جستجوی تصادفی

رویکرد جستجوی شبکه ای زمانی که ترکیب های نسبتا کمی را کاوش می کنید، مانند مثال قبلی، خوب است، اما کلاس (رندومایزد سرچ سی وی) اغلب ترجیح داده می شود، به ویژه زمانی که فضای جستجوی فراپارامتر بزرگ است. این کلاس را میتوان تقریبا به روشی مشابه کلاس (گرید سرچ سی وی) استفاده کرد، اما به جای آزمایش همه ترکیب های ممکن، تعداد ثابتی از ترکیب ها را ارزیابی می کند، و در هر تکرار یک مقدار تصادفی برای هر ابرپارامتر انتخاب می کند. این ممکن است تعجب آور به نظر برسد، امااین روش چندین مزیت دارد:

- اگربرخی از ابرپارامترهای شما پیوسته باشند (یا گسسته اما با مقادیر بسیار زیاد) ،و به جستجوی تصادفی اجازه دهید مثلا 1000 تکرار انجام شود، آنگاه 1000 مقدارمختلف برای هر یک از این ابرپارامترها بررسی می شود، در حالی که جستجوی شبکه ای فقط مقدار کمی که برای هر یک لیست کرده اید.

- فرض کنید یک ابرپارامتر در واقع تفاوت زیادی ایجاد نمی کند، اما شما هنوز آن را نمیدانید. اگر 10 مقدار ممکن داشته باشد و آن را به جستجوی شبکه خود اضافه کنید،آموزش 10 برابر بیشتر طول می کشد. اما اگر آن را به یک جستجوی تصادفی اضافه کنید، هیچ تفاوتی نخواهد داشت.

- اگر 6 فراپارامتر برای کاوش وجود داشته باشد که هر کدام دارای 10 مقدار ممکن است، جستجوی شبکه ای هیچ انتخاب دیگری جز آموزش یک میلیون بار مدل ارائه نمی دهد، در حالیکه جستجوی تصادفی همیشه می تواند برای هر تعداد تکراری که شما انتخاب می کنید اجراشود.

برای هر ابرپارامتر، باید لیستی از مقادیر ممکن یا توزیع احتمال ارائه کنید:

```
from sklearn.model_selection import RandomizedSearchCV

from scipy.stats import randint

param_distribs = {'preprocessing__geo__n_clusters': randint(low=3, high=50),
'random_forest__max_features': randint(low=2, high=20)}

rnd_search = RandomizedSearchCV(

full_pipeline, param_distributions=param_distribs, n_iter=10, cv=3,

scoring='neg_root_mean_squared_error', random_state=42) rnd_search.fit(housing,
housing_labels)
```

(سایکیت لرن:یک کتابخانه رایگان در زبان پایتون)همچنین دارای کلاس های جستجوی ابرپارامتری (هالوینگ رندوم سرچ سی وی) و (هالوینگ گرید سرچ سی وی) است. هدف آنها استفاده کارآمدتر از منابع محاسباتی است، چه برای آموزش سریعتر و چه برای کشف فضای ابرپارامتری بزرگتر. نحوه کار آنها به این صورت است: در دور اول، بسیاری از

ترکیبات فراپارامتر (به نام "کاندیدا") با استفاده از رویکرد شبکه یا رویکرد تصادفی تولید می شوند. سپس از این نامزدها برای آموزش مدل هایی استفاده می شود که طبق معمول با استفاده از اعتبارسنجی متقابل ارزیابی می شوند. با این حال، آموزش از منابع محدودی استفاده می کند که این دور اول را به میزان قابل توجهی سرعت می بخشد. به طورپیش فرض، "منابع محدود" به این معنی است که مدل ها در بخش کوچکی از مجموعه آموزشی آموزش داده می شوند. با این حال، محدودیت های دیگری نیز ممکن است، مانند کاهش تعداد تکرارهای آموزشی اگر مدل دارای یک فراپارامتر برای تنظیم آن باشد. هنگامی که هر نامزد ارزیابی شد، فقط بهترین ها به دور دوم می روند، جایی که به آنها اجازه داده می شود منابع بیشتری برای رقابت داشته باشند. پس از چندین دور، نامزدهای نهایی با استفاده از منابع کامل ارزیابی می شوند. این ممکن است در زمان تنظیم هایپرپارامترها صرفه جویی کند.

