

TEST KESİM UYGULAMASI

Uygulama hakkında detaylara geçmeden önce çalışma sürecini özetlemek istiyorum.

Öncelikle Google Drive'da yer alan **Ortak Drive**'da yer almamız gerekmektedir. Bu kısım kitap kaynaklarını ve ileride eğiteceğimiz model dosyalarını tutmak için kullanacağımız depolama birimidir.

Burada yer alan **Hata durumları** dosyasından (17.05.2023 itibariyle) hatalı olarak tespit edilen ve hata nedenlerinin yer aldığı bir Google Docs dosyasıdır. Hatalar son kullanıcıdan gelip bu kısma eklenderek geliştiricilere sunulmaktadır.

Buradaki hatalara ve açıklamalarına bakarak hataların çözüm aşamasına başlamaktayız.

Hatalar yazılımsal ve eğitimsel hata olarak 2 aşamaya ayrılmaktadır. Yazılımcı bu ayırmaları dokümantasyonun devamında daha iyi tespit edebilir haline gelecektir.

Erişmek İçin =>[Google Docs](#)

Sonrasında hataların ayrıntısını görmek için kod ortamında test edilir.

Yazılımsal bir hata var ise bu hata düzeltilerek yeni bir .exe halinde proje teslim edilir.

Eğer Model bazlı bir hata mevcut ise Roboflow platformuna bu resim yüklenerek etiketlenmesi sağlanır. Ve etiketlenen bu hatalı veriler, veri kümesine eklenir. Sonrasında Google Colab tarafından verilerin tekrardan eğitimi sağlanarak en uygun modelin seçilmesi sağlanır. Ve yeni model uygun isim standartlarında kullanıcıya güncelleme olarak çıkarılır.

1.1 Uygulamanın Amacı

Uygulamanın temel amacı, 16 farklı class tipinin YoloV7 ile beraber soru kalıplarını tespit edip sonrasında içerisindeki sınıfları tespit etmeye gereken; ait olduğu soru kalıbına veri olarak eklenmesi amaçlanmıştır.

Gelen veriler manipüle edilerek .xml kısmında kurumun kullanabileceği şekilde sunulmaktadır.

Uygulamanın açıklamalarından ve kullanılan teknolojileri ele aldıktan sonra konu başlıklarımız şu şekilde olacaktır:

- Verilerin Etiketlenmesi
- Model Dosyasının Eğitilmesi
- Kaynak Dosyalarına Giriş
- Projenin Execute Edilmesi

1.2 Kullanılan Teknolojiler Ve Derleyici

Uygulama Python platformunda çalışıyor olup .execute hale getirebilir bir şekilde tasarlanmıştır.

Kaynak kodlarının yer aldığı kısımda kullandığımız önemli teknolojiler:

1. Yolo v7
2. Tesseract OCR

- 3. Easy OCR
- 4. Python (3.9.12)
- 5. Anaconda

Projenin bel kemiği olan bu kütüphaneler ve yapılar olup; aşağıdaki konularda bahsedeceğim kütüphanelerle uygulama bir bütün haline gelmektedir.

Kaynak kodlarının dışında modellerimizi eğitmek ve testlerimizi gerçekleştireceğimiz platform:

=> Google Colab

olmaktadır.

Kullanılacak IDE tercihi olarak bir standart belirlenmemiş olup, Pycharm veya Visual Studio tercih edilebilir.

Bu vermiş olduğum teknolojilerin detayları aşağıda yer almaktadır.

1.2.1 YOLO V7 Algoritması Ve Neden Uygulamada Tercih Edildi ?

Yolov7, YOLO (You Only Look Once) adı verilen bir nesne algılama yaklaşımının yedinci sürümüdür. YOLO yaklaşımı, geleneksel yöntemlerden farklı olarak nesne algılama işlemini tek bir aşamada gerçekleştirir. Bu, YOLO modellerinin diğerlerine göre daha hızlı çalışmasını sağlar.

Yolov7'nin önemli özellikleri şunlardır:

Hızlı Nesne Algılama: Yolov7, gerçek zamanlı nesne algılama için optimize edilmiştir. Diğer yöntemlere kıyasla daha hızlı çalışır ve yüksek çözünürlükli görüntülerde bile etkili sonuçlar üretебilir. Bu, gerçek zamanlı uygulamalar için idealdir, örneğin otomotiv sürücü, güvenlik sistemleri veya video analiz uygulamaları.

Yüksek Doğruluk: Yolov7, yüksek doğruluk oranlarına sahiptir. Geliştirilmiş algoritma ve mimarisi sayesinde nesneleri daha iyi tespit edebilir ve sınıflandırabilir. Özellikle çoklu sınıf problemlerde ve karmaşık sahnelerde yüksek başarı elde edebilir.

Genellemeye Yetkin: Yolov7, genellemeye yetenekleri açısından güçlidür. Model, farklı veri setleri üzerinde eğitilerek farklı ortamlarda ve nesnelerde de başarılı sonuçlar verebilir. Bu, modelin çeşitli uygulamalarda kullanılabilmesini sağlar.

Ölçeklenebilirlik: Yolov7, daha büyük ve karmaşık veri setleri üzerinde eğitilebilir ve daha fazla nesne sınıfını tanıyalabilir. Bu, çeşitli uygulamalarda kullanılan geniş veri setleriyle çalışmak isteyen araştırmacılar ve mühendisler için önemlidir.

Yolov7, bilgisayarlı görüş ve yapay zeka alanında geniş bir kullanım alanına sahiptir. Otonom araçlar, güvenlik sistemleri, nesne takibi, nesne sınıflandırma gibi birçok alanda etkili bir şekilde kullanılabilir. Yolov7'nin hızlı çalışması, gerçek zamanlı uygulamaların yanı sıra yüksek hacimli verileri işleyen sistemlerde de avantaj sağlar.

Sonuç olarak, Yolov7, hızlı, doğru ve genellemeye yetenekleri güçlü olan bir nesne algılama modelidir. Bu özellikleri sayesinde birçok uygulama alanında kullanılabilir ve bilgisayarlı görüş ve yapay zeka alanındaki ilerlemeleri destekleyebilir.

Uygulamamızda ise yüksek doğruluk oranlarına bu şekilde çıkabilmekteyiz.

1.2.2 OCR Ve Tercihlerimiz

Öncelikle OCR'in tanımından bahsedeyece olursak:

OCR (Optical Character Recognition), optik karakter tanıma anlamına gelir ve yazılı metni tarayarak veya görüntüleyerek bilgisayar tarafından okunabilir metne dönüştürme işlemidir. OCR teknolojisi, kağıt tabanlı belgeleri veya dijital görüntülerdeki metni tanıtmak ve çıktı olarak almak için kullanılır.

Uygulamımızda bir kaç kısımda bu teknolojiyi sayıların ve metin kutucuklarının yolo algoritmasının tespiti sonrasında recognize edilmesi için OCR kullanıyor olacağız.

Bu işlemleri yapmadan önce input olan gelen verileri manipüle etmemiz gerekmektedir. Bunun nedeni ham verinin okunmasındaki zorlukluktan kaynaklıdır, bu yüzden bu tarz bir işleme tabii tutarız.

Konu ile ilgili bilgileri aşağıdaki başlıklarda belirtiyor olacağım.

Aşağıda kullandığımız OCR tipleri güncel olarak projede yer alan kütüphanelere aittir.

1.2.2.1 PyTesseract OCR

PytesseractOCR, Google tarafından geliştirilen Tesseract OCR motorunu Python ile kullanmayı sağlayan bir kütüphanedir. Tesseract, güçlü bir metin tanıma motorudur ve çeşitli dillerdeki metinleri tanıyalabilir. PytesseractOCR, görüntülerini Tesseract motoruna ileterek metin çıktısı almanızı sağlar. Kullanımı kolaydır ve çeşitli metin tanıma projelerinde tercih edilir.

Uygulamamızın öncelikli OCR kütüphanesidir. Genellikle ön planda işleme almaya çalışırız, bunun nedeni doğruluk oranının daha yüksek olmasından kaynaklıdır.

Aktif bir şekilde kullanmanız için aşağıdaki bağlantıdan TesseractOCR'a ait exe'yi kurmanız gerekmektedir.

Erişmek İçin =>[TesseractOCR](#)

1.2.2.2 Easy OCR

EasyOCR, çeşitli dillerde metni tanıma yeteneğine sahip güçlü bir OCR kütüphanesidir. Tesseract'in yanı sıra diğer OCR motorlarını da kullanabilir. EasyOCR, önceden eğitilmiş dil modelleri kullanarak metin tanıma yapar ve çoklu dil desteği sağlar. Kullanımı basittir.

Tespit algoritmalarımızın son doğrulama kalesidir. Eğer PyTesseract recognize kısmında başarısız olacak olursa EasyOCR devreye girerek farklı manipülasyon işlemlerine sokulmuş bir image ile tekrardan recognize edilmeye çalışacaktır.

OCR ile ilgili detay bilgileri Kod kısmına geçtiğimde aktarıyor olacağım.

