项目总结

波士顿房价回归预测

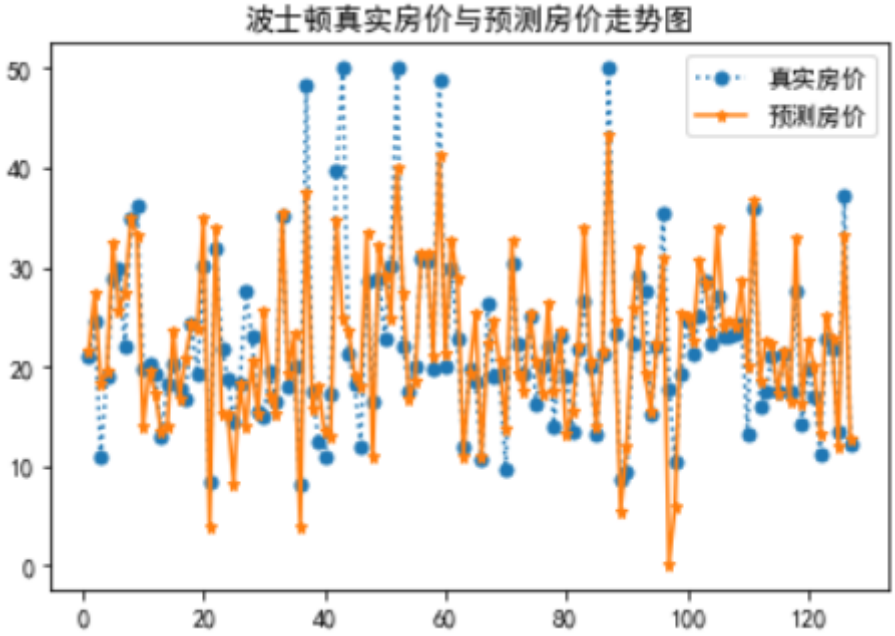
```
'''
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute
14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0
otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


This dataset was taken from the StatLib library which is maintained at Carnegie Mellon
University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.
```

```
The Boston house-price data has been used in many machine learning papers that address
regression
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and
Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In
Proceedings on the Tenth International Conference of Machine Learning, 236-243,
University of Massachusetts, Amherst. Morgan Kaufmann.
'''
```

线性回归

```
#导入波士顿房价模块
from sklearn.datasets import load_boston
#导入数据集拆分工具
from sklearn.model_selection import train_test_split

#导入boston数据集
boston = load_boston()
#将数据集分拆分为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target,
random_state=36)
#构建模型
lr_boston = LinearRegression()
#训练模型
lr_boston.fit(X_train, y_train)
#预测数据
y_predict = lr_boston.predict(X_test)
#导入均方误差评估模块
from sklearn.metrics import mean_squared_error
#均方误差评估
mse = mean_squared_error(y_test, y_predict)
print("线性回归模型的均方误差为：\n", mse)
'''
线性回归模型的均方误差为：
 24.004430052241922
'''
# 创建画布
plt.figure()
# 默认不支持中文
# 修改RC参数，来让其支持中文
plt.rcParams['font.sans-serif'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False
# 绘图
x = np.arange(1, y_predict.shape[0] + 1)
# 真实值的走势
plt.plot(x, y_test, marker="o", linestyle=":", markersize=5)
# 预测值的走势
plt.plot(x, y_predict, marker="*", markersize=5)
# 添加图例
plt.legend(["真实房价", "预测房价"])
# 添加标题
plt.title("波士顿真实房价与预测房价走势图")
# 展示
```

```
plt.show()
```

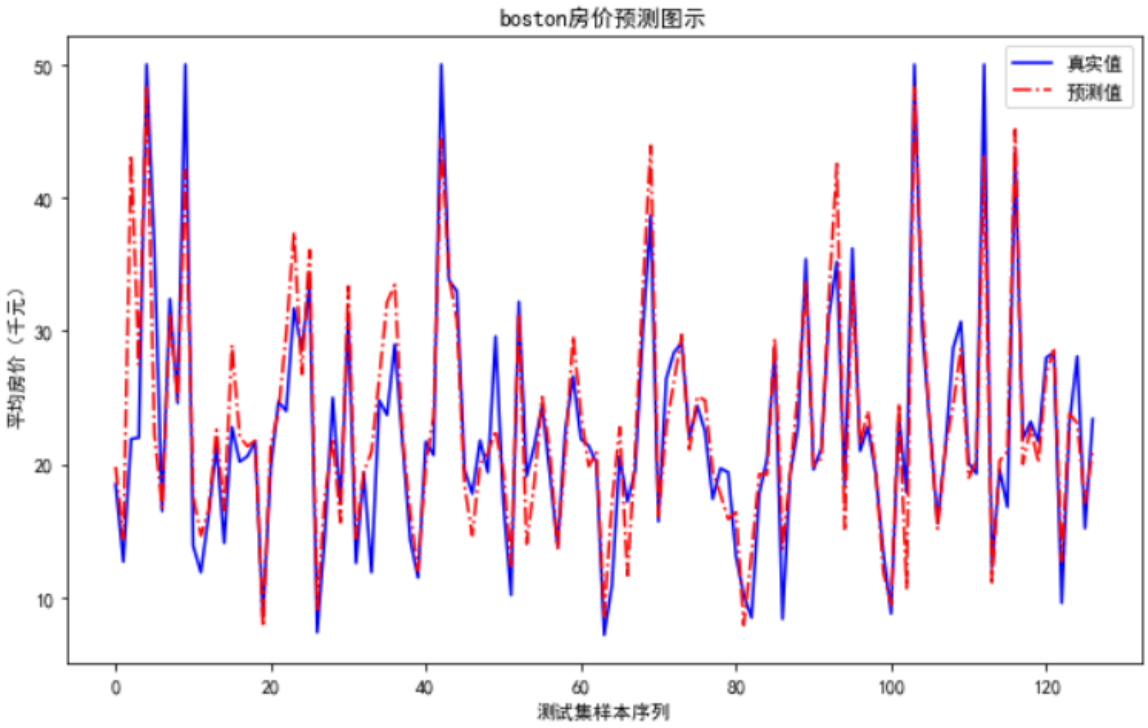



```
# 导入数据集拆分工具
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# 建立训练数据集和测试数据集
X, y = boston.data, boston.target
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=8)
reg = LinearRegression()      # 建立线性回归模型对象
reg.fit(X_train, y_train)     # 训练模型
# 打印模型评分
print('线性回归训练集得分：{:.3f}'.format(reg.score(X_train, y_train)))
print('线性回归测试集得分：{:.3f}'.format(reg.score(X_test, y_test)))
'''
线性回归训练集得分：0.744
线性回归测试集得分：0.719
'''
```

随机森林

```
# 导入波士顿房价数据集
from sklearn.datasets import load_boston
# 导入数据集拆分工具
from sklearn.model_selection import train_test_split
# 导入随机森林回归器
from sklearn.ensemble import RandomForestRegressor

boston = load_boston()
# 建立训练数据集和测试数据集
X, y = boston.data, boston.target
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=8)
# 使用随机森林回归模型拟合数据
rf_reg = RandomForestRegressor(n_estimators=20, random_state=32)
rf_reg.fit(X_train, y_train)
# 打印模型评分
```

```
print('随机森林训练集得分：{:.3f}'.format(rf_reg.score(X_train, y_train)))
print('随机森林测试集得分：{:.3f}'.format(rf_reg.score(X_test, y_test)))
'''
随机森林训练集得分：0.982
随机森林测试集得分：0.823
'''
%matplotlib inline
import matplotlib.pyplot as plt

# 预测测试集结果
y_pred = rf_reg.predict(X_test)
plt.rcParams['font.sans-serif'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False
plt.figure(figsize=(10, 6))
plt.plot(range(y_test.shape[0]), y_test, c='blue', linestyle='-', label='真实值')
plt.plot(range(y_test.shape[0]), y_pred, c='red', linestyle='-.', label='预测值')
plt.title('boston房价预测图示')
plt.xlabel('测试集样本序列')
plt.ylabel('平均房价（千元）')
plt.legend()
plt.show()
```



