T.C.
HARRAN ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

ELEKTRİK – ELEKTRONİK MÜHENDİSLİĞİ PROJESİ

MAKİNE ÖĞRENMESİNDE KÜMELEME VE SINIFLANDIRMA

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

HAZIRLAYAN ÖĞRENCİNİN ADI-SOYADI VE NUMARASI

SABRİ BEREKET - 190507003

DANIŞMAN
Doç. Dr. BUKET SONBAŞ

2022

ŞANLIURFA

İçindekiler

1. GİRİŞ

Çağın en önemli konularından biri veri bilimidir. İş dünyasında rekabet avantajı elde etmek için çok sayıda verinin işlenmesi ve analiz edilmesi gerekir. Bu rapor, veri bilimi alanında önemli bir konu olan makine öğrenmesi ve temel algoritmaların incelenmesine odaklanıyor. Ek olarak, kümelemeyi (kümelenme) ve sınıflandırmayı (sınıflandırmayı) makine öğrenmesi yöntemlerinden ayıran temel farklılıkları ve bu yöntemlerin her birinin uygulama alanlarını ele almaktadır.

Raporun ikinci bölümünde veri bilimi hakkında bir açıklama bulunmaktadır. Veri bilimi, verilerin toplanması, analizi ve değerlendirilmesi yoluyla bilgi toplamayı amaçlar. Temel algoritmalar ve makine öğrenmesi sonraki konulardır. Gözetimli öğrenme ve gözetimsiz öğrenme hakkında konuşuluyor. Kümelenme veya gruplama, gözlemsiz öğrenme yöntemlerinden biri olarak daha ayrıntılı bir şekilde ele alınmaktadır. Clustering algoritmaları ve avantajları ve dezavantajları, tanımının bir parçasıdır. Benzer şekilde, gözetimli öğrenme yöntemlerinden biri olan sınıflandırma (sınıflandırma) da incelenmektedir.

Üçüncü bölüm, klasörleme ile klasörleme arasındaki temel farkları ele alır. Bu bölüm, sınıflandırma ve gruplama ile ilgili sorunları, hedeflerini, veri özelliklerini, hazırlık sürecini, algoritma seçimini ve sonuç değerlendirme süreçlerini karşılaştırmalı olarak ele alır. Ek olarak, gruplama ve gruplama performans kriterleri incelenmekte ve karşılaştırılmaktadır.

Son olarak, dördüncü bölüm uygulama alanlarını ele alıyor. Bu bölümde, sınıflandırma modelleriyle e-posta filtreleme ve gruplama algoritmalarıyla müşteri segmentasyonu gibi örnek uygulamalar ele alınmaktadır. Bu uygulama alanlarında, çeşitli algoritma teknikleri kullanılarak elde edilen sonuçlar karşılaştırılmakta ve değerlendirilmektedir.

Bu raporun amacı, veri bilimi, makine öğrenmesi, gruplandırma ve kategorize etme konularında temel bir anlayış sağlamak. Raporun devamında her bir konu ayrıntılı bir şekilde incelenecektir ve ilgili uygulamalarla bağlantılandırılacaktır.

2.ARAŞTIRMA

2.1. VERİ BİLİMİ TANIMI

Veri bilimi, günümüzün en popüler ve hızla gelişen alanlarından biridir. Büyük veri setleri arasında anlamlı bilgiler elde etmek için farklı disiplinlerin birleşiminden oluşan bir disiplindir. Veri bilimi, matematik, istatistik, bilgisayar bilimi ve makine öğrenimi gibi farklı alanlarda uzmanlıklara sahip kişiler tarafından yürütülmektedir.

Veri bilimi, büyük veri setleri arasında trendleri, desenleri, ilişkileri ve diğer bilgileri keşfetmek için kullanılır. Bu keşifler, işletmeler, hükümetler ve diğer kuruluşlar tarafından daha iyi kararlar almak için kullanılabilir. Örneğin, sağlık sektöründe, veri bilimi hastalıkların önceden tahmin edilmesi ve tedavi planlarının geliştirilmesi gibi birçok fayda sağlamaktadır.

Veri bilimi ayrıca, işletmelerin müşteri davranışlarını analiz etmelerine, verimliliklerini artırmalarına, işletme maliyetlerini azaltmalarına ve yeni ürünler geliştirmelerine yardımcı olabilir. Ayrıca, hükümetlerin de vatandaşlara daha iyi hizmet vermesi için veri biliminden yararlanması mümkündür.

Veri bilimi, veri toplama, depolama, işleme ve analiz gibi bir dizi adımdan oluşur. Veri toplama yöntemleri arasında anketler, gözlem, deneyler ve sensörler gibi farklı yöntemler bulunurken, veri depolama yöntemleri arasında veritabanları, Hadoop ve Spark gibi teknolojiler yer alır.

Veri analiz, veri biliminin en önemli adımlarından biridir. Veri analiz yöntemleri arasında istatistiksel analiz, makine öğrenimi ve veri madenciliği yer alır. İstatistiksel analiz, verilerin özetlenmesi, görselleştirilmesi ve yorumlanması için kullanılan bir yöntemdir. Makine öğrenimi, verilerin modelleme ve tahmin yapmak için kullanılması için geliştirilmiş bir algoritma grubudur. Veri madenciliği, büyük veri setleri arasındaki desenleri ve ilişkileri keşfetmek için kullanılır.

Veri bilimi, bir dizi farklı araç ve teknolojiyi kullanır. Bu araçlar arasında programlama dilleri, veritabanları, veri depolama sistemleri, veri analiz araçları ve yapay zeka tabanlı araçlar bulunur. Veri bilimciler, programlama dilleri arasında Python, R, SQL ve Scala gibi dilleri kullanırken, veritabanları arasında MySQL, Oracle ve Microsoft SQL Server gibi veritabanları tercih ederler. Veri depolama sistemleri arasında ise Hadoop, Spark, NoSQL ve Cassandra gibi sistemler bulunur.

Veri analiz araçları arasında SAS, SPSS, MATLAB ve Excel gibi araçlar bulunurken, yapay zeka tabanlı araçlar arasında TensorFlow, Keras, PyTorch ve Scikit-learn gibi araçlar bulunur. Bu araçlar, veri bilimcilerin büyük veri setlerini analiz etmelerine ve anlamlı bilgiler elde etmelerine yardımcı olur.

Veri bilimi, birçok sektörde kullanılan önemli bir disiplindir. İşletmeler, hükümetler, sağlık sektörü, eğitim sektörü ve diğer birçok sektör, veri biliminin sunduğu faydalardan yararlanırlar. Veri bilimi, işletmelerin verimliliğini artırmasına, maliyetlerini azaltmasına ve müşteri deneyimini iyileştirmesine yardımcı olabilir. Ayrıca, veri bilimi sağlık sektöründe hastalıkların daha önceden tahmin edilmesine, tedavi planlarının geliştirilmesine ve daha iyi hizmet sunulmasına yardımcı olur.

Veri bilimi, gelecekte de hızla gelişmeye devam edecektir. Yapay zeka ve derin öğrenme gibi yeni teknolojiler, veri bilimi alanında büyük bir etkiye sahip olacak ve veri bilimcilerin daha karmaşık sorunları çözmelerine olanak sağlayacaktır. Ayrıca, büyük veri setlerinin artmasıyla birlikte, veri biliminin önemi de artacaktır.

Sonuç olarak, veri bilimi, büyük veri setleri arasında anlamlı bilgiler elde etmek için farklı disiplinlerin birleşiminden oluşan bir disiplindir. Matematik, istatistik, bilgisayar bilimi ve makine öğrenimi gibi farklı alanlarda uzmanlıklara sahip kişilerin bir araya gelerek yürüttüğü bir çalışmadır. Veri bilimi, birçok sektörde fayda sağlamaktadır ve gelecekte de gelişmeye devam edecektir. Bu nedenle, veri bilimine ilgi duyan kişilerin, farklı alanlarda uzmanlaşarak bu alanda başarılı bir kariyere sahip olmaları mümkündür.

2.2 MAKİNE ÖĞRENMESİ VE TEMEL ALGORİTMALARIN İNCELEMESİ

2.2.1 Gözetimsiz Öğrenme

Gözetimsiz öğrenme, etiketlenmemiş veri setlerinde yapıları ve desenleri tanımlama yeteneği sayesinde veri keşfi ve anlayışında önemli bir rol oynar. Bu yöntemde algoritma, bir veri setini analiz ederek verilerin birbirleriyle nasıl ilişkili olduğunu ve benzerlerini bulmaya çalışır. Bu nedenle, veri setinden önemli bilgiler elde edebilir ve veri analizi için değerli sonuçlar elde edebilir.

Gözetimsiz öğrenme, çok çeşitli alanlarda faydalı olabilir. Örneğin, pazarlamacılar müşteri segmentasyonunu tanımlayabilir. Algoritma, tüketicilerin davranışını, tercihlerini veya

demografik özelliklerini analiz ederek benzer müşteri grupları oluşturabilir ve bu gruplara dayalı olarak pazarlama planları oluşturabilir.

Gözetimsiz öğrenme teknikleri, metin verilerindeki önemli kelimeleri bulmak, duygusal tonlamaları kavramak veya metinleri benzer temalar veya konular altında gruplandırmak için doğal dil işleme alanında kullanılabilir. Bu nedenle, metin tabanlı uygulamalarda metin verilerinden önemli bilgileri otomatik olarak çıkarabilir.

Görüntü işleme, genetik veri analizi ve sinyal işleme gibi çok sayıda alanda gözlemsiz öğrenme etkili bir şekilde kullanılabilir. Örneğin, görüntü işleme algoritması, farklı görüntüler arasındaki benzerlikleri ve desenleri belirleyerek nesneleri veya yüzleri sınıflandırabilir veya tanıyabilir.

Gözetimsiz öğrenmede bazı sorunlar vardır. Etiketlenmemiş veri setlerinde desenleri bulmak ve sonuçları yorumlamak daha zor olabilir. Gözetimsiz öğrenme algoritmalarının doğruluğunu objektif bir şekilde değerlendirmek ve sonuçları değerlendirmek çok önemlidir. Bu nedenle, gözetimsiz öğrenme süreçleri sürekli olarak geliştirilmektedir.

Bu nedenle, gözetimsiz öğrenme, veri analitiği ve yapay zeka için güçlü bir yaklaşımdır. Etiketlenmemiş veri setlerindeki yapıları ve desenleri keşfetme yeteneği sayesinde bilgi çıkarma ve veri analizi süreçlerinde faydalı bilgiler sağlar. Gözetimsiz öğrenme teknikleri, veri setlerindeki gizli verileri ortaya çıkarabilir ve gelecekte yeni keşiflerin yapılmasını sağlayabilir.

2.2.1.1. Clustering Nedir ?

Makine öğrenmesi veri analizinde kümelemeyi kullanır. Basitçe, ortak özelliklere sahip veri noktalarını gruplandırmayı amaçlar. Veri setindeki ilişkileri bulmak ve bu ilişkileri kullanarak verileri gruplara ayırmak için kümeleme algoritmaları kullanılır.

Kümeleme, denetimsiz bir öğrenme yöntemidir, bu nedenle örneklerin sınıfları veya etiketleri önceden bilinmez. Veri noktalarının benzer özelliklere sahip olduğunu belirleyerek algoritma kendiliğinden grupları bulmaya çalışır. Kümeleme algoritmaları, farklı gruplar arasındaki benzerlikleri azaltmak için benzer veri noktalarını aynı grupta birleştirir.

Çalışma prensipleri çok çeşitli olsa da, kümeleme algoritmaları genellikle aşağıdakileri içerir:

Veri Setinin Hazırlanması: İlk adım, kümeleme algoritması için uygun bir veri seti seçmek veya oluşturmaktır. Veri seti, analiz edilmesi gereken özellikleri taşıyan veri noktalarını içermelidir.

Kümeleme Algoritmasını Seçmek: İkinci adımda kullanılacak kümeleme algoritması seçilir. Birçok farklı algoritma, farklı veri setleri ve problemler için daha etkili olabilir. Örneğin, yoğunluk tabanlı kümeleme, hiyerarşik kümeleme ve k-means gibi algoritmalar kullanılabilir.

Başlangıç Merkezleri: Bazı kümeleme algoritmaları başlangıç merkezleri belirlemeyi gerektirir. Bu aşamada, küme merkezlerinin rastgele seçilmesi veya başka bir yöntemle belirlenmesi mümkündür.

Kümeleme İterasyonları: Algoritma kümeleme yapmak için iteratif bir teknik kullanır. Veri noktaları her iterasyonda kümelere atanır ve küme merkezleri güncellenir. Kümeleme sonuçları, güncelleme ve yeniden atama süreçlerinin tekrarlanması yoluyla en üst düzeye çıkarılır.

Kümeleme algoritması tamamlandıktan sonra sonuçlar değerlendirilir. Kümeleme sonuçları, veri noktalarının hangi gruplara ait olduğunu ve gruplar arasındaki benzerliklerin düzeyini gösterir. Sonuçlar grafiğe dönüştürülebilir veya istatistiksel analizler kullanılabilir.

Birçok farklı alan, kümeleme algoritmalarını kullanır. Örneğin, pazarlama araştırmaları, sosyal ağ analizi, görüntü işleme, biyoinformatik ve müşteri segmentasyonu gibi alanlarda kümeleme teknikleri yaygın olarak kullanılmaktadır.

Veri setlerinin nasıl düzenlendiğini anlamak, benzer veri noktalarını gruplandırmak ve daha derinlemesine analizler yapmak için önemli bir araç kümelemedir. Bununla birlikte, uygun bir kümeleme algoritması seçmek ve veri setinin özelliklerini dikkate almak, doğru sonuçlar elde etmek için çok önemlidir.

2.2.1.2. Clustering Algoritmalarının İncelemesi

K-Means:

- K-Means, en sık kullanılan ve basit clustering algoritmasıdır.
- Veri noktalarını k belirtilen sayıda kümelere ayırır.
- Başlangıçta rastgele seçilen k küme merkeziyle başlar.
- Veri noktaları en yakın küme merkezine atanır ve küme merkezleri yeniden hesaplanır.
- İteratif olarak atama ve yeniden hesaplama adımları tekrarlanır, kümeleme optimize edilir.
- Kümeleme sonuçları, veri noktalarının hangi kümeye ait olduğunu gösterir.

Hiyerarşik Kümeleme:

- Hiyerarşik kümeleme, veri noktalarını bir hiyerarşik ağaç yapısı olarak gruplar.
- Bu algoritma, aglomeratif (birleştirici) veya bölümleyici (ayırıcı) olabilir.
- Aglomeratif yöntemde, her veri noktası başlangıçta ayrı bir küme olarak kabul edilir ve benzerlik ölçüleri kullanılarak kümeleme gerçekleştirilir.
- Bölümleyici yöntemde, tüm veri noktaları başlangıçta tek bir küme olarak kabul edilir ve benzerlik ölçüleri kullanılarak kümeleme gerçekleştirilir.
- Kümeleme işlemi, veri noktaları arasındaki benzerlik veya uzaklık metriklerine dayanır.
- Hiyerarşik kümeleme sonucunda dendrogram adı verilen bir ağaç yapısı elde edilir.

Yoğunluk Tabanlı Kümeleme (DBSCAN):

- DBSCAN, veri noktalarını yoğunluk temelinde gruplandırır.
- Yoğun bölgeleri belirler ve düşük yoğunluklu alanları ayırır.
- Epsilon (ε) ve MinPts parametreleri kullanılır.
- Epsilon, bir veri noktasının çekirdek noktası olarak kabul edileceği bir yarıçap değeridir.
- MinPts, bir veri noktasının çekirdek noktası olarak kabul edilebilmesi için gereken minimum komşu sayısını belirtir.
- Veri noktaları çekirdek noktaları, sınırlayıcı noktalar veya gürültü noktaları olarak etiketlenir.

Gaussian Karışım Modelleri (GMM):

- GMM, veri noktalarını olasılık dağılımlarıyla temsil eder.
- Her bir küme, birbirinden bağımsız Gauss dağılımına sahip bileşenlerle temsil edilir.
- Bir veri noktası, bileşenlere olan olasılıkları dikkate alınarak kümelere atanır.
- Veri noktalarının hangi bileşene ait olduğu ve bu bileşenlerin parametreleri tahmin edilir.
- GMM, veri setindeki karmaşık dağılımları ve gizli yapıları modellemek için kullanılır.

Bu sadece bazı clustering algoritmalarının örnekleri olup, başka birçok algoritma ve varyasyonları da mevcuttur. Hangi algoritmanın kullanılacağı, veri setinin özelliklerine,

büyüklüğüne ve problem ihtiyaçlarına bağlıdır. Doğru algoritmanın seçimi, iyi bir kümeleme performansı ve anlamlı sonuçların elde edilmesi için önemlidir.

2.2.1.3. Clustering Algoritmalarının Avantajları ve Dezavantajları

Avantajları:

- Veri Keşfi: Kümeleme algoritmaları sayesinde veri setindeki bilinmeyen yapıları keşfedebiliriz. Veri noktalarını benzerlik ölçütlerine göre gruplandırarak farklı veri kümeleri ve desenleri oluşturur.
- Veri Segmentasyonu: Büyük ve karmaşık veri setlerini daha küçük ve daha homojen gruplara bölmek için kümeleme algoritmaları kullanılabilir. Bu, verilerin daha iyi anlaşılmasını ve analizini kolaylaştırır.
- Öngörülemeyen Desenlerin Tespiti: Beklenmeyen desenler ve yapılar, kümeleme algoritmaları kullanılarak bulunabilir. Kümeleme algoritmaları, veri setindeki gizli desenler ve ilişkileri bulmak için kullanılabilir.
- Veri Azaltma: Kümeleme algoritmaları, büyük boyutlu veri setlerini azaltabilir. Benzer özelliklere sahip veri noktaları aynı kümelere atanabilir. Bu, veri setindeki gereksiz özelliklerin sayısını azaltır.

Dezavantajları:

- Başlangıç Noktasının Seçimi: Kümeleme algoritmaları başlangıç noktasını genellikle rastgele seçer ve başlangıç noktasına bağlı olarak sonuçlar değişebilir. Yanlış kümeler, yanlış başlangıç noktalarından kaynaklanabilir.
- Aykırı Değerler: Kümeleme algoritmaları aykırı değerlere hassas olabilir. Kümeleme sonuçları, yanlış değerler nedeniyle bozulabilir veya yanlış gruplara atanabilir.
- Veri Boyutları: Büyük veri seti kümeleme algoritmaları için zor olabilir. Boyutluluk arttıkça, veri setinin kümeleme doğruluğu düşebilir ve yoğunluk farkları azalabilir.
- Optimum Küme Sayısı: Kümeleme algoritmalarıyla çalışırken, ideal küme sayısını bulmak zor olabilir. Performans ve yorumlanabilirlik, doğru kümeyi seçmekle etkilenebilir.
- Ölçeklendirme Sorunu: Bazı kümeleme algoritmaları, veri setindeki özelliklerin büyüklüğüne göre ayarlanabilir. Ölçeklendirme yapılmazsa, belirli özellikler diğerlerine göre daha fazla etkileyebilir ve yanlış sonuçlara yol açabilir.

2.2.2 Gözetimli Öğrenme

Makine öğrenmesinin en yaygın yöntemlerinden biri gözetimlidir. Bu yöntem, bir modelin girdi verileriyle çıktı verilerinin nasıl ilişkili olduğunu öğrenmesine izin verir. Yeni girdi verilerine dayanarak, model bu ilişkiyi kullanarak doğru çıktıları tahmin edebilir. Gözetimli öğrenme, veriye dayalı karar verme süreçlerinde etkili bir araçtır ve birçok uygulama alanında kullanılır.

Etiketli veri tipik olarak gözetimli öğrenme sürecinin temelini oluşturur. Etiketlenmiş veri, her girdi verisinin doğru çıktısıyla eşleştirildiği verilerdir. Bu veri kümesi eğitim için kullanılır ve modelin öğrenme sürecinde kritik bir bileşendir. Örneğin, görüntü sınıflandırma problemlerinde girdi verileri görüntülerdir ve etiketler bu görüntülerin doğru sınıflarını gösterir. Bu etiketler, gözetimli öğrenme sürecinde modelin girdi verileriyle ilişkiyi öğrenmesini sağlar.

