Hafta 8 (DecisionTrees - Random Forests - Gradient Boosted Trees - Light GBM - XGBoost - Automated Hypterparameter Optimization - Automated ML Pipeline)

@mebaysan

16/10/2021

İlgili Okuma Listesi

- https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/1e5b123b-4fe1-4dd5-b006-cac372077dd6/8.hafta_carsamba_konu_ozeti.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/61a213c5-0b25-42ef-adfe-2dd46efdf1be/8.hafta_persembe_konu_ozeti.pdf https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e0bcdc3a-3a16-4147-9be5-57aa00cd03c9/GiniHesaplama.pdf

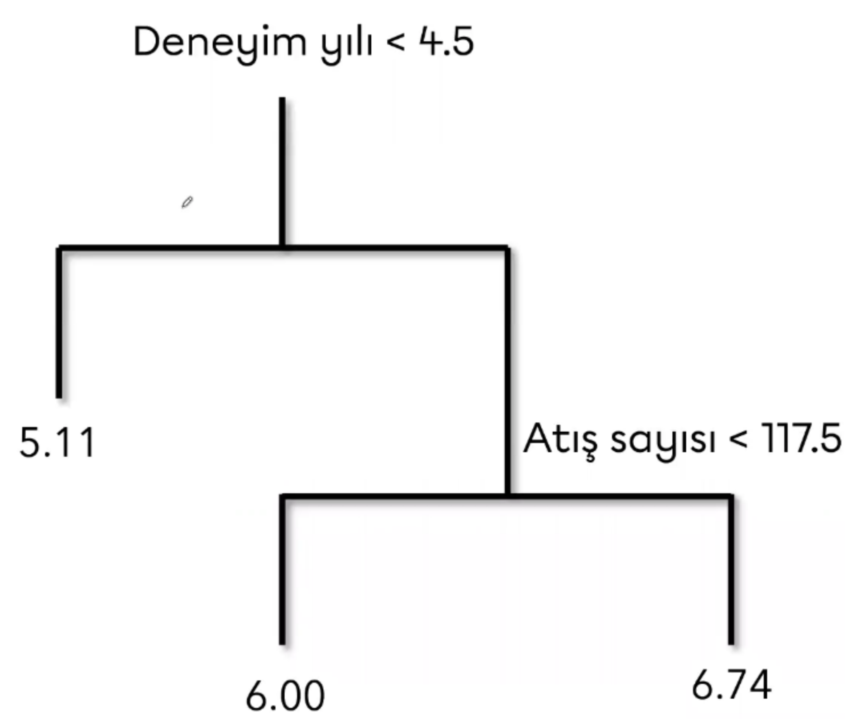
Benim Yazdığım Yazılar:

-

Classification And Regression Tree (CART)

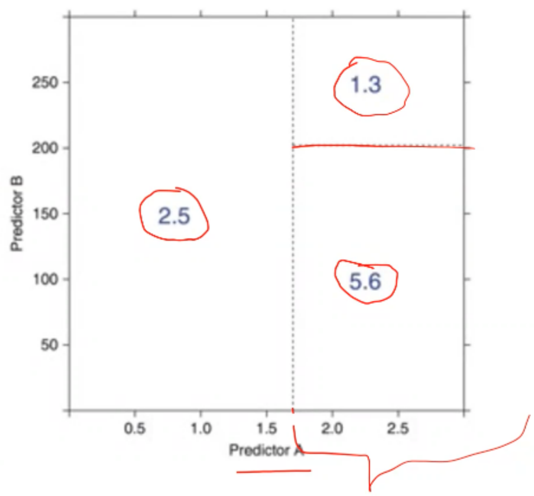
Amaç veri seti içerisindeki karmaşık yapıları basit karar yapılarına dönüştürmektir. Heterojen veri setleri ile belirlenmiş bir hedef değişkene göre homojen alt gruplara ayrılır.