روش های گروهی

راه دیگر برای تنظیم دقیق سیستم این است که سعی کنید مدل هایی را که بهترین عملکرد را دارند ترکیب کنید. گروه (یا "ترکیب") اغلب بهتر از بهترین مدل فردی عمل میکند - درست مانند جنگل های تصادفی که بهتر از درخت های تصمیم گیری فردی که به آنها تکیه می کنند عمل می کنند - به خصوص اگر مدل های فردی انواع بسیار متفاوتی از خطاهارا داشته باشند. به عنوان مثال می توانید آموزش دهید و تعدیل و تنظیم کنید الگوریتم (کی-نیرست نیبرز مادل) را. سپس یک مدل مجموعه ای ایجاد کنید که فقط میانگین پیش بینی جنگل تصادفی و پیش بینی آن مدل را پیش بینی می کند. در ادامه به این موضوع خواهیم پرداخت در فصل 7.

تجزیه و تحلیل بهترین مدل ها و خطاهای آنها

شما اغلب با بررسی بهترین مدل ها بینش خوبی در مورد مشکل به دست خواهید آورد.به عنوان مثال، (رگرسور جنگل تصادفی) می تواند اهمیت نسبی هر ویژگی را برای پیش بینی های دقیق نشان دهد:

```
>>> final_model = rnd_search.best_estimator_ # includes preprocessing

>>> feature_importances = final_model["random_forest"].feature_importances_

>>> feature_importances.round(2)

array([0.07, 0.05, 0.05, 0.01, 0.01, 0.01, 0.01, 0.19, [...], 0.01])
```

بیایید این امتیازهای اهمیت را به ترتیب نزولی مرتب کنیم و آنها را در کنار نام ویژگی های مربوطه نمایش دهیم:

```
>>> sorted(zip(feature_importances,

... final_model["preprocessing"].get_feature_names_out()),

... reverse=True)

... [(0.18694559869103852, 'log__median_income'),

(0.0748194905715524, 'cat__ocean_proximity_INLAND'),

(0.06926417748515576, 'bedrooms__ratio'),
```

(0.05446998753775219, 'rooms_per_house__ratio'),

(0.05262301809680712, 'people_per_house__ratio'),

(0.03819415873915732, 'geo__Cluster 0 similarity'),

[...]

(0.00015061247730531558, 'cat__ocean_proximity_NEAR BAY'),

(7.301686597099842e-05, 'cat__ocean_proximity_ISLAND')]

بااین اطلاعات، ممکن است بخواهید برخی از ویژگی های کمتر مفید را حذف کنید (به عنوان مثال، ظاهراً فقط یک دسته "مجارت اقیانوس" واقعا مفید است، بنابراین می توانیدبقیه را حذف کنید).

نکته

مبدل SKLEARN.FEATURE_SELECTION.SELECTFROMMODEL می تواند به طور خودکار کمترین ویژگی های مفید را برای شما حذف کند: وقتی آن را جا دادید، یک مدل را آموزش می دهد (معمولاً یک جنگل تصادفی)، با توجه به صفت های مهم ظاهری و مفیدترین ویژگی ها را انتخاب می کند. سپس وقتی TRANSFORM() را فراخوانی می کنید، سایر ویژگی ها را حذف می کند.

همچنین باید به خطاهای خاصی که سیستم شما مرتکب می شود نگاه کنید، سپس سعی کنید بفهمید که چرا آنها را ایجاد می کند و چه چیزی می تواند مشکل را برطرف کند: اضافه کردن ویژگی های اضافی یاخلاص شدن از شر موارد غیر اطلاعاتی، تمیز کردن نقاط پرت و غیره.

اکنون نیز زمان خوبی است تا مطمئن شوید که مدل شما نه تنها به طور متوسط، بلکه درهمه دسته های مناطق، اعم از روستایی یا شهری، ثروتمند یا فقیر، شمالی یا جنوبی، اقلیت یا غیره، خوب کار می کند. ایجاد زیر مجموعه ها مجموعه اعتبارسنجی شما برای هردسته کمی کار می برد، اما مهم است: اگر مدل شما در کل دسته بندی مناطق ضعیف عمل می کند، احتمالاً تا زمانی که مشکل حل نشود، نباید آن را مستقر کنید، یا دست کم نباید برای پیش بینی برای آن دسته استفاده شود، زیرا ممکن است ضرر بیشتری نسبت به فایده داشته باشد.

