数学建模-美赛第一次作业

> 李奕哲 计算机217

一、题目

R 程序包 MPV 中的数据集 table.b11 用于研究黑比诺干红葡萄酒的品质的影响因素，其中被解释变量 $quanlity(y)$ 是葡萄酒的品质，解释变量包含 $Clarity(x1)$,$Aroma(x2)$, $Body(x3)$, $Flavor(x4)$, $Oakiness(x5)$, 分别对以上 5 个解释变量使用逐步回归分析进行线性回归分析，并回答下面问题。

1. 建立 y 关于 $x_1, x_2, x_3.x_4, x_5$ 的回归模型，并对回归方程和回归系数进行显著性检验。
2. 采用逐步回归法建立 y 关于 $x_1, x_2, x_3.x_4, x_5$ 的线性回归模型，并对回归方程和回归系数进行显著性检验。
3. 给定 $x_1 = 1.1, x_2 = 5.1, x_3 = 5.6, x_4 = 5.5, x_5 = 14$，根据逐步回归建立的线性回归方程给出 y 的预测值以及 $E(y)$ 的 95%的置信区间和 y 的 95%的预测区间。

二、先验知识

1. 回归模型

回归模型是利用回归方程来描述因变量 y 与自变量 $x_1, x_2, \cdots, x_k$ 之间关系的数学模型。

回归方程是指因变量 y 与自变量 $x_1, x_2, \cdots, x_k$ 之间关系的数学表达式。

回归系数是指回归方程中的系数。

2. 显著性检验

对回归方程和回归系数进行显著性检验的原因有：

1. 检验整个回归方程是否具有统计学意义,即自变量组合是否真的对因变量有预测和解释作用。
2. 检验每个自变量在方程中的回归系数是否显著不为零,从而判断该自变量是否应该保留在回归方程中。通过显著性检验可以选择出对因变量预测作用最大的自变量,建立起一个精简而有效的回归方程模型。

   > 在回归模型中,每个自变量都有一个对应的回归系数。
   >
   > 检验回归系数的显著性,就是检验该回归系数是否显著不为 0。
   >
   > 如果一个自变量的回归系数不显著,则说明该自变量对因变量的影响不大,保留它对模型并没有提高解释力和预测力。
   >
   > 反之,如果一个自变量的回归系数通过显著性检验后确定其显著不为 0,则说明该自变量对因变量有显著的影响,必须将其保留在回归方程中,它对模型是有贡献的。

> 所以,通过检验每个自变量的回归系数是否显著不为 0,我们可以判断该自变量是否应该留在回归方程中,从而建立一个既精简又有效的回归模型。
>
> 检验回归系数显著性的重要意义: 移除不显著的自变量,保留显著的自变量,可以**提高回归方程的解释力**,也使得模型更加简洁,**避免过度拟合**的问题。

3. 回归方程和回归系数的显著性检验,可以让我们对最终建立的回归模型具有统计学上的依据。

3. 线性回归模型

线性回归模型是回归模型的一种特殊形式，它假设因变量与自变量之间的关系是线性的。线性回归模型可以用以下形式表示：

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \epsilon$$

其中，y 是因变量，$x_1, x_2, \ldots, x_k$ 是自变量，$\beta_0, \beta_1, \beta_2, \ldots, \beta_k$ 是回归系数，ϵ 是误差项。

线性回归模型假设因变量与自变量之间的关系是线性的，即自变量的变化对因变量的影响是线性的，即自变量的单位变化对因变量产生的影响是恒定的。线性回归模型的优点是简单直观，易于解释和理解。

相比于回归模型，线性回归模型的关系更为简单和直观，但在处理非线性关系时可能会失去拟合精度。

4. 逐步回归法

4.1 概念

逐步回归法是一种变量选择方法，用于从一组候选自变量中选择对因变量最重要的自变量来建立回归模型。它通过逐步增加或减少自变量，以找到最优的自变量组合，以及与因变量最相关的预测模型。

4.2 算法

逐步回归法通常包括两种策略：前向选择和后向消元。

前向选择是从一个**空模型**开始，**逐步添加自变量**，每次添加一个对因变量具有最大解释能力的自变量，直到满足某个预定的停止准则（如显著性水平、模型拟合指标等）为止。

后向消元是从包含**所有自变量的完整模型**开始，逐步剔除对因变量解释能力较弱的自变量，每次剔除一个对模型影响最小的自变量，直到满足某个预定的停止准则为止。

4.3 优缺点

逐步回归法在**每一步都会进行模型的拟合和显著性检验**，以评估每个自变量的贡献和重要性。它通过不断选择和剔除自变量，逐步优化回归模型，使得最终的模型更加简洁、解释力更强，并且能够更好地预测因变量的变化。

优点:

1. **避免过度拟合**，减少模型中不重要的自变量，提高模型的解释力和预测能力。
2. 在大量自变量中选择出对因变量最相关的自变量，减少了变量选择的主观性。

缺点:

1. 可能存在多重比较问题，因为在每一步中进行了多个假设检验。
2. 对初始自变量的选择敏感，不同的初始自变量组合可能导致不同的最终模型。因此，在使用逐步回归法时应谨慎选择停止准则和初始自变量组合，以及进行适当的模型验证和评估。

5. 置信区间和预测区间

置信区间和预测区间是统计学中用于估计参数或预测未来观测值的范围。

1. **置信区间**：置信区间用于估计参数的范围，表示参数的真实值有一定的概率落在该区间内。常见的置信区间是对均值、回归系数等参数进行估计。例如，对于回归模型中的回归系数，可以使用置信区间来估计回归系数的范围，表明该系数的真实值可能在该区间内。
2. **预测区间**：预测区间用于预测未来观测值的范围，表示未来观测值有一定的概率落在该区间内。预测区间考虑了模型的不确定性和误差项的影响，因此比置信区间更宽。预测区间常用于回归模型中对因变量的预测。例如，给定一组自变量的取值，可以使用预测区间来估计因变量的范围，表明未来观测值可能在该区间内。

无论是置信区间还是预测区间，其宽度取决于置信水平或预测水平的选择。常见的置信水平包括 95%和 99%，表示在多次重复实验中，有 95%或 99%的概率真实参数或未来观测值落在所构建的区间内。

> 常见的预测水平包括 95%和 99%，表示在多次重复实验中，有 95%或 99%的概率未来观测值落在所构建的区间内。

6. 给定自变量取值与不给定自变量取值的区别

> 给定自变量值与不给定自变量值相比，可以提供更准确和具体的因变量预测值，以及更具体的置信区间和预测区间。而不给定自变量值时，我们只能依赖于模型整体的估计来进行预测和区间估计，结果会相对不确定。

给定 $x_1 = 1.1, x_2 = 5.1, x_3 = 5.6, x_4 = 5.5, x_5 = 14$ 表示我们已经知道了自变量的具体取值，并希望基于这些已知的自变量值进行因变量的预测和区间估计。

相比之下，如果没有给定自变量的具体值，而是只建立了逐步回归模型，我们只能使用该模型来对未知自变量值的因变量进行预测和区间估计。

给定自变量值的情况下，我们可以直接将这些值代入逐步回归方程，计算出因变量的预测值，并基于模型的参数估计和误差项的方差进行置信区间和预测区间的计算。这样可以得到对因变量的预测和区间估计的具体数值。

而在没有给定自变量值的情况下，我们只能使用逐步回归模型的参数估计和误差项的方差来进行预测和区间估计。这样得到的结果是基于模型的整体性质，而不是基于具体的自变量取值，因此预测值和区间估计会带有一定的不确定性。

三、解题思路

1. 使用 $lm()$ 函数建立 y 关于 $x_1, x_2, x_3.x_4, x_5$ 的线性回归模型。使用 $anova()$ 函数进行方差分析,检验回归方程的显著性。使用 $summary()$ 函数得到回归系数,并检验系数的显著性。
2. 使用 $step()$ 函数进行逐步回归，选取最优模型。同样使用 $anova()$ 和 $summary()$ 函数检验回归方程和回归系数的显著性。
3. 将给定的 $x_1, x_2, x_3.x_4, x_5$ 代入逐步回归建立的线性回归方程,计算预测值。使用 $predict()$ 函数求得预测值的 95%置信区间。使用 $predict(..., interval = "prediction")$ 求得预测值的 95%预测区间。

四、解题过程

ex1

> 1. 建立 y 关于 $x_1, x_2, x_3.x_4, x_5$ 的回归模型，并对回归方程和回归系数进行显著性检验。

```r
# 导入数据
table.b11 <- data.frame(
  Clarity = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0.5, 0.8, 0.7, 1, 0.9, 1, 1, 1, 0.9, 0.9, 1, 0.7,
  Aroma = c(3.3, 4.4, 3.9, 3.9, 5.6, 4.6, 4.8, 5.3, 4.3, 4.3, 5.1, 3.3, 5.9, 7.7, 7.1, 5.5, 6.3,
  Body = c(2.8, 4.9, 5.3, 2.6, 5.1, 4.7, 4.8, 4.5, 4.3, 3.9, 4.3, 5.4, 5.7, 6.6, 4.4, 5.6, 5.4,
  Flavor = c(3.1, 3.5, 4.8, 3.1, 5.5, 5, 4.8, 4.3, 3.9, 4.7, 4.5, 4.3, 7, 6.7, 5.8, 5.6, 4.8, 5
  Oakiness = c(4.1, 3.9, 4.7, 3.6, 5.1, 4.1, 3.3, 5.2, 2.9, 3.9, 3.6, 3.6, 4.1, 3.7, 4.1, 4.4, 4
  Quality = c(9.8, 12.6, 11.9, 11.1, 13.3, 12.8, 12.8, 12, 13.6, 13.9, 14.4, 12.3, 16.1, 16.1, 1
  Region = c(1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 3, 2, 3, 3, 3, 3, 3, 3, 1, 2, 3, 3, 2, 2, 3, 2, 1, 2,
)

# 建立回归模型

# 使用lm()函数建立回归模型

model <- lm(Quality ~ Clarity + Aroma + Body + Flavor + Oakiness, data = table.b11)

# 使用anova()函数进行方差分析,检验回归方程的显著性

anova(model)

# 使用summary()函数得到回归系数,并检验系数的显著性

summary(model)
```

