



Teknoloji Fakültesi

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

CROWDPREDICTOR

BİTİRME PROJESİ FİNAL RAPORU

Bilgisayar Mühendisliği Bölümü

DANIŞMAN

Doç. Dr. Öğr. Üyesi BUKET DOĞAN

İSTANBUL, 2025

MARMARA ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Marmara Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Öğrencileri Asuman Baş,

Elif Gökçe Ünver ve Mehmet Ali Onur Yavuz tarafından “YAPAY ZEKA DESTEKLİ

TAHMİN ALGORİTMASI İLE KULLANICI ODAKLI TRAFİK YOĞUNLUĞU

ÖNGÖRÜSÜ SAĞLAYAN WEB TABANLI GÖRSELLEŞTİRME SİSTEMİ

TASARIMI” başlıklı proje çalışması, 19/06/2025 tarihinde savunulmuş ve jüri üyeleri

tarafından başarılı bulunmuştur.

Jüri Üyeleri

Doç. Dr. Buket DOĞAN (Danışman)
Marmara Üniversitesi (İMZA).....

Prof. Dr. Ali Buldu (Üye)
Marmara Üniversitesi (İMZA).....

Arş. Gör. Asaf Karataş (Üye)
Marmara Üniversitesi (İMZA).....

ÖNSÖZ

Bu bitirme projesi kapsamında, yapay zekâ destekli ve harita tabanlı bir trafik tahmin sistemi geliştirerek bu alandaki çözüm arayışına küçük de olsa bir katkı sağlamayı amaçlanmıştır. Gelişen şehircilik ve artan araç yoğunluğu, günümüzde ulaşım problemlerini daha karmaşık hâle getirmiştir. Bu bağlamda, trafik yoğunluğunu önceden tahmin edebilen sistemlerin geliştirilmesi hem bireysel kullanıcılar hem de kent planlamacıları için büyük önem taşımaktadır.

Proje süresince karşılaştığımız her teknik ve akademik zorlukta, bilgi ve rehberliklerini bizimle paylaşan, her zaman yanımızda olduğunu hissettiren çok değerli hocam **Sayın Doç. Dr. Buket DOĞAN**'a en içten teşekkürlerimizi sunarız.

Bu çalışmanın, trafik yönetimi ve veri tabanlı karar sistemleri alanında yeni çalışmalara ilham vermesini temenni ederiz.

Haziran 2025 , Asuman BAŞ, Elif Gökçe ÜNVER, Mehmet Ali Onur YAVUZ

İçindekiler Tablosu

ÖZET	i
ABSTRACT	ii
1. GİRİŞ	1
1.1 Proje Çalışmasının Amacı ve Önemi	1
2. BULGULAR VE TARTIŞMA	2
3. LİTERATÜR TARAMASI	3
4. VERİ TOPLAMA	3
5. VERİ HAZIRLAMA VE MODEL EĞİTİMİ	4
5.1 Veri Birleştirme ve Ön İşleme.....	5
5.2 Veri Dengesizliği Dengeleme	5
5.3 Veri Dengesizliği ve Dengeleme.....	6
5.4 Özellik Setinin Genişletilmesi	7
5.5 Model Eğitimi ve Kayıt.....	8
5.6 Metrik Karşılaştırması ve Grafik Oluşturma.....	8
6. MODEL ENTEGRASYONU	10
6.1 Flask Tabanlı API Yapısı.....	10
6.2 Flask Tercihinin Gerekçeleri	11
6.3 Uçtan Uca Entegrasyon Akışı.....	11
6.4 Geleceğe Yönelik Genişletme Önerileri.....	12

7.	<i>ARAYÜZ GELİŞTİRME</i>	12
7.1	<i>Kullanıcı Giriş ve Kayıt Arayüzü</i>	13
7.2	<i>Veri Tabanı Altyapısı</i>	15
7.3	<i>Harita Entegrasyonu ve Kullanılan Teknolojiler</i>	16
7.4	<i>Harita Görüntüleme Ekranına Genel Bakış</i>	17
7.5	<i>Karşılaşılan Sorunlar</i>	19
7.6	<i>Sonuç ve Değerlendirme</i>	20
8.	<i>GENEL DEĞERLENDİRME</i>	23

ÖZET

YAPAY ZEKA DESTEKLİ TAHMİN ALGORİTMASI İLE KULLANICI ODAKLI TRAFİK YOĞUNLUĞU ÖNGÖRÜSÜ SAĞLAYAN WEB TABANLI GÖRSELLEŞTİRME SİSTEMİ TASARIMI

Bu projede, kullanıcıların belirli bir tarih ve saat için trafik yoğunluğu tahmini alabilecekleri, yapay zekâ destekli bir web tabanlı sistemin geliştirilmesi amaçlanmıştır. Trafik yoğunluğu, özellikle büyük şehirlerde bireylerin günlük yaşamlarını doğrudan etkileyen önemli bir sorundur. Mevcut trafik uygulamaları yalnızca anlık bilgiler sunarken, bu projede geleceğe yönelik tahminler yapılması hedeflenmiştir.

Sistem, React.js ile geliştirilmiş kullanıcı dostu bir arayüz, Flask ile yazılmış bir backend sunucu ve Python tabanlı bir makine öğrenmesi modeli ile bütünleşerek çalışacak şekilde tasarlanmıştır. Kullanıcılar giriş yaptıktan sonra başlangıç noktası, varış noktası ve yolculuk zamanı gibi bilgileri sisteme girerek trafik tahmini alabilmektedir. Tahmin sonuçları, kullanıcıya yoğunluk durumuna göre uygun görsellerle sunulmaktadır.

Proje kapsamında kullanıcıların ulaşım planlamasını kolaylaştıran, akıllı ve kişiselleştirilmiş bir trafik tahmin sistemi ortaya konması hedeflenmektedir.

Mart, 2025

Öğrenciler

Asuman BAŞ	170421019
Elif Gökçe ÜNVER	170421011
Mehmet Ali Onur YAVUZ	170421038

ABSTRACT

DESIGN OF A WEB-BASED VISUALIZATION SYSTEM FOR USER-CENTERED TRAFFIC CONGESTION PREDICTION USING AN AI-BASED ESTIMATION ALGORITHM

In this project, the aim is to develop an AI-powered web-based system where users can obtain traffic congestion forecasts for a specific date and time. Traffic congestion is a significant issue that directly affects individuals' daily lives, especially in major cities. While existing traffic applications only provide real-time information, this project aims to make future predictions.

The system is designed to operate through the integration of a user-friendly interface developed with React.js, a backend server written in Flask, and a Python-based machine learning model. After logging in, users can input information such as the starting point, destination, and travel time into the system to receive traffic forecasts. The prediction results are presented to the user with appropriate visuals based on the level of congestion.

The project aims to introduce an intelligent and personalized traffic prediction system that facilitates transportation planning for users.

March, 2025

Students

Asuman BAŞ	170421019
Elif Gökçe ÜNVER	170421011
Mehmet Ali Onur YAVUZ	170421038

ŞEKİL VE TABLO LİSTESİ

Şekil 2.1: Rota bilgileri.....	3
Şekil 2.2 :Harita üzerinde trafik yoğunluğu gösterimi	3
Şekil 4.1: Veri Seti Özellikleri ve Değişken Tanımları.....	5
Şekil 5.3.1:Trafik Yoğunluğu Tahmin Performansı.....	7
Şekil 5.3.2:Smote Tekniği Trafik Yoğunluğu Tahmin Performansı	7
Şekil 5.4.1:Özellik ve etiket belirleme kod örneği.....	8
Şekil 5.4.2:Genişletilmiş özellik seti kod örneği	8
Şekil 5.4.3:Genişletilmiş Özelliklerle Tahmin Performansı.....	8
Şekil 5.6.1:Model Karşılaştırması:Macro F1-Score.....	10
Şekil 5.6.2:Model Karşılaştırması:Macro Recall	10
Şekil 7.1.1:Kullanıcı Giriş Yap Ekranı.....	14
Şekil 7.1.2:Kullanıcı Kayıt Ol Ekranı	15
Şekil 7.2.1:MySQL Tablo Kayıtları	17
Şekil 7.6.1:Karışıklık Matrisi.....	29
Şekil 7.6.2: Ortalama Hız ve Araç Sayısı.....	30
Şekil 7.6.3: Genel Model Performansı.....	30
Şekil 7.6.4: İstanbul Trafik Haritası.....	31
Şekil 7.6.5: Günlere Göre Trafik Dağılımı.....	31

1.GİRİŞ

Modern şehirlerde artan nüfus ve araç yoğunluğu, ulaşım sistemleri üzerinde ciddi baskılar oluşturmakta ve günlük yaşamda zaman kaybı, yakıt israfı ve çevresel etkiler gibi pek çok sorunu beraberinde getirmektedir.[1] Trafik sıkışıklığı, yalnızca sürücüler için değil, toplu taşıma kullanıcıları ve yayalar için de büyük bir sorun teşkil etmektedir. Özellikle işe gidiş ve dönüş saatlerinde, büyük şehirlerde trafik yoğunluğu tahammül edilemeyecek seviyelere ulaşmakta; bu da bireylerin günlük planlamalarını yapmalarını zorlaştırmaktadır.

Mevcut trafik uygulamaları, genellikle anlık veriler sunmakta ve sadece mevcut trafik durumuna dair bilgi vermektedir. Ancak birçok kullanıcı, gelecekteki bir gün ve saate ait trafik tahmini yaparak planlama yapmak istemektedir. Örneğin; işe, sınava ya da havaalanına zamanında ulaşmak için gelecekteki trafik yoğunluğunun bilinmesi büyük avantaj sağlamaktadır. Bu ihtiyaçtan yola çıkarak geliştirilen bu proje, kullanıcının belirli bir gün ve saat için trafik tahmini alabileceği, geçmiş sorgularını görebileceği ve favori rotalarını saklayabileceği bir sistem sunmayı amaçlamaktadır.[2]

Projede, kullanıcıların arayüz üzerinden giriş yaparak gidecekleri güzergâhı ve zamanı belirtmeleri sağlanmakta, bu bilgiler bir makine öğrenmesi modeline gönderilmekte ve trafik yoğunluğu tahmini görsel olarak kullanıcıya sunulmaktadır. Başlangıç aşamasında rastgele tahminler ile çalışan sistem, ilerleyen dönemlerde gerçek veriyle eğitilmiş yapay zeka modelleri ile daha isabetli öngörüler sağlayacaktır.

1.1.Proje Çalışmasının Amacı ve Önemi

Bu projenin temel amacı, kullanıcıların gelecekteki belirli bir tarih ve saat için trafik yoğunluğu tahmini alabilecekleri bir sistem geliştirmektir. Mevcut trafik uygulamaları genellikle sadece anlık trafik verilerini sunmakta ve kullanıcılara ileriye dönük planlama yapma olanağı vermemektedir.[3] Ancak günlük hayatın birçok alanında, önceden yapılacak trafik öngörülerini büyük önem taşımaktadır. Özellikle işe gidiş, önemli randevular, sınavlara veya havaalanı gibi zaman duyarlı yerlere ulaşım için gelecekteki trafik durumu hayati önem arz etmektedir.

Bu ihtiyaca yönelik olarak geliştirilen sistem, kullanıcıların giriş yaptıktan sonra başlangıç ve varış noktaları ile tarih ve saat seçimi yapmalarına imkân tanımaktadır. Bu bilgiler yapay zekâ tabanlı bir tahmin modeli tarafından analiz edilerek, kullanıcının seyahat edeceği zamana ait trafik yoğunluğu seviyesi tahmin edilmekte ve kullanıcıya görsel olarak sunulmaktadır. Görsel sunumlar sayesinde kullanıcılar yoğunluk durumunu kolayca anlayabilmekte ve alternatif planlar yapabilmektedir.

