VERİ BİLİMİ PROJE RAPORU

1. GİRİŞ

Bu rapor, İstanbul'un yoğun ve karmaşık kentsel trafik ağındaki dinamikleri anlamlandırmak ve optimize etmek üzere hazırlanmıştır. Projenin temel amacı, elde edilen kapsamlı trafik verilerini analiz ederek, trafik yoğunluğu davranışlarında yinelenen ve belirgin kalıpları tanımlamaktır. Bu kalıpların ortaya konması, trafik sıkışıklığının temel sebeplerini, zamansal değişimlerini ve coğrafi dağılımını derinlemesine kavramak için stratejik bir öneme sahiptir.

Çalışma kapsamında, hız, araç sayısı, günün saati ve haftanın günleri ana trafik karakteristikleri kullanılarak, trafik akışını temsil eden farklı kümelenmeler oluşturulacaktır. Bu kümeleme sonuçları, şehir planlamacılarına, trafik yönetim birimlerine ve ilgili paydaşlara, trafik sorunlarına yönelik daha isabetli, veri odaklı ve bölgesel/zamansal olarak hedeflenmiş çözümler geliştirebilmeleri için somut ve uygulanabilir içgörüler sunmayı amaçlamaktadır.

2. VERİ SETİ TANIMI

Bu proje, İstanbul'un trafik yoğunluğunu analiz etmek amacıyla derlenmiş geniş kapsamlı bir veri setini temel almaktadır. 'traffic_density_202501.csv' dosyasından çekilen bu veri seti, Ocak 2025 dönemine ait İstanbul Trafik Verilerini içermektedir. Veri setinde yer alan temel değişkenler ve açıklamaları aşağıda sunulmuştur:

- **DATE_TIME**: Her bir gözlemin kaydedildiği tarih ve zaman bilgisidir; trafik akışının zamansal desenlerini çıkarmak için kritik bir özelliktir.
- **LATITUDE** ve **LONGITUDE**: Trafik verisinin toplandığı coğrafi enlem ve boylam koordinatlarını ifade eder.
- **GEOHASH**: Konum bilgisinin kısa, kodlanmış bir temsilidir.
- **MINIMUM_SPEED**: Belirli bir zaman aralığında kaydedilen en düşük trafik hızıdır.
- **MAXIMUM_SPEED**: Belirli bir zaman aralığında kaydedilen en yüksek trafik hızıdır.
- **AVERAGE_SPEED**: Gözlem anındaki ortalama trafik hızıdır. Bu değişken, trafik akışının dinamiklerini doğrudan yansıttığı ve sıkışıklık düzeylerini belirlemede temel bir gösterge olduğu için kümeleme analizinde merkezi bir rol oynamıştır.
- **NUMBER_OF_VEHICLES**: Belirli bir zaman diliminde algılanan araç sayısını belirtir ve bölgedeki trafik yoğunluğunun önemli bir ölçüsüdür.

Veri setinin yüksek hacimli yapısı ve detaylı coğrafi-zamansal bilgileri barındırması, trafik davranışlarındaki karmaşık örüntüleri kümeleme analizi aracılığıyla ortaya çıkarmak için oldukça elverişli bir zemin sunmaktadır. Özellikle LATITUDE, LONGITUDE, AVERAGE_SPEED ve NUMBER_OF_VEHICLES gibi sütunlar, trafik akışının temel karakteristiklerini doğrudan yansıttığı ve küme oluşumunda belirleyici olduğu için modelleme sürecinde ana özellikler olarak kullanılmıştır.

3. VERİ HAZIRLAMA

Bu aşama, ham trafik veri setinin kümeleme analizi için uygun, temiz ve anlamlı bir formata dönüştürülmesini içeren kritik ön işleme adımlarını kapsamaktadır. Veri kalitesini artırmak, modelin performansını optimize etmek ve sonuçların doğruluğunu sağlamak amacıyla çeşitli dönüşümler ve iyileştirmeler uygulanmıştır.

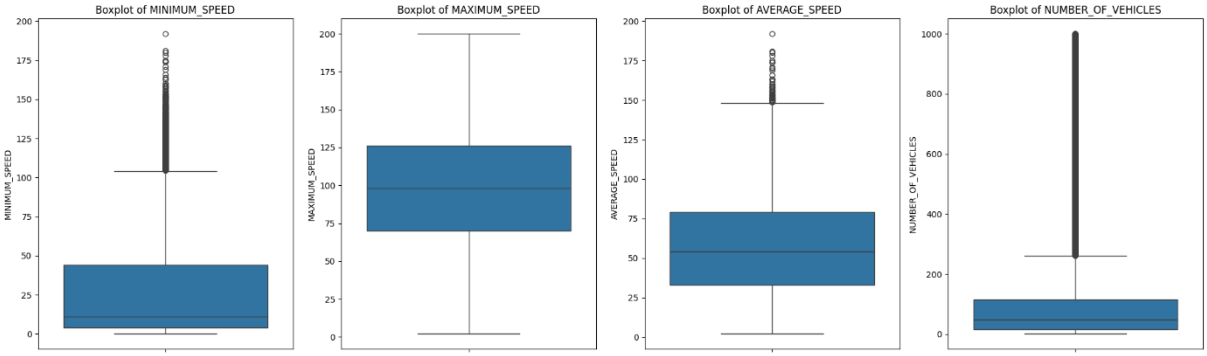
3.1.Veri Setinin İlk İncelemesi ve Eksik Değerlerin Yönetimi:

Veri seti yüklemesinin ardından ilk olarak genel yapısı, sütun tipleri ve eksik değer durumları incelenmiştir. info() ve isnull().sum() gibi temel fonksiyonlar kullanılarak veri setindeki eksik gözlemlerin varlığı kontrol edilmiştir. Başlangıçta tespit edilen eksik değerlerin, ilgili sütunların medyan (ortanca) değerleri ile doldurulmasına karar verilmiştir. Medyanın tercih edilme nedeni, aykırı değerlerden daha az etkilenerek veri dağılımının bozulmasını en aza indirmesidir.

3.2.Aykırı Değerlerin Temizlenmesi:

Sayısal özellikler olan MINIMUM_SPEED, MAXIMUM_SPEED, AVERAGE_SPEED ve NUMBER_OF_VEHICLES sütunlarında, fiziksel olarak gerçekçi olmayan veya aşırı değerler içeren aykırı gözlemler tespit edilmiştir. Bu aykırı değerler, kümeleme algoritmasının performansını ve küme merkezlerinin doğruluğunu olumsuz etkileyebilir. Bu sorunu gidermek amacıyla, sektörel ve mantıksal sınırlar belirlenmiştir:

- Hız değişkenleri (MINIMUM_SPEED, MAXIMUM_SPEED, AVERAGE_SPEED) için değerler 0 ile 200 km/s aralığında sınırlandırılmıştır. Bu aralığın dışındaki değerler NaN olarak işaretlenmiştir.
- Araç sayısı (NUMBER_OF_VEHICLES) için değerler 0 ile 1000 araç aralığında sınırlandırılmıştır. Bu aralığın dışındaki değerler yine NaN olarak işaretlenmiştir. Bu belirlenen NaN değerleri, ilgili sütunların ortalama değerleri (mean) ile doldurularak veri bütünlüğü sağlanmış ve aykırı değerlerin etkisi minimize edilmiştir. Aykırı değer temizliği öncesindeki hız değişkenlerinin dağılımlarını ve potansiyel aykırı değerlerini görselleştiren kutu grafikleri aşağıda sunulmuştur:



Görsel 3.2: Hız Değişkenlerine Ait Kutu Grafikleri (Aykırı Değer Temizliği Öncesi). Bu grafikler, Minimum Hız, Maksimum Hız ve Ortalama Hız değişkenlerinin dağılımlarını ve üst uçtaki yoğun aykırı değerleri göstermektedir.