1.2.3 Google Colaboratory

- Model Eğitimi ve Dağıtım: Roboflow, model eğitimi sürecini destekler ve farklı derin öğrenme çerçeveleriyle entegrasyon sağlar. Veri setinizi eğitim için kullanmaya hazır hale getirebilir ve dilediğiniz derin öğrenme çerçevesinde modelinizi eğitebilirsiniz. Ayrıca, modelinizi test edebilir ve dağıtabilirsiniz.
- Model İzleme ve Hata Analizi: Roboflow, eğitim sürecinde model performansını izlemenizi ve hata analizi yapmanızı sağlar. Modelin başarısını değerlendirebilir, yanlış etiketlemeleri kontrol edebilir ve performansı iyileştirmek için gerekli önlemleri alabilirsiniz.

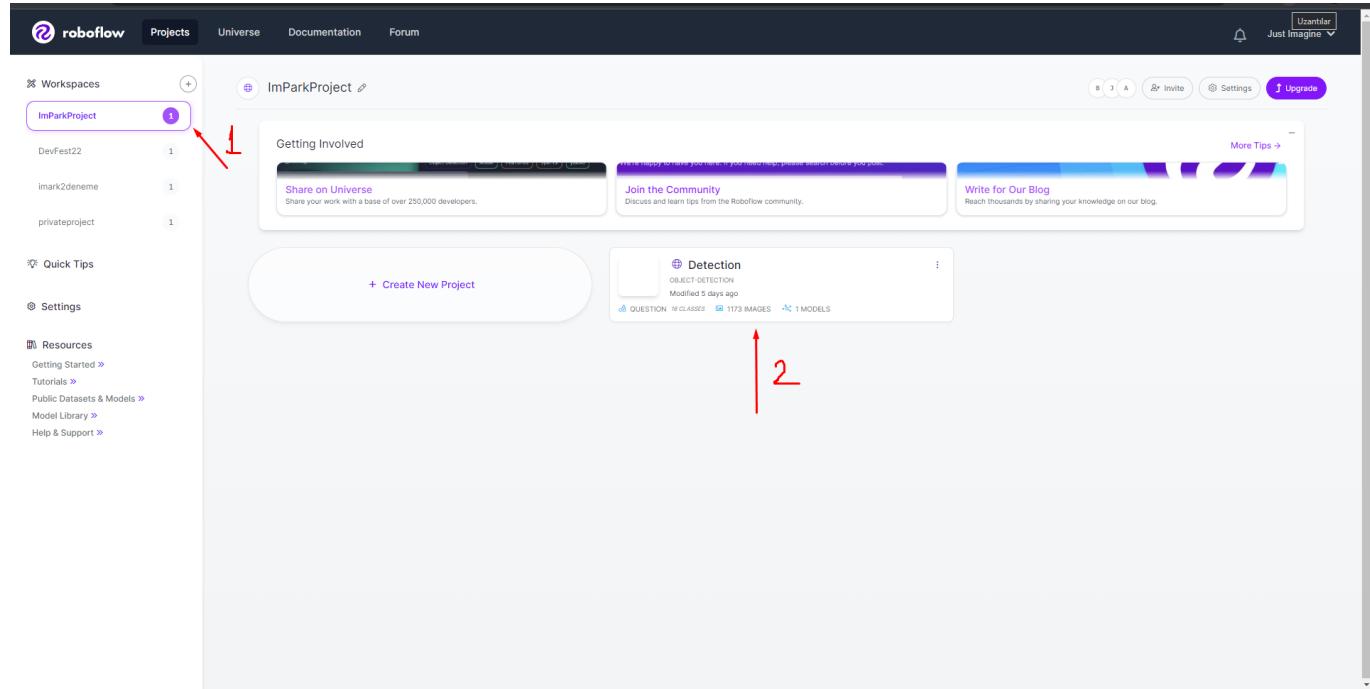
Ayrıca yazmak istediğim en önemli kısmı ise ekip çalışmasında son derece verimli olarak çalışmasıdır. Burada görevlendirmeleri yaparak task'ler oluşturabilmektedir. Başlıklarda ele alıyo olacağım.

2.2 Verilerin Ekleneceği

Bu adıma geçmeden önce model kaynaklı hatalara sahip olan sayfaların ayıklanması gerekmektedir. Bunun için ilerleyen aşamalarda yer alan yönergelere uymak ve bilgilerin alınması gereklidir. Program testleri kısmında bu konudan bahsedilecektir.

Öncelikle yetkili olan bir kullanıcının sizi platforma düzenleyici olarak eklemesi gerekmektedir. Bu işlem oldukça basittir yetkili kişinin proje ayarlarına girerek mail adresinizi sisteme tanımlamalıdır.

Platformumuz olan roboflow'a girdiğimizde 1 numara ile ok ile gösterilen Workspaces altındaki **ImParkProject** adlı workspace'e tıklıyoruz buradan 2 numaralı ok ile gösterilen **Detection** olarak açılmış projemize giriyoruz.



Bu işleminden sonra Karşımıza projenin ana sayfası çıkacaktır. Bu anasayfadan ok ile gösterilen Annotate[Etiketleme] kısmına giriş yapıyoruz.

The screenshot shows the Roboflow web interface for a project titled "IMPAKPROJECT". The left sidebar includes options like "Show Public View", "Detection", "Universe Page", "Upload", "Assign", "Annotate" (which is highlighted with a red arrow), "Dataset" (containing 1140 images), "Generate", "Versions" (31), "Deploy", and "Health Check". The main area displays "SAMPLES FROM TEST SET" with two images labeled "Question Box" and "Table". The right side shows "detection-ywfib/1" with metrics: 89.6% mAP, 86.1% precision, and 84.9% recall. A confidence threshold slider is set at 50%. The JSON output for detected objects includes:

```

{
  "predictions": [
    {
      "x": 211.5,
      "y": 237,
      "width": 43,
      "height": 34,
      "confidence": 0.757,
      "class": "answer_a"
    },
    {
      "x": 212.5,
      "y": 2181.5,
      "width": 39,
      "height": 31,
      "confidence": 0.737,
      "class": "answer_a"
    }
  ],
  ...
}
  
```

Bu bölüm üç bölüme ayrılmaktadır. **UNASSIGNED** olan bölümde yeni yüklenmiş ve hiç bir işlem yapılmamış veri görevlerini görebilmektedir.

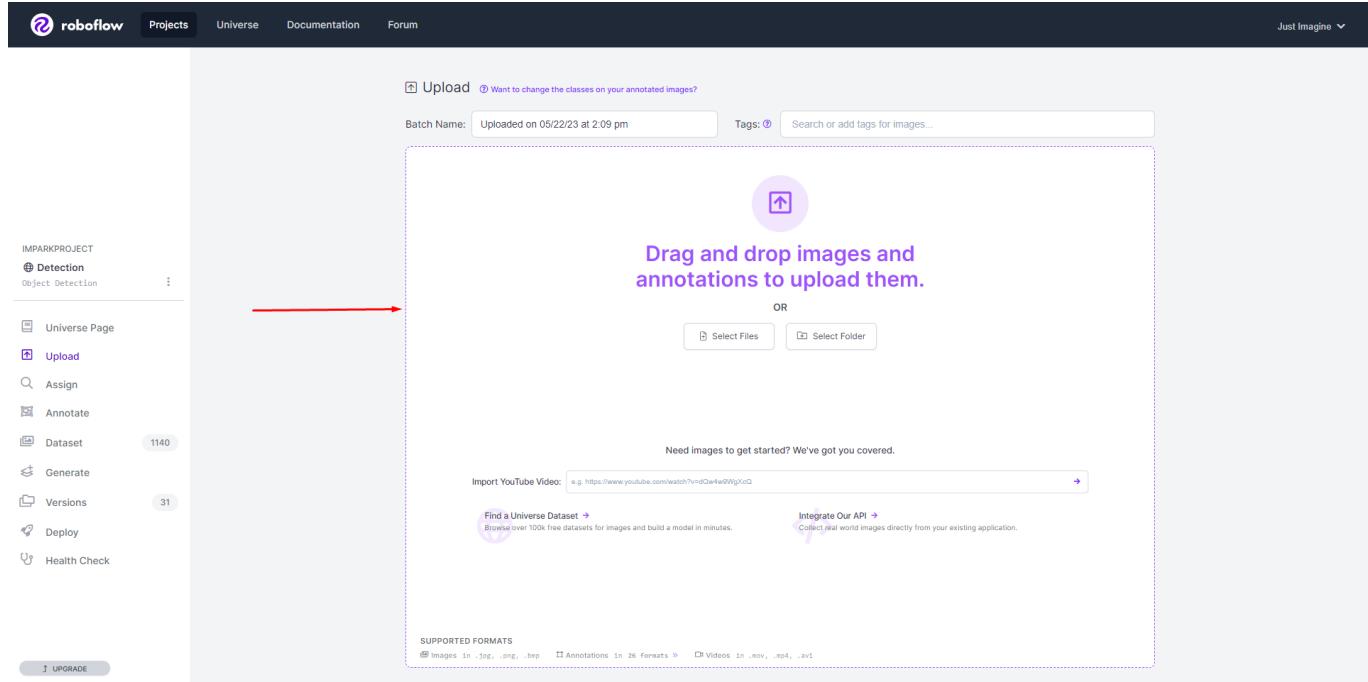
Ortada yer alan **ANNOTATING** kısmı etiketlenmeye başlanmış veya görev atanmış projelerin yer aldığı sütundur.

