算盘实践案例集

企业健康检测预警平台

产品名称：雪浪云健康实时监测预警平台

应用场景：企业、工厂等

功能简介：通过基础信息录入与实时在线数据采集在数字化大屏上实时显示全厂疫情监控预警，实现对防控物资、员工体温、场地消毒的实时跟踪与防控。

下载地址：www.xuelangyun.com

联系人：李白

邮箱：libai.lzj@xuelangyun.com

技术支撑公司：无锡雪浪数制科技有限公司

操作说明

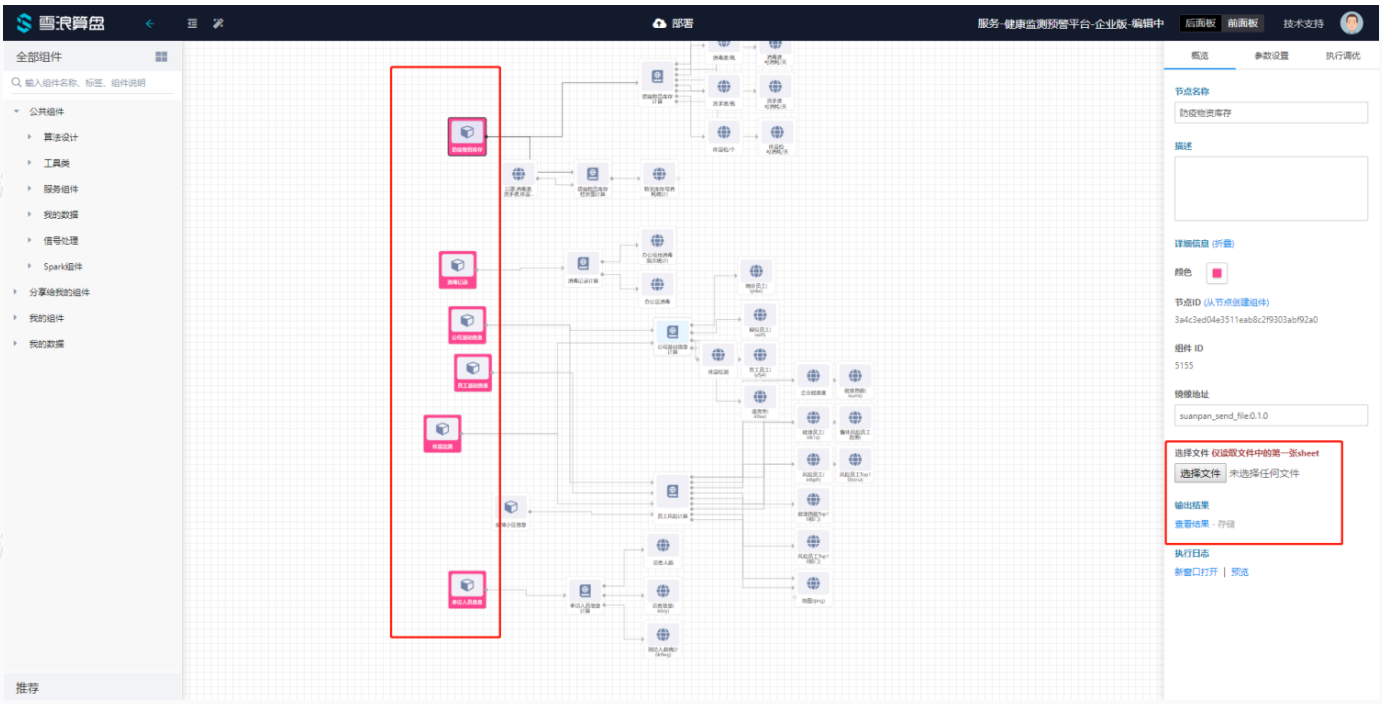
1、登录使用

登录雪浪云官网并注册 www.xuelangyun.com

2、登录后进入控制台，在左侧目录入口选择 **雪浪算盘**，在应用开发模板中点击 **健康监测预警平台-企业版**

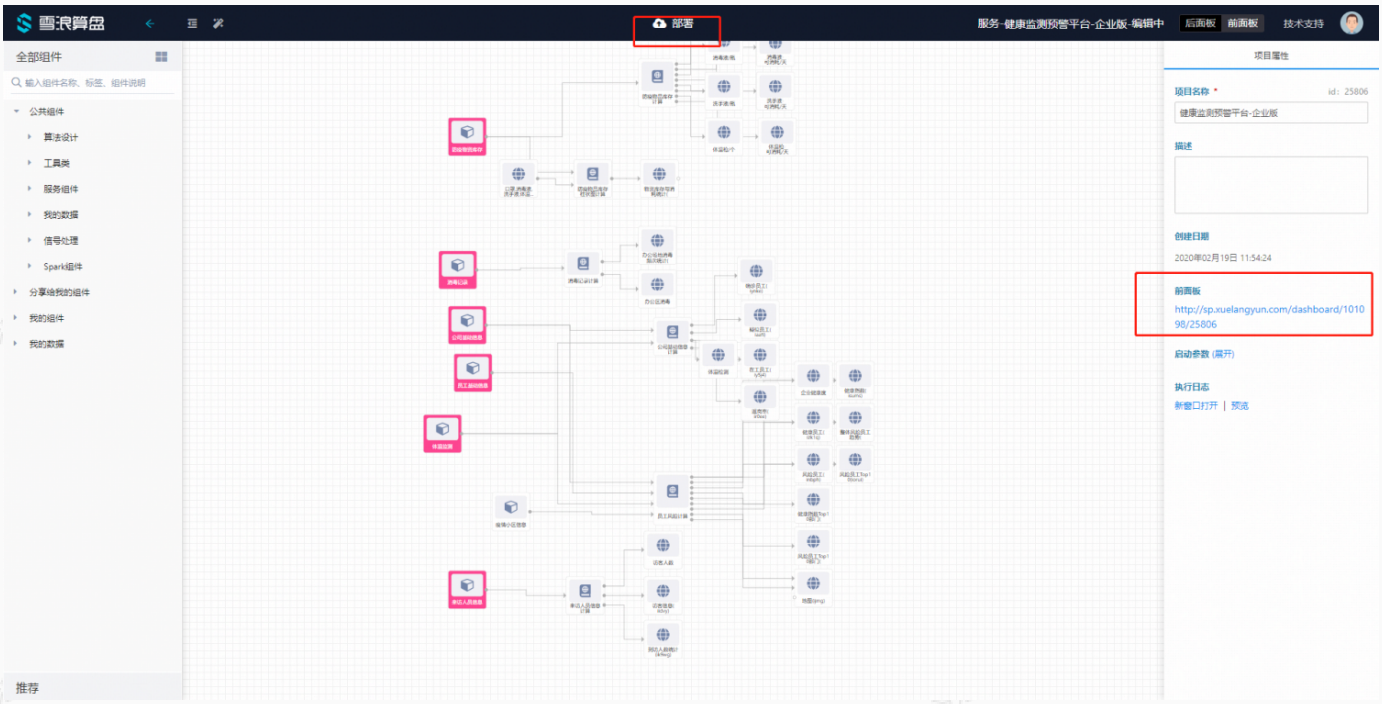
3、下载excel模板与上传

选择红色的数据上传组件，在右侧属性面板 **查看结果** 中下载excel模板，按照规范填写企业信息并在通过 **选择文件** 上传到组件。



4、部署运行

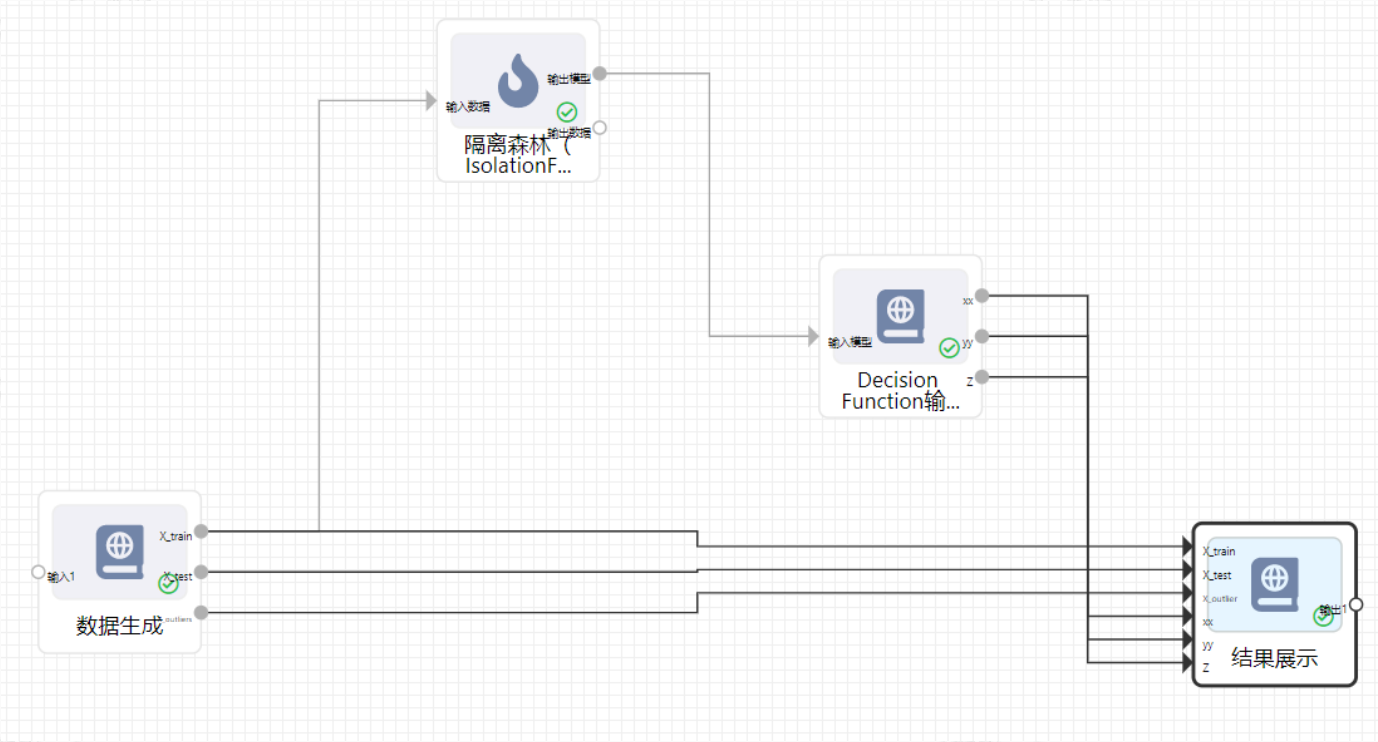
上传好数据后，点击顶部部署，鼠标双击画布区任意空白点，点击 **前面板** 链接，在新开页面就可以看到企业健康大屏。



孤立森林模板

https://scikit-learn.org/stable/auto_examples/ensemble/plot_isolation_forest.html#sphx-glr-auto-examples-ensemble-plot-isolation-forest-py

下图为项目概览：



项目分为四部分：

- 第一部分：数据生成，生成用于训练以及预测的数据

```
import numpy as np
import pandas as pd

import suanpan
from suanpan.app import app
from suanpan.app.arguments import Csv


def toDataFrame(X):
    featureColumns = ["feature_{}".format(i) for i in range(X.shape[1])]
    return pd.DataFrame(X, columns=featureColumns)
```

```
12
13
14 @app.output(Csv(key="outputData1"))
15 @app.output(Csv(key="outputData2"))
16 @app.output(Csv(key="outputData3"))
17 def Demo(context):
18     args = context.args
19
20     rng = np.random.RandomState(42)
21
22     # Generate train data
23     X = 0.3 * rng.randn(100, 2)
24     X_train = np.r_[X + 2, X - 2]
25     # Generate some regular novel observations
26     X = 0.3 * rng.randn(20, 2)
27     X_test = np.r_[X + 2, X - 2]
28     # Generate some abnormal novel observations
29     X_outliers = rng.uniform(low=-4, high=4, size=(20, 2))
30
31     return toDataFrame(X_train), toDataFrame(X_test), toDataFrame
   (X_outliers)
32
33
34 if __name__ == "__main__":
35     suanpan.run(app)
```

- 第二部分：孤立森林节点，接收第一部分生成训练数据，训练模型
  - 模型参数：max_samples=100, contamination=0.1
- 第三部分：Decision Function输出，使用模型的decision_function功能，输出结果

```
1 import joblib
2 import numpy as np
3
4 import suanpan
5 from suanpan.app import app
6 from suanpan.app.arguments import File, Npy
7
8
```

```
@app.input(
    File(
        key="inputModel1", alias="inputModel", name="model", type
="model", required=True
    )
)
@app.output(Npy(key="outputData1"))
@app.output(Npy(key="outputData2"))
@app.output(Npy(key="outputData3"))
def Demo(context):
    args = context.args

    model = joblib.load(args.inputModel)

    xx, yy = np.meshgrid(np.linspace(-5, 5, 50), np.linspace(-5,
5, 50))
    Z = model.decision_function(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    return xx, yy, Z


if __name__ == "__main__":
    suanpan.run(app)
```

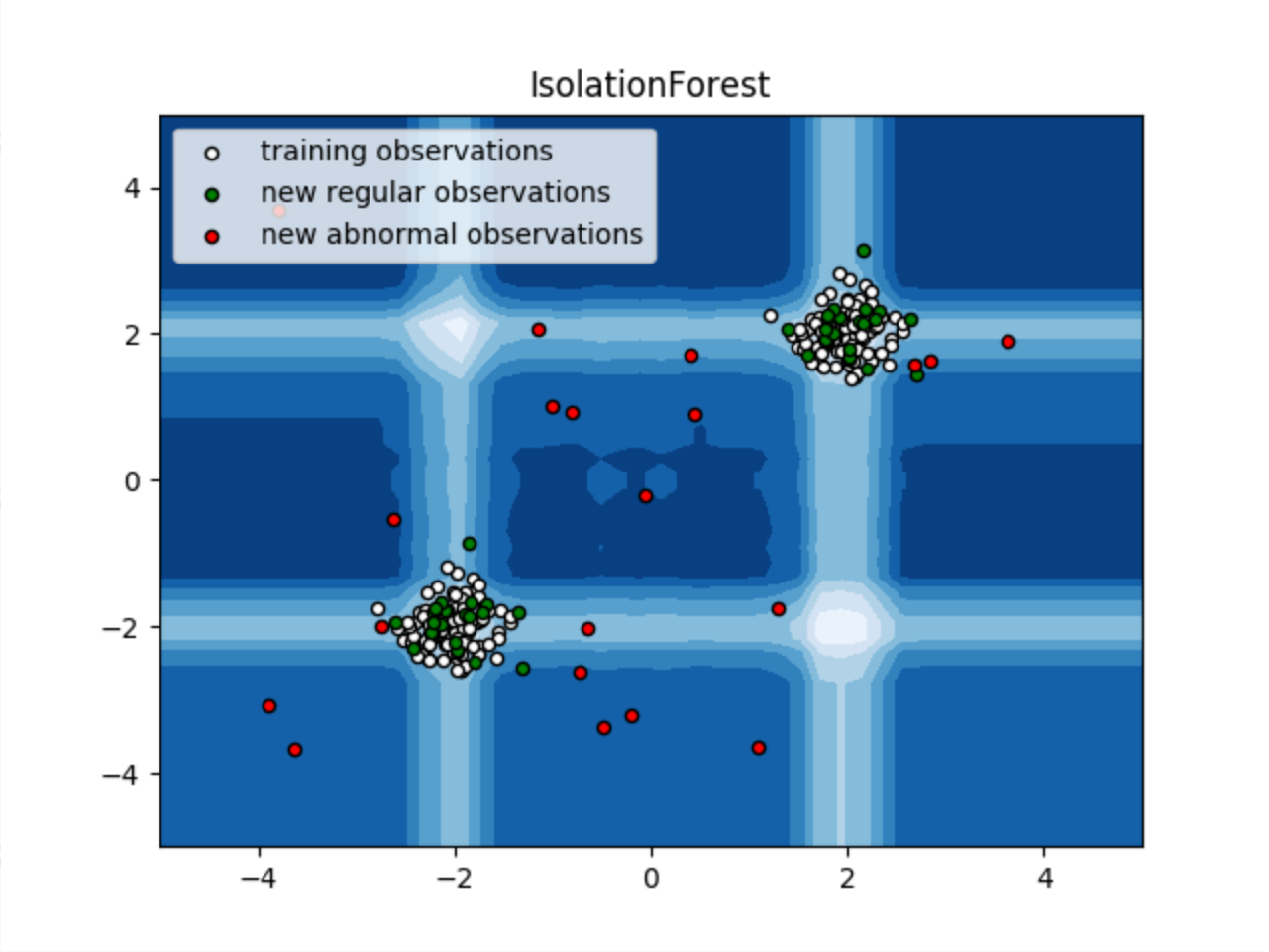
- 第四部分：结果展示

```
import os

from matplotlib import pyplot as plt

import suanpan
from suanpan.app import app
from suanpan.app.arguments import Csv, Folder, Npy

TMP_FOLDER = "/tmp/result"


```

```
@app.input(Csv(key="inputData1", alias="X_train"))
@app.input(Csv(key="inputData2", alias="X_test"))
@app.input(Csv(key="inputData3", alias="X_outliers"))
@app.input(Npy(key="inputData4", alias="xx"))
@app.input(Npy(key="inputData5", alias="yy"))
@app.input(Npy(key="inputData6", alias="Z"))
@app.output(Folder(key="outputData1"))
def Demo(context):
    args = context.args

    X_train = args.X_train.values
    X_test = args.X_test.values
    X_outliers = args.X_outliers.values

    if not os.path.exists(TMP_FOLDER):
        os.makedirs(TMP_FOLDER)

    plt.title("IsolationForest")
    plt.contourf(
        args.xx, args.yy, args.Z, cmap=plt.cm.Blues_r  # pylint: disable=no-member
    )

    b1 = plt.scatter(X_train[:, 0], X_train[:, 1], c="white", s=20, edgecolor="k")
    b2 = plt.scatter(X_test[:, 0], X_test[:, 1], c="green", s=20, edgecolor="k")
    c = plt.scatter(X_outliers[:, 0], X_outliers[:, 1], c="red", s=20, edgecolor="k")
    plt.axis("tight")
    plt.xlim((-5, 5))
    plt.ylim((-5, 5))
    plt.legend(
        [b1, b2, c],
        [
            "training observations",
            "new regular observations",
            "new abnormal observations",
        ],
        loc="upper left",
```

```
    )
    plt.savefig(os.path.join(TMP_FOLDER, "isolation_forest.png"),
format="png")

    return TMP_FOLDER


if __name__ == "__main__":
    suanpan.run(app)
```

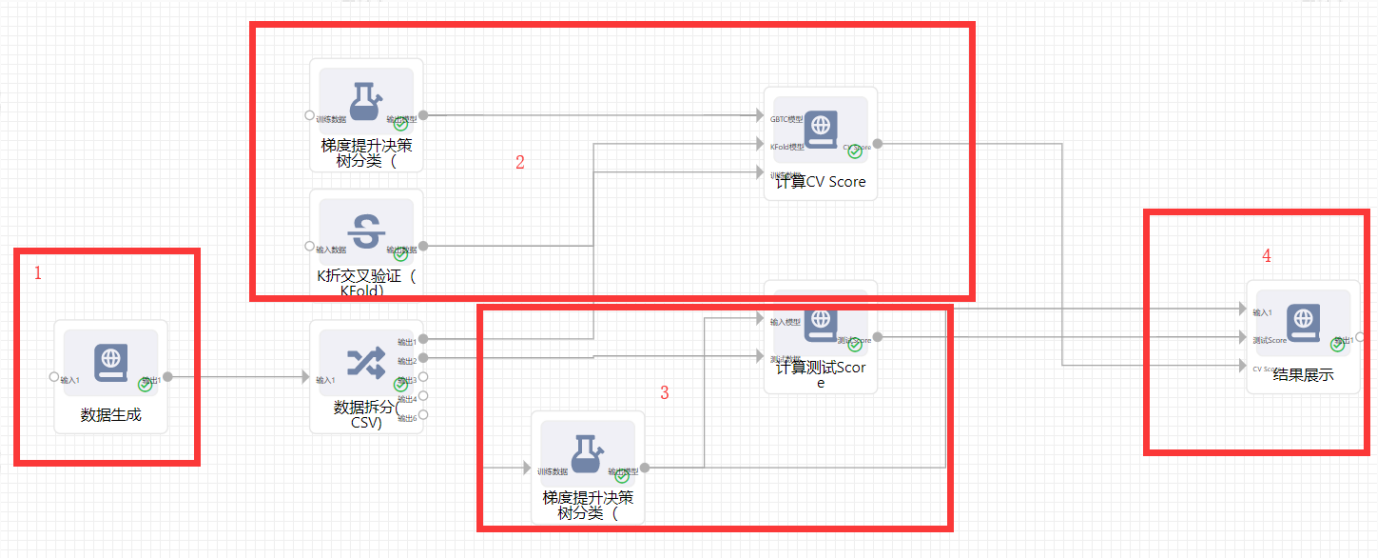
效果如下：



梯度增强OutOfBag预测模板

https://scikit-learn.org/stable/auto_examples/ensemble/plot_gradient_boosting_oob.html#sphx-glr-auto-examples-ensemble-plot-gradient-boosting-oob-py

下图为项目总览：



项目分为四个部分：

第一部分，数据生成，生成用于训练以及预测的数据

```python
import numpy as np
import pandas as pd
from scipy.special import expit

import suanpan
from suanpan.app import app
from suanpan.app.arguments import Csv


@app.output(Csv(key="outputData1"))
def Demo(context):
    args = context.args

    # Generate data (adapted from G. Ridgeway's gbm example)
    n_samples = 1000
```

```
    random_state = np.random.RandomState(13)
    x1 = random_state.uniform(size=n_samples)
    x2 = random_state.uniform(size=n_samples)
    x3 = random_state.randint(0, 4, size=n_samples)

    p = expit(np.sin(3 * x1) - 4 * x2 + x3)
    y = random_state.binomial(1, p, size=n_samples)

    X = np.c_[x1, x2, x3]

    X = X.astype(np.float32)

    feature_columns = ["feature_{}".format(i) for i in range(X.shape[1])]
    label_columns = ["label"]

    feature = pd.DataFrame(X, columns=feature_columns)
    label = pd.DataFrame(y, columns=label_columns)

    return pd.concat([feature, label], axis=1, join_axes=[feature.index])


if __name__ == "__main__":
    suanpan.run(app)
```

第二部分，计算得到交叉验证分数

- 梯度提升决策树分类模型
- KFold模型
- 计算分数的节点

```
import joblib
import numpy as np

import suanpan
from suanpan.app import app
from suanpan.app.arguments import Csv, File, Json, ListOfString, String

n_estimators = 1200
```

```
9
10
11 def heldout_score(clf, X_test, y_test):
12     score = np.zeros((n_estimators,), dtype=np.float64)
13     for i, y_pred in enumerate(clf.staged_decision_function(X_tes
   t)):
14         score[i] = clf.loss_(y_test, y_pred)
15     return score
16
17
18 def cv_estimate(kfold, classifier, X_train, y_train):
19     val_scores = np.zeros((n_estimators,), dtype=np.float64)
20     for train, test in kfold.split(X_train, y_train):
21         classifier.fit(X_train[train], y_train[train])
22         val_scores += heldout_score(classifier, X_train[test], y_
   train[test])
23     val_scores /= kfold.n_splits
24     return val_scores
25
26
27 @app.input(
28     File(
29         key="inputModel1", alias="gbtcModel", name="model", type=
   "model", required=True
30     )
31 )
32 @app.input(
33     File(
34         key="inputModel2", alias="kfoldModel", name="model", type
   ="model", required=True
35     )
36 )
37 @app.input(Csv(key="inputData3", alias="trainData", required=True
   ))
38 @app.param(ListOfString(key="param1", alias="featureColumns", req
   uired=True))
39 @app.param(String(key="param2", alias="labelColumn", required=Tru
   e))
40 @app.output(Json(key="outputData1"))
41 def Demo(context):
```

```
    args = context.args

    classifier = joblib.load(args.gbtcModel)
    kfold = joblib.load(args.kfoldModel)
    df = args.trainData

    X_train = df[args.featureColumns].values
    y_train = df[args.labelColumn].values

    return {"cv_score": cv_estimate(kfold, classifier, X_train, y_train)}


if __name__ == "__main__":
    suanpan.run(app)
```

第三部分，计算得到测试分数

- 梯度提升决策树分类模型
- 计算分数节点

```
import joblib
import numpy as np

import suanpan
from suanpan.app import app
from suanpan.app.arguments import Csv, File, Json, ListOfString, String

n_estimators = 1200


def heldout_score(classifier, X_test, y_test):
    score = np.zeros((n_estimators,), dtype=np.float64)
    for i, y_pred in enumerate(classifier.staged_decision_function(X_test)):
        score[i] = classifier.loss_(y_test, y_pred)
    return score


```

```python
@app.input(
    File(
        key="inputModel1", alias="inputModel", name="model", type="model", required=True
    )
)
@app.input(Csv(key="inputData2", alias="testData", required=True))
@app.param(ListOfString(key="param1", alias="featureColumns", required=True))
@app.param(String(key="param2", alias="labelColumn", required=True))
@app.output(Json(key="outputData1"))
def Demo(context):
    args = context.args

    classifier = joblib.load(args.inputModel)

    df = args.testData
    X_test = df[args.featureColumns].values
    y_test = df[args.labelColumn].values

    return {"test_score": heldout_score(classifier, X_test, y_test)}


if __name__ == "__main__":
    suanpan.run(app)
```

