电子商务数据分析

第6章 使用Scikit-Learn进行商务数据挖掘

朱桂祥 (9120201070@nufe.edu.cn)
南京财经大学信息工程学院
江苏省电子商务重点实验室
电子商务信息处理国家级国际联合研究中心
电子商务交易技术国家地方联合工程实验室

校训是什么？
西南联大：刚毅坚卓 精神永驻

中国抗日战争开始后高校内迁设于昆明的一所综合性大学。1937年11月1日，由国立北京大学、国立清华大学、私立南开大学在长沙组建成立的国立长沙临时大学在长沙开学（这一天也成为西南联大校庆日）。由于长沙连遭日机轰炸，1938年2月中旬，经中华民国教育部批准，长沙临时大学分三路西迁昆明。1938年4月，改称国立西南联合大学。西南联大前后共存在了8年零11个月，“内树学术自由之规模，外来民主堡垒之称号”，保存了抗战时期的重要科研力量，培养了一大批卓有成就的优秀人才，为中国和世界的发展进步作出了杰出贡献。

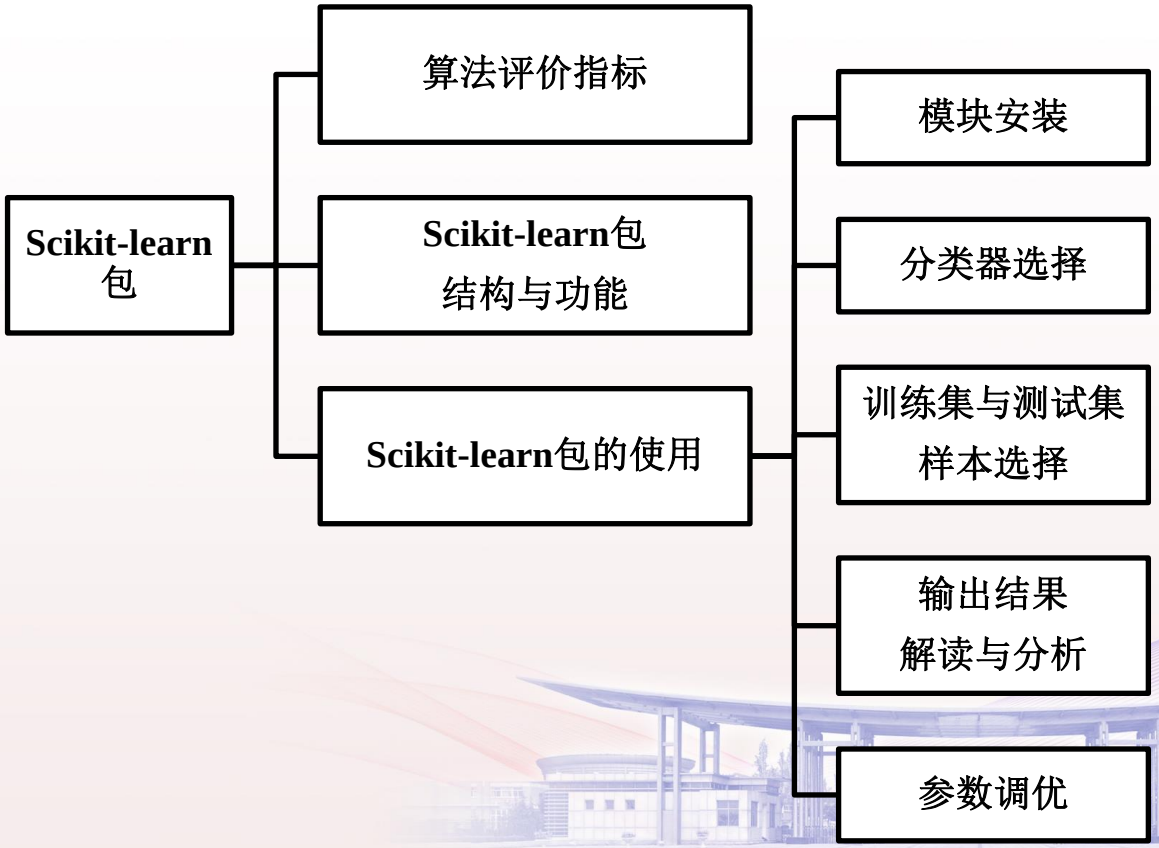
https://tv.cctv.com/2014/10/30/VIDE1414633443593241.shtml

