Lineer Regresyon ve scikit-learn

Herkese tekrardan merhabalar. ML101 yazı serimizin ikinci yazısı ile karşınızdayım. Bu yazıda lineer regresyon (yazının ilerleyen kısmında bundan LR olarak bahsedilecektir) algoritmasının scikit-learn kütüphanesi ile uygulamasını yapacağız. scikit-learn, makine öğrenimi algoritmalarını içeren kullanımı ücretsiz olan Python dilinde yazılmış bir kütüphanedir. Şimdi LR algoritmasının tanımını yapıp scikit-learn ile uygulamasına geçelim.

LR, formülü (**y = ax + b**) benzeri bir doğru formülü için veri setinde bulunan tüm noktaların hatalarının kareleri toplamını (bir önceki yazıdan da hatırlayacağınız üzere, ters işaretli hatalar birbirini sıfırlamasın diye kareler toplamı alınmaktadır) en küçükleyen **a** ve **b** katsayılarını bulmaya çalışan algoritmadır. LR algoritması; 4 parametre, 4 öznitelik ve 5 metot ile çalışmaktadır. Şimdi parametrelerini inceleyerek uygulamanın detaylarına geçelim. Yazının ilerleyen bölümlerinde, ayrı ayrı başlıklar altında tüm bunlar detaylı bir şekilde açıklanacaktır. Yazının sonunda ise örnek uygulamanın yapıldığı bir notebook sizlerle paylaşılacaktır.

Parametreler

- *fit_intercept:*

Bu parametre **boolean** değerler alarak çalışmaktadır. Parametrenin adındaki **intercept**, doğru denklemimizdeki sabit sayıyı yani **b** değerini kastetmektedir. Eğer bu parametrenin değerini **False** atarsanız, **b** değerine 0 atanır ve doğru denkleminiz (**y = ax**) şeklinde olur. **True** atanması durumunda ise toplam hatayı en küçükleyecek şekilde (**y = ax + b**) doğrusunu oluşturmaya çalışır.

- *normalize:*

Bu parametre, veri setinde bulunan girdi değerlerini yani X değerlerimizi normalleştirmeye yarar. Nasıl çalıştığına geçmeden önce normalleştirme nedir ve neden kullanılır bunu inceleyelim. Diyelim ki, veri setinizde aşağıdakine benzer iki tane sütun olsun.

Yaş	Maaş
25	7500
34	16876
15	500
28	56892
60	2400

İlk sütunumuzda bulunan yaş değeri (15,60) aralığında değişirken, ikinci sütunumuzdaki aylık gelir değeri (500, 56892) aralığında değişmektedir. Yaş sütununun ortalaması 32,4; aylık gelir sütunun ortalaması 16833,6 olarak karşımıza çıkmaktadır. Yani aylık gelir sütununun ortalaması yaş sütununun ortalamasının 519,5 katıdır. Ancak bu fark aylık gelir sütunundaki

değerlerin, yaş sütunundaki değerlerden 519,5 kat daha değerli olduğu anlamına gelmemektedir. Sütunlar arasındaki oluşan bu gibi farkları ortadan kaldırmak için normalleştirmeden faydalanılır. Normalleştirme, değerlerinizi (-1, 1) veya (0, 1) gibi alt aralıklara indirgeyerek değerler arası farklılığın etkisini yok etmeyi amaçlamaktadır. Bu işlemi yapmanın farklı yolları vardır ve biz bu farklı yolları farklı algoritmalarda göreceğiz. Ama şimdi LR nasıl normalleştirme yapıyor bunu inceleyelim.

LR, normalleştirme yapmak için iki değer kullanır. Bu değerler **ortalama** ve **L2-norm** değerleridir. Bunlardan ilki olan ortalama büyük ihtimalle çoktan duymuşsunuzdur ama yine de tanımını yapacak olursak ortalama (veya bir sütunun ortalaması), veri setinde bulunan noktaların (veya sütunda bulunan noktaların) değerleri toplamının nokta sayısına bölünmesi ile bulunur. L2-norm yani **Length2-norm** ise ilgili noktaların '**Öklid uzunluğu**' değerini temsil eder. Şimdi LR bu değerleri nasıl hesaplayıp nasıl kullanarak normalleştirme yapar inceleyelim. İlk olarak küçük bir veri seti oluşturalım.

$$\boldsymbol{V_0 = [1,2,3,4,5]}$$

Şimdi bu veri setinin ortalamasını hesaplayarak başlayalım. Ortalama hesaplamanın formülü,