第四部分，结果展示

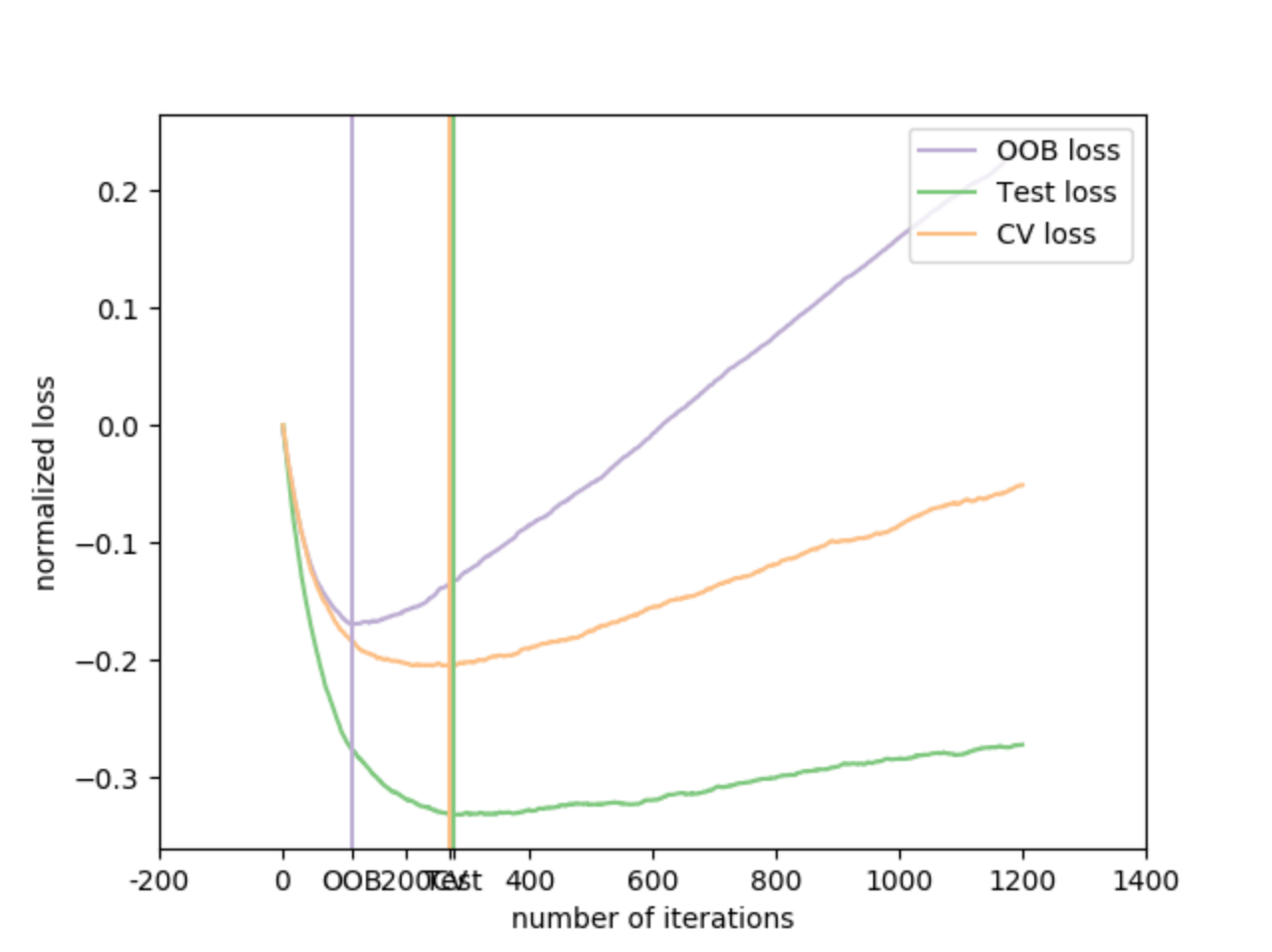
```python
import os

import joblib
import numpy as np
from matplotlib import pyplot as plt

import suanpan
from suanpan.app import app
```

```
from suanpan.app.arguments import File, Folder, Json

n_estimators = 1200
TMP_FOLDER = "/tmp/result"


@app.input(
    File(
        key="inputModel1", alias="inputModel", name="model", type
="model", required=True
    )
)
@app.input(Json(key="inputData2", alias="testScore", required=Tru
e))
@app.input(Json(key="inputData3", alias="cvScore", required=True)
)
@app.output(Folder(key="outputData1"))
def Demo(context):
    args = context.args

    if not os.path.exists(TMP_FOLDER):
        os.makedirs(TMP_FOLDER)

    x = np.arange(n_estimators) + 1

    classifier = joblib.load(args.inputModel)
    # Estimate best n_estimator using cross-validation
    cv_score = np.array(args.cvScore["cv_score"])

    # Compute best n_estimator for test data
    test_score = np.array(args.testScore["test_score"])

    # negative cumulative sum of oob improvements
    cumsum = -np.cumsum(classifier.oob_improvement_)

    # min loss according to OOB
    oob_best_iter = x[np.argmin(cumsum)]

    # min loss according to test (normalize such that first loss
 is 0)
```

```
    test_score -= test_score[0]
    test_best_iter = x[np.argmin(test_score)]

    # min loss according to cv (normalize such that first loss is
0)
    cv_score -= cv_score[0]
    cv_best_iter = x[np.argmin(cv_score)]

    # color brew for the three curves
    oob_color = list(map(lambda x: x / 256.0, (190, 174, 212)))
    test_color = list(map(lambda x: x / 256.0, (127, 201, 127)))
    cv_color = list(map(lambda x: x / 256.0, (253, 192, 134)))

    # plot curves and vertical lines for best iterations
    plt.plot(x, cumsum, label="OOB loss", color=oob_color)
    plt.plot(x, test_score, label="Test loss", color=test_color)
    plt.plot(x, cv_score, label="CV loss", color=cv_color)
    plt.axvline(x=oob_best_iter, color=oob_color)
    plt.axvline(x=test_best_iter, color=test_color)
    plt.axvline(x=cv_best_iter, color=cv_color)

    # add three vertical lines to xticks
    xticks = plt.xticks()
    xticks_pos = np.array(
        xticks[0].tolist() + [oob_best_iter, cv_best_iter, test_b
est_iter]
    )
    xticks_label = np.array(
        list(map(lambda t: int(t), xticks[0])) + ["OOB", "CV", "T
est"]
    )
    ind = np.argsort(xticks_pos)
    xticks_pos = xticks_pos[ind]
    xticks_label = xticks_label[ind]
    plt.xticks(xticks_pos, xticks_label)

    plt.legend(loc="upper right")
    plt.ylabel("normalized loss")
    plt.xlabel("number of iterations")
    plt.savefig(os.path.join(TMP_FOLDER, "oob.png"), format="png"
```

```
    )
82
83      return TMP_FOLDER
84
85
86  if __name__ == "__main__":
87      suanpan.run(app)
```

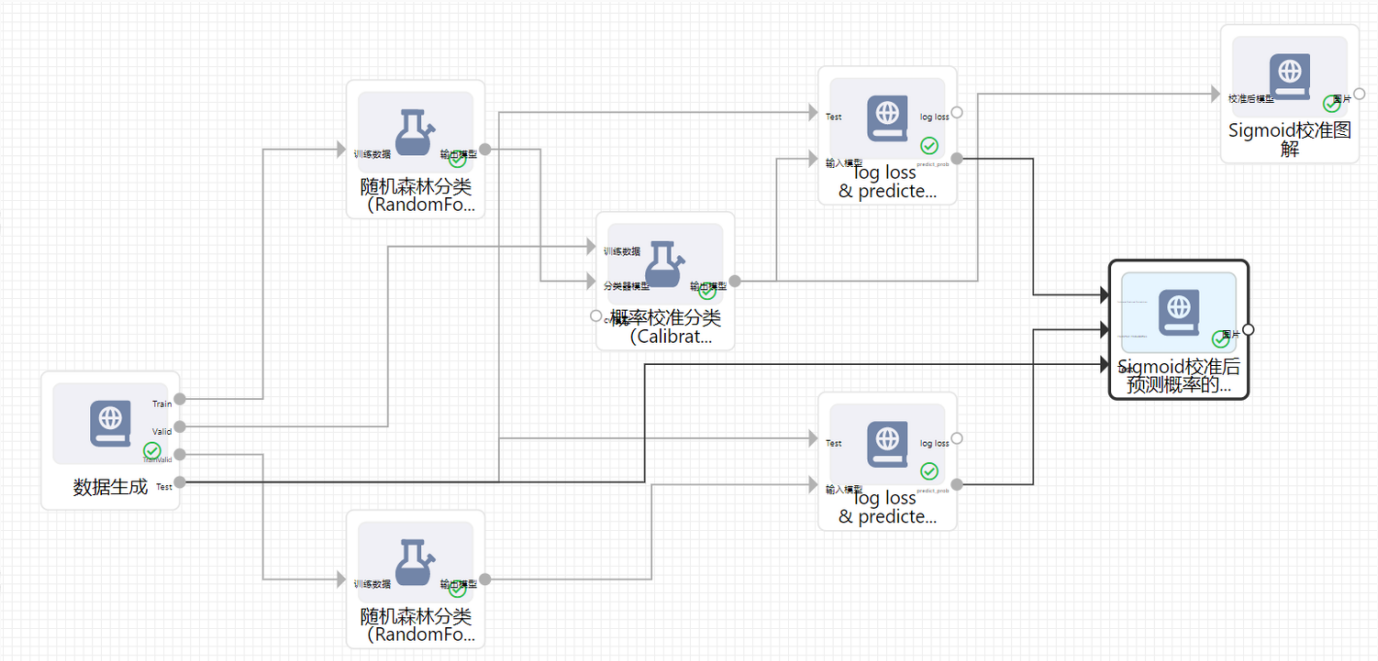
效果图如下：



3分类问题概率校准模板

https://scikit-learn.org/stable/auto_examples/calibration/plot_calibration_multiclass.html#sphx-glr-auto-examples-calibration-plot-calibration-multiclass-py

下图为项目总览：



项目可以分为四个部分：

- 第一部分，数据生成，分别生成训练数据，校准数据以及测试数据

```
import numpy as np
import pandas
from sklearn.datasets import make_blobs

import suanpan
from suanpan.app import app
from suanpan.app.arguments import Csv


def createDataFrame(X, y):
    featureCount = X.shape[1]

    featureColumns = ["feature_{}".format(i) for i in range(featu
```

```
reCount)]
    labelColumns = ["label"]

    features = pandas.DataFrame(X, columns=featureColumns)
    label = pandas.DataFrame(y, columns=labelColumns)

    return pandas.concat([features, label], axis=1, join_axes=[features.index])


@app.output(Csv(key="outputData1"))
@app.output(Csv(key="outputData2"))
@app.output(Csv(key="outputData3"))
@app.output(Csv(key="outputData4"))
def Demo(context):
    args = context.args

    np.random.seed(0)

    # Generate data
    X, y = make_blobs(n_samples=1000, n_features=2, random_state=42, cluster_std=5.0)
    X_train, y_train = X[:600], y[:600]
    X_valid, y_valid = X[600:800], y[600:800]
    X_train_valid, y_train_valid = X[:800], y[:800]
    X_test, y_test = X[800:], y[800:]

    return (
        createDataFrame(X_train, y_train),
        createDataFrame(X_valid, y_valid),
        createDataFrame(X_train_valid, y_train_valid),
        createDataFrame(X_test, y_test),
    )


if __name__ == "__main__":
    suanpan.run(app)
```

- 第二部分，模型训练
  - 随机森林分类模型

        - 参数：n_estimators=25
    - 随机森林分类模型加上概率校准分类模型
        - 随机森林参数：n_estimators=25
        - 概率校准模型参数：method=sigmoid，prefit=True
- 第三部分，计算两种模型的log loss以及预测概率

```python
import joblib
from sklearn.metrics import log_loss

import suanpan
from suanpan.app import app
from suanpan.app.arguments import Csv, File, Json, ListOfString, Npy, String


@app.input(Csv(key="inputData1", alias="inputData", required=True))
@app.input(
    File(
        key="inputModel2", alias="inputModel", name="model", type="model", required=True
    )
)
@app.column(ListOfString(key="param1", alias="featureColumns", required=True))
@app.column(String(key="param2", alias="labelColumn", required=True))
@app.output(Json(key="outputData1"))
@app.output(Npy(key="outputData2"))
def Demo(context):
    args = context.args

    df = args.inputData
    clf = joblib.load(args.inputModel)

    clf_probs = clf.predict_proba(df[args.featureColumns].values)
    score = log_loss(df[args.labelColumn].values, clf_probs)

    return {"score": score}, clf_probs
```

```
29
30
31 if __name__ == "__main__":
32     suanpan.run(app)
```

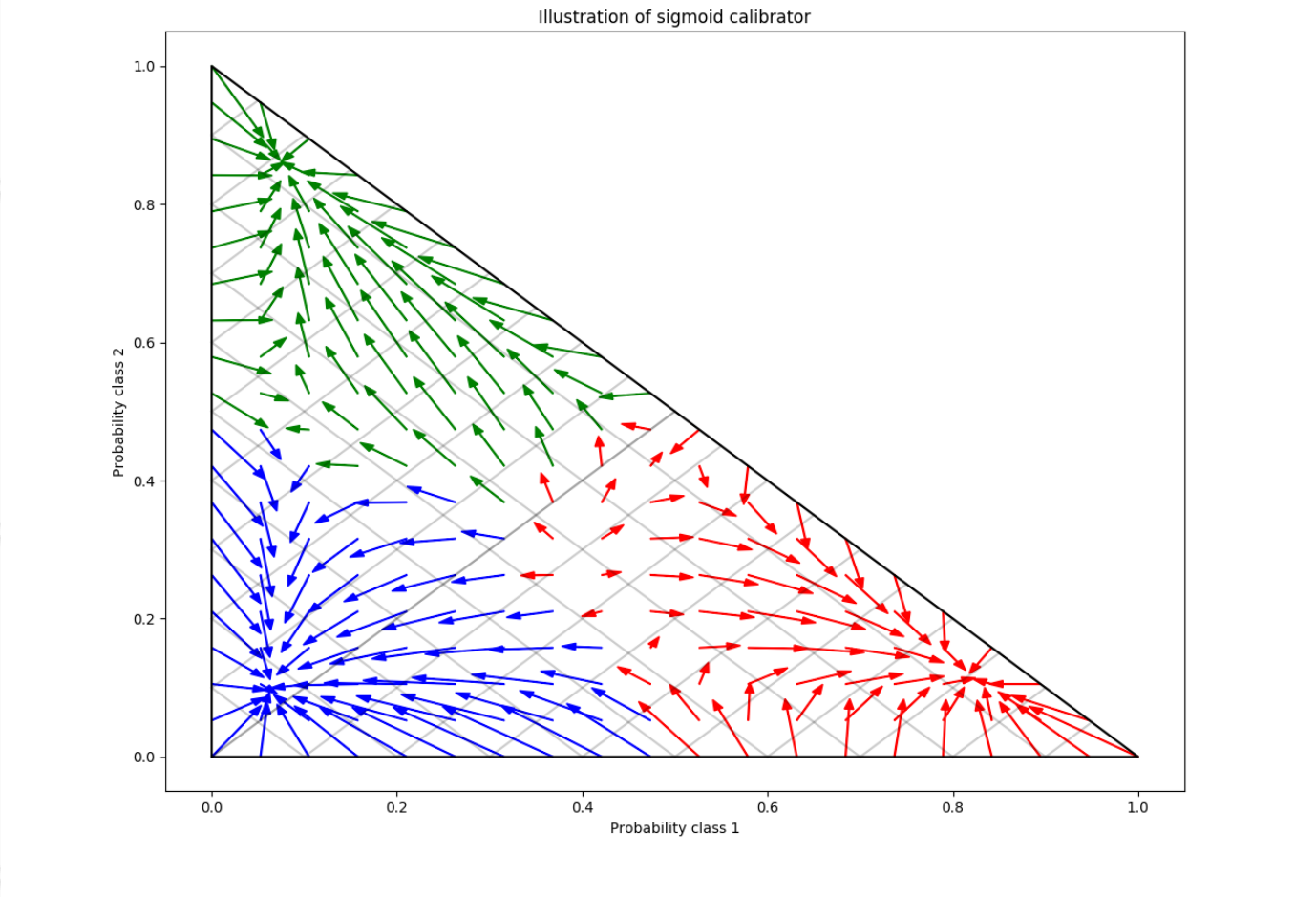
- 第四部分，概率校准图解
  - Sigmoid校准图解

```
1  import os
2
3  import joblib
4  import numpy as np
5  from matplotlib import pyplot as plt
6
7  import suanpan
8  from suanpan.app import app
9  from suanpan.app.arguments import File, Folder
10
11 TMP_FOLDER = "/tmp/result"
12
13
14 @app.input(
15     File(
16         key="inputModel1",
17         alias="calibratedModel",
18         name="model",
19         type="model",
20         required=True,
21     )
22 )
23 @app.output(Folder(key="outputData1"))
24 def Demo(context):
25     args = context.args
26
27     if not os.path.exists(TMP_FOLDER):
28         os.makedirs(TMP_FOLDER)
29
30     calibratedModel = joblib.load(args.calibratedModel)
31
32     # Illustrate calibrator
```

```
    plt.figure(figsize=(12.8, 9.6))
    colors = ["r", "g", "b"]
    # generate grid over 2-simplex
    p1d = np.linspace(0, 1, 20)
    p0, p1 = np.meshgrid(p1d, p1d)
    p2 = 1 - p0 - p1
    p = np.c_[p0.ravel(), p1.ravel(), p2.ravel()]
    p = p[p[:, 2] >= 0]

    calibratedClassifier = calibratedModel.calibrated_classifiers_[0]
    prediction = np.vstack(
        [
            calibrator.predict(this_p)
            for calibrator, this_p in zip(calibratedClassifier.calibrators_, p.T)
        ]
    ).T
    prediction /= prediction.sum(axis=1)[:, None]

    # Plot modifications of calibrator
    for i in range(prediction.shape[0]):
        plt.arrow(
            p[i, 0],
            p[i, 1],
            prediction[i, 0] - p[i, 0],
            prediction[i, 1] - p[i, 1],
            head_width=1e-2,
            color=colors[np.argmax(p[i])],
        )
    # Plot boundaries of unit simplex
    plt.plot([0.0, 1.0, 0.0, 0.0], [0.0, 0.0, 1.0, 0.0], "k", label="Simplex")

    plt.grid(False)
    for x in [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]:
        plt.plot([0, x], [x, 0], "k", alpha=0.2)
        plt.plot([0, 0 + (1 - x) / 2], [x, x + (1 - x) / 2], "k", alpha=0.2)
```

```
68          plt.plot([x, x + (1 - x) / 2], [0, 0 + (1 - x) / 2], "k",
    alpha=0.2)
69
70      plt.title("Illustration of sigmoid calibrator")
71      plt.xlabel("Probability class 1")
72      plt.ylabel("Probability class 2")
73      plt.xlim(-0.05, 1.05)
74      plt.ylim(-0.05, 1.05)
75      plt.savefig(os.path.join(TMP_FOLDER, "calibration.png"), form
    at="png")
76
77      return TMP_FOLDER
78
79
80  if __name__ == "__main__":
81      suanpan.run(app)
```

效果图如下：



- Sigmoid校准后预测概率的变化图

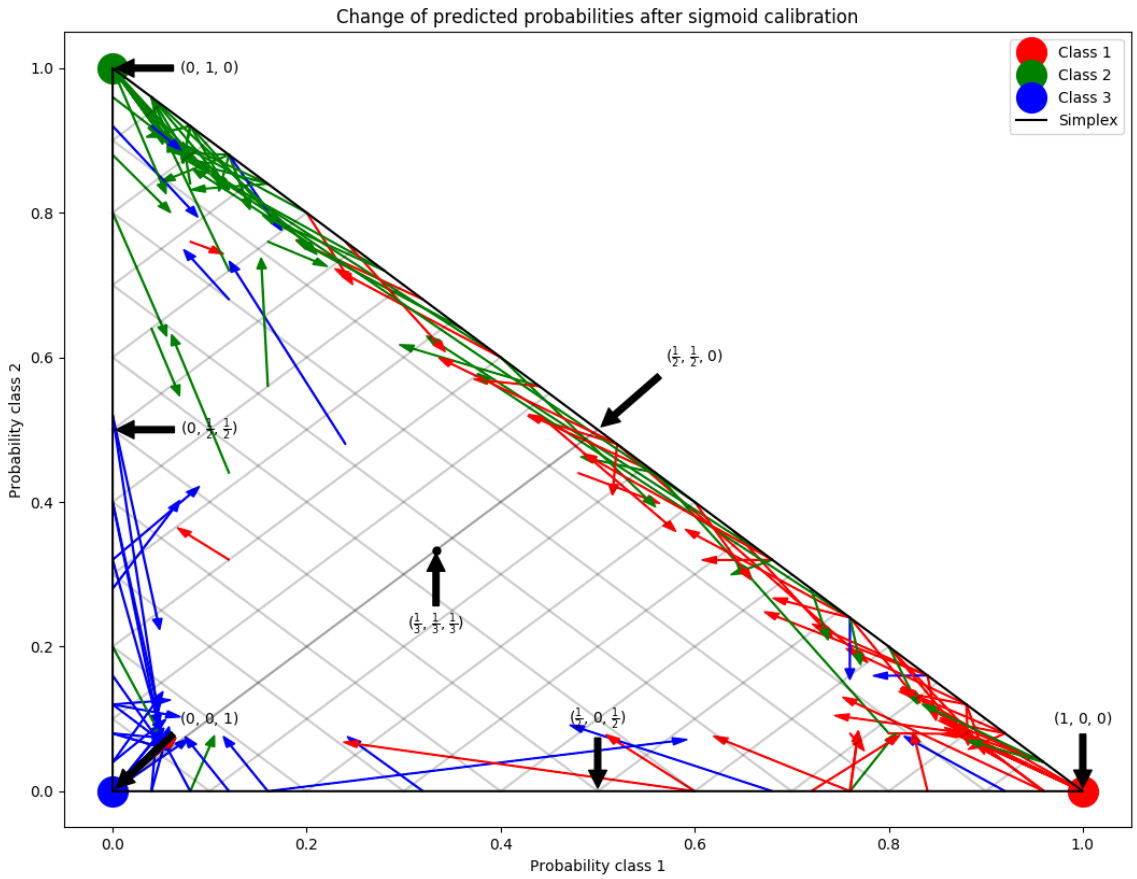
```python
import os

from matplotlib import pyplot as plt

import suanpan
from suanpan.app import app
from suanpan.app.arguments import Csv, Folder, Npy, String

TMP_FOLDER = "/tmp/result"


@app.input(Npy(key="inputData1", alias="calibratedProbs", required=True))
@app.input(Npy(key="inputData2", alias="probs", required=True))
@app.input(Csv(key="inputData3", alias="testData", required=True))
@app.param(String(key="param1", alias="labelColumn", required=True))
@app.output(Folder(key="outputData1"))
def Demo(context):
    args = context.args

    if not os.path.exists(TMP_FOLDER):
        os.makedirs(TMP_FOLDER)

    calibratedProbs = args.calibratedProbs
    probs = args.probs
    yTest = args.testData[args.labelColumn].values

    # Plot changes in predicted probabilities via arrows
    plt.figure(figsize=(12.8, 9.6))
    colors = ["r", "g", "b"]
    for i in range(probs.shape[0]):
        plt.arrow(
            probs[i, 0],
            probs[i, 1],
            calibratedProbs[i, 0] - probs[i, 0],
            calibratedProbs[i, 1] - probs[i, 1],
```

```
            color=colors[yTest[i]],
            head_width=1e-2,
        )

    # Plot perfect predictions
    plt.plot([1.0], [0.0], "ro", ms=20, label="Class 1")
    plt.plot([0.0], [1.0], "go", ms=20, label="Class 2")
    plt.plot([0.0], [0.0], "bo", ms=20, label="Class 3")

    # Plot boundaries of unit simplex
    plt.plot([0.0, 1.0, 0.0, 0.0], [0.0, 0.0, 1.0, 0.0], "k", la
bel="Simplex")

    # Annotate points on the simplex
    plt.annotate(
        r"($\frac{1}{3}$, $\frac{1}{3}$, $\frac{1}{3}$)",
        xy=(1.0 / 3, 1.0 / 3),
        xytext=(1.0 / 3, 0.23),
        xycoords="data",
        arrowprops=dict(facecolor="black", shrink=0.05),
        horizontalalignment="center",
        verticalalignment="center",
    )
    plt.plot([1.0 / 3], [1.0 / 3], "ko", ms=5)
    plt.annotate(
        r"($\frac{1}{2}$, $0$, $\frac{1}{2}$)",
        xy=(0.5, 0.0),
        xytext=(0.5, 0.1),
        xycoords="data",
        arrowprops=dict(facecolor="black", shrink=0.05),
        horizontalalignment="center",
        verticalalignment="center",
    )
    plt.annotate(
        r"($0$, $\frac{1}{2}$, $\frac{1}{2}$)",
        xy=(0.0, 0.5),
        xytext=(0.1, 0.5),
        xycoords="data",
        arrowprops=dict(facecolor="black", shrink=0.05),
        horizontalalignment="center",
```

```
        verticalalignment="center",
    )
    plt.annotate(
        r"($\frac{1}{2}$, $\frac{1}{2}$, $0$)",
        xy=(0.5, 0.5),
        xytext=(0.6, 0.6),
        xycoords="data",
        arrowprops=dict(facecolor="black", shrink=0.05),
        horizontalalignment="center",
        verticalalignment="center",
    )
    plt.annotate(
        r"($0$, $0$, $1$)",
        xy=(0, 0),
        xytext=(0.1, 0.1),
        xycoords="data",
        arrowprops=dict(facecolor="black", shrink=0.05),
        horizontalalignment="center",
        verticalalignment="center",
    )
    plt.annotate(
        r"($1$, $0$, $0$)",
        xy=(1, 0),
        xytext=(1, 0.1),
        xycoords="data",
        arrowprops=dict(facecolor="black", shrink=0.05),
        horizontalalignment="center",
        verticalalignment="center",
    )
    plt.annotate(
        r"($0$, $1$, $0$)",
        xy=(0, 1),
        xytext=(0.1, 1),
        xycoords="data",
        arrowprops=dict(facecolor="black", shrink=0.05),
        horizontalalignment="center",
        verticalalignment="center",
    )

    # Add grid
```

```
    plt.grid(False)
    for x in [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]:
        plt.plot([0, x], [x, 0], "k", alpha=0.2)
        plt.plot([0, 0 + (1 - x) / 2], [x, x + (1 - x) / 2], "k", alpha=0.2)
        plt.plot([x, x + (1 - x) / 2], [0, 0 + (1 - x) / 2], "k", alpha=0.2)

    plt.title("Change of predicted probabilities after sigmoid calibration")
    plt.xlabel("Probability class 1")
    plt.ylabel("Probability class 2")
    plt.xlim(-0.05, 1.05)
    plt.ylim(-0.05, 1.05)
    plt.legend(loc="best")
    plt.savefig(os.path.join(TMP_FOLDER, "calibration.png"), format="png")

    return TMP_FOLDER


if __name__ == "__main__":
    suanpan.run(app)
```

效果图如下：



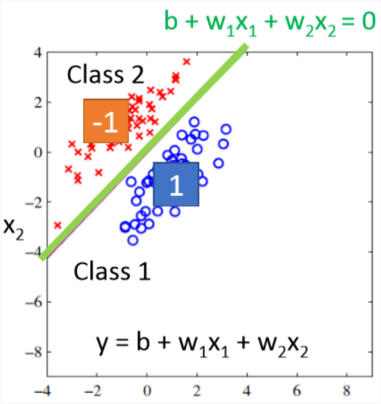
LightGBM分类模板

LightGBM分类模板

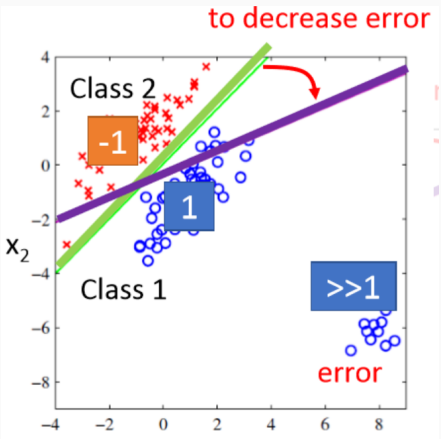
分类问题

机器学习中一个重要的任务——分类（classification），即找一个函数判断输入数据所属的类别，可以是二类别问题（是/不是），也可以是多类别问题（在多个类别中判断输入数据具体属于哪一个类别）。与回归问题（regression）相比，分类问题的输出不再是连续值，而是离散值，用来指定其属于哪个类别。分类问题在现实中应用非常广泛，比如垃圾邮件识别，手写数字识别，人脸识别，语音识别等。

首先思考一个问题，能不能用回归问题的解法去求解分类问题呢？ 以二分类问题为例，对于类别1，我们假设其目标值为1，对于类别2，假设其目标值为-1。在回归问题中，我们需要找到一个函数，使得函数的预测值尽可能的接近目标值。如下图所示，为了方便可视化，假设每个样本只用两维特征表示。



上图中，蓝色的点目标值是1，代表类别1；红色的点目标值是-1，代表类别2。现在需要找到一个函数对这些点进行拟合，如图中绿线所代表的函数b+w1x1+w2x2=0b+w1x1+w2x2=0。对于一个样本点，如果其对应的函数值大于0，则认为其属于类别1，反之属于类别2。这么做看起来好像也是可以的，但现在我们考虑另外一种情况，如下图所示：