3.3.Zaman Bilgisinin Çıkarımı ve Yeni Özelliklerin Üretilmesi:

DATE_TIME sütunu, doğrudan kümeleme için kullanılamayacak ham bir formattadır. Trafik yoğunluğu desenlerinin günün saatine, haftanın gününe ve hafta sonu durumuna göre önemli ölçüde değiştiği bilindiğinden, bu sütundan zamansal olarak anlamlı yeni özellikler türetilmiştir:

- HOUR: Her bir gözlemin gerçekleştiği saati (0-23 arası) temsil eder. Trafiğin gün içindeki zirve ve sakin saatlerini yansıtır.
- DAY_OF_WEEK: Haftanın hangi günü olduğunu (Pazartesi=0, Pazar=6) belirtir. Hafta içi ve hafta sonu trafik karakteristiklerindeki farkları yakalamak için önemlidir.
- WEEKEND: Gözlemin hafta sonuna (1) mı yoksa hafta içine (0) mı ait olduğunu gösteren ikili bir değişkendir. Özellikle hafta sonu gezileri ve iş trafiği ayrımını sağlar.
- MONTH: Gözlemin yapıldığı ayı (Ocak=1) belirtir. Mevsimsel trafik değişimleri için faydalıdır.
  DAY_OF_MONTH: Ayın kaçıncı günü olduğunu belirtir.

Bu türetilen zamansal özellikler, trafik davranışlarının zamana bağlı varyasyonlarını daha etkin bir şekilde yakalamak ve kümeleri bu farklılıklara göre daha iyi ayrıştırmak için modelleme sürecine dahil edilmiştir.

3.4.Nihai Özellik Seçimi ve Veri Ölçekleme:

Kümeleme analizi için kullanılacak nihai özellik seti, veri setinin hem coğrafi hem de zamansal ve trafik akışına ilişkin tüm ilgili boyutlarını kapsayacak şekilde dikkatlice seçilmiştir. Seçilen özellikler şunlardır: LATITUDE, LONGITUDE, AVERAGE_SPEED, NUMBER_OF_VEHICLES, HOUR, DAY_OF_WEEK, WEEKEND ve DAY_OF_MONTH.

Farklı ölçü birimlerine ve değer aralıklarına sahip bu özelliklerin (örn. enlem-boylam koordinatları ile araç sayısı veya hız değerleri) kümeleme algoritması üzerindeki orantısız etkisini önlemek amacıyla, tüm seçili özellikler StandardScaler kullanılarak normalleştirilmiştir. Bu işlem, her özelliğin ortalamasını sıfıra ve standart sapmasını bire eşitleyerek, algoritmanın tüm özelliklere eşit ağırlık vermesini ve öklid mesafesi tabanlı kümeleme metotlarının doğru çalışmasını sağlamıştır. Ölçekleme sonrası veri matrisi (X_scaled), boyut indirgeme ve K-Means kümeleme adımları için optimize edilmiş ve hazır hale getirilmiştir.

4. MODELLEME

Bu bölümde, ham veri setinden elde edilen özelliklerin analizi ve kümelenmesi için kullanılan Temel Bileşen Analizi (PCA) ve K-Means kümeleme algoritmalarının detayları sunulmaktadır. Amacımız, trafik verisi içinde doğal olarak oluşan desenleri ve trafik yoğunluğu karakteristiklerini temsil eden anlamlı kümeler oluşturmaktır.

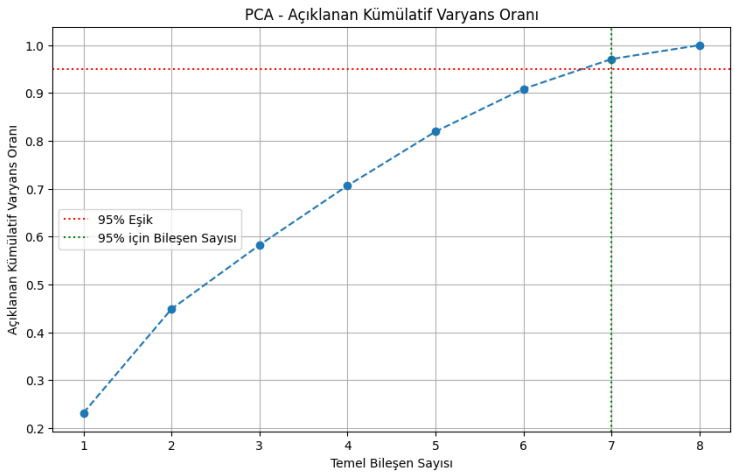
4.1.Boyut İndirgeme: Temel Bileşen Analizi (PCA) Uygulaması

Trafik veri setimiz, konum (LATITUDE, LONGITUDE), hız bilgileri (MINIMUM_SPEED, MAXIMUM_SPEED, AVERAGE_SPEED), araç sayısı (NUMBER_OF_VEHICLES) ve türetilmiş zamansal özellikler (HOUR, DAY_OF_WEEK, WEEKEND, MONTH, DAY_OF_MONTH) gibi birden fazla boyutu içermektedir. Bu yüksek boyutluluk, kümeleme algoritmasının performansını olumsuz etkileyebilir ve özellikle görselleştirme ve yorumlama süreçlerini zorlaştırabilir. Bu zorlukları aşmak için, veri kümesine Temel Bileşen Analizi (PCA) uygulanmıştır.

PCA, veri setindeki orijinal değişkenlerin doğrusal kombinasyonlarından oluşan, birbirine dik (ortogonal) ve en fazla varyansı açıklayan yeni bileşenler (Temel Bileşenler - Principal Components) üretir. Bu sayede, veri setindeki bilginin büyük bir kısmı korunarak boyut sayısı azaltılır ve gereksiz gürültü etkisi minimize edilir.

Projemizde, PCA'nın uygulanması iki temel amaca hizmet etmiştir:

1.Varyans Açıklaması ve Bilgi Korunumu: Veri setindeki toplam varyansın önemli bir yüzdesini (genellikle %80-%95) açıklayan minimum temel bileşen sayısını belirlemek, veri setinin içsel karmaşıklığını ve bilgi yoğunluğunu anlamamızı sağlar. Bu analiz için, tüm olası temel bileşenler hesaplanmış ve her birinin açıkladığı varyans oranı ile kümülatif varyans oranı incelenmiştir. Aşağıdaki grafik, temel bileşen sayısı arttıkça kümülatif açıklanan varyansın nasıl değiştiğini göstermektedir:



Görsel 4.1: PCA Açıklanan Kümülatif Varyans Oranı

Bu grafik, veri setindeki bilgi kaybını minimumda tutarak kaç temel bileşenin orijinal varyansın önemli bir kısmını (%95) açıkladığını göstermektedir. Grafikteki kesikli çizgi ve nokta, %95 eşiğini ve bu eşiğe ulaşmak için gereken bileşen sayısını belirtmektedir.

Yapılan analizde, toplam varyansın %95'ini açıklamak için **7** adet temel bileşene ihtiyaç duyulduğu tespit edilmiştir. Bu sonuç, veri setimizin, orijinal boyutluluğuna kıyasla, önemli bir bilgi kaybı olmadan önemli ölçüde indirgenebileceğini göstermektedir.