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

şeklindedir. Buna göre örnek veri seti için $\bar{x}$ değerimiz 3 olarak bulunur. Şimdi bu değeri veri setinde bulunan tüm noktalardan çıkaralım ve aşağıda bulunan yeni vektörü elde edelim.

$$\boldsymbol{V_1 = [-2,-1,0,1,2]}$$

Normalleştirmenin son aşamasında ise L2-norm değerini bulup yukarıdaki vektörde bulunan tüm değerleri L2-norm'a bölmek. L2-norm yani Öklid uzunluğunu, noktaların sıfıra olan uzaklıklarının kareleri toplamının karekökü olarak nitelendirebiliriz. Formülü ise aşağıdaki gibidir.

$$L_2 = \sqrt[2]{\sum_{i=1}^{n} |x_i|^2}$$

Noktaların L2-norm değeri $\sqrt{55}$ yani yaklaşık olarak 7,42 olarak bulunur. Bu değer hesaplanırken V_0 yani orijinal vektörümüzde bulunan değerler kullanılır. Son vektörümüzü yani V_1 vektörünü bu değere bölersek normalleştirilmiş vektörümüz yani V_n vektörü aşağıdaki gibi olmaktadır.

$$\boldsymbol{V_n = [-0.2695,-0.1347, 0,0.1347,0.2695]}$$

Bu parametre yani **normalise** parametresi, aynı **fit_intercept** parametresi gibi **boolean** değerler ile çalışmaktadır. Ancak kullanımında dikkat edilmesi gereken bir nokta vardır. Eğer **fit_intercept** parametresinin değeri **False** atanırsa, **normalise** parametresi çalışmaz.

- *copy_x:*

Bu parametre **boolean** değerler ile çalışan ve **True** olması durumunda algoritmaya girdi olarak verdiğimiz X değerlerinin bir kopyasını almaya yarayan parametredir. Bu şu anlama gelmektedir. Sizler modelin öğrenmesi için kendisine birazdan öğreneceğimiz **fit** metodunu kullanarak girdi verilerini vereceksiniz. Model ise bu parametrenin değerinin **True** olması durumunda verdiğiniz X değerlerini kullanmak yerine, o değerlerin bir kopyasını alır ve işlemleri kopya üzerine uygular. Yani orijinal verilerinizde bir değişim olmaz.

- *n_jobs:*

Bu parametre ise bilgisayarınızın veya çalıştığınız ortamın işlem gücünü ayarlamanızı sağlar ve tam sayılı değerler ile çalışır. Diyelim ki elinizde 4 çekirdekli bir işlemci var. Eğer **n_jobs = 1** atarsanız yapılan hesaplamalar için işlemcinizin sadece 1 çekirdeği kullanılırken, **n_jobs = -1** atamanız durumunda ise 4 çekirdek de aktif olarak kullanılır. Veri setinizdeki nokta sayısına bağlı olarak bu parametreye atadığınız farklı değerler beklediğiniz iyileşmeyi sağlayamayabilir. Örneğin küçük bir veri seti için işlemler zaten milisaniyeler boyutunda gerçekleştiği için değişiklik de yeteri kadar etkili olmayabilir. Bu parametremiz ile birlikte LR algoritmasının parametreleri bitti. Sırada özniteliklerini incelemek var.

Öznitelikler

- *coef_ :*

İlk özniteliğimiz **coef_** özniteliğidir. LR algoritmamız çalıştıktan sonra hatırlayacağınız üzere (y = ax + b) şeklinde bir doğru çiziliyordu. Bu doğrunun **a** ve **b** katsayılarının ne olduğunu görmek için bu özniteliği çağırıyoruz. Çıktı olarak bizlere katsayıları içeren bir satır vektörü vermektedir. Unutmayalım ki **fit_intercept** parametresinin değerini **False** olarak atarsak çıktı vektöründe yalnızca **a** katsayısı görülecektir.

- *rank_ :*

Bu öznitelik ise lineer cebirden bildiğimiz rank değerini bulmaya yarar. LR algoritmasına girdi olarak verdiğimiz X veri setinin rank değerinin kaç olduğunu döndürür. Ben bu yazıda sizlerin matrisin rank değeri nedir ve nasıl bulunur konularını bilmediğinizi varsayarak adım adım anlatacağım. En temel anlamı ile rank, bir matristeki lineer bağımsız sütun sayısıdır. Peki lineer bağımsız veya lineer bağımlı sütun ne demektir şimdi gelin bunu temel bir örnek ile anlayalım.

