Ankara Üniversitesi

Mühendislik Fakültesi

Bilgisayar Mühendisliği Bölümü

Bulut Bilişim ve Uygulamaları Proje Dokümanı

Timur Malkoç

22290719

Tuğrul Özgün

22290082

Linkler

Proje 3: Akıllı Veri Analitiği ve Makine Öğrenmesi Uygulaması Github:
https://github.com/TimurMalkc/Cloud-Midterm-ML

Proje 3: Akıllı Veri Analitiği ve Makine Öğrenmesi Uygulaması Video:

https://www.youtube.com/watch?v=dnbNbAbPI-8

Proje 1: Çift Katmanlı Web Uygulaması (Web API + Frontend) Github:

https://github.com/Forcipus/todoapp

Proje 7 : Azure IoT Hub Tabanlı Akıllı Şehir Sensör Simülasyonu ve Gerçek Zamanlı

Veri Analizi Github:

https://github.com/Forcipus/IoT-City

Proje 7 : Azure IoT Hub Tabanlı Akıllı Şehir Sensör Simülasyonu ve Gerçek Zamanlı

Veri Analizi Video: https://youtu.be/MX7MNJgEx08

Proje 4: E-Ticaret Uygulaması (Otomatik Ölçeklendirme ve Yönetim) Github:

https://github.com/TimurMalkc/Cloud-Final-ECommerce

Proje 4: E-Ticaret Uygulaması (Otomatik Ölçeklendirme ve Yönetim) Video:

https://www.youtube.com/watch?v=GUNPMw-wAG0

Proje 2: Gerçek Zamanlı Veri Akışı ve İşleme (IoT veya WebSocket Uygulaması) Github:
https://github.com/Forcipus/IoT-Deprem

Proje 2: Gerçek Zamanlı Veri Akışı ve İşleme (IoT veya WebSocket Uygulaması) Video:

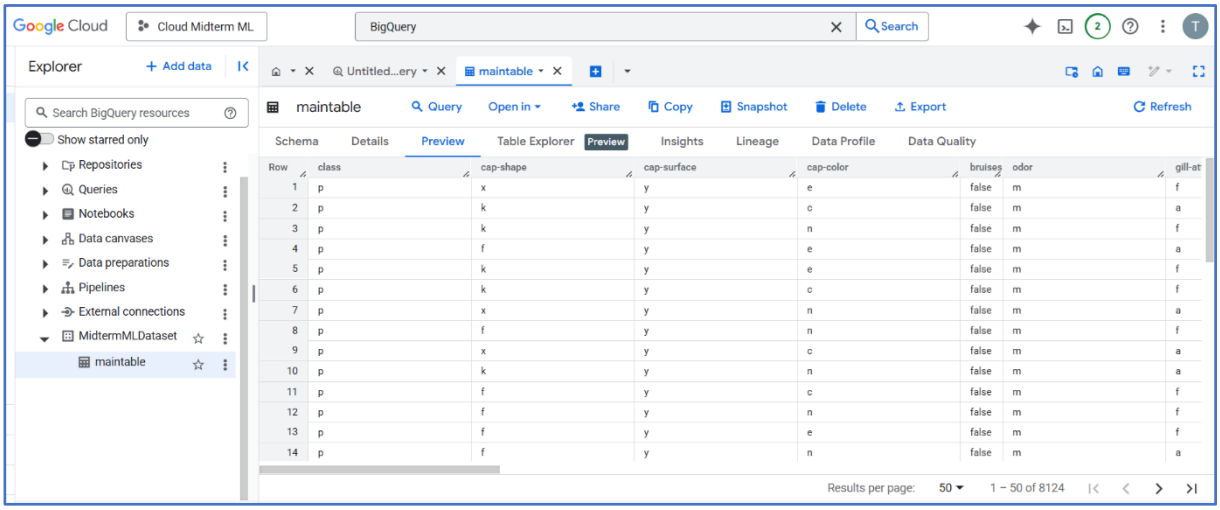
https://youtu.be/RTdKM2Jt36k

Hocam merhaba, Vize döneminde teslim ettiğim "Çift Katmanlı Web Uygulaması" projesine ait video ile ilgili bir sorun fark ettim. Bu konuda size daha önce e-posta yoluyla ulaşmaya çalıştım ancak geri dönüş alamadığım için bu açıklamayı buradan yapmak istedim. Projeyi, kota yetersizliği nedeniyle silmek zorunda kaldığım için videoyu yeniden çekme imkânım olmadı. Bu nedenle yerine "Gerçek Zamanlı Veri Akışı" projesini tamamladım ve bu projeye ait tüm dokümanlar ve gerekli bilgiler eksiksiz şekilde hazır. Bu projeyi değerlendirmeye alabilirseniz çok memnun olurum. İyi çalışmalar dilerim.

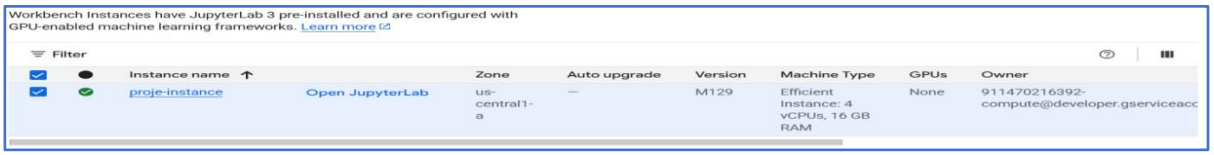
Proje 3: Akıllı Veri Analitiği ve Makine Öğrenmesi Uygulaması

Proje içerisinde back-end dili olarak Python, makine öğrenmesi kütüphanesi olarak Scikit-learn, veritabanı olarak BigQuery ve bulut platformu olarak Google Cloud kullanılmıştır. Kullanılacak veri seti olarak Kaggle üzerinden alınmış “Mushroom Classification” seçilmiştir.

Veri seti indirildikten sonra bir dataset ve table oluşturulup veri seti, BigQuery üzerine aktarılmıştır. BigQuery hali hazırda Google Cloud üzerinde bulunmaktadır ve bağlantı buradan sağlanmıştır.



Python programlaması için ise tekrardan Google Cloud üzerinde bulunan Jupyter Notebook kullanılmıştır. Jupyter Notebook kullanımı için Vertex AI platformu üzerinden bir workbench instance açılmıştır.



Kod, verinin her seferinde tekrar tekrar yüklenmemesi için ve okunabilirlik açısından 4 Jupyter Notebook hücresine bölünmüştür. Birinci hücrede gerekli kütüphaneler ve metotlar import edilip veri tabanı BigQuery üzerinden çekilmiştir.

