6.2.2. K-En Yakın Komşu

1967 yılında T. M. Cover ve P. E. Hart tarafından önerilen, örnek veri noktasının bulunduğu sınıfın ve en yakın komşunun, K değerine göre belirlendiği bir sınıflandırma yöntemidir. Denetimli öğrenmede sınıflandırma ve regresyon için kullanılan algoritmalardan biridir. En basit makine öğrenmesi algoritması olarak kabul edilir.

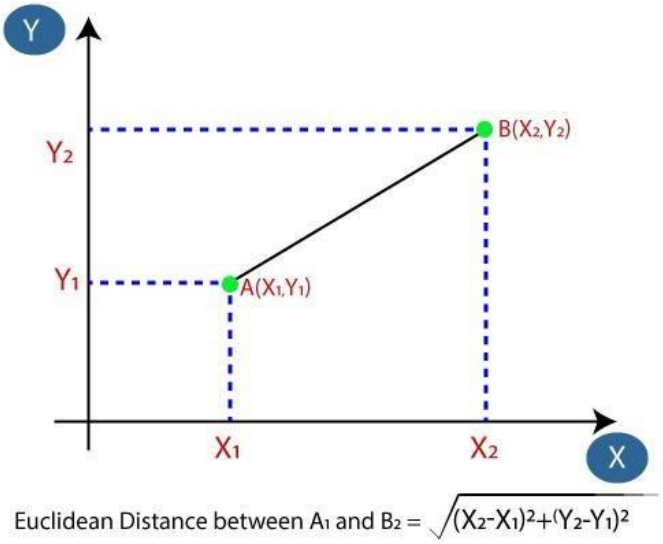
KNN amacı, yeni bir örnek geldiğinde var olan öğrenme verisi üzerinde sınıflandırma yaparak onun en yakın K komşusuna bakarak örneğin sınıfına karar verir.

K-NN algoritması sınıflandırma algoritmasıdır. Sınıflandırmak, belirli bir veri kümesini farklı sınıflara ayırma işlemidir. Sınıflandırma hem yapılandırılmış hem de yapılandırılmamış veri türleri üzerinde uygulanabilir.

KNN algoritması sınıflandırılmak istenen bir veriyi daha önceki verilerle olan yakınlık ilişkisine göre sınıflandıran bir algoritmadır. Algoritma adının içinde bulunduğu “K” algoritmaya dahil edilecek veri kümesindeki veri sayısını ifade etmektedir. Yani algoritmada “k” adet komşu aranır. Bir tahmin yapmak istediğimizde, tüm veri setinde en yakın komşuları arar. Algoritmanın çalışmasında bir K değeri belirlenir. Bu K değerinin anlamı bakılacak eleman sayısıdır. Bir değer geldiğinde en yakın K kadar eleman alınarak gelen değer arasındaki uzaklık hesaplanır. İlgili uzaklılardan en yakın k komşu ele alınır. Öznitelik değerlerine göre k komşu veya komşuların sınıfına atanır. Seçilen sınıf, tahmin edilmesi beklenen gözlem değerinin sınıfı olarak kabul edilir. Yani yeni veri etiketlenmiş (label) olur.

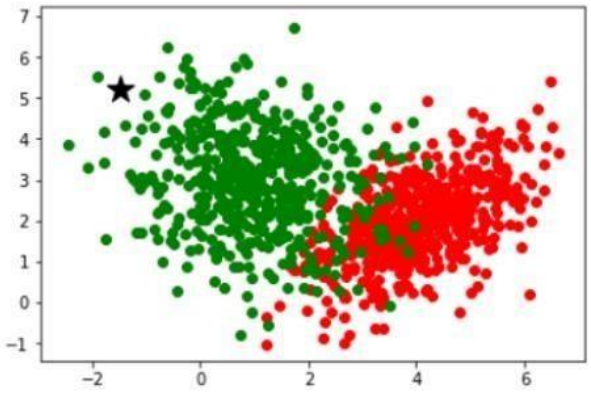
K-NN non-parametric (parametrik olmayan), lazy (tembel) bir öğrenme algoritmasıdır. lazy kavramını anlamaya çalışırsak “eager learning” aksine “lazy learning”’in bir eğitim aşaması yoktur. Eğitim verilerini öğrenmez, bunun yerine eğitim veri kümesini “ezberler”. Uzaklık hesaplama işleminde genelde Öklid fonksiyonu kullanılır.

Öklid fonksiyonuna alternatif olarak Manhattan, Minkowski ve Hamming fonksiyonlarıda kullanılabilir. Uzaklık hesaplandıktan sonra sıralanır ve gelen değer uygun olan sınıfa atanır.



Bir tahmin yapmak için KNN algoritması, lojistik veya doğrusal regresyonda olduğu gibi bir eğitim veri kümesinden öngörücü bir model hesaplamaz. Aslında, KNN'nin tahmine dayalı bir model oluşturmasına gerek yoktur. Bu nedenle, KNN için gerçek bir öğrenme aşaması yoktur. Bu yüzden genellikle tembel bir öğrenme yöntemi olarak kategorize edilir. Bir tahmin yapabilmek için, KNN herhangi bir eğitim aşaması olmadan bir sonuç üretmek için veri setini kullanır.

KNN, bir tahmin yapmak için tüm veri kümesini depolar. KNN herhangi bir tahmine dayalı modeli hesaplamaz ve tembel öğrenme algoritmaları ailesinin bir parçasıdır. KNN, bir girdi gözlemi ile veri kümesindeki farklı gözlemler arasındaki benzerliği hesaplayarak tam zamanında (anında) tahminler yapar.



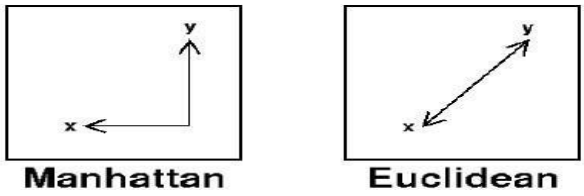
Yukarıdaki şekilde, kırmızı veya yeşil olarak sınıflandırılan veri noktalarında siyah bir veri noktası kırmızı ya da yeşil olabilecek iki sınıfı göstermektedir. Siyah noktanın ne olduğu KNN algoritması tarafından nasıl hesaplanır?

KNN algoritmaları, sınıflandırılacak veri noktasına en yakın komşu olan bir k sayısına karar verir.
K değeri 5 ise, o veri noktasına en yakın 5 Komşuyu arayacaktır. Bu örnekte, k = 4. KNN en yakın 4 komşuyu bulur. Bu veri noktasının bu komşulara yakın olması nedeniyle sadece bu sınıfa ait olacağı görülmektedir.