Projenin uzun vadede sağladığı katkı, bireylerin zaman yönetimini iyileştirmek, trafik kaynaklı stres ve belirsizliği azaltmak ve şehir içi ulaşımı daha verimli hale getirmektir. Ayrıca, sistemin ilerleyen sürümlerinde favori rotalar, geçmiş sorgular ve gerçek zamanlı öğrenme kabiliyeti ile daha kişiselleştirilmiş ve akıllı bir ulaşım asistanı olarak geliştirilmesi hedeflenmektedir.

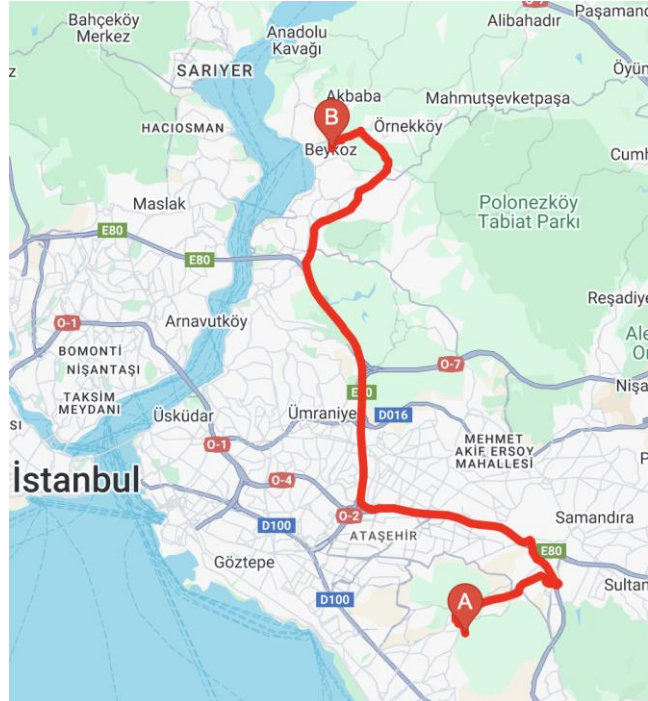
2.BULGULAR VE TARTIŞMA

Projenin bu aşamasına kadar olan süreçte, sistemin temel iskeleti oluşturulmuş ve kullanıcı etkileşimini sağlayacak olan arayüz tasarımı tamamlanmıştır. Kullanıcıların sisteme giriş yapabileceği bir ekran geliştirilmiş, ardından seyahat edecekleri güzergâhı ve zamanı girebilecekleri bir form ekranı hazırlanmıştır. Bu form aracılığıyla alınan veriler, Spring Boot ile geliştirilen backend servisine iletilmekte ve bu bilgiler doğrultusunda, test sürecinde kullanılmak üzere rastgele belirlenen trafik yoğunluğu tahmini frontend tarafına iletilmektedir. Kullanıcılara bu tahmin sonucu, yoğunluk seviyesine uygun görseller ile sunulmakta, böylece sistem hem işlevsel hem de görsel olarak anlaşılır bir yapı kazanmaktadır.

Frontend kısmı React.js kullanılarak oluşturulmuş, her ekran bileşen tabanlı yapıda geliştirilmiş ve sayfalar arasında router sistemiyle geçiş sağlanmıştır. Backend tarafında ise RESTful API mimarisi kullanılarak frontend ile veri alışverişi sağlanmıştır. Şu anki durumda trafik tahminleri, yapay zekâ modeli henüz entegre edilmediği için rastgele verilerle oluşturulmakta, ancak bu yapı gerçek modelle çalışacak şekilde tasarlanmıştır.

Bundan sonraki süreçte, Python ile eğitilen yapay zekâ modelinin backend yapısına entegre edilmesi ve kullanıcıdan alınan verilere göre gerçek tahminlerin yapılması hedeflenmektedir. Ayrıca kullanıcı kayıt/giriş sistemi geliştirilecek, favori rotalar ve geçmiş arama kayıtları gibi kullanıcıya özel veri yönetimi sağlanacaktır. Bununla birlikte, Google Maps API ile harita üzerinde rota ve trafik yoğunluğu görselleştirmesi yapılması planlanmaktadır. Yapılacak bu geliştirmeler sayesinde, sistem hem daha akıllı hale gelecek hem de kullanıcı dostu ve işlevsel bir ulaşım asistanına dönüşecektir.

Şekil 2.1: Rota bilgileri



Şekil 2.2 :Harita üzerinde trafik yoğunluğu gösterimi

3.LİTERATÜR TARAMASI

Son yıllarda kentleşmenin hızla artmasıyla birlikte trafik yoğunluğu tahmini, ulaşım sistemlerinin verimli yönetimi açısından kritik bir alan haline gelmiştir. Özellikle İstanbul gibi megakentlerde, nüfus ve araç sayısındaki artışın beraberinde getirdiği trafik problemleri, çeşitli veri odaklı yaklaşımlarla analiz edilmekte ve tahmin edilmeye çalışılmaktadır. Bu kapsamda literatürde farklı zaman ölçeklerinde (kısa, orta, uzun vadeli) trafik tahmini yapan modeller geliştirilmiştir.[4] Yıldız Teknik Üniversitesi kaynaklı bir çalışmada, İstanbul'daki trafik sensörlerinden alınan hız verileriyle, kullanıcıya harita üzerinde görsel analiz sağlayan ve kısa-orta-uzun vadeli tahminleme yapan bir sistem önerilmiştir. Bu sistemde regresyon tabanlı üç farklı model geliştirilmiş, %23.99 MAPE ile 1 haftaya kadar tahmin başarısı elde edilmiştir. Aynı sistem, Google Maps ve Yandex gibi servislerin eksik kaldığı özelleştirilmiş zaman aralıkları ve kullanıcı seçimli analiz senaryoları sunarak dikkat çekmiştir.[5] Diğer yandan, yapay sinir ağlarına dayalı modellerde doğruluk oranları çok daha yüksek seviyelere ulaşmıştır.[6] Örneğin, İstanbul Büyükşehir Belediyesi verilerini kullanan bir yüksek lisans tezinde MLP, RBF, LSTM ve GRU gibi farklı sinir ağı yapılarına dayalı tahmin modelleri geliştirilmiş ve univariate ile multivariate zaman serileri üzerinde denenmiştir.[7] Özellikle hibrit RBF-MLP yapısının çok değişkenli girişlerde daha kararlı sonuçlar verdiği gözlemlenmiştir. Derin öğrenme tabanlı bir başka çalışmada ise, LSTM mimarisi ile Mahmutbey kavşağında araç sayısı ve ortalama hız tahmini yapılmış, %0.9 R² değeriyle yüksek doğruluk elde edilmiştir.[8] Bu çalışmada ayrıca RF, SVM ve CNN gibi modellerle karşılaştırmalı analiz yapılmış ve LSTM'in tahmin gücü öne çıkmıştır.[9] Literatürde ayrıca trafik tahmininde hava durumu, sıcaklık, nem ve rüzgâr gibi çevresel faktörlerin etkisinin de önemli olduğu vurgulanmıştır.[10] Örneğin, rüzgar ve sıcaklığın hız üzerinde negatif etkileri olduğu; bulutlu, yağışlı hava ve mesai saatlerinin tahmin doğruluğunu etkilediği görülmüştür.[11] Daha genel bir perspektiften bakıldığında, GPS verilerine dayalı spatio-temporal analizlerin büyük veri ortamlarında uygulanabilirliği de gösterilmiş ve farklı zaman dilimleri için trafik akışlarının modellenebileceği belirtilmiştir. [12] Tüm bu çalışmalar ışığında, Crowd Predictor projesi, hem kullanıcıdan alınan rota ve saat bilgisiyle tahminleme yapması, hem de Google Maps API ile harita üzerinde renk kodlu yoğunluk gösterimi yapabilmesi bakımından literatürdeki boşlukları hedeflemekte ve kullanıcı deneyimini ön planda tutan bir yaklaşım sunmaktadır.

4.VERİ TOPLAMA

Projenin başlangıç aşamasında, trafik yoğunluğu tahmini için ihtiyaç duyulan verilerin nasıl toplanacağına ilişkin kapsamlı bir değerlendirme yapılmıştır. İlk olarak, veriyi kendi sistemimiz üzerinden gerçek zamanlı olarak toplamak hedeflenmiştir. Bu doğrultuda, belirli rotalar için trafik bilgilerini günde iki kez (örneğin sabah ve akşam saatlerinde) Google Maps Distance Matrix API üzerinden çekerek oluşturulan database içerisine günlük olarak kayıt altına alma planı oluşturulmuştur. Böylece zamanla büyüyen bir trafik verisi arşivi elde edilerek, modelin ileri tarihli tahmin yapma yetkinliği artırılmak istenmiştir.

Ancak bu yöntemin uygulanmasında çeşitli zorluklarla karşılaşmıştır. Google Maps API, yalnızca gerçek zamanlı trafik verisi sunduğundan, belirli bir geleceğe yönelik trafik öngörüsüne izin vermemektedir.[13] Ayrıca API kullanımındaki kota limitleri, maliyet hesaplamaları ve veri bütünlüğünü sürdürülebilir biçimde sağlama gerekliliği gibi faktörler, bu yaklaşımın uygulanabilirliğini sınırlamıştır.

Bu nedenlerle proje yönü değiştirilerek, hazır ve güvenilir bir açık veri kaynağı olan İstanbul Büyükşehir Belediyesi Açık Veri Portalı (<https://data.ibb.gov.tr>) tercih edilmiştir. Bu platform, İstanbul genelindeki trafik sensörlerinden saatlik çözünürlükte veri sağlayarak projeye yüksek zaman duyarlılığı kazandırmaktadır.

Veri toplama sürecinde, en güncel tarihli ve bütünlüğü yüksek veri olarak Ocak 2025 ayı seçilmiş; modele zamansal çeşitlilik kazandırmak amacıyla Aralık 2024 ve Kasım 2024 aylarına ait veriler de kapsam dahiline alınmıştır. Böylece modelin farklı gün, hafta ve hava koşullarını temsil edebilecek örüntüleri öğrenmesi amaçlanmıştır. Proje geliştirilmeye devam edildikçe veri setini genişletmek planlanmıştır.

Bu veriler, veri tabanı sistemlerine aktarılmaksızın doğrudan CSV dosyaları olarak projeye entegre edilmiştir. Bu tercih, erken prototipleme aşamasında sistemin karmaşıklığını azaltmak, hızlı model eğitimi ve test sürecine odaklanmak amacıyla yapılmıştır.

Veri setinde yer alan anahtar değişkenler tabloda özetlenmiştir.

Özellik	Açıklama	Veri Tipi
DATE_TIME	Ölçüm zamanı	DateTime
LATITUDE	Enlem koordinatı	Float
LONGITUDE	Boylam koordinatı	Float
GEOHASH	Coğrafi hash değeri	String
MINIMUM_SPEED	Minimum hız (km/h)	Float
MAXIMUM_SPEED	Maksimum hız (km/h)	Float
AVERAGE_SPEED	Ortalama hız (km/h)	Float
NUMBER_OF_VEHICLES	Araç sayısı	Integer

Şekil 4.1: Veri Seti Özellikleri ve Değişken Tanımları

Gelecek aşamalarda bu veri yapısının, **PostgreSQL** veya benzeri ilişkisel veritabanlarına aktarılması; böylece daha etkin veri sorgulama, zaman serisi analizi ve kullanıcıya özel veriye erişim olanaklarının artırılması planlanmaktadır.[14]

5. VERİ HAZIRLAMA VE MODEL EĞİTİMİ

Bu bölümde, ham verinin ön işleme adımlarından geçirilerek dengelenmesi, özellik mühendisliği ile anlamlı hale getirilmesi ve ardından model eğitimi ile sınıflandırma sisteminin oluşturulması adımları detaylandırılmaktadır. Ayrıca farklı aşamalardaki model başarımları karşılaştırmalı olarak sunulmakta ve görsellerle desteklenmektedir.

