SAKARYA
UYGULAMALI BİLİMLER
ÜNİVERSİTESİ

Adı :Mehmet Emin

Soyadı :BOZDOĞAN

Öğrenci No :23010903051

Adı :Emirhan

Soyadı :Güngör

Öğrenci No:23010903097

Ödev: Bu projede, bir deprem sonrası insani yardım malzemelerinin (tıbbi malzeme ve yiyecek) ihtiyaç noktalarına en hızlı ve verimli şekilde ulaştırılması amaçlanmıştır. Toplamda 45 adet ihtiyaç noktası ve 3 adet dağıtım deposu (Kuzey, Güney, Doğu) belirlenmiştir. Dağıtımın otonom dronlar ile yapılması planlanmış olup, her dronun taşıma kapasitesi 30 birim olarak sınırlandırılmıştır. Projenin temel hedefi, bu kısıtlar altında en kısa rotayı ve en uygun yükleme stratejisini belirleyen bir yapay zeka sistemi geliştirmektir.

Akadamisyen: Doç. Dr. Ekin EKİNCİ

Arş. Gör. Furkan ATBAN

GİTHUB:
https://github.com/Mehmetettt646131/yapay_ogrenme_dersiproje_SUBU

Veri_hazirlik.py

1. Koordinatların Üretilmesi:

İlk olarak, simülasyonun gerçekleştirileceği 100x100 km'lik sanal bir alan tanımlanmıştır. Bu alan üzerinde 45 adet ihtiyaç noktası ve 3 adet dağıtım deposu için rastgele X ve Y koordinatları (enlem-boylam benzeri) üretilmiştir. Bu koordinatlar, hem noktaların harita üzerinde görselleştirilmesine olanak sağlamış hem de mesafe hesaplamaları için temel referans noktası olmuştur. Elde edilen ham konum verileri 'koordinatlar.xlsx' dosyasına kaydedilmiştir.


koordinatlar.xlsx

2. Mesafe Matrisinin Hesaplanması:

Üretilen koordinatlar kullanılarak, tüm noktaların birbirine olan uzaklıkları hesaplanmıştır. 48x48 boyutundaki bu devasa veri matrisinin manuel yöntemlerle hatasız oluşturulması mümkün olmadığından; Python'un 'NumPy' kütüphanesi kullanılarak noktalar arası Öklid (Euclidean) mesafeleri hassas bir şekilde hesaplanmış ve 'mesafeler.xlsx' dosyasına aktarılmıştır.


mesafeler.xlsx

3. Talep Miktarlarının Belirlenmesi:

Afet yönetimindeki belirsizliği modellemek amacıyla, her bir ihtiyaç noktası için rastgele talep miktarları atanmıştır. Python'un 'random' modülü ile her nokta için 1-10 birim Tıbbi Malzeme ve 1-15 birim Yiyecek ihtiyacı belirlenmiş ve bu veriler 'talepler.xlsx' dosyasına işlenmiştir. Bu sayede, her noktanın yükünün farklı olduğu (heterojen) ve dronların 30 birimlik taşıma kapasitesini zorlayacak dinamik bir senaryo kurgulanmıştır.


talepler.xlsx

Dersodevv.py

İhtiyaç Noktalarının Kümelenmesi (K-Means Algoritması)

