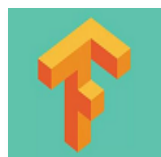


【Python数据挖掘课程】九.回归模型

LinearRegression简单分析氧化物数据

原创 Eastmount 最后发布于2017-03-05 18:39:46 阅读数 12008 ☆ 收藏

编辑 展开



Python+TensorFlow人工智能

该专栏为人工智能入门专栏，采用Python3和TensorFlow实现人工智能相...



Eastmount

¥9.90

去订阅

这篇文章主要介绍三个知识点，也是我《数据挖掘与分析》课程讲课的内容。同时主要参考学生的课程提交作业内容进行讲述，包括：

- 1.回归模型及基础知识；
- 2.UCI数据集；
- 3.回归模型简单数据分析。

前文推荐：

- 【Python数据挖掘课程】一.安装Python及爬虫入门介绍
- 【Python数据挖掘课程】二.Kmeans聚类数据分析及Anaconda介绍
- 【Python数据挖掘课程】三.Kmeans聚类代码实现、作业及优化
- 【Python数据挖掘课程】四.决策树DTC数据分析及鸢尾数据集分析
- 【Python数据挖掘课程】五.线性回归知识及预测糖尿病实例
- 【Python数据挖掘课程】六.Numpy、Pandas和Matplotlib包基础知识
- 【Python数据挖掘课程】七.PCA降维操作及subplot子图绘制
- 【Python数据挖掘课程】八.关联规则挖掘及Apriori实现购物推荐

希望这篇文章对你有所帮助，尤其是刚刚接触数据挖掘以及**大数据**的同学，这些基础知识真的非常重要。如果文章中存在不足或错误的地方，还请海涵~

感谢ZJ学生提供的数据集与作品相关报告，学生确实还是学到了些东西。

授课知识强推第五节课内容：五.线性回归知识及预测糖尿病实例

一. 算法简介-回归模型

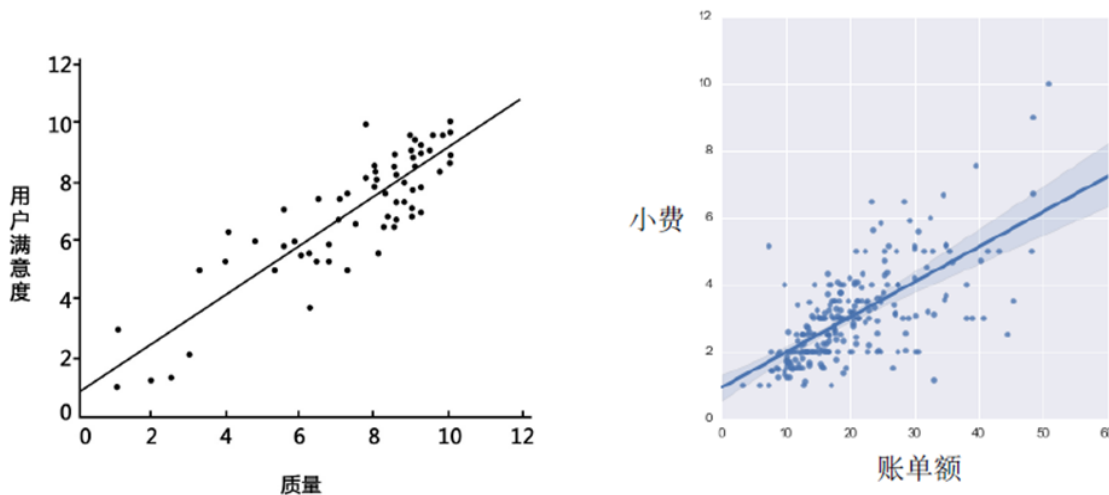
1.初识回归

“子女的身高趋向于高于父母的身高的平均值，但一般不会超过父母的身高。”

-- 《遗传的身高向平均数方向的回归》

回归(Regression)这一概念最早由英国生物统计学家高尔顿和他的学生皮尔逊在研究父母亲 and 子女的身高遗传特性时提出。如今，我们做回归分析时所讨论的“回归”和这种趋势中效应已经没有任何瓜葛了，它只是指源于高尔顿工作的那样——用一个或多个自变量来预测因变量的数学方法。

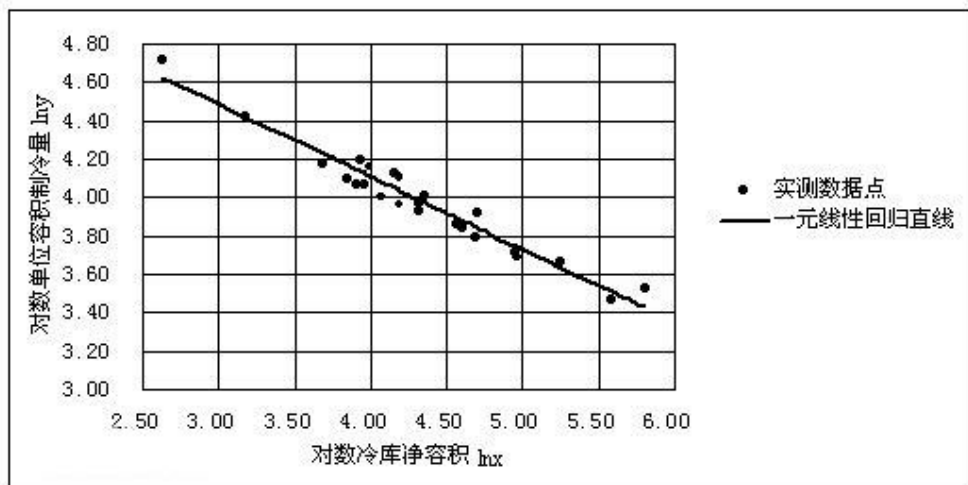
在一个回归模型中，我们需要关注或预测的变量叫做因变量（响应变量或结果变量），我们选取的用来解释因变量变化的变量叫做自变量（解释变量或预测变量）。



