

# 特征工程和结果可视化

**主讲老师：高彦杰**

# 课程说明

围绕着“如何去加工出来高质量的、有助于预测的特征”而做的一个工程性的工作

在完成一些预测问题的时候，这些重要的影响因素，这些影响因素，就像原来面对的数据库表里的属性，一个一个的属性就是一个一个的特征  
列——对应特征  
行——对应样本

## ✓ 内容简介：

✓ 特征工程：空值处理，独热编码，特征选择，特征扩展，标准化，归一化等

每个技术点后  
面有公式在

✓ 模型保存，降维

三大类，5、6  
种方法

在新的数据场景里，将新的内容  
吸纳进自己的知识体系里

✓ 自动化机器学习：超参数调优

## ✓ 企业级应用领域：

✓ AI竞赛

✓ 工作中机器学习业务场景

某些场景下，可以带来实质性提升

## 课程推荐环境

vscode最新版

Anaconda3-4.2

python3.5

matplotlib == 1.5.3

numpy == 1.13.3

pandas == 0.18.1

scikit\_learn == 0.19.1

# 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(★★★) 以知识体系来带动它
- 4) 模型加载, 模型存储(★★)
- 5) 分析结果可视化(★★)
- 6) 自动化机器学习 超参数调优(★★)
- 7) 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

易上手  
工作中也经常  
用

## 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

处理各种数据, 如特定机型  
等, 也有可能自己写脚本, 但  
最终目的是将数据处理成向量  
和矩阵格式

## 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

追加列名(行名)——列是特  
征  
相当于把numpy变成数据库  
用它时是想知道数据语义了

# Python机器学习

## 3) 科学计算

底层实现尽量  
依赖库

Numpy

处理矩阵数据

数值编程工具

ML、AI 应用于系统领域

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

AI 中用可视化是期望一种交互式模式

- 1、看数据分布
- 2、做特征选择的时候，（土）的方式画出xy散点图
- 3、训练时，看准确度、err等（称之为学习曲线learning curve）
- 4、可视化出来报告，可视化展示分析结果



机器学习阶段，推荐接触纯机器学习数据，避免图像\语音等特征

任何数据都要to Vector

转化为向量后，才能给后面的算法去处理

参数搜索

HPO超参搜索  
现在属于auto ML范畴，有两个分支：  
HPO和NAS

机器学习结构稳定，没有模型结构的问题，没有NAS可做

原始数据: 结构化数据, 文本, 图像...

数据文件

为模型准备高质量的特征

特征工程->向量化数据

可视化

启发思路，  
诊断问题

若团队比较大，pipeline比较重要；  
各个阶段有不同的人  
留个备份，防止重复处理，每个阶段分开（重点）

模型训练

模型保存与加载

模型预测：使用模型

模型文件

构建应用时的数据后处理

对应字典展示  
出来

# 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

有一些numpy的gpu优化库



- 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

numpy的array、arange可以用来fake data  
特定的造出符合特定分布的数据  
做一些测试的时候，比如深度学习精度到了提高速度时可以用

```
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)
```

- 1、矩阵运算
- 2、fake data
- 3、提供数学函数
- 4、很多机器学习库，接收数组格式，即使是pandas也要转成numpy

# 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

表模式—列名  
行索引不常用  
csv文件 推荐  
pandas去读

CSV读进来之后就是一个dataframe数据结构

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
```

index	a	b
0	10	11
1	20	21

一个column

## 框架类型

常见数据没有那么大规模，所以mlib使用较少  
当成sklearn的分布式版，应对大规模数据，但是大规模数据并不是主流

- ✓ 通用机器学习：Scikit-learn, Spark MLlib
- ✓ 特定系列算法：XGBoost
- ✓ 深度学习：TensorFlow, Keras

竞赛神器：决策树 gbdk 高效实现

回归也能拿这些算法编写出来；  
只是sklearn中比较成熟的回归算法

迁移pytorch平滑

如果对某个算法有极致追求的时候，推荐使用特定系列的算法

sklearn理解为推导硬编码，所以只能实现经典机器学习算法  
深度学习框架可以随便写算法模型，可以自动求导，把人工推导过程推导出来

但是深度学习没哟取代机器学习  
像sklearn有非常成熟的机器学习框架

什么样的算法决定使用什么样的框架，你有什么样的问题规模，决定使用什么样的框架；  
以及你是否对某一个算法有非常深入的了解，觉得这个算法是非常适合这个数据集的，再去选择特定算法的框架

- ✓ 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

文档友好，示例可直接运行

### 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]))
```

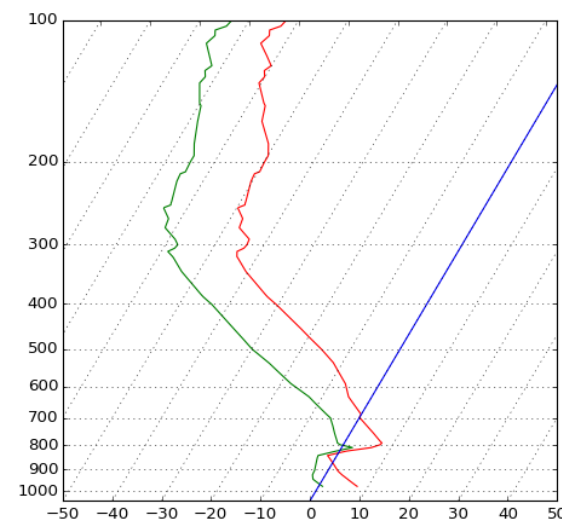
行是sample样本  
列是feature特征  
因为此处只有一个x, 即只有一个特征  
所以此处追加一个列

机器学习：轻  
量，快

# 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

matplotlib



# matplotlib

```
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()
```

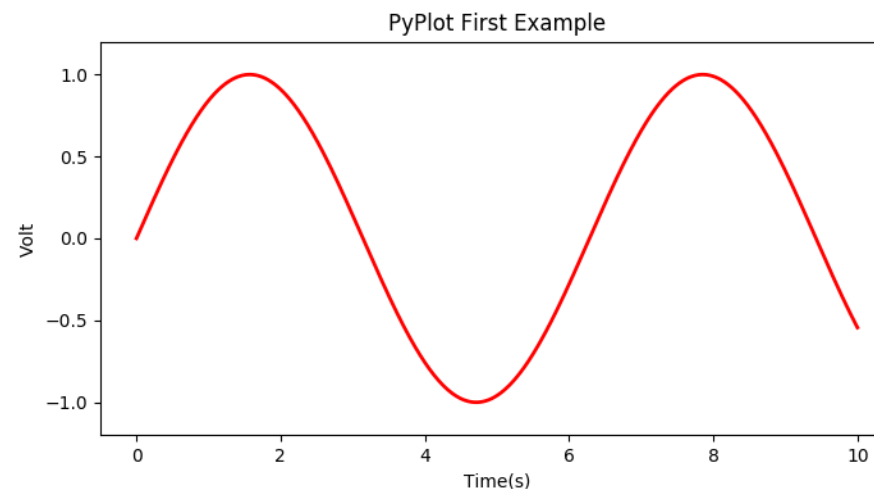
"""