2.K-Means Kümeleme ve Görselleştirme İçin Boyut İndirgeme: K-Means kümeleme algoritmasının performansını artırmak ve elde edilen kümeleri 2 boyutlu bir uzayda kolayca görselleştirebilmek amacıyla, veri seti nihai olarak 2 temel bileşene indirgenmiştir: (optimal_pca_components = 2). Bu 2 bileşen, orijinal veri setinin en kritik varyasyonlarını yakalayarak kümelerin daha net ayrışmasına olanak tanımıştır. PCA'dan geçirilmiş bu veri (X_pca), daha sonra K-Means algoritmasının girişini oluşturmuştur.

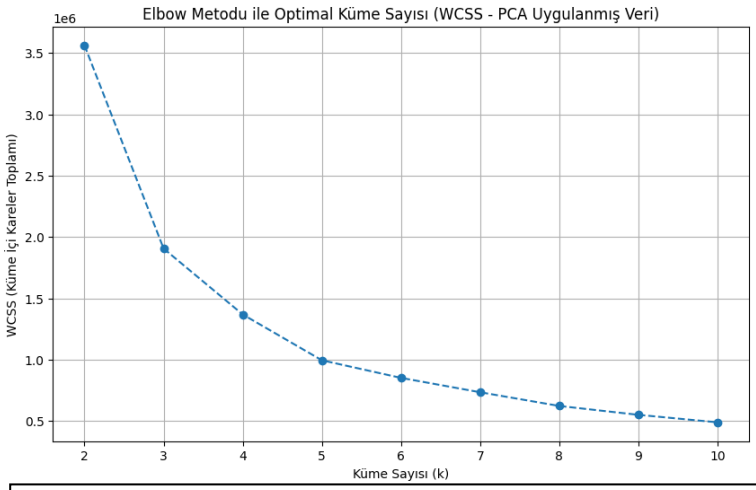
4.2. Kümeleme: K-Means Algoritması Uygulaması

Boyut indirgeme sonrası elde edilen X_pca veri setine K-Means kümeleme algoritması uygulanmıştır. K-Means, belirlenen bir k değeri etrafında veri noktalarını gruplayan, uzaklık tabanlı bir denetimsiz öğrenme algoritmasıdır. Algoritma, küme içi benzerliği (yani, küme içindeki noktaların birbirine yakınlığını) maksimize etmeyi, aynı zamanda kümeler arası farklılığı da maksimize etmeyi hedefler. Bunu da Küme İçi Kareler Toplamı (WCSS) metriği ile değerlendireceğiz.

K-Means algoritması için k (küme sayısı) değerini belirlemek amacıyla Elbow (Dirsek) Metodu kullanılmıştır. Bu metot, farklı k değerleri için WCSS değerlerini hesaplar ve bu değerleri bir grafik üzerinde görselleştirir. WCSS, k arttıkça doğal olarak azalma

eğilimindedir, çünkü daha fazla küme ile veri noktaları küme merkezlerine daha yakın olabilir. Ancak, WCSS'deki düşüşün belirgin bir şekilde yavaşladığı, grafiğin "dirsek" gibi kırıldığı nokta, optimal k değerini işaret eder. Bu noktadan sonra küme eklemek çok az iyileşme sağlayacağı için optimum değer olarak kabul edilir.

Aşağıdaki grafik, Elbow Metodunun görselleştirmesini ve optimal k değerinin nasıl belirlendiğini sunmaktadır:



Görsel 4.2: Elbow Metodu ile Optimal Küme Sayısı (WCSS - PCA Uygulanmış Veri)

Bu grafik, farklı küme sayıları (k) için hesaplanan Küme İçi Kareler Toplamı (WCSS) değerlerini göstermektedir. Grafikteki belirgin "dirsek" noktası, veri setindeki doğal gruplamaları en iyi şekilde yansıtan optimal küme sayısını işaret eder.

Bu Elbow grafiği incelendiğinde, WCSS değerindeki düşüşün belirgin şekilde yavaşladığı "dirsek" noktası gözlemlenmiştir. Bu noktaya göre optimal küme sayısı 4 olarak belirlenmiştir. Bu değer, hem küme içi homojenliği sağlamak hem de küme sayısını makul bir seviyede tutmamızı sağlayacaktır.

4.2.2. K-Means Uygulaması ve Küme Ataması:

Belirlenen optimal k değeri (optimal_k = [k değeri]) kullanılarak K-Means algoritması, PCA ile indirgenmiş X_pca veri setine uygulanmıştır. Algoritma, belirlenen küme merkezlerini bulmak ve her bir veri noktasını en yakın küme merkezine atamak için 10 farklı başlangıç noktası (n_init=10) ile çalıştırılmış ve en iyi sonuç (init='k-means++') seçilmiştir.

Kümeleme sonucunda, her bir trafik gözlemi için bir küme etiketi atanmış ve bu etiketler (Cluster_PCA sütunu) orijinal df_processed DataFrame'ine yeni bir sütun olarak

eklenmiştir. Bu etiketler, her bir trafik olayının hangi belirlenen kümede yer aldığını göstermektedir. Kümeleme sonuçlarının dağılımı aşağıdaki gibidir:

Her bir kümedeki veri noktası sayısı (PCA sonrası):

Küme Dağılımı: Cluster_PCA

0 288279 adet

1 803530 adet

2 166021 adet

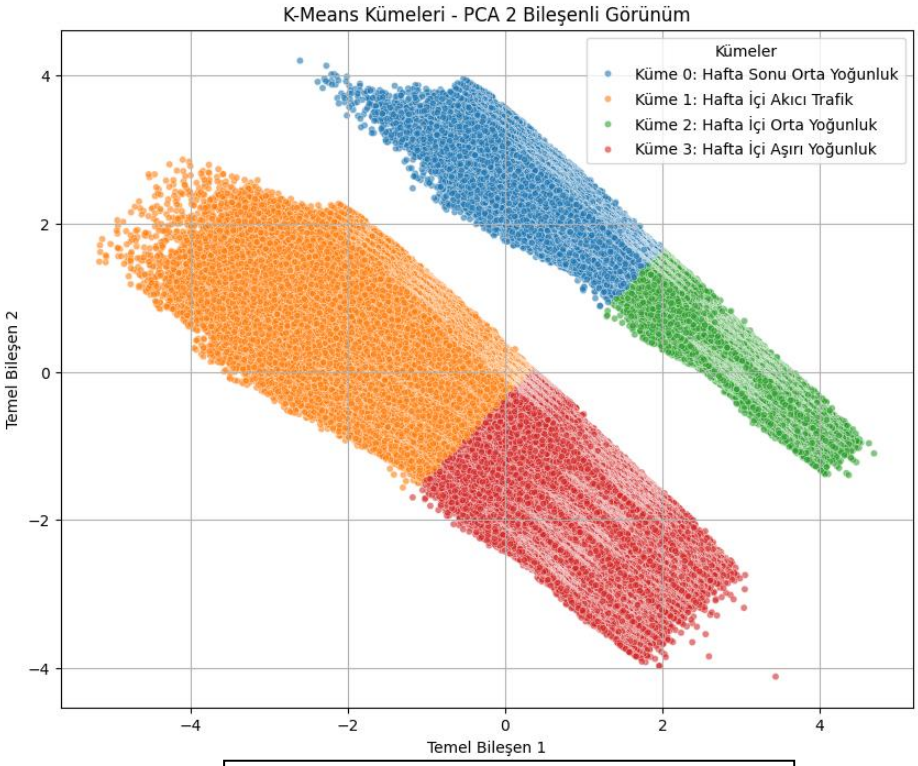
3 506133 adet

Name: count, dtype: int64

4.2.PCA Bileşenleri Üzerinde Küme Sonuçlarının Görselleştirilmesi

Elde edilen K-Means kümelerini daha iyi anlamak ve görselleştirmek amacıyla, PCA ile 2 boyuta indirgenmiş veri (X_pca) üzerinde kümelerin dağılımı bir dağılım grafiği (scatter plot) ile gösterilmiştir. Bu görselleştirme, farklı kümelerin temel bileşenler uzayında nasıl ayrıştığını ve kümeleme algoritmasını gözlemlememizi sağlar.

