ملزومات یادگیری ماشین

راهنمای عملی در R

Alboukadel KASSAMBARA

نسخه 1 sthda.com/english

ملزومات یادگیری ماشین

Alboukadel KASSAMBARA

حق چاپ ©2017 توسط Alboukadel Kassambara. تمامی حقوق محفوظ است.

**منتشر شده توسط STHDA (** <http://www.sthda.com>)، Alboukadel Kassambara

**تماس** : Alboukadel Kassambara < [alboukadel.kassambara@gmail.com](mailto:alboukadel.kassambara@gmail.com) >

هیچ بخشی از این نشریه را نمی‌توان بدون اجازه کتبی قبلی ناشر تکثیر کرد، در یک سامانه بازیابی ذخیره کرد، یا به هر شکل یا به هر وسیله ای، الکترونیکی، مکانیکی، فتوکپی، ضبط، اسکن یا موارد دیگر منتقل کرد. درخواست‌های دریافت مجوز از ناشر باید به STHDA ( <http://www.sthda.com>) ارسال شود.

محدودیت مسئولیت/ سلب مسئولیت از گارانتی: در حالی که ناشر و نویسنده تمام تلاش خود را در تهیه این کتاب به کار گرفته اند، هیچ گونه اظهارنظر یا ضمانتی در رابطه با صحت یا کامل بودن مطالب این کتاب نمی‌کنند و به‌طور خاص از هرگونه ضمانت ضمنی خرید و فروش بودن خودداری می‌کنند. یا برازش اندام برای یک هدف خاص هیچ ضمانتی نمی‌تواند توسط نمایندگان فروش یا مواد فروش کتبی ایجاد یا تمدید شود.

نه ناشر و نه نویسندگان، مشارکت‌کنندگان، یا ویراستاران هیچ‌گونه مسئولیتی در قبال آسیب و/یا آسیب وارده به اشخاص یا دارایی‌ها به دلیل مسئولیت محصولات، سهل‌انگاری یا موارد دیگر، یا استفاده یا عملیاتی از هر روش، محصول، دستورالعمل نمی‌پذیرند.، یا ایده‌های موجود در مطالب اینجا.

برای اطلاعات عمومی با Alboukadel Kassambara < [alboukadel.kassambara@gmail.com](mailto:alboukadel.kassambara@gmail.com) > تماس بگیرید.

فهرست

[0.1 آنچه یاد خواهید](#bookmark6)  [گرفت ix](#bookmark6)

[0.2 ویژگی‌های کلیدی این](#bookmark12)  [کتاب](#bookmark12)  [ix](#bookmark12)

[0.3 وب سایت کتاب](#bookmark14)  [x](#bookmark14)

[**درباره نویسنده**](#bookmark16) [**xi**](#bookmark16)

[**I**](#bookmark18) [**مبانی**](#bookmark18) [**1**](#bookmark18)

1. [**مقدمه‌ای بر**](#bookmark21) [**R2**](#bookmark21)
   1. [R و RStudio](#bookmark23)  [2 را نصب کنید](#bookmark23)
   2. [بسته‌های R مورد نیاز را نصب و بارگیری کنید](#bookmark25)  [2](#bookmark25)
   3. [فرمت داده](#bookmark29)  [3](#bookmark29)
   4. [داده‌های خود را در R](#bookmark31)  [3 وارد کنید](#bookmark31)
   5. [مجموعه داده‌های نمایشی](#bookmark33)  [3](#bookmark33)
   6. [دستکاری داده‌ها](#bookmark35)  [4](#bookmark35)
   7. [تجسم داده‌ها](#bookmark37)  [4](#bookmark37)
   8. [4 ببندید](#bookmark39)
2. **تحلیل رگرسیون 5**
3. [**مقدمه**](#bookmark42) [**6**](#bookmark42)
   1. [نمونه‌هایی از مجموعه داده](#bookmark44)  [8](#bookmark44)
4. [**رگرسیون خطی**](#bookmark53) [**10**](#bookmark53)
   1. [مقدمه](#bookmark55)  [10](#bookmark55)
   2. [فرمول](#bookmark57)  [11](#bookmark57)
   3. [بارگیری بسته‌های R مورد نیاز](#bookmark59)  [12](#bookmark59)
   4. [آماده‌سازی داده‌ها](#bookmark61)  [12](#bookmark61)
   5. [محاسبه رگرسیون خطی](#bookmark63)  [13](#bookmark63)
   6. [تفسیر](#bookmark71)  [15](#bookmark71)
   7. [پیش‌بینی](#bookmark83)  [18](#bookmark83)
   8. [بحث](#bookmark85)  [19](#bookmark85)
5. [**اثرات متقابل در رگرسیون چندگانه**](#bookmark88) [**21**](#bookmark88)
   1. [مقدمه](#bookmark90)  [21](#bookmark90)
   2. [معادله](#bookmark92)  [21](#bookmark92)
   3. [بارگیری بسته‌های R مورد نیاز](#bookmark94)  [22](#bookmark94)
   4. [آماده‌سازی داده‌ها](#bookmark96)  [22](#bookmark96)
   5. [محاسبات](#bookmark98)  [23](#bookmark98)
   6. [تفسیر](#bookmark104)  [24](#bookmark104)
   7. [مدل](#bookmark106)  [های افزودنی و برهمکنش](#bookmark106)  [25](#bookmark106)
   8. [بحث](#bookmark108)  [25](#bookmark108)
6. [**رگرسیون با متغیرهای طبقه‌ای**](#bookmark111) [**26**](#bookmark111)
   1. [مقدمه](#bookmark113)  [26](#bookmark113)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark115)  [26](#bookmark115)
   3. [نمونه‌ای از مجموعه داده](#bookmark117)  [26](#bookmark117)
   4. [متغیرهای طبقه‌بندی با دو سطح](#bookmark119)  [27](#bookmark119)
   5. [متغیرهای طبقه‌ای با بیش از دو سطح](#bookmark121)  [28](#bookmark121)
   6. [بحث](#bookmark123)  [30](#bookmark123)
7. [**رگرسیون غیرخطی**](#bookmark126) [**31**](#bookmark126)
   1. [مقدمه](#bookmark128)  [31](#bookmark128)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark130)  [31](#bookmark130)
   3. [آماده‌سازی داده‌ها](#bookmark132)  [32](#bookmark132)
   4. [رگرسیون خطی {linear-reg}](#bookmark134)  [32](#bookmark134)
   5. [رگرسیون چند جمله‌ای](#bookmark136)  [33](#bookmark136)
   6. [تبدیل لاگ](#bookmark138)  [35](#bookmark138)
   7. [رگرسیون اسپلاین](#bookmark140)  [36](#bookmark140)
   8. [مدل‌های افزودنی تعمیم یافته](#bookmark142)  [37](#bookmark142)
   9. [مقایسه مدل‌های](#bookmark144)  [38](#bookmark144)
   10. [بحث](#bookmark146)  [38](#bookmark146)

**III تشخیص رگرسیون 39**

1. [**مقدمه**](#bookmark149) [**40**](#bookmark149)
2. [**پیش فرض‌ها و تشخیص‌های رگرسیون**](#bookmark152) [**41**](#bookmark152)
   1. [مقدمه](#bookmark154)  [41](#bookmark154)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark156)  [42](#bookmark156)
   3. [نمونه داده](#bookmark158)  [42](#bookmark158)
   4. [ساخت مدل رگرسیون](#bookmark160)  [42](#bookmark160)
   5. [مقادیر مناسب و باقیمانده](#bookmark162)  [42](#bookmark162)
   6. [مفروضات رگرسیون](#bookmark164)  [44](#bookmark164)
   7. [تشخیص رگرسیون {reg-diag}](#bookmark168)  [44](#bookmark168)
   8. [خطی بودن داده‌ها](#bookmark172)  [46](#bookmark172)
   9. [همگنی واریانس](#bookmark174)  [47](#bookmark174)
   10. [نرمال بودن باقیمانده‌ها](#bookmark176)  [49](#bookmark176)
   11. [نقاط پرت و اهرم بالا](#bookmark178)  [50](#bookmark178)
   12. [مقادیر تأثیرگذار](#bookmark184)  [51](#bookmark184)
   13. [بحث](#bookmark186)  [53](#bookmark186)
3. [**چند خطی بودن**](#bookmark189) [**54**](#bookmark189)
   1. [مقدمه](#bookmark191)  [54](#bookmark191)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark193)  [54](#bookmark193)
   3. [آماده‌سازی داده‌ها](#bookmark195)  [54](#bookmark195)
   4. [ساخت مدل رگرسیون](#bookmark197)  [55](#bookmark197)
   5. [تشخیص چند خطی بودن](#bookmark199)  [55](#bookmark199)
   6. [مقابله با چند خطی](#bookmark201)  [55](#bookmark201)
   7. [بحث](#bookmark203)  [56](#bookmark203)
4. [**متغیرهای مخدوش کننده**](#bookmark206) [**57**](#bookmark206)

**اعتبار سنجی مدل رگرسیون IV 58**

1. [**مقدمه**](#bookmark209) [**59**](#bookmark209)
2. [**معیارهای دقت مدل رگرسیون**](#bookmark212) [**60**](#bookmark212)
   1. [مقدمه](#bookmark214)  [60](#bookmark214)
   2. [معیارهای عملکرد مدل](#bookmark216)  [60](#bookmark216)
   3. [بارگیری بسته‌های R مورد نیاز](#bookmark218)  [61](#bookmark218)
   4. [نمونه داده](#bookmark220)  [61](#bookmark220)
   5. [ساخت مدل‌های رگرسیون](#bookmark222)  [61](#bookmark222)
   6. [ارزیابی کیفیت مدل](#bookmark224)  [62](#bookmark224)
   7. [مقایسه عملکرد مدل‌های رگرسیون](#bookmark226)  [63](#bookmark226)
   8. [بحث](#bookmark228)  [63](#bookmark228)
3. [**اعتبارسنجی متقابل**](#bookmark231) [**65**](#bookmark231)
   1. [مقدمه](#bookmark233)  [65](#bookmark233)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark235)  [65](#bookmark235)
   3. [نمونه داده](#bookmark237)  [66](#bookmark237)
   4. [معیارهای عملکرد مدل](#bookmark239)  [66](#bookmark239)
   5. [روشهای اعتبارسنجی متقابل](#bookmark241)  [66](#bookmark241)
   6. [بحث](#bookmark251)  [70](#bookmark251)
4. [**نمونه برداری مجدد بوت استرپ**](#bookmark254) [**71**](#bookmark254)
   1. [مقدمه](#bookmark256)  [71](#bookmark256)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark258)  [71](#bookmark258)
   3. [نمونه داده](#bookmark260)  [71](#bookmark260)
   4. [روش بوت استرپ](#bookmark262)  [72](#bookmark262)
   5. [ارزیابی عملکرد یک مدل پیش‌بینی](#bookmark264)  [72](#bookmark264)
   6. [کمی کردن فاصله عدم قطعیت و اطمینان برآوردگر](#bookmark266)  [73](#bookmark266)
   7. [بحث](#bookmark268)  [74](#bookmark268)

**انتخاب مدل V 75**

[**15 مقدمه**](#bookmark271) [**76**](#bookmark271)

1. [**بهترین زیرمجموعه رگرسیون**](#bookmark274) [**77**](#bookmark274)
   1. [مقدمه](#bookmark276)  [77](#bookmark276)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark278)  [77](#bookmark278)
   3. [نمونه داده](#bookmark280)  [77](#bookmark280)
   4. [محاسبه بهترین زیرمجموعه رگرسیون](#bookmark282)  [77](#bookmark282)
   5. [انتخاب مدل بهینه](#bookmark284)  [78](#bookmark284)
   6. [بحث](#bookmark290)  [81](#bookmark290)
2. [**رگرسیون گام به گام**](#bookmark293) [**82**](#bookmark293)
   1. [مقدمه](#bookmark295)  [82](#bookmark295)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark297)  [82](#bookmark297)
   3. [محاسبه رگرسیون گام به گام](#bookmark299)  [83](#bookmark299)
   4. [بحث](#bookmark301)  [85](#bookmark301)
3. [**رگرسیون جریمه شده:**](#bookmark304) [**Ridge، Lasso و Elastic Net**](#bookmark304) [**87**](#bookmark304)
   1. [مقدمه](#bookmark306)  [87](#bookmark306)
   2. [روش‌های انقباض](#bookmark308)  [87](#bookmark308)
   3. [بارگیری بسته‌های R مورد نیاز](#bookmark316)  [89](#bookmark316)
   4. [آماده‌سازی داده‌ها](#bookmark318)  [89](#bookmark318)
   5. [محاسبه رگرسیون خطی جریمه شده](#bookmark320)  [89](#bookmark320)
   6. [بحث](#bookmark345)  [95](#bookmark345)
4. [**رگرسیون مولفه اصلی و حداقل مربعات جزئی**](#bookmark348) [**96**](#bookmark348)
   1. [مقدمه](#bookmark350)  [96](#bookmark350)
   2. [رگرسیون مؤلفه اصلی](#bookmark352)  [96](#bookmark352)
   3. [رگرسیون حداقل مربعات جزئی](#bookmark354)  [97](#bookmark354)
   4. [بارگیری بسته‌های R مورد نیاز](#bookmark356)  [97](#bookmark356)
   5. [تهیه داده‌ها](#bookmark358)  [97](#bookmark358)
   6. [محاسبات](#bookmark360)  [97](#bookmark360)
   7. [بحث](#bookmark366)  [100](#bookmark366)

**طبقه‌بندی VI 102**

1. [**مقدمه**](#bookmark369) [**103**](#bookmark369)
   1. [نمونه‌هایی از](#bookmark371)  [مجموعه داده](#bookmark371)  [103](#bookmark371)
2. [**رگرسیون لجستیک**](#bookmark378) [**105**](#bookmark378)
   1. [مقدمه](#bookmark380)  [105](#bookmark380)
   2. [تابع لجستیک](#bookmark382)  [105](#bookmark382)
   3. [بارگیری بسته‌های R مورد نیاز](#bookmark384)  [106](#bookmark384)
   4. [آماده‌سازی داده‌ها](#bookmark386)  [106](#bookmark386)
   5. [محاسبه رگرسیون لجستیک](#bookmark388)  [107](#bookmark388)
   6. [تفسیر](#bookmark396)  [109](#bookmark396)
   7. [پیش‌بینی](#bookmark398)  [110](#bookmark398)
   8. [ارزیابی دقت مدل](#bookmark402)  [بیمار](#bookmark402)
   9. [بحث](#bookmark404)  [بیمار](#bookmark404)
3. [**رگرسیون لجستیک گام به گام**](#bookmark407) [**113**](#bookmark407)
   1. [بارگیری بسته‌های R مورد نیاز](#bookmark409)  [113](#bookmark409)
   2. [آماده‌سازی داده‌ها](#bookmark411)  [113](#bookmark411)
   3. [محاسبه رگرسیون لجستیک گام به گام](#bookmark413)  [114](#bookmark413)
   4. [بحث](#bookmark423)  [115](#bookmark423)
4. [**رگرسیون لجستیک مجازات شده**](#bookmark426) [**116**](#bookmark426)
   1. [مقدمه](#bookmark428)  [116](#bookmark428)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark430)  [116](#bookmark430)
   3. [آماده‌سازی داده‌ها](#bookmark432)  [117](#bookmark432)
   4. [رگرسیون](#bookmark434)  [لجستیک جریمه شده](#bookmark434)  [117](#bookmark434)
   5. [بحث](#bookmark452)  [121](#bookmark452)
5. [**پیش فرض‌ها و تشخیص‌های رگرسیون لجستیک**](#bookmark455) [**122**](#bookmark455)
   1. [مقدمه](#bookmark457)  [122](#bookmark457)
   2. [مفروضات رگرسیون لجستیک](#bookmark459)  [122](#bookmark459)
   3. [بارگیری بسته‌های R مورد نیاز](#bookmark461)  [122](#bookmark461)
   4. [مدل](#bookmark463)  [رگرسیون لجستیک](#bookmark463)  [123](#bookmark463)
   5. [تشخیص رگرسیون لجستیک](#bookmark465)  [123](#bookmark465)
   6. [بحث](#bookmark473)  [126](#bookmark473)
6. [**رگرسیون لجستیک چند جمله‌ای**](#bookmark476) [**127**](#bookmark476)
   1. [مقدمه](#bookmark478)  [127](#bookmark478)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark480)  [127](#bookmark480)
   3. [آماده‌سازی داده‌ها](#bookmark482)  [127](#bookmark482)
   4. [محاسبه رگرسیون لجستیک چند جمله‌ای](#bookmark484)  [128](#bookmark484)
   5. [بحث](#bookmark486)  [128](#bookmark486)
7. [**تحلیل تفکیک کننده**](#bookmark489) [**129**](#bookmark489)
   1. [مقدمه](#bookmark491)  [129](#bookmark491)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark493)  [130](#bookmark493)
   3. [آماده‌سازی داده‌ها](#bookmark495)  [130](#bookmark495)
   4. [تحلیل تشخیص خطی - LDA](#bookmark497)  [130](#bookmark497)
   5. [تحلیل تفکیک درجه دوم - QDA](#bookmark507)  [133](#bookmark507)
   6. [تحلیل تفکیک مخلوط - MDA](#bookmark509)  [133](#bookmark509)
   7. [تحلیل تفکیک انعطاف‌پذیر - FDA](#bookmark511)  [134](#bookmark511)
   8. [تحلیل تفکیک منظم](#bookmark513)  [135](#bookmark513)
   9. [بحث](#bookmark515)  [135](#bookmark515)
8. [**طبقه‌بندی کننده ساده بیز**](#bookmark518) [**136**](#bookmark518)
   1. [مقدمه](#bookmark520)  [136](#bookmark520)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark522)  [136](#bookmark522)
   3. [آماده‌سازی داده‌ها](#bookmark524)  [136](#bookmark524)
   4. [Computing Naive Bayes](#bookmark526)  [137](#bookmark526)
   5. [با استفاده از پکیج caret R](#bookmark528)  [137](#bookmark528)
   6. [بحث](#bookmark530)  [137](#bookmark530)
9. [**پشتیبان ماشین بردار**](#bookmark533) [**138**](#bookmark533)
   1. [مقدمه](#bookmark535)  [138](#bookmark535)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark537)  [138](#bookmark537)
   3. [نمونه مجموعه داده](#bookmark539)  [138](#bookmark539)
   4. [طبقه‌بندی کننده خطی SVM](#bookmark541)  [139](#bookmark541)
   5. [طبقه‌بندی کننده SVM با استفاده از کرنل غیر خطی](#bookmark543)  [140](#bookmark543)
   6. [بحث](#bookmark547)  [141](#bookmark547)
10. [**ارزیابی مدل طبقه‌بندی**](#bookmark550) [**143**](#bookmark550)
    1. [مقدمه](#bookmark552)  [143](#bookmark552)
    2. [بارگیری بسته‌های R مورد نیاز](#bookmark554)  [143](#bookmark554)
    3. [ساخت مدل طبقه‌بندی](#bookmark556)  [144](#bookmark556)
    4. [دقت طبقه‌بندی کلی](#bookmark558)  [144](#bookmark558)
    5. [ماتریس درهم‌ریختگی](#bookmark560)  [145](#bookmark560)
    6. [دقت، یادآوری و ویژگی](#bookmark562)  [146](#bookmark562)
    7. [منحنی ROC](#bookmark564)  [148](#bookmark564)
    8. [تنظیمات چند کلاسه](#bookmark572)  [151](#bookmark572)
    9. [بحث](#bookmark574)  [152](#bookmark574)

**VII یادگیری ماشین آماری 153**

[**30 مقدمه**](#bookmark577) [**154**](#bookmark577)

[**31 KNN - k-نزدیکترین همسایگان**](#bookmark580) [**155**](#bookmark580)

* 1. [1 مقدمه](#bookmark582)  [155](#bookmark582)
  2. [2 الگوریتم KNN](#bookmark584)  [155](#bookmark584)
  3. [3 بارگیری بسته‌های R مورد نیاز](#bookmark592)  [156](#bookmark592)
  4. [4 طبقه‌بندی](#bookmark594)  [156](#bookmark594)
  5. [5 KNN برای رگرسیون](#bookmark600)  [157](#bookmark600)
  6. [6 بحث](#bookmark604)  [158](#bookmark604)

1. [**مدل‌های درخت تصمیم**](#bookmark607) [**160**](#bookmark607)
   1. [مقدمه](#bookmark609)  [160](#bookmark609)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark611)  [160](#bookmark611)
   3. [الگوریتم درخت تصمیم](#bookmark613)  [160](#bookmark613)
   4. [درختان طبقه‌بندی](#bookmark621)  [162](#bookmark621)
   5. [درختان رگرسیون](#bookmark629)  [167](#bookmark629)
   6. [درخت استنتاج مشروط](#bookmark635)  [169](#bookmark635)
   7. [بحث](#bookmark637)  [171](#bookmark637)
2. [**بگینگ و جنگل تصادفی**](#bookmark640) [**172**](#bookmark640)
   1. [مقدمه](#bookmark642)  [172](#bookmark642)
   2. [بارگیری بسته‌های R مورد نیاز](#bookmark644)  [172](#bookmark644)
   3. [طبقه‌بندی](#bookmark646)  [173](#bookmark646)
   4. [رگرسیون](#bookmark654)  [176](#bookmark654)
   5. [فراپارامترها](#bookmark660)  [177](#bookmark660)
   6. [بحث](#bookmark662)  [178](#bookmark662)
3. [**تقویت**](#bookmark665) [**179**](#bookmark665)
   1. [بارگیری بسته‌های R مورد نیاز](#bookmark667)  [179](#bookmark667)
   2. [طبقه‌بندی](#bookmark669)  [180](#bookmark669)
   3. [رگرسیون](#bookmark677)  [181](#bookmark677)
   4. [بحث](#bookmark683)  [182](#bookmark683)

[**هشتم یادگیری بدون نظارت**](#bookmark686) [**183**](#bookmark686)

**35 یادگیری بدون نظارت 184**

* 1. [مقدمه](#bookmark688)  [184](#bookmark688)
  2. [روش‌های مؤلفه اصلی](#bookmark690)  [184](#bookmark690)
  3. [بارگیری بسته‌های R مورد نیاز](#bookmark692)  [185](#bookmark692)
  4. [تحلیل خوشه‌ای](#bookmark700)  [191](#bookmark700)
  5. [بحث](#bookmark712)  [195](#bookmark712)

پیشگفتار

0.1 آنچه خواهید آموخت

هر روز حجم زیادی از داده‌ها در زمینه‌های مختلف از جمله بازاریابی، زیست پزشکی و امنیت ثبت می‌شود. برای کشف دانش از این داده ها، به تکنیک‌های یادگیری ماشین نیاز دارید که به دو دسته طبقه‌بندی می‌شوند:

1. روش‌های یادگیری ماشین بدون نظارت :

اینها عمدتاً شامل روش‌های *خوشه‌بندی* و *تحلیل مؤلفه‌های اصلی است* . هدف ­گروه‌بندی شناسایی الگو یا گروه‌هایی از اشیاء مشابه در یک مجموعه داده مورد علاقه است. روش‌های مؤلفه اصلی شامل خلاصه کردن و تجسم مهمترین اطلاعات موجود در یک مجموعه داده چند متغیره است.

این روش‌ها «بدون نظارت» هستند، زیرا ما با ایده‌های پیشینی هدایت نمی‌شویم که کدام متغیرها یا نمونه‌ها به کدام دسته‌ها یا گروه‌ها تعلق دارند. الگوریتم ماشین "می آموزد" که چگونه داده‌ها را خوشه‌بندی یا خلاصه کند.

1. روش‌های یادگیری ماشین تحت نظارت :

یادگیری تحت نظارت شامل ساخت مدل‌های ریاضی برای پیش‌بینی نتیجه مشاهدات آینده است. مدل‌های پیش‌بینی را می‌توان به دو گروه اصلی طبقه‌بندی کرد:

* *تحلیل رگرسیون* برای پیش‌بینی متغیر پیوسته. برای مثال، ممکن است بخواهید امید به زندگی را بر اساس شاخص‌های اجتماعی-اقتصادی پیش‌بینی کنید.
* *طبقه‌بندی* برای پیش‌بینی طبقه (یا گروه) افراد. به‌عنوان مثال، ممکن است بخواهید احتمال دیابت مثبت بودن را بر اساس غلظت گلوکز ­در پلاسمای بیماران پیش‌بینی کنید.

این روش‌ها تحت نظارت هستند زیرا ما مدل را بر اساس مقادیر نتیجه شناخته شده می‌سازیم. یعنی ماشین از نتایج مشاهدات شناخته شده یاد می‌گیرد تا نتیجه موارد آینده را پیش‌بینی کند.

در این کتاب، ما یک راهنمای عملی برای روش‌های یادگیری ماشین برای کاوش مجموعه داده‌ها و همچنین برای ساخت مدل‌های پیش‌بینی ارائه می‌کنیم.

شما ایده‌های اساسی هر روش و کدهای R قابل تکرار را برای محاسبه آسان تعداد زیادی از تکنیک‌های یادگیری ماشین یاد خواهید گرفت.

0.2 ویژگی‌های کلیدی این کتاب

هدف ما نوشتن یک راهنمای عملی برای یادگیری ماشین برای همه بود.

بخش‌های اصلی کتاب عبارتند از:

* **روش‌های یادگیری بدون نظارت،** برای کشف و کشف دانش از ­مجموعه داده‌های چند متغیره بزرگ با استفاده از روش‌های خوشه‌بندی و اجزای اصلی. شما روش‌های خوشه‌بندی مرتب، k-means، تحلیل مؤلفه‌های اصلی و روش‌های تحلیل مکاتبات را خواهید آموخت.
* **تحلیل رگرسیون،** برای پیش‌بینی یک مقدار نتیجه کمی با استفاده از رگرسیون خطی و استراتژی‌های رگرسیون غیر خطی.
* **تکنیک‌های طبقه‌بندی،** برای پیش‌بینی یک مقدار نتیجه کیفی با استفاده از رگرسیون لجستیک، تحلیل تفکیک‌کننده، طبقه‌بندی کننده ساده‌لوح و ماشین‌های بردار پشتیبان.
* **روش‌های یادگیری ماشین پیشرفته،** برای ساخت مدل‌های رگرسیون و طبقه‌بندی قوی با استفاده از روش‌های k-نزدیک‌ترین همسایه، مدل‌های درخت تصمیم، روش‌های مجموعه ( ­کیسه‌سازی، جنگل تصادفی و تقویت)
* **روش‌های انتخاب مدل،** برای انتخاب خودکار بهترین ترکیب از متغیرهای پیش‌بین برای ساخت یک مدل پیش‌بینی بهینه. اینها شامل بهترین روش‌های انتخاب زیر مجموعه، رگرسیون گام به گام و رگرسیون جریمه شده ( ­مدل‌های رگرسیون ریج، لاسو و رگرسیون شبکه الاستیک ) است. ما همچنین روش‌های رگرسیون مبتنی بر مؤلفه‌های اصلی را ارائه می‌کنیم که زمانی مفید هستند که داده‌ها حاوی متغیرهای پیش‌بینی کننده همبستگی متعددی باشند.
* **تکنیک‌های اعتبارسنجی و ارزیابی مدل** برای سنجش عملکرد یک مدل پیش‌بینی.
* **تشخیص مدل** برای تشخیص و رفع مسائل بالقوه در یک مدل پیش‌بینی.

این کتاب اصول اولیه این وظایف را ارائه می‌کند و مثال‌های زیادی را در R ارائه می‌کند.

ویژگی‌های کلیدی:

* الگوریتم یادگیری ماشین و پیاده‌سازی را پوشش می‌دهد
* مفاهیم کلیدی ریاضی ارائه شده است
* فصل‌های کوتاه و مستقل همراه با مثال‌های کاربردی. این به این معنی است که شما نیازی به خواندن فصل‌های مختلف به ترتیب ندارید.

در پایان هر فصل، بخش‌های آزمایشگاه R را ارائه می‌کنیم که در آن به‌طور سامانه‌اتیک از طریق کاربردهای روش‌های مختلف مورد بحث در آن فصل کار می‌کنیم.

0.3 وب سایت کتاب

<http://www.sthda.com/english>

درباره نویسنده

Alboukadel Kassambara دکترای بیوانفورماتیک و زیست شناسی سرطان است. او سال‌هاست روی تحلیل داده‌های ژنومی و تجسم کار می‌کند (بیشتر بخوانید: <http://www.alboukadel.com/>).

او دارای تجربیات کاری در روش‌های آماری و محاسباتی برای شناسایی علائم بیومارکر پیش آگهی و پیش‌بینی کننده از طریق تحلیل یکپارچه مجموعه داده‌های ژنومی و بالینی در مقیاس بزرگ است.

او یک ابزار وب بیوانفورماتیک به نام GenomicScape ( [www.genomicscape.com](http://www.genomicscape.com) ) ایجاد کرد که یک ابزار وب با استفاده آسان برای تحلیل و تجسم داده‌های بیان ژن است.

او همچنین یک وب‌سایت آموزشی در زمینه علم داده‌ها به نام STHDA (ابزارهای آماری برای تحلیل داده‌های با توان بالا، [www.sthda.com/english](http://www.sthda.com/english) ) ایجاد کرد که حاوی آموزش‌های زیادی در مورد تحلیل و تجسم داده‌ها با استفاده از نرم‌افزار و بسته‌های R است.

او نویسنده بسیاری از بسته‌های R محبوب برای موارد زیر است:

* تحلیل داده‌های چند متغیره ( **factoextra،** <http://www.sthda.com/english/rpkgs/>factoextra )،
* تحلیل بقا ( **survminer،** <http://www.sthda.com/english/rpkgs/survminer/>)،
* تحلیل همبستگی ( **ggcorrplot،** [http://www.sthda.com/english/wiki/ggcorrplot- visualization-of-a-correlation-matrix-using-ggplot2](http://www.sthda.com/english/wiki/ggcorrplot-visualization-of-a-correlation-matrix-using-ggplot2) )،
* ایجاد طرح‌های آماده انتشار در R ( **ggpubr،** <http://www.sthda.com/english/rpkgs/>ggpubr ).

او اخیراً چندین کتاب در زمینه تحلیل و تجسم داده منتشر کرده است:

1. راهنمای عملی تحلیل خوشه‌ای در R ( <https://goo.gl/yhhpXh>)
2. راهنمای عملی روش‌های اجزای اصلی در R ( <https://goo.gl/d4Doz9>)
3. R Graphics Essentials برای تجسم عالی داده‌ها ( <https://goo.gl/oT8Ra6>)
4. تحلیل شبکه و تجسم در R ( <https://goo.gl/WBdn4n>)

*درباره نویسنده*

xii

**قسمت اول**

مبانی

فصل 1

معرفی R

R یک نرم‌افزار آماری رایگان و قدرتمند برای تحلیل و تجسم داده‌ها است.

در این فصل، نحوه نصب R و بسته‌های مورد نیاز و همچنین نحوه وارد کردن داده‌های خود را به R یاد خواهید گرفت.

* 1. R و RStudio را نصب کنید

RStudio یک محیط توسعه یکپارچه برای R است که استفاده از R را آسان‌تر می‌کند. R و RStudio را می‌توان بر روی پلتفرم‌های Windows، MAC OSX و Linux نصب کرد.

1. R را می‌توان از صفحه وب شبکه جامع آرشیو R (CRAN) دانلود و نصب کرد ( <http://cran.r-project.org/>)
2. پس از نصب نرم‌افزار R، نرم‌افزار RStudio موجود در آدرس: http://www را نیز نصب کنید. rstu.dio.com/produ.cts/RStu.dio/ .
3. RStudio را راه اندازی کنید و از R در داخل R Studio استفاده کنید.
   1. بسته‌های R مورد نیاز را نصب و بارگیری کنید

بسته R مجموعه‌ای از قابلیت‌ها است که قابلیت‌های پایه R را گسترش می‌دهد. برای استفاده از کد R ارائه شده در این کتاب، باید بسته‌های R زیر را نصب کنید:

* tidyverse برای دستکاری و تجسم آسان داده ها.
* caret برای گردش کار آسان یادگیری ماشین.

1. نصب بسته‌ها :

mypkgs <- **c (**"tidyverse"، "caret")

**install.packages (**mypkgs)

3. **بسته‌های مورد نیاز را بارگیری** کنید. پس از نصب، ابتدا باید بسته را برای استفاده از توابع موجود در بسته بارگذاری کنید. تابع library() برای این کار استفاده می‌شود. برای مثال این را تایپ کنید:

**کتابخانه (**"tidyverse")

**کتابخانه (**"caret")

اکنون می‌توانیم از توابع R موجود در این بسته استفاده کنیم.

اگر می‌خواهید در مورد یک تابع داده شده بیشتر بدانید، بگویید mean()، این را در کنسول R تایپ کنید: ?mean .

* 1. فرمت داده

داده‌های شما باید در قالب مستطیل شکل باشند که در آن ستون‌ها متغیر و ردیف‌ها مشاهده (افراد یا نمونه) هستند.

* نام ستون‌ها باید با قراردادهای نامگذاری R سازگار باشد. از ستون‌هایی با فضای خالی و کاراکترهای خاص خودداری کنید. نام ستون‌های خوب: پرش بلند یا بلند پرش . نام ستون بد: پرش طول .
* از شروع نام ستون‌ها با عدد خودداری کنید. به جای آن از حرف استفاده کنید. نام ستون‌های خوب: sport\_100m یا xlOOm . نام ستون بد: 100 متر
* NA جایگزین کنید (برای موجود نیست)

به‌عنوان مثال، داده‌های شما باید به شکل زیر باشد:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| کاسبرگ. | .کاسبرگ طول.گلبرگ عرض | | .طول گلبرگ.عرض | | گونه ها |
| 44 | 5.0 | 3.5 | 1.6 | 0.6 | ستوزا |
| 118 | 7.7 | 3.8 | 6.7 | 2.2 | ویرجینیکا |
| 61 | 5.0 | 2.0 | 3.5 | 1.0 | رنگارنگ |
| 130 | 7.2 | 3.0 | 5.8 | 1.6 | ویرجینیکا |

بیشتر بخوانید در: بهترین روش‌ها در آماده‌سازی فایل‌های داده برای وارد کردن به R 1

* 1. داده‌های خود را در R وارد کنید

ابتدا داده‌های خود را در فرمت‌های فایل txt یا csv ذخیره کنید و آن را به صورت زیر وارد کنید (از شما خواسته می‌شود فایل را انتخاب کنید):

* *فایل‌های جدا شده از برگه را می‌خواند (برگه txt.)* my\_data <- **read.delim (** **file.choose (**))
* *فایل‌های محدود شده با کاما (،) (.csv)* my\_data <- **read.csv (** **file.choose (**)) را می‌خواند.
* *فایل‌های جدا شده با نقطه ویرگول (;) (.csv) را می‌خواند*

my\_data <- **read.csv2 (** **file.choose (**))

درباره نحوه وارد کردن داده‌ها به R در این پیوند بیشتر بخوانید: <http://www.sthda.com/english/>wiki/importing-data-into-r

* 1. مجموعه داده‌های نمایشی

R با چندین مجموعه داده آزمایشی برای بازی با توابع R ارائه می‌شود. پرکاربردترین مجموعه داده‌های نمایشی R شامل **زنبق** است. برای بارگیری یک مجموعه داده آزمایشی، از تابع **داده (**) به صورت زیر استفاده کنید. تابع head() برای بررسی داده‌ها استفاده می‌شود.

1. [http://www.sthda.com/english/wiki/best-practices-in-preparing-data-files-for-importing- into-r](http://www.sthda.com/english/wiki/best-practices-in-preparing-data-files-for-importing-into-r)

**داده (**"زنبق") *# در حال بارگیری*

**head (**زنبق، n = 3) *# n = 3 ردیف اول را چاپ کنید*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ## | کاسبرگ.کاسبرگ طول | | .گلبرگ عرض.گلبرگ طول.عرض | | | گونه ها |
| ## | 1 | 5.1 | 3.5 | 1.4 | 0.2 | ستوزا |
| ## | 2 | 4.9 | 3.0 | 1.4 | 0.2 | ستوزا |
| ## | 3 | 4.7 | 3.2 | 1.3 | 0.2 | ستوزا |

برای کسب اطلاعات بیشتر در مورد مجموعه داده‌های زنبق، این را تایپ کنید:

؟زنبق

پس از تایپ کد R بالا، توضیحات مجموعه داده زنبق را مشاهده خواهید کرد.

* 1. دستکاری داده

پس از وارد کردن داده‌های خود در R، می‌توانید به راحتی آن‌ها را با استفاده از بسته‌های [[1]](#footnote-1)tidyverse دستکاری کنید .

پس از بارگیری tidyverse، می‌توانید از توابع R زیر استفاده کنید:

* filter() : ردیف‌ها (مشاهدات/نمونه‌ها) را بر اساس مقادیرشان انتخاب کنید.
* distinct() : ردیف‌های تکراری را حذف کنید.
* arrange() : ردیف‌ها را دوباره مرتب کنید.
* select() : ستون‌ها (متغیرها) را با نام آنها انتخاب کنید.
* rename() : تغییر نام ستون ها.
* mutate() : افزودن/ایجاد متغیرهای جدید.
* summarise() : خلاصه‌های آماری را محاسبه کنید (مثلاً محاسبه میانگین یا مجموع)
* group\_by() : روی زیرمجموعه‌های مجموعه داده عمل می‌کند.

توجه داشته باشید که بسته‌های tidyverse امکان استفاده از عملگر زنجیره‌ای لوله‌های جلو (%>%) را برای ترکیب چندین عملیات فراهم می‌کنند. برای مثال x %>% f معادل f(x) است. با استفاده از لوله (%>%) خروجی هر عملیات به عملیات بعدی منتقل می‌شود. این امر برنامه نویسی R را آسان می‌کند.

بیشتر در: [http://r4ds.had.co.nz/transform.html بخوانید](http://r4ds.had.co.nz/transform.html) .

* 1. تجسم داده ها

بسته‌های گرافیکی مختلفی در R برای تجسم داده‌های شما موجود است. در این کتاب، عمدتاً با استفاده از سامانه ggplot2 که متعلق به بسته‌های tidyverse است، گرافیک ایجاد می‌کنیم.

اگر در ggplot2 مبتدی هستید، آموزش ما را در مورد R Graphics Essantials بخوانید [[2]](#footnote-2).

* 1. جلسه R/RStudio خود را ببندید

هر بار که R/RStudio را می‌بندید، از شما پرسیده می‌شود که آیا می‌خواهید داده‌های جلسه R خود را ذخیره کنید یا خیر. اگر تصمیم به ذخیره داشته باشید، داده‌ها در جلسات R آینده در دسترس خواهند بود.

**قسمت دوم**

**تحلیل رگرسیون**

فصل 2

مقدمه

**تحلیل رگرسیون (**یا **مدل رگرسیون** ) شامل مجموعه‌ای از روش‌های *یادگیری ماشین است* که به ما امکان می‌دهد یک متغیر نتیجه پیوسته (y) را بر اساس مقدار یک یا چند متغیر پیش‌بینی‌کننده (x) پیش‌بینی کنیم.

به‌طور خلاصه، هدف مدل رگرسیون ساخت یک معادله ریاضی است که y را تابعی از متغیرهای x تعریف می‌کند. در مرحله بعد، می‌توان از این معادله برای پیش‌بینی نتیجه (y) بر اساس مقادیر جدید متغیرهای پیش‌بینی کننده (x) استفاده کرد.

**رگرسیون خطی** ساده ترین و محبوب ترین تکنیک برای پیش‌بینی متغیر پیوسته ­است. یک رابطه خطی بین نتیجه و متغیرهای پیش‌بینی کننده را فرض می‌کند. به فصل 3 مراجعه کنید.

معادله رگرسیون خطی را می‌توان به صورت y = bO + b\*x نوشت که در آن:

* bO عرض از مبدأ است،
* b وزن یا ضریب رگرسیون مرتبط با متغیر پیش‌بینی x است.

از نظر فنی، ضرایب رگرسیون خطی به گونه‌ای تعیین می‌شوند که خطا در پیش‌بینی مقدار نتیجه به حداقل برسد. این روش محاسبه ضرایب بتا را روش **حداقل مربعات معمولی می‌نامند** .

هنگامی که چندین متغیر پیش‌بینی دارید، مثلاً xl و x2، معادله رگرسیون را می‌توان به صورت y = bO + bl\*xl + b2\*x2 نوشت . در برخی شرایط، ممکن است یک **اثر متقابل** بین برخی از پیش‌بینی کننده‌ها وجود داشته باشد، به‌عنوان مثال، افزایش مقدار متغیر پیش‌بینی xl ممکن است اثربخشی پیش‌بینی x2 را در توضیح تغییر متغیر نتیجه افزایش دهد. فصل 4 را ببینید.

همچنین توجه داشته باشید که مدل‌های رگرسیون خطی می‌توانند **متغیرهای پیش‌نمایش پیوسته و طبقه‌ای را در بر گیرند ­**. فصل 5 را ببینید.

هنگامی که مدل رگرسیون خطی را می‌سازید، باید **تشخیص دهید** که آیا مدل خطی برای داده‌های شما مناسب است یا خیر. فصل 8 را ببینید.

در برخی موارد، رابطه بین پیامد و متغیرهای پیش‌بینی خطی نیست. در این مواقع، شما باید یک **رگرسیون غیر خطی** مانند رگرسیون *چند جمله‌ای و اسپلاین بسازید* . فصل 6 را ببینید.

هنگامی که چندین پیش‌بینی در مدل رگرسیون دارید، ممکن است بخواهید بهترین ترکیب از متغیرهای پیش‌بینی کننده را برای ساخت یک مدل پیش‌بینی بهینه انتخاب کنید. این فرآیند که **انتخاب مدل نامیده می‌شود،** شامل مقایسه مدل‌های متعدد حاوی مجموعه‌های مختلف پیش‌بینی کننده‌ها به منظور انتخاب بهترین مدلی است که خطای پیش‌بینی را به حداقل می‌رساند. رویکردهای انتخاب مدل خطی شامل **رگرسیون بهترین زیرمجموعه (**فصل 16) و **رگرسیون گام به گام (**فصل 17) است.

در برخی موقعیت‌ها، مانند زمینه‌های ژنومی، ممکن است مجموعه داده‌های چند متغیره بزرگی داشته باشید که ­حاوی برخی پیش‌بینی‌کننده‌های مرتبط باشد. در این مورد، اطلاعات در مجموعه داده‌های اصلی را می‌توان در چند متغیر جدید (به نام اجزای اصلی) که ترکیبی خطی ­از متغیرهای اصلی هستند، خلاصه کرد. این چند مؤلفه اصلی را می‌توان برای ساخت یک مدل خطی استفاده کرد که ممکن است برای داده‌های شما کارایی بیشتری داشته باشد. این رویکرد به‌عنوان روش‌ها**ی مبتنی بر مولفه اصلی شناخته می‌شود (**فصل 19) که شامل: **رگرسیون مولفه اصلی ­**و رگرسیون حداقل **مربعات جزئی** است.

یک روش جایگزین برای ساده‌سازی یک مدل چند متغیره بزرگ، استفاده از **رگرسیون جریمه شده است (**فصل 18) که مدل را به دلیل داشتن متغیرهای زیاد جریمه می‌کند. شناخته شده ترین رگرسیون جریمه شده شامل رگرسیون ریجو رگرسیون **لاسو** است.

شما می‌توانید تمام این مدل‌های رگرسیون مختلف را روی داده‌های خود اعمال کنید، مدل‌ها را مقایسه کنید و در نهایت بهترین رویکردی را انتخاب کنید که به خوبی داده‌های شما را توضیح می‌دهد. برای انجام این کار، به برخی معیارهای آماری برای مقایسه عملکرد مدل‌های مختلف در توضیح داده‌های خود و پیش‌بینی نتیجه داده‌های آزمایش جدید نیاز دارید.

بهترین مدل به مدلی گفته می‌شود که کمترین خطای پیش‌بینی را داشته باشد. محبوب ترین معیارها برای مقایسه مدل‌های رگرسیون عبارتند از:

* **ریشه میانگین مربعات خطا که** خطای پیش‌بینی مدل را اندازه‌گیری می‌کند. این ­با میانگین تفاوت بین مقادیر مشاهده شده شناخته شده نتیجه و مقدار پیش‌بینی شده توسط مدل مطابقت دارد. RMSE به صورت RMSE = mean((مشاهده - پیش‌بینی)~2) %>% sqrt() محاسبه می‌شود. هرچه RMSE کمتر باشد، مدل بهتر است.
* **مربع R تنظیم شده،** نشان دهنده نسبت تغییرات (یعنی اطلاعات)، در داده‌های شما که توسط مدل توضیح داده شده است. این با کیفیت کلی مدل مطابقت دارد. هرچه R2 تنظیم شده بالاتر باشد، مدل بهتر است

توجه داشته باشید که معیارهای ذکر شده در بالا باید بر روی یک داده آزمایشی جدید که برای آموزش (یعنی ساخت) مدل استفاده نشده است، محاسبه شود. اگر مجموعه داده‌ای بزرگ دارید، با رکوردهای زیاد، می‌توانید به‌طور تصادفی داده‌ها را به مجموعه آموزشی (۸۰٪ برای ساخت مدل پیش‌بینی‌کننده) و مجموعه تست یا مجموعه اعتبارسنجی (۲۰٪ برای ارزیابی عملکرد مدل) تقسیم کنید.

یکی از قوی‌ترین و محبوب‌ترین رویکردها برای تخمین عملکرد مدل، **اعتبارسنجی متقاطع k-fold است** . می‌توان آن را حتی بر روی یک مجموعه داده کوچک اعمال کرد. اعتبارسنجی متقابل k-fold به شرح زیر عمل می‌کند:

1. به‌طور تصادفی مجموعه داده‌ها را به k-زیر مجموعه (یا k-fold) تقسیم کنید (به‌عنوان مثال 5 زیر مجموعه)
2. یک زیر مجموعه را رزرو کنید و مدل را روی همه زیر مجموعه‌های دیگر آموزش دهید
3. مدل را روی زیر مجموعه رزرو شده تست کنید و خطای پیش‌بینی را ثبت کنید
4. این فرآیند را تا زمانی تکرار کنید که هر یک از k زیر مجموعه به‌عنوان مجموعه آزمایشی عمل کنند.
5. میانگین k خطای ثبت شده را محاسبه کنید. به این خطای اعتبارسنجی متقابل می‌گویند که به‌عنوان معیار عملکرد مدل عمل می‌کند.

در مجموع، بهترین مدل مدلی است که کمترین خطای اعتبارسنجی متقاطع، RMSE را دارد.

در این قسمت روش‌های مختلف تحلیل رگرسیون را یاد می‌گیرید و مثال عملی را در **R ارائه می‌دهیم** . تکنیک‌های زیر شرح داده شده است:

• حداقل مربعات معمولی (فصل 3)

* رگرسیون خطی ساده
* رگرسیون خطی چندگانه

• روش‌های انتخاب مدل:

* رگرسیون بهترین زیر مجموعه‌ها (فصل 16)
* رگرسیون گام به گام (فصل 17)

• روش‌های مبتنی بر مؤلفه اصلی (فصل 19):

* رگرسیون مؤلفه اصلی (PCR)
* رگرسیون حداقل مربعات جزئی (PLS)

• رگرسیون مجازات شده (فصل 18):

* رگرسیون ریج
* رگرسیون لاسو
  1. نمونه‌هایی از مجموعه داده ها

ما از سه مجموعه داده مختلف استفاده خواهیم کرد: بازاریابی [بسته داده]، مجموعه داده داخلی R سوئیس و مجموعه داده بوستون موجود در بسته MASS R.

* + 1. داده‌های بازاریابی

مجموعه داده‌های بازاریابی [بسته داده] حاوی تأثیر سه رسانه تبلیغاتی (یوتیوب، فیس بوک و روزنامه) بر فروش است. برای پیش‌بینی واحدهای فروش بر اساس میزان پول صرف شده در سه رسانه تبلیغاتی استفاده می‌شود.

داده‌ها بودجه تبلیغاتی به هزار دلار همراه با فروش است. آزمایش تبلیغات 200 بار با بودجه‌های مختلف تکرار شده و فروش مشاهده شده ثبت شده است.

ابتدا بسته دادهاریوم را نصب کنید :

**if (** **'require (**devtools)) **install.packages (**"devtools")

devtools **::install\_github (**"kassambara/datarium")

سپس مجموعه داده‌های بازاریابی را به صورت زیر بارگیری کنید: **داده (**"بازاریابی"، بسته = "datarium") **سر (**بازاریابی، 3)

* # فروش روزنامه فیس بوک یوتیوب
* # 1 276.1 45.4 83.0 26.5
* # 2 53.4 47.2 54.1 12.5
* # 3 20.6 55.1 83.2 11.2

2.1.2 داده‌های سوئیس

سوئیس 5 شاخص اجتماعی-اقتصادی را توصیف می‌کند که در حدود سال 1888 مشاهده شد که برای پیش‌بینی امتیاز باروری 47 استان فرانسوی زبان سوئیس استفاده شد.

داده‌ها را بارگیری و بررسی کنید:

**داده (**"سوئیسی")

**سر (**سوئیسی، 3)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ##  ## | دادگاه | آزمون باروری کشاورزی آموزش کاتولیک | | | | |
| 80.2 | 17.0 | 15 | 12 | 9.96 |
| ## | دیلمونت | 83.1 | 45.1 | 6 | 9 | 84.84 |
| ## | Franches-Mnt | 92.5 | 39.7 | 5 | 5 | 93.40 |

##

Infant.Mortality

## دادگاه 22.2

## Delemont 22.2

## Franches-Mnt 20.2

داده‌ها حاوی متغیرهای زیر هستند:

* Ig باروری: معیار استاندارد باروری رایج
* کشاورزی: درصد مردانی که به‌عنوان شغل به کشاورزی مشغول هستند
* معاینه: درصد سربازان وظیفه بالاترین نمره را در آزمون ارتش دریافت کردند
* تحصیلات: % آموزش فراتر از دبستان برای سربازان سرباز.
* کاتولیک: % «کاتولیک» (در مقابل «پروتستان»).
* مرگ و میر نوزادان: تولدهایی که کمتر از 1 سال زندگی می‌کنند.

2.1.3 داده‌های بوستون

بوستون [در بسته MASS ] برای پیش‌بینی مقدار خانه متوسط ( mdev )، در حومه بوستون، با استفاده از متغیرهای پیش‌بین مختلف استفاده می‌شود:

* جرم، سرانه جرم و جنایت بر اساس شهر
* zn، نسبت زمین مسکونی پهنه‌بندی شده برای زمین‌های بیش از 25000 فوت مربع
* هکتارهای تجاری غیر خرده فروشی در هر شهر
* chas، متغیر ساختگی رودخانه چارلز (= 1 اگر مسیر به رودخانه محدود می‌شود؛ 0 در غیر این صورت)
* nox، غلظت اکسیدهای نیتریک (قسمت در هر 10 میلیون)
* rm، میانگین تعداد اتاق در هر خانه
* سن، نسبت واحدهای تحت اشغال ساخته شده قبل از سال 1940
* dis , فاصله‌های وزنی تا پنج مرکز استخدامی بوستون
* راد، شاخص دسترسی به بزرگراه‌های شعاعی
* مالیات، نرخ مالیات بر دارایی با مقدار کامل به ازای هر 10000 دلار آمریکا
* ptratio، نسبت دانش آموز به معلم بر اساس شهر
* سیاه، 1000(B - 0.63)^2 که در آن B نسبت سیاهپوستان بر اساس شهر است
* ایستات، درصد پایین بودن وضعیت جمعیت
* medv، مقدار متوسط خانه‌های تحت سکونت در 1000 دلار آمریکا

داده‌ها را بارگیری و بررسی کنید:

**داده (**"بوستون"، بسته = "MASS")

**سر (**بوستون، 3)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ## | | 0. | جرم .00632 | zn indus | | | چس | | نوکس  .538 6 | rm 0.58 | سن  65.2 | دیس  4.09 | راد  1 | مالیات  296 | ptratio | | سیاه  397 | lstat | |
| ## | 1 | 18 | 2. | .31 | 0 | 0 | 15. | .3 | 4. | .98 |
| ## | 2 | 0. | .02731 | 0 | 7. | .07 | 0 | 0 | .469 6 | .42 | 78.9 | 4.97 | 2 | 242 | 17. | .8 | 397 | 9. | . 14 |
| ##  ## | 3 | 0.02729 medv | | 0 | 7. | .07 | 0 | 0 | .469 7 | .18 | 61.1 | 4.97 | 2 | 242 | 17. | .8 | 393 | 4. | 0.03 |

* # 1 24.0
* # 2 21.6
* # 3 34.7

فصل 3

رگرسیون خطی

* 1. مقدمه

**رگرسیون خطی (**یا **مدل خطی** ) برای پیش‌بینی یک متغیر نتیجه کمی (y) بر اساس یک یا چند متغیر پیش‌بینی‌کننده (x) استفاده می‌شود (جیمز و همکاران، 2014، بروس و بروس (2017)).

هدف ساخت یک فرمول ریاضی است که y را تابعی از متغیر x تعریف می‌کند. هنگامی که یک مدل آماری معنی‌دار ساختیم، می‌توان از آن برای پیش‌بینی نتیجه آینده بر اساس مقادیر x جدید استفاده کرد.

هنگامی که یک مدل رگرسیونی ایجاد می‌کنید، باید عملکرد مدل پیش‌بینی را ارزیابی کنید. به عبارت دیگر، باید ارزیابی کنید که مدل در پیش‌بینی نتیجه یک داده آزمایش جدید که برای ساخت مدل استفاده نشده است، چقدر خوب است.

دو معیار مهم معمولا برای ارزیابی عملکرد مدل رگرسیون پیش‌بینی استفاده می‌شود:

* **ریشه میانگین مربعات خطا که** خطای پیش‌بینی مدل را اندازه‌گیری می‌کند. این ­با میانگین تفاوت بین مقادیر مشاهده شده شناخته شده نتیجه و مقدار پیش‌بینی شده توسط مدل مطابقت دارد. RMSE به صورت RMSE = mean((مشاهده - پیش‌بینی)~2) %>% sqrt() محاسبه می‌شود. هرچه RMSE کمتر باشد، مدل بهتر است.
* **R-square،** نشان دهنده همبستگی مجذور بین مقادیر نتیجه شناخته شده مشاهده شده و مقادیر پیش‌بینی شده توسط مدل است. هرچه R2 بالاتر باشد، مدل بهتر است.

یک گردش کار ساده برای ساختن یک مدل رگرسیون پیش‌بینی به شرح زیر است:

1. به‌طور تصادفی داده‌های خود را به مجموعه آموزشی (80%) و مجموعه تست (20%) تقسیم کنید.
2. مدل رگرسیون را با استفاده از مجموعه آموزشی بسازید
3. با استفاده از مجموعه تست پیش‌بینی کنید و معیارهای دقت مدل را محاسبه کنید

در این فصل یاد خواهید گرفت:

* مبانی و فرمول رگرسیون خطی،
* نحوه محاسبه مدل‌های رگرسیون ساده و چندگانه در R،
* چگونه می‌توان نتیجه داده‌های جدید را پیش‌بینی کرد،
* نحوه ارزیابی عملکرد مدل

3.2 فرمول

فرمول ریاضی رگرسیون خطی را می‌توان به صورت زیر نوشت:

y = bO + bl\*x + e

ما این را به این صورت می‌خوانیم که "y به صورت betal ( bl ) ضربدر x، به اضافه یک betaO ثابت (bO )، به اضافه یک جمله خطا e مدل شده است."

هنگامی که چندین متغیر پیش‌بینی دارید، معادله را می‌توان به صورت y = bO + bl\*xl + b2\*x2 + ... + bn\*xn نوشت که در آن:

* bO عرض از مبدأ است،
* bl، b2، ...، bn وزن‌ها یا ضرایب رگرسیونی مرتبط با پیش‌بینی کننده‌های xl، x2، ...، xn هستند.
* e عبارت *خطا (*همچنین به‌عنوان *خطاهای باقیمانده شناخته می‌شود* )، بخشی از y است که می‌تواند توسط مدل رگرسیون توضیح داده شود.

توجه داشته باشید که bO، bl، b2، ... و bn به‌عنوان ضرایب یا پارامترهای رگرسیون بتا شناخته می‌شوند.

شکل زیر یک مدل رگرسیون خطی ساده را نشان می‌دهد که در آن:

* خط رگرسیون مناسب به رنگ آبی است
* نقطه قطع (bO) و شیب (bl) به رنگ سبز نشان داده شده است
* عبارات خطا (e) با خطوط قرمز عمودی نشان داده می‌شود

از نمودار پراکندگی بالا، می‌توان دریافت که همه نقاط داده دقیقاً روی خط رگرسیون برازش قرار نمی‌گیرند. برخی از نقاط بالای منحنی آبی و برخی در زیر آن قرار دارند. به‌طور کلی، خطاهای باقیمانده (e) تقریباً میانگین صفر دارند.

مجموع مربعات خطاهای **باقیمانده را مجموع مربعات باقیمانده** یا **RSS می‌نامند** .

میانگین تغییرات نقاط اطراف خط رگرسیون برازش شده، **خطای Standard باقیمانده (** **RSE** ) نامیده می‌شود. این یکی از معیارهایی است که برای ارزیابی کیفیت کلی مدل رگرسیون برازش استفاده می‌شود. هرچه RSE کمتر باشد، بهتر است.

از آنجایی که میانگین عبارت خطا صفر است، متغیر نتیجه y را می‌توان تقریباً به صورت زیر تخمین زد:

y ~ bO + bl\*x

از نظر ریاضی، ضرایب بتا (bO و bl) به گونه‌ای تعیین می‌شوند که RSS تا حد امکان حداقل باشد. این روش برای تعیین ضرایب بتا از نظر فنی رگرسیون **حداقل مربعات** یا رگرسیون **حداقل مربعات معمولی (**OLS) نامیده می‌شود.

پس از محاسبه ضرایب بتا، آزمون t انجام می‌شود تا بررسی شود که آیا این ضرایب به‌طور قابل‌توجهی با صفر متفاوت هستند یا خیر. ضرایب بتا غیر صفر به این معنی است که بین پیش‌بینی کننده‌ها (x) و متغیر نتیجه (y) رابطه معنی‌داری وجود دارد.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید

**کتابخانه (**tidyverse)

**کتابخانه (**کارت)

**مجموعه\_تم (** **theme\_bw (**))

* 1. آماده‌سازی داده ها

ما از مجموعه داده‌های بازاریابی، معرفی شده در فصل 2، برای پیش‌بینی واحدهای فروش بر اساس میزان پول صرف شده در سه رسانه تبلیغاتی (یوتیوب، فیس بوک و روزنامه) استفاده خواهیم کرد.

ما داده‌ها را به صورت تصادفی به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم می‌کنیم. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *داده‌ها را بارگذاری کنید*

**داده (**"بازاریابی"، بسته = "datarium")

* *داده‌ها را بررسی کنید*

**sample\_n (**بازاریابی، 3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ## | یوتیوب | فیس بوک | روزنامه | حراجی |
| ## 158 | 18O | 1.56 | 29.2 | 12.1 |
| ## 82 | 288 | 4.92 | 44.3 | 14.8 |
| ## 175 | 267 | 4.O8 | 15.7 | 13.8 |

* *تقسیم داده‌ها به مجموعه آموزشی و تست* **set.seed (**l23)

training.samples <- marketing$sales °/ o >°/o

**createDataPartition (**p = O.8، list = FALSE) train.data <- marketingftraining.samples، ] test.data <- marketingf **-** training.samples، ]

3.5 محاسبه رگرسیون خطی

تابع R lm() برای محاسبه مدل رگرسیون خطی استفاده می‌شود.

* + 1. کد R شروع سریع
* *مدل را بسازید*

مدل <- **lm (**فروش **~** .، داده = train.data)

* *مدل را خلاصه کنید*

**خلاصه (**مدل)

* *پیش‌بینی کنید*

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**test.data)

* *عملکرد مدل*
* *(الف) خطای پیش‌بینی، RMSE*

**RMSE (**پیش‌بینی‌ها، test.data **$** فروش)

* *(ب) R-square*

**R2 (**پیش‌بینی‌ها، test.data **$** فروش)

* + 1. رگرسیون خطی ساده

**رگرسیون خطی ساده** برای پیش‌بینی یک متغیر نتیجه پیوسته (y) بر اساس یک متغیر پیش‌بینی‌کننده منفرد (x) استفاده می‌شود.

در مثال زیر، یک مدل خطی ساده برای پیش‌بینی واحدهای فروش بر اساس بودجه تبلیغاتی صرف شده در یوتیوب می‌سازیم. معادله رگرسیون را می‌توان به صورت فروش = bO + bl\*youtube نوشت .

تابع R lm() را می‌توان برای تعیین ضرایب بتای مدل خطی به صورت زیر استفاده کرد:

مدل <- **Im (**فروش **~** یوتیوب، داده = train.data) **خلاصه (**مدل) **$** coef

* # Estimate Std. خطای t مقدار Pr(>|t|)
* # (Intercept) 8.3839 O.62442 13.4 5.22e-28
* # youtube O.O468 O.OO3O1 15.6 7.84e-34

خروجی بالا تخمین ضرایب بتای رگرسیون (تخمین ستونی) و سطوح معنی‌داری آنها (ستون Pr(>|t|) را نشان می‌دهد. فاصله ( bO ) 8.38 و ضریب متغیر یوتیوب 0.046 است.

معادله رگرسیون برآورد شده را می‌توان به صورت زیر نوشت: فروش = 8.38 + O.O46\*youtube . با استفاده از این فرمول، برای هر بودجه تبلیغاتی جدید یوتیوب، می‌توانید تعداد واحدهای فروش را پیش‌بینی کنید.

مثلا:

* برای بودجه تبلیغاتی یوتیوب برابر با صفر، می‌توان انتظار فروش 8.38 واحد را داشت.
* برای بودجه تبلیغاتی یوتیوب برابر با 1000، می‌توان انتظار فروش 8.38 + 0.046 \* 1000 = 55 واحد را داشت.

با استفاده از تابع R () predict می‌توان پیش‌بینی‌ها را به راحتی انجام داد . در مثال زیر، واحدهای فروش را برای دو بودجه تبلیغاتی یوتیوب پیش‌بینی می‌کنیم: 0 و 1000.

newdata <- **data.frame (**youtube = **c (**0, 1000)) model **/,>/. پیش‌بینی (**داده‌های جدید)

##12

## 8.38 55.19

3.5.3 رگرسیون خطی چندگانه

**رگرسیون خطی چندگانه** توسعه رگرسیون خطی ساده برای پیش‌بینی ­متغیر خروجی (y) بر اساس متغیرهای پیش‌بینی‌کننده متمایز چندگانه (x) است.

به‌عنوان مثال، با سه متغیر پیش‌بینی کننده (x)، پیش‌بینی y با معادله زیر بیان می‌شود: y = b0 + b1\*x1 + b2\*x2 + b3\*x3.

ضرایب رگرسیون بتا ارتباط بین هر متغیر پیش‌بینی کننده و نتیجه را اندازه‌گیری می‌کند. "b\_j" را می‌توان به‌عنوان اثر متوسط روی y یک واحد افزایش در "x\_j" تعبیر کرد که همه پیش‌بینی کننده‌های دیگر را ثابت نگه می‌دارد.

