



Makine Öğrenmesinin Matematiksel Yöntemleri

Hazırlayan: HASAN ÇELİK

Öğrenci No : 090180305

Teslim Tarihi: 11.06.2023

Danışman : PROF. DR. ELİF ÖZKARA CANFES

İçindekiler

1	Makine Öğrenmesi	5
1.1	Makine Öğrenmesi Nedir?	5
1.2	Sınıflandırma Problemi ve Algoritmaları	5
1.3	Lojistik Regresyon	6
1.4	Destek Vektör Makineleri(SVM)	8
1.4.1	Belirli bir hata ile SVM (Soft Margin)	10
1.4.2	Çekirdek Hilesi	11
1.5	K En Yakın Komşu Algoritması(KNN)	12
1.6	Karar Ağaçları	14
1.7	Yapay Sinir Ağları	19
1.7.1	Tek Katmanlı Yapay sinir Ağı	19
1.7.2	Çok Katmanlı Sinir Ağları	21
1.8	Evrişimli Sinir Ağları (CNN)	23
1.8.1	Evrişim Katmanı	24
1.8.2	Ortaklama (pooling) katmanı	25
1.8.3	Düzleştirme (flattening) katmanı	26
1.8.4	Tam bağlantı (Fully-Connected) katmanı	26
1.9	Yinelemeli Sinir Ağları(RNN)	27
1.10	Kayıp Fonksiyonunun Minimize Edilemsi	28
2	Makine Öğrenmesi Modellerini Geliştirecek Teknikler	31
2.1	Kategorik Değişken Dönüşümleri	31
2.1.1	Label Encoding	31
2.1.2	One-Hot Encoding	31
2.2	Özellik Ölçeklendirme ve Standartlaştırma	32
2.2.1	Maksimum - Minimum Dönüşümü	32
2.2.2	Veri Standartizasyonu	32
3	Makine Öğrenmesi Tekniklerinin Uygulanması	34
3.1	Deprem Binalar Üzerindeki Hasarının Tahmini	34
3.1.1	Label Encoding Metodu ile Ölçeklendirme Yapılmadan Oluşturulan Model	34
3.1.2	One hot encoding metodu ile ölçeklendirme yapılmadan oluşturulan model	36
3.1.3	One hot encoding ve standartlaştırma metodlarıyla oluşturulan model	36
3.1.4	Label encoding ve standartizasyon teknikleri ile oluşturulan model	37
3.2	Oluşturulan Modellerin Kullanılabilirliği	38

Şekil Listesi

1	Hiper düzlem ile sınıflandırma	6
2	sigmoid Fonksiyon	7
3	Destek vektör makineleri ile sınıflandırma	8
4	Destek vektörleri ve sınır aralığı	9
5	Destek vektör makineleri hata toleransı	10
6	Çekirdek hilesi	11
7	RBF çekirdek dönüşümü	12
8	K en yakın komşu algortması ile sınıflandırma	13
9	Kayak yapılabilme için örnek veri kümesi	14
10	Kayak yapılabilme örnek veri kümesi için karar sınırları	14
11	Karar ağacı	15
12	Karar ağaçları için uygulama veri kümesi	16
13	c1 sütun değerlerine göre ayrılma	17
14	c2 sütun değerlerine göre ayrılma	18
15	Yapay sinir ağları	19
16	Tek katmanlı yapay sinir ağı	20

17	Tek katman ve birden çok nöron ile yapay sinir ağı	21
18	Çok katmanlı yapay sinir ağı	21
19	Lineer aktivasyon fonksiyonu	22
20	ReLU aktivasyon fonksiyonu	22
21	Tanh aktivasyon fonksiyonu	23
22	RGB panelleri	23
23	Piksel değerleri	24
24	Evrişim işlemi	25
25	CNN dolgu işlemi	25
26	Ortaklama katmanı	26
27	Düzleştirme katmanı	26
28	Tam bağlantı katmanı	26
29	Yinelemeli sinir ağı	27
30	Bire çok RNN	28
31	Çoka bir RNN	28
32	Çoka çok RNN	28
33	Gradyan inişi algoritması	29
34	Gradyan inişi algoritmasında adım boyutu etkisi	29
35	İki parametrelili fonksiyonlarda gradyan inişi algoritması	30
36	Kategorik dönüşüm için örnek veri kümesi	31
37	label encoding	31
38	one-hot encoding	32
39	Örnek veri kümesi üzerinde standartlaştırma	33
40	Bina Veri Kümesi	34
41	Label Encoding Metodu ile Ölçeklendirme Yapılmadan Oluşturulan Model	34
42	Label encoding metodu ile ölçeklendirme yapılmadan oluşturulan lojistik regresyon modeli	35
43	Label encoding metodu ile ölçeklendirme yapılmadan oluşturulan SVM modeli	35
44	Label encoding metodu ile ölçeklendirme yapılmadan oluşturulan KNN modeli	35
45	Label encoding metodu ile ölçeklendirme yapılmadan oluşturulan karar ağaçları modeli	35
46	Label encoding metodu ile ölçeklendirme yapılmadan oluşturulan yapay sinir ağı modeli	36
47	One hot encoding metodu ile ölçeklendirme yapılmadan oluşturulan modelin doğruluk skorları	36
48	One hot encoding ve standartiasyon teknikleri ile oluşturulan modelin doğruluk skorları	36
49	One hot encoding ve standartiasyon teknikleri ile oluşturulan lojistik regresyon modeli	37
50	One hot encoding ve standartiasyon teknikleri ile oluşturulan SVM modeli	37
51	Label encoding ve standartiasyon teknikleri ile oluşturulan modelin doğruluk skorları	37
52	One hot encoding ve maks-min dönüşümü teknikleri ile oluşturulan modelin doğruluk skorları	38
53	Label encoding ve maks-min dönüşümü teknikleri ile oluşturulan modelin doğruluk skorları	38
54	One hot encoding ve standartlaştırma teknikleri ile oluşturulan SVM modeli	38

Önsöz

Bu çalışma, İstanbul Teknik Üniversitesi Matematik Mühendisliği Bölümü Lisans Programı bitirme projesi olarak yapılmıştır. Bu proje kapsamında, makine öğrenmesinin matematiksel yöntemleri araştırılmış ve analiz edilmiştir.

Bu araştırmayı yapma fırsatını bana sağlayan ve süreç boyunca büyük bir özveriyle beni destekleyen danışmanım, Prof. Dr. Elif Özkara Canfes'e en içten teşekkürlerimi sunarım.

Ayrıca, projeyi gerçekleştirmem için gerekli becerileri bana kazanmamı sağlayan diğer öğretim üyelerine, arkadaşlarıma ve aileme de teşekkür etmek istiyorum.

Hasan Çelik
Haziran 2023

Özet

Bu çalışmanın amacı makine öğrenmesindeki sınıflandırma algoritmalarını ve tekniklerini matematiksel olarak araştırıp algoritmaların hangi veriler üzerinde daha iyi çalıştığını anlamaktır. Tasarım sırasında öncelikle makine öğrenmesinin temel prensipleri ve makine öğreniminin alt dalı olan sınıflandırma problemleri açıklanmıştır. Sınıflandırma problemlerinde yaygın olarak kullanılan makine öğrenimi algoritmaların çalışma prensipleri matematiksel formüllerle tanımlanmıştır.

Algoritmaların daha doğru ve performanslı çalışması için kullanılabilecek değişken dönüşümü ve veri ölçekleme teknikleri araştırılmıştır. Araştırılan algoritma ve tekniklerle, binaların deprem sonrası hasar durumunu tahmin etmek için modeller oluşturulmuştur. Makine öğrenimi modellerinin performansını ve kullanılabilirliğini test etmek için gereken istatistiksel yöntemlere modeller değerlendirilmiştir.

1 Makine Öğrenmesi

1.1 Makine Öğrenmesi Nedir?

Makine öğrenmesi insan öğrenme sürecini otomatize edip yeni bilgiler edinmek için bilgisayarları kullanan bir sistemdir ve bilgisayarların performansını ve doğruluğunu geliştirmek için üzerinde çalışılan bir konudur. Bilgisayar bilgi edinmek için girdileri kullanır ve girdilerin yapısını farklı tekniklerle işleyerek çıktı üretir. Makine öğrenimi bu girdilere göre denetimli ve denetimsiz öğrenme olarak ikiye ayrılır. Denetimli öğrenme, girdilerin bir hedef değişkene sahip olduğu makine öğrenmesi türüdür. Algoritmalar hedef değişken dışındaki değişkenleri kullanarak hedef değişkeni tahmin etmeyi amaçlar. Sınıflandırma ve regresyon modelleri denetimli öğrenme problemleridir. Bu araştırmada sınıflandırma problemleri üzerinde çalışacağız.

1.2 Sınıflandırma Problemi ve Algoritmaları

Sınıflandırma, bir modelin girdi verilerine bir sınıf etiketi atamak üzere eğitildiği bir gözetimli makine öğrenimi tekniğidir. Öğrenim sürecinde algoritmalar girdi olarak veriler ile birlikte bu verilerin sahip olduğu sınıf etiketlerini alır. Girdi olarak alınan veri kümesi bağımlı değişken sınıf etiketlerini ise bağımsız değişken olarak düşündüğümüzde, algoritmalar bağımsız değişkenlere göre bağımlı değişkeni tahmin etmek için her bağımsız değişkene ağırlık vererek sınıf etiketine karar verir. Algoritmaların yaklaşımlarına göre yapılan sınıflandırmalar farklılık gösterir. Çalışmanın devamında sınıflandırma problemleri için uygulanan farklı algoritmaların çalışma prensiplerini incelenerek sınıflandırmanın nasıl yapıldığı araştırılacaktır. Birçok sınıflandırma algoritması olmasına rağmen yaygın olarak kullanılan algoritmalar üzerinde çalışılacaktır. Bu algoritmalar:

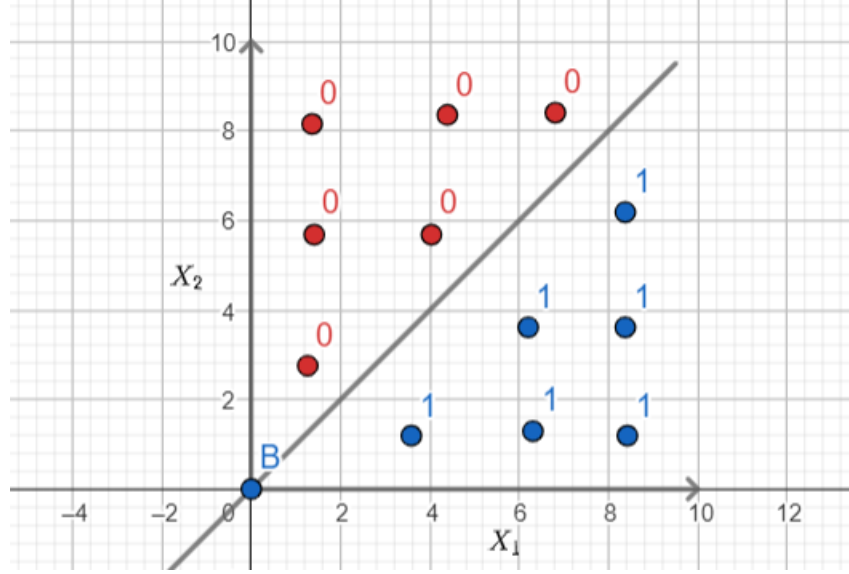
- Lojistik Regresyon
- Destek Vektör Makinaları
- K En Yakın Komşu Algoritması
- Karar Ağaçları
- Sinir Ağları
- Evrişimli Sinir Ağları
- Yinelemeli Sinir Ağları

1.3 Lojistik Regresyon

Lojistik regresyonu anlamak için öncelikle bir hiper düzlem ile ayrılan verilerin nasıl sınıflandırıldığı incelenmelidir. \mathbf{X}_i 'lerin bağımlı değişken, \mathbf{Y}_i 'lerin ise hedef değişken olduğu $(\mathbf{X}_i, \mathbf{Y}_i)$ veri noktaları için $\mathbf{X} \in R^n, \mathbf{Y} \in \{1, 0\}$ olsun. Bu veri noktalarını normal \mathbf{n} olan ve yer değiştirmesi \mathbf{b} olan $\mathbf{nX} + \mathbf{b}$ hiperdüzlemi ile kesildiğinde

$$\begin{aligned} nX + b &> 0, Y = 1 \\ nX + b &< 0, Y = 0 \end{aligned} \quad (1)$$

sınıflandırması yapılırsa, $n = [1 \ -1]$ için sınıflandırma aşağıdaki gibi olur.



