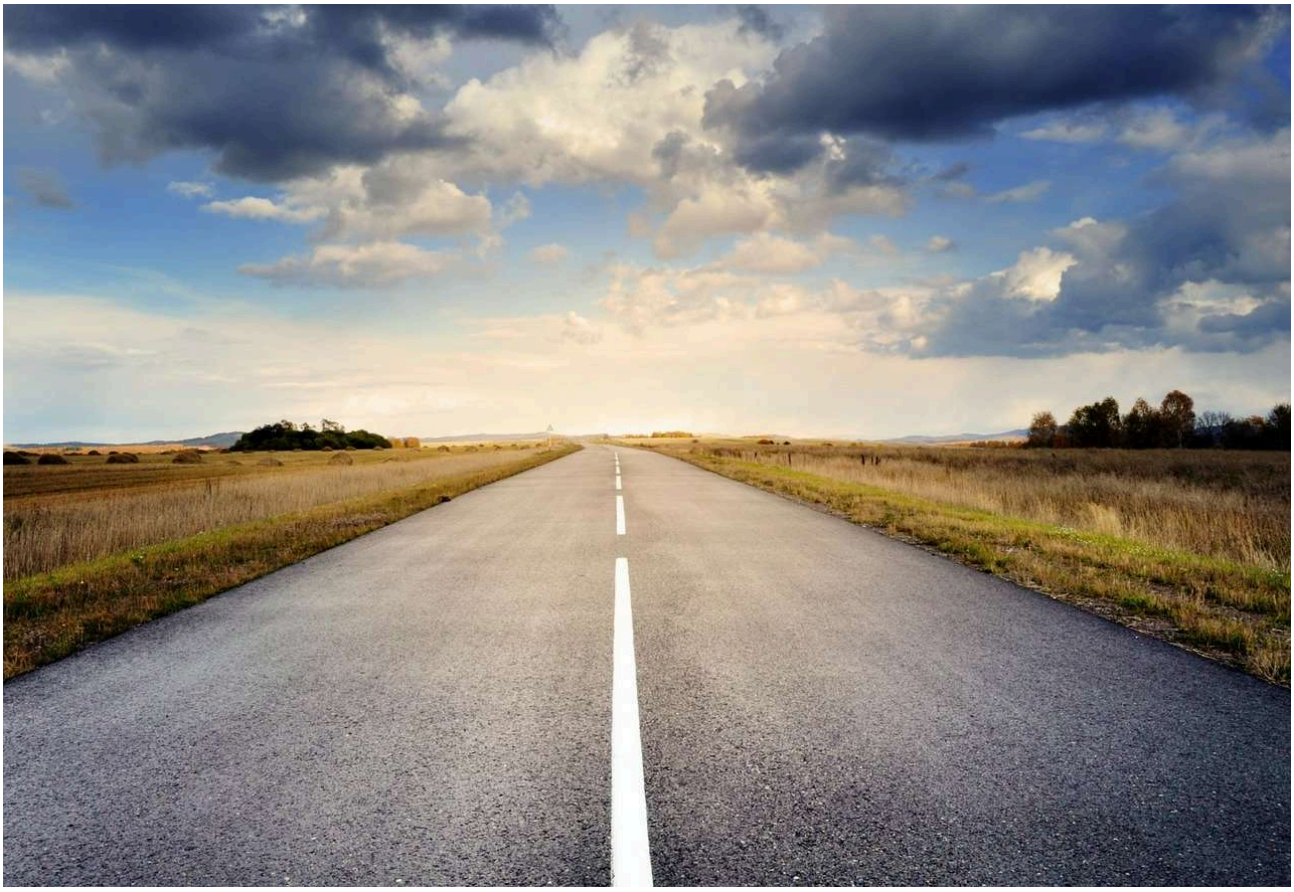


Github上大神总结的Python学习路线

阿良
半途而废的医学生/AI+RPA从业者/《自学区块链》作者

关注他

收录于 • 编程学习路线与面试资源 >

俗话说，师傅领进门，修行在个人。随着Python的流行，越来越多的人希望学习Python，苦于没有师傅来领进门。本文将分享几个GitHub大佬总结的经验和免费学习资料，供同学们学习参考。