此时，在类别1中加入了右下角这些点，如果仍然采用绿色那条线所代表的函数进行预测，这些新加入进来的点的误差将特别的大，为了缓解由此带来的误差，绿色的线将往右下角偏移，以此减少误差。但这么做明显是不符合常理的，误差虽然减少了，但也带来了更严重的分类类别错误问题。造成这个问题的本质是损失函数定义的不恰当，回归问题种将损失函数定位为误差函数是因为回归的目标是尽可能拟合样本点，但分类问题的目标是尽可能将样本点分类到正确的类别中，仍然使用误差函数作为损失函数显然是不合适的。会闹出这样的笑话：对于那些类别明显的点，回归问题求解的惩罚反而更加严重。另外对于多分类问题种，我们给每个类别指定一个对应的目标值，在机器看来这些目标值之间是有联系的，比如：类别1目标值指定为1，类别2目标值指定为2，类别3目标值指定为3……计算机在寻找样本之间的关系时，会默认类别2和类别3要比类别1和类别3更加有关联，因为3和2更加靠近。

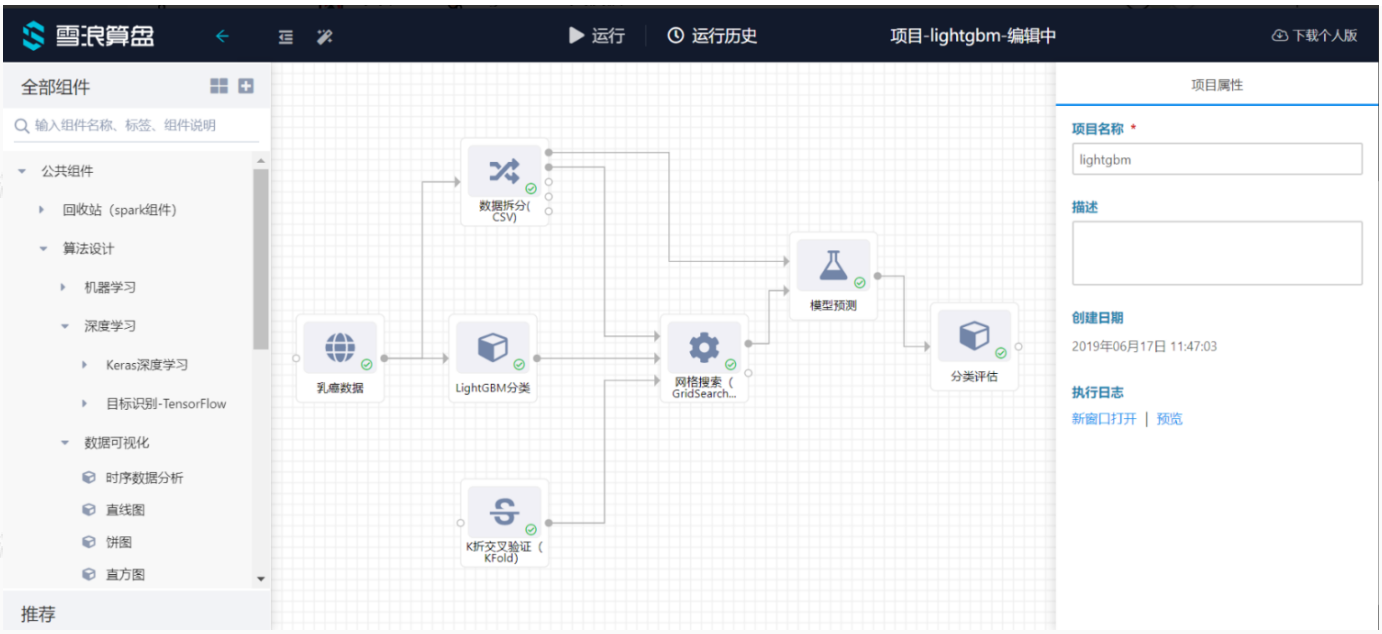
LightGBM原理

LightGBM是一个用于梯度增强机器学习的开源框架。默认情况下，LightGBM将训练一个梯度增强决策树(GBDT)，但它也支持随机森林、DART回归树和基于梯度的单边抽样(Goss)。 该框架速度快，专为分布式训练设计。它支持大规模的数据集和GPU上的训练。在许多情况下，LightGBM被发现比XGBoost更精确、更快，尽管这与需要解决的问题有关。 LightGBM和XGBoost都得到了广泛的应用，并提供了高度优化的、可伸缩的和快速的梯度增强机(GBMs)实现。

Gradient Boosting原理

在考虑集成学习时，主要有两种方法:bagging和boosting。bagging包括对许多独立模型的训练，并通过某种形式的聚合(平均、投票等)将它们的预测组合起来。bagging的一个例子是随机森林。 boosting按顺序训练模型，其中每个模型都从前一个模型的错误中学习。从一个弱基模型开始，对模型进行迭代训练，每个模型都加入到前一个模型的预测中，从而产生一个较强的整体预测。

模板流程



参数设置

数据拆分(CSV)：数据1比重:0.2 数据2比重:0.8

K折交叉验证（KFold）：nSplits:3

网格搜索（GridSearchCV）：Param Grid:{“n_estimators”:[100,200,500,1000]} 记分函数:accuracy

字段选择

特征字段:
radius_mean,texture_mean,perimeter_mean,area,smoothness_mean,compactness_mean,concavity_mean,concave_points_mean,symmetry_mean,fractal_dimension_mean,radius_se,texture_se,perimeter_se,area_se,smoothness_se,compactness_se,concavity_se,concave_points_se,symmetry_se,fractal_dimension_se,radius_worst,texture_worst,perimeter_worst,area_worst,smoothness_worst,compactness_worst,concavity_worst,concave_points_worst,symmetry_worst,fractal_dimension_worst

标识字段: cancer

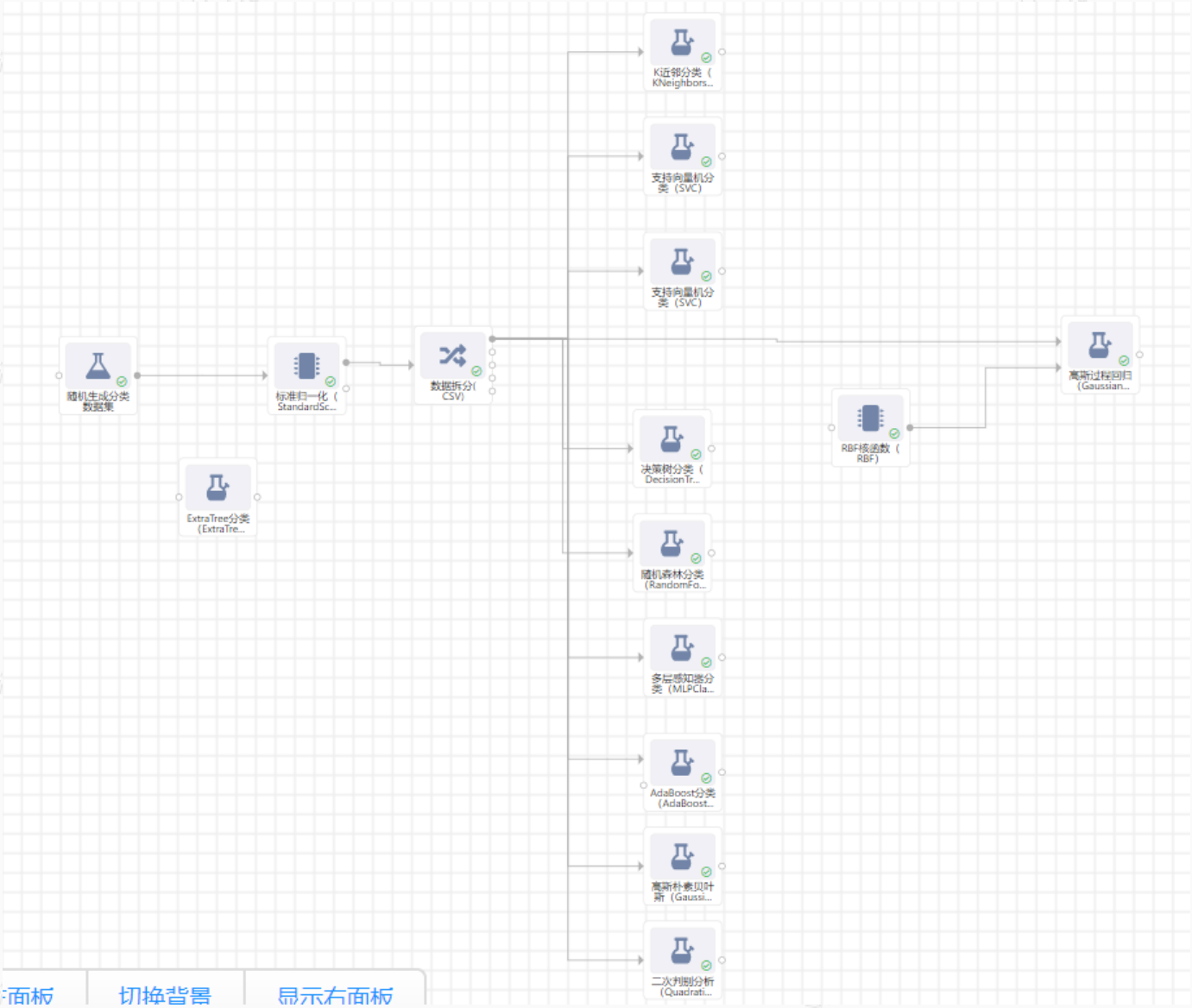
参数设置与字段选择根据不同的数据集灵活设置

分类算法比较案例

分类算法比较案例

项目id：3375

https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-p



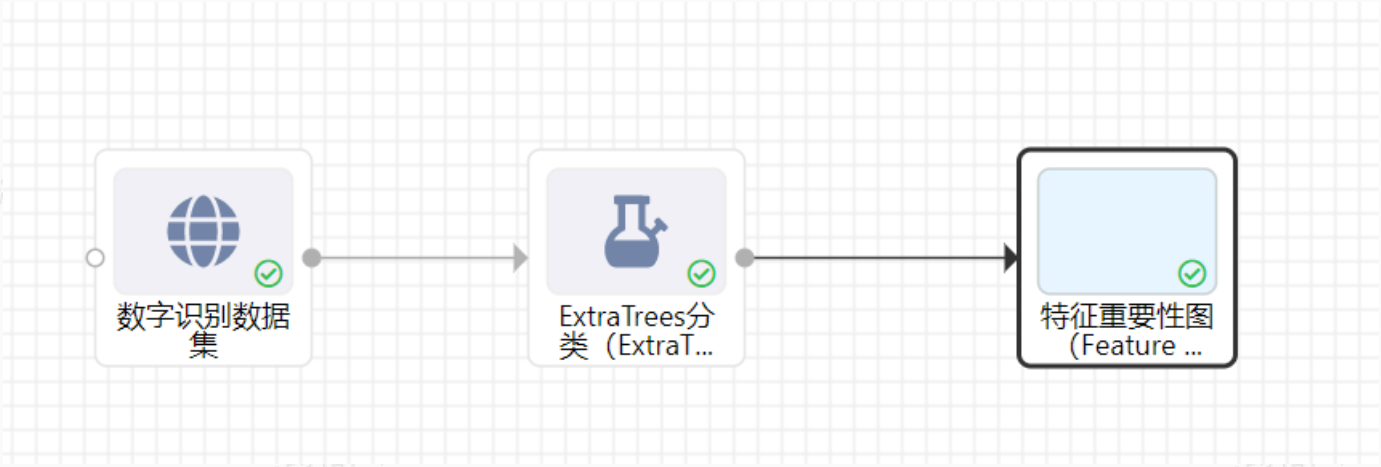
特征重要性

特征重要性

项目id：6229

https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html#sphx-glr-auto-examples-ensemble-plot-forest-importances-py

下图为项目概览：



从图上可以看出，项目分为3个节点：

数据产生节点，产生后面用于训练的数据

模型节点，这里使用了ExtraTrees分类模型，参数为n_estimators=250，random_state=0

特征重要性组件节点，展示特征重要性

LightGBM回归模板

LightGBM回归模板

回归问题

分类和回归的区别在于输出变量的类型。定量输出称为回归，或者说是连续变量预测。

我们做线性回归是为了得到一个最优回归系数向量w使得当我们给定一个x能够通过y=xw预测y的值。假定输入数据存放在矩阵 X 中，而回归系数存在在向量 w 中。那么对于给定的数据X1，预测结果将会通过Y1=XT1w给出。

那么怎样的w才是最优的呢？在标准线性回归中我们需要找到是误差最小的w，即预测的y值与真实的y值之间的差值，为了避免简单累加造成的正负差值相互抵消，这里采用了平方误差:

$$\sum_{i=1}^{m}(y_i - x_i^T w)^2$$

用矩阵可表示为：(y−Xw)T(y−Xw)，)因为要求函数的极小值，对w求导可得:

$$-2X^T(y - Xw)$$

使其等于0，便可求出W的最优解:

$$X^T(y - Xw) = 0$$
$$X^T y = X^T X w$$
$$\hat{w} = (X^T X)^{-1} X^T y$$

上述求w最优解的方法也叫做最小二乘法。

需要注意的是，上述公式中包含(XTX)−1，也就是需要对矩阵求逆，因此这个方程只有在逆矩阵存在的时候有用，所以我们写代码时必须需要事先确定矩阵是否可逆。

此外，我们知道线性回归的方程的一般形式为：y=wx+b；即存在一定的偏移量b，于是，我们可以将回归系数和特征向量均增加一个维度，将函数的偏移值b也算作回归系数的一部分，占据回归系数向量中的一个维度，比如w=(w1,w2,...,wN,b)。相应地，将每条数据特征向量的第一个维度设置为1.0，即所有特征向量的第一个维度值均为1.0,这样，最后得到的回归函数形式为y=w[0]+w[1]*x1+...+w[N]*xN.

LightGBM原理

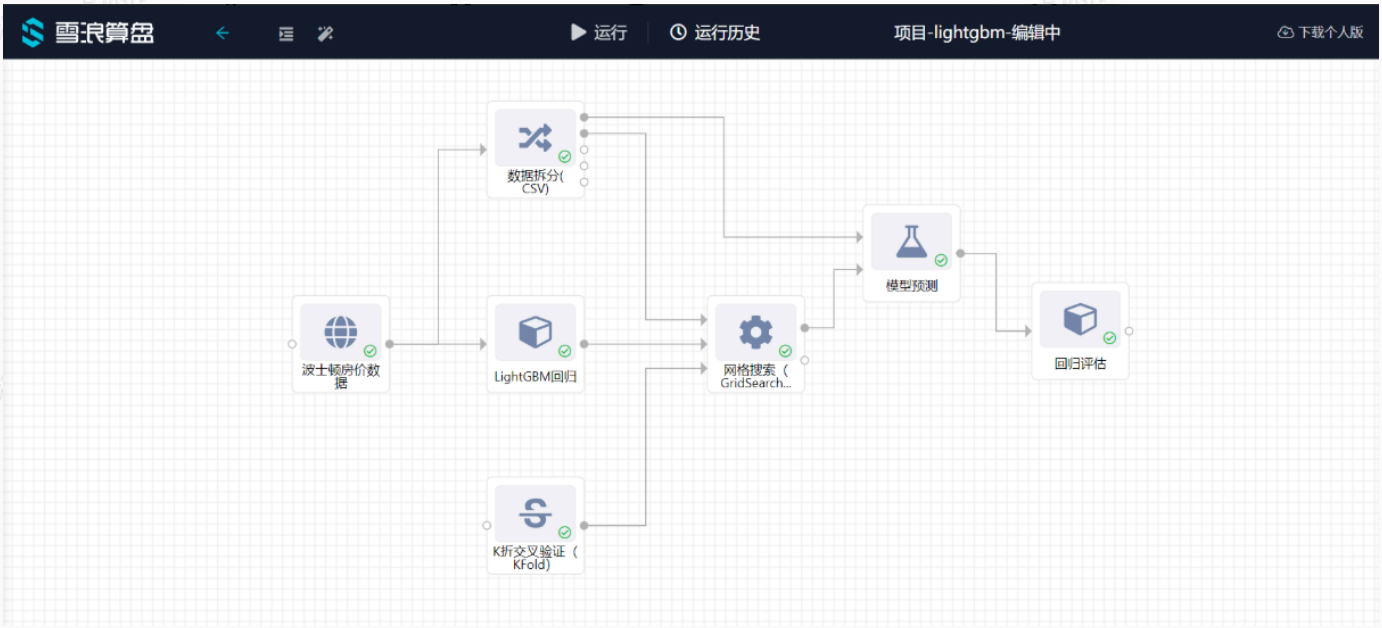
LightGBM是一个用于梯度增强机器学习的开源框架。默认情况下，LightGBM将训练一个梯度增强决策树(GBDT)，但它也支持随机森林、DART回归树和基于梯度的单边抽样(Goss)。 该框架速度快，专为分布式训练设计。它支持大规模的数据集和GPU上的训练。在许多情况下，LightGBM被发现比XGBoost更精

确、更快，尽管这与需要解决的问题有关。 LightGBM和XGBoost都得到了广泛的应用，并提供了高度优化的、可伸缩的和快速的梯度增强机(GBMs)实现。

Gradient Boosting原理

在考虑集成学习时，主要有两种方法:bagging和boosting。bagging包括对许多独立模型的训练，并通过某种形式的聚合(平均、投票等)将它们的预测组合起来。bagging的一个例子是随机森林。 boosting按顺序训练模型，其中每个模型都从前一个模型的错误中学习。从一个弱基模型开始，对模型进行迭代训练，每个模型都加入到前一个模型的预测中，从而产生一个较强的整体预测。

模板流程

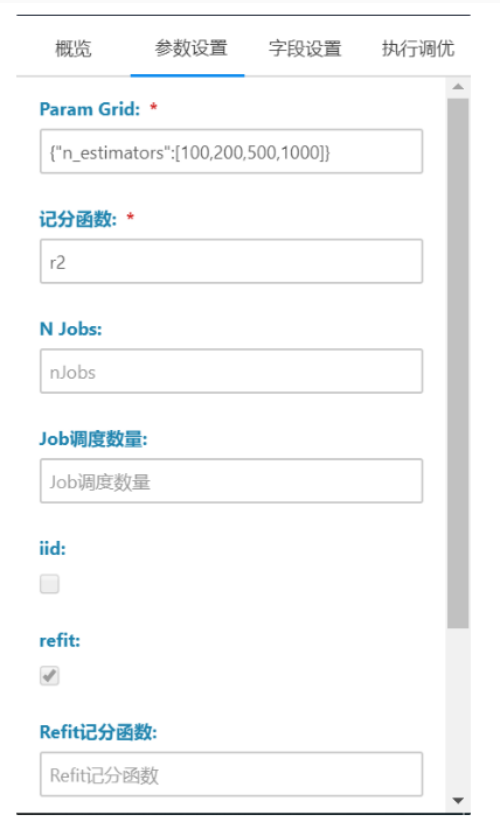


参数设置

数据拆分(CSV)：数据1比重:0.2 数据2比重:0.8

K折交叉验证（KFold）：nSplits:3

网格搜索（GridSearchCV）：Param Grid:{“n_estimators”:[100,200,500,1000]} 记分函数:r2



字段选择

特征字段: CRIM,ZN,INDUS,CHAS,NOX,RM,AGE,DIS,RAD,TAX,PTRATIO,B,LSTAT

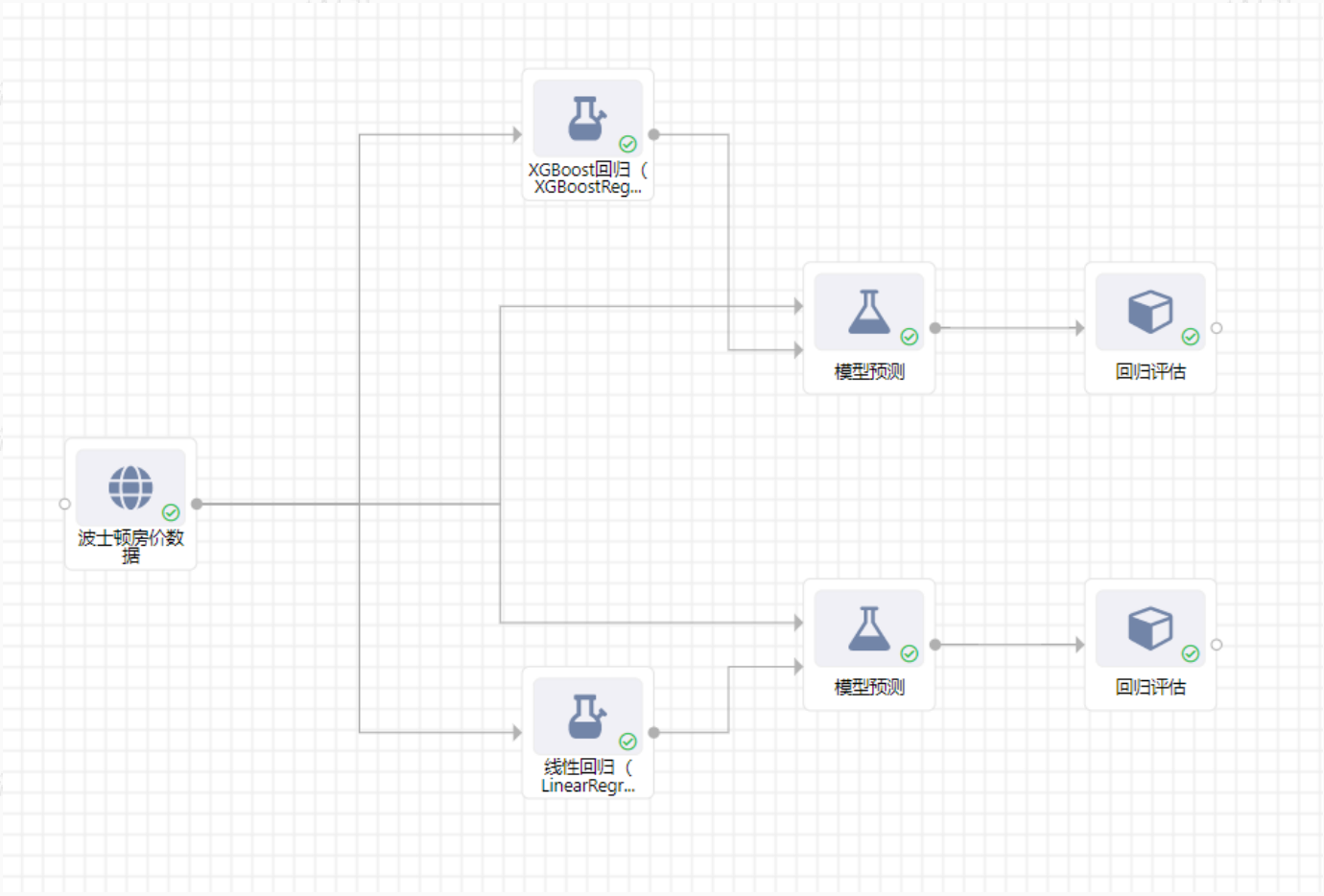
标识字段: MEDV

参数设置与字段选择根据不同的数据集灵活设置

XGBoost回归案例

XGBoost回归案例

项目id：3727



LSTM Network 回归模型

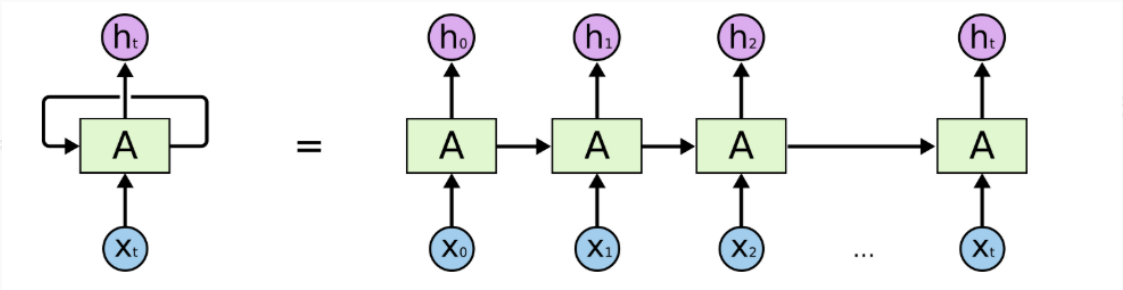
LSTM Network 回归模型

LSTM Network 长短期时间序列模型介绍

核心算法介绍:

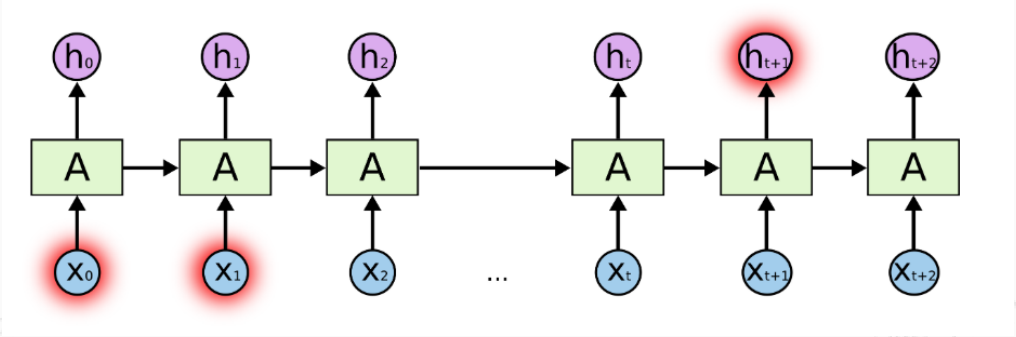
传统的RNN神经网络:

RNN之所以称为循环神经网络，即“一个序列的当前输出与前面的输出是有相关性”。具体实质体现在后面层数的输入值要加入前面层的输出值，即隐藏层之间不再是不相连的而是有连接的。



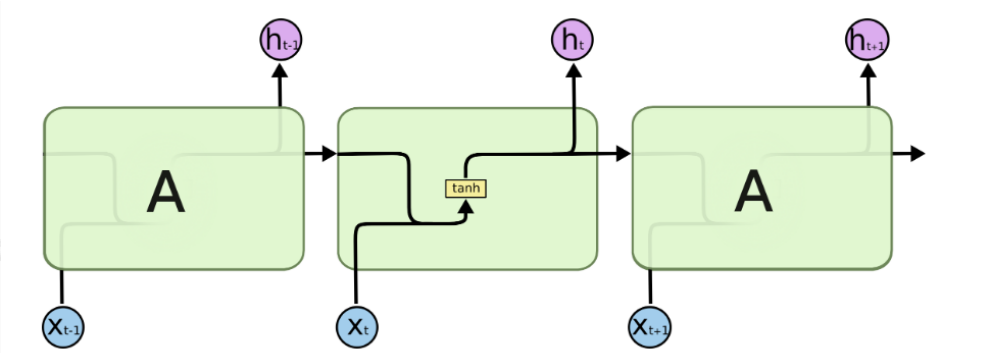
展开的递归循环神经网络

在学习信息情况下，如果相关信息与所需信息之间的差距很小，则RNN可以学习使用过去的信息，表现出较好的训练能力，比如：我们试图预测“云在天空中”的最后一次“天空”,RNN模型就能非常容易的识别出来。但是，如果相关信息与所需要信息之间的距离相差过远时，RNN模型就会很难学习连接这些关系。



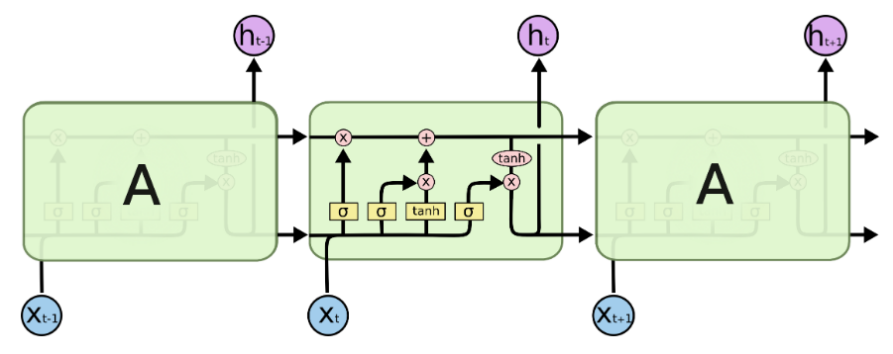
LSTM 神经网络:

长短期时间序列模型(LSTM)是一种特殊的RNN，能够学习长期的依赖性。LSTM的优点就是能够明确旨在避免长期依赖性问题。它能够长时间记住信息，解决了RNN信息距离过长而丧失学习的能力的缺点。所有递归神经网络都具有具有神经网络重复模块链的形式。在标准的RNN中，该重复模块具有非常简单的结构，就是单一的tanh层。



标准RNN中的重复模块包含单个层

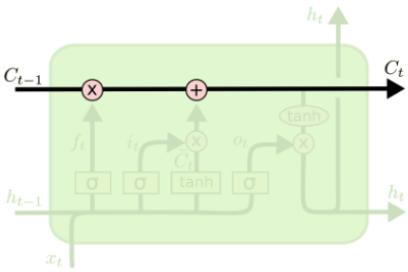
同样LSTM也具有这种类似链的结构，但是重复模块缺失具有不同的结构，它们分别是传输层，遗忘门，输入门，输出门。



LSTM中的重复模块包含四个交互层

1.传输层

传输层水平贯穿图的顶部，直接沿着整个链运行，进行着一些次要的线性交互。从上图中我们可以看出，每个序列索引位置t时刻向前传播中除了和RNN一样具有隐藏状态$h(t)$,还多了另一个隐藏状态，图中上面的横线我们一般称为细胞状态，记为$C(t)$。



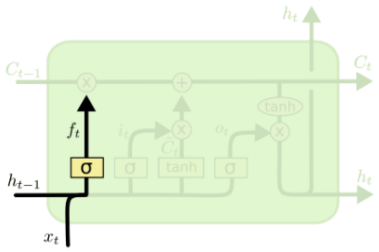
传输层中的细胞状态

2.遗忘门

遗忘门是以一定的概率控制是否遗忘上一层的隐藏细胞状态。图中我们可以看出输入的有上一序列的隐藏状态 $h(t-1)$ 和本序列数据 $x(t)$,通过激活函数sigmoid，得到遗忘门的输出 $f(t)$.由于sigmoid函数的输出 $f(t)$ 是位于[0,1]之间，所以这里的输出 $f(t)$ 就是代表了能否遗忘上一层隐藏细胞状态的概率。 该数字表达可以为:

$$f^{(t)} = \sigma(W_f h^{(t-1)} + U_f x(t) + b_f)$$

其中 *Wf* , *Uf* , *bf* 分别为线性关系的权重和偏移值。



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

遗忘门的体系架构

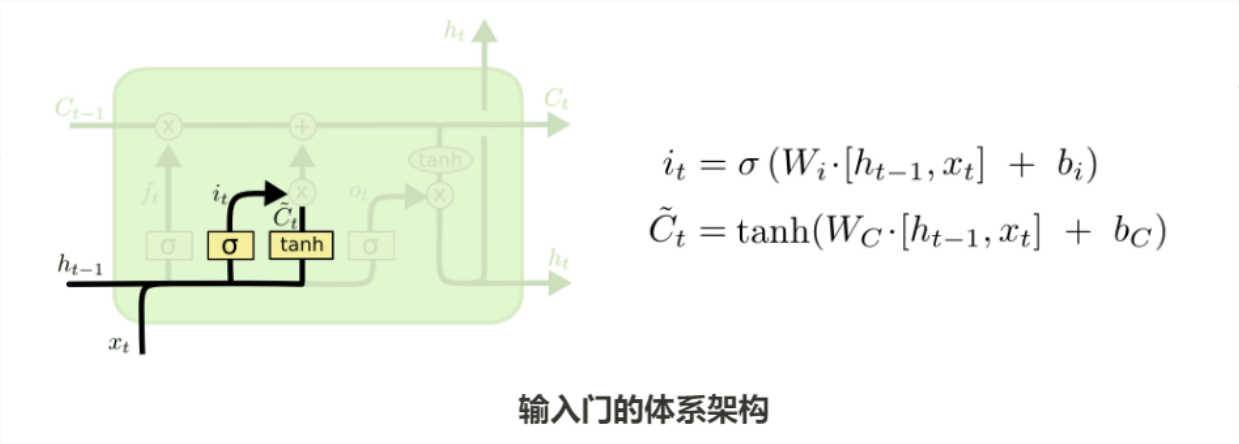
3.输入门

输入门层决定我们在单元状态中存储哪些新的信息。其中有两部分组成，一部分称为“输入门层的sigmoid层”,这层决定我们将更新之前的哪些值，输出为 *i*(*t*)，第二部分为“输入门层的tanh层”,这层决定我们创建新的候选值,输出为 *a*(*t*)，最后结合两个创建状态进行更新。 该数学表达式可以为:

$$\boldsymbol{i^{(t)} = \sigma(W_i h^{(t-1)} + U_i x(t) + b_i)}$$

$$\boldsymbol{a^{(t)} = \tanh(W_a h^{(t-1)} + U_a x(t) + b_a)}$$

其中 *Wi*, *Ui*, *bi*, *Wa*, *Ua*, *ba*分别为线性关系的权重和偏移值。



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

输入门的体系架构

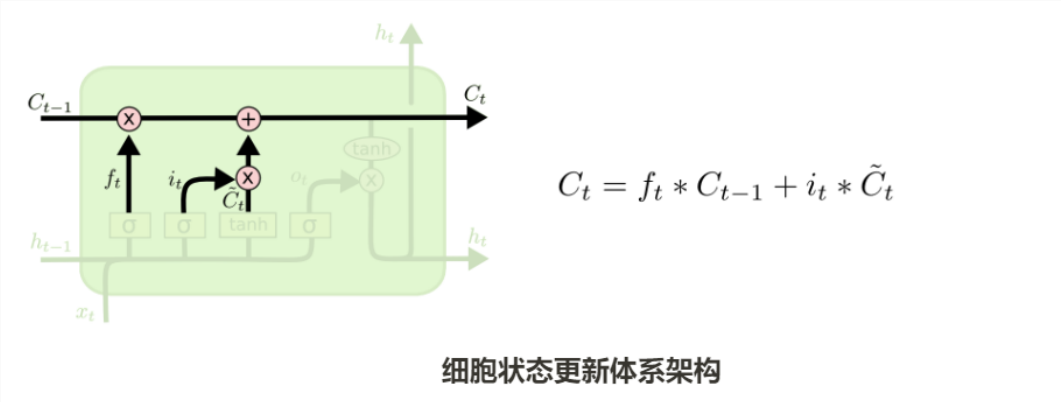
4.细胞状态更新

此过程是决定将旧的细胞状态*C(t−1)*更新至新的细胞状态*C(t)*。在这个过程中首先我们将*ft*乘以*C(t−1)*来决定是否忘记之前的事情。其次我们添加了

$$\boldsymbol{i_t * \tilde{C}_t}$$

来决定我们需要更新多少新的状态值。即:

$$\boldsymbol{C^{(t)} = C^{(t-1)} \odot f^{(t)} + i^{(t)} \odot a^{(t)}}$$



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

细胞状态更新体系架构

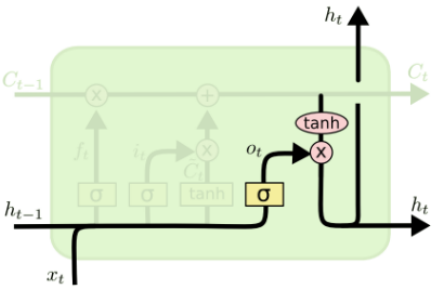
5.输出门

输出门是决定我们要输出的内容。首先，先运行的是一个sigmoid层，决定输出的单元状态属于隐藏状态部分 *h(t−1)*, 还是本序列数据部分 *x(t)*.接着，运行的是tan层，将细胞状态 *Ct* 乘以sigmoid层的输出，从而获取我们所决定的部分。即：

$$o^{(t)} = \sigma(W_o h^{(t-1)} + U_o x^{(t)} + b_o)$$

$$h^{(t)} = o^{(t)} \odot \tanh(C^{(t)})$$

同理其中的 *Wo*, *Uo*, *bo*分别为线性关系的权重和偏移值。



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

输出门的体系架构

Keras LSTM 算法模型：

LSTM模型构建:

1.首先初始化模型Sequential()

2.接着添加LSTM模型层架构，其中*hiddenLayer*代表为隐藏神经元个数，通俗易懂的解释可以为模型中每个sigmoid层或者tanh层。(注意:LSTM训练模型格式矩阵内容[samples,time_steps,features],简单的说就是[n,1,m]架构，即规定了多少特征输入，一个输出)

3.添加模型全连接层

4.编译模型，选择对应的损失函数和优化器，其中常用的损失函数是MSE,MAE一用于回归，binary–crossentropy一用于二值化分类，categorical–crossentropy一用于标签分类。优化的选择我们常用的自适应学习率优化算法 AdaGrad, AdaDelta,Adam。其他类型不建议使用。

```
model = Sequential()
model.add(LSTM(hiddenLayer, input_shape=(train_x.shape[1], train_
x.shape[2])))
model.add(Dense(1))model.compile(loss=loss, optimizer=optimizer, m
etrics=["accuracy"])
```

LSTM模型训练:

将设置好的LSTM模型进行训练，其中*epochs*为循环迭代的次数，*batchsize*即每次循环所走的分支数，*validationdata*即采用交叉验证的方式对数据进行验证

```
# Fit network
checkpointer = ModelCheckpoint(filepath=outputModel, verbose=1, s
ave_best_only=True)
```

```
3
4 history = model.fit(
5 train_x,train_y,
6 epochs=epochs,
7 batch_size=batchSize,
8 validation_data=(test_x, test_y),
9 verbose=2,shuffle=False,callbacks=[checkpointer],)
10
11 logger.info(model.summary())
12 show_picture(history, imageFolder)
```

LSTM模型预测:

模型预测后的得到的结果为[n,1,m]类型，需要重新reshape得到最终的结果

```
1 yhat = model.predict(test_X)
```

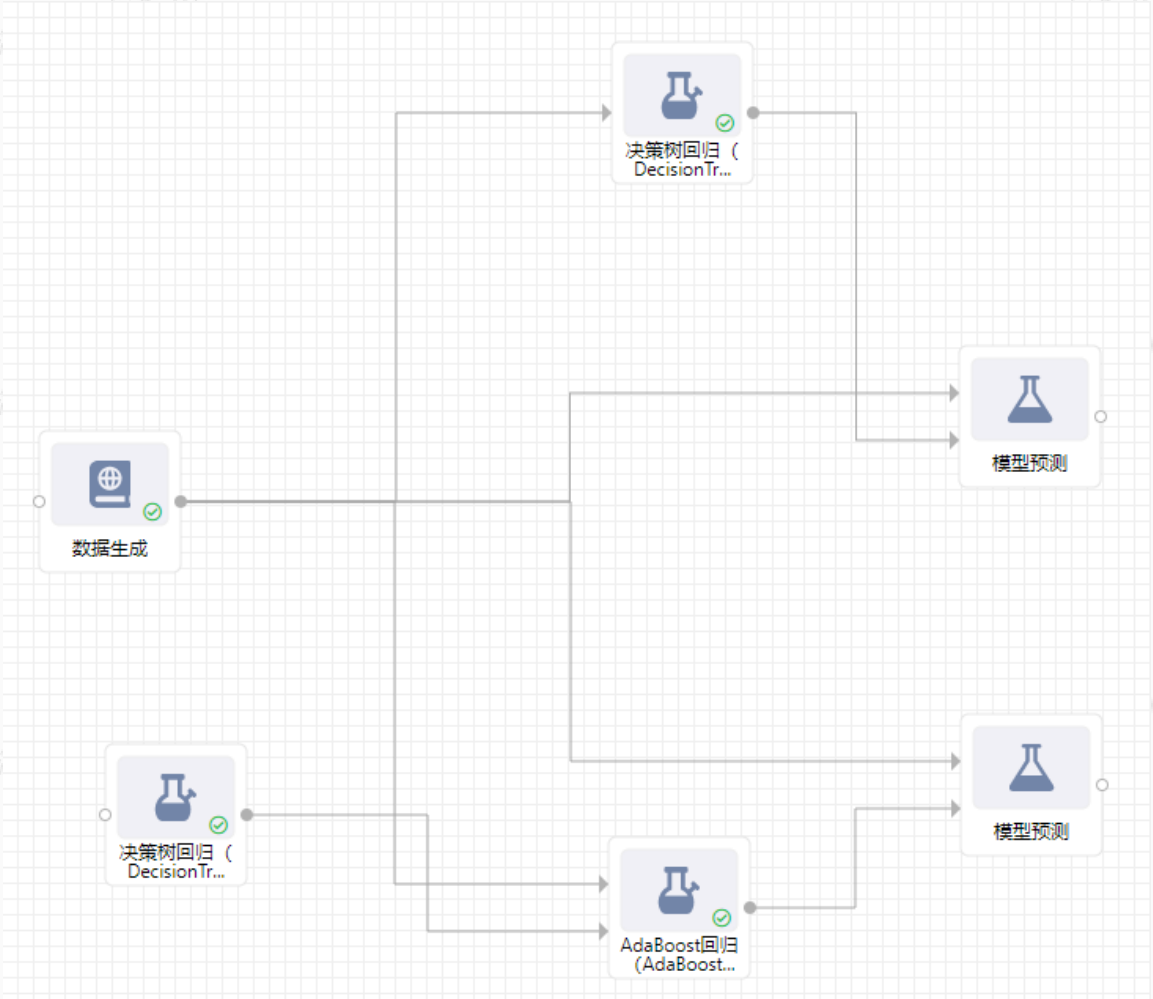
决策树与AdaBoost回归模板

决策树与AdaBoost回归模板

项目id：5347

https://scikit-learn.org/stable/auto_examples/ensemble/plot_adaboost_regression.html#sphx-glr-auto-examples-ensemble-plot-adaboost-regression-py

下图为项目总览：



项目分为三个部分

第一部分，数据生成，生成用于预测的数据

```
rng = np.random.RandomState(1)
X = np.linspace(0, 6, 100)[:, np.newaxis]
y = np.sin(X).ravel() + np.sin(6 * X).ravel() + rng.normal(0, 0.1,
X.shape[0])
```

第二部分，模型训练与预测，这里使用了两个模型，一个是决策树模型，一个是AdaBoost模型
决策树模型：最大深度为4
AdaBoost模型：接收一个最大深度为4的决策树模型，estimator数量为300

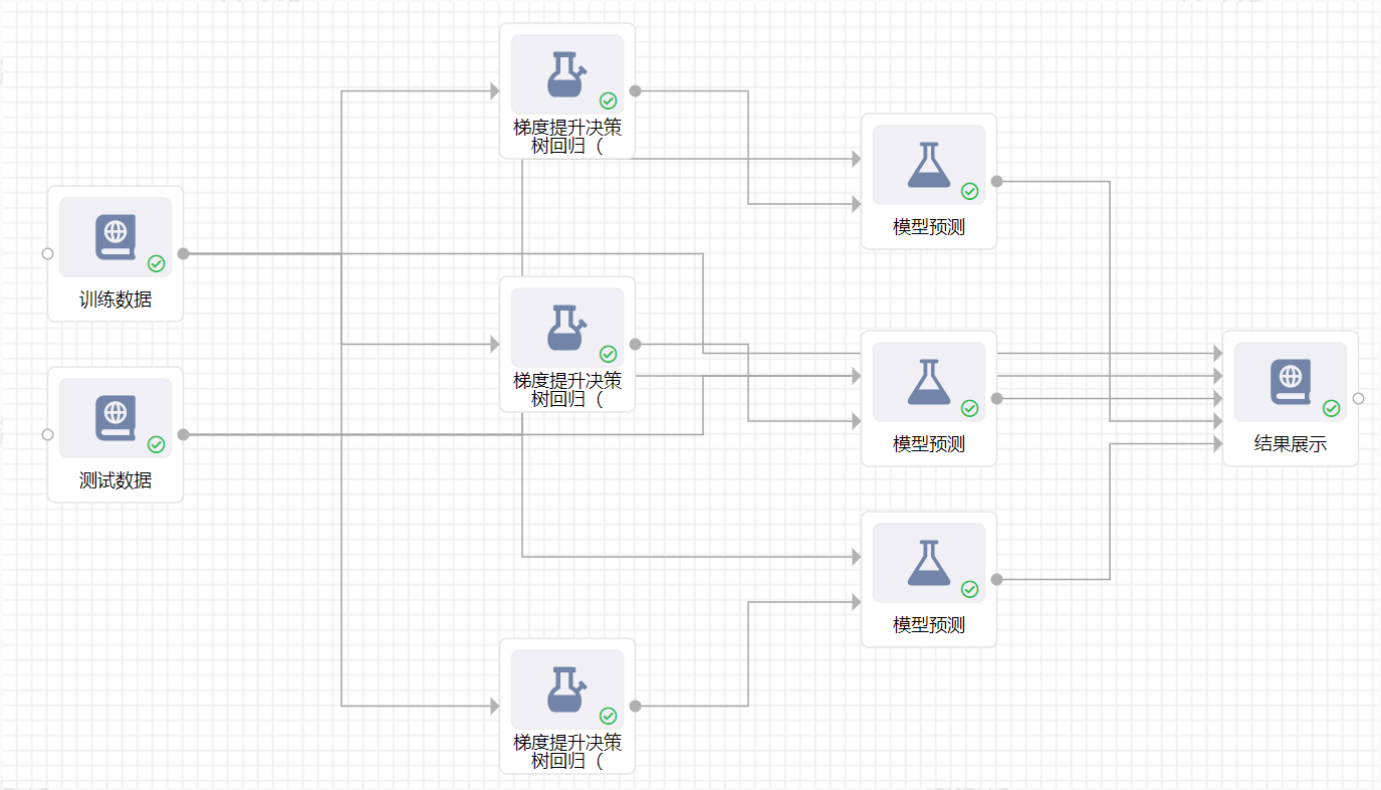
梯度提升回归的预测区间模板

梯度提升回归的预测区间模板

项目id：6543

https://scikit-learn.org/stable/auto_examples/ensemble/plot_gradient_boosting_quantile.html#sphx-glr-auto-examples-ensemble-plot-gradient-boosting-quantile-py

下图为项目总览：



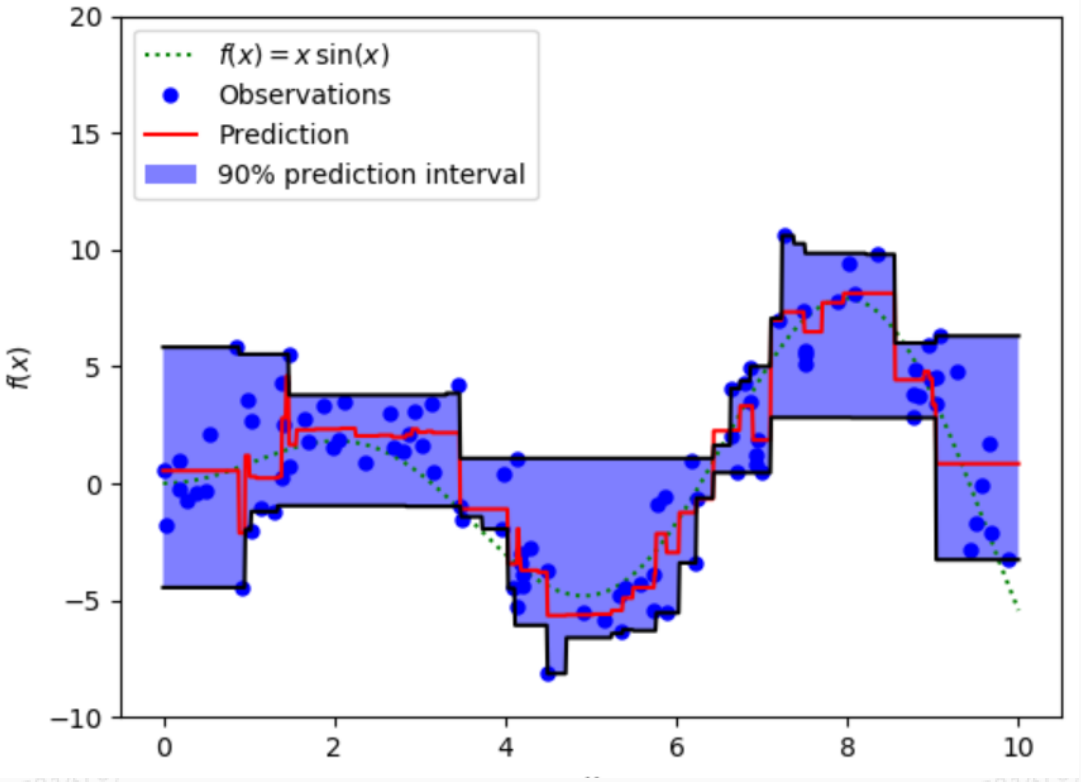
项目分为三个部分

第一部分，数据生成，生成用于训练以及预测的数据

第二部分，模型训练与预测，这里使用了三个模型

第三部分，预测结果分析，展示

最终效果如下：

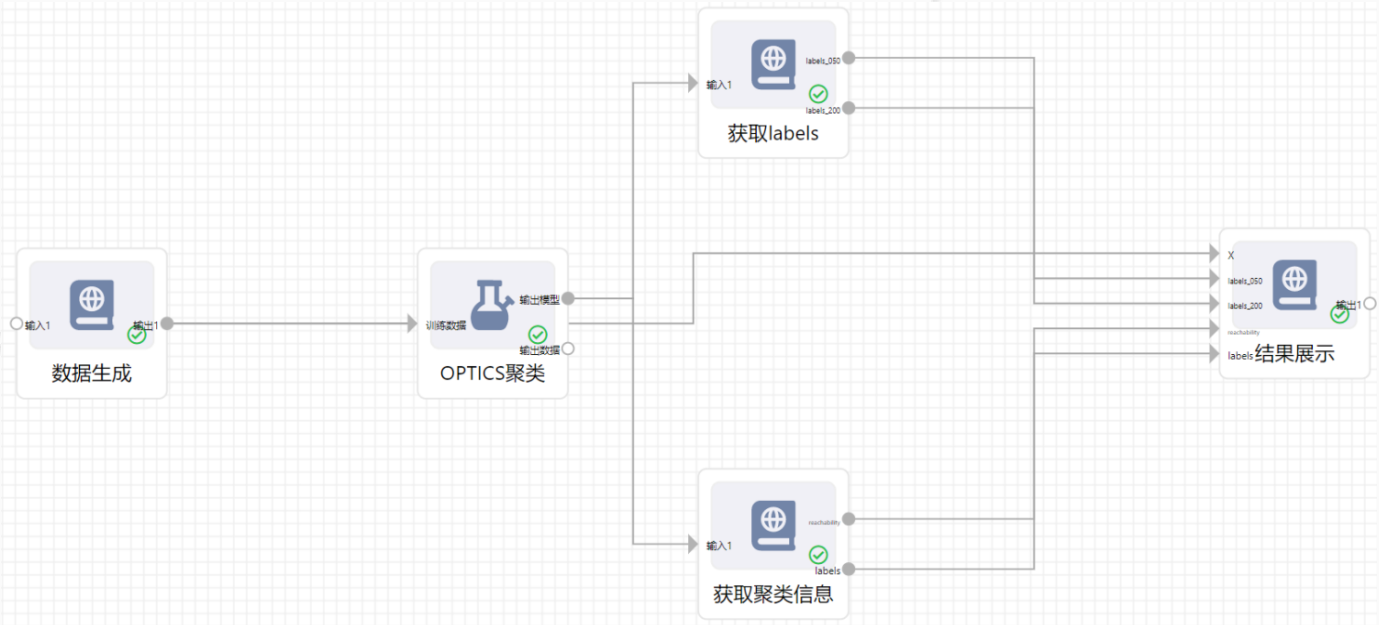


OPTICS聚类算法演示

项目id：6863

https://scikit-learn.org/stable/auto_examples/cluster/plot_optics.html#sphx-glr-auto-examples-cluster-plot-optics-py

下图为项目总览：



项目分为四个部分：

- 第一部分，数据生成，生成用于聚类的数据

```python
import numpy as np
import pandas as pd

import suanpan
from suanpan.app import app
from suanpan.app.arguments import Csv


@app.output(Csv(key="outputData1"))
def Demo(_):
    # Generate sample data
    np.random.seed(0)
    n_points_per_cluster = 250

```

```
    C1 = [-5, -2] + 0.8 * np.random.randn(n_points_per_cluster, 2
)
    C2 = [4, -1] + 0.1 * np.random.randn(n_points_per_cluster, 2)
    C3 = [1, -2] + 0.2 * np.random.randn(n_points_per_cluster, 2)
    C4 = [-2, 3] + 0.3 * np.random.randn(n_points_per_cluster, 2)
    C5 = [3, -2] + 1.6 * np.random.randn(n_points_per_cluster, 2)
    C6 = [5, 6] + 2 * np.random.randn(n_points_per_cluster, 2)
    X = np.vstack((C1, C2, C3, C4, C5, C6))

    print(X.shape)

    return pd.DataFrame(X, columns=["feature_1", "feature_2"])


if __name__ == "__main__":
    suanpan.run(app)
```

- 第二部分，OPTICS聚类，所用参数为min_samples=50，xi=.05, min_cluster_size=.05
- 第三部分，模型信息提取
  - 生成label

```
import joblib
from sklearn.cluster import cluster_optics_dbscan

import suanpan
from suanpan.app import app
from suanpan.app.arguments import File, Npy


def load_model(model_file):
    return joblib.load(model_file)


@app.input(
    File(
        key="inputModel1", alias="inputModel", name="model", type
="model", required=True
    )
```

```
)
@app.output(Npy(key="outputData1"))
@app.output(Npy(key="outputData2"))
def Demo(context):
    args = context.args

    model = load_model(args.inputModel)

    labels_050 = cluster_optics_dbscan(
        reachability=model.reachability_,
        core_distances=model.core_distances_,
        ordering=model.ordering_,
        eps=0.5,
    )
    labels_200 = cluster_optics_dbscan(
        reachability=model.reachability_,
        core_distances=model.core_distances_,
        ordering=model.ordering_,
        eps=2,
    )

    return labels_050, labels_200


if __name__ == "__main__":
    suanpan.run(app)
```

- 获取模型信息

```
import joblib

import suanpan
from suanpan.app import app
from suanpan.app.arguments import File, Npy


def load_model(model_file):
    return joblib.load(model_file)
```

