بخش ۲:

۱.

رگرسیون Ridge:

در رگرسیون خطی، از روش کمترین مربعات خطا برای تخمین ضرایب استفاده می‌کردیم و به دنبال یافتن ضرایبی بودیم که خطای زیر را کمینه کند:

$$\mathrm{RSS} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j X_{ij}\right)^2$$

روش رگرسیون ridge بسیار مشابه روند بالاست اما یک تفاوتی دارد. در این روش به دنبال یافتن ضرایبی هستیم که مقدار خطای زیر را کمینه کند:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j X_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = \mathrm{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2$$

که $\lambda \geq 0$ یک پارامتر تنظیم‌کننده است. در واقع با این تابع خطا هم به دنبال کمینه کردن خطای آموزشی و هم به دنبال کم کردن واریانس ضرایب و فشرده کردن آن‌ها به سمت صفر هستیم. هر چه ضریب λ بزرگتر باشد، فشرده‌سازی بیشتر صورت می‌گیرد و ضرایب بسیار به صفر نزدیک می‌شوند. به ازای هر مقدار λ دسته ضرایب $\widehat{\beta}_R^{\lambda}$ متفاوتی خواهیم داشت و باید مقدار λ با CV به درستی انتخاب شود.

کاربرد:

وقتی $p > n$ باشد، روش کمترین مربعات خطا جواب یکتا ندارد، اما روش ridge می‌تواند با تحمل مقدار کمی بایاس،واریانس را به شدت کاهش دهد و جواب معقولی پیدا کند. علاوه بر این، روش ridge به لحاظ محاسباتی بسیار کارا است زیرا برخلاف روش انتخاب بهترین زیرمجموعه که می‌بایست 2^p مدل را بررسی کنیم در روش ridge به طور موازی می‌توان مدل را به ازای هر مقدار λ با محاسباتی به سرعت تخمین یک مدل روش کمترین مربعات خطا پیاده‌سازی کرد.

رگرسیون Lasso:

در روش Lasso هدف تخمین ضرایب به نحوی است که تابع خطای زیر کمینه شود:

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j X_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|$$

در هر دو روش محدودیت فشرده‌سازی به عرض از مبدا اعمال نمی‌شود. مقدار عرض از مبدا به نوعی نماینده میانگین خروجی‌ها در زمانی‌که ورودی‌ها صفر هستند، می‌باشد.

مقایسه:

تابع خطای Ridge:

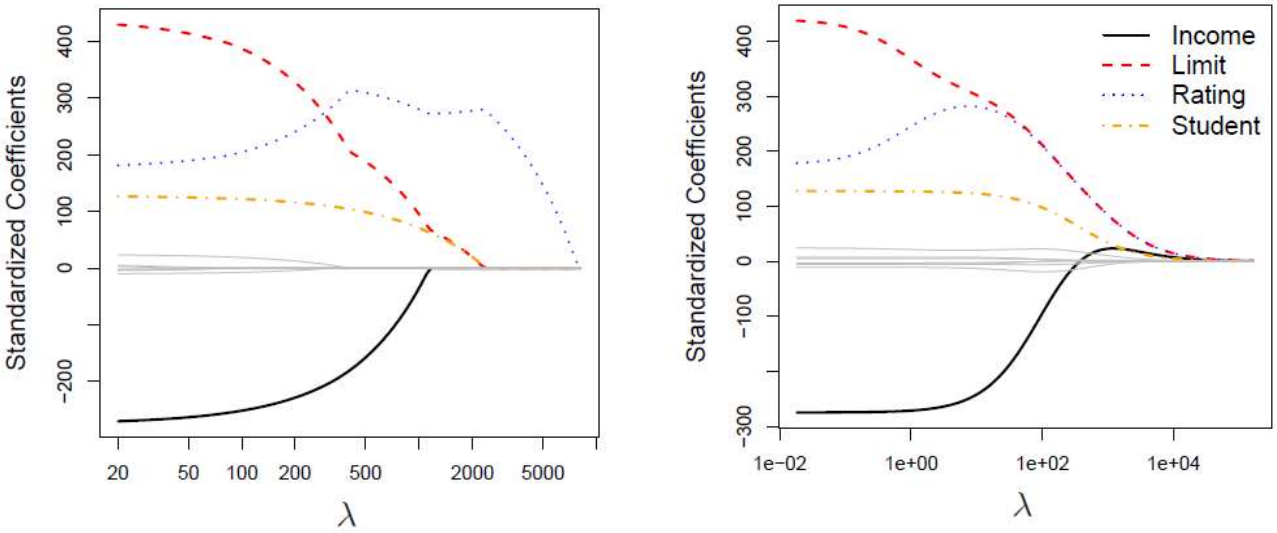
$$\boldsymbol{RSS+\ \lambda\|\beta\|_2^2}$$