Sağ tarafta **DATASET** yazan kısmında etiketlenme görevi sonlandırılmış bloklar yer almaktadır.

Veri yüklemek için **UNASSIGNED** kısmına tıklayarak devam ediyoruz.

The screenshot shows the Roboflow "Annotate" section. The left sidebar is identical to the previous screenshot. The main area has three sections: "UNASSIGNED 0" (with a "Upload More Images" button), "ANNOTATING (4)" (listing four jobs with 3, 1, 14, and 14 images respectively), and "DATASET (54)" (listing 129 images uploaded from 10/06/22 to 02/21/23). A red arrow points from the "UNASSIGNED 0" section towards the "DATASET (54)" section.

Yüklemek istediğiniz verileri sürükleyebilir veya gerekli butonlara basarak seçiminizi sağlayabilirsiniz. Üst kısmında yer alan Batch Name kısmından görevinizi isimlendirebilir veya varsayılan tarih bazlı isimlendirmeyi kullanabilirsiniz.



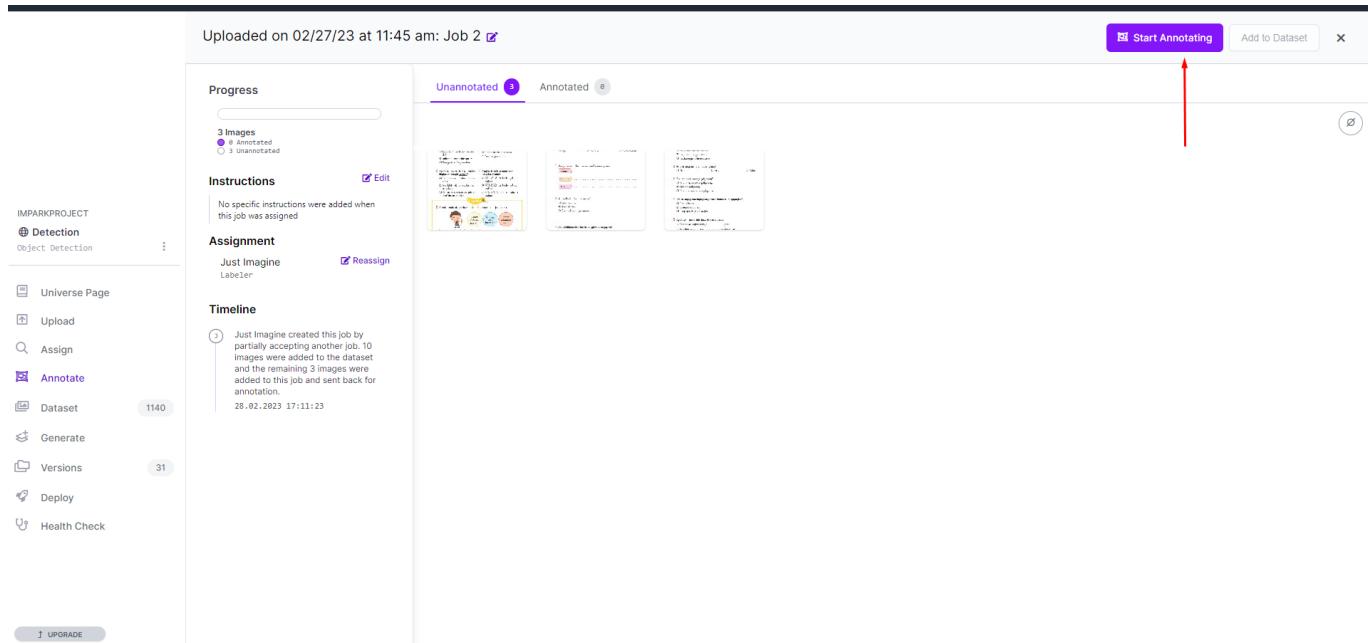
Tekrardan **ANNOTATE** sekmesine geri dönüyoruz. Oluşturduğumuz görev ismini bularak görevde girmekteyiz.

Bu kısımda yüklediğimiz verileri görüntülebilmekteyiz.

Bu kısımda ulaşabileceğimiz 2 kısım bulunmaktadır. Etiketlenmemiş veriler **Unannotated** kısmında; etiketlenmiş veya etiketlenmeye başlanmış veriler ise **Annotated** kısmında yer almaktadır.

Veri etiketleme aşamasına için herhangi bir resime tıklayabilir veya sağ üstte yer alan **Start Annotating** kısmından ulabiliyoruz.

UYARI: Veri etiketlemeniz eksik kaldıysa **Annotated** kısmında yer alacaktır bunun nedeni etiketlenmeye başlanmış olmasıdır.

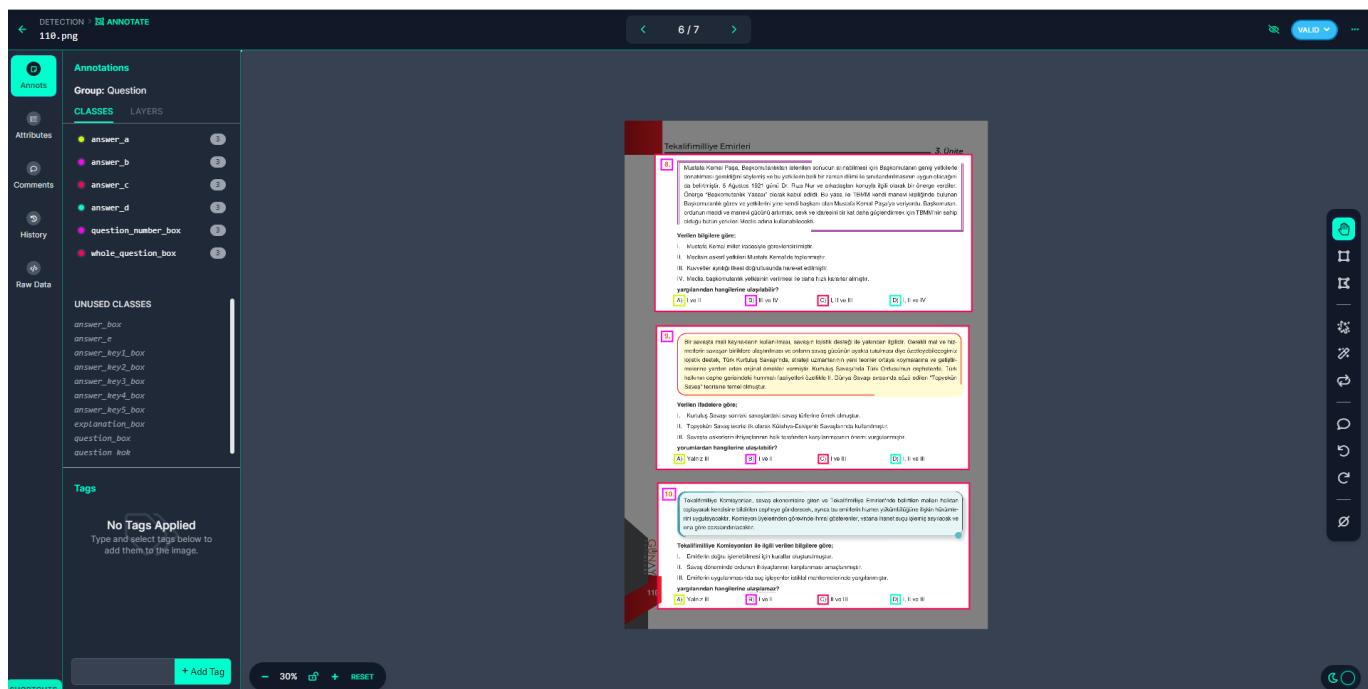


Etiketleme ekranına girdiğimizde karşımıza bir çok panel olan bir çalışma alanıyla karşılaşmaktayız. Bu ekran üzerinden etiketleme işlemlerini gerçekleştirip sınıflandırmalarımıza ayarlıyor olacağız.

BAŞLAMADAN ÖNCE DİKKAT EDİLMESİ GEREKENLER

Verileri etiketlerken bıaktığımız boşluk çok önemlidir. Burada genel olarak bir standart olmamakla beraber örneklemeleri inceleyerek yola çıkılabilir.

- Fazla boşluk veya çok yakından yapılan etiketlemler,
- Etketlenen verinin yanlış sınıflandırma ismiyle yer olması,
- Yamuk veya orantısız bir şekilde yapılan etiketlemler, Veri setinin bozulması gibi tehditlere yol açabilmektedir. Bütün bu maddelere dikkat ederek eğitim sonuçlarının daha verimliliğini ve eğitim sonrası doğruluk oranlarını artırlabiliz.



Aşağıda yer tabloda projenin barındırdığı sınıflar görüntülebilmekteyiz. Projede mevcut olarak bazı sınıflar kullanılmaktadır. Bu sınıfları ele alalım:

- answer_a
- answer_b
- answer_c
- answer_d
- answer_e
- question_number_box
- whole_question_box
- question_kok_box

Olarak ayrılmaktadır. Diğer sınıfların var olması ve henüz kullanılmamasının nedeni projede istenilen verilerin kısıtlanmış olmasından kaynaklıdır. Eğer ilerleyen versiyonlarında talep olursa bu sınıflar aktifleştirelerek kod kısmında entegre edilebilir hale gelebilir.