Örnek bir karar ağacı: 5 yıl deneyimli ve atış sayısı 120 olan bir kişinin maaşı 6.74'tür



Regresyon problemlerinde temel mantık şudur: bağımsız değişkenleri bir yerlerden böleceğiz ve kalan değerlerin ortalamasını alacağız.

```
if Predictor A >= 1.7 then
|   if Predictor B >= 202.1 then Outcome = 1.3
|   else Outcome = 5.6
else Outcome = 2.5
```



Karar Ağaçlarında Regresyon Problemleri İçin Cost Fonksiyonu

$$\text{RSS (SSE):} \quad \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

R_j: bölge/yaprak/kutu

Karar Ağacında Örnek Bir Dallanma

Örnek olarak aşağıdaki bağımsız değişkeni (x) 70 noktasından böldük. Kalan değerlerin ortalamasını alıp, bağımlı değişkenin (y) yeni değerini (y_hat) **böldüğümüz noktanın dışında kalan tüm gözlemlerin ortalamasını** yeni bağımlı değişken değeri (y_pred, y_hat) olarak atıyoruz (her iki bölüm için).

metre_kare (x)	fiyat (y)	y_pred (y_hat)
70	300	300
73	310	452
75	310	452
80	350	452
82	350	452
83	400	452
85	420	452
90	450	452
92	450	452
94	470	452
100	490	452
105	500	452
107	510	452
110	520	452
115	530	452
118	530	452
120	550	452
125	550	452

Hatayı (SSE) hesaplamak için; her bir gözlem için bağımlı değişkenin tahmin öncesi değerinden (y) tahmin edilen değerini (y_pred, y_hat) çıkarırız. Elde ettiğimiz hataların karelerini alır toplarız.

metre_kare (x)	fiyat (y)	y_pred (y_hat)	(y - y_pred)	(y - y_pred)^2
70	300	300	0	0
73	310	452	-142	20164
75	310	452	-142	20164
80	350	452	-102	10404
82	350	452	-102	10404
83	400	452	-52	2704
85	420	452	-32	1024
90	450	452	-2	4
92	450	452	-2	4
94	470	452	18	324
100	490	452	38	1444
105	500	452	48	2304
107	510	452	58	3364
110	520	452	68	4624
115	530	452	78	6084
118	530	452	78	6084
120	550	452	98	9604
125	550	452	98	9604
				108308

Peki doğru yerden bölüp bölmediğimizi nerden bilebiliriz? En az SSE'yi elde edene kadar böleriz ve en küçük SSE'yi veren bölüm noktası o bağımsız değişken için bölüm noktası olur.

metre_kare (x)	fiyat (y)	yi-y_ort	(yi-y_ort)^2
70	300	-130	16900
73	310	-120	14400
75	310	-120	14400
80	350	-80	6400
82	350	-80	6400
83	400	-30	900
85	420	-10	100
90	450	20	400
92	450	20	400
94	470	40	1600
100	490	60	3600
105	500	70	4900
107	510	80	6400
110	520	90	8100
115	530	100	10000
118	530	100	10000
120	550	0	0
125	550	0	0
			104900

Bölüm noktaları (x)	Hatalar (SSE)	MSE?	MAE?
73	108308		
75	86826		
94	41242		
120	104900		

1'den fazla bağımsız değişkenimiz olduğunda da bu işlemi tüm bağımsız değişkenler için uygularız.

MSE'yi bulmak için SSE'i gözlem sayısına böleriz.

MAE'yi bulmak için ise SSE hesaplarken hatanın karesini almayız da mutlak değerini alırız ve gözlem sayısına böleriz.

Feature Importance

Hatayı en çok azaltan değişken en önemli değişkendir. Karar ağaçlarının en tepesindeki değişken en önemli değişkendir.

Sınıflandırma Problemleri İçin Cost Fonksiyonu

K → Sınıf sayısı. 2 sınıf olduğunu düşünelim: 1 ve 0. K = 2'dir bu durumda.