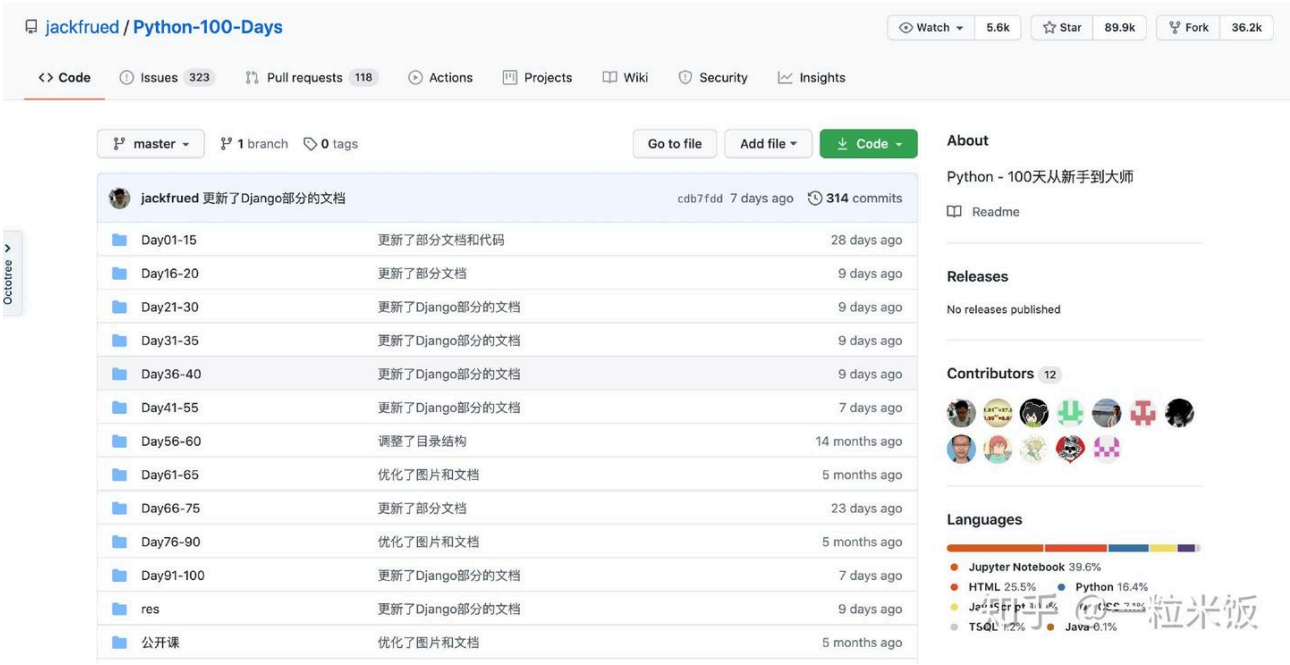
1. Python - 100天从新手到大师

大神骆昊 (github账号：jackfrued)为大家规划了一条从“**从新手到大师**”的Python百天之路。

一百天的内容如下：

Day01~15 - Python语言基础
Day16~Day20 - Python语言进阶
Day21~30 - Web前端入门
Day31~35 - 玩转Linux操作系统
Day31~35 - 玩转Linux操作系统

Day41~55 - 实战Django+
Day56~60 - 实战Flask+
Day61~65 - 实战Tornado+
Day66~75 - 爬虫开发
Day76~90 - 数据分析和机器学习
Day91~100 - 团队项目开发



github:

github.com/jackfrued/Py

⭐ **star**:89.9k

给初学者的几个建议：

- Make English as your working language. （让英语成为你的工作语言）
- Practice makes perfect. （熟能生巧）
- All experience comes from mistakes. （所有的经验都源于你犯过的错误）
- Don't be one of the leeches. （不要当伸手党）
- Either outstanding or out. （要么出众，要么出局）

Day01~15 - Python语言基础

Day01 - 初识Python

- Python简介 - Python的历史 / Python的优缺点 / Python的应用领域
- 搭建编程环境 - Windows环境 / Linux环境 / MacOS环境
- 从终端运行Python程序 - Hello, world / print函数 / 运行程序
- 使用IDLE - 交互式环境(REPL) / 编写多行代码 / 运行程序 / 退出IDLE
- 注释 - 注释的作用 / 单行注释 / 多行注释

Day02 - 语言元素

- 程序和进制 - 指令和程序 / 冯诺依曼机 / 二进制和十进制 / 八进制和十六进制
- 变量和类型 - 变量的命名 / 变量的使用 / input函数 / 检查变量类型 / 类型转换
- 数字和字符串 - 整数 / 浮点数 / 复数 / 字符串 / 字符串基本操作 / 字符编码
- 运算符 - 数学运算符 / 赋值运算符 / 比较运算符 / 逻辑运算符 / 身份运算符 / 运算符的优先级
- 应用案例 - 华氏温度转换成摄氏温度 / 输入圆的半径计算周长和面积 / 输入年份判断是不是闰年

Day03 - 分支结构

- 分支结构的应用场景 - 条件 / 缩进 / 代码块 / 流程图
- if语句 - 简单的if / if-else结构 / if-elif-else结构 / 嵌套的if
- 应用案例 - 用户身份验证 / 英制单位与公制单位互换 / 掷骰子决定做什么 / 百分制成绩转等级制 / 分段函数求值 / 输入三条边的长度如果能构成三角形就计算周长和面积

Day04 - 循环结构

- 循环结构的应用场景 - 条件 / 缩进 / 代码块 / 流程图
- while循环 - 基本结构 / break语句 / continue语句
- for循环 - 基本结构 / range类型 / 循环中的分支结构 / 嵌套的循环 / 提前结束程序
- 应用案例 - 1~100求和 / 判断素数 / 猜数字游戏 / 打印九九表 / 打印三角形图案 / 猴子吃桃 / 百钱百鸡

Day05 - 构造程序逻辑

- 经典案例：水仙花数 / 百钱百鸡 / Craps赌博游戏
- 练习题目：斐波那契数列 / 完美数 / 素数

Day06 - 函数和模块的使用

- 函数的作用 - 代码的坏味道 / 用函数封装功能模块
- 定义函数 - def语句 / 函数名 / 参数列表 / return语句 / 调用自定义函数
- 调用函数 - Python内置函数 / 导入模块和函数
- 函数的参数 - 默认参数 / 可变参数 / 关键字参数 / 命名关键字参数
- 函数的返回值 - 没有返回值 / 返回单个值 / 返回多个值
- 作用域问题 - 局部作用域 / 嵌套作用域 / 全局作用域 / 内置作用域 / 和作用域相关的关键字
- 用模块管理函数 - 模块的概念 / 用自定义模块管理函数 / 命名冲突的时候会怎样（同一个模块和不同的模块）