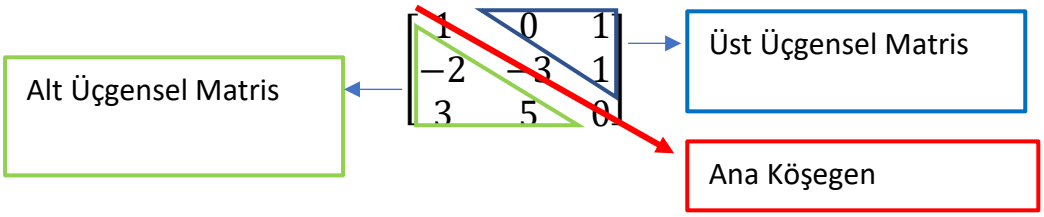
$$\begin{bmatrix} 1 & 0 & 1 \\ -2 & -3 & 1 \\ 3 & 3 & 0 \end{bmatrix}$$

Elimizde şu anda 3x3 boyutunda bir matrisimiz var. Bir matristeki **lineer bağımsız sütun** demek, diğer sütun veya sütunlar kullanılarak hesaplanamayan sütun demektir. Matrisimizdeki sütunları C_1, C_2 ve C_3; satırları ise R_1, R_2 ve R_3 olarak ifade ettiğimizi düşünün.

Bu durumda C_3 sütunu aslında C_2 - C_1 işlemi ile elde edilebiliyor. Yani C_3 sütunumuz lineer bağımlı bir sütun. Aynı şekilde R_3 ise R_1 - R_2 işlemi ile elde edilebiliyor. Matrisimizdeki en az 1 sütun lineer bağımlı ve elimizdeki matriste de 3 sütun olduğundan artık bu matriste en fazla 2 lineer bağımsız sütun olabilir (C_1 ve C_2). Bu durumda da rank değeri maksimum 2 olabilir. Ancak henüz hesabımız bitmedi yani hâlâ rank değerine ulaşamadık. Bu durumda hesabımıza lineer bağımlı olan satır ve sütunu silip devam edeceğiz. Yani C_3 ve R_3 silinecek. Elde ettiğimiz yeni matris ise aşağıdaki gibi olacaktır.

$$\begin{bmatrix} 1 & 0 \\ -2 & -3 \end{bmatrix}$$

Bir önceki adımda yaptıklarımızın aynısını yaparak bu matriste kaç adet lineer bağımsız sütun olduğunu bulmaya çalışalım. Elimizde artık yalnızca C_1 ve C_2 sütunları kaldı. Ne C_1 sütununu ne de C_2 sütununu bir diğerinin katı şeklinde ifade edemiyoruz. Bu yüzden de bu sütunlarımız aradığımız lineer bağımsız sütunlar olarak karşımıza çıktı. Matrisimizde 2 adet lineer bağımsız sütun olduğu için rank değerimiz de 2 olmaktadır. Şimdi rank bulma işleminin daha sistematik yolu olan **Gauss Eliminasyon** yöntemi ile rank hesaplamayı öğrenelim.



Gauss eliminasyon yöntemine başlamadan önce ne olduğunu öğrenmemiz gereken birkaç kavram var. Bu kavramlardan ilki **ana köşegen** yani İngilizcesi ile **main diagonal**. Bu köşegen kare bir matrisin sol üst köşesini sağ alt köşesine bağlayan ve (0,0), (1,1) veya (2,2) gibi elemanları bir kapsayan doğrudur. Siz bir kare matriste bu doğruyu çizerseniz matrisi aslında ortadan ikiye bölmüş olursunuz. Matrisin köşegenin altında kalan kısmı **alt üçgensel matris** olarak adlandırılırken, üstünde kalan kısım ise **üst üçgensel matris** olarak adlandırılır. Gauss eliminasyon yöntemi ise **temel satır işlemlerini** (**elementary row operations**) kullanarak bu matrisin **eşdeğer formunu** (**echelon form**) bulmayı amaçlamaktadır. İlk kez duyduğunuzu varsaydığım temel satır işlemleri ise, bir matrise uygulandığında onun karakteristiğini değiştirmeyen üç adet işleme verdiğimiz isimdir. Bu üç işlem ise:

- İki satırın yerini değiştirmek.
- Herhangi bir satırı sıfırdan farklı bir sayı ile çarpmak.
- Herhangi bir satırın bir katını başka bir satıra eklemek.

