## DATA_İNGESTİON

Bu kod, bir makine öğrenimi projesi için gereken veri indirme ve işleme adımlarını otomatikleştirir. İlk olarak, yapılandırma dosyalarını okuyarak proje ayarlarını yükler ve projede gerekli klasörleri oluşturur. Ardından, belirtilen bir URL'den veri indirir ve bu veriyi hedef klasöre kaydeder. İndirilen dosya bir zip dosyasıysa, içeriğini açarak istenen klasöre çıkarır. Kodun sonunda, bu adımlar toplu halde çalıştırılır ve hata yönetimi yapılır. Bu sayede, veri hazırlığı süreci hızlı ve düzenli bir şekilde yürütülmüş olur.

## PREPARE_BASE_MODEL

Bu kod, derin öğrenme modelini projeye özgü hale getirmek için çeşitli adımları otomatikleştirir. İlk olarak, kod yapılandırma ve parametre dosyalarını okuyarak proje için gerekli ayarları yükler ve çalıştırılacak dizinleri oluşturur. Ardından, PrepareBaseModel sınıfı VGG16 modelini önceden eğitilmiş ağırlıklarla indirir, modelin katmanlarını belirli parametrelere göre yapılandırır ve ilk model dosyasını kaydeder. Sonrasında, model güncellenir ve projeye özel hale getirilir: Bu işlem sırasında VGG16 modelinin tüm katmanları dondurulur, ardından projeye özgü sınıflar için yeni bir Dense katman eklenir. Model, belirlenen öğrenme hızı ve kayıp fonksiyonu ile derlenir ve özet çıktısı ile birlikte güncellenmiş model olarak tekrar kaydedilir. Kodun sonunda, bu işlemler bir hata yakalama bloğu içinde çalıştırılır, bu da herhangi bir sorun oluştuğunda kullanıcıyı bilgilendirir. Bu yapı, projede veri hazırlığından model yapılandırmasına kadar birçok işlemi düzenli ve otomatik bir akış içinde yürütmek için kullanılır.

## PREPARE_CALLBACK

Bu kod, TensorFlow model eğitim sürecinde kullanılacak **TensorBoard** ve model kontrol noktası (checkpoint) geri çağrıları (callbacks) için gerekli ayarları otomatik olarak hazırlar. İlk olarak, **ConfigurationManager** sınıfı yapılandırma dosyalarını okur, gerekli klasörleri (**artifacts**, **checkpoint** ve **tensorboard** gibi) oluşturur ve ayarları **PrepareCallbacksConfig** sınıfına iletir. Ardından, **PrepareCallback** sınıfı **TensorBoard** geri çağrısı için belirli bir zaman damgasıyla bir kayıt dizini oluşturur ve model eğitimi sırasında gerçek zamanlı ölçümleri gözlemlemek üzere yapılandırır. Aynı sınıf, modelin sadece en iyi performans gösteren hali kaydedilmek üzere bir checkpoint geri çağrısı (callback) de oluşturur. Son olarak, bu geri çağrılar bir liste halinde döndürülür ve böylece model eğitimi sırasında hem **TensorBoard** takibi yapılabilir hem de en iyi model durumu otomatik olarak kaydedilir.

## Training

Bu kod, tavuk hastalığı sınıflandırma modeli için bir eğitim sürecini başlatmak amacıyla geri çağrı ve eğitim yapılandırmalarını oluşturan bir yapı geliştirmektedir. İlk olarak **ConfigurationManager** sınıfı, **config.yaml** ve **params.yaml** dosyalarını okuyarak **artifacts** ve **training** gibi gerekli dizinleri oluşturur ve bu dosyalardaki ayarlara göre eğitim sürecini yapılandırır.

Geri çağrı yapılandırmaları için **PrepareCallbacksConfig** sınıfını kullanan **PrepareCallback**, model eğitim süresince **TensorBoard** ve en iyi modeli kaydedecek bir kontrol noktası (checkpoint) geri çağrıları oluşturur. **TensorBoard**, her bir eğitim sürecine bir zaman damgası ekleyerek eğitim ölçümlerinin görsel analizini sağlarken, checkpoint geri çağrısı ise modelin sadece en iyi durumunu kaydeder.

Modelin eğitim süreci **Training** sınıfı içinde gerçekleşir. **train_valid_generator** yöntemi, eğitim ve doğrulama veri kümelerini yüklemek için **ImageDataGenerator** sınıfını kullanır ve veri artırma işlemi (augmentation) uygulanıp uygulanmayacağını kontrol eder. Bu sayede model, eğitim sırasında farklı varyasyonlarla daha genellenebilir bir hale gelir. Eğitim süreci, belirlenen epoch sayısınca devam eder ve her epoch boyunca eğitim ve doğrulama verileri modelin performansını artırmak amacıyla yeniden işlenir. Eğitim tamamlandıktan sonra model, belirlenen yolda kaydedilir.

## Model_Evaluation

Bu kod, tavuk hastalığı sınıflandırma modelinin doğrulama verileri üzerinde performansını değerlendirmek için geliştirilmiştir. İlk olarak, ConfigurationManager sınıfı yapılandırma dosyalarını (config.yaml ve params.yaml) okuyarak artifacts gibi gerekli dizinleri oluşturur ve doğrulama sürecinde kullanılacak ayarları belirler. Bu ayarları, modelin yolunu, eğitim verilerini ve diğer parametreleri içeren EvaluationConfig sınıfına aktarır.