Gözetimli öğrenme sürecinde tipik olarak aşağıdaki adımlar bulunur:

- Veri Toplama: Gözetimli öğrenme için uygun bir veri kümesidir. Bilgi, öğrenme amacına uygun olarak belirli kalite ve formatta olmalıdır. Veri kaynaklarına ve sorunun ihtiyaçlarına bağlı olarak, veri toplama süreci çeşitli yöntemler kullanabilir.
- Veri Hazırlama: Model, toplanan verileri uygun bir şekilde dönüştürmelidir. Bu aşamada gereksiz verilerin temizlenmesi, eksik verilerin doldurulması veya çıkarılması ve veri normalizasyonu gibi veri işleme teknikleri kullanılır.
- Veri Bölümleme: Bir veri kümesini eğitim, doğrulama ve test bölümlerine ayırmak yaygın bir uygulamadır. Modelin öğrenme sürecinde kullanılacak bilgiler eğitim kümesi tarafından sağlanır. Modelin eğitim sırasında performansını değerlendirmek için doğrulama kümesi kullanılır. Ayrıca hiperparametreyi ayarlamak için de kullanılır. Modelin eğitimi tamamlandıktan sonra genel performansını değerlendirmek için ise test kümesi kullanılır.
- Özelliklerin Seçimi: Gözetimli öğrenme için girdi verilerinden önemli özelliklerin seçilmesi veya çıkarılması gerekir. Bu adım, modelin daha iyi öğrenmesine yardımcı olabilir ve verinin boyutunu azaltabilir. Veri analizi ve istatistiksel yöntemler kullanılarak özellik seçilebilir.
- Model Seçimi ve Eğitimi: Gözetimli öğrenme için bir dizi model türü kullanılabilir. Destek vektör makineleri, karar ağaçları, doğrusal regresyon, lojistik regresyon ve

derin sinir ağları çok popüler modellerdir. Seçilen model, eğitim verileri üzerinde çalışır ve girdi verileriyle hedef çıktılar arasındaki ilişkiyi araştırır. Bu aşamada modelin hiperparametreleri de değiştirilmelidir.

- Modelin Değerlendirilmesi: Doğrulama ve test kümeleri, eğitilen modelin performansını değerlendirir. Bu aşamada, modelin tahminlerinin doğruluğu, başarı ölçütleri ve hata metrikleri kullanılarak değerlendirilir.
- Model Ayarlanması: Modelin hiperparametreleri doğrulama kümesi üzerinde ayarlanır ve optimal edilir, böylece en iyi performansı sağlar. Bu aşamada, doğrulama kümeleri ve hiperparametre arama teknikleri üzerinde deneyler yapılır.
- Tahminler: Yeni girdi verilerine dayanarak eğitilmiş model doğru çıktıları tahmin edebilir. Bu, modelin gerçek verilerle test edilebilmesini sağlar.

Gözetimli öğrenme, birçok uygulama alanında etkili bir yöntemdir. Örneğin, metin analizi, ses tanıma, dil çevirisi ve görüntü sınıflandırma gibi birçok alanda gözetimli öğrenme teknikleri kullanılmaktadır. Veri analitiği, tahminleme, karar destek sistemleri ve yapay zeka gibi çok sayıda alanda bu teknik çok önemlidir.

Gerçek hayattaki sorunları çözmek için gözetimli öğrenme teknikleri harikadır. Bununla birlikte, veri kalitesi, veri etiketlemesi, özellik seçimi ve hiperparametre ayarlaması gibi sorunlar ortaya çıkabilir. Ek olarak, model aşırı uyma (overfitting) veya aşırı genelleme (underfitting) gibi sorunlarla karşılaşabilir. Sonuç olarak, gözetimli öğrenme başarılı olmak için veri analizi, deneyim ve uzmanlık gerekebilir.

Gözetimli öğrenme, makine öğrenmesi alanında önemli bir ilerleme kaydetmiş ve çok sayıda uygulamada başarılı olmuştur. Daha karmaşık modellerin ve veri setlerinin kullanılmasıyla birlikte gözetimli öğrenmenin etkinliği artacak ve daha da yaygınlaşacaktır.

2.2.2.1 Classification Nedir?

Makine öğreniminde sınıflandırma, veri analizi ve tahminleme sorunlarını çözmek için kullanılan bir tekniktir. Giriş verilerinin belirli sınıf etiketlerine atanması sınıflandırmanın konusudur. Örnek veri noktaları çeşitli sınıflar arasında doğru bir şekilde kategorize edilir. Veri setindeki örneklerin özellikleri analiz edilerek ve öğrenilir.

Denetimli öğrenme yaklaşımı, sınıflandırmayı içerir. Denetimli öğrenme için eğitim veri seti, özellikler olarak bilinen giriş verilerini ve doğru çıktı etiketlerini içerir. Bu eğitim veri setine

dayanarak sınıflandırma algoritması bir model veya karar sınırları oluşturur. Giriş verilerinin özelliklerine göre model sınıf tahminleri yapar.

Sınıflandırma tipik olarak iki aşamadan oluşur:

Eğitim:

- Eğitim verileri kullanılarak sınıflandırma algoritması bir model oluşturur.
- Eğitim veri seti, giriş verilerini (özellikler) ve doğru çıktı sınıf etiketlerini içerir.
- Algoritma, eğitim veri setini analiz eder ve veri noktalarının özelliklerine dayanarak sınıf ayrımını bulur.
- Sınıflandırma algoritması, sınıf etiketlerini ve özellikleri birleştiren bir model oluşturur.

Tahminler:

- Eğitim sürecinden elde edilen model, yeni ve görülmemiş verilere sınıf etiketleri atar.
- Giriş verilerinin özelliklerini modelle eşleştirerek sınıf etiketlerini tahmin etmek tahmin sürecinin bir parçasıdır.
- Model, özellikleri analiz ederek farklı sınıflara veri noktalarını atar.
- Bu şekilde, model öğrendiği sınıf ayrımını eğitim sürecinde uygulayarak doğru tahminler yapar.
- Sınıflandırma algoritmalarının çalışma prensipleri ve özellikleri çoktur. İşte sınıflandırma algoritmalarından bazıları:

2.2.2.2. Clustering Algoritmalarının İncelemesi

Karar Ağaçları:

Veri noktalarını bir ağaç yapısına göre sınıflandıran algoritma. Bir özellik testi, her düğümü dallara ayırır. Sınıf etiketleri, yaprak düğümleri ile sonuçlanır.

Destek Vektör Makineleri veya SVM'ler, veri noktalarını sınıflar arasındaki ayrımı en üst düzeye çıkarmak için bir hipers düzlem keşfeder. Sınıflandırma için destek vektörleri kullanır.

Lojistik regresyon, giriş verilerini bir veya daha fazla bağımlı değişken (sınıf etiketi) ile ilişkilendiren bir istatistiksel modeldir. Sınıf etiketini hesaplamak için lojistik fonksiyonu kullanır.

K-En Yakın Komşuluk (K-En Yakın Komşular - KNN): K en yakın komşusuna dayalı olarak veri noktalarını sınıflandıran bir algoritmadır. Benzerlik metrikleri sınıflandırmayı kolaylaştırır.

Rastgele Ormanlar: Karar ağaçlarının bir grupudur. Birden çok karar ağacı oluşturur ve sonuçları birleştirir.

Derin Öğrenme Algoritmaları (Deep Learning Algorithms): Uzun Kısa Süreli Bellek Ağları (LSTM), Evrişimli Sinir Ağları (CNN) ve Yapay Sinir Ağları gibi derin öğrenme algoritmaları, karmaşık yapıları ve büyük veri setlerini sınıflandırmak için kullanılır.

Bu algoritmalardan her biri, çeşitli veri yapıları, büyüklükleri ve özelliklerine uygun durumlarda kullanılabilir. Veri setinin özellikleri, sorunun gereksinimleri ve performans ölçütleri, sınıflandırma algoritması seçimini etkiler.

Makine öğreniminde sıklıkla kullanılan önemli bir yöntem sınıflandırmadır. İnsan sağlığından finansal tahminlere, spam filtrelemesinden görüntü tanımaya kadar çok sayıda uygulamada kullanılır. Sınıflandırma algoritmaları, verileri anlama, desenleri tespit etme ve gelecekteki olayları tahmin etme yeteneğiyle veri analizi ve tahminleme süreçlerine önemli ölçüde yardımcı olur.

2.2.2.3. Clustering Algoritmalarının Avantajları ve Dezavantajları

Avantajlar:

- Yüksek Doğruluk: İyi eğitilmiş bir sınıflandırma modeli, veri setinde yüksek bir doğruluk oranına sahip olabilir. Özellikle büyük ve temsilci veri setleriyle çalışıldığında, sınıflandırma algoritmaları doğru tahminler yapabilir.
- Hızlı Tahminleme: Sınıflandırma algoritmaları genellikle hızlı tahminleme yetenekleri sunar. Öğrenme aşamasında oluşturulan model, yeni veri noktalarını hızlı bir şekilde sınıflandırabilir. Bu, gerçek zamanlı uygulamalar ve büyük veri setleri için avantaj sağlar.

- Genel Uygulanabilirlik: Sınıflandırma algoritmaları, çeşitli veri türleri ve problemler için genel olarak uygulanabilir. Örneğin, metin sınıflandırmasından görüntü tanımaya, sayısal verilerden kategorik verilere kadar farklı veri türlerini işleyebilir.
- Yorumlanabilirlik: Bazı sınıflandırma algoritmaları, özellikle karar ağaçları gibi, sonuçları insanlar tarafından yorumlanabilir hale getirir. Modelin nasıl bir karar verdiğini anlamak ve açıklamak kolay olabilir. Bu, karar alma sürecini anlamak ve doğruluk açıklamaları yapmak için önemli olabilir.

Dezavantajlar:

- Overfitting ve Underfitting: Sınıflandırma modelleri, eğitim veri setine fazla uyum sağlayarak overfitting veya yetersiz uyum sağlayarak underfitting sorunlarına yol açabilir. Overfitting, modelin eğitim veri setine aşırı uyum sağlaması ve genelleme yeteneğinin düşük olması anlamına gelir. Underfitting ise modelin eğitim veri setinde yeterli öğrenme yapmaması ve düşük doğruluk sağlaması anlamına gelir.
- Veri Dengelemesi: Eğer sınıf etiketleri arasında dengesizlik varsa, yani bazı sınıflar diğerlerinden daha fazla örneğe sahipse, sınıflandırma sonuçları dengesizlikten etkilenebilir. Az sayıda örneğe sahip sınıfların doğru sınıflandırılması zor olabilir ve modelin genel performansını etkileyebilir.
- Duyarlılık Verisi: Bazı sınıflandırma algoritmaları, eğitim veri setindeki hatalı etiketlere veya gürültülü verilere duyarlı olabilir. Bu, modelin yanlış sınıflandırma yapabileceği anlamına gelir. Eğitim veri setinin kalitesi ve temsilciliği, modelin doğruluğunu etkileyebilir.
- Özellik Seçimi: Sınıflandırma algoritmalarının başarısı, kullanılan özelliklerin kalitesine bağlıdır. Yanlış veya önemsiz özelliklerin kullanılması, modelin performansını düşürebilir. Doğru özellik seçimi, sınıflandırma doğruluğunu artırabilir.

Bu avantajlar ve dezavantajlar, sınıflandırma algoritmalarının farklı durumlarda kullanımını etkileyebilir. Hangi algoritmanın seçileceği, veri setinin özelliklerine, problemin gereksinimlerine ve performans ölçütlerine bağlıdır.

2.3. CLUSTERİNG VE CLASSİFİCATİON ARASINDAKİ TEMEL FARKLAR

- Amaç:

Clustering: Kümeleme, veri setindeki benzerliklere dayanarak veri noktalarını gruplara ayırmayı hedefler. Temel amacı, veri içindeki doğal yapıları ve ilişkileri keşfetmek ve veri noktalarını bir araya getirilen gruplara atamaktır. Burada, etiketler veya sınıf bilgileri önceden bilinmez.

Classification: Sınıflandırma, veri noktalarını belirli kategorilere veya sınıflara atamayı hedefler. Temel amacı, veri noktalarının özelliklerini kullanarak doğru sınıf etiketlerini tahmin etmektir. Sınıf etiketleri eğitim veri setinde önceden belirlenmiş olarak bulunur.

- Öğrenme Yaklaşımı:

Clustering: Kümeleme, denetimsiz öğrenme yaklaşımını kullanır. Eğitim veri setindeki etiketler veya sınıf bilgileri önceden bilinmez. Algoritma, veri noktalarındaki benzerliklere dayanarak gruplamaları otomatik olarak belirler. Bu nedenle, veri setindeki doğal yapıyı keşfeder.

Classification: Sınıflandırma, denetimli öğrenme yaklaşımını kullanır. Eğitim veri setinde her veri noktası için doğru sınıf etiketleri veya sınıf bilgileri sağlanır. Algoritma, bu etiketleri kullanarak yeni veri noktalarının sınıf etiketlerini tahmin eder.

- Bilgi Kaynağı:

Clustering: Kümeleme algoritmaları, veri noktaları arasındaki benzerlik veya uzaklık metriklerine dayanan istatistiksel veya geometrik bilgilere dayanır. Veri noktalarının özelliklerini analiz ederek, benzer özelliklere sahip veri noktalarını gruplar.

Classification: Sınıflandırma algoritmaları, eğitim veri setindeki özelliklerin sınıf etiketleriyle olan ilişkisini öğrenir. Bu ilişkiyi kullanarak, yeni veri noktalarını sınıflandırır.

- Sonuçlar:

Clustering: Kümeleme sonuçları, veri noktalarının birbirleriyle olan benzerliklerine ve gruplamalara dayanır. Her bir küme, içerdiği veri noktalarının benzer özelliklere sahip olduğunu gösterir. Ancak, küme etiketleri veya anlamları genellikle algoritma tarafından sağlanmaz.

Classification: Sınıflandırma sonuçları, her veri noktasının doğru sınıf etiketlerini tahmin etmeye çalışır. Algoritma, yeni veri noktalarını belirli bir sınıfa atar ve bu sınıflara ait olasılıklar veya tahmin güvenirlikleri sağlayabilir.

- Veri Etiketleri:

Clustering: Kümeleme, veri setindeki doğal gruplamaları bulmaya çalışır ve veri noktalarını gruplara ayırır. Bu nedenle, etiketler veya sınıf bilgileri önceden bilinmez.

Classification: Sınıflandırma, veri noktalarını belirli sınıflara atamayı hedefler. Eğitim veri setindeki veri noktaları için doğru sınıf etiketleri veya sınıf bilgileri sağlanır.

- Kullanım Alanları:

Clustering: Kümeleme, veri keşfi, pazar segmentasyonu, görüntü işleme, örüntü tanıma gibi alanlarda kullanılır. Veri setindeki yapıyı ve benzerlikleri anlamak için kullanışlıdır.

Classification: Sınıflandırma, spam filtreleme, hastalık teşhisi, el yazısı tanıma, müşteri segmentasyonu, duygu analizi gibi alanlarda kullanılır. Veri noktalarını belirli sınıflara atamak ve sınıfları tahmin etmek için kullanışlıdır.

2.3.1. Clustering ve Classification Problem Tanımı ve Hedefleri

Clustering (Kümeleme) ve Classification (Sınıflandırma) problemleri farklı problem tanımları ve hedeflere sahiptir.

Clustering (Kümeleme):

Problem Tanımı: Clustering, veri setindeki benzerliklere dayalı olarak veri noktalarını gruplara ayırma işlemidir. Veri noktaları arasındaki benzerlik veya uzaklık ölçüleri kullanılarak veri noktaları doğal gruplara bölünür. Bu gruplar, içindeki veri noktalarının benzer özelliklere sahip olduğu yapısal veya ilişkisel birimlerdir.

Hedef: Kümeleme, veri setindeki doğal yapıları, desenleri veya ilişkileri keşfetmeyi hedefler. Kümeleme algoritmaları, veri setindeki veri noktalarını gruplandırarak farklı küme türlerini veya veri noktalarının birbirleriyle olan ilişkilerini belirlemeyi amaçlar. Bu şekilde, veri setinin daha iyi anlaşılması ve içerdiği bilginin keşfedilmesi sağlanır.

Classification (Sınıflandırma):

Problem Tanımı: Sınıflandırma, veri noktalarını belirli kategorilere veya sınıflara atama işlemidir. Sınıflandırma algoritmaları, veri noktalarının özelliklerini analiz ederek her bir veri noktasını doğru sınıf etiketiyle ilişkilendirir. Eğitim veri setindeki örneklerin doğru sınıf etiketleri kullanılarak bir model veya karar sınırları oluşturulur.

Hedef: Sınıflandırmanın temel amacı, veri noktalarını belirli sınıflara atayarak sınıf etiketlerini tahmin etmektir. Model, eğitim veri setindeki örneklerin özelliklerine dayanarak yeni veri noktalarının doğru sınıf etiketlerini tahmin etmeyi hedefler. Sınıflandırma algoritmaları, veri analizi, desen tanıma, tahminleme ve karar destek sistemleri gibi birçok alanda kullanılır.

2.3.2. Clustering ve Classification Veri Özellikleri ve Hazırlık Süreci

1. Clustering (Kümeleme):

Veri Özellikleri:

- Clustering için kullanılan veri genellikle sayısal veya kategorik özelliklere sahip olabilir.
- Veri noktaları arasındaki benzerlik veya uzaklık ölçümleri genellikle özellikler arasında hesaplanır.
- Veri setindeki örneklerin sınıf etiketleri veya hedef değerleri genellikle bilinmez.

Hazırlık Süreci:

- Veri Temizleme: Veri setinde eksik veya yanlış veriler varsa, bunları düzeltmek veya çıkarmak önemlidir.
- Özellik Seçimi: Kümeleme algoritmasının performansını etkileyebilecek özelliklerin seçimi önemlidir. İrrelevant veya düşük varyanslı özellikler çıkarılabilir.
- Özellik Ölçeklendirme: Özelliklerin farklı ölçeklerde olması durumunda, veri ölçeklendirme işlemi uygulanabilir. Bunun için normalizasyon veya standartlaştırma gibi yöntemler kullanılabilir.
- Özellik Dönüşümü: Özelliklerin uygun bir şekilde temsil edilmesi için dönüşüm işlemleri yapılabilir, örneğin, kategorik verilerin one-hot encoding veya label encoding gibi dönüşüm yöntemleriyle sayısal formata dönüştürülmesi.

2. Classification (Sınıflandırma):

Veri Özellikleri:

- Sınıflandırma için kullanılan veri genellikle sayısal veya kategorik özelliklere sahip olabilir.
- Her bir veri noktası için hedef sınıf etiketi (sınıf bilgisi) bilinmelidir.

Hazırlık Süreci:

- Veri Temizleme: Sınıflandırma için kullanılan veri seti üzerinde eksik veya yanlış verileri düzeltmek veya çıkarmak önemlidir.
- Özellik Seçimi: Sınıflandırma algoritmasının performansını etkileyebilecek özelliklerin seçimi önemlidir. İrrelevant veya düşük varyanslı özellikler çıkarılabilir.
- Özellik Ölçeklendirme: Özelliklerin farklı ölçeklerde olması durumunda, veri ölçeklendirme işlemi uygulanabilir. Bunun için normalizasyon veya standartlaştırma gibi yöntemler kullanılabilir.
- Özellik Dönüşümü: Özelliklerin uygun bir şekilde temsil edilmesi için dönüşüm işlemleri yapılabilir, örneğin, kategorik verilerin one-hot encoding veya label encoding gibi dönüşüm yöntemleriyle sayısal formata dönüştürülmesi.

2.3.3. Clustering ve Classification Algoritma Seçimi ve Uygulama

1. Clustering (Kümeleme):

Algoritma Seçimi:

- K-Means: En popüler ve temel kümeleme algoritmasıdır. Veri noktalarını belirli sayıda küme merkezi (centroid) etrafında gruplar. Veri noktalarını belirli bir uzaklık metriği kullanarak gruplara ayırır.
- Hierarchical Clustering: Veri noktalarını bir ağaç yapısı olarak gruplandırır. Agglomerative veya divisive olmak üzere iki ana yaklaşım vardır. Veri noktaları benzerliklerine göre gruplara ayrılır ve bu gruplar birleştirilerek veya bölünerek daha büyük veya daha küçük gruplara dönüştürülür.
- DBSCAN: Yoğunluk tabanlı bir kümeleme algoritmasıdır. Veri noktalarını yoğun bölgelere yerleştirir ve düşük yoğunluğa sahip bölgeleri ayırır. Esnek bir yapıya sahiptir ve küme sayısını önceden belirlemek zorunda değildir.
- Gaussian Mixture Models (GMM): Veri noktalarını olasılık dağılımlarına dayalı olarak gruplandırır. Her bir küme, bir veya daha fazla Gauss dağılımının bir karışımı olarak temsil edilir.

Uygulama Süreci:

- Veri hazırlığı: Veri seti temizlenir, eksik veriler giderilir ve özellikler uygun şekilde dönüştürülür veya ölçeklendirilir.
- Algoritma Seçimi: Veri setinin özelliklerine ve problemin gereksinimlerine bağlı olarak uygun kümeleme algoritması seçilir.
- Parametre Ayarı: Seçilen algoritmanın parametreleri, veri setine uygun şekilde ayarlanır. Örneğin, K-Means için küme sayısı belirlenir.
- Kümeleme: Seçilen algoritma ve parametreler kullanılarak veri seti kümeleme işlemi uygulanır ve veri noktaları gruplara ayrılır.
- Sonuçların Analizi: Elde edilen kümeleme sonuçları görselleştirilir ve analiz edilir. Kümeleme sonuçlarının anlamlı ve tutarlı olması hedeflenir.