目 录
Contents

01 Scikit-Learn简介

一、Scikit-Learn简介

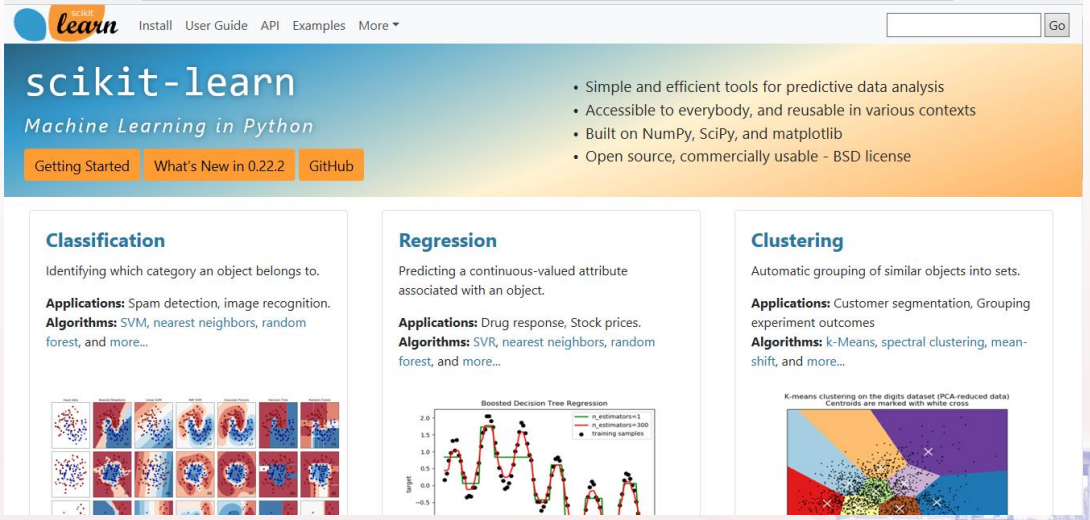
1. 知识框架图

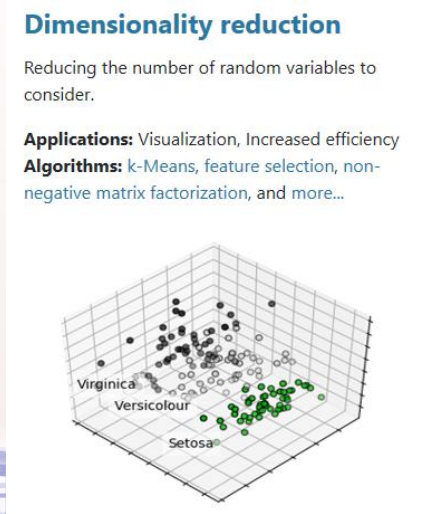


一、Scikit-Learn简介

2. Scikit-Learn的主要功能

- Scikit-Learn项目是由数据科学家 David Cournapeau 于 2007 年发起的，它目前已经成了Python语言中专门针对机器学习应用的一款开源框架，几乎覆盖了机器学习的所有主流算法。
- 其官方网址为http://Scikit-Learn.org/stable/ ，如下图所示。





一、Scikit-Learn简介

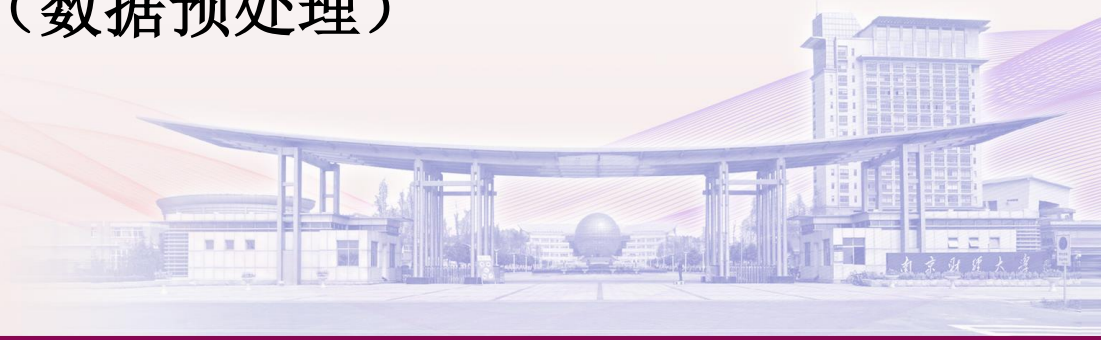
2. Scikit-Learn的主要功能

- 作为数据挖掘算法分析、设计和实施的开发人员，通常希望能够掌握算法的具体思路、实现流程和参数调优。
- 但对于参与数据挖掘工作的人而言，他们可能并不关注算法的实现细节，而更关注如何从业务分析的角度来选择重要的数据属性和合适的算法。
- Scikit-Learn为数据挖掘算法开发人员和业务专家参与算法与模型分析的人士提供了一个共同的接口，它是目前广受欢迎的简单高效的数据挖掘和数据分析工具。