در این بخش، یک مدل رگرسیون چندگانه برای پیش‌بینی فروش بر اساس بودجه سرمایه گذاری شده در سه رسانه تبلیغاتی ایجاد می‌کنیم: یوتیوب، فیس بوک و روزنامه. فرمول به شرح زیر است: فروش = b0 + b1 \* یوتیوب + b2 \* فیس بوک + b3 \* روزنامه

می توانید ضرایب مدل رگرسیون چندگانه را در R به صورت زیر محاسبه کنید:

مدل <- **Im (**فروش **~** یوتیوب **+** فیس بوک **+** روزنامه، داده = train.data)

**خلاصه (**مدل) **$** coef

توجه داشته باشید که اگر متغیرهای پیش‌بینی کننده زیادی در داده‌های خود دارید، می‌توانید به سادگی تمام متغیرهای موجود را با استفاده از ~ در مدل قرار دهید. :

مدل <- **lm (**فروش **-** .، داده = train.data)

**خلاصه (**مدل) **$** coef

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ## | تخمین زدن | Std. | خطا | مقدار t | Pr(>|t|) |
| ## (عرض از مبدأ) | 3.39188 | 0. | .44062 | 7.698 | 1.41e-12 |
| ## یوتیوب | 0.04557 | 0. | 00159 | 28.630 | 2.03e-64 |
| ## فیس بوک | 0.18694 | 0. | 00989 | 18.905 | 2.07e-42 |
| ## روزنامه | 0.00179 | 0. | 00677 | 0.264 | 7.92e-01 |

از خروجی بالا، جدول ضرایب تخمین ضرایب بتا و سطوح اهمیت آنها را نشان می‌دهد. ستون‌ها عبارتند از:

* تخمین : تخمین‌های وقفه (bO) و ضریب بتا مربوط به هر ­متغیر پیش‌بینی کننده
* Std.Error : خطای استاندارد تخمین ضریب. این نشان دهنده دقت ضرایب است. هر چه خطای استاندارد بزرگتر باشد، اطمینان کمتری نسبت به برآورد داریم.
* مقدار t : آماره t که تخمین ضریب (ستون 2) تقسیم بر ­خطای استاندارد برآورد (ستون 3) است.
* Pr(>|t|) : مقدار p مربوط به آماره t. هرچه مقدار p کوچکتر باشد، تخمین مهمتر است.

همانطور که قبلا توضیح داده شد، می‌توانید به راحتی با استفاده از تابع R predict پیش‌بینی کنید :

* *بودجه تبلیغاتی جدید*

newdata <- **data.frame (**

یوتیوب = 2000، فیس بوک = 1000، روزنامه = 1000

)

* *پیش‌بینی مقادیر فروش*

مدل **/,>/. پیش‌بینی (**داده‌های جدید)

## 1

## 283

3.6 تفسیر

قبل از استفاده از یک مدل برای پیش‌بینی، باید اهمیت آماری مدل را ارزیابی کنید. این را می‌توان با نمایش خلاصه آماری مدل به راحتی بررسی کرد.

* + 1. خلاصه مدل

خلاصه آماری مدل را به صورت زیر نمایش دهید:

**خلاصه (**مدل) ##

* # زنگ زدن:
* # lm(فرمول = فروش - .، داده = train.data)
* #
* # باقیمانده ها:
* # حداقل 1Q میانه 3Q حداکثر
* # -10.412 -1.110 0.348 1.422 3.499
* #
* # ضرایب:
* # Estimate Std. خطای t مقدار Pr(>|t|)
* # (Intercept) 3.39188 0.44062 7.70 1.4e-12 \*\*\*
* # youtube 0.04557 0.00159 28.63 < 2e-16 \*\*\*
* # facebook 0.18694 0.00989 18.90 < 2e-16 \*\*\*
* # روزنامه 0.00179 0.00677 0.26 0.79
* #
* # Signif. کدها: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 1 1 1
* #
* # خطای استاندارد باقیمانده: 2.12 در 158 درجه آزادی
* # R-squared چندگانه: 0.89، R-squared تنظیم شده: 0.888
* # آمار F: 427 در 3 و 158 DF، p-value: <2e-16

خروجی‌های خلاصه 6 مؤلفه را نشان می‌دهد که شامل:

* **فراخوانی**. فراخوانی تابع مورد استفاده برای محاسبه مدل رگرسیون را نشان می‌دهد.
* **باقیمانده‌ها** یک نمای سریع از توزیع باقیمانده‌ها که طبق تعریف دارای میانگین صفر هستند ارائه دهید. بنابراین، میانه نباید از صفر دور باشد و حداقل و حداکثر باید تقریباً در مقدار مطلق برابر باشند.
* **ضرایب** . ضرایب بتای رگرسیون و اهمیت آماری آنها را نشان می‌دهد. متغیرهای پیش‌بینی که به‌طور قابل‌توجهی با متغیر نتیجه مرتبط هستند، با ستاره مشخص می‌شوند.
* **خطای استاندارد باقیمانده (**RSE)، **R-squared (**R2) و **آماره F** معیارهایی هستند که برای بررسی میزان برازش مدل با داده‌های ما استفاده می‌شوند.

اولین گام در تفسیر تحلیل رگرسیون چندگانه، بررسی آماره F و مقدار p مربوط به آن در انتهای خلاصه مدل است.

در مثال ما، می‌توان مشاهده کرد که p-value از آماره F کمتر از 2.2e-16 است که بسیار معنی‌دار است. این بدان معنی است که حداقل یکی از متغیرهای پیش‌بینی کننده به‌طور قابل‌توجهی با متغیر نتیجه مرتبط است.

3.6.2 اهمیت ضرایب

برای اینکه ببینید کدام متغیرهای پیش‌بینی‌کننده مهم هستند، می‌توانید جدول ضرایب را بررسی کنید که تخمین ضرایب بتای رگرسیون و مقادیر p آماره t مرتبط را نشان می‌دهد.

**خلاصه (**مدل) **$** coef

* # Estimate Std. خطای t مقدار Pr(>|t|)
* # (Intercept) 3.39188 0.44062 7.698 1.41e-12
* # youtube 0.04557 0.00159 28.630 2.03e-64
* # facebook 0.18694 0.00989 18.905 2.07e-42
* # روزنامه 0.00179 0.00677 0.264 7.92e-01

برای یک پیش‌بینی‌کننده معین، آماره t ارزیابی می‌کند که آیا ارتباط معنی‌داری بین پیش‌بینی‌کننده و متغیر نتیجه وجود دارد یا خیر، یعنی اینکه آیا ضریب بتا پیش‌بینی‌کننده به‌طور معنی‌داری با صفر متفاوت است.

مشاهده می‌شود که تغییر در بودجه تبلیغات یوتیوب و فیس بوک به‌طور قابل‌توجهی با تغییرات در فروش مرتبط است در حالی که تغییرات در بودجه روزنامه به میزان قابل‌توجهی ­با فروش مرتبط نیست.

برای یک متغیر پیش‌بینی‌کننده معین، ضریب (b) را می‌توان به‌عنوان اثر متوسط روی y یک واحد افزایش در پیش‌بینی‌کننده تفسیر کرد که همه پیش‌بینی‌کننده‌های دیگر را ثابت نگه می‌دارد.

به‌عنوان مثال، برای مقدار ثابتی از بودجه تبلیغاتی یوتیوب و روزنامه، صرف 1000 دلار اضافی برای تبلیغات فیس بوک منجر به افزایش فروش تقریباً 0.1885\*1000 = 189 واحد فروش می‌شود.

ضریب یوتیوب نشان می‌دهد که به ازای هر 1000 دلار افزایش در بودجه تبلیغات یوتیوب، با ثابت نگه داشتن سایر پیش‌بینی‌کننده‌ها، می‌توان انتظار افزایش 0.045\*1000 = 45 واحد فروش را داشت.

ما دریافتیم که روزنامه در مدل رگرسیون چندگانه معنادار نیست. این بدان معناست که برای مقدار ثابتی از بودجه تبلیغاتی یوتیوب و فیس بوک، تغییرات در بودجه تبلیغات روزنامه تاثیر قابل‌توجهی بر واحدهای فروش نخواهد داشت.

از آنجایی که متغیر روزنامه معنی‌دار نیست، می‌توان آن را از مدل حذف کرد:

مدل <- **lm (**فروش **~** یوتیوب **+** فیس بوک، داده = train.data) **خلاصه (**مدل)

* #
* # زنگ زدن:
* # lm (فرمول = فروش ~ یوتیوب + فیس بوک، داده = train.data)
* #
* # باقیمانده ها:
* # حداقل IQ میانه 3Q Max
* # -10.481 -1.104 0.349 1.423 3.486

##

* # ضرایب:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ## | تخمین زدن | Std. | خطای تی | ارزش | Pr(>|t|) |
| ## (عرض از مبدأ) | 3.43446 | 0. | .40877 | 8.4 | 2.3e-14 \*\*\* |
| ## یوتیوب | 0.04558 | 0. | 00159 | 28.7 | < 2e-16 \*\*\* |
| ## فیس بوک | 0.18788 | 0. | 00920 | 20.4 | < 2e-16 \*\*\* |
| ##  ## Signif. کدها: 0 '\*\*\*' 0. | | | 001 '\*\*' | 1 0.01 | '\*' 0.05 '.' 0.1 1 1 1 |

##

## خطای استاندارد باقیمانده: 2.11 در 159 درجه آزادی

## مربع R چندگانه: 0.89، مربع R تنظیم شده: 0.889

## آماره F: 644 در 2 و 159 DF، p-value: <2e-16

در نهایت، معادله مدل ما را می‌توان به صورت زیر نوشت: فروش = 3.43+ 0.045\*youtube + 0.187\*facebook .

3.6.3 دقت مدل

هنگامی که تشخیص دادید که حداقل یک متغیر پیش‌بینی کننده به‌طور قابل‌توجهی با نتایج مرتبط ­است، باید تشخیص را با بررسی میزان برازش مدل با داده‌ها ادامه دهید. از این فرآیند به‌عنوان *خوب بودن برازش نیز یاد می‌شود*

کیفیت کلی برازش رگرسیون خطی را می‌توان با استفاده از سه کمیت زیر که در خلاصه مدل نمایش داده شده است ارزیابی کرد:

1. خطای استاندارد باقیمانده (RSE)،
2. مربع R (R2) و R2 تنظیم شده،
3. آماره F که قبلا در قسمت قبل توضیح داده شد

## rse r.squared f.statistic p.value

## 1 2.11 0.89 644 5.64e-77

1. خطای استاندارد باقیمانده (RSE).

RSE (یا مدل *سیگما* )، مربوط به خطای پیش‌بینی، تقریباً میانگین تفاوت بین مقادیر نتیجه مشاهده‌شده و مقادیر پیش‌بینی‌شده توسط مدل را نشان می‌دهد. هرچه RSE کمتر باشد، بهترین مدل با داده‌های ما مطابقت دارد.

تقسیم RSE بر مقدار متوسط متغیر نتیجه، نرخ خطای پیش‌بینی را به شما می‌دهد که باید تا حد امکان کوچک باشد.

در مثال ما، تنها با استفاده از متغیرهای پیش‌بینی یوتیوب و فیس بوک، RSE = 2.11، به این معنی که مقادیر فروش مشاهده شده به‌طور متوسط تقریباً 2.11 واحد از مقادیر پیش‌بینی شده انحراف دارند.

این مربوط به نرخ خطای 2.11/mean(train.data$sales) = 2.11/16.77 = 13% است که پایین است.

1. **R-squared و Adjusted R-squared** :

R-squared (R2) از 0 تا 1 متغیر است و نشان دهنده نسبت تغییرات در متغیر نتیجه که می‌تواند توسط متغیرهای پیش‌بینی مدل توضیح داده شود، است.

برای یک رگرسیون خطی ساده، R2 مربع ضریب همبستگی پیرسون بین متغیرهای نتیجه و پیش‌بینی است. در رگرسیون خطی چندگانه، R2 نشان دهنده ضریب همبستگی بین مقادیر نتیجه مشاهده شده و مقادیر پیش‌بینی شده است.

R2 اندازه‌گیری می‌کند که مدل چقدر با داده‌ها مطابقت دارد. هرچه R2 بالاتر باشد، مدل بهتر است. با این حال، یک مشکل با R2، این است که، زمانی که متغیرهای بیشتری به مدل اضافه می‌شوند، همیشه افزایش می‌یابد، حتی اگر آن متغیرها فقط با نتیجه ارتباط ضعیفی داشته باشند (جیمز و همکاران، 2014). یک راه حل این است که R2 را با در نظر گرفتن تعداد متغیرهای پیش‌بینی تنظیم کنید.

تعدیل در مقدار "مربع R تنظیم شده" در خروجی خلاصه اصلاحی برای تعداد متغیرهای x موجود در مدل پیش‌بینی است.

1. شما باید به‌طور عمده R-squared تعدیل شده را در نظر بگیرید که یک R2 جریمه شده برای تعداد بیشتر پیش‌بینی کننده است.

* یک R2 (تعدیل شده) که نزدیک به 1 است نشان می‌دهد که بخش زیادی از تغییرپذیری در نتیجه توسط مدل رگرسیون توضیح داده شده است.
* عدد نزدیک به 0 نشان می‌دهد که مدل رگرسیون بسیاری از تغییرات در نتیجه را توضیح نمی‌دهد.

در مثال ما، R2 تنظیم شده 0.88 است که خوب است.

1. F-آمار :

به یاد بیاورید که آماره F اهمیت کلی مدل را نشان می‌دهد. ارزیابی می‌کند که آیا حداقل یک متغیر پیش‌بینی دارای ضریب غیر صفر است یا خیر.

در یک رگرسیون خطی ساده، این آزمون واقعاً جالب نیست زیرا فقط اطلاعات ­داده شده توسط آزمون t موجود در جدول ضرایب را کپی می‌کند.

هنگامی که ما شروع به استفاده از پیش‌بینی کننده‌های متعدد مانند رگرسیون خطی چندگانه می‌کنیم، آماره F اهمیت بیشتری پیدا می‌کند.

یک عدد آماره F بزرگ با یک مقدار p معنی‌دار آماری مطابقت دارد (05/0p<). در مثال ما، آماره F برابر با 644 است که مقدار p برابر 1.46e-42 را تولید می‌کند که بسیار معنی‌دار است.

3.7 انجام پیش‌بینی

برای ارزیابی عملکرد مدل رگرسیونی خود، با استفاده از داده‌های آزمون، پیش‌بینی‌هایی انجام خواهیم داد.

روال به شرح زیر است:

1. مقادیر فروش را بر اساس بودجه‌های تبلیغاتی جدید در داده‌های آزمایشی پیش‌بینی کنید
2. ارزیابی عملکرد مدل با محاسبه:

* خطای پیش‌بینی RMSE (ریشه میانگین مربعات خطا)، نشان‌دهنده اختلاف میانگین بین مقادیر نتیجه شناخته شده مشاهده‌شده در داده‌های آزمون و مقادیر نتیجه پیش‌بینی‌شده توسط مدل است. هرچه RMSE کمتر باشد، مدل بهتر است.
* R-square (R2) که نشان دهنده همبستگی بین مقادیر نتیجه مشاهده شده و مقادیر نتیجه پیش‌بینی شده است. هرچه R2 بالاتر باشد، مدل بهتر است.
* *بسازید، پیش‌بینی کنید*

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**test.data)

* *عملکرد مدل*
* *(الف) خطای پیش‌بینی، RMSE را محاسبه کنید*

**RMSE (**پیش‌بینی‌ها، test.data **$** فروش)

* # [1] 1.58
* *(ب) R-square را محاسبه کنید*

**R2 (**پیش‌بینی‌ها، test.data **$** فروش)

* # [1] 0.938

از خروجی بالا، R2 0.93 است، به این معنی که مقادیر نتیجه مشاهده شده و پیش‌بینی شده همبستگی بالایی دارند که بسیار خوب است.

خطای پیش‌بینی RMSE 1.58 است که نشان‌دهنده نرخ خطای 1.58/mean(test.data$sales) = 1.58/17 = 9.2٪ است که خوب است.

3.8 بحث

این فصل اصول رگرسیون خطی را تشریح می‌کند و مثال‌های عملی در R برای محاسبه مدل‌های رگرسیون خطی ساده و چندگانه ارائه می‌کند. ما همچنین نحوه ارزیابی عملکرد مدل را برای پیش‌بینی‌ها شرح دادیم.

توجه داشته باشید که رگرسیون خطی یک رابطه خطی بین نتیجه و متغیرهای پیش‌بینی را فرض می‌کند. این را می‌توان به راحتی با ایجاد نمودار پراکندگی متغیر نتیجه در مقابل متغیر پیش‌بینی بررسی کرد.

به‌عنوان مثال، کد R زیر واحدهای فروش را در مقابل بودجه تبلیغاتی یوتیوب نشان می‌دهد. ما همچنین یک خط صاف اضافه می‌کنیم:

**ggplot (**بازاریابی، **aes (**x = یوتیوب، y = فروش)) **+**

**geom\_point (**) **+**

**stat\_smooth (**)

نمودار بالا یک رابطه خطی رو به افزایش بین فروش و یوتیوب را نشان می‌دهد

متغیرها که چیز خوبی است.

علاوه بر مفروضات خطی بودن، روش رگرسیون خطی ­فرضیات دیگری را در مورد داده‌های شما ایجاد می‌کند (به فصل 8 مراجعه کنید). باید مطمئن شوید که این مفروضات برای داده‌های شما صادق است.

مسائل بالقوه عبارتند از: الف) وجود مشاهدات تأثیرگذار در داده‌ها (فصل 8)، غیر خطی بودن بین نتیجه و برخی متغیرهای پیش‌بینی کننده (6) و وجود همبستگی قوی بین متغیرهای پیش‌بینی کننده (فصل 9).

فصل 4

اثرات متقابل در رگرسیون چندگانه

* 1. مقدمه

این فصل نحوه محاسبه رگرسیون خطی چندگانه با **اثرات متقابل را شرح می‌دهد** .

قبلاً، نحوه ساخت یک مدل رگرسیون خطی چندگانه (فصل 3) برای پیش‌بینی یک متغیر پیامد پیوسته (y) بر اساس متغیرهای پیش‌بینی‌کننده چندگانه (x) را شرح داده‌ایم.

به‌عنوان مثال، برای پیش‌بینی فروش، بر اساس بودجه‌های تبلیغاتی صرف شده در یوتیوب و فیس‌بوک، معادله مدل فروش = bO + bl\*youtube + b2\*facebook است که در آن، bO عرض از مبدأ است. bl و b2 ضرایب رگرسیونی هستند که به ترتیب با متغیرهای پیش‌بینی کننده youtube و facebook مرتبط هستند.

معادله فوق که به‌عنوان *مدل افزایشی نیز شناخته می‌شود،* تنها اثرات اصلی پیش‌بینی‌کننده‌ها را بررسی ­می‌کند. فرض می‌کند که رابطه بین یک متغیر پیش‌بینی‌کننده معین و نتیجه مستقل از سایر متغیرهای پیش‌بینی‌کننده است (جیمز و همکاران، 2014، بروس و بروس (2017)).

با در نظر گرفتن مثال ما، مدل افزودنی فرض می‌کند که تأثیر بر فروش با تبلیغات یوتیوب ­مستقل از تأثیر تبلیغات فیس بوک است.

این فرض ممکن است درست نباشد. به‌عنوان مثال، صرف هزینه برای تبلیغات فیس بوک ممکن است اثربخشی تبلیغات یوتیوب بر فروش را افزایش دهد. در بازاریابی، این به‌عنوان اثر هم افزایی شناخته می‌شود و در آمار به‌عنوان اثر متقابل نامیده می‌شود (جیمز و همکاران، 2014).

در این فصل یاد خواهید گرفت:

* معادله رگرسیون خطی چندگانه با اندرکنش
* کدهای R برای محاسبه ضرایب رگرسیون مرتبط با اثرات اصلی و اثرات متقابل
* چگونه اثر تعامل را تفسیر کنیم
  1. معادله

معادله رگرسیون خطی چندگانه، با اثرات متقابل بین دو پیش‌بینی کننده (xl و x2)، می‌تواند به صورت زیر نوشته شود:

y = bO + bl\*xl + b2\*x2 + b3\*(xl\*x2)

با توجه به مثال ما، تبدیل می‌شود:

فروش = bO + bl\*youtube + b2\*facebook + b3\*(youtube\*facebook)

این را می‌توان به صورت زیر نیز نوشت:

فروش = bO + (bl + b3 \* فیس بوک) \* یوتیوب + b2 \* فیس بوک

یا به صورت:

فروش = bO + bl\*youtube + (b2 +b3\*youtube)\*facebook

b3 را می‌توان به‌عنوان افزایش اثربخشی تبلیغات یوتیوب برای افزایش یک واحد در تبلیغات فیس بوک (یا برعکس) تفسیر کرد.

در بخش‌های بعدی نحوه محاسبه ضرایب رگرسیون در R را یاد خواهید گرفت.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید

**کتابخانه (**Tidyverse)

**کتابخانه (**کارت)

* 1. آماده‌سازی داده ها

ما از مجموعه داده‌های بازاریابی، معرفی شده در فصل 2، برای پیش‌بینی واحدهای فروش بر اساس میزان پول صرف شده در سه رسانه تبلیغاتی (یوتیوب، فیس بوک و روزنامه) استفاده خواهیم کرد.

ما داده‌ها را به صورت تصادفی به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم می‌کنیم.

* *داده‌ها را بارگذاری کنید*

**داده (**"بازاریابی"، بسته = "datarium")

* *داده‌ها را بررسی کنید*

**sample\_n (**بازاریابی، 3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ## | یوتیوب | فیس بوک | روزنامه | حراجی |
| ## 158 | 18O | 1.56 | 29.2 | 12.1 |
| ## 82 | 288 | 4.92 | 44.3 | 14.8 |
| ## 175 | 267 | 4.O8 | 15.7 | 13.8 |

* *تقسیم داده‌ها به مجموعه آموزشی و تست* **set.seed (**l23)

training.samples <- marketing$sales °/ o >°/o

**createDataPartition (**p = O.8، list = FALSE) train.data <- marketingftraining.samples، ] test.data <- marketingf **-** training.samples، ]

* 1. محاسبات
     1. مدل افزودنی

مدل رگرسیون خطی استاندارد را می‌توان به صورت زیر محاسبه کرد:

* *مدل را بسازید*

modell <- **lm (**فروش **~** یوتیوب **+** فیس بوک، داده = train.data) *# خلاصه مدل را* **خلاصه کنید (**modell)

##

## زنگ زدن:

* # lm (فرمول = فروش - یوتیوب + فیس بوک، داده = train.data)
* #
* # باقیمانده ها:
* # حداقل lQ میانه 3Q حداکثر
* # -10.481 -1.104 0.349 1.423 3.486
* #
* # ضرایب:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ## | تخمین زدن | Std. | خطای تی | ارزش | Pr(>|t|) |
| ## (عرض از مبدأ) | 3.43446 | 0. | .40877 | 8.4 | 2.3e-14 \*\*\* |
| ## یوتیوب | 0.04558 | 0. | 00159 | 28.7 | < 2e-16 \*\*\* |
| ## فیس بوک | 0.18788 | 0. | 00920 | 20.4 | < 2e-16 \*\*\* |
| ##  ## Signif. کدها: 0 '\*\*\*' 0. | | | 001 '\*\*' | 1 0.01 | '\*' 0.05 '.' 0.1 1 1 1 |

* #
* # خطای استاندارد باقیمانده: 2.11 در 159 درجه آزادی
* # R-squared چندگانه: 0.89، R-squared تنظیم شده: 0.889
* # آمار F: 644 در 2 و 159 DF، p-value: <2e-16
* *پیش‌بینی کنید*

پیش‌بینی‌ها <- model1 °/ o >°/o **پیش‌بینی (**test.data)

* *عملکرد مدل*
* *(الف) خطای پیش‌بینی، RMSE*

**RMSE (**پیش‌بینی‌ها، test.data **$** فروش)

* # [1] 1.58

*# (ب) R-square*

**R2 (**پیش‌بینی‌ها، test.data **$** فروش) ## [1] 0.938

* + 1. اثرات متقابل

در R، شما تعاملات بین متغیرها را با استفاده از عملگر \* وارد می‌کنید:

* *مدل را بسازید*
* *از این استفاده کن:*

model2 <- **Im (**فروش **~** یوتیوب **+** فیس بوک **+** یوتیوب **:** فیس بوک، داده = بازاریابی)

* *یا به سادگی، از این استفاده کنید:*

model2 <- **lm (**فروش **~** یوتیوب **\*** فیس بوک، داده = train.data)

* *مدل را خلاصه کنید*

**خلاصه (**مدل 2)

|  |
| --- |
| ## |
| ## زنگ زدن: |
| ## lm (فرمول = فروش - یوتیوب \* فیس بوک، داده = train.data) |
| ## |
| ## باقیمانده ها: |
| ## حداقل 1Q میانه 3Q حداکثر |
| ## -7.438 -0.482 0.231 0.748 1.860 |
| ## |
| ## ضرایب: |
| ## برآورد Std. خطای t مقدار Pr(>|t|) |
| ## (Intercept) 7.90e+00 3.28e-01 24.06 <2e-16 \*\*\* |
| ## youtube 1.95e-02 1.64e-03 11.90 <2e-16 \*\*\* |
| ## facebook 2.96e-02 9.83e-03 3.01 0.003 \*\* |
| ## youtube:facebook 9.12e-04 4.84e-05 18.86 <2e-16 \*\*\* |
| ## |
| ## Signif. کدها: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 1 1 1 |
| ## |

* # خطای استاندارد باقیمانده: 1.18 در 158 درجه آزادی ## چندگانه R-squared: 0.966، Adjusted R-squared: 0.966 ## F-statistic: 1.51e+03 در 3 و 158 DF، p-value: <2e-16
* *پیش‌بینی کنید*

پیش‌بینی‌ها <- model2 °/ o >°/o **پیش‌بینی (**test.data)

* *عملکرد مدل*
* *(الف) خطای پیش‌بینی، RMSE*

**RMSE (**پیش‌بینی‌ها، test.data **$** فروش)

* # [1] 0.963
* *(ب) R-square*

**R2 (**پیش‌بینی‌ها، test.data **$** فروش)

* # [1] 0.982
  1. تفسیر

مشاهده می‌شود که همه ضرایب، از جمله ضریب عبارت تعامل، از نظر آماری ­معنی‌دار هستند که نشان می‌دهد بین دو متغیر پیش‌بینی کننده (تبلیغات یوتیوب و فیس بوک) رابطه متقابل وجود دارد.

معادله مدل ما به شکل زیر است:

فروش = 7.89 + 0.019\*youtube + 0.029\*facebook + 0.0009\*youtube\*facebook

می‌توانیم این را اینگونه تعبیر کنیم که افزایش 1000 دلاری در تبلیغات یوتیوب با افزایش فروش (b1 + b3\*facebook)\*1000 = 19 + 0.9 \* فیس‌بوک همراه است . و افزایش 1000 دلاری در تبلیغات فیس بوک با افزایش فروش (b2 + b3\*youtube)\*1000 = 28 + 0.9 \* واحدهای یوتیوب همراه خواهد بود .

توجه داشته باشید که گاهی اوقات، این مورد است که عبارت تعامل مهم است اما اثرات اصلی نیست. اصل سلسله مراتبی بیان می‌کند که، اگر یک تعامل را در یک مدل قرار دهیم، باید اثرات اصلی را نیز لحاظ کنیم، حتی اگر مقادیر p مرتبط با ضرایب آنها معنی‌دار نباشد [@james2014].

* 1. مقایسه مدل‌های افزودنی و برهمکنش

خطای پیش‌بینی RMSE مدل برهمکنش 0.963 است که کمتر از خطای پیش‌بینی مدل افزایشی (1.58) است.

علاوه بر این، مقدار R-square (R2) مدل برهمکنش 98٪ در مقایسه با تنها 93٪ برای مدل افزایشی است.

این نتایج نشان می‌دهد که مدل با عبارت تعامل بهتر از مدلی است که فقط دارای اثرات اصلی است. بنابراین، برای این داده خاص، باید به سراغ مدل با مدل تعامل برویم.

* 1. بحث

این فصل نحوه محاسبه رگرسیون خطی چندگانه با اثرات متقابل را شرح می‌دهد. در ­شرایط تراکشن اگر به‌طور قابل‌توجهی باشد باید در مدل گنجانده شود.

فصل 5

رگرسیون با متغیرهای طبقه‌بندی

* 1. مقدمه

این فصل نحوه محاسبه **رگرسیون را با متغیرهای طبقه‌بندی توضیح می‌**دهد.

**متغیرهای طبقه‌ای (**همچنین به‌عنوان متغیرهای *عاملی* یا *کیفی* شناخته می‌شوند) متغیرهایی هستند که مشاهدات را در گروه‌هایی طبقه‌بندی می‌کنند. آنها دارای تعداد محدودی از مقادیر مختلف هستند که سطوح نامیده می‌شوند. به‌عنوان مثال جنسیت افراد یک متغیر طبقه‌بندی است که می‌تواند دو سطح داشته باشد: مذکر یا زن.

تحلیل رگرسیون به متغیرهای عددی نیاز دارد. بنابراین، زمانی که محققی بخواهد یک متغیر طبقه‌بندی را در یک مدل رگرسیونی قرار دهد، مراحل تکمیلی لازم است تا نتایج قابل تفسیر باشد.

در این مراحل، متغیرهای طبقه‌بندی به مجموعه‌ای از متغیرهای باینری جداگانه کدگذاری می‌شوند. این کدگذاری مجدد "کدگذاری ساختگی" نامیده می‌شود و منجر به ایجاد جدولی به نام *ماتریس کنتراست می‌شود* . این کار به صورت خودکار توسط نرم‌افزارهای آماری مانند R انجام می‌شود.

در اینجا، نحوه ساخت و تفسیر یک مدل رگرسیون خطی با متغیرهای پیش‌بینی کننده طبقه‌ای را خواهید آموخت. ما همچنین مثال‌های عملی را در R ارائه خواهیم داد.

* 1. بارگیری بسته‌های R مورد نیاز

• مرتب برای دستکاری و تجسم آسان داده ها

**کتابخانه (**tidyverse)

* 1. نمونه‌ای از مجموعه داده ها

ما از مجموعه داده حقوق و دستمزد [ بسته خودرو ] استفاده خواهیم کرد که شامل حقوق نه ماهه تحصیلی 2008-09 برای استادیاران، دانشیاران و اساتید در کالج در ایالات متحده است.

داده‌ها به‌عنوان بخشی از تلاش‌های مداوم مدیریت کالج برای نظارت بر تفاوت حقوق بین اعضای هیئت علمی مرد و زن جمع‌آوری شد.

* *داده‌ها را بارگذاری کنید*

**داده‌ها (**"حقوق"، بسته = "ماشین")

* *داده‌ها را بررسی کنید*

**sample\_n (**حقوق، 3)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ## | | رتبه | انضباط yrs.since.phd سالهای خدمت | | | رابطه ی جنسی | حقوق |
| ## | 3l3 | پروفسور | آ | 29 | l9 | نر | 9435O |
| ## | l62 | پروفسور | ب | 26 | l9 | نر | l765OO |
| ## | 349 | استادیار | ب | 4 | 3 | نر | 8Ol39 |

5.4 متغیرهای طبقه‌بندی شده با دو سطح

به یاد بیاورید که معادله رگرسیون برای پیش‌بینی متغیر نتیجه (y) بر اساس متغیر پیش‌بینی‌کننده (x)، می‌تواند به سادگی به صورت y = bO + bl\*x نوشته شود. bO و 'bl ضرایب بتای رگرسیونی هستند که به ترتیب نشان‌دهنده فاصله و شیب هستند.

فرض کنید که ما می‌خواهیم تفاوت حقوق بین زن و مرد را بررسی کنیم.

بر اساس متغیر جنسیت، می‌توانیم یک متغیر ساختگی جدید ایجاد کنیم که مقدار:

* مقدار 1 برای آقا
* مقدار 0 برای خانم

و از این متغیر به‌عنوان پیش‌بینی کننده در معادله رگرسیون استفاده کنید که منجر به مدل زیر می‌شود:

* bO + bl اگر فرد مذکر باشد
* bo اگر فرد زن باشد

ضرایب را می‌توان به صورت زیر تفسیر کرد:

1. bO میانگین حقوق در میان زنان است،
2. bO + bl میانگین حقوق در بین مردان است،
3. و bl میانگین تفاوت حقوق بین زن و مرد است.

برای نشان دادن ساده، مثال زیر تفاوت حقوق بین زن و مرد را با محاسبه یک مدل رگرسیون خطی ساده بر روی مجموعه داده حقوق [ بسته خودرو ] مدل می‌کند. R به‌طور خودکار متغیرهای ساختگی ایجاد می‌کند:

* *مدل را محاسبه کنید*

مدل <- **Im (**حقوق **~** جنسیت، داده = حقوق) **خلاصه (**مدل) **$** coef

* # Estimate Std. خطای t مقدار Pr(>|t|)
* # (عرض از مبدأ) lOlOO2 48O9 2l.OO 2.68e-66
* # sexMale l4O88 5O65 2.78 5.67e-O3

101002 + 14088 = 115090 تخمین زده می‌شود . تفاوت میانگین حقوق بین جنسیت ها

تابع ()contrast کدگذاری را که R برای ایجاد متغیرهای ساختگی استفاده کرده است برمی گرداند:

**تضادها (**حقوق **$** جنس)

* # نر
* # زن 0
* # مرد 1

R یک متغیر ساختگی sexMale ایجاد کرده است که اگر جنسیت Male باشد 1 و در غیر این صورت 0 می‌گیرد. تصمیم برای کد کردن مردان به‌عنوان 1 و زنان به‌عنوان 0 (خط پایه) دلخواه است و تأثیری بر محاسبه رگرسیون ندارد، اما تفسیر ضرایب را تغییر می‌دهد.

می‌توانید از تابع relevelO برای تنظیم دسته پایه برای مردان به صورت زیر استفاده کنید:

حقوق <- حقوق °/ o >°/o

**جهش** fsex = **سطح** fsex، ref = "مذکر"))

خروجی برازش رگرسیون می‌شود:

مدل <- **Im (**حقوق **~** جنسیت، داده = حقوق)

**خلاصه (**مدل) **$** coef

* # Estimate Std. خطای t مقدار Pr(>|t|)
* # (عرض از مبدأ) 115090 1587 72.50 2.46e-230
* # sexFemale -14088 5065 -2.78 5.67e-03

ضریب جنس زن در خروجی رگرسیون نشان می‌دهد که زن بودن با کاهش حقوق (نسبت به مردان) همراه است.

اکنون برآوردهای bo و b1 به ترتیب 115090 و -14088 است که بار دیگر منجر به پیش‌بینی میانگین حقوق 115090 برای مردان و پیش‌بینی 115090 - 14088 = 101002 برای زنان می‌شود.

روش دیگر، به جای یک طرح کدگذاری 0/1، می‌توانیم یک متغیر ساختگی -1 (مرد) / 1 (مونث) ایجاد کنیم. این منجر به مدل می‌شود:

* b0 - b1 اگر فرد مرد باشد
* b0 + b1 اگر فرد مؤنث باشد

بنابراین، اگر متغیر طبقه‌ای به صورت ۱- و ۱ کد شود، اگر ضریب رگرسیون مثبت باشد، از گروهی که با کد ۱- کد شده است، به گروهی که با کد ۱- کد شده است، اضافه می‌شود. اگر ضریب رگرسیون منفی باشد، جمع می‌شود. و تفریق برعکس می‌شود.

5.5 متغیرهای طبقه‌بندی با بیش از دو سطح

به‌طور کلی، یک متغیر طبقه‌بندی با n سطح به n-1 متغیر هر کدام با دو سطح تبدیل می‌شود. این n-1 متغیر جدید حاوی اطلاعات یکسانی نسبت به متغیر تک هستند. این کدگذاری مجدد جدولی به نام **ماتریس کنتراست ایجاد می‌کند** .

به‌عنوان مثال رتبه در داده‌های حقوق دارای سه سطح است: "AsstProf"، "AssocProf" و "Prof". این متغیر می‌تواند به صورت ساختگی به دو متغیر، یکی AssocProf و دیگری Profes کدگذاری شود:

* اگر رتبه = AssocProf باشد، ستون AssocProf با 1 و Prof با 0 کدگذاری می‌شود.
* اگر رتبه = پروفسور باشد، ستون AssocProf با 0 و Prof با 1 کدگذاری می‌شود.
* اگر رتبه = AsstProf باشد، هر دو ستون "AssocProf" و "Prof" با 0 کد گذاری می‌شوند.

این کدگذاری ساختگی به‌طور خودکار توسط R انجام می‌شود. برای هدف نمایش، می‌توانید از تابع model.matrixf برای ایجاد یک ماتریس کنتراست برای یک متغیر عامل استفاده کنید: res <- **model.matrix (** **~** رتبه، داده = حقوق) **head (**res[، **-** 1])

|  |  |  |  |
| --- | --- | --- | --- |
| ## |  | رتبه AssocProf | رتبه پروفسور |
| ## | 1 | 0 | 1 |
| ## | 2 | 0 | 1 |
| ## | 3 | 0 | 0 |
| ## | 4 | 0 | 1 |
| ## | 5 | 0 | 1 |
| ## | 6 | 1 | 0 |

هنگام ساخت مدل خطی، روش‌های مختلفی برای رمزگذاری متغیرهای طبقه‌بندی شده وجود دارد که به سامانه‌های کدگذاری کنتراست معروف هستند. گزینه پیش فرض در R این است که از اولین سطح فاکتور به‌عنوان مرجع استفاده کنید و سطوح باقیمانده را نسبت به این سطح تفسیر کنید.

توجه داشته باشید که ANOVA (تحلیل واریانس) فقط یک مورد خاص از مدل خطی است که در آن ­عوامل پیش‌بینی متغیرهای طبقه‌بندی هستند. و از آنجایی که R این واقعیت را درک می‌کند که ANOVA و رگرسیون هر دو نمونه‌هایی از مدل‌های خطی هستند، به شما امکان می‌دهد جدول ANOVA کلاسیک را از ­مدل رگرسیون خود با استفاده از تابع R base anova() یا تابع AnovaO [در بسته خودرو ] استخراج کنید. ما به‌طور کلی عملکرد AnovaO را توصیه می‌کنیم زیرا به‌طور خودکار از طرح‌های نامتعادل مراقبت می‌کند.

نتایج پیش‌بینی حقوق با استفاده از روش رگرسیون چندگانه در زیر ارائه شده است.

**کتابخانه (**ماشین)

model2 <- **Im (**حقوق **~** سال خدمات **+** رتبه **+** رشته **+** جنس، داده = حقوق)

**Anova (**model2)

|  |  |  |  |
| --- | --- | --- | --- |
| ## | میز آنووا | (نوع II | آزمایشات) |
| ## |  |  |  |
| ## | پاسخ: حقوق و دستمزد | |  |
| ## |  | مجموع مربع | مقدار Df F Pr(>F) |
| ## | سال خدمات | 3.24e+08 | 1 0.63 0.43 |
| ## | رتبه | 1.03e+11 | 2 100.26 < 2e-16 \*\*\* |
| ## | انضباط | 1.74e+10 | 1 33.86 1.2e-08 \*\*\* |
| ## | رابطه ی جنسی | 7.77e+08 | 1 1.51 0.22 |
| ## | باقیمانده ها | 2.01e+11 | 391 |
| ## | - |  |  |
| ## | Signif. کدها: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 | | |

1 1

با در نظر گرفتن سایر متغیرها (سال خدمت، رتبه و رشته)، می‌توان دریافت که متغیر طبقه‌بندی جنسیت دیگر به‌طور قابل‌توجهی با تغییر در حقوق بین افراد مرتبط نیست. متغیرهای مهم رتبه و رشته هستند.

اگر می‌خواهید کنتراست متغیر طبقه‌ای را تفسیر کنید، این را تایپ کنید:

**خلاصه (**model2) ## ## تماس:

## lm(فرمول = حقوق - سال. خدمات + رتبه + نظم + جنس،

## داده = حقوق)

##

## باقیمانده ها:

*##* حداقل IQ میانه 3Q Max

## -64202 -14255 -1533 10571 99163

##

## ضرایب:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ## | تخمین زدن | Std. خطا | مقدار t | Pr(>|t|) |
| ## (عرض از مبدأ) | 73122.9 | 3245.3 | 22.53 | < 2e-16 \*\*\* |
| ## سال خدمات | -88.8 | 111.6 | -0.80 | 0.42696 |
| ## رتبه AssocProf | 14560.4 | 4098.3 | 3.55 | 0.00043 \*\*\* |
| ## رتبه پروفسور | 49159.6 | 3834.5 | 12.82 | < 2e-16 \*\*\* |
| ## رشته B | 13473.4 | 2315.5 | 5.82 | 1.2e-08 \*\*\* |
| ## جنسیت زن | -4771.2 | 3878.0 | -1.23 | 0.21931 |
| ## |  |  |  |  |
| ## Signif. کدها: | : 0 '\*\*\*' | 0.001 '\*\*' | 0.01 '\* | 1 0.05 '.' 0.1 1 1 1 |

##

* # خطای استاندارد باقیمانده: 22700 در 391 درجه آزادی
* # R-squared چندگانه: 0.448، R-squared تنظیم شده: 0.441
* # آمار F: 63.4 در 5 و 391 DF، p-value: <2e-16

به‌عنوان مثال مشاهده می‌شود که بودن از رشته B (بخش‌های کاربردی) به‌طور معنی‌داری با افزایش میانگین 13473.38 حقوق نسبت به رشته A (گروه‌های نظری) همراه است.

5.6 بحث

در این فصل توضیح دادیم که چگونه متغیرهای طبقه‌بندی در مدل رگرسیون خطی گنجانده شده‌اند. از آنجایی که رگرسیون به ورودی‌های عددی نیاز دارد، متغیرهای طبقه‌ای باید در مجموعه‌ای از متغیرهای باینری کدگذاری شوند.

ما مثال‌های عملی برای موقعیت‌هایی ارائه می‌کنیم که در آن شما متغیرهای طبقه‌بندی حاوی دو یا چند سطح دارید.

توجه داشته باشید که برای متغیرهای دسته‌بندی با تعداد سطوح زیاد، ممکن است گروه‌بندی برخی از سطوح با هم مفید باشد.

برخی از متغیرهای طبقه‌بندی دارای سطوحی هستند که مرتب شده اند. آنها را می‌توان به مقادیر عددی تبدیل کرد و همانطور که هست استفاده کرد. به‌عنوان مثال، اگر نمرات استاد ("AsstProf"، "AssocProf" و "Prof") معنای خاصی دارند، می‌توانید آنها را به مقادیر عددی، مرتب شده از پایین به بالا، مطابق با اساتید درجه بالاتر تبدیل کنید.

فصل 6

رگرسیون غیر خطی

* 1. مقدمه

در برخی موارد، رابطه واقعی بین نتیجه و یک متغیر پیش‌بینی‌کننده ممکن است خطی نباشد.

راه حل‌های مختلفی برای گسترش مدل رگرسیون خطی (فصل 3) برای گرفتن این اثرات غیرخطی وجود دارد، از جمله:

* **رگرسیون چند جمله‌ای (Polynomial regression)** این رویکرد ساده برای مدل‌سازی روابط غیر خطی است. عبارات چند جمله‌ای یا عبارات درجه دوم (مربع، مکعب و غیره) را به یک رگرسیون اضافه می‌کند.
* **رگرسیون اسپلاین** . با یک منحنی صاف با یک سری قطعات چند جمله‌ای مطابقت دارد. مقادیری که بخش‌های spline را محدود می‌کنند، **گره (Knots) نامیده می‌شوند** .
* **مدل‌های افزایشی تعمیم یافته (**GAM). متناسب با مدل‌های اسپلاین با انتخاب خودکار گره ها.

در این فصل، نحوه محاسبه مدل‌های رگرسیون غیرخطی و نحوه مقایسه مدل‌های مختلف به منظور انتخاب بهترین مدل‌ها را خواهید آموخت.

از معیارهای RMSE و R2 برای مقایسه مدل‌های مختلف استفاده می‌شود (به فصل @ref (رگرسیون خطی) مراجعه کنید).

به یاد بیاورید که RMSE خطای پیش‌بینی مدل را نشان می‌دهد که میانگین تفاوت مقادیر نتیجه مشاهده‌شده و مقادیر نتیجه پیش‌بینی‌شده است. R2 نشان دهنده همبستگی مجذور بین مقادیر نتیجه مشاهده شده و پیش‌بینی شده است. بهترین مدل مدلی است که کمترین RMSE و بالاترین R2 را داشته باشد.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید

**کتابخانه (**Tidyverse)

**کتابخانه (**کارت)

**مجموعه\_تم (** **تم\_کلاسیک (**))

* 1. آماده‌سازی داده ها

ما از مجموعه داده بوستون [در بسته MASS ]، معرفی شده در فصل 2، برای پیش‌بینی ­مقدار خانه متوسط ( mdev )، در حومه بوستون، بر اساس متغیر پیش‌بینی‌کننده Istat (درصد وضعیت پایین‌تر جمعیت) استفاده خواهیم کرد.

ما داده‌ها را به صورت تصادفی به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم می‌کنیم. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *داده‌ها را بارگذاری کنید*

**داده (**"بوستون"، بسته = "MASS")

* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**123)

training.samples <- Boston$medv °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE)

train.data <- Bostonftraining.samples، ]

test.data <- Bostonf **-** training.samples, ]

ابتدا نمودار پراکندگی متغیرهای medv در مقابل Istat را به صورت زیر تجسم کنید:

**ggplot (**train.data، **aes (**Istat، medv) ) **+**

**geom\_point (**) **+**

**stat\_smooth (**)

نمودار پراکندگی بالا یک رابطه غیر خطی بین دو متغیر را نشان می‌دهد

در بخش‌های بعدی، با محاسبه مدل‌های رگرسیون خطی و غیر خطی شروع می‌کنیم. در مرحله بعد، مدل‌های مختلف را با هم مقایسه می‌کنیم تا بهترین را برای داده‌های خود انتخاب کنیم.

6.4 رگرسیون خطی {linear-reg}

معادله مدل رگرسیون خطی استاندارد را می‌توان به صورت medv = b0 + b1\*lstat نوشت . مدل رگرسیون خطی را محاسبه کنید: *# مدل* مدل را بسازید <- **lm (**medv **~** lstat, data = train.data)

* *بسازید، پیش‌بینی کنید*

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**test.data)

* *عملکرد مدل*

**data.frame (**

RMSE = **RMSE (**پیش‌بینی‌ها، test.data **$** medv)، R2 = **R2 (**پیش‌بینی‌ها، test.data **$** medv)

)

* # RMSE R2
* # 1 6.07 0.535

تجسم داده ها:

**ggplot (**train.data، **aes (**lstat، medv) ) **+ geom\_point (**) **+**

**stat\_smooth (**روش = Im، فرمول = y **~** x)

6.5 رگرسیون چند جمله ای

رگرسیون چند جمله‌ای عبارت‌های چند جمله‌ای یا درجه دوم را به صورت زیر به معادله رگرسیون اضافه می‌کند:

*medv = b* 0 + *b* 1 \* *Istat* + *b* 2 \* *Istat* 2

در R، برای ایجاد یک پیش‌بینی x""2 باید از تابع I() به صورت زیر استفاده کنید: I(x~2) . این x را به توان 2 افزایش می‌دهد.

رگرسیون چند جمله‌ای را می‌توان در R به صورت زیر محاسبه کرد:

**lm (**medv **~** Istat **+ I (**lstat **~** 2)، داده = train.data)

یک راه حل ساده جایگزین استفاده از این است:

**lm (**medv **~ poly (**Istat, 2)، data = train.data)

##

## زنگ زدن:

*##* lm(فرمول = medv ~ poly(lstat، 2)، داده = train.data)

##

## ضرایب:

* # (Intercept) poly(lstat, 2)1 poly(lstat, 2)2
* # 22.7 -139.4 57.7

خروجی شامل دو ضریب مرتبط با Istat است: یکی برای جمله خطی (Istat^l) و دیگری برای عبارت درجه دوم (lstat~2).

مثال زیر یک برازش چند جمله‌ای مرتبه ششم را محاسبه می‌کند:

**Im (**medv **~ poly (**lstat, 6), data = train.data) °/ o >°/o

**خلاصه (**) ##

* # زنگ زدن:
* # lm(فرمول = medv ~ poly(lstat، 6)، داده = train.data)
* #
* # باقیمانده ها:
* # حداقل 1Q میانه 3Q حداکثر

## -14.23 -3.24 -0.74 2.02 26.50

##

## ضرایب:

|  |  |  |  |
| --- | --- | --- | --- |
| ## | برآورد Std. | خطای t مقدار Pr | (>|t|) |
| ## (عرض از مبدأ) | 22.739 | 0.262 86.83 < | 2e-16 \*\*\* |
| ## poly(lstat, 6)1 | -139.357 | 5.284 -26.38 < | 2e-16 \*\*\* |
| ## poly(lstat, 6)2 | 57.728 | 5.284 10.93 < | 2e-16 \*\*\* |
| ## poly(lstat, 6)3 | -25.923 | 5.284 -4.91 1 | .4e-06 \*\*\* |
| ## poly(lstat، 6)4 | 21.378 | 5.284 4.05 6 | .3e-05 \*\*\* |
| ## poly(lstat، 6)5 | -13.817 | 5.284 -2.62 | 0.0093 \*\* |
| ## poly(lstat, 6)6 | 7.268 | 5.284 1.38 | 0.1697 |
| ## |  |  |  |
| ## Signif. کدها: | 0 '\*\*\*' 0.001 | '\*\*' 0.01 '\*' 0. | 05 '.' 0.1 1 1 1 |

##

* # خطای استاندارد باقیمانده: 5.28 در 400 درجه آزادی
* # مربع R چندگانه: 0.684، مربع R تنظیم شده: 0.679
* # آمار F: 144 در 6 و 400 DF، p-value: <2e-16

از خروجی بالا، می‌توان دریافت که عبارات چند جمله‌ای فراتر از مرتبه برازش معنی‌دار نیستند. بنابراین، فقط یک مدل رگرسیون چند جمله‌ای مناسب به صورت زیر ایجاد کنید:

* *مدل را بسازید*

مدل <- **lm (**medv **~ poly (**lstat, 5)، data = train.data)

* *پیش‌بینی کنید*

پیش‌بینی‌ها <- model **/>/ پیش‌بینی (**test.data)

* *عملکرد مدل*

**data.frame (**

RMSE = **RMSE (**پیش‌بینی‌ها، test.data **$** medv)،

R2 = **R2 (**پیش‌بینی‌ها، test.data **$** medv)

)

* # RMSE R2
* # 1 4.96 0.689

خط رگرسیون چند جمله‌ای مناسب را به صورت زیر تجسم کنید:

**ggplot (**train.data، **aes (**Istat، medv) ) **+**

**geom\_point (**) **+**

**stat\_smooth (**روش = Im، فرمول = y **~ پلی (**x، 5))

* 1. تبدیل لاگ

هنگامی که یک رابطه غیر خطی دارید، می‌توانید تبدیل لگاریتمی متغیرهای پیش‌بینی را نیز امتحان کنید:

* *مدل را بسازید*

مدل <- **lm (**medv **~ log (**lstat)، داده = train.data)

* *پیش‌بینی کنید*

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**test.data)

* *عملکرد مدل*

**data.frame (**

RMSE = **RMSE (**پیش‌بینی‌ها، test.data **$** medv)،

R2 = **R2 (**پیش‌بینی‌ها، test.data **$** medv)

)

* # RMSE R2
* # 1 5.24 0.657

تجسم داده ها:

**ggplot (**train.data، **aes (**Istat، medv) ) **+**

**geom\_point (**) **+**

**stat\_smooth (**روش = Im، فرمول = y **~ log (**x))

* 1. رگرسیون اسپلاین

رگرسیون چند جمله‌ای فقط مقدار مشخصی از انحنا را در یک رابطه غیرخطی ثبت می‌کند. یک رویکرد جایگزین و اغلب برتر برای مدل‌سازی روابط غیرخطی، استفاده از اسپلاین است (بروس و بروس، 2017).

Splines راهی برای درون یابی هموار بین نقاط ثابت فراهم می‌کند که گره نامیده می‌شود. رگرسیون چند جمله‌ای بین گره‌ها محاسبه می‌شود. به عبارت دیگر، اسپلاین‌ها مجموعه‌ای از بخش‌های چندجمله‌ای هستند که به هم متصل می‌شوند و در گره‌ها به هم می‌پیوندند (بروس و بروس، 2017).

خطوط بسته R شامل تابع bs برای ایجاد یک عبارت b-spline در یک مدل رگرسیونی است.

شما باید دو پارامتر را مشخص کنید: درجه چند جمله‌ای و محل گره ها. در مثال ما، گره‌ها را در چارک پایین، چارک میانی و چارک بالایی قرار می‌دهیم:

گره‌ها <- چندک (train.data **$ lstat،** p = **c (**0.25، 0.5، 0.75))

ما یک مدل با استفاده از یک spline مکعبی (درجه = 3) ایجاد می‌کنیم:

**کتابخانه (**اسپلین) *# گره‌های مدل را بسازید* <- **quantile (**train.data **$** lstat, p = **c (**0.25, 0.5, 0.75)) model <- **lm (**medv **~ bs (**lstat, knots = knots), data = train. داده) *# پیش‌بینی کنید*

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**test.data)

* *مدل عملکرد* **data.frame (**

RMSE = **RMSE (**پیش‌بینی‌ها، test.data **$** medv)، R2 = **R2 (**پیش‌بینی‌ها، test.data **$** medv))

* # RMSE R2
* # 1 4.97 0.688

توجه داشته باشید که ضرایب برای عبارت spline قابل تفسیر نیستند.

اسپلاین مکعبی را به صورت زیر تجسم کنید:

**ggplot (**train.data، **aes (**Istat، medv) ) **+**

**geom\_point (**) **+**

**stat\_smooth (**روش = Im، فرمول = y **~** splines **::bs (**x، df = 3))

* 1. مدل‌های افزودنی تعمیم یافته

هنگامی که یک رابطه غیر خطی را در داده‌های خود شناسایی کردید، ممکن است عبارات چند جمله‌ای به اندازه کافی انعطاف‌پذیر نباشند تا رابطه را به تصویر بکشند و عبارات spline نیاز به مشخص کردن گره‌ها دارند.

مدل‌های افزودنی تعمیم‌یافته یا GAM، تکنیکی برای برازش خودکار یک رگرسیون اسپلاین است. این را می‌توان با استفاده از بسته mgcv R انجام داد:

**کتابخانه (**mgcv) *# مدل* مدل را بسازید <- **gam (**medv **~ s (**lstat), data = train.data) *# پیش‌بینی کنید*

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**test.data)

* *مدل عملکرد* **data.frame (**

RMSE = **RMSE (**پیش‌بینی‌ها، test.data **$** medv)، R2 = **R2 (**پیش‌بینی‌ها، test.data **$** medv)

)

* # RMSE R2
* # 1 5.02 0.684

عبارت s(lstat) به تابع gam() می‌گوید بهترین گره‌ها را برای یک عبارت spline پیدا کند.

تجسم داده ها:

**ggplot (**train.data، **aes (**lstat، medv) ) **+ geom\_point (**) **+ stat\_smooth (**روش = گام، فرمول = y **~ s (**x))

* 1. مقایسه مدل ها

از تحلیل معیارهای RMSE و R2 مدل‌های مختلف، می‌توان مشاهده کرد که رگرسیون چند جمله‌ای، رگرسیون اسپلاین و مدل‌های افزایشی تعمیم‌یافته از مدل رگرسیون خطی و رویکردهای تبدیل log بهتر عمل می‌کنند.

* 1. بحث

این فصل نحوه محاسبه مدل‌های رگرسیون غیرخطی با استفاده از R را شرح می‌دهد.

**قسمت سوم**

**تشخیص رگرسیون**

فصل 7

مقدمه

پس از ساخت یک مدل رگرسیون خطی (فصل 3)، باید برخی از تشخیص‌ها را برای تشخیص مسائل احتمالی در داده‌ها انجام دهید.

در این قسمت یاد خواهید گرفت:

* مفروضات و تشخیص (diagnostics) رگرسیون خطی (فصل 8)
* مسائل احتمالی هنگام محاسبه مدل رگرسیون خطی، از جمله:
* رابطه غیر خطی بین نتیجه و پیش‌بینی کننده‌ها (فصل 6)
* چند خطی (فصل 9)
* متغیرهای مخدوش کننده (Confounding variables) (فصل 10)

فصل 8

مفروضات و تشخیص رگرسیون

* 1. مقدمه

رگرسیون خطی (فصل 3) چندین فرض را در مورد داده‌های در دست ایجاد می‌کند. این فصل **مفروضات رگرسیون را توصیف می‌**کند و نمودارهای داخلی برای **تشخیص رگرسیون** در زبان برنامه نویسی R ارائه می‌دهد.

پس از انجام تحلیل رگرسیون، همیشه باید بررسی کنید که آیا مدل برای داده‌های موجود خوب کار می‌کند یا خیر.

اولین گام از این تشخیص رگرسیون، بررسی اهمیت ضرایب بتای رگرسیون و همچنین R2 است که به ما می‌گوید مدل رگرسیون خطی چقدر با داده‌ها برازش دارد. این موضوع در فصل‌های 3 و 13 توضیح داده شده است.

در این فصل، مراحل بیشتری را برای ارزیابی میزان برازش مدل با داده‌ها خواهید آموخت.

به‌عنوان مثال، مدل رگرسیون خطی این فرض را ایجاد می‌کند که رابطه بین پیش‌بینی کننده‌ها (x) و متغیر نتیجه خطی است. این ممکن است درست نباشد. این رابطه می‌تواند چند جمله‌ای یا لگاریتمی باشد.

علاوه بر این، داده‌ها ممکن است حاوی برخی مشاهدات تأثیرگذار مانند مقادیر پرت (یا مقادیر شدید) باشند که می‌توانند بر نتیجه رگرسیون تأثیر بگذارند.

بنابراین، شما باید مدل رگرسیونی را که ساخته اید از نزدیک تشخیص دهید تا مسائل احتمالی را شناسایی کنید و بررسی کنید که آیا مفروضات مدل رگرسیون خطی برآورده شده است یا خیر.

برای انجام این کار، ما به‌طور کلی توزیع **خطاهای باقیمانده را بررسی می‌کنیم که** می‌تواند اطلاعات بیشتری در مورد داده‌های شما به شما بگوید.

در این فصل،

* ما با توضیح **خطاهای باقیمانده** و **مقادیر برازش شروع می‌کنیم** .
* **مفروضات رگرسیون** خطی و همچنین مسائل بالقوه‌ای را که می‌توانید هنگام انجام تحلیل رگرسیون با آن مواجه شوید، ارائه می‌کنیم.
* در نهایت، ما برخی از **نمودارهای تشخیصی داخلی را** در R برای آزمایش مفروضات زیربنایی مدل رگرسیون خطی توصیف می‌کنیم.
  1. بارگیری بسته‌های R مورد نیاز
* آیه مرتب برای دستکاری و تجسم آسان داده ها
* جارو : یک چارچوب داده مرتب از نتایج آزمون آماری ایجاد می‌کند

**کتابخانه (**Tidyverse)

**کتابخانه (**جارو)

**مجموعه\_تم (** **تم\_کلاسیک (**))

* 1. نمونه‌ای از داده ها

ما از بازاریابی مجموعه داده [بسته داده] که در فصل 2 معرفی شد استفاده خواهیم کرد. # داده‌های **داده** *را بارگذاری کنید (*"بازاریابی"، بسته = "datarium") **#** *نمونه داده را بازرسی کنید (*بازاریابی، 3)

## فروش روزنامه فیس بوک یوتیوب

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ## | 158 | 180 | 1. | 56 | 29. | .2 | 12. | . 1 |
| ## | 82 | 288 | 4. | 92 | 44. | 3 | 14. | .8 |
| ## | 175 | 267 | 4. | .08 | 15. | .7 | 13. | .8 |

* 1. ساخت مدل رگرسیون

ما مدلی برای پیش‌بینی فروش بر اساس بودجه تبلیغاتی صرف شده در رسانه‌های یوتیوب می‌سازیم.

مدل <- **Im (**فروش **~** یوتیوب، داده = بازاریابی) مدل ## ## تماس بگیرید:

* # lm (فرمول = فروش - یوتیوب، داده = بازاریابی)
* #
* # ضرایب:
* # (عرض از مبدأ) یوتیوب
* # 8.4391 0.0475

معادله رگرسیون ما این است: y = 8.43 + 0.07\*x، یعنی فروش = 8.43 + 0.047\*youtube .

قبل از توصیف مفروضات رگرسیون و تشخیص رگرسیون، با توضیح دو مفهوم کلیدی در تحلیل رگرسیون شروع می‌کنیم: مقادیر برازش و خطاهای باقیمانده. اینها برای درک نمودارهای تشخیصی ارائه شده در آینده مهم هستند.

* 1. مقادیر برازش شده و باقیمانده

مقادیر **برازش شده (** یا **پیش‌بینی‌شده** ) مقادیر y هستند که برای مقادیر x داده‌شده با توجه به مدل رگرسیون ساخته‌شده (یا از نظر بصری، بهترین برازش خط رگرسیون مستقیم) انتظار دارید.

در مثال ما، برای یک بودجه تبلیغاتی مشخص در یوتیوب، مقدار فروش برازش شده (پیش‌بینی‌شده) خواهد بود، فروش = 8.44 + 0.0048\*youtube .

از نمودار پراکندگی زیر، می‌توان دریافت که همه نقاط داده دقیقاً روی خط رگرسیون تخمینی قرار نمی‌گیرند. این بدان معناست که برای یک بودجه تبلیغاتی در یوتیوب، مقادیر فروش مشاهده شده (یا اندازه‌گیری شده) می‌تواند با مقادیر فروش پیش‌بینی‌شده متفاوت باشد. این تفاوت، **خطاهای باقیمانده نامیده می‌شود که** با خطوط قرمز عمودی نشان داده می‌شود.

در R، با استفاده از تابع augmentO [بسته جارو] می‌توانید به راحتی داده‌های خود را برای اضافه کردن مقادیر برازش شده و باقیمانده‌ها افزایش دهید. اجازه دهید خروجی را model.diag.metrics بنامیم زیرا شامل چندین معیار مفید برای تشخیص رگرسیون است. بعداً موضوع را شرح خواهیم داد.

model.diag.metrics <- **افزایش (**مدل)

**سر (**model.diag.metrics)

## فروش یوتیوب .fitted .se.fit .resid .hat .sigma .cooksd .std.resid

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ## | 1 | 26. | .52 | 276. | .1 | 21. | .56 | 0. | .385 | 4. | .955 | 0.00970 | 3. | 90 | 7. | .94e-03 | 1. | .2733 |
| ## | 2 | 12. | .48 | 53. | .4 | 10. | .98 | 0. | .431 | 1. | 502 | 0.01217 | 3. | .92 | 9. | .20e-04 | 0. | .3866 |
| ## | 3 | 11. | .16 | 20. | .6 | 9. | .42 | 0. | 502 | 1. | .740 | 0.01649 | 3. | .92 | 1. | 0.69e-03 | 0. | .4486 |
| ## | 4 | 22. | .20 | 181. | .8 | 17. | .08 | 0. | .277 | 5. | .119 | 0.00501 | 3. | 90 | 4. | 0.34e-03 | 1. | .3123 |
| ## | 5 | 15. | .48 | 217. | .0 | 18. | .75 | 0. | .297 | -3. | .273 | 0.00578 | 3. | .91 | 2. | 0.05e-03 | -0. | .8393 |
| ## | 6 | 8. | .64 | 10. | .4 | 8. | .94 | 0. | .525 | -0. | .295 | 0.01805 | 3. | .92 | 5. | 0.34e-05 | -0. | 0.0762 |

در میان ستون‌های جدول، موارد زیر وجود دارد:

* یوتیوب : بودجه تبلیغاتی یوتیوب سرمایه گذاری شده است
* فروش : مقادیر فروش مشاهده شده
* .fitted : مقادیر فروش برازش شده
* .resid : خطاهای باقیمانده
* ...

کد R زیر خطای باقیمانده (به رنگ قرمز) را بین مقادیر مشاهده شده و خط رگرسیون متناسب ترسیم می‌کند. هر بخش قرمز عمودی نشان دهنده خطای باقیمانده بین مقدار فروش مشاهده شده و مقدار پیش‌بینی شده (یعنی برازش) مربوطه است.

**ggplot (**model.diag.metrics، **aes (**youtube، sales)) **+**

**geom\_point (**) **+**

**stat\_smooth (**روش = Im، se = FALSE) **+**

**geom\_segment (** **aes (**xend = youtube، yend = .fitted)، رنگ = "قرمز"، اندازه = 0.3)

به منظور بررسی مفروضات رگرسیون، توزیع باقیمانده‌ها را بررسی می‌کنیم.

