

**YAPAY ÖĞRENMENİN TEMELLERİ PROJE RAPORU**

**ESMANUR YILDIZ 22010903062**

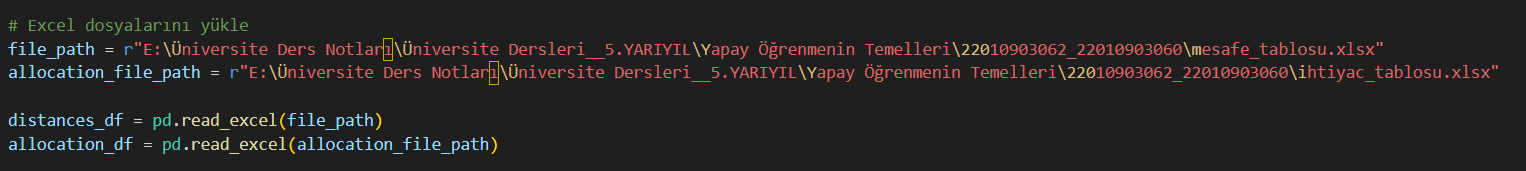
**SEDANUR PEKER 22010903060**

**Kütüphaneler:**

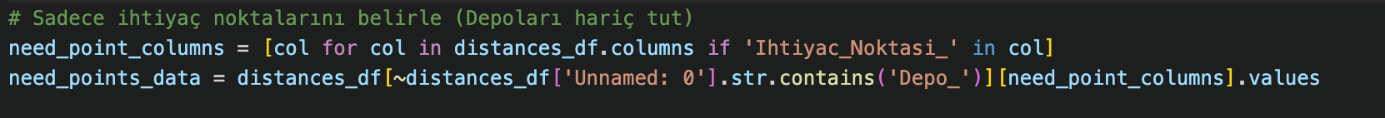
metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

* **Pandas:** Veri analizi ve manipülasyonu için kullanılan bir kütüphane. Excel dosyalarının okunması ve işlenmesi bu kütüphane ile yapılır.
* **NumPy:** Sayısal hesaplamalar ve dizi işlemleri için kullanılan bir kütüphane.
* **scikit-learn (sklearn.cluster.KMeans):** Makine öğrenimi algoritmalarını içerir. Burada K-means kümeleme algoritması için kullanılır.
* **Networkx:** Grafik ve ağ yapıları oluşturmak ve analiz etmek için kullanılan bir kütüphane. Mesafe tablosu üzerinden grafik oluşturulup, en kısa yollar hesaplanır.

**Excel dosyalarının yüklenmesi:**

* **Mesafe\_tablosu.xlsx**: Mesafeleri içeren tablo (mesafeler matrisi).
* **İhtiyaç\_tabosu.xlsx**: İhtiyaç noktalarındaki talep miktarlarını içeren tablo.

**İhtiyaç Noktalarının Belirlenmesi**:

* Depolar, lojistik süreçte malzemelerin depolandığı ve dağıtıldığı yerlerdir, ancak doğrudan ihtiyaç duyulan yerler değildir. Depoları dahil etmek, analizde gereksiz karmaşıklık yaratabilir.
* Kodun amacı, yardım malzemelerini ihtiyaç noktalarına ulaştırmak olduğu için analizin hedefinde ihtiyaç noktaları bulunur. Bu nedenle ihtiyaç noktaları belirlerken depoları hariç tutarız.

metin, çizgi, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu**Kümeleme (K-Means Algoritması):**

**Elbow Yöntemi Mantığı:**

Elbow yöntemi, k-ortalamalar (k-means) kümeleme algoritmasında optimal küme sayısını (K) belirlemek için kullanılır. Amaç, veri kümelerini birbirinden iyi bir şekilde ayrıştırabilecek en uygun K değerini seçmektir. Bunun için aşağıdaki mantık izlenir:

1. **Hata Değerini Ölçme (SSE - Sum of Squared Errors):**  
   Her bir küme için, kümeye dahil olan noktaların küme merkezine (centroid) olan uzaklıklarının karelerinin toplamı hesaplanır. Bu, her bir K değeri için tekrarlanır.
2. **Hata Azalımı (Eğim Değişimi):**  
   K artarken, her bir noktanın bir kümeye daha yakın olması beklendiğinden SSE değeri azalır. Ancak, belirli bir noktadan sonra (genelde bir "dirsek" gibi görünen yerde), SSE'nin azalım hızı önemli ölçüde düşer.
3. metin, yazı tipi, ekran görüntüsü, sayı, numara içeren bir resim

   Açıklama otomatik olarak oluşturuldu**Optimal Noktayı Belirleme (Dirsek Noktası):**  
   Grafikte SSE'nin azaldığı, ancak düşüş hızının yavaşladığı noktaya "dirsek" (elbow) denir. Bu nokta, ek küme eklemenin veriye anlamlı bir katkı sağlamadığı yeri temsil eder.