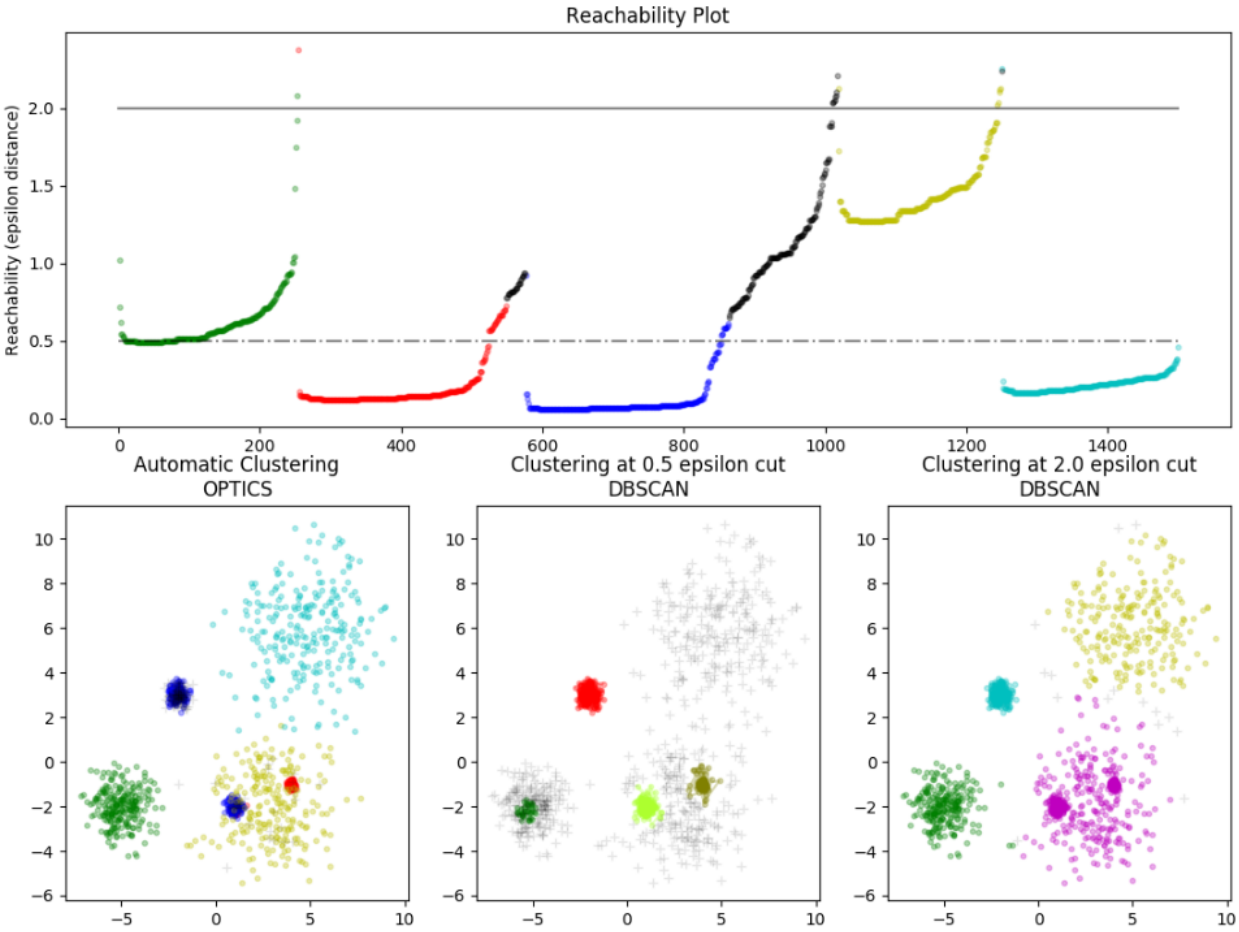
```
10
11
12 @app.input(
13     File(
14         key="inputModel1", alias="inputModel", name="model", type
   ="model", required=True
15     )
16 )
17 @app.output(Npy(key="outputData1"))
18 @app.output(Npy(key="outputData2"))
19 def HelloWorld(context):
20     args = context.args
21
22     model = load_model(args.inputModel)
23     reachability = model.reachability_[model.ordering_]
24     labels = model.labels_[model.ordering_]
25
26     return reachability, labels
27
28
29 if __name__ == "__main__":
30     suanpan.run(app)
```

- 第四部分，结果展示

```
1 import os
2
3 import matplotlib.gridspec as gridspec
4 import numpy as np
5 from matplotlib import pyplot as plt
6
7 import suanpan
8 from suanpan.app import app
9 from suanpan.app.arguments import Csv, Folder, Npy
10
11 TMP_FOLDER = "/tmp/result"
12
13
```

```
@app.input(Csv(key="inputData1"))
@app.input(Npy(key="inputData2"))
@app.input(Npy(key="inputData3"))
@app.input(Npy(key="inputData4"))
@app.input(Npy(key="inputData5"))
@app.output(Folder(key="outputData1"))
def Demo(context):
    args = context.args

    if not os.path.exists(TMP_FOLDER):
        os.makedirs(TMP_FOLDER)

    X = args.inputData1.values
    labels_050 = args.inputData2
    labels_200 = args.inputData3
    reachability = args.inputData4
    labels = args.inputData5
    space = np.arange(len(X))

    plt.figure(figsize=(12.8, 9.6))
    G = gridspec.GridSpec(2, 3)
    ax1 = plt.subplot(G[0, :])
    ax2 = plt.subplot(G[1, 0])
    ax3 = plt.subplot(G[1, 1])
    ax4 = plt.subplot(G[1, 2])

    # Reachability plot
    colors = ["g.", "r.", "b.", "y.", "c."]
    for klass, color in zip(range(0, 5), colors):
        Xk = space[labels == klass]
        Rk = reachability[labels == klass]
        ax1.plot(Xk, Rk, color, alpha=0.3)
    ax1.plot(space[labels == -1], reachability[labels == -1],
"k.", alpha=0.3)
    ax1.plot(space, np.full_like(space, 2.0, dtype=float), "k-",
alpha=0.5)
    ax1.plot(space, np.full_like(space, 0.5, dtype=float), "k-.",
alpha=0.5)
    ax1.set_ylabel("Reachability (epsilon distance)")
    ax1.set_title("Reachability Plot")
```

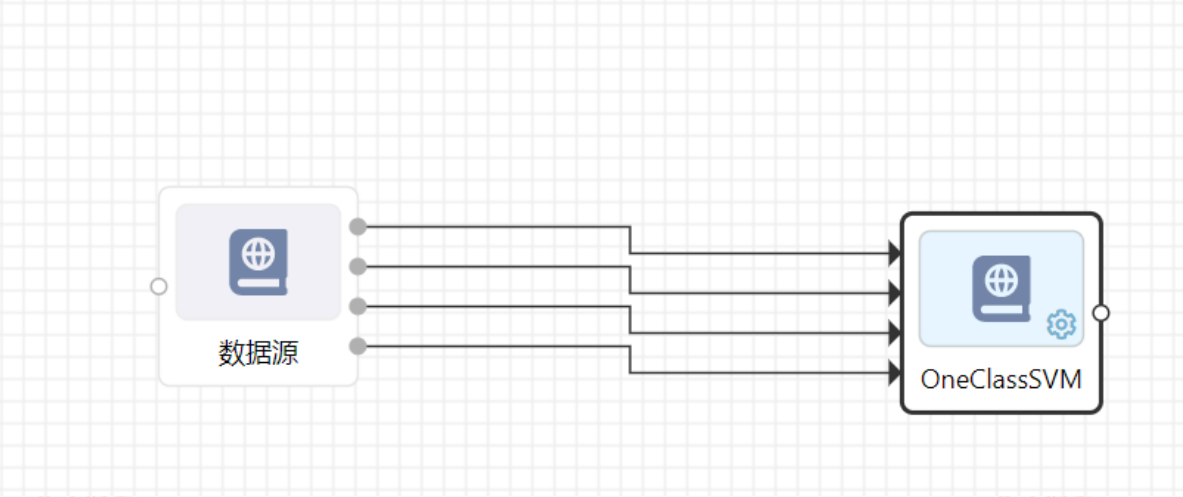
```
51
52      # OPTICS
53      colors = ["g.", "r.", "b.", "y.", "c."]
54      for klass, color in zip(range(0, 5), colors):
55          Xk = X[labels == klass]
56          ax2.plot(Xk[:, 0], Xk[:, 1], color, alpha=0.3)
57      ax2.plot(X[labels == -1, 0], X[labels == -1, 1], "k+", alpha=
    0.1)
58      ax2.set_title("Automatic Clustering\nOPTICS")
59
60      # DBSCAN at 0.5
61      colors = ["g", "greenyellow", "olive", "r", "b", "c"]
62      for klass, color in zip(range(0, 6), colors):
63          Xk = X[labels_050 == klass]
64          ax3.plot(Xk[:, 0], Xk[:, 1], color, alpha=0.3, marker="."
    )
65      ax3.plot(X[labels_050 == -1, 0], X[labels_050 == -1, 1], "k+"
    , alpha=0.1)
66      ax3.set_title("Clustering at 0.5 epsilon cut\nDBSCAN")
67
68      # DBSCAN at 2.
69      colors = ["g.", "m.", "y.", "c."]
70      for klass, color in zip(range(0, 4), colors):
71          Xk = X[labels_200 == klass]
72          ax4.plot(Xk[:, 0], Xk[:, 1], color, alpha=0.3)
73      ax4.plot(X[labels_200 == -1, 0], X[labels_200 == -1, 1], "k+"
    , alpha=0.1)
74      ax4.set_title("Clustering at 2.0 epsilon cut\nDBSCAN")
75
76      plt.savefig(os.path.join(TMP_FOLDER, "optics.png"), format="p
    ng")
77
78      return TMP_FOLDER
79
80
81  if __name__ == "__main__":
82      suanpan.run(app)
83
```



One-class SVM with non-linear kernel (RBF)

One-class SVM with non-linear kernel (RBF)

项目ID：4645



本项目分为2部分，第一部分为数据源，第二部分为OneClassSVM节点
数据源有四个输出，分别为X, X_outliers,xx, yy

- 数据源节点代码如下：

```
# coding=utf-8

import numpy as np

import suanpan
from suanpan.app import app
from suanpan.app.arguments import Npy


# 定义输出
@app.output(Npy(key="outputData1"))
@app.output(Npy(key="outputData2"))
@app.output(Npy(key="outputData3"))
@app.output(Npy(key="outputData4"))
def Demo(context):
```

```
    xx, yy = np.meshgrid(np.linspace(-5, 5, 500), np.linspace(-5, 5, 500))
    # Generate train data
    X = 0.3 * np.random.randn(100, 2)
    # Generate some abnormal novel observations
    X_outliers = np.random.uniform(low=-4, high=4, size=(20, 2))

    return X, X_outliers, xx, yy


if __name__ == "__main__":
    suanpan.run(app)
```

- OneClassSVM节点代码如下：

```
# coding=utf-8

import os

import matplotlib.font_manager
import matplotlib.pyplot as plt
import numpy as np

import suanpan
from sklearn import svm
from suanpan.app import app
from suanpan.app.arguments import Folder, Npy

TMP_FOLDER = "/tmp/result"

# 定义输入
@app.input(Npy(key="inputData1", required=True))
@app.input(Npy(key="inputData2", required=True))
@app.input(Npy(key="inputData3", required=True))
@app.input(Npy(key="inputData4", required=True))
# 定义输出
@app.output(Folder(key="outputData1", required=True))
def Demo(context):
    args = context.args
```

```
25
26      if not os.path.exists(TMP_FOLDER):
27          os.makedirs(TMP_FOLDER)
28
29      X_outliers = args.inputData2
30      xx = args.inputData3
31      yy = args.inputData4
32
33      X = 0.3 * np.random.randn(100, 2)
34      X_train = np.r_[X + 2, X - 2]
35      # Generate some regular novel observations
36      X = 0.3 * np.random.randn(20, 2)
37      X_test = np.r_[X + 2, X - 2]
38
39      clf = svm.OneClassSVM(nu=0.1, kernel="rbf", gamma=0.1)
40      clf.fit(X_train)
41      y_pred_train = clf.predict(X_train)
42      y_pred_test = clf.predict(X_test)
43      y_pred_outliers = clf.predict(X_outliers)
44      n_error_train = y_pred_train[y_pred_train == -1].size
45      n_error_test = y_pred_test[y_pred_test == -1].size
46      n_error_outliers = y_pred_outliers[y_pred_outliers == 1].size
47
48      Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
49      Z = Z.reshape(xx.shape)
50
51      plt.title("Novelty Detection")
52      plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), 0, 7), cm
    ap=plt.cm.PuBu)
53      a = plt.contour(xx, yy, Z, levels=[0], linewidths=2, colors=
    "darkred")
54      plt.contourf(xx, yy, Z, levels=[0, Z.max()], colors="paleviol
    etred")
55
56      s = 40
57      b1 = plt.scatter(X_train[:, 0], X_train[:, 1], c="white", s=s
    , edgecolors="k")
58      b2 = plt.scatter(X_test[:, 0], X_test[:, 1], c="blueviolet",
    s=s, edgecolors="k")
59      c = plt.scatter(X_outliers[:, 0], X_outliers[:, 1], c="gold",
```

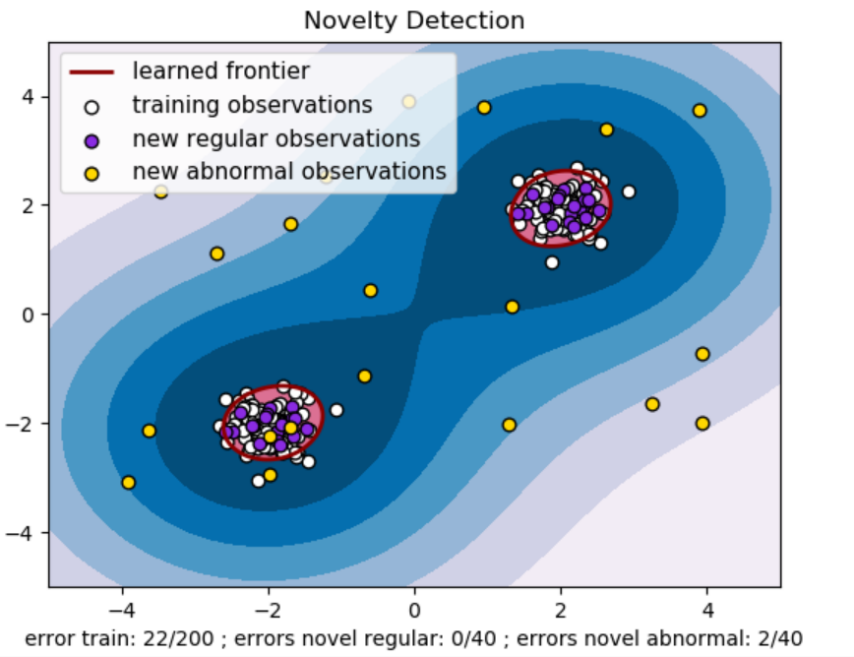
```
    s=s, edgecolors="k")
    plt.axis("tight")
    plt.xlim((-5, 5))
    plt.ylim((-5, 5))
    plt.legend(
        [a.collections[0], b1, b2, c],
        [
            "learned frontier",
            "training observations",
            "new regular observations",
            "new abnormal observations",
        ],
        loc="upper left",
        prop=matplotlib.font_manager.FontProperties(size=11),
    )
    plt.xlabel(
        "error train: %d/200 ; errors novel regular: %d/40 ; "
        "errors novel abnormal: %d/40" % (n_error_train, n_error_test, n_error_outliers)
    )
    plt.savefig(os.path.join(TMP_FOLDER, "test.png"), format="png")

    return TMP_FOLDER


if __name__ == "__main__":
    suanpan.run(Demo)
```

1.异常点检测节点，接受上一个节点的四个输入，输出图片结果

第一步，创建OneClassSVM对象，使用前一个节点的数据进行训练

第二步，模型预测异常点

第三步，画出图形



拉普拉斯映射LE

拉普拉斯映射LE

1、介绍

拉普拉斯特征映射（Laplacian Eigenmaps）是一种不太常见的降维算法，它看问题的角度和常见的降维算法不太相同，是从局部的角度去构建数据之间的关系。也许这样讲有些抽象，具体来讲，拉普拉斯特征映射是一种基于图的降维算法，它希望相互间有关系的点（在图中相连的点）在降维后的空间中尽可能的靠近，从而在降维后仍能保持原有的数据结构。

2、推导

拉普拉斯特征映射通过构建邻接矩阵为（邻接矩阵定义见这里）的图来重构数据流形的局部结构特征。其主要思想是，如果两个数据实例和很相似，那么和在降维后目标子空间中应该尽量接近。设数据实例的数目为，目标子空间即最终的降维目标的维度为。定义大小的矩阵，其中每一个行向量是数据实例在目标维子空间中的向量表示（即降维后的数据实例）。我们的目的是让相似的数据样例和降维后的目标子空间里仍旧尽量接近，故拉普拉斯特征映射优化的目标函数如下：

下面开始推导：

其中是图的邻接矩阵，对角矩阵是图的度矩阵，成为图的拉普拉斯矩阵。

变换后的拉普拉斯特征映射优化的目标函数如下：

其中限制条件保证优化问题有解，下面用拉格朗日乘子法对目标函数求解：

其中用到了矩阵的迹的求导，具体方法见迹求导。为一个对角矩阵，另外、均为实对称矩阵，其转置与自身相等。对于单独的向量，上式可写为: ,这是一个广义特征值问题。通过求得个最小非零特征值所对应的特征向量，即可达到降维的目的。

关于这里为什么要选择个个最小（PCA选择最大）非零特征值所对应的特征向量，将带回到中，由于有着约束条件的限制，可以得到。即为特征值之和。我们为了目标函数最小化，要选择最小的mm个特征值所对应的特征向量。

3、步骤

使用时算法具体步骤为：

步骤1：构建图

使用某一种方法来将所有的点构建成一个图，例如使用KNN算法，将每个点最近的K个点连上边。K是一个预先设定的值。

步骤2：确定权重

确定点与点之间的权重大小，例如选用热核函数来确定，如果点和点相连，那么它们关系的权重设定为：

步骤3：特征映射

计算拉普拉斯矩阵L的特征向量与特征值：
使用最小的个非零特征值对应的特征向量作为降维后的结果输出。

4、核心代码
KNN算法：

```
def knn(inX, dataSet, kNeighbour):
"""KNN algorithm.
Arguments:
inX {np.array} -- data of i
dataSet np.array} -- data
k_neighbour {Int} -- the k near number of Knn
Returns:
sortedDistIndicies -- Index of points adjacent to i
"""
dataSetSize = dataSet.shape[0]
diffMat = tile(inX, (dataSetSize, 1)) - dataSet
sqDiffMat = np.array(diffMat) ** 2
sqDistances = sqDiffMat.sum(axis=1)
distances = sqDistances ** 0.5
sortedDistIndicies = distances.argsort()
return sortedDistIndicies[0:kNeighbour]
```

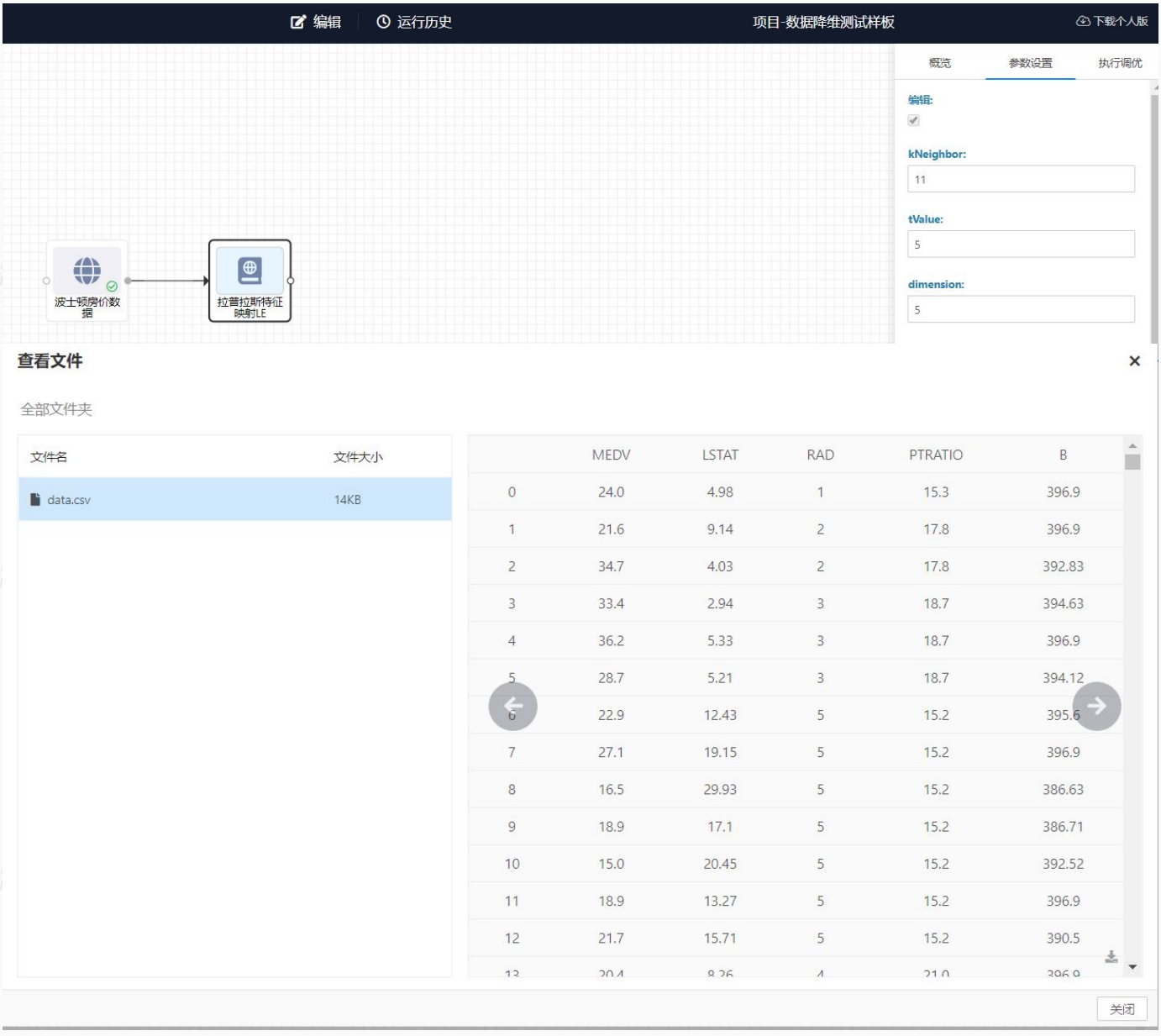
LE算法:

```
def laplaEigen(data, k_neighbour, t_value):
"""feature function.
Arguments:
data {np.array} -- data
k_neighbour {Int} -- the k near number of Knn
t_value {int} -- w Weights need parameter
Returns:
feature_value -- Eigenvalues.
featur_vector -- Eigenvector.
"""
m = np.shape(data)[0]
W = np.mat(np.zeros([m, m]))
D = np.mat(np.zeros([m, m]))
for i in range(m):
k_index = knn(data[i, :], data, k_neighbour)
for j in range(k_neighbour):
sqDiffVector = data[i, :] - data[k_index[j], :]
sqDiffVector = np.array(sqDiffVector) ** 2
```

```
sqDistances = sqDiffVector.sum()
W[i, k_index[j]] = np.math.exp(-sqDistances / t_value)
D[i, i] += W[i, k_index[j]]
L = D - W
X = np.dot(D.I, L)
# Calculate eigenvalues and eigenvectors of a matrix
feature_value, feature_vector = np.linalg.eig(X)
return feature_value, feature_vector
```

5、执行结果

选取KNN临近点个数kNeighbot,W权重的参数点tvalue和降维后的维数dimension.此次测试针对波士顿房价中的14个特征参数进行降维，最后选取最佳的5个特征值作为降维后的数据特征。



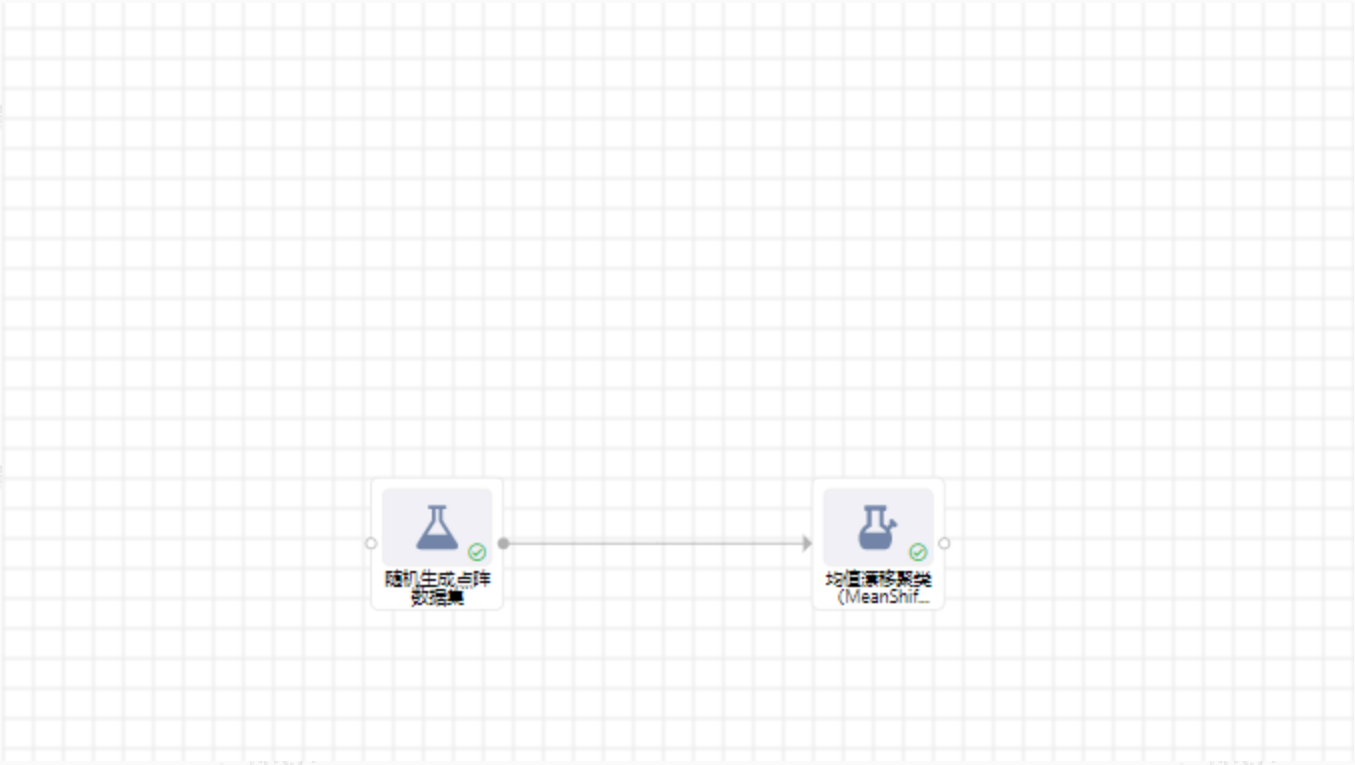
降维后选取的最佳的5个特征值数据集

MeanShift模板

MeanShift模板

项目id：3728

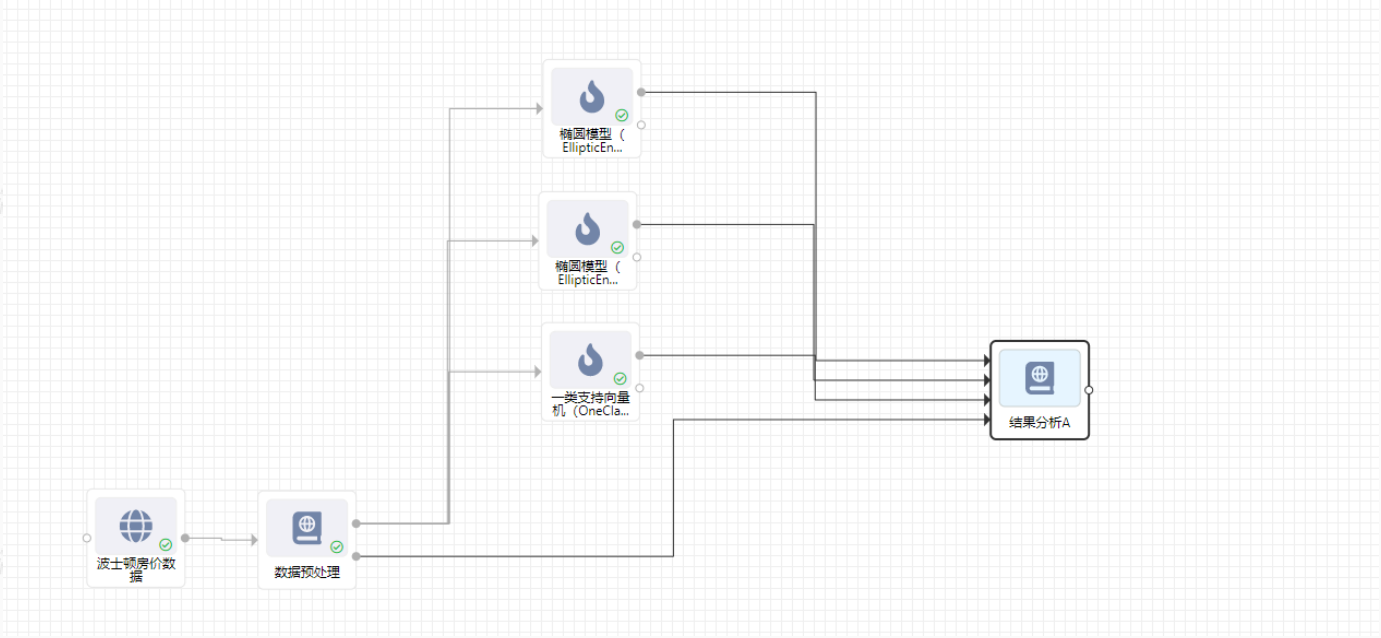
https://scikit-learn.org/stable/auto_examples/cluster/plot_mean_shift.html#sphx-glr-auto-examples-cluster-plot-mean-shift-py



