Teknoloji Fakültesi

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

CROWDPREDICTOR

BİTİRME PROJESİ 2. ARA RAPORU

Bilgisayar Mühendisliği Bölümü

DANIŞMAN

Doç. Dr. Öğr. Üyesi BUKET DOĞAN

İSTANBUL, 2025

1 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* .....

4. *VERİ TOPLAMA* .....

5. *VERİ HAZIRLAMA VE MODEL EĞİTİMİ* .....

5.1 *Veri Birleştirme ve Ön İşleme* .....

5.2 *Veri Dengesizliği ve Dengeleme* .....

5.3 *Özellik Setinin Genişletilmesi* .....

5.4 *Model Eğitimi ve Kaydı* .....

5.5 *Metrik Karşılaştırması ve Grafik Oluşturma* .....

6. *MODEL ENTEGRASYONU* .....

6.1 *Flask Tabanlı API Yapısı* .....

6.2 *Flask Tercihinin Gerekçeleri* .....

6.3 *Uçtan Uca Entegrasyon Akışı* .....

6.4 *Geleceğe Yönelik Genişletme Önerileri* .....

7. *ARAYÜZ GELİŞTİRME* .....

7.1 Kullanıcı Giriş ve Kayıt Arayüzü .....

7.2 Veri Tabanı Altyapısı .....

7.3 Harita Entegrasyonu ve Kullanılan Teknolojiler .....

7.4 Harita Görüntüleme Ekranına Genel Bakış .....

7.5 Karşılaşılan Sorunlar .....

7.6 Sonuç ve Değerlendirme .....

8. *GENEL DEĞERLENDİRME* .....

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, Spring Boot 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. İlk aşamada test amaçlı rastgele tahminler sunulmakta, ancak altyapı gerçek verilerle çalışan bir modele geçişe hazır şekilde oluşturulmuştur. Tahmin sonuçları, kullanıcıya yoğunluk durumuna göre uygun görsellerle sunulmaktadır.

Proje ilerleyen aşamalarda kullanıcı kayıt sistemi, geçmiş aramalar, favori rotalar ve harita entegrasyonu gibi özelliklerle geliştirilecektir. Böylece 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

This project aims to develop an AI-supported, web-based system that allows users to receive traffic congestion predictions for a specific date and time. Traffic congestion is one of the most critical problems affecting daily life, especially in large cities. While existing traffic applications generally provide real-time data, this project focuses on predictive traffic information for future planning.

The system is designed to operate in an integrated manner using a user-friendly interface built with React.js, a backend server developed with Spring Boot, and a machine learning model implemented in Python. After logging in, users can input route details such as departure point, destination, and travel time to receive a traffic prediction. Initially, the system provides randomly generated predictions for testing purposes; however, the architecture is ready for integration with a real machine learning model. The prediction results are presented visually, using illustrations that correspond to traffic intensity levels.

In the future stages, the system will be enhanced with features such as user registration, search history tracking, favorite routes, and map integration. Ultimately, the project aims to deliver a smart and personalized traffic prediction system that helps users plan their travel more efficiently.

March, 2025

Students

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

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. 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.

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. Ancak günlük hayatın birçok alanında, önceden yapılacak trafik öngörüleri 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 (Görsel 1). 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 (Görsel 2), 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.

Görsel 1

Görsel 2

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. 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 [2]. Diğer yandan, yapay sinir ağlarına dayalı modellerde doğruluk oranları çok daha yüksek seviyelere ulaşmıştır. 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. Özellikle hibrit RBF-MLP yapısının çok değişkenli girişlerde daha kararlı sonuçlar verdiği gözlemlenmiştir [3]. Derin öğrenme tabanlı bir başka çalışmada ise, LSTM mimarisi ile Mahmutbey kavşağında araç sayısı ve ortalama hız tahmini yapılmış, %0.9 R^2 değeriyle yüksek doğruluk elde edilmiştir. Bu çalışmada ayrıca RF, SVM ve CNN gibi modellerle karşılaştırmalı analiz yapılmış ve LSTM'in tahmin gücü öne çıkmıştır [4]. Literatürde ayrıca trafik tahmininde hava durumu, sıcaklık, nem ve rüzgâr gibi çevresel faktörlerin etkisinin de önemli olduğu vurgulanmıştır. Örneğin, rüzgar ve sıcaklığın hız üzerinde negatif etkileri olduğu; bulutlu, yağışlı hava ve mesai saatlerinin tahmin doğruluğunu etkilediği görülmüştür [5]. 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 [1]. 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 karşılaşılmıştır. Google Maps API, yalnızca gerçek zamanlı trafik verisi sunduğundan, belirli bir geleceğe yönelik trafik öngörüsüne izin vermemektedir. 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.