Aşağıdaki grafik, K-Means tarafından atanan küme etiketlerine göre veri noktalarının PCA'nın ilk iki temel bileşeni üzerindeki dağılımını göstermektedir:



Görsel 4.3: K-Means Kümeleri - PCA 2 Bileşenli Görünüm.

Bu grafik, PCA'nın ilk iki temel bileşeni üzerinde kümeleme sonuçlarını renk kodlamasıyla göstermektedir. Her renk, farklı bir trafik kümesini temsil etmekte olup, benzer özelliklere sahip gözlemlerin aynı kümede gruplandığını ve kümeler arası ayrımı ortaya koymaktadır.

5. SONUÇLAR

Bu bölüm, veri ön işleme, boyut indirgeme (PCA) ve kümeleme (K-Means) gibi analitik adımlar sonucunda elde edilen bulguları ve nicel çıktıları içermektedir. Bu sonuçlar, trafik verisi içindeki desenleri ve karakteristikleri doğrudan ortaya koyan veriye dayalı bulgulardır.

5.1. Veri Hazırlık ve Özellik Mühendisliği Bulguları

- **Eksik ve Aykırı Değer Yönetimi:** Veri setindeki eksik ve aykırı değerler, veri kalitesini artırmak ve analitik güvenilirliği sağlamak amacıyla ilgili değişkenlerin medyan/ortalama değerleriyle başarıyla yönetilmiştir. Özellikle MINIMUM_SPEED, MAXIMUM_SPEED, AVERAGE_SPEED ve NUMBER_OF_VEHICLES sütunlarındaki temizlik işlemleri gerçekleştirilmiştir. Bu süreçte hız değişkenleri ve araç sayısı dağılımları kutu grafikleri ile görselleştirilmiştir (Bkz. **Görsel 3.1: Hız ve Araç Sayısı Değişkenlerine Ait Kutu Grafikleri**). Bu sayede analiz için tutarlı ve gürültüden arındırılmış bir veri seti elde edilmiştir.
- **Yeni Özellik Türetimi:** DATE_TIME bilgisinden türetilen HOUR, DAY_OF_WEEK, WEEKEND, MONTH, ve DAY_OF_MONTH gibi zamana dayalı yeni özellikler veri setine eklenmiştir. Bu özellikler, trafik davranışlarındaki zamansal dinamikleri ve mevsimsel/günsel varyasyonları yakalamayı hedeflemiştir.

5.2. Boyut İndirgeme (PCA) Bulguları

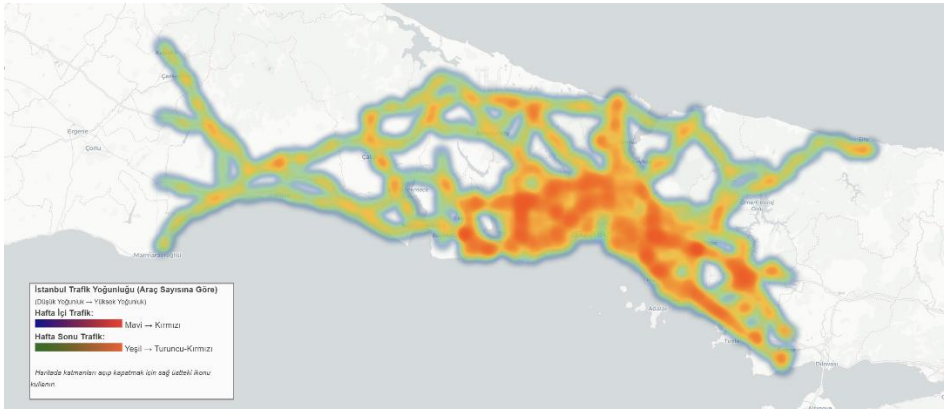
- **Varyans Açıklaması:** PCA analizi sonucunda, veri setindeki toplam varyansın %95'ini açıklamak için **7** adet temel bileşene ihtiyaç duyulduğu tespit edilmiştir. Bu bulgu, orijinal veri setindeki bilginin büyük bir kısmının daha az sayıda boyutta temsil edilebileceğini göstermektedir (Bkz. **Görsel 4.1: PCA - Açıklanan Kümülatif Varyans Oranı**).
- **Kümeleme İçin Boyut İndirgeme:** K-Means kümelemesi için veri, algoritma verimliliğini artırmak ve küme sonuçlarının 2 boyutlu uzayda görselleştirilmesine olanak tanımak amacıyla 2 temel bileşene indirgenmiştir (X_pca).

5.3. K-Means Kümeleme Bulguları

- **Optimal Küme Sayısının Belirlenmesi:** Elbow Metodu kullanılarak yapılan analizde, WCSS (Küme İçi Kareler Toplamı) değerindeki belirgin "dirsek" noktasına dayanarak optimal küme sayısı **4** olarak belirlenmiştir. Bu sonuç, trafik verisi içinde dört temel ve anlamlı trafik deseni bulunduğunu işaret etmektedir (Bkz. **Görsel 4.2: Elbow Metodu ile Optimal Küme Sayısı (WCSS - PCA Uygulanmış Veri)**).
- **Küme Dağılımı:** K-Means uygulamasının ardından, gözlemler belirlenen 4 kümeye göre gruplandırılmış ve Cluster_PCA sütunu oluşturulmuştur. Her bir kümedeki veri noktası sayısı aşağıdaki gibidir:
    - Küme 0: 288279 adet
    - Küme 1: 803530 adet
    - Küme 2: 166021 adet
    - Küme 3: 506133 adet
- **Küme Yapılarının Görsel Ayrımı:** PCA'nın ilk iki temel bileşeni üzerinde küme dağılımı görselleştirildiğinde, 4 kümenin sanal boyut uzayında belirgin bir şekilde ayrıştığı gözlemlenmiştir (Bkz. **Görsel 4.3: K-Means Kümeleri - PCA 2 Bileşenli Görünüm**). Bu görsel bulgu, kümeleme algoritmasının veri setindeki gizli desenleri başarılı bir şekilde yakaladığının güçlü bir kanıtıdır.

