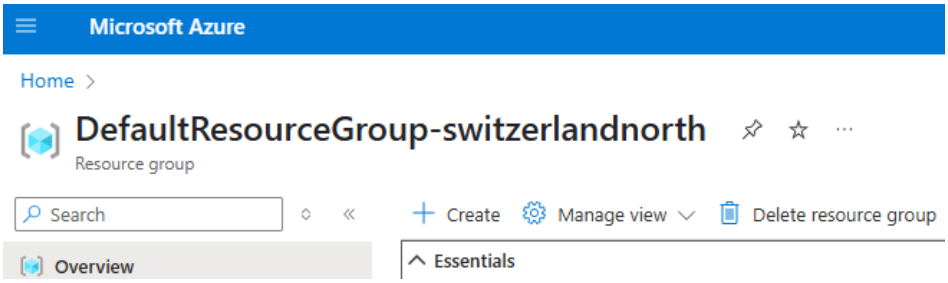
Proje: Diyabet Tahmin Uygulaması

Projenin Amacı: Bu projenin amacı, kullanıcıdan alınan sağlık verilerine göre diyabet hastalığına yakalanma riskini tahmin eden bir uygulama geliştirmektir. Model, Microsoft Azure üzerinde eğitilmiş ve deploy edilmiştir. Uygulama, kullanıcı verilerini bu modele göndererek gerçek zamanlı tahminler sunar.

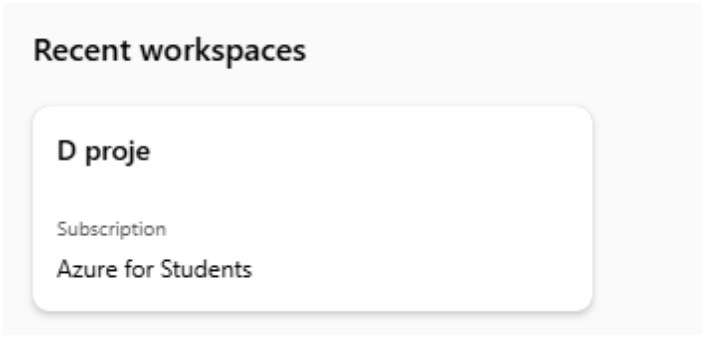
GELİŞTİRME SÜRECİ

Azure

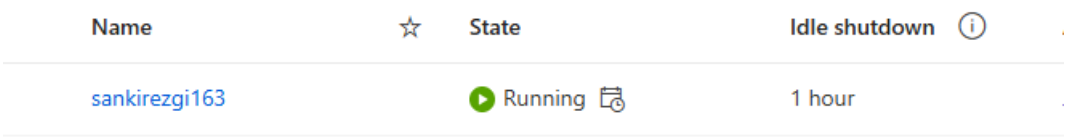
Microsoft Azure hesabı açıldı. Ana ekrandaki Azure Services kısmından "Create a resource" denilerek resource group oluşturuldu.



Azure AI Machine Learning Studio'ya girilip Azure Portal bilgileri ile giriş yapıldı. Burada "Create Workspace" diyerek çalışma alanı oluşturuyoruz. Oluştururken "Resource Group" seçeneğini daha önce oluşturduğumuz "DefaultResourceGroup-switzerlandnorth" olarak seçeceğiz.



Oluşturduğumuz Workspace'e giriş yapıyoruz. Manage kısmından Compute' a tıklayarak sanal makine oluşturuyoruz.



Ardından Assets kısmından Data'ya giriyoruz. Create dataset diyerek diabetes.csv dosyamızı buraya yüklüyoruz.

DiabetesData	1	workspaceblobstore	Jun 27, 2025 5:32 PM

Authoring kısmından Notebook açıyoruz. Klasör oluşturuyoruz. Bu dosyanın içerisine Python dosyası oluşturuyoruz. Bu Python dosyasında Data kısmında oluşturduğumuz veri setine bağlanarak makine öğrenmesi gerçekleştireceğiz.