Evaluation sınıfı doğrulama işlemini yürütür. _valid_generator yöntemi, eğitim verilerini içeren dizinden doğrulama veri kümesini oluşturur. Bu veri kümesi %30 oranında ayrılır ve model, bu verilere dayalı olarak doğrulama işlemi gerçekleştirecek şekilde yapılandırılır. evaluation yöntemi, önceden eğitilmiş modeli model.h5 dosyasından yükler ve doğrulama verileri üzerinde modeli değerlendirir, doğrulama süreci sonunda modelin doğruluk (accuracy) ve kayıp (loss) skorlarını döndürür.

Son olarak, save_score yöntemi ile bu değerlendirme sonuçları JSON formatında bir scores.json dosyasına kaydedilir ve modelin doğrulama verileri üzerindeki performansı kalıcı hale getirilir.

## AŞAMA 1: config.yaml dosyasının güncellenmesi

Bu YAML dosyası, tavuk hastalığı sınıflandırma projesindeki dosya yapısını ve veri yollarını tanımlar. data_ingestion bölümü verinin indirileceği ve açılacağı dizini belirlerken, prepare_base_model temel modelin dosyalarını saklamak için yapılandırma sağlar. prepare_callbacks ise TensorBoard ve model kontrol noktası gibi geri çağırmaları ayarlamak için gerekli dizinleri tanımlar.

## AŞAMA 2: params.yaml dosyanının güncellenmesi

Bu yapılandırma, model eğitimi için kullanılan hiperparametreleri içerir; burada görüntü boyutu VGG 16 modeline uygun olarak 224x224 piksel olarak ayarlanmış ve veri artırma (augmentation) etkinleştirilmiştir. Ayrıca, modelin katmanları arasında en üst katman dahil edilmemiş, 1 epoch için 16'lık bir batch boyutu belirlenmiş ve öğrenme hızı 0.01 olarak ayarlanmıştır.

## AŞAMA 3: entity dosyasını güncellemesi

Çeşitli yapılandırma ayarlarını tanımlamak için veri sınıfları kullanır ve her bir sınıf, projenin farklı aşamalarında gerekli olan parametreleri saklar. Örneğin, DataIngestionConfig, veri kaynakları ve yerel dosya yollarını tanımlarken, TrainingConfig, modelin eğitim süreci için gerekli olan parametreleri (epoch sayısı, batch boyutu, görüntü boyutu gibi) belirler. Bu yapı, projenin yapılandırmalarını merkezi bir yerde tutarak kodun daha düzenli ve okunabilir olmasını sağlar.

## AŞAMA 4:config dosyanın güncellenmesi

Bu kod, bir ConfigurationManager sınıfı aracılığıyla proje yapılandırmalarını yönetir, böylece veri alma, model hazırlama, geri çağırma ayarları, eğitim ve değerlendirme için gerekli parametreleri sağlar. Her bir yöntem, yapılandırma dosyalarından bilgileri okuyarak ilgili veri sınıflarını oluşturur ve gerekli dizinleri oluşturarak projenin düzenli bir şekilde çalışmasını sağlar. Bu yapı, projenin esnekliğini artırırken, yapılandırmaların merkezi bir şekilde yönetilmesini mümkün kılar.

## AŞAMA 5: components klasörünün güncellenmesi

Bu dosyalar, bir makine öğrenimi projesinin temel bileşenlerini oluşturur ve her biri belirli bir işlevsellik sunar. data_ingestion.py veri setini indirirken, prepare_base_model.py modelin mimarisini ayarlar, prepare_callbacks.py eğitim sırasında geri çağırma fonksiyonlarını yönetir, training.py modelin eğitilmesini sağlarken, evaluation.py modelin performansını değerlendirir. Bu yapı, kodun okunabilirliğini ve bakımını kolaylaştırarak projenin modülerliğini artırır.

## AŞAMA 6: pipeline klasörünün güncellenmesi

bu dosyalar, makine öğrenimi projesinin akışını düzenleyen bir pipeline'ın parçalarını temsil eder ve her biri belirli bir aşamanın sorumluluğunu taşır. stage_01_data_ingestion.py veri setini hazırlar, stage_02_prepare_base_model.py modeli yapılandırır, stage_03_training.py modeli eğitir ve stage_04_evaluation.py eğitim sonrasında modelin performansını değerlendirir. Pipeline yapısı, süreçleri sistematik bir şekilde yöneterek sürecin izlenebilirliğini ve tekrar kullanılabilirliğini artırır.

## AŞAMA 7: Main.py dosyasının güncellenmesi

makine öğrenimi projesinin farklı aşamalarını yönetmek için tasarlanmış bir kontrol akışını temsil eder ve her aşama için hata yönetimi ve günlükleme (logging) mekanizmaları içerir. Data Ingestion, Prepare Base Model, Training ve Evaluation aşamaları, sırasıyla veri yükleme, model hazırlama, model eğitme ve model değerlendirme işlevlerini yerine getirir; bu yapı, sürecin sistematik bir şekilde yürütülmesini ve her aşamanın başarılı bir şekilde tamamlanıp tamamlanmadığını izlemeyi sağlar. Ayrıca, hata durumlarında detaylı bilgi sunarak kullanıcıların sorunları daha hızlı tespit etmelerine ve çözmelerine yardımcı olur.

## AŞAMA 8: dvc.yaml dosyasının güncellenmesi

Bu yapılandırma dosyası, makine öğrenimi projesinin çeşitli aşamalarını yönetmek için bir görev akışı tanımlar ve her aşamanın bağımlılıklarını, çıktıları ve parametrelerini belirler. Data ingestion, model hazırlama, eğitim ve değerlendirme aşamaları için komutlar ve bağımlılıkları tanımlanarak, sürecin sistematik bir şekilde yürütülmesi sağlanır. Bu yapı, projenin modülerliğini artırır, aşamalar arası veri akışını düzenler ve her adımın düzgün çalışmasını garanti eder.