LOTUS AI

YAPAY ZEKA VE BİLİŞİM ÇÖZÜMLERİ A.Ş.

# WATER QUALITY KÜMELEME RAPORU

HAZIRLAYANIN;

ADI: SENANUR
SOYADI: BAYRAM

02/12/2024

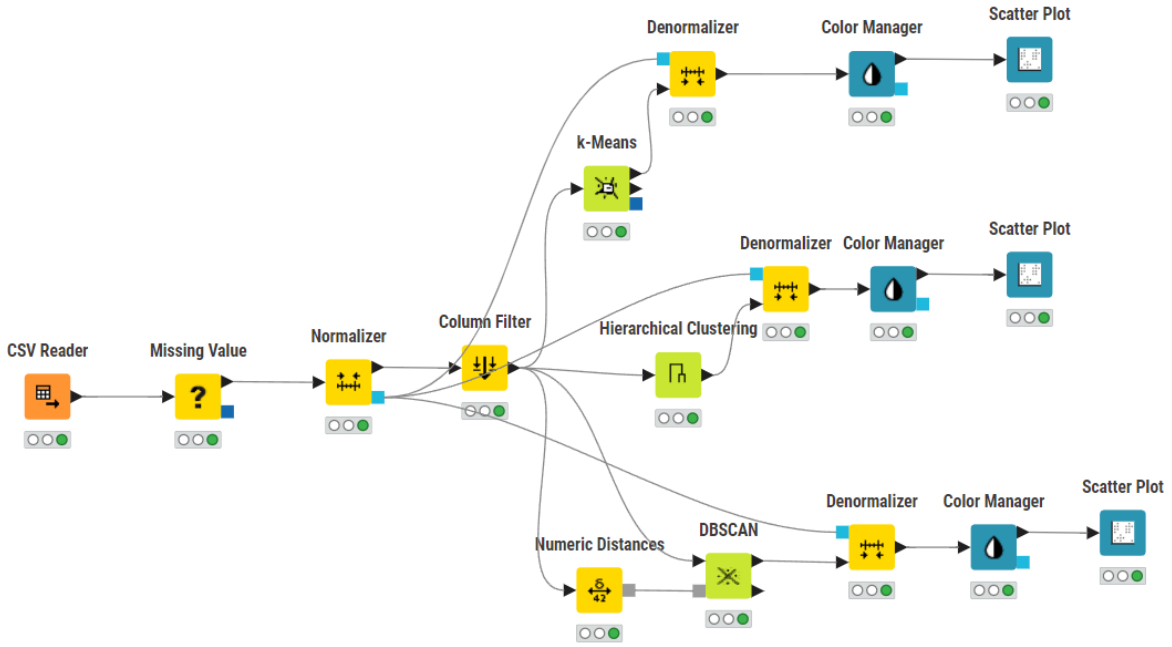
## Giriş

Su kalitesinin izlenmesi, çevresel faktörlerin değerlendirilmesi ve su kaynaklarının korunması için kritik bir öneme sahiptir. Bu çalışmada, su kalitesini etkileyen çeşitli çevresel parametrelerin analizi ve kümeleme yöntemleriyle değerlendirilmesi amaçlanmaktadır. Veri seti, suyun tuzluluk, çözünmüş oksijen, pH, sekki derinliği, su derinliği, su sıcaklığı ve hava sıcaklığı gibi temel özelliklerini içermektedir. Bu parametreler, su ekosistemlerinin sağlığını ve suyun kullanım amacına uygunluğunu belirlemek için önemli göstergelerdir.

Bu çalışma, verilerin kümeleme (clustering) yöntemleri ile analiz edilmesine dayanmaktadır. Kümeleme, benzer özelliklere sahip verileri gruplandırarak, çevresel faktörlerin su kalitesi üzerindeki etkilerini anlamaya yardımcı olur. Hierarchical clustering, dbscan ve k-means gibi yöntemler, verilerin arasındaki benzerlikleri belirleyerek su kalitesi hakkında bilgi sunar. KNIME ve Python platformları, bu kümeleme işlemlerinin gerçekleştirilmesinde kullanılacaktır. KNIME, görsel programlama arayüzü ile veri analizi sürecini kolaylaştırırken, Python ise güçlü makine öğrenmesi kütüphaneleri ile modelleme ve değerlendirme işlemleri için tercih edilecektir.

