型评测

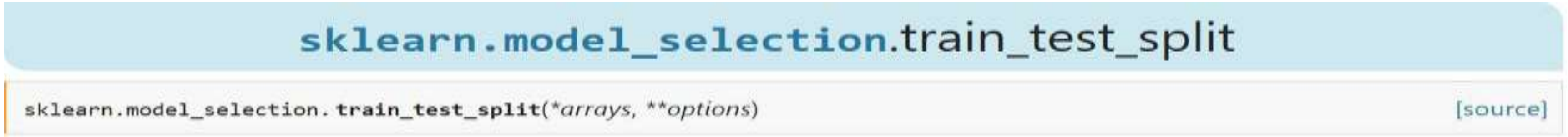
数据集



datasets.fetch_california_housing(*[, ...])	Load the California housing dataset (regression).
datasets.fetch_covtype(*[, data_home, ...])	Load the covertype dataset (classification).
datasets.fetch_kddcup99(*[, subset, ...])	Load the kddcup99 dataset (classification).
datasets.fetch_lfw_pairs(*[, subset, ...])	Load the Labeled Faces in the Wild (LFW) pairs dataset (classification).
datasets.fetch_lfw_people(*[, data_home, ...])	Load the Labeled Faces in the Wild (LFW) people dataset (classification).
datasets.fetch_olivetti_faces(*[, ...])	Load the Olivetti faces data-set from AT&T (classification).
datasets.fetch_openml([name, version, ...])	Fetch dataset from openml by name or dataset id.
datasets.fetch_rcv1(*[, data_home, subset, ...])	Load the RCV1 multilabel dataset (classification).
datasets.fetch_species_distributions(*[, ...])	Loader for species distribution dataset from Phillips et.
datasets.get_data_home([data_home])	Return the path of the scikit-learn data directory.
datasets.load_breast_cancer(*[, return_X_y, ...])	Load and return the breast cancer wisconsin dataset (classification).
datasets.load_diabetes(*[, return_X_y, ...])	Load and return the diabetes dataset (regression).
datasets.load_digits(*[, n_class, ...])	Load and return the digits dataset (classification).
datasets.load_files(container_path, *[, ...])	Load text files with categories as subfolder names.
datasets.load_iris(*[, return_X_y, as_frame])	Load and return the iris dataset (classification).
datasets.load_linnerud(*[, return_X_y, as_frame])	Load and return the physical exercise Linnerud dataset.
datasets.load_sample_image(image_name)	Load the numpy array of a single sample image.
datasets.load_sample_images()	Load sample images for image manipulation.
datasets.load_svmlight_file(f, *[, ...])	Load datasets in the svmlight / libsvm format into sparse CSR matrix.
datasets.load_svmlight_files(files, *[, ...])	Load dataset from multiple files in SVMlight format.
datasets.load_wine(*[, return_X_y, as_frame])	Load and return the wine dataset (classification).

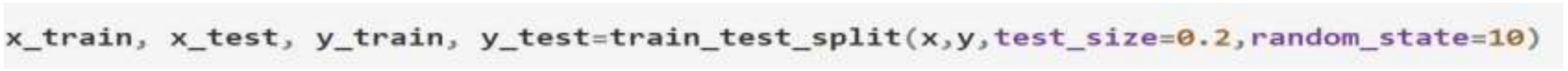
https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets

拆分数据集



sklearn.model_selection.train_test_split

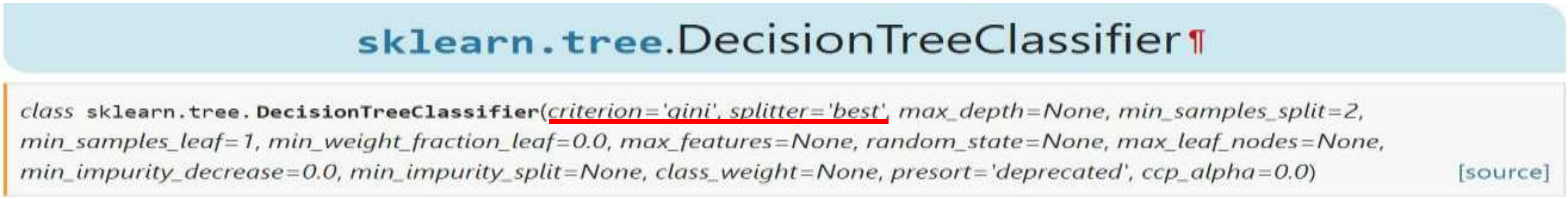
```
sklearn.model_selection.train_test_split(*arrays, **options)
```



```
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.2,random_state=10)
```

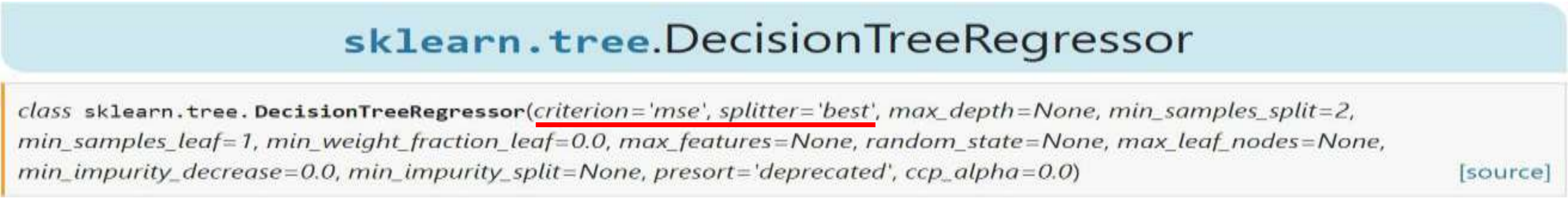
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split

决策树模型



sklearn.tree.DecisionTreeClassifier¶

```
class sklearn.tree.DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort='deprecated', ccp_alpha=0.0)
```
[source]

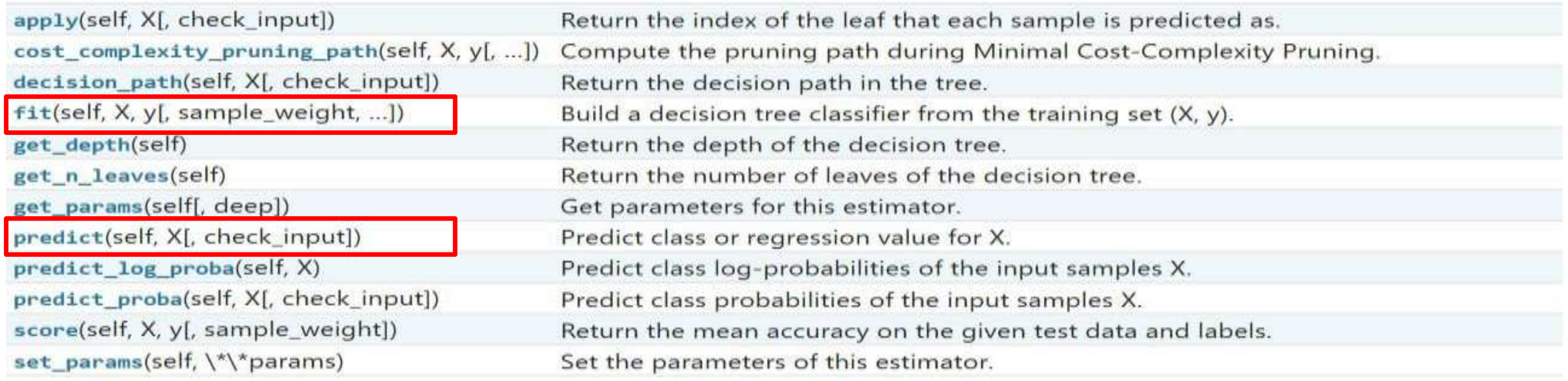


sklearn.tree.DecisionTreeRegressor

```
class sklearn.tree.DecisionTreeRegressor(criterion='mse', splitter='best', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None, presort='deprecated', ccp_alpha=0.0)
```
[source]