● answer_a	5
● answer_b	4
● answer_box	4
● answer_c	4
● answer_d	4
● answer_e	4
● answer_key1_box	3
● answer_key2_box	4
● answer_key3_box	4
● answer_key4_box	4
● answer_key5_box	4
● explanation_box	4
● question_box	3
● question_kok	1
● question_number_box	4
● whole_question_box	4

Kullanımda olan bu sınıfları ele alalım ve projeyi daha yakından tanıyarak temellere olan hakimiyetimizi arttıralım.

answer_a Bu sınıflar genel olarak soru şıklarını kapsamaktadır.

answer_b İlkokul, Ortaokul ve Lise kaynaklarında şıkların sayısının değiştiğini ve farklı

answer_c konumlarda geldiğini görebiliriz.

answer_d

answer_e

A) Yalnız III

B) I ve II

C) II ve III

D) I, II ve III

question_number_box Soruların genelde sol üst kısmında yer alan soru numarası sınıfıdır. Uygulamanın en önemli sınıflarından biridir. Bunun nedeni bütün sıralama ve uygulamanın temelinde yatan .xml çıktılarının oluşturulmasında önemli bir etkendir. Bu yüzden olabildiğince dikkatli kesimleri uygularak sınıflandırmamız lazımdır.

8.

whole_question_box Soruların dış çerçevesine verilen sınıflandırma adıdır. Bütün sınıfları içinde bulunduğu için soruların temelidir. Burada dikkat edilmesi gereken önemli noktalardan biri diğer sınıfların etiketleriyle belli oranda boşluk bırakılmasıdır. Biraz daha geniş tutularak içine bütün verileri alması sağlanır. Örneklemelerin incelenerek pekiştirilmesi önerilir.

4. Bu sureden;

- I. Allah (c.c.) evrenin idaresini yürütmektedir.
 - II. Kulluk görevinin dışında dayanışmaya da önem verilmelidir.
 - III. İnsanlara yaşadıkları olaylar karşısında sabır tavsiye edilmelidir.
 - IV. İbadetleri özüne uygun ihlasl yapmamız gereklidir.
- mesajlarından hangileri çıkarılabilir?**

A) I ve II

B) I ve III

C) II ve IV

D) III ve IV

question_kok soruları cevaplayınız ve farklı varyanslarla da karşımıza çıkabilen bağlantı soruların açıklamasının yer aldığı sınıfıdır. İngilizce veya Türkçe şeklinde gelebilen bu metinler ilerleyen kısımlarda OCR tarafından tespit edilip, bazı işlemlere sokulmaktadır. Ve XML'e bağlı soru olarak eklenmektedir.



Kullanım Tuşları

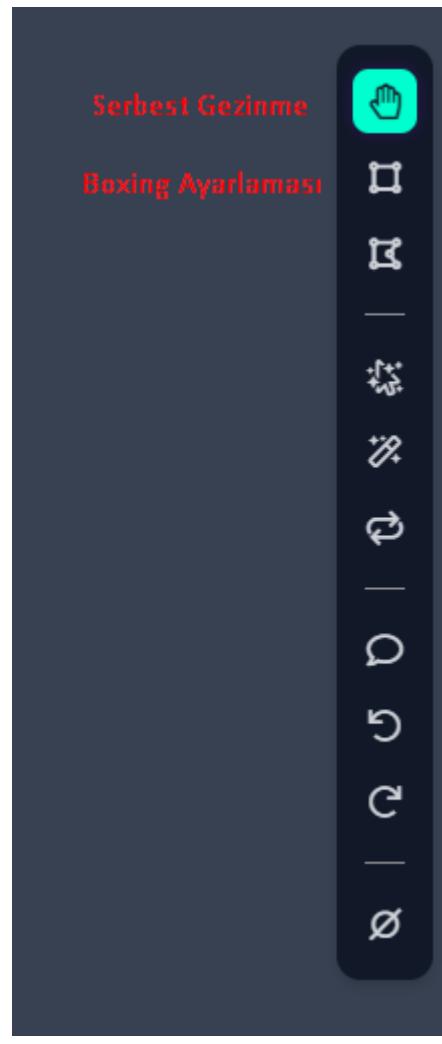
Arayüz üzerinde işlem yapabilmemize imkan tanıyan 2 adet tool vardır, bunlar:

- Drag Tool : Bu tool sayesinde veri üzerinde dolaşabiliriz, aynı zamanda bu tool'u CTRL'ye basılı tutarak yakınlaştırma ve uzaklaştırma için de kullanabilmekteyiz.

Kısayol tuşu '**D**' harfidir. Bu tuşa basarak Drag Tool'a geçiş sağlayabiliriz.

- Bounding Box Tool : Bu tool ile bütün etiketlemelerimizi gerçekleştirmektedir. Yukarıdaki yönergelere uygun bir şekilde işaretlemeleri bu tool sayesinde yapmaktadır.

Kısayol tuşu '**B**' harfidir. Bu tuşa basarak Bounding Box Tool'a geçiş sağlayabiliriz. Aşağıdaki resimde yer alan tool bar ile de bu ayarlara ulaşabilmekteyiz. Sayfanın sağ tarafında konumlandırılmış şekilde görebiliriz.



Etiketleme işlemleri bittikten sonra sayfanın sol üst kısmında yer alan geri gel butonuna basarak bir sayfa geri geliyoruz. Sağ üstte yer alan Add **n** image to Dataset butonuna basıyoruz.

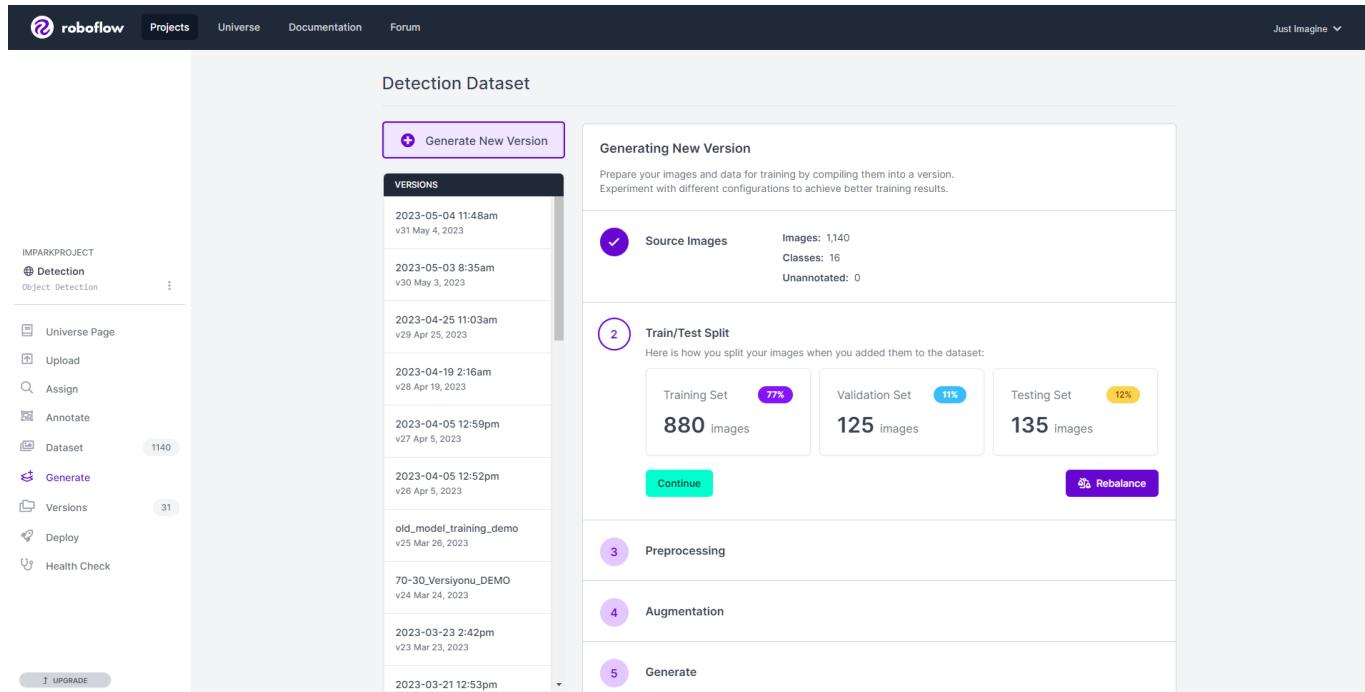
Burada veri setinin böülümlendirmesiyle ilgili bir bölüm bizi karşılaşacaktır. Şuanki bir işlem yapmayarak **Add Images** butonuna basarak devam ediyoruz.

Ekstra

Eğer önceden eğitilmiş ve hala hatalı sonuç aldığınız veriler varsa, bu verileri 100% oranda **Train** kısmına ekleyebilirsiniz.

2.3 Veri Setinin Oluşturulması

Bu işlemleri tamamladıktan sonra sol tarafta yer alan proje barından **Generate** kısmına geliyoruz. Bu aşamada dataset'i oluştururak Colab platformuna aktaracağız.



