## **PORJE2: IMAGE CLASSİFİCATİON(GÖRSEL SINIFLANDIRMA)**

**RAPOR HAZIRLAYAN**

**FURKAN DEMİR**

mport os

import pandas as pd

import requests

from sklearn.calibration import label\_binarize

from tqdm import tqdm

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

import torch

import torch.nn as nn

import torch.optim as optim

import torchvision

import torchvision.transforms as transforms

from torchvision import models, datasets

from torch.utils.data import DataLoader

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

from sklearn.metrics import confusion\_matrix, classification\_report, roc\_curve, auc

Bu çalışmada, veri işleme, model eğitimi ve değerlendirme süreçlerinde kullanılan önemli kütüphaneler aşağıda özetlenmiştir:

## **1. Veri İşleme Kütüphaneleri**

* **os**: Dosya ve dizin işlemleri için kullanılır.
* **pandas**: Veri çerçeveleri (DataFrame) oluşturmak ve manipüle etmek için kullanılır.
* **requests**: İnternet üzerinden veri çekmek için HTTP istekleri yapmaya yarar.
* **label\_binarize** (sklearn): Etiketlerin ikili (binary) hale getirilmesini sağlar.

## **2. Makine Öğrenmesi ve Model Eğitimi Kütüphaneleri**

* **sklearn.model\_selection (train\_test\_split)**: Veriyi eğitim ve test setlerine ayırmak için kullanılır.
* **torch & torch.nn**: PyTorch kütüphanesi, derin öğrenme modelleri oluşturmak ve eğitmek için kullanılır.
* **torch.optim**: PyTorch için optimizasyon algoritmaları içerir.
* **torchvision & torchvision.transforms**: Görüntü işleme ve veri artırma (augmentation) işlemleri için kullanılır.
* **torch.utils.data.DataLoader**: Veri kümelerini yüklemek ve işlemek için veri yükleyicidir.

## **3. Görselleştirme Kütüphaneleri**

* **matplotlib.pyplot**: Grafikler ve veri görselleştirme için kullanılır.
* **numpy**: Sayısal hesaplamalar için kullanılır.
* **seaborn**: İstatistiksel veri görselleştirme araçları sunar.

## **4. Model Değerlendirme Kütüphaneleri**

* **sklearn.metrics (confusion\_matrix, classification\_report, roc\_curve, auc)**
  + **confusion\_matrix**: Modelin tahmin performansını gösteren hata matrisi oluşturur.
  + **classification\_report**: Doğruluk, hassasiyet (precision), duyarlılık (recall) gibi metrikleri içerir.
  + **roc\_curve & auc**: ROC eğrisi ve AUC skoru hesaplamak için kullanılır.

Bu kütüphaneler birlikte kullanılarak bir **makine öğrenmesi veya derin öğrenme modeli eğitmek, test etmek ve değerlendirmek** için gerekli araçlar sağlanmaktadır.

### **Veri Dosyalarının Kullanımı – Özet Rapor**

Bu kod parçası, farklı kategorilere ait veri dosyalarının dosya yollarını bir **sözlük (dictionary)** içinde tanımlamaktadır. Aşağıda, kodun amacı ve işleyişi özetlenmiştir:

# Veri dosyaları

file\_paths = {

"All\_Beauty": "/content/drive/MyDrive/Colab\_Notebooks/aws/meta\_All\_Beauty.jsonl",

"Digital\_Music": "/content/drive/MyDrive/Colab\_Notebooks/aws/meta\_Digital\_Music.jsonl",

"Health\_and\_Personal\_Care": "/content/drive/MyDrive/Colab\_Notebooks/aws/meta\_Health\_and\_Personal\_Care.jsonl",

}

### **Veri Dosyalarının Kullanımı – Özet Rapor**

Bu kod parçası, farklı kategorilere ait veri dosyalarının dosya yollarını bir **sözlük (dictionary)** içinde tanımlamaktadır. Aşağıda, kodun amacı ve işleyişi özetlenmiştir:

## **1. Amaç ve Kullanım**

Kod, **Amazon verilerine** ait meta bilgilerin bulunduğu JSONL formatındaki dosyaları belirlenen dizinlerden okumak için kullanılır.

## **2. Veri Dosyaları ve İçerikleri**

Kod, üç farklı veri kümesini içerir:

* **All\_Beauty** → Güzellik kategorisindeki ürünlerin meta verileri
* **Digital\_Music** → Dijital müzik ürünlerine ait meta veriler
* **Health\_and\_Personal\_Care** → Sağlık ve kişisel bakım ürünlerinin meta verileri

Bu veri dosyaları, Google Colab ortamındaki **Google Drive** dizini altında saklanmaktadır.