Gini (LG) → İlgili yapraktaki; 1 sınıfının gerçekleşmesi olasılığı * 1 sınıfının gerçekleşmemesi olasılığı + 0 Sınıfının gerçekleşmesi olasılığı * 0 sınıfının gerçekleşmemesi olasılığı

$$\mathcal{L}_G(\mathcal{N}_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

$$\mathcal{L}_E(\mathcal{N}_m) = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

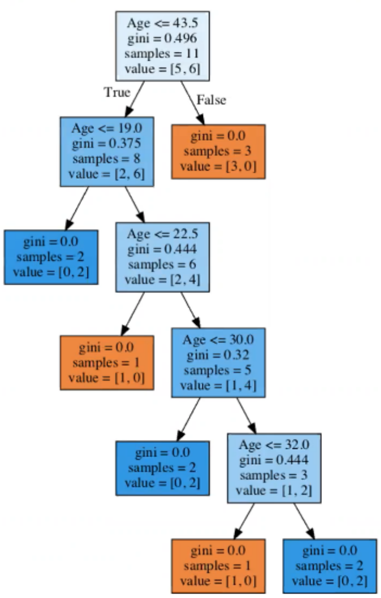
The weighted loss: $\mathcal{L}(S_m) = f_L \cdot \mathcal{L}(C_m^L) + f_R \cdot \mathcal{L}(C_m^R).$

Örnek Gini Hesaplama

1. kutu (node, bölündü) için hesaplama (0.496):

- (5/11 * (1 - 5/11)) + (6/11 * (1 - 6/11))
  - 5/11 * (1 - 5/11)
    - 5 / 11
      - 11 → İlgili node'da bulunan toplam gözlem sayısı
      - 5 → X sınıfından olan 5 gözlem
      - 5 / 11 → Bir gözlemin X sınıfında olma olasılığı
    - 1 - (5 / 11)
      - 1 → %100
      - 5 / 11 → Bir gözlemin X sınıfında olma olasılığı
      - 1 - (5 / 11) → Bir gözlemin X sınıfında **olmama** olasılığı

    - 6/11 * (1 - 6/11)
        - 6 / 11
            - 11 → İlgili node'da bulunan toplam gözlem sayısı
            - 6 → Y sınıfında olan 6 gözlem
            - 6 / 11 → Bir gözlemin Y sınıfında olma olasılığı
        - 1 - (6 / 11)
            - 1 → %100
            - 6 / 11 → Bir gözlemin Y sınıfında olma olasılığı
            - 1 - (6 / 11) → Bir gözlemin Y sınıfında **olmama** olasılığı
    - (5/11 * (1 - 5/11)) + (6/11 * (1 - 6/11)) → (Bir gözlemin X sınıfında olma olasılığı * Bir gözlemin X sınıfında olmama olasılığı) + (Bir gözlemin Y sınıfında olma olasılığı * Bir gözlemin Y sınıfında olmama olasılığı) = Gini
- Python'da: `round((5/11 * (1 - 5/11)) + (6/11 * (1 - 6/11)),3)`



$$\mathcal{L}_G(\mathcal{N}_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

Aşağı doğru her node için bu işlemler yapılarak Gini bulunur. Gini 0 ise o node'da çeşitlilik yoktur diyebiliriz.

Random Forests (Rassal Ormanlar)

Temeli birden çok karar ağacının ürettiği tahminlerin bir araya getirilerek değerlendirilmesine dayanır.

Bagging (Breiman, 1996) ile Random Subspace (Ho, 1998) yöntemlerinin birleşiminden oluşmuştur.

Random Forests için **gözlemler bagging yöntemi** ile **değişkenler random subspace yöntemi** ile seçilir.

Karar ağacının her bir düğümünde en iyi dallara ayırıcı (bilgi kazanıcı) değişken tüm değişkenler arasından rastgele seçilen daha az sayıdaki değişken arasından seçilir.

