HİPERSPEKTRAL GÖRÜNTÜLERİN ANALİZİ

Hazırlayan : **Tutku Gündüz**
Öğrenci No : **090150332**

Teslim Tarihi : **20.06.2021**

Ders : **MAT 492**
Danışman : **PROF. DR. ATABEY KAYGUN**

İçindekiler Tablosu

ÖNSÖZ

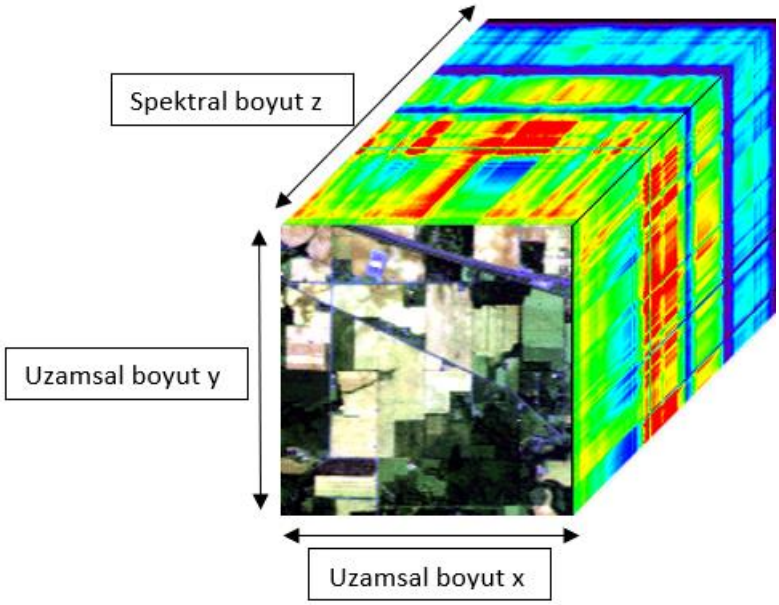
Lisans eğitimimin son projesi olan tez çalışmamda öncelikle konumun belirlenmesinde ve daha sonra hayata geçirilmesi sürecinde, bilgi ve tecrübelerinden faydalandığım, araştırmamın her bir aşamasında görüşleriyle beni destekleyen, samimiyetini her zaman hissettiren ve bana yol gösteren saygıdeğer danışman hocam Prof. Dr. Atabey KAYGUN'a, en içten şükranlarımı sunarım. Ayrıca üniversite hayatım boyunca desteklerini hiçbir zaman eksik etmeyen arkadaşlarıma ve bana olan güvenlerini hiç kaybetmeden her zaman yanımda olan aileme teşekkür ederim.

Tutku Gündüz
Haziran, 2021

Şekil Tablosu

1 Giriş

Gelişen ve gelişmeye devam etmekte olan teknoloji ile bilgisayar sistemleri aracılığıyla görüntü işleme uygulamalarının kullanım alanları gün geçtikçe artmaktadır. Bu uygulamalar yaygın olarak güvenlik, sağlık, askeriye, robot ve araç endüstirisi, insansız hava araçları gibi alanlarda kullanılmakla beraber günlük hayatta karşımıza çıkan birçok problemi çözmek veya işlemleri kolaylaştırmak amacıyla da kullanılmaktadır. Görüntü işlemenin önemli ve hızla gelişen bir alanı olan uzaktan algılama ile elde edilen hiperspektral görüntülere (HSG) ait veri setleri, yüksek boyutlu spektral ve uzamsal bilgiye sahip olduğundan standart görüntü işleme yöntemlerinden farklı işlemler içermektedir. Uzaktan algılama ile toplanan verinin elektromanyetik spektrumu uzamsal ve spektral bilgilerden oluştuğun için veri spektrumundaki bilgilerin işlenmesi ile istenen materyaller, konum ve fiziksel bilgilerden yola çıkılarak tespit edilebilmektedir [1]. Uydu görüntülerinin işlenmesi ile elde edilen bu veriler, insan gözünün görebildiği dalga boylarının haricinde kızıl ötesi bölgelere ait dar ve bitişik çok fazla sayıda dalga boyuna ve yüksek sayıda piksele sahip olmakla beraber her bir piksel yüzlerce spektral bant içermektedir [2]. İlk iki boyutu uzamsal ve üçüncü boyutu spektral bilgi içeren hiperspektral görüntüler, Şekil 1.1'de Indian Pines veri seti üzerinden gösterildiği gibi veri küpü olarak elde edilmektedir. Spektral bantlar, görüntü verisine ait bilgileri içerdiğinden veri üstündeki işlemlerde oldukça önemli olması ile beraber bant sayısı arttıkça içerdiği bilgiler de artmaktadır. Uzaktan algılama teknolojisi ile elde edilen görüntünün bulunduğu arazi bilgisi üzerinden sınıf tahmini yaparken spektral bilgiden yararlanılarak bulunan spektral işaret kullanılır. Elde edilen verilerin ayrıntılı ve fazla bilgi içermesi, birbirine yakın nesneleri ayırmada ve tespit etmede, madencilik, tarım arazileri kullanımı, astronomi [3], askeriye [4], ve tıp [5] gibi birçok alanda gelişime katkı sağlamaktadır.



Şekil 1.1: Indian Pines veri seti küpü

Hiperspektral görüntüler üzerinde uygulanan algoritmalarda karşılaşılan en önemli problemlerden biri veri setinin büyüklüğünden kaynaklanmaktadır. Spektral bantların yüksek sayıda bulunması ve dar konumlanması, bilgi fazlalığına neden olarak algoritmanın verimliliğine faydası olmaksızın aynı veri işlemlerinin tekrarına neden olmaktadır [6]. Bu durum da zaman ve maliyet kaybına yol açmaktadır. Bu problemin giderilmesi için veri setleri üzerinde boyut indirge uygulanmaktadır. Bu uygulama sonucunda gereksiz ve fazla bilgiden kurtulan veri, temel özellikler ile bilgi kaybı olmaksızın işlenebilmektedir. HSG veri setleri üzerinde işlem kolaylığı sağlayan boyut indirgeme kadar önemli bir başka uygulama ise görüntünün konumlandığı bölgeye ait sınıfların tespitidir. Sınıflandırma, görüntü ve arazi bilgilerini içeren verilerin makine öğrenmesi algoritmaları tarafından kullanılması ile uygulanarak verimli bir şekilde gerçekleştirilebilmektedir.

Bu çalışmada Indian Pines, Salinas ve Botswana olmak üzere 3 farklı veri setleri üzerinde boyut indirgeme algoritması olarak temel bileşenler analizi (TBA), sınıflandırma algoritmaları olarak ise destek vektör makineleri (DVM) ile yapay sinir ağları (YSA) kullanılarak bu algoritmaların analizlerinin yapılması amaçlanmıştır. Aynı algoritmada kullanılan farklı veri setleri sonuçları birbiri ile kıyaslanırken sınıflandırma algoritmalarının ise aynı veri seti üzerinden birbiri ile kıyaslaması yapılmıştır. Kıyaslama başarı oranları ve sınıf tahmin sayıları üzerinden yapılırken deneyler için 3 farklı hiperspektral veri seti kullanılmıştır.

İlk bölümde hiperspektral görüntüler ve işlenmesi hakkında genel bilgi verilerek çalışmanın amacına ve kullanılan algoritmalara değinilmiştir.

Bu çalışamada kullanılan veri setlerinin özellikleri açıklanmıştır.

İkinci bölümde, çalışmada kullanılan temel bileşenler analizi, destek vektör makineleri ve yapay sinir ağları yöntemlerinin anlatıldığı metodoloji bölümü yer almaktadır.

Üçüncü bölümde veri setleri üzerinde Python yazılım dili ile uygulanan algoritmalara ait deneyler adım adım açıklanmıştır.

Dördüncü bölümde, hiperspektral veri setlerine uygulanan deneylerden elde edilen sonuçlar analiz edilmiştir.

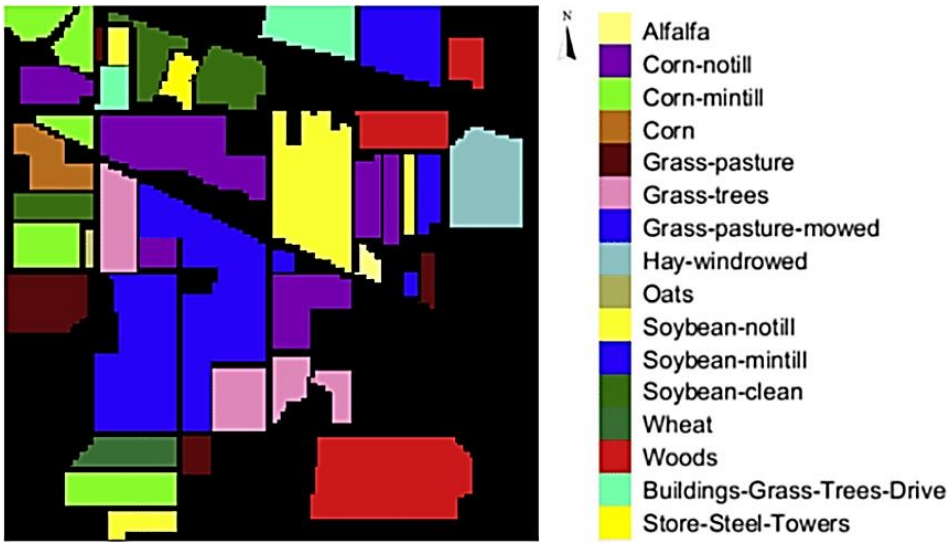
Beşinci bölümde çalışmada referans gösterilen kaynaklar verilmiştir.

Son olarak altıncı bölümde, çalışmada kullanılan deneylerin Python yazılım dilinde yazılmış olan kodları verilmiştir.

1.1 Indian Pines Veri Seti

Kuzeybatı Indiana bölgesindeki test sahası üzerinden NASA'nın AVIRIS sensörü tarafından alınan Şekil 1.2'de arazi görüntüsü yer alan hiperspektral görüntünün veri seti olan Indian Pines, 400 ile 2500 nm aralığında dalga boyuna sahiptir. Şekil 1.3'de görüleceği 16 bitki sınıfı içeren arazi örtüsü ve 20m uzamsal çözünürlüğe sahip 145*145 piksel barındıran Indian Pines veri seti, 4'ü bozuk ve yüksek gürültülü bantlar olmak üzere toplam 224 spektral banttan oluşmaktadır. Su emilimi bölgesini kaplayan spektral bantlar ile atmosfer etkileri ve sensör yapısından kaynaklı meydana gelebilen gürültülü bantlar çıkarılarak 200 spektral bant üstünden hesaplamalar yapılabilmektedir ancak bu çalışmada deneyler 220 spektral bant üzerinden yapılmıştır.

Şekil 1.2: Indian Pines veri setine ait gerçek renk haritası

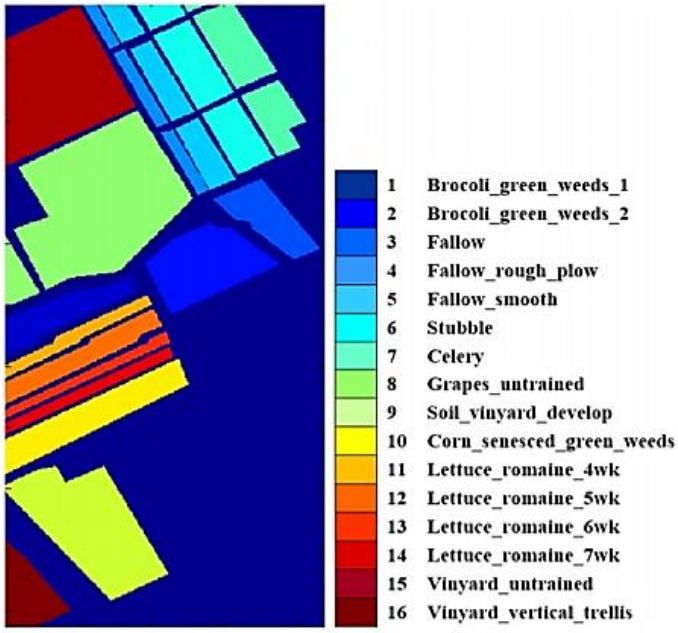


Şekil 1.3: Indian Pines veri setine ait arazi bilgisi

1.2 Salinas Veri Seti

Salinas, Amerika'nın batısında yer alan Kaliforniya eyaletindeki Şekil 1.4'te gösterilen büyük ve verimli tarım arazisi olan Salinas Vadisi üzerinde, NASA'ya ait AVIRIS sensörü tarafından alınan hiperspektral görüntüye ait veri setidir. Arazi örtüsü Şekil 1.5'te gösterildiği üzere 16 sınıfa ayrılan Salinas, her biri 3.7m uzamsal çözünürlüğe sahip 512*217 pikselden ve 224 spektral banttan oluşmaktadır. 20 su emilimi bölgesine ait spektral bant çıkarılarak 204 bant üzerinden de işlem yapılabilmektedir. Bu çalışmada 224 spektral üzerinden deneyler yapılmıştır.

