Python机器学习过程

数据分析与数据挖掘

机器学习

- 基于过去的事实和数据，用来发现趋势和模式
- 机器学习模型提供了对于结果的洞察力，机器学习帮助揭示未来的一个结果的概 率而不仅仅是过去发生的事情
- 历史的数据和统计建模被用于概率进行预测

传统的数据分析旨在回答关于过去的事实，机器学习的目的是回答关于未来事件的可能性的问题!

机器学习的应用场景

个性化 – 提供个性化的电子商务体验

文档聚类 – 按照文档上下文自动分类

欺诈检测 – 发现异常的规律行为，识别和标记欺诈交易

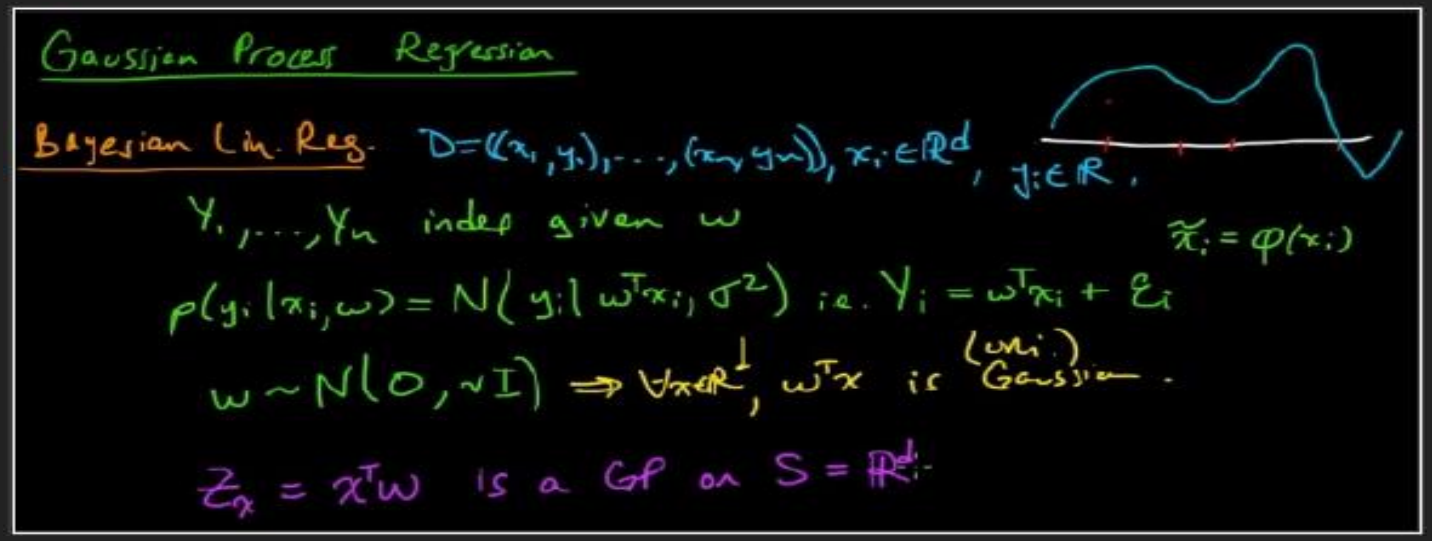
推荐引擎

客户流失预测

...

机器学习 – 学习方式

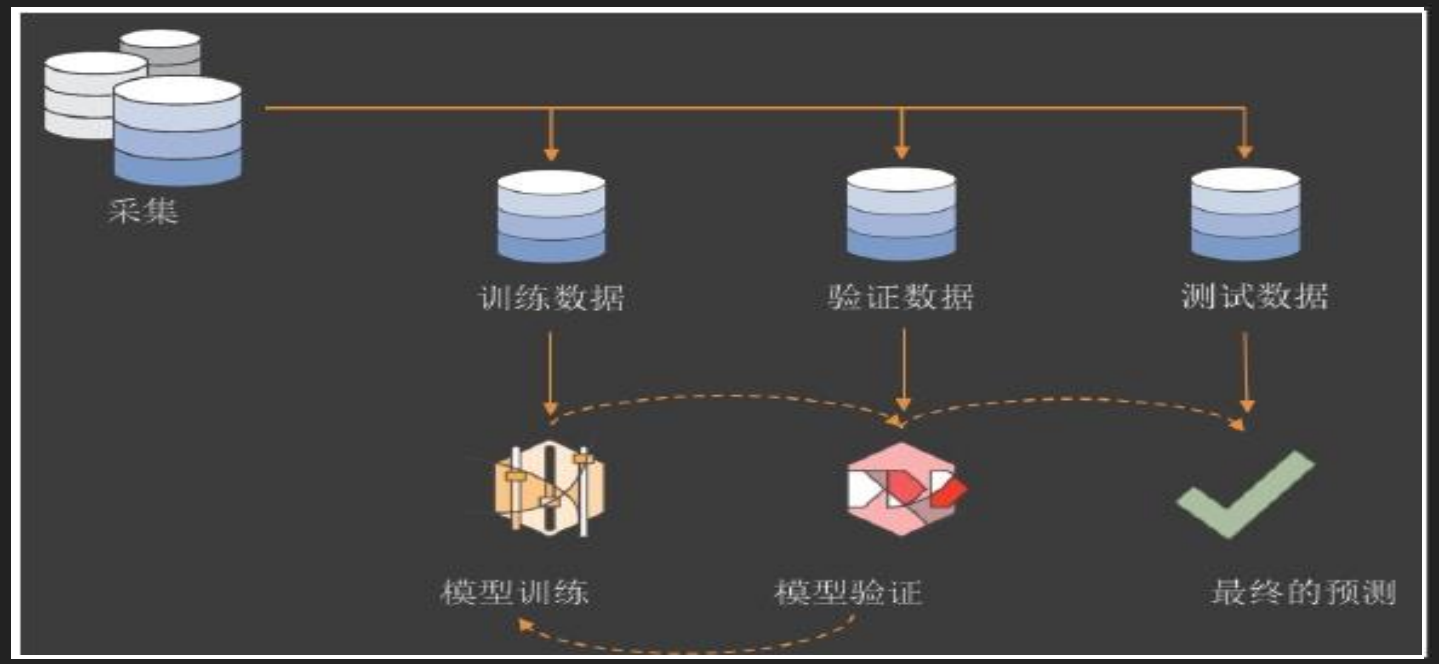
- 监督学习- 人工干预和验证的要求,算法：Logistic Regression，Back Propagation Neural Network 等。例如：照片分类和标签
- 无监督学习- 无人工干预的要求, 算法: Apriori算法以及k-Means。例如：对于文档的基于上下文的自动分类
- 半监督学习 - 介于监督学习和无监督学习之间，算法: Graph Inference 或者Laplacian SVM
- 强化学习- 通过观察来学习做成如何的动作, 算法：Q-Learning以及时间差学习



机器学习 – 方法及流程

- 输入特征选择 – 基于什么进行预测
- 目标 – 预测什么
- 预测功能 – 回归、聚类、降维...

Xn -> F(xn) -> T(x)



机器学习 – 举例

$\chi_n \epsilon F(xn)^{\square}$; Target: y

X_1	X_2	X_3	X_4	X_5	Y
0.3	0.25	0.4	0.34	0.2	1
0.14	0.17	0.2	0.3	0.2	0
0.24	0.21	0.19	0.15	0.35	1
0.3	0.25	0.35	0.4	0.45	1

机器学习 – 举例

- 如何让机器分辨出来他 / 她是谁 ？
- 图像分析 –
  输入特征选择 ->面部特征、发型、裙子、身高、手势...