回归是统计学中最有力的工具之一。机器学习监督学习算法分为分类算法和回归算法两种，其实就是根据类别标签分布类型为离散型、连续性而定义的。

分类算法用于离散型分布预测，如KNN、决策树、朴素贝叶斯、adaboost、SVM、Logistic回归都是分类算法；回归算法用于连续型分布预测，针对的是数值型的样本，使用回归，可以在给定输入的时候预测出一个数值，这是对分类方法的提升，因为这样可以预测连续型数据而不仅仅是离散的类别标签。

回归的目的就是建立一个回归方程用来预测目标值，回归的求解就是求这个回归方程的回归系数。预测的方法即回归系数乘以输入值再全部相加就得到了预测值。



回归最简单的定义：给出一个点集D，用一个函数去拟合这个点集，并且使得点集与拟合函数间的误差最小，如果这个函数曲线是一条直线，那就被称为线性回归，如果曲线是一条二次曲线，就被称为二次回归。

2.线性回归

假定预测值与样本特征间的函数关系是线性的，回归分析的任务，就在于根据样本X和Y的观察值，去估计函数h，寻求变量之间近似的函数关系。定义：

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

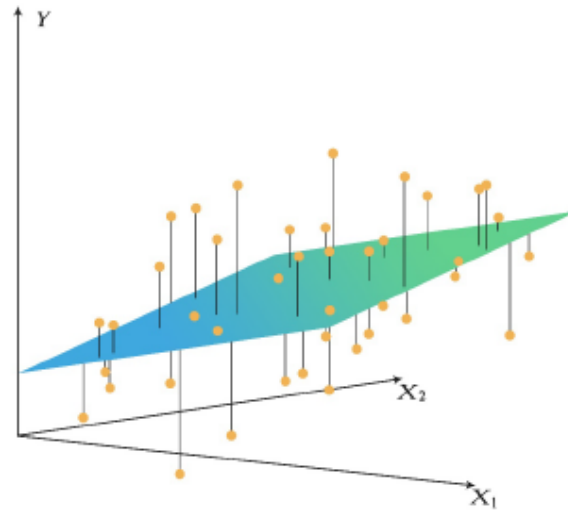
其中，n = 特征数目； x_j = 每个训练样本第j个特征的值，可以认为是特征向量中的第j个值。

为了方便，记 $x_0 = 1$ ，则多变量线性回归可以记为：

$$h_{\theta}(x) = \theta^T x ,$$

其中， θ 、 x 都表示(n+1, 1)维列向量。

注意：多元和多次是两个不同的概念，“多元”指方程有多个参数，“多次”指的是方程中参数的最高次幂。多元线性方程是假设预测值y与样本所有特征值符合一个多元一次线性方程。



3.求解线性回归

回归常常指线性回归，回归的求解就是多元线性回归方程的求解。假设有连续型值标签(标签值分布为Y)的样本，有 $X=\{x_1, x_2, \dots, x_n\}$ 个特征，回归就是求解回归系数 $\theta = \theta_0, \theta_1, \dots, \theta_n$ 。那么，手里有一些X和对应的Y,怎样才能找到 θ 呢？

在回归方程里，求得特征对应的最佳回归系数的方法是最小化误差的平方和。这里的误差是指预测y值和真实y值之间的差值，使用该误差的简单累加将使得正差值和负差值相互抵消，所以采用平方误差（**最小二乘法**）。平方误差可以写做：

$$\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

在数学上，求解过程就转化为求一组 θ 值使求上式取到最小值，那么求解方法有梯度下降法、Normal Equation等等。

梯度下降有如下特点：需要预先选定步长a、需要多次迭代、特征值需要Scaling（统一到同一个尺度范围）。因此比较复杂，还有一种不需要迭代的求解方式——Normal Equation，简单、方便、不需要Feature Scaling。Normal Equation方法中需要计算X的转置与逆矩阵，计算量很大，因此特征个数多时计算会很慢，只适用于特征个数小于100000时使用；当特征数量大于100000时使用梯度法。

另外，当X不可逆时就有岭回归算法的用武之地了。

3.1 梯度下降法 (Gradient Descent)

根据平方误差，定义该线性回归模型的损耗函数（Cost Function）为：

$$J(\theta) = J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

线性回归的损耗函数的值与回归系数 θ 的关系是碗状的，只有一个最小点。线性回归的求解过程如同Logistic回归，区别在于学习模型函数 $h_{\theta}(x)$ 不同。

3.2 普通最小二乘法 (Normal Equation)