Versions sekmesi altında önceden oluşturulmuş setleri görüntülebilmekteyiz. Biz en üstte yer alan **Generate New Version** kısmını bu aşamada kullanacağız.

Öncelikle **Train/Test Split** kısmına geliyoruz. Bu aşamada setimizdeki gerekli ayırmaları yapacağız. Bu ayırlara geçmeden önce biraz bu kısmı tanıyalım.

Yolov7, nesne algılama modeli için kullanılan bir derin öğrenme algoritmasıdır. Train/Validation/Test ayrımı, modelin eğitilmesi, doğrulanması ve performansının değerlendirilmesi için veri kümelerinin bölünmesi anlamına gelir.

Bu ayırmın neden yapıldığına gelirsek:

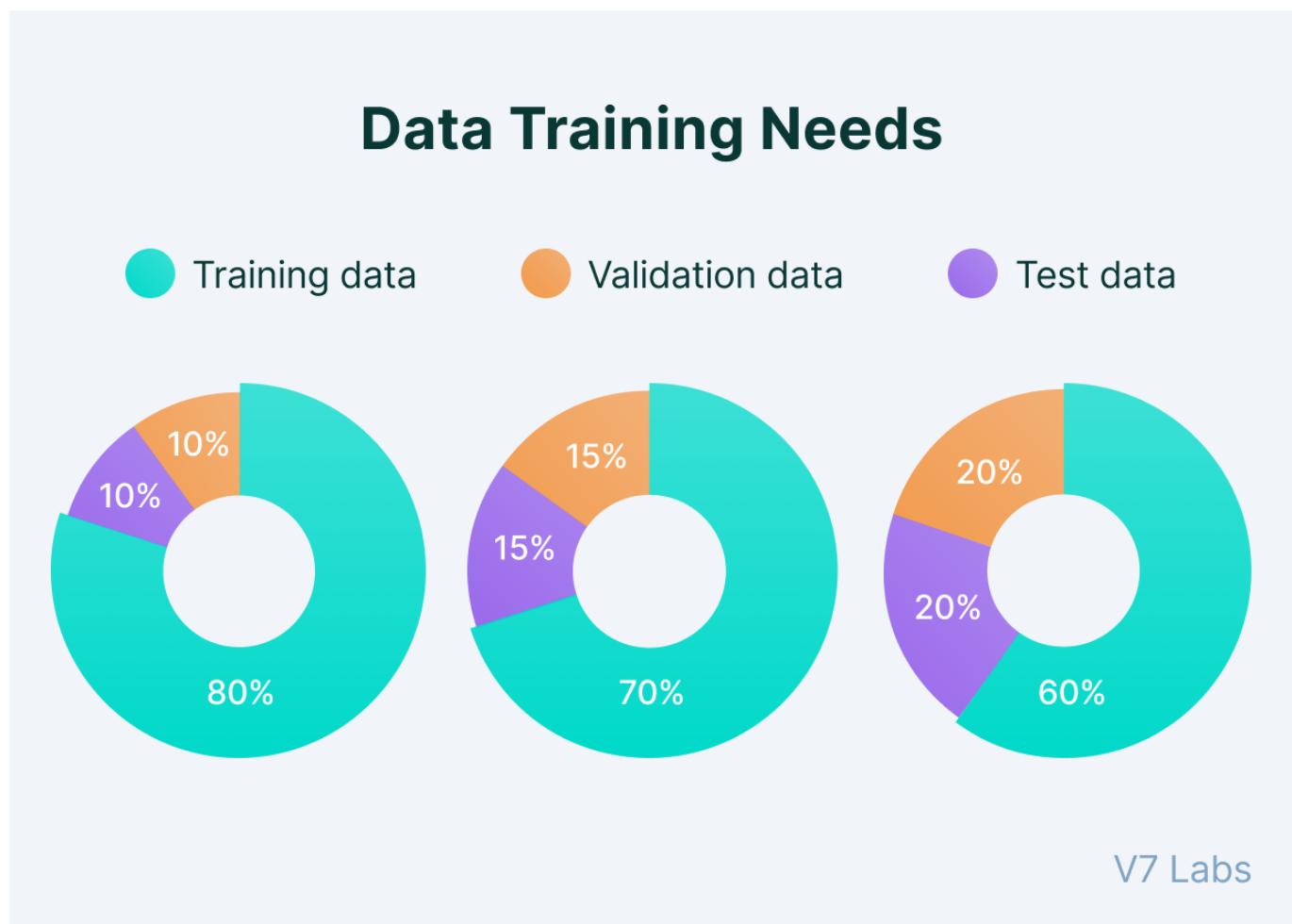
- **Eğitim (Train):** Eğitim veri kümesi, modelin öğrenmesi için kullanılır. Bu veri kümesi, genellikle etiketlenmiş görüntülerden oluşur. Yani her bir görüntüye ilişkin nesne etiketleri ve konum bilgileri bulunur. Model, bu veri kümesindeki görüntüler üzerinde iteratif olarak eğitilir ve nesne algılama görevini öğrenir. Eğitim veri kümesi genellikle en büyük veri kümesidir ve modelin en fazla zaman harcadığı aşamadır.
- **Doğrulama (Validation):** Doğrulama veri kümesi, modelin eğitim sırasında performansını izlemek ve hiperparametreleri ayarlamak için kullanılır. Eğitim veri kümesinden farklı olmalıdır ve modelin daha önce

görmediği verilere benzer olmalıdır. Bu, modelin genelleştirilebilirliğini değerlendirmek için önemlidir. Doğrulama veri kümesi, modelin doğruluğunu ve diğer performans metriklerini değerlendirirken kullanılır. Hiperparametre ayarlaması ve modelin iyileştirilmesi bu aşamada gerçekleştirilir.

- Test: Test veri kümesi, modelin performansının nihai olarak değerlendirildiği aşamadır. Modelin gerçek dünya verileri üzerindeki performansını ölçmek için kullanılır. Bu veri kümesi, modelin eğitim ve doğrulama sırasında görmediği tamamen yeni verilerden oluşur. Test veri kümesi, modelin genel performansını değerlendirmek ve diğer modellerle karşılaştırmak için kullanılır. Modelin test veri kümelerinde iyi performans göstermesi, genelleştirilebilir olduğunu gösterir.

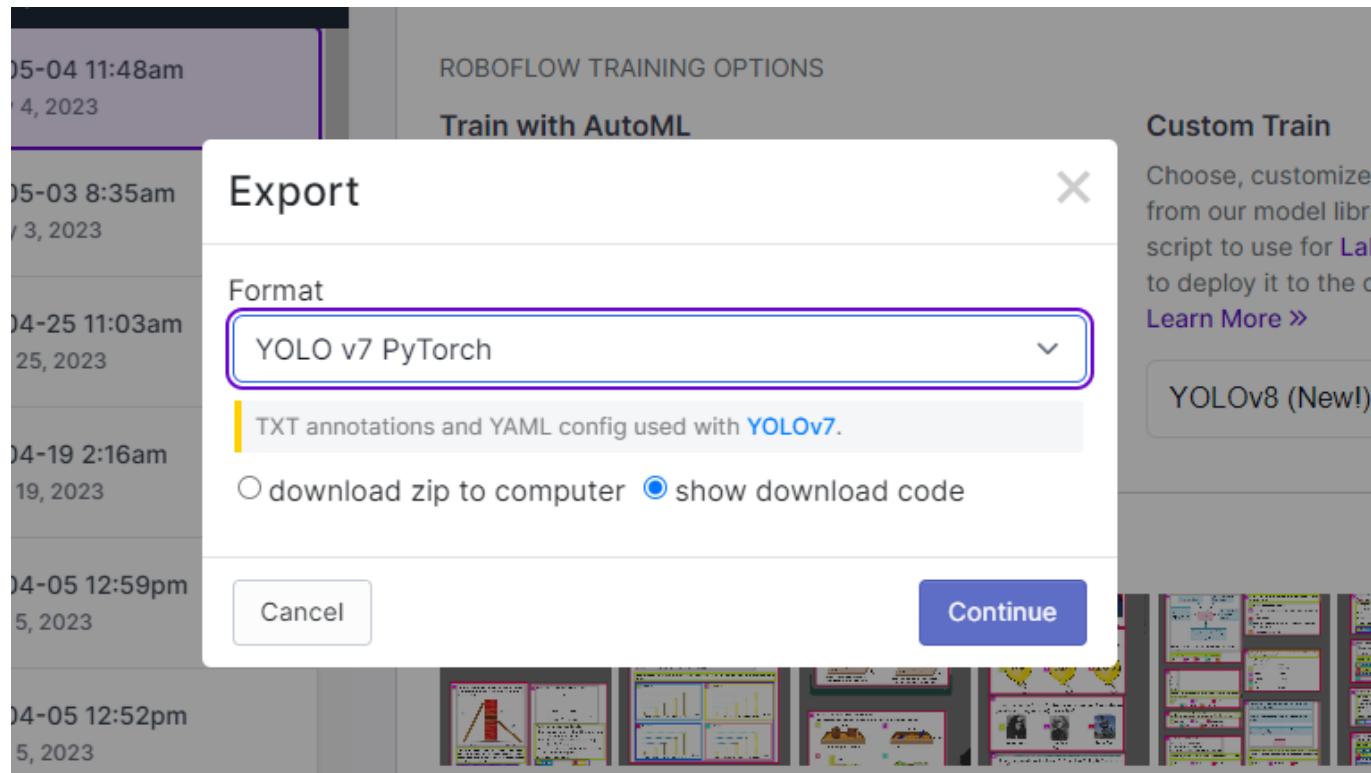
Bu üç ayrılmış, modelin verileri nasıl işlediğini ve performansını değerlendirdiğini kontrol etmek gereklidir. Eğitim veri kümesi, modelin öğrenmesi için kullanılırken, doğrulama veri kümesi hiperparametre ayarlaması ve modelin iyileştirilmesi için kullanılır. Test veri kümesi ise modelin genel performansının değerlendirilmesi için kullanılır. Bu ayrılmış, modelin gerçek dünya verileri üzerinde ne kadar iyi performans göstereceğini tahmin etmek için önemlidir.

