Fəsil 6. Qərar Ağacları (Decision Trees)

Qərar Ağacları da təsnifat, reqressiya, və hətta çox çıxışlı tapşırıqları icra edə bilən müxtəlif Maşın Öyrənmə alqoritmləridir. Onlar mürəkkəb verilənlər toplularına uyğunlaşa bilmə qabiliyyətinə malik olan güclü alqoritmlərdir. Məsələn, siz 2-ci fəsildə Kaliforniya mənzil verilənlər toplusu olan DecisionTreeRegressor modelinin təlimini onu mümkün qədər uygun yerləşdirərək (əslində həddindən artıq uyğunlaşdıraraq) gördünüz.

Qərar Ağacları həmçinin bu gün mövcud olub ən güclü Maşın Öyrənmə alqoritmlərindən biri olan Təsadüfi Meşələrin (bax Fəsil 7) əsas komponentlərindən biridir.

Biz bu fəsilə təlimlər etmək, vizuallaşdırmaq və proqnozlar hazırlamaq üçün Qərar Ağaclarından necə istifadə edəcəyimizi müzakirə edərək başlayacağıq. Sonra Scikit-Öyrənmə tərəfindən istifadə olunan CART təlim alqoritmini tədqiq edəcəyik və ağacları nəzərdə tutaraq onları reqressiya tapşırıqları üçün necə istifadə edəcəyimiz barədə danışacağıq. Son olaraq, Qərar Ağaclarının bəzi məhdudiyyətlərini müzakirə edəcəyik.

Qərar Ağacı Təlimi və Vizuallaşdırılması

Qərar Ağaclarını anlamaq üçün, gəlin ondan bir ədəd quraq və proqnozları necə etdiyinə nəzər yetirək. Aşağıda verilmiş kod verilənlər toplusunda DecisionTreeClassifier-ı (QərarAğacıTəsnifatı) təlim edir (bax Fəsil 4):

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier

iris = load_iris(as_frame=True)
X_iris = iris.data[["petal length (cm)", "petal width (cm)"]].values

y_iris = iris.target

tree_clf = DecisionTreeClassifier(max_depth=2, random_state=42)

tree_clf.fit(X_iris, y_iris)
```

Təlim edilmiş Qərar Ağacını vizuallaşdırmaq üçün əvvəlcə export_graphviz() metodundan istifadə edərək iris_tree.dot adlı bir qrafik təsvir faylı yarada bilərsiniz:

```
from sklearn.tree import export_graphviz

export_graphviz( tree_clf,

        out_file="iris_tree.dot",
        feature_names=["petal length (cm)", "petal width (cm)"],
class_names=iris.target_names,
        rounded=True,
        filled=True

        )
```

Daha sonra siz graphviz.Source.from_file()-dan faylı Jüpiter notebook-da yükləmək və ya göstərmək üçün istifadə edə bilərsiniz.