2.Classification (Sınıflandırma):

Algoritma Seçimi:

- Decision Trees: Karar ağaçları, veri noktalarını bir ağaç yapısı olarak sınıflandırır. Her düğüm, bir özellik testiyle temsil edilir ve dallara ayrılır. Yaprak düğümleri sınıf etiketlerini temsil eder.
- Naive Bayes: İstatistiksel bir sınıflandırma algoritmasıdır. Bayes teoremi prensibine dayanır ve özellikler arasındaki bağımsızlığı varsayar. Önceden hesaplanmış olasılık dağılımlarını kullanarak sınıf etiketlerini tahmin eder.
- Support Vector Machines (SVM): Veri noktalarını sınıflar arasındaki ayrımı en üst düzeye çıkaran bir hipers düzlem bulmaya çalışır. Destek vektörleri kullanarak sınıflandırma yapar.
- Logistic Regression: Giriş verilerini bir veya daha fazla bağımlı değişken (sınıf etiketi) ile ilişkilendiren bir istatistiksel modeldir. Sınıf etiketini tahmin etmek için lojistik fonksiyon kullanır.

Uygulama Süreci:

- Veri hazırlığı: Veri seti temizlenir, eksik veriler giderilir ve özellikler uygun şekilde dönüştürülür veya ölçeklendirilir.

- Algoritma Seçimi: Veri setinin özelliklerine ve problemin gereksinimlerine bağlı olarak uygun sınıflandırma algoritması seçilir.
- Model Eğitimi: Seçilen algoritma ve parametreler kullanılarak veri seti üzerinde bir sınıflandırma modeli eğitilir.
- Model Doğrulama: Eğitilen model, doğrulama veri seti veya çapraz doğrulama yöntemleriyle değerlendirilir. Modelin performansı değerlendirilir ve hata analizi yapılır.
- Tahminleme: Eğitilen model, yeni veri noktalarının sınıf etiketlerini tahmin etmek için kullanılır.

2.3.4. Clustering ve Classification Sonuçların Değerlendirilmesi

Clustering (Kümeleme):

- İçsel Değerlendirme: Kümeleme sonuçlarının değerlendirilmesi için bazı içsel değerlendirme metrikleri kullanılabilir. Örneğin, küme içi benzerlikleri ölçmek için SSE (Sum of Squared Errors) veya siluet skoru kullanılabilir. Bu metrikler, kümeleme sonuçlarının kalitesini ve tutarlılığını değerlendirmeye yardımcı olur.
- Dışsal Değerlendirme: Eğer veri setinde önceden bilinen sınıf etiketleri veya hedef değerleri varsa, kümeleme sonuçları bu bilgilere göre değerlendirilebilir. Örneğin, homojenlik, tamamlık, F-skoru gibi dışsal değerlendirme metrikleri kullanılabilir. Bu metrikler, kümeleme sonuçlarının gerçek sınıflara ne kadar iyi uyduğunu ölçer.

Classification (Sınıflandırma):

- Doğruluk (Accuracy): Sınıflandırma sonuçlarının doğruluğunu değerlendirmek için genellikle doğruluk metriği kullanılır. Doğruluk, doğru sınıf etiketi tahminlerinin toplam veri noktalarına oranını ifade eder. Ancak, doğruluk tek başına yeterli olmayabilir, özellikle dengesiz sınıf dağılımına sahip veri setlerinde.
- Hassasiyet (Precision) ve Geri Çağırma (Recall): Sınıflandırma sonuçlarının hassasiyet ve geri çağırma gibi metriklerle değerlendirilmesi önemlidir. Hassasiyet,

pozitif olarak tahmin edilen örneklerin gerçekten pozitif olanların oranını ifade ederken, geri çağırma, gerçekten pozitif olan örneklerin ne kadarının doğru bir şekilde pozitif olarak tahmin edildiğini ifade eder. Bu metrikler, yanlış pozitifler ve yanlış negatifler arasındaki dengeyi değerlendirmeye yardımcı olur.

- F1-Skoru: Hassasiyet ve geri çağırma arasındaki dengeyi ölçen bir metriktir. F1-skoru, hassasiyet ve geri çağırmanın harmonik ortalamasını ifade eder ve modelin dengeli bir şekilde performansını değerlendirmeye yardımcı olur.

2.4. Clustering ve Classification Performans Ölçütleri

2.4.1. Clustering Performans Ölçütleri

Clustering performans ölçütleri, bir veri kümesinin veya bir gruplama algoritmasının başarısını değerlendirmek için kullanılan metriklerdir. Clustering, verileri benzer özelliklere sahip gruplara ayırma işlemidir ve bu ölçütler, elde edilen gruplamanın ne kadar doğru ve anlamlı olduğunu değerlendirmek için kullanılır.

- Doğruluk (Accuracy): Kümeleme sonuçlarının ne kadar doğru olduğunu ölçer. Her bir veri noktası, gerçek etiketine (eğer varsa) veya bir referans kümeleme etiketine karşı tahmin edilen kümeleme etiketiyle karşılaştırılır. Doğru tahmin edilen veri noktalarının yüzdesi, doğruluk olarak hesaplanır.
- Yarı İçsel Kümeleme Ölçüsü (Silhouette Score): Kümeleme sonuçlarının ne kadar homojen olduğunu ve veri noktalarının doğru kümelere atanıp atanmadığını ölçer. Silhouette skoru, her veri noktasının ait olduğu kümeye olan benzerliği ve diğer kümeler arasındaki farklılığı dikkate alır. Silhouette skoru, -1 ile 1 arasında bir değer alır, daha yüksek değerler daha iyi bir kümeleme performansını gösterir.

- Çalışma Süresi (Runtime): Kümeleme algoritmasının çalışma süresini ölçer. Büyük veri setlerinde veya karmaşık algoritmalarda, hızlı ve verimli bir kümeleme algoritması önemlidir.
- SSE (Sum of Squared Errors): Küme merkezleri ile veri noktaları arasındaki mesafelerin karelerinin toplamını hesaplar. SSE, kümeleme sonuçlarının ne kadar homojen olduğunu ölçer. Daha düşük SSE değerleri, daha iyi bir kümeleme performansını gösterir.
- Rand Endeksi (Rand Index): Bir referans kümeleme etiketiyle tahmin edilen kümeleme etiketleri arasındaki uyumu ölçer. Rand endeksi, doğru olarak eşleştirilen veri noktalarının yüzdesini hesaplar. Değer 1'e yakınsa, kümeleme sonuçları referans kümelemeyle iyi bir uyum içindedir.
- Normalleştirilmiş Karşılaştırma Kümeleme Ölçütü (Normalized Mutual Information, NMI): Kümeleme sonuçları ile referans kümeleme arasındaki bilgi paylaşımını ölçer. NMI, kümeleme sonuçlarının referans kümelemeyle ne kadar uyumlu olduğunu gösterir. Değer 1'e yakınsa, kümeleme sonuçları referans kümelemeyle yüksek bir uyum içindedir.

- Rand Geçerlilik İndeksi (Rand Index): Kümeleme sonuçlarının bir referans kümeleme ile uyumunu ölçer. Gerçek ve tahmin edilen kümeleme etiketleri arasındaki uyum oranını hesaplar. Değer 1'e yakınsa, kümeleme sonuçları referans kümelemeyle uyumludur.
- Düzeltilmiş Rastgelelik İndeksi (Adjusted Rand Index, ARI): Rand Endeksi'ne benzer şekilde çalışır, ancak rastgelelik faktörünü düzeltmek için bir düzeltme faktörü uygular. ARI, kümeleme sonuçlarının referans kümelemeyle uyumunu ölçer ve değeri 1'e yaklaştıkça daha iyi bir uyumu gösterir.
- Homojenlik ve Tamamlık (Homogeneity and Completeness): Homojenlik, her gerçek sınıfın yalnızca bir kümede temsil edilip edilmediğini ölçerken, tamamlık, her bir kümenin yalnızca bir gerçek sınıfa ait verileri içerip içermediğini ölçer. Her ikisi de 0 ile 1 arasında değer alır, daha yüksek değerler daha iyi bir performansı gösterir.
- F-Measure: Doğruluk ve tamamlık ölçümlerinin harmonik ortalamasını temsil eder. F-Measure, kümeleme sonuçlarının hem homojenlik hem de tamamlık açısından nasıl performans gösterdiğini değerlendirmek için kullanılır.

- Küme Sayısı Seçimi Ölçütleri: Küme sayısını belirlemek için kullanılan ölçütlerdir. Örneğin, Dirichlet Bayesian Bilgi Kriteri (Dirichlet Bayesian Information Criterion, BIC) ve Elbow yöntemi küme sayısı seçimi için yaygın olarak kullanılan yaklaşımlardır.
- Dışsal Validite Ölçütleri: Kümeleme sonuçlarının harici bir kriterle (örneğin, etiketlenmiş veri) ne kadar uyumlu olduğunu değerlendiren ölçütlerdir. Örneğin, Adjusted Mutual Information (AMI) ve Rand Index gibi ölçütler dışsal validiteyi değerlendirmek için kullanılabilir.

2.4.2. Classification Performans Ölçütleri

Sınıflandırma performans ölçütleri, bir sınıflandırma modelinin başarısını değerlendirmek için kullanılan metriklerdir. Sınıflandırma, verileri belirli sınıflara atama işlemidir ve bu ölçütler, modelin ne kadar doğru ve güvenilir bir şekilde sınıflandırma yaptığını değerlendirmek için kullanılır.

- Doğruluk (Accuracy): Sınıflandırma algoritmasının doğru tahmin ettiği veri noktalarının yüzdesini ölçer. Toplam doğru tahmin edilen veri noktalarının toplam veri noktalarına oranı olarak hesaplanır. Doğruluk, temel bir performans ölçütüdür ancak dengesiz veri setlerinde yanıltıcı olabilir.
- Hassasiyet (Precision): Pozitif olarak tahmin edilen veri noktalarının gerçek pozitiflere oranını ölçer. Yanlış pozitiflerin sayısını azaltmayı hedefler. Hassasiyet, yanlış pozitiflere karşı duyarlı olduğu için yanlış pozitiflerin önemli olduğu durumlarda önemli bir ölçüttür.
- Geri Çağırma (Recall): Gerçek pozitiflerin doğru tahmin edilen pozitiflere oranını ölçer. Yanlış negatiflerin sayısını azaltmayı hedefler. Geri çağırma, yanlış negatiflerin önemli olduğu durumlarda önemli bir ölçüttür.
- F1 Skoru: Hassasiyet ve geri çağırmanın harmonik ortalamasını temsil eder. Dengesiz veri setleri veya sınıflar arasında doğru bir denge sağlamak için kullanılır. F1 skoru, hem hassasiyeti hem de geri çağırma oranını göz önünde bulundurarak sınıflandırma performansını değerlendirir.
- Hassaslık (Specificity): Negatif olarak tahmin edilen veri noktalarının gerçek negatiflere oranını ölçer. Yanlış negatifleri azaltmayı hedefler. Özellikle yanlış negatiflerin önemli olduğu durumlarda kullanılır.

- ROC Eğrisi (Receiver Operating Characteristic Curve) ve AUC (Area Under the Curve): Sınıflandırma algoritmasının duyarlılık (recall) ve özgüllük (specificity) arasındaki performans ticaretini görselleştirir. ROC eğrisi, farklı sınırlayıcı (threshold) değerlerinde sınıflandırma algoritmasının performansını gösterir. AUC, ROC eğrisinin altındaki alanı temsil eder ve 1'e yaklaştıkça daha iyi bir performansı gösterir.
- Log Kaybı (Log Loss): Sınıflandırma algoritmasının tahminlerinin gerçek etiketlerle ne kadar uyumlu olduğunu ölçer. Daha düşük log kaybı değerleri daha iyi bir performansı gösterir. Özellikle olasılık tabanlı sınıflandırma algoritmalarında kullanılır.
- F-Beta Skoru: Hassasiyet (precision) ve geri çağırma (recall) arasındaki dengeyi kontrol etmek için kullanılır. F-Beta skoru, hassasiyet ve geri çağırma arasındaki ağırlığı kontrol eden bir beta parametresine sahiptir. F1 skoru, beta parametresinin 1 olduğu özel bir durumu temsil eder.
- Hamarı İzleme Ölçütü (Hamming Loss): Etiketlenmiş çok sınıflı sınıflandırma problemleri için kullanılır. Hamming loss, her bir veri noktasının doğru etiketlere sahip olup olmadığını ölçer. Daha düşük Hamming loss değerleri daha iyi bir performansı gösterir.
- Ağırlıklı İzleme Ölçütü (Weighted Accuracy): Dengesiz sınıflandırma problemlerinde kullanılır. Farklı sınıfların doğru sınıflandırılma oranlarına göre ağırlıklandırılmış bir doğruluk ölçüsüdür. Sınıflar arasındaki dengesizliği dikkate alır ve performansı objektif bir şekilde değerlendirmeye yardımcı olur.
- ROC ve AUC (Area Under the Curve): Sınıflandırma algoritmasının doğruluk ve yanlış pozitif oranı arasındaki ticareti değerlendirmek için kullanılır. ROC eğrisi, farklı sınırlayıcı (threshold) değerlerinde algoritmanın performansını gösterir. AUC, ROC eğrisinin altındaki alanı temsil eder ve 1'e yaklaştıkça daha iyi bir performansı gösterir.
- Cohen's Kappa: Etiketlenmiş veri setlerinde rastgele tahminin ötesindeki sınıflandırma doğruluğunu ölçer. Cohen's Kappa, algoritmanın rastgele tahminden ne kadar iyi olduğunu gösterir. Değer 1'e yakınsa, tahminler gerçek etiketlere daha yakın demektir.
- Bu performans ölçütleri, sınıflandırma algoritmalarının sonuçlarını objektif bir şekilde değerlendirmek ve farklı algoritmaların veya parametrelerin karşılaştırmasını yapmak için kullanılır. Seçilecek olan ölçütler, veri setinin özelliklerine, sınıflandırma probleminin gereksinimlerine ve analiz hedeflerine bağlı olarak değişebilir.

2.4.3. Classification ve Clustering Performans Ölçütlerinin Karşılaştırılması ve Yorumlanması

Amaç:

Classification: Sınıflandırma performans ölçütleri, bir modelin verilen örnekleri doğru bir şekilde sınıflandırma yeteneğini değerlendirmek için kullanılır. Sınıflandırma modelleri genellikle eğitim verilerine dayalı olarak yeni örnekleri belirli sınıflara atamak için kullanılır.

Clustering: Clustering performans ölçütleri, bir gruplama algoritmasının veri kümesini benzer özelliklere sahip gruplara ayırma yeteneğini değerlendirmek için kullanılır. Clustering algoritmaları, belirli bir kriter veya mesafe ölçütüne göre benzerliklerine göre örnekleri gruplandırır.

Hedef:

Classification: Sınıflandırma performans ölçütleri, modelin doğruluğunu, hassasiyetini, duyarlılığını ve F1-skorumu değerlendirir. Bu ölçütler, modelin pozitif ve negatif sınıfları doğru bir şekilde tahmin etme yeteneğini ölçer.

Clustering: Clustering performans ölçütleri, gruplar arasındaki homojenliği, tamamlığı, ayırma skorunu ve siluet katsayısını değerlendirir. Bu ölçütler, gruplar arasındaki benzerlik ve grup içi birlikte tutulma düzeyini ölçer.

Değerlendirme:

Classification: Sınıflandırma performans ölçütleri, eğitim verileri üzerinde doğruluk sağlama yeteneğini değerlendirir. Modelin yanlış pozitif ve yanlış negatif oranlarını gösterir. Bu ölçütler, sınıflandırma probleminin özelliklerine bağlı olarak yorumlanmalıdır. Örneğin, hassasiyetin önemli olduğu durumlarda yanlış pozitiflerin minimize edilmesi önemlidir.

Clustering: Clustering performans ölçütleri, gruplar arasındaki homojenlik ve ayırma düzeyini değerlendirir. Homojenlik, grupların benzerlik açısından ne kadar tutarlı olduğunu gösterirken, ayırma skoru gruplar arasındaki farklılığı belirler. Bu ölçütler, veri kümesinin doğasına ve gruplama hedeflerine göre yorumlanmalıdır.

Uygulama:

Classification: Sınıflandırma performans ölçütleri, genellikle denetimli öğrenme algoritmalarının değerlendirilmesinde kullanılır. Örneğin, tıbbi teşhis, spam filtreleme, görüntü sınıflandırma gibi alanlarda sınıflandırma performansı önemlidir.

Clustering: Clustering performans ölçütleri, veri keşfi, pazar segmentasyonu, sosyal ağ analizi gibi alanlarda kullanılır. Gruplamanın veri kümesinde anlamlı yapıları ortaya çıkarma ve benzer örnekleri bir araya getirme amacı vardır.

2.5. Clustering ve Classification Uygulama Alanları

2.5.1. Clustering Uygulama Alanları

Kümeleme yöntemleri, çeşitli alanlarda kullanılarak farklı amaçlara hizmet eder.

Pazar Segmentasyonu: Pazarlama stratejileri için müşterileri belirli gruplara ayırmak önemlidir. Kümeleme, müşterilerin demografik verileri, satın alma alışkanlıkları veya tercihleri gibi kriterlere dayanarak farklı pazar segmentlerini tanımlamak için kullanılabilir. Bu, pazarlamacıların müşteri ihtiyaçlarını daha iyi anlamalarına ve özelleştirilmiş pazarlama kampanyaları oluşturmalarına yardımcı olur. Örneğin, bir perakende şirketi, müşterilerini belirli segmentlere ayırarak, her bir segment için farklı promosyonlar ve ürün önerileri sunabilir.

Müşteri Sadakati Analizi: Kümeleme, müşteri sadakati analizi için kullanılabilir. Bir şirket, müşterilerini belirli gruplara ayırarak, sadakat düzeyine, satın alma alışkanlıklarına ve diğer davranış özelliklerine göre farklı grupları tanımlayabilir. Bu gruplar üzerinde yapılan analizler, müşteri sadakati stratejilerinin belirlenmesine ve müşteri kaybının azaltılmasına yardımcı olabilir.

Sağlık Verileri Analizi: Kümeleme, sağlık verileri analizinde de kullanılır. Örneğin, genetik verilerin analizinde kümeleme yöntemleri, genetik profillere dayalı olarak benzer özelliklere sahip bireyleri gruplandırabilir. Bu gruplar, belirli genetik özelliklere veya hastalıklara yatkınlığı olan bireyleri tanımlamak için kullanılabilir.

Coğrafi Veri Analizi: Kümeleme, coğrafi verilerin analizinde de yaygın olarak kullanılır. Örneğin, bir şehirdeki nüfusu belirli demografik özelliklere göre gruplandırarak, sosyoekonomik farklılıkları veya nüfus yoğunluğunu inceleyebiliriz. Bu bilgiler, şehir planlaması, kaynak tahsisi veya hizmet sunumunda kullanılabilir.

İnternet ve Sosyal Medya Analizi: İnternet ve sosyal medya analizinde kümeleme yöntemleri, kullanıcıları benzer ilgi alanlarına sahip gruplara ayırmak için kullanılabilir. Bu gruplar,

pazarlama kampanyalarının hedef kitlelerinin belirlenmesinde veya içerik öneri sistemlerinde kullanılabilir.

Müşteri Segmentasyonu: Perakende sektöründe, müşterilerin alışveriş tercihlerine göre gruplandırılması ve belirli segmentlere ayrılması önemlidir. Kümeleme, müşteri segmentasyonu için kullanılan etkili bir yöntemdir. Örneğin, bir perakende şirketi, müşterilerini fiyat duyarlılığına, marka sadakatine veya ürün tercihlerine göre farklı gruplara ayırarak pazarlama stratejilerini optimize edebilir.

Görüntü İşleme: Kümeleme, görüntü işleme alanında da yaygın olarak kullanılır. Görüntü kümeleme, benzer özelliklere sahip pikselleri gruplayarak görüntülerdeki yapıyı ortaya çıkarır. Bu, görüntü sınıflandırma, nesne tespiti veya görüntü segmentasyonu gibi diğer görüntü işleme görevlerinin temelini oluşturabilir.

Betimleyici Veri Analizi: Kümeleme yöntemleri, betimleyici veri analizinde kullanılarak veri setlerindeki yapıyı anlamak için kullanılır. Veri setindeki benzer özelliklere sahip veri noktalarını gruplandırarak, veri setindeki farklı kümeleri belirleyebiliriz. Bu, veri setindeki trendleri, dağılımları ve ilişkileri daha iyi anlamamıza yardımcı olur.

İşaret İşleme: İşaret işleme alanında kümeleme yöntemleri, benzer özelliklere sahip işaretleri gruplandırmak için kullanılabilir. Örneğin, konuşma tanıma sistemlerinde, benzer özelliklere sahip ses sinyalleri kümeleme algoritmaları kullanılarak gruplandırılabilir. Bu, konuşma tanıma doğruluğunu artırabilir ve farklı konuşmacıları veya konuşma türlerini tanımlamak için kullanılabilir.

Hava Durumu Tahmini: Kümeleme yöntemleri, meteoroloji alanında hava durumu tahmininde kullanılabilir. Benzer özelliklere sahip hava koşullarını gruplayarak, belirli hava durumu modelleri ve eğilimlerini tanımlayabiliriz. Bu, hava durumu tahmin modellerinin geliştirilmesine ve daha doğru tahminler yapılmasına yardımcı olabilir.

Finansal Analiz: Kümeleme yöntemleri, finansal analizde de kullanılabilir. Örneğin, hisse senedi piyasasında benzer özelliklere sahip hisse senetlerini gruplandırarak, belirli hisse senedi segmentlerini tanımlayabiliriz. Bu, portföy yönetimi, risk analizi ve piyasa trendlerinin analizi gibi finansal kararlar için değerli bilgiler sağlayabilir.

