In [ ]:

```
preprocessing.scale(boston.data)[:2]
```

模型的保存（持久化）

可以直接使用通过使用Python的pickle模块将训练好的模型保存为外部文件，但最好使用sklearn中的joblib模块进行操作。

In [ ]:

```
# 保存为外部文件
from sklearn.externals import joblib

joblib.dump(std, 'f:/std.pkl')
joblib.dump(reg, 'f:/reg.pkl')
```

In [ ]:

```
# 读入外部保存的模型文件
reg2 = joblib.load('f:/reg.pkl')
reg2.coef_
```

1.6 实战练习

尝试使用不同的三种调用方式调用同一个模型。

加载sklearn自带的iris数据集，熟悉该数据集的各种属性，并尝试将其转换为数据框。

尝试在不参考任何帮助文档的情况下，按照sklearn中的标准API操作方式，使用BP神经网络对iris数据进行拟合，并返回各案例的预测类别、预测概率等结果。

> BP神经网络对应的类为：class sklearn.neural_network.MLPClassifier()
> 此处只为API操作演示，不进一步讨论模型拟合前的数据预处理问题

将上题中生成的模型存储为外部文件，并重新读入。

2 方差分析模型

2.1 一般线性模型概述

2.2 方差分析模型的statsmodels实现

statsmodles对方差分析模型的实现，本质上是对通用的一般线性模型框架做了重新的结果解读。

拟合方差分析模型时，使用模型表达式会更为灵活便捷。

patsy包可以提供对模型表达式更好的支持，这里不展开讨论。

2.2.1 方差分析模型的实现

In [ ]:

```
ccss = pd.read_excel('ccss_sample.xlsx', sheet_name = 'CCSS')

ccss.head()
```

In [ ]:

```
# 对数据做图形化观察
sns.pointplot(ccss.s0, ccss.index1)
```

In [ ]:

```
from statsmodels.formula.api import ols

# 使用公式方式定义模型
lmmodel = ols('index1 ~ C(s0)', ccss)
lmres = lmmodel.fit()
type(lmres) # 注意lmres的类型
```

In [ ]:

```
lmres.model.formula
```

In [ ]:

```
lmres.summary()
```

In [ ]:

```
from statsmodels.stats.anova import anova_lm

# 给出方差分析模型框架下的输出结果
anova_lm(lmres)
```

In [ ]:

```
type(anova_lm(lmres))
```

In [ ]:

```
anova_lm(lmres).F[0]
```

2.2.2 两两比较的实现

statsmodels.sandbox.stats.multicomp包中提供了比较完整的两两比较方法，但使用上比较复杂

直接进行P值校正
直接给出两两比较的结果

statsmodels.sandbox.stats.multicomp.multipletests(

```
    pvals : 类数组格式的原始P值
    alpha = 0.05 : 希望控制的总Alpha水准，FWER
    method = 'hs' : 具体的校正方法，写全称或者可区别的前几个字母均可
        'bonferroni' : one-step correction
        'sidak' : one-step correction

        'holm-sidak' : step down method using Sidak adjustments
        'holm' : step-down method using Bonferroni adjustments

        'simes-hochberg' : step-up method  (independent)
        'hommel' : closed method based on Simes tests (non-negative)
        'fdr_bh' : Benjamini/Hochberg  (non-negative)
        'fdr_by' : Benjamini/Yekutieli (negative)
        'fdr_tsbh' : two stage fdr correction (non-negative)
        'fdr_tsbky' : two stage fdr correction (non-negative)
    is_sorted = 'False' : 是否将结果按照P值升序排列
```

) # 返回值：检验结果、检验P值、alphacSidak、alphacBonf

In [ ]:

```
from statsmodels.sandbox.stats import multicomp as mc

mc.multipletests([0.1, 0.2, 0.3], method = 'b')
```