[26]:
```
import pandas as pd
import pandas_gbq as pdq
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report

sql = "SELECT * FROM `cloud-midterm-ml.MidtermMLDataset.maintable`"
mushroomData = pdq.read_gbq(sql)

mushroomData
```

```
Downloading: 100%|
```

[26]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat
0	p	x	y	e	False	m	f	c	b	w	...	y	c	c	p	w	n	n	w	c	d
1	p	k	y	c	False	m	a	c	b	w	...	y	c	c	p	w	n	n	w	c	d
2	p	k	y	n	False	m	f	c	b	w	...	y	c	c	p	w	n	n	w	c	d
3	p	f	y	e	False	m	a	c	b	w	...	y	c	c	p	w	n	n	w	c	d
4	p	k	y	e	False	m	f	c	b	y	...	y	c	c	p	w	n	n	w	c	d
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8119	e	f	y	n	True	n	f	c	b	w	...	s	e	e	p	w	t	e	w	c	w
8120	e	k	s	e	True	n	f	c	b	e	...	s	e	e	p	w	t	e	w	c	w
8121	e	x	y	n	True	n	f	c	b	w	...	s	e	w	p	w	t	e	w	c	w

İkinci hücrede kategorik verinin işlenebilmesi için LabelEncoder metotu kullanılarak veriler her birine karşılık gelen sayılar ile kodlanmıştır. Hangi verinin hangi sayıya karşılık geldiği de enumerator aracılığı ile çıktı olarak gösterilmiştir. Veri setinin sütunlarının tamamının kullanılması overfitting bir modele sebep olmasından ve benzer özelliklerin modeli gereksiz karmaşıklaştırmasından dolayı tüm sütunların içerisinden 6 tanesi seçilip modelde kullanılmıştır.

[27]:
```
selectedCols = ['class','bruises','gill-attachment','gill-spacing','gill-size','stalk-shape']
labelEncoder = LabelEncoder()

for col in selectedCols:
    mushroomData[col] = labelEncoder.fit_transform(mushroomData[col])
    print(f"\nEncoding for column: {col}")
    for i, class_ in enumerate(labelEncoder.classes_):
        print(f"{class_} => {i}")
```

```
Encoding for column: class
e => 0
p => 1

Encoding for column: bruises
0.0 => 0
1.0 => 1

Encoding for column: gill-attachment
a => 0
f => 1

Encoding for column: gill-spacing
c => 0
w => 1

Encoding for column: gill-size
b => 0
n => 1

Encoding for column: stalk-shape
e => 0
t => 1
```

Üçüncü hücrede Sci-kit learn makine öğrenmesi metotları kullanılarak model eğitilmiştir.

```
X = mushroomData[['bruises', 'gill-attachment', 'gill-spacing', 'gill-size', 'stalk-shape']]
y = mushroomData['class']
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=42)

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train, y_train)

y_pred = knn.predict(x_test)

print("\nKNN Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
KNN Accuracy: 0.9098966026587888

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.94      0.92      2123
           1       0.93      0.88      0.90      1939

    accuracy                           0.91      4062
   macro avg       0.91      0.91      0.91      4062
weighted avg       0.91      0.91      0.91      4062
```

Son hücrede kullanıcıdan girdi alma sistemi eklenmiştir ve belirtilen özelliklere göre model bir mantarın zehirli mi değil mi olduğunu söyleyebilmektedir.

```
bruises = int(input("Bruises (1 for Yes, 0 for No): "))
gillA = int(input("Gill Attachment: (1 for Free, 0 for Attached): "))
gillSP = int(input("Gill Spacing: (1 for Crowded, 0 for Close): "))
gillSZ = int(input("Gill Size: (1 for Narrow, 0 for Broad): "))
stalk = int(input("Stalk Shape: (1 for Tapering, 0 for Enlarging): "))

mushroomInput = pd.DataFrame({'bruises':[bruises],'gill-attachment':[gillA],'gill-spacing':[gillSP],
                              'gill-size':[gillSZ],'stalk-shape':[stalk]})

pred = knn.predict(mushroomInput)

if pred == 0:
    print("This mushroom is poisonous")
else:
    print("This mushroom is edible")
```

```
Bruises (1 for Yes, 0 for No):  1
Gill Attachment: (1 for Free, 0 for Attached):  0
Gill Spacing: (1 for Crowded, 0 for Close):  0
Gill Size: (1 for Narrow, 0 for Broad):  1
Stalk Shape: (1 for Tapering, 0 for Enlarging):  0
This mushroom is edible
```

Proje 1: Çift Katmanlı Web Uygulaması (Web API + Frontend)

Task Manager Uygulaması (Frontend + Backend Deployment)

Bu proje, kullanıcıların görev (task) oluşturmasını, güncellemesini, silmesini ve filtrelemesini sağlayan çift katmanlı bir web uygulamasıdır. Arka uç RESTful API olarak Node.js (Express) ile geliştirilmiş, ön yüz ise React.js kullanılarak inşa edilmiştir. Veriler MongoDB veritabanında saklanmaktadır ve AWS de deploylanmaktadır. Veriler güvenlik için JWT encryption ile şifrelenmektedir. Proje, ölçeklenebilirlik ve erişilebilirlik açısından AWS EC2 ortamında barındırmaya uygun olarak yapılandırılmıştır.

Uygulama Özellikleri

- Görev ekleme (başlık ve tarih ile)
- Görev düzenleme (başlık, tamamlanma durumu, tarih)
- Görev silme
- Tamamlanma durumuna ve son teslim tarihine göre filtreleme
- Karanlık/Aydınlık mod desteği
- Tarih bilgilerini Türkçe formatta görüntüleme

Görevler

Yeni görev ekle | gg . aa . yyyy | Ekle

Duruma göre filtrele: Hepsi | Son tarihe göre filtrele: gg . aa . yyyy

ee	Tamamlandı	01.05.2025	Durumu Değiştir	Sil
dd	Bekliyor	07.05.2025	Durumu Değiştir	Sil
cc	Bekliyor	12.05.2025	Durumu Değiştir	Sil
bb	Tamamlandı	24.05.2025	Durumu Değiştir	Sil
aa	Tamamlandı	03.05.2025	Durumu Değiştir	Sil

Bulut Sunucuları (1/2) Bilgi

Son güncelleme 1 minute önce | Bağlan | Bulut sunucusu durumu | Eylemler | Bulut sunucularını başlatın

Bulut Sunucusu Özniteliğe veya etikete göre (case-sensitive) bulun | Tüm duru... | 1

Name	Bulut sunucusu kimliği	Bulut sunucus...	Bulut sunuc...	Durum kontrolü	Alarm durumu	Erişilebilirlik Al...
todo-app--inst...	i-05984ae76aae52490	Çalışıyor	t3.micro	3/3 denetim geçti	Alarmları görünt	eu-north-1a
todo1	i-004ecbef3e375bab2	Sonlandırıldı	t3.micro	–	Alarmları görünt	eu-north-1a

Proje 7 : Azure IoT Hub Tabanlı Akıllı Şehir Sensör Simülasyonu ve Gerçek Zamanlı Veri Analizi

Proje Amacı

Bu proje, bir akıllı şehir senaryosunda sıcaklık ve nem gibi çevresel verilerin simüle edilmiş IoT cihazları aracılığıyla toplanmasını, bu verilerin Azure IoT Hub üzerinden buluta iletilmesini ve gelen verilerin gerçek zamanlı olarak işlenip analiz edilmesini amaçlamaktadır. Amaç, IoT sistemlerinin temel bileşenlerini kullanarak uçtan uca bir veri akışı mimarisi oluşturmaktır.