一、Scikit-Learn简介

2.Scikit-Learn的主要功能

- **Scikit-Learn的主要功能包括如下六个方面：**
  - **（1）Classification（分类）**
  - **（2）Regression（回归）**
  - **（3）Clustering（聚类）**
  - **（4）Dimensionality Reduction**
  - **（5）Model Selection**
  - **（6）Preprocessing（数据预处理）**



一、Scikit-Learn简介

3. Scikit-Learn安装

- 使用Scikit-Learn进行数据挖掘时需要NumPy和SciPy等包的支持，因此在安装Scikit-Learn之前需要安装这些支持包。
- 请读者通过网址http://Scikit-Learn.org/stable/install.html查看Scikit-Learn的官方文档，了解所需安装的支持包具体列表和教程。
- 如果本机安装了Anaconda，则系统已默认安装好Scikit-Learn相关的包。
- 如果本机安装的是Miniconda，则用户需要运行以下命令来完成Scikit-Learn的安装：

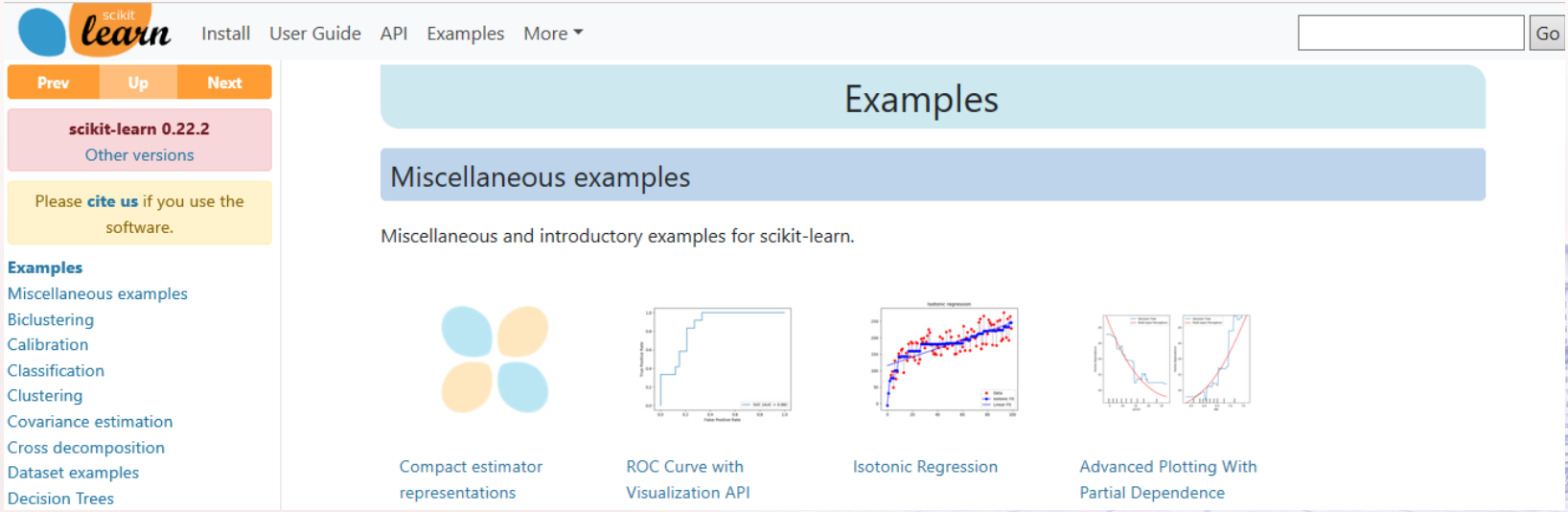
- conda install scikit-learn

附： 建议安装较新的scikit-learn 1.2.2版本

一、Scikit-Learn简介

3. Scikit-Learn安装

- 另外，通过Jupyter Notebook工具也可运行Scikit-Learn样例。
- 先从官方提供的样例库（http://Scikit-Learn.org/stable/auto_examples/index.html#general-examples）中选择一个样例，在该页面中下载其Python源码或者IPython notebook文件，将这些下载好的文件置于Jupyter Notebook运行就可以看到样例的结果。



02

使用Scikit-Learn进行数据挖掘

二、使用Scikit-Learn进行数据挖掘

1. 数据挖掘的基本流程：

(1)Look at the big picture：进行整体宏观考察，确定项目需求、目标和整体路径。

(2)Get the data：获取数据，可能来自于已有数据，也可能需要重新采集。

(3)Discover and visualize the data to gain insights：借助图表深入发现了解数据，可以通过Pandas读入数据文件，借助Matplotlib或者Seaborn绘制图表。

