

GeoAI Kullanım Dokumani

Kurulum, Konfigurasyon, Çalıştırma ve Saha Operasyonu (Genişletilmiş L^AT_EX Surumu)

GeoAI

19 Şubat 2026

Özet

Bu dokuman, GeoAI boru hattını (pipeline) kurulumdan üretim koşularına kadar uctan uca anlatır. Demo (sentetik veri) ile hızlı doğrulama, gerçek veri ile proje çalışma (`run_project.py`), ve programatik API (`GeoAIPipeline`) kullanım senaryoları kapsanır. Ayrıca çıktı dosyalarının yorumu, hata giderme, operasyon oncesi kontrol listesi ve parametre seçim rehberi sunulur.

İçindekiler

1 Hızlı Başlangıç	3
1.1 En kısa yol: demo	3
1.2 Gerçek veri ile kosu	3
1.3 Programatik API	3
2 Kurulum	3
2.1 Gereksinimler	3
2.2 Proje yapısı	4
3 Çalışma Mantığı: Girdi–Çıktı ve Skorlar	4
4 Demo Modu	4
4.1 Katman yoksa otomatik demo fallback	4
5 Gerçek Veri ile Çalışma: run_project.py	4
5.1 Temel konfigurasyon değişkenleri	4
5.2 Örnek akış	5
6 Veri Hazırlama Kuralları	5
6.1 Grid katmanları	5
6.2 Kuyu CSV formatı	5
6.3 CRS/extent uyumu için minimum kontrol	5
7 API Referansı: GeoAIPipeline	6
8 Çıktıları Yorumlama	6
8.1 Haritalar	6
8.2 Rapor metrikleri	6
9 Saha Operasyonu için Onerilen İş Akışı	6
10 Hata Giderme	7

11 Uretim Ortami Onerileri	7
12 Komut Ozeti	7
A Ayrintili Konfigurasyon Rehberi	7
B Cikti Dosya Sozlugu	8
C Operasyon Oncesi Kontrol Listesi	8
D Sik Sorulan Sorular	8

1 Hizli Baslangic

1.1 En kisa yol: demo

Kurulumun dogru calistigini doğrulamak icin demo kosusu önerilir:

```
python demo.py
```

Demo, sentetik grid katmanlari ve sentetik kuyu etiketleri uretir ve ciktiları varsayılan olarak `output/` altına yazar.

1.2 Gercek veri ile kosu

Gercek veriyle calistirmak icin:

```
python run_project.py
```

`run_project.py` icinde KATMANLAR bos ise sistem (ayar aciksa) otomatik olarak demo fallback calistirabilir (bkz. Bolum [4.1](#)).

1.3 Programatik API

Python icinden:

```
from geoai.pipeline import GeoAIPipeline

pipe = GeoAIPipeline(project_name="Saha_A")
pipe.add_layer("data/magnetic.grd", name="magnetic")
pipe.add_layer("data/gravity.grd", name="gravity")
pipe.add_wells("data/wells_mineral.csv", target_type="mineral")
results = pipe.run(target_types=["mineral"], target_nx=300, target_ny
=300)
```

2 Kurulum

2.1 Gereksinimler

GeoAI, Python tabanlidir. Tipik bagimliliklar:

Paket	Amac
numpy, scipy, pandas	Numerik hesaplama ve veri islemleri
scikit-learn	Modelleme, CV ve metrikler
matplotlib	Harita/cikti gorselleri
rasterio (onerilir)	GeoTIFF koordinatli okuma
Pillow	TIFF fallback okuma

Kurulum:

```
python -m pip install -r requirements.txt
# veya hizli kurulum:
python setup_geoai.py
```

2.2 Proje yapisi

Tipik dizin agaci:

```
geoai/
    pipeline.py
    io/loaders.py
    core/preprocessor.py
    models/prospectivity.py
    viz/maps.py
    utils/reporting.py
demo.py
run_project.py
```

3 Calisma Mantigi: Girdi–Ciktig ve Skorlar

GeoAI, her piksel konumu $\mathbf{s} = (x, y)$ icin bir ozellik vektoru $\mathbf{x}(\mathbf{s}) \in \mathbb{R}^{n_f}$ olusturur. Denetimli senaryoda hedef olasılığı:

$$P(\mathbf{s}) \approx \mathbb{P}(y(\mathbf{s}) = 1 | \mathbf{x}(\mathbf{s})).$$

Model cesitlilikinden turetilen belirsizlik (or. ensemble dagilimi):

$$U(\mathbf{s}) = \sqrt{\sum_{m=1}^M w_m (\hat{p}_m(\mathbf{s}) - P(\mathbf{s}))^2}, \quad \sum_{m=1}^M w_m = 1.$$

Operasyonel siralama icin risk-duzeltlimis skor:

$$S(\mathbf{s}) = \text{clip}(P(\mathbf{s}) - \lambda U(\mathbf{s}), 0, 1),$$

burada λ (or. 0.3) belirsizlik cezasini ayarlar.