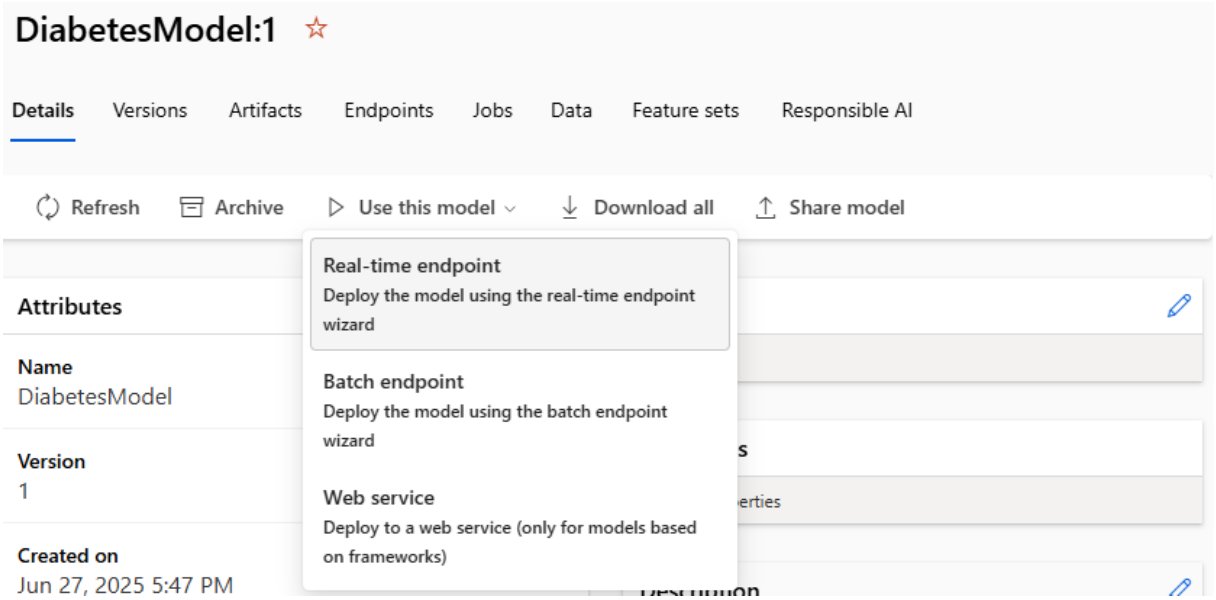
Assets >Jobs kısmından erişiyoruz.

Display name (1 visualized)	Parent job name	Status	Created on ↓
joyful_truck_vy10rxw7		Completed	Jun 27, 2025 5:50 PM

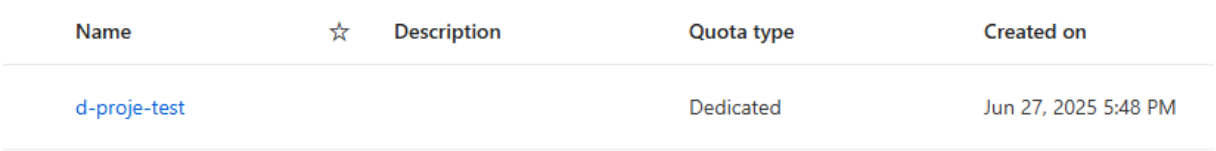
Aynı sayfadan Register model kısmından modelimizi oluşturuyoruz. Oluşturduğumuz Model Assets>Models kısmında gözüküyor.