```
from graphviz import

Source Source.from_file("iris_tree.dot")
```

Graphviz, açıq mənbəli qraf vizualizasiya proqram təminatıdır. O həm də *.dot* fayllarını PDF və ya PNG kimi müxtəlif formatlara çevirmək üçün bir nöqtə command-line xidmətinə sahibdir.
Sizin ilk qərar ağacınız belə görsənir Figure 6-1.

Şəkil 6-1. İris qərar ağacı

Təxminlər etmək

Gəlin Şəkil 6-1-də təsvir olunan ağacın təxminləri necə etdiyinə baxaq. Təsəvvür edin ki, bir iris çiçəyi tapdınız və onu ləçəklərinə əsasən sinifləndirmək istəyirsiniz. Siz əvvəlcə kök düyünü ilə (dərinlik 0, yuxarıda) başlayırsınız: bu düyün çiçəyin ləçəyinin uzunluğunun 2.45 sm-dən kiçik olup olmadığını soruşur. Əgər elədirsə, o zaman onu kökün sol alt balaca düyününə (dərinlik 1, sola) keçirirsiniz. Bu halda, o yarpaq düyünüdür (yəni, onun heç bir yetirmə düyünü yoxdur), ona görə də hiç bir sual vermir: sadəcə bu düyün üçün təxmin olunan sinifə diqqət yetirərək çiçəyinizin bir *Iris Setosa* (sinif=setosa) olduğunu görə bilərsiniz.

İndi təsəvvür edin ki, başqa bir çiçək tapdınız, ancaq bu dəfə çiçəyin ləçəyinin uzunluğu 2.45 sm-dən böyükdür. Bu halda, kökün sağ balaca düyününə (dərinlik 1, sağa) keçməlisiniz ki, bu yarpaq düşün olmur. Buna görə də başqa bir sual verir: çiçəyin ləçəyi 1.75 sm-dən kiçikdirsə, o zaman çiçəyinizin böyük ehtimal bir Iris-Versicolor (dərinlik 2, sola) olduğunu düşünə bilərsiniz. Əgər deyilsə, deməli onun bir Iris-Virginica (dərinlik 2, sağa) olması ehtimalı yüksəkdir. Əslində bu qədər sadədir.

> NOT
>
> Qərar Ağaclarının bir çox üstünlüklərindən biri də onların çox az məlumat hazırlanmasına ehtiyac duymasıdır. Xüsusilə, onlar heç bir halda xüsusiyyət ölçülməsi və ya mərkəzlənmə tələb etmirlər.

Bir düyümün nümunələr atributu ona tətbiq edilən təlim hadisələrinin sayını hesablayır. Məsələn, 100 təlim hadisəsi arasında ləçəyinin uzunluğu 2.45 sm-dən böyük olan çiçəklər var (dərinlik 1, sağ), onların arasında isə 54 nümunənin ləçəyinin genişliyi 1.75 sm-dən kiçikdir (dərinlik 2, sol). Bir düyümün dəyər atributu sizə bu düyümün hər bir sinifdə neçə təlim hadisəsinə tətbiq olunduğunu bildirir: məsələn, alt-sağ düyümü təqribən 0 Iris-Setosa, 1 Iris-Versicolor və 45 Iris-Virginica təlim hadisəsinə tətbiq olunur. Nəhayət, bir düyümün gini atributu onun Gini "natəmizliyini" ölçür: əgər düyüm tətbiq olunan bütün təlim hadisələri eyni sinifə aiddirsə həmin düyüm "təmiz"dir. Məsələn, dərinlik-1 sol düyümü ancaq Iris-Setosa təlim hadisələrinə tətbiq olunduğu üçün o "təmiz"dir və onun Gini hesabı 0-dır.

6-1 nömrəli bərabərlik hesablanma alqoritminin formulunu göstərir. Dərinlik-2 sol düyümünün Gini impurity hesabı $1 – (0/54)^2 – (49/54)^2 – (5/54)^2 ≈ 0.168$. bərabərliyi ilə hesalanır.

6-1. Gini "natəmizliyi" üçün bərabərlik:

$$Gi = 1 - \sum k = 1npi, k2$$

Bu bərabərlikdə:

* G_i Gini impurity-nin i nömrəli düyümüdür.
* $p_{i,k}$ i nömrəli düyümə tətbiq olunan təlim nümunələrinin, onlar arasında olan k sinifinin nümunələrinə olan nisbətidir.

NOT