机器学习 – 何时使用

你不需要机器学习，如果 -

- 使用简单的规则和计算，你可以预测答案
- 你能够预先了解到所需要的步骤不需要任何数据驱动的学习

你需要机器学习，如果 -

- 简单的聚类规则是不充分的
- 面对大量的数据集的可伸缩性的问题

机器学习 – 总结

由已知答案的数据开始

明确目标 – 从数据中希望可以预测什么

选择可以被用来预测目标的模式所需要的变量 / 特性

使用已知目标答案的数据训练机器学习模型

对于未知答案的数据，使用训练过的模型预测目标

评估模型的准确性

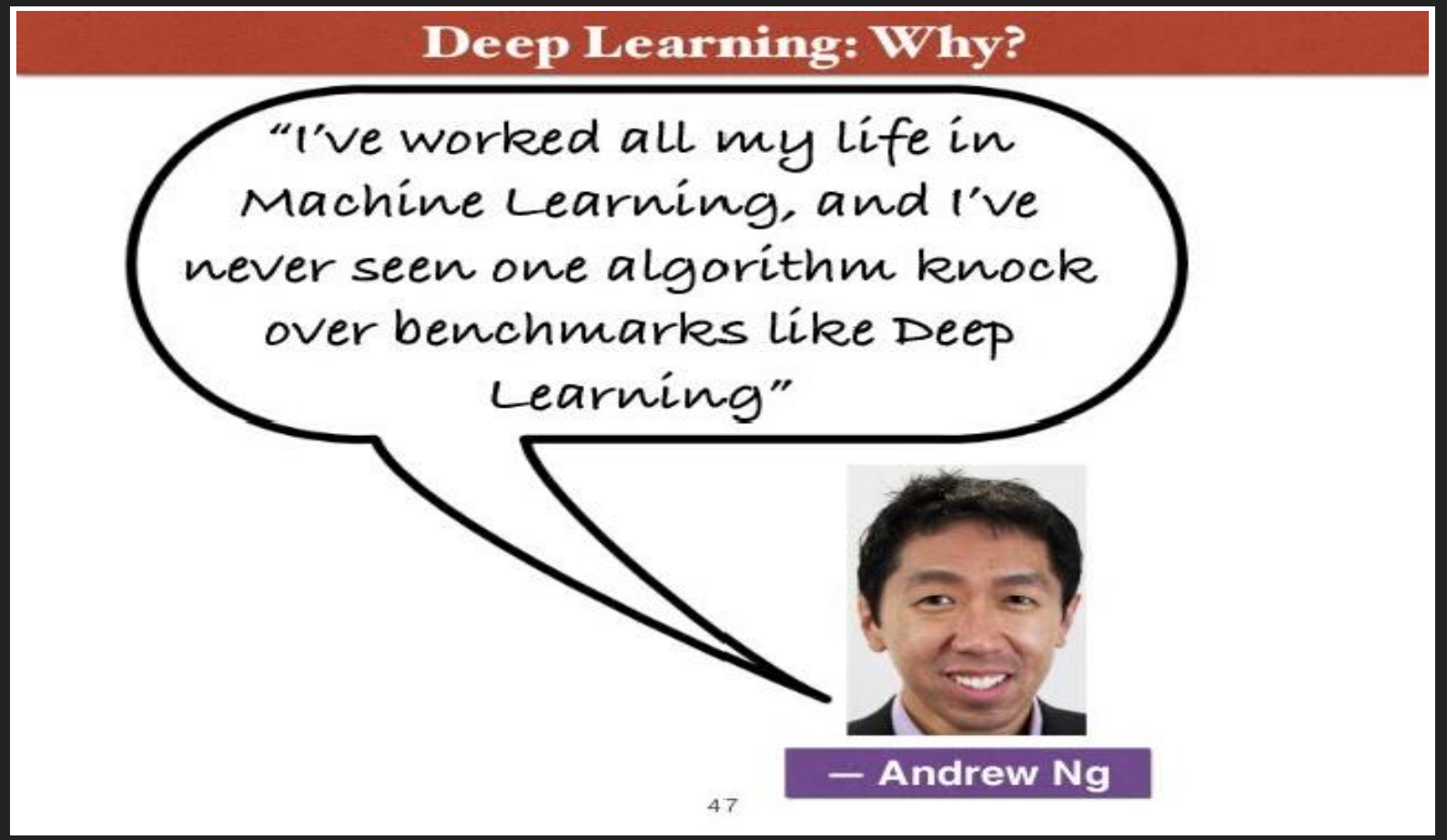
提高模型精度

深度学习 **VS.** 机器学习

- ML 的算法包括监督学习和无监督学习
- 适用非线性处理单元的多层次的特征提取和转换
- 基于对多个层的特征或者表象的学习，形成一个由低级到高级的层次结构特征

传统的机器学习关注于特征工程,深度学习关注于端到端的基于原始数据的学习

为什么需要深度学习？


Deep Learning: Why?
"I've worked all my life in Machine Learning, and I've never seen one algorithm knock over benchmarks like Deep Learning"
— Andrew Ng
47

深度学习- 举例

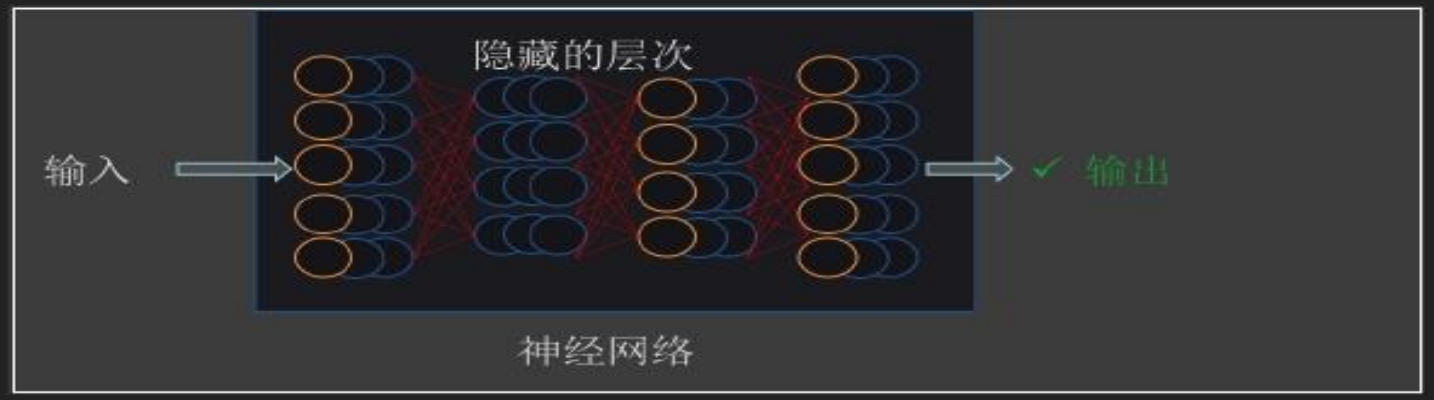

熊
鹰
人
日落
目标: 熊
位置: 黄石公园
动作: 觅食
目标: 鹰
位置: 沃库拉泉， 佛罗里达
动作: 休息
目标: 多个 – 人、 球
位置: 蒙大拿
动作: 玩耍
目标: ?
位置: 蒙大拿
动作: ?