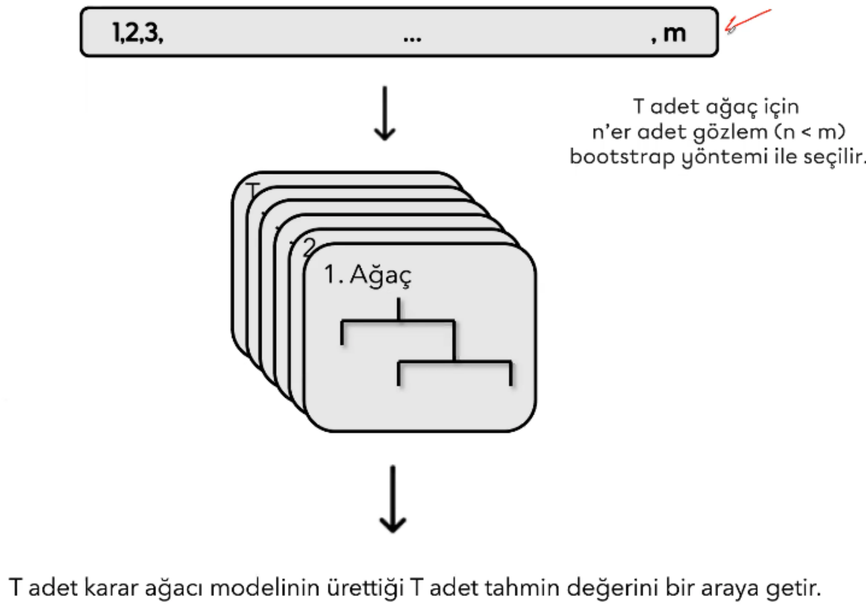
Ağaç oluşturmada veri setinin 2/3'ü kullanılır. Dışarıda kalan veri ağaçların performans değerlendirmesi ve değişken öneminin belirlenmesi için kullanılır.

Her düğüm noktasında rastgele değişken seçimi yapılır. (regresyon'da p/3, sınıflamada karekök p)

Random Forests'da veri setini train ve test olmak üzere ayırmamıza gerek yoktur. RF bunu kendi içerisinde yapmaktadır.

Bagging (Bootstrap Aggregation)

Veri setinden rastgele N gözlem seçip ağaç fit edilir. Sonra bir başka N gözlem seçilip bir başka ağaç fit edilir.



Random Subspace

Veri setindeki değişkenlerden rastgele değişkenler seçilir ve bunlar üzerine model kurulur.

Gradient Boosting Machines (GBM)

Hatalar/artıklar üzerine tek bir tahminsel model formunda olan modeller serisi kurulur.

- Boosting + Gradient Descent
- Gradient boosting tek bir tahminsel model formunda olan modeller serisi oluşturulur
- Seri içerisindeki bir model serideki bir önceki modelin tahmin artıklarının/hatalarının (residuals) üzerine kurularak (fit) oluşturulur
- GBM diferansiyellenebilen herhangi bir kayıp fonksiyonunu optimize edebilen Gradient Descent algoritmasını kullanmaktadır
- Tek bir tahminsel model formunda olan modeller serisi additive (eklemeli) şeklinde kurulur

Boosting yöntemleri ile bagging yöntemleri arasında ne fark vardır?

Boosting yöntemlerinde modeller birbirine bağlıdır.

Bagging yöntemlerinde modeller birbirinden bağımsızdır.

Ada Boost (Adaptive Boosting)

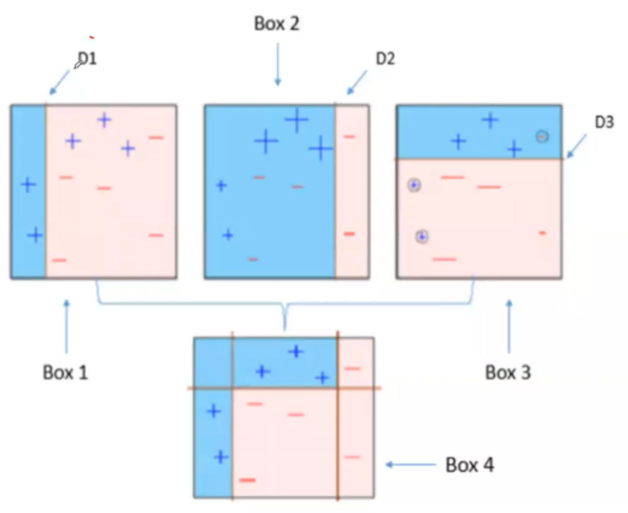
GBM yönteminin temellerindendir.