Day07 - 字符串和常用数据结构

- 字符串的使用 - 计算长度 / 下标运算 / 切片 / 常用方法
- 列表基本用法 - 定义列表 / 用下表访问元素 / 下标越界 / 添加元素 / 删除元素 / 修改元素 / 切片 / 循环遍历
- 列表常用操作 - 连接 / 复制(复制元素和复制数组) / 长度 / 排序 / 倒转 / 查找
- 生成列表 - 使用range创建数字列表 / 生成表达式 / 生成器
- 元组的使用 - 定义元组 / 使用元组中的值 / 修改元组变量 / 元组和列表转换
- 集合基本用法 - 集合和列表的区别 / 创建集合 / 添加元素 / 删除元素 / 清空
- 集合常用操作 - 交集 / 并集 / 差集 / 对称差 / 子集 / 超集
- 字典的基本用法 - 字典的特点 / 创建字典 / 添加元素 / 删除元素 / 取值 / 清空
- 字典常用操作 - keys()方法 / values()方法 / items()方法 / setdefault()方法
- 基础练习 - 跑马灯效果 / 列表找最大元素 / 统计考试成绩的平均分 / Fibonacci数列 / 杨辉三角
- 综合案例 - 双色球选号 / 井字棋

Day08 - 面向对象编程基础

- 类和对象 - 什么是类 / 什么是对象 / 面向对象其他相关概念
- 定义类 - 基本结构 / 属性和方法 / 构造器 / 析构器 / __str__方法
- 使用对象 - 创建对象 / 给对象发消息
- 面向对象的四大支柱 - 抽象 / 封装 / 继承 / 多态
- 基础练习 - 定义学生类 / 定义时钟类 / 定义图形类 / 定义汽车类

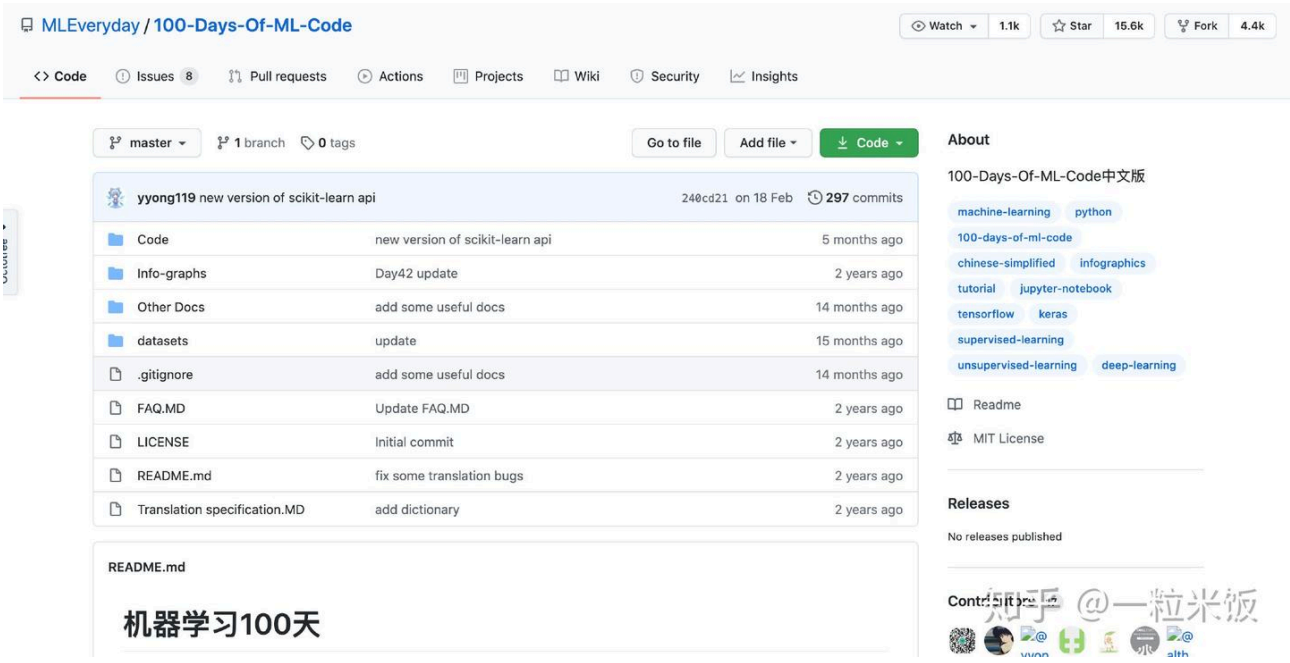
<<< 左右滑动见更多 >>>

2.机器学习100天

这个项目开始由Avik-Jain创建了一个英文版，然后由MLEveryday维护更新一个中文版。当前还没到完整的100天规划，但也足够作为参考学习了，当前目录如下：