5 küme (K=5) noktasında SSE’nin azalım hızı yavaşladığı için optimal küme sayısı olarak K=5 seçilmiş.

**K-Means algoritması nedir?**

* Verileri, belirli sayıda (k) kümeye ayırır.
* Her küme, yakınlık temeline dayanarak bir merkez nokta (centroid) etrafında toplanır.
* Kümeleme işlemi, noktaların bir merkeze olan yakınlıklarını minimize etmeyi hedefler.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**1. K-Means Modelinin Tanımlanması**

* **KMeans**:
  + **n\_clusters=**num\_clusters: Algoritmanın kaç küme oluşturacağını belirler.
  + **random\_state=42**: Rastgelelik kontrolü sağlar, böylece algoritma her çalıştırıldığında aynı sonucu üretir.
  + **n\_init=10**: Algoritma, farklı başlangıç noktalarıyla 10 kez çalıştırılır ve en iyi sonucu seçer.

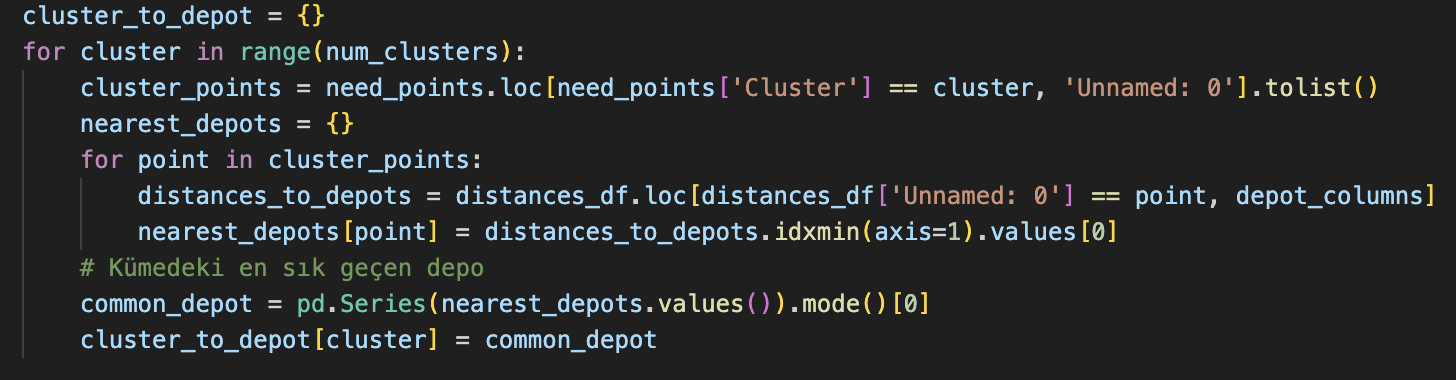
**2. Modelin Uygulanması**

* **fit\_predict(need\_points\_data)**:
  + İhtiyaç noktalarına ait veriler (need\_points\_data), algoritmaya verilir.
  + Algoritma, her noktayı bir kümeye atar.
  + Çıktı olarak, her nokta için bir küme etiketi döner.

**3. Küme Etiketlerinin Eklenmesi**

* **need\_points['Cluster']**:
  + Kümeleme sonuçları, ihtiyaç noktalarının bulunduğu veri setine yeni bir sütun olarak eklenir.
  + Her bir ihtiyaç noktası, hangi kümeye ait olduğunu gösteren bir etiket alır (örneğin: 0, 1, 2, ...).

**Her Küme İçin Depo Belirleme:**

****

* Her küme için en uygun depo belirlenir.
* İhtiyaç noktalarına olan mesafeler hesaplanır ve her kümede en sık geçen depo seçilir.
* Her küme için en uygun depo atanır ve bu bilgi daha sonraki aşamalarda kullanılır.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**Grafik Oluşturma:**

Bu kod bölümü, ihtiyaç noktaları ve depolar arasındaki mesafeleri temsil eden bir grafik (ağ) yapısı oluşturur. Amaç, rota ve mesafe optimizasyonu için kullanılan bir yapı hazırlamaktır. Kod, NetworkX kütüphanesi ile bir yönlü grafik (directed graph) oluşturur.

Grafik oluşturma işlemi, kodumuzun ilerleyen bölümlerindeki rota ve mesafe hesaplamaları için temel bir altyapı sağlar. Terminalde görsel bir grafik çıktısı üretmese de, bu grafik yapısı, algoritmaların arka planda çalışabilmesi için önemlidir.