K-en yakın komşu sınıflandırıcı algoritmalarının basit versiyonu, en yakın komşu sınıfı bularak hedef etiketini tahmin etmektir. Sınıflandırılacak noktaya en yakın sınıf, Öklid mesafesi kullanılarak hesaplanır.

Mesafe, benzerliği ölçmek için kullanılır. İki örnek arasındaki mesafeyi ölçmenin birçok yolu vardır.

- Manhattan Distance, |X1-X2| + |Y1-Y2|
- Euclidean Distance, √((x1−x2)²)+√(y1−y2)²



Mesafe Ölçüm yöntemleri:

Minkowsky:

$$D(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{i=1}^{m} |x_i - y_i|^r\right)^{1/r}$$

Euclidean:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Manhattan / city-block:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} |x_i - y_i|$$

Camberra:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:

$$D(\boldsymbol{x},\boldsymbol{y}) = \max_{i=1}^{m} |x_i - y_i|$$

Quadratic:

$$D(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}-\boldsymbol{y})^T Q(\boldsymbol{x}-\boldsymbol{y}) = \sum_{j=1}^{m}\left(\sum_{i=1}^{m}(x_i - y_i)q_{ji}\right)(x_j - y_j)$$

Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:

$$D(\boldsymbol{x},\boldsymbol{y}) = [\det V]^{1/m}(\boldsymbol{x}-\boldsymbol{y})^{\mathrm{T}} V^{-1}(\boldsymbol{x}-\boldsymbol{y})$$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_{i=1}^{m}(x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m}(x_i - \overline{x_i})^2 \sum_{i=1}^{m}(y_i - \overline{y_i})^2}}$$

$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{1}{sum_i}\left(\frac{x_i}{size_{\boldsymbol{x}}} - \frac{y_i}{size_{\boldsymbol{y}}}\right)^2$$

sum_i is the sum of all values for attribute i occuring in the training set, and $size_{\boldsymbol{x}}$ is the sum of all values in the vector $\boldsymbol{x}$.

Kendall's Rank Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{2}{n(n-1)}\sum_{i=1}^{m}\sum_{j=1}^{i-1}\mathrm{sign}(x_i - x_j)\mathrm{sign}(y_i - y_j)$$

$\mathrm{sign}(x)$=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

Figure 1. Equations of selected distance functions.
($\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of m attribute values).

Mesafe ölçüsünün de yalnızca sürekli değişkenler için geçerli olduğu unutulmamalıdır. Kategorik değişkenler durumunda Hamming mesafesi kullanılmalıdır. Veri setinde sayısal ve kategorik değişkenlerin bir karışımı olduğunda, 0 ile 1 arasındaki sayısal değişkenlerin standardizasyonu konusunu da gündeme getirir.

Hamming Distance

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

K'nin önemi nedir?

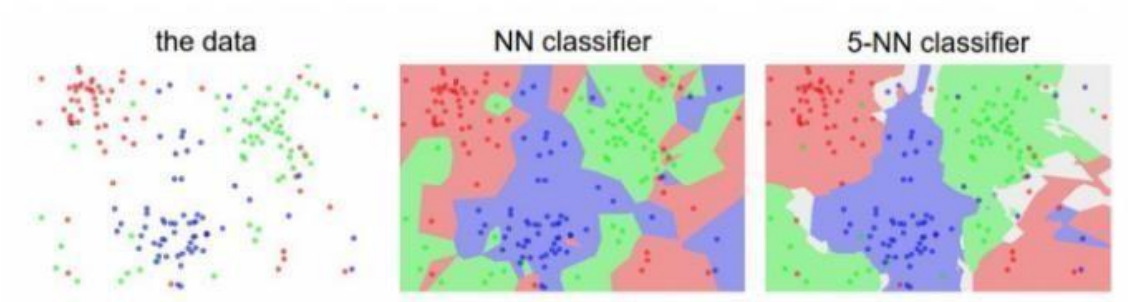
K değeri büyüdükçe tahmine duyulan güveni artırır. Öte yandan K çok büyük bir değere sahipse, kararlar çarpık olabilir.

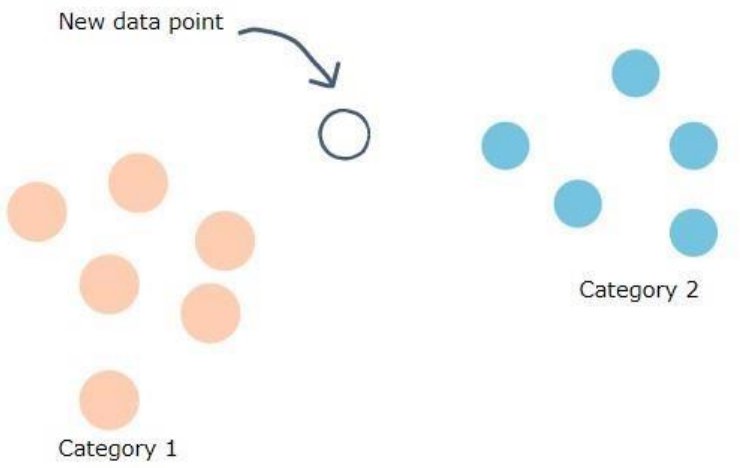
K nasıl seçilir?

Algoritma, yeni bir veri noktasının diğer tüm eğitim veri noktalarına olan mesafesini hesaplar. Mesafe herhangi bir türde olabilir, örneğin Öklid, Manhattan, vb. Algoritma daha sonra k'ye en yakın veri noktalarını seçer, burada k herhangi bir tam sayı olabilir. Sayısal değerlerin hangi özelliği temsil ettiğine bakılmaksızın, seçimini diğer veri noktalarına yakınlığına göre yapar. Son olarak, veri noktasını benzer veri noktalarının bulunduğu sınıfa atar.

Seçilen veri kümesine uyan K değerini seçmek için, KNN algoritması farklı K değerleri ile defalarca çalıştırılr. Sonra, algoritma, yeni değerler için hassas tahminler yapma yeteneğini korurken karşılaşılan hata sayısını azaltan K'yi seçilir.

- K'ye karar vermek, K-en yakın Komşular'ın en kritik kısmıdır.
- K değeri küçükse, gürültü sonuca daha fazla bağımlı olacaktır. Bu gibi durumlarda modelin aşırı uyumu çok fazladır.
- K'nin değeri ne kadar büyükse, KNN'nin arkasındaki prensibi yok edecektir.
- Çapraz doğrulamayı kullanarak K'nin optimum değeri bulunabilir.