有监督学习
- 数据预处理
- 简单线性回归
- 多元线性回归
- 逻辑回归
- k近邻法(k-NN)
- 支持向量机(SVM)
- 决策树
- 随机森林

无监督学习
- K-均值聚类
- 层次聚类



github:

github.com/MLEveryday/1

⭐ **star**: 15.6k

#100DaysOfMLCode
Day 1
©Avik Jain

数据预处理

机器学习初步

第1步：导入需要的库

这两个是我们每次都需要导入的库。
NumPy包含数学计算函数。
Pandas用于导入和管理数据集。

第2步：导入数据集

数据集通常是.csv格式。CSV文件以文本形式保存表格数据。文件的每一行是一条数据记录。我们使用Pandas的read_csv方法读取本地csv文件为一个数据帧。然后，从数据帧中制作自变量和因变量的矩阵和向量。

第3步：处理丢失数据

我们得到的数据很少是完整的。数据可能因为各种原因丢失，为了不降低机器学习模型的性能，需要处理数据。我们可以用整列的平均值或中间值替换丢失的数据。我们用sklearn.preprocessing库中的Imputer类完成这项任务。

第4步：解析分类数据

分类数据指的是含有标签值而不是数字值的变量。取值范围通常是固定的。例如“Yes”和“No”不能用于模型的数学计算，所以需要解析成数字。为实现这一功能，我们从sklearn.preprocesing库导入LabelEncoder类。

第5步：拆分数据集为测试集合和训练集合

把数据集拆分成两个：一个是用来训练模型的训练集合，另一个是用来验证模型的测试集合。两者比例一般是80:20。我们导入sklearn.cross-ssvalidation库中的train_test_split()方法。

第6步：特征缩放

大部分模型算法使用两点间的欧式距离表示，但此特征在幅度、单位和范围姿态问题上变化很大。在距离计算中，高幅度的特征比低幅度特征权重更大。可用特征标准化或Z值归一化解决。导入sklearn.preprocessing库的StandardScalar类。

Check out The complete Implementation at: github.com/Avik-Jain/100-Days-Of-ML-Code

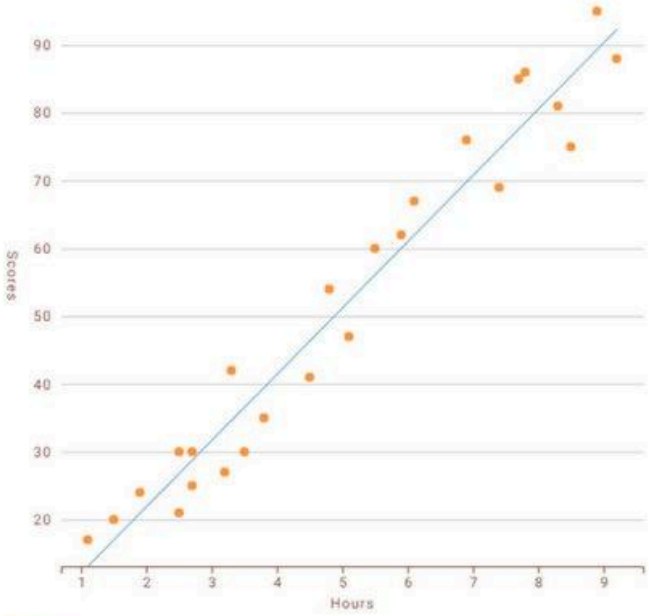
Follow Me For More Updates

机器学习 100 天
第二天
©Avik Jain

简单线性回归

使用单一特征来预测响应值

这是一种基于自变量值 (X) 来
预测因变量值 (Y) 的方法。
假设这两个变量是线性相关的。
因此，我们尝试寻找一种
根据特征或自变量 (x) 的线性函数
来精确预测响应值 (y)。



怎样找到最佳的拟合线

在这个回归任务中，我们将通过找到
“ 最佳拟合线 ” 来最小化预测误差
—回归线的误差将是最小的。
我们试图最小化观测值 (Yi) 和
模型预测值 (Yp) 之间的长度

预测值
实际值
$$y_i \quad y_p$$

$$\min\{\mathrm{SUM}(y_i - y_p)^2\}$$

因变量

$$y = b_0 + b_1 x_1$$

自变量

在这个回归任务中，我们将预测
一个学生根据所学习的小时数来
计算分数的百分比。

斜率

$$Score = b_0 + b_1 * hours$$

y• 截距

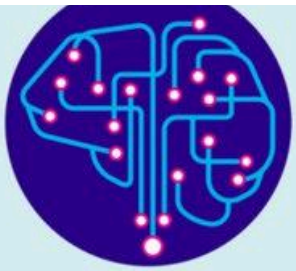
步骤 1：数据预处理

我们将按照之前的数据预处理信息图表那样来执行相同的步骤
•导入相关库
•导入数据集
•检查缺失数据
•划分数据集
•特征缩放我们将使用简单线性模型的相关库来进行

步骤 2：通过训练集来训练简单线性回归模型

为了使用模型来训练数据集，

我们将使用来自 **sklearn.linear_model** 库的
LinearRegression 类
然后我们创建一个 **LinearRegression** 类的 **regressor** 对象
最后我们将使用 **LinearRegression** 类的 **fit()** 方法
将 **regressor** 对象对数据集进行训练。



步骤 3：预测结果

现在我们将预测来自测试集的观察结果。
我们将把输出保存在向量 **Y_pred** 中。
我们使用前一步中训练的回归模型 **regressor**
的 **LinearRegression** 类的预测方法
来对结果进行预测。

步骤 4：可视化

最后一步是将结果可视化，我们将
使用 **matplotlib.pyplot** 库对
我们的训练集结果和测试集结果做散点图，
以查看我们的模型预测效果。

Check out The complete Implementation at: github.com/Avik-Jain/100-Days-Of-ML-Code