Değişken Adı	Açıklama
DATE_TIME	Verinin kaydedildiği tarih ve saat bilgisi (saatlik çözünürlük)
LATITUDE & LONGITUDE	Sensör konumunu tanımlayan coğrafi koordinatlar
MINIMUM_SPEED	İlgili saatte ölçülen minimum hız (km/s)
MAXIMUM_SPEED	Aynı saat diliminde ölçülen maksimum hız (km/s)
AVERAGE_SPEED	Ortalama hız değeri (km/s)
NUMBER_OF_VEHICLES	Kayıt anındaki toplam araç sayısı

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.

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 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:

```
Trafik Yoğunluğu Tahmin Performansı:
              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
```

Bu tabloda görüldüğü üzere, sınıf 0 (Az Yoğun) dışındaki trafik seviyeleri (1 ve 2) neredeyse hiç tanınamamaktadır (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. 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.

Dengelenmiş Trafik Yoğunluğu Tahmin Performansı:

	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

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

5.3 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:

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

```python
# 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"]
```

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
```

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 oluşturmasına yardımcı olduğunu göstermektedir.

5.4 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. 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.

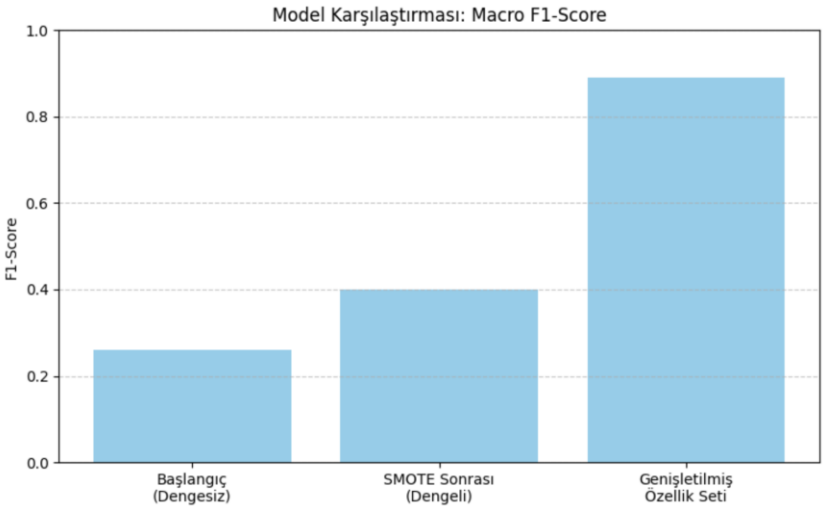
5.5 Metrik Karşılaştırması ve Grafik Oluşturma

Karşılaştırmalı değerlendirme için üç ayrı model çıktısı metriksel olarak karşılaştırılabilir:

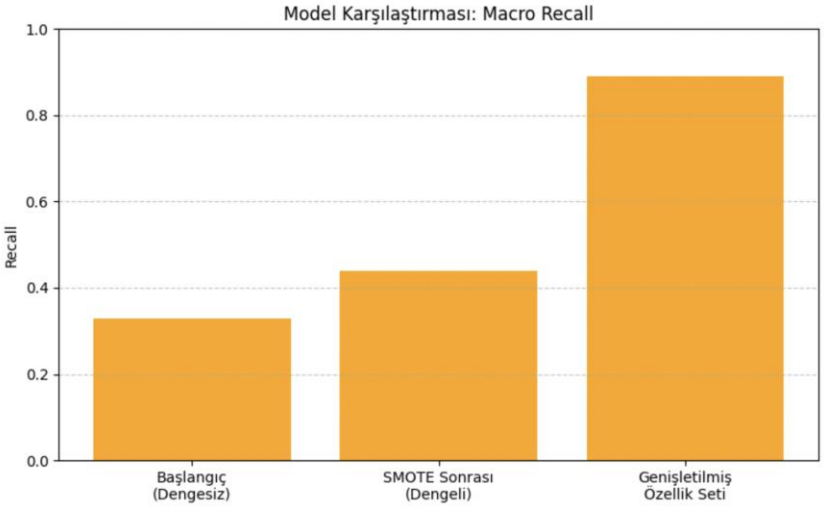
Model Yapısı	Accuracy	Macro F1	Sınıf 2 F1
Temel özellik seti	0.64	0.26	0.00
Dengeleme sonrası	0.44	0.40	0.55
Genişletilmiş özellik seti	0.89	0.89	0.90

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:



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.



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. 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:** Eğitimli 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.
- **Autocomplete** ile başlangıç/varış noktalarının otomatik önerilmesi
- **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

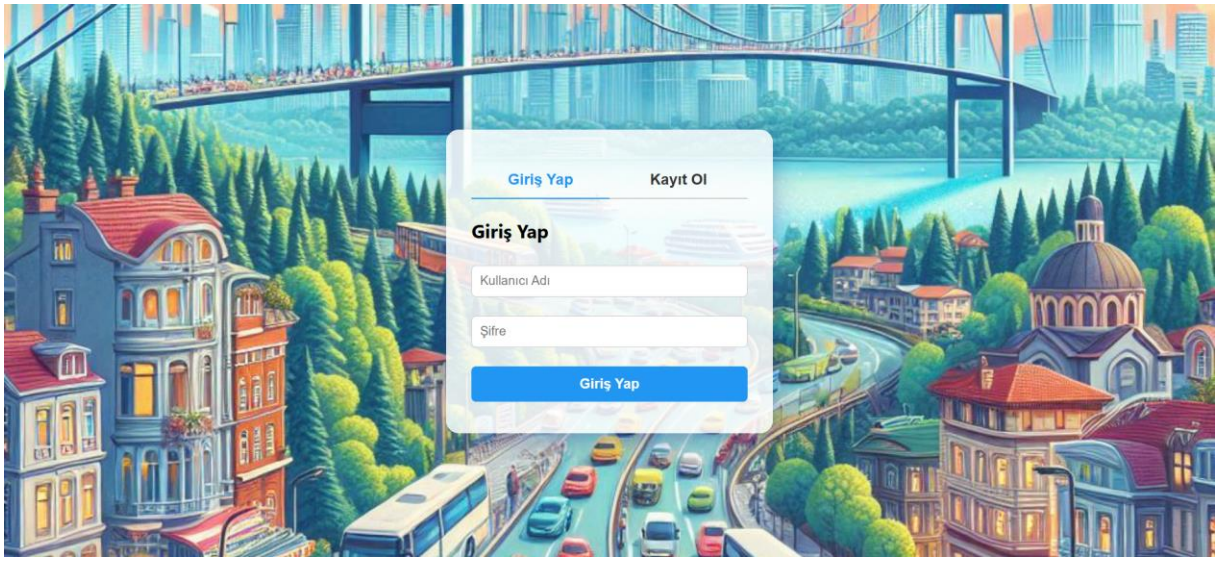
7. ARAYÜZ GELİŞTİRME

Crowd Predictor projesinde, kullanıcıdan alınan rota ve zaman bilgisine dayalı olarak trafik yoğunluğu tahmini yapılmakta ve bu tahmin, Google Maps üzerinde görsel olarak sunulmaktadır. Bu işlem zinciri yalnızca modelin doğruluğu kadar, tahmin sonucunun kullanıcıya nasıl aktarıldığıyla da doğrudan ilişkilidir. Bu nedenle, kullanıcı deneyimini

önceleyen, dinamik, sade ve etkileşimli bir arayüz tasarımı hedeflenmiştir. Projede ön yüz geliştirmesi için **React.js** teknolojisi kullanılmış; harita entegrasyonu için ise **Google Maps JavaScript API** ve **Places API** tercih edilmiştir.

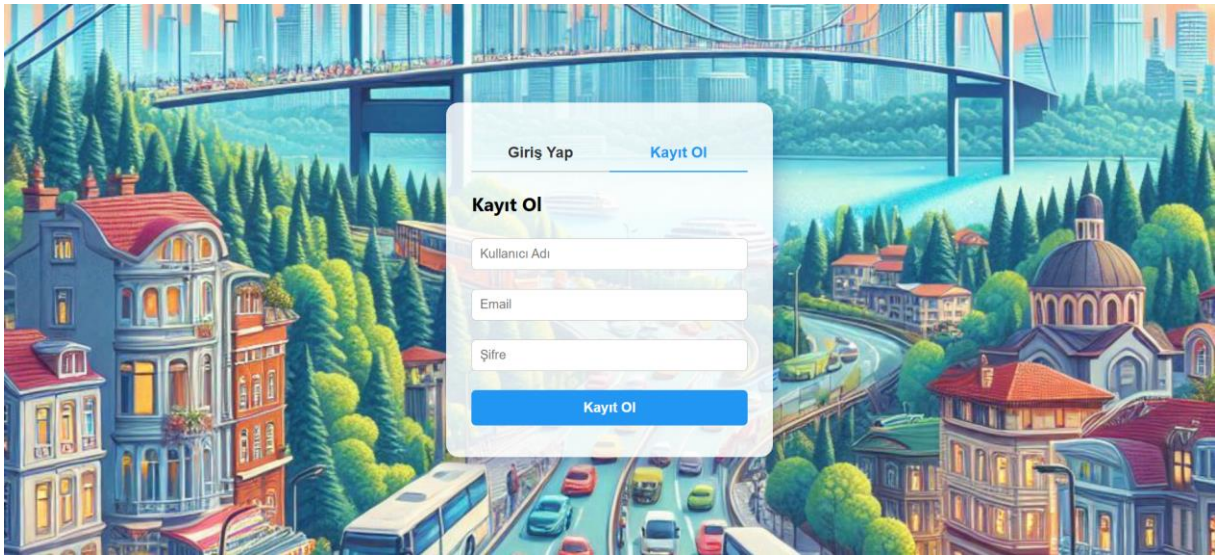
7.1 Kullanıcı Giriş ve Kayıt Arayüzü

React uygulaması üç temel ekrandan oluşmaktadır: giriş, kayıt ve harita görüntüleme. Giriş ekranında modern ve kullanıcı odaklı "Tabbed Login/Register UI" tasarımı kullanılmıştır. Bu yapı sayesinde "Giriş Yap" ve "Kayıt Ol" işlemleri sekmeler aracılığıyla tek sayfa üzerinden yönetilmekte; sayfa yenilenmesi olmadan daha hızlı bir kullanıcı deneyimi sağlanmaktadır.



