In [ ]:

```
sm.stats.diagnostic.acorr_ljungbox(lmres.resid)
```

In [ ]:

```
sm.stats.diagnostic.acorr_ljungbox(lmres.resid, lags = [1,2,3,4])
```

4.5.2 自回归

class statsmodels.tsa.arima_model.ARIMA(

endog : 时间序列变量
order = (0, 0, 0) : (p,d,q)三个参数分别代表自回归、差分、移动平均
exog : optional, 自变量列表
dates : , optional, 日期时间变量
freq : 时间序列数据的间隔频率

)

In [ ]:

```
from statsmodels.tsa.arima_model import ARIMA

model = ARIMA(lmres.resid, order=(2, 0, 0))
results_AR = model.fit()
```

In [ ]:

```
results_AR.summary()
```

In [ ]:

```
res2 = ARIMA(lmres.resid, order=(1, 0, 0)).fit()
res2.summary()
```

In [ ]:

```
sm.stats.diagnostic.acorr_ljungbox(res2.resid)
```

In [ ]:

```
ARIMA(dfgdp.gdp, (1, 0, 0), dfgdp.iloc[:, [1,2,3]]).fit().summary()
```

4.6 实战练习

使用本章中学到的知识，尝试对boston数据所建立的回归方程进行全面的回归诊断。

尝试对boston数据拟合稳健回归模型、岭回归模型，比较分析结果和普通回归模型的差异。

5 logitsic回归

5.1 logistic回归模型的基本原理

5.2 两分类因变量logistic回归的实现

class statsmodels.discrete.discrete_model.Logit(

```
    endog
    exog
    missing = 'none' : 'none', 'drop', and 'raise'
```

)

logit类的方法（部分）：

```
    cdf(X)    The logistic cumulative distribution function
    fit([start_params, method, maxiter, …])
    fit_regularized([start_params, method, …])
    information(params)    Fisher information matrix of model
    loglike(params)    Log-likelihood of logit model.
    loglikeobs(params)    Log-likelihood of logit model for each obs.
    pdf(X)    The logistic probability density function
    predict(params[, exog, linear])
    score(params)    Logit model score vector of the log-likelihood
    score_obs(params)
```

In [ ]:

```
dflogit = pd.read_excel('dmdata.xlsx', sheet_name = 'logit')
dflogit.head()
```

In [ ]:

```
from statsmodels.formula.api import logit

# 使用公式方式定义模型
logitmodel = logit('低出生体重儿 ~ 产妇年龄 + 产妇体重 + C(种族) + 妊娠期间是否吸烟\
                    + 本次妊娠前早产次数 + 是否患有高血压 + 应激性 + 随访次数',
                    dflogit)
logitres = logitmodel.fit()
```

In [ ]:

```
logitres.summary()
```

In [ ]:

```
logitres.prsquared
```

In [ ]:

```
logitres.params
```

In [ ]:

```
paradf = pd.DataFrame(logitres.params, columns = ['b'])
paradf['expb'] = np.exp(paradf.b)
paradf
```

In [ ]:

```
logitres.conf_int()
```

In [ ]:

```
# 给出当前模型对于样本的预测表格
logitres.pred_table()
```

In [ ]:

```
dflogit.低出生体重儿.value_counts()
```

5.3 模型中的各种检验方法

In [ ]:

```
logitres.llnull, logitres.llf
```

In [ ]:

```
logitres.llr, logitres.llr_pvalue
```

使用似然比检验进行变量筛选

In [ ]:

```
logitres2 = logit('低出生体重儿 ~ 产妇年龄 + 产妇体重 + 妊娠期间是否吸烟\
                   + 本次妊娠前早产次数 + 是否患有高血压 + 应激性 + 随访次数',
                   dflogit).fit()
```

In [ ]:

```
logitres2.llf
```

In [ ]:

```
2 * (logitres.llf - logitres2.llf)
```

In [ ]:

```
import scipy.stats as ss

1- ss.chi2.cdf(2 * (logitres.llf - logitres2.llf), 2)
```

5.4 哑变量

In [ ]:

```
dfdumm = pd.get_dummies(dflogit.种族)
dfdumm.head()
```

In [ ]:

```
tempX = pd.merge(left = dflogit.drop(columns =['id', '出生体重(克)',
                                               '低出生体重儿', '种族']),
         right = dfdumm[['其他种族', '黑人']],
         left_index = True, right_index = True
        )
logitres = sm.Logit(dflogit.低出生体重儿, tempX).fit()
logitres.summary()
```

In [ ]:

```
# 考虑能否将非白人种族合并建模
tempX2 = pd.merge(left = dflogit.drop(columns =['id', '出生体重(克)',
                                               '低出生体重儿', '种族']),
         right = dfdumm[['白人']],
         left_index = True, right_index = True
        )
logitres2 = sm.Logit(dflogit.低出生体重儿, tempX2).fit()
logitres2.summary()
```