Kullanılan Teknolojiler ve Araçlar

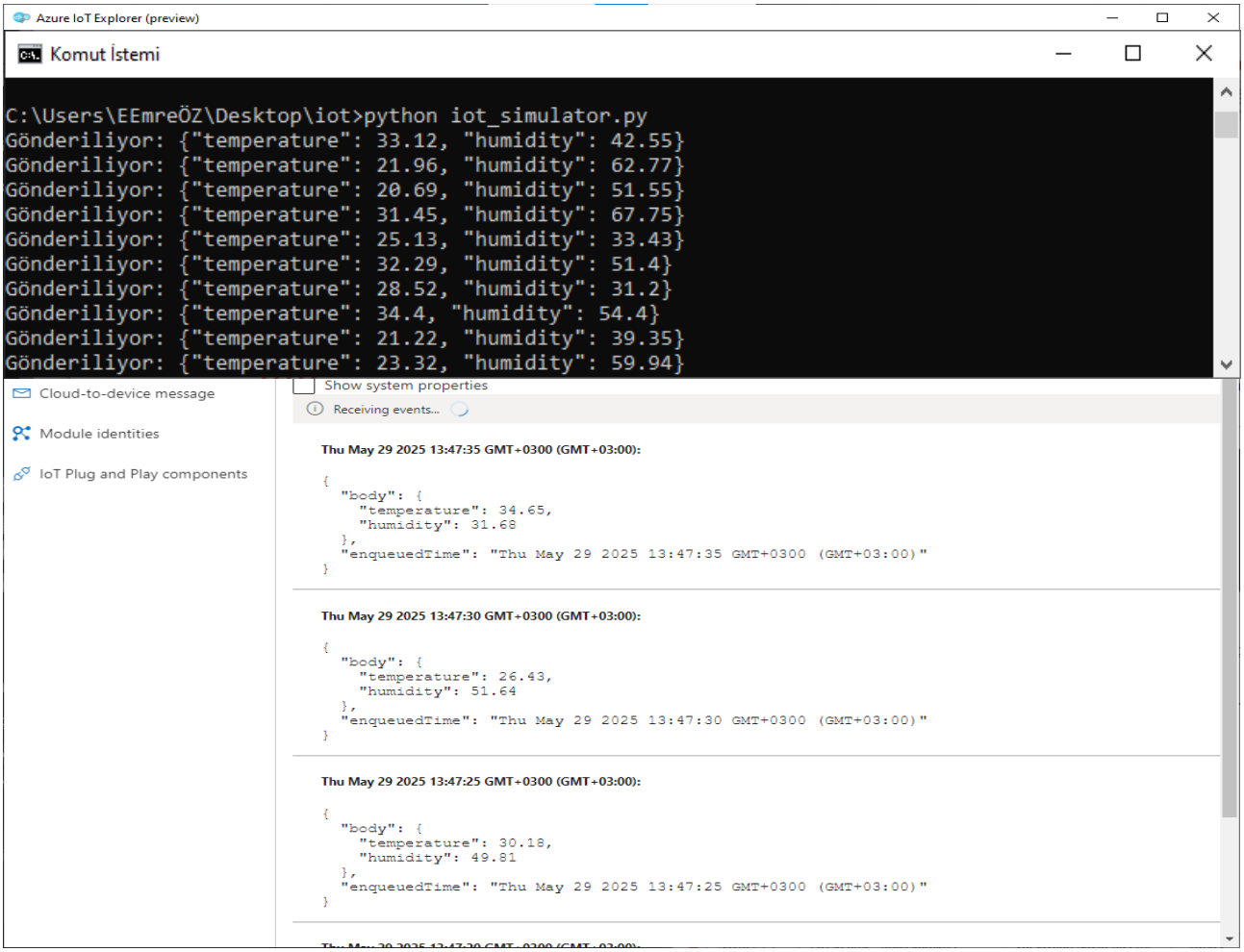
- **Programlama Dili**: Python
- **Bulut Platformu**: Microsoft Azure
    - Azure IoT Hub
    - Azure Event Hub (IoT Hub'ın Event Hub uyumlu uç noktası)
- **Veri Analizi**: Python ile anlık ortalama sıcaklık ve nem hesaplamaları
- **Cihaz Simülasyonu**: Python betiği ile rastgele veri üretimi

1. Azure IoT Hub Oluşturulması

- Microsoft Azure portalına giriş yapılarak, bir "IoT Hub" kaynağı oluşturuldu.
- IoT Hub içindeki "Devices" sekmesinden bir cihaz kaydedildi. Bu cihazın adı ve **connection string** bilgisi daha sonra simülasyon kodunda kullanılmak üzere not edildi.

2. IoT Cihazı Emulatörü (Veri Gönderimi)

- Python ile iot_simulator.py adında bir dosya oluşturuldu.
- Bu dosyada:
    - Azure IoT Device SDK (azure.iot.device) kullanılarak doğrudan Azure IoT Hub'a bağlantı sağlandı.
    - Rastgele sıcaklık ve nem verileri üreten bir algoritma yazıldı.
    - Her 5 saniyede bir bu veriler JSON formatında Azure IoT Hub'a gönderildi.
    - Kod çalıştırıldığında terminalde mesajların başarıyla gönderildiği görüldü



3. Azure IoT Explorer ile Doğrulama

- Azure tarafından sunulan masaüstü aracı **Azure IoT Explorer** kullanılarak gönderilen veriler görüntülendi.
- Cihaz bağlantısı kurularak gelen mesajların formatı ve frekansı gerçek zamanlı izlendi.