Kullanıcı Giriş Özellikleri:

- Kullanıcı adı veya şifre alanlarından birinin eksik olması durumunda uyarı mesajları gösterilir.
- Kullanıcı adı sistemde kayıtlı değilse "Kullanıcı bulunamadı." hatası verilir.
- Şifre hatalıysa "Şifre hatalı." uyarısı çıkar.
- Giriş başarılı olduğunda kullanıcı harita görüntüleme ekranına yönlendirilir.

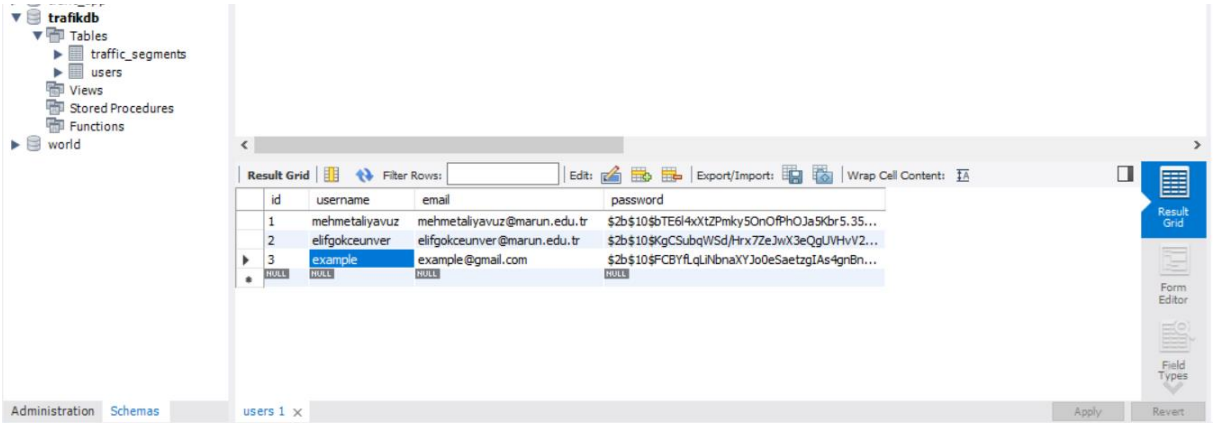


Kayıt Olma Özellikleri:

- Üç zorunlu alan: kullanıcı adı, email ve şifre
- Email alanı "@" içerip içermediğine göre kontrol edilir
- Mevcut kullanıcı adı veya email ile kayıt yapılmak istenirse sistem uyarı verir
- Kayıtlı bilgiler **MySQL veritabanına** kaydedilir

7.2 Veri Tabanı Altyapısı

Frontend tarafında kullanıcıdan alınan veriler, **MySQL veritabanına** kaydedilmekte ve şifreler bcryptjs kütüphanesi ile hashlenerek güvenli şekilde saklanmaktadır. Kullanıcı bilgileri "trafficdb" isimli veritabanındaki "users" tablosunda tutulmaktadır.



Bu yapı, sistem yeniden başlatılsa bile kullanıcı bilgilerinin korunmasını sağlar. Ayrıca bcrypt algoritmasının kasıtlı olarak yavaş çalışması, brute-force saldırılarına karşı sistemi daha dirençli hale getirir.

7.3 Harita Entegrasyonu ve Kullanılan Teknolojiler

Kullanıcının giriş yapmasının ardından yönlendirildiği harita ekranında, **Google Maps API** servisleri ile dinamik bir trafik yoğunluğu görselleştirme deneyimi sunulmaktadır. Kullanıcı arayüzünde şu teknolojiler kullanılmıştır:

- **Maps JavaScript API:** Haritanın web sayfasına gömülmesi ve interaktif marker/rota çizimi için.
- **Places API:** Otomatik adres tamamlama (autocomplete) ve yer önerileri için.
- **Autocomplete:** Kullanıcı adres girdilerinde hata payını azaltmak ve hızlı seçim sağlamak için.
