Veri Madenciliği

Proje

***1.Giriş***

**Bu projede veri madenciliği sınıflandırma algoritmaları kullanılarak iris veri seti üzerinde çalışma yapılmıştır. Bu projede veri seti üzerinde ön işlemler sonrasında sınıflandırma algoritmalarından lojistik regresyon, k en yakın komşu, süper vektör makine, naif bayes , karar ağacı ve rassal orman kullanılarak projeye uygun modellin seçilmesi ve seçilme aşamasında bu algoritmaların kıyaslanması amaçlanmıştır.**

***2.Veri Detayları ve Modeller***

**Veriler , Bilkav sitesi üzerinden alınmıştır. Verinin sütunları sepal lenght(Çanak yaprağı uzunluğu),** **sepal width(Çanak yaprağı genişilği),** **petal lenght(Taç yaprağı uzunluğu),petal width(Taç yaprağı genişliği) ve İris’ten ve satır ise 151 veriden oluşmaktadır. Veriler hazır veri seti olup kategorik veriler içerdiğinden sınıflandırma algoritmaları kullanılmasına ve karşılaştırılmasına uygundur.**

**Öncellikle bu projede problemim için hangi modelin seçileceği veya seçilen modelin problemde artı ve eksileri nelerdir? Gibi sorulara cevaplar bulunması gerekiyor. Tabi bu sorunun cevabı birçok parametreye bağlı örneğin seçilen problemin doğrusal mı yoksa doğrusal olmayan özellikte mi olduğunu anlamamız ve buna göre hareket etmemiz gerekir. Eğer problem doğrusal ise sınıflandırma algoritmalarından lojistik regresyon ya da süper vektör makinesi kullanılabiliriz. Eğer lineer değil ise K-nn, naif bayes gibi algoritmaları kullanabiliriz. Tabi bu algoritmaların içerisinde olan parametrelerde modelimizin veriyi öğrendikten sonra sonuç döndürmesinde oldukça etkili örneğin K-nn’ i ele alalım k en yakın komşuyu 3 ve 3’ten küçük için daha doğru sonuçlar üretirken 3 ve üstü komşularda sonucun doğruluk değerinde azalma oluyor, bunları problemimizi anlatırken de anlatmaya çalışacağım.**

***3.Veri ön işleme adımları ve veri görselleştirme***

**Verilerin sınıflandırma algoritmaları kullanılmadan önce, veri ön işleme adımlarından geçmesi gerekiyordu bu adımlar sırasıyla; veri seçme ,veri kümesinin test ve veri olarak bölünmesi, verileri ölçeklendirme.**

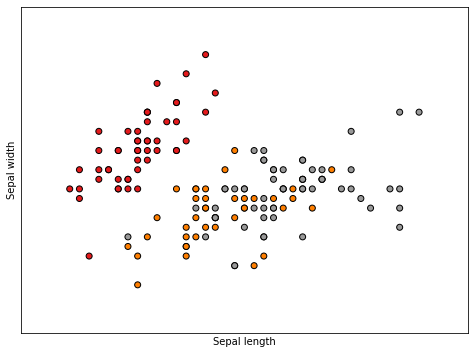
**Bu işlemler sırasında şunlar yapıldı;**

* **Öncellikle veriler excel’de xsl türünde kaydedildi ve çalıştığımız python dosyasıyla aynı klasörde kaydedildi. Ardından pandas kütüphanesi import edildikten sonra veriler çağrıldı.**
* **Veriler makine öğrenimi için bağımlı ve bağımsız değişkenler olmak üzere ikiye ayırıldı. Bağımsız değişkenler 1’den 4 sütün’ a kadar olan özellikler iken bağımlı değişkenimiz için sadece iris sütünü ele alındı.**
* **Veriler python içim makine öğrenim modülü olan sklearn.model.selection altında train.test.spilt import edildi ve verilerin rastgele %33 lük kısmı makine öğrenimi için ayrıldı ve bu kısım x\_test ve y\_test’e aktarıldı kalan %67 lik kısım ise x\_train ve y\_train’e aktarıldı tabi burada amaç verilerin ayrıldıktan sonra %33’lük kısmının öğrenip kalan eğitim verileri üzerinde kullanılacak algoritmaya göre test kümesindeki başarıyı ölçmek.**
* **Ardından Sclearnpreprocessing altında Standartscaler kütüphanesi import edildi ve fit\_transform ile xtrain ve xtest için -1 ile 1 aralığında bir normalizasyon uygulandı ve bunlarda X\_train ve X\_test e aktarıldı.**