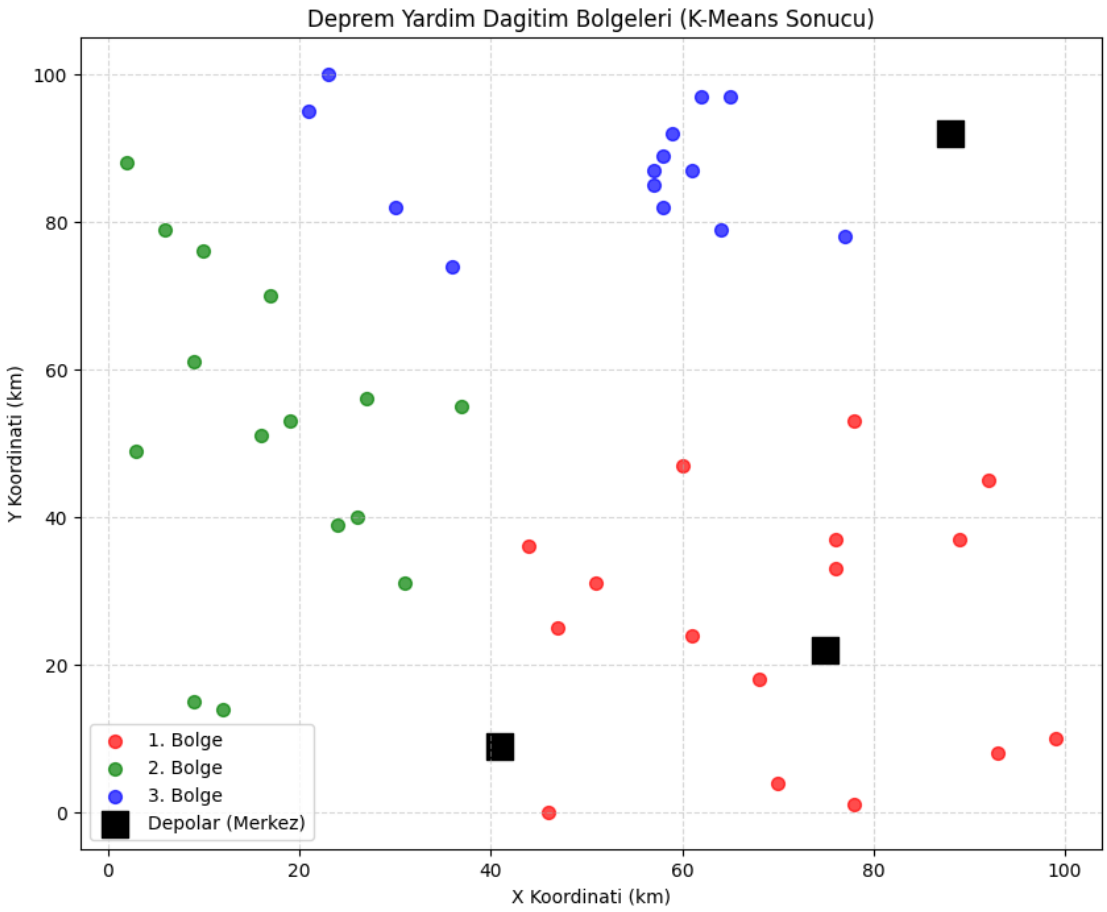
Veri seti oluşturulduktan sonra, 45 adet dağınık ihtiyaç noktasının 3 farklı lojistik merkezden (Depo) yönetilebilmesi için gruplandırılması gerekmiştir. Bu işlem için proje dosyasında belirtilen **'Dersodevv.py'** kod dosyası içinde **K-Means Kümeleme Algoritması** kullanılmıştır.

Algoritma şu adımları izlemiştir:

1. **Giriş Verisi:** veri_hazirlik.py tarafından üretilen 'koordinatlar.xlsx' dosyası okunmuştur.
2. **Kümeleme:** Depo sayısı 3 olduğu için küme sayısı **K=3** olarak belirlenmiş ve noktalar coğrafi yakınlıklarına göre 3 ana bölgeye ayrılmıştır.
3. **Çıktı:** Her noktanın hangi bölgeye (Küme 0, Küme 1, Küme 2) ait olduğu belirlenmiş ve bu bilgiler **'kumelenmis_veri.xlsx'** dosyasına kaydedilmiştir.


kumelenmis_veri.xlsx

Ayrıca algoritmanın başarısını görselleştirmek için **'kumeleme_haritasi.png'** oluşturulmuştur. Haritada her bölge farklı bir renkle (Kırmızı, Yeşil, Mavi) temsil edilerek, lojistik sınırların net bir şekilde çizilmesi sağlanmıştır.



Neden K-Means

1. Depo Sayısının Sabit Olması (Neden DBSCAN Seçilmedi?):

Proje gereksinimlerinde 3 adet dağıtım deposu kesin olarak belirtilmiştir.

