Proje 3. Akıllı Veri Analitiği ve Makine Öğrenmesi Uygulaması

Bu projede temel bir makine öğrenmesi problemi olan regresyon genelinde yaptığımız projeyi anlatacağım. Yazdığım script Türkiye'nin belirli yıllardaki nüfus oranlarını alıp gelecekte ne olacağı ile ilgili nüfus tahmin yapıyor. Verilerimi [World Bank](#)'in Türkiye datalarından aldım.

Öncelikle veride küçük ön işlemeler yaptım. Örneğin null ayıklama, veri düzenleme gibi.

```
[7]: data.dropna(inplace=True)
     data
```

[7]:

	Series Name	Series Code	Country Name	Country Code	2000 [YR2000]	2001 [YR2001]	2002 [YR2002]	2003 [YR2003]	2004 [YR2004]	2005 [YR2005]	...	2014 [YR2014]	2015 [YR2015]	2016 [YR2016]	2017 [YR2017]	2018 [YR2018]	2019 [YR2019]	2020 [YR2020]	2021 [YR2021]	[YR
0	Population, total	SP.POP.TOTL	Turkiye	TUR	65425961.0	66245128.0	67048795.0	67831730.0	68592617.0	69329557.0	...	77181884.0	78218479.0	79277962.0	80312698.0	81407204.0	82579440.0	83384680.0	84147318.0	849
1	Urban population	SP.URB.TOTL	Turkiye	TUR	42357421.0	43284567.0	44234772.0	45177289.0	46111387.0	47033171.0	...	56402205.0	57577405.0	58771924.0	59948610.0	61171815.0	62454830.0	63459911.0	64430760.0	6545

2 rows × 28 columns

```
[8]: data.drop(index=1, inplace=True)
```

```
[9]: data
```

[9]:

	Series Name	Series Code	Country Name	Country Code	2000 [YR2000]	2001 [YR2001]	2002 [YR2002]	2003 [YR2003]	2004 [YR2004]	2005 [YR2005]	...	2014 [YR2014]	2015 [YR2015]	2016 [YR2016]	2017 [YR2017]	2018 [YR2018]	2019 [YR2019]	2020 [YR2020]	2021 [YR2021]	[YR
0	Population, total	SP.POP.TOTL	Turkiye	TUR	65425961.0	66245128.0	67048795.0	67831730.0	68592617.0	69329557.0	...	77181884.0	78218479.0	79277962.0	80312698.0	81407204.0	82579440.0	83384680.0	84147318.0	849

1 rows × 28 columns

```
[11]: long_data = pd.melt(data, var_name='Year', value_name='Value')

      # Extract just the numeric year from the 'Year' column.
      long_data['Year'] = long_data['Year'].str.extract(r'(\d{4})').astype(int)

      # Display the transformed data
      print(long_data.head())
```

```
   Year       Value
0  2000  65425961.0
1  2001  66245128.0
2  2002  67048795.0
3  2003  67831730.0
4  2004  68592617.0
```

Daha sonra modelimi eğittim ve tahmin ettirdim.

```
[13]: X = long_data[['Year']]
      y = long_data['Value']
```

```
[14]: model = LinearRegression()
      model.fit(X, y)
```