Şekil 1.4: Salinas veri setine ait gerçek renk haritası

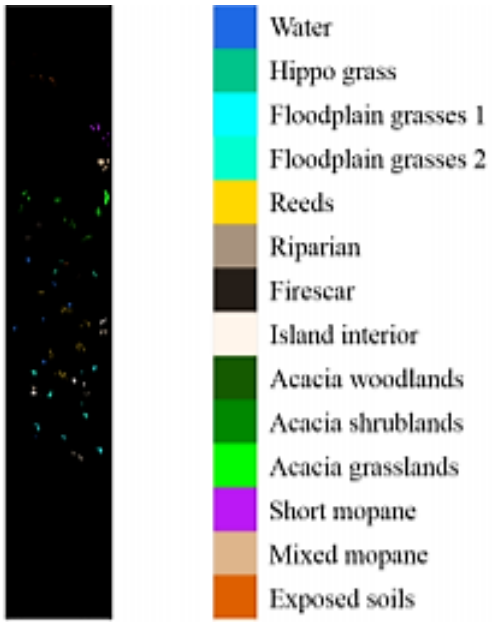


Şekil 1.5: Salinas veri setine ait arazi bilgisi

1.3 Botswana Veri Seti

Afrika kıtasının güneyinde yer alan Botsvana ülkesinde bulunan Şekil 1.6'da görüntüsü yer alan Okavango Deltası üzerinde NASA EO-1 uydusundaki Hyperion sensörü ile alınan hiperspektral görüntü verisi Botswana, 242 spektral banttan ve 30m uzamsal çözünürlüğe sahip 1476*256 pikselden oluşmaktadır. Şekil 1.7'de gösterildiği üzere 14 sınıfa ayrılan arazi bilgisine sahip Botswana verisi 400-2500 nm aralığında dalga boyuna sahiptir. Su emilimi bölgelerini kapsayan ve gürültü içeren bantların kaldırılmasıyla 145 spektral bant elde edilir. Bu çalışmada yapılan deneylerde Botswana veri setinin spektral bantları 145 olarak belirlenmiştir.

Şekil 1.6: Botswana veri setine ait gerçek renk haritası



Şekil 1.7: Botswana veri setine ait arazi bilgisi

2 METODOLOJİ

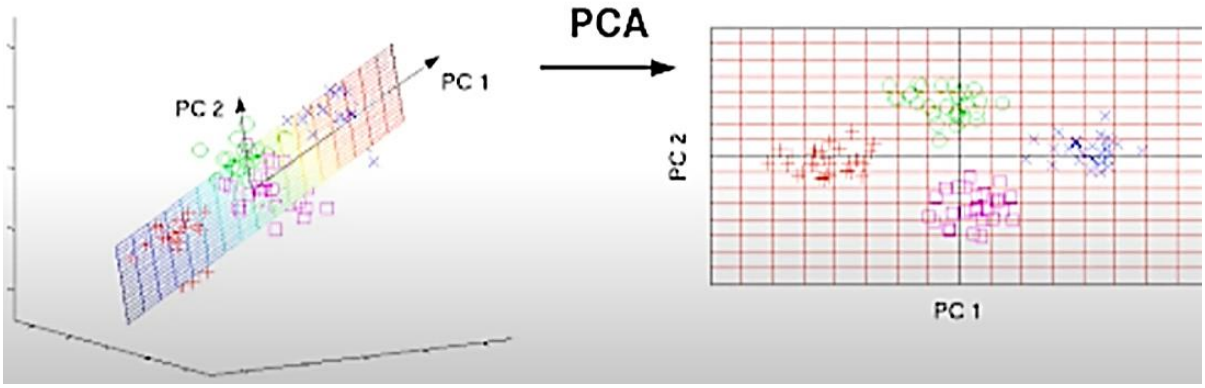
Temel bileşenler analizi, destek vektör makineleri ve yapay sinir ağları metotları, Python'da bulunan çeşitli kütüphanelerin kullanılmasıyla Indian Pines, Salinas ve Botswana veri setleri üzerinde deneyleri yapılmıştır. Bu bölümde veri setleri üzerinde deneyleri yapılan bu sınıflandırma ve boyut indirgeme algoritmaları ayrıntılı olarak incelenmektedir.

2.1 Görüntü Analizi

Indian Pines veri kümesi yüksek boyutlu bir küme olduğundan işlem yapabilmek oldukça zordur. Bu sebeple hiperspektral görüntüyü barındıran veri setlerinde daha verimli uygulamalar yapabilmek adına öncelikli hedef boyut indirgeme yaparak temel verileri daha düşük boyutlu bir uzaya konumlandırabilmektir.

2.1.1 Temel Bileşenler Analizi (TBA)

Hiperspektral görüntüler birbirine komşu olan çok sayıda sıkışık banttan oluşmaktadır ve bu bantlar görüntüye ait bilgileri içermektedir. Birbirine çok yakın bulunan bantlar genellikle görüntü hakkında aynı veya gereksiz bilgileri içerebilmektedir. Bu durum bantlar arasında bir korelasyon bulunduğunu göstermektedir. 1901 yılında Karl Pearson tarafından geliştirilen TBA, istatiksel yöntemlerle yüksek boyutlu verilerin sahip olduğu bilgileri özetlemeyi amaçlamaktadır [7]. Karhunen Loeve dönüşümü olarak da bilinen TBA uygularken amaç bantların istatiksel bilgilerini kullanarak aralarındaki korelasyonu ortadan kaldırıp gerekli bilgileri kaybetmeden birbiri ile aynı olan veya gürültü barındıran bantlardan kurtularak boyutu indirgemek ve böylelikle veriyi olabilecek en verimli şekilde doğrusal olarak dönüştürmektir [8]. Bu dönüşüm yapılırken esas veri yeni boyutlu koordinat sistemine, piksel değerlerinin varyansının en büyük bulunduğu yönde alınan ve verinin kovaryans matrisine özdeğer-özvektör dönüşümü yapılmasıyla elde edilen izdüşüm vektörlerine göre taşınmaktadır. En büyük özdeğerden başlanarak sıralanır ve karşılık gelen özvektörlerden ise sütunlar yazılır. Böylelikle izdüşüm matrisi elde edilir [9]. TBA işlemleri ile beraber yeni boyuttaki değişkenler doğrusal olmakla beraber birbirinden bağımsız ve dikey hale getirilmiş olduğundan bilgi fazlalığından kurtulma ile beraber boyut indirgenmiş olur. Şekil 2.1, temel bileşen analizi ile alt uzay boyutuna indirgenmiş olan veri seti örneğinden algoritmanın işleyişi anlaşılabilmektedir. Boyut indirgenmesi esnasında tekrar eden bilgilerden kurtulmanın yanı sıra görüntüyü bozan ve işlemleri yavaşlatan gürültü de azalmış olacaktır. Böylelikle temel olan gerekli bilgiler üzerinden algoritmalar uygulanması ile işlem ve zaman tasarrufu yapılarak deneylerin doğruluk performansı da arttırılmış olur.



Şekil 2.1: 3 boyutlu uzayda bulunan veri setinin temel bileşenler analizi ile 2 boyutlu uzaya konumlanması

Özdeğer ve özvektörler bulunabilmesi ve temel bileşenler elde edilmesi için yapılan matematiksel işlemler aşağıda anlatılmaktadır [10, 11].

$X_n = [X_{n1}, X_{n2} \dots X_{nF}]^T$ hiperspektral görüntü küpüne ait bir spektral vektör olmaz üzere n değeri 1 ile S aralığında bulunmaktadır. Buradaki S değeri görüntü küpünün uzamsal boyutunu, F değeri ise spektral bant sayısını temsil etmektedir. Temel bileşenler analizi ile görüntü küpü bir veri matrisine dönüştürülür. Ardından düzeltilmiş ortalama spektral vektör, denklem 1.2 ile elde edilir ve bu işlemlerde kullanılan ortalama spektral vektör ise denklem 1.1 ile ifade etmektedir.

$$I' = \frac{I}{S} \sum_{n=1}^{S} X_n \tag{1.1}$$

$$I_n = X_n - I' \tag{1.2}$$

Düzeltilmiş ortalama vektörlerden oluşan matris $I_0 = [I_1, I_2, \dots I_F]$ olmak üzere denklem1.3 ile kovaryans matris elde edilmektedir.

$$C = E\{(X_n - E\{X_n\})\,(X_n - E\{X_n\})^T\} = E\{I_n I_n^T\} \tag{1.3}$$

Denklem 1.3'te yer alan E{}, matematiksel beklenti formülüdür. I_0 matrisi yüksek boyutlu olduğundan kovaryans matrisinin hesaplanması zorludur. Bu sebeple kovaryans matrisi ayrıştırılarak özdeğer ve özvektörler elde edilir. Kovaryans matrisinin özdeğer ve özvektörler ile ifade edilişi denklem 1.4 ile gösterilmiştir. D = diag ($\lambda_1, \lambda_2, \dots \lambda_F$), kovaryans matrisi C'nin özdeğerlerinden oluşan diyagonal matrisini ve 'V' ise özvektörlerden oluşan matrisi ifade etmektedir.

$$C = VDV^T \tag{1.4}$$

Denklem 1.5'te I_n vektörünün korelasyonsuz vektöre dönüşümü verilmiştir.

$$y_n = V^T I_n \tag{1.5}$$

Özdeğerler büyükten küçüğe sıralanarak en küçük değerler çıkartılır. Böylelikle 'q' değeri temel bileşenlerin ayıklanmış hali olmak üzere $z_n = [z_{n1}, z_{n2} \ldots\ldots z_{nq}]^T$ olarak elde edilmiştir.

2.2 Sınıflandırma

Sınıflandırma işlemi, programa girilen veri kümelerini kendi içerisinde etiketlendirme yöntemidir. Farklı etiketlere eşleştirme yani sınıflara ayırma işlemi makine öğrenmesinde, programa tahmin edebilme yeteneği kazandırılarak yapılmaktadır. Kullanılan algoritmalarda sınıflandırma işlemi yapılırken veri kümelerinden ayrılan test ve öğretme alt kümeleri beraberinde, model oluşturma ve sınıflandırma fonksiyonları ile program tahmin becerileri geliştirilebilmektedir. Bu bölümde, sınıflandırma işlemleri için destek vektör makineleri ile yapay sinir ağları anlatılmaktadır.

2.2.1 Destek Vektör Makineleri (DVM)

Denetimli sınıflandırma yöntemlerinden biri olan SVM, uzaktan algılanan hiperspektral görüntülerin arazi bilgilerine sahip veri setindeki etiketlerin görüntüdeki bölgelerle eşleştirilmesiyle yapılan ve yaygın olarak kullanılan bir yöntemdir. Veri kümesindeki sınıflar arasında dikey mesafeyi maksimize (marj maksimizasyonu) ederek bir hiper düzlem oluşturup birbirinden en doğru şekilde ayırmayı amaçlamaktadır. Bu yöntem de DVM'nin sınıflandırma yapabilmek adına oluşturduğu hiper düzlemi istatiksel tabanlı yaklaşımlardan ve dağılımlardan değil geometrik bir kriter olarak marj maksimizasyonu kullanarak elde ettiği görülmektedir [12]. Genellikle, doğrusal olarak ayrılabilen ikili sınıflandırma işlemlerinde kullanılan DVM algoritmasında birbirine en yakın değişkenler seçilmesiyle sınır çizgilerinden oluşturulan doğrular destek vektörleri, bu sınır çizgileri arasında çizilen doğru ise karar doğrusu adını almaktadır. Bu ikili işlem DVM algoritmasının basit bir uygulamasıdır ve yapılan işlemler ile en yüksek marj bulunarak destek vektörleri belirlenip oluşturulan hiper düzlem ile iki sınıf arasında yapılan ayrıştırma işlemi Şekil 2.2'de gösterilmektedir. Yüksek boyutlu verilerde de kullanılan DVM, birden fazla sınıfın sahip olduğu örneklerin fazlalığı nedeniyle optimizasyonun yükü önemli ölçüde artar ve bu durum sınıf parametrelerinin tahminini daha zor hale getirerek basit ikili sınıflandırmada olduğu kadar başarılı bir sonuç vermesini engeller [13]. Verideki bilgiler eğitim için ve sonucu test etmek için kullanılır. Bir kısmı ile eğitim kümesi oluşturularak programın daha sonraki örnekleri doğru sınıflandırabilmesi amaçlanmaktadır. Uydudan alınan hiperspektral görüntülerde sınıflandırma doğrusal olarak ayırılamamaktadır. Bu durumda ise çekirdek fonksiyonları yardımı ile verinin doğrusal olarak ayrılması sağlanabilir. Temelde ikili

sınıflandırma yöntemi olan DVM, hiperspektral görüntülemede ikiden fazla sınıfı farklı yöntemler ile iki sınıf gibi ele alarak standart yöntemi uygulayabilmektedir. Daha yüksek boyutlu sınıf ayrımları için hiper düzlem hesaplanırken matematiksel varsayımlar ile işlem yapılmaktadır. 2 boyutlu koordinat sisteminde hesaplanan lineer doğru denkleminden yola çıkarak iki sınıfı birbirinden ayıran hiper düzlem;

$$W.X + b = 0 \tag{2.1}$$

denklemi ile her boyut için hesaplanabilmektedir. Burada w vektörü çizgi normali, x eğitim verisi ve b değeri ise sapmadır.