Scikit-Öyrənmə, yalnızca ikili ağaclar (binary trees) yaradan CART alqoritmasını istifadə edir: yaprak düyümlərinin həmişə iki forması olur (məsələn, sualların cavabları yalnız bəli/xeyr olur). Halbuki, ID3 kimi başqa alqoritmalarda çoxlu formalı düyümlərə sahib Qərar Ağacları yaratmak mümkündür.

Şəkil 6-2 Qərar Ağacının qərar sərhədlərini göstərir. Qalın şaquli xətt əsas düyümün (dərinlik 0) qərar sərhədini təmsil edir: ləçək uzunluğu = 2.45 sm. Sol hissə tamamilə təmiz olduğuna görə (yalnızca Iris-Setosa), o daha da artıq bölünə bilməz. Lakin bunu sağ hissə haqqında deyə bilmərik, bu səbəbdən dərinlik-1 sağ düyümün ləçək eni = 1.75 sm (şəkildəki kəsik xətt ilə təsvir olunur). Max_depth (maksimum dərinlik) dəyəri 2-ə təyin olunduğuna görə, Qərar Ağacı orada dayanır. Əgər max_depth (maksimum dərinlik) dəyərini 3 olaraq təyin etsə idik, onda iki ədəd dərinlik-2 düyümü başqa bir qərar sərhədi daha əlavə edəcəkdir (şəkildəki kəsik xətt ilə təsvir olunur).

Şəkil 6-2. Qərar Ağacının qərar sərhədləri

İPUCU

Şəkil 6-1-də göstərilən bütün məlumatlatlar da daxil olmaqla ağaç strukturu təsnifatçının tree_atributu vasitəsile əldə edilə bilər. Təfərrüatlar üçün help(tree_clf.tree_) yazın, nümunə üçün bu fəslin qeyd dəftərinə baxın.

Model Təsviri: Ağ Qutu yoxsa Qara Qutu

Gördüyünüz kimi, Qərar Ağacları olduqca mükəmməldir və onların qərarları asanlıqla təsvir oluna bilir. Belə modellərə çox zaman “ağ qutu modelləri" adı da verilir. Əksinə, gördüyünüz kimi, təsadüfi meşələr və neyron səbəkələr ümumiyyətlə qara qutu modelləri kimi hesab olunur. Onlar mükəmməl proqnozlar verir və siz bu proqnozları verərkən edilmiş hesablama nəticələrini asanlıqla yoxlaya bilərsiniz; buna baxmayaraq, proqnozların nə üçün edildiyini sadə ifadələrlə izah etmək adətən çətin olur. Məsələn, neyron şəbəkəsi müəyyən bir şəxsin bir şəkildə göründüyünü söyləyirsə, bu proqnoza nəyin töhfə verdiyini bilmək çətindir: Model həmin şəxsin gözlərini tanıyıb? Onların ağzı? Onların burnu? Onların ayaqqabıları? Və ya hətta oturduqları divanı? Əksinə, qərar ağacları isə lazım olduqda hətta əl ilə tətbiq oluna biləcək qədər gözəl, sadə təsnifat qaydaları təmin edir (məsələn, çiçəklərin təsnifatı üçün). Şərh edilə bilən ML sahəsi öz qərarlarını insanların başa düşə biləcəyi şəkildə izah edə bilən ML sistemlərini yaratmağı hədəfləyir. Bu, bir çox sahələrdə vacibdir - məsələn, sistemin ədalətsiz qərarlar verməməsini təmin etmək.

Sinif Ehtimallarının Qiymətləndirilməsi

Bir Qərar Ağacı həmçinin nümunənin müəyyən bir k sinifə aid olma ehtimalını da qiymətləndirə bilər. Bu misalda, əvvəlcə o yarpaq düşünü tapma üçün ağacdan keçir, sonra isə bu qovşaqda k sinifinin təlim nümunələrinin nisbətini qaytarır. Məsələn, ləçəklərinin uzunluğu 5 sm və eni 1,5 sm olan bir çiçək tapmısınız. Müvafiq yarpaq düyünü dərinlik-2 sol düyündür, buna görə də qərar ağacı aşağıdakı ehtimalları verir: İris setosa üçün 0% (0/54), Iris versicolor üçün 90.7% (49/54) və Iris virginica üçün 9.3% (5/54). Ondan sinfi proqnozlaşdırmasını xahiş etsəniz, ən yüksək ehtimala malik olduğu üçün Iris versicolor (sinif 1) qaytaracaq. Gəlin bunu yoxlayaq:

```
>>> tree_clf.predict_proba([[5, 1.5]]).round(3)
array([[0. , 0.907, 0.093]])
>>> tree_clf.predict([[5, 1.5]])
array([1])
```

Mükəmməl! Diqqət yetirin ki, təxmin edilən ehtimallar Şəkil 6-2-nin sağ alt düzbucağının hər yerində eyni olacaq – məsələn, ləçəklər 6 sm uzunluğunda və 1,5 sm enində olsaydı (bunun çox güman ki, İris virginika olacağı aydın görünsə də)

CART Təlim Alqoritmi

Scikit-Öyrənmə, Qərar Ağaclarını təlim etmək üçün Klassifikasiya və Regressiya Ağacı (CART) alqoritmindən istifadə edir (həmçinin "böyüyən" ağaclar kimi adlandırılır). Alqoritm əvvəla bir təlim kursunu iki alt-qrupa bölmək üçün tək xüsusiyyətli k və bir eşik t_k (məsələn, "ləpə uzunluğu ≤ 2.45 sm") istifadə edir. K və t_k necə seçilir? Alqoritm, ən təmiz alt-qrupları istehsal edən cüt (k, t_k) axtarır. 6-2 nömrəli bərabərlik alqoritmin minimuma endirməyə çalışdığı xərc funksiyasını verir.

6-2 Nömrəli bərabərlik. Klassifikasiya üçün CART maliyyə funksiyası:

J (k , t k) = m sol m G sol + m sağ m G sağ harada ki G sol/sağ sol/sağ alt çoxluğun çirkliliyini ölçür m sol/sağ sol/sağ alt çoxluqdakı nümunələrin sayıdır.

CART alqoritmi nümunəni uğurla iki hissəyə böldükdən sonra eyni məntiqdən istifadə edərək alt çoxluqları, sonra isə alt-alt çoxluqları və s. rekursiv şəkildə bölür. Maksimum dərinliyə çatdıqdan sonra (maksimum_dərinlik hiperparametri ilə müəyyən edilir) və ya çirkləri azaldacaq bir parçalanma tapa bilmədikdən sonra təkrarlanma dayandırılır. Bir neçə digər hiperparametrlər (bir anda təsvir olunur) əlavə dayandırma şərtlərinə nəzarət edir: min_samples_split (minimum_nümunə_bölgüsü), min_samples_leaf (minimum_nümunə_yarpağı), minimum__fraction_leaf (minimum_yarpaq_fraksiyası) və max_leaf_nodes maksimum_yarpaq_düyünləri).

Xəbərdarlıq:

Gördüyünüz kimi, CART alqoritmi *acgöz* bir alqoritmdir: o, acgözlüklə yuxarı səviyyədə optimal bölünmə axtarır, sonra isə həmin prosesi hər bir sonrakı səviyyədə təkrarlayır. Parçalanmanın mümkün olan ən aşağı çirklənməni daha bir neçə səviyyə aşağı da çıxarıb-çıxarmayacağını yoxlamır. Acgöz alqoritm tez-tez kifayət qədər yaxşı, lakin optimal olacağı zəmanət verilməyən həllər istehsal edir.

Təəssüf ki, optimal ağacın tapılması *NP-tam* problem kimi hesab olunur. O (exp(m)) vaxt tələb edir və bu problemi hətta kiçik nümunə dəstləri üçün də həll olunmaz edir. Buna görə qərar ağaclarını öyrənərkən "məqbul dərəcədə yaxşı" həll yolu ilə kifayətlənməliyik.

Hesablama mürəkkəbliyi

Proqnozlar vermək qərar ağacını kökdən yaprağa kimi keçməyi tələb edir. Qərar ağacları ümumilikdə təxminən balanslıdırlar, bu səbəbdən qərar ağacını keçmək üçün təxminən O(log2(m)) qovşaqlarından keçmək lazımdır harada ki log2(m) m-nin ikili logarifmidir. Hər bir düyün yalnız bir xüsusiyyətin dəyərinin yoxlanılmasını tələb etdiyinə görə, ümumi proqnozlaşdırma mürəkkəbliyi funksiyaların sayından asılı olmayaraq O(log2(m)) təşkil edir. Beləliklə, böyük təlim dəstləri ilə məşğul olanda belə proqnozlar çox sürətli olur. Təlim alqoritmi hər bir qovşaqdakı bütün nümunələrdə bütün xüsusiyyətləri (max_features təyin olunarsa daha az) müqayisə edir. Hər bir düyündə bütün nümunələrdəki bütün xüsusiyyətlərin müqayisəsi O(n × m log2(m)) mürəkkəbliyi ilə nəticələnir.

Gini Təmizliyi yoxsa Entropiya?

Əsasən, DecisionTreeClassifier (QərarAğacıTəsnifatı) sinifi Gini impurity ölçüsünü istifadə edir, lakin siz bunun əvəzinə meyar hiperparametrini "entropiya" olaraq təyin edərək entropiya impurity ölçüsünü seçə bilərsiniz. Entropiya anlayışı əvvəla termodinamikada molekulyar pozğunluğun ölçüsü kimi yaranmışdır: molekullar hərəkətsiz və nizamlı olduqda entropiya sıfıra yaxınlaşır. Daha sonra entropiya konsepti müxtəlif sahələrə, o cümlədən Şennonun məlumat nəzəriyyəsinə də yayıldı. Entropiya burada gördüyümüz kimi mesajın orta məlumat məzmununu ölçür ki, bu barədə 4. bölmədə məlumat tapa bilərsiniz. Entropiya bütün mesajlar eyni olduğu zaman sıfıra bırabər olur. Maşın-öyrənmədə entropiya tez-tez bir impurity ölçüsü kimi istifadə olunur: çoxluğun entropiyası həmin çoxluq yalnız bir sinifdən ibarət olduqda sıfırdır. 6-3-cü tənlik i-ci düyünün entropiyasının tərifini göstərir. Məsələn, Şəkil 6-1-dəki dərinlik-2 sol qovşağının entropiyası – (49/54) $\log_2$ (49/54) – (5/54) $\log_2$ (5/54) ≈ 0.445 kimi bir qiymətə malikdir.