Name	☆	Version	Type	Source
DiabetesModel		1	MLFLOW	This workspace

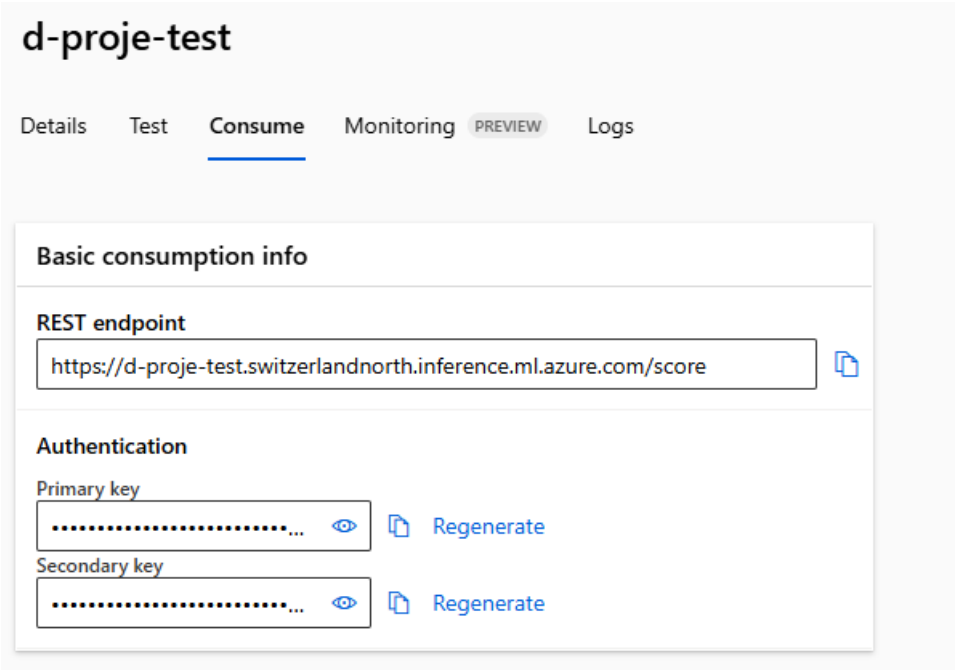
Bu modele tıklayıp Real-time endpoint seçeneğine tıklıyoruz ve endpoint oluşturuyoruz.



Oluşturduğumuz endpoint'i Assets>Endpoints kısmından görebiliriz.

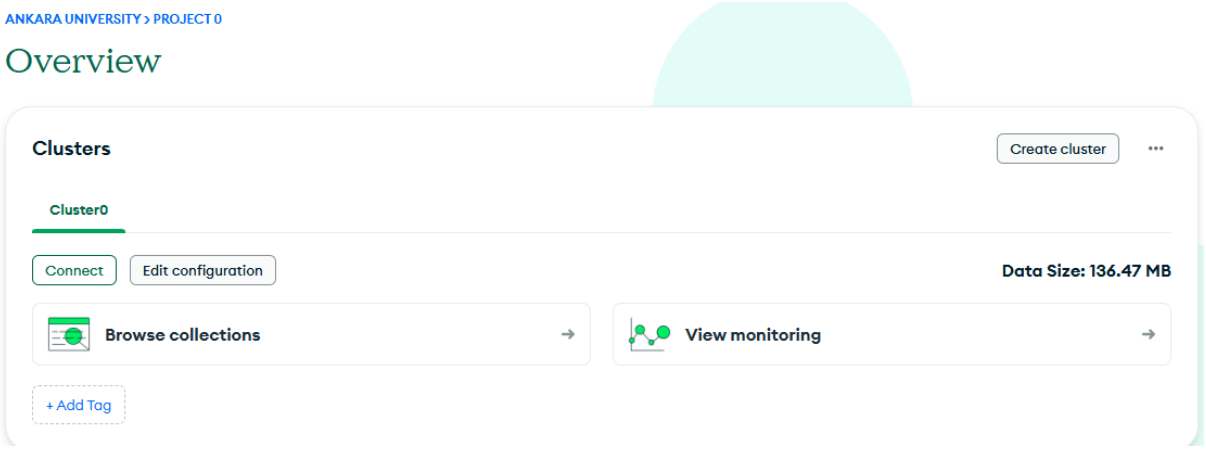


Kodumuzda kullanacağımız key’leri alıyoruz.