异常点检测

异常点检测

项目id：5784
https://scikit-learn.org/stable/auto_examples/applications/plot_outlier_detection_housing.html#sphx-glr-auto-examples-applications-plot-outlier-detection-housing-py
下图为项目总览：



项目分为三个部分：
第一部分，数据生成，生成用于预测的数据，这里使用的是波士顿房价的数据，再进行了预处理

- 数据预处理节点代码

```
import suanpan
from suanpan.app import app
from suanpan.app.arguments import Csv, Npy
import pandas as pd

def transformDF(df, startIndex, stopIndex):
```

```
    df = df.values[:, [startIndex, stopIndex]]
    columns = df.shape[1]
    columns = ["feature_{}".format(str(index)) for index in range(columns)]
    return pd.DataFrame(df, columns=columns)


@app.input(Csv(key="inputData1"))
@app.output(Csv(key="outputData1"))
@app.output(Csv(key="outputData2"))
def Demo(context):
    args = context.args

    df = args.inputData1

    return transformDF(df, 8, 10), transformDF(df, 5, 12)


if __name__ == "__main__":
    suanpan.run(app)
```

第二部分，模型训练与预测，这里针对两种数据，分别使用了三种模型
第三部分，分析模型的结果，绘图

- 结果分析
  - 代码

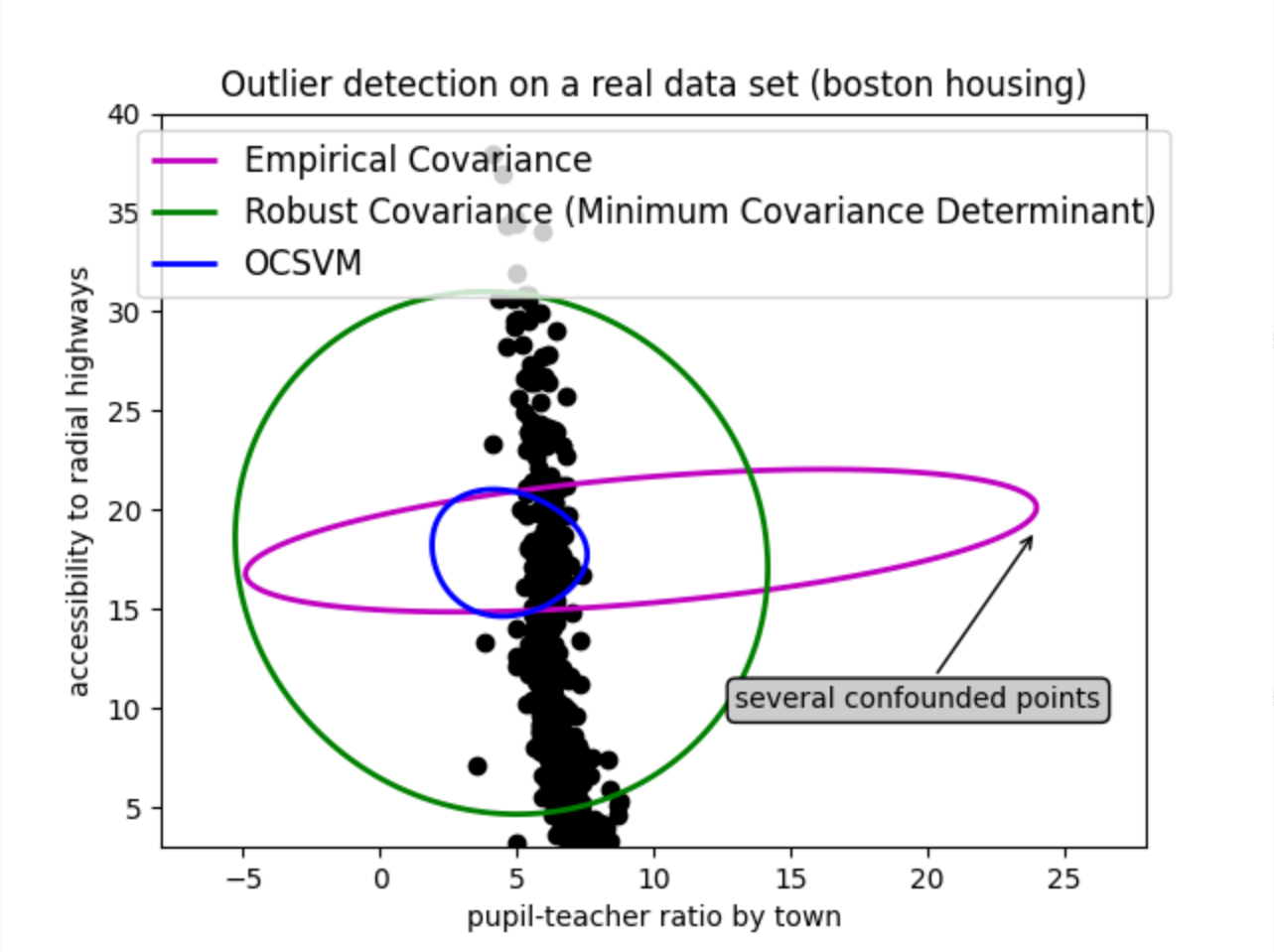
```
import os

import joblib
import matplotlib.font_manager
import matplotlib.pyplot as plt
import numpy as np

import suanpan
from suanpan.app import app
from suanpan.app.arguments import Csv, File, Folder, Model, String

TMP_FOLDER = "/tmp/result"
```

```
13
14
15 @app.input(File(key="inputModel1", name="model", type="model"))
16 @app.input(File(key="inputModel2", name="model", type="model"))
17 @app.input(File(key="inputModel3", name="model", type="model"))
18 @app.input(Csv(key="inputData4", required=True))
19 @app.output(Folder(key="outputData1"))
20 def Demo(context):
21     args = context.args
22
23     if not os.path.exists(TMP_FOLDER):
24         os.makedirs(TMP_FOLDER)
25
26     X = args.inputData4.values
27     classifiers = {
28         "Empirical Covariance": joblib.load(args.inputModel1),
29         "Robust Covariance (Minimum Covariance Determinant)": job
   lib.load(
30             args.inputModel2
31         ),
32         "OCSVM": joblib.load(args.inputModel3),
33     }
34     colors = ["m", "g", "b"]
35     legend = {}
36
37     # Learn a frontier for outlier detection with several classif
   iers
38     xx1, yy1 = np.meshgrid(np.linspace(-8, 28, 500), np.linspace(
   3, 40, 500))
39     xx2, yy2 = np.meshgrid(np.linspace(3, 10, 500), np.linspace(-
   5, 45, 500))
40
41     for i, (clf_name, clf) in enumerate(classifiers.items()):
42         plt.figure(1)
43         Z1 = clf.decision_function(np.c_[xx1.ravel(), yy1.ravel()
   ])
44         Z1 = Z1.reshape(xx1.shape)
45         legend[clf_name] = plt.contour(
46             xx1, yy1, Z1, levels=[0], linewidths=2, colors=colors
   [i]
```

```
        )

    legend_values_list = list(legend.values())
    legend_keys_list = list(legend.keys())

    # Plot the results (= shape of the data points cloud)
    plt.figure(1)  # two clusters
    plt.title("Outlier detection on a real data set (boston housing)")
    plt.scatter(X[:, 0], X[:, 1], color="black")
    bbox_args = dict(boxstyle="round", fc="0.8")
    arrow_args = dict(arrowstyle="->")
    plt.annotate(
        "several confounded points",
        xy=(24, 19),
        xycoords="data",
        textcoords="data",
        xytext=(13, 10),
        bbox=bbox_args,
        arrowprops=arrow_args,
    )
    plt.xlim((xx1.min(), xx1.max()))
    plt.ylim((yy1.min(), yy1.max()))
    plt.legend(
        (
            legend_values_list[0].collections[0],
            legend_values_list[1].collections[0],
            legend_values_list[2].collections[0],
        ),
        (legend_keys_list[0], legend_keys_list[1], legend_keys_list[2]),
        loc="upper center",
        prop=matplotlib.font_manager.FontProperties(size=12),
    )
    plt.ylabel("accessibility to radial highways")
    plt.xlabel("pupil-teacher ratio by town")
    plt.savefig(os.path.join(TMP_FOLDER, "demo_result.png"), format="png")

    return TMP_FOLDER
```

```
84
85
86 if __name__ == "__main__":
87     suanpan.run(app)
```

- 效果图



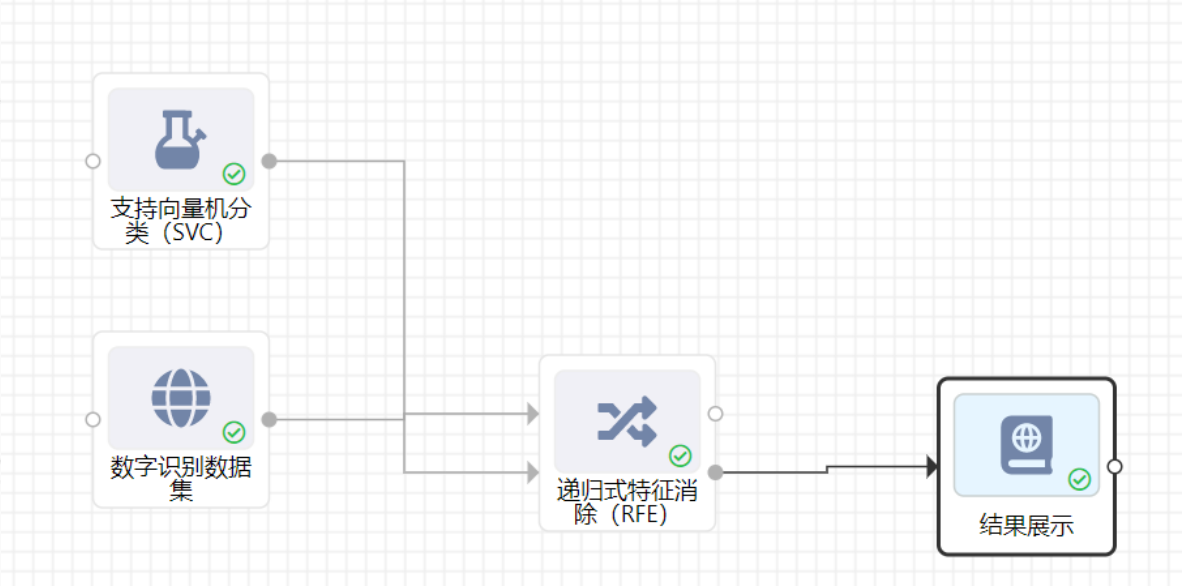
递归特征消除

递归特征消除

项目id：6227

https://scikit-learn.org/stable/auto_examples/feature_selection/plot_rfe_digits.html#sphx-glr-auto-examples-feature-selection-plot-rfe-digits-py

下图为项目总览：



项目有四个节点组成：

数据加载：数据生成，生成用于训练的数据

模型定义：这里使用了SVC支持向量分类模型，参数为C=1，kernel=“linear”

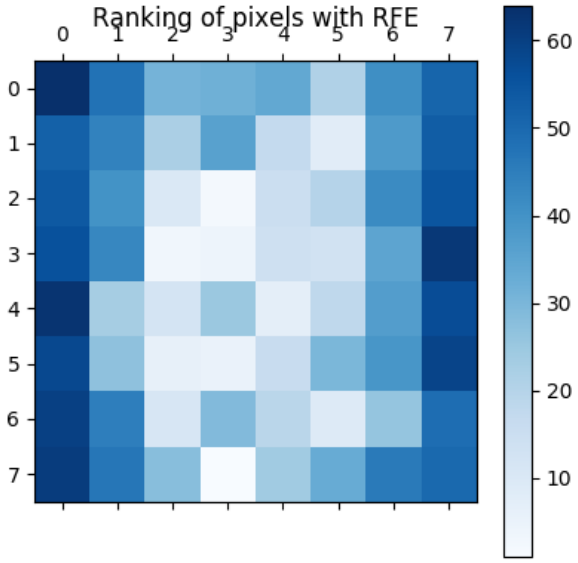
递归特征消除组件设置：设置保留一个特征

结果展示

```
import os

import joblib
from matplotlib import pyplot as plt

import suanpan
from arguments import SklearnModel
from suanpan.app import app
from suanpan.app.arguments import File, Folder

```

```python
TMP_FOLDER = "/tmp/result"


@app.input(File(key="inputModel1", name="model", type="model"))
@app.output(Folder(key="outputData1"))
def HelloWorld(context):
    args = context.args

    if not os.path.exists(TMP_FOLDER):
        os.makedirs(TMP_FOLDER)

    model = joblib.load(args.inputModel1)
    ranking = model.ranking_.reshape(8, 8)

    # Plot pixel ranking
    plt.matshow(ranking, cmap=plt.cm.Blues)
    plt.colorbar()
    plt.title("Ranking of pixels with RFE")

    plt.savefig(os.path.join(TMP_FOLDER, "result.png"), format="png")

    return TMP_FOLDER


if __name__ == "__main__":
    suanpan.run(app)
```

最终结果如下：



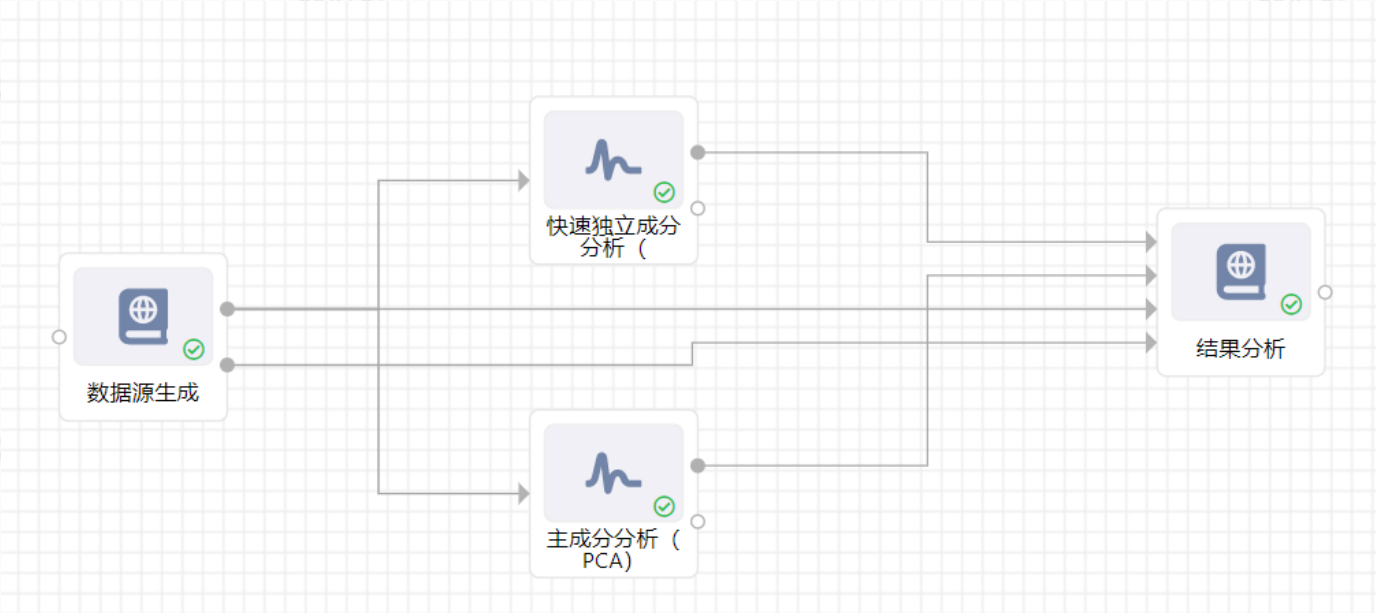
使用FastICA进行数据源分离

使用FastICA进行数据源分离

项目ID：4645

https://scikit-learn.org/stable/auto_examples/decomposition/plot_ica_blind_source_separation.html#sphx-glr-auto-examples-decomposition-plot-ica-blind-source-separation-py

项目总览：



项目分为三个部分：

第一部分，数据生成，生成用于分离的数据源

第二部分，进行PCA以及ICA分析

第三部分，分析结果，绘图

- 结果分析节点代码

```python
import os

import matplotlib.pyplot as plt

import suanpan
from arguments import SklearnModel
from suanpan.app import app
from suanpan.app.arguments import Csv, Folder, Npy
```

```python

TMP_FOLDER = "/tmp/result"

@app.input(Csv(key="inputData1", alias="icaData"))
@app.input(Csv(key="inputData2", alias="pcaData"))
@app.input(Csv(key="inputData3", alias="X"))
@app.input(Npy(key="inputData4", alias="S"))
@app.output(Folder(key="outputData1"))
def Demo(context):
    args = context.args

    if not os.path.exists(TMP_FOLDER):
        os.makedirs(TMP_FOLDER)

    S_ = args.icaData.values
    H = args.pcaData.values
    X = args.X.values
    S = args.S
    imageFile = args.outputData1

    plt.figure(figsize=(14.4, 9.6))
    models = [X, S, S_, H]

    names = [
        "Observations (mixed signal)",
        "True Sources",
        "ICA recovered signals",
        "PCA recovered signals",
    ]
    colors = ["red", "steelblue", "orange"]
    for ii, (model, name) in enumerate(zip(models, names), 1):
        plt.subplot(4, 1, ii)
        plt.title(name)
        for sig, color in zip(model.T, colors):
            plt.plot(sig, color=color)

    plt.subplots_adjust(0.09, 0.04, 0.94, 0.94, 0.26, 0.46)
    plt.savefig(os.path.join(TMP_FOLDER, "demo.png"), format="pn
g")

```

```
48      return TMP_FOLDER
49
50
51  if __name__ == "__main__":
52      suanpan.run(app)
```

案例2.2 视频教程

2.2.mov

题目：数据集包含100个样本，其中正、反例各一半，假定学习算法所产生的模型是将新样本预测为训练样本数较多的类别（训练样本数相同时进行随机猜测），试给出用10折交叉验证法和留一法分别对错误率进行评估所得的结果。

步骤1：在“项目模板”中找到周志华习题，打开第二章模型评估与选择习题案例2.2，创建模板。

步骤2：在“CSV上传模块”上传data数据，一般为csv格式。这里就是上传的是包含100个的数据集。

步骤3：在全部组件列表搜索“数据拆分”模块，拖到操作界面内，

在参数设置的数据1比重与数据2比重输入0.7:0.3，数据1做为训练集，数据2作为测试集。

步骤4：全部组件列表搜索“岭分类”RidgeClassifier模块，只需在字段设置中设置需要的特征字段。参数设置一般不改。

岭分类即使用岭回归的分类器。对于多类别分类，以一对多的方法训练n_class分类器。通锅利用ridge中的多变量响应支持来实现的。

Alpha表示正则强度，必须为正浮点数。Fit intercept为是否计算此模型的截距。

Tol为解决方案的精度。

步骤5：全部组件列表搜索“带交叉验证的岭分类”模块，只需在字段设置中设置需要的特征字段。参数设置一般不改。

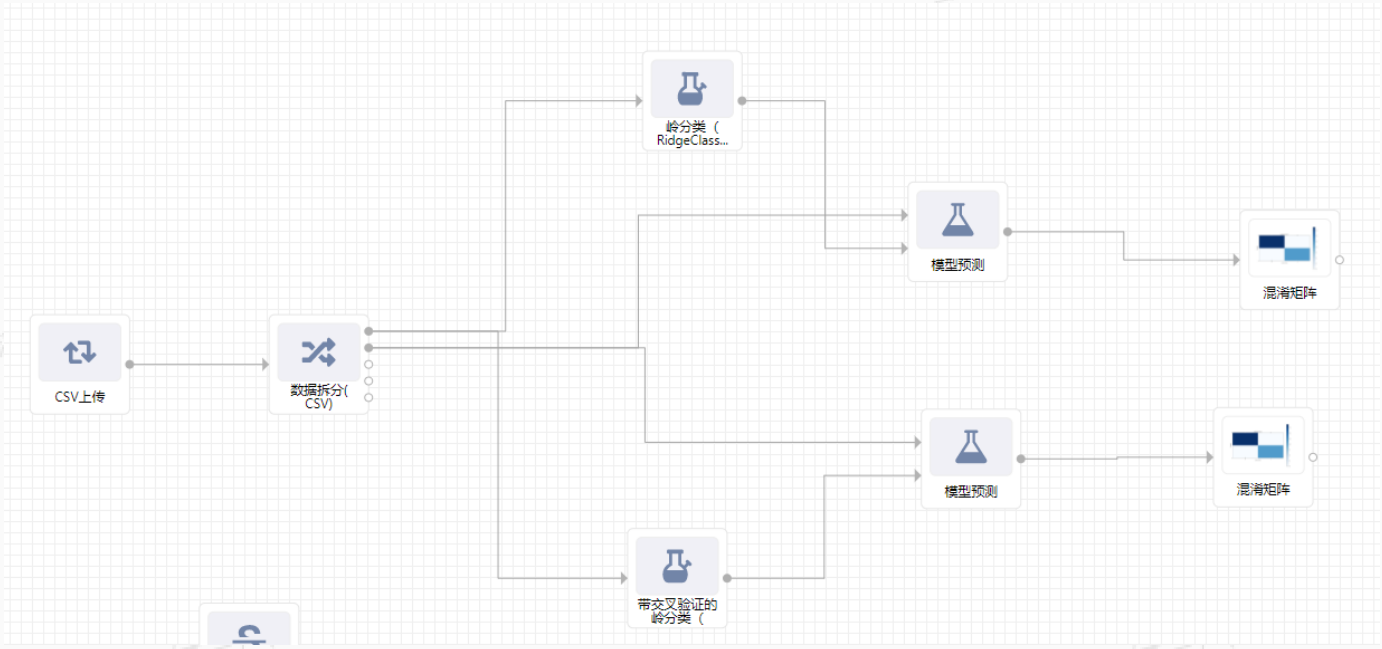
步骤6：全部组件列表搜索“模型预测”模块，分别连接预待预测数据和经过岭分类处理的训练集。设置下特征字段。

步骤7：全部组件列表搜索“混淆矩阵”CSV模块。“标签字段”为“class”，预测字段为“prediction”，参数设置不需要更改。

混淆矩阵confusion matrix 又被称为错误矩阵。每一行代表预测值，每一列代表实际类别。

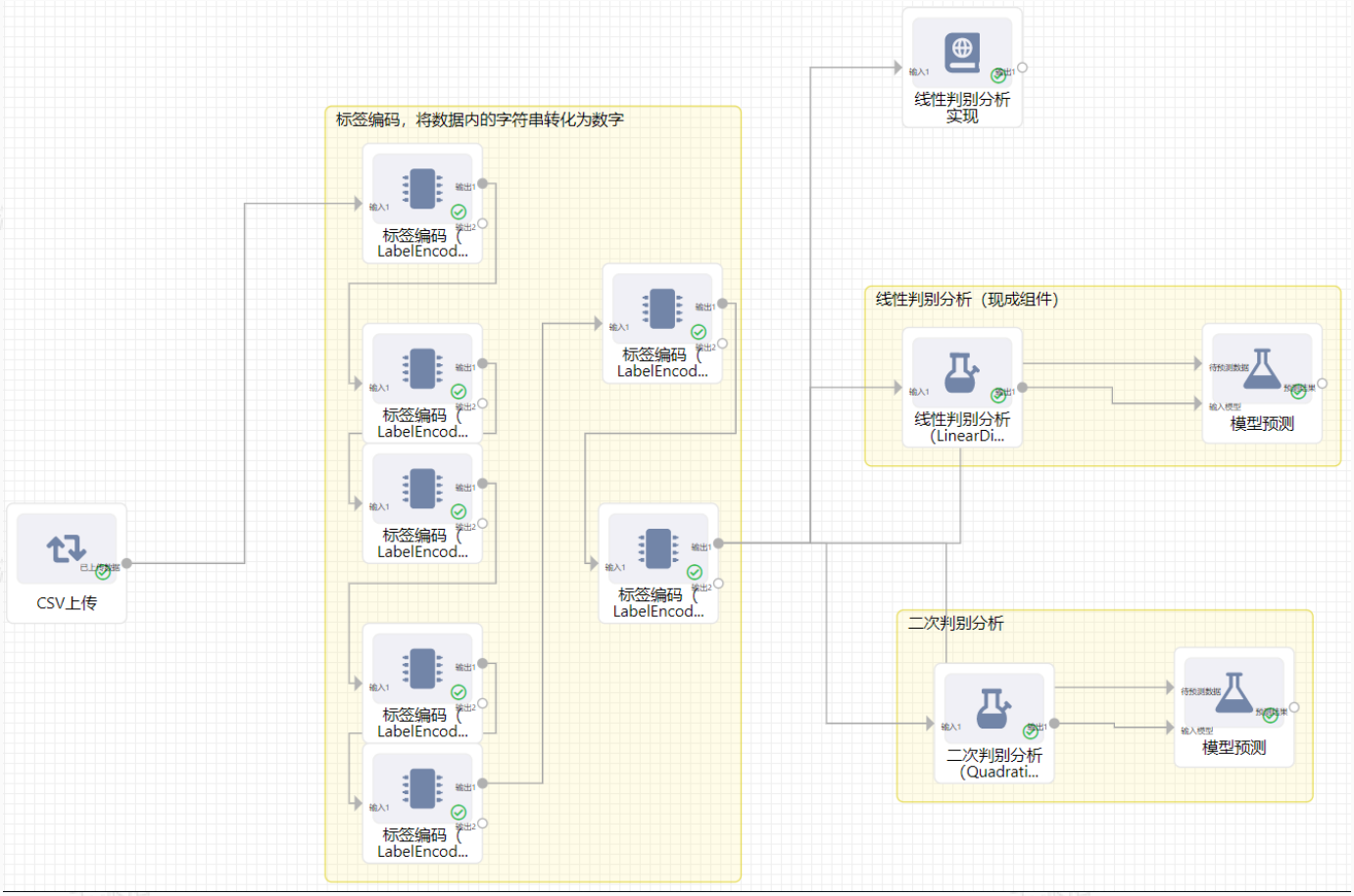
作出正确判断的肯定记录（真阳性）、作出错误判断的肯定记录（假阴性）、作出正确判断的否定记录（真阴性）以及作出错误判断的否定记录（假阳性）。