Şekil 1: Hiper düzlem ile sınıflandırma

Lojistik regresyon böyle bir sınıflandırma yapmak yerine olasılık oranı yaklaşımını kullanır. Bir olayın gerçekleşme olasılığı p olduğunda gerçekleşmeme olasılığı $1 - p$ dir ve olasılık oranı $p/1 - p$ olur. Varsayılan olarak

$$\begin{aligned} P &> 0.5, y = 1 \\ P &< 0.5, y = 0 \end{aligned} \quad (2)$$

sınıflandırması yapılacağı düşünüldüğünde $nX + b$ değerlerini $(0,1)$ aralığındaki olasılık değerlerine dönüştürmek için denklem (1)'deki hiper düzlem olasılık oranına eşitlediğinde aşağıdaki denklem elde edilir.

$$nx + b = \frac{p}{1 - p} \quad (3)$$

Denklem (3) de eşitliğin sol tarafından elde edilen değer arttıkça p olasılığının arttığı , değer azaldıkça p olasılığının azaldığı görülür. Denklem (1)'deki sınıflandırma mantığına yaklaşılmış olsa da denklem (3)'de eşitliğin sağ tarafı 0 ile $+\infty$ arasında değerler alır. (3) Denkleminin sağ tarafına log dönüşümü yapıldığında

$$nx + b = \log \left(\frac{p}{1 - p} \right) \quad (4)$$

denkleminin sağ taraf değerleri $(-\infty, +\infty)$ aralığına genişletilmiş olur. Artık (4) ve (2) eşitliklerini kullanarak elde edilen p değerleri ile sınıflandırma yapılabilir. (4) denkleminde sağ taraftaki p değeri yalnız

bırakılırsa

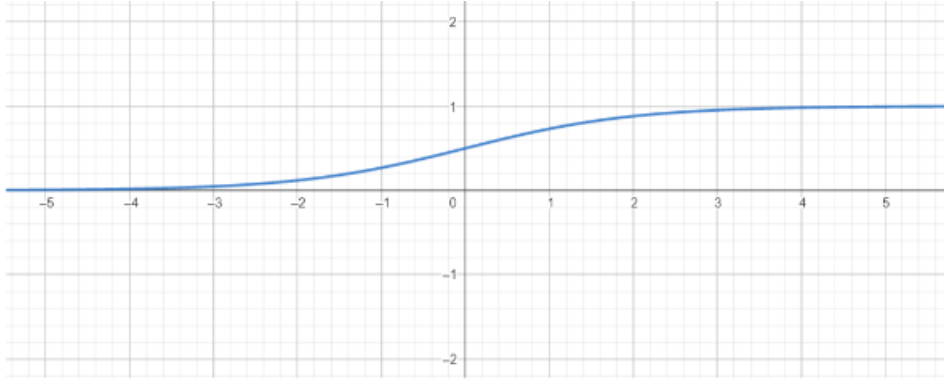
$$y = \log \left(\frac{p}{1-p} \right) \quad (5)$$

$$p = \frac{e^y}{1+e^y} = \frac{1}{1+e^{-y}}$$

denklemleri elde edilir. Denklemin sağ tarafındaki fonksiyon sigmoid fonksiyondur ve $(-\infty, +\infty)$ aralığındaki y değerlerinin, $(0,1)$ aralığındaki p değerlerine dönüştürülmesini sağlar. Sigmoid fonksiyonu veri noktalarını ayıran hiperdüzleme uygulandırsa

$$p = \frac{e^{nx+b}}{1+e^{nx+b}} = \frac{1}{1+e^{-nx-b}} \quad (6)$$

eşitliği elde edilir ve grafiği aşağıdaki gibi olur.



Şekil 2: sigmoid Fonksiyon

Lojistik Regresyon, denklem (6)'yı kullanılarak bir veri noktası için $nx + b$ değerine karşılık gelen p değerini hesaplar ve denklem (2) deki p olasılık değerine göre sınıflandırma yapar. Grafiğe bakıldığında $nx + b$ nin 0 değeri için p değeri 0.5 e karşılık gelir yani herahngi bir sınıflandırma yapılamaz. Bu değer aynı zamanda (1) denkleminde veri noktasının hiper düzelmnin üzerinde olma durumudur. Diğer değerlere bakıldığında lojistik regresyonun 0 ın etrafındaki küçük bir değişim için veri noktasının o yöndeki sınıfta olma olasılığını çok hızlı arttırdığı görülür. Herhangi bir verinin doğru sınıfta olma olasılığı aşağıdaki gibi tanımlanır.

$$P(x, y) = \left(\frac{1}{1+e^{-nx-b}} \right)^y \left(\frac{1}{1+e^{nx+b}} \right)^{1-y} \quad (7)$$

Öyleyse bütün veriler için toplam olasılık değeri,

$$P(T) = \prod_{i=1}^n P(x_i, y_i) \quad (8)$$

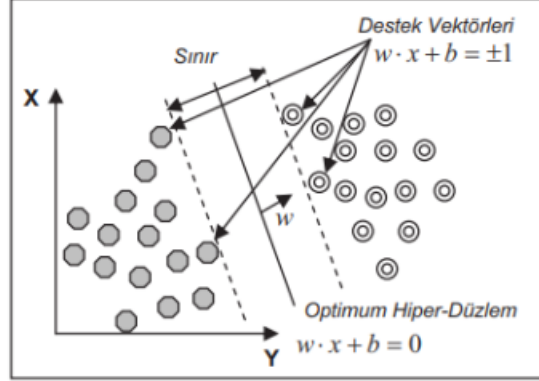
olur. n ve b parametrelerini belirlemek için (8) denklemi maksimize edilir. Denklem (7) deki ifadenin logaritması alınarak işlemler kolaylaştırılabilir ve minimize edilmek istenilen hata fonksiyonu

$$LE(n, b) = \frac{1}{n} \sum_{i=1}^n (y_i \log (1+e^{-nx_i-b}) + (1-y_i) \log (1+e^{nx_i+b})) \quad (9)$$

olur. Lojistik regresyon maksimum olabilirlik yöntemi ile (n, b) parametrelerine karar verir.

1.4 Destek Vektör Makineleri(SVM)

Destek vektör makineleri sınıflandırma problemlerinde kullanılan denetimli makine öğrenmesi algoritmalarından biridir. Destek vektör makineleri sınıflandırma yaparken iki sınıfı birbirinden ayıracak bir hiperdüzlem bulur. İki sınıfı birbirinden ayıran birçok hiperdüzlem bulunabilir ama SVM algoritması en geniş sınır aralıklı hiper düzlemi bulmayı amaçlar. Aralığı maksimuma çıkararak en uygun ayrımı yapan hiper düzlem, optimum hiper düzlem ve bu sınır aralığını oluşturan noktalar ise destek vektörleri olarak adlandırılır.



Şekil 3: Destek vektör makineleri ile sınıflandırma

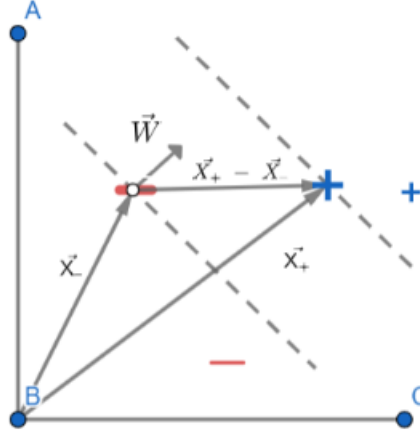
Doğrusal olarak ayrılabilen iki sınıflı bir sınıflandırma probleminde SVM'in eğitimi için k sayıda örnekten oluşan eğitim verisini olduğu kabul edilirse, $i = 1, 2, \dots, k$ için optimum hiper düzleme ait eşitsizlikler aşağıdaki gibi olur.

$$\begin{aligned} W \cdot X_i + b &\geq +1, y = +1 \\ W \cdot X_i + b &\leq -1, y = -1 \end{aligned} \quad (10)$$

Bu ifadeler aşağıdaki gibi tek bir ifade halinde yazılabilir.

$$y_i(W \cdot X_i + b) \geq 1 \quad (11)$$

SVM algoritması sınıflandırma yapmak için sınır aralığı maksimum olan hiper düzlemi bulmayı amaçladığı için sınır aralığını W ve b cinsinden ifade edilirse maksimizasyon problemi elde edilebilir. Pozitif taraftaki destek vektörünü belirleyen vektöre X_+ negatif taraftaki vektöre X_- denilirse. Bu iki vektörün farkı alındığında $X_+ - X_-$ vektörü elde edilir.



Şekil 4: Destek vektörleri ve sınır aralığı

Böylece $X_+ - X_-$ vektörü W yönündeki birim vektör ile çarpıldığında sınır aralığının uzunluğu ifade edilmiş olur.

$$d = (X_+ - X_-) \cdot \frac{\vec{W}}{\|\vec{W}\|} \quad (12)$$

(12) denklemini düzenlenirse

$$d = \frac{(X_+ \cdot \vec{W} - X_- \cdot \vec{W})}{\|\vec{W}\|} \quad (13)$$

eşitliği elde edilir. Denklem (10)'da destek vektörleri üzerindeki noktalar için

$$\begin{aligned} W \cdot X_+ + b &= +1 \\ W \cdot X_- + b &= -1 \end{aligned} \quad (14)$$

olduğundan $W \cdot X_+$ ve $W \cdot X_-$ ifadeleri yalnız bırakılırsa

$$\begin{aligned} W \cdot X_+ &= +1 - b \\ W \cdot X_- &= -1 - b \end{aligned}$$

eşitlikleri elde edilir. Bu eşitlikler denklem (13) de yerine yazıldığında

$$d = \frac{((1 - b) - (-1 - b))}{\|\vec{W}\|} = \frac{2}{\|\vec{W}\|} \quad (15)$$

denklemini elde edilir. Buradan sınır aralığını maksimize etmek için hiper düzlemin normalinin normunun minimize edilmesi gerektiği sonucuna ulaşılır. Minimize edilmek istenilen fonksiyon matematiksel kolaylık için

$$\frac{1}{2} \|W\|^2 \quad (16)$$

şeklinde belirlenebilir.

Denklem (16)'dan elde edilen sonuç, denklem (14) de tanımlanan destek vektörleri tarafından sınırlandırıldığı için minimizasyon yaparken Lagrange çarpanlarından yararlanılır. Lagrange çarpanları minimizasyon problemini çift(dual) probleme dönüştürerek problemin daha kolay çözülmesini sağlar ve eşitlik aşağıdaki gibi olur(Ayhan Erdoğmuş,2014).

$$L(w, b, a) = \frac{1}{2} \|W\|^2 - \sum_i \alpha_i (y_i (W \cdot X_i + b) - 1). \quad (17)$$

Lagrange fonksiyonunun w ve b 'ye göre kısmi türevleri alınarak aşağıdaki Karush-Kuhn-Tucker koşulları elde edilir.

$$\frac{\partial}{\partial W} L_p = W - \sum_{i=1}^n \alpha_i y_i X_i = 0 \quad (18)$$

$$W = \sum_{i=1}^n \alpha_i y_i X_i \quad (19)$$

$$\frac{\partial}{\partial b} L_p = - \sum_{i=1}^n \alpha_i y_i = 0. \quad (20)$$

(19) ve (20) Denklemlerindeki asal(primal) denklem olan (17) denkleminde yerine yazıldığında

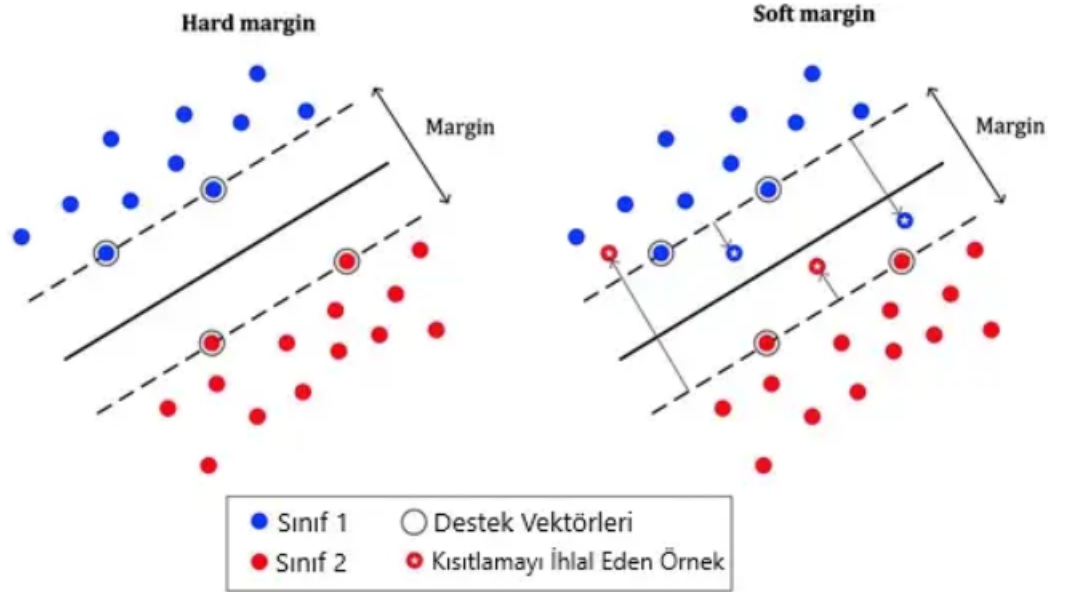
$$L(w, b, a) = \frac{1}{2} \left(\sum_i (\alpha_i y_i X_i) \sum_j \alpha_j y_j X_j \right) - \left(\sum_i \alpha_i y_i X_i \sum_j \alpha_j y_j X_j \right) - \sum_i \alpha_i y_i + \sum_i \alpha_i \quad (21)$$

$$L(w, b, a) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j X_i X_j \quad (22)$$

bulunur.

1.4.1 Belirli bir hata ile SVM (Soft Margin)

Bazı veri kümelerinde bütün veri noktalarını tam olarak sınıflandıracak lineer hiper düzlemi bulmak mümkün olmayabilir. Bu durumda SVM algoritması bazı veri noktaları için hata toleransına izin verir.



Şekil 5: Destek vektör makineleri hata toleransı

Bu sayede hiper düzlemin sınır aralığı arttırılarak model daha genel hale getirilebilir ve yeni veriler için model daha iyi performans gösterebilir. Tolerans gösterilen veri noktaları denklem (10)'da tanımlanan denklemleri sağlamadığından eşitliklerin sağlanması için bir hata miktarı tanımlanması gerekir. ϵ değeri tolerans gösterilen verilerin, sınıflarını sınırlandıran destek vektörlerine olan uzaklık olarak tanımlanırsa, hata miktarı bu epsilon değeri olur ve denklem aşağıdaki gibi olur.

$$\begin{aligned} W \cdot X_i + b &\geq +1 - \epsilon, y = +1 \\ W \cdot X_i + b &\leq -1 + \epsilon, y = -1 \\ y_i(\langle W, X_i \rangle + b) &\geq 1 - \epsilon_i \\ \xi_i &\geq 0. \end{aligned} \quad (23)$$

SVM algoritmasının yanlış sınıflandırma ihtimalini düşürmek için minimizasyon problemi

$$\frac{1}{2} ||W||^2 + C \sum_i \epsilon_i \quad (24)$$

olarak tanımlanır. Epsilon değeri bir hata terimidir. Hata terimi için bir fonksiyon tanımlanır ve denklem (24)'de yerine yazılır.

