

# 大数据导论：项目二

11711217, Iydon Leong

2019 年 5 月 28 日

## 目录

<b>0 文档与代码的规范</b>	<b>3</b>
0.1 文档的规范与约定 . . . . .	3
0.2 代码的规范与约定 . . . . .	3
<b>1 共享单车的背景与意义</b>	<b>3</b>
1.1 共享单车的背景 . . . . .	3
1.2 共享单车数据的意义 . . . . .	4
<b>2 数据探索与可视化</b>	<b>5</b>
2.1 数据探索 . . . . .	5
2.2 数据可视化 . . . . .	6
2.2.1 训练集 . . . . .	8
2.2.2 测试集 . . . . .	8
2.2.3 时间相关变量数据集 . . . . .	8
<b>3 数据预处理</b>	<b>12</b>
3.1 数据清洗 . . . . .	12
3.2 数据变换 . . . . .	12
3.3 数据简化 . . . . .	13
<b>4 回归模型的构造</b>	<b>14</b>
4.1 评价与误差函数 . . . . .	14
4.2 回归模型的选择 . . . . .	14

目录	2
5 未来工作	16
6 结论	16
7 参考文献	18
A 模块之间依赖关系	19
B 运行结果数据结构	19
C Python 代码附录	20
C.1 第三方库依赖 . . . . .	20
C.2 主文件 . . . . .	20
C.3 分析模块 . . . . .	22
C.4 数据模块 . . . . .	26
C.5 可视化模块 . . . . .	28
C.6 回归模型模块 . . . . .	30
C.7 配置文件模块 . . . . .	30
C.8 彩色字符串模块 . . . . .	31
C.9 调试模块 . . . . .	33
C.10 进度条模块 . . . . .	33
C.11 计时模块 . . . . .	33
C.12 消除警告模块 . . . . .	34

# 0 文档与代码的规范

## 0.1 文档的规范与约定

- `file`: 标志文件 `file`;
- `code`: 标志 Python 中的行内代码片段 `code`;
- `attribute`: 标志数据集中的属性 `attribute`。

## 0.2 代码的规范与约定

项目代码遵循PEP 8规范。但是会根据自己的风格对原规范进行适当的修改与拓展，即保持代码风格的统一性。

# 1 共享单车的背景与意义

## 1.1 共享单车的背景

自 2006 年出现首家专业租车网站以来，中国在线出行服务行业经历了“线下重资产 + 线上服务”向“互联网 + 共享经济/轻资产重服务”的转变，同时也实现了 PC 端向移动端使用场景的转变；2012 年起移动端出行服务模式大量涌现，主流服务包括租车、拼车、代驾、出租车、专车（快车）与分时租赁等，其中，专/快车、共享单车等共享出行服务成为其主流服务。陈正豪 2017 从三个方面看一看共享单车的行业背景可以将在线出行服务发展历程归结为三个转变：

1. 运营模式：从“线下重资产 + 线上服务”转向“互联网 + 共享经济/轻资产重服务”；
2. 使用场景：经历了 PC 端到移动端的转变，这和智能手机的普及和渗透是离不开的；
3. 量级：2012 年起移动端出行服务模式大量涌现。

共享单车的发展离不开公共自行车管理系统的发展，从 1965 年第一代公共自行车系统在荷兰问世，该系统采用完全免费，不上锁且无人监管，市民可随意取用，但无法解决自行车偷盗损毁问题；在这之后的 1995 年第二代公共自行车系统又出现了：固定存取地点，投币开锁，但管理难度较大；再到 20 世纪 90 年代末第三代基于信息技术与会员制管理，这一模式成为了当前政府的一个主流模式；终于在 2010 年，美国 Social Bicycles 创意性地提出了借助手机 APP 和 GPS 定位，快速租用归还的无桩模式运营，这便是共享单车的雏形。陈正豪 2017 从三个方面看一看共享单车的行业背景

作为在线出行行业的主流服务，共享出行包含专车、快车等网约车服务，分时租赁服务，以及 2016 年火爆市场的共享单车服务；相比网约车的专/快车及顺风车，共享单车解决了用户

“最后一公里”的出行问题，节约用户等车的时间成本及服务的费用成本；相比分时租赁，共享单车使用方便，取还车灵活，使用性价比高，目前共享单车用户覆盖率增长迅速，远超分时租赁用户群体<sup>[3]</sup>。  
陈正豪 2017 从三个方面看一看共享单车的行业背景，李琨浩 2017 基于共享经济视角下城市共享单车发展对策研究

采用 PEST 分析法可知，共享单车的发展环境主要得益于政府、经济、社会和技术四方面<sup>[4]</sup>：

1. 政府层面：强调改善出行环境，规范出行规则，鼓励资本合作，推动智能创造；
2. 经济层面：整体经济呈温和上升趋势，智能手机渗透率增长迅速；
3. 社会层面：空气污染严重，城市交通拥堵等出行问题亟待解决；
4. 技术层面：智能手机与 4G 网络的普及，物联网技术发展，为共享单车的普及搭建了良好的平台。

我们可以对共享单车的用户画像进行分析：

1. 性别男性用户偏多，摩拜男性用户更多；
2. 摩拜和 OFO 以年轻用户为主，相对来看永安行用户年龄偏大，主要还是以年轻用户为主。
3. 新兴共享单车以一线城市用户为主，市政单车在二三线渗透更深。
4. 用户设备以华为、三星等为主，摩拜用户相对高端。

最后我们总结出这样一个用户画像（如图 1）：<sup>F:user-portrait</sup>共享单车主要用户为中高等收入且追求健康生活方式的中青年男性<sup>[5]</sup>。  
陈正豪 2017 从三个方面看一看共享单车的行业背景

## 1.2 共享单车数据的意义

除了共享单车系统在现实世界中的有趣应用外，由系统产生的数据也吸引着研究人员从事相关方面的研究。共享单车在使用时间与起止位置上都是精确记录的，在这一点上不像是诸如公共汽车、地铁等的交通，而这一特性使得共享单车系统可以在一定程度上作为虚拟传感器网络，可用于感知城市范围内的转移。因此，通过对数据的探索从而预测城市内大多数的重要事件。  
范东旭刘威 2018 沈阳市共享单车大数据分析与发展对策研究,data



