ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Vize Proje Raporu

BLM3522- BULUT BİLİŞİM VE UYGULAMALARI

Hazırlayanlar:

- Eda Nur Arslan – 22290210

- Ömer Doğan – 22290528

Çift Katmanlı Web Uygulaması Destek Talep Sistemi:

Github: https://github.com/BLM3522-Bulut-Bilisim-Projeleri/Cift-Katmanl-Web-Uygulamas-Web-API-Frontend

https://www.youtube.com/watch?v=EyBI3W_OHvs&t=6s

Akıllı Veri Analitiği ve Makine Öğrenmesi Uygulaması:

Github: https://github.com/BLM3522-Bulut-Bilisim-Projeleri/Akill-Veri-Analitigi-ve-Makine-Ogrenmesi-Uygulamasi-

https://www.youtube.com/watch?v=6Xe0H6ozY-s

Çift Katmanlı Web Uygulaması (Web API + Frontend)

Projenin Amacı:
Bu proje, şirket çalışanlarının yaşadıkları teknik sorunları bilgi işlem departmanına iletebilecekleri, taleplerin durumlarını takip edebilecekleri, öncelik sırasına göre yönetebilecekleri ve bilgi işlem ekibinin çözüm sürecini şeffaf bir şekilde paylaşabileceği bir destek talep sisteminin geliştirilmesini amaçlar.

Kullanılan Teknolojiler:

- **Backend:** Java (Spring Boot)
- **Frontend:** React.js (Vite yapılandırması ile)
- **Veritabanı:** PostgreSQL
- **Bulut Servisi:** AWS EC2 (Amazon Linux)
- **Bağlantı Şekli:** RESTful API

Proje Yapısı ve Katmanlar:

1. **Frontend (React)**
    - Kullanıcıların destek talebi oluşturabileceği, mevcut talepleri listeleyip filtreleyebileceği ve durumlarını güncelleyebileceği bir arayüz sağlar.
    - Modal pencere ile talepler düzenlenebilir.
    - Axios ile backend'e HTTP istekleri gönderilir.
2. **Backend (Spring Boot)**
    - RESTful API yapısında çalışır.
    - CRUD (Create, Read, Update, Delete) operasyonlarını yapabilecek /api/tickets uç noktasını sunar.
    - Veritabanında Ticket adında bir tablo oluşturarak, taleplerin kaydını tutar.
    - PostgreSQL kullanılmıştır. Veritabanı otomatik tabloları Hibernate üzerinden oluşturur.
    -

Proje Kurulumu - Adım Adım:

1. Geliştirme Ortamı

- Java 17
- Node.js + npm
- IntelliJ IDEA / VS Code
- PostgreSQL 17
- AWS EC2 (Amazon Linux 2023)

2. Spring Boot Projesi

- start.spring.io üzerinden proje olusturuldu (Dependency: Spring Web, Spring Data JPA, PostgreSQL Driver, Lombok)
- Ticket entity'si, TicketRepository, TicketController dosyaları hazırlandı.
- application.properties PostgreSQL bilgileri ile dolduruldu:

spring.datasource.url=jdbc:postgresql://localhost:5432/support_ticket_db

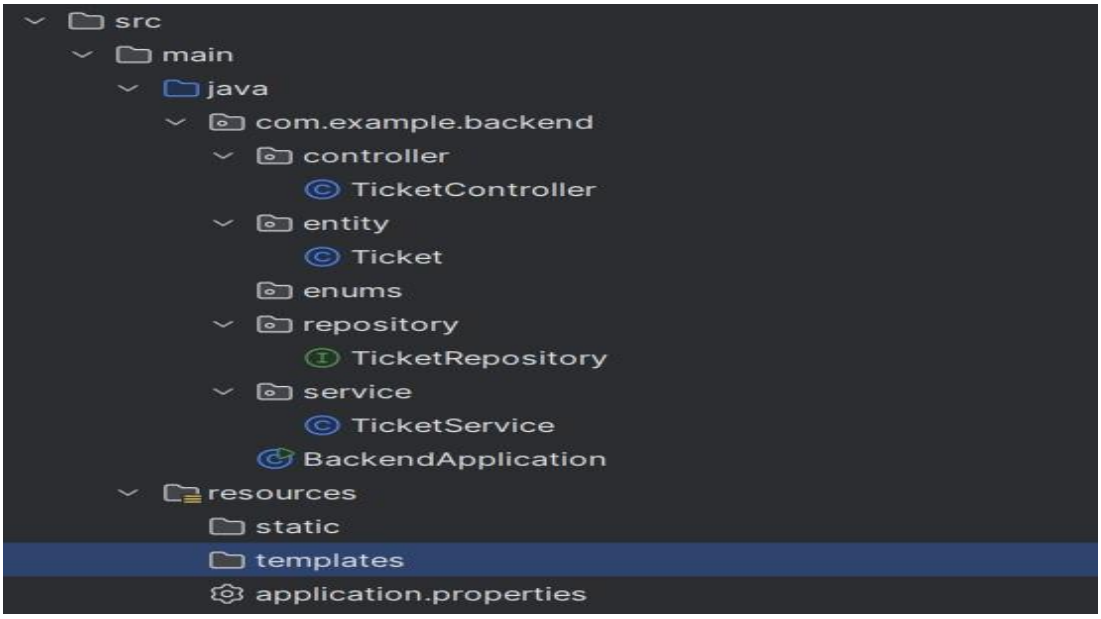
spring.datasource.username=myuser

spring.datasource.password=mypassword

spring.jpa.hibernate.ddl-auto=create

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect



3. React Projesi

- npm create vite@latest frontend --template react

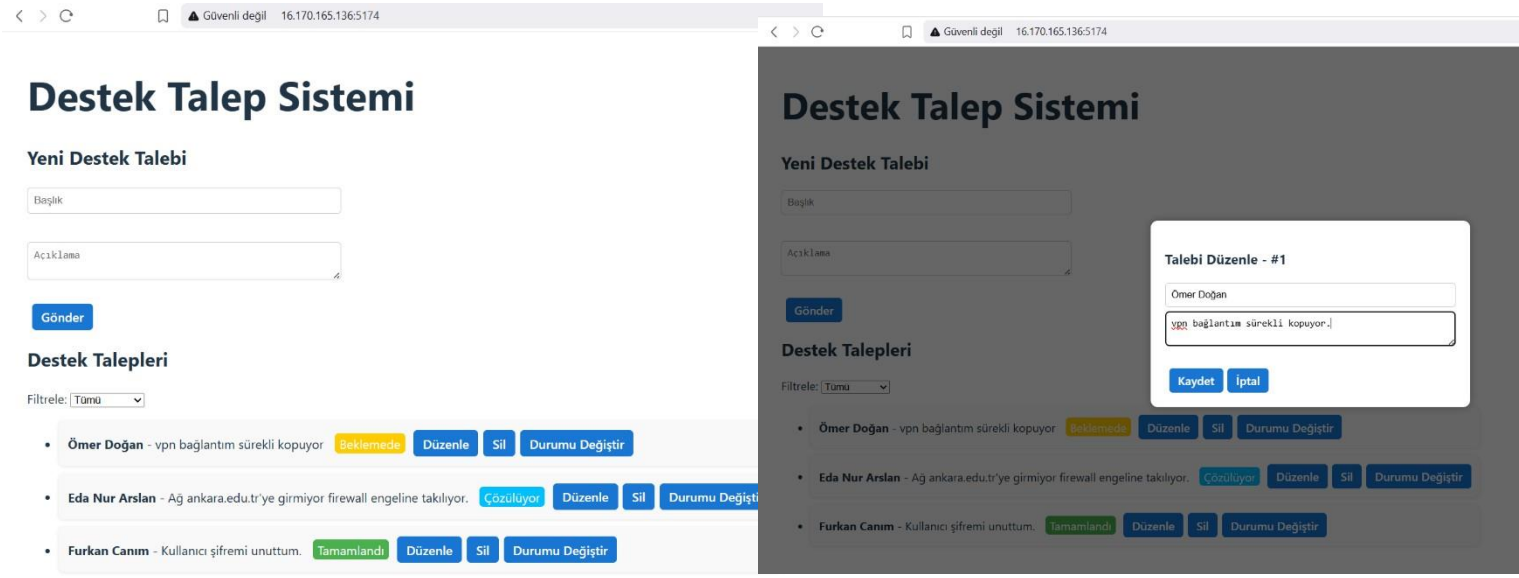
- Tailwind kurulumu: npm install -D tailwindcss postcss autoprefixer
- .env dosyası eklendi:

VITE_API_URL=http://<EC2-IP>:8080

- Axios istekleri bu adres üzerinden yapılandırıldı:

const API_URL = import.meta.env.VITE_API_URL;

- TicketList ve TicketForm bileşenleri yazıldı.
- Modal ve filtreleme destekleri eklendi.



4. AWS Kurulum ve Yayın

- EC2 instance oluşturuldu (Amazon Linux 2023)
- Java, Node.js, Git, PostgreSQL yüklendi
- PostgreSQL veritabanı init edildi, PostgreSQL dinleme portu ayarlandı
- React uygulaması vite.config.js ile host: true ve port: 5174 ayarlandı

server: {

host: true,

port: 5174

}

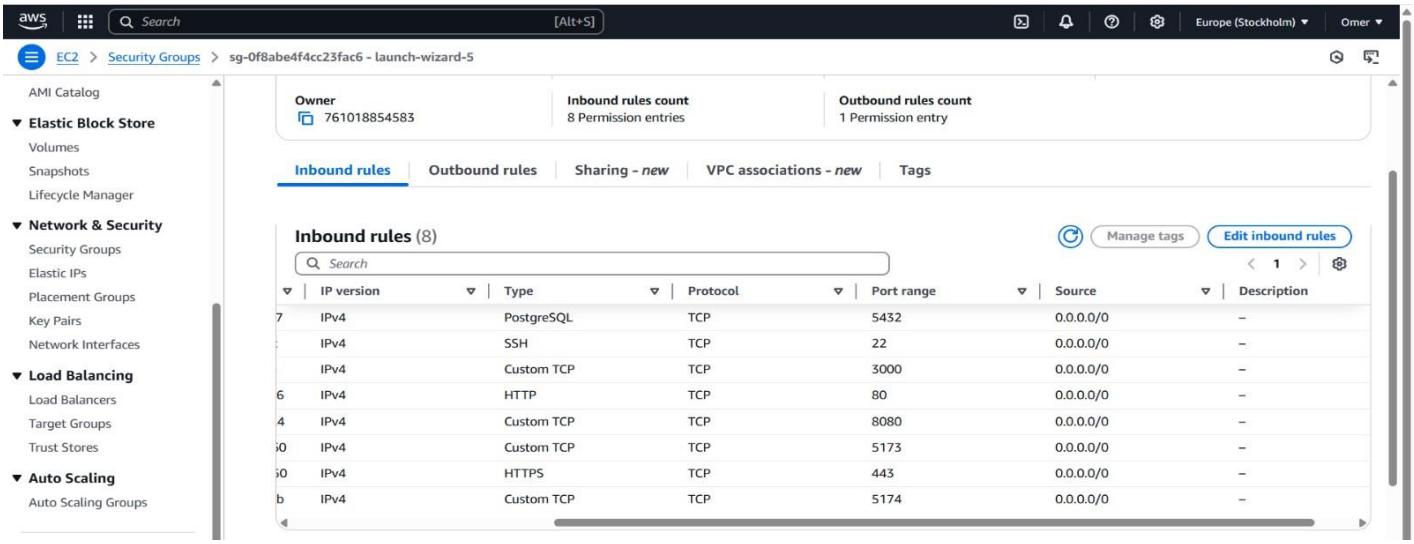
- Spring Boot uygulaması çalıştırıldı:

./mvnw spring-boot:run

- Frontend başlatıldı:

npm run dev

- AWS Security Group inbound kurallarına 8080 ve 5174 TCP portları eklendi.



