# **فصل ۱۲. انتخاب مدل**

# **12.0 مقدمه**

در یادگیری ماشین، از الگوریتم‌های آموزشی برای یادگیری پارامترهای یک مدل با کمینه کردن یک تابع خطا استفاده می‌کنیم. با این حال، علاوه بر این بسیاری از الگوریتم‌های یادگیری (مانند دسته‌بندی ماشین بردار پشتیبان و جنگل‌های تصادفی) همچنین دارای ابرپارامترهایی هستند که باید خارج از فرآیند یادگیری تعریف شوند. به عنوان مثال، جنگل‌های تصادفی مجموعه‌ای از درخت‌های تصمیم هستند (بنابراین کلمه «جنگل» استفاده شده است)؛ اما تعداد درخت‌های تصمیم در جنگل توسط الگوریتم یادگیری یادگرفته نمی‌شود و باید قبل از آموزش مدل تعیین شود. این معمولا به عنوان تنظیم ابرپارامتر، بهینه‌سازی ابرپارامتر یا انتخاب مدل شناخته می‌شود. علاوه بر این، اغلب ممکن است بخواهیم چندین الگوریتم یادگیری را امتحان کنیم (به عنوان مثال، امتحان کردن هر دو الگوریتم دسته‌بندی ماشین بردار پشتیبان و جنگل‌های تصادفی) تا ببینیم کدام روش یادگیری مدل بهتری تولید می‌کند.

در حالی که تنوع گسترده‌ای در استفاده از اصطلاحات در این حوزه وجود دارد، در این کتاب به هر دو موضوع انتخاب بهترین الگوریتم یادگیری و بهترین ابرپارامترهای آن به عنوان «انتخاب مدل» اشاره می کنیم. دلیل این امر آسان است: تصور کنید که ما داده‌هایی داریم و می‌خواهیم یک دسته‌بند ماشین بردار پشتیبان با ۱۰ مقدار ابرپارامتر کاندید و یک دسته‌بند جنگل تصادفی با ۱۰ مقدار ابرپارامتر کاندید را آموزش دهیم. نتیجه این می‌شود که ما سعی می‌کنیم بهترین مدل را از میان ۲۰ مدل انتخاب کنیم.

در این فصل، ما تکنیک‌هایی را پوشش می‌دهیم که به طور کامل بهترین مدل را از مجموعه کاندیدها انتخاب کنیم.

در سرتاسر این فصل به ابرپارامترهای خاص مانند C (معکوس قدرت منظم‌سازی[[1]](#footnote-1)) اشاره خواهیم کرد. اگر نمی‌دانید ابرپارامترها چیست، نگران نباشید. در فصول بعدی به آنها خواهیم پرداخت. درعوض، فقط با ابرپارامترها مانند تنظیمات الگوریتم یادگیری رفتار کنید که باید قبل از شروع آموزش انتخاب کنیم.

### **12.1 انتخاب بهترین مدل‌ها با استفاده از جستجوی جامع**

### **مسئله**

شما می‌خواهید بهترین مدل را با جستجو در طیف وسیعی از ابرپارامترها انتخاب کنید.

### **راه‌حل**

از scikit-learn's GridSearchCV استفاده کنید:

### **بررسی**

GridSearchCV یک رویکرد جستجوی فراگیر[[2]](#footnote-2) به انتخاب مدل با استفاده از cross-validation است. به طور خاص، کاربر مجموعه‌هایی از مقادیر ممکن برای یک یا چند ابرپارامتر تعریف می‌کند، و سپس GridSearchCV یک مدل با استفاده از هر مقدار و یا ترکیب مقادیر آموزش می‌دهد. مدل با بهترین امتیاز عملکرد به عنوان بهترین مدل انتخاب می‌شود. به عنوان مثال، در راه‌حل ما از رگرسیون لجستیک به عنوان الگوریتم یادگیری استفاده کرده‌ایم که دارای دو ابرپارامتر C و regularization penalty است. نگران نشوید اگر نمی‌دانید C و regularization به چه معناست؛ ما در چند فصل آتی به آن‌ها پرداخته‌ایم. فقط بدانید که C و regularization penalty می‌توانند مجموعه‌ای از مقادیر را اختیار کنند که قبل از آموزش مشخص می‌شوند. برای C، ما ۱۰ مقدار ممکن تعریف می‌کنیم:



و به طور مشابه، ما دو مقدار ممکن برای regularization penalty تعریف می‌کنیم:['l1', 'l2'] . برای هر ترکیبی از مقادیر C و regularization penalty ، مدل را آموزش می‌دهیم و با استفاده از k-fold cross-validation، آن را ارزیابی می‌کنیم. در راه‌حل ما، ما ۱۰ مقدار ممکن برای C، ۲ مقدار ممکن برای regularization penalty و ۵ برش (5-fold)داشتیم. آن‌ها 100= 5 × 2 × 10 مدل کاندید ایجاد کردند که از میان آن‌ها بهترین انتخاب شد.

هنگامی که GridSearchCV کامل شد، می توانیم ابرپارامتر بهترین مدل را ببینیم: 

به طور پیش‌فرض، پس از شناسایی بهترین ابرپارامترها، GridSearchCV یک مدل را با استفاده از بهترین ابرپارامترها روی کل مجموعه داده مجدداً آموزش می‌دهد (به جای رها کردن یک برش برای

cross-validation). ما می‌توانیم از این مدل برای پیش‌بینی مقادیر مانند هر مدل دیگری در scikit-learn استفاده کنیم:



یکی از پارامترهای مهم GridSearchCV به نام verbose است. اگرچه استفاده از این پارامتر ضروری نیست، ولی در صورتی که فرآیند جستجو طولانی شود، دیدن نشانه‌ای از پیشرفت جستجو و اجرای کد می‌تواند اطمینان‌بخش باشد. پارامتر verbose میزان پیام‌های خروجی‌ در طول جستجو را تعیین می‌کند، به‌طوری که اگر مقدار آن 0 تعیین شود، به این معنی است که هیچ خروجی یا پیامی در صفحه نمایش داده نشود و اگر مقدار آن 1 تا 3 تعیین شود خروجی با جزییات بیشتر نشان داده می‌شود (کمترین میزان جزییات در پیام خروجی مربوط به مقدار 1 و بیشترین مقدار جزییات در پیام خروجی مربوط به مقدار 3 است).

### **همچنین ببینید**

* [از scikit-learn documentation: GridSearchCVاستفاده کنید](http://bit.ly/2Fuctc4)

## **12.2 انتخاب بهترین مدل‌ها با استفاده از جستجوی تصادفی**

### **مسئله**

شما می‌خواهید یک روش محاسباتی ارزان‌تر از جستجوی جامع برای انتخاب بهترین مدل داشته باشید.

### **راه‌حل**

از scikit-learn's RandomizedSearchCV استفاده کنید:

### **بحث**

در [دستوالعمل ۱۲.۱](#_12.1_انتخاب_بهترین)، ما از GridSearchCV بر روی مجموعه‌ای از مقادیر ابرپارامترهای تعریف ‌شده توسط کاربر استفاده کردیم تا با توجه به تابع امتیاز، بهترین مدل را جستجو کنیم. روشی که کارآمدتر از جستجوی کامل GridSearchCV است، جستجو در تعداد مشخصی از ترکیب‌های تصادفی از مقادیر ابرپارامترها از توزیع‌های ارائه‌شده توسط کاربر (مانند توزیع نرمال یا یکنواخت) است. scikit-learn این تکنیک جستجوی تصادفی را با استفاده از RandomizedSearchCV پیاده‌سازی کرده است. با استفاده از RandomizedSearchCV، اگر ما یک توزیع مشخص کنیم، scikit-learn به طور تصادفی و بدون جایگزینی مقادیر ابرپارامترها را از آن توزیع نمونه‌برداری می‌کند. به عنوان مثال از مفهوم کلی، در اینجا ما به طور تصادفی ۱۰ مقدار را از یک توزیع یکنواخت در بازه ۰ تا ۴ نمونه‌برداری می‌کنیم:

از سوی دیگر، اگر ما یک لیستی از مقادیر مشخص کنیم، مانند دو مقدار ابرپارامتر  
 regularization penalty ، ['l1', 'l2']، RandomizedSearchCV از این لیست به صورت تصادفی و با جایگزینی نمونه‌برداری می‌کند.