图 1: 共享单车用户画像

## 2 数据探索与可视化

### 2.1 数据探索

数据集来自capitalbikeshare<sup>data</sup>。hour.csv 数据集有 17379 条数据且无缺失，总共记录了 17379 小时；day.csv 数据集有 731 条数据且无缺失，总共记录了两年的单车使用状况。除了

---

维度不存在于 day.csv 外，两个数据集都有以下的维度：

- instant: 记录的索引
- dteday: 日期
- season: 季节（1 至 4 分别代表春夏秋冬）
- yr: 年（0 至 1 分别代表 2011 年与 2012 年）
- mnth: 月份（1 至 12）
- hr: 小时（0 至 23）
- holiday: 是否为节假日（根据dc.gov网站）
- weekday: 该日是一周的第几天
- workingday: 是否为工作日

- weathersit: 天气情况 (1 至 4 分别代表晴天、多云、小雨、暴雨)
- temp: 标准化的摄氏度下温度 (除以最大值 41)
- atemp: 标准化的摄氏度下的体感温度 (除以最大值 50)
- hum: 标准化的湿度 (除以最大值 100)
- windspeed: 标准化的风速 (除以最大值 67)
- casual: 未注册用户的数量
- registered: 注册用户的数量
- cnt: 注册与未注册用户的总数量

从直观来说，共享单车的使用过程与周围的环境性与季节性的变化密切相关，例如天气的状态、降雨量、季节、一周内相对时间、一天内相对时间等。于是我们可以得到下述假设：

- 天气晴朗适宜单车出行而阴雨不适宜；
- 温度舒适适宜单车出行而过低过高不适宜；
- 位于工作日由于上下班需求单车使用率高而周末出门游玩单车使用率可能也较高；
- 一天内上下班高峰期与中午时间单车使用率高而其他时间使用率相对较低。

因此我们使用时间相关的变量去预测注册用户与未注册用户的数量，同时使用环境相关的变量去预测注册用户与未注册用户的数量。我们可以将此问题看作回归问题，同时构造回归模型。

首先我们将数据集分为训练集与测试集，以此判断回归模型的好坏与避免模型的欠拟合与过拟合。按照规则，我们提取出每月前 19 天作为训练集，其余天数作为测试集。将除 casual、registered 与 cnt 这三列的数据视做变量向量  $\mathbf{X}$ ，将 cnt 列视做响应向量  $\mathbf{Y}$ 。

其次我们计算出数据的描述性统计量，以此大致判断数据集的分布及形状。由表 T:train-day-x-data 与 T:train-day-y-data 可以得出训练集的个数、均值、方差、最小最大值与分位数。部分列向量已经过归一化处理，不难看出数据基本不符合正态分布，同时由 scipy.stats.kstest 计算可得出上述结论。

## 2.2 数据可视化

接下来我们对数据进行可视化处理，以期得到更加直观的结论，并且简要验证上述的假设。数据由两个数据集构成，我们分别将数据集分为训练集与测试集，同时衍生出时间相关变量数据集。

	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed
count	456	-	-	-	-	-	-	-	-	-	-
mean	2.50	0.5	6.50	0.03	3.00	0.68	1.39	0.49	0.47	0.62	0.19
std	1.12	0.5	3.46	0.17	2.01	0.47	0.54	0.18	0.16	0.14	0.08
min	1.00	0.0	1.00	0.00	0.00	0.00	1.00	0.11	0.10	0.00	0.02
25%	1.75	0.0	3.75	0.00	1.00	0.00	1.00	0.34	0.34	0.51	0.14
50%	2.50	0.5	6.50	0.00	3.00	1.00	1.00	0.50	0.49	0.62	0.18
75%	3.25	1.0	9.25	0.00	5.00	1.00	2.00	0.65	0.60	0.72	0.23
max	4.00	1.0	12.00	1.00	6.00	1.00	3.00	0.86	0.80	0.97	0.51

表 1: `day.csv` 训练集变量数据探索

	casual	registered	cnt
count	456	-	-
mean	859.95	3713.47	4573.41
std	698.91	1494.48	1868.74
min	9.00	491.00	605.00
25%	318.00	2696.00	3305.50
50%	722.00	3700.00	4585.50
75%	1141.75	4814.25	5987.50
max	3410.00	6911.00	8714.00

表 2: `day.csv` 训练集相应数据探索

	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed
count	10886	-	-	-	-	-	-	-	-	-	-	-
mean	2.51	0.5	6.52	11.54	0.03	3.00	0.68	1.42	0.49	0.47	0.62	0.19
std	1.12	0.5	3.44	6.92	0.17	2.01	0.47	0.63	0.19	0.17	0.19	0.12
min	1.00	0.0	1.00	0.00	0.00	0.00	0.00	1.00	0.02	0.02	0.00	0.00
25%	2.00	0.0	4.00	6.00	0.00	1.00	0.00	1.00	0.34	0.33	0.47	0.10
50%	3.00	1.0	7.00	12.00	0.00	3.00	1.00	1.00	0.50	0.48	0.62	0.19
75%	4.00	1.0	10.00	18.00	0.00	5.00	1.00	2.00	0.64	0.62	0.77	0.25
max	4.00	1.0	12.00	23.00	1.00	6.00	1.00	4.00	1.00	0.91	1.00	0.85

表 3: `hour.csv` 训练集变量数据探索

	casual	registered	cnt
count	10886	-	-
mean	36.02	155.55	191.57
std	49.96	151.04	181.14
min	0.00	0.00	1.00
25%	4.00	36.00	42.00
50%	17.00	118.00	145.00
75%	49.00	222.00	284.00
max	367.00	886.00	977.00

表 4: `hour.csv` 训练集相应数据探索

train-day\_x-data

train-day\_y-data

train-hour\_x-data

train-hour\_y-data

### 2.2.1 训练集

如图 2a, `day.csv` 训练集中响应（注册用户数目、未注册用户数目与总用户数目）趋势基本一致，并且与温度、体感温度的趋势趋于一致，而湿度与风速则直观感觉不到相似的趋势。

如图 2b, `hour.csv` 训练集的可视化中记录了一周内 24 小时的用户数量。可以看出响应（注册用户数目、未注册用户数目与总用户数目）趋势基本一致，类似于周期函数，且一共有 14 个峰值，分别出现在早高峰与晚高峰时段，而工作日（周一至周五）的用户数量是多于周末（周六与周日）的。直观来说，不同时间段用户数量与温度、体感温度、湿度、风速均有一定的联系，不好判断哪一个变量为主要变量，于是就需要训练回归模型预测。

