万字长文，演绎八种线性回归算法最强总结！

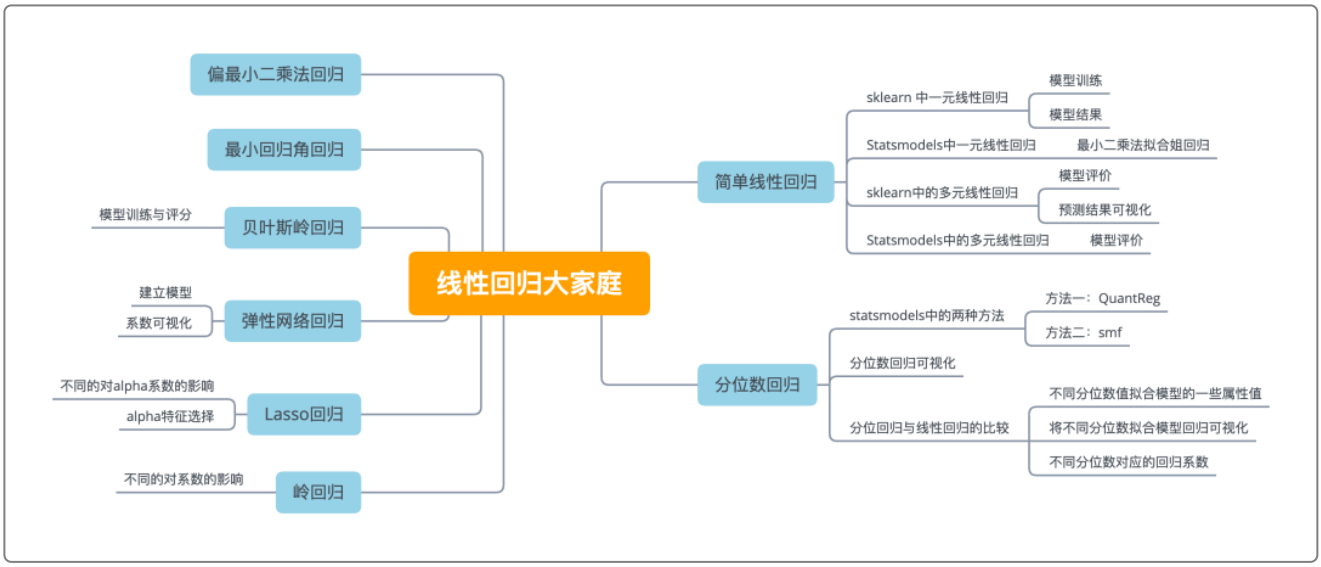
原创 云朵君 数据STUDIO 2021-05-19 11:30

收录于合集
#机器学习 110 #精选总结 94

导读： 本文以应用为导向，简单总结九种线性回归理论概念，重点说明如何用Python实现。从数据准备、数据预处理、模型建立、模型调参、模型评价及结果可视化方面介绍常用的线性回归模型及应用。

1. 简单线性回归
2. 岭回归
3. Lasso回归
4. 弹性网络回归
5. 贝叶斯岭回归
6. 最小回归角回归
7. 偏最小二乘法回归
8. 分位数回归

公众号



回归分析是一种预测性的建模技术，它研究的是因变量(目标)和自变量(预测器)之间的关系。这种技术通常用于预测分析、时间序列模型以及发现变量之间的因果关系。

- 回归分析是一种通过建立模型来研究变量之间相互关系的密切程度、结构状态及进行模型预测的有效工具。
- 回归分析是建模和分析数据的重要工具。

- 回归分析估计了两个或多个变量之间的关系。
- 回归分析表明自变量和因变量之间的显著关系。
- 回归分析表明多个自变量对一个因变量的影响强度。
- 回归分析也允许我们去比较那些衡量不同尺度的变量之间的相互影响。

这些技术主要有三个度量

- 自变量的个数
- 因变量的类型
- 回归线的形状

数据准备

这里使用AMD股市数据，具体数据准备可参见Python用于金融数据准备。

```
dataset.head()
```

Date	Open	High	Low	Close	Adj Close	Volume	Open_Close	High_Low	Increase_Decrease	Buy_Sell_on_Open	Buy_Sell	Returns
2014-01-03	3.98	4.00	3.88	4.00	4.00	22887200	-0.005025	0.030928	1	1	1	0.012658
2014-01-06	4.01	4.18	3.99	4.13	4.13	42398300	-0.029925	0.047619	1	1	1	0.032500
2014-01-07	4.19	4.25	4.11	4.18	4.18	42932100	0.002387	0.034063	0	1	0	0.012107
2014-01-08	4.23	4.26	4.14	4.18	4.18	30678700	0.011820	0.028986	0	0	0	0.000000
2014-01-09	4.20	4.23	4.05	4.09	4.09	30667600	0.026190	0.044444	0	0	1	-0.021531

本文中所有可视化图形均有源码，在公众号「数据STUDIO」中回复【线性回归可视化】即可获取。

简单线性回归

线性回归是回归的最简单形式。因变量是连续的，因变量与自变量之间的关系假设为线性关系。

> $y = \beta x + \theta$
> y:因变量(目标)
> x:自变量(预测器)
> β:常数和斜率或坡度
> θ: 残差或截距项

线性回归有一些重要前提：

- 自变量和因变量之间必须有线性关系。
- 不应该出现任何异常值。

- 没有异方差性。
- 样本应该是独立同分布。
- 误差项应均值为0，方差恒定的服从正态分布。
- 不存在多重共线性和自相关。

sklearn 中一元线性回归

```
X = dataset.iloc[ : , 'Open'].values
Y = dataset.iloc[ : , 'Adj Close'].values
X = np.array(X).reshape(X.shape[0],-1)
Y = np.array(Y).reshape(Y.shape[0],-1)
X_train, X_test, Y_train, Y_test = train_test_split(
    X, Y, test_size = 1/4, random_state = 0)
```