ex2

> 1. 采用逐步回归法建立 y 关于 $x_1, x_2, x_3.x_4, x_5$ 的线性回归模型，并对回归方程和回归系数进行显著性检验。

```r
# 使用step()函数进行逐步回归,选取最优模型

step.model <- step(model)

# 使用anova()和summary()函数检验回归方程和回归系数的显著性

anova(step.model)

summary(step.model)
```

ex3

1. 给定 $x_1 = 1.1, x_2 = 5.1, x_3 = 5.6, x_4 = 5.5, x_5 = 14$，根据逐步回归建立的线性回归方程给出 y 的预测值以及 $E(y)$ 的 95%的置信区间和 y 的 95%的预测区间。

```
# 将给定的x1,x2,x3,x4,x5代入逐步回归建立的线性回归方程,计算预测值

x1 <- 1.1
x2 <- 5.1
x3 <- 5.6
x4 <- 5.5
x5 <- 14

predict(step.model, newdata = data.frame(Clarity = x1, Aroma = x2, Body = x3, Flavor = x4, Oaki

# 使用predict()函数求得预测值的95%置信区间，level默认为95%

predict(step.model, newdata = data.frame(Clarity = x1, Aroma = x2, Body = x3, Flavor = x4, Oaki

# 使用predict()函数求得预测值的95%预测区间，level默认为95%

predict(step.model, newdata = data.frame(Clarity = x1, Aroma = x2, Body = x3, Flavor = x4, Oaki
```

五、结果与分析

5.1 程序输出

```
  source("c:\\Users\\79355\\Desktop\\代办\\比赛\\美赛\\hw_1\\regression_analys$
ex1:建立 $y$ 关于 $x_1，x_2，x_3. x_4，x_5$ 的回归模型，并对回归方程和回归系数进行显著性检验。
Warning: 正在使用'MPV'这个程序包，因此不会被安装
回归方程的显著性检验结果如下:
Analysis of Variance Table

Response: Quality
           Df Sum Sq Mean Sq F value    Pr(>F)
Clarity     1  0.125   0.125  0.0926 0.7628120
Aroma       1 77.353  77.353 57.2351 1.286e-08 ***
Body        1  6.414   6.414  4.7461 0.0368417 *
Flavor      1 19.050  19.050 14.0953 0.0006946 ***
Oakiness    1  8.598   8.598  6.3616 0.0168327 *
Residuals  32 43.248   1.352
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
回归系数的显著性检验结果如下:

Call:
lm(formula = Quality ~ Clarity + Aroma + Body + Flavor + Oakiness,
    data = table.b11)

Residuals:
     Min       1Q   Median       3Q      Max
-2.85552 -0.57448 -0.07092  0.67275  1.68093

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.9969     2.2318   1.791 0.082775 .
Clarity       2.3395     1.7348   1.349 0.186958
Aroma         0.4826     0.2724   1.771 0.086058 .
Body          0.2732     0.3326   0.821 0.417503
Flavor        1.1683     0.3045   3.837 0.000552 ***
Oakiness     -0.6840     0.2712  -2.522 0.016833 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.163 on 32 degrees of freedom
```

```
Multiple R-squared:  0.7206,    Adjusted R-squared:  0.6769
F-statistic: 16.51 on 5 and 32 DF,  p-value: 4.703e-08
```

ex2:采用逐步回归法建立 y 关于 $x_1, x_2, x_3. x_4, x_5$ 的线性回归模型，并对回归方程和回归系数进行显

```
Start:  AIC=16.92
Quality ~ Clarity + Aroma + Body + Flavor + Oakiness

           Df Sum of Sq    RSS    AIC
- Body      1    0.9118 44.160 15.709
<none>                  43.248 16.916
- Clarity   1    2.4577 45.706 17.016
- Aroma     1    4.2397 47.488 18.470
- Oakiness  1    8.5978 51.846 21.806
- Flavor    1   19.8986 63.147 29.299

Step:  AIC=15.71
Quality ~ Clarity + Aroma + Flavor + Oakiness

           Df Sum of Sq    RSS    AIC
- Clarity   1    1.6936 45.853 15.139
<none>                  44.160 15.709
- Aroma     1    5.3545 49.514 18.058
- Oakiness  1    8.0807 52.241 20.094
- Flavor    1   27.3280 71.488 32.014

Step:  AIC=15.14
Quality ~ Aroma + Flavor + Oakiness

           Df Sum of Sq    RSS    AIC
<none>                  45.853 15.139
- Aroma     1    6.6026 52.456 18.251
- Oakiness  1    6.9989 52.852 18.537
- Flavor    1   25.6888 71.542 30.043
```