Normal Equation算法也叫做普通最小二乘法（ordinary least squares），其特点是：给定输入矩阵 X ，如果 $X^T X$ 的逆存在并可以求得的话，就可以直接采用该方法求解。其求解理论也十分简单：既然是求最小误差平方和，另其导数为0即可得出回归系数。

$$\theta = (X^T X)^{-1} X^T y$$

矩阵 X 为 $(m, n+1)$ 矩阵（ m 表示样本数、 n 表示一个样本的特征数）， y 为 $(m, 1)$ 列向量。上述公式中包含 $X^T X$ ，也就是需要对矩阵求逆，因此这个方程只在逆矩阵存在的时候适用。

4.回归模型性能度量

数据集上计算出的回归方程并不一定意味着它是最佳的，可以使用预测值 y_{Hat} 和原始值 y 的相关性来度量回归方程的好坏。相关性取值范围0~1，值越高说明回归模型性能越好。

线性回归是假设值标签与特征值之间的关系是线性的，但有些时候数据间的关系可能会更加复杂，使用线性的模型就难以拟合，就需要引入多项式曲线回归（多元多次拟合）或者其他回归模型，如回归树。

注意：

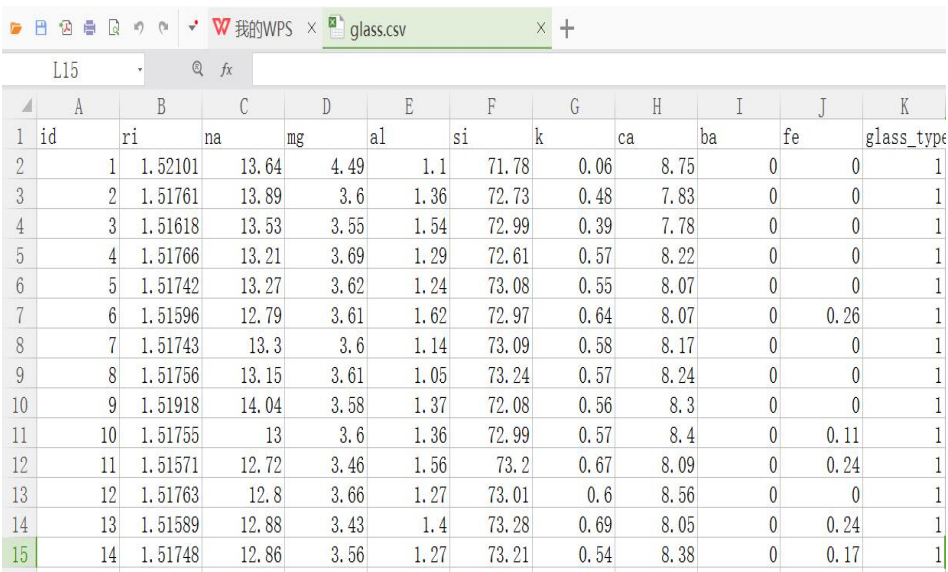
多元回归存在多重共线性，自相关性和异方差性。线性回归对异常值非常敏感。它会严重影响回归线，最终影响预测值。多重共线性会增加系数估计值的方差，使得在模型轻微变化下，估计非常敏感，结果就是系数估计值不稳定。

二. 数据集介绍

在数据分析中数据集是最重要的信息，推荐数据集UCI：

<http://archive.ics.uci.edu/ml/machine-learning-databases/>

该数据集包括6种类型的玻璃，各个特征是定义它们的氧化物含量（即钠，铁，钾等）。Mg：镁 Ai：铝 Si：硅 K：钾 Ca：钙 Ba：钡 Fe：铁 Type of glass：级属性。数据集位glass.csv文件，如下图所示：



	A	B	C	D	E	F	G	H	I	J	K
1	id	ri	na	mg	al	si	k	ca	ba	fe	glass_type
2	1	1.52101	13.64	4.49	1.1	71.78	0.06	8.75	0	0	1
3	2	1.51761	13.89	3.6	1.36	72.73	0.48	7.83	0	0	1
4	3	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0	0	1
5	4	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0	0	1
6	5	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0	0	1
7	6	1.51596	12.79	3.61	1.62	72.97	0.64	8.07	0	0.26	1
8	7	1.51743	13.3	3.6	1.14	73.09	0.58	8.17	0	0	1
9	8	1.51756	13.15	3.61	1.05	73.24	0.57	8.24	0	0	1
10	9	1.51918	14.04	3.58	1.37	72.08	0.56	8.3	0	0	1
11	10	1.51755	13	3.6	1.36	72.99	0.57	8.4	0	0.11	1
12	11	1.51571	12.72	3.46	1.56	73.2	0.67	8.09	0	0.24	1
13	12	1.51763	12.8	3.66	1.27	73.01	0.6	8.56	0	0	1
14	13	1.51589	12.88	3.43	1.4	73.28	0.69	8.05	0	0.24	1
15	14	1.51748	12.86	3.56	1.27	73.21	0.54	8.38	0	0.17	1

详细内容如下：

id	ri	na	mg	al	si	k	ca	ba
fe	glass_type							
1	1.52101	13.64	4.49	1.1	71.78	0.06	8.75	0
0	1							
2	1.51761	13.89	3.6	1.36	72.73	0.48	7.83	0
0	1							
3	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0
0	1							
4	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0
0	1							
5	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0
0	1							
6	1.51596	12.79	3.61	1.62	72.97	0.64	8.07	0
0.26	1							
7	1.51743	13.3	3.6	1.14	73.09	0.58	8.17	0
0	1							
8	1.51756	13.15	3.61	1.05	73.24	0.57	8.24	0
0	1							
9	1.51918	14.04	3.58	1.37	72.08	0.56	8.3	0
0	1							
10	1.51755	13	3.6	1.36	72.99	0.57	8.4	0
0.11	1							