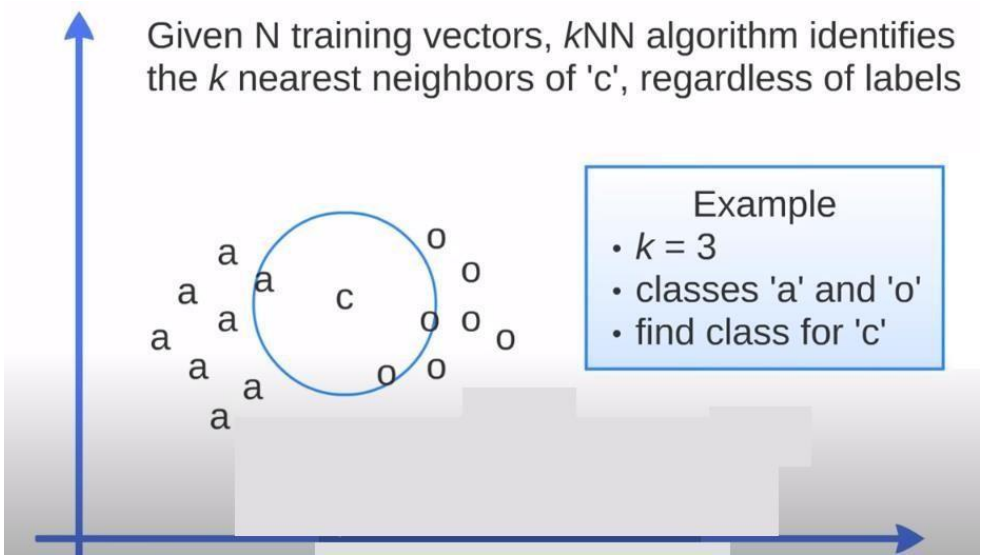


KNN algoritması sözde kod uygulaması

- İstenilen verileri yüklenir.
- k parametresi belirlenir. Bu parametre verilen bir noktaya en yakın komşuların sayısıdır. Örneğin: k=2 olsun. Bu durumda en yakın 2 komşuya göre sınıflandırma yapılacaktır.
- Örnek veri setine katılacak olan yeni verinin, mevcut verilere göre uzaklığı tek tek hesaplanır. İlgili uzaklık fonksiyonları yardımıyla.
- İlgili uzaklılardan en yakın k komşu ele alınır. Öznitelik değerlerine göre k komşu veya komşuların sınıfına atanır.
- Seçilen sınıf, tahmin edilmesi beklenen gözlem değerinin sınıfı olarak kabul edilir. Yani yeni veri etiketlenmiş (label) olur.

KNN, anlaşılması oldukça basit bir algoritmadır. Bunun başlıca nedeni, tahmin yapabilmek için bir modele ihtiyaç duymamasıdır. Bunun tersi, tahminini yapabilmek için tüm gözlemlerini hafızasında tutması gerektiğidir. Bu nedenle, girdi veri kümesinin boyutuna dikkat etmeniz gerekir. Ayrıca, mesafenin hesaplanması için yöntemin seçimi ve komşuların K sayısı hemen belli olmayabilir. Kullanım durumunuz için tatmin edici bir sonuç elde etmek için birkaç kombinasyon denemeniz ve algoritmayı ayarlamanız gerekebilir.

Örnek:



	Girdi Öznitelikleri		Çıktı Özniteliği
Öğrencinin Adı	Sayfa Sayısı	Soru Sayısı	Başarı Durumu
Selin	2100	1500	BAŞARILI
Mert	2280	1600	BAŞARILI
Caner	800	400	BAŞARISIZ
Şenay	2400	1750	BAŞARILI
Efe	250	350	BAŞARISIZ
Burak	180	220	BAŞARISIZ
Esra	1800	1200	?

Öklid uzaklığı kullanılarak altı adet öğrencinin Esra adlı öğrenciye olan uzaklıkları:

Öğrencinin Adı	Sayfa Sayısı	Soru Sayısı	Başarı Durumu	Uzaklık Hesaplaması	Sonuç
Selin	2100	1500	BAŞARILI	$\sqrt{(2100-1800)^2+(1500-1200)^2}$	424.2
Mert	2280	1600	BAŞARILI	$\sqrt{(2280-1800)^2+(1600-1200)^2}$	624.8
Caner	800	400	BAŞARISIZ	$\sqrt{(800-1800)^2+(400-1200)^2}$	1280
Şenay	2400	1750	BAŞARILI	$\sqrt{(2400-1800)^2+(1750-1200)^2}$	813.9
Efe	250	350	BAŞARISIZ	$\sqrt{(250-1800)^2+(350-1200)^2}$	1767
Burak	180	220	BAŞARISIZ	$\sqrt{(180-1800)^2+(220-1200)^2}$	1893
Esra	1800	1200	?		

Örneğin K=3 seçersek ilk 3 öğrencinin başarı durumuna bakacağız.

Öğrencinin Adı	Başarı Durumu	Sonuç	Sıra No
Selin	BAŞARILI	424.2	1
Mert	BAŞARILI	624.8	2
Şenay	BAŞARILI	813.9	3
Caner	BAŞARISIZ	1280	4
Efe	BAŞARISIZ	1767	5
Burak	BAŞARISIZ	1893	6
Esra	?		

Sonuç BAŞARILI

Örnek:

KNN, eğitim seti yardımıyla veri noktasını belirli bir kategoriye sınıflandırmaya çalıştığımız parametrik olmayan denetimli bir öğrenme tekniğidir. Basit bir deyişle, tüm eğitim durumlarının bilgilerini yakalar ve yeni durumları benzerliğe göre sınıflandırır.

Yeni bir örnek (x) için, en benzer K durum (komşular) için tüm eğitim seti aranarak ve bu K durumlar için çıktı değişkeni özetlenerek tahminler yapılır. Sınıflandırmada bu, mod (veya en yaygın) sınıf değeridir.

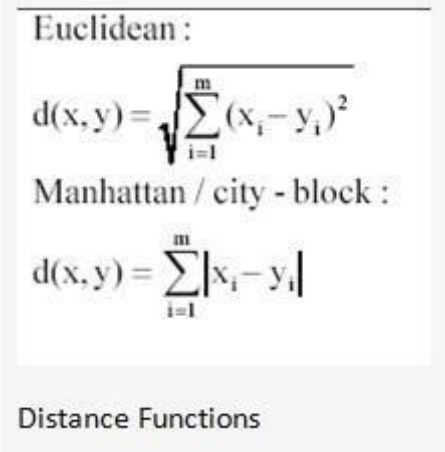
KNN algoritması nasıl çalışır?