تابع خطای Lasso:

$$\boldsymbol{RSS+\ \lambda\|\beta\|_1}$$

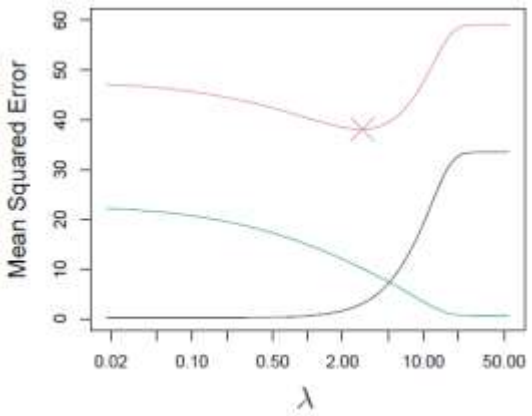
تابع خطای Lasso بسیار شبیه به تابع خطای ridge است اما با این تفاوت که در بخش فشرده‌سازی به جای نرم $\boldsymbol{\ell_2}$ از نرم $\boldsymbol{\ell_1}$ استفاده شده است و همین تفاوت منجر به صفر شدن بسیاری از ضرایب و تولید مدل‌های اسپارس می‌شود. بنابراین با روش Lasso به نوعی انتخاب زیرمجموعه هم انجام می‌شود و این باعث می‌شود تفسیرپذیری مدل آسان‌تر شود. اما در روش ridge، تمام متغیرها در مدل باقی می‌مانند و این باعث می‌شود تفسیرپذیری مدل دشوار باشد.

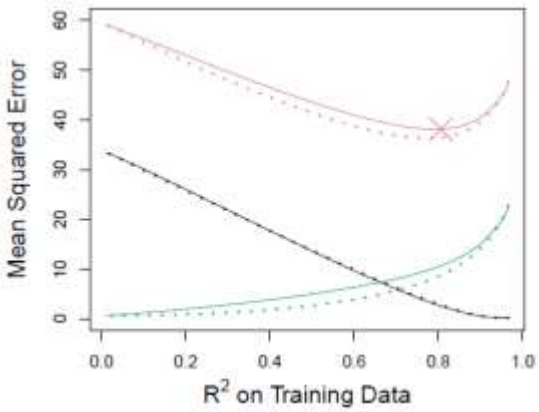
در دو شکل زیر مثالی از اعمال روش Ridge (شکل سمت راست) و اعمال روش Lasso (شکل سمت چپ) بر روی مسئله‌ی اعتبار کارت بانکی نشان داده شده‌است.

