特征工程和结果可视化

主讲老师：高彦杰

内容概要

1） Python机器学习库

2） Python AI主流库介绍和典型使用

3） 特征工程

4） 模型加载，模型存储

5） Python特征选择PCA

6） 分析结果可视化

7） Python参数搜索

8） 泰坦尼克求生实战案例

Python机器学习库

Python机器学习

准备工作：

1）Anaconda

包管理

环境管理

解决各种第三方包安装问题

https://www.anaconda.com/download/

2）Pip

Python包管理工具。

```
$ pip -V       #查看pip版本
```

Python机器学习

1）机器学习

Scikit-learn

基于NumPy, SciPy, Matplotlib

开源，涵盖分类，回归和聚类算法

代码和文档完备

2）数据处理

Pandas

基于NumPy和Matplotlib

数据分析和数据可视化

数据结构DataFrame

Python机器学习

3）科学计算

Numpy

数值编程工具

矩阵数据类型、矢量处理

4）可视化

Matplotlib

绘图库

和matlab相似的命令API

Python AI主流库介绍和典型使用

Numpy

- ✓ 数组操作数据提升数据的处理效率
- ✓ 类似于R的向量化操作
- ✓ 可以进行数组和矢量计算

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Numpy

```
01. >>> a= np.array([20,30,40,50])
02. >>> b= np.arange( 4)
03. >>> b
04. array([0, 1, 2, 3])
05. >>> c= a-b
06. >>> c
07. array([20, 29, 38, 47])
08. >>> b**2
09. array([0, 1, 4, 9])
10. >>> 10*np.sin(a)
11. array([ 9.12945251,-9.88031624, 7.4511316, -2.62374854])
12. >>> a<35
13. array([True, True, False, False], dtype=bool)
```

Pandas

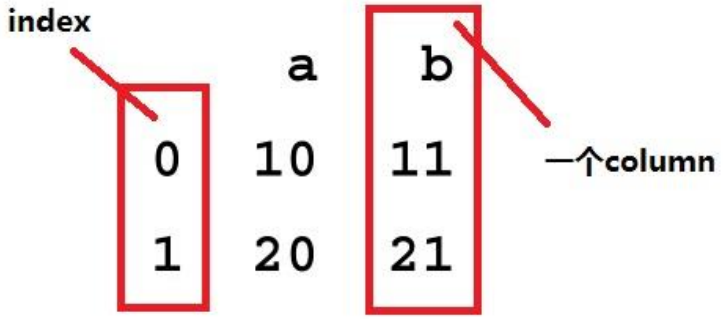
- ✓ **带有标签的列和索引**
- ✓ **csv 类型的文件中导入数据**
- ✓ **对数据进行复杂的转换和过滤等操作**
- ✓ **series 和 dataframe**

series 是一种一维的数据类型，其中的每个元素都有各自的标签。

dataframe 是一个二维的、表格型的数据结构。可以把它当作一个 series 的字典。

Pandas

```
import numpy as np
import pandas as pd
from pandas import Series, DataFrame
data = DataFrame(np.arange(16).reshape(4,4),index=list('abcd'),columns=list('wxyz'))
data['w']  #选择表格中的'w'列，使用类字典属性,返回的是Series类型
data.w    #选择表格中的'w'列，使用点属性,返回的是Series类型
data[['w']]  #选择表格中的'w'列，返回的是DataFrame类型
data[['w','z']]  #选择表格中的'w'、'z'列
```



scikit-learn

- ✓ 机器学习的Python库
- ✓ 分类、聚类以及回归分析方法
- ✓ 强大的功能、优异的拓展性以及易用性
- ✓ 著名的一个开源项目之一

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, … — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, … — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, … — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction.