Hata fonksiyonu

$$L_h = \max(0, 1 - y(\langle w, x \rangle + b)) \quad (25)$$

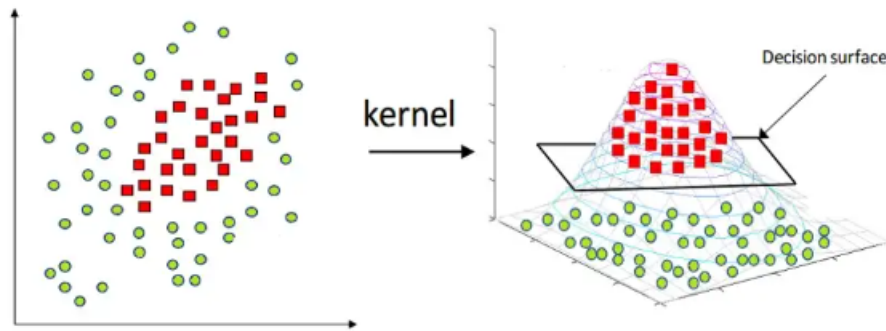
olarak ifade edilebilir ve (24) deki minimizasyon problemi

$$\frac{1}{2} ||W||^2 + L_h \quad (26)$$

şeklinde tekrar yazılır.

1.4.2 Çekirdek Hilesi

SVM ile lineer olarak ayrılamayan veri kümelerini sınıflandırmak için kullanabilecek diğer yöntem çekirdek hilesidir. Bu yöntem veri noktalarını daha yüksek boyutlarda temsil ederek sınıf ayrımını yapabilmeyi amaçlar. Örnek:



Şekil 6: Çekirdek hilesi

Yukarıda görüldüğü gibi lineer olarak ayrılamayan örnek veri kümesine bir çekirdek fonksiyonu uygulandığında örnek veri kümesi daha yüksek bir boyutta temsil edilir ve sınıf ayrımı yapılabilir.

Denklem (22)'ye bakıldığında amaç fonksiyonu, x_i ve x_j örneklerin iç çarpımı ile oluşur. Bu nedenle SVM'deki değişiklik iç çarpımı değiştirmek olacaktır. $\phi(x_i)$, x_i 'yi temsil eden bir dizi özellik olarak düşünüldüğünde, $\phi(.)$ tanımını yapmak ve x_i ve x_j örnekleri arasındaki iç çarpımı hesaplamak yerine, çekirdek fonksiyonu x_i ve x_j arasında bir benzerlik fonksiyonu olan $k(x_i, x_j)$ olarak tanımlanır. (Deisenroth et al., 2020)

$$k(x_i, x_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \quad (27)$$

Kullanılabilecek birçok çekirdek fonksiyonu olmasına rağmen yaygın olarak kullanılanlar aşağıda verildiği gibidir.

Lineer çekirdek fonksiyonu:

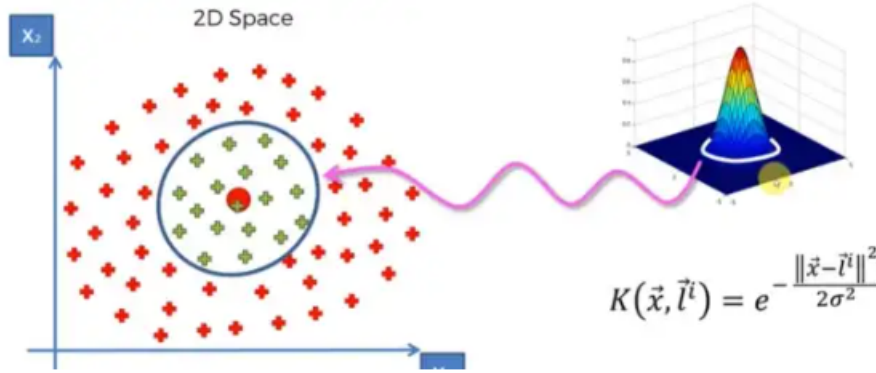
$$k(x_i, x_j) = x_i^T x_j + c \quad (28)$$

Gaussian RBF çekirdek fonksiyonu:

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \quad (29)$$

Sigmoid çekirdek fonksiyonu :

$$k(x_i, x_j) = \tanh(ax_i^T x_j + c) \quad (30)$$

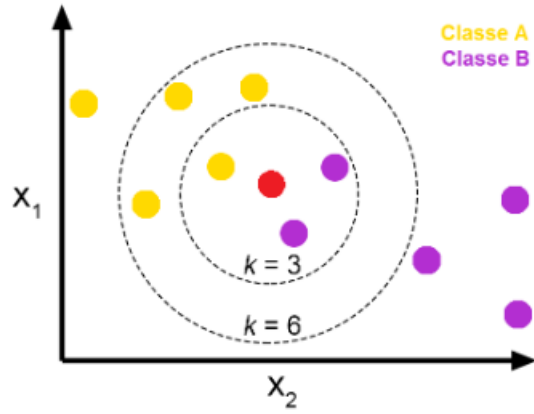


Şekil 7: RBF çekirdek dönüşümü

1.5 K En Yakın Komşu Algoritması(KNN)

Sınıflandırma problemlerinde K-En Yakın Komşu algoritması, veri kümesindeki bir noktanın en yakın komşularını bulmak ve bu komşuların sınıf etiketlerine göre veri noktasını sınıflandırmak için kullanılan bir yöntemdir. Algoritma, birbirine yakın veri noktalarının sınıf etiketlerinin benzer olacağı yaklaşımı ile sınıflandırma yapar. KNN ile sınıflandırma yapılırken, verilen bir örneğe en yakın 'K' verinin sınıf etiketlerinden maximum sayıda olan etiket örneğin sınıfı etiketi olarak belirlenir. Burda belirlediğimiz K değerlerine göre sınıflandırmalar farklılık gösterir.

Örnek:



Şekil 8: K en yakın komşu algortması ile sınıflandırma

k=3 değeri için Euclid mesafesine göre kırmızı veri noktasına en yakın 3 verinin 2 tanesi B, 1 tanesi A sınıfdan olduğu için bu veriye B sınıf etiketi atanır. K=6 değeri için belirtilen verinin en yakın 6 komşusundan 4'ü A, 2'si B sınıf etiketine sahip olduğundan veriye B sınıf etiketi atanır.

Birbirine uzaklıkları olarak yakın olan veri noktalarının sınıflarının benzer olacağı düşünüldüğünde, bahsedilen uzaklığın bir benzerlik kavramı olduğu söylenebilir. O zaman farklı uzaklık ölçüleri kullanarak bu benzerlik farklı yaklaşımlarla ölçülebilir. Kullanabilecek bazı uzaklık tanımları aşağıda verildiği gibidir.

Euclid Mesafesi :

$$d = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Manhattan Mesafesi :

$$d = \sum_{i=1}^n |x_i - y_i|$$

Minkowski mesafesi :

$$d = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$

Kosinüs Mesafesi:

$$d = 1 - \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|}$$

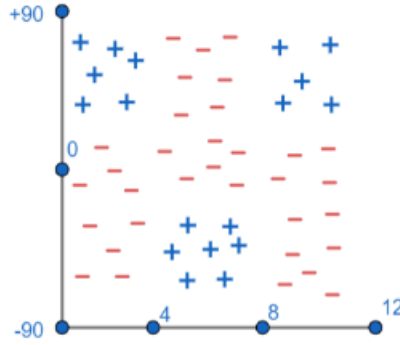
KNN algoritmasında K değeri bir çift sayı olarak belirlendiğinde bazı verilerin en yakın komşularının sınıf etiketlerinin sayısı birbirine eşit olabilir. Bu durumda algoritmanın bu veri noktalarını yanlış sınıflandırma ihtimali yükselir. Bu yüzden K değerini tek sayı olarak belirlemek daha uygun olacaktır.

1.6 Karar Ağaçları

Karar ağaçları, karmaşık veri kümelerini sınıflandırmak için belirli karar kurallarıyla veri kümesini daha küçük veri kümelerine ayırarak sınıflandırma yapar.

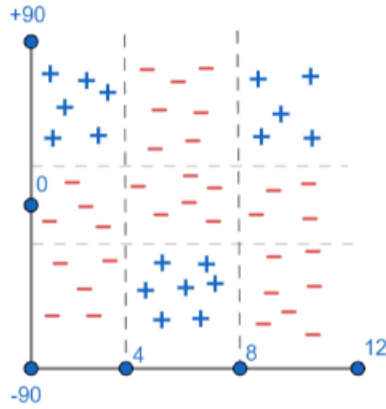
Örnek:

X_1 değişkeninin ayları, X_2 değişkeninin enlemleri belirttiğini varsayalım ve $y=0,1$ sınıf etiketleri kayak yapılabilmesi durumu olsun. Şekil(9)'da 0 sınıfını -, 1 sınıfını + değerler olduğunu kabul edelim. Kuzey yarım kürede sonbahar ve kış mevsimleri yılın ilk ve son aylarında, Güney yarım kürede ise yılın orta aylarında görüldüğü için aşağıdaki gibi bir veri kümesi elde edilir.



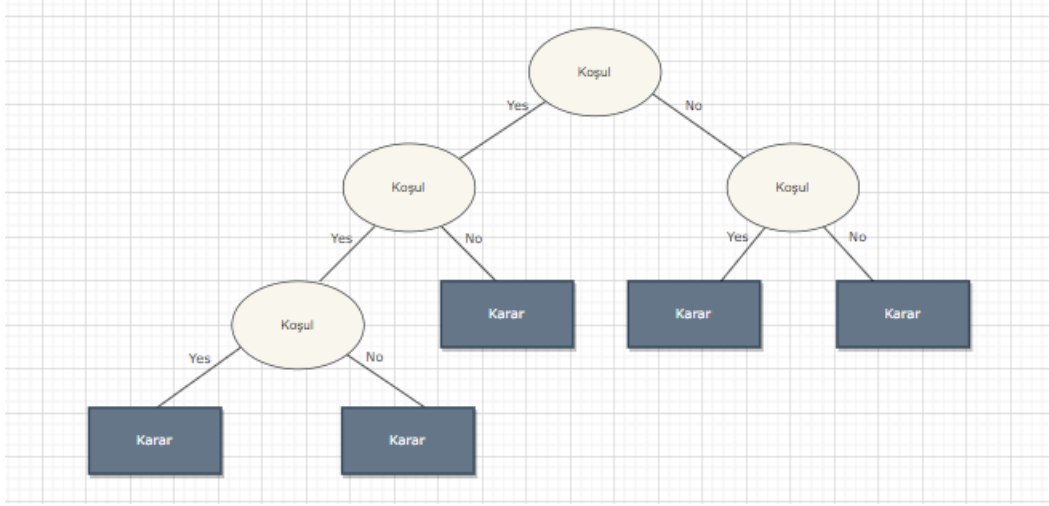
Şekil 9: Kayak yapılabilme için örnek veri kümesi

Bu verileri sınıflandırabilmek için enlemlere ve aylara karar sınırları oluşturabilir. -15 ve +15 enlem derecelerine ve 4. ve 8. aylara sınırlar çizildiğinde aşağıdaki bölünmüş veri kümesi elde edilir.



Şekil 10: Kayak yapılabilme örnek veri kümesi için karar sınırları

Bu yapı artık bir model olarak kullanılırsa, yeni bir veri noktası bulunduğu bölgedeki sınıf etiketlerine bakılarak sınıflandırılabilir. Karar ağaçlarının tanımında bahsedildiği gibi bu sınıflandırma süreci bir dizi karar kuralını takip ederek gerçekleşir ve ağaç yapısı aşağıdaki gibi olur.



Şekil 11: Karar ağacı

Karar ağaçlarında verileri bölmek için oluşturulan karar yapıları, sınıf etiketlerinin düzenine göre oluşturulur. Daha doğru sınıflandırma yapılabilmesi için aynı bölümde olan verilerin sınıf etiketlerinin düzenli yani tekdüze olması hedeflenir. Karar ağaçları verilerin düzenli olarak bölebilmek için bilgi kazancından faydalanır. Bilgi kazancını tanımlanabilmesi için öncelikle entropi kavramının anlaşılması gerekir. Şanon'un teorisine göre bir olayın gerçekleşmesinde rastgelelik ne kadar fazla ise taşıdığı bilgi o kadar fazladır. Entropi ise rastgeleliğin beklenen değeridir. Buradaki rastgelelik değeri gerçekleşme sıklığı ile ilişkilidir ve

$$I = \log_2 \left(\frac{1}{P(X)} \right) \quad (31)$$

formülü ile hesaplanır. Rastgeleliğin beklenen değeri hesaplandığında entropi formülü

$$H(X) = E(\log_2 \left(\frac{1}{P(X)} \right)) = - \sum (\log_2(P(X)) \cdot P(X)) \quad (32)$$

olarak elde edilir.

Örnek:

$$S1 = [0,1,0,1,0,1,0,1,0,1]$$

$$S2 = [1,1,1,1,1,1,1,1,1,1]$$

$S1$ ve $S2$ dizileri üzerinde için bu kavramlar incelenirse, yukarıda belirtilen teoriye göre $S1$ sınıfı $S2$ sınıfına göre daha fazla rastgelelik içerdiği için $S1$ sınıfı daha çok bilgi taşıması beklenir. Entropi değerleri incelendiğinde

$$\begin{aligned} H(S1) &= -0.5 \cdot \log(0.5) - 0.5 \cdot \log(0.5) = 1 \\ H(S2) &= -1 \cdot \log(1) - 0 \cdot \log(0) = 0 \end{aligned} \quad (33)$$

olarak elde edilir. Veri sıklığı daha da düzensiz olan $S1$ dizisinin entropisinin ve taşıdığı bilginin daha yüksek olduğu söylenebilir.