**Rota ve Mesafe Hesaplama:**

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu bölüm, her küme için:

1. Optimum bir rota oluşturmayı,
2. Toplam mesafeyi hesaplamayı amaçlar.

Kod, Dijkstra Algoritması ve diğer optimizasyon tekniklerini kullanarak, ihtiyaç noktaları arasında en kısa yolu bulur ve toplam mesafeyi döner. Bu işlem, yardım malzemelerini taşımak için rotaların etkin bir şekilde planlanmasını sağlar.

**Dijkstra Algoritması Nedir?**

Dijkstra algoritması, bir grafikteki bir başlangıç noktasından (kaynak) diğer tüm düğümlere olan en kısa yolları bulmak için kullanılan bir algoritmadır. Bu algoritma, özellikle yönlendirilmiş veya yönlendirilmemiş, kenar ağırlıklı grafikleri çözmek için uygundur.

1. **Alt Grafik Oluşturma**

* **Amaç**: Belirli bir depo ve onunla ilişkili noktalar için alt bir grafik oluşturulur.
* depot\_name: Depo, rotanın başlangıç ve bitiş noktasıdır
* relevant\_points: Ziyaret edilecek ihtiyaç noktaları.

1. **Dijkstra Algoritması ile En Kısa Mesafeleri Hesaplama**

* **Amaç**: Depodan (depot\_name) diğer tüm noktalara en kısa mesafeyi hesaplamak.
* **weight="weight"**: Grafikte kenar ağırlıkları (mesafeler) kullanılarak hesaplama yapılır.

**3.Rotanın Oluşturulması**

**Başlangıç**:

* route: Hesaplanan rotayı depolayan liste. Depo başlangıç olarak eklenir.
* visited: Ziyaret edilen düğümleri takip eden bir küme.
* current\_node: Şu anda üzerinde çalışılan düğüm. Başlangıçta depo olarak ayarlanır.
* total\_distance: Rotanın toplam mesafesini depolayan değişken. Başlangıçta sıfırdır.current\_node, o anki düğümü temsil eder.

metin, yazı tipi, ekran görüntüsü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**En Yakın Noktanın Seçilmesi ve Rotaya Eklenmesi:**

rotayı oluştururken bir sonraki ziyaret edilecek noktayı belirlemek için kullanılır.

**Rotayı Güncelleme**:

* next\_node rotaya eklenir ve ziyaret edilmiş olarak işaretlenir.
* Mevcut düğüm, bir sonraki düğümle değiştirilir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**4. Depoya Dönüş**

* Rota tamamlandıktan sonra merkez noktaya geri dönülür.

**5. Çıktı**

* **route**: Ziyaret edilen noktaların sırası.
* **total\_distance**: Rotanın toplam mesafesi.

**Drone Yönetimi**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu bölüm, bir küme için hesaplanan rota ve ilgili talep miktarlarına dayanarak, drone'ların dağıtımı ve kapasitelerine uygun bir plan yapmayı hedefler. **Drone Yönetimi** fonksiyonu, her bir rotada ihtiyaçları karşılamak için kaç drone gerektiğini ve her bir drone'un taşıdığı yükü hesaplar.

**Drone Yönetiminin Amacı**

1. **Optimum Kaynak Kullanımı**:

* Drone kapasitelerini maksimum şekilde kullanarak fazla sefer yapılmasını önler.

1. **Zaman ve Maliyet Tasarrufu**:

* Gereken minimum drone sayısını hesaplayarak lojistik maliyetleri düşürür.

1. **Esneklik**:

* Farklı talep ve kapasitelere kolayca uyarlanabilir.

**1.Drone Kapasitesini Tanımlama**

* Drone başına taşıma kapasitesi tanımlanır (örneğin, 30 birim).
* Bu değer, droneların ne kadar yük taşıyabileceğini temsil eder.
* Rota üzerindeki her ihtiyaç noktası için:
  + **Tıbbi Malzeme Talebi (Medical Demand)** ve
  + **Yiyecek Talebi (Food Demand)** toplanır.
* Bu, her rota için toplam talebi temsil eder.
* Merkez noktası (başlangıç/dönüş noktası) bu hesaplamaya dahil edilmez.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**2.Drone Kullanımını Planlama**

* **while total\_demand > 0**:
  1. Toplam talep sıfırlanana kadar işlem tekrarlanır.
* **Drone Sayısı (drones\_used)**:
  1. Her yeni drone kullanımı için bu sayaç artırılır.
* **Taşıma Planı**:
  1. Eğer toplam talep, bir drone'un kapasitesinden küçükse:
     + Kalan tüm talep, bir drone tarafından karşılanır.
  2. Eğer toplam talep, bir drone'un kapasitesinden büyükse:
     + Drone maksimum kapasiteyi taşır, ve talep yeniden güncellenir.