11	1.51571	12.72	3.46	1.56	73.2	0.67	8.09	0
0.24	1							
12	1.51763	12.8	3.66	1.27	73.01	0.6	8.56	0
0	1							
13	1.51589	12.88	3.43	1.4	73.28	0.69	8.05	0
0.24	1							
14	1.51748	12.86	3.56	1.27	73.21	0.54	8.38	0
0.17	1							
15	1.51763	12.61	3.59	1.31	73.29	0.58	8.5	0
0	1							
16	1.51761	12.81	3.54	1.23	73.24	0.58	8.39	0
0	1							
17	1.51784	12.68	3.67	1.16	73.11	0.61	8.7	0
0	1							
18	1.52196	14.36	3.85	0.89	71.36	0.15	9.15	0
0	1							
19	1.51911	13.9	3.73	1.18	72.12	0.06	8.89	0
0	1							
20	1.51735	13.02	3.54	1.69	72.73	0.54	8.44	0
0.07	1							
21	1.5175	12.82	3.55	1.49	72.75	0.54	8.52	0
0.19	1							
22	1.51966	14.77	3.75	0.29	72.02	0.03	9	0
0	1							
23	1.51736	12.78	3.62	1.29	72.79	0.59	8.7	0
0	1							
24	1.51751	12.81	3.57	1.35	73.02	0.62	8.59	0
0	1							
25	1.5172	13.38	3.5	1.15	72.85	0.5	8.43	0
0	1							
26	1.51764	12.98	3.54	1.21	73	0.65	8.53	0
0	1							
27	1.51793	13.21	3.48	1.41	72.64	0.59	8.43	0
0	1							
28	1.51721	12.87	3.48	1.33	73.04	0.56	8.43	0
0	1							
29	1.51768	12.56	3.52	1.43	73.15	0.57	8.54	0
0	1							
30	1.51784	13.08	3.49	1.28	72.86	0.6	8.49	0
0	1							
31	1.51768	12.65	3.56	1.3	73.08	0.61	8.69	0
0.14	1							
32	1.51747	12.84	3.5	1.14	73.27	0.56	8.55	0
0	1							
33	1.51775	12.85	3.48	1.23	72.97	0.61	8.56	0.09
0.22	1							

34	1.51753	12.57	3.47	1.38	73.39	0.6	8.55	0
0.06	1							
35	1.51783	12.69	3.54	1.34	72.95	0.57	8.75	0
0	1							
36	1.51567	13.29	3.45	1.21	72.74	0.56	8.57	0
0	1							
37	1.51909	13.89	3.53	1.32	71.81	0.51	8.78	0.11
0	1							
38	1.51797	12.74	3.48	1.35	72.96	0.64	8.68	0
0	1							
39	1.52213	14.21	3.82	0.47	71.77	0.11	9.57	0
0	1							
40	1.52213	14.21	3.82	0.47	71.77	0.11	9.57	0
0	1							
41	1.51793	12.79	3.5	1.12	73.03	0.64	8.77	0
0	1							
42	1.51755	12.71	3.42	1.2	73.2	0.59	8.64	0
0	1							
43	1.51779	13.21	3.39	1.33	72.76	0.59	8.59	0
0	1							
44	1.5221	13.73	3.84	0.72	71.76	0.17	9.74	0
0	1							
45	1.51786	12.73	3.43	1.19	72.95	0.62	8.76	0
0.3	1							
46	1.519	13.49	3.48	1.35	71.95	0.55	9	0
0	1							
47	1.51869	13.19	3.37	1.18	72.72	0.57	8.83	0
0.16	1							
48	1.52667	13.99	3.7	0.71	71.57	0.02	9.82	0
0.1	1							
49	1.52223	13.21	3.77	0.79	71.99	0.13	10.02	0
0	1							
50	1.51898	13.58	3.35	1.23	72.08	0.59	8.91	0
0	1							
51	1.5232	13.72	3.72	0.51	71.75	0.09	10.06	0
0.16	1							
52	1.51926	13.2	3.33	1.28	72.36	0.6	9.14	0
0.11	1							
53	1.51808	13.43	2.87	1.19	72.84	0.55	9.03	0
0	1							
54	1.51837	13.14	2.84	1.28	72.85	0.55	9.07	0
0	1							
55	1.51778	13.21	2.81	1.29	72.98	0.51	9.02	0
0.09	1							
56	1.51769	12.45	2.71	1.29	73.7	0.56	9.06	0
0.24	1							