Karar ağaçları verileri hangi sütun özelliğine göre bölüneceğine karar vermek için her sütun için bilgi kazancını hesaplar. Hedef değişkenin entropi değeri $H(T)$, veri kümesinin a sütun özelliğine göre bölündüğünde ve ağırlıklı entropisi $H(T|a)$ olduğunda bilgi kazancı aşağıdaki gibi olur(Shalev-Shwartz Ben-David,2014).

$$IG(T, a) = H(T) - H(T|a) \quad (34)$$

Herhangi a sütunundaki tekil değerler $(u_1, u_2 \dots u_i)$ ve sütunun bu değerlere göre bölünmesiyle elde edilen parçalar $(S_1, S_2 \dots s_i)$ olduğunda ağırlıklı entropi değeri

$$H(T|a) = \sum_{i=1}^n \left(\frac{|s_i|}{T} \cdot H(s_i) \right) \quad (35)$$

olarak tanımlanır. Bilgi kazancı ise

$$IG(T, a) = H(T) - \sum_{i=1}^n \left(\frac{|s_i|}{T} \cdot H(s_i) \right) \quad (36)$$

şeklinde tekrar ifade edilebilir. Karar ağaçları veri kümesini hangi sütun özelliğine göre bölüneceğine karar vermek için bütün sütunlar için bilgi kazancını hesaplar ve bilgi kazancı maksimum olan sütuna göre veri kümesini böler. Bölünme sonucunda geriye kalan verileri üzerinde aynı işlemlerin uygulanmasıyla ağaç yapısı dallanmaya devam eder.

Örnek:

	c1	c2	Hedef Değişken
0	1	0	A
1	0	1	B
2	1	0	A
3	1	0	B
4	1	1	A
5	0	0	B
6	0	1	A
7	1	1	A
8	0	0	B
9	0	1	B

Şekil 12: Karar ağaçları için uygulama veri kümesi

Yukarıdaki veri kümesi üzerinden bilgi kazancı kavramını incelenirse denklem (32)' den

$$H(T) = -(0.5 \log_2(0.5) + 0.5 \log_2(0.5)) = 1$$

olarak elde edilir.

c1 sütunu için ağırlıklı entropiyi hesaplamak için c1 sütununa göre ayrılma incelenirse,

```
df[df["c1"]==1]
```

	c1	c2	Hedef Değişken
0	1	0	A
2	1	0	A
3	1	0	B
4	1	1	A
7	1	1	A

```
df[df["c1"]==0]
```

	c1	c2	Hedef Değişken
1	0	1	B
5	0	0	B
6	0	1	A
8	0	0	B
9	0	1	B

Şekil 13: c1 sütun değerlerine göre ayrılma

c1 sütunundaki 1 değeri için 4 A sınıfından, 1 B sınıfından veri olduğu için entropi değeri

$$-(\frac{4}{5} \log_2(\frac{4}{5}) + \frac{1}{5} \log_2(\frac{1}{5})) = 0.72$$

olur.0 değeri için 1 A sınıfından , 4 B sınıfından veri olduğu için entropi değeri

$$-(\frac{1}{5} \log_2 \frac{1}{5} + \frac{4}{5} \log_2 \frac{4}{5}) = 0.72$$

olur. c1 sütunu için denklem (36) kullanılarak bilgi kazancı hesaplanırsa,

$$1 - (1/2 * 0.72) + (1/2 * 0.72) = 0.28$$

sonucuna ulaşılır.c2 sütunu için bilgi kazancı hesaplanırsa,

```
df[df["c2"]==1]
```

	c1	c2	Hedef Değişken
1	0	1	B
4	1	1	A
6	0	1	A
7	1	1	A
9	0	1	B

```
df[df["c2"]==0]
```

	c1	c2	Hedef Değişken
0	1	0	A
2	1	0	A
3	1	0	B
5	0	0	B
8	0	0	B

Şekil 14: c2 sütun değerlerine göre ayrılma

c2 sütunundaki 1 değeri için 3 A sınıfından, 2 B sınıfından veri olduğu için entropi değeri

$$-\left(\frac{3}{5} \log_2 \frac{3}{5} + \frac{2}{5} \log_2 \frac{2}{5}\right) = 0.97$$

olur.0 değeri için 2 A sınıfından , 3 B sınıfından veri olduğu için entropi değeri

$$-\left(\frac{2}{5} \log_2 \frac{2}{5} + \frac{3}{5} \log_2 \frac{3}{5}\right) = 0.97$$

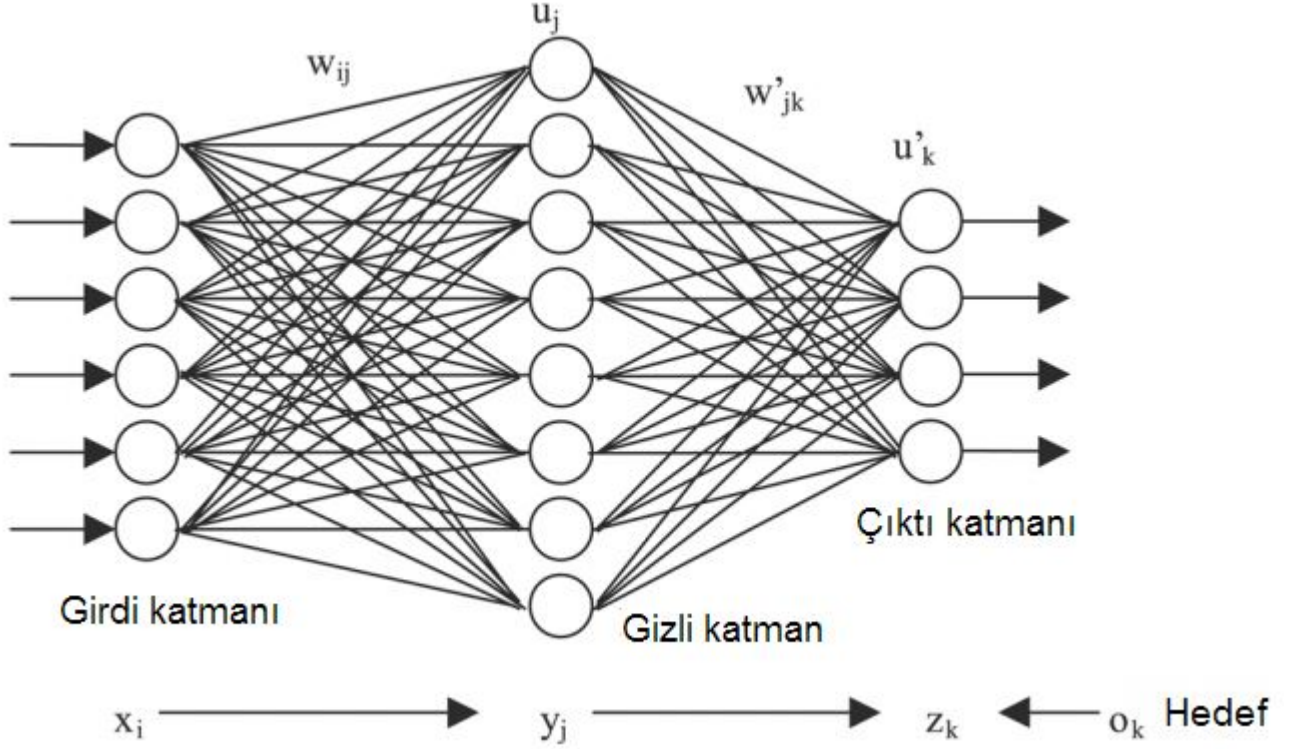
olur ve c2 sütunu için denklem (36) kullanılarak bilgi kazancı hesaplanırsa

$$1 - (1/2 * 0.72) + (1/2 * 0.72) = 0.03 \text{ sonucuna ulaşılır.}$$

Bilgi kazancı c1 sütunu için daha yüksek olduğu için veri kümesi öncelikli olarak c1 sütununa göre bölünür. Ayrıca veri kümesine bakıldığında c1 sütununda sınıf etiketlerinin daha homojen dağıldığı farkedilir yani karar ağaçları bilgi kazanımı kullanılarak sezgisel olarak doğru şekilde verileri bölebilir. Karar ağaçlarında verilerin sürekli olarak bölünmesi sonucunda model,eğitim veri kümesini aşırı öğrenecektir ve yeni veriler üzerinde yanlış tahminler yapacaktır. Bu sebepten dolayı bölünme işleminin bir bölünme sayısından sonra durdurulması gerekebilir.

1.7 Yapay Sinir Ağları

Yapay sinir ağları, insan beyninin çalışma şeklini simüle ederek öğrenmeyi amaçlayan bir makine öğrenmesi algoritmasıdır. Sinir ağları, birçok farklı katmandan oluşur. Katmanlar ise nöron olarak adlandırılan küçük işlem birimlerinden oluşur. Her nöron, bir girdi alıp işleyerek bir çıktı üretir. Bu çıktıların da diğer nöronlara ve katmanlara aktarılması ile öğrenme süreci devam eder.



Şekil 15: Yapay sinir ağları

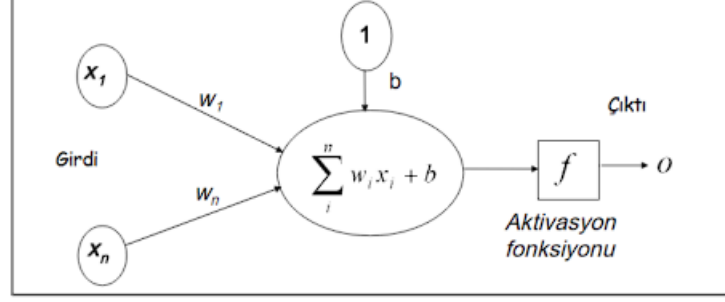
Yapay sinir ağları 3 katmandan oluşur.

- Girdi katmanı
- Gizli katmanlar
- Çıktı katmanı

Girdi katmanı bir sinir ağındaki ilk katmandır. Girdi katmanında Veri kümesinin özellikleri sinir ağına alınır ve işlenmek üzere diğer katmanlara aktarılır. Girdi katmanından alınan bilgiler gizli katmanlara aktarılır. Gizli katmanlarda, Verileri birbirine bağlayan bir dizi katman sayesinde algoritmalar daha karmaşık özellikleri keşfetmeyi amaçlar. Çıktı katmanı, gizli katmanların oluşturduğu özellikleri kullanarak girdi verileri için sonuç üretir.

1.7.1 Tek Katmanlı Yapay sinir Ağı

Tek katmanlı sinir ağları yapay sinir ağlarının en basit formudur. Girdi ve çıktı katmanları arasında bir gizli katman bulunur. Yapay sinir ağlarının çalışma prensibi tek katmanlı sinir ağı üzerinden daha kolay anlaşılabilir.



Şekil 16: Tek katmanlı yapay sinir ağı

Yukarıdaki görselde görüldüğü gibi $x_1, x_2, \dots, x_n \in R^d$ girdi özellikleri, ilgili ağırlıklarla çarpılarak gizli katmana iletilir. Bu toplam aşağıdaki şekilde ifade edilebilir.

$$x \cdot w = \sum_{i=1}^n x_i \cdot w_i$$

Sonrasında çarpımların toplamından elde edilen değere b sabiti eklenir. Gizli katmandaki nöronlara iletilen bu $z = w \cdot x + b$ değerine aktivasyon fonksiyonu uygulanarak elde edilen bilgi, çıktı katmanına iletilir ve modelin çıktısı elde edilir. Aktivasyon fonksiyonu olarak sigmoid fonksiyonu seçilirse çıktı aşağıdaki gibi elde edilir.

$$z = w^T x + b$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (37)$$

Yapay sinir ağlarında genellikle bir katmanda birden fazla nöron bulunur ve nöronlar α ile ifade edilir. Nöronun bulunduğu katman, üst indis gösterimi ile ifade edilir. İlgili nöronun katmandaki kaçınıcı nöronu belirttiğini göstermek için ise alt indis gösterimi kullanılır. Yani l'ninci katmandaki i'ninci nöron $a_i^{(l)}$ şeklinde gösterilir. Sigmoid aktivasyon fonksiyonu kullanılarak birden fazla nöron oluşturmak için girdi katmanı farklı w vektörleri ile çarpılır. Örneğin 1. katmanda 3 adet nöron bulunuyorsa denklemleri aşağıdaki gibi olur.

$$\begin{aligned} a_1^{[1]} &= \sigma(w_1^{[1]T} x + b) \\ a_2^{[1]} &= \sigma(w_2^{[1]T} x + b) \\ a_3^{[1]} &= \sigma(w_3^{[1]T} x + b) \end{aligned} \quad (38)$$

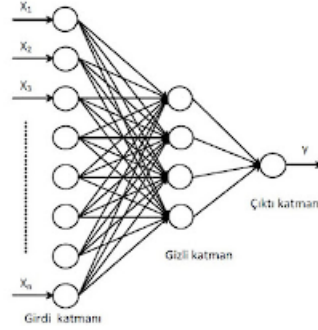
Girdi değerleri ile çarpılan w vektörleri aşağıdaki gibi matrislerle ifade edilir.

$$W^{[1]} = \begin{bmatrix} -- & w_1^{[1]T} & -- \\ -- & w_2^{[1]T} & -- \\ & \cdot & \\ -- & w_m^{[1]T} & -- \end{bmatrix} \in R^{m \times d}$$

$z = [z_1, \dots, z_m] \in R^m$ vektörü matris çarpımı ile aşağıdaki gibi ifade edilebilir.

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} = \begin{bmatrix} -- & w_1^{[1]T} & -- \\ -- & w_2^{[1]T} & -- \\ & \vdots & \\ -- & w_m^{[1]T} & -- \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ \vdots \\ b_m^{[1]} \end{bmatrix}$$

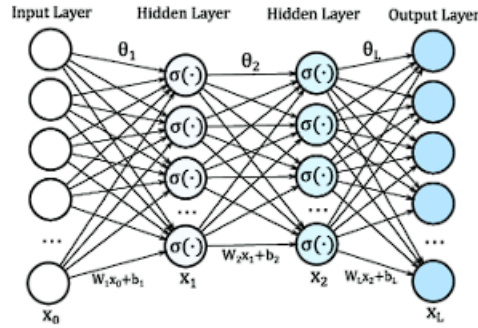
Matris temsili yukarıdaki gibi olan z değerlerine aktivasyon fonksiyonları uygulanmasıyla tek katmanlı yapay sinir ağının gizli katmanı oluşturulabilir. Bu gizli katmandaki nöronların yine ağırlıklarla çarpılarak çıkış katmanına iletilmesi ile modelin çıktısı elde edilir. Oluşturulan yapay sinir ağı aşağıdaki gibi bir yapıda olur.