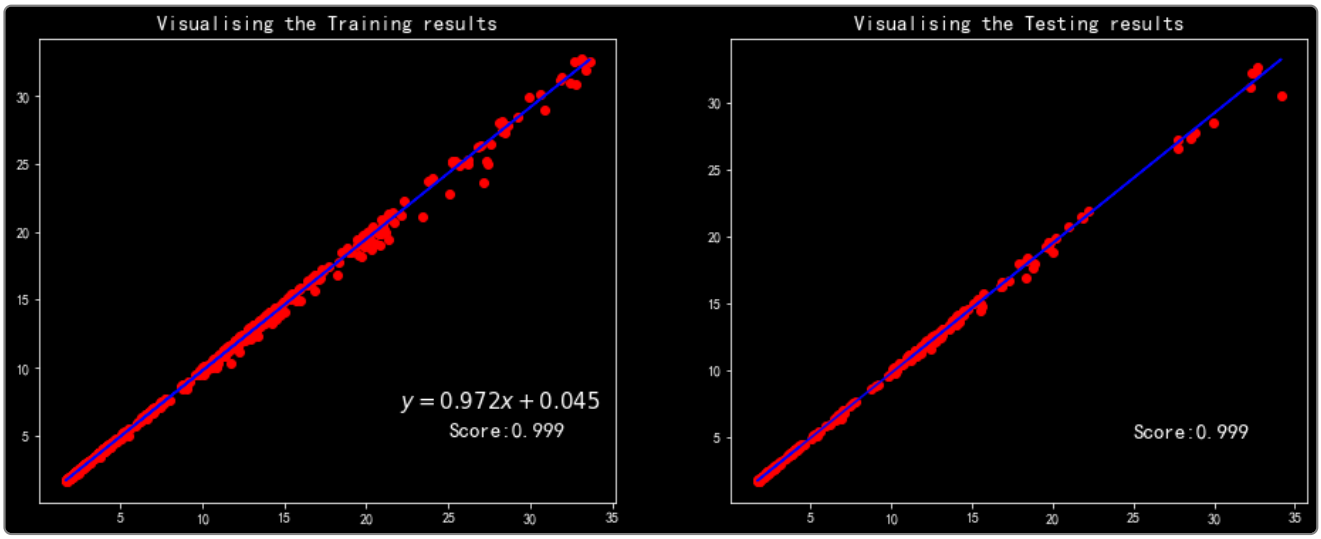
模型训练

从数据集中获取数据点并将其转换为训练点。然后使用训练点来拟合模型。

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor = regressor.fit(X_train, Y_train)
Y_pred = regressor.predict(X_test)
```

模型结果

模型在测试集（右图）与训练集（左图）上的真实值与预测线性模型的可视化。线性模型方程为 $y = 0.972x + 0.045$。



Statsmodels中一元线性回归

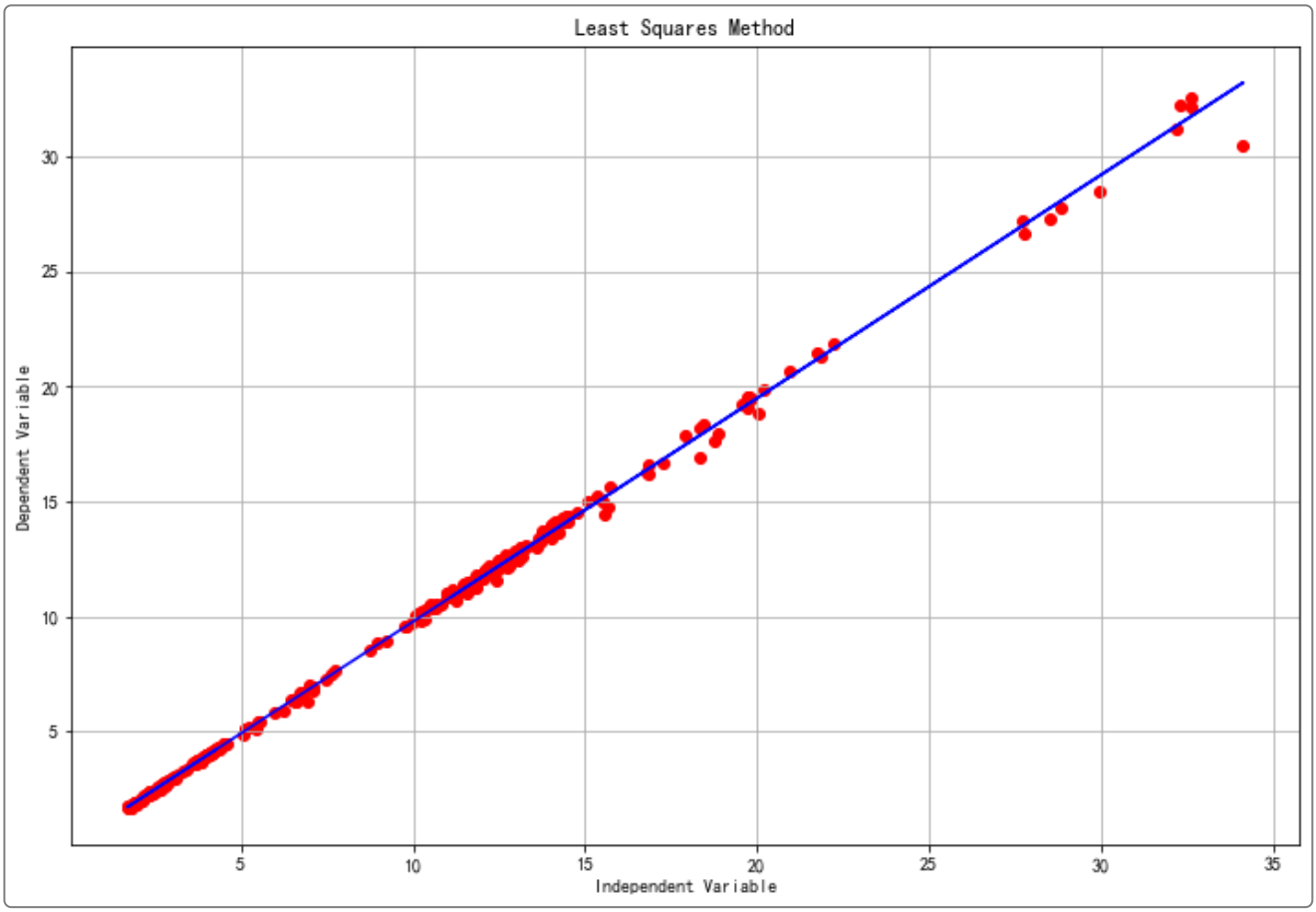
Statsmodels中线性回归用的是**最小二乘法**，而最小二乘法是拟合回归线最常用的方法。该方法通过使每个数据点到直线的垂直偏差平方和最小化来计算观测数据的最佳拟合直线。

```
import statsmodels.api as sm
X = sm.add_constant(X) # 添加一个常量
model = sm.OLS(Y, X).fit()
predictions = model.predict(X)
print(model.summary())
```

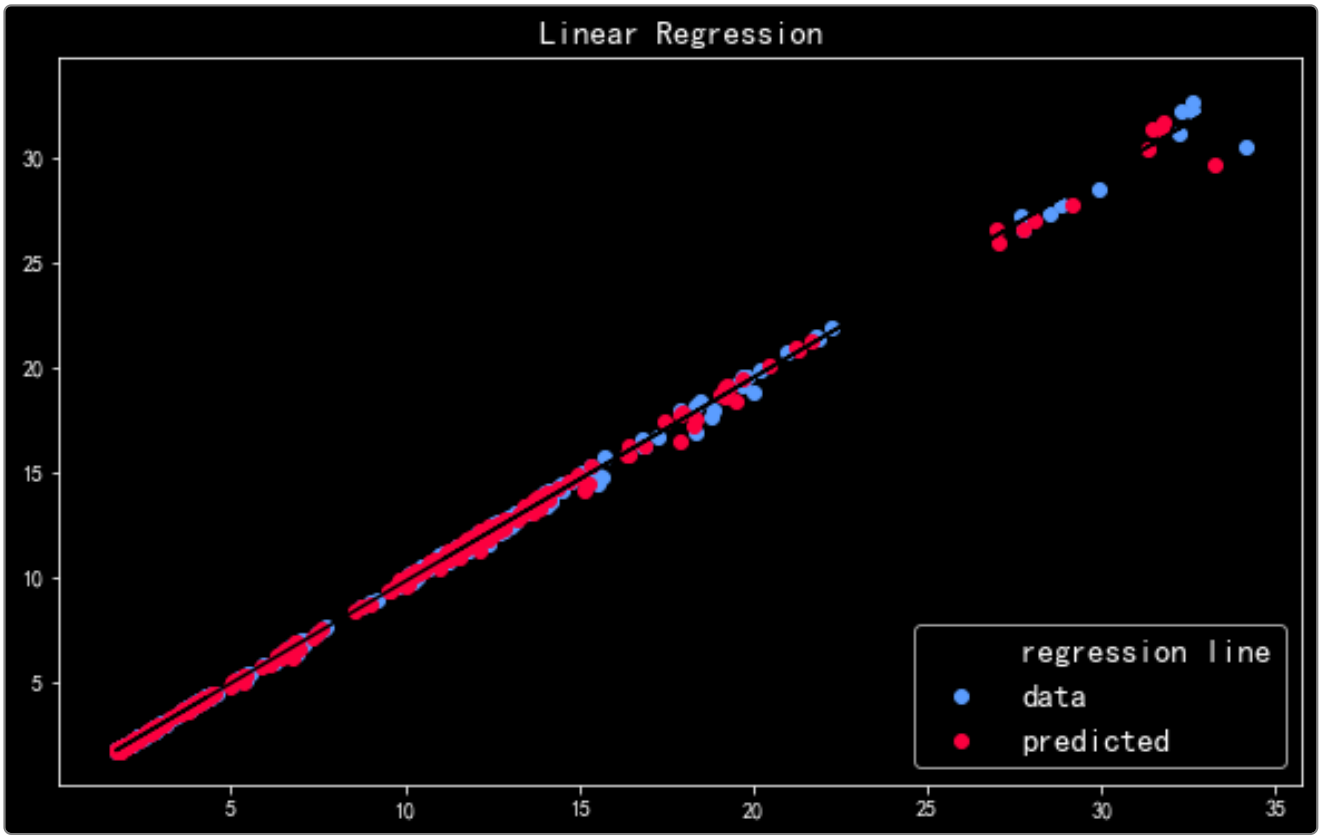
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                 8.707e+05
Date:                Mon, 26 Apr 2021   Prob (F-statistic):               0.00
Time:                        16:42:30   Log-Likelihood:                -19.230
No. Observations:                1259   AIC:                             42.46
Df Residuals:                    1257   BIC:                             52.74
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0451      0.011      4.058      0.000       0.023       0.067
x1             0.9718      0.001    933.097      0.000       0.970       0.974
==============================================================================
Omnibus:                     1084.120   Durbin-Watson:                   2.103
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            56843.489
Skew:                          -3.678   Prob(JB):                         0.00
Kurtosis:                      35.086   Cond. No.                         17.2
==============================================================================
```