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.tree

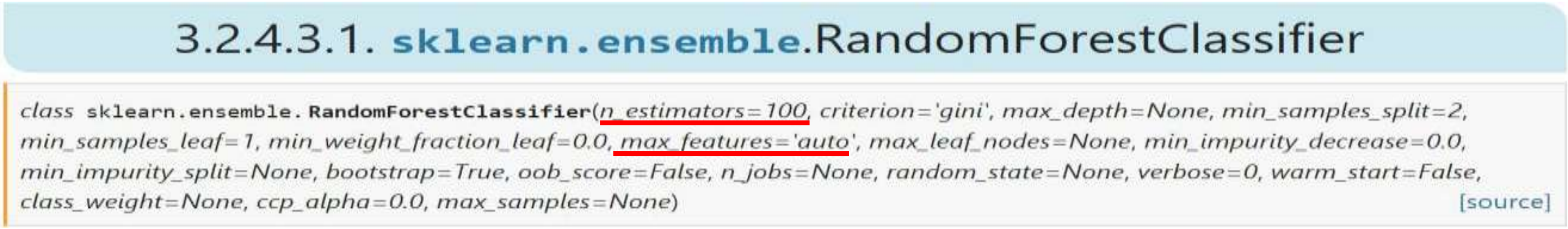
决策树模型



apply(self, X[, check_input])	Return the index of the leaf that each sample is predicted as.
cost_complexity_pruning_path(self, X, y[, ...])	Compute the pruning path during Minimal Cost-Complexity Pruning.
decision_path(self, X[, check_input])	Return the decision path in the tree.
fit(self, X, y[, sample_weight, ...])	Build a decision tree classifier from the training set (X, y).
get_depth(self)	Return the depth of the decision tree.
get_n_leaves(self)	Return the number of leaves of the decision tree.
get_params(self[, deep])	Get parameters for this estimator.
predict(self, X[, check_input])	Predict class or regression value for X.
predict_log_proba(self, X)	Predict class log-probabilities of the input samples X.
predict_proba(self, X[, check_input])	Predict class probabilities of the input samples X.
score(self, X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
set_params(self, **params)	Set the parameters of this estimator.

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.tree

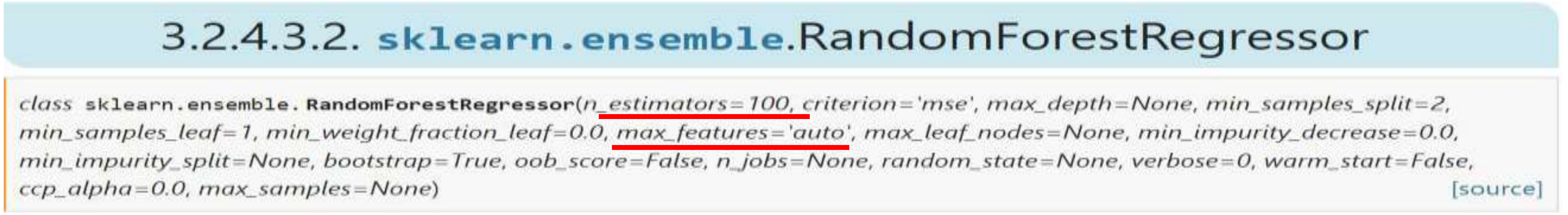
随机森林



3.2.4.3.1. sklearn.ensemble.RandomForestClassifier

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=100, criterion='gini', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0,
min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False,
class_weight=None, ccp_alpha=0.0, max_samples=None)                                                          [source]
```

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier

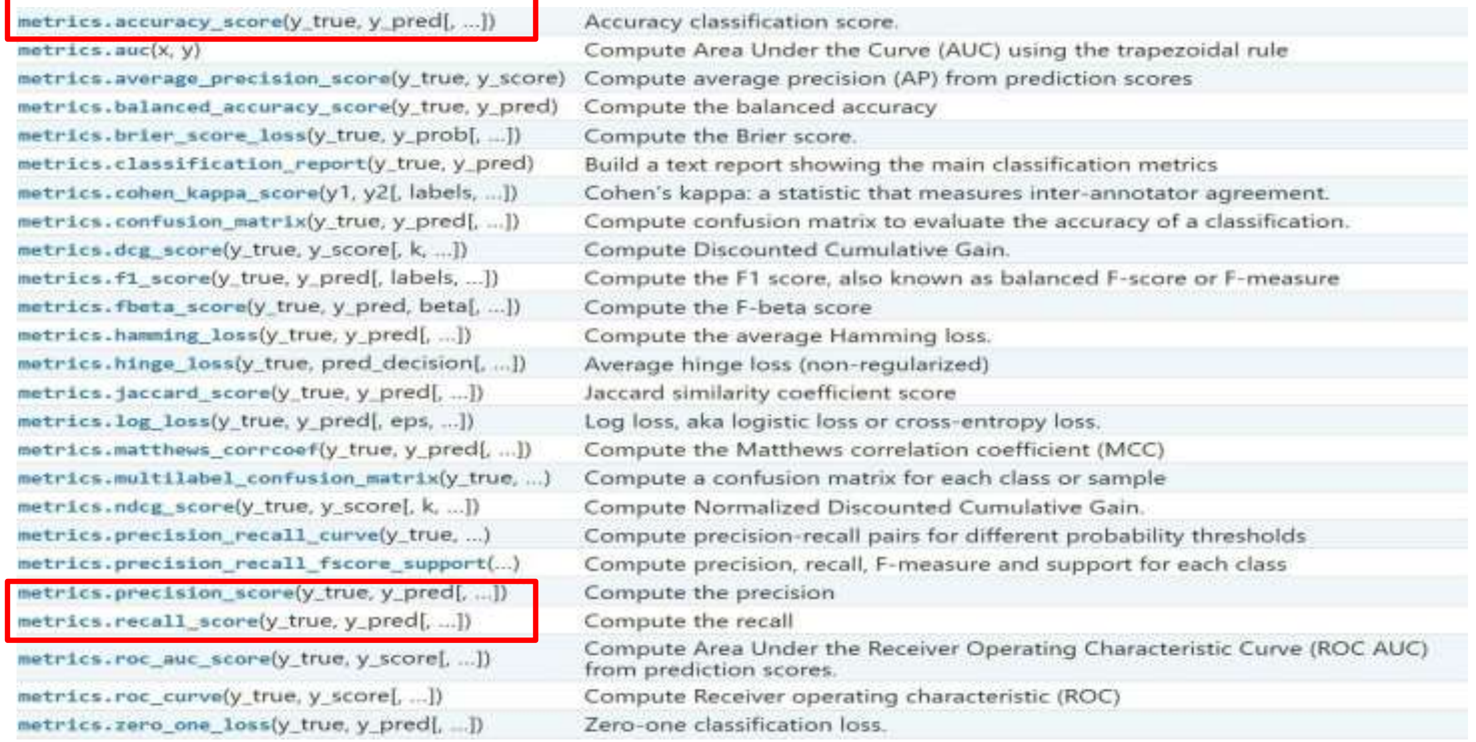


3.2.4.3.2. sklearn.ensemble.RandomForestRegressor