### 2.2.2 测试集

如图 3a 与图 3b 可知，按照天数来说，用户数量的趋势与温度较为类似，而在小时尺度下则呈现不出明显的规律，有待于进一步的研究与建模。

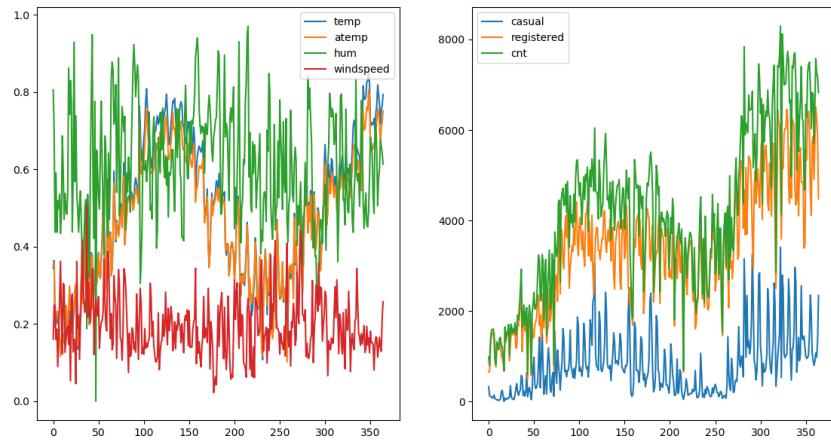
### 2.2.3 时间相关变量数据集

时间相关变量的训练集与测试集的数据趋势相差不大，因此我们只讨论训练集的数据可视化结果，如图 4。

由图 4a 可知， 不同颜色的线条对应着不同的时间点，而图中颜色呈现分层的效果，所以说用户数量与时间点之间存在较强的相关性。由图 4b 与图 4c 可知，2011 年与 2012 年的数据呈现相似性，即大体趋势基本一致而对应月份的用户数量大致呈现倍数增加的关系。同时夏秋季的用户数量多于春冬季节，符合温度对用户数量存在影响的假设。最后由图 4d 可知，用户数量多的时间绝大部分为工作日，只有少数几天例外。