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。 """



## 对应练习及文件

```

└─ a0_python
   └─ basic_syntax
   └─ virtual_env
└─ a1_numpy_pandas
   └─ numpy
      ├── numpy_array.py
      ├── Numpy.ipynb
      └── nupmy_func.py
   └─ pandas
      ├── read_csv_save_pickle
      ├── 1_pandas_intro.py
      ├── 3_set_value.py
      ├── 4_nan.py
      ├── 5_concat.py
      ├── 6_merge.py
      ├── 7_plot.py
      └── 8_pandas_col_row.py
```

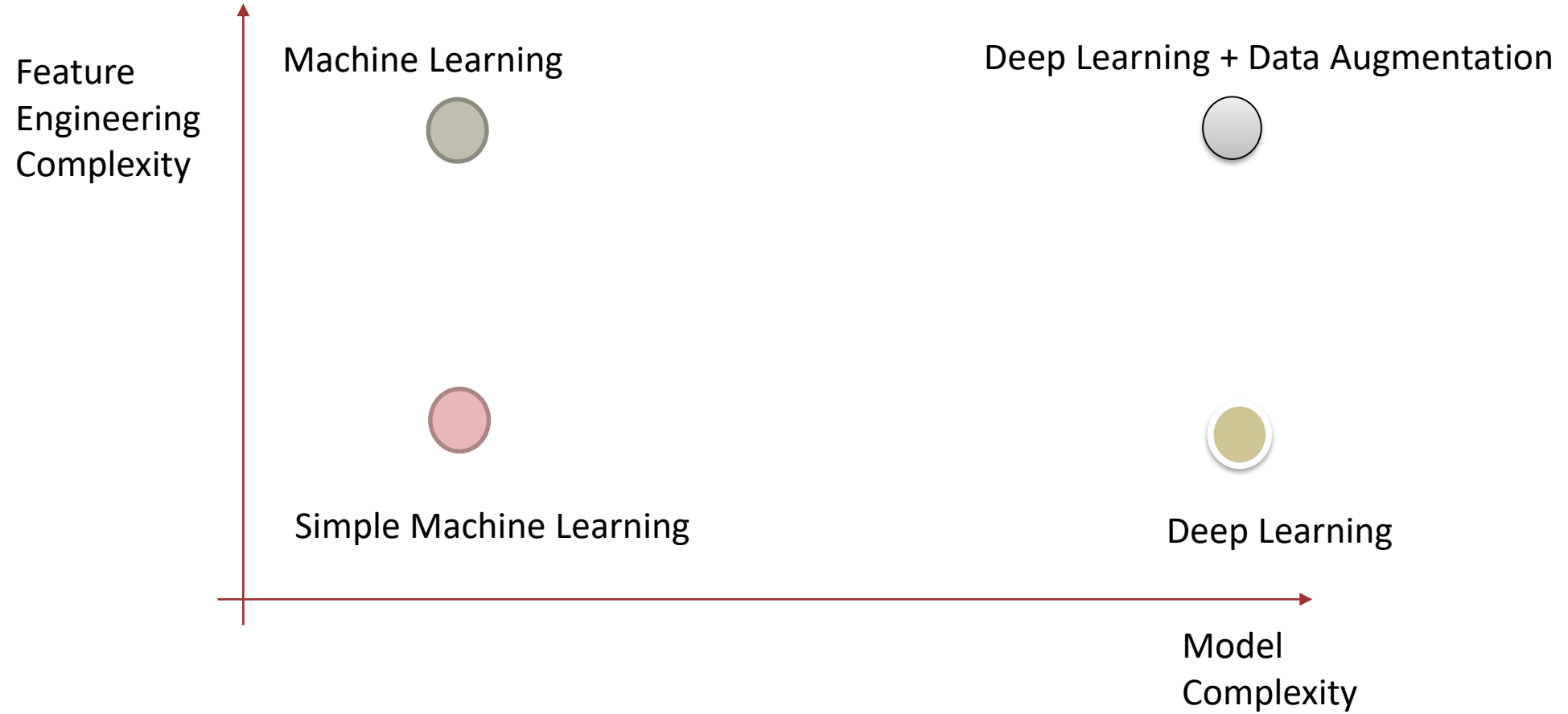
# 特征工程

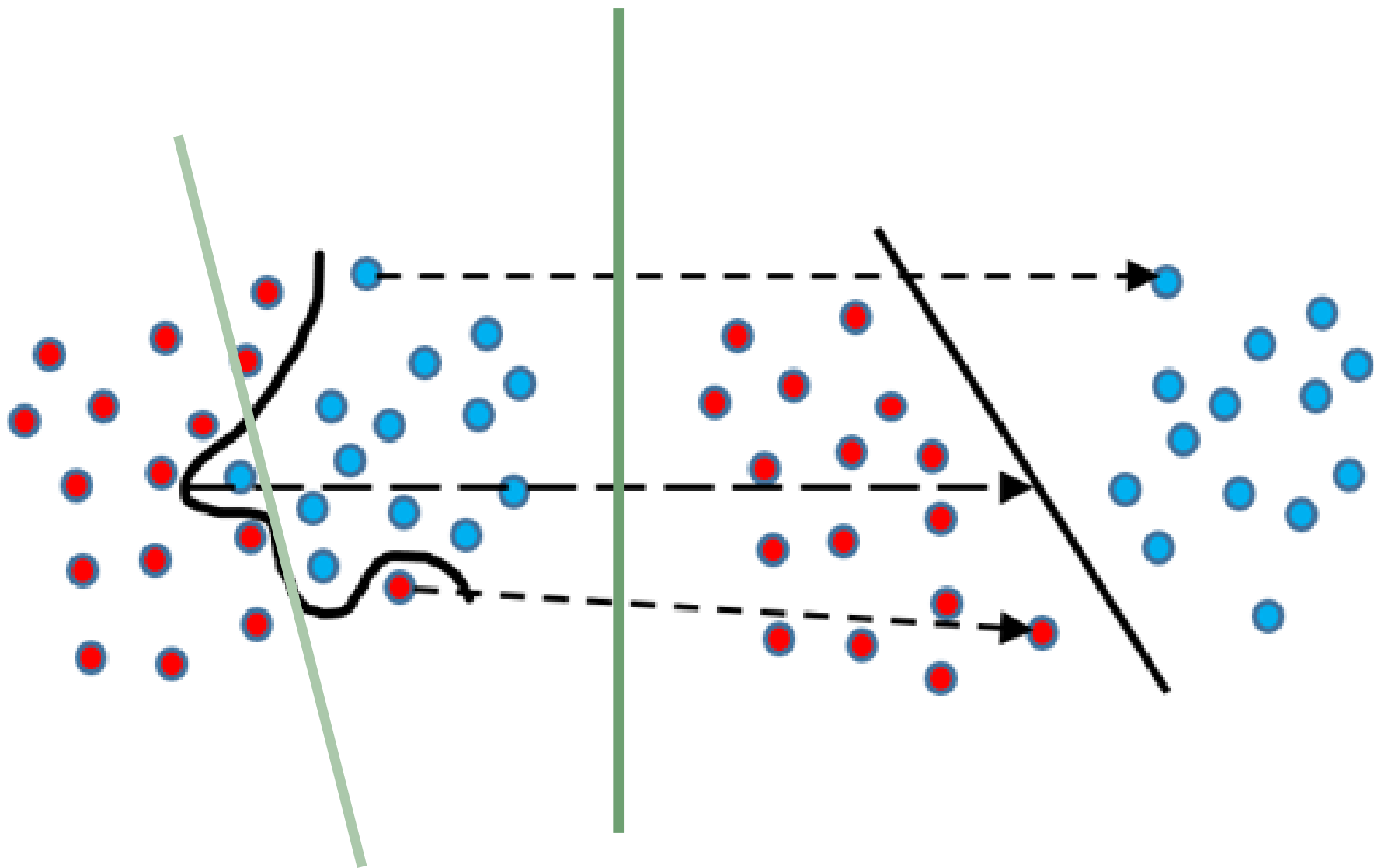
## 特征工程

**“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。”**

**“复杂特征工程配合简单模型(ML) <-> 简单特征工程配合复杂模型(DL)。”**








## 特征工程-解决的一些问题

- 加工数据为模型可处理的格式---向量化和规范化
- 能让数据变得使模型更容易拟合：简化数据 / 简化机器学习问题
- 某些方法能加速一些优化算法的收敛性（含有 $\theta^*X$ ）