(4)Prepare the data for Machine Learning algorithms：进行数据整理（填充缺失值、处理异常/离群值、非数值型转换为数值型等）、属性选择（发现属性之间的相关性，进行重点分析，还可以通过丢弃非相关/弱相关属性，降低计算工作量）、训练集/测试集分离（选择部分数据用于训练算法模型，另一部分数据用于测试检查训练好的模型效果）等。

二、使用Scikit-Learn进行数据挖掘

1. 数据挖掘的基本流程：

(5)Select a model and train it：选择合适的数据挖掘模型，从各种数据挖掘算法中进行选择和尝试，并进行训练。

(6)Fine-tune your model：模型参数调优，进一步提升算法模型的效果。

(7)Present your solution.：提交解决方案。

(8)Launch, monitor, and maintain your system：发布、监控和维护系统。

二、使用Scikit-Learn进行数据挖掘

2. Scikit-Learn的主要模块：

- ✓ 预处理器（Sklearn Prepressing）：预处理主要分为规范化和编码，规范化主要包括最大最小值规范化（MinMaxScaler）、归一化（Normalize，使特征值和为1）和白化（StandartScaler，使特征值均值为0，方差为1）；编码主要包括LabelEncoder（字符串转化为整型）、OneHotEncoder（特征由一个二进制数字表示）、Binarizer（特征二值化）和MultiLabelBinarizer（多标签二值化）等。
- ✓ 转换器（Transformer）：用于数据预处理和数据转换。
- ✓ 估计器（Estimator）：估计器实际上就是封装成类的各种数据挖掘算法。每个算法都支持两个函数：fit(x,y) 和 predict(x)，分别用于训练和预测。以分类估计器为例，函数fit(x,y)中的x是训练集（没有分类标签），y则是x对应的分类标签，其参数格式均为numpy数组或类似格式。
- ✓ 模型评估（度量），交叉验证：主要包含评分方法，性能度量，成对度量和距离计算等。

二、使用Scikit-Learn进行数据挖掘

3. Scikit-Learn进行数据挖掘

以Iris数据集（鸢尾花数据集）的分类为例介绍使用Scikit-Learn完成数据挖掘工作的典型流程。

- Iris数据集是一个常见的用于测试数据挖掘分类算法效果的数据集，Scikit-Learn中可以直接使用load_iris()载入这个数据集。
- 该数据集一共有150行数据（samples），每行数据由4个特征和1个分类标签组成。4个特征分别是花萼片的长度、花萼片的宽度、花瓣的长度、花瓣的宽度；分类标签则记载了它的类属，即Iris Setosa（山鸢尾）、Iris Versicolour（杂色鸢尾），以及Iris Virginica（维吉尼亚鸢尾）等。

二、使用Scikit-Learn进行数据挖掘

3. Scikit-Learn进行数据挖掘

Iris数据:

萼片的长度 (Sepal Length)	萼片的宽度 (Sepal Width)	花瓣的长度 (Petal Length)	花瓣的宽度 (Petal Width)	类别 (Target)
4.8	3	1.4	0.3	setosa
5.1	3.8	1.6	0.2	setosa
4.6	3.2	1.4	0.2	setosa
5.3	3.7	1.5	0.2	setosa
5	3.3	1.4	0.2	setosa
……	……	……	……	……
6.5	3	5.2	2	virginica
6.2	3.4	5.4	2.3	virginica
5.9	3	5.1	1.8	virginica

二、使用Scikit-Learn进行数据挖掘

3. Scikit-Learn进行数据挖掘

- 载入数据：

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn import datasets

        iris = datasets.load_iris()
        print('Iris数据集前5行：')
        print(iris.data[0:5,:])
        print('Iris数据集前5行分类：')
        print(iris.target[0:5])
```

```
Iris数据集前5行：
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
Iris数据集前5行分类：
[0 0 0 0 0]
```

导入所需的模块
加载Iris数据
查看该数据集前5行

二、使用Scikit-Learn进行数据挖掘

3. Scikit-Learn进行数据挖掘

- 预处理阶段：

Iris数据集归一化处理

In [2]:
```
#预处理阶段
from sklearn.preprocessing import MinMaxScaler
#区间缩放，返回值为缩放到[0, 1]区间的数据
iris_data=MinMaxScaler().fit_transform(iris.data)
print('Iris数据集归一化处理后前5行：')
print(iris_data[0:5,:])
iris_df=pd.DataFrame(iris_data,columns=['Sepal Length',
                                        'Sepal Width','Petal Length','Petal Width'])
iris_df['target']=iris.target
```