支持向量机

```
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# 导入数据集
boston = load_boston()
data = boston.data
target = boston.target
# 数据预处理
X_train,X_test,Y_train,Y_test = train_test_split(data,target,test_size=0.3)
```

```
# 特征进行标准化
Stand_X = StandardScaler()
# 标签也是数值，也需要进行标准化
Stand_Y = StandardScaler()
X_train = Stand_X.fit_transform(X_train)
X_test = Stand_X.transform(X_test)
Y_train = Stand_Y.fit_transform(Y_train.reshape(-1,1))
Y_test = Stand_Y.transform(Y_test.reshape(-1,1))
from sklearn.svm import SVR,LinearSVR
# 线性内核函数
clf = LinearSVR(C=1)
clf.fit(X_train,Y_train)
y_pred = clf.predict(X_test)
print("训练集评分：", clf.score(X_train,Y_train))
print("测试集评分：", clf.score(X_test,Y_test))
'''
训练集评分： 0.6820315908477788
测试集评分： 0.7822623925103901
'''
from sklearn.svm import SVR
# 高斯内核函数
clf = SVR(kernel='rbf',C=100,gamma=0.1)
clf.fit(X_train,Y_train)
y_pred = clf.predict(X_test)
print("训练集评分：", clf.score(X_train,Y_train))
print("测试集评分：", clf.score(X_test,Y_test))
'''
训练集评分： 0.9898779723613051
测试集评分： 0.8255846746793745
'''
```

乳腺肿瘤分类预测

```
'''
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)
```

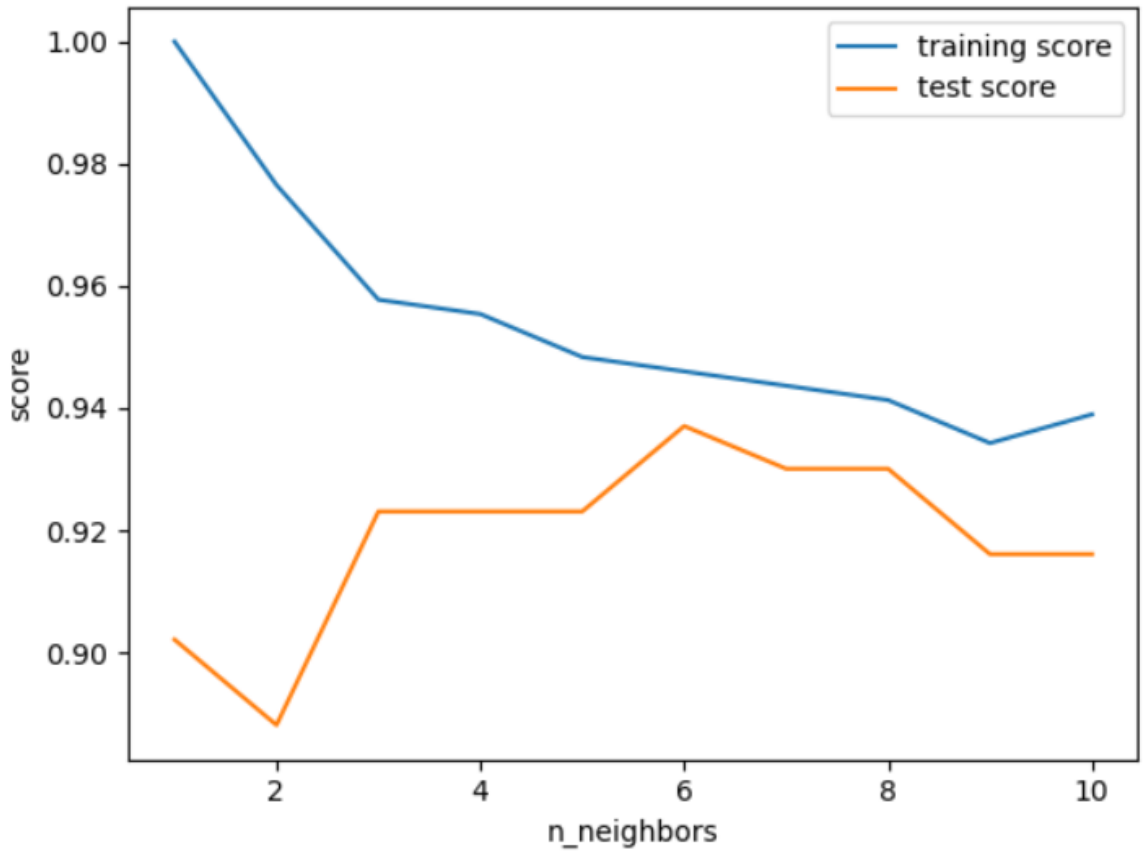
```
        The mean, standard error, and "worst" or largest (mean of the three
        worst/largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 0 is Mean Radius, field
        10 is Radius SE, field 20 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:

    ===================================== ====== ======
                                           Min    Max
    ===================================== ====== ======
    radius (mean):                        6.981  28.11
    texture (mean):                       9.71   39.28
    perimeter (mean):                     43.79  188.5
    area (mean):                          143.5  2501.0
    smoothness (mean):                    0.053  0.163
    compactness (mean):                   0.019  0.345
    concavity (mean):                     0.0    0.427
    concave points (mean):                0.0    0.201
    symmetry (mean):                      0.106  0.304
    fractal dimension (mean):             0.05   0.097
    radius (standard error):              0.112  2.873
    texture (standard error):             0.36   4.885
    perimeter (standard error):           0.757  21.98
    area (standard error):                6.802  542.2
    smoothness (standard error):          0.002  0.031
    compactness (standard error):         0.002  0.135
    concavity (standard error):           0.0    0.396
    concave points (standard error):      0.0    0.053
    symmetry (standard error):            0.008  0.079
    fractal dimension (standard error):   0.001  0.03
    radius (worst):                       7.93   36.04
    texture (worst):                      12.02  49.54
    perimeter (worst):                    50.41  251.2
    area (worst):                         185.2  4254.0
    smoothness (worst):                   0.071  0.223
    compactness (worst):                  0.027  1.058
    concavity (worst):                    0.0    1.252
    concave points (worst):               0.0    0.291
    symmetry (worst):                     0.156  0.664
    fractal dimension (worst):            0.055  0.208
    ===================================== ====== ======

    :Missing Attribute Values: None

    :Class Distribution: 212 - Malignant, 357 - Benign

    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

    :Donor: Nick Street

    :Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
```

```
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References

   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
     San Jose, CA, 1993.
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
     prognosis via linear programming. Operations Research, 43(4), pages 570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994)
     163-171.
'''
```

KNN

```python
#导入breast_cancer模块
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt

#从sklearn的datasets模块载入数据集
cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target,
stratify=cancer.target, random_state=66)
training_score = []
test_score = []
# n_nighbors取值为从1到10
neighbors_settings = range(1, 11)
for n_neighbors in neighbors_settings:
    # 构建模型
    clf = KNeighborsClassifier(n_neighbors=n_neighbors)
    clf.fit(X_train, y_train)
    # 记录训练集评分
```

```
    training_score.append(clf.score(X_train, y_train))
    # 记录测试集评分
    test_score.append(clf.score(X_test, y_test))
plt.plot(neighbors_settings, training_score, label="training score")
plt.plot(neighbors_settings, test_score, label="test score")
plt.ylabel("score")
plt.xlabel("n_neighbors")
plt.legend()
```