57 0.31	1.51215 1	12.99	3.47	1.12	72.98	0.62	8.35	0
58 0	1.51824 1	12.87	3.48	1.29	72.95	0.6	8.43	0
59 0	1.51754 1	13.48	3.74	1.17	72.99	0.59	8.03	0
60 0.11	1.51754 1	13.39	3.66	1.19	72.79	0.57	8.27	0
61 0	1.51905 1	13.6	3.62	1.11	72.64	0.14	8.76	0
62 0	1.51977 1	13.81	3.58	1.32	71.72	0.12	8.67	0.69
63 0.11	1.52172 1	13.51	3.86	0.88	71.79	0.23	9.54	0
64 0	1.52227 1	14.17	3.81	0.78	71.35	0	9.69	0
65 0.07	1.52172 1	13.48	3.74	0.9	72.01	0.18	9.61	0
66 0	1.52099 1	13.69	3.59	1.12	71.96	0.09	9.4	0
67 0.17	1.52152 1	13.05	3.65	0.87	72.22	0.19	9.85	0
68 0.17	1.52152 1	13.05	3.65	0.87	72.32	0.19	9.85	0
69 0.16	1.52152 1	13.12	3.58	0.9	72.2	0.23	9.82	0
70 0.03	1.523 1	13.31	3.58	0.82	71.99	0.12	10.17	0
71 0.12	1.51574 2	14.86	3.67	1.74	71.87	0.16	7.36	0
72 0.32	1.51848 2	13.64	3.87	1.27	71.96	0.54	8.32	0
73 0	1.51593 2	13.09	3.59	1.52	73.1	0.67	7.83	0
74 0	1.51631 2	13.34	3.57	1.57	72.87	0.61	7.89	0
75 0	1.51596 2	13.02	3.56	1.54	73.11	0.72	7.9	0
76 0	1.5159 2	13.02	3.58	1.51	73.12	0.69	7.96	0
77 0	1.51645 2	13.44	3.61	1.54	72.39	0.66	8.03	0
78 0	1.51627 2	13	3.58	1.54	72.83	0.61	8.04	0
79 0.14	1.51613 2	13.92	3.52	1.25	72.88	0.37	7.94	0

80	1.5159	12.82	3.52	1.9	72.86	0.69	7.97	0
0	2							
81	1.51592	12.86	3.52	2.12	72.66	0.69	7.97	0
0	2							
82	1.51593	13.25	3.45	1.43	73.17	0.61	7.86	0
0	2							
83	1.51646	13.41	3.55	1.25	72.81	0.68	8.1	0
0	2							
84	1.51594	13.09	3.52	1.55	72.87	0.68	8.05	0
0.09	2							
85	1.51409	14.25	3.09	2.08	72.28	1.1	7.08	0
0	2							
86	1.51625	13.36	3.58	1.49	72.72	0.45	8.21	0
0	2							
87	1.51569	13.24	3.49	1.47	73.25	0.38	8.03	0
0	2							
88	1.51645	13.4	3.49	1.52	72.65	0.67	8.08	0
0.1	2							
89	1.51618	13.01	3.5	1.48	72.89	0.6	8.12	0
0	2							
90	1.5164	12.55	3.48	1.87	73.23	0.63	8.08	0
0.09	2							
91	1.51841	12.93	3.74	1.11	72.28	0.64	8.96	0
0.22	2							
92	1.51605	12.9	3.44	1.45	73.06	0.44	8.27	0
0	2							
93	1.51588	13.12	3.41	1.58	73.26	0.07	8.39	0
0.19	2							
94	1.5159	13.24	3.34	1.47	73.1	0.39	8.22	0
0	2							
95	1.51629	12.71	3.33	1.49	73.28	0.67	8.24	0
0	2							
96	1.5186	13.36	3.43	1.43	72.26	0.51	8.6	0
0	2							
97	1.51841	13.02	3.62	1.06	72.34	0.64	9.13	0
0.15	2							
98	1.51743	12.2	3.25	1.16	73.55	0.62	8.9	0
0.24	2							
99	1.51689	12.67	2.88	1.71	73.21	0.73	8.54	0
0	2							
100	1.51811	12.96	2.96	1.43	72.92	0.6	8.79	0.14
0	2							
101	1.51655	12.75	2.85	1.44	73.27	0.57	8.79	0.11
0.22	2							
102	1.5173	12.35	2.72	1.63	72.87	0.7	9.23	0
0	2							

103	1.5182	12.62	2.76	0.83	73.81	0.35	9.42	0
0.2	2							
104	1.52725	13.8	3.15	0.66	70.57	0.08	11.64	0
0	2							
105	1.5241	13.83	2.9	1.17	71.15	0.08	10.79	0
0	2							
106	1.52475	11.45	0	1.88	72.19	0.81	13.24	0
0.34	2							
107	1.53125	10.73	0	2.1	69.81	0.58	13.3	3.15
0.28	2							
108	1.53393	12.3	0	1	70.16	0.12	16.19	0
0.24	2							
109	1.52222	14.43	0	1	72.67	0.1	11.52	0
0.08	2							
110	1.51818	13.72	0	0.56	74.45	0	10.99	0
0	2							
111	1.52664	11.23	0	0.77	73.21	0	14.68	0
0	2							
112	1.52739	11.02	0	0.75	73.08	0	14.96	0
0	2							
113	1.52777	12.64	0	0.67	72.02	0.06	14.4	0
0	2							
114	1.51892	13.46	3.83	1.26	72.55	0.57	8.21	0
0.14	2							
115	1.51847	13.1	3.97	1.19	72.44	0.6	8.43	0
0	2							
116	1.51846	13.41	3.89	1.33	72.38	0.51	8.28	0
0	2							
117	1.51829	13.24	3.9	1.41	72.33	0.55	8.31	0
0.1	2							
118	1.51708	13.72	3.68	1.81	72.06	0.64	7.88	0
0	2							
119	1.51673	13.3	3.64	1.53	72.53	0.65	8.03	0
0.29	2							
120	1.51652	13.56	3.57	1.47	72.45	0.64	7.96	0
0	2							
121	1.51844	13.25	3.76	1.32	72.4	0.58	8.42	0
0	2							
122	1.51663	12.93	3.54	1.62	72.96	0.64	8.03	0
0.21	2							
123	1.51687	13.23	3.54	1.48	72.84	0.56	8.1	0
0	2							
124	1.51707	13.48	3.48	1.71	72.52	0.62	7.99	0
0	2							
125	1.52177	13.2	3.68	1.15	72.75	0.54	8.52	0
0	2							