İstanbul Büyükşehir Belediyesi'ne ait üç farklı aya (Kasım 2024, Aralık 2024, Ocak 2025) ait trafik verileri birleştirilmiş, veri üzerinde çeşitli ön işleme ve dengeleme adımları gerçekleştirilmiş ve nihayetinde Random Forest algoritması ile trafik yoğunluğu sınıflandırması yapılmıştır.

5.1. Veri Birleştirme ve Ön İşleme

Aşağıda özetlenen veri hazırlık sürecinde üç adet CSV formatındaki dosya pandas kütüphanesi kullanılarak okunmuş ve tek bir veri kümesinde birleştirilmiştir:

- ibb_traffic_2024_11.csv
- ibb_traffic_2024_12.csv
- ibb_traffic_2025_01.csv

Ham veri setinde bulunan DATE_TIME, LATITUDE, LONGITUDE, AVERAGE_SPEED, MINIMUM_SPEED, MAXIMUM_SPEED ve NUMBER_OF_VEHICLES değişkenleri temel alınarak yeni türevsel değişkenler oluşturulmuştur. İlk olarak Zaman bilgisini içeren DATE_TIME sütunu timestamp formatına dönüştürülerek geçersiz kayıtlar elenmiştir. Ardından saat (hour), haftanın günü (day_of_week) ve haftasonu bilgisi (is_weekend) gibi türevsel özellikler elde edilmiştir.

- hour: Saat bilgisi (0–23)
- day_of_week: Haftanın günü (0: Pazartesi – 6: Pazar)
- is_weekend: Haftasonu kontrolü (1: Cumartesi/Pazar, 0: diğer)

Hedef değişken olarak ise AVERAGE_SPEED değeri, belirli eşiklere göre 3 sınıfa ayrılmıştır:

- 0: Az Yoğun (> 40 km/s)
- 1: Orta Yoğun (21–40 km/s)
- 2: Yoğun (≤ 20 km/s)

5.2. Hedef Değişken Tanımı

Trafik seviyesi, ortalama hız değerine göre üç sınıfta kategorize edilmiştir:

```
```python
def classify_traffic(avg_speed):
 if avg_speed <= 30: # Yoğun trafik
 return 2
 elif avg_speed <= 50: # Orta trafik
 return 1
 else: # Az trafik
 return 0
```
```

Şekil 5.2.1: Trafik seviyesine ait açıklayıcı kod bloğu

Sınıflandırma Kriterleri:

Sınıf 0 (Az Trafik): Ortalama hız > 50 km/h

Sınıf 1 (Orta Trafik): $30 \text{ km/h} < \text{Ortalama hız} \leq 50 \text{ km/h}$

Sınıf 2 (Yoğun Trafik): Ortalama hız $\leq 30 \text{ km/h}$

5.3. Veri Dengesizliği ve Dengeleme

İlk aşamada yalnızca üç temel özellik kullanılarak bir model oluşturulmuş; ancak modelin sınıf ayrımındaki başarısı düşüktür. Ham veri incelendiğinde, traffic_level=0 (az yoğun) sınıfına ait verilerin, diğer sınıflara kıyasla çok fazla olduğu görülmüştür. Bu durum, modelin dengesiz veri ile eğitildiğinde yalnızca baskın sınıfı öğrenmesine yol açmıştır. İlk model çıktısı (Şekil 3.2) bu sorunu açıkça ortaya koymaktadır. Aşağıda, bu sınırlı özellik setiyle elde edilen performans çıktısı sunulmaktadır:

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.64 | 1.00 | 0.78 | 227309 |
| 1 | 0.00 | 0.00 | 0.00 | 107383 |
| 2 | 0.00 | 0.00 | 0.00 | 18101 |
| accuracy | | | 0.64 | 352793 |
| macro avg | 0.21 | 0.33 | 0.26 | 352793 |
| weighted avg | 0.42 | 0.64 | 0.50 | 352793 |

Şekil 5.3.1: Trafik Yoğunluğu Tahmin Performansı

Bu tabloda görüldüğü üzere, sınıf 0 (Az Yoğun) dışındaki trafik seviyeleri (1 ve 2) neredeyse hiç tanınmamaktadır (precision, recall ve f1-score = 0.00). Bunun temel nedeni, sınıf dengesizliğidir. Az yoğun trafik verisi, diğer sınıflara kıyasla çok daha fazladır ve model, bu dengesizliği aşmakta zorlanmaktadır.

Veri setindeki ciddi sınıf dengesizliğini gidermek amacıyla, **SMOTE (Synthetic Minority Over-sampling Technique)** uygulanmıştır.[15] Bu yöntemle, azınlık sınıflarına ait örneklerin sayısı yapay olarak artırılmış ve dengeli bir eğitim kümesi oluşturulmuştur. Dengelemeyi takiben modelin tüm sınıflar için daha dengeli çıktılar verdiği gözlemlenmiştir.

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.45 | 0.48 | 0.47 | 18190 |
| 1 | 0.37 | 0.14 | 0.20 | 18296 |
| 2 | 0.45 | 0.71 | 0.55 | 18333 |
| accuracy | | | 0.44 | 54819 |
| macro avg | 0.42 | 0.44 | 0.40 | 54819 |
| weighted avg | 0.42 | 0.44 | 0.40 | 54819 |

Şekil 5.3.2:Smote Tekniği Trafik Yoğunluğu Tahmin Performansı

Bu adımdan sonra sınıf 1 ve sınıf 2 için de anlamlı metrik değerleri elde edilmiştir. Örneğin, sınıf 2 (Yoğun) için f1-score 0.55 seviyesine çıkmıştır.

5.4.Özellik Setinin Genişletilmesi

Modelin sınıflandırma başarımını artırmak amacıyla yalnızca zaman bilgisi değil, aynı zamanda trafikle doğrudan ilişkili fiziksel veriler de özellik kümesine dahil edilmiştir. Bu amaçla MINIMUM_SPEED, MAXIMUM_SPEED ve NUMBER_OF_VEHICLES değişkenleri de modele eklenmiştir:

```
# 5. Özellik ve etiket belirle
X = df_balanced[["hour", "day_of_week", "is_weekend"]]
y = df_balanced["traffic_level"]
```

Şekil 5.4.1:Özellik ve etiket belirleme kod örneği

```
# 6. Genişletilmiş özellik seti
X = df_balanced[["hour", "day_of_week", "is_weekend",
"MINIMUM_SPEED", "MAXIMUM_SPEED", "NUMBER_OF_VEHICLES"
]]
y = df_balanced["traffic_level"]
```

Şekil 5.4.2:Genişletilmiş özellik seti kod örneği

Bu genişletilmiş özellik seti ile eğitilen model, önceki yapılarına göre kayda değer bir performans artışı göstermiştir. Özellikle f1-score, precision ve recall metrikleri tüm sınıflar için dengeli ve yüksek çıkmıştır.

| Genişletilmiş Özelliklerle Tahmin Performansı: | | | | |
|--|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.94 | 0.92 | 0.93 | 53181 |
| 1 | 0.83 | 0.82 | 0.83 | 53054 |
| 2 | 0.89 | 0.92 | 0.90 | 53493 |
| accuracy | | | 0.89 | 159728 |
| macro avg | 0.89 | 0.89 | 0.89 | 159728 |
| weighted avg | 0.89 | 0.89 | 0.89 | 159728 |

Şekil 5.4.3:Genişletilmiş Özelliklerle Tahmin Performansı

Sonuç olarak:

- **Accuracy:** 0.89
- **Macro avg f1-score:** 0.89
- **Sınıf 2 (Yoğun) için f1-score:** 0.90

Bu gelişme, eklenen hız ve araç sayısı değişkenlerinin modelin karar sınırlarını daha anlamlı şekilde oluşturmaya yardımcı olduğunu göstermektedir.

5.5.Model Eğitimi ve Kaydı

Model eğitimi sürecinde **Random Forest Classifier** algoritması kullanılmıştır. Bu algoritma, sınıf dengesizliği ile baş edebilmesi, açıklanabilirlik sağlaması ve aşırı öğrenmeyi (overfitting) kontrol altına alabilmesi nedeniyle tercih edilmiştir. [16] Model, train_test_split ile %80 eğitim, %20 test verisi üzerinden değerlendirilmiştir.

Model, joblib kütüphanesi ile .pkl formatında kaydedilmiş ve Flask API ile entegrasyon amacıyla yüklenmiştir.

Kod içerisindeki; joblib.dump(model, "trafik_model.pkl") modeli kaydeder, joblib.load("trafik_model.pkl") kaydedilmiş modeli tekrar yükler, model.predict(...) Yeni veri ile tahmin yapılmasını sağlar.

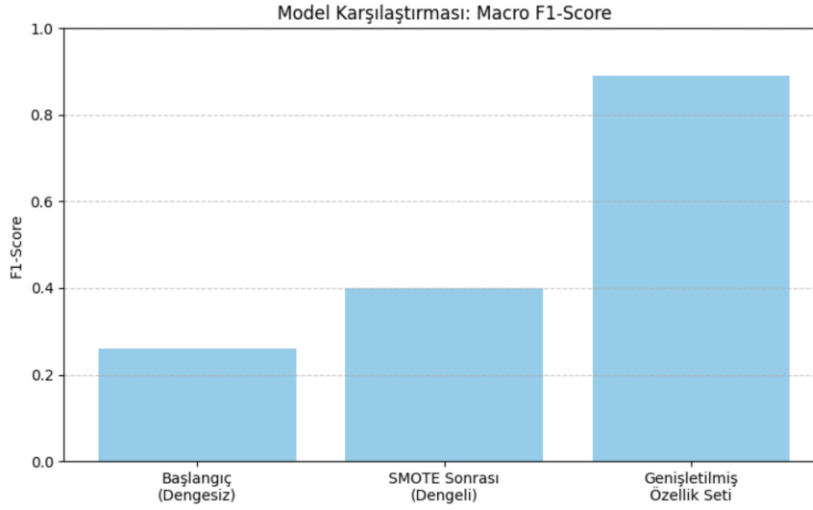
5.6.Metrik Karşılaştırması ve Grafik Oluşturma

Karşılaştırmalı değerlendirme için üç ayrı model çıktısı metrik olarak karşılaştırılabilir:

| Model Yapısı | Accuracy | Macro F1 | Sınıf 2 F1 |
|----------------------------|----------|----------|------------|
| Temel özellik seti | 0.64 | 0.26 | 0.00 |
| Dengeleme sonrası | 0.44 | 0.40 | 0.55 |
| Genişletilmiş özellik seti | 0.89 | 0.89 | 0.90 |

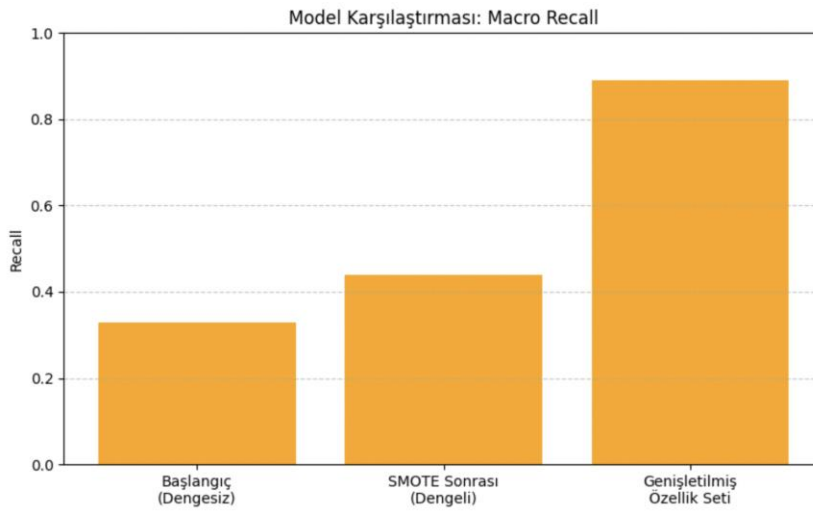
Bu tablo, özellikle sınıf 2'nin (yüksek yoğunluk) daha önce neredeyse hiç tahmin edilemediğini, ancak özellik genişletmesi ve dengeleme ile yüksek doğruluk seviyelerine ulaşıldığını göstermektedir.