Otomatik Sunucu Başlatma Komut Dosyası (baslat.sh)

Projede sunucunun manuel olarak her bileşen için ayrı ayrı başlatılmasını önlemek ve dağıtımı kolaylaştırmak amacıyla baslat.sh adında bir Bash scripti hazırlanmıştır.

Bu script aşağıdaki adımları sırasıyla gerçekleştirmektedir:

1. **PostgreSQL Veritabanı Servisini Başlatır:**

sudo -u postgres /usr/bin/pg_ctl -D /var/lib/pgsql/17/data -l /var/lib/pgsql/17/logfile start

PostgreSQL 17 servisini başlatır ve log dosyasını belirlenen konuma yönlendirir.

2. **Spring Boot Backend Servisini Başlatır:**

cd ~/Cift-Katmanl-Web-Uygulamas-Web-API-Frontend/backend

./mvnw spring-boot:run &

Backend dizinine gidip Spring Boot uygulamasını arka planda başlatır.

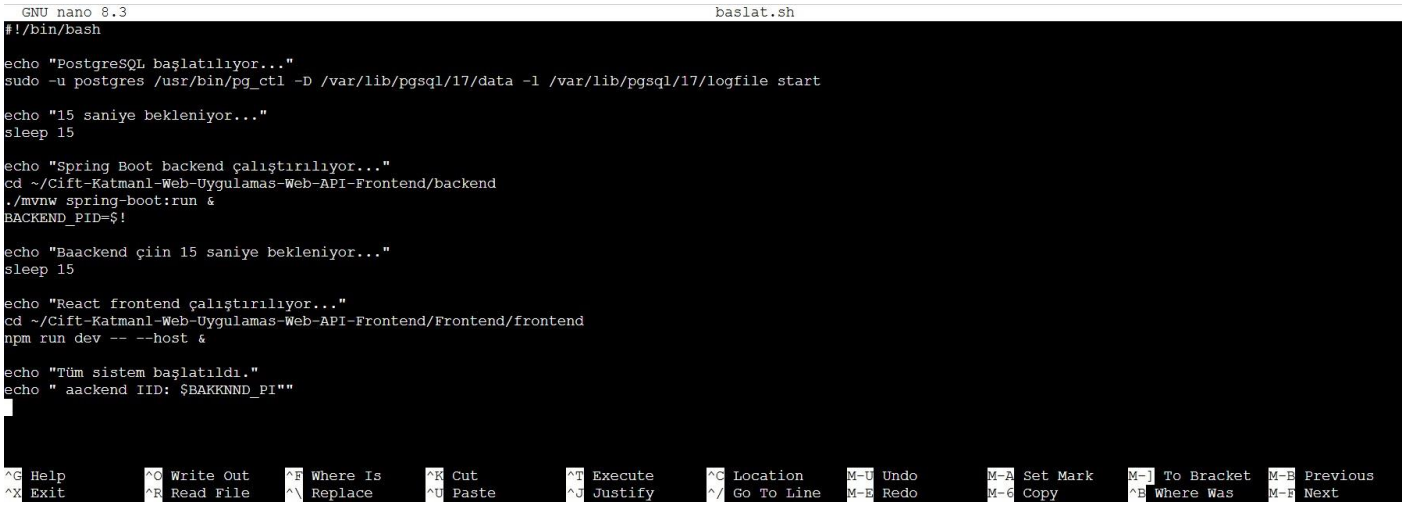
3. **React Frontend Servisini Başlatır:**

cd ~/Cift-Katmanl-Web-Uygulamas-Web-API-Frontend/Frontend/frontend

npm run dev -----host &

Frontend dizinine geçerek uygulamayı ağ üzerinden erişilebilir şekilde (--host) başlatır.

Aşağıdaki görsel baslat.sh scriptinin içerik yapısını göstermektedir. Bu script sayesinde tüm sistem tek komutla ayağa kaldırılabilir.