* 1. مفروضات رگرسیون

رگرسیون خطی چندین فرض را در مورد داده‌ها ایجاد می‌کند، مانند:

1. **خطی بودن داده‌ها** رابطه بین پیش‌بینی کننده (x) و نتیجه (y) خطی فرض می‌شود.
2. **نرمال بودن باقیمانده‌ها** فرض بر این است که خطاهای باقیمانده به‌طور نرمال توزیع شده اند.
3. **همگنی واریانس باقیمانده** . فرض می‌شود که باقیمانده‌ها دارای یک واریانس ثابت هستند ( **هموسکداستیسیته** )
4. استقلال عبارات خطای باقیمانده .

باید بررسی کنید که آیا این فرضیات درست هستند یا خیر. مسائل احتمالی عبارتند از:

1. **غیر خطی بودن** نتیجه - روابط پیش‌بینی کننده
2. **ناهمگونی** : واریانس غیر ثابت عبارات خطا.
3. **وجود مقادیر تأثیرگذار** در داده‌ها که می‌تواند:

* پرت: مقادیر شدید در متغیر نتیجه (y).
* نقاط اهرمی بالا: مقادیر شدید در متغیر پیش‌بینی کننده (x).

تمام این مفروضات و مسائل بالقوه را می‌توان با تولید برخی نمودارهای تشخیصی که خطاهای باقی مانده را تجسم می‌کند، بررسی کرد.

8.7 تشخیص رگرسیون {reg-diag}

* + 1. نمودارهای تشخیصی

نمودارهای تشخیص رگرسیون را می‌توان با استفاده از تابع پایه R plot() یا تابع autoplotO [ggfortify بسته] ایجاد کرد که یک گرافیک مبتنی بر ggplot2 ایجاد می‌کند.

* نمودارهای تشخیصی را با تابع پایه R ایجاد کنید:

**par** fmfrow = **c (**2، 2))

**طرح (**مدل)

10 15 20 25

* نمودارهای تشخیصی را با استفاده از ggfortify ایجاد کنید:

**کتابخانه (**ggfortify) **اتوپلات (**مدل)

نمودارهای تشخیصی باقیمانده‌ها را به چهار روش مختلف نشان می‌دهند:

1. **باقیمانده در مقابل برازش** شده برای بررسی مفروضات رابطه خطی استفاده می‌شود. یک خط افقی، بدون الگوهای متمایز، نشانه یک رابطه خطی است، مناسب است.
2. **QQ** معمولی برای بررسی اینکه آیا باقیمانده‌ها به‌طور نرمال توزیع شده اند یا خیر استفاده می‌شود. اگر نقاط باقیمانده از خط چین مستقیم پیروی کنند، خوب است.
3. **Scale-Location (**یا Spread-Location). برای بررسی همگنی واریانس باقیمانده‌ها (همسان سازی) استفاده می‌شود. خط افقی با نقاط پراکنده یکسان نشانه خوبی از همسویی است. این مورد در مثال ما نیست، جایی که ما مشکل ناهمسانی داریم.
4. باقیمانده‌ها مقابل **اهرم** برای شناسایی موارد تأثیرگذار استفاده می‌شود، یعنی مقادیر شدیدی که ممکن است در صورت گنجاندن یا حذف از تحلیل بر نتایج رگرسیون تأثیر بگذارند. این طرح در بخش‌های بعدی بیشتر توضیح داده خواهد شد.

چهار نمودار، 3 نقطه داده فوق العاده را نشان می‌دهد که با شماره ردیف داده‌ها در مجموعه داده برچسب زده شده است. آنها ممکن است بالقوه مشکل ساز باشند. ممکن است بخواهید به‌طور جداگانه به آنها نگاهی دقیق بیندازید تا بررسی کنید که آیا چیز خاصی برای موضوع وجود دارد یا اینکه ممکن است صرفاً خطاهای ورودی داده باشد. در بخش‌های بعدی در این مورد بحث خواهیم کرد.

معیارهای مورد استفاده برای ایجاد نمودارهای فوق در داده‌های model.diag.metrics موجود است که در بخش قبل توضیح داده شد.

* *اضافه کردن شاخص‌های مشاهدات و*
* *چند ستون (.se.fit، .sigma) را برای ساده‌سازی رها کنید*

model.diag.metrics <- model.diag.metrics °/ o >°/o

**جهش (**index = 1 **:nrow (**model.diag.metrics)) **/>/**

**انتخاب کنید (**شاخص، **همه چیز (**)، **-** .se.fit، **-** .sigma)

* *داده‌ها را بررسی کنید*

**سر (**model.diag.metrics, 4)

## فهرست فروش یوتیوب .fitted .resid .hat .cooksd .std.resid

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ## | 1 | 1 | 26. | .5 | 276. | . 1 | 21. | .56 | 4. | 0.96 0. | 0.00970 0.00794 | 1. | .273 |
| ## | 2 | 2 | 12. | .5 | 53. | .4 | 10. | .98 | 1. | 0.50 0. | 0.01217 0.00092 | 0. | .387 |
| ## | 3 | 3 | 11. | .2 | 20. | .6 | 9. | .42 | 1. | 0.74 0. | 0.01649 0.00169 | 0. | .449 |
| ## | 4 | 4 | 22. | .2 | 181. | .8 | 17. | .08 | 5. | . 12 0. | 00501 0.00434 | 1. | .312 |

ما عمدتا از ستون‌های زیر استفاده خواهیم کرد:

* .fitted : مقادیر برازش شده
* .resid : خطاهای باقیمانده
* .hat : مقادیر کلاه، برای تشخیص نقاط اهرمی بالا (یا مقادیر شدید در متغیرهای پیش‌بینی‌کننده x) استفاده می‌شود.
* .std.resid : باقیمانده‌های استاندارد شده که باقیمانده‌ها بر خطاهای استاندارد آنها تقسیم می‌شوند. برای تشخیص نقاط پرت (یا مقادیر شدید در متغیر نتیجه y) استفاده می‌شود.
* .cooksd : فاصله کوک، برای تشخیص مقادیر تاثیرگذار استفاده می‌شود که می‌تواند یک نقطه پرت یا یک نقطه اهرم بالا باشد.

در بخش بعدی، نحوه استفاده از این نمودارها و معیارها برای بررسی مفروضات رگرسیون و تشخیص مسائل بالقوه در مدل را به تفصیل شرح خواهیم داد.

* 1. خطی بودن داده ها

فرض خطی بودن را می‌توان با بررسی نمودار **باقیمانده در مقابل برازش (** نمودار 1): **نمودار (**مدل، 1) بررسی کرد.

باقیمانده در مقابل نصب شده

10 15 20 25

مقادیر   
مناسب lm (فروش ~ یوتیوب)

در حالت ایده آل، نمودار باقیمانده هیچ الگوی متناسبی را نشان نخواهد داد. یعنی خط قرمز باید ­تقریباً افقی در صفر باشد. وجود یک الگو ممکن است نشان دهنده مشکل در برخی از جنبه‌های مدل خطی باشد.

در مثال ما، هیچ الگوی در نمودار باقیمانده وجود ندارد. این نشان می‌دهد که می‌توانیم رابطه خطی بین پیش‌بینی کننده‌ها و متغیرهای نتیجه را فرض کنیم.

توجه داشته باشید که اگر نمودار باقیمانده نشان دهنده یک رابطه غیرخطی در داده‌ها باشد، یک روش ساده استفاده از تبدیل‌های غیرخطی پیش‌بینی کننده‌ها مانند log(x)، sqrt(x) و x""2 است. مدل رگرسیون

* 1. همگنی واریانس

این فرض را می‌توان با بررسی نمودار *مقیاس-مکان که به‌عنوان نمودار مکان گسترش ­*نیز شناخته می‌شود، بررسی کرد .

**طرح (**مدل، 3)

مقادیر   
مناسب lm (فروش ~ یوتیوب)

این نمودار نشان می‌دهد که آیا باقیمانده‌ها به‌طور مساوی در امتداد محدوده پیش‌بینی کننده‌ها پخش می‌شوند. خوب است اگر یک خط افقی با نقاط پراکنده یکسان ببینید. در مثال ما اینطور نیست.

مشاهده می‌شود که تغییرپذیری (واریانس) نقاط باقیمانده با مقدار متغیر نتیجه برازش شده افزایش می‌یابد که نشان دهنده واریانس‌های غیر ثابت در خطاهای باقیمانده (یا *ناهمسانی* ) است.

یک راه حل ممکن برای کاهش مشکل ناهمسانی، استفاده از تبدیل log یا ریشه دوم متغیر نتیجه (y) است.

model2 <- **Im (** **log (**فروش) **~** youtube، داده = بازاریابی) **نمودار (**model2, 3)

مقیاس-موقعیت

0131

2.2 2.4 2.6 2.8 3.0 3.2

مقادیر   
مناسب lm (ورود (فروش) ~ یوتیوب)

* 1. نرمال بودن باقیمانده ها

نمودار QQ باقیمانده را می‌توان برای بررسی بصری فرض نرمال بودن استفاده کرد. نمودار احتمال عادی باقیمانده‌ها باید تقریباً از یک خط مستقیم پیروی کند.

در مثال ما، تمام نقاط تقریباً در امتداد این خط مرجع قرار می‌گیرند، بنابراین می‌توانیم نرمال بودن را فرض کنیم.

**طرح (**مدل، 2)

Quantiles نظری   
lm (فروش ~ یوتیوب)

* 1. نقاط پرت و اهرم بالا

پرت :

نقطه پرت نقطه‌ای است که دارای یک مقدار متغیر نتیجه شدید است. وجود نقاط پرت ممکن است بر تفسیر مدل تأثیر بگذارد، زیرا RSE را افزایش می‌دهد.

نقاط پرت را می‌توان با بررسی *باقیمانده استاندارد شده (*یا *باقیمانده دانشجویی (studentized residual)* ) که باقیمانده تقسیم بر خطای استاندارد تخمینی آن است، شناسایی کرد. باقیمانده‌های استاندارد شده را می‌توان به‌عنوان تعداد خطاهای استاندارد دور از خط رگرسیون تفسیر کرد.

مشاهداتی که باقیمانده‌های استاندارد شده آنها بیشتر از 3 در قدر مطلق است، نقاط پرت ممکن هستند (جیمز و همکاران، 2014).

نقاط اهرمی بالا (High leverage points) :

یک نقطه داده در صورتی که دارای مقادیر x پیش‌بینی‌کننده شدید باشد، اهرم بالایی دارد. این را می‌توان با بررسی آمار اهرمی یا *مقدار کلاه تشخیص* داد. مقدار این آمار بالاتر از 2 (p + l)/n نشان دهنده یک مشاهده با اهرم بالا است (Bruce and Bruce, 2017). که در آن p تعداد پیش‌بینی‌ها و n تعداد مشاهدات است.

نقاط پرت و اهرم بالا را می‌توان با بررسی نمودار *باقیمانده در مقابل اهرم شناسایی* کرد:

**طرح (**مدل، 5)

باقیمانده‌ها در مقابل اهرم

**cn co**

**"O**

فاصله آشپزی

0.000 0.005 0.010 0.015 0.020

اهرمی   
(فروش ~ یوتیوب)

نمودار بالا 3 نقطه افراطی بالا (#26، #36 و #179)، با باقیمانده استاندارد شده زیر -2 را برجسته می‌کند. با این حال، هیچ نقطه پرت وجود ندارد که بیش از 3 انحراف استاندارد باشد، چه خوب است.

علاوه بر این، هیچ نقطه اهرم بالایی در داده‌ها وجود ندارد. یعنی همه نقاط داده دارای یک آمار اهرمی زیر 2 (p + 1)/n = 4/200 = 0.02 هستند.

* 1. مقادیر تاثیرگذار

مقدار تأثیرگذار مقداری است که گنجاندن یا حذف آن می‌تواند نتایج تحلیل رگرسیون را تغییر دهد. چنین مقداری با یک باقیمانده بزرگ همراه است.

همه نقاط پرت (یا نقاط داده شدید) در تحلیل رگرسیون خطی تأثیرگذار نیستند.

آماردانان معیاری به نام *فاصله کوک* برای تعیین تأثیر یک مقدار ایجاد کرده اند. این شاخص تأثیر را به‌عنوان ترکیبی از اهرم و اندازه باقیمانده تعریف می‌کند.

4/(n - p - 1) بیشتر شود، یک مشاهده تأثیر زیادی دارد (بروس و بروس، 2017) که در آن n تعداد مشاهدات و p تعداد متغیرهای پیش‌بینی کننده است.

طرح *Residuals vs Leverage می‌*تواند به ما در یافتن مشاهدات تأثیرگذار در صورت وجود کمک کند. در این نمودار، مقادیر بیرونی معمولاً در گوشه بالا سمت راست یا در گوشه پایین سمت راست قرار دارند. این نقاط مکان‌هایی هستند که نقاط داده می‌توانند در برابر خط رگرسیون تأثیرگذار باشند.

نمودارهای زیر فاصله کوک و اهرم مدل ما را نشان می‌دهد:

* *فاصله آشپزی*

**طرح (**مدل، 4)

* *باقیمانده‌ها در مقابل اهرم*

**طرح (**مدل، 5)

قدرت نفوذ

0.000 0.010 0.020

به‌طور پیش‌فرض، 3 مقدار بالای افراطی در نمودار فاصله کوک برچسب‌گذاری شده‌اند. اگر می‌خواهید 5 مقدار فوق العاده بالا را برچسب گذاری کنید، گزینه id.n را به صورت زیر مشخص کنید:

**نمودار (**مدل، 4، id.n = 5)

اگر می‌خواهید به این 3 مشاهدات برتر با بالاترین فاصله کوک نگاه کنید تا بخواهید آنها را بیشتر ارزیابی کنید، این کد R را تایپ کنید:

model.diag.metrics °/ o >°/o

**top\_n (**3، wt = .cooksd)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ##  ## | 1 | فهرست مطالب  26 | فروش یوتیوب | | | .مناسب  23.4 | | اقامت | | | .کلاه  0142 | .cooksd 0.0389 | | .std.resid  -2.33 | |
| 14.4 | | 315 | -9. | .04 | 0. |
| ## | 2 | 36 | 15. | .4 | 349 | 25. | .0 | -9. | .66 | 0. | .0191 | 0. | 0605 | -2. | .49 |
| ## | 3 | 179 | 14. | .2 | 332 | 24. | .2 | -10. | .06 | 0. | 0165 | 0. | 0.0563 | -2. | .59 |

هنگامی که نقاط داده امتیازات فاصله کوک بالایی دارند و در سمت راست بالا یا پایین نمودار اهرم قرار دارند، به این معنی که بر نتایج رگرسیون تأثیرگذار هستند. اگر آن موارد را حذف کنیم، نتایج رگرسیون تغییر خواهد کرد.

در مثال ما، داده‌ها هیچ نقطه تاثیرگذاری را نشان نمی‌دهند. خطوط فاصله کوک (یک خط چین قرمز) در نمودار Residuals vs Leverage نشان داده نمی‌شوند زیرا همه نقاط به خوبی در داخل خطوط فاصله کوک قرار دارند.

بیایید اکنون مثال دیگری را نشان دهیم که در آن داده‌ها حاوی دو مقدار افراطی با تأثیر بالقوه بر نتایج رگرسیون هستند:

df2 <- **data.frame (**

x = **c (**بازاریابی در یوتیوب، 500، 600 **دلار )،**

y = **c (**فروش دلارهای بازاریابی،80، 100)

)

model2 <- **lm (**y **~** x، df2)

نمودار *Residuals vs Leverage* دو مدل را ایجاد کنید:

* *فاصله آشپزی*

**طرح (**مدل 2، 4)

*#* **نمودار باقیمانده در مقابل اهرم (**مدل 2، 5)

Cook's distance

cc

co

TD w

Lc:

2(1

36

0 50 100 150 200

Obs. اهرم شماره

در نمودار Residuals vs Leverage، به دنبال نقطه داده خارج از خط چین، فاصله کوک باشید. وقتی امتیازها خارج از فاصله آشپز هستند، به این معنی است که امتیازات فاصله کوک بالایی دارند. در این حالت، مقادیر بر نتایج رگرسیون تأثیرگذار هستند. اگر آن موارد را حذف کنیم، نتایج رگرسیون تغییر خواهد کرد.

در مثال 2 بالا، دو نقطه داده بسیار فراتر از خطوط فاصله کوک هستند. باقی مانده‌ها به صورت خوشه‌ای در سمت چپ ظاهر می‌شوند. طرح، مشاهده تاثیرگذار را به‌عنوان #20 شناسایی کرد! و #202. اگر این نقاط را از تحلیل حذف کنید، ضریب شیب از 0.06 به 0.04 و R2 از 0.5 به 0.6 تغییر می‌کند. تاثیر بسیار بزرگ!

8.13 بحث

این فصل مفروضات رگرسیون خطی را تشریح می‌کند و نحوه تشخیص مسائل احتمالی در مدل را نشان می‌دهد.

تشخیص اساساً با تجسم باقیمانده‌ها انجام می‌شود. وجود الگوها در باقیمانده‌ها نشان دهنده توقف نیست. مدل رگرسیون فعلی شما ممکن است بهترین راه برای درک داده‌های شما نباشد.

مسائل احتمالی ممکن است:

* روابط غیر خطی بین نتیجه و متغیرهای پیش‌بینی کننده. هنگام مواجهه با این مشکل، یک راه حل این است که عبارت درجه دوم را شامل شود، مانند عبارت‌های چند جمله‌ای یا تبدیل log. فصل 6 را ببینید.
* وجود متغیرهای مهمی که از مدل خود حذف کرده اید. متغیرهای دیگری که وارد نکرده اید (به‌عنوان مثال، سن یا جنسیت) ممکن است نقش مهمی در مدل و داده‌های شما داشته باشند. به فصل 10 مراجعه کنید.
* وجود نقاط پرت. اگر فکر می‌کنید که به دلیل اشتباه در جمع‌آوری و ورود داده‌ها، یک نقطه پرت رخ داده است، یک راه‌حل این است که به سادگی مشاهدات مربوطه را حذف کنید.

فصل 9

چند خطی

* 1. مقدمه

در رگرسیون چندگانه (فصل 3)، دو یا چند متغیر پیش‌بینی ممکن است با یکدیگر همبستگی داشته باشند. این وضعیت به‌عنوان *هم خطی (collinearity) نامیده می‌شود* .

یک وضعیت شدید به نام **چند خطی (multicollinearity)** وجود دارد که در آن همخطی بین سه یا چند متغیر وجود دارد حتی اگر هیچ جفت متغیری همبستگی بالایی نداشته باشد. این بدان معنی است که بین متغیرهای پیش‌بینی افزونگی (redundancy) وجود دارد.

در صورت وجود چند خطی، حل مدل رگرسیون ناپایدار می‌شود.

برای یک پیش‌بینی‌کننده معین (p)، چند خطی بودن را می‌توان با محاسبه نمره‌ای به نام ضریب تورم واریانس (**vari­ance inflation factor**) (یا VIF) ارزیابی **کرد که اندازه‌گیری** می‌کند که چقدر واریانس یک ضریب رگرسیون ­به دلیل چند خطی بودن در مدل متورم شده است.

کوچکترین مقدار ممکن VIF یک است (عدم وجود چند خطی). به‌عنوان یک قاعده کلی، یک مقدار VIF که بیش از 5 یا 10 باشد، مقدار مشکلی از همخطی بودن را نشان می‌دهد (جیمز و همکاران، 2014).

هنگامی که با چند خطی بودن مواجه می‌شویم، متغیرهای مربوطه باید حذف شوند، زیرا وجود چند خطی به این معنی است که اطلاعاتی که این متغیر در مورد پاسخ ارائه می‌دهد در حضور سایر متغیرها اضافی است (جیمز و همکاران، 2014، بروس و بروس (2017) )).

این فصل نحوه تشخیص چند خطی در یک مدل رگرسیون با استفاده از R را توضیح می‌دهد.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید

**کتابخانه (**Tidyverse)

**کتابخانه (**کارت)

* 1. آماده‌سازی داده ها

ما از مجموعه داده‌های بوستون [در بسته MASS ]، معرفی شده در فصل 2، برای پیش‌بینی مقدار خانه متوسط ( mdev )، در حومه بوستون، بر اساس متغیرهای پیش‌بینی کننده چندگانه استفاده خواهیم کرد.

ما داده‌ها را به صورت تصادفی به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم می‌کنیم. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *داده‌ها را بارگذاری کنید*

**داده (**"بوستون"، بسته = "MASS")

* *تقسیم داده‌ها به مجموعه آموزشی و تست* **set.seed (**123)

training.samples <- Boston$medv °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE) train.data <- Bostonftraining.samples، ] test.data <- Bostonf **-** training.samples، ]

* 1. ساخت مدل رگرسیون

مدل رگرسیون زیر شامل همه متغیرهای پیش‌بینی می‌شود: *# مدل را بسازید*

modell <- **lm (**medv **~** ., data = train.data)

* *پیش‌بینی کنید*

پیش‌بینی <- modell 0 / 0 > 0 / 0 **پیش‌بینی (**test.data)

* *عملکرد مدل*

**data.frame (**

RMSE = **RMSE (**پیش‌بینی‌ها، test.data **$** medv)،

R2 = **R2 (**پیش‌بینی‌ها، test.data **$** medv)

)

* # RMSE R2
* # 1 4.99 0.67
  1. تشخیص چند خطی

تابع R vif() [بسته خودرو] می‌تواند برای تشخیص چند خطی بودن در یک مدل رگرسیونی استفاده شود: car **::vif (**modell)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ## | جنایت | zn | صنعت | چس | نوکس | rm | سن | دیس | راد |
| ## | 1.87 | 2.36 | 3.90 | 1.06 | 4.47 | 2.01 | 3.02 | 3.96 | 7.80 |
| ## | مالیات | ptratio | سیاه | lstat |  |  |  |  |  |
| ## | 9.16 | 1.91 | 1.31 | 2.97 |  |  |  |  |  |

مالیات متغیر پیش‌بینی کننده بسیار بالا است (VIF = 9.16). این ممکن است مشکل ساز باشد.

* 1. برخورد با چند خطی (Dealing with multicollinearity)

در این بخش، مدل خود را با حذف متغیرهای پیش‌بینی کننده با مقدار VIF بالا به روز می‌کنیم:

*# مدلی بدون در نظر گرفتن متغیر مالیات بسازید*

model2 <- **lm (**medv **~** . **-** مالیات، داده = train.data)

*# بسازید، پیش‌بینی کنید*

پیش‌بینی‌ها <- model2 °/ o >°/o **پیش‌بینی (**test.data) *# عملکرد مدل*

**data.frame (**

RMSE = **RMSE (**پیش‌بینی‌ها، test.data **$** medv)، R2 = **R2 (**پیش‌بینی‌ها، test.data **$** medv)

)

## RMSE R2

## 1 5.01 0.671

مشاهده می‌شود که حذف متغیر مالیاتی بر معیارهای عملکرد مدل تأثیر چندانی ندارد.

9.7 بحث

این فصل نحوه تشخیص و مقابله با چند خطی بودن در مدل‌های رگرسیون را شرح می‌دهد. مسائل ­چند خطی شامل، در مدل، متغیرهای مختلف است که رابطه پیش‌بینی مشابهی با نتیجه دارند. این را می‌توان برای هر پیش‌بینی با محاسبه مقدار VIF ارزیابی کرد.

هر متغیری با مقدار VIF بالا (بالای 5 یا 10) باید از مدل حذف شود. این منجر به یک مدل ساده تر بدون به خطر انداختن دقت مدل می‌شود که خوب است.

توجه داشته باشید که در یک مجموعه داده بزرگ که چندین متغیر پیش‌بینی کننده همبسته را ارائه می‌کند، می‌توانید رگرسیون مؤلفه اصلی و استراتژی‌های رگرسیون حداقل مربعات جزئی را انجام دهید. به فصل 19 مراجعه کنید.

متغیرهای مخدوش کننده (Confounding Variables)

یک **متغیر مداخله گر (Confounding variable) متغیر** مهمی است که باید در مدل پیش‌بینی گنجانده شود، اما شما آن را حذف می‌کنید. تفسیر ساده چنین مدل‌هایی می‌تواند منجر به نتیجه گیری‌های نامعتبر شود.

به‌عنوان مثال، در نظر بگیرید که می‌خواهیم هزینه زندگی در کشورهای مختلف را بر اساس تولید ناخالص داخلی سرانه، با استفاده از مجموعه داده‌های gapminder ، مدل کنیم :

**کتابخانه (**gapminder)

**lm (**lifeExp **~** gdpPercap، داده = gapminder)

در این مثال، واضح است که قاره یک متغیر مهم است: تخمین زده می‌شود که کشورهای اروپایی در مقایسه با کشورهای آفریقایی از امید به زندگی بالاتری برخوردار باشند. بنابراین، قاره یک متغیر مداخله گر است که باید در مدل گنجانده شود:

**lm (**lifeExp **~** gdpPercap **+** قاره، داده = gapminder)

**قسمت چهارم**

**اعتبارسنجی مدل رگرسیون**

مقدمه

هنگام ساخت یک مدل رگرسیون (فصل 3)، باید خوب بودن مدل را ارزیابی کنید، یعنی اینکه مدل چقدر با داده‌های آموزشی مورد استفاده برای ساخت مدل مطابقت دارد و چقدر مدل در پیش‌بینی نتیجه مشاهدات آزمایشی دیده نشده جدید دقیق است. .

در این بخش، تکنیک‌هایی برای ارزیابی دقت مدل رگرسیون و اعتبارسنجی عملکرد مدل را یاد خواهید گرفت. ما همچنین مثال‌های عملی را در R ارائه خواهیم داد.

فصول زیر پوشش داده شده است:

* معیارهای دقت مدل رگرسیون (فصل 12) برای سنجش عملکرد یک مدل رگرسیونی.
* اعتبارسنجی متقابل (فصل 13) و نمونه برداری مجدد بوت استرپ (فصل 14) برای اعتبارسنجی مدل در داده‌های آزمایشی.

معیارهای دقت مدل رگرسیون

* 1. مقدمه

در این فصل ما **معیارهای رگرسیون آماری مختلف را** برای سنجش عملکرد یک مدل رگرسیون توضیح خواهیم داد (فصل 3).

در مرحله بعد، مثال‌های عملی در R برای مقایسه عملکرد دو مدل ارائه می‌کنیم تا بهترین مدل را برای داده‌های خود انتخاب کنیم.

* 1. معیارهای عملکرد مدل

در مدل رگرسیون، متداول‌ترین معیارهای ارزیابی عبارتند از:

1. **R-squared (**R2) که نسبت تغییرات در نتیجه است که توسط متغیرهای پیش‌بینی توضیح داده می‌شود (یعنی چند درصد از تغییرات متغیر پاسخ توسط متغیرهای توصیفی توضیح داده شده است). در مدل‌های رگرسیون چندگانه، R2 با همبستگی مجذور بین مقادیر نتیجه مشاهده شده و مقادیر پیش‌بینی شده توسط مدل مطابقت دارد. هر چه R-squared بالاتر باشد، مدل بهتر است.
2. **ریشه میانگین مربعات خطا (**RMSE) که میانگین خطای انجام شده توسط مدل را در پیش‌بینی نتیجه یک مشاهده اندازه‌گیری می‌کند. از نظر ریاضی، RMSE جذر *میانگین مربعات خطا (MSE)* است که میانگین مجذور اختلاف بین مقادیر خروجی واقعی مشاهده شده و مقادیر پیش‌بینی‌شده توسط مدل است. بنابراین، MSE = mean((مشاهده - پیش‌بینی)~2) و RMSE = sqrt(MSE ). هرچه RMSE کمتر باشد، مدل بهتر است.
3. **خطای استاندارد باقیمانده (RSE) که به‌عنوان** *مدل سیگما* نیز شناخته می‌شود، نوعی از RMSE است که برای تعداد پیش‌بینی کننده‌ها در مدل تنظیم شده است. هرچه RSE کمتر باشد، مدل بهتر است. در عمل، تفاوت بین RMSE و RSE بسیار کوچک است، به ویژه برای داده‌های چند متغیره بزرگ.
4. **میانگین خطای مطلق (**MAE)، مانند RMSE، MAE خطای پیش‌بینی را اندازه‌گیری می‌کند. از نظر ریاضی، میانگین تفاوت مطلق بین خروجی مشاهده شده و پیش‌بینی شده، MAE = میانگین (abs(مشاهده - پیش‌بینی شده)) است. MAE نسبت به RMSE حساسیت کمتری نسبت به مقادیر پرت دارد.

مشکل معیارهای فوق این است که آنها نسبت به گنجاندن متغیرهای اضافی در مدل معقول هستند، حتی اگر آن متغیرها سهم قابل‌توجهی در توضیح نتیجه نداشته باشند. به عبارت دیگر، گنجاندن متغیرهای اضافی در مدل همیشه باعث افزایش R2 و کاهش RMSE می‌شود. بنابراین، برای هدایت انتخاب مدل به یک معیار قوی‌تر نیاز داریم.

در مورد R2، یک نسخه تنظیم شده به نام **Adjusted R-squared** وجود دارد که R2 را برای داشتن متغیرهای بیش از حد در مدل تنظیم می‌کند.

علاوه بر این، چهار معیار مهم دیگر وجود دارد - **AIC،** **AICc،** **BIC** و **Mallows Cp** - که معمولاً برای ارزیابی و انتخاب مدل استفاده می‌شوند. اینها یک برآورد بی طرفانه از خطای پیش‌بینی مدل MSE هستند. هرچه این معیارها کمتر باشد، او مدل بهتری خواهد داشت.

1. **AIC** مخفف ( *Akaike's Information Criteria* )، معیاری است که توسط آماردان ژاپنی، Hirotugu Akaike، 1970 توسعه یافته است. ایده اصلی AIC جریمه کردن گنجاندن متغیرهای اضافی در یک مدل است. جریمه‌ای اضافه می‌کند که هنگام اضافه کردن شرایط اضافی، خطا را افزایش می‌دهد. هرچه AIC کمتر باشد، مدل بهتر است.
2. **AICc** نسخه‌ای از AIC است که برای اندازه‌های نمونه کوچک تصحیح شده است.
3. **BIC (**یا *معیارهای اطلاعات بیزی یا Bayesian information criteria*) گونه‌ای از AIC با جریمه قوی‌تر برای گنجاندن متغیرهای اضافی به مدل است.
4. **Mallows Cp** : گونه‌ای از AIC که توسط Colin Mallows توسعه یافته است.

به‌طور کلی، متداول ترین معیارهای مورد استفاده برای اندازه‌گیری کیفیت مدل رگرسیون و مقایسه مدل‌ها عبارتند از: Adjusted R2، AIC، BIC و Cp.

در بخش‌های بعدی، نحوه محاسبه این معیارهای ذکر شده در بالا را به شما نشان خواهیم داد.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم داده ها
* مدلر توابع کمکی را برای محاسبه معیارهای عملکرد مدل رگرسیون فراهم می‌کند
* بروم به راحتی یک قاب داده مرتب ایجاد می‌کند که شامل معیارهای آماری مدل است

**کتابخانه (**Tidyverse)

**کتابخانه (**مدل ساز)

**کتابخانه (**جارو)

* 1. نمونه‌ای از داده ها

ما از داده‌های داخلی R سوئیس، معرفی شده در فصل 2، برای پیش‌بینی امتیاز باروری بر اساس شاخص‌های اجتماعی-اقتصادی استفاده خواهیم کرد.

*# داده‌های داده را بارگیری کنید (*"سوئیس") *# نمونه داده‌ها را بازرسی کنید (* **سوئیسی،** 3)

* 1. ساخت مدل‌های رگرسیون

ما با ایجاد دو مدل شروع می‌کنیم:

1. مدل 1، شامل همه پیش‌بینی ها
2. مدل 2، شامل همه پیش‌بینی کننده‌ها به جز متغیر Examination

modell <- **Im (**باروری **~** .، داده = سوئیسی)

model2 <- **Im (**باروری **~** . **-** معاینه، داده = سوئیس)

* 1. ارزیابی کیفیت مدل

توابع و بسته‌های R زیادی برای ارزیابی کیفیت مدل وجود دارد، از جمله:

* summary() [بسته آماری]، R-squared، R-squared تنظیم شده و RSE را برمی گرداند.
* AIC() و BIC() [بسته آماری]، AIC و BIC را به ترتیب **خلاصه (**مدل) **AIC (**مدل) **BIC (**مدل) محاسبه می‌کند.
* rsquare()، rmse() و mae() [بسته مدلr]، به ترتیب R2، RMSE و MAE را محاسبه می‌کند.

**library (**modelr) **data.frame (**

R2 = **rsquare (**مدل، داده = سوئیس)،

RMSE = **rmse (**مدل، داده = سوئیس)، MAE = **mae (**مدل، داده = سوئیس)

)

* R2()، RMSE() و MAE() [بسته مراقبت]، به ترتیب R2، RMSE و MAE را محاسبه می‌کند.

**کتابخانه (**کارت)

پیش‌بینی‌ها <- modell °/ o >°/o **پیش‌بینی (**سوئیسی) **data.frame (**

R2 = **R2 (پیش‌بینی‌ها،** باروری **دلار** سوئیس )،

RMSE = **RMSE (**پیش‌بینی‌ها، باروری **دلار سوئیس** )، MAE = **MAE (**پیش‌بینی‌ها، باروری **دلار سوئیس)**

)

* glance() [بسته جارو]، R2، R2 تنظیم شده، سیگما (RSE)، AIC، BIC را محاسبه می‌کند. **کتابخانه (**جارو) **نگاه (**مدل)
* محاسبه دستی R2، RMSE و MAE:

*# پیش‌بینی کنید و # R2، RMSE و MAE* سوئیس °/o>°/o را محاسبه کنید

**add\_predictions (**مدل) **°/ o >°/o**

**خلاصه کردن (**

R2 = **cor (**باروری، pred) **~** 2،

MSE = **میانگین (**(باروری **-** پیش از) **~** 2)،

RMSE = **sqrt (**MSE)، MAE = **میانگین (** **abs (**باروری **-** قبل))

)

* 1. مقایسه عملکرد مدل‌های رگرسیون

در اینجا، ما از تابع ()glance برای مقایسه کیفیت کلی دو مدل خود استفاده می‌کنیم:

* *معیارهای مدل 1*

**نگاه (**مدل) °/ o >°/o

**انتخاب کنید (**adj.r.squared، sigma، AIC، BIC، p.value)

* # adj.r.squared sigma AIC BIC p.value
* # l 0.671 7.17 326 339 5.59e-10
* *معیارهای مدل 2*

**نگاه (**model2) **/>/**

**انتخاب کنید (**adj.r.squared، sigma، AIC، BIC، p.value)

* # adj.r.squared sigma AIC BIC p.value
* # 1 0.671 7.17 325 336 1.72e-10

از خروجی بالا می‌توان دریافت که:

1. دو مدل دقیقاً همان **R2 تنظیم شده (**0.67) دارند، به این معنی که آنها در توضیح نتیجه، در اینجا امتیاز باروری، معادل هستند. علاوه بر این، آنها همان مقدار **خطای استاندارد باقیمانده را دارند (**RSE یا sigma = 7.17). با این حال، مدل 2 ساده تر از مدل 1 است زیرا متغیرهای کمتری را در خود جای می‌دهد. همه چیز برابر است، مدل ساده همیشه در آمار بهتر است.
2. AIC و BIC مدل 2 کمتر از مدل مدل است. در استراتژی‌های مقایسه مدل، مدلی که کمترین امتیاز AIC و BIC را داشته باشد ترجیح داده می‌شود.
3. در نهایت، مقدار p. آماره F مدل 2 کمتر از مدل 1 است. این به این معنی است که مدل 2 از نظر آماری در مقایسه با مدل 1 معنادارتر است که با نتیجه‌گیری فوق مطابقت دارد.

توجه داشته باشید که RMSE و RSE در مقیاسی مشابه با متغیر نتیجه اندازه‌گیری می‌شوند. تقسیم RSE بر مقدار متوسط متغیر نتیجه، نرخ خطای پیش‌بینی را به شما می‌دهد که باید تا حد امکان کوچک باشد:

**سیگما (**مدل 1) **/ میانگین (** باروری **دلار** سوئیس )

## [1] 0.102

در مثال ما میانگین نرخ خطای پیش‌بینی 10 درصد است.

12.8 بحث

این فصل چندین معیار را برای ارزیابی عملکرد کلی یک مدل رگرسیون توصیف می‌کند.

مهمترین معیارها عبارتند از Adjusted R-square، RMSE، AIC و BIC. این معیارها همچنین به‌عنوان مبنای مقایسه مدل و انتخاب مدل بهینه استفاده می‌شوند.

توجه داشته باشید که این معیارهای رگرسیون همه معیارهای داخلی هستند، یعنی بر روی همان داده‌هایی که برای ساخت مدل رگرسیون استفاده شده است، محاسبه شده اند. آنها به شما می‌گویند که چقدر مدل با داده‌های در دست مطابقت دارد که مجموعه داده‌های آموزشی نامیده می‌شود.

به‌طور کلی، ما واقعاً اهمیتی نمی‌دهیم که این روش چقدر روی داده‌های آموزشی کار می‌کند. در عوض، ما به صحت پیش‌بینی‌هایی که هنگام اعمال روش خود بر روی داده‌های آزمایشی که قبلاً دیده نشده بود، به دست می‌آوریم، علاقه‌مندیم.

با این حال، داده‌های تست همیشه در دسترس نیستند و تخمین خطای تست را بسیار دشوار می‌کند. در این شرایط، روش‌هایی مانند **اعتبارسنجی متقابل (**فصل 13) و **بوت استرپ (**فصل 14) برای تخمین خطای آزمون (یا میزان خطای پیش‌بینی) با استفاده از داده‌های آموزشی استفاده می‌شود.

فصل 13

اعتبارسنجی متقابل

* 1. مقدمه

**اعتبارسنجی متقابل** به مجموعه‌ای از روش‌ها برای سنجش عملکرد یک مدل پیش‌بینی در مجموعه داده‌های آزمایشی جدید اشاره دارد.

ایده اصلی، پشت تکنیک‌های اعتبارسنجی متقابل، شامل تقسیم داده‌ها به دو مجموعه است:

1. مجموعه آموزشی، مورد استفاده برای آموزش (یعنی ساخت) مدل.
2. و مجموعه تست (یا مجموعه اعتبارسنجی) که برای آزمایش (یعنی اعتبارسنجی) مدل با تخمین خطای پیش‌بینی استفاده می‌شود.

اعتبارسنجی متقابل نیز به‌عنوان *روش نمونه‌گیری مجدد شناخته می‌شود* زیرا شامل برازش روش آماری یکسان چندین بار با استفاده از زیر مجموعه‌های مختلف داده است.

در این فصل یاد خواهید گرفت:

1. متداول ترین معیارهای آماری مورد استفاده (فصل 12) برای سنجش عملکرد یک مدل رگرسیون در پیش‌بینی نتیجه داده‌های آزمون جدید.
2. روش‌های مختلف اعتبارسنجی متقابل برای ارزیابی عملکرد مدل ما رویکردهای زیر را پوشش می‌دهیم:

* رویکرد مجموعه اعتبارسنجی (یا تقسیم داده ها)
* Leave One Out Cross Validation
* k-fold Cross Validation
* تکرار k-fold Cross Validation

هر کدام از این روش‌ها مزایا و معایب خود را دارند. از روشی استفاده کنید که به بهترین وجه با مشکل شما سازگار است. به‌طور کلی، (تکرار) اعتبار متقاطع k-fold توصیه می‌شود.

1. مثال‌های عملی از کدهای R برای محاسبه روش‌های اعتبارسنجی متقابل.
   1. بارگیری بسته‌های R مورد نیاز

* tidyverse برای دستکاری و تجسم آسان داده ها
* محاسبه آسان روش‌های اعتبارسنجی متقابل اهمیت دهید

**کتابخانه (**Tidyverse)

**کتابخانه (**کارت)

* 1. نمونه‌ای از داده ها

ما از داده‌های داخلی R سوئیس، معرفی شده در فصل 2، برای پیش‌بینی امتیاز باروری بر اساس شاخص‌های اجتماعی-اقتصادی استفاده خواهیم کرد. *# داده‌های داده را بارگیری کنید (*"سوئیس") *# نمونه داده‌ها را بازرسی کنید (* **سوئیسی،** 3)

* 1. معیارهای عملکرد مدل

پس از ساخت یک مدل، ما علاقه مندیم که دقت این مدل را در پیش‌بینی نتیجه مشاهدات نادیده جدید که برای ساخت مدل استفاده نشده اند، تعیین کنیم. به عبارت دیگر، می‌خواهیم خطای پیش‌بینی را تخمین بزنیم.

برای انجام این کار، استراتژی اساسی این است که:

1. مدل را بر روی یک مجموعه داده آموزشی بسازید
2. برای پیش‌بینی، مدل را روی مجموعه داده‌های آزمایشی جدید اعمال کنید
3. خطاهای پیش‌بینی را محاسبه کنید

در فصل 12، ما چندین معیار آماری را برای کمی کردن کیفیت کلی مدل‌های رگرسیون توضیح دادیم. این شامل:

* **R-squared (**R2)، نشان دهنده همبستگی مجذور بین مقادیر نتیجه مشاهده شده و مقادیر پیش‌بینی شده توسط مدل است. هرچه R2 تنظیم شده بالاتر باشد، مدل بهتر است.
* **ریشه میانگین مربعات خطا (**RMSE) که میانگین خطای پیش‌بینی ایجاد شده توسط مدل در پیش‌بینی نتیجه یک مشاهده را اندازه‌گیری می‌کند. یعنی میانگین تفاوت بین مقادیر نتیجه شناخته شده مشاهده شده و مقادیر پیش‌بینی شده توسط مدل. هرچه RMSE کمتر باشد، مدل بهتر است.
* **میانگین خطای مطلق (**MAE)، جایگزینی برای RMSE که حساسیت کمتری به مقادیر پرت دارد. با میانگین تفاوت مطلق بین نتایج مشاهده شده و پیش‌بینی شده مطابقت دارد. هرچه MAE کمتر باشد، مدل بهتر است

در تنظیم طبقه‌بندی، میزان خطای پیش‌بینی به‌عنوان نسبت مشاهدات طبقه‌بندی اشتباه تخمین زده می‌شود.

R2، RMSE و MAE برای سنجش عملکرد مدل رگرسیون در طول **اعتبارسنجی متقابل استفاده می‌شوند ­**.

در بخش بعدی، اصول اعتبارسنجی متقاطع را توضیح خواهیم داد و مثال عملی را با استفاده از بسته caret R ارائه خواهیم داد.

* 1. روش‌های اعتبارسنجی متقابل

به‌طور خلاصه، الگوریتم‌های اعتبارسنجی متقابل را می‌توان به صورت زیر خلاصه کرد:

1. نمونه کوچکی از مجموعه داده را رزرو کنید
2. مدل را با استفاده از بخش باقی مانده از مجموعه داده بسازید (یا آموزش دهید).
3. آزمایش اثربخشی مدل بر روی نمونه رزرو شده مجموعه داده ها. اگر مدل روی مجموعه داده‌های آزمایشی به خوبی کار کند، خوب است.

بخش‌های زیر تکنیک‌های مختلف اعتبارسنجی متقابل را توضیح می‌دهند.

* + 1. رویکرد مجموعه اعتبارسنجی

رویکرد مجموعه اعتبارسنجی شامل تقسیم تصادفی داده‌ها به دو مجموعه است: یک مجموعه برای آموزش مدل و مجموعه دیگر برای آزمایش مدل استفاده می‌شود.

فرآیند به صورت زیر عمل می‌کند:

1. مدل را بر روی مجموعه داده‌های آموزشی بسازید (آموزش دهید).
2. برای پیش‌بینی نتیجه مشاهدات نادیده جدید، مدل را در مجموعه داده‌های آزمایشی اعمال کنید
3. خطای پیش‌بینی را به صورت میانگین مجذور اختلاف بین مقادیر نتیجه مشاهده شده و پیش‌بینی‌شده محاسبه کنید.

مثال زیر مجموعه داده سوئیس را به گونه‌ای تقسیم می‌کند که 80% برای آموزش مدل رگرسیون خطی و 20% برای ارزیابی عملکرد مدل استفاده می‌شود.

*# داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید* **set.seed (**123) training.samples <- swiss$Fertility °/ o >°/o **createDataPartition (**p = 0.8, list = FALSE) train.data <- swiss[training.samples , ] test.data <- swiss[ **-** training.samples, ] *# مدل* مدل را بسازید <- **Im (**باروری **~** .، داده = سوئیس) *# پیش‌بینی کنید و پیش‌بینی‌های R2، RMSE و MAE را* محاسبه کنید <- model **/>/ پیش‌بینی (**test.data) **data.frame (**R2 = **R2 (**پیش‌بینی‌ها، test.data **$** باروری)، RMSE = **RMSE (**پیش‌بینی‌ها، test.data **$** باروری)، MAE = **MAE (**پیش‌بینی‌ها، test.data **$** باروری))

## R2 RMSE MAE ## 1 0.503 7.79 6.35

هنگام مقایسه دو مدل، مدلی که کمترین نمونه آزمایشی RMSE را تولید می‌کند، ­مدل ترجیحی است.

RMSE و MAE در همان مقیاس با متغیر نتیجه اندازه‌گیری می‌شوند. تقسیم RMSE بر مقدار متوسط متغیر نتیجه، نرخ خطای پیش‌بینی را به شما می‌دهد که باید تا حد امکان کوچک باشد:

**RMSE (**پیش‌بینی‌ها، test.data **$** باروری) **/mean (**test.data **$** باروری)

## [1] 0.109

توجه داشته باشید که روش مجموعه اعتبارسنجی تنها زمانی مفید است که مجموعه داده‌های بزرگی داشته باشید که بتوان آن را پارتیشن‌بندی کرد. یک عیب این است که ما یک مدل را فقط بر روی کسری از مجموعه داده‌ها می‌سازیم و احتمالاً اطلاعات جالبی در مورد داده‌ها حذف می‌کنیم که منجر به بایاس بیشتر می‌شود. بنابراین، بسته به اینکه کدام مشاهدات در مجموعه آموزشی و کدام مشاهدات در مجموعه اعتبار سنجی گنجانده شده است، میزان خطای آزمون می‌تواند بسیار متغیر باشد.

* + 1. یکی از اعتبارسنجی متقاطع را کنار بگذارید - LOOCV

این روش به صورت زیر عمل می‌کند:

* . یک نقطه داده را کنار بگذارید و مدل را بر روی بقیه مجموعه داده بسازید
* . مدل را در برابر نقطه داده‌ای که در مرحله 1 کنار گذاشته شده است آزمایش کنید و خطای تست مرتبط با پیش‌بینی را ثبت کنید
* . فرآیند را برای تمام نقاط داده تکرار کنید
* . با در نظر گرفتن میانگین تمام این تخمین‌های خطای آزمون ثبت شده در مرحله 2، خطای کلی پیش‌بینی را محاسبه کنید.

مثال عملی در R با استفاده از بسته caret :

* *کنترل تمرین را تعریف کنید*

train.control <- **trainControl (**روش = "LOOCV")

* *مدل را آموزش دهید*

مدل <- **قطار (**باروری **~** .، داده = سوئیس، روش = "lm"، trControl = train.control)

* *خلاصه کردن نتایج* **چاپ (**مدل)
* # رگرسیون خطی
* #
* # 47 نمونه
* پیش‌بینی شماره 5
* #
* # بدون پیش پردازش
* # نمونه برداری مجدد: اعتبارسنجی متقابل را ترک کنید
* # خلاصه حجم نمونه: 46، 46، 46، 46، 46، 46، ...
* # نتایج نمونه‌گیری مجدد:
* #
* # RMSE Rsquared MAE
* # 7.74 0.613 6.12
* #
* # پارامتر تنظیم 'intercept' در مقدار TRUE ثابت نگه داشته شد

مزیت روش LOOCV این است که ما از تمام نقاط داده استفاده می‌کنیم تا بایاس بالقوه را کاهش دهیم.

با این حال، فرآیند به تعداد دفعاتی که نقاط داده وجود دارد تکرار می‌شود، در نتیجه زمانی که n بسیار بزرگ است، زمان اجرای بالاتری خواهد داشت.

علاوه بر این، ما عملکرد مدل را در برابر یک نقطه داده در هر تکرار آزمایش می‌کنیم. اگر برخی از نقاط داده پرت باشند، این امر ممکن است منجر به تغییرات بیشتر در خطای پیش‌بینی شود. بنابراین، ما به نسبت خوبی از نقاط داده آزمایشی نیاز داریم، راه حلی که با **روش اعتبارسنجی متقاطع k-fold ارائه می‌شود** .

* + 1. اعتبارسنجی متقابل K-fold

روش اعتبارسنجی متقابل k-fold عملکرد مدل را در زیر مجموعه‌های مختلف داده‌های آموزشی ارزیابی می‌کند و سپس میانگین میزان خطای پیش‌بینی را محاسبه می‌کند. الگوریتم به شرح زیر است:

1. به‌طور تصادفی مجموعه داده‌ها را به k-زیر مجموعه (یا k-fold) تقسیم کنید (به‌عنوان مثال 5 زیر مجموعه)
2. یک زیر مجموعه را رزرو کنید و مدل را روی همه زیر مجموعه‌های دیگر آموزش دهید
3. مدل را روی زیر مجموعه رزرو شده تست کنید و خطای پیش‌بینی را ثبت کنید
4. این فرآیند را تا زمانی تکرار کنید که هر یک از k زیر مجموعه به‌عنوان مجموعه آزمایشی عمل کنند.
5. میانگین k خطای ثبت شده را محاسبه کنید. به این خطای اعتبارسنجی متقابل می‌گویند که به‌عنوان معیار عملکرد مدل عمل می‌کند.

اعتبارسنجی متقابل K-fold (CV) یک روش قوی برای تخمین دقت یک مدل است.

بارزترین مزیت CV k-fold در مقایسه با LOOCV محاسباتی است. یک مزیت کمتر آشکار اما بالقوه مهمتر K-fold CV این است که اغلب تخمین‌های دقیق تری از میزان خطای آزمون نسبت به LOOCV ارائه می‌دهد (جیمز و همکاران، 2014).

سوال معمولی، این است که چگونه مقدار صحیح k را انتخاب کنیم؟

مقدار کمتر K، بایاس تر و در نتیجه نامطلوب است. از سوی دیگر، مقدار بالاتر K بایاس کمتری دارد، اما می‌تواند از تغییرات زیادی رنج ببرد. دیدن اینکه مقدار کوچکتر k (مثلا k = 2) همیشه ما را به سمت رویکرد مجموعه اعتبار سنجی می‌برد، در حالی که مقدار بالاتر k (مثلا k = تعداد نقاط داده) ما را به رویکرد LOOCV هدایت می‌کند.

در عمل، به‌طور معمول اعتبارسنجی متقابل k برابر با استفاده از k = 5 یا k = 10 انجام می‌شود، زیرا این مقادیر به‌طور تجربی نشان داده شده است که تخمین‌های نرخ خطای آزمون را ارائه می‌دهند که نه از بایاس بیش از حد بالا و نه از واریانس بسیار بالا رنج می‌برند.

مثال زیر از اعتبارسنجی متقاطع 10 برابری برای تخمین خطای پیش‌بینی استفاده می‌کند. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *کنترل تمرین را تعریف کنید*

**set.seed (**123)

train.control <- **trainControl (**روش = "cv"، عدد = 10)

* *مدل را آموزش دهید*

مدل <- **قطار (**باروری **~** .، داده = سوئیس، روش = "Im"، trControl = train.control)

* *نتایج را خلاصه کنید*

**چاپ (**مدل)

* # رگرسیون خطی

##

* # 47 نمونه
* پیش‌بینی شماره 5

##

* # بدون پیش پردازش
* # نمونه برداری مجدد: اعتبار متقاطع (10 برابر)
* # خلاصه حجم نمونه: 43، 42، 42، 41، 43، 41، ...
* # نتایج نمونه‌گیری مجدد:

##

* # RMSE Rsquared MAE
* # 7.38 0.751 6.03

##

* # پارامتر تنظیم 'intercept' در مقدار TRUE ثابت نگه داشته شد
  + 1. اعتبارسنجی متقاطع K-fold مکرر

فرآیند تقسیم داده‌ها به k-fold می‌تواند چندین بار تکرار شود که به آن اعتبارسنجی متقاطع k-fold مکرر گفته می‌شود.

خطای مدل نهایی به‌عنوان خطای میانگین از تعداد تکرار گرفته می‌شود.

مثال زیر از اعتبارسنجی متقاطع 10 برابری با 3 تکرار استفاده می‌کند:

* *کنترل تمرین را تعریف کنید*

**set.seed (**123)

train.control <- **trainControl (**روش = "repeatedcv"، عدد = 10، تکرار = 3) *# آموزش مدل*

مدل <- **قطار (**باروری **~** .، داده = سوئیس، روش = "Im"، trControl = train.control)

* *خلاصه کردن نتایج* **چاپ (**مدل)

13.6 بحث

در این فصل، ما 4 روش مختلف را برای ارزیابی عملکرد یک مدل بر روی داده‌های آزمایش غیرقابل مشاهده توضیح دادیم.

این روش‌ها عبارتند از: رویکرد مجموعه اعتبار سنجی، اعتبارسنجی متقابل ترک تک، اعتبارسنجی متقابل k-fold ­و اعتبار متقاطع k-fold مکرر.

ما عموماً اعتبارسنجی متقاطع k-fold (تکرار) را برای تخمین میزان خطای پیش‌بینی توصیه می‌کنیم. می‌توان از آن در تنظیمات رگرسیون و طبقه‌بندی استفاده کرد.

جایگزین دیگری برای اعتبارسنجی متقاطع، روش‌های نمونه‌گیری مجدد بوت استرپ است (فصل 14) که شامل انتخاب مکرر و تصادفی نمونه‌ای از n مشاهدات از مجموعه داده اصلی و ارزیابی عملکرد مدل در هر نسخه است.

فصل 14

نمونه برداری مجدد بوت استرپ

* 1. مقدمه

مشابه تکنیک‌های اعتبارسنجی متقابل (فصل 13)، روش **نمونه‌گیری مجدد** **بوت استرپ (bootstrap resampling)** می‌تواند برای اندازه‌گیری دقت یک مدل پیش‌بینی‌کننده استفاده شود. علاوه بر این، می‌توان از آن برای اندازه‌گیری عدم قطعیت مرتبط با هر برآوردگر آماری استفاده کرد.

نمونه‌برداری مجدد بوت استرپ شامل انتخاب مکرر نمونه‌ای از n مشاهدات از مجموعه داده اصلی و ارزیابی مدل در هر نسخه است. سپس یک خطای استاندارد متوسط محاسبه می‌شود و نتایج نشان‌دهنده واریانس کلی عملکرد مدل است.

این فصل اصول راه‌اندازی را تشریح می‌کند و مثال‌های عملی را در R برای محاسبه خطای پیش‌بینی مدل ارائه می‌کند. علاوه بر این، نحوه محاسبه عدم قطعیت برآوردگر را با استفاده از تکنیک‌های بوت استرپ به شما نشان خواهیم داد.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* مراقبت برای محاسبات آسان **کتابخانه روش‌های اعتبارسنجی متقابل (** tidyverse )

**کتابخانه (**کارت)

* 1. نمونه‌ای از داده ها

ما از داده‌های داخلی R سوئیس، معرفی شده در فصل 2، برای پیش‌بینی امتیاز باروری بر اساس شاخص‌های اجتماعی-اقتصادی استفاده خواهیم کرد.

*# داده‌های داده را بارگیری کنید (*"سوئیس") *# نمونه داده‌ها را بازرسی کنید (* **سوئیسی،** 3)

* 1. رویه بوت استرپ

روش بوت استرپ برای تعیین کمیت عدم قطعیت مرتبط با یک برآوردگر آماری داده شده یا با یک مدل پیش‌بینی استفاده می‌شود.

این شامل انتخاب تصادفی یک نمونه از n مشاهدات از مجموعه داده اصلی است. این زیرمجموعه به نام مجموعه داده بوت استرپ برای ارزیابی مدل استفاده می‌شود.

این روش چندین بار تکرار می‌شود و سپس خطای استاندارد تخمین بوت استرپ محاسبه می‌شود. نتایج نشان‌دهنده واریانس عملکرد مدل‌ها هستند.

توجه داشته باشید که نمونه برداری با جایگزینی انجام می‌شود، به این معنی که مشاهده یکسان می‌تواند بیش از یک بار در مجموعه داده بوت استرپ رخ دهد.

* 1. ارزیابی عملکرد یک مدل پیش‌بینی

مثال زیر از یک بوت استرپ با 100 نمونه مجدد برای آزمایش مدل رگرسیون خطی استفاده می‌کند:

* *کنترل تمرین را تعریف کنید*

train.control <- **trainControl (**روش = "boot"، عدد = 100) *# آموزش مدل*

مدل <- **قطار (**باروری **~** .، داده = سوئیس، روش = "Im"، trControl = train.control)

* *خلاصه کردن نتایج* **چاپ (**مدل)
* # رگرسیون خطی
* #
* # 47 نمونه
* پیش‌بینی شماره 5
* #
* # بدون پیش پردازش
* # نمونه برداری مجدد: بوت استرپ (100 تکرار)
* # خلاصه حجم نمونه: 47، 47، 47، 47، 47، 47، ...
* # نتایج نمونه‌گیری مجدد:
* #

|  |  |  |  |
| --- | --- | --- | --- |
| ## | RMSE | Rsquared | MAE |
| ## | 8.13 | 0.608 | 6.54 |
| ## |  |  |  |
| ## | تنظیم | پارامتر | 'عرض از مبدأ' در مقدار TRUE ثابت نگه داشته شد |

خروجی میانگین عملکرد مدل را در 100 نمونه مجدد نشان می‌دهد.

RMSE (ریشه میانگین مربعات خطا) و MAE (میانگین خطای مطلق)، دو معیار متفاوت از خطای پیش‌بینی مدل را نشان می‌دهند. هرچه RMSE و MAE کمتر باشد، مدل بهتر است. R-squared نشان دهنده نسبت تغییرات در نتیجه است که توسط متغیرهای پیش‌بینی موجود در مدل توضیح داده شده است. هر چه R-squared بالاتر باشد، مدل بهتر است. بیشتر در مورد این معیارها در فصل 12 بخوانید.

14.6 کمی کردن عدم قطعیت برآوردگر و اطمینان در ­بازه ها

رویکرد بوت استرپ را می‌توان برای تعیین کمیت عدم قطعیت (یا خطای استاندارد) مرتبط با هر برآوردگر آماری مورد استفاده قرار داد.

برای مثال، ممکن است بخواهید دقت ضرایب بتای رگرسیون خطی را با استفاده از روش بوت استرپ تخمین بزنید.

مراحل مختلف به شرح زیر است:

1. یک تابع ساده به نام model\_coef() ایجاد کنید که مجموعه داده‌های سوئیسی و همچنین اندیس‌های مشاهدات را می‌گیرد و ضرایب رگرسیون را برمی گرداند.
2. ، تابع ()boot\_fun را به مجموعه داده کامل از 47 مشاهدات اعمال کنید.

ما با ایجاد تابعی شروع می‌کنیم که ضرایب مدل رگرسیون را برمی گرداند:

model\_coef <- **تابع (**داده، شاخص){

**coef (** **Im (**باروری **~** .، داده = داده، زیر مجموعه = شاخص))

1

**مدل\_ضریب (**سوئیسی، 1 **:** 47)

* # (عرض از مبدأ) آموزش امتحانی کشاورزی
* # 66.915 -0.172 -0.258 -0.871
* # نوزاد کاتولیک. مرگ و میر
* # 0.104 1.077

سپس، از تابع boot() [بسته راه‌اندازی] برای محاسبه خطاهای استاندارد 500 تخمین بوت استرپ برای ضرایب استفاده می‌کنیم:

**کتابخانه (**بوت)

**چکمه (**سوئیسی، مدل\_coef، 500)

* #
* # چکمه غیرپارامتری معمولی
* #
* #
* # زنگ زدن:
* # بوت (داده = سوئیس، آمار = مدل\_ضریب، R = 500)
* #
* #
* # آمار بوت استرپ:

|  |  |  |  |
| --- | --- | --- | --- |
| ## | اصلی | جانبداری | std. خطا |
| ## t1\* | 66.915 | 0.686934 | 12.2030 |
| ## t2\* | -0.172 | -0.001180 | 0.0680 |
| ## t3\* | -0.258 | -0.009005 | 0.2735 |
| ## t4\* | -0.871 | 0.009360 | 0.2269 |
| ## t5\* | 0.104 | 0.000603 | 0.0323 |
| ## t6\* | 1.077 | -0.028493 | 0.4560 |

در خروجی بالا،

* ستون اصلی مربوط به ضرایب رگرسیون است. خطاهای استاندارد مرتبط در ستون std.error آورده شده است.
* tl مربوط به عرض از مبدأ، t2 مربوط به کشاورزی و غیره است.

به‌عنوان مثال، مشاهده می‌شود که خطای استاندارد (SE) ضریب رگرسیون مرتبط با کشاورزی 06/0 است.

توجه داشته باشید که خطاهای استاندارد تغییرپذیری/دقت ضرایب بتا را اندازه‌گیری می‌کند. می‌توان از آن برای محاسبه فاصله اطمینان ضرایب استفاده کرد.

به‌عنوان مثال، فاصله اطمینان 95% برای یک ضریب معین b به صورت b +/- 2\*SE(b) تعریف می‌شود که در آن:

* حد پایین b = b - 2\*SE(b) = -0.172 - (2\*0.0680) = -0.308 (برای متغیر کشاورزی)
* حد بالایی b = b + 2\*SE(b) = -0.172 + (2\*0.0680) = -0.036 (برای ­متغیر کشاورزی)

یعنی تقریباً 95 درصد احتمال وجود دارد که بازه [308/0-، 036/0-] حاوی مقدار واقعی ضریب باشد.

استفاده از تابع lm() استاندارد خطاهای استاندارد کمی متفاوت می‌دهد، زیرا مدل خطی چند فرض را در مورد داده‌ها ایجاد می‌کند:

**خلاصه (** **Im (**باروری **~** .، داده = سوئیس)) coef **$**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ## | تخمین زدن | Std. خطای تی | ارزش | Pr(>|t|) |
| ## (عرض از مبدأ) | 66.915 | 10.7060 | 6.25 | 1.91e-07 |
| ## کشاورزی | -0.172 | 0.0703 | -2.45 | 1.87e-02 |
| ## معاینه | -0.258 | 0.2539 | -1.02 | 3.15e-01 |
| ## تحصیلات | -0.871 | 0.1830 | -4.76 | 2.43e-05 |
| ## کاتولیک | 0.104 | 0.0353 | 2.95 | 5.19e-03 |
| ## نوزاد. مرگ و میر | 1.077 | 0.3817 | 2.82 | 7.34e-03 |

رویکرد بوت استرپ به هیچ یک از این مفروضات ساخته شده توسط مدل خطی متکی نیست و بنابراین احتمالاً تخمین دقیق تری از خطاهای استاندارد ضرایب نسبت به تابع summary () ارائه می‌دهد.

14.7 بحث

این فصل روش نمونه‌گیری مجدد بوت استرپ را برای ارزیابی دقت مدل پیش‌بینی‌کننده و همچنین برای اندازه‌گیری عدم قطعیت مرتبط با یک تخمین‌گر آماری معین توصیف می‌کند.

یک رویکرد جایگزین برای بوت استرپ، برای ارزیابی عملکرد یک مدل پیش‌بینی‌کننده، تکنیک‌های اعتبارسنجی متقابل است (فصل 13).

**قسمت پنجم**

**انتخاب مدل**

مقدمه

هنگامی که شما متغیرهای پیش‌بینی‌کننده زیادی در یک مدل پیش‌بینی‌کننده دارید، روش‌های **انتخاب مدل** امکان انتخاب خودکار بهترین ترکیب از متغیرهای پیش‌بینی‌کننده را برای ساخت یک مدل پیش‌بینی بهینه ارائه می‌دهند.