Makro ortalama F1-score ve recall metrikleri karşılaştırmalı olarak aşağıdaki grafiklerde sunulmuştur:



Şekil 5.6.1: Model Karşılaştırması: Macro F1-Score

Bu grafik, üç farklı modelin makro ortalama F1-score değerlerini göstermektedir. Dengesiz veri ile elde edilen 0.26'lık değer, SMOTE sonrası 0.40'a yükselmiş, genişletilmiş özellik setiyle 0.89'a kadar çıkmıştır. Bu artış, modelin tüm sınıfları başarılı şekilde ayırt edebildiğini göstermektedir.



Şekil 5.6.2: Model Karşılaştırması: Macro Recall

Recall metriği, her sınıfın ne kadarının doğru tahmin edildiğini gösterir. Başlangıç modeli yalnızca sınıf 0'ı öğrenmiştir. SMOTE sonrası her sınıf biraz daha iyi tahmin edilmiştir. Genişletilmiş özellikli model ise neredeyse %90 başarı ile tüm sınıfları kapsayıcı sonuçlar vermiştir.

Sonuç olarak bu grafiklerden açıkça görülmektedir ki:

- Zaman bilgisiyle sınırlı ve dengesiz veri kullanıldığında model yalnızca bir sınıfı tahmin edebilmiştir.
- SMOTE ile dengesizlik giderilmiş olsa da sınırlı özelliklerle başarı kısıtlı kalmıştır.
- Genişletilmiş özellik kümesi sayesinde modelin sınıflandırma yetkinliği belirgin biçimde artmıştır (accuracy = %89, f1-score = 0.89).

Modelin başarı performansı, uygulanan her stratejiyle sistematik olarak artmış ve nihai model, ileri tarihli trafik tahminlerinde kullanılabilecek güvenilir bir yapı sunmuştur. Özellikle NUMBER_OF_VEHICLES ve hız bilgilerinin dahil edilmesi, modelin sınıflar arası ayrım yeteneğini ciddi şekilde artırmıştır.

6.MODEL ENTEGRASYONU

Makine öğrenmesi modellerinin başarıyla eğitilmesi, gerçek dünya uygulamalarına entegrasyonu ile anlam kazanır. Bu bağlamda, geliştirilen trafik yoğunluğu tahmin modeli, web tabanlı bir kullanıcı arayüzü (frontend) üzerinden çalıştırılabilir hale getirilmiştir. Modelin entegrasyon süreci, Python tabanlı bir mikro servis çatısı olan **Flask** aracılığıyla gerçekleştirilmiş; modelin API aracılığıyla tahmin sonuçlarını dış dünyaya sunması sağlanmıştır.

6.1.Flask Tabanlı API Yapısı

Flask, basit ve modüler yapısı sayesinde özellikle prototipleme ve akademik projelerde sıkça tercih edilen hafif bir web framework'üdür.[17] Bu projede, Flask kullanılarak RESTful bir API tasarlanmış ve eğitim sonucu elde edilen model bu API üzerinden kullanılabilir hale getirilmiştir.

Servis mimarisi şu adımları içermektedir:

1. Model Yükleme:

Model, `joblib.load("trafik_model.pkl")` komutu ile belleğe alınmakta ve tüm isteklerde yeniden eğitim ihtiyacı olmadan kullanılmaktadır.

2. /predict Endpoint'i:

Kullanıcıdan alınan rota ve tarih/saat bilgileri (örneğin "2025-05-27T08:30" formatında), bu endpoint'e JSON formatında gönderilmektedir.

3. Özellik Çıkarımı:

API tarafında gelen datetime verisinden hour, day_of_week, is_weekend gibi özellikler çıkarılmakta; bu veri modeli beslemek üzere dönüştürülmektedir.

4. Tahmin İşlemi:

Flask uygulaması, önceden eğitilmiş modeli çağırarak tahmin işlemini gerçekleştirmekte ve tahmin edilen trafik yoğunluğu sınıfını (0, 1, 2) JSON formatında frontend tarafına iletmektedir.

5. CORS Yapılandırması:

React ile geliştirilen kullanıcı arayüzünün farklı bir port üzerinden çalışması nedeniyle ortaya çıkan **CORS (Cross-Origin Resource Sharing)** hataları, flask-cors kütüphanesi aracılığıyla çözülmüştür.

6.2.Flask Tercihinin Gerekçeleri

Model servisleştirme aşamasında birçok web framework'ü arasından Flask'ın tercih edilmesinin temel nedenleri şu şekilde sıralanabilir:

- **Hafiflik ve Hızlı Kurulum:** Flask, ek yapılandırma gerektirmeyen yalın bir yapıya sahiptir. Böylece, geliştiricinin doğrudan uygulamanın mantıksal akışına odaklanmasına olanak tanır.
- **Modülerlik ve Genişletilebilirlik:** İleride yapılacak özellik eklemeleri (örneğin kullanıcı doğrulama, model güncelleme) Flask üzerinde kolayca entegre edilebilir.
- **Python ile Doğal Entegrasyon:** Modelin Python diliyle eğitilmiş olması, Flask ile doğrudan ve sorunsuz bir entegrasyon sağlamıştır. joblib ile kayıt edilen model, Flask üzerinde doğrudan çalıştırılabilmiştir.
- **React Frontend ile Uyum:** JSON tabanlı veri iletişimi sayesinde React.js arayüzüyle haberleşme kolaylıkla sağlanmıştır.

6.3.Uçtan Uca Entegrasyon Akışı

Aşağıda, modelin frontend ve backend bileşenleri arasında nasıl çalıştığı açıklanmaktadır:

1. **Kullanıcı Girdisi:** React tabanlı arayüzde kullanıcı, origin, destination ve datetime bilgilerini girer.
2. **API İsteği:** Bu bilgiler, axios kütüphanesi aracılığıyla Flask servisinin /predict endpoint'ine iletilir.
3. **Özellik Dönüşümü:** Flask tarafı bu tarih bilgisini saat, gün, haftasonu gibi modellenecek özelliklere çevirir.
4. **Tahmin ve Yanıt:** Eğitilmiş model tahminini yapar; örneğin 2 → “Yoğun Trafik”. Bu sonuç JSON formatında React'e döner.
5. **Görsel Gösterim:** Frontend tarafı, tahmin edilen sınıfa göre Google Maps üzerinde rotayı uygun renkle (yeşil, sarı, kırmızı) işaretler.

6.4.Geleceğe Yönelik Genişletme Önerileri

Flask ile oluşturulan bu temel yapı, ileride aşağıdaki işlevlerle zenginleştirilebilir:

- Kullanıcının geçmiş sorgularına dayalı öneri sistemi (ör. “önceki pazartesi saat 18:00’de burası yoğundu”).
- Harita üzerinde alternatif rotalara göre tahmin karşılaştırması.
- API erişimi için kullanıcı bazlı kimlik doğrulama.
- Gerçek zamanlı trafik yoğunluğu ile tahmin karşılaştırması yapılarak hata analizi.
- **Daha geniş tarih aralığı** için veri seti genişletmesi
- **Trafik yoğunluğu değişimi zaman çizelgesi** (line chart)
- **Geohash tabanlı** daha hassas konum tahmini ve grid bölgeleme
- **Mobil versiyon** ve bildirim desteği

7.ARAYÜZ GELİŞTİRME

Crowd Predictor projesinde, şehir içi trafik yoğunluğunun kullanıcıdan alınan rota ve zaman bilgileri doğrultusunda tahmin edilmesini ve bu tahminlerin görsel olarak harita üzerinde sunulmasını amaçlayan bir web tabanlı uygulamadır. Proje kapsamında geliştirilen sistem, hem veri bilimi hem de kullanıcı arayüzü tasarımı açısından bütüncül bir yaklaşım benimsemektedir. Bu bağlamda, sadece yapay zekâ modelinin sunduğu tahminlerin doğruluk oranı değil, aynı zamanda bu tahminlerin kullanıcıya anlaşılır, sezgisel ve işlevsel bir biçimde aktarılması da projenin temel öncelikleri arasında yer almaktadır.

Trafik yoğunluğu gibi zaman ve konum tabanlı değişkenliği yüksek olan bir olgunun doğru şekilde tahmin edilmesi ve kullanıcıya etkili biçimde sunulması, çok katmanlı bir yazılım mimarisi gerektirmektedir. Bu doğrultuda, sistemin kullanıcı arayüzü tarafında modern, hızlı ve yeniden kullanılabilir bileşenlerle geliştirme yapılmasına imkân tanıyan React.js kütüphanesi tercih edilmiştir. React.js’in duruma duyarlı bileşen mimarisi sayesinde, kullanıcı etkileşimlerine hızlı tepki veren ve kullanıcı dostu bir arayüz tasarımı hayata geçirilmiştir. Arayüz tasarımında sade, dinamik ve etkileşimli bir yapı hedeflenmiş; kullanıcıların kolayca rota bilgisi girebilmesi, zaman aralığı seçebilmesi ve tahmin edilen yoğunluğu anlık olarak harita üzerinde görebilmesi sağlanmıştır.

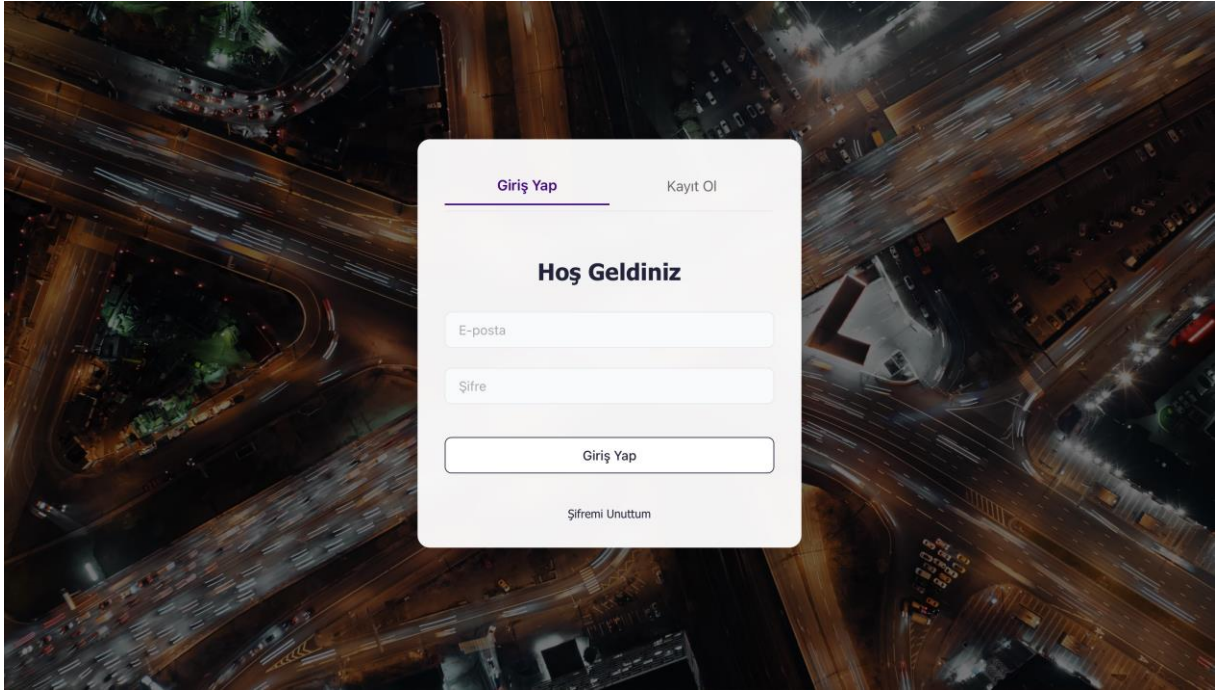
Harita entegrasyonu için ise Google Maps JavaScript API kullanılmış olup, bu API sayesinde kullanıcıların girdiği rota bilgileri dinamik olarak harita üzerinde çizilmekte ve tahmin edilen trafik yoğunluğu farklı renk skalaları ile görselleştirilmektedir. Bu sayede kullanıcı, gitmeyi planladığı güzergâhtaki yoğunluk seviyelerini sezgisel biçimde algılayabilmektedir. Ayrıca, kullanıcı konumuna en yakın noktaların ve önemli mekânların belirlenmesini sağlamak amacıyla Google Places API entegrasyonu gerçekleştirilmiştir. Böylece sistem, sadece rota üzerindeki yoğunluğu göstermekle kalmayıp, konum bazlı yardımcı öneriler sunarak kullanıcı deneyimini daha işlevsel hale getirmektedir.

