Yapay Zeka Ödev2

Kürşat Kömürcü 18014038

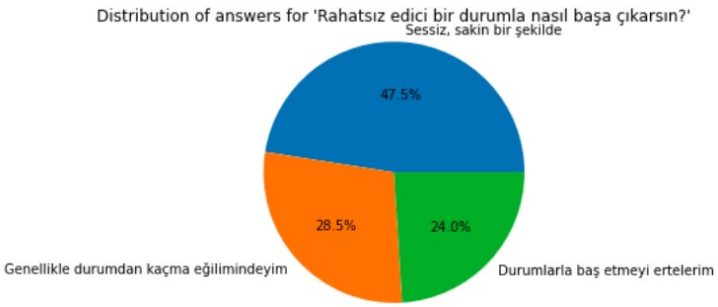
Ödev Konusu: Bir veri kümesi oluşturup makine öğrenmesi algoritmalarını çalıştırmak

Veri Kümesi:

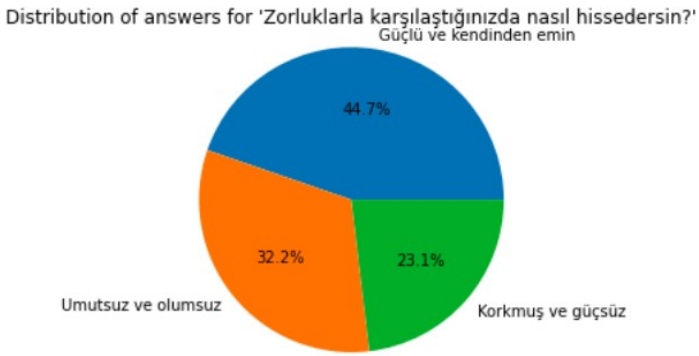
Veri kümesi 12 soruluk bir test hazırlanarak elde edilmiştir. Bu test özgüven seviyesi ile ilgilidir. Kişiler soruları cevapladıktan sonra, son soru olan “Son olarak 1-5 arasında öz güvenini derecelendirir misin?” sorusunu cevaplayarak veri etiketlemişlerdir. Ödevde ise diğer 11 soru kullanılarak öz güven sınıflandırılması yapılmaktadır.

Aşağıdaki grafiklerde her bir sorunun cevap oranı grafikleri görünmektedir.

Soru 1:

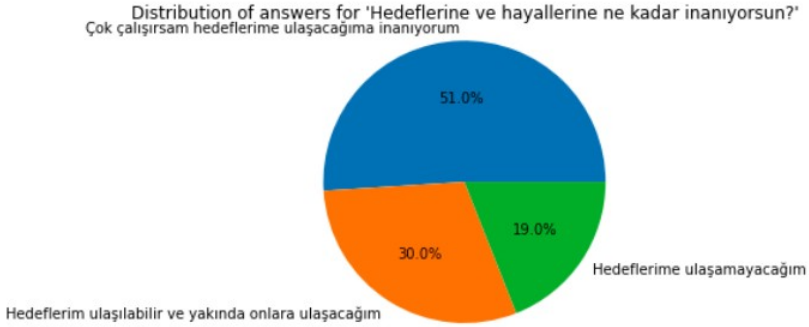


Soru 2:

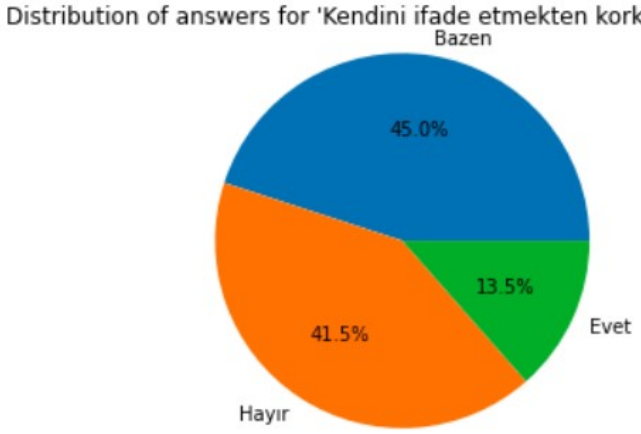


Soru 3:

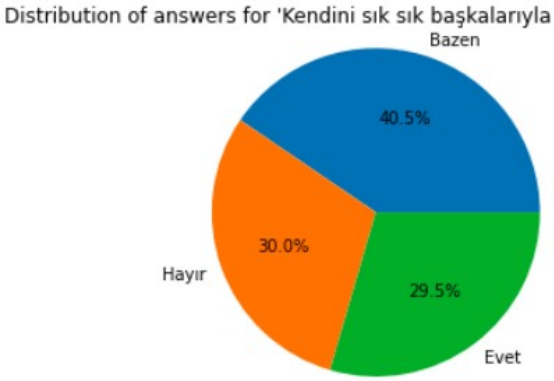
Soru 4:



Soru 5:



Soru 6:

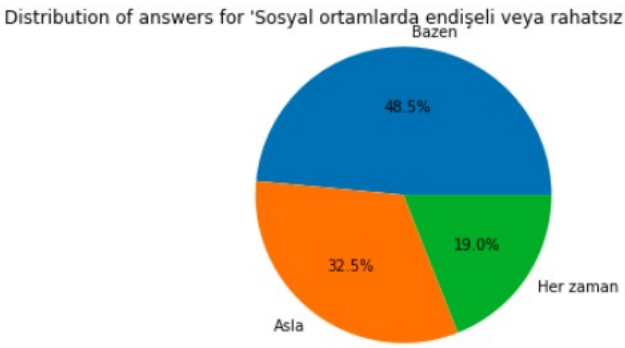


Soru 7:

Soru 8:

Soru 9:

Soru 10:



Soru 11:

Soru 12:

Veri Ön İşleme:
Testte son soru hariç her sorunun 3 farklı cevabı vardır. Bu kategorik verilerdir. Kategorik verileri sayısal verilere dönüştürmek için **One Hot Encoding** yöntemi kullanılmıştır.
One Hot Encoding, kategorik değişkenlerin her bir farklı değerini bir diziye dönüştürür. Bu dizi, değişkenin alabileceği tüm değerler için sıfırlardan oluşurken, sadece seçilen değeri temsil eden bir tane 1 içerir. Yani her bir kategori, ayrı bir sütun olarak temsil edilir.
Örnek olarak, bir "renk" değişkeni ele alalım ve bu değişkenin alabileceği değerler "kırmızı," "mavi" ve "yeşil" olsun. One Hot Encoding uygulandığında, "renk" değişkeni "kırmızı" için [1, 0, 0], "mavi" için [0, 1, 0] ve "yeşil" için [0, 0, 1] şeklinde temsil edilir.
One Hot Encoding'in amacı, kategorik verileri sayısal formata dönüştürerek makine öğrenimi algoritmalarının daha etkili bir şekilde çalışmasını sağlamaktır. Bu yöntem, kategori değerlerinin birbirleriyle doğrusal bir ilişkisi olmadığını vurgular ve bu nedenle algoritmaların veriyi daha doğru bir şekilde yorumlamasını sağlar.
One Hot Encoding, özellikle makine öğrenimi modellerinde kategorik değişkenlerin işlenmesi gerektiği durumlarda yaygın olarak kullanılan bir tekniktir.

Aşağıdaki verisetinin One Hot Encoding öncesi ve sonrası görülmektedir.

```
x = df.iloc[:, 0:12]
y = df.iloc[:, 12]
y = y.to_numpy()
x
```

0.0s Python

	Unnamed: 0	Rahatsız edici bir durumla nasıl başa çıkarsın?	Zorluklarla karşılaştığınızda nasıl hissedersin?	Bir engeli aştıktan sonra en çok neyi düşünürsün?	Hedeflerine ve hayallerine ne kadar inanıyorsun?	Kendini ifade etmekten korkuyor musun?	Kendini sık sık başkalarıyla karşılaştırır mısın?	"Başkalarının benim hakkımda düşündükleri ile ilgileniyorum." cümlesine ne kadar katılıyorsun?	İnsanlarla dolu bir odaya girdiğinizde nasıl hissediyorsun?	Karar vermekte veya onların arkasında durmakta zorlanıyor musun?	Sosyal ortamlarda endişeli veya rahatsız hissediyor musun?	İnsanların arkandan dedikodu yaptığını düşünüyor musun?
0	0	Sessiz, sakin bir şekilde	Umutsuz ve olumsuz	Çıkardığım dersleri	Çok çalışırsam hedeflerime ulaşacağıma inanıyorum	Hayır	Hayır	Katılmıyorum	Kötü hissediyorum	Bazen	Her zaman	Hayır
1	1	Sessiz, sakin bir şekilde	Umutsuz ve olumsuz	Yapmam gereken şeyler için pişman olduğumu	Çok çalışırsam hedeflerime ulaşacağıma inanıyorum	Bazen	Hayır	Ne katılıyorum ne katılmıyorum	Hiç bir şey hissetmiyorum	Bazen	Bazen	Hayır
2	2	Sessiz, sakin bir şekilde	Güçlü ve kendinden emin	Çıkardığım dersleri	Çok çalışırsam hedeflerime ulaşacağıma inanıyorum	Hayır	Bazen	Ne katılıyorum ne katılmıyorum	Güvende ve güçlü hissediyorum	Hayır	Asla	Evet ama umursamıyorum
3	3	Sessiz, sakin bir şekilde	Güçlü ve kendinden emin	Yapmam gereken şeyler için pişman olduğumu	Çok çalışırsam hedeflerime ulaşacağıma inanıyorum	Bazen	Bazen	Katılmıyorum	Hiç bir şey hissetmiyorum	Bazen	Bazen	Evet ama umursamıyorum
4	4	Sessiz, sakin bir şekilde	Güçlü ve kendinden emin	Çıkardığım dersleri	Hedeflerim ulaşılabilir ve yakında onlara ulaş...	Hayır	Bazen	Ne katılıyorum ne katılmıyorum	Hiç bir şey hissetmiyorum	Bazen	Asla	Evet ama umursamıyorum
...	...	...	...	...	...	...	...	...	...	...	...	...
195	195	Sessiz, sakin bir şekilde	Güçlü ve kendinden emin	Çıkardığım dersleri	Hedeflerime ulaşamayacağım	Evet	Evet	Ne katılıyorum ne katılmıyorum	Hiç bir şey hissetmiyorum	Evet, sürekli	Her zaman	Evet ve bu beni endişelendiriyor
196	196	Durumlarla baş etmeyi ertelerim	Güçlü ve kendinden emin	Çıkardığım dersleri	Çok çalışırsam hedeflerime ulaşacağıma inanıyorum	Bazen	Bazen	Ne katılıyorum ne katılmıyorum	Hiç bir şey hissetmiyorum	Bazen	Bazen	Evet ama umursamıyorum
197	197	Sessiz, sakin bir şekilde	Umutsuz ve olumsuz	Yaşadığım utançlar için endişeleniyorum	Hedeflerime ulaşamayacağım	Evet	Evet	Katılıyorum	Kötü hissediyorum	Evet, sürekli	Her zaman	Evet ve bu beni endişelendiriyor
198	198	Genellikle durumdan kaçma eğilimindeyim	Umutsuz ve olumsuz	Yaşadığım utançlar için endişeleniyorum	Hedeflerime ulaşamayacağım	Evet	Evet	Katılıyorum	Kötü hissediyorum	Evet, sürekli	Her zaman	Evet ve bu beni endişelendiriyor

```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder()
x = ohe.fit_transform(x).toarray()
x
```