最小二乘法拟合姐回归

其模型结果与sklearn中一元线性回归中几乎一样。



另外将真实值与预测值比较，检查回归的拟合效果。可以看出真实值(蓝色点)与预测值红色点)几乎是重合的，因此此时模型拟合效果非常棒。



sklearn中的多元线性回归

```
X = dataset.drop(['Adj Close','Close'], axis=1)
y = dataset[['Adj Close']]
X_train, X_test, y_train, y_test = train_test_split(
      X, y, test_size=0.2,random_state=1)
lr = LinearRegression()
lr.fit(X_train, y_train)
print('Slope: ',lr.coef_)
print('Intercept: ',lr.intercept_)
```

```
Slope:  [[ -6.296422e-01   8.056899e-01
          8.253628e-01  -6.697286e-11]]
Intercept:  [-0.00183821]
```

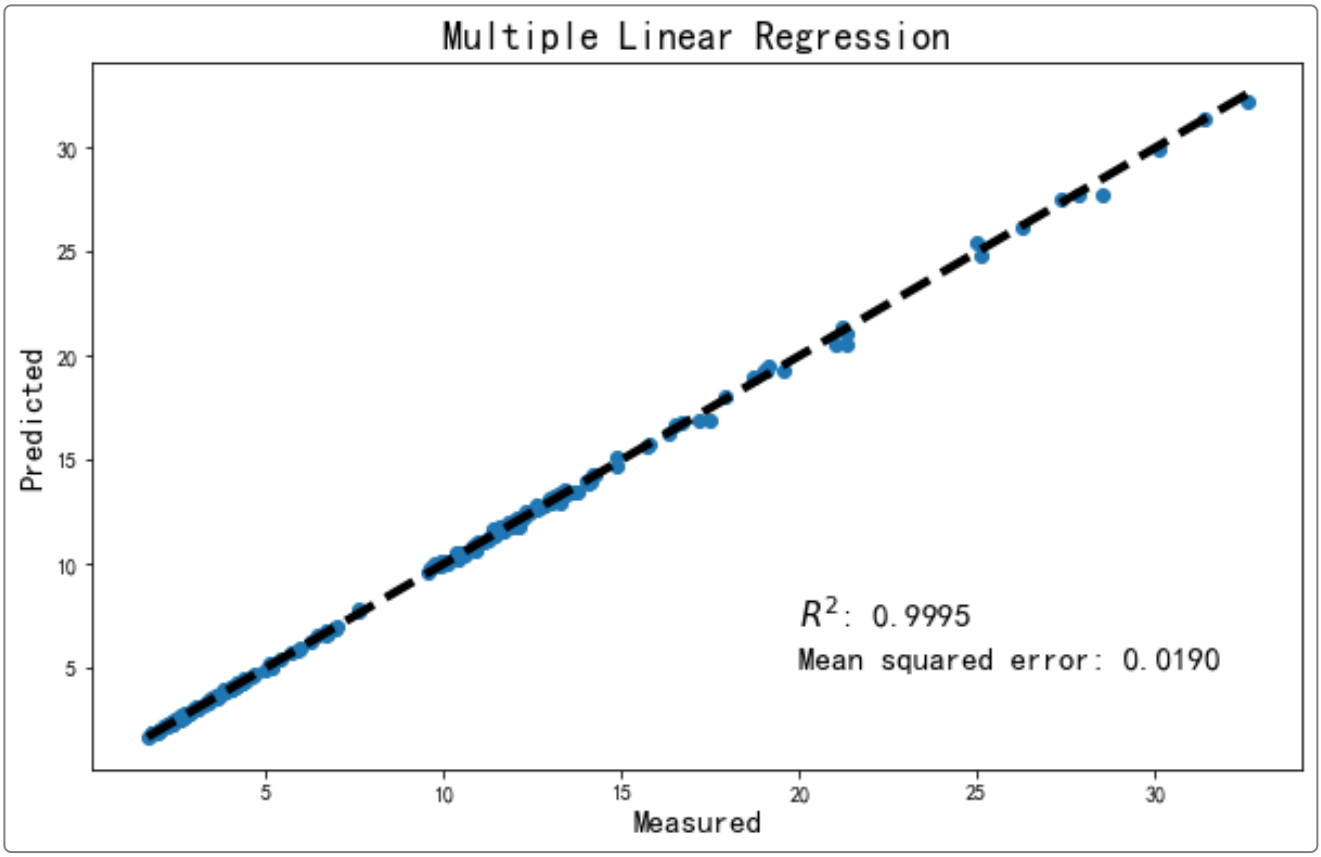
模型评价

```
from sklearn import metrics
print("Explained Variance Score: ",
    metrics.explained_variance_score(y_test, y_pred))
print('Mean Absolute Error:',
    metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:',
    metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:',
    np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Explained Variance Score:  0.99953658296
Mean Absolute Error: 0.083477153166
Mean Squared Error: 0.0189969604490
Root Mean Squared Error: 0.137829461469
```

预测结果可视化

x 轴是观测时(Measured) y 轴是预测值(Predicted)，此处绘制他们的线性关系图，看此时多元线性回归模型预测效果。



Statsmodels中的多元线性回归

```
# 接口一
import statsmodels.api as sm
X = sm.add_constant(X) # 加一个常数
model = sm.OLS(Y, X).fit()
predictions = model.predict(X)
print_model = model.summary()
model.rsquared
# 接口二
from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm
model = ols("Close ~ Open + High + Low", dataset).fit()
# 输出概要
print(model.summary())
print("\nRetrieving manually the parameter estimates:")
print(model._results.params)
# 对拟合线性模型进行方差分析
anova_results = anova_lm(model)
print('\nANOVA results')
print(anova_results)
```