模型评估与预测案例— 2.2



上图为例2.2在算盘中的示例

线性模型案例二 例3.5



上图即为例3.5在算盘中的示例

视频教程：https://www.yuque.com/docs/share/161ee5fd-1a24-49bb-bfed-fd317d7e5c37#

示例项目大致分为5个部分

- 第一部分：CSV上传，上传CSV数据文件，这里的数据即为西瓜数据集3.0α
- 第二部分：图中的标签编码部分，将数据中的字符串列进行编码，比如将'好瓜'列中的'好'，'坏'变为0，1
- 第三部分：图中的线性判别分析实现节点，手动用python实现了线性判别分析算法，代码如下：
  - lda.py

```python
import numpy as np
from sklearn.metrics import accuracy_score

def _class_means(X, y):
    classes, y = np.unique(y, return_inverse=True)
    cnt = np.bincount(y)
    means = np.zeros(shape=(len(classes), X.shape[1]))
    np.add.at(means, y, X)
```

```
    means /= cnt[:, None]
    return means


class LinearDiscriminantAnalysis(object):
    def __init__(self):
        self.classes_ = np.array([0, 1])
        self.tol = 1e-4

    def train(self, X, y):
        n_samples, _ = X.shape
        n_classes = len(self.classes_)

        _, y_t = np.unique(y, return_inverse=True)  # non-negati
ve ints
        self.priors_ = np.bincount(y_t) / float(len(y))
        if not np.isclose(self.priors_.sum(), 1.0):
            logger.warn("The priors do not sum to 1. Renormalizi
ng", UserWarning)
            self.priors_ = self.priors_ / self.priors_.sum()

        # Maximum number of components no matter what n_componen
ts is
        # specified:
        max_components = min(len(self.classes_) - 1, X.shape[1])

        self._max_components = max_components
        self._solve_svd(X, y)

        if self.classes_.size == 2:  # treat binary case as a sp
ecial case
            self.coef_ = np.array(
                self.coef_[1, :] - self.coef_[0, :], ndmin=2, dt
ype=X.dtype
            )
            self.intercept_ = np.array(
                self.intercept_[1] - self.intercept_[0], ndmin=1
, dtype=X.dtype
            )
        return self
```

```
43
44      def predict(self, X):
45          scores = self._decision_function(X)
46          if len(scores.shape) == 1:
47              indices = (scores > 0).astype(np.int)
48          else:
49              indices = scores.argmax(axis=1)
50          return self.classes_[indices]
51
52      def evaluate(self, X, y):
53          return accuracy_score(y, self.predict(X))
54
55      def _solve_svd(self, X, y):
56          n_samples, _ = X.shape
57          n_classes = len(self.classes_)
58
59          self.means_ = _class_means(X, y)
60
61          Xc = []
62          for idx, group in enumerate(self.classes_):
63              Xg = X[y == group, :]
64              Xc.append(Xg - self.means_[idx])
65
66          self.xbar_ = np.dot(self.priors_, self.means_)
67
68          Xc = np.concatenate(Xc, axis=0)
69
70          # 1) within (univariate) scaling by with classes std-dev
71          std = Xc.std(axis=0)
72          # avoid division by zero in normalization
73          std[std == 0] = 1.0
74          fac = 1.0 / (n_samples - n_classes)
75
76          # 2) Within variance scaling
77          X = np.sqrt(fac) * (Xc / std)
78          # SVD of centered (within)scaled data
79          U, S, V = np.linalg.svd(X, full_matrices=False)
80
81          rank = np.sum(S > self.tol)
82          # Scaling of within covariance is: V' 1/S
```

```
        scalings = (V[:rank] / std).T / S[:rank]

        # 3) Between variance scaling
        # Scale weighted centers
        X = np.dot(
            (
                (np.sqrt((n_samples * self.priors_) * fac))
                * (self.means_ - self.xbar_).T
            ).T,
            scalings,
        )
        # Centers are living in a space with n_classes-1 dim (maximum)
        # Use SVD to find projection in the space spanned by the
        # (n_classes) centers
        _, S, V = np.linalg.svd(X, full_matrices=0)

        self.explained_variance_ratio_ = (S ** 2 / np.sum(S ** 2))[
            : self._max_components
        ]
        rank = np.sum(S > self.tol * S[0])
        self.scalings_ = np.dot(scalings, V.T[:, :rank])
        coef = np.dot(self.means_ - self.xbar_, self.scalings_)
        self.intercept_ = -0.5 * np.sum(coef ** 2, axis=1) + np.log(self.priors_)
        self.coef_ = np.dot(coef, self.scalings_.T)
        self.intercept_ -= np.dot(self.xbar_, self.coef_.T)

    def _decision_function(self, X):
        n_features = self.coef_.shape[1]
        if X.shape[1] != n_features:
            raise ValueError(
                "X has %d features per sample; expecting %d" % (X.shape[1], n_features)
            )

        scores = X @ self.coef_.T + self.intercept_
        return scores.ravel() if scores.shape[1] == 1 else scores
```

```
118
```

- main.py

```python
from sklearn.model_selection import train_test_split

import suanpan
from suanpan.app import app
from suanpan.app.arguments import Csv, Json, ListOfString, String
from suanpan.log import logger
from lda import LinearDiscriminantAnalysis


@app.input(Csv(key="inputData1"))
@app.param(ListOfString(key="param1", alias="featureColumns"))
@app.param(String(key="param2", alias="labelColumn"))
@app.output(Json(key="outputData1"))
def LinearDiscriminantAnalysisImplmentation(context):
    args = context.args

    df = args.inputData1

    X = df[args.featureColumns].values
    y = df[args.labelColumn].values

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.33, random_state=42
    )

    lda = LinearDiscriminantAnalysis()

    lda.train(X_train, y_train)

    score = lda.evaluate(X_test, y_test)
    logger.info("Predicted Results: {}".format(lda.predict(X_test
)))

    return {"accuracy": score}


```

```
36 if __name__ == "__main__":
37     suanpan.run(app)
```

    - 单独运行该节点，然后点击 

即可进入vscode查看，修改代码
    - 该节点会输出分类的准确率
- 第四部分，图中的线性判别分析（现成组件）部分，直接使用了算盘中已有的线性判别分析分类组件
- 第五部分，图中的二次判别分析部分，直接使用了算盘中已有的二次判别分析分类组件

案例3.2 视频教程

3.2.mov

题目：

3.2 试证明，对于参数 $\boldsymbol{w}$，对率回归的目标函数(3.18)是非凸的，但其对数似然函数(3.27)是凸的.

步骤1：在“项目模板”中找到周志华习题，打开第三章线性模型习题案例3.2，创建模板。

步骤2：在“CSV上传模块”上传data数据，一般为csv格式。这里用到的是西瓜数据4.0。

版。

步骤3：在全部组件列表搜索“标签编码”模块，拖到操作界面内，

因为上传的数据特征为“色泽 根蒂 敲声 纹理 脐部 触感”等，需要转化为数字，“标签编码（LabelEncoder）”可以通过设置“字段设置”中的“目标字段”，来把特征取值转化为数字。

依次拖出“标签编码”，在字段设置的目标字段中设置“色泽”“根蒂”“敲声”“纹理”“脐部”“触感”，即可将数据中的字符串进行编码。

步骤4：在全部组件列表中搜索“逻辑回归”LogisticRegression模块，在字段设置中输入“色泽，根蒂，敲声，纹理，脐部，触感，密度，含糖率”特征字段，标识字段为“好瓜”。参数不用改。

逻辑回归分类器 在多类情况下，multi_class选项设置为ovr，使用liblinear库，默认情况下正则化。

步骤5：在全部组件列表搜索“模型预测”模块，设置好特征字段和预测字段，输入待预测数据和经过逻辑回归训练完的模型，输出预测后的数据可直接查看。

步骤6：在全部组件列表搜索“分类评估”模块，分类评估组件，用于二分类模型预测结果的评估，在“评估指标”选项中选择accuracy_score，该函数可以计算正确预测的精度。在字段设置中的标签列输入“好瓜”，预测列为“prediction”。输出结果中的accuracy_score，true_negatives，false_positives,false_negatives,true_positives分别代表：

真阳性（True Positive，简称TP），也就是预计为真，实际上也为真的数据

假阳性（False Positive，简称FP）,也就是预计为真，但实际上为假的数据。

假阴性(False Negative，简称FN)，也就是预计为假，但实际上为真的数据

真阴性（True Negative，简称TN），也就是预计为假，实际上也为假的数据。

案例3.4 视频教程

3.4.mov

题目：选择两个UCI数据集，比较10折交叉验证法和留一法所估计出的对率回归的错误率。

步骤1：在“项目模板”中找到周志华习题，打开第三章线性模型习题案例3.4，创建模板。

步骤2：在“CSV上传模块”上传data数据，一般为csv格式。这里用到的是葡萄酒数据集。

版。

：骤3：拖出“数据拆分CSV”模块，在“参数设置”中将数据1比重和数据2比重设置为0.7:0.3，第一份数据作为待预测数据的测试集，用于评估分类器，剩下的作为训练集，用于构建分类器。

步骤4：在全部组件中选择带交叉验证的逻辑回归模块LogisticRegressionCV。在“参数设置”中，把Cs改为0.0001,0.0001.较小的值制定更强的正则化。是正则强度的倒数。Cv交叉验证生成器设置为10，意思为分层10折，10折交叉验证即把数据集随机分成10份，其中9份用于训练而另一份用作测试。该过程重复10次，每次用的测试数据不同。具有随机性，非确定性。特征字段里勾选跟前面的相同。其他参数不用更改。

步骤5：在全部组件里搜索“模型预测”，把测试集数据和经过10折交叉验证的数据模型连接，设置好特征字段。

步骤6：在全部组件里搜索“多分类评估”MultiClass模块，多分类评估组件时专门用于多分类模型的预测结果评估。在“字段设置”中的“标签列”输入“class”，预测列为prediction。在“参数设置”中的“评估指标”中选择accuracy_score即可。Accuracy_score函数计算正确预测的精度。

至此已经可以从头开始运行，查看10折交叉验证的准确率结果。

0.7037037037

步骤7：在全部组件中搜索并拖出“逻辑回归”LogisticRegression模块。组件ID为1721.

在“字段设置”的“特征字段”中输入“Alcohol,Malic acid,Ash,AshAlcalinity of ash,Magnesium,Total phenols,Flavanoids,Nonflavanoid phenols,Proanthocyanins,Color intensity,Hue,OD280/OD315 of diluted wines,Proline”，“标识字段”为class。

logistic回归分类器，multi class设置为“ovr”，训练算法使用一对多休息方案。在求解器支持“多项式”选项时可以选择“多项式”。“liblinear”求解器实现正则逻辑回归，支持L1和 L2正则化。默认情况下正则化应用，它可以处理密集和稀疏输入。适合小型数据集。

c为正则强度的倒数，默认为1。其他参数基本不用动，如果需要改，可以点击模块，查看帮助文档链接。

留一法：n折交叉验证（n为数据集中样本的数目）。每次迭代都使用了最大可能数目的样本来训练。具有确定性。但是计算量很大，所以适合小型数据集。

步骤8：拖出模型预测组件和多分类评估组件，参数设置和字段设置跟10折交叉验证相同。

点击从头开始运行，同时进行留一法和10折交叉验证，查看结果。0.94。

案例3.5 视频教程

3.5.mov

题目：编程实现线性判别分析，并给出西瓜数据集3.0α上的结果。

步骤1：在“项目模板”中找到周志华习题，打开第三章线性模型习题案例3.5，创建模板。

步骤2：在“CSV上传模块”上传data数据，一般为csv格式。这里用到的是西瓜数据集3.0α。

版。

步骤3：在全部组件列表搜索“标签编码”模块，拖到操作界面内，

因为上传的数据特征为“色泽 根蒂 敲声 纹理 脐部 触感”等，需要转化为数字，“标签编码（LabelEncoder）”可以通过设置“字段设置”中的“目标字段”，来把特征取值转化为数字。

依次拖出“标签编码”，在字段设置的目标字段中设置“色泽”“根蒂”“敲声”“纹理”“脐部”“触感”，即可将数据中的字符串进行编码，

步骤4：在全部组件列表搜索“线性判别分析”linear discrimination analysis模块，组件ID为1731，在算法设计的机器学习中，“分类”选项卡下。在“参数设置”中设置特征字段“色泽，根蒂，敲声，纹理，脐部，触感，密度，含糖率”，“标识字段”设置为“好瓜”。可勾选“store covariance”存储协方差。“tol”设置为0.0001.较小的tol参数可以尽量避免相同的输入数据输出略微不同的结果。

Solver中选择svd求解器。线性判别分析下一般默认求解器为“svd”，它可以执行分类和变换，并且不依赖于协方差矩阵的计算。Lsqr求解器是仅使用于分类的高效算法。本征求解器基于类间散度到类内散度比的优化，它可以用于分类和转换，但是需要计算协方差矩阵，因此可能不太适合具有大量特征的情况。

“线性判别分析”是一个具有线性决策边界的分类器，通过将类的密度拟合到数据并且使用贝叶斯规则生成的。假设所有类别共享相同的协方差矩阵，则该模型将高斯密度拟合到每个类别。拟合模型还可以通过将其投影到最有区别的方向来减少输入的维数。

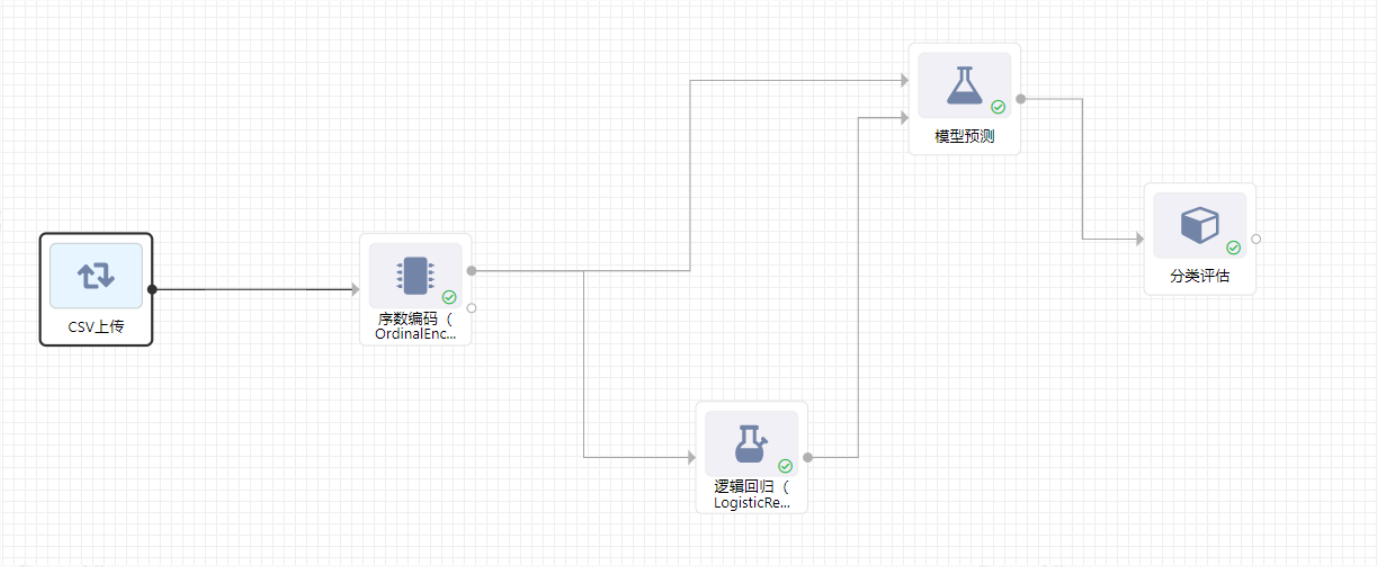
步骤5：在全部组件列表搜索“模型预测”，把经过标签编码数字化的数据作为待预测数据，把经过线性判别分析的数据输出为输入模型，“字段设置”与待预测数据的特征字段相同。

步骤6：也可以时手动用python实现线性判别分析算法。

步骤7：在全部组件列表搜索“二次判别分析”Quadratic Discriminant Analysis模块。跟线性判别分析模块一样，字段设置相同，参数设置可不做更改。拖出“模型预测”模块，设置线性判别分析。

二次判别分析是通过将类别条件密度拟合到数据并使用贝叶斯规则生成具有二次决策边界的分类器。该模型将高斯密度拟合到每个类别。

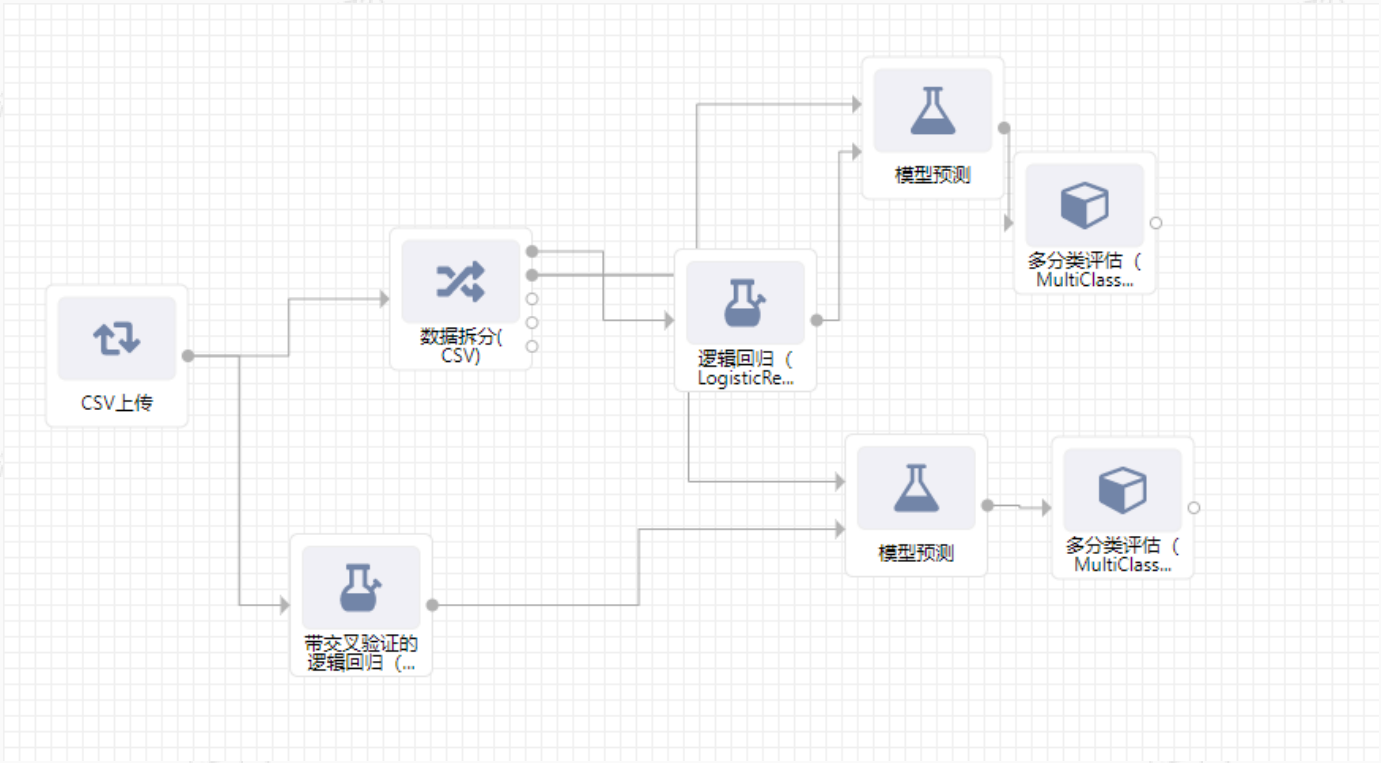
线性模型案例— 3.2



上图为例3.2在算盘中的示例

第一部分：CSV上传，上传CSV数据文件，这里用的到的是西瓜数据集4.0.
第二部分：序数编码，将数据中的字符串进行编码，比如将“色泽”，“根蒂”，“敲声”等转化为0,1,2.
第三部分：数据训练，逻辑回归组件，底层的C实现在拟合模型时使用随机数生成器选择特征。因此，对于相同的输入数据具有略微不同的结果并不罕见。如果发生这种情况，请尝试使用较小的tol参数。
在某些情况下，预测输出可能与独立liblinear的输出不匹配。
第四部分：模型预测，将数字化的数据集作为测试集，经过逻辑回归训练的数据模型作为训练数据。
第五部分：评估模型预测结果，分类评估组件有多个评估指标，可以量化预测的质量。

线性模型案例三 3.4



上图为例3.4在算盘中的示例

案例4.3 4.4 视频教程

4.3 4.4.mp4

题目：

4.3 试编程实现基于信息熵进行划分选择的决策树算法，为表4.3中数据生成一棵决策树。

4.4 试编程实现基于基尼指数进行划分选择的决策树算法，为表4.2中数据生成预剪枝、后剪枝决策树，并与未剪枝决策树进行比较。

表 4.2 西瓜数据集 2.0 划分出的训练集(双线上部)与验证集(双线下部)

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
10	青绿	硬挺	清脆	清晰	平坦	软粘	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否
编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	否

表 4.3 西瓜数据集 3.0

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.460	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	0.774	0.376	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	0.634	0.264	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	0.608	0.318	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	0.556	0.215	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	0.403	0.237	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	0.481	0.149	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	0.437	0.211	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	0.666	0.091	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	0.243	0.267	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	0.245	0.057	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	0.343	0.099	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	0.639	0.161	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	0.657	0.198	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	0.360	0.370	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	0.593	0.042	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	0.719	0.103	否

步骤1：在“项目模板”中找到周志华习题，打开第四章决策树习题案例4.3和4.4，创建模板。

步骤2：在“EXCEL上传”模块上传excel文件，这里用到的是西瓜数据集3.0。

步骤3：在全部组件列表搜索“VS Code Python”模块，拖到操作界面内，在“参数设置”中勾选上“编辑”，在点击运行该节点，在操作中“编辑VS Code”，编写将excel文件的数据转换为CSV格式的文件的代码。注意编写完之后要重新勾掉“编辑”选项，才可以运行程序。

步骤4：在全部组件列表搜索“序数编码”模块，拖到操作界面内，在字段设置的“目标字段”中输入“色泽，根蒂，敲声，纹理，脐部，触感，好瓜”特征字段。该模块主要用来将分类特征编码为整数数组。该模块的输入应为整数或者字符串的数组，表示分类特征采用的值。目的是找到每个特征的唯一值，然后将数据转换为叙述编码。
步骤5：在全部组件列表搜索“数据拆分CSV”模块，在“参数设置”的数据1比重和数据2比重设置为0.8:0.2，数据1集作为训练集，数据2作为测试集。
步骤6：在全部组件列表搜索“决策树分类Decision Tree Classifer”模块，首先在“字段设置”中输入特征字段“编号，色泽，根蒂，敲声，纹理，脐部，触感，密度，含糖率”和标识字段“好瓜”。
在“参数设置”的“criterion”中选择“entropy”时，使用的是ID3算法。
在“参数设置”的“criterion”中选择“gini”时，使用的是CART算法。当max depth，max leaf nodes参数大小为默认值时，会导致树完全生长和未修剪。为了减少内存消耗，可通过设置max depth为5，max leaf nodes 为3来控制树的复杂性和大小。
步骤7：在全部组件列表搜索“模型预测”模块，可以查看最终在树上的数据。可以自己采取一些可视化手段来更清晰的查看最终结果。如plot_tree函数绘制树：

```
树。plot_tree （CLF 。拟合（虹膜。数据， 虹膜。目标））
```

案例4.6 视频教程

4.6.mp4

题目：试选择4个UCI数据集，对上述3种算法所产生的未剪枝、预剪枝、后剪枝决策树进行实验比较，并进行适当的统计显著性检验。

步骤1：在“项目模板”中找到周志华习题，打开第四章决策树习题案例.3和4.4，创建模板。

步骤2：在常用数据集中找到“乳癌数据”，作为本次模板的数据集。

步骤3：在全部组件列表中找到“数据拆分CSV”模块，在“参数设置”中将数据1比重和数据2比重设置为0.8:0.2，数据1作为训练集，数据2作为测试集。

步骤4：在全部组件列表中找到“决策树分类DecisionTreeClassifer”模块，在参数设置的criterion中选择gini基尼指数作为纯度。这里用到的是CART算法。CART（classification and regression tree）算法，二元切分法。采用基尼指数gini来选择最好的数据分割的特征，gini描述的是纯度，与信息熵的含义相似。

步骤5：在全部组件列表中找到“模型预测”模块设置特征字段和预测字段。

步骤6：在全部组件列表中找到“分类评估”模块，因为CART算法采用的是二元切分法，所以可以采用分类评估组件，用于二分类模型预测结果的评估。“参数设置”中的评估指标不设置，最终会有所有的具体数据，在字段设置中的“标签列”输入“cancer”，预测列输入prediction。总体来说后剪枝会比预剪枝保留更多的分支。剪枝操作是提升模型泛化能力的重要途径。

步骤7：运行完所有步骤程序，点击“分类评估”组件，可以查看所有结果。根据具体题目需要的数据，可以在“评估指标”中增减删改。此模型的优势是，当输出之间没有相关性时，仍旧可以通过建立一个模型来同时预测n个输出的每个模型。通过添加VS Code Python 组件编写代码可以查看未剪枝，预剪枝，后剪枝对测试样本的输出结果。

案例6.2 视频教程

6.2.mp4

题目:试使用LIBSVM，在西瓜数据集3.0α上分别用线性核和高斯核训练一个SVM，并比较其支持向量的差别。

步骤1：在“项目模板”中找到周志华习题，打开第六章向量机习题案例6.2，创建模板。

步骤2：在“CSV上传”模块上传CSV文件，这里用到的是西瓜数据集3.0。

步骤3：在全部组件列表搜索“序数编码”模块，拖到操作界面内，在字段设置的“目标字段”中输入“色泽，根蒂，敲声，纹理，脐部，触感，好瓜”特征字段。该模块主要用来将分类特征编码为整数数组。该模块的输入应为整数或者字符串的数组，表示分类特征采用的值。目的是找到每个特征的唯一值，然后将数据转换为叙述编码。

步骤4：在全部组件列表搜索“数据拆分CSV”模块，在“参数设置”的数据1比重和数据2比重设置为0.7:0.3，数据1集作为训练集，数据2作为测试集。

步骤5：在全部组件列表搜索“支持向量机分类SVC”组件，该实现基于libsvm，拟合时间与样本数量成平方比例。在参数设置的“kernel”字符串中，默认为rbf，指的是指定算法中要使用的内核类型。Linear为线性核，rbf为高斯核。其他参数均可选择默认参数。C为误差项的惩罚参数。一般来说，在支持向量机中，惩罚系数越大，正则化程度越低。高斯核的支持向量数目会较少，而线性核的会几乎没有变化。

步骤6：在全部组件列表搜索“模型预测”模块，将测试集与经过训练的训练集作为输入，设置“特征字段”输出模型到“分类评估”组件。特征字段为“色泽，根蒂，敲声，纹理，脐部，触感，密度，含糖率”。

步骤7：在全部组件列表搜索“分类评估”模块，在参数设置中选择accuracy_score，字段设置的票钱列为“好瓜”，预测列为prediction。

步骤8：运行全部程序，查看结果，由于西瓜数据集3.0线性不可分，即使C惩罚系数高，还是会出现误分类的情况，使用高斯核在惩罚系数较大时，可以完全拟合训练数据。可以通过调整C惩罚系数来提高精度。

案例6.3 视频教程

6.3.mp4

题目：

选择两个UCI数据集，分别用线性核和高斯核训练一个SVM，并与BP神经网络和C4.5决策树进行实验比较。

步骤1：在“项目模板”中找到周志华习题，打开第六章向量机习题案例6.2，创建模板。

步骤2：在“CSV上传”模块上传CSV文件，这里用到的是鸢片花瓣的一个UCI数据集。

步骤3：在全部组件列表搜索“数据拆分CSV”模块，在“参数设置”的数据1比重和数据2比重设置为0.7:0.3，前70个数据是数据1集作为训练集，后30个样本是数据2作为测试集。

步骤4：在全部组件列表搜索“支持向量机分类SVC”组件，该实现基于libsvm，拟合时间与样本数量成平方比例。在参数设置的“kernel”字符串中，默认为rbf，指的是指定算法中要使用的内核类型。Linear为线性核，rbf为高斯核。其他参数均可选择默认参数。C为误差项的惩罚参数。一般来说，在支持向量机中，惩罚系数越大，正则化程度越低。高斯核的支持向量数目会较少，而线性核的会几乎没有变化。

分别在kernel选一个rbf和linear使用高斯核和线性核。同时设置好特征字段和标识字段。

步骤5：在全部组件列表搜索“决策树分类”DecisionTreeClassifer组件，criterion参数选择entropy，C4.5算法是ID3算法的延伸。区别于ID3算法通过信息增益选择分裂属性，C4.5算法通过信息增益率选择分裂属性。Max depth 为10，min samples split为2，min samples leaf为1。Max leaf nodes 为2. min impurity decrease为1.