Sonuç olarak, Crowd Predictor projesi, yapay zekâ destekli trafik tahmini sistemlerinin yalnızca teknik doğruluğu ile değil, aynı zamanda kullanıcıya erişebilir ve anlamlı biçimde sunulabilirliği ile de değerlendirildiği gerçeğinden yola çıkarak geliştirilmiştir. Bu kapsamda hem arka uçta güçlü bir tahmin modeli hem de ön uçta kullanıcı odaklı bir arayüz tasarımı bir araya getirilmiş, böylelikle şehir içi ulaşım planlamasını kolaylaştıran, yenilikçi ve işlevsel bir yazılım çözümü ortaya konmuştur.

7.1.Kullanıcı Giriş ve Kayıt Arayüzü

Geliştirilen React tabanlı web uygulaması, kullanıcıların sisteme erişimini ve sistemle etkileşim kurmasını kolaylaştırmak amacıyla üç temel ekrandan oluşacak şekilde yapılandırılmıştır. Bu ekranlar sırasıyla **kullanıcı giriş ekranı**, **kayıt ekranı** ve **harita görüntüleme ekranıdır**. Uygulama arayüzü tasarımında kullanıcı deneyimini ön planda tutan, sade ve işlevsel bir yaklaşım benimsenmiştir.[18]

Kullanıcıların sisteme giriş yapabilmesi veya yeni bir hesap oluşturabilmesi için geliştirilen arayüzde, modern bir yaklaşım olan "**Sekmeli Giriş/Kayıt Arayüzü**" (Tabbed Login/Register UI) tasarımı tercih edilmiştir. Bu yapı sayesinde kullanıcılar, ayrı sayfalara yönlendirilmeden tek bir ekran üzerinde hem giriş yapma hem de kayıt olma işlemlerini gerçekleştirebilmektedir. Sekmeli yapı, sayfa yenilenmesi (refresh) olmadan içerik geçişine olanak tanıdığı için daha hızlı ve kesintisiz bir kullanıcı deneyimi sunmaktadır. Aynı zamanda, bu yaklaşımın kullanıcı odaklı bir tasarım örneği olduğu, günümüzde pek çok modern web uygulamasında tercih edildiği bilinmektedir.



Şekil 7.1.1:Kullanıcı Giriş Yap Ekranı

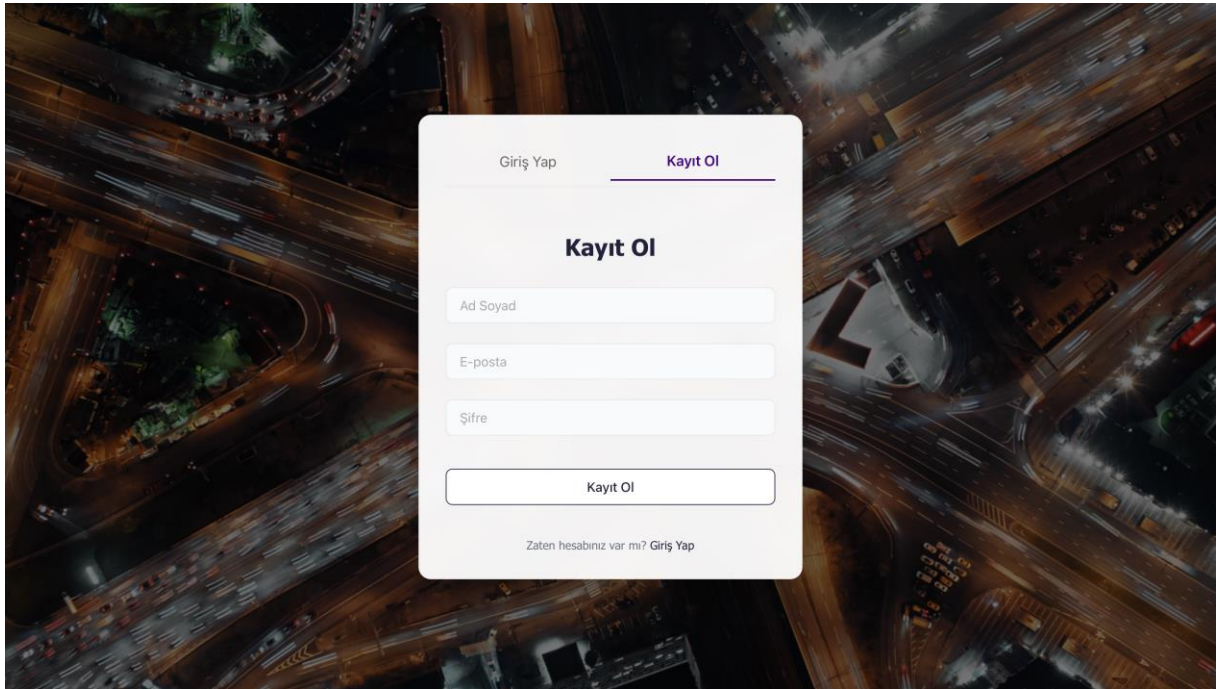
Kullanıcı Giriş Özellikleri:

Sisteme giriş işlemi sırasında kullanıcıdan **kullanıcı adı** ve **şifre** olmak üzere iki temel bilgi talep edilmektedir. Bu bilgilerin eksik, hatalı veya sistemde bulunmaması durumunda kullanıcıya yönelik çeşitli geri bildirim mekanizmaları devreye girmektedir:

- Eğer kullanıcı adı ya da şifre alanlarından herhangi biri boş bırakılmışsa, sistem kullanıcıyı bilgilendirmek amacıyla ekranda uyarı mesajı göstermektedir. Bu sayede formun eksik gönderilmesi engellenmekte ve kullanıcı yönlendirilmektedir.

- Kullanıcı adı sistem veri tabanında kayıtlı değilse, "Kullanıcı bulunamadı." ifadesiyle kullanıcıya, ilgili bilginin sistemde yer almadığı açıkça bildirilmektedir. Bu tür bir bilgilendirme, kullanıcıların yanlış kullanıcı adı girmeleri durumunda hızlı şekilde farkındalık kazanmalarını sağlamaktadır.
- Eğer kullanıcı adı doğru, ancak girilen şifre sistemde kayıtlı olanla uyuşmuyorsa, bu durumda "Şifre hatalı." uyarı mesajı ekranda gösterilmektedir. Böylelikle kullanıcıya özel ve açıklayıcı bir hata bildirimi sunulurken, sistemle kurduğu etkileşimin sağlıklı şekilde devam etmesi sağlanmaktadır.
- Kullanıcının kimlik doğrulaması başarıyla gerçekleştiğinde, sistem otomatik olarak **harita görüntüleme ekranına yönlendirme işlemi** gerçekleştirmektedir. Bu ekran, sistemin temel işlevlerinden biri olan trafik yoğunluğu görselleştirmesini sunmaktadır. Başarılı girişin ardından doğrudan bu ekranın açılması, kullanıcıya hızlı bir şekilde uygulamanın esas hizmetine ulaşma olanağı sunmaktadır.

Sonuç olarak, kullanıcı giriş ve kayıt arayüzü, modern kullanıcı arayüzü tasarımı ilkelerine uygun biçimde yapılandırılmış, etkileşimli, hızlı ve kullanıcı hatalarına karşı rehberlik edici bir yapıya sahiptir. Bu yönüyle, sadece sistemin kullanılabilirliğini artırmakla kalmayıp, aynı zamanda kullanıcı memnuniyetine de doğrudan katkı sağlamaktadır.



Şekil 7.1.2:Kullanıcı Kayıt Ol Ekranı

Kayıt Olma Özellikleri:

Kullanıcılardan kayıt işlemi sırasında aşağıdaki üç temel bilgiyi sağlamaları beklenmektedir:

1. **Kullanıcı Adı (Username)**
2. **E-posta Adresi (Email)**
3. **Şifre (Password)**

Bu üç alan, sistem açısından **zorunlu** olup, herhangi birinin eksik bırakılması durumunda formun gönderilmesine izin verilmemektedir. Bu sayede eksik bilgiyle yapılan kayıt denemeleri önlenmekte ve veri bütünlüğü sağlanmaktadır.

E-posta Doğrulama Mekanizması

Kayıt formunda girilen e-posta adresi, en temel düzeyde geçerlilik kontrolünden geçirilmekte; bu kapsamda e-posta adresinin "@" karakterini içerip içermediği kontrol edilmektedir. "@" karakteri içermeyen bir e-posta adresi, biçimsel olarak geçerli kabul edilmediği için kullanıcıya uygun bir hata mesajı gösterilmekte ve kayıt işlemi tamamlanamamaktadır. Bu kontrol, daha gelişmiş doğrulama süreçleri için ön adım niteliği taşımakta ve kullanıcıdan gelen bilgilerin tutarlılığını artırmaktadır.

Benzersizlik Kontrolleri

Sistemdeki kullanıcı bilgilerinin benzersizliğini sağlamak amacıyla, kayıt sırasında girilen **kullanıcı adı** ve **e-posta adresi**, mevcut kayıtlarla karşılaştırılmaktadır. Eğer kullanıcı adı veya e-posta adresi daha önce sisteme kayıtlı bir kullanıcıya aitse, sistem bu durumu kullanıcıya bildirerek kayıt işlemi durdurmaktadır. Bu kontroller sayesinde her kullanıcıya özgü bir kimlik sağlanmakta ve veri tabanında çakışmaların önüne geçilmektedir.

Veri Tabanı Kaydı

Tüm geçerlilik kontrollerinden başarıyla geçen kullanıcı bilgileri, **MySQL ilişkisel veri tabanı** sistemine kaydedilmektedir. MySQL, veri bütünlüğünü koruyan ve güvenli veri depolamaya olanak tanıyan yaygın bir veritabanı yönetim sistemidir. Kayıt işlemi sonucunda kullanıcıya ait bilgiler, ilgili tabloda düzenli biçimde saklanmakta; böylece kullanıcı daha sonraki giriş denemelerinde bu bilgilerle kimliğini doğrulayabilmektedir.

Bu yapıyla kayıt olma modülü, hem kullanıcı güvenliğini hem de sistemsel tutarlılığı sağlayacak şekilde tasarlanmış; kullanıcı odaklı ve sürdürülebilir bir yazılım mimarisi çerçevesinde uygulamaya entegre edilmiştir. Ayrıca, kullanıcı girdilerine verilen anlık geri bildirimlerle etkileşimli bir deneyim sunulmuş, hata olasılıkları minimuma indirilerek kullanıcı memnuniyetinin artırılması hedeflenmiştir.

