Python ile Pratik Makine Öğrenmesi Projeleri

Mehmet Akif BUZPINAR

SERÜVEN YAYINEVİ

Genel Yayın Yönetmeni / Editor in Chief • C. Cansın Selin Temana

Kapak & İç Tasarım / Cover & Interior Design • Serüven Yayınevi

Birinci Basım / First Edition • © Aralık 2023

ISBN • 978-625-6760-68-4

© copyright
Bu kitabın yayın hakkı Serüven Yayınevi'ne aittir.
Kaynak gösterilmeden alıntı yapılamaz, izin almadan hiçbir yolla çoğaltılamaz.
The right to publish this book belongs to Serüven Publishing. Citation can not be shown without the source, reproduced in any way without permission.

Serüven Yayınevi / Serüven Publishing
Türkiye Adres / Turkey Address: Kızılay Mah. Fevzi Çakmak 1. Sokak
Ümit Apt No: 22/A Çankaya/ANKARA
Telefon / Phone: 05437675765
web: www.seruvenyayinevi.com
e-mail: seruvenyayinevi@gmail.com

Baskı & Cilt / Printing & Volume
Sertifika / Certificate No: 47083

Python ile Pratik Makine Öğrenmesi Projeleri

Mehmet Akif BUZPINAR

SERÜVEN YAYINEVİ

İÇİNDEKİLER

ÖNSÖZ IX
GİRİŞ X
1. BÖLÜM 1
Giriş 1
1.1 Bir Şablon Üreterek Başlama 2
1.2 Projelerle Makine Öğrenimi 2
1.2.1 Adım Adım Yapılandırılmış Bir Akış Süreci İzleyin 3
1.3 Python'da Makine Öğrenimi Proje Şablonu 4
1.3.1 Şablon Özeti ve Kullanımı 4
1.3.3 Problemi Detaylandırarak Hazırlama 6
1.3.4 Veriyi Özetleme 6
1.3.3 Veriyi Hazırlama 7
1.3.4 Algoritmaları Değerlendirme 8
1.3.5 Doğruluğu (Accuracy) Yükseltme 8
1.3.6 Modeli Sonlandır 9
1.4 Şablonu İyi Kullanma İpuçları 9
1.5 Özet 10
1.6 Sonraki Adım 10
2. BÖLÜM 11
Giriş 11
2.1 Makine Öğrenimine Giriş: İris Çiçeği Sınıflandırma 12
2.2 Veriyi Yükleme 12
2.2.1 Kütüphaneleri Yükleme 13
2.2.2 Veri Setini Yükleme 13
2.3 Veri Kümesini Özetleme 13
2.3.1 Veri setinin Boyutları 14
2.3.2 Veriyi İnceleyin 14
2.3.3 İstatistiksel Özet 15
2.3.4 Sınıf Dağılımı 16
2.4 Veri Görselleştirme 17
2.4.1 Tek Değişkenli Grafikler 17
2.4.2 Çoklu Öznitelik Grafikleri 19
2.5 Bazı Algoritmaların Değerlendirilmesi 20
2.5.1 Bir Doğrulama Veri Kümesi Oluşturma 21

2.5.2 Model Testi (Test Harnes) 21
2.5.3 Modeller Oluşturma 22
2.5.4 En Başarılı modelin Seçimi 22
2.6 Tahminler Yapın 24
2.7 Özet 25
BÖLÜM 3 26
Giriş 26
3.1 Makine Öğrenmesiyle Regresyon Problemi 26
3.2 Veri Kümesini Yükleme 29
3.3 Veri Analizi 30
3.3.1 Tanımlayıcı İstatistikler 30
3.4 Veri Görselleştirmeleri 32
3.4.1 Tek Modlu Veri Görselleştirmeleri 32
3.4.2 Çok Modlu Veri Görselleştirme 35
3.4.3 Çıkarımların Özeti 37
3.5 Doğrulama Veri Seti 38
3.6 Algoritmaları Değerlendirme 38
3.7 Algoritmaları Değerlendirme: Standartlaştırma 40
3.8 Parametre Ayarlama ve Sonuçları İyileştirme 42
3.9 Ensemble Yöntemleri 44
3.10 Ensemble Yöntemleriyle Ayarlama Yapma 45
3.11 Modeli Sonlandırma 46
3.12 Özet 47
BÖLÜM 4 49
Giriş 49
4.1 Makine Öğrenmesiyle İkili Sınıflandırma Problemi 49
4.2 Veri setini Yükleme 49
Liste 4.1 Kütüphanelerin yüklenmesi 50
Liste 4.2 Veri setini yükleme 50
4.3 Veriyi Analiz Etme 50
4.3.1 Tanımlayıcı İstatistik 50
4.3.2 Tek Modlu Veri Görselleştirme 54
4.3.3 Çok Modlu Veri Görselleştirme 57
4.4 Doğrulama Veri Seti 57
4.5 Algoritmaları Değerlendirme 58

4.6 Algoritmaların Standardize Veri Seti ile Değerlendirilmesi..... 60
4.7 Başarılı Algoritmalarda Parametre Ayarlama..... 62
4.7.1 KNN'i Ayarlama..... 62
4.8 Birleştirilmiş (Ensemble) Yöntemler 65
4.9 Boosting Yöntemleri: AdaBoost (AB) ve Gradient Boosting (GBM) 66
4.9 Modelin Sonlandırılması..... 67
4.10 Özet..... 68
4.11 Sonraki Adım..... 68
5. BÖLÜM MAKİNE ÖĞRENMESİYLE KÜMELEME PROBLEMİ . 69
Giriş 69
5.1 Problemin Tanımı 69
5.2 Veri setini Tanıma 69
5.3 Gerekli Kütüphaneleri Yükleme..... 71
5.3 Veriyi Analiz Etme 72
5.3.1 Tanımlayıcı İstatistik..... 72
5.3.2 Veriyi Temizleme..... 74
5.4 Kümeleme Algoritmaları ile Modelleme..... 75
5.4.1 K-Means Algortiması ile Modelleme 76
6.2 GMM (Gaussian Mixture Models) Algortiması ile Modelleme..... 78
6.3 MeanShift Algortiması ile Modelleme 81
6.4 Algoritmaları Değerlendirme..... 83
5.7 Özet..... 87
5.8 SONUÇ..... 88
KAYNAKLAR.....89

ÖNSÖZ

Sevgili Okuyucu,

"Python ile Pratik Makine Öğrenmesi Projeleri" kitabına hoş geldiniz! Bu kitap, makine öğrenmesi projeleri geliştirmek isteyen herkese pratik bir rehber sunmayı hedeflemektedir. Amaç, sizi adım adım yönlendirme yöntemiyle Python dilini kullanarak gerçek dünya projelerini nasıl hayata geçirebileceğinizi öğretmektir.

Makine öğrenmesi, günümüzün en heyecan verici yapay zekâ alanlarından biridir. Veri analizi, tahmin yapma, desenleri tanıma ve otomatik karar verme gibi birçok alanda büyük bir etki yaratmaktadır. Ancak, makine öğrenmesi projeleri geliştirmek karmaşık ve zorlu olabilir. Bu kitap, bu zorlukları aşmanıza yardımcı olacak pratik bilgiler ve gerçek dünya problem çözümleri sunmayı amaçlamaktadır.

GİRİŞ

Makine öğrenmesi projeleri geliştirmek, birçok farklı adım ve süreç içerir. Bu süreçlerin her biri, veri hazırlığından model oluşturmaya, modelin eğitiminden sonuçların değerlendirilmesine kadar çeşitli aşamalardan oluşur. Bu kitap, bu aşamaları uygulatacak ve makine öğrenmesi projelerinin nasıl gerçekleştirileceğini gösterecektir.

Kitabın içeriği, proje bazlı bir yaklaşıma dayanmaktadır. Her bölümde, bir makine öğrenmesi projesiyle ilgili adım adım talimatlar ve pratik ipuçları sunulmaktadır. Öncelikle, projenin amacı ve kullanılacak veri seti tanıtılır. Ardından, veri setinin keşfedilmesi, temizlenmesi ve özellik mühendisliği gibi önemli adımlar ele alınır.

Daha sonra, modelin oluşturulması ve eğitimi için Python dilinde kullanılan temel makine öğrenmesi kütüphaneleri tanıtılır. Modelin performansını değerlendirmek için farklı metrikler ve yöntemler kullanılır ve sonuçların yorumlanması için pratik bilgiler sunulur. Ayrıca, projenin sonuçlarını sunmak ve paylaşmak için veri görselleştirme ve raporlama tekniklerine de değinilir.

Bu kitap, temel seviye Python bilgisine sahip her okuyucuya hitap etmektedir. Makine öğrenmesi ve Python konularında hiç deneyiminiz olmasa bile, bu kitap size projelerinizi gerçekleştirmek için gerekli adımları öğretecektir. Ayrıca, deneyimli geliştiriciler için de pratik bilgiler ve ipuçları sunarak projelerinizi daha hızlı geliştirmenize yardımcı olacaktır.

Umarım bu kitap, sizin için destekleyici bir ilk adım kaynağı olur ve makine öğrenmesi projeleri konusundaki yeteneklerinize geliştirmenize yardımcı olur.

Saygılarımla,

Mehmet Akif BUZPINAR

1. BÖLÜM

Giriş

Makine öğrenmesi projeleri, günümüzün veri odaklı teknolojik ilerlemede merkezi bir yere sahiptir. Bu projeler, geniş veri setlerinden anlamlı öngörüler ve güçlü tahmin modelleri çıkarmayı hedefler ve bu nedenle veri analizi ve yapay zekâ çalışmalarında sıkça başvurulan yöntemlerdir. Ancak, bu projelerin başarıya ulaşabilmesi için, sadece teknik bilgi yeterli değildir; aynı zamanda projelerin etkin bir plan ve yöntemle yürütülmesi gerekmektedir.

Bir makine öğrenmesi projesi, çok sayıda karmaşık adımı barındırır: Veri toplamaktan başlayarak, ön işlemeyi geçip, uygun modeli seçmek, modeli eğitmek ve sonuçları değerlendirmek bunlardan bir kısmıdır. Bu süreç, her adımın titizlikle yönetilmesini ve optimizasyonunu gerektirir. İşte burada, projelerin yönetimini kolaylaştıran ve sistematik bir yaklaşım sunan şablonların önemi devreye girmektedir [1].

Bir makine öğrenmesi proje şablonu, proje aşamalarını planlamak ve uygulamak için bir yol haritası görevi görür. Şablonlar kullanarak, projenin her bir adımını daha iyi anlayabilir ve bu adımları daha verimli bir şekilde uygulayarak iş akışınızı düzenleyebilirsiniz. Şablonlar, projenin gidişatını izlemenin yanı sıra, her bir adım arasındaki ilişkileri netleştirir ve projenin hangi aşamada olduğunu, bir sonraki adıma ne zaman ve nasıl geçileceğini belirlemenize yardımcı olur. Ayrıca şablonlar, projenin her adımında ne yapılması gerektiğini hatırlatır, böylece kritik aşamaların gözden kaçırılması riskini azaltır[2].

Şablon kullanmanın avantajları sadece düzen ve netlikle sınırlı değildir; projenin tekrarlanabilirliğini ve ölçeklenebilirliğini artırmak gibi önemli yararlar da sağlar. Bir şablon, kullanılacak yöntemleri, araçları ve teknikleri açıkça belirtir, bu da benzer projelerde zaman kazanımını ve daha hızlı ilerlemeyi sağlar. Şablonlar, aynı zamanda projelerin birden fazla kişi tarafından yürütülmesinde iş birliğini ve koordinasyonu kolaylaştırır[3].

Sonuç olarak, şablonlar makine öğrenmesi projelerinin yönetiminde büyük değer taşır. Projeyi daha sistematik bir biçimde yönetmeyi, işleri tekrarlanabilir kılmayı, ölçeklendirme imkânlarını genişletmeyi ve projeyi özelleştirme ihtiyaçlarına göre uyarlamayı mümkün kılarlar. Bu nedenle, bir makine öğrenmesi projesine başlarken, doğru bir şablon kullanmak veya kişisel gereksinimlerinize uygun bir şablon oluşturmak, projenin başarısını büyük ölçüde artırma potansiyeline sahiptir. Bu şablonun hangi alt görevlerden oluştuğu ve nasıl işlediği adım adım sunularak projelere nasıl sağlam bir temelle başlanacağını gösterilecektir.

1.1 Bir Şablon Üreterek Başlama

Makine öğreniminin uygulamalı bir disiplin olduğu, deneyim ve pratiğin bu alandaki gelişim için kritik önem taşıdığı, kitap ve makalelerin bilgiyi arttırmasına rağmen, gerçek dünya becerilerinin kazanılması için aktif uygulama gerektiği genel kabul gören bir gerçektir. Bu bağlamda, öğrencilerin ve uygulayıcıların Python programlama dili kullanılarak makine öğrenimi projelerine hızlı ve etkin bir giriş yapabilmeleri için tasarlanmış 6 aşamalı bir proje şablonunun incelenmesi bu kitabın temel odağını oluşturacaktır. İlgili bölümlerde katılımcılar, aşağıdaki temel bilgi ve uygulama tekniklerini edineceklerdir:

1. Makine öğrenmesi projelerinin nasıl yapılandırılacağı: Okuyucu, başlangıçtan itibaren, projenin modelleme sürecinin nasıl adım adım planlanması gerektiğini anlayacak ve bu bilgileri kendi projelerinde uygulayabilecektir. Projelerini tahmin modelleme projeleri olarak yapılandırmak için gerekli adımları ve yönergeleri öğrenecektir.

2. Görevlerin projede nasıl entegre edileceği: Veri toplama, ön işleme, model seçimi gibi bir dizi sürecin akıcı ve mantıklı bir sıra içinde nasıl yürütüleceğini ve bu görevlerin projenin genel başarısına nasıl etki edeceğini kavrayacaktır. Bu sayede, görevleri birbirleriyle uyumlu bir şekilde entegre ederek projelerinin verimliliğini artırabileceklerdir.

3. Yapılandırılmış proje şablonunu veri kümesi için nasıl en iyi şekilde kullanacaklarını: Kullanılan veri setinin özelliklerine ve ihtiyaçlarına göre proje şablonunu nasıl özelleştireceğini ve sonuçların güvenilirliğini artırmak için hangi stratejileri benimseyeceğini anlayacaktır. Bu şekilde, veri setlerine özgü gereksinimleri dikkate alarak proje şablonunu optimize edebilecek ve elde ettikleri sonuçların kalitesini iyileştirebilecektir.

Bu bölümde, makine öğrenimi projeleri üzerinde çalışırken kullanılabilecek metodolojik bir çerçeve sunulmuştur. Teorik bilginin pratik uygulamalarla pekiştirilmesi ve katılımcıların gerçek dünya problemlerine çözüm üretebilme yetkinliklerinin geliştirilmesi amaçlanmaktadır.

1.2 Projelerle Makine Öğrenimi

Makine öğrenimi projelerinin uygulanma sürecinde ortaya çıkabilecek sorunlarla mücadele, bu sorunların üstesinden gelme ve sonuca ulaşabilme yeteneği, bilim dalının özünde kritik bir rol oynamaktadır. Elbette, makine öğrenimi teorisi üzerine eğitim almak ve basit, tek kullanımlık algoritma reçetelerini test etmek mümkündür. Ancak, uygulamalı makine öğreniminin gerçek potansiyelini açığa çıkarmak için, bir veri setini başından sonuna kadar analiz edip işlemek gereklidir.

Bir projenin üzerinde çalışmak, sadece tercih edilen bölümlere odak-

lanmak ve başarılı modeller geliştirmekle sınırlı kalmamalıdır; aynı zamanda modelin kullanım amacı ve işleyişine dair derinlemesine düşünmeyi, varsayımların eleştirel bir biçimde sorgulanmasını ve projenin her bir bileşeninde yetkinlik kazanmayı gerektirir.

Tahminsel modelleme alanında makine öğrenimi projelerine yönelik becerilerin geliştirilmesinde açık kaynaklardan (UCI Makine Öğrenimi Reposu'ndan) elde edilen standart veri kümelerinin kullanımı önerilmektedir. Bu yaklaşım, uygulamalı eğitim sürecinde somut ve ölçülebilir gelişmelerin elde edilmesine olanak tanımaktadır.

1.2.1 Adım Adım Yapılandırılmış Bir Akış Süreci İzleyin

Denetimli bir makine öğrenmesi genel akış şeması Şekil 1.1'de paylaşılmıştır. Herhangi bir makine öğrenimi tahminleme proje süreci aşağıda sunulan 6 adımlık alt görevlerin yerine getirilmesiyle mümkün olabilir. Bu alt görevler şu şekilde sıralanabilir:

1. Problemin Tanımlanması: Proje kapsamında çözülmesi gereken problemin ve beklenen sonucun tanımlanması.

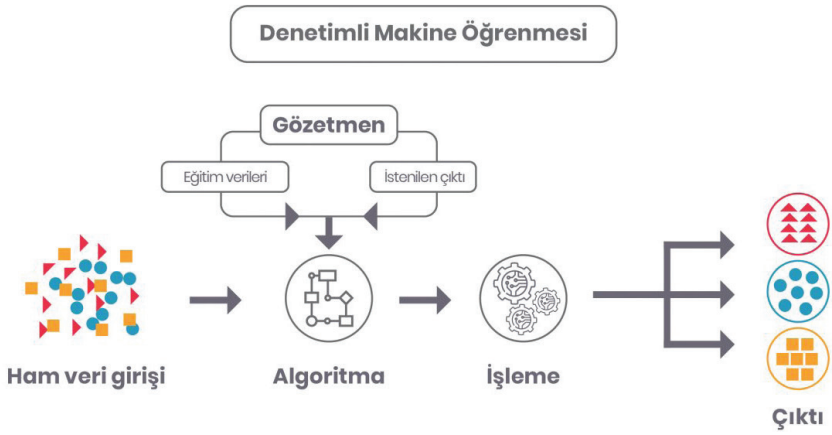
2. Veri Özetleme: Veri setinin ana hatlarıyla incelenmesi ve çıktıyla olan ilgili girdilerin istatistiksel yöntemler ve gözlemleme ile anlaşılmaya çalışılması.

3. Veri Hazırlama: Verinin modelleme aşamasına uygun hale getirilmesi, ön işleme ve temizleme işlemlerinin gerçekleştirilmesi.

4. Algoritmaları Değerlendirme: Farklı makine öğrenimi algoritmalarının performanslarının değerlendirilmesi ve en başarılı olanının seçilmesi.

5. Sonuçları İyileştirme: Seçilen modelin performansını artıracak parametre ayarlamalarının yapılması ve iyileştirmelerin uygulanması.

6. Sonuçları Sunma: Modelleme sonuçlarının ilgili paydaşlara anlaşılır bir biçimde sunulması.



Şekil 1.1 *Denetimli makine öğrenmesi genel akış şeması[4]*

Bu görevlerin her biri birleştirilebilir veya daha detaylı alt görevlere ayrılabilir; ancak, bu altı maddelik liste makine öğrenimi modelleme sürecinin temel yapısını oluşturur [5]. Python programlama dilinde bir tahminleme modeli geliştirerek sorun çözme süreci bu görevlerin dikkate alınmasıyla şekillenir. Python'daki ilgili görevler, uygun kütüphaneleri kullanarak adım adım yerine getirilmelidir; örneğin, veri yükleme için Pandas ve modelleme için scikit-learn gibi kütüphaneler tercih edilebilir. İlerleyen bölümlerde bu aşamaların her biri, yararlanabileceğiniz alt görevler ve kullanılacak kütüphanelerle daha ayrıntılı bir şekilde ele alınacaktır.

1.3 Python'da Makine Öğrenimi Proje Şablonu

Bu bölümde, Python programlama dilinde makine öğrenimi problemlerini ele alırken kapsamlı bir şekilde kullanabileceğiniz, başlangıçtan sonuca dek rehberlik edecek bir proje şablonu sunulmaktadır.

1.3.1 Şablon Özeti ve Kullanımı

Python'da makine öğrenimi projelerinizde aşağıdaki proje şablonunu kullanabilirsiniz.

Makine Öğrenimi Proje Şablonunun Aşamaları
1. Problemi Ayrıntılandırarak Hazırla
   a) Kütüphaneleri Yükleme
   b) Veri kümesini yükleme

2. Veriyi Özetle
   a) Tanımlayıcı istatistikleri hesapla
   b) Veriyi grafiklerle görselleştir
3. Veriyi Hazırla
   a) Veriyi Temizleme
   b) Özellikleri Seç
   c) Veri Dönüşümlerini Gerçekleştir

4. Algoritmaları Değerlendir
   a) Doğrulama veri kümesini ayır
   b) Test seçenekleri ve değerlendirme metriği
   c) Algoritmaları kontrol et
   d) Algoritmaları karşılaştır

5. Doğruluğu İyileştir
   a) Algoritma Ayarlama
   b) Ensemble'lar

6. Modeli Sonlandır
   a) Doğrulama veri kümesi üzerinde tahminler
   b) Tüm eğitim veri kümesi üzerinde bağımsız model oluştur
   c) Modeli daha sonra kullanmak için kaydet

Tablo 1.1 *Makine Öğrenmesi Proje Şablonu*

Proje şablonu uygulaması, aşağıda özetlenen sıralı ve basit adımlarla yürütülür:

1. Yeni bir proje dosyası oluşturarak işe başlayın (örnek: proje_aciklayici_adi.py).

2. Önceden hazırlanmış olan proje şablonunun içeriğini kopyalayın.

3. Bu şablonu, yeni oluşturduğunuz boş proje dosyasına yapıştırın.

Bu kitapta ve diğer kaynaklarda bulunan yöntem ve reçeteleri kullanarak, şablonu doldurmaya başlayın ve projenizi şekillendirin [6].

Sırada şablonun adımları hakkında daha fazla ayrıntı belirlemeye geldi.

1.3.3 Problemi Detaylandırarak Hazırlama

Bu aşama, projenin temel çalışmasını başlatmak için gerekli olan tüm öğelerin yüklenmesini içerir. Bu öğeler şunlardır:

1. Kullanmayı düşündüğünüz Python kütüphanelerini içe aktarın: Sınıflar, fonksiyonları ve projenizin gereksinim duyduğu tüm yazılım bileşenlerini içe aktarın (import).

2. Veri setinizi CSV dosyasından yükleyin: Analiz edeceğiniz verileri içeren CSV dosyasını sisteminize yükleyin ve uygun bir şekilde okuyun.

Aynı zamanda, projeniz için zorunlu olan herhangi bir global ayarı bu aşamada gerçekleştirmeniz gerekebilir (belirli bir Python sürümünü tercih etmek gibi).

Eğer veri setiniz çok büyükse ve işlenmesi pratik olmayacak kadar zaman alıyorsa, bu aşamada veri setinin boyutunu azaltmanız önerilir. İdeal olarak, veri setinizin boyutu bir modeli bir dakika içinde eğitebilecek veya bir görselleştirme üretebilecek kadar küçük olmalıdır; bir modelin eğitim süresinin ideal olarak yaklaşık 30 saniye olması hedeflenebilir. Performansı tatmin edici olan modelleriniz için ilerleyen aşamalarda, veri setinin boyutunu artırmayı düşünebilirsiniz. Bu, modelin ölçeklenebilirliğini ve genelleştirme kapasitesini doğrulamak için etkili bir stratejidir.

1.3.4 Veriyi Özetleme

Bu adım, elimizdeki veri setinin daha derinlemesine incelenmesi ve anlaşılmasına olanak tanıyan iki temel süreçten oluşur:

1. Tanımlayıcı İstatistikler: Veri setiniz hakkında özet bilgiler sunan istatistiksel ölçütleri hesaplayın. Bu, verinin merkezi eğilimi, yayılımı ve dağılımı gibi temel niteliklerini kapsar.

2. Veri Görselleştirme: Matplotlib gibi grafik kütüphaneleri kullanarak verilerinizi görselleştirin. Veri seti üzerinde özet bilgilere hızlı erişim sağlayan ve veri görselleştirme işlemlerini kolaylaştıran Pandas kütüphanesinin sağladığı fonksiyonlardan faydalanmak bu süreçte en uygun yaklaşım olacaktır.

Bu bölüme yeterince zaman ayırarak, elde ettiğiniz sonuçları dikkatle inceleyin. Bu sonuçlar, ilerleyen aşamalarda üzerinde çalışabileceğiniz bir dizi soru, varsayım ve hipotez oluşturmanıza yardımcı olacaktır. Bu, özel modellerinizi geliştirme aşamasına geçmeden önce veri setiniz hakkında derin bir kavrayışa sahip olmanızı sağlayacak kritik bir ön hazırlık adımıdır.

1.3.3 Veriyi Hazırlama

Bu aşama, veri setinin, problemin doğasını ve öznitelikler ile çıktı değişkeni arasındaki ilişkiyi en etkili biçimde yansıtacak şekilde düzenlenmesini içerir. İlgili süreç şu adımlardan meydana gelir:

1. Adım Veri Toplama: İlk adım, kullanılacak veriyi toplamak ve kaydetmektir.

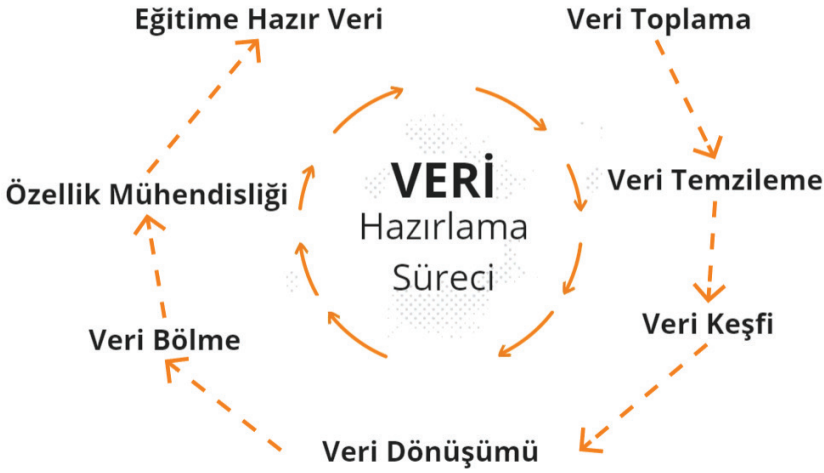
2. Adım Veri Temizleme: Bu adımda, eksik veya hatalı verileri tespit edip düzeltme işlemleri yapılır.

3. Adım Veri Keşfi: Veri setinin genel yapısını anlamak için istatistiksel analizler ve görselleştirmeler yapılır.

4. Adım Veri Dönüşümü: Gerektiğinde, veriyi makine öğrenimi algoritmalarına uygun formata dönüştürme işlemleri gerçekleştirilir.

5. Adım Veri Bölme: Veri seti eğitim ve test setlerine ayrılır.

6. Adım Özellik Mühendisliği: Var olan özellikler üzerinde çalışma veya yeni özellikler oluşturma adımlarıdır.



Şekil 1.2 *Veri temizleme*

İlk olarak basit bir yaklaşım benimseyin ve modelleme sürecinizde sürekli olarak bu aşamaya geri dönün. Doğru veya yeterli doğrulukta modeller üretebileceğiniz, veri sunumunun ve algoritmaların uygun bir kombinasyonunu bulana kadar bu süreçteki adımları iteratif olarak tekrarlayın. Bu iteratif döngü, algoritma ayarlarının optimize edilmesi, modelin eğitimi ve elde edilen sonuçların değerlendirilmesi gibi aşamaları içermelidir. Bu yöntemle, modelin performansını sürekli olarak geliştirecek ve daha iyi sonuçlara ulaşacaksınız [7].

1.3.4 Algoritmaları Değerlendirme

Bu adımda hedeflenen, veri setinin yapısına ve problemin türüne uygun bir algoritmanın tercih edilmesinin sağlanmasıdır. Bu kapsamda birden fazla algoritma yapısal olarak problem uyumu değerlendirilebilir ve eğitilerek performans değerleri incelenebilir. Süreç, aşağıdaki aşamaları kapsamaktadır:

1. Doğrulama Veri Kümesini Belirleme: Modelinizin performansını değerlendirip doğrulamak üzere bir bölüm veriyi ayrı bir doğrulama seti olarak ayırın.

2. Parametre Ayarlama: Scikit-learn kullanarak çeşitli algoritmaların parametrelerinin farklı kombinasyonlarını test ederek en iyi sonucu sağlayan kombinasyonları belirleyin.

3. Lineer ve Non-lineer Algoritmaları Deneme: Hem lineer hem de non-lineer makine öğrenimi algoritmalarının bir dizi testini gerçekleştirmek.

4. Algoritma Performanslarını Karşılaştırma: Farklı algoritmaların tahmini doğruluk oranlarını birbiriyle karşılaştırarak en yüksek performansı sergileyenleri belirlemek.

Bu ve önceki adım üzerinde geçireceğiniz zamanın büyük bir kısmı, belirli bir problem üzerinde en iyi performansı gösteren 3 ila 5 makine öğrenimi algoritmasını saptamak için kullanılacaktır. Bu, modellerinizi geliştirme ve rafine etme sürecindeki kritik bir evredir ve etkili bir model seçimi için dikkatli ve metodolojik bir yaklaşım gerektirir.

1.3.5 Doğruluğu (Accuracy) Yükseltme

Bu aşamada, yüksek düzeyde tahmin performansı sergileyen (doğruluk değeri başarılı kabuşl edilebilir seviyede yüksek) makine öğrenimi algoritmalarının bir alt grubunu belirlemek amaçlanmaktadır. İşlem aşağıdaki adımları içermektedir:

1. En İyi Parametrelerin Belirlenmesi: Scikit-learn kütüphanesi kullanılarak çeşitli algoritmalar için en etkili parametre kombinasyonlarını araştırmak ve bunları tespit edilmelidir

2. Ensemble Yöntemleri Kullanımı: Birden fazla modelin tahminlerini bir araya getirerek daha güçlü bir tahmin modeli oluşturmak için ensemble yöntemleri uygulanmalıdır.

Bu ve önceki adım, projenin ilerleyen safhalarında birbirine geçişli hale gelebilir ve bu nedenle aralarındaki sınır bulanıklaşabilir. Önceki aşamada belirli ölçüde algoritma parametre ayarlamaları yapılabilirken, bu aşamada ensemble modelleri oluşturmak için daha geniş bir algoritma

listesi ele alınabilir. Bu süreç, projenin özelliklerine ve gereksinimlerine göre dinamik bir yaklaşım gerektirir ve her iki aşamada da yapılan çalışmaların birbirini tamamlaması beklenir. Bu tamamlayıcı sürecin amacı, veri setine ve problem tanımına en uygun, doğru tahminler yapabilen ve genelleştirme gücü yüksek modeller geliştirmektir. Ensemble ile modellerin birleştirilmesi genel olarak performansı arttırıcı etki gösterdiğinden ensemble model test etme önemli bir basamaktır[8].