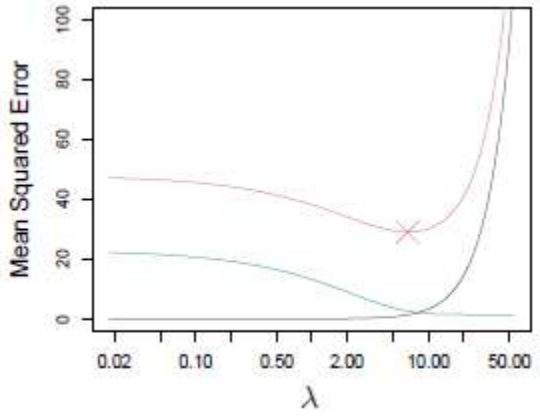


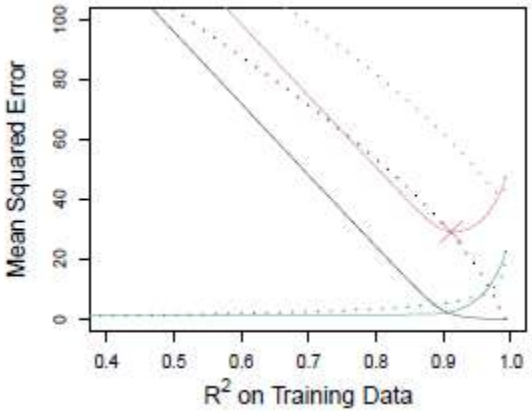
وقتی λ صفر است، ضرایب مانند روش کمترین مربعات خطا و وقتی λ بزرگ است، مدل پوچ است. اما رفتار lasso و ridge در این بین متفاوت است. تصویر سمت چپ(Lasso) را درنظر بگیرید: اگر از سمت راست به چپ برویم، ابتدا مدل تنها شامل پارامتر rating است، سپس پارامترهای student و limit وارد مدل می‌شوند و کمی بعد income به مدل اضافه می‌شود و نهایتا سایر پارامترها وارد مدل می‌شوند. بنابراین با تغییر λ، تعداد پارامترهای مدل تغییر می‌کند؛ درحالی‌که در روش ridge همواره تمام پارامترها در مدل هستند و با تغییر λ تنها میزان حضورشان تغییر می‌کند.

در دو شکل زیر دو مثال از اعمال ridge و lasso بر روی داده‌های شبیه‌سازی شده را مشاهده می‌کنید. در تصاویر سمت چپ تاثیر اعمال lasso بر بایاس، واریانس و MSE نشان داده شده است. همانند Ridge، lasso نیز با افزایش اندکی بایاس، باعث کاهش شدید واریانس می‌شود. نقاط بهینه در نمودارها با ضربدر نمایان شده است. در تصاویر سمت راست، مقایسه ridge (با خط‌چین) و lasso (با خط ممتد) نشان داده شده است. در مثال اول هر دو روش دارای بایاس برابری هستند اما ridge واریانس کمتر و در نتیجه MSE کمتری دارد. درحالی‌که در مثال دوم برعکس این ماجرا اتفاق افتاده و lasso عملکرد بهتری دارد. یک نکته که در این دو مثال بسیار مهم است این است که در مثال اول، داده‌ها به نحوی تولید شده‌اند که تمام ۴۵ متغیر به خروجی مرتبط هستند (یعنی مقادیر واقعی هیچ کدام از ضرایب متغیرها صفر نیست) و چون lasso بعضی از متغیرها را حذف می‌کند، نتوانسته عملکرد خوبی در مثال اول داشته باشد. اما در مثال دوم، خروجی تنها به دو متغیر از بین ۴۵ متغیر وابسته است و برعکس lasso توانسته بهتر عمل کند. این دو مثال نشان می‌دهد که هیچ کدام از این دو روش به طور عام بر دیگری ارجحیت ندارد. به طور کلی در شرایطی که تعداد کمی از متغیرها به خروجی مرتبط هستند، lasso و زمانی‌که خروجی به پارامترهای متعددی وابسته است، ridge بهتر عمل می‌کند. البته این اطلاعات معمولا مجهول است و باید از cross-validation برای کشف آن‌ها کمک گرفت.