Diyelim ki bazı müşterilerin boy, kilo ve tişört bedenleri var ve yeni bir müşterinin Tişört bedenini sadece sahip olduğumuz boy ve kilo bilgisine göre tahmin etmemiz gerekiyor. Boy, kilo ve tişört beden bilgilerini içeren veriler aşağıda gösterilmiştir.

Height (in cms)	Weight (in kgs)	T Shirt Size
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L

Adım 1: Uzaklık fonksiyonuna göre Benzerliği hesaplayın

Pek çok uzaklık fonksiyonu vardır ama en yaygın olarak kullanılan ölçü Ökliddir. Esas olarak veriler sürekli olduğunda kullanılır. Manhattan mesafesi de sürekli değişkenler için çok yaygındır.



Distance Functions

Mesafe ölçümü kullanma fikri, yeni örnek ve eğitim durumları arasındaki mesafeyi (benzerliği) bulmak ve ardından boy ve ağırlık açısından yeni müşteriye en yakın kmüşterileri bulmaktır.

'Monica' adlı yeni müşteri 161cm boyunda ve 61kg ağırlığındadır. İlk gözlem ile yeni gözlem (monica) arasındaki Öklid uzaklığı aşağıdaki gibidir:

=SQRT((161-158)^2+(61-58)^2)
Benzer şekilde, yeni vaka ile tüm eğitim vakalarının mesafesini hesaplayacağız ve mesafe açısından sıralamayı hesaplayacağız. En küçük mesafe değeri 1 olarak sıralanır ve en yakın komşu olarak kabul edilir.

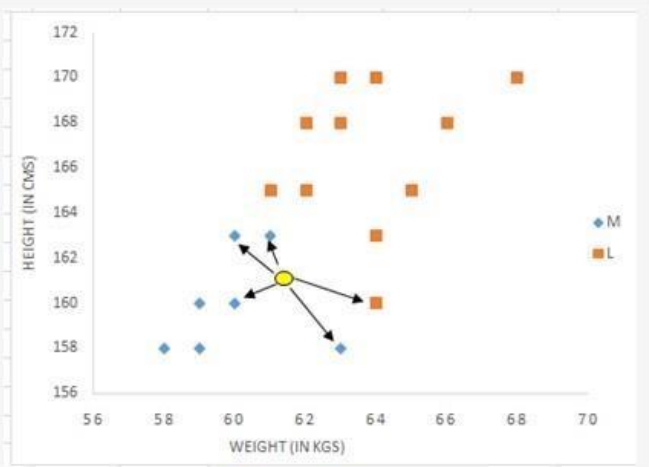
Step 2 : Find K-Nearest Neighbors
Let k be 5. Then the algorithm searches for the 5 customers closest to Monica, i.e. most similar to Monica in terms of attributes, and see what categories those 5 customers were in. If 4 of them had ‘Medium T shirt sizes’ and 1 had ‘Large T shirt size’ then your best guess for Monica is ‘Medium T shirt. See the calculation shown in the snapshot below –

2. Adım : K-En Yakın Komşuları Bulunması
K=5 olsun. Ardından algoritma, özellikler açısından Monica'ya en yakın, yani Monica'ya en çok benzeyen 5 müşteriyi arar ve bu 5 müşterinin hangi kategorilerde olduğunu görür. 4 tanesi 'Orta T shirt bedenleri' ve 1 tanesi ise 'Büyük T gömlek bedeni' vardı, o zaman Monica için en iyi tahmininiz 'Orta T gömlek. Aşağıdaki anlık görüntüde gösterilen hesaplamaya bakın

=SQRT((A21-A6)^2+(B21-B6)^2)

	A	B	C	D	E
1	Height (in cms)	Weight (in kgs)	T Shirt Size	Distance	
2	158	58	M	4.2	
3	158	59	M	3.6	
4	158	63	M	3.6	
5	160	59	**M**	2.2	**3**
6	160	60	**M**	1.4	**1**
7	163	60	**M**	2.2	**3**
8	163	61	**M**	2.0	**2**
9	160	64	**L**	3.2	**5**
10	163	64	L	3.6	
11	165	61	L	4.0	
12	165	62	L	4.1	
13	165	65	L	5.7	
14	168	62	L	7.1	
15	168	63	L	7.3	
16	168	66	L	8.6	
17	170	63	L	9.2	
18	170	64	L	9.5	
19	170	68	L	11.4	
20					
21	**161**	**61**			

Aşağıdaki grafikte ikili bağımlı değişken (T-shirt bedeni) mavi ve turuncu renkte görüntülenmektedir. 'Orta boy tişört' mavi renkte ve 'Büyük boy tişört' turuncu renktedir. Yeni müşteri bilgileri sarı daire içinde sergilenir. Dört mavi vurgulanmış veri noktası ve bir turuncu vurgulanmış veri noktası sarı daireye yakındır. bu nedenle yeni vaka için tahmin, Orta T-shirt boyutu olan mavi vurgulu veri noktasıdır.



KNN: Visual Representation

KNN Varsayımları:

1. Standardizasyon

Antrenman verilerindeki bağımsız değişkenler farklı birimlerde ölçüldüğünde, mesafeyi hesaplamadan önce değişkenleri standardize etmek önemlidir. Örneğin, bir değişken cm cinsinden yüksekliği temel alıyorsa ve diğeri kg cinsinden ağırlığı temel alıyorsa, uzunluk mesafe hesaplamasını daha fazla etkileyecektir. Bunları karşılaştırılabilir hale getirmek için, aşağıdaki yöntemlerden herhangi biriyle yapılabilecekleri standartlaştırmamız gerekir:

$$Xs = \frac{X - mean}{s.d.}$$

$$Xs = \frac{X - mean}{max - min}$$

$$Xs = \frac{X - min}{\max - min}$$

Standardization

Standardizasyondan önce yükseklik hakim olduğundan, standardizasyondan sonra 5. en yakın değer değişmiştir. Bu nedenle, K-en yakın komşu algoritmasını çalıştırmadan önce tahmin edicileri standart hale getirmek önemlidir.

	A	B	C	D	E
1	Height (in cms)	Weight (in kgs)	T Shirt Size	Distance	
2	-1.39	-1.64	M	1.3	
3	-1.39	-1.27	M	1.0	
4	-1.39	0.25	M	1.0	
5	-0.92	-1.27	**M**	0.8	**4**
6	-0.92	-0.89	**M**	0.4	**1**
7	-0.23	-0.89	**M**	0.6	**3**
8	-0.23	-0.51	**M**	0.5	**2**
9	-0.92	0.63	L	1.2	
10	-0.23	0.63	L	1.2	
11	0.23	-0.51	**L**	0.9	**5**
12	0.23	-0.13	L	1.0	
13	0.23	1.01	L	1.8	
14	0.92	-0.13	L	1.7	
15	0.92	0.25	L	1.8	
16	0.92	1.39	L	2.5	
17	1.39	0.25	L	2.2	
18	1.39	0.63	L	2.4	
19	1.39	2.15	L	3.4	
20					
21	**-0.7**	**-0.5**			

Knn after standardization

2. Aykırı Değer

Düşük k-değeri aykırı değerlere duyarlıdır ve daha yüksek bir K-değeri, daha fazla seçmenin tahmine karar vereceğini düşündüğü için aykırı değerlere karşı daha dirençlidir.