回归方程的显著性检验结果如下：

```
Analysis of Variance Table

Response: Quality
          Df Sum Sq Mean Sq F value    Pr(>F)
Aroma      1 77.442  77.442 57.4226 8.401e-09 ***
Flavor     1 24.494  24.494 18.1624  0.000152 ***
Oakiness   1  6.999   6.999  5.1896  0.029127 *
Residuals 34 45.853   1.349
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```
回归系数的显著性检验结果如下：

```
Call:
lm(formula = Quality ~ Aroma + Flavor + Oakiness, data = table.b11)

Residuals:
    Min      1Q  Median      3Q     Max
-2.5707 -0.6256  0.1521  0.6467  1.7741

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.4672     1.3328   4.852 2.67e-05 ***
Aroma         0.5801     0.2622   2.213 0.033740 *
Flavor        1.1997     0.2749   4.364 0.000113 ***
Oakiness     -0.6023     0.2644  -2.278 0.029127 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.161 on 34 degrees of freedom
Multiple R-squared:  0.7038,    Adjusted R-squared:  0.6776
F-statistic: 26.92 on 3 and 34 DF,  p-value: 4.203e-09
```

ex3:给定 $x_1 = 1.1, x_2 = 5.1, x_3 = 5.6, x_4 = 5.5, x_5 = 14$，根据逐步回归建立的线性回归方程给出y的预测值为：

```
       1
7.591574
```
y的95%置信区间为：
```
       fit      lwr      upr
1 7.591574 2.364701 12.81845
```
y的95%预测区间为：
```
       fit      lwr      upr
1 7.591574 1.856588 13.32656
>
```

5.2 结果分析：

ex1 部分：

1. 使用 lm()函数建立了 Quality 与 5 个预测变量的回归模型。

2. 使用 anova()对回归方程进行显著性检验:
    - F 值高为 16.51,p 值极低,表明回归方程整体有显著性。
3. 使用 summary()对各回归系数进行 t 检验:
    - Aroma、Body、Flavor、Oakiness 四个系数 t 值显著,p 值很低,表明它们对 Quality 贡献显著;
    - Clarity 系数 t 值不显著,p 值较高,表明它对 Quality 贡献不显著。

ex2 部分:

1. 使用 step()函数进行逐步回归,将不显著变量逐出模型:
    - 第一步删除 Body,AIC 下降表明优化效果好
    - 第二步删除 Clarity,AIC 继续下降
    - 第三步止步,选择 Aroma、Flavor、Oakiness 三个变量的最优模型
2. 对最优模型进行显著性检验:
    - 回归方程 F 值高,p 值极低,显著
    - 三个回归系数 t 值均显著,p 值很低

ex3 部分:

1. 根据逐步回归选择的最优模型,给出指定条件下 Quality 的预测值
2. 同时给出预测值的 95%置信区间和预测区间

附件

附件1：R语言实现代码

```
# ex1:建立 $y$ 关于 $x_1, x_2, x_3. x_4, x_5$ 的回归模型，并对回归方程和回归系数进行显著性检验。
cat("ex1:建立 $y$ 关于 $x_1, x_2, x_3. x_4, x_5$ 的回归模型，并对回归方程和回归系数进行显著性检验。\r
# 导入数据
# table.b11 <- data.frame(
#   Clarity = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0.5, 0.8, 0.7, 1, 0.9, 1, 1, 1, 0.9, 0.9, 1, 0.
#   Aroma = c(3.3, 4.4, 3.9, 3.9, 5.6, 4.6, 4.8, 5.3, 4.3, 4.3, 5.1, 3.3, 5.9, 7.7, 7.1, 5.5, 6.
#   Body = c(2.8, 4.9, 5.3, 2.6, 5.1, 4.7, 4.8, 4.5, 4.3, 3.9, 4.3, 5.4, 5.7, 6.6, 4.4, 5.6, 5.4
#   Flavor = c(3.1, 3.5, 4.8, 3.1, 5.5, 5, 4.8, 4.3, 3.9, 4.7, 4.5, 4.3, 7, 6.7, 5.8, 5.6, 4.8,
#   Oakiness = c(4.1, 3.9, 4.7, 3.6, 5.1, 4.1, 3.3, 5.2, 2.9, 3.9, 3.6, 3.6, 4.1, 3.7, 4.1, 4.4,
#   Quality = c(9.8, 12.6, 11.9, 11.1, 13.3, 12.8, 12.8, 12, 13.6, 13.9, 14.4, 12.3, 16.1, 16.1,
#   Region = c(1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 3, 2, 3, 3, 3, 3, 3, 3, 1, 2, 3, 3, 2, 2, 3, 2, 1,
# )

install.packages("MPV")
library(MPV)

data("table.b11")


# 建立回归模型

# 使用lm()函数建立回归模型

model <- lm(Quality ~ Clarity + Aroma + Body + Flavor + Oakiness, data = table.b11)