F:train-day

(a) day.csv 训练集数据可视化

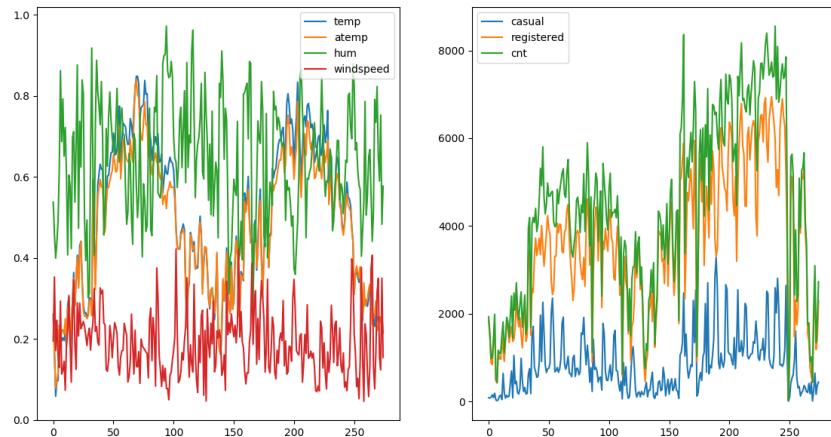


F:train-hour

(b) hour.csv 训练集数据可视化

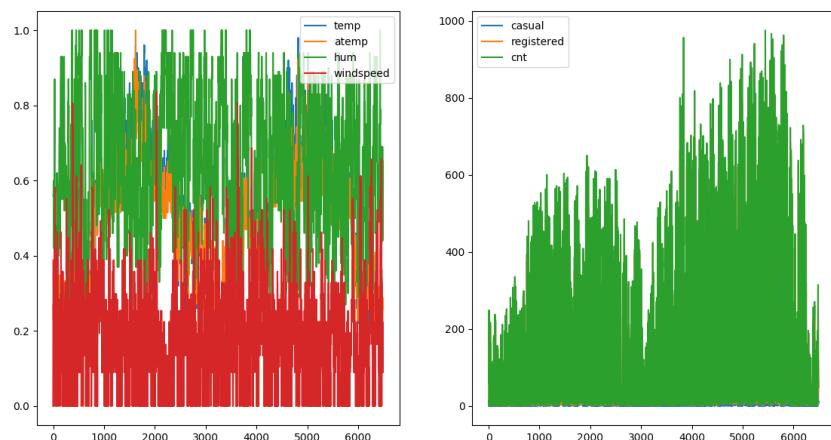
F:train

图 2: 训练集数据可视化



F:test-day

(a) day.csv 测试集数据可视化



F:test-hour

(b) hour.csv 测试集数据可视化

F:test

图 3: 测试集数据可视化

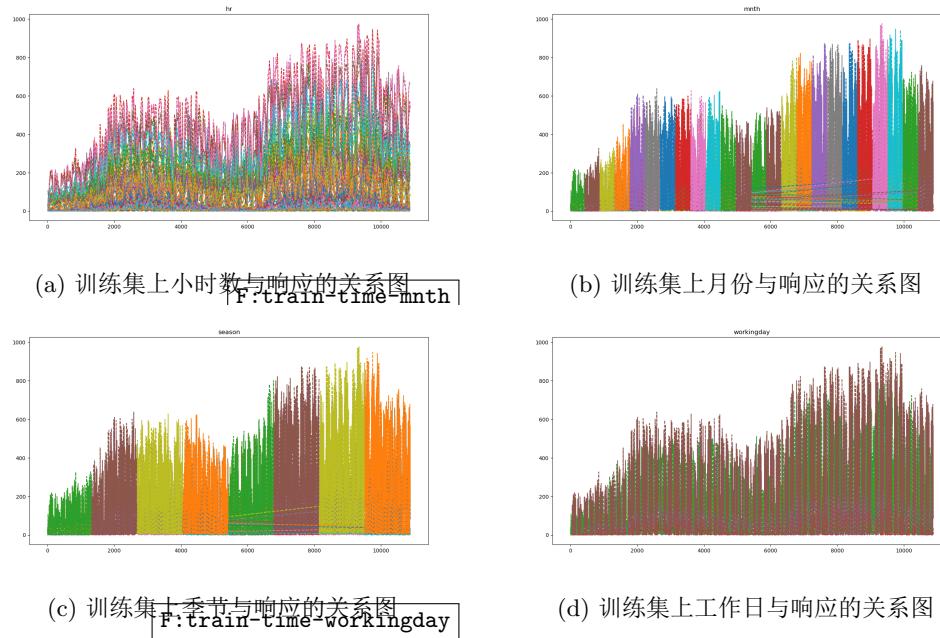


图 4: 训练集上时间有关变量与响应的关系图

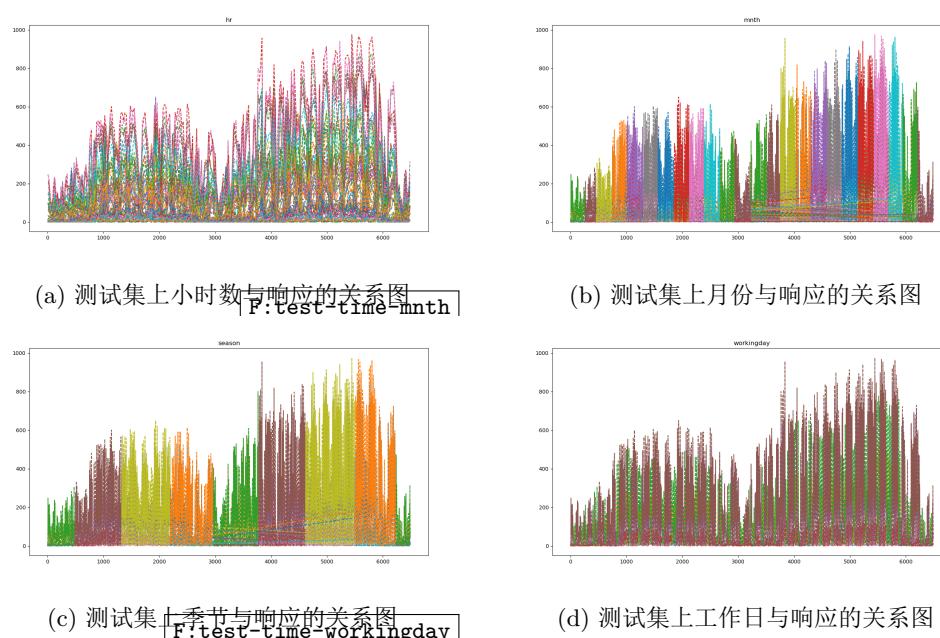


图 5: 测试集上时间有关变量与响应的关系图

### 3 数据预处理

数据往往因缺乏统一标准与结构导致杂乱性，因对客观事物在数据库中存在两个以上的描述导致重复性，因信息模糊、随机与缺失导致不完整性。因此为增加模型性能，简化模型计算，  
我们需要对数据进行预处理。  
撒味大白菜 2018 机器学习笔记：特征工程与数据降维，刘明吉 2000 数据挖掘中的数据预处理

#### 3.1 数据清洗

数据清洗主要包括如下部分：

P: 缺失值

1. 缺失值处理；

P: 噪声

2. 噪声数据处理；

P: 异常值

3. 异常值处理；

P: 脏数据

4. 脏数据处理；

P: 去重

5. 去重处理。

根据数据集的探索我们可以得出，原数据不存在缺失值，并且结构完善与统一，无重复数据，所以我们只需要进行第 2 部分与第 3 部分的数据清洗即可。

通过主观判断与可视化判断，数据虽然存在被随机误差予离群值，但是数据取值范围正常却数量远小于数据总量，所以本文并未对数据进行严格的噪声数据处理与异常值数据处理，这将补充到第 5 节的未来工作当中。

#### 3.2 数据变换

数据变换一般包括如下部分：

1. 平滑：去掉数据中的噪音。
2. 聚集：为多粒度数据分析构造数据。
3. 数据泛化：用高层次概念代替低层次原始数据。
4. 规范化：将属性数据按比例缩放，使之落入一个特定区间。
5. 属性构造：构造新的属性并添加到属性集中。

根据数据集的探索我们可以得出，原数据已经进行了数据泛化与规范化处理，所以本文并未对数据进行数据泛化、规范化等数据变换过程，这将补充到第 5 节的未来工作当中。

### 3.3 数据简化

数据简化，又称数据规约，指在尽可能保持数据原貌的前提下，最大限度地精简数据量。  
同时，用于数据简化的时间不应当超过之后在数据挖掘中节省的时间<sup>[10]</sup>。数据简化有以下三种常见方法：

1. 维度规约；
2. 数量规约；
3. 数据压缩。

本文主要是用维度规约，从高维特征空间向低维特征空间进行映射，即降维。一般来说，现实生活中所得到的数据具备共线性、稀疏性等特征，因此对数据进行降维有助于减少特征属性之间的个数，同时确保特征属性之间是相互独立的。降维主要有以下方法（均来自 `scikit-learn`, `sklearn_api`, `DataAnalyst2019` 机器学习之 `sklearn` 中的降维算法, `Cyrille2015sne` 模块）<sup>[11]</sup>：

- 主成分分析：
  - `PCA`;
  - `IncrementalPCA`;
  - `KernelPCA`;
  - `MiniBatchSparsePCA`;
  - `SparsePCA`;
  - `TruncatedSVD`.
- 因子分析：
  - `FactorAnalysis`.
- 独立成分分析：
  - `FastICA`.
- 字典学习：
  - `DictionaryLearning`;
  - `MiniBatchDictionaryLearning`.
- 高级矩阵分解：

- LatentDirichletAllocation;
- NMF;
- SparseCoder.

本文主要采用PCA与TruncatedSVD,通过实际的计算发现,选取阈值为95%以上的TruncatedSVD降维方法所得的效果最好,即 $R^2$ 相对较高。所以在接下来的构造回归模型中,我们采用该降维方法对数据进行处理。

## 4 回归模型的构造

### 4.1 评价与误差函数

我们选择以下评价函数与误差函数来评定模型的优劣(均来自于sklearn.metrics模块):

- 评价函数:

- explained\_variance\_score;
- r2\_score.

- 误差函数:

- max\_error;
- mean\_absolute\_error;
- mean\_squared\_error;
- mean\_squared\_log\_error;
- median\_absolute\_error.