0.0s

```
array([[1., 0., 0., ..., 0., 0., 1.],
       [0., 1., 0., ..., 0., 0., 1.],
       [0., 0., 1., ..., 1., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 1., 0.]])
```

Makine Öğrenmesi İle Sınıflandırma:
Sınıflandırma için 5 tane makine öğrenmesi algoritması seçilmiştir. Bu algoritmalar Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Tree (DT), Support Vector Classifier (SVC) ve Random Forest (RF) algoritmalarıdır.
Genellikle veri setleri %70 eğitim, %30 test veya %75 eğitim %25 test olarak bölünmesine karşın, bu çalışmada çok veri toplanamadığı için (200 satır) veri seti %90 eğitim, %10 test şeklinde bölünmüştür.

Not: Aşağıdaki algoritmaların ödevdeki veriye göre karar sınırları (decision boundry) veriye PCA uygulanmış halidir. PCA ödevin ilerleyen kısımlarında anlatılacaktır.

Logistic Regression (LR):
Logistic Regression, bir bağımlı değişkenin bağımsız değişkenlerle olan ilişkisini değerlendirmek ve tahmin etmek için kullanılan istatistiksel bir regresyon yöntemidir. İkili sınıflandırma problemlerinde sıklıkla kullanılan bir algoritmadır.

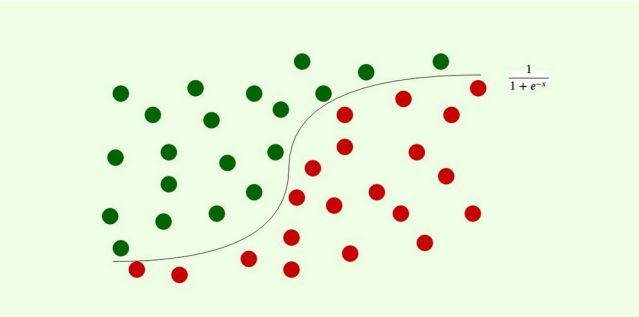
Logistic Regression, lineer regresyon yöntemine benzer bir şekilde çalışır, ancak farklı bir çıktıya sahiptir. Lineer regresyonda, bağımlı değişken sürekli bir değer alırken, Logistic Regressionda bağımlı değişkenin olasılık değerini tahmin eder. Bu olasılık değeri, 0 ile 1 arasında bir değerdir ve genellikle bir sınıfın (etiketin) ait olma olasılığını ifade eder.

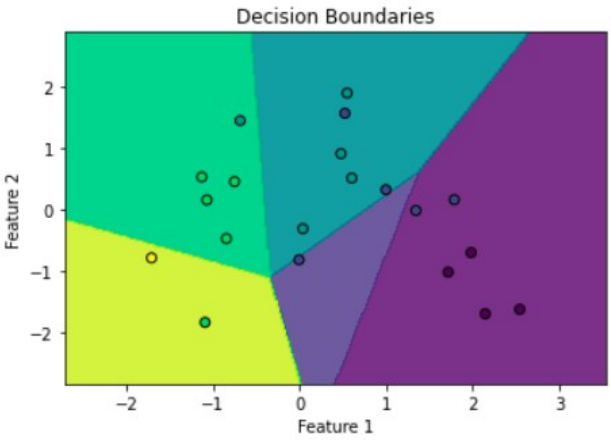
Logistic Regression, bağımlı değişkenin olasılık değerini hesaplamak için sigmoid (lojistik) fonksiyonunu kullanır. Bu fonksiyon, herhangi bir gerçek sayıyı 0 ile 1 arasında bir değere dönüştürür.

Logistic Regression modelinin eğitimi, en uygun ağırlık değerlerini bulmak için genellikle maksimum olabilirlik (maximum likelihood) veya gradyan inişi (gradient descent) gibi optimizasyon yöntemlerini kullanır. Eğitim tamamlandıktan sonra, model yeni veriler üzerinde tahminler yapabilir ve bir gözlemi belirli bir sınıfa atayabilir.

Logistic Regression modelinin eğitimi, en uygun ağırlık değerlerini bulmak için genellikle maksimum olabilirlik (maximum likelihood) veya gradyan inişi (gradient descent) gibi optimizasyon yöntemlerini kullanır. Eğitim tamamlandıktan sonra, model yeni veriler üzerinde tahminler yapabilir ve bir gözlemi belirli bir sınıfa atayabilir.

Logistic Regression, sınıflandırma problemlerinde sıklıkla kullanılır ve birçok uygulama alanı bulunur. Örneğin, pazarlama çalışmalarında müşteri churn (kayıp) tahmini, tıp alanında hastalık teşhisi, kredi riski değerlendirmesi gibi birçok alanda kullanılabilir.