- **DBSCAN:** Yoğunluk tabanlı bir algoritmadır ve küme sayısını (K) baştan belirlememize izin vermez. Verinin yoğunluğuna göre 2 küme de bulabilir, 5 küme de. Ayrıca DBSCAN, seyrek bölgelerdeki noktaları "gürültü" (noise) olarak işaretleyip dışlayabilir. Ancak insani yardım senaryosunda, en uzaktaki (seyrek) noktaya bile yardım gitmesi zorunludur.
- **K-Means:** Küme sayısının (K=3) baştan verilmesini zorunlu kılar. Bu özellik, elimizdeki 3 depo kaynağına göre sahanın tam olarak 3 bölgeye bölünmesi gerekliliğiyle birebir örtüşmektedir.

2. Kesin Atama Gerekliliği (Neden GMM Seçilmedi?):

- **Gaussian Mixture Model (GMM):** Olasılıksal bir modeldir ve "Yumuşak Kümeleme" (Soft Clustering) yapar. Yani bir noktanın %60 A deposuna, %40 B deposuna ait olduğunu söyler.
- **K-Means:** Lojistik planlamada bir noktanın hangi araç tarafından gidileceğinin net olması gerekir ("Sert Kümeleme / Hard Clustering"). K-Means, her noktayı kesin olarak tek bir merkeze atadığı için operasyonel planlamaya daha uygundur.

3. İşlem Hızı ve Geometrik Uygunluk (Neden Agglomerative Seçilmedi?):

- **Agglomerative Clustering:** Hiyerarşik bir yapı kurar ancak veri sayısı arttıkça işlem maliyeti (O(n^3)) K-Means'e (O(n)) göre çok daha yüksektir.
- **K-Means:** İhtiyaç noktaları coğrafi bir düzlem üzerindedir ve Öklid mesafesi kullanılarak yapılan küresel (spherical) gruplandırmalar, coğrafi bölge paylaşımı (Voronoi diyagramı mantığı) için en ideal yöntemdir.

4. **Elbow Method :**
   Elbow Method tek başına bir kümeleme algoritması değil, K-Means için ideal küme sayısını bulmaya yarayan bir yardımcı yöntemdir. Ancak projede küme sayısı (Depo sayısı = 3) zaten fiziksel bir kısıt olarak verildiği için, optimum kümeyi aramaya gerek kalmamış, doğrudan K=3 değeri kullanılmıştır.

Sonuç olarak; sabit küme sayısı kısıtı, her noktanın mutlaka bir kümeye dahil edilme zorunluluğu ve kesin sınırların belirlenmesi gerekliliği nedeniyle K-Means algoritması tercih edilmiştir.

Rota_optimizasyonu.py

Ben ödevimi yaparken Aco(Karınca Kolonisi) algoritmasını kullandım.

Neden Dijkstra veya A* (A-Star) Değil de ACO (Karınca Kolonisi) Kullanıldı?

Bu en kritik soru. Cevabı problemin **türüyle** ilgili:

- **Neden Dijkstra Değil?**
    - **Dijkstra'nın Amacı:** A noktasından B noktasına giden *tek bir en kısa yolu* bulmaktır (Örneğin: Evden okula gitmek).
    - **Senin Problemin (TSP):** Senin problemin A'dan B'ye gitmek değil; A, B, C, D, E... noktalarının hepsine uğrayıp **hangi sırayla** gidileceğini bulmaktır.
    - **Sonuç:** Dijkstra, "Hangi sırayla uğrarsam yol en kısa olur?" sorusunu çözemez. Sadece iki nokta arasındaki mesafeyi ölçer.
- ***Neden A* (A-Star) Değil?***
    - ***A*'ın Amacı**: Dijkstra'nın daha akıllı versiyonudur. Hedefe ne kadar yaklaştığını tahmin ederek (heuristic) labirentte yol bulur.
    - **Senin Problemin:** Ortada bir labirent veya engel yok. Senin sorunun "permütasyon" (sıralama) sorunu.
    - **Sonuç:** A* algoritması da nokta-nokta arası rota bulur, 50 tane müşteriyi hangi sırayla dizeceğini hesaplamak için tasarlanmamıştır.
- **Neden ACO (Karınca Kolonisi)?**
    - **Kombinatoryal Problem:** Müşteri sayısı arttıkça olası rota sıralamaları milyarları bulur. Tek tek denemek imkansızdır.
    - **Doğadan İlham:** Karıncalar rastgele yollar dener, kısa yolu bulanlar arkasında "feromon" (iz) bırakır. Diğer karıncalar bu izi takip eder.