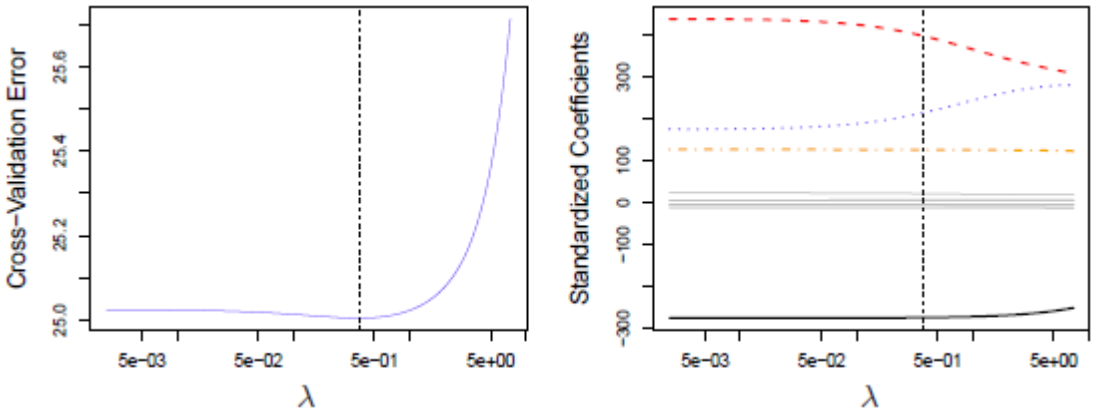






.۲

برای تعیین مقدارλ از روش cross-validation استفاده می‌کنیم. به ازای هر مقدار مختلف λ یک بار cross-validation می‌زنیم و نهایتا مقداری را برای λ انتخاب می‌کنیم که مدل آن دارای خطای کمتری باشد.



شکل بالا اعمال LOOCV بر روی روش ridge برای مثال داده‌های اعتبار کارت بانکی را نشان می‌دهد. خط‌چین عمودی نشان‌گر مقدار بهینه برای انتخاب λ است. در تصویر سمت راست نیز ضرایبی که در به ازای مقادیر مختلف λ تخمین زده شده را نشان می‌دهد. خط‌چین عمودی در این تصویر نمایان‌گر ضرایب بهینه در روش ridge هستند. همین تکنیک را می‌توان برای lasso نیز به کار برد.

۳.

با افزایش فولد بایاس کاهش می یابد زیرا فولدها به مجموعه کل داده نزدیکتر خواهند شد اما هر چه به LOOCV نزدیک تر می شویم واریانس و مدت زمان اجرا افزایش می یابد.

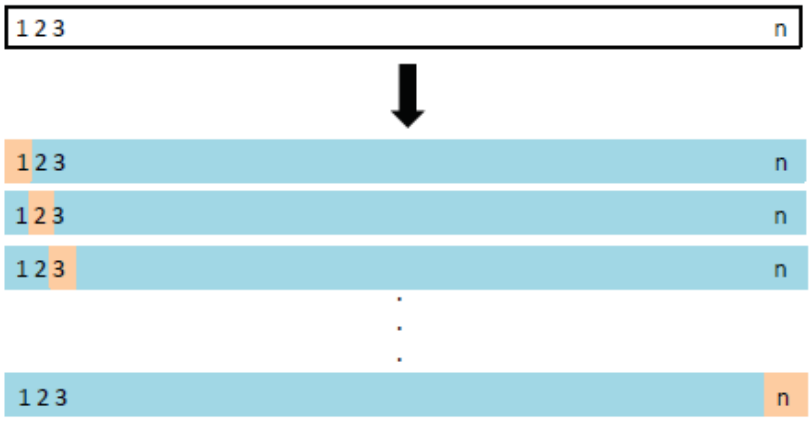
۴.