深度学习 – 神经网络

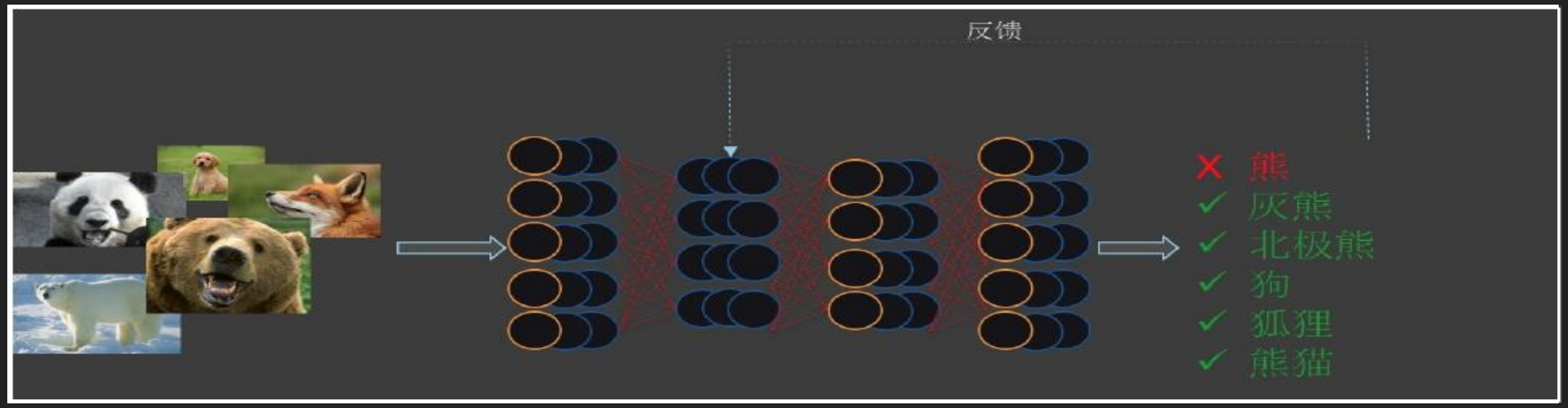
是一种模仿生物神经网络(例如大脑)的结构和功能的计算模型

是一种非线性统计性数据建模工具，对输入和输出间复杂的关系进行建模

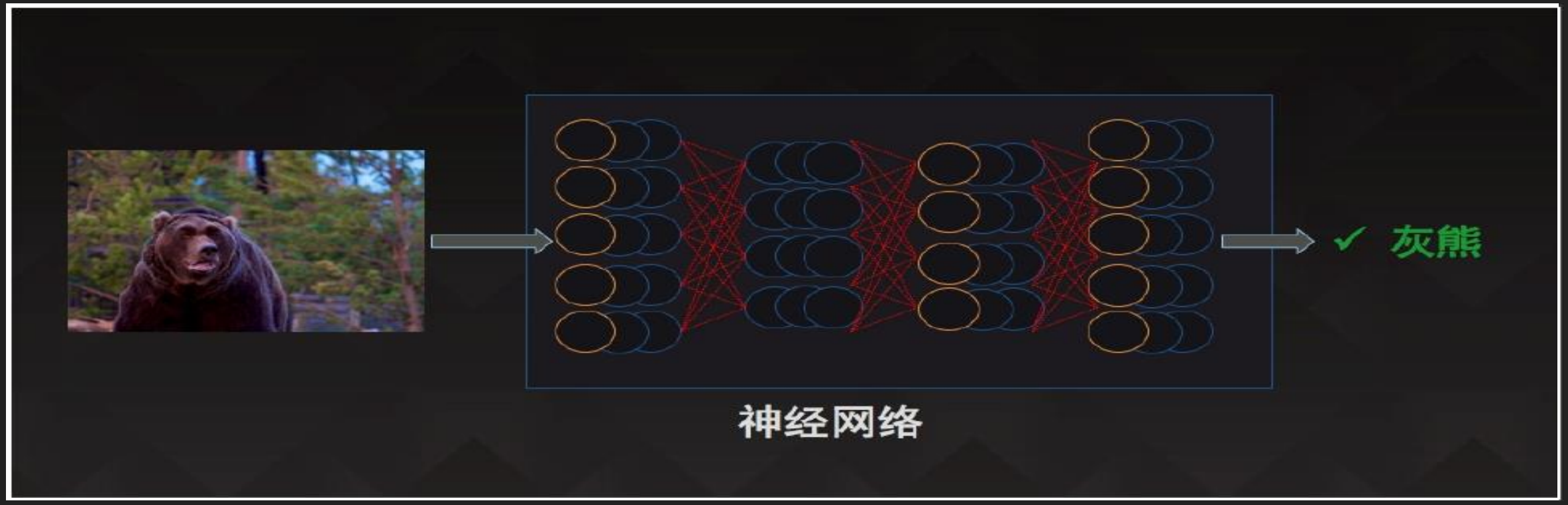
一组简单可以训练的数学单元集合，共同学习复杂的功能



深度学习 – 训练



深度学习 – 部署



深度学习 – 数据表现

- 表现层次
- 图片– 像素、主题、部分、轮廓、边缘等等
- 视频– 图像帧、每帧的像素、每一帧的deltas 值等等
- 文本– 字符、词、从句、句子等等
- 语音– 音频、频段、波长、调制等等

…

深度学习的优势

- 特性自动推导和预期结果的优化调整
- 可变的自动学习的健壮性
- 重用性 – 相同的神经网络的方法可用于许多应用和数据类型
- 通过利用GPU的大规模并行计算 – 可扩展的大容量数据

深度学习的开发框架

- Torch (NYU,2002), Facebook AI, Google Deepmind
- Theano (University of Montreal, ~2010), 学院派
- Kersa, “Deep Learning library for Theano and TensorFlow”
- Caffe (Berkeley)，卷积神经网络,贾扬清
- TensorFlow (Google)
- Spark MLLib

深度学习中的开发框架框架

python has a wide range of deep learning-related libraries available

and of course: scikit learn

High level

Lasagne (theano-extension, models in python code, theano not hidden)

Keras (theano-wrapper, models in python code, abstracts theano away)

Pylearn2 (wrapper for theano, yaml, experiment-oriented)

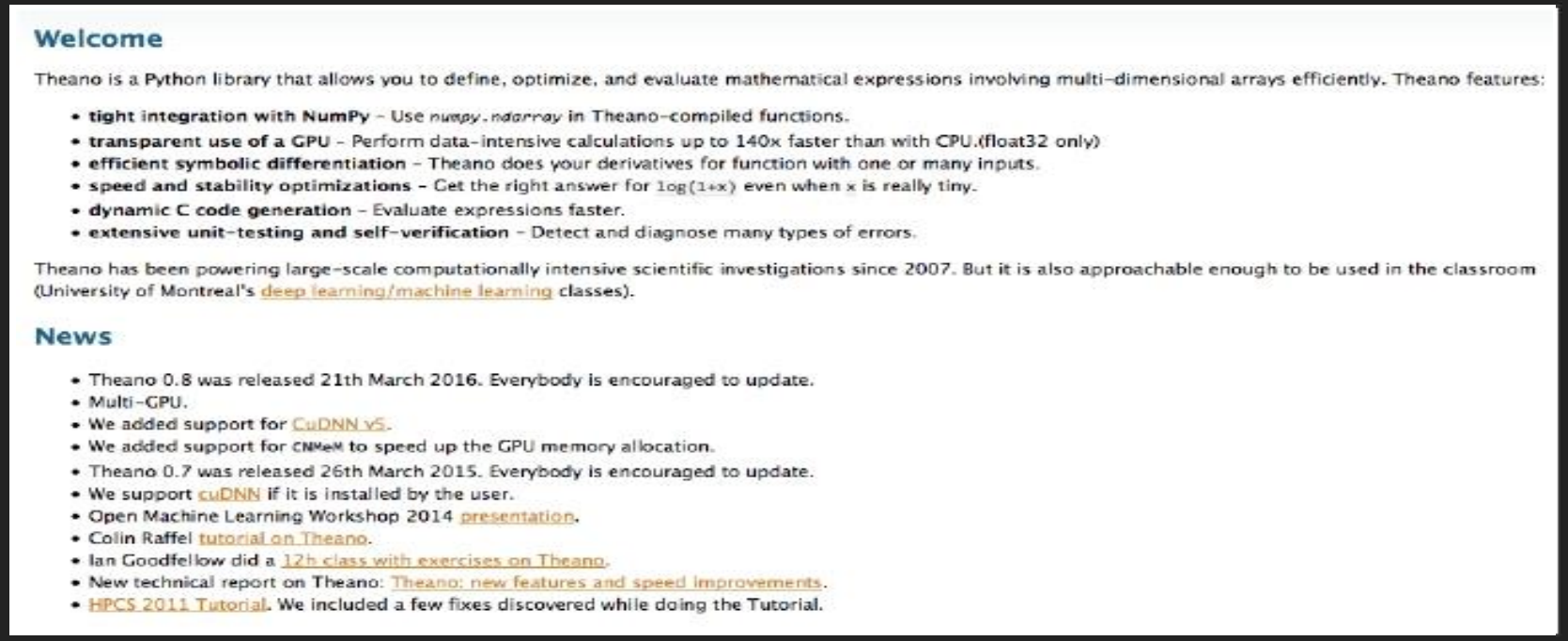
Caffe (computer-vision oriented DL framework, model-zoo, prototxt model definitions) pythonification ongoing!

Low level

theano (efficient gpu-powered math)

THEANO

- 学院派血统,Montreal University
- 非常灵活，非常复杂
- 通过底层借口可以做到大量的定制
- 衍生了大量的丰富的项目Keras, PyLearn2, Lasagne…
- Pythonic API, 非常好的文档



Welcome

Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. Theano features:

- **tight integration with NumPy** – Use *numpy.ndarray* in Theano-compiled functions.
- **transparent use of a GPU** – Perform data-intensive calculations up to 140x faster than with CPU.(float32 only)
- **efficient symbolic differentiation** – Theano does your derivatives for function with one or many inputs.
- **speed and stability optimizations** – Get the right answer for `log(1+x)` even when x is really tiny.
- **dynamic C code generation** – Evaluate expressions faster.
- **extensive unit-testing and self-verification** – Detect and diagnose many types of errors.

Theano has been powering large-scale computationally intensive scientific investigations since 2007. But it is also approachable enough to be used in the classroom (University of Montreal's deep learning/machine learning classes).

News

- Theano 0.8 was released 21th March 2016. Everybody is encouraged to update.
- Multi-GPU.
- We added support for CuDNN v5.
- We added support for `CNMeM` to speed up the GPU memory allocation.
- Theano 0.7 was released 26th March 2015. Everybody is encouraged to update.
- We support cuDNN if it is installed by the user.
- Open Machine Learning Workshop 2014 presentation.
- Colin Raffel tutorial on Theano.
- Ian Goodfellow did a 12h class with exercises on Theano.
- New technical report on Theano: Theano: new features and speed improvements.
- HPCS 2011 Tutorial. We included a few fixes discovered while doing the Tutorial.