Şekil 17: Tek katman ve birden çok nöron ile yapay sinir ağı

1.7.2 Çok Katmanlı Sinir Ağları

Yapay sinir ağları tanımlanırken bahsedildiği gibi sinir ağları birçok katmandan oluşabilir. Çok katmanlı yapay sinir ağları modeli bir katmandaki nöronların tekrar ağırlık vektörleri ile çarpılıp diğer katmanlara iletilmesiyle oluşturulur ve modelin yapısı tek katmanlıya benzer şekilde aşağıdaki gibi olur.



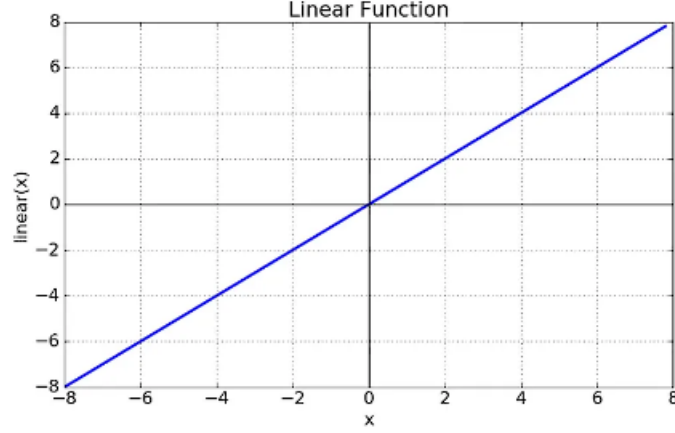
Şekil 18: Çok katmanlı yapay sinir ağı

Tek katmanlı sinir ağlarında olduğu gibi girdi katmanındaki değerler ağırlık vektörü ile çarpılarak gizli katmana iletilir. İlk gizli katmandaki nöronların bir diğer ağırlık vektörü ile çarpılıp diğer katmanlara aktarılmasıyla çok katmanlı ve birbirine bağlı sinir ağları yapısı oluşturulur ve denklemleri aşağıdaki gibi olur.

$$\begin{aligned} a_k^{[1]} &= \sigma(w_k^{[1]T} \cdot x + b) \\ a_k^{[2]} &= \sigma(w_k^{[2]T} \cdot a^{[1]} + b) \\ a_k^{[3]} &= \sigma(w_k^{[3]T} \cdot a^{[2]} + b) \end{aligned} \tag{39}$$

Gizli katman sayısı ve katmanlardaki nöronların sayısı problemin ihtiyacına göre belirlenebilir. Kullanılacak aktivasyon fonksiyonu da aynı şekilde değiştirilebilir. Problemin ihtiyacına göre aşağıdaki gibi farklı aktivasyon fonksiyonları kullanılabilir.

Lineer aktivasyon fonksiyonu:



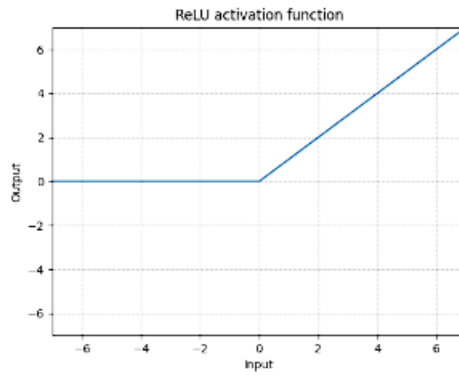
Şekil 19: Lineer aktivasyon fonksiyonu

$$f(x) = ax$$

Nöronlar arasında doğrusal ilişkiler oluşturur. Sadece lineer aktivasyon fonksiyonu ile kurulan modeller etkisizdir çünkü doğrusal bir fonksiyonun diğer doğrusal bir şekilde birleşimi yine doğrusal bir fonksiyon olur.

Sigmoid aktivasyon fonksiyonu: Şekil 2 'de belirtilen fonksiyondur 0 ile 1 değerleri arasında çıktı üretir.

ReLU aktivasyon fonksiyonu:

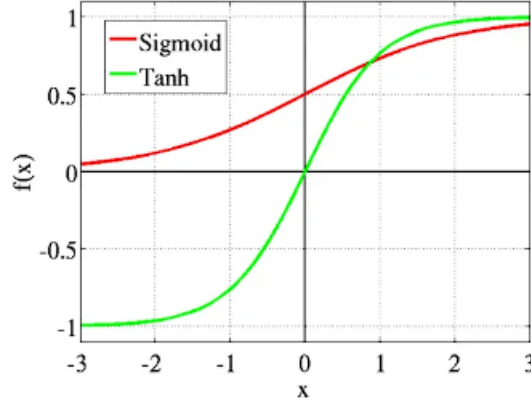


Şekil 20: ReLU aktivasyon fonksiyonu

$$f(x) = \max(0, x)$$

Pozitif ekseninde doğrusal fonksiyon ile aynı özelliklere sahiptir. Negatif ekseninde 0 değerini alır.

Tanh aktivasyon fonksiyonu:



Şekil 21: Tanh aktivasyon fonksiyonu

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Yukarıdaki aktivasyon fonksiyonlarının dışında birçok aktivasyon fonksiyonu kullanılabilir. Hangi aktivasyon fonksiyonlarının kullanılacağı problemin tanımına göre değişkenlik gösterir. Sinir ağları modeli oluşturulduktan sonra model parametresi olan W ağırlık matrisleri maliyet fonksiyonunu minimum yapacak şekilde ayarlanır. Yapay sinir ağlarında sınıflandırma problemleri için kayıp fonksiyonu olarak aşağıdaki çapraz entropi fonksiyonu kullanılır.

$$-\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (40)$$

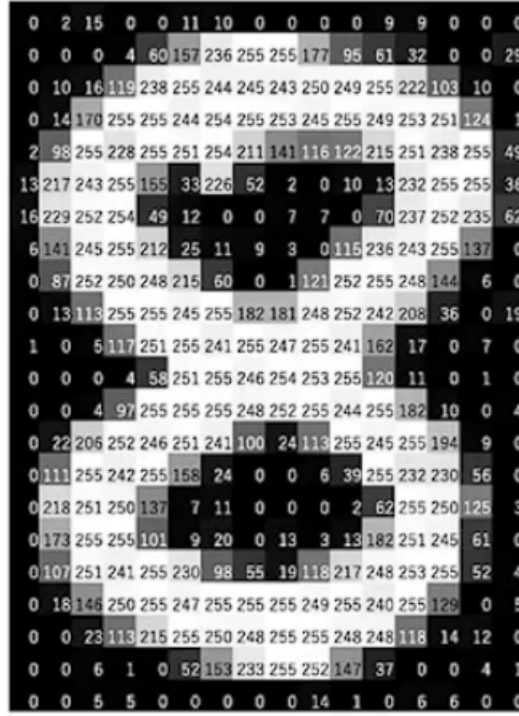
1.8 Evrişimli Sinir Ağları (CNN)

Evrişimli sinir ağları görüntü işleme ve desen tanıma gibi problemlerde yaygın olarak kullanılan yapay sinir ağı modelidir. CNN algoritmasının açıklanabilmesi için öncelikle görüntü işleme probleminin anlaşılması gerekir. Makine öğreniminde görüntü işleme görüntü sınıflandırma, nesne tanıma, görüntü restorasyonu gibi çalışmaları kapsar. Bilgisayarlar dijital görüntüleri tanıyarak işlemler yapar. Bir dijital görüntü, piksel olarak adlandırılan küçük resim elementlerinden oluşur. Her piksel görüntünün en küçük temel birimidir ve 0 dan 255 e kadar sayısal değerler alır. Örneğin 64x64 pikselden oluşan basit bir görüntü 64x64 matris formunda 0 dan 255 e kadar olan sayılarla ifade edilir. Bir görüntünün rengini oluşturan Kırmızı (R), Yeşil (G) ve Mavi (B) olmak üzere 64x64 boyutunda 3 panelin birleşimi oluşturur ve örnekte belirtilen 64x64 boyutundaki resim RGB panelleriyle 64x64x3 matris formatında olur.



Şekil 22: RGB panelleri

Paneldeki 0 değeri en koyu rengi temsil ederken 255 değeri en açık rengi temsil eder. Örnek olarak siyah beyaz bir resmin piksel değerleri aşağıdaki gibidir.



Şekil 23: Piksel değerleri

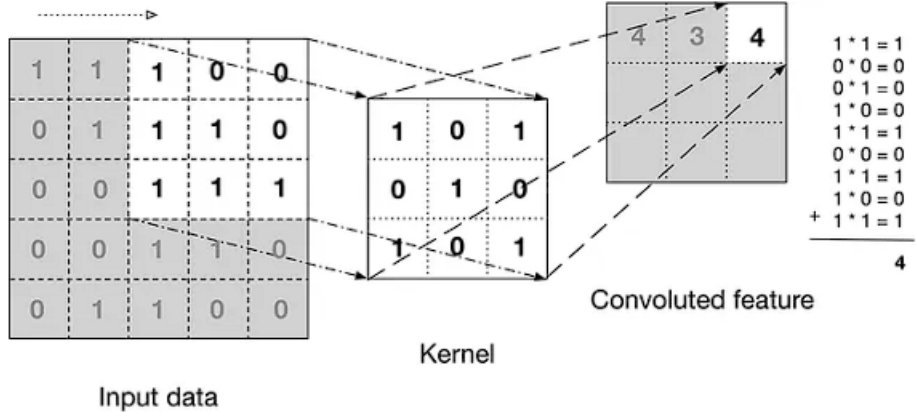
Resimlerden oluşan bir veri kümesini makine öğrenmesi modellerinde kullanılabilecek bir veri kümesi haline getirmek için her satırın bir resmi temsil etmesi gerekir. Bunun için bir resimdeki piksel değerleri sırayla sütun değerlerini oluşturur ve 64x64 boyutundaki bir resim, bir satırda 1x4096 boyutunda bir vektör ile temsil edilir.

Bir resmin sınıfını tanımlayabilmek için resmin bazı özelliklerinin keşfedilmesi gerekmektedir. Örneğin şekil 22'deki resmin bir köpek resmi olduğunu anlamak için öncelikle resimde kulak burun gibi özelliklerin köpeğe ait olduğu keşfedilmelidir. Fakat resimler bahsedildiği gibi vektörize edildiğinde herhangi bir pikselin aşağısındaki, yukarısındaki veya çaprazındaki bir piksel ilgili pikselden çok daha uzak bir sütunda bulunabilir. Bu durumda veri kümesindeki özelliklerin keşfedilmesi zorlaşır.

CNN algoritmasının yapısı, birbiri ardına sıralanmış katmanlardan oluşur. Temel olarak girdi katmanı, evrişim katmanları ve tam bağlantılı katmanı olmak üzere üç katmandan bahsedilebilir. Girdi katmanı görüntü verilerinin algılamaya alındığı katmandır. Evrişim katmanı girdi değerlerinin filtreleme gibi operasyonlarla analiz edilerek özelliklerin çıkarıldığı katmandır. Bu sayede görüntüdeki kenarlar, köşeler, desenler gibi özellikler belirlenebilir. Tam bağlantılı katmanlar, evrişim katmanlarından gelen özelliklerden temel sinir ağıları metodları ile daha büyük özellikleri keşfederek sınıflandırma probleminin çıktısını üretir.

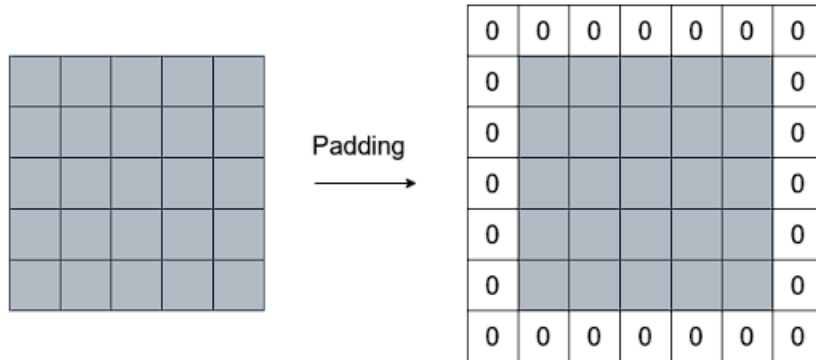
1.8.1 Evrişim Katmanı

Evrişim katmanı görüntülerin özelliklerini keşfetmek için belirli bir boyutta filtre matrisi kullanır ve bu matrisi görüntü üzerinde kaydırarak evrişim işlemini uygular. Filtre matrisi bulunduğu bölgedeki piksel değerleriyle çarpılır ve yoğunluk haritası elde edilir. Evrişim katmanında birden fazla yoğunluk haritası oluşturabilir. Bu sayede, farklı özellikler aynı anda tespit edebilir ve farklı yoğunluk haritaları ile görüntünün karmaşık özellikleri temsil edebilir. Filtreleme işlemi ile yoğunluk haritasının oluşumu şekil 24'deki gibidir.



Şekil 24: Evrişim işlemi

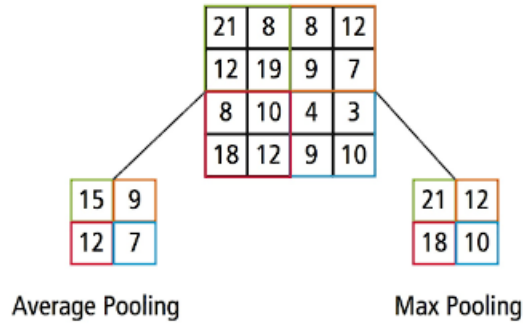
Yukarıda görüldüğü gibi 3x3 boyutundaki filtre matrisi öncelikle resim üzerindeki ilk 3x3 matris ile çarpılmıştır ve bu çarpımın sonucu özellik matrisindeki ilk değeri oluşturmuştur. Daha sonra filtre matrisi girdi matrisi üzerinde kaydırılarak özellik matrisi oluşturulmaya devam edilmiştir. Bu örnekte kaydırma işlemi 1 adım ötelenerek gerçekleşmiştir. Adım aralığı parametresi değiştirilerek filtrenin bir adımda daha fazla kaydırılması sağlanabilir. Bu işlem sonrasında özellik haritası oluşturulur fakat çıktı olarak girdi katmanından daha küçük boyutta bir matris elde edilir. Aynı zamanda girdi matrisinde kenar kısımlardaki pikseller, ortadaki piksellere göre özellik haritasına daha az etki eder ve bu durum bilgi kaybına neden olabilir. Bu durumu önlemek için dolgulama tekniği ile, girdi matrisi ile özellik matrisinin boyutları aynı olacak şekilde girdi matrisinin kenarları 0 değerleri ile doldurulabilir.