广泛地对模型进行评价,同时加上模型的训练时间与预测时间,多维度地评价模型的优劣。

### 4.2 回归模型的选择

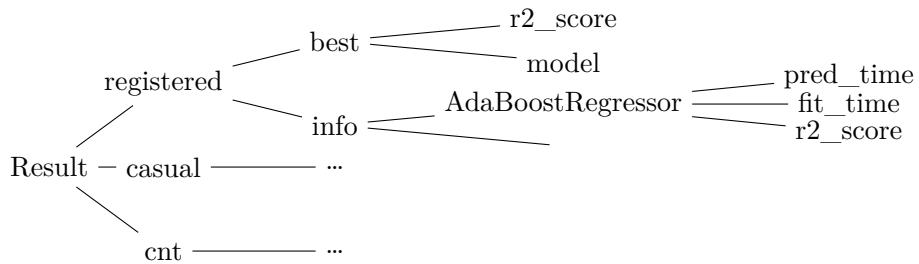
广泛地训练模型,从而取效果最好的,因此我们采用以下的回归模型scikit-learn,sklearn\_api:

- sklearn.ensemble:

- AdaBoostRegressor;
- BaggingRegressor;

- `ExtraTreesRegressor`;
- `GradientBoostingRegressor`;
- `IsolationForest`.
- `sklearn.gaussian_process`:
  - `GaussianProcessRegressor`.
- `sklearn.linear_model`:
  - `ElasticNetCV`;
  - `HuberRegressor`;
  - `LassoLarsCV`;
  - `LogisticRegression`;
  - `PassiveAggressiveRegressor`;
  - `RANSACRegressor`;
  - `RidgeCV`;
  - `SGDRegressor`;
  - `TheilSenRegressor`.
- `sklearn.neighbors`:
  - `KNeighborsRegressor`;
  - `RadiusNeighborsRegressor`.
- `sklearn.neural_network`:
  - `MLPRegressor`.

最后生成一个数据结构如下的结果：



因此我们可以通过上述思想编制程序，从而得出以下的结论（得分均按照`r2_score`）：

1 `day.csv` 数据集中:

- 未注册用户预测最佳模型: GradientBoostingRegressor, 分数为 0.83;
- 注册用户预测最佳模型: GradientBoostingRegressor, 分数为 0.75;
- 总用户预测最佳模型: GradientBoostingRegressor, 分数为 0.78。

2 `hour.csv` 数据集中:

- 未注册用户预测最佳模型: BaggingRegressor, 分数为 0.81;
- 注册用户预测最佳模型: ExtraTreesRegressor, 分数为 0.88;
- 总用户预测最佳模型: ExtraTreesRegressor, 分数为 0.89。

3. 时间相关变量数据集中:

- 未注册用户预测最佳模型: GradientBoostingRegressor, 分数为 0.68;
- 注册用户预测最佳模型: GaussianProcessRegressor, 分数为 0.82;
- 总用户预测最佳模型: GaussianProcessRegressor, 分数为 0.79。

## 5 未来工作

S:future-work

1. 完善数据预处理部分，并且从分析模块中抽离出单独的数据预处理模块，遵从 SRP（单一职责原则）。其中分析模块中包括数据清洗与变换，以应对不进行数据预处理的数据集。
2. 在模型选择阶段由于各个模型之间不存在关系，因此可以进行并行化处理，即采用并行计算加速模型选择的速度。
3. 在模型选择的同时增加参数选择的步骤，或者对默认参数评估函数最高的模型进行参数选择，从而训练出更好的模型。
4. 文章适当加上原理介绍及分析，提高可信度。

## 6 结论

模型训练结果如下，且所有模型的训练时间不超过 500ms。

1 `day.csv` 数据集中:

- 未注册用户预测最佳模型: GradientBoostingRegressor;

- 注册用户预测最佳模型: GradientBoostingRegressor;
- 总用户预测最佳模型: GradientBoostingRegressor。

2 `hour.csv` 数据集中:

- 未注册用户预测最佳模型: BaggingRegressor;
- 注册用户预测最佳模型: ExtraTreesRegressor;
- 总用户预测最佳模型: ExtraTreesRegressor。

3. 时间相关变量数据集中:

- 未注册用户预测最佳模型: GradientBoostingRegressor;
- 注册用户预测最佳模型: GaussianProcessRegressor;
- 总用户预测最佳模型: GaussianProcessRegressor。

## 7 参考文献

### 参考文献

- 共享单车的行业背景 [1] 陈正豪. 从三个方面来看一看共享单车的行业背景[EB/OL]. 2017. <http://www.woshipm.com/evaluating/667758.html>.
- 共享单车发展研究分析 [2] 刘亚楠. 共享单车发展研究分析[J]. 时代金融, 2017(8).
- 共享单车发展对策研究 [3] 李琨浩. 基于共享经济视角下城市共享单车发展对策研究[J]. 城市, 2017(3):66-69.
- 分析与发展对策研究 [4] 范东旭, 刘威. 沈阳市共享单车大数据分析与发展对策研究[C]//共享与品质——2018 中国城市规划年会论文集 (06 城市交通规划) . [出版地不详: 出版者不详], 2018.
- data [5] FANAEH-T H, GAMA J. Event labeling combining ensemble detectors and background knowledge[J/OL]. Progress in Artificial Intelligence, 2013:1-15. <http://dx.doi.org/10.1007/s13748-013-0040-3>.
- 特征工程与数据降维 [6] 撇味大白菜. 机器学习笔记: 特征工程与数据降维[EB/OL]. 2018. [https://blog.csdn.net/weixin\\_42219368/article/details/81009387](https://blog.csdn.net/weixin_42219368/article/details/81009387).
- 挖掘中的数据预处理 [7] 刘明吉, 王秀峰, 黄亚楼. 数据挖掘中的数据预处理[J]. 计算机科学, 2000, 27(4):54-57.
- scikit-learn [8] PEDREGOSA F, VAROQUAUX G, GRAMFORT A, et al. Scikit-learn: Machine learning in Python[J]. Journal of Machine Learning Research, 2011, 12:2825-2830.
- sklearn\_api [9] BUITINCK L, LOUPPE G, BLONDEL M, et al. API design for machine learning software: experiences from the scikit-learn project[C]//ECML PKDD Workshop: Languages for Data Mining and Machine Learning. [S.l.: s.n.], 2013: 108-122.
- earn 中的降维算法 [10] ANALYST D. 机器学习之 sklearn 中的降维算法[EB/OL]. 2019. <https://zhuanlan.zhihu.com/p/59591749>.
- Cyrille2015sne [11] ROSSANT C. An illustrated introduction to the t-sne algorithm[EB/OL]. 2015. <https://www.oreilly.com/learning/an-illustrated-introduction-to-the-t-sne-algorithm>.
- 2012 统计学习方法 [12] 李航. 统计学习方法[M]. [出版地不详]: 清华大学出版社, 2012.

