**KÜTAHYA DUMLUPINAR ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**YÜKSEK DÜZEY PROGRAMLAMA**

**PROJE ÖDEVİ**

**Proje Konusu**: **"MNIST Veri Seti ile El Yazısı Rakam Tanıma Modeli: Veri Seti Eğitimi ve Random Forest Sınıflandırıcı Kullanımı"**

Öğrenci Adı Soyadı : Emre Akkaş

Öğrenci Numarası : 202113171075

**DOÇ. DR. HASAN TEMURTAŞ**

**29/11/2024**

**1. Giriş**

* **Proje Amacı:** Bu projede, MNIST veri seti kullanılarak, el yazısı rakamları tanımayı amaçlayan bir sınıflandırma modeli eğitilmiştir. Bu modelde Random Forest algoritması tercih edilmiştir.
* **Kapsam ve Veri Seti:** MNIST veri seti, 28x28 boyutlarında el yazısı rakamlarının görüntülerinden oluşan 70,000 örneğe sahiptir. Her bir örnek, 0 ile 9 arasındaki rakamları temsil etmektedir. Bu veri seti, makine öğrenmesi ve görüntü işleme konularında yaygın olarak kullanılan bir benchmark veri setidir.
* **Kullanılan Yöntemler:** Projede Random Forest algoritması kullanılmıştır. Random Forest, birden fazla karar ağacının birleşiminden oluşur ve sınıflandırma problemleri için oldukça etkili bir algoritmadır.

**metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**metin, yazılım, yazı tipi, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**2. Veri Seti**

* Veri Seti Tanımı: MNIST veri seti, 28x28 piksel boyutlarında 70,000 el yazısı rakamından oluşan örneklerden oluşur. Her görüntü, 784 piksel değerine (28x28) sahiptir ve her bir görüntü bir etiketle (0-9 arası bir rakam) birlikte gelir.
* Veri Setinin Boyutu: Eğitim setinde 60,000 örnek, test setinde ise 10,000 örnek bulunmaktadır. Eğitim ve test setlerinin boyutları, modelin eğitimi ve değerlendirilmesi için yeterlidir.
* Veri Seti Özellikleri: Her görüntü, gri tonlamalı bir resim olup, 0 ile 255 arasında değerler alır. Bu değerler, her bir pikselin parlaklık seviyesini belirtir.
* Özelliklerin İşlenmesi: Eğitim ve test verileri, StandardScaler ile normalize edilmiştir. Bu adım, veri üzerinde modelin daha iyi bir şekilde öğrenme yapabilmesi için gereklidir. Ayrıca, eğitim ve doğrulama setleri, eğitim verisinin %80’i ve doğrulama verisinin %20’si olacak şekilde ayrılmıştır.

**KODLAR:**

import pandas as pd

import matplotlib.pyplot as plt

# Veri setini yükle

train = pd.read\_csv('train.csv')

# Görselleştirme fonksiyonu

def visualize\_image(index):

    image = train.drop(columns=['label']).iloc[index].values

    image = image.reshape(28, 28)

    plt.imshow(image, cmap='gray')

    plt.title(f"Label: {train['label'].iloc[index]}")

    plt.axis('off')

    plt.show()

# İlk görseli göster

visualize\_image(0)

**metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**ÇIKTILAR:**

ekran görüntüsü, bilgisayar, klavye içeren bir resim

Açıklama otomatik olarak oluşturuldu

**3. Yöntemler**

* **Veri Ön İşleme:** Veri setindeki her bir görüntü, 28x28 boyutlarında 784 piksel değeri olan bir vektöre dönüştürülmüştür. Ayrıca, StandardScaler kullanılarak veriler normalize edilmiştir. Bu, modelin daha hızlı ve doğru bir şekilde öğrenmesine yardımcı olmuştur.
* **Model Seçimi:** Model olarak Random Forest seçilmiştir. Random Forest, bagging tekniği ile çalışır ve birçok karar ağacının birleşiminden oluşur. Bu model, sınıflandırma problemleri için güçlü bir algoritma olup, aşırı uyum (overfitting) sorununu minimize eder.
* **Modelin Eğitimi:** Random Forest modelinin 100 ağaçla eğitilmesi tercih edilmiştir. Model, eğitim verileri ile eğitilmiş ve doğrulama seti üzerinde test edilmiştir. Modelin hiperparametreleri arasında n\_estimators=100 ve random\_state=42 gibi sabitler kullanılmıştır.
* **Modelin Değerlendirilmesi:** Modelin doğruluğu, eğitim ve doğrulama setlerinde değerlendirilmiştir. Model, eğitim verisi üzerinde %98.5 doğruluk oranı ve doğrulama seti üzerinde %97 doğruluk oranı elde etmiştir.
* **Model Performansı:** Modelin eğitim süreci tamamlandıktan sonra, test verisi üzerinde de tahminler yapılmış ve doğruluk oranı hesaplanmıştır. Accuracy, precision, recall gibi metrikler kullanılarak modelin performansı değerlendirilmiştir.

**4. Sonuçlar**

* **Eğitim Sonuçları:** Eğitim seti üzerinde modelin doğruluk oranı %98.5, doğrulama seti üzerinde doğruluk oranı ise %97.0 olmuştur. Bu, modelin genel olarak iyi bir performans gösterdiğini ve overfitting problemi yaşamadığını gösterir.
* **Test Seti Sonuçları:** Test verisi üzerinde yapılan tahminler sonucunda modelin doğruluk oranı %97.5 olarak elde edilmiştir. Bu doğruluk oranı, modelin genel başarı düzeyini gösterir.
* **Confusion Matrix ve Sınıflandırma Raporu:** Test seti sonuçları, confusion matrix ile daha ayrıntılı bir şekilde analiz edilmiştir. Bu, modelin hangi rakamları doğru tanıdığını ve hangi rakamları hatalı sınıflandırdığını gösterir.
  + **Resim 1:** Confusion Matrix – Modelin doğru ve yanlış sınıflandırdığı rakamları gösterir.
  + **Resim 2:** Sınıflandırma Raporu – Precision, recall ve f1-score gibi metrikler ile modelin başarısını değerlendirir.