K-Nearest Neighbors (KNN):
K-Nearest Neighbors (KNN), makine öğrenimi ve veri madenciliği alanlarında sınıflandırma ve regresyon problemlerini çözmek için kullanılan bir algoritmadır. KNN, temelde bir örneğin komşularına dayalı olarak sınıflandırma veya tahmin yapar.

KNN algoritması çalışırken, veri setindeki her bir örneğin komşularını belirlemek için kullanılan bir metrik kullanılır.

KNN algoritması, veri setindeki her bir örneği bu mesafeye göre diğer örneklerle karşılaştırır ve k-en yakın komşuları bulur. "k" parametresi, komşu sayısını belirtir. Bir tahmin yaparken, KNN çoğunluk sınıfı prensibiyle çalışır. Yani, k-en yakın komşuların sınıflarına bakar ve çoğunluğun sınıfını tahmin olarak verir. Örneğin, k=3 olarak belirlendiğinde, en yakın 3 komşunun sınıfları incelenir ve çoğunluğa göre bir sınıf tahmini yapılır.

KNN algoritması ayrıca regresyon problemlerinde de kullanılabilir. Regresyon için, k-en yakın komşuların değerleri alınır ve bunların ortalaması veya ağırlıklı ortalaması, tahmin edilen değeri oluşturur.

KNN algoritmasının bazı önemli özellikleri vardır:

1. KNN algoritması basit ve anlaşılır bir yöntemdir.

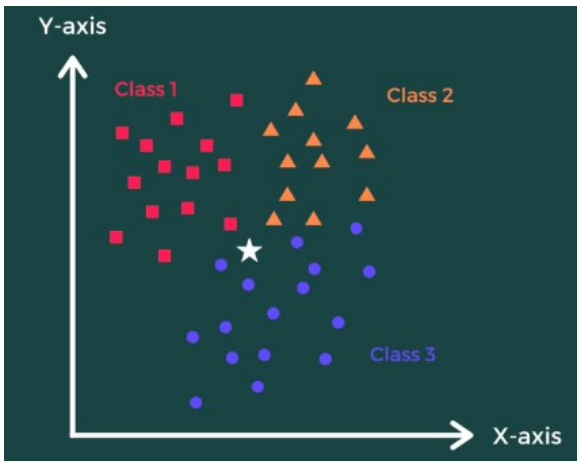
2. Eğitim süreci yoktur, veri seti tamamen hafızada tutulur.

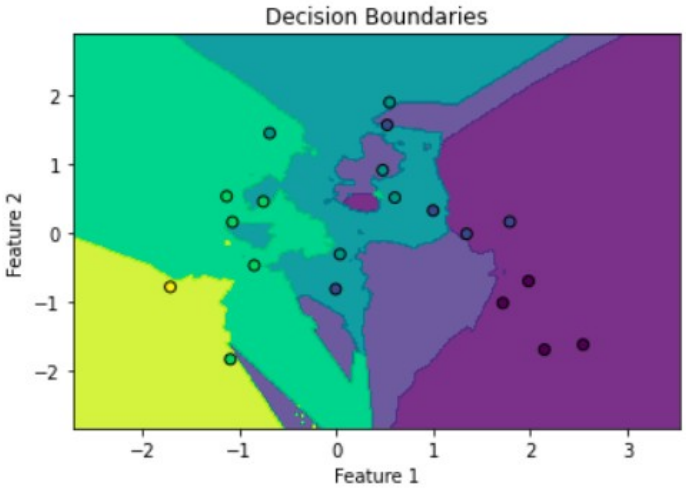
3. KNN'nin performansı, komşu sayısı ve mesafe ölçümü gibi parametrelere bağlıdır.

4. KNN algoritması, veri setindeki anormalliklere ve gürültülere karşı hassas olabilir.

5. Veri setinin boyutu büyüdükçe KNN algoritmasının hesaplama maliyeti artar.

KNN, sınıflandırma ve regresyon problemlerinde yaygın olarak kullanılan bir algoritmadır. Ancak büyük veri setleri ve yüksek boyutlu verilerde hesaplama maliyeti nedeniyle pratikte bazı kısıtlamalara sahip olabilir.





Decision Tree (DT):
Karar ağaçları, veri tabanındaki öznitelikleri kullanarak kararlar vermek için kullanılan ağaç yapısını temsil eder.

Karar ağacı algoritması, veri setindeki öznitelikleri kullanarak bir dizi karar kuralı oluşturur ve bu kurala göre veriyi sınıflandırır veya tahmin yapar. Karar ağaçları, veri setindeki öznitelikleri kullanarak sınıflandırma veya regresyon işlemi yaparken, her bir iç düğüm (node) bir öznitelik testiyle bölünür ve her bir dal (branch) bir öznitelik değerine karşılık gelir.

Karar ağaçlarının yapısı, veri setindeki özniteliklerin ve sınıfların bilgi kazancına (information gain) dayalı olarak bölünmesiyle oluşturulur. Bilgi kazancı, bir özniteliğin kullanılmasıyla elde edilen bilgi miktarını ölçer. Amaç, enformasyon kazancı en yüksek olan özniteliği seçerek her adımda veri setini en iyi şekilde bölmektir.