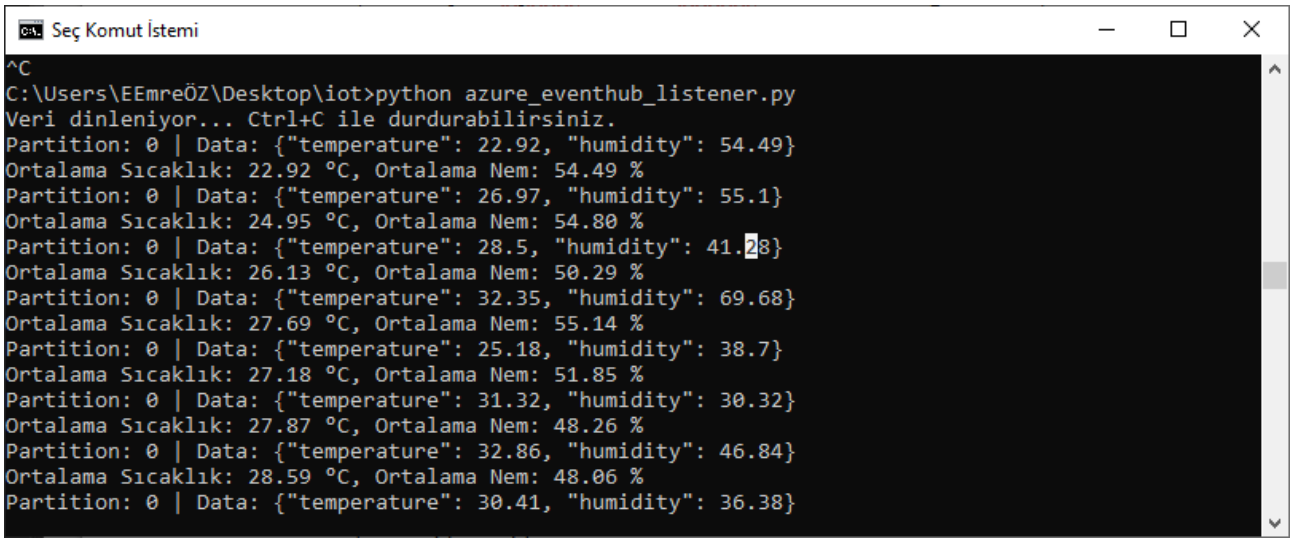
4. Event Hub Uyumlu Uç Nokta ile Veri Dinleme

- IoT Hub içerisindeki **Event Hub uyumlu uç nokta** bilgileri ve erişim anahtarları alındı.
- azure-eventhub kütüphanesi kullanılarak bir Python dosyası (azure_eventhub_listener.py) oluşturuldu.
- Bu kod:
    - Event Hub'a bağlandı.
    - Her gelen mesajı parse ederek sıcaklık ve nem değerlerini terminalde yazdırdı.
    - Tüm gelen mesajların ortalamasını hesaplayarak analiz yaptı.

1. IoT Cihazı (Simülasyon: Python kodu ile MQTT üzerinden veri gönderimi)
↓
2. Azure IoT Hub (Veriyi alır ve Event Hub'a yönlendirir)
↓
3. Event Hub Uyumlu Uç Nokta (Veriyi dış sistemlere açar)
↓
4. Python Listener (azure-eventhub-listener.py ile veriyi dinler ve işler)

5. Gerçek Zamanlı Veri Analizi

- Kod çalışırken her gelen veri üzerinde:
    - Anlık olarak sıcaklık ve nem verileri gösterildi.
    - Tüm veriler bir listede saklandı.
    - Bu liste üzerinde mean() hesaplamalarıyla ortalama değerler anlık olarak terminalde verildi.
- Bu kısım, veri görselleştirme yerine ilk analiz katmanı olarak işlev gördü.



6. Grafik Çizimi ve Görselleştirme

Veri analizinin bir sonraki adımı olarak, simülasyon sonucu elde edilen sıcaklık ve nem verilerinin grafiksel olarak görselleştirilmesi gerçekleştirildi. Bu sayede verilerin zamana göre değişimi incelenerek daha anlamlı çıkarımlar elde edildi.

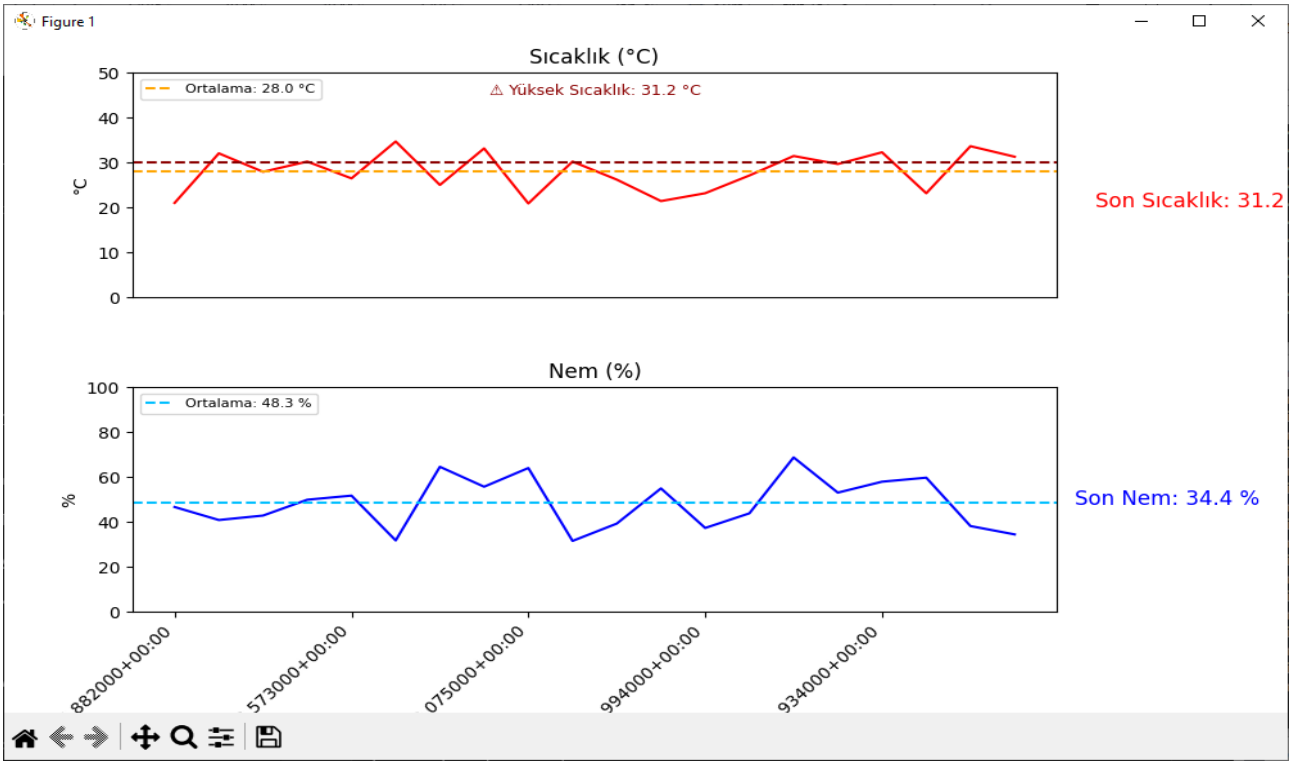
A. Python ile Grafik Çizimi (grafik_çiz.py)

- matplotlib ve json kütüphaneleri kullanılarak basit bir görselleştirme aracı geliştirildi.
- azure_eventhub_listener.py betiği ile toplanan sıcaklık ve nem verileri bir .json dosyasına kaydedildi.
- Ardından bu JSON dosyası grafik_çiz.py dosyası tarafından okunarak çizgi grafik şeklinde görselleştirildi.