THEANO 实践

```
1  import theano
2  from theano import tensor as T
3
4  a = T.scalar()
5  b = T.scalar()
6
7  y = a * b
8
9  multiply = theano.function(inputs=[a, b], outputs=y)
10
11 print multiply(1, 2) #2
12 print multiply(3, 3) #9
13
```

imports

theano symbolic variable initialization

our model

compiling to a python function

usage

THEANO 实践

```
1  import theano
2  from theano import tensor as T
3  import numpy as np
4
5  trX = np.linspace(-1, 1, 101)
6  trY = 2 * trX + np.random.randn(*trX.shape) * 0.33
7
8  X = T.scalar()
9  Y = T.scalar()
10
11 def model(X, w):
12     return X * w
13
14 w = theano.shared(np.asarray(0., dtype=theano.config.floatX))
15 y = model(X, w)
16
17 cost = T.mean(T.sqr(y - Y))
18 gradient = T.grad(cost=cost, wrt=w)
19 updates = [[w, w - gradient * 0.01]]
20
21 train = theano.function(inputs=[X, Y], outputs=cost, updates=updates, allow_input_downcast=True)
22
23 for i in range(100):
24     for x, y in zip(trX, trY):
25         train(x, y)
26
```

imports

training data generation

symbolic variable initialization

our model

model parameter initialization

metric to be optimized by model
learning signal for parameter(s)
how to change parameter based on learning signal

compiling to a python function

iterate through data 100 times and train model
on each example of input, output pairs

THEANO 中的卷积极神经网络

```
1  import theano
2  from theano import tensor as T
3  import numpy as np
4
5  trX = np.linspace(-1, 1, 101)
6  trY = 2 * trX + np.random.randn(*trX.shape) * 0.33
7
8  X = T.scalar()
9  Y = T.scalar()
10
11 def model(X, w):
12     return X * w
13
14 w = theano.shared(np.asarray(0., dtype=theano.config.floatX))
15 y = model(X, w)
16
17 cost = T.mean(T.sqr(y - Y))
18 gradient = T.grad(cost=cost, wrt=w)
19 updates = [[w, w - gradient * 0.01]]
20
21 train = theano.function(inputs=[X, Y], outputs=cost, updates=updates, allow_input_downcast=True)
22
23 for i in range(100):
24     for x, y in zip(trX, trY):
25         train(x, y)
26
```

imports

training data generation

symbolic variable initialization

our model

model parameter initialization

metric to be optimized by model
learning signal for parameter(s)
how to change parameter based on learning signal

compiling to a python function

iterate through data 100 times and train model
on each example of input, output pairs

为什么是 **PYTHON** ？

最好的"胶水"代码用于研究、快速开发

iPython, 数据可视化

丰富的框架资源Theano, Kersa, TensorFlow

海量的社区、开源的支持

为什么需要 GPU？

Hardware [edit]

An early version of AlphaGo was tested on hardware with various numbers of CPUs and GPUs, running in asynchronous or distributed mode. ratings are listed below.[6] In the matches with more time per move higher ratings are achieved.

Configuration and performance

Configuration	Search threads	No. of CPU	No. of GPU	Elo rating
Single[6] p.10-11	40	48	1	2,151
Single	40	48	2	2,738
Single	40	48	4	2,850
Single	40	48	8	2,890
Distributed	12	428	64	2,937
Distributed	24	764	112	3,079
Distributed	40	1,202	176	3,140
Distributed	64	1,920	280	3,168

为什么需要 GPU？

- CPU - 指令并行执行，数据并行运算
- GPU - 矩阵类型的数值计算，尤其浮点运算

利用python进行数据分析

- 分析步骤
  - 定义数据分析目标：明确挖掘数据的目标和达到的效果。
  - 数据取样：采集目标相关样本数据子集，确保数据的相关性、可靠性、有效性。
  - 数据探索：对样本数据探索、审核、加工处理，保证样本数据的质量。
  - 数据预处理：改善数据质量，包括数据筛选、数据变量转换、缺失值数据处理等。
  - 挖掘建模：确定分析问题类型（分类，聚类、关联等），选择相应算法构建模型。
  - 模型评价：从建立模型中找到一个最好的模型，并应用到实际业务中。

4.1 数据探索

- 数据质量分析

  主要任务是检查原始数据中是否存在脏数据，即不符合要求，不能直接处理的数据，包括**缺失值分析、异常值分析、一致性分析。**

- 数据特征分析
  - **分布分析：**揭示数据的分布特征和分布类型，通过绘制频率分布表、茎叶图等直观分析
  - **对比分析：**把两个相互联系的指标进行比较，从数量上展示和说明研究对象规模的大小，水平的高低，速度的快慢，以及各种关系是否协调。
  - **统计量分析：**用统计量指标对定量数据进行统计描述，常从集中趋势和离中趋势两个方面进行分析。
  - **相关性分析：**分析连续变量之间线性相关程度的强弱，并用适当的统计指标表示出来。

4.2 数据预处理

- 数据清洗
  - 删除原始数据集中的无关数据、重复数据、平滑噪声数据、无关数据，处理缺失值和异常值。
- 数据集成
  - 将多个数据源合并存放在一个一致的数据存储（如数据仓库）中的过程。
- 数据变换
  - 主要是对数据进行规范化处理，将数据转换成适当的形式，以适用于挖掘任务和算法的需要。
- 数据规约
  - 产生更小但保持数据完整性的新数据集，在规约后的数据集上进行分析和挖掘更有效率。

Python主要数据预处理函数

函数名	函数功能	所属扩展库
interpolate	一维、高维数据插值	Scipy
unique	去除数据中重复元素，得到单值元素列表	Pandas/Numpy
isnull	判断是否是空值	Pandas
notnull	判断是否非空值	Pandas
PCA	对指标变量矩阵进行主成分分析	Scikit-Learn
random	生成随机矩阵	Numpy

4.3 挖掘建模

- 分类与预测
  - 分类：构造一个分类模型，输入样本的属性值，输出对应的类别，将每个样本映射到预先定义好的类别
  - 预测：建立两种或两种以上变量间相互依赖的函数模型，然后进行预测和控制
  - 实现过程

① 学习步，通过归纳分析训练样本集来建立分类模型得到分类规则

② 分类步，先用一直的测试样本集评估分类规则的准确率，如果准确率是可以接受的，则使用该模型对未知类标号的待测样本集进行预测

- 常用的分类与预测算法

算法分析	算法描述
回归分析	回归分析是确定去测属性（数值型）与其他变量间相互依赖的定量关系最常用的统计学方法。包括线性回归、非线性回归、Logistic回归、岭回归、主成分回归、偏最小二乘回归等模型
决策树	决策树采用自顶向下的递归方式，在内部节点进行属性值的比较，并根据不同的属性值从该节点向下分支，最终得到的叶节点是学习划分的类
人工神经网络	人工神经神经网络是一种模仿大脑神经网络和功能而建立的信息处理系统，表示神经网络的输入与输出变量之间关系的模型
贝叶斯网络	贝叶斯网络又称信度网络，是Bayes方法的扩展，是目前不确定知识表达和推理领域最有效的理论模型之一
支持向量机	支持向量机是一种通过魔种非线性映射，把低纬的非线性可分转化为高维的线性可分，在高维空间进行线性分析的算法

- 主要回归模型分类

回归模型名称	试用条件	算法描述
线性回归	因变量与自变量是线性关系	对一个或多个自变量和因变量之间的线性关系进行建模可用最小二乘法求解模型系数
非线性回归	因变量与自变量之间不都是线性关系	对一个或多个自变量和因变量之间的非线性关系进行建模。如果非线性关系可以通过简单的函数变换转化成线性关系，用线性回归的思想求解；如果不能转化，用非线性最小二乘法方法求解
Logistic	因变量一般有1和0两种取值	是广义线性回归模型的特例，利用Logistic函数将因变量的取值范围控制在0和1之间，表示取值为1的概率
岭回归	参与建模的自变量之间具有多重共线性	是一种改进最小二乘估计的方法
主成分回归	参与建模的自变量之间具有多重共线性	主成分回归是根据主成分分析的思想提出来，是对最小二乘法的一种改进，它是参数估计的一种有偏估计。可以消除自变量之间的多重共线性