```
Iris数据集归一化处理后前5行：
[[0.22222222 0.625      0.06779661 0.04166667]
 [0.16666667 0.41666667 0.06779661 0.04166667]
 [0.11111111 0.5        0.05084746 0.04166667]
 [0.08333333 0.45833333 0.08474576 0.04166667]
 [0.19444444 0.66666667 0.06779661 0.04166667]]
```

二、使用Scikit-Learn进行数据挖掘

3. Scikit-Learn进行数据挖掘

- 训练集和测试集划分

训练集与测试集

```
#训练集与测试集分离阶段
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(iris_df.iloc[:,0:4],
            iris_df['target'], random_state= 14)
print('Iris训练集前5行：')
print(X_train.head(5))
print('Iris训练集分类结果：')
print(y_train.head(5).values)
print('Iris测试集前5行：')
print(X_test.head(5))
print('Iris测试集原始分类结果')
print(y_test.head(5).values)
```

输出结果

```
Iris训练集前5行：
     Sepal Length  Sepal Width  Petal Length  Petal Width
119      0.472222     0.083333      0.677966     0.583333
51       0.583333     0.500000      0.593220     0.583333
125      0.805556     0.500000      0.847458     0.708333
46       0.222222     0.750000      0.101695     0.041667
15       0.388889     1.000000      0.084746     0.125000
Iris训练集分类结果：
[2 1 2 0 0]
Iris测试集前5行：
     Sepal Length  Sepal Width  Petal Length  Petal Width
24       0.138889     0.583333      0.152542     0.041667
42       0.027778     0.500000      0.050847     0.041667
6        0.083333     0.583333      0.067797     0.083333
53       0.333333     0.125000      0.508475     0.500000
113      0.388889     0.208333      0.677966     0.791667
Iris测试集原始分类结果
[0 0 0 1 2]
```

二、使用Scikit-Learn进行数据挖掘

3. Scikit-Learn进行数据挖掘

- **KNN算法实现：**

```
#算法实施阶段
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()     #估计器，使用KNN算法
knn.fit(X_train, y_train)
y_predicted = knn.predict(X_test)
print('Iris测试集真实结果')
print(y_test.values)
print('Iris测试集KNN算法预测结果')
print(y_predicted)
```

```
Iris测试集真实结果
[0 0 0 1 2 1 0 1 0 1 2 0 2 2 0 1 0 2 2 1 0 0 0 1 0 2 0 1 1 0 0 1 1 0 1 0 2
 1]
Iris测试集KNN算法预测结果
[0 0 0 1 2 1 0 1 0 1 1 0 2 2 0 1 0 2 2 1 0 0 0 1 0 2 0 1 1 0 0 1 1 0 1 0 2
 1]
```

```
#性能评估阶段
accuracy = np.mean(y_predicted == y_test) *100
print('当前分类评估器是：knn')
print('当前Accuracy是：%.1f' %accuracy + '%')
```

```
当前分类评估器是：knn
当前Accuracy是：97.4%
```

- Iris测试集真实结果
- Iris测试集KNN算法预测结果
- 分类评估
- 准确率Accuracy：97.4%

二、使用Scikit-Learn进行数据挖掘

3. Scikit-Learn进行数据挖掘

- 多种分类算法比较：

用多种内置算法对以上数据进行了训练和测试，并输出它们的Accuracy指标。

```
#与多种算法比较
from sklearn import tree, svm, naive_bayes,neighbors
from sklearn.ensemble import BaggingClassifier, AdaBoostClassifier, \
            RandomForestClassifier, GradientBoostingClassifier #随机森林等算法

clfs = {'svm': svm.SVC(gamma='auto'),\
        'decision_tree':tree.DecisionTreeClassifier(),
        'naive_gaussian': naive_bayes.GaussianNB(), \
        'naive_mul':naive_bayes.MultinomialNB(),\
        'K_neighbor' : neighbors.KNeighborsClassifier(),\
        'bagging_knn' : BaggingClassifier(neighbors.KNeighborsClassifier(), max_samples=0.5,max_features=0.5), \
        'bagging_tree': BaggingClassifier(tree.DecisionTreeClassifier(), max_samples=0.5,max_features=0.5),
        'random_forest' : RandomForestClassifier(n_estimators=50),\
        'adaboost':AdaBoostClassifier(n_estimators=50),\
        'gradient_boost' : GradientBoostingClassifier(n_estimators=50, learning_rate=1.0,max_depth=1, random_state=0)
        }

#用样本数据训练模型，用测试数据测试训练后的算法正确率
def try_different_method(clf):
    clf.fit(X_train,y_train)
    score = clf.score(X_test,y_test)*100 #默认情况下分类估计器的score函数返回的是Accrracy
    print('当前Accuracy是：%.1f'%score + '%')