B. MATLAB ile İleri Seviye Görselleştirme

MATLAB, bilimsel veri analizi ve istatistiksel grafikler oluşturmak için güçlü bir araçtır. Bu projede Python ile kaydedilen veriler MATLAB ortamına aktarılmış ve burada daha gelişmiş analizler yapılmıştır.

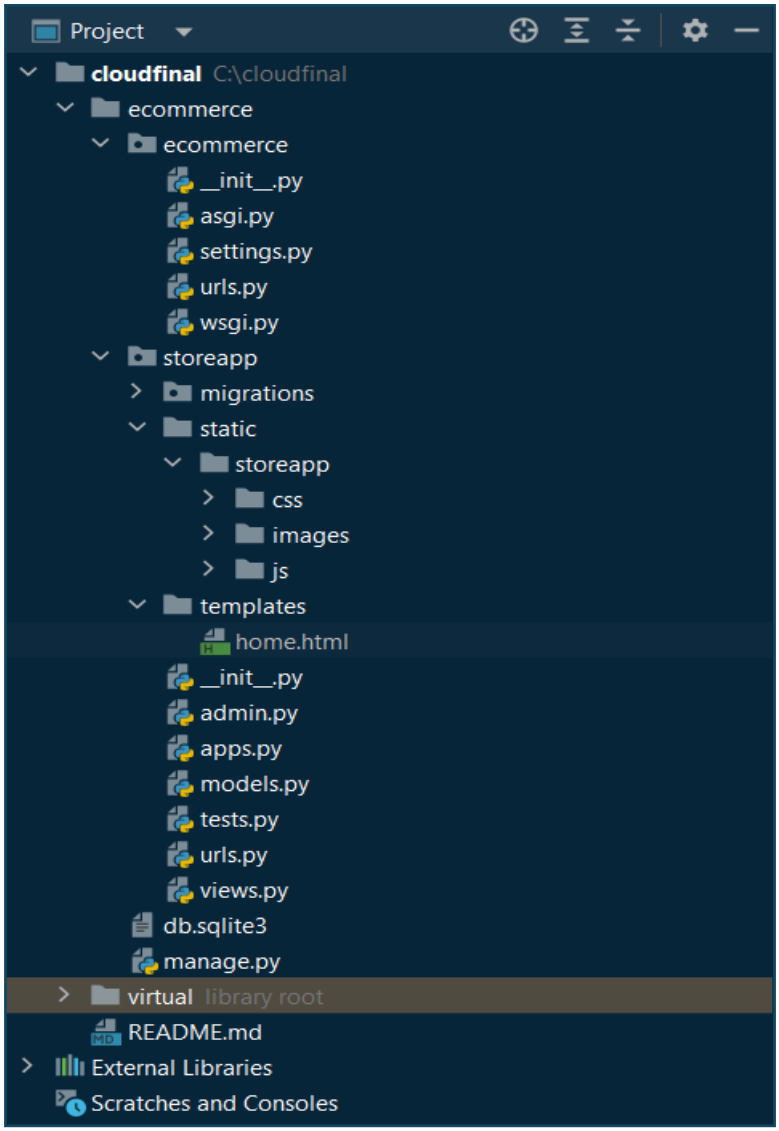
- Python'da oluşturulan JSON dosyası, CSV formatına dönüştürüldü.
- MATLAB'de CSV dosyası içe aktarıldı:
- İkili eksen kullanan detaylı bir grafik oluşturuldu:



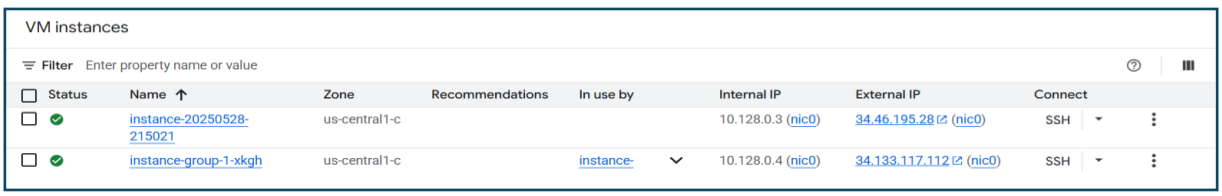
- Grafik aldığı JSON dosyasındaki verileri gerçek zamanlı tabloya aktardı.
- Eğer sıcaklık ve nem belirlenen değerin üzerinde ise uyarı verdi
- O ana kadar girilen verilerden yola çıkarak hesaplanan ortalama sıcaklık ve nemi gösterdi

Proje 4: E-Ticaret Uygulaması (Otomatik Ölçeklendirme ve Yönetim)

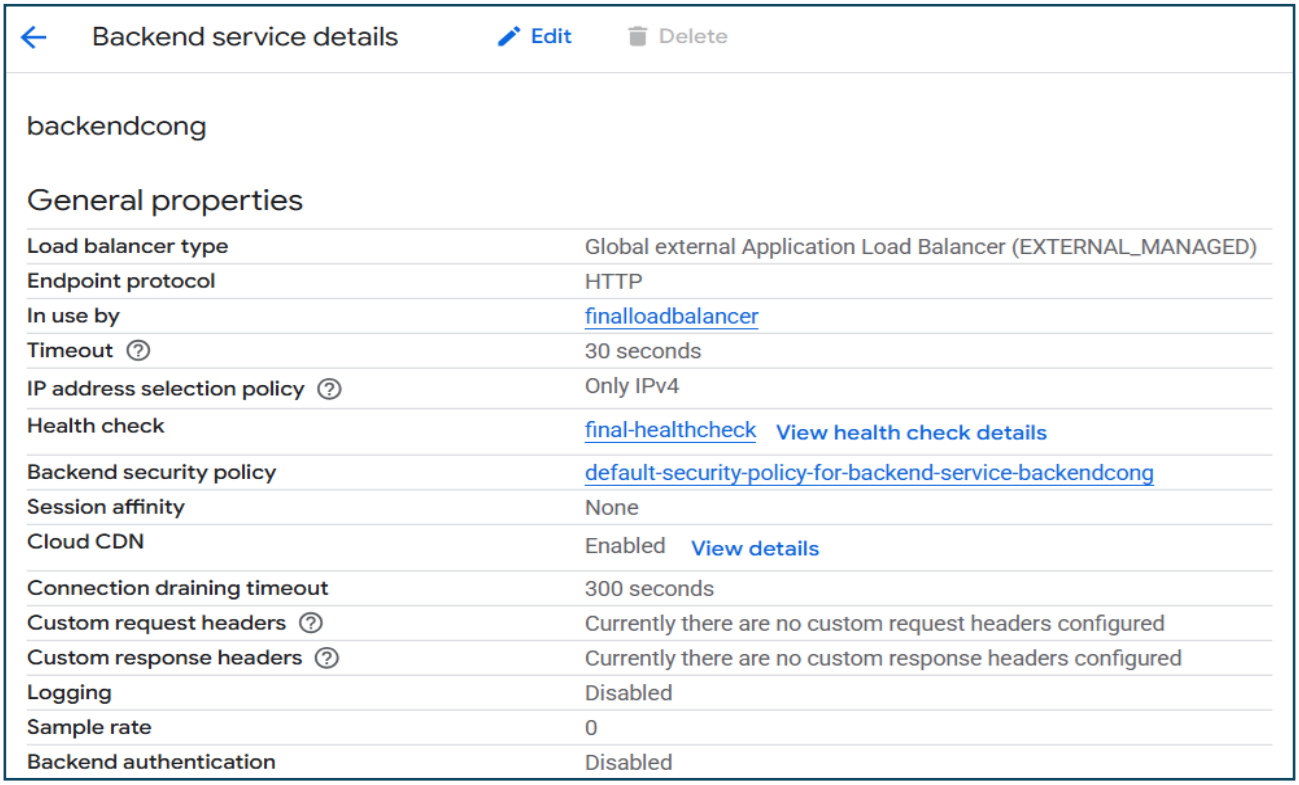
Proje içerisinde back-end dili olarak Python, back-end kütüphanesi olarak Django, veritabanı olarak MySQL ve bulut platformu olarak Google Cloud kullanılmıştır. E-Ticaret sitesinin görünümü internet üzerinden hazır olarak alınıp bir Django uygulaması üzerine aktarılmıştır. Python üzerinde html dosyası templates klasörüne; diğer css, js ve png dosyaları da static klasörüne atılmış, bağlantılar yeni dosya yollarına göre düzenlenerek back-end kısmı tamamlanmıştır.