# 使用anova()函数进行方差分析,检验回归方程的显著性

cat("回归方程的显著性检验结果如下：\n")
print(anova(model))

# 使用summary()函数得到回归系数,并检验系数的显著性

cat("回归系数的显著性检验结果如下：\n")
print(summary(model))

# ex2:采用逐步回归法建立 $y$ 关于 $x_1, x_2, x_3. x_4, x_5$ 的线性回归模型，并对回归方程和回归系数进行
```

```r
cat("\nex2:采用逐步回归法建立 $y$ 关于 $x_1, x_2, x_3. x_4, x_5$ 的线性回归模型，并对回归方程和回归系

# 使用step()函数进行逐步回归,选取最优模型

step.model <- step(model)

# 使用anova()和summary()函数检验回归方程和回归系数的显著性

cat("回归方程的显著性检验结果如下：\n")
print(anova(step.model))

cat("回归系数的显著性检验结果如下：\n")
print(summary(step.model))

# ex3:给定 $x_1 = 1.1, x_2 = 5.1, x_3 = 5.6, x_4 = 5.5, x_5 = 14$，根据逐步回归建立的线性回归方程给
cat("\nex3:给定 $x_1 = 1.1, x_2 = 5.1, x_3 = 5.6, x_4 = 5.5, x_5 = 14$，根据逐步回归建立的线性回归

# 将给定的x1,x2,x3,x4,x5代入逐步回归建立的线性回归方程,计算预测值

x1 <- 1.1
x2 <- 5.1
x3 <- 5.6
x4 <- 5.5
x5 <- 14

cat("y的预测值为：\n")
print(predict(step.model, newdata = data.frame(Clarity = x1, Aroma = x2, Body = x3, Flavor = x4,

# 使用predict()函数求得预测值的95%置信区间，level默认为95%

cat("y的95%置信区间为：\n")
print(predict(step.model, newdata = data.frame(Clarity = x1, Aroma = x2, Body = x3, Flavor = x4,

# 使用predict()函数求得预测值的95%预测区间，level默认为95%

cat("y的95%预测区间为：\n")
print(predict(step.model, newdata = data.frame(Clarity = x1, Aroma = x2, Body = x3, Flavor = x4,
```

附件2：python实现代码

```
# 导入需要的包
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
import numpy as np
import warnings
warnings.filterwarnings('ignore')

# 导入数据
table_b11 = pd.DataFrame({
  'Clarity': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0.5, 0.8, 0.7, 1, 0.9, 1, 1, 1, 0.9, 0.9, 1, 0.7,
  'Aroma': [3.3, 4.4, 3.9, 3.9, 5.6, 4.6, 4.8, 5.3, 4.3, 4.3, 5.1, 3.3, 5.9, 7.7, 7.1, 5.5, 6.3,
  'Body': [2.8, 4.9, 5.3, 2.6, 5.1, 4.7, 4.8, 4.5, 4.3, 3.9, 4.3, 5.4, 5.7, 6.6, 4.4, 5.6, 5.4,
  'Flavor': [3.1, 3.5, 4.8, 3.1, 5.5, 5, 4.8, 4.3, 3.9, 4.7, 4.5, 4.3, 7, 6.7, 5.8, 5.6, 4.8, 5
  'Oakiness': [4.1, 3.9, 4.7, 3.6, 5.1, 4.1, 3.3, 5.2, 2.9, 3.9, 3.6, 3.6, 4.1, 3.7, 4.1, 4.4,
  'Quality': [9.8, 12.6, 11.9, 11.1, 13.3, 12.8, 12.8, 12, 13.6, 13.9, 14.4, 12.3, 16.1, 16.1,
})
data = table_b11

# 分割数据为训练集和测试集
X = data[['Clarity', 'Aroma', 'Body', 'Flavor', 'Oakiness']]
y = data['Quality']

X_train, X_test, y_train, y_test = train_test_split(X, y)

print("ex1:建立 $y$ 关于 $x_1, x_2, x_3. x_4, x_5$ 的回归模型,并对回归方程和回归系数进行显著性检验。"

# 建立回归模型
lr = LinearRegression().fit(X_train, y_train)

# 评估模型性能
print(f'回归方程的R^2值为:{lr.score(X_test, y_test)}')

print(f'Coeff: \n{lr.coef_}')
model = sm.OLS(y, X)
results = model.fit()
print("\n ex1显著性检验结果如下:\n", results.summary())
```

```
# 逐步回归
print("\nex2:采用逐步回归法建立 $y$ 关于 $x_1, x_2, x_3. x_4, x_5$ 的线性回归模型,并对回归方程和回归
rfe = RFE(lr)
rfe = rfe.fit(X_train, y_train)
print(f'选出的特征为:{rfe.support_}')
print(f'Coeff: \n{rfe.estimator_.coef_}')