RESTful API ve Entegrasyon Açıklaması:

- RESTful API, istemciden gelen HTTP isteklerini (GET, POST, PUT, DELETE) karşılayarak belirli URL desenlerine cevap verir.
- @RestController anotasyonlu sınıf bu API'yi expose eder.
- React tarafından Axios ile yapılan istekler bu endpoint'lere gider.
- API ve frontend arasındaki bağlantı environment dosyası aracılığıyla sağlanır.

Sonuç ve Kazanımlar:

- Kullanıcı dostu bir destek talep sistemi geliştirildi.
- Spring Boot ile REST API kurulumu ve PostgreSQL ile entegrasyonu gerçekleştirildi.
- React ile filtreleme, durum güncelleme ve modal bileşenli UI geliştirildi.
- AWS uzerinden yayınlanarak frontend ve backend başarıyla entegre çalıştırıldı.

Akıllı Veri Analitiği ve Makine Öğrenmesi Uygulaması

Projenin Amacı:

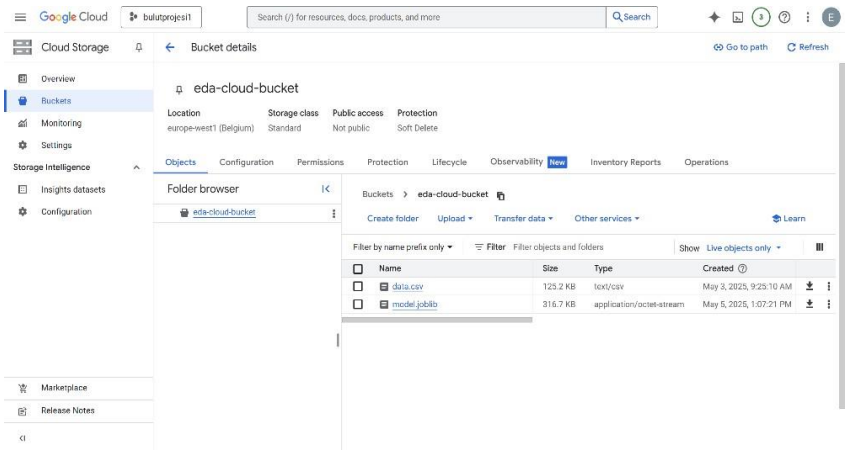
Bu projenin amacı, bir makine öğrenmesi modelini geliştirerek sağlık verileri üzerinden tahminlerde bulunmak ve bu modeli Google Cloud üzerinde dağıtarak gerçek zamanlı API üzerinden tahmin yapılmasını sağlamaktır.

Kullanılan Teknolojiler:

- **Backend**: Python (FastAPI)
- **Makine Öğrenmesi Kütüphaneleri**: Scikit-learn, Joblib
- **Veri Depolama**: Google Cloud Storage (CSV + Model)
- **Bulut Platformu**: Google Cloud Platform
    - Cloud Run Functions (model API dağıtımı)
    - Cloud Storage (Bucket yapısı için)
    - Cloud Build (Docker image oluşturma)
    - Vertex AI (Jupyter Lab üzerinden model eğitme)
    - Artifact Registry (Container image saklama)
    - BigQuery (Veri saklama)
    - Cloud Shell (terminal)
- **Diğer**: Docker, Requests (client test için)

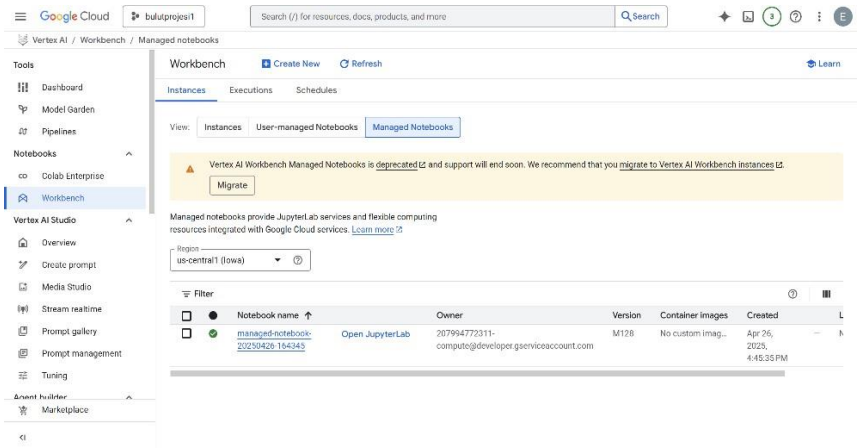
Adımlar:

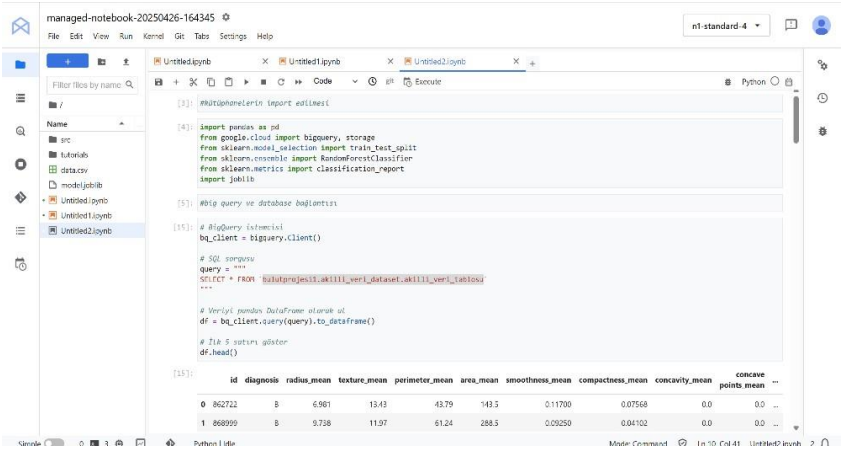
1. **Veri Yükleme ve Ön İşleme**
    - **Wisconsin Breast Cancer (Meme Kanseri Veri seti)** veri seti data.csv dosyası olarak Cloud Storage'a bucket ile yüklendi.
    - Verideki kategorik hedef (diagnosis) dönüştürüldü.



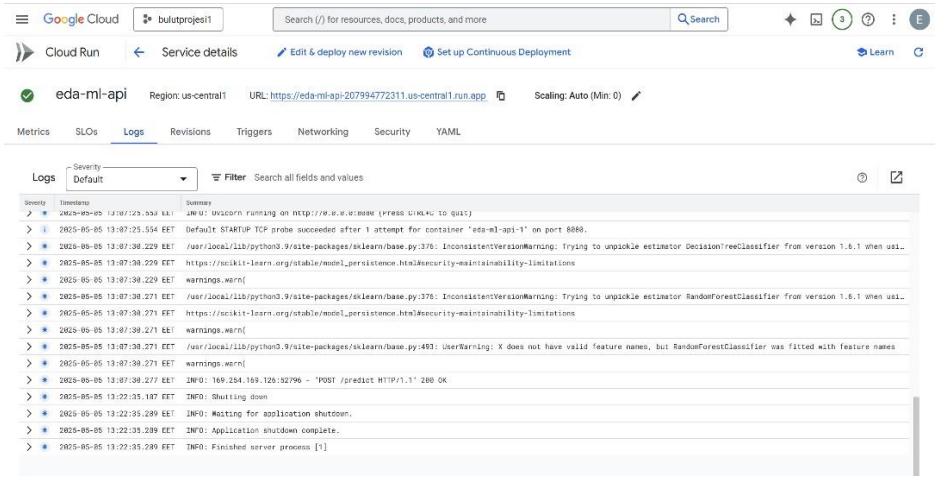
2. Model Geliştirme

- Scikit-learn ile Random Forest sınıflandırma modeli eğitildi.
- Model, joblib ile model.joblib dosyasına kaydedildi ve bukcet'a yüklendi.
- Model, feature_names_in_ ile birlikte toplam **32 özelliğe** göre tahmin yapmaktadır.