7.2. Veri Tabanı Altyapısı

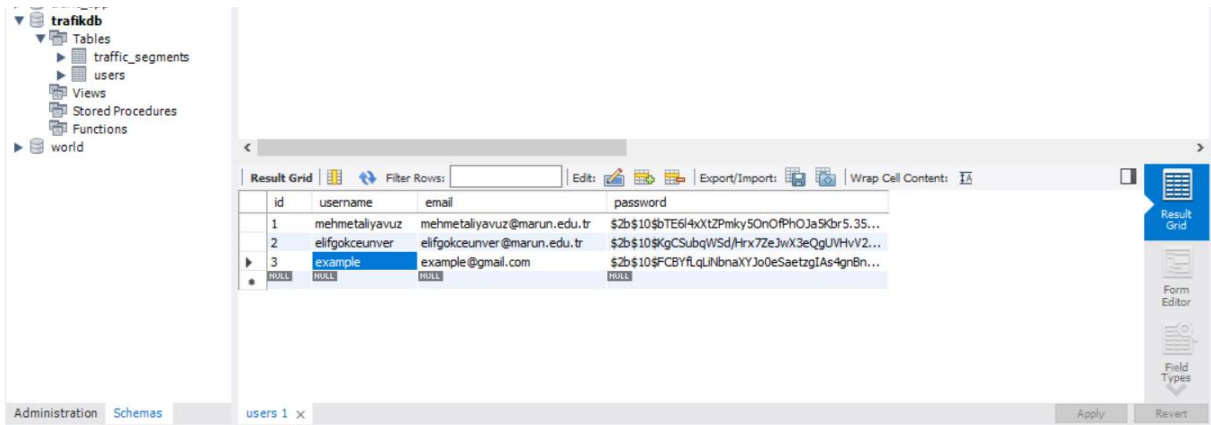
Geliştirilen web tabanlı uygulamanın güvenilir ve sürdürülebilir biçimde çalışabilmesi için güçlü bir arka uç veri yönetimi altyapısına ihtiyaç duyulmuştur. Bu kapsamda, kullanıcıdan ön yüz (frontend) aracılığıyla alınan verilerin güvenli ve tutarlı biçimde depolanabilmesi amacıyla **MySQL** tabanlı bir ilişkisel veri tabanı sistemi tercih edilmiştir. MySQL, açık kaynak kodlu yapısı, yüksek performansı, veri bütünlüğü sağlama yeteneği ve geniş topluluk desteği sayesinde, web uygulamaları için oldukça yaygın biçimde kullanılan bir veritabanı yönetim sistemidir.

Uygulama kapsamında kullanıcıların kayıt ve giriş işlemleri sırasında ilettikleri bilgiler, **veritabanı ile arayüz arasında kurulan bağlantı** aracılığıyla arka uça işlenmekte ve uygun doğrulama kontrollerinden geçirildikten sonra kalıcı olarak **“trafficdb”** isimli veri tabanına kaydedilmektedir. Kullanıcıya ait veriler, bu veri tabanı içerisinde yer alan ve kullanıcı bilgilerini içermek üzere özel olarak yapılandırılmış **“users”** adlı tabloda saklanmaktadır. Bu

tablo; kullanıcı adı, e-posta adresi, şifre hash değeri, kullanıcı ID'si gibi temel alanlardan oluşmaktadır.

Veri güvenliği, sistem mimarisinde kritik bir öncelik olarak ele alınmıştır. Özellikle şifre (parola) gibi hassas verilerin doğrudan veri tabanında açık biçimde tutulması, kullanıcı gizliliğini tehlikeye atabilecek ciddi bir güvenlik açığı oluşturmaktadır. Bu nedenle, şifrelerin doğrudan saklanması yerine, modern bir şifreleme algoritması olan **bcrypt** yöntemi kullanılmıştır. JavaScript tabanlı **bcryptjs** kütüphanesi aracılığıyla şifreler kayıt öncesinde **hash** işlemine tabi tutulmakta ve veri tabanına yalnızca bu hash değeri kaydedilmektedir. Böylece, olası veri ihlallerinde dahi kullanıcı şifrelerinin doğrudan ele geçirilmesi önlenmiş olmakta; sistem, çağdaş siber güvenlik standartlarıyla uyumlu hale getirilmektedir.

Bcrypt algoritması, sadece şifreleri tek yönlü olarak şifrelemekle kalmaz; aynı zamanda hash işlemi sırasında kullanılan **salt (tuz)** mekanizması sayesinde her bir şifrenin farklı bir biçimde kodlanmasını sağlar. Bu durum, aynı şifrenin farklı kullanıcılar tarafından kullanılması halinde dahi veri tabanında benzersiz ve ayırt edilebilir şekilde saklanmasına olanak tanır. Böylece **rainbow table** ve **brute-force** saldırılarına karşı ileri düzey koruma sağlanmış olur.



| id | username | email | password |
|----|----------------|-----------------------------|---|
| 1 | mehmetaliyavuz | mehmetaliyavuz@marun.edu.tr | \$2b\$10\$bTE6l4x1ZPmkY5OnOfPhOJa5Kbr5.35... |
| 2 | elifgokceunver | elifgokceunver@marun.edu.tr | \$2b\$10\$KqCSubqWSd/Hrx7ZeJwX3eQgUHVHV2... |
| 3 | example | example@gmail.com | \$2b\$10\$FCBYflqUNbnaXYJo0eSaeztgIA54gnBn... |

Şekil 7.2.1:MySQL Tablo Kayıtları

Sonuç olarak, Crowd Predictor projesinin veri tabanı altyapısı, hem işlevsellik hem de güvenlik açısından özenle planlanmış ve uygulanmıştır. MySQL'in sunduğu ilişkisel yapı sayesinde veri bütünlüğü korunmuş; bcryptjs tabanlı şifreleme yaklaşımı ile kullanıcı verilerinin gizliliği ve güvenliği en üst düzeye çıkarılmıştır. Bu altyapı, uygulamanın kullanıcı yönetimi modülünün sürdürülebilirliğini ve güvenilirliğini doğrudan destekleyen temel unsurlardan biri olarak işlev görmektedir.

7.3.Harita Entegrasyonu ve Kullanılan Teknolojiler

Crowd Predictor projesinin temel bileşenlerinden biri olan **trafik yoğunluğu tahmini ve görselleştirme işlemi**, kullanıcı deneyiminin merkezinde yer almaktadır. Bu bağlamda, sistem mimarisi kullanıcıların sisteme başarıyla giriş yapmalarının ardından otomatik olarak **harita görüntüleme ekranına** yönlendirilmesini sağlamaktadır. Bu ekranda, **Google Maps API servisleri** aracılığıyla interaktif, dinamik ve gerçek zamanlı bir trafik görselleştirme altyapısı sunulmaktadır. Harita tabanlı görselleştirmenin sistemle bütünleşik biçimde çalışması, hem kullanıcı etkileşimini artırmakta hem de uygulamanın işlevselliğini görünür kılmaktadır.

Google Maps Platformu, dünya genelinde yaygın olarak kullanılan bir harita ve konum servisleri platformudur ve uygulamaya entegre edilen çeşitli API'leri sayesinde kullanıcıların konum bazlı etkileşimlerde bulunmasına olanak tanımaktadır. Crowd Predictor projesinde harita entegrasyonu için bu platformun farklı servisleri birlikte kullanılmıştır:

1. Maps JavaScript API

Bu API, Google Maps'in bir web sayfasına dinamik biçimde entegre edilebilmesini sağlayan temel bileşenlerden biridir. Harita bileşeni, React.js tabanlı kullanıcı arayüzüne doğrudan gömülerek kullanıcılara harita üzerinde konum görüntüleme, nokta (marker) yerleştirme ve rota çizimi gibi interaktif işlemler yapma imkânı sunmaktadır. Bu yapı, kullanıcıların harita üzerinden daha etkili biçimde bilgi almasını ve sistemin tahmin çıktılarıyla doğrudan etkileşime geçmesini mümkün kılmaktadır.

2. Places API

Google Places API, kullanıcıların adres, mekan adı ya da konum bilgisi girmelerine olanak tanıyan, konum bazlı öneri sistemleri sunan bir servistir. Bu proje kapsamında **otomatik adres tamamlama (autocomplete)** ve **yer önerileri** özelliklerinden yararlanılmıştır. Kullanıcı bir adres ya da konum girmeye başladığında, bu API aracılığıyla ilgili sonuçlar gerçek zamanlı olarak öneri listesi şeklinde sunulmakta; bu sayede kullanıcı girdilerinin doğruluğu artırılmakta ve kullanım kolaylığı sağlanmaktadır.

3. React-Axios Kütüphanesi

Harita bileşeni ve trafik tahmin modeli arasında veri alışverişi yapılabilmesi için ön yüz ile arka uç arasında HTTP tabanlı bir iletişim kurulmuştur. Bu amaçla, React uygulamasında yaygın olarak kullanılan ve asenkron veri taleplerini yönetebilen **Axios** kütüphanesinin React sürümü olan **react-axios** kullanılmıştır. Kullanıcı tarafından belirlenen başlangıç ve varış noktalarına ait rota bilgileri, bu kütüphane aracılığıyla Flask tabanlı REST API'ye gönderilmekte; ilgili trafik yoğunluğu tahmini alındıktan sonra harita üzerine görsel olarak yansıtılmaktadır. Böylece, ön yüz ile arka uç arasındaki veri alışverişi güvenilir ve optimize edilmiş bir yapı üzerinden gerçekleşmektedir.

Sonuç olarak, Crowd Predictor projesi kapsamında Google Maps teknolojileri ile gerçekleştirilen harita entegrasyonu; konum bazlı veri girişlerinin kolaylaştırılması, kullanıcı girdilerinin doğruluk oranının artırılması ve tahmin sonuçlarının görsel olarak etkili biçimde sunulması açısından kritik bir rol üstlenmektedir. Kullanılan tüm bu teknolojiler, projenin kullanıcı odaklı, modern ve yüksek etkileşimli bir sistem olmasını mümkün kılmıştır.

7.4. Harita Görüntüleme Ekranına Genel Bakış

Crowd Predictor uygulamasının kullanıcı etkileşimini en yoğun şekilde barındıran bileşenlerinden biri olan **harita görüntüleme ekranı**, kullanıcıların trafik tahmin sonuçlarını doğrudan ve görsel biçimde takip edebileceği, etkileşimli bir arayüz olarak tasarlanmıştır. Bu ekran, kullanıcı sisteme giriş yaptıktan sonra yönlendirildiği ana sayfa niteliğindedir ve uygulamanın temel işlevlerini bütüncül bir şekilde bir araya getirmektedir. Harita bileşeni, Google Maps JavaScript API entegrasyonu ile dinamik olarak çalışmakta olup, çeşitli görsel ayarlar ve kullanıcı seçenekleriyle zenginleştirilmiştir.

Varsayılan Konumlandırma ve Görsel Ayarlar

Harita, sistem tarafından varsayılan olarak **İstanbul ili üzerine konumlandırılmış** bir başlangıç noktasıyla yüklenmektedir. Bu tercih, projenin hedef kitlesi ve kullanım senaryoları doğrultusunda yapılmış olup, İstanbul'un yoğun ve değişken trafik yapısı göz önünde bulundurularak kullanıcıya gerçekçi bir trafik tahmin deneyimi sunmayı amaçlamaktadır.

Kullanıcılara farklı harita modları arasında geçiş yapabilme imkânı sunulmuştur. Harita arayüzünün **sol üst bölümünde** yer alan mod seçme paneli sayesinde kullanıcı, harita görünümünü kendi tercihlerine göre özelleştirebilmektedir. Bu panel aracılığıyla:

- **“Harita” (map)** görünümü ile sade bir şehir planı elde edilebilir,
- **“Uydu” (satellite)** modu ile yerleşim yerlerinin ve yolların gerçek uydu görüntüleri izlenebilir,
- Ayrıca **“Arazi” (terrain)** ve **“Etiketler” (labels)** gibi ek görünüm seçenekleriyle coğrafi özellikler veya yol isimleri gibi yardımcı veriler harita üzerine yansıtılabilir.