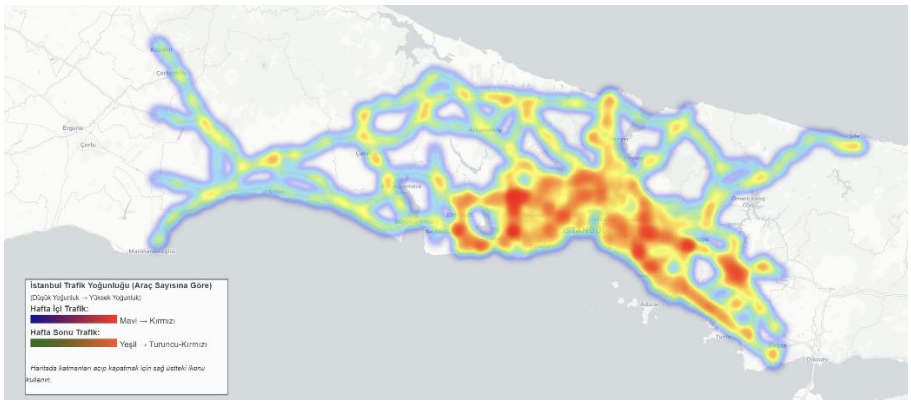
5.4. Trafik Yoğunluğu Haritaları (Folium) Bulguları

- Folium kullanılarak oluşturulan ısı haritaları(heatmap), trafik yoğunluğunun coğrafi dağılımını somutlaştırmıştır. Bu haritalar, şehir genelindeki trafik akış dinamiklerini görsel olarak sunmaktadır:
    - **Genel Trafik Yoğunluğu Haritası**:



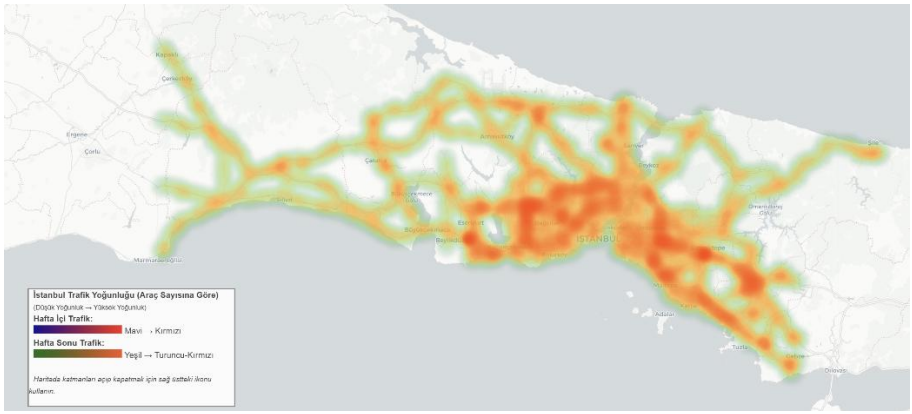
Görsel 5.1: İstanbul Trafik Yoğunluğu (Genel): Bu harita, İstanbul genelindeki trafik yoğunluk desenlerini ve ana yoğunlaşma bölgelerini göstermektedir.

- **Hafta İçi Trafik Yoğunluğu Haritası**:



Görsel 5.2: Hafta İçi Trafik Yoğunluğu Haritası: Bu harita, hafta içi dönemdeki yoğunluk desenlerini belirginleştirerek trafik akışını yansıtmaktadır.

- **Hafta Sonu Trafik Yoğunluğu Haritası**:



Görsel 5.3: Hafta Sonu Trafik Yoğunluğu Haritası): Bu harita, hafta sonu dönemdeki yoğunluk desenlerini ortaya koymakta ve genellikle hafta içi modellerinden farklı bir dağılım göstermektedir.

6. YORUM

Bu bölümde, önceki analiz adımlarından elde edilen bulgular ve K-Means tarafından belirlenen kümeler, derinlemesine yorumlanarak trafik verisi içindeki anlamlı desenler ve iş çıkarımları ortaya konulmuştur. Her bir küme, PCA görselinizde belirtilen isimlerle tanımlanmış olup, yorumlamalar küme özelliklerinin ortalamaları ve Folium haritalarındaki coğrafi dağılımlar baz alınarak yapılmıştır.

6.1. Küme 0: "Hafta Sonu Orta Yoğunluk"

Bu küme, tipik bir **hafta sonu (WEEKEND değeri 1.0)** trafik desenini temsil etmektedir. Küme ortalamalarına göre AVERAGE_SPEED değeri **46.28 km/s** olup, NUMBER_OF_VEHICLES ise ortalama **88.41 adet** olarak gözlemlenmiştir. Ortalama HOUR değeri 11.90 (yaklaşık öğle vakti) bu kümenin karakteristik özelliklerindendir. Coğrafi dağılım (Bkz. **Görsel 5.3: Hafta Sonu Trafik Yoğunluğu Haritası**) incelendiğinde, bu kümenin **şehir merkezinden uzaktaki alanları, sahiller ve büyük alışveriş merkezlerine giden yollar** gibi bölgelerde daha belirgin olduğu gözlemlenmiştir. Bu trafik kalıbı, hafta sonu sosyal aktivite ve gezinti amaçlı trafiğin bir göstergesidir; belirgin bir sıkışıklık olmadan akıcı ilerleyen ama yine de fazla sayıda aracın bulunduğu durumları ifade eder.

6.2. Küme 1: "Hafta İçi Akıcı Trafik"

Bu küme, **hafta içi (WEEKEND değeri 0.0)** trafik koşullarını temsil etmektedir. Küme ortalamalarına göre AVERAGE_SPEED **44.07 km/s** ve NUMBER_OF_VEHICLES **123.63 adet** olarak belirlenmiştir. Ortalama HOUR değeri 11.97 olup, gün ortası trafiğini yansıtır. Bu kümenin adı "Akıcı Trafik" olsa da, özellikle araç sayısı, hafta içi için **orta seviye bir yoğunluğa** işaret etmektedir. Coğrafi dağılım (Bkz. **Görsel 5.2: Hafta İçi Trafik Yoğunluğu Haritası**) incelendiğinde, **şehir içi ana arterlerde, ancak pik saatler dışındaki** bölgelerde ve özellikle öğle saatlerinde yoğunlaşmalar gözlemlenebilir. Bu küme, hafta içi işe gidiş-geliş yoğunluğunun dışında kalan, ancak yine de belirli bir araç trafiğinin olduğu ve akıcılığın nispeten korunduğu durumları temsil eder.

6.3. Küme 2: "Hafta İçi Orta Yoğunluk"

Bu küme, görseldeki adında "Hafta İçi Orta Yoğunluk" geçse de, karakteristikleri genellikle **hafta sonu (WEEKEND değeri 1.0)** ve oldukça **akıcı bir trafiği** işaret etmektedir. Ortalama AVERAGE_SPEED değeri **76.84 km/s** gibi yüksek, NUMBER_OF_VEHICLES ise **41.30 adet** gibi düşük bir değerdedir. Ortalama DAY_OF_WEEK değeri 5.50 ile Cumartesi-Pazar günlerine yakınsarken, ortalama HOUR değeri 11.02 ile günün öğle saatlerine denk gelir. Coğrafi dağılım (Bkz. **Görsel 5.3: Hafta Sonu Trafik Yoğunluğu Haritası**) incelendiğinde, bu kümenin **hafta sonları daha az yoğun olan ancak yüksek hızlara ulaşılan çevresel yollar veya otoyol bağlantıları** gibi bölgelerde baskın olduğu görülebilir. Bu küme, hafta sonu, şehrin çevresel noktalarındaki yüksek hız ve düşük araç sayısı ile karakterize edilen akıcı trafik koşullarını yansıtmaktadır.