Biyoinformatik: Kümeleme yöntemleri, biyoinformatik alanında da kullanılır. Genomik verilerin analizinde, benzer gen ifade profillerine sahip genler veya genlerin gruplandırılması

için kümeleme algoritmaları kullanılabilir. Bu, gen ifadesi paternlerini anlamamıza ve belirli gen gruplarını tanımlamamıza yardımcı olur.

İnternet Trafik Analizi: Kümeleme yöntemleri, internet trafiği analizinde de kullanılır. Benzer özelliklere sahip ağ trafiği verilerini gruplandırarak, belirli ağ trafiği desenlerini ve anomali durumlarını tanımlayabiliriz. Bu, ağ güvenliği, hizmet kalitesi optimizasyonu ve ağ yönetimi için önemli bir araçtır.

Müşteri Hedefleme: Kümeleme, müşteri hedefleme stratejileri için kullanılabilir. Benzer özelliklere sahip müşterileri gruplandırarak, belirli hedef kitlelere yönelik pazarlama kampanyaları oluşturabiliriz. Bu, pazarlama bütçesini optimize etmeye yardımcı olur ve müşteri çekme ve elde tutma stratejilerini geliştirir.

İlaç Keşfi: Kümeleme yöntemleri, ilaç keşfi alanında da yaygın olarak kullanılır. Benzer kimyasal veya biyolojik özelliklere sahip bileşikleri gruplandırarak, potansiyel ilaç adayları veya terapötik hedeflerin keşfinde yardımcı olur. Bu, ilaç keşfi sürecini hızlandırır ve daha etkili tedavilerin geliştirilmesine katkı sağlar.

İnsan Kaynakları Yönetimi: Kümeleme yöntemleri, insan kaynakları yönetimi alanında da kullanılabilir. Benzer özelliklere sahip çalışanları gruplandırarak, işe alım sürecini optimize edebilir, ekip oluşumlarını yönetebilir ve performans değerlendirmelerini yapabiliriz. Bu, işletmelerin çalışanlarına daha iyi odaklanmalarını ve kaynaklarını daha etkili bir şekilde yönetmelerini sağlar.

Sosyal Ağ Analizi: Kümeleme yöntemleri, sosyal ağ analizinde de kullanılır. Benzer sosyal ağ profillerine sahip kullanıcıları gruplandırarak, belirli sosyal grupları veya toplulukları tanımlayabiliriz. Bu, sosyal ağların yapısal analizi, etkileşim modellemesi ve kullanıcı davranışının anlaşılması için önemli bir araçtır.

Jeoloji ve Yer Bilimleri: Kümeleme yöntemleri, jeoloji ve yer bilimlerinde de kullanılır. Benzer jeolojik özelliklere veya yer şekillerine sahip bölgeleri gruplandırarak, jeolojik formasyonları ve doğal kaynak potansiyelini anlamamıza yardımcı olur. Bu, yer bilimleri araştırmalarında önemli bir analitik araçtır.

2.5.2. Clustering Uygulama Alanları

Spam Filtrasyonu: E-posta hizmet sağlayıcıları ve mesajlaşma uygulamaları, spam filtresi olarak sınıflandırma tekniklerini kullanır. Sınıflandırma algoritmaları, gelen e-postaları spam veya gerçek e-posta olarak etiketleyerek kullanıcının gelen kutusunu düzenler.

Hastalık Teşhisi: Tıp alanında sınıflandırma, hastalık teşhisi ve prognoz tahmini gibi önemli bir rol oynar. Bir hastanın semptomlarını ve test sonuçlarını kullanarak, sınıflandırma algoritmaları hastalıkları teşhis edebilir ve tedavi planlarını önerir.

Müşteri Segmentasyonu: Pazarlama alanında, müşteri segmentasyonu sınıflandırma yöntemleriyle yapılır. Müşteriler, demografik özellikler, davranışlar veya tercihler gibi faktörlere göre farklı segmentlere ayrılır. Bu segmentasyon, pazarlama stratejilerini belirlemek ve müşterilere özelleştirilmiş teklifler sunmak için kullanılır.

Görüntü Sınıflandırması: Görüntü tanıma ve sınıflandırma, otomasyon, güvenlik, tıp ve diğer birçok alanda kullanılır. Örneğin, nesne tanıma uygulamaları, fotoğraflardaki nesneleri otomatik olarak tanır ve sınıflandırır.

Duygu Analizi: Sosyal medya analitiği veya müşteri hizmetleri gibi alanlarda, metin tabanlı duygu analizi sınıflandırma algoritmalarıyla gerçekleştirilir. Bu yöntem, metinlerdeki duygusal durumları (olumlu, olumsuz veya tarafsız) belirlemek için kullanılır.

Finansal Risk Değerlendirmesi: Finans sektöründe, kredi başvuruları ve yatırım kararları gibi risk değerlendirmeleri sınıflandırma yöntemleriyle yapılır. Algoritmalar, müşterilerin risk profilini belirlemek ve riskli işlemleri tespit etmek için kullanılır.

Toplumsal Medya Analizi: Sosyal medya platformlarında, kullanıcıların mesajlarını veya içeriklerini sınıflandırmak için algoritmalar kullanılır. Bu sınıflandırma, trendleri belirlemek, kullanıcı davranışını analiz etmek ve içeriği kişiselleştirmek için önemli bir rol oynar.

Ürün Sınıflandırması: E-ticaret platformları, müşterilere ürünleri daha kolay bulmalarını sağlamak için sınıflandırma algoritmalarını kullanır. Ürünleri doğru kategorilere yerleştirerek, kullanıcıların aradıkları ürünleri daha hızlı bulmalarını sağlar.

Hedefleme ve Kişiselleştirme: Pazarlama kampanyalarında, sınıflandırma algoritmaları, kullanıcıların ilgi alanlarına ve davranışlarına dayalı olarak hedefleme ve kişiselleştirme yapar. Bu şekilde, kullanıcılara daha uygun içerik ve teklifler sunulur.

Belge Sınıflandırması: Metin belgelerini kategorilere ayırmak için sınıflandırma algoritmaları kullanılır. Örneğin, bir belgenin haber, makale veya rapor gibi farklı kategorilere ait olup olmadığını belirlemek için kullanılabilir.

Müşteri Hizmetleri Yönetimi: Müşteri hizmetleri alanında, sınıflandırma algoritmaları müşteri taleplerini otomatik olarak sınıflandırabilir. Bu sayede, taleplerin aciliyet derecesine veya doğru departmana yönlendirilmesine yardımcı olur.

Suç Analitiği: Polis departmanları ve güvenlik birimleri, suç analitiği için sınıflandırma algoritmalarını kullanır. Örneğin, güvenlik kameraları tarafından kaydedilen görüntüleri analiz ederek, şüpheli kişileri veya olayları tanımlamak için kullanılabilir.

Biyomedikal Uygulamalar: Biyomedikal araştırmalarda, sınıflandırma algoritmaları hastalıkları teşhis etmek, genetik verileri analiz etmek veya ilaç keşfi gibi alanlarda kullanılır.

Mühendislik ve Üretim: Mühendislik ve üretim süreçlerinde, sınıflandırma algoritmaları hataları veya kalite problemlerini tespit etmek için kullanılır. Örneğin, bir üretim hattında oluşan arızaları veya anormallikleri tespit etmek için sensör verilerini analiz edebilir.

2.6. Clustering ve Classification Alanlarında Yapılmış Bilimsel Çalışmalar

Bu alanda clustering ve classification algoritmalarının kullanarak yapılmış çalışmaları inceleyeceğiz.

2.6.1. Makine Öğrenmesi Algoritmaları ile Hava Kirliliği Tahmini Üzerine Karşılaştırmalı Bir Değerlendirme

Günün en büyük sorunlarından biri hava kirliliğidir. Endüstrinin gelişmesiyle birlikte hava kirliliği, nüfusun artması, kentsel gelişme ve büyüme giderek daha önemli hale geliyor. Hava kirleticilerinin insanlara, canlılara ve çevreye zararlı etkileri genellikle zaman, mekan, etki süresi, konsantrasyon ve diğer faktörlere bağlı olarak farklı şekillerde dağılmaktadır. Kirletici örnekleri ve eğilimleri modelleme veya ölçme konusundaki bu karmaşıklık, insanların maruz kaldığı seviyeleri tahmin etmeyi de zorlaştırıyor. Hava kirliliğini önleme çalışmalarının en önemli aşamalarından biri, hava kirliliğini bir model aracılığıyla değerlendirmektir. Bu çalışmada Kastamonu ilinde hava kirliliğinin tahmininde bazı meteorolojik değişkenler

kullanılarak modeller geliştirilmiştir. Bu algoritmalar, meteoroloji ve çevre uygulamalarında oldukça yeni ve başarılı sonuçlar elde etti. Öğrenciler, Minimum-Maksimum (Min-Max) normalizasyon tekniğini kullanmıştır. Tahmin modellerinde Rastgele Orman (Random Forest), K-En Yakın Komşu (K-Nearest Neighborhood), Lojistik Regresyon (Logistic Regression), Lineer Regresyon (Linear Regression), Karar Ağacı (Decision Tree), Yapay Sinir Ağları (YSA) ve Basit Bayes kullanılmıştır. Çalışma, problemin çözümüne ilişkin en iyi tahmin algoritmasını belirlemek için performans değerlerini karşılaştırdı. Veri setinin yüzde yetmişi eğitim ve yüzde otuzu test verisi olarak ayrılmıştır. Çalışma sonucunda, YSA modeli doğru tahmin oranı %87 ile en başarılı tahmin modeliydi. Diğer makine öğrenmesi modellerinden Rastgele Orman doğruluk oranı %99 ve Karar Ağacı doğruluk oranı %99 idi. Lineer Regresyon yöntemi, %30 doğruluk oranı ile oldukça yetersizdir. Kastamonu veri setinde kullanılan yöntemler arasında, Açıklayıcılık Katsayısı (R2), Ortalama Karesel Hata (Mean Squared Error-MSE), Ortalama Hata Kare Kökü (Root Mean Squared Error-RMSE) ve Ortalama Mutlak Hata (Mean Absolute Error) metrikleri açısından önemli istatistiksel farklılıklar bulunmuştur.

2.6.2. K-Means Algoritması İle Otomatik Kümeleme

K-Means kümeleme algoritması verileri K giriş parametre sayısı kadar kümeye bölmektedir. Bu çalışmanın amacı ise kümelemeyi otomatik hale getirmek ve dışarıdan K parametresinin girilmesine gerek kalınmadan verileri uygun küme sayısınca kümelere yerleştirmektir. Geliştirilen otomatik K-Means algoritması sayısal veriler ve görüntüler üzerinde test edilmiş ve başarılı sonuçlara ulaşılmıştır.

2.6.3. Düzce İlinin Hayvansal Atıklardan Üretilebilecek Biyogaz Potansiyeli ve K-Means Kümeleme İle Optimum Tesis Konumunun Belirlenmesi

Fosil yakıtların sınırlı olması ve çevreye verdiği zarar nedeniyle, nüfus arttıkça yeni enerji kaynakları arayışı başladı. Bu nedenle sürekli, yenilenebilir ve çevreye zararsız enerji kaynaklarının önemi artmıştır. Anaerobik sindirim veya biyolojik maddelerin fermantasyonu, biyogazı üretmek için iki yöntemdir. Fermente olmuş gübre, diğer enerji türlerinden daha değerlidir ve ısı değeri yüksektir. Bu çalışmanın amacı, Düzce il ve ilçelerinde hayvansal atıklardan biyogaz üretme potansiyelini belirlemektir. Ayrıca, tesisleri konumlarına göre K-means kümeleme algoritması kullanılarak kümelere ayırmaktır. Daha sonra, her tesisin Euklid uzaklıkları toplamı en küçük değeri veren (tüm tesislere en yakın) tesis konumunu bulmak için K kümesini tek bir kümeye kümelenmektir. Bu nedenle, 2013 yılı Türkiye İstatistik Kurumu verilerini dikkate almıştır.

2.6.4. Makine Öğrenmesi Algoritmaları Kullanılarak İtfaiye İstasyonu İhtiyacının Sınıflandırılması

Kalabalık şehirlerde itfaiye istasyonlarının doğru yer seçimi, yangınlara hızlı müdahale etmek ve can ve mal kaybını en aza indirmek için çok önemlidir. İtfaiye istasyonu yer seçiminde, kentin tamamını belirli bölgelere ayırarak, her bölge için itfaiye istasyonu ihtiyacı sorgulanmalıdır. Bu çalışmada, mevcut itfaiye istasyonlarından yola çıkarak makine öğrenmesi algoritmaları kullanarak itfaiye istasyonlarının bölgelere göre ihtiyaç duydukları sınıflandırma da yapılmıştır. Çalışma, itfaiye istasyonlarının ihtiyacını tahmin etmek için her bir bölgeye ait itfaiye araçlarının ulaşım süreleri, bölgenin nüfus yoğunluğu, o bölgeye giden ortalama ana ve yardımcı araç sayısı ve itfaiye istasyonu bulunma durumu bilgilerini kullanarak sınıflandırma çalışması gerçekleştirdi. Bu çalışmanın amacı, İzmir Büyükşehir Belediyesinin belirlediği 808 bölgeye ait itfaiye istasyonlarının ihtiyaçlarını sınıflandırmak için en iyi algoritmayı bulmak. 2015 ile 2017 yılları arasındaki yangın kayıtlarının analizi, Random Forest algoritmasının bölgeleri sınıflandırmada %93.84 ile en başarılı olduğunu gösterdi. En iyi algoritmanın belirlenmesinde doğruluk, kök hata kareler ortalaması (RMSE), ortalama mutlak hata (MAE) ve Kappa değerleri dikkate alınmıştır.

2.6.5. DBSCAN, OPTICS ve K-Means Kümeleme Algoritmalarının Uygulamalı Karşılaştırılması

Bu çalışmada, veri madenciliğinde güncel kümeleme algoritmalarından DBSCAN, OPTICS ile geçmişi daha eskilere dayanan K-means algoritması karşılaştırılmıştır. Karşılaştırma sentetik veritabanı üzerinde gösterdikleri küme bulma performansları değerlendirilerek yapılmıştır. Sonuçta, yakın zamanda literatüre giren DBSCAN ve OPTICS algoritmalarının Kmeans algoritmasından daha üstün küme oluşturma özelliklerine sahip olduğu tespit edilmiştir.

2.6.6. Makine Öğrenmesi Algoritmaları Kullanarak Gişehasılatının Tahmini

Yeni efektler ve 3 boyutlu çekimler gibi güncel gelişmeler film endüstrisindeki rekabeti arttırmaktadır. Film endüstrisindeki pahalı ve riskli yatırımlar için üretim öncesi analizler giderek önem kazanmaktadır. Bu noktada, gişe hasılatı tahmini önemli bir araştırma konusu olmuştur. Bu bağlamda, bu çalışma gişe hasılatı tahmini için makine öğrenmesi algoritmaları kullanarak bir yaklaşım sunmayı amaçlamaktadır. Geleneksel yapay zeka metotlarından yapay sinir ağları ve destek vektör makineleri algoritmaları, karar ağaçları algoritmalarından rastgele ağaç, rastgele orman ve C4.5 algoritmaları kullanılmıştır. Daha sonra, bu algoritmalar ile topluluk algoritmalarından torbalama algoritması kullanılarak melez bir model

önerilmiştir. Tahmin modelleri doğru sınıflandırma yüzdesi, kappa istatistiği, ROC alanı ile değerlendirilmiştir.

2.6.7. Makine Öğrenmesi Algoritmaları Kullanılarak Kayısı İç Çekirdeklerinin Sınıflandırılması

Kayısı çekirdeği üretimi ve tüketimi fazla olan bir gıda ürünüdür. Kayısı iç çekirdeklerinin makine öğrenmesi algoritmaları kullanılarak, tatlı veya acı olarak sınıflandırılması bu çalışmanın konusunu oluşturmaktadır. Hem tatlı hem de acı kayısı iç çekirdeği için talep miktarı oldukça fazladır. Depolama şartları gibi nedenlerden dolayı kayısı iç çekirdekleri zaman zaman birbirine karışabilmektedir. Bu durum tüketiciler tarafından istenmeyen bir durumdur. Kayısı iç çekirdeğinin ayrıştırılması, gözle her zaman mümkün olmamaktadır. Bu çalışmanın amacı, insan faktörünü ortadan kaldırabilecek bir sınıflandırma yönteminin geliştirilmesidir. Bu sınıflandırma işlemi için k En Yakın Komşu, Destek Vektör Makinesi, Karar Ağacı, Rasgele Orman, Adaptive Boosting, Gaussian Naive Bayes ve Çok Katmanlı Algılayıcı algoritmaları kullanılmıştır. Yeterli sayıda öznitelik ile algoritmaların yarıdan fazlası sınıflandırma işlemini %100 başarı ile elde edebilmektedir. En az sayıda öznitelik kullanarak en iyi başarı Rasgele Orman algoritması ile elde edilmiştir. Sonuçlar göstermiştir ki, kayısı iç çekirdeklerinin sınıflandırılması işlemi makine öğrenmesi algoritmaları ile başarılı bir şekilde gerçekleştirilebilmektedir

2.6.8. Öğrencilerin Dersteki Niteliklerinin Makine Öğrenmesi Teknikleri Kullanılarak Sınıflandırılması

Öğrencilerin akademik başarılarını tahmin etme ve eksik oldukları alanları giderme anlamında yapılan bu çalışma, Bilişim Sistemleri Mühendisliğine Giriş dersi alan öğrencilere uygulanmıştır. Bu öğrencilerin dönem başı bilgisayar bilgi düzeylerinin, dönem sonunda elde ettikleri başarı notu üzerine etkisi makine öğrenmesi yöntemleri uygulanarak eğitim kalitesinin arttırılması amaçlanmıştır. Çalışmaya katılan öğrencilere ait veriseti eğitim ve test verisi olmak üzere ayrıldığında veri yetersizliğinden dolayı anlamsız sonuçlar ortaya çıkmıştır. Bu nedenle makine öğrenmesi algoritmalarının başarımını arttırmak için “Sentetik Azınlık Örneklem Arttırma Yöntemi (SMOTE)” çalışmada veri çoğaltma tekniği olarak seçilmiştir. Veri çoğaltma işlemi yapıldıktan sonra, veri seti üzerinde uygulanan K-en yakın komşu (KNN), Destek vektör makinesi (DVM), Lojistik Regresyon (LR), Rasgele Orman (RF), Karar ağaçları (DT) ve Naive Bayes makine öğrenmesi yöntemlerine göre en iyi sonucu en yakın komşuluk- KNN algoritması ile oluşturulmuş model vermiştir. Bu model, eğitim

setinden bağımsız 300 öğrenciden oluşan test verisinin sınıflandırma işlemini, %97.66 doğrulukla tahmin etmiştir.

2.6.9. Makine Öğrenmesi Algoritmaları İle Satış Tahmini

Günümüz dijital dünyasında satın alma gittikçe arttığından veriler çok büyük boyutlara ulaşmıştır. Endüstrinin getirdiği kavramlardan en belirgini ise boyutluluk laneti olmuştur. Bu sebeple işletmeler satın alma kararlarını alırken büyük zorluk yaşamaktadır. Uzun ya da kısa vadede satış tahmininin doğru yapılamaması müşteri memnuniyetsizliği, para kaybı, ham madde ihtiyacı gibi birçok soruna yol açacaktır. Tedarik zinciri elemanlarından üretici, perakendeci, tedarikçi ve müşteriye kadar birçok taraf yanlış ya da eksik satış tahmininden zarar görebilir. Yapay zekâ çağının getirdiği yeniliklerden olan makine öğrenmesi de birçok mühendislik uygulamasının getirdiği sorunlara olduğu gibi satış tahmini problemlerine de hızlı şekilde cevap verebilecek bir alandır. Bu çalışmada uçtan uca bir makine öğrenmesi proje süreci ele alınmıştır. Herhangi bir makine öğrenmesi projesinin adımları ve veriye yaklaşım boyutu tanıtılmıştır. Uygulama bölümünde makine öğrenmesi algoritmalarından doğrusal regresyon, Ridge, Lasso, Elastic Net, K-en yakın komşu ve Rastgele Orman algoritmaları kullanılarak gerçek veri seti için bir satış tahmin modeli geliştirilmiştir. Geliştirilen modelde en düşük hatayı veren algoritma Rastgele Orman algoritması olmuştur.

2.6.10. Makine Öğrenmesi İle Ürün Sınıflandırma İncelemesi

Günümüzde internet kullanımı arttıkça, internetten satış yapan web sitelerinin sayısı da artıyor. Bununla birlikte, internet üzerinden satın alınabilecek ürünlerin çeşitliliği de artıyor. Bu ürünlerin sınıflandırılmasına da zaman alacaktır. Bu çalışmada, TeksoSA şirketinin web sayfasında bulunan akıllı telefon, cep telefonu ve tablet bilgisayarların özellikleri kullanılarak yapılan bir sınıflandırmanın sonuçları sunulmuştur.

Bu çalışmanın amacı, elektronik ticaret firmalarına ürünü hızlı ve etkili bir şekilde sınıflandırmanın yollarını öğretmektir.