模型评价

```
from sklearn.metrics import explained_variance_score, mean_absolute_error, mean_squared_error, r2_
ex_var_score = explained_variance_score(Y_test, y_pred)
m_absolute_error = mean_absolute_error(Y_test, y_pred)
m_squared_error = mean_squared_error(Y_test, y_pred)
r_2_score = r2_score(Y_test, y_pred)
print("Explained Variance Score: "+str(ex_var_score))
print("Mean Absolute Error "+str(m_absolute_error))
print("Mean Squared Error "+str(m_squared_error))
print("R Squared Error "+str(r_2_score))
```

```
Explained Variance Score: -0.01849265642
Mean Absolute Error 3.158291415
Mean Squared Error 33.27808014
R Squared Error -0.4544513401
```

如果想了解线性回归模型完整原理可参见简单而强大的线性回归详解。

岭回归

即使最小二乘估计是无偏的，它们的方差很大，但因多重共线性的存在，它们可能离真实值很远。**岭回归**是一种分析多重共线性的多元回归的技术。岭回归也称为吉洪诺夫正则化。

该模型求解的回归模型的损失函数为线性最小二乘函数，正则化采用`l2-范数`。这个估计器内置了对多变量回归的支持(即，当y是形状的`2d数组(n_samples, n_targets)`)。

最小化目标函数: $||y - Xw||_2^2 + \alpha||w||_2^2$

> 正则化是为了解决训练数据的过拟合问题，即模型对训练数据表现良好，而在验证测试数据上表现很差。此外，正则化通过在目标函数中增加一个罚项来解决问题，并利用罚项来控制模型的复杂性。
>
> 在正则化中有两个损失函数:
>
> `L1损失函数或L1正则化`是通过在系数绝对值和上添加惩罚项来最小化目标函数。这叫做最小绝对偏差法。
>
> `L2损失函数或L2正则化`是通过在系数平方和上添加惩罚项来最小化目标函数。

同样，线性回归中的多重共线性与岭回归 中已详细介绍了岭回归模型。

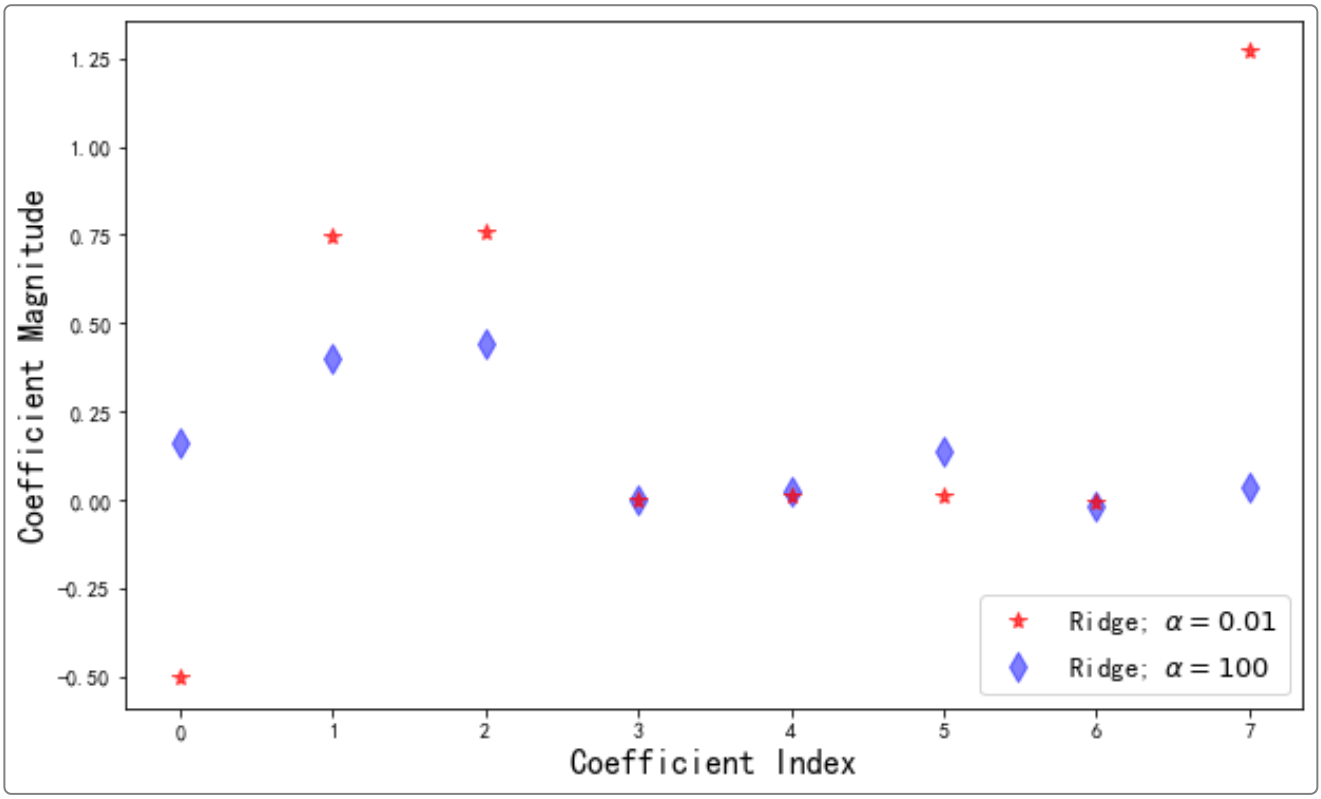
```
X = dataset.drop(['Adj Close', 'Close'], axis=1)
y = dataset['Adj Close'].values.reshape(-1,1)
from sklearn.linear_model import Ridge, LinearRegression
X_train,X_test,y_train,y_test=train_test_split(
      X,y,test_size=0.3,random_state=3)
```

α值越高，对系数的限制越多;
α低则对系数几乎没有限制，其泛化能力更强，在这种情况下，线性回归和岭回归类似。

- 欠拟合，则降低α值
- 过拟合，则增加α值

不同的α对系数的影响

较小α(红色五角星)与较大α(蓝色菱形)所得到的回归模型系数对比图。在岭回归中，再大的α是将系数无限逼接近于零但不等于零，这是Lasso不一样的地方。




```
rr = Ridge(alpha=0.01)
rr.fit(X_train, y_train)
print('Coefficients:', rr.coef_)
```

```
Coefficients:
      [[-5.01889592e-01   7.48981360e-01
         7.56617616e-01  -1.84156278e-11
         1.01448011e-02   1.11935174e-02
        -7.88132703e-03   1.26942083e+00]]
```

Lasso回归

套索回归(Least Absolute Shrinkage and Selection Operator)是一种既进行变量选择又进行正则化的方法，以提高其生成的统计模型的预测精度和可解释性。

Lasso的优化目标为:

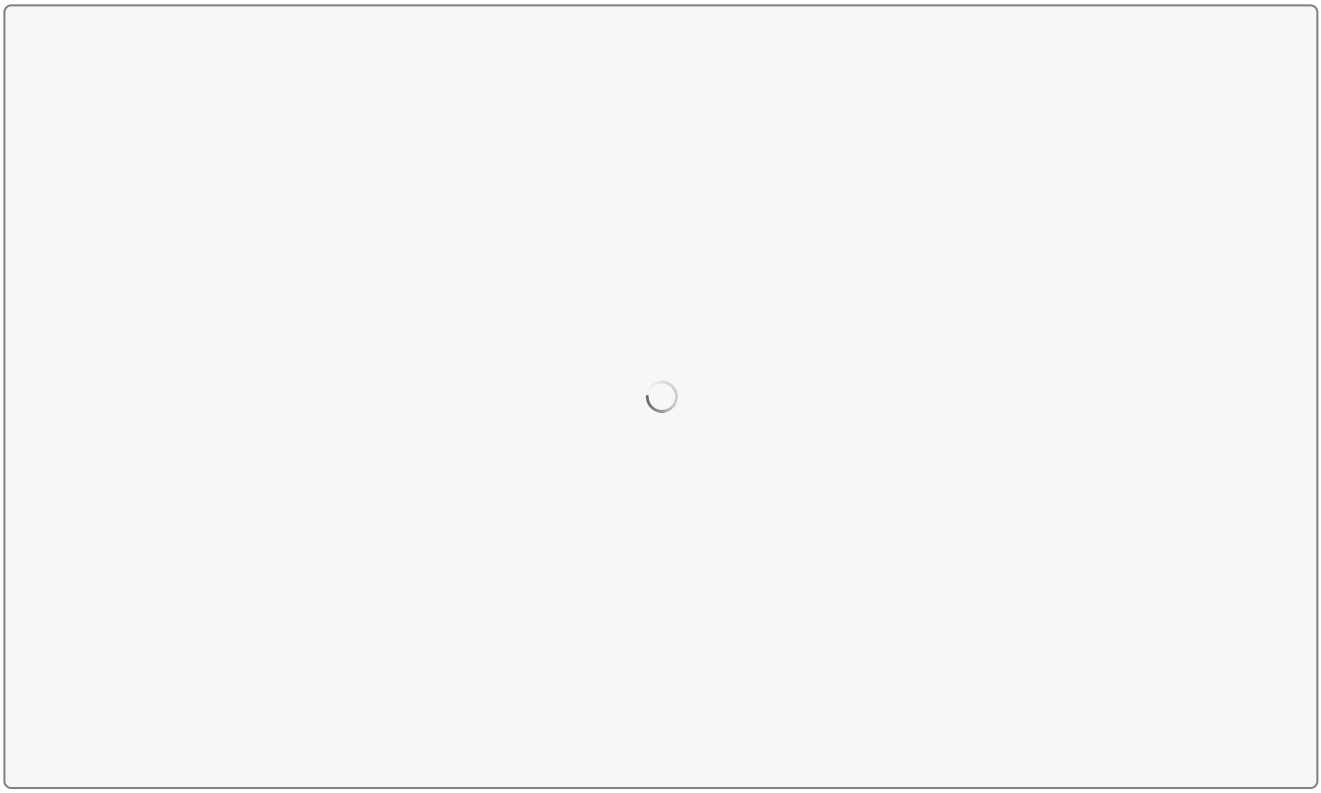
此处不做过多原理部分阐述，详情可参见深度理解Lasso回归分析。

```
X = dataset[['Open','High', 'Low']]
Y = dataset['Adj Close']
from sklearn.linear_model import Lasso
X_train,X_test,y_train,y_test=train_test_split(
      X,Y, test_size=0.3, random_state=31)
```

不同的α对系数的影响

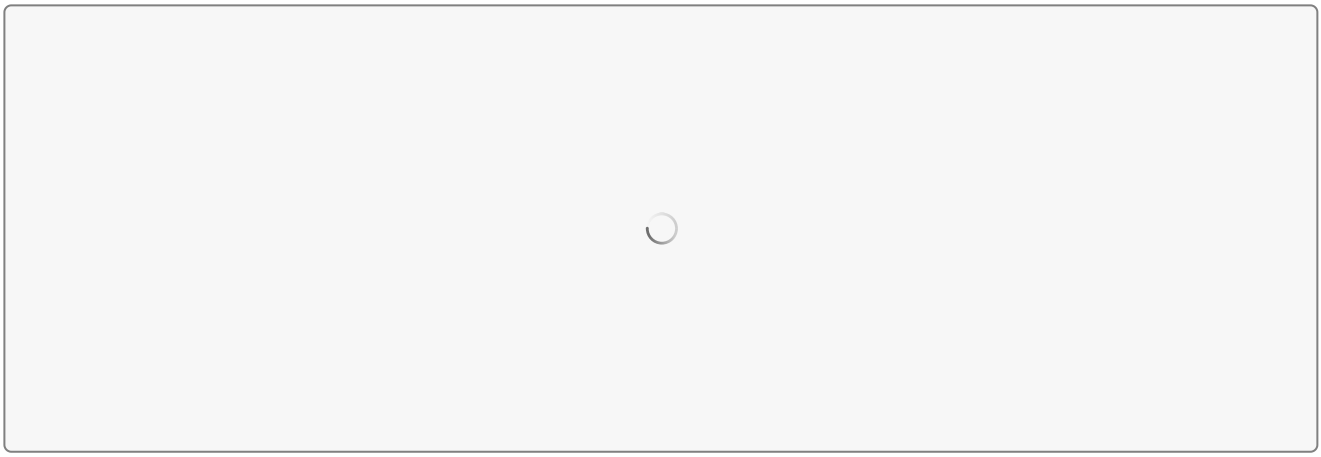
图中当α等于1(红色五角星)时，其已经将三个变量中的其中一个变量系数直接压缩为0，另一个已经接近于0。

这样就可以做特征选择：系数不为零的那个特征所包含的信息就能囊括其余两个系数为零的特征所包含的信息了。



α特征选择

```
scaler = StandardScaler()
X = scaler.fit_transform(dataset)
y = dataset["Adj Close"]
# 创建lasso函数
def lasso(alphas):
    '''
    接受一个alphas列表。
    输出包含每个alpha的Lasso回归系数的DataFrame。
    '''
    df = pd.DataFrame()
    df['Feature Name'] = dataset.columns
    # 循环获取列表中的每个alpha值
    for alpha in alphas:
        lasso = Lasso(alpha=alpha)  # 用alpha值创建一个Lasso回归
        lasso.fit(X, y)  # 训练lasso 回归
        # 为alpha值创建一个列名
        column_name = 'Alpha = %f' % alpha
        # 创建一个列保存系数值
        df[column_name] = lasso.coef_
    return df   # 返回DataFrame
# 运行Lasso函数
lasso([.0001, .01, .1, .5, 1, 10])
```



弹性网络回归

弹性网络回归（ElasticNet Regression）是对岭回归和Lasso回归的融合，其惩罚项是对 `L1范数` 和 `L2范数` 的一个权衡。方法是先将有共线性的自变量分成一组，如果其中有一个自变量与因变量有强相关关系，那么就将这一组所有自变量都输入线性模型。

看下三个带正则项回归的正则化项：

1. 岭回归 Ridge Regression:正则化项为$\alpha||\vec{w}||_2^2, \alpha \geq 0$。
2. LASSO回归 Lasso Regression:正则化项为$\alpha||\vec{w}||_1, \alpha \geq 0$。

