

T.C.  
KÜTAHYA DUMLUPINAR ÜNİVERSİTESİ  
Mühendislik Fakültesi

**DIGIT RECOGNIZER**

**ESLEM OLUK**

**202113171028**

**YÜKSEK DÜZEY PROGRAMLAMA**

**PROJE ÖDEVİ**

**DOÇ.DR HASAN TEMURTAŞ**

KÜTAHYA, 2024

## Projenin Amaç ve Kapsamı

Bu projenin amacı, el yazısı ile yazılmış rakamları otomatik olarak tanıyabilen bir makine öğrenimi modeli geliştirmektir. El yazısı tanıma, günümüzde dijital belgelerin işlenmesi, posta kodlarının sınıflandırılması, fatura işlemleri ve dijital eğitim materyallerinde sıklıkla kullanılan bir teknolojidir. Bu proje, MNIST veri setinden türetilmiş bir veri kümesi kullanarak, rakam sınıflandırma sorununa bir çözüm sunmayı hedeflemiştir.

## Projede Kullanılan Model

Bu projede, el yazısı rakamların sınıflandırılması için **Çok Katmanlı Perceptron (MLP)** modeli tercih edilmiştir. MLP, denetimli öğrenme algoritmaları ile çalışan ve özellikle sınıflandırma ve regresyon problemlerinde sıkça kullanılan bir yapay sinir ağı türüdür.

Proje kapsamında kullanılan MLP modeli şu özelliklere sahiptir:

1. **Giriş Katmanı**:
   * Modelin giriş verisi, 28x28 pikselden oluşan MNIST el yazısı rakamlarından oluşmaktadır.
   * Bu görüntüler düzleştirilerek (flattening) 784 boyutlu bir giriş vektörüne dönüştürülmüştür.
2. **Gizli Katman**:
   * Modelde bir adet gizli katman bulunmaktadır.
   * Bu katman, giriş verilerini işlerken **ReLU (Rectified Linear Unit)** aktivasyon fonksiyonunu kullanmaktadır.
   * Ağırlıklar ve önyargılar, modelin öğrenme sürecini optimize edecek şekilde başlatılmıştır.
3. **Çıkış Katmanı**:
   * Çıkış katmanı, 0'dan 9'a kadar toplam 10 sınıfı temsil eden 10 nörondan oluşmaktadır.
   * Çıkışlar, **SoftMax aktivasyon fonksiyonu** yardımıyla sınıf olasılıklarına dönüştürülmüştür.

## Kodlar ve Çıktıları

**Görüntü Toplama:** Öncelikle rakamların (0'dan 9'a kadar) görüntülerini toplamamız gerektirir. Görüntüleri toplamak için pyscreenshot paketini kullandım. Bu paket pip, Python Paket Yükleyicisi ile indirilebilir. Bu pip, Python yüklendiğinde otomatik olarak yüklenir. Pyscreenshot'ı yüklemek için terminali açtım ve şu komutu girdim :

pip install pyscreenshot

Ayrıca bu kod, pyscreenshot kütüphanesiyle ekranın belirli bir bölgesinin görüntüsünü alır ve her 8 saniyede bir kaydeder. Ekran görüntüleri, belirtilen koordinatlar arasından alınarak , belirlenen klasöre artan numaralarla .png formatında kaydedilir. Döngü, 100 kez çalışır ve her kaydın ardından ekranın temizlenmesi gerektiği belirtilir.

metin, ekran görüntüsü, yazı tipi, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Veri Setini Oluşturma :** Bu kodda, el yazısı rakamlarının bulunduğu görüntüleri işleyerek bir veri kümesi oluşturup bu veriyi CSV formatında kaydettim. İlk olarak, her bir görüntüyü temsil eden 784 piksel değerini ve etiketini içerecek şekilde başlıkları belirledim. Başlıklar arasında "label" ve her piksel için "pixel0", "pixel1", ... "pixel783" gibi sütunlar bulunuyor. Daha sonra, her rakam için (0-9) ilgili görüntü dosyalarını belirledim ve her etiket için o etikete ait tüm görüntüleri işleyerek, her birini 28x28 boyutlarına küçültüp gri tonlamalı hale getirdim. Görüntü üzerinde gürültü azaltma için bulanıklaştırma işlemi gerçekleştirdim ve ardından her bir pikseli 0 ya da 1 olarak sınıflandırarak (beyaz olan pikselleri 1, siyah olanları ise 0 olarak) bir veri dizisi oluşturup CSV dosyasına kaydettim. Bu işlem sayesinde, her bir görüntüdeki piksel verileri ve etiketleri CSV formatında birleştirerek bir veri kümesi oluşturmuş oldum.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Veri Setini Yükleme :** Daha sonra veri setini açıp karıştırmamız, yani her bir veri satırının pozisyonunu değiştirmemiz ve görüntülememiz gerekiyor.