$$W = (w_0, w_1) \tag{2.2}$$

$$X = (x, y) \tag{2.3}$$

Değerleri kullanılarak;

$$w_0 x + w_1 y + b = 0 \tag{2.4}$$

$$w_1.y = -b - w_0.x \tag{2.5}$$

$$y = -\left(\frac{w_0 x}{w_1}\right) - \left(\frac{b}{w_1}\right) \tag{2.6}$$

elde edilir. Ardından

$$a = -\frac{w_0}{w_1} \tag{2.7}$$

$$c = -\frac{b}{w_1} \tag{2.8}$$

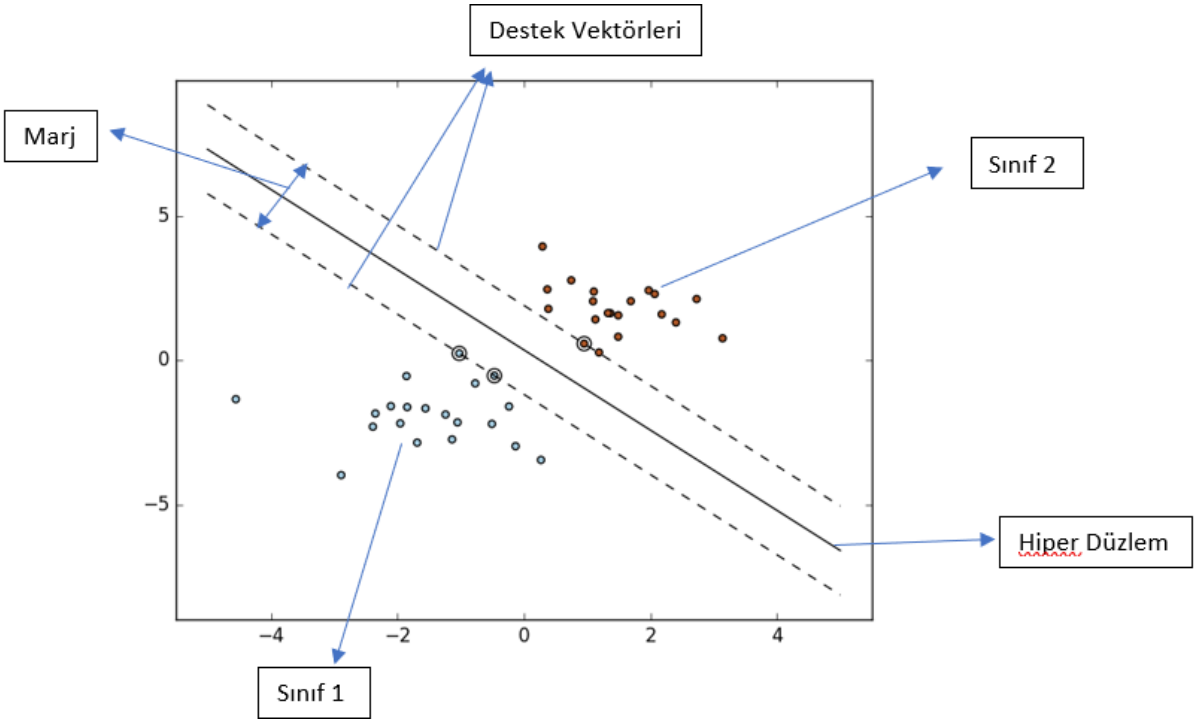
değerleri ile

$$y = ax + c \tag{2.9}$$

olarak tanımlanabilmektedir. Bu durumda n-boyutlu bir hiper düzlem için gerekli matematiksel manipülasyonlar yapıldığında hiper düzleme en yakın verinin uzaklığı aşadaki gibi hesaplanabilmektedir.

$$W' = arg_W max \frac{1}{\|W\|_2} \left[min_n y_n \left[W^T \emptyset(x_n) + b\right]\right] \tag{2.10}$$

DVM algoritmasının matematiksel ifadesi kısaca bu denklemler ile gösterilebilmektedir [14].

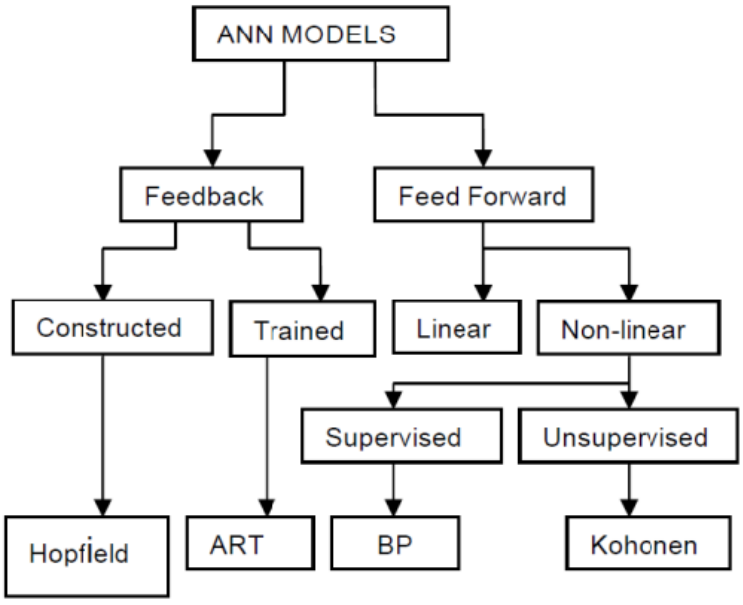


Şekil 2.2: Destek vektör makinasının ikili sınıflandırması

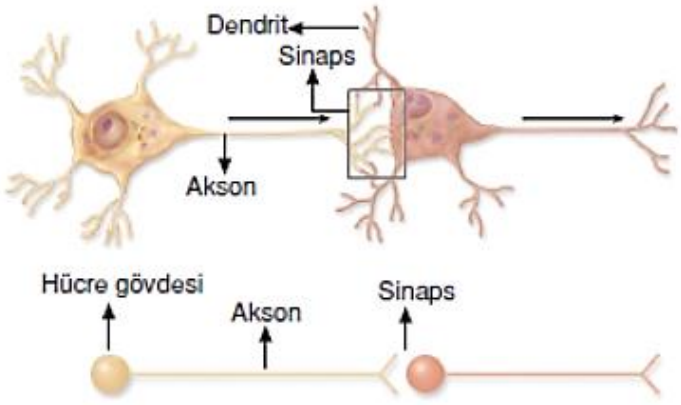
2.2.2 Yapay Sinir Ağları (YSA)

Makine öğrenmesi alt kümesi olan derin öğrenmenin algoritmalarından olan yapay sinir ağları görüntü işlemenin birçok alanında kullanılmakla beraber özellikle uzaktan algılama ile elde edilen uydu görüntülerindeki arazi bilgilerinin sınıflandırılmasında verimli bir şekilde uygulanmaktadır. Yapay sinir ağları bilgisayar ortamında matematiksel olarak insan öğrenmesini kullanmak amacıyla geliştirilmiştir. Temelinde insan beyninin işleyişinden esinlenilen yapay sinir ağları girilen verilerin alt kümelere ayrılmasıyla elde edilen eğitim verilerinin, oluşturulan sinir ağları modeli ile yapılandırılmasıyla doğru tahminli çıktılar elde edilmesi amaçlanmaktadır. Böylelikle insan düşünce sisteminde olduğu gibi öğrenilen bilgilerin hafızada tutulması ile daha sonra gelen bilgiler işlenirken deneyimlerden yararlanılarak tahminlerde bulunulur. Deneyim çokluğu aynı insan öğrenmesinden de örnek gösterilebileceği gibi beraberinde tahmin doğruluğunu da arttırmaktadır. Bilgisayar ortamına taşınan bu yapının benzerliğinin ve işleyişinin daha iyi anlaşılabilmesi adına yapısı Şekil 2.4'te gösterilmiş olan nöronun biyolojik yapının da anlaşılması önemlidir. İnsan beyninde, gövdeyi saran ve bilgiyi alan dentritlerden, bilgiyi hücre gövdesinden diğer uca taşıyan aksondan ve akson ile dentrit arasındaki iletişimi sağlayan sinaptik boşluktan oluşan tipik sinir hücreleri birbiriyle sinapslar oluşturarak mesaj iletimi sağlarken oluşan mesaj sinyalinin yüksekliği nöronlar arası kurulan bağlantı gücüyle doğru orantılıdır. [15]. YSA işlemleri gerçekleştirilirken, nöronların vücutta gerekli işlevleri iletmesini sağlayan sinapsların görevini bilgisayar ortamında matematiksel olarak modellene sinir ağlarında bulunan ağırlıklar temsil ederken, bu ağırlıkların

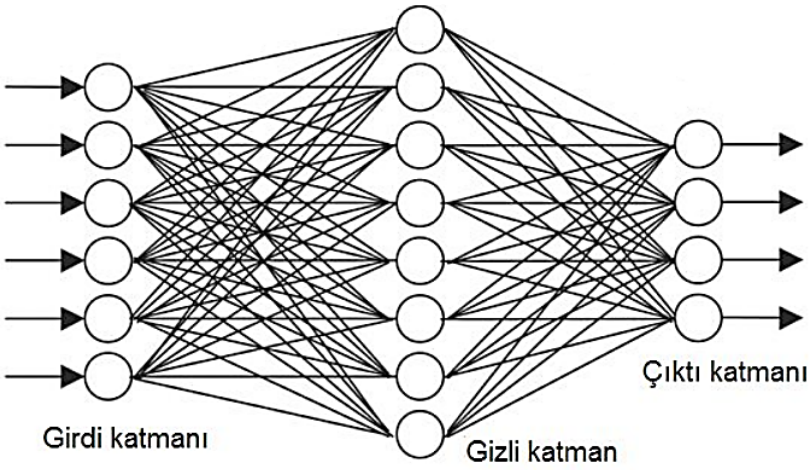
girdi verileri ile işleme girmesinin ardından aktivasyon fonksiyonu uygulanmasıyla çıktı verisi elde edilmektedir. [16]. Genellikle 3 katmandan oluşan ve katman yapıları ile çokluğuna göre çeşitlendirilerek farklı isimlere ayrılan sinir ağları modeli Şekil 2.5'te de gösterildiği üzere giriş katmanı, gizli katman ve çıkış katmanı olarak ayrılmaktadır. Bu katman sayıları kullanılan veriye ve istenilen sonuçtan beklenen doğruluk oranına göre değiştirilerek farklı sinir ağı modelleri oluşturulabilmektedir. Ancak fazla katman ve nöron her zaman daha iyi tahmin anlamına gelmemekle beraber zaman ve işlem karmaşıklığına neden olabilmektedir. Bu nedenle programda deneyler yapılırken çeşitli model yapıları denenerek zaman ve sonuç ilişkisine göre en verimli yapı seçilmeye çalışılır. Temel yapay sinir ağları modelleri Şekil 2.3 'te gözüktüğü üzere çeşitli alt başlıklara ayrılmıştır. Veri setinin özelliklerine ve başarım oranlarının farklılıklarına göre karar verilerek hangi modelin uygulanacağına karar verilir. Bu tablodan farklı olarak da çok sayıda model geliştirilmiştir [17]. Eğitim verileri tahmin edici modelde uygulanırken test verileri başarı oranını hesaplayan işlemlerde kullanılmaktadır. Elde edilen sonuç tahmin edicinin gerçek değere yakınlığını gösterir. Sinir ağları modeli oluşturmada en önemli faktörlerden biri de katmanlardaki aktivasyon fonksiyonunun belirlenmesidir. Bu çalışmada aktivasyon fonksiyonlarından 'relu' ile 'softmax' kullanılmıştır.



Şekil 2.2: Yapay sinir ağlarının sınıflandırması



Şekil 2.3: İnsan sinir hücresi (nöron) yapısı



Şekil 2.4: Yapay sinir ağları modeli

3 DENEYLER

Bu bölümde önceki bölümlerde anlatılan metotlar Indian Pines, Salinas ve Botswana veri seti üzerinde uygulanmıştır. Deneyler genelde başlangıç olarak Indian Pines veri seti üzerinden anlatılmıştır ancak her ne kadar üç veri seti de farklı bölgelere ait olsa da, hiperspektral görüntülerden oluşan benzer yapılarda verilere sahip olduklarından deneyler birbiriyle aynı işlemleri içermektedir. Bu sebeple Indian Pines veri setine ait deneyler, aynı kütüphaneler, fonksiyonlar ve algoritmalar kullanılarak diğer veri setlerine de uygulanmaktadır. Ek-A.1, Ek-A.2, Ek-A.3, Ek-B.1, Ek-B.2, Ek-B.3 ve Ek-C.1 bölümünde üç veri setine de uygulanan TBA, DVM ve YSA deneylerine ait kodlar verilmiştir.