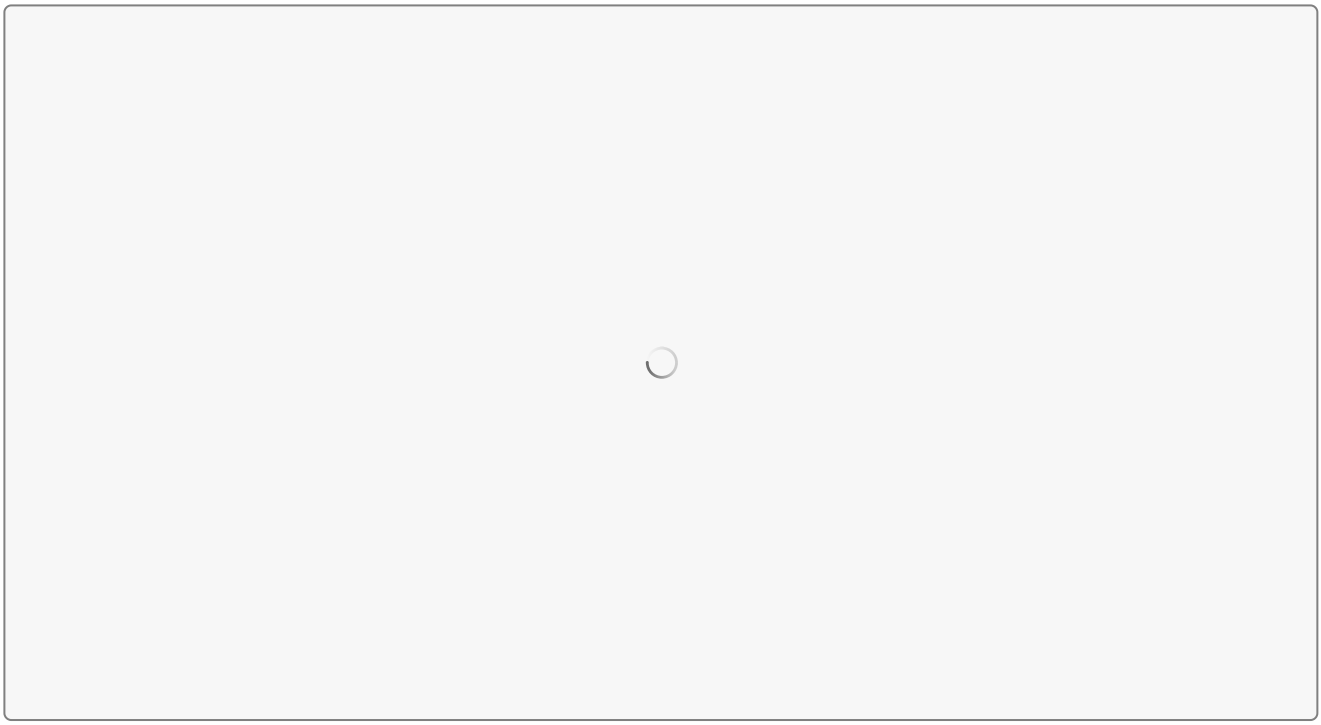
3. 弹性网络回归 Elastic Net:正则化项为 。

建立模型

```
from sklearn.linear_model import ElasticNet
X = dataset[['Open', 'High', 'Low', 'Volume']]
Y = dataset['Adj Close']
X_train, X_test, Y_train, Y_test = train_test_split(
      X, Y, test_size = 0.2, random_state = 0)
enet = ElasticNet(random_state=0)
y_pred_enet = enet.fit(X_train, Y_train).predict(X_test)
r2_score_enet = r2_score(Y_test, y_pred_enet)
```

系数可视化

由于弹性网络中 `L1` 正则化可将系数缩减到0，因此该例子中第四个系数被缩减到0。



贝叶斯岭回归

贝叶斯岭回归(Bayesian Ridge Regression)的引入主要是在最大似然估计中很难决定模型的复杂程度，Ridge回归加入的惩罚参数其实也是解决这个问题的，同时可以采用的方法还有对数据进行正规化处理，另一个可以解决此问题的方法就是采用贝叶斯方法。

```
X = dataset['Open'].values.reshape(1171,-1)
y = dataset['Adj Close'].values.reshape(1171,-1)
from sklearn.linear_model import BayesianRidge, LinearRegression
X = dataset['Open'].values.reshape(1258,-1)
y = dataset['Adj Close'].values.reshape(1258,-1)
X_train, X_test, y_train, y_test = train_test_split(
      X, y, test_size = 0.2, random_state = 0)
```

模型训练与评分

```
# 拟合贝叶斯岭回归和OLS以比较
model_br = BayesianRidge(compute_score=True)
model_lr = LinearRegression()
model_br.fit(X_train, y_train)
model_lr.fit(X_train, y_train)
model_br.coef_
model_lr.coef_
model_br.scores_
model_lr.score(X_test, y_test)
y_pred = model.predict(X_test)
from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test, y_pred) ** 0.5)
print(model.score(X_test, y_test))
```

最小回归角回归

最小回归角回归(Least-Angled Regression (LARS))算法的过程大致如下：

1. 首先，像传统的前向选择一样，将所有系数 $\hat{\beta}_j$ 置为0，然后选择一个与响应值相关度最大的变量（如x_{j1}），并在这个方向上前进尽可能大的一步（增大/小系数 $\hat{\beta}_{j1}$ ），直到另一个变量（如x_{j2}），与目前的残差有同样大的相关度。
2. 此时LARS算法和前向选择分道扬镳。前向选择中继续沿x_{j1}方向前进，而LARS选择 x_{j1} 与 x_{j2} 的角平分线方向前进（即同时等量增大/小 $\hat{\beta}_{j1}$ 和 $\hat{\beta}_{j2}$ ），直到第三个变量 x_{j3} 达到相关度的要求，进入到这个"最相关"的集合中，然后再沿这三个变量的角平分线方向前进（同时等量增大/小 $\hat{\beta}_{j1}$ $\hat{\beta}_{j2}$ 和 $\hat{\beta}_{j3}$ ）。
3. 依次类推。

最小角回归法是一个适用于高维数据的回归算法，其主要的优点有：

- 特别适合于特征维度n 远高于样本数m的情况。
- 算法的最坏计算复杂度和最小二乘法类似，但是其计算速度几乎和前向选择算法一样
- 可以产生分段线性结果的完整路径，这在模型的交叉验证中极为有用。

主要的缺点是：

- 由于LARS的迭代方向是根据目标的残差而定，所以该算法对样本的噪声极为敏感。

```
X = np.array(dataset['Returns']).reshape(1, -1)
y = np.array(dataset['Adj Close']).reshape(1, -1)
from sklearn.linear_model import Lars
reg = Lars(n_nonzero_coefs=1)
reg.fit(X, y)
print(reg.coef_)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

偏最小二乘法回归

偏最小二乘回归(Partial least squares regression， PLS回归)是一种统计学方法，与主成分回归有关系，但不是寻找响应和独立变量之间最小方差的超平面，而是通过投影预测变量和观测变量到一个新空间来寻找一个线性回归模型。

偏最小二乘回归提供一种多对多线性回归建模的方法，特别当两组变量的个数很多，且都存在多重相关性，而观测数据的数量又较少时，**甚至比变量的维度还少**，用**偏最小二乘回归**建立的模型具有传统的经典回归分析等方法所没有的优点。

偏最小二乘回归分析在建模过程中**集中了主成分分析，典型相关分析和线性回归分析**方法的特点，因此在分析结果中，除了可以提供一个更为合理的回归模型外，还可以同时完成一些类似于主成分分析和典型相关分析的研究内容，提供更丰富、深入的一些信息。

```
X = np.asarray(dataset[['Open', 'High', 'Low',
                        'Adj Close', 'Volume']])
y = np.asarray(dataset['Buy_Sell'])
X = preprocessing.StandardScaler().fit(X).transform(X)
X_train, X_test, y_train, y_test = train_test_split(
```

```
        X, y, test_size = 0.25, random_state = 0)

from sklearn.cross_decomposition import PLSRegression
pls = PLSRegression(n_components=2)
pls.fit(X_train, y_train)
y_pred = pls.predict(X_test)
pls.coef_
```