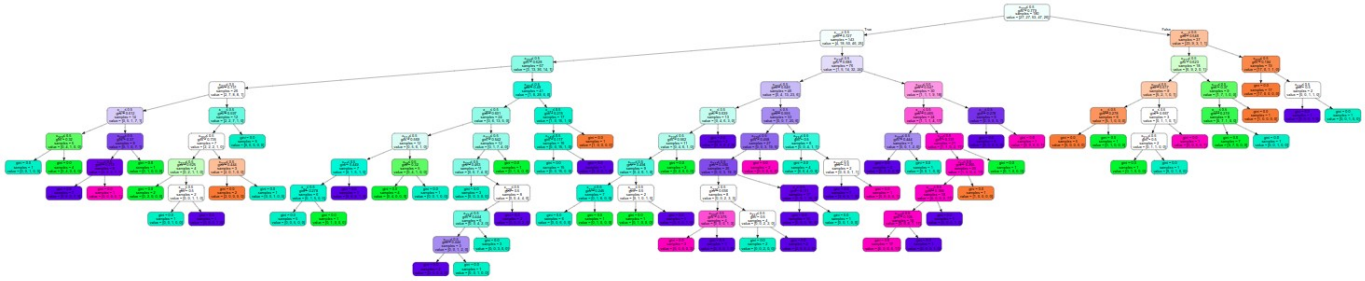
Karar ağacı algoritması, sınıflandırma problemleri için kullanıldığında, her bir yaprak düğümü (leaf node) bir sınıf etiketi temsil eder. Tahmin yapmak için, girdi veri seti ağaç yapısını takip ederek ilgili yaprak düğümüne ulaşır ve o düğümün sınıf etiketi tahmin olarak verilir.

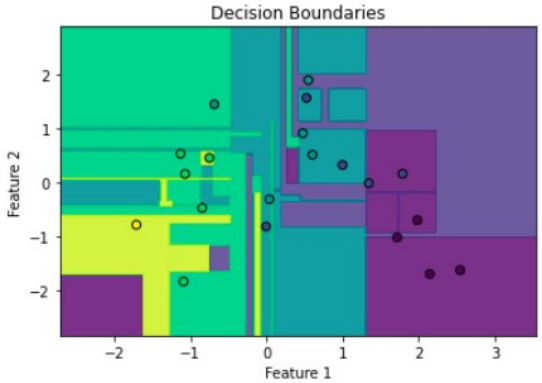
Regresyon problemleri için, yaprak düğümleri gerçek sayısal değerler içerir. Regresyon tahmini yapmak için, girdi veri seti ağaç yapısını takip ederek ilgili yaprak düğümüne ulaşır ve o düğümün içindeki değeri tahmin olarak verir.

Karar ağaçları, anlaşılması ve yorumlanması kolaydır. Ayrıca, kategorik ve sayısal öznitelikleri bir arada kullanabilme yeteneğine sahiptirler. Bunun yanı sıra, karar ağaçları doğal olarak çoklu sınıflarla başa çıkabilir ve eksik veri değerlerini yönetme yeteneği vardır.

Ancak, karar ağaçlarının aşırı uymaya (overfitting) eğilimli olabilmesi ve veri setindeki küçük değişikliklere duyarlı olması gibi bazı zayıflıkları vardır. Ayrıca, çok büyük ve karmaşık veri setlerinde performans sorunları ortaya çıkabilir.

Aşağıdaki şekilde ödevde kullanılan verinin karar ağacı görünmektedir.





Support Vector Classifier (SVC):
Destek vektör sınıflandırıcısı, veri noktalarını sınıflara ayırmak için destek vektörleri kullanır.

Destek vektör sınıflandırıcısı, iki sınıf arasında bir karar sınırı (decision boundary) oluşturmayı hedefler. Bu karar sınırı, en iyi şekilde sınıfları ayıran bir hiperdüzlem (hyperplane) olarak tanımlanır. Destek vektörleri, bu hiperdüzlem üzerindeki en yakın noktalardır ve karar sınırının belirlenmesinde önemli bir rol oynarlar.

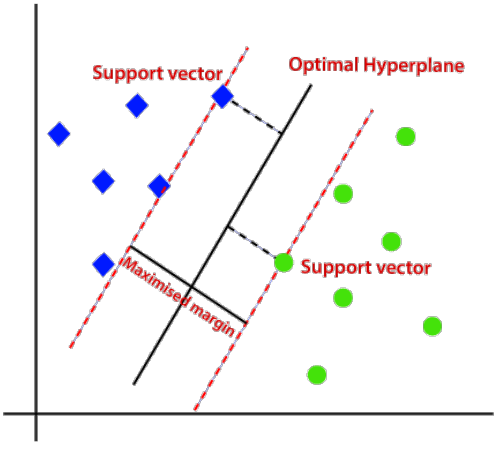
Destek vektör sınıflandırıcısı, lineer olarak ayrılabilir veri setlerinde kullanılabileceği gibi, doğrusal olarak ayrılamayan veri setleri için de çeşitli çekirdek fonksiyonları (kernel functions) kullanarak genişletilebilir. Çekirdek fonksiyonları, veri noktalarını yüksek boyutlu bir özellik uzayına dönüştürür ve böylece veri seti doğrusal olarak ayrılabilir hale gelir.