## A 模块之间依赖关系

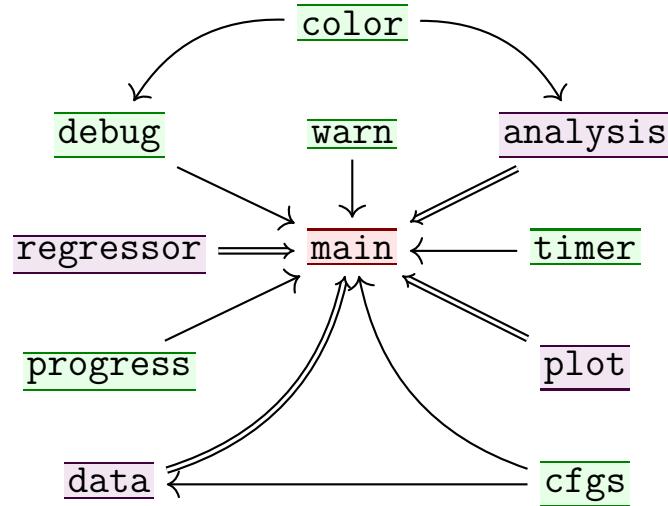
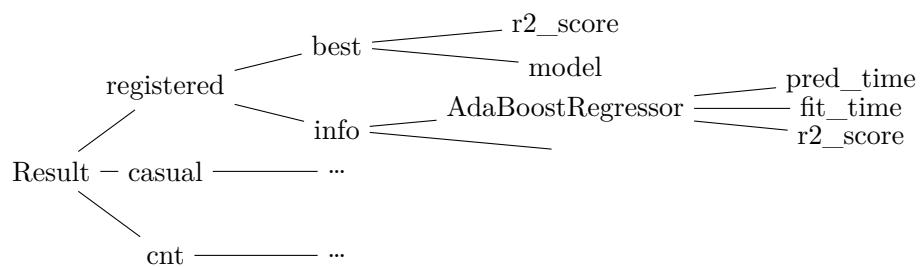


图 6: 程序间依赖关系

## B 运行结果数据结构



## C Python 代码附录

### C.1 第三方库依赖

#### 第三方库依赖

```
1 ipython==7.4.0
2 matplotlib==2.1.1
3 numpy==1.14.0
4 pandas==0.24.2
5 prettytable==0.7.2
6 scikit-learn==0.21.0
7 sklearn==0.0
8 tqdm==4.19.5
```

### C.2 主文件

#### 主文件

```
1 # -*- encoding: utf-8 -*-
2 from analysis import calc_accuracy, dimensionality_reduction
3 from cfgs import CFGS
4 from data import day_train, day_test, hour_train, hour_test, time_train
5     , time_test
6 from debug import debug
7 from plot import visualize_data
8 from progress import bar
9 from regressor import regressors
10 from timer import Timer
11 from warn import ignore_warnings
12
13 # y's tag
14 y_tags = [ "CASCOL" , "REGCOL" , "CNICOL" ]
15 # train and test data(hour or day)
16 trains = [day_train, hour_train, time_train]
17 tests = [day_test, hour_test, time_test]
```

```
18 index = 0
19 # result container
20 result = dict()
21 # if process dimensionality reduction
22 dec_flag = True
23 # if visualize data
24 vis_flag = True
25
26
27 if vis_flag:
28     visualize_data(trains, is_train=True)
29     visualize_data(tests, is_train=False)
30
31 for y_tag in y_tags:
32     result[y_tag] = dict()
33     result[y_tag][ "info" ] = dict()
34
35 # train and test data
36 train_x = trains[index][0]
37 train_y = trains[index][-1][CFGS[ "DATA" ][y_tag]]
38
39 test_x = tests[index][0]
40 test_y = tests[index][-1][CFGS[ "DATA" ][y_tag]]
41
42 # dimensionality reduction
43 if dec_flag:
44     decomodel,train_x_ = dimensionality_reduction(train_x, n_dim=6,
45                                                 method="TruncatedSVD")
46     test_x_ = decomodel.transform(test_x)
47 else:
48     train_x_,test_x_ = train_x, test_x
49
50 # sklearn model
51 best_score = 0.0
52 best_model = None
53 fit_time = 0.0
54 pred_time = 0.0
55 for regressor in bar(regressors):
```

```
55     with Timer() as t:
56         model = regressor()
57         model.fit(train_x_, train_y)
58     fit_time = t.toc()
59     y_true = test_y.to_numpy().astype(float)
60     with Timer() as t:
61         y_pred = model.predict(test_x_)
62     pred_time = t.toc()
63     accuracy = calc_accuracy(y_true, y_pred, display=True)
64
65     result[y_tag]["info"][regressor] = {
66         "r2_score": accuracy["score"]["r2_score"],
67         "fit_time": fit_time,
68         "pred_time": pred_time,
69     }
70     if result[y_tag]["info"][regressor]["r2_score"] > best_score:
71         best_score = result[y_tag]["info"][regressor]["r2_score"]
72         best_model = model
73     result[y_tag]["best"] = {
74         "r2_score": best_score,
75         "model": best_model,
76     }
77
78 debug(locals())
```