سیستم خود را در مجموعه تست ارزیابی کنید

پس از مدتی از بهینه سازی مدل های خود، در نهایت سیستمی خواهید داشت که به اندازه کافی خوب عمل می کند. شما آماده ارزیابی مدل نهایی در مجموعه تست هستید. هیچ چیز خاصی درمورد این روند وجود ندارد. فقط پیش بینی کننده ها و برچسب ها را از مجموعه آزمایشی خوددریافت کنید و مدل نهایی خود را برای تبدیل داده ها و پیش بینی ها اجرا کنید، سپس این پیش بینی ها را ارزیابی کنید:

```
X_test = strat_test_set.drop("median_house_value", axis=1)

y_test = strat_test_set["median_house_value"].copy()

final_predictions = final_model.predict(X_test)

final_rmse = mean_squared_error(y_test, final_predictions, squared=False)

print(final_rmse) # prints 41424.40026462184
```

دربرخی موارد، چنین تخمین نقطه ای از خطای تعمیم برای متقاعد کردن شما برای راه اندازی کافی نخواهد بود: اگر فقط 0.1 درصد بهتر از مدل فعلی در تولید باشد، چه؟ ممکناست بخواهید ایده ای درباره دقیق بودن این تخمین داشته باشید. برای این، می توانید٪95 *فاصله اطمینان* برای خطای تعمیم را محاسبه کنید با استفاده از Spicy.stats.t.interval() . شما یک بازه نسبتاً بزرگ از 39275 تا 43467 دریافت می کنید و تخمین امتیاز قبلی شما از 41424 تقریباً در وسط آن است.

```
>>> from scipy import stats

>>> confidence = 0.95

>>> squared_errors = (final_predictions - y_test) ** 2

>>> np.sqrt(stats.t.interval(confidence, len(squared_errors) - 1,

... loc=squared_errors.mean(),

... scale=stats.sem(squared_errors)))

... array([39275.40861216, 43467.27680583])
```

اگرتنظیم ابرپارامتر زیادی انجام داده اید، عملکرد معمولا کمی بدتر از آنچه با استفاده از اعتبارسنجی متقاطع اندازه گیری کرده اید خواهد بود. این به این دلیل است که سیستم شما در نهایت به خوبی تنظیم می شود تا در داده های اعتبارسنجی عملکرد خوبی داشته باشد و احتمالا در مجموعه داده های ناشناخته عملکرد خوبی نخواهد داشت. این مورد در این مثال نیست زیرا آزمون RMSE کمتر از اعتبارسنجی RMSE است، اما وقتی این اتفاق می افتد، باید در مقابل وسوسه تغییر پارامترها مقاومت کنید تا اعداد درمجموعه آزمایشی خوب به نظر برسند. بعید است که پیشرفت ها به داده های جدید تعمیم یابد.

اکنون مرحله پیش راه اندازی پروژه فرا می رسد: باید راه حل خود را ارائه دهید (با برجسته کردن چیزهایی که آموخته اید، چه چیزهایی مؤثر بوده و چه چیزی انجام نشده است،چه فرضیاتی ساخته شده است، و محدودیت های سیستم شما چیست)، همه چیزرا مستند کنید، و ارائه های خوبی با تجسم های واضح ایجاد کنید. و جملاتی که به راحتی قابل به خاطر سپردن هستند (به عنوان مثال، "درآمد متوسط، پیش بینی کننده شماره یک قیمت مسکن است"). در این نمونه مسکن کالیفرنیا، عملکرد نهایی سیستم خیلی بهتر از تخمین قیمت کارشناسان نیست، که اغلب تا 30 درصد کاهش داشت، اما همچنان ممکن است راه اندازی آن ایده خوبی باشد، به خصوص اگر این کار مقداری زمان راآزاد کند. برای متخصصان تا بتوانند روی کارهای جالب و سازنده تری کار کنند.