3. UYGULAMALAR

3.1. Classifcation Modellerini Kullanarak E – Posta Filtrelemek

Bu uygulamanın temel amacı gelen e-postaların içeriklerinde neler yazdığı maillerin spam olup olmadığı tarzda bir araya toplanıp oluşturulmuş bir veri seti üzerine sınıflandırma

algoritmaları üreterek gelen elektronik postaların spam bir mail olup olmadıklarını tahmin etmektir.

3.1.1. Veri Seti Hikayesi

CSV dosyası 5172 satır içeriyor, her satır bir e-posta için. Toplamda 3002 sütun bulunuyor. İlk sütun, E-posta adını gösteriyor. Adlar, gizliliği korumak için numaralarla belirlenmiş ve alıcıların gerçek adlarını içermiyor. Son sütun, tahmin için etiketleri içeriyor: 1 spam için, 0 spam değil için. Geriye kalan 3000 sütun, e-postalardaki en yaygın 3000 kelimeyi içeriyor. Bu kelimelerdeki alfabetik olmayan karakterler/kelimeler hariç tutulmuştur. Her satır için, ilgili hücrelerde o e-postadaki her kelimenin(sütunun) sayısı saklanmıştır. Bu şekilde, tüm 5172 e-posta hakkındaki bilgiler ayrı metin dosyaları yerine sıkıştırılmış bir veri çerçevesinde depolanmıştır.

3.1.2. Verilerin Yüklenmesi ve Ön İşlemeler

Gerekli kütüphaneler (numpy, pandas, matplotlib, seaborn) ve sınıflandırma algoritmaları için gerekli modüller (GaussianNB, MultinomialNB, XGBClassifier, DecisionTreeClassifier, RandomForestClassifier, LogisticRegression, SVC) import edilir.

"emails.csv" dosyası pd.read_csv fonksiyonu ile bir DataFrame olarak okunur ve "df" değişkenine atanır.

```
1. import numpy as np
2. import pandas as pd
3. import matplotlib.pyplot as plt
4. import seaborn as sns
5. from sklearn.model_selection import train_test_split
6. from sklearn.naive_bayes import GaussianNB, MultinomialNB
7. from xgboost import XGBClassifier
8. from sklearn.tree import DecisionTreeClassifier
9. from sklearn.ensemble import RandomForestClassifier
10.      from sklearn.linear_model import LogisticRegression
11.      from sklearn.svm import SVC
12.      from    sklearn.metrics    import    precision_score,    recall_score,
   accuracy_score,    f1_score,    confusion_matrix,    ConfusionMatrixDisplay,
   PrecisionRecallDisplay, RocCurveDisplay
13.
14.
15.      import os
16.      for dirname, _, filenames in os.walk('/kaggle/input'):
17.          for filename in filenames:
18.              print(os.path.join(dirname, filename))
19.
20.      import warnings
21.      warnings.filterwarnings("ignore")
```

```
22.      df = pd.read_csv("emails.csv")
```

In [3]:
```
1 df
```

Out[3]:

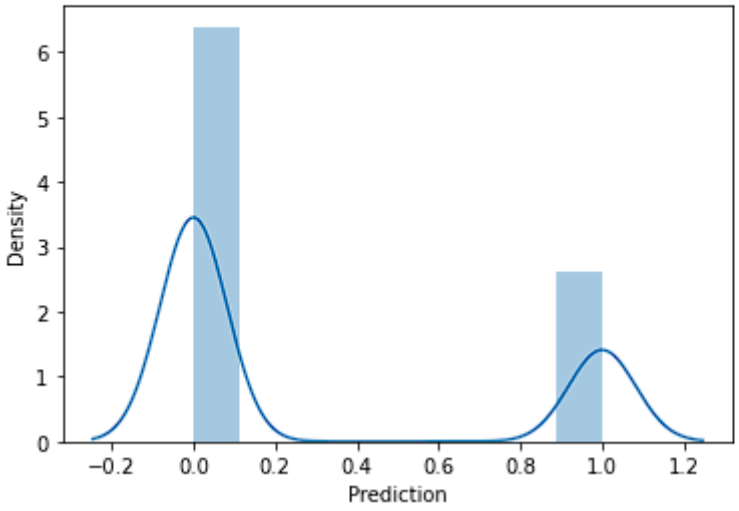
	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Prediction
0	Email 1	0	0	1	0	0	0	2	0	0	...	0	0	0	0	0	0	0	0	0	0
1	Email 2	8	13	24	6	6	2	102	1	27	...	0	0	0	0	0	0	0	1	0	0
2	Email 3	0	0	1	0	0	0	8	0	0	...	0	0	0	0	0	0	0	0	0	0
3	Email 4	0	5	22	0	5	1	51	2	10	...	0	0	0	0	0	0	0	0	0	0
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5167	Email 5168	2	2	2	3	0	0	32	0	0	...	0	0	0	0	0	0	0	0	0	0
5168	Email 5169	35	27	11	2	6	5	151	4	3	...	0	0	0	0	0	0	0	1	0	0
5169	Email 5170	0	0	1	1	0	0	11	0	0	...	0	0	0	0	0	0	0	0	0	1
5170	Email 5171	2	7	1	0	2	1	28	2	0	...	0	0	0	0	0	0	0	1	0	1
5171	Email 5172	22	24	5	1	6	5	148	8	2	...	0	0	0	0	0	0	0	0	0	0

5172 rows × 3002 columns

Spam emaillerin yoğunluğunu grafik olarak görüyoruz.

```
1 sns.distplot(df.Prediction)
```

```
<AxesSubplot:xlabel='Prediction', ylabel='Density'>
```



Sonuçları diğer verilerden ayırıp iki grup oluşturuyoruz.

```
1. y = df.iloc[:, -1].values
2. x = df.iloc[:, :3000].values
```

Train ve test ayırımını yapıyoruz.

```
1. x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,
   random_state = 1)
```

Bu veri seti üzerinde birçok teoremi deneyeceğimiz için aynı kodları tekrarlamamak amacıyla bir tahmin fonksiyonu oluşturuyoruz.

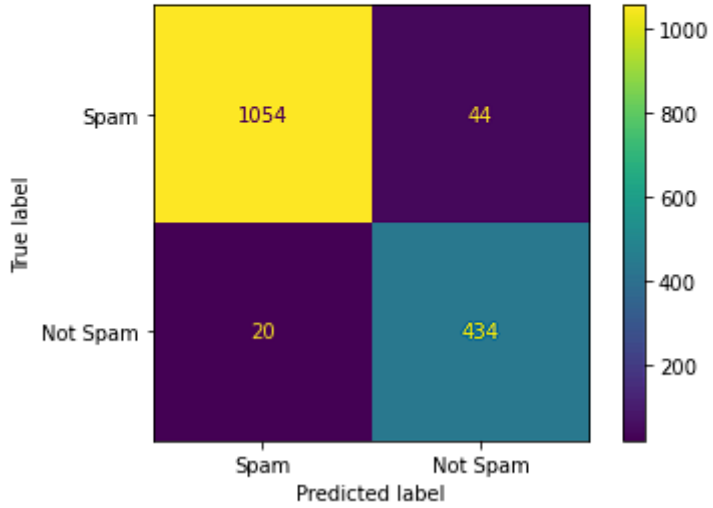
```
1.  def perform(y_pred):
2.      print("Precision : ", precision_score(y_test, y_pred))
3.      print("Recall : ", recall_score(y_test, y_pred))
4.      print("Accuracy Score : ", accuracy_score(y_test, y_pred))
5.      print("F1 Score : ", f1_score(y_test, y_pred))
6.      print("\n", confusion_matrix(y_test, y_pred))
7.      print("")
8.
9.      cm_display = ConfusionMatrixDisplay(confusion_matrix = confusion_matrix(y_test,
    y_pred), display_labels=['Spam', 'Not Spam'] )
10.     cm_display.plot()
11.     plt.show()
```

3.1.3. Gaussian Naive Bayes Teoremi

Gaussian Naive Bayes teoremini uyguluyoruz.

```
1.  model_nb = GaussianNB()
2.  model_nb.fit(x_train, y_train)
3.  y_pred_nb = model_nb.predict(x_test)
4.  perform(y_pred_nb)
```

```
Precision :  0.9079497907949791
Recall :  0.9559471365638766
Accuracy Score :  0.9587628865979382
F1 Score :  0.9313304721030043

 [[1054   44]
 [  20  434]]
```



Hassasiyet (Precision)

Duyarlılık (Recall)

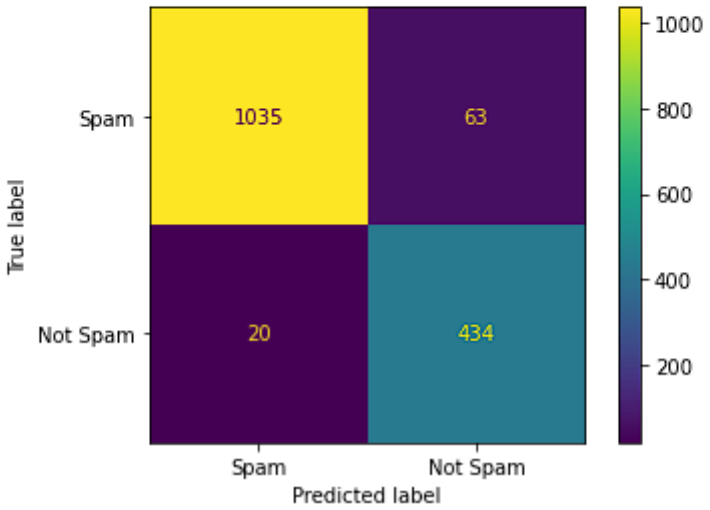
Doğruluk (Accuracy)

F1 skoru, hassasiyet ve duyarlılık arasında denge

3.1.4. Multinomial Naive Bayes Teoremi

```
1.  model_mnb = MultinomialNB()
2.  model_mnb.fit(x_train, y_train)
3.  y_pred_mnb = model_mnb.predict(x_test)
4.  perform(y_pred_mnb)
```

```
In [22]:  1  perform(y_pred_mnb)

Precision :  0.8732394366197183
Recall :  0.9559471365638766
Accuracy Score :  0.946520618556701
F1 Score :  0.9127234490010515

 [[1035   63]
 [  20  434]]
```

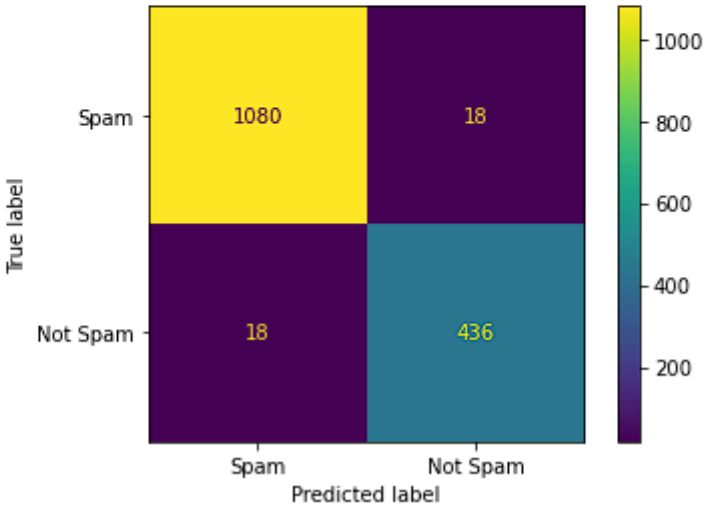


3.1.5. XGBoost Classifier

```
1.  xgb = XGBClassifier().fit(x_train, y_train)
2.  y_pred_xgb = xgb.predict(x_test)
3.  perform(y_pred_xgb)
```

```
In [25]:  1  perform(y_pred_xgb)
```

```
Precision :  0.960352422907489
Recall :  0.960352422907489
Accuracy Score :  0.9768041237113402
F1 Score :  0.960352422907489

 [[1080   18]
 [  18  436]]
```

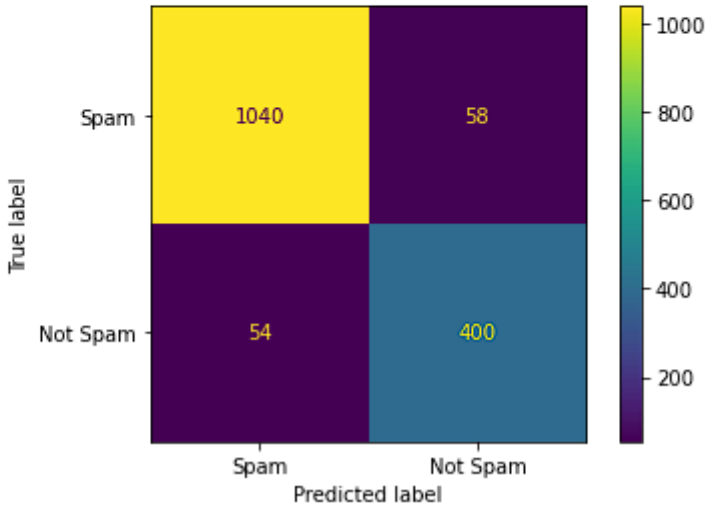


3.1.6. Decision Tree

```
1.  model_dt = DecisionTreeClassifier()
2.  model_dt.fit(x_train, y_train)
3.  y_pred_dt = model_dt.predict(x_test)
4.  perform(y_pred_dt)
```

```
In [28]:  1  perform(y_pred_dt)
```

```
Precision :  0.8733624454148472
Recall :  0.8810572687224669
Accuracy Score :  0.9278350515463918
F1 Score :  0.8771929824561404

 [[1040   58]
 [  54  400]]
```

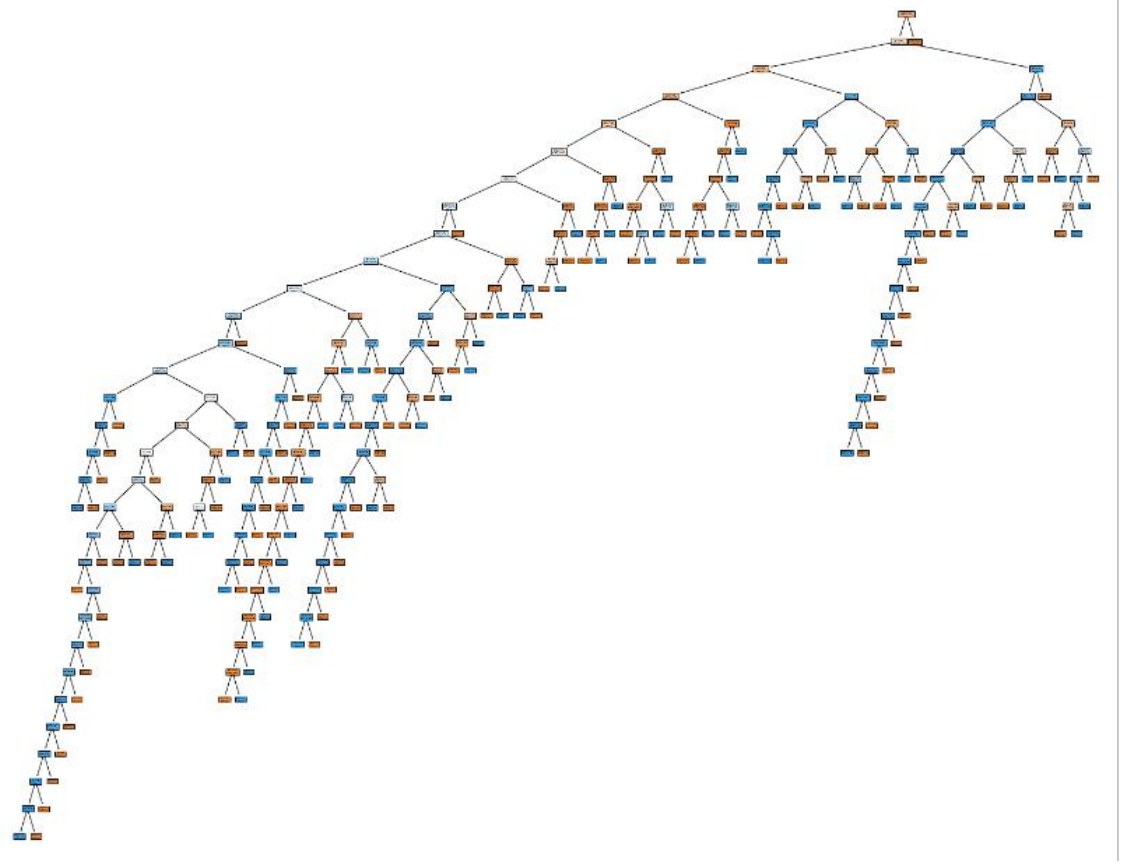


3.1.6.1. Decision Tree Görselleştirme

```
1.  feature_names = df.columns[0:3000]
2.  viz = df.copy()
3.  viz["Prediction"]=viz["Prediction"].values.astype(str)
4.  print(viz.dtypes)
5.  target_names = viz['Prediction'].unique().tolist()
6.  from sklearn.tree import plot_tree # tree diagram
7.
8.  plt.figure(figsize=(25, 20))
9.  plot_tree(model_dt, feature_names = feature_names, class_names = target_names,
    filled = True, rounded = False)
10.
11. plt.savefig('tree_visualization.png')
```

In [30]:

```
from sklearn.tree import plot_tree # tree diagram

plt.figure(figsize=(25, 20))
plot_tree(model_dt, feature_names = feature_names, class_names = target_names, filled = True, rounded = False)

plt.savefig('tree_visualization.png')
```

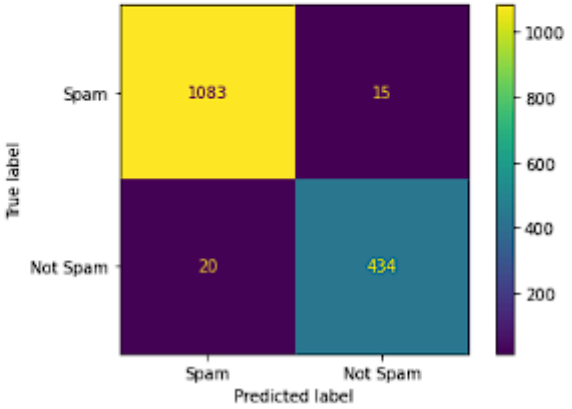


3.1.7. Random Forest

```
model_rf = RandomForestClassifier()
model_rf.fit(x_train, y_train)
y_pred_rf = model_rf.predict(x_test)
perform(y_pred_rf)
```

```
In [33]:  1  perform(y_pred_rf)
```

```
Precision :  0.9665924276169265
Recall :  0.9559471365638766
Accuracy Score :  0.9774484536082474
F1 Score :  0.9612403100775194

 [[1083   15]
 [  20  434]]
```

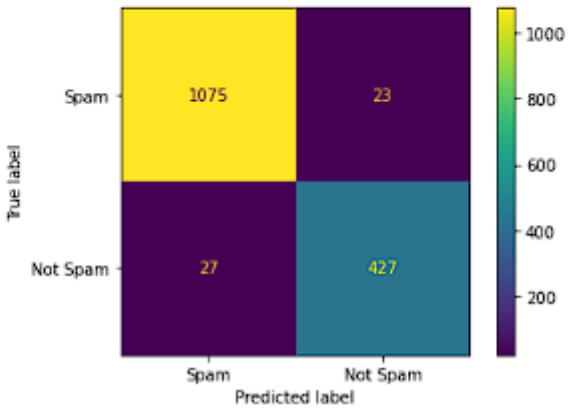


3.1.8. Logistic Regresyon

```
1.  model_lr = LogisticRegression(max_iter = 700)
2.  model_lr.fit(x_train, y_train)
3.  y_pred_lr = model_lr.predict(x_test)
4.  perform(y_pred_lr)
```

```
In [36]:  1  perform(y_pred_lr)
```

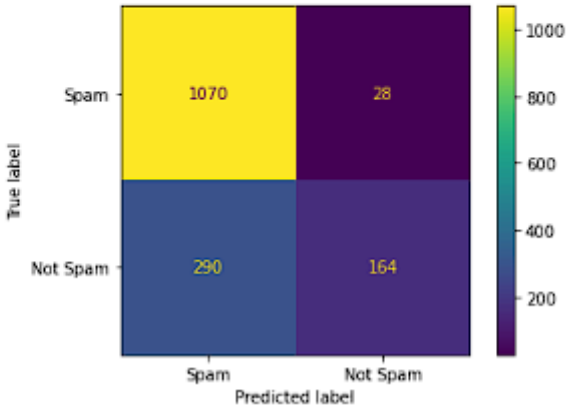
```
Precision :  0.9488888888888889
Recall :  0.9405286343612335
Accuracy Score :  0.9677835051546392
F1 Score :  0.9446902654867257

 [[1075   23]
 [  27  427]]
```



3.1.9. Support Vector Clasifier

```
model_svc = SVC()
model_svc.fit(x_train, y_train)
y_pred_svc = model_svc.predict(x_test)
perform(y_pred_svc)
```

```
In [39]: perform(y_pred_svc)

Precision :  0.8541666666666666
Recall :  0.36123348017621143
Accuracy Score :  0.7951030927835051
F1 Score :  0.5077399380804953

 [[1070   28]
 [ 290  164]]
```

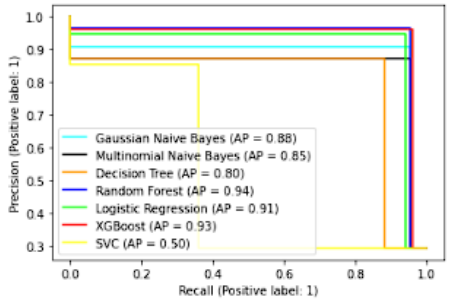


3.1.10. Karşılaştırma

```
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_curve
fig, ax = plt.subplots()
PrecisionRecallDisplay.from_predictions(y_test, y_pred_nb, name=f"Gaussian Naive Bayes", color="cyan", ax=ax)
PrecisionRecallDisplay.from_predictions(y_test, y_pred_mnb, name=f"Multinomial Naive Bayes", color="black", ax=ax)
PrecisionRecallDisplay.from_predictions(y_test, y_pred_dt, name=f"Decision Tree", color="darkorange", ax=ax)
PrecisionRecallDisplay.from_predictions(y_test, y_pred_rf, name=f"Random Forest", color="blue", ax=ax)
PrecisionRecallDisplay.from_predictions(y_test, y_pred_lr, name=f"Logistic Regression", color="lime", ax=ax)
PrecisionRecallDisplay.from_predictions(y_test, y_pred_xgb, name=f"XGBoost", color="red", ax=ax)
PrecisionRecallDisplay.from_predictions(y_test, y_pred_svc, name=f"SVC", color="yellow", ax=ax)
```

```
In [40]:
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_curve
fig, ax = plt.subplots()
PrecisionRecallDisplay.from_predictions(y_test, y_pred_nb, name=f"Gaussian Naive Bayes", color="cyan", ax=ax)
PrecisionRecallDisplay.from_predictions(y_test, y_pred_mnb, name=f"Multinomial Naive Bayes", color="black", ax=ax)
PrecisionRecallDisplay.from_predictions(y_test, y_pred_dt, name=f"Decision Tree", color="darkorange", ax=ax)
PrecisionRecallDisplay.from_predictions(y_test, y_pred_rf, name=f"Random Forest", color="blue", ax=ax)
PrecisionRecallDisplay.from_predictions(y_test, y_pred_lr, name=f"Logistic Regression", color="lime", ax=ax)
PrecisionRecallDisplay.from_predictions(y_test, y_pred_xgb, name=f"XGBoost", color="red", ax=ax)
PrecisionRecallDisplay.from_predictions(y_test, y_pred_svc, name=f"SVC", color="yellow", ax=ax)
```

```
Out[40]: <sklearn.metrics._plot.precision_recall_curve.PrecisionRecallDisplay at 0x1872d244100>
```



Bu kullanılan sınıflandırma algoritmaları arasında bu proje içerisinde doğruluk oranları büyükten küçüğe

1. Gaussian Naive Bayes
2. Multinomial Naive Bayes
3. Decision Tree
4. Random Forest
5. Logistic Regression
6. XGBoost
7. SVC

olarak görülmüştür.