逻辑回归

```
# 导入breast_cancer[（美国）威斯康辛乳腺肿瘤]数据集
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
# 导入StandardScaler
from sklearn.preprocessing import StandardScaler
# 导入Logistic回归模型
from sklearn.linear_model import LogisticRegression

# 加载breast_cancer数据集
cancer = load_breast_cancer()
# “data”是特征数据
X = cancer.data
# “target”是目标变量数据(肿瘤的类别标签)
y = cancer.target
# 导入数据集拆分工具
# 将数据集拆分为训练数据集和测试数据集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=23)
# 对训练集进行拟合生成规则
```

```
scaler = StandardScaler().fit(X_train)
# 对训练集数据进行转换
X_train_scaled = scaler.transform(X_train)
# 对测试集数据进行转换
X_test_scaled = scaler.transform(X_test)
# 构建模型对象
log_reg = LogisticRegression(solver='lbfgs')
# 训练模型
log_reg.fit(X_train_scaled, y_train)
print('训练集得分：{:.2f}'.format(log_reg.score(X_train_scaled, y_train)))
'''训练集得分：0.99'''
# 测试集的准确率
test_score = log_reg.score(X_test_scaled, y_test)
# 预测类别标签
test_pred = log_reg.predict(X_test_scaled)
# 类别的概率估计
test_prob = log_reg.predict_proba(X_test_scaled)
print('测试集准确率为：{:.2f}'.format(test_score))
print('预测测试集前5个结果为：', test_pred[:5])
print('测试集前5个对应类别的概率为：\n', np.round(test_prob[:5], 3))
'''
测试集准确率为：0.98
预测测试集前5个结果为： [1 0 0 1 0]
测试集前5个对应类别的概率为：
 [[0.004 0.996]
 [0.54  0.46 ]
 [1.    0.   ]
 [0.034 0.966]
 [0.998 0.002]]
'''
```

支持向量机

```
#导入breast_cancer模块
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm

#从sklearn的datasets模块载入数据集
cancer = load_breast_cancer()
X_train, X_test, Y_train, Y_test = train_test_split(cancer.data, cancer.target,
                                                    stratify=cancer.target,
                                                    random_state=66)
# 标准化数据
nn = StandardScaler()
X_train = nn.fit_transform(X_train)
X_test = nn.transform(X_test)
svm_classifier = svm.SVC(C=1.0, kernel='rbf', decision_function_shape='ovr',
                         gamma='auto')
svm_classifier.fit(X_train, Y_train)
print("训练集:", svm_classifier.score(X_train, Y_train))
print("测试集:", svm_classifier.score(X_test, Y_test))
'''
训练集: 0.9859154929577465
测试集: 0.972027972027972
'''
```

神经网络

```
#导入breast_cancer数据集
from sklearn.datasets import load_breast_cancer
#从sklearn的datasets模块载入数据集
cancer = load_breast_cancer()
#导入数据集拆分工具
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(cancer.data,
        cancer.target, stratify=cancer.target, random_state=66)
# 标准化数据
from sklearn.preprocessing import StandardScaler
nn = StandardScaler()
X_train = nn.fit_transform(X_train)
X_test = nn.transform(X_test)
from sklearn.neural_network import MLPClassifier
mlp=MLPClassifier(solver='lbfgs',hidden_layer_sizes=[10,10],
                  activation='tanh',alpha=1)
mlp.fit(X_train,Y_train)
print("================================")
print('测试数据集得分：{:.2f}%'.format(mlp.score(X_test,Y_test)*100))
print("================================\n")
'''
================================
测试数据集得分：96.50%
================================
'''
```

鸢尾花分类预测

```
'''
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
```

```
                Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
'''
```

逻辑回归

```
# 导入必要的模块
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
# 加载iris（鸢尾花）数据集
iris = load_iris()
X = iris.data
y = iris.target
# 拆分训练集和测试集
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=8)
# 创建逻辑回归模型对象
clf1 = LogisticRegression(solver='liblinear', multi_class='ovr')
# 训练模型
clf1.fit(X_train, y_train)
# 评估模型
print('训练集得分：{:.2f}'.format(clf1.score(X_train, y_train)))
print('测试集得分：{:.2f}'.format(clf1.score(X_test, y_test)))
'''
训练集得分：0.96
测试集得分：0.95
'''
# 修改参数重新创建逻辑回归模型对象
clf2 = LogisticRegression(solver='lbfgs', multi_class='multinomial')
# 训练模型
clf2.fit(X_train, y_train)
# 评估模型
print('训练集得分：{:.2f}'.format(clf2.score(X_train, y_train)))
print('测试集得分：{:.2f}'.format(clf2.score(X_test, y_test)))
'''
训练集得分：0.97
测试集得分：0.92
'''
```

支持向量机

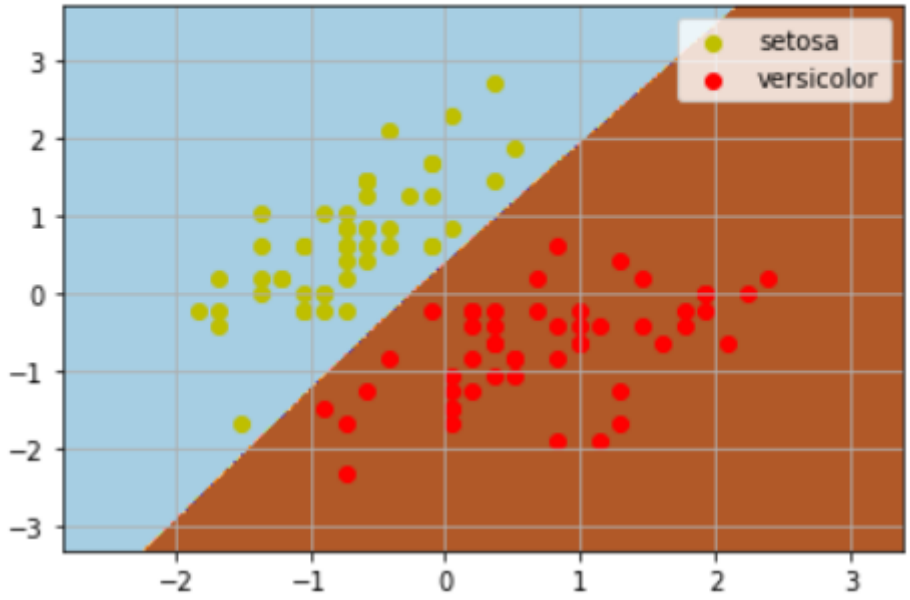
```
#导入数据
from sklearn import datasets
iris = datasets.load_iris()
# 数据集拆分
from sklearn.model_selection import train_test_split
# 特征
feature = iris.data
# 分类标签
label = iris.target
X_train, X_test, Y_train, Y_test = train_test_split(feature, label,
                                                    test_size=0.25,
                                                    random_state=62)
#定义模型
from sklearn import svm
svm_classifier = svm.SVC(C=1.0, kernel='rbf', decision_function_shape='ovr',
                         gamma='auto')
svm_classifier.fit(X_train, Y_train)
print("训练集:", svm_classifier.score(X_train, Y_train))
print("测试集:", svm_classifier.score(X_test, Y_test))
'''
训练集: 0.9910714285714286
测试集: 0.9473684210526315
'''
```