**Tüm Süreçlerin Entegrasyonu**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Kod, her bir küme için:

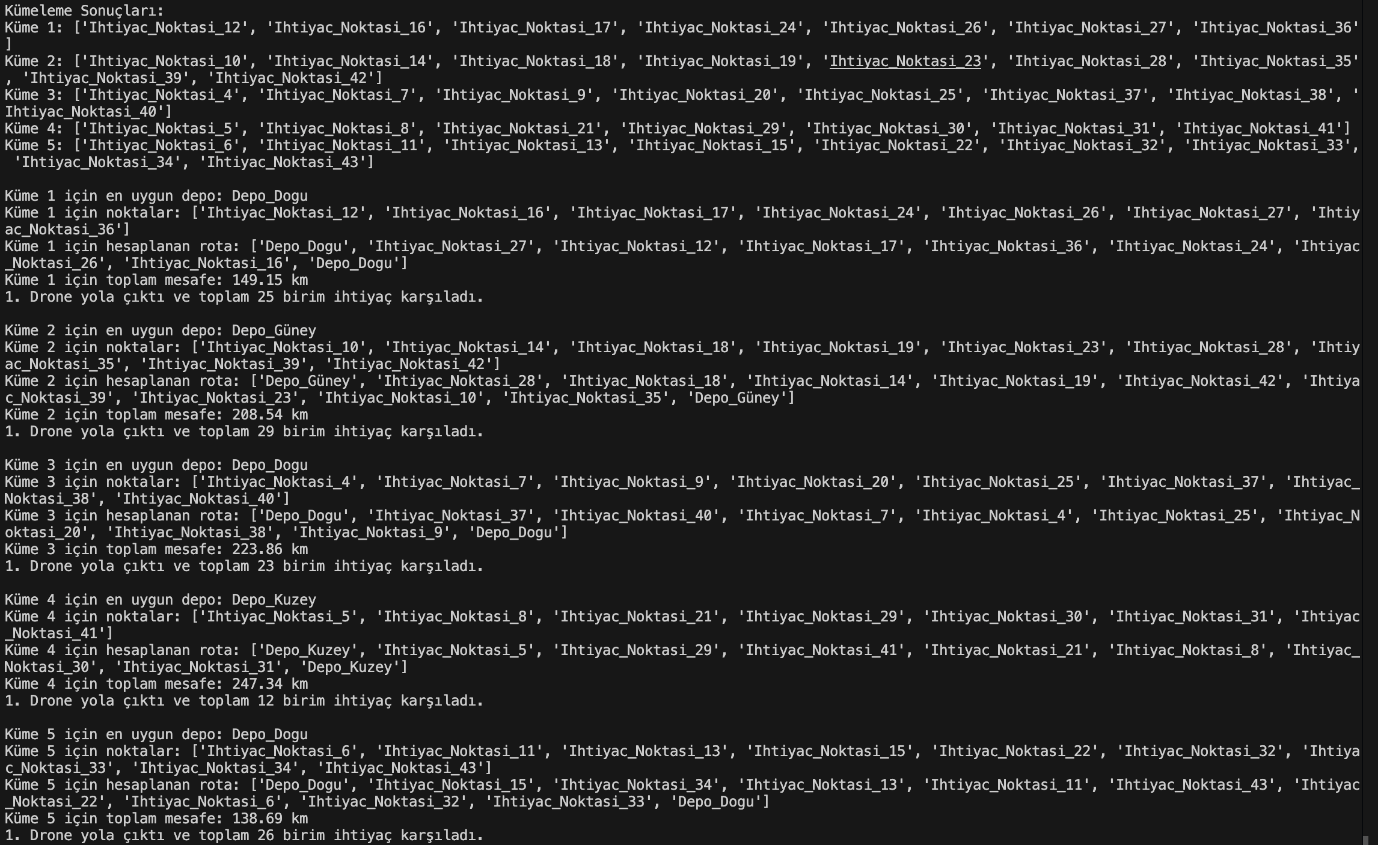
1. Küme noktalarını belirliyor.
2. Rota ve toplam mesafeyi hesaplıyor.
3. Drone yönetimini gerçekleştiriyor.

**Kodun Çıktıları:**

1. **Kümeleme Sonuçları**: Hangi noktanın hangi kümeye ait olduğu.
2. **Rota ve Mesafeler**: Her küme için optimal rota ve mesafe.
3. **Drone Yönetimi**: Her rota için gereken drone sayısı ve taşıma kapasitesi.

Kod, ihtiyaç noktalarını kümelere ayırarak her küme için etkili bir rota planı ve drone kullanım planı oluşturur.

**Ekran Çıktısı**

****