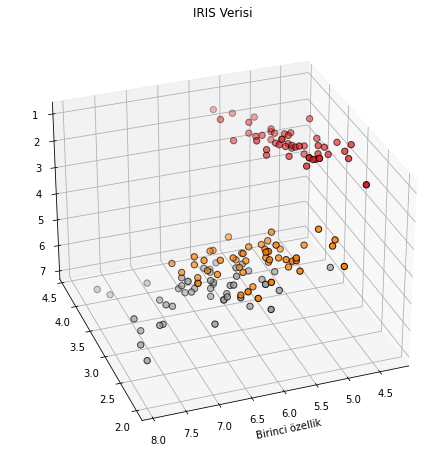
**Tabi burada bazı önemli durumlar söz konusu burada test edilecek verileri bölerken kullandığım yöntem split validation yöntemi. Bu yöntem veri kümemizi rastgele bir şekilde belirli bir yüzdelikle bölmekte ve bunları test ve eğitim kümesi olarak ayırmakta bu da şu gibi durumların oluşmasını sağlamakta örneğin genel olarak split validation da test için seçilen verilerin 2/3 eğitim, 1/3 test olarak rastgele ayrılır ama test verilerindeki verilerin ne kadar modelle uyumlu veriler olduğu aslında sonucu etkileyen etkenler arasında. Bu yüzde de k fold cross validation gibi bir değerlendirme yöntemi kullanmak daha doğru sonuçların oluşmasına katkı sunacaktır. Peki cross validation nedir?**

**Örnek vermek gerekirse cross validition şu şekilde çalışır;**

**Bir veri kümesini önce 1/4’ü test 3/4' ü eğitim için ayrılır daha sonra veri setinin 3/4' lük kısmından yeni bir test verisi çıkarılır ve yine 1/4’ü test 3/4' ü eğitim için ayrılır, böylece bütün veri kümesi için her birinin hem test hem de eğitim aşamasına girdiği bir döngü haline gelmiş olur ve bütün veri seti için rastgelelik ortadan kalkmış olur.**



**1.1. 2 Boyutlu iris verisi**



**1.2. 3 Boyutlu iris verisi**

**Verileri görselleştirme aşamasında mpl\_toolkits\_mplot3d altında Axes 3d python kütüphansi kullanılmıştır ve 3D boyutlu çizim sağlanmıştır. Tabi iris veri kümemiz içerisinde 4 özellik vardır yani 4 boyutun olması gerekir. 4 boyut çizecek bir teknoloji olmadığından bu işlemde x koordinatı için birinci özellik, y koordinatı için ikinci özellik ve z koordinatı için ise üçüncü özellik kullanılmıştır.**

**Bu görselleri yorumlamak gerekirse 2 boyutlu görselde şu yorumu yapabiliriz kırmızı veri kümesi ayrışmışken turuncu ve mavi veri kümesi iç içe geçmiş durumda tabi 3d görselde daha net bir şekilde ortaya çıkıyor. Tabi birazdan değineceğim ama buradaki ayrışma ve verilerin iç içe geçme durumu kullanacağımız algoritmaların hata ve doğruluk oranlarını da anlamamıza yardımcı olacak. Yine 2 boyutlu veri kümemize baktığımda lineer olarak ayırıldığını görüyorum tabi net olmamakla birlikte burada lineer algoritma kullanımı sonucu hata oranlarının düşük çıkacağını düşünüyorum çünkü veri kümesi kırmızı veriler net bir şekilde ayrılmış. Eğer daha iç içe bir veri kümesinden bahsetseydik burada random forest,gaussian process ve Rbf Svm kullanabilirdik. Burda kısaca K-nn algoritmasına da değinmek gerekir çünkü çok güçlü bir algoritma kendisi veri kümesi hangi dağılımda olursa olsun çok başarılı sonuçlar çıkartabiliyor kendisi, veri kümesinin lineer olup olmamasının bir öneminin olmadığı algoritmalardan bir tanesi.**

***4.Sınıflama Modelleri***

*4.1.Lojistik regresyon*

**Lojistik regresyon, bir sonucu belirleyen bir veya daha fazla bağımsız değişken bulunan bir veri kümesini analiz etmek için kullanılan istatistiksel bir yöntemdir. Sonuç, ikili bir değişkenle ölçülür (yalnızca iki olası sonuç vardır).  
Lojistik regresyonda, bağımlı değişken ikili veya ikili, yani yalnızca 1 (DOĞRU, başarı, hamile vb.) Veya 0 (YANLıŞ, hata, gebe olmayan vb.) Olarak kodlanmış verileri içeriyor.**

**Lojistik regresyonun amacı,  iki yönlü karakteristiği (bağımlı değişken = yanıt veya sonuç değişkeni) ile ilgili bir dizi bağımsız (öngörücü veya açıklayıcı) değişken arasındaki ilişkiyi tanımlamak için en uygun (henüz biyolojik olarak makul) modeli bulmaktır. Lojistik regresyon, ilgi karakteristiklerinin varlığının olasılığını logit dönüşümünü tahmin etmek için bir formülün katsayılarını (ve standart hatalarını ve önem seviyelerini) üretir:**

****

**Burada p, karakteristik özelliğinin var olma olasılığıdır.**

**formul_logit2**

**ve**

**formul_logit3**

**Karekök hataların toplamını en aza indirgeyen parametreleri seçmek yerine (sıradan regresyon gibi), lojistik regresyonda tahmin, örnek değerlerin gözlem olasılığını en yükseğe çıkaran parametreleri seçer.**

*4.2.K – En Yakın Komşu*

**KNN en basit anlamı ile içerisinde tahmin edilecek değerin bağımsız değişkenlerinin oluşturduğu vektörün en yakın komşularının hangi sınıfta yoğun olduğu bilgisi üzerinden sınıfını tahmin etmeye dayanır.**