### C.3 分析模块（数据预处理、数据降维、模型评价）

#### 分析模块（数据预处理、数据降维、模型评价）

```
1  # -*- encoding: utf-8 -*-
2  from prettytable import PrettyTable
3
4  import numpy as np
5  from sklearn import metrics
6  from sklearn import preprocessing
7  from sklearn import decomposition
```



```
45 ignore.append(item)
46 if display:
47     tabu, numerical = None, None
48     for name in names:
49         tabu = PrettyTable([ "Name\u201d of \u201d%os\u201d%color(name) , "Value" ])
50         for item in locals().__dict__.get(name):
51             if item in ignore:
52                 continue
53             numerical = "%.3e" % result[name][item]
54             tabu.add_row([ color(item, "青") , numerical])
55         print(tabu)
56 return result
57
58
59 def dimensionality_reduction(data, n_dim=None, thold=0.9, method="PCA",
60                               help_me=False, **arg_d):
61     """Dimensionality reduction
62
63     Parameters
64     ======
65     data : n_samples x n_features
66     n_dim: n-dimension
67     thold: Float, threshold
68     method: String, decomposition method
69         {"主成分分析": ["PCA", "IncrementalPCA", "KernelPCA",
70                      "MiniBatchSparsePCA", "SparsePCA",
71                      "TruncatedSVD"],
72          "因子分析": ["FactorAnalysis"], 
73          "独立成分分析": ["FastICA"], 
74          "字典学习": ["DictionaryLearning",
75                      "MiniBatchDictionaryLearning"], 
76          "高级矩阵分解": ["LatentDirichletAllocation", "NMF"], 
77          "其他矩阵分解": ["SparseCoder"]},
78     help_me: Boolean
79     arg_d: Dict[args of sklearn.decomposition.?]
80                  Dict{name:value}
81
82     Return
```

```
82 =====
83     decomodel: decomposition model
84     data_ :       data after transform
85
86     Examples
87 =====
88     >>> dec, data_ = dimensionality_reduction(data, thold=0.9, method="PCA"):
89
90     Reference
91 =====
92     - [Zhihu](https://zhuanlan.zhihu.com/p/59591749)
93     - [Oreilly](https://www.oreilly.com/learning/an-illustrated-
94         introduction-to-the-t-sne-algorithm)
95     - [Scikit](http://lidiancheng0614.github.io/scikit-learn/modules/
96         generated/sklearn.decomposition.PCA.html)
97     """
98     methods = ["DictionaryLearning", "FactorAnalysis", "FastICA",
99                 "IncrementalPCA", "KernelPCA", "
100                     LatentDirichletAllocation",
101                     "MiniBatchDictionaryLearning", "MiniBatchSparsePCA", "
102                     NMF",
103                     "PCA", "SparseCoder", "SparsePCA", "TruncatedSVD"]
104     if method not in methods:
105         raise NameError("Method %s not in sklearn.decomposition"%color(
106             method))
107     alias = getattr(decomposition, method)
108     if n_dim is not None:
109         decomodel = alias(n_components=n_dim, **arg_d)
110         decomodel.fit(data)
111         return decomodel, decomodel.transform(data)
112     elif thold is not None:
113         decomodel = alias(**arg_d)
114         decomodel.fit(data)
115         if "explained_variance_ratio_" in dir(decomodel):
116             ratio = decomodel.explained_variance_ratio_
117             index = np.sum(np.cumsum(ratio) < thold)
118             decomodel.components_ = decomodel.components_[:index]
```

```
114     else:
115         raise NotImplementedError("TODO")
116     return decomodel, decomodel.transform(data)
117 else:
118     raise ValueError("There must be at most two None in n_dim and
119                      thold .")
120
121 def data_preprocess(*argl, **argd):
122     """TODO
123     """
124     raise NotImplementedError("TODO")
```

## C.4 数据模块（训练集与测试集）

### 数据模块

```
1  # -*- encode: utf-8 -*-
2  import os
3  from datetime import datetime
4  import pandas as pd
5  from cfgs import CFGS
6
7
8  def split_data(data):
9      """Split data into train and test.
10      """
11      def is_train(date_str):
12          date = datetime.strptime(date_str,
13                                  CFGS["TIME"]["PATTERN"])
14          return date.day < 20
15
16      index_tr = list(map(is_train,
17                          data[CFGS["DATA"]["DATECOL"]]))
18      index_te = list(map(lambda x: not x, index_tr))
19      return data[index_tr], data[index_te]
```

```
20
21 def split_xy(data):
22     """Split data into covariate vector and response.
23     """
24     igno_index = [ "INSCOL" , "DATECOL" ]
25     igno_index = [CFGS[ "DATA" ][ i ] for i in igno_index]
26     resp_index = [ "CASCOL" , "REGCOL" , "CNTCOL" ]
27     resp_index = [CFGS[ "DATA" ][ i ] for i in resp_index]
28     data_ = data . drop(igno_index , axis=1)
29     return data_.drop(resp_index , axis=1) , data_[ resp_index ]
30
31
32
33
34 day = pd.read_csv(
35     os.path.join(
36         CFGS[ "DATA" ][ "DIR" ] , CFGS[ "DATA" ][ "DAY" ] ) ,
37         encoding=CFGS[ "IO" ][ "ENCODE" ])
38 day_train , day_test = split_data(day)
39 day_train = split_xy(day_train)
40 day_test = split_xy(day_test)
41
42 hour = pd.read_csv(
43     os.path.join(
44         CFGS[ "DATA" ][ "DIR" ] , CFGS[ "DATA" ][ "HOUR" ] ) ,
45         encoding=CFGS[ "IO" ][ "ENCODE" ])
46 hour_train , hour_test = split_data(hour)
47 hour_train = split_xy(hour_train)
48 hour_test = split_xy(hour_test)
49
50 time_index = eval(CFGS[ "DATA" ][ "TIMECOL" ])
51 time_train = [hour_train[0][time_index] , hour_train[-1]]
52 time_test = [hour_test[0][time_index] , hour_test[-1]]
```