Bu modlar, kullanıcıya farklı senaryolar altında daha fazla bağlamsal bilgi sunarak yön bulma sürecini kolaylaştırmakta ve harita kullanımını daha esnek hale getirmektedir.

Rota Oluşturma ve Marker (İşaretleyici) Sistemi

Harita ekranı, kullanıcıya başlangıç ve varış noktalarını belirtme imkânı sunan iki ayrı **adres giriş kutusu** içermektedir. Bu alanlar üzerinden kullanıcı, tahmin edilmesini istediği rotayı tanımlar. Adres girişleri, Google Places API desteğiyle çalışmakta olup otomatik tamamlama özelliği ile kullanıcıların doğru ve hızlı seçim yapmasını sağlamaktadır.

Kullanıcının belirlediği başlangıç ve varış noktaları arasında bir **rota oluşturulur** ve bu rota, harita üzerine görsel olarak çizilir. Rota boyunca **marker (işaretleyici) öğeleri** başlangıç ve bitiş noktalarına yerleştirilerek kullanıcıya konum bilgisi hakkında ek referanslar sağlanır.

Trafik Yoğunluğu Sınıfına Göre Renk Kodlaması

Flask API üzerinden makine öğrenmesi tabanlı trafik tahmin modeli tarafından üretilen çıktı, ilgili rotaya karşılık gelen **trafik yoğunluğu sınıfını** belirtir. Bu sınıf, üç seviyeli bir yapıdadır ve Google Maps üzerinde **renkli çizgiler** aracılığıyla görsel olarak ifade edilir. Renk kodlaması şu şekilde yapılandırılmıştır:

- **Yeşil Renk (0):** Az Yoğun Trafik
- **Sarı Renk (1):** Orta Yoğun Trafik
- **Kırmızı Renk (2):** Yoğun Trafik

Bu renkli görselleştirme sistemi, kullanıcıya rota üzerindeki olası trafik durumunu sezgisel bir biçimde aktarmakta; karar alma sürecinde hızlı bir değerlendirme yapmasına yardımcı olmaktadır.

Tam Ekran Modu ve Erişilebilirlik Özellikleri

Harita bileşeni, kullanıcı isteğine bağlı olarak **tam ekran moduna** da alınabilmektedir. Bu mod, özellikle küçük ekranlı cihazlarda veya dikkat dağınıklığını azaltmak isteyen kullanıcılar için avantaj sağlamaktadır. Tam ekran görünüm, harici menülerin veya görsel öğelerin ekran dışına alınmasıyla, haritanın tüm ekranı kaplamasını mümkün kılar. Bu da daha geniş bir görüş alanı ve daha yüksek **erişilebilirlik** sunarak kullanıcı odaklı tasarım anlayışını destekler.

Sonuç olarak, harita görüntüleme ekranı, sistemin sunduğu trafik tahmin sonuçlarının kullanıcıya aktarılmasında merkezi bir rol üstlenmektedir. Farklı harita modları, adres girişi özellikleri, dinamik rota oluşturma ve renk kodlamalı trafik görselleştirmesi gibi bileşenler sayesinde kullanıcılar, trafik durumunu hem işlevsel hem de görsel olarak etkili bir biçimde takip edebilmekte; bu da uygulamanın kullanılabilirliğini ve etkileşim kalitesini artırmaktadır.

7.5.Karşılaşılan Sorunlar

Crowd Predictor projesi, temel olarak kullanıcılara belirli bir tarih ve saat aralığı için öngörülse trafik yoğunluğu verisi sunmayı amaçlayan, konum tabanlı bir tahmin ve görselleştirme sistemidir. Bu temel amaca ek olarak, projeye başlangıçta ikinci bir hedef daha entegre edilmeye çalışılmıştır: İstanbul ili genelinde on iki aya ait trafik yoğunluk verilerinden yararlanarak yıllık ortalama trafik yoğunluğu haritasının oluşturulması. Bu ek modül, kullanıcılara genel trafik eğilimlerini sunarak uzun vadeli planlamalarda fayda sağlayabilecek bir altyapı sunmayı hedeflemiştir.

Bu bağlamda, İstanbul Büyükşehir Belediyesi (İBB) Açık Veri Portalı üzerinden erişilebilen "Saatlik Trafik Yoğunluk Veri Seti" projede kullanılmak üzere tercih edilmiştir. Ancak bu verisetinin en güncel versiyonu, İBB tarafından 2025 yılı Ocak ayında yayımlandığı için, sistemde 2024 yılına ait trafik verileri kullanılarak işlem yapılması uygun görülmüştür. Bu veriler, proje dizininde yer alan "data" klasörü altında her ay için ayrı ayrı olmak üzere toplam 12 adet CSV veri dosyası halinde toplanmıştır.

Veri Birleştirme ve İlk Aşama Sorunlar

İlk olarak, aylık bazda elde edilen bu veriler doğrudan ilişkisel bir veritabanına aktarılmaksızın, Python programlama dili kullanılarak yazılan özel bir veri işleme betiği (script) aracılığıyla birleştirilmiştir. Bu işlem sonucunda oluşturulan bütünleştirilmiş veri kümesi, traffic_data.json adlı bir tekil JSON dosyasında saklanmıştır. Bu dosya, yıllık ortalama trafik yoğunluğuna ilişkin verileri içermekte olup, her bir veri noktası ilgili geohash konum bilgisi, zaman dilimi ve yoğunluk seviyesi gibi parametreleri içerecek biçimde yapılandırılmıştır.

Ancak, bu kapsamlı JSON dosyası web uygulamasına entegre edilmek istendiğinde, tarayıcı tabanlı veri işleme sürecinde ciddi performans problemleri ile karşılaşmıştır. JSON dosyasının boyutunun büyük olması, veri aktarımı sırasında sunucunun bellek yükünü aşarak çökmesine (server crash) neden olmuştur. Bu sorun, özellikle istemci (client) tarafında büyük veri kümeleriyle doğrudan işlem yapılmasının performans açısından verimli olmadığına işaret etmiştir.[19]

Geohash Temelli Görselleştirme Problemi

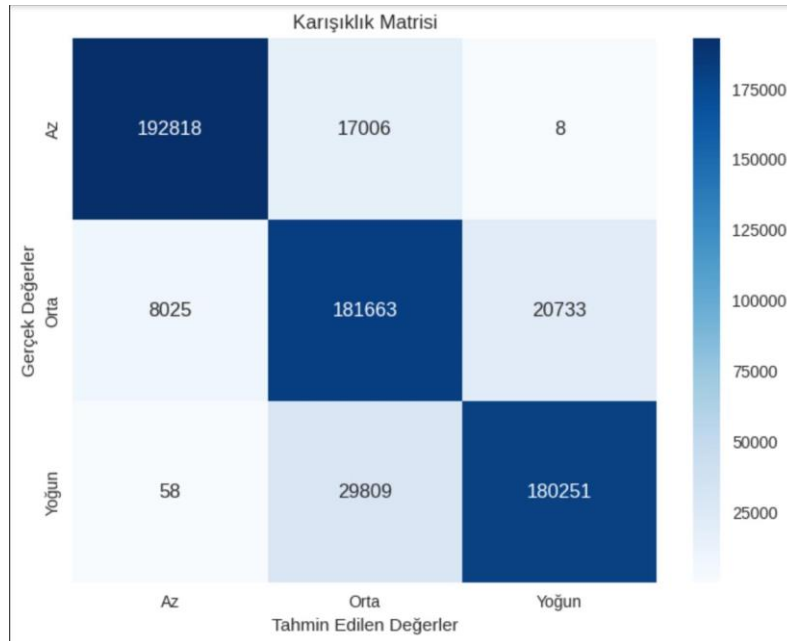
Proje kapsamında karşılaşılan bir diğer önemli zorluk ise, verilerin görsel olarak harita üzerine yansıtılması sırasında ortaya çıkmıştır. Kullanılan veri setinde her kayıt, konum bilgisi olarak geohash kodlaması ile tanımlanmıştır. Geohash algoritması, coğrafi koordinatları (enlem ve boylam) belirli boyutlardaki kutucuklara (grid) dönüştüren bir sistemdir. Ancak bu kodlama yöntemi, verilerin gerçek yol geometrilerini değil, yalnızca kare biçiminde alanları temsil etmesine neden olmuştur.

Bu durum, verilerin harita üzerinde gösterilmesi sırasında birtakım coğrafi uyumsuzluklara yol açmıştır. Örneğin, bir yol güzergâhı boyunca ilerleyen trafik yoğunluğu yerine, geohash kutularının rastgele ve yolu temsil etmeyen biçimde haritada görünmesi gibi görsel bozulmalar meydana gelmiştir. Bu problem, kullanıcıya doğru bir trafik yoğunluğu deneyimi sunmayı engellediğinden, projenin temel işleviyle çelişen bir duruma neden olmuştur.

Bu teknik zorlukların etkisiyle, başlangıçta opsiyonel bir özellik olarak planlanan yıllık ortalama trafik yoğunluğu haritasının projeye entegre edilmesi mümkün olmamıştır. Dolayısıyla bu özellik, mevcut sürümde proje kapsamından çıkarılmış; yalnızca bireysel rotalara yönelik anlık ve kısa vadeli trafik yoğunluğu tahminine odaklanılmıştır.

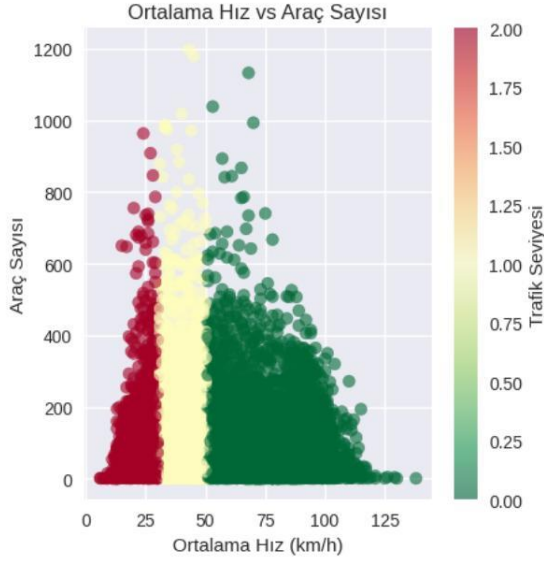
7.6.Sonuç ve Değerlendirme

Bu bölümde, geliştirilen yapay zekâ destekli trafik tahmin sisteminin başarı düzeyi hem nicel metriklerle hem de görsel analizlerle detaylandırılmıştır. Modelin sınıflandırma performansı, tahmin doğruluğu, trafik yoğunluğu ile hız ve araç sayısı ilişkisi, mekânsal dağılım ve zamana bağlı trafik örüntüleri üzerinden değerlendirilmiştir. Aşağıdaki grafik ve tablolar, sistemin farklı senaryolardaki çıktılarının görsel temsillerini sunmakta ve geliştirilen çözümün işlevselliğini somut biçimde ortaya koymaktadır.