**KNN (K-Nearest Neighbors) Algoritması iki temel değer üzerinden tahmin yapar;**

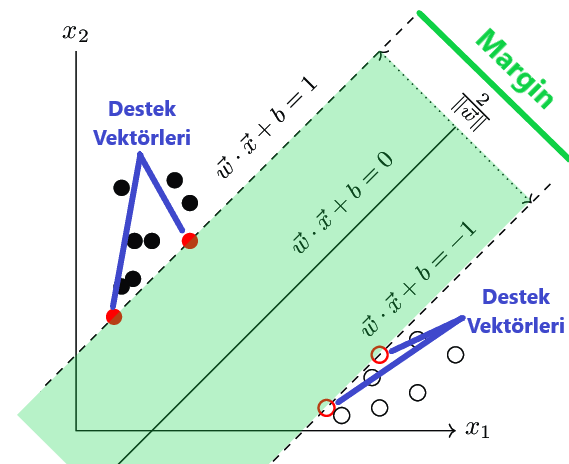
* **Distance (Uzaklık): Tahmin edilecek noktanın diğer noktalara uzaklığı hesaplanır. Bunun için Minkowski uzaklık hesaplama fonksiyonu kullanılır.**
* **K (komuşuluk sayısı): En yakın kaç komşu üzerinden hesaplama yapılacağını söyleriz. K değeri sonucu direkt etkileceyecektir. K 1 olursa overfit etme olasılığı çok yüksek olacaktır. Çok büyük olursada çok genel sonuçlar verecektir. Bu sebeple optimum K değerini tahmin etmek problemin asıl konusu olarak karşımızda durmaktadır. K değerinin önemini aşağıdaki grafik çok güzel bir şekilde göstermektedir. Eğer K=3 ( düz çizginin olduğu yer) seçersek sınıflandırma algoritması ? işareti ile gösterilen noktayı, kırmızı üçgen sınıfı olarak tanımlayacaktır. Fakat K=5 (kesikli çizginin olduğu alan) seçersek sınıflandırma algoritması, aynı noktayı mavi kare sınıfı olarak tanımlayacaktır.**

**KNN (K-Nearest Neighbors) Algoritması ile üretilmiş bir modelin modelin başarımını ölçmek için genel olarak kullanılan 3 adet indikatör vardır.**

* **Jaccard Index: Doğru tahmin kümesi ile gerçek değer kümesinin kesişim kümesinin bunların birleşim kümesine oranıdır.1 ile 0 arası değer alır. 1 en iyi başarım anlamına gelir.**
* **F1-Score: Confusion Matriks üzerinden hesaplanan Precission ve Recall değerlerinden hesaplanır. Pre=TP/(TP+FP) Rec=TP/(TP+FN) F1-Score= 2*((Pre*Rec)/(Pre+Rec)) 1 ile 0 arası değer alır. 1 en iyi başarım anlamına gelir.**
* **LogLoss: Logistic Regresyon sonunda tahminlerin olasılıkları üzerinden LogLoss değeri hesaplanır. 1 ile 0 arası değer alır. Yukarıdaki iki değerden farklı olarak 0 en iyi başarım anlamına gelir.**

*4.3.Super Vektor Machine*

**Destek Vektör Makineleri (Support Vector Machine) genellikle sınıflandırma problemlerinde kullanılan gözetimli öğrenme yöntemlerinden biridir. Bir düzlem üzerine yerleştirilmiş noktaları ayırmak için bir doğru çizer. Bu doğrunun, iki sınıfının noktaları için de maksimum uzaklıkta olmasını amaçlar. Karmaşık ama küçük ve orta ölçekteki veri setleri için uygundur.**

****

**Tabloda siyahlar ve beyazlar olmak üzere iki farklı sınıf var. Sınıflandırma problemlerindeki asıl amacımız gelecek verinin hangi sınıfta yer alacağını karar vermektir. Bu sınıflandırmayı yapabilmek için iki sınıfı ayıran bir doğru çizilir ve bu doğrunun ±1'i arasında kalan yeşil bölgeye Margin adı verilir. Margin ne kadar geniş ise iki veya daha fazla sınıf o kadar iyi ayrıştırılır.**