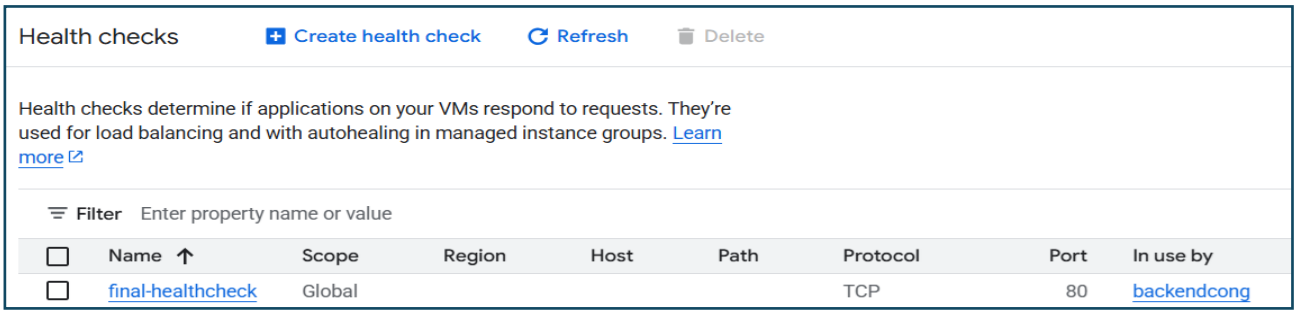


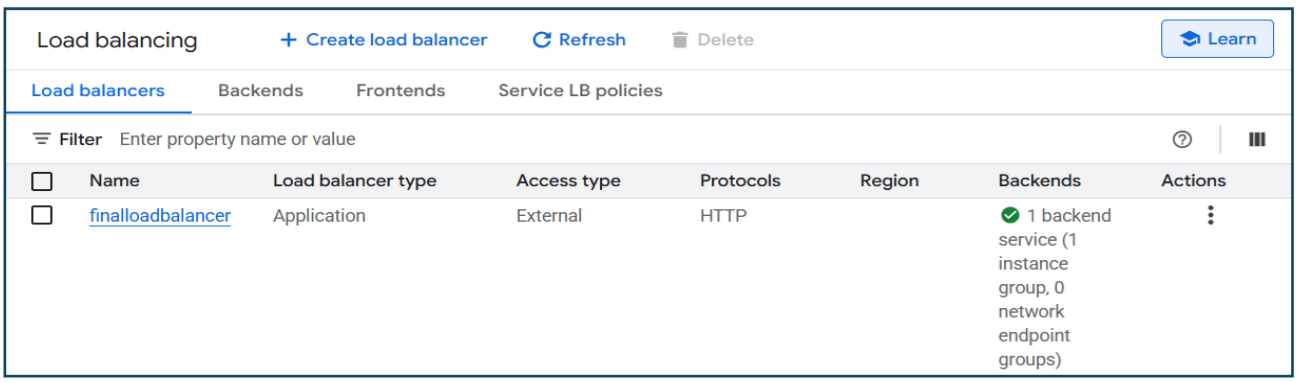
Google Cloud üzerinde bulunan Compute Engine hizmeti kullanılarak websitesi için bir instance ve instance group oluşturulmuştur.



Oluşturalan instance group haricinde bir back-end config ve health check hazırlanmıştır. Bu hizmetlerin üçü kullanılarak Google Cloud Load Balancer hizmeti ile otomatik ölçeklendirme yapılmıştır.







Compute Engine instance ile sağlanan SSH (secure shell) kullanılarak websitesine bir MySQL veri tabanı eklenmiştir.

```
(venv) timurmlkc@instance-20250528-215021:~/Cloud-Final-ECommerce$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.42 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

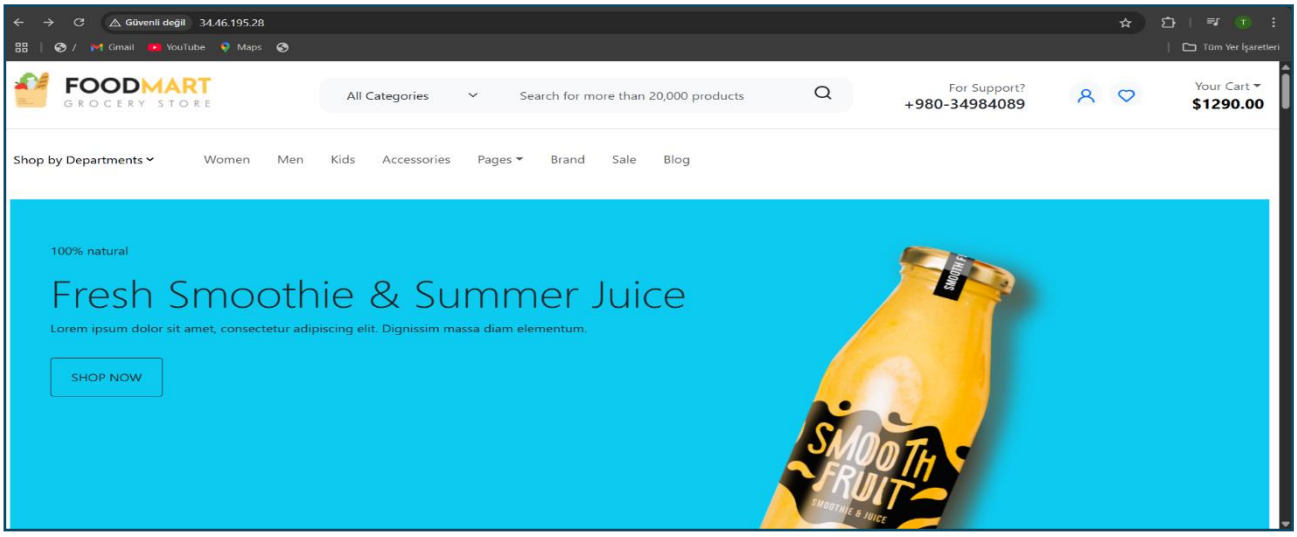
Bütün özelliklerin geliştirilmesi tamamlandıktan sonra SSH üzerinden Nginx ve Gunicorn yardımı ile istenilen ip adresi üzerinde bir sunucu açılmıştır.