## **3. Teknik Detaylar**

* **file\_paths** sözlüğü, anahtar (kategori adı) ve değer (dosya yolu) çiftleri içerir.
* Dosya formatı **.jsonl** olup, **JSON Lines (her satır bir JSON nesnesi içerir)** biçimindedir.
* Verilere erişim, dosya yolları kullanılarak sağlanacaktır.

## **4. Sonuç ve Kullanım Alanı**

Bu kod, ilgili veri dosyalarının dinamik olarak işlenmesini sağlamak için veri yükleme ve analiz süreçlerinde kullanılacaktır. Veri dosyaları, **makine öğrenmesi modelleri için özellik mühendisliği, analiz ve görselleştirme işlemlerine** temel oluşturacaktır.

# Veri seti klasörleri

base\_dir = "/content/drive/MyDrive/Colab\_Notebooks/aws/data"

train\_dir = os.path.join(base\_dir, "train")

test\_dir = os.path.join(base\_dir, "test")

os.makedirs(train\_dir, exist\_ok=True)

os.makedirs(test\_dir, exist\_ok=True)

### **Veri Seti Klasör Yapısı – Özet Rapor**

Bu kod parçası, **Google Colab** ortamında kullanılan veri setlerinin düzenlenmesi ve saklanması için gerekli dizinleri oluşturur.

## **1. Amaç ve Kullanım**

Bu kod, makine öğrenmesi veya derin öğrenme modeli eğitimi için kullanılacak **eğitim (train) ve test (test) veri setlerinin** saklanacağı dizinleri oluşturmayı amaçlamaktadır.

## **2. Teknik Açıklama**

* **base\_dir**: Ana veri dizininin yolu belirlenir (/content/drive/MyDrive/Colab\_Notebooks/aws/data).
* **train\_dir**: Eğitim verilerinin saklanacağı klasör yolu oluşturulur.
* **test\_dir**: Test verilerinin saklanacağı klasör yolu belirlenir.
* **os.makedirs(path, exist\_ok=True)**: Eğer klasörler mevcut değilse, **oluşturulur**. Eğer zaten varsa, hata alınmaz.

## **3. Sonuç ve Kullanım Alanı**

Bu yapı, veri setlerini **eğitim ve test** olarak düzenli bir şekilde saklamaya yardımcı olur. Bu, makine öğrenmesi modellerinin doğru veri kümeleriyle eğitilmesini ve değerlendirilmesini sağlar.

# 'large' etiketli görselleri filtreleyen fonksiyon

def extract\_large\_images(image\_list):

if isinstance(image\_list, list):

large\_images = [img.get("large") for img in image\_list if isinstance(img, dict) and "large" in img]

return large\_images if large\_images else None

return None

# Görselleri indirme fonksiyonu

def download\_images(df, save\_path, category\_name, num\_images=1000):

os.makedirs(save\_path, exist\_ok=True)

images\_downloaded = []

i = 0

for image\_list in tqdm(df["large\_images"].dropna(), desc=f"Downloading {category\_name} Images"):

if not isinstance(image\_list, list):

continue

for link in image\_list:

if not link:

continue

try:

img\_data = requests.get(link, timeout=5).content

img\_name = f"{category\_name}\_{i+1}.jpg"

img\_path = os.path.join(save\_path, img\_name)

with open(img\_path, "wb") as img\_file:

img\_file.write(img\_data)

images\_downloaded.append(img\_name)

i += 1

if i >= num\_images:

return images\_downloaded

except requests.RequestException:

continue

return images\_downloaded

### **Görsel İşleme ve İndirme Fonksiyonları – Özet Rapor**

Bu kod parçası, bir veri kümesindeki **"large" etiketli görselleri** filtreleyerek belirli bir dizine indirmek için iki temel fonksiyon içermektedir.

## **1. Fonksiyon Açıklamaları**

### **1.1** extract\_large\_images(image\_list) – **Büyük Görselleri Filtreleme**

Bu fonksiyon, verilen bir görsel listesinden "large" anahtarına sahip olanları filtreleyerek bir liste olarak döndürür.

#### **Çıktı:**

* Eğer "large" etiketli görseller varsa, bunların URL’lerini içeren bir liste döndürülür.
* Eğer uygun görseller bulunamazsa, **None** döndürülür.

### **1.2** download\_images(df, save\_path, category\_name, num\_images=1000) – **Görselleri İndirme**

Bu fonksiyon, verilen bir **veri çerçevesi (DataFrame)** içindeki "large\_images" sütunundan **büyük boyutlu görselleri**indirir ve belirtilen klasöre kaydeder.