— Examples

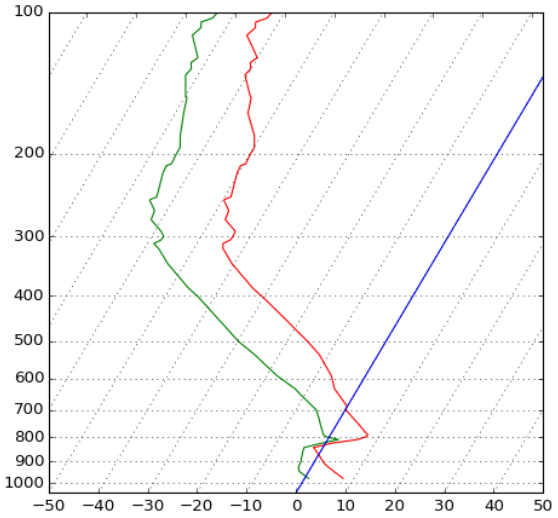
scikit-learn

```
import numpy as np
import random
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

def linear_regression_demo(n = 25):
    #模拟一个 y = k * x 的数据集,并做一个线性回归,求解k,并做预测
    #首先随机构造一个近似于y = k * x + b 的数据集
    k = random.random()
    b = random.random() * 1
    x = np.linspace(0,n,n)
    y = [ item * k +(random.random() - 0.5) * k * 5 + b for item in x]
    true_y = [ item * k for item in x]
    #进行一元线性回归
    model = LinearRegression()
    model.fit(np.reshape(x,[len(x),1]), np.reshape(y,[len(y),1]))
    yy = model.predict(np.reshape(x,[len(x),1]))
```

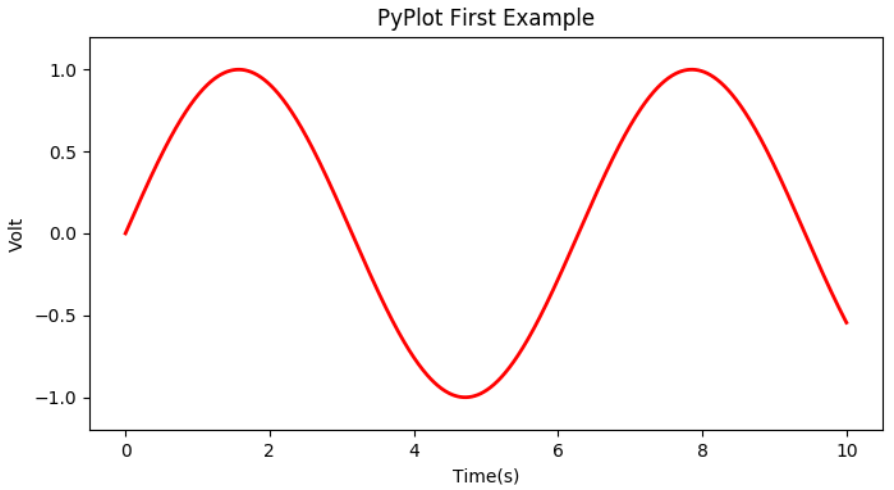
Matplotlib

- ✓ 2D绘图库
- ✓ 各种格式
- ✓ 跨平台的交互式环境
- ✓ 生成出版质量级别的图形



matploitlib

```python
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 10, 1000)
y = np.sin(x)
plt.figure(figsize=(8,4))
plt.plot(x,y,label="$sin(x)$",color="red",linewidth=2)
plt.xlabel("Time(s)")
plt.ylabel("Volt")
plt.title("PyPlot First Example")
plt.ylim(-1.2,1.2)
plt.show()
"""
通过一系列函数设置当前Axes对象的各个属性：
xlabel、ylabel：分别设置X、Y轴的标题文字。
title：设置子图的标题。
xlim、ylim：分别设置X、Y轴的显示范围。"""
```



特征工程

特征工程

业界广为流传的一句话：

”数据和特征决定了机器学习的上限，而模型和算法只是逼近这个上限而已。”