PS：现在正在步入第四科学范式，第一范式是实验（哥白尼），第二范式是理论（牛顿），第三范式是计算（四色填充地图），第四范式是数据。

三. 回归模型分析

回归模型分析代码如下：

注意：1) pandas、Matplotlib、seaboard三种不同方法绘制图形，基本类似。

2) 代码对应结果不进行详细分析，只提供方法，为提升学生阅读代码能力。

```
# -*- coding: utf-8 -*-
"""
Created on Sun Mar 05 18:10:07 2017

@author: eastmount & zj
"""

# 导入玻璃识别数据集
import pandas as pd
glass=pd.read_csv("glass.csv")
#显示前6行数据
print(glass.shape)
print(glass.head(6))

import seaborn as sns
import matplotlib.pyplot as plt
sns.set(font_scale=1.5)
sns.lmplot(x='al', y='ri', data=glass, ci=None)
#利用Pandas画散点图
glass.plot(kind='scatter', x='al', y='ri')
plt.show()

#利用matplotlib做等效的散点图
plt.scatter(glass.al, glass.ri)
plt.xlabel('al')
plt.ylabel('ri')

#拟合线性回归模型
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
feature_cols = ['al']
X = glass[feature_cols]
```

```

y = glass.ri
linreg.fit(X, y)
plt.show()

#对于所有的x值做出预测
glass['ri_pred'] = linreg.predict(X)
print("预测的前六行:")
print(glass.head(6))

#用直线表示预测结果
plt.plot(glass.al, glass.ri_pred, color='red')
plt.xlabel('al')
plt.ylabel('Predicted ri')
plt.show()

#将直线结果和散点图同时显示出来
plt.scatter(glass.al, glass.ri)
plt.plot(glass.al, glass.ri_pred, color='red')
plt.xlabel('al')
plt.ylabel('ri')
plt.show()

#利用相关方法线性预测
linreg.intercept_ + linreg.coef_ * 2
#使用预测方法计算Al = 2的预测
linreg.predict(2)

#铝检验系数
ai=zip(feature_cols, linreg.coef_)
print(ai)

#使用预测方法计算Al = 3的预测
pre=linreg.predict(3)
print(pre)

#检查glass_type
sort=glass.glass_type.value_counts().sort_index()
print(sort)

#类型1、2、3的窗户玻璃
#类型5, 6, 7是家用玻璃
glass['household'] = glass.glass_type.map({1:0, 2:0, 3:0, 5:1, 6:1, 7:1})
print(glass.head())

plt.scatter(glass.al, glass.household)
plt.xlabel('al')

```

```
plt.ylabel('household')
plt.show()

#拟合线性回归模型并存储预测
feature_cols = ['al']
X = glass[feature_cols]
y = glass.household
linreg.fit(X, y)
glass['household_pred'] = linreg.predict(X)
plt.show()

#包括回归线的散点图
plt.scatter(glass.al, glass.household)
plt.plot(glass.al, glass.household_pred, color='red')
plt.xlabel('al')
plt.ylabel('household')
plt.show()
```