MongoDb

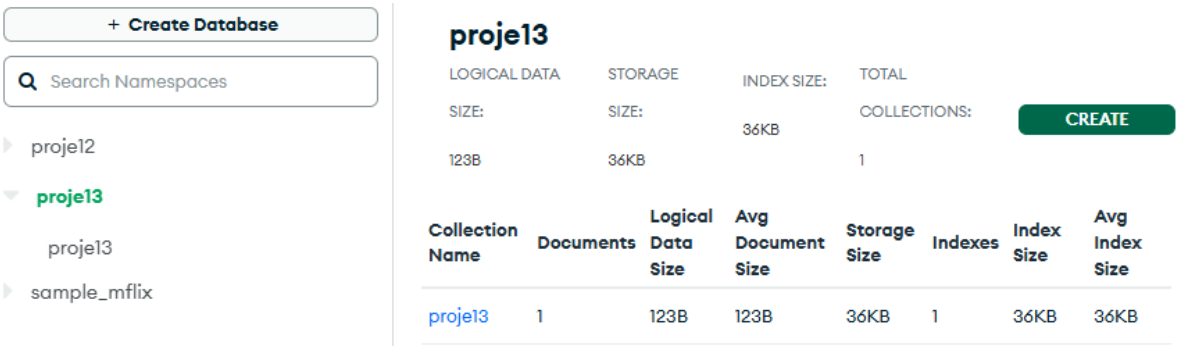
MongoDB Atlas hesabı açıldı. **Proje** oluşturuldu. Cluster oluşturuldu. (Cluster 0)



Connect kısmına girip, ardından Drivers kısmına girip Cluster’a bağlanabilmek için URI alındı.

```
mongodb+srv://<db_username>:<db_password>@cluster0.18atnrk.mongodb.net/?
retryWrites=true&w=majority&appName=Cluster0
```

Collection ve Database oluşturuldu. Verilerimiz buraya kaydedilecek.



KODLAR

app.py

Uygulamamızı bu dosyayı çalıştırarak başlatacağız.

Gerekli kütüphaneler içe aktarılır.

```python
app.py > predict
from flask import Flask, request, jsonify, render_template
import requests
import os
from pymongo import MongoClient
```

- Flask: Web uygulamasını oluşturmak için kullanılır.
- request: HTTP isteklerini almak için kullanılır.
- jsonify: JSON formatında cevap döndürmek için.
- render_template: HTML dosyalarını yüklemek için.
- requests: Azure API'sine HTTP isteği göndermek için.
- pymongo: MongoDB ile bağlantı kurmak için.

MongoDB bağlantısı kurulur.

```python
# MongoDB bağlantısı
MONGO_URI = "mongodb+srv://user11:test11@cluster0.18atnrk.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0"
client = MongoClient(MONGO_URI)

# Veritabanı ve koleksiyon seç
db = client["proje13"]
collection = db["proje13"]
```

Flask uygulaması başlatılır.

```python
app = Flask(__name__)
```

Azure üzerinde eğitilen makine öğrenmesi modelinin endpoint'i ve API anahtarı yazılır.

```
AZURE_ENDPOINT = "https://d-proje-test.switzerlandnorth.inference.ml.azure.com/score"  # örnek: https://
AZURE_API_KEY = "5twuxkYoXwITTsf7aJVNLjKsArIwzL9ECgpzM2PhgH51H6oG6CTHJQQJ99BFAAAAAAAAAAAAINFRAZML4Cxz"
```

Azure API'sine yapılacak isteklerde gerekli olan header bilgileri belirlenir. JSON formatı ve kimlik doğrulama için API anahtarı kullanılır.

```
headers = {
    "Content-Type": "application/json",
    "Authorization": f"Bearer {AZURE_API_KEY}"
}
```

Anasayfa tanımlanır. Kullanıcı siteye girdiğinde index.html dosyası tarayıcıda görüntülenir.

```
@app.route("/")
def index():
    return render_template("index.html")
```