Follow Me For More Updates

#100DaysOfMLCode

第三天

©Avik Jain

多元线性回归

多元线性回归尝试通过用一个线性方程来适配观测数据，这个线性方程是在两个以上（包括两个）的特征和响应之间构建的一个关系。多元线性回归的实现步骤和简单线性回归很相似，在评价部分有所不同。你可以用它来找出在预测结果上哪个因素影响力最大，以及不同变量是如何相互关联的。

Dependent variable

multiple independent variables

$$y = b_0 + b_1 x_1 + b_2 x_2 \ldots\ldots b_n x_n$$

前提

想要有一个成功的回归分析，
确认这些假定很重要

1、线性：自变量和因变量的关系应该是线性的（也即特征值和预测值是线性相关）

2、保持误差项的方差齐性（常数方差）：误差项的分散（方差）必须等同

3、多元正态分布：多元回归假定残差符合正态分布。

4、缺少多重共线性：假设数据有极少甚至没有多重共线性。当特征（或自变量）不是相互独立时，会引发多重共线性。

注意

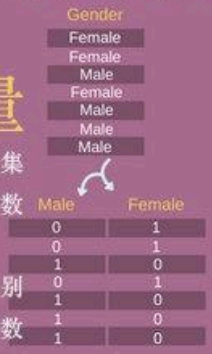
过多的变量可能会降低模型的精确度，尤其是如果存在一些对结果无关的变量，或者存在对其他变量造成很大影响的变量时。这里介绍一些选择合适变量的方法：

1、向前选择法

2、向后选择法（也称 向后剔除法 / 向后消元法）

3、向前向后法：即结合了上面说的向前法和向后法，先用

虚（拟）变量



在多元回归模型中，当遇到数据集是非数值数据类型时，使用分类数据是一个非常有效的方法。

分类数据，是指反映（事物）类别的数据，是离散数据，其数值个数（分类属性）有限（但可能很多）且值之间无序。比如，按性别分为男、女两类。在一个回归模型中，这些分类值可以用虚变量来表示，变量通常取诸如 1 或 0 这样的值，来表示肯定类型或否定类型。

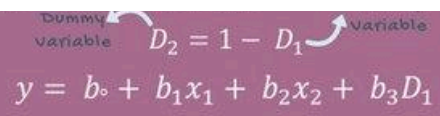
虚拟变量陷阱

虚拟变量陷阱是指两个以上（包括两个）变量之间高度相关的情形。简而言之，就是存在一个能够被其他变量预测出的变量。我们举一个存在重复类别（变量）的直观例子：假使我们舍弃男性类别，那么，该类别也可以通过女性类别来定义（女性值为 0 时，表示男性，为 1，表示女性），反之亦然。

解决虚拟变量陷阱的方法是，类别变量减去一：假如有 m 个类别，那么在模型构建时取 m-1 个虚拟变量，减去的那个变量可以看作是参照值。

Dummy

向前法筛选一遍，再用向后法筛选一遍，直到最后无论怎么筛选模型变量都不再发生变化，就算是结束了



1 数据预处理

1. 导入相关库
2. 导入数据集
3. 检查缺失数据
4. 数据分类
5. 有必要的话，编辑虚拟变量并注意避免虚拟变量陷阱
6. 特征缩放我们将用简单线性回归模型的相关库来做

2 在训练集上训练模型

这一步和简单线性回归模型的处理完全一样。
我们用 sklearn.linear_model 库的 LinearRegression 类，在数据集上训练模型。首先，创建一个 LinearRegression 的对象 regressor，接着，用 LinearRegression 类的 fit() 方法，用对象 regressor 在数据集上进行训练。

3 预测结果

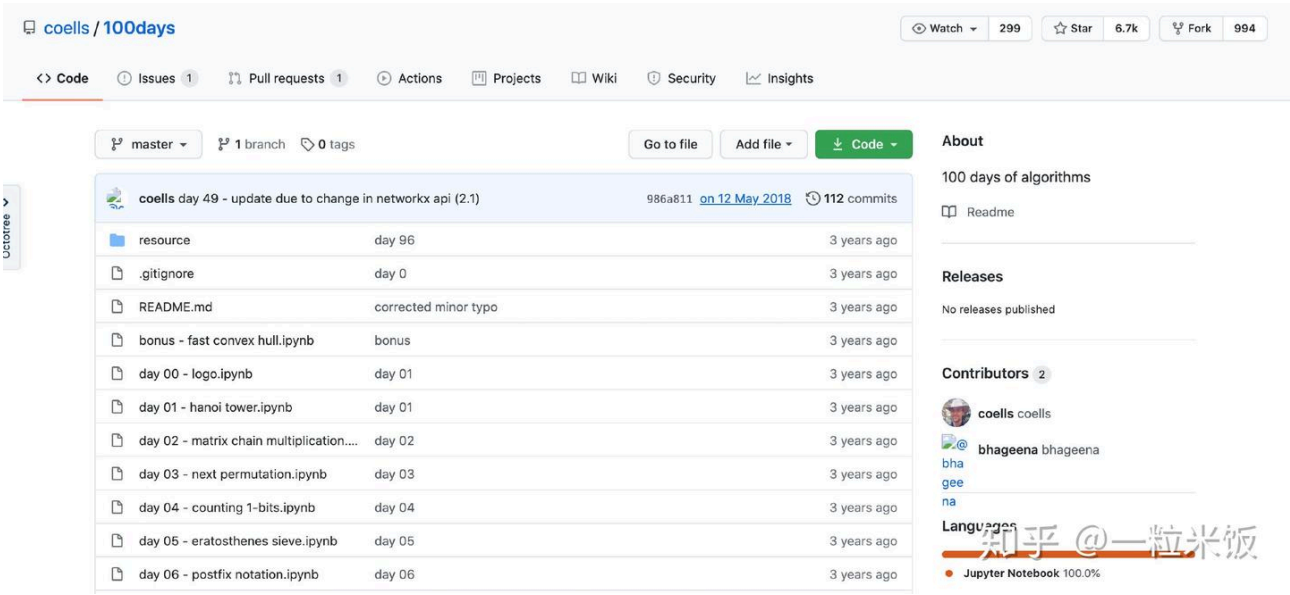
在测试集上进行预测，并观察结果。我们将把输出结果保存在向量 Y_pred 中。使用上一步训练时我们用的类 LinearRegression 的对象 regressor，在训练完之后，用其 predict() 方法来预测结果。