Zayıf sınıflandırıcıları bir araya getirerek güçlü bir sınıflandırıcı oluşturması fikrine dayanır.

Örnek: Amacımız artı ve eksileri sınıflandırmak.

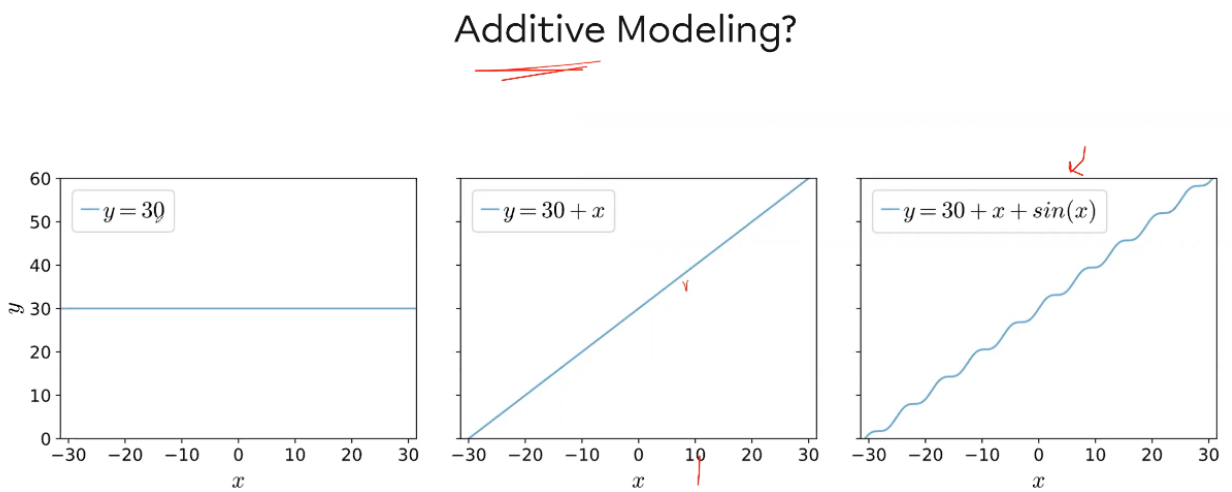
Box(i) → İterasyon

Box(i), Box(i-1)'e bağımlıdır. Her iterasyonda bir önceki sınıflandırıcı dikkate alınarak yeni bir sınıflandırıcı oluşturulur.



Additive Modeling

Hatayı modelleyerek yeni bir terim ekleyeceğiz. Ada Boost'taki her iterasyonda yapılan bölümleme işlemini hassaslaştırmış olacağız.

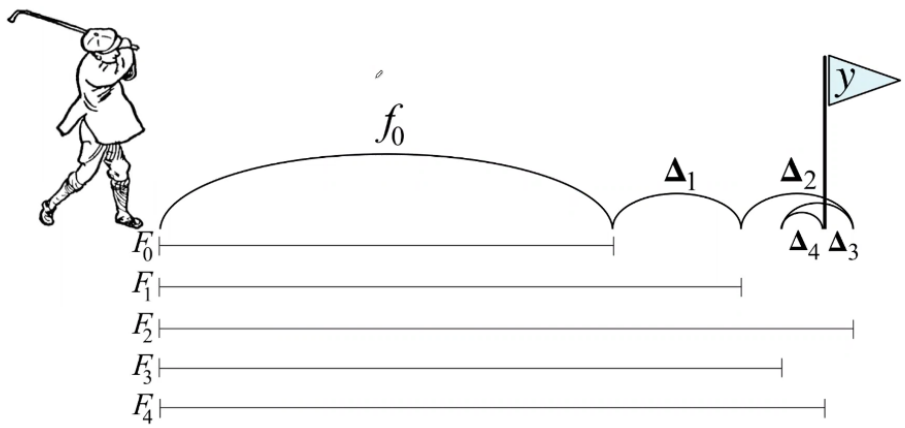


Golf görseli artık model kavramı için harika bir örnektir.