Bu yüzden aşağıdaki grafiği kullanarak setlerimizi bu şekilde ayarlıyoruz.

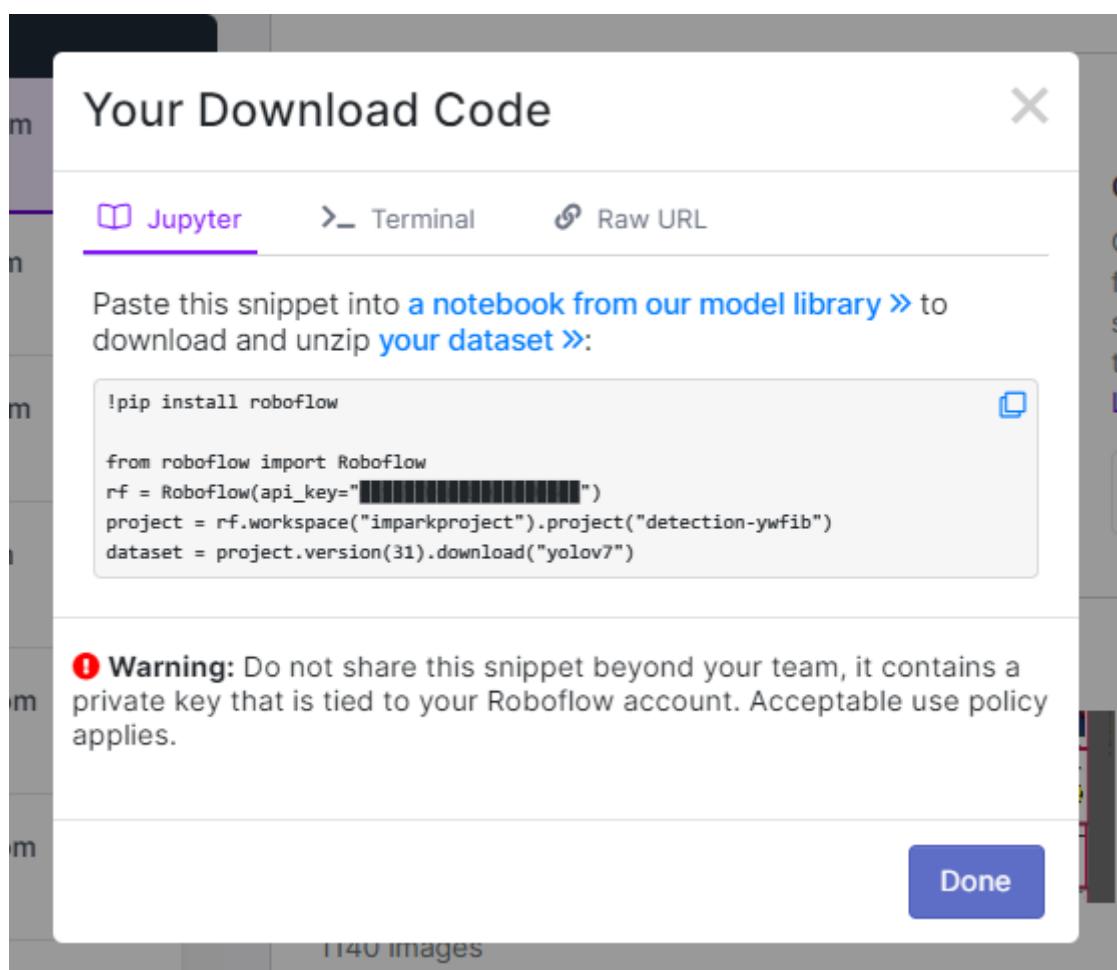


Projemiz üzerinde aktif olarak 80% - 10% - 10% böülümlendirmesini kullanmaktayız, daha verimli olması ve senaryomuza uygun bir ölçek olduğundan bu şekilde bir tercihten yana olduk.

Burada dikkat etmemiz gereken bir istisnamız vardır.



Format ayarını **Yolo V7 PyTorch**'a alarak **Show Download Code** kısmına tıklıyoruz, sonrasında devam ederek setimizi oluşturuyoruz.

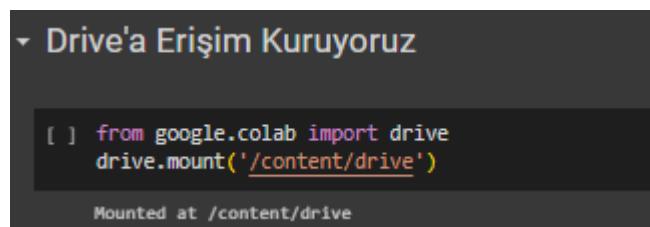


Karşımıza DataSeti Colab platformunda kullanmamız için gerekli kodlar çıkıyor. Bu aşamadaki kodumuzu bir sekmede bekletebiliriz, sonrasında tekrardan bu kodu kullanacağımız.

The screenshot shows the Google Colab pricing page. It features three main subscription options:

- Pay As You Go**: This plan is free and charges £162.84 for 100 Compute Units or £813.02 for 500 Compute Units. A message indicates you currently have 7784 compute units.
- Colab Pro**: This plan costs £162.84 per month. It includes:
 - 100 compute units per month (with a note about expiration)
 - Faster GPUs
 - More memory
 - Terminal
- Colab Pro+**: This plan costs £813.02 per month. It includes:
 - 500 compute units per month (with a note about expiration)
 - Faster GPUs
 - More memory
 - Background execution
 - Terminal

Eğer compute uniti kalmadıysa diğer GPU'ların performansları değerlendirilir ve seçim bu şekilde gerçekleştirilecektir.



İlk kod bloğunda yer alan kod parçası ile Drive'ımıza erişim vermektedir. Bu şekilde model dosyalarını kalıcı bir şekilde saklıyoruz.

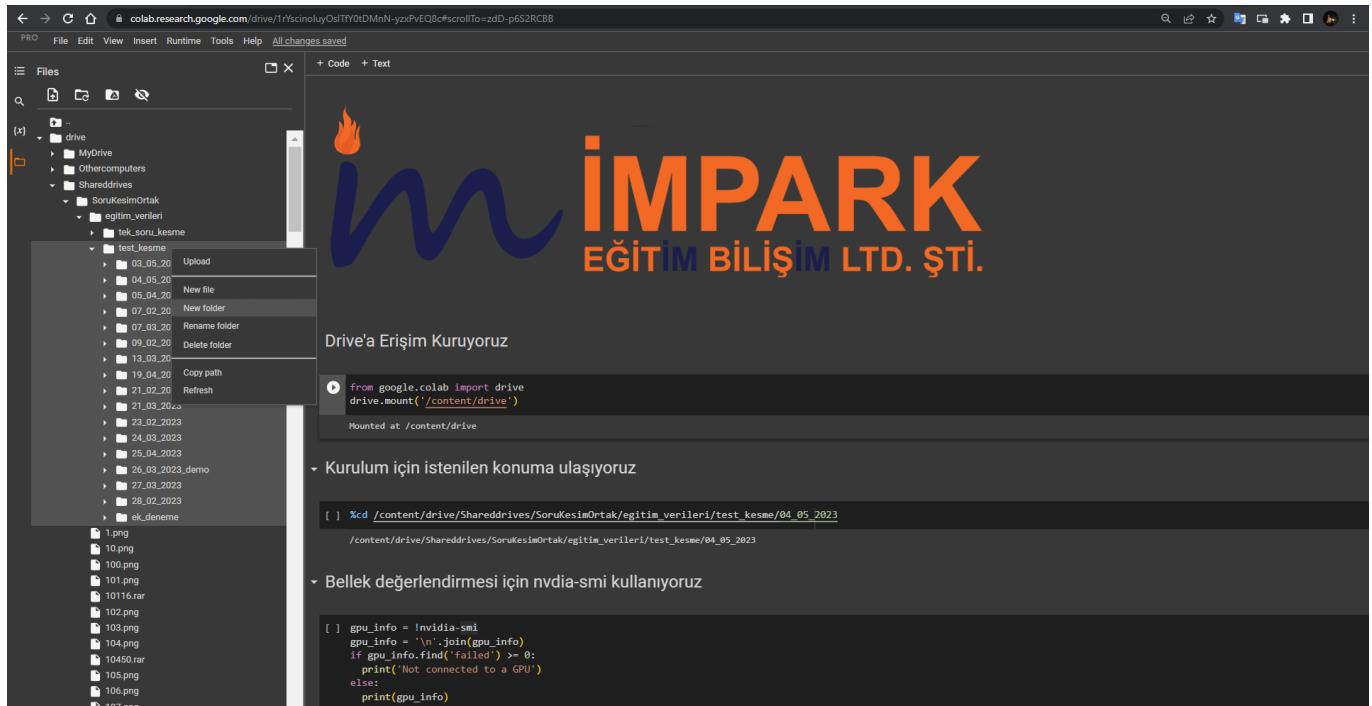
Gerekli izinleri verdikten sonra sol tarafta yer alan **Files** sekmesinin altında **drive** klasörünü görebilir hale geleceğiz.