حذف متغیرهای نامربوط منجر به یک مدل قابل تفسیرتر و ساده تر می‌شود. با کارایی یکسان، یک مدل ساده تر همیشه باید در ترجیح مدل‌های پیچیده تر استفاده شود.

علاوه بر این، استفاده از رویکردهای انتخاب مدل در برخی موقعیت‌ها که در آن مجموعه داده‌های چند متغیره بزرگ با متغیرهای پیش‌بینی‌کننده بسیاری دارید، حیاتی است. این اغلب در ناحیه ژنومی صدق می‌کند، جایی که یک چالش اساسی از این واقعیت ناشی می‌شود که تعداد متغیرهای ژنومی (p) معمولاً بسیار بیشتر از تعداد افراد (n) است (یعنی p >> n) (Bovelstad et al. .، 2007).

به خوبی شناخته شده است که، وقتی p >> n، یافتن پیش‌بینی‌کننده‌هایی که عملکرد عالی بر روی داده‌های برازش داشته باشند، آسان است، اما در اعتبارسنجی خارجی شکست می‌خورند و منجر به قوانین پیش‌بینی ضعیف می‌شوند. علاوه بر این، می‌تواند تنوع زیادی در برازش حداقل مربعات وجود داشته باشد که منجر به بیش برازش و در نتیجه پیش‌بینی‌های ضعیف در مشاهدات آینده می‌شود که در آموزش مدل استفاده نمی‌شوند (جیمز و همکاران، 2014).

یک استراتژی ممکن شامل آزمایش همه ترکیب ممکن از پیش‌بینی‌ها و سپس انتخاب بهترین مدل است. این روش به نام **رگرسیون بهترین زیرمجموعه‌ها (**فصل 16) از نظر ­محاسباتی گران است و برای یک مجموعه داده بزرگ با متغیرهای زیاد غیرممکن می‌شود.

جایگزین بهتری برای بهترین رگرسیون زیرمجموعه ها، استفاده از روش **رگرسیون گام به گام (**فصل 17) است که شامل افزودن و حذف پیش‌بینی کننده‌ها به منظور یافتن بهترین مدل با مجموعه‌ای از متغیرها کاهش یافته است.

روش‌های دیگر برای داده‌های با ابعاد بالا، حاوی متغیرهای پیش‌بینی‌کننده متعدد، شامل **رگرسیون جریمه‌شده (**رگرسیون ریج و لاسو، فصل 18) و روش‌ها**ی رگرسیون مبتنی بر اجزای اصلی (**PCR و PLS، فصل 19) است.

در این بخش، سه دسته مختلف از رویکردها را برای انتخاب یک مدل خطی بهینه برای داده‌های چند متغیره بزرگ پوشش خواهیم داد. این شامل:

* انتخاب بهترین زیر مجموعه‌ها (فصل 16)
* انتخاب گام به گام (فصل 17)
* رگرسیون مجازات شده (یا روش‌های انقباض) (فصل 18)
* روش‌های کاهش ابعاد (فصل 19)

بهترین رگرسیون زیر مجموعه ها

* 1. مقدمه

**رگرسیون بهترین زیرمجموعه یک** رویکرد انتخاب مدل است که شامل آزمایش همه ترکیب ممکن از متغیرهای پیش‌بینی‌کننده و سپس انتخاب بهترین مدل بر اساس برخی معیارهای آماری است.

در این فصل، نحوه محاسبه رگرسیون بهترین زیرمجموعه‌ها با استفاده از R را شرح خواهیم داد.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید
* جهش، برای محاسبه رگرسیون بهترین زیر مجموعه ها

**library (**tidyverse) **library (**caret) **library (**جهش)

* 1. نمونه‌ای از داده ها

ما از داده‌های داخلی R سوئیس، معرفی شده در فصل 2، برای پیش‌بینی امتیاز باروری بر اساس شاخص‌های اجتماعی-اقتصادی استفاده خواهیم کرد.

*# داده‌های داده را بارگیری کنید (*"سوئیس") *# نمونه داده‌ها را بازرسی کنید (* **سوئیسی،** 3)

* 1. محاسبه رگرسیون بهترین زیر مجموعه ها

تابع R regsubsets() [ leaps package] می‌تواند برای شناسایی بهترین مدل‌های مختلف با اندازه‌های مختلف استفاده شود.

شما باید گزینه nvmax را مشخص کنید که نشان دهنده حداکثر تعداد پیش‌بینی کننده‌ها برای گنجاندن در مدل است. به‌عنوان مثال، اگر nvmax = 5 باشد، تابع به بهترین مدل 5 متغیری برمی گردد، یعنی بهترین مدل متغیر f، بهترین مدل 2 متغیری، ...، بهترین مدل‌های 5 متغیری را برمی گرداند. .

در مثال ما فقط 5 متغیر پیش‌بینی در داده‌ها داریم. بنابراین، ما از nvmax = 5 استفاده خواهیم کرد.

مدل‌ها <- **زیر مجموعه‌ها (**باروری **-** .، داده = سوئیس، nvmax = 5) **خلاصه (**مدل‌ها)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ## شی انتخاب زیر مجموعه  ## تماس: regsubsets.formula(باروری - ., | | | | | داده = سوئیس، nvmax = 5) | | |
| ##  ##  ##  ##  ##  ##  ##  ##  ##  ## | 5 متغیر (و عرض از مبدأ) اجباری در خروج اجباری  کشاورزی FALSE FALSE  امتحان FALSE FALSE  آموزش و پرورش FALSE FALSE  کاتولیک FALSE FALSE  نوزاد. مرگ و میر FALSE FALSE  1 زیر مجموعه از هر اندازه تا 5 الگوریتم انتخاب: آزمون جامع کشاورزی آموزش کاتولیک | | | | | نوزاد. | مرگ و میر |
| ##  ##  ##  ##  ## | 1 ( 1 ) " "  2 ( 1 ) " "  3 ( 1 ) " "  4 ( 1 ) "\*"  5 ( 1 ) "\*" |  | ""  ""  ""  ""  "\*" | "\*"  "\*"  "\*"  "\*"  "\*" | ""  "\*"  "\*"  "\*"  "\*" | ""  ""  "\*"  "\*"  "\*" |  |

تابع summary() بهترین مجموعه متغیرها را برای هر اندازه مدل گزارش می‌کند. از خروجی بالا، یک ستاره مشخص می‌کند که یک متغیر معین در مدل مربوطه گنجانده شده است.

به‌عنوان مثال، مشاهده می‌شود که بهترین مدل 2 متغیری فقط شامل متغیرهای آموزش و کاتولیک ( باروری - آموزش + کاتولیک ) است. بهترین مدل سه متغیری ( باروری - تحصیلات + کاتولیک + مرگ و میر نوزاد ) و غیره است.

یک سوال طبیعی این است: در نهایت کدام یک از این بهترین مدل‌ها را برای تحلیل پیش‌بینی خود انتخاب کنیم؟

* 1. انتخاب مدل بهینه

برای پاسخ به این سوال، به معیارهای آماری یا استراتژی‌هایی نیاز دارید تا عملکرد کلی مدل‌ها را مقایسه کنید و بهترین را انتخاب کنید. شما باید خطای پیش‌بینی هر مدل را تخمین بزنید و مدلی را انتخاب کنید که خطای پیش‌بینی کمتری دارد.

* + 1. معیارهای انتخاب مدل: R2، Cp و BIC تنظیم شده

تابع summary() برخی از معیارها را برمی گرداند - R2، Cp و BIC تنظیم شده (به فصل 12 مراجعه کنید) - به ما امکان می‌دهد بهترین مدل کلی را شناسایی کنیم، جایی که best به‌عنوان مدلی تعریف می‌شود که R2 تنظیم شده را به حداکثر می‌رساند و خطای پیش‌بینی را به حداقل می‌رساند (RSS، cp و BIC).

R2 تعدیل شده نشان دهنده نسبت تغییرات در نتیجه است که با تغییر در مقادیر پیش‌بینی کننده توضیح داده می‌شود. هرچه R2 تنظیم شده بالاتر باشد، مدل بهتر است.

بهترین مدل، با توجه به هر یک از این معیارها، به صورت زیر قابل استخراج است: res.sum <- **summary (**models)

**data.frame (**

Adj.R2 = **which.max (**res.sum **$** adjr2)،

CP = **which.min (**res.sum **$** cp)، BIC = **which.min (**res.sum **$** bic)

)

## Adj.R2 CP BIC

## 1 544

هیچ راه حل صحیح واحدی برای انتخاب مدل وجود ندارد، هر یک از این معیارها به مدل‌های کمی متفاوت منجر می‌شوند. به یاد داشته باشید که "همه مدل‌ها اشتباه هستند، برخی از مدل‌ها مفید هستند" [[3]](#footnote-3).

در اینجا، R2 تنظیم شده به ما می‌گوید که بهترین مدل مدلی است که همه 5 متغیر پیش‌بینی کننده را داشته باشد. اما با استفاده از معیارهای BIC و Cp باید به سراغ مدلی با 4 متغیر برویم.

بنابراین، بسته به معیارهایی که در نظر می‌گیریم، «بهترین» مدل‌های مختلفی داریم. ما به استراتژی‌های اضافی نیاز داریم.

همچنین توجه داشته باشید که R2، BIC و Cp تنظیم شده بر روی داده‌های آموزشی که برای برازش مدل استفاده شده است، محاسبه می‌شوند. این بدان معناست که انتخاب مدل، با استفاده از این معیارها، احتمالاً در معرض بیش برازش است و ممکن است هنگام اعمال بر داده‌های جدید، عملکرد خوبی نداشته باشد.

یک رویکرد دقیق‌تر انتخاب مدل‌هایی بر اساس خطای پیش‌بینی محاسبه‌شده بر روی داده‌های آزمایشی جدید با استفاده از تکنیک‌های اعتبارسنجی متقاطع k-fold است (فصل 13).

* + 1. اعتبارسنجی متقابل K-fold

**اعتبارسنجی متقاطع k-fold شامل** ابتدا تقسیم داده‌ها به k زیر مجموعه است که به‌عنوان k-fold نیز شناخته می‌شود که در آن k معمولاً 5 یا 10 تنظیم می‌شود. (90%) به‌عنوان داده‌های آموزشی. میانگین خطای اعتبار متقابل به‌عنوان خطای پیش‌بینی مدل محاسبه می‌شود.

اعتبار متقاطع k-fold را می‌توان به راحتی با استفاده از تابع train() [ بسته caret ] (فصل 13) محاسبه کرد.

در اینجا، روش زیر را دنبال می‌کنیم:

1. فرمول‌های مختلف مدل را از شی مدل استخراج کنید
2. یک مدل خطی بر روی فرمول با استفاده از اعتبارسنجی متقاطع k برابر (با k=5) آموزش دهید و خطای پیش‌بینی هر مدل را محاسبه کنید.

ما با تعریف دو تابع کمکی شروع می‌کنیم:

1. get\_model\_formula() که امکان دسترسی آسان به فرمول مدل‌های برگردانده شده توسط تابع regsubsets () را فراهم می‌کند. کد زیر را در کنسول R خود کپی و پیست کنید:

* *شناسه: شناسه مدل*
* *آبجکت: شیء را زیر مجموعه‌ها بازگرداند*
* *داده: داده‌هایی که برای برازش زیر مجموعه‌های رگس استفاده می‌شوند*

get\_model\_formula <- **تابع (**id, object)!

* *داده‌های مدل‌ها را دریافت کنید*

مدل‌ها <- **خلاصه (**شیء) **$** which[id, **-** 1]

* *متغیر نتیجه را دریافت کنید*

فرم <- **as.formula (**اشیاء **$** call[[2]])

نتیجه <- **all.vars (**فرم)Cl]

* *پیش‌بینی‌های مدل را دریافت کنید*

پیش‌بینی‌کننده‌ها <- **نام‌ها (** **که (**مدل‌ها **==** درست)) پیش‌بینی‌کننده‌ها <- **چسباندن (**پیش‌بینی‌کننده‌ها، جمع‌کردن = "+")

* *فرمول مدل بسازید*

**as.formula (** **pasteO (**نتیجه، "~"، پیش‌بینی کننده ها)) }

به‌عنوان مثال برای داشتن بهترین فرمول مدل 3 متغیری، این را تایپ کنید:

**get\_model\_formula (**3، مدل)

## باروری - آموزش + کاتولیک + نوزاد. مرگ و میر

## محیط: 0x7fef642bd4d8>

2. get\_cv\_error()، برای دریافت خطای اعتبارسنجی متقابل (CV) برای یک مدل معین:

get\_cv\_error <- **تابع (**model.formula, data){

**set.seed (**l)

train.control <- **trainControl (**روش = "cv"، عدد = 5) cv <- **train (**model.formula، داده = داده، روش = "lm"، trControl = train.control)

cv **$** نتایج **$** RMSE }

در نهایت، از توابع کمکی تعریف شده در بالا برای محاسبه خطای پیش‌بینی بهترین مدل‌های مختلف که توسط تابع regsubsets () بازگردانده شده‌اند، استفاده کنید:

* *محاسبه خطای اعتبارسنجی متقابل* model.ids <- l **:** 5

cv.errors <- **map (**model.ids، get\_model\_formula، models) °/ o >°/o **نقشه (**get\_cv\_error, data = swiss) 0 / 0 > 0 / 0 **unlist (**)

cv.errors

* # Cl] 9.42 8.45 7.93 7.68 7.92
* *مدلی را انتخاب کنید که خطای CV را به حداقل برساند* **which.min (**cv.errors)
* # Cl] 4

مشاهده می‌شود که مدل با 4 متغیر بهترین مدل است. خطای پیش‌بینی کمتری دارد. ضرایب رگرسیون این مدل را می‌توان به صورت زیر استخراج کرد:

**ضریب (**مدل، 4)

* # (عرض از مبدأ) آموزش کشاورزی کاتولیک
* # 62.l0l -0.l55 -0.980 0.l25
* # نوزاد. مرگ و میر
* # l.078

16.6 بحث

این فصل بهترین روش رگرسیون زیرمجموعه را برای انتخاب بهترین ­مدل رگرسیون خطی توصیف می‌کند که داده‌های ما را توضیح می‌دهد.

توجه داشته باشید که این روش از نظر محاسباتی گران است و برای یک مجموعه داده بزرگ با متغیرهای زیاد غیرممکن می‌شود. جایگزین بهتری با روش **رگرسیون گام به گام ارائه می‌**شود. به فصل 17 مراجعه کنید.

فصل 17

رگرسیون گام به گام

* 1. مقدمه

**رگرسیون گام به گام (** یا انتخاب گام به گام) شامل افزودن و حذف مکرر پیش‌بینی کننده ها، در مدل پیش‌بینی، به منظور یافتن زیرمجموعه متغیرها در مجموعه داده است که منجر به بهترین مدل می‌شود، یعنی مدلی که خطای پیش‌بینی را کاهش می‌دهد.

سه استراتژی رگرسیون گام به گام وجود دارد (جیمز و همکاران، 2014، بروس و بروس (2017)):

1. **انتخاب رو به جلو که** بدون هیچ پیش‌بینی‌کننده‌ای در مدل شروع می‌شود، به‌طور مکرر مؤثرترین پیش‌بینی‌کننده‌ها را اضافه می‌کند و زمانی متوقف می‌شود که بهبود دیگر از نظر آماری معنی‌دار نباشد.
2. **انتخاب معکوس (**یا **حذف به عقب** ) که با همه پیش‌بینی‌کننده‌های مدل (مدل کامل) شروع می‌شود، به‌طور مکرر کمترین پیش‌بینی‌کننده‌ها را حذف می‌کند و زمانی که مدلی دارید که همه پیش‌بینی‌کننده‌ها از نظر آماری معنی‌دار هستند، متوقف می‌شود.
3. **انتخاب گام به گام (**یا جایگزینی متوالی) که ترکیبی از انتخاب‌های جلو و عقب است. شما بدون هیچ پیش‌بینی‌کننده شروع می‌کنید، سپس به‌طور متوالی ­پیش‌بینی‌کننده‌های مؤثر را اضافه می‌کنید (مانند انتخاب رو به جلو). پس از اضافه کردن هر متغیر جدید، هر متغیری را که دیگر بهبودی در برازش مدل ایجاد نمی‌کند (مانند انتخاب به عقب) حذف کنید.

توجه داشته باشید که،

* انتخاب رو به جلو و انتخاب گام به گام را می‌توان در پیکربندی ابعاد بالا به کار برد، جایی که تعداد نمونه n کمتر از تعداد پیش‌بینی کننده‌های p است، مانند زمینه‌های ژنومی.
* انتخاب معکوس مستلزم آن است که تعداد نمونه‌های n بزرگتر از تعداد متغیرهای p باشد تا بتوان مدل کامل را برازش کرد.

در این فصل، نحوه محاسبه روش‌های رگرسیون گام به گام در R را خواهید آموخت.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید
* جهش، برای محاسبه رگرسیون گام به گام

**library (**tidyverse) **library (**caret) **library (**جهش)

* 1. محاسبه رگرسیون گام به گام

توابع و بسته‌های R زیادی برای محاسبه رگرسیون گام به گام وجود دارد. این شامل:

• stepAIC() [بسته MASS] که بهترین مدل را توسط AIC انتخاب می‌کند. گزینه‌ای به نام جهت دارد که می‌تواند مقادیر زیر را داشته باشد: i) "both" (برای رگرسیون گام به گام، هر دو انتخاب به جلو و عقب). "به عقب" (برای انتخاب به عقب) و "به جلو" (برای انتخاب به جلو). بهترین مدل نهایی را برمی گرداند.

**کتابخانه (**MASS)

* *متناسب با مدل کامل*

full.model <- **lm (**باروری **-** .، داده = سوئیس)

* *مدل رگرسیون گام به گام*

step.model <- **stepAIC (**full.model، جهت = "هر دو"، trace = FALSE)

**خلاصه (**step.model)

* regsubsets() [leapspack] که دارای پارامتر تنظیم nvmax است که حداکثر ­تعداد پیش‌بینی‌کننده‌ها را برای گنجاندن در مدل مشخص می‌کند (به فصل 16 مراجعه کنید). چندین مدل را با اندازه‌های مختلف تا nvmax برمی گرداند. برای انتخاب بهترین، باید عملکرد مدل‌های مختلف را با هم مقایسه کنید. ­regsubsets () دارای روش گزینه است که می‌تواند مقادیر "backward"، "forward" و "seqrep" را بگیرد (seqrep = جایگزینی متوالی، ترکیبی از انتخاب‌های جلو و عقب).

مدل‌ها <- **زیر مجموعه‌ها (**باروری **-** .، داده = سوئیس، nvmax = 5، روش = "seqrep")

**خلاصه (**مدل ها)

توجه داشته باشید که تابع train() [بسته caret] گردش کار آسانی را برای انجام انتخاب‌های گام به گام با استفاده از جهش‌ها و بسته‌های MASS فراهم می‌کند. گزینه‌ای به نام متد دارد که می‌تواند مقادیر زیر را بگیرد:

* "leapBackward"، برای برازش رگرسیون خطی با **انتخاب رو به عقب**
* "leapForward"، برای برازش رگرسیون خطی با **انتخاب رو به جلو**
* "leapSeq"، برای برازش رگرسیون خطی با **انتخاب گام به گام** .

شما همچنین باید پارامتر تنظیم nvmax را مشخص کنید که مربوط به حداکثر تعداد پیش‌بینی کننده‌هایی است که باید در مدل گنجانده شود.

به‌عنوان مثال، می‌توانید nvmax را از 1 تا 5 تغییر دهید. در این حالت، عملکرد با جستجو در بهترین مدل‌های مختلف با اندازه‌های مختلف شروع می‌شود تا بهترین مدل 5 متغیری. یعنی بهترین مدل 1 متغیری، بهترین مدل 2 متغیری، ...، بهترین مدل‌های 5 متغیری را جستجو می‌کند.

مثال زیر انتخاب معکوس ( روش = "جهش به عقب" ) را با استفاده از مجموعه داده‌های سوئیسی انجام می‌دهد تا بهترین مدل برای پیش‌بینی باروری بر اساس شاخص‌های اجتماعی-اقتصادی شناسایی شود.

از آنجایی که مجموعه داده فقط شامل 5 پیش‌بینی کننده است، nvmax را از 1 تا 5 تغییر می‌دهیم و در نتیجه ­5 مدل برتر را با اندازه‌های مختلف شناسایی می‌کنیم: بهترین مدل 1 متغیری، بهترین مدل 2 متغیری، ... بهترین مدل 5 متغیری

ما از اعتبارسنجی متقاطع 10 برابری برای تخمین میانگین خطای پیش‌بینی (RMSE) هر یک از 5 مدل استفاده خواهیم کرد (به فصل 13 مراجعه کنید). معیار آماری RMSE برای مقایسه 5 مدل و انتخاب خودکار بهترین مدل استفاده می‌شود، جایی که بهترین به‌عنوان مدلی تعریف می‌شود که RMSE را به حداقل می‌رساند.

* *تنظیم دانه برای تکرارپذیری* **set.seed (**123)
* *تنظیم مکرر k-fold-validation* train.control <- **trainControl (**روش = "cv"، عدد = 10) *# آموزش مدل* step.model <- **train (**باروری **~** .، داده = سوئیسی، روش = "جهش به عقب" , tuneGrid = **data.frame (**nvmax = 1 **:** 5), trControl = train.control ) step.model **$** نتایج

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ## | nvmax RMSE | Rsquared | MAE | RMSESD | RsquaredSD | MAESD |
| ## 1 | 1 9.30 | 0.408 | 7.91 | 1.53 | 0.390 | 1.65 |
| ## 2 | 2 9.08 | 0.515 | 7.75 | 1.66 | 0.247 | 1.40 |
| ## 3 | 3 8.07 | 0.659 | 6.55 | 1.84 | 0.216 | 1.57 |
| ## 4 | 4 7.27 | 0.732 | 5.93 | 2.14 | 0.236 | 1.67 |
| ## 5 | 5 7.38 | 0.751 | 6.03 | 2.23 | 0.239 | 1.64 |

خروجی بالا معیارهای مختلف و انحراف معیار آنها را برای مقایسه دقت ­5 مدل برتر نشان می‌دهد. ستون‌ها عبارتند از:

* nvmax : تعداد متغیرهای مدل. به‌عنوان مثال nvmax = 2، بهترین مدل 2 متغیری را مشخص کنید
* RMSE و MAE دو معیار مختلف هستند که خطای پیش‌بینی هر مدل را اندازه‌گیری می‌کنند. هر چه RMSE و MAE کمتر باشد، مدل بهتر است.
* Rsquared همبستگی بین مقادیر نتیجه مشاهده شده و مقادیر پیش‌بینی شده توسط مدل را نشان می‌دهد. هر چه مجذور R بیشتر باشد، مدل بهتر است.

در مثال ما، می‌توان دید که مدلی با 4 متغیر (nvmax = 4) مدلی است که کمترین RMSE را دارد. شما می‌توانید بهترین مقادیر تنظیم (nvmax) را که به‌طور خودکار توسط تابع train() انتخاب می‌شود، به صورت زیر نمایش دهید:

step.model **$** bestTune ## nvmax

## 4 4

این نشان می‌دهد که بهترین مدل مدلی است که nvmax = 4 متغیر دارد. تابع summary() بهترین مجموعه متغیرها را برای هر اندازه مدل، تا بهترین مدل 4 متغیری گزارش می‌کند.

**خلاصه (**step.model **$** finalModel)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ## شی انتخاب زیر مجموعه | | | | | | | | | |
| ## 5 متغیر | | | | | (و عرض از مبدأ) | | | | |
| ## |  | | | | اجباری در اجباری خارج شد | | | | |
| ## | کشاورزی | | | | نادرست | | | نادرست | |
| ## | معاینه | | | | نادرست | | | نادرست | |
| ## | تحصیلات | | | | نادرست | | | نادرست | |
| ## | کاتولیک | | | | نادرست | | | نادرست | |
| ## | نوزاد. مرگ و میر نادرست | | | | | | | نادرست | |
| ## | 1 | زیر مجموعه‌های هر اندازه | | | | | تا 4 | |  | |  |  |
| ## | الگوریتم انتخاب: | | | | | رو به عقب | | |  | |  |  |
| ## |  |  |  | کشاورزی | | معاینه | | | تحصیلات | | کاتولیک | نوزاد. مرگ و میر |
| ## | 1 | ( | 1 | ) "" | | " | " | | "\*" | | "" | "" |
| ## | 2 | ( | 1 | ) "" | | " | " | | "\*" | | "\*" | "" |
| ## | 3 | ( | 1 | ) "" | | " | " | | "\*" | | "\*" | "\*" |
| ## | 4 | ( | 1 | ) "\*" | | " | " | | "\*" | | "\*" | "\*" |

یک ستاره مشخص می‌کند که یک متغیر معین در مدل مربوطه گنجانده شده است. به‌عنوان مثال، می‌توان مشاهده کرد که بهترین مدل 4 متغیری شامل کشاورزی، آموزش، کاتولیک، مرگ و میر نوزادان ( باروری *~* کشاورزی + آموزش + کاتولیک + مرگ و میر نوزادان ) است.

ضرایب رگرسیون مدل نهایی (id = 4) به صورت زیر قابل دسترسی است:

**coef (**step.model **$** finalModel, 4)

یا با محاسبه مدل خطی فقط با استفاده از پیش‌بینی کننده‌های انتخاب شده:

**Im (**باروری **~** کشاورزی **+** آموزش **+** کاتولیک **+** نوزاد. مرگ و میر، داده = سوئیس)

##

* # زنگ زدن:
* # lm(فرمول = باروری - کشاورزی + آموزش + کاتولیک +
* # نوزاد. مرگ و میر، داده = سوئیس)
* #
* # ضرایب:
* # (عرض از مبدأ) آموزش کشاورزی کاتولیک
* # 62.101 -0.155 -0.980 0.125
* # نوزاد. مرگ و میر
* # 1.078

17.4 بحث

این فصل روش‌های رگرسیون گام به گام را به منظور انتخاب یک مدل ساده بهینه، بدون به خطر انداختن دقت مدل، شرح می‌دهد.

ما نحوه استفاده از بسته جهشی R را برای محاسبه رگرسیون گام به گام نشان داده ایم. جایگزین دیگر تابع stepAIC() موجود در بسته MASS است. گزینه‌ای به نام جهت دارد که می‌تواند مقادیر زیر را داشته باشد: "Both"، "Forward"، "Backward".

**کتابخانه (**MASS)

res.lm <- **Im (**باروری **~** .، داده = سوئیس)

step <- **stepAIC (**res.lm، جهت = "هر دو"، trace = FALSE) گام

علاوه بر این، بسته caret روشی برای محاسبه رگرسیون گام به گام با استفاده از بسته MASS دارد ( روش = "lmStepAIC" ):

*# آموزش مدل*

step.model <- **قطار (**باروری **~** .، داده = سوئیس،

روش = "lmStepAIC"،

trControl = train.control،

ردیابی = نادرست)

* *دقت مدل*

step.model **$** نتایج

* *ضرایب مدل نهایی*

step.model **$** finalModel

* *خلاصه‌ای از مدل*

**خلاصه (**step.model **$** finalModel)

رگرسیون گام به گام برای داده‌های با ابعاد بالا حاوی ­متغیرهای پیش‌بینی کننده چندگانه بسیار مفید است. جایگزین‌های دیگر عبارتند از رگرسیون جریمه شده (رگرسیون ریج و لاسو) (فصل 18) و روش‌های رگرسیون مبتنی بر اجزای اصلی (PCR و PLS) (فصل 19).

فصل 18

رگرسیون جریمه شده: ریج، لاسو و شبکه الاستیک

* 1. مقدمه

مدل خطی استاندارد (یا روش حداقل مربعات معمولی) در شرایطی که شما یک مجموعه داده چند متغیره بزرگ حاوی تعدادی متغیر برتر از تعداد نمونه دارید، عملکرد ضعیفی دارد.

یک جایگزین بهتر، **رگرسیون جریمه‌شده است** که اجازه می‌دهد یک مدل رگرسیون خطی ایجاد کند که به دلیل داشتن متغیرهای زیاد در مدل، با افزودن یک محدودیت در معادله جریمه می‌شود (جیمز و همکاران، 2014، بروس و بروس (2017)). این روش به روش‌های **انقباض** یا **منظم‌سازی** نیز معروف است.

پیامد اعمال این جریمه، کاهش (یعنی کوچک کردن) مقادیر ضرایب به سمت صفر است. این اجازه می‌دهد تا متغیرهای کمتر کمک کننده ضریب نزدیک به صفر یا مساوی صفر داشته باشند.

توجه داشته باشید که انقباض نیاز به انتخاب یک پارامتر تنظیم (لامبدا) دارد که میزان انقباض را تعیین می‌کند.

در این فصل ما متداول‌ترین روش‌های رگرسیون جریمه‌شده، از جمله **رگرسیون ریج،** رگرسیون **لاسو** و رگرسیون **شبکه الاستیک را شرح خواهیم داد** . ما همچنین مثال‌های عملی را در R ارائه خواهیم داد.

* 1. روش‌های انقباض
     1. رگرسیون ریج

رگرسیون ریج ضرایب رگرسیون را کوچک می‌کند، به‌طوری که متغیرها با سهم جزئی در نتیجه، ضرایب آنها نزدیک به صفر هستند.

انقباض ضرایب با جریمه کردن مدل رگرسیون با یک جمله جریمه به نام **L2-norm** که مجموع ضرایب مجذور است به دست می‌آید.

مقدار جریمه را می‌توان با استفاده از ثابتی به نام لامبدا () تنظیم کرد. انتخاب یک مقدار خوب برای لامبدا بسیار مهم است.

زمانی که لامبدابرابر با صفر باشد، عبارت جریمه تاثیری ندارد و رگرسیون خطی ضرایب حداقل مربعات کلاسیک را تولید می‌کند. با این حال، با افزایش لامبدابه بی نهایت، تاثیر جریمه انقباض افزایش می‌یابد و ضرایب رگرسیون ریج به صفر نزدیک می‌شود.

توجه داشته باشید که بر خلاف رگرسیون حداقل مربعات معمولی، رگرسیون ریج به شدت تحت تأثیر مقیاس پیش‌بینی کننده‌ها قرار می‌گیرد. بنابراین، بهتر است پیش‌بینی‌کننده‌ها را قبل از اعمال رگرسیون ریج (James et al., 20J4) استاندارد (یعنی مقیاس) کنید تا همه پیش‌بینی‌کننده‌ها در یک مقیاس باشند.

استانداردسازی یک پیش‌بین x را می‌توان با استفاده از فرمول x 1 = x / sd(x) به دست آورد که در آن sd(x) انحراف استاندارد x است. نتیجه این امر این است که همه پیش‌بینی‌کننده‌های استاندارد دارای انحراف استاندارد یک خواهند بود که اجازه می‌دهد برازش نهایی به مقیاسی که پیش‌بینی‌کننده‌ها در آن اندازه‌گیری می‌شوند، بستگی نداشته باشد.

یکی از مزیت‌های مهم رگرسیون ریج این است که در مقایسه با روش حداقل مربعات معمولی (فصل 3)، در شرایطی که داده‌های چند متغیره بزرگ با تعداد پیش‌بینی‌کننده‌ها (p) بزرگتر از تعداد مشاهدات (n). باشد همچنان خوب عمل میکند

یکی از معایب رگرسیون ریج این است که بر خلاف روش‌های رگرسیون گام به گام (فصل 17) که معمولاً مدل‌هایی را انتخاب می‌کنند که شامل مجموعه‌ای از متغیرهای کاهش‌یافته هستند، همه پیش‌بینی‌کننده‌های مدل نهایی را شامل می‌شود.

رگرسیون ریج ضرایب را به سمت صفر کوچک می‌کند، اما هیچ یک از آنها را دقیقاً صفر نمی‌کند. رگرسیون لاسو جایگزینی است که بر این اشکال غلبه می‌کند.

* + 1. رگرسیون لاسو

Lasso مخفف Least Absolute Shrinkage and Selection Operator یعنی «اپراتور حداقل انقباض و انتخاب» است. ضرایب رگرسیون را با جریمه کردن مدل رگرسیون با یک جمله جریمه به نام **Ll-norm** که مجموع ضرایب مطلق است، به سمت صفر کوچک می‌کند.

در مورد رگرسیون لاسو، جریمه باعث می‌شود برخی از تخمین‌های ضرایب، با کمک جزئی به مدل، دقیقاً برابر با صفر باشند. این بدان معنی است که لاسو می‌تواند به‌عنوان جایگزینی برای روش‌های انتخاب زیر مجموعه برای انجام انتخاب متغیر به منظور کاهش پیچیدگی مدل دیده شود.

مانند در رگرسیون ریج، انتخاب یک مقدار خوب از لامبدابرای لاسو بسیار مهم است.

یکی از مزیت‌های آشکار رگرسیون لاسو نسبت به رگرسیون ریج این است که مدل‌های ساده‌تر و قابل تفسیرتری را تولید می‌کند که تنها مجموعه کاهش‌یافته‌ای از پیش‌بینی‌کننده‌ها را در بر می‌گیرد. با این حال، نه رگرسیون ریج و نه لاسو به‌طور کامل بر دیگری تسلط ندارند.

به‌طور کلی، لاسو ممکن است در شرایطی که برخی از پیش‌بینی‌کننده‌ها ضرایب بزرگی دارند و سایر پیش‌بینی‌کننده‌ها ضرایب بسیار کوچکی دارند، بهتر عمل کند.

رگرسیون ریج زمانی بهتر عمل می‌کند که نتیجه تابعی از بسیاری از پیش‌بینی کننده‌ها باشد، همه با ضرایب تقریباً برابر (جیمز و همکاران، 2014).

روش‌های اعتبارسنجی متقابل را می‌توان برای شناسایی اینکه کدام یک از این دو تکنیک در یک مجموعه داده خاص بهتر است استفاده کرد.

* + 1. شبکه الاستیک

Elastic Net یک مدل رگرسیونی تولید می‌کند که هم با **Ll-norm** و هم L2-norm جریمه می‌**شود** . نتیجه این امر کوچک کردن مؤثر ضرایب (مانند رگرسیون ریج) و صفر کردن برخی از ضرایب (مانند LASSO) است.

* 1. بارگیری بسته‌های R مورد نیاز
* آیه مرتب برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید
* glmnet، برای محاسبه رگرسیون جریمه شده

**کتابخانه (**tidyverse)

**کتابخانه (**کارت)

**کتابخانه (**glmnet)

* 1. آماده‌سازی داده ها

ما از مجموعه داده‌های بوستون [در بسته MASS ]، معرفی شده در فصل 2، برای پیش‌بینی مقدار خانه متوسط ( mdev )، در حومه بوستون، بر اساس متغیرهای پیش‌بینی کننده چندگانه استفاده خواهیم کرد.

ما داده‌ها را به صورت تصادفی به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم می‌کنیم. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *داده‌ها را بارگذاری کنید*

**داده (**"بوستون"، بسته = "MASS")

* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**123)

training.samples <- Boston$medv °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE)

train.data <- Bostonftraining.samples، ]

test.data <- Bostonf **-** training.samples, ]

* 1. محاسبه رگرسیون خطی جریمه شده
     1. آماده‌سازی داده‌های اضافی

شما باید دو شی بسازید:

* y برای ذخیره متغیر نتیجه
* x برای نگهداری متغیرهای پیش‌بینی کننده. این باید با استفاده از تابع model.matrix () ایجاد شود که به‌طور خودکار هر متغیر کیفی (در صورت وجود) را به متغیرهای ساختگی تبدیل می‌کند (فصل 5) که مهم است زیرا glmnet() فقط می‌تواند ورودی‌های عددی و کمی را دریافت کند. پس از ایجاد ماتریس مدل، مولفه intercept را در index = 1 حذف می‌کنیم.
* *متغیرهای پیش‌بینی کننده*

x <- **model.matrix (**medv **~** ., train.data)[، **-** 1]

* *متغیر نتیجه*

y <- train.data **$** medv

18.5.2 توابع R

ما از تابع R glmnet() [بسته glmnet] برای محاسبه مدل‌های رگرسیون خطی جریمه شده استفاده می‌کنیم.

فرمت ساده شده به شرح زیر است:

**glmnet (**x، y، آلفا = 1، لامبدا = NULL)

* x : ماتریس متغیرهای پیش‌بینی کننده
* y : متغیر پاسخ یا نتیجه که یک متغیر باینری است.
* آلفا : پارامتر اختلاط شبکه الاستیک. مقادیر مجاز عبارتند از:
* «1»: برای رگرسیون لاسو
* "0": برای رگرسیون ریج
* مقداری بین 0 و 1 (مثلا 0.3) برای رگرسیون شبکه الاستیک.

• lamba : یک مقدار عددی که میزان انقباض را تعیین می‌کند. باید توسط تحلیلگر مشخص شود.

در رگرسیون جریمه شده، باید یک لامبدا ثابت را برای تنظیم میزان انقباض ضریب مشخص کنید. بهترین لامبدا برای داده‌های شما، می‌تواند به‌عنوان لامبدا تعریف شود که میزان خطای پیش‌بینی اعتبار متقاطع را به حداقل می‌رساند. این را می‌توان به‌طور خودکار با استفاده از تابع ()cv.glmnet تعیین کرد.

در بخش‌های بعدی، با محاسبه مدل‌های رگرسیون ریج، لاسو و رگرسیون شبکه الاستیک شروع می‌کنیم. در مرحله بعد، مدل‌های مختلف را با هم مقایسه می‌کنیم تا بهترین را برای داده‌های خود انتخاب کنیم.

بهترین مدل به‌عنوان مدلی تعریف می‌شود که کمترین خطای پیش‌بینی را دارد، RMSE (فصل 12).

18.5.3 محاسبه رگرسیون ریج

* *بهترین لامبدا را با استفاده از cross-validation* **set.seed (**123) بیابید

cv <- **cv.glmnet (**x، y، آلفا = 0)

* *بهترین مقدار لامبدا را نمایش دهید*

cv **$** lambda.min

* # [1] 0.758
* *مدل نهایی را بر روی داده‌های آموزشی قرار دهید*

مدل <- **glmnet (**x، y، آلفا = 0، lambda = cv **$** lambda.min)

* *نمایش ضرایب رگرسیون*

**ضریب (**مدل)

* # 14 x 1 ماتریس پراکنده از کلاس "dgCMatrix"

|  |  |
| --- | --- |
| ## | s0 |
| ## (عرض از مبدأ) | 28.69633 |
| ## جنایت | -0.07285 |
| ## zn | 0.03417 |
| ## صنعت | -0.05745 |
| ## چس | 2.49123 |
| ## نوکس | -11.09232 |
| ## دور | 3.98132 |
| ## سن | -0.00314 |
| ## دیس | -1.19296 |
| ## راد | 0.14068 |
| ## مالیات | -0.00610 |
| ## ptratio | -0.86400 |

* # مشکی 0.00937
* # lstat -0.47914
* *روی داده‌های آزمون پیش‌بینی کنید*

x.test <- **model.matrix (**medv **~** ., test.data)[, **-** 1]

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**x.test) °/ o >°/o **as.vectorO**

* *معیارهای عملکرد مدل*

**data.frame (**

RMSE = **RMSE (**پیش‌بینی‌ها، test.data **$** medv)، Rsquare = **R2 (**پیش‌بینی‌ها، test.data **$** medv)

)

* # RMSE Rsquare
* # 1 4.98 0.671

توجه داشته باشید که به‌طور پیش فرض، تابع glmnet () متغیرها را استاندارد می‌کند تا مقیاس آنها قابل مقایسه باشد. با این حال، ضرایب همیشه در مقیاس اصلی برگردانده می‌شوند.

* + 1. محاسبه رگرسیون لاسو

تنها تفاوت بین کد R مورد استفاده برای رگرسیون ریج این است که برای رگرسیون لاسو باید به جای آلفا = 0 آرگومان alpha = 1 را مشخص کنید (برای رگرسیون ریج).

* *بهترین لامبدا را با استفاده از اعتبارسنجی متقاطع پیدا کنید*

**set.seed (**123)

cv <- **cv.glmnet (**x، y، آلفا = 1)

* *بهترین مقدار لامبدا را نمایش دهید*

cv **$** lambda.min

* # [1] 0.00852
* *مدل نهایی را بر روی داده‌های آموزشی قرار دهید*

مدل <- **glmnet (**x، y، آلفا = 1، lambda = cv **$** lambda.min)

* *ضرایب رگرسیون Dsiplay*

**ضریب (**مدل)

* # 14 x 1 ماتریس پراکنده از کلاس "dgCMatrix"

|  |  |
| --- | --- |
| ## | s0 |
| ## (عرض از مبدأ) | 36.90539 |
| ## جنایت | -0.09222 |
| ## zn | 0.04842 |
| ## صنعت | -0.00841 |
| ## چس | 2.28624 |
| ## نوکس | -16.79651 |
| ## دور | 3.81186 |
| ## سن | . |
| ## دیس | -1.59603 |
| ## راد | 0.28546 |
| ## مالیات | -0.01240 |
| ## ptratio | -0.95041 |
| ## مشکی | 0.00965 |
| ## lstat | -0.52880 |

* *ایجاد، پیش‌بینی در مورد داده‌های آزمون*

x.test <- **model.matrix (**medv **~** ., test.data)[, **-** 1]

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**x.test) °/ o >°/o **as.vectorO**

* *معیارهای عملکرد مدل*

**data.frame (**

RMSE = **RMSE (**پیش‌بینی‌ها، test.data **$** medv)،

Rsquare = **R2 (**پیش‌بینی‌ها، test.data **$** medv)

)

* # RMSE Rsquare
* # 1 4.99 0.671
  + 1. محاسبه رگرسیون شبکه الاستیک

رگرسیون شبکه الاستیک را می‌توان به راحتی با استفاده از گردش کار caret محاسبه کرد که بسته glmnet را فراخوانی می‌کند .

ما از caret برای انتخاب خودکار بهترین پارامترهای تنظیم آلفا و لامبدا استفاده می‌کنیم. بسته‌های caret طیفی از مقادیر آلفا و لامبدا ممکن را آزمایش می‌کنند، سپس بهترین مقادیر را برای لامبدا و آلفا انتخاب می‌کنند و به مدل نهایی می‌رسند که یک مدل شبکه الاستیک است.

در اینجا، ترکیب 10 مقدار مختلف را برای آلفا و لامبدا آزمایش خواهیم کرد. این با استفاده از گزینه tuneLength مشخص می‌شود .

بهترین مقادیر آلفا و لامبدا مقادیری هستند که خطای اعتبارسنجی متقاطع را به حداقل می‌رسانند (فصل 13).

* *با استفاده از مجموعه آموزشی مدل را بسازید*

**set.seed (**123)

مدل <- **قطار (**

medv **~** .، داده = train.data، روش = "glmnet"،

trControl = **trainControl (**"cv"، عدد = 10)،

طول کوک = 10

)

* *بهترین پارامتر تنظیم*

مدل **$** bestTune

* #آلفا لامبدا
* # 6 0.1 0.21
* *ضریب مدل نهایی. تو نیاز داری*
* *برای تعیین بهترین لامبدا*

**coef (**مدل **$** finalModel، مدل **$** bestTune **$** lambda)

* # 14 x 1 ماتریس پراکنده از کلاس "dgCMatrix"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ## | | | 1 | |
| ## (عرض از مبدأ) | | | 33.04083 | |
| ## جنایت | | | -0.07898 | |
| ## zn | | | 0.04136 | |
| ## صنعت | | | -0.03093 | |
| ## چس | | | 2.34443 | |
| ## نوکس | | | -14.30442 | |
| ## دور  ## سن | | | 3.90863. | |
| ## | دیس | -1. | | .41783 |
| ## | راد | 0. | | .20564 |
| ## | مالیات | -0. | | 00879 |
| ## | ptratio | -0. | | .91214 |
| ## | سیاه | 0. | | 00946 |
| ## | lstat | -0. | | .51770 |

* *ایجاد، پیش‌بینی در مورد داده‌های آزمون*

x.test <- **model.matrix (**medv **~** ., test.data)[, **-** 1] predictions <- model °/ o >°/o **predict (**x.test)

* *معیارهای عملکرد مدل*

**data.frame (**

RMSE = **RMSE (**پیش‌بینی‌ها، test.data **$** medv)، Rsquare = **R2 (**پیش‌بینی‌ها، test.data **$** medv)

)

* # RMSE Rsquare
* # 1 4.98 0.672
  + 1. مقایسه مدل‌های مختلف

معیارهای عملکرد مدل‌های مختلف قابل مقایسه هستند. با استفاده از لاسو یا رگرسیون شبکه الاستیک ، ضریب متغیر پیش‌بینی‌کننده سن را صفر می‌کنیم که منجر به مدل ساده‌تری در مقایسه با رگرسیون ریج می‌شود که شامل همه متغیرهای پیش‌بینی‌کننده است.

همه چیز برابر است، ما باید به سراغ مدل ساده تر برویم. در مثال ما، می‌توانیم مدل‌های لاسو یا رگرسیون شبکه الاستیک را انتخاب کنیم.

توجه داشته باشید که ما به راحتی می‌توانیم رگرسیون ریج، شبکه الاستیک و لاسو را با استفاده از گردش کار Caret محاسبه و مقایسه کنیم.

caret به‌طور خودکار بهترین مقادیر پارامتر تنظیم را انتخاب می‌کند، مدل نهایی را محاسبه می‌کند و عملکرد مدل را با استفاده از تکنیک‌های اعتبارسنجی متقابل ارزیابی می‌کند.

* + 1. استفاده از پکیج کارت

0. یک محدوده شبکه‌ای از مقادیر لامبدا را تنظیم کنید :

lambda <- 10 **~ seq (** **-** 3، 3، طول = 100)

1. محاسبه رگرسیون ریج :

* *مدل را بسازید*

**set.seed (**123)

خط الراس <- **قطار (**

medv **~** .، داده = train.data، روش = "glmnet"، trControl = **trainControl (**"cv"، عدد = 10)، tuneGrid = **expand.grid (**alpha = 0، lambda = lambda) )

* *ضرایب مدل*

**coef (**رج **$** مدل نهایی، رج **$** bestTune **$** lambda)

* *پیش‌بینی کنید*

پیش‌بینی‌ها <- رج 0 / 0 > 0 / 0 **پیش‌بینی (**test.data)

* **data.frame** *عملکرد پیش‌بینی مدل (*

RMSE = **RMSE (**پیش‌بینی‌ها، test.data **$** medv)، Rsquare = **R2 (**پیش‌بینی‌ها، test.data **$** medv)

1. محاسبه رگرسیون لاسو :

*# ساخت مدل* **set.seed (**123) lasso <- **train (** medv **~** ., data = train.data, method = "glmnet"، trControl = **trainControl (**"cv"، عدد = 10)، tuneGrid = **expand.grid (** آلفا = 1، لامبدا = لامبدا))

* *ضرایب مدل*

**coef (**lasso **$** finalModel, lasso **$** bestTune **$** lambda)

* *پیش‌بینی کنید*

پیش‌بینی‌ها <- lasso °/ o >°/o **پیش‌بینی (**test.data)

* **data.frame** *عملکرد پیش‌بینی مدل (*

RMSE = **RMSE (**پیش‌بینی‌ها، test.data **$** medv)، Rsquare = **R2 (**پیش‌بینی‌ها، test.data **$** medv)

1. رگرسیون شبکه الاستیک :

* *ساخت مدل* **set.seed (**123) الاستیک <- **train (** medv **~** .، داده = train.data، روش = "glmnet"، trControl = **trainControl (**"cv"، عدد = 10)، tuneLength = 10 )
* *ضرایب مدل*

**ضریب (**الاستیک **$** مدل نهایی، الاستیک **$** bestTune **$** lambda)

* *پیش‌بینی کنید*

پیش‌بینی <- elastic 0 / 0 > 0 / 0 **پیش‌بینی (**test.data)

* **data.frame** *عملکرد پیش‌بینی مدل (*

RMSE = **RMSE (**پیش‌بینی‌ها، test.data **$** medv)، Rsquare = **R2 (**پیش‌بینی‌ها، test.data **$** medv)

1. مقایسه عملکرد مدل‌ها :

عملکرد مدل‌های مختلف رگرسیون ریج، شبکه الاستیک و لاسو را می‌توان به راحتی با استفاده از کارت مقایسه کرد. بهترین مدل به‌عنوان مدلی تعریف می‌شود که خطای پیش‌بینی را به حداقل می‌رساند.

مدل‌ها <- **فهرست (**رج = رج، لاسو = لاسو، الاستیک = الاستیک) **نمونه‌های مجدد (**مدل‌ها) **°/ o >°/o خلاصه (** متریک = "RMSE") ##

* # زنگ زدن:
* # summary.resamples(object = ., Metric = "RMSE")

##

* # مدل ها: رج، لاسو، الاستیک
* # تعداد نمونه مجدد: 10

##

* # RMSE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ## | حداقل | 1 ق. | میانگین میانه 3rd Qu. | حداکثر | NA |
| ## | خط الراس 3.10 | 3.96 | 4.38 4.73 5.52 | 7.43 | 0 |
| ## | لاسو 3.16 | 4.03 | 4.39 4.73 5.51 | 7.27 | 0 |
| ## | الاستیک 3.13 | 4.00 | 4.37 4.72 5.52 | 7.32 | 0 |

مشاهده می‌شود که مدل شبکه الاستیک دارای کمترین میانه RMSE است.

18.6 بحث

در این فصل ما متداول‌ترین روش‌های رگرسیون جریمه‌شده، از جمله رگرسیون ریج، رگرسیون لاسو و رگرسیون شبکه الاستیک را شرح دادیم. این روش‌ها در شرایطی که مجموعه داده‌های چند متغیره بزرگی دارید بسیار مفید هستند.

فصل 19

مؤلفه اصلی و رگرسیون حداقل مربعات جزئی

* 1. مقدمه

این فصل روش‌های رگرسیون مبتنی بر تکنیک‌های کاهش ابعاد را ارائه می‌کند که می‌تواند زمانی بسیار مفید باشد که مجموعه داده‌ای بزرگ با متغیرهای پیش‌بینی‌کننده همبسته متعدد داشته باشید.

به‌طور کلی، همه روش‌های کاهش ابعاد ابتدا پیش‌بینی‌کننده‌های اصلی را در چند متغیر جدید به نام مؤلفه‌های اصلی (PC) خلاصه می‌کنند که سپس به‌عنوان پیش‌بینی‌کننده برای برازش مدل رگرسیون خطی استفاده می‌شوند. این روش‌ها از چند خطی شدن بین پیش‌بینی‌کننده‌ها که در تنظیم رگرسیون مشکل بزرگی است، پرهیز می‌کنند (به فصل 9 مراجعه کنید).

هنگام استفاده از روش‌های کاهش ابعاد، معمولاً توصیه می‌شود که هر پیش‌بینی‌کننده را استاندارد کنید تا قابل مقایسه باشد. استانداردسازی شامل تقسیم پیش‌بینی کننده بر انحراف معیار آن است.

در اینجا، ما دو روش رگرسیون معروف بر اساس کاهش ابعاد را شرح دادیم: **رگرسیون مؤلفه اصلی (** **PCR** ) و رگرسیون **حداقل مربعات جزئی (** **PLS** ). ما همچنین نمونه‌های عملی را در R ارائه می‌دهیم.

* 1. رگرسیون مؤلفه اصلی

رگرسیون مؤلفه اصلی (PCR) ابتدا تحلیل مؤلفه اصلی را اعمال می‌کند[[4]](#footnote-4) در مجموعه داده‌ها برای خلاصه کردن متغیرهای پیش‌بینی‌کننده اصلی به چند متغیر جدید که به‌عنوان اجزای اصلی (PC) نیز شناخته می‌شوند که ترکیبی خطی از داده‌های اصلی هستند.

سپس از این مؤلفه‌های اصلی برای ساخت مدل رگرسیون خطی استفاده می‌شود. تعداد مؤلفه‌های اصلی ­برای گنجاندن در مدل، با اعتبارسنجی متقابل (cv) انتخاب می‌شود. توجه داشته باشید که PCR زمانی مناسب است که مجموعه داده حاوی پیش‌بینی کننده‌های بسیار همبسته باشد.

* 1. رگرسیون حداقل مربعات جزئی

یک اشکال احتمالی PCR این است که ما هیچ تضمینی نداریم که اجزای اصلی انتخاب شده با نتیجه مرتبط باشند. در اینجا، انتخاب اجزای اصلی برای گنجاندن در مدل توسط متغیر نتیجه نظارت نمی‌شود.

جایگزینی برای PCR، رگرسیون **حداقل مربعات جزئی (**PLS) است که مؤلفه‌های اصلی جدیدی را شناسایی می‌کند که نه تنها پیش‌بینی‌کننده‌های اصلی را خلاصه می‌کنند، بلکه با نتیجه مرتبط هستند. سپس از این مولفه‌ها برای برازش مدل رگرسیون استفاده می‌شود. بنابراین، در مقایسه با PCR، PLS از یک استراتژی کاهش ابعاد استفاده می‌کند که توسط نتیجه نظارت می‌شود.

مانند PCR، PLS برای داده‌هایی با پیش‌بینی‌کننده‌های همبستگی بسیار مناسب است. تعداد مؤلفه‌های اصلی مورد استفاده در PLS عموماً با اعتبارسنجی متقابل انتخاب می‌شود. پیش‌بینی کننده‌ها و متغیرهای نتیجه باید به‌طور کلی استاندارد شوند تا متغیرها قابل مقایسه باشند.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید
* pls، برای محاسبه PCR و PLS

**کتابخانه (**tidyverse)

**کتابخانه (**کارت)

**کتابخانه (**pls)

* 1. آماده‌سازی داده ها

ما از مجموعه داده‌های بوستون [در بسته MASS ]، معرفی شده در فصل 2، برای پیش‌بینی مقدار خانه متوسط ( mdev )، در حومه بوستون، بر اساس متغیرهای پیش‌بینی کننده چندگانه استفاده خواهیم کرد.

ما داده‌ها را به صورت تصادفی به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم می‌کنیم. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *داده‌ها را بارگذاری کنید*

**داده (**"بوستون"، بسته = "MASS")

* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**123)

training.samples <- Boston$medv °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE)

train.data <- Bostonftraining.samples، ]

test.data <- Bostonf **-** training.samples, ]

19.6 محاسبات

R function train() [ caret package] یک گردش کار آسان برای محاسبه PCR و PLS با فراخوانی بسته pls فراهم می‌کند. گزینه‌ای به نام متد دارد که می‌تواند مقدار pcr یا pls را بگیرد.

یک آرگومان اضافی scale = TRUE برای استانداردسازی متغیرها به منظور قابل مقایسه کردن آنهاست ­.

caret از اعتبارسنجی متقاطع برای شناسایی خودکار تعداد بهینه اجزای اصلی ( ncomp ) برای گنجاندن در مدل استفاده می‌کند.

در اینجا، 10 مقدار مختلف پارامتر تنظیم ncomp را آزمایش خواهیم کرد. این با استفاده از گزینه tuneLength مشخص می‌شود . تعداد بهینه مولفه‌های اصلی به گونه‌ای انتخاب می‌شود که خطای اعتبارسنجی متقابل (RMSE) به حداقل برسد.

* + 1. محاسبه رگرسیون مؤلفه اصلی
* *ساخت مدل بر روی مجموعه آموزشی* **set.seed (**123)

مدل <- **قطار (**

medv **~** .، داده = train.data، روش = "pcr"،

مقیاس = درست،

trControl = **TrainControl (**"cv"، عدد = 10)، طول تنظیم = 10)

* *مدل نمودار RMSE در مقابل مقادیر مختلف* **نمودار اجزا (**مدل)
* *بهترین پارامتر تنظیم ncomp را چاپ کنید*
* *خطای اعتبارسنجی متقابل،* مدل RMSE **$** bestTune را به حداقل برسانید
* # ncomp
* شماره 5 5
* **خلاصه** *مدل نهایی (*مدل **$** finalModel)
* # داده: ابعاد X: 407 13
* بعد # Y: 407 1
* # روش مناسب: svdpc
* # تعداد اجزای در نظر گرفته شده: 5
* # آموزش: درصد واریانس توضیح داده شد
* # 1 comps 2 comps 3 comps 4 comps 5 comps
* # X 47.48 58.40 68.00 74.75 80.94
* # .نتیجه 38.10 51.02 64.43 65.24 71.17
* *پیش‌بینی کنید*

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**test.data)

* *معیارهای عملکرد مدل*

**data.frame (**

RMSE = caret **::RMSE (**پیش‌بینی‌ها، test.data **$** medv)، Rsquare = caret **::R2 (**پیش‌بینی‌ها، test.data **$** medv)

)

* # RMSE Rsquare
* # 1 5.18 0.645

#اجزاء

نمودار خطای پیش‌بینی (RMSE، فصل 12) ساخته شده توسط مدل را با توجه به تعداد اجزای اصلی گنجانده شده در مدل نشان می‌دهد.

تحلیل ما نشان می‌دهد که، انتخاب پنج جزء اصلی (ncomp = 5) کوچکترین خطای پیش‌بینی RMSE را می‌دهد.

تابع summary() همچنین درصد واریانس توضیح داده شده در پیش‌بینی کننده‌ها (x) و در نتیجه ( medv ) را با استفاده از تعداد مولفه‌های مختلف ارائه می‌دهد.

به‌عنوان مثال، 80.94٪ از تغییرات (یا اطلاعات) موجود در پیش‌بینی کننده‌ها توسط 5 مؤلفه اصلی ( ncomp = 5 ) گرفته شده است. علاوه بر این، تنظیم ncomp = 5، 71٪ از اطلاعات را در متغیر نتیجه ( medv ) دریافت می‌کند که خوب است.

در مجموع، اعتبارسنجی متقابل ncomp = 5 را به‌عنوان تعداد بهینه مؤلفه‌های اصلی که خطای پیش‌بینی (RMSE) را به حداقل می‌رسانند شناسایی می‌کند و تغییرات کافی در پیش‌بینی‌کننده‌ها و نتیجه را توضیح می‌دهد.

19.6.2 محاسبه حداقل مربعات جزئی

کد R دقیقاً مانند روش PCR است.

* *ساخت مدل بر روی مجموعه آموزشی* **set.seed (**123)

مدل <- **قطار (**

medv **~** .، داده = train.data، روش = "pls"، مقیاس = TRUE، trControl = **trainControl (**"cv"، عدد = 10)، طول = 10 )

* *مدل نمودار RMSE در مقابل مقادیر مختلف* **نمودار اجزا (**مدل)
* *بهترین پارامتر تنظیم ncomp را چاپ کنید*
* *خطای اعتبارسنجی متقابل،* مدل RMSE **$** bestTune را به حداقل برسانید
* # ncomp
* # 9 9
* **خلاصه** *مدل نهایی (*مدل **$** finalModel)

## داده: ابعاد X: 407 13

*##* بعد Y: 407 1

## روش برازش: oscorespls

## تعداد اجزای در نظر گرفته شده: 9

## آموزش: درصد واریانس توضیح داده شد

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ## | 1 کامپوز | 2 کامپوز | 3 کامپوزیت | 4 کامپوزیت | 5 کامپوزیت | 6 کامپوزیت | 7 کامپوزیت |
| ## ایکس | 46.19 | 57.32 | 64.15 | 69.76 | 75.63 | 78.66 | 82.85 |
| ## .نتیجه | 50.90 | 71.84 | 73.71 | 74.71 | 75.18 | 75.35 | 75.42 |

## 8 comps 9 comps

|  |  |  |
| --- | --- | --- |
| ## ایکس | 85.92 | 90.36 |
| ## .نتیجه | 75.48 | 75.49 |

* *بسازید، پیش‌بینی کنید*

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**test.data)

* *معیارهای عملکرد مدل*

**data.frame (**

RMSE = caret **::RMSE (**پیش‌بینی‌ها، test.data **$** medv)،

Rsquare = caret **::R2 (**پیش‌بینی‌ها، test.data **$** medv)

* # RMSE Rsquare
* # 1 4.99 0.671

#اجزاء

تعداد بهینه مولفه‌های اصلی موجود در مدل PLS 9 است. این 90 درصد از تغییرات در پیش‌بینی‌کننده‌ها و 75 درصد از تغییرات در متغیر نتیجه ( medv ) را نشان می‌دهد.

در مثال ما، خطای اعتبارسنجی متقاطع RMSE به دست آمده با مدل PLS کمتر از RMSE به دست آمده با استفاده از روش PCR است. بنابراین، مدل PLS بهترین مدل برای توضیح داده‌های ما در مقایسه با مدل PCR است.

19.7 بحث

این فصل روش‌های رگرسیون مبتنی بر مؤلفه‌های اصلی، از جمله رگرسیون مؤلفه اصلی (PCR) و رگرسیون حداقل مربعات جزئی (PLS) را توصیف می‌کند. این روش‌ها برای داده‌های چند متغیره حاوی پیش‌بینی‌های همبسته بسیار مفید هستند.

وجود همبستگی در داده‌ها اجازه می‌دهد تا داده‌ها را در چند مؤلفه غیر زائد خلاصه کنیم که می‌توانند در مدل رگرسیون استفاده شوند.

در مقایسه با رگرسیون ریج و لاسو (فصل 18)، تفسیر مدل‌های PCR و PLS نهایی دشوارتر است، زیرا آنها هیچ نوع انتخاب متغیری را انجام نمی‌دهند یا حتی به‌طور مستقیم تخمین ضریب رگرسیون را تولید نمی‌کنند.

قسمت ششم

طبقه‌بندی

فصل 20

مقدمه

قبلاً مدل رگرسیون را توضیح دادیم (فصل 2) که برای پیش‌بینی یک متغیر پیامد کمی یا پیوسته بر اساس یک یا چند متغیر پیش‌بینی‌کننده استفاده می‌شود.

در **طبقه‌بندی،** متغیر نتیجه کیفی (یا مقوله‌ای) است. **طبقه‌بندی** به مجموعه‌ای از روش‌های یادگیری ماشین برای پیش‌بینی کلاس (یا دسته) افراد بر اساس یک یا چند متغیر پیش‌بینی کننده اشاره دارد.

در این قسمت به موضوعات زیر می‌پردازیم:

* رگرسیون لجستیک، برای وظایف طبقه‌بندی باینری (فصل 21)
* رگرسیون لجستیک گام به گام و جریمه شده برای انتخاب متغیرها (فصل 22 و 23)
* مفروضات و تشخیص رگرسیون لجستیک (فصل 24)
* رگرسیون لجستیک چند جمله ای، توسعه رگرسیون لجستیک برای ­وظایف طبقه‌بندی چند طبقه (فصل 25).
* تحلیل متمایز، برای مسائل طبقه‌بندی باینری و چند کلاسه (فصل 26)
* طبقه‌بندی کننده بیز ساده (فصل 27)
* ماشین‌های بردار پشتیبان (فصل 28)
* ارزیابی مدل طبقه‌بندی (فصل 29)

بیشتر الگوریتم‌های طبقه‌بندی، احتمال تعلق به یک کلاس معین را محاسبه می‌کنند. سپس مشاهدات به کلاسی که بالاترین امتیاز احتمال را دارد اختصاص داده می‌شود.

به‌طور کلی، شما باید یک برش (مرز) احتمال را تعیین کنید که بالاتر از آن، مشاهده را متعلق به یک کلاس معین در نظر بگیرید.

* 1. نمونه‌هایی از مجموعه داده ها
     1. مجموعه داده PimaIndiansDiabetes2

مجموعه داده Pima Indian Diabetes در بسته mlbench موجود است. برای طبقه‌بندی باینری استفاده خواهد شد.