```
class sklearn.ensemble.RandomForestRegressor(n_estimators=100, criterion='mse', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0,
min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False,
ccp_alpha=0.0, max_samples=None)                                                                             [source]
```

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#sklearn.ensemble.RandomForestRegressor

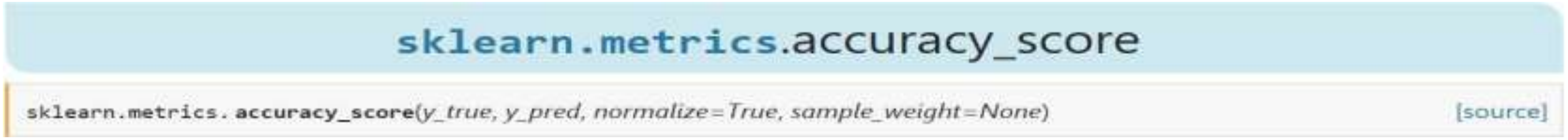
模型评测 Classification metrics



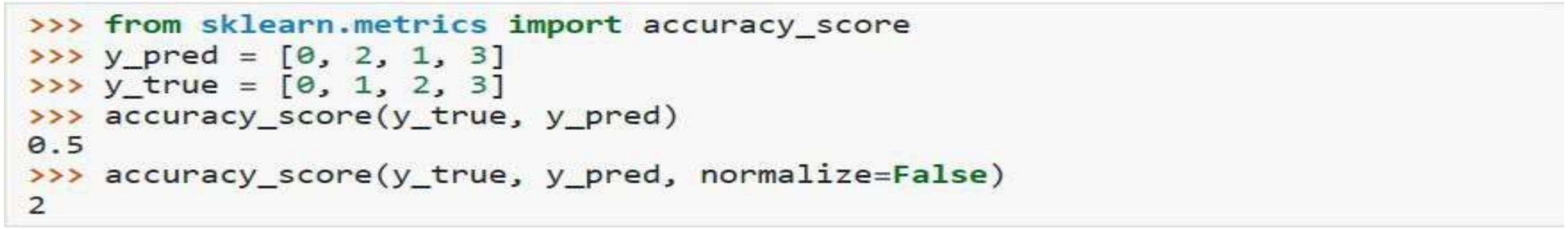
metrics.accuracy_score(y_true, y_pred[, ...])	Accuracy classification score.
metrics.auc(x, y)	Compute Area Under the Curve (AUC) using the trapezoidal rule
metrics.average_precision_score(y_true, y_score)	Compute average precision (AP) from prediction scores
metrics.balanced_accuracy_score(y_true, y_pred)	Compute the balanced accuracy
metrics.brier_score_loss(y_true, y_prob[, ...])	Compute the Brier score.
metrics.classification_report(y_true, y_pred)	Build a text report showing the main classification metrics
metrics.cohen_kappa_score(y1, y2[, labels, ...])	Cohen's kappa: a statistic that measures inter-annotator agreement.
metrics.confusion_matrix(y_true, y_pred[, ...])	Compute confusion matrix to evaluate the accuracy of a classification.
metrics.dcg_score(y_true, y_score[, k, ...])	Compute Discounted Cumulative Gain.
metrics.f1_score(y_true, y_pred[, labels, ...])	Compute the F1 score, also known as balanced F-score or F-measure
metrics.fbeta_score(y_true, y_pred, beta[, ...])	Compute the F-beta score
metrics.hamming_loss(y_true, y_pred[, ...])	Compute the average Hamming loss.
metrics.hinge_loss(y_true, pred_decision[, ...])	Average hinge loss (non-regularized)
metrics.jaccard_score(y_true, y_pred[, ...])	Jaccard similarity coefficient score
metrics.log_loss(y_true, y_pred[, eps, ...])	Log loss, aka logistic loss or cross-entropy loss.
metrics.matthews_corrcoef(y_true, y_pred[, ...])	Compute the Matthews correlation coefficient (MCC)
metrics.multilabel_confusion_matrix(y_true, ...)	Compute a confusion matrix for each class or sample
metrics.ndcg_score(y_true, y_score[, k, ...])	Compute Normalized Discounted Cumulative Gain.
metrics.precision_recall_curve(y_true, ...)	Compute precision-recall pairs for different probability thresholds
metrics.precision_recall_fscore_support(...)	Compute precision, recall, F-measure and support for each class
metrics.precision_score(y_true, y_pred[, ...])	Compute the precision
metrics.recall_score(y_true, y_pred[, ...])	Compute the recall
metrics.roc_auc_score(y_true, y_score[, ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
metrics.roc_curve(y_true, y_score[, ...])	Compute Receiver operating characteristic (ROC)
metrics.zero_one_loss(y_true, y_pred[, ...])	Zero-one classification loss.

https://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics

模型评测

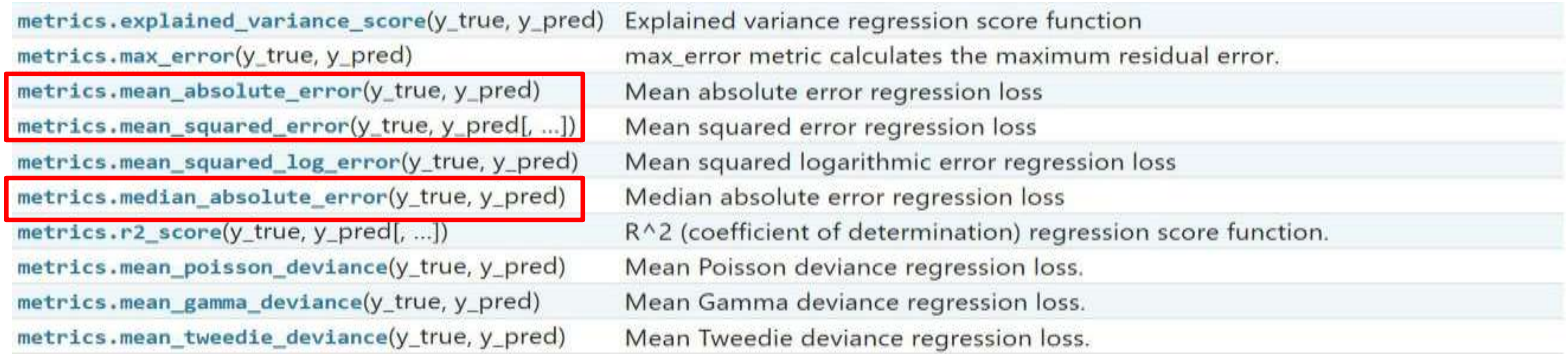

sklearn.metrics.accuracy_score

sklearn.metrics.accuracy_score(*y_true*, *y_pred*, *normalize=True*, *sample_weight=None*) [source]


```
>>> from sklearn.metrics import accuracy_score
>>> y_pred = [0, 2, 1, 3]
>>> y_true = [0, 1, 2, 3]
>>> accuracy_score(y_true, y_pred)
0.5
>>> accuracy_score(y_true, y_pred, normalize=False)
2
```

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html#sklearn.metrics.accuracy_score

模型评测 Regression metrics



`metrics.explained_variance_score`(y_true, y_pred)	Explained variance regression score function
`metrics.max_error`(y_true, y_pred)	max_error metric calculates the maximum residual error.
`metrics.mean_absolute_error`(y_true, y_pred)	Mean absolute error regression loss
`metrics.mean_squared_error`(y_true, y_pred[, ...])	Mean squared error regression loss
`metrics.mean_squared_log_error`(y_true, y_pred)	Mean squared logarithmic error regression loss
`metrics.median_absolute_error`(y_true, y_pred)	Median absolute error regression loss
`metrics.r2_score`(y_true, y_pred[, ...])	R^2 (coefficient of determination) regression score function.
`metrics.mean_poisson_deviance`(y_true, y_pred)	Mean Poisson deviance regression loss.
`metrics.mean_gamma_deviance`(y_true, y_pred)	Mean Gamma deviance regression loss.
`metrics.mean_tweedie_deviance`(y_true, y_pred)	Mean Tweedie deviance regression loss.

https://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics

机器学习

线性模型

实验任务二：

- 1.使用逻辑回归模型（LogisticRegression）完成对乳腺癌数据集(dataset.load_breast_cancer())的分类任务
- 2.使用线性回归模型（LinearRegression）完成对加州房价数据集(dataset.fetch_california_housing())的回归任务
- 3.自定义实现ridge 回归并与sklearn实现做比较

实验要求和评分细则

- 使用逻辑回归模型完成在乳腺癌数据集上的分类任务（10分）
- 使用线性回归模型完成在加州房价数据集的回归任务（10分）
- 自定义ridge回归完成加州房价数据集回归任务（10分）

实验要求和评分标准

加分项：

- 将sklearn结果可视化并对模型进行参数优化（5分）
- 将自定义ridge回归结果可视化并进行参数优化（10分）
- sklearn实现的实验中包含2个以上数据集，并将不同数据集上的结果加以对比分析（5分）

任务二提交内容：

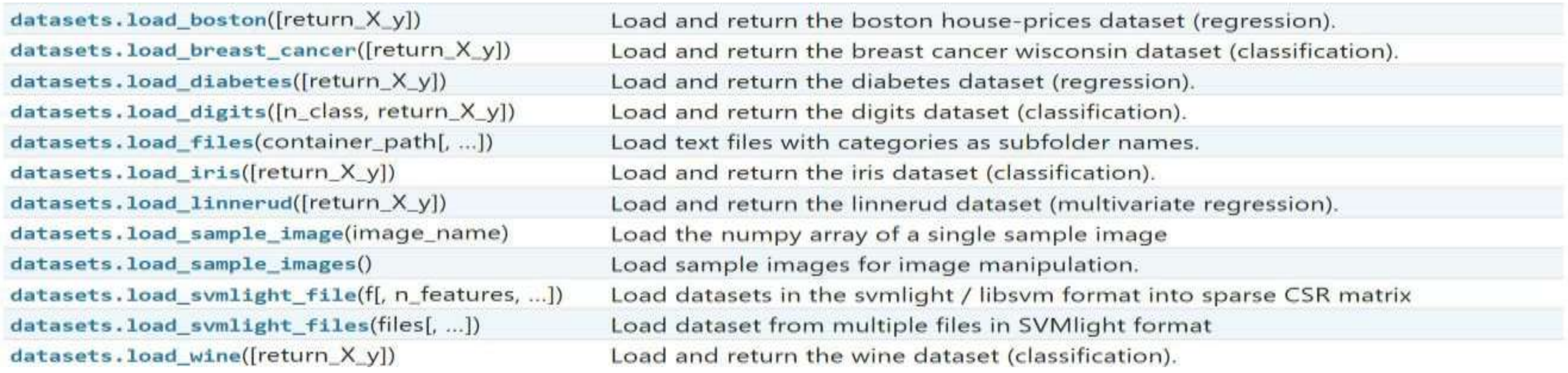
- 所有代码文件
- 每个模型运行后的结果截图（保存到word中，word文档的命名规则为：姓名-学号-实验1-任务2. docx）

实验步骤

https://scikit-learn.org/

- 加载数据集
- 拆分数据集
- 构建模型
- 获取在训练集中的模型
- 在测试集上预测结果
- 模型评测

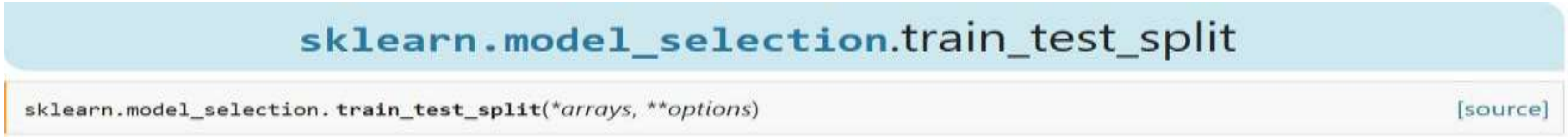
数据集



datasets.load_boston([return_X_y])	Load and return the boston house-prices dataset (regression).
datasets.load_breast_cancer([return_X_y])	Load and return the breast cancer wisconsin dataset (classification).
datasets.load_diabetes([return_X_y])	Load and return the diabetes dataset (regression).
datasets.load_digits([n_class, return_X_y])	Load and return the digits dataset (classification).
datasets.load_files(container_path[, ...])	Load text files with categories as subfolder names.
datasets.load_iris([return_X_y])	Load and return the iris dataset (classification).
datasets.load_linnerud([return_X_y])	Load and return the linnerud dataset (multivariate regression).
datasets.load_sample_image(image_name)	Load the numpy array of a single sample image
datasets.load_sample_images()	Load sample images for image manipulation.
datasets.load_svmlight_file(f[, n_features, ...])	Load datasets in the svmlight / libsvm format into sparse CSR matrix
datasets.load_svmlight_files(files[, ...])	Load dataset from multiple files in SVMlight format
datasets.load_wine([return_X_y])	Load and return the wine dataset (classification).

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets

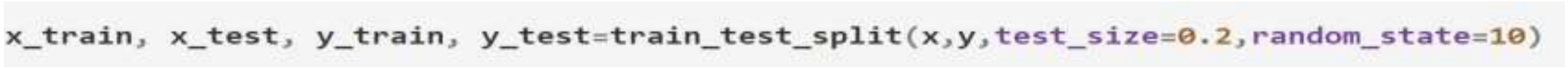
拆分数据集



sklearn.model_selection.train_test_split