F0 ilk modeldir. Toplamda 4 artık model olduğunu F4'den anlayabiliyoruz. Fn adet model olabilir.

Delta1 artıkların üzerine kurulan ilk modeldir.

f0 base learner olarak düşünebiliriz.



Additive modellerin matematiksel formulü:

Y_ŞAPKA'yı elde etmek için tek bir tahminsel formu olan modeller serisi oluşturulur (1.satır)

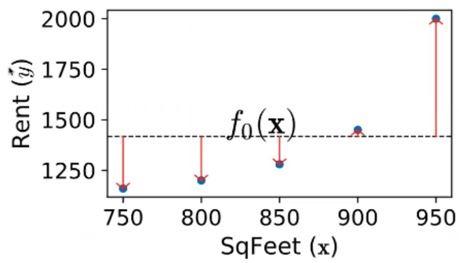
Yani; Y_ŞAPKA = İLK MODEL + ARTIK MODELLER TOPLAMI

$$
\begin{aligned}
\hat{y} &= f_0(\mathbf{x}) + \Delta_1(\mathbf{x}) + \Delta_2(\mathbf{x}) + \ldots + \Delta_M(\mathbf{x}) \\
&= f_0(\mathbf{x}) + \sum_{m=1}^{M} \Delta_m(\mathbf{x}) \\
&= F_M(\mathbf{x})
\end{aligned}
$$

$$
\begin{aligned}
F_0(\mathbf{x}) &= f_0(\mathbf{x}) \\
F_m(\mathbf{x}) &= F_{m-1}(\mathbf{x}) + \Delta_m(\mathbf{x})
\end{aligned}
$$

F0 aslında Y_ŞAPKA'dır olarak düşünebiliriz. y-F0'ları yeni bağımlı değişken olarak kabul ederek yeni modelleri (artık) oluşturacağız.

sqfeet	rent	F_0	$y - F_0$
750	1160	1418	-258
800	1200	1418	-218
850	1280	1418	-138
900	1450	1418	32
950	2000	1418	582



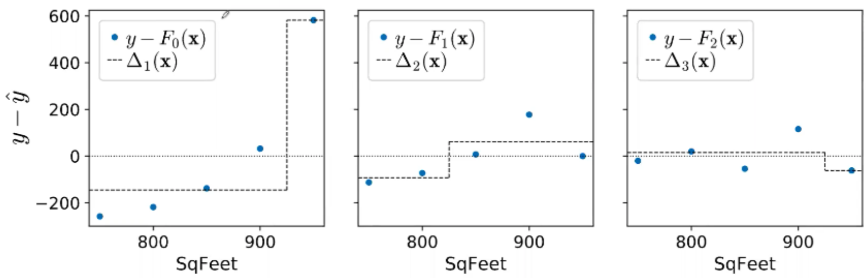
Aşağıdaki görsel için adım adım gitmeye çalışalım:

- 1. İterasyonda
  - Bağımsız değişken sqfeet
  - Bağımlı değişken rent (Y0)
  - Y_ŞAPKA0 (yeni tahmin edilen değer) F0'dır
  - Bunları modelleyerek hatamız = y - F0
- 2. İterasyonda
  - Bağımsız değişken sqfeet
  - Bağımlı değişken y - F0 (Y1)
  - Yeni bir model kurulur sqfeet ile Y1 arasında
    - Bu model kurulurken bağımsız değişken aynı kalır; sqfeet fakat bağımlı değişken y - F0 olur
    - Aslında; Delta1, hataların (y - F0) modellenmesinden elde edilen sonuçtur. Bu Delta1 değerini bir önceki iterasyonda elde edilen Y_ŞAPKA (tahmin edilen) değere ekleyerek yeni bir Y_ŞAPKA değeri elde edeceğiz.
  - Y_ŞAPKA1 (yeni tahmin edilen) ise F1 olur