* *مجموعه داده را بارگیری کنید*

**داده (**"PimaIndiansDiabetes2"، بسته = "mlbench")

* *داده‌ها را بررسی کنید*

**سر (**Pima!ndiansDiabetes2, 4)

* فشار گلوکز باردار توده انسولین سه سر سر شجره سن دیابت
* # 1 6 148 72 35 NA 33.6 0.627 50 pos

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ## | 2 | 1 | 85 | 66 | 29 | NA | 26.6 | 0.351 | 31 | نفی |
| ## | 3 | 8 | 183 | 64 | NA | NA | 23.3 | 0.672 | 32 | pos |
| ## | 4 | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | نفی |

داده‌ها شامل 768 فرد (زن) و 9 متغیر بالینی برای پیش‌بینی احتمال مثبت یا منفی بودن افراد به دیابت است:

* باردار: تعداد دفعات بارداری
* گلوکز: غلظت گلوکز پلاسما
* فشار: فشار خون دیاستولیک (میلی متر جیوه)
* عضله سه سر: ضخامت چین‌های پوستی سه سر بازو (میلی متر)
* انسولین: انسولین سرم 2 ساعته (mu U/ml)
* جرم: شاخص توده بدن (وزن بر حسب کیلوگرم/(قد بر حسب متر) ~ 2)
* شجره: دیابت عملکرد شجره نامه
* سن: سن (سال)
* دیابت: متغیر کلاس

20.1.2 مجموعه داده زنبق

مجموعه داده زنبق برای کارهای طبقه‌بندی چند کلاسه استفاده خواهد شد. این شامل طول و عرض کاسبرگ و گلبرگ برای سه گونه زنبق است. ما می‌خواهیم گونه‌ها را بر اساس پارامترهای کاسبرگ و گلبرگ پیش‌بینی کنیم.

* *بارگیری داده‌های* **داده (**"زنبق")
* *داده‌ها را بررسی کنید*

**سر** فیریس، 4)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ## | کاسبرگ. | .کاسبرگ طول | | .گلبرگ پهن. | | .طول گلبرگ | | .گونه‌های پهن | | |
| ## | 1 | 5. | 1 | 3. | .5 | 1. | .4 | 0. | .2 | ستوزا |
| ## | 2 | 4. | 9 | 3. | .0 | 1. | .4 | 0. | .2 | ستوزا |
| ## | 3 | 4. | 7 | 3. | .2 | 1. | .3 | 0. | .2 | ستوزا |
| ## | 4 | 4. | 6 | 3. | . 1 | 1. | .5 | 0. | .2 | ستوزا |

فصل 21

رگرسیون لجستیک

* 1. مقدمه

**رگرسیون لجستیک** برای پیش‌بینی کلاس (یا دسته) افراد بر اساس یک یا چند متغیر پیش‌بینی کننده (x) استفاده می‌شود. از آن برای مدل‌سازی یک نتیجه باینری استفاده می‌شود، یعنی یک متغیر که می‌تواند تنها دو مقدار ممکن داشته باشد: 0 یا 1، بله یا خیر، بیمار یا غیر بیمار.

رگرسیون لجستیک متعلق به خانواده‌ای به نام *مدل خطی تعمیم یافته (* *GLM* ) است که برای گسترش مدل رگرسیون خطی (فصل 3) به موقعیت‌های دیگر توسعه یافته است. مترادف‌های دیگر *رگرسیون لجستیک باینری،* رگرسیون *لجستیک دو جمله‌ای* و *مدل لاجیت* هستند.

رگرسیون لجستیک مستقیماً کلاس مشاهدات را بر نمی‌گرداند. این به ما امکان می‌دهد تا احتمال (p) عضویت در کلاس را تخمین بزنیم. احتمال بین 0 و 1 خواهد بود. شما باید آستانه احتمالی را تعیین کنید که در آن دسته از یکی به دیگری تغییر می‌کند. به‌طور پیش‌فرض، این مقدار p=0.5 تنظیم شده است، اما در واقع باید بر اساس هدف تحلیل تسویه شود.

در این فصل شما یاد خواهید گرفت که چگونه:

* معادله رگرسیون لجستیک و عبارات کلیدی مانند log-odds و logit را تعریف کنید
* رگرسیون لجستیک را در **R انجام دهید** و نتایج را تفسیر کنید
* پیش‌بینی داده‌های آزمایشی جدید و ارزیابی دقت مدل
  1. عملکرد لجستیک

تابع رگرسیون لجستیک استاندارد، برای پیش‌بینی نتیجه یک مشاهده با توجه به متغیر پیش‌بینی‌کننده (x)، یک منحنی s شکل است که به صورت p = exp(y) / [1 + exp(y)] تعریف می‌شود (جیمز و همکاران، 2014). این را می‌توان به سادگی به صورت p = 1/[1 + exp(-y)] نوشت که: bO و bl ضرایب بتای رگرسیونی هستند. bl مثبت نشان می‌دهد که افزایش x با افزایش p همراه خواهد بود . برعکس، bl منفی نشان می‌دهد که افزایش x با کاهش p همراه خواهد بود .

کمیت log[p/(lp)] لگاریتم فرد نامیده می‌شود که به نام **log-odd** یا **logit** نیز شناخته می‌شود .

**شانس منعکس** کننده احتمال وقوع رویداد است. می‌توان آن را به‌عنوان نسبت "موفقیت ها" به "عدم موفقیت ها" در نظر گرفت. از نظر فنی، شانس عبارت است از احتمال وقوع یک رویداد تقسیم بر احتمال عدم وقوع آن (بروس و بروس، 2017). به‌عنوان مثال، اگر احتمال مثبت بودن دیابت 0.5 باشد، احتمال "نمی شود" 1-0.5 = 0.5 است و احتمال آن 1.0 است.

توجه داشته باشید که احتمال را می‌توان از شانس p = Odds/(l + Odds) محاسبه کرد.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید

**کتابخانه (**Tidyverse)

**کتابخانه (**کارت)

**مجموعه\_تم (** **theme\_bw (**))

* 1. آماده‌سازی داده ها

متغیرهای پیش‌بینی کننده پیوسته و/یا طبقه‌بندی هستند.­

انجام مراحل زیر ممکن است دقت مدل شما را بهبود بخشد

* موارد پرت بالقوه را حذف کنید
* اطمینان حاصل کنید که متغیرهای پیش‌بینی به‌طور نرمال توزیع شده اند. اگر نه، می‌توانید از تبدیل log، root، box-cox استفاده کنید.
* برای به حداقل رساندن بیش برازش، پیش‌بینی کننده‌های بسیار مرتبط را حذف کنید. وجود ­پیش‌بینی‌کننده‌های بسیار مرتبط ممکن است به یک راه‌حل مدل ناپایدار منجر شود.

در اینجا، از PimaIndiansDiabetes2 [در بسته mlbench ]، معرفی شده در فصل 20، برای پیش‌بینی احتمال مثبت بودن دیابت بر اساس متغیرهای بالینی متعدد استفاده خواهیم کرد.

ما داده‌ها را به صورت تصادفی به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم می‌کنیم. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *داده‌ها را بارگذاری کنید و NA‌ها را حذف کنید*

**داده (**"PimaIndiansDiabetes2"، بسته = "mlbench")

PimaIndiansDiabetes2 <- **na.omit (**PimaIndiansDiabetes2)

* *داده‌ها را بررسی کنید*

**sample\_n (**Pima!ndiansDiabetes2, 3)

* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**l23)

training.samples <- PimaIndiansDiabetes2$diabetes °/ o >°/o

**createDataPartition (**p = O.8، لیست = FALSE)

train.data <- PimaIndiansDiabetes2[training.samples, ]

test.data <- PimaIndiansDiabetes2[ **-** training.samples, ]

* 1. محاسبه رگرسیون لجستیک

تابع R glm() برای مدل خطی تعمیم یافته می‌تواند برای محاسبه رگرسیون لجستیک استفاده شود. شما باید گزینه خانواده = دوجمله‌ای را مشخص کنید که به R می‌گوید که می‌خواهیم رگرسیون لجستیک را متناسب کنیم.

* + 1. کد R شروع سریع
* *برازش مدل* مدل <- **glm (** دیابت **~** .، داده = train.data، خانواده = دو جمله ای) *# مدل را خلاصه کنید*

**خلاصه (**مدل)

* *پیش‌بینی کنید*

probabilities <- model °/ o >°/o **predict (**test.data, type = "response") predicted.classes <- **ifelse (**احتمالات **>** 0.5، "pos"، "neg")

* *دقت مدل*

**میانگین (**predicted.classes **==** test.data **$** دیابت)

* + 1. رگرسیون لجستیک ساده

از رگرسیون لجستیک ساده برای پیش‌بینی احتمال عضویت در کلاس بر اساس یک متغیر پیش‌بینی‌کننده استفاده می‌شود.

کد R زیر مدلی را برای پیش‌بینی احتمال دیابت مثبت بر اساس غلظت گلوکز پلاسما ایجاد می‌کند:

مدل <- **glm (** دیابت **~** گلوکز، داده = train.data، خانواده = دو جمله ای) **خلاصه (**مدل) ضریب **$**

* # Estimate Std. خطای z مقدار Pr(>|z|)
* # (Intercept) -6.3267 0.7241 -8.74 2.39e-18
* # glucose 0.0437 0.0054 8.09 6.01e-16

خروجی بالا تخمین ضرایب بتای رگرسیون و سطوح معنی‌داری آنها را نشان می‌دهد. فاصله ( b0 ) 32/6- و ضریب متغیر گلوکز 043/0 است.

معادله لجستیک را می‌توان به صورت p = exp(-6.32 + 0.043\*گلوکز)/ [1 + exp(-6.32 + 0.043\*گلوکز)] نوشت . با استفاده از این فرمول، برای هر مقدار جدید غلظت گلوکز پلاسمایی، می‌توانید احتمال ابتلای افراد به دیابت را پیش‌بینی کنید.

با استفاده از تابع ()predict می‌توان پیش‌بینی‌ها را به راحتی انجام داد . از نوع گزینه = "پاسخ" برای به دست آوردن مستقیم احتمالات استفاده کنید

newdata <- **data.frame (**گلوکز = **c (**20، 180))

احتمالات <- مدل 0 / 0 > 0 / 0 **predict (**newdata، type = "response") predicted.classes <- **ifelse (**احتمالات **>** 0.5، "pos"، "neg") predicted.classes

تابع لجستیک یک منحنی احتمال s شکل را ارائه می‌دهد که به صورت زیر نشان داده شده است:

train.data °/ o >°/o

**جهش (**prob = **ifelse (**دیابت **==** "pos"، 1، 0)) °/ o >°/o

**ggplot (** **aes (**گلوکز، prob)) **+**

**geom\_point (**آلفا = 0.2) **+**

**geom\_smooth (**روش = "glm"، method.args = **لیست (**خانواده = "دو جمله ای")) **+ آزمایشگاه‌ها (**

عنوان = "مدل رگرسیون لجستیک"، x = "غلظت گلوکز پلاسما"، y = "احتمال ابتلا به دیابت")

مدل رگرسیون لجستیک

* + 1. رگرسیون لجستیک چندگانه

رگرسیون لجستیک چندگانه برای پیش‌بینی احتمال عضویت در کلاس بر اساس متغیرهای پیش‌بینی‌کننده چندگانه، به شرح زیر استفاده می‌شود:

مدل <- **glm (**دیابت **~** گلوکز **+** توده **+** باردار، داده = قطار.داده، خانواده = دوجمله ای)

**خلاصه (**مدل) **$** coef

در اینجا، ما می‌خواهیم تمام متغیرهای پیش‌بینی موجود در مجموعه داده را شامل شود. این کار با استفاده از ~ انجام می‌شود. :

مدل <- **glm (** دیابت **~** .، داده = train.data، خانواده = دوجمله ای) **خلاصه (**مدل) **$** coef

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ## | تخمین زدن | Std. | خطا | مقدار z | Pr(>|z|) |
| ## (عرض از مبدأ) | -9.50372 | 1. | .31719 | -7.215 | 5.39e-13 |
| ##باردار | 0.04571 | 0. | 06218 | 0.735 | 4.62e-01 |
| ## گلوکز | 0.04230 | 0. | 00657 | 6.439 | 1.20e-10 |
| ## فشار | -0.00700 | 0. | 01291 | -0.542 | 5.87e-01 |
| ## سه سر بازو | 0.01858 | 0. | .01861 | 0.998 | 3.18e-01 |
| ## انسولین | -0.00159 | 0. | 00139 | -1.144 | 2.52e-01 |

## mass 0.04502

*##* pedigree 0.96845

## age 0.04256

0.02887 1.559 1.19e-01

0.46020 2.104 3.53e-02

0.02158 1.972 4.86e-02

از خروجی بالا، جدول ضرایب تخمین ضرایب بتا و سطوح اهمیت آنها را نشان می‌دهد. ستون‌ها عبارتند از:

* تخمین : تخمین‌های عرض از مبداء (b0) و ضریب بتا مربوط به هر ­متغیر پیش‌بینی کننده
* Std.Error : خطای استاندارد تخمین ضریب. این نشان دهنده دقت ضرایب است. هر چه خطای استاندارد بزرگتر باشد، اطمینان کمتری نسبت به برآورد داریم.
* مقدار z : آماره z که تخمین ضریب (ستون 2) تقسیم بر ­خطای استاندارد برآورد (ستون 3) است.
* Pr(>|z|) : مقدار p مربوط به آمار z. هرچه مقدار p کوچکتر باشد، تخمین مهمتر است.

توجه داشته باشید که از توابع coef () و summaryO می‌توان فقط برای استخراج ضرایب به شرح زیر استفاده کرد:

**ضریب (**مدل)

**خلاصه (**مدل ) **$** coef

* 1. تفسیر

مشاهده می‌شود که تنها 5 مورد از 8 پیش‌بینی کننده به‌طور قابل‌توجهی با نتیجه مرتبط هستند. این موارد عبارتند از: باردار، گلوکز، فشار، توده و شجره نامه.

برآورد ضریب متغیر گلوکز 045/0 = b است که مثبت است. این بدان معنی است که افزایش گلوکز با افزایش احتمال ابتلا به دیابت مرتبط است. اما ضریب فشار متغیر b = -0.007 است که منفی است. این بدان معنی است که افزایش فشار خون با کاهش احتمال ابتلا به دیابت همراه خواهد بود.

یک مفهوم مهم برای درک، برای تفسیر ضرایب بتا لجستیک، **نسبت شانس است** . نسبت شانس ارتباط بین متغیر پیش‌بینی کننده (x) و متغیر نتیجه (y) را اندازه‌گیری می‌کند. نسبت شانس وقوع یک رویداد ( رویداد = 1 ) را با توجه به حضور پیش‌بینی کننده x ( x = 1 )، در مقایسه با شانس وقوع رویداد در غیاب آن پیش‌بینی کننده ( x = 0 ) نشان می‌دهد.

برای یک پیش‌بینی‌کننده معین (مثلا xl)، ضریب بتای مرتبط (bl) در تابع رگرسیون لجستیک با لاگ نسبت شانس آن پیش‌بینی‌کننده مطابقت دارد.

اگر نسبت شانس 2 باشد، آنگاه شانس وقوع رویداد ( رویداد = 1 ) دو برابر بیشتر است زمانی که پیش‌بینی کننده x وجود داشته باشد ( x = 1 ) در مقابل x وجود ندارد ( x = 0 ).

به‌عنوان مثال، ضریب رگرسیون برای گلوکز 0.042 است. این نشان می‌دهد که یک واحد افزایش در غلظت گلوکز، شانس ابتلا به دیابت مثبت را با exp(0.042) 1.04 برابر افزایش می‌دهد.

از نتایج رگرسیون لجستیک می‌توان متوجه شد که برخی از متغیرها - عضله سه سر، انسولین و سن - از نظر آماری معنی‌دار نیستند. نگه داشتن آنها در مدل ممکن است به بیش برازش کمک کند. بنابراین، آنها باید حذف شوند. این را می‌توان به‌طور خودکار با استفاده از تکنیک‌های آماری، از جمله روش‌های **رگرسیون گام به گام** و رگرسیون **جریمه** انجام داد. این روش‌ها در قسمت بعدی توضیح داده شده است. به‌طور خلاصه، آنها شامل انتخاب یک مدل بهینه با مجموعه‌ای از متغیرها کاهش یافته، بدون به خطر انداختن دقت مدل هستند.

در اینجا، از آنجایی که ما تعداد کمی پیش‌بینی داریم (n = 9)، می‌توانیم به صورت دستی مهم ترین را انتخاب کنیم:

مدل <- **glm (** دیابت **~** باردار **+** گلوکز **+** فشار **+** جرم **+** شجره نامه، داده = قطار. داده، خانواده = دوجمله ای)

* 1. پیش‌بینی

برای ارزیابی عملکرد مدل رگرسیون لجستیک خود، پیش‌بینی‌هایی را با استفاده از داده‌های آزمون انجام خواهیم داد.

روال به شرح زیر است:

1. احتمال عضویت در کلاس مشاهدات را بر اساس متغیرهای پیش‌بینی کننده پیش‌بینی کنید
2. مشاهدات را به کلاسی با بالاترین امتیاز احتمال (یعنی بالای 0.5) اختصاص دهید.

تابع R predictO می‌تواند برای پیش‌بینی احتمال مثبت بودن دیابت با توجه به مقادیر پیش‌بینی کننده استفاده شود.

**احتمال** ابتلا به دیابت مثبت را پیش‌بینی کنید:

احتمالات <- مدل °/ o >°/o **پیش‌بینی (**test.data, type = "response") **head (**احتمالات)

* # 21 25 28 29 32 36
* # 0.3914 0.6706 0.0501 0.5735 0.6444 0.1494

این احتمالات به کدام کلاس‌ها اشاره دارد؟ در مثال ما، خروجی احتمال مثبت بودن تست دیابت است. می‌دانیم که این مقادیر به جای منفی بودن، با احتمال مثبت بودن تست مطابقت دارند، زیرا تابع ()contrasts نشان می‌دهد که R یک متغیر ساختگی با 1 برای "pos" و "0" برای neg ایجاد کرده است. احتمالات همیشه به کلاس ساختگی با کد "1" اشاره می‌کنند.

کدگذاری ساختگی را بررسی کنید:

(test.data **$ دیابت** )

* # پست
* # منفی 0
* # پوز 1

پیش‌بینی طبقه افراد :

کد R زیر افراد را بر اساس توانایی‌های احتمالی پیش‌بینی‌شده (p) مثبت بودن دیابت به دو گروه دسته‌بندی می‌کند. افراد با p بالای 0.5 (حدس تصادفی) به‌عنوان دیابت مثبت در نظر گرفته می‌شوند.

predicted.classes <- **ifelse (**احتمالات **>** 0.5، "pos"، "neg") **head (**predicted.classes)

* # 21 25 28 29 32 36
* # "neg" "pos" "neg" "pos" "pos" "neg"
  1. ارزیابی دقت مدل

دقت مدل به‌عنوان نسبت مشاهداتی که به درستی طبقه‌بندی شده اند اندازه‌گیری می‌شود. برعکس، خطای طبقه‌بندی به‌عنوان نسبت مشاهداتی که به اشتباه طبقه‌بندی شده اند تعریف می‌شود.

نسبت مشاهدات طبقه‌بندی شده صحیح:

**میانگین (**پیش‌بینی شده. کلاس‌ها، تست. داده **$** دیابت)

## [1] NA

صحت پیش‌بینی طبقه‌بندی حدود 77 درصد است که خوب است. میزان خطای طبقه‌بندی اشتباه 23 درصد است.

توجه داشته باشید که چندین معیار برای ارزیابی عملکرد یک مدل طبقه‌بندی وجود دارد (فصل 29).

* 1. بحث

در این فصل، نحوه عملکرد رگرسیون لجستیک را شرح داده ایم و کدهای R را برای محاسبه رگرسیون لجستیک ارائه کرده ایم. علاوه بر این، نحوه پیش‌بینی و ارزیابی دقت مدل را نشان دادیم. تفسیر خروجی مدل رگرسیون لجستیک در مقایسه با سایر روش‌های طبقه‌بندی بسیار آسان است. علاوه بر این، به دلیل سادگی آن، نسبت به روش‌های انعطاف‌پذیر مانند درخت‌های تصمیم‌گیری، کمتر مستعد بیش‌برازش است.

توجه داشته باشید که بسیاری از مفاهیم رگرسیون خطی برای مدل‌سازی رگرسیون لجستیک صادق است. به‌عنوان مثال، شما باید برخی از تشخیص‌ها را انجام دهید (فصل 24) تا مطمئن شوید که مفروضات ساخته شده توسط مدل برای داده‌های شما برآورده شده است.

علاوه بر این، باید اندازه‌گیری کنید که مدل در پیش‌بینی نتیجه مشاهدات داده‌های آزمایشی جدید چقدر خوب است. در اینجا، نحوه محاسبه دقت طبقه‌بندی خام را شرح دادیم، اما معیارهای عملکرد مهم دیگری وجود دارد (فصل 29)

در شرایطی که پیش‌بینی‌کننده‌های زیادی دارید، می‌توانید بدون به خطر انداختن صحت پیش‌بینی، فهرست حداقلی از متغیرهای پیش‌بینی‌کننده را انتخاب کنید که بیشترین سهم را در مدل با استفاده از رگرسیون گام به گام (فصل 22) و تکنیک‌های رگرسیون لاسو (فصل 23) دارند.

علاوه بر این، می‌توانید عبارات تعاملی را در مدل اضافه کنید، یا عبارت‌های spline را اضافه کنید.

همان مسائل مربوط به متغیرهای مخدوش کننده و همبسته برای رگرسیون لجستیک اعمال می‌شود (به فصل 10 و 9 مراجعه کنید).

همچنین می‌توانید *مدل‌های افزایشی تعمیم‌یافته (*فصل 6) را زمانی که خطی بودن پیش‌بینی‌کننده را نمی‌توان در نظر گرفت، مناسب کنید. این را می‌توان با استفاده از بسته mgcv انجام داد :

**کتابخانه (**"mgcv") *# متناسب با مدل* gam.model <- **gam (**دیابت **~ s (**گلوکز) **+** جرم **+** حامله، داده = train.data، خانواده = "دوجمله ای") *# خلاصه مدل* **خلاصه (**gam.model ) *# پیش‌بینی کنید*

probabilities <- gam.model °/ o >°/o **predict (**test.data, type = "response") predicted.classes <- **ifelse (**احتمالات **>** 0.5، "pos"، "neg")

*# دقت مدل*

**میانگین (**predicted.classes **==** test.data **$** دیابت)

رگرسیون لجستیک فقط به مسائل طبقه‌بندی دو طبقه محدود می‌شود. یک فرمت به نام *رگرسیون لجستیک چند جمله‌ای* برای مسئله طبقه‌بندی چند طبقه وجود دارد (فصل 25).

توجه داشته باشید که محبوب ترین روش، برای کارهای چند کلاسه، *تحلیل تشخیصی خطی است (*فصل 26).

رگرسیون لجستیک گام به گام

**رگرسیون لجستیک گام به گام** شامل انتخاب خودکار تعداد کاهش یافته متغیرهای پیش‌بینی برای ساخت بهترین مدل رگرسیون لجستیک است. در فصل 17 بیشتر بخوانید.

این فصل نحوه محاسبه رگرسیون لجستیک گام به گام در **R را شرح می‌**دهد.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید

**کتابخانه (**tidyverse)

**کتابخانه (**کارت)

* 1. آماده‌سازی داده ها

مجموعه داده ها: PimaIndiansDiabetes2 [در بسته mlbench ]، معرفی شده در فصل 20، برای پیش‌بینی ­احتمال مثبت بودن دیابت بر اساس متغیرهای بالینی متعدد.

ما داده‌ها را به صورت تصادفی به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم می‌کنیم. اطمینان حاصل کنید که seed را برای تولید مثل بچینید.

* *داده‌ها را بارگذاری کنید و NA‌ها را حذف کنید*

**داده (**"PimaIndiansDiabetes2"، بسته = "mlbench")

PimaIndiansDiabetes2 <- **na.omit (**PimaIndiansDiabetes2)

* *داده‌ها را بررسی کنید*

**sample\_n (**Pima!ndiansDiabetes2, 3)

* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**123)

training.samples <- PimaIndiansDiabetes2$diabetes °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE)

train.data <- PimaIndiansDiabetes2[training.samples, ]

test.data <- PimaIndiansDiabetes2[ **-** training.samples, ]

* 1. محاسبه رگرسیون لجستیک گام به گام

رگرسیون لجستیک گام به گام را می‌توان به راحتی با استفاده از تابع R stepAICO موجود در بسته MASS محاسبه کرد. انتخاب مدل توسط AIC را انجام می‌دهد. گزینه‌ای به نام جهت دارد که می‌تواند مقادیر زیر را داشته باشد: "هر دو"، "به جلو"، "عقب" (به فصل 17 مراجعه کنید).

* + 1. کد R شروع سریع

**کتابخانه (**MASS)

* *مدل را برازش کنید*

مدل <- **glm (**دیابت **~** .، داده = train.data، خانواده = دوجمله ای) °/ o >°/o **stepAIC (**ردیابی = FALSE)

* *مدل نهایی انتخاب شده را خلاصه کنید*

**خلاصه (**مدل)

* *پیش‌بینی کنید*

احتمالات <- مدل 0 / 0 > 0 / 0 **predict (**test.data، type = "response") predicted.classes <- **ifelse (**احتمالات **>** 0.5، "pos"، "neg")

* *دقت مدل*

**میانگین (**predicted.classes **==** test.data **$** دیابت)

* + 1. مدل رگرسیون لجستیک کامل

مدل کامل شامل همه پیش‌بینی کننده ها:

full.model <- **glm (**دیابت **~** .، داده = train.data، خانواده = دوجمله ای) ضریب ( **full.model** )

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ## | (عرض از مبدأ) | حامله | گلوکز | فشار | سه سر | انسولین |
| ## | -9.50372 | 0.04571 | 0.04230 | -0.00700 | 0.01858 | -0.00159 |
| ## | جرم | شجره نامه | سن |  |  |  |
| ## | 0.04502 | 0.96845 | 0.04256 |  |  |  |

* + 1. انتخاب متغیر گام به گام را انجام دهید

متغیرهای مؤثر را انتخاب کنید:

|  |  |  |  |
| --- | --- | --- | --- |
| **کتابخانه (**MASS)  step.model <- full.model °/ o >°/o **coef (**step.model) | | **stepAIC (**ردیابی = FALSE) | |
| ## (عرض از مبدأ) | گلوکز | شجره نامه انبوه | سن |
| ## -9.5612 | 0.0379 | 0.0523 0.9697 | 0.0529 |

تابع یک مدل نهایی را انتخاب کرد که در آن یک متغیر از مدل کامل اصلی حذف شده است. پیش‌بینی افت شده عبارت است از: عضله سه سر .

* + 1. مدل‌های کامل و گام به گام را مقایسه کنید

در اینجا، ما عملکرد مدل‌های لجستیک کامل و گام به گام را با هم مقایسه می‌کنیم. بهترین مدل به‌عنوان مدلی تعریف می‌شود که کمترین میزان خطای طبقه‌بندی را در پیش‌بینی کلاس داده‌های تست جدید داشته باشد:

صحت پیش‌بینی مدل رگرسیون لجستیک کامل:

* *پیش‌بینی کنید*

probabilities <- full.model °/ o >°/o **predict (**test.data, type = "response") predicted.classes <- **ifelse (**احتمالات **>** 0.5، "pos"، "neg")

* *صحت پیش‌بینی*

مشاهده شده.کلاس <- test.data **$** دیابت

**میانگین (**predicted.classes **==** observed.classes)

* # [1] 0.808

صحت پیش‌بینی مدل رگرسیون لجستیک گام به گام:

* *پیش‌بینی کنید*

probabilities <- **predict (**step.model، test.data، type = "response") predicted.classes <- **ifelse (**احتمالات **>** 0.5، "pos"، "neg")

* *صحت پیش‌بینی*

مشاهده شده.کلاس <- test.data **$** دیابت

**میانگین (**predicted.classes **==** observed.classes)

* # [1] 0.795
  1. بحث

این فصل نحوه انجام رگرسیون لجستیک گام به گام در R را شرح می‌دهد. در مثال ما، رگرسیون گام به گام تعداد کاهش یافته‌ای از متغیرهای پیش‌بینی را انتخاب کرده است که منجر به یک مدل نهایی می‌شود که عملکرد مشابه مدل کامل بود.

بنابراین، انتخاب گام به گام پیچیدگی مدل را بدون به خطر انداختن دقت آن کاهش داد. توجه داشته باشید که همه چیز برابر است، ما همیشه باید مدل ساده‌تر را انتخاب کنیم، در اینجا مدل نهایی که با رگرسیون گام به گام برمی‌گردد.

جایگزین دیگری برای روش گام به گام، برای انتخاب مدل، رویکرد رگرسیون جریمه شده است (فصل 23) که مدل را به دلیل داشتن دو متغیر زیاد جریمه می‌کند.

رگرسیون لجستیک مجازات شده

* 1. مقدمه

هنگامی که چندین متغیر در مدل رگرسیون لجستیک خود دارید، یافتن مجموعه‌ای از متغیرها که منجر به یک مدل عملکرد بهینه می‌شود، ممکن است مفید باشد (به فصل 18 مراجعه کنید).

**رگرسیون لجستیک** جریمه‌ای برای مدل لجستیک به دلیل داشتن متغیرهای زیاد اعمال می‌کند. این منجر به کوچک شدن ضرایب متغیرهای کمتر کمک کننده به سمت صفر می‌شود. این نیز به‌عنوان **منظم شناخته می‌شود** .

متداول ترین رگرسیون جریمه شده عبارتند از:

* **رگرسیون ریج** : متغیرهایی با سهم جزئی دارای ضرایب نزدیک به صفر هستند. با این حال، تمام متغیرها در مدل گنجانده شده است. این زمانی مفید است که همه متغیرها باید با توجه به دانش دامنه در مدل گنجانده شوند.
* **رگرسیون لاسو** : ضرایب برخی از متغیرهای کمتر کمک کننده مجبور هستند دقیقاً صفر باشند. فقط مهم ترین متغیرها در مدل نهایی نگهداری می‌شوند.
* **رگرسیون شبکه الاستیک** : ترکیبی از رگرسیون ریج و لاسو. برخی از ضرایب را به سمت صفر کوچک می‌کند (مثل رگرسیون ریج) و برخی ضرایب را دقیقاً به صفر می‌رساند (مانند رگرسیون لاسو)

این فصل نحوه محاسبه رگرسیون لجستیک جریمه شده، مانند رگرسیون لاسو، برای انتخاب خودکار یک مدل بهینه حاوی بیشترین متغیرهای پیش‌بینی کننده را شرح می‌دهد.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید
* glmnet، برای محاسبه رگرسیون جریمه شده

**کتابخانه (**tidyverse)

**کتابخانه (**کارت)

**کتابخانه (**glmnet)

* 1. آماده‌سازی داده ها

مجموعه داده ها: PimaIndiansDiabetes2 [در بسته mlbench ]، معرفی شده در فصل 20، برای پیش‌بینی ­احتمال مثبت بودن دیابت بر اساس متغیرهای بالینی متعدد.

ما داده‌ها را به صورت تصادفی به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم می‌کنیم. اطمینان حاصل کنید که seed را برای تولید مثل بچینید.

* *داده‌ها را بارگذاری کنید و NA‌ها را حذف کنید*

**داده (**"PimaIndiansDiabetes2"، بسته = "mlbench")

PimaIndiansDiabetes2 <- **na.omit (**PimaIndiansDiabetes2)

* *داده‌ها را بررسی کنید*

**sample\_n (**Pima!ndiansDiabetes2, 3)

* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**123)

training.samples <- PimaIndiansDiabetes2$diabetes °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE)

train.data <- PimaIndiansDiabetes2[training.samples, ]

test.data <- PimaIndiansDiabetes2[ **-** training.samples, ]

23.4 محاسبه رگرسیون لجستیک جریمه شده

* + 1. آماده‌سازی داده‌های اضافی

تابع R model.matrixO به ایجاد ماتریس پیش‌بینی کننده‌ها کمک می‌کند و همچنین به‌طور خودکار پیش‌بینی کننده‌های طبقه‌بندی را به متغیرهای ساختگی مناسب تبدیل می‌کند که برای تابع glmnetO لازم است.

* *متغیرهای پیش‌بینی طبقه‌بندی کد ساختگی*

x <- **model.matrix (**دیابت **-** .، train.data)[، **-** 1]

* *نتیجه (کلاس) را به یک متغیر عددی تبدیل کنید*

y <- **ifelse (**train.data **$** diabeti **==** "pos"، 1، 0)

* + 1. توابع R

ما از تابع R glmnet() [بسته glmnet] برای محاسبه رگرسیون لجستیک جریمه شده استفاده خواهیم کرد.

فرمت ساده شده به شرح زیر است:

**glmnet (**x، y، خانواده = "دو جمله ای"، آلفا = 1، لامبدا = NULL)

* x : ماتریس متغیرهای پیش‌بینی کننده
* y : متغیر پاسخ یا نتیجه که یک متغیر باینری است.
* خانواده : نوع پاسخ از "دو جمله ای" برای یک متغیر نتیجه باینری استفاده کنید
* آلفا : پارامتر اختلاط شبکه الاستیک. مقادیر مجاز عبارتند از:
* «1»: برای رگرسیون لاسو
* "0": برای رگرسیون خط الراس
* مقداری بین 0 و 1 (مثلا 0.3) برای رگرسیون شبکه الاستیک .

• lamba : یک مقدار عددی که میزان انقباض را تعیین می‌کند. باید توسط تحلیلگر مشخص شود.

در رگرسیون جریمه شده، باید یک لامبدا ثابت را برای تنظیم میزان انقباض ضریب مشخص کنید. بهترین لامبدا برای داده‌های شما، می‌تواند به‌عنوان لامبدا تعریف شود که میزان خطای پیش‌بینی اعتبار متقاطع را به حداقل می‌رساند. این را می‌توان به‌طور خودکار با استفاده از تابع ()cv.glmnet تعیین کرد.

در کد R زیر نحوه محاسبه رگرسیون لاسو را با تعیین گزینه alpha = 1 نشان خواهیم داد. همچنین می‌توانید رگرسیون ریج را با استفاده از آلفا = 0 امتحان کنید تا ببینید کدام برای داده‌های شما بهتر است.

* + 1. کد R شروع سریع

برازش مدل رگرسیون جریمه شده لاسو:

**کتابخانه (**glmnet)

* *بهترین لامبدا را با استفاده از اعتبارسنجی متقاطع پیدا کنید*

set.seed (123)

cv.lasso <- **cv.glmnet (**x، y، آلفا = 1، خانواده = "دو جمله ای")

* *مدل نهایی را بر روی داده‌های آموزشی قرار دهید*

مدل <- **glmnet (**x، y، آلفا = 1، خانواده = "دوجمله ای"، lambda = cv.lasso **$** lambda.min)

* *نمایش ضرایب رگرسیون*

**ضریب (**مدل)

* *روی داده‌های آزمون پیش‌بینی کنید*

x.test <- **model.matrix (**دیابت **~** .، test.data)[، **-** 1] احتمالات <- مدل °/ o >°/o **پیش‌بینی (**newx = x.test)

predicted.classes <- **ifelse (**احتمالات **>** 0.5، "pos"، "neg")

* *دقت مدل*

Observed.classes <- test.data **$ میانگین** دیابت (predicted.classes **==** observed.classes)

* + 1. رگرسیون لاسو را محاسبه کنید

مقدار بهینه لامبدا را بیابید که خطای اعتبارسنجی متقاطع را به حداقل می‌رساند : library (glmnet) set.seed (123)

cv.lasso <- **cv.glmnet (**x، y، آلفا = 1، خانواده = "دو جمله ای") **نمودار (**cv.lasso)

log (لامبدا)

نمودار خطای اعتبارسنجی متقاطع را با توجه به گزارش لامبدا نشان می‌دهد. خط عمودی چین دار سمت چپ نشان می‌دهد که لاگ مقدار بهینه لامبدا تقریباً 5- است که خطای پیش‌بینی را به حداقل می‌رساند. این مقدار لامبدا دقیق ترین مدل را ارائه می‌دهد. مقدار دقیق لامبدا را می‌توان به صورت زیر مشاهده کرد:

cv.lasso **$** lambda.min

## [1] 0.00871

به‌طور کلی، هدف از قاعده‌سازی، ایجاد تعادل بین دقت و سادگی است. این به این معنی است که مدلی با کمترین تعداد پیش‌بینی کننده که دقت خوبی نیز ارائه می‌دهد. برای این منظور، تابع cv.glmnetO همچنین مقدار لامبدا را پیدا می‌کند که ساده ترین مدل را ارائه می‌دهد، اما همچنین در یک خطای استاندارد مقدار بهینه لامبدا قرار دارد. این مقدار lambda.lse نامیده می‌شود .

cv.lasso **$** lambda.1se

## [1] 0.0674

استفاده از lambda.min به‌عنوان بهترین لامبدا، ضرایب رگرسیون زیر را به دست می‌دهد: **coef (**cv.lasso، cv.lasso **$** lambda.min)

|  |  |
| --- | --- |
| ##9x1 ماتریس پراکنده از کلاس "dgCMatrix | |
| ## | 1 |
| ## (عرض از مبدأ) ■ | -8.615615 |
| ##باردار | 0.035076 |
| ## گلوکز | 0.036916 |
| ## فشار | . |
| ## سه سر بازو | 0.016484 |
| ## انسولین | -0.000392 |
| ## جرم | 0.030485 |
| ## شجره نامه | 0.785506 |
| ## سن | 0.036265 |

از خروجی بالا، تنها عضله سه سر زنده ضریب دقیقاً برابر با صفر است.

استفاده از lambda.lse به‌عنوان بهترین لامبدا، ضرایب رگرسیون زیر را به دست می‌دهد:

**coef (**cv.lasso، cv.lasso **$** lambda.lse)

|  |  |
| --- | --- |
| ## 9 xl ماتریس پراکنده کلاس "dgCMatrix" | |
| ## | ل |
| ## (عرض از مبدأ) ■ | -4.65750 |
| ##باردار | . |
| ## گلوکز | 0.02628 |
| ## فشار | . |
| ## سه سر بازو | 0.00l9l |
| ## انسولین | . |
| ## جرم | . |
| ## شجره نامه | . |
| ## سن | 0.0l734 |

با استفاده از lambda.lse، تنها 5 متغیر دارای ضرایب غیر صفر هستند. ضرایب همه متغیرهای دیگر توسط الگوریتم لاسو صفر شده است و پیچیدگی مدل را کاهش می‌دهد.

تنظیم lambda = lambda.lse مدل ساده‌تری را در مقایسه با lambda.min ایجاد می‌کند، اما ممکن است دقت مدل کمی کمتر از مدل به‌دست‌آمده با lambda.min باشد.

در بخش‌های بعدی، مدل نهایی را با استفاده از lambda.min محاسبه می‌کنیم و سپس دقت مدل را در برابر داده‌های آزمایش ارزیابی می‌کنیم. همچنین در مورد نتایج به دست آمده از برازش مدل با استفاده از lambda = lambda.lse بحث خواهیم کرد.

مدل نهایی لاسو را محاسبه کنید :

• مدل نهایی را با استفاده از lambda.min محاسبه کنید :

* *مدل نهایی با lambda.min*

lasso.model <- **glmnet (**x، y، آلفا = l، خانواده = "دو جمله ای"، lambda = cv.lasso **$** lambda.min)

* *داده‌های آزمایش را پیش‌بینی کنید*

x.test <- **model.matrix (**دیابت **~** ., test.data)[, **-** l] احتمالات <- lasso.model °/ o >°/o **پیش‌بینی (**newx = x.test) پیش‌بینی شده. کلاس‌ها <- **ifelse (** احتمالات **>** 0.5، "pos"، "neg")

* *دقت مدل*

Observed.classes <- test.data **$ میانگین** دیابت (predicted.classes **==** observed.classes)

* # [l] 0.769

• مدل نهایی را با استفاده از lambda.lse محاسبه کنید :

* *مدل نهایی با lambda.lse*

lasso.model <- **glmnet (**x، y، آلفا = l، خانواده = "دوجمله ای"، lambda = cv.lasso **$** lambda.lse)

* *داده‌های آزمایش را پیش‌بینی کنید*

x.test <- **model.matrix (**دیابت **~** ., test.data)[, **-** l] احتمالات <- lasso.model 0 / 0 > 0 / 0 **predict (**newx = x.test) predicted.classes <- **ifelse (**احتمالات **>** 0.5، "pos"، "neg")

* *میزان دقت مدل*

مشاهده شده.کلاس <- test.data **$** دیابت

**میانگین (**predicted.classes **==** observed.classes)

* # [1] 0.705

در بخش‌های بعدی، دقت به‌دست‌آمده با رگرسیون لاسو را با دقت به‌دست‌آمده با استفاده از مدل رگرسیون لجستیک کامل (شامل همه پیش‌بینی‌کننده‌ها) مقایسه می‌کنیم.

23.4.5 مدل لجستیک کامل را محاسبه کنید

* *مدل را برازش کنید*

full.model <- **glm (**دیابت **~** .، داده = train.data، خانواده = دوجمله ای)

* *پیش‌بینی کنید*

probabilities <- full.model °/ o >°/o **predict (**test.data, type = "response") predicted.classes <- **ifelse (**احتمالات **>** 0.5، "pos"، "neg")

* *دقت مدل*

مشاهده شده.کلاس <- test.data **$** دیابت

**میانگین (**predicted.classes **==** observed.classes)

* # [1] 0.808

23.5 بحث

در این فصل نحوه محاسبه مدل رگرسیون لجستیک جریمه شده در R توضیح داده شده است. در اینجا، ما بر روی مدل لاسو تمرکز کردیم، اما شما همچنین می‌توانید رگرسیون ریج را با استفاده از آلفا = 0 در تابع glmnet() برازش دهید. برای رگرسیون شبکه الاستیک ، باید مقدار آلفا را بین 0 و 1 انتخاب کنید. این کار را می‌توان به‌طور خودکار با استفاده از بسته caret انجام داد. به فصل 18 مراجعه کنید.

تحلیل ما نشان داد که رگرسیون لاسو، با استفاده از lambda.min به‌عنوان بهترین لامبدا، منجر به مدل ساده‌تر می‌شود بدون اینکه عملکرد مدل در داده‌های آزمایشی در مقایسه با مدل لجستیک کامل به خطر بیفتد.

دقت مدلی که با lambda.lse به دست آورده‌ایم کمی کمتر از آنچه در مدل پیچیده‌تر با استفاده از همه متغیرهای پیش‌بینی‌کننده (n=8) یا استفاده از lambda.min در رگرسیون لاسو به دست آورده‌ایم است. حتی با lambda.lse، دقت به‌دست‌آمده علاوه بر سادگی مدل به‌اندازه کافی خوب باقی می‌ماند.

این بدان معناست که مدل ساده‌تر به‌دست‌آمده با رگرسیون لاسو، حداقل به اندازه مدل پیچیده‌تر، با اطلاعات موجود در داده‌ها کار می‌کند. با توجه به مبادله بایاس واریانس، همه چیز یکسان و مدل ساده‌تر باید همیشه ترجیح داده شود زیرا احتمال کمتری دارد که داده‌های آموزشی بیش برازش شود.

برای انتخاب متغیر، یک جایگزین برای تکنیک‌های رگرسیون لجستیک جریمه‌شده، رگرسیون لجستیک مرحله‌ای است ­که در فصل ۲۲ توضیح داده شد.

فصل 24

مفروضات و تشخیص رگرسیون لجستیک

* 1. مقدمه

مدل **رگرسیون لجستیک** چندین **فرض را** در مورد داده‌ها ایجاد می‌کند.

این فصل مفروضات اصلی را تشریح می‌کند و راهنمای عملی را در R ارائه می‌کند تا بررسی کند که آیا این مفروضات برای داده‌های شما صادق هستند که برای ساخت یک مدل خوب ضروری است.

مطمئن شوید که اصول رگرسیون لجستیک در فصل 21 را خوانده اید.

* 1. مفروضات رگرسیون لجستیک

روش رگرسیون لجستیک فرض می‌کند که:

* نتیجه یک متغیر باینری یا دوگانه است مانند بله در مقابل خیر، مثبت در مقابل منفی، 1 در مقابل 0.
* یک رابطه خطی بین منطق نتیجه و هر یک از متغیرهای پیش‌بینی کننده وجود دارد. به یاد بیاورید که تابع logit logit(p) = log(p/(lp)) است که در آن p احتمالات نتیجه است (به فصل 21 مراجعه کنید).
* در پیش‌بینی‌کننده‌های پیوسته، هیچ مقدار تأثیرگذاری (مقادیر شدید یا پرت) وجود ندارد
* هیچ ارتباط متقابل بالایی (یعنی چند خطی) در بین پیش‌بینی کننده‌ها وجود ندارد.

برای بهبود دقت مدل خود، باید مطمئن شوید که این مفروضات برای داده‌های شما صادق هستند. در بخش‌های بعدی، نحوه تشخیص مسائل احتمالی در داده‌ها را شرح خواهیم داد.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* جارو : یک چارچوب داده مرتب از نتایج آزمون آماری ایجاد می‌کند

**کتابخانه (**tidyverse)

**کتابخانه (**جارو)

**مجموعه\_تم (** **تم\_کلاسیک (**))

* 1. ساخت یک مدل رگرسیون لجستیک

ما با محاسبه نمونه‌ای از مدل رگرسیون لجستیک با استفاده از PimaIndiansDiabetes2 [بسته mlbench]، معرفی شده در فصل 20، برای پیش‌بینی احتمال مثبت بودن تست دیابت بر اساس متغیرهای بالینی، شروع می‌کنیم.

* *داده‌ها را بارگذاری کنید*

**داده (**"PimaIndiansDiabetes2"، بسته = "mlbench")

PimaIndiansDiabetes2 <- **na.omit (**PimaIndiansDiabetes2)

* *برازش مدل رگرسیون لجستیک*

مدل <- **glm (**دیابت **~** .، داده = PimaIndiansDiabetes2، خانواده = دوجمله ای)

* *پیش‌بینی احتمال (p)* احتمال مثبت دیابت <- **پیش‌بینی (**مدل، نوع = "پاسخ") پیش‌بینی شده. کلاس‌ها <- **ifelse (**احتمالات **>** 0.5، "pos"، "neg") **سر (**پیش‌بینی‌شده. کلاس‌ها)

## 4 5 7 9 14 15

## "neg" "pos" "neg" "pos" "pos" "pos"

24.5 تشخیص رگرسیون لجستیک

* + 1. فرض خطی بودن

در اینجا، رابطه خطی بین متغیرهای پیش‌بینی‌کننده پیوسته و منطق نتیجه را بررسی می‌کنیم. این را می‌توان با بازرسی بصری نمودار پراکندگی بین هر پیش‌بینی کننده و مقادیر لاجیت انجام داد.

1. متغیرهای کیفی را از قاب داده اصلی حذف کنید و مقادیر لاجیت را به داده‌ها متصل کنید:

* *فقط پیش‌بینی‌های عددی را انتخاب کنید*

mydata <- PimaIndiansDiabetes2 °/ o >°/o **dplyr ::select\_if (**is.numeric)

پیش‌بینی کننده <- **colnames (**mydata)

* *logit را متصل کنید و داده‌ها را برای نمودار مرتب کنید*

mydata <- mydata 0 / 0 > 0 / 0

**جهش (**logit = **log (**احتمالات **/ (**1 **-** احتمالات))) °/ o >°/o **جمع‌آوری (**کلید = "پیش‌بینی کننده ها"، مقدار = "پیش‌بینی کننده. ارزش"، **-** logit)

1. نمودارهای پراکندگی را ایجاد کنید:

**ggplot (**mydata، **aes (**logit، predictor.value)) **+ geom\_point (**اندازه = 0.5، آلفا = 0.5) **+ geom\_smooth (**روش = "loess") **+**

**theme\_bw (**) **+**

**facet\_wrap (** **~** پیش‌بینی کننده ها، مقیاس‌ها = "free\_y")

نمودارهای پراکندگی هموار نشان می‌دهد که متغیرهای گلوکز، جرم، حاملگی، فشار و عضله سه سر همگی به‌طور خطی با نتیجه دیابت در مقیاس لاجیت مرتبط هستند.

متغیر سن و شجره خطی نیست و ممکن است نیاز به تغییراتی داشته باشد. اگر نمودار پراکندگی غیرخطی بودن را نشان می‌دهد، برای ساخت مدل به روش‌های دیگری مانند عبارت‌های 2 یا 3 توانی، چندجمله‌ای کسری و تابع spline نیاز دارید (فصل 6).

* + 1. مقادیر تاثیرگذار

مقادیر تأثیرگذار نقاط داده‌ای شدید هستند که می‌توانند کیفیت مدل رگرسیون لجستیک را تغییر دهند.

شدیدترین مقادیر در داده‌ها را می‌توان با تجسم مقادیر فاصله کوک بررسی کرد. در اینجا ما 3 مقدار برتر را برچسب گذاری می‌کنیم:

**نمودار (**مدل که =4، id.n = 3)

100 200 300 400

Obs. عدد   
glm (دیابت ~.)

توجه داشته باشید که همه موارد پرت مشاهدات تأثیرگذار نیستند. برای بررسی اینکه آیا داده‌ها حاوی مشاهدات تأثیرگذار بالقوه هستند، می‌توان خطای باقیمانده استاندارد شده را بررسی کرد. نقاط داده با باقیمانده‌های استاندارد شده مطلق بالای 3 نشان دهنده نقاط پرت احتمالی هستند و ممکن است مستحق توجه بیشتر باشند.

کد R زیر باقیمانده‌های استاندارد شده ( .std.resid ) و فاصله Cook ( .cooksd ) را با استفاده از تابع R augmentO [بسته جارو] محاسبه می‌کند.

*# نتایج مدل را استخراج کنید*

model.data <- **افزایش (**مدل) °/ o >°/o

**جهش (**شاخص = l **:n (**))

داده‌های 3 بزرگ‌ترین مقدار، با توجه به فاصله کوک، می‌تواند به صورت زیر نمایش داده شود:

model.data **/>/ top\_n (**3، .cooksd)

بقایای استاندارد شده را رسم کنید:

**ggplot (**model.data، **aes (**index، .std.resid)) **+ geom\_point (** **aes (**رنگ = دیابت)، آلفا = 0.5) **+ theme\_bw (**)

1

"U w ® 0

2 to

-1

diabetes

o neg

o pos

-2

-3 0 100 200 300 400

فهرست مطالب

نقاط داده تاثیرگذار بالقوه را با abs(.std.res) > 3 : model.data °/ o >°/o فیلتر کنید

**فیلتر (** **abs (**.std.resid) **>** 3)

هیچ مشاهدات تاثیرگذاری در داده‌های ما وجود ندارد.

هنگامی که در یک پیش‌بینی‌کننده پیوسته مقادیر پرت دارید، راه‌حل‌های بالقوه عبارتند از:

* حذف سوابق مربوطه
* داده‌ها را به مقیاس ورود به سامانه تبدیل کنید
* از روش‌های غیر پارامتریک استفاده کنید

24.5.3 چند خطی

چند خطی به موقعیتی مربوط می‌شود که در آن داده‌ها حاوی متغیرهای پیش‌بینی کننده بسیار همبسته هستند. در فصل 9 بیشتر بخوانید.

چند خطی بودن موضوع مهمی در تحلیل رگرسیون است و باید با حذف متغیرهای مربوطه رفع شود. می‌توان آن را با استفاده از تابع R vif() [بسته خودرو] ارزیابی کرد که فاکتورهای تورم واریانس را محاسبه می‌کند:

ماشین **::vif (**مدل)

## باردار فشار گلوکز سه سر انسولین توده شجره نامه سن

## 1.89 1.38 1.19 1.64 1.38 1.83 1.03 1.97

به‌عنوان یک قاعده کلی، مقدار VIF که بیش از 5 یا 10 است، مقدار مشکلی از همخطی بودن را نشان می‌دهد. در مثال ما، هیچ خطی وجود ندارد: همه متغیرها مقدار VIF بسیار کمتر از 5 دارند.

24.6 بحث

این فصل مفروضات اصلی مدل رگرسیون لجستیک را تشریح می‌کند و نمونه‌هایی از کد R برای مسائل بالقوه تشخیصی در داده‌ها، از جمله غیر خطی بودن بین متغیرهای پیش‌بینی‌کننده و منطق نتیجه، وجود مشاهدات تأثیرگذار در داده‌ها و چند خطی بودن در میان پیش‌بینی‌کننده‌ها ارائه می‌کند. رفع این مسائل بالقوه ممکن است به‌طور قابل‌توجهی خوب بودن مدل را بهبود بخشد. همچنین ببینید، معیارهای عملکرد اضافی برای بررسی اعتبار مدل شما در فصل 29 توضیح داده شده است.

رگرسیون لجستیک چند جمله‌ای

* 1. مقدمه

**رگرسیون لجستیک چند** جمله‌ای بسط رگرسیون لجستیک (فصل 21) برای وظایف طبقه‌بندی چند طبقه است. زمانی استفاده می‌شود که نتیجه شامل بیش از دو کلاس باشد.

در این فصل، ما به شما نشان خواهیم داد که چگونه رگرسیون لجستیک چند جمله‌ای را در R محاسبه کنید.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری آسان داده ها
* مراقبت برای مدل‌سازی پیش‌بینی آسان
* nnet برای محاسبه رگرسیون لجستیک چند جمله ای

**library (**tidyverse) **library (**caret) **library (**nnet)

* 1. آماده‌سازی داده ها

ما از مجموعه داده‌های زنبق، معرفی شده در فصل 20، برای پیش‌بینی گونه‌های زنبق بر اساس متغیرهای پیش‌بینی Sepal.Length، Sepal.Width، Petal.Length، Petal.Width استفاده خواهیم کرد.

ما با تقسیم تصادفی داده‌ها به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) شروع می‌کنیم. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* **های داده** *را بارگیری کنید (*"زنبق") *# نمونه داده‌ها را بازرسی کنید (* **زنبق،** 3)
* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**123)

training.samples <- iris$Species °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE) train.data <- irisftraining.samples، ] test.data <- irisf **-** training.samples، ]

* 1. محاسبه رگرسیون لجستیک چند جمله ای
* *مدل را برازش کنید*

model <- nnet **::multinom (**Species **~** ., data = train.data) *# خلاصه مدل را* **خلاصه کنید (**مدل) *# پیش‌بینی کنید*

predicted.classes <- model °/ o >°/o **predict (**test.data) **head (**predicted.classes)

* *دقت مدل*

**میانگین (**predicted.classes **==** test.data **$** گونه)

دقت مدل:

**میانگین (**predicted.classes **==** test.data **$** گونه)

* # [1] 0.967

مدل ما در پیش‌بینی دسته‌های مختلف با دقت 97 درصد بسیار خوب است.

* 1. بحث

این فصل نحوه محاسبه رگرسیون لجستیک چند جمله‌ای را در R توضیح می‌دهد. این روش برای مسائل چند کلاسه استفاده می‌شود. در عمل، اغلب استفاده نمی‌شود. تحلیل متمایز (فصل 26) برای طبقه‌بندی چند طبقه محبوب تر است.

تحلیل تشخیصی

* 1. مقدمه

**تحلیل تفکیک** کننده برای پیش‌بینی احتمال تعلق به یک کلاس معین (یا طبقه‌بندی ­) بر اساس یک یا چند متغیر پیش‌بینی استفاده می‌شود. با متغیرهای پیش‌بینی کننده مستمر و/یا طبقه‌بندی کار می‌کند.

قبلاً، ما رگرسیون لجستیک را برای مسائل طبقه‌بندی دو کلاسه توضیح داده‌ایم، یعنی زمانی که متغیر نتیجه دارای دو مقدار ممکن است (0/1، خیر/بله، منفی/مثبت).

در مقایسه با رگرسیون لجستیک، تحلیل تفکیکی برای پیش‌بینی مقوله مشاهده در شرایطی که متغیر نتیجه شامل بیش از دو کلاس است، مناسب‌تر است. علاوه بر این، از رگرسیون لجستیک برای مسائل طبقه‌بندی چند طبقه پایدارتر است.

توجه داشته باشید که هم رگرسیون لجستیک و هم تحلیل تفکیکی می‌توانند برای وظایف طبقه‌بندی باینری استفاده شوند.

در این فصل، پرکاربردترین تکنیک‌های تحلیل متمایز و برنامه‌های افزودنی را ­خواهید آموخت. علاوه بر این، ما کد R را برای انجام انواع مختلف تحلیل ارائه خواهیم کرد.

روش‌های تحلیل متمایز زیر شرح داده خواهد شد:

* **تحلیل تشخیص خطی (** **LDA** ): از ترکیب خطی پیش‌بینی کننده‌ها برای پیش‌بینی کلاس یک مشاهده معین استفاده می‌کند. فرض می‌کنیم که متغیرهای پیش‌بینی‌کننده (p) به‌طور نرمال توزیع شده‌اند و کلاس‌ها دارای واریانس‌های یکسان (برای تحلیل تک متغیره، p = 1) یا ماتریس‌های کوواریانس یکسان (برای تحلیل چند متغیره، p > 1) هستند.
* **تحلیل تفکیک درجه دوم (** **QDA** ): انعطاف پذیرتر از LDA. در اینجا، هیچ فرضی وجود ندارد که ماتریس کوواریانس کلاس‌ها یکسان باشد.
* **تحلیل تفکیک مخلوط (** **MDA** ): هر کلاس یک ترکیب گاوسی ­از زیر کلاس‌ها فرض می‌شود.
* **تحلیل تفکیک انعطاف‌پذیر (** **FDA** ): از ترکیبات غیر خطی پیش‌بینی کننده‌ها مانند اسپلاین استفاده می‌شود.
* **تحلیل تفکیک منظم (** **RDA** ): منظم‌سازی (یا انقباض) تخمین ماتریس‌های کوواریانس را در شرایطی که تعداد پیش‌بینی‌ها از تعداد نمونه‌ها در داده‌های آموزشی بیشتر است، بهبود می‌بخشد. این منجر به بهبود تحلیل تمایز می‌شود.
  1. بارگیری بسته‌های R مورد نیاز
* آیه مرتب برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید

**کتابخانه (**tidyverse)

**کتابخانه (**کارت)

**مجموعه\_تم (** **تم\_کلاسیک (**))

* 1. آماده‌سازی داده ها

ما از مجموعه داده‌های زنبق، معرفی شده در فصل 20، برای پیش‌بینی گونه‌های زنبق بر اساس متغیرهای پیش‌بینی Sepal.Length، Sepal.Width، Petal.Length، Petal.Width استفاده خواهیم کرد.

تحلیل تمایز می‌تواند تحت تأثیر مقیاس/واحدی که در آن متغیرهای پیش‌بینی اندازه‌گیری می‌شوند، قرار گیرد. به‌طور کلی توصیه می‌شود قبل از تحلیل، پیش‌بینی کننده پیوسته را استاندارد/نرمال کنید.

1. داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید:

* *بارگیری داده‌های* **داده (**"زنبق")
* *تقسیم داده‌ها به آموزش (80%) و مجموعه تست (20%)* **set.seed (**123)

training.samples <- iris$Species °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE) train.data <- irisftraining.samples، ] test.data <- irisf **-** training.samples، ]

1. داده‌ها را نرمال کنید. متغیرهای دسته‌بندی به‌طور خودکار نادیده گرفته می‌شوند.

* *تخمین پارامترهای پیش پردازش*

preproc.param <- train.data 0 / 0 > 0 / 0

**پیش فرآیند (**روش = **c (**"مرکز"، "مقیاس"))

* *تبدیل داده‌ها با استفاده از پارامترهای تخمینی* train.transformed <- preproc.param °/ o >°/o **predict (**train.data) test.transformed <- preproc.param °/ o >°/o **predict (**test.data)
  1. تحلیل تشخیص خطی - LDA

الگوریتم LDA با یافتن جهت‌هایی که جدایی بین کلاس‌ها را به حداکثر می‌رساند، شروع می‌شود، سپس از این جهت‌ها برای پیش‌بینی طبقه افراد استفاده می‌کند. این جهات که تفکیک کننده‌های خطی نامیده می‌شوند، ترکیبی خطی از متغیرهای پیش‌بینی کننده هستند.

LDA فرض می‌کند که پیش‌بینی‌کننده‌ها معمولاً توزیع می‌شوند (توزیع گاوسی) و کلاس‌های مختلف دارای میانگین‌های خاص کلاس و واریانس/کوواریانس برابر هستند.

قبل از انجام LDA، در نظر بگیرید:

* بررسی توزیع‌های تک متغیره هر متغیر و اطمینان از عدم ­توزیع نادرست آنها. در غیر این صورت، می‌توانید آنها را با استفاده از log و root برای توزیع‌های نمایی و باکس-کاکس برای توزیع‌های اریب تبدیل کنید.
* حذف مقادیر پرت از داده‌های خود و استانداردسازی متغیرها برای مقایسه مقیاس آنها ­.

تحلیل تفکیک خطی را می‌توان به راحتی با استفاده از تابع lda() [MASS pack ­age] محاسبه کرد.

کد R شروع سریع :

**کتابخانه (**MASS) *# متناسب با مدل* مدل <- **lda (**Species **~** ., data = train.transformed) *# پیش‌بینی کنید*

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**تست. تبدیل شده)

*# دقت مدل*

**میانگین (**پیش‌بینی **$** class **==** test.transformed **$** گونه)

محاسبه LDA :

**کتابخانه (**MASS)

مدل <- **lda (**Species **~** ., data = train.transformed) model

* # زنگ زدن:
* # lda (گونه - .، داده = قطار. تبدیل شده)

##

* # احتمالات قبلی گروه ها:
* # setosa versicolor virginica
* # 0.333 0.333 0.333

##

* #گروه یعنی:

|  |  |  |  |
| --- | --- | --- | --- |
| ## | کاسبرگ.طول | کاسبرگ | .گلبرگ عرض.گلبرگ طول.عرض |
| ## | setosa -1.012 |  | 0.787 -1.293 -1.250 |
| ## | versicolor 0.117 | ■ | -0.648 0.272 0.154 |
| ## | ویرجینیکا 0.895 | ■ | -0.139 1.020 1.095 |
| ## |  |  |  |
| ## | ضرایب خطی | متمایز کننده ها: | |
| ## | LD1 | LD2 |  |
| ## | Sepal.Length 0.911 0. | 0318 |  |
| ## | Sepal.Width 0.648 0. | 8985 |  |
| ## | گلبرگ.طول -4.082 -2. | 2272 |  |
| ## | گلبرگ. عرض -2.313 2. | 6544 |  |

* #
* # نسبت ردیابی:
* # LD1 LD2
* # 0.9905 0.0095

LDA میانگین گروه را تعیین می‌کند و برای هر فرد، احتمال تعلق به گروه‌های مختلف را محاسبه می‌کند. سپس فرد به گروهی که بالاترین امتیاز احتمال را دارد تحت تأثیر قرار می‌گیرد.

خروجی lda() حاوی عناصر زیر است:

* *احتمالات قبلی گروه‌ها* : نسبت مشاهدات آموزشی در هر گروه. به‌عنوان مثال، 31 درصد از مشاهدات تمرینی در گروه ستوزا وجود دارد
* *گروه یعنی* : مرکز ثقل گروه. میانگین هر متغیر را در هر گروه نشان می‌دهد.
* *ضرایب تمایز خطی* : ترکیب خطی متغیرهای پیش‌بینی را نشان می‌دهد که برای تشکیل قانون تصمیم LDA استفاده می‌شود، به‌عنوان مثال، LD1 = 0.91 \* Sepal. Length + 0.64 \* Sepal. Width - 4.08 \* Petal. Length - 2.3 \* Petal.Width، به‌طور مشابه، LD2 =

0.03 \* کاسبرگ. طول + 0.89 \* کاسبرگ. عرض - 2.2 \* گلبرگ. طول - 2.6 \* گلبرگ . عرض ,

با استفاده از تابع plot() نمودارهایی از متمایز کننده‌های خطی را تولید می‌کند که با محاسبه LD1 و LD2 برای هر یک از مشاهدات آموزشی به دست می‌آید.