```
sklearn.model_selection.train_test_split(*arrays, **options)
```

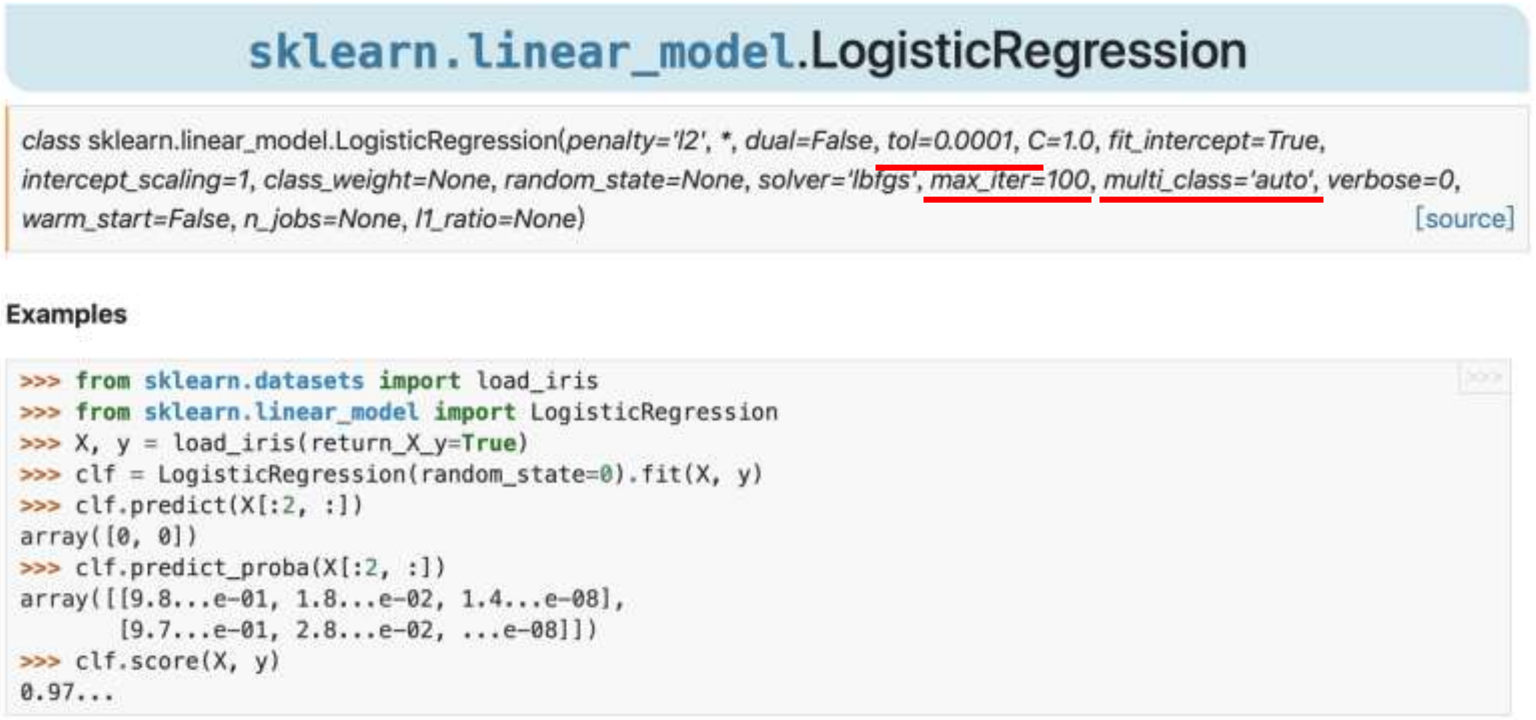
[source]



```
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.2,random_state=10)
```

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split

逻辑回归模型



sklearn.linear_model.LogisticRegression

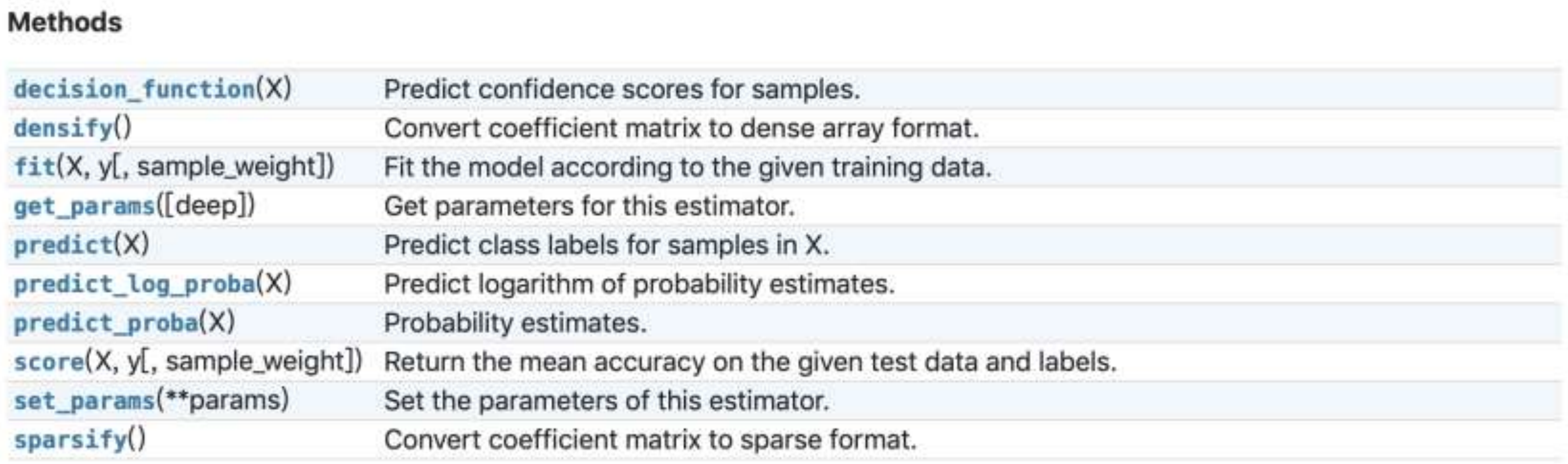
class sklearn.linear_model.LogisticRegression(*penalty='l2'*, ***, *dual=False*, *tol=0.0001*, *C=1.0*, *fit_intercept=True*, *intercept_scaling=1*, *class_weight=None*, *random_state=None*, *solver='lbfgs'*, *max_iter=100*, *multi_class='auto'*, *verbose=0*, *warm_start=False*, *n_jobs=None*, *l1_ratio=None*) [source]

Examples

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0).fit(X, y)
>>> clf.predict(X[:2, :])
array([0, 0])
>>> clf.predict_proba(X[:2, :])
array([[9.8...e-01, 1.8...e-02, 1.4...e-08],
       [9.7...e-01, 2.8...e-02, ...e-08]])
>>> clf.score(X, y)
0.97...
```