## 模型输入正确

ValueError: Input contains NaN, infinity or a value too large for dtype('float32')



I got ValueError when predicting test data using a RandomForest model.

asked 2 years, 8 months


17

```
clf = RandomForestClassifier(n_estimators=10, max_depth=6, n_jobs=1, verbose=0)
clf.fit(X_fit, y_fit)

df_test.fillna(df_test.mean())
X_test = df_test.values
y_pred = clf.predict(X_test)
```

viewed 62,475 times

active 2 months ago



★

6

Linked

1 [ValueError: Input con](#)

# 映射

```
>>> X = np.array([[1, 2], [2, 5], [4, 6]])
>>> X
array([[1, 2],
       [2, 5],
       [4, 6]])
>>> Y = np.array([[1], [0], [1]])
>>> Y
array([[1],
       [0],
       [1]])
```

体重  
x2

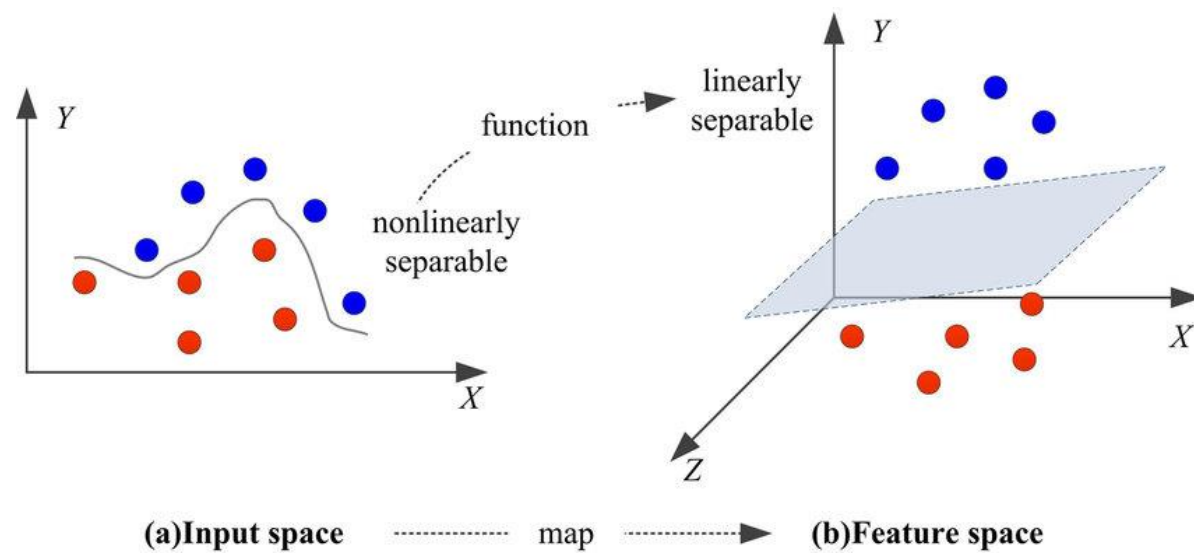
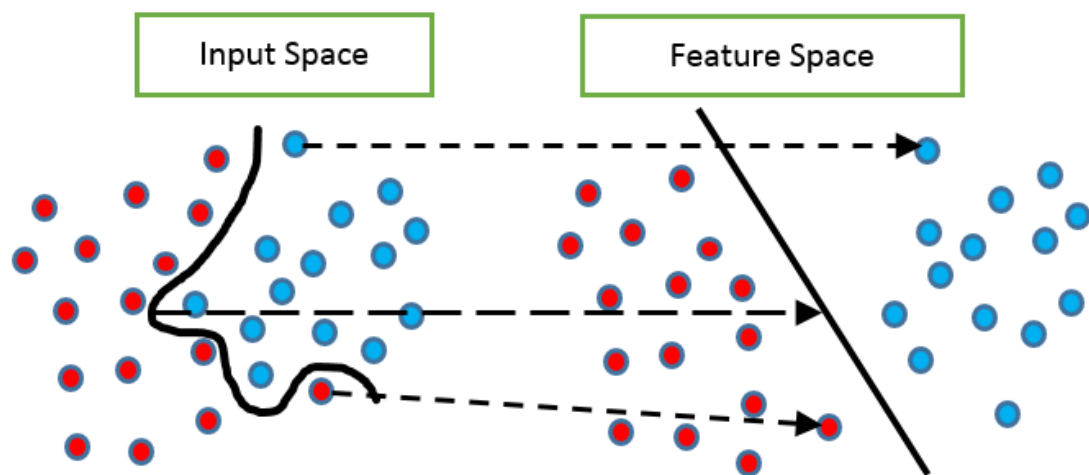
A类人群