```
array([[ 0.0027826 ],
       [ 0.00032104],
       [-0.00116041],
       [-0.00329928],
       [ 0.01212888]])
```

分位数回归

分位数回归是统计和计量经济学中使用的一种回归分析。而最小二乘法估计条件均值跨预测变量的值的响应变量的，位数回归估计条件中值(或其它位数的响应可变的)。分位数回归是在不满足线性回归条件时使用的线性回归的扩展。

分位数回归是估计一组回归变量X与被解释变量Y的分位数之间线性关系的建模方法。

OLS回归估计量的计算是基于最小化残差平方。

分位数回归估计量的计算也是基于一种非对称形式的绝对值残差最小化。其中，中位数回归运用的是最小绝对值离差估计。

分位数回归的优点

1. 能够更加全面的描述被解释变量条件分布的全貌，而不是仅仅分析被解释变量的条件期望(均值)，也可以分析解释变量如何影响被解释变量的中位数、分位数等。
2. 不同分位数下的回归系数估计量常常不同，即解释变量对不同水平被解释变量的影响不同。
3. 中位数回归的估计方法与最小二乘法相比，估计结果对离群值则表现的更加稳健，而且，分位数回归对误差项并不要求很强的假设条件，因此对于非正态分布而言，分位数回归系数估计量则更加稳健。

当数据中存在异常值、高偏态和异方差时，使用这种回归。

statsmodels中的两种方法

方法一：QuantReg

```
X = dataset['Open']
Y = dataset['Adj Close']
from statsmodels.regression.quantile_regression import QuantReg
quant = QuantReg(X, Y)
qt = quant.fit(q=0.55)
print(qt.summary())
```

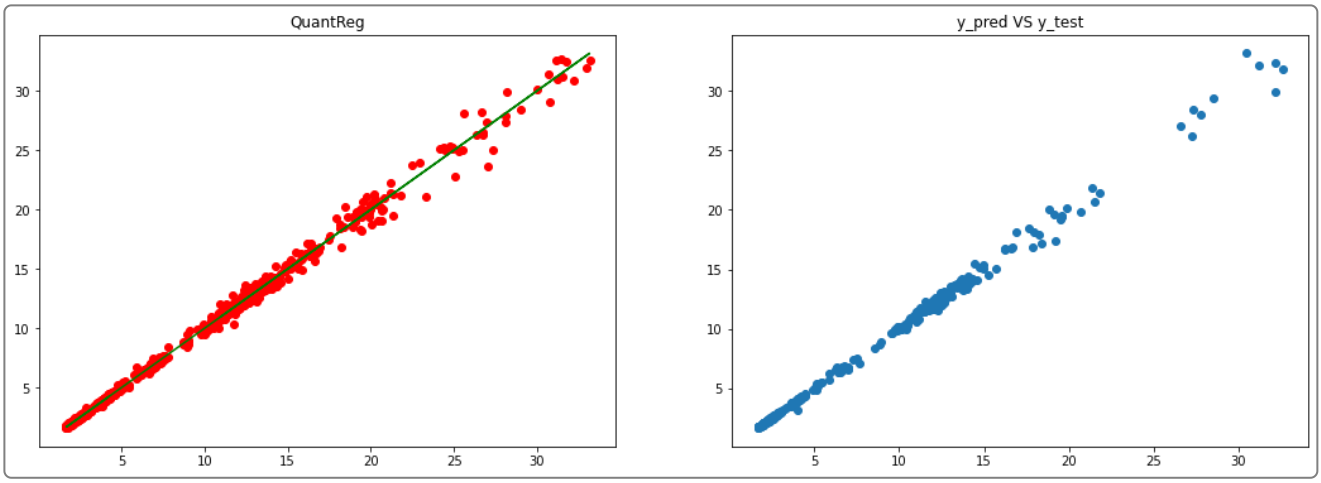
方法二：smf

```
import statsmodels.formula.api as smf
mod = smf.quantreg('Close ~ Open + High + Low', dataset)
res = mod.fit(q=.5)
print(res.summary())
```

```
                         QuantReg Regression Results
==============================================================================
Dep. Variable:                  Close   Pseudo R-squared:               0.9842
Model:                       QuantReg   Bandwidth:                     0.02594
Method:                 Least Squares   Sparsity:                       0.1293
Date:                Tue, 27 Apr 2021   No. Observations:                 1259
Time:                        21:49:35   Df Residuals:                     1255
                                        Df Model:                            3
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.0098      0.003     -3.263      0.001      -0.016      -0.004
Open          -0.6677      0.009    -76.398      0.000      -0.685      -0.651
High           0.8556      0.009     98.847      0.000       0.839       0.873
Low            0.8132      0.008    103.072      0.000       0.798       0.829
==============================================================================
```

分位数回归可视化

分位数回归真实值与拟合直线(左图),及预测值与真实值散点图(右图)，从两张图可以看出模型拟合效果还是不错的。



分位回归与线性回归的比较

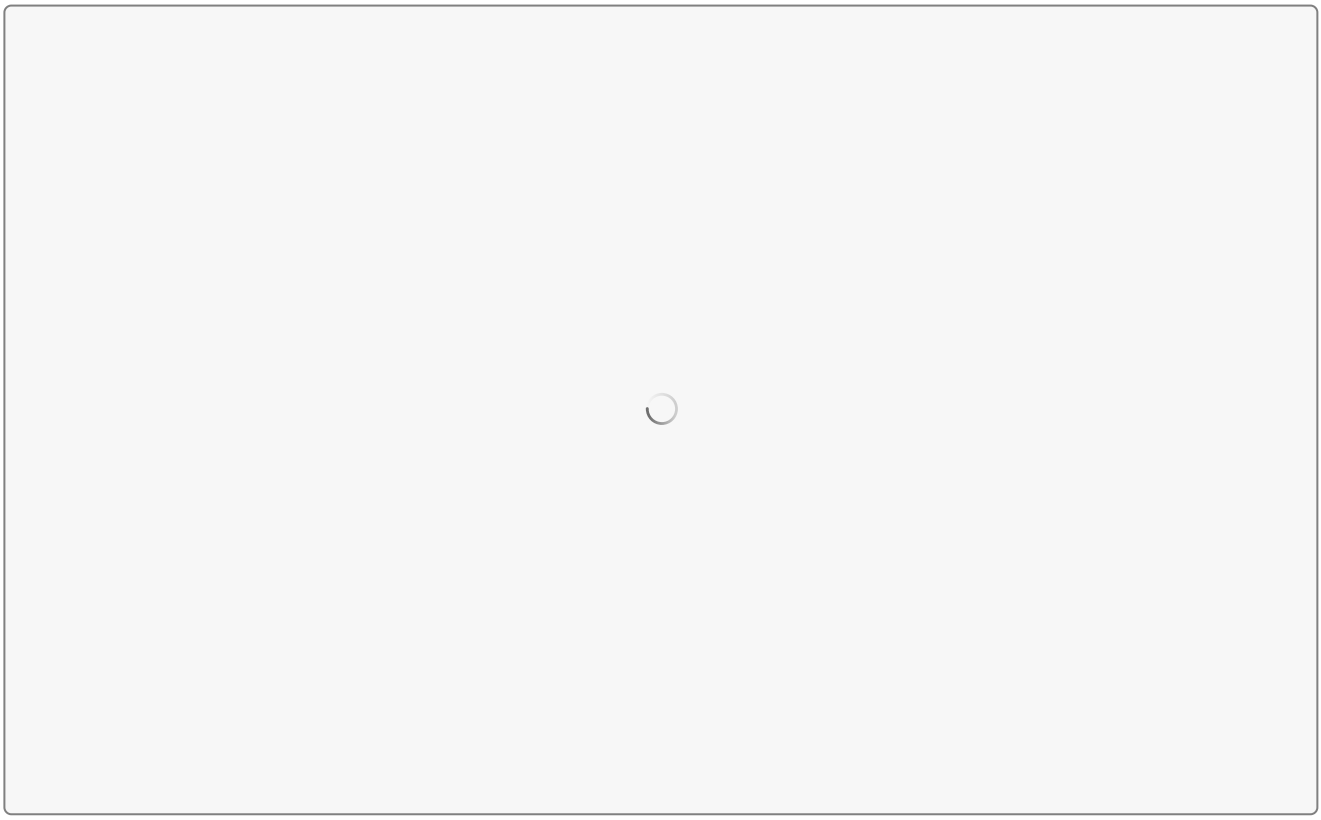
不同分位数值拟合模型的一些属性值