şeklinde sıralanabilir. Şimdi örnekteki matrisimize Gauss Eliminasyon uygulayarak hem adım adım eşdeğer forma indirgemeyi görelim hem de bu konu hakkında öğrenmemiz gereken birkaç ek terimi daha uygulamalı olarak öğrenelim. Diyelim ki elinizde aşağıdakine benzer bir denklem sistemi olsun.

$$2x + 2y = 6$$

$$3x - 5y = 6$$

Bu denklem sistemini çözmek için yapacağınız işlemler aşağıdakine benzer olacaktır:

- İlk satırı -3/2 ile çarpmak (Temel Satır İşlemi-2).
- Birinci ve ikinci satırı toplamak (Temel Satır İşlemi-3).
- Elde kalan -8y = -3 denklemini çözmek için her iki tarafı -8 ile bölmek (Temel Satır İşlemi-2).
- y = 3/8 değerini herhangi bir denklemde yerine koyup x değerini 21/8 olarak bulmak.

Sonuca baktığımızda aslında elde ettiğimiz denklemler;

$$1x + 0y = 21/8$$

$$0x + 1y = 3/8$$

denklemleridir. Her iki denklemi de aynı değerler sağlamaktadır. Yani aslında ikinci denklem kümesi ile temel satır işlemlerini uyguladığımız ana denklemler **eşdeğer formdadır**. Bir denklemin birden fazla eşdeğer formu olabilir. Ama Gauss eliminasyon yöntemi denklem sistemini yukarıdaki basit forma kadar indirgemeyi amaçlamaktadır. Yöntemin nasıl çalıştığını ve bazı temel kavramları temel bir örnek üzerinden gördük. Şimdi bu kavramlara yenilerini ekleyerek gauss eliminasyon yardımı ile rank bulmayı görelim.

$$\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$$

Yukarıdaki matris bizim rank değerini bulmaya çalıştığımız bir matris olsun. Bu matrisin satırlarını ilk satırı R_1 , ikinci satırı R_2 ve son satırı da R_3 olacak şekilde adlandıralım. Adımlara başlarken gauss eliminasyon yönteminde kullanılan işlem gösterimini de öğrenmiş olalım. Bu gösterim ile ilgili detaylı açıklama sadece ilk adımda yapılacaktır. Sonraki adımlarda yalnızca işlemler ve sonuçlar gösterilecektir.

- $R_2 \rightarrow 2r_1 + r_2$

Mavi kutu ile işaretlediğimiz kısım, işlemin yapıldığı satırı temsil etmektedir. Yeşil ile işaretlediğimiz kısım ise ilgili satıra yapılan işlemdir. Yani bu örnekte, birinci satırın iki katı ile ikinci satırdaki değerler toplanıp ikinci satıra yazılmıştır. Elde edilen matris ise aşağıda gösterilmiştir.

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 3 \\ 3 & 5 & 0 \end{bmatrix}$$

- R_3 = -3r_1 + r_3

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 3 \\ 0 & -1 & -3 \end{bmatrix}$$

- R_3 = r_2 + r_3

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 0 \end{bmatrix}$$

İşlemlerimize bu noktada bir ara veriyoruz. Çünkü artık başlangıçtaki matrisimizin **rank** değerini bulabiliyoruz. Bir matrisin rank değerinin iki farklı tanımı var. İlk tanıma matris rank'ı; temel satır işlemleri uygulandıktan sonra elde edilen satır eşdeğer matristeki elemanlarının tamamı sıfırdan farklı olan satırların sayısıdır. Bizim örneğimizde yalnızca son satırımızda sıfırlar olduğu için ilgili matrisin rank değeri 2'dir. İkinci bir tanıma göreyse rank, **pivot** eleman sayısına eşittir. Peki pivot eleman ne demektir?

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 0 \end{bmatrix}$$

Yukarıdaki matriste bazı değerler kırmızı ile gösterilmiştir. Bu değerler bizim **pivot** değerlerimizdir. Pivot, bir satırda sıfırdan farklı olan ve en solda bulunan eleman demektir. Pivot elemanlar aynı zamanda **lider katsayı** olarak da adlandırılır. Bir matristeki pivot elemanları bulmak için matris öncelikle satır eşdeğer formuna veya **indirgenmiş satır eşdeğer formuna** ulaşılıncaya kadar çözülmelidir. Peki bu yeni duyduğumuz indirgenmiş satır eşdeğer formu nedir? Bir önceki halde bulunan matrise uygulayabileceğimiz bir temel satır işlemi daha var ve o da aşağıdaki işlem.