**طرح (**مدل)

پیش‌بینی کنید :

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**تست. تبدیل شده) **نام‌ها (**پیش‌بینی‌ها) ## [1] "کلاس" "پسین" "x"

تابع ()predict عناصر زیر را برمی گرداند:

* *کلاس* : طبقات پیش‌بینی شده مشاهدات.
* *posterior* : ماتریسی است که ستون‌های آن گروه‌ها، ردیف‌ها افراد و مقادیر احتمال پسین آن هستند که مشاهده مربوطه به گروه‌ها تعلق دارد.
* *x* : حاوی متمایز کننده‌های خطی است که در بالا توضیح داده شد

نتایج را بررسی کنید: *#* **سر کلاس‌های پیش‌بینی شده (** کلاس **$ پیش‌بینی،** 6) *# احتمالات پیش‌بینی شده عضویت در کلاس.* **سر (**پیش‌بینی‌های **$** عقب، 6) # **سر** *تشخیص‌دهنده خطی (*پیش‌بینی **$** x، 3)

توجه داشته باشید که می‌توانید نمودار LDA را با استفاده از ggplot2 به صورت زیر ایجاد کنید:

lda.data <- **cbind (**train.transformed, **predict (**model) **$** x) **ggplot (**lda.data, **aes (**LD1, LD2)) **+**

**geom\_point (** **aes (**رنگ = گونه))

می توانید دقت مدل را به صورت زیر محاسبه کنید:

**میانگین (**پیش‌بینی **$** class **==** test.transformed **$** گونه)

## [1] 1

مشاهده می‌شود که مدل ما 100% مشاهدات را به درستی طبقه‌بندی کرده است که بسیار عالی است.

توجه داشته باشید که به‌طور پیش‌فرض، مرز یا آستانه احتمال مورد استفاده برای تصمیم‌گیری عضویت در گروه 0.5 است (حدس زدن تصادفی). به‌عنوان مثال، تعداد مشاهدات در گروه ستوزا را می‌توان با استفاده از موارد زیر دوباره محاسبه کرد:

**مجموع (**پیش‌بینی‌های **$** قبلی[ ,1] **>=** 0.5)

## [1] 10

در برخی شرایط، ممکن است بخواهید دقت مدل را افزایش دهید. در این حالت می‌توانید مدل را با تنظیم مرز یا آستانه احتمال خلفی تنظیم کنید. به‌عنوان مثال، می‌توانید آستانه را کم یا زیاد کنید.

انتخاب متغیر :

توجه داشته باشید که اگر متغیرهای پیش‌بینی‌کننده قبل از محاسبه LDA استاندارد شده باشند، وزن تفکیک‌کننده را می‌توان به‌عنوان معیارهایی با اهمیت متغیر برای انتخاب ویژگی استفاده کرد.

* 1. تحلیل تفکیک درجه دوم - QDA

QDA کمی انعطاف پذیرتر از LDA است، به این معنا که برابری واریانس/کوواریانس را در نظر نمی‌گیرد. به عبارت دیگر، برای QDA ماتریس کوواریانس می‌تواند برای هر کلاس متفاوت باشد.

زمانی که مجموعه آموزشی کوچکی دارید، LDA بهتر از QDA است.

در مقابل، اگر مجموعه آموزشی بسیار بزرگ باشد، QDA توصیه می‌شود، به‌طوری که واریانس طبقه‌بندی‌کننده مسئله اصلی نیست، یا اگر فرض یک ماتریس کوواریانس مشترک برای کلاس‌های K به وضوح غیرقابل دفاع باشد (جیمز و همکاران، 2014). ).

QDA را می‌توان با استفاده از تابع R qda() [بسته MASS] محاسبه کرد

**کتابخانه (**MASS)

* *مدل را برازش کنید*

مدل <- **qda (**Species **~** ., data = train.transformed) model

* *پیش‌بینی کنید*

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**تست. تبدیل شده)

* *دقت مدل*

**میانگین (**پیش‌بینی **$** class **==** test.transformed **$** گونه)

* 1. تحلیل تفکیک مخلوط - MDA

طبقه‌بندی کننده LDA فرض می‌کند که هر کلاس از یک توزیع نرمال (یا گاوسی) منفرد می‌آید. این خیلی محدود کننده است.

برای MDA، کلاس‌هایی وجود دارد و هر کلاس یک مخلوط گوسی از زیر کلاس‌ها فرض می‌شود که در آن هر نقطه داده احتمال تعلق به هر کلاس را دارد. برابری ماتریس کوواریانس، در بین طبقات، هنوز فرض می‌شود.

**کتابخانه (**mda)

* *مدل را برازش کنید*

مدل <- **mda (**Species **~** ., data = train.transformed) model

* *پیش‌بینی کنید*

predicted.classes <- model °/ o >°/o **predict (**test.transformed)

* *دقت مدل*

**میانگین (**predicted.classes **==** test.transformed **$** گونه)

همانطور که در زیر نشان داده شده است، ممکن است MDA بهتر از LDA باشد. در داده‌های این مثال، ما 3 گروه اصلی از افراد داریم که هر کدام دارای 3 زیرگروه مجاور نیستند. خطوط مشکی یکپارچه در نمودار نشان دهنده مرزهای تصمیم LDA، QDA و MDA است. مشاهده می‌شود که طبقه‌بندی کننده MDA زیر کلاس‌ها را در مقایسه با LDA و QDA به درستی شناسایی کرده است که در مدل‌سازی این داده‌ها اصلا خوب نبوده اند.

LDA Decision Boundari

MDA Decision Boundar

کد تولید نمودارهای فوق از جان رامی است[[5]](#footnote-5)

26.7 تحلیل تفکیک انعطاف‌پذیر - FDA

FDA توسعه انعطاف‌پذیر LDA است که از ترکیبات غیر خطی پیش‌بینی کننده‌ها مانند splines استفاده می‌کند. FDA برای مدل‌سازی روابط چند متغیره غیر نرمال یا غیرخطی بین متغیرهای درون هر گروه مفید است و امکان طبقه‌بندی دقیق‌تر را فراهم می‌کند.

**کتابخانه (**mda)

* *مدل را برازش کنید*

مدل <- **fda (**گونه **~** .، داده = قطار. تبدیل شده)

* *پیش‌بینی کنید*

predicted.classes <- model 0 / 0 > 0 / 0 **پیش‌بینی (**تست. تبدیل شده)

* *دقت مدل*

**میانگین (**predicted.classes **==** test.transformed **$** گونه)

* 1. 8 تحلیل متمایز منظم

RDA با منظم کردن ماتریس‌های کوواریانس گروهی یک قانون طبقه‌بندی ایجاد می‌کند (فریدمن، 1989) که به مدل قوی‌تری در برابر چند خطی بودن داده‌ها اجازه می‌دهد. این ممکن است برای مجموعه داده‌های چند متغیره بزرگ که حاوی پیش‌بینی‌کننده‌های بسیار همبسته است، بسیار مفید باشد.

تحلیل تفکیک منظم نوعی مبادله بین LDA و QDA است. به یاد بیاورید که در LDA، ماتریس کوواریانس برابری را برای همه کلاس‌ها فرض می‌کنیم. QDA ماتریس‌های کوواریانس متفاوتی را برای همه کلاس‌ها در نظر می‌گیرد. تحلیل تفکیک منظم یک واسطه بین LDA و QDA است.

RDA کوواریانس‌های جداگانه QDA را به سمت یک کوواریانس مشترک مانند LDA کوچک می‌کند. این تخمین ماتریس‌های کوواریانس را در موقعیت‌هایی که تعداد پیش‌بینی‌کننده‌ها از تعداد نمونه‌ها در داده‌های آموزشی بیشتر است، بهبود می‌بخشد و به‌طور بالقوه منجر به بهبود دقت مدل می‌شود.

**کتابخانه (**klaR)

* *مدل را برازش کنید*

مدل <- **rda (**گونه **~** .، داده = قطار. تبدیل شده)

* *پیش‌بینی کنید*

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**تست. تبدیل شده)

* *دقت مدل*

**میانگین (**پیش‌بینی **$** class **==** test.transformed **$** گونه)

* 1. 9 بحث

ما تحلیل تفکیک خطی (LDA) و الحاقات را برای پیش‌بینی کلاس یک مشاهدات بر اساس متغیرهای پیش‌بینی‌کننده چندگانه توصیف کرده‌ایم. تحلیل افتراقی در مقایسه با رگرسیون لجستیک برای مسائل طبقه‌بندی چند طبقه مناسب تر است (فصل 21).

LDA فرض می‌کند که کلاس‌های مختلف ماتریس واریانس یا کوواریانس یکسانی دارند. ما بسیاری از پسوندهای LDA را در این فصل شرح داده‌ایم. محبوب ترین توسعه LDA، تحلیل تفکیک درجه دوم (QDA) است که از LDA انعطاف پذیرتر است به این معنا که برابری ماتریس‌های کوواریانس گروهی را فرض نمی‌کند.

LDA برای مجموعه داده‌های کوچک بهتر از QDA است. QDA برای مجموعه داده‌های آموزشی بزرگ توصیه می‌شود.

طبقه‌بندی کننده ساده بیز

* 1. مقدمه

طبقه‌بندی کننده **Naive Bayes** یک روش ساده و قدرتمند است که می‌تواند برای مسائل طبقه‌بندی باینری و چند کلاسه استفاده شود.

طبقه‌بندی کننده ساده بیز، احتمال عضویت در کلاس مشاهدات را با استفاده از قضیه بیز که بر اساس احتمال شرطی است، پیش‌بینی می‌کند که احتمال وقوع چیزی است، با توجه به اینکه اتفاق دیگری قبلاً رخ داده است.

مشاهدات به کلاسی که بیشترین امتیاز احتمال را دارد اختصاص داده می‌شود.

در این فصل، نحوه انجام طبقه‌بندی ساده بیز در R با استفاده از بسته klaR و caret را خواهید آموخت.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید

**کتابخانه (**Tidyverse)

**کتابخانه (**کارت)

* 1. آماده‌سازی داده ها

متغیرهای پیش‌بین ورودی می‌توانند متغیرهای مقوله‌ای و/یا عددی باشند.

در اینجا، از PimaIndiansDiabetes2 [در بسته mlbench ]، معرفی شده در فصل 20، برای پیش‌بینی احتمال مثبت بودن دیابت بر اساس متغیرهای بالینی متعدد استفاده خواهیم کرد.

ما داده‌ها را به صورت تصادفی به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم می‌کنیم. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *داده‌ها را بارگذاری کنید و NA‌ها را حذف کنید*

**داده (**"PimaIndiansDiabetes2"، بسته = "mlbench")

PimaIndiansDiabetes2 <- **na.omit (**PimaIndiansDiabetes2)

* *داده‌ها را بررسی کنید*

**sample\_n (**Pima!ndiansDiabetes2, 3)

* *تقسیم داده‌ها به مجموعه آموزشی و تست* **set.seed (**123)

training.samples <- PimaIndiansDiabetes2$diabetes °/ o >°/o **createDataPartition (**p = 0.8، لیست = FALSE)

train.data <- PimaIndiansDiabetes2[training.samples, ]

test.data <- PimaIndiansDiabetes2[ **-** training.samples, ]

* 1. محاسبات ساده بیز

**کتابخانه (**"klaR") *# متناسب با مدل*

مدل <- **NaiveBayes (**دیابت **~** .، داده = train.data)

* *پیش‌بینی کنید*

پیش‌بینی <- مدل 0 / 0 > 0 / 0 **پیش‌بینی (**test.data)

* *دقت مدل*

**میانگین (**پیش‌بینی **$** class **==** test.data **$** دیابت)

* # [1] 0.821
  1. با استفاده از پکیج caret R

بسته caret R می‌تواند به‌طور خودکار مدل را آموزش دهد و دقت مدل را با استفاده از اعتبارسنجی متقاطع k-fold فصل 13 ارزیابی کند.

**کتابخانه (**klaR)

* *ساخت مدل* **set.seed (**123)

مدل <- **train (**دیابت **~** .، داده = train.data، روش = "nb"، trControl = **trainControl (**"cv"، عدد = 10))

* *پیش‌بینی کنید*

predicted.classes <- model °/ o >°/o **predict (**test.data)

* *دقت مدل n*

**میانگین (**predicted.classes **==** test.data **$** دیابت)

* 1. بحث

این فصل اصول طبقه‌بندی Naive Bayes را معرفی می‌کند و مثال‌های عملی را در R با استفاده از بسته klaR و caret ارائه می‌دهد.

ماشین بردار پشتیبان

* 1. مقدمه

**ماشین بردار پشتیبان (**یا **SVM** ) یک تکنیک یادگیری ماشین است که اغلب برای کارهای *طبقه‌بندی استفاده می‌شود.* به‌طور خلاصه، SVM با شناسایی مرز تصمیم‌گیری بهینه که نقاط داده را از گروه‌ها (یا کلاس‌های) مختلف جدا می‌کند، کار می‌کند و سپس کلاس مشاهدات جدید را بر اساس این مرز جداسازی پیش‌بینی می‌کند.

بسته به موقعیت، گروه‌های مختلف ممکن است با یک خط مستقیم خطی یا با یک خط مرزی غیرخطی قابل تفکیک باشند.

روش‌های ماشین بردار پشتیبان می‌توانند هم مرزهای کلاس خطی و هم غیرخطی را کنترل کنند. می‌توان از آن برای مسائل طبقه‌بندی دو کلاسه و چند کلاسه استفاده کرد.

در داده‌های واقعی، مرز جداسازی عموماً غیرخطی است. از نظر فنی، الگوریتم SVM با استفاده از آنچه که ترفند هسته نامیده می‌شود، یک طبقه‌بندی غیرخطی انجام می‌دهد [[6]](#footnote-6). متداول‌ترین تبدیل‌های هسته، *هسته چند جمله‌ای* و هسته *شعاعی هستند* .

توجه داشته باشید که یک پسوند SVM برای رگرسیون نیز وجود دارد که به آن رگرسیون بردار پشتیبان می‌گویند.

در این فصل، نحوه ساخت طبقه‌بندی کننده SVM با استفاده از بسته *caret* R را شرح خواهیم داد.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید

**کتابخانه (**tidyverse)

**کتابخانه (**کارت)

* 1. نمونه‌ای از مجموعه داده ها

مجموعه داده ها: PimaIndiansDiabetes2 [در بسته mlbench ]، معرفی شده در فصل 20، برای پیش‌بینی ­احتمال مثبت بودن دیابت بر اساس متغیرهای بالینی متعدد.

ما داده‌ها را به صورت تصادفی به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم می‌کنیم. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *داده‌ها را بارگذاری کنید*

**داده‌ها (**"Pima!ndiansDiabetes2"، بسته = "mlbench")

pima.data <- **na.omit (**PimaIndiansDiabetes2)

* *داده‌ها را بررسی کنید*

**sample\_n (**pima.data، 3)

* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**123)

training.samples <- pima.data$diabetes °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE) train.data <- pima.data[training.samples, ] test.data <- pima.data[ **-** training.samples, ]

28.4 طبقه‌بندی کننده خطی SVM

در مثال زیر متغیرها نرمال شده اند تا مقیاس آنها قابل مقایسه باشد. این به‌طور خودکار قبل از ساخت طبقه‌بندی کننده SVM با تنظیم گزینه preProcess = c("center", "scale") انجام می‌شود.

* *مدل را بر روی مجموعه آموزشی* **set.seed قرار دهید (**123)

مدل <- **قطار (**

دیابت **~** .، داده = train.data، روش = "svmLinear"، trControl = **trainControl (**"cv"، عدد = 10)، preProcess = **c (**"مرکز"، "مقیاس")

* *روی داده‌های آزمون پیش‌بینی کنید*

predicted.classes <- model 0 / 0 > 0 / 0 **پیش‌بینی (**test.data) **head (**predicted.classes)

* # [1] neg pos neg pos pos neg
* # سطوح: پوز منفی
* *محاسبه میزان دقت مدل*

**میانگین (**predicted.classes **==** test.data **$** دیابت)

* # [1] 0.782

توجه داشته باشید که یک پارامتر تنظیم C وجود دارد که به‌عنوان *Cost* نیز شناخته می‌شود که طبقه‌بندی‌های اشتباه احتمالی را تعیین می‌کند. اساساً جریمه‌ای را برای خطا به مدل تحمیل می‌کند: هرچه مقدار C بیشتر باشد، احتمال اینکه الگوریتم SVM یک نقطه را اشتباه طبقه‌بندی کند، کمتر می‌شود.

به‌طور پیش فرض caret طبقه‌بندی کننده خطی SVM را با استفاده از C = 1 می‌سازد. با تایپ مدل در کنسول R می‌توانید این موضوع را بررسی کنید .

محاسبه خودکار SVM برای مقادیر مختلف C و انتخاب بهینه که دقت اعتبارسنجی متقابل مدل را به حداکثر می‌رساند، ممکن است.

کد R زیر SVM را برای مقادیر شبکه‌ای C محاسبه می‌کند و به‌طور خودکار مدل نهایی را برای پیش‌بینی‌ها انتخاب می‌کند:

* *مدل را در مجموعه آموزشی قرار* **دهید set.seed (**123) model <- **train (**

دیابت **~** .، داده = train.data، روش = "svmLinear"، trControl = **trainControl (**"cv"، عدد = 10)، tuneGrid = **expand.grid (**C = **seq (**0، 2، طول = 20))، preProcess = **c (**"مرکز"، "مقیاس") )

* *دقت مدل پلات در مقابل مقادیر مختلف* **نمودار هزینه (**مدل)

هزینه

* *بهترین پارامتر تنظیم C را که # دقت مدل را به حداکثر می‌رساند چاپ کنید*

مدل **$** bestTune

* #سی
* # 12 1.16
* *روی داده‌های آزمون پیش‌بینی کنید*

predicted.classes <- model °/ o >°/o **predict (**test.data)

* *محاسبه میزان دقت مدل*

**میانگین (**predicted.classes **==** test.data **$** دیابت)

* # [1] 0.782

28.5 طبقه‌بندی کننده SVM با استفاده از هسته غیر خطی

برای ساخت یک طبقه‌بندی کننده SVM غیر خطی، می‌توانیم از هسته چند جمله‌ای یا هسته شعاعی استفاده کنیم. باز هم می‌توان از بسته caret برای محاسبه آسان مدل‌های غیرخطی چند جمله‌ای و شعاعی SVM استفاده کرد.

بسته به‌طور خودکار مقادیر بهینه را برای پارامترهای تنظیم مدل انتخاب می‌کند، جایی که بهینه به‌عنوان مقادیری تعریف می‌شود که دقت مدل را به حداکثر می‌رساند.

• **محاسبه SVM با استفاده از هسته پایه شعاعی** :

* *مدل را در مجموعه آموزشی قرار* **دهید set.seed (**123) model <- **train (**

دیابت **~** .، داده = train.data، روش = "svmRadial"، trControl = **trainControl (**"cv"، عدد = 10)، preProcess = **c (**"مرکز"، "مقیاس")، طول = 10)

* *بهترین پارامتر تنظیم سیگما و C را چاپ کنید که # دقت مدل را به حداکثر می‌رساند*

مدل **$** bestTune

* # سیگما سی
* # 1 0.136 0.25
* *روی داده‌های آزمون پیش‌بینی کنید*

predicted.classes <- model °/ o >°/o **predict (**test.data)

* *محاسبه میزان دقت مدل*

**میانگین (**predicted.classes **==** test.data **$** دیابت)

* # [1] 0.795

• محاسبه SVM با استفاده از هسته پایه چند جمله‌ای :

* *مدل را در مجموعه آموزشی قرار* **دهید set.seed (**123) model <- **train (**

دیابت **~** .، داده = train.data، روش = "svmPoly"، trControl = **trainControl (**"cv"، عدد = 10)، preProcess = **c (**"مرکز"، "مقیاس")، طول = 4)

* *بهترین پارامتر تنظیم سیگما و C را چاپ کنید که # دقت مدل را به حداکثر می‌رساند*

مدل **$** bestTune

* مقیاس # درجه C
* # 8 1 0.01 2
* *روی داده‌های آزمون پیش‌بینی کنید*

predicted.classes <- model 0 / 0 > 0 / 0 **پیش‌بینی (**test.data)

* *محاسبه میزان دقت مدل*

**میانگین (**predicted.classes **==** test.data **$** دیابت)

* # [1] 0.795

در مثال‌های ما مشاهده می‌شود که طبقه‌بندی کننده SVM با استفاده از هسته غیر خطی نتیجه بهتری نسبت به مدل خطی می‌دهد.

28.6 بحث

این فصل نحوه استفاده از ماشین بردار پشتیبان را برای کارهای طبقه‌بندی توضیح می‌دهد. بومی‌های ­جایگزین دیگری مانند رگرسیون لجستیک (فصل 21) وجود دارند.

شما باید عملکرد روش‌های مختلف را روی داده‌های خود ارزیابی کنید تا بهترین را انتخاب کنید.

فصل 29

ارزیابی مدل طبقه‌بندی

* 1. مقدمه

پس از ساخت یک مدل طبقه‌بندی پیش‌بینی‌کننده، باید **عملکرد مدل را ارزیابی کنید،** یعنی مدل در پیش‌بینی نتیجه داده‌های آزمایش مشاهدات جدید که برای آموزش مدل استفاده نشده است، چقدر خوب است.

به عبارت دیگر شما باید *صحت پیش‌بینی مدل* و خطاهای پیش‌بینی را با استفاده از مجموعه داده‌های آزمایشی جدید تخمین بزنید. از آنجایی که ما نتیجه واقعی مشاهدات را در مجموعه داده‌های آزمون می‌دانیم، عملکرد مدل پیش‌بینی را می‌توان با مقایسه مقادیر نتیجه پیش‌بینی شده با مقادیر نتیجه شناخته شده ارزیابی کرد.

این فصل معیارها و روش‌های رایج مورد استفاده برای ارزیابی عملکرد مدل‌های طبقه‌بندی پیش‌بینی را توصیف می‌کند، از جمله:

* **صحت طبقه‌بندی متوسط،** نشان‌دهنده نسبت مشاهدات طبقه‌بندی شده صحیح است.
* **ماتریس درهم‌ریختگی که** جدول 2x2 است که چهار پارامتر شامل تعداد مثبت واقعی، منفی درست، منفی کاذب و مثبت کاذب را نشان می‌دهد.
* **دقت، یادآوری و خاصیت که** سه معیار عملکرد اصلی هستند که ­یک مدل طبقه‌بندی پیش‌بینی‌کننده را توصیف می‌کنند.
* **منحنی ROC که** خلاصه‌ای گرافیکی از عملکرد کلی مدل است، ­نسبت مثبت واقعی و مثبت کاذب را در تمام مقادیر ممکن مرز احتمال نشان می‌دهد. ناحیه **زیر منحنی (** **AUC** ) عملکرد کلی طبقه‌بندی کننده را خلاصه می‌کند.

ما مثال‌های عملی را در R برای محاسبه این معیارهای بالا و همچنین برای ایجاد نمودار ROC ارائه می‌کنیم.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید

**کتابخانه (**tidyverse)

**کتابخانه (**کارت)

* 1. ساخت مدل طبقه‌بندی

برای ساده نگه داشتن چیزها، یک طبقه‌بندی باینری انجام می‌دهیم که در آن متغیر نتیجه می‌تواند تنها دو مقدار ممکن داشته باشد: منفی در مقابل مثبت.

ما نمونه‌ای از مدل تحلیل متمایز خطی را با استفاده از PimaIndiansDiabetes2 [بسته mlbench]، معرفی‌شده در فصل 20، برای پیش‌بینی احتمال مثبت بودن تست دیابت بر اساس متغیرهای بالینی محاسبه می‌کنیم.

1. داده‌ها را به آموزش (80% برای ساخت مدل استفاده می‌شود) و مجموعه تست (20% برای ارزیابی عملکرد مدل استفاده می‌شود) تقسیم کنید:

* *داده‌ها را بارگذاری کنید*

**داده (**"PimaIndiansDiabetes2"، بسته = "mlbench")

pima.data <- **na.omit (**PimaIndiansDiabetes2)

* *داده‌ها را بررسی کنید*

**sample\_n (**pima.data، 3)

* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**123)

training.samples <- pima.data$diabetes °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE)

train.data <- pima.data[training.samples, ]

test.data <- pima.data[ **-** training.samples, ]

1. مدل LDA را بر روی مجموعه آموزشی قرار دهید و روی داده‌های تست پیش‌بینی کنید:

**کتابخانه (**MASS)

* *مناسب LDA*

مناسب <- **lda (**دیابت **~** .، داده = train.data)

* *روی داده‌های آزمون پیش‌بینی کنید*

پیش‌بینی‌ها <- **پیش‌بینی (**مناسب، test.data)

prediction.probabilities <- predictions **$** posterior[,2]

predicted.classes <- predictions **$** class مشاهده شده.classes <- test.data **$** دیابت

* 1. صحت طبقه‌بندی کلی

**صحت طبقه‌بندی** کلی مربوط به نسبت مشاهداتی است که به درستی طبقه‌بندی شده اند. تعیین صحت طبقه‌بندی خام اولین گام در ارزیابی عملکرد یک مدل است.

برعکس، **نرخ خطای طبقه‌بندی** به‌عنوان نسبت مشاهداتی که به اشتباه طبقه‌بندی شده اند تعریف می‌شود. میزان خطا = 1 - صحت

صحت طبقه‌بندی خام و خطا را می‌توان با مقایسه کلاس‌های مشاهده‌شده در داده‌های آزمون با کلاس‌های پیش‌بینی‌شده توسط مدل به راحتی محاسبه کرد:

دقت <- **mean (**observed.classes **==** predicted.classes) دقت

## [1] 0.808

خطا <- **mean (**observed.classes **!=** predicted.classes) خطا

## [1] 0.192

از خروجی بالا، تحلیل متمایز خطی به درستی نتیجه فردی را در 81٪ موارد پیش‌بینی کرد. این به مراتب بهتر از حدس زدن تصادفی است. نرخ خطای طبقه‌بندی اشتباه ­را می‌توان به صورت 100 - 81٪ = 19٪ محاسبه کرد.

در مثال ما، یک طبقه‌بندی کننده باینری می‌تواند دو نوع خطا ایجاد کند:

* می تواند به اشتباه فردی که دیابت مثبت است به دسته دیابت منفی اختصاص دهد
* می تواند به اشتباه فردی را که دیابت منفی است به دسته دیابت مثبت اختصاص دهد.

نسبت این دو نوع خطا را می‌توان با ایجاد یک  **ماتریس درهم‌ریختگی تعیین** کرد که مقادیر نتیجه پیش‌بینی شده را با مقادیر نتیجه شناخته شده مقایسه می‌کند.

29.5 ماتریس درهم‌ریختگی

جدول تابع R می‌توان برای تولید یک  **ماتریس درهم‌ریختگی استفاده** کرد تا مشخص شود که چه تعداد مشاهدات به درستی یا نادرست طبقه‌بندی شده اند. این مقادیر نتایج مشاهده شده و پیش‌بینی شده را مقایسه می‌کند و تعداد پیش‌بینی‌های صحیح و نادرست را بر اساس نوع نتیجه طبقه‌بندی می‌کند.

* *ماتریس درهم‌ریختگی، تعداد موارد*

**جدول (**مشاهده شده. کلاس ها، پیش‌بینی شده. کلاس ها)

* # پیش‌بینی شده . کلاس‌ها
* # مشاهده شد \_
* # neg 48 4
* # پست 11 15
* *ماتریس به هم ریختگی، نسبت موارد*

**جدول (**مشاهده شده. کلاس ها، پیش‌بینی شده. کلاس ها) °/ o >°/o

**prop.tableQ />/ گرد (**رقم = 3)

* # پیش‌بینی شده . کلاس‌ها
* # مشاهده شد
* # neg 0.615 0.051
* # pos 0.141 0.192

عناصر مورب ماتریس درهم‌ریختگی پیش‌بینی‌های درست را نشان می‌دهند، در حالی که قطرهای خارج از آن، پیش‌بینی‌های نادرست را نشان می‌دهند. بنابراین، نرخ طبقه‌بندی صحیح، مجموع عدد روی قطر تقسیم بر حجم نمونه در داده‌های آزمون است. در مثال ما، (48 + 15)/78 = 81٪ است.

هر خانه از جدول معنای مهمی دارد:

• موارد **مثبت واقعی (**د): این موارد مواردی است که پیش‌بینی می‌کردیم افراد دیابتی مثبت باشند و آنها هم بودند.

**Predicted Gasses**

| **Observed Classes (Reference)** | |
| --- | --- |
| **Negative** | **Positive** |
| **a** | **b** |
| **c** | **d** |

**Negative**

**Positive**

**شکل 29.1:**  ماتریس درهم‌ریختگی

* **منفی‌های واقعی (**الف): ما دیابت را منفی پیش‌بینی کردیم و افراد دیابتی ­منفی بودند.
* **مثبت کاذب (**ب): ما دیابت را مثبت پیش‌بینی کردیم، اما افراد در واقع دیابت نداشتند. (به‌عنوان *خطای نوع I نیز شناخته می‌شود* .)
* **منفی کاذب (**ج): ما دیابت را منفی پیش‌بینی کردیم، اما آنها دیابت داشتند. (به‌عنوان *خطای نوع II نیز شناخته می‌شود .)*

از نظر فنی، صحت پیش‌بینی خام مدل به صورت (TruePositives + TrueNegatives)/SampleSize تعریف می‌شود .

* 1. دقت، یادآوری و خاصیت

علاوه بر صحت طبقه‌بندی خام، معیارهای بسیاری دیگر وجود دارد که به‌طور گسترده برای بررسی عملکرد یک مدل طبقه‌بندی استفاده می‌شود، از جمله:

**دقت که** نسبت مثبت واقعی در بین تمام افرادی است که توسط مدل پیش‌بینی شده است که دیابت مثبت هستند. این نشان دهنده دقت یک نتیجه مثبت پیش‌بینی شده است. Precision = TruePositives/(TruePositives + FalsePositives) .

**حساسیت (**یا **یادآوری** ) که **نرخ مثبت واقعی (**TPR) یا نسبت مثبت‌های شناسایی شده در میان جمعیت مثبت دیابتی (کلاس = 1) است.

Sensitivity = TruePositives/(TruePositives + FalseNegatives) .

**خاصیت که نرخ منفی واقعی (**TNR) را اندازه‌گیری می‌کند که نسبت منفی‌های شناسایی شده در میان جمعیت دیابت منفی (کلاس = 0) است. Specificity = TrueNegatives/(TrueNegatives + FalseNegatives) .

**نرخ مثبت کاذب (**FPR) که نشان دهنده نسبت مثبت‌های شناسایی شده در میان افراد سالم (یعنی دیابت منفی) است. این را می‌توان به‌عنوان یک هشدار نادرست در نظر گرفت. FPR را می‌توان به‌عنوان 1-specificity نیز محاسبه کرد. هنگامی که موارد مثبت نادر هستند، FPR می‌تواند بالا باشد که منجر به وضعیتی می‌شود که مثبت پیش‌بینی شده به احتمال زیاد منفی است.

**حساسیت** و **خاصیت** معمولا برای سنجش عملکرد یک مدل پیش‌بینی استفاده می‌شود.

این معیارهای ذکر شده در بالا را می‌توان به راحتی با استفاده از تابع confusionMatrix () [بسته مراقبت] محاسبه کرد.

در تنظیمات دو کلاسه، ممکن است لازم باشد آرگومان اختیاری مثبت را مشخص کنید که یک رشته کاراکتری برای سطح عاملی است که با نتیجه "مثبت" مطابقت دارد (اگر برای داده‌های شما منطقی باشد). اگر فقط دو سطح عامل وجود داشته باشد، پیش فرض استفاده از سطح اول به‌عنوان نتیجه "مثبت" است.

**confusionMatrix (**predicted.classes,observed.classes, positive = "pos")

## ماتریس درهم‌ریختگی و آمار

##

## مرجع

|  |  |  |  |
| --- | --- | --- | --- |
| ## پیش‌بینی وضعیت منفی | | | |
| ##  ##  ##  ## | neg 48 11  پست 4 15  دقت | 0. | 808 |
| ## | 95% CI | (0.703, 0.888) | |
| ## | بدون نرخ اطلاعات | 0. | 667 |
| ## | P-Value [Acc > NIR] | 0. | 00439 |
| ##  ## | کاپا | 0. | 536 |
| ## | آزمون مک نمار P-Value | 0. | 12134 |
| ##  ## | حساسیت | 0. | 577 |
| ## | اختصاصی | 0. | 923 |
| ## | Pos Pred Value | 0. | 789 |
| ## | Neg Pred Value | 0. | 814 |
| ## | شیوع | 0. | 333 |
| ## | میزان تشخیص | 0. | 192 |
| ## | شیوع تشخیص | 0. | 244 |
| ## | دقت متعادل | 0. | 750 |
| ##  ## | کلاس "مثبت". | pos | |

##

نتایج فوق معیارهای آماری مختلفی را نشان می‌دهد که مهمترین آنها عبارتند از:

* جدول‌بندی متقابل بین پیش‌بینی و نتیجه شناخته شده مرجع
* دقت مدل 81%
* کاپا[[7]](#footnote-7) (54%) که دقت تصحیح شده برای شانس است.

در مثال ما، حساسیت ~ 58٪ است که نسبت افراد دیابتی مثبت ­است که به درستی توسط مدل به‌عنوان دیابت مثبت شناسایی شدند.

خاصیت مدل ~ 92٪ است که نسبت افراد دیابتی منفی است که به درستی توسط مدل به‌عنوان دیابت منفی شناسایی شدند.

دقت مدل یا نسبت مقدار پیش‌بینی شده مثبت 79 درصد است.

در علم پزشکی، حساسیت و خاصیت دو معیار مهمی هستند که عملکرد طبقه‌بندی کننده یا تست غربالگری را مشخص می‌کنند. اهمیت بین حساسیت و خاصیت بستگی به زمینه دارد. به‌طور کلی، ما نگران یکی از این معیارها هستیم.

در تشخیص پزشکی، مانند مثال ما، ما احتمالاً بیشتر نگران حداقل تشخیص مثبت اشتباه هستیم. بنابراین، ما بیشتر نگران خاصیت بالا هستیم. در اینجا، خاصیت مدل 92٪ است که بسیار خوب است.

در برخی شرایط، ممکن است بیشتر نگران تنظیم یک مدل باشیم تا حساسیت/دقت بهبود یابد. برای این منظور، می‌توانید مرز یا آستانه احتمال ‌های مختلف را آزمایش کنید تا تصمیم بگیرید کدام افراد مثبت و کدام یک منفی هستند.

توجه داشته باشید که در اینجا ما از p > 0.5 به‌عنوان آستانه احتمال استفاده کرده ایم که بالاتر از آن، افراد مورد نظر را دیابت مثبت اعلام می‌کنیم. با این حال، اگر نگران پیش‌بینی نادرست ­وضعیت مثبت دیابت برای افرادی هستیم که واقعاً مثبت هستند، می‌توانیم این آستانه را کاهش دهیم: p > 0.2 .

* 1. منحنی ROC
     1. مقدمه

منحنی **ROC (**یا **منحنی مشخصه عملکرد سیستم**) یک معیار گرافیکی محبوب برای ارزیابی عملکرد یا دقت طبقه‌بندی‌کننده است که با نسبت کل مشاهدات طبقه‌بندی شده درست مطابقت دارد.

برای مثال، دقت یک تست تشخیصی پزشکی را می‌توان با در نظر گرفتن دو نوع خطا ارزیابی کرد: مثبت کاذب و منفی کاذب. از نظر طبقه‌بندی، زمانی که احتمال پیش‌بینی‌شده مربوطه که توسط الگوریتم طبقه‌بندی‌کننده برگردانده می‌شود، بالاتر از یک آستانه ثابت باشد، آزمایش مثبت اعلام می‌شود. این آستانه معمولاً روی 0.5 (یعنی 50٪) تنظیم می‌شود که با احتمال حدس تصادفی مطابقت دارد.

1. با اشاره به مثال داده‌های دیابت ما، برای یک مرز یا آستانه احتمال ثابت معین:

* نرخ **مثبت واقعی (**یا کسری) نسبت مثبت‌های شناسایی شده در میان جمعیت دیابت مثبت است. به یاد بیاورید که این به‌عنوان **حساسیت** مدل طبقه‌بندی کننده پیش‌بینی نیز شناخته می‌شود.
* و **نرخ مثبت کاذب** نسبت مثبت‌های شناسایی شده در میان افراد سالم (یعنی دیابت منفی) است. این نیز به‌عنوان ویژگی 1 تعریف می‌شود که در آن **ویژگی، نرخ منفی ­واقعی** را اندازه می‌گیرد که نسبت منفی‌های شناسایی شده در میان جمعیت منفی دیابت است.

از آنجایی که معمولاً مرز احتمال را از قبل نمی‌دانیم، منحنی ROC معمولاً برای رسم نرخ مثبت واقعی (یا حساسیت روی محور y) در برابر نرخ مثبت کاذب (یا «ویژگی 1» در محور x) استفاده می‌شود. در تمام مرز‌های احتمالی ممکن این مبادله بین نرخی که شما می‌توانید به درستی چیزی را پیش‌بینی کنید با نرخ پیش‌بینی نادرست چیزی را نشان می‌دهد. یکی دیگر از نمایش‌های بصری نمودار ROC این است که به سادگی نمایش حساسیت در برابر خاصیت را نشان دهد.

ناحیه **زیر منحنی (** **AUC** ) عملکرد کلی طبقه‌بندی‌کننده را بر روی تمام مرز‌های احتمالی ممکن خلاصه می‌کند. این نشان دهنده توانایی یک الگوریتم طبقه‌بندی برای تشخیص 1ها از 0ها (یعنی رویدادها از غیررویدادها یا مثبتها از منفیها) است.

برای یک مدل خوب، منحنی ROC باید به شدت افزایش یابد که نشان می‌دهد نرخ مثبت واقعی (محور y) سریعتر از نرخ مثبت کاذب (محور x) با کاهش آستانه احتمال افزایش می‌یابد.

بنابراین، "نقطه ایده آل" گوشه سمت چپ بالای نمودار است که نرخ مثبت کاذب صفر و نرخ مثبت واقعی یک است. این خیلی واقع بینانه نیست، اما به این معنی است که هر چه AUC بزرگتر باشد، طبقه‌بندی کننده بهتر است.

شاخص AUC بین 0.50 (طبقه‌بندی کننده تصادفی) و 1.00 متغیر است. مقادیر بالای 0.80 نشان دهنده یک طبقه‌بندی خوب است.

در این بخش، نحوه محاسبه و ترسیم منحنی ROC در R را برای کارهای طبقه‌بندی دو کلاسه و چند کلاسه به شما نشان خواهیم داد. ما از تحلیل تشخیص خطی برای طبقه‌بندی افراد به گروه‌ها استفاده خواهیم کرد.

29.7.2 محاسبه و ترسیم منحنی ROC

تحلیل ROC را می‌توان به راحتی با استفاده از بسته R PROC انجام داد .

**کتابخانه (**pROC)

* *راک را محاسبه کنید*

res.roc <- **roc (**observed.classes, prediction.probabilities)

**plot.roc (**res.roc، print.auc = TRUE)

اختصاصی

خط مورب خاکستری نشان دهنده طبقه‌بندی کننده‌ای است که بهتر از شانس تصادفی نیست.

یک طبقه‌بندی کننده با کارایی بالا دارای یک ROC است که به شدت به گوشه سمت چپ بالا می‌رود، یعنی به درستی بسیاری از موارد مثبت را بدون طبقه‌بندی اشتباه بسیاری از موارد منفی به‌عنوان مثبت شناسایی می‌کند.

در مثال ما، AUC 0.85 است که نزدیک به حداکثر است (max = 1). بنابراین، طبقه‌بندی کننده ما را می‌توان بسیار خوب در نظر گرفت. انتظار می‌رود طبقه‌بندی کننده‌ای که عملکرد بهتری از شانس ندارد، AUC 0.5 داشته باشد که در یک مجموعه تست مستقل ارزیابی شود که برای آموزش مدل استفاده نمی‌شود.

اگر یک مدل طبقه‌بندی کننده با ویژگی حداقل 60 درصد بخواهیم، حساسیت آن حدود 0.88 درصد است. آستانه احتمال مربوطه را می‌توان به صورت زیر استخراج کرد:

* *نتایج جالبی را استخراج کنید*

roc.data <- **data\_frame (**

آستانه = res.roc **$** آستانه،

حساسیت = حساسیت‌های res.roc **$،**

ویژگی = res.roc ویژگی‌های **$**

)

* *آستانه احتمال برای ویژگی = 0.6 را دریافت کنید*

roc.data **°/ o >°/o فیلتر (**ویژگی **>=** 0.6)

## # یک تیبل: 44x3

## آستانه

## <dbl> <dbl> <dbl>

## 1 0.111 0.885 0.615

## 2 0.114 0.885 0.635

## 3 0.114 0.885 0.654

## 4 0.115 0.885 0.673

## 5 0.119 0.885 0.692

## 6 0.131 0.885 0.712 ## # ... با. 38 ردیف دیگر

بهترین آستانه با بیشترین حساسیت جمع + ویژگی را می‌توان به صورت زیر چاپ کرد. ممکن است بیش از یک آستانه وجود داشته باشد.

**plot.roc (**res.roc، print.auc = TRUE، print.thres = "بهترین")

اختصاصی

در اینجا، بهترین مرز یا آستانه احتمال 0.335 است که منجر به یک طبقه‌بندی پیش‌بینی‌کننده با ویژگی 0.84 و حساسیت 0.660 می‌شود.

توجه داشته باشید که print.thres همچنین می‌تواند یک بردار عددی باشد که حاوی تعریف مستقیم آستانه‌هایی برای نمایش است:

**plot.roc (**res.roc، print.thres = **c (**0.3، 0.5، 0.7))

29.7.3 منحنی‌های ROC چندگانه

اگر متغیرهای گروه‌بندی در داده‌های خود دارید، ممکن است بخواهید چندین منحنی ROC را در یک نمودار ایجاد کنید. این را می‌توان با استفاده از ggplot2 انجام داد.

* *یک متغیر گروه‌بندی ایجاد کنید*

glucose <- **ifelse (**test.data **$** glucose **<** 127.5، "glu.low"، "glu.high")

سن <- **ifelse (**test.data **$** سن **<** 28.5، "جوان"، "پیر")

roc.data <- roc.data °/ o >°/o

**فیلتر (**آستانه‌ها **!=-** Inf) **0 / 0 > 0 / 0**

**جهش (**گلوکز = گلوکز، سن = سن)

* *منحنی ROC را ایجاد کنید*

**ggplot (**roc.data، **aes (**ویژگی، حساسیت)) **+**

**geom\_path (** **aes (**رنگ = سن)) **+**

**scale\_x\_reverse (**expand = **c (**0,0)) **+**

**scale\_y\_continuous (**expand = **c (**0,0)) **+**

**geom\_abline (**تقاطع = 1، شیب = 1، نوع خط = "خطوط") **+ theme\_bw (**)

29.8 تنظیمات چند کلاسه

ما با ساخت یک مدل تشخیص خطی با استفاده از مجموعه داده زنبق که شامل طول و عرض کاسبرگ‌ها و گلبرگ‌ها برای سه گونه زنبق است، شروع می‌کنیم. ما می‌خواهیم گونه‌ها را بر اساس پارامترهای کاسبرگ و گلبرگ با استفاده از LDA پیش‌بینی کنیم.

* *داده‌ها را بارگذاری کنید*

**داده (**"زنبق")

* *تقسیم داده‌ها به آموزش (80%) و مجموعه تست (20%)*

**set.seed (**123)

training.samples <- iris$Species °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE)

train.data <- irisftraining.samples، ]

test.data <- irisf **-** training.samples، ]

* *مدل را روی مجموعه قطار بسازید*

**کتابخانه (**MASS)

مدل <- **lda (**گونه **~** .، داده = train.data)

معیارهای عملکرد (حساسیت، ویژگی، ...) مدل پیش‌بینی را می‌توان به‌طور ­جداگانه برای هر کلاس محاسبه کرد و هر سطح عامل را با سطوح باقی مانده مقایسه کرد (یعنی رویکرد "یک در مقابل همه").

* *روی داده‌های آزمون پیش‌بینی کنید*

پیش‌بینی <- مدل 0 / 0 > 0 / 0 **پیش‌بینی (**test.data)

* *دقت مدل*

**confusionMatrix (**پیش‌بینی کلاس **$،** test.data **$** گونه)

* *#*  ماتریس درهم‌ریختگی و آمار

##

* # مرجع

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ## | پیش‌بینی | ستوزا | رنگارنگ | ویرجینیکا |
| ## | ستوزا | 10 | 0 | 0 |
| ## | رنگارنگ | 0 | 10 | 0 |
| ## | ویرجینیکا | 0 | 0 | 10 |

* #
* # آمار کلی
* #
* # دقت: 1
* # 95% CI: (0.884, 1)
* # بدون اطلاعات نرخ : 0.333
* # P-Value [Acc > NIR]: 4.86e-15
* #
* # کاپا: 1
* # P-Value آزمون مک نمار: NA
* #
* # آمار بر اساس کلاس:
* #
* # کلاس: setosa کلاس: versicolor کلاس: virginica

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ## | حساسیت | 1000 | 1000 | 1000 |
| ## | اختصاصی | 1000 | 1000 | 1000 |
| ## | Pos Pred Value | 1000 | 1000 | 1000 |
| ## | Neg Pred Value | 1000 | 1000 | 1000 |
| ## | شیوع | 0.333 | 0.333 | 0.333 |
| ## | میزان تشخیص | 0.333 | 0.333 | 0.333 |
| ## | شیوع تشخیص | 0.333 | 0.333 | 0.333 |
| ## | دقت متعادل | 1000 | 1000 | 1000 |

توجه داشته باشید که منحنی‌های ROC معمولاً در طبقه‌بندی باینری استفاده می‌شوند اما برای مسائل طبقه‌بندی چند کلاسه استفاده نمی‌شوند.

29.9 بحث

این فصل معیارهای مختلفی را برای ارزیابی عملکرد مدل‌های طبقه‌بندی توضیح داد. این معیارها عبارتند از:

* صحت طبقه‌بندی،
* ماتریس درهم‌ریختگی،
* دقت، یادآوری و خاصیت،
* و منحنی ROC

برای ارزیابی عملکرد مدل‌های رگرسیون، فصل 12 را مطالعه کنید.

قسمت هفتم

یادگیری ماشین آماری

مقدمه

**یادگیری ماشین آماری** به مجموعه‌ای از الگوریتم‌های خودکار قدرتمند اشاره دارد که برای پیش‌بینی یک متغیر نتیجه بر اساس چندین متغیر پیش‌بینی‌کننده استفاده می‌شوند. الگوریتم‌ها به‌طور خودکار عملکرد خود را از طریق «یادگیری» از داده‌ها بهبود می‌بخشند، یعنی داده‌محور هستند و به دنبال تحمیل ساختار خطی یا کلی دیگر بر داده‌ها نیستند (بروس و بروس، 2017). این بدان معنی است که آنها ناپارامتریک هستند.

روش‌های مختلف یادگیری ماشین را می‌توان برای هر دو مورد استفاده قرار داد:

* **طبقه‌بندی که** در آن متغیر نتیجه یک متغیر طبقه‌بندی است، برای مثال مثبت در مقابل منفی
* و **رگرسیون که** در آن متغیر نتیجه یک متغیر پیوسته است.

در این قسمت به روش‌های زیر می‌پردازیم:

* **K-نزدیکترین همسایه** ها که نتیجه یک مشاهدات جدید x را به‌عنوان میانگین نتیجه k شبیه ترین مشاهدات به x پیش‌بینی می‌کنند (فصل 31).
* **درختان تصمیم که** مجموعه‌ای از قوانین تصمیم را می‌سازند که رابطه بین پیش‌بینی کننده‌ها و نتیجه را توصیف می‌کند. این قوانین برای پیش‌بینی نتیجه ­مشاهدات جدید استفاده می‌شود (فصل 32).
* **یادگیری گروهی،** از جمله **بسته‌بندی،** **جنگل تصادفی** و **تقویت** . این الگوریتم‌های یادگیری ماشین بر اساس درخت‌های تصمیم‌گیری هستند. آنها مدل‌های درختی زیادی را از مجموعه داده‌های آموزشی تولید می‌کنند و از میانگین به‌عنوان مدل پیش‌بینی کننده استفاده می‌کنند. این نتایج به تکنیک‌های مدل‌سازی پیش‌بینی با عملکرد برتر می‌شود. به فصل 33 و 34 مراجعه کنید.

KNN - k-نزدیکترین همسایه ها

* 1. مقدمه

الگوریتم **k نزدیکترین همسایه (** **KNN** ) یک روش *یادگیری ماشین ساده* است که برای طبقه‌بندی و رگرسیون استفاده می‌شود. الگوریتم kNN نتیجه یک مشاهده جدید را با مقایسه آن با k مورد مشابه در مجموعه داده‌های آموزشی که در آن k توسط تحلیلگر تعریف می‌شود، پیش‌بینی می‌کند.

در این فصل، ما با تشریح اصول اولیه الگوریتم KNN برای تنظیمات طبقه‌بندی و رگرسیون شروع می‌کنیم. در مرحله بعد، مثال عملی در *R* برای تهیه داده‌ها و محاسبه مدل KNN ارائه می‌دهیم.

علاوه بر این، شما یاد خواهید گرفت که چگونه پیش‌بینی کنید و عملکرد مدل ساخته شده را در پیش‌بینی نتیجه مشاهدات آزمایشی جدید ارزیابی کنید.

* 1. الگوریتم KNN
* الگوریتم KNN برای طبقه‌بندی :

برای طبقه‌بندی یک مشاهده جدید (new\_obs)، روش k-نزدیک‌ترین همسایه‌ها با شناسایی ­k شبیه‌ترین مشاهدات آموزشی (یعنی همسایه‌ها) به new\_obs ما شروع می‌شود و سپس new\_obs را به کلاسی که اکثر همسایگانش را در بر می‌گیرد اختصاص می‌دهد.

* الگوریتم KNN برای رگرسیون :

به‌طور مشابه، برای پیش‌بینی یک مقدار نتیجه پیوسته برای مشاهدات جدید داده شده (new\_obs)، الگوریتم KNN میانگین مقدار نتیجه مشاهدات آموزشی k را که شبیه‌ترین به new\_obs هستند محاسبه می‌کند و این مقدار را به‌عنوان مقدار نتیجه پیش‌بینی‌شده new\_obs برمی‌گرداند.

* اقدامات مشابه :

توجه داشته باشید که (ناهمسانی) بین مشاهدات عموماً با استفاده از اندازه‌گیری فاصله اقلیدسی تعیین [[8]](#footnote-8)می‌شود که به مقیاسی که اندازه‌گیری‌های متغیر پیش‌بینی‌کننده روی آن انجام می‌شود بسیار حساس است. بنابراین، به‌طور کلی توصیه می‌شود که متغیرهای پیش‌بینی‌کننده را استاندارد (یعنی نرمال‌سازی) کنید تا مقیاس‌های آنها قابل مقایسه باشد.

بخش‌های زیر نشان می‌دهد که چگونه می‌توان یک مدل پیش‌بینی k-نزدیک‌ترین همسایه برای تنظیمات طبقه‌بندی و رگرسیون ساخت.

* 1. بارگیری بسته‌های R مورد نیاز
* آیه مرتب برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید

**کتابخانه (**tidyverse)

**کتابخانه (**کارت)

* 1. طبقه‌بندی
     1. نمونه‌ای از مجموعه داده ها

مجموعه داده ها: PimaIndiansDiabetes2 [در بسته mlbench ]، معرفی شده در فصل 20، برای پیش‌بینی ­احتمال مثبت بودن دیابت بر اساس متغیرهای بالینی متعدد.

ما داده‌ها را به صورت تصادفی به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم می‌کنیم. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *داده‌ها را بارگذاری کنید و NA‌ها را حذف کنید*

**داده (**"PimaIndiansDiabetes2"، بسته = "mlbench")

PimaIndiansDiabetes2 <- **na.omit (**PimaIndiansDiabetes2)

* *داده‌ها را بررسی کنید*

**sample\_n (**Pima!ndiansDiabetes2, 3)

* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**123)

training.samples <- PimaIndiansDiabetes2$diabetes °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE)

train.data <- PimaIndiansDiabetes2[training.samples, ]

test.data <- PimaIndiansDiabetes2[ **-** training.samples, ]

* + 1. محاسبات طبقه‌بندی کننده KNN

ما از بسته caret استفاده می‌کنیم که به‌طور خودکار مقادیر مختلف k ممکن را آزمایش می‌کند، سپس k بهینه را انتخاب می‌کند که خطای اعتبارسنجی متقاطع ("cv") را به حداقل می‌رساند و بهترین مدل KNN نهایی را که بهترین داده‌های ما را توضیح می‌دهد مطابقت می‌دهد.

علاوه بر این caret می‌تواند به‌طور خودکار داده‌ها را به منظور نرمال‌سازی متغیرهای پیش‌بینی‌کننده، پیش پردازش کند.

از آرگومان‌های زیر در تابع train() استفاده می‌کنیم :

* trControl، برای تنظیم اعتبار متقاطع 10 برابری
* پردازش، برای نرمال‌سازی داده ها
* tuneLength، برای تعیین تعداد k مقادیر ممکن برای ارزیابی
* *مدل را روی مجموعه آموزشی قرار دهید*

**set.seed (**123)

مدل <- **قطار (**

دیابت **~** .، داده = train.data، روش = "knn"،

trControl = **trainControl (**"cv"، عدد = 10)،

preProcess = **c (**"مرکز"، "مقیاس")، طول = 20

)

* *دقت مدل پلات در مقابل مقادیر مختلف K* **نمودار (**مدل)

#همسایه ها

* *بهترین پارامتر تنظیم k را چاپ کنید که # دقت* مدل را به حداکثر می‌رساند **. $** bestTune
* # k
* # 5 13
* *روی داده‌های آزمون پیش‌بینی کنید*

predicted.classes <- model °/ o >°/o **predict (**test.data) **head (**predicted.classes)

* # [1] neg pos neg pos pos neg
* # سطوح: پوز منفی
* *محاسبه میزان دقت مدل*

**میانگین (**predicted.classes **==** test.data **$** دیابت)

* # [1] 0.769

صحت پیش‌بینی کلی مدل ما 76.9٪ است که خوب است (برای یادگیری معیارهای کلیدی مورد استفاده برای ارزیابی عملکرد مدل طبقه‌بندی، به فصل 29 مراجعه کنید).

31.5 KNN برای رگرسیون

در این بخش، نحوه پیش‌بینی یک متغیر پیوسته با استفاده از KNN را شرح خواهیم داد.

ما از مجموعه داده‌های بوستون [در بسته MASS ]، معرفی شده در فصل 2، برای پیش‌بینی مقدار خانه متوسط ( mdev )، در حومه بوستون، با استفاده از متغیرهای پیش‌بینی کننده مختلف استفاده خواهیم کرد.

1. **به‌طور تصادفی داده‌ها** را به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم کنید. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *داده‌ها را بارگذاری کنید*

**داده (**"بوستون"، بسته = "MASS")

* *داده‌ها را بررسی کنید*

**sample\_n (**بوستون، 3)

* *تقسیم داده‌ها به مجموعه آموزشی و تست* **set.seed (**123)

training.samples <- Boston$medv °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE) train.data <- Bostonftraining.samples، ] test.data <- Bostonf **-** training.samples، ]

2. محاسبه KNN با استفاده از caret .

بهترین k آنی است که خطای پیش‌بینی RMSE (ریشه میانگین مربعات خطا) را به حداقل برساند.

RMSE مربوط به ریشه دوم تفاوت میانگین بین مقادیر نتیجه شناخته شده مشاهده شده و مقادیر پیش‌بینی شده است . هرچه RMSE کمتر باشد، مدل بهتر است.

* *مدل را در مجموعه آموزشی قرار* **دهید set.seed (**123) model <- **train (**

medv **~** .، داده = train.data، روش = "knn"، trControl = **trainControl (**"cv"، عدد = 10)، preProcess = **c (**"مرکز"، "مقیاس")، طول = 10)

* *خطای مدل رسم RMSE در مقابل مقادیر مختلف k* **نمودار (**مدل)
* *بهترین پارامتر تنظیم k که* مدل RMSE **$** bestTune را به حداقل می‌رساند
* *روی داده‌های آزمون پیش‌بینی کنید*

پیش‌بینی <- مدل 0 / 0 > 0 / 0 **پیش‌بینی (**test.data) **head (**پیش‌بینی)

* *خطای پیش‌بینی RMSE را محاسبه کنید*

**RMSE (**پیش‌بینی‌ها، test.data **$** medv)

31.6 بحث

این فصل اصول مدل‌سازی KNN (k-نزدیک‌ترین همسایگان) را شرح می‌دهد که از نظر مفهومی، یکی از روش‌های یادگیری ماشین ساده‌تر است.

توصیه می‌شود هنگام انجام تحلیل KNN، داده‌ها را استاندارد کنید. ما کدهای R را برای محاسبه آسان مدل پیش‌بینی KNN و ارزیابی عملکرد مدل بر روی داده‌های تست ارائه کردیم.

هنگام برازش الگوریتم KNN، تحلیلگر باید تعداد همسایگان (k) را که باید در الگوریتم KNN برای پیش‌بینی نتیجه یک مشاهده در نظر گرفته شود، مشخص کند. انتخاب k به‌طور قابل‌توجهی بر خروجی KNN تأثیر می‌گذارد. k = 1 مربوط به یک روش بسیار انعطاف‌پذیر است که منجر به نرخ خطای آموزشی 0 (بیش برازش) می‌شود، اما میزان خطای آزمون ممکن است بسیار بالا باشد.

برای تعیین مقدار بهینه برای داده‌های خود، باید چندین k-value را آزمایش کنید. این را می‌توان به‌طور خودکار با استفاده از بسته caret انجام داد که مقدار k را انتخاب می‌کند که خطای اعتبارسنجی متقاطع 13 را به حداقل می‌رساند.

فصل 32

مدل‌های درخت تصمیم

* 1. مقدمه

روش **درخت تصمیم** یک تکنیک یادگیری ماشین قدرتمند و محبوب است که هم برای *طبقه‌بندی* و هم برای *رگرسیون استفاده می‌*شود. بنابراین، آن را به‌عنوان **درختان طبقه‌بندی و رگرسیون (** **CART** ) نیز می‌شناسند.

توجه داشته باشید که پیاده‌سازی R الگوریتم CART *RPART (*Recursive Partition ­ing And Regression Trees) نامیده می‌شود که در بسته‌ای به همین نام موجود است.

در این فصل، اصول اولیه مدل‌های درختی را شرح می‌دهیم و کدهای R را برای محاسبه طبقه‌بندی ­و درختان رگرسیون ارائه می‌کنیم.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید
* rpart برای محاسبه مدل‌های درخت تصمیم

**کتابخانه (**tidyverse)

**کتابخانه (**کارت)

**کتابخانه (**rpart)

* 1. الگوریتم درخت تصمیم
     1. مبانی و نمایش بصری

الگوریتم مدل‌های درخت تصمیم با تقسیم مکرر داده‌ها به فضاهای فرعی متعدد کار می‌کند، به‌طوری که نتایج در هر زیرفضای نهایی تا حد امکان همگن باشد. این رویکرد از نظر فنی *پارتیشن‌بندی بازگشتی نامیده می‌شود* .

نتیجه تولید شده شامل مجموعه‌ای از قوانین است که برای پیش‌بینی متغیر نتیجه استفاده می‌شود که می‌تواند یکی از موارد زیر باشد:

* یک متغیر پیوسته، برای درختان رگرسیون
* یک متغیر طبقه‌بندی، برای درختان طبقه‌بندی

قوانین تصمیم‌گیری تولید شده توسط مدل پیش‌بینی CART عموماً به صورت یک درخت باینری تجسم می‌شوند.

مثال زیر مدل درختی را نشان می‌دهد که گونه‌های گل زنبق را بر اساس طول (بر حسب سانتی متر) و عرض کاسبرگ و گلبرگ پیش‌بینی می‌کند.

**کتابخانه (**rpart)

مدل <- **rpart (**گونه **~** .، داده = زنبق)

**par (**xpd = NA) *# در غیر این صورت، در برخی دستگاه‌ها متن بریده می‌شود*

**طرح (**مدل)

**متن (**مدل، ارقام = 3)

گلبرگ. طول< 2.45

گلبرگ.پهلو کمتر از 1.75 ستوزا

virginica

رنگارنگ

نمودار قواعد مختلف تقسیم ممکن را نشان می‌دهد که می‌تواند برای پیش‌بینی موثر نوع نتیجه (در اینجا، گونه‌های زنبق) استفاده شود. به‌عنوان مثال، تقسیم بالا مشاهداتی با طول گلبرگ <2.45 به شاخه سمت چپ اختصاص می‌دهد، جایی که گونه‌های پیش‌بینی شده setosa هستند .

قوانین مختلف در درخت را می‌توان به صورت زیر چاپ کرد:

**چاپ (**مدل، ارقام = 2) ## n= 150 ##

* # گره)، تقسیم، n، از دست دادن، yval، (yprob)
* # \* نشان دهنده گره ترمینال است
* #
* # 1) ریشه 150 100 ستوزا (0.333 0.333 0.333)
* # 2) گلبرگ. طول< 2.5 50 0 ستوزا (1.000 0.000 0.000) \*
* # 3) گلبرگ. طول>=2.5 100 50 versicolor (0.000 0.500 0.500)
* # 6) گلبرگ. عرض< 1.8 54 5 versicolor (0.000 0.907 0.093) \*
* # 7) Petal.Width>=1.8 46 1 virginica (0.000 0.022 0.978) \*

این قوانین با تقسیم مکرر متغیرهای پیش‌بینی کننده، با متغیری شروع می‌شود که بیشترین ارتباط را با متغیر پاسخ دارد. این روند تا زمانی ادامه می‌یابد که برخی از معیارهای توقف از پیش تعیین شده برآورده شوند.

درخت حاصل از *گره‌های تصمیم* , *شاخه‌ها* و *گره‌های برگ تشکیل شده* است . درخت از وارونه به پایین قرار می‌گیرد، بنابراین *ریشه* در بالا و برگ‌ها نشان دهنده نتیجه در پایین قرار می‌گیرند.

هر گره تصمیم مربوط به یک متغیر پیش‌بینی کننده ورودی و یک برش تقسیم بر روی آن متغیر است. گره‌های برگ درخت متغیر نتیجه هستند که برای پیش‌بینی استفاده می‌شوند. درخت از بالا (ریشه) رشد می‌کند، در هر گره الگوریتم بهترین برش تقسیم را تعیین می‌کند که منجر به بیشترین خلوص (یا همگنی) در هر بخش فرعی می‌شود.

رشد درخت با سه معیار زیر متوقف می‌شود (ژانگ، 2016):

1. تمام گره‌های برگ خالص با یک کلاس واحد هستند.
2. حداقل تعداد از پیش تعیین شده مشاهدات آموزشی که نمی‌توان به هر گره برگ با هیچ روش تقسیم‌بندی اختصاص داد.
3. تعداد مشاهدات در گره برگ به حداقل یک مورد از پیش تعیین شده می‌رسد.

یک درخت کاملاً رشد کرده با داده‌های آموزشی بیش برازش دارد و مدل حاصل ممکن است برای پیش‌بینی نتیجه داده‌های آزمایش جدید کارایی نداشته باشد. برای کنترل این مشکل از تکنیک‌هایی مانند **هرس استفاده می‌شود.**

* + 1. انتخاب نقاط تقسیم درختان

فنی، **برای مدل‌سازی رگرسیون،** برش تقسیم به‌گونه‌ای تعریف می‌شود که مجموع مجذور خطا (RSS) در نمونه‌های آموزشی که در زیربخش قرار می‌گیرند به حداقل برسد. به یاد بیاورید که RSS مجموع اختلاف مجذور بین مقادیر نتیجه مشاهده شده و مقادیر پیش‌بینی شده است، RSS = sum((مشاهده - پیش‌بینی شده)~2) . به فصل 3 مراجعه کنید

**در تنظیمات طبقه‌بندی،** نقطه تقسیم به گونه‌ای تعریف می‌شود که جمعیت در پارتیشن‌ها تا حد امکان خالص باشد. دو معیار خلوص به‌طور کلی استفاده می‌شود، از جمله *شاخص جینی* و *آنتروپی (*یا *افزایش اطلاعات* ).

برای یک زیربخش معین، Gini = مجموع(p(lp)) و آنتروپی = -l\*sum(p\*log(p)) که p نسبت مشاهدات طبقه‌بندی نادرست درون پارتیشن است.

مجموع در بین دسته‌ها یا کلاس‌های مختلف در متغیر نتیجه محاسبه می‌شود. شاخص جینی و آنتروپی از 0 (بزرگترین خلوص) تا 1 (حداکثر درجه ناخالصی) متغیر است.