其本质是一项工程活动，目的是最大限度地从原始数据中提取特征以供算法和模型使用。

特征工程

特征如何使用？ - 业务理解，可用性评估

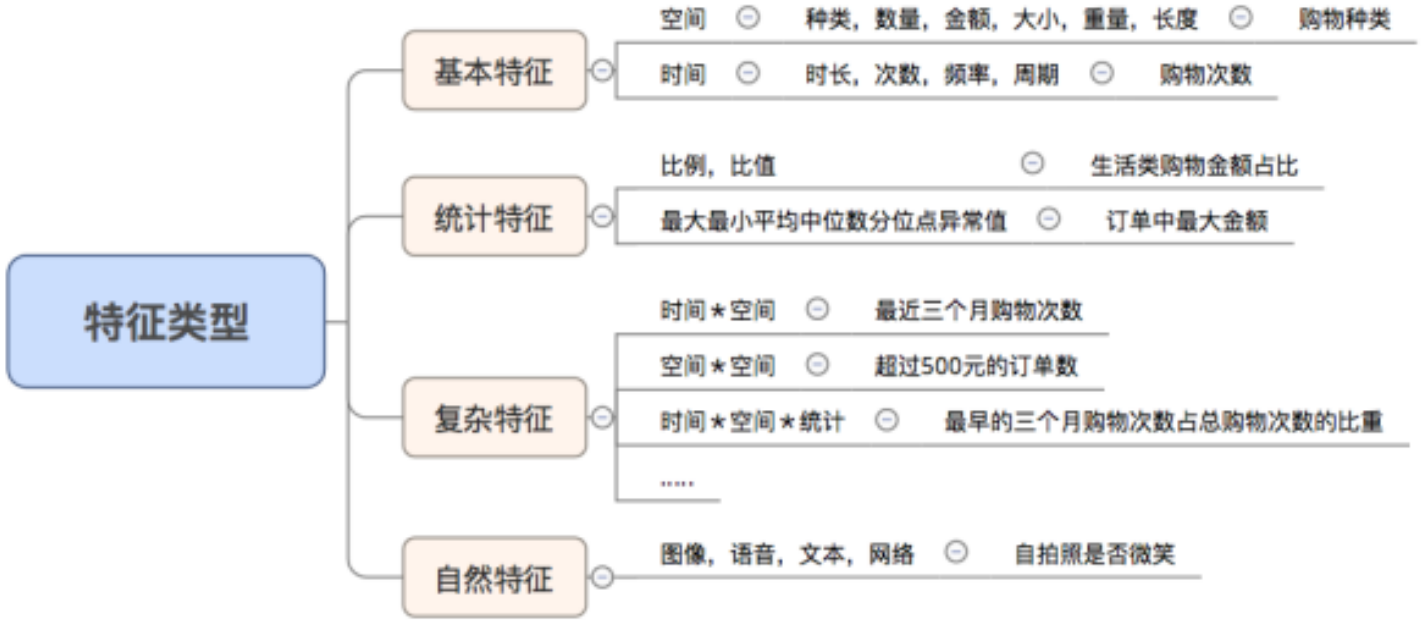
特征如何获取？ - 获取与存储

特征如何处理？

- 清洗，标准化，特征选择，特征扩展

更新特征？

特征类型


特征类型
基本特征
空间
种类，数量，金额，大小，重量，长度
购物种类
时间
时长，次数，频率，周期
购物次数
统计特征
比例，比值
生活类购物金额占比
最大最小平均中位数分位点异常值
订单中最大金额
复杂特征
时间★空间
最近三个月购物次数
空间★空间
超过500元的订单数
时间★空间★统计
最早的三个月购物次数占总购物次数的比重
.....
自然特征
图像，语音，文本，网络
自拍照是否微笑

scikit-learn 特征工程

类	功能	说明
StandardScaler	数据标准化	标准化，基于特征矩阵的列，将特征值转换至服从标准正态分布
MinMaxScaler	数据标准化	区间缩放，基于最大最小值，将特征值转换到[0, 1]区间上
Normalizer	归一化	基于特征矩阵的行，将样本向量转换为“单位向量”
Binarizer	二值化	基于给定阈值，将定量特征按阈值划分
OneHotEncoder	哑编码	将定性数据编码为定量数据
Imputer	缺失值计算	计算缺失值，缺失值可填充为均值等
PolynomialFeatures	多项式数据转换	多项式数据转换
FunctionTransformer	自定义单元数据转换	使用单变元的函数来转换数据