1.3.6 Modeli Sonlandır

Bir modelin eğitim süreci tamamlandığında ve görünmeyen veriler üzerinde güvenilir tahminlerde bulunabileceği kanısına varıldığında, modeli sonlandırma sürecine geçilir. Bu süreç, şu alt görevleri kapsayabilir:

1. Optimum Parametrelerle Tahmin Yapma: Scikit-learn tarafından belirlenmiş en iyi parametrelerle donatılmış modeli kullanarak, henüz karşılaşmadığı veri üzerinde tahminlerde bulunmak.

2. Bağımsız Model Oluşturma: Scikit-learn ile ayarlanan optimal parametreler kullanılarak, bağımsız bir modelin kurulması ve ayarlanması.

3. Modeli Kaydetme: Optimize edilmiş modeli, ileride kullanmak üzere bir dosya formatında saklamak.

Bu aşamaya ulaştığınızda, modelinizi ilgili paydaşlara (stakeholderlara) sunmaya veya modeli gerçek dünya verileri üzerinde tahminler yapmak üzere aktif olarak kullanmaya başlamaya hazır durumda olduğunuz anlamına gelir. Bu süreç, modelin uygulanabilirliğini ve değerini kanıtlayacak ve proje çıktılarını ilgili taraflarla paylaşmanıza olanak tanıyacaktır.

1.4 Şablonu İyi Kullanma İpuçları

Python'da bir makine öğrenimi proje şablonunu etkin bir şekilde kullanmanız için faydalı olabilecek bazı stratejiler aşağıdaki gibi sıralanabilir.

1. Hızlı Başlangıç: Proje adımlarını mümkün olan en kısa sürede tamamlayın. Bu hızlı ilerleme, projenin tüm temel unsurlarını yerine getirdiğiniz ve geliştirmeye açık bir temele sahip olduğunuz konusundaki özgüveninizi artıracaktır.

2. İteratif Süreç: Proje süreci doğrusal değil, iteratif bir yapıdadır. Farklı adımlar arasında geçiş yapacaksınız ve muhtemelen 3., 4. ve 5. adımları içeren iterasyonlarda en çok zamanınızı harcayacaksınız; yeterli bir doğruluk seviyesine ulaşana veya zamanınız dolduğunda bu döngülerde kalacaksınız.

3. Her Adımı Gerçekleştirme: Özellikle o adımdaki görevlere hâkim olmadığınız veya aşina olmadığınız durumlarda adımları atlamak cazip

gelebilir. Bu eylem doğrudan doğruluk (accuracy) üzerinde belirgin bir etki yaratmasa her adımda bir eylem gerçekleştirmek önemlidir. Deneyler yaparak farklı yaklaşımlar geliştirebilirsiniz. Adımları atlamayın, ancak gerektiğinde odaklarını düşürün.

4. Model Doğruluğuna Odaklanma: Projelerin temel amacı yüksek model doğruluğuna ulaşmaktır. Gerçekleştirdiğiniz her adım bu hedefe hizmet etmelidir. Yapılan değişikliklerin, model doğruluğunu en etkili şekilde artırma yönünde olmasını hedefleyin. Model doğruluğunu bir pusula gibi görün ve diğer adımları bu pusulaya göre ayarlayın. Unutmayın ki doğruluk, her iki yönde de olumlu veya olumsuz hareket edebilen bir metriktir.

5. Esnek Olma: İlerleyen zamanlarda ve şablona dair deneyiminiz arttıkça, adımları projenizin gereksinimlerine göre esnek bir şekilde uyarlayın. Özellikle algoritma değerlendirme ve iyileştirme adımları arasındaki sınırları daha esnek hale getirin.

Scikit-learn gibi araçlar bol miktarda parametre ayarlaması sunar ve bu parametrelerin kombinasyonlarını en iyi sonuçları elde etmek için sürekli olarak test etmek gerekebilir. Bu nedenle, model doğruluğunu optimize etmek amacıyla süreci esnek bir biçimde adaptasyon ve ayarlama beklenir.

1.5 Özet

Bu derste, Python'da bir makine öğrenimi proje şablonu keşfettiniz. Model doğruluğunu maksimize etme amacıyla tahmin modelleme makine öğrenimi projesi adımlarını düzenledi. Şablonu kopyalayıp yapıştırarak, mevcut veya gelecekteki Python makine öğrenimi projenizi hızlandırmak için kullanabilirsiniz.

1.6 Sonraki Adım

Artık Python'da tahmin modelleme makine öğrenimi projesini nasıl yapılandırılacağını biliyorsunuz, bu bilgiyi kullanmaya ihtiyacınız olacaktır. Sonraki derste, basit bir vaka çalışması problemini baştan sona çalışacaksınız. Bu ünlü bir vaka çalışmasıdır ve makine öğrenimi projelerinin merhaba dünyasıdır.

2. BÖLÜM

Giriş

Makine öğrenimi tabanlı tahmin modellemesi projeleri, veri toplama, veri ön işleme, öznitelik mühendisliği, model seçimi, eğitim, değerlendirme ve sonuçların raporlanması gibi, bir dizi karmaşık alt bileşenin entegrasyonuyla oluşur.

İlk evre, veri akışının başlatıldığı ve ilgili veri kaynaklarından gerekli bilgilerin elde edildiği veri toplama sürecidir. Takip eden ön işleme evresi, elde edilen verinin temizlenmesi, düzenlenmesi ve işlenebilir özelliklerin çıkarılması işlemlerini içerir. Öznitelik mühendisliği, verinin modellenmesi için gerekli olan ve veri setinin yapısal karakteristiklerini en iyi yansıtan özniteliklerin belirlenmesi ve bu özniteliklerin model tarafından efektif bir şekilde işlenmesini sağlamak için uygulanan bir dizi işlemi kapsar[9].

Model seçimi aşamasında, projenin ihtiyaçları ve verinin doğası dikkate alınarak uygun makine öğrenimi modeli veya modelleri seçilir. Bu, analitik yetenek ve uzmanlık gerektiren bir süreçtir. Model eğitimi süreci, belirlenen modelin veya modellerin, veri seti üzerinden eğitilerek öğrenme süreçlerinin tamamlanması işlemini içerir. Modelin performansının değerlendirilmesi ise eğitim sonrası yapılan analizlerle, modelin tahmin yeteneğinin ölçülmesine yöneliktir[10].

Son aşama, modelin sunumudur. Bu süreçte, modelden elde edilen sonuçlar, projenin hedeflediği amaçlar ve belirlenen hedef kitle gözetilerek raporlanır ve ilgili taraflarla paylaşılır. Bu bileşenlerin bir araya gelmesiyle, tahmin modellemesi makine öğrenimi projesi tamamlanmış olur. İşbu ders kapsamında, siz değerli öğrenciler Python programlama dilini kullanarak ilk makine öğrenimi projenizi oluşturacak ve hayata geçireceksiniz[11].

Adım adım yürütülecek eğitim süreci boyunca aşağıdaki faaliyetler gerçekleştirilecektir:

1. Python'un bilimsel hesaplama kütüphanelerinden olan SciPy ekosistemini indirme ve bu ekosistem içerisinde yer alan makine öğrenimi için kritik öneme sahip paketlere erişim sağlama.

2. Belirli bir veri setini yükleme ve bu veri setinin yapısını istatistiksel özetler ve görselleştirme teknikleri kullanarak analiz etme.

3. Birden fazla makine öğrenimi modeli geliştirme, bu modeller arasından en performanslı olanı seçme ve tahmin doğruluğunun güvenilirliğini teyit etme.

Makine öğrenimi metodolojilerine yeni başlayan ve Python programlama dilinde yetkinlik kazanmayı amaçlayanlar için bu eğitim modülü özel olarak düzenlenmiştir. Şimdi, bu heyecan verici eğitim sürecine birlikte adım atalım.

2.1 Makine Öğrenimine Giriş: İris Çiçeği Sınıflandırma

Bir programlama dilinde becerilerinizi ilk kez sergilemek amacıyla ideal bir başlangıç projesi, genellikle İris çiçeği veri setinin sınıflandırılması olarak kabul edilir. Bu veri seti, ilk deneyler için mükemmel bir seçimdir, zira üzerinde geniş çapta çalışılmış ve temel özellikleri iyi anlaşılmıştır.

Özelliklerin niceliksel olması, verinin nasıl yükleneceği ve işleneceği konusunda temel becerilerin kazanılmasını zorunlu kılar. Sınıflandırma problemi kategorisinde yer aldığı için, gözetimli öğrenme algoritmalarıyla deneyim kazanmak için daha basit bir çıkış noktası sunar.

Problem, çoklu sınıflandırma niteliğindedir (multinomial), bu da özel işlem tekniklerinin gerekebileceğini işaret eder.

Sadece dört özelliğe ve 150 gözleme sahip olması, veri setinin küçük ve yönetilebilir olmasını sağlar; bu, verinin hafızada (ve bir ekran veya tek bir sayfa üzerinde) kolayca işlenebilmesini mümkün kılar.

Tüm sayısal özellikler aynı birimlerde ve ölçekte olduğundan, başlangıç için özel bir ölçekleme ya da dönüşüm işlemine gerek yoktur.

Bu modülde, küçük ölçekli bir makine öğrenimi projesini başından sonuna kadar yürüteceğiz. İzleyeceğimiz adımlar şunlardır:

1. Veri kümesinin yüklenmesi,
2. Veri kümesinin özetlenmesi ve temel istatistiklerin çıkarılması,
3. Veri kümesinin görselleştirilmesi,
4. Seçilen algoritmaların değerlendirilmesi,
5. Tahminlerin yapılması.

Her bir adımı özenle uygulamayı hedefleyin ya da hız kazanmak için, yazılı komutları kopyala-yapıştır yöntemiyle deneyin. Python etkileşimli ortamınızı hazırlayın ve ilk makine öğrenimi projenize, bu klasik "Merhaba Dünya" örneğiyle başlayarak, algoritmik düşünme yeteneklerinizi test edin.

2.2 Veriyi Yükleme

Bu evrede, belirtilen internet kaynağından gerekli kütüphaneleri ve Iris türlerine ait verileri içeren CSV dosyasını indireceğiz.

2.2.1 Kütüphaneleri Yükleme

Projenin ilerleyen aşamalarında verinin işlenmesi, analitik özeti ve görselleştirilmesi gibi işlevleri yerine getirebilmesi için ilk etapta temel öneme sahip olan kütüphanelerin entegrasyonunu gerçekleştirmemiz gerekmektedir. Bu kütüphaneler, NumPy, Pandas ve Matplotlib'tir. Bu kütüphaneler, **veri yükleme**, **özetleme** ve **görselleştirme** işlemleri için gereklidir. Aşağıdaki kod bloğu ile projelerde bu kütüphaneler yüklenebilir:

```
# Liste 2.1 Kütüphaneleri Yükleme
from pandas import read_csv
from pandas.tools.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from pandas.plotting import scatter_matrix
```

Liste 2.1 *Kütüphaneleri Yükleme*

2.2.2 Veri Setini Yükleme

Kullanacağımız açık kaynak İris veri setini UCI Makine Öğrenimi deposundan indirebilirsiniz. Veriyi yüklemek için Pandas'ı kullanıyoruz. Ayrıca, veriyi tanımlayıcı istatistikler ve veri görselleştirme yöntemleriyle keşfetmek için de Pandas'ı kullanacağız. Veriyi yüklerken sütunların adlarını belirtmeyi unutmayın; bu veriyi keşfetmekte daha sonra yardımcı olacaktır.

```
# Veri Seti Yükleme
filename = 'iris.data.csv'
oznitelik_isimleri = ['canak-uzunluk', 'canak-genislik',
'tac-uzunluk', 'tac-genislik', 'sinif']
dataset = read_csv(filename, names=names)
```

Liste 2.2 *İris veri setini CSV dosyasından yükleme*

2.3 Veri Kümesini Özetleme

Data setinin incelenmesi süreci, verilerin birden çok boyutta analiz edilmesini gerektirmektedir. Bu kapsamlı inceleme, aşağıdaki adımları içerecektir:

1. Data setinin dimensionalitesinin ve hacminin tespiti, veri setinin kapsadığı gözlem ve öznitelik sayısının belirlenmesi ile gerçekleştirilecektir.

2. İlgili veri seti içerisindeki değerlerin tespiti, verinin genel bir yapı ve içerik çözümlemesi yapılmasını sağlayacaktır.

3. Veri setinin çeşitli özelliklerinin istatistiksel özetlerine ulaşılması, bu özetlerin merkezi eğilim, değişkenlik ve dağılım gibi temel istatistiksel ölçütleri içermesi beklenmektedir.

4. Sınıf değişkeninin dağılım analizi, çeşitli sınıf kategorilerinin veri seti içerisindeki frekans ve oranlarının anlaşılmasına olanak tanıyacaktır.

Bu çözümleme aşamaları için kullanılacak olan komutlar, veri biliminde sıklıkla başvurulan temel komutlardır ve her bir incelenen boyut genellikle tek bir komut satırı ile ifade edilebilir. Dolayısıyla, bu komutlar gelecekteki projelere uygulanabilir ve tekrar edilebilir nitelikte olup, veri analizinin temel araçlarındandır. Bu sebeple, bu inceleme süreci boyunca duyulan endişe için bir gereklilik bulunmamaktadır.

2.3.1 Veri setinin Boyutları

Veri setinin içerdiği örnek (satır) ve özellik (sütun) sayısının tespiti, basit ve hızlı bir işlemdir. Veri setinin boyutlarını, "shape" komutu kullanılarak elde etmek mümkündür (Liste.2.3).

```
#shape verisetinin satır ve sütun değerlerini döndürür.
print(veriseti.shape)
```

Liste 2.3 *İris verisetini CSV dosyasından yükleme*

Iris veri seti üzerinde gerçekleştirilen *"shape"* komutu analizi neticesinde, toplam 150 adet gözlem veya örnek ile 5 adet öznitelik veya değişken bulunduğu tespit edilmiştir.

```
(150, 5)
```

Liste 2.4 *İris veriseti boyut sonucu*

2.3.2 Veriyi İnceleyin

Veri setlerinin detaylı bir şekilde gözden geçirilmesi, veri analizi sürecinde izlenmesi önerilen önemli bir yaklaşımdır.

```
# veriseti.head(adet)  ilk "adet" kadar veriyi gösterir.
print(veriseti.head(20))
```

Liste 2.5 *Iris veriseti boyut sonucu*

Veri setinin tüm sütunlarına ve ilk yirmi satırına dair bir gösterim Liste 2.6'da sunulmuştur.

```
    canak-uzunluk  canak-genislik  tac-uzunluk  tac-genislik        sinif
0             5.1             3.5          1.4           0.2  Iris-setosa
1             4.9             3.0          1.4           0.2  Iris-setosa
2             4.7             3.2          1.3           0.2  Iris-setosa
3             4.6             3.1          1.5           0.2  Iris-setosa
4             5.0             3.6          1.4           0.2  Iris-setosa
5             5.4             3.9          1.7           0.4  Iris-setosa
6             4.6             3.4          1.4           0.3  Iris-setosa
7             5.0             3.4          1.5           0.2  Iris-setosa
8             4.4             2.9          1.4           0.2  Iris-setosa
9             4.9             3.1          1.5           0.1  Iris-setosa
10            5.4             3.7          1.5           0.2  Iris-setosa
11            4.8             3.4          1.6           0.2  Iris-setosa
12            4.8             3.0          1.4           0.1  Iris-setosa
13            4.3             3.0          1.1           0.1  Iris-setosa
14            5.8             4.0          1.2           0.2  Iris-setosa
15            5.7             4.4          1.5           0.4  Iris-setosa
16            5.4             3.9          1.3           0.4  Iris-setosa
17            5.1             3.5          1.4           0.3  Iris-setosa
18            5.7             3.8          1.7           0.3  Iris-setosa
19            5.1             3.8          1.5           0.3  Iris-setosa
```

Liste 2.6 *Veri setinin sütunları ve ilk 20 satırı*

2.3.3 İstatistiksel Özet

Her bir özniteliğin istatistiksel özetine ilişkin analiz şimdi gerçekleştirilebilir. Bu istatistiksel özet, gözlem sayısı, ortalamalar, minimum ve maksimum değerler, ile birlikte belirli yüzdelik dilimleri kapsamakta olup, Liste 2.7 ile kodlanıp Liste 2.8'de incelenebilmektedir.

```
# Verisetinin temel istatistiksel temel değerleri
print(veriseti.describe())
```

Liste 2.7 *Tanımlayıcı veri setinin tanımlayıcı istatistiklerinin hesaplanması*

```
       canak-uzunluk  canak-genislik  tac-uzunluk  tac-genislik
count     150.000000      150.000000   150.000000    150.000000
mean        5.843333        3.054000     3.758667      1.198667
std         0.828066        0.433594     1.764420      0.763161
min         4.300000        2.000000     1.000000      0.100000
25%         5.100000        2.800000     1.600000      0.300000
50%         5.800000        3.000000     4.350000      1.300000
75%         6.400000        3.300000     5.100000      1.800000
max         7.900000        4.400000     6.900000      2.500000
```

Liste 2.8 *Tanımlayıcı veri setinin tanımlayıcı istatistikleri gösterimi*

Sayısal değerlerin tamamının santimetre cinsinden eş ölçekte olduğunu ve değerlerin 0 ile 8 santimetre aralığında benzer dağılımlar sergilediğini gözlemlemekteyiz. Bu durum, veri setinin ölçüm değerlerinde bir standardizasyonun var olduğuna işaret etmekte olup, bu da sütun bazında herhangi bir ön işleme ihtiyaç duyulmadığını göstermektedir.

2.3.4 Sınıf Dağılımı

Her bir sınıfın içerisinde yer alan örneklerin toplam sayısını inceleyelim. Bu, sınıfların gözlem veya ölçüm sayılarına dair bir değerlendirmedir (Liste 2.9).

```
# Sınıflara göre veri(gözlem) sayıları
print(veriseti.groupby('sinif').size())
```

Liste 2.9 *Veri setindeki sınıflara göre gözlem sayılarının dağılımlarını yazdırma*

Veri setinde farklı sınıfların dağılımı, elde edilen veri setinin dengeli olup olmadığını belirlemede kritik bir gösterge olarak ön plana çıkar. Dengeli bir veri seti, her sınıfın benzer sayıda örnek içerdiği durumu ifade ederken, dengesiz veri setleri, bir veya birkaç sınıfın diğerlerine kıyasla aşırı temsil edildiği veya yetersiz temsil edildiği durumları tanımlamak için kullanılır. Dengesiz veri setleri, model eğitimi sırasında ve algoritma seçiminde özel dikkat ve hassasiyet gerektirir, zira bu durum, modelin genelleştirme yeteneği ve performansını olumsuz yönde etkileyebilecek faktörlerdendir (Liste 2.9).

```
# Tanımlamalar describe veri setini tanımlama
sinif
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

Liste 2.10 *Veri setindeki sınıf gözlem sayıları*

2.4 Veri Görselleştirme

Temel bilgiler veri seti hakkında elde edilmiştir. Ancak, veri anlatısını daha detaylı bir biçimde kavramak amacıyla bu bilgilerin grafiksel görselleştirmelerle zenginleştirilmesi gereklidir. Bu hedef doğrultusunda, iki ayrı görselleştirme yaklaşımı benimsenecektir:

1. Tek değişkenli grafikler: Bu grafikler, her bir özelliğin bağımsız olarak incelenmesine ve özelliklerin dağılımı, merkezi eğilim ve varyans gibi temel istatistiklerin görselleştirilmesine imkân tanıyarak daha derin bir anlam sağlayacaktır.

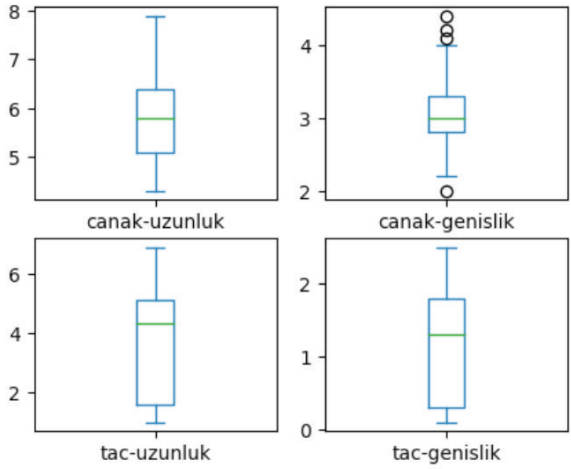
2. Çok değişkenli grafikler: Bu yaklaşım, özellikler arasındaki ilişkileri ve etkileşimleri keşfetmeye olanak sağlar. Çok değişkenli grafikler, veri setindeki öznitelikler arasındaki korelasyonları ve ilişkileri ortaya koyarak potansiyel kalıp ve trendleri açığa çıkaracaktır.

2.4.1 Tek Değişkenli Grafikler

Tek değişkenli analiz süreci, her bir değişkenin bağımsız grafiklerinin incelenmesiyle başlatılacaktır. Giriş değişkenlerinin sayısal nitelikler taşıması nedeniyle, her birinin kutu (box and whisker) grafikleri oluşturularak başlanabilir (Liste2.11). Bu grafikler, her özelliğin merkezi eğilimi, dağılımı ve aykırı değerlerinin görsel olarak sergilenmesi yoluyla, her bir değişkenin istatistiksel özelliklerine dair anlayışın derinleştirilmesine yardımcı olacaktır (Şekil 2.1). Bu sayede, her bir değişkenin istatistiksel özelliklerine dair anlayış derinleştirilecektir.

```
#Kutu (Box and Whisker) diyagramları çizdirme
veriseti.
plot(kind="box",subplots=True,layout=(2,2),sharex=False,
figsize=(5,4))
# Grafiği göster
pyplot.show()
```

Liste 2.11 *Kutu diyagram grafiği*



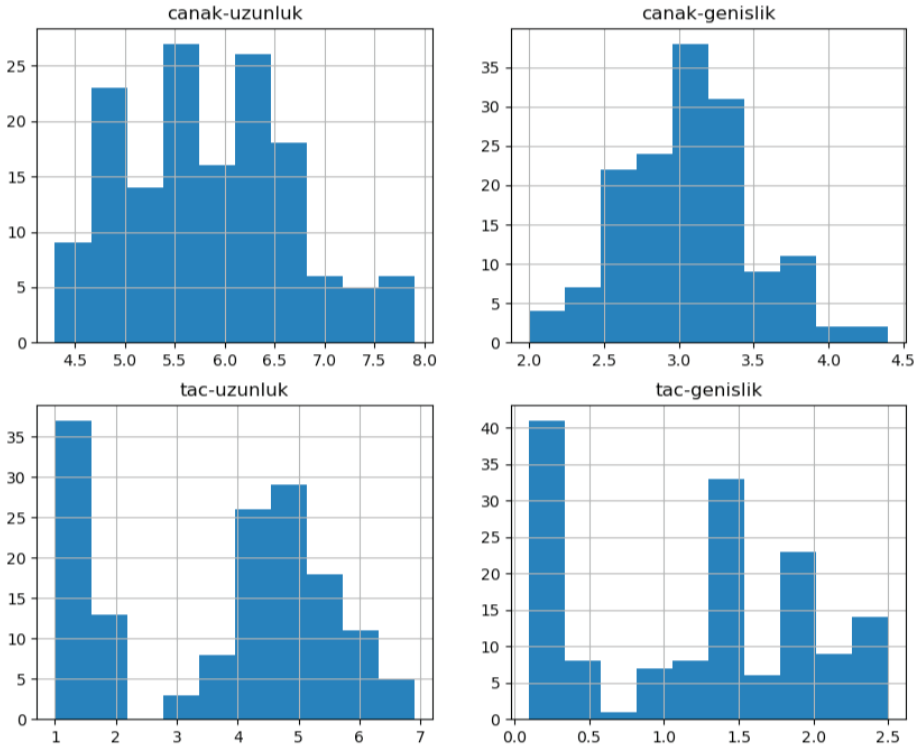
Şekil 2.1 *Her bir özniteliğin kutu grafiği*

Veri setinin dağılımını ve yapısal özelliklerini keşfetme amacıyla, histogramlar gibi grafiksel araçların kullanılması, veri analizi açısından oldukça faydalıdır. Histogramlar, veri noktalarını belirlenen sıklık aralıklarına bölerek, her bir aralıkta bulunan gözlem sayısını görsel bir biçimde sunulmasına olanak tanır. Bu sayede, veri setinin genel dağılım profili ve yoğunluk bölgeleri gibi önemli bilgilere erişmek mümkün olmaktadır (Liste 2.12). Şu anda, giriş değişkenlerinin her birinin histogramı çizilerek, bu özelliklerin dağılımı detaylı bir şekilde incelenmektedir (Şekil 2.2).

```
# Veriler histogram grafiği olarak çizdirilir
veriseti.hist(figsize=(10,8))
# Alt ve üst, sol ve sağ grafikler arasındaki boşluğu ayarla
pyplot.subplots_adjust(hspace=0.2)
# Sol ve sağ grafikler arasındaki boşluğu ayarla
pyplot.subplots_adjust(wspace=0.2)
# Grafiği göster
pyplot.show()
```

Liste 2.12 *Histogram dağılımı çizdirme*

Giriş değişkenlerinden olan çanak yaprak uzunluğu ve genişliğinin, örneklerin dağılımı bakımından Gauss dağılımı veya normal dağılım olarak bilinen karakteristik şekli sergilediği gözlemlenmiştir. Bu tespit, veri analizi ve model seçimi açısından önem taşımaktadır. Zira Gauss dağılımını varsaydığı ve bu dağılımı temel alarak daha verimli çalışan algoritmalar, model eğitiminde tercih edilebilir. Bu nedenle, bu tür dağılımlara uygunluk gösteren ve Gauss dağılımından faydalanarak başarılı modeller eğiten algoritmaların kullanılması, modelleme sürecinde potansiyel olarak daha etkili sonuçlar elde edilmesine olanak sağlayabilir.



Şekil 2.2 *Her bir özniteliğin histogram dağılımı*

2.4.2 Çoklu Öznitelik Grafikleri

Makine öğrenimi modellerinin performansını iyileştirmek için, girdi değişkenleri arasındaki ilişkileri ve etkileşimleri incelemek önemlidir. Bu, değişkenler arasındaki olası korelasyonları ve birliktelik kurallarını belirlemeye yardımcı olabilir.

İlişki Türleri

Öznitelikler arasındaki ilişkiler, iki ana kategoriye ayrılabilir:

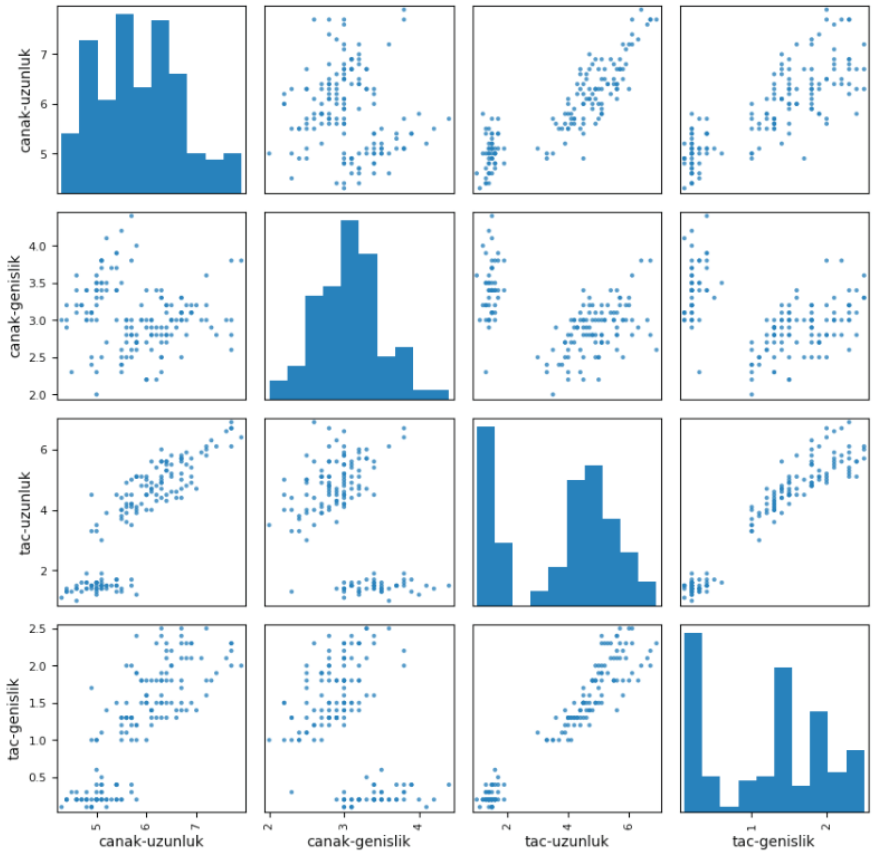
- Lineer ilişkiler: Bu ilişkilerde, bir değişkenin değeri, diğer değişkenin değeriyle doğrusal bir şekilde ilişkilidir. Örneğin, bir kişinin boyu ile kilosu arasındaki ilişki, doğrusal bir ilişkidir.

- Lineer Olmayan ilişkiler: Bu ilişkilerde, bir değişkenin değeri, diğer değişkenin değeriyle doğrusal olmayan bir şekilde ilişkilidir. Örneğin, bir kişinin IQ'su ile üniversite not ortalaması arasındaki ilişki, lineer olmayan (non-linear) bir ilişkidir.

Tüm özellik çiftlerinin dağılım grafiklerini çizdirerek karşılaştırma görsel olarak değerlendirilmelidir. Bu, girdi değişkenler arasındaki yapılandırılmış ilişkileri tespit etmeye yardımcı olabilir.