B类人群

B类人群

x1  
身高

# 特征映射



# Feature Scaling

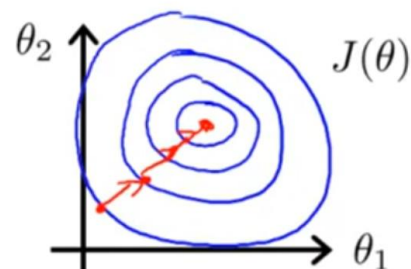
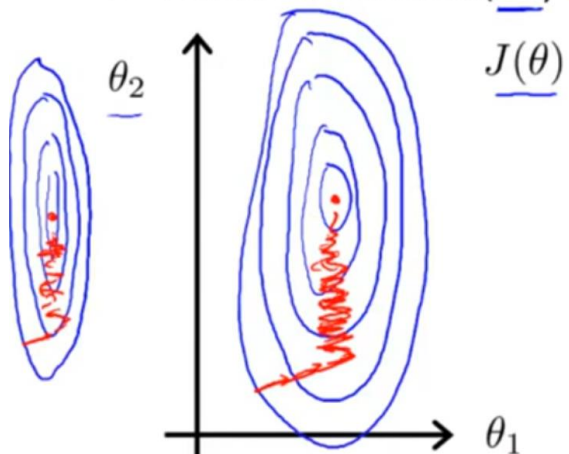
Idea: Make sure features are on a similar scale.

E.g.  $x_1 = \text{size (0-2000 feet}^2\text{)}$  ←

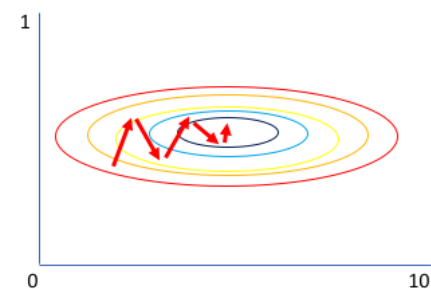
$x_2 = \text{number of bedrooms (1-5)}$  ←

$$\rightarrow x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

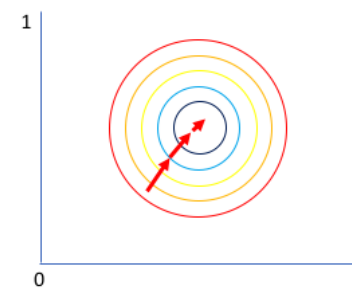
$$\rightarrow x_2 = \frac{\text{number of bedrooms}}{5}$$



Why normalize?



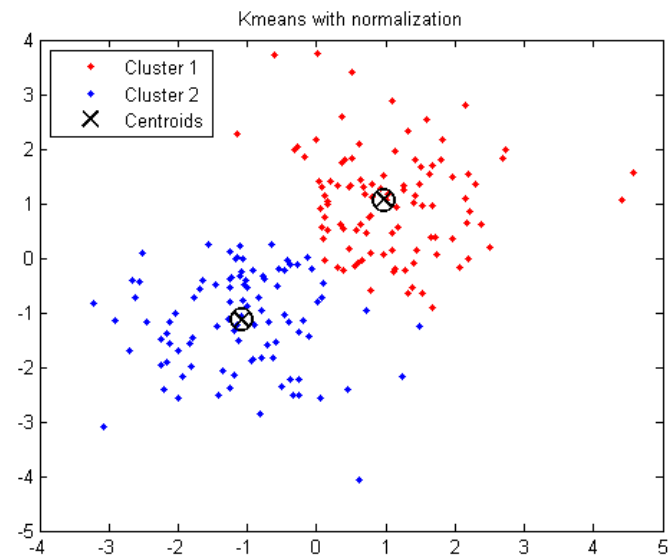
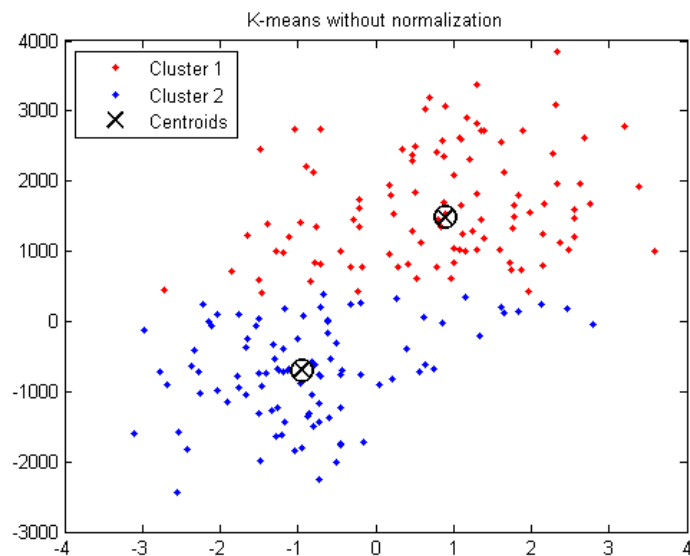
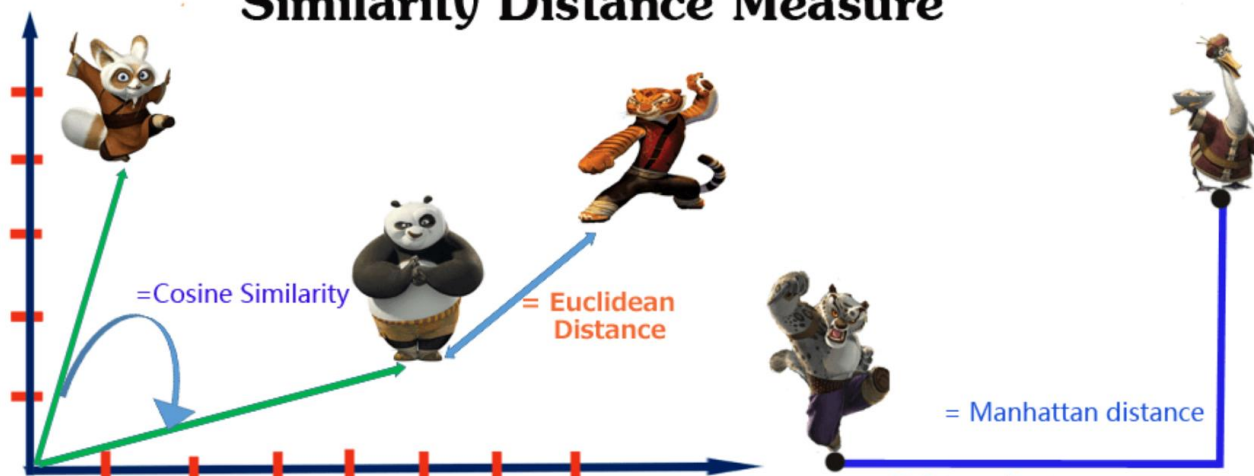
Gradient of larger parameter dominates the update