Bu analizle, su kalitesini etkileyen faktörlerin ve bu faktörlerle ilişkili kümelerin belirlenmesi amaçlanmaktadır. Kümeleme teknikleri, su kirliliği seviyelerinin ve çevresel değişkenlerin daha iyi anlaşılmasını sağlayarak, çevresel yönetim süreçlerine değerli bilgiler sunmaktadır. Sonuç olarak, bu proje, su kaynaklarının korunması ve yönetilmesi için analitik araçlar sunarak, su kalitesinin sürdürülebilir bir şekilde izlenmesini sağlayacak önemli bir adım teşkil etmektedir.

## 1. KNIME




Bu akışın amacı, bir veri kümesini ön işleme adımlarından geçirerek farklı kümeleme algoritmalarıyla analiz etmek ve sonuçları görselleştirmektir. Süreç, CSV formatındaki bir veri kümesini içe aktarma ve eksik değerlerin işlenmesiyle başlar. Daha sonra veriler, kümeleme algoritmalarının performansını artırmak için normalize edilir. İlgili sütunlar seçildikten sonra k-Means, Hiyerarşik Kümeleme ve DBSCAN gibi popüler kümeleme algoritmaları uygulanır. Her algoritma, veriyi özelliklerine göre gruplara ayırarak veri kümesi hakkında içgörüler sunar. Sonuçlar, normalize edilen verilerin orijinal ölçeğine döndürülmesi, renk kodlaması ve dağılım grafikleri ile görselleştirilir. Bu akış, verilerin gizli yapısını anlamak ve farklı kümeleme algoritmalarının sonuçlarını karşılaştırmak için güçlü bir araç sunar.

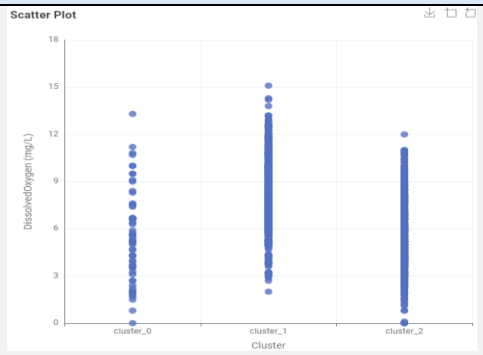
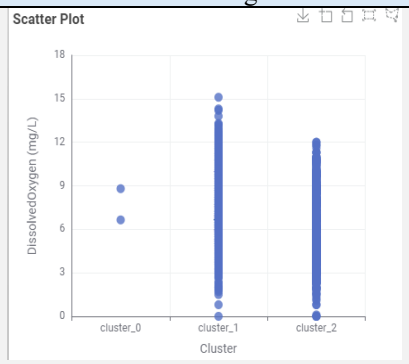
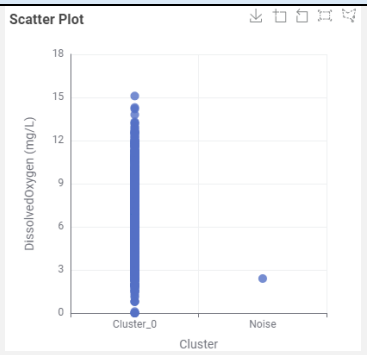
Aşağıda adım adım açıklaması yer almaktadır:

1. **CSV Reader**:
   - Veri kümesini bir CSV (Comma-Separated Values) dosyasından yükler. Bu düğüm, veri akışını başlatır ve diğer düğümlere giriş sağlar.
2. **Missing Value**:
   - Eksik veri noktalarını tespit eder ve bunları işlemek için belirli yöntemler uygular (örneğin, doldurma, silme veya diğer stratejiler).
3. **Normalizer**:

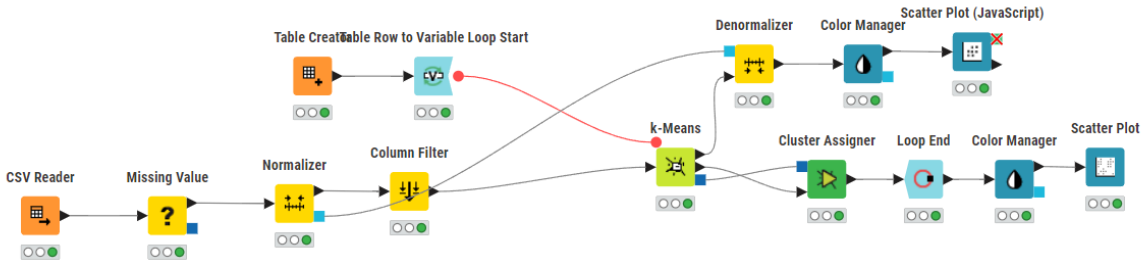
Veri setindeki değerleri belirli bir ölçeğe indirger (örneğin, 0 ile 1 arasında). Salinity (ppt), DissolvedOxygen (mg/L), pH, SecchiDepth (m), WaterDepth (m), WaterTemp (C), AirTemp (C) sütunlarını max min kullanarak normalize edilmiştir.

4. **Column Filter**:
   - Veri kümesindeki belirli sütunları seçmek veya dışlamak için kullanılır. Date sütunu filtrelenmiştir.
5. **k-Means**:
   - Veri noktalarını gruplara (kümeleme) ayıran bir makine öğrenimi algoritmasıdır. Veriler, özelliklerine göre k sayıda kümede toplanır.
6. **Denormalizer** (k-Means sonrası):
   - Normalizasyon işlemiyle ölçeklendirilmiş değerleri, orijinal ölçeklerine geri döndürür.
7. **Color Manager** (k-Means sonrası):
   - Verileri kategorilere ayırarak farklı grupları renklerle ifade eder, bu da görselleştirmeyi kolaylaştırır.
8. **Scatter Plot** (k-Means sonrası):
   - Verilerin dağılımını iki boyutlu bir düzlemde görselleştirir. Kümeleme sonuçlarını incelemek için kullanılır.
9. **Hierarchical Clustering**:
   - Hiyerarşik bir kümeleme yöntemi kullanarak veri noktalarını gruplara ayırır. Kümeleme yapısını ağaç benzeri bir biçimde (dendrogram) oluşturur.
10. **Numeric Distances**:
    - Veri noktaları arasındaki mesafeleri hesaplar. Özellikle mesafeye dayalı algoritmalarda (örneğin, DBSCAN) temel girdidir.
11. **DBSCAN**:
    - Yoğunluk tabanlı bir kümeleme algoritmasıdır. Kümeleme sınırlarını belirlemek için noktalar arasındaki mesafeleri kullanır.

**Scatter plot çıktıları:**

| Algoritmalar | K- Means | Hierarchical Clustering | Dbscan |
|---|---|---|---|
| Çıktılar |  |  |  |

Bir de AutoML kullanarak tekrar sonuçları inceleyelim:




Bu akış, farklı parametrelerle k-means kümeleme algoritmasını uygulayarak veri kümesinin yapısını analiz etmeyi ve bu yapıların görselleştirilmesini hedefler. Döngü, parametrelerin (örneğin küme sayısının) optimizasyonunu sağlar ve farklı senaryolarda kümeleme performansını değerlendirir. Elde edilen sonuçlar, hem ara süreçte hem de nihai olarak görselleştirilir. Bu, analizciye daha esnek bir kümeleme değerlendirmesi sunar. Aşağıda adım adım açıklaması yer almaktadır:

**1. CSV Reader**

- **Amaç**: Veri kümesini bir CSV dosyasından yükler ve işleme başlamak için ilk veriyi sağlar.

**2. Missing Value**

- **Amaç**: Eksik değerleri işler. Eksik veri noktalarını doldurabilir veya uygun bir şekilde temizleyebilir.

**3. Normalizer**

- **Amaç**: Veriyi belirli bir ölçeğe (örneğin, 0 ile 1 arası) indirger. Bu adım, kümeleme algoritmalarının daha etkili çalışması için önemlidir.

**4. Column Filter**

- **Amaç**: Veri setindeki ilgili sütunları seçer veya analizde gereksiz olan sütunları çıkarır.