Check out The complete Implementation at: github.com/Avik-Jain/100-Days-Of-ML-Code
Follow Me For More Updates

<<< 左右滑动见更多 >>>

3.100 days of algorithms（英文）

算法学习100天，基于Jupyter+开发学习。作者为了提高自己的算法能力设置的100天挑战，这些挑战很有趣，但实现不是一件轻松的事情，感兴趣的挑战一下吧



github:

github.com/coells/100da

⭐ **star**:6.7k

algorithm

```
In [1]: def mult(chain):
            n = len(chain)

            # single matrix chain has zero cost
            aux = {(i, i): (0,) + chain[i] for i in range(n)}

            # i: length of subchain
            for i in range(1, n):
                # j: starting index of subchain
                for j in range(0, n - i):
                    best = float('inf')

                    # k: splitting point of subchain
                    for k in range(j, j + i):
                        # multiply subchains at splitting point
                        lcost, lname, lrow, lcol = aux[j, k]
                        rcost, rname, rrow, rcol = aux[k + 1, j + i]
                        cost = lcost + rcost + lrow * lcol * rcol
                        var = '(%s%s)' % (lname, rname)

                        # pick the best one
                        if cost < best:
                            best = cost
                            aux[j, j + i] = cost, var, lrow, rcol

            return dict(zip(['cost', 'order', 'rows', 'cols'], aux[0, n - 1]))
```

run

```
In [2]: mult([('A', 10, 20), ('B', 20, 30), ('C', 30, 40)])
Out[2]: {'cols': 40, 'cost': 18000, 'order': '((AB)C)', 'rows': 10}
```


algorithm

```
In [1]: def permute(values):
            n = len(values)

            # i: position of pivot
            for i in reversed(range(n - 1)):
                if values[i] < values[i + 1]:
                    break
            else:
                # very last permutation
                values[:] = reversed(values[:])
                return values

            # j: position of the next candidate
            for j in reversed(range(i, n)):
                if values[i] < values[j]:
                    # swap pivot and reverse the tail
                    values[i], values[j] = values[j], values[i]
                    values[i + 1:] = reversed(values[i + 1:])
                    break

            return values
```

run

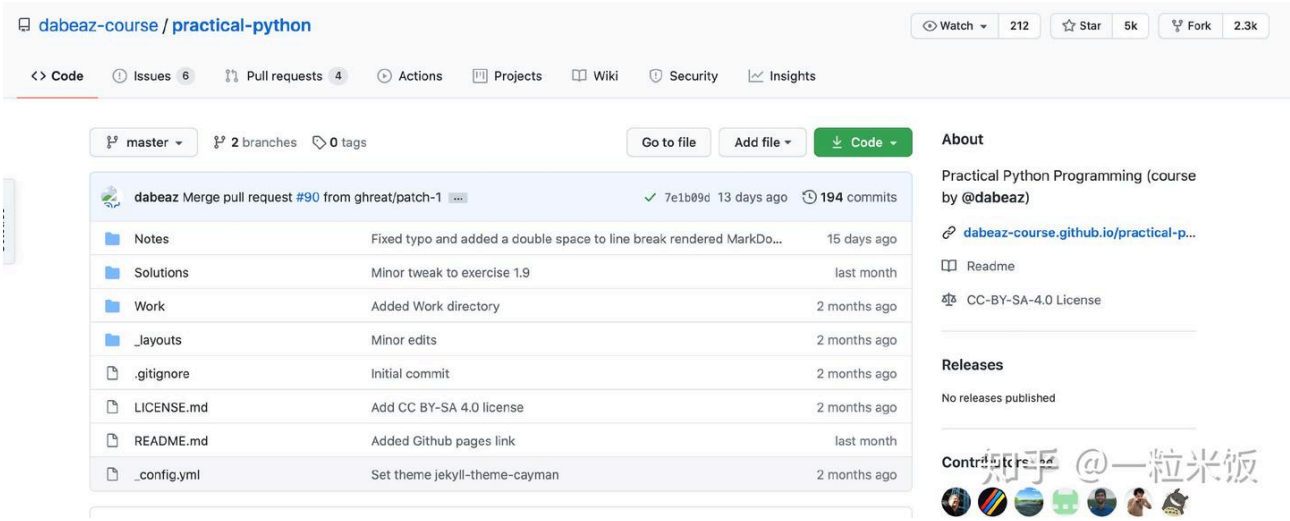
```
In [2]: x = [4, 3, 2, 1]
        for i in range(25):
            print(permute(x))
```

```
[1, 2, 3, 4]
[1, 2, 4, 3]
[1, 3, 2, 4]
[1, 3, 4, 2]
[1, 4, 2, 3]
[1, 4, 3, 2]
[2, 1, 3, 4]
[2, 1, 4, 3]
[2, 3, 1, 4]
```