#依次调用clfs中的不同算法
for clf_key in clfs.keys():
    print('当前分类评估器是:',clf_key)
    clf = clfs[clf_key]
    try_different_method(clf)
```

```
当前分类评估器是: svm
当前Accuracy是：97.4%
当前分类评估器是: decision_tree
当前Accuracy是：97.4%
当前分类评估器是: naive_gaussian
当前Accuracy是：97.4%
当前分类评估器是: naive_mul
当前Accuracy是：65.8%
当前分类评估器是: K_neighbor
当前Accuracy是：97.4%
当前分类评估器是: bagging_knn
当前Accuracy是：97.4%
当前分类评估器是: bagging_tree
当前Accuracy是：97.4%
当前分类评估器是: random_forest
当前Accuracy是：97.4%
当前分类评估器是: adaboost
当前Accuracy是：97.4%
当前分类评估器是: gradient_boost
当前Accuracy是：97.4%
```

03 案例

三、案例

1. 案例：房地产区域价格分析

- 通过抽样调查得到美国加利福利亚州的房屋价格数据（其部分数据如下表所示）。采集的字段包括房屋坐落区域经纬度、该区域的样本房屋使用年限中位数、样本房屋房间总数、样本房屋卧室总数、人口数量、样本户数、收入中位数、离海滩距离和样本房价中位数等。

longitude	latitude	housing_ median_ age	total_ rooms	total_ bedrooms	Popu lation	House holds	median_ income	ocean_ proximity	median_ house_ value
-122.23	37.88	41	880	129	322	126	8.3252	NEAR BAY	452600
-122.22	37.86	21	7099	1106	2401	1138	8.3014	NEAR BAY	358500
-122.24	37.85	52	1467	190	496	177	7.2574	NEAR BAY	352100
-122.25	37.85	52	1274	235	558	219	5.6431	NEAR BAY	341300
.....									

三、案例

1. 案例：房地产区域价格分析

本案例的任务是使用Python数据挖掘工具实现一个能以前9列数据为输入，以median_house_value为预测目标输出的模型。

```
import pandas as pd
import numpy as np

#加利福利亚房价数据
caDF = pd.read_csv('Section5 california_housing.csv')
print(caDF.shape)
print('加州房价数据前5行：')
print(caDF.head())

#观察数据，发现存在缺失值和非数值型字段
sample_incompletedata = caDF[caDF.isnull().any(axis=1)]
print('存在缺失值的数据前5行：')
print(sample_incompletedata.head())

#因为total_rooms和total_bedrooms线性相关性较强，
#为了减少运算量，将total_bedrooms列舍去
caDF.drop(['total_bedrooms'],axis=1,inplace=True)

#ocean_proximity列是文本标签，描述了房屋离海滩距离
#使用LabelEncoder将文本标签转换为数值标签
from sklearn.preprocessing import LabelEncoder
caDF[['ocean_proximity']]=caDF[['ocean_proximity']].apply\
                                    (LabelEncoder().fit_transform)
```

```
#使用SkLearn的Imputer处理缺失数据
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="median") #用该列中位数填充缺失值
imputer.fit(caDF)
#将所有数据转换为float类型
caDF=pd.DataFrame(caDF,dtype=np.float)
print('整理后数据前5行，请注意比原始数据少了一列：')
print(caDF.head())

#分离训练集与测试集，median_house_value列的数据是研究的目标
from sklearn.model_selection import train_test_split
caDFdata=caDF.drop(['median_house_value'],axis=1)
caDFprice=caDF['median_house_value']
Train_X,Test_X,Train_y,Test_y=train_test_split(caDFdata,caDFprice,
                                               test_size=0.2,random_state=42)

#数据标准化，将每一列数据转换为均值为0，方差为1的数据，便于后续处理
from sklearn.preprocessing import StandardScaler
ss= StandardScaler().fit(Train_X)
Train_X = pd.DataFrame(ss.transform(Train_X), columns=Train_X.columns)
Test_X  = pd.DataFrame(ss.transform(Test_X),  columns=Test_X.columns)