- **Uyumluluk:** ACO, bu tip "Geçilecek 50 nokta var, en iyi sırayı bul" (TSP) problemleri için en iyi **sezgisel (heuristic)** yöntemlerden biridir. Kesin sonucu garanti etmez ama çok kısa sürede "mükemmele yakın" sonucu verir.

Kod çalıştığında hedef_klasor içine şu dosyaları bırakır:

Excel Dosyaları

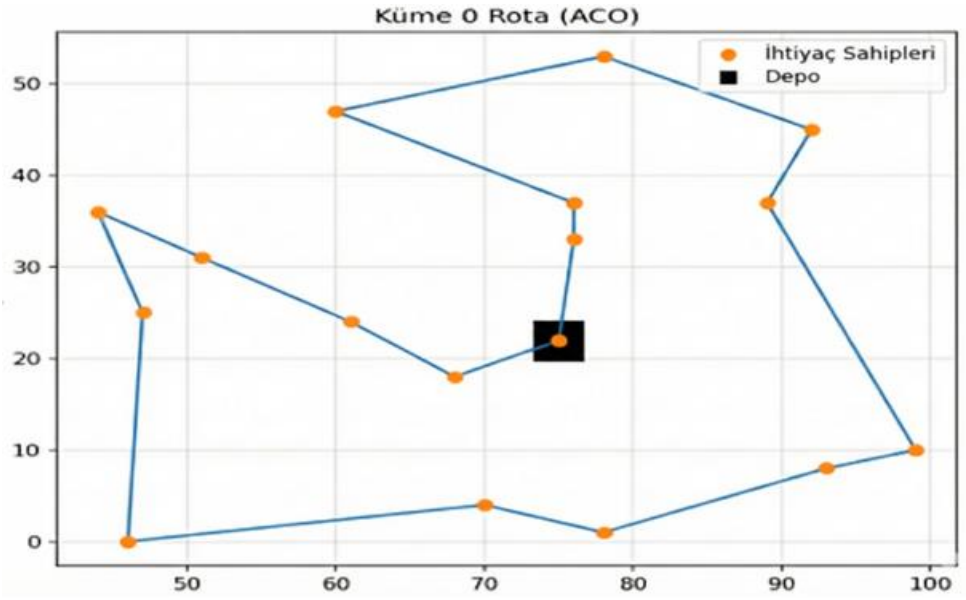
Ne işe yarar?: Her bir araç (küme) için drona verilecek yol haritasıdır.

İçeriği:

- order: Gidilecek sıra numarası (0, 1, 2...).
- id: Müşteri veya deponun kimlik numarası.
- x, y: Koordinatlar

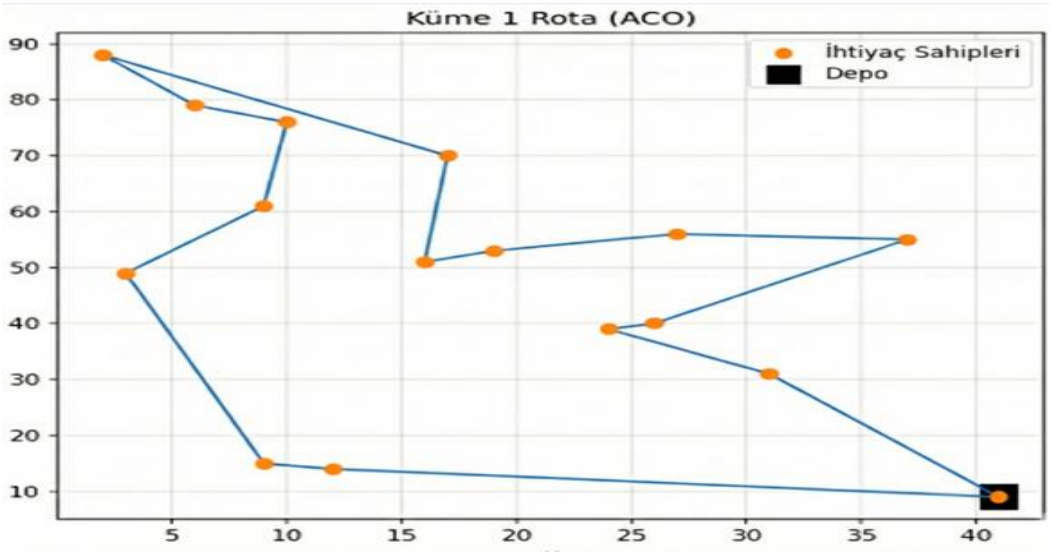
Neden var?: Drona "Malzemeleri kafana göre dağıt." denmez. Bu liste, optimizasyonun belirlediği "önce buraya, sonra şuraya git" emridir.

