

data.py Python



recal_test.py

Python

tamam şimdi sana vereceğim scriptleri satır satır konuya hakim olmayan birine anlatır gibi anlat moduler sistemde nerde ne olduğunu falanda

Süper, şimdi sana bu üç scripti (data.py, boob_monster.py, recal_test.py) hem satır satır hem de modüler sistem mantığıyla açıklayacağım.

Amaç: konuya hiç hakim olmayan biri bile neyin nerede ne işe yaradığını anlasın.

🧱 MODÜLER SİSTEM MİMARİSİ

Modül	Dosya	Görevi
🛂 Veri hazırlama	data.py	Ham veriyi al, temizle, . csv yap
Model eğitimi	boob_monster.py	Veriyi al, eğit, başarıyı ölç, kaydet
Nodel testi	recal_test.py	Kaydedilen modeli al, test verisinde dene

🔍 1. data.py – Veri Hazırlama Aşaması

Bu dosya, UCI'den veriyi indirir ve modelin kullanabileceği şekilde .csv dosyasına yazar.

```
from ucimlrepo import fetch_ucirepo # UCI veri setini otomatik indir
data = fetch_ucirepo(id=17) # 17 numaralı veri seti: meme kanseri

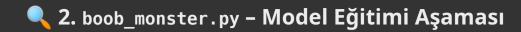
x = data.data.features # Özellikler (30 sütun)
y = data.data.targets.replace({'B': 0, 'M': 1}) # Etiketleri sayıya çevir (0 = iyi, 1 = kötü)

df = x.copy() # x'ten yeni tablo oluştur
df["Diagnosis"] = y # Etiketleri sona ekle

df.to_csv("data/veri.csv", index=False) # CSV dosyasına kaydet
```

礡 Üretilen dosya:

```
bash
data/veri.csv
```



🖰 Bu dosya modeli eğitir, başarıyı değerlendirir ve modeli .pkl dosyası olarak kalıcı hale getirir.

```
python
df = pd.read csv("data/veri.csv")
                                      # Hazırladığın CSV dosyasını al
x = df.drop("Diagnosis", axis=1)
                                       # Etiket dısındaki 30 sütun
y = df["Diagnosis"]
                                       # Etiket sütunu (0 veya 1)
X_train, X_test, y_train, y_test = train_test_split(...) # Veriyi %80 eğitim, %20 test olarak
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
model = LogisticRegression()
                                      # Basit, ama güçlü bir model seç
model.fit(X_train_scaled, y_train) # Eğit
y_pred = model.predict(X_test_scaled) # Test verisiyle tahmin yap
print(confusion_matrix(y_test, y_pred)) # Ne kadar doğru tahmin etmiş
print(classification_report(y_test, y_pred)) # Precision, recall, F1-score
# Eğitilen her şeyi kaydet
joblib.dump(model, "model/boob_monster.pkl")
                                                  # Model
joblib.dump(scaler, "modet/boob_monster.pkl")
joblib.dump(X_test, "data/X_test.pkl")
joblib.dump(y_test, "data/y_test.pkl")
                                                  # Veriyi nasıl ölçeklediğini
                                                  # Test etiketleri
```

🧳 Üretilen dosyalar:

```
bash

model/boob_monster.pkl
model/bm_scaler.pkl
data/X_test.pkl
data/y_test.pkl
```

3. recal_test.py - Modeli Geri Çağırıp Test Etme

Bu dosya, daha önce eğitilmiş modeli ve scaler'ı kullanarak test verisine tahmin yaptırır ve doğruluğunu ölçer.

```
model = joblib.load("model/boob_monster.pkl")  # Eğitilen modeli al
scaler = joblib.load("model/bm_scaler.pkl")  # Aynı ölçekleme kurallarını al

X_test = joblib.load("data/X_test.pkl")  # Test verisi
y_test = joblib.load("data/y_test.pkl")  # Gerçek etiketler

X_test_scaled = scaler.transform(X_test)  # Test verisini eğitimdeki gibi ölçekle
y_pred = model.predict(X_test_scaled)  # Tahmin et

print(confusion_matrix(y_test, y_pred))  # Hangi sınıflarda doğru/yanlış tahmin var
print(classification_report(y_test, y_pred))  # Detaylı başarı raporu
```

📌 Özet: Bu Sistemin Akışı

```
bash

data.py → veriyi hazırlar ve kaydeder

boob_monster.py → modeli eğitir, performansı <mark>test</mark> eder, her şeyi kaydeder

recal_test.py → modeli geri çağırır, yeniden <mark>test</mark> eder
```

Bu yapı modülerdir:

- Her dosya sadece bir iş yapar.
- data.py çalıştırılmadan boob_monster.py çalışmaz.
- boob_monster.py olmadan recal_test.py test yapamaz.

İstersen şimdi bu sistemin içine svm_model.py ya da random_forest.py gibi yeni modeller ekleyerek genişletebiliriz.

Nasıl devam edelim?