C4.5算法对ID3算法主要做了一下几点改进：

（1）通过信息增益率选择分裂属性，克服了ID3算法中通过信息增益倾向于选择拥有多个属性值的属性作为分裂属性的不足；

（2）能够处理离散型和连续型的属性类型，即将连续型的属性进行离散化处理；

（3）构造决策树之后进行剪枝操作；

（4）能够处理具有缺失属性值的训练数据。

步骤:6：在全部组件列表搜索“模型预测”组件，设置好字特征字段和标识字段。

步骤7：在全部组件列表搜索“分类评估”组件，在“参数设置”的“评估指标”中选择accuracy_score。字段设置的标签列中输入class，预测列输入prediction。

案例6.8 视频教程

6.8.mp4

题目：以西瓜数据集3.0α的“密度”为输入，“含糖率”为输出，试使用LIBSVM训练一个SVR。

步骤1：在“项目模板”中找到周志华习题，打开第六章向量机习题案例6.8，创建模板。

步骤2：在“CSV上传”模块上传CSV文件，这里用到的是西瓜数据集3.0α。

步骤3：在全部组件列表搜索“支持向量机分类SVR”组件，该实现基于libsvm（支持向量机的一个库）。拟合时间的复杂度是样本数量的两倍以上。Kernel参数选择rbf默认值，模型中的自由参数是C（误差项的惩罚参数C）和epsilon（epsilon-SVR模型中的Epsilon），epsilon支持向量回归。参数不同，训练结果不同。题目中“密度”作为输入，“含糖率”为输出，所以特征字段为密度，标识字段为含糖率。

步骤4：在全部组件列表搜索“VS code Python”组件，编写代码作为可视化模块。偏置0.20，最终结果可看出含糖率与密度并没有很大关系。

案例7.3 视频教程

7.3.mp4

题目：试编程实现拉普拉斯修正的朴素贝叶斯分类器，并以西瓜数据集3.0位训练集，对p151“测1”样本进行判别。

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
测 1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.460	?

步骤1：在“项目模板”中找到周志华习题，打开第七章贝叶斯分类器习题案例7.3，创建模板。

步骤2：在“CSV上传”模块上传CSV文件，这里用到的是西瓜数据集3.0。

步骤3：在全部组件列表搜索“序数编码”模块，拖到操作界面内，在字段设置的“目标字段”中输入“色泽，根蒂，敲声，纹理，脐部，触感，好瓜”特征字段。该模块主要用来将分类特征编码为整数数组。该模块的输入应为整数或者字符串的数组，表示分类特征采用的值。目的是找到每个特征的唯一值，然后将数据转换为叙述编码。

步骤4：拉普拉斯修正实质上假设了属性值与类别均匀分布，这是在朴素贝叶斯学习过程中额外引入的关于数据的先验。拉普拉斯修正可以避免因训练集样本不充分而导致概率估值为0的问题。在全部组件中搜索贝叶斯分类器，选择一个朴素贝叶斯分类器，如多项式朴素贝叶斯分类器，也适用于具有离散特征的分类。参数设置无需修改，特征字段为“色泽，根蒂，敲声，纹理，脐部，触感，密度，含糖率”。标识字段为“好瓜”。

步骤5：在全部组件列表中选择“CSV上传”组件，上传P151中的测1样本。即使样本数量较少，也不会导致概率估计为0.步骤6：在全部组件列表中选择“模型预测”组件，测1样本作为测试集，经过贝叶斯分类器学习的西瓜数据集作为训练集，这两个数据集作为输入，就可以对测1样本进行判别。也可以通过书上的公式去计算。

案例7.6 视频教程

7.6.mp4

题目：试编程实现AODE分类器，并以西瓜数据集3.0为训练集，对p151的“测1”样本进行判别。

步骤1：在“项目模板”中找到周志华习题，打开第七章贝叶斯分类器习题案例7.6，创建模板。

步骤2：在“CSV上传”模块上传CSV文件，这里用到的是西瓜数据集3.0。

步骤3：在全部组件列表中搜索“vs code python”组件，拖到界面内，点击该组件，勾选上参数设置中的“编辑”，运行该结点，在代码编辑页面内编写半朴素贝叶斯分类模型，根据属性下各取值的样本数量，决定某些属性可以作为父属性，AODE本身旨在取值样本数量低于一定阈值30的属性去除，所以连续值不能作为父属性，但在样本小的情况下，相比于朴素贝叶斯，AODE对西瓜数据集的拟合效果更好，错误率更低。

案例8.3 8.5 视频教程

8.3 8.5.mp4

题目：

8.3 从网上下载或自己编程实现AdaBoost，以不剪枝决策树为基学习器，在西瓜数据集3.0α上训练一个AdaBoost集成，并与图8.4进行比较。

8.5 试编程实现Bagging集成，并与图8.6进行比较。

步骤1：在“项目模板”中找到周志华习题，打开第八章集成学习习题案例8.3和8.5，创建模板。

步骤2：在“EXCEL上传”模块上传excel，这里用到的是西瓜数据集3.0.

步骤3：在全部组件列表搜索“VS Code Python”模块，拖到操作界面内，在“参数设置”中勾选上“编辑”，在点击运行该节点，在操作中“编辑VS Code”，编写将excel文件的数据转换为CSV格式的文件的代码。注意编写完之后要重新勾掉“编辑”选项，才可以运行程序。

步骤4：在全部组件列表搜索“序数编码”模块，拖到操作界面内，在字段设置的“目标字段”中输入“色泽，根蒂，敲声，纹理，脐部，触感，好瓜”特征字段。该模块主要用来将分类特征编码为整数数组。该模块的输入应为整数或者字符串的数组，表示分类特征采用的值。目的是找到每个特征的唯一值，然后将数据转换为叙述编码。

步骤5：在全部组件列表搜索“AdaBoost”分类组件，拖到操作页面，adaboost分类器是一种元估计器，它首先将分类器拟合到原始数据集上，然后将分类器的其他副本拟合到同一数据集上，但是对错误分类的实例的权重进行了调整。参数N_estimators表示终止增强的估计器的最大数量，设为5，learning_rate学习率为1，用来缩小每个分类器的贡献。算法SAMME.R是实际增强算法。字段设置中的“特征设置”输入“色泽,根蒂,敲声,纹理,脐部,触感,密度,含糖率”。标识字段为“好瓜”。

根据个体学习器的生成方式，目前的集成学习方法大致可分为两类，即个体学习器间存在强依赖关系、必须串行生成的序列化方法，如boosting；以及个体学习器间不存在强依赖关、可同时生成的并行化方法，如bagging和“随机森林”random forest。

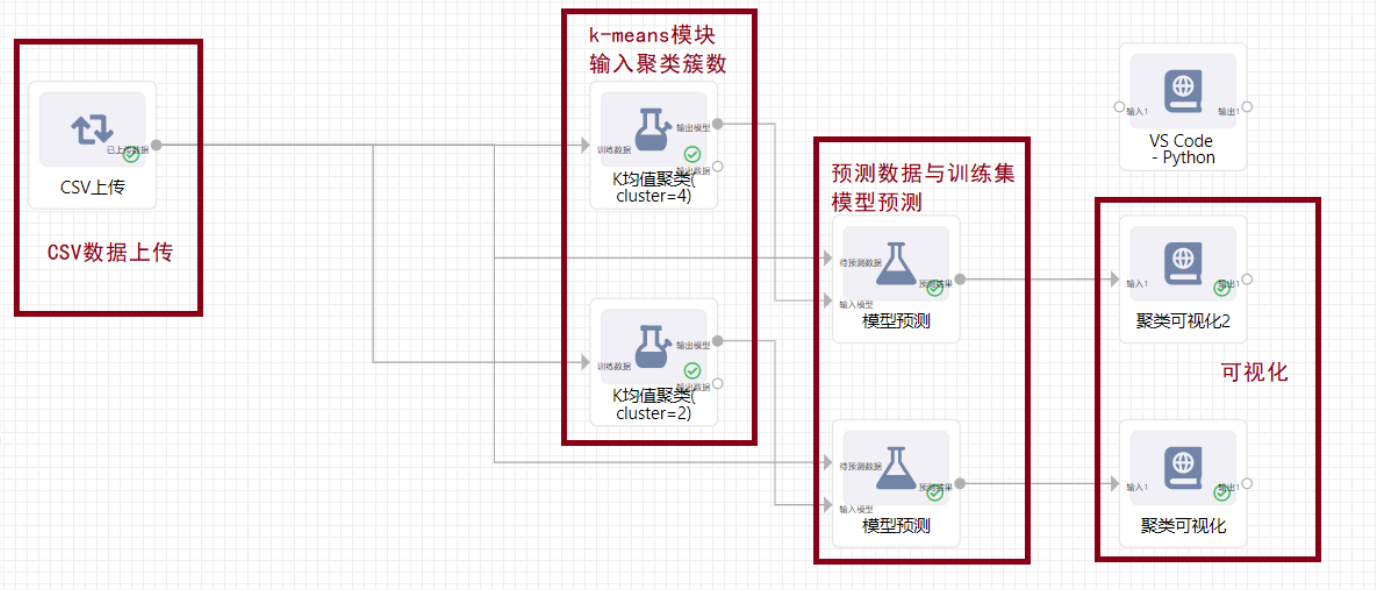
标准AdaBoost只适用于二分类任务不同，bagging能不经修改地用于多分类、回归等任务。

步骤5：在全部组件列表搜索“bagging分类”分类组件，拖到操作页面，bagging分类器是一个集合元估计器，它使每个基本分类器适合原始数据集的随机子集，然后通过投票或平均奖其单个预测进行汇总以形成最终预测。参数设置中的n_estimators为集合中的基本估计量，设置为50，如在完美配合的情况下，学习程序会尽早停止。其他参数均为默认值。字段设置中的“特征设置”输入“色泽,根蒂,敲声,纹理,脐部,触感,密度,含糖率”。标识字段为“好瓜”。

步骤6：在全部组件列表搜索“模型预测”组件，把经过将特征字符串编码成整数值的西瓜数据集作为测试集，经过adaboost和bagging集成训练的数据集为训练集，同时作为模型预测的输入。输出分别为adaboost和bagging集成的模型结果。

聚类算法案例一 例题9.4

9.4 试编程实现k均值算法，设置三组不同的k值、三组不同初始中心点，在西瓜数据集4.0上进行试验比较，并讨论什么样的初始中心有利于取得好结果。



第一部分：CSV上传，上传CSV数据文件，这里数据用的是西瓜数据集4.0.
这是一个包含密度与含糖率两个属性值的二维向量。

Number（编号），density（密度），sugercontent（含糖率）

第二部分：直接使用k均值聚类（k-means）模块，同时输入聚类簇数。

k均值算法介绍：
给定样本集D，k-means算法（k均值算法）通过迭代优化针对聚类所得簇划分C最小化平方误差E，E镖师簇内样本围绕簇均值向量的紧密程度，E越小簇内样本相似度越高。这是一种基于划分的聚类算法，计算量大，但很容易发现数据库中的球状簇。

计算流程：
输入：样本集D；聚类簇数k。
过程：1初始化：从D中随机选取k个样本作为初始均值向量，
2簇划分：计算各个样本与各均值向量的距离，根据距离最近的均值向量确定各个样本的簇标记，然后将各个样本划入相应的簇。
3均值向量迭代更新：计算新的均值向量，重复划分样本的操作。
若迭代更新后聚类结果保持不变，
输出：簇划分C。

第三部分：模型预测，将待预测数据和经过k-means算法训练后的数据进行模型预测。

第四部分：用VS code Python模块将输出结果可视化。
完成前面模块的运行后，运行该节点，勾选编辑，点击编辑操作编辑VS code ，进入编辑页面。

主函数：

```
# coding=utf-8
from __future__ import absolute_import, print_function
import pandas as pd
import suanpan
from suanpan.docker import DockerComponent as dc
from suanpan.docker.arguments import Folder,File
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

# 定义输入
@dc.input(Folder(key="inputData1", required=True))
# 定义输出
@dc.output(File(key="outputData1",type='png',name='picture',requi
red=True))
def Demo(context):
    # 从 Context 中获取相关数据
    args = context.args
    # 查看上一节点发送的 args.inputData1 数据
    #print(args.inputData1)
    data=pd.read_csv(args.inputData1+'/data.csv')
    #print(data1)
    data1=data.drop(['number','prediction'],axis=1)
    #data2=data1.drop(['prediction'],axis=1)
    #print(data1)
    plt.rcParams['font.sans-serif'] = ['SimHei'] #用来正常显示中文标
签
    plt.rcParams['axes.unicode_minus'] = False #用来正常显示负号
    #不同类别用不同颜色和样式绘图
    d = data1[data['prediction'] == 0]
    plt.plot(d['density'],d['sugercontent'], 'r.')
    d = data1[data['prediction'] == 1]
```

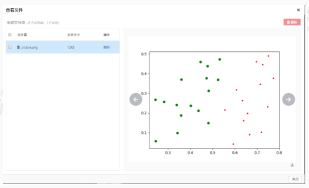
```
    plt.plot(d['density'],d['sugercontent'], 'go')
    #plt.show()
    plt.savefig(args.outputData1)
    # 自定义代码

    # 将 args.outputData1 作为输出发送给下一节点
    return args.outputData1


if __name__ == "__main__":
     suanpan.run(Demo)
```

需求：

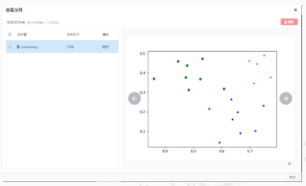
matplotlib

完成编辑，运行，查看结果。

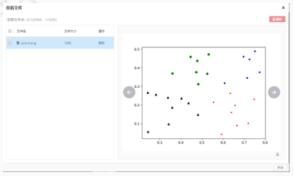
K=2



K=3



K=4

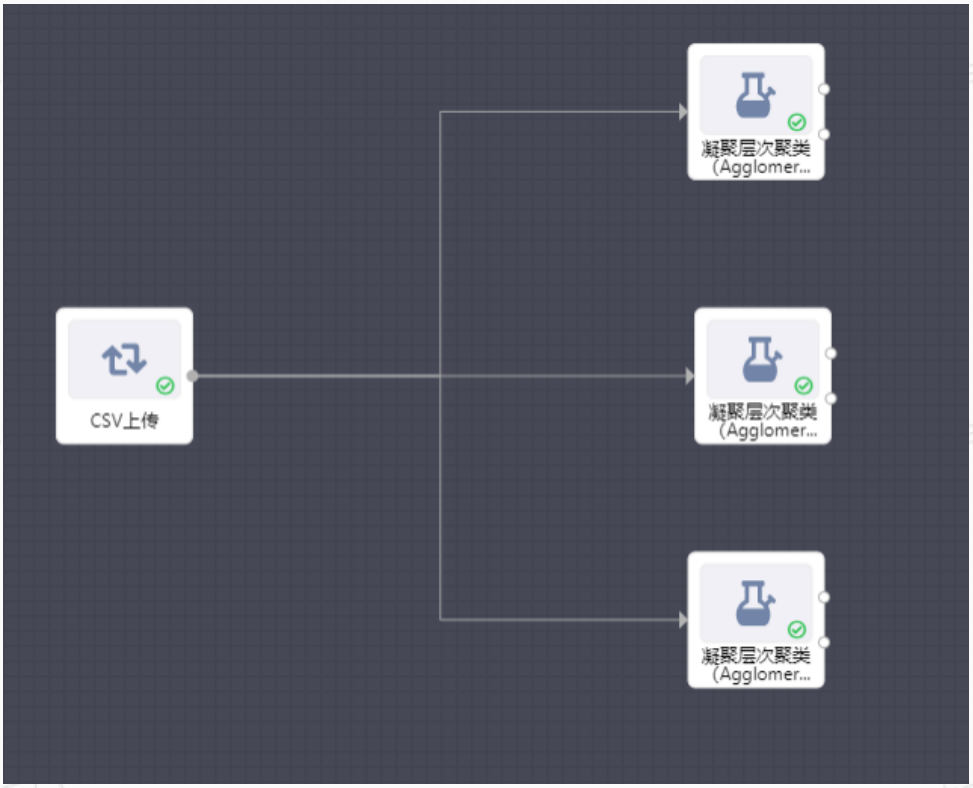


Tips：初始随机的中心点不同会导致算法的迭代次数与最终结果有很大的不同。

一般来说，初始的中心点越集中且越靠近边缘，则会使得迭代次数更多。初始中心点越分散，迭代次数越少，结果越好。

聚类算法案例二 例题9.6

9.6 试析AGNES算法使用最小距离和最大距离的区别。



最大距离可以认为是所有类别先生成一个能包围所有类内样本的最小圆，然后所有圆同时慢慢扩大相同的半径，哪个类圆能完全包围另一个类则停止，并合并这两个类。由于此时的圆已经包含另一个类的全部样本，所以称为全连接。

最小距离则是扩大时遇到第一个非自己类的点就停止，并合并这两个类。由于此时的圆只包含另一个类的一个点，所以称为单连接。

当两个类簇比较大且距离比较远，但是有两个点距离对方比较近时，那么单链接算法会把这两个类簇合并，导致产生拉长的类簇而不是一般情况下的圆形类簇，这被称为链式效应。因为这个算法经常由于链式效应而把不相似的对象放到同一类簇中，所以是空间压缩的(space contracting)。

当两个类簇中至少有一对比较远离的对象时，全链接算法会最后合并这两个类簇，于是相似对象会长时间待在不同类簇中，这被称为分离效果(dissection effect)。所以，全链接算法是空间扩张的(space dilating)。

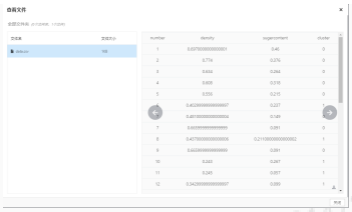
最小距离: $d_{\min}(C_i, C_j) = \min_{\boldsymbol{x} \in C_i, \boldsymbol{z} \in C_j} \operatorname{dist}(\boldsymbol{x}, \boldsymbol{z})$,

最大距离: $d_{\max}(C_i, C_j) = \max_{\boldsymbol{x} \in C_i, \boldsymbol{z} \in C_j} \operatorname{dist}(\boldsymbol{x}, \boldsymbol{z})$,

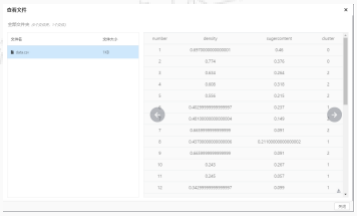
平均距离: $d_{\text{avg}}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{\boldsymbol{x} \in C_i} \sum_{\boldsymbol{z} \in C_j} \operatorname{dist}(\boldsymbol{x}, \boldsymbol{z})$.

当聚类簇距离由最小距离，最大距离，平均距离计算时，AGNES算法被相应地称为单链接single-linkage，全连接complete-linkage或均连接average-linkage算法。

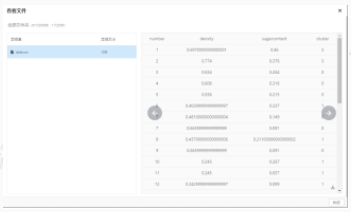
K=2



K=3



K=4



案例9.4 视频教程

9.4.mov

题目：试编程实现k均值算法，设置三组不同的k值、三组不同初始中心点，在西瓜数据集4.0上进行试验比较，并讨论什么样的初始中心有利于取得好结果。

步骤1：在“项目模板”中找到周志华习题，打开第九章聚类习题案例9.4，创建模板。

步骤2：在“CSV上传模块”上传data数据，一般为csv格式。这里就是上传需要输入的样本集D。

步骤3：在全部组件列表搜索“K均值聚类”模块，拖到操作界面内，

这个就是k-means算法的实现，k-means算法是给定样本集D，然后通过迭代优化针对聚类所得簇划分C最小化平方误差E，这是一种基于划分的聚类算法，计算量大，但很容易发现数据库中的球状簇。

“k均值聚类”模块左端需要输入训练数据，通过k-means算法会输出数据和模型。

在“参数设置”里把“N clusters”设置好聚类簇数k，其他参数就不用动了。在“字段设置”中设置“特征字段”“density,sugercontent”，在类别字段中设置“cluster”。“执行调优”选项卡中不用填。

然后把CSV上传模块和K均值聚类模块连接，这里可以点击运行查看下结果。

步骤4：在全部组件列表搜索“模型预测”模块，连接“K均值聚类”和“模型预测”。

滚轮放大视图可以看到“模型预测”组件左边是接收2个输入的，这个模板里一个是我们上传的需要预测的数据，另一个是经过k-means算法训练的模型。通过这个模块直接可以查看预测结果。预测结果为csv格式。在“公共组件”的“算法设计”选项里，找到数据可视化，可以实现预测结果的可视化。

步骤5：在全部组件列表搜索“VS Code – Python”，这里我们使用Python来实现结局可视化，用散点图来表示最终结果。

点击“VS Code Python”，在“参数设置”里勾选“编辑”，然后点击运行该节点，然后在“概览”中点击操作“编辑 VS Code”，进入编辑页面，然后编写主函数及matplot。

步骤6：在编写完VS Code后别忘了将“参数设置”里的“编辑”勾选掉。然后点击“CSV上传”，从此处运行开始下载项目和启动项目。

步骤7：运行完程序，可以再“聚类可视化”模块中查看结果。

案例10.1 视频教程

10.1.mp4

题目：编程实现k近邻分类器，在西瓜数据集3.0α上比较其分类边界与决策树分类边界之异同。

步骤1：在“项目模板”中找到周志华习题，打开第十章降维与度量学习习题案例10.1，创建模板。

步骤2：在“EXCEL上传”模块上传excel文件，这里用到的是西瓜数据集3.0α。

步骤3：在全部组件列表搜索“VS Code Python”模块，拖到操作界面内，在“参数设置”中勾选上“编辑”，在点击运行该节点，在操作中“编辑VS Code”，编写将excel文件的数据转换为CSV格式的文件的代码。注意编写完之后要重新勾掉“编辑”选项，才可以运行程序。

步骤4：在全部组件列表搜索“序数编码”模块，拖到操作界面内，在字段设置的“目标字段”中输入“色泽，根蒂，敲声，纹理，脐部，触感，好瓜”特征字段。该模块主要用来将分类特征编码为整数数组。该模块的输入应为整数或者字符串的数组，表示分类特征采用的值。目的是找到每个特征的唯一值，然后将数据转换为叙述编码。

步骤5：在全部组件列表搜索“决策树分类”组件，参数设置均为默认值不做选择，设置好特征字段和标识字段。决策树只有水平和垂直边界。

步骤6：同样的，在全部组件列表搜索“k近邻分类”组件，参数设置均为默认值不做选择，设置好特征字段和标识字段。k近邻分类器是在该样本最近的k个样本集合中，选择分类最多的一个作为该样本的分类。k近邻分类器不仅有水平和垂直边界不同，还可以有曲线边界。

步骤7： 在全部组件列表中搜索“模型预测”组件，将经过标签编码的数据集作为测试集，经过k近邻分类训练和决策树分类训练的数据集分别作为训练集，作为模型预测组件的输入。预测的数据存储在该组件中，可以通过其他可视化组件来查看更清晰的结果。

案例10.6 视频教程

10.6.mp4

题目：试使用MATLAB中的PCA函数对Yale人脸数据集进行降维，并观察前20个特征向量所对应的图像。

步骤1：在“项目模板”中找到周志华习题，打开第十章降维与度量学习习题案例10.6，创建模板。

步骤2：在“文件夹上传”模块上传文件夹，这里用到的是yale人脸数据集。Yale人脸数据集捡

http：//vision.ucsd.edu/content/yale-face-database.

步骤3：通过VS Code Python 组件来实现图片矩阵化，“参数设置”中勾选上“编辑”，在点击运行该节点，在操作中“编辑VS Code”，编写将图片矩阵化的代码。注意编写完之后要重新勾掉“编辑”选项，才可以运行程序。模板中将文件夹的信息转换成了一个CSV数据文件。因为矩阵化的图片再进行降维操作可以将矩阵拉伸为向量，再进行降维操作有更好的性能。

步骤4：在全部组件列表中搜索“主成分分析PCA”组件，主要是使用数据的奇异值分解将线性位数减伤以将其投影到较地位空间。N_components为要保留的组件数，未设置则是保留所有组件。默认是自动选择求解器。如果输入数据大于500*500，且需提取的组件数小于数据最小维度的80%，则启用效率更高的“随机化”方法。主成分分析是一种无监督的线性降维方法，监督线性降维方法最著名的是线性判别分析。特征字段设置为“0，1,2,3,4,5，6,7,8,9,10,11,12,13,14,15”。

案例11.1 视频教程

11.1.mp4

题目：试编程实现Relief算法，并考察其在西瓜数据集3.0上的运行结果。

步骤1：在“项目模板”中找到周志华习题，打开第十一章特征选择与稀疏学习习题案例11.1，创建模板。

步骤2：在“EXCEL上传”模块上传excel，这里用到的是西瓜数据集3.0.

步骤3：在全部组件列表搜索“VS Code Python”模块，拖到操作界面内，在“参数设置”中勾选上“编辑”，在点击运行该节点，在操作中“编辑VS Code”，编写将excel文件的数据转换为CSV格式的文件的代码。注意编写完之后要重新勾掉“编辑”选项，才可以运行程序。

步骤4：在全部组件列表搜索“序数编码”模块，拖到操作界面内，在字段设置的“目标字段”中输入“色泽，根蒂，敲声，纹理，脐部，触感，好瓜”特征字段。该模块主要用来将分类特征编码为整数数组。该模块的输入应为整数或者字符串的数组，表示分类特征采用的值。目的是找到每个特征的唯一值，然后将数据转换为叙述编码。

步骤5：第一种方法，在全部组件列表搜索“选择K个最高分特征SelectKbest”，参数设置中的score_func为函数接收两个数据并返回一对或单个数组，默认为f_classif，适用于分类任务。k表示要选择的主要功能数。这里设置k为5.表示最终剩余特征数为5。输出结果为五个特征的排序。

步骤6：第二种方法，在全部组件列表中找到“岭分类”组件，属于使用ridge回归的分类器，设置好特征字段。岭回归就是在基本的线性回归中加入了正则项。

步骤7：在全部组件列表中找到“递归式特征消除RFE”组件，是给定将权重分配给特征的外部估计器，目标是通过递归考虑越来越少的特征集来选择特征。参数设置中的，特征数表示要选择的功能数量。这里设置特征数为5，Step对应于每次迭代要删除的特征数。这里设置为1.每次删除1个特征数。设置好特征字段，注意不要将编号选进去。最终结果为5个特征值的排序。

步骤8：第三种方法，通过VS Code Python 组件来实现relief算法，“参数设置”中勾选上“编辑”，在点击运行该节点，在操作中“编辑VS Code”，编写将图片矩阵化的代码。注意编写完之后要重新勾掉“编辑”选项，才可以运行程序。输出数据为各个特征的权重值。可以根据这个值来删选特征。