应用举例一

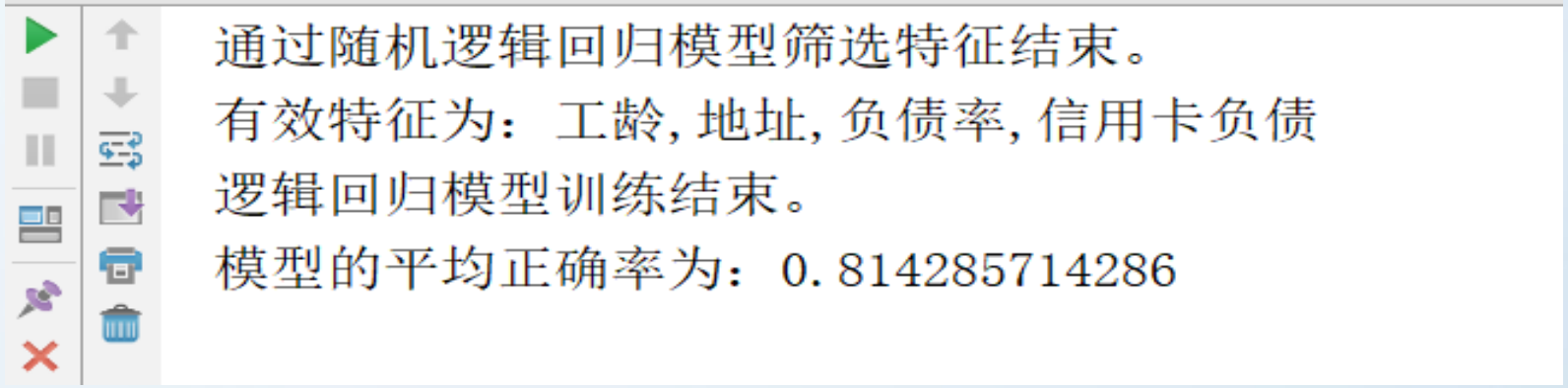
对某银行在降低贷款拖欠率的数据进行逻辑回归建模，数据示例如下表

	A	B	C	D	E	F	G	H	I
1	年龄	教育	工龄	地址	收入	负债率	信用卡负债	其他负债	违约
2	41	3	17	12	176.00	9.30	11.36	5.01	1
3	27	1	10	6	31.00	17.30	1.36	4.00	0
4	40	1	15	14	55.00	5.50	0.86	2.17	0
5	41	1	15	14	120.00	2.90	2.66	0.82	0
6	24	2	2	0	28.00	17.30	1.79	3.06	1
7	41	2	5	5	25.00	10.20	0.39	2.16	0
8	39	1	20	9	67.00	30.60	3.83	16.67	0
9	43	1	12	11	38.00	3.60	0.13	1.24	0
10	24	1	3	4	19.00	24.40	1.36	3.28	1
11	36	1	0	13	25.00	19.70	2.78	2.15	0
12	27	1	0	1	16.00	1.70	0.18	0.09	0
13	25	1	4	0	23.00	5.20	0.25	0.94	0
14	52	1	24	14	64.00	10.00	3.93	2.47	0
15	37	1	6	9	29.00	16.30	1.72	3.01	0
16	48	1	22	15	100.00	9.10	3.70	5.40	0
17	36	2	9	6	49.00	8.60	0.82	3.40	1
18	36	2	13	6	41.00	16.40	2.92	3.81	1

- Python代码

```python
#-*- coding: utf-8 -*-
#逻辑回归 自动建模
import pandas as pd

#参数初始化
filename = 'D://data/bankloan.xls'
data = pd.read_excel(filename)
x = data.iloc[:,:8].as_matrix()
y = data.iloc[:,8].as_matrix()

from sklearn.linear_model import LogisticRegression as LR
from sklearn.linear_model import RandomizedLogisticRegression as RLR
rlr = RLR() #建立随机逻辑回归模型，筛选变量
rlr.fit(x, y) #训练模型
rlr.get_support() #获取特征筛选结果，也可以通过.scores_方法获取各个特征的分数
print(u'通过随机逻辑回归模型筛选特征结束。')
print(u'有效特征为：%s' % ','.join(data.columns[rlr.get_support()]))
x = data[data.columns[rlr.get_support()]].as_matrix() #筛选好特征
lr = LR() #建立逻辑货柜模型
lr.fit(x, y) #用筛选后的特征数据来训练模型
print(u'逻辑回归模型训练结束。')
print(u'模型的平均正确率为：%s' % lr.score(x, y)) #给出模型的平均正确率，本例为81.4%
```

- 运行结果



- 结果分析
  - 随机逻辑回归剔除变量，分别剔除了x2、x8、x1、x5，最终构建模型包含的变量为常量x3、x4、x6、x7。
  - 在建立逻辑回归模型时，使用了默认的阈值0.25。

- 聚类分析
  - 在没有给定划分类别的情况下，根据数据相似度进行样本分组的一种方法。

常用聚类方法

类别	包括的主要算法
划分方法	K-Means算法、K-MEDOIDS算法、CLARANS算法
层次分析法	BIRCH算法、CURE算法、CHAMELEON算法
基于密度的方法	DBCSCAN算法、DENCLUE算法、OPTICS算法
基于网格的方法	STING算法、CLIOUE算法、WAVE——CLUSTER算法
基于模型的方法	统计学方法、神经网络方法

- 常用聚类分析算法

算法名称	算法描述
K-Means	K-均值聚类也称为快速聚类法，在最小化误差函数的基础上将数据划分为预定的类数K。该算法原理简单并便于处理大量数据
K-中心点	K-均值算法对孤立点的敏感性，K-中心点算法不采用簇中对象的平均值作为簇中心，而选用簇中离平均值最近的对象作为簇中心
系统聚类	系统聚类也称为多层次聚类，分类的单位由高到低呈树形结构，且所处的位置越低，其包含的对象就越少，但这些对象间的共同特征越多。该聚类方法只适用在小数据量的时候使用，数据量大的时候速度会非常慢

K-Means聚类算法

- 算法过程
  - 从N个样本数据中随机选取K个对象作为初始的聚类中心
  - 分别计算每个样本到各个聚类中心的距离，将对象分配到距离不最近的聚类中
  - 所有对象分配完成后，重新计算K个聚类的中心
  - 与前一次计算得到的K个聚类中心比较，如果聚类中心发生变化，转第二步，否则转下一步
  - 当质心不发生变化时停止并输出聚类不结果

应用举例二

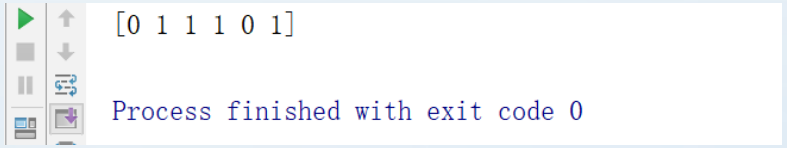
找出下列谁是学霸？

	高数	英语	C++	音乐
小明	88	64	96	85
大明	92	99	95	94
小鹏	91	87	99	95
大鹏	78	99	97	81
小萌	88	78	98	84
大萌	100	95	100	92

- 使用Kmeans对学生成绩进行聚类

```python
from numpy import vstack
from scipy.cluster.vq import kmeans,vq

list1 = [88.0, 64.0, 96.0, 85.0]
list2 = [92.0, 99.0, 95.0, 94.0]
list3 = [91.0, 87.0, 99.0, 95.0]
list4 = [78.0, 99.0, 97.0, 81.0]
list5 = [88.0, 78.0, 98.0, 84.0]
list6 = [100.0, 95.0, 100.0, 92.0]

data = vstack((list1, list2, list3, list4, list5, list6)) #vstack:堆积成绩数据
centroids,_ = kmeans(data, 2)     #kmeans函数返回的centroids就是聚类中心；2表示分类的类别数目
result,_ = vq(data,centroids)       #vq函数用来获取每一位同学所属的
print(result)
```

- 运行结果



```
[0 1 1 1 0 1]

Process finished with exit code 0
```

- 结论
  - 大明、小鹏、大鹏、大萌是学霸

5.1 Titanic数据集分析

- 前期准备
  - 数据下载
    https://d17h27t6h515a5.cloudfront.net/topher/2016/December/584bcec3_titanic-data/titanic-data.csv
  - 软件准备

    python3.6+ anaconda 或 使用集成开发环境 pycharm

- 数据格式

titanic-data.csv

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,7.925,,S
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,1,0,113803,53.1,C123,S
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.05,,S
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.8625,E46,S
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,349909,21.075,,S
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)",female,27,0,2,347742,11.1333,,S
```

PassengerId => 乘客ID
Survived => 是否生还
Pclass => 乘客等级(1/2/3等舱位)
Name => 乘客姓名
Sex => 性别
Age => 年龄
SibSp => 堂兄弟/妹个数
Parch => 父母与小孩个数
Ticket => 船票信息
Fare => 票价
Cabin => 客舱
Embarked => 登船港口

1、导入数据&查看基本信息

wocanmei.py

```
1  import numpy as np
2  import pandas as pd
3  import matplotlib.pyplot as plt
4  data_src='D://titanic-data.csv'
5  df = pd.read_csv(data_src,header=0) # 导入数据
6  print(df.info())  # 查看数据集的基本信息,
7  print(df.describe()) # 查看数据的摘要信息
8  print(df.head())  # 查看前几行数据，方便了解数据具体情况
```

– 运行结果

titanic

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId     891 non-null int64
Survived        891 non-null int64
Pclass          891 non-null int64
Name            891 non-null object
Sex             891 non-null object
Age             714 non-null float64
SibSp           891 non-null int64
Parch           891 non-null int64
Ticket          891 non-null object
Fare            891 non-null float64
Cabin           204 non-null object
Embarked        889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
None
```

titanic

	PassengerId	Survived	Pclass	Age	SibSp	\
count	891.000000	891.000000	891.000000	714.000000	891.000000	
mean	446.000000	0.383838	2.308642	29.699118	0.523008	
std	257.353842	0.486592	0.836071	14.526497	1.102743	
min	1.000000	0.000000	1.000000	0.420000	0.000000	
25%	223.500000	0.000000	2.000000	20.125000	0.000000	
50%	446.000000	0.000000	3.000000	28.000000	0.000000	
75%	668.500000	1.000000	3.000000	38.000000	1.000000	
max	891.000000	1.000000	3.000000	80.000000	8.000000	

	Parch	Fare
count	891.000000	891.000000
mean	0.381594	32.204208
std	0.806057	49.693429
min	0.000000	0.000000
25%	0.000000	7.910400
50%	0.000000	14.454200
75%	0.000000	31.000000
max	6.000000	512.329200