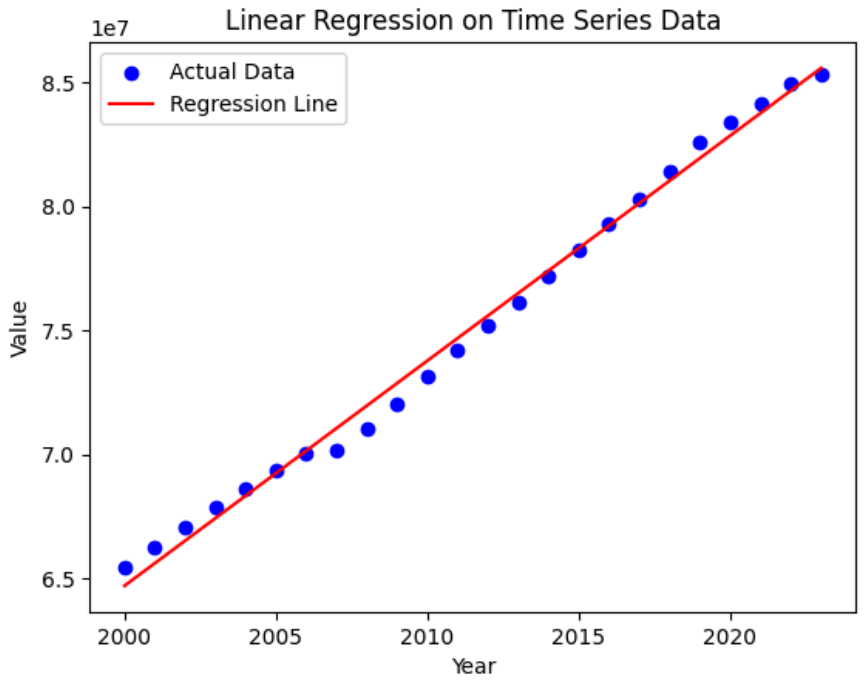
```
[14]: LinearRegression
      LinearRegression()
```

```
print("\nLinear Regression Model:")
print("Coefficient (slope):", model.coef_[0])
print("Intercept:", model.intercept_)
```

```
Linear Regression Model:
Coefficient (slope): 907862.4026086956
Intercept: -1751028851.7640579
```

```
# Predict values for the years in the data and add as a new column in the long DataFrame
long_data['Predicted'] = model.predict(X)
```

```
# Step 5: Visualize the actual data and the regression line
plt.scatter(long_data['Year'], long_data['Value'], label='Actual Data', color='blue')
plt.plot(long_data['Year'], long_data['Predicted'], label='Regression Line', color='red')
plt.xlabel('Year')
plt.ylabel('Value')
plt.title('Linear Regression on Time Series Data')
plt.legend()
plt.show()
```




```
[18]: future_years = pd.DataFrame({'Year': np.arange(2024, 2029)})
future_predictions = model.predict(future_years)
future_years['Predicted_Value'] = future_predictions / 1e6

print("\nFuture Predictions:")
print(future_years)
```

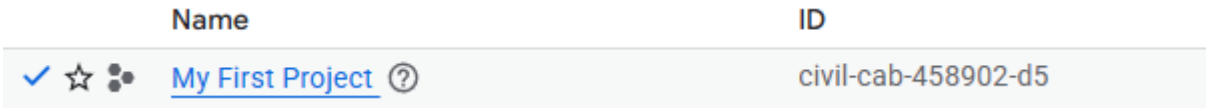
```
Future Predictions:
   Year  Predicted_Value
0  2024        86.484651
1  2025        87.392514
2  2026        88.300376
3  2027        89.208238
4  2028        90.116101
```

Ve daha sonra modelimi kaydettim. Daha sonradan cloud sdk ile kullanabilmek için de model dosyasını bucket'ımın içine kopyaladım.

```
[19]: joblib.dump(model, 'linear_model.pkl')

[19]: ['linear_model.pkl']

[22]: !gsutil cp linear_model.pkl gs://my_first_project_bucket_for_cloud/models/model.joblib

Copying file://linear_model.pkl [Content-Type=application/octet-stream]...
- [1 files][  864.0 B/  864.0 B]
Operation completed over 1 objects/864.0 B.
```

Makine öğrenmesi tarafı bu şekilde temel bir örnekti. Şimdi de cloud tarafına bakalım.

Öncelikle cloud'da ilk projemi oluşturdum ve ardından şu apileri aktifleştirdim:



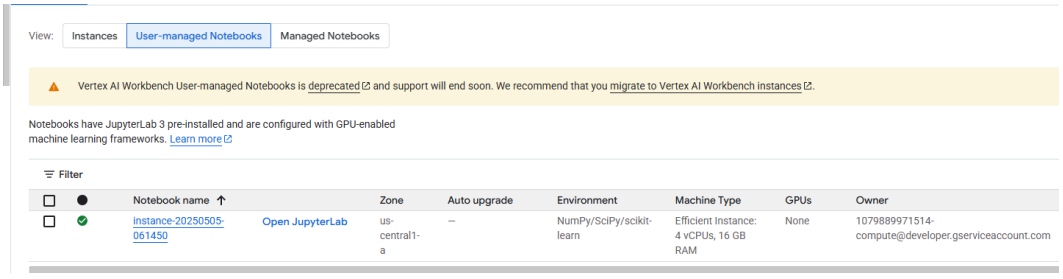

```
pictureofyouinstead@cloudshell:~ (civil-cab-458902-d5)$ gcloud services enable aiplatform.googleapis.com storage
Operation "operations/acat.p2-1079889971514-18421152-56d6-4672-8a85-92eed750e9a7" finished successfully.
```