print('标准化后训练集前5行：')
print(Train_X.head())
```

三、案例

1. 案例：房地产区域价格分析

- 输出结果：

Out: (20640, 10)

加州房价数据前 5 行：

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	\
0	-122.23	37.88	41.0	880.0	129.0	
1	-122.22	37.86	21.0	7099.0	1106.0	
2	-122.24	37.85	52.0	1467.0	190.0	
3	-122.25	37.85	52.0	1274.0	235.0	
4	-122.25	37.85	52.0	1627.0	280.0	

	population	households	median_income	median_house_value	ocean_ proximity
0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	496.0	177.0	7.2574	352100.0	NEAR BAY
3	558.0	219.0	5.6431	341300.0	NEAR BAY
4	565.0	259.0	3.8462	342200.0	NEAR BAY

存在缺失值的数据前 5 行：

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	\
290	-122.16	37.77	47.0	1256.0	NaN	
341	-122.17	37.75	38.0	992.0	NaN	
538	-122.28	37.78	29.0	5154.0	NaN	
563	-122.24	37.75	45.0	891.0	NaN	
696	-122.10	37.69	41.0	746.0	NaN	

	population	households	median_income	median_house_value	ocean_ proximity
0	290	570.0	218.0	4.3750	NEAR BAY
1	341	732.0	259.0	1.6196	NEAR BAY
2	538	3741.0	1273.0	2.5762	NEAR BAY
3	563	384.0	146.0	4.9489	NEAR BAY
4	696	387.0	161.0	3.9063	NEAR BAY

整理后数据前 5 行，请注意比原始数据少了一列：

	longitude	latitude	housing_median_age	total_rooms	population	\
0	-122.23	37.88	41.0	880.0	322.0	
1	-122.22	37.86	21.0	7099.0	2401.0	
2	-122.24	37.85	52.0	1467.0	496.0	
3	-122.25	37.85	52.0	1274.0	558.0	
4	-122.25	37.85	52.0	1627.0	565.0	

	households	median_income	median_house_value	ocean_ proximity
0	126.0	8.3252	452600.0	3.0
1	1138.0	8.3014	358500.0	3.0
2	177.0	7.2574	352100.0	3.0
3	219.0	5.6431	341300.0	3.0
4	259.0	3.8462	342200.0	3.0

三、案例

1. 案例：房地产区域价格分析

标准化后训练集前 5 行：

	longitude	latitude	housing_median_age	total_rooms	population	\
0	1.272587	-1.372811	0.348490	0.222569	0.768276	
1	0.709162	-0.876696	1.618118	0.340293	-0.098901	
2	-0.447603	-0.460146	-1.952710	-0.342597	-0.449818	
3	1.232698	-1.382172	0.586545	-0.561490	-0.007434	
4	-0.108551	0.532084	1.142008	-0.119565	-0.485877	

	households	median_income	ocean_proximity
0	0.322906	-0.326196	2.005932
1	0.672027	-0.035843	2.005932
2	-0.430461	0.144701	2.005932
3	-0.380587	-1.017864	2.005932
4	-0.314962	-0.171488	-0.112427

第10-13行检查是否存在缺失数据，并在第25-28行进行了中位数填充缺失值；第15-17行通过人工分析，舍去了冗余的数据列；第19-23行将非数值数据转换为数值型数据；第41-46行将数据转换为均值为0、方差为1的标准化数据。这种处理对某些算法效果和执行效率有比较明显的提升

三、案例

1. 案例：房地产区域价格分析

- 其他算法的比较：随机森林回归，人工神经网络回归

```
#随机森林回归，是集成了多棵决策树而成的森林
from sklearn import ensemble
model_rf= ensemble.RandomForestRegressor(n_estimators=20)  # 使用20个决策树
model_rf.fit(Train_X,Train_y)
rf_score=model_rf.score(Test_X,Test_y)*100
print('sklearn随机森林模型得分：%.1f' %rf_score + '%')
rf_pred=model_rf.predict(Test_X)
```

```
sklearn随机森林模型得分：80.6%
```

```
#人工神经网络回归，Sklearn中，又称多层感知机MLP
from sklearn.neural_network import MLPRegressor
model_mlp = MLPRegressor(solver='lbfgs', hidden_layer_sizes=(5, 5),
                         random_state=1)
model_mlp.fit(Train_X,Train_y)
mlp_score=model_mlp.score(Test_X,Test_y)*100
print('sklearn人工神经网络回归模型得分：%.1f' %mlp_score + '%')
```

```
sklearn人工神经网络回归模型得分：69.0%
```

使用随机森林模型来对影响因素（即前9列的数据）和结果（即最后的房价中位数）进行回归拟合。随机森林模型是集成了多棵决策“树”而成的“森林”。这种模型往往能在拟合求解速度和模型效果之间取得较好的平衡。

使用Scikit-Learn中的人工神经网络（在Scikit-Learn中又称多层感知机MLP）来拟合则效果略差。若借助Scikit-Learn调整模型参数，则可以进一步提升效果。

三、案例

1. 案例：房地产区域价格分析

- 人工神经网络的优化：
- 加大神经网络中隐藏层节点的个数可以提升算法的学习效果。若进一步加大'hidden_layer_sizes': [(5,5),(10,10)]的搜索范围，则还可能进一步提升学习效果，但需要更多的计算能力和运行时间。