Both parameters can be updated in equal proportions

# 向量相似度

## Similarity Distance Measure



# 上午课程重点与难点

✓ 重点:

✓ Python机器学习库与数据预处理

✓ 难点:

✓ 理解特征工程目标

# 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？

- 探索，清洗，标准化，特征选择，特征扩

展

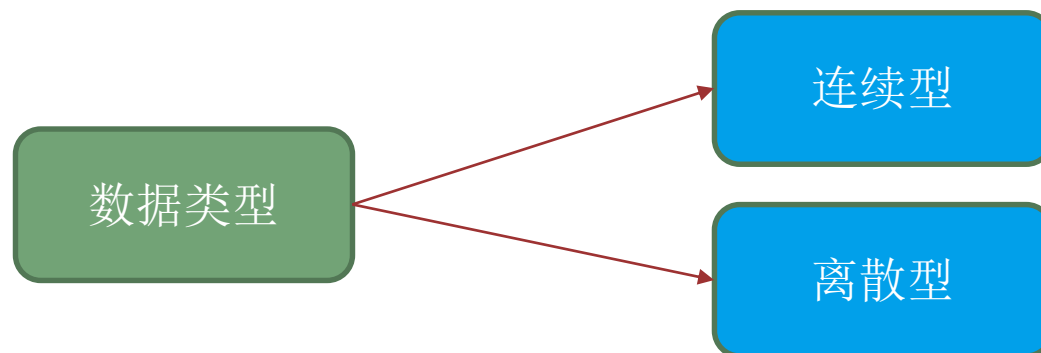
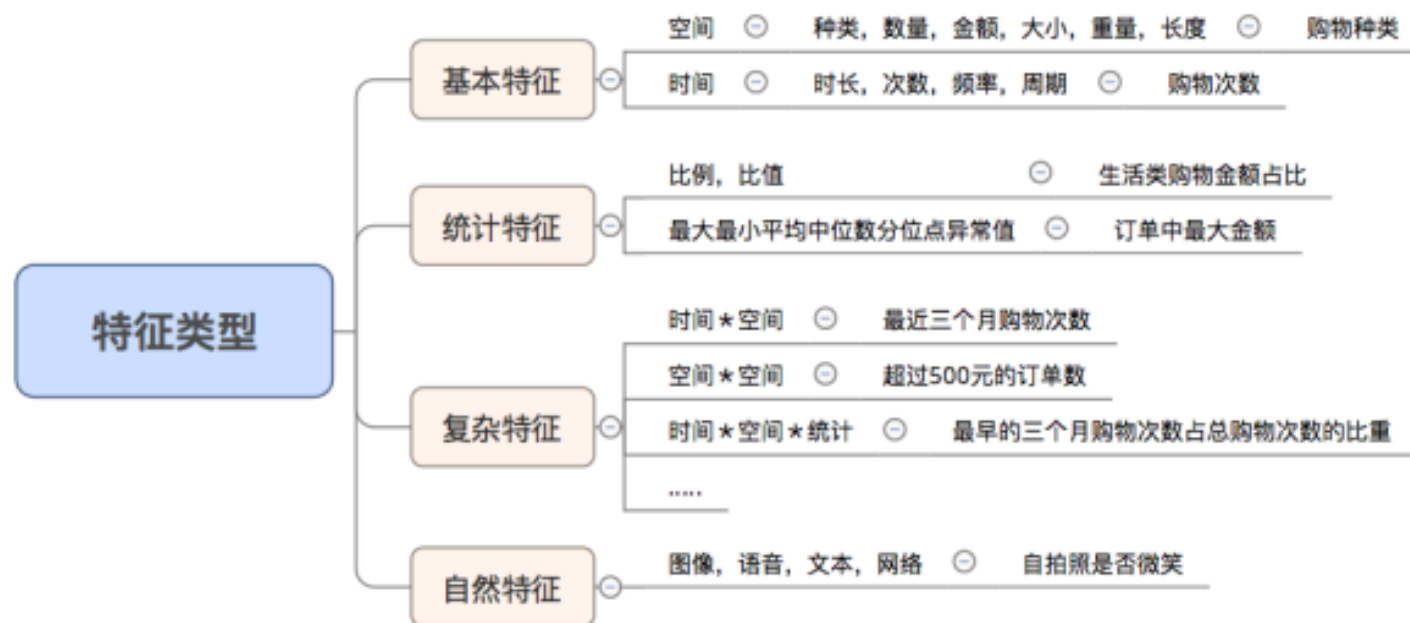
更新特征？



# 特征工程思考流程

- 探索：
  - 特征状况：类型，空值，分布（特征选择），异常点，量纲
  - 标签状况：类型，分布
- 清洗 / 向量化
  - 类别特征编码转为向量
  - 空值处理
- 标准化
  - 异常点
  - 量纲是否需要统一
  - 是否需要归一化
  - 分桶离散
  - 标签均衡
- 特征选择 (过拟合后看看)
  - Filter
  - Wrapper
  - Embed
- 特征扩展 (欠拟合后看看)
  - 业务总结
  - 组合已有特征（组合，聚合）

# 特征类型及业务特征扩展实例



## scikit-learn 特征工程API

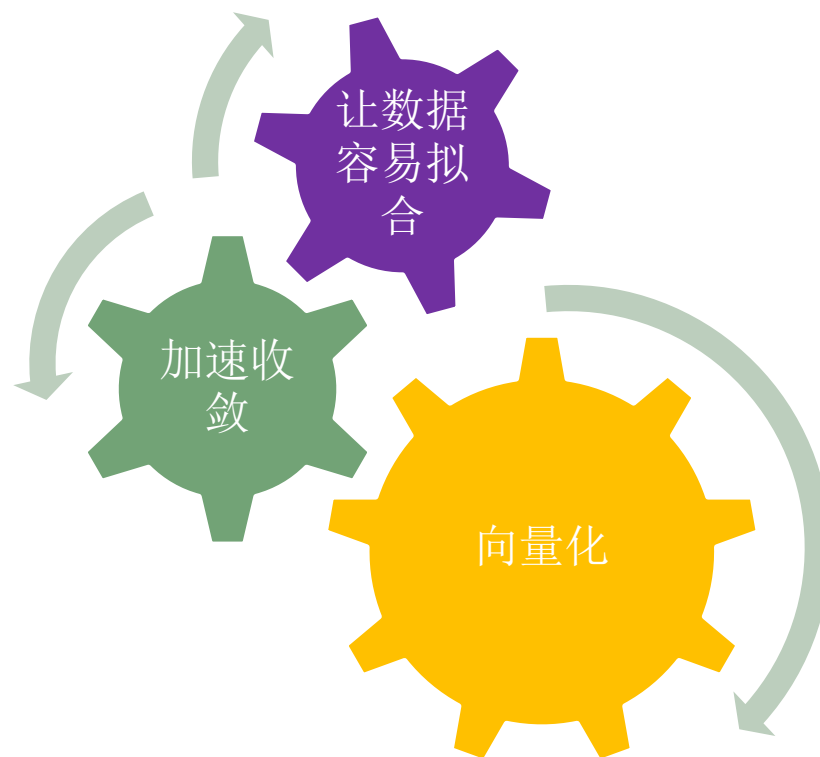
类	功能	说明
StandardScaler	数据标准化	标准化，基于特征矩阵的列，将特征值转换至服从标准正态分布
MinMaxScaler	数据标准化	区间缩放，基于最大最小值，将特征值转换到[0, 1]区间上
Normalizer	归一化	基于特征矩阵的行，将样本向量转换为“单位向量”
Binarizer	二值化	基于给定阈值，将定量特征按阈值划分
OneHotEncoder	哑编码	将定性数据编码为定量数据
Imputer	缺失值计算	计算缺失值，缺失值可填充为均值等
PolynomialFeatures	多项式数据转换	多项式数据转换
FunctionTransformer	自定义单元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