3.2. Clustering Algoritmalarını Kullanarak Müşteri Segmantasyonu

Bir süpermarketin üyelik kartı dönüşüm oranını artırmak için yardımcı olmak için, farklı kümeleme tekniklerini keşfedecek ve müşteri segmentasyon analizi yapılacaktır. Müşteri segmentasyonu, alışveriş tercihleri ve satın alma geçmiği gibi ortak noktalara dayanarak benzer grupları belirleme sürecini tanımlar ve şirketin her gruba daha etkili pazarlama yapmasını sağlar.

3.2.1. Veri Seti Hikayesi

Veri seti, 200 müşteriden oluşmaktadır ve müşterilerin yaş, cinsiyet, yıllık gelir ve harcama puanıyla ilgili bilgilerini içermektedir. Harcama puanı, davranış parametreleri ve satın alma verilerine dayanarak müşterilere atanan 1 ila 100 arasında değişen sayısal bir değişkendir.

Veri seti ayrıca müşterinin kimlik numarasını da içermektedir, ancak analize başlamadan önce bu kimlik numarası çıkarılacaktır.

3.2.2. Verilerin Yüklenmesi ve Ön İşlemeler

```
1.  import os, warnings
2.  warnings.filterwarnings("ignore")
3.  import pandas as pd
4.  import numpy as np
5.  import seaborn as sns
6.  import matplotlib.pyplot as plt
7.  import plotly.express as px
8.  import plotly.graph_objects as go
9.  import plotly.figure_factory as ff
10. from plotly.subplots import make_subplots
11. from plotly.offline import plot, iplot, init_notebook_mode
12. from sklearn.preprocessing import StandardScaler, MinMaxScaler
13. from scipy.cluster import hierarchy
14. from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
15.
16.
17. cust = pd.read_csv('Mall_Customers.csv', sep=',')
18. cust.rename(columns={"Annual Income (k$)": "Annual Income", "Spending Score (1-
    100)": "Spending Score"}, inplace=True)
19. print("Veri setinde {:,}  gözlem ve {} kolon bulunmaktadır.".format(cust.shape[0],
    cust.shape[1]))
20. print("Verilerde {} eksik değer var.".format(cust.isna().sum().sum()))
21. cust.head()
```

```
Veri setinde 200  gözlem ve 5 kolon bulunmaktadır.
Verilerde 0 eksik değer var.
```

Out[8]:

	CustomerID	Gender	Age	Annual Income	Spending Score
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

Özet istatistiği görüntülüyoruz.

```
1.  cust.drop('CustomerID', axis=1, inplace=True)
2.  pd.DataFrame(cust.describe()).style.set_caption("Sayısal      Değişkenlerin      Özet
    İstatistikleri")
```

In [9]:
```
cust.drop('CustomerID', axis=1, inplace=True)
pd.DataFrame(cust.describe()).style.set_caption("Sayısal Değişkenlerin Özet İstatistikleri")
```

Out[9]:

Sayısal Değişkenlerin Özet İstatistikleri

	Age	Annual Income	Spending Score
count	200.000000	200.000000	200.000000
mean	38.850000	60.560000	50.200000
std	13.969007	26.264721	25.823522
min	18.000000	15.000000	1.000000
25%	28.750000	41.500000	34.750000
50%	36.000000	61.500000	50.000000
75%	49.000000	78.000000	73.000000
max	70.000000	137.000000	99.000000

Kategorik Değişkenlerin Özet İstatistiklerini görüntülüyoruz.

```
cust['Gender'] = ['Women' if i == 'Female' else 'Men' for i in cust.Gender]
pd.DataFrame(cust.select_dtypes('object').describe().T).style.set_caption("Summary Statistics of Categorical Variables")
```

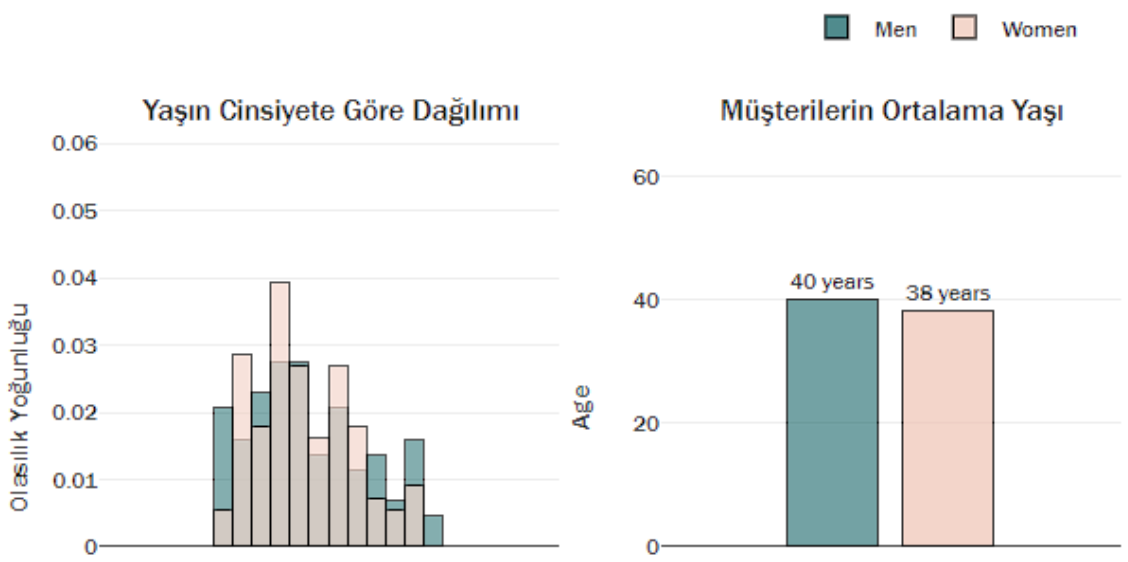
	count	unique	top	freq
Gender	200	2	Women	112

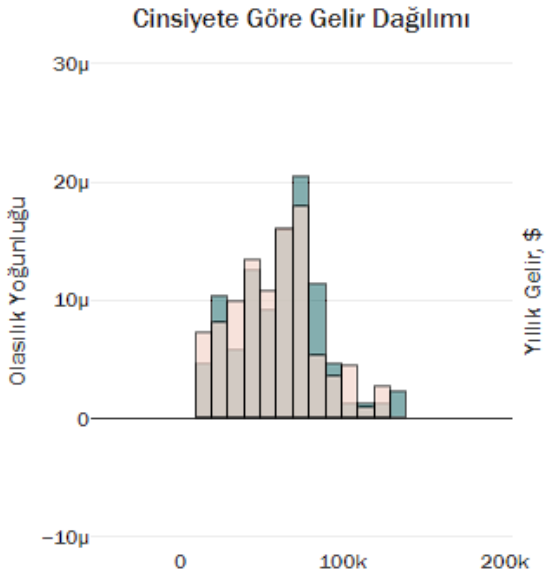
Keşifsel veri analizi yapıp bunu görselleştiriyoruz.

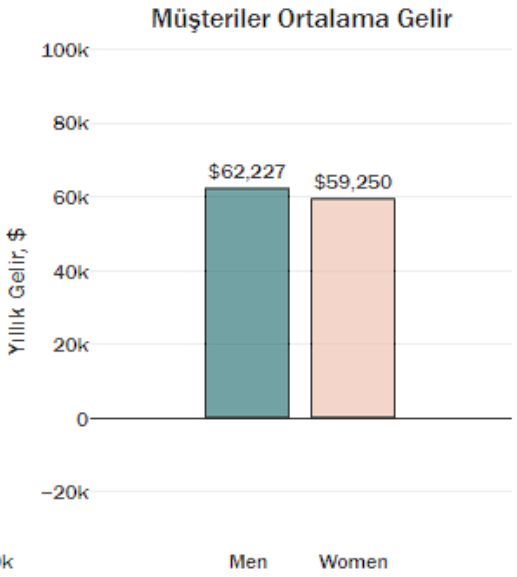
```
init_notebook_mode(connected=True)
plot_df=cust.copy()
plot_df['Annual Income']=plot_df['Annual Income'].mul(1000)
p1=plot_df.groupby('Gender')['Age'].mean().round(0).astype(int).reset_index()
p2=plot_df.groupby('Gender')['Annual Income'].mean().reset_index()
p3=plot_df.groupby('Gender')['Spending Score'].mean().round(0).astype(int).reset_index()

temp = dict(layout=go.Layout(font=dict(family="Franklin Gothic", size=12)))
fig = make_subplots(rows=3, cols=2,
                    subplot_titles=("Yaşın Cinsiyete Göre Dağılımı",
                                    "Müşterilerin Ortalama Yaşı",
                                    "Cinsiyete Göre Gelir Dağılımı",
                                    "Müşteriler Ortalama Gelir",
                                    "Harcamaların Cinsiyete Göre Dağılımı",
                                    "Müşterilerin Ortalama Harcaması")
                   )

fig.add_trace(go.Histogram(x=plot_df[plot_df.Gender=='Men']['Age'], histnorm='probability density',
                           marker=dict(color='#508B8D',opacity=0.7, line=dict(width=1, color='#000000')),
                           nbinsx=20, name="Men"),
              row=1, col=1)
fig.add_trace(go.Histogram(x=plot_df[plot_df.Gender=='Women']['Age'], histnorm='probability density',
                           marker=dict(color='#F3D6CB',opacity=0.7, line=dict(width=1, color='#000000')),
                           nbinsx=20, name="Women"),
              row=1, col=1)

```

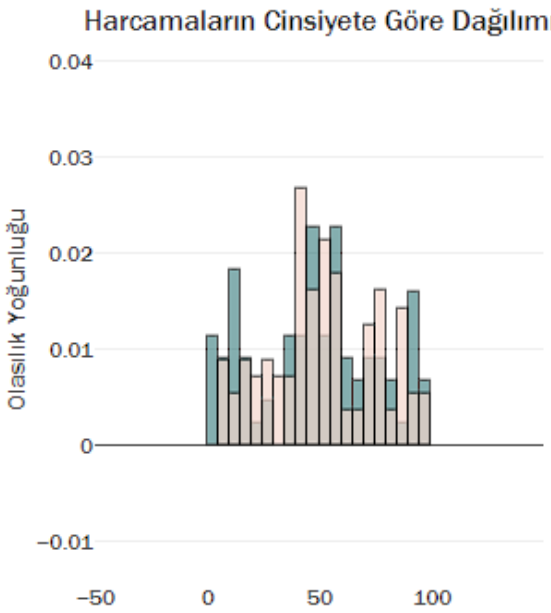
```
fig.add_trace(go.Bar(x=p1['Gender'], y=p1['Age'], text=p1['Age'],
texttemplate='%{text} years', textposition='outside',
                    marker=dict(color=['#508B8D', '#F0CABD'],
opacity=0.8),width=.8,
                    hovertemplate='Average Age Among %{x} = %{y}
years<extra></extra>', showlegend=False),
              row=1, col=2)

fig.add_trace(go.Histogram(x=plot_df[plot_df.Gender=='Men']['Annual Income'],
histnorm='probability density',
                           marker=dict(color='#508B8D', line=dict(width=1,
color='#000000')),
                           opacity=0.7, name="Men", nbinsx=20, showlegend=False),
              row=2, col=1)
fig.add_trace(go.Histogram(x=plot_df[plot_df.Gender=='Women']['Annual Income'],
histnorm='probability density',
                           marker=dict(color='#F3D6CB', line=dict(width=1,
color='#000000')),
                           opacity=0.7, name="Women", nbinsx=20, showlegend=False),
              row=2, col=1)
fig.add_trace(go.Bar(x=p2['Gender'], y=p2['Annual Income'], text=p2['Annual
Income'],
                    texttemplate='$%{text:,.0f}', textposition='outside',
                    marker=dict(color=['#508B8D', '#F0CABD'],
opacity=0.8),width=.8,
                    hovertemplate='Average Income Among %{x} =
$%{y}<extra></extra>', showlegend=False),
              row=2, col=2)
fig.add_trace(go.Histogram(x=plot_df[plot_df.Gender=='Men']['Spending Score'],
histnorm='probability density',
                           marker=dict(color='#508B8D', line=dict(width=1,
color='#000000')),
                           opacity=0.7, name="Men", nbinsx=20, showlegend=False),
              row=3, col=1)
fig.add_trace(go.Histogram(x=plot_df[plot_df.Gender=='Women']['Spending Score'],
histnorm='probability density',
                           marker=dict(color='#F3D6CB', line=dict(width=1,
color='#000000')),
                           opacity=0.7, name="Women", nbinsx=20, showlegend=False),
              row=3, col=1)
fig.add_trace(go.Bar(x=p3['Gender'], y=p3['Spending Score'], text=p3['Spending
Score'],
                    texttemplate='%{text}', textposition='outside',
                    marker=dict(color=['#508B8D', '#F0CABD'],
opacity=0.8),width=.8,
                    hovertemplate='Average Spending Score Among %{x} =
%{y}<extra></extra>', showlegend=False),
              row=3, col=2)
fig.update_traces(marker=dict(line=dict(width=1, color='#000000')))
fig.update_layout(template=temp,barmode='overlay', height=1500, width=700,
                  legend=dict(orientation="h", yanchor="bottom", xanchor="right",
y=1.03, x=.97),
                  xaxis1_title="Yaş", yaxis1_title='Olasılık Yoğunluğu',
                  xaxis2_title="Cinsiyet", yaxis2_title="Age", yaxis2_range=[0,45],
                  xaxis3_title="Yıllık Gelir, $", yaxis3_title='Olasılık Yoğunluğu',
                  xaxis4_title="Cinsiyet", yaxis4_title="Yıllık Gelir, $",
yaxis4_range=[0,69e3],
                  xaxis5_title="Harcama Puanı", yaxis5_title='Olasılık Yoğunluğu',
                  xaxis6_title="Cinsiyet", yaxis6_title="Harcama Puanı",
yaxis6_range=[0,59]
                 )
fig.show()

# Pairplots
fig = ff.create_scatterplotmatrix(cust, diag='box', index='Gender',
colormap=['#508B8D', '#F0CABD'])
```

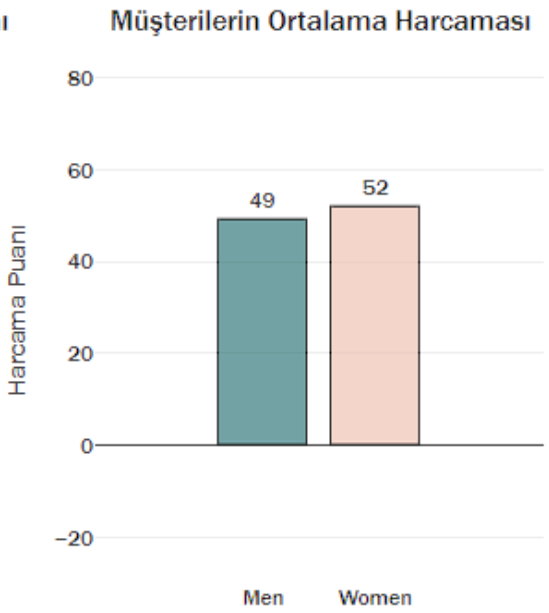
```
72. fig.update_traces(marker=dict(size=9, opacity=0.85, line=dict(width=1,
    color='#F7F7F7')))
73. fig.update_layout(title="Mall Customer Pair Plots", template=temp,
74.                   legend=dict(orientation="h", yanchor="bottom", y=1.02, x=.35),
75.                   height=900, width=700)
76. fig.show()
77.
78. # Correlations
79. corr=cust.corr()
80. x = corr.columns.tolist()
81. y = corr.index.tolist()
82. z = corr.values
83. text = corr.values.round(2)
84.
85. fig = ff.create_annotated_heatmap(z=z, x=x, y=y, annotation_text=text,
    colorscale='mint',
86.                                  reversescale=True, showscale=True,
87.                                  hovertemplate="Correlation of %{x} and %{y}=
    %{z:.3f}")
88. fig.update_layout(template=temp, title="AVM Müşteri İlişkileri", yaxis_tickangle=-
    30)
89. fig.show()
```