神经网络

```
import pandas as pd
```

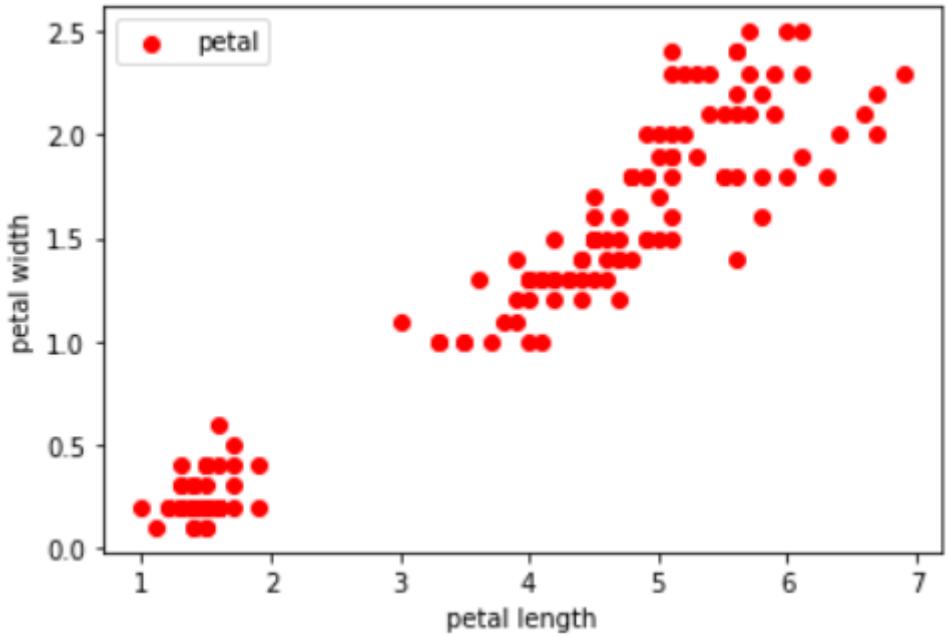
```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_iris
#iris数据集
dataset = load_iris()
data = pd.DataFrame(dataset.data, columns=dataset.feature_names)
data['class'] = dataset.target
#这里只取两类
data = data[data['class']!=2]
#这里取两个属性为例
scaler = StandardScaler()
X = data[['sepal length (cm)','sepal width (cm)']]
scaler.fit(X)
#标准化数据集
X = scaler.transform(X)
Y = data[['class']]
#将数据集拆分为训练集和测试集
X_train, X_test, Y_train, Y_test =train_test_split(X, Y)
#MLP分类器
mpl = MLPClassifier(solver='lbfgs',activation='relu')
mpl.fit(X_train, Y_train)
#显示分类的正确率
print ('score:\n',mpl.score(X_test, Y_test))
h = 0.02
x_min, x_max = X[:,0].min() - 1, X[:,0].max() + 1
y_min, y_max = X[:,1].min() - 1, X[:,1].max() + 1
nn, mm = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = mpl.predict(np.c_[nn.ravel(), mm.ravel()])
Z = Z.reshape(nn.shape)
plt.contourf(nn, mm, Z, cmap=plt.cm.Paired)
class1_x = X[Y['class']==0,0]
class1_y = X[Y['class']==0,1]
l1 = plt.scatter(class1_x,class1_y,color='y',label=dataset.target_names[0])
class2_x = X[Y['class']==1,0]
class2_y = X[Y['class']==1,1]
l2 = plt.scatter(class2_x,class2_y,color='r',label=dataset.target_names[1])
plt.legend(handles = [l1, l2], loc = 'best')
plt.grid(True)
plt.show()
```

```
score:
 1.0
```

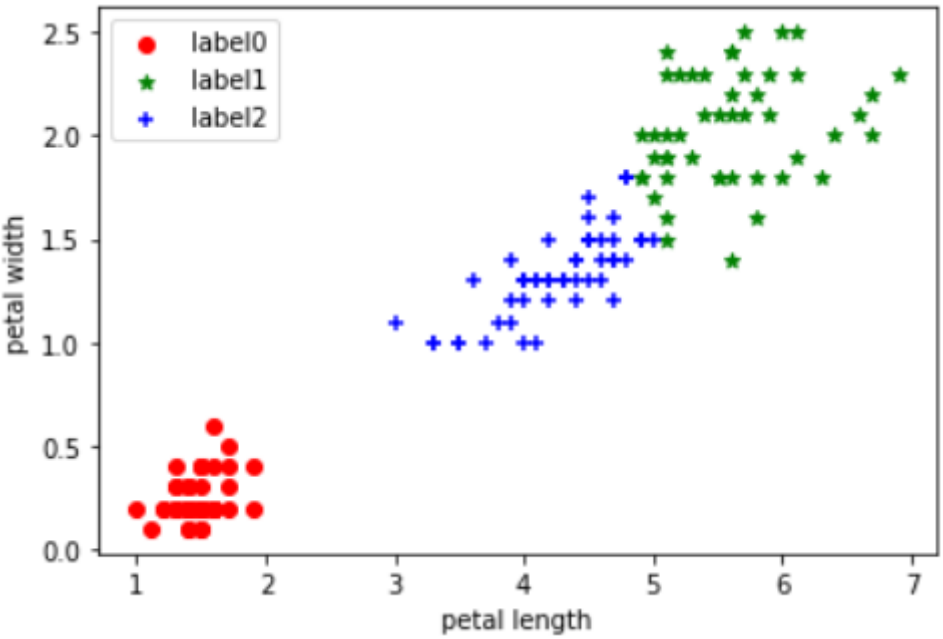


k-means

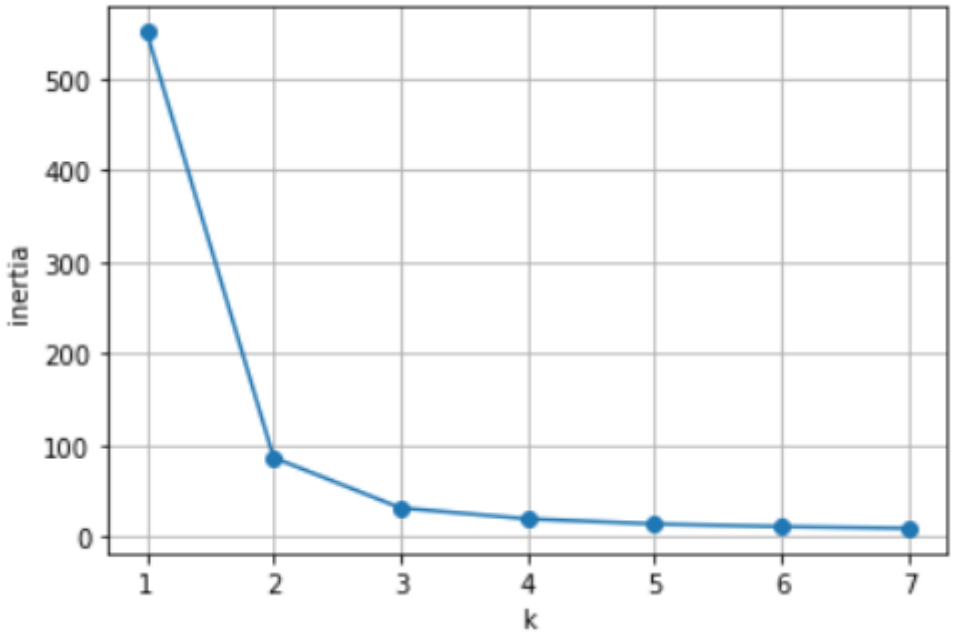
```python
import matplotlib.pyplot as plt
import numpy as np
#导入k-means工具
from sklearn.cluster import KMeans
from sklearn import datasets
#导入鸢尾花数据集
from sklearn.datasets import load_iris
iris=load_iris()
#我们只取特征空间中的后两个维度
X = iris.data[:, 2:4]
#绘制数据分布图
plt.scatter(X[:, 0], X[:, 1], c = "red", marker='o', label='petal')
plt.xlabel('petal length')
plt.ylabel('petal width')
plt.legend(loc=2)
plt.show()
```




```
#构造聚类器，将数据聚为3类
kmeans = KMeans(n_clusters=3)
#拟合数据
kmeans.fit(X)
#获取聚类标签
label_pred = kmeans.labels_
#绘制聚类效果图
x0 = X[label_pred == 0]
x1 = X[label_pred == 1]
x2 = X[label_pred == 2]
plt.scatter(x0[:, 0], x0[:, 1], c = "red", marker='o', label='label0')
plt.scatter(x1[:, 0], x1[:, 1], c = "green", marker='*', label='label1')
plt.scatter(x2[:, 0], x2[:, 1], c = "blue", marker='+', label='label2')
plt.xlabel('petal length')
plt.ylabel('petal width')
plt.legend(loc=2)
plt.show()
```



```
Inertia=[]  #存储k值对应的簇内距离平方和
for i in range(1,8):    #k取值1~8，做k-means聚类
    km=KMeans(n_clusters=i)
    km.fit(X)
    Inertia.append(km.inertia_)  #inertia_属性获取簇内距离平方和

plt.xlabel('k')
plt.ylabel('inertia')
plt.plot(range(1, 8),Inertia,'o-')
plt.grid(True)
plt.show()
```