**5. Table Creator**

- **Amaç**: Farklı parametreler (örneğin, k-means algoritması için küme sayısı) içeren bir tablo oluşturur. Bu parametreler döngü içinde kullanılacaktır.

**6. Table Row to Variable Loop Start**

- **Amaç**: Tabloyu satır satır okuyarak her döngüde bir satırdaki parametreleri değişkenlere dönüştürür. Bu sayede farklı parametrelerle işlem yapılır.

**7. k-Means**

- **Amaç**: k-means algoritmasını kullanarak verileri gruplara (kümelere) ayırır. Küme sayısı, döngü içinde belirlenen parametrelerden alınır.

**8. Denormalizer**

- **Amaç**: Normalizasyon sırasında ölçeklendirilmiş verileri orijinal değer aralığına döndürür.

**9. Color Manager**

- **Amaç**: Kümeleri renk kodlarıyla ifade eder. Görselleştirme ve analiz için grupların ayrımını kolaylaştırır.

**10. Scatter Plot (JavaScript)**

- **Amaç**: Verilerin iki boyutlu dağılımını bir grafik üzerinde gösterir. Bu, döngüdeki her parametre için ayrı bir görselleştirme sağlar.

**11. Cluster Assigner**

- **Amaç**: K-means algoritmasının sonuçlarını (her veri noktasının hangi kümeye ait olduğunu) veri setine yazar.

**12. Loop End**

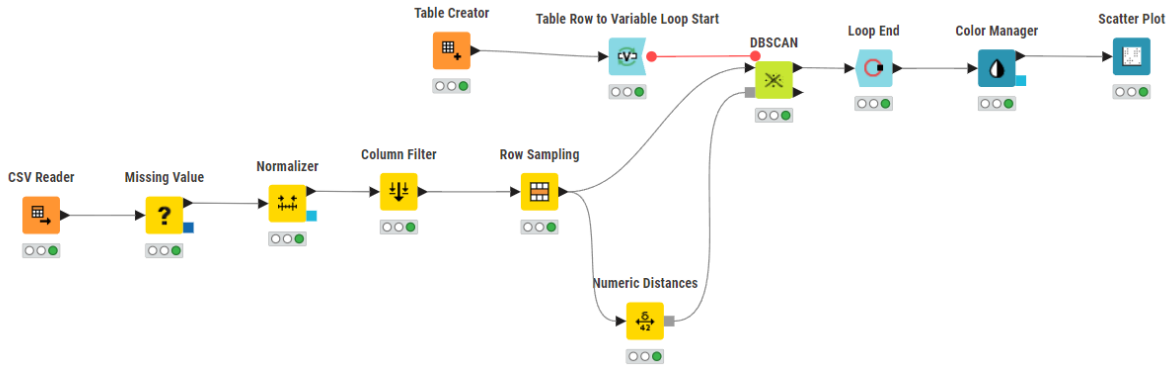
- **Amaç**: Döngüyü sonlandırır ve her parametre için yapılan işlemleri birleştirir.

**13. Color Manager (Sonuç Görselleştirme)**

- **Amaç**: Döngü sonunda elde edilen nihai kümeleri renk kodlarıyla ifade eder.

**14. Scatter Plot (Sonuç Görselleştirme)**

- **Amaç**: Döngüden çıkan nihai sonuçların görselleştirilmesini sağlar.




Bu akış, DBSCAN algoritmasını kullanarak veri kümesini yoğunluk tabanlı kümelere ayırmayı hedefler. Epsilon ve minimum nokta sayısı gibi parametreler için döngüyle farklı değerler test edilerek en iyi sonuçları elde etme amacı güdülür. İşlem sonucunda, kümeler görselleştirilir ve kullanıcı, parametrelerin etkisini analiz etme şansı bulur. Özellikle büyük veri kümelerinde gürültü noktalarını ayırt etmek ve yoğun kümeleri keşfetmek için etkili bir yöntem sunar. Aşağıda adım adım açıklaması yer almaktadır:

**1. CSV Reader**

- **Amaç**: Bir CSV dosyasından veri setini yükler ve işleme başlamak için giriş sağlar.