**Formüle bakacak olursak:**

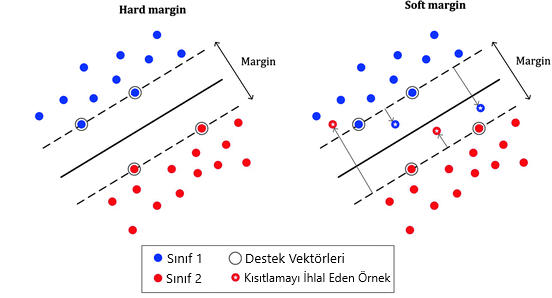
**metin, saat, anten içeren bir resim

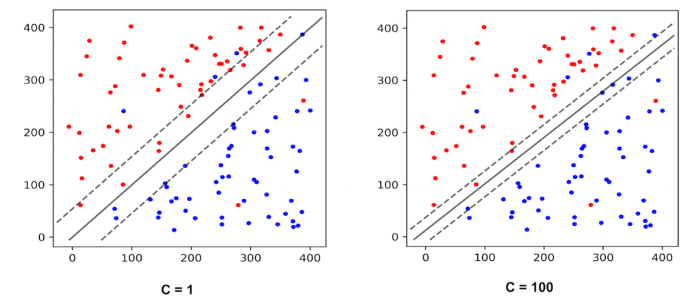
Açıklama otomatik olarak oluşturuldu**

**Aslında değişen pek bir şey yok. w; ağırlık vektörü (θ1), x; girdi vektörü, b; sapmadır (θ0). Yeni bir değer için çıkan sonuç 0'dan küçükse, beyaz noktalara daha yakın olacaktır. Tam tersi, çıkan sonuç 0'a eşit veya büyükse, bu durumda siyah noktalara daha yakın olacaktır.**

**Hard Margin vs Soft Margin**

**Marginimiz her zaman bu şekilde olmayabilir. Bazen örneklerimiz Margin bölgesine girebilir. Buna Soft Margin denir. Hard Margin, verimiz doğrusal olarak ayrılabiliyorsa çalışır ve aykırı değerlere karşı çok duyarlıdır. Bu yüzden bazı durumlarda Soft Margin’i tercih etmemiz gerekebilir.**

**İkisi arasındaki dengeyi SVM içerisindeki C hiperparametresi ile kontrol edebiliriz. C ne kadar büyükse Margin o kadar dardır.**

****

**Ayrıca model overfit olursa C’yi azaltmamız gerekir.**

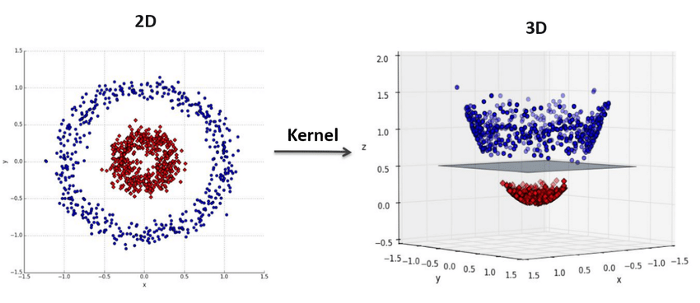
**Kernel Trick**

**Düşük boyutlar karmaşık veri setlerini açıklamada yeterli olmayabilir. Boyutu arttırsak işlemler artacağı için çok uzun sürer. İşte Kernel Trick burada devreye giriyor. Elimizdeki koordinatları belirli Kernel Fonksiyonları ile çarparak çok daha anlamlı hale getirebiliyoruz.**

**Bu yöntemlerden en çok kullanılan ikisini detaylı olarak açıklayacağım.**

***1-) Polynomial Kernel:***

**Bu yöntemde problemimizi çözmek için 2 boyuttan çıkıp 3 veya daha fazlası boyutta işlem yapıyormuş gibi hareket ediyoruz.**

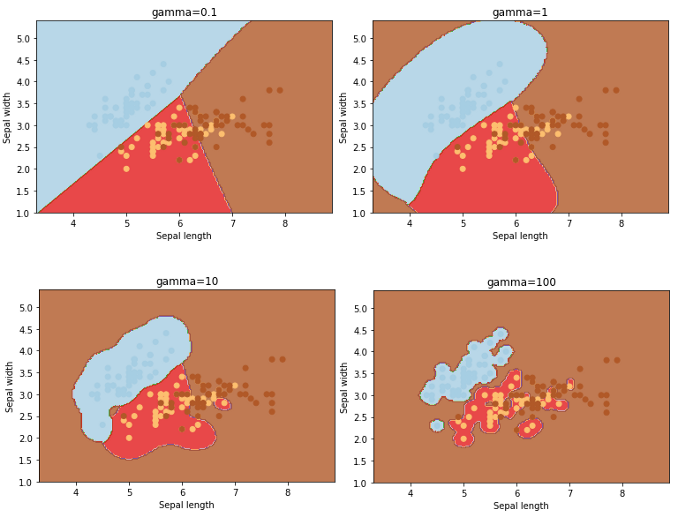
****

**Soldaki (2 boyut) dağılımı bir doğru ile sınıflandıramayız. Bunun için bu gibi problemlerde Polynomial Kernel’i kullanabiliriz. 3. boyutta işlem yaparken sınıflara ayırmak için doğru yerine bir düzlem kullanılırız ve çok daha düzgün bir şekilde sınıflandırabiliriz.**

**Modelimiz overfit olmuşsa derecesini düşürmeniz, underfit olmuşsa derecesini yükseltmeniz gerekir. Ayrıca coef0 hiperparametresiyle modelinizin yüksek dereceli denklemlerden ne kadar etkileneceğini ayarlayabilirsiniz(sadece ‘poly’ ve ‘sigmoid’ kernelda etkili olur).**