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model

逻辑回归模型

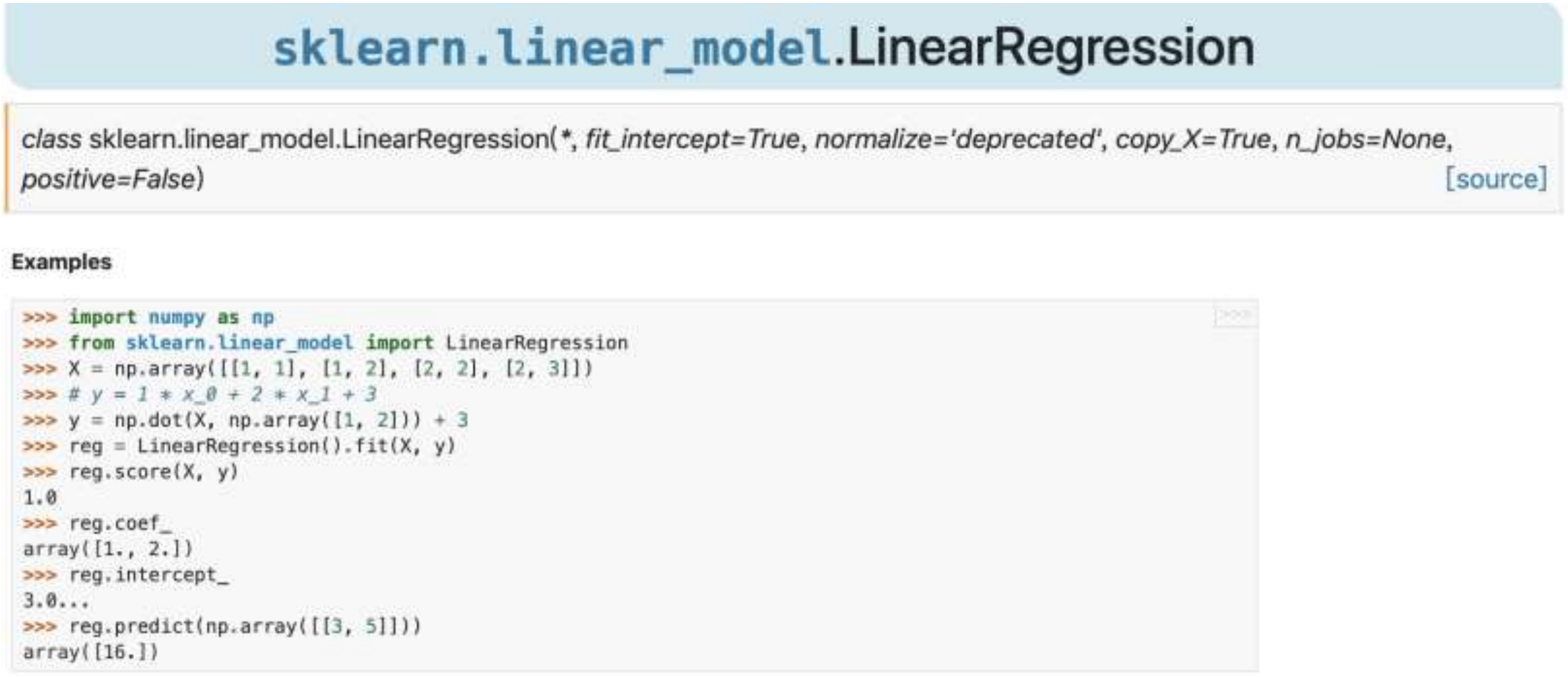


Methods

decision_function(X)	Predict confidence scores for samples.
densify()	Convert coefficient matrix to dense array format.
fit(X, y[, sample_weight])	Fit the model according to the given training data.
get_params([deep])	Get parameters for this estimator.
predict(X)	Predict class labels for samples in X.
predict_log_proba(X)	Predict logarithm of probability estimates.
predict_proba(X)	Probability estimates.
score(X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
set_params(**params)	Set the parameters of this estimator.
sparsify()	Convert coefficient matrix to sparse format.

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model

线性回归模型



sklearn.linear_model.LinearRegression

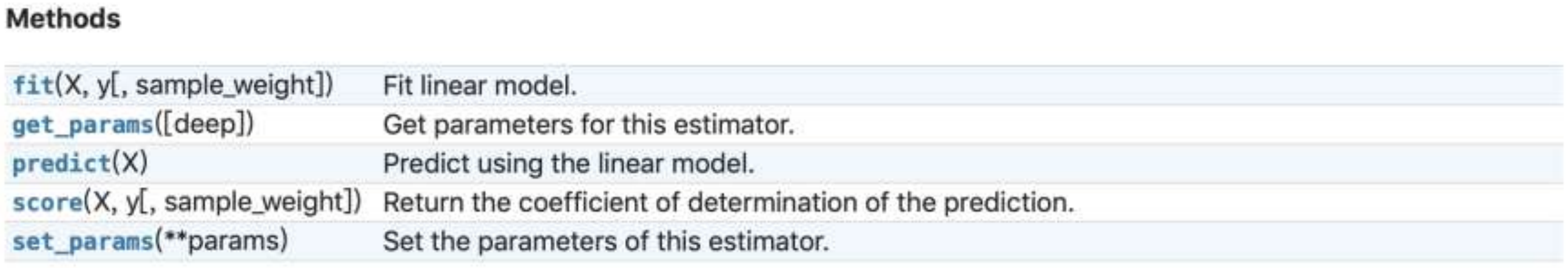
class sklearn.linear_model.LinearRegression(**, fit_intercept=True, normalize='deprecated', copy_X=True, n_jobs=None, positive=False*) [source]

Examples

```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
>>> X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
>>> # y = 1 * x_0 + 2 * x_1 + 3
>>> y = np.dot(X, np.array([1, 2])) + 3
>>> reg = LinearRegression().fit(X, y)
>>> reg.score(X, y)
1.0
>>> reg.coef_
array([1., 2.])
>>> reg.intercept_
3.0...
>>> reg.predict(np.array([[3, 5]]))
array([16.])
```