/predict adresine POST isteği gönderildiğinde çalışacak olan predict() fonksiyonu tanımlanır.

```
@app.route("/predict", methods=["POST"])
def predict():
```

Kullanıcıdan gelen veriler JSON formatında alınır. Azure modeli için gerekli giriş formatı oluşturulur. Bu kısımda kullanıcıdan alınan veriler belirli bir sıraya göre bir liste içine yerleştirilir.

```
try:
    data = request.get_json()

    payload = {
        "input_data": {
            "columns": [
                "Pregnancies",
                "Glucose",
                "BloodPressure",
                "SkinThickness",
                "Insulin",
                "BMI",
                "DiabetesPedigreeFunction",
                "Age"
            ],
            "index": [0],
            "data": [[
                data["pregnancies"],
                data["glucose"],
                data["bloodPressure"],
                data["skinThickness"],
                data["insulin"],
                data["bmi"],
                data["diabetesPedigreeFunction"],
                data["age"]
            ]]
        }
    }
```

Azure modeline POST isteği gönderilir. Raise_for_status() hata varsa istisna ortaya çıkartır. Gelen sonuç JSON formatına dönüştürülür.

```
response = requests.post(AZURE_ENDPOINT, headers=headers, json=payload)
response.raise_for_status()
result = response.json()
```

Azure'dan gelen yanıtın biçimi kontrol edilir. Eğer liste biçimindeyse sadece tahmin değeri alınır. Eğer sözlük (dict) biçimindeyse predicted_label ve probability (olasılık) değerleri alınır. Beklenmeyen formatta yanıt alınırsa hata döndürülür.

```
if isinstance(result, list):
    prediction = result[0]
    probability = None
elif isinstance(result, dict) and "result" in result:
    prediction = result["result"][0].get("predicted_label")
    probability = result["result"][0].get("probability")
else:
    return jsonify({"error": "Beklenmeyen yanıt formatı", "received_response": result}), 500
```

Kullanıcının girdiği veriler ve tahmin sonucu save_data sözlüğüne aktarılır. Bu veri MongoDB veritabanına kaydedilir.

```
# MongoDB'ye veriyi kaydet
save_data = {
    "input": payload["input_data"]["data"][0],
    "prediction": prediction,
    "probability": probability
}
collection.insert_one(save_data)
```

Sonuçlar (tahmin ve varsa olasılık) JSON formatında kullanıcıya döndürülür.

```
return jsonify({
    "prediction": prediction,
    "probability": probability
})
```

Azure API isteği sırasında veya başka bir hata oluşursa uygun hata mesajı verilir ve 500 hata kodu döndürülür.

```
except requests.exceptions.RequestException as e:
    print("API İsteği Hatası:", str(e))
    return jsonify({"error": f"API isteği başarısız: {str(e)}"}), 500
except Exception as e:
    print("Hata oluştu:", str(e))
    return jsonify({"error": str(e)}), 500
```

Uygulama doğrudan çalıştırıldığında Flask sunucusu başlatılır. debug=True, geliştirme aşamasında hataların kolayca görülmesini sağlar.

```
if __name__ == "__main__":
    app.run(debug=True)
```

index.html

Bu dosya, kullanıcıların diyabet riskini hesaplayabileceği bir web arayüzüdür. Temiz ve anlaşılır bir form tasarımıyla 8 farklı sağlık parametresini sorgular. Arka planda çalışan JavaScript kodu, kullanıcı girdilerini alıp sunucuya göndererek tahmin sonucunu ekranda gösterir. Form gönderildiğinde JavaScript ile /predict endpoint'ine AJAX isteği gönderir ve sonucu dinamik olarak görüntüler.