Eğer bu kısım gözükmemezse yine aynı sekme altında olan klasör ve döngü iconun bulunduğu tuşa basarak dosyaların güncellenmesini sağlayabilmektedir.

Sonrasında kurumun sizi eklediği **Shareddrives** üzerinden **SoruKesimOrtak** drive'ına erişmemiz lazımdır.

Path = drive/Shareddrives/SoruKesimOrtak/egitim_verileri/test_kesme

Yukarıdaki lokasyona ulaşıyoruz. Sonrasında dosyalarımızın kaydedileceği güncel bir klasör yapısı oluşturuyoruz.



İlgili konuma geldikten sonra sağ tıklayarak, eğitimin yapıldığı tarihi bu kısma giriyoruz. Sonrasında klasöre tekrardan sağ tıklayarak **Copy Path** seçeneğine tıklıyoruz.

Bu aldığımız lokasyonu %cd {Copy Path} şeklinde yazıyoruz. Bu komut yapacağımız işlemleri bu klasör içerisinde yapmamıza olanak sağlayacaktır.

▼ Bellek değerlendirmesi için nvidia-smi kullanıyoruz

```
[ ] gpu_info = !nvidia-smi
gpu_info = '\n'.join(gpu_info)
if gpu_info.find('failed') >= 0:
    print('Not connected to a GPU')
else:
    print(gpu_info)
```

```
Wed May 31 05:20:46 2023
+-----+
| NVIDIA-SMI 525.85.12      Driver Version: 525.85.12      CUDA Version: 12.0 |
+-----+
| GPU  Name      Persistence-M| Bus-Id      Disp.A  | Volatile Uncorr. ECC | |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util  Compute M. |
|                               |             |            | MIG M. |
+-----+
| 0  NVIDIA A100-SXM... Off   | 00000000:00:04.0 Off |          0 |
| N/A   30C     P0    44W / 400W |        0MiB / 40960MiB |     0%  Default |
|                               |                  |            Disabled |
+-----+
+
| Processes:
| GPU  GI  CI      PID  Type  Process name                    GPU Memory |
| ID   ID
+-----+
| No running processes found
+-----+
```

Colab'a girişte yaptığımız GPU seçiminin ayrıntılarına bu kod satırından ulaşabilmekteyiz ve değerlendirmelerimizi sağlayabilmektedir.

▼ Model Dosyalarının Aktarılması

```
[ ] !git clone https://github.com/WongKinYiu/yolov7
%cd yolov7
!pip install -r requirements.txt
```

Sonrasında Yolo dosyalarını dizinimize klonluyoruz ve gerekli kütüphaneleri entegre ediyoruz.

▼ Veri Setinin Entegrasyonu

```
%cd /content/drive/Shareddrives/SoruKesimOrtak/egitim_verileri/test_kesme/04_05_2023/yolov7
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="61vhYY03mNSDFJHU0Bqk")
project = rf.workspace("imparkproject").project("detection-ywfib")
dataset = project.version(31).download("yolov7")
```

Bir önceki konu başlığı altında yaptığımız **Etiketleme** bölümünde elde ettiğimiz kod satırını bu kısma ekliyoruz.

Dikkat etmemiz gereken bir başka nokta %cd kod satırıdır, Yolov7 dosyalarını klonladıkten sonra tekrardan yolov7 klasörünün içerisinde ulaşmamız gerekmektedir. Burada tekrardan sol taraftan path'i alabilir veya en son aldığınız path'in son kısmına '**/yolov7**' ekleyebilirsiniz.

▼ Wandb kütüphanesinin kurulurak veri takibinin izlenmesi

```
[ ] !pip install wandb
```

API CODE

```
1a69fde257488395fd768388a2ea651422b45c72
```

▼ Veri Eğitimi

```
# Custom Eğitim Denemesi
%cd /content/drive/Shareddrives/SoruKesimOrtak/egitim_verileri/test_kesme/04_05_2023/yolov7
!python train.py --batch 40 --epochs 350 --data {dataset.location}/data.yaml --weights 'best.pt' --device 0
```

Eğitimi başlatmadan önce verilerimizin takibi için gerekli olan wandb kütüphanesini ekliyoruz.

Wandb, "Weights and Biases" olarak da bilinen bir Python kütüphanesidir. Bu kütüphane, makine öğrenimi ve derin öğrenme projeleri için model eğitimi, hiperparametre optimizasyonu, model takibi ve deneysel sonuçların görselleştirilmesi gibi işlemleri kolaylaştırmayı amaçlamaktadır.

Sonrasında eğitim kod bloğuna gidiyoruz burada dikkat etmemiz gerekenler:

- Batch: GPU'umuzun var olan VRAM'ini bu kısıma girerek eğitim sürecinde hedef alacağı belleği belirtiyoruz.
- Epochs: Eğitimin süreceği adım sayısıdır. Bu kısım deney sürecinde edindiğimiz verilerle orantılı olarak değişmektedir. Eğer grafik yorumlamasında verilerin eksik kaldığını görürsek veya overfitte olduğunu tespit edersek arttirma - azaltma işlemlerini uygulabiliriz.
- Weights: Model ağırlığını belirttiğimiz kısımdır. Eğitimin hangi verileri alacağını gösteren bir checkpointtir.

Bu kısımda **Başarılı** bir şekilde sonlanmış bir önceki eğitime ait **best.pt** modelini checkpoint dosyası olarak kullanacağız. Bu veriye ulaşmak için:

Path: drive/Shareddrives/SoruKesimOrtak/egitim_verileri/test_kesme/BİR ÖNCEKİ OLUŞTURULMUŞ TARİH/yolov7/runs/train/exp/weights

Bu kısımda sol tarafta yer alan listeden **best.pt**'yi yeni konuma sürükleyebilir veya drive üzerinden bu konuma ulaşarak dosya kopyalanabilir ve yeni konuma aktarılır isim düzenlemesinden sonra kullanımına alabilir.

Bu işlemleri yaptıktan sonra eğitime başlıyoruz.

Kod bloğunu çalıştırıldıktan sonra **Wandb** kütüphanesi bize bir kaç soru soracaktır. Bu sorulara aşağıdaki gibi cevap vermemiz ve gerekli API'yi eklememiz gerekmektedir.

Veri Eğitimi

```
# Custom Eğitim Denemesi
%cd /content/drive/Shareddrives/SoruKesimOrtak/egitim_verileri/test_kesme/04_05_2023/yolov7
!python train.py --batch 16 --epochs 350 --data {dataset.location}/data.yaml --weights 'best.pt' --device 0

...
/content/drive/Shareddrives/SoruKesimOrtak/egitim_verileri/test_kesme/04_05_2023/yolov7
2023-05-31 06:30:56.513092: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions to best utilize the performance of your system. To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-05-31 06:30:57.610097: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT YOLOR ✨ v0.1-122-g3b41c2c torch 2.0.1+cu118 CUDA:0 (Tesla V100-SXM2-16GB, 16150.875MB)

Namespace(weights='best.pt', cfg='', data='/content/drive/Shareddrives/SoruKesimOrtak/egitim_verileri/test_kesme/04_05_2023/yolov7/Detection-31/datasets',
tensorboard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
hyperparameters: lr=0.01, lrf=0.1, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.01
wandb: (1) Create a W&B account
wandb: (2) Use an existing W&B account
wandb: (3) Don't visualize my results
wandb: Enter your choice: 2
```

API CODE

1a69fde257488395fd768388a2ea651422b45c72

Veri Eğitimi

```

# Custom Eğitim Denemesi
%cd /content/drive/Shareddrives/SoruKesimOrtak/egitim_verileri/test_kesme/04_05_2023/yolov7
!python train.py --batch 16 --epochs 350 --data {dataset.location}/data.yaml --weights 'best.pt' --device 0

...
```

 # Custom Eğitim Denemesi
 %cd /content/drive/Shareddrives/SoruKesimOrtak/egitim_verileri/test_kesme/04_05_2023/yolov7
 !python train.py --batch 16 --epochs 350 --data {dataset.location}/data.yaml --weights 'best.pt' --device 0

 ...
```


```

```

# Custom Eğitim Denemesi
%cd /content/drive/Shareddrives/SoruKesimOrtak/egitim_verileri/test_kesme/04_05_2023/yolov7
!python train.py --batch 40 --epochs 350 --data {dataset.location}/data.yaml --weights 'best1.pt' --device 0

...
```

 # Custom Eğitim Denemesi
 %cd /content/drive/Shareddrives/SoruKesimOrtak/egitim_verileri/test_kesme/04_05_2023/yolov7
 !python train.py --batch 40 --epochs 350 --data {dataset.location}/data.yaml --weights 'best1.pt' --device 0

 ...
```


```

Eğitimimiz artık başlamış olup işaretle gösterilen linkten verilerimizi izleyebiliriz. Bu aşamada uymamız ve takip etmemiz gereken parametrelere ait bilgilere aşağıda belirtilen dokumantasyonlardan ulaşabiliriz.