3.1 Temel Bileşenler Analizi (TBA)

Indian Pines veri üzerinde python programında bulunan fonksiyonlar ile PCA uygulanarak boyut indirgenmesi amaçlanmıştır. Öncelikle bu deneyde kullanılacak olan fonksiyonların bulunduğu kütüphaneler ve modüller tanımlanmalıdır. Aksi takdirde fonksiyonlar işlem esnasında içeri aktarılamadığından tanımlanamaz ve deney çalışamayarak hata verir. PCA uygulamasında önemli olan matplotlib, pandas, numpy, sklearn.decomposition, scikit-learn, ve scipy.io adlı modül ve kütüphanelerin tanımlanmasıdır. İlerleyen adımlarda uygulanacak işlemler için gerekli fonksiyonların kullanılabilecek duruma getirilmesinin ardından bilgisayar ortamında matlab dosyası olarak bulunan Indian Pines veri seti loadmat fonksiyonu ile programa indirilir. Böylelikle Indian Pines veri seti üzerinde işlemler yapılabilir. Ancak bu veri seti yalnızca bir banttan oluşmaktadır bu sebeple işlemlerden doğru sonucu alabilmek için yeniden boyutlandırılmalıdır. Daha önceki bölümlerde de bahsedildiği üzere Indian Pines veri seti 145x145 piksel ve 220 banttan oluşmaktadır. Böylelikle (145*145, 220) olarak boyut düzenlemesi yapılarak devam edilmelidir. Daha sonraki adımda ise PCA fonksiyonu çağırılarak 220 banttan oluşan Indian Pines veri setini 3 banta indirgemek için fonksiyonda bulunan n temel bileşen 3'e eşitlenir. Bu modelin tamamlanması için 3 temel bileşen olarak tanımlanan fonksiyon sonraki adımda fit ve fit_transform işlemleri veri setin üzerinde uygulanır. Yeni modelin boyutuna bakıldığında artık 220 değil 3 banttan oluştuğu görülebilmektedir. Son olarak boyutu indirgenen veri setinden görüntü yazdırmak için yeniden boyutlandırma yapılarak tek bir bantta görüntü elde edilebilir. Bu boyutlandırma işlemi (145, 145, 3) şeklinde yapılır ve matplotlib kütüphanesinden plt.imshow fonksiyonu ile alınabilir.

3.2 Destek Vektör Makineleri (DVM)

SVM sınıflandırma yöntemi, Indian Pines veri seti üzerinde python programı ile uygulanmıştır. Bu uygulamada amaç sınıflandırma yöntemi ile Indian Pines veri setinde bulunan 16 sınıfın birbirinden en iyi şekilde ayrılabilmesini sağlamaktır. PCA uygulamasında da olduğu gibi öncelikle gerekli modül ve kütüphaneler tanımlanmalıdır. Bunlar matplotlib, pandas, numpy, scipy.io, seaborn ve sklearn kütüphaneleri ve modülleridir. SVM uygulamasında PCA uygulamasından farklı olarak sadece hiperspektral görüntüye ait veri seti değil aynı zamanda bölgenin arazi bilgisine sahip veri seti de indirilmelidir. Böylelikle görüntüdeki bölgelerin etiketleri eşleştirilerek sınıflandırma yapılabilir. Her iki veri seti de tüm pikselleri kapsayacak şekilde diğer algoritmalarda da uygulandığı gibi yeniden boyutlandırılır ve arazi bölgesine ait isim etiketlerinin bulunduğu bir dizi tanımlanır. Verilerin yeni boyutunun nasıl olduğu görülmek istenirse shape ile programda basılabilir. Böylelikle verilerde işlem uygulanabilmesi için eşit sayıda örnek içeren boyuta getirildiği de görülebilmektedir. Bunun gibi temel adımlar tamamlandıktan sonra Support Vector Machine algoritmasının uygulanabilmesi için gerekli fonksiyonların uygulandığı adımlara geçilir. Makine öğrenmesinde genellikle algoritmalar, özellikler benzer bir ölçekte ve normal dağılıma yakın olduğunda iyi ve hızlı bir performans gösterir. Aksi halde veri setlerinde farklı ölçeklerde bulunan özellikler model işlevlerine eşit katkıda bulunamaz ve sapma yaratabilir. Bunun sonucunda ise tahmincinin doğru öğrenmesine engel olabileceğinden, öncelikle hiperspektral görüntünün bulunduğu Indian Pines veri setinin standardizasyonu yapılmalıdır. Standardizasyon metodu verideki özelliklerin ortalama değerini '0' ve standart sapmasını ise '1' alarak dağılımı normale yakınlaştırmaktadır. Bu metodun uygulanabilmesi için bir nesne tanımlanarak, bu nesneye StandardScaler() fonksiyonu tanımlanır ve standardizasyon işlevini uygulayacak olan fonksiyon kütüphaneden çekilmiş olur. Ardından standartlaştırma fonksiyonuna tanımlanan nesne, bir sonraki adımda fit_transform() işlevi ile hiperspektral görüntünün bulunduğu veri setine uygulanarak model verilere uydurulur ve modele göre veri özellikleri dönüştürülür. Model uydurma ve standartlaştırma işlevleri tamamlandıktan sonra veri setleri train_test_split fonksiyonu kullanılarak test ve öğrenme verileri olmak üzere rastgele iki alt kümeye ayrılırlar. Test için ne kadar veri ayrılacağı kullanılan fonksiyon içerisinde test_size parametresi ile belirlenirken random_state parametresi ise bu ayrılmaya karar verecek olan rastgele sayı üreticisidir. Sabit bir sayı seçilmesi her seferinde aynı rastgele sayı dizisini oluşturmayı sağlamakla beraber sonucun doğruluğunu arttırmaktadır. Support Vector Machine uygulamasına hazırlık adımları tamamlanmasıyla beraber sınıflandırma işlemi ile bir model oluşturulur. Sınıflandırma fonksiyonu tanımlandıktan sonra parametre değerleri girilir ve eğitim verileri üzerinde 'fit()' işlemi uygulanır. Ardından tahmin için bir nesne tanımlanır. Eğitilmiş modelde etiket tahminini sağlayacak olan bu işlem test edilecek olan veriyi kullanarak predict() fonksiyonu ile sınıflandırma işlemi üzerinde uygulanır. Elde edilen bu tahmin modeli ile arazi verisinden ayrılan test verisi confusion_matrix() işlemi ile beraber yeni bir nesneye tanımlanır. Bu işlem karmaşıklık matrisi adı verilen yapıyı oluşturarak test verisi ile bulunan tahmin değerlerinin karşılaştırılmasıyla oluşturulan modelin performansını göstermeyi amaçlamaktadır. Bunun

sonucunda oluşturulan modelin verilerini grafik üzerinde yazdırarak elde edilen sınıflandırmada bulunan tahmin değerler görülerek başarı oranı gözlenebilir.

3.3 Yapay Sinir Ağları (YSA)

Yapay sinir ağları, oluşturulan sinir ağları modelinin eğitilmesi ile parametreleri tahmin etmeye yarayan bir makine öğrenimi yöntemidir. Her deneyde olduğu gibi öncelikle işlemlerde uygulanması gereken fonksiyonları kullanılabilmesi adına gerekli modül ve kütüphaneler tanımlanmalıdır. Bunlar ise diğer deneylerden farklı olarak başta tensorflow olmak üzere matplotlib, pandas, numpy, scipy.io, seaborn ile sklearn modül ve kütüphaneleridir. Kütüphaneler programa tanımlandıktan sonra SVM deneyinde de olduğu gibi iki ayrı veri seti de indirilmeli ve doğru bir şekilde tekrardan boyutlandırılmalıdır. Hiperspektral görüntünün yer aldığı veri seti üzerinde yine SVM deneyinde olduğu gibi standardizasyon yapılır ve her iki veri seti de test ve eğitim verileri olmak üzere iki alt kümeye ayrılır. Bununla birlikte görüntünün bulunduğu veriden ayrılarak oluşturulan test ve eğitim verileri üzerinde normalleştirme işlemi uygulanarak değerler 0 ile 1 aralığında olmak üzere yeniden ölçeklendirilir ve böylelikle performans arttırılması sağlanır. Standart işlemlerin ardından yapay sinir ağları uygulamasının temeli olan ağ modeli oluşturmaya geçilir. Ağ modeli oluşturmak sinir ağları uygulamasının performans hızını ve doğruluğunu önemli ölçüde etkileyen kilit bir işlem olmakla beraber isteğe göre çeşitlendirilebilmektedir. Bu deneyde kullanılan model için öncelikle kullanılacak modül ve fonksiyonları barındıran Sequential sınıfı keras işlemi ile model adı verilen nesneye atanır. Böylelikle sonraki adımlarda add() fonksiyonu model üstünden uygulanarak katmanlar eklenir ve yapay sinir ağları geliştirilir. Öncelikle tf.keras.layers.Flatten() eklenir ve böylelikle girdi verisi tek boyutlu bir diziye dönüştürülür. Bu düzleştirme işleminin yapılma sebebi sinir ağlarında piksellerin nöron işlevi görmesidir. İlk katman veride yer alan toplam piksel kadar olmalıdır. Sonraki adımda tf.keras.layers.Dense() ile ara katmanlar eklenir. Dense, oluşturulan gizli katmandaki her nöronu önceki katmandaki nöronlara bağlayan bir sinir ağıdır. Bu modele 'Dense' katmanları eklenirken aktivasyon fonksiyonu olarak 'Relu' kullanılmıştır. 'Rectified linear unit' olarak isimlendirilen aktivasyon fonksiyonun kısaltması olan Relu, doğrultulmuş lineer birim olarak çevrilebilmektedir. Relu, doğrusal olmayan bir aktivasyon fonksiyonudur ve girdi verilerini pozitif ise değiştirmeden, negatif ise 0'a dönüştürerek hesaplar. Modeli oluştururken son olarak çıktı katmanı girilir. Bu katmandaki nöron sayısı sınıflandırma işlemi uygulanan hiperspektral verinin sahip olduğu arazi sınıfı sayısına göre belirlenmektedir. Aktivasyon fonksiyonu olarak son katmanda 'Softmax' kullanılmıştır. Softmax, çıkış katmanına gelen girdileri normalize eden ve hangi sınıfa ait olduklarına dair [0,1] aralığından olasılıklarını döndüren çıkış katmanına ait bir aktivasyon fonksiyonudur. Çıkış katmanının da eklenmesiyle beraber tamamlanan model, ardından model.compile() işlevi ile derlenir. Bu fonksiyon içerisinde optimizer adı alan parametre, model eğitilirken verimliliğini arttırmaya amaçlayan algoritmanın fonksiyona atanması için tanımlanır. Bu deneyde açılımı 'adaptive moment estimation' olan, bellek gereksinimi düşük

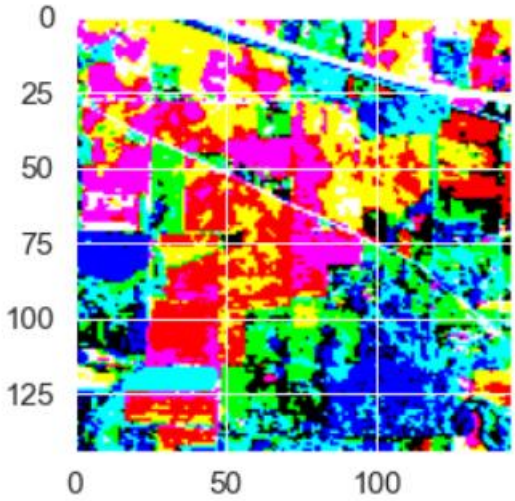
ve verimli hesaplama yapabilen ‘adam’ algoritması kullanılmıştır. Bir diğer parametre olan loss ise modelin kaybını hesaplayarak ne kadar iyi öğrendiğini ölçmeye yarayan ve verimi arttırmaya çalışan uygulamadır. Bir diğer parametre olan metrics=['accuracy'] ise loss parametresi gibi modelin performansını değerlendirir. Derleme işlevleri tamamlandıktan sonra model.fit() fonksiyonu ile model uydurma yapılarak, modelin kullandığı eğitim verilerinden programın ne kadar öğrendiği ve buna göre ne kadar doğru sonuç verdiği ölçülür. Bu fonksiyon, veriden ayrılan eğitim setlerinden, eğitim güncellemesi yaparken seçilen eğitim örneği sayısı belirlemede kullanılan batch_size parametresinden ve oluşturulan modelin çalıştırma tekrarının belirlendiği epochs parametresinden oluşmaktadır. Model eğitimi yapıldıktan sonra model.predict_classes() fonksiyonu ile hiperspektral görüntüye ait veri setinden ayrılan test seti üzerinden tahmin yapılması sağlanır. Bu fonksiyona atanan parametrenin oluşturduğu dizi ile sınıf bilgilerinin bulunduğu test verileri dizisi confusion_matrix() fonksiyonu kullanılarak beraber bir karışıklık matrisi oluşturur. Karışıklık matrisi program aracılığıyla ekrana yazdırılarak tahmin değerleri görselleştirilebilir. Böylelikle doğru ve hatalı tahminler tespit edilebilir. Başarı oranı, bu matrisin köşegenlerine ait olan doğru tahmin sayılarının toplamının, matrisin bu bölge dışında kalan alandaki hatalı tahminleri gösteren değerlerin toplamına oranı ile hesaplanır. Hesaplama işlemi deneyde accuracy_score() fonksiyonuna karışıklık matrisini oluşturan dizilerin yazılması ile hesaplanabilmektedir. Sonraki adımda model.evaluate() fonksiyonu test verilerini kullanarak, model derleme işlemlerinin yapıldığı adımda hesaplanan ‘loss’ ve metrics=['accuracy']) yani kayıp ile başarı oranlarını bulur. Son olarak deneyde tahmin edilen değerler görülmek istenirse model.predict() fonksiyonuna bir nesne atanarak test verisi üzerinden işlenir ve bu nesne herhangi bir parametre ile çağırıldığında bulduğu diziyi ekrana verir. Benzer şekilde eğitim verisinde de parametre belirtilerek bulunan dizi print() fonksiyonu ile yazdırılabilir.