```
ssh.cloud.google.com/v2/ssh/projects/midterm-commerce/zones/us-central1-c/instances/instance-20250528-215021?authuser=2&hl=en_...
ssh.cloud.google.com/v2/ssh/projects/midterm-commerce/zones/us-central1-c/instances/instance-20250528-215021?authuser=2...
SSH-in-browser                                   UPLOAD FILE   DOWNLOAD FILE

(venv) timurmlkc@instance-20250528-215021:~/Cloud-Final-ECommerce$ sudo systemctl status gunicorn
● gunicorn.service - gunicorn daemon for Django app
     Loaded: loaded (/etc/systemd/system/gunicorn.service; enabled; preset: enabled)
     Active: active (running) since Thu 2025-05-29 22:48:58 UTC; 7min ago
   Main PID: 2610 (gunicorn)
      Tasks: 4 (limit: 1136)
     Memory: 104.8M
        CPU: 1.610s
     CGroup: /system.slice/gunicorn.service
             ├─2610 /home/timurmlkc/venv/bin/python3 /home/timurmlkc/venv/bin/gunicorn --access-logfile - --work>
             ├─2612 /home/timurmlkc/venv/bin/python3 /home/timurmlkc/venv/bin/gunicorn --access-logfile - --work>
             ├─2613 /home/timurmlkc/venv/bin/python3 /home/timurmlkc/venv/bin/gunicorn --access-logfile - --work>
             └─2614 /home/timurmlkc/venv/bin/python3 /home/timurmlkc/venv/bin/gunicorn --access-logfile - --work>