style.css

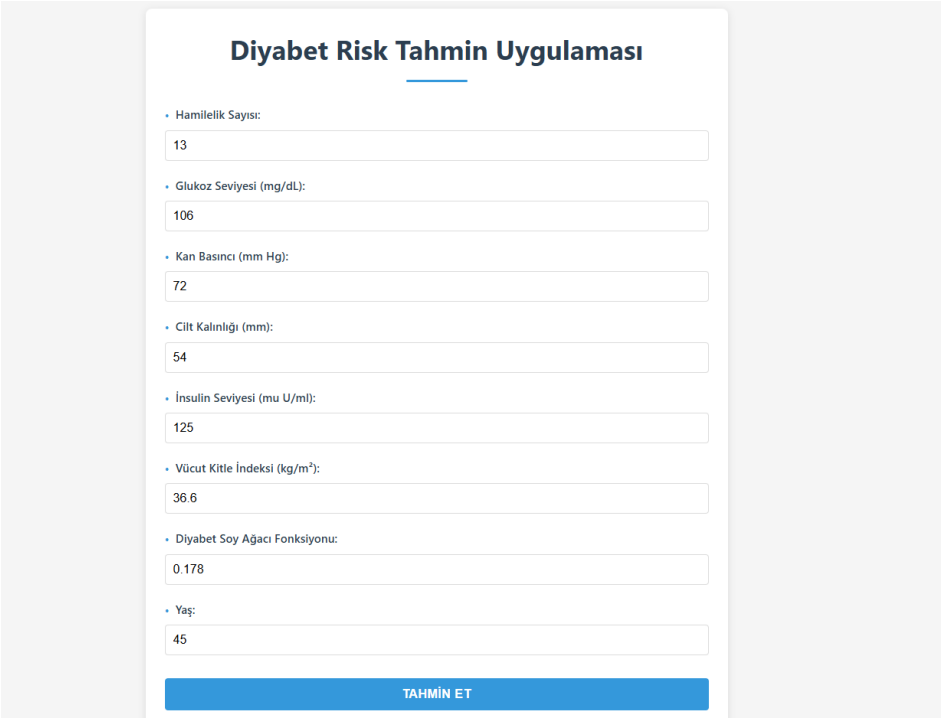
Form alanlarına odaklanma efektleri ve tahmin sonuçları için özel tasarlanmış görsel vurgular bulunur. Özellikle diyabet risk sonuçları için "pozitif/negatif" durumlara özel renk kodlaması yapılmıştır.

UYGULAMANIN ÇALIŞMASI

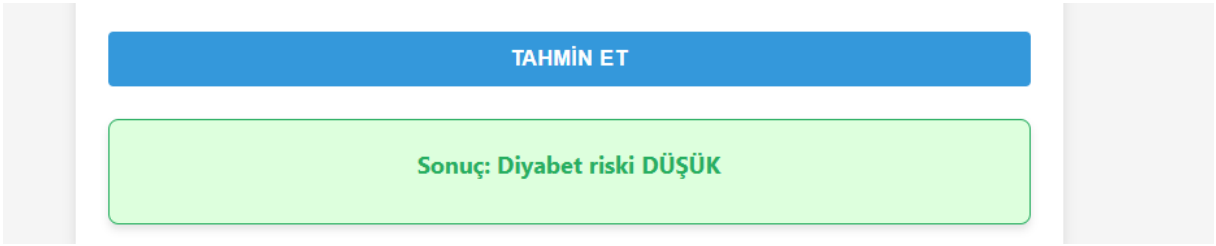
app.py dosyası çalıştırılır. http://127.0.0.1:5000 adresine tıklanır.

```
PS C:\Users\ezgis\Desktop\bulut bilişim\proje6> python -u "c:\Users\ezgis\Desktop\bulut bilişim\proje6\app.py"
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 852-342-102
```

Uygulama açılır ve bilgiler girilir.



TAHMİN ET butonuna basılır. Eğittiğimiz modele göre bu verilere sahip birinin Diyabet riskinin olup olmadığı sonucuna varırız.



Girdiğimiz bilgiler ve sonuç veritabanımızda gözükür. Prediction 0 ise diyabet riski düşüktür,1 ise yüksektir.