```
mod = smf.quantreg('Close ~ Open', dataset)
res = mod.fit(q=.5)
quantiles = np.arange(.05, .86, .2)
def fit_model(q):
    res = mod.fit(q=q)
    return [q, res.params['Intercept'], res.params['Open']] + \
            res.conf_int().loc['Open'].tolist()
# res.conf_int() 计算拟合参数的置信区间。
models = [fit_model(x) for x in quantiles]
models = pd.DataFrame(models, columns=['分位数', '截距项', '系数','置信区间下限','置信区间上限'])

ols = smf.ols('Close ~ Open', dataset).fit()
ols_ci = ols.conf_int().loc['Open'].tolist()
ols = dict(截距项 = ols.params['Intercept'],
           系数 = ols.params['Open'],
           置信区间下限 = ols_ci[0],
           置信区间上限 = ols_ci[1])

print(models)
print(ols)
```

```
   分位数     截距项       系数     置信区间下限   置信区间上限
0  0.05  0.053096  0.938381  0.936103  0.940660
1  0.25  0.018050  0.979016  0.977595  0.980438
2  0.45  0.007944  0.995150  0.993928  0.996372
3  0.65 -0.008459  1.010540  1.009271  1.011810
4  0.85 -0.018962  1.030739  1.028971  1.032508
{'截距项': 0.01995, '系数': 0.99734,
 '置信区间下限': 0.99417, '置信区间上限': 1.00051}
```

将不同分位数拟合模型回归可视化

对该数据同时进行最小二乘法回归(得到条件均值的方程)和分位数回归(得到10个条件p分位数方程，p的取值为 `5%，15%，……，95%`)如下图所示。



从这个图中可以观察到以下结论：

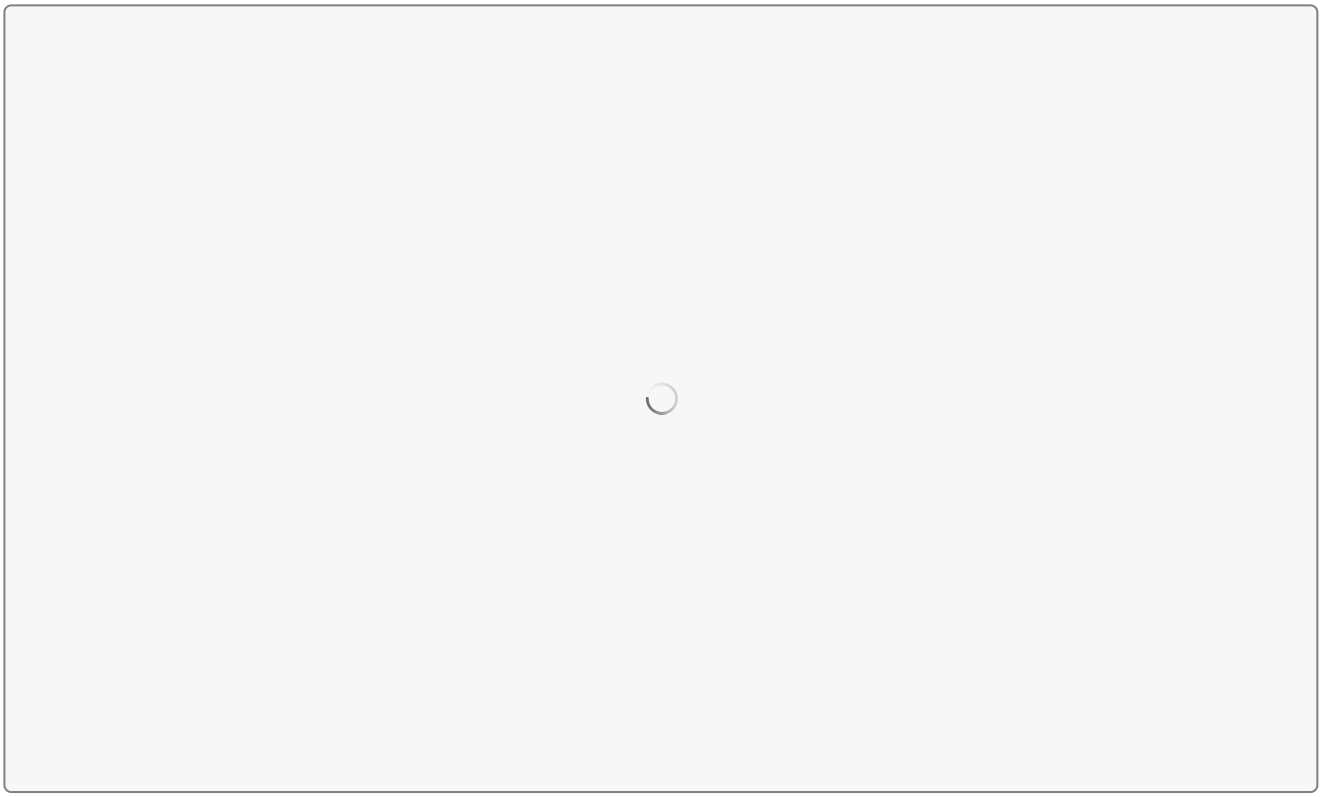
- 收盘价随开盘价而增加；
- 收盘价的分布随开盘价增加变得越来越宽(高分位数和低分位数之间的间隔越来越大)；
- 最小二乘法回归对于低开盘价对应的观测点的拟合度较差；
- 从图中可见，最小二乘法的红色曲线处于很多低开盘价观测点之上。

不同分位数对应的回归系数

上述分位数回归的结果说明，在收盘价分布的不同位置(不同分位数)，开盘价对其的影响是不同的。下图展示了这一点。

图中横坐标为收盘价的分位数，纵坐标为不同分位数回归的系数 ，它表示一个单位的开盘价变化带来多大的收盘价。

对于最小二乘法(红色)来说，它假设开盘价对收盘价的影响在整个分布上是恒定的；但是分位数回归(黑色)正好得到不同的结论。显然，分位数回归提供了开盘价和收盘价之间更为丰富的关系。



对于金融投资中的很多变量，比如收益率，我们往往更关心它在分布尾部的特性。在这方面，分位数回归是一个有力的工具，它让我们研究收益率和不同的解释变量在全分布上的相关性。

当变量的分布明显偏离正态分布或者存在异常值时，传统的最小二乘法回归就不那么有效了。然而分位数回归不受这些弊端的影响。此外，分位数回归满足单调变换不变性。

往期精彩

简单而强大的线性回归详解

2020-11-04

线性回归中的多重共线性与岭回归

2020-11-08

深度理解Lasso回归分析

2020-11-13

数据预处理 | 数据标准化及归一化

2020-10-21

机器学习中样本不平衡，怎么办？

2020-10-18

长按👇关注- 数据STUDIO - 选择星标，干货速递

收录于合集 #机器学习 110

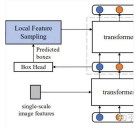
‹ 上一篇
机器学习｜关联规则与购物篮分析实战

下一篇 ›
几种特征选择方法的比较，孰好孰坏？

喜欢此内容的人还喜欢

辞旧迎新-2023兔年大吉
R语言数据分析指南

ECCV｜数据高效的Transformer目标检测器
计算机视觉研究院



【Part2_交通网络分析篇】12.公交行驶线路计算及覆盖度分析
码农设计师