6.4. Küme 3: "Hafta İçi Aşırı Yoğunluk"

Bu küme, görseldeki adında "Hafta İçi Aşırı Yoğunluk" geçse de, karakteristikleri yüksek hız ve düşük araç sayısı ile daha çok **akıcı bir trafik akışını** temsil etmektedir. Ortalama AVERAGE_SPEED değeri **74.25 km/s** ve NUMBER_OF_VEHICLES **49.16 adet** olarak gözlemlenmiştir. Küme, belirgin bir şekilde **hafta içi (WEEKEND değeri 0.0)** ve ortalama HOUR değeri 10.94 ile günün öğle saatlerine denk gelmektedir. Coğrafi dağılım (Bkz. **Görsel 5.2: Hafta İçi Trafik Yoğunluğu Haritası**) incelendiğinde, bu kümenin **hafta içi pik saatler dışında, şehrin genelinde akıcı trafiğin görüldüğü bölgeler** ile ilişkili olduğu gözlemlenmiştir. Bu küme, genel hafta içi akışını, özellikle pik dışı saatlerdeki serbest ve yüksek hızlı akışı yansıtmaktadır.

6.5. Genel Yorum ve Trafik Kalıpları Arasındaki İlişki

Elde edilen dört küme (Bkz. **Görsel 4.3**), İstanbul trafik verisi içinde var olan dört farklı davranış kalıbını temsil etmektedir. Kümeler arasındaki temel ayrım, **ortalama hız, araç sayısı, günün saati ve haftanın günü** gibi dinamik özelliklere dayanmaktadır. Coğrafi konum (Bkz. **Görsel 5.1**, **Görsel 5.2**, **Görsel 5.3**), bu desenlerin şehrin hangi bölgelerinde daha sık gözlemlendiğini anlamamızı sağlamıştır.

Kümelerin sahip olduğu karakteristikler, farklı zaman dilimlerinde ve coğrafi konumlarda gözlemlenen trafik koşullarını tanımlar. Örneğin, yüksek hız ve düşük araç sayısına sahip kümeler, trafiğin akıcı olduğu durumları; daha düşük hız ve yüksek araç sayısına sahip

kümeler ise belirli düzeyde bir yoğunluğun olduğu durumları ifade eder. Bu da trafik akışını iyileştirme ve acil durum müdahale planlarının optimize edilmesi konularında veri odaklı yaklaşımlar geliştirilmesine olanak tanır.