Ve console'dan bir bucket oluşturdum:

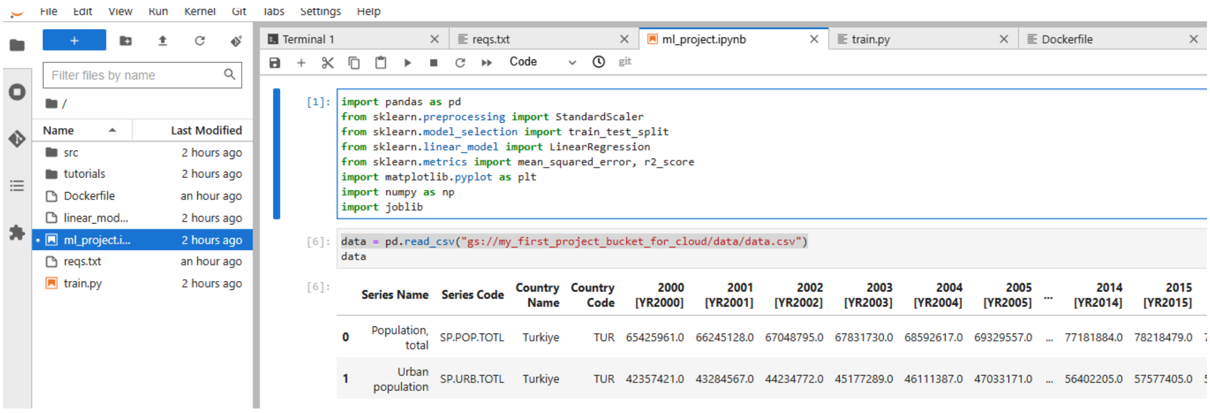
```
pictureofyouinstead@cloudshell:~ (civil-cab-458902-d5)$ gsutil mb -l europe-west1 gs://my_first_project_bucket_for_cloud
Creating gs://my_first_project_bucket_for_cloud/...
```

Sonra bucket'a data'mı yükledim ve Vertex AI'da bir notebook instance'ı oluşturup dosyalarımı oraya taşıdım.

```
C:\Users\aymin\AppData\Local\Google\Cloud SDK>gsutil cp C:\Users\aymin\Desktop\bulut\data.csv gs://my_first_project_buck
et_for_cloud/data/data.csv
Copying file://C:\Users\aymin\Desktop\bulut\data.csv [Content-Type=text/csv]...
/ [1 files][  1.1 KiB/  1.1 KiB]
Operation completed over 1 objects/1.1 KiB.
```



Ardından bu ortamda çalıştırabilmek hem veri çektiğim kısmı güncelledim ve entrypointlerle çalıştırılabilmesi için kodumun train.py haline getirdim.



Sonra train.py kodunu bağımlılıklarıyla birlikte kapsülleştirerek hem yerelde hem de Cloud Build üzerinde sorunsuz çalıştırmak için Dockerfile oluşturdum.

```
FROM python:3.10-slim
COPY train.py reqs.txt ./
RUN pip install -r reqs.txt
ENTRYPOINT ["python", "train.py"]
```