KNN neden parametrik değildir?
Parametrik olmayan, temel alınan veri dağılımı üzerinde herhangi bir varsayımda bulunmamak anlamına gelir. Parametrik olmayan yöntemler, modelde sabit sayıda parametreye sahip değildir. Benzer şekilde KNN'de model parametreleri aslında eğitim veri seti ile birlikte büyür - her eğitim durumunu modelde bir "parametre" olarak düşünebilirsiniz.

KNN ve K-ortalama: Birçok insan bu iki istatistiksel teknik - K-ortalama ve K-en yakın komşu arasında kafa karıştırır. Aşağıdaki farklardan bazılarına bakın:

- K-ortalama denetimsiz bir öğrenme tekniğidir (bağımlı değişken yoktur), KNN denetimli bir öğrenme algoritmasıdır (bağımlı değişken vardır)
- K-ortalama, veri noktalarını her kümedeki noktalar birbirine yakın olacak şekilde Kkümelerine ayırmaya çalışan bir kümeleme tekniğidir, K-en yakın komşu ise bir noktanın sınıflandırmasını belirlemeye çalışır, K sınıflandırmasını en yakın noktalara birleştirmeye çalışır.

KNN regresyon için kullanılabilir mi?
Evet, regresyon için K-en yakın komşu kullanılabilir. Başka bir deyişle, bağımlı değişken sürekli olduğunda K-en yakın komşu algoritması uygulanabilir. Bu durumda, tahmin edilen değer, en yakın k komşusunun değerlerinin ortalamasıdır.

KNN'nin Artıları ve Eksileri Artıları:

- Anlaması kolay
- Veriler hakkında varsayım yok
- Hem sınıflandırma hem de regresyona uygulanabilir
- Çok sınıflı problemlerde kolayca çalışır

Eksileri:

- Yoğun Bellek / Hesaplama açısından pahalı
- Veri ölçeğine duyarlı
- Nadir olay (çarpık) hedef değişkeni üzerinde iyi çalışmıyor
- Çok sayıda bağımsız değişken olduğunda mücadele
- Herhangi bir problem için, küçük bir k değeri, tahminlerde büyük bir varyansa yol açacaktır. Alternatif olarak, k değerini büyük bir değere ayarlamak, büyük bir model yanlılığına yol açabilir.

KNN'de kategorik değişkenler nasıl ele alınır?

Kategorik bir değişkenden kukla değişkenler oluşturun ve orijinal kategorik değişken yerine bunları dahil edin. Regresyondan farklı olarak, (k-1) yerine k kukla oluşturun. Örneğin, "Departman" adlı kategorik bir değişkenin 5 benzersiz düzeyi/kategorisi vardır. Böylece 5 kukla değişken oluşturacağız. Her kukla değişkenin departmanına karşı 1 ve 0'a sahiptir.

En iyi K değeri nasıl bulunur?
Çapraz doğrulama, optimal K değerini bulmanın akıllı bir yoludur. Model oluşturma sürecinden eğitim setinin bir alt kümesini dışarıda tutarak doğrulama hata oranını tahmin eder.

Çapraz doğrulama (diyelim ki 10 kat doğrulama), eğitim setinin rastgele olarak yaklaşık eşit büyüklükte 10 gruba veya katlara bölünmesini içerir. Verilerin %90'ı modeli eğitmek için, kalan %10'u ise onu doğrulamak için kullanılır. Yanlış sınıflandırma oranı daha sonra %10 doğrulama verisi üzerinden hesaplanır. Bu prosedür 10 kez tekrarlanır. Farklı gözlem grupları, 10 defadan her biri bir doğrulama seti olarak ele alınır. Daha sonra ortalaması alınan doğrulama hatasının 10 tahminiyle sonuçlanır.

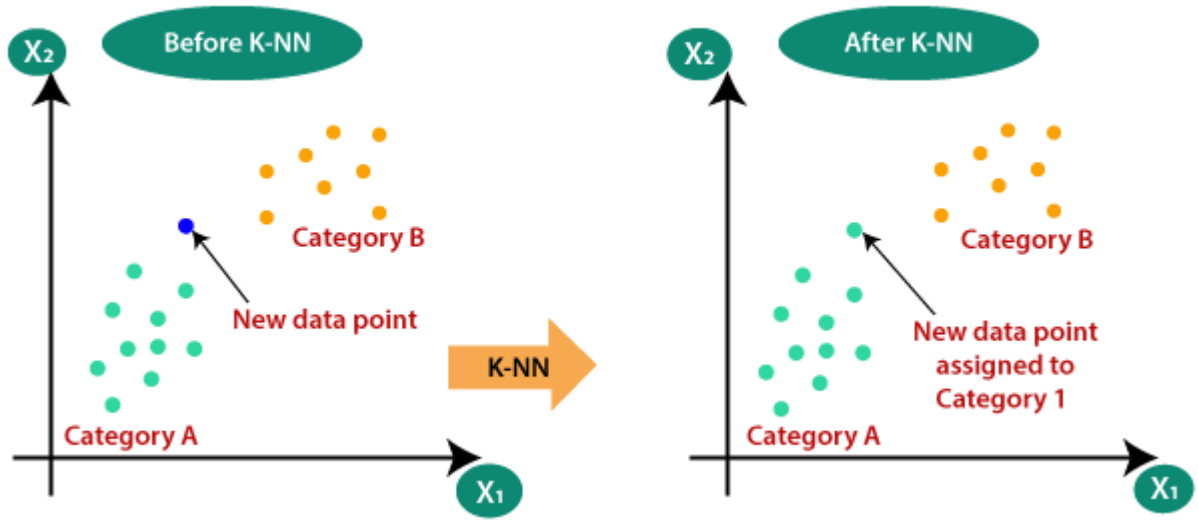
K-Nearest Neighbor(KNN) Algorithm for Machine Learning ○ K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem,

