线性回归

**线性回归**：在统计学中，线性回归（Linear Regression）是利用称为线性回归方程的最小平方函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析。这种函数是一个或多个称为回归系数的模型参数的线性组合（自变量都是一次方）。只有一个自变量的情况称为简单回归，大于一个自变量情况的叫做多元回归。

**优点**：结果易于理解，计算上不复杂。

**缺点**：对非线性数据拟合不好。

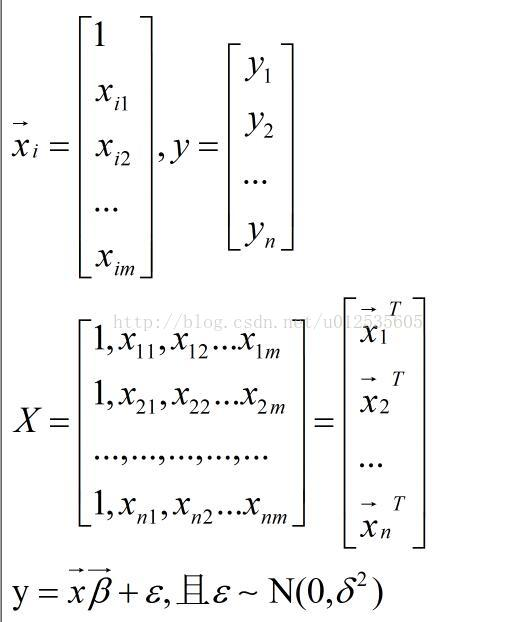
**适用数据类型**：数值型和标称型数据。

**算法类型**：回归算法

假设有如下方程，就是一个回归方程，系数wi为回归系数，求回归系数的过程被称为回归。



线性回归意味着可以将输入项分别乘以一些常量，再将结果加起来得到输出。



**（请事先安装好所需要的库pandas\matplotlib\sklearn）**

**思路分析：**

Advertising.csv里的数据共4 列200 行，每一行为一个特定的商品，前3为输入特征，最后一列为输出特征。

**数据可以从这个链接直接下载得到**：

http://www-bcf.usc.edu/~gareth/ISL/Advertising.csv

输入特征：

(1)TV：该商品用于电视上的广告费用（以千元为单位，下同）

(2)Radio：在广播媒体上投资的广告费用

(3)Newspaper：用于报纸媒体的广告费用

输出特征：

Sales：该商品的销量

**代码**：

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cross\_validation import train\_test\_split

from sklearn.linear\_model import LinearRegression

import numpy as np

# 1.读取数据

data = pd.read\_csv('Advertising.csv')

data.head()

X = data[['TV', 'radio', 'newspaper']]

y = data['sales']

#画图

plt.figure(figsize=(9,12))

plt.subplot(311)

plt.plot(data['TV'], y, 'ro')

plt.title('TV')

plt.grid()

plt.subplot(312)

plt.plot(data['radio'], y, 'g^')

plt.title('radio')

plt.grid()

plt.subplot(313)

plt.plot(data['newspaper'], y, 'b\*')

plt.title('newspaper')

plt.grid()

plt.tight\_layout()

plt.show()

#创建一个Python特性名称列表

feature\_cols = ['TV', 'radio', 'newspaper']

#使用该列表选择原始帧的一个子集

X = data[feature\_cols]

print (X.head())

#检查x的类型和形状。 print (type(X))

print (X.shape)

#选择从一系列的数据帧

y = data['sales']

print (y.head())

#构建训练集与测试集

X\_train,X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=1)

print (X\_train.shape)

print (y\_train.shape)

print (X\_test.shape)

print (y\_test.shape)

#sklearn的线性回归

linreg = LinearRegression()

model=linreg.fit(X\_train, y\_train)

print (model)

print (linreg.intercept\_)

print (linreg.coef\_)

# pair the feature names with the coefficients

zip(feature\_cols, linreg.coef\_)

# 预测

y\_pred = linreg.predict(X\_test)

print (y\_pred)

print (type(y\_pred))

#回归问题的评价测度

print (type(y\_pred),type(y\_test))

print (len(y\_pred),len(y\_test))

print (y\_pred.shape,y\_test.shape)

sum\_mean=0

for i in range(len(y\_pred)):

sum\_mean+=(y\_pred[i]-y\_test.values[i])\*\*2

print ("RMSE by hand:", np.sqrt(sum\_mean/len(y\_pred)))

#作图

plt.figure()

plt.plot(range(len(y\_pred)),y\_pred,'b',label="predict")

plt.plot(range(len(y\_pred)),y\_test,'r',label="test")