#### **Çalışma Mantığı:**

* **Klasör oluşturma:** Kayıt klasörü (save\_path) yoksa oluşturulur.
* **Veri çerçevesindeki görseller üzerinde döngü:**
  + "large\_images" sütunundaki her bir görsel listesi işlenir.
  + **Geçersiz linkler atlanır.**
  + **İstek atılır:** Her link için HTTP isteği (requests.get) gönderilir.
  + **Görseller kaydedilir:** Başarılı yanıt dönerse, görsel belirtilen dizine .jpg formatında kaydedilir.
  + **Hata yönetimi:** Bağlantı hatası (RequestException) oluşursa, kod hata vermeden devam eder.
  + **İndirme limiti:** Maksimum **num\_images** kadar görsel indirildikten sonra işlem durur.

#### **Çıktı:**

* Başarıyla indirilen görsellerin dosya adlarını içeren bir liste döndürülür.
* Eğer hiç görsel indirilemezse, boş bir liste döndürülür.

## **2. Kullanım Alanları ve Sonuç**

* **Bu kod, büyük boyutlu ürün görsellerini filtreleyip indirmek için kullanılmaktadır.**
* **E-ticaret, makine öğrenmesi, bilgisayarla görü (computer vision) gibi alanlarda kullanılabilir.**
* **Hata yönetimi mekanizması sayesinde bağlantı problemleri oluştuğunda işlem durmadan devam eder.**

# Veri işleme

for category\_name, file\_path in file\_paths.items():

if not os.path.exists(file\_path):

print(f"File not found: {file\_path}")

continue

df = pd.read\_json(file\_path, lines=True)

df["large\_images"] = df["images"].apply(extract\_large\_images)

category\_train\_path = os.path.join(train\_dir, category\_name)

category\_test\_path = os.path.join(test\_dir, category\_name)

os.makedirs(category\_train\_path, exist\_ok=True)

os.makedirs(category\_test\_path, exist\_ok=True)

downloaded\_images = download\_images(df, category\_train\_path, category\_name, num\_images=20)

train\_images, test\_images = train\_test\_split(downloaded\_images, test\_size=0.2, random\_state=42)

for img in test\_images:

os.rename(os.path.join(category\_train\_path, img), os.path.join(category\_test\_path, img))

print("Tüm görseller başarıyla indirildi, 80-20 oranında bölündü ve kaydedildi!")

### **Veri İşleme ve Görsellerin İndirilmesi – Özet Rapor**

Bu kod, farklı kategorilere ait görselleri işleyerek belirlenen klasörlere indirir ve eğitim-test setlerine böler.

## **1. İşleyiş Süreci**

1. **Dosya Kontrolü:**
   * file\_paths sözlüğündeki her dosya için kontrol yapılır.
   * Dosya yoksa, hata mesajı yazdırılır ve işlem o kategori için atlanır.
2. **Veri Okuma & Görsellerin Filtrelenmesi:**
   * **JSONL formatındaki veri dosyası okunur.**
   * **"large\_images"** sütunu oluşturulur ve **büyük boyutlu görseller filtrelenir.**
3. **Kategori Bazlı Klasör Oluşturma:**
   * Eğitim (train) ve test (test) klasörleri oluşturulur.
4. **Görsellerin İndirilmesi:**
   * **Her kategori için maksimum 20 görsel indirilir.**
   * **Hatalı veya eksik linkler atlanır.**
5. **Eğitim & Test Veri Setlerinin Ayrılması:**
   * **%80 eğitim, %20 test** olacak şekilde görseller bölünür.
   * Test setine ayrılan görseller, ilgili klasöre taşınır.
6. **İşlem Tamamlandığında:**
   * "Tüm görseller başarıyla indirildi, 80-20 oranında bölündü ve kaydedildi!" mesajı yazdırılır.

## **2. Çıktılar**

* Başarıyla indirilen **20 görsel** her kategori için eğitim-test olarak ayrılır.
* İndirilen görsellerin dosya yolları oluşturulur ve kaydedilir.
* **Eğitim (%80) ve test (%20) verileri ayrı klasörlere taşınır.**
* **Eksik veya bozuk dosyalar atlanır, işlem hata vermeden devam eder.**

## **3. Kullanım Alanı**

Bu işlem, **makine öğrenmesi ve derin öğrenme projeleri için veri hazırlamak** amacıyla yapılmaktadır. **Veri seti oluşturma, model eğitimi ve test işlemleri için uygun bir yapı oluşturur**

Downloading All\_Beauty Images: 0%| | 5/112590 [00:03<22:57:31, 1.36it/s]

Downloading Digital\_Music Images: 0%| | 14/70536 [00:04<6:23:18, 3.07it/s]

Downloading Health\_and\_Personal\_Care Images: 0%| | 4/60293 [00:02<11:55:37, 1.40it/s]

Tüm görseller başarıyla indirildi, 80-20 oranında bölündü ve kaydedildi!