```
scatter_matrix(veriseti, figsize=(10,10), diagonal='hist',
alpha=0.5)
# Alt ve üst, sol ve sağ grafikler arasındaki boşluğu ayarla
pyplot.subplots_adjust(hspace=0.1)
# Sol ve sağ grafikler arasındaki boşluğu ayarla
pyplot.subplots_adjust(wspace=0.1)
# Grafiği göster
pyplot.show()
```

Liste 2.14 *Köşegen Histogram grafiği kodları*



Şekil 2.3 *İkili değişkenlerin histogram dağılım grafiği*

Bazı özellik çiftlerinin diyagonal gruplandırılmasına dikkat edilmelidir. Bu, yüksek bir korelasyon ve tahmin edilebilir bir ilişki önerir.

2.5 Bazı Algoritmaların Değerlendirilmesi

Veri ön incelemesi tamamlandıktan sonra, görülmemiş verilere göre eğitilen modellerin doğruluğunu tahmin etmek için modelleme aşamasına geçilir. Bu aşamada, aşağıdaki işlemler gerçekleştirilir:

1. Doğrulama veri kümesi ayrılır.
2. Test haznesi, 10-katlamalı çapraz doğrulama için ayarlanır.
3. Çiçek ölçümlerinden türleri tahmin etmek için 5 farklı model oluşturulur.
4. En başarılı model seçilir.

2.5.1 Bir Doğrulama Veri Kümesi Oluşturma

Bir yapay zekâ modeli eğitmek başarılı tahminler üreten bir zekâ geliştirmek demek değildir. Oluşturulan her model test edilerek performansı ortaya konmalı ve en başarılı en az hatalı tahmin yapan model en iyi model olarak seçilmelidir. Eğitilen modeller görülmemiş test verileri üzerindeki hata miktarları istatistiksel yöntemlerle hesaplanarak başarısı net şekilde ortaya konmalıdır. Son aşamada gerçek görülmemiş veriler üzerinde değerlendirme yapılmalı ve algoritmalara gösterilemeyecek bazı veriler kullanılarak en iyi modelin gerçekte ne kadar doğru olma ihtimalinin ikinci ve bağımsız bir fikrine ulaşılmalıdır.

Bu çalışmada yüklenen veri kümesini ikiye bölünecektir. Veri setinin %80'ini model eğitiminde kullanılırken ve %20'si doğrulama veri kümesi olarak ayrılacaktır (Liste 2.15).

```
# Verisetini test ve eğitim verisi olarak ikiye bölme
dizi = veriseti.values
X = dizi[:,0:4]
Y = dizi[:,4]
dogrulama_boyutu = 0.20
tohum = 7
X_egitim, X_test, Y_egitim, Y_test = train_test_split(X, Y, test_
size=0.33, random_state=tohum)
```

Liste 2.15 *Veri setinin eğitim ve test olarak ayrılması*

Şimdi *X_egitim* ve *Y_egitim* verilerinden model hazırlamak için eğitim verileri ve doğrulama test verileri ayrılmıştır.

2.5.2 Model Testi (Test Harnes)

Doğruluğu tahmin etmek için 10-katlamalı çapraz doğrulama (k=10 k-fold cross validation) kullanacağız. Bu, veri kümesini 10'a bölecek, 9'unu eğitecek ve 1'ini test edecektir. Bu adımlar eğitim-test bölme kombinasyonları için tekrar edecektir. Modelleri değerlendirirken genellikle doğruluk (accuracy) ölçütü kullanılır. Bu, veri kümesindeki toplam örnek sayısına bölünen doğru tahmin edilen örneklerin sayısının oranıdır ve yüzde olarak verilir (örneğin, %95 doğruluk). Bir sonraki adımda her modeli inşa etme ve değerlendirme sırasında puanlama (scoring) değişkeni kullanılacaktır.

2.5.3 Modeller Oluşturma

Bu problem için hangi algoritmaların iyi olacağı ve hangi yapılandırma ve ayarların kullanılacağı bilinmemektedir. Grafiklere bakılarak sınıfların bazılarının bazı boyutlarda kısmen doğrusal olarak ayrıştırılabilir olduğu öngörülmektedir. Bu nedenle, genellikle lineer algoritmaların iyi sonuçlar vereceği beklenmektedir. Şimdi, başarılı olabilecek farklı algoritmaları değerlendirelim:

1. Lojistik Regresyon (LR).
2. Lineer Diskriminant Analizi (LDA).
3. K-En Yakın Komşular (KNN).
4. Sınıflandırma ve Regresyon Ağaçları (CART).
5. Gaussian Naive Bayes (NB).
6. Destek Vektör Makinaları (SVM).

Bu liste, basit lineer (LR ve LDA) ve lineer olmayan (KNN, CART, NB ve SVM) algoritmalarının iyi bir karışımıdır. Her çalışmadan önce rastgele sayı tohumunu sıfırlarız, böylece her algoritmanın değerlendirilmesi tam olarak aynı veri bölümleri kullanılarak yapılır. Bu, sonuçların doğrudan karşılaştırılabilir olmasını sağlar. Şimdi, algoritmalarımızla modelimizi oluşturalım ve değerlendirelim:

```
# Verisetini test ve eğitim verisi olarak ikiye bölme
dizi = veriseti.values
X = dizi[:,0:4]
Y = dizi[:,4]
dogrulama_boyutu = 0.20
tohum = 7
X_egitim, X_test, Y_egitim, Y_test = train_test_split(X, Y, test_
size=0.33, random_state=tohum)
```

Liste 2.16 *Veri setinin eğitim ve test olarak ayrılması*

Artık model eğitmek için X_egitim ve Y_egitim, eğitilen modelin başarısını ölçmek için kullanmak için X_test ve Y_test kümeleri hazırlanmıştır.

2.5.4 En Başarılı modelin Seçimi

Veriler altı farklı algoritma ile eğitilerek modeller geliştirilecektir ve her birinin doğruluk tahmini elde edilecektir. Modelleri karşılaştırarak en başarılı veya doğruluk değeri en yüksek olan seçilmelidir. **Liste 2.16**'daki kodlar çalıştırıldığında **Liste 2.17**'deki ham sonuçlar elde edilmektedir.

```
LR: 0.980000 (0.040000)
LDA: 0.970000 (0.045826)
KNN: 0.960000 (0.066332)
CART: 0.970000 (0.045826)
NB: 0.970000 (0.045826)
SVM: 0.980000 (0.040000)
```

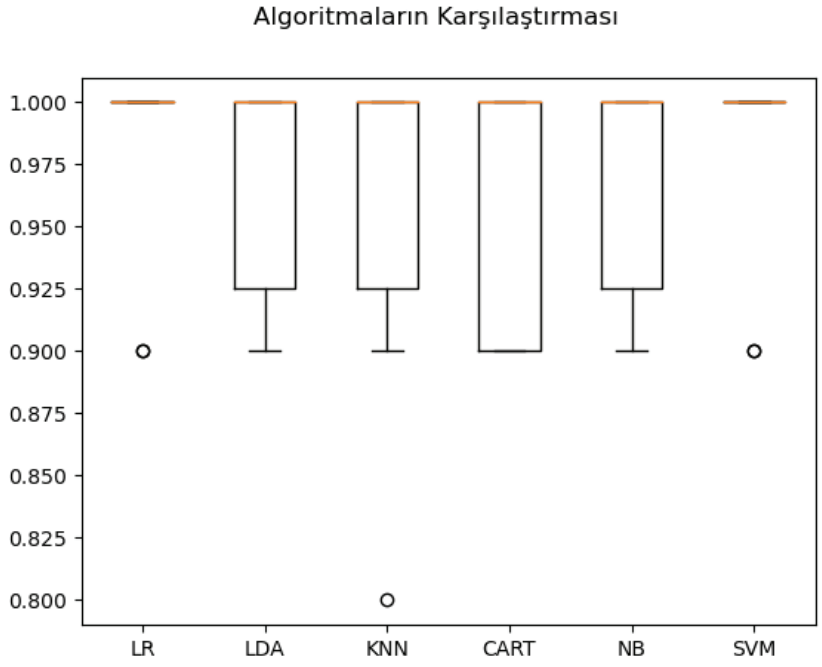
Liste 2.17 *Veri setinin eğitim sonucu algoritmalara göre doğruluk (accuracy) değerleri*

KNN'nin en büyük tahmin doğruluk puanına sahip olduğu görülmektedir. Model değerlendirme sonuçlarının bir grafiğini oluşturabilir ve her modelin yayılımını ve ortalama doğruluğunu karşılaştırarak başarı farkı daha rahat gözlemlenebilir. Her algoritma için bir doğruluk ölçümü popülasyonu vardır, çünkü her algoritma k-kat kesişim doğrulama (k fold cross validation) yöntemi ile 10 kez değerlendirilmiştir.

```
#Algoritmaları Karşılaştır
#Bu kod bloğu, farklı algoritmaların performansını karşılaştırmak
için kutu grafiği kullanır.
#Veri kümesi için farklı algoritmaların sonuçlarını toplamak ve
#Kutu grafiği olarak göstermek için pyplot kütüphanesi kullanılır.
figur = pyplot.figure()#Kutu grafiği için bir figür oluşturulur.
figur.suptitle("Algoritmaların Karşılaştırması") #Başlık eklenir.
#Grafiğin ekseni belirlenir.
eksen = figur.add_subplot(111)
#Kutu grafiği çizilir ve adlar eklenir.
pyplot.boxplot(sonuclar)
eksen.set_xticklabels(isimler)
#Grafiği göster.
pyplot.show()
```

Liste 2.18 *Algoritmaların her birinin doğruluk (accuracy) değer dağılımları*

Çok sayıda örnekten 100% doğruluk elde edildiği için kutu grafiklerinde aralığın en üstte sıkıştığı görülebilir.



Şekil 2.4 *Algoritmaların karşılaştırılması*

2.6 Tahminler Yapın

Örnek olarak verilen problem ve veri seti için test sonucuna göre en doğru model KNN algoritmasıdır. Şimdi, modelin doğruluk oranının doğrulama veri setinde nasıl değiştiği görülmek istenmektedir. Bu, eğitilen modelin doğruluğuna dair tarafsız bir son kontrol sağlayacaktır. Doğrulama setini eğitimde kullanılmaması ve test için ayrılmasının önemli olduğu unutulmamalıdır. Çünkü eğitim sırasında bir hata yapılmışsa veya eğitim setine aşırı uyum sağlanarak bir eğitim gerçekleştiyse (eğitim setine veri sızıntısı olduysa) eğitim modelş doğruluk değeri de test doğruluk değeri de aşırı başarılı bir sonuç verecektir. KNN modeli doğrulama seti üzerinde doğrudan çalıştırılabilir ve sonuçları, son doğruluk puanı, bir karışıklık (confusion) matrisi ve bir sınıflandırma raporu olarak özetlenebilir.

```
# Make predictions on validation dataset
knn = KNeighborsClassifier()
knn.fit(X_egitim, Y_egitim)
predictions = knn.predict(X_dogrulama)
print(accuracy_score(Y_dogrulama, predictions))
print(confusion_matrix(Y_dogrulama, predictions))
print(classification_report(Y_dogrulama, predictions))
```

Liste 2.19 *Doğrulama veri setinden tahminleme*

```
0.92
[[14  0  0]
 [ 0 17  1]
 [ 0  3 15]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       0.85      0.94      0.89        18
 Iris-virginica       0.94      0.83      0.88        18

       accuracy                           0.92        50
      macro avg       0.93      0.93      0.93        50
   weighted avg       0.92      0.92      0.92        50
```

Liste 2.20 *Doğrulama Setinden yapılan tahminlerin çıktısı*

Liste 2.20'den doğruluk (accuracy) oranının 0.92 veya %92 olduğunu görülebilir. Karışıklık (karmaşıklık) matrisi `[[ 7 0 0] [ 0 11 1] [ 0 2 9]]`, yapılan 3 hata hakkında bir fikir verir. Karmaşıklık matrisinin köşegen (7,11,9) değerleri dışında kalan kısımlarındaki değerler tahminlerdeki hatalı değerleri ifade etmektedir. Son olarak, sınıflandırma raporu, her sınıfın doğruluk (accuracy), hassasiyet(recall), f1-skor (f1-score) ve destek (support) değerini göstererek doğrulama veri seti küçük olsa da mükemmel sonuçlar sağlamaktadır.

2.7 Özet

Bu bölümde, Python'da ilk makine öğrenimi projesinin adım adım nasıl tamamlanacağı keşfedildi. Verilerin yüklenmesi ve tahminleme yapılmasını içeren küçük bir son-nokta projesi gerçekleştirildi. Bu sayede, uygulama yapmanın platformu tanımanın en iyi yolu olduğunun keşfedildiği umulmaktadır.

BÖLÜM 3

Giriş

Bir bölgedeki ev fiyatlarının tahminini sınıflandırma probleminden farklı olarak bir değer tahminleme problemidir. Regresyon analizi, bağımlı değişken veya sınıf (ev fiyatları) ile bağımsız değişkenler veya öznitelikler (odaların sayısı, banyo sayısı, konum gibi faktörler) arasındaki ilişkiyi modellemek için kullanılır.

Housing veri seti üzerinde yapılan bir regresyon analizi, ev fiyatlarını etkileyen faktörleri belirlemek ve bu faktörlerin fiyatlardaki etkisini nicel olarak ölçmek için önemli bir araçtır. Bu analiz, emlak sektöründe yatırım yapmak veya ev fiyatlarını belirlemek isteyen bireyler veya kurumlar için değerli bilgiler sunabilir.

3.1 Makine Öğrenmesiyle Regresyon Problemi

Bu projede, Python'da regresyon tahmin problemi modelleme üzerine bir vaka çalışılacaktır. Proje tamamlandıktan sonra aşağıdaki beceriler kazanılmış olacaktır. Bir regresyon tahmin problemini modellemede aşağıdaki adımlar izlenecektir:

1. Veri setini anlama: İlk adım, veri setini incelemek ve içindeki değişkenleri anlamaktır. Veri setinin yapısı, hedef değişken ve özellikler hakkında bilgi edinmek önemlidir.

2. Veri ön işleme: Veri setinde eksik veya yanlış değerler varsa, bunları düzeltmek veya çıkarmak gerekmektedir. Ayrıca, kategorik değişkenleri sayısal değerlere dönüştürmek veya özellik mühendisliği teknikleriyle yeni özellikler oluşturmak da gerekebilir.

3. Veri bölme: Veri setini eğitim ve test veri setlerine ayırmak önemlidir. Eğitim veri seti üzerinde model eğitilecek ve test veri seti üzerinde performansı değerlendirilecektir.

4. Model seçimi: Regresyon tahminsel modelleme sorunları için çeşitli modeller vardır, örneğin doğrusal regresyon, lojistik regresyon, karar ağaçları, rastgele ormanlar vb. Veri setine ve soruna bağlı olarak en uygun model seçilecektir.

5. Model eğitimi: Seçilen modeli eğitim veri seti üzerinde eğitilmesi, modelin özellikleri ve hedef değişkeni arasındaki ilişkiyi öğrenmesini sağlar.

6. Model değerlendirmesi: Eğitilen modeli test veri seti üzerinde değerlendirin. Bu, modelin gerçek dünya verilerine nasıl genelleştiğini

gösterir. Değerlendirme metriklerini kullanarak modelin performansını ölçebilirsiniz.

7. Model iyileştirme: Modelin performansını artırmak için çeşitli yöntemler deneyebilirsiniz. Özellik mühendisliği, hiperparametre ayarlaması, ensemble modelleri gibi teknikler kullanarak modelin performansı geliştirilebilir.

Sonuçların yorumlanması: Son olarak, elde edilen sonuçlar yorumlanır ve iş problemine nasıl uygulayabileceğini değerlendirilir[12].

Model performansını artırmak için veri dönüşümlerinin nasıl kullanacağı gösterilecektir. Veri seti üzerinde dönüşümler yapılarak modelin performansı artırılabilir. Örneğin, sayısal değerlerin normalleştirilmesi, kategorik değişkenlerin kodlanması veya özelliklerin polinom terimlerine dönüştürülmesi gibi dönüşümler kullanılabilir.

Model performansını artırmak için algoritma ayarlamasının nasıl yapıldığı gösterilecektir. Algoritma ayarlaması, modelin hiperparametrelerini optimize etmeyi içerir. Grid arama veya rastgele arama gibi yöntemler kullanarak farklı hiperparametre kombinasyonu denenerek modelin performansı artırılabilir.

Model performansını artırmak için topluluk yöntemleri ve topluluk yöntemlerinin ayarlanmasının nasıl yapıldığı gösterilecektir. Topluluk yöntemleri, birden fazla modelin kombinasyonunu kullanarak daha iyi bir performans elde etmeyi amaçlar. Örneğin, Bagging ve Boosting gibi yöntemler topluluk yöntemlerine örnek olarak verilebilir. Bu yöntemlerde, farklı modelleri eğiterek ve tahminlerini birleştirerek daha güçlü ve daha istikrarlı bir model elde edilebilir.

Topluluk yöntemlerinin ayarlanmasında her bir modelin nasıl eğitileceği, hangi hiperparametrelerin kullanılacağı ve tahminlerin nasıl birleştirileceği gibi adımlar yer alır. Bu ayarlamalar, topluluk yöntemlerinin performansını artırmak için önemlidir. Örneğin, Bagging yönteminde farklı alt örneklemeler oluşturmak veya Boosting yönteminde zayıf öğrenicilerin ağırlıklarını ayarlamak gibi stratejiler kullanılabilir.

Bu projede "Boston Ev Fiyatları" veri seti kullanılacaktır. Her kayıt, Boston'un bir banliyösünü veya kasabasını tarif eder. Veri, 1970 yılında Boston Standart Metropolitan İstatistik Bölgesi (SMSA)'ndan elde edilmiştir. Öznitelikler aşağıdaki şekilde tanımlanmıştır (UCI Makine Öğrenimi Online Veri seti deposundan alınmıştır [13]). Boston Ev Fiyatı veri seti, Boston'un çeşitli banliyölerindeki ev fiyatlarını tahmin etmek için kullanılan bir veri setidir. Veri seti 13 sütundan oluşur ve her sütun bir özelliği temsil eder. Sütun isimlerinin İngilizce ve Türkçesi şu şekilde özetlenebilir:

- CRIM: Şehir başına suç oranı

- ZN: 25.000 metrekareden büyük konut parsellerinin oranı

- INDUS: Şehirdeki perakende iş alanının oranı

- CHAS: Charles Nehri'ne yakın olup olmadığı (1 = evet, 0 = hayır)

- NOX: Nitrik oksit konsantrasyonu (parça başına 10 milyon)

- RM: Ortalama oda sayısı

- AGE: 1940'tan önce inşa edilen sahip evlerin oranı

- DIS: Beş Boston istihdam merkezine ağırlıklı mesafeler

- RAD: Radial otoyollara erişilebilirlik indeksi

- TAX: Tam değer vergisi oranı (10.000$ başına)

- PTRATIO: Şehirdeki öğrenci/öğretmen oranı

- B: Siyah nüfusun yüzdesi (1000(Bk - 0.63)^2)

- LSTAT: Düşük statülü nüfusun yüzdesi

Boston veri setindeki sütunların ingilizce kısaltmaları ve Türkçe karşılıkları şöyledir:

- CRIM: Crime Rate (Suç Oranı). Türkçe'de "SO" olarak kısaltılabilir.

- ZN: Zone (Bölge). Türkçe'de "B" olarak kısaltılabilir.

- INDUS: Industry (Sanayi). Türkçe'de "S" olarak,

- CHAS: Charles River (Charles Nehri). Türkçe'de "CN" olarak,

- NOX: Nitric Oxide (Nitrik Oksit). Türkçe'de "NO" olarak,

- RM: Room Number (Oda Sayısı). Türkçe'de "OS" olarak,

- AGE: Age (Yaş). Türkçe'de "Y" olarak,

- DIS: Distance (Mesafe). Türkçe'de "M" olarak,

- RAD: Radial Highway (Radial Otoyol). Türkçe'de "RO" olarak,

- TAX: Tax Rate (Vergi Oranı). Türkçe'de "VO" olarak,

- PTRATIO: Pupil Teacher Ratio (Öğrenci Öğretmen Oranı). Türkçe'de "ÖÖO" olarak,

- B: Black Population Percentage (Siyah Nüfus Yüzdesi). Türkçe'de "SNY" olarak,

- LSTAT: Lower Status Population Percentage (Düşük Statülü Nüfus Yüzdesi). Türkçe'de "DSNY" olarak kısaltılabilir.

Son olarak hedef değişken Boston verisetinde "MEDV" sütunu, evlerin ortanca değerini (median value) gösterir. Türkçe kısaltması "OD" olabilir.

Liste olarak [SO, B, S, CN, NO, OS, Y, M, RO, VO, OOO, SNY, DSNY, OD] şeklinde gösterilebilir.

3.2 Veri Kümesini Yükleme

Bu proje için gerekli kütüphaneleri yükleyerek başlanmalıdır.

```
# Regresyon Kütühanelerinin Yüklenmesi
import numpy
from numpy import arange
from matplotlib import pyplot
from pandas import read_csv
from pandas import set_option
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_squared_error
```

Liste 3.1 *Kütüphanelerin yüklenmesi*

UCI Machine Learning veri havuzundan veri seti bilgisayara ".csv" dosya uzantısıyla indirilmiş ve uygulamaya dahil edilmiştir.

```
# Verisetinin Yüklenmesi
filename = 'housing.csv'
isimler = ['SO', 'B', 'S', 'CN', 'NO', 'OS', 'Y', 'M', 'RO', 'VO',
'OOO', 'SNY', 'DSNY', 'OD']
veriseti = read_csv(filename, delim_whitespace = True,
names=isimler)
```

Liste 3.2 *Veri setinin Yüklenmesi*

Her özniteliğin kısa adları tanımlanmış ve bir isim listesi olarak atanmıştır. Ayrıca, bu dosyadaki özniteliklerin boşluklarla değil, virgüllerle ayrıldığını ve bunu read_csv() fonksiyonuna "delim_whitespace" argümanı aracılığıyla belirtildiğine dikkat edilmeldir. Artık veriler python ortamına başarılı bir şekilde yüklenmiştir.

3.3 Veri Analizi

Veri teknik analize hazır hale gelmiştir. İstatistiksel olarak incelemek faydalı olacaktır.

3.3.1 Tanımlayıcı İstatistikler

Veri kümesinin boyutları onaylanarak başlanırsa veri kaybı ve ortama akatraımad bir sorun olup olmadığı kolaylıkla anlaşılabilir. Örneğin satır ve sütun sayısı Liste 3.3 ile kontrol edilsin.

```
#veri seti boyutu kontrolü
print(veriseti.shape)
```

Liste 3.3 *Veri setinin boyutlarını sorgulama*

Çalışılacak 506 örnek var ve verilerin OD (Orta Değer) de dahil olmak üzere 14 özniteliği olduğunu onaylanabilir (Liste 3.4):

```
#veri seti boyut çıktısı
(506, 14)
```

Liste 3.4 *Veri setinin boyut çıktısı*

Öznitelik veri tipi incelenirse (Liste 3.5'teki kodlarla) tüm özelliklerin sayısal olduğu Liste 3.6'dan görülmektedir. Çoğunlukla reel sayılar (float) ve bazıları tamsayı (int)'dır.

```
#veri seti boyut çıktısı
print(veriseti.dtypes)
```

Liste 3.5 *Her bir özniteliğin türü yazdırılır*

```
SO      float64
B       float64
S       float64
CN        int64
NO      float64
OS      float64
Y       float64
M       float64
RO        int64
VO      float64
OOO     float64
SNY     float64
DSNY    float64
OD      float64
dtype: object
```

Liste 3.6 *Her bir özniteliğin tür çıktısı*

Verisetinin ilk 20 satırı incelenerek özniteliklerin hangi veri aralığında olduğu hakkında temel bazı bilgiler elde edilebilir (Liste 3.7). Verile-

rin veri tipinin ve değer aralıklarının doğruluğunun gözle kontrolü çok önemlidir (Liste 3.8).

```
# veriseti.head(adet)  ilk "adet" kadar veriyi gösterir.
print(veriseti.head(20))
```

Liste 3.7 *Veri seti ilk 20 satır*

	SO	B	S	CN	NO	OS	Y	M	RO	VO	OOO	SNY	DSNY	OD
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222.0	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311.0	15.2	395.60	12.43	22.9
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311.0	15.2	396.90	19.15	27.1
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311.0	15.2	386.63	29.93	16.5
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311.0	15.2	386.71	17.10	18.9
10	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311.0	15.2	392.52	20.45	15.0
11	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311.0	15.2	396.90	13.27	18.9
12	0.09378	12.5	7.87	0	0.524	5.889	39.0	5.4509	5	311.0	15.2	390.50	15.71	21.7
13	0.62976	0.0	8.14	0	0.538	5.949	61.8	4.7075	4	307.0	21.0	396.90	8.26	20.4
14	0.63796	0.0	8.14	0	0.538	6.096	84.5	4.4619	4	307.0	21.0	380.02	10.26	18.2
15	0.62739	0.0	8.14	0	0.538	5.834	56.5	4.4986	4	307.0	21.0	395.62	8.47	19.9
16	1.05393	0.0	8.14	0	0.538	5.935	29.3	4.4986	4	307.0	21.0	386.85	6.58	23.1
17	0.78420	0.0	8.14	0	0.538	5.990	81.7	4.2579	4	307.0	21.0	386.75	14.67	17.5
18	0.80271	0.0	8.14	0	0.538	5.456	36.6	3.7965	4	307.0	21.0	288.99	11.69	20.2
19	0.72580	0.0	8.14	0	0.538	5.727	69.5	3.7965	4	307.0	21.0	390.95	11.28	18.2

Liste 3.8 *Veri setinin ilk 20 satırının gözlemlenmesi*

Verisetinin istatistiksel açıklamaları yazdırılmıştır. Artık özniteliklerin istatiksel değerleri Liste 3.9 ile hesaplanır ve Liste3.10'da tablo halinde gösterilmiştir.

```
#veri setini 1 hane ondalıklı olarak yazdır tanımlama
set_option("display.precision",1)
print(veriseti.describe())
```

Liste 3.9 *Her bir özniteliğin istatistiksel tanımı yazdırılır*

Minimum ve maksimum değerler de çok değişken araçlardır. Verileri yeniden ölçeklendirerek muhtemelen daha iyi sonuçlar elde edilebilecektir.

	SO	B	S	CN	NO	OS	Y	M	RO	VO	OOO	SNY	DSNY	OD
count	5.1e+02	506.0	506.0	5.1e+02	506.0	506.0	506.0	506.0	506.0	506.0	506.0	506.0	506.0	506.0
mean	3.6e+00	11.4	11.1	6.9e-02	0.6	6.3	68.6	3.8	9.5	408.2	18.5	356.7	12.7	22.5
std	8.6e+00	23.3	6.9	2.5e-01	0.1	0.7	28.1	2.1	8.7	168.5	2.2	91.3	7.1	9.2
min	6.3e-03	0.0	0.5	0.0e+00	0.4	3.6	2.9	1.1	1.0	187.0	12.6	0.3	1.7	5.0
25%	8.2e-02	0.0	5.2	0.0e+00	0.4	5.9	45.0	2.1	4.0	279.0	17.4	375.4	6.9	17.0
50%	2.6e-01	0.0	9.7	0.0e+00	0.5	6.2	77.5	3.2	5.0	330.0	19.1	391.4	11.4	21.2
75%	3.7e+00	12.5	18.1	0.0e+00	0.6	6.6	94.1	5.2	24.0	666.0	20.2	396.2	17.0	25.0
max	8.9e+01	100.0	27.7	1.0e+00	0.9	8.8	100.0	12.1	24.0	711.0	22.0	396.9	38.0	50.0

Liste 3.10 *Her bir özniteliğin temel istatistiksel çıktısı*

Özniteliklerin (bağımsız değişkenlerin) korelasyonları Liste 3.11 ile hesaplanarak Liste 3.12'deki değerler elde edilir.

```
# Korelasyon hesaplama (correlation)
set_option('display.precision',2)
print(veriseti.corr(method='pearson'))
```

Liste 3.11 *Her bir özniteliğin istatistiksel tanımı yazdırılır*

	SO	B	S	CN	NO	OS	Y	M	RO	VO	OOO	SNY	DSNY	OD
SO	1.00	-0.20	0.41	-5.59e-02	0.42	-0.22	0.35	-0.38	6.26e-01	0.58	0.29	-0.39	0.46	-0.39
B	-0.20	1.00	-0.53	-4.27e-02	-0.52	0.31	-0.57	0.66	-3.12e-01	-0.31	-0.39	0.18	-0.41	0.36
S	0.41	-0.53	1.00	6.29e-02	0.76	-0.39	0.64	-0.71	5.95e-01	0.72	0.38	-0.36	0.60	-0.48
CN	-0.06	-0.04	0.06	1.00e+00	0.09	0.09	0.09	-0.10	-7.37e-03	-0.04	-0.12	0.05	-0.05	0.18
NO	0.42	-0.52	0.76	9.12e-02	1.00	-0.30	0.73	-0.77	6.11e-01	0.67	0.19	-0.38	0.59	-0.43
OS	-0.22	0.31	-0.39	9.13e-02	-0.30	1.00	-0.24	0.21	-2.10e-01	-0.29	-0.36	0.13	-0.61	0.70
Y	0.35	-0.57	0.64	8.65e-02	0.73	-0.24	1.00	-0.75	4.56e-01	0.51	0.26	-0.27	0.60	-0.38
M	-0.38	0.66	-0.71	-9.92e-02	-0.77	0.21	-0.75	1.00	-4.95e-01	-0.53	-0.23	0.29	-0.50	0.25
RO	0.63	-0.31	0.60	-7.37e-03	0.61	-0.21	0.46	-0.49	1.00e+00	0.91	0.46	-0.44	0.49	-0.38
VO	0.58	-0.31	0.72	-3.56e-02	0.67	-0.29	0.51	-0.53	9.10e-01	1.00	0.46	-0.44	0.54	-0.47
OOO	0.29	-0.39	0.38	-1.22e-01	0.19	-0.36	0.26	-0.23	4.65e-01	0.46	1.00	-0.18	0.37	-0.51
SNY	-0.39	0.18	-0.36	4.88e-02	-0.38	0.13	-0.27	0.29	-4.44e-01	-0.44	-0.18	1.00	-0.37	0.33
DSNY	0.46	-0.41	0.60	-5.39e-02	0.59	-0.61	0.60	-0.50	4.89e-01	0.54	0.37	-0.37	1.00	-0.74
OD	-0.39	0.36	-0.48	1.75e-01	-0.43	0.70	-0.38	0.25	-3.82e-01	-0.47	-0.51	0.33	-0.74	1.00

Liste 3.12 *Her bir özniteliğin korelasyon çıktısı*

Birçok özniteliğin güçlü bir korelasyonu olduğunu görülmektedir ki bu da ilginçtir (Liste 3.12).

Örneğin:

S ve NO 0.77 ile

S ve M -0.71 ile

VO ve S 0.72 ile

Y ve NO 0.73 ile

M ve NO -0.78 korelasyon değeriyle ile ilişkilidir.

Ayrıca, DSNY özniteliğinin çıktı değişkeni OD ile -0.74 değeriyle iyi bir negatif korelasyona sahip olduğu görülmektedir.

3.4 Veri Görselleştirmeleri

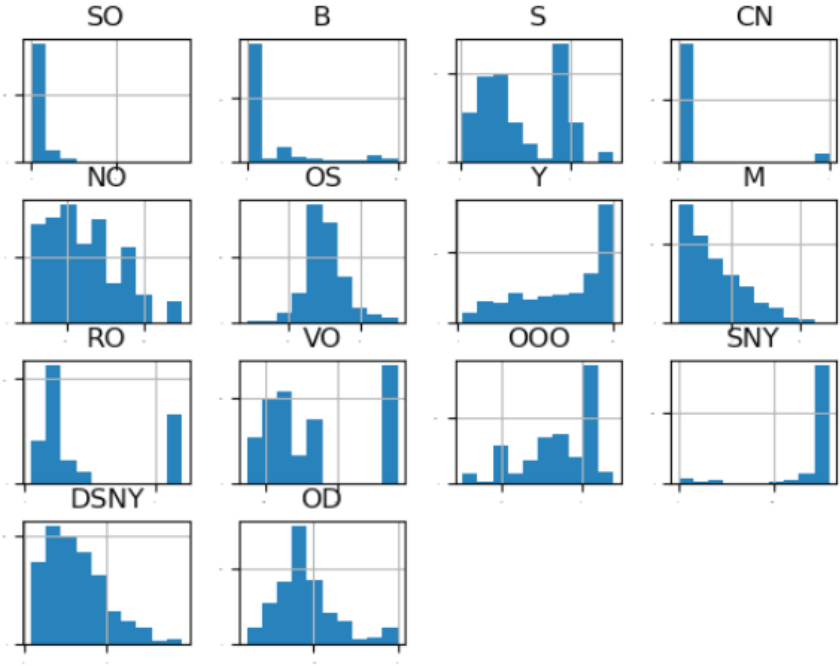
Veri görselleştirmesi, verileri etkili bir şekilde anlamak, keşfetmek ve iletmek için görsel grafikler ve grafiksel temsiller kullanma sürecidir. Veri görselleştirme, karmaşık veri kümelerini daha anlaşılır ve erişilebilir hale getirirken, veri analizi ve yorumlama süreçlerini destekler. Bu yöntem, verilerdeki desenleri, eğilimleri, ilişkileri ve aykırı değerleri keşfetmek ve veri tabanlı kararlar almak için önemli bir araçtır[14].

3.4.1 Tek Modlu Veri Görselleştirmeleri

Bireysel özelliklerin görselleştirmeleri dikkatle incelenerek farklı bakış açıları elde edilmeye çalışılmalı ve fikirlerin harekete geçirilmesi sağlanmalıdır. Bu perspektif ile bakıldığında görselleştirmek ön görülemeyen falyadalar sağlayabilir. Veri dağılımlarını anlamak için her bir özelliğin histogramına bakılabilir (Liste 3.13).