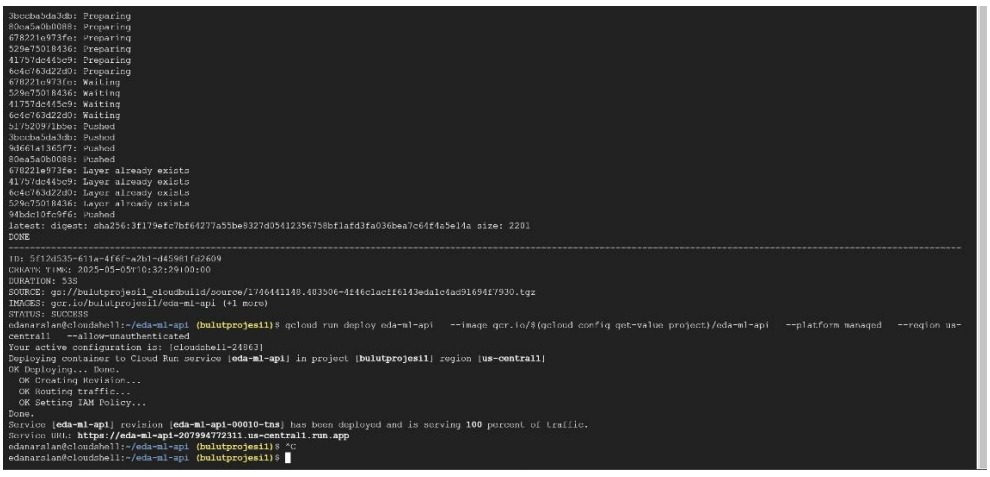


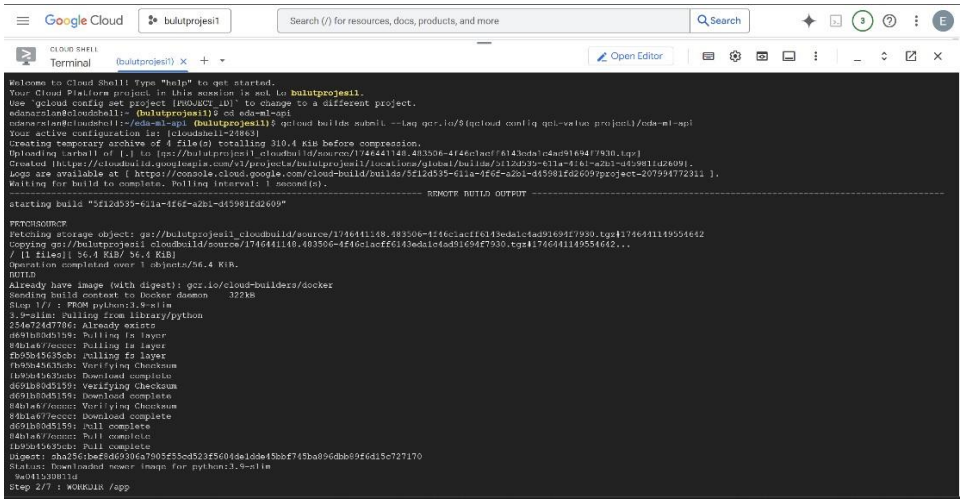


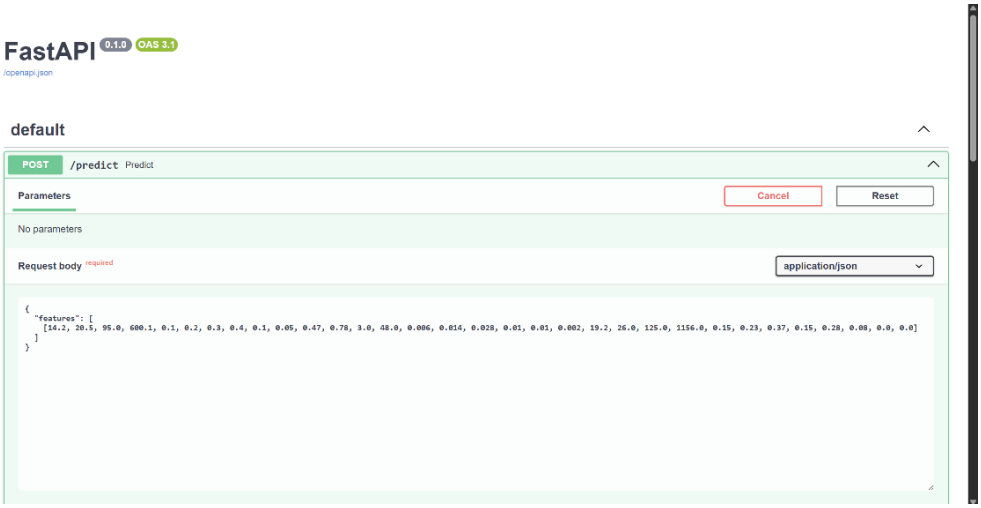
3. **Modelin Dockerlaştırılması**
    - Dockerfile, requirements.txt, main.py ve model.joblib dosyaları kullanılarak container image oluşturuldu.
    - gcloud builds submit ile image oluşturulup Container Registry'ye yüklendi.
4. **Cloud Run Dağıtımı**
    - "gcloud run deploy" komutu ile uygulama çalıştırıldı.
    - Modelin çalıştığı API son noktası: https://eda-ml-api-207994772311.us-central1.run.app/predict

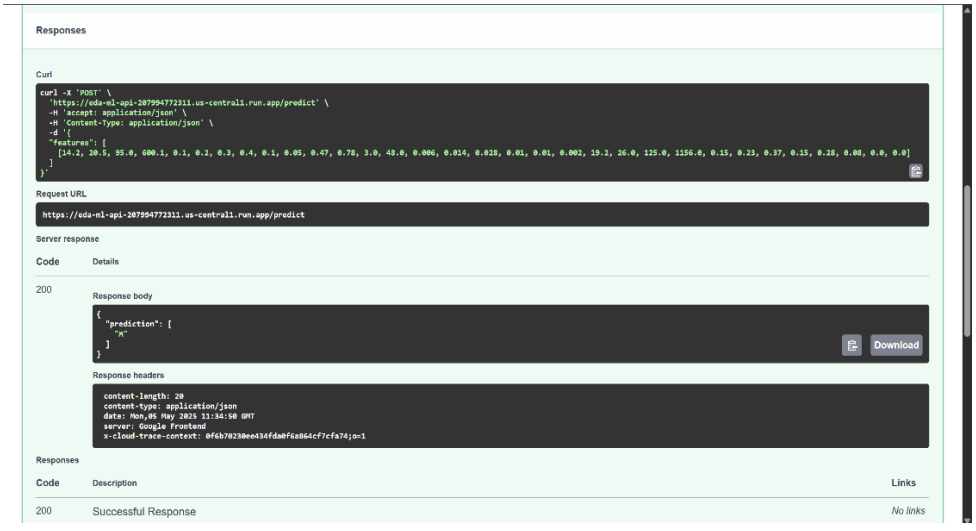


5. **Modelin Test Edilmesi**
    - API, FastAPI ile /predict endpoint'inde POST isteklerini kabul edecek şekilde yapılandırıldı.
    - Örnek veri gönderilerek doğru sonuç (örneğin 'M') alındı ve en yakın veri satırıyla eşleştiği doğrulandı.









Elde Edilen Çıktılar

- Makine öğrenmesi modeli geliştirildi.
- Google Cloud üzerinde model dağıtımı başarıyla yapıldı.

- API aracılığıyla tahminler alındı ve test edildi.
- Veriden bilgi çıkarma adımı en yakın veri noktasının karşılaştırılması ile gerçekleştirildi.

Sonuçlar:

Bu makine öğrenmesi projesi, Google Cloud platformu üzerinden geliştirilip dağıtıldı. Uygulama, hem modelleme hem de modern dağıtım tekniklerini kullanarak gerçek dünya senaryolarına uygun bir çözüm sunmaktadır. Cloud servisine tamamiyle uyumlu olacak şekilde geliştirilmiş ve buna uygun olarak entegrasyonu yapılmıştır.