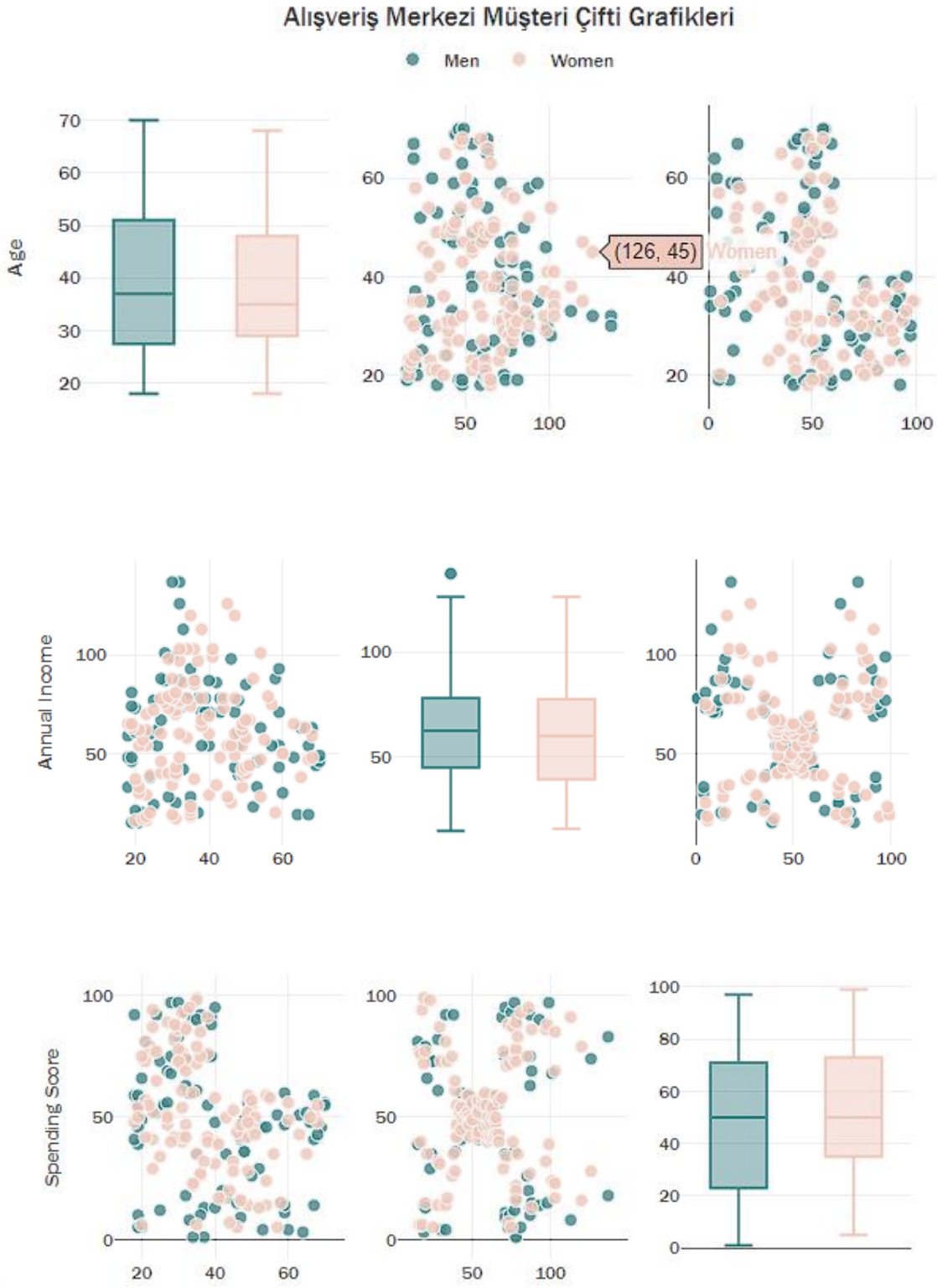


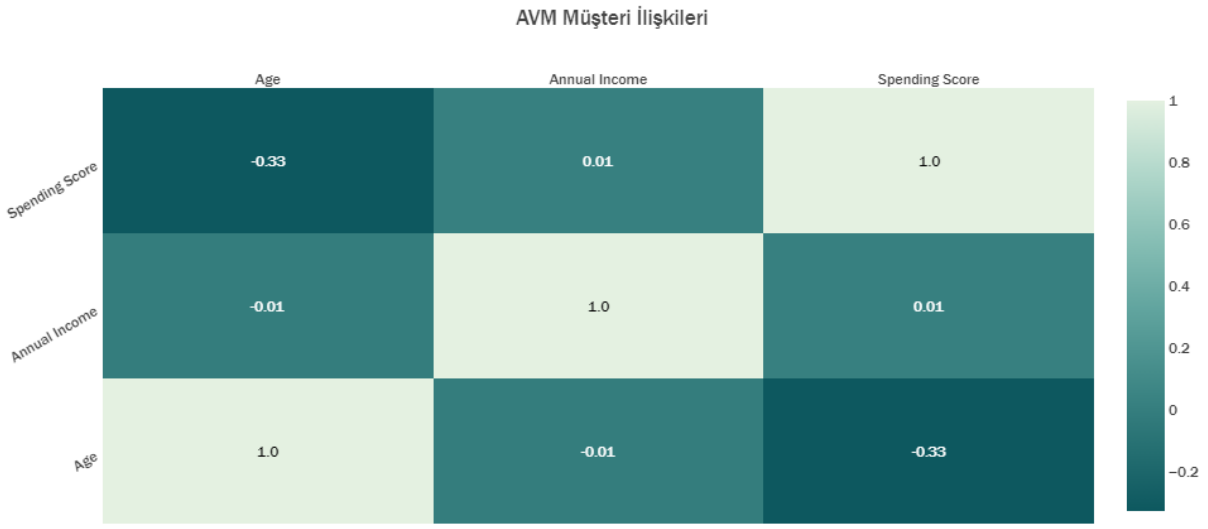

Cinsiyete Göre Gelir Dağılımı
Olasılık Yoğunluğu
30µ
20µ
10µ
0
−10µ
0
100k
200k


Müşteriler Ortalama Gelir
Yıllık Gelir, $
100k
80k
60k
40k
20k
0
−20k
$62,227
$59,250
Men
Women


Harcamaların Cinsiyete Göre Dağılımı
Olasılık Yoğunluğu
0.04
0.03
0.02
0.01
0
−0.01
−50
0
50
100


Müşterilerin Ortalama Harcaması
Harcama Puanı
80
60
40
20
0
−20
49
52
Men
Women

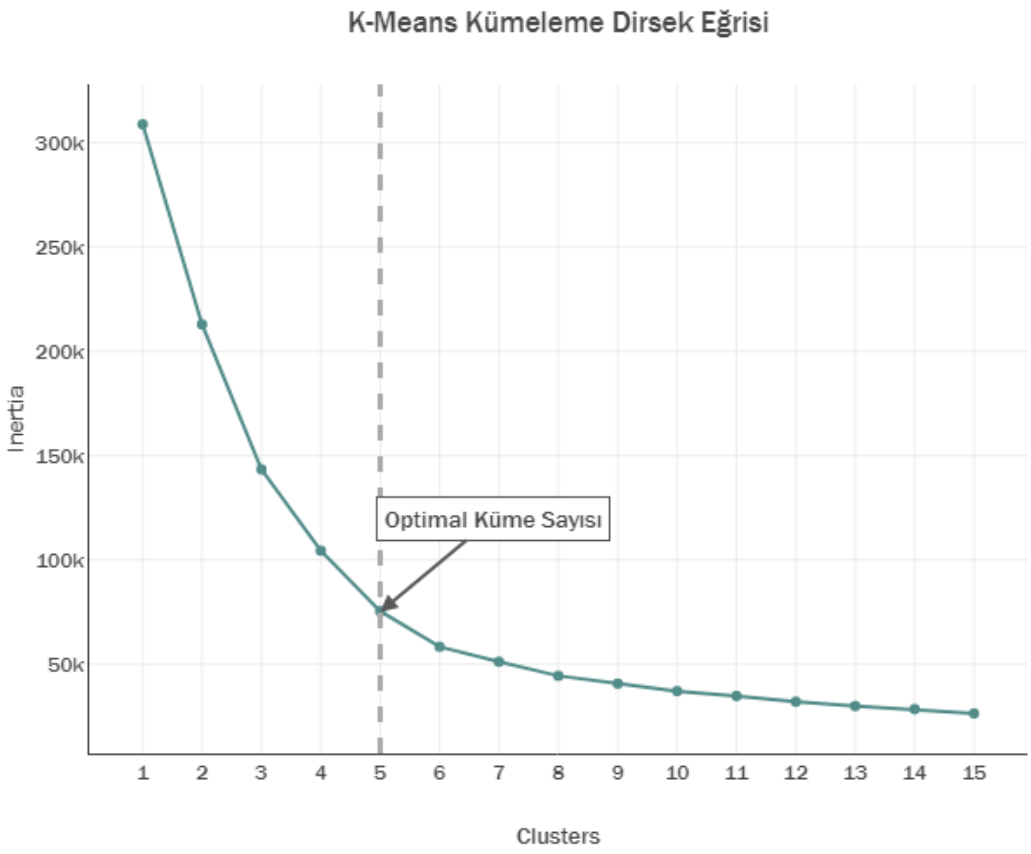

Alışveriş Merkezi Müşteri Çifti Grafikleri
Men
Women
Age
Annual Income
Spending Score
(126, 45) Women



3.2.3. K-Means Clustering

```
clust_df = cust.copy()
clust_df['Gender'] = [1 if i == "Women" else 0 for i in clust_df.Gender]

k_means = list()
for clust in range(1,16):
    km = KMeans(n_clusters=clust, init='k-means++', random_state=21).fit(clust_df)
    k_means.append(pd.Series({'Clusters': clust,
                              'Inertia': km.inertia_,
                              'model': km}))

    plot_km = (pd.concat(k_means, axis=1).T
               [['Clusters','Inertia']]
               .set_index('Clusters'))

    fig = px.line(plot_km, x=plot_km.index, y='Inertia', markers=True)
    fig.add_vline(x=5, line_width=3, line_dash="dash", line_color="darkgrey")
    fig.add_annotation(
        xref="x domain",
        yref="y",
        x=.31,
        y=75e3,
        text="Optimal Küme Sayısı",
        axref="x domain",
        ayref="y",
        ax=.43,
        ay=12e4,
        arrowhead=2,
        bordercolor="#585858",
        borderpad=4,
```

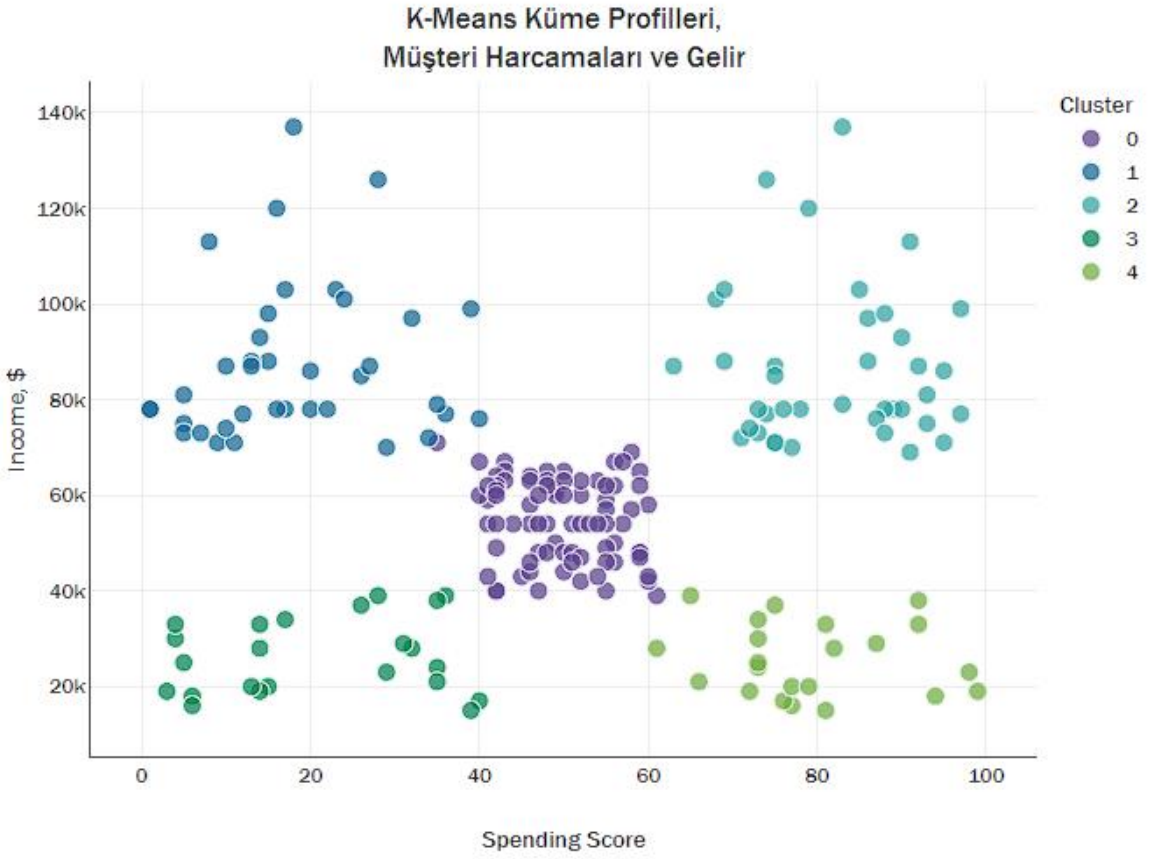
```
        bgcolor='white',
        font=dict(size=14)
    )
    fig.update_traces(line_color='#518C89')
    fig.update_layout(template=temp, title="K-Means Kümeleme Dirsek Eğrisi",
                      xaxis=dict(tickmode = 'linear', showline=True), yaxis=dict(showline=True), width=700)
    fig.show()
```



En optimize küme sayısını 5 olarak buluyoruz.

```
km = KMeans(n_clusters=5, random_state=21)
km_pred = km.fit_predict(clust_df)
plot_km=clust_df.copy()
plot_km['K-Means Cluster'] = km_pred
plot_km=plot_km.sort_values(by='K-Means Cluster')
plot_km['K-Means Cluster'] = plot_km['K-Means Cluster'].astype(str)

# Plot of clusters
fig = px.scatter(plot_km, x="Spending Score", y="Annual Income", color="K-Means Cluster",
                 color_discrete_sequence=px.colors.qualitative.Prism)
fig.update_traces(marker=dict(size=11, opacity=0.75, line=dict(width=1, color='#F7F7F7')))
fig.update_layout(template=temp, title="K-Means Küme Profilleri,<br>Müşteri Harcamaları ve Gelir",
                  width=700, legend_title='Cluster',
```

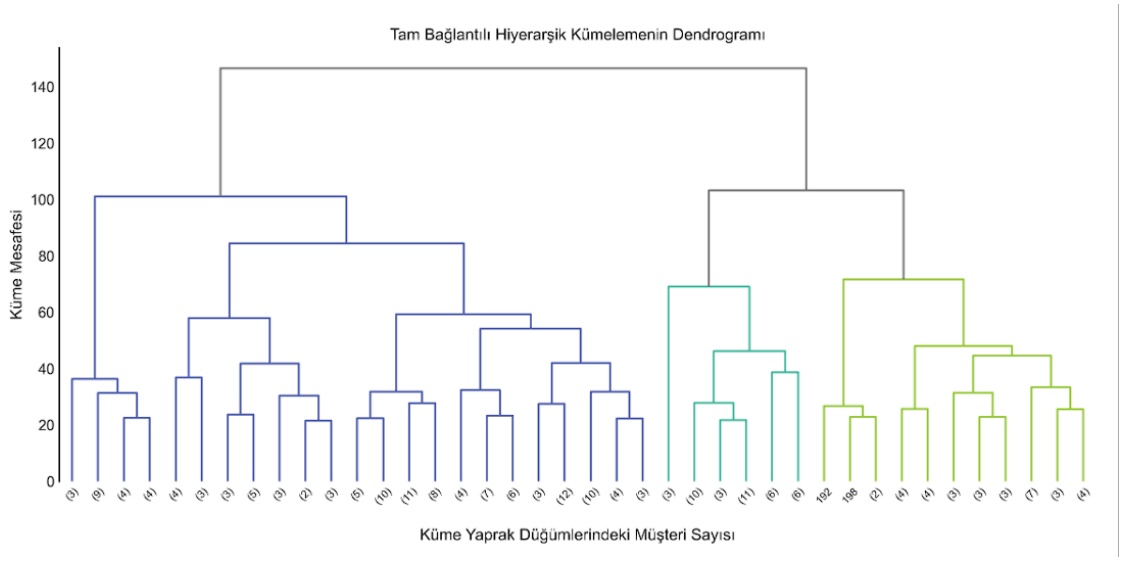
```
                  xaxis=dict(title='Spending Score', showline=True, zeroline=False),
                  yaxis=dict(title='Income, $', ticksuffix='k', showline=True))
fig.show()
```



3.2.4. Hiyerarşik Clustering

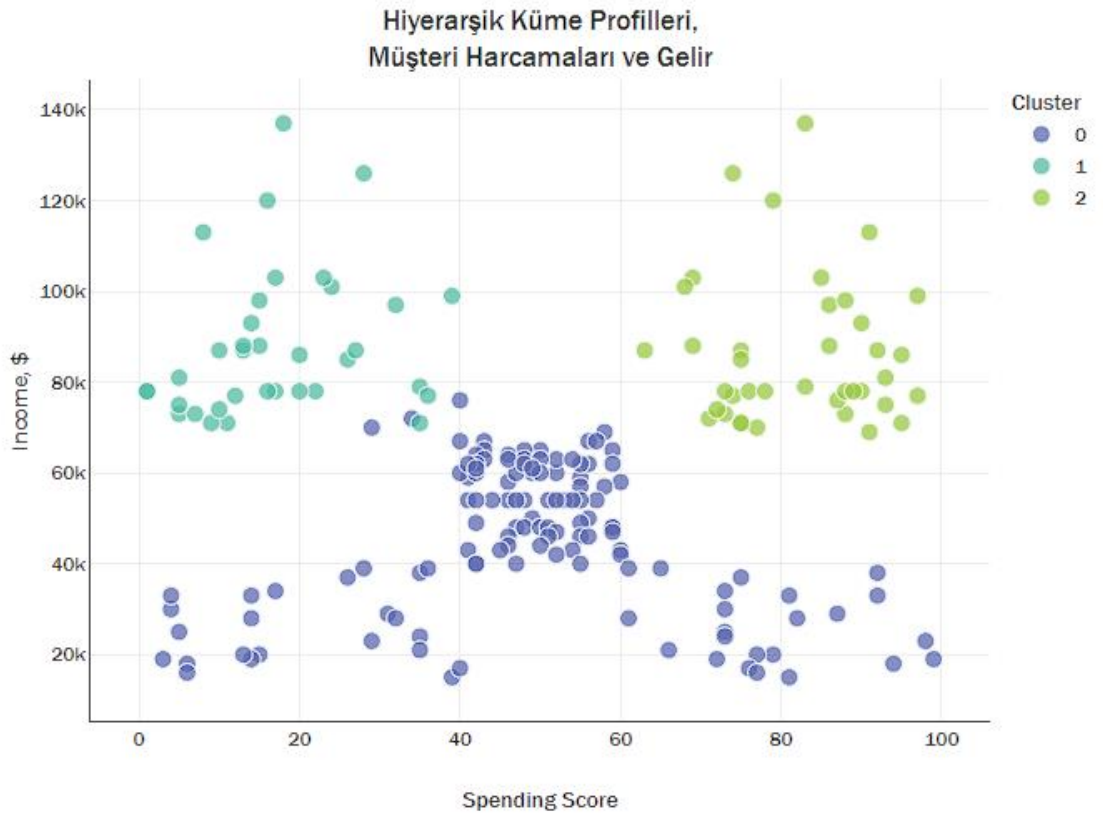
```
sns.set(rc={'figure.dpi':400})
sns.set_context('notebook')
sns.set_style('ticks')

Z = hierarchy.linkage(clust_df, method='complete', metric='euclidean')
fig, ax = plt.subplots(figsize=(14,6))
hierarchy.set_link_color_palette(['#5d69b1', '#52bca3', '#99c945'])
den = hierarchy.dendrogram(Z, orientation='top', color_threshold=102,
                           p=40, truncate_mode='lastp',
                           show_leaf_counts=True, ax=ax,
                           above_threshold_color='grey')
#ax.axhline(101, color='grey', linestyle='--')
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['bottom'].set_visible(False)
ax.tick_params(axis=u'both', which=u'both',length=0)
ax.set_xlabel('\nKüme Yaprak Düğümlerindeki Müşteri Sayısı')
ax.set_ylabel('Küme Mesafesi')
ax.set_title('\nTam Bağlantılı Hiyerarşik Kümelemenin Dendrogramı')
fig.show()
```

Hiyerarşi görünümü



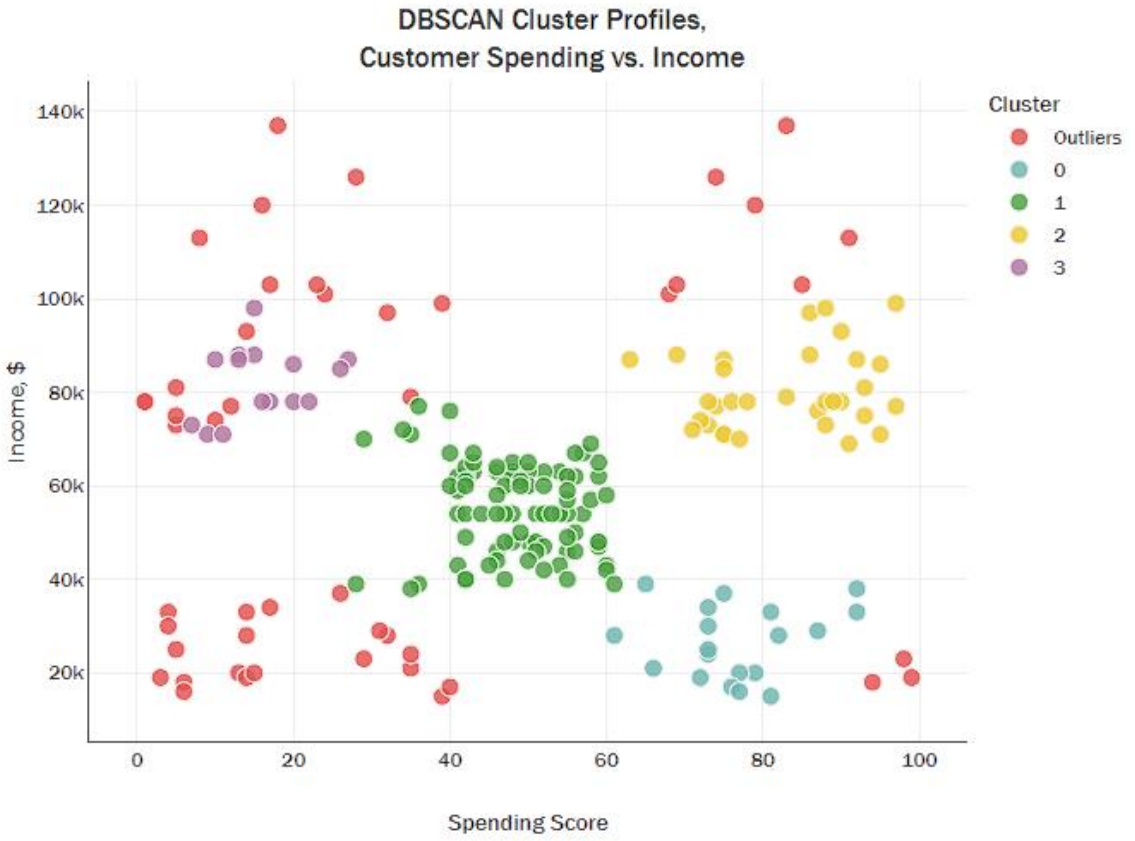
Hiyerarşik clustering grafiği

```
1. hc = AgglomerativeClustering(3, affinity='euclidean', linkage='complete',
   compute_full_tree=False)
2. hc_pred = hc.fit_predict(clust_df)
3. plot_hc=clust_df.copy()
4. plot_hc["Hierarchical Cluster"]=hc_pred
5. plot_hc=plot_hc.sort_values(by='Hierarchical Cluster')
6. plot_hc['Hierarchical Cluster'] = plot_hc['Hierarchical
   Cluster'].astype(str)
7.
8. # Plot of clusters
9. fig = px.scatter(plot_hc, x="Spending Score", y="Annual Income",
   color="Hierarchical Cluster",
10.                    color_discrete_sequence=px.colors.qualitative.Vivid[
   1:])
11.     fig.update_traces(marker=dict(size=11,                opacity=0.75,
   line=dict(width=1, color='#F7F7F7')))
12.     fig.update_layout(template=temp,       title="Hiyerarşik       Küme
   Profilleri,<br>Müşteri Harcamaları ve Gelir",
13.                       width=700, legend_title = 'Cluster',
14.                       xaxis=dict(title='Spending    Score',showline=True,
   zeroline=False),
15.                       yaxis=dict(title='Income,
   $',ticksuffix='k',showline=True))
16.     fig.show()
```