rota_kume_0_aco.xlsx: 1. Depoya bağlı dronun uçuş kartıdır.



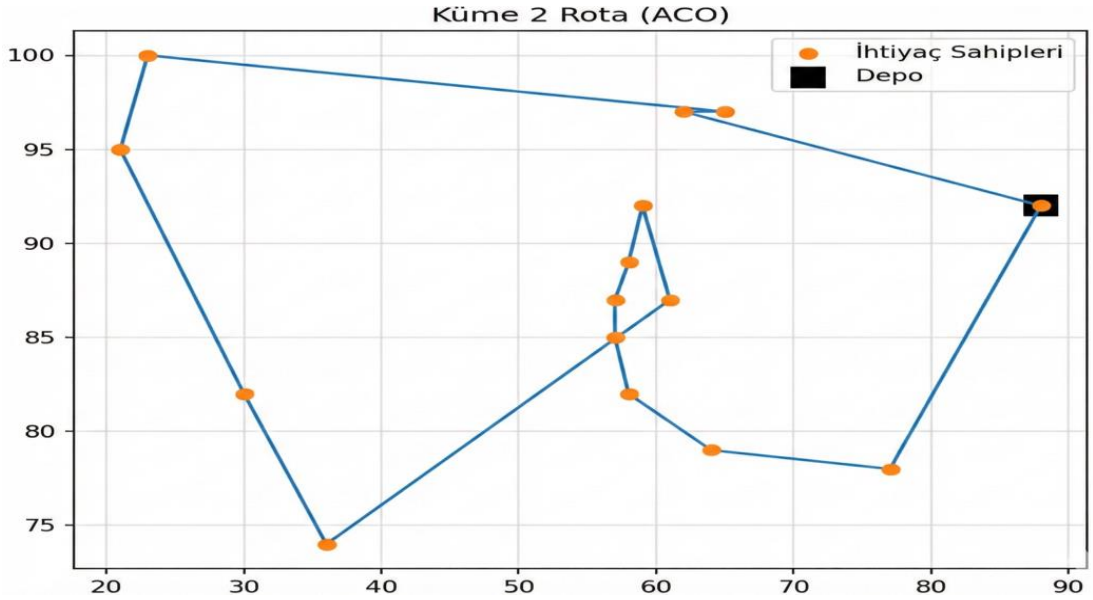
rota_kume_0_aco.xlsx

rota_kume_1_aco.xlsx: 2. Depoya bağlı dronun uçuş kartıdır.



rota_kume_1_aco.xlsx

rota_kume_2_aco.xlsx: 3. Depoya bağlı dronun uçuş kartıdır.