**2-) Gaussian RBF (Radial Basis Function) Kernel:**

**Anlaması biraz güç olabilir ama sonsuz boyuttaki Destek Vektör Makinelerini bulur ve her bir noktanın belirli bir noktaya ne kadar benzediğini normal dağılım ile hesaplar, ona göre sınıflandırır. Dağılımın genişliğini gamma hiperparametresi ile kontrol ederiz. *Gamma*ne kadar küçükse dağılım o kadar geniş olur. C hiperparametresindeki gibi, model overfit olmuşsa gamma değerini düşürmemiz, model underfit olmuşsa gamma değerini yükseltmemiz gerekir.**

****

**Veri setiniz aşırı büyük değilse genellikle RBF Kernel tercih edilir.**

*4.3.Naive Bayes*

**Naive Bayes sınıflandırıcı,**[**örüntü tanıma**](https://tr.wikipedia.org/wiki/%C3%96r%C3%BCnt%C3%BC_tan%C4%B1ma)**problemine ilk bakışta oldukça kısıtlayıcı görülen bir**[**önerme**](https://tr.wikipedia.org/wiki/%C3%96nerme)**ile kullanılabilen [olasılıksal](https://tr.wikipedia.org/wiki/Olas%C4%B1l%C4%B1k" \o "Olasılık) bir yaklaşımdır. Bu önerme, örüntü tanımada kullanılacak her bir tanımlayıcı**[**öznitelik**](https://tr.wikipedia.org/wiki/%C3%96znitelik_(makine_%C3%B6%C4%9Frenmesi))**ya da parametrenin**[**istatistik**](https://tr.wikipedia.org/wiki/%C4%B0statistik)**açıdan bağımsız olması gerekliliğidir. Her ne kadar bu önerme Naive Bayes sınıflandırıcının kullanım alanını kısıtlasa da istatistik bağımsızlık koşulu esnetilerek kullanıldığında da daha karmaşık**[**yapay sinir ağları**](https://tr.wikipedia.org/wiki/Yapay_sinir_a%C4%9Flar%C4%B1)**gibi metotlarla karşılaştırabilir sonuçlar vermektedir. Bir Naive Bayes sınıflandırıcı, her özniteliğin birbirinden koşulsal bağımsız olduğu ve öğrenilmek istenen kavramın tüm bu özniteliklere koşulsal bağlı olduğu bir [Bayes ağı](https://tr.wikipedia.org/wiki/Bayes_a%C4%9F%C4%B1" \o "Bayes ağı) olarak da düşünülebilir.**

**Naive Bayes sınıflandırıcısı Bayes teoreminin bağımsızlık önermesiyle basitleştirilmiş halidir. Bayes teoremi aşağıdaki denklemle ifade edilir;**

**P(A|B)=P(B|A)\P(A)/P(B)**

**P(A|B) ; B olayı gerçekleştiği durumda A olayının meydana gelme olasılığıdır.**

**P(B|A) ; A olayı gerçekleştiği durumda B olayının meydana gelme olasılığıdır.**

**P(A) ve P(B) ; A ve B olaylarının önsel olasılıklarıdır.**

**Burada önsel olasılık Bayes teoreminine öznellik katar. Diğer bir ifadeyle örneğin P(A) henüz elde veri toplanmadan A olayı hakkında sahip olunan bilgidir. Diğer taraftan P(B|A) ardıl olasılıktır çünkü veri toplandıktan sonra, A olayının gerçekleşmiş olduğu durumlarda B olayının gerçekleşme ihtimali hakkında bilgi verir.**

***5.Sınıflama Modellerinin Projedeki Kullanımı ve Sonuçları***

**Projede 5 sınıflandırma algoritmasına yer verilmiştir. Veri önişleme adımı sonrasında sırasıyla logistic regression ,Knn,Super vektör machine, naive bayes,decision tree ve random forest algoritmaları ile verilerimiz arasından oluşturduğumuz tahmin değerleri ve yine verilerimizdeki gerçek değerler karşılaştırıldı. Hangi algoritma daha başaralı hangi algoritma daha başarısız bunlar değerlendirildi.**

***Lojictic regression:***

**Lojistik regresyon için sklear\_linear\_model altında Logistic regression import edildikten sonra lojistik regresyon nesnesi Logr oluşturuldu ve fit edildi. Burada fit ile eğitim sağlanmış oldu ve ardından predict ile tahmin değerleri, oluşturduğumuz y\_pred ’e aktarıldı. Ardından oluşturduğumuz tahmin değerlerinin verilerimizdeki gerçek değerlere karşılaştırmak için Confusion matrix kullandı. Confusion matrix için Sklearn\_metrics altında confusion\_matrix import edildi ve confusion matrix’ten cm adında bir nesne oluşturuldu. Bu nesnede oluşturduğumuz tahmin değerlerimizin tutulduğu y\_pred ve gerçek değerlerimizin tutulduğu y\_test arasındaki uyumluluk test edildi ve ne kadarı doğru tahmin ne kadarı yanlış tahmin olduğu bastırıldı.**