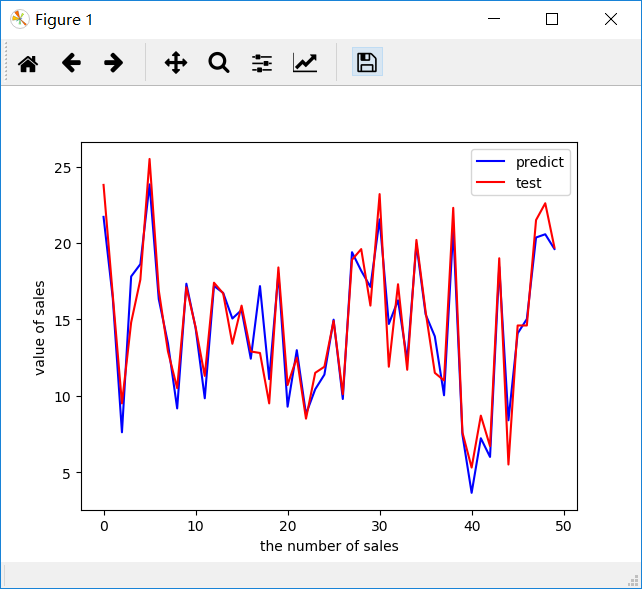
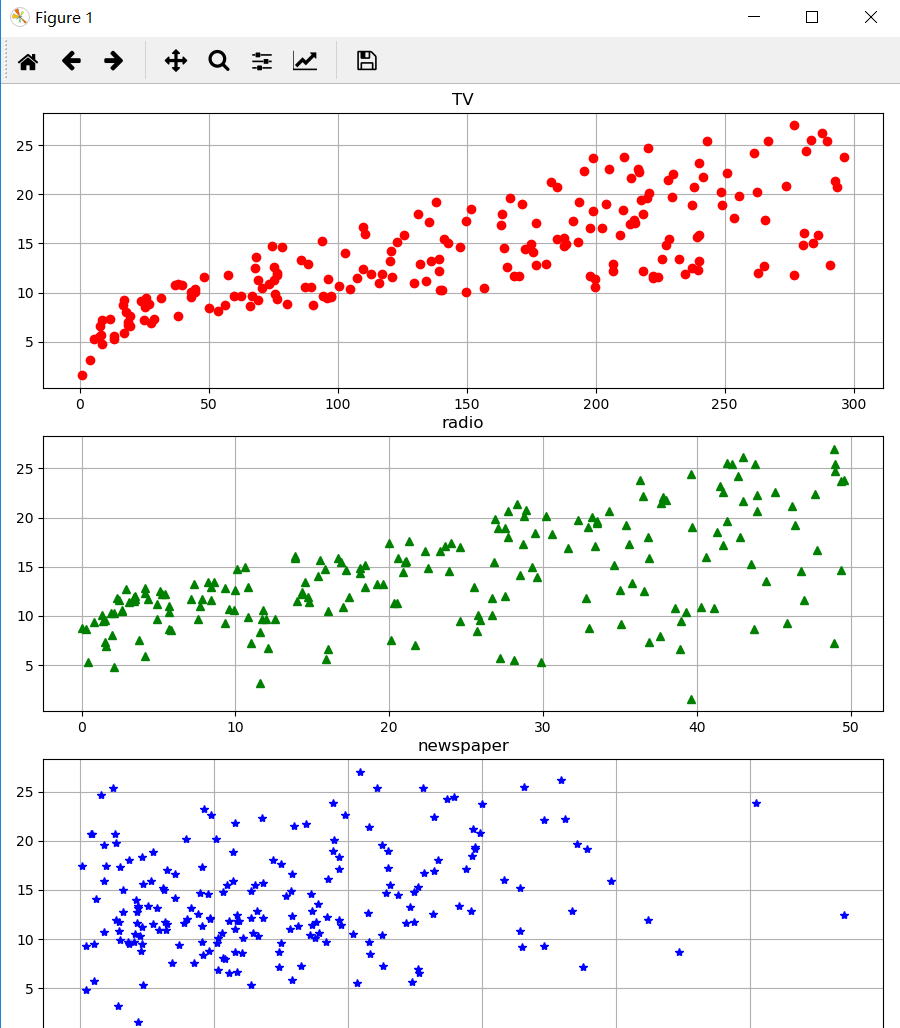
plt.legend(loc="upper right") #显示图中的标签

plt.xlabel("the number of sales")

plt.ylabel('value of sales')

plt.show()

**执行结果**：



**注意事项：**

本模型虽然简单，但它涵盖了机器学习的相当部分的内容。如使用75%的训练集和25%的测试集，这往往是探索机器学习模型的第一步。分析结果的权值和特征的数据分布，我们使用了最为简单的方法：直接删除；但这样做，仍然得到了更好的预测结果。

在机器学习中有“奥卡姆剃刀”的原理，即：如果能够用简单模型解决问题，则不使用更为复杂的模型。因为复杂模型往往增加了不确定性，造成过多的人力和物力成本，且容易过拟合。