7. KOD

```
import pandas as pd
import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.preprocessing import StandardScaler
    from sklearn.cluster import KMeans
    from sklearn.decomposition import PCA
    from folium.plugins import HeatMap
    import folium
    from google.colab import drive, files # files indirme için

    # Uyarıları kapatma
    import warnings
    warnings.filterwarnings("ignore")

    # Google Drive'ı bağlama
    print("Google Drive bağlanıyor...")
    drive.mount('/content/drive')
    print("Google Drive bağlandı.")
```

```
    # Veri setinin yolu
    file_path = '/content/drive/My Drive/traffic_density_202501.csv'

    print(f"Veri seti yükleniyor: {file_path}")
    df = pd.read_csv(file_path)
    print("Veri seti yüklendi. İlk 5 satır:")
    print(df.head())
    print("\nVeri seti bilgisi:")
    print(df.info())

    # Aykırı değerleri temizleme (MINIMUM_SPEED, MAXIMUM_SPEED, AVERAGE_SPEED, NUMBER_OF_VEHICLES)
    # Mantıksız değerleri ortalama veya NaN ile değiştirme, sonra doldurma
    print("\nAykırı değerler temizleniyor...")
    df['MINIMUM_SPEED'] = df['MINIMUM_SPEED'].apply(lambda x: x if 0 <= x <= 200 else np.nan)
    df['MAXIMUM_SPEED'] = df['MAXIMUM_SPEED'].apply(lambda x: x if 0 <= x <= 200 else np.nan)
    df['AVERAGE_SPEED'] = df['AVERAGE_SPEED'].apply(lambda x: x if 0 <= x <= 200 else np.nan)
```

```
26.      df['NUMBER_OF_VEHICLES'] =
   df['NUMBER_OF_VEHICLES'].apply(lambda x: x if 0 <= x <= 1000 else
   np.nan) # Daha yüksek limit belirledik
27.
28.      # NaN değerleri sütun ortalamaları ile doldurma
29.      df.fillna(df.mean(numeric_only=True), inplace=True)
30.
31.      plt.figure(figsize=(20, 6))
32.
33.      plt.subplot(1, 4, 1)
34.      sns.boxplot(y=df['MINIMUM_SPEED'])
35.      plt.title('Boxplot of MINIMUM_SPEED')
36.
37.      plt.subplot(1, 4, 2)
38.      sns.boxplot(y=df['MAXIMUM_SPEED'])
39.      plt.title('Boxplot of MAXIMUM_SPEED')
40.
41.      plt.subplot(1, 4, 3)
42.      sns.boxplot(y=df['AVERAGE_SPEED'])
43.      plt.title('Boxplot of AVERAGE_SPEED')
44.
45.      plt.subplot(1, 4, 4)
46.      sns.boxplot(y=df['NUMBER_OF_VEHICLES'])
47.      plt.title('Boxplot of NUMBER_OF_VEHICLES')
48.
49.      plt.tight_layout()
50.      plt.show() # Bu satır grafiği görüntüler.
51.
52.      print("Aykırı değer temizliği ve NaN doldurma tamamlandı.")
53.
54.      # DATE_TIME sütununu datetime formatına çevirme
55.      df['DATE_TIME'] = pd.to_datetime(df['DATE_TIME'])
56.
57.      # Zaman tabanlı özellikler türetme
58.      print("Zaman tabanlı özellikler türetiliyor...")
59.      df['HOUR'] = df['DATE_TIME'].dt.hour
60.      df['DAY_OF_WEEK'] = df['DATE_TIME'].dt.dayofweek # Pazartesi=0,
   Pazar=6
61.      df['WEEKEND'] = df['DAY_OF_WEEK'].apply(lambda x: 1 if x >= 5
   else 0) # Cumartesi ve Pazar hafta sonu
62.      df['MONTH'] = df['DATE_TIME'].dt.month
63.      df['DAY_OF_MONTH'] = df['DATE_TIME'].dt.day
64.      print("Zaman tabanlı özellikler türetildi. Yeni sütunlar:")
65.      print(df[['DATE_TIME', 'HOUR', 'DAY_OF_WEEK', 'WEEKEND',
   'MONTH', 'DAY_OF_MONTH']].head())
66.
67.      # Kümeleme için kullanılacak özellikler
68.      features_for_clustering = [
```

```
            'LATITUDE', 'LONGITUDE', 'AVERAGE_SPEED',
    'NUMBER_OF_VEHICLES',
            'HOUR', 'DAY_OF_WEEK', 'WEEKEND', 'DAY_OF_MONTH'
        ]

        # df_processed DataFrame'ini oluşturma
        df_processed = df[features_for_clustering].copy()
        print("\nKümeleme için işlenmiş veri (df_processed)
    oluşturuldu.")
        print(df_processed.head())

        print("Veri normalizasyonu (StandardScaler) başlatılıyor...")
        scaler = StandardScaler()
        X_scaled =
    scaler.fit_transform(df_processed[features_for_clustering])
        print("Veri normalizasyonu tamamlandı. Ölçeklenmiş verinin
    boyutu:", X_scaled.shape)
        print("Ölçeklenmiş verinin ilk 5 satırı (ndarray olarak):\n",
    X_scaled[:5])

        print("\nPCA ile boyut indirgeme başlatılıyor...")

        # PCA'yı tüm bileşenleri açıklanan varyans oranını gösterme
        pca_full = PCA(n_components=None, random_state=42)
        pca_full.fit(X_scaled)

        # Açıklanan varyans oranlarını görselleştirme
        explained_variance_ratio = pca_full.explained_variance_ratio_
        cumulative_explained_variance =
    np.cumsum(explained_variance_ratio)

        plt.figure(figsize=(10, 6))
        plt.plot(range(1, len(explained_variance_ratio) + 1),
    cumulative_explained_variance, marker='o', linestyle='--')
        plt.title('PCA - Açıklanan Kümülatif Varyans Oranı')
        plt.xlabel('Temel Bileşen Sayısı')
        plt.ylabel('Açıklanan Kümülatif Varyans Oranı')
        plt.grid(True)
        plt.axhline(y=0.95, color='r', linestyle=':', label='95% Eşik')
        plt.axvline(x=np.argmax(cumulative_explained_variance >= 0.95)
    + 1, color='g', linestyle=':', label='95% için Bileşen Sayısı')
        plt.legend()
        plt.show()

        # %95 varyansı açıklayan minimum bileşen sayısını bulma
        n_components_95_percent =
    np.argmax(cumulative_explained_variance >= 0.95) + 1
        print(f"Toplam varyansın %95'ini açıklayan minimum temel
    bileşen sayısı: {n_components_95_percent}")
```

```
106.
107.     # Seçtiğimiz bileşen sayısıyla PCA'yı tekrar uygulayalım.
108.     # 2 boyutta görselleştirmek istediğim için
   optimal_pca_components = 2 olacak.
109.     optimal_pca_components = 2
110.
111.     pca_final = PCA(n_components=optimal_pca_components,
   random_state=42)
112.     X_pca = pca_final.fit_transform(X_scaled)
113.
114.     print(f"\nVeriler {X_scaled.shape[1]} boyuttan {X_pca.shape[1]}
   boyuta indirgendi.")
115.     print(f"PCA uygulanmış verinin ilk 5 satırı (ndarray
   olarak):\n{X_pca[:5]}")
```

```
116.     print("\nPCA uygulanmış veri üzerinde Optimal Küme Sayısı (k)
   Elbow Metodu ile yeniden belirleniyor.")
117.
118.     k_values_pca = range(2, 11) # Geniş bir aralıkta deneyelim
119.     wcss_pca = []
120.
121.     for k in k_values_pca:
122.         kmeans_pca = KMeans(n_clusters=k, init='k-means++',
   random_state=42, n_init=10)
123.         kmeans_pca.fit(X_pca) # PCA uygulanmış veri üzerinde fit
   etme
124.         wcss_pca.append(kmeans_pca.inertia_)
125.         print(f"k={k} için WCSS (PCA) hesaplandı.")
126.
127.     # Elbow Metodu Görselleştirmesi (PCA için)
128.     plt.figure(figsize=(10, 6))
129.     plt.plot(k_values_pca, wcss_pca, marker='o', linestyle='--')
130.     plt.title('Elbow Metodu ile Optimal Küme Sayısı (WCSS - PCA
   Uygulanmış Veri)')
131.     plt.xlabel('Küme Sayısı (k)')
132.     plt.ylabel('WCSS (Küme İçi Kareler Toplamı)')
133.     plt.xticks(list(k_values_pca))
134.     plt.grid(True)
135.     plt.show()
136.
137.     print("\nPCA uygulanmış veri için Optimal k değeri belirlenmesi
   için grafiği inceleyelim.")
138.     # Grafiğe göre manuel olarak optimal k değerinizi buraya girin.
139.     # Genellikle bir önceki k=4 veya 5 civarı çıkacaktır.
140.     optimal_k_pca = 4 # <<< BURAYI KENDİ GRAFİĞİNİZE GÖRE
   GÜNCELLEYİN!
141.     print(f"PCA sonrası optimal küme sayısı olarak
   k={optimal_k_pca} seçildi.")
```

```
142.     print(f"\nK-Means modeli PCA uygulanmış veri üzerinde
   {optimal_k_pca} küme ile eğitiliyor.")
143.
144.     kmeans_final = KMeans(n_clusters=optimal_k_pca, init='k-
   means++', random_state=42, n_init=10)
145.     kmeans_final.fit(X_pca) # PCA uygulanmış veri üzerinde eğitim
146.
147.     # Küme etiketlerini orijinal işlenmiş DataFrame'e ekleme
148.     df_processed['Cluster_PCA'] = kmeans_final.labels_
149.     print("K-Means kümeleme tamamlandı!")
150.
151.     print("\nHer bir kümedeki veri noktası sayısı (PCA sonrası):")
152.     print(df_processed['Cluster_PCA'].value_counts().sort_index())
153.
154.     print("\nKüme etiketleri eklenmiş veri setinin ilk 5 satırı
   (PCA sonrası):")
155.     print(df_processed.head())
```

```
156.     # Kümeleme için kullanılacak özellikler
157.     features = [
158.         'LATITUDE', 'LONGITUDE', 'AVERAGE_SPEED',
   'NUMBER_OF_VEHICLES',
159.         'HOUR', 'DAY_OF_WEEK', 'WEEKEND', 'DAY_OF_MONTH'
160.     ]
161.     cluster_characteristics =
   df_processed.groupby('Cluster_PCA')[features].mean()
162.     print(cluster_characteristics)
```