Youtube üzerinde yer alan bir eğitim *Weights & Biases Guide*.

Dokumantasyon -1- [Dokumantasyon](#).

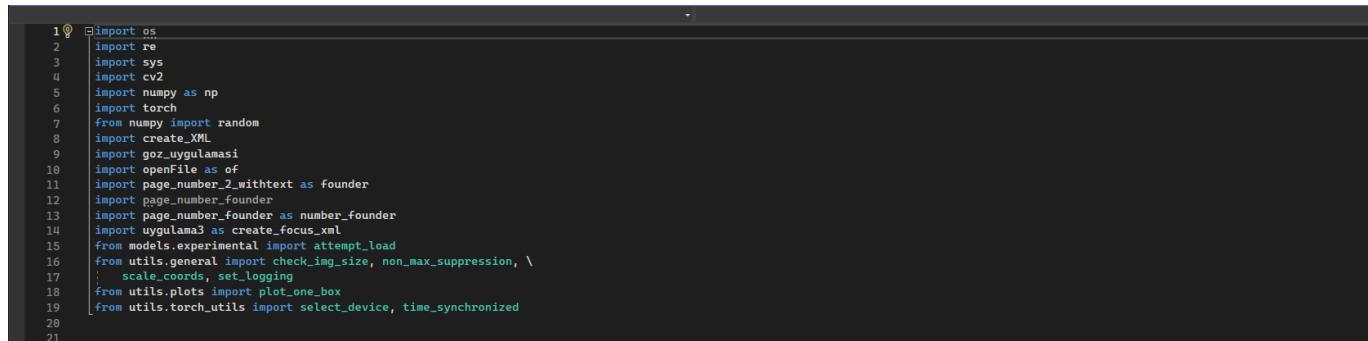
Dokumantasyon -2- [Dokumantasyon](#).

Eğitim sonlandıktan sonra elde ettiğimiz sonuçları somut olarak tekrardan görüntülebilmektedir.

Bu dosya üzerinden yönetilen kod blokları şu şekildedir:

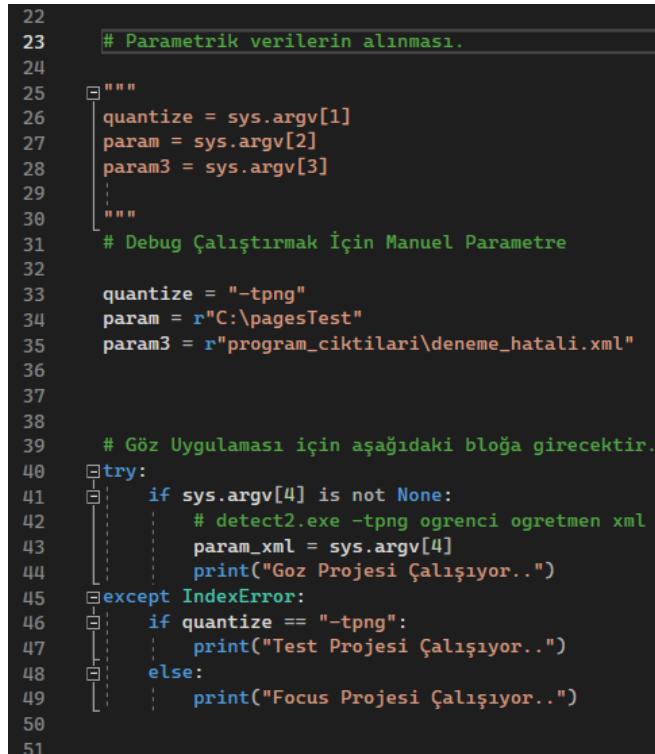
```
└── detect2.py
    ├── page_number_founder.py
    ├── openFile.py
    ├── create_XML.py
    └── openFile.py
```

`detect2.py` üzerinden kodlarımızı tanımlamaya başlıyoruz.



```
1 import os
2 import re
3 import sys
4 import cv2
5 import numpy as np
6 import torch
7 from numpy import random
8 import create_XML
9 import goz_ugulaması
10 import openfile as of
11 import page_number_2_withtext as founder
12 import page_number_Founder
13 import page_number_Founder as number_founder
14 import ugulama3 as create_focus_xml
15 from models.experimental import attempt_load
16 from utils.general import check_img_size, non_max_suppression, \
17     scale_coords, set_logging
18 from utils.plots import plot_one_box
19 from utils.torch_utils import select_device, time_synchronized
20
21
```

Gerekli kütüphaneleri ve yan fonksiyonları import ediyoruz.



```
22
23     # Parametrik verilerin alınması.
24
25 """
26 quantize = sys.argv[1]
27 param = sys.argv[2]
28 param3 = sys.argv[3]
29 """
30
31 # Debug Çalıştırmak İçin Manuel Parametre
32
33 quantize = "-tpng"
34 param = r"C:\pagesTest"
35 param3 = r"program_ciktiları\deneme_hatali.xml"
36
37
38
39     # Göz Uygulaması için aşağıdaki bloğa girecektir.
40 try:
41     if sys.argv[4] is not None:
42         # detect2.exe -tpng ogrenci ogretmen xml
43         param_xml = sys.argv[4]
44         print("Göz Projesi Çalışıyor..")
45 except IndexError:
46     if quantize == "-tpng":
47         print("Test Projesi Çalışıyor..")
48     else:
49         print("Focus Projesi Çalışıyor..")
50
51
```

Bu kısımda yer alan kodlar uygulamayı konsol üzerinden başlatmamıza yarayan kodlardır. Üst kısımda açıklama satırı ile belirtilen kod bloğu uygulama canlıya çıktığında açık kalması gereken kısımdır, burayı aktif ettiğimizde alt tarafı deaktif etmemiz gerekmektedir.

Eğer test yapmak ve Debug modda kodları denetlemek istiyorsak bu sefer alt kısmı kullanıma alıp üst kısmı açıklama kısmına almamız gerekmektedir.

Test Tespit Ve Kesim Programı; 3 parametre kullanmaktadır, bunlar:

python detect2.py -tpng kitap_lokasyonu xml_adi_ve_lokasyonu şeklinde uygulamamızı konsol üzerinden çalıştırabiliriz.

- quantize: Bu kısımda uygulamanın tipini belirliyoruz. Eğer uygulama da uygulamasını kullanmak istiyorsak **-tpng**, Göz Uygulamasını kullanmak istiyorsak **-gpng**, Focus uygulamasını kullanmak istiyorsak **-fpng** değerini kullanmamız gerekmektedir.
- param: Bu kısımda ise .png olarak oluşturulmuş kitap klasör yapısının yolu belirtilmelidir.
- param3: Uygulamanın çıktısı olan .xml yapısının lokasyonu ve ismi bu kısımda verilmelidir.

Yukarıda verilen kod çıktısına bu kısım entegre edilerek kullanabilir hale gelebilir. (Sadece test için kullanılabilir.)

```

52
53     def letterbox(image, new_shape=(640, 640), color=(114, 114, 114), auto=True, scaleFill=False, scaleup=True, stride=32):
54         # Çoklu kısıtlamalarını karşıtlarken görüntüyü yeniden boyutlandırır ve doldur
55         shape = image.shape[:2] # Anlık boyutlandırma [height, width]
56         if isinstance(new_shape, int):
57             new_shape = (new_shape, new_shape)
58
59         # Scale ratio (new / old)
60         r = min(new_shape[0] / shape[0], new_shape[1] / shape[1])
61         if not scaleup: # Sadece küçült, büyütme ! (for better test mAP)
62             r = min(r, 1.0)
63
64         # Compute padding
65         ratio = r, r # width, height ratios
66         new_unpad = int(round(shape[1] * r)), int(round(shape[0] * r))
67         dw, dh = new_shape[1] - new_unpad[0], new_shape[0] - new_unpad[1] # wh padding
68         if auto: # minimum rectangle
69             dw, dh = np.mod(dw, stride), np.mod(dh, stride) # wh padding
70         elif scaleFill: # stretch
71             dw, dh = 0.0, 0.0
72             new_unpad = (new_shape[1], new_shape[0])
73             ratio = new_shape[1] / shape[1], new_shape[0] / shape[0] # width, height ratios
74
75         dw /= 2 # divide padding into 2 sides
76         dh /= 2
77
78         if shape[::-1] != new_unpad: # resize
79             image = cv2.resize(image, new_unpad, interpolation=cv2.INTER_LINEAR)
80             top, bottom = int(round(dh - 0.1)), int(round(dh + 0.1))
81             left, right = int(round(dw - 0.1)), int(round(dw + 0.1))
82             image = cv2.copyMakeBorder(image, top, bottom, left, right, cv2.BORDER_CONSTANT, value=color) # add border
83
84         return image, ratio, (dw, dh)

```