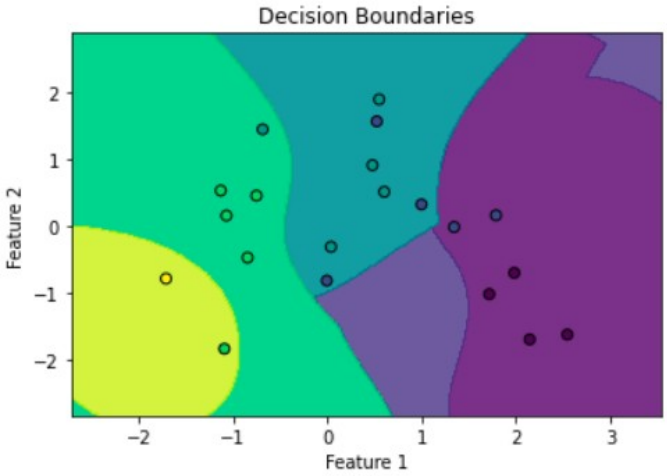
Destek vektör sınıflandırıcısı, veri noktalarını sınıflandırmak için destek vektörlerinin ve karar sınırının oluşturulmasında optimizasyon problemlerini çözer. Bu optimizasyon problemleri, maksimum marjin (maximum margin) ilkesine dayanır. Marjin, karar sınırı ile en yakın destek vektörleri arasındaki mesafeyi ifade eder ve maksimum marjin ilkesine göre, sınıfları en iyi şekilde ayıran karar sınırı, marjini maksimize eder.

Destek vektör sınıflandırıcısı aynı zamanda "kernel trick" adı verilen bir yöntemi kullanarak yüksek boyutlu özellik uzaylarında da etkili bir şekilde çalışabilir. Bu sayede doğrusal olarak ayrılamayan veri setlerini doğrusal bir hiperdüzlemle ayrılabilir hale getirir.

Destek vektör sınıflandırıcısı, aşırı uymaya (overfitting) eğilimli olmadığı için genellikle iyi bir genelleme performansına sahiptir. Ayrıca, daha küçük boyutlu veri setlerinde etkili çalışır ve aykırı değerlere (outliers) karşı dirençlidir.

Destek vektör sınıflandırıcısı, sınıflandırma problemlerinde yaygın olarak kullanılan bir algoritmadır ve özellikle ikili sınıflandırma problemlerinde tercih edilir.





Random Forest (RF):
Random Forest (Rastgele Orman), makine öğrenimi alanında sınıflandırma ve regresyon problemlerini çözmek için kullanılan bir ensamble öğrenme yöntemidir. Random Forest, birden çok karar ağacını bir araya getirerek daha güçlü ve kararlı bir model oluşturur.

Random Forest algoritması, temel olarak birçok karar ağacının (decision tree) oluşturulması ve bu ağaçların sonuçlarının bir araya getirilmesiyle çalışır. Her bir karar ağacı, veri setinin rastgele örneklemlerini (bootstrap samples) kullanarak eğitilir. Bootstrap örnekleme, veri setinden rastgele örneklemler çekerek yeni alt veri setleri oluşturma işlemidir. Bu sayede, her bir ağaç farklı örneklemlerle eğitilir ve çeşitlilik sağlanır.

Karar ağaçları oluşturulurken, her bir düğümde bir öznitelik alt kümesi rastgele seçilir ve bu öznitelikler arasından en iyi bölme kriterine göre ağaç büyütülür. Bu özellik seçim işlemi, ağaçlar arasında çeşitlilik sağlar ve genel modelin daha iyi bir genelleme yapmasını sağlar.

Sınıflandırma problemlerinde, Random Forest modeli, karar ağaçlarının sınıf oylamalarını birleştirerek çoğunluk sınıfını tahmin eder. Regresyon problemlerinde ise, ağaçların çıktıları birleştirilerek ortalama veya medyan gibi bir tahmin değeri elde edilir.

Random Forest algoritması birçok avantaja sahiptir:

1. Random Forest, yüksek boyutlu veri setlerinde ve çoklu özniteliklerle çalışmada etkilidir.

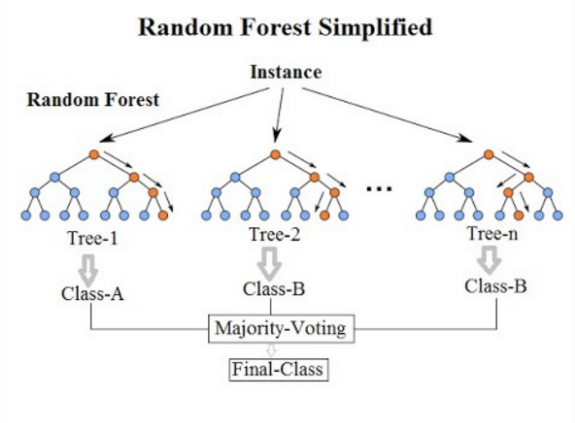
2. Aşırı uymaya (overfitting) karşı dirençlidir ve genelleme yeteneği yüksektir.

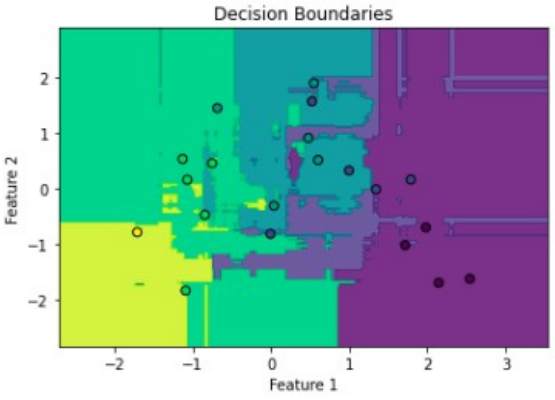
3. Hem sınıflandırma hem de regresyon problemlerinde başarılı sonuçlar verir.