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model

线性回归模型

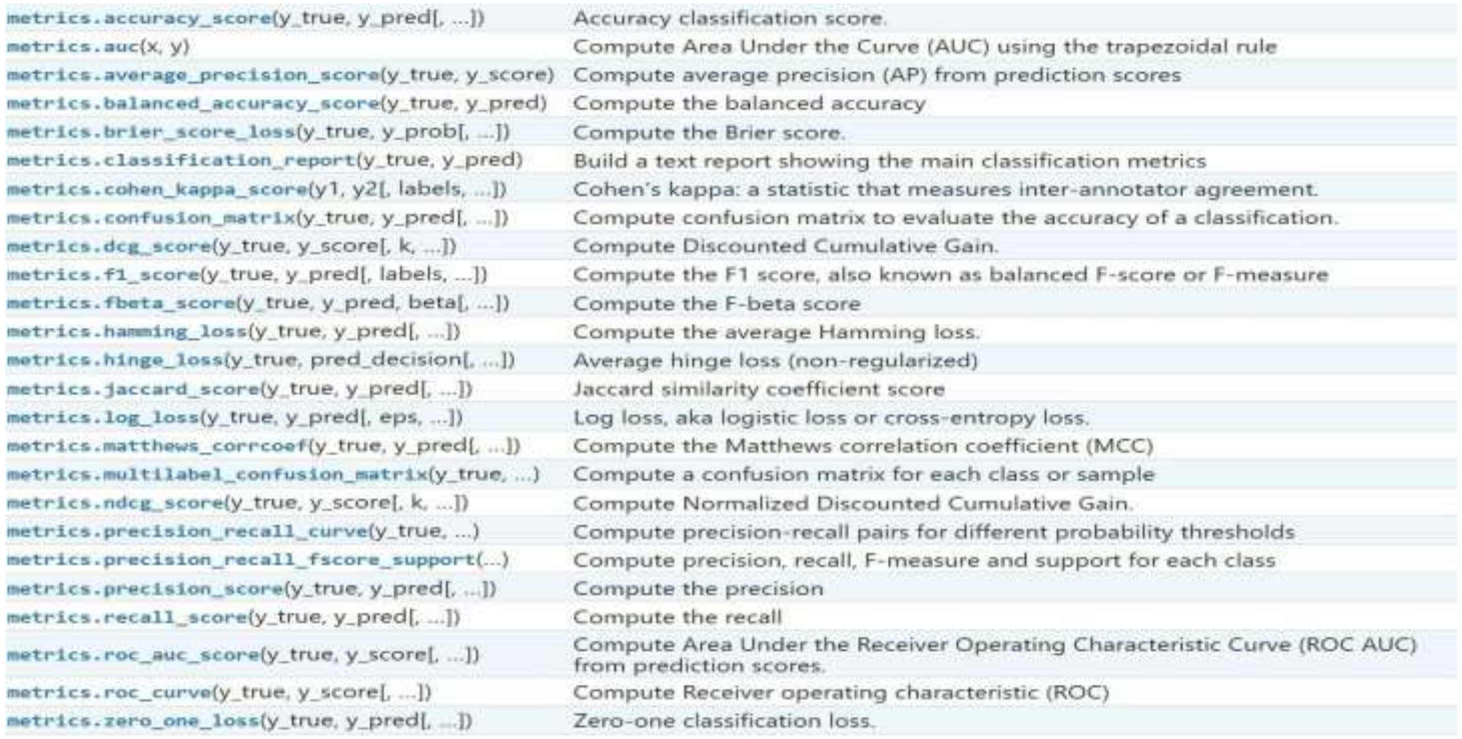


Methods

`fit`(X, y[, sample_weight])	Fit linear model.
`get_params`([deep])	Get parameters for this estimator.
`predict`(X)	Predict using the linear model.
`score`(X, y[, sample_weight])	Return the coefficient of determination of the prediction.
`set_params`(**params)	Set the parameters of this estimator.

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model

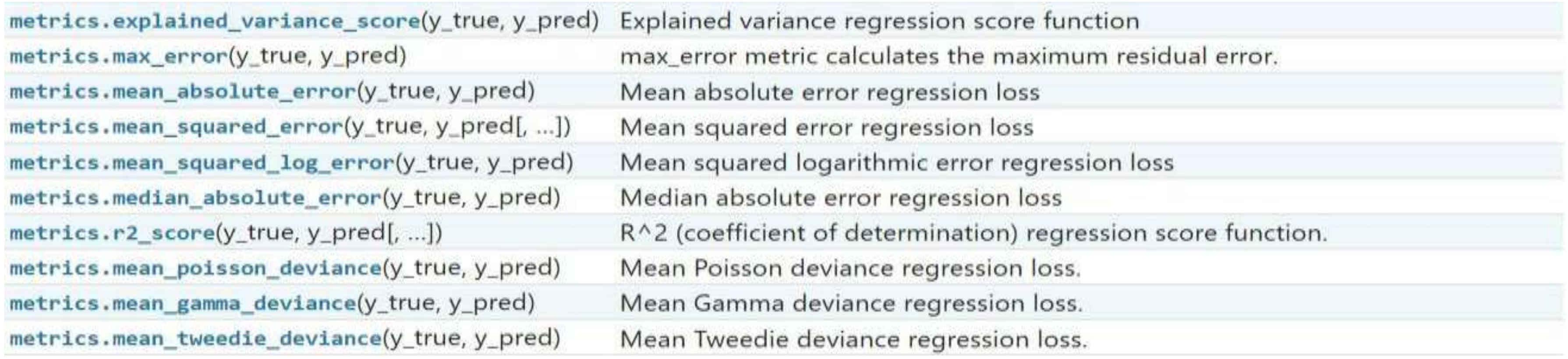
模型评测 Classification metrics



metrics.accuracy_score(y_true, y_pred[, ...])	Accuracy classification score.
metrics.auc(x, y)	Compute Area Under the Curve (AUC) using the trapezoidal rule
metrics.average_precision_score(y_true, y_score)	Compute average precision (AP) from prediction scores
metrics.balanced_accuracy_score(y_true, y_pred)	Compute the balanced accuracy
metrics.brier_score_loss(y_true, y_prob[, ...])	Compute the Brier score.
metrics.classification_report(y_true, y_pred)	Build a text report showing the main classification metrics
metrics.cohen_kappa_score(y1, y2[, labels, ...])	Cohen's kappa: a statistic that measures inter-annotator agreement.
metrics.confusion_matrix(y_true, y_pred[, ...])	Compute confusion matrix to evaluate the accuracy of a classification.
metrics.dcg_score(y_true, y_score[, k, ...])	Compute Discounted Cumulative Gain.
metrics.f1_score(y_true, y_pred[, labels, ...])	Compute the F1 score, also known as balanced F-score or F-measure
metrics.fbeta_score(y_true, y_pred, beta[, ...])	Compute the F-beta score
metrics.hamming_loss(y_true, y_pred[, ...])	Compute the average Hamming loss.
metrics.hinge_loss(y_true, pred_decision[, ...])	Average hinge loss (non-regularized)
metrics.jaccard_score(y_true, y_pred[, ...])	Jaccard similarity coefficient score
metrics.log_loss(y_true, y_pred[, eps, ...])	Log loss, aka logistic loss or cross-entropy loss.
metrics.matthews_corrcoef(y_true, y_pred[, ...])	Compute the Matthews correlation coefficient (MCC)
metrics.multilabel_confusion_matrix(y_true, ...)	Compute a confusion matrix for each class or sample
metrics.ndcg_score(y_true, y_score[, k, ...])	Compute Normalized Discounted Cumulative Gain.
metrics.precision_recall_curve(y_true, ...)	Compute precision-recall pairs for different probability thresholds
metrics.precision_recall_fscore_support(...)	Compute precision, recall, F-measure and support for each class
metrics.precision_score(y_true, y_pred[, ...])	Compute the precision
metrics.recall_score(y_true, y_pred[, ...])	Compute the recall
metrics.roc_auc_score(y_true, y_score[, ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
metrics.roc_curve(y_true, y_score[, ...])	Compute Receiver operating characteristic (ROC)
metrics.zero_one_loss(y_true, y_pred[, ...])	Zero-one classification loss.