```
# histogramlar
veriseti.hist(sharex=False,sharey=False,
xlabelsize=1,ylabelsize=1)
plt.show()
```

Liste 3.13 *Her bir özniteliğin histogramını çizdirme*

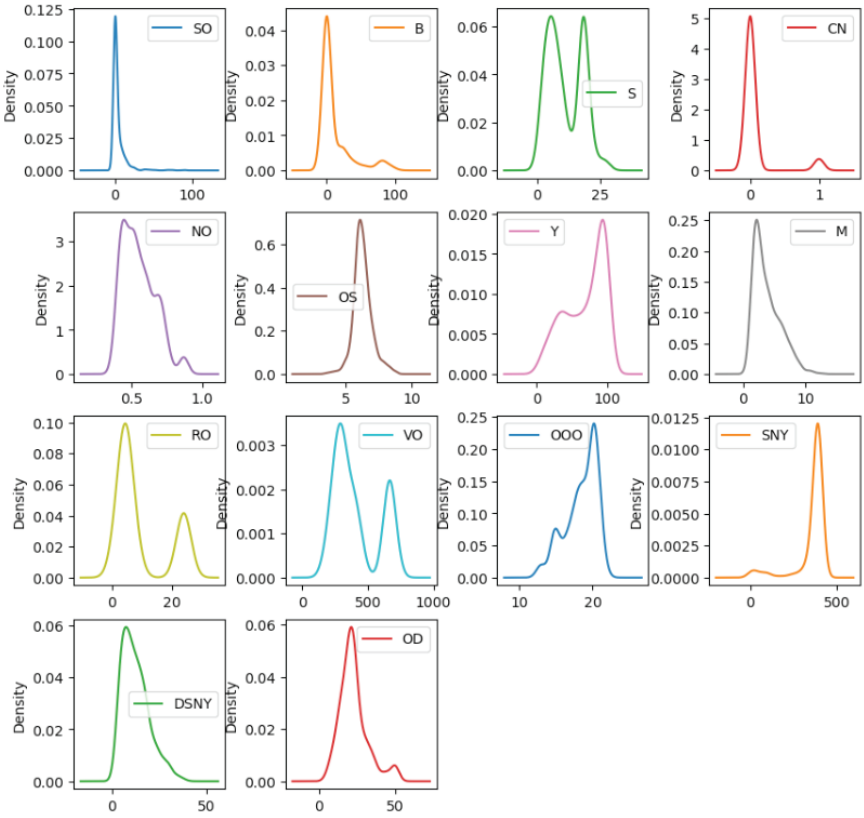


Şekil 3.1 *Her bir özniteliğin histogram grafikleri*

CN, OD, Y ve B gibi bazı özniteliklerin üstel bir dağılıma sahip olduğu Şekil 3.1'den görülebilir). OOO ve DSNY gibi diğerlerinin çift modlu bir dağılıma sahip olabileceği görülmektedir. Her bir özniteliğin yoğunluk grafiği (density plot) çizdirilir (Liste 3.14) ve Şekil 3.2'deki grafikler elde edilir.

```
# yoğunluk
# Veriler yoğunluk grafiği olarak çizdirilir
veriseti.plot(kind="density", subplots=True, layout=(4,4),
sharex=False, figsize=(10,10))
# Alt ve üst, sol ve sağ grafikler arasındaki boşluğu ayarla
pyplot.subplots_adjust(hspace=0.2)
# Sol ve sağ grafikler arasındaki boşluğu ayarla
pyplot.subplots_adjust(wspace=0.4)
# Grafiği göster
pyplot.show()
```

Liste 3.14 *Her bir özniteliğin yoğunluk grafiğini (density plot) çizdirme*

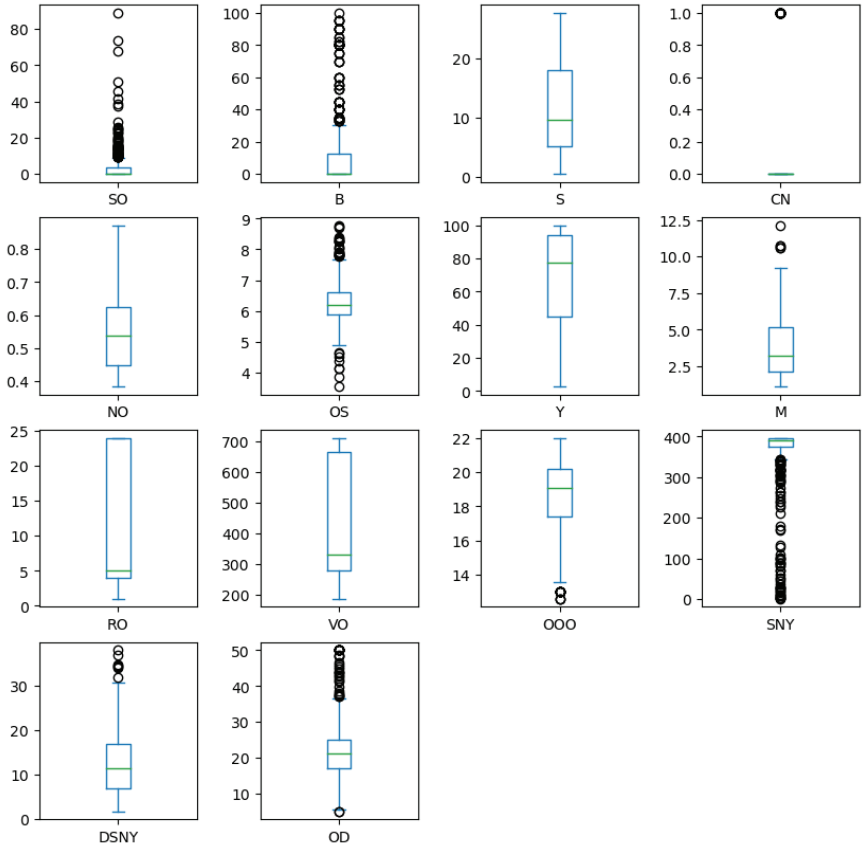


Şekil 3.2 *Her bir özniteliğin yoğunluk grafiği*

Özniteliklerin kutu grafiği Liste 3.15 ile çizdirdilirse Şekil 3.3'teki kutu grafikleri elde edilecektir.

```
# Veriler yoğunluk grafiği olarak çizdirilir
veriseti.plot(kind="density", subplots=True, layout=(4,4),
sharex=False, figsize=(10,10))
pyplot.subplots_adjust(hspace=0.2)
pyplot.subplots_adjust(wspace=0.4)
# Grafiği göster
pyplot.show()
```

Liste 3.15 *Her bir özniteliğin yoğunluk grafiğini (density plot) çizdirerek görselleştirelim*



Şekil 3.3 *Her bir özniteliğin kutu grafiği*

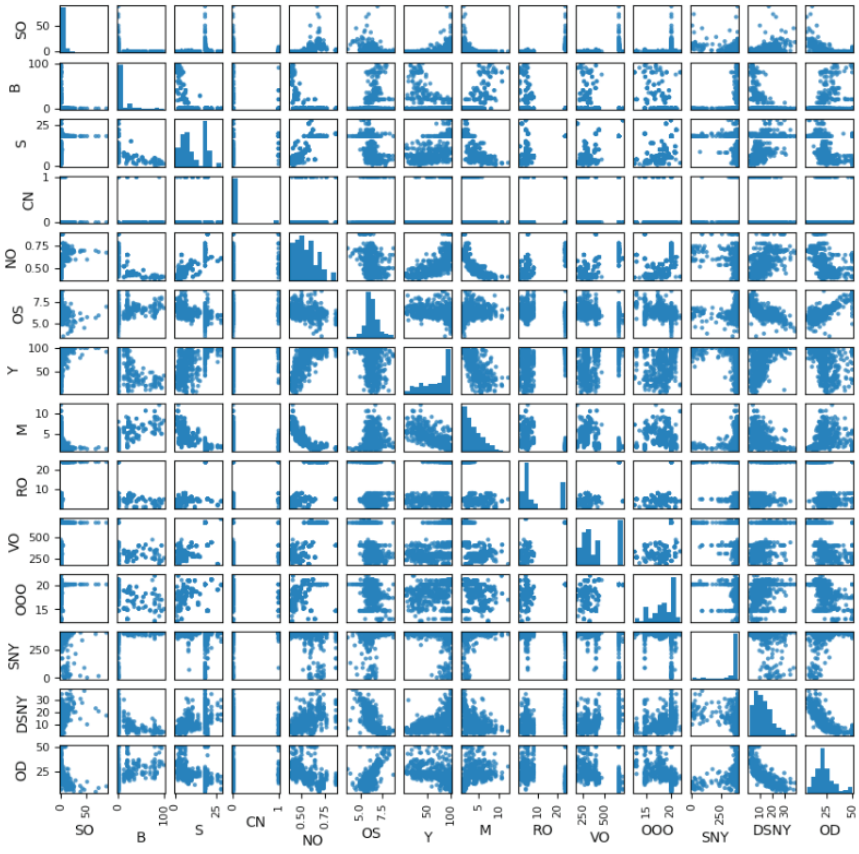
Bu grafikler birçok dağılımda çarpıklığı göstermeye yardımcı olur, bu nedenle kutu grafiklerinin uç noktalarındaki veriler aykırı veri olarak görülebilir.

3.4.2 Çok Modlu Veri Görselleştirme

Değişkenler arasındaki etkileşimlerin görselleştirmelerine bakalım. Başlamak için en iyi yer, bir saçılım (scatter) çizelgesi matrisidir. Evet, scatter_matrix komutunda diagonal parametresi varsayılan olarak "hist" değerindedir, bu nedenle çizgi matrisi içindeki diyagonal (köşegen) grafikler histogramlar olacaktır. Saçılma grafiğini çizdiren kod Liste 3.16'da verilmiştir.

```
# Scatter çizgi matrisini oluştur
scatter_matrix = pd.plotting.scatter_matrix(veriseti, figsize=(10,
10), alpha=0.8, diagonal='hist')
pyplot.subplots_adjust(wspace=0.2, hspace=0.2)
pyplot.show()
```

Liste 3.16 *Her bir özniteliğin diyagonal histogramını çizdirme*



Şekil 3.4 *Özniteliklerin diyagonal matrisi*

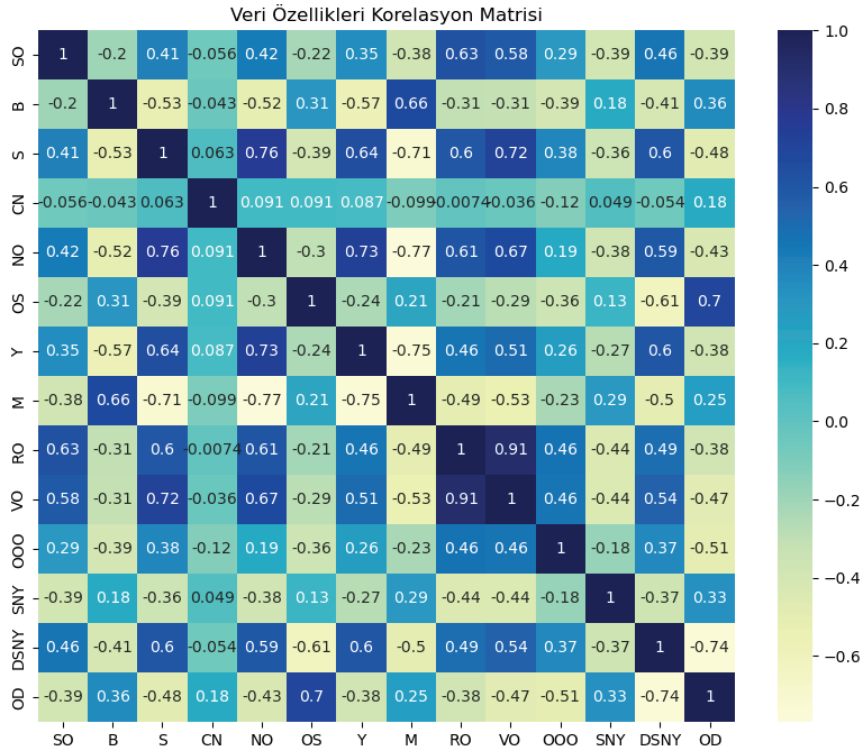
Değişkenler arasındaki etkileşimleri ve ilişkileri gözlemlemek için bazı görselleştirmelere bakalım. En iyi başlangıç noktası, korelasyon dağılım grafiği matrisidir (Şekil 3.5). Korelasyonu yüksek bazı özniteliklerin ilişki içinde iyi bir yapı gösterdiklerini görebiliyoruz. Doğrusal değil, ancak tahmin edilebilir eğri ilişkileri vardır. Korelasyonu görebilmek için Liste 3.17'deki kodlar kullanılır.

```
# Seaborn kütüphanesi ile ısı haritası Korelasyona göre oluşturma
import seaborn as sns
corr_matrix = veriseti.corr()
pyplot.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap="YlGnBu")
pyplot.title('Veri Özellikleri Korelasyon Matrisi')
pyplot.show()
```

Liste 3.17 *Öznitelikler arasındaki korelasyon*

Koyu mavi renk pozitif korelasyonu, sarı renk ise negatif korelasyonu gösterir (Şekil 3.5). Ayrıca, daha sonra modellerin doğruluğunu iyileş-

tirmeyi amaçlayan kandidozları çıkarmaya yönelik bazı koyu kırmızı ve koyu mavi renkler de görebiliriz.



Şekil 3.5 *Özniteliklerin Korelasyon Matrisi*

3.4.3 Çıkarımların Özeti

Bu veri kümesinde birçok yapı vardır. Modelleme doğruluğunu artırabilecek yapının daha iyi ortaya çıkarılabilmesi için daha sonra kullanabilecek dönüşümlerin düşünülmesi gerekmektedir. Aşağıdaki ek adımlar model iyileştirmek adına denenebilir. Bunlar:

1. Özellik seçimi yaparak en çok korelasyonlu özellikleri çıkarma,
2. Değişen ölçeklerin etkisini azaltmak için veri setini normalize etme ve
3. Farklı veri dağılımlarının etkisini azaltmak için veri setini standartlaştırma gibi.

Verilerin ayrıştırılması (discretization) seçeneği de keşfedilebilir. Bu karar ağacı algoritmaları için genellikle doğruluğu arttırır.

3.5 Doğrulama Veri Seti

Veri ayırma (hold-out) seti kullanmak doğrulama için iyi bir yaklaşımdır. Bu, analiz ve modelde kullanılan veri setinin bir kısmıdır. Bu veri seti, proje sonunda modelin doğruluğunu görülmemiş veride test etmek için kullanılacaktır. Bu doğrulama adımı uygulama eğitim sürecinde bir hata yapıp yapılmadığının görülmesine ve görülmemiş veride tahmin doğruluğunun test edilmesine imkân vermektedir. Veri setlerinin %80'ini modelleme için %20'si doğrulama için eğitimden önce Liste 3.18'deki kod parçasıyla ayrılabilir.

```
# Veriyi doğrulama için eğitimden önce ayırma
dizi = veriseti.values
X = dizi[:,0:13]
Y = dizi[:,13]
dogrulama_boyutu = 0.20
tohum = 7
X_egitim, X_dogrulama, Y__egitim, Y_dogrulama = train_test_
split(X, Y, test_size=dogrulama_boyutu, random_state=tohum)
```

Liste 3.18 *Veri setinin eğitim ve doğrulama olmak üzere ayrılması*

3.6 Algoritmaları Değerlendirme

Temelde bu problemde hangi algoritmaların iyi sonuç vereceği özgülemeyebilir. İçimizden bir his Lineer Regresyon ve ElasticNet gibi regresyon algoritmalarının iyi sonuç vereceğini öngörebilir. Ayrıca, karar ağaçları ve SVM'nin de iyi sonuç verebilmesi de mümkündür. Öngörülerin yönünün tayin edilmesi ve modellemenin ilk adımı olarak 10-katlı çapraz doğrulamada kullanılmak üzere Test setinin oluşturulmasıyla başlanır. Veri kümesi çok küçük değil ve bu, iyi bir standart test seti yapılandırmasıdır. Algoritmaları, Ortalama Kare Hata (MSE) metriği kullanarak değerlendireceğiz. MSE, tüm tahminlerin ne kadar yanlış olduğunu (0 hatasızlık durumudur) genel olarak gösterecektir (Liste 3.19).

```
# Test seçenekleri ve doğrulama parametreleri
k_fold_sayisi = 10
tohum = 7
skor = "neg_mean_squared_error"
```

Liste 3.19 *Test seçenekleri ve doğrulama parametreleri*

Bu problemde performans ölçümü için birkaç farklı algoritma ile eğitimler yapılmalı ve performans değerleri görülmelidir. Bu regresyon problemine özgü çeşitli algoritmalar tercih edilmelidir (Liste 3.20).

Bunlar:

1. **Lineer Algoritmalar:** Lineer Regresyon (LR), Lasso Regresyon (LASSO) ve ElasticNet (EN).

2. **Nonlineer Algoritmalar:** Sınıflandırma ve Regresyon Ağaçları (CART), Destek Vektörü Regresyonu (SVR) ve k-En Yakın Komşular (KNN).

```
# Hızlı test için algoritmalar
models = []
models.append(('LR', LinearRegression()))
models.append(('LASSO', Lasso()))
models.append(('EN', ElasticNet()))
models.append(('KNN', KNeighborsRegressor()))
models.append(('CART', DecisionTreeRegressor()))
models.append(('SVR', SVR())) # Support Vector Regression
```

Liste 3.20 *Eğitimi gerçekleştirilecek algoritmalar listesi oluşturuldu*

Tüm algoritmalar varsayılan ayarlama parametrelerini kullanmaktadır. Her algoritma için MSE(Mean Square Error)'nin ortalamasını ve standart sapması hesaplanarak sonuçlar ayrı ayrı tutulmalıdır (Liste 3.22).

```
# Hızlı test için algoritmalar
sonuclar = []
isimler = []
for isim, model in modeller:
    kfold = KFold(n_splits=k_fold_sayisi, shuffle=True, random_
state=tohum)
    cv_sonuclar = cross_val_score(model, X_egitim, Y_egitim,
cv=kfold, scoring=skor)
    results.append(cv_sonuclari)
    names.append(isim)
    msj = "%s: %f (%f)" % (isim, cv_sonuclar.mean(), cv_sonuclar.
std())
    print(msj)
```

Liste 3.22 *Algoritma performanslarının değerlendirilmesi*

MSE değerlerine göre, en düşük MSE değeri ilk sırada LR (Lineer Regresyon) ve hemen ardından CART (Sınıflandırma ve Regresyon Ağacı) modelleri yer almaktadır.

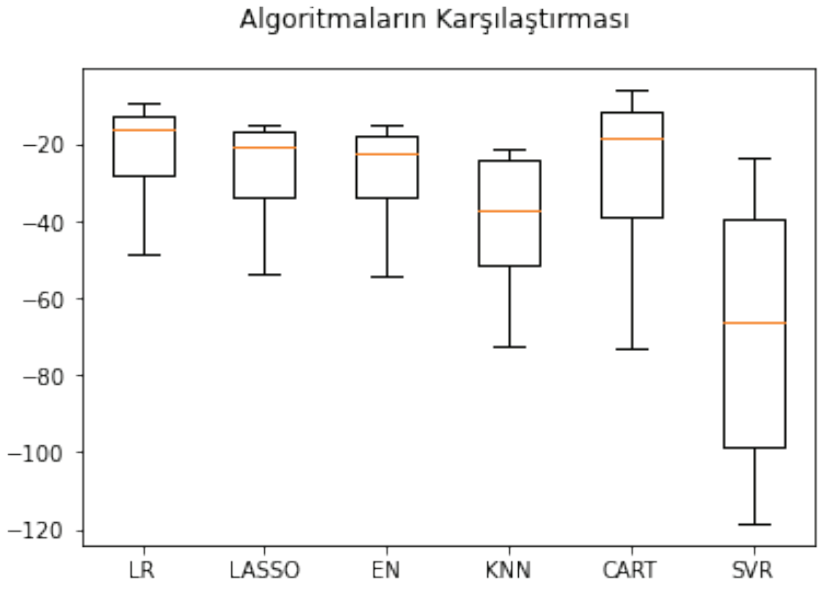
```
LR: -22.006009 (12.188886)
LASSO: -27.105803 (13.165915)
EN: -27.923014 (13.156405)
KNN: -39.808936 (16.507968)
CART: -25.206493 (18.290140)
SVR: -67.824705 (32.801530)
```

Liste 3.23 *Algoritmaların değerlendirilme sonucu*

Regresyon algoritmaları için benzer dağılımlar gözlemlenebilir ve belki de CART modeli için skorların daha dar bir dağılımına sahip olunabilir. Liste 3.24'teki kod ile algoritma performansını görselleştirmek için bir kutu grafiği çizdirilmiştir.

```
#Algoritmaları Karşılaştırılması
figur = pyplot.figure()
#Başlık eklenir.
figur.suptitle("Algoritmaların Karşılaştırması")
#Grafiğin ekseni belirlenir.
eksen = figur.add_subplot(111)
#Kutu grafiği çizilir ve adlar eklenir.
pyplot.boxplot(sonuclar)
eksen.set_xticklabels(isimler)
#Grafiği göster.
pyplot.show()
```

Liste 3.24 *Algoritma performansındaki farklılıkların görselleştirilmesi*



Şekil 3.6 *Algoritma performanslarının karşılaştırılması*

Verilerin farklı ölçekleri, tüm algoritmaların başarısını olumsuz etkileyebilir ve muhtemelen SVR ve KNN algoritmaları için daha fazla etki edebilir (Şekil 3.6).

3.7 Algoritmaları Değerlendirme: Standartlaştırma

Ham verilerin farklı ölçekleri bazı algoritmaların başarısını olumsuz etkileyebilmektedir. Aynı algoritmaları veri setinin standartlaştırılmış bir kopyasıyla değerlendirelim. Bu, her bir öznitelik için ortalamasının sıfır ve standart sapmasının 1 olduğu şekilde verinin dönüştürüldüğü bir süreçtir. Veri dönüşümü sırasında veri sızıntısı (data leakage) oluşmasının önlenmesi gerekmektedir. Sızıntıyı önlemenin iyi bir yolu, veriyi standartlaştıran boru hatları (pipelines) kullanılmaktadır. Borular, çapraz doğrulama test kümesindeki her katman için, veriyi standartlaştıran ve modeli kuran

hatlar oluşturacaktır (Liste 3.24). Bu sayede, standartlaştırılmış veri ile her bir modelin ne kadar performans gösterdiğine dair adil bir tahmin elde edilebilir.

```
# Veri kümesini standartlaştırın (Ölç.=Ölçeklendirilmiş, Ölçci.=
Ölçeklendirici)
borular = []
borular.append((Ölç.LR', Pipeline([('Ölçci',
StandardScaler()),('LR', LinearRegression())])))
borular.append((Ölç.Lasso', Pipeline([('Ölçci',
StandardScaler()),('Lasso', Lasso())])))
borular.append((Ölç.ElasticNet', Pipeline([('Ölçci',
StandardScaler()),('EN', ElasticNet())])))
borular.append((Ölç.KNN', Pipeline([('Ölçci',
StandardScaler()),('KNN', KNeighborsRegressor())])))
borular.append((Ölç.CART', Pipeline([('Ölçci',
StandardScaler()),('CART', DecisionTreeRegressor())])))
borular.append((Ölç.SVR', Pipeline([('Ölçci',
StandardScaler()),('SVR', SVR())])))
sonuclar = []
isimler = []
# Modelleri ölçeklendirilmiş veri ile eğiterek sonuçları
değerlendir
for isim, model in borular:
    k_fold = KFold(n_splits=k_fold_sayisi, shuffle=True, random_
state=tohum)
    cv_sonuclar = cross_val_score(model, X_egitim, Y_egitim, cv=k_
fold, scoring=skor)
    sonuclar.append(cv_sonuclar)
    isimler.append(isim)
    msj = "%s: %f (%f)" % (isim, cv_sonuclar.mean(), cv_sonuclar.
std())
    print(msj)
```

Liste 3.24 *Standartlaştırılmış veri setinde algoritmaları değerlendirme*

Örneği çalıştırdığımızda, ortalama kare hataların listesini gösteren bir sonuç elde ederiz. Ölçeklendirmenin KNN modelini diğer modellere göre daha düşük bir hata ile etkilediği görülebilir (Liste 3.25).

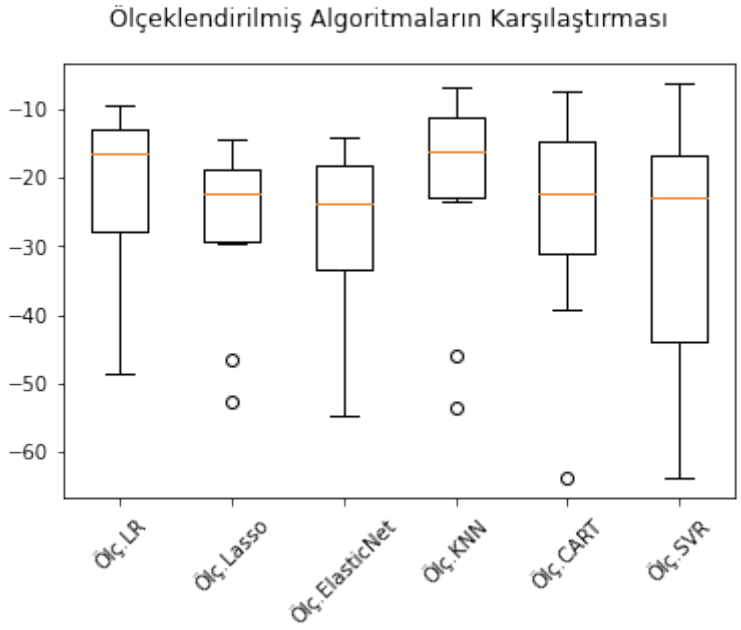
```
Ölç.LR:          -22.006009 (12.188886)
Ölç.Lasso:       -27.205896 (12.124418)
Ölç.ElasticNet: -28.301160 (13.609110)
Ölç.KNN:         -21.456867 (15.016218)
Ölç.CART:        -25.515105 (17.950251)
Ölç.SVR:         -29.570433 (18.052964)
```

Liste 3.25 *Standartlaştırılmış veri setinde sonuçlar*

Şimdi, çapraz doğrulama katları arasındaki skorların (puanların) dağılımına Liste 3.26'daki kodlarla bakıldığında Şekil 3.7'deki dağılım gözlemlenir.