GroupsStats和MultiComparison命令的输出更接近oneway ANOVA的需求，但是目前尚未完善

In [ ]:

```
poshoc = mc.MultiComparison(ccss.index1, ccss.time)
res = poshoc.tukeyhsd()
res
```

In [ ]:

```
res.summary()
```

scikit_posthocs的实现

功能简单完善，短期内可以考虑作为posthoc输出的工具

pip install scikit_posthocs

scikit_posthocs.posthoc_conover(

```
    data
    val_col =
    group_col =
    p_adjust =
        'bonferroni' : one-step correction
        'sidak' : one-step correction
        'holm-sidak' : step-down method using Sidak adjustments
        'holm' : step-down method using Bonferroni adjustments
        'simes-hochberg' : step-up method  (independent)
        'hommel' : closed method based on Simes tests (non-negative)
        'fdr_bh' : Benjamini/Hochberg  (non-negative)
        'fdr_by' : Benjamini/Yekutieli (negative)
        'fdr_tsbh' : two stage fdr correction (non-negative)
        'fdr_tsbky' : two stage fdr correction (non-negative)
)
```

In [ ]:

```
import scikit_posthocs as sp

pc = sp.posthoc_conover(ccss, val_col='index1', group_col='time',
                        p_adjust = 'bonferroni')
pc
```

In [ ]:

```
# 使用热力图显示比较结果
heatmap_args = {'linewidths': 0.25, 'linecolor': '0.5', 'clip_on':False,
                'square': True, 'cbar_ax_bbox': [0.80, 0.35, 0.04, 0.3]}
sp.sign_plot(pc, **heatmap_args)
```

In [ ]:

```
# 自定义热力图参数
# Format: diagonal, non-significant, p<0.001, p<0.01, p<0.05
cmap = ['1', '#fb6a4a',  '#08306b',  '#4292c6', '#c6dbef']
heatmap_args = {'cmap': cmap, 'linewidths': 0.25, 'linecolor': '0.5',
                'clip_on': False, 'square': True,
                'cbar_ax_bbox': [0.80, 0.35, 0.04, 0.3]}
sp.sign_plot(pc, **heatmap_args)
```

2.3 多因素方差分析模型

2.3.1 交互作用的图形观察

多因素间的交互作用可以使用seaborn中的功能直接绘图观察，也可使用statsmodels提供的功能进行图形考察。

使用seaborn的绘图功能直接考察

In [ ]:

```
# 使用线图+点图方式呈现
sns.pointplot(ccss.time.astype('str'), ccss.index1, hue = ccss.s0, ci = None)
```

In [ ]:

```
# 使用线图方式呈现
sns.lineplot(ccss.time, ccss.index1, hue = ccss.s0, ci = None)
```

In [ ]:

```
# 使用条图形式呈现
sns.barplot(x = ccss.time, y = ccss.index1, hue = ccss.s0)
```

使用interaction_plot函数

statsmodels.graphics.factorplots.interaction_plot(

```
    x, trace, response : x轴变量、分组变量、y轴变量
    func = <function mean> : pandas.DataFrame.aggregate可以使用的任何汇总函数
    ax = None
    plottype = 'b' : {'line', 'scatter', 'both'}, 绘制的图形种类
    xlabel = None, ylabel = None, colors = None, markers = None
    linestyles = None, legendloc = 'best', legendtitle = None, **kwargs
```

)

In [ ]:

```
from statsmodels.graphics.api import interaction_plot

fig = interaction_plot(ccss.time.astype('str'), ccss.s0, ccss.index1)
```

2.3.2 拟合多因素模型

In [ ]:

```
lmres2 = ols('index1 ~ C(s0) + C(time) + C(s0)*C(time) + s3', ccss).fit()
type(lmres2)
```