سیستم خود را راه اندازی، نظارت و نگهداری کنید

عالی است، مجوز راه اندازی را دریافت کردید! اکنون باید راه حل خود را برای تولید آماده کنید.(به عنوان مثال، کد را صیقل دهید، مستندات و آزمایش ها را بنویسید، و غیره). سپس می توانید مدل خود را در محیط تولید خود مستقر کنید. ابتدایی ترین راه برای انجام این کار این است که بهترین مدلی را که آموزش داده اید ذخیره کنید، فایل را به محیط تولید خود منتقل کنید و آن را بارگذاری کنید. برای ذخیره مدل، می توانید از کتابخانه joblib به صورت زیر استفاده کنید:

```
import joblib

joblib.dump(final_model, "my_california_housing_model.pkl")
```

نکته

اغلب ایده خوبی است که هر مدلی را که با آن آزمایش می کنید ذخیره کنید تا بتوانید به راحتی به هر مدلی که می خواهید برگردید. همچنین می توانید نمرات اعتبارسنجی متقابل و شاید پیش بینی های واقعی را در مجموعه اعتبارسنجی ذخیره کنید. این به شما امکان می دهد به راحتی نمرات را در انواع مدلها مقایسه کنید و انواع خطاهای آنها را مقایسه کنید.

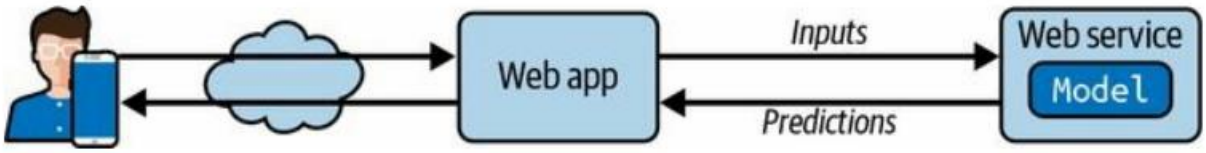
هنگامی که مدل شما به تولید منتقل شد، می توانید آن را بارگیری کرده و از آن استفاده کنید. برای این کار، ابتدا باید کلاس ها و توابع سفارشی ای که مدل به آن ها متکی است را وارد کنید (که به معنای انتقال کد به تولید است)، سپس مدل را با استفاده از joblib بارگیری کنید و از آن برای پیش بینی استفاده کنید:

```
import joblib

[...] # import KMeans, BaseEstimator, TransformerMixin, rbf_kernel, etc.

def column_ratio(X): [...]

def ratio_name(function_transformer, feature_names_in): [...]

class ClusterSimilarity(BaseEstimator, TransformerMixin): [...]

final_model_reloaded = joblib.load("my_california_housing_model.pkl")

new_data = [...] # some new districts to make predictions for

predictions = final_model_reloaded.predict(new_data)
```

به عنوان مثال، شاید این مدل در یک وب سایت استفاده شود: کاربر برخی از داده های مربوط به یک منطقه جدید را تایپ می کند و روی دکمه برآورد قیمت کلیک می کند. این یک کوئری حاوی داده ها را به وب سرور ارسال می کند، که آن را به برنامه وب شما ارسال می کند، و در نهایت کد شما به سادگی متد مدل Predict() را فراخوانی می کند (شما می خواهید مدل را هنگام راه اندازی سرور بارگذاری کنید، نه هر بار. این مدل استفاده می شود).

از طرف دیگر، می توانید مدل را در یک وب سرویس اختصاصی قرار دهید که برنامه وب شما می تواند از طریق REST API

پرس و جو کند.(نگاه کنید به شکل 20-2). این کار ارتقای مدل خود را به نسخه های جدید بدون ایجاد وقفه در برنامه اصلی آسان تر میکند. همچنین مقیاس بندی را ساده تر می کند، زیرا می توانید به تعداد مورد نیاز سرویس های وب را راه اندازی کنید و درخواست های دریافتی از برنامه وب شما را در این سرویس های وب، بارگذاری کنید. علاوه بر این، به برنامه وب شما اجازه می دهد تا از هر زبان برنامه نویسی استفاده کند، نه فقط پایتون.



شکل2-20. مدلی که به عنوان یک وب سرویس مستقر شده و توسط یک برنامه وب استفاده می شود

یکی دیگر از استراتژی های محبوب، استقرار مدل خود در فضای ابری است، به عنوان مثال در Vertax AI Google

(که قبلا به عنوان Google cloud AI platform و Google cloud ML engine شناخته میشد):فقط مدل خود را با استفاده از joblib

ذخیره کنید و آن را در Google cloud storage:GCS آپلود کنید. سپس به Vertax AI بروید و یک نسخه مدل جدید ایجاد کنید و آن را به فایل GCS نشان دهید. همین! این به شما یک وب سرویس ساده میدهد که از تعادل بار و مقیاس بندی برای شما مراقبت می کند. درخواست های Json حاوی داده های ورودی (مثلاً یک منطقه) را می گیرد و پاسخ های Json حاوی پیش بینی ها را برمی گرداند.سپس می توانید از این وب سرویس در وب سایت خود (یا هر محیط تولیدی که استفاده میکنید) استفاده کنید. همان طور که در فصل 19 خواهید دید، استقرار مدل های Tensor Flow در Vertax AI تفاوت های زیادی با مدل های Scikit-learn ندارد.