روش Leave-One-Out-Cross-Validation (LOOCV)

نرخ خطای تست، یک معیار مناسب برای ارزیابی عملکرد مدل ست. اما معمولا مجموعه تست در دسترس نیست و باید به روش‌های دیگری مدل را ارزیابی کنیم. در این روش ما نرخ خطای تست را با استفاده از نرخ خطای آموزشی تخمین می‌زنیم. هر بار یک زیرمجموعه از داده‌های آموزشی را کنار می‌گذاریم، مدل را با استفاده از بقیه داده‌ها آموزش می‌دهیم و از داده‌های کنار گذاشته شده به عنوان داده‌های تست استفاده می‌کنیم.

در این روش ابتدا تنها داده آموزشی $(x_1.y_1)$ به عنوان داده ارزیابی کنار گذاشته می‌شود و مدل بر اساس $n-1$ داده آموزشی دیگر یعنی $\{(x_2.y_2).\ldots.(x_n.y_n)\}$ بدست می‌آید. سپس خروجی برای تنها داده ارزیابی یعنی x_1 پیش‌بینی می‌شود و $\hat{y}_1$ بدست می‌آید. حال مقدار میزان خطا بر روی تنها داده ارزیابی محاسبه

می‌شود که مقدار این خطا برابر است با $MSE_1 = (y_1 - \hat{y}_1)^2$. این خطا، تخمینی از نرخ خطای تست است. اما از آن‌جایی که این خطا بسیار وابسته به تک داده ارزیابی است، این مراحل را به ازای تک تک داده‌های آموزش انجام می‌دهیم. هر بار یکی از داده‌های آموزشی را به عنوان داده ارزیابی کنار گذاشته، مدل را بر روی $n-1$ داده دیگر بدست آورده و خطای مدل را بر روی تک داده ارزیابی حساب می‌کنیم و به این ترتیب مقادیر $MSE_1 . MSE_2 . \ldots . MSE_n$ محاسبه می‌شوند. در شکل زیر نمایی کلی از این روش را می‌بینید. (هربار داده نارنجی رنگ داده ارزیابی است.)



نهایتا روش LOOCV خطای زیر را به عنوان تخمینی از نرخ خطای تست گزارش می‌کند:

$$CV_{(n)} = \frac{1}{n}\Sigma_{i=1}^{n} MSE_i$$

۵.

روش bootstrapping

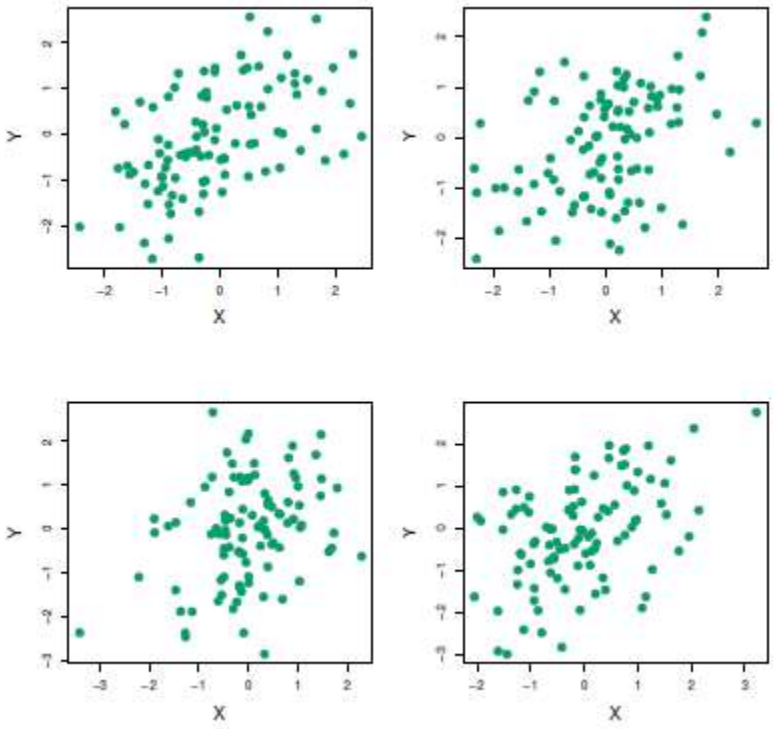
روش bootstrapping از روش‌های نمونه‌گیری مجدد است و روش کارآمدی برای محاسبه میزان دقت و خطای استاندارد (standard error) متغیر تخمین زده شده است. در روش bootstrapping به صورت تصادفی n عضو با جایگذاری انتخاب می‌کنیم.