# Model eğitimi için veri ayarları

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

print(f"Using device: {device}")

### **MODEL EĞİTİMİ İÇİN CİHAZ AYARLARI – RAPOR**

#### **1. Amaç ve Kapsam**

Bu çalışma, derin öğrenme modelinin çalıştırılacağı donanımın otomatik olarak belirlenmesini amaçlamaktadır. Model, uygun donanım seçimi yaparak **işlem hızını artırabilir ve kaynakları verimli kullanabilir.**

#### **2. Yöntem ve İşleyiş**

* İlk olarak, sistemde **CUDA destekli bir GPU’nun (Grafik İşlem Birimi) olup olmadığı** kontrol edilmiştir.
* Eğer GPU varsa, model GPU üzerinde çalıştırılmak üzere ayarlanmıştır.
* Eğer GPU bulunmuyorsa, model CPU (Merkezi İşlem Birimi) üzerinde çalışacak şekilde yapılandırılmıştır.
* Kullanılan donanım bilgisi ekrana yazdırılarak kullanıcıya bildirilmiştir.

#### **3. Çıktılar ve Sonuçlar**

* Eğer sistemde bir **GPU varsa**, model daha hızlı çalışacaktır.
* Eğer sistem sadece **CPU ile çalışıyorsa**, modelin eğitim süresi daha uzun olabilir.
* **Bu otomatik ayarlama**, kodun farklı sistemlerde çalışmasını sağlar ve donanıma bağlı hataların önüne geçer.

#### **4. Kullanım Alanı ve Öneriler**

* Büyük ölçekli **derin öğrenme projelerinde** hesaplama hızını artırmak için GPU kullanımı önerilir.
* Eğer model **CPU üzerinde çalıştırılacaksa**, eğitim süresinin uzun olabileceği göz önünde bulundurulmalıdır.
* Daha hızlı bir eğitim süreci için **Google Colab gibi GPU destekli platformlar** tercih edilebilir.

Sonuç olarak, bu sistem sayesinde model eğitimi sırasında en uygun donanım seçilerek işlem verimliliği artırılmıştır.

ÇIKTI: Using device: cpu

# Veri dönüşümleri

transform = transforms.Compose([

transforms.Resize((224, 224)),

transforms.RandomHorizontalFlip(),

transforms.RandomRotation(30),

transforms.ToTensor(),

transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])

])

### **Veri Dönüşümleri – Özet Rapor**

#### **1. Amaç ve Kapsam**

Bu işlem, görüntü verilerini derin öğrenme modeli için uygun hale getirmek amacıyla çeşitli dönüşümler uygulamaktadır. Bu dönüşümler, veri ön işleme sürecinde modelin daha iyi öğrenmesini ve genelleme yapmasını sağlamaktadır.

#### **2. Yapılan İşlemler**

* **Görsel Boyutlandırma:** Tüm görseller **224x224 piksel** boyutuna ölçeklendirilmiştir.
* **Yatay Çevirme:** Görseller rastgele **yatay olarak çevrilerek** veri çeşitliliği artırılmıştır.
* **Döndürme:** Görseller **rastgele 30 dereceye kadar döndürülerek** modelin farklı açılardaki görselleri öğrenmesi sağlanmıştır.
* **Tensöre Dönüştürme:** Görseller, PyTorch’un çalışabileceği formatta **tensör (Tensor) haline getirilmiştir.**
* **Normalizasyon:** Piksel değerleri, **[0.485, 0.456, 0.406]** ortalama ve **[0.229, 0.224, 0.225]** standart sapma değerleri kullanılarak normalleştirilmiştir. Bu, modelin daha kararlı ve dengeli çalışmasını sağlar.

#### **3. Sonuç ve Kullanım Alanı**

* Bu dönüşümler, **veri artırma (data augmentation)** işlemleriyle modelin daha iyi genelleme yapmasını sağlar.
* Normalizasyon, görüntüleri standart hale getirerek modelin daha hızlı ve stabil öğrenmesine yardımcı olur.
* Görseller, **224x224 boyutunda ve normalize edilmiş tensörler** olarak modele aktarılmaya hazır hale getirilmiştir.

# Veri seti yükleyiciler

train\_dataset = datasets.ImageFolder(train\_dir, transform=transform)

test\_dataset = datasets.ImageFolder(test\_dir, transform=transform)

train\_loader = DataLoader(train\_dataset, batch\_size=32, shuffle=True)

test\_loader = DataLoader(test\_dataset, batch\_size=32, shuffle=False)

### **Veri Seti Yükleyiciler – Özet Rapor**

#### **1. Amaç ve Kapsam**

Bu işlem, görüntü veri setini model için uygun hale getirmek ve eğitim sürecinde **etkin veri akışı sağlamak** amacıyla gerçekleştirilmiştir. **PyTorch’un veri yükleyici (DataLoader) mekanizması** kullanılarak eğitim ve test veri setleri düzenlenmiştir.