红酒分类预测

```
.. _wine_dataset:

Wine recognition dataset
------------------------

**Data Set Characteristics:**

    :Number of Instances: 178 (50 in each of three classes)
    :Number of Attributes: 13 numeric, predictive attributes and the class
    :Attribute Information:
        - Alcohol
        - Malic acid
        - Ash
        - Alcalinity of ash
        - Magnesium
        - Total phenols
        - Flavanoids
        - Nonflavanoid phenols
        - Proanthocyanins
        - Color intensity
        - Hue
        - OD280/OD315 of diluted wines
        - Proline
```

```
    - class:
            - class_0
            - class_1
            - class_2

    :Summary Statistics:

    ============================== ==== ===== ======= =====
                                    Min   Max   Mean     SD
    ============================== ==== ===== ======= =====
    Alcohol:                      11.0  14.8    13.0   0.8
    Malic Acid:                   0.74  5.80    2.34  1.12
    Ash:                          1.36  3.23    2.36  0.27
    Alcalinity of Ash:            10.6  30.0    19.5   3.3
    Magnesium:                    70.0 162.0    99.7  14.3
    Total Phenols:                0.98  3.88    2.29  0.63
    Flavanoids:                   0.34  5.08    2.03  1.00
    Nonflavanoid Phenols:         0.13  0.66    0.36  0.12
    Proanthocyanins:              0.41  3.58    1.59  0.57
    Colour Intensity:              1.3  13.0     5.1   2.3
    Hue:                          0.48  1.71    0.96  0.23
    OD280/OD315 of diluted wines: 1.27  4.00    2.61  0.71
    Proline:                       278  1680     746   315
    ============================== ==== ===== ======= =====

    :Missing Attribute Values: None
    :Class Distribution: class_0 (59), class_1 (71), class_2 (48)
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

This is a copy of UCI ML Wine recognition datasets.
https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data

The data is the results of a chemical analysis of wines grown in the same
region in Italy by three different cultivators. There are thirteen different
measurements taken for different constituents found in the three types of
wine.

Original Owners:

Forina, M. et al, PARVUS -
An Extendible Package for Data Exploration, Classification and Correlation.
Institute of Pharmaceutical and Food Analysis and Technologies,
Via Brigata Salerno, 16147 Genoa, Italy.

Citation:

Lichman, M. (2013). UCI Machine Learning Repository
[https://archive.ics.uci.edu/ml]. Irvine, CA: University of California,
School of Information and Computer Science.

.. topic:: References

  (1) S. Aeberhard, D. Coomans and O. de Vel,
  Comparison of Classifiers in High Dimensional Settings,
  Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of
  Mathematics and Statistics, James Cook University of North Queensland.
  (Also submitted to Technometrics).
```

```
  The data was used with many others for comparing various
  classifiers. The classes are separable, though only RDA
  has achieved 100% correct classification.
  (RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-transformed data))
  (All results using the leave-one-out technique)

  (2) S. Aeberhard, D. Coomans and O. de Vel,
  "THE CLASSIFICATION PERFORMANCE OF RDA"
  Tech. Rep. no. 92-01, (1992), Dept. of Computer Science and Dept. of
  Mathematics and Statistics, James Cook University of North Queensland.
  (Also submitted to Journal of Chemometrics).
```

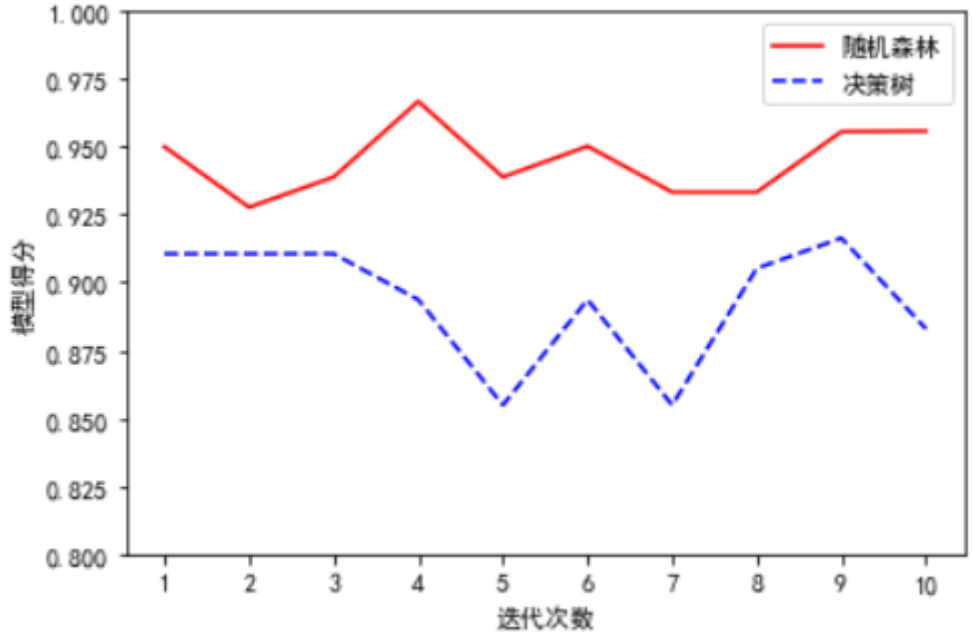
KNN

```
#导入wine模块
from sklearn.datasets import load_wine
#导入wine数据集
wine_dataset = load_wine()
#导入数据集拆分工具
from sklearn.model_selection import train_test_split

#将数据集拆分为训练集和测试集
X_train,X_test,y_train,y_test =
train_test_split(wine_dataset['data'],wine_dataset['target'],random_state=0)
#导入KNN算法分类模型
from sklearn.neighbors import KNeighborsClassifier
#指定模型的n_neighbors参数值为1
knn = KNeighborsClassifier(n_neighbors = 1)
#训练模型
knn.fit(X_train,y_train)
#输出模型得分
print("================================")
print('测试集得分：{:.2f}'.format(knn.score(X_test,y_test)))
print("================================\n")
'''
================================
测试集得分：0.76
================================
'''
```