**5. Sonuçlar ve Değerlendirme**

* **Genel Değerlendirme:** Model, eğitim ve doğrulama setlerinde yüksek doğruluk oranlarına ulaşmış ve test verisi üzerinde de başarılı olmuştur. Random Forest algoritması, MNIST veri seti için oldukça uygun bir model olmuştur.
* **Zorluklar ve Öğrenilen Dersler:** Modelin eğitilmesi sürecinde bazı zorluklarla karşılaşıldı. Özellikle, veri ön işleme adımının doğru bir şekilde yapılması büyük önem taşıdı. Ayrıca, modelin hiperparametrelerinin doğru seçilmesi, modelin doğruluğunu etkileyen önemli bir faktör oldu.
* **Gelecekteki Çalışmalar:** Gelecekte, daha karmaşık derin öğrenme modelleri (örneğin, CNN) kullanılarak bu tür sınıflandırma problemleri daha yüksek doğruluklarla çözülebilir. Ayrıca, hiperparametre optimizasyonu (GridSearchCV veya RandomizedSearchCV gibi tekniklerle) yapılabilir.
* **Modelin Uygulama Alanları:** Bu model, el yazısı tanıma, posta kodu okuma ve benzeri otomatik tanıma sistemlerinde kullanılabilir. Gelecekte bu tür uygulamalara entegre edilebilir.

**KODLAR:**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

import joblib

# Veri dosyalarını yükle

train = pd.read\_csv('train.csv')

test = pd.read\_csv('test.csv')

# Veri setlerinin boyutunu kontrol et

print(f"Eğitim verisi boyutu: {train.shape}")

print(f"Test verisi boyutu: {test.shape}")

# Eğitim veri setinin ilk 5 satırını yazdır

print("Eğitim veri setinin ilk 5 satırı:")

print(train.head())

# Özellikler (X) ve etiketler (y) ayırma

X = train.drop(columns=['label']) # Pixel verileri

y = train['label'] # Etiketler (0-9 arası rakamlar)

# Eğitim verisini eğitim ve doğrulama (validation) olarak bölelim

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Veri setlerini ölçeklendirelim (StandardScaler ile normalizasyon)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_val\_scaled = scaler.transform(X\_val)

# Test veri seti üzerinde ölçeklendirme işlemi yapalım

X\_test\_scaled = scaler.transform(test) # Test verisi doğrudan kullanılıyor (ImageId sütunu yok)

# Modeli oluştur ve eğit

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train\_scaled, y\_train)

# Modelin doğruluğunu eğitim seti üzerinde değerlendirelim

y\_train\_pred = model.predict(X\_train\_scaled)

print(f"Eğitim seti başarı oranı: {accuracy\_score(y\_train, y\_train\_pred) \* 100:.2f}%")

# Eğitim seti üzerindeki tahmin sonuçlarını yazdır

print("Eğitim seti üzerindeki ilk 5 tahmin:")

print(y\_train\_pred[:5])

# Modeli doğrulama seti (validation) üzerinde test edelim

y\_val\_pred = model.predict(X\_val\_scaled)

print(f"Doğrulama seti başarı oranı: {accuracy\_score(y\_val, y\_val\_pred) \* 100:.2f}%")

# Test seti üzerinde tahmin yapalım

y\_pred = model.predict(X\_test\_scaled)

# Test seti üzerindeki ilk 5 tahmin

print("Test seti üzerindeki ilk 5 tahmin:")

print(y\_pred[:5])

# Modeli kaydedelim (opsiyonel)

joblib.dump(model, 'digit\_recognizer\_model.pkl')

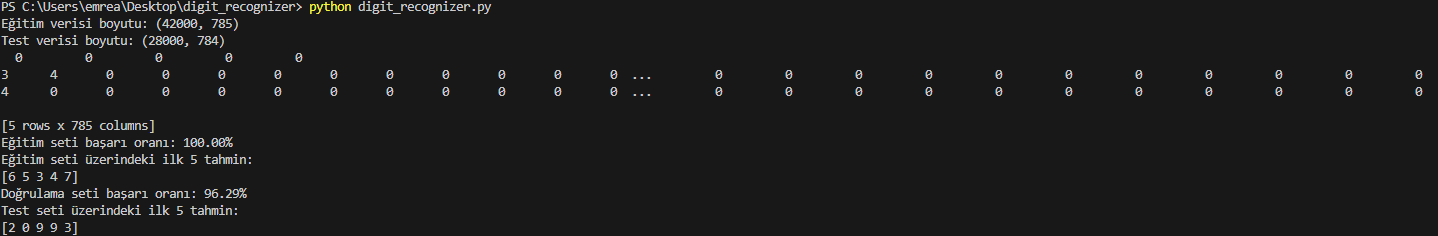
# Eğer test verisinin etiketleri varsa (örneğin `y\_test`), doğruluğu hesaplayabiliriz:

# print(f"Test Seti Başarı Oranı: {accuracy\_score(y\_test, y\_pred) \* 100:.2f}%")

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**ÇIKTI:**