3.2.5. DBSCAN (Gürültülü Uygulamaların Yoğunluk Tabanlı Mekansal Kümelenmesi)

```
db=DBSCAN(eps=15, min_samples=11, metric='euclidean')  #17,15 14,7 12 7 115
db_preds=db.fit_predict(clust_df)
plot_db=clust_df.copy()
plot_db['DB Cluster'] = db_preds
plot_db=plot_db.sort_values(by='DB Cluster')
plot_db['DB Cluster'] = plot_db['DB Cluster'].astype(str).apply(lambda x: 'Outliers' if x == '-1' else x)


# Plot of clusters
fig = px.scatter(plot_db, x="Spending Score", y="Annual Income", color="DB Cluster",
                 color_discrete_sequence=px.colors.qualitative.T10[2:])
fig.update_traces(marker=dict(size=11,        opacity=0.85,        line=dict(width=1, color='#F7F7F7')))
fig.update_layout(template=temp,    title="DBSCAN    Cluster    Profiles,<br>Customer Spending vs. Income",
                  width=700, legend_title = 'Cluster',
                  xaxis=dict(title='Spending Score',showline=True, zeroline=False),
                  yaxis=dict(title='Income, $',ticksuffix='k',showline=True))
fig.show()
```



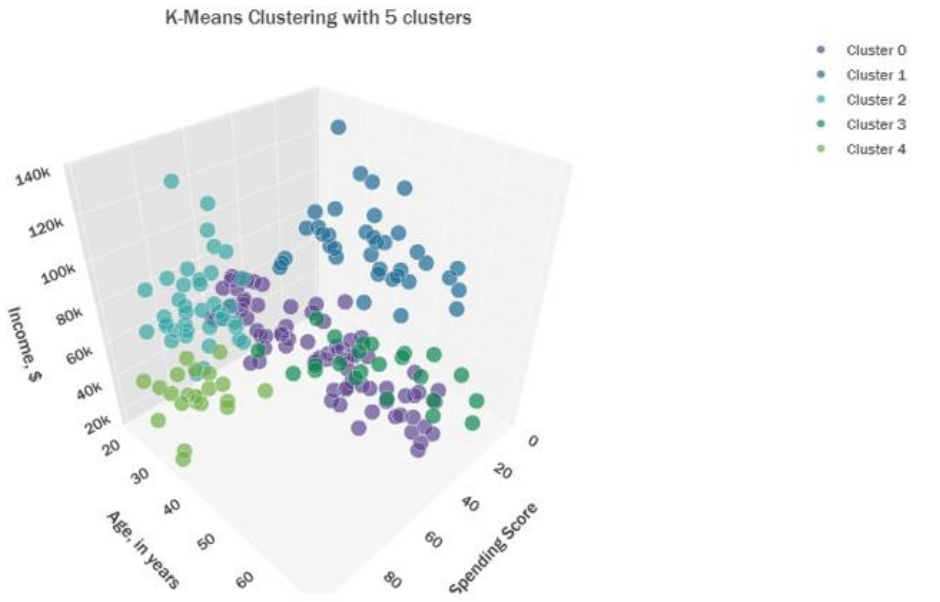
3.2.6. Karşılaştırma

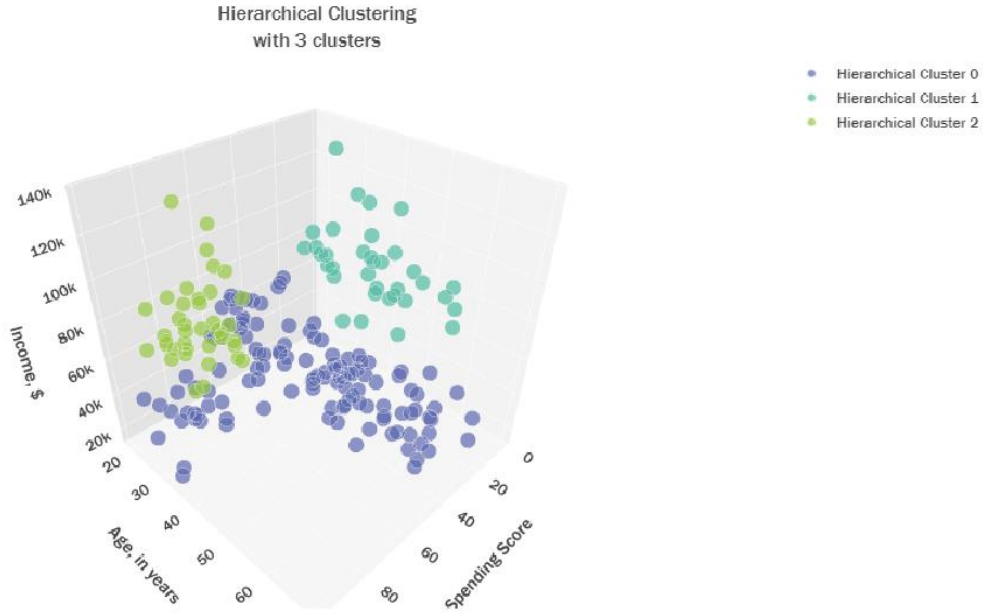
```
fig = make_subplots(rows=3, cols=1,
                    vertical_spacing=0.1,
                    specs=[[{'type': 'scatter3d'}],
                           [{'type': 'scatter3d'}],
                           [{'type': 'scatter3d'}]],
                     subplot_titles=("K-Means Clustering with 5 clusters",
                                     "Hierarchical    Clustering<br>with    3 clusters",
                                     "DBSCAN<br>with 4 clusters")
                   )

      # Adding clusters to scatterplots
      plot_km['K-Means Cluster'] = plot_km['K-Means Cluster'].astype(int)
      plot_km=plot_km.sort_values(by='K-Means Cluster')
      for i in range(0,5):
          fig.add_trace(go.Scatter3d(x = plot_km[plot_km['K-Means Cluster'] == i]['Spending Score'],
                                     y = plot_km[plot_km['K-Means Cluster'] == i]['Age'],
                                     z = plot_km[plot_km['K-Means Cluster'] == i]['Annual Income'],
                                     mode = 'markers', marker=dict(
                                         size=7,
```

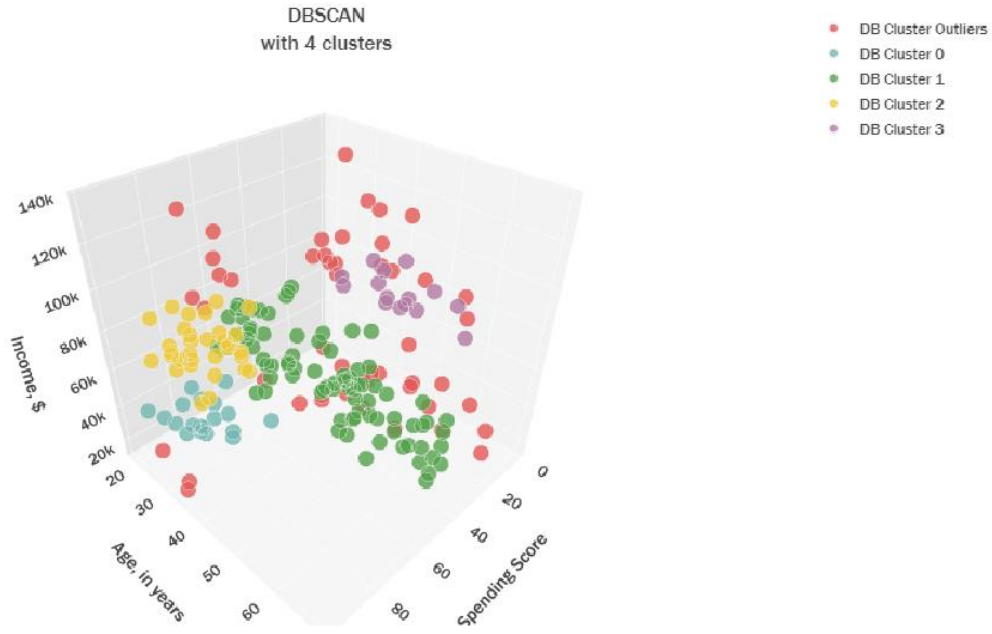
```
                                        color                                =
px.colors.qualitative.Prism[i],
                                        line_width = 1,
                                        line_color='#F7F7F7',
                                        opacity=0.7),
                                    name    =    str('Cluster    '+str(i)),
legendgroup = 1),
                          row=1, col=1)

      plot_hc['Hierarchical      Cluster']      =      plot_hc['Hierarchical
Cluster'].astype(int)
      plot_hc=plot_hc.sort_values(by='Hierarchical Cluster')
      for i in range(0,3):
          fig.add_trace(go.Scatter3d(x    =    plot_hc[plot_hc['Hierarchical
Cluster'] == i]['Spending Score'],
                                    y    =    plot_hc[plot_hc['Hierarchical
Cluster'] == i]['Age'],
                                    z    =    plot_hc[plot_hc['Hierarchical
Cluster'] == i]['Annual Income'],
                                    mode = 'markers', marker=dict(
                                        size=7,
                                        color                                =
px.colors.qualitative.Vivid[i+1],
                                        line_width = 1,
                                        line_color='#F7F7F7',
                                        opacity=0.7),
                                    name    =    str('Hierarchical    Cluster
'+str(i)), legendgroup = 2),
                           row=2, col=1)

      for i, j in enumerate(plot_db['DB Cluster'].unique()):
          fig.add_trace(go.Scatter3d(x  =  plot_db[plot_db['DB  Cluster']  ==
j]['Spending Score'],
                                    y  =  plot_db[plot_db['DB  Cluster']  ==
j]['Age'],
                                    z  =  plot_db[plot_db['DB  Cluster']  ==
j]['Annual Income'],
                                    mode = 'markers', marker=dict(
                                        size=7,
                                        color                                =
px.colors.qualitative.T10[i+2],
                                        line_width = 1,
                                        line_color='#F7F7F7',
                                        opacity=0.8),
                                    name   =   str('DB   Cluster   '+str(j)),
legendgroup = 3),
                           row=3, col=1)

      fig.update_traces(hovertemplate='Customer       Spending       Score:
%{x}<br>Income: $%{z}<br>Age: %{y}')
      fig.update_layout(title="Customer Segments based on Income, Spending,
and Age",
```

```
                    template=temp, height=1800, legend_tracegroupgap = 500,
                    scene=dict(aspectmode='cube',
                               xaxis = dict(title='Spending Score',
                                            backgroundcolor="#F3F3F3",
                                            gridcolor="white",
                                            showbackground=True,
                                            zerolinecolor="white",),
                               yaxis = dict(title='Age, in years',
                                            backgroundcolor="#E4E4E4",
                                            gridcolor="white",
                                            showbackground=True,
                                            zerolinecolor="white"),
                               zaxis = dict(title='Income, $',
                                            ticksuffix='k',
                                            backgroundcolor="#F6F6F6",
                                            gridcolor="white",
                                            showbackground=True,
                                            zerolinecolor="white")),
                    scene2=dict(aspectmode='cube',
                               xaxis = dict(title='Spending Score',
                                            backgroundcolor="#F3F3F3",
                                            gridcolor="white",
                                            showbackground=True,
                                            zerolinecolor="white",),
                               yaxis = dict(title='Age, in years',
                                            backgroundcolor="#E4E4E4",
                                            gridcolor="white",
                                            showbackground=True,
                                            zerolinecolor="white"),
                               zaxis = dict(title='Income, $',
                                            ticksuffix='k',
                                            backgroundcolor="#F6F6F6",
                                            gridcolor="white",
                                            showbackground=True,
                                            zerolinecolor="white")),
                    scene3=dict(aspectmode='cube',
                               xaxis = dict(title='Spending Score',
                                            backgroundcolor="#F3F3F3",
                                            gridcolor="white",
                                            showbackground=True,
                                            zerolinecolor="white",),
                               yaxis = dict(title='Age, in years',
                                            backgroundcolor="#E4E4E4",
                                            gridcolor="white",
                                            showbackground=True,
                                            zerolinecolor="white"),
                               zaxis = dict(title='Income, $',
                                            ticksuffix='k',
                                            backgroundcolor="#F6F6F6",
                                            gridcolor="white",
```

```
                                                    showbackground=True,
                                                    zerolinecolor="white"))
                        )
    fig.show()
```






DBSCAN
with 4 clusters
DB Cluster Outliers
DB Cluster 0
DB Cluster 1
DB Cluster 2
DB Cluster 3
Income, $
140k
120k
100k
80k
60k
40k
20k
Age, in years
20
30
40
50
60
Spending Score
0
20
40
60
80

4.SONUÇ

Bu rapor, veri bilimi, makine öğrenmesi, clustering ve classification konularına odaklanarak temel bilgileri sunmayı amaçlamaktadır. Veri bilimi, günümüzde büyük bir öneme sahip olan bir disiplindir ve verilerin analiz edilerek bilgiye dönüştürülmesi sürecini kapsar. Makine öğrenmesi ise veri biliminin önemli bir alanıdır ve algoritmaların veri üzerinde öğrenme yeteneği geliştirilerek tahminler ve kararlar oluşturulması sağlanır.

Bu raporda, gözetimsiz öğrenme ve gözetimli öğrenme kavramları detaylı bir şekilde incelenmiştir. Gözetimsiz öğrenme yöntemlerinden biri olan clustering, veri kümesinde benzer özelliklere sahip verileri gruplandırmak için kullanılırken, gözetimli öğrenme yöntemlerinden biri olan classification, verilerin belirli sınıflara ayrılması için kullanılır. Her iki yöntemin algoritmaları, avantajları ve dezavantajları üzerinde durulmuştur.

Ayrıca, clustering ve classification arasındaki temel farklar ele alınmıştır. Clustering ve classification'ın problem tanımı, hedefleri, veri özellikleri, hazırlık süreci, algoritma seçimi ve sonuç değerlendirme süreçleri karşılaştırmalı olarak incelenmiştir. Her iki yöntemin performans ölçütleri de açıklanmış ve karşılaştırılmıştır.

Raporun son bölümünde, uygulama alanlarına odaklanılmıştır. Classification modelleri kullanarak e-posta filtreleme ve clustering algoritmaları kullanarak müşteri segmentasyonu gibi gerçek hayat uygulamaları ele alınmıştır. Bu uygulamalar üzerinde farklı algoritma yöntemleri kullanılarak elde edilen sonuçlar karşılaştırılmış ve değerlendirilmiştir.

Sonuç olarak, veri bilimi, makine öğrenmesi, clustering ve classification gibi alanlar, günümüzde iş dünyasında ve birçok sektörde büyük öneme sahiptir. Bu rapor, temel bilgileri sunmanın yanı sıra, ilgili uygulamalarla birlikte bu alanlara dair daha derin bir anlayış kazanmanızı sağlamayı hedeflemektedir. Veri bilimi ve makine öğrenmesinin gelecekteki gelişimleri, daha da ileri analiz ve tahmin yetenekleri sunarak işletmelere rekabet avantajı sağlayacaktır.

5.KAYNAKÇA

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

Mitchell, T. M. (1997). Machine Learning. McGraw Hill.

Raschka, S., & Mirjalili, V. (2020). Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2. Packt Publishing.

Scikit-learn documentation: https://scikit-learn.org/

TensorFlow documentation: https://www.tensorflow.org/

Altunbaşak, Y., Polat, F., & Güzeliş, C. (2007). Veri Madenciliği: İstatistiksel Öğrenme ve Desicion Tree Yöntemleri. Papatya Yayıncılık.

Alpaydın, E. (2010). Veri Madenciliği: Makine Öğrenmesi ve İstatistiksel Desicion Tree Yöntemleri. Birsen Yayınevi.

Güvenir, H. A., & Günay, A. (2012). Veri Madenciliği: Kuram ve Uygulama. Seçkin Yayıncılık.

Aydın, H., & Güvenir, H. A. (2011). Veri Madenciliği ve Bulanık Mantık Yöntemleriyle Bankacılıkta Kredi Riski Analizi. Nobel Yayın Dağıtım.

Doğan, S., & Aydın, N. (2019). Makine Öğrenmesi: Veri Madenciliği, İstatistiksel Öğrenme ve Yapay Zeka. Detay Yayıncılık.

Güvenir, H. A., & Übeyli, E. D. (2013). Makine Öğrenmesi ve Veri Madenciliği. Seçkin Yayıncılık.

Polat, H., & Polat, H. (2020). Makine Öğrenmesi: İleri Düzey İstatistiksel Modelleme. Babil Yayıncılık.

Aydın, M. E. (2021). Makine Öğrenmesi: Teori ve Uygulama. Vize Yayınları.

Polat, H., & Polat, H. (2021). Kümeleme: Veri Madenciliği ve Makine Öğrenmesi Yöntemleri. Babil Yayıncılık.

Çakır, M., & Balcı, S. (2020). Makine Öğrenmesi Yöntemleriyle Kümeleme Analizi. İsmail Akgün Yayıncılık.

Polat, H., & Polat, H. (2019). Sınıflandırma: Veri Madenciliği ve Makine Öğrenmesi Yöntemleri. Babil Yayıncılık.

Öztürk, C. (2018). Python ile Makine Öğrenmesi. Kodlab Yayınları.

Çayır, B., & Şahin, F. (2019). Makine Öğrenmesi: Temel Algoritmalar ve Uygulamalar. Seçkin Yayıncılık.

Yıldırım, E., & Sert, N. (2019). Makine Öğrenmesi ve Veri Bilimi İçin Python: Derinlemesine Öğrenme ve Yapay Zeka. Seçkin Yayıncılık.

Öztürk, M., & Şahin, F. (2020). Python ile Makine Öğrenmesi: Sınıflandırma ve Kümeleme. Nobel Akademik Yayıncılık.

Übeyli, E. D., & İlçim, A. (2019). Biyomedikal Veri Analizi ve Makine Öğrenmesi. Seçkin Yayıncılık.

Polat, H., & Polat, H. (2021). Makine Öğrenmesi: İleri Düzey İstatistiksel Modelleme. Babil Yayıncılık.

Übeyli, E. D., & İlçim, A. (2019). Biyomedikal Veri Analizi ve Makine Öğrenmesi. Seçkin Yayıncılık.

Raschka, S., & Mirjalili, V. (2020). Python Makine Öğrenmesi. Medipol Üniversitesi Yayınları.

Übeyli, E. D., & İlçim, A. (2021). Python ile Biyomedikal Veri Analizi ve Makine Öğrenmesi. Nobel Akademik Yayıncılık.

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.

https://www.researchgate.net/publication/335524597_Makine_Ogrenmesi_Algoritmalari_ile_Hava_Kirliligi_Tahmini_Uzerine_Karsilastirmali_Bir_Degerlendirme

https://dergipark.org.tr/en/download/article-file/230830

https://dergipark.org.tr/en/download/article-file/56477

https://dergipark.org.tr/tr/download/article-file/579066

https://dergipark.org.tr/tr/download/article-file/387021

https://dergipark.org.tr/en/download/article-file/1149998

https://dergipark.org.tr/tr/download/article-file/2056203

https://dergipark.org.tr/tr/download/article-file/1347140

https://ab.org.tr/ab14/bildiri/98.pdf

https://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv?resource=download

https://www.kaggle.com/code/kellibelcher/customer-segmentation-and-clustering-analysis/input