May 29 22:49:12 instance-20250528-215021 gunicorn[2613]: Not Found: /images/ad-image-1.png
May 29 22:49:12 instance-20250528-215021 gunicorn[2613]:  - - [29/May/2025:22:49:12 +0000] "GET /images/ad-image>
May 29 22:49:12 instance-20250528-215021 gunicorn[2613]: Not Found: /images/ad-image-2.png
May 29 22:49:12 instance-20250528-215021 gunicorn[2613]:  - - [29/May/2025:22:49:12 +0000] "GET /images/ad-image>
May 29 22:49:12 instance-20250528-215021 gunicorn[2614]: Not Found: /images/ad-image-3.png
May 29 22:49:12 instance-20250528-215021 gunicorn[2614]:  - - [29/May/2025:22:49:12 +0000] "GET /images/ad-image>
May 29 22:49:12 instance-20250528-215021 gunicorn[2613]: Not Found: /images/ad-image-4.png
May 29 22:49:12 instance-20250528-215021 gunicorn[2613]:  - - [29/May/2025:22:49:12 +0000] "GET /images/ad-image>
May 29 22:49:13 instance-20250528-215021 gunicorn[2612]: Not Found: /images/bg-leaves-img-pattern.png
May 29 22:49:13 instance-20250528-215021 gunicorn[2612]:  - - [29/May/2025:22:49:12 +0000] "GET /images/bg-leave>
log file: udo nginx -t
[1]+  Stopped                 sudo systemctl status gunicorn
(venv) timurmlkc@instance-20250528-215021:~/Cloud-Final-ECommerce$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
(venv) timurmlkc@instance-20250528-215021:~/Cloud-Final-ECommerce$ sudo systemctl restart nginx
(venv) timurmlkc@instance-20250528-215021:~/Cloud-Final-ECommerce$
```

Google Cloud tarafından sağlanılan ip adresi ile websitesine erişilebilmektedir.



Proje 2: Gerçek Zamanlı Veri Akışı ve İşleme (IoT veya WebSocket Uygulaması) Github:

Proje Amacı

Bu proje, deprem sensörlerinden alınan verilerin bir IoT senaryosu ile nasıl gerçek zamanlı şekilde toplanabileceğini, bulut platformları üzerinden işlenebileceğini ve veritabanına kaydedilip görselleştirilebileceğini göstermek amacıyla geliştirilmiştir.

Gerçek hayattaki deprem izleme sistemlerinin küçük ölçekli bir simülasyonu yapılmıştır.
Proje şu ihtiyaçlara cevap vermektedir:

- Gerçek zamanlı veri akışı ve yönetimi
- IoT sensör verilerinin bulut üzerinde toplanması
- Verilerin kalıcı bir veritabanında saklanması
- Türkiye haritası üzerinde deprem verilerinin görselleştirilmesi

Kullanılan Teknolojiler ve Araçlar

- **Programlama Dili**: Python
- **Bulut Platformu**: Microsoft Azure
    - Azure IoT Hub
    - Azure Event Hub (IoT Hub'ın Event Hub uyumlu uç noktası)
- **Paho MQTT Client:** MQTT protokolü ile sensör verilerini yayınlama ve dinleme
- **Veri Depolama:** Mongodb
- **Cihaz Simülasyonu**: Python betiği ile rastgele veri üretimi

1. Azure IoT Hub Oluşturulması

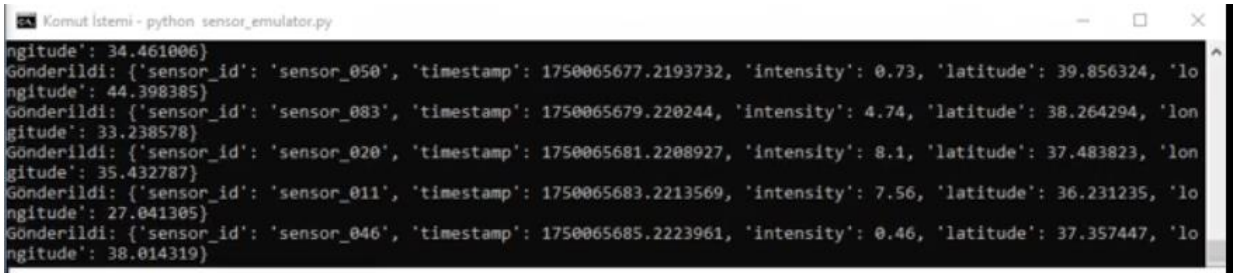
- Microsoft Azure portalına giriş yapılarak, bir "IoT Hub" kaynağı oluşturuldu.
- IoT Hub içindeki "Devices" sekmesinden bir cihaz kaydedildi. Bu cihazın adı ve **connection string** bilgisi daha sonra simülasyon kodunda kullanılmak üzere not edildi.

2.sensor_emulator.py **(Veri Gönderimi)**

- Rasgele sensör verileri üretir:
    - sensor_id : sensor_001 - sensor_100 arası bir kimlik
    - timestamp : UNIX zaman damgası

- intensity : 0-10 arası rasgele deprem şiddeti
- latitude ve longitude : Türkiye sınırları içinde rasgele koordinatlar

- Üretilen JSON verisini deprem/sensor konusuna MQTT broker'a yollar.
- 2 saniyede bir yeni veri üretir.



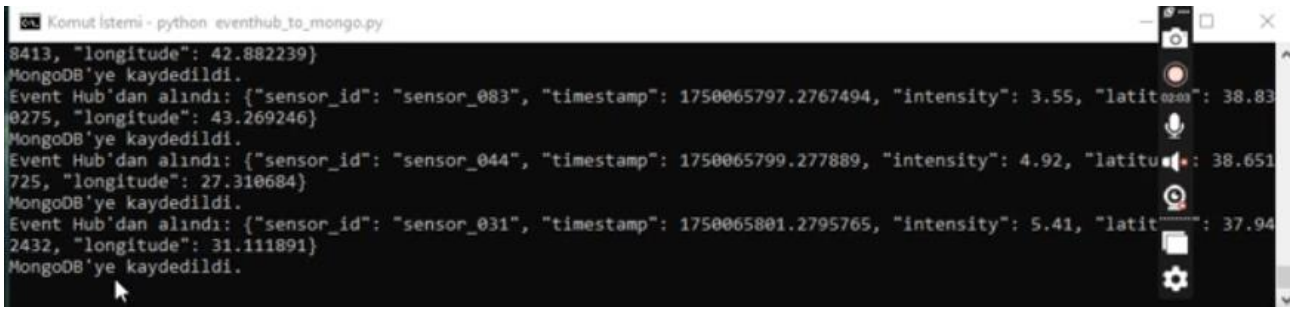
3. mqtt_to_eventhub.py

- broker.hivemq.com adresindeki deprem/sensor konusunu dinler.
- Gelen veriyi Azure Event Hub'a yollar.
- Event Hub bağlantısı Event Hub connection string ve eventhub adı üzerinden yapılır.

- Veriler JSON string formatında Azure'a aktarılır.

4. eventhub_to_mongo.py

- Azure Event Hub'dan veri alır.
- Veriyi parse ederek (JSON nesnesi) MongoDB'de depremDB.sensorData koleksiyonuna kaydeder.
- Veriyi aldıktan sonra Event Hub checkpoint'i günceller (tekrar işlememek için).



[Sensör Simülatörü (Python)]

|

▼ (MQTT yayın)

[broker.hivemq.com (MQTT Broker)]

|

▼ (MQTT dinleme)

[mqtt_to_eventhub.py (Azure Event Hub'a iletim)]

|

▼

[Azure Event Hub (mesaj kuyruğu)]

|

▼

[eventhub_to_mongo.py (MongoDB'ye kaydet)]

|

▼

[MongoDB (deprem veritabanı)]

|

▼

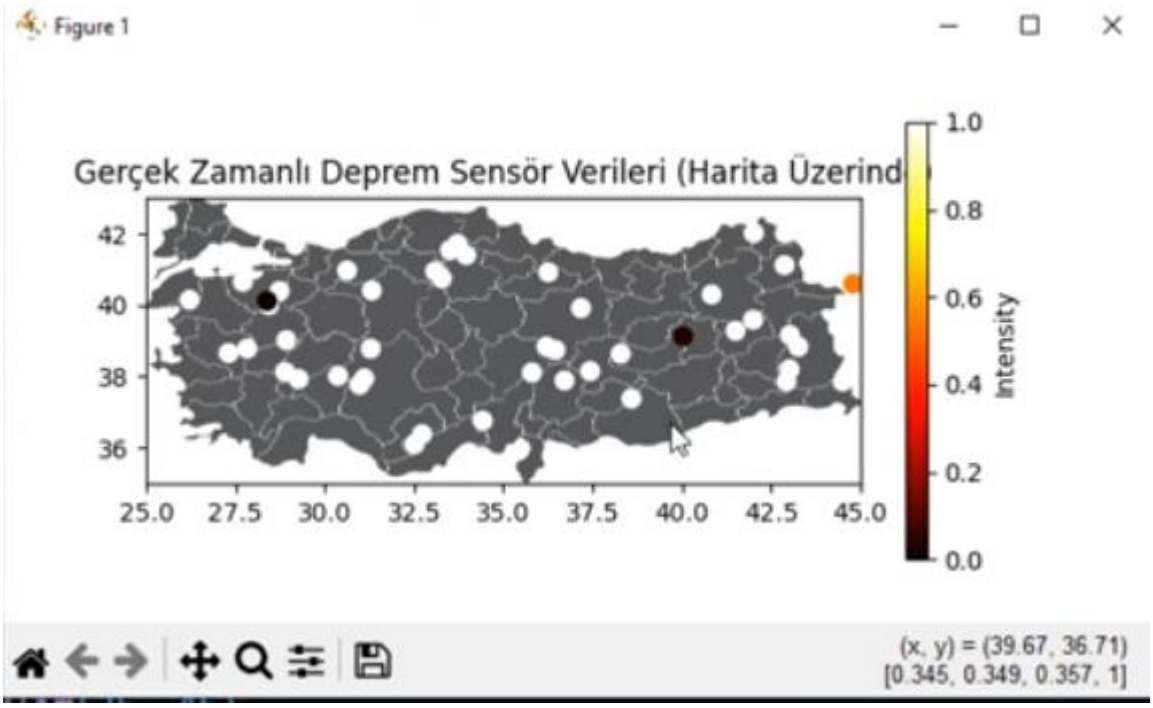
[map.py (Harita görselleştirme)]

5. map.py

- MongoDB’deki tüm verileri okur.
- Türkiye haritasını (PNG) arka plana koyar.
- Alınan koordinatlara göre scatter plot ile noktalar çizer.

6. Grafik Çizimi ve Görselleştirme

Veri analizinin bir sonraki adımı olarak, simülasyon sonucu elde edilen verilerinin Türkiye sınırları içinde üretilen koordinatlar harita üzerinde noktalar olarak grafiksel olarak görselleştirilmesi gerçekleştirildi.



- Bu proje, IoT cihazlarından alınan verilerin bulut platformlarıyla nasıl entegre edileceğini ve görselleştirileceğini basit ama etkili bir örnekle göstermektedir. Gerçek sistemlerde benzeri bir mimari, erken uyarı ve izleme sistemlerinin temelini oluşturabilir.