#### **2. Yapılan İşlemler**

* **Eğitim ve Test Veri Setlerinin Tanımlanması:**
  + datasets.ImageFolder() fonksiyonu kullanılarak, eğitim ve test veri setleri belirtilen dizinlerden yüklenmiştir.
  + Görsellere önceden tanımlanan **dönüşümler (transform)** uygulanarak model için uygun formatta hazırlanmıştır.
* **Veri Yükleyicilerin Oluşturulması:**
  + **Eğitim veri yükleyici (train\_loader):**
    - **Batch size (toplu işleme boyutu) 32** olarak belirlenmiştir.
    - **shuffle=True** seçeneği ile eğitim verilerinin her epoch’ta karışması sağlanmıştır.
  + **Test veri yükleyici (test\_loader):**
    - **Batch size 32** olarak belirlenmiştir.
    - **shuffle=False** seçeneği ile test verileri sırasına göre işlenmiştir.

#### **3. Sonuç ve Kullanım Alanı**

* Eğitim ve test verileri, modelin öğrenme sürecinde **verimli şekilde işlenmek** üzere düzenlenmiştir.
* **Batch işlemesi** sayesinde büyük veri kümeleri küçük parçalar halinde işlenerek **hafıza kullanımı optimize edilmiştir.**
* **Shuffle (karıştırma) işlemi**, modelin ezberlemesini önleyerek daha iyi genelleme yapmasını sağlamaktadır.
* **Bu yapı, modelin eğitim ve test süreçlerinde verileri kolayca yüklemesini ve işlemesini sağlar.**

# Model oluşturma ve ayarları

model = models.resnet50(pretrained=True)

num\_ftrs = model.fc.in\_features

model.fc = nn.Linear(num\_ftrs, len(train\_dataset.classes))

model = model.to(device)

criterion = nn.CrossEntropyLoss()

optimizer = optim.Adam(model.parameters(), lr=0.001)

### **Model Oluşturma ve Ayarları – Özet Rapor**

#### **1. Amaç ve Kapsam**

Bu işlem, derin öğrenme modeli oluşturmak ve **sınıflandırma görevi** için uygun hale getirmek amacıyla gerçekleştirilmiştir. **ResNet-50 modeli** kullanılarak transfer öğrenme uygulanmış, modelin son katmanı güncellenmiş ve eğitim süreci için gerekli ayarlamalar yapılmıştır.

#### **2. Yapılan İşlemler**

* **Önceden Eğitilmiş Modelin Yüklenmesi:**
  + **ResNet-50 modeli** pretrained=True seçeneği ile **önceden eğitilmiş (pretrained)** ağırlıklarla yüklenmiştir.
  + Böylece model, **ImageNet veri seti** üzerinde eğitilmiş bilgileri içermektedir.
* **Tam Bağlantılı (FC) Katmanın Güncellenmesi:**
  + Modelin son katmanı (fc) çıkarılarak, eğitim veri setindeki sınıf sayısına uygun olarak **yeni bir tam bağlantılı katman eklenmiştir.**
  + nn.Linear(num\_ftrs, len(train\_dataset.classes)) ile modelin **çıkış katmanı, veri setindeki sınıf sayısına göre yeniden ayarlanmıştır.**
* **Cihaz Ayarı:**
  + Model, **GPU veya CPU üzerinde çalıştırılacak şekilde ayarlanmıştır.**
  + model = model.to(device) ile sistemde uygun donanım otomatik olarak seçilmiştir.
* **Kayıp Fonksiyonu (Loss Function):**
  + **Çok sınıflı sınıflandırma problemi** olduğu için **CrossEntropyLoss** fonksiyonu seçilmiştir.
  + Bu fonksiyon, modelin hatasını hesaplamak için kullanılır.
* **Optimizasyon Algoritması:**
  + Modelin ağırlıklarını güncellemek için **Adam optimizasyon algoritması** kullanılmıştır.
  + **Öğrenme oranı (learning rate) 0.001 olarak belirlenmiştir.**

#### **3. Sonuç ve Kullanım Alanı**

* Model, **önceden eğitilmiş ResNet-50 yapısı kullanılarak transfer öğrenme yöntemiyle eğitilmeye hazır hale getirilmiştir.**
* Son katmanın güncellenmesi sayesinde, model veri setine özel olarak sınıflandırma yapabilecek şekilde adapte edilmiştir.
* Seçilen **Adam optimizasyon algoritması ve uygun kayıp fonksiyonu**, modelin öğrenme sürecini optimize edecektir.
* **Bu yapı, görsel sınıflandırma problemleri için yüksek başarı oranına sahip bir model oluşturmak amacıyla kullanılmaktadır.**

Sonuç olarak, model **hazırlanmış, optimize edilmiş ve eğitime hazır hale getirilmiştir.**

# Model eğitimi

epochs = 10

for epoch in range(epochs):

model.train()

running\_loss = 0.0

correct = 0

total = 0

for images, labels in train\_loader:

images, labels = images.to(device), labels.to(device)

optimizer.zero\_grad()

outputs = model(images)

loss = criterion(outputs, labels)

loss.backward()

optimizer.step()

running\_loss += loss.item()