* + 1. پیش‌بینی

مجموعه قوانین مختلف ایجاد شده در درخت برای پیش‌بینی نتیجه یک داده آزمایش جدید استفاده می‌شود. کد R زیر گونه‌های گل زنبق جمع‌آوری شده جدید را پیش‌بینی می‌کند: newdata <- **data.frame** f

Sepal.Length = 6.5، Sepal.Width = 3.0،

Petal.Length =5.2، Petal.Width =2.0 ) model **/,>/. پیش‌بینی (**داده‌های جدید، "کلاس") ## l ## virginica ## سطوح: setosa versicolor virginica

پیش‌بینی می‌شود داده‌های جدید ویرجینیکا باشد.

32.4 درختان طبقه‌بندی

* + 1. نمونه‌ای از مجموعه داده ها

مجموعه داده ها: PimaIndiansDiabetes2 [در بسته mlbench ]، معرفی شده در فصل 20، برای پیش‌بینی ­احتمال مثبت بودن دیابت بر اساس متغیرهای بالینی متعدد.

ما داده‌ها را به صورت تصادفی به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم می‌کنیم. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *داده‌ها را بارگذاری کنید و NA‌ها را حذف کنید*

**داده (**"PimaIndiansDiabetes2"، بسته = "mlbench") PimaIndiansDiabetes2 <- **na.omit (**PimaIndiansDiabetes2) *# نمونه داده را بازرسی کنید (*Pima!ndiansDiabetes2, 3 **)**

* *تقسیم داده‌ها به مجموعه آموزشی و تست* **set.seed (**123)

training.samples <- PimaIndiansDiabetes2$diabetes °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE)

train.data <- PimaIndiansDiabetes2[training.samples, ] test.data <- PimaIndiansDiabetes2[ **-** training.samples, ]

* + 1. درختان کاملا رشد کرده

در اینجا، ما یک درخت کاملاً رشد یافته ایجاد خواهیم کرد که همه متغیرهای پیش‌بینی کننده در مجموعه داده را نشان می‌دهد.

* *مدل را بسازید*

**set.seed (**123)

modell <- **rpart (**دیابت **~** .، داده = train.data، روش = "کلاس") *# درختان را ترسیم کنید*

**par (**xpd = NA) *# از برش متن در برخی دستگاه‌ها خودداری کنید*

**طرح (**مدل)

**متن (**مدل، ارقام = 3)

گلوکز<127.5

pedigree< 0.6345

| pregngnt< 7.5

neg

neg pos

age< 23.5

glucose< 165.5

pos

neg

pedigree< 0.7245 pedigree>=0.6l6

pressu'e>=71

pos

neg

pressure< 77

pos

پوز نفی

* *پیش‌بینی‌ها روی داده‌های آزمایشی* پیش‌بینی می‌شود. کلاس‌ها <- modell °/ o >°/o

**پیش‌بینی (**test.data, type = "class") **head (**predicted.classes) ## 2l 25 28 29 32 36

* # نِگ پو نِگ پوس نِگ
* # سطوح: پوز منفی
* *محاسبه میزان دقت مدل بر روی* **میانگین داده‌های آزمایش (**predicted.classes **==** test.data **$** دیابت)
* # Cl] 0.782

صحت کلی مدل درختی ما 78 درصد است که چندان بد نیست.

با این حال، به نظر می‌رسد این درخت کامل شامل همه پیش‌بینی‌کننده‌ها بسیار پیچیده است و در شرایطی که مجموعه داده‌های بزرگی با پیش‌بینی‌کننده‌های متعدد دارید، تفسیر آن دشوار است.

علاوه بر این، به راحتی می‌توان دریافت که یک درخت کاملاً رشد کرده با داده‌های آموزشی بیش برازش دارد و ممکن است منجر به عملکرد ضعیف مجموعه تست شود.

یک استراتژی برای محدود کردن این بیش برازش این است که درخت را هرس کنید و در نتیجه درختی ساده‌تر با شکاف‌های کمتر و تفسیر بهتر به قیمت کمی بایاس ایجاد کنید (جیمز و همکاران، 2014، بروس و بروس (2017)).

* + 1. هرس درخت

به‌طور خلاصه، هدف ما در اینجا این است که ببینیم آیا یک زیردرخت کوچکتر می‌تواند نتایج قابل مقایسه‌ای را با درخت کاملاً رشد یافته به ما بدهد. اگر بله، باید به سراغ درخت ساده‌تر برویم زیرا احتمال بیش برازش را کاهش می‌دهد.

یکی از استراتژی‌های قوی ممکن برای هرس درخت (یا متوقف کردن رشد درخت) شامل پرهیز از تقسیم یک پارتیشن است اگر تقسیم به‌طور قابل‌توجهی کیفیت کلی مدل را بهبود نبخشد.

در بسته rpart، این امر توسط *پارامتر پیچیدگی (*cp) کنترل می‌شود که جریمه‌ای را به درخت برای داشتن دو تقسیم زیاد اعمال می‌کند. مقدار پیش فرض 0.01 است. هر چه cp بالاتر باشد درخت کوچکتر است.

مقدار خیلی کم cp منجر به بیش برازش می‌شود و مقدار زیاد cp منجر به درخت خیلی کوچک می‌شود. هر دو مورد عملکرد پیش‌بینی مدل را کاهش می‌دهند.

cp بهینه را می‌توان با آزمایش مقادیر مختلف cp و با استفاده از روش‌های اعتبارسنجی متقابل برای تعیین صحت پیش‌بینی مربوطه مدل تخمین زد. سپس بهترین cp به‌عنوان چیزی تعریف می‌شود که دقت اعتبارسنجی متقاطع را به حداکثر می‌رساند (فصل 13).

هرس را می‌توان به راحتی در گردش کار بسته caret انجام داد که از روش rpart برای آزمایش خودکار مقادیر مختلف cp استفاده می‌کند، سپس cp بهینه را انتخاب می‌کند که دقت اعتبار متقاطع ("cv") را به حداکثر می‌رساند و بهترین مدل CART نهایی را متناسب می‌کند. که بهترین داده‌های ما را توضیح می‌دهد.

می توانید از آرگومان‌های زیر در تابع trainO [از بسته caret] استفاده کنید:

* trControl، برای تنظیم اعتبار متقاطع 10 برابری
* tuneLength، برای تعیین تعداد مقادیر cp ممکن برای ارزیابی. مقدار پیش فرض 3 است، در اینجا از 10 استفاده می‌کنیم.
* *مدل را در مجموعه آموزشی قرار دهید.seed (*123) model2 < **- train (**

دیابت **~** .، داده = train.data، روش = "rpart"، trControl = **trainControl (**"cv"، عدد = 10)، طول = 10

)

* *دقت مدل رسم در مقابل مقادیر مختلف*
* *cp (پارامتر پیچیدگی)*

**طرح (**مدل 2)

پارامتر پیچیدگی

*# بهترین پارامتر تنظیم cp را چاپ کنید که # دقت مدل را به حداکثر* می‌رساند model2 **$** bestTune

* # cp
* # 2 0.0321
* *مدل درخت نهایی را رسم کنید*

**par (**xpd = NA) *# از برش متن در برخی دستگاه‌ها خودداری کنید*

**طرح (**model2 **$** finalModel)

**متن (**model2 **$** finalModel، اعداد = 3)

گلوکز<127.5

< 165.5

neg

سن کمتر از 23.5

pos

پوز نفی

* *قوانین تصمیم‌گیری در مدل* model2 **$** finalModel ## n= 314

##

* # گره)، تقسیم، n، از دست دادن، yval، (yprob)
* # \* نشان دهنده گره ترمینال است
* #
* # 1) ریشه 314 104 neg (0.6688 0.3312)
* # 2) گلوکز< 128 188 26 neg (0.8617 0.1383) \*
* # 3) گلوکز>=128 126 48 pos (0.3810 0.6190)
* # 6) گلوکز< 166 88 44 neg (0.5000 0.5000)
* # 12) سن < 23.5 16 1 neg (0.9375 0.0625) \*
* # 13) سن>=23.5 72 29 pos (0.4028 0.5972) \*
* # 7) گلوکز>=166 38 4 pos (0.1053 0.8947) \*
* *پیش‌بینی‌هایی را روی داده‌های آزمایش* پیش‌بینی کنید. کلاس‌ها <- model2 °/ o >°/o **پیش‌بینی (**test.data) *# محاسبه نرخ دقت مدل بر روی میانگین داده‌های آزمون (*predicted.classes **==** test.data **$** دیابت)
* # [1] 0.795

از خروجی بالا، می‌توان دریافت که بهترین مقدار برای پارامتر پیچیدگی (cp) 0.032 است که به درختی ساده تر، تفسیر آسان، با صحت کلی 79% امکان می‌دهد که با صحت (78%) قابل مقایسه است. که با درخت کامل به دست آورده ایم. صحت پیش‌بینی درخت هرس شده در مقایسه با درخت کامل بهتر است.

با هم، باید به سراغ این مدل ساده تر برویم.

32.5 درختان رگرسیون

قبلاً نحوه ساخت یک درخت طبقه‌بندی برای پیش‌بینی گروه (یعنی کلاس) مشاهدات را توضیح دادیم.

در این بخش، نحوه ساخت یک درخت برای پیش‌بینی یک متغیر پیوسته، روشی به نام تحلیل رگرسیون (فصل 2) را شرح خواهیم داد.

کد R با آنچه در بخش‌های قبلی دیده‌ایم یکسان است. هرس نیز باید در اینجا اعمال شود تا بیش برازش محدود شود.

مشابه درخت‌های طبقه‌بندی، کد R زیر از بسته caret برای ساخت درخت‌های رگرسیون و پیش‌بینی خروجی مجموعه داده‌های آزمایشی جدید استفاده می‌کند.

* + 1. نمونه‌ای از مجموعه داده ها

مجموعه داده: ما از مجموعه داده‌های بوستون [در بسته MASS ]، معرفی شده در فصل 2، برای پیش‌بینی ­مقدار خانه متوسط ( mdev )، در حومه بوستون، با استفاده از متغیرهای پیش‌بینی کننده مختلف استفاده خواهیم کرد.

ما داده‌ها را به صورت تصادفی به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم می‌کنیم. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *داده‌ها را بارگذاری کنید*

**داده (**"بوستون"، بسته = "MASS")

* *داده‌ها را بررسی کنید*

**sample\_n (**بوستون، 3)

* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**123)

training.samples <- Boston$medv °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE) train.data <- Bostonftraining.samples، ] test.data <- Bostonf **-** training.samples، ]

* + 1. درخت رگرسیون را ایجاد کنید

در اینجا بهترین مقدار cp مقداری است که خطای پیش‌بینی RMSE (ریشه میانگین مربعات خطا) را به حداقل می‌رساند.

خطای پیش‌بینی توسط RMSE اندازه‌گیری می‌شود که مربوط به میانگین اختلاف بین مقادیر مشاهده‌شده معلوم نتیجه و مقدار پیش‌بینی‌شده توسط مدل است. RMSE به صورت RMSE = mean((مشاهده - پیش‌بینی)~2) %>% sqrt() محاسبه می‌شود. هرچه RMSE کمتر باشد، مدل بهتر است.

بهترین مقدار cp را انتخاب کنید:

* *مدل را روی مجموعه آموزشی قرار دهید*

**set.seed (**123)

مدل <- **قطار (**

medv **~** .، داده = train.data، روش = "rpart"،

trControl = **trainControl (**"cv"، عدد = 10)،

طول کوک = 10

)

* *خطای مدل رسم در مقابل مقادیر مختلف*
* *cp (پارامتر پیچیدگی)*

**طرح (**مدل)

* *بهترین پارامتر تنظیم cp را چاپ کنید*
* *مدل RMSE را به حداقل برسانید*

مدل **$** bestTune

مدل درخت نهایی را ترسیم کنید:

* *مدل درخت نهایی را رسم کنید*

**par (**xpd = NA) *# از برش متن در برخی دستگاه‌ها خودداری کنید*

**نمودار (**مدل **$** finalModel)

**متن (**مدل **$** finalModel، رقم = 3)

lstat>=14.4

rm< 6.974

rm< 7.437

ptratio>=17.6

32.3 38.9 47

lstat>=9.95

nox>=0.607

crim>=11.37 9.81 14.4 18.3

| nox<0.618

20.7 r m< 6.543 |

23.1 28.1 36.6

*# قوانین تصمیم‌گیری در مدل*

مدل **$** finalModel

* *پیش‌بینی داده‌های آزمون را* پیش‌بینی کنید <- مدل °/ o >°/o **پیش‌بینی (**test.data) **head (**predictions)
* *محاسبه خطای پیش‌بینی RMSE* **RMSE (**پیش‌بینی‌ها، test.data **$** medv)

32.6 درخت استنتاج مشروط

درخت استنتاج شرطی (ctree) از روش‌های آزمون معناداری برای انتخاب و تقسیم بازگشتی مرتبط‌ترین متغیرهای پیش‌بین به نتیجه استفاده می‌کند. این می‌تواند بیش برازش را در مقایسه با الگوریتم کلاسیک rpart محدود کند.

در هر مرحله تقسیم، اگر بین متغیرهای پیش‌بینی کننده و متغیر نتیجه وابستگی وجود نداشته باشد، الگوریتم متوقف می‌شود. در غیر این صورت متغیری که بیشترین ارتباط را با نتیجه دارد برای تقسیم انتخاب می‌شود.

درخت شرطی را می‌توان به راحتی با استفاده از گردش کار caret محاسبه کرد که تابع ctree() موجود در بسته party را فراخوانی می‌کند .

1. داده‌های آزمایشی: PimaIndiansDiabetes2 . ابتدا داده‌ها را به آموزش (80%) و مجموعه تست (20%) تقسیم کنید.

* *داده‌ها را بارگذاری کنید*

**داده (**"PimaIndiansDiabetes2"، بسته = "mlbench")

pima.data <- **na.omit (**PimaIndiansDiabetes2)

* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**123)

training.samples <- pima.data$diabetes °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE) train.data <- pima.data[training.samples, ] test.data <- pima.data[ **-** training.samples, ]

1. درخت‌های شرطی را با استفاده از پارامترهای تنظیم حداکثر عمق و حداقل معیار برای کنترل اندازه درخت بسازید. بسته caret مقادیر تنظیم بهینه داده‌های شما را به‌طور خودکار انتخاب می‌کند، اما در اینجا حداکثر عمق و mincriterion را مشخص می‌کنیم .

مثال زیر یک درخت طبقه‌بندی ایجاد می‌کند:

**کتابخانه (**پارتی)

**set.seed (**123)

مدل <- **قطار (**

دیابت **~** .، داده = train.data، روش = "ctree2"، trControl = **trainControl (**"cv"، عدد = 10)، tuneGrid = **expand.grid (**maxdepth = 3، mincriterion = 0.95 )

**نمودار (**مدل **$** finalModel)

Node 4 (n =

**CD ' CD**

- 0.8

- 0.6

- 0.4

0.2 0

*# ساخت، پیش‌بینی‌ها روی داده‌های آزمایشی* پیش‌بینی شد. کلاس‌ها <- مدل 0 / 0 > 0 / 0 **پیش‌بینی (**test.data) *# محاسبه نرخ دقت مدل بر روی میانگین داده‌های آزمون (*predicted.classes **==** test.data **$** دیابت) ## [1] 0.744

p-value ارتباط بین یک متغیر پیش‌بینی‌کننده معین و متغیر نتیجه را نشان می‌دهد ­. به‌عنوان مثال، اولین گره تصمیم در بالا نشان می‌دهد که گلوکز متغیری است که به شدت با دیابت با مقدار ap <0.001 مرتبط است و بنابراین به‌عنوان اولین گره انتخاب می‌شود.

32.7 بحث

این فصل نحوه ایجاد درخت طبقه‌بندی و رگرسیون را در R توضیح می‌دهد. درختان ابزار بصری را ارائه می‌دهند که تفسیر و توضیح آن برای مردم بسیار آسان است.

مدل‌های درختی ممکن است در مقایسه با مدل رگرسیون خطی (فصل 3) بسیار کارآمد باشند، زمانی که روابط بسیار غیرخطی و پیچیده‌ای بین متغیر نتیجه و پیش‌بینی‌کننده‌ها وجود دارد.

با این حال، ساختن تنها یک درخت منفرد از مجموعه داده‌های آموزشی ممکن است به یک مدل پیش‌بینی با عملکرد کمتر منجر شود. یک درخت منفرد ناپایدار است و ساختار ممکن است با تغییرات کوچک در داده‌های آموزشی تغییر یابد.

برای مثال، نقطه تقسیم دقیق یک متغیر پیش‌بینی‌کننده معین و پیش‌بینی‌کننده‌ای که در هر مرحله از الگوریتم انتخاب می‌شود، به شدت به مجموعه داده‌های آموزشی وابسته است. استفاده از داده‌های آموزشی کمی متفاوت ممکن است اولین متغیر را تغییر دهد و ساختار درخت را کاملاً تغییر دهد.

سایر الگوریتم‌های یادگیری ماشین - از جمله *بسته‌بندی،* *جنگل تصادفی* و *تقویت* - می‌توانند برای ساخت چندین درخت مختلف از یک مجموعه داده منفرد استفاده شوند که منجر به عملکرد پیش‌بینی بهتری می‌شود. **اما**، با این روش ها، تفسیرپذیری مشاهده شده برای یک درخت از بین می‌رود. توجه داشته باشید که همه این استراتژی‌های ذکر شده در بالا بر اساس الگوریتم CART هستند. به فصل 33 و 34 مراجعه کنید.

فصل 33

بسته‌بندی و جنگل تصادفی

* 1. مقدمه

در فصل 32، نحوه ساخت درخت تصمیم برای مدل‌سازی پیش‌بینی را شرح داده ایم.

مدل درخت تصمیم استاندارد، CART برای درختان طبقه‌بندی و رگرسیون، تنها یک درخت می‌سازد که سپس برای پیش‌بینی نتیجه مشاهدات جدید استفاده می‌شود. خروجی این استراتژی بسیار ناپایدار است و ساختار درختی ممکن است تحت تأثیر یک تغییر کوچک در مجموعه داده‌های آموزشی قرار گیرد.

جایگزین‌های قدرتمند متفاوتی برای الگوریتم CART کلاسیک وجود دارد، از جمله **بسته بندی،** **جنگل تصادفی** و **تقویت** .

**Bagging** مخفف *bootstrap aggregating* است. این شامل ساخت چندین مدل درخت تصمیم متفاوت از یک مجموعه داده آموزشی با استفاده مکرر از چندین زیرمجموعه بوت استرپ داده و میانگین‌گیری مدل‌ها است. در اینجا، هر درخت به‌طور مستقل برای دیگران ساخته می‌شود. بیشتر در مورد بوت استرپینگ در فصل 14 بخوانید.

**جنگل تصادفی،** یکی از رایج ترین و قدرتمندترین تکنیک‌های یادگیری ماشین است. این نوع خاصی از بسته‌بندی است که برای درختان تصمیم اعمال می‌شود.

در مقایسه با مدل استاندارد CART (فصل 32)، جنگل تصادفی یک پیشرفت قوی را ارائه می‌کند که شامل اعمال بسته‌بندی به داده‌ها و نمونه‌گیری بوت استرپ برای متغیرهای پیش‌بینی‌کننده **در هر تقسیم** است (جیمز و همکاران، 2014، بروس و بروس (2017) )). این بدان معناست که در هر مرحله تقسیم الگوریتم درخت، یک نمونه تصادفی از n پیش‌بینی‌کننده به‌عنوان کاندیدهای تقسیم از مجموعه کامل پیش‌بینی‌کننده‌ها انتخاب می‌شود.

جنگل تصادفی می‌تواند هم برای طبقه‌بندی (پیش‌بینی یک متغیر طبقه‌بندی) و هم برای رگرسیون (پیش‌بینی یک متغیر پیوسته) استفاده شود.

در این فصل، نحوه محاسبه الگوریتم جنگل تصادفی در R برای ساخت یک ­مدل پیش‌بینی قدرتمند را شرح خواهیم داد. علاوه بر این، یاد خواهید گرفت که چگونه متغیرهای پیش‌بینی کننده را بر اساس اهمیت آنها در کمک به دقت مدل رتبه‌بندی کنید.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید
* randomForest برای محاسبه الگوریتم جنگل تصادفی

**کتابخانه (**tidyverse) **کتابخانه (**caret) **کتابخانه (**RandomForest)

* 1. طبقه‌بندی
     1. نمونه‌ای از مجموعه داده ها

مجموعه داده ها: PimaIndiansDiabetes2 [در بسته mlbench ]، معرفی شده در فصل 20، برای پیش‌بینی ­احتمال مثبت بودن دیابت بر اساس متغیرهای بالینی متعدد.

به‌طور تصادفی داده‌ها را به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم کنید. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *داده‌ها را بارگذاری کنید و NA‌ها را حذف کنید*

**داده (**"PimaIndiansDiabetes2"، بسته = "mlbench")

PimaIndiansDiabetes2 <- **na.omit (**PimaIndiansDiabetes2)

* *داده‌ها را بررسی کنید*

**sample\_n (**Pima!ndiansDiabetes2, 3)

* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**123)

training.samples <- PimaIndiansDiabetes2$diabetes °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE)

train.data <- PimaIndiansDiabetes2[training.samples, ]

test.data <- PimaIndiansDiabetes2[ **-** training.samples, ]

* + 1. محاسبه طبقه‌بندی جنگل تصادفی

ما از گردش کار caret استفاده می‌کنیم که تابع randomforest() [randomForest pack ­age] را فراخوانی می‌کند تا به‌طور خودکار تعداد بهینه ( mtry ) از متغیرهای پیش‌بینی‌کننده را که به‌طور تصادفی به‌عنوان نامزد در هر تقسیم نمونه‌گیری شده‌اند، انتخاب کنیم و بهترین مدل جنگل تصادفی نهایی را برازش کنیم. بهترین داده‌های ما را توضیح می‌دهد.

از آرگومان‌های زیر در تابع train() استفاده می‌کنیم :

• trControl، برای تنظیم اعتبار متقاطع 10 برابری

* *مدل را در مجموعه آموزشی قرار* **دهید set.seed (**123) model <- **train (**

دیابت **~** .، داده = train.data، روش = "rf"،

trControl = **trainControl (**"cv"، عدد = 10)، اهمیت = TRUE

)

* *بهترین* مدل پارامتر تنظیم **$** bestTune
* # متری
* # 3 8
* *مدل نهایی*

مدل **$** finalModel

##

*##* تماس بگیرید:

* # randomForest(x = x، y = y، mtry = param$mtry، اهمیت = TRUE)
* # نوع جنگل تصادفی: طبقه‌بندی
* # تعداد درخت : 500
* # تعداد متغیرهای امتحان شده در هر کدام، تقسیم: 8

##

* # برآورد OOB از میزان خطا: 22٪
* # ماتریس درهم‌ریختگی:
* # neg pos class.error
* # neg 185 25 0.119
* # pos 44 60 0.423
* *روی داده‌های آزمون پیش‌بینی کنید*

predicted.classes <- model **%>% predict (**test.data) **head (**predicted.classes)

* # [1] نگ پوس نگ نگر پوس نگ
* # سطوح: پوز منفی
* *محاسبه میزان دقت مدل*

**میانگین (**predicted.classes **==** test.data **$** دیابت)

* # [1] 0.808

به‌طور پیش فرض، 500 درخت آموزش داده می‌شوند. تعداد بهینه متغیرهای نمونه برداری شده در هر تقسیم 8 است.

هر درخت بسته‌بندی شده از حدود دو سوم مشاهدات استفاده می‌کند. یک سوم باقی مانده از مشاهداتی که برای برازش با یک درخت بسته‌بندی شده مورد استفاده قرار نگرفته اند، مشاهدات خارج از کیسه (OOB) نامیده می‌شوند (جیمز و همکاران، 2014).

برای یک درخت معین، خطای خارج از کیسه (OOB) خطای مدل در پیش‌بینی داده‌های باقی مانده از مجموعه آموزشی آن درخت است (بروس و بروس، 2017). OOB روشی بسیار ساده برای تخمین خطای تست یک مدل بسته‌بندی شده، بدون نیاز به انجام اعتبارسنجی متقاطع یا رویکرد مجموعه اعتبار سنجی است.

در مثال ما، برآورد OBB از میزان خطا 24٪ است.

صحت پیش‌بینی در داده‌های آزمایش جدید 79 درصد است که خوب است.

33.3.3 اهمیت متغیر

اهمیت هر متغیر را می‌توان با استفاده از تابع important() [randomForestpack] چاپ کرد:

**اهمیت (**مدل **$** finalModel)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ## | | نفی | | | pos | MeanDecreaseAccuracy MeanDecreaseGini | | | |
| ## | حامله | 11 | .57 | 0. | .318 | 10. | .36 | 8. | .86 |
| ## | گلوکز | 38 | .93 | 28. | .437 | 46. | .17 | 53. | .30 |
| ## | فشار | -1 | .94 | 0. | .846 | -1. | .06 | 8. | .09 |
| ## | سه سر | 6 | .19 | 3. | .249 | 6. | .85 | 9. | .92 |
| ## | انسولین | 8 | .65 | -2. | 0.037 | 6. | .01 | 12. | .43 |
| ## | جرم | 7 | .71 | 2. | .299 | 7. | .57 | 14. | .58 |
| ## | شجره نامه | 6 | .57 | 1. | 0.083 | 5. | .66 | 14. | .50 |

## age 9.51 12.310

15.75 16.76

نتیجه نشان می‌دهد:

* MeanDecreaseAccuracy که میانگین کاهش دقت مدل در پیش‌بینی نتیجه نمونه‌های خارج از کیسه زمانی است که یک متغیر خاص از مدل حذف می‌شود.
* MeanDecreaseGini که میانگین کاهش ناخالصی گره است که از تقسیم بر روی آن متغیر حاصل می‌شود. شاخص ناخالصی جینی فقط برای مشکل طبقه‌بندی استفاده می‌شود. در رگرسیون ناخالصی گره با RSS مجموعه آموزشی اندازه‌گیری می‌شود. این معیارها که با استفاده از مجموعه آموزشی محاسبه می‌شوند، نسبت به اندازه‌گیری که بر روی داده‌های خارج از کیسه محاسبه می‌شود، قابل اعتمادتر هستند. برای اندازه‌گیری ناخالصی گره (شاخص جینی و RSS) به فصل 32 مراجعه کنید.

توجه داشته باشید که به‌طور پیش فرض (اهمیت آرگومان = FALSE)، randomForest فقط شاخص ناخالصی جینی را محاسبه می‌کند. با این حال، محاسبه دقت مدل توسط متغیر (اهمیت استدلال ­= TRUE) به محاسبات تکمیلی نیاز دارد که ممکن است در موقعیت‌هایی که هزاران مدل (درخت) برازش می‌شوند، زمان‌بر باشد.

معیارهای اهمیت متغیرها را می‌توان با استفاده از تابع varlmpPlotO [بسته جنگل تصادفی] ترسیم کرد:

* *نمودار MeanDecreaseAccuracy*

**varlmpPlot (**مدل **$** finalModel، نوع = 1)

* *طرح MeanDecreaseGini*

**varlmpPlot (**مدل **$** finalModel، نوع = 2)

model$finalModel model$finalModel

MeanDecreaseAccuracy

MeanDecreaseGini

|  |  |  |  |
| --- | --- | --- | --- |
| گلوکز | o | گلوکز | o |
| سن | -o- | سن | o |
| حامله | -o- | جرم | o |
| جرم | ■o- | شجره نامه | o |
| سه سر | ... س .... | انسولین | o |
| انسولین | -o- | سه سر | o |
| شجره نامه | ... س .... | حامله | o |
| فشار | o | فشار | o |
|  | من من من من \_  0 10 20 30 40 |  | 1 1 1 1 1 1 1  0 10 20 30 40 50 |

نتایج نشان می‌دهد که در تمام درختان در نظر گرفته شده در جنگل تصادفی، متغیرهای گلوکز و سن دو متغیر مهم هستند.

تابع varlmpO [incaret] اهمیت متغیرها را در درصد نشان می‌دهد:

**varlmp (**مدل)

* اهمیت متغیر # rf
* #

|  |  |
| --- | --- |
| ## | اهمیت |
| ## گلوکز | 100.0 |
| ## سن | 33.5 |
| ##باردار | 19.0 |
| ## جرم | 16.2 |

*##* سه سر 15.4

## شجره نامه

12.8

## انسولین 11.2 ## فشار 0.0

* 1. رگرسیون

به‌طور مشابه، شما می‌توانید یک مدل جنگل تصادفی برای انجام رگرسیون، یعنی پیش‌بینی یک متغیر پیوسته بسازید.

* + 1. نمونه‌ای از مجموعه داده ها

ما از مجموعه داده‌های بوستون [در بسته MASS ]، معرفی شده در فصل 2، برای پیش‌بینی مقدار خانه متوسط ( mdev )، در حومه بوستون، با استفاده از متغیرهای پیش‌بینی کننده مختلف استفاده خواهیم کرد.

به‌طور تصادفی داده‌ها را به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم کنید.

* *داده‌ها را بارگذاری کنید*

**داده (**"بوستون"، بسته = "MASS")

* *داده‌ها را بررسی کنید*

**sample\_n (**بوستون، 3)

* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**123)

training.samples <- Boston$medv °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE) train.data <- Bostonftraining.samples، ] test.data <- Bostonf **-** training.samples، ]

* + 1. محاسبه رگرسیون جنگل تصادفی

در اینجا خطای پیش‌بینی توسط RMSE اندازه‌گیری می‌شود که مربوط به میانگین تفاوت بین مقادیر مشاهده‌شده مشاهده‌شده نتیجه و مقدار پیش‌بینی‌شده توسط مدل است. RMSE به صورت RMSE = mean((مشاهده - پیش‌بینی)~2) %>% sqrt() محاسبه می‌شود. هرچه RMSE کمتر باشد، مدل بهتر است.

* *مدل را در مجموعه آموزشی قرار* **دهید set.seed (**123) model <- **train (**

medv **~** .، داده = train.data، روش = "rf"، trControl = **trainControl (**"cv"، عدد = 10) )

* *بهترین پارامتر تنظیم mtry*

مدل **$** bestTune

* *روی داده‌های آزمون پیش‌بینی کنید*

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**test.data) **head (**پیش‌بینی)

* *محاسبه میانگین خطای پیش‌بینی RMSE* **RMSE (**پیش‌بینی‌ها، test.data **$** medv)
  1. فراپارامترها

توجه داشته باشید که الگوریتم جنگل تصادفی دارای مجموعه‌ای از فراپارامترها است که باید با استفاده از اعتبارسنجی متقاطع تنظیم شوند تا از بیش برازش جلوگیری شود.

این شامل:

* گره : حداقل اندازه گره‌های ترمینال. مقدار پیش فرض برای طبقه‌بندی 1 و پیش فرض برای رگرسیون 5 است.
* maxnodes : حداکثر تعداد گره‌های پایانی که درختان در جنگل می‌توانند داشته باشند. اگر داده نشود، درختان تا حداکثر ممکن رشد می‌کنند (با توجه به اندازه گره‌ها، محدودیت‌هایی وجود دارد).

نادیده گرفتن این پارامترها ممکن است منجر به بیش برازش در مجموعه داده‌های پر نویز شود (بروس و بروس، 2017). اعتبارسنجی متقاطع می‌تواند برای آزمایش مقادیر مختلف به منظور انتخاب مقدار بهینه استفاده شود.

بسته caret می‌توان هایپرپارامترها را به صورت دستی تنظیم کرد . برای یک پارامتر معین، این رویکرد شامل برازش بسیاری از مدل‌ها با مقادیر مختلف پارامترها و سپس مقایسه مدل‌ها است.

مثال زیر مقادیر مختلف اندازه گره را با استفاده از مجموعه داده PimaIndiansDiabetes2 برای طبقه‌بندی آزمایش می‌کند:

(زمان اجرا 1-2 دقیقه طول می‌کشد)

**داده‌ها (**"PimaIndiansDiabetes2"، بسته = "mlbench") مدل‌های <- **لیست (**)

**برای (**اندازه گره **در c (**1، 2، 4، 8)) {

**set.seed (**123)

مدل <- **قطار (**

دیابت **-** .، داده = **na.omit (**PimaIndiansDiabetes2)، روش = "rf"، trControl = **trainControl (**روش = "cv"، شماره = 10)، متریک = "دقت"، nodesize = اندازه گره

)

model.name <- **toString (**nodesize) modelsffmodel.name]] <- مدل

}

*# نتایج را مقایسه کنید*

**نمونه‌های مجدد (**مدل ها) 0 / 0 > 0 / 0 **خلاصه (**متریک = "دقت")

* #
* # زنگ زدن:
* # summary.resamples(object = .، متریک = "دقت")

##

* # مدل ها: 1، 2، 4، 8
* # تعداد نمونه مجدد: 10
* #
* # دقت

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ## | حداقل | 1 ق. | میانه | منظور داشتن | 3 ق. | حداکثر | NA |
| ## 1 | 0.692 | 0.750 | 0.785 | 0.793 | 0.840 | 0.897 | 0 |
| ## 2 | 0.692 | 0.744 | 0.808 | 0.788 | 0.841 | 0.850 | 0 |
| ## 4 | 0.692 | 0.744 | 0.795 | 0.786 | 0.825 | 0.846 | 0 |
| ## 8 | 0.692 | 0.750 | 0.808 | 0.796 | 0.841 | 0.897 | 0 |

مشاهده می‌شود که استفاده از مقدار گره 2 یا 8 منجر به متوسط ترین مقدار دقت می‌شود.

33.6 بحث

این فصل اصول اولیه الگوریتم‌های یادگیری ماشین بسته‌بندی و جنگل تصادفی را توضیح می‌دهد. ما همچنین مثال‌های عملی را در R برای تحلیل طبقه‌بندی و رگرسیون ارائه می‌دهیم.

یکی دیگر از جایگزین‌های جنگل تصادفی، تقویت است (فصل 34).

فصل 34

تقویت

قبلاً، ما الگوریتم‌های بسته‌بندی و یادگیری ماشین جنگل تصادفی را برای ساخت ­یک مدل پیش‌بینی قدرتمند شرح داده‌ایم (فصل 33).

به یاد داشته باشید که بسته‌بندی شامل گرفتن زیرمجموعه‌های متعدد از مجموعه داده‌های آموزشی، سپس ساختن چندین مدل درخت تصمیم مستقل و سپس میانگین مدل‌هایی است که امکان ایجاد یک مدل پیش‌بینی بسیار کارآمد را در مقایسه با مدل کلاسیک CART فراهم می‌کند (فصل 32).

این فصل یک روش جایگزین به نام **تقویت را توضیح** می‌دهد که شبیه به روش بسته‌بندی است، با این تفاوت که درختان به صورت متوالی رشد می‌کنند: هر درخت متوالی با استفاده از اطلاعات درختان قبلی رشد می‌کند، با هدف به حداقل رساندن خطای مدل‌های قبلی (جیمز) و همکاران، 2014).

به‌عنوان مثال، با توجه به یک مدل درخت رگرسیون فعلی، روش به شرح زیر است:

1. یک درخت تصمیم را با استفاده از خطاهای باقیمانده مدل به‌عنوان متغیر نتیجه برازش دهید.
2. این درخت تصمیم جدید را که توسط پارامتر انقباض لامبدا تنظیم شده است، به تابع برازش اضافه کنید تا بقایای باقیمانده را به روز کنید. لامبدا یک مقدار مثبت کوچک است که معمولاً بین 0.01 و 0.001 است (جیمز و همکاران، 2014).

این رویکرد منجر به بهبود آهسته و متوالی مدل برازش شده و منجر به یک مدل بسیار کارآمد می‌شود. Boosting پارامترهای تنظیم متفاوتی دارد از جمله:

* تعداد درختان B
* پارامتر انقباض لامبدا
* تعداد تقسیمات در هر درخت

انواع مختلفی از تقویت وجود دارد، از جمله *Adaboost،* *تقویت گرادیان* و تقویت *گرادیان تصادفی* .

تقویت گرادیان تصادفی، پیاده‌سازی شده در بسته R *xgboost،* رایج‌ترین تکنیک تقویتی است که شامل نمونه‌برداری مجدد از مشاهدات و ستون‌ها در هر دور است. بهترین عملکرد را ارائه می‌دهد. xgboost مخفف تقویت حدی *گرادیان* است.

Boosting را می‌توان برای مسائل طبقه‌بندی و رگرسیون استفاده کرد.

در این فصل نحوه محاسبه تقویت در R را شرح خواهیم داد.

* 1. بارگیری بسته‌های R مورد نیاز
* tidyverse برای دستکاری و تجسم آسان داده ها
* گردش کار آسان یادگیری ماشین مراقبت کنید
* xgboost برای الگوریتم تقویت محاسباتی

**کتابخانه (**tidyverse)

**کتابخانه (**کارت)

**کتابخانه (**xgboost)

* 1. طبقه‌بندی
     1. نمونه‌ای از مجموعه داده ها

مجموعه داده ها: PimaIndiansDiabetes2 [در بسته mlbench ]، معرفی شده در فصل 20، برای پیش‌بینی ­احتمال مثبت بودن دیابت بر اساس متغیرهای بالینی متعدد.

به‌طور تصادفی داده‌ها را به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم کنید. مطمئن شوید که seed را برای تکرارپذیری تنظیم کنید.

* *داده‌ها را بارگذاری کنید و NA‌ها را حذف کنید*

**داده (**"PimaIndiansDiabetes2"، بسته = "mlbench")

PimaIndiansDiabetes2 <- **na.omit (**PimaIndiansDiabetes2)

* *داده‌ها* **نمونه\_n را بررسی کنید (**Pima!ndiansDiabetes2, 3)
* *تقسیم داده‌ها به مجموعه آموزشی و تست* **set.seed (**123)

training.samples <- PimaIndiansDiabetes2$diabetes °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE)

train.data <- PimaIndiansDiabetes2[training.samples, ]

test.data <- PimaIndiansDiabetes2[ **-** training.samples, ]

* + 1. درختان طبقه‌بندی تقویت شده

ما از گردش کار caret استفاده می‌کنیم که بسته xgboost را فراخوانی می‌کند تا به‌طور خودکار مقادیر پارامتر مدل را تنظیم کنیم و بهترین درخت تقویت‌شده نهایی را که بهترین داده‌های ما را توضیح می‌دهد، برازش دهیم.

از آرگومان‌های زیر در تابع train() استفاده می‌کنیم :

* trControl، برای تنظیم اعتبار متقاطع 10 برابری
* *مدل را در مجموعه آموزشی قرار* **دهید set.seed (**123) model <- **train (**

دیابت **~** .، داده = train.data، روش = "xgbTree"، trControl = **trainControl (**"cv"، عدد = 10) )

* *بهترین* مدل پارامتر تنظیم **$** bestTune
* # nrounds max\_depth eta gama colsample\_bytree min\_child\_weight subsample
* # 18 150 1 0.3 0 0.8 1 1
* *روی داده‌های آزمون پیش‌بینی کنید*

predicted.classes <- model 0 / 0 > 0 / 0 **پیش‌بینی (**test.data) **head (**predicted.classes)

* # [1] نگ پوس نگ نگر پوس نگ
* # سطوح: پوز منفی
* *محاسبه میزان صحت پیش‌بینی مدل*

**میانگین (**predicted.classes **==** test.data **$** دیابت)

* # [1] 0.744

صحت پیش‌بینی در داده‌های آزمایشی جدید 74 درصد است که خوب است.

برای توضیح بیشتر در مورد پارامترهای تنظیم تقویت، ?xgboost را در R تایپ کنید تا مستندات را ببینید.

* + 1. اهمیت متغیر

تابع varlmpO [در حواشی] اهمیت متغیرها را در درصد نشان می‌دهد: **varlmp (**مدل) ## xgbدرخت اهمیت متغیر

##

|  |  |
| --- | --- |
| ## | به‌طور کلی |
| ## گلوکز | 100.00 |
| ## جرم | 20.23 |
| ##باردار | 15.83 |
| ## انسولین | 13.15 |
| ## فشار | 9.51 |
| ## سه سر بازو | 8.18 |
| ## شجره نامه | 0.00 |
| ## سن | 0.00 |

* 1. رگرسیون

به‌طور مشابه، شما می‌توانید یک مدل جنگل تصادفی برای انجام رگرسیون، یعنی پیش‌بینی یک متغیر پیوسته بسازید.

* + 1. نمونه‌ای از مجموعه داده ها

ما از مجموعه داده‌های بوستون [در بسته MASS ]، معرفی شده در فصل 2، برای پیش‌بینی مقدار خانه متوسط ( mdev )، در حومه بوستون، با استفاده از متغیرهای پیش‌بینی کننده مختلف استفاده خواهیم کرد.

به‌طور تصادفی داده‌ها را به مجموعه آموزشی (80٪ برای ساخت یک مدل پیش‌بینی) و مجموعه تست (20٪ برای ارزیابی مدل) تقسیم کنید.

* *داده‌ها را بارگذاری کنید*

**داده (**"بوستون"، بسته = "MASS")

* *داده‌ها را بررسی کنید*

**sample\_n (**بوستون، 3)

* *داده‌ها را به مجموعه آموزشی و آزمایشی تقسیم کنید*

**set.seed (**123)

training.samples <- Boston$medv °/ o >°/o

**createDataPartition (**p = 0.8، لیست = FALSE) train.data <- Bostonftraining.samples، ] test.data <- Bostonf **-** training.samples، ]

* + 1. درختان رگرسیون تقویت شده

در اینجا خطای پیش‌بینی توسط RMSE اندازه‌گیری می‌شود که مربوط به میانگین تفاوت بین مقادیر مشاهده‌شده مشاهده‌شده نتیجه و مقدار پیش‌بینی‌شده توسط مدل است.

* *مدل را روی مجموعه آموزشی قرار دهید*

**set.seed (**123)

مدل <- **قطار (**

medv **-** .، داده = train.data، روش = "xgbTree"، trControl = **trainControl (**"cv"، عدد = 10) )

* *بهترین پارامتر تنظیم mtry*

مدل **$** bestTune

* *روی داده‌های آزمون پیش‌بینی کنید*

پیش‌بینی‌ها <- مدل °/ o >°/o **پیش‌بینی (**test.data)

**سر (**پیش‌بینی)

* *میانگین خطای پیش‌بینی RMSE را محاسبه کنید*

**RMSE (**پیش‌بینی‌ها، test.data **$** medv)

* 1. بحث

این فصل تکنیک‌های تقویت یادگیری ماشین را توضیح می‌دهد و مثال‌هایی را در R برای ساخت یک مدل پیش‌بینی ارائه می‌دهد. همچنین به روش‌های جنگل‌بندی و تصادفی در فصل 33 مراجعه کنید.

قسمت هشتم

یادگیری بدون نظارت

فصل 35

یادگیری بدون نظارت

* 1. مقدمه

**یادگیری بدون نظارت** به مجموعه‌ای از تکنیک‌های آماری برای کاوش و کشف دانش، از داده‌های چند متغیره، بدون ساخت مدل‌های پیش‌بینی اشاره دارد.

تجسم رابطه بین متغیرها و همچنین شناسایی گروه‌هایی از افراد مشابه (یا مشاهدات) را ممکن می‌سازد.

محبوب ترین روش‌های یادگیری بدون نظارت عبارتند از:

* **روش‌های مؤلفه اصلی که** شامل خلاصه کردن و تجسم مهمترین اطلاعات موجود در یک مجموعه داده چند متغیره است.
* **تحلیل خوشه‌ای** برای شناسایی گروه‌هایی از مشاهدات با مشخصات مشابه بر اساس یک معیار خاص. این تکنیک‌ها شامل خوشه‌بندی سلسله مراتبی و خوشه‌بندی k-means است.

قبلاً دو کتاب با عنوان «راهنمای عملی تحلیل خوشه‌ای در R1 » و «راهنمای عملی روش‌های مؤلفه اصلی» منتشر کرده‌ایم.[[9]](#footnote-9) [[10]](#footnote-10)". بنابراین، این فصل فقط یک نمای کلی از تکنیک‌های یادگیری بدون نظارت و مثال‌های عملی در R برای تجسم ­مجموعه داده‌های چند متغیره ارائه می‌دهد.

* 1. روش‌های اجزای اصلی

روش‌های مؤلفه اصلی به ما اجازه می‌دهد تا مهم‌ترین اطلاعات ­موجود در یک مجموعه داده چند متغیره را خلاصه و تجسم کنیم.

نوع روش‌های مؤلفه اصلی برای استفاده به انواع متغیرهای موجود در مجموعه داده بستگی دارد. این راهنمای عملی روش‌های زیر را شرح می‌دهد:

1. **تحلیل اجزای اصلی (PCA)** که یکی از محبوب ترین روش‌های ­تحلیل چند متغیره است. هدف PCA خلاصه کردن اطلاعات موجود در داده‌های چند متغیره پیوسته (یعنی کمی) با کاهش ابعاد داده‌ها بدون از دست دادن اطلاعات مهم است.
2. **تحلیل متناظر (CA) که** توسعه‌ای از تحلیل مؤلفه اصلی برای تحلیل یک جدول احتمالی بزرگ است که توسط دو *متغیر کیفی (*یا داده‌های طبقه‌بندی شده) تشکیل شده است.
3. **تحلیل مکاتبات چندگانه (MCA) که** انطباق CA با یک جدول داده حاوی بیش از دو متغیر طبقه‌بندی است.
   1. بارگیری بسته‌های R مورد نیاز

* FactoMineR برای محاسبه روش‌های مؤلفه اصلی
* factoextra برای تجسم خروجی FactoMineR

**کتابخانه (**FactoMineR)

**کتابخانه (**factoextra)

* + 1. تحلیل مؤلفه‌های اصلی

PCA داده‌ها را به چند بعد (یا محور) جدید کاهش می‌دهد که ترکیبی خطی از متغیرهای اصلی است. شما می‌توانید یک داده چند متغیره را با رسم نمودار پراکندگی دو بعد اول که حاوی مهمترین اطلاعات در داده‌ها است، تجسم کنید. ادامه مطلب در: <https://goo.gl/kabVHq>.

* مجموعه داده‌های نمایشی: USArrests
* محاسبه PCA با استفاده از تابع R PCA() [FactoMineR]
* خروجی را با استفاده از پکیج factoextra R تجسم کنید (پسوندی برای ggplot2)

1. PCA را با استفاده از مجموعه داده‌های نمایشی USArrests محاسبه کنید. مجموعه داده‌ها شامل آمار دستگیری به ازای هر 100000 نفر از ساکنان به دلیل حمله، قتل و تجاوز جنسی در هر یک از 50 ایالت ایالات متحده در سال 1973 است.

**داده‌ها (**"USArrests")

res.pca <- **PCA (**USArrests، graph = FALSE)

1. مقادیر ویژه (یا *نمودار scree* ) که درصد تغییرات (یا اطلاعات) در داده‌ها است، توسط هر جزء اصلی توضیح داده شده است.

**fviz\_eig (**res.pca)

طرح اسکری

ابعاد

1. نمودار افراد را تجسم کنید. افراد با مشخصات مشابه با هم گروه‌بندی می‌شوند.

**fviz\_pca\_ind (**res.pca، repel = TRUE)

افراد - PCA

.p

’4’ CM

CM

E

Q

Hawaii Massachusetts

Rhode Island \*U"h /New Jersey California

Connecticut—. Washington Colorado

Minnesota Ohio Oregon \*New YNevada

W^consin'-Pe^k Illinois\*A^zona \*

Iowa Kansas—

0 .al \_ \_ Nebra;

• •Texas Michigan

Delaware- -Missouri -Florida,

-1

New Hampshirendiana •Virginia New Mexico Maine Idaho e Wyoming Maryland\* North DakotaDntanKentuJeyinessee- e Louisiana

Alabama

South Dakota

Vermont

West Virginia

Arkansas

Alaska

Georgia

کارولینای جنوبی

-2

کارولینای شمالی

-2

کم نور 1 (62%)

توجه داشته باشید که ابعاد (Dim) 1 و 2 حدود 87% (62% + 24.7%) از کل اطلاعات موجود در مجموعه داده را حفظ کردند که بسیار خوب است.

1. نمودار متغیرها را تجسم کنید. متغیرهای همبسته مثبت به همان سمت طرح اشاره می‌کنند. متغیرهای همبسته منفی به طرف مقابل نمودار اشاره می‌کنند.

**fviz\_pca\_var (**res.pca)

1. یک بای پلات از افراد و متغیرها ایجاد کنید

**fviz\_pca\_biplot (**res.pca، repel = TRUE)

1. دسترسی به نتایج:

* *مقادیر ویژه*

res.pca **$** eig

* *نتایج برای متغیرها*

res.var <- res.pca **$** var

res.var **$** coord *# مختصات*

res.var **$** contrib *# مشارکت در مؤلفه‌های اصلی*

res.var **$** cos2 *# کیفیت نمایش*

* *نتایج برای افراد*

res.ind <- res.pca **$** var

res.ind **$** coord *# مختصات*

res.ind **$** contrib *# مشارکت در مؤلفه‌های اصلی*

res.ind **$** cos2 *# کیفیت نمایندگی*

1. بیشتر بخوانید در: Principal Component Analyis 3

35.3.2 تحلیل مکاتبات

تحلیل مکاتبات توسعه‌ای از تحلیل مؤلفه اصلی برای تحلیل یک جدول احتمالی بزرگ است که توسط دو متغیر کیفی (یا داده‌های طبقه‌بندی شده) تشکیل شده است. مانند تحلیل مؤلفه‌های اصلی، راه حلی برای خلاصه کردن و تجسم مجموعه داده‌ها در نمودارهای دو بعدی ارائه می‌دهد.

نمودار ارتباط بین نقاط ردیف و ستون جدول احتمالی را نشان می‌دهد.

* مجموعه داده‌های نمایشی: Housetasks [در FactoMineR] که یک جدول اقتضایی است که شامل تعداد دفعات اجرای 13 وظیفه خانه در زوج است.
* محاسبه CA با استفاده از تابع R CA() [FactoMineR]
* خروجی را با استفاده از پکیج factoextra R تجسم کنید
* *مجموعه داده را بارگیری و بررسی کنید*

**داده (**"وظایف خانه")

**سر (**وظایف خانه، 4)

## همسر جایگزین شوهر مشترک

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ## خشکشویی | 156 | 14 | 2 | 4 |
| ## غذای اصلی | 124 | 20 | 5 | 4 |
| ## شام | 77 | 11 | 7 | 13 |
| ## صبحانه | 82 | 36 | 15 | 7 |

* *محاسبه تحلیل مکاتبات* res.ca <- **CA (**وظایف خانه، نمودار = FALSE) **fviz\_ca\_biplot (**res.ca، repel = TRUE)

CA - Biplot

Repairs\*

Husband ▲

Laundry Breakfeast J Driving

MainWfea

شام متناوب f^^   
0.0 ;

-0.5

-1.0

-1.5

Tidying Shopping Insurance

Dishes j  
Finances

I

। Jointly I A

I  
I

[ Holidays

-1.0 -0.5

0.0

0.5

1.0

1.5

Dim1 (48.7%)

از نمودار بالا می‌توان فهمید که:

* کارهای خانه مانند شام، صبحانه، شستن لباس‌ها بیشتر توسط همسر انجام می‌شود
* رانندگی و تعمیرات بیشتر توسط شوهر انجام می‌شود

بیشتر بخوانید در: تحلیل مکاتبات در R 4

35.3.3 تحلیل مکاتبات چندگانه

تحلیل **مکاتبات چندگانه (** **MCA** ) توسعه‌ای از تحلیل مکاتبات ساده برای خلاصه کردن و تجسم یک جدول داده حاوی بیش از دو متغیر طبقه‌بندی شده است. همچنین می‌توان آن را به‌عنوان تعمیم تحلیل مؤلفه‌های اصلی در نظر گرفت که متغیرهای مورد تحلیل به جای کمی، مقوله‌ای باشند.

**MCA** به‌طور کلی برای تحلیل مجموعه داده‌های نظرسنجی استفاده می‌شود. هدف شناسایی موارد زیر است: 1) گروهی از افراد با مشخصات مشابه در پاسخ به سؤالات. 2) ارتباط بین دسته‌های متغیر

* مجموعه داده‌های نمایشی: poison [در FactoMineR]. این داده‌ها نتیجه یک نظرسنجی است که بر روی کودکان دبستانی که از مسمومیت غذایی رنج می‌بردند انجام شده است. از آنها در مورد علائم و آنچه می‌خوردند سؤال شد.
* محاسبه MCA با استفاده از تابع R MCA() [FactoMineR]
* خروجی را با استفاده از پکیج factoextra R تجسم کنید

*# بارگذاری داده ها*

**داده (**"سم")

**سر (**زهر[، 1 **:** 8]، 4)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ## | | سن  9 | زمان  22 | مریض  Sick\_y | رابطه ی جنسی  اف | حالت تهوع  حالت تهوع | استفراغ  استفراغ\_n | شکم  Abdo\_y | تب  Fever\_y |
| ## | 1 |
| ## | 2 | 5 | 0 | Sick\_n | اف | تهوع\_n | استفراغ\_n | Abdo\_n | Fever\_n |
| ## | 3 | 6 | 16 | Sick\_y | اف | تهوع\_n | استفراغ\_y | Abdo\_y | Fever\_y |
| ## | 4 | 9 | 0 | بیمار n | اف | تهوع n | استفراغ n | Abdo n | تب n |

داده‌ها حاوی چند متغیر تکمیلی هستند. آنها در MCA شرکت نمی‌کنند. مختصات این متغیرها پیش‌بینی خواهد شد.

* *متغیرهای کمی تکمیلی (*quanti.sup): ستون‌های 1 و 2 به ترتیب مربوط به *سن* و *زمان ستون‌ها* هستند.
* *متغیرهای کیفی تکمیلی (*quali.sup: ستون‌های 3 و 4 به ترتیب مربوط به ستون‌های *Sick* و *Sex* . از این متغیرهای عامل برای رنگ‌آمیزی افراد بر اساس گروه‌ها استفاده می‌شود.

MCA را محاسبه و تجسم کنید:

* *محاسبه تحلیل مکاتبات چندگانه:*

res.mca <- **MCA (**poison, quanti.sup = 1 **:** 2, quali.sup = 3 **:** 4, graph=FALSE)

* *نمودار افراد، رنگ آمیزی شده بر اساس گروه ("بیمار")*

**fviz\_mca\_ind (**res.mca، repel = TRUE، habillage = "بیمار"، addEllipses = TRUE)

افراد - MCA

\39 35^?

Sick

• Sick\_n

• Sick\_y

*# Graph of variable categories* **fviz\_mca\_var**(res.mca, repel = TRUE)

Variable categories - MCA

ACourg\_n ■

Icecream\_n

09 c\i

CM

E

Q

1 n —

Vomit y

Fever\_y

Diarr Sick\_

Nausea\_y

-1 — —

Potato\_y

Cheese\_y>F Nausea-1 -k “irFislrj—

Cheese\_n

-2

-1.0

Mayo\_y Icecream\_y

Fish\_n

Potato\_n

-0.5

0.0

n Mayo\_n

-DiaTrhea\_mSfck\_n\_n-

. Fever\_n

Vomit\_n

0.5

1.0

1.5

Dim1 (33.5%)

تفسیر:

* نمودار افراد: افراد با مشخصات مشابه با هم گروه‌بندی می‌شوند.
* نمودار دسته‌بندی متغیرها: نمودار بالا روابط بین دسته‌های متغیر را نشان می‌دهد. می‌توان آن را به صورت زیر تفسیر کرد:

**-** دسته بندی‌های متغیر با نمایه مشابه با هم گروه‌بندی می‌شوند.

**-** دسته‌های متغیر با همبستگی منفی در دو طرف طرح قرار می‌گیرند

منشاء (ربع مخالف).

**-** فاصله بین نقاط دسته و مبدا کیفیت دسته متغیر را در نقشه عامل اندازه‌گیری می‌کند. نقاط دسته‌ای که از مبدأ دور هستند به خوبی در نقشه عامل نشان داده می‌شوند.

Biplot افراد و متغیرها. نمودار بالا یک الگوی جهانی را در داده‌ها نشان می‌دهد. ردیف‌ها (افراد) با نقاط آبی و ستون‌ها (دسته‌های متغیر) با مثلث‌های قرمز نشان داده می‌شوند. متغیرها و افرادی که ارتباط مثبت دارند در یک سمت طرح قرار دارند.

**fviz\_mca\_biplot (**res.mca، repel = TRUE، ggtheme = **theme\_minimal (**))

بیشتر بخوانید در: تحلیل مکاتبات چندگانه 5

35.4 تحلیل خوشه ای

* + 1. مبانی

تحلیل خوشه‌ای برای شناسایی گروه‌هایی از اشیاء مشابه در مجموعه داده‌های چند متغیره جمع‌آوری شده از زمینه‌هایی مانند بازاریابی، زیست پزشکی و جغرافیایی استفاده می‌شود. بیشتر بخوانید در: <http://www.sthda>. com/english/articles/25-cluster-analysis-in-r-practical-guide/ .

**نوع خوشه‌بندی** . انواع مختلفی از روش‌های خوشه‌بندی وجود دارد، از جمله:

* خوشه‌بندی پارتیشن بندی: داده‌ها را به مجموعه‌ای از گروه‌های k تقسیم می‌کند.
* خوشه‌بندی سلسله مراتبی: گروه‌ها را در داده‌ها بدون تقسیم‌بندی آن‌ها شناسایی کنید.

گیری **فاصله** طبقه‌بندی مشاهدات به گروه‌ها نیازمند روش‌هایی برای محاسبه فاصله یا (ناهمسانی) بین هر جفت مشاهدات است. نتیجه این محاسبه به‌عنوان ماتریس عدم تشابه یا فاصله شناخته می‌شود. روش‌های مختلفی برای اندازه‌گیری فاصله وجود دارد که عبارتند از:

* فاصله ی اقلیدسی
* همبستگی مبتنی بر فاصله

**چه نوع اندازه‌گیری فاصله را انتخاب کنیم؟** انتخاب معیارهای فاصله بسیار مهم است، زیرا تأثیر زیادی بر نتایج خوشه‌بندی دارد. برای اکثر نرم‌افزارهای خوشه‌بندی متداول، اندازه‌گیری فاصله پیش فرض فاصله اقلیدسی است.

بسته به نوع داده‌ها و سؤالات محقق، سایر معیارهای عدم تشابه ممکن است ترجیح داده شوند.

اگر بخواهیم خوشه‌هایی از مشاهدات را با نمایه‌های کلی یکسان بدون توجه به بزرگی آنها شناسایی کنیم، باید *فاصله مبتنی بر همبستگی را* به‌عنوان معیار عدم تشابه انتخاب کنیم. این مورد به ویژه در تحلیل داده‌های بیان ژنی است، جایی که ممکن است بخواهیم ژن‌ها را زمانی که «بالا» و «پایین» با هم هستند مشابه در نظر بگیریم. اگر بخواهیم گروهی از خریداران را بدون توجه به حجم اقلامی که خریده‌اند، از نظر اقلام با اولویت یکسان شناسایی کنیم، در بازاریابی نیز چنین است.

اگر فاصله اقلیدسی انتخاب شود، مشاهدات با مقادیر بالایی از ویژگی‌ها با هم خوشه می‌شوند. همین امر در مورد مشاهدات با مقادیر کم ویژگی صادق است.

**استاندارد‌سازی داده‌ها** قبل از تحلیل خوشه ای، توصیه می‌شود داده‌ها را مقیاس (یا نرمال سازی) کنید تا متغیرها قابل مقایسه باشند. این به ویژه زمانی توصیه می‌شود که متغیرها در مقیاس‌های مختلف اندازه‌گیری شوند (به‌عنوان مثال: کیلوگرم، کیلومتر، سانتی متر، ...). در غیر این صورت، ­اقدامات ناهماهنگ به دست آمده به شدت تحت تاثیر قرار خواهد گرفت. تابع R برای مقیاس‌بندی داده ها: scaleO، مقیاس‌بندی را بر روی ستون داده‌ها (متغیرها) اعمال می‌کند.

* + 1. بارگیری بسته‌های R مورد نیاز
* خوشه برای تحلیل خوشه
* factoextra برای تجسم خوشه

**کتابخانه (**خوشه ای)

**کتابخانه (**factoextra)

* + 1. آماده‌سازی داده ها

ما از مجموعه داده‌های نمایشی USArrests استفاده خواهیم کرد. ما با استاندارد کردن داده‌ها شروع می‌کنیم:

mydata <- **scale (**USArrests)

* + 1. خوشه‌بندی پارتیشن بندی

الگوریتم‌های پارتیشن‌بندی تکنیک‌های خوشه‌بندی هستند که مجموعه‌های داده را به مجموعه‌ای از k گروه تقسیم می‌کنند، جایی که k تعداد گروه‌هایی است که از قبل توسط تحلیلگر مشخص شده است.

انواع مختلفی از روش‌های پارتیشن‌بندی خوشه‌بندی وجود دارد. محبوب ترین **خوشه‌بندی K-means** است که در آن، هر خوشه با مرکز یا میانگین نقاط داده متعلق به خوشه نشان داده می‌شود. روش K-means به نقاط پرت حساس است.

جایگزینی برای خوشه‌بندی k-means، خوشه‌بندی **K-medoids** یا PAM (Partitioning Around Medoids) است که در مقایسه با k-means حساسیت کمتری به نقاط پرت دارد.

بیشتر بخوانید: روش‌های خوشه‌بندی پارتیشن‌بندی [[11]](#footnote-11).

کدهای R زیر نحوه تعیین تعداد بهینه خوشه‌ها و نحوه محاسبه k-means و خوشه‌بندی PAM را در R نشان می‌دهد.

1. **تعیین تعداد بهینه خوشه‌ها** : استفاده از factoextra::fviz\_nbclust() **fviz\_nbclust (**mydata، kmeans، روش = "gap\_stat")

تعداد بهینه خوشه ها

2. **Compute and visualize k-means clustering**: **set.seed**(123) *# for reproducibility* km.res <- **kmeans**(mydata, 3, nstart = 25) *# Visualize*

**fviz\_cluster**(km.res, data = mydata, palette = "jco",  
ggtheme = **theme\_minimal**())

Cluster plot

CM

CM

E

Q

Nevada

Maine

cluster

1

2

3

Or

کم نور 1 (62%)

3. به‌طور مشابه، خوشه‌بندی **k-medoids/pam** را می‌توان به صورت زیر محاسبه و تجسم کرد:

pam.res <- **pam (**mydata, 3)

**fviz\_cluster (**pam.res)

35.4.5 خوشه‌بندی سلسله مراتبی

خوشه‌بندی سلسله مراتبی یک رویکرد جایگزین برای پارتیشن‌بندی خوشه‌بندی برای شناسایی گروه‌ها در مجموعه داده است. این نیازی به از قبل مشخص کردن تعداد خوشه‌هایی که باید تولید شوند نیست.

نتیجه خوشه‌بندی سلسله مراتبی، نمایش درختی از اشیاء است که به‌عنوان دندروگرام نیز شناخته می‌شود. مشاهدات را می‌توان با برش دندروگرام در سطح شباهت دلخواه به گروه‌هایی تقسیم کرد.

* محاسبه: تابع R: hclust() . ماتریس عدم تشابه را به‌عنوان ورودی می‌گیرد که با استفاده از تابع dist() محاسبه می‌شود .
* تجسم: fviz\_dend() [in factoextra]

کد R برای محاسبه و تجسم خوشه‌بندی سلسله مراتبی:

res.hc <- **hclust (** **dist (**mydata)، روش = "ward.D2")

**fviz\_dend (**res.hc، cex = 0.5، k = 4، پالت = "jco")

دندروگرام خوشه ای

10

سی تی

**سی دی**

T 5

نقشه حرارتی روش دیگری برای تجسم خوشه‌بندی سلسله مراتبی است. به آن تصویر با رنگ کاذب نیز می‌گویند که در آن مقادیر داده‌ها به مقیاس رنگی تبدیل می‌شوند. نقشه‌های حرارتی به ما این امکان را می‌دهند که به‌طور همزمان گروه‌هایی از نمونه‌ها و ویژگی‌ها را تجسم کنیم. شما به راحتی می‌توانید یک نقشه حرارتی زیبا با استفاده از پکیج R pheatmap ایجاد کنید.