## 特征工程 方法作用

1 让算法能正常运行的，输入向量化与规范  
null和onehot encoder等

2 加速算法收敛的  
标准化相关的

3 让数据更加可分(解决欠拟合和过拟合现象)，问题更加简化  
特征扩展和特征选择



# 人工智能中的数据处理

深度学习阶段内容

原始数据类型

结构化数据

图像

文本

语音

其他数据：  
卫星遥感  
/ 基因 等

向量化

特征编码

空值

离散化

图像向  
量化

文本向  
量化

语音向  
量化

数据向  
量化

规范化

标准化量纲

异常点处理

归一化

BatchNormal

组合与  
筛选

特征扩展

特征选择

降维

模型内组合特征

应用向量

模型输入

向量相似度应用

# Everything to vector



Area, Value, Room, Living, School, Year, Floor

# 企业级应用

✓ 企业级应用领域：

✓ 各个机器学习业务场景

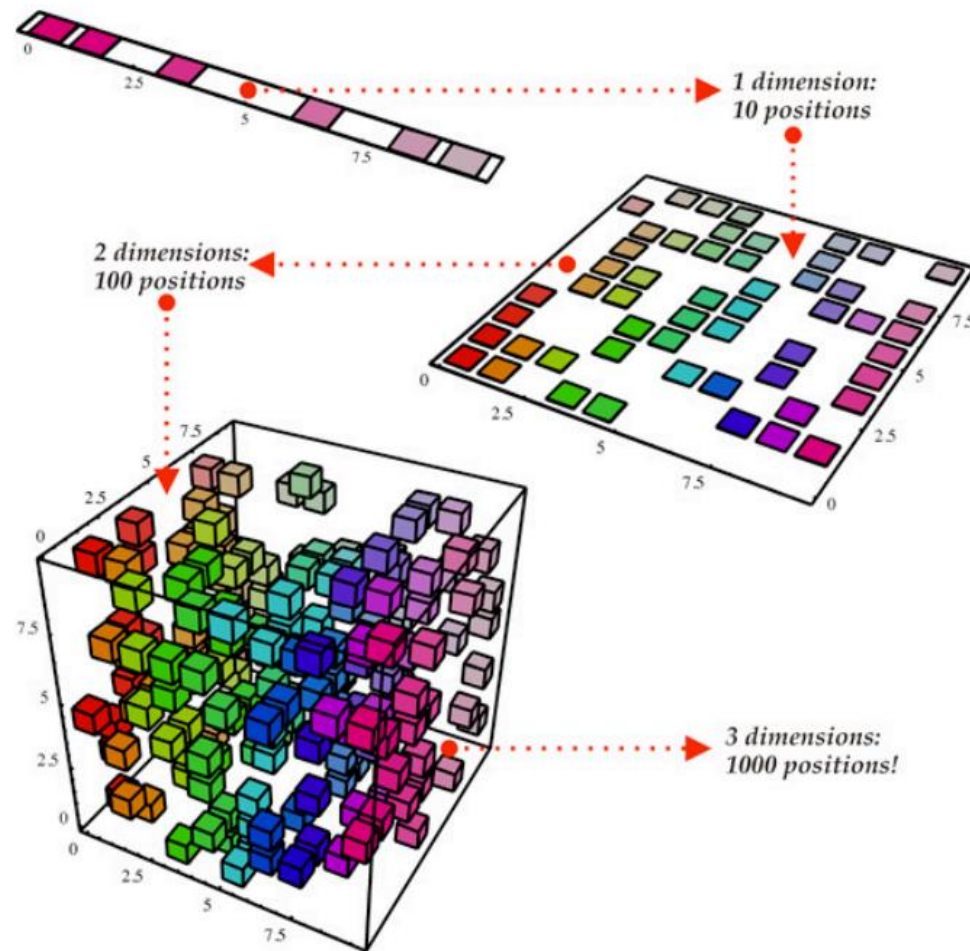
# 降维



# 降维与PCA

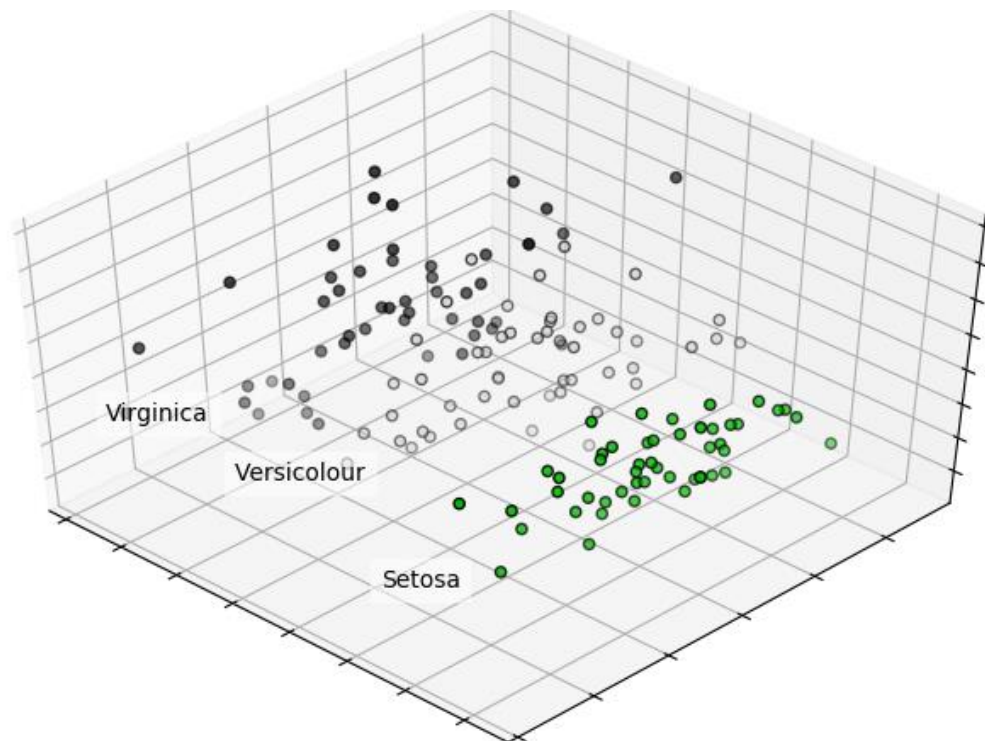