4 ANALİZLER

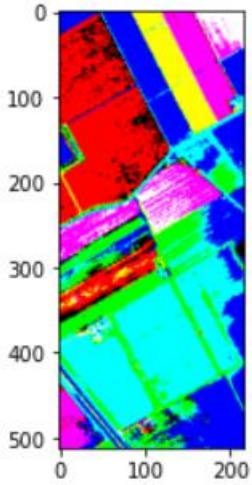
Bu bölümde hiperspektral görüntüler üzerinde uygulanan temel bileşenler analizi, destek vektör makineleri ve yapay sinir ağları deneylerinden elde edilen değerler, yapılar ve sonuçlar analiz edilmektedir.

4.1 Temel Bileşenler Analizi (TBA)

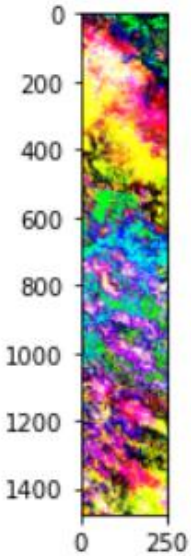
Temel bileşenler analizinde deney bölümünde anlatılan adımların uygulanmasıyla yüksek bant sayısına sahip hiperspektral görüntülerin boyut indirgeme ile 3 bantlı RGB renk uzayına konumlanması sağlanmaktadır. Böylelikle görüntü, temel özellikler korunmakla beraber gürültü ve üst üste binen bilgi fazlalığından arındırılır. Indian Pines, Botswana ve Salinas veri setlerine ait hiperspektral görüntüler, insan gözünün göremediği dalga boylarına ait renk uzayları da barındırmaktadır. Temel bileşenler analizi sonucunda RGB renk uzayı ile ifadesi sağlanabilmiştir. Böylelikle farklı deneylerde karşılaşılabilecek zorlu işlemlerde temel bileşenler analizi kullanılmasıyla veri setinin boyutu indirgenerek zaman ve işlem kazancı sağlanabilir. Buna ek olarak görüntü boyutundan dolayı programda ekrana batırabilmek mümkün değilken bu algoritmadan yeni boyutunu görüntülemek mümkündür. Veri setlerinden elde edilen farklı bant ve boyutlardaki temel bileşenler analizi yapılmış görüntüler ve matris ifadeleri aşağıda verilmiştir. Şekil 4.1'de yer bilgisi RGB uzayına göre renklendirilen (145, 145, 3) şekline sahip, 3 boyutlu renk uzayına indirgenen Indian Pines hiperspektral görüntüsü gösterilmiştir. Aynı temel bileşen analizi uygulamaları yapılmış Salinas veri setine ait (512, 217, 3) boyutundaki görüntü Şekil 4.2'de ve Botswana veri setine ait (1476, 256, 3) boyutundaki görüntü ise Şekil 4.3'te verilmiştir. Elde edilen bu zemin görüntüleri bantlarda bulunan önemli bilgileri içermekte olduğundan farklı deneylerde kullanılarak daha verimli sınıflandırma ve regresyon işlemleri uygulanabilir.



Şekil 4.1: Indian Pines veri setinin temel bileşenler analizi sonrası RGB renk uzayına konumlandırılmış görüntüsü



Şekil 4.2: Salinas veri setinin temel bileşenler analizi sonrası RGB renk uzayına konumlandırılmış görüntüsü

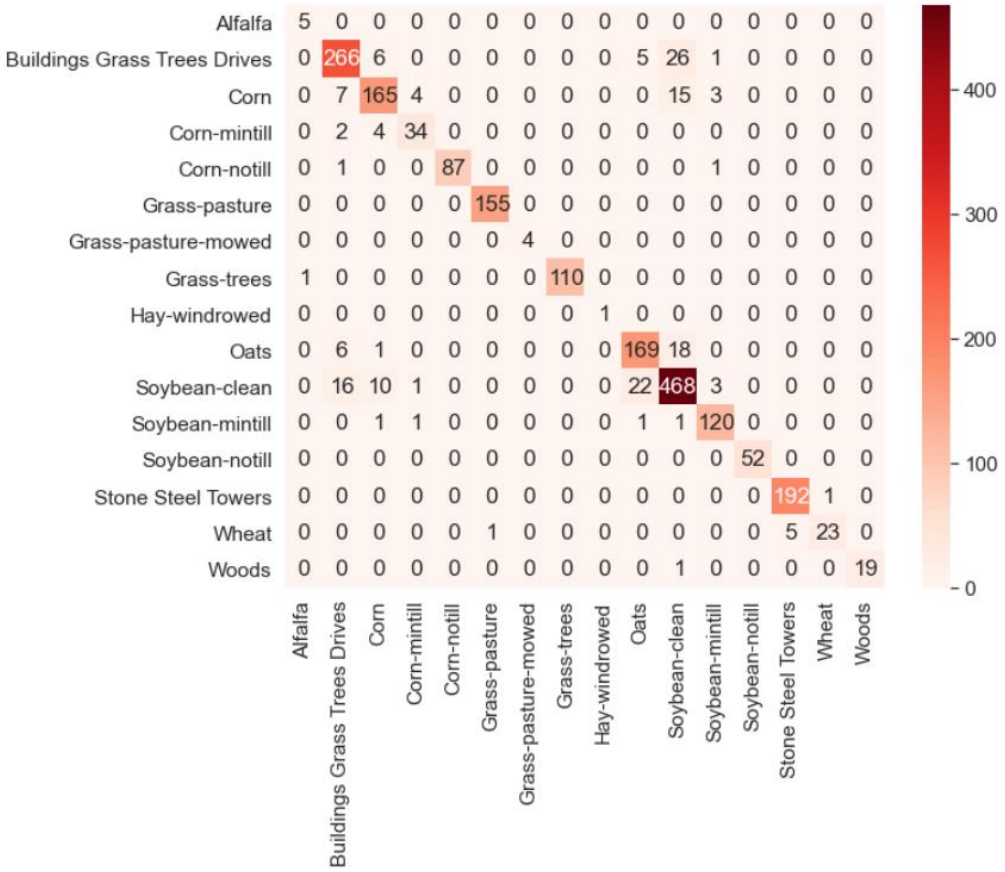


Şekil 4.3: Botswana veri setinin temel bileşenler analizi sonrası RGB renk uzayına konumlandırılmış görüntüsü

4.2 Destek Vektör Makineleri (DVM)

Çekirdek fonksiyonları kullanılarak uygulanan algoritmalarından destek vektör makineleri, deney bölümünde anlatılan adımların uygulanması sonucunda bu çalışmada kullanılan veri setlerinde sınıflandırma işlemleri yapmaktadır. Deney bölümünde Indian Pines üzerinden anlatılan uygulama diğer veri setlerinde de aynı adımları içermektedir. Yalnızca indirilen veri setleri farklı bölgelere ait hiperspektral görüntüler olduğundan sınıfları, bant sayıları ve pikselleri birbirinden farklıdır ve dolayısıyla aynı işlemler farklı işlem sürelerinde ve doğrulukta sonuçlar vermektedir. Bunun yanı sıra sınıflandırma fonksiyonunda eğitim verileri kullanıldığı ve tahmin modeli için ise test verilerinden yararlanıldığında her çalıştırmada farklı rastgele sayılar içeren veriler elde etmemek adına bu alt kümeler oluşturulurken random_state ifadesi sabit bir sayıya eşitlenir. Bu sebeple farklı sayılarda çalıştırma yine aynı sonucu verecektir. Eğer destek vektör makineleri algoritmasında sınıflandırma için tahmin doğruluğu arttırılmak istenirse sınıflandırma fonksiyonundaki değerlerde oynama yapılabilir fakat bu işlemler deney süresinin çok uzamasına neden olabilmektedir. Bu nedenle oluşturulan sınıflandırma sisteminin, denemelerle ile zaman-doğruluk açısından en verimli hali saptanabilmektedir. Bu bölümde 16 sınıftan oluşan Indian Pines ve Salinas veri setleri ile 14 sınıftan oluşan Botswana veri seti analiz edilerek başarı oranları yorumlanmıştır. Şekil 4.4'te Indian Pines veri setinin destek vektör makinesi uygulamasının sonucunda elde edilen karışıklık matrisinin grafiği yer almaktadır. Karışıklık matrisi, sınıfların olduğu test verileri ile hiperspektral görüntünün bulunduğu verinin alt kümesi olan test verisinden elde edilen tahmin verisinden oluşmaktadır. Bir araya getirilen bu iki dizi karışıklık matrisinin grafiği ile doğru ve yanlış tahmin edilen sınıfları ve değerlerini göstermektedir. Bu grafiğin yanında belirtilen renk skalası ise değerin büyüklüğüne göre grafikteki köşegenleri koyu işaretlemektedir. Test ile tahminin doğru kesiştiği köşegen noktaları doğru yapılan sınıflandırmaları gösterirken köşegenler haricindeki her sayı değeri yanlış tahmini göstermektedir. Örneğin Şekil 4.4'te görülebileceği üzere 15 tane Corn sınıfına ait görüntü program tarafından karıştırılarak Soybean-clean sınıfı olarak tahmin edilmiştir. En yüksek değere sahip köşegen olan Soybean-clean sınıfı, 468 tane test verisinin doğru tahmin edildiğini göstermektedir. Karışıklık matrisine ait grafik bu şekilde okunarak değerlendirilebilmekte ve doğruluk oranı hesaplanabilmektedir. Doğruluk oranı köşegenlerdeki değerlerin toplamının tüm matris değerleri toplamına oranıdır. Bu başarı oranı programda accuracy_score() fonksiyonuna yine karışıklık matrisine yazılan dizilerin girilmesi ile 0-1 aralığında hesaplanmaktadır. Şekil 4.5'te görüleceği üzere Indian Pines için elde edilen başarı oranı '0.8004565341449495' yani %80'dir. Bir diğer hiperspektral görüntüye ait olan veri seti Salinas için de aynı destek vektör makineleri algoritmaları uygulanarak yapılan sınıflandırma işlemleri sonucu elde edilen karışıklık matrisi grafiği Şekil 4.6'da verilmiştir. Indian Pines veri setine kıyasla çok daha fazla sayıda örnek içeren Salinas veri setinde yapılan sınıflandırma ile tahmin edilen veriler de dolayısıyla daha fazla bulunmaktadır. Indian Pines ile tahmin edilen en düşük sayıdaki sınıf 5 tane örnekten oluşurken Salinas veri setinde bu sayı 207'dir. Bununla beraber Indian Pines veri setine yapılan sınıflandırmada hatalı tahmin sayısı en çok 26 iken Salinas veri setinde bu sayı 438'e kadar çıkmaktadır. Ancak Salinas veri setinde

yanlış tahmin edilen sınıf sayısı çok azdır ve 438 tane yanlış tahmin edilen Vinyard_vertical_trellis sınıfı 1299 adet doğru tahmin içermektedir. Bu durum da oran olarak bakıldığında daha başarılı bir sonuç vermektedir. Nitekim doğruluk oranı hesaplandığında, Salinas veri setine ait sınıflandırma Şekil 4.7'de görüleceği üzere, '0.9245751728110599' değerinde yani %92'lik bir yüzdede başarı oranına sahiptir. Buradan da anlaşılabileceği üzere örnek sayısının artması programın daha iyi öğrenmesini ve bu durum da deneylerde doğrudan daha yüksek bir başarı oranı alınmasını sağlamaktadır. Bir diğer yandan Botswana data setine ait başarı oranı ise Şekil 4.9'de verilmiştir. Şekil 4.8'de de görüldüğü üzere sınıf tahmin sayısı oldukça az olmasına karşın tespit edilenlerin tamamı doğru tahmin edildiği yani hatalı tahmin sayısı 0'a eşit olduğu için Şekil 4.9'da da görüleceği üzere başarı oranı '0.9923886348238482' yani %99 olarak çok yüksek hesaplanmaktadır.