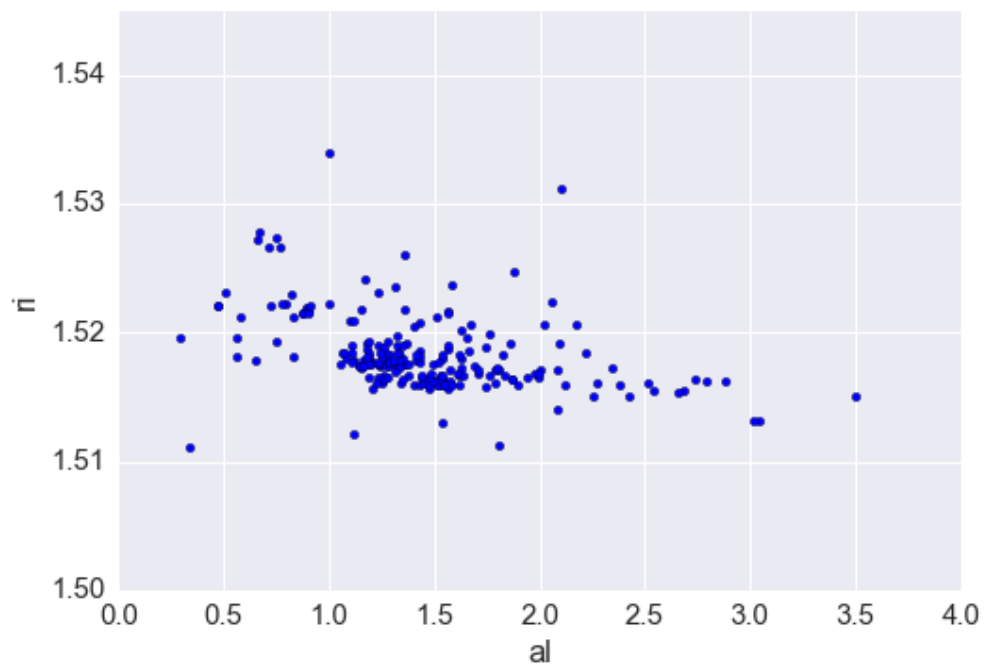
输出结果如下：

预测的前六行：

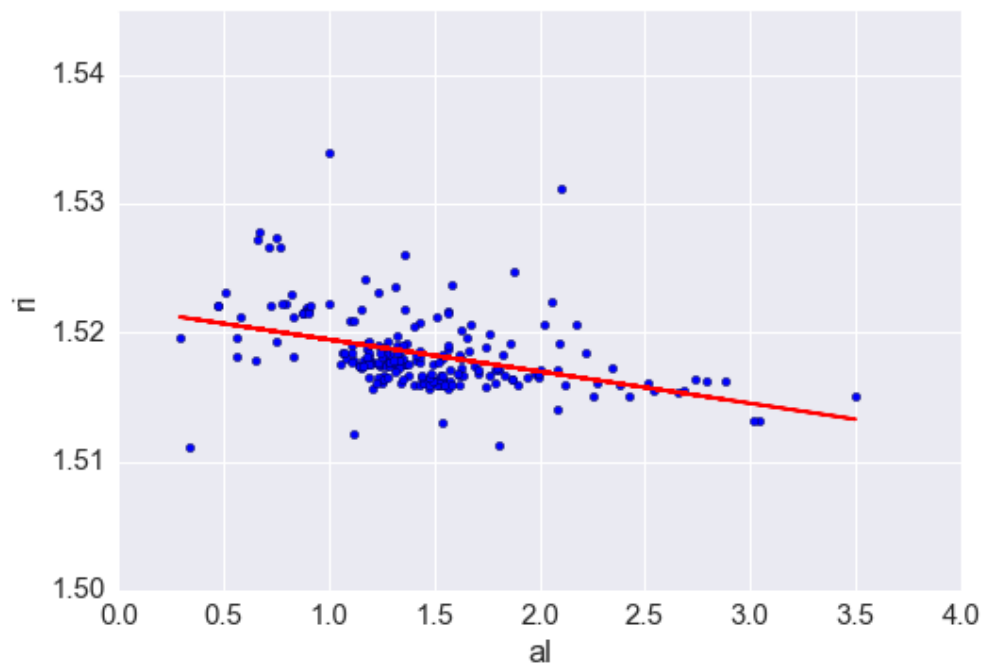
	id	ri	na	mg	al	si	k	ca	ba	fe	glass_type \
0	1	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.00	1
1	2	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.00	1
2	3	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.00	1
3	4	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.00	1
4	5	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.00	1
5	6	1.51596	12.79	3.61	1.62	72.97	0.64	8.07	0.0	0.26	1

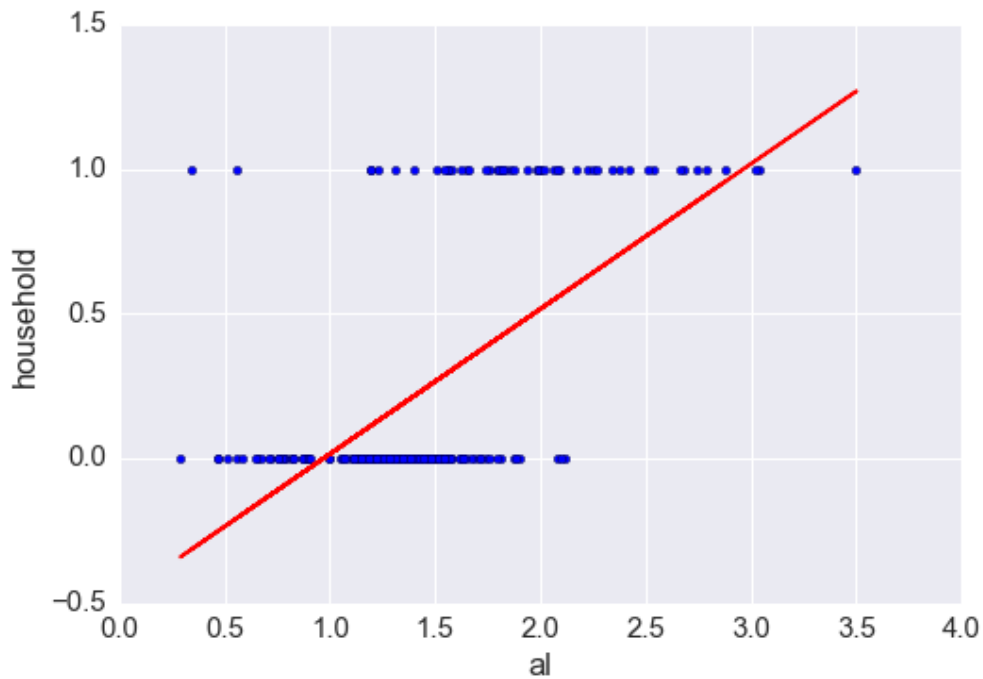
	ri_pred
0	1.519220
1	1.518576
2	1.518130
3	1.518749
4	1.518873
5	1.517932