Python特征选择PCA

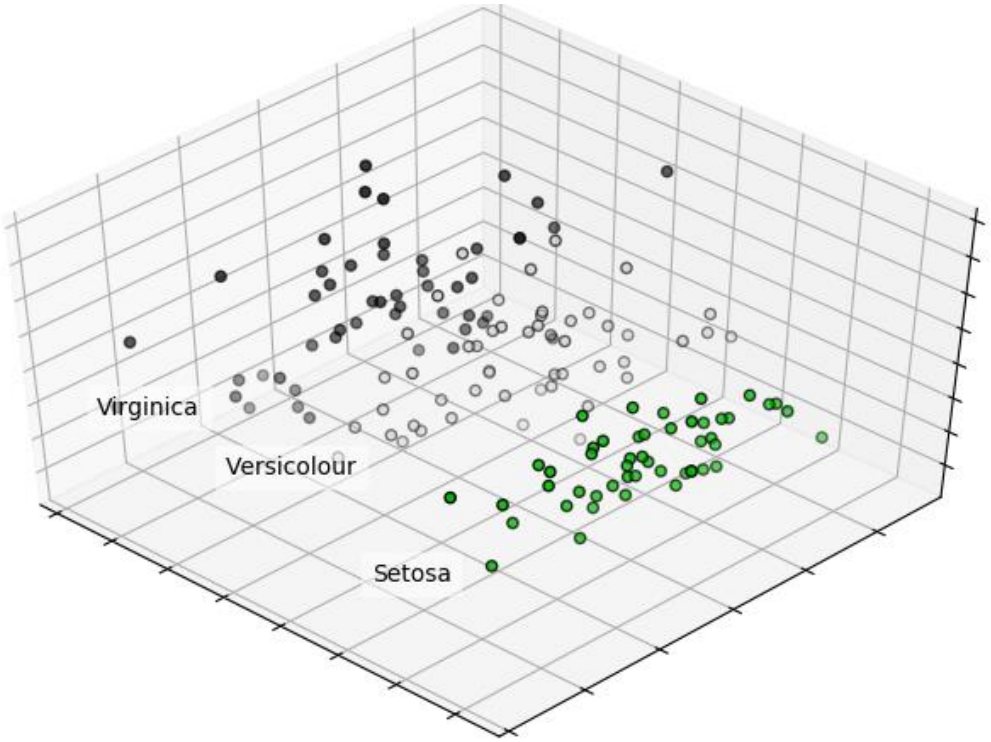
降维与PCA

降维致力于解决：

- ✓ **1. 缓解维度灾难；**
- ✓ **2. 压缩数据时让信息损失最小化；**
- ✓ **3. 高维数据可视化；**

降维与PCA

```
from sklearn import decomposition
from sklearn import datasets
import numpy as np
np.random.seed(5)
iris = datasets.load_iris()
X = iris.data
y = iris.target
pca = decomposition.PCA(n_components=3)
pca.fit(X)
X = pca.transform(X)
print(X)
```



模型加载，模型存储

模型存储

#scikit-learn已经有了模型持久化的操作，导入joblib即可

```
from sklearn.externals import joblib
#模型保存
from sklearn import svm
X = [[0, 0], [1, 1]]
y = [0, 1]
clf = svm.SVC()
clf.fit(X, y)
joblib.dump(clf, "train_model.m")
```

模型加载

```
# 通过joblib的dump可以将模型保存到本地，clf是训练的分类器
# 模型从本地调回
clf = joblib.load("train_model.m")
# 通过joblib的load方法，加载保存的模型。
# 然后就可以在测试集上测试了
clf.predict(X)
```

分析结果可视化

可视化

可视化Python练习

Python参数搜索

参数搜索

在我们日常的进行超参数优化工作时，可以手动去试，也可以使用随机搜索、批量随机搜索和网格搜索等方法调到好的参数，关于网格搜索，sklearn中GridSearchCV用于系统地遍历多种参数组合，通过交叉验证确定最佳效果参数。

参数搜索

```
import numpy as np
from sklearn import datasets
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
# load the diabetes datasets
dataset = datasets.load_diabetes()
# prepare a range of alpha values to test
alphas = np.array([1,0.1,0.01,0.001,0.0001,0])
# create and fit a ridge regression model, testing each alpha
model = Ridge()
grid = GridSearchCV(estimator=model, param_grid=dict(alpha=alphas))
grid.fit(dataset.data, dataset.target)
print(grid)
# summarize the results of the grid search
print(grid.best_score_)
print(grid.best_estimator_.alpha)
```

```
GridSearchCV(cv=None, error_score='raise',
       estimator=Ridge(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=None,
   normalize=False, random_state=None, solver='auto', tol=0.001),
       fit_params={}, iid=True, n_jobs=1,
       param_grid={'alpha': array([  1.00000e+00,   1.00000e-01,   1.00000e-02,   1.00000e-03,
         1.00000e-04,   0.00000e+00])},
       pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
       scoring=None, verbose=0)
0.488790204461
0.001
```

实战案例