决策树与随机森林

```
import numpy as np
import pandas as pd
# 导入数据集获取工具
from sklearn.datasets import load_wine
#用于拆分训练数据和样本数据
from sklearn.model_selection import train_test_split

# 读取酒的数据集
wine = load_wine()
X = wine.data
y = wine.target
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

# 决策树和随机森林模型
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# 对比随机森林分类器vs决策树
clf = DecisionTreeClassifier(max_depth=3)
rfc = RandomForestClassifier(n_estimators=20, random_state=8)
clf = clf.fit(X_train, y_train)
rfc = rfc.fit(X_train, y_train)
score_c = clf.score(X_test, y_test)
score_r = rfc.score(X_test, y_test)

print("决策树模型得分：{:.2f}".format(score_c))
print("随机森林模型得分：{:.2f}".format(score_r))
'''
决策树模型得分：0.93
随机森林模型得分：0.98
'''
%matplotlib inline
import matplotlib.pyplot as plt
# 用于交叉验证
from sklearn.model_selection import cross_val_score
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

#为了观察更稳定的结果，下面进行十组交叉验证
rfc_l = []
clf_l = []
for i in range(10):
    rfc = RandomForestClassifier(n_estimators=20)
    rfc_s = cross_val_score(rfc, X, y, cv=3).mean()
    rfc_l.append(rfc_s)
    clf = DecisionTreeClassifier(max_depth=3)
    clf_s = cross_val_score(clf, X, y, cv=3).mean()
    clf_l.append(clf_s)

# 绘制交叉验证评分图
plt.figure()
plt.plot(range(1,11), rfc_l, 'r-', label = "随机森林")
plt.plot(range(1,11), clf_l, 'b--', label = "决策树")
plt.xticks(ticks=range(1,11))
plt.ylim((0.8, 1))
plt.xlabel('迭代次数')
plt.ylabel('模型得分')
plt.legend()
plt.show()
```



糖尿病患者病情预测

```
'''
.. _diabetes_dataset:

Diabetes dataset
----------------

Ten baseline variables, age, sex, body mass index, average blood
pressure, and six blood serum measurements were obtained for each of n =
442 diabetes patients, as well as the response of interest, a
quantitative measure of disease progression one year after baseline.

**Data Set Characteristics:**

  :Number of Instances: 442

  :Number of Attributes: First 10 columns are numeric predictive values

  :Target: Column 11 is a quantitative measure of disease progression one year after
baseline

  :Attribute Information:
      - age     age in years
      - sex
      - bmi     body mass index
      - bp      average blood pressure
      - s1      tc, T-Cells (a type of white blood cells)
      - s2      ldl, low-density lipoproteins
      - s3      hdl, high-density lipoproteins
      - s4      tch, thyroid stimulating hormone
      - s5      ltg, lamotrigine
      - s6      glu, blood sugar level

Note: Each of these 10 feature variables have been mean centered and scaled by the
standard deviation times `n_samples` (i.e. the sum of squares of each column totals
1).
```

```
Source URL:
https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

For more information see:
Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle
Regression," Annals of Statistics (with discussion), 407-499.
(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)
'''
```

线性回归

```
#导入糖尿病患者病情模块
from sklearn.datasets import load_diabetes
#导入数据集拆分工具
from sklearn.model_selection import train_test_split
#导入绘图模块
import matplotlib.pyplot as plt
#均方误差评估模块
from sklearn.metrics import mean_squared_error

#导入数据集
diabetes = load_diabetes()
#导入特征变量
X = diabetes['data']
#导入目标变量
y = diabetes['target']
#将数据集拆分为训练集和测试集
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=8)
#创建线性模型对象
diabetes_lr = LinearRegression()
#训练模型
diabetes_lr.fit(X_train,y_train)
#预测数据
y_predict = diabetes_lr.predict(X_test)
mse = mean_squared_error(y_test, y_predict)
print("均方误差为：\n", mse)
'''
均方误差为：
 3108.04109825321
'''
```

个人收入等级预测

```
shape:(32561, 15)
target:
<=50K   24720
>50K    7841
```

高斯朴素贝叶斯

```
# 读取数据
import pandas as pd
# 用pandas打开csv文件
df = pd.read_csv('adult.csv', header=None, index_col=False,
                 names=['年龄','单位性质','权重','学历','受教育时长',
                        '婚姻状况','职业','家庭情况','种族','性别',
                        '资产所得','资产损失','周工作时长','原籍',
                        '收入'])
display(df.head())
```

	年龄	单位性质	权重	学历	受教育时长	婚姻状况	职业	家庭情况	种族	性别	资产所得	资产损失	周工作时长	原籍	收入
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

```
df.groupby(by='收入').agg({'收入': 'count'})
# 根据‘收入’进行分组
group_income = df.groupby(by='收入')
# 收入<=50K的分组
income_lessthan50k = dict([x for x in group_income])[' <=50K']
# 收入>50K的分组
income_morethan50k = dict([x for x in group_income])[' >50K']
# 合并数据分组并排序
data = pd.concat([income_lessthan50k[:10000], income_morethan50k], axis=0)
data = data.sort_index()
display(data[:10])
```

	年龄	单位性质	权重	学历	受教育时长	婚姻状况	职业	家庭情况	种族	性别	资产所得	资产损失	周工作时长	原籍	收入
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
5	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
6	49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
7	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States	>50K
8	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States	>50K
9	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178	0	40	United-States	>50K

```
import numpy as np
from sklearn.preprocessing import LabelEncoder

# 定义一个用于标签编码的函数
def get_data_encoded(data):
    # 将数据全部转为字符类型
    data = np.array(data.astype(str))
    # 定义标记编码器对象列表
    encoder_list = []
    # 准备一个数组存储数据集编码后的结果
    data_encoded = np.empty(data.shape)
    # 将字符串转换为数值数据
```

```
    for i, item in enumerate(data[0]):
        # 判断该特征向量是否为数值数据
        if item.isdigit():
            data_encoded[:, i] = data[:, i]
        # 如果不是数值数据则进行标记编码
        else:
            # 将所有的标记编码器保存在列表中，以便在后面测试数据时使用
            encoder_list.append(LabelEncoder())
            # 将字符串数据的特征列逐个进行编码
            data_encoded[:, i] = encoder_list[-1].fit_transform(data[:, i])

    # 返回数据编码结果和编码器列表
    return data_encoded, encoder_list

data_encoded, encoder_list = get_data_encoded(data)

# 将编码处理完成的数据集拆分成特征矩阵X和类别标签y
X = data_encoded[:, :-1].astype(int)
# 数据集最后一列“收入”作为分类的类别标签，‘<=50K’为0，‘>50K’为1
y = data_encoded[:, -1].astype(int)
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# 拆分数据集为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=5)
# 对数值进行预处理
scaler = StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 使用高斯朴素贝叶斯拟合数据
gnb = GaussianNB()
gnb.fit(X_train_scaled, y_train)
# 打印模型评分
print('训练集得分：{:.3f}'.format(gnb.score(X_train_scaled, y_train)))
print('测试集得分：{:.3f}'.format(gnb.score(X_test_scaled, y_test)))
'''
训练集得分：0.713
测试集得分：0.720
'''
```

手写数字识别

神经网络

```
#导入数据集
from scipy.io import loadmat
data = loadmat('F:/10_digital.mat')
data
```

```
{'__header__': b'MATLAB 5.0 MAT-file, Platform: GLNXA64, Created on: Sun Oct 16 13:09:09 2011',
'__version__': '1.0',
'__globals__': [],
'X': array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]]),
'y': array([[10],
       [10],
       [10],
       ...,
       [ 9],
       [ 9],
       [ 9]], dtype=uint8)}
```

```python
#处理数据
from sklearn.preprocessing import StandardScaler
# 把X、y转化为数组形式，以便于计算
X = data['X']
Y = data['y']
#X、Y的形状
X.shape, Y.shape
# 标准化转换
scaler = StandardScaler()
# 训练标准化对象
scaler.fit(X)
# 转换数据集
X = scaler.transform(X)
#模型训练
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.utils import check_random_state
#以25%的数据构建测试样本，剩余作为训练样本
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.25,
                                               random_state =2)
X_train.shape,X_test.shape,Y_train.shape,Y_test.shape
mlp = MLPClassifier(solver='lbfgs',hidden_layer_sizes=[200,100],
                    activation='relu', alpha = 1,random_state=62)
mlp.fit(X_train,Y_train)
print("================================")
print('测试数据集得分：{:.2f}%'.format(mlp.score(X_test,Y_test)*100))
print("================================\n")
'''
================================
测试数据集得分：94.32%
================================
'''
```