در نقشه حرارتی، به‌طور کلی، ستون‌ها نمونه و ردیف‌ها متغیر هستند. بنابراین ما با جابجایی داده‌ها قبل از ایجاد نقشه حرارتی شروع می‌کنیم.

**کتابخانه (**پیت مپ)

**pheatmap (** **t (**mydata)، cutree\_cols = 4)

35.5 بحث

این فصل متداول‌ترین روش‌های یادگیری ماشین بدون نظارت را برای خلاصه کردن و تجسم مجموعه داده‌های چند متغیره بزرگ ارائه می‌کند. می‌توانید اطلاعات بیشتر را در وب سایت STHDA در آدرس زیر بخوانید:

* راهنمای عملی روش‌های اجزای اصلی[[12]](#footnote-12) و
* راهنمای عملی تحلیل خوشه‌ای در R[[13]](#footnote-13)

کتابشناسی - فهرست کتب

Bovelstad, H., Nygard, S., Storvold, H., Aldrin, M., Borgan, o., Frigessi, A., and Lingjoerde, O. (2007). پیش‌بینی بقا از داده‌های ریزآرایه - یک مطالعه مقایسه‌ای *بیوانفورماتیک،* 23 (16): 2080-2087.

بروس، پی و بروس، ا. (2017). *آمار عملی برای دانشمندان داده* . رسانه اوریلی

فریدمن، جی اچ (1989). تحلیل متمایز منظم. *مجله انجمن آماری آمریکا،* 84 (405): 165-175.

جیمز، جی، ویتن، دی، هستی، تی و تبشیرانی، آر (2014). *مقدمه‌ای بر یادگیری آماری: با کاربردها در* R. شرکت انتشارات Springer، ثبت شده است.

ژانگ، زی (2016). مدلسازی درخت تصمیم با استفاده از r. *Annals of Translational Medicine،* 4(15).

فهرست مطالب

ROC curve, 148

RPART, 160

RSE, 11, 17, 60

RSS, 11

Sensitivity, 146

Specificity, 146

Variance inflation factor, 54

VIF, 54

AIC، 61

آگوست، 148

BIC، 61

سبد خرید، 160

دقت طبقه‌بندی، 144

تحلیل خوشه ای، 191

فاصله کوک، 51

ماتریس درهم‌ریختگی، 145

کدگذاری ساختگی، 26

آماره اف، 18

شاخص جینی، 162

خوشه‌بندی سلسله مراتبی، 194

k-means، 192

کاپا، 147

KNN، 157

LDA، 129

لاجیت، 106

MAE، 60

Mallows Cp, 61

فرد، 106

حداقل مربعات معمولی، 12

PCA، 185

PCR، 96

PLS، 96

دقت، 146

تحلیل مؤلفه‌های اصلی، 185

هرس، 165

QDA، 129

R-squared, 17, 60

R2, 60

به یاد بیاورید، 146

خطای استاندارد باقیمانده، 11، 17

مجموع مربعات باقیمانده، 11

RMSE، 60

• y = b0 + b1\*x،

• exp() نمایی و است

• p احتمال وقوع رویداد (1) با توجه به x است. از نظر ریاضی، این به صورت p(رویداد=1|x) نوشته می‌شود و به اختصار p(x) نوشته می‌شود، بنابراین px = 1/[1 + exp(-(b0 + b1\*x))]'

با کمی دستکاری، می‌توان نشان داد که p/(1-p) = exp(b0 + b1\*x) . با گرفتن لگاریتم هر دو طرف، فرمول به ترکیبی خطی از پیش‌بینی کننده‌ها تبدیل می‌شود: log[p/(1-p)] = b0 + b1\*x .

هنگامی که چندین متغیر پیش‌بینی دارید، تابع لجستیک به نظر می‌رسد: log[p/(1-p)] = b0 + b1\*x1 + b2\*x2 + ... + bn\*xn

|  |
| --- |
| **#Part I: Basics**  **#Chapter1: Introduction to R**  **mypkgs <- c("tidyverse", "caret")**  **install.packages (mypkgs)**  **library("tidyverse")**  **library("caret")**  **# Reads tab delimited files (.txt tab)**  **my\_data<-read.delim(file.choose())**  **#Reads comma (,)delimited files(.csv)**  **my\_data<-read.csv(file.choose())**  **# Reads semicolon(;) separated files(.csv)**  **my\_data<-read.csv2(file.choose())**  **data("iris") # Loading**  **head(iris, n= 3) # Print the first n = 3 rows**  **?iris**  **#-------------------------------------------------------------------------------**  **#Part II: Regression Analysis**  **#Chapter2:Introduction**  **if (!require(devtools)) install.packagee("devtools")**  **devtools:: install\_github("kassambara/datarium")**  **data("marketing", package = "datarium")**  **head (marketing, 3)**  **data("swiss")**  **head(swiss, 3)**  **data ("Boston" , package = "MASS")**  **head(Boston, 3)**  **#-------------------------------------------------------------------------------**  **#Chapter3: Linear Regression**  **install.packages('tidyverse')**  **library (tidyverse)**  **library (caret)**  **theme\_set(theme\_bw())**  **#Load the data**  **data("marketing", package = "datarium")**  **# Inspect the data**  **sample\_n(marketing, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- marketing$sales %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- marketing[training.samples, ]**  **test.data <- marketing[-training.samples, ]**  **# Build the model**  **model <- lm(sales ~., data = train.data)**  **# Summarize the model**  **summary(model)**  **# make predictions**  **predictions <- model %>% predict(test.data)**  **# model performance**  **# (a) Prediction error, RMSE**  **RMSE(predictions, test.data$sales)**  **# (b) R-square**  **R2(predictions, test.data$sales)**  **model <- lm(sales ~ youtube, data = train.data)**  **summary (model)$coef**  **newdata <- data.frame( youtube = c (0 , 1000))**  **model %>% predict(newdata)**  **model <- lm(sales ~ youtube + facebook + newspaper,data = train.data)**  **summary (model)$coef**  **model <- lm(sales ~.,data = train.data)**  **summary (model)$coef**  **# new advertising budgets**  **newdata <- data.frame(**  **youtube = 2000, facebook = 1000,**  **newspaper = 1000**  **)**  **# Predict sates values**  **model %>% predict(newdata)**  **summary (model)**  **summary (model)$coef**  **model <- lm(sales ~ youtube + facebook,data = train.data)**  **summary (model)**  **# Make predictions**  **predictions <- model %>% predict(test.data)**  **# Model performance**  **#(a) Compute the prediction error, RMSE**  **RMSE(predictions, test.data$sales)**  **#(b) Compute R-square**  **R2(predictions, test.data$sales)**  **ggplot (marketing, aes(x = youtube, y = sales)) + geom\_point() + stat\_smooth()**  **#-------------------------------------------------------------------------------**  **#Chapter4:Interaction Effects in Multiple Regression**  **library("tidyverse")**  **library("caret")**  **# Load the data**  **data("marketing", package = "datarium")**  **# Inspect the data**  **sample\_n(marketing, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- marketing$sales %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- marketing[training.samples, ]**  **test.data <- marketing[-training.samples, ]**  **# Build the model**  **model <- lm(sales ~., data = train.data)**  **# Summarize the model**  **summary(model)**  **# make predictions**  **predictions <- model %>% predict(test.data)**  **# model performance**  **# (a) Prediction error, RMSE**  **RMSE(predictions, test.data$sales)**  **# (b) R-square**  **R2(predictions, test.data$sales)**  **# Build the model**  **# Use this:**  **model2 <- lm(sales ~ youtube + facebook +youtube:facebook,data = marketing)**  **summary(model2)**  **# or simply , use this:**  **model2 <- lm(sales ~ youtube\*facebook, data = train.data)**  **# Summarize the model**  **summary(model2)**  **# Model performance**  **# (a) Compute the prediction error, RMSE**  **RMSE(predictions, test.data$sales)**  **#(b) Compute R-square**  **R2(predictions, test.data$sales)**  **#-------------------------------------------------------------------------------**  **#Chapter5: Regression with Categorical Variables**  **library(tidyverse)**  **library(car)**  **# Load the data**  **data("Salaries", package = "car")**  **# Inspect the data**  **sample\_n(Salaries, 3)**  **#compute the model**  **model <- lm(salary ~ sex ,data = Salaries)**  **summary (model)$coef**  **contrasts(Salaries$sex)**  **Salaries <- Salaries %>%**  **mutate(sex = relevel(sex, ref = "Male"))**  **contrasts(Salaries$sex)**  **#Salaries$sex <- relevel(Salaries$sex, ref = "Male")**  **#contrasts(Salaries$rank)**  **#Salaries <- Salaries %>%**  **# mutate(rank = relevel(rank, ref = "AssocProf"))**  **model <- lm(salary ~ sex ,data = Salaries)**  **summary (model)$coef**  **res <- model.matrix(~rank, data = Salaries)**  **head(res[, -1])**  **library(car)**  **model2 <- lm(salary ~ yrs.service + rank + discipline + sex, data = Salaries)**  **Anova(model2)**  **summary (model2)**  **#-------------------------------------------------------------------------------**  **#Chapter6: Nonlinear Regression**  **library("tidyverse")**  **library("caret")**  **theme\_set(theme\_classic())**  **#Load the data**  **data("Boston", package = "MASS")**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- Boston$medv %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- Boston[training.samples, ]**  **test.data <- Boston[-training.samples, ]**  **ggplot (train.data, aes(x = lstat,medv)) + geom\_point() + stat\_smooth()**  **# Build the model**  **model <- lm(medv ~ lstat, data = train.data)**  **# make predictions**  **predictions <- model %>% predict(test.data)**  **# model performance**  **data.frame(**  **RMSE = RMSE(predictions, test.data$medv),**  **R2 = R2(predictions, test.data$medv)**  **)**  **ggplot (train.data, aes(lstat,medv)) + geom\_point() + stat\_smooth(method = lm, formula = y ~ x)**  **lm(medv ~ lstat + I(lstat^2), data = train.data)**  **lm(medv ~ poly(lstat, 2), data = train.data) #**  **lm(medv ~ poly(lstat, 6), data = train.data) %>%**  **summary ()**  **# Build the model**  **model <- lm(medv ~ poly(lstat, 5), data = train.data)**  **# make predictions**  **predictions <- model %>% predict(test.data)**  **# model performance**  **data.frame(**  **RMSE = RMSE(predictions,test.data$medv),**  **R2 = R2(predictions,test.data$medv)**  **)**  **ggplot (train.data, aes(lstat,medv)) + geom\_point() + stat\_smooth(method = lm, formula = y ~ poly(x, 5))**  **# Build the model**  **#Log transformation**  **model <- lm(medv ~ log(lstat), data = train.data)**  **# make predictions**  **predictions <- model %>% predict(test.data)**  **# model performance**  **data.frame(**  **RMSE = RMSE(predictions, test.data$medv),**  **R2 = R2(predictions, test.data$medv)**  **)**  **ggplot (train.data, aes(lstat,medv)) + geom\_point() + stat\_smooth(method = lm, formula = y ~ log(x))**  **#Spline regression**  **knots <- quantile(train.data$lstat, p = c(0.25,0.5,0.75))**  **library(splines)**  **# Build the model**  **knots <- quantile(train.data$lstat, p = c(0.25,0.5,0.75))**  **model <- lm(medv ~ bs(lstat, knots = knots, degree = 2), data = train.data)**  **# make predictions**  **predictions <- model %>% predict(test.data)**  **# model performance**  **data.frame(**  **RMSE = RMSE(predictions, test.data$medv),**  **R2 = R2(predictions, test.data$medv))**  **ggplot (train.data, aes(lstat,medv)) +**  **geom\_point() +**  **stat\_smooth(method = lm, formula = y ~ splines::bs(x, df = 3))**  **#Generalized additive models, or GAM = technique to automatically fit a spline**  **library("mgcv")**  **# Build the model**  **model <- gam(medv ~ s(lstat), data = train.data)**  **# make predictions**  **predictions <- model %>% predict(test.data)**  **# model performance**  **data.frame(**  **RMSE = RMSE(predictions, test.data$medv),**  **R2 = R2(predictions, test.data$medv)**  **)**  **ggplot (train.data, aes(lstat,medv)) + geom\_point() + stat\_smooth(method = gam, formula = y ~ s(x))**  **#-------------------------------------------------------------------------------**  **#III Regression Diagnostics**  **#Chapter7: Introduction**  **#Chapter8: Regression Assumptions and Diagnostics**  **library(tidyverse)**  **library(broom)**  **theme\_set(theme\_classic())**  **#Load the data**  **data("marketing", package = "datarium")**  **# Inspect the data**  **sample\_n(marketing, 3)**  **model <- lm(sales ~ youtube, data = marketing)**  **model**  **#Fitted values and residuals**  **model.diag.metrics <- augment(model)**  **head(model.diag.metrics)**  **ggplot (model.diag.metrics, aes(youtube, sales)) + geom\_point() + stat\_smooth(method = lm , se = FALSE)+ geom\_segment(aes(xend =youtube, yend =.fitted), color = "red",size=0.3)**  **#Regression diagnostics**  **par(mfrow = c(2, 2))**  **plot(model)**  **library(ggfortify)**  **autoplot(model)**  **# Add observations indices and**  **# drop some columns (.se.fit,.sigma) for simplification**  **model.diag.metrics <- model.diag.metrics %>%**  **mutate (index = 1:nrow(model.diag.metrics)) %>%**  **select (index, everything(), -.se.fit, -.sigma)**  **# Inspect the data**  **head(model.diag.metrics, 4)**  **plot(model, 1)#?**  **plot(model, 3)**  **model2 <- lm(log(sales) ~ youtube, data = marketing)**  **plot(model2, 3)**  **plot(model, 2)**  **plot(model, 5)**  **#cook's distance**  **plot(model, 4)**  **#residuals vs leverage**  **plot(model, 5)**  **plot(model, 4 ,id.= 5)**  **model.diag.metrics %>%**  **top\_n(3, wt = cooksd)**  **df2 <- data.frame(**  **x = c(marketing$youtube, 500, 600),**  **y = c(marketing$sales, 80, 100)**  **)**  **model2 <- lm(y ~ x, df2)**  **#cook's distance**  **plot(model2, 4)**  **#residuals vs leverage**  **plot(model2, 5)**  **#-------------------------------------------------------------------------------**  **#Chapter9: Multicollinearity**  **library (tidyverse)**  **library (caret)**  **#Load the data**  **data ("Boston" , package = "MASS")**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- Boston$medv %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- Boston[training.samples, ]**  **test.data <- Boston[-training.samples, ]**  **# Build the model**  **model1 <- lm(medv ~., data = train.data)**  **# make predictions**  **predictions <- model1 %>% predict(test.data)**  **# model performance**  **data.frame(**  **RMSE = RMSE(predictions, test.data$medv),**  **R2 = R2(predictions, test.data$medv)**  **)**  **#Detecting multicollinearity**  **car::vif(model1)**  **# Build a model excluding the tax variable**  **model2 <- lm(medv ~.-tax, data = train.data)**  **# Make predictions**  **predictions <- model2 %>% predict (test.data)**  **# Model performance**  **data.frame(**  **RMSE = RMSE(predictions, test.data$medv),**  **R2 = R2(predictions, test.data$medv)**  **)**  **#-------------------------------------------------------------------------------**  **#Chapter10: Confounding Variables**  **library (gapminder)**  **lm(lifeExp ~ gdpPercap, data = gapminder)**  **lm(lifeExp ~ gdpPercap + continent, data = gapminder)**  **#-------------------------------------------------------------------------------**  **#IV Regression Model Validation**  **#Chapter11: Introduction**  **#Chapter12:Regression Model Accuracy Metrics**  **library (tidyverse)**  **library (modelr)**  **library (broom)**  **#Load the data**  **data("swiss")**  **# Inspect the data**  **sample\_n(swiss, 3)**  **model1 <- lm(Fertility ~.,data = swiss)**  **model2 <- lm(Fertility ~.-Examination, data = swiss)**  **summary (model1)**  **AIC(model1)**  **BIC(model1)**  **library (modelr)**  **data.frame(**  **R2 = rsquare(model1, data = swiss),**  **RMSE = rmse(model1, data = swiss),**  **MAE = mae(model1, data = swiss)**  **)**  **library(caret)**  **predictions <- model1 %>% predict (swiss)**  **data.frame(**  **R2 = R2(predictions, swiss$Fertility),**  **RMSE = RMSE(predictions, swiss$Fertility),**  **MAE = MAE(predictions, swiss$Fertility)**  **)**  **library (broom)**  **glance (model1)**  **# Make predictions and compute the a**  **# R2, RMSE and MAE**  **swiss %>%**  **add\_predictions(model1) %>%**  **summarise(**  **R2 = cor(Fertility, pred)^2,**  **MSE = mean((Fertility - pred)^2),**  **RMSE = sqrt (MSE),**  **MAE = mean(abs(Fertility - pred))**  **)**  **# Metrics for model 1**  **glance(model1) %>%**  **select(adj.r.squared, sigma, AIC, BIC, p.value)**  **# Metrics for model 2**  **glance(model2) %>%**  **select(adj.r.squared, sigma, AIC, BIC, p.value)**  **sigma(model1)/mean(swiss$Fertility)**  **#-------------------------------------------------------------------------------**  **#Chapter13:Cross-validation**  **library (tidyverse)**  **library (caret)**  **#Load the data**  **data("swiss")**  **# Inspect the data**  **sample\_n(swiss, 3)**  **# Split the data into training and test set a**  **set.seed(123)**  **training.samples <- swiss$Fertility %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- swiss[training.samples, ]**  **test.data <- swiss[-training.samples, ]**  **# Build the model**  **model <- lm(Fertility ~., data = swiss)**  **# Make predictions and compute the R2, RMSE and MAE**  **predictions <- model %>% predict (test.data)**  **data.frame( R2 = R2(predictions, test.data$Fertility),**  **RMSE = RMSE(predictions, test.data$Fertility),**  **MAE = MAE(predictions, test.data$Fertility))**  **RMSE = RMSE(predictions, test.data$Fertility)/mean(test.data$Fertility)**  **#Leave one out cross validation - LOOCV**  **# Define training control**  **train.control <- trainControl(method = "LOOCV")**  **# Train the model**  **model <- train(Fertility ~., data = swiss, method = "lm",**  **trControl = train.control)**  **# Summarize the results**  **print (model)**  **#K-fold cross-validation**  **# Define training control**  **set.seed(123)**  **train.control <- trainControl(method = "cv", number = 10)**  **# Train the model**  **model <- train(Fertility ~., data = swiss, method = "lm",**  **trControl = train.control)**  **# Summarize the results**  **print (model)**  **#Repeated K-fold cross-validation**  **# Define training control**  **set.seed (123)**  **train.control <- trainControl(method = "repeatedcv",**  **number = 10, repeats = 3)**  **# Train the model**  **model <- train(Fertility ~., data = swiss, method = "lm",**  **trControl = train.control)**  **# Summarize the results**  **print (model)**  **#-------------------------------------------------------------------------------**  **#Chapter14: Bootstrap Resampling**  **library (tidyverse)**  **library (caret)**  **#Load the data**  **data("swiss")**  **# Inspect the data**  **sample\_n(swiss, 3)**  **# Define training control**  **set.seed(123)**  **train.control <- trainControl(method = "boot", number = 100)**  **# Train the model**  **model <- train(Fertility ~., data = swiss, method = "lm",**  **trControl = train.control)**  **# Summarize the results**  **print (model)**  **model\_coef <- function(data, index){**  **coef(lm(Fertility ~., data = data, subset = index))**  **}**  **model\_coef(swiss, 1:47)**  **library(boot)**  **boot (swiss, model\_coef, 500)**  **summary(lm(Fertility ~., data = swiss))$coef**  **#-------------------------------------------------------------------------------**  **#V Model Selection**  **#Chapter15: Introduction**  **#Chapter16: Best Subsets Regression**  **library (tidyverse)**  **library (caret)**  **library (leaps)**  **#Load the data**  **data("swiss")**  **# Inspect the data**  **sample\_n(swiss, 3)**  **#Computing best subsets regression**  **models <- regsubsets(Fertility ~., data = swiss, nvmax = 5)**  **summary (models)**  **res.sum <- summary (models)**  **data.frame(**  **Adj.R2 = which.max(res.sum$adjr2),**  **CP = which.min(res.sum$cp),**  **BIC = which.min(res.sum$bic)**  **)**  **# id: model id**  **# object: regsubsets object**  **# data: data used to fit regsubsets**  **get\_model\_formula <- function(id, object){**  **# get models data**  **models <- summary (object) $which[id,-1]**  **# Get outcome variable**  **form <- as.formula(object$call[[2]])**  **outcome <- all.vars(form) [1]**  **# Get model predictors**  **predictors <- names(which(models == TRUE))**  **predictors <- paste(predictors, collapse = "+")**  **# Build model formula**  **as.formula(paste0(outcome, "~", predictors))**  **}**  **get\_model\_formula(3, models)**  **get\_cv\_error <- function(model.formula, data){**  **set.seed(1)**  **train.control <- trainControl(method = "cv", number = 5)**  **cv <- train(model.formula, data = data, method = "lm",**  **trControl = train.control)**  **cv$results$RMSE**  **}**  **# Compute cross-validation error**  **model.ids <- 1:5**  **cv.errors <- map(model.ids, get\_model\_formula, models) %>%**  **map(get\_cv\_error, data = swiss) %>%**  **unlist()**  **cv.errors**  **# Select the model that minimize the CV error**  **which.min(cv.errors)**  **coef (models, 4)**  **#-------------------------------------------------------------------------------**  **#Chapter17: Stepwise Regression**  **library (tidyverse)**  **library (caret)**  **library (leaps)**  **library (MASS)**  **# Fit the full model**  **full.model <- lm(Fertility ~., data = swiss)**  **#Stepwise regression model**  **step.model <- stepAIC(full.model, direction = "both",**  **trace = FALSE)**  **summary (step.model)**  **models <- regsubsets(Fertility~., data = swiss, nvmax = 5,**  **method = "seqrep")**  **summary (models)**  **# Set seed for reproducibility**  **set.seed(123)**  **# Set up repeated k-fold cross-validation**  **train.control <- trainControl (method = "cv", number = 10)**  **# Train the model**  **step.model <- train(Fertility ~., data = swiss,**  **method = "leapBackward",**  **tuneGrid = data.frame(nvmax = 1:5),**  **trControl = train.control**  **)**  **step.model$results**  **step.model$bestTune**  **summary(step.model$finalModel)**  **coef(step.model$finalModel,4)**  **lm(Fertility ~ Agriculture + Education + Catholic + Infant.Mortality, data = swiss)**  **library (MASS)**  **res.lm <- lm(Fertility ~., data = swiss)**  **step <- stepAIC(res.lm, direction = "both", trace = FALSE)**  **step**  **# Train the model**  **step.model <- train(Fertility ~., data = swiss,**  **method = "lmStepAIC",**  **trControl = train.control,**  **trace = FALSE**  **)**  **# Model accuracy**  **step.model$results**  **# Final model coefficients**  **step.model$finalModel**  **# Summary of the model**  **summary (step.model$finalModel)**  **#-------------------------------------------------------------------------------**  **#Chapter18: Penalized Regression: Ridge, Lasso and Elastic Net**  **library (tidyverse)**  **library (caret)**  **library (glmnet)**  **#Load the data**  **data ("Boston" , package = "MASS")**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- Boston$medv %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- Boston[training.samples, ]**  **test.data <- Boston[-training.samples, ]**  **#predictor variables**  **x<- model.matrix(medv~.,train.data)[,-1]**  **#outcome variable**  **y <- train.data$medv**  **glmnet(x,y,alpha = 1,lambda = NULL)**  **# Find the best lambda using cross-validation**  **set.seed (123)**  **cv <- cv.glmnet(x, y, alpha = 0)**  **# Display the best Lambda value**  **cv$lambda.min**  **# Fit the final model on the training data**  **model <- glmnet(x, y, alpha = 0, lambda = cv$lambda.min)**  **# Display regression coefficients**  **coef (model)**  **# Make predictions on the test data**  **x.test <- model.matrix(medv ~., test.data)[,-1]**  **predictions <- model %>% predict(x.test) %>% as.vector()**  **# Model performance metrics**  **data.frame(**  **RMSE = RMSE(predictions, test.data$medv),**  **Rsquare = R2(predictions, test.data$medv)**  **)**  **# Find the best lambda using cross-validation**  **set.seed (123)**  **cv <- cv.glmnet(x, y, alpha = 1)**  **# Display the best lambda value**  **cv$lambda.min**  **# Fit the final model on the training data**  **model <- glmnet(x, y, alpha = 1, lambda = cv$lambda.min)**  **# Dsiplay regression coefficients**  **coef (model)**  **# Make predictions on the test data**  **x.test <- model.matrix(medv ~., test.data)[,-1]**  **predictions <- model %>% predict(x.test) %>% as.vector()**  **# Model performance metrics**  **data.frame(**  **RMSE = RMSE(predictions, test.data$medv),**  **Rsquare = R2(predictions, test.data$medv)**  **)**  **# Build the model using the training set**  **set.seed (123)**  **model <- train(**  **medv ~., data = train.data, method = "glmnet",**  **trControl = trainControl("cv", number = 10),**  **tuneLength = 10**  **)**  **# Best tuning parameter**  **model$bestTune**  **# Coefficient of the final model.You need**  **# to specify the best lambda**  **coef (model$finalModel, model$bestTune$lambda)**  **# Make predictions on the test data**  **x.test <- model.matrix(medv ~., test.data)[,-1]**  **predictions <- model %>% predict (x.test)**  **# Model performance metrics**  **data.frame(**  **RMSE = RMSE(predictions, test.data$medv),**  **Rsquare = R2(predictions, test.data$medv)**  **)**  **lambda <- 10^seq(-3, 3, length = 100)**  **# Build the model**  **set.seed(123)**  **ridge <- train(**  **medv ~., data = train.data, method = "glmnet",**  **trControl = trainControl("cv", number = 10),**  **tuneGrid = expand.grid(alpha = 0, lambda = lambda)**  **)**  **# Model coefficients**  **coef (ridge$finalModel, ridge$bestTune$lambda)**  **# Make predictions**  **predictions <- ridge %>% predict (test.data)**  **# Model prediction performance**  **data.frame(**  **RMSE = RMSE(predictions, test.data$medv),**  **Rsquare = R2(predictions, test.data$medv)**  **)**  **# Build the model**  **set.seed(123)**  **lasso <- train(**  **medv~.,data=train.data,method ="glmnet",**  **trControl = trainControl("cv", number = 10),**  **tuneGrid = expand.grid(alpha = 1, lambda = lambda)**  **)**  **# model coefficients**  **coef (lasso$finalModel, lasso$bestTune$lambda)**  **# make predictions**  **predictions <- lasso %>% predict(test.data)**  **# Model prediction performance**  **data.frame(**  **RMSE = RMSE(predictions, test.data$medv),**  **Requare = R2(predictions, test.data$medv)**  **)**  **# Build the model**  **set.seed(123)**  **elastic <- train(**  **medv ~., data = train.data, method = "glmnet",**  **trControl = trainControl("cv", number = 10),**  **tuneLength = 10**  **)**  **# Model coefficients**  **coef (elastic$finalModel, elastic$bestTune$lambda)**  **# Make predictions**  **predictions <- elastic %>% predict (test.data)**  **# Model prediction performance**  **data.frame(**  **RMSE = RMSE(predictions, test.data$medv),**  **Rsquare = R2(predictions, test.data$medv)**  **)**  **models <- list (ridge = ridge, lasso = lasso, elastic = elastic)**  **resamples(models) %>% summary( metric = "RMSE")**  **#-------------------------------------------------------------------------------**  **#Chapter19: Principal Component and Partial Least Squares Regression**  **library (tidyverse)**  **library (caret)**  **library (pls)**  **#Load the data**  **data ("Boston" , package = "MASS")**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- Boston$medv %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- Boston[training.samples, ]**  **test.data <- Boston[-training.samples, ]**  **#Computing principal component regression**  **#Build the model using the training set**  **set.seed (123)**  **model <- train(**  **medv ~., data = train.data, method = "pcr",**  **scale = TRUE,**  **trControl = trainControl("cv", number = 10),**  **tuneLength = 10**  **)**  **#plot model RMSE vs different values of components**  **plot (model)**  **#print the best tuning parameter ncomp that**  **#minimize the cross-validation error,RMSE**  **model$bestTune**  **#summarize the final model**  **summary(model$finalmodel)**  **# Make predictions**  **predictions <- model %>% predict (test.data)**  **# Model prediction Metrics**  **data.frame(**  **RMSE = caret::RMSE(predictions, test.data$medv),**  **Rsquare = caret::R2(predictions, test.data$medv)**  **)**  **#Computing partial least squares**  **# Build the model using the training set**  **set.seed (123)**  **model <- train(**  **medv ~., data = train.data, method = "pls",**  **scale = TRUE,**  **trControl = trainControl("cv", number = 10),**  **tuneLength = 10**  **)**  **#plot model RMSE vs different values of components**  **plot (model)**  **#print the best tuning parameter ncomp that**  **#minimize the cross-validation error,RMSE**  **model$bestTune**  **#summarize the final model**  **summary(model$finalmodel)**  **# Make predictions**  **predictions <- model %>% predict (test.data)**  **# Model performance metrics**  **data.frame(**  **RMSE = caret::RMSE(predictions, test.data$medv),**  **Requare = caret::R2(predictions, test.data$medv)**  **)**  **#-------------------------------------------------------------------------------**  **#VI Classification**  **#Chapter20: Introduction**  **#install.packages('mlbench')**  **library(mlbench)**  **#Load the data**  **data ("PimaIndiansDiabetes2" , package = "mlbench")**  **# Inspect the data**  **head(PimaIndiansDiabetes2,4)**  **#Load the data**  **data("iris")**  **# Inspect the data**  **head(iris,4)**  **#-------------------------------------------------------------------------------**  **#Chapter21: Logistic Regression**  **library (tidyverse)**  **library (caret)**  **theme\_set(theme\_bw())**  **#Load the data**  **data("PimaIndiansDiabetes2", package = "mlbench")**  **PimaIndiansDiabetes2 <- na.omit(PimaIndiansDiabetes2)**  **# Inspect the data**  **sample\_n(PimaIndiansDiabetes2, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- PimaIndiansDiabetes2$diabetes %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- PimaIndiansDiabetes2[training.samples, ]**  **test.data <- PimaIndiansDiabetes2[-training.samples, ]**  **# Fit the model**  **model <- glm( diabetes ~., data = train.data, family = binomial)**  **# Summarize the model**  **summary (model)**  **#Make predictions**  **probabilities <- model %>% predict (test.data, type = "response")**  **predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")**  **# Model accuracy**  **mean (predicted.classes == test.data$diabetes)**  **#Simple logistic regression**  **model <- glm( diabetes ~ glucose, data = train.data, family = binomial)**  **summary(model)$coef**  **newdata <- data.frame(glucose = c(20, 180))**  **probabilities <- model %>% predict (newdata, type = "response")**  **predicted.clasees <- ifelse (probabilities > 0.5,"pos","neg")**  **predicted.classes**  **train.data %>% #?**  **mutate (prob = ifelse(diabetes == "pos",1,0)) %>%**  **ggplot(aes(glucose, prob)) +**  **geom\_point(alpha = 0.2) +**  **geom\_smooth(method = "glm", method.arge = list(family="binomial"))+**  **labs(**  **title = "Logistic Regression Model",**  **x= "Plasma Glucose Concentration",**  **y = "Probability of being diabete-pos"**  **)**  **model <- glm(diabetes~glucose+mass+pregnant,data = train.data,family=binomial)**  **summary(model)$coef**  **model <- glm(diabetes~.,data = train.data, family = binomial)**  **summary(model)$coef**  **coef(model)**  **summary(model)$coef**  **model <- glm(diabetes~pregnant+glucose+pressure+mass+pedigree,data = train.data,family=binomial)**  **probabilities <- model %>% predict (test.data, type = "response")**  **head(probabilities)**  **contrasts(test.data$diabetes)**  **predicted.clasees <- ifelse (probabilities > 0.5,"pos","neg")**  **head(predicted.classes)**  **mean(predicted.classes,test.data$diabetes)**  **library("mgcv")**  **# Fit the model**  **gam.model <- gam( diabetes ~s(glucose)+mass+pregnant, data = train.data, family = "binomial")**  **# Summarize the model**  **summary (gam.model)**  **#Make predictions**  **probabilities <- gam.model %>% predict (test.data, type = "response")**  **predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")**  **# Model accuracy**  **mean (predicted.classes == test.data$diabetes)**  **#-------------------------------------------------------------------------------**  **#Chapter22: Stepwise Logistic Regression**  **library (tidyverse)**  **library (caret)**  **#Load the data AND remove NAs**  **data("PimaIndiansDiabetes2", package = "mlbench")**  **PimaIndiansDiabetes2 <- na.omit(PimaIndiansDiabetes2)**  **# Inspect the data**  **sample\_n(PimaIndiansDiabetes2, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- PimaIndiansDiabetes2 $ diabetes %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- PimaIndiansDiabetes2[training.samples, ]**  **test.data <- PimaIndiansDiabetes2[-training.samples, ]**  **library("MASS")**  **# Fit the model**  **model <- glm( diabetes ~., data = train.data, family = binomial) %>%**  **stepAIC(trace=FALSE)**  **# Summarize the final selected model**  **summary (model)**  **#Make predictions**  **probabilities <- model %>% predict (test.data, type = "response")**  **predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")**  **# Model accuracy**  **mean (predicted.classes == test.data$diabetes)**  **full.model <-glm( diabetes ~., data = train.data, family = binomial)**  **coef(full.model)**  **library("MASS")**  **step.model <- full.model %>% stepAIC(trace=FALSE)**  **coef(step.model)**  **# Make predictions**  **probabilities <- full.model %>% predict(test.data, type = "response")**  **predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")**  **# Prediction accuracy**  **observed.classes <- test.data$diabetes**  **mean (predicted.classes == observed.classes)**  **# Make predictions**  **probabilities <- predict(step.model, test.data, type = "response")**  **predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")**  **# Prediction accuracy**  **observed.classes <- test.data$diabetes**  **mean(predicted.classes == observed.classes)**  **#-------------------------------------------------------------------------------**  **#Chapter23: Penalized Logistic Regression**  **library (tidyverse)**  **library (caret)**  **library (glmnet)**  **#Load the data AND remove NAs**  **data("PimaIndiansDiabetes2", package = "mlbench")**  **PimaIndiansDiabetes2 <- na.omit(PimaIndiansDiabetes2)**  **# Inspect the data**  **sample\_n(PimaIndiansDiabetes2, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- PimaIndiansDiabetes2 $ diabetes %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- PimaIndiansDiabetes2[training.samples, ]**  **test.data <- PimaIndiansDiabetes2[-training.samples, ]**  **# Dumy code categorical predictor variables**  **x <- model.matrix(diabetes~., train.data)[,-1]**  **# Convert the outcome (class) to a numerical variable**  **y <- ifelse(train.data$diabetes == "pos", 1, 0)**  **glmnet(x, y, family = "binomial", alpha = 1, lambda = NULL)**  **library (glmnet)**  **# Find the best Lambda using cross-validation**  **set.seed (123)**  **cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial")**  **# Fit the final model on the training data**  **model <- glmnet(x, y, alpha = 1, family = "binomial")**  **lambda = cv.lasso$lambda.min**  **# Display regression coefficients**  **coef (model)**  **# Make predictions on the test data**  **x.test <- model.matrix(diabetes ~., test.data)[,-1]**  **probabilities <- model %>% predict(newx = x.test)**  **predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")**  **# Model accuracy**  **observed.classes <- test.data$diabetes**  **mean(predicted.classes == observed.classes)**  **library (glmnet)**  **set.seed (123)**  **cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial")**  **plot(cv.lasso)**  **cv.lasso$lambda.min**  **cv.lasso$lambda.1se**  **coef(cv.lasso,cv.lasso$lambda.min)**  **coef(cv.lasso,cv.lasso$lambda.1se)**  **# Final model with lambda.min**  **lasso.model <- glmnet(x, y, alpha = 1, family = "binomial",**  **lambda = cv.lasso$lambda.min)**  **# Make prediction on test data**  **x.test <- model.matrix (diabetes ~., test.data)[,-1]**  **probabilities <- lasso.model %>% predict (newx = x.test)**  **predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")**  **# Model accuracy**  **observed.classes <- test.data$diabetes**  **mean (predicted.classes == observed.classes)**  **# Final model with lambda.1se**  **lasso.model <- glmnet(x, y, alpha = 1, family = "binomial",**  **lambda = cv.lasso$lambda.1se)**  **# Make prediction on test data**  **x.test <- model.matrix (diabetes ~., test.data)[,-1]**  **probabilities <- lasso.model %>% predict (newx = x.test)**  **predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")**  **# Model accuracy rate**  **observed.classes <- test.data$diabetes**  **mean(predicted.classes == observed.classes)**  **# Fit the model**  **full.model <- glm(diabetes ~., data = train.data, family = binomial)**  **# Make predictions**  **probabilities <- full.model %>% predict(test.data, type = "response")**  **predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")**  **# Model accuracy**  **observed.classes <- test.data$diabetes**  **mean(predicted.classes == observed.classes)**  **#-------------------------------------------------------------------------------**  **#Chapter24: Logistic Regression Assumptions and Diagnostics**  **library(tidyverse)**  **library(broom)**  **theme\_set(theme\_classic())**  **#Load the data**  **data("PimaIndiansDiabetes2", package = "mlbench")**  **PimaIndiansDiabetes2 <- na.omit(PimaIndiansDiabetes2)**  **# Fit the logistic regression model**  **model <- glm( diabetes ~., data = PimaIndiansDiabetes2 , family = binomial)**  **#predict the probability (p) of diabete positivity**  **probabilities <- predict (model, type = "response")**  **predicted.clasees <- ifelse (probabilities > 0.5,"pos","neg")**  **head(predicted.clasees)**  **# Select only numeric predictors**  **mydata <- PimaIndiansDiabetes2 %>%**  **dplyr::select\_if(is.numeric)**  **predictors <- colnames(mydata)**  **# Bind the logit and tidying the data for plot**  **mydata <- mydata %>%**  **mutate(logit = log(probabilities/(1-probabilities))) %>%**  **gather(key = "predictors", value = "predictor.value", -logit)**  **ggplot(mydata, aes(logit, predictor.value))+**  **geom\_point(size = 0.5, alpha = 0.5) +**  **geom\_smooth(method = "loess") +**  **theme\_bw() +**  **facet\_wrap(~predictors, scales = "free\_y")**  **plot(model, which = 4 , id.n=3)**  **# Extract model results**  **model.data <- augment (model) %>%**  **mutate(index = 1:n())**  **model.data %>% top\_n(3,.cooksd)**  **ggplot(model.data, aes(index,.std.resid)) +**  **geom\_point(aes(color = diabetes), alpha =.5) +**  **theme\_bw()**  **model.data %>%**  **filter(abs(.sdt.resid)>3)**  **car::vif(model)**  **#-------------------------------------------------------------------------------**  **#Chapter25: Multinomial Logistic Regression**  **library (tidyverse)**  **library (caret)**  **library (nnet)**  **#Load the data**  **data("iris")**  **# Inspect the data**  **sample\_n(iris, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- iris$Species %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- iris[training.samples, ]**  **test.data <- iris[-training.samples, ]**  **# Fit the model**  **model <- nnet::multinom(Species ~., data = train.data)**  **# Summarize the model**  **summary (model)**  **# Make predictions**  **predicted.classes <- model %>% predict(test.data)**  **head (predicted.classes)**  **# Model accuracy**  **mean(predicted.classes == test.data$Species)**  **mean(predicted.classes == test.data$Species)**    **#-------------------------------------------------------------------------------**  **#Chapter26: Discriminant Analysis**  **library(tidyverse)**  **library(caret)**  **theme\_set(theme\_classic())**  **#Load the data**  **data("iris")**  **# Inspect the data**  **sample\_n(iris, 3)**  **# Split the data into training (80%)and test set(20%)**  **set.seed(123)**  **training.samples <- iris$Species %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- iris[training.samples, ]**  **test.data <- iris[-training.samples, ]**  **# Estimate preprocessing parameters**  **preproc.param <- train.data %>%**  **preProcess(method = c("center", "scale"))**  **# Transform the data using the estimated parameters**  **train.transformed <- preproc.param %>% predict (train.data)**  **test.transformed <- preproc.param %>% predict (test.data)**  **library (MASS)**  **# Fit the model**  **model <- lda(Species~., data = train.transformed)**  **# Make predictions**  **predictions <- model %>% predict (test.transformed)**  **# Model accuracy**  **mean (predictions$class==test.transformed$Species)**  **library (MASS)**  **model <- lda(Species~., data = train.transformed)**  **model**  **plot(model)**  **predictions <- model %>% predict(test.transformed)**  **names(predictions)**  **# Predicted classes**  **head(predictions$class, 6)**  **# Predicted probabilities of class memebership.**  **head(predictions$posterior, 6)**  **# Linear discriminants**  **head(predictions$x, 3)**  **lda.data <- cbind(train.transformed, predict (model)$x)**  **ggplot(lda.data, aes(LD1, LD2)) +**  **geom\_point(aes(color = Species))**  **mean(predictions$class==test.transformed$species)**  **sum(predictions$posterior[ ,1] >=.5)**  **library (MASS)**  **# Fit the model**  **model <- qda(Species~., data = train.transformed)**  **model**  **# Make predictions**  **predictions <- model %>% predict (test.transformed)**  **# Model accuracy**  **mean(predictions$class == test.transformed$Species)**  **library (mda)**  **# Fit the model**  **model <- mda(Species~., data = train.transformed)**  **model**  **# Make predictions**  **predicted.classes <- model %>% predict(test.transformed)**  **# Model accuracy**  **mean(predicted.classes == test.transformed$Species)**  **library (mda)**  **# Fit the model**  **model <- mda(Species~., data = train.transformed)**  **model**  **# Make predictions**  **predicted.classes <- model %>% predict(test.transformed)**  **# Model accuracy**  **mean(predicted.classes == test.transformed$Species)**  **library (klaR)**  **# Fit the model**  **model <- rda(Species~., data = train.transformed)**  **# Make predictions**  **predictions <- model %>% predict (test.transformed)**  **# Model accuracy**  **mean(predictions$class == test.transformed$Species)**  **#-------------------------------------------------------------------------------**  **#Chapter27: Naive Bayes Classifier**  **library (tidyverse)**  **library (caret)**  **#Load the data AND remove NAs**  **data("PimaIndiansDiabetes2", package = "mlbench")**  **PimaIndiansDiabetes2 <- na.omit(PimaIndiansDiabetes2)**  **# Inspect the data**  **sample\_n(PimaIndiansDiabetes2, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- PimaIndiansDiabetes2 $ diabetes %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- PimaIndiansDiabetes2[training.samples, ]**  **test.data <- PimaIndiansDiabetes2[-training.samples, ]**  **library (klaR)**  **# Fit the model**  **model <- NaiveBayes(diabetes ~., data = train.data)**  **# Make predictions**  **predictions <- model %>% predict (test.data)**  **# Model accuracy**  **mean(predictions$class == test.data$diabetes)**  **#Using caret R package**  **library (klaR)**  **# Build the model**  **set.seed(123)**  **model <- train(diabetes ~., data = train.data,method ="nb",**  **trControl = trainControl("cv", number=10))**  **# Make predictions**  **predicted.classes <- model %>% predict (test.data)**  **# Model n accuracy**  **mean(predicted.clasees == test.data$diabetes)**  **#-------------------------------------------------------------------------------**  **#Chapter28: Support Vector Machine**  **library (tidyverse)**  **library (caret)**  **#Load the data AND remove NAs**  **data("PimaIndiansDiabetes2", package = "mlbench")**  **pima.data <- na.omit(PimaIndiansDiabetes2)**  **# Inspect the data**  **sample\_n(pima.data, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- pima.data $ diabetes %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- pima.data [training.samples, ]**  **test.data <- pima.data [-training.samples, ]**  **# Fit the model on the training set**  **set.seed(123)**  **model <- train(**  **diabetes ~., data = train.data, method = "svmLinear",**  **trControl = trainControl("cv", number = 10),**  **preProcess = c("center","scale")**  **)**  **# Make predictions on the test data**  **predicted.classes <- model %>% predict(test.data)**  **head(predicted.classes)**  **# Compute model accuracy rate**  **mean(predicted.classes == test.data$diabetes)**  **# Fit the model on the training set**  **set.seed(123)**  **model <- train(**  **diabetes ~., data = train.data, method = "svmLinear",**  **trControl = trainControl("cv", number = 10),**  **tuneGrid = expand.grid(C = seq(0, 2, length = 20)),**  **preProcess = c("center","scale")**  **)**  **# Plot model accuracy vs different values of Cost**  **plot (model)**  **# Print the best tuning parameter C that**  **# maximizes model accuracy**  **model$bestTune**  **# Make predictions on the test data**  **predicted.classes <- model %>% predict (test.data)**  **# Compute model accuracy rate**  **mean(predicted.classes == test.data$diabetes)**  **#Fit the model on the training set**  **set.seed (123)**  **model <- train(**  **diabetes ~., data = train.data, method = "svmRadial",**  **trControl = trainControl("cv", number = 10),**  **preProcess = c("center","scale"),**  **tuneLength = 10**  **)**  **# Print the best tuning parameter sigma and C that**  **# maximizes model accuracy**  **model$bestTune**  **# Make predictions on the test data**  **predicted.classes <- model %>% predict (test.data)**  **# Compute model accuracy rate**  **mean(predicted.classes == test.data$diabetes)**  **# Fit the model on the training set**  **set.seed (123)**  **model <- train(**  **diabetes ~., data = train.data, method = "svmPoly",**  **trControl = trainControl("cv", number = 10),**  **preProcess = c("center","scale"),**  **tuneLength = 4**  **)**  **# Print the best tuning parameter sigma and C that**  **# maximizes model accuracy**  **model$bestTune**  **# Make predictions on the test data**  **predicted.classes <- model %>% predict (test.data)**  **# Compute model accuracy rate**  **mean(predicted.classes == test.data$diabetes)**  **#-------------------------------------------------------------------------------**  **#Chapter29: Classification Model Evaluation**  **library (tidyverse)**  **library (caret)**  **# Load the data**  **data("PimaIndiansDiabetes2", package = "mlbench")**  **PimaIndiansDiabetes2 <- na.omit(PimaIndiansDiabetes2)**  **# Inspect the data**  **sample\_n(pima.data, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples < pima.data$diabetes %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- pima.data[training.samples, ]**  **test.data <- pima.data[-training.samples, ]**  **library (MASS)**  **# Fit LDA**  **fit <- lda(diabetes ~., data = train.data)**  **# Make predictions on the test data**  **predictions <- predict(fit, test.data)**  **prediction.probabilities <- predictions$posterior[,2]**  **predicted.classes <- predictions$class**  **observed.classes <- test.data$diabetes**  **accuracy <- mean(observed.classes == predicted.classes)**  **accuracy**  **error <- mean(observed.classes != predicted.classes)**  **error**  **# Confusion matriz, number of cases**  **table(observed.classes, predicted.classes)**  **# Confusion matric, proportion of cases**  **table(observed.classes, predicted.classes) %>%**  **prop.table() %>% round(digits = 3)**  **confusionMatrix(predicted.classes, observed.classes,**  **positive = "pos")**  **library (pROC)**  **# Compute roc**  **res.roc <- roc(observed.classes, prediction.probabilities)**  **plot.roc(res.roc, print.auc=TRUE)**  **# Extract some interesting results**  **roc.data <- data\_frame(**  **thresholds = res.roc$thresholds,**  **sensitivity = res.roc$sensitivities,**  **specificity = res.roc$specificities**  **)**  **# Get the probality threshold for specificity = 0.6**  **roc.data %>% filter(specificity >= 0.6)**  **plot.roc(res.roc, print.auc = TRUE, print.thres = "best")**  **plot.roc(res.roc, print.thres = c(0.3, 0.5, 0.7))**  **# Create some grouping variable**  **glucose <- ifelse(test.data$glucose < 127.5, "glu.low", "glu.high")**  **age <- ifelse(test.data$age < 28.5, "young", "old")**  **roc.data <- roc.data %>%**  **filter(thresholds !=-Inf) %>%**  **mutate (glucose = glucose, age = age)**  **# Create ROC curve**  **ggplot(roc.data, aes(specificity, sensitivity)) +**  **geom\_path(aes(color = age))+**  **scale\_x\_reverse(expand = c(0,0))+**  **scale\_y\_continuous(expand = c(0,0))+**  **geom\_abline(intercept = 1, slope = 1, linetype = "dashed")+**  **theme\_bw()**  **# Load the data**  **data("iris")**  **# Split the data into training (80%) and test set (20%)**  **set.seed(123)**  **training.samples <- iris$Species %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- iris[training.samples, ]**  **test.data <- iris[-training.samples, ]**  **# Build the model on the train set**  **library (MASS)**  **model <- lda(Species ~., data = train.data)**  **# Make predictions on the test data**  **predictions <- model %>% predict (test.data)**  **# Model accuracy**  **confusionMatrix(predictions$class, test.data$Species)**  **#-------------------------------------------------------------------------------**  **#VII Statistical Machine Learning**  **#Chapter30: Introduction**  **#Chapter31: KNN - k-Nearest Neighbors**  **library (tidyverse)**  **library (caret)**  **#Load the data AND remove NAs**  **data("PimaIndiansDiabetes2", package = "mlbench")**  **PimaIndiansDiabetes2 <- na.omit(PimaIndiansDiabetes2)**  **# Inspect the data**  **sample\_n(PimaIndiansDiabetes2, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- PimaIndiansDiabetes2$ diabetes %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- PimaIndiansDiabetes2 [training.samples, ]**  **test.data <- PimaIndiansDiabetes2 [-training.samples, ]**  **#Computing KNN classifier**  **# Fit the model on the training set**  **set.seed(123)**  **model <- train(**  **diabetes ~., data = train.data, method = "knn",**  **trControl = trainControl("cv", number = 10),**  **preProcess = c("center","scale"),**  **tuneLength = 20**  **)**  **# Plot model accuracy vs different values of k**  **plot (model)**  **# Print the best tuning parameter k that**  **# maximizes model accuracy**  **model$bestTune**  **# Make predictions on the test data**  **predicted.classes <- model %>% predict (test.data)**  **head(predicted.classes)**  **# Compute model accuracy rate**  **mean(predicted.classes == test.data$diabetes)**  **#KNN for regression**  **#Load the data**  **data ("Boston" , package = "MASS")**  **# Inspect the data**  **sample\_n(Boston, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- Boston$medv %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- Boston[training.samples, ]**  **test.data <- Boston[-training.samples, ]**  **# Fit the model on the training set**  **set.seed(123)**  **model <- train(**  **medv ~., data = train.data, method = "knn",**  **trControl = trainControl("cv", number = 10),**  **preProcess = c("center","scale"),**  **tuneLength = 10**  **)**  **# Plot model error RMSE vs different values of k**  **plot (model)**  **# Best tuning parameter k that minimize the RMSE**  **model$bestTune**  **# Make predictions on the test data**  **predictions <- model %>% predict(test.data)**  **head(predictions)**  **# Compute the prediction error RMSE**  **RMSE(predictions, test.data$medv)**  **#-------------------------------------------------------------------------------**  **#Chapter32: Decision Tree Models**  **library (tidyverse)**  **library (caret)**  **library (rpart)**  **model <- rpart (Species ~., data = iris)**  **par(xpd = NA) # otherwise on some devices the text is clipped**  **plot (model)**  **text(model, digits = 3)**  **print(model, digits = 2)**  **newdata <- data.frame(**  **Sepal.Length = 6.5 , Sepal.Width= 3.0 ,**  **Petal.Length = 5.2 , Petal.Width=2.0**  **)**  **model %>% predict(newdata, "class")**  **# Load the data and remove NAs**  **data("PimaIndiansDiabetes2", package = "mlbench")**  **PimaIndiansDiabetes2 <- na.omit (PimaIndiansDiabetes2)**  **# Inspect the data**  **sample\_n(PimaIndiansDiabetes2, 3)**  **# Split the data into training and test set**  **set.seed (123)**  **training.samples <- PimaIndiansDiabetes2$diabetes %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- PimaIndiansDiabetes2[training.samples, ]**  **test.data <- PimaIndiansDiabetes2[-training.samples, ]**  **# Build the model**  **set.seed(123)**  **model1 <- rpart(diabetes ~., data = train.data, method = "class")**  **# Plot the trees**  **par(xpd = NA) # Avoid clipping the text in some device**  **plot (model1)**  **text (model1, digits = 3)**  **# Make predictions on the test data**  **predicted.classes <- model1 %>%**  **predict(test.data, type = "class")**  **head (predicted.classes)**  **# Compute model accuracy rate on test data**  **mean(predicted.classes == test.data$diabetes)**  **# Fit the model on the training set**  **set.seed (123)**  **model2 <- train(**  **diabetes ~., data = train.data, method = "rpart",**  **trControl = trainControl("cv", number = 10),**  **tuneLength = 10**  **)**  **# Plot model accuracy vs different values of**  **# cp (complexity parameter)**  **plot (model2)**  **# Print the best tuning parameter cp that**  **# maximizes the model accuracy**  **model2$bestTune**  **# Plot the final tree model**  **par(xpd = NA) # Avoid clipping the text in some device**  **plot (model2$finalModel)**  **text(model2$finalModel, digits = 3)**  **#Decision rules in the model**  **model2$finalModel**  **# Make predictions on the test data**  **predicted.classes <- model2 %>% predict (test.data)**  **# Compute model accuracy rate on test data**  **mean(predicted.classes == test.data$diabetes)**  **# Load the data**  **data("Boston", package = "MASS")**  **# Inspect the data**  **sample\_n(Boston, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- Boston$medv %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- Boston[training.samples, ]**  **test.data <- Boston[-training.samples, ]**  **# Fit the model on the training set**  **set.seed (123)**  **model <- train(**  **medv ~., data = train.data, method = "rpart",**  **trControl = trainControl("cv", number = 10),**  **tuneLength = 10**  **)**  **# Plot model error vs different values of**  **# cp (complexity parameter)**  **plot (model)**  **# Print the best tuning parameter cp that**  **# minimize the model RMSE**  **model$bestTune**  **# Plot the final tree model**  **par(xpd = NA) # Avoid clipping the text in some device**  **plot (model$finalModel)**  **text (model$finalModel, digits = 3)**  **# Decision rules in the model**  **model$finalModel**  **# Make predictions on the test data**  **predictions <- model %>% predict (test.data)**  **head (predictions)**  **# Compute the prediction error RMSE**  **RMSE (predictions, test.data$medv)**  **# Load the data**  **data("PimaIndiansDiabetes2", package = "mlbench")**  **pima.data <- na.omit (PimaIndiansDiabetes2)**  **# Split the data into training and test set**  **set.seed (123)**  **training.samples <- pima.data$diabetes %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- pima.data[training.samples, ]**  **test.data <- pima.data[-training.samples, ]**  **library (party)**  **set.seed (123)**  **model <- train(**  **diabetes ~., data = train.data, method = "ctree2",**  **trControl = trainControl("cv", number = 10),**  **tuneGrid = expand.grid(maxdepth = 3, mincriterion = 0.95 )**  **)**  **plot (model$finalModel)**  **# Make predictions on the test data**  **predicted.classes <- model %>% predict (test.data)**  **# Compute model accuracy rate on test data**  **mean(predicted.classes == test.data$diabetes)**  **#-------------------------------------------------------------------------------**  **#Chapter33: Bagging and Random Forest**  **library (tidyverse)**  **library (caret)**  **library (randomForest)**  **#Load the data AND remove NAs**  **data("PimaIndiansDiabetes2", package = "mlbench")**  **PimaIndiansDiabetes2 <- na.omit(PimaIndiansDiabetes2)**  **# Inspect the data**  **sample\_n(PimaIndiansDiabetes2, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- PimaIndiansDiabetes2$ diabetes %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- PimaIndiansDiabetes2 [training.samples, ]**  **test.data <- PimaIndiansDiabetes2 [-training.samples, ]**  **# Fit the model on the training set**  **set.seed(123)**  **model <- train(**  **diabetes ~., data = train.data, method = "rf",**  **trControl = trainControl("cv", number = 10),**  **importance = TRUE**  **)**  **#Best tuning parameter**  **model$bestTune**  **#Final model**  **model$finalModel**  **# Make predictions on the test data**  **predicted.classes <- model %>% predict (test.data)**  **head(predicted.classes)**  **# Compute model accuracy rate**  **mean(predicted.classes == test.data$diabetes)**  **importance(model$finalModel)**  **# Plot MeanDecreaseAccuracy**  **varImpPlot (model$finalModel, type=1)**  **# Plot MeanDecreaseGint**  **varImpPlot (model$finalModel, type=2)**  **varImp(model)**  **# Load the data**  **data("Boston", package = "MASS")**  **# Inspect the data**  **sample\_n(Boston, 3)**  **# Split the data into training and test set**  **set.seed (123)**  **training.samples <- Boston$medv %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- Boston[training.samples, ]**  **test.data <- Boston[-training.samples, ]**  **# Fit the model on the training set**  **set.seed (123)**  **model <- train(**  **medv ~., data = train.data, method = "rf",**  **trControl = trainControl("cv", number = 10)**  **)**  **# Best tuning parameter mtry**  **model$bestTune**  **# Make predictions on the test data**  **predictions <- model %>% predict (test.data)**  **head(predictions)**  **# Compute the average prediction error RMSE**  **RMSE(predictions, test.data$medv)**  **data("PimaIndiansDiabetes2", package = "mlbench")**  **models <- list()**  **for (nodesize in c(1, 2, 4, 8)) {**  **set.seed(123)**  **model <- train(**  **diabetes~., data = na.omit(PimaIndiansDiabetes2), method="rf",**  **trControl = trainControl(method="cv", number=10) ,**  **metric = "Accuracy",**  **nodesize = nodesize**  **)**  **model.name <- toString (nodesize)**  **models [[model.name]] <- model**  **}**  **# Compare results**  **resamples(models) %>% summary(metric = "Accuracy")**  **#-------------------------------------------------------------------------------**  **#Chapter34: Boosting**  **library (tidyverse)**  **library (caret)**  **library (xgboost)**  **#Load the data AND remove NAs**  **data("PimaIndiansDiabetes2", package = "mlbench")**  **PimaIndiansDiabetes2 <- na.omit(PimaIndiansDiabetes2)**  **# Inspect the data**  **sample\_n(PimaIndiansDiabetes2, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- PimaIndiansDiabetes2$ diabetes %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- PimaIndiansDiabetes2 [training.samples, ]**  **test.data <- PimaIndiansDiabetes2 [-training.samples, ]**  **# Fit the model on the training set**  **set.seed(123)**  **model <- train(**  **diabetes ~., data = train.data, method = "xgbTree",**  **trControl = trainControl("cv", number = 10),**  **)**  **#Best tuning parameter**  **model$bestTune**  **# Make predictions on the test data**  **predicted.classes <- model %>% predict (test.data)**  **head(predicted.classes)**  **# Compute model prediction accuracy rate**  **mean(predicted.classes == test.data$diabetes)**  **varImp(model)**  **# Load the data**  **data("Boston", package = "MASS")**  **# Inspect the data**  **sample\_n(Boston, 3)**  **# Split the data into training and test set**  **set.seed(123)**  **training.samples <- Boston$medv %>%**  **createDataPartition(p = 0.8, list = FALSE)**  **train.data <- Boston[training.samples, ]**  **test.data <- Boston[-training.samples, ]**  **# Fit the model on the training set**  **set.seed (123)**  **model <- train(**  **medv ~., data = train.data, method = "xgbTree",**  **trControl = trainControl("cv", number = 10)**  **)**  **# Best tuning parameter mtry**  **model$bestTune**  **# Make predictions on the test data**  **predictions <- model %>% predict (test.data)**  **head(predictions)**  **# Compute the average prediction error RMSE**  **RMSE(predictions, test.data$medv)**  **#-------------------------------------------------------------------------------**  **#VIII Unsupervised Learning**  **#Chapter35: Unsupervised Learning**  **library (FactoMineR)**  **library (factoextra)**  **data("USArrests")**  **res.pca <- PCA(USArrests, graph = FALSE)**  **fviz\_eig(res.pca)**  **fviz\_pca\_ind(res.pca, repel = TRUE)**  **fviz\_pca\_var(res.pca)**  **fviz\_pca\_biplot(res.pca, repel = TRUE)**  **# Eigenvalues**  **res.pca$eig**  **# Results for Variables**  **res.var <- res.pca$var**  **res.var$coord # Coordinates**  **res.var$contrib # Contributions to the PCs**  **res.var$cos2 # Quality of representation**  **# Results for individuals**  **res.ind <- res.pca$var**  **res.ind$coord # Coordinates**  **res.ind$contrib # Contributions to the PCs**  **res.ind$cos2 # Quality of representation**  **# Load and inspect the data set**  **data("housetasks")**  **head(housetasks, 4)**  **# Compute correspondence analysis**  **res.ca <- CA(housetasks, graph = FALSE)**  **fviz\_ca\_biplot(res.ca, repel = TRUE)**  **# Load data**  **data("poison")**  **head(poison[, 1:8], 4)**  **# Compute multiple correspondence analysis:**  **res.mca <- MCA(poison, quanti.sup = 1:2,**  **quali.sup = 3:4, graph=FALSE)**  **# Graph of individuals, colored by groups ("Sick")**  **fviz\_mca\_ind(res.mca, repel = TRUE, habillage = "Sick",**  **addEllipses = TRUE)**  **# Graph of variable categories**  **fviz\_mca\_var(res.mca, repel = TRUE)**  **fviz\_mca\_biplot(res.mca, repel = TRUE,**  **ggtheme=theme\_minimal())**  **library (cluster)**  **library (factoextra)**  **mydata <- scale(USArrests)**  **fviz\_nbclust(mydata,kmeans,method="gap\_stat")**  **set.seed(123) # for reproducibility**  **km.res <- kmeans(mydata, 3, nstart = 25)**  **# Visualize**  **fviz\_cluster(km.res, data = mydata, palette = "jco",**  **ggtheme = theme\_minimal())**  **pam.res <- pam(mydata, 3)**  **fviz\_cluster (pam.res)**  **res.hc <- hclust (dist (mydata), method ="ward.D2")**  **fviz\_dend(res.hc, cex = 0.5, k = 4, palette = "jco")**  **library (pheatmap)**  **pheatmap(t(mydata), cutree\_cols = 4)** |

1. <http://r4ds.had.co.nz/transform.html> [↑](#footnote-ref-1)
2. http://www.sthda.com/english/articles/32-r-graphics-essentials/ [↑](#footnote-ref-2)
3. https://en.wikipedia.org/wiki/All\_models\_are\_wrong [↑](#footnote-ref-3)
4. [http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical- guide/112-pca-principal-component-analysis-essentials/](http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/112-pca-principal-component-analysis-essentials/) [↑](#footnote-ref-4)
5. <https://www.r-bloggers.com/a-brief-look-at-mixture-discriminant-analysis/> [↑](#footnote-ref-5)
6. https://en.wikipedia.org/wiki/Support\_vector\_machine [↑](#footnote-ref-6)
7. https://en.wikipedia.org/wiki/Cohen%27s\_kappa [↑](#footnote-ref-7)
8. http://www.sthda.com/english/articles/26-clustering-basics/86-clustering-distance-measures- Essentials/ [↑](#footnote-ref-8)
9. <http://www.sthda.com/english/articles/25-cluster-analysis-in-r-practical-guide/> [↑](#footnote-ref-9)
10. http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/ [↑](#footnote-ref-10)
11. http://www.sthda.com/english/articles/27-partitioning-clustering-essentials/ [↑](#footnote-ref-11)
12. http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/ [↑](#footnote-ref-12)
13. http://www.sthda.com/english/articles/25-cluster-analysis-in-r-practical-guide/ [↑](#footnote-ref-13)