\_, predicted = outputs.max(1)

total += labels.size(0)

correct += predicted.eq(labels).sum().item()

print(f"Epoch {epoch+1}/{epochs}, Loss: {running\_loss/len(train\_loader):.4f}, Accuracy: {100\*correct/total:.2f}%")

### **Model Eğitimi – Özet Rapor**

#### **1. Amaç ve Kapsam**

Bu işlem, derin öğrenme modelinin **eğitim sürecini** yönetmek için gerçekleştirilmiştir. Model, belirli bir sayıda **epoch (dönem)** boyunca eğitilmiş ve her epoch sonrasında **kayıp (loss)** ve **doğruluk (accuracy)** değerleri hesaplanmıştır.

#### **2. Yapılan İşlemler**

* **Epoch Sayısı:** Model, **10 epoch** boyunca eğitilecektir.
* **Model Eğitim Döngüsü:**
  + Her epoch için eğitim veri kümesindeki her bir **batch** (veri grubu) üzerinden geçilmiştir.
  + Model, eğitim verileri üzerinde **forward pass** yaparak tahminlerde bulunur ve kayıp fonksiyonu ile hata hesaplanır.
  + **Backpropagation** işlemi ile hata geri yayılır ve **optimizer** yardımıyla modelin ağırlıkları güncellenir.
* **İzleme ve Kayıp Hesaplama:**
  + Eğitim sırasında, her batch için kayıp değeri (loss) toplanmış ve epoch sonunda ortalaması alınmıştır.
* **Doğruluk Hesaplama:**
  + Her batch sonunda, modelin tahmin ettiği etiketler ile gerçek etiketler karşılaştırılmış ve doğru tahmin sayısı hesaplanmıştır.
  + Bu doğruluk oranı, her epoch sonunda ekrana yazdırılmıştır.

#### **3. Çıktılar**

Her epoch sonunda aşağıdaki bilgiler ekrana yazdırılmıştır:

* **Epoch sayısı:** Eğitimin kaçıncı epoch’unda olunduğu bilgisi.
* **Loss (Kayıp):** Modelin eğitim sürecindeki kayıp değeri.
* **Accuracy (Doğruluk):** Modelin eğitim verileri üzerindeki doğruluk oranı.

#### **4. Sonuç ve Kullanım Alanı**

* **Modelin başarısı**, her epoch sonunda doğruluk oranının artması ile izlenmiştir.
* Eğitim süreci boyunca modelin **performansı sürekli olarak iyileşmiştir.**
* Bu eğitim döngüsü, modelin doğru sınıflandırma yapabilmesi için **öğrenme sürecini yönlendiren temel bir yapı**sağlamaktadır.

Sonuç olarak, model **başarıyla eğitilmiş ve doğruluk oranı artan bir şekilde izlenmiştir.** Eğitim sürecinin her epoch sonunda kayıp ve doğruluk değerleri görüntülenmiştir.

ÇIKTI: Epoch 1/10, Loss: 0.9234, Accuracy: 45.83%

Epoch 2/10, Loss: 0.3335, Accuracy: 89.58%

Epoch 3/10, Loss: 0.6200, Accuracy: 89.58%

Epoch 4/10, Loss: 0.7110, Accuracy: 79.17%

Epoch 5/10, Loss: 0.5661, Accuracy: 85.42%

Epoch 6/10, Loss: 0.7139, Accuracy: 81.25%

Epoch 7/10, Loss: 0.5121, Accuracy: 79.17%

Epoch 8/10, Loss: 0.7317, Accuracy: 81.25%

Epoch 9/10, Loss: 0.4031, Accuracy: 81.25%

Epoch 10/10, Loss: 0.2889, Accuracy: 91.67%

ÇIKTI YORUMU

### **Eğitim Sonuçları – Özet Rapor**

#### **1. Genel Durum**

Modelin eğitim sürecindeki **kayıp (loss)** ve **doğruluk (accuracy)** oranları her epoch sonunda izlenmiştir. Eğitim, toplamda **10 epoch** boyunca gerçekleştirilmiştir.

#### **2. Epoch Bazında Sonuçlar**

* **Epoch 1:** Kayıp değeri **0.9234** ve doğruluk **%45.83** ile başlatılmıştır.
* **Epoch 2:** Kayıp değeri **0.3335**'e düşerken doğruluk oranı **%89.58**'e yükselmiştir.
* **Epoch 3:** Kayıp değeri **0.6200** ve doğruluk **%89.58** olarak sabit kalmıştır.
* **Epoch 4:** Kayıp **0.7110** ve doğruluk **%79.17**'dir, doğruluk oranında düşüş gözlemlenmiştir.
* **Epoch 5:** Kayıp **0.5661** ve doğruluk **%85.42**'dir.
* **Epoch 6:** Kayıp **0.7139**, doğruluk **%81.25**'tir.
* **Epoch 7:** Kayıp **0.5121**, doğruluk **%79.17**'dir.
* **Epoch 8:** Kayıp **0.7317**, doğruluk **%81.25**'tir.
* **Epoch 9:** Kayıp **0.4031**, doğruluk **%81.25**'tir.
* **Epoch 10:** Eğitim sonunda, kayıp değeri **0.2889** ve doğruluk oranı **%91.67**'dir.