https://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics

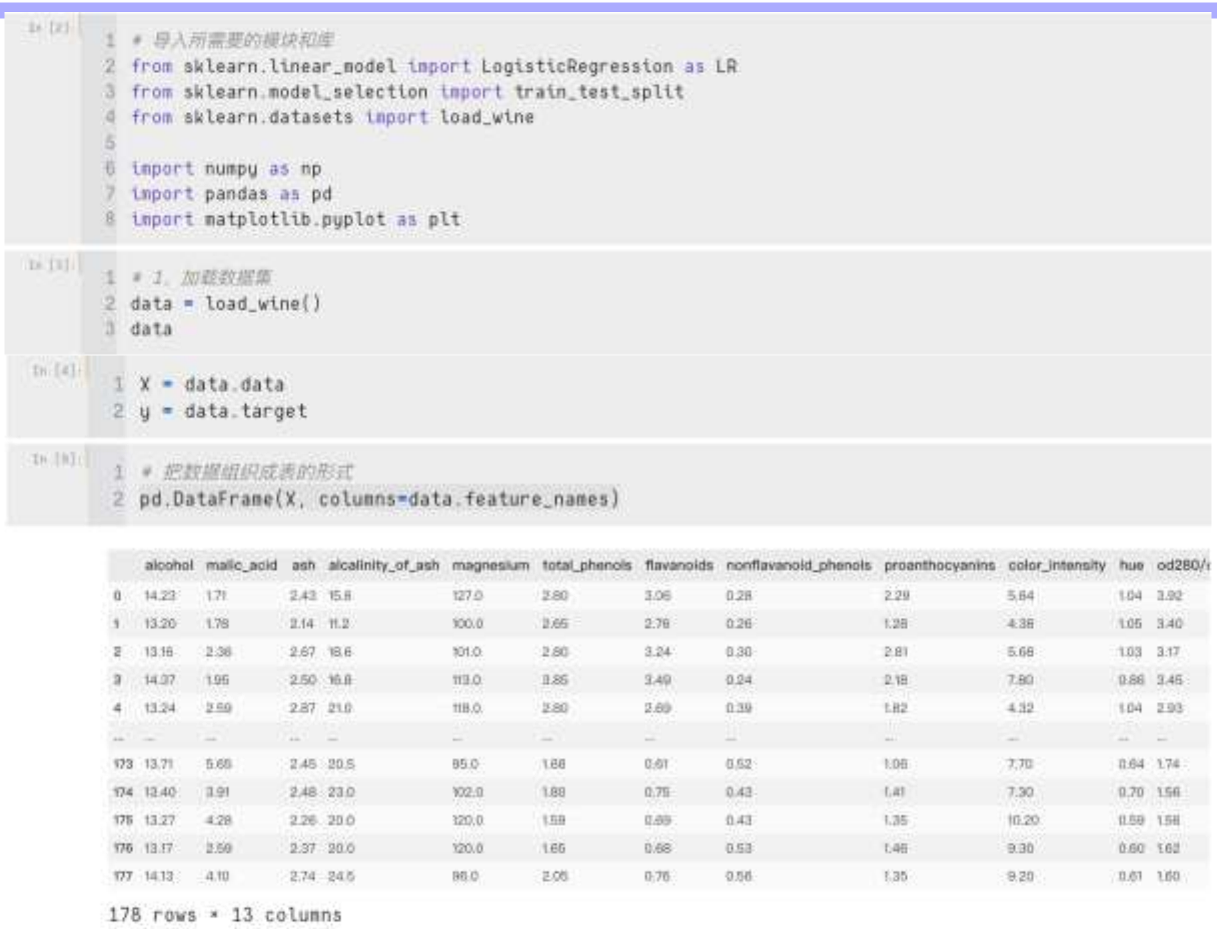
模型评测 Regression metrics



metrics.explained_variance_score(y_true, y_pred)	Explained variance regression score function
metrics.max_error(y_true, y_pred)	max_error metric calculates the maximum residual error.
metrics.mean_absolute_error(y_true, y_pred)	Mean absolute error regression loss
metrics.mean_squared_error(y_true, y_pred[, ...])	Mean squared error regression loss
metrics.mean_squared_log_error(y_true, y_pred)	Mean squared logarithmic error regression loss
metrics.median_absolute_error(y_true, y_pred)	Median absolute error regression loss
metrics.r2_score(y_true, y_pred[, ...])	R^2 (coefficient of determination) regression score function.
metrics.mean_poisson_deviance(y_true, y_pred)	Mean Poisson deviance regression loss.
metrics.mean_gamma_deviance(y_true, y_pred)	Mean Gamma deviance regression loss.
metrics.mean_tweedie_deviance(y_true, y_pred)	Mean Tweedie deviance regression loss.

https://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics

示例：使用逻辑回归模型做分类预测任务



```
In [2]:
# 导入所需要的模块和库
from sklearn.linear_model import LogisticRegression as LR
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_wine

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

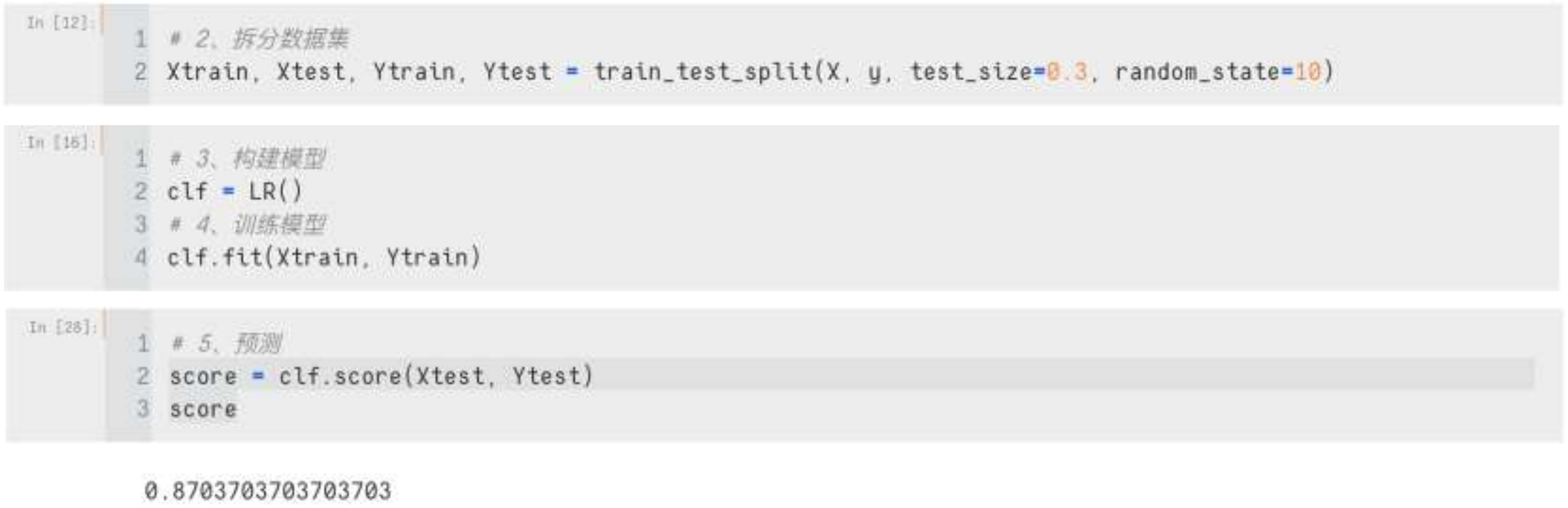
```
In [3]:
# 1. 加载数据集
data = load_wine()
data
```

```
In [4]:
X = data.data
y = data.target
```

```
In [8]:
# 把数据组织成表的形式
pd.DataFrame(X, columns=data.feature_names)
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93
...	...	...	...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.06	7.70	0.64	1.74
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.41	7.30	0.70	1.56
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35	10.20	0.59	1.56
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46	9.30	0.60	1.62
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35	9.20	0.61	1.60

```
178 rows × 13 columns
```

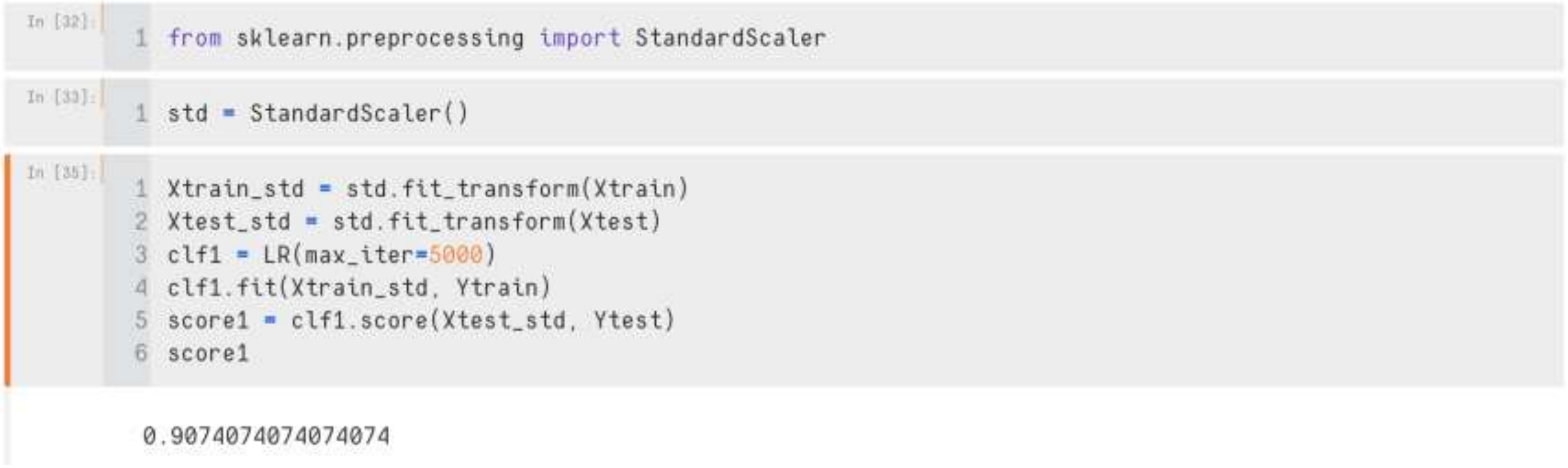


```
In [12]:
# 2、拆分数据集
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, y, test_size=0.3, random_state=10)
```

```
In [16]:
# 3、构建模型
clf = LR()
# 4、训练模型
clf.fit(Xtrain, Ytrain)
```

```
In [28]:
# 5、预测
score = clf.score(Xtest, Ytest)
score
```

```
0.8703703703703703
```