4. Veri setindeki eksik değerler ve aykırı değerlerle başa çıkma yeteneği vardır.

5. Random Forest, özniteliklerin önem sıralamasını sağlayarak veri setindeki önemli öznitelikleri belirleme imkanı sunar.

Random Forest algoritması, sınıflandırma ve regresyon problemlerinde yaygın olarak kullanılan ve başarılı sonuçlar veren bir yöntemdir. Ayrıca, modelin yüksek doğruluk sağlaması, genel kullanımını ve popülerliğini artırmaktadır.





Yukarıda bahsedilen 5 Makine Öğrenmesi Algoritması ile sınıflandırma yapıldıktan sonra Principal Component Analysis (PCA) kullanılıp aynı algoritmalar tekrar edilmiştir.

Principal Component Analysis (PCA):
PCA (Principal Component Analysis), veri analizi ve boyut indirgeme için kullanılan istatistiksel bir tekniktir. PCA, bir veri kümesindeki değişkenlik yapılarını anlamak ve veri setini daha az sayıda değişkenle temsil etmek için kullanılır.

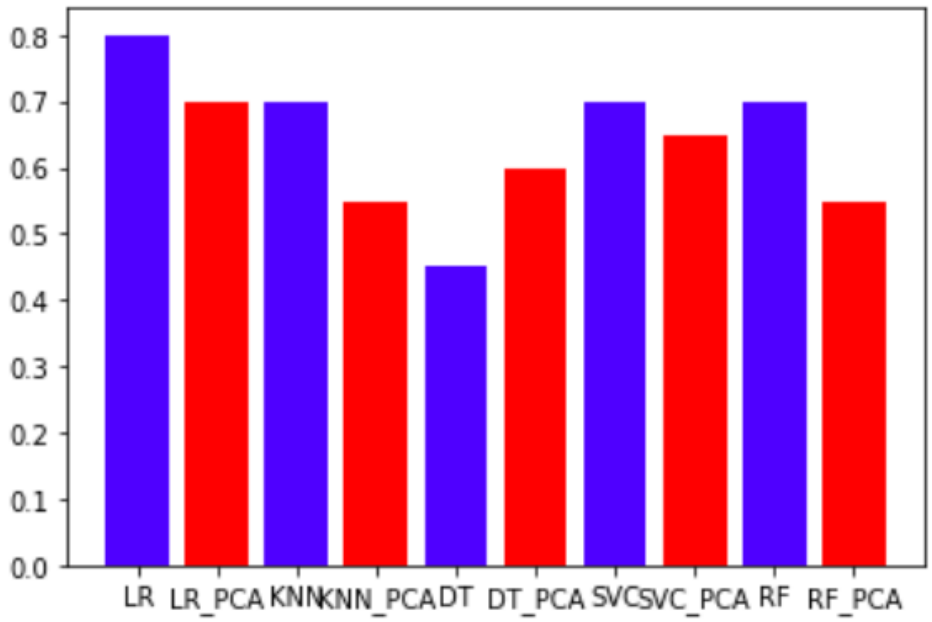
PCA'nın temel amacı, çok boyutlu bir veri setindeki değişkenler arasındaki ilişkileri özetlemektir. Bu ilişkiler, orijinal veri setindeki değişkenler arasındaki kovaryans matrisine dayalı olarak belirlenir. PCA, bu kovaryans matrisini analiz ederek, veri setindeki değişkenlik yapılarını tanımlayan yeni bir değişken seti olan "ana bileşenler" elde eder.

PCA, veri setini daha az sayıda boyuta indirgemek için kullanılır. Bu, veri setinin boyutunu azaltırken mümkün olduğunca fazla değişkenlik bilgisini koruma amacını taşır. Bu şekilde, veri setindeki gürültüyü azaltır, gereksiz bilgiyi ortadan kaldırır ve analiz için daha kolay anlaşılabilir bir yapı sağlar.

PCA'nın sonucunda elde edilen ana bileşenler, orijinal veri setindeki değişkenlik yapısını yansıtır. İlk ana bileşen, veri setindeki en fazla değişkenliği açıklar ve sıralı olarak diğer ana bileşenler, kalan değişkenlik miktarını açıklar. Bu sayede, istenen boyuta indirgenmiş bir veri seti elde edilir.

PCA, veri analizi, desen tanıma, görüntü işleme, biyoinformatik ve diğer birçok alanda yaygın olarak kullanılan bir yöntemdir. Boyut indirgeme, veri görselleştirme ve gürültü azaltma gibi birçok uygulaması bulunmaktadır.

Aşağıdaki Şekilde ödevde kullanılan Makine Öğrenmesi Algoritmalarının ve veriye PCA uygulandıktan sonraki hallerinin doğruluk oranları görünmektedir.



T-Test:
T-test, iki örneklem grubunun ortalamaları arasında istatistiksel olarak anlamlı bir fark olup olmadığını değerlendirmek için kullanılan bir hipotez testidir. İki grup arasındaki farkın tesadüfi olup olmadığını belirlemek için kullanılan bir istatistiksel testtir.

T-testi sonucunda elde edilen istatistiksel bir değer olan t değeri, grupların ortalamaları arasındaki farkın büyüklüğünü ve örneklem büyüklüğünü dikkate alır. Bu değer, kritik t değerleri ile karşılaştırılarak istatistiksel anlamlılık elde edilip edilmediği belirlenir.

T-testi, istatistiksel analizde sıklıkla kullanılan bir yöntemdir ve hipotez testleri, güven aralıkları ve p-değerleri gibi istatistiksel sonuçların elde edilmesinde önemli bir rol oynar.