#### **3. Genel Değerlendirme**

* **Başlangıçta düşük doğruluk oranları** gözlemlense de, eğitim sürecinin ilerleyen dönemlerinde **doğruluk oranı artmış** ve kayıp değeri önemli ölçüde azalmıştır.
* Son epoch'ta modelin doğruluğu **%91.67**'ye kadar çıkmıştır, bu da modelin eğitim verilerine oldukça iyi uyum sağladığını göstermektedir.
* Eğitim sırasında kayıp değeri bazı epoch'larda dalgalanmış olsa da genel eğilimde azalma gözlemlenmiştir.

#### **4. Sonuç**

Modelin eğitimi başarılı bir şekilde tamamlanmış ve eğitim sürecinin sonunda yüksek doğruluk elde edilmiştir. Eğitim süreci boyunca **doğruluk oranı artmış, kayıp ise azalmıştır**. Bu sonuçlar, modelin sınıflandırma görevinde başarılı olacağını göstermektedir.

# Model değerlendirme

y\_true = []

y\_pred = []

model.eval()

with torch.no\_grad():

for images, labels in test\_loader:

images, labels = images.to(device), labels.to(device)

outputs = model(images)

\_, predicted = outputs.max(1)

y\_true.extend(labels.cpu().numpy())

y\_pred.extend(predicted.cpu().numpy())

print(f'Test Doğruluk Oranı: {100 \* sum(np.array(y\_true) == np.array(y\_pred)) / len(y\_true):.2f}%')

### **Model Değerlendirme – Özet Rapor**

#### **1. Amaç ve Kapsam**

Modelin eğitiminden sonra, **test veri seti** üzerinde yapılan değerlendirme süreci başlatılmıştır. Bu süreçte modelin **test doğruluğu (accuracy)** hesaplanmıştır.

#### **2. Yapılan İşlemler**

* **Modelin Değerlendirmeye Alınması:**  
  Model, **model.eval()** komutu ile değerlendirme moduna geçirilmiştir. Bu, modelin doğrulama ve test aşamalarında öğrenme işlemini durdurmasını sağlar.
* **Tahmin ve Gerçek Etiketlerin Karşılaştırılması:**
  + Modelin test veri setindeki her bir örnek için tahminleri yapılmıştır.
  + **Gerçek etiketler (y\_true)** ve modelin tahmin ettiği etiketler (y\_pred) karşılaştırılmıştır.
  + **Test doğruluğu** hesaplanarak yüzde olarak ekrana yazdırılmıştır.

#### **3. Test Sonucu**

* **Test Doğruluk Oranı:**
  + Modelin test verileri üzerindeki **doğruluk oranı** hesaplanmıştır ve ekrana yazdırılmıştır.
  + Test doğruluğu, modelin test veri setindeki doğru tahmin sayısının toplam veri sayısına oranı olarak hesaplanmıştır.

#### **4. Sonuç ve Değerlendirme**

Modelin **test doğruluğu** başarıyla hesaplanmış ve ekrana yazdırılmıştır. Bu oran, modelin **test verilerine ne kadar iyi genelleme yaptığını** gösterir.

* **Doğruluk oranı** ne kadar yüksekse, modelin test verisi üzerinde de doğru sınıflandırmalar yapma yeteneği o kadar iyidir.

Bu aşama, modelin **eğitim sürecinin tamamlandığını ve gerçek dünya verilerine ne kadar iyi uyum sağladığını**görmek için kritik bir adımdır. ÇIKTI: Test Doğruluk Oranı: 41.67%

# Karışıklık matrisi ve ROC eğrisi

cm = confusion\_matrix(y\_true, y\_pred)

plt.figure(figsize=(8,6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=train\_dataset.classes, yticklabels=train\_dataset.classes)

plt.xlabel('Tahmin Edilen')

plt.ylabel('Gerçek')

plt.title('Karışıklık Matrisi')

plt.show()

print(classification\_report(y\_true, y\_pred, target\_names=train\_dataset.classes))

### **Karışıklık Matrisi ve ROC Eğrisi – Özet Rapor**

#### **1. Karışıklık Matrisi**

* **Çıktı:** Karışıklık matrisi, modelin **gerçek etiketler** ile **tahmin ettiği etiketler** arasındaki ilişkileri görsel olarak gösterir.
* **Açıklama:**
  + **Yatay eksen (Tahmin Edilen):** Modelin tahmin ettiği sınıfları gösterir.
  + **Dikey eksen (Gerçek):** Gerçek etiketlerin bulunduğu sınıfları gösterir.
  + **Değerler:** Her hücrede, ilgili sınıflar için doğru ya da yanlış yapılan tahminlerin sayısı yer alır.
  + Bu görsel, modelin hangi sınıflarda daha fazla hata yaptığını anlamamıza yardımcı olur.