we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



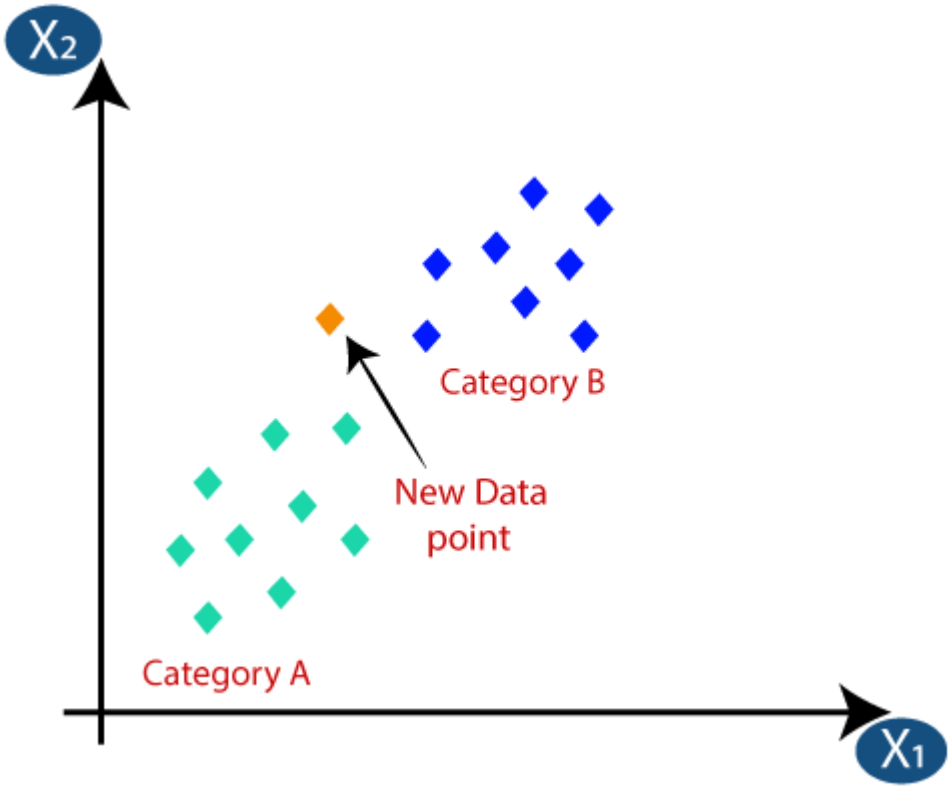
How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm: ○

Step1: Select the number K of the neighbors ○ **Step-2:** Calculate the Euclidean distance of **K number of neighbors** ○ **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.

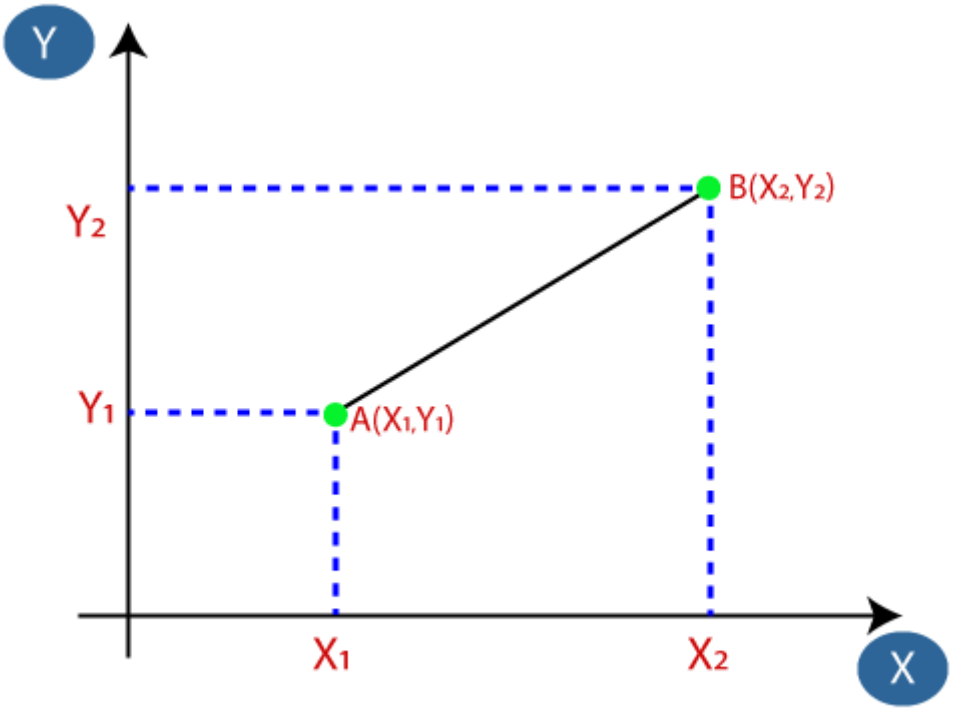
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



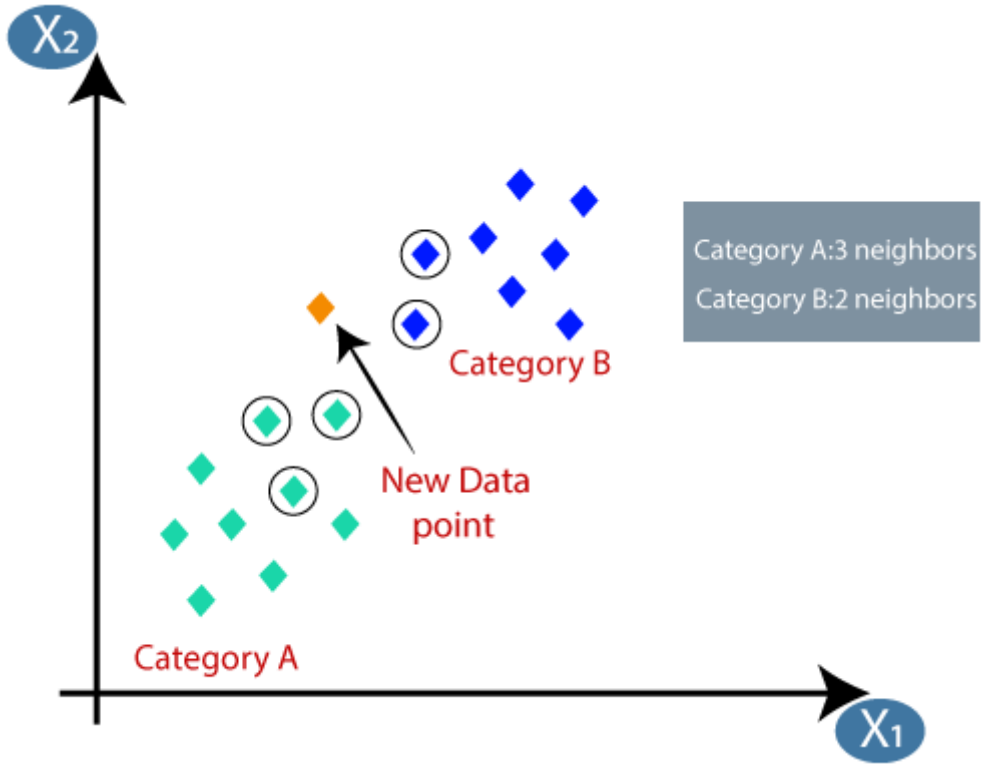
Firstly, we will choose the number of neighbors, so we will choose the k=5.

- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

9.3M

196

HTML Tutorial

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

Python implementation of the KNN algorithm

To do the Python implementation of the K-NN algorithm, we will use the same problem and dataset which we have used in Logistic Regression. But here we will improve the performance of the model. Below is the problem description:

Problem for K-NN Algorithm: There is a Car manufacturer company that has manufactured a new SUV car. The company wants to give the ads to the users who are interested in buying that SUV. So for this problem, we have a dataset that contains multiple user's information through the social network. The dataset contains lots of information but the **Estimated Salary** and **Age** we will consider for the independent variable and the **Purchased variable** is for the dependent variable. Below is the dataset:

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

Steps to implement the K-NN algorithm:

- Data Pre-processing step
- Fitting the K-NN algorithm to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result.

Data Pre-Processing Step:

The Data Pre-processing step will remain exactly the same as Logistic Regression. Below is the code for it:

```
1. # importing libraries
2. import numpy as nm
3. import matplotlib.pyplot as mtp
```

```
4. import pandas as pd
5. #importing datasets
6. data_set= pd.read_csv('user_data.csv')
7. #Extracting Independent and dependent Variable
8. x= data_set.iloc[:, [2,3]].values
9. y= data_set.iloc[:, 4].values
10. # Splitting the dataset into training and test set.
11. from sklearn.model_selection import train_test_split
12. x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0
    )
13. #feature Scaling
14. from sklearn.preprocessing import StandardScaler
15. st_x= StandardScaler()
16. x_train= st_x.fit_transform(x_train)
17. x_test= st_x.transform(x_test)
```

By executing the above code, our dataset is imported to our program and well preprocessed. After feature scaling our test dataset will look like:

From the above output image, we can see that our data is successfully scaled. ○ **Fitting K-NN classifier to the Training data:**

Now we will fit the K-NN classifier to the training data. To do this we will import the **KNeighborsClassifier** class of **Sklearn Neighbors** library. After importing the class, we will create the **Classifier** object of the class. The Parameter of this class will be ○ **n_neighbors:** To define the required neighbors of the algorithm. Usually, it takes 5.

- **metric='minkowski':** This is the default parameter and it decides the distance between the points.
- **p=2:** It is equivalent to the standard Euclidean metric.

And then we will fit the classifier to the training data. Below is the code for it:

```
1. #Fitting K-NN classifier to the training set
2. from sklearn.neighbors import KNeighborsClassifier
3. classifier= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 ) 4. classifier.fit(x_train, y_train)
```

Output: By executing the above code, we will get the output as:

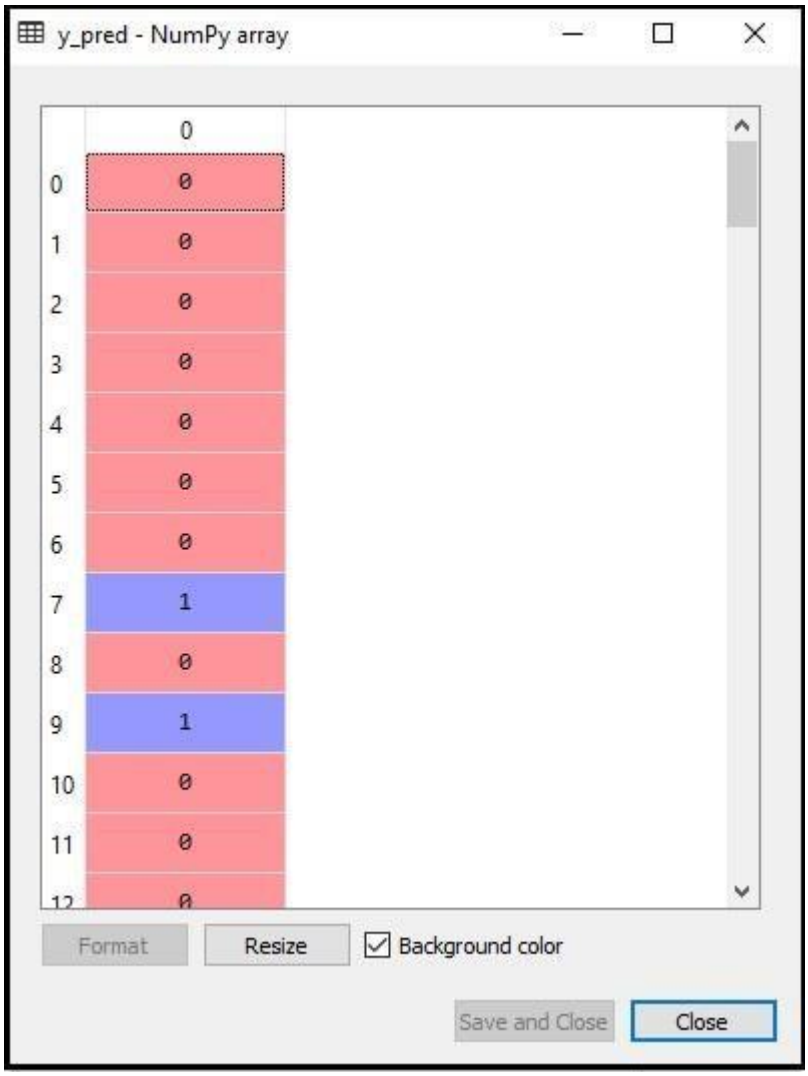
Out[10]:

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski', metric_params=None, n_jobs=None, n_neighbors=5, p=2, weights='uniform') ○ **Predicting the Test Result:** To predict the test set result, we will create a **y_pred** vector as we did in Logistic Regression. Below is the code for it:

```
1. #Predicting the test set result
2. y_pred= classifier.predict(x_test)
```

Output:

The output for the above code will be:



- **Creating the Confusion Matrix:**

  Now we will create the Confusion Matrix for our K-NN model to see the accuracy of the classifier. Below is the code for it:

```
1. #Creating the Confusion matrix
2. from sklearn.metrics import confusion_matrix
3. cm= confusion_matrix(y_test, y_pred)
```

In above code, we have imported the confusion_matrix function and called it using the variable cm.