- R_1 = -2r_2+ r_1

$$\begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & 3 \\ 0 & 0 & 0 \end{bmatrix}$$

Yukarıda bildirdiğimiz temel satır işlemini yaptığımızda gördüğünüz gibi ikinci pivot değerimizin üstündeki değer yani [1,2] hücresindeki değer de artık 0 oldu. Eğer bir matrisin tüm lider katsayıları (yani pivot değerleri) 1'e eşitse ve lider katsayının bulunduğu sütundaki diğer elemanlar sıfıra eşitse, bu matrise indirgenmiş satır eşdeğer formu adı verilir. Matrisin rank değeri her iki durumda da değişmez. LR algoritmamızın "rank_" özniteliği sizleri tüm bu zahmetli süreçten kurtarıp, girdi olarak verdiğiniz X matrisinin kaçıncı derece olduğunu yani rank değerini döndürüyor.

- *singular_*

Bu özniteliğimiz ise yine lineer cebirde karşımıza çıkan **tekil değer ayrışımı** (singular value decomposition) yönteminden faydalanarak X girdi matrisinin tekil değerini/değerlerini azalan sıralamada döndürmektedir. Eğer tekli değer ayrışımının ne olduğunu biliyorsanız zaten bu özniteliğin nasıl çalıştığını da anlamışsınızdır. Ama eğer bu yöntemi bilmiyorsanız veya hatırlamak istiyorsanız sizleri şu yazıya alalım.

Tekil değer ayrışımı nasıl yapılır konusuna geçmeden önce ne olduğuna değinelim. Tekil değer ayrışımı (**singular value decomposition**); elinizde bulunan ve çok sayıda eleman içeren bir veri setinin, daha az değer içeren ama ana veri setinin güvenilirliği hakkında sizlere bilgi veren daha küçük bir matrise dönüştürülmesi sürecine verilen addır. Elde edilen bu matrisin elemanlarına **tekil değerler** denmektedir. Şimdi bu ayrışımın nasıl yapıldığını öğrenelim. Elimizde aşağıdaki gibi bir **C** matrisi olsun.

$$\begin{bmatrix} 5 & 5 \\ -1 & 7 \end{bmatrix}$$

Bir matrisin tekil değerlerini bulabilmek için, o matrisin öz değerlerinden (**eigen values**) faydalanılır. Öz değer ayrışımı uygulanıp bulunan değerlerin karekökleri, **Σ** ile ifade ettiğimiz tekil değerler matrisinin ana köşegeni üstünde bulunan değerlerdir. Öz değerler aşağıdaki denklemi sağlayan değerler kümesidir.

$$\det |C^T C - \lambda * I| = 0$$

Bu denklemi çözdüğümüzde elde edeceğimiz değerler kümesi matrisin öz değerleri olacaktır. Bu sistemi çözmek için ilk olarak $C^T C$ çarpımını bulalım.

$$\begin{bmatrix}5 & -1\\ 5 & 7\end{bmatrix} * \begin{bmatrix}5 & 5\\ -1 & 7\end{bmatrix} = \begin{bmatrix}26 & 18\\ 18 & 74\end{bmatrix}$$

Bu matristen, $\lambda * I$ matrisini çıkarıp determinantını alacağız. Elde edeceğimiz matris aşağıdaki gibi olacaktır.

$$\begin{bmatrix}26-\lambda & 18\\ 18 & 74-\lambda\end{bmatrix}$$

Şimdi λ değerlerini bulmak için bu matrisin determinantını alıp sıfıra eşitleyeceğiz. Bulduğumuz λ değerleri tekil değerler matrisimizin ana köşegenini oluşturmak için kullanacağımız değerler olacak. Bildiğiniz gibi 2x2 bir matrisin determinantı, ana köşegen üzerinde bulunan değerlerin çarpımı ile diğer değerlerimizin çarpımının farkına eşittir. Şimdi hem determinantı bulalım hem de elde edeceğimiz denklem sistemini çözerek öz değerleri bulalım.

$$[(26-\lambda)*(74-\lambda)] - [(18)*(18)] = 0$$

$$\lambda^2 - 100 * \lambda + 1600 = 0$$

$$(\lambda - 20) * (\lambda - 80) = 0$$

$$\lambda_1 = 20, \lambda_2 = 80$$

Şimdi bu değerleri kullanarak bulmayı amaçladığımız değerleri yani tekil değerleri bulalım. Bunun için Σ matrisini oluşturacağız. Bu matris; ana köşegeninde yukarıda bulduğumuz λ değerlerinin karekökleri olan, alt ve üst üçgensel matrislerinde ise 0 olan bir matris olacak.

$$\Sigma = \begin{bmatrix}2\sqrt{5} & 0\\ 0 & 4\sqrt{5}\end{bmatrix}$$

Bu matristeki $4\sqrt{5}$ ve $2\sqrt{5}$ değerleri matrisin tekil değerleridir (**singular values**). Bir matrisin tekil değer sayısı, matrisin **rank** değerine eşittir. Bu örneğimizdeki matrisin rank değeri 2'dir bu yüzden de 2 tane tekil değer elde edilmiştir.