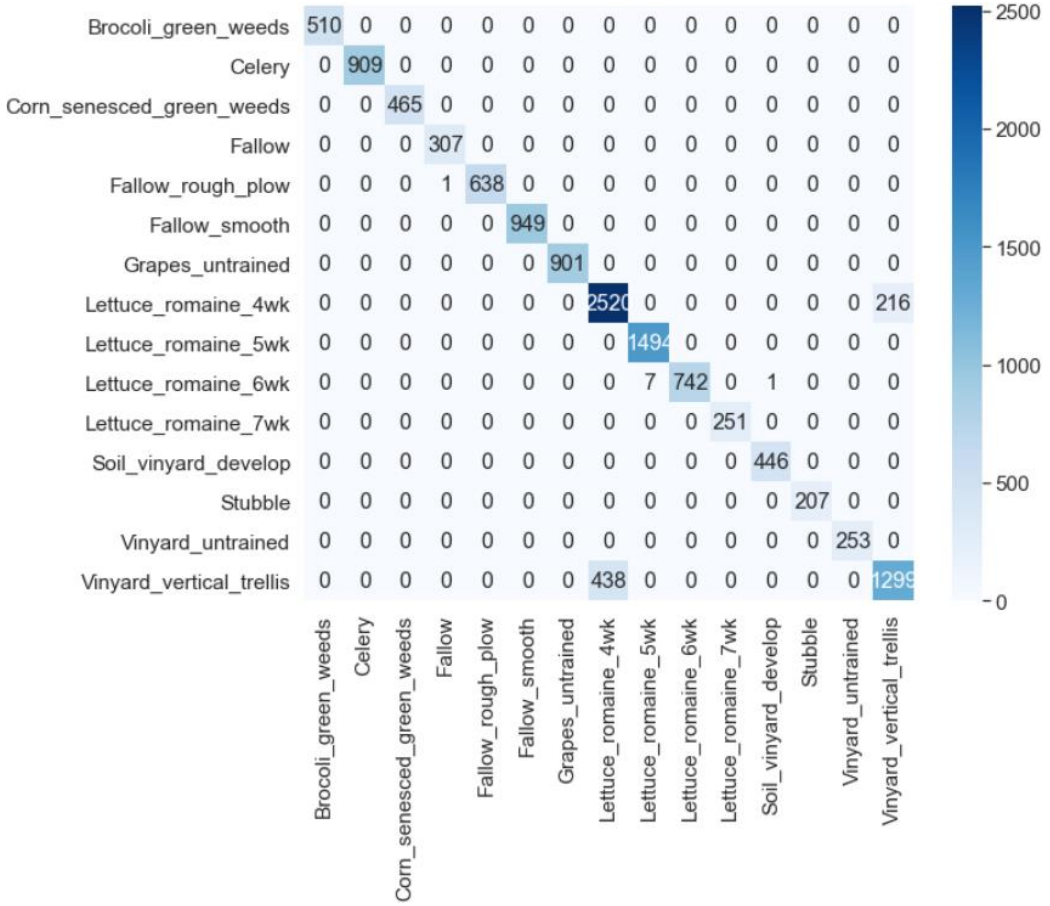
Şekil 4.4: Indian Pines veri seti üzerinde DVM ile uygulanan sınıflandırma işlemine ait karışıklık matrisi

```
#Indian Pines
from sklearn.metrics import accuracy_score

accuracy_score(Y_Pred, Y_Test)
```

```
0.8004565341449495
```

Şekil 4.5: Indian Pines veri setine uygulanan DVM algoritmasının sınıf tahmin başarısı



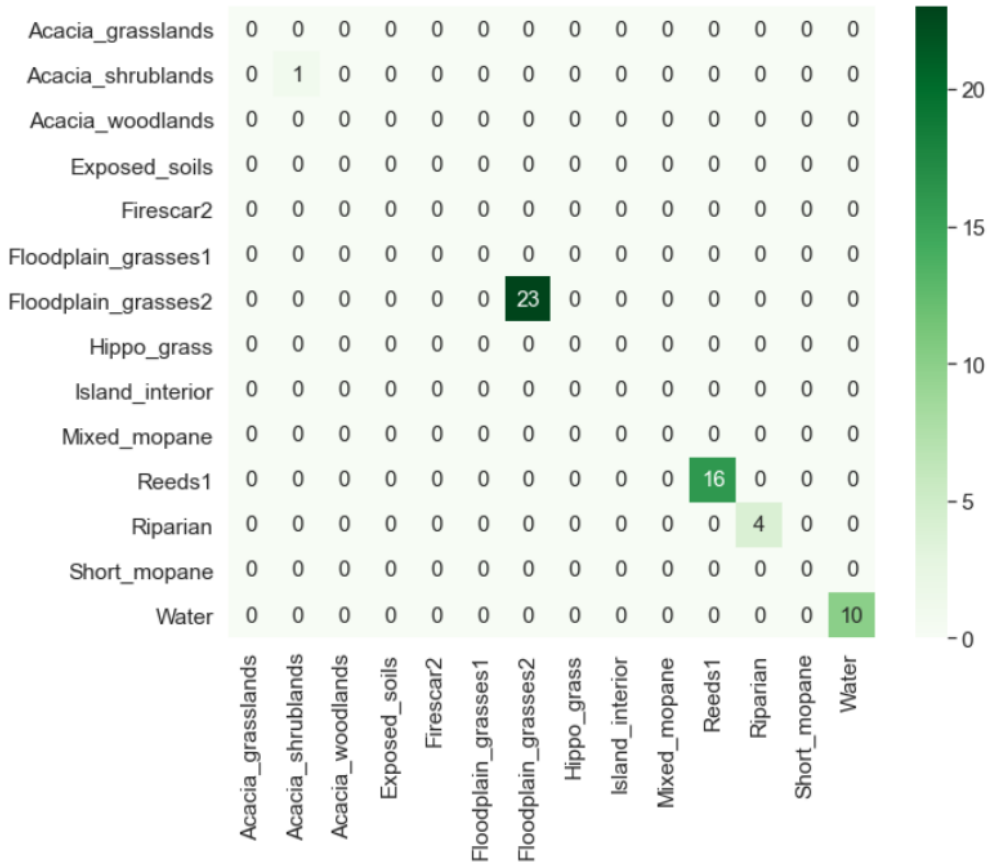
Şekil 4.6: Salinas veri seti üzerinde DVM ile uygulanan sınıflandırma işlemine ait karışıklık matrisi

```
from sklearn.metrics import accuracy_score

accuracy_score(Y_Pred, Y_Test)
```

```
0.9245751728110599
```

Şekil 4.7: Salinas veri setine uygulanan DVM algoritmasının sınıf tahmin başarısı



Şekil 4.8: Botswana veri seti üzerinde DVM ile uygulanan sınıflandırma işlemine ait karışıklık matrisi

```
from sklearn.metrics import accuracy_score

accuracy_score(Y_Pred, Y_Test)
```

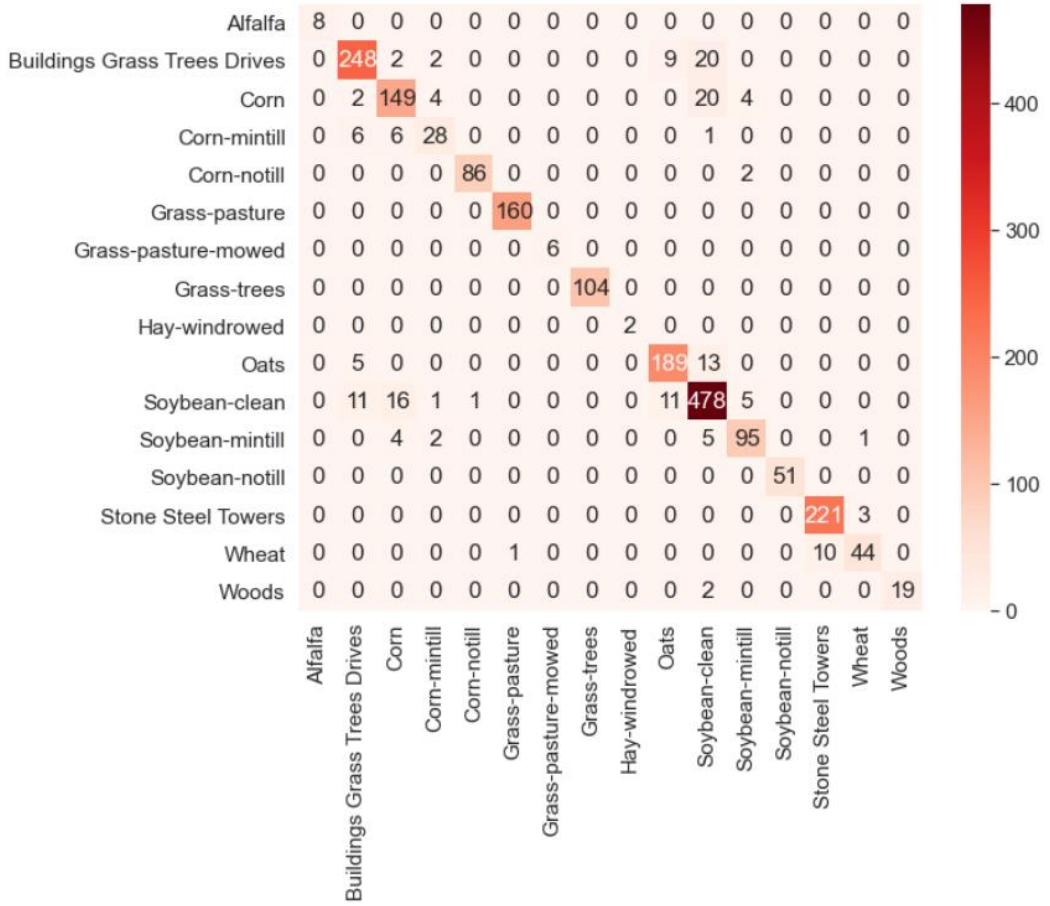
```
0.9923886348238482
```

Şekil 4.9: Botswana veri setine uygulanan DVM algoritmasının sınıf tahmin başarısı

4.3 Yapay Sinir Ağları (YSA)

Yapay sinir ağları uygulanarak oluşturulan model yardımıyla, deneyde anlatılan adımların gerçekleştirilmesi ile hiperspektral veri setleri üzerinde sınıflandırmalar gerçekleştirilmiştir. Destek vektör makinelerinde olduğu gibi eğitim ve test verilerinden yararlanılan yapay sinir ağları algoritmaları sınıflandırma yaparken destek vektör makinelerinden farklı olarak algoritmaya ait fonksiyon üzerinden değil, sinir ağları modeli kullanmaktadır. Oluşturulan modelin katman sayısı ile katmanlarda bulunan nöron sayısı deney hızını ve doğruluğunu değiştirebilmektedir. Katmanların yapıları ve nöron sayıları değiştirilerek farklı modeller oluşturulabilir. Elde edilen modelin eğitim işlemleri ardından destek vektör makinelerinde olduğu gibi test verisi kullanılarak elde edilen tahminler ile sınıf bilgisinin bulunduğu test verisinden karışıklık matrisi oluşturulmaktadır. İki dizinin karışıklık matrisine ait köşegenler doğru tahmin edilen sınıfların sayısını gösterirken köşegen dışında kalan bölgelerdeki değerler hatalı tespitlerin sayısını vermektedir. Başarı oranı, destek vektör makineleri deneyinin analizinde de anlatıldığı gibi doğru tahminlerin yani köşegen elemanlarının toplamının tüm matris elemanlarının toplamına oranı ile hesaplanmaktadır. Indian Pines veri setinde Şekil 4.11'de görüldüğü üzere doğruluk '0.7869507074356079' olarak %78 başarı oranında ölçülürken 'loss' yani kayıp ise '0.79703289270401' yani %79 oranında bulunmuştur. %70 üstünde hesaplanan başarı iyi kabul edilebilir olsa da kayıp oranını düşürmek bu değeri daha başarılı hale getirecektir. Diğer bir yandan destek vektör makineleri ile yapılan sınıflandırma deneyinin karışıklık matrisi Şekil 4.4 ile yapay sinir ağları deneyinden elde edilen karışıklık matrisi Şekil 4.10 kıyaslandığında bazı sınıflandırmalarda daha fazla doğru tahmin içerirken bazılarında daha az doğru tahmin bulundurduğu görülmektedir. Ancak sonuç olarak başarı oranına bakıldığında yapay sinir ağları ile sınıflandırma %78, destek vektör makinelerini ise %80 oranında doğruluk oranı elde ederek daha başarılı bir sınıflandırma yapmıştır. Aynı yapay sinir ağları deneyinin uygulandığı Salinas veri setinin karışıklık matrisi Şekil 4.12 ile gösterilmektedir. Yine destek vektör makinesi ile yapılan sınıflandırma ile kıyaslama yapılırsa bazı sınıflardaki doğruluk tahminin daha yüksek olmasının yanı sıra bazılarının ise daha düşük olduğu ve hatalı tespit edilmiş sınıf sayısının da destek vektör makinelerinden daha az bulunduğu görülecektir. Birbirine çok yakın sınıflandırma değerleri içeren iki algoritmanın Salinas verisi üzerinden doğruluk oranı kıyaslandığında Şekil 4.13'de görüldüğü üzere yapay sinir ağları deneyinden elde edilen oran '0.9342597723007202' yani %93 olarak ölçülerek destek vektör makinelerinden küçük bir farkla daha iyi bir sınıflandırma sonucu verdiği görülmüştür. Bununla beraber 'loss' yani kayıp oranı ise '0.2041158229112625' değerinde bulunarak %20 ölçülmüştür. Indian Pines veri setine kıyasla oldukça düşük bir kayıp oranına sahip olan Salinas veri setine bakılarak, başarı oranını arttırmak için kayıp oranını düşürmenin gerekli olduğu yorumu yapılabilir. Görüldüğü üzere kayıp oranı azaldıkça başarı oranı artmaktadır. Ayrıca veri setinde bulunan örnek sayısının fazla olması başarı oranını, destek vektör makinelerinde olduğu gibi yapay sinir ağları ile yapılan sınıflandırmada da olumlu etkilediği görülmektedir. Ancak bir diğer yandan Botswana veri setinin DVM algoritmasına ait deneyde olduğu gibi Şekil 4.14'te de görüldüğü üzere oldukça düşük sayıda sınıf tahmini

yaptığı görülmektedir. Bununla beraber Şekil 4.15'te görüldüğü üzere yine DVM algoritmasında olduğu gibi Botswana veri setinin hatalı tahmini olmadığından %99 gibi yüksek bir başarı oranı ve oldukça düşük kayıp elde etmektedir.