**2. Missing Value**

- **Amaç**: Veri setinde eksik değerleri tespit eder ve uygun yöntemlerle bu değerleri işler (örneğin, doldurma veya silme).

**3. Normalizer**

- **Amaç**: Verileri belirli bir ölçeğe (örneğin, 0 ile 1 arası) indirger. DBSCAN algoritması, uzaklıklara dayalı olduğundan, ölçeklendirme işlemi doğru çalışması için önemlidir.

**4. Column Filter**

- **Amaç**: Analiz için yalnızca gerekli sütunları seçer ve gereksiz olanları dışlar. Bu, algoritmanın yalnızca ilgili verilerle çalışmasını sağlar.

**5. Row Sampling**

- **Amaç**: Daha büyük veri setlerini temsil eden bir alt küme seçer. Bu, özellikle büyük veri setleriyle çalışılırken işlemlerin hızlanmasını sağlar.

**6. Numeric Distances**

- **Amaç**: Veri noktaları arasındaki mesafeleri hesaplar. DBSCAN, mesafeye dayalı bir algoritma olduğundan bu adım kritik öneme sahiptir.

**7. Table Creator**

- **Amaç**: DBSCAN algoritmasında kullanılan parametreler (örneğin, epsilon ve minimum nokta sayısı) için bir tablo oluşturur. Bu parametreler döngü içinde farklı değerlerle test edilecektir.

**8. Table Row to Variable Loop Start**

- **Amaç**: Parametreleri satır satır okuyarak her döngüde bir satırdaki değerleri değişkenlere dönüştürür. Böylece her döngüde DBSCAN farklı parametrelerle çalışır.

**9. DBSCAN**

- **Amaç**: Yoğunluk tabanlı bir kümeleme algoritmasıdır. Veriyi gruplara (yoğun kümeler) ve dışarıdaki noktaları (gürültü) ayırır. Döngüden gelen parametrelerle bu işlem tekrarlanır.

**10. Loop End**

- **Amaç**: Döngüyü sonlandırır ve tüm farklı parametre kombinasyonları için yapılan işlemleri birleştirir.

**11. Color Manager**

- **Amaç**: DBSCAN tarafından tespit edilen kümeleri renklerle ifade eder. Bu görselleştirme için kritik bir adımdır.

**12. Scatter Plot**

- **Amaç**: DBSCAN sonuçlarını bir grafik üzerinde görselleştirir. Kümeler ve gürültü noktaları bu grafikte kolayca ayırt edilir.