- *intercept_ :*

Bu öznitelik, **y = a*x + b** doğru denklemindeki sabit sayımız olan **b** değerinin kaç olduğunu döndürür. Eğer ***fit_intercept*** parametresinin değerini False olarak atarsanız, bu öznitelik çıktı olarak sıfır döndürür.

Metotlar

- *fit:*

sklearn kütüphanesinde bulunan neredeyse tüm makine öğrenimi algoritmalarını eğitmek için kullandığımız metot **fit** metodudur. Bu metot çalışmak için sizden 3 farklı parametre beklemektedir. Bu parametrelerden ilki X parametresidir. Eğitim için kullanacağınız veri setindeki X değerlerini bu parametreye atayarak metoda gönderirsiniz. İkinci parametremiz ise **y** değerleridir. Metot sizden, eğitim için kullanacağı X değerlerinin her birisine karşılık gelen **y** değerlerini beklemektedir. Son parametremiz ise **sample_weight** parametresidir. Şimdi bu parametrenin kullanımını inceleyelim. Diyelim ki elinizde aşağıdaki gibi bir veri seti olsun.

X	Y
3	5
7	6
8	5
9	6
2	6

Veri setinizde bulunan y = 6 değerlerinin sayısı daha fazla bu yüzden de sizler algoritma tarafından o değerin daha önemli olduğunun algılanmasını isteyebilirsiniz. Bunun için **sample_weight** parametresinden faydalanıyoruz. Algoritmanın öğrenirken hangi satırları diğerlerine oranla daha önemli kabul etmesini istiyorsak, o satıra karşılık gelen değerler için **sample_weight** değerlerini daha yüksek seçiyoruz. Örneğin bu durumda y = 6 değerlerinin daha önemli olması için aşağıdakine benzer bir girdi sağlayabiliriz.

sample_weight = [2,4,2,4,4]

Bu kullanım ile siz çıktı değeri 5 olan örneklerin ağırlığını 2 olarak belirlerken, çıktı değeri 6 olan örneklerin ağırlığını ise 4 olarak belirlemiş oldunuz. Bu parametreye değer atamak zorunlu bir durum değildir. Eğer bu parametreyi değer atamadan kullanırsanız algoritma, tüm çıktı değerlerinin ağırlığını eşit kabul edecektir. Peki biz **fit** metodunu çağırıp çalıştırdığımızda bir lineer regresyon doğrusu nasıl çiziliyor? Şimdi de o doğrunun oluşumunun arkasında yatan matematik temellerini inceleyelim ve sample_weight parametresinde kullandığımız veri noktaları için bir lineer regresyon doğrusunu kendimiz oluşturalım. Hatırlayacağınız üzere doğru denklemimiz y = a * x + b şeklindeydi. Bu denklem oluşturulurken LR algoritması ilk olarak **a** katsayısını hesaplar. Bu kat sayı ise aşağıdaki formül ile hesaplanır.

$$a = \frac{\sum[(x_i - \bar{x}) * (y_i - \bar{y})]}{\sum[((x_i - \bar{x})^2)]}, \forall i$$

Bu hesabı yapmak için gördüğünüz gibi x ve y değerlerinin ortalamasına ihtiyacımız var. Tablomuzda bulunan x değerlerinin ortalaması 5.8, y değerlerinin ortalaması ise 5.6 olmaktadır. Şimdi denklemimizde bulunan diğer değerleri bulacağız ancak bunun için Nümerik Analizde sıklıkla kullanılan bir tablo yapısından faydalanacağız.

x_i	y_i	$x_i - \bar{x}$	$y_i - \bar{y}$	$(x_i - \bar{x}) * (y_i - \bar{y})$
3	5	-2.8	-0.6	1.68
7	6	1.2	0.4	0.48
8	5	2.2	-0.6	-1.32
9	6	3.2	0.4	1.28
2	6	-3.8	0.4	-1.52