4 Demo Modu

Demo, kurulum dogrulaması ve uctan uca test icin tasarlanmistir.

```
python demo.py
```

4.1 Katman yoksa otomatik demo fallback

`run_project.py` icinde katman listesi bos ise, belirli bir bayrak ile sistem demo fallback calistirabilir. Bu davranisin amaci, “bos konfig” durumunda bile boru hattinin smoke-test yapilabilmesidir.

5 Gercek Veri ile Calisma: `run_project.py`

5.1 Temel konfigurasyon degiskenleri

Degisken	Anlam	Tipik deger
PROJE_ADI	Rapor/ciktig adlarina yansir	Saha_A_2026
CIKTI_KLASORU	PNG/CSV/TXT cikis dizini	output/
HEDEF_TIPLERI	Problem tipi	['mineral']
KATMANLAR	Katman adi → dosya yolu	{...}
KUYU_DOSYALARI	Hedef tipi → kuyu CSV	{...}
GRID_NX, GRID_NY	Referans grid cozunurlugu	300, 300

5.2 Ornek akis

```
from geoai.pipeline import GeoAIPipeline

pipe = GeoAIPipeline(project_name="Saha_A")
pipe.add_layer("data/magnetic.grd", name="magnetic")
pipe.add_layer("data/gravity.grd", name="gravity")
pipe.add_layer("data/resistivity.tif", name="resistivity")

pipe.add_wells("data/wells_mineral.csv", target_type="mineral")

results = pipe.run(
    target_types=["mineral"],
    target_nx=300,
    target_ny=300,
    n_targets=15,
    min_prob=0.35,
    cv_folds=5,
    output_dir="output/",
    show_plots=False,
    save_plots=True,
)
```

6 Veri Hazirlama Kurallari

6.1 Grid katmanlari

Pratik kurallar:

- Tum katmanlari mumkunse ayni CRS'e (koordinat referans sistemi) getir.
- Extent (kapsam) benzer olmalı; asiri farklı extent co-registration sırasında bilgi kaybi veya NaN artısı doğurur.
- GeoTIFF için `rasterio` kullanımı önerilir (koordinat sadakatı).
- CSV/XYZ verilerinde X, Y, Z kolonları açık ve sayısal olmalıdır.
- Çok yüksek NaN oranı, hem feature kalitesini hem de eğitim örnek sayısını düşürür.

6.2 Kuyu CSV formatı

Beklenen minimal kolonlar: X, Y, LABEL. İsteğe bağlı: TARGET_TYPE, DEPTH, NOTES.

```
X,Y,LABEL,TARGET_TYPE,DEPTH,NOTES
312000,4012000,1,mineral,150,cevherli zon
320000,4018000,0,mineral,,steril
```

LABEL yalnızca {0,1} olmalıdır. Gecersiz değerlerde yükleme aşamasında hata üretilmesi beklenir.

6.3 CRS/extent uyumu için minimum kontrol

GIS tarafından hızlı kontrol:

1. Katmanların CRS'ini doğrula (EPSG kodu).
2. Extent'leri overlay ederek kuyu noktalarının katmanların içine dursunu kontrol et.
3. Büyük kaymalar varsa önce reprojection/warp uygula.

7 API Referansi: GeoAIPipeline

Metod	Ne yapar	Not
<code>add_layer(path, ...)</code>	Diskten katman ekler	Format otomatik algilanir
<code>add_layer_array(grid, x, y, name)</code>	Numpy ile katman ekler	Test/demo icin hizli
<code>add_wells(path, target_type)</code>	Kuyu CSV yukler	Kolonlar otomatik bulunabilir
<code>add_wells_dataframe(df, target_type)</code>	DataFrame ile kuyu ekler	x,y,label zorunlu
<code>run(...)</code>	Tam boru hatti	Sonuclari dict dondurur
<code>run_demo(...)</code>	Sentetik veriyle demo	Smoke test

8 Ciktiları Yorumlama

8.1 Haritalar

- **Prospectivity haritasi:** sicak renkler yüksek $P(\mathbf{s})$ değerini temsil eder.
- **Belirsizlik haritasi:** yüksek $U(\mathbf{s})$ karar kalitesinin düşük olabileceğini gösterir.
- **Guven skoru:** $S(\mathbf{s}) = P(\mathbf{s}) - \lambda U(\mathbf{s})$ ile daha risk-averse bir sıralama sağlar.