Şekil 25: CNN dolgu işlemi

1.8.2 Ortaklama (pooling) katmanı

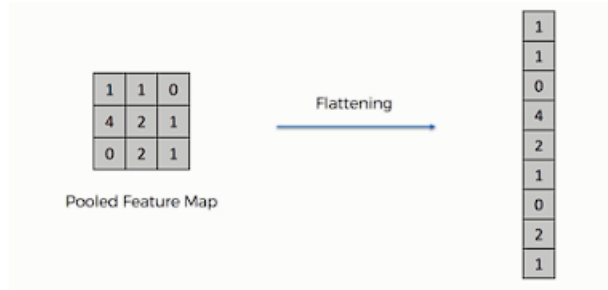
Ortaklama katmanının çıktısı olan yoğunluk haritası girdi özelliklerinin konumuna duyarlı bir yapıdır bu durumu önlemek için evrişim katmanının çıktısına bir işlem daha uygulanarak özellik haritasındaki özellikleri özetleyen ortaklama katmanı kullanılır. Temel olarak ortalama ve maksimum olmak üzere iki tür ortaklama metodu uygulanır.



Şekil 26: Ortaklama katmanı

1.8.3 Düzleştirme (flattening) katmanı

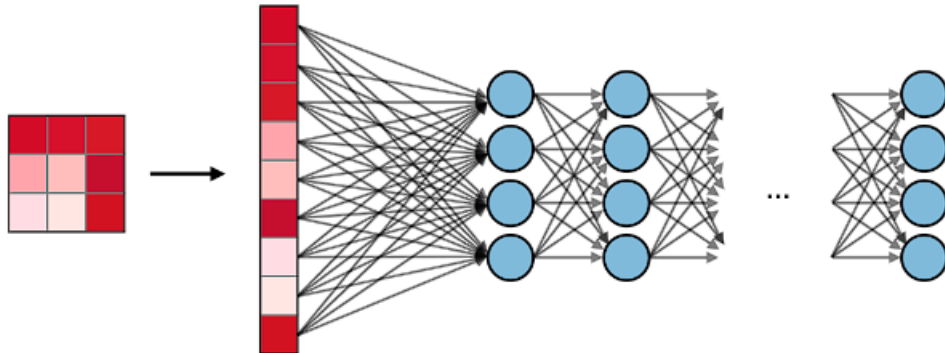
Flattenig katmanı girdi matrisinden elde edilen özelliklerin bir sonraki adımda işlenebilmesi için özellik matrisinin vektörize edildiği katmandır



Şekil 27: Düzleştirme katmanı

1.8.4 Tam bağlantı (Fully-Connected) katmanı

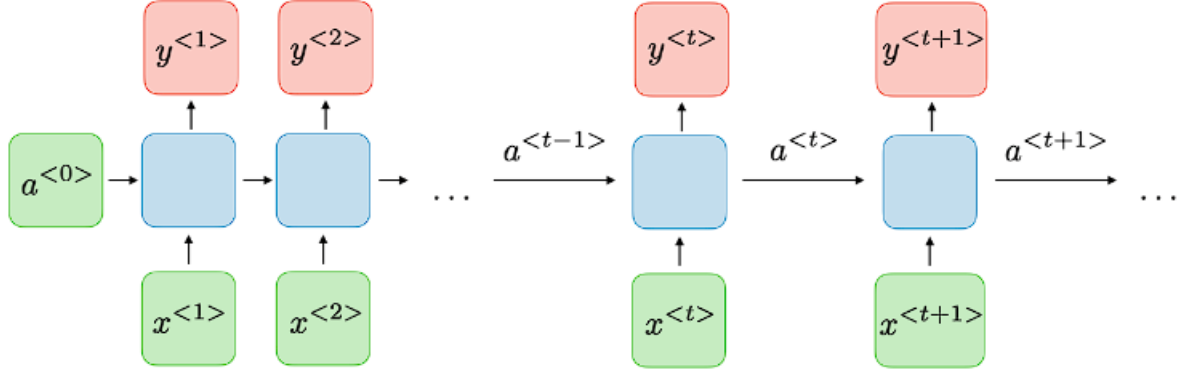
Tam bağlantılı katmanda, düzleştirme katmanının çıktısı olarak alınan özellik vektörleri girdi olarak alınır ve 3.7 de bahsedilen yapay sinir ağlarına benzer şekilde özellikler işlenerek problemin amacına yönelik çıktı üretilir.



Şekil 28: Tam bağlantı katmanı

1.9 Yinelemeli Sinir Ağları(RNN)

Zaman serisi ve dil işleme gibi problemlerde herhangi bir zamanda üretilen çıktı önceki zaman adımlarının çıktılarıyla bağlantılı olabilir. RNN algoritması hafıza hücreleri kullanarak geçmiş bilgileri hatırlar ve girdi verilerini zaman açısından sıralı olarak işler. Bu sayede herhangi bir zaman adımında geçmiş zaman adımlarının bilgileri de kullanılarak tahmin yapılır.



Şekil 29: Yinelemeli sinir ağı

T zaman adımında üretilen çıktı y_t olarak belirtilirse bu çıktı yukarıda belirtildiği gibi hem mevcut zaman adımındaki girdilere hem de geçmiş bilgilere bağlıdır.

$$y_t = f(x_t, h(t-1)) \quad (41)$$

Burdaki $h(t-1)$ "gizli durum" veya "hafıza" olarak adlandırılan h nöronunun t zaman adımındaki çıktısını temsil eder. RNN algoritması, her zaman adımında $x(t)$ girdi verisini alır ve $h(t)$ hafıza vektörünü günceller. Bu güncelleme yapay sinir ağlarındaki gibi ağırlık matrisleri kullanılarak yapılır. t zamanında h vektörünü belirleyen $x(t)$ nin ağırlık matrisine w_{hx} ve geçmiş bilgiyi temsil eden $h(t-1)$ in ağırlık matrisine w_{hh} denilirse, h vektörünün t zamanındaki değeri aşağıdaki formül ile hesaplanır.

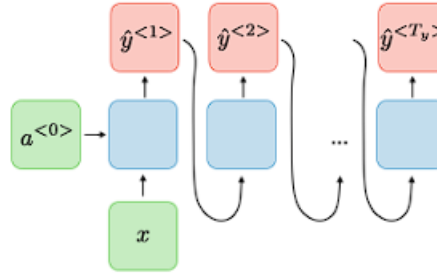
$$h(t) = f_1(W_{hx} * x(t) + W_{hh} * h(t-1) + b_h) \quad (42)$$

f fonksiyonu, hafıza hücresinin çıktısını elde etmek için kullanılan aktivasyon fonksiyonunu temsil eder. İlk zaman adımında hafıza hücresi oluşmadığı için ilk zaman adımında h vektörüne 0 vektörü atanır. Denklem 42 de $h(t)$ nin $x(t)$ ve $h(t-1)$ e bağlı olduğu farkedilebilir. t zaman adımında modelin ürettiği çıktı y_t , $h(t)$ ye ağırlık matri ile birlikte tekrar aktivasyon fonksiyonu uygulanmasıyla elde edilir.

$$y_t = f_2(W_{yh} * h(t) + b_y) \quad (43)$$

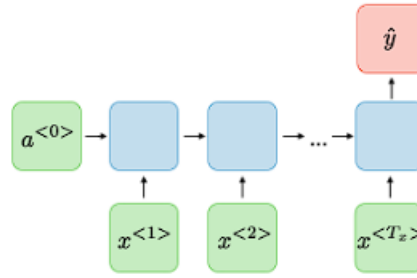
Girdi ve çıktıların yapılarına göre farklı rnn algoritmaları vardır.

Bire çok RNN: Bire çok RNN'lerde algoritma tek bir giriş alır ve sonraki zaman adımları için birden fazla çıkış üretir. Örneğin, algoritma bir resmi giriş olarak alarak resmi açıklayan bir kelime dizisi üretebilir.



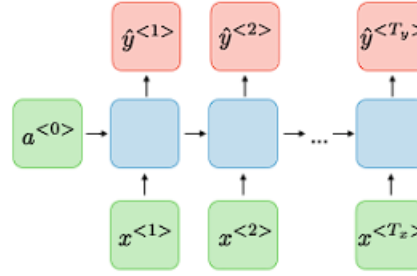
Şekil 30: Bire çok RNN

Çoka bir RNN: Çoka bir RNN’lerde algoritma birden fazla giriş alır ve tek bir çıkış üretir. Örneğin bir kelime dizisi giriş olarak alınıp, metnin duygusu pozitif veya negatif olarak sınıflandırılabilir.



Şekil 31: Çoka bir RNN

Çoka çok RNN: Çoka çok RNN’lerde algoritma birden fazla giriş alır ve birden fazla çıkış üretir. Her zaman adımının, veri kümesindeki verileri sırasıyla temsil ettiği modeller çoka çok RNN örneğidir.



Şekil 32: Çoka çok RNN

1.10 Kayıp Fonksiyonunun Minimize Edilemsi

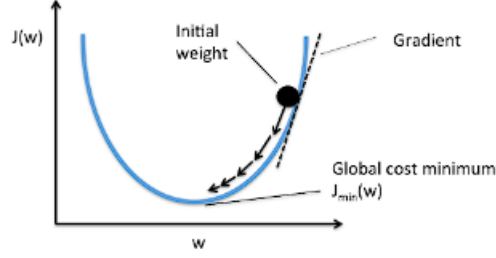
Bahsedilen makine öğrenmesi algoritmalarının en doğru şekilde çıktı üretebilmesi için kayıp fonksiyonlarını minimize edecek şekilde parametrelerinin ayarlanması gerekir. Makine öğrenmesi algoritmalarının, kayıp fonksiyonlarını minimize etmek için kullanılabileceği farklı yöntemler vardır fakat bu araştırmada yaygın olarak kullanılan "Gradyan İnişi" algoritmasından bahsedilecektir.

Gradyan inişi algoritması bir fonksiyonun minimum değerini bulmak için kullanılan bir yöntemdir. Algoritma rastgele bir başlangıç noktası seçer ve fonksiyonun eğimine bağlı olarak yeni bir nokta belirlenir. Fonksiyonun minimum noktasına ulaşana kadar bu işlem tekrarlanır. Makine öğrenmesi algoritmalarında, modelin performansını ölçmek için kullanılan kayıp fonksiyonu, modelin parametrelerine göre değişiklik gösterir. Kayıp fonksiyonu $L(\theta)$ olarak ifade edilirse θ modelin parametrelerini ifade eder. Gradyan inişi algoritmasının adımları sırasıyla aşağıda belirtildiği gibidir.

- θ_0 Başlangıç noktası belirlenir.
- " α " ile belirtilen adım boyutu belirlenir.
- Fonksiyonun seçilen noktadaki eğimi adım boyutu ile çarpılarak mevcut noktadan çıkartılır ve elde edilen değer yeni nokta olarak belirlenir.

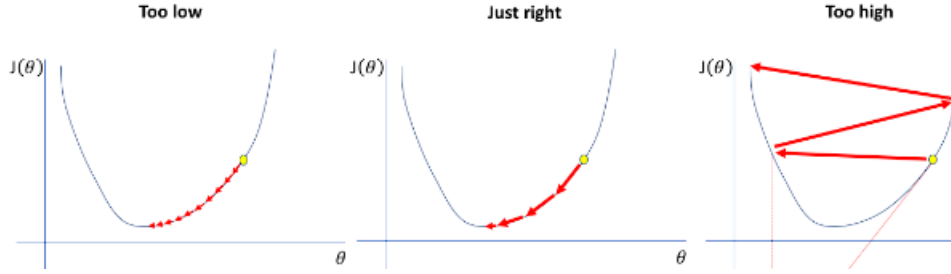
Adım sayısı t ile belirtilirse, algoritma aşağıdaki formül ile iteratif olarak çalışır.

$$\theta_{(t+1)} = \theta_{(t)} - \alpha \nabla f(\theta_{(t)}) \quad (44)$$



Şekil 33: Gradyan inişi algoritması

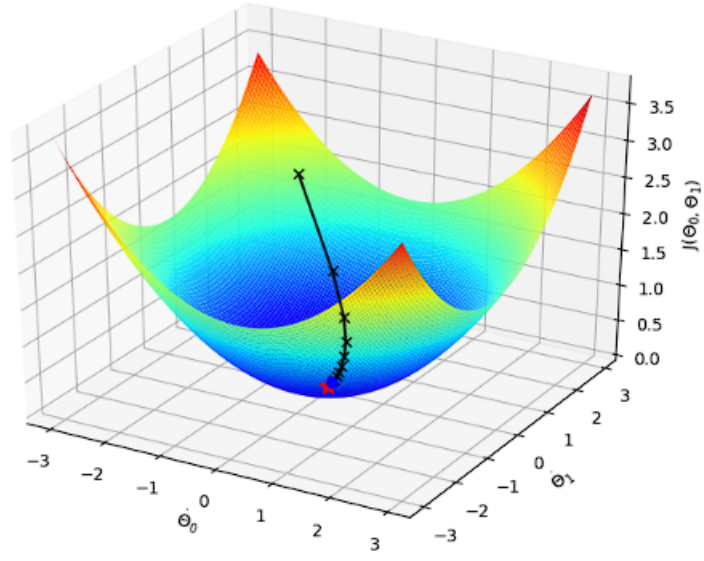
Gradyan inişi algoritmasında adım boyutu her adımda ne kadar ilerleneciğini belirten bir çarpandır. Adım boyutunun değeri minimum noktaya ulaşmak için gereken adım sayısını etkiler. Eğer adım boyutu çok küçük seçilirse minimum noktaya ulaşmak için çok sayıda adım atılması gerekebilir ve işlemi yavaşlatabilir. Adım boyutunun çok büyük seçilmesi durumunda ise minimum nokta etrafında salınım gerçekleşebilir ve optimum çözüme ulaşamayabilir.