مشابه GridSearchCV، می‌توانیم مقادیر ابرپارامترهای مدل بهتر را مشاهده کنیم:

و به طرز مشابهی با GridSearchCV، پس از اتمام جستجو، RandomizedSearchCV یک مدل جدید با استفاده از بهترین ابرپارامترها را بر روی کل مجموعه داده آموزش می‌دهد. می‌توانیم از این مدل مانند سایر مدل‌ها در scikit-learn استفاده کنیم؛ به عنوان مثال، برای انجام پیش‌بینی:

تعداد ترکیب‌های نمونه‌برداری شده از ابرپارامترها (به عبارت دیگر، تعداد مدل‌های کاندید آموزش دیده شده) با تنظیم n\_iter (تعداد تکرارها) مشخص می‌شود.

### **همچنین ببینید**

* [اسناد scikit-learn: RandomizedSearchCV](http://bit.ly/2B7p1zT)
* [جستجوی تصادفی برای بهینه سازی](http://bit.ly/2FrUinf) ابرپارامتر

## **12.3 انتخاب بهترین مدل‌ها از بین چند الگوریتم یادگیری**

### **مسئله**

شما می‌خواهید با جستجو در بین تعدادی از الگوریتم‌های یادگیری و ابرپارامترهای مرتبط با آن‌ها، بهترین مدل را انتخاب کنید.

### **راه‌حل**

ایجاد یک dictionary از الگوریتم‌های یادگیری کاندید و ابرپارامترهای آن‌ها:

### **بحث**

در دو دستورالعمل قبلی ما بهترین مدل را با جستجوی مقادیر احتمالی ابرپارامتر یک الگوریتم یادگیری پیدا کردیم. با این حال، اگر مطمئن نیستیم که از کدام الگوریتم یادگیری استفاده کنیم چه؟ نسخه‌های اخیر scikit-learn به ما اجازه می‌دهد تا الگوریتم‌های یادگیری را به عنوان بخشی از فضای جستجو در نظر بگیریم. در راه‌حل ما یک فضای جستجو تعریف می‌کنیم که شامل دو الگوریتم یادگیری رگرسیون لجستیک و دسته‌بندی جنگل تصادفی می‌شود. هر الگوریتم یادگیری دارای ابرپارامترهای خود است و ما مقادیر کاندید خود را با استفاده از فرمت classifier\_\_[hyperparameter name] تعریف می کنیم. به عنوان مثال، برای رگرسیون لجستیک، برای تعریف مجموعه‌ای از مقادیر ممکن برای فضای ابرپارامتر regularization ، C و انواع بالقوه penalty های تنظیم‌کننده‌ها، penalty، ما یک dictionary ایجاد می‌کنیم:

ما همچنین می‌توانیم یک dictionary مشابه برای ابرپارامترهای جنگل تصادفی نیز ایجاد کنیم:

پس از اتمام جستجو، می‌توانیم از best\_estimator\_ استفاده کنیم.

مشاهده الگوریتم و ابرپارامترهای یادگیری بهترین مدل:

درست مانند دو دستورالعمل قبلی، پس از انجام جستجوی انتخاب مدل و آموزش مدل، می‌توانیم از این بهترین مدل به همان شیوه‌ای که از سایر مدل‌های scikit-learn استفاده می‌کنیم، بهره‌برداری کنیم.

## **12.4 انتخاب بهترین مدل ها در هنگام پیش‌پردازش**

### **مسئله**

شما می‌خواهید در فرآیند انتخاب مدل یک مرحله پیش‌پردازش را نیز در نظر بگیرید.