8.2 Rapor metrikleri

- **ROC-AUC:** 0.5 rastgele, 1.0 mükemmel ayırma.
- **PR-AUC:** pozitif sınıf az olduğunda daha anlamlı.
- **Brier skoru:** daha düşük değer daha iyi olasılık kalibrasyonu.
- **Optimal threshold:** sınıf karar esigini otomatik optimize eder (örn. F1 maksimize).

9 Saha Operasyonu icin Onerilen Is Akisi

Onerilen pratik sira:

1. Tüm jeofizik katmanları tek CRS'e getir.
2. Kuyu etiketlerini (0/1) saha logları ile doğrula.
3. Küçük grid ile hızlı deneme yap (örn. 200×200).
4. Metrikleri incele; gerekiyorsa katman/feature setini revize et.
5. Nihai kosusu daha yüksek çözünürlükte al (örn. 500×500).
6. Top hedef listesini jeoloji ekibi ile birleştir ve sondaj planı çıkar.

10 Hata Giderme

Belirti	Muhtemel sebep	Cozum
KATMANLAR bos hatasi	Konfig bos	Demo fallback acik birak veya katman yolunu gir
Hedef cikmiyor	Kuyu yok / az pozitif	Kuyu setini artir; gecici unsupervised ciktigii kullan
GeoTIFF koordinati bo-zuk	rasterio yok / CRS sorunlu	rasterio kur; katmani GIS ile kontrol et
CV grafikleri bos	Metrik anahtar uyumsuzlugu	Guncel isimleri kontrol et (<code>auc_roc/auc_pr</code>)
Kuyular kenara yigiliyor	CRS/extent uyumsuz	Reprojection/warp uygula; extent'i eslestir

11 Uretim Ortami Onerileri

- Kosulari `timestamp` ve `run-id` ile arsivle.
- Veri seti ve model konfigurasyonunu birlikte versiyonla (orn. Git + ciktig manifest).
- Buyuk gridler icin batch boyutlarini ve CPU/RAM kullanımını izle.
- Ciktig CSV/TXT dosyalarini QA kontrolunden gecirerek sahaya aktar.

12 Komut Ozeti

```
# Demo
python demo.py

# Gercek veri (KATMANLAR bossa otomatik demo fallback olabilir)
python run_project.py

# Regresyon testleri (varsa)
python -m unittest tests.test_regressions -v
```

A Ayrıntılı Konfigurasyon Rehberi

Aynı kod tabanı ile farklı hedef tipleri için ayrı konfigurasyon kullanılabilir. Başlangıç değerleri:

Senaryo	HEDEF_TIPLERI	Grid	min_prob
Maden arama	['mineral']	400×400	0.35
Yeraltı suyu	['groundwater']	300×300	0.30
Coklu hedef	['mineral', 'groundwater']	350×350	(opsiyonel)

B Cikti Dosya Sozlugu

Dosya	Icerik	Tipik kullanim
input_layers.png	Tum giriş katmanlari	Hizalama/NaN kontrolu
prospectiveity_<tip>.png	P, U ve skor panelleri	Hedef bolge secimi
report_<tip>.png	Hedef siralama + metrikler	Toplanti sunumu/QA
multi_target_comparison.png	Tipler arasi karsilastirma	Kaynak dagitimi
targets.csv	Koordinatli hedef listesi	GIS/CAD aktarimi
report.txt	Metrik ve hedef ozet metni	Arsiv/izlenebilirlik

C Operasyon Oncesi Kontrol Listesi

- run_project.py icindeki dosya yollari dogru mu?
- Kuyu CSV'sinde `x, y, label` kolonlari dogru ve 0/1 etiketli mi?
- output/ dizininde yazma izni var mi?
- Grid cozunurlugu (`nx, ny`) donanim kapasitesine uygun mu?
- Son kosu metrikleri (PR-AUC, Brier) kabul esiginde mi?

D Sik Sorulan Sorular

Soru	Kisa cevap
Kuyu verisi yoksa ne olur?	Pipeline, unsupervised (anomaly tabanli) fallback ile harita uretir.
Neden PR-AUC onemli?	Pozitifler az oldugunda ROC-AUC tek basina yaniltici olabilir.
Neden hedef sayisi sinirli?	Bagli bilesen + minimum mesafe filtreleri tekrarlari azaltir.
Koordinatlar neden kaymis?	CRS/extent tutarsizligi co-registration kalitesini bozar.