Tənlik 6-3. Entropiya

$$Hi = -\sum k = 1pi, k \neq 0npi, klog2(pi, k)$$

Beləliklə, siz Gini impurity-ni yoxsa entropiyanı istifadə etməlisiniz? Əslində, əksər hallarda bunlar böyük bir fərq yaratmazlar: onlar oxşar ağacların yaranmasına səbəb olurlar. Gini impurity-nin hesablanması bir az daha sürətli olduğuna görə o daha yaxşı standartdır. Ancaq, onlar fərqləndikdə, Gini impurity ağacı onun öz şöbəsində ən çox təkrarlanan sinifi izolə etməyə meyllidir, əksinə, entropiya isə bir az daha balanslı ağaclar yaradır.

Tənzimlənmə hiperparametrləri

Qərar ağaclarının təlim məlumatları ilə bağlı çox az fərziyyə irəli sürür (verilənlərin xətti olduğunu güman edən xətti modellərdən fərqli olaraq). Məhdudiyyətsiz qaldıqda, ağac strukturu özünü təlim məlumatlarına uyğunlaşdıracaqç hətta çox güman ki, həddindən artıq uyğunlaşdıracaq. Belə model tez-tez *qeyri-parametrik model* adlanır, çünki onun heç bir parametri yoxdur, lakin parametrlərin sayı təlimdən əvvəl müəyyən edilmədiyinə görə model strukturu məlumatlara yaxından yapışmaqda sərbəstdir. . Bunun əksinə, xətti model kimi parametrik model əvvəlcədən müəyyən edilmiş sayda parametrlərə malikdir, buna görə də onun sərbəstlik dərəcəsi məhduddur ki bu da həddindən artıq uyğunlaşma riskini azaldır (lakin az uyğunlaşma riskini artırır).

Təlim məlumatlarına həddindən artıq uyğunlaşmanı aradan qaldırmaq üçün məşq zamanı qərar ağacının azadlığını məhdudlaşdırmalısınız. Bildiyiniz kimi, buna nizamlama deyilir. Tənzimləmə hiperparametrləri istifadə olunan alqoritmdən asılıdır, lakin ümumiyyətlə, ən azından qərar ağacının maksimum dərinliyini məhdudlaşdıra bilərsiniz. Scikit-Öyrənmədə bu, max_depth hiperparametri (maksimum_dərinlik_hiperparametri) ilə idarə olunur. Varsayılan dəyər None-dir, yəni limitsizdir. max_depth-in (maksimum_dərinliyin) azaldılması modeli nizamlayacaq və beləliklə, həddindən artıq uyğunlaşma riskini azaldacaq. DecisionTreeClassifires (QərarAğacıTəsnifatı) sinfində qərar ağacının formasını eyni şəkildə məhdudlaşdıran bir neçə başqa parametr var:

max_features (maksimum_xüsusiyyət)

Hər qovşaqda bölünmə üçün qiymətləndirilən xüsusiyyətlərin maksimum sayı

max_leaf_nodes (maksimum_yarpaq_düyünləri)

Yarpaq düyünlərinin maksimum sayı

min_samples_split (minimum_nümunə_bölünməsi)

Bir düyümün bölünməzdən əvvəlki nümunələrin minimum sayı

min_samples_leaf (minimum_nümunə_yarpağı)

Bir yarpaq düyünü yaratmaq üçün minimum nümunə sayı

min_weight_fraction_leaf (Minimum_çəki_fraksiya_yarpağı)

Min_samples_leaf (minimum_nümunə_yarpağı) ilə eynidir, lakin ölçülmüş nümunələrin ümumi sayının bir hissəsi kimi ifadə edilir

Min_* hiperparametrləri artırmaq və ya max_* hiperparametrləri azaltmaq modeli nizamlayacaq.

NOT

Digər alqoritmlər əvvəlcə qərar ağacını məhdudiyyətsiz öyrətməklə, sonra lazımsız qovşaqları budamaqla (silməklə) işləyir. Övladlarının hamısı yarpaq düyünləri olan düyünün təmin etdiyi təmizliyin statistik cəhətdən əhəmiyyətli olmaması onu lazımsız edir. χ^2 testi (xi-kvadrat testi) kimi standart statistik testlər təkmilləşmənin sırf təsadüf nəticəsində olması ehtimalını qiymətləndirmək üçün istifadə olunur (buna *null hipotezi* deyilir). Əgər p-dəyəri adlanan bu ehtimal verilmiş həddən yüksəkdirsə (adətən 5%, hiperparametrlə idarə olunur), onda düyün lazımsız hesab edilir və onun balaca düyünləri silinir. Budama bütün lazımsız düyünlər budanana və ya silinənə qədər davam edir.

Gəlin 5-ci Fəsildə təqdim olunan aylar çoxluğunda nizamlanmanı sınaqdan keçirək. Biz bir qərar ağacını nizamlamadan, digərini isə min_samples_leaf=5 (minimum_nümunə_yarpağı=5) ilə öyrədəcəyik. Kod aşağıdaki kimidir; Şəkil 6-3 hər bir ağacın qərar sərhədlərini göstərir:

```
from sklearn.datasets import make_moons

X_moons, y_moons = make_moons(n_samples=150, noise=0.2, random_state=42)

tree_clf1 = DecisionTreeClassifier(random_state=42)
tree_clf2 = DecisionTreeClassifier(min_samples_leaf=5, random_state=42)

tree_clf1.fit(X_moons, y_moons)
tree_clf2.fit(X_moons, y_moons)
```

Şəkil 6-3. Tənzimlənməmiş ağacın (solda) və nizamlı ağacın (sağda) qərar sərhədləri

Aydındır ki, soldakı nizamsız model həddən artıq uyğundur və sağdakı nizamlanmış model, yəqin ki, daha yaxşı ümumiləşdirməyə malikdir. Biz bunu fərqli təsadüfi toxumdan istifadə edərək yaradılan test dəstində hər iki ağacı qiymətləndirməklə yoxlaya bilərik:

```
>>> X_moons_test, y_moons_test = make_moons(n_samples=1000, noise=0.2,
...                                         random_state=43)
>>> tree_clf1.score(X_moons_test, y_moons_test)

0.898
>>> tree_clf2.score(X_moons_test, y_moons_test)

0.92
```

Həqiqətən, ikinci ağac test dəstində daha yaxşı dəqiqliyə malikdir.

Reqressiya

Qərar ağacları həm də reqressiya tapşırıqlarını yerinə yetirməyə qadirdir. Gəlin, Scikit-Öyrənmənin DecisionTreeRegressor (QərarAğacларıReqressoru) sinfindən istifadə edərək reqressiya ağacı quraq və onu max_depth=2 (maksimum_dərinlik=2) olan səsli kvadratik verilənlər bazasında məşq edək:

```
import numpy as np
from sklearn.tree import DecisionTreeRegressor
np.random.seed(42)
X_quad = np.random.rand(200, 1) - 0.5 # a single random input feature

y_quad = X_quad ** 2 + 0.025 * np.random.randn(200, 1)

tree_reg = DecisionTreeRegressor(max_depth=2, random_state=42)

tree_reg.fit(X_quad, y_quad)
```

Yaranan ağaç Şəkil 6-4-də göstərilmişdir.

Şəkil 6-4 Reqessiya üçün qərar ağacı

Bu ağac əvvəllər qurduğunuz təsnifat ağacına çox bənzəyir. Əsas fərq ondan ibarətdir ki, hər bir qovşaqda bir sinif proqnozlaşdırmaq əvəzinə, bir dəyəri proqnozlaşdırılırş Məsələn, tutaq ki, siz $x_1 = 0.2$ olan yeni bir nümunə üçün proqnoz vermək istəyirsiniz. Kök düğünü sizdən $x_1 \leq 0.197$ olub olmadığını soruşur. Düzgün olmadığına görə alqoritm $x_1 \leq 0.772$ olub-olmadığını soruşan sağ alt düyünə keçir. Bunun düzgün olduğuna görə alqoritm sol yetirmə düyünə keçir. Bu yarpaq qovşağıdır və dəyəri 0,111 ilə proqnozlaşdırır. Bu proqnoz bu yarpaq qovşağı ilə əlaqəli 110 təlim nümunəsinin orta hədəf dəyəridir və bu 110 nümunə üzərindən 0,015-ə bərabər olan orta kvadrat xəta ilə nəticələnir.

Bu modelin proqnozları Şəkil 6-5-də solda göstərilmişdir. Əgər max_depth=3 (maksimum_dərinlik=3) təyin etsəniz, sağda göstərilən proqnozları alırsınız. Hər bir bölgə üçün proqnozlaşdırılan dəyərin həmişə həmin bölgədəki nümunələrin orta hədəf dəyəri olduğuna diqqət yetirin. Alqoritm hər bölgəni elə bölür ki, əksər təlim nümunələrini həmin proqnozlaşdırılan dəyərə mümkün qədər yaxın edir.