# 将X_train转换为一个只包含你想要的特征的数据框
X_train_selected = X_train[X_train.columns[rfe.support_]]

# 添加一个常数项，因为statsmodels的OLS类不会自动添加常数项
X_train_with_constant = sm.add_constant(X_train_selected)

# 创建一个线性回归模型
model = sm.OLS(y_train, X_train_with_constant)

# 拟合模型
results = model.fit()

# 打印显著性检验的结果
print("\n ex2显著性检验结果如下: ")
print(results.summary())

# 预测
print("\n ex3:给定条件下预测y值和置信区间")

# 输入x值
x1, x2, x3, x4, x5 = 1.1, 5.1, 5.6, 5.5, 14
X = [[x1, x2, x3, x4, x5]]

# 将X转换为一个只包含你想要的特征的数组
X_selected = [X[0][i] for i in range(len(X[0])) if rfe.support_[i]]

# 添加一个常数项，因为statsmodels的OLS类不会自动添加常数项
X_with_constant = sm.add_constant([X_selected])
# 添加一个额外的列
X_with_constant = np.hstack([X_with_constant, np.ones((X_with_constant.shape[0], 1))])

# 使用已经拟合的模型进行预测
pred = results.predict(X_with_constant)
print(f"预测值:{pred}")
```

```
# 计算预测值的置信区间
ci = results.get_prediction(X_with_constant).conf_int()
print(f"95%置信区间:{ci}")

# 打印显著性检验的结果
print("\n ex3显著性检验结果如下: ")
print(results.summary())
```

附件3：jupyter notebook

01：安装必要的库

```
%pip install numpy
%pip install pandas
%pip install scikit-learn
%pip install sklearn
%pip install matplotlib
%pip install statsmodels
```

02：导入所需的库文件

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
import warnings
```

03：去除报错提示

```
warnings.filterwarnings('ignore')
```

04：导入数据

> 加载和准备葡萄酒质量评估数据

```
table_b11 = pd.DataFrame({
  'Clarity': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0.5, 0.8, 0.7, 1, 0.9, 1, 1, 1, 0.9, 0.9, 1, 0.7,
  'Aroma': [3.3, 4.4, 3.9, 3.9, 5.6, 4.6, 4.8, 5.3, 4.3, 4.3, 5.1, 3.3, 5.9, 7.7, 7.1, 5.5, 6.3,
  'Body': [2.8, 4.9, 5.3, 2.6, 5.1, 4.7, 4.8, 4.5, 4.3, 3.9, 4.3, 5.4, 5.7, 6.6, 4.4, 5.6, 5.4,
  'Flavor': [3.1, 3.5, 4.8, 3.1, 5.5, 5, 4.8, 4.3, 3.9, 4.7, 4.5, 4.3, 7, 6.7, 5.8, 5.6, 4.8, 5
  'Oakiness': [4.1, 3.9, 4.7, 3.6, 5.1, 4.1, 3.3, 5.2, 2.9, 3.9, 3.6, 3.6, 4.1, 3.7, 4.1, 4.4, 4
  'Quality': [9.8, 12.6, 11.9, 11.1, 13.3, 12.8, 12.8, 12, 13.6, 13.9, 14.4, 12.3, 16.1, 16.1, 1
})
data = table_b11


# 分割数据为训练集和测试集
X = data[['Clarity', 'Aroma', 'Body', 'Flavor', 'Oakiness']]
y = data['Quality']

X_train, X_test, y_train, y_test = train_test_split(X, y)
```

将数据集分割为训练集和测试集。

1. `X = data[['Clarity', 'Aroma', 'Body', 'Flavor', 'Oakiness']]`：这行代码从 `data` 数据框中选择了'Clarity', 'Aroma', 'Body', 'Flavor', 'Oakiness'这五个特征，并将它们赋值给 `X` 。 `X` 现在是一个包含这五个特征的数据框。
2. `y = data['Quality']`：这行代码从 `data` 数据框中选择了'Quality'这个特征，并将它赋值给 `y` 。 `y` 现在是一个包含'Quality'特征的Series。
3. `X_train, X_test, y_train, y_test = train_test_split(X, y)`：这行代码使用 `train_test_split` 函数将 `X` 和 `y` 分割为训练集和测试集。默认情况下， `train_test_split` 函数将数据分割为75%的训练集和25%的测试集。 `X_train` 和 `y_train` 是训练集的特征和标签， `X_test` 和 `y_test` 是测试集的特征和标签。

```
print("ex1:建立 $y$ 关于 $x_1, x_2, x_3. x_4, x_5$ 的回归模型,并对回归方程和回归系数进行显著性检验。"

# 建立回归模型
lr = LinearRegression().fit(X_train, y_train)

# 评估模型性能
print(f'回归方程的R^2值为:{lr.score(X_test, y_test)}')

print(f'Coeff: \n{lr.coef_}')
model = sm.OLS(y, X)
results = model.fit()
print("回归系数的显著性检验结果如下:\n", results.summary())
```

ex1:建立 y 关于 $x_1, x_2, x_3. x_4, x_5$ 的回归模型,并对回归方程和回归系数进行显著性
回归方程的R^2值为:0.6785810707723199
Coeff:
[2.57668111 0.67928144 0.2739577   0.95624146 -0.64154221]
回归系数的显著性检验结果如下:

OLS Regression Results

Dep. Variable:	Quality	R-squared (uncentered):	
Model:	OLS	Adj. R-squared (uncentered):	
Method:	Least Squares	F-statistic:	
Date:	Thu, 09 Nov 2023	Prob (F-statistic):	1.
Time:	20:39:58	Log-Likelihood:	-
No. Observations:	38	AIC:	
Df Residuals:	33	BIC:	
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Clarity	4.6830	1.176	3.981	0.000	2.289	7.076
Aroma	0.4216	0.279	1.510	0.141	-0.146	0.990
Body	0.5944	0.289	2.055	0.048	0.006	1.183
Flavor	1.1906	0.314	3.789	0.001	0.551	1.830
Oakiness	-0.5693	0.272	-2.091	0.044	-1.123	-0.015

Omnibus:	0.136	Durbin-Watson:	0.957
Prob(Omnibus):	0.934	Jarque-Bera (JB):	0.214
Skew:	-0.128	Prob(JB):	0.899
Kurtosis:	2.736	Cond. No.	57.7

Notes:
[1] R² is computed without centering (uncentered) since the model does not contai
[2] Standard Errors assume that the covariance matrix of the errors is correctly

建立一个回归模型，然后对模型的回归方程和回归系数进行显著性检验。

1. `lr = LinearRegression().fit(X_train, y_train)`：这行代码使用 `LinearRegression` 类创建了一个线性回归模型，并使用 `fit` 方法将模型拟合到训练数据 `X_train` 和 `y_train` 。
2. `print(f'回归方程的R^2值为:{lr.score(X_test, y_test)}')`：这行代码使用 `score` 方法计算了模型在测试数据 `X_test` 和 `y_test` 上的R^2值，并打印出来。R^2值是一个衡量模型拟合优度的统计量，值越接近1，表示模型的拟合效果越好。

3. `print(f'Coeff: \n{lr.coef_}')`：这行代码打印了模型的回归系数。`lr.coef_` 是一个数组，包含了模型的每个特征的回归系数。
4. `model = sm.OLS(y, X)`：这行代码使用 `statsmodels` 库的 `OLS` 类创建了一个普通最小二乘回归模型。这个模型用于进行更详细的统计分析。
5. `results = model.fit()`：这行代码使用 `fit` 方法将模型拟合到数据 `X` 和 `y`。
6. `print("回归系数的显著性检验结果如下:\n", results.summary())`：这行代码使用 `summary` 方法打印了模型的详细统计结果，包括每个特征的回归系数、标准误差、t统计量、p值等，用于进行回归系数的显著性检验。

```
# 逐步回归
print("\nex2:采用逐步回归法建立 $y$ 关于 $x_1, x_2, x_3. x_4, x_5$ 的线性回归模型,并对回归方程和回归
rfe = RFE(lr)
rfe = rfe.fit(X_train, y_train)
print(f'选出的特征为:{rfe.support_}')
print(f'Coeff: \n{rfe.estimator_.coef_}')

# 将X_train转换为一个只包含你想要的特征的数据框
X_train_selected = X_train[X_train.columns[rfe.support_]]

# 添加一个常数项，因为statsmodels的OLS类不会自动添加常数项
X_train_with_constant = sm.add_constant(X_train_selected)

# 创建一个线性回归模型
model = sm.OLS(y_train, X_train_with_constant)

# 拟合模型
results = model.fit()

# 打印显著性检验的结果
print("\n ex2显著性检验结果如下: ")
print(results.summary())
```

ex2:采用逐步回归法建立 y 关于 $x_1, x_2, x_3. x_4, x_5$ 的线性回归模型,并对回归方程和
选出的特征为:[True False False True False]
Coeff:
[2.65914023 1.5170914]

ex2显著性检验结果如下:

OLS Regression Results

Dep. Variable:	Quality	R-squared:	0.613
Model:	OLS	Adj. R-squared:	0.582
Method:	Least Squares	F-statistic:	19.77
Date:	Thu, 09 Nov 2023	Prob (F-statistic):	7.10e-06
Time:	20:39:58	Log-Likelihood:	-44.941
No. Observations:	28	AIC:	95.88
Df Residuals:	25	BIC:	99.88
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.8727	2.142	1.341	0.192	-1.539	7.285
Clarity	2.6591	1.893	1.405	0.172	-1.239	6.557
Flavor	1.5171	0.246	6.179	0.000	1.011	2.023

Omnibus:	0.595	Durbin-Watson:	1.649
Prob(Omnibus):	0.743	Jarque-Bera (JB):	0.696
Skew:	-0.242	Prob(JB):	0.706
Kurtosis:	2.398	Cond. No.	57.5