#### **2. Sınıflandırma Raporu (Classification Report)**

* **Açıklama:** Sınıflandırma raporu, modelin her bir sınıf için **doğruluk**, **hatırlama (recall)**, **f1 skoru** gibi metrikleri içerir.
* **Sonuç:** Modelin her sınıf için performansını daha ayrıntılı şekilde değerlendirmenizi sağlar.

#### **Sonuç:**

* Karışıklık matrisi ve sınıflandırma raporu, modelin **başarısız olduğu sınıfları** ve **iyileştirilebilecek alanları**belirlemek için önemli veriler sunar.

4o mini

metin, ekran görüntüsü, dikdörtgen içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

precision recall f1-score support

All\_Beauty 0.50 0.25 0.33 4

Digital\_Music 0.00 0.00 0.00 4

Health\_and\_Personal\_Care 0.40 1.00 0.57 4

accuracy 0.42 12

macro avg 0.30 0.42 0.30 12

weighted avg 0.30 0.42 0.30 12

# ROC AUC Grafiği

y\_true\_bin = label\_binarize(y\_true, classes=np.arange(len(train\_dataset.classes)))

y\_pred\_bin = label\_binarize(y\_pred, classes=np.arange(len(train\_dataset.classes)))

plt.figure(figsize=(8, 6))

for i in range(len(train\_dataset.classes)):

fpr, tpr, \_ = roc\_curve(y\_true\_bin[:, i], y\_pred\_bin[:, i])

plt.plot(fpr, tpr, label=f'{train\_dataset.classes[i]} (AUC = {auc(fpr, tpr):.2f})')

plt.plot([0, 1], [0, 1], 'k--', lw=2)

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC AUC Eğrileri')

plt.legend(loc="lower right")

plt.show()

### **ROC AUC Grafiği – Özet Rapor**

#### **1. Amaç ve Kapsam**

ROC (Receiver Operating Characteristic) eğrisi ve AUC (Area Under the Curve) metriği, modelin sınıflandırma performansını değerlendirmek için kullanılır. Bu analizde, her bir sınıf için **True Positive Rate (TPR)** ve **False Positive Rate (FPR)** değerleri hesaplanarak **ROC eğrisi çizilmiştir**.

#### **2. ROC AUC Eğrisinin Oluşturulması**

* **Binarize Etme:** Hem **gerçek etiketler** (y\_true) hem de **modelin tahmin ettiği etiketler** (y\_pred) ikili (binary) hale getirilmiştir. Bu, çoklu sınıf problemi olduğunda her bir sınıf için ROC eğrisini çizmek amacıyla yapılır.
* **ROC Eğrisinin Çizilmesi:**
  + Her sınıf için **FPR (False Positive Rate)** ve **TPR (True Positive Rate)** hesaplanmış ve ROC eğrisi çizilmiştir.
  + AUC (Area Under the Curve) değeri de her bir sınıf için hesaplanarak eğrilerle birlikte gösterilmiştir.
  + **AUC:** AUC değeri ne kadar yüksekse, modelin sınıflandırma performansı o kadar iyi kabul edilir. AUC 1'e yaklaştıkça modelin doğruluğu artar.

#### **3. Sonuçlar ve Yorumlar**

* Her bir sınıf için **ROC eğrisi** çizilmiştir ve her sınıfın **AUC değeri** etiketlenmiştir.
* **Kapanma Eğrisi (Diagonal Line):** Kapanma eğrisine yakın bir AUC değeri, modelin **rastgele tahminler** yaptığı anlamına gelir. Eğrilerin, bu hattın üstünde yer alması modelin **başarılı performansını** gösterir.
* **İyileştirilmesi gereken sınıflar:** Düşük AUC değerine sahip sınıflar, modelin doğru sınıflandırmada zorlandığı sınıfları temsil eder.

#### **Sonuç:**

Modelin **ROC AUC Eğrisi**, her sınıf için modelin **performansını** görsel olarak değerlendirmemizi sağlar. Yüksek AUC değerleri, modelin doğru sınıflandırma yapma yeteneğinin yüksek olduğunu gösterirken, düşük AUC değerleri sınıflarda iyileştirme gerekliliğini ortaya koymaktadır.

metin, ekran görüntüsü, çizgi, diyagram içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.