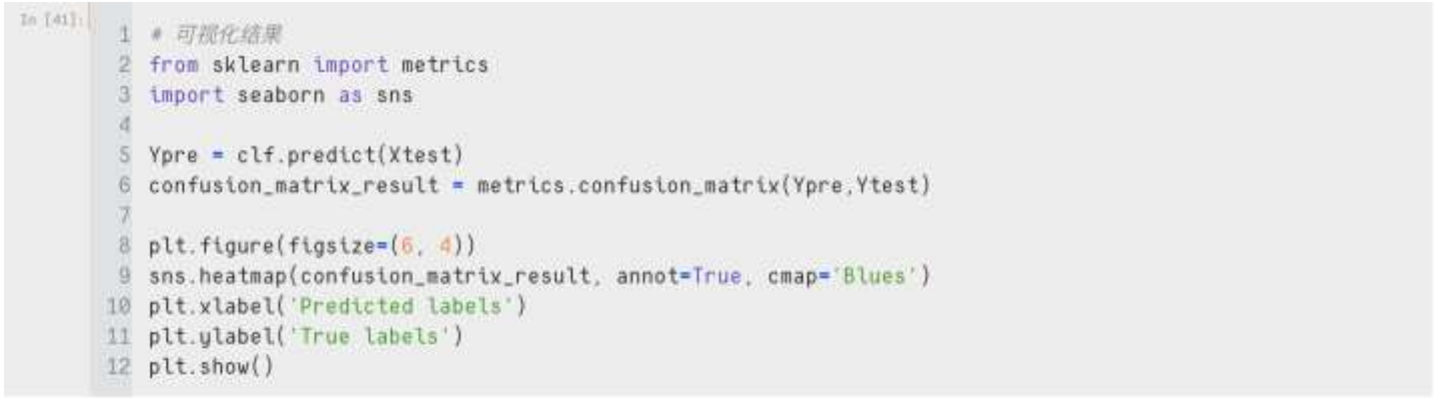


```
In [32]: from sklearn.preprocessing import StandardScaler
```

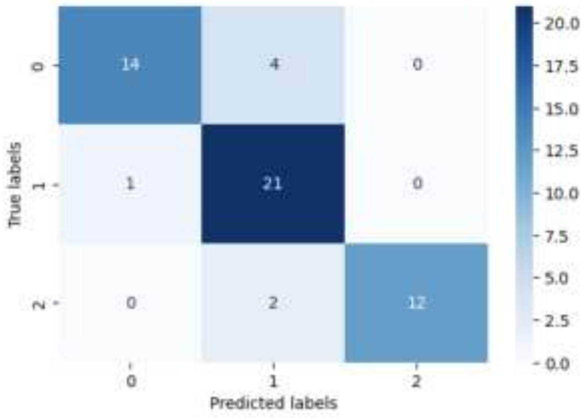
```
In [33]: std = StandardScaler()
```

```
In [35]: Xtrain_std = std.fit_transform(Xtrain)
         Xtest_std = std.fit_transform(Xtest)
         clf1 = LR(max_iter=5000)
         clf1.fit(Xtrain_std, Ytrain)
         score1 = clf1.score(Xtest_std, Ytest)
         score1
```

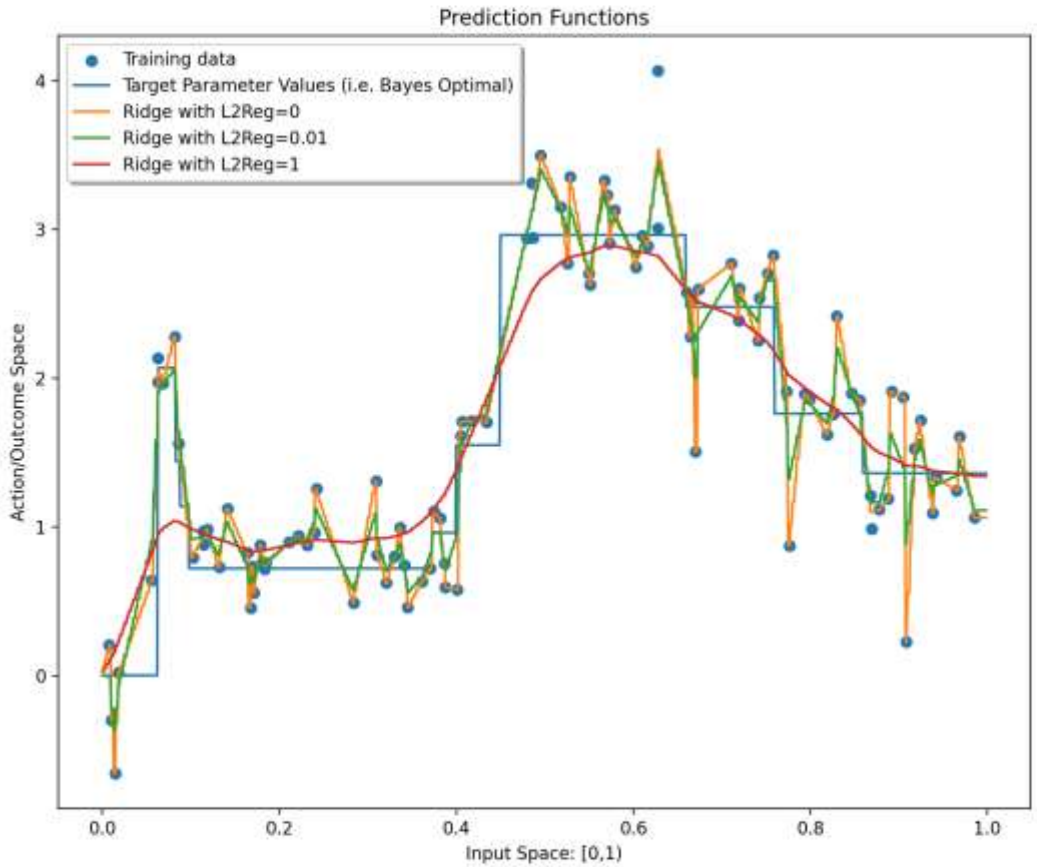
```
0.9074074074074074
```



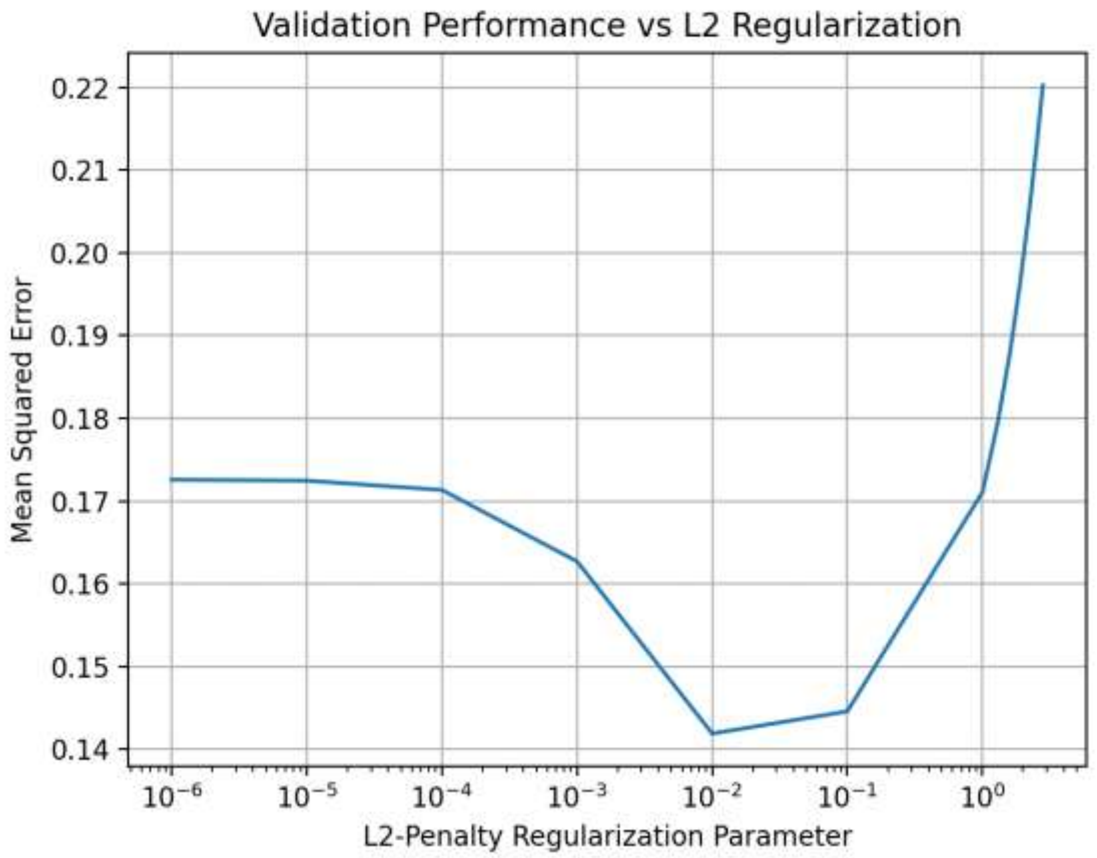
```
In [41]:
# 可视化结果
from sklearn import metrics
import seaborn as sns

Ypre = clf.predict(Xtest)
confusion_matrix_result = metrics.confusion_matrix(Ypre,Ytest)

plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix_result, annot=True, cmap='Blues')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.show()
```



自定义ridge，可视化



自定义ridge，可视化


Validation Performance vs L2 Regularization
Mean Squared Error
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
L2-Penalty Regularization Parameter