```
#Algoritmaların Karşılaştırılması
figur = pyplot.figure()
#Başlık eklenir.
figur.suptitle("Algoritmaların Karşılaştırması")
eksen = figur.add_subplot(111) #Grafiğin ekseni belirlenir.
pyplot.boxplot(sonuclar) #Kutu grafiği çizilir ve adlar eklenir.
eksen.set_xticklabels(isimler)
pyplot.show()#Grafiği göster.
```

Liste 3.26 *Algoritma performansındaki farklılıkların görselleştirme*



Şekil 3.7 *Standartlaştırılmış veride algoritma performanslarının karşılaştırılması*

Hata oranının hem dar bir dağılımı olduğu hem de en düşük skora sahip olan modelin KNN olduğunu görebiliriz.

3.8 Parametre Ayarlama ve Sonuçları İyileştirme

Önceki bölümde elde edilen sonuçlar, KNN'nin veri kümesinin ölçeklendirilmiş bir versiyonunda iyi sonuçlar aldığını göstermektedir. Ancak daha iyi yapılabilir mi? KNN için varsayılan komşu sayısı değeri 7'dir. Grid search kullanarak farklı komşu sayılarının bir setini deneyebilir ve puanı iyileştirip iyileştiremeyeceği görülebilir. Liste 3.27'deki örnekte 1'den 21'e kadar tek k değerleri denenmektedir. Her k değeri (komşu sayısı), eğitim veri kümesinin standartlaştırılmış bir kopyasında 10-katlı çapraz doğrulama kullanılarak değerlendirilir.

Ayrıca veri ön işleme kısmında StandartScaler'ı kullanır ve ardından KNN algoritmasının en iyi parametrelerini bulmak ve performans karşılaştırması yapmak için Grid Search yöntemini kullanır. K değişkeni değiştirilerek KNN'nin performansı değerlendirilir (Liste 3.27). En iyi k değeri bulunur. GridSearchCV() fonksiyonu çağırırken skorlama parametresi 'neg_mean_squared_error''dir. Grid Search işlemi sonucunda ortalama ve standart sapma yazdırılır (Liste 3.28). En iyi sonuçlar ve parametreler görüntülenir.

```
# KNN Algoritması Ayarlamaları
scaler = StandardScaler().fit(X_egitim)
ölçeneklendirilmis_X = scaler.transform(X_egitim)
k_degerleri = numpy.array([1,3,5,7,9,11,13,15,17,19,21])
parametre_gridi = dict(n_neighbors=k_degerleri)
model = KNeighborsRegressor()
kfold = KFold(n_splits=k_fold_sayisi, shuffle=True, random_
state=tohum)
grid = GridSearchCV(estimator=model, param_grid=parametre_gridi,
scoring=skor, cv=kfold)
grid_result = grid.fit(ölçeneklendirilmis_X, Y_egitim)
```

Liste 3.27 *Standartlaştırılmış Veri setinden Ayarlamaları yapılmış KNN ile sonuç alma*

```
print("En İyi Sonuç: %f, En İyi Parametreler: %s" % (grid_result.
best_score_, grid_result.best_params_))
ortalama = grid_result.cv_results_["mean_test_score"]
standart_sapma = grid_result.cv_results_["std_test_score"]
parametreler = grid_result.cv_results_["params"]
for ortalama, ss, parametre in zip(ortalama, standart_sapma,
parametreler):
    print("%f (%f) için parametreler: %r" % (ortalama, ss,
parametre))
```

Liste 3.28 *Ayarlama Sonrası KNN Algoritması çıktısı*

Aşağıda, k için en iyi performans gösteren değeri, ortalama ve standart sapma görülmektedir (Liste 3.29).

```
En İyi Sonuç: -19.497829, En İyi Parametreler: {'n_neighbors': 1}
-19.497829 (15.769847) için parametreler: {'n_neighbors': 1}
-19.977984 (13.803973) için parametreler: {'n_neighbors': 3}
-21.270967 (14.833544) için parametreler: {'n_neighbors': 5}
-21.577292 (14.952592) için parametreler: {'n_neighbors': 7}
-21.001075 (14.701297) için parametreler: {'n_neighbors': 9}
-21.490306 (14.866957) için parametreler: {'n_neighbors': 11}
-21.268533 (14.454969) için parametreler: {'n_neighbors': 13}
-21.968092 (14.209894) için parametreler: {'n_neighbors': 15}
-22.739880 (14.492752) için parametreler: {'n_neighbors': 17}
-23.506901 (14.903224) için parametreler: {'n_neighbors': 19}
-24.240303 (15.156565) için parametreler: {'n_neighbors': 21}
```

Liste 3.29 *Ayarlama Sonrası KNN Algoritması çıktısı*

3.9 Ensemble Yöntemleri

Bu problemdeki algoritmaların performansını iyileştirmek için kullanabileceğimiz başka bir yöntem de ensemble yöntemleridir. Bu bölümde, iki boosting ve iki bagging yönteminden dört farklı ensemble makine öğrenimi algoritmasını değerlendirilecektir:

1. Boosting Yöntemleri: AdaBoost (AB) ve Gradient Boosting (GBM).

2. Bagging Yöntemleri: Rastgele Ormanlar (RF) ve Extra Trees (ET).

Önceki gibi, 10-katlı çapraz doğrulama ve her kat için eğitim verisini standartlaştıran borular kullanılacaktır.

```
# Topluluk Yöntemleri
ensembles = []
ensembles.append(('Ölçekli AB', Pipeline([('Ölçci',
StandardScaler()),('AB', AdaBoostRegressor())])))
ensembles.append(('Ölçekli GB', Pipeline([('Ölçci',
StandardScaler()),('GBM', GradientBoostingRegressor())])))
ensembles.append(('Ölçekli RF', Pipeline([('Ölçci',
StandardScaler()),('RF', RandomForestRegressor())])))
ensembles.append(('Ölçekli ET', Pipeline([('Ölçci',
StandardScaler()),('ET', ExtraTreesRegressor())])))
sonuclar = []
isimler = []
for isim, model in ensembles:
    kfold = KFold(n_splits=k_fold_sayisi, shuffle=True, random_
state=tohum)
    cv_sonuclar = cross_val_score(model, X, Y, cv=kfold,
scoring='neg_mean_squared_error')
    sonuclar.append(cv_sonuclar)
    isimler.append(isim)
    mesaj = "%s: %f (%f)" % (isim, cv_sonuclar.mean(), cv_
sonuclar.std())
    print(mesaj)
```

Liste 3.30 *Standartlaştırılmış veri setinde ensemle değerlendirme*

Yukarıdaki (Liste 3.30'daki) kodu çalıştırmak, varsayılan parametreleri kullanarak her yöntem için ortalama kare hata hesaplar. Önceki bölümlerdeki lineer ve nonlineer algoritmalardan genellikle daha iyi puanlar alındığını görülebilir (Liste 3.31).

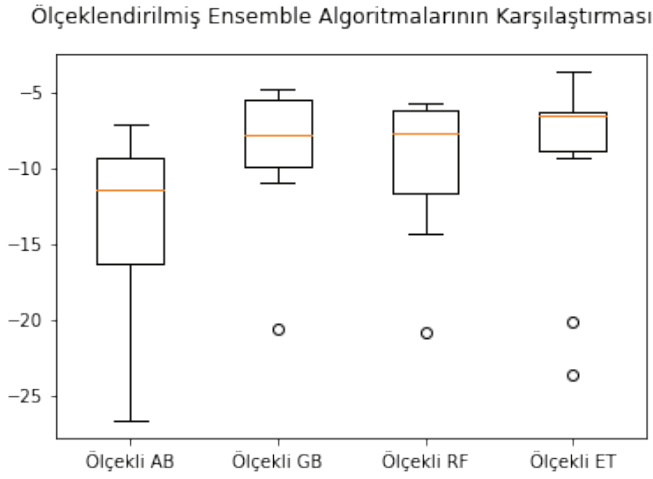
```
Ölçekli AB: -13.706568 (6.200656)
Ölçekli GB: -8.828411 (4.366591)
Ölçekli RF: -10.803347 (5.842784)
Ölçekli ET: -9.182650 (5.361260)
```

Liste 3.31 *Ensemle algoritmalarının değerlendirme çıktıları*

Çapraz doğrulama katmanları boyunca puanların dağılım grafiğe de çizebilir (Liste 3.32) ve (Şekil 3.8).

```
# Algoritmaları Karşılaştır
fig = pyplot.figure()
fig.suptitle('Ölçeklendirilmiş Ensemble Algoritmalarının
Karşılaştırması')
ax = fig.add_subplot(111)
pyplot.boxplot(sonuclar)
ax.set_xticklabels(isimler)
pyplot.show()
```

Liste 3.32 *Standartlaştırılmış veri setinde ensemble algoritma performansındaki farklılıklar*



Şekil 3.8 *Ölçeklendirilmiş Ensemble Algoritmalarının Karşılaştırması*

Gradient Boosting'in daha iyi bir ortalama puanı olduğu görülmektedir. Ayrıca Extra Trees'in benzer bir dağılıma sahip olduğu ve belki de daha iyi bir medyan puanına sahip olduğu görülmektedir. Ensemble tekniklerinin varsayılan parametreleri kullandığı düşünüldüğünde ve parametre ayarlaması yapıldığında muhtemelen daha iyi bir performans elde edilecektir. Bir sonraki bölümde, Gradient Boosting'in performansını daha da iyileştirmek için ayarlamalar yapmaya çalışılacaktır.

3.10 Ensemble Yöntemleriyle Ayarlama Yapma

Boosting (performans iyileştirme) için varsayılan aşamaların sayısı (n_estimators) 100'dür. Bu, Gradient Boosting için ayarlamaya uygun iyi bir parametre olabilir. Genellikle boost etme aşama sayısı arttıkça performans iyileşir ancak eğitim süresi uzar. Bu bölümde, gradient boosting için aşamaların sayısı ayarlanmaya çalışılacaktır. Aşağıda, 50'den 400'e 50 artışlarla n estimators değerlerini tanımlayan bir parametre grubu (grid'i) tanımlanmaktadır (Liste 3.33). Her ayar, 10-katlı çapraz doğrulama kullanılarak değerlendirilir.

```
# Ölçekli GBM'i Ayarlama
scaler = StandardScaler().fit(X_egitim)
yenidenOlceklenmisX = scaler.transform(X_egitim)
param_grid = dict(n_estimators=numpy.
array([50,100,150,200,250,300,350,400]))
model = GradientBoostingRegressor(random_state=tohum)
kfold = KFold(n_splits=k_fold_sayisi,shuffle=True, random_state=tohum)
grid = GridSearchCV(estimator=model, param_grid=param_grid,
scoring=skor, cv=kfold)
grid_sonucu = grid.fit(yenidenOlceklenmisX, Y_egitim)
```

Liste 3.33 *Ölçeklenmiş veri setinde GBM'i ayarlama*

Önceki gibi, en iyi yapılandırma özetlenebilir ve her farklı yapılandırmayla performansın nasıl değiştiği gözlemlenebilir (Liste 3.34).

```
# En iyi sonuçları gösterme
print("%s kullanarak en iyi sonuç: %f" % (grid_result.best_params_,
grid_result.best_score_))
means = grid_result.cv_results_["mean_test_score"]
stds = grid_result.cv_results_["std_test_score"]
params = grid_result.cv_results_["params"]
for mean, stdev, param in zip(means, stds, params):
    print("%r iken Ortalama=%f Standart Sapma=(%f) " % (param, mean,
stdev ))
```

Liste 3.34 *Ölçeklenmiş veri setinde GBM'in performansını gösterme*

En iyi yapılandırmanın n_estimators= 400 olduğu ve -9.356471 ortalama karesel hatayla sonuçlandığı görülmektedir (Liste 3.35). Bu, ayarlanmamış yöntemden yaklaşık 0.65 birim daha iyidir.

```
# En iyi sonuçlar
{'n_neighbors':1} kullanarak en iyi sonuç: -19.497829
{'n_neighbors':1} iken Ortalama=-19.497829 Standart Sapma=(15.769847)
{'n_neighbors':3} iken Ortalama=-19.977984 Standart Sapma=(13.803973)
{'n_neighbors':5} iken Ortalama=-21.270967 Standart Sapma=(14.833544)
{'n_neighbors':7} iken Ortalama=-21.577292 Standart Sapma=(14.952592)
{'n_neighbors':9} iken Ortalama=-21.001075 Standart Sapma=(14.701297)
{'n_neighbors':11} iken Ortalama=-21.490306 Standart Sapma=(14.866957)
{'n_neighbors':13} iken Ortalama=-21.268533 Standart Sapma=(14.454969)
{'n_neighbors':15} iken Ortalama=-21.968092 Standart Sapma=(14.209894)
{'n_neighbors':17} iken Ortalama=-22.739880 Standart Sapma=(14.492752)
{'n_neighbors':9} iken Ortalama=-23.506901 Standart Sapma=(14.903224)
{'n_neighbors':21} iken Ortalama=-24.240303 Standart Sapma=(15.156565)
```

Liste 3.35 *Ölçeklenmiş veri setinde GBM'in performansın çıktısı*

Artık model kaydedilerek modelleme süreci sonlandırılarak farklı amaçlar için tekrar tekrar kullanabilir.

3.11 Modeli Sonlandırma

Bu bölümde, gradient boosting modeli sonlandırılacaktır. Eğitimden önce ayırmış doğrulama veri setinde performans değerlendirmesi yapılacak-

tır. İlk olarak, modelin hazırlanması ve tam eğitim veri setinde eğitilmesi gerekmektedir. Bu, eğitimden önce veri setini standartlaştırmayı gerektirir (Liste 3.36).

```
#Modeli Hazırlama
scaler=StandardScaler().fit(X_egitim)
yenidenOlceklenmisX = scaler.transform(X_egitim)
model =GradientBoostingRegressor(random_state=tohum,n_
estimators=400)
model.fit(yenidenOlceklenmisX, Y_egitim)
```

Liste 3.36 *Model Sonlandırma Yapısı*

Daha sonra, doğrulama veri seti için girdileri ölçeklendirebilir ve tahminler oluşturabiliriz.

```
# Doğrulama Verisetini dönüştürme
yenidenOlceklenmisDogrulamaX = scaler.transform(X_dogrulama)
tahminler = model.predict(yenidenOlceklenmisDogrulamaX)
print(mean_squared_error(Y_dogrulama, tahminler))
#çıktı değeri
11.902132586880027
```

Liste 3.37 *Sonlandırılmış Modeli Değerlendirme ve Çıktısı*

Tahmini MSE (ortalama karesel hata) 11.90 olarak hesaplanmıştır.

3.12 Özet

Bu bölümde, Python kullanarak bir regresyon tahmini makine öğrenimi problemi çözüldü. Sırayla gerçekleştirilen adımlar şunlardı:

1. Problem tanımı (Boston ev fiyatları verileri).
2. Veri kümesini yükleme.
3. Verileri analiz etme (bazı çarpık dağılımlar ve korelasyonlu öznitelikler)
4. Algoritmaları değerlendirme (Doğrusal Regresyon uygun bulundu)
5. Standartlaştırma ile Algoritmaları Değerlendirme (KNN uygun bulundu)
6. Algoritma Ayarlama (KNN için K=3 en başarılı olandı)
7. Ensemble Yöntemleri (Bagging ve Boosting, Gradient Boosting uygun bulundu)
8. Ensemble Yöntemlerini Ayarlama (Gradient Boosting'den en iyi şekilde yararlanma)
9. Final Modeli (tüm eğitim verileri kullanıldı ve doğrulama veri seti kullanarak doğrulama) gerçekleştirildi.

Bu çalışma, belirli makine öğrenimi görevleri için küçük adımların nasıl birleştirilerek tam bir projeye dönüştürülebileceğini göstermiştir. Bu sırayı takip etmek, Python ve scikit-learn kullanarak uygulamalı makine öğrenimi konusunda iyi bir pratik sağlamaktadır.

BÖLÜM 4

Giriş

Birçok gerçek dünya probleminde, verilerin ikili sınıflara ayrılması ve bu sınıfların doğru bir şekilde tahmin edilmesi büyük önem taşır. Makine öğrenmesi, ikili sınıflandırma problemlerini çözmek için güçlü bir araçtır ve bu bölümde sonar mines vs rock veri seti üzerinde makine öğrenmesiyle ikili sınıflandırma problemini ele alınacaktır. Bu veri seti, denizaltıların ve kayaların denizdeki yankılarını ölçen sesli sinyallerden oluşur. Amaç veri setindeki bu ses sinyallerini analiz ederek, bir nesnenin denizaltı mı yoksa kaya mı olduğunu doğru bir şekilde tahmin eden yapay zekâ modeli geliştirmektir.

4.1 Makine Öğrenmesiyle İkili Sınıflandırma Problemi

Bu ders boyunca, uygulamalı makine öğrenimi sürecinin her adımını içeren Python'da bir sınıflandırma tahmini modelleme problemi üzerinde çalışılacaktır. Bu projeyi tamamladıktan sonra;

- Bir sınıflandırma tahmini modelleme problemi nasıl baştan sona çözüleceğini,
- Model performansını artırmak için veri dönüştürmelerinin nasıl kullanılacağı,
- Model performansını artırmak için algoritma ayarlamanın nasıl kullanılacağı,
- Model performansını artırmak için ensemble yöntemleri ve ensemble yöntemlerinin ayarlanmasının nasıl kullanılacağı anlatılacaktır.

Bu projenin odak noktası, "Sonar Mines vs Rocks" veri kümesi olacaktır. Bu, su altı nesnelerinin sonar verilerinden metal veya kaya nesnelerinin tahmin edilmesi problemidir. Her veri örgüsü, 0.0 ile 1.0 aralığındaki 60 sayı kümesidir. Her sayı, belirli bir frekans bandındaki enerjiyi belirli bir süre boyunca entegre eder. Her kayıtla ilişkili etiket, nesnenin bir kaya ise R harfini ve bir maden ise M harfini içerir. Etiketlerdeki sayıların artan açı sırasına göre sıralı olduğu, ancak açıyı doğrudan kodlamadığı belirtilmelidir.

4.2 Veri setini Yükleme

Bu projeye gereken kütüphaneleri yükleyerek başlanmalıdır.

```
# Kütüphaneleri Yükleme
import numpy
from matplotlib import pyplot
from pandas import read_csv
from pandas import set_option
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
```

Liste 4.1 *Kütüphanelerin yüklenmesi*

Veri kümesini UCI Machine Learning deposu web sitesinden indirilebilir ve sonar.all-data.csv adıyla yerel çalışma dizininde kaydedilebilir (Liste 4.2)[15].

```
# Load dataset
data=pd.read_csv("sonar.all-data.csv")
```

Liste 4.2 *Veri setini yükleme*

Veri setinin yapısı gereği bu çalışmada özniteliklerin isimleri belirtilmemiştir. Bu, sınıf özniteliği (son sütun) dışında, değişkenlerin anlamlı isimleri olmamasından kaynaklanmaktadır. Ayrıca, başlık bilgisi olmadığı da kodlamada belirtilecektir. Bu, ilk satırın sütun adları olarak alınmasını önlemek için olacaktır. Veri kümesini yükledikten sonra sütun isimleri eklenebilecektir.

4.3 Veriyi Analiz Etme

Veri kümesi daha yakından incelenecektir.

4.3.1 Tanımlayıcı İstatistik

Veri kümesinin boyutlarını, yani satır ve sütun sayıları doğrulanarak başlanacaktır (Liste 4.3).

```
# Veri seti boyutu
print(data.shape)
```

Liste 4.3 *Veri setinin yüklenmesi*

Çalışma için 207 örneklem mevcuttur ve verinin sınıf özelliği dahil olmak üzere 61 özelliği doğrulanabilir (Liste 4.4).

```
(207, 61)
```

Liste 4.4 *Veri seti boyutları çıktısı*

Ayrıca, her özniteliğin veri türleri incelenmek istenirse Liste 4.5'teki kod listelenir.

```
# types
set_option("display.max_rows", 500)
print(veriseti.dtypes)
```

Liste 4.5 *Veri setindeki veri türleri*

Tüm verilerin float tipinde nümerik, sınıf değerinin bir nesne (object) olduğu Liste 4.6'da görülmektedir.

```
# types
0     float64
1     float64
2     float64
3     float64
4     float64
5     float64
6     float64
7     float64
8     float64
9     float64
10    float64
11    float64
12    float64
13    float64
14    float64
15    float64
16    float64
17    float64
18    float64
19    float64
20    float64
21    float64
22    float64
23    float64
24    float64
25    float64
26    float64
27    float64
28    float64
29    float64
30    float64
31    float64
32    float64
33    float64
34    float64
35    float64
36    float64
37    float64
38    float64
39    float64
40    float64
41    float64
42    float64
43    float64
44    float64
45    float64
46    float64
47    float64
48    float64
49    float64
50    float64
51    float64
52    float64
53    float64
54    float64
55    float64
56    float64
57    float64
58    float64
59    float64
60     object
dtype: object
```

Liste 4.6 *Veri setindeki öznitelik türlerinin çıktısı*

Şimdi verisetindeki ilk 20 veriyi incelenmektedir.

```
set_option("display.width", 100)
print(veriseti.head(20))
```

Liste 4.7 *Veri setindeki veri türleri*

Tüm öznitelikler değerlerinin ilk 20 değeri Liste 4.8'de veri değerleri örneği görülmektedir.

```
       0       1       2       3       4       5       6       7       8       9   ...      51  \
0  0.0200  0.0371  0.0428  0.0207  0.0954  0.0986  0.1539  0.1601  0.3109  0.2111  ...  0.0027
1  0.0453  0.0523  0.0843  0.0689  0.1183  0.2583  0.2156  0.3481  0.3337  0.2872  ...  0.0084
2  0.0262  0.0582  0.1099  0.1083  0.0974  0.2280  0.2431  0.3771  0.5598  0.6194  ...  0.0232
3  0.0100  0.0171  0.0623  0.0205  0.0205  0.0368  0.1098  0.1276  0.0598  0.1264  ...  0.0121
4  0.0762  0.0666  0.0481  0.0394  0.0590  0.0649  0.1209  0.2467  0.3564  0.4459  ...  0.0031
5  0.0286  0.0453  0.0277  0.0174  0.0384  0.0990  0.1201  0.1833  0.2105  0.3039  ...  0.0045
6  0.0317  0.0956  0.1321  0.1408  0.1674  0.1710  0.0731  0.1401  0.2083  0.3513  ...  0.0201
7  0.0519  0.0548  0.0842  0.0319  0.1158  0.0922  0.1027  0.0613  0.1465  0.2838  ...  0.0081
8  0.0223  0.0375  0.0484  0.0475  0.0647  0.0591  0.0753  0.0098  0.0684  0.1487  ...  0.0145
9  0.0164  0.0173  0.0347  0.0070  0.0187  0.0671  0.1056  0.0697  0.0962  0.0251  ...  0.0090
10 0.0039  0.0063  0.0152  0.0336  0.0310  0.0284  0.0396  0.0272  0.0323  0.0452  ...  0.0062
11 0.0123  0.0309  0.0169  0.0313  0.0358  0.0102  0.0182  0.0579  0.1122  0.0835  ...  0.0133
12 0.0079  0.0086  0.0055  0.0250  0.0344  0.0546  0.0528  0.0958  0.1009  0.1240  ...  0.0176
13 0.0090  0.0062  0.0253  0.0489  0.1197  0.1589  0.1392  0.0987  0.0955  0.1895  ...  0.0059
14 0.0124  0.0433  0.0604  0.0449  0.0597  0.0355  0.0531  0.0343  0.1052  0.2120  ...  0.0083
15 0.0298  0.0615  0.0650  0.0921  0.1615  0.2294  0.2176  0.2033  0.1459  0.0852  ...  0.0031
16 0.0352  0.0116  0.0191  0.0469  0.0737  0.1185  0.1683  0.1541  0.1466  0.2912  ...  0.0346
17 0.0192  0.0607  0.0378  0.0774  0.1388  0.0809  0.0568  0.0219  0.1037  0.1186  ...  0.0331
18 0.0270  0.0092  0.0145  0.0278  0.0412  0.0757  0.1026  0.1138  0.0794  0.1520  ...  0.0084
19 0.0126  0.0149  0.0641  0.1732  0.2565  0.2559  0.2947  0.4110  0.4983  0.5920  ...  0.0092

       52      53      54      55      56      57      58      59 60
0  0.0065  0.0159  0.0072  0.0167  0.0180  0.0084  0.0090  0.0032  R
1  0.0089  0.0048  0.0094  0.0191  0.0140  0.0049  0.0052  0.0044  R
2  0.0166  0.0095  0.0180  0.0244  0.0316  0.0164  0.0095  0.0078  R
3  0.0036  0.0150  0.0085  0.0073  0.0050  0.0044  0.0040  0.0117  R
4  0.0054  0.0105  0.0110  0.0015  0.0072  0.0048  0.0107  0.0094  R
5  0.0014  0.0038  0.0013  0.0089  0.0057  0.0027  0.0051  0.0062  R
6  0.0248  0.0131  0.0070  0.0138  0.0092  0.0143  0.0036  0.0103  R
7  0.0120  0.0045  0.0121  0.0097  0.0085  0.0047  0.0048  0.0053  R
8  0.0128  0.0145  0.0058  0.0049  0.0065  0.0093  0.0059  0.0022  R
9  0.0223  0.0179  0.0084  0.0068  0.0032  0.0035  0.0056  0.0040  R
10 0.0120  0.0052  0.0056  0.0093  0.0042  0.0003  0.0053  0.0036  R
11 0.0265  0.0224  0.0074  0.0118  0.0026  0.0092  0.0009  0.0044  R
12 0.0127  0.0088  0.0098  0.0019  0.0059  0.0058  0.0059  0.0032  R
13 0.0095  0.0194  0.0080  0.0152  0.0158  0.0053  0.0189  0.0102  R
14 0.0057  0.0174  0.0188  0.0054  0.0114  0.0196  0.0147  0.0062  R
15 0.0153  0.0071  0.0212  0.0076  0.0152  0.0049  0.0200  0.0073  R
16 0.0158  0.0154  0.0109  0.0048  0.0095  0.0015  0.0073  0.0067  R
17 0.0131  0.0120  0.0108  0.0024  0.0045  0.0037  0.0112  0.0075  R
18 0.0010  0.0018  0.0068  0.0039  0.0120  0.0132  0.0070  0.0088  R
19 0.0035  0.0098  0.0121  0.0006  0.0181  0.0094  0.0116  0.0063  R

[20 rows x 61 columns]
```

Liste 4.8 *ilk 20 öznitelik değerinin çıktısı*

Her özniteliği Liste 4.9'da kod ile temel istatistiksel özet alınmıştır.

```
# descriptions, change precision to 3 places
set_option("precision", 3)
print(veriseti.describe())
```

Liste 4.9 *Veri setindeki özniteliklerin istatistiksel tanımı*

Liste 4.10 incelendiğinde veri aynı aralıkta olmasına rağmen, ilginç olarak farklı ortalama değerlere sahiptir. Veriyi standardize etmek algoritma performanslarını artırabilir.

```
              0         1         2         3         4         5         6         7         8         9  \
count   208.000  2.080e+02   208.000   208.000   208.000   208.000   208.000   208.000   208.000   208.000
mean      0.029  3.844e-02     0.044     0.054     0.075     0.105     0.122     0.135     0.178     0.208
std       0.023  3.296e-02     0.038     0.047     0.056     0.059     0.062     0.085     0.118     0.134
min       0.002  6.000e-04     0.002     0.006     0.007     0.010     0.003     0.005     0.007     0.011
25%       0.013  1.645e-02     0.019     0.024     0.038     0.067     0.081     0.080     0.097     0.111
50%       0.023  3.080e-02     0.034     0.044     0.062     0.092     0.107     0.112     0.152     0.182
75%       0.036  4.795e-02     0.058     0.065     0.100     0.134     0.154     0.170     0.233     0.269
max       0.137  2.339e-01     0.306     0.426     0.401     0.382     0.373     0.459     0.683     0.711

          ...        50         51         52        53         54         55         56         57  \
count     ...   208.000  2.080e+02  2.080e+02   208.000  2.080e+02  2.080e+02  2.080e+02  2.080e+02
mean      ...     0.016  1.342e-02  1.071e-02     0.011  9.290e-03  8.222e-03  7.820e-03  7.949e-03
std       ...     0.012  9.634e-03  7.060e-03     0.007  7.088e-03  5.736e-03  5.785e-03  6.470e-03
min       ...     0.000  8.000e-04  5.000e-04     0.001  6.000e-04  4.000e-04  3.000e-04  3.000e-04
25%       ...     0.008  7.275e-03  5.075e-03     0.005  4.150e-03  4.400e-03  3.700e-03  3.600e-03
50%       ...     0.014  1.140e-02  9.550e-03     0.009  7.500e-03  6.850e-03  5.950e-03  5.800e-03
75%       ...     0.021  1.673e-02  1.490e-02     0.015  1.210e-02  1.058e-02  1.043e-02  1.035e-02
max       ...     0.100  7.090e-02  3.900e-02     0.035  4.470e-02  3.940e-02  3.550e-02  4.400e-02

             58         59
count  2.080e+02  2.080e+02
mean   7.941e-03  6.507e-03
std    6.181e-03  5.031e-03
min    1.000e-04  6.000e-04
25%    3.675e-03  3.100e-03
50%    6.400e-03  5.300e-03
75%    1.033e-02  8.525e-03
max    3.640e-02  4.390e-02

[8 rows x 60 columns]
```

Liste 4.10 *Veri setindeki özniteliklerin istatistiksel tanım çıktısı*

Veri setinin sınıf dağılımı Liste 4.11 ile incelendiğinde, Liste 4.12'deki sonuçlar elde edilmiştir.

```
# sınıf dağılımı
print(veriseti.groupby(60).size())
```

Liste 4.11 *Veri setindeki sınıf dağılımı*

Grup dağılım boyutu görülüyor ki madenler M(Mines/madenler) ve kayalar R(Rocks/kayalar) sıfırları düzenli dağıtılmıştır.

```
# sınıf dağılımı
60
M    111
R     97
dtype: int64
```

Liste 4.12 *Veri setindeki sınıf dağılımı çıktısı*

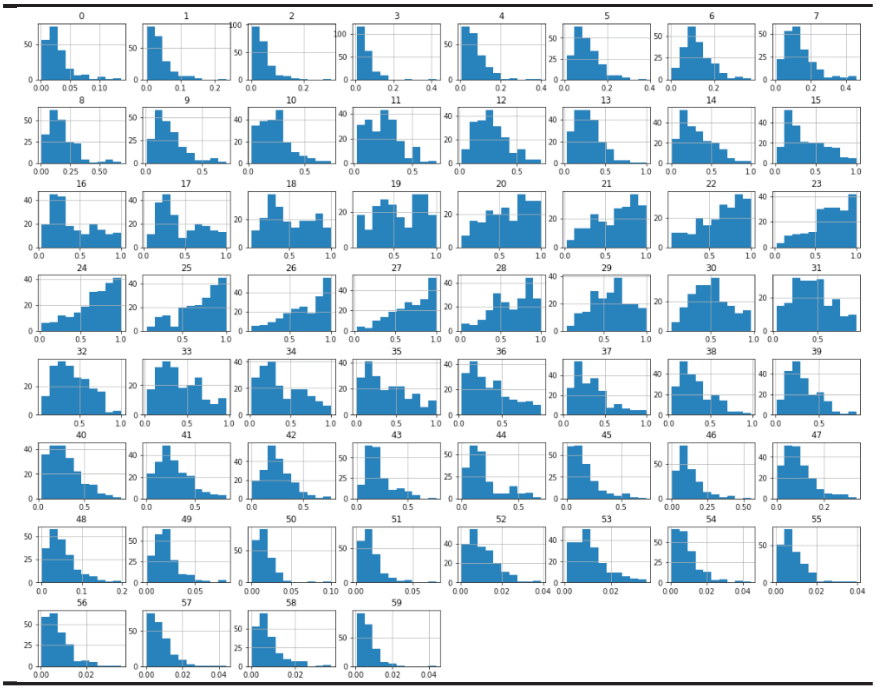
4.3.2 Tek Modlu Veri Görselleştirme

Tek tek özniteliklerin grafikleri incelemek faydalı olabilmektedir. Verilerden fikir üretmek için veriyi çeşitli görselleştirme yöntemleriyle bakmak çoğu zaman çok kullanışlıdır (Liste 4.13). Veri dağılımları hakkında bir fikir edinmek için her öznitelik histogramları incelenebilir (Şekil 4.1).