## C.5 可视化模块

### 可视化模块

```
1  # -*- encode: utf-8 -*-
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from cfgs import CFGS
5
6
7  def visualize_data(data, is_train=True):
8      day, hour, time = data
9      titles = ["Day", "Hour", "Time"]
10     if is_train:
11         visualize_day_data(day, range(365))
12         visualize_hour_data(hour, range(24*7))
13     else:
14         visualize_day_data(day)
15         visualize_hour_data(hour)
16     visualize_time_data(time)
17
18
19  def visualize_day_data(day, index=None):
20      """Example
21      >>> day = [day_train_x, day_train_y]
22      >>> visualize_day_data(day)
23      """
24      cols = eval(CFGS["DATA"]["NOMCOL"])
25      length = day[0].shape[0]
26      index = index or range(length)
27      data_x = day[0][cols].iloc[index]
28      data_y = day[1].iloc[index]
29      plt.subplot(121)
30      plt.plot(index, data_x)
31      plt.legend(data_x.columns)
32      plt.subplot(122)
33      plt.plot(index, data_y)
34      plt.legend(data_y.columns)
35      plt.show()
```

```
36
37
38 def visualize_hour_data(hour, index=None):
39     """Example
40     >>> hour = [hour_train_x, hour_train_y]
41     >>> visualize_hour_data(hour)
42     """
43     cols = eval(CFGS[ "DATA" ][ "NOMCOL" ])
44     length = hour[0].shape[0]
45     index = index or range(length)
46     data_x = hour[0][cols].iloc[index]
47     data_y = hour[1].iloc[index]
48     plt.subplot(121)
49     plt.plot(index, data_x)
50     plt.legend(data_x.columns)
51     plt.subplot(122)
52     plt.plot(index, data_y)
53     plt.legend(data_y.columns)
54     plt.show()
55
56
57 def visualize_time_data(time, index=None):
58     """Example
59     >>> time = [time_train_x, time_train_y]
60     >>> visualize_time_data(time)
61     """
62     length = time[0].shape[0]
63     index = np.array(index or range(length))
64     time_x = time[0].iloc[index]
65     time_y = time[1].iloc[index]
66     for key, val in eval(CFGS[ "DATA" ][ "PLOTTIMECOL" ]).items():
67         for i in range(val[0], val[1]+1):
68             idx = time_x[key] == i
69             plt.plot(index[idx], time_y[idx], "--")
70     plt.title(key)
71     # plt.legend(["springer", "summer", "fall", "winter"])
72     plt.show()
```

## C.6 回归模型模块

### 回归模型模块

```
1  # -*- encoding: utf-8 -*-
2  from sklearn.ensemble import AdaBoostRegressor, BaggingRegressor, \
3      ExtraTreesRegressor, GradientBoostingRegressor, IsolationForest
4
5  from sklearn.gaussian_process import GaussianProcessRegressor
6
7  from sklearn.linear_model import ElasticNetCV, HuberRegressor, \
8      LassoLarsCV, LogisticRegression, PassiveAggressiveRegressor, \
9      RANSACRegressor, RidgeCV, SGDRegressor, TheilSenRegressor
10
11 from sklearn.neighbors import KNeighborsRegressor,
12     RadiusNeighborsRegressor
13
14
15
16 regressors = [AdaBoostRegressor, BaggingRegressor, ExtraTreesRegressor,
17     GradientBoostingRegressor, IsolationForest,
18     GaussianProcessRegressor,
19     ElasticNetCV, HuberRegressor, LassoLarsCV, LogisticRegression,
20     PassiveAggressiveRegressor, RANSACRegressor, RidgeCV, SGDRegressor,
21     TheilSenRegressor, KNeighborsRegressor, RadiusNeighborsRegressor,
22     MLPRegressor]
```

## C.7 配置文件模块

### 配置文件模块

```
1  # -*- encoding: utf-8 -*-
2  from configparser import ConfigParser
3
4
5  string = """
```

```

6  [IO]
7  ENCODE=utf-8
8
9  [DATA]
10 DIR=data
11 DAY=day.csv
12 HOUR=hour.csv
13 DATECOL=dteday
14 CASCOL=casual
15 REGCOL=registered
16 CNTCOL=cnt
17 INSCOL=instant
18 TIMECOL=["season", "yr", "mnth", "hr", "holiday", "weekday", "
    workingday"]
19 PLOTTIMECOL={"season": [1, 4], "mnth": [1, 12], "hr": [0, 23], "workingday"
    : [0, 1]}
20 NOMCOL=["temp", "atemp", "hum", "windspeed"]
21
22 [TIME]
23 PATTERN=%Y-%m-%d
24 """
25
26 CFGS = ConfigParser()
27 CFGS.read_string(string)

```

## C.8 彩色字符串模块

### 彩色字符串模块

```

1  # -*- encoding: utf-8 -*-
2  def color(string:str, fore="紫", back="黑", disp="默认"):
3      """Coloring text.
4
5      Parameters
6      ======
7          string: String, text to be colored.

```

```
8         fore : String , foreground color .
9         back : String , background color .
10        disp : String , display effect .
11
12    Return
13    =====
14    Text with color information .
15
16    Examples
17    =====
18    >>> _color( "Hello world" , "蓝" , "黑" , "默认" )
19    """
20    colormap = { "黑" : [ "30" , "40" ] ,
21                "红" : [ "31" , "41" ] ,
22                "绿" : [ "32" , "42" ] ,
23                "黄" : [ "33" , "43" ] ,
24                "蓝" : [ "34" , "44" ] ,
25                "紫" : [ "35" , "45" ] ,
26                "青" : [ "36" , "46" ] ,
27                "白" : [ "37" , "47" ] , }
28    disp_map = { "默认" :"0" ,
29                 "高亮" :"1" ,
30                 "下划" :"4" ,
31                 "闪烁" :"5" ,
32                 "反白" :"7" ,
33                 "隐藏" :"8" , }
34    return "\033[%" s ; %s ; %sm%s \033[0m" %
35          ( disp_map [ disp ] , colormap [ fore ] [ 0 ] , \
36            colormap [ back ] [ -1 ] , string )
```

## C.9 调试模块

### 调试模块

```
1  #-*- encode: utf-8 -*-
2  from IPython import embed
3
4  from color import color
5
6
7  def debug(locals_=None):
8      for k, v in locals_.items():
9          locals_()[k] = v
10     print("\n\n")
11     hint = "Enter `if not debug:`"
12     if input(color(hint, "蓝")):
13         del k, v, locals_, hint
14         embed(colors="Neutral")
```

## C.10 进度条模块

### 进度条模块

```
1  #-*- encode: utf-8 -*-
2  from tqdm import tqdm_gui as bar
```

## C.11 计时模块

### 计时模块

```
1  #-*- encode: utf-8 -*-
2  import time
3
4
5  class Timer:
```

```
6     def __enter__(self):
7         self.TIC_TOC = time.time()
8         return self
9
10    def __exit__(self, type, value, traceback):
11        self.TIC_TOC = time.time() - self.TIC_TOC
12
13    def toc(self):
14        return self.TIC_TOC
```

## C.12 消除警告模块

### 消除警告模块

```
1  # -*- encoding: utf-8 -*-
2  import warnings
3
4
5  warnings.filterwarnings("ignore")
6  ignore_warnings = True
```