```
#采用GridSearchCV来进行参数调整实验，找出最佳参数组合
from sklearn.model_selection import GridSearchCV
param_grid = {'solver':['lbfgs','sgd','adam'],
              'hidden_layer_sizes': [(5,5),(10,10)]
             }
#对param_grid中的各参数进行组合，传递进MPL回归器。
#cv=3,3折交叉验证，将数据集随机分为3份，每次将一份作为测试集，其他为训练集
#n_jobs=-1，使用CPU核心数，-1表示所有可用的核
best_mlp =GridSearchCV(MLPRegressor(max_iter=200),param_grid,cv=3,n_jobs=-1)
best_mlp.fit(Train_X,Train_y)
print('当前最佳参数组合：',best_mlp.best_params_)
best_score=best_mlp.score(Test_X,Test_y)*100
print('sklearn人工神经网络上述参数得分：%.1f' %best_score + '%')
#用以上模型对Test_X进行预测
mlp_pred = best_mlp.predict(Test_X)
```

```
当前最佳参数组合： {'hidden_layer_sizes': (10, 10), 'solver': 'lbfgs'}
sklearn人工神经网络上述参数得分：72.0%
```

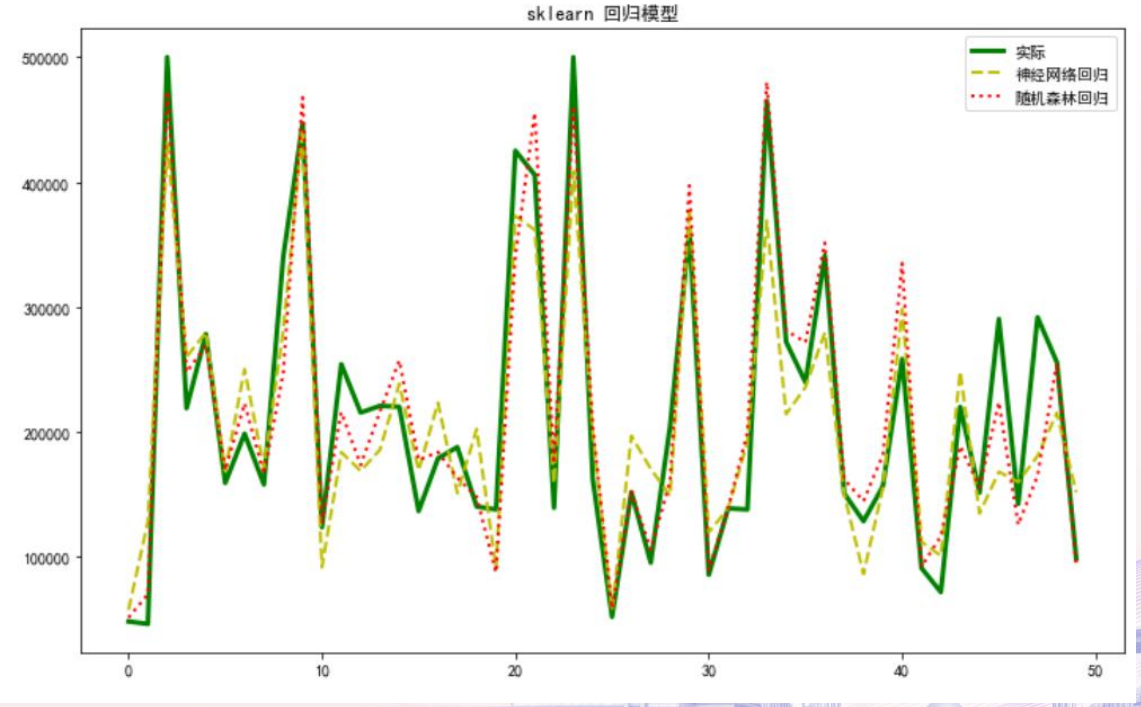
三、案例

1. 案例：房地产区域价格分析

- 绘图比较模型预测房价与真实房价的差异：

该图直观地展示了两种模型对测试集中数据的预测值及算法效果的差异。

```
#绘图比较模型预测房价与真实房价的差异
import matplotlib.pyplot
from matplotlib import r
plt.rcParams['font.sans-s
plt.rcParams['axes.unicod
fig = plt.figure(figsize=
#axes = fig.add_subplot(
#为便于观察，只取部分结果
T1=Test_y[:50]
P1=mlp_pred[:50]
R1=rf_pred[:50]
plt.plot(range(len(T1)),T
plt.plot(range(len(P1)),P
plt.plot(range(len(P1)),P
fig.tight_layout()
plt.legend()
plt.title('sklearn 回归模
plt.savefig('skl_01.png',
plt.show()
```



谢谢观赏 下节课见