**Confusion matrix tablosu Logistic regresion için aşağıda gösterildiği gibidir. Bu tabloyu yorumlamak gerekirse;**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N=50** | **Tahmin değeri:**  **İris-setosa** | **Tahmin değeri:**  **İris-versicolor** | **Tahmin değeri:**  **İris-virgina** |  |
| **Gerçek değeri:**  **İris-setosa** | **16** | **0** | **0** | **16** |
| **Gerçek değeri:**  **İris-versicolor** | **0** | **18** | **2** | **20** |
| **Gerçek değeri:**  **İris-virgina** | **0** | **1** | **13** | **14** |
|  | **16** | **19** | **15** |  |

**Bu tabloda doğru tahmin yapılan değerler Gerçek pozitif ve Gerçek negatiflerin toplamı: 16 + 18+ 13 =47**

**Yanlış tahmin yapılan değerler Yanlış pozitifler ve Yanlış negatiflerin**

**toplamı : 1+2 =3**

**Accuracy rate(Doğruluk oranı)=47/50=0.94**

**Misclassfication rate(Yanlışlık oranı) =1-0.94=0.06**

**Burada kullandığımız algoritma 0.94’ lük bir başarı oranı yakalamış ve bu algoritma için başaralı diyebiliriz tabi bu başarı etkileyen parametreler var bu algoritma için örneğin random\_state parametresi mevcut ve biz bunu 0 olarak aldık bu sayıyı değiştirdiğimiz taktirde başarı değişebilirdi veya lineer algoritma ya uygun olmayan bir veri yapımız olsaydı eğer bu başarı oranımızı daha düşük çıkartabilirdi. Tabi seçtiğim veri, veri madenciliğindeki birçok modelle ve birçok algoritmaya uygun bu yüzden göreceğiz ki diğer algoritmalarla da uygun bir şekilde çalışacaktır.**

***K-nn:***

**K-nn için sklear\_Neighbors altında KNeigborsClasssifier import edildikten sonra K-nn nesnesi için knn oluşturuldu.Knn nesnesi için en yakın komşu parametresi için K=1 alındı, uzaklık ölçme adına metric parametresi ‘Minkowski’ alındı ve fit edildi. Burada fit ile eğitim sağlanmış oldu ve ardından predict ile tahmin değerleri, oluşturduğumuz y\_pred ’e aktarıldı. Ardından oluşturduğumuz tahmin değerlerinin verilerimizdeki gerçek değerlere karşılaştırmak için Confusion matrix kullandı. Confusion matrix için Sklearn\_metrics altında confusion\_matrix import edildi ve confusion matrix’ten cm adında bir nesne oluşturuldu. Bu nesnede oluşturduğumuz tahmin değerlerimizin tutulduğu y\_pred ve gerçek değerlerimizin tutulduğu y\_test arasındaki uyumluluk test edildi ve ne kadarı doğru tahmin ne kadarı yanlış tahmin olduğu bastırıldı.**

**Confusion matrix tablosu K-nn Algoritması için aşağıda gösterildiği gibidir. Bu tabloyu yorumlamak gerekirse;**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N=50** | **Tahmin değeri:**  **İris-setosa** | **Tahmin değeri:**  **İris-versicolor** | **Tahmin değeri:**  **İris-virgina** |  |
| **Gerçek değeri:**  **İris-setosa** | **16** | **0** | **0** | **16** |
| **Gerçek değeri:**  **İris-versicolor** | **0** | **18** | **2** | **20** |
| **Gerçek değeri:**  **İris-virgina** | **0** | **1** | **13** | **14** |
|  | **16** | **19** | **15** |  |

**Bu tabloda doğru tahmin yapılan değerler Gerçek pozitif ve Gerçek negatiflerin toplamı: 16 + 18+ 13 =47**

**Yanlış tahmin yapılan değerler Yanlış pozitifler ve Yanlış negatiflerin**

**toplamı : 1+2 =3**

**Accuracy rate(Doğruluk oranı)=47/50=0.94**

**Misclassfication rate(Yanlışlık oranı) =1-0.94=0.06**

**Burada kullandığımız algoritma 0.94’ lük bir başarı oranı yakalamış ve bir önceki algoritmayla aynı ve bu algoritma için başaralı diyebiliriz. Zaten K-nn algoritması çok güçlü bir algoritma dolasıyla genelde verilerin doğrusal olup olmaması algoritmayı etkilemiyor bizim verimizde de başarı oranı yüksek bir sınıflandırma algoritması oldu. Bu algoritmada da birçok parametre mevcut örneğin k en yakın komşu bu veri kümesinde artığı zaman başarı oranımız düşüyor bu da algoritmayı daha az başaralı kılıyor veya aldığımız ölçüm metriği minkowski bu Öklid veya başka bir ölçüm metriği olduğunda yine başarı oranımız değişiyor.**

**Örnek vermek gerekirse k en yakın komşuyu 3 seçtiğimizde başarı oranımız artıyor ve yaklaşık %96’lık bir başarı yakalıyor fakat komşuyu 3 ve üstü olduğunda örnek 40 aldığımızda başarı bir anda %65 değerine ulaşıyor. Bu da komşunun doğrusal bir şekilde artığında başaranın da artacağı tezini çürütüyor.**