## 2. PYTHON

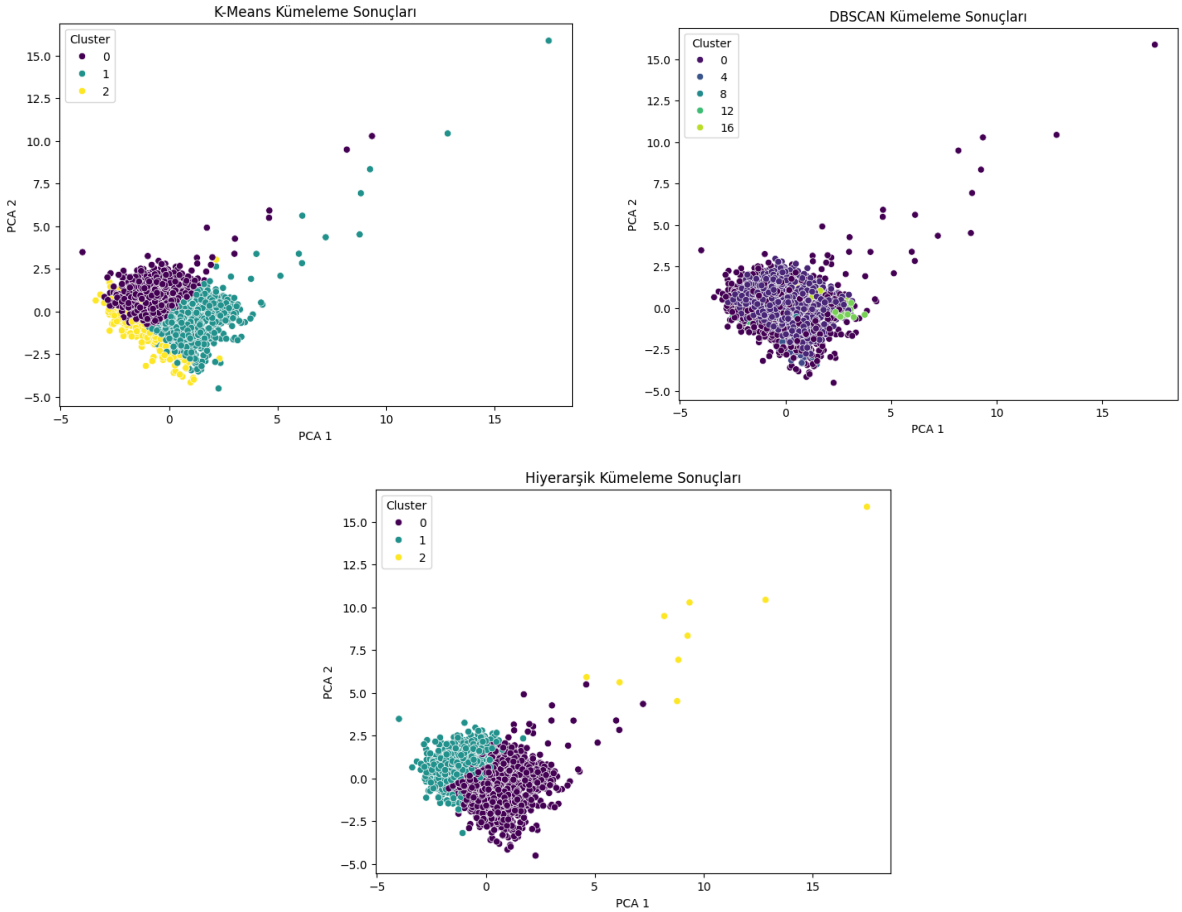
Bu Python projesinde, su kalitesi ile ilgili bir veri setinde üç farklı kümeleme algoritmasını (KMeans, DBSCAN ve Hiyerarşik Kümeleme) kullanarak veriyi gruplamayı amaçlamaktadır. Kodun temel amacı, veri setindeki belirli sayısal özellikler üzerinden kümeler oluşturmak, ardından bu kümeleri görselleştirerek kümelerin nasıl oluştuğunu incelemektir.

İlk olarak, veri setindeki eksik değerler çeşitli yöntemlerle doldurulmaktadır. Bu yöntemler arasında, eksik verilerin sütunlarındaki ortalama değerlerle (mean), ileri (forward) veya geri (backward) doldurma yöntemleri bulunmaktadır. Eksik veriler tamamlandıktan sonra, veriler normalize edilerek kümeleme algoritmalarına uygun hale getirilir. Normalizasyon, verilerin ölçek farklarını ortadan kaldırarak algoritmaların daha etkili çalışmasını sağlar.

Daha sonra, her bir kümeleme algoritması sırasıyla uygulanır. KMeans algoritmasında, küme sayısı ve başlatma yöntemi optimize edilmiştir. DBSCAN algoritması, yoğunluk tabanlı kümeleme yaparak, özellikle gürültüleri ayırmada etkilidir. Hiyerarşik Kümeleme algoritması ise veri noktalarını birbirine yakınlıklarına göre kümeler. Bu algoritmaların her biri, verinin doğal yapısını keşfetmeye çalışır.

Son olarak, her algoritmanın kümeleme sonuçları, boyut indirgeme tekniği olan PCA (Principal Component Analysis) kullanılarak iki boyutlu bir alana indirgenir ve her bir küme farklı renklerle görselleştirilir. Bu görselleştirme, algoritmaların veriyi nasıl grupladığını ve hangi noktaların hangi kümelere ait olduğunu daha anlaşılır kılar.

Aşağıda kümeleme çıktıları verilmiştir.