从数据集的基本信息可以看出，Age \ Cabin \ Embarked 是存在缺失值的，其中Cabin字段缺失值过多。
常用的方法是去除和补齐，数值型的数据是可以根据统计学的方法或者机器学习的方法将其进行补齐的

2、分析乘客存活率与各单变量之间的关系

- 查看总存活率

```
survived_rate = float(df['Survived'].sum()) / df['Survived'].count()
Print(‘survived_rate: ',survived_rate)
```

- 输出结果

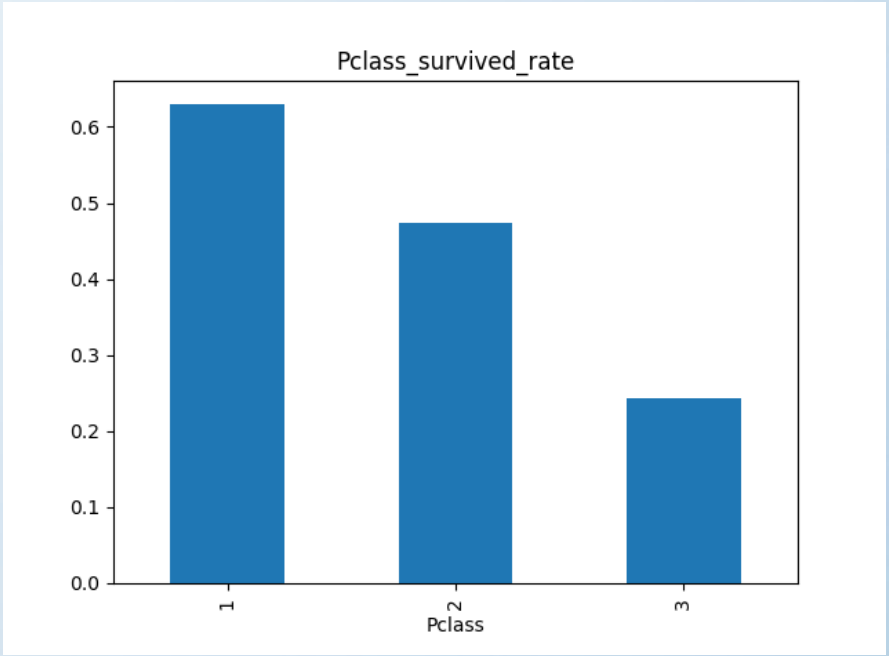
```
survived_rate:  0.383838383838
```

- **舱位与存活率关系**

```
x=[df[(df.Pclass==1)]['Pclass'].size,df[(df.Pclass==2)]['Pclass'].size,df[(df.Pclass==3)]['Pclass'].size]
y=[df[(df.Pclass==1) & (df.Survived == 1)]['Pclass'].size,df[(df.Pclass==2) & (df.Survived == 1)]['Pclass'].size,df[(df.Pclass == 3) & (df.Survived == 1)]['Pclass'].size]
print('1 Pclass number:' + str(x[0]) + '     ' +'1 Pclass survive:' + str(y[0]) + '     ' +'1 Pclass survive rat:', float(y[0]) / x[0])
print('2 Pclass number:' + str(x[1]) + '     ' +'2 Pclass survive:' + str(y[1]) + '     ' +'2 Pclass survive rat:', float(y[1]) / x[1])
print('3 Pclass number:' + str(x[2]) + '     ' +'3 Pclass survive:' + str(y[2]) + '     ' +'3 Pclass survive rat:', float(y[2]) / x[2])

Pclass_survived_rate = (df.groupby(['Pclass']).sum() / df.groupby(['Pclass']).count())['Survived']
Pclass_survived_rate.plot(kind='bar')
plt.title('Pclass_survived_rate')
plt.show()
```

- **运行结果**

```
titanic
1 Pclass number:216    1 Pclass survive:136    1 Pclass survive rat: 0.6296296296296297
2 Pclass number:184    2 Pclass survive:87    2 Pclass survive rat: 0.47282608695652173
3 Pclass number:491    3 Pclass survive:119    3 Pclass survive rat: 0.24236252545824846
```

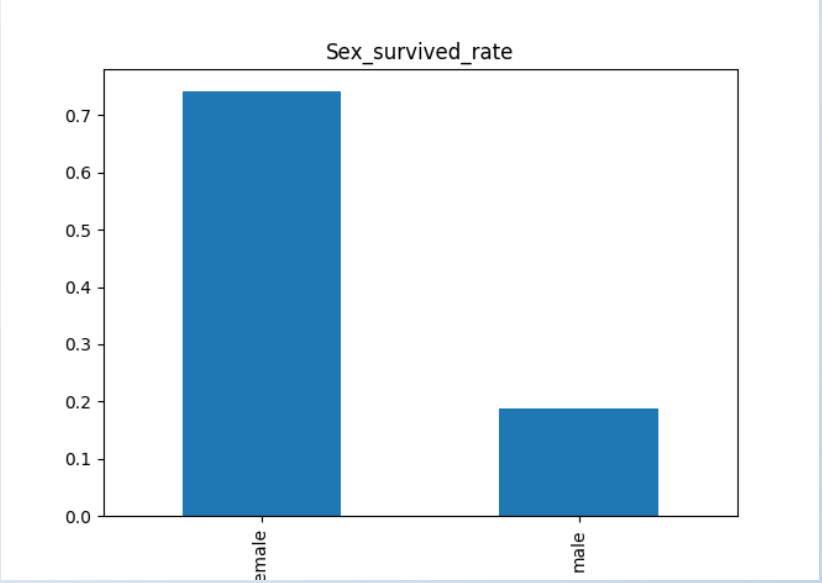


- **性别与存活率关系**

```
male_suviced=df[(df.Sex=='male')]['Sex'].size
female_suviced=df[(df.Sex=='female')]['Sex'].size
print('male survive:',male_suviced)
print('female survive:',female_suviced)
Sex_survived_rate = (df.groupby(['Sex']).sum() / df.groupby(['Sex']).count())['Survived']
Sex_survived_rate.plot(kind='bar')
plt.title('Sex_survived_rate')
plt.show()
```

- **运行结果**

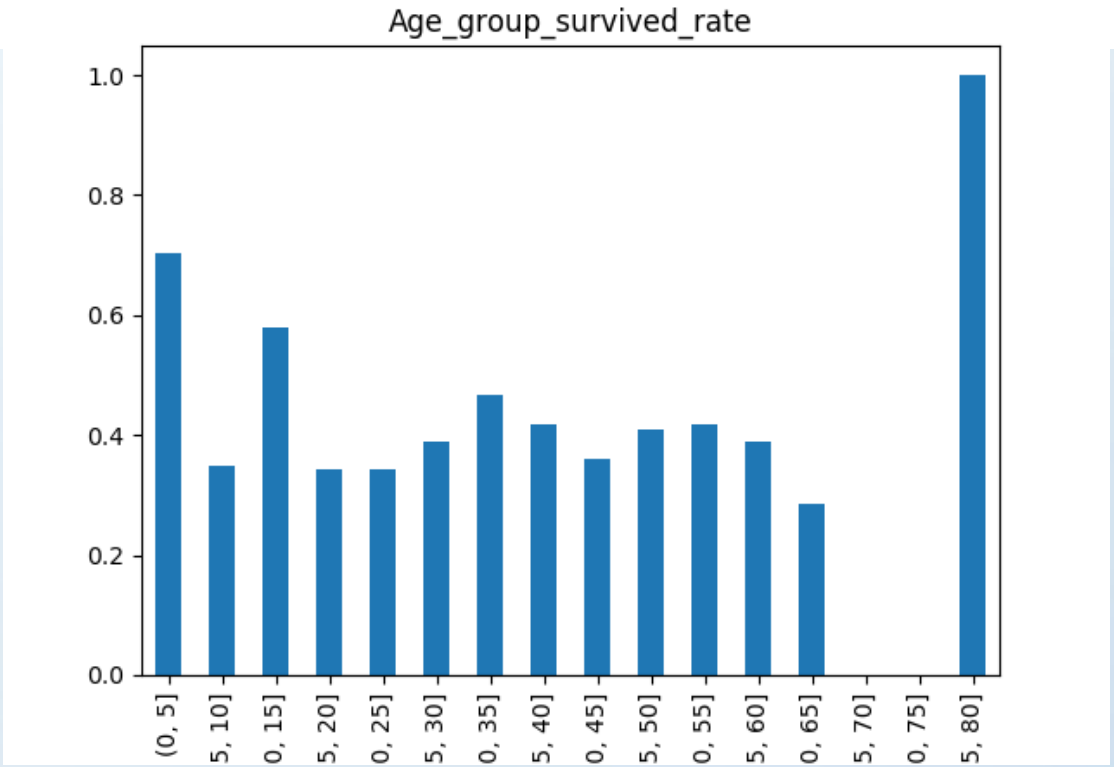
```
titanic
male survive: 577
female survive: 314

Process finished with exit code 0
```