Şekil 34: Gradyan inişi algoritmasında adım boyutu etkisi

Birden çok parametresi olan fonksiyonlarda gradyan inişi algoritması her adımda her bir parametreyi ayrı ayrı güncelleyerek işlem yapar. Örnek olarak iki parametreye sahip olan kayıp fonksiyonu ele alındığında θ_i gösterimi i 'nci parametreyi ifade eder ve gradyan inişi algoritmasının formülü aşağıda gösterildiği gibi olur.

$$\begin{aligned} \theta_0^{(t+1)} &= \theta_0^{(t)} - \alpha \frac{\partial J(\theta_0^{(t)}, \theta_1^{(t)})}{\partial \theta_0} \\ \theta_1^{(t+1)} &= \theta_1^{(t)} - \alpha \frac{\partial J(\theta_0^{(t)}, \theta_1^{(t)})}{\partial \theta_1} \end{aligned} \quad (45)$$



Şekil 35: İki parametrelili fonksiyonlarda gradyan inişli algoritması

2 Makine Öğrenmesi Modellerini Geliştirecek Teknikler

2.1 Kategorik Değişken Dönüşümleri

Veri kümelerinin makine öğrenmesi algoritmalarıyla işlenebilmesi için veri kümesindeki bütün özellik vektörlerinin sayısal değişken içeriyor olması gerekir. Eğer veri kümesinin özellik vektörleri kategorik değişken içeriyorsa bu özellik vektörleri sayısal değişkene çevirilmelidir. Kategorik değişkenleri sayısal değişkene çevirmek için farklı yöntemler vardır.

2.1.1 Label Encoding

Label encoding yöntemi ile özellik vektöründeki her kategorik değişken bu değişkeni temsil edecek benzersiz bir sayı değerine dönüştürülür.

2.1.2 One-Hot Encoding

One-Hot encoding yönteminde özellik vektöründeki her kategorik değişken için bir özellik vektörü oluşturulur. Bu özellik vektörünü ilgili kategorik değişkenin bulunduğu satırlara 1, diğer satırlara 0 atanmasıyla oluşturulur.

Aşağıdaki veri kümesi üzerinde dönüşümler incelendiğinde

	genre	lyrics	Artist	Song
0	prog	I am just a new boy,\nStranger in this town.,\n...	Pink Floyd	Young Lust Lyrics
1	prog	Eins, zwei, drei, alle!,\nOoooh You cannot reach...	Pink Floyd	Waiting For The Worms Lyrics
2	prog	All alone, or in twos,\nThe ones who really lo...	Pink Floyd	Outside The Wall Lyrics
3	prog	Into the distance a ribbon of black,\nStretche...	Pink Floyd	Learning To Fly Lyrics
4	prog	You got to be crazy, gotta have a real need,\n...	Pink Floyd	Dogs Lyrics
...

Şekil 36: Kategorik dönüşüm için örnek veri kümesi

”Artist” sütununa label encoding dönüşümü uygulandığında yeni sütun aşağıdaki gibi elde edilir.

```
In [5]: data["Artist"]=le.fit_transform(data["Artist"])
```

```
In [6]: data
```

```
Out[6]:
```

	genre	lyrics	Artist	Song
0	prog	I am just a new boy,\nStranger in this town.,\n...	6	Young Lust Lyrics
1	prog	Eins, zwei, drei, alle!,\nOoooh You cannot reach...	6	Waiting For The Worms Lyrics
2	prog	All alone, or in twos,\nThe ones who really lo...	6	Outside The Wall Lyrics
3	prog	Into the distance a ribbon of black,\nStretche...	6	Learning To Fly Lyrics
4	prog	You got to be crazy, gotta have a real need,\n...	6	Dogs Lyrics
...
825	pop	I Don't Know,\nThat You Want To Try,\nEvertime...	4	Remember The Time Lyrics
826	pop	Girl, close your eyes,\nLet that rhythm get in...	4	Rock With You Lyrics
827	pop	Help,\nI have done it again,\nI have been here...	9	Breathe Me Lyrics
828	pop	Yeah, Rihanna, Good Girl Gone Bad,\nTake three,...	8	Umbrella Lyrics
829	pop	Work, work, work, work, work, work,\nHe said me ...	8	Work Lyrics

830 rows × 4 columns

Şekil 37: label encoding

”Artist” sütununa one-hot encoding dönüşümü yapıldığında ise aşağıdaki veri kümesi elde edilir.

```
In [25]: data = pd.get_dummies(data,columns=["Artist"],drop_first=False)

In [35]: data.iloc[:,[2,3,7,9]]

Out[35]:
```

	Song	Artist_Britney Spears	Artist_Michael Jackson	Artist_Pink Floyd
0	Young Lust Lyrics	0	0	1
1	Waiting For The Worms Lyrics	0	0	1
2	Outside The Wall Lyrics	0	0	1
3	Learning To Fly Lyrics	0	0	1
4	Dogs Lyrics	0	0	1
...
825	Remember The Time Lyrics	0	1	0
826	Rock With You Lyrics	0	1	0
827	Breathe Me Lyrics	0	0	0
828	Umbrella Lyrics	0	0	0
829	Work Lyrics	0	0	0

830 rows × 4 columns

Şekil 38: one-hot encoding

Örnek veri kümelerinde görüldüğü gibi label encoding metoduyla dönüşüm yapıldığında ”Artist” sütunundaki her bir farklı değer için bir sayı ataması yapılmıştır. One-hot encoding metodunda ise” her bir farklı değer için yeni bir sütun oluşturulmuştur. Label encoding metodunda değerlerin değişken sayısına kadar değişebildiği görülür. Makine öğrenmesi algoritmaları hesaplamalarını sayılar üzerinde yaptığından dolayı ”Artist” sütunundaki kategorik değişkenlerin sayılar ile temsil edilmesi algoritmaları yanılabilir. Kategorik değişkenlerin sayısal olarak birbirinden hiçbir üstünlüğü olmamasına rağmen label encoding dönüşümü değişkenleri sayısal olarak ifade ettiği için değişkenler birbirleri arasında sayısal üstünlük kazanır. Bu durumdan kaçınmak için one-hot encoding yöntemi kullanılarak her bir değişken için yeni bir sütun oluşturulabilir ve sayısal üstünlük engellenebilir.

2.2 Özellik Ölçeklendirme ve Standartlaştırma

Makine öğrenmesi algoritmaları girdi özelliklerinin ölçeği ve dağılımı üzerinde hassas olabilirler. Bu nedenle, özellikleri belirli bir aralıkta tutmak veya belirli bir şekilde normalleştirmek, bir modelin daha iyi performans göstermesine yardımcı olabilir.

2.2.1 Maksimum - Minimum Dönüşümü

Ölçeklendirme yönteminde verilerin en büyük ve en küçük değerleri ele alınır. Diğer veriler, bu değerlere göre normalleştirilir. küçük değer 0 ve en büyük değer 1 olacak şekilde ölçekleme yapılır ve diğer bütün veriler bu 0-1 aralığına yayılır.

$$x = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (46)$$

2.2.2 Veri Standartizasyonu

Veri standartizasyonu verilerin birbirleriyle karşılaştırılabilir hale getirilmesini sağlar. Bu teknikte, verilerden veri kümesinin ortalaması çıkarılır ve standart sapmaya bölünür. Bu işlem, veri kümesinin dağılımının merkezleşmesine ve birbirine benzer ölçeklere sahip olmasına yardımcı olur. Bazı makine öğrenmesi algoritmaları verilerin standartlaştırılmış hale getirilmesini gerektirir. Veri standartizasyonu,

veri kümesinin analizlerde daha doğru sonuçlar vermesine yardımcı olur.

$$x_{standart} = \frac{x - \mu}{\sigma} \quad (47)$$

Veri setindeki özelliklerin ölçekleri ve dağılımları, algoritmanın karar sınırlarını belirlemesinde büyük bir rol oynar. Özellikle, özelliklerin ölçeği farklı olduğunda, bir algoritma bir özelliğin diğerinden daha önemli olduğunu düşünebilir. Standartlaştırma ve normalizasyon, özelliklerin birbirleriyle doğru bir şekilde karşılaştırılmasını sağlar ve böylece daha iyi karar sınırları elde edilir. Veri seti büyük olduğunda, özelliklerin ölçekleri ve dağılımları algoritmanın eğitim süresini olumsuz yönde etkileyebilir. Özellikle, büyük ölçekli özellikler, algoritmanın hızını yavaşlatabilir ve daha uzun bir eğitim süresi gerektirebilir. Standartlaştırma ve normalizasyon, özelliklerin ölçeklerini ve dağılımlarını sınırlar, bu da algoritmanın daha hızlı eğitilmesine ve daha hızlı sonuçlar elde edilmesine yardımcı olur. Örnek olarak, bölüm 3.1.1 de bahsedilen ve pozitif değerli sayılardan oluşan veri kümesi üzerine standartlaştırma uygulandığında aşağıdaki veri kümesi elde edilir.

df_norm						
	damage_grade	source_water_pre_eq	source_water_post_eq	source_cooking_fuel_pre_eq	source_cooking_fuel_post_eq	source_light_pre_eq
0	1.355491	-0.133871	-0.115002	0.302699	0.304025	-0.417755
1	-2.388341	-0.133871	-0.115002	0.302699	0.304025	2.600968
2	-1.452383	-0.133871	-0.115002	0.302699	0.304025	2.600968
3	-0.516425	-0.133871	-0.115002	0.302699	0.304025	2.600968
4	-0.516425	-0.133871	-0.115002	0.302699	0.304025	2.600968
...
16745	-0.516425	-0.133871	-0.115002	0.302699	0.304025	2.600968
16746	0.419533	-0.133871	-0.115002	0.302699	0.304025	2.600968
16747	0.419533	-0.133871	-0.115002	0.302699	0.304025	-0.417755
16748	1.355491	-0.133871	-0.115002	0.302699	0.304025	-0.417755
16749	-0.516425	-0.133871	-0.115002	0.302699	0.304025	-0.417755

16750 rows × 58 columns

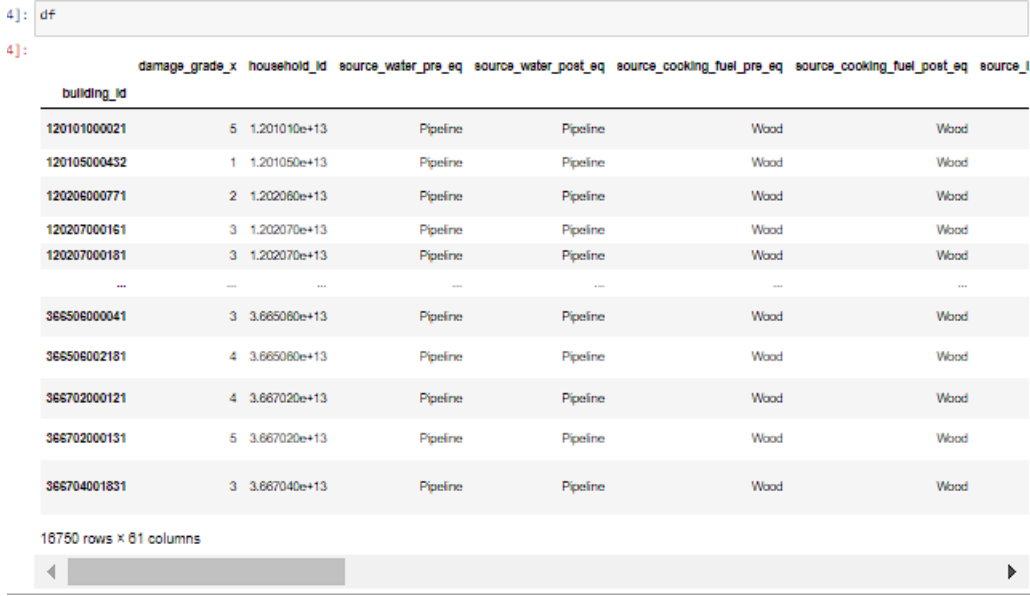
Şekil 39: Örnek veri kümesi üzerinde standartlaştırma

3 Makine Öğrenmesi Tekniklerinin Uygulanması

Bu kısımda araştırmada bahsedilen makine öğrenmesi algortimaları ve teknikleri birkaç problem üzerinde uygulanarak değerlendirilecektir.

3.1 Deprem Binalar Üzerindeki Hasarının Tahmini

Bu problemde, binaların yapı özellikleri ve depremde aldığı hasar düzeyini içeren bir veri kümesi kullanılarak yapı özelliklerine göre hasar düzeyini tahmin etmeyi amaçlayan makine öğrenmesi modeli oluşturulacaktır. Veri kümesi şekil(39)'da verildiği gibidir.



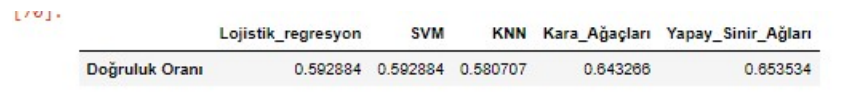
building_id	damage_grade_x	household_id	source_water_pre_eq	source_water_post_eq	source_cooking_fuel_pre_eq	source_cooking_fuel_post_eq	source_l
120101000021	5	1.201010e+13	Pipeline	Pipeline	Wood	Wood	
120105000432	1	1.201050e+13	Pipeline	Pipeline	Wood	Wood	
120206000771	2	1.202060e+13	Pipeline	Pipeline	Wood	Wood	
120207000161	3	1.202070e+13	Pipeline	Pipeline	Wood	Wood	
120207000181	3	1.202070e+13	Pipeline	Pipeline	Wood	Wood	
...
366506000041	3	3.665060e+13	Pipeline	Pipeline	Wood	Wood	
366506002181	4	3.665060e+13	Pipeline	Pipeline	Wood	Wood	
366702000121	4	3.667020e+13	Pipeline	Pipeline	Wood	Wood	
366702000131	5	3.667020e+13	Pipeline	Pipeline	Wood	Wood	
366704001831	3	3.667040e+13	Pipeline	Pipeline	Wood	Wood	

Şekil 40: Bina Veri Kümesi

Bu uygulamada, Bölüm 3' de bahsedilen makine öğrenmesi algortimaları ile bölüm 4' de bahsedilen akine öğrenmesi modellerini geliştirecek teknikler kullanılarak farklı modeller oluşturulacaktır ve modellerin performansları değerlendirilecektir.