اما استقرار پایان ماجرا نیست. همچنین باید کد مانیتورینگ بنویسید تا عملکرد زنده سیستم خود را در فواصل زمانی معین بررسی کنید و هنگام افت آن هشدارها را راه اندازی کنید.ممکن است خیلی سریع از بین برود، برای مثال اگر یک جزء در زیرساخت شما خراب شود، اما توجه داشته باشید که می تواند بسیار آهسته خراب شود، که به راحتی می تواند برای .مدت طولانی مورد توجه قرار نگیرد؛ این کاملا رایج است.

به دلیل پوسیدگی مدل: اگر مدل با داده های سال گذشته آموزش داده شده باشد، ممکن است با دادههای امروزی سازگار نباشد.

بنابراین،شما باید عملکرد زنده مدل خود را نظارت کنید. اما چگونه این کار را انجام می دهید؟ خب بستگی داره در برخی موارد، عملکرد مدل را می توان از معیارهای پایین دستی استنباط کرد.به عنوان مثال، اگر مدل شما بخشی از یک سیستم توصیه کننده است و محصولاتی را پیشنهادمی کند که کاربران ممکن است به آن ها علاقه داشته باشند، نظارت بر تعداد محصولات پیشنهادی فروخته شده در هر روز آسان است. اگر این عدد کاهش یابد (در مقایسه بامحصولات غیرتوصیه شده)، پس مظنون اصلی مدل است. این ممکن است به این دلیل باشدکه خط لوله داده شکسته شده است، یا شاید مدل نیاز به آموزش مجدد بر روی داده های تازه داشته باشد (همانطور که به زودی در مورد آن صحبت خواهیم کرد).

بااین حال، ممکن است برای ارزیابی عملکرد مدل نیز به تحلیل انسانی نیاز داشته باشید. به عنوان مثال، فرض کنید یک مدل طبقه بندی تصویر را آموزش داده اید (ما به این موارد در ادامهنگاه خواهیم کرد در فصل 3) برای تشخیص عیوب مختلف محصول در یک خط تولید. چگونه میتوانید در صورت کاهش عملکرد مدل، قبل از ارسال هزاران محصول معیوب به مشتریان، هشداردریافت کنید؟ یک راه حل این است که نمونه ای از تمام تصاویری را که مدل طبقه بندی کرده است (مخصوصاً تصاویری که مدل در مورد آنها چندان مطمئن نبود) برای ارزیابی کنندگانانسانی ارسال شود. بسته به وظیفه، ارزیابی کنندگان ممکن است نیاز داشته باشند که متخصص باشند، یا می توانند غیرمتخصص باشند، مانند کارگرانی که در یک پلتفرم جمع سپاری مثل (Amazom Mecanical Turk) هستند. در برخی از برنامه ها، آن ها حتی می توانندخود کاربران باشند و به عنوان مثال، از طریق نظرسنجی یا کپچای تغییر کاربری داده شده،پاسخ دهند.

درهر صورت، شما باید یک سیستم نظارتی (با یا بدون رتبه دهنده انسانی برای ارزیابی مدل زنده) و همچنین تمام فرآیندهای مربوطه را برای تعریف اقدامات لازم در صورت خرابیو نحوه آماده سازی برای آنها ایجاد کنید. متأسفانه، این می تواند کار زیادی باشد. درواقع، اغلب کار بسیار بیشتر از ساختن و آموزش یک مدل است.

اگرداده ها به تکامل خود ادامه دهند، باید مجموعه داده های خود را به روز کنید و مدلخود را مرتباً مجدداً آموزش دهید. احتمالاً باید کل فرآیند را تا حد امکان خودکار کنید.در اینجا چند چیز وجود دارد که می توانید خودکار کنید:

داده های تازه را به طور منظم جمع آوری کنید و آنها را برچسب گذاری کنید (مثلاً با استفاده از ارزیابی کننده های انسانی).