<<< 左右滑动见更多 >>>

4. Practical Python（英文）

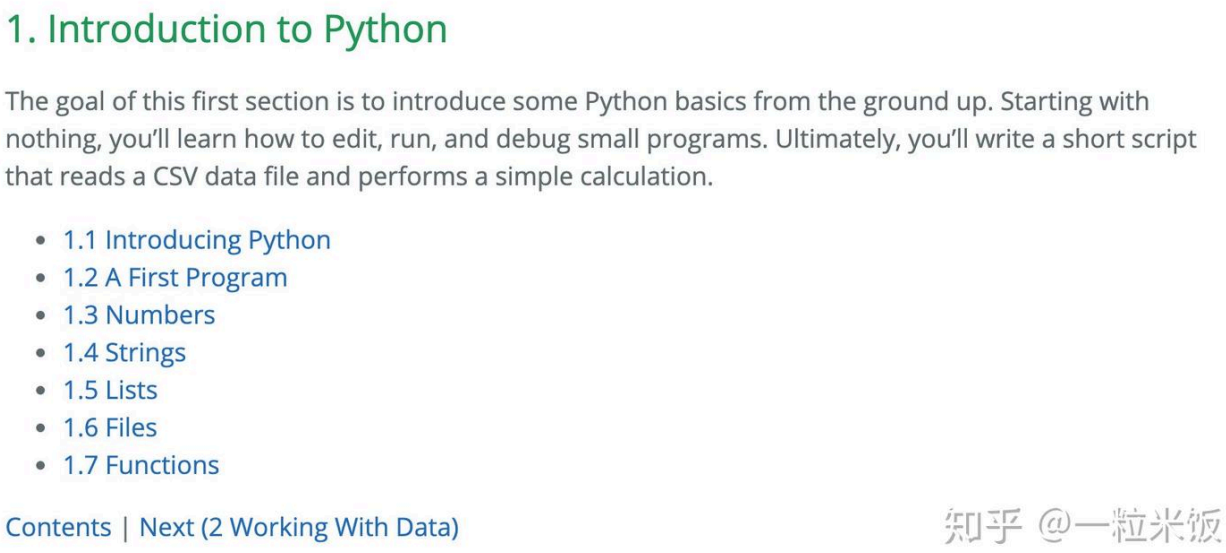
一位25年Python编程经验✦的大神分享的Python学习资料。这些学习经验经过实践验证并一直在不断发展，这个项目中包含大约130个动手编码练习。



github:

github.com/dabeaz-cours

⭐ **star**:5k



2. Working With Data

To write useful programs, you need to be able to work with data. This section introduces Python's core data structures of tuples, lists, sets, and dictionaries and discusses common data handling idioms. The last part of this section dives a little deeper into Python's underlying object model.

- 2.1 Datatypes and Data Structures
- 2.2 Containers
- 2.3 Formatted Output
- 2.4 Sequences
- 2.5 Collections module
- 2.6 List comprehensions
- 2.7 Object model

Contents | Prev (1 Introduction to Python) | Next (3 Program Organization)

3. Program Organization

So far, we've learned some Python basics and have written some short scripts. However, as you start to write larger programs, you'll want to get organized. This section dives into greater details on writing functions, handling errors, and introduces modules. By the end you should be able to write programs that are subdivided into functions across multiple files. We'll also give some useful code templates for writing more useful scripts.

- 3.1 Functions and Script Writing
- 3.2 More Detail on Functions
- 3.3 Exception Handling
- 3.4 Modules
- 3.5 Main module
- 3.6 Design Discussion about Embracing Flexibilty

Contents | Prev (2 Working With Data) | Next (4 Classes and Objects)