```
# Histogram grafiği olarak çizdirilir
veriseti.hist(figsize=(20,16))
# Alt ve üst, sol ve sağ grafikler arasındaki boşluğu ayarla
pyplot.subplots_adjust(hspace=0.5)
# Sol ve sağ grafikler arasındaki boşluğu ayarla
pyplot.subplots_adjust(wspace=0.2)
# Grafiği göster
pyplot.show()
```

Liste 4.13 *Veri setindeki her bir özniteliğin histogramı*

Özniteliklerin histogramlarında birçok Gauss dağılımına ve bazı özniteliklerin dağılımında ise bazı üstele benzer dağılımlar gözlemlenebilir.



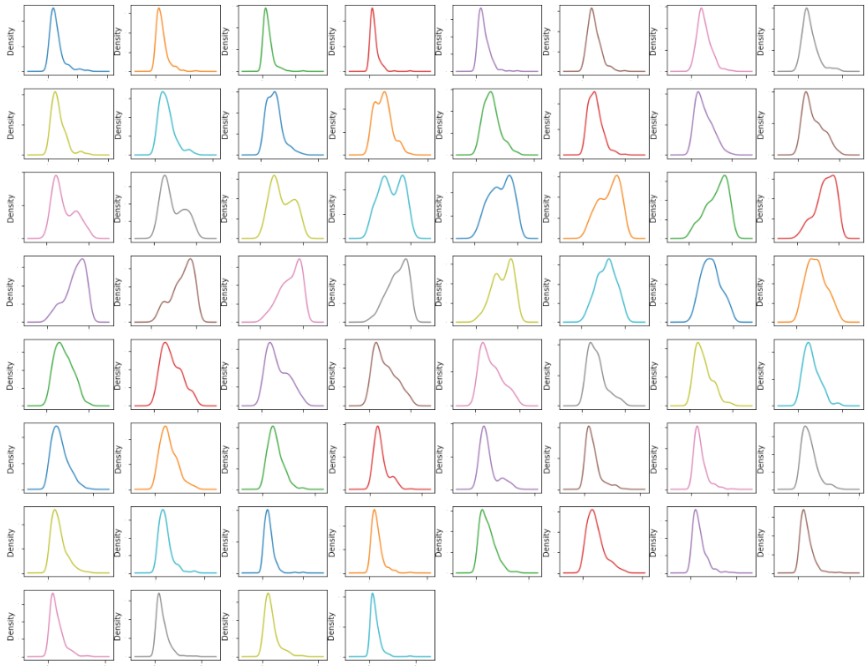
Şekil 4.1 *Veri setindeki her bir özniteliğin histogram çıktısı*

Aynı bakış açısıyla devam edilirse özniteliklerin yoğunluk dağılımları (Liste 4.14) ile incelendiğinde Şekil 4.2'deki öznitelik yoğunluk grafiği elde edilir.

```
#density
veriseti.plot(kind='density', subplots=True, layout=(8,8),
              sharex=False, legend=False,
              fontsize=1,figsize=(20,16))
pyplot.show()
```

Liste 4.14 *Veri setindeki her bir özniteliğin yoğunluk grafiği*

Bu yaklaşım faydalıdır, özniteliklerin birçoğunun eğik dağılımı olduğunu görülebilmektedir. Dağılımların eğikliğini düzeltmek için Box-Cox gibi bir güç dönüşüm kullanışlı olabilir.

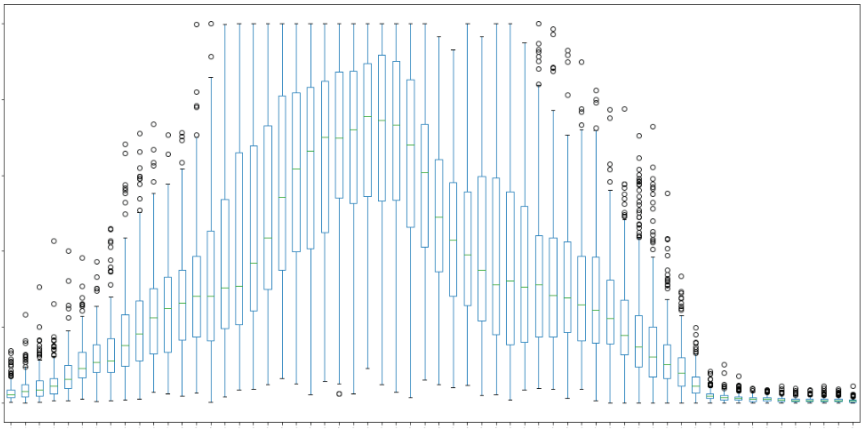


Şekil 4.2 *Veri setindeki her bir özniteliğin yoğunluk grafik çıktısı*

Numerik özniteliklerin kutu grafiklerini incelemek, değerlerin yayılımı hakkında bir fikir edinmek için başka bir bakış açısı kazandıracaktır (Şekil 4.3).

```
#kutu grafiği
veriseti.plot(kind='box', subplots=False,
layout=(8,8),sharex=False, legend=False,fontsize=1,figsize=(20,10))
pyplot.show()
```

Liste 4.15 *Veri setindeki her bir özniteliğin kutu grafiğinin çizdirilmesi*



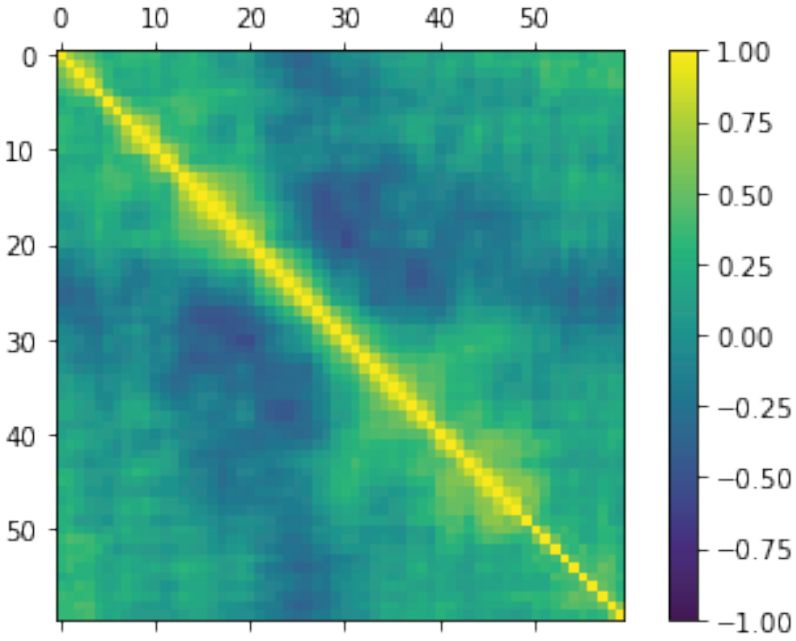
Şekil 4.3 *Her bir özniteliğin kutu grafik çıktısı*

4.3.3 Çok Modlu Veri Görselleştirme

Öznitelikler arasındaki ilişki, korelasyon matrisi çiziminde kullanılarak çoklu görselleştirmeye başlanabilir.

```
# correlation matrix
fig = pyplot.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(veriseti.corr(), vmin=-1, vmax=1,
interpolation='none')
fig.colorbar(cax)
ticks = numpy.arange(0,4,1)
pyplot.show()
```

Liste 4.16 *Veri setindeki öznitelikler arasındaki ilişkinin görselleştirilmesi*



Şekil 4.4 *Veri setindeki öznitelikler arasındaki ilişkinin görselleştirilmesi*

Öznitelikler arasında bir ilişkiler grubunun oluşturduğu bir örüntü görülüyor (Şekil 4.4). Köşegen etrafındaki sayı bölge, birbirlerine yakın özniteliklerin birbirleriyle genellikle daha ilişkili olduğunu göstermektedir. Lacivert zirveler, sıralamada birbirlerinden uzak olan öznitelikler arasında orta düzeyde negatif bir korelasyon olduğunu da gösterir. Bu durum, öznitelikler sırasının deniz radarı cırıldaması sensör açısına karşılık geldiğinde anlam kazanır.

4.4 Doğrulama Veri Seti

Bir yapay zekâ modelini test etmek için veri setinden bir kısım veri ayırmak iyi bir yaklaşımdır. Bu, analiz ve modellemeye katılmayan ve-

rilerdir. Proje sonunda nihai modelin doğruluğunu ortaya koymak için kullanılacaktır. Böylece eğitim esnasındaki verilerden farklı verilerdeki model başarısı daha güvenilir bir şekilde görülebilecektir. Modelleme için veri setinin %80'i kullanılacak ve %20'sini doğrulama için ayrılacaktır (Liste 4.17).

```
# Verisetini test ve eğitim verisi olarak ikiye bölme
dizi = veriseti.values
X = dizi[:,0:60]
Y = dizi[:,60]
dogrulama_boyutu = 0.20
tohum = 7
X_egitim, X_dogrulama, Y_egitim, Y_dogrulama = train_test_split(X,
Y, test_size=0.33, random_state=tohum)
```

Liste 4.17 *Veri setini eğitim ve test için ayrıma*

4.5 Algoritmaları Değerlendirme

Bu veri setinde hangi algoritmaların başarılı olacağı öngörülememektedir. İç güdüsel olarak, K-En Yakın Komşu ve Destek Vektör Makine gibi mesafe tabanlı algoritmaların iyi çalışabileceğini düşünebiliriz. Test çatımızı tasarlamaya başlayalım. 10-katlı çapraz doğrulama kullanılacaktır (Liste 4.18). Veri seti çok küçük değil ve bu iyi bir standart test çatısı yapılandırmasıdır. Algoritmalar, doğruluk metriği kullanarak değerlendirecektir. Bu, verilen modelin ne kadar doğru olduğu hakkında hızlı bir fikir veren aşırı bir metriktir. Bu gibi ikili sınıflandırma problemlerinde daha yararlıdır.

```
# Test seçenekleri ve doğrulama metrikleri
k_fold_sayisi = 10
tohum = 7
skor = "accuracy"
```

Liste 4.18 *Veri setini eğitim ve test için ayrıma*

Bu problem üzerinde performansın bir temelini oluşturalım ve birkaç farklı algoritmayı denetim noktası olarak inceleyelim. Bu sınıflandırma problemi üzerinde çalışabilecek çeşitli algoritmalar seçilecektir. Seçilen altı algoritma şunları içerir:

- Lineer Algoritmalar: Lojistik Regresyon (LR) ve Lineer Ayırıcı Analiz (LDA).
- Non-Lineer Algoritmalar: Sınıflandırma ve Regresyon Ağaçları (CART), Destek Vektör Makine (SVM), Gauss Naive Bayessian (NB) ve K-En Yakın Komşular (KNN).

```
modeller = []
modeller.append(('LR', LogisticRegression()))
modeller.append(('LDA', LinearDiscriminantAnalysis()))
modeller.append(('KNN', KNeighborsClassifier()))
modeller.append(('CART', DecisionTreeClassifier()))
modeller.append(('NB', GaussianNB()))
modeller.append(('SVM', SVC()))
```

Liste 4.19 *Algoritmaları Hazırlama ve Değerlendirme*

Tüm algoritmalar varsayılan parametreleri kullanır (Liste 4.19). Algoritmalar karşılaştırılarak her algoritma için doğruluk ortalaması ve standart sapması hesaplanarak sonraki kullanım için sonuçlar toplanacaktır(Liste 4.20).

```
sonuclar = []
isimler = []
for isim, model in modeller:
    kfold = KFold(n_splits=k_fold_sayisi, shuffle=True, random_
state=tohum)
    cv_sonuclar = cross_val_score(model, X_egitim, Y_egitim,
cv=kfold, scoring=skor)
    sonuclar.append(cv_sonuclar)
    isimler.append(isim)
    msj = "%s: %f (%f)" % (isim, cv_sonuclar.mean(), cv_sonuclar.
std())
    print(msj)
```

Liste 4.20 *Algoritmaları Hazırlama ve Değerlendirme*

Liste 4.20'deki kod çalıştırıldığında Liste 4.21'deki çıktı elde edilecektir. Sonuçlar, Lojistik Regresyon (LR), Lineer Diskriminant Analizi (LDA), Vektör Destek Makineleri (SVM) ve K-En Yakın Komşularının daha ayrıntılı incelemeye değer olabileceği önerisinde bulunur.

```
LR: 0.756044 (0.105923)
LDA: 0.735714 (0.115396)
KNN: 0.734066 (0.131368)
CART: 0.662088 (0.142418)
NB: 0.641209 (0.093473)
SVM: 0.784066 (0.084660)
```

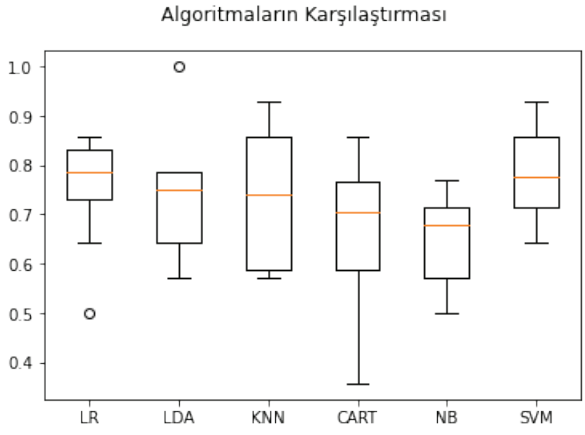
Liste 4.21 *Algoritmaları Hazırlama ve Değerlendirme*

Sadece ortalama doğruluk değerlerinden oluşan Liste 4.21'deki algoritmaların daha ayrıntılı incelenmesi gerekmektedir. Kutu grafiği ile karşılaştırılarak ayrıntılı bir inceleme Liste 4.22'deki kod ile gerçekleştirilebilir.

```
#Algoritmaları Karşılaştır
figur = pyplot.figure()
#Başlık eklenir.
figur.suptitle("Algoritmaların Karşılaştırması")
#Grafiğin ekseni belirlenir.
eksen = figur.add_subplot(111)
#Kutu grafiği çizilir ve adlar eklenir.
pyplot.boxplot(sonuclar)
eksen.set_xticklabels(isimler)
#Grafiği göster.
pyplot.show()
```

Liste 4.22 *Algoritma hazırlama ve değerlendirme*

Sonuçlarda, LR için basık bir dağılım oluşturduğu görülmektedir. Bu da düşük bir varyansı ifade etmektedir. Gaussian (NB) dağılımının basık bir dağılım gösterdiği halde düşük doğruluk vermesi ise şaşırtıcıdır.



Şekil 4.5 *Algoritmaların kutu grafiği çıktısı ile karşılaştırılması*

Öznitelerin farklı dağılımının, SVM (Destek Vektör Makineleri) gibi algoritmaların doğruluğu üzerinde etkili olabileceği mümkündür. Bir sonraki bölümde, eğitim veri setinin standartlaştırılmış bir kopyasıyla bu denetim noktası tekrar edilecektir.

4.6 Algoritmaların Standardize Veri Seti ile Değerlendirilmesi

Algoritmaların matematiksel altyapıya dayanan davranışları gereği, verilerin farklı dağılımları bazı modellerin başarısını olumsuz etkileyebilmektedir. Şimdi bu olumsuz etkiyi ortadan kaldırmak için bir veri seti kopyasını standartlaştıralım. Bu tüm verilerin ortalamasının sıfıra çekilmesi varyansınız da 1'e getirilmesi anlamına gelmektedir. Ayrıca veri dönüşümü esnasındaki veri sızıntısının da farkında olunmalıdır. Bu sızıntını engellemenin en iyi yolunun çapraz doğrulamalı bir boru hattı kullanmak

olduğundan bahsedilmişti. Böylece her bir model için adil bir doğrulama ve model eğitim ortamı oluşturulmuş olacak ve görülmemiş ve üzerinde yüksek doğrulukta ve güvenilirlikte bir model eğitilmiş olacaktır (Liste 4.23).

```
# Veri kümesini standartlaştırma
borular = []
borular.append(('Ölçekli.LR', Pipeline([('Ölçeklyci',
StandardScaler()),('LR', LogisticRegression())])))
borular.append(('Ölçekli.LDA', Pipeline([('Ölçeklyci',
StandardScaler()),('LDA', LinearDiscriminantAnalysis())])))
borular.append(('Ölçekli.NB', Pipeline([('Ölçeklyci',
StandardScaler()),('NB', GaussianNB())])))
borular.append(('Ölçekli.KNN', Pipeline([('Ölçeklyci',
StandardScaler()),('KNN', KNeighborsClassifier())])))
borular.append(('Ölçekli.CART', Pipeline([('Ölçeklyci',
StandardScaler()),('CART', DecisionTreeClassifier())])))
borular.append(('Ölçekli.SVM', Pipeline([('Ölçeklyci',
StandardScaler()),('SVM', SVC())])))
sonuclar = []
isimler = []
# Modelleri ölçeklendirilmiş veri ile eğiterek sonuçları
değerlendir
for isim, model in borular:
    k_fold = KFold(n_splits=k_fold_sayisi, shuffle=True, random_
state=tohum)
    cv_sonuclar = cross_val_score(model, X_egitim, Y_egitim, cv=k_
fold, scoring=skor)
    sonuclar.append(cv_sonuclar)
    isimler.append(isim)
    msj = "%s: %f (%f)" % (isim, cv_sonuclar.mean(), cv_sonuclar.
std())
    print(msj)
```

Liste 4.23 *Algoritmaların standartlaştırılmış hali ile değerlendirilmesi*

Örnek çalıştırıldığında aşağıdaki sonuçlar listelenir. KNN'nin önceki ayarlarından daha başarılı olduğu görülebilir. Ayrıca verinin standartlaştırılmasının, SVM'yi test edilen en doğru algoritma haline getirdiği de görülebilir (Liste 4.24).

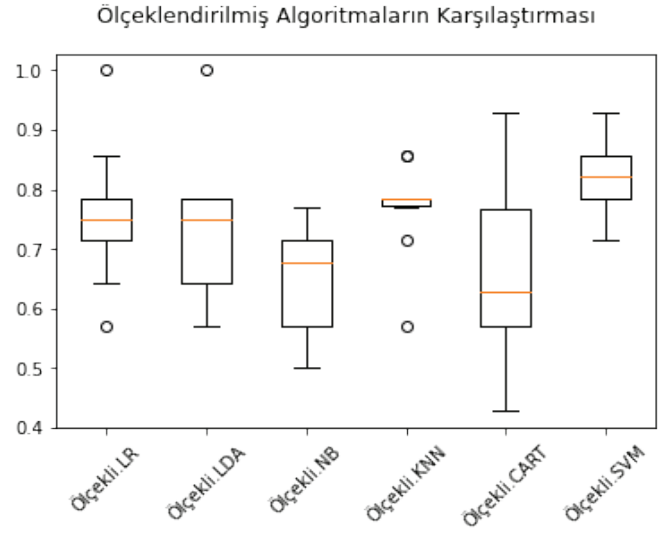
```
Ölçekli.LR: 0.757143 (0.111575)
Ölçekli.LDA: 0.735714 (0.115396)
Ölçekli.NB: 0.641209 (0.093473)
Ölçekli.KNN: 0.769780 (0.076784)
Ölçekli.CART: 0.624725 (0.149488)
Ölçekli.SVM: 0.820879 (0.072402)
```

Liste 4.24 *Algoritmaların standartlaştırılmış hali ile değerlendirilmesi*

```
# Algoritmaları karşılaştıralım
fig = pyplot.figure()
fig.suptitle('Ölçeklendirilmiş Algoritmaların Karşılaştırması')
ax = fig.add_subplot(111)
pyplot.boxplot(sonuclar)
ax.set_xticklabels(isimler, rotation=45)
pyplot.show()
```

Liste 4.25 *Algoritmaların standartlaştırılmış veri seti ile doğruluk karşılaştırması*

Sonuçlar, SVM ve KNN algoritmalarını daha da ayrıntılı incelemeyi önerir. Muhtemelen varsayılanın ötesinde bir yapılandırma daha doğru modellerin eğitilmesine imkân verebilir. Yine, doğruluk değerlerinin dağılımını kutu grafikleri kullanarak çizildiğinde (Şekil 4.6).



Şekil 4.6 *Standardize veride algortimaların kutu grafiği ile karşılaştırması*

4.7 Başarılı Algoritmalarda Parametre Ayarlama

Varsayılan parametrelerde yüksek başarının elde edildiği KNN ve SVM algoritmalarının parametrelerinin ayarlanarak daha başarılı modeller elde edilip edilemeyeceği incelenecektir.

4.7.1 KNN'i Ayarlama

KNN için komşu sayısı ayarlanarak başlanabilir. Varsayılan komşu sayısı 7'dir. Liste 4.26'da 7'nin varsayılan değerini kapsayan 1 ile 21 arasındaki tek değerleri deneyelim. Her k değeri, eğitim standartlaştırılmış veri setinde 10-katlı çapraz doğrulama kullanılarak değerlendirilir.

```
#Parametreleri değiştirilmiş KNN
scaler = StandardScaler().fit(X_egitim)
rescaledX = scaler.transform(X_egitim)
k_degerleri = numpy.array([1,3,5,7,9,11,13,15,17,19,21])
param_grid = dict(n_neighbors=k_degerleri)
model = KNeighborsClassifier()
kfold = KFold(n_splits=k_fold_sayisi,shuffle=True, random_
state=tohum)
grid = GridSearchCV(estimator=model, param_grid=param_grid,
scoring=skor, cv=kfold)
grid_result = grid.fit(rescaledX, Y_egitim)
print("En İyi Sonuç: %f, En İyi Parametreler: %s" % (grid_result.
best_score_, grid_result.best_params_))
ortalama = grid_result.cv_results_["mean_test_score"]
standart_sapma = grid_result.cv_results_["std_test_score"]
parametreler = grid_result.cv_results_["params"]
for ortalama, ss, parametre in zip(ortalama, standart_sapma,
parametreler):
    print("%f (%f) için parametreler: %r" % (ortalama, ss,
parametre))
```

Liste 4.26 *Algoritmaların standartlaştırılmış veri seti ile KNN ayarlama*

En yüksek doğruluğu veren yapılandırmayı, denenen tüm değerlerin doğruluğu da yazdırılabilir. Liste 4.26'daki kodlar çalıştırıldığında Liste 4.27'deki sonuçları elde edilir.

```
En İyi Sonuç: 0.813736, En İyi Parametreler: {'n_neighbors': 1}
0.813736 (0.056057) için parametreler: {'n_neighbors': 1}
0.785165 (0.077254) için parametreler: {'n_neighbors': 3}
0.769780 (0.076784) için parametreler: {'n_neighbors': 5}
0.732967 (0.104083) için parametreler: {'n_neighbors': 7}
0.718132 (0.113007) için parametreler: {'n_neighbors': 9}
0.696703 (0.141877) için parametreler: {'n_neighbors': 11}
0.696703 (0.122585) için parametreler: {'n_neighbors': 13}
0.682418 (0.127912) için parametreler: {'n_neighbors': 15}
0.732418 (0.143596) için parametreler: {'n_neighbors': 17}
0.732967 (0.122126) için parametreler: {'n_neighbors': 19}
0.726374 (0.105483) için parametreler: {'n_neighbors': 21}
```

Liste 4.27 *Standartlaştırılmış veri seti ile KNN farkı değer sonuçları*

En başarılı yapılandırmanın k=1 olduğunu görebiliriz. Bu, algoritmanın yalnızca eğitim veri setindeki en benzer örnekleri kullanarak tahminler yapacağı için ilginçtir.

```
# Tune scaled SVM
scaler = StandardScaler().fit(X_egitim)
rescaledX = scaler.transform(X_egitim)
c_degerleri = [0.1, 0.3, 0.5, 0.7, 0.9, 1.0, 1.3, 1.5, 1.7, 2.0]
kernel_degerleri = ["linear", "poly", "rbf", "sigmoid"]
param_grid = dict(C=c_degerleri, kernel=kernel_degerleri)
model = SVC()
kfold = KFold(n_splits=k_fold_sayisi,shuffle=True, random_
state=tohum)
grid = GridSearchCV(estimator=model, param_grid=param_grid,
scoring=skor, cv=kfold)
grid_result = grid.fit(rescaledX, Y_egitim)
print("En İyi Sonuç: %f, En İyi Parametreler: %s" % (grid_result.
best_score_, grid_result.best_params_))
ortalama = grid_result.cv_results_["mean_test_score"]
standart_sapma = grid_result.cv_results_["std_test_score"]
parametreler = grid_result.cv_results_["params"]
for ortalama, ss, parametre in zip(ortalama, standart_sapma,
parametreler):
    print("%f (%f) için parametreler: %r" % (ortalama, ss,
parametre))
```

Liste 4.28 *Standartlaştırılmış veri seti ile SVM Ayarlama*

Liste 4.28'deki kod çalıştırıldığında en iyi yapılandırmayı, doğruluğu ve tüm yapılandırma kombinasyonlarının doğruluğu yazdırılır (Liste 4.29).

```
En İyi Sonuç: 0.842308, En İyi Parametreler: {'C': 1.7, 'kernel': 'rbf'}
0.807143 (0.101267) için parametreler: {'C': 0.1, 'kernel': 'linear'}
0.502747 (0.136641) için parametreler: {'C': 0.1, 'kernel': 'poly'}
0.488462 (0.101892) için parametreler: {'C': 0.1, 'kernel': 'rbf'}
0.676374 (0.115788) için parametreler: {'C': 0.1, 'kernel': 'sigmoid'}
0.735714 (0.119736) için parametreler: {'C': 0.3, 'kernel': 'linear'}
0.633516 (0.101791) için parametreler: {'C': 0.3, 'kernel': 'poly'}
0.712088 (0.084772) için parametreler: {'C': 0.3, 'kernel': 'rbf'}
0.747802 (0.112696) için parametreler: {'C': 0.3, 'kernel': 'sigmoid'}
0.742857 (0.102020) için parametreler: {'C': 0.5, 'kernel': 'linear'}
0.712088 (0.096058) için parametreler: {'C': 0.5, 'kernel': 'poly'}
0.734066 (0.089855) için parametreler: {'C': 0.5, 'kernel': 'rbf'}
0.763187 (0.070426) için parametreler: {'C': 0.5, 'kernel': 'sigmoid'}
0.728571 (0.114286) için parametreler: {'C': 0.7, 'kernel': 'linear'}
0.776923 (0.087195) için parametreler: {'C': 0.7, 'kernel': 'poly'}
0.776923 (0.092862) için parametreler: {'C': 0.7, 'kernel': 'rbf'}
0.748901 (0.071642) için parametreler: {'C': 0.7, 'kernel': 'sigmoid'}
0.714286 (0.115175) için parametreler: {'C': 0.9, 'kernel': 'linear'}
0.776923 (0.087195) için parametreler: {'C': 0.9, 'kernel': 'poly'}
0.806044 (0.071092) için parametreler: {'C': 0.9, 'kernel': 'rbf'}
0.734615 (0.076925) için parametreler: {'C': 0.9, 'kernel': 'sigmoid'}
0.721429 (0.112712) için parametreler: {'C': 1.0, 'kernel': 'linear'}
0.784066 (0.090486) için parametreler: {'C': 1.0, 'kernel': 'poly'}
0.820879 (0.072402) için parametreler: {'C': 1.0, 'kernel': 'rbf'}
0.741758 (0.063603) için parametreler: {'C': 1.0, 'kernel': 'sigmoid'}
0.720879 (0.091638) için parametreler: {'C': 1.3, 'kernel': 'linear'}
0.776374 (0.094596) için parametreler: {'C': 1.3, 'kernel': 'poly'}
0.820879 (0.072402) için parametreler: {'C': 1.3, 'kernel': 'rbf'}
0.741209 (0.096286) için parametreler: {'C': 1.3, 'kernel': 'sigmoid'}
0.720879 (0.091638) için parametreler: {'C': 1.5, 'kernel': 'linear'}
0.769231 (0.090851) için parametreler: {'C': 1.5, 'kernel': 'poly'}
0.813736 (0.084995) için parametreler: {'C': 1.5, 'kernel': 'rbf'}
0.770330 (0.088214) için parametreler: {'C': 1.5, 'kernel': 'sigmoid'}
0.713736 (0.099856) için parametreler: {'C': 1.7, 'kernel': 'linear'}
0.762088 (0.086355) için parametreler: {'C': 1.7, 'kernel': 'poly'}
0.842308 (0.094277) için parametreler: {'C': 1.7, 'kernel': 'rbf'}
0.763187 (0.083670) için parametreler: {'C': 1.7, 'kernel': 'sigmoid'}
0.713736 (0.099856) için parametreler: {'C': 2.0, 'kernel': 'linear'}
0.776374 (0.083112) için parametreler: {'C': 2.0, 'kernel': 'poly'}
0.835165 (0.095579) için parametreler: {'C': 2.0, 'kernel': 'rbf'}
0.748901 (0.078441) için parametreler: {'C': 2.0, 'kernel': 'sigmoid'}
```

Liste 4.29 *Standartlaştırılmış veri seti ile SVM ayarlanmış sonucu*

En doğru yapılandırmanın, C değeri 1.7 olan ve RBF çekirdeğine sahip bir SVM olduğunu görebiliriz. Doğruluk oranı %0.842308, KNN'nin elde edebileceğinden daha iyi görülmektedir.

4.8 Birleştirilmiş (Ensemble) Yöntemler

Algoritmaların performansını iyileştirmeyi amaçlayan başka bir yöntem de ensemble yöntemleridir. Bu bölümde, iki boosting ve iki bagging yöntemine sahip dört farklı ensemble makine öğrenme algoritmasını değerlendireceğiz.

4.9 Boosting Yöntemleri: AdaBoost (AB) ve Gradient Boosting (GBM)

Bagging Yöntemleri: Rastgele Ormanlar (Random Forest /RF) ve **Extra Trees (ET)**.

Bu defa öncekiyle aynı test arabirimi olan 10-katlı çapraz doğrulama kullanılacaktır (Liste 4.30). Bu durumda, veride standartlaştırma kullanılmaz, çünkü dört ensemble algoritmasının hepsi veri dağılımlarına duyarlı olmayan karar ağaçlarına dayanır.