### **راه‌حل**

ایجاد یک pipeline که شامل مرحله پیش‌پردازش و هر یک از پارامترهای آن است:

### **بحث**

خیلی اوقات ما برای استفاده از داده‌های خود برای آموزش مدل نیاز به پیش‌پردازش داریم. ما باید مراقب باشیم که به درستی پیش‌پردازش را در هنگام انجام انتخاب مدل انجام دهیم. ابتدا، GridSearchCV از  
 cross-validation برای تعیین اینکه کدام مدل دارای بالاترین عملکرد است، استفاده می‌کند. با این حال، در cross-validation، ما به طور واقعی داریم وانمود می‌کنیم که برش نگه داشته شده به عنوان مجموعه آزمایشی، به عبارت دیگر مجموعه آزمون، دیده نمی‌شود و در نتیجه بخشی از فرآیند تنظیم هیچ گام پیش‌پردازشی (مانند مقیاس‌بندی یا استانداردسازی) نمی‌شود. به همین دلیل، ما نمی‌توانیم داده‌ها را پیش‌پردازش کنیم و سپس GridSearchCV را اجرا کنیم. بلکه گام‌های پیش‌پردازش باید به عنوان بخشی از مجموعه اقدامات انجام‌شده توسط GridSearchCV باشد. اگرچه این ممکن است پیچیده به نظر برسد، واقعیت این است که scikit-learn این را ساده می‌کند. FeatureUnion به ما این امکان را می‌دهد که چندین اقدام پیش‌پردازش را به درستی ترکیب کنیم. در راه‌حل ما از FeatureUnion برای ترکیب دو مرحله پیش‌پردازش استفاده می‌کنیم: استانداردسازی مقادیر ویژگی‌ها (StandardScaler) و تجزیه و تحلیل مؤلفه‌های اصلی.(PCA) این شیء پیش‌پردازش (preprocess) نامیده می‌شود و دو مرحله پیش‌پردازش ما را شامل می‌شود. سپس پیش‌پردازش را به عنوان یک pipelineبه همراه الگوریتم یادگیری‌مان ترکیب می‌کنیم. نتیجه نهایی این است که این امکان را به ما می‌دهد که مراحل مناسب (و پیچیده) تنظیم، تبدیل و آموزش مدل‌ها با ترکیب‌های مختلف ابرپارامترها را به scikit-learn بسپاریم.

دوماً، برخی از روش‌های پیش‌پردازش دارای پارامترهای خود هستند که معمولاً توسط کاربر باید تعیین شوند. به عنوان مثال، کاهش ابعاد با استفاده از تجزیه و تحلیل مؤلفه‌های اصلی PCA نیازمند تعریف تعداد اجزای اصلی است که برای تولید مجموعه ویژگی تبدیل شده است. بهتر است تعداد اجزایی را انتخاب کنیم که مدلی با بیشترین عملکرد برای معیار ارزیابی آزمایشی انتخاب کند. خوشبختانه، scikit-learn این کار را آسان می‌کند. وقتی مقادیر اجزای کاندیدا را در فضای جستجو اضافه می‌کنیم، آن‌ها مانند سایر ابرپارامترها به عنوان مقدار جستجو مورد انتظار مورد استفاده قرار می‌گیرند. در راه‌حل ما، ما  
 features\_\_pca\_\_n\_components': [1, 2, 3] را در فضای جستجو تعریف کردیم تا نشان دهیم که می‌خواهیم بیابیم که آیا یک، دو یا سه مؤلفه اصلی بهترین مدل را تولید می‌کند.

پس از اتمام انتخاب مدل، می‌توانیم مقادیر پیش‌پردازش را که مدل بهتر را تولید کرده است، مشاهده کنیم. به عنوان مثال، می‌توانیم بهترین تعداد اجزای اصلی را مشاهده کنیم:





## **12.5 افزایش سرعت انتخاب مدل با استفاده از موازی‌سازی**

### **مسئله**

شما باید سرعت انتخاب مدل را افزایش دهید.

### **راه‌حل**

با تنظیم n\_jobs=-1، از تمام هسته‌های موجود در دستگاه خود استفاده کنید.