In [ ]:

```
lmres2.summary()
```

In [ ]:

```
anova_lm(lmres2)
```

In [ ]:

```
ols('index1 ~ C(s0) + C(time) + s3', ccss).fit().summary()
```

In [ ]:

```
anova_lm(ols('index1 ~ C(s0) + C(time) + s3', ccss).fit())
```

2.4 模型框架下的自定义检验

按照指定的各种系数的数值组合方式进行联合检验。
可使用矩阵方式，也可直接使用变量表达式方式进行检验内容的设定。

GenericLikelihoodModelResults.f_test(

r_matrix : (array-like, str, or tuple), 希望进行的自定义检验
array : r * k array, r是限定条件数，k是模型参数数量
str : 以字符串方式表达的无效假设组合
tuple : A tuple of arrays in the form (R, q)

基本无需设定的参数
cov_p - alter estimate for the parameter covariance matrix.
scale - Default is 1.0 for no scaling.
invcov - q x q array to specify an inverse covariance matrix.

)

2.4.1 和普通检验方式的对照

只含有主效应的简单模型

In [ ]:

```
lmres = ols('index1 ~ C(s0)', ccss).fit()
lmres.summary()
```

In [ ]:

```
# 对常数项进行检验
lmres.f_test([1,0,0])
```

In [ ]:

```
np.sqrt(lmres.f_test([1,0,0]).fvalue)
```

In [ ]:

```
# 将结果存为对象使用
res = lmres.f_test([1,0,0])
res.summary
```

In [ ]:

```
# 对北京的系数进行检验
lmres.f_test([0,1,0])
```

In [ ]:

```
# 对s0进行整体检验（模型中所有alpha均为0）
lmres.f_test([[0,1,0],
              [0,0,1]])
```

In [ ]:

```
lmres.f_test([0,1,-1])
```

In [ ]:

```
# 使用字符串方式设定无效假设
lmres.f_test('C(s0)[T.北京] = 0')
```

In [ ]:

```
lmres.f_test('C(s0)[T.北京] = C(s0)[T.广州] = 0')
```

In [ ]:

```
lmres.f_test("(C(s0)[T.北京] = 0), (C(s0)[T.广州] = 0)")
```

含交互项和协变量的复杂模型

In [ ]:

```
lmres2 = ols('index1 ~ C(s0) + C(time) + C(s0)*C(time) + s3',
            ccss).fit()
lmres2.summary()
```

In [ ]:

```
lmres2.f_test("s3 = 0")
```

In [ ]:

```
lmres2.f_test("C(s0)[T.广州]:C(time)[T.200912] = 0")
```

In [ ]:

```
lmres2.f_test([0,0,0,0,0,0,0,0,0,0,0,1,0])
```

2.4.2 各种组合检验需求的实现

系数具体值的检验

In [ ]:

```
lmres.f_test("Intercept = -1")
```

In [ ]:

```
lmres.f_test("C(s0)[T.北京] = 3")
```

多个系数的联合检验

In [ ]:

```
lmres2.f_test("(C(s0)[T.北京] = C(s0)[T.广州]), (Intercept = -0.7)")
```

控制交互项之后系数的检验

In [ ]:

```
lmres2.summary()
```

In [ ]:

```
# # 检验time = 200704时，北京和上海的系数是否相同
lmres2.f_test("(C(s0)[T.北京] = 0)")
```

In [ ]:

```
# 检验time= 200712时，北京和广州的系数是否相同
lmres2.f_test([0,
               1, -1,
               0, 0, 0,
               1, -1,
               0, 0, 0, 0,
               0])
```

2.5 实战练习

请尝试使用标准数据矩阵方式而不是模型表达式方式拟合2.3.2节中拟合的多因素方差分析模型。

提示：对于分类自变量需要先手工转换为哑变量。

请尝试使用自定义检验方式实现如下检验需求：

上海在2007年12月和2009年12月的系数是否相同。
上海在2008年12月的系数和广州在2009年12月的系数是否相同。

3 线性回归模型

3.1 基本原理

3.1.1 线性回归模型概述

3.1.2 线性回归模型的适用条件