降维致力于解决：

- ✓ 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)
```



# 企业级应用

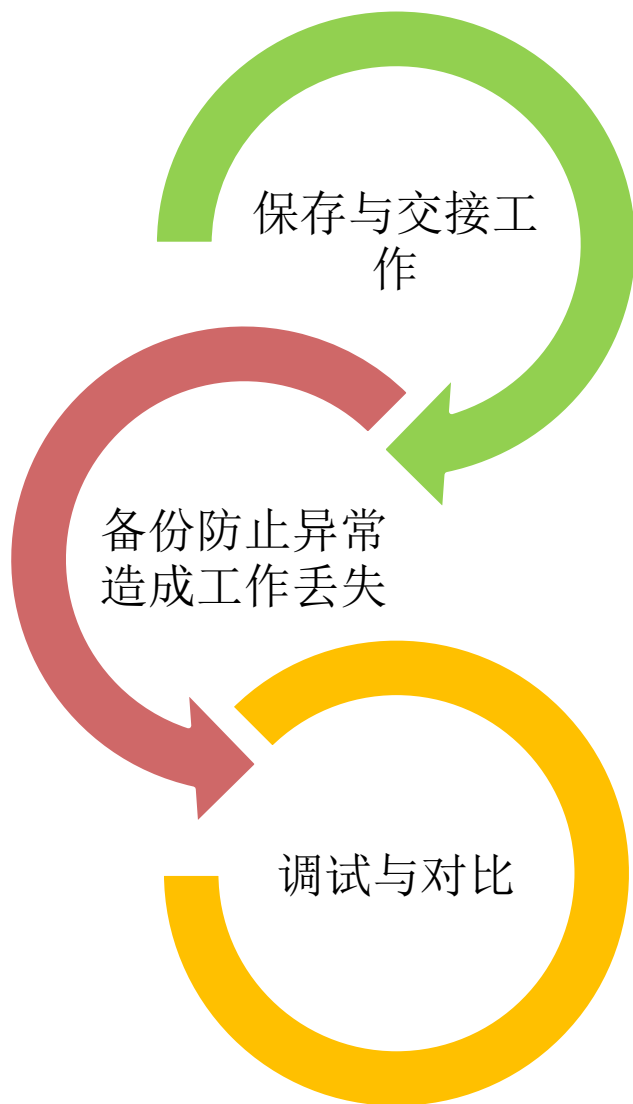
- ✓ 企业级应用领域：
  - ✓ 高维数据可视化
  - ✓ 图像特征压缩

## 对应练习及文件

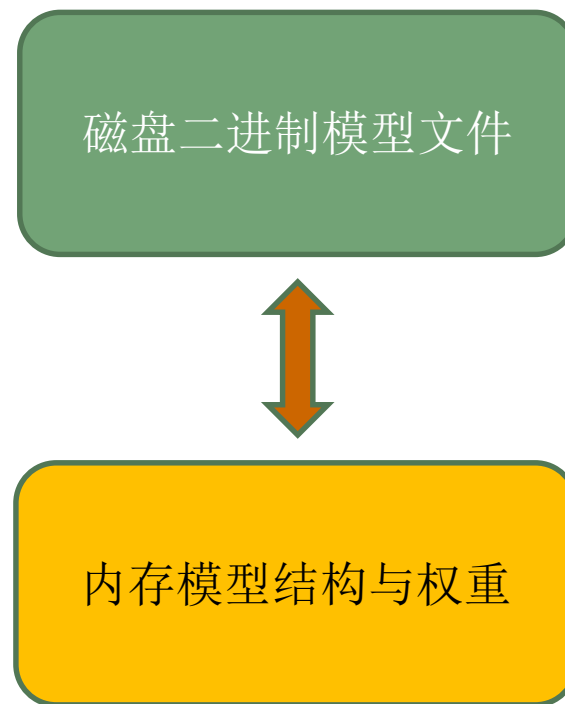
```
✓ a2_feature_engineering
  > 0_eda
  > 1_encoder
  > 2_normalize
  > 3_feature_extend
  > 4_feature_selection
  > 5_reduce_dimension
  > 6_continue_to_bin
  > 7_unbalance
  > doc
```

# 模型加载，模型存储

# 模型存储与加载



模型被保存的文件 = 权重 $w, b$  + meta (函数形式)



## 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)
```