- **年龄与存活率关系**

```
7   age_clean_date=df[~np.isnan(df['Age'])]      #去除年龄数据中的NaN
8   ages=np.arange(0,81,5)                       #0~80岁，每5岁一段（年龄最大80岁）
9   age_cut=pd.cut(age_clean_date.Age,ages)
10  age_cut_grouped=age_clean_date.groupby(age_cut)
11  age_Survival_Rate=(age_cut_grouped.sum()/age_cut_grouped.count())['Survived'] #计算每年龄段的幸存率
12  age_Survival_Rate.plot(kind='bar')
13  plt.title('Age_group_survived_rate')
14  plt.show()
15
```

- **运行结果**

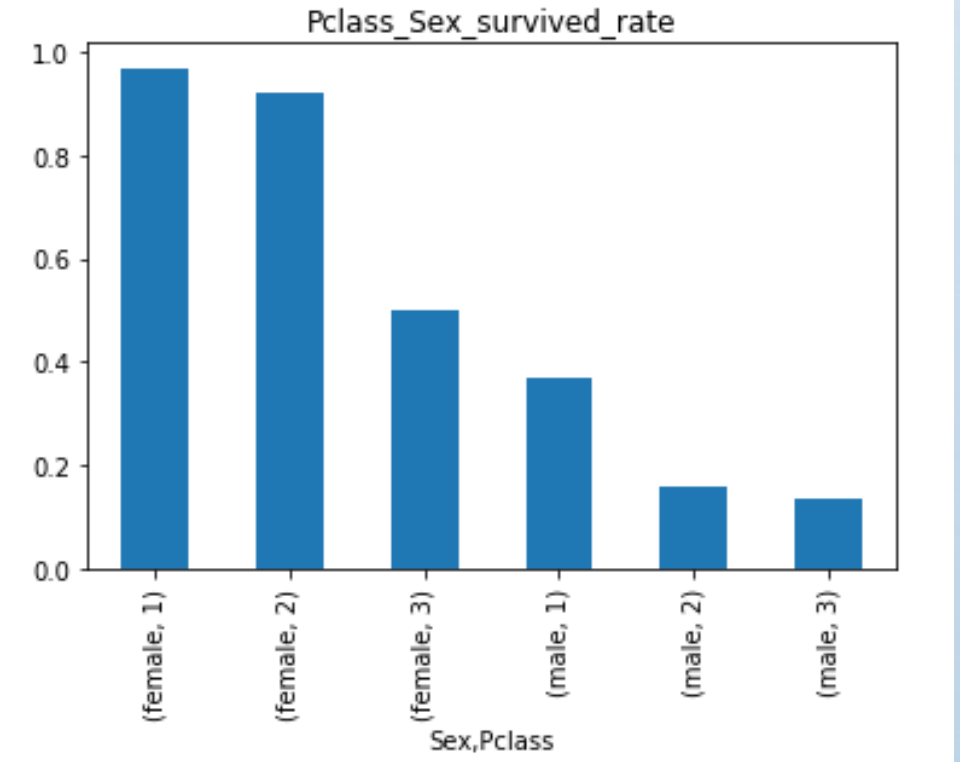


3、分析乘客存活率与复合变量之间的关系

- 根据上述变量分析，舱位和性别对存活率影响都很大，但哪一个占的比重更大呢？
- 将舱位和性别整合为复合变量进行分析

```
7   Pclass_Sex_survived_rate=(df.groupby(['Sex','Pclass']).sum()/df.groupby(['Sex','Pclass']).count())['Survived']
8   Pclass_Sex_survived_rate.plot(kind='bar')
9   plt.title('Pclass_Sex_survived_rate')
10  plt.show()
11
```

- **输出结果**



- **结果分析**
  - **船舱等级越高，存活率越高**
  - **女性存活率高于男性**

5.2 餐饮客户价值分析

- 部分餐饮客户的消费行为特征数据如下，根据数据将客户分成不同客户群，并评价这些客户群的价值

	A	B	C	D	E
1	Id	R	F	M	
2	1	27	6	232.61	
3	2	3	5	1507.11	
4	3	4	16	817.62	
5	4	3	11	232.81	
6	5	14	7	1913.05	
7	6	19	6	220.07	
8	7	5	2	615.83	
9	8	26	2	1059.66	
10	9	21	9	304.82	
11	10	2	21	1227.96	
12	11	15	2	521.02	
13	12	26	3	438.22	
14	13	17	11	1744.55	

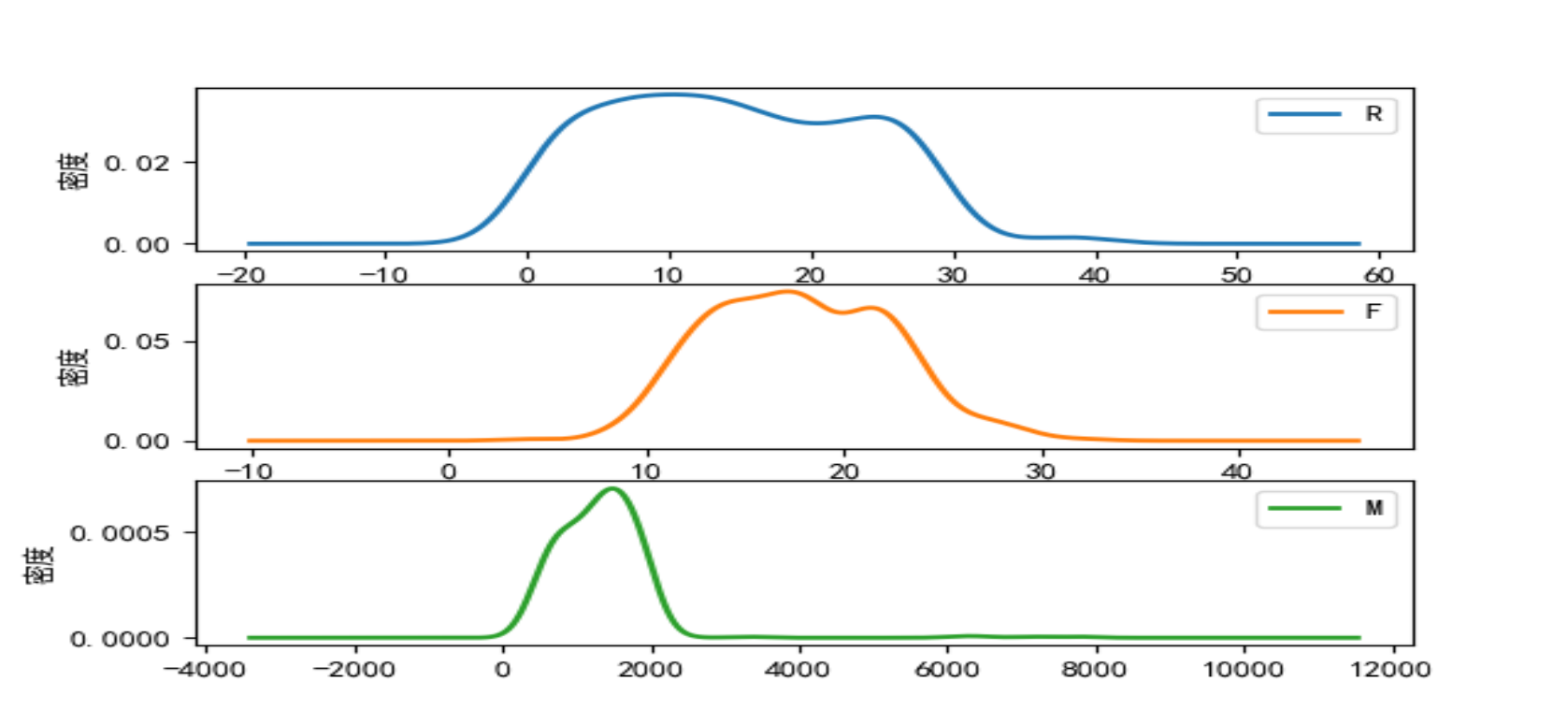
- Python代码

```python
#-*- coding: utf-8 -*-
#使用K-Means算法聚类消费行为特征数据
import pandas as pd
#参数初始化
inputfile = '../data/consumption_data.xls' #销量及其他属性数据
outputfile = '../tmp/data_type.xls' #保存结果的文件名
k = 3 #聚类的类别
iteration = 500 #聚类最大循环次数
data = pd.read_excel(inputfile, index_col = 'Id') #读取数据
data_zs = 1.0*(data - data.mean())/data.std() #数据标准化
from sklearn.cluster import KMeans
model = KMeans(n_clusters = k, n_jobs = 1, max_iter = iteration) #分为k类，并发数4
model.fit(data_zs) #开始聚类
#简单打印结果
r1 = pd.Series(model.labels_).value_counts() #统计各个类别的数目
r2 = pd.DataFrame(model.cluster_centers_) #找出聚类中心
r = pd.concat([r2, r1], axis = 1) #横向连接（0是纵向），得到聚类中心对应的类别下的数目
r.columns = list(data.columns) + [u'类别数目'] #重命名表头
#详细输出原始数据及其类别
r = pd.concat([data, pd.Series(model.labels_, index = data.index)], axis = 1)  #详细输出每个样本对应的类别
r.columns = list(data.columns) + [u'聚类类别'] #重命名表头
r.to_excel(outputfile) #保存结果
def density_plot(data): #自定义作图函数
  import matplotlib.pyplot as plt
  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
  p = data.plot(kind='kde', linewidth = 2, subplots = True, sharex = False)
  [p[i].set_ylabel(u'密度') for i in range(k)]
  plt.legend()
  return plt
pic_output = '../tmp/pd_' #概率密度图文件名前缀
for i in range(k):
  density_plot(data[r[u'聚类类别']==i]).savefig(u'%s%s.png' %(pic_output, i))