Output: By executing the above code, we will get the matrix as below:

In the above image, we can see there are 64+29= 93 correct predictions and 3+4= 7 incorrect predictions, whereas, in Logistic Regression, there were 11 incorrect predictions. So we can say that the performance of the model is improved by using the K-NN algorithm. ○ **Visualizing the Training set result:**

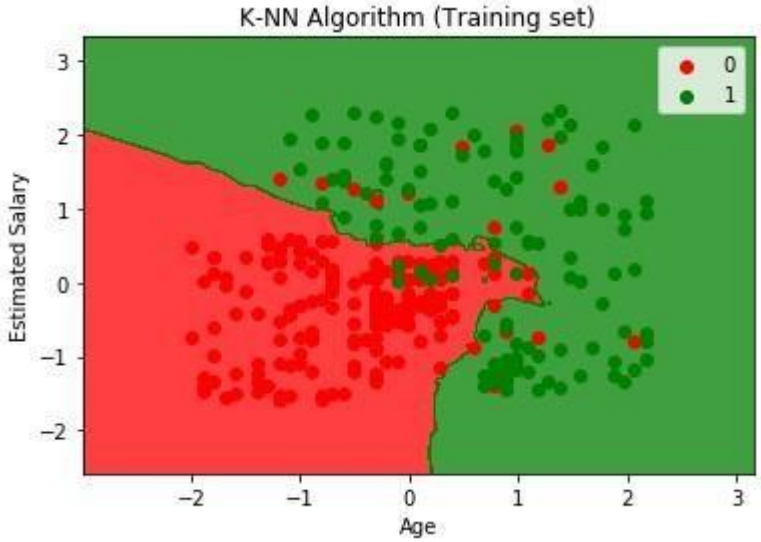
Now, we will visualize the training set result for K-NN model. The code will remain same as we did in Logistic Regression, except the name of the graph. Below is the code for it:

```
1. #Visulaizing the trianing set result
2. from matplotlib.colors import ListedColormap
3. x_set, y_set = x_train, y_train
4. x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() -
   1, stop = x_set[:, 0].max() + 1, step  =0.01),
5. nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
6. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.
   shape),
7. alpha = 0.75, cmap = ListedColormap(('red','green' ))) 8. mtp.xlim(x1.min(),
   x1.max()) 9. mtp.ylim(x2.min(), x2.max())
10.     for i, j in enumerate(nm.unique(y_set)):
11.     mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
12.     c = ListedColormap(('red', 'green'))(i), label = j)
13.     mtp.title('K-NN Algorithm (Training set)')
14.     mtp.xlabel('Age')
15.     mtp.ylabel('Estimated Salary')
```

16. mtp.legend()
17. mtp.show()

Output:

By executing the above code, we will get the below graph:



The output graph is different from the graph which we have occurred in Logistic Regression. It can be understood in the below points: ○ As we can see the graph is showing the red point and green points. The green points are for Purchased(1) and Red Points for not Purchased(0) variable.

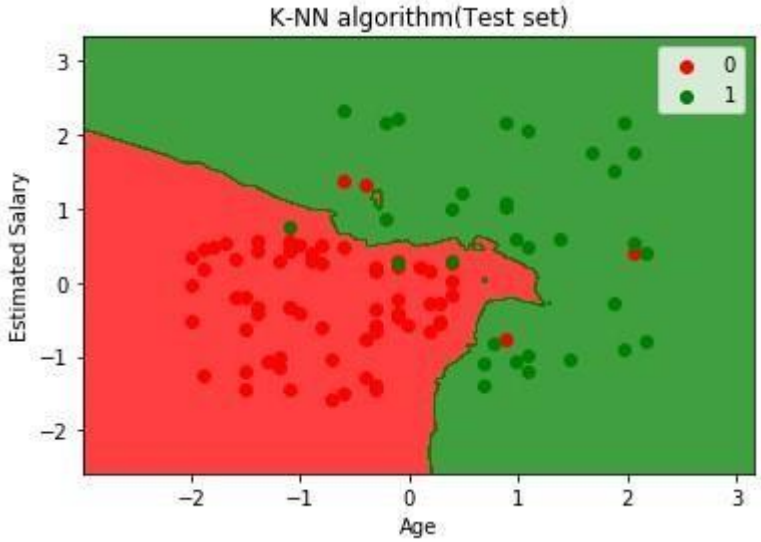
- The graph is showing an irregular boundary instead of showing any straight line or any curve because it is a K-NN algorithm, i.e., finding the nearest neighbor. ○ The graph has classified users in the correct categories as most of the users who didn't buy the SUV are in the red region and users who bought the SUV are in the green region.
- The graph is showing good result but still, there are some green points in the red region and red points in the green region. But this is no big issue as by doing this model is prevented from overfitting issues.
- Hence our model is well trained. ○ **Visualizing the Test set result:**

After the training of the model, we will now test the result by putting a new dataset, i.e., Test dataset. Code remains the same except some minor changes: such as **x_train and y_train** will be replaced by **x_test and y_test**. Below is the code for it:

1. #Visualizing the test set result
2. from matplotlib.colors **import** ListedColormap
3. x_set, y_set = x_test, y_test
4. x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() -
   * 1, stop = x_set[:, 0].max() + 1, step =0.01),
5. nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
6. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.

shape),

7. alpha = 0.75, cmap = ListedColormap(('red','green')))
8. mtp.xlim(x1.min(), x1.max()) 9. mtp.ylim(x2.min(), x2.max())
10. **for** i, j in enumerate(nm.unique(y_set)):
11. mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
12. c = ListedColormap(('red', 'green'))(i), label = j)
13. mtp.title('K-NN algorithm(Test set)')
14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show() **Output:**



The above graph is showing the output for the test data set. As we can see in the graph, the predicted output is well good as most of the red points are in the red region and most of the green points are in the green region.

However, there are few green points in the red region and a few red points in the green region. So these are the incorrect observations that we have observed in the confusion matrix(7 Incorrect output).

Örnek:

4 sınıfa ayrılmış askerlere boy, ağırlık ve kafa ölçülerine göre kask dağıtılmaktadır. Yeni bir asker girişi yapılmış. K=8 seçilmiş. KNN yakınlık ölçütleri: G1'e yakınlık 4, G2'ye yakınlık 1, G3'e yakınlık 2 adet, G4'e yakınlık 1 adet ise yeni giriş yapan asker hangi gruptandır.

P(G1)=4/8
P(G2)=1/8
P(G3)=2/8
P(G4)=1/8

O halde asker grup-1'e aittir.