航空公司客户价值预测

目标：使用航空公司客户数据，结合客户价值分析模型，采用k-means算法，对客户进行分群，比较不同类别客户的客户价值。
客户价值识别，应用最广泛的模型采用3个指标，即消费时间间隔（recency）、消费频率（frequency）、消费金额（monetary），简称RFM模型。它的作用是识别高价值的客户。
LRFMC模型指标含义如下。
L：会员入会时间距观测窗口结束的月数。
R：客户最近一次乘坐航空公司飞机距离观测窗口结束的月数。
F：客户在观测窗口内乘坐航空公司飞机的次数。
M：客户在观测窗口内累计的飞行总里程。
C：客户在观测窗口内乘坐仓位所对应的折扣系数的平均值。
shape:(52529, 44)

k-means

```
import numpy as np
import pandas as pd
data = pd.read_csv('air_data.csv',encoding='gb18030')
print(data.shape)
print(data)
```

	MEMBER_NO	FFP_DATE	FIRST_FLIGHT_DATE	GENDER	FFP_TIER	WORK_CITY	\
0	54993	2006/11/2	2008/12/24	男	6	.	
1	28065	2007/2/19	2007/8/3	男	6	NaN	
2	55106	2007/2/1	2007/8/30	男	6	.	
3	21189	2008/8/22	2008/8/23	男	5	Los Angeles	
4	39546	2009/4/10	2009/4/15	男	6	贵阳	
...	...	...	...	...	...	...	
52524	16415	2013/1/20	2013/1/20	女	4	北京	
52525	18375	2011/5/20	2013/6/5	女	4	广州	
52526	36041	2010/3/8	2013/9/14	男	4	佛山	
52527	45690	2006/3/30	2006/12/2	女	4	广州	
52528	61027	2013/2/6	2013/2/14	女	4	广州	

	WORK_PROVINCE	WORK_COUNTRY	AGE	LOAD_TIME	...	ADD_Point_SUM	\
0	北京	CN	31.0	2014/3/31	...	39992	
1	北京	CN	42.0	2014/3/31	...	12000	
2	北京	CN	40.0	2014/3/31	...	15491	
3	CA	US	64.0	2014/3/31	...	0	
4	贵州	CN	48.0	2014/3/31	...	22704	
...	...	...	...	...	...	...	
52524	.	CN	35.0	2014/3/31	...	0	
52525	广东	CN	25.0	2014/3/31	...	12318	
52526	广东	CN	38.0	2014/3/31	...	106972	
52527	广东	CN	43.0	2014/3/31	...	0	
52528	广东	CN	36.0	2014/3/31	...	0	

	Eli_Add_Point_Sum	L1Y_ELi_Add_Points	Points_Sum	L1Y_Points_Sum	\
0	114452	111100	619760	370211	
1	53288	53288	415768	238410	
2	55202	51711	406361	233798	
3	34890	34890	372204	186100	
4	64969	64969	338813	210365	
...	...	...	...	...	
52524	0	0	0	0	
52525	12318	12123	12318	12123	
52526	106972	56506	106972	56506	
52527	0	0	0	0	
52528	0	0	0	0	

```
data = data[[ 'LOAD_TIME', 'FFP_DATE', 'LAST_TO_END', 'FLIGHT_COUNT', 'SEG_KM_SUM',
'avg_discount']]
data
```

	LOAD_TIME	FFP_DATE	LAST_TO_END	FLIGHT_COUNT	SEG_KM_SUM	avg_discount
0	2014/3/31	2006/11/2	1	210	580717	0.961639
1	2014/3/31	2007/2/19	7	140	293678	1.252314
2	2014/3/31	2007/2/1	11	135	283712	1.254676
3	2014/3/31	2008/8/22	97	23	281336	1.090870
4	2014/3/31	2009/4/10	5	152	309928	0.970658
...	...	...	...	...	...	...
52524	2014/3/31	2013/1/20	437	2	3848	0.000000
52525	2014/3/31	2011/5/20	297	2	1134	0.000000
52526	2014/3/31	2010/3/8	89	4	8016	0.000000
52527	2014/3/31	2006/3/30	29	2	2594	0.000000
52528	2014/3/31	2013/2/6	400	2	3934	0.000000

52529 rows × 6 columns

```
#L指标，将 data 的'LOAD_TIME'列、'FFP_DATE'列转换为时间格式，取 data 的'LOAD_TIME'列减去 data 的'FFP_DATE'列，将结果赋值给变量 L
L = pd.to_datetime(data['LOAD_TIME']) - pd.to_datetime(data['FFP_DATE'])
#将 L 转为 str 类型，以逗号分隔，并取出第 0 个元素，赋值给变量 L
L = L.astype('str').str.split().str[0]
#将 L 转为 int 类型，除以30，得到月份数，并赋值给变量 L
L = L.astype('int')/30
#合并指标，提取'LAST_TO_END'、'FLIGHT_COUNT'、'SEG_KM_SUM'、'avg_discount'四列与L指标，利用pandas.concat()进行列合并，并赋值给变量features
features =
pd.concat((L,data[['LAST_TO_END','FLIGHT_COUNT','SEG_KM_SUM','avg_discount']]),axis=1)
#将变量 features 的列名重命名为 'L','R','F','M','C'
features.columns = ['L','R','F','M','C']
#输出变量 features
features
```

	L	R	F	M	C
0	90.200000	1	210	580717	0.961639
1	86.566667	7	140	293678	1.252314
2	87.166667	11	135	283712	1.254676
3	68.233333	97	23	281336	1.090870
4	60.533333	5	152	309928	0.970658
...	...	...	...	...	...
52524	14.500000	437	2	3848	0.000000
52525	34.866667	297	2	1134	0.000000
52526	49.466667	89	4	8016	0.000000
52527	97.433333	29	2	2594	0.000000
52528	13.933333	400	2	3934	0.000000

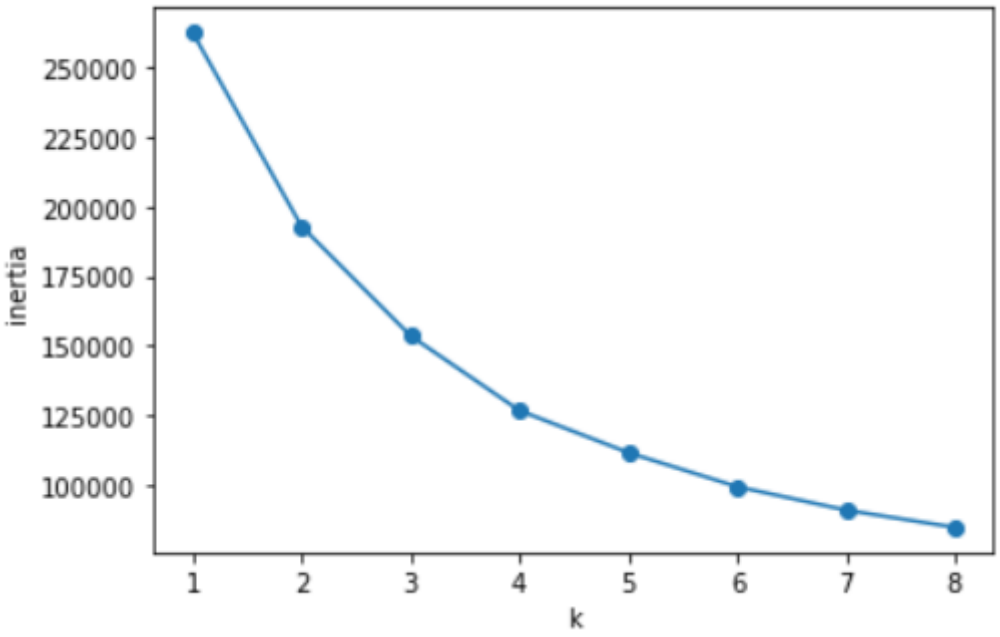
52529 rows × 5 columns

```
#查看各组数据的最大值、最小值
features.describe()
```