Makine öğrenmesi modelleriyle yapılan t-test sonuçlarının pozitif veya negatif çıkması, farklı modellerin performansları arasında anlamlı bir farkın olduğunu göstermektedir. Ancak, bu sonuçların anlamını daha iyi anlayabilmek için bazı faktörleri dikkate almak önemlidir.

1. T-testin uygulandığı veri: Farklı modellerin performansını karşılaştırmak için kullanılan veri seti önemlidir. Veri seti, özelliklerin sayısı, veri dağılımı, hedef değişkenin niteliği gibi faktörlerle birlikte farklılık gösterebilir. Bu nedenle, farklı veri setleri üzerinde yapılan test sonuçları da farklılık gösterebilir.

2. T-testin türü: Hangi tür t-testini uyguladığınız da sonuçları etkileyebilir. Bağımsız iki örneklem t-testi mi yoksa başka bir tür t-test mi kullandığınız önemlidir. Modeller arasında anlamlı bir farkın olup olmadığını belirlemek için uygun t-test türünün seçilmesi gerekmektedir.

3. Yönlendirme: T-test sonuçları pozitif veya negatif olabilir, ancak bu durum modellerin hangi yönde daha iyi olduğunu göstermez. Pozitif t-test sonucu, bir modelin diğerinden daha iyi performans gösterdiğini ifade edebilirken, negatif t-test sonucu ise diğer modelin daha iyi performans gösterdiğini gösterebilir. Sonuçların yönü, uyguladığınız test ve karşılaştırdığınız modellerin bağlamına bağlıdır.

4. İstatistiksel anlamlılık: T-test sonucunun pozitif veya negatif olması tek başına anlam ifade etmez. İstatistiksel anlamlılığı belirlemek için p-değerini değerlendirmek önemlidir. p-değeri, elde edilen farkın tesadüfi olup olmadığını belirler. Düşük p-değerleri, farklı modeller arasında anlamlı bir fark olduğunu gösterirken, yüksek p-değerleri farkın tesadüfi olduğunu gösterir.

Sonuç olarak, t-test sonuçlarının pozitif veya negatif olması, farklı modellerin performansları arasında anlamlı bir fark olduğunu gösterir. Ancak, bu sonuçların anlamını tam olarak değerlendirmek için diğer faktörleri dikkate almak ve istatistiksel anlamlılığı değerlendirmek önemlidir.

Aşağıda ödevde kullanılan algoritmaların t-testleri görünmektedir.

```python
from mlxtend.evaluate import paired_ttest_5x2cv

t, p = paired_ttest_5x2cv(estimator1=lr, estimator2=knn, X=x, y=y)
print('t statistic between lr and knn: %.3f' % t)

t, p = paired_ttest_5x2cv(estimator1=lr, estimator2=dt, X=x, y=y)
print('t statistic between lr and dt: %.3f' % t)

t, p = paired_ttest_5x2cv(estimator1=lr, estimator2=svc, X=x, y=y)
print('t statistic between lr and svc: %.3f' % t)

t, p = paired_ttest_5x2cv(estimator1=lr, estimator2=rf, X=x, y=y)
print('t statistic between lr and rf: %.3f' % t)

t, p = paired_ttest_5x2cv(estimator1=knn, estimator2=dt, X=x, y=y)
print('t statistic between knn and dt: %.3f' % t)

t, p = paired_ttest_5x2cv(estimator1=knn, estimator2=svc, X=x, y=y)
print('t statistic between knn and svc: %.3f' % t)

t, p = paired_ttest_5x2cv(estimator1=knn, estimator2=rf, X=x, y=y)
print('t statistic between knn and rf: %.3f' % t)

t, p = paired_ttest_5x2cv(estimator1=dt, estimator2=svc, X=x, y=y)
print('t statistic between dt and svc: %.3f' % t)

t, p = paired_ttest_5x2cv(estimator1=dt, estimator2=rf, X=x, y=y)
print('t statistic between dt and rf: %.3f' % t)

t, p = paired_ttest_5x2cv(estimator1=svc, estimator2=rf, X=x, y=y)
print('t statistic between svc and rf: %.3f' % t)
```

✓ 6.0s

```
t statistic between lr and knn: 4.140
t statistic between lr and dt: 1.592
t statistic between lr and svc: 1.186
t statistic between lr and rf: -2.203
t statistic between knn and dt: 1.085
t statistic between knn and svc: 0.908
t statistic between knn and rf: -0.603
t statistic between dt and svc: -1.818
t statistic between dt and rf: -0.691
t statistic between svc and rf: 1.683
```

Yorum:

Oluşturulan veri setinde denenen algoritmalar arasında en yüksek doğruluk oranı olan (%80) Logistic Regression, en düşük doğruluk oranı olan ise (%45) Decision Tree olmuştur. PCA uygulandıktan sonra dört algoritmanın doğruluk oranları biraz düşerken Decision Tree algoritmasının doğruluk oranı yükselmiştir.

Daha fazla veri toplanarak, teste daha fazla soru ekleyerek ya da bu beş algoritmadan farklı algoritmalar tercih edilerek doğruluk oranı daha da arttırılabilir.

Yararlanılan Kaynaklar:

1. https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/

2. https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

3. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

4. https://www.geeksforgeeks.org/decision-tree/

5. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

6. https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

7. https://builtin.com/data-science/step-step-explanation-principal-component-analysis

8. https://www.analyticsvidhya.com/blog/2019/05/statistics-t-test-introduction-r-implementation/