<<< 左右滑动见更多 >>>

5.Useful Python snippets（英文）

这个项目的目的是收集有用的Python代码段，以增强pythoneers的编码体验，可分为基础和进阶两部分。

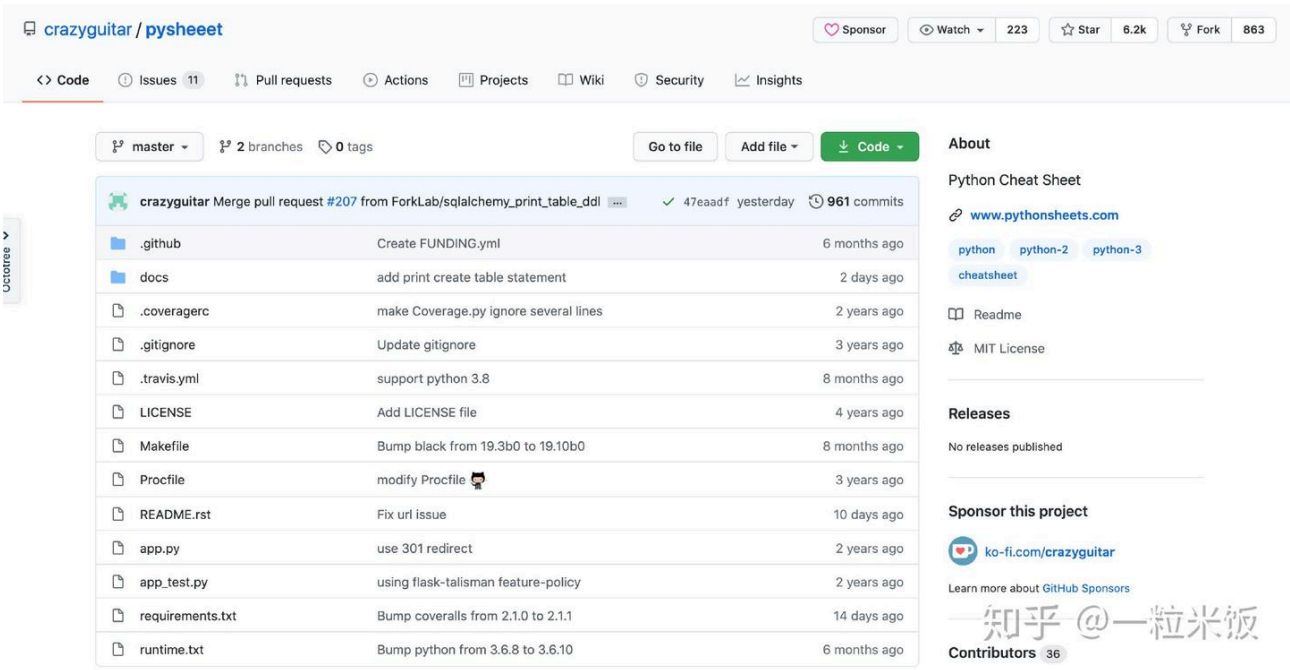
基础部分如下：

Style
From Scratch
Future
Unicode
List

Set
Dictionary
Function
Classes and Objects
Generator
Typing
Files and I/O

高级部分如下：

Regular Expression
Socket
Asyncio
Concurrency
SQLAlchemy
Security
Secure Shell
Boto3
Test
C Extensions



github:

github.com/crazyguitar/

⭐ **star**: 6.2k

Table of Contents

Naming

Class

Bad

```
class fooClass: ...
class foo_class: ...
```

Good

```
class FooClass: ...
```

Function

Bad

```
def CapCamelCase(*a): ...
def mixCamelCase(*a): ...
```


Good

Hello world!

When we start to learn a new language, we usually learn from printing **Hello world!**. In Python, we can use another way to print the message by importing `__hello__` module. The source code can be found on frozen.c.

```
>>> print("Hello world!")
Hello world!
>>> import __hello__
Hello world!
>>> import __phello__
Hello world!
>>> import __phello__.spam
Hello world!
```

Python Version

It is important for a programmer to know current Python version because not every syntax will work in the current version. In this case, we can get the Python version by `python -V` or using the module, `sys`.

```
>>> import sys
>>> print(sys.version)
3.7.1 (default, Nov  6 2018, 18:46:03)
[Clang 10.0.0 (clang-1000.11.45.5)]
```

We can also use `platform.python_version` to get Python version.

```
>>> import platform
>>> platform.python_version()
'3.7.1'
```

Dictionary

Get All Keys

```
>>> a = {"1":1, "2":2, "3":3}
>>> b = {"2":2, "3":3, "4":4}
>>> a.keys()
['1', '3', '2']
```

Get Key and Value

```
>>> a = {"1":1, "2":2, "3":3}
>>> a.items()
```

Find Same Keys

```
>>> a = {"1":1, "2":2, "3":3}
>>> b = {"2":2, "3":3, "4":4}
>>> [_ for _ in a.keys() if _ in b.keys()]
['3', '2']
>>> # better way
>>> c = set(a).intersection(set(b))
>>> list(c)
['3', '2']
>>> # or
>>> [_ for _ in a if _ in b]
['3', '2']
[('1', 1), ('3', 3), ('2', 2)]
```


<<< 左右滑动见更多 >>>

希望对正在学习Python的小哥哥小姐姐有所帮助，也欢迎交流补充~

所属专栏 · 2022-03-06 10:27 更新

编程学习路线与面试资源

阿良

5 篇内容 · 1172 赞同

订阅

最热内容 · Github大神的Golang学习资源

发布于 2020-07-23 08:41

Python　机器学习　项目开发

农夫马

谢谢分享，入坑学习中

2024-07-02 回复 3

xiadjs

123

06-16 回复 喜欢

牛栓柱

py.qizhen.xyz/

2024-11-05 回复 1

虫鸣

好东西，到我收藏夹里吃东西去吧！！！

2021-04-30 回复 2

Beloved

感谢

07-17 回复 喜欢

舍命压制油饼

每次想学学一点就因为各种事中断了，希望这次能坚持下去

07-05 回复 喜欢

xiadjs

123

06-16 回复 喜欢

知也无涯

好😚

2024-11-09 回复 喜欢

一二南

那个github怎么登啊

2024-09-26 回复 喜欢

坚持长跑

收一份吃灰。

2023-12-12 回复 喜欢

启智

不会用GitHub

2023-09-21 回复 喜欢

点击查看全部评论 ›

理性发言，友善互动