	L	R	F	M	C
count	52529.000000	52529.000000	52529.000000	52529.000000	52529.000000
mean	49.835265	184.031259	13.263416	19119.549715	0.728049
std	28.218882	191.047992	14.836998	22162.391961	0.182073
min	12.166667	1.000000	2.000000	368.000000	0.000000
25%	24.733333	28.000000	4.000000	5644.000000	0.620998
50%	42.600000	110.000000	8.000000	11831.000000	0.716170
75%	72.833333	295.000000	17.000000	24063.000000	0.810741
max	114.566667	731.000000	213.000000	580717.000000	1.500000

```
#导入标准化函数 StandardScaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
#对变量 features 进行标准化处理
features_scaler = scaler.fit_transform(features)
#模型构建
#导入k-means算法
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
Inertia = []
for k in range(1,9):
    estimator = KMeans(n_clusters=k)
    estimator.fit(features_scaler)
    Inertia.append(estimator.inertia_)#样本到最近的聚类中心的距离平方之和
X = range(1,9)
```

```
plt.xlabel('k')
plt.ylabel('inertia')
plt.plot(X,Inertia,'o-')
plt.show()
```

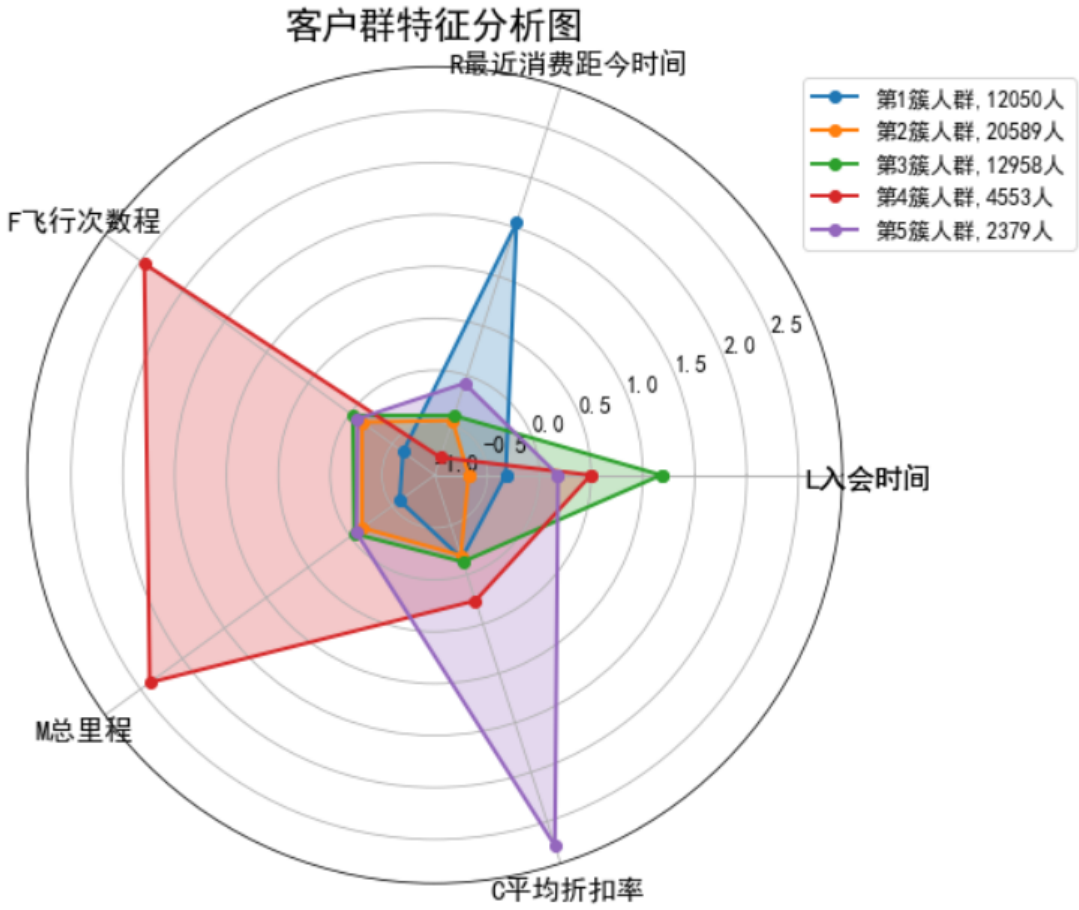



```
#构建k-means模型，聚类中心数 n_clusters=5
kmodel = KMeans(n_clusters=5)
#训练模型
kmodel.fit(features_scaler)
#查看样本的类别标签
print(kmodel.labels_)
'''[3, 3, 3, ..., 1, 2, 0]'''
import numpy as np
#统计各个类别的数目
r1 = pd.Series(kmodel.labels_).value_counts()
#找出聚类中心
r2 = pd.DataFrame(kmodel.cluster_centers_)
# 所有簇中心点坐标值中最大值和最小值
max = r2.values.max()
min = r2.values.min()
#横向连接，得到聚类中心对应的类别下的数目
r = pd.concat([r2,r1],axis=1)
r
```

	0	1	2	3	4	0
0	-0.324093	1.546683	-0.630392	-0.590496	-0.164311	12050
1	-0.681628	-0.456518	-0.141958	-0.143560	-0.196041	20589
2	1.176918	-0.406601	-0.032869	-0.048243	-0.130552	12958
3	0.496901	-0.830301	2.453945	2.391011	0.263406	4553
4	0.180998	-0.079775	-0.088250	-0.072395	2.744500	2379

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
```

```
# 绘制雷达图
# 中文和负号的正常显示
plt.rcParams['font.sans-serif'] = 'SimHei'
plt.rcParams['font.size'] = 12.0
plt.rcParams['axes.unicode_minus'] = False

fig = plt.figure(figsize=(10, 8))
# 以极坐标的形式绘制图形
ax = fig.add_subplot(111, polar=True)

# 设置特征标签名
feature_names = ["L入会时间", "R最近消费距今时间", "F飞行次数程", "M总里程", "C平均折扣
率"]
# 特征数目
N = len(feature_names)
# 设置雷达图的角度，用于平分切开一个圆面
angles = np.linspace(0, 2*np.pi, N, endpoint=False)
# 分割圆周长，并让其闭合
angles = np.concatenate((angles,[angles[0]]))
# 添加每个特征的标签名
labels = np.concatenate((feature_names,[feature_names[0]]))
ax.set_thetagrids(angles * 180/np.pi, labels, fontsize=15)

# 获取聚类分析结果
center_num = r.values
# 绘制每类人群的折线图
for i, v in enumerate(center_num):
    # 为了使雷达图一圈封闭起来，需要下面的步骤
    center = np.concatenate((v[:-1],[v[0]]))
    # 绘制折线图
    ax.plot(angles, center, 'o-', linewidth=2, label = "第%d簇人群,%d人"% (i+1,v[-1]))
    # 填充颜色
    ax.fill(angles, center, alpha=0.25)

# 添加标题
plt.title('客户群特征分析图', fontsize=20)
# 设置图例
plt.legend(loc='upper right', bbox_to_anchor=(1.3,1.0))
# 显示图形
plt.show()
```


客户群特征分析图
R最近消费距今时间
F飞行次数程
L入会时间
M总里程
C平均折扣率
第1簇人群, 12050人
第2簇人群, 20589人
第3簇人群, 12958人
第4簇人群, 4553人
第5簇人群, 2379人
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
-1.0