Şəkil 6-5. İki qərar ağacı reqressiya modelinin proqnozları

CART alqoritmi əvvəllər təsvir edildiyi kimi işləyir, təlim dəstini çirkləri minimuma endirəcək şəkildə bölməyə çalışmaq əvəzinə, onun MSE-ni minimuma endirəcək şəkildə bölməyə çalışması istisna olmaqla. 6-4-cü bərabərlik alqoritmin minimuma endirməyə çalışdığı xərc funksiyasını göstərir.

Tənlik 6-4. Reqressiya üçün CART xərc funksiyası

$$J(k,tk) = mleftmMSEleft + mrightmMSErightwhereMSEnode = \sum i \in node(y^node - y(i))2mnodey^node = \sum i \in nodey(i)mnode$$

Təsnifat tapşırıqlarında olduğu kimi, qərar ağacları da reqressiya tapşırıqları ilə məşğul olan zaman həddindən artıq uyğunlaşmaya meyllidir. Heç bir nizamlanma olmadan (yəni, standart hiperparametrlərdən istifadə etməklə) Şəkil 6-6-da solda olan proqnozları əldə edirsiniz.

Bu proqnozlar, aydındır ki, məşq dəstinə düzgün uyğun gəlmir. Sadəcə min_samples_leaf=10 (minimum_nümunə_yarpaqları) təyin etmək Şəkil 6-6-da sağda göstərilən kimi daha uyğun bir modellə nəticələnəcək.

Şəkil 6-6. Tənzimlənməmiş reqressiya ağacının (solda) və nizamlı ağacın (sağda) proqnozları

Ox (axis) oriyentasiyasına həssaslıq

Ümid edirik ki, siz artıq qərar ağaclarının bir çox xüsusiyyətlərə malik olduğuna əminsiniz: onları başa düşmək və şərh etmək nisbətən daha asandır, istifadəsi sadə, çox yönlü və güclüdür. Bununla belə, onların bəzi məhdudiyyətləri var. Birincisiç artıq fərqinə vardığınız kimi, qərar ağacları ortoqonal qərar sərhədlərini sevir (bütün bölünmələr oxa perpendikulyardır), bu da onları məlumatların oriyentasiyası barəsində həssas qılır. Məsələn, Şəkil 6-7 sadə, xətti olaraq ayrıla bilən verilənlər toplusunu göstərir: solda qərar ağacı onu asanlıqla parçalaya bilər, sağda isə verilənlər dəsti 45° fırlanandan sonra qərar sərhədi lazımsız şəkildə bükülmüş görünür. Hər iki qərar ağacı təlim dəstinə mükəmməl uyğun gəlsə də, çox güman ki, sağdakı model yaxşı ümumiləşdirə bilməyəcək.

Şəkil 6-7. Təlim dəstinin fırlanmasına həssaslıq

Bu problemi məhdudlaşdırmağın bir yolu verilənlərin miqyasını artırmaq, sonra əsas element analizinin transformasiyasını tətbiq etməkdir. PCA-ya Fəsil 8-də ətraflı baxacayıq, lakin hələlik yalnız bilmək lazımdır ki, o, verilənləri xüsusiyyətlər arasında korrelyasiyanı azaldacaq şəkildə fırladır ki, bu da tez-tez (həmişə deyil) ağaclar üçün işləri asanlaşdırır.

Gəlin məlumatları miqyaslandıran və onu PCA-dan istifadə edərək döndərən kiçik bir boru xətti yaradaq, sonra həmin məlumatlarda DecisionTreeClassifier-i (QərarAğacıTəsnifatı) məşq edək. Şəkil 6-8 həmin ağacın qərar sərhədlərini göstərir: gördüyünüz kimi, fırlanma sadəcə bir xüsusiyyətdən, ilkin ləçək uzunluğunun və eninin xətti funksiyası olan z_1 funksiyasından istifadə edərək verilənlər toplusunu kifayət qədər yaxşı uyğunlaşdırmağa imkan verir. Kod belədir:

```
from sklearn.decomposition import PCA
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

pca_pipeline = make_pipeline(StandardScaler(), PCA())
X_iris_rotated = pca_pipeline.fit_transform(X_iris)
tree_clf_pca = DecisionTreeClassifier(max_depth=2, random_state=42)
tree_clf_pca.fit(X_iris_rotated, y_iris)
```

Şəkil 6-8. Ölçülmüş və PCA ilə fırlanan iris verilənlər bazasında ağacın qərar sərhədləri

Qərar Ağaclarının bir çox Fərqlilikləri Var

Ümumiyyətlə, qərar ağacları ilə bağlı əsas məsələ onların kifayət qədər yüksək dispersiyaya malik olmasıdır: hiperparametrlərdə və ya verilənlərdə edilən kiçik dəyişikliklər çox fərqli modellər yarada bilər. Əslində, Scikit-Öyrənmə tərəfindən istifadə edilən təlim alqoritmi stoxastik olduğundan - o hər bir qovşaqda qiymətləndirmək üçün xüsusiyyətlər dəstini təsadüfi seçir - hətta eyni qərar ağacının eyni məlumat üzərində yenidən hazırlanması belə çox fərqli bir model yarada bilər. Şəkil 6-9-da göstərilmişdir (random_state (təsadüfi_vəziyyət) hiperparametrini təyin etmədiyiniz halda). Gördüyünüz kimi, əvvəlki qərar ağacından çox fərqli görünür (Şəkil 6-2).

Şəkil 6-9. Eyni modelin eyni verilənlər üzərində yenidən hazırlanması çox fərqli bir model yarada bilər

Xoşbəxtlikdən, bir çox ağac üzərində proqnozların orta hesabını alaraq, fərqi əhəmiyyətli dərəcədə azaltmaq mümkündür. Belə ağac ansamblı təsadüfi meşə adlanır və bu, növbəti fəsildə görəcəyiniz kimi günümüzdə mövcud olan ən güclü model növlərindən biridir.

Tapşırıqlar

1. İçində bir milyon nümunə olan təlim dəstində hazırlanmış qərar ağacının (məhdudiyyətlər olmadan) təxmini dərinliyi nədir?

2. Bir qovşağın Gini çirkliliyi, ümumiyyətlə, valideynindəkindən daha aşağıdır, yoxsa daha yüksəkdir? Çox vaxt daha aşağı/daha yüksək olur, yoxsa həmişə daha aşağı/daha yüksəkdir?

3. Əgər məşq dəstinə həddindən artıq uyğun gəlirsəç max_depth-i(maksimum_dərinliyi) azaltmaq yaxşı fikirdir?

4. Əgər qərar ağacı təlim dəstinə az uyğun gəlirsə, daxiletmə xüsusiyyətlərini genişləndirmək yaxşı fikirdir?

5. Qərar Ağacını bir milyon nüsxədən ibarət olan təlim dəstində işlətmək bir saat vaat aparırsa, on milyon nümunədən ibarət təlim dəstində başqa bir qərar ağacını işlətmək təxminən nə qədər vaxt aparacaq? İpucu: CART alqoritminin hesablama mürəkkəbliyini nəzərdən keçirin.

6. Verilmiş təlim dəstində qərar ağacının hazırlanması bir saat var aparırsa, funksiyaların sayını iki dəfə artırsanız, təxminən nə qədər vaxt aparacaq?

7. Aşağıdakı addımları yerinə yetirərək ayların məlumat dəsti üçün qərar ağacını hazırlayın və dəqiqləşdirin:

    a. Ay məlumat dəstini yaratmaq üçün make_moons (yarat_ay) (n_samples(nümunələr)=10000, noise(səs)=0.4) istifadə edin.

    b. Məlumat dəstini təlim dəstinə və test dəstinə bölmək üçün train_test_split() (təlim_tes_ istifadə edin.

    c. DecisionTreeClassifier-da (QərarAğacıTəsnifatı)yaxşı hiperparametr dəyərləri tapmaq üçün çarpaz doğrulama ilə (GridSearchCV sinifinin köməyi ilə) şəbəkə axtarışından istifadə edin. İpucu: max_leaf_nodes (maksimum_yarpaq_düyünləri) üçün müxtəlif dəyərləri sınayın.

d. Bu hiperparametrlərdən istifadə edərək onu tam təlim dəsti üzərində məşq edin və test dəstində modelinizin performansını ölçün. Təxminən 85% -dən 87% -ə qədər dəqiqlik əldə etməlisiniz.

8. Bu addımları yerinə yetirərək bir meşə yetişdirin:

   a. Əvvəlki məşqi davam etdirərək, hər biri təsadüfi seçilmiş 100 nümunədən ibarət təlim dəstinin 1000 alt dəstini yaradın. İpucu: bunun üçün Scikit-Öyrənmənin ShuffleSplit (QarışıqBölünmə) sinfindən istifadə edə bilərsiniz.

   b. Əvvəlki məşqdə tapılan ən yaxşı hiperparametr dəyərlərindən istifadə edərək, hər bir alt çoxluqda bir qərar ağacını məşq edin. Bu 1000 qərar ağacını test dəstində qiymətləndirin. Daha kiçik dəstlər üzərində öyrədildikləri üçün bu qərar ağacları ilk qərar ağacından daha pis çıxış edəcək və sadəcə təxminən 80% dəqiqliyə nail olacaq.

   c. İndi isə sehr hissəsi gəlir. Hər bir test dəsti nümunəsi üçün 1000 qərar ağacının proqnozlarını yaradın və yalnız ən tez-tez verilən proqnozu saxlayın (bunun üçün SciPy-nin mode() funksiyasından istifadə edə bilərsiniz). Bu yanaşma sizə test dəsti üzərində səs çoxluğu ilə bağlı proqnozlar verir.

   d. Test dəstində bu proqnozları qiymətləndirin: indi ilk modelinizdən bir qədər yüksək dəqiqlik əldə etməlisiniz (təxminən 0,5-1,5% yüksək). Təbrik edirik, təsadüfi meşə təsnifatını hazırladınız!

Bu məşqlərin həlli bu fəslin sonunda, https://homl.info/colab3 ünvanında mövcuddur.

1 P *polinominal zamanda* həll edilə bilən problemlər çoxluğudur (yəni verilənlər dəstinin ölçüsünün polinomu). NP həlli polinominal zamanda yoxlanıla bilən məsələlər toplusudur. NP-çətin məsələsi polinom zamanında məlum NP-çətin məsələyə endirilə bilən problemdir. NP-tam problem həm NP, həm də NP-çətindir. Əsas açıq riyazi sual P = NP olub-olmamasıdır. Əgər P ≠ NP (bu ehtimal görünür), onda heç bir NP tam problemi üçün heç bir polinom alqoritmi tapılmayacaq (bəlkə də bir gün kvant kompüterində tapılacaq).

2 Daha ətraflı məlumat üçün Sebastian Raschka-nın maraqlı təhlilinə baxın.

English	Azerbaijani
Node	Düyün
Impurity	?
Train	Təlim, Məşq
Scikit-Learn	Scikit-Öyrənmə
Dataset	Verilənlər çokluğu
Data	Verilən Məlumat
Children	Yetirmə
Null	?
Axis	Ox