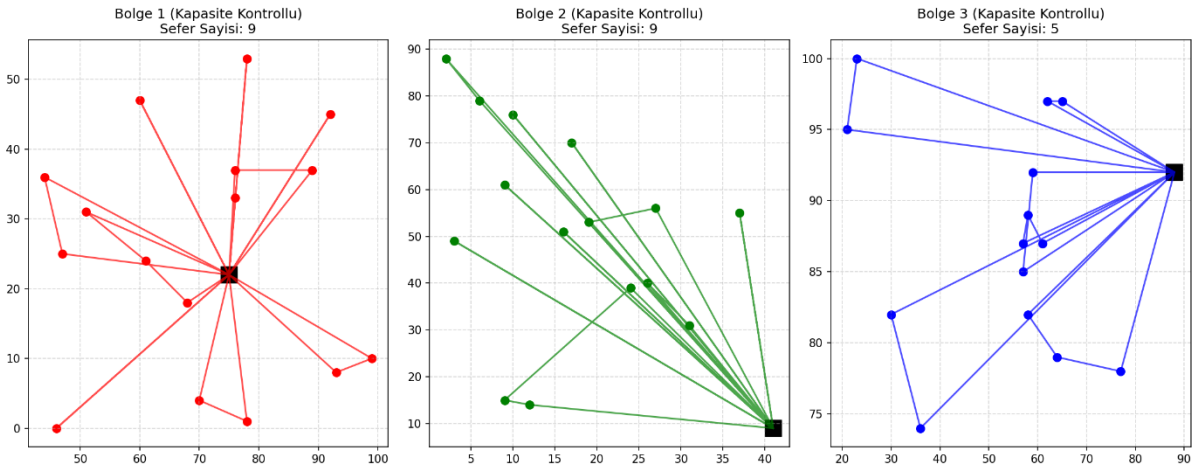
rota_kume_2_aco.xlsx

rota_ozeti_aco.xlsx: En son 3 tablonun özetidir.

rota_ozeti_aco.xlsx

yukleme_stratejisi.py

- **Algoritma:** Kod, "En Yakın Komşu" (Nearest Neighbor) mantığı ile çalışır ancak buna bir **Kapasite Kontrolü (Capacity Constraint)** ekler.
- **Mantık:**
    1. Dron depodan boşa çıkar.
    2. En yakın ihtiyaç noktasına gider.
    3. Eğer (Mevcut Yük + Yeni Talep) > 30 ise; **DUR!** Depoya dön, yükü boşalt (sıfırla), sonra o noktaya git.
    4. Kapasite yetiyorsa devam et.



Excel Raporları:

Sütunların Anlamı:

- Adim: Hareket sırası.
- ID: Gidilen yerin kimliği.
- Tip: Burası çok önemli. **"DEPO (Dolum)"** yazan satırlar, dronun kapasitesi dolduğu için merkeze döndüğü anları gösterir.
- Talep: O noktaya bırakılan yük miktarı.

kapasiteli_dagitim_ozeti.xlsx: Bu tablo, sistemin genel performans metriklerini özetlemektedir. 3 farklı operasyon bölgesine atanan dronların toplam iş yükü ve lojistik gereksinimleri burada raporlanmıştır.

kapasiteli_dagitim_ozeti.xlsx

kapasiteli_rota_bolge_1.xlsx: Kuzey Bölgesi (Bölge 1) için oluşturulan dinamik rota planıdır. 'DEPO (Dolum)' etiketli satırlar, dronun kapasitesinin (30 birim) dolduğu ve ikmal için merkeze döndüğü anları temsil eder.

kapasiteli_rota_bolge_1.xlsx

kapasiteli_rota_bolge_2.xlsx: Güney Bölgesi (Bölge 2) uçuş detaylarıdır. Bu bölgedeki birim talep miktarları yüksek olduğu için dron daha sık aralıklarla depoya dönmek zorunda kalmıştır.

kapasiteli_rota_bolge_2.xlsx

kapasiteli_rota_bolge_3.xlsx: Doğu Bölgesi (Bölge 3) rota planıdır. Toplam talep diğer bölgelere göre düşük (95 birim) olduğu için sefer sayısı daha az gerçekleşmiştir.

kapasiteli_rota_bolge_3.xlsx

Final_tablo.py

Bu modül, projenin önceki aşamalarında elde edilen ham operasyonel verilerin karar vericilere sunulmak üzere **otomatik, görsel ve anlaşılır** bir formata dönüştürülmesini sağlar. Manuel veri işleme hatalarını ortadan kaldırarak, projenin "Yükleme Stratejisi" sonuçlarını standartlaştırılmış bir tablo halinde sunar.

Insani Yardim Dagitim - Operasyonel Ozet

Bolge	Top_Talep	Sefer_Sayisi	Top_Mesafe
1	2	16	201
2	1	15	216
3	0	14	150

Kullanılan Kaynaklar

- https://gemini.google.com
- https://www.python.org
- https://scikit-learn.org/stable/modules/clustering.html#k-means
- https://youtu.be/gQC51rlndpc?si=xa61a7Te0D2xAH7R