Denklemin payını bulmak için en sağdaki kırmızı ile renklendirdiğimiz sütundaki değerleri topluyoruz. Bu durumda denklemimizin payı (0.6) olarak bulunur. Paydayı bulmak için de yeşil sütunu kullanıyoruz. Bu sütundaki değerlerin karelerini topladığımızda ise paydayı (38.8) olarak bulacağız. Artık yapmamız gerek tek şey, bu iki değeri bölerek LR doğrumuzun "**a**" katsayısı olan (0.01546391) değerini bulmak. Doğruyu oluşturmak için sırada **b** değerimiz var. Bu değeri bulmak için de **a** katsayısı kullanılır. Eğer LR modelinizi oluştururken ***fit_intercept*** parametresinin değerini **False** olarak atarsanız **b** sabiti hesaplanmaz onun yerine sıfır olarak atanır, **True** olarak atarsanız da bu sabit değer aşağıdaki formül kullanılarak hesaplanır.

$$b = \bar{y} - a * \bar{x}$$

Yukarıda bulduğumuz $\bar{y}$, $\bar{x}$ ve a değerlerini denklemde yerine koyduğumuzda b sabitimiz (5.510309322) olarak bulunur. Buna göre LR doğrusunun denklemi

$$\hat{y} = 0.01546391 * x + 5.510309322$$

olarak bulunur. Burada gördüğünüz gibi doğru denklemini yazarken *y* yerine $\hat{y}$ yazdık. Çünkü bu denklem ile bulacağımız değerler bizim tahminlerimiz olacak ve tahmin değerleri genellikle $\hat{y}$ ile ifade edilir. Bir önceki yazımızda, oluşturulan lineer regresyon doğrusunun bir hatası olduğunu ve bu hatanın genelde hata karelerinin toplamı yoluyla bulunduğundan bahsetmiştik. Şimdi aşağıdaki tablodan faydalanarak doğrumuzdaki her bir noktanın hatasını ve doğrunun toplam hatasını bulalım.

x_i	y_i	$\hat{y}$	$e_i^2 = (y_i - \hat{y})^2$
3	5	5,55670105	0,30991606
7	6	5,61855669	0,145499
8	5	5,6340206	0,40198212
9	6	5,64948451	0,12286111
2	6	5,54123714	0,21046336

Tablonun en sağ sütununda, veri setindeki tüm noktalar için hata değerlerini görüyorsunuz. Hata genellikle **ε** sembolü ile ifade edilir. LR doğrusunun hatasını hesaplamak için, her bir noktanın gerçek değeri ile tahmin değeri arasındaki farkın karesini aldık. Doğrunun toplam hatası (1.19072165) olarak bulunur. Bu değeri bulmak için **hata karelerinin toplamını** aldık.

Bir modelin hatası değerlendirilirken, hata ne kadar düşükse model o kadar başarılıdır diyebiliriz. Ancak buradaki düşük hata ile kastettiğimiz yalnızca hatanın değerinin düşük olması demek değildir. Örneğin iki farklı LR modelini 10 veri içeren fakat değerleri farklı olan iki veri seti ile eğittiğimizi düşünelim. Birinci veri setindeki y değerleri (0, 10) aralığında, ikincide ki y değerleri ise (15000, 20000) aralığında değişsin. Her iki modelin hatası da 100 olarak çıkmış olsun. Bu durumda sizin her nokta için ortalama 10 birimlik bir hatanız vardır diyebiliriz. Değerleri (0, 10) aralığında değişen bir kümede her noktada 10 birim hata yapmak demek modelin olağanüstü derecede kötü çalıştığını gösterirken, (15000, 20000) aralığında bu kadar düşük bir hata olması da modelin neredeyse tüm verileri nokta atış ile tahmin ettiğini yani bir ezberleme/aşırı uyum (**overfitting**) durumu söz konusu olabileceğini göstermekte. Bu yüzden bir genelleme yapamamakla birlikte hata ne kadar düşük olursa model de o kadar başarılıdır diyebiliriz.

- *get_params:*

Bu metot ise, oluşturduğunuz LR modelinin çalıştığı parametrelerin ve bu parametrelerin değerlerinin ne olduğunun çıktısını verir. Çalışmak için bir argümana ihtiyaç duymamaktadır.