```
# Topluluk Yöntemleri
ensembles = []
ensembles.append(('AB', AdaBoostClassifier()))
ensembles.append(('GBM', GradientBoostingClassifier()))
ensembles.append(('RF', RandomForestClassifier()))
ensembles.append(('ET', ExtraTreesClassifier()))
skor = "accuracy"
sonuclar = []
isimler = []
for isim, model in ensembles:
    kfold = KFold(n_splits=k_fold_sayisi, shuffle=True, random_
state=tohum)
    cv_sonuclar = cross_val_score(model, X, Y, cv=kfold,
scoring=skor)
    sonuclar.append(cv_sonuclar)
    isimler.append(isim)
    mesaj = "%s: %f (%f)" % (isim, cv_sonuclar.mean(), cv_
sonuclar.std())
    print(mesaj)
```

Liste 4.30 *Birleştirilmiş Algoritmaları Uygulama*

Uygulanan algoritmalar sonucunda Liste 4.31'deki sonuçlar alınmıştır.

```
AB: 0.821667 (0.075836)
GBM: 0.855000 (0.105099)
RF: 0.797381 (0.085757)
ET: 0.864524 (0.070102)
```

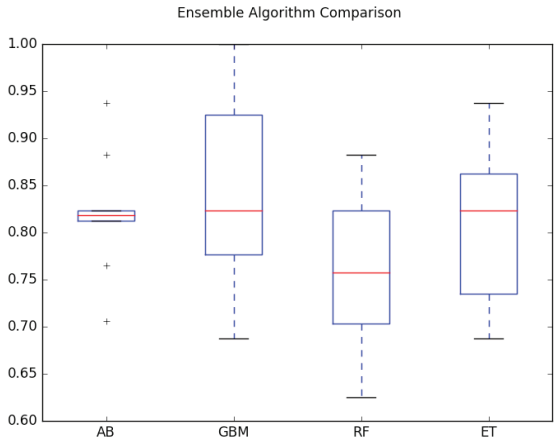
Liste 4.31 *Birleştirilmiş Algoritmaların Uygulama Sonucu*

Boosting tekniklerinin her ikisi de varsayılan yapılandırmalar ile yüksek doğruluk değerleri sağlamaktadır. Bu doğruluk değerlerinin dağılımı çapraz doğrulama katmanları üzerinde çizdirilebilir (Liste 4.32).

```
# Algoritmaları Karşılaştırılması
fig = pyplot.figure()
fig.suptitle('Ensemble Algoritmalarının Karşılaştırması')
ax = fig.add_subplot(111)
pyplot.boxplot(sonuclar)
ax.set_xticklabels(isimler)
pyplot.show()
```

Liste 4.32 *Standartlaştırılmış veri seti ile SVM ayarlanmış sonucu*

Sonuçlar, doğruluk oranının yüksek, 0.90'lara doğru eğimli yayılımı olan ve ayrıca güçlü ortalama bir değer sunan GBM'nin daha ayrıntılı bir incelemeye değer olduğunu göstermektedir (Şekil 4.7).



Şekil 4.7 *Ensemble algoritmalarının kutu grafikleri*

4.9 Modelin Sonlandırılması

SVM, bu problem için düşük karmaşıklıkta ve istikrarlı bir model olarak en umut verici sonuçları vermiştir. Bu bölümde, model tüm eğitim veri seti üzerinde eğitilecektir ve doğrulama veriseti için tahminler yaparak modelin başarısı doğrulanacaktır. Bulguların bir parçası, SVM'nin verisetinin standartlaştırılmış olduğu durumda daha iyi performans gösterdiği görülmüştü. Bu, tüm eğitim veri setinden hesaplanabilir ve aynı dönüşüm doğrulama veri setinden girdi özniteliklerine uygulanabilir. Böylece tüm özniteliklerin ortalama değeri 0 ve standart sapma değeri 1 olur (Liste 4.33).

```
# Prapare the model
scaler = StandardScaler().fit(X_egitim)
rescaledX = scaler.transform(X_egitim)
model = SVC(C=1.5)
model.fit(rescaledX, Y_egitim)
# estimate accuracy on validation dataset
rescaledValidationX = scaler.transform(X_dogrulama)
predictions = model.predict(rescaledValidationX)
print(accuracy_score(Y_dogrulama, predictions))
print(confusion_matrix(Y_dogrulama, predictions))
print(classification_report(Y_dogrulama, predictions))
```

Liste 4.33 *Standartlaştırılmış veriseti ile parametre ayarlanmış SVM*

Saklı tutulan doğrulama veri setinde yaklaşık yüzde 86 oranında bir doğruluk elde edildiği görülebilir. Bu, SVM'nin ayarlanması sırasında yukarıda tahmin edilen değerlere yakın bir skordur.

```
0.9130434782608695
[[36  2]
 [ 4 27]]
              precision    recall  f1-score   support

           M       0.90      0.95      0.92        38
           R       0.93      0.87      0.90        31

    accuracy                           0.91        69
   macro avg       0.92      0.91      0.91        69
weighted avg       0.91      0.91      0.91        69
```

Liste 4.34 *Standartlaştırılmış veri seti ile parametre ayarlanmış SVM sonucu*

4.10 Özet

Bu bölümde, Python kullanarak makine öğrenimi modelleme uygulaması gerçekleştirmek için gereken tüm adımlar gerçekleştirilmiştir. Bu adımları kısaca hatırlanırsa:

1. Problem tanımının yapılması
2. Veri setinin Python ortamına yüklenmesi
3. Verinin analizi (aynı ölçek ancak farklı veri dağılımları).
4. Algoritmaların değerlendirilmesi (KNN ilk incelemede başarılı görüldü).
5. Standartlaştırma ile algoritmaların değerlendirilmesi (KNN ve SVM başarılı olarak gözlemlendi).
6. Algoritma Ayarlaması (KNN için K=1 başarılı, SVM için rbf çekirdek ve c=1.5 en başarılı).
7. Ensemble Yöntemleri (Bagging ve Boosting, SVM'nin kadar iyi değil).
8. Modelin Sonlandırılması (tüm eğitim verisini kullan ve doğrulama veri setiyle doğrulama).

Bu çalışma sayesinde, makine öğrenimi görevlerinin aşama aşama yerine getirilerek tam bir projeye nasıl dönüştürülebileceği gösterildi. Bu çalışma sayesinde, Python kullanarak uygulamalı makine öğrenimi alanında tam bir uygulama örneği gerçekleştirilmiştir.

4.11 Sonraki Adım

Bu, projeden sonra sonuçlarını öngöremediğimiz bir sınıflandırma yaklaşımı olan kümeleme algoritmalarıyla bir örnek proje gerçekleştirilecektir. Kümeleme yaklaşımı farklı süreçleri boruhattı ve şablon yaklaşımındaki esnek yöntem ve yaklaşımlar geliştirilebileceğini gösterecektir.

5. BÖLÜM MAKİNE ÖĞRENMESİYLE KÜMELEME PROBLEMİ

Giriş

Kümeleme, veri setlerinde benzer özelliklere sahip örnekleri gruplandırma sürecidir. Bu grupların oluşturulması, veri analizi, veri keşfi ve desen tanıma gibi birçok uygulama alanında faydalıdır. Temel amaç, benzerlik ölçüleri kullanarak veri noktalarını birbirine yakın olan gruplara ayırmaktır. Bu sayede veri setinin içindeki gizli yapıları ve ilişkilerin anlaşılması sağlanmış olur. Örneğin, bir müşteri veri setindeki benzer özelliklere sahip müşterileri gruplayarak, pazarlama stratejilerini geliştirebiliriz veya bir gen ifade veri setinde benzer gen ifade profillerine sahip genlerin birlikte çalıştığı biyolojik süreçleri anlayabiliriz[16].

Bu bölümde, kümeleme problemine yönelik farklı yapıları anlamak ve çözümlemek için temel yöntemler ve algoritmaların nasıl uygulandığını, farklı kümeleme algoritmalarının nasıl eğitildiğini ve sonuçlarının nasıl değerlendirilip ve doğrulanacağını keşfetmiş olacaksınız. Diğer bölümlerden farklı bir yaklaşım izlense de, aslında temel adımların aynı olduğunu göreceksiniz.

5.1 Problemin Tanımı

Kredi kartı işlemlerinin analizi ve müşteri segmentasyonu, finansal kuruluşlar ve perakende sektörü gibi birçok sektörde önemli bir rol oynamaktadır. Bu bölümde, bir Banka, pazarlama stratejisini tanımlamak için bir müşteri segmentasyonu geliştirmek istemektedir. Örnek veri seti, 8950 aktif kredi kartı sahibinin son 6 aydaki kullanım davranışını özetlemektedir. Dosya, 18 davranışsal değişkeni içeren müşteri düzeyindedir.

Bu sayede, banka müşterilerini daha iyi anlayabilecek, pazarlama stratejilerini optimize edebilir ve hedef kitleye özelleştirilmiş teklifler sunabilecektir.

5.2 Veri setini Tanıma

Gelişmiş veri hazırlığı: Aşağıdakiler gibi "akıllı" KPI'lar türeterek zenginleştirilmiş bir müşteri profili oluşturmak için aşağıdaki adımların uygulanması önerilmektedir.

- Aylık ortalama alışveriş ve nakit avans tutarı
- Türüne göre satın almalar (tek seferlik, taksitli)
- Alışveriş ve nakit avans işlemi başına ortalama tutar,

- Limit kullanımı (bakiye/kredi limiti oranı),
- Ödemelerin asgari ödemelere oranı vb.
- Gelişmiş raporlama için müşteri profilleri hakkında bilgi edinmek için türetilmiş KPI'ları kullanma
- Hizmetler arasındaki ilişkilerin/yakınlıkların tanımlanması
- Kümeleme: Kredi kartı sahiplerinin davranışsal bölümlerini ortaya çıkarmayı sağlamak için bir veri azaltma, faktör analizi ve bir kümeleme algoritması uygulama
- Ayrıntılı profil oluşturmayı kullanarak kümenin küme özelliklerini belirleme.
- Belirli kümelenme özellikleri için stratejik içgörüler ve stratejilerin uygulanmasını sağlama.

Veri setini tanıma ve analiz etme atılacak adımları belirlemek açısından son derece önemlidir. Aşağıda veri seti sözlüğü sunulmuştur.

- CUST_ID: Kredi kartı sahibi kimliği
- BAKİYE: Aylık ortalama bakiye (günlük bakiye ortalamalarına göre)
- BALANCE_FREQUENCY: Son 12 ayın bakiyeli oranı
- SATIN ALIMLAR: Son 12 ayda harcanan toplam satın alma tutarı
- ONEOFF_PURCHASES: Bir kerelik satın alma işlemlerinin toplam tutarı
- INSTALLMENTS_PURCHASES: Taksitli satın alma işlemlerinin toplam tutarı
- CASH_ADVANCE: Toplam nakit avans tutarı
- SATIN ALMA SIKLIĞI: Satın alma sıklığı (En az bir satın alma işleminin gerçekleştiği ayların yüzdesi)
- ONEOFF_PURCHASES_FREQUENCY: Tek seferlik satın almaların sıklığı PURCHASES_INSTALLMENTS_FREQUENCY: Taksitli satın almaların sıklığı
- CASH_ADVANCE_ FREKANS: Nakit Avans sıklığı
- AVERAGE_PURCHASE_TRX: Satın alma başına ortalama tutar işlem
- CASH_ADVANCE_TRX: Nakit avans başına ortalama tutar işlem

- PURCHASES_TRX: Satın alma başına ortalama tutar işlem
- CREDIT_LIMIT: Kredi limiti
- ÖDEMELER: Toplam ödemeler (müşteri tarafından ekstre bakiyesinin azaltılması için ödenen tutar) dönem
- MINIMUM_PAYMENTS: Toplam minimum dönemde ödenmesi gereken ödemeler.
- PRC_FULL_PAYMEN: Dolu ayların yüzdesi vadesi gelen ekstre bakiyesinin ödenmesi
- SÜRE: Müşteri olarak geçirilen ay sayısı

Veri seti özniteliklerini tanıma model geliştirme adımlarında doğru algoritmanın ve parametrelerin seçilmesine katkı sağlayacaktır.

5.3 Gerekli Kütüphaneleri Yükleme

Bu projeye gereken kütüphaneleri yükleyerek başlanmalıdır (Liste 5.1).

```
# Kütüphaneleri Yükleme
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.preprocessing import StandardScaler,normalize
from sklearn.cluster import KMeans
from sklearn.cluster import MiniBatchKMeans
from sklearn.cluster import AgglomerativeClustering
import scipy.cluster.hierarchy as shc
from sklearn.cluster import DBSCAN
from sklearn.mixture import GaussianMixture
from sklearn.cluster import MeanShift
from sklearn.cluster import estimate_bandwidth
from sklearn import metrics
from sklearn.decomposition import PCA
import warnings
warnings.filterwarnings('ignore')
```

Liste 5.1 *Kütüphanelerin yüklenmesi*

Veri kümesini Kaggle web sitesinden[2] indirilebilir ve "*CC GENERAL.csv*" adıyla yerel çalışma dizininde kaydedilebilir (Liste 5.2).

```
df=pd.read_csv("CC GENERAL.csv")
```

Liste 5.2 *Veri setini yükleme*

Veri setinin yapısı gereği bu çalışmada özniteliklerin isimleri belirtilmemiştir. Bu, sınıf özniteliği (son sütun) dışında, değişkenlerin anlamlı isimleri olmamasından kaynaklanmaktadır. Ayrıca, başlık bilgisi olmadığı da kodlamada belirtilecektir. Bu, ilk satırın sütun adları olarak alınmasını önlemek için olacaktır. Veri kümesini yükledikten sonra sütun isimleri eklenebilecektir.

5.3 Veriyi Analiz Etme

Veri kümesinin analizi veriyi görselleştirme, eksik ve hatalı verileri veri türlerinin dağılımı hakkında bilgi edinme gibi pek çok yönden incelemeyi içermektedir.

5.3.1 Tanımlayıcı İstatistik

Veri kümesinin boyutları, her bir özniteliğin türü ve örnek adedi göstermek için Liste 5.3'teki kod parçası kullanılmaktadır.

```
print('Bu veri seti {} satır  ve {} sütundan oluşmaktadır.\n'
.format(df.shape[0], df.shape[1])) df.info()
```

Liste 5.3 *Veri seti bilgilerinin incelenmesi*

Veri setinde 8950 satır 18 sütun veri bulunmaktadır (Liste 5.4). Veri seti float, nesne ve int türünde değerler içeren özniteliklere sahiptir. Ve bazı öznitelikler verilerin eksik olduğu görülmektedir.

```
Bu veri seti 8950 satır ve 18 sütundan oluşmaktadır.
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   CUST_ID                             8950 non-null   object
 1   BALANCE                             8950 non-null   float64
 2   BALANCE_FREQUENCY                   8950 non-null   float64
 3   PURCHASES                           8950 non-null   float64
 4   ONEOFF_PURCHASES                    8950 non-null   float64
 5   INSTALLMENTS_PURCHASES              8950 non-null   float64
 6   CASH_ADVANCE                        8950 non-null   float64
 7   PURCHASES_FREQUENCY                 8950 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY          8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY    8950 non-null   float64
 10  CASH_ADVANCE_FREQUENCY              8950 non-null   float64
 11  CASH_ADVANCE_TRX                    8950 non-null   int64
 12  PURCHASES_TRX                       8950 non-null   int64
 13  CREDIT_LIMIT                        8949 non-null   float64
 14  PAYMENTS                            8950 non-null   float64
 15  MINIMUM_PAYMENTS                    8637 non-null   float64
 16  PRC_FULL_PAYMENT                    8950 non-null   float64
 17  TENURE                              8950 non-null   int64
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

Liste 5.4 *Veri seti öznitelik veri türleri çıktısı*

Veri setindeki özniteliklerin istatistiksel özetini sağlamak için kullanılmaktadır. Bu fonksiyon, veri setinin temel istatistiksel özelliklerini hesaplar ve bu özellikleri içeren bir özet tablo döndürür.

Özet tablo, veri setindeki sayısal sütunlar için aşağıdaki istatistiksel bilgileri içerir:

Count: Sütundaki değerlerin toplam sayısı.

Mean: Sütundaki değerlerin ortalaması.

Std: Sütundaki değerlerin standart sapması.

Min: Sütundaki en küçük değer.

25%: Sütundaki değerlerin ilk çeyreği.

50%: Sütundaki değerlerin medyanı veya ikinci çeyreği.

75%: Sütundaki değerlerin üçüncü çeyreği.

Max: Sütundaki en büyük değer.

Bu istatistikler, veri setinin dağılımı ve merkezi eğilimleri hakkında bilgi sağlar. df.describe() fonksiyonu, veri setini hızlı bir şekilde anlamak, veri değerlerindeki anormallikleri tespit etmek ve temel istatistikleri görselleştirmek için kullanışlıdır (Liste 5.5 ve Şekil 5.1).

```
df.describe()
pd.set_option('display.max_columns', 10)  # En fazla 10 sütun
göster
print(df.describe())
```

Liste 5.5 *Veri setin istatistiksel olarak özetleme*

```
             BALANCE  BALANCE_FREQUENCY     PURCHASES  ONEOFF_PURCHASES  \
count    8950.000000        8950.000000   8950.000000       8950.000000
mean     1564.474828           0.877271   1003.204834        592.437371
std      2081.531879           0.236904   2136.634782       1659.887917
min         0.000000           0.000000      0.000000          0.000000
25%       128.281915           0.888889     39.635000          0.000000
50%       873.385231           1.000000    361.280000         38.000000
75%      2054.140036           1.000000   1110.130000        577.405000
max     19043.138560           1.000000  49039.570000      40761.250000

       INSTALLMENTS_PURCHASES  ...  CREDIT_LIMIT      PAYMENTS  \
count             8950.000000  ...   8949.000000   8950.000000
mean               411.067645  ...   4494.449450   1733.143852
std                904.338115  ...   3638.815725   2895.063757
min                  0.000000  ...     50.000000      0.000000
25%                  0.000000  ...   1600.000000    383.276166
50%                 89.000000  ...   3000.000000    856.901546
75%                468.637500  ...   6500.000000   1901.134317
max              22500.000000  ...  30000.000000  50721.483360

       MINIMUM_PAYMENTS  PRC_FULL_PAYMENT       TENURE
count       8637.000000       8950.000000  8950.000000
mean         864.206542          0.153715    11.517318
std         2372.446607          0.292499     1.338331
min            0.019163          0.000000     6.000000
25%          169.123707          0.000000    12.000000
50%          312.343947          0.000000    12.000000
75%          825.485459          0.142857    12.000000
max        76406.207520          1.000000    12.000000

[8 rows x 17 columns]
```

Şekil 5.1 *Veri setin istatistiksel olarak özeti*

5.3.2 Veriyi Temizleme

Veri setindeki eksik değerler Liste 5.6 ile belirlenmiştir. "*MINIMUM_PAYMENTS*" özniteliğinden 313, *"CREDIT_LIMIT"* özniteliğinden 1 eksik veri vardır.

```
#Eksik değerleri kontrol edip uygun yöntemle dolduralım
def null_values(df):
    nv=pd.DataFrame(df.isnull().sum()).rename(columns={0:'Missing_
Records'})
    return nv[nv.Missing_Records>0].sort_values('Missing_Records',
ascending=False)
null_values(df)
Missing_Records
MINIMUM_PAYMENTS       313
CREDIT_LIMIT   1
```

Liste 5.6 *Veri setindeki eksik verileri belirleme*

Eksik verileri silmek bir tercih olarak görülse de veri kaybına ve modelleme eksik bir faktöre sebep olabilir. Bu sebeple eksik değerli hücreler ortalama değer ile tamamlamak uygun bir yaklaşım olacaktır (Liste5.7).

```
df['MINIMUM_PAYMENTS']=df['MINIMUM_PAYMENTS'].fillna(df.
MINIMUM_PAYMENTS.mean())
df['CREDIT_LIMIT']=df['CREDIT_LIMIT'].fillna(df.CREDIT_
LIMIT.mean())
null_values(df).sum()
Missing_Records    0
dtype: int64
```

Liste 5.7 *Veri setindeki eksik verilere "mean" ortalama değer atama*

Kullanılmayan sütunların silinmesi veri setini küçülterek daha hızlı işlem yapmak içinm uygun bir yaklaşım olacaktır.

```
df=df.drop('CUST_ID',axis=1)
```

Liste 5.8 *Veri setindeki veri türleri*

Veri üzerinde standardizasyon ve normalizasyon uygulama genellikle performans arttırıcı bir etki göstermektedir. Bu sebeple Liste 5.9'da standardizasyon ve normalizasyon gerçekleştirilmektedir.

```
norm=normalize(df_scl)
df_norm=pd.DataFrame(norm)
```

Liste 5.9 *Veri setini standardize ve normalize etme*

Veri seti modelleme için hazır hale getirildikten sonra farklı algoritmalar ve performans karşılaştırmaları gerçekleştirilecektir.

5.4 Kümeleme Algoritmaları ile Modelleme

Kredi kartı kümeleme için kullanılan birçok farklı algoritma vardır, ancak en yaygın olarak kullanılan algoritmalar arasında K-Means, GMM (Gaussian Mixture Models) ve Meanshift algoritmaları sayılabilir. Bu çalışmada bu üç algortima performans açısından karşılaştırılacak ve en başarılı model sonuçları incelenecektir.

Algoritma seçimi, veri setinin özelliklerine, veri noktalarının sayısına, performans gereksinimlerine ve analiz hedeflerine bağlı olabilir. Örneğin, K-Means algoritması hızlı ve ölçeklenebilir olması nedeniyle büyük veri setleri için uygun olabilirken, GMM Gaus dağılımı gösteren veri setlerinde doğal bir kümüleme işlemi gerçekleştirebilmektedir. Bir diğer güçlü kümeleme algoritması olarak Meanshift algoritması K-Means algoritmasından parametre gerektirmemesi açısında ayrılır. Algoritma seçiminde elde edilen küme sonuçlarının amaç ve ihtiyaçlara uygun ve analiz hedeflerine hizmet etmesi de önemlidir.

5.4.1 K-Means Algortiması ile Modelleme

Algoritma, her bir kümenin merkezini hesaplar ve ardından her bir veri örneğini, en yakın merkezi olan kümeye atar. Bu süreç, merkezlerin değiştiği ve veri örneklerinin kümelerine yeniden atandığı bir döngü içinde tekrarlanır. İnertial terimi, k-means algoritmasında kullanılan bir performans ölçüsüdür. İnertial, bir kümenin merkezindeki ortalama mesafenin karesidir (Eşitlik 5.1). İnertial, kümelerin homojenliğini ve ayırt edilebilirliğini ölçmek için kullanılır.

$$\text{İnertia} = \Sigma(x - \mu)^2 \qquad 5.1$$

Eşitlik 5.1'de; x: Bir veri örneği, μ: Bir kümenin merkezi, Σ: Toplamı ifade etmektedir. İnertial, kümelerin homojenliğini ve ayırt edilebilirliğini ölçmek için kullanılır. Bir küme ne kadar homojen olursa, inertial o kadar düşük olur. Bir küme ne kadar ayırt edilebilir olursa, inertial o kadar yüksek olur.İnertial, k-means algoritmasının performansını iyileştirmek için kullanılabilir. Örneğin, inertial yüksekse, algoritma daha fazla yineleme gerçekleştirebilir. Bu, kümelerin daha homojen ve ayırt edilebilir olmasını sağlayabilir.

```
import numpy as np
from sklearn.cluster import KMeans
def optimal_k(data, n_init=10, max_iter=300):
    # Küme sayısının bir listesini oluştur
    k_list = np.arange(2, 14)
    n_init=10
    max_iter=300
  # Her bir küme sayısı için inertial değerlerini hesapla
    inertia_list = []
    for k in k_list:
        kmeans = KMeans(n_clusters=k, n_init=n_init, max_iter=max_
iter)
        kmeans.fit(df_norm)
        inertia_list.append(kmeans.inertia_)
  # Elbow yöntemini kullanarak optimal küme sayısını bul
    elbow = np.diff(inertia_list).argmin() + 2
    return k_list[elbow]
optimal_k()
Çıktı :optimal k değeri 4 olarak hesaplanmıştır.
```

Liste 5.10 *Otomatik ideal kümeleme (k) değerini hesaplama*

Normalize ve standardize edilmiş veri setine göre önerilen k değeri 4 olarak hesaplanmıştır. Optimal yaklaşım ne kadar bir başlangıç değeri önerse de diğer k değerleri Sihoutte Score değeri hesaplanarak kontrol edilmelidir ve k=2'den 11'e kadar Liste 5.11 ile hesaplanmıştır.

```
for i in range(2,11):
    kmeans_labels=KMeans(n_clusters=i,random_state=123).
fit_predict(df_norm)
    print("Silhouette score for {} clusters k-means : {}
".format(i,metrics.silhouette_score(df_norm,kmeans_labels,
metric='euclidean').round(3)))
Çıktı:
Silhouette score for 2 clusters k-means : 0.227
Silhouette score for 3 clusters k-means : 0.207
Silhouette score for 4 clusters k-means : 0.207
Silhouette score for 5 clusters k-means : 0.229
Silhouette score for 6 clusters k-means : 0.222
Silhouette score for 7 clusters k-means : 0.238
Silhouette score for 8 clusters k-means : 0.239
Silhouette score for 9 clusters k-means : 0.219
Silhouette score for 10 clusters k-means : 0.217
```

Liste 5.11 *K-Mean algoritmasında k değerlerine göre Sihoutte Score*

Bir diğer performans göstergesi olan Davies-Bouldin indeksi liste 5.12'de hesaplanmıştır.

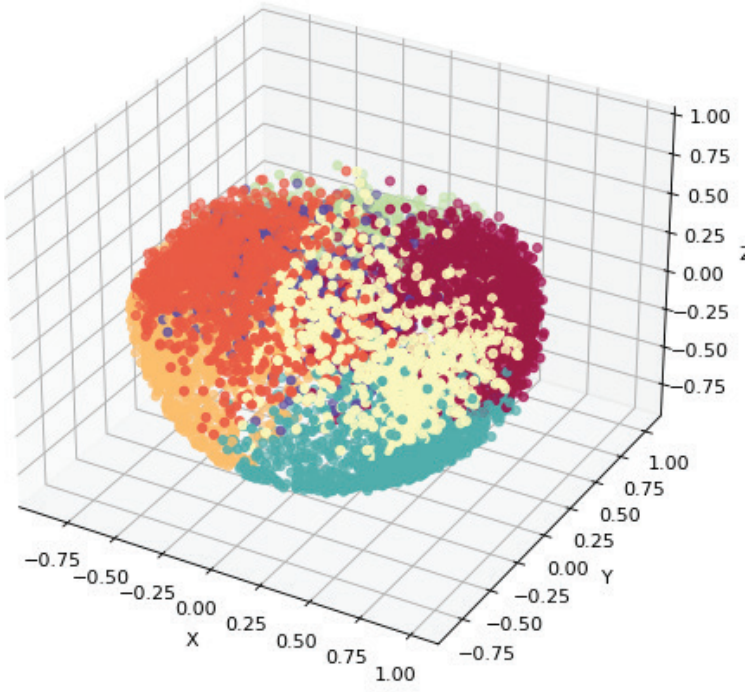
```
for i in range(2,11):
    kmeans_labels=KMeans(n_clusters=i,random_state=123).fit_
predict(df_norm)
    print(' for k='+str(i)+' Davies Bouldin Score:'+str(metrics.
davies_bouldin_score(df_norm,kmeans_labels).round(3)))
 for k=2 Davies Bouldin Score:1.786
 for k=3 Davies Bouldin Score:1.662
 for k=4 Davies Bouldin Score:1.72
 for k=5 Davies Bouldin Score:1.525
 for k=6 Davies Bouldin Score:1.465
 for k=7 Davies Bouldin Score:1.354
 for k=8 Davies Bouldin Score:1.415
 for k=9 Davies Bouldin Score:1.474
 for k=10 Davies Bouldin Score:1.466
```

Liste 5.12 *Davies-Bouldin skoru özniteliklerin istatistiksel tanımı*

Otomatik kümeleme hesaplayıcı sonucunda k=4 bulunmasına rağmen Davies Bouldin ve Silhouette değerleri açısından problem daha iyi analiz etmek için daha fazla küme sayısının da faydası düşünülerek k=7 değerine göre K-means algoritması eğitilerek 7 kümeli bir veri dağıtımı Liste 5.13'te gerçekleştirilmiştir. Veri dağılımları öznitelik sayısı fazla olduğundan PCA gerçekleştirilerek spektral grafik çizilmiştir.

```
kmeans_labels=KMeans(n_clusters=7,random_state=123).fit_predict(df_
norm)
pca = PCA(n_components=3).fit_transform(df_norm)
fig = plt.figure(figsize=(12, 7), dpi=80, facecolor='w',
edgecolor='k')
ax = plt.axes(projection="3d")
ax.scatter3D(pca.T[0],pca.T[1],pca.T[2],c=kmeans_
labels,cmap='Spectral')
xLabel = ax.set_xlabel('X')
yLabel = ax.set_ylabel('Y')
zLabel = ax.set_zlabel('Z')
```

Liste 5.13 *Veri setindeki özniteliklerin istatistiksel tanım çıktısı*



Şekil 5.2 *K-means k=7 için PCA gerçekleştirilmiş spektral grafiği*

6.2 GMM (Gaussian Mixture Models) Algortiması ile Modelleme

Gauss Karışım Modelleri (GMM'ler), çok sayıda Gauss dağılımının olduğunu ve bunların her birinin bir kümeyi temsil ettiğini varsayar. Bu nedenle Gauss Karışım Modeli, tek bir dağılıma ait veri noktalarını bir arada gruplandırma eğilimindedir. Optimum küme sayısı (k) AIC ve BIC'i en aza indirgeyen küme sayısıdır ve GMM'i uygulamak için öncelikle belirlenmelidir.