برای مثال فرض کنید مقدار معینی پول داریم که می‌خواهیم بخشی α درصد آن را در یک پروژه و $1-\alpha$ درصد آن را در یک پروژه دیگر سرمایه‌گذاری کنیم. اگر میزان سود پروژه اول X و میزان سود پروژه دوم Y باشد، به دنبال یافتن αای هستیم که به ازای آن، میزان ریسکمان یعنی $Var(\alpha X + (1-\alpha)Y)$ کمینه شود. می‌توان نشان داد که مقدار ریسک به ازای مقدار زیر کمینه می‌شود:

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$$

که $\sigma_X^2 = Var(X)$ و $\sigma_Y^2 = Var(Y)$ و $\sigma_{XY} = Cov(X.Y)$. در موارد طبیعی، مقادیر σ_X^2، σ_Y^2 و σ_{XY} نامعلوم هستند و با استفاده از داده‌ها تخمین زده می‌شوند، سپس مقدار بهینه α نیز توسط رابطه زیر تخمین زده می‌شود:

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$$



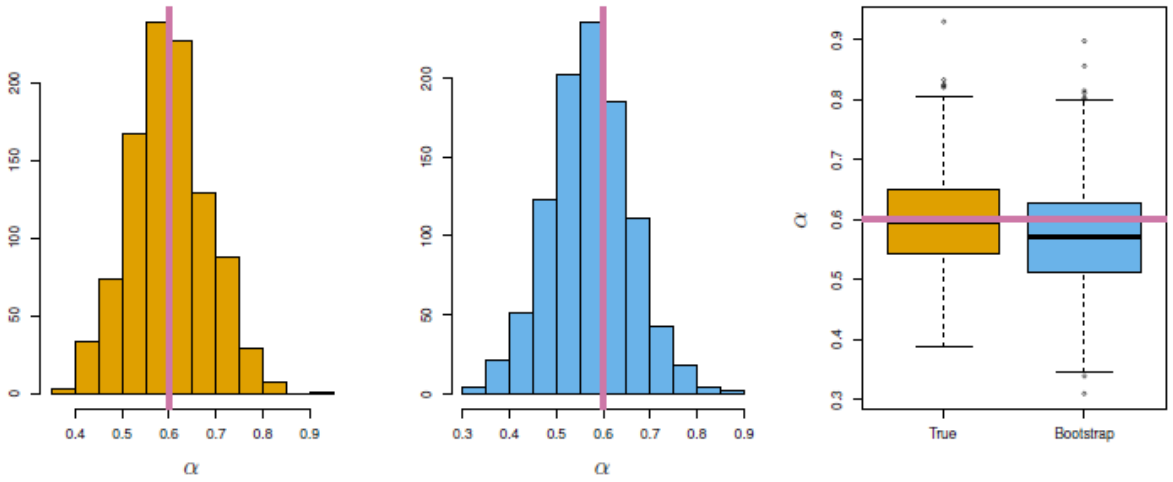
در شکل بالا، در هر قسمت ۱۰۰ جفت سرمایه‌گذاری (X,Y) نشان داده شده است. سپس مقادیر واریانس بر اساس داده‌ها تخمین زده شده و مقادیر بهینه $\hat{\alpha}$ محاسبه شده‌اند. مقدار $\hat{\alpha}$ برای داده‌های بالا سمت چپ ۰.۵۷۶، بالا سمت راست ۰.۵۳۲، پایین سمت چپ ۰.۶۵۷ و پایین سمت راست ۰.۶۵۱ است. برای محاسبه خطای استاندارد $\hat{\alpha}$، روند تولید ۱۰۰ جفت داده (X,Y) و محاسبه $\hat{\alpha}$ را ۱۰۰۰ بار تکرار می‌کنیم. در انتها، ۱۰۰۰ مقدار تخمینی $\hat{\alpha}_1. \hat{\alpha}_2 \ldots . \hat{\alpha}_{1000}$ خواهیم داشت. در شکل زیر قسمت سمت راست ۱۰۰۰ بار مجموعه داده‌های ۱۰۰ جفتی از جامعه‌ای با $\sigma_X^2 = ۱$، $\sigma_Y^2 = ۱.۲۵$ و $\sigma_{XY} = ۰.۵$ تولید کرده‌ایم. می‌دانیم که مقدار بهینه α برای چنین جامعه‌ای، ۰.۶ است که با خط عمودی نشان داده شده است. مقادیر تخمین زده شده $\hat{\alpha}$ برای این ۱۰۰۰ نمونه،محاسبه شده و در هیستوگرام نارنجی رنگ نشان داده شده است. میانگین مقادیر تخمینی برابر است با:

$$\bar{\alpha} = \frac{1}{1000} \sum_{r=1}^{1000} \hat{\alpha}_r = 0.5996$$

که بسیار نزدیک به ۰.۶ است. خطای استاندارد آن برابر است با:

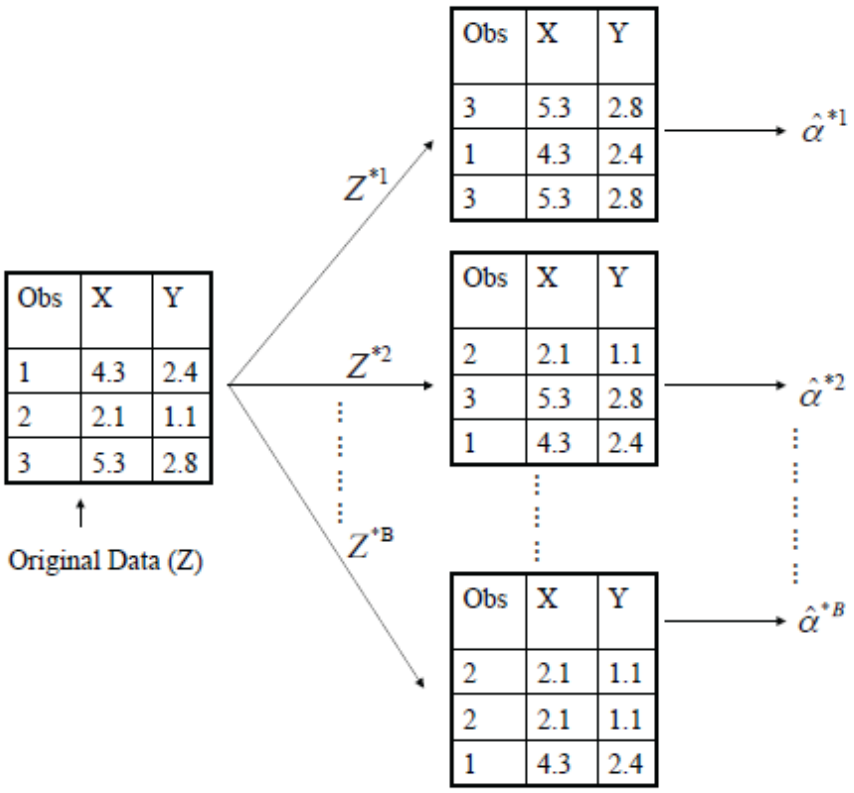
$$\sqrt{\frac{1}{1000-1}\sum_{r=1}^{1000}(\hat{\alpha}_r - \bar{\alpha})^2} = 0.083$$