- *predict:*

Üçüncü metodumuz olan **predict** metodu çağırılmadan önce model eğitilmelidir. Yani öncesinde **fit** metodu çalıştırılarak LR doğrusunun denkleminin oluşturulması gerekir. predict metodu; eğitilen bir modelin daha önce görmediği verileri kullanarak tahmin yapmasını sağlar. Çalışmak için tek parametreye ihtiyaç duyar o da tahmin yapmak için kullanacağı x değerlerini içeren veri setidir. Yani **fit** metodu ile oluşturduğumuz doğru denklemindeki x değeri yerine girdi olarak aldığı x değerlerini yazarak tahmin kümesini yani $\hat{y}$ değerlerini döndürür.

- *score:*

Bir LR modelinin başarısını ölçmemizi sağlayan iki değer vardır. Bunlardan birincisi modelin hatası ikincisi ise modelin R^2skorudur. Bu skor hesaplanırken modeli eğittiğimiz değerler değil, modeli test etmek için kullandığımız değerler kullanılır. Çalışma parametreleri aynı **fit** metodunun parametreleri gibidir. Yani tahmin yapmak için kullanacağı X değerleri, tahmin sonuçlarını değerlendirmek için kullanacağı y değerleri ve varsa sample_weight parametresi için örneklem ağırlıkları. Algoritma bu değerleri kullanarak R^2 skorunu hesaplamak için aşağıdaki formülden faydalanır.

$$R^2 = 1 - \frac{\sum {\varepsilon_i}^2}{\sum (y_i - \bar{y})^2}$$

Burada yine uygulama olması için örnek bir veri setinde R^2 skorunu hesaplayalım. Kullanacağımız örnek veri setindeki y değerlerinin ortalaması 6'dır.

x_i	y_i	$\hat{y}$	${e_i}^2 = (y_i - \hat{y})^2$	$(y_i - \bar{y})^2$
4	5	5,57216496	0,32737274	1
6	7	5,60309278	1,95134978	1
9	6	5,64948451	0,12286111	0

Şimdi tablomuzdaki değerleri kullanarak R^2 denkleminin pay ve paydasını bulalım. Denklemin pay kısmında hata karelerinin toplamı yani (2.401583627) değeri yer alırken paydada (2) olacak. Bu durumda modelimizin R^2 skoru (-0.20079) olarak bulunacaktır. R^2 skorunun alabileceği en yüksek değer 1'dir ve bu durum da hatanın 0 olduğu duruma karşılık gelmektedir.

Buradaki örneğimizde R^2 değerimiz negatif geldi. Bu da karşılaşabileceğiniz özel durumlardan birisidir. Eğer R^2 skoru negatif gelirse model tahmin yapma aşamasında kötü çalışıyor demektir. Bu durumda ya yanlış parametreler ile çalışıyordur ya elinizdeki veri seti modelin sağlıklı bir şekilde çalışmasına imkân vermeyecek kadar küçüktür (bizim örneğimizde negatif gelme nedeni bundan dolayıdır) ya da modeliniz o veri seti ile çalışmaya uygun değildir. Eğer ilk durum söz konusu ise model parametrelerini değiştirip veri seti için en iyi olan parametre kombinasyonunu bulunuz. İkinci durum oluştuğunda veri sayısını artırıp modeli yeniden çalıştırınız. Eğer veri setiniz zaten elde edebileceğiniz son haliyse yani üçüncü durumdaysanız farklı bir makine öğrenimi modeli kullanmayı deneyin.

Karşınıza çıkabilecek diğer bir özel durum ise R^2 değerinin 0 olduğu durumdur. Bu durumda ise modeliniz girdi değerlerinden bağımsız olarak her seferinde y değerlerinin beklenen değerini yani y verilerinizin ortalamasını tahmin ediyor demektir. Bu durum ise sabit model (**constant model**) durumu olarak adlandırılır.

- *set_params:*

Diyelim ki modeli değerlendirme aşamasında R^2 skorunuz istediğiniz gibi gelmedi ve model parametrelerini değiştirerek yeniden modeli çalıştıracaksınız. İşte bu durumda **set_params** metodu sizleri modeli tekrar tekrar kurma zahmetinden kurtarıyor. Bu metot modeldeki değiştirmek istediğiniz parametreleri ve onların yeni değerlerini girdi olarak alıp, kurmuş olduğunuz modelde o değerleri değiştirmeyi sağlar. Yalnızca ve yalnızca sizin değiştirmeyi istediğiniz yani kendisine girdi olarak verdiğiniz parametrelerin değerini değiştirir. Diğer parametreler önceki hallerinde kalırlar.

LR algoritmasını anlattığımız yazı serisi burada sona eriyor. Bir sonraki yazı serisinde **Lojistik Regresyon** konusunu işleyeceğiz.

Tekrar görüşünceye dek sağlıcakla kalın…