- **React-axios:** Kullanıcıdan alınan rota bilgilerini Flask API'ye göndermek ve tahmin sonucunu almak için.

7.4 Harita Görüntüleme Ekranına Genel Bakış

Özellikler:

- Harita varsayılan olarak İstanbul üzerine konumlandırılmıştır.
- Sol üst kısımda kullanıcı harita modunu "harita" veya "uydu" olarak seçebilir.
- "Arazi" veya "etiketler" modları aktifleştirilerek farklı görünüm seçenekleri sunulmaktadır.
- Başlangıç ve varış noktası inputları ile kullanıcı rota oluşturur.
- Rota üzerine **marker** yerleştirilir; Google Maps üzerinden tahmin sonucu gelen sınıfa göre rota şu şekilde renklendirilir:
    - **Yeşil:** Az Yoğun (0)
    - **Sarı:** Orta Yoğun (1)
    - **Kırmızı:** Yoğun (2)
- Harita tam ekran moduna alınabilir. Bu mod, dikkat dağıtıcı unsurları ortadan kaldırarak kullanıcıya daha geniş görüş alanı ve daha yüksek erişilebilirlik sunar.

7.5 Karşılaşılan Sorunlar

Proje kapsamında, yalnızca ileri tarihli trafik tahminleri değil, aynı zamanda 12 aylık veriye dayalı **yıllık ortalama trafik yoğunluğu haritası** gösterimi de planlanmıştır. Ancak bu özellik aşağıdaki nedenlerle kapsam dışı bırakılmıştır:

- İBB'den çekilen saatlik trafik verileri işlenip traffic_data.json dosyasına kaydedilmiştir. Ancak bu dosya tarayıcıda gösterilmek istendiğinde **JSON boyutunun çok büyük olması** sunucunun çökmesine neden olmuştur.
- JSON yerine MySQL veritabanında bu veriler tutulmuş olsa da, görselleştirme sırasında **geohash bölgelerinin coğrafi yollara tam karşılık gelmemesi** sebebiyle harita üzerinde anlamsız bloklar oluşmuştur.
- Bu nedenle, yıllık ortalama yoğunluk haritası uygulamadan çıkarılmıştır.

7.6 Sonuç ve Değerlendirme

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.