3.1.1 Label Encoding Metodu ile Ölçeklendirme Yapılmadan Oluşturulan Model

Veri kümesi incelendiğinde veri kümesinde kategorik değişken içeren sütunların olduğu farkedilir. 4.1 de bahsedildiği gibi bu bilgileri kaybetmeden model oluşturabilmek için kategorik değişken dönüşümü uygulanması gerekmektedir. Label encoding işlemi uyguladıktan sonra algortimaların performansı aşağıdaki gibi olur.



	Lojistik_regresyon	SVM	KNN	Kara_Ağaçları	Yapay_Sinir_Ağları
Doğruluk Oranı	0.592884	0.592884	0.580707	0.643266	0.653534

Şekil 41: Label Encoding Metodu ile Ölçeklendirme Yapılmadan Oluşturulan Model

Doğruluk skorlarında çok yüksek skorlar elde edilemediği görülüyor birden fazla sınıf olduğu için tahmin matrisleri incelenebilir.

Lojistik Regresyon:

```
In [131]: print(confusion_matrix(y_test, y_predict_lojistik_reg))
```

[[49	7	22	286	0]
[16	3	15	294	0]
[8	1	19	596	0]
[6	1	22	2412	0]
[0	0	2	429	0]]

Şekil 42: Label encoding metodu ile ölçeklendirme yapılmadan oluşturulan lojistik regresyon modeli

Destek Vektör Makineleri:

```
In [135]: print(confusion_matrix(y_test, y_predict_svm))
```

[[5	0	0	359	0]
[3	0	0	325	0]
[1	0	0	623	0]
[2	0	0	2439	0]
[1	0	0	430	0]]

Şekil 43: Label encoding metodu ile ölçeklendirme yapılmadan oluşturulan SVM modeli

KNN:

```
In [141]: print(confusion_matrix(y_test, y_predict_knn))
```

[[102	35	30	190	7]
[50	51	34	189	4]
[38	42	69	465	10]
[48	46	107	2191	49]
[11	11	13	377	19]]

Şekil 44: Label encoding metodu ile ölçeklendirme yapılmadan oluşturulan KNN modeli

Karar Ağaçları:

```
In [138]: print(confusion_matrix(y_test, y_predict_dt))
```

[[283	39	21	19	2]
[29	160	64	63	12]
[19	90	267	211	37]
[5	80	232	1862	262]
[5	11	28	244	143]]

Şekil 45: Label encoding metodu ile ölçeklendirme yapılmadan oluşturulan karar ağaçları modeli

Yapay Sinir Ağları:

```
In [153]: print(confusion_matrix(yy_test, yy_pred_ysa))
```

```
[[ 264  48  20  32   0]
 [  19 106 101 102   0]
 [  14  59 146 402   3]
 [   4  40 110 2277  10]
 [   0   2  14  400  15]]
```

Şekil 46: Label encoding metodu ile ölçeklendirme yapılmadan oluşturulan yapay sinir ağı modeli

Değerlendirme

Modellerin tahmin matrisleri incelendiğinde karar ağaçları ve yapay sinir ağlarının diğer modellere göre daha iyi tahmin performansı gösterdiği söylenebilir. Lojistik regreyon ve SVM modellerinde ise algoritmaların bütün sınıf etiketlerini 4. sınıf etiketi olarak tahmin ettiği görülüyor. Yüzde 59 doğruluk oranı olan bu modellerin performansı bu bakımdan kötü görünmüyor olabilir fakat yüksek doğruluk oranı elde edilmesinin sebebi hedef değişkenlerin çoğunluğunun 4. sınıf etiketinden olmasıdır. Algoritmaların sınıf etiketlerine çoğunlukla 4. sınıf etiketini atamasının sebebi de hedef değişkenin çoğunluğunun 4. sınıf etiketinden olmasıdır. Bu problem diğer tekniklerin de uygulanmasıyla çözülmeye çalışılacaktır.

3.1.2 One hot encoding metodu ile ölçeklendirme yapılmadan oluşturulan model

Doğruluk Oranları:

	Lojistik_regresyon	SVM	KNN	Kara_Ağaçları	Yapay_Sinir_Ağları
Doğruluk Oranı	0.588676	0.583572	0.570917	0.649713	0.676457

Şekil 47: One hot encoding metodu ile ölçeklendirme yapılmadan oluşturulan modelin doğruluk skorları

Label encoding ve one hot encoding ile oluşturulan modellerin skorları incelendiğinde doğruluk oranları arasında belirgin bir fark gözlemlenmemiştir. Ayrıca lojistik regresyon ve SVM modelleri 5.1.1’ de olduğu gibi neredeyse bütün verilerin sınıf etiketini 4. sınıf olarak tahmin etmiştir.

3.1.3 One hot encoding ve standartlaştırma metodlarıyla oluşturulan model

4.2’de bahsedilen veri ölçeklendirme ve standartlaştırma metodlarının, algoritmaların performansı üzerindeki etkisini incelemek için veri setine standartizasyon uygulanırsa doğruluk skorları aşağıdaki gibi elde edilir. İlk olarak one hot encoding ile dönüşüm yapıldıktan sonra standartlaştırma yapılarak oluşturulan modelin doğruluk skorları incelenirse

```
res
```

	Lojistik_regresyon	SVM	KNN	Kara_Ağaçları	Yapay_Sinir_Ağları
Doğruluk Oranı	0.675263	0.691977	0.672875	0.654489	0.674089

Şekil 48: One hot encoding ve standartizasyon teknikleri ile oluşturulan modelin doğruluk skorları

Doğruluk skorlarına bakıldığında özellikle lojistik regresyon ve SVM algoritmalarının standartizasyon yapılmadan oluşturulan modele göre daha iyi performans gösterdiği görülür. Farkı anlamak için lojistik regresyon ve SVM algoritmalarının doğruluk matrisleri incelenebilir.

Lojistik regresyon

```
print(confusion_matrix(y_test, y_predict_lojistik_reg))
```

```
[[ 251  31  11  71  0]
 [ 44  89  52 142  1]
 [ 30  63  64 465  2]
 [ 27  52  43 2312  7]
 [ 4  3  4 408 12]]
```

Şekil 49: One hot encoding ve standartizasyon teknikleri ile oluşturulan lojistik regresyon modeli

SVM

```
print(confusion_matrix(y_test, y_predict_svm))
```

```
[[ 265  46  5  48  0]
 [ 11 144  37 136  0]
 [ 3  80 132 407  2]
 [ 1  34  56 2339 11]
 [ 1  12  3  397 18]]
```

Şekil 50: One hot encoding ve standartizasyon teknikleri ile oluşturulan SVM modeli

Yukarıdaki doğruluk matrislerinde de görüldüğü gibi standartlaştırma işlemi uygulandıktan sonra lojistik regresyon ve SVM algoritmalarının bütün sınıf etiketlerini 4. sınıf etiketi olarak tahmin etme hatası giderilmiştir ve SVM algoritmasında yüzde 69 ile yüksek bir doğruluk skoru elde edilmiştir. Veri kümelerini standartlaştırırken kategorik dönüşüm tekniğinin önemini incelemek için label encoding yöntemi ile dönüşüm yapılan verilere standartlaştırma işlemi uygulayarak model oluşturulabilir.

3.1.4 Label encoding ve standartizasyon teknikleri ile oluşturulan model

Label encoding ve standartlaştırma teknikleri ile oluşturulan modelin doğruluk skorları aşağıdaki gibi olur.

	Lojistik_regresyon	SVM	KNN	Kara_Ağaçları	Yapay_Sinir_Ağları
Doğruluk Oranı	0.651385	0.667622	0.648997	0.646848	0.67001

Şekil 51: Label encoding ve standartizasyon teknikleri ile oluşturulan modelin doğruluk skorları

Algoritmaların doğruluk oranlarına bakıldığında ,lebel encoding ile oluşturulan modelin doğruluk oranlarının one hot encoding metoduna göre biraz daha düşük olduğu görülür. Bu veri setinde farklı kategorik değişken dönüşümleri ve ölçekleme tekniklerinden hangilerinin daha iyi performans gösterdiğini incelemek için kategorik dönüşümlerin maksimum-minimum ölçekleme tekniği ile işlendiği modellerin skorları incelenebilir.

One hot encoding ve min-maks ölçekleme teknikleri ile oluşturulan model:

```
res
```

	Lojistik_regresyon	SVM	KNN	Kara_Ağaçları	Yapay_Sinir_Ağları
Doğruluk Oranı	0.652101	0.658548	0.639446	0.639446	0.683859

Şekil 52: One hot encoding ve maks-min dönüşümü teknikleri ile oluşturulan modelin doğruluk skorları

Label encoding ve min-maks ölçekleme teknikleri ile oluşturulan model:

```
res
```

	Lojistik_regresyon	SVM	KNN	Kara_Ağaçları	Yapay_Sinir_Ağları
Doğruluk Oranı	0.652101	0.658548	0.639446	0.652579	0.683338

Şekil 53: Label encoding ve maks-min dönüşümü teknikleri ile oluşturulan modelin doğruluk skorları

Değerlendirme

Doğruluk skorlarına bakıldığında, one hot encoding ve standartlaştırma metodlarıyla oluşturulan modelde SVM algoritmasının 0.69 doğruluk oranıyla en yüksek skor ile tahmin yaptığı görülüyor. Ayrıca one hot encoding ve min maks dönüşümü tekniklerinin uygulandığı model ile yapay sinir ağları, 0.68 doğruluk oranı ile tahminleme yapmıştır. Buradan veri kümesinde çok değişkenli kategorik özellikler olduğunda, one hot encoding dönüşümü uygulanmasının faydalı olabileceği çıkarılabilir.

3.2 Oluşturulan Modellerin Kullanılabilirliği

Oluşturulan modellerden en yüksek doğruluk skoru elde edilen modelin doğruluk matrisi aşağıdaki gibidir.

```
: print(confusion_matrix(y_test, y_predict_svm))  
[[ 265   46    5   48    0]  
 [  11  144   37  136    0]  
 [   3   80  132  407    2]  
 [   1   34   56 2339   11]  
 [   1   12    3  397  18]]
```

Şekil 54: One hot encoding ve standartlaştırma teknikleri ile oluşturulan SVM modeli

Deprem binalar üzerindeki hasar durumunu tahmin etmeyi amaçlayan bir model için yüzde 68 doğruluk oranı yüksek bir skor olarak değerlendirilmeyebilir. Problemin tanımı göz önünde bulundurulduğunda, tahmin edilen hasar dereceleri ile normal hasar dereceleri arasında fazla fark olmamalıdır. Fakat örnek olarak 4 hasar derecesine sahip olan bir binanın 5 derece olarak tahmin edilmesi, 1 olarak tahmin edilmesinden daha iyi olacaktır. Bu sayede modelin kullanılabilirliği söz konusu olabilir. Modelin, yakın değerleri de dahil ederek ne kadar iyi tahminleme yaptığını gözlemlemek için tahmin değerlerine aşağıdaki dönüşüm uygulanabilir.

- 1 - 2 hasar derecesine sahip binalar : 1 hasar derecesi
- 3 hasar derecesine sahip binalar : 3 hasar derecesi
- 4-5 hasar derecesine sahip binalara : 5 hasar derecesi

Yukarıdaki dönüşüm ile hasar derecelerinin az,orta ve ağır olmak üzere 3 sınıfta değerlendirilmesi amaçlanmıştır. Bu dönüşüm hem tahmin değerlerine hem de gerçek değerlere uygulandığında one hot encoding ve standartlaştırma teknikleriyle oluşturulan modelde, SVM algoritmasının performansı aşağıdaki gibi olur.

```
print(classification_report(yy_test, yy_pred))
```

	precision	recall	f1-score	support
1	0.73	0.61	0.66	692
3	0.57	0.16	0.25	624
5	0.81	0.97	0.88	2872
accuracy			0.79	4188
macro avg	0.71	0.58	0.60	4188
weighted avg	0.76	0.79	0.75	4188

Bu şekilde değerlendirilen bir model için 0.79 doğruluk oranı elde edildiği görülüyor. Sonuç olarak problemin tanımına göre alanında uzman kişilerce, veri özelliklerinin artırılması veya yeni değişkenlerin oluşturulması ile doğruluk skorları kullanılabilir bir düzeye çıkarılabilir.

Kaynaklar

- [1] Deisenroth, M. P., Faisal, A. A., Ong, C. S. (2020). Mathematics for Machine Learning. Cambridge University Press.
- [2] Ayhan, S., Erdoğan, Ş. (2014). Destek Vektör Makineleriyle Sınıflandırma Problemlerinin Çözümü İçin Çekirdek Fonksiyonu Seçimi . Eskişehir Osmangazi Üniversitesi İİBF DERGİSİ, 9(1), 175–198.
- [3] Shalev-Shwartz, S., Ben-David, S. (2013). Understanding machine learning: From theory to algorithms. Understanding Machine Learning: From Theory to Algorithms (Vol. 9781107057135, pp. 1–397). Cambridge University Press.
- [4] Ng, A. (n.d.). CS229 Lecture Notes [PDF]. Stanford University.
https://cs229.stanford.edu/notes2022fall/main_notes.pdf
- [5] CS 230 - Recurrent neural networks Cheatsheet. (n.d.). Stanford University.
<https://stanford.edu/shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>
- [6] Wang, N. (n.d.). Recurrent Neural Network [PDF]. Toronto University.
https://www.cs.toronto.edu/tingwuwang/rnn_tutorial.pdf