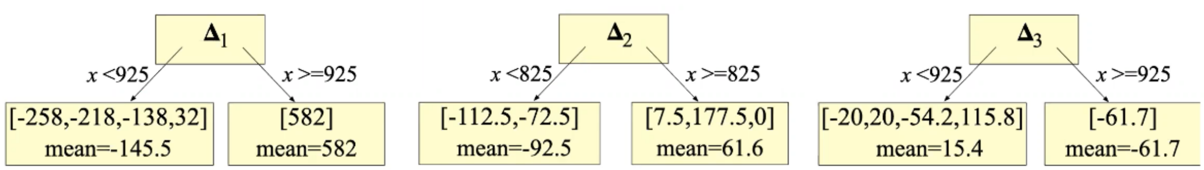
        - F0 + Delta1
    - Bu iterasyonda yeni hatamız y (rent) - F1 olur
- 3. İterasyonda
    - Bağımsız değişken sqfeet
    - Bağımlı değişken y - F1 (Y2)
    - Yeni bir model kurulur sqfeet ile Y2 arasında
        - Delta2 bulmak için
    - Y_ŞAPKA2 (yeni tahmin edilen değer) F2 olur
        - F1 + Delta2
    - Bu iterasyonda yeni hatamız y (rent) - F2 olur

sqfeet	**rent**	F_0	$\mathbf{y} - F_0$	Δ_1	F_1	**y**-F_1	Δ_2	F_2	**y** - F_2	Δ_3	F_3
750	1160	1418	-258	-145.5	1272.5	-112.5	-92.5	1180	-20	15.4	1195.4
800	1200	1418	-218	-145.5	1272.5	-72.5	-92.5	1180	20	15.4	1195.4
850	1280	1418	-138	-145.5	1272.5	7.5	61.7	1334.2	-54.2	15.4	1349.6
900	1450	1418	32	-145.5	1272.5	177.5	61.7	1334.2	115.8	15.4	1349.6
950	2000	1418	582	582	2000	0	61.7	2061.7	-61.7	-61.7	2000

sqfeet	rent	F_0	$\mathbf{y} - F_0$	Δ_1	F_1	y-F_1	Δ_2	F_2	y - F_2	Δ_3	F_3
750	1160	1418	-258	-145.5	1272.5	-112.5	-92.5	1180	-20	15.4	1195.4
800	1200	1418	-218	-145.5	1272.5	-72.5	-92.5	1180	20	15.4	1195.4
850	1280	1418	-138	-145.5	1272.5	7.5	61.7	1334.2	-54.2	15.4	1349.6
900	1450	1418	32	-145.5	1272.5	177.5	61.7	1334.2	115.8	15.4	1349.6
950	2000	1418	582	582	2000	0	61.7	2061.7	-61.7	-61.7	2000



sqfeet	rent	F_0	$\mathbf{y} - F_0$	Δ_1	F_1	y-F_1	Δ_2	F_2	y - F_2	Δ_3	F_3
750	1160	1418	-258	-145.5	1272.5	-112.5	-92.5	1180	-20	15.4	1195.4
800	1200	1418	-218	-145.5	1272.5	-72.5	-92.5	1180	20	15.4	1195.4
850	1280	1418	-138	-145.5	1272.5	7.5	61.7	1334.2	-54.2	15.4	1349.6
900	1450	1418	32	-145.5	1272.5	177.5	61.7	1334.2	115.8	15.4	1349.6
950	2000	1418	582	582	2000	0	61.7	2061.7	-61.7	-61.7	2000



XGBoost (eXtreme Gradient Boosting)

XGBoost, GBM'in hız ve tahmin performansını arttırmak üzere optimize edilmiş, ölçeklenebilir ve farklı platformlara entegre edilebilir versiyonudur.

LightGBM

LightGBM, XGBoost'un eğitim süresi performansını arttırmaya yönelik geliştirilen bir diğer GBM türüdür.

Level-wise büyüme stratejisi yerine Leaf-wise büyüme stratejisi ile daha hızlıdır.

CatBoost

Kategorik değişkenler ile otomatik olarak mücadele edilebilen, hızlı, başarılı bir diğer GBM türevi.