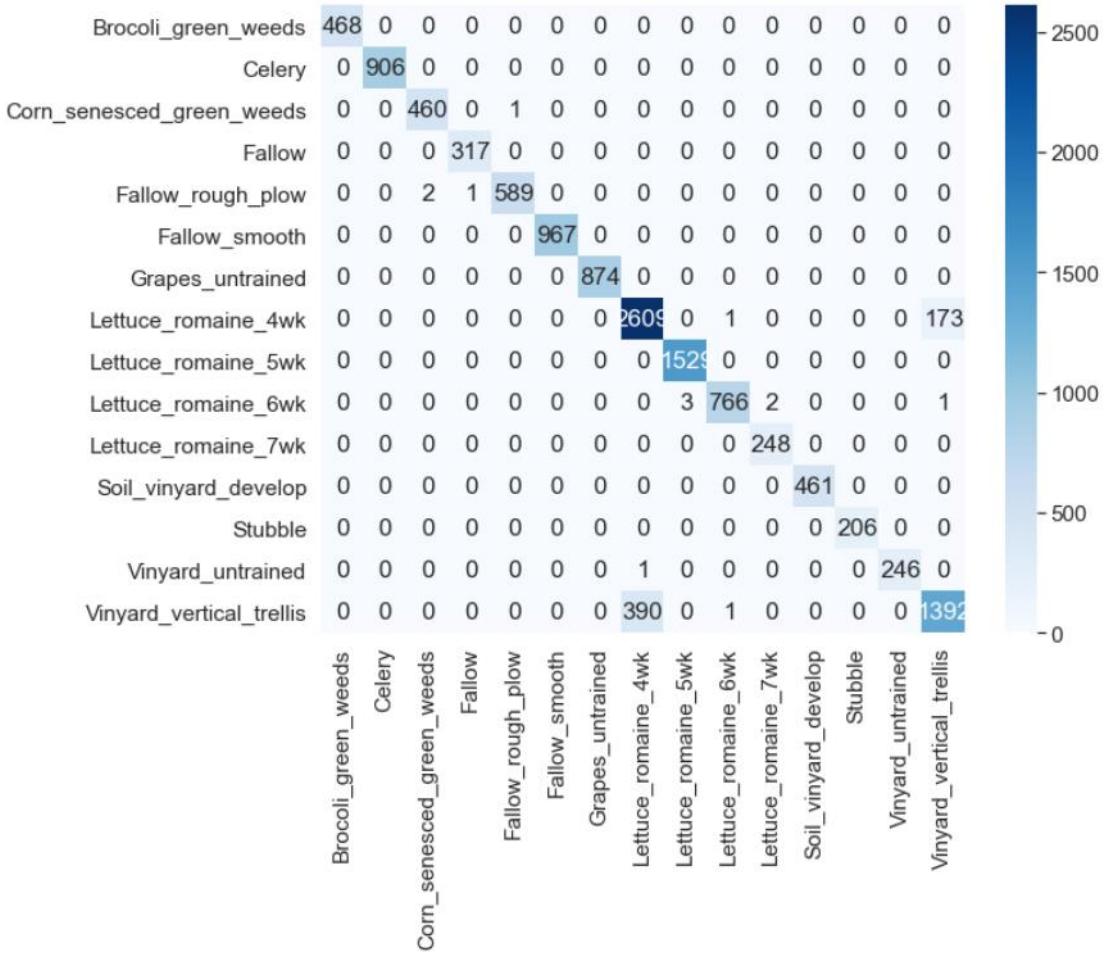


Şekil 4.10: Indian Pines veri setine, YSA ile uygulanan sınıflandırma işlemine ait karışıklık matrisi

```
val_loss, val_accuracy = model.evaluate(x_test, y_test)
print(val_loss, val_accuracy)
```

```
165/165 [==============================] - 0s 991us/step - loss: 0.7970 - accuracy: 0.7870
0.79703289270401 0.7869507074356079
```

Şekil 4.11: Indian Pines veri setine uygulanan YSA algoritmasının sınıf tahmin başarısı ve kayıp oranı

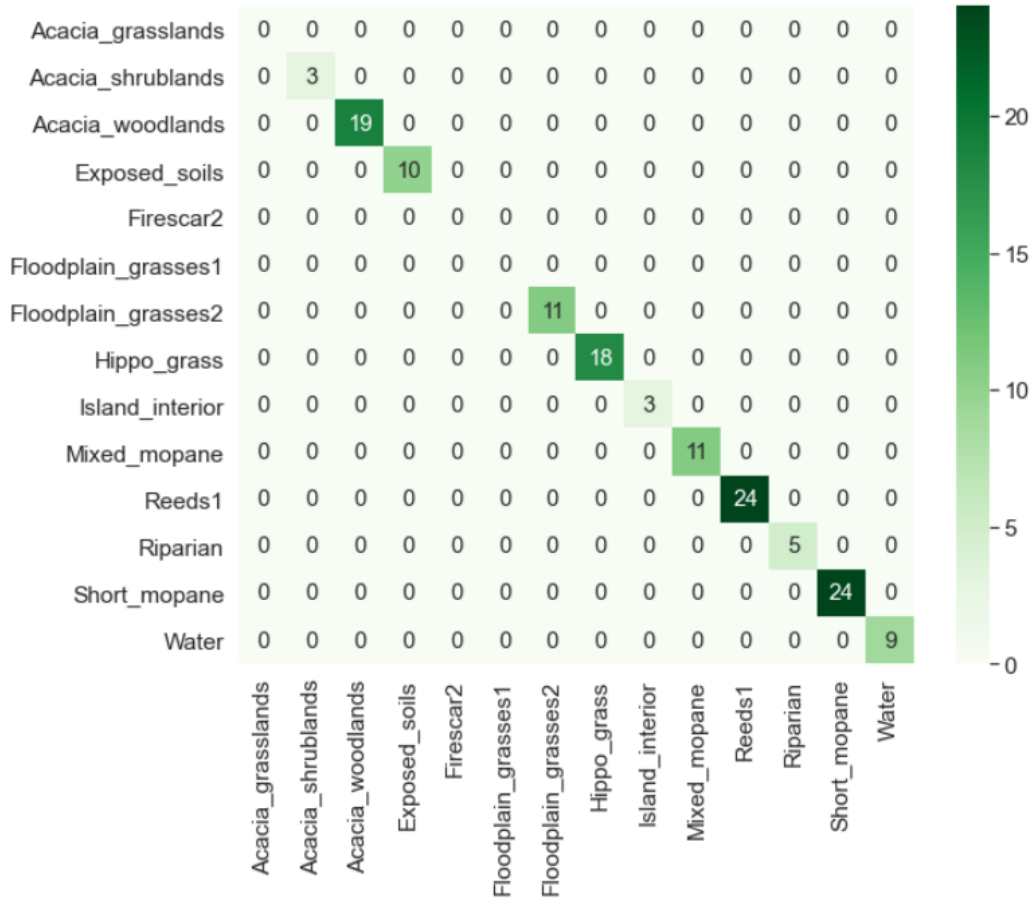


Şekil 4.12: Salinas veri setine, YSA ile uygulanan sınıflandırma işlemine ait karışıklık matrisi

```
val_loss, val_accuracy = model.evaluate(x_test, y_test)
print(val_loss, val_accuracy)
```

```
868/868 [==============================] - 1s 952us/step - loss: 0.2041 - accuracy: 0.9343
0.2041158229112625 0.9342597723007202
```

Şekil 4.13: Salinas veri setine uygulanan YSA algoritmasının sınıf tahmin başarısı ve kayıp oranı



Şekil 4.14: Botswana veri setine, YSA ile uygulanan sınıflandırma işlemine ait karışıklık matrisi

```
val_loss, val_accuracy = model.evaluate(x_test, y_test)
print(val_loss, val_accuracy)
```

```
2952/2952 [==============================] - 3s 938us/step - loss: 0.0693 - accuracy: 0.9909
0.06932112574577332 0.9908536672592163
```

Şekil 4.15: Botswana veri setine uygulanan YSA algoritmasının sınıf tahmin başarısı ve kayıp oranı

5 Kaynaklar

[1] Kang, X., Li, S., & Benediktsson, J. A. (2014). Spectral–Spatial Hyperspectral Image Classification With Edge-Preserving Filtering. *IEEE Transactions on Geoscience and Remote Sensing*, *52*(5), 2666–2677.

[2] Govender, M., Chetty, K., & Bulcock, H. (2007). A review of hyperspectral remote sensing and its application in vegetation and water resource studies. *Water SA*, *33*(2), 145–152.

[3] Yang, X., Ye, Y., Li, X., Lau, R. Y. K., Zhang, X., & Huang, X. (2018). Hyperspectral Image Classification With Deep Learning Models. *IEEE Transactions on Geoscience and Remote Sensing*, 1–17.

[4] Tiwari, K. C., Arora, M. K., & Singh, D. An assessment of independent component analysis for detection of military targets from hyperspectral images. International Journal of Applied Earth Observation and Geoinformation, 13(5), 730–740.

[5] Liu, Z., Wang, H., & Li, Q. (2012). Tongue Tumor Detection in Medical Hyperspectral Images. *Sensors*, *12*(1), 162–174.

[6] Du, Q., & Yang, H. (2008). Similarity-Based Unsupervised Band Selection for Hyperspectral Image Analysis. *IEEE Geoscience and Remote Sensing Letters*, *5*(4), 564–568.

[7] Altaher, A. W., & Abbas, S. K. (2020). Image processing analysis of sigmoidal Hadamard wavelet with PCA to detect hidden object. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, *18*(3), 1216–1223.

[8] Rodarmel, C., & Shan, J. (2002). Principal Component Analysis for Hyperspectral Image Classification.

[9] Genc, L., & Smith, S. (2016). Assessment Of Principal Component Analysis (PCA) For Moderate and High Resolution Satellite Data.

[10] Gowtham, B., Reddy, I. A. K., Reddy, T. S., Harikiran, J., & Chandana, B. S. (2021). Hyperspectral Image Analysis using Principal Component Analysis and Siamese Network. *Turkish Journal of Computer and Mathematics Education*, *12*(7), 1191–1198.

[11] Deepa, P., & Thilagavathi, K. (2015). Feature extraction of hyperspectral image using principal component analysis and folded-principal component analysis. *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, 656-660.

[12] Melgani, F., & Bruzzone, L. (2004). Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, *42*(8), 1778–1790.

[13] Moughal, T. A. (2013). Hyperspectral image classification using Support Vector Machine. *Journal of Physics: Conference Series*, *439*(1).

[14] Gönen, M., & Alpaydin, E. (2011). Multiple Kernel Learning Algorithms. *J. Mach. Learn. Res., 12*, 2211-2268.

[15] Mehrotra, K., Mohan, C. K., & Ranka, S. (1996). *Elements of Artificial Neural Nets*. A Bradford Book.

[16] Abraham, A. (2005). 129 Artificial Neural Networks.

[17] Malik, Nitin. "Artificial Neural Networks And Their Applications" National Conference on 'Unearthing Technological Developments & their Transfer for Serving Masses' GLA ITM, Mathura, India 17-18 April (2005).

6 EKLER

Bu bölümde, çalışmada yapılan deneylerin Python yazılım diline ait kodları verilmiştir.