AIC, modelin uyumunun karmaşıklığı ve genelleme yeteneği arasındaki dengeyi sağlamak için kullanılır. AIC değeri, küme sayısının azalmasıyla birlikte modelin uyumunu düşürme eğilimindedir. AIC'nin düşük olması, daha iyi bir model uyumu ve daha az karmaşıklık anlamına gelir. İstatistiksel bir model ele alalım. L modeli için olabilirlik fonksiyonu maksimize değeri olsun; k modelde tahmin edilen parametre sayısı olsun. Böylece modelin akaike (AIC) ölçütü değeri denklem 5.2'ile ifade edilmektedir[17].

$$AIC = 2k - 2\ln(L) \quad 5.2$$

BIC, AIC'ye benzer şekilde model seçiminde kullanılan bir kriterdir. BIC, modelin uyumunu değerlendirmek için AIC'ye benzer bir terim içerir, ancak biraz daha fazla düzenleme yapar. BIC, modelin karmaşıklığını daha fazla cezalandırır ve daha az karmaşık modelleri tercih eder. Burada, L ve k değerleri AIC'dekiyle aynıdır, n ise veri noktalarının sayısını temsil eder. BIC değeri en düşük olan model, en iyi model olarak kabul edilir[18].

$$BIC = kln(n) - 2\ln(\text{maks}(L)) \quad 5.3$$

AIC ve BIC gibi kümeleme parametreleri, farklı küme sayılarını değerlendirerek en uygun küme sayısını belirlemeye yardımcı olur. Daha düşük AIC veya BIC değeri, daha iyi bir model uyumu ve daha az karmaşıklık anlamına gelir.

```
import numpy as np
from sklearn.mixture import GaussianMixture
def optimal_k(data, n_components_list, n_init=10, max_iter=300):

  # Küme sayısının bir listesini oluştur
    k_list = np.array(n_components_list)
  # Her bir küme sayısı için BIC ve AIC değerlerini hesapla
    bic_list = []
    aic_list = []
    for k in k_list:
        gmm = GaussianMixture(n_components=k, n_init=n_init, max_
iter=max_iter)
        gmm.fit(data)
        bic_list.append(gmm.bic(data))
        aic_list.append(gmm.aic(data))
  # BIC ve AIC değerlerini karşılaştırarak optimal küme sayısını
bul
        bic_min_index = np.argmin(bic_list)
        aic_min_index = np.argmin(aic_list)
        optimal_k = k_list[bic_min_index] if bic_list[bic_min_
index] < aic_list[aic_min_index] else k_list[aic_min_index]
    return optimal_k
# Örnek kullanım
data = df_norm
optimal_k = optimal_k(data, n_components_list=[2, 3, 4, 5, 6])
print(optimal_k)
Çıktı: optimal k=6 olarak elde edilmiştir.
```

Liste 5.14 *GMM için optimal kümeleme sayısı*

Optimal küme sayısı Liste 5.14 ile k = 6 olarak belirlendikten sonra Silhouette Score ve Davies Bouldin Score performans değerleri belirlenir (Liste 5.15).

```
parameters=['full','tied','diag','spherical']
n_clusters=np.arange(1,21)
results_=pd.DataFrame(columns=['Covariance Type','Number of
Cluster','Silhouette Score','Davies Bouldin Score'])
for i in parameters:
    for j in n_clusters:
        gmm_cluster=GaussianMixture(n_components=j,covariance_
type=i,random_state=123)
        clusters=gmm_cluster.fit_predict(df_norm)
        if len(np.unique(clusters))>=2:
          results_=results_.append({"Covariance Type":i,'Number
of Cluster':j,"Silhouette Score":metrics.silhouette_score(df_
norm,clusters),
                                    'Davies Bouldin
Score':metrics.davies_bouldin_score(df_norm,clusters)}
                                   ,ignore_index=True)
```

Liste 5.15 *Veri setindeki özniteliklerin istatistiksel tanım çıktısı*

Performans değerlerinin kümüleme sayısı 1 ile 21 aralığındaki tüm Silhouette ve Davies Bouldin değerleri belirlendikten sonra en yüksek Silhouette'değeri 1. Sırada olacak şekilde ilk 10 başarılı metot Liste 5.16'da gösterilmiştir. Genel olarak Davies değerinin düşük, Silhouette değerinin büyük olması istenmektedir. Kredi kartı müşteri kategorizasyonu gerçekleştirmek için küme sayısının daha fazla olması daha doğru bir hedef kitleyi belirlemeye imkân verdiğinden "spherical" kovaryans tipinde küme sayısının 5 olduğu parametreler daha başarılı bir model parametresi tercihi olacaktır.

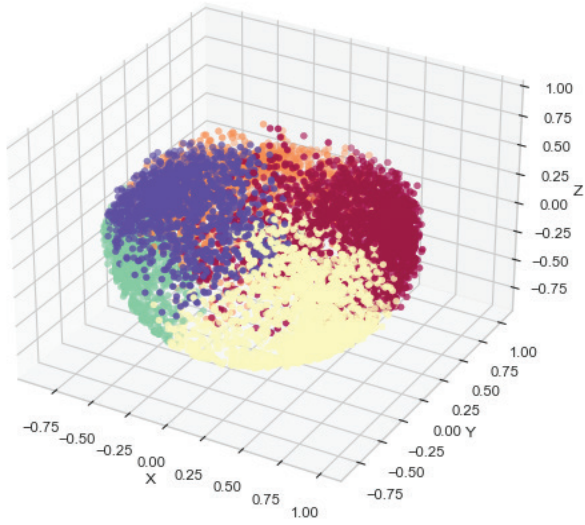
```
display(results_.sort_values(by=["Silhouette Score"],
ascending=False)[:10])
Çıktı:
```

	Covariance Type	Number of Cluster	Silhouette Score	Davies Bouldin Score
57	spherical	2	0.225882	1.788540
19	tied	2	0.221231	1.820339
60	spherical	5	0.207279	1.558893
59	spherical	4	0.205902	1.713049
38	diag	2	0.203435	1.861149
20	tied	3	0.202024	1.593513
61	spherical	6	0.199689	1.546152
58	spherical	3	0.194698	1.539195
26	tied	9	0.194011	1.632332
22	tied	5	0.192110	1.707954

Liste 5.16 *Silhouette değerine göre en başarılı 10 model parametreleri*

Elde edilen sonuçlar temel alınarak 5 kümeli "spherical" (küresel) tipte bir model eğitimi ve tahminleme sonuçlarını Şekil 5.3'te gösterelim.

```
gmm_cluster=GaussianMixture(n_components=5,covariance_
type="spherical",random_state=123)
gmm_cluster.fit(df_norm)
gmm_labels = gmm_cluster.predict(df_norm)
fig = plt.figure(figsize=(10, 7), dpi=80, facecolor='w',
edgecolor='k')
ax = plt.axes(projection="3d")
ax.scatter3D(pca.T[0],pca.T[1],pca.T[2],c=gmm_
labels,cmap='Spectral')
xLabel = ax.set_xlabel('X')
yLabel = ax.set_ylabel('Y')
zLabel = ax.set_zlabel('Z')
Çıktı:
```



Liste 5.17 *Gauss Mixture Model'in tahminleme grafiği*

6.3 MeanShift Algortiması ile Modelleme

MeanShift algoritması, denetimsiz öğrenme alanında kullanılan etkili bir kümeleme algoritmasıdır. K-ortalamalı kümeleme yöntemiyle karşılaştırıldığında, herhangi bir önceden belirlenmiş varsayım gerektirmez ve bu nedenle parametrik olmayan bir algoritmadır[19].

MeanShift algoritması, veri noktalarını kümelere atamak için yüksek yoğunluğa sahip olan veri noktalarının yönünde iteratif olarak kaydırma işlemi yapar. Bu işlem, veri noktalarının küme merkezine doğru hareket etmesini sağlar. Algoritma, her bir veri noktasını, o noktanın etrafındaki

yoğunluğa göre ağırlıklı bir şekilde hareket ettirir. Bu şekilde, veri noktaları yüksek yoğunluğa sahip bölgelere doğru hareket eder ve sonunda küme merkezlerine yakınsar. Algoritma, bu kaydırma işlemini iteratif olarak gerçekleştirerek, veri noktalarını en uygun kümelere atar.

MeanShift algoritması, veri setinin şekli ve dağılımı hakkında herhangi bir önceden bilgiye ihtiyaç duymaz. Bu nedenle, farklı veri setlerinde ve karmaşık veri yapılarında da iyi performans gösterebilir. Yoğunluk tabanlı bir yaklaşım olduğu için, veri setindeki yoğun bölgelere odaklanır ve veri noktalarını bu bölgelere doğru çeker.

Sonuç olarak, MeanShift algoritması, parametrik olmayan bir kümeleme yöntemi olarak veri noktalarını en yüksek yoğunluklu bölgelere doğru iteratif olarak kaydırarak kümelere atar. Bu algoritma, veri setinin yapısı hakkında önceden bilgiye ihtiyaç duymadan, çeşitli veri setlerinde etkili bir şekilde kullanılabilir. Kredi kartı müşteri kategorizasyonu için Liste 5.18'de önerilen kümleme sayısının belirlenmesi gösterilmiştir.

```
# Meanshift
est_bandwidth = estimate_bandwidth(df_norm,quantile=0.1,n_
samples=10000)
mean_shift = MeanShift(bandwidth= est_bandwidth, bin_
seeding=True).fit(df_norm)
labels_unique=np.unique(mean_shift.labels_)
n_clusters_=len(labels_unique)
print("Kümeleme sayısı: %d" % n_clusters_)
Çıktı:
# Önerilen kümeleme sayısı = 3
```

Liste 5.18 *MeanShift algoritmasıyla önerilen kümeleme sayısı tespiti*

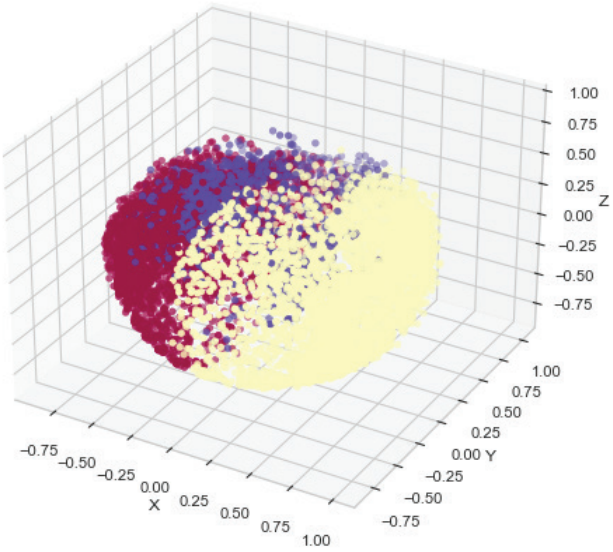
Kümeleme sayısı 3 için MeanShift algoritması performans Davies ve Silhouette değerleri liste 5.19'da hesaplanmıştır.

```
print('Silhouette değeri:'+str(metrics.silhouette_score(df_
norm,mean_shift.labels_,metric='euclidean').round(3)))
print('Davies Değeri:'+str(metrics.davies_bouldin_score(df_
norm,mean_shift.labels_).round(3)))
Çıktı:
Silhouette değeri:0.181
Davies Bouldin değeri:2.114
```

Liste 5.19 *MeanShift algoritması performans Davies ve Silhouette değerleri*

Elde edilen performans değerlerinden sonra model kümeleme grafiği Liste 5.20'de çizdirilmiştir.

```
fig = plt.figure(figsize=(12, 7), dpi=80, facecolor='w',
edgecolor='k')
ax = plt.axes(projection="3d")
ax.scatter3D(pca.T[0],pca.T[1],pca.T[2],c=mean_shift.
labels_, cmap='Spectral')
xLabel = ax.set_xlabel('X')
yLabel = ax.set_ylabel('Y')
zLabel = ax.set_zlabel('Z')
```



Liste 5.20 *MeanShift algoritması kümeleme dağılımı*

6.4 Algoritmaları Değerlendirme

Kredi kartı veri seti ile müşteri eğilimlerinin belirlenmesi için kümeleme algoritmalarından K-means, GMM ve MeanShift algoritmaları eğitilmiştir. En başarıl algoritması belirleyebilmek için Liste 5.21'de Silhouette ve Davies Boulding puanları listelenerek karşılaştırılmıştır.

```
algorithms=["K-Means","Gaussian Mixture Model","MeanShift"]
# Silhouette Score
ss=[metrics.silhouette_score(df_norm,kmeans_
labels),metrics.silhouette_score(df_norm,gmm_
labels),metrics.silhouette_score(df_norm,mean_shift.
labels_)]
# Davies Bouldin Score
db=[metrics.davies_bouldin_score(df_norm,kmeans_
labels),metrics.davies_bouldin_score(df_norm,gmm_
labels),metrics.davies_bouldin_score(df_norm,mean_shift.
labels_)]
comprsn={"Algortimalar":algorithms,"Davies
Bouldin":db,"Silhouette Score":ss}
compdf=pd.DataFrame(comprsn)
display(compdf.sort_values(by=["Silhouette Score"],
ascending=False))
```

	Algortimalar	Davies Bouldin	Silhouette Score
0	K-Means	1.466383	0.216968
1	Gaussian Mixture Model	1.558893	0.207279
2	MeanShift	2.114327	0.180770

Liste 5.21 *Modellerin performansa göre listelenmesi*

Kredi kartı kümeleme problemindeki en başarılı modeli incelersek:

K-Means: Davies Bouldin Score değeri 1.466383 ve Silhouette Score değeri 0.216968 olarak verilmiştir. K-Means algoritması, düşük Davies Bouldin Score ve yüksek Silhouette Score değerleriyle performansı en iyi olan modeldir. Bu, küme içi veri noktalarının birbirlerine yakın olduğunu ve küme merkezlerinin birbirinden uzak olduğunu gösterir. Bu nedenle, K-Means algoritması kredi kartı kümeleme probleminde en başarılı model olarak değerlendirilebilir.

Gaussian Mixture Model: Davies Bouldin Score değeri 1.558893 ve Silhouette Score değeri 0.207279 olarak verilmiştir. Gaussian Mixture Model, K-Means algoritmasına kıyasla biraz daha yüksek Davies Bouldin Score değerine sahip olsa da hala iyi bir performans sergilemektedir. Silhouette Score değeri de kabul edilebilir bir düzeydedir. Bu nedenle, Gaussian Mixture Model kredi kartı kümeleme probleminde ikinci en başarılı model olarak değerlendirilebilir.

MeanShift: Davies Bouldin Score değeri 2.114327 ve Silhouette Score değeri 0.180770 olarak verilmiştir. MeanShift algoritması, diğer iki modele kıyasla daha yüksek bir Davies Bouldin Score değerine ve daha düşük bir Silhouette Score değerine sahiptir. Bu, küme merkezlerinin birbirine daha yakın olduğunu ve veri noktalarının daha az ayrıştığını gösterir. Bu

nedenle, MeanShift algoritması kredi kartı kümeleme probleminde en az başarılı model olarak değerlendirilebilir.

Sonuç olarak, verilen sonuçlara göre en başarılı model K-Means algoritmasıdır. K-Means, düşük Davies Bouldin Score ve yüksek Silhouette Score değerleriyle daha iyi bir performans sergilemektedir ve kredi kartı kümeleme probleminde en uygun çözümü sunmaktadır.

K-means algortması en başarılı algortimala olarak belirlenmiştir. Son olarak kümeler ve özellikleri incelenecektir. Liste 5.22'de kümelerdeki müşteri sayısı gösterilmektedir.

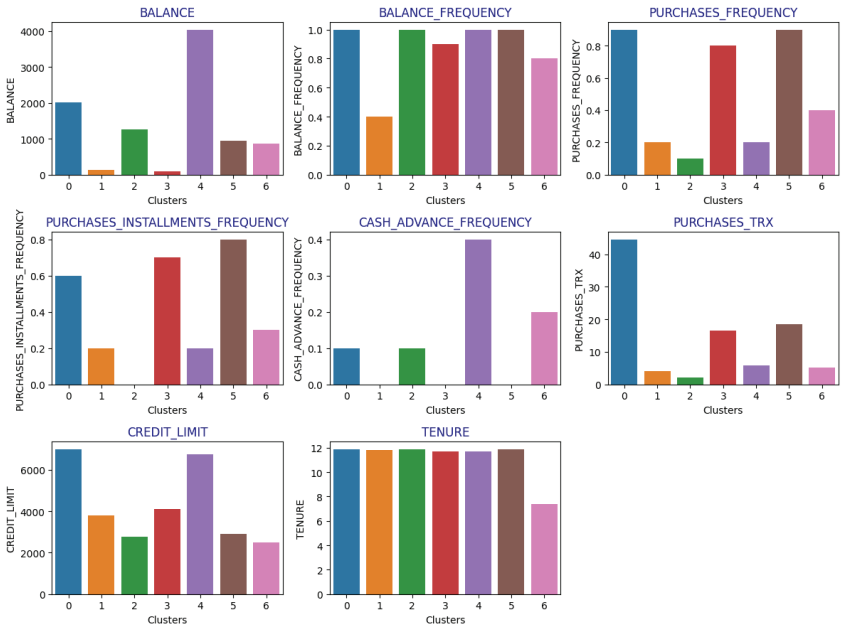
```
# Test seçenekleri ve doğrulama metrikleri
df['Clusters']=list(kmeans_labels)
customers=pd.DataFrame(df['Clusters'].value_counts()).
rename(columns={'Clusters':'Number of Customers'})
customers.T
Çıktı:
```

	0	1	2	3	4	8	7	5	9	6
Number of Customers	1250	1152	1109	1028	857	781	763	720	650	640

Liste 5.22 *K-means kümelere göre müşteri sayısı*

Müşteri dağılımı kümelendikten sonra belirlenen öznitekilere göre dağılımı Liste 5.23'te gösterilmiştir.

```
import matplotlib.pyplot as plt
import seaborn as sns
df.set_index('Clusters')
grouped=df.groupby(by='Clusters').mean().round(1)
features=["BALANCE","BALANCE_FREQUENCY","PURCHASES_
FREQUENCY","PURCHASES_INSTALLMENTS_FREQUENCY","CASH_ADVANCE_
FREQUENCY","PURCHASES_TRX","CREDIT_LIMIT","TENURE"]
plt.figure(figsize=(12,9))
for i,j in enumerate(features):
    plt.subplot(3,3,i+1)
    sns.barplot(x=grouped.index, y=grouped[j])
    plt.title(j,fontdict={'color':'darkblue'})
plt.tight_layout()
plt.show()
Çıktı:
```



Liste 5.23 *K-means kümelere göre müşteri sayısı*

Kümeleri tanımlamak için önemli olduğu düşünülen öznitelikler seçilmiş ve 7 kümeli bir model önerilmiştir. Kümeler ve özellikleri:

Küme 0: Bu kümeye ait müşteriler en yüksek satın alma sıklığına sahiptir. Ayrıca, taksitli ödemeye eğilim gösterirler. Bu müşteriler, diğer müşterilere göre daha yüksek kredi limitine ve uzun süreli müşteri olma özelliğine sahiptir. Yani, daha sık alışveriş yaparlar ve satın aldıkları ürünleri taksitler halinde ödemeyi tercih ederler.

Küme 1: Bu kümedeki müşterilerin bakiyeleri ve satın alma sıklıkları oldukça düşüktür. Kredi kartını nadiren kullanırlar ve daha düşük kredi

limitine sahiptirler. Bu müşteriler genellikle kredi kartlarını aktif bir şekilde kullanmazlar ve daha az harcama yaparlar.

Küme 2: Bu küme, en fazla müşteri sayısına sahip olan kümelerden biridir, ancak kredi kartlarını daha az kullanma eğilimindedirler. Bu müşteriler genellikle pasif müşterilerdir ve kredi kartlarını aktif bir şekilde kullanmazlar. Ayrıca, uzun süreli müşterilerdir, yani uzun bir süre boyunca müşteriler olarak kalma eğilimindedirler.

Küme 3: Bu kümedeki müşteriler taksitli ödemeye yüksek bir eğilime sahiptir ve diğer müşterilere göre daha yüksek satın alma sıklığına sahiptirler. Ayrıca, müşterilerin süreleri ortalamanın üzerindedir, yani uzun bir süre boyunca müşteri olarak kalmaya meyillidirler.

Küme 4: Bu kümedeki müşteriler en yüksek bakiye miktarına sahiptir, ancak satın alma sıklıkları çok yüksek değildir. Bu müşteriler nakit avans yapma eğilimindedirler ve diğer müşterilere göre daha yüksek kredi limitine sahiptirler. Ancak, para harcamaktan pek hoşlanmazlar ve daha tutumlu bir tutum sergilerler.

Küme 5: Bu kümedeki müşteriler ikinci en yüksek satın alma sıklığına sahiptir ve taksitli ödemelerde daha yüksek bir eğilime sahiptir. Aynı zamanda bu müşteriler uzun süreli müşterilerdir, yani uzun bir süre boyunca müşteri olarak kalmayı tercih ederler.

Küme 6: Bu kümedeki müşteri sayısı en az olan gruptur. Bu müşterilerin satın alma sıklığı ortalamanın altında ve süreleri kısa sürelidir. Yani, bu müşteriler genellikle daha az alışveriş yaparlar ve kısa bir süre boyunca müşteri olarak kalırlar.

5.7 Özet

Kredi kartı veri seti üzerinden kümeleme makine öğrenmesi adımları Python kullanılarak nasıl gerçekleştirlir bu bölümde göstermeye çalışılmıştır. Diğer bölğmlerden farklı olarak üç farklı algoritmanın kendi özelliklerine göre bir boru hattı yaklaşımı izlenmiştir. Veri ön işlemesi aşamasında, veri setini analize en uygun hale getirmek için eksik verilerin ortalama değer ile değiştirilmesi, aykırı değerlerin bilgi kaybı yaşanmaması için ölçeklenmesi (normalleştirme ve standartlaştırma) işlemleri gerçekleştirildi. Bu adımların amacı, veri setindeki gürültü etkisi gösteren eksik veya hatalı verileri azaltmak ve veriyi modellemeye uygun hale getirmekti.

Kümelenme algoritmaları, veri setinde benzerliklere dayalı olarak örüntüleri tanımlamak ve veri noktalarını belirli gruplara ayırmak için kullanılmaktadır. Bu adımda, farklı kümelenme algoritmaları denenerek performansları karşılaştırıldı. Performans ölçütleri arasında Davies Bouldin ve Silhouette Score değerleri dikkate alındı.

K-Means algoritması, en yaygın kullanılan kümelenme algoritmalarından biri olarak bilinmektedir. K-Means algoritmasının avantajı, basitliği ve başarısı GMM ve MeanShift algoritmalarından daha yüksek olduğu gözlemlenmiştir.

Sonuç olarak, veri ön işlemesi, kümelenme algoritmalarının seçimi, K-Means modelinin kullanılması ve verinin yedi kümeye bölünmesi gibi adımlar, müşteri davranışlarının analiz edilmesi ve özelleştirilmiş stratejilerin geliştirilmesi için önemli bir süreci temsil etmektedir. Bu yaklaşım, işletmelerin müşterilerine daha iyi hizmet sunmak ve rekabet avantajı elde etmek için veriye dayalı kararlar almasına yardımcı olacaktır.

5.8 SONUÇ

Bu kitap boyunca makine öğrenimi projelerinin ne olduğunu ve nasıl yapıldığını öğrendiniz. Ayrıca, Python platformunu kullanarak makine öğrenimi projelerini nasıl geliştireceğinizi öğrendiniz. Bir makine öğrenmesi projesinin veri keşfi, ön işleme, model seçimi ve modelleme gibi adımları kapsadığını sonra, makine öğrenimi modellerini nasıl değerlendirebileceğinizi ve nasıl optimize edebileceğinizi öğrendiniz. Bu, doğrulama kümelerini kullanarak model başarısını ölçmeyi, hiper parametreleri ayarlamayı ve model eğitimini iyileştirmeyi sağlamaktaydı. Teknik ayrıntılarda boğulmadan Python ile makine öğrenimi projelerine nasıl yaklaşacağınızı ve komutlar ve kütüphaneleri nasıl uygulayabileceğinizi öğrendiniz. Bu kitabı bitirerek çok cesur bir adım attınız.

Tebrikler.

Bir an durup ve ne kadar ilerlediğinizi gözden geçirin ve bu motivasyonla internetteki yapılmış çalışmaları öğrendiğiniz boruhattı ve şablon yaklaşımıyla gerçekleştirmeye devam etmelisiniz.

Profesyonel bir veri bilimci olma yolundaki bu büyük adımı yalnız bırakmamamınız ümidiyle.

KAYNAKLAR

[1] M. E. Fortunato, C. W. Coley, B. C. Barnes, ve K. F. Jensen, "Data Augmentation and Pretraining for Template-Based Retrosynthetic Prediction in Computer-Aided Synthesis Planning", *J. Chem. Inf. Model.*, c. 60, sy 7, ss. 3398-3407, Tem. 2020, doi: 10.1021/acs.jcim.0c00403.

[2] M. N. Mahdi *vd.*, "Software Project Management Using Machine Learning Technique—A Review", *Appl. Sci.*, c. 11, sy 11, s. 5183, Haz. 2021, doi: 10.3390/app11115183.

[3] A. Thakkar, T. Kogej, J.-L. Reymond, O. Engkvist, ve E. J. Bjerrum, "Datasets and their influence on the development of computer assisted synthesis planning tools in the pharmaceutical domain", *Chem. Sci.*, c. 11, sy 1, ss. 154-168, 2020, doi: 10.1039/C9SC04944D.

[4] "Makine Öğrenmesi (Machine Learning) Nedir? • Turhost Blog". Erişim: 07 Aralık 2023. [Çevrimiçi]. Erişim adresi: https://www.turhost.com/blog/makine-ogrenmesi-machine-learning-nedir/

[5] A. L. Beam ve I. S. Kohane, "Big Data and Machine Learning in Health Care", *JAMA*, c. 319, sy 13, s. 1317, Nis. 2018, doi: 10.1001/jama.2017.18391.

[6] K. Lin, Y. Xu, J. Pei, ve L. Lai, "Automatic retrosynthetic route planning using template-free models", *Chem. Sci.*, c. 11, sy 12, ss. 3355-3364, 2020, doi: 10.1039/C9SC03666K.

[7] T. R. Browning, "Applying the design structure matrix to system decomposition and integration problems: a review and new directions", *IEEE Trans. Eng. Manag.*, c. 48, sy 3, ss. 292-306, Ağu. 2001, doi: 10.1109/17.946528.

[8] M. Fayaz, A. Khan, J. U. Rahman, A. Alharbi, M. I. Uddin, ve B. Alouffi, "Ensemble Machine Learning Model for Classification of Spam Product Reviews", *Complexity*, c. 2020, ss. 1-10, Ara. 2020, doi: 10.1155/2020/8857570.

[9] C. Grova *vd.*, "Concordance between distributed EEG source localization and simultaneous EEG-fMRI studies of epileptic spikes", *NeuroImage*, c. 39, sy 2, ss. 755-774, Oca. 2008, doi: 10.1016/j.neuroimage.2007.08.020.

[10] J. Otsuka, *Thinking About Statistics: The Philosophical Foundations*, 1. bs. New York: Routledge, 2022. doi: 10.4324/9781003319061.

[11] "MODEL SELECTION IN MACHINE LEARNING", *Int. Res. J. Mod. Eng. Technol. Sci.*, Oca. 2023, doi: 10.56726/IRJMETS32850.

[12] Y. Ye, Y. Xiong, Q. Zhou, J. Wu, X. Li, ve X. Xiao, "Comparison of Machine Learning Methods and Conventional Logistic Regressions for Predicting Gestational Diabetes Using Routine Clinical Data: A Retrospective Cohort Study", *J. Diabetes Res.*, c. 2020, ss. 1-10, Haz. 2020, doi: 10.1155/2020/4168340.

[13] "Home - UCI Machine Learning Repository". Erişim: 12 Aralık 2023. [Çevrimiçi]. Erişim adresi: https://archive.ics.uci.edu/

[14] J. Heer ve B. Shneiderman, "Interactive Dynamics for Visual Analysis: A taxonomy of tools that support the fluent and flexible use of visualizations", *Queue*, c. 10, sy 2, ss. 30-55, Şub. 2012, doi: 10.1145/2133416.2146416.

[15] R. G. Terry Sejnowski, "Connectionist Bench (Sonar, Mines vs. Rocks)". UCI Machine Learning Repository, 1988. doi: 10.24432/C5T01Q.

[16] R. Xu ve D. Wunsch, "Survey of clustering algorithms", *IEEE Trans. Neural Netw.*, c. 16, sy 3, ss. 645-678, May. 2005, doi: 10.1109/TNN.2005.845141.

[17] H. Bozdogan, "Model selection and Akaike's Information Criterion (AIC): The general theory and its analytical extensions", *Psychometrika*, c. 52, sy 3, ss. 345-370, Eyl. 1987, doi: 10.1007/BF02294361.

[18] D. Posada ve T. R. Buckley, "Model Selection and Model Averaging in Phylogenetics: Advantages of Akaike Information Criterion and Bayesian Approaches Over Likelihood Ratio Tests", *Syst. Biol.*, c. 53, sy 5, ss. 793-808, Eki. 2004, doi: 10.1080/10635150490522304.

[19] D. Comaniciu ve P. Meer, "Mean shift: a robust approach toward feature space analysis", *IEEE Trans. Pattern Anal. Mach. Intell.*, c. 24, sy 5, ss. 603-619, May. 2002, doi: 10.1109/34.1000236.

Python ile Pratik Makine Öğrenmesi Projeleri

"Python ile Pratik Makine Öğrenmesi Projeleri" kitabına hoş geldiniz! Bu kitap, makine öğrenmesi projeleri geliştirmek isteyen herkese pratik bir rehber sunmayı hedeflemektedir. Amaç, sizi adım adım yönlendirme yöntemiyle Python dilini kullanarak gerçek dünya projelerini nasıl hayata geçirebileceğinizi öğretmektir.

Makine öğrenmesi, günümüzün en heyecan verici yapay zekâ alanlarından biridir. Veri analizi, tahmin yapma, desenleri tanıma ve otomatik karar verme gibi birçok alanda büyük bir etki yaratmaktadır. Ancak, makine öğrenmesi projeleri geliştirmek karmaşık ve zorlu olabilir. Bu kitap, bu zorlukları aşmanıza yardımcı olacak pratik bilgiler ve gerçek dünya problem çözümleri sunmayı amaçlamaktadır.

SERÜVEN YAYINEVİ

www.seruvenyayinevi.com

/seruvenyayinevi /seruvenyayinevi /seruvenyayinevi

9 786256 760684