Son olarak su kalitesi verisi üzerinde üç farklı kümeleme algoritmasını (KMeans, DBSCAN ve Hiyerarşik Kümeleme) kullanarak kümeleme performanslarını K-Fold Cross Validation yöntemiyle değerlendirildi. Kodun amacı, her bir algoritmanın başarısını farklı veri alt kümeleri üzerinde test etmek ve bunları karşılaştırmak için kullanılan bir model doğrulama tekniği olan K-Fold Cross Validation'ı uygulamaktır.

İlk olarak, veri seti yüklenir ve eksik veriler sayısal sütunlarda ortalama ile doldurulur. Ardından, veriler normalize edilerek, algoritmaların farklı ölçeklerdeki verilere karşı etkili bir şekilde çalışabilmesi sağlanır. Kümeleme işlemi için KMeans, DBSCAN ve Hiyerarşik Kümeleme algoritmaları uygulanır ve her algoritmanın başarısı, veri setinin her bir katmanı üzerinde hesaplanan Silhouette skoru ile ölçülür. Silhouette skoru, bir kümeleme algoritmasının ne kadar iyi çalıştığını gösteren bir metrik olup, her bir veri noktasının kendi kümesine ne kadar yakın olduğunu ve diğer kümelerden ne kadar uzak olduğunu değerlendirir.

Son olarak, K-Fold Cross Validation ile her algoritmanın performansı 5 katlı (5-fold) bir geçerlilik testiyle belirlenir. Her bir algoritmanın ortalama Silhouette skoru hesaplanarak, kümeleme algoritmalarının genel başarısı değerlendirilir. Bu işlem, modelin genel performansını ve verinin yapısını daha güvenilir bir şekilde anlamak için kullanılır.

Silhouette skorları ise şu şekildedir:

| **Algoritmalar** | **K-Means** | **DBSCAN** | **Hierarchical Kümeleme** |
|---|---|---|---|
| **Silhouette Skorları** | 0.2349 | -0.2790 | 0.2127 |

Verilen Silhouette skorları, kümeleme algoritmalarının performansını değerlendirmek için önemli bir araçtır. Silhouette skoru, kümelerin içindeki homojenlik ve kümeler arasındaki ayrışmayı ölçer. Skor -1 ile +1 arasında değişir; +1, mükemmel bir kümelenmeyi, 0, belirgin olmayan bir kümelenmeyi, -1 ise kötü kümelenmeyi ifade eder.

Bu durumda, elde edilen sonuçlar şu şekildedir:

1. **KMeans Kümeleme Silhouette Skoru (K-Fold CV): 0.2349**
    - Bu değer, KMeans kümelemesinin orta seviyede bir başarıya sahip olduğunu gösterir. Skor 0'a yakın olduğunda, kümeler arasındaki ayrımın zayıf olduğu veya kümelerin çok overlapped olduğu anlamına gelir. Ancak, pozitif bir skor olduğundan, KMeans algoritması veri üzerinde belirli bir kümelenme düzeni ortaya çıkarmakta bir miktar başarı sağlamış.
2. **DBSCAN Kümeleme Silhouette Skoru (K-Fold CV): -0.2790**
    - Bu negatif skor, DBSCAN algoritmasının bu veri seti üzerinde iyi bir kümelenme yapamadığını gösterir. DBSCAN genellikle verideki yoğun bölgeleri algılayarak kümeler oluşturur, ancak bu durumda büyük ihtimalle veri setindeki gürültü ve/veya yoğunluk farklılıkları nedeniyle DBSCAN doğru kümeler oluşturamamış. Negatif bir Silhouette skoru, kümeler arasında belirgin bir ayrım olmadığını ve hatta bazı verilerin yanlış kümelerde yer almış olabileceğini işaret eder.
3. **Hierarchical Kümeleme Silhouette Skoru (K-Fold CV): 0.2127**
    - Hiyerarşik kümeleme algoritması da orta seviyede bir başarı göstermiştir. KMeans'e benzer şekilde, skor 0'a yakın ancak pozitif olması, bu algoritmanın bazı kümeler oluşturabildiğini ancak yine de kümeler arasındaki ayrımın tam olarak belirgin olmadığını gösteriyor. Bu yöntem, veri setindeki hiyerarşik yapıyı daha iyi yansıtabilir, ancak burada sınıflandırma çok güçlü olmamış.