Vertex AI Custom Job'larında kullanmak üzere Docker imajımın Artifact Registry'de hazır olması gerekiyordu bu yüzden de cloud build ile imajı Artifact Registry'ye push'ladım.

```
(base) jupyter@instance-20250505-061450:~$ docker build -t europe-west1-docker.pkg.dev/civil-cab-458902-d5/ml-repo/train:latest .
[+] Building 0.4s (8/8) FINISHED                                                                              docker:default
 => [internal] load build definition from Dockerfile                                                                    0.0s
 => => transferring dockerfile: 146B                                                                                    0.0s
 => [internal] load metadata for docker.io/library/python:3.10-slim                                                     0.3s
 => [internal] load .dockerignore                                                                                       0.0s
 => => transferring context: 2B                                                                                         0.0s
 => [internal] load build context                                                                                       0.0s
 => => transferring context: 57B                                                                                        0.0s
 => [1/3] FROM docker.io/library/python:3.10-slim@sha256:57038683f4a259e17fcff1ccef7ba30b1065f4b3317dabb5bd7c82640a5ed64f 0.0s
 => CACHED [2/3] COPY train.py reqs.txt ./                                                                              0.0s
 => CACHED [3/3] RUN pip install -r reqs.txt                                                                            0.0s
 => exporting to image                                                                                                  0.0s
 => => exporting layers                                                                                                 0.0s
 => => writing image sha256:3b8c13aabbf352c458381a66afa9bdc0f4b941fc3135233945f72913770bce4c                            0.0s
 => => naming to europe-west1-docker.pkg.dev/civil-cab-458902-d5/ml-repo/train:latest                                   0.0s
(base) jupyter@instance-20250505-061450:~$ docker push europe-west1-docker.pkg.dev/civil-cab-458902-d5/ml-repo/train:latest
The push refers to repository [europe-west1-docker.pkg.dev/civil-cab-458902-d5/ml-repo/train]
6b3946be8774: Pushed
cb99e1679de5: Pushed
e695530a5684: Pushed
c2b802b98844: Pushed
dccc9f799c16: Pushed
6c4c763d22d0: Pushed
latest: digest: sha256:9b7d0d43bfdcdf94fbe2ec02fdd8cf97a7f1cb8f68388a16e687e17a3ead2b88 size: 1579
(base) jupyter@instance-20250505-061450:~$ gcloud ai custom-jobs create \
  --region=europe-west1 \
  --display-name=rf-train-job \
  --worker-pool-spec=machine-type=n1-standard-4,replica-count=1,container-image-uri=europe-west1-docker.pkg.dev/civil-cab-458902-d5/ml-repo/train:latest
Using endpoint [https://europe-west1-aiplatform.googleapis.com/]
CustomJob [projects/1079889971514/locations/europe-west1/customJobs/7616248944602906624] is submitted successfully.
```

Sonra modelimi Vertex AI Model Registry'ye yükledim.

```
(base) jupyter@instance-20250505-061450:~$ gcloud ai models upload --region=europe-west1  --display-name=serve-model  --artifact-uri=gs://my_first_project_bucket_for_cloud/models/ --container-image-uri=europe-west1-docker.
pkg.dev/civil-cab-458902-d5/ml-repo/serve:latest --container-predict-route="/predict" --container-health-route="/healthz"
Using endpoint [https://europe-west1-aiplatform.googleapis.com/]
Waiting for operation [8345659117313982464]...done.
```

Endpointleri oluşturdum.

```
(base) jupyter@instance-20250505-061450:~$ gcloud ai models upload --region=europe-west1  --display-name=serve-model  --artifact-uri=gs://my_first_project_bucket_for_cloud/models/ --container-image-uri=europe-west1-docker.
pkg.dev/civil-cab-458902-d5/ml-repo/serve:latest --container-predict-route="/predict" --container-health-route="/healthz"
Using endpoint [https://europe-west1-aiplatform.googleapis.com/]
Waiting for operation [8345659117313982464]...done.
(base) jupyter@instance-20250505-061450:~$ gcloud ai endpoints create --region=europe-west1 --display-name=rf-endpoint
Using endpoint [https://europe-west1-aiplatform.googleapis.com/]
Waiting for operation [851669337369477120]...done.
Created Vertex AI endpoint: projects/1079889971514/locations/europe-west1/endpoints/6978248457773383368.
```

Ve deploy ettim, artık dışarıdan data sağlandığında kullanılmaya hazır.

```
(base) jupyter@instance-20250505-061450:~$ gcloud ai endpoints deploy-model projects/1079889971514/locations/europe-west1/endpoints/697824845777338368 --region=europe-west1 --model=8168276280794415104 --machine-type=n1-sta
ndard-4 --min-replica-count=1 --display-name=serve-model --max-replica-count=4
Using endpoint [https://europe-west1-aiplatform.googleapis.com/]
```