```

- 运行结果

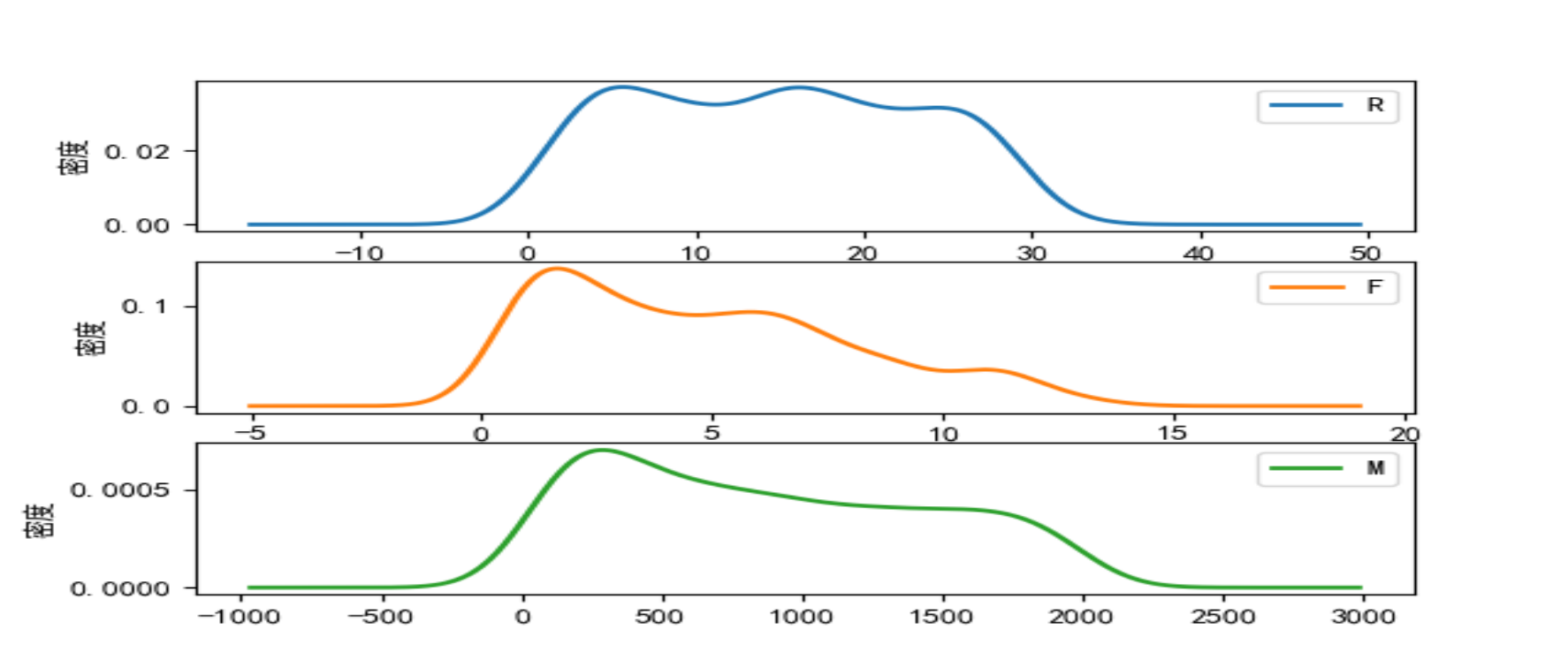
	A	B	C	D	E	F
1	Id	R	F	M	聚类类别	
2	1	27	6	232.61	1	
3	2	3	5	1507.11	1	
4	3	4	16	817.62	0	
5	4	3	11	232.81	1	
6	5	14	7	1913.05	1	
7	6	19	6	220.07	1	
8	7	5	2	615.83	1	
9	8	26	2	1059.66	1	
10	9	21	9	304.82	1	
11	10	2	21	1227.96	0	
12	11	15	2	521.02	1	
13	12	26	3	438.22	1	
14	13	17	11	1744.55	0	
15	14	30	16	1957.44	0	
16	15	5	7	1713.79	1	
17	16	4	21	1768.11	0	
18	17	93	2	1016.34	2	

- 分群一结果分析



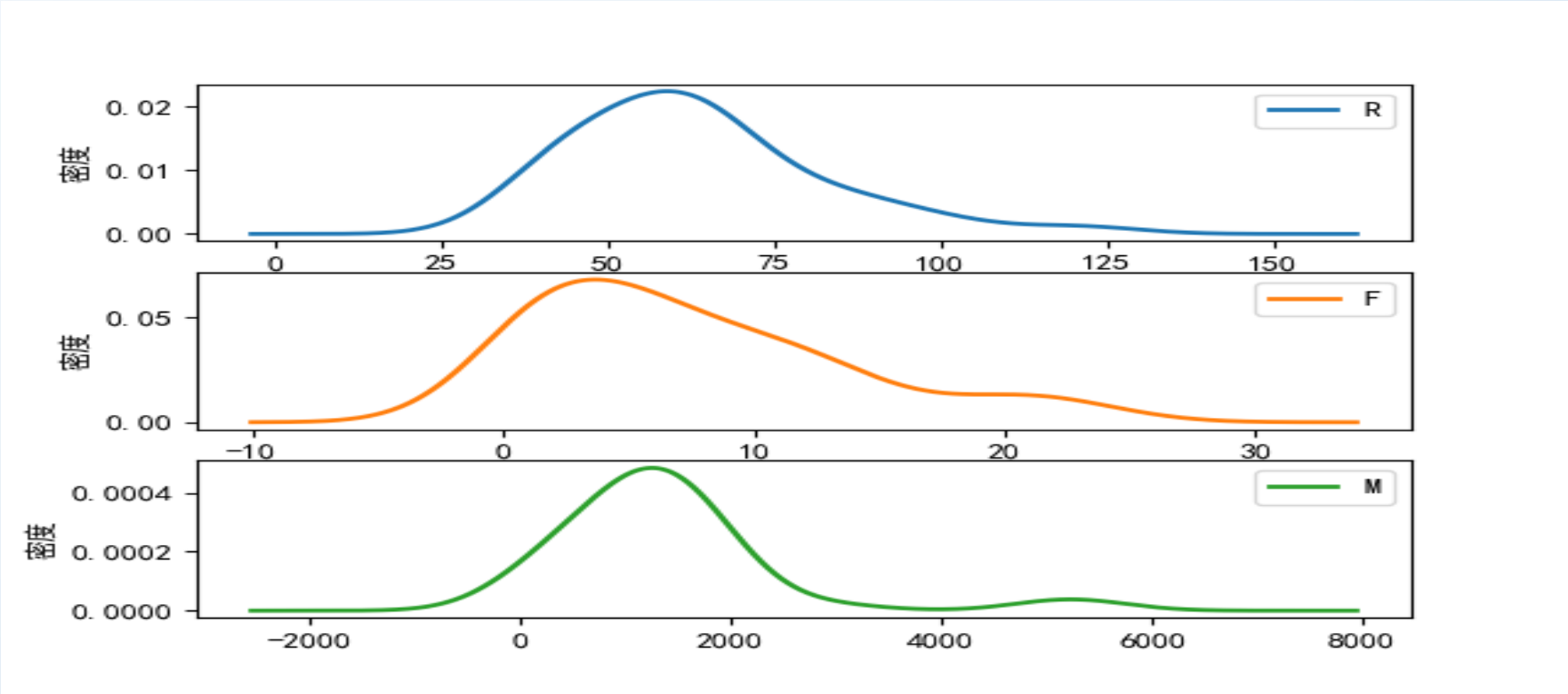
分群一的R间隔相对较小，主要集中在0~30天，消费次数集中在10~25次，消费金额在500~2000

- 分群二结果分析



分群二的R间隔分布在0~30天，消费次数集中在0~12次，消费金额在0~1800

- 分群三结果分析



分群三的R间隔较大，间隔分布在30~80天，消费次数集中在0~15次，消费金额在0~2000

- 对比分析
  - 分群1的时间间隔较短，消费次数多，而且消费金额较大，时高消费、高价值人群。
  - 分群二的时间间隔、消费次数和消费金额处于中等水平，代表着一般客户。
  - 分群三的时间间隔较长，消费次数较少，消费金额也不是特别高，是价值较低的客户群体。