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

Açıklama otomatik olarak oluşturuldu

**Modeli Eğitme ve Doğruluğunu Hesaplama :** Modelimizi eğitmemiz ve doğruluğunu hesaplamamız gerekir. Modelimizi eğitmek için scikit-learn'ü kullandım. Scikit-learn'ü yüklemek için şu komutu çalıştırdım:

pip install scikit-learn

**Bağımlı ve bağımsız değişkenleri ayırma:** Daha sonra bağımlı (Y) ve bağımsız değişkeni (X) ayırmamız gerekir. Piksel değeri (0 ile 1 arasında) bağımsız değişkenimiz olacaktır. Rakamımız (0 ile 9 arasında) bağımlı değişkenimiz olacaktır.



**Matplotlib kullanarak bir görüntünün önizlemesi :** Bu aşamada ise belirli bir görüntüyü 28x28 boyutuna getirip etiketini aldım. Ardından, plt.imshow(img) komutuyla görüntüyü ekranda gösterdim. Bu işlemle, belirli bir görüntüyü ve etiketini görselleştirdim.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, diyagram, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Daha sonra, veri kümesini eğitim ve test setlerine böldüm. train\_test\_split(X, Y, test\_size=0.2) komutuyla, X (özellikler) ve Y (etiketler) veri setlerini, %80 eğitim ve %20 test verisi olacak şekilde rastgele ayırdım. train\_x ve train\_y, eğitim setindeki özellikler ve etiketler, test\_x ve test\_y ise test setindeki özellikler ve etiketlerdir.

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Ardından SVC sınıflandırıcı modeli oluşturup, eğitim verisiyle eğittim. Eğitilen modeli joblib kullanarak bir dosyaya kaydettim.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Modelin Doğruluğunu Hesaplama** : Eğitilen sınıflandırıcı modelini test verisi üzerinde kullanarak tahminler yaptım. classifier.predict(test\_x) komutuyla test verilerine karşı tahminler elde ettim. Ardından, metrics.accuracy\_score fonksiyonuyla modelin doğruluk oranını hesapladım ve ekrana yazdırdım.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Verilerimin %80'ini eğitim kısmına, kalan %20'sini ise test kısmına veriyorum (test\_size = 0.2 olarak tanımladığım gibi). Eğitim görüntüleri, modelimizi oluşturmak için, test görüntüleri ise doğruluğu hesaplamak için kullanılır. Eğitim kısmında, modelimizi eğitiyoruz, yani modelimize piksel değeri (0 ve 1'lerden oluşan bir grup) ve ayrıca etiket (yani bu rakamın hangisi olduğunu) veriyoruz. Bu adımdan sonra modelimiz eğitilmiş oluyor. Şimdi, test kısmında, modelimize yalnızca piksel değeri veriyoruz, modelimiz bu rakamı tahmin etmek zorunda. Modelimizin ne kadar doğru cevap döndürdüğünü sayıyoruz. Bu şekilde doğruluğu hesaplıyoruz. Doğruluğumuz 0,97 ise, bu 100 veriden 97'sinin doğru tahmin edildiği ve 3'ünün yanlış tahmin edildiği anlamına gelir.

**Paint ile Çizilen Görüntünün Tahmini :** Bu aşamada **i**lk olarak, daha önce eğitilen modelimi yükledim ve kullanıcı tarafından çizilen rakamın bulunduğu ekran bölgesinin görüntüsünü aldım. Alınan bu görüntüyü gri tonlamaya dönüştürdüm ve gürültüyü azaltmak için bulanıklaştırdım. Ardından, resmi ikili hale getirip, modelin eğitimi için uygun olan 28x28 piksel boyutuna yeniden boyutlandırdım. Görüntüdeki her pikselin değeri 0 veya 1 olarak kaydedilip, tahmin yapılmak üzere modelime verildi. Modelim, bu veriye dayanarak tahmin yapıp, tahmin edilen rakamı ekrana yazdırdı. Son olarak, işlem yaptığım resim ve tahmin, ekranda gösterildi. Bu işlem, kullanıcı Enter tuşuna basana kadar devam edecek.

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

Açıklama otomatik olarak oluşturuldu**

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

Açıklama otomatik olarak oluşturuldu**