***Super Vector Machine:***

**Super Vector Machine için sklear\_Svm altında SVC import edildikten sonra Super Vector Machine nesnesi Svc oluşturuldu kernel parametresi için ‘poly’ seçildi ve fit edildi. Burada fit ile eğitim sağlanmış oldu ve ardından predict ile tahmin değerleri, oluşturduğumuz y\_pred ’e aktarıldı. Ardından oluşturduğumuz tahmin değerlerinin verilerimizdeki gerçek değerlere karşılaştırmak için Confusion matrix kullandı. Confusion matrix için Sklearn\_metrics altında confusion\_matrix import edildi ve confusion matrix’ten cm adında bir nesne oluşturuldu. Bu nesnede oluşturduğumuz tahmin değerlerimizin tutulduğu y\_pred ve gerçek değerlerimizin tutulduğu y\_test arasındaki uyumluluk test edildi ve ne kadarı doğru tahmin ne kadarı yanlış tahmin olduğu bastırıldı.**

**Confusion matrix tablosu Super Vector Machine için aşağıda gösterildiği gibidir. Bu tabloyu yorumlamak gerekirse;**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N=50** | **Tahmin değeri:**  **İris-setosa** | **Tahmin değeri:**  **İris-versicolor** | **Tahmin değeri:**  **İris-virgina** |  |
| **Gerçek değeri:**  **İris-setosa** | **16** | **0** | **0** | **16** |
| **Gerçek değeri:**  **İris-versicolor** | **0** | **19** | **4** | **23** |
| **Gerçek değeri:**  **İris-virgina** | **0** | **0** | **11** | **11** |
|  | **16** | **19** | **15** |  |

**Bu tabloda doğru tahmin yapılan değerler Gerçek pozitif ve Gerçek negatiflerin toplamı: 16 + 18+ 13 =47**

**Yanlış tahmin yapılan değerler Yanlış pozitifler ve Yanlış negatiflerin**

**toplamı : 1+2 =3**

**Accuracy rate(Doğruluk oranı)=46/50=0.92**

**Misclassfication rate(Yanlışlık oranı) =1-0.94=0.08**

**Burada kullandığımız algoritma 0.92’ lük bir başarı oranı yakalamış ve bu algoritma için yine başaralı diyebiliriz fakat diğer algoritmalar arasında en az başarılı olan algoritma. Tabi yine seçilen parametrenin polinom olması hata oranı fazla çıkaran etken örneğin ‘rbf’ alındığında hata sayısının 2’ye düşmesi veya ‘linear’ alındığında hata sayısını 1’e düşmesi dolasıyla başarı oranın artması algoritmanın da daha iyi sonuç vermesi anlamına geliyor.**

***Naive Bayes:***

**Naive Bayes için sklear\_naive\_bayes altında GaussianNb import edildikten sonra GaussionNb nesnesi gnb oluşturuldu ve fit edildi. Burada fit ile eğitim sağlanmış oldu ve ardından predict ile tahmin değerleri, oluşturduğumuz y\_pred ’e aktarıldı. Ardından oluşturduğumuz tahmin değerlerinin verilerimizdeki gerçek değerlere karşılaştırmak için Confusion matrix kullandı. Confusion matrix için Sklearn\_metrics altında confusion\_matrix import edildi ve confusion matrix’ten cm adında bir nesne oluşturuldu. Bu nesnede oluşturduğumuz tahmin değerlerimizin tutulduğu y\_pred ve gerçek değerlerimizin tutulduğu y\_test arasındaki uyumluluk test edildi ve ne kadarı doğru tahmin ne kadarı yanlış tahmin olduğu bastırıldı.**

**Confusion matrix tablosu Naive Bayes için aşağıda gösterildiği gibidir. Bu tabloyu yorumlamak gerekirse;**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N=50** | **Tahmin değeri:**  **İris-setosa** | **Tahmin değeri:**  **İris-versicolor** | **Tahmin değeri:**  **İris-virgina** |  |
| **Gerçek değeri:**  **İris-setosa** | **16** | **0** | **0** | **16** |
| **Gerçek değeri:**  **İris-versicolor** | **0** | **19** | **2** | **20** |
| **Gerçek değeri:**  **İris-virgina** | **0** | **0** | **13** | **14** |
|  | **16** | **19** | **15** |  |

**Bu tabloda doğru tahmin yapılan değerler Gerçek pozitif ve Gerçek negatiflerin toplamı: 16 + 19+ 13 =48**

**Yanlış tahmin yapılan değerler Yanlış pozitifler ve Yanlış negatiflerin**

**toplamı : 0+0+2 =3**

**Accuracy rate(Doğruluk oranı)=48/50=0.96**

**Misclassfication rate(Yanlışlık oranı) =1-0.96=0.04**

**Burada kullandığımız algoritma 0.96’ lük bir başarı oranı yakalamış ve bu algoritma için gayet başaralı diyebiliriz. Başarıyı artıran veya azaltan herhangi bir parametre bulunmuyor. Naive bayes genellikle sarmal veri grupları için uygun bir algoritma bu proje içinde bu kadar yüksek performans vermesinin büyük ihtimale verilerimizin test için ayrılan bölümünün düşük olması olabilir. Ancak test için ayrılan değerlerin artırıldığında diğer algoritmalara nazaran en başarılı çalışan algoritma yine naive bayes algoritması oluyor. Buda tahmin ettiğim görüşün hatalı olduğu anlamına geliyor.**