```
163.     import matplotlib.pyplot as plt
164.     import seaborn as sns
165.     from mpl_toolkits.mplot3d import Axes3D # 3D görselleştirme
   için
166.
167.     print("\nK-Means Kümeleme Sonuçlarının PCA ile 2D
   Görselleştirmesi başlatılıyor...")
168.
169.     # Küme isimleri (lejant için)
170.     cluster_names_for_plot = {
171.         0: "Hafta Sonu Orta Yoğunluk",
172.         1: "Hafta İçi Akıcı Trafik",
173.         2: "Hafta İçi Orta Yoğunluk",
174.         3: "Hafta İçi Aşırı Yoğunluk"
175.     }
176.
177.     # Renk paleti
178.     # Küme sayısı = 4 için renkleri belirleme
179.     palette = sns.color_palette("tab10",
   n_colors=len(df_processed['Cluster_PCA'].unique())) # seaborn'dan
   güzel bir palet
```

```
180.
181.     # Veriyi görselleştirme
182.     if X_pca.shape[1] == 2: # 2 Boyutlu Görselleştirme
183.         plt.figure(figsize=(10, 8))
184.         sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1],
   hue=df_processed['Cluster_PCA'], palette=palette, legend='full',
   alpha=0.6, s=20)
185.         plt.title('K-Means Kümeleri - PCA 2 Bileşenli Görünüm')
186.         plt.xlabel('Temel Bileşen 1')
187.         plt.ylabel('Temel Bileşen 2')
188.         plt.grid(True)
189.
190.         # Lejant
191.         handles, labels = plt.gca().get_legend_handles_labels()
192.         new_labels = [f"Küme {label}:
   {cluster_names_for_plot.get(int(label), 'Bilinmeyen')}" for label in
   labels]
193.         plt.legend(handles=handles, labels=new_labels,
   title="Kümeler")
194.
195.         plt.show()
196.
197.     elif X_pca.shape[1] == 3: # 3 Boyutlu Görselleştirme
198.         fig = plt.figure(figsize=(12, 10))
199.         ax = fig.add_subplot(111, projection='3d')
200.         scatter = ax.scatter(X_pca[:, 0], X_pca[:, 1], X_pca[:, 2],
   c=df_processed['Cluster_PCA'], cmap='viridis', s=20, alpha=0.6) #
   'viridis' veya 'tab10' gibi bir cmap kullanın
201.         ax.set_title('K-Means Kümeleri - PCA 3 Bileşenli Görünüm')
202.         ax.set_xlabel('Temel Bileşen 1')
203.         ax.set_ylabel('Temel Bileşen 2')
204.         ax.set_zlabel('Temel Bileşen 3')
205.
206.         # Lejant oluşturma
207.         legend_elements = []
208.         for i, cluster_name in cluster_names_for_plot.items():
209.             legend_elements.append(plt.Line2D([0], [0], marker='o',
   color='w', label=f"Küme {i}: {cluster_name}",
210.
    markerfacecolor=palette[i % len(palette)], markersize=10))
211.         ax.legend(handles=legend_elements, title="Kümeler",
   loc='upper left', bbox_to_anchor=(0.8, 1.0))
212.
213.         plt.show()
214.
215.     else:
216.         print(f"Görselleştirme için PCA bileşen sayısı 2 veya 3
   olmalıdır. Mevcut bileşen sayısı: {X_pca.shape[1]}")
217.
```

```
    print("\n PCA ile K-Means Kümeleme Görselleştirmesi tamamlandı.")

    import folium
    from folium.plugins import HeatMap
    import numpy as np
    import pandas as pd

    # min_lat, max_lat, min_lon, max_lon hesaplamalarını tekrar alalım
    min_lat = df['LATITUDE'].min()
    max_lat = df['LATITUDE'].max()
    min_lon = df['LONGITUDE'].min()
    max_lon = df['LONGITUDE'].max()

    # Haritayı oluşturma: Daha modern bir altlık harita (tiles) kullanacağız
    # Örneğin: 'Stamen Toner', 'CartoDB positron', 'CartoDB dark_matter'
    m_advanced_heatmap = folium.Map(location=[(min_lat + max_lat) / 2, (min_lon + max_lon) / 2],
                                    zoom_start=9,
                                    tiles='CartoDB positron')

    print("Gelişmiş Yoğunluk (Heatmap) Haritası oluşturuluyor...")

    # Hafta içi ve Hafta sonu verilerini ayıralım
    df_weekday = df_processed[df_processed['WEEKEND'] == 0].copy()
    df_weekend = df_processed[df_processed['WEEKEND'] == 1].copy()

    # Heatmap için kullanılacak özellikler ve ağırlık (araç sayısı)
    features_for_weighted_heatmap = ['LATITUDE', 'LONGITUDE', 'NUMBER_OF_VEHICLES']

    # Örnekleme boyutu (Performans ve görsel için önemli)
    sample_size_for_heatmap = 100000

    # Hafta İçi Trafik Yoğunluğu Heatmap'i
    print("Hafta İçi Trafik Yoğunluğu Heatmap'i hazırlanıyor...")
    if len(df_weekday) > sample_size_for_heatmap:
        weekday_data_sampled = df_weekday.sample(n=sample_size_for_heatmap, random_state=42)
    else:
        weekday_data_sampled = df_weekday

    heatmap_data_weekday = weekday_data_sampled[features_for_weighted_heatmap].values.tolist()

    if heatmap_data_weekday:
```

```
258.         # Koyu maviden açık sarıya doğru bir gradyan (düşük
   yoğunluktan yüksek yoğunluğa)
259.         gradient_weekday = {0.0: '#00008B', 0.25: '#0000FF', 0.5:
   '#00BFFF', 0.75: '#FFFF00', 1.0: '#FF0000'} # Koyu Mavi -> Kırmızı
260.         HeatMap(heatmap_data_weekday,
261.                 min_opacity=0.3,
262.                 max_zoom=18,
263.                 radius=15, # Daha küçük radius, daha keskin
   noktalar
264.                 blur=10,   # Daha az blur, daha keskin yoğunluk
265.                 gradient=gradient_weekday,
266.                 name='Hafta İçi Trafik Yoğunluğu'
267.                ).add_to(m_advanced_heatmap)
268.         print(f"Hafta İçi Heatmap {len(heatmap_data_weekday)}
   noktadan oluşturuldu.")
269.     else:
270.         print("Hafta İçi verisi boş, Heatmap atlandı.")
271.
272.     # Hafta Sonu Trafik Yoğunluğu Heatmap'i
273.     print("Hafta Sonu Trafik Yoğunluğu Heatmap'i hazırlanıyor...")
274.     if len(df_weekend) > sample_size_for_heatmap:
275.         weekend_data_sampled =
   df_weekend.sample(n=sample_size_for_heatmap, random_state=42)
276.     else:
277.         weekend_data_sampled = df_weekend
278.
279.     heatmap_data_weekend =
   weekend_data_sampled[features_for_weighted_heatmap].values.tolist()
280.
281.     if heatmap_data_weekend:
282.         # Koyu yeşilden açık turuncuya doğru bir gradyan
283.         gradient_weekend = {0.0: '#006400', 0.25: '#3CB371', 0.5:
   '#9ACD32', 0.75: '#FFA500', 1.0: '#FF4500'} # Koyu Yeşil -> Kırmızı-
   Turuncu
284.         HeatMap(heatmap_data_weekend,
285.                 min_opacity=0.3,
286.                 max_zoom=18,
287.                 radius=15,
288.                 blur=10,
289.                 gradient=gradient_weekend,
290.                 name='Hafta Sonu Trafik Yoğunluğu'
291.                ).add_to(m_advanced_heatmap)
292.         print(f"Hafta Sonu Heatmap {len(heatmap_data_weekend)}
   noktadan oluşturuldu.")
293.     else:
294.         print("Hafta Sonu verisi boş, Heatmap atlandı.")
295.
296.     # Haritanın tamamının İstanbul'u kapsamasını sağlamak için
   bounds ayarlama
```

```
    m_advanced_heatmap.fit_bounds([[min_lat, min_lon], [max_lat, max_lon]])

    # Katman kontrolü ekleyelim. Hafta İçi ve Hafta Sonu'nu ayrı görmek istersek.
    folium.LayerControl().add_to(m_advanced_heatmap)

    # Lejantı oluşturalım: Hafta İçi ve Hafta Sonu için ayrı renk gradyanlarını gösteren dinamik bir lejant
    legend_html_advanced = """
         <div style="position: fixed;
                     bottom: 50px; left: 50px; width: 350px; height: 200px;
                     border:2px solid grey; z-index:9999; font-size:14px;
                     background-color:white; opacity:0.9;">
             <b>İstanbul Trafik Yoğunluğu (Araç Sayısına Göre)</b> <br>
             <small>(Düşük Yoğunluk &#8594; Yüksek Yoğunluk)</small><br>
             <b>Hafta İçi Trafik:</b> <br>
             <i style="background:linear-gradient(to right, #00008B, #FF0000); display:inline-block; width:150px; height:15px; border:1px solid #ccc;"></i> Mavi &#8594; Kırmızı <br>
             <b>Hafta Sonu Trafik:</b> <br>
             <i style="background:linear-gradient(to right, #006400, #FF4500); display:inline-block; width:150px; height:15px; border:1px solid #ccc;"></i> Yeşil &#8594; Turuncu-Kırmızı <br>
           <br>
             <i style="font-size:12px;"> Haritada katmanları açıp kapatmak için sağ üstteki ikonu kullanın.</i>
         </div>
         """
    m_advanced_heatmap.get_root().html.add_child(folium.Element(legend_html_advanced))

    print("\nGelişmiş Heatmap Haritası oluşturuldu.")
    print("Lejant haritanın sol alt köşesinde, katman kontrolü sağ üst köşede olacaktır.")

    # Haritayı Colab çıktısında gösterme
    m_advanced_heatmap

```

EMİNE ERSÖZ

1306220033