پس $SE(\hat{\alpha}) = 0.083$. یعنی هر نمونه تصادفی از جامعه که بگیریم، $\hat{\alpha}$ تقریبا ۰.۰۸ از α فاصله دارد. در عمل، انجام روند گفته شده برای محاسبه $SE(\hat{\alpha})$ تقریبا ناممکن است زیرا معمولا نمی‌توان از جامعه، نمونه‌های جدید تولید کرد. در عوض، می‌توان با نمونه‌گیری مجدد از همان نمونه اولیه روند بالا را تقلید کرد. در واقع، هربار به طور تصادفی و با جایگذاری از نمونه اولیه، مجددا نمونه‌گیری می‌کنیم. در قسمت وسط شکل زیر ۱۰۰۰ نمونه با استفاده از روش نمونه‌گیری مجدد تولید شده و مقادیر $\hat{\alpha}$ برای این ۱۰۰۰ نمونه تخمین زده شده و هیستوگرام آبی رنگ رسم شده است. شکل سمت راست نیز بیانگر آن است که توزیع $\hat{\alpha}$ در دو حالتی که نمونه‌ها به صورت جدید از جامعه تولید شوند یا به صورت تکراری و با نمونه‌گیری مجدد به دست آیند، یکسان است.



روش bootstrap در شکل زیر نمایش داده شده است که در آن مجموعه داده‌ای به نام Z با $n = 3$ عضو داریم. به صورت تصادفی n عضو با جایگذاری انتخاب می‌کنیم و مجموعه داده جدید Z_1^* را می‌سازیم. منظور از انتخاب با جایگذاری آن است که یک داده می‌تواند چندین بار انتخاب شود. به عنوان مثال در Z_1^*، داده سوم دوبار انتخاب شده است، داده اول یک بار انتخاب شده است و داده دوم اصلا انتخاب نشده است. لازم به ذکر است که وقتی داده‌ای انتخاب می‌شود، هم X و هم Y انتخاب می‌شود. با این روش مجموعه داده Z_1^* را ساخته و تخمین جدیدی از α به نام $\hat{\alpha}_1^*$ بدست می‌آوریم. این روند را B بار تکرار می‌کنیم و B مجموعه داده $Z_1^*. Z_2^*. \ldots . Z_B^*$ ساخته و تخمین‌های $\hat{\alpha}_1^*. \hat{\alpha}_2^*. \ldots . \hat{\alpha}_B^*$ را محاسبه می‌کنیم. سپس خطای استاندارد $\hat{\alpha}$ از طریق فرمول زیر حساب می‌شود:

$$SE_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\alpha}_r^* - \frac{1}{\mathrm{B}}\sum_{r'=1}^{B}\hat{\alpha}_{r'}^*\right)^2}$$



در روش cross validation هر بار یک زیرمجموعه از داده‌های آموزشی را کنار می‌گذاریم، مدل را با استفاده از بقیه داده‌ها آموزش می‌دهیم و از داده‌های کنار گذاشته شده به عنوان داده‌های تست استفاده می‌کنیم.

۶.

در 2-fold cross-validation داده ها را بصورت رندوم به ۲ بخش هم اندازه تقسیم می کنیم.یک بخش به عنوان داده‌ی ارزیابی و دیگری بعنوان داده‌ی آموزشی استفاده می شود و بار دیگر همین کار را با بخش دیگر انجام میدهیم در نتیجه دو تا معیار خطای تست خواهیم داشت که می توانیم از آن ها میانگین بگیریم و بعنوان خطای تست مدل گزارش دهیم.مزیت: هر بخش یکبار بعنوان داده‌ی ارزیابی استفاده شده است واز همه‌ی مشاهدات استفاده کرده‌ایم.

در 2 fold cross-validation *5، ۵ بار 2-fold cross-validation را اجرا می کنیم. می توان خطای تست در این مورد با میانگین گرفتن از ۵ نتیجه‌ی2 fold cross-validation بدست آورد.

این روش که در واقع تکرار روش k-fold cv است باعث می شود تا تخمین بهتری از خطای تست بدست آید