***Decision Tree:***

**Decision Tree için sklear\_tree altında DecisionTreeClassifier import edildikten sonra DecisionTreeClassifier nesnesi Dtc oluşturuldu parametresi için splitter random seçildi ve fit edildi. Burada fit ile eğitim sağlanmış oldu ve ardından predict ile tahmin değerleri, oluşturduğumuz y\_pred ’e aktarıldı. Ardından oluşturduğumuz tahmin değerlerinin verilerimizdeki gerçek değerlere karşılaştırmak için Confusion matrix kullandı. Confusion matrix için Sklearn\_metrics altında confusion\_matrix import edildi ve confusion matrix’ten cm adında bir nesne oluşturuldu. Bu nesnede oluşturduğumuz tahmin değerlerimizin tutulduğu y\_pred ve gerçek değerlerimizin tutulduğu y\_test arasındaki uyumluluk test edildi ve ne kadarı doğru tahmin ne kadarı yanlış tahmin olduğu bastırıldı.**

**Confusion matrix tablosu Naive Bayes için aşağıda gösterildiği gibidir. Bu tabloyu yorumlamak gerekirse;**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N=50** | **Tahmin değeri:**  **İris-setosa** | **Tahmin değeri:**  **İris-versicolor** | **Tahmin değeri:**  **İris-virgina** |  |
| **Gerçek değeri:**  **İris-setosa** | **16** | **0** | **0** | **16** |
| **Gerçek değeri:**  **İris-versicolor** | **0** | **13** | **2** | **15** |
| **Gerçek değeri:**  **İris-virgina** | **0** | **6** | **13** | **19** |
|  | **16** | **19** | **15** |  |

**Bu tabloda doğru tahmin yapılan değerler Gerçek pozitif ve Gerçek negatiflerin toplamı: 16 + 13+ 13 =42**

**Yanlış tahmin yapılan değerler Yanlış pozitifler ve Yanlış negatiflerin**

**toplamı : 0+6+2 =8**

**Accuracy rate(Doğruluk oranı)=42/50=0.84**

**Misclassfication rate(Yanlışlık oranı) =1-0.84=0.16**

**Burada kullandığımız algoritma 0.84’ lük bir başarı oranı yakalamış ve bu algoritma için nispeten başaralı diyebiliriz. Bu algoritma da parametre olarak splitter=’random’ seçilmişti, bu parametre bilerek seçildi çünkü hata oranını yüksek olduğu parametreyi de göstermek istedim. Daha başarılı olan örneğin criteion{‘gini’,’entropi’} , splitter{‘best’} veya max\_dept gibi parametrelerde mevcut, bu parametrelerde hata oranı çok daha düşük seviyede bu da algoritmamızın başarılı çalıştığını gösteriyor.**

***Algoritmaların Karşılaştırılması ve Genel Özet:***

**Algoritmaların karşılaştırılması açısından çok büyük farklar yok çünkü seçilen iris verisi, yapısı itibariyle birçok sınıflandırma algoritmalarına uygun bir veri grubu. Bu da birçok algoritmaların başarılı bir şekilde çalışmasını sağladı. Tabi bu algoritmaların yüzde yüz başarı sağlaması anlamına gelmiyor fakat parametrelerin doğru seçilmesi durumundaki birkaç denemeyle yüzde yüz başarı sağlayan algoritmalarda oldu. Bu algoritmalar arasında K-nn ve Naive Bayes en başarılı algoritmalar oldu. Bu algoritmaların başarısını etkileyen birçok unsur mevcut bunlar arasında seçilen verinin normalizasyon aşamasından sonra makineye öğretme aşamasında split validation yöntemi yerine cross validation yöntemi başarıyı artıran unsur olurken bunun yanı sıra, seçilen algoritmalar içindeki parametrelerin seçimi de başarıyı artıran unsurlar arasında yer alıyor. Daha kompleks veri yapıları için bu kullanılan yöntemlerin seçimi için önemli bazı soruların cevap bulması gerekiyor. Bunun yanı sıra seçilen yöntemlerin hangi parametrelerle daha başarılı olacağının seçimide önem arz ediyor. Bunun için GridSearchCv kullanılabiliriz fakat bu veri yapısında bu özelliğe gerek duymadık.**

***6.Kaynaklar***

* [***https://bilkav.com/egitimler***](https://bilkav.com/egitimler)
* [***https://www.udemy.com/course/makine-ogrenmesi/learn/lecture/10130618#questions***](https://www.udemy.com/course/makine-ogrenmesi/learn/lecture/10130618#questions)
* [***https://veribilimcisi.com/2017/07/18/karisiklik-matrisi-nedir/***](https://veribilimcisi.com/2017/07/18/karisiklik-matrisi-nedir/)
* [***https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html***](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html)
* [***https://archive.ics.uci.edu/ml/datasets/iris***](https://archive.ics.uci.edu/ml/datasets/iris)