EK-A.1 : Indian Pines veri setine ait TBA kodu
EK-A.2 : Indian Pines veri setine ait DVM kodu
EK-A.3 : Indian Pines veri setine ait YSA kodu
EK-B.1 : Salinas veri setine ait TBA kodu
EK-B.2 : Salinas veri setine ait DVM kodu
EK-B.3 : Salinas veri setine ait YSA kodu
EK-C.1 : Botswana veri setine ait TBA kodu
EK-C.2: Botswana veri setine ait DVM kodu
EK-C.3: Botswana veri setine ait YSA kodu

EK-A.1

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.io import loadmat
from pandas import read_csv
from sklearn.decomposition import PCA

pine = loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Indian_pines
(1).mat')['indian_pines']
pinere = pine.reshape((145*145, 220))
pine.shape

pca = PCA(n_components = 3)
pca.fit(pine_gt)
pine_re_pca = pca.fit_transform(pine_gt)
pine_re_pca.shape

pine_re_pca.shape
pca.fit_transform(pine_re)

resim = pine_re_pca.reshape((145, 145, 3))
plt.imshow(resim)
```

EK-A.2

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.io import loadmat
import seaborn as sn
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split

X= loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Indian_pines
(1).mat')['indian_pines']

Y= loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Indian_pines_gt
(1).mat')['indian_pines_gt']

names = ['Alfalfa',      'Corn-notill', 'Corn-mintill',    'Corn', 'Grass-pasture','Grass-trees',
'Grass-pasture-mowed','Hay-windrowed','Oats','Soybean-notill','Soybean-mintill',
'Soybean-clean', 'Wheat',  'Woods',   'Buildings Grass Trees Drives', 'Stone Steel Towers']

x_ = X.reshape((145*145, 220))
y_ = Y.reshape((145*145, 1))
print(x_.shape, y_.shape)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()

x = sc_X.fit_transform(x_)
X_Train,  X_Test, Y_Train, Y_Test = train_test_split(x, y_, test_size = 0.25, random_state = 0)

print(Y_Train.shape)

classifier =  SVC(C = 100, kernel = 'rbf', cache_size = 10*1024)
classifier.fit(X_Train, Y_Train)

Y_Pred = classifier.predict(X_Test)
print(Y_Pred)

from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(Y_Test, Y_Pred)

print(cm)
cm.shape
df_cm = pd.DataFrame(cm[1:17, 1:17] , columns=np.unique(names), index =
np.unique(names))

plt.figure(figsize = (10,8))
sn.set(font_scale=1.4) #for label size
sn.heatmap(df_cm, cmap="Reds", annot=True,annot_kws={"size": 17}, fmt='d')

plt.savefig('cmap.png', dpi=300)
from sklearn.metrics import accuracy_score

accuracy_score(Y_Pred, Y_Test)
```

EK-A.3

```
import tensorflow as tf
from scipy.io import loadmat
from sklearn.model_selection import train_test_split
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np


X=loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Indian_pines(1).mat')['india
n_pines']

Y=loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Indian_pines_gt(1).mat')['in
dian_pines_gt']

x_ = X.reshape((145*145, 220))
y = Y.reshape((145*145, 1))

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()

x = sc_X.fit_transform(x_)

(x_train, x_test, y_train, y_test) = train_test_split(x, y, test_size = 0.25, random_state=0,
stratify=y)

x_train = tf.keras.utils.normalize(x_train, axis = 1)
x_test = tf.keras.utils.normalize(x_test, axis = 1)

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation = tf.nn.relu))
model.add(tf.keras.layers.Dense(256, activation = tf.nn.relu))
model.add(tf.keras.layers.Dense(17, activation = tf.nn.softmax))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=256, epochs=150)
```

```
y_pred =  model.predict_classes(x_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

names = ['Alfalfa',       'Corn-notill', 'Corn-mintill',     'Corn',  'Grass-pasture','Grass trees',
'Grass-pasture-mowed','Hay-windrowed','Oats','Soybean-notill','Soybean-mintill', 'Soybean-
clean', 'Wheat',   'Woods',      'Buildings Grass Trees Drives',  'Stone Steel Towers']

df_cm = pd.DataFrame(cm[1:17, 1:17] , columns=np.unique(names), index =
np.unique(names))

plt.figure(figsize = (10,8))

sn.set(font_scale=1.4) #for label size
sn.heatmap(df_cm, cmap="Reds", annot=True,annot_kws={"size": 17}, fmt='d')
plt.savefig('cmap.png', dpi=300)

from sklearn.metrics import accuracy_score

accuracy_score(y_pred, y_test)

val_loss, val_accuracy = model.evaluate(x_test, y_test)
print(val_loss, val_accuracy)

y_predicted = model.predict(x_test)
y_predicted[integer]

import numpy as np
print(np.argmax(y_predicted[integer]))
```

EK-B.1

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.io import loadmat
from pandas import read_csv
from sklearn.decomposition import PCA

salinas =
loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Salinas.mat')['salinas']

salinasre = salinas.reshape((512*217, 224))

pca = PCA(n_components = 3)
pca.fit(salinasre)
salinasre_pca = pca.fit_transform(salinasre)
salinasre_pca.shape

salinasre_pca.shape
pca.fit_transform(salinasre)

resim = salinasre_pca.reshape((512, 217, 3))
plt.imshow(resim)
```

EK-B.2

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.io import loadmat
import seaborn as sn
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split

X= loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Salinas.mat')['salinas']
Y=loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Salinas_gt.mat')['salinas_gt']


x_ = X.reshape((512*217, 224))
y_ = Y.reshape((512*217, 1))
print(x_.shape, y_.shape)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()

x = sc_X.fit_transform(x_)
X_Train,  X_Test, Y_Train, Y_Test = train_test_split(x, y_, test_size = 0.25, random_state = 0)

print(Y_Train.shape)

classifier =  SVC(C = 100, kernel = 'rbf', cache_size = 10*1024)
classifier.fit(X_Train, Y_Train)

Y_Pred = classifier.predict(X_Test)
print(Y_Pred)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_Test, Y_Pred)

print(cm)
cm.shape
```

```
names = ['Brocoli_green_weeds', 'Brocoli_green_weeds' ,'Fallow',
'Fallow_rough_plow','Fallow_smooth','Stubble','Celery' ,'Grapes_untrained'
,'Soil_vinyard_develop'
,'Corn_senesced_green_weeds','Lettuce_romaine_4wk','Lettuce_romaine_5wk','Lettuce_r
maine_6wk' ,'Lettuce_romaine_7wk','Vinyard_untrained','Vinyard_vertical_trellis']

df_cm = pd.DataFrame(cm[1:16, 1:16] , columns=np.unique(names), index =
np.unique(names))

plt.figure(figsize = (10,8))

sn.set(font_scale=1.4)

sn.heatmap(df_cm, cmap="Blues", annot=True,annot_kws={"size": 16}, fmt='d')

plt.savefig('cmap.png', dpi=300)

from sklearn.metrics import accuracy_score

accuracy_score(Y_Pred, Y_Test)
```

EK-B.3

```
import tensorflow as tf
from scipy.io import loadmat
from sklearn.model_selection import train_test_split
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np


X= loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Salinas.mat')['salinas']
Y=loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Salinas_gt.mat')['salinas_gt']

x_ = X.reshape((512*217, 224))
y = Y.reshape((512*217, 1))
print(x_.shape, y_.shape)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()

x = sc_X.fit_transform(x_)

(x_train, x_test, y_train, y_test) = train_test_split(x, y, test_size = 0.25, random_state=0,
stratify=y)

x_train = tf.keras.utils.normalize(x_train, axis = 1)
x_test = tf.keras.utils.normalize(x_test, axis = 1)
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation = tf.nn.relu))
model.add(tf.keras.layers.Dense(256, activation = tf.nn.relu))
model.add(tf.keras.layers.Dense(17, activation = tf.nn.softmax))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=256, epochs=150)

y_pred =  model.predict_classes(x_test)
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

names = ['Brocoli_green_weeds', 'Brocoli_green_weeds' ,'Fallow',
'Fallow_rough_plow','Fallow_smooth','Stubble','Celery' ,'Grapes_untrained'
,'Soil_vinyard_develop',
'Corn_senesced_green_weeds','Lettuce_romaine_4wk','Lettuce_romaine_5wk','Lettuce_ro
maine_6wk' ,'Lettuce_romaine_7wk','Vinyard_untrained','Vinyard_vertical_trellis']

df_cm = pd.DataFrame(cm[1:16, 1:16] , columns=np.unique(names), index =
np.unique(names))

plt.figure(figsize = (10,8))

sn.set(font_scale=1.4)

sn.heatmap(df_cm, cmap="Blues", annot=True,annot_kws={"size": 17}, fmt='d')

plt.savefig('cmap.png', dpi=300)

from sklearn.metrics import accuracy_score

accuracy_score(y_pred, y_test)

val_loss, val_accuracy = model.evaluate(x_test, y_test)
print(val_loss, val_accuracy)
y_predicted = model.predict(x_test)
y_predicted[integer]

print(np.argmax(y_predicted[integer]))
```

EK-C.1

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.io import loadmat
from pandas import read_csv
from sklearn.decomposition import PCA

botswana=loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Botswana.mat')['Bo
tswana']

botswanare = botswana.reshape((1476*256, 145))

pca = PCA(n_components = 3)
pca.fit(botswanare)
botswanare_pca = pca.fit_transform(botswanare)
botswanare_pca.shape

botswanare_pca.shape
pca.fit_transform(botswanare)

resim = botswanare_pca.reshape((1476, 256, 3))
plt.imshow(resim)
```

EK-C.2

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.io import loadmat
import seaborn as sn
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split

X=loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Botswana.mat')['Botswana']

Y=loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Botswana_gt.mat')['Botswa
na_gt']

x_ = X.reshape((1476*256, 145))
y_ = Y.reshape((1476*256, 1))
print(x_.shape, y_.shape)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()

x = sc_X.fit_transform(x_)
X_Train,  X_Test, Y_Train, Y_Test = train_test_split(x, y_, test_size = 0.25, random_state = 0)

print(Y_Train.shape)

classifier =  SVC(C = 100, kernel = 'rbf', cache_size = 10*1024)
classifier.fit(X_Train, Y_Train)

Y_Pred = classifier.predict(X_Test)
print(Y_Pred)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_Test, Y_Pred)

print(cm)
cm.shape

names = ['Water', 'Hippo_grass' ,'Floodplain_grasses1',
'Floodplain_grasses2','Reeds1','Riparian','Firescar2'
,'Island_interior' ,'Acacia_woodlands'
,'Acacia_shrublands','Acacia_grasslands','Short_mopane','Mixed_mopane'
,'Exposed_soils']
```

```
df_cm = pd.DataFrame(cm[1:15, 1:15] , columns=np.unique(names), index =
np.unique(names))

#df_cm.index.name = 'Actual'

#df_cm.columns.name = 'Predicted'

plt.figure(figsize = (10,8))

sn.set(font_scale=1.4) #for label size

sn.heatmap(df_cm, cmap="Pinks", annot=True,annot_kws={"size": 15}, fmt='d')

plt.savefig('cmap.png', dpi=30)

from sklearn.metrics import accuracy_score

accuracy_score(Y_Pred, Y_Test)
```

EK-C.3

```
import tensorflow as tf
from scipy.io import loadmat
from sklearn.model_selection import train_test_split
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt


X=loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Botswana.mat')['Botswana']
Y=loadmat(r'C:\Users\Tutku\PycharmProjects\pythonProject24\Botswana_gt.mat')['Botswa
na_gt']
x_ = X.reshape((1476*256, 145))
y = Y.reshape((1476*256, 1))
print(x_.shape, y.shape)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()

x = sc_X.fit_transform(x_)

(x_train, x_test, y_train, y_test) = train_test_split(x, y, test_size = 0.25, random_state=0,
stratify=y)

x_train = tf.keras.utils.normalize(x_train, axis = 1)
x_test = tf.keras.utils.normalize(x_test, axis = 1)

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation = tf.nn.relu))
model.add(tf.keras.layers.Dense(256, activation = tf.nn.relu))
model.add(tf.keras.layers.Dense(15, activation = tf.nn.softmax))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

import numpy as np

model.fit(x_train, y_train, batch_size=256, epochs=150)

y_pred =  model.predict_classes(x_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
names = ['Water', 'Hippo_grass' ,'Floodplain_grasses1',
'Floodplain_grasses2','Reeds1','Riparian','Firescar2'
,'Island_interior' ,'Acacia_woodlands'
,'Acacia_shrublands','Acacia_grasslands','Short_mopane','Mixed_mopane'
,'Exposed_soils']

df_cm = pd.DataFrame(cm[1:15, 1:15] ) columns=np.unique(names), index =
np.unique(names))


plt.figure(figsize = (10,8))

sn.set(font_scale=1.4) #for label size

sn.heatmap(df_cm, cmap="Reds", annot=True,annot_kws={"size": 15}, fmt='d')

plt.savefig('cmap.png', dpi=300)

from sklearn.metrics import accuracy_score

accuracy_score(y_pred, y_test)

val_loss, val_accuracy = model.evaluate(x_test, y_test)
print(val_loss, val_accuracy)

y_predicted = model.predict(x_test)
y_predicted[integer]
```