部分输出如下图所示，绘制图形al和ri基本点状图：



将预测的线性回归直线结果和散点图同时显示出来：





拟合逻辑回归代码：

```
# -*- coding: utf-8 -*-
"""
Created on Sun Mar 05 18:28:56 2017

@author: eastmount & zj
"""

#-----逻辑回归-----

#拟合Logistic回归模型，存储类预测
import numpy as np
nums = np.array([5, 15, 8])
np.where(nums > 10, 'big', 'small')

#将household_pred转换为 1或0
glass['household_pred_class'] = np.where(glass.household_pred >= 0.5, 1, 0)
print(glass.head(6))
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(C=1e9)
feature_cols = ['al']
X = glass[feature_cols]
y = glass.household
logreg.fit(X, y)
glass['household_pred_class'] = logreg.predict(X)
```


#绘图- 显示预测结果

```
plt.scatter(glass.al, glass.household)
plt.plot(glass.al, glass.household_pred_class, color='red')
plt.xlabel('al')
plt.ylabel('household')
plt.show()
```

```
glass['household_pred_prob'] = logreg.predict_proba(X[:, 1])
```

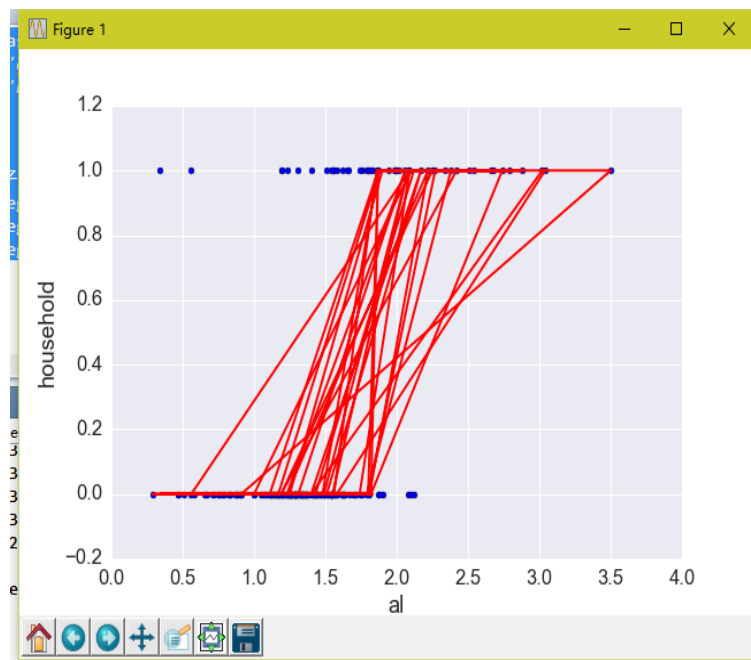
#绘图 绘制预测概率

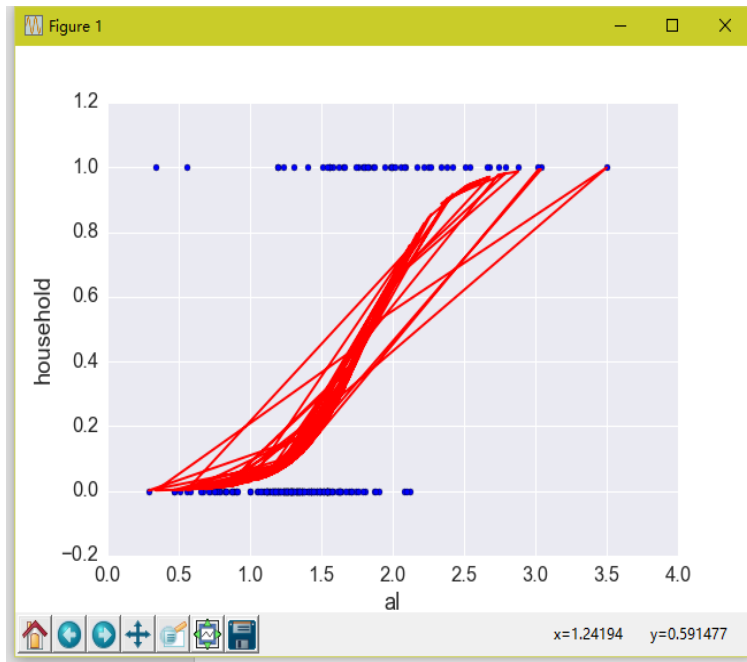
```
plt.scatter(glass.al, glass.household)
plt.plot(glass.al, glass.household_pred_prob, color='red')
plt.xlabel('al')
plt.ylabel('household')
plt.show()
```

#检查一些例子的预测

```
print (logreg.predict_proba (1))
print (logreg.predict_proba(2))
print (logreg. predict_proba (3))
```

输出如下图所示：





最后希望这篇文章对你有所帮助，尤其是我的学生和接触数据挖掘、**机器学习**的博友。新学期开始，各种事情，专注教学、科研及项目，加油~

爱你的一切，伴着尤克里里的琴声写了一下午，谢谢我的女神。

(By:Eastmount 2017-03-05 下午6点半 <http://blog.csdn.net/eastmount/>)

👍 点赞 12 ☆ 收藏 ➦ 分享



Eastmount  博客专家

发布了444 篇原创文章 · 获赞 5908 · 访问量 484万+