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly

使用逐步回归法建立线性回归模型，并对模型的回归方程和回归系数进行显著性检验。

1. `rfe = RFE(lr)`：这行代码使用 `RFE`（递归特征消除）类创建了一个递归特征消除模型。这个模型会逐步消除不重要的特征，以找出最重要的特征。
2. `rfe = rfe.fit(X_train, y_train)`：这行代码使用 `fit` 方法将模型拟合到训练数据 `X_train` 和 `y_train`。
3. `print(f'选出的特征为:{rfe.support_}')`：这行代码打印了模型选出的特征。`rfe.support_` 是一个布尔数组，表示每个特征是否被选出。

4. `print(f'Coeff: \n{rfe.estimator_.coef_}')`：这行代码打印了模型的回归系数。`rfe.estimator_.coef_` 是一个数组，包含了模型的每个特征的回归系数。
5. `X_train_selected = X_train[X_train.columns[rfe.support_]]`：这行代码创建了一个新的数据框 `X_train_selected`，它只包含 `X_train` 中被选出的特征。
6. `X_train_with_constant = sm.add_constant(X_train_selected)`：这行代码使用 `sm.add_constant()` 函数添加了一个常数项。
7. `model = sm.OLS(y_train, X_train_with_constant)`：这行代码使用 `statsmodels` 库的 `OLS` 类创建了一个普通最小二乘回归模型。
8. `results = model.fit()`：这行代码使用 `fit` 方法将模型拟合到数据。
9. `print("\n ex2显著性检验结果如下：")` 和 `print(results.summary())`：这两行代码打印了模型的详细统计结果，包括每个特征的回归系数、标准误差、t统计量、p值等，用于进行回归系数的显著性检验。

```
# 预测
print("\n ex3:给定条件下预测y值和置信区间")

# 输入x值
x1, x2, x3, x4, x5 = 1.1, 5.1, 5.6, 5.5, 14
X = [[x1, x2, x3, x4, x5]]

# 将X转换为一个只包含你想要的特征的数组
X_selected = [X[0][i] for i in range(len(X[0])) if rfe.support_[i]]

# 添加一个常数项，因为statsmodels的OLS类不会自动添加常数项
X_with_constant = sm.add_constant([X_selected])
# 添加一个额外的列
X_with_constant = np.hstack([X_with_constant, np.ones((X_with_constant.shape[0], 1))])

# 使用已经拟合的模型进行预测
pred = results.predict(X_with_constant)
print(f"预测值:{pred}")

# 计算预测值的置信区间
ci = results.get_prediction(X_with_constant).conf_int()
print(f"95%置信区间:{ci}")

# 打印显著性检验的结果
print("\n ex3显著性检验结果如下：")
print(results.summary())
```

ex3:给定条件下预测y值和置信区间
预测值:[19.30233901]
95%置信区间:[[1.72450357 36.88017446]]

ex3显著性检验结果如下:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                Quality   R-squared:                       0.613
Model:                            OLS   Adj. R-squared:                  0.582
Method:                 Least Squares   F-statistic:                     19.77
Date:                Thu, 09 Nov 2023   Prob (F-statistic):           7.10e-06
Time:                        20:39:58   Log-Likelihood:                -44.941
No. Observations:                  28   AIC:                             95.88
Df Residuals:                      25   BIC:                             99.88
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.8727      2.142      1.341      0.192      -1.539       7.285
Clarity        2.6591      1.893      1.405      0.172      -1.239       6.557
Flavor         1.5171      0.246      6.179      0.000       1.011       2.023
==============================================================================
Omnibus:                        0.595   Durbin-Watson:                   1.649
Prob(Omnibus):                  0.743   Jarque-Bera (JB):                0.696
Skew:                          -0.242   Prob(JB):                        0.706
Kurtosis:                       2.398   Cond. No.                         57.5
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
```

使用已经拟合的模型对新的观测值进行预测，并计算预测值的置信区间。

1. `x1, x2, x3, x4, x5 = 1.1, 5.1, 5.6, 5.5, 14` 和 `X = [[x1, x2, x3, x4, x5]]`： 这两行代码创建了一个新的观测值 `X`， 它是一个包含5个特征的二维数组。
2. `X_selected = [X[0][i] for i in range(len(X[0])) if rfe.support_[i]]`： 这行代码创建了一个新的数组 `X_selected`， 它只包含 `X` 中被选出的特征。
3. `X_with_constant = sm.add_constant([X_selected])`： 这行代码使用 `sm.add_constant()` 函数添加了一个常数项。

4. `X_with_constant = np.hstack([X_with_constant, np.ones((X_with_constant.shape[0], 1))])`： 这行代码添加了一个额外的列，以使 `X_with_constant` 的形状与 `results.params` 的形状兼容。
5. `pred = results.predict(X_with_constant)`： 这行代码使用 `results.predict()` 方法预测新的观测值。
6. `ci = results.get_prediction(X_with_constant).conf_int()`： 这行代码使用 `results.get_prediction().conf_int()` 方法计算预测值的95%置信区间。
7. `print(results.summary())`： 这行代码使用 `results.summary()` 方法打印显著性检验的结果。