### **بحث**

در دستور‌های این فصل، تعداد مدل‌های کاندیدا را کم نگه داشته‌ایم تا کد به سرعت کامل اجرا شود. با این حال، در دنیای واقعی ما اغلب هزاران یا ده‌ها هزار مدل برای اموزش خواهیم داشت. نتیجه نهایی این است که ساعت‌ها طول می‌کشد تا بهترین مدل را پیدا کنید. برای سرعت بخشیدن به این فرایند، scikit-learn به ما اجازه می‌دهد تا چندین مدل را به طور همزمان آموزش دهیم. بدون وارد شدن به جزئیات فنی بیش از حد، scikit-learn می‌تواند به طور همزمان مدل‌ها را تا تعداد هسته‌های موجود در دستگاه آموزش دهد. اکثر لپ‌تاپ‌های مدرن دارای چهار هسته هستند، بنابراین (با فرض اینکه شما در حال حاضر در حال استفاده از یک لپ‌تاپ هستید) ما به طور بالقوه می‌توانیم چهار مدل را همزمان آموزش دهیم. این به طور چشمگیری سرعت فرایند انتخاب مدل ما را افزایش می‌دهد. پارامتر n\_jobs تعداد مدل‌هایی را که به طور موازی آموزش می‌دهند تعریف می‌کند.

در راه‌حل ما، ما پارامتر n\_jobs را برابر با 1- تنظیم می‌کنیم، که به scikit-learn می‌گوید از تمام هسته‌ها استفاده کند. با این حال، به صورت پیش‌فرض n\_jobs برابر با 1 تنظیم شده است که به معنای استفاده از یک هسته است. برای نمایش این موضوع، اگر همان GridSearch را که در راه‌حل داشتیم، با تنظیم n\_jobs برابر با 1 اجرا کنیم، می توانیم ببینیم که پیدا کردن بهترین مدل به طور قابل توجهی طول می کشد (توجه داشته باشید که زمان دقیق به کامپیوتر شما بستگی دارد):





## **12.6 سرعت بخشیدن به انتخاب مدل با استفاده از الگوریتم‌های خاص**

### **مسئله**

شما باید سرعت انتخاب مدل را افزایش دهید.

### **راه‌حل**

اگر از تعداد محدودی از الگوریتم‌های یادگیری استفاده می‌کنید، از تنظیم ابرپارامترهای مخصوص مدل و cross-validation، scikit-learn استفاده کنید. به عنوان مثال، :LogisticRegressionCV





### **بحث**

گاهی اوقات ویژگی‌های یک الگوریتم یادگیری به ما اجازه می‌دهد که به دنبال بهترین ابرپارامترها با سرعت قابل توجهی نسبت به روش‌های جستجوی کلی یا تصادفی مدل باشیم. در scikit-learn، بسیاری از الگوریتم‌های یادگیری (مانند ridge ، lasso و رگرسیون شبکه‌ای elastic) دارای یک روش اختصاصی برای cross-validation با استفاده از این ویژگی‌ها هستند. به عنوان مثال، LogisticRegression یک دسته‌بند رگرسیون لجستیک استاندارد است، در حالی که LogisticRegressionCV یک دسته‌بند رگرسیون لجستیک کارا را پیاده‌سازی می‌کند که توانایی شناسایی مقدار بهینه ابرپارامتر C را دارد.

روش LogisticRegressionCV scikit-learn شامل پارامتر Cs است. اگر یک لیست ارائه شود، Cs مقادیر ابرپارامتر کاندید برای انتخاب است. اگر یک عدد صحیح ارائه شود، پارامتر Cs یک لیستی از مقادیر کاندید با آن تعداد می‌سازد. این مقادیر کاندید به صورت لگاریتمی از دامنه‌ای بین ۰٫۰۰۰۱ تا ۱۰۰۰۰ (دامنه‌ای از مقادیر معقول برای C تولید می‌شوند.).