In [ ]:

```
import scipy.stats as ss

1- ss.chi2.cdf(2 * (logitres.llf - logitres2.llf), 1)
```

5.5 多分类因变量logistic回归

class statsmodels.discrete.discrete_model.MNLogit(endog, exog, missing = 'none')

In [ ]:

```
from sklearn.datasets import load_iris
iris = load_iris()

irisdf = pd.DataFrame(iris.data, columns = iris.feature_names)
irisdf.head()
```

In [ ]:

```
# 只使用花萼宽进行模型拟合
mnlogitres = sm.MNLogit(iris.target,
                        sm.add_constant(irisdf.iloc[:,[1]])).fit()
```

In [ ]:

```
# 模型以第一个类别作为参照类
mnlogitres.params
```

In [ ]:

```
mnlogitres.pvalues
```

In [ ]:

```
mnlogitres.summary()
```

In [ ]:

```
# 类别间差异过于明显，模型无法正常拟合
mnlogitres = sm.MNLogit(iris.target, sm.add_constant(iris.data)).fit()
```

In [ ]:

```
mnlogitres.summary()
```

In [ ]:

```
# 使用带正则惩罚的拟合方法
mnlogitres = sm.MNLogit(iris.target,
                        sm.add_constant(iris.data) ).fit_regularized()
```

In [ ]:

```
mnlogitres.summary()
```

5.6 logistic回归的sklearn实现

class sklearn.linear_model.LogisticRegression(

```
    penalty = 'l2', dual = False, tol = 0.0001, C = 1.0
    fit_intercept = True, intercept_scaling = 1
    class_weight = None : dict or 'balanced', 各类的权重
         权重以{class_label: weight}形式提供，None时默认均为1
         'balanced' :  权重和频次成反比，样本量/(类别数*np.bincount(y))
    random_state = None
    solver = 'liblinear' : 具体的拟合方法
        {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}
        'liblinear' 适用于小数据集，'sag'和'saga'针对大数据集拟合速度更快
        多分类目标变量只能使用'newton-cg', 'sag', 'saga'和'lbfgs'拟合
        'newton-cg', 'lbfgs'和'sag'只能使用L2正则化
        'liblinear'和'saga'则可以处理L1正则化

    max_iter = 100, multi_class = 'ovr', verbose = 0
    warm_start = False, n_jobs = 1
```

)

LogisticRegression类的属性：

```
    coef_ : array, shape (1, n_features) or (n_classes, n_features)
    intercept_ : array, shape (1,) or (n_classes,)
    n_iter_ : array, shape (n_classes,) or (1, )
```

LogisticRegression类的方法：

```
decision_function(X) : Predict confidence scores for samples.
densify() : Convert coefficient matrix to dense array format.
fit(X, y[, sample_weight])
get_params([deep]) : Get parameters for this estimator.
predict(X) : Predict class labels for samples in X.
predict_log_proba(X) : 对数概率估计
predict_proba(X) : 概率估计
score(X, y[, sample_weight]) : 返回给定测试集类别预测的平均准确度
set_params(**params) : Set the parameters of this estimator.
sparsify() : Convert coefficient matrix to sparse format.
```

两分类因变量的情形

In [ ]:

```
from sklearn.preprocessing import binarize

tmpy = binarize(iris.target.reshape(-1, 1))
```

In [ ]:

```
tmpy[:10]
```

In [ ]:

```
from sklearn import linear_model

reg = linear_model.LogisticRegression()

reg.fit(iris.data, tmpy)
```

In [ ]:

```
reg.intercept_, reg.coef_
```

In [ ]:

```
reg.predict_proba(iris.data)[:10]
```

In [ ]:

```
reg.predict(iris.data)[:10]
```

In [ ]:

```
reg.score(iris.data, tmpy)
```

In [ ]:

```
from sklearn.metrics import classification_report

print(classification_report(tmpy, reg.predict(iris.data)))
```

多分类因变量的情形

sklearn可以做到模型的正则拟合，但模型架构和一般介绍的形式有所差异。

In [ ]:

```
from sklearn import linear_model

reg = linear_model.LogisticRegression()

reg.fit(iris.data, iris.target)
```

In [ ]:

```
reg.intercept_, reg.coef_
```

In [ ]:

```
print(classification_report(iris.target, reg.predict(iris.data)))
```

In [ ]:

```
from sklearn import linear_model

reg = linear_model.LogisticRegression(solver = 'newton-cg',
                                      multi_class = 'multinomial')

reg.fit(iris.data, iris.target)
```

In [ ]:

```
reg.intercept_, reg.coef_
```

In [ ]:

```
print(classification_report(iris.target, reg.predict(iris.data)))
```

5.7 实战练习

尝试使用似然比检验对logit表单数据进行变量筛选，得到最终的logistic回归模型。

6 决策树模型

6.1 树模型的基本原理

6.2 各种树模型算法

6.3 树模型的sklearn实现

6.3.1 拟合决策树模型

class sklearn.tree.DecisionTreeClassifier(