## 对应练习及文件

▼ a3\_model\_load

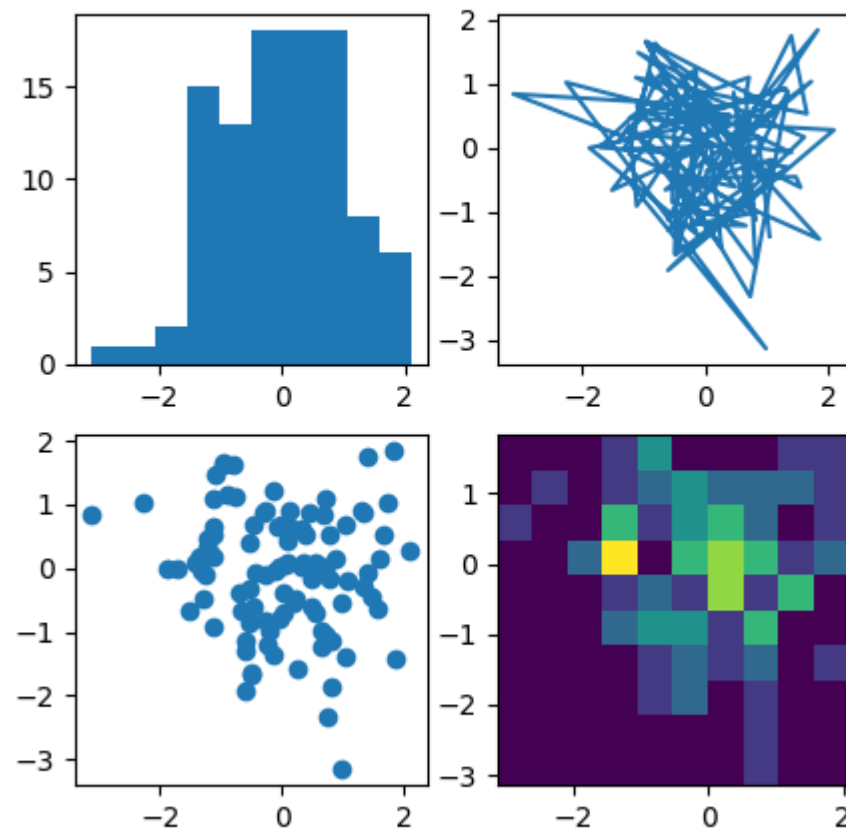
🔗 cross\_validation.py

🔗 model\_save\_load.py

## 分析结果可视化

# 可视化

## 可视化Python练习



## 对应练习及文件

▼ a4\_visualize

🔗 bar.py

🔗 figure.py

🔗 hotpot.py

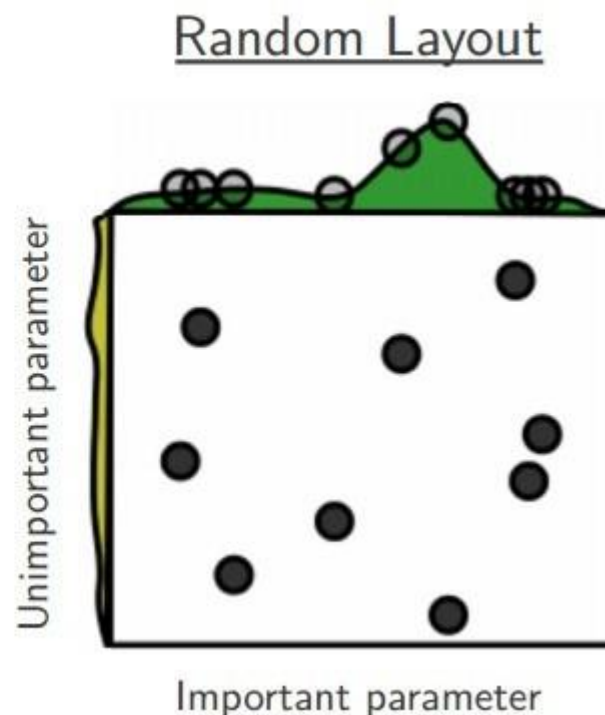
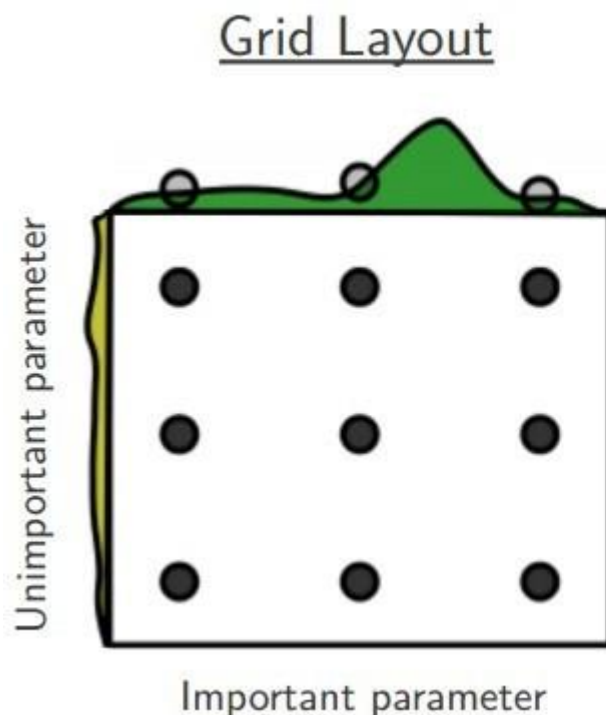
🔗 scatter.py

🔗 simple\_plot.py

# 自动化机器学习 超参数调优

# 参数搜索 (hyperparameter optimization - HPO)

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

The "grid search" process covered in the video is well-known: You define a **set of parameter values** that you want to try with a given model, and then you use cross-validation to evaluate **every possible combination** of those values in order to choose between them.

In a **random search process**, you search only a random subset of the provided parameter values. This allows you to explicitly control the number of different parameter combinations that are attempted, which you can alter depending on the computational time you have available.

## 参数搜索

```
2 import numpy as np
3 from sklearn import datasets
4 from sklearn.linear_model import Ridge
5 from sklearn.model_selection import GridSearchCV
6 # load the diabetes datasets
7 dataset = datasets.load_diabetes()
8 # prepare a range of alpha values to test
9 alphas = np.array([1,0.1,0.01,0.001,0.0001,0])
10 # create and fit a ridge regression model, testing each alpha
11 model = Ridge()
12 grid = GridSearchCV(estimator=model, param_grid=dict(alpha=alphas))
13 grid.fit(dataset.data, dataset.target)
14 print(grid)
15 # summarize the results of the grid search
16 print(grid.best_score_)
17 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
```



## 对应练习及文件

▼ a5\_param\_search

🔗 grid\_search\_random\_search.py

🔗 knn\_grid\_search.py

## 实战案例

## 对应练习及文件

- ▼ a6\_titanic
  - > data
  - > data\_process
  - > error\_study
  - > optimization
  - 📄 feature\_statistic.py
  - 📄 model\_train.py
  - 📄 util.py
  - 📄 数据集说明.md

# 本节课程重点与难点（适用复习）

## ✓ 重点：

- ✓ Python机器学习库与数据预处理
- ✓ 特征工程： 标准化，特征选择，特征编码
- ✓ 超参数优化

## ✓ 难点：

- ✓ 特征清洗与标准化
- ✓ 特征选择
- ✓ 特征扩展

## 参考资料

- Feature Engineering (FE) Tools and Techniques for Better Classification Performance
- Feature engineering
- Feature Engineering and Classifier Ensemble for KDD Cup 2010
- <https://scikit-learn.org/stable/modules/preprocessing.html>

# 预习重点与难点

## ✓ 重点:

- ✓ Python机器学习库与数据预处理
- ✓ 特征工程： 标准化, 特征选择, 特征编码
- ✓ Python参数搜索

## ✓ 难点:

- ✓ 特征清洗与标准化
- ✓ 特征选择
- ✓ 特征扩展

# 学习建议与小助手

- 如何解决英文阅读困难？
  - 有道翻译
  - Google翻译
- 如何解决Bug和环境问题？
  - 百度
  - Bing
  - Google
- 如何生成漂亮作业与报告？
  - 可视化matplotlib
  - Jupyter-notebook
- 课上走神？
  - 回看视频
- 知识理解和交流？
  - 组内小伙伴
- 在线笔记？
  - Onenote
  - 有道笔记
  - evennote