با این حال، یک نقطه ضعف عمده در LogisticRegressionCV وجود دارد و آن این است که تنها می‌تواند یک دامنه از مقادیر برای C جستجو کند. در [دستور العمل 12.1](#_12.1_انتخاب_بهترین) فضای ابرپارامتر ممکن ما هم C و هم ابرپارامتر دیگر را شامل می‌شود ( regularization penalty norm). این محدودیت در بسیاری از رویکردهای اختصاصی مدل scikit-learn وجود دارد.

### **همچنین ببینید**

* [مستندات scikit-learn: LogisticRegressionCV](http://bit.ly/2GPJvjY)
* [مستندات scikit-learn: cross-validation خاص مدل](http://bit.ly/2F0TQsL)

## **12.7 ارزیابی عملکرد پس از انتخاب مدل**

### **مسئله**

شما می‌خواهید عملکرد یک مدل که از طریق انتخاب مدل یافته شده است را ارزیابی کنید.

### **راه‌حل**

استفاده از cross-validation تودرتو برای جلوگیری از ارزیابی سوگیرانه:





### **بحث**

Cross-validation تودرتو در طول انتخاب مدل مفهومی دشوار برای بسیاری از مردم برای اولین بار است. به یاد داشته باشید که در cross-validation k-fold، ما مدل خود را بر روی k–1 برش از داده‌ها آموزش می‌دهیم، از این مدل برای پیش‌بینی در برش k ام داده‌ها که باقی‌ مانده است استفاده می‌کنیم و سپس مدل خود را بر اساس اینکه چقدر پیش‌بینی‌های مدل ما با مقادیر واقعی مطابقت دارد، ارزیابی می‌کنیم. سپس این فرآیند را k بار تکرار می‌کنیم.

در جستجوهای انتخاب مدل شرح داده شده در این فصل (یعنی GridSearchCV و RandomizedSearchCV)، ما از cross-validation برای ارزیابی اینکه کدام مقادیر ابرپارامتر بهترین مدل‌ها را تولید می‌کنند، استفاده کردیم. با این حال، یک مسئله ظریف و به طور کلی کم‌ارزش پیش می‌آید: از آنجایی که ما از داده‌ها برای انتخاب بهترین مقادیر ابرپارامتر استفاده کردیم، نمی‌توانیم از همان داده‌ها برای ارزیابی عملکرد مدل استفاده کنیم. راه‌حل چیست؟ cross-validation استفاده شده برای جستجوی مدل را در یک اعتبارسنجی تو در تو قرار دهیم! در اعتبارسنجی تو در تو، اعتبارسنجی «داخلی» بهترین مدل را انتخاب می‌کند، در حالی که اعتبارسنجی «بیرونی» ارزیابی بی‌طرفانه‌ای از عملکرد مدل به ما ارائه می‌دهد. در راه‌حل ما، اعتبارسنجی داخلی شیء GridSearchCV ماست که سپس در یک اعتبارسنجی بیرونی با استفاده از cross\_val\_score قرار می‌دهیم.

اگر سردرگم شده‌اید، یک آزمایش ساده انجام دهید. ابتدا verbose=1 را تنظیم کنید تا بتوانیم ببینیم چه اتفاقی می‌افتد:



سپس، gridsearch.fit(features, target) را اجرا کنید، که cross-validation داخلی ما برای یافتن بهترین مدل استفاده می‌شود:





از خروجی می‌توانید ببینید که cross-validation داخلی 20 مدل کاندید را پنج بار آموزش داده است، که مجموعاً 100 مدل می‌شود. سپس، clf را در داخل یک cross-validation جدید، که به صورت پیش‌فرض سه برش (3-fold) دارد، قرار دهید:





از خروجی مشاهده می‌شود که cross-validation تو در تو 20 مدل را پنج بار آموزش داد تا بهترین مدل را پیدا کند، و این مدل با استفاده از cross-validation بیرونی با three-fold ارزیابی شد، که در مجموع 300 مدل آموزش داده شده است.

1. the inverse of regularization strength [↑](#footnote-ref-1)
2. brute-force [↑](#footnote-ref-2)