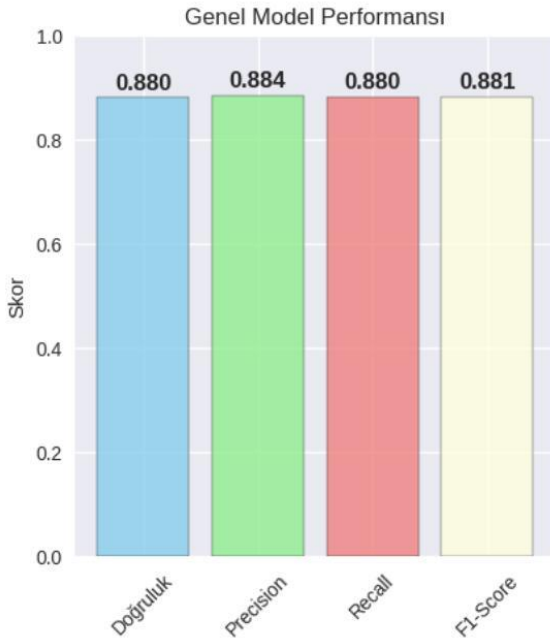
Şekil 7.6.1: Karışıklık Matrisi

Yukarıdaki karışıklık matrisi, tahmin edilen trafik seviyeleri ile gerçek etiketlerin karşılaştırmasını göstermektedir. Model, özellikle “Az” ve “Yoğun” sınıflarında oldukça yüksek doğrulukla tahmin yapabirmiştir. “Orta” seviye tahminlerinde ise sınıf karışıklıkları gözlemlenmiştir. Bu durum orta yoğunluk seviyesinin az ve yoğun sınıflarıyla daha fazla örtüşmesinden kaynaklanmaktadır.



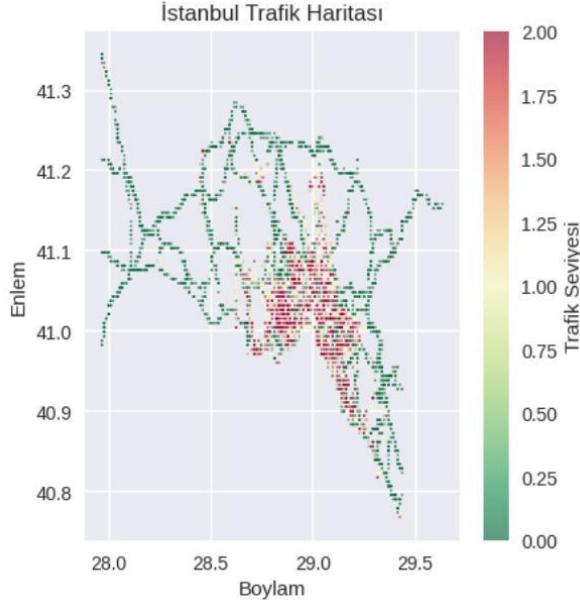
Şekil 7.6.2: Ortalama Hız ve Araç Sayısı

Bu grafik, ortalama hız ile araç sayısı arasındaki ilişkiyi trafik seviyelerine göre renklendirerek göstermektedir. Düşük hız ve yüksek araç sayısı genellikle yoğun trafiğe (kırmızı), yüksek hız ve düşük yoğunluk ise az trafik (yeşil) sınıfına karşılık gelmektedir. Grafik, modelin bu değişkenleri ayırt etmede neden etkili olduğunu görselleştirmektedir.



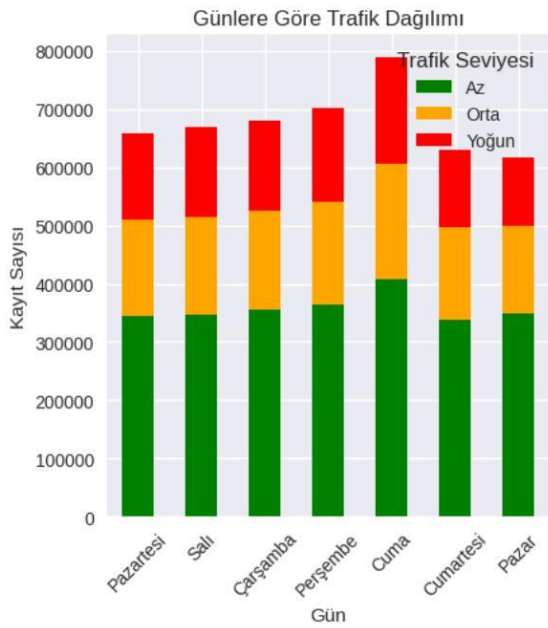
Şekil 7.6.3: Genel Model Performansı

Modelin doğruluk (%88.0), precision (%88.4), recall (%88.0) ve F1-score (%88.1) değerleri, tüm sınıflar üzerinde dengeli bir performansa sahip olduğunu göstermektedir. Bu skorlar, modelin eğitim ve test verisinde overfitting yapmadan genelleme yeteneği olduğunu kanıtlamaktadır.



Şekil 7.6.4: İstanbul Trafik Haritası

Grafikte İstanbul genelinde trafik seviyeleri coğrafi koordinatlara göre görselleştirilmiştir. Kırmızı noktalar yoğun trafik olan bölgeleri, yeşil noktalar ise akıcı trafiği göstermektedir. Özellikle merkezi ve sahil hattı gibi bölgelerde yoğun trafik gözlemlenmiştir. Bu tür coğrafi analizler, modelin mekânsal olarak da anlamlı sonuçlar ürettiğini göstermektedir.



Şekil 7.6.5: Günlere Göre Trafik Dağılımı

Bu çubuk grafik haftanın günlerine göre trafik yoğunluk seviyelerinin dağılımını göstermektedir. Cuma günü en yoğun trafik gözlemlenirken, hafta sonu (özellikle Pazar günü) az yoğunluk seviyeleri artmıştır. Bu durum, modelin haftalık kalıpları başarıyla öğrenebildiğini ve zaman bağımlı öngörüler yapabildiğini göstermektedir.

React ve Google Maps entegrasyonu sayesinde sistem, hem etkileşimli hem de fonksiyonel bir arayüz sunmaktadır. Kullanıcı, ileri tarihli bir zaman dilimi için kolayca sorgu yapabilmekte ve tahmin sonuçlarını anında, görsel olarak rota üzerinde görebilmektedir. Tasarımda odak noktası sadelik, erişilebilirlik ve performans olmuştur. Arayüzün modüler yapısı, ileride eklenebilecek yeni özellikler için de uygun zemin hazırlamaktadır.

8.GENEL DEĞERLENDİRME

Crowd Predictor projesi, veri toplama, modelleme, API geliştirme ve görselleştirme adımlarının tümünü başarıyla tamamlamış ve entegre bir trafik tahmin sistemi ortaya koymuştur. Model doğruluğu %89 olup, sistem kullanıcıya sade ve etkili bir arayüz sunmaktadır. Literatürdeki eksik yönleri (özelleştirilebilir zaman, ileri tarihli analiz) hedefleyerek somut bir katkı sunmaktadır.

Bu bitirme projesi kapsamında, trafik yoğunluğu tahmini yapabilen, yapay zekâ destekli web tabanlı bir sistemin temel yapısı oluşturulmuştur. Projenin bu aşamasında kullanıcı arayüzü tasarlanmış, giriş işlemi, rota ve zaman bilgisi alma ekranları geliştirilmiş, kullanıcıdan alınan veriler backend servisine gönderilerek, tahmin sonucuna göre görsel sunum yapılması sağlanmıştır. Tahminler şu anda test amaçlı rastgele olarak üretilmektedir ve bu yapı, daha sonra entegre edilecek gerçek bir makine öğrenmesi modeline hazır olacak şekilde tasarlanmıştır.

İlerleyen süreçte, Python ile eğitilen trafik tahmin modeli backend'e entegre edilecek ve kullanıcıdan alınan verilere göre gerçek zamanlı tahminler sunulacaktır. Ayrıca kullanıcı kayıt ve giriş sistemi geliştirilecek; favori rotalar, geçmiş aramalar ve harita tabanlı görselleştirme özellikleri sisteme eklenecektir. Bu geliştirmeler tamamlandığında, sistem hem kullanıcı deneyimi açısından daha zengin hale gelecek hem de trafik tahmini konusunda daha doğru ve anlamlı sonuçlar üretebilecektir. Projenin nihai hedefi, kullanıcıların geleceğe dönük ulaşım planlamalarını kolaylaştıracak akıllı ve kişiselleştirilmiş bir karar destek sistemi ortaya koymaktır.

REFERANSLAR

- [1] D. M. Levinson and K. Krizek, “Planning for Place and Plexus: Metropolitan Land Use and Transport,” Routledge, 2008.
- [2] Y. Yuan and M. van Eck, “Predicting Traffic with AI Models: A Survey,” *ACM Computing Surveys*, vol. 55, no. 3, 2023.
- [3] C. Antoniou et al., “Traffic Forecasting: Intelligent Transportation Systems,” *IEEE Transactions on ITS*, vol. 22, no. 1, pp. 1–15, 2021.
- [4] M. T. Asif et al., “Spatiotemporal patterns in large-scale traffic speed prediction,” *Transportation Research Part C*, vol. 59, 2015.
- [5] İ. Takak, H. Görmez, H. İ. Türkmen, and M. A. Güvensan, “Kısa, Orta ve Uzun Vadeli Trafik Akış Hızı Tahmini ve Görselleştirilme Aracı,” *Int. J. Adv. Eng. Pure Sci.*, vol. 33, no. 4, pp. 568–580, 2021. [Online]. Available: <https://dergipark.org.tr/tr/pub/jeps/issue/67265/883711>
- [6] F. Zhao et al., “LSTM Networks for Traffic Flow Forecasting: A Comparative Study,” *Neural Networks*, vol. 119, pp. 123–135, 2020.
- [7] M. Nas, “Yapay Sinir Ağları ile Trafik Yoğunluğu Tahmini,” M.S. thesis, Elektronik Müh. Programı, İstanbul Teknik Üniversitesi, İstanbul, Türkiye, 2024. [Online]. Available: <https://polen.itu.edu.tr/items/768514a6-6a86-4bb9-a0fb-3c138c6f06b4>
- [8] J. Chen and Y. Liu, “Deep Learning Based Prediction of Urban Traffic Flow: A Case Study of LSTM Networks,” *Sensors*, vol. 19, no. 14, 2019.
- [9] A. Utku, “Derin Öğrenme Tabanlı Trafik Yoğunluğu Tahmini: İstanbul İçin Bir Vaka Çalışması,” *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, vol. 11, pp. 1584–1598, 2023. [Online]. Available: <https://dergipark.org.tr/tr/pub/dubited/issue/79166/1139534>
- [10] T. Zhang, L. Zheng, and Q. Yang, “Weather Impacts on Urban Traffic Flow: A Data-Driven Approach,” *Transportation Research Part C*, vol. 93, 2018.
- [11] N. Taş and B. Sezen, “Yapay Sinir Ağları ile Trafik Yoğunluğu Tahmini,” *Afyon Kocatepe Üniversitesi Sosyal Bilimler Dergisi*, vol. 22, no. 4, pp. 1020–1034, 2020. [Online]. Available: <https://dergipark.org.tr/tr/pub/akusosbil/issue/59165/746349>
- [12] A. T. A. Rahman et al., “Spatiotemporal modeling of traffic congestion using GPS big data,” *Transportation Research Procedia*, vol. 25, pp. 820–831, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S235214652030524X>
- [13] Google Maps Platform Documentation, “Distance Matrix API Overview,” [Online]. Available: <https://developers.google.com/maps/documentation/distance-matrix/>
- [14] Z. Lv, H. Li, and S. Zhang, “Traffic flow prediction with big data: A deep learning approach,” *IEEE Access*, vol. 7, 2019.

- [15] N. Chawla et al., “SMOTE: Synthetic Minority Over-sampling Technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
- [16] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, pp. 5–32, 2001.
- [17] M. Grinberg, “Flask Web Development,” O’Reilly Media, 2018.
- [18] R. Banks, “Learning React: Functional Web Development with React and Redux,” O’Reilly, 2020.
- [19] H. V. Jagadish et al., “Big data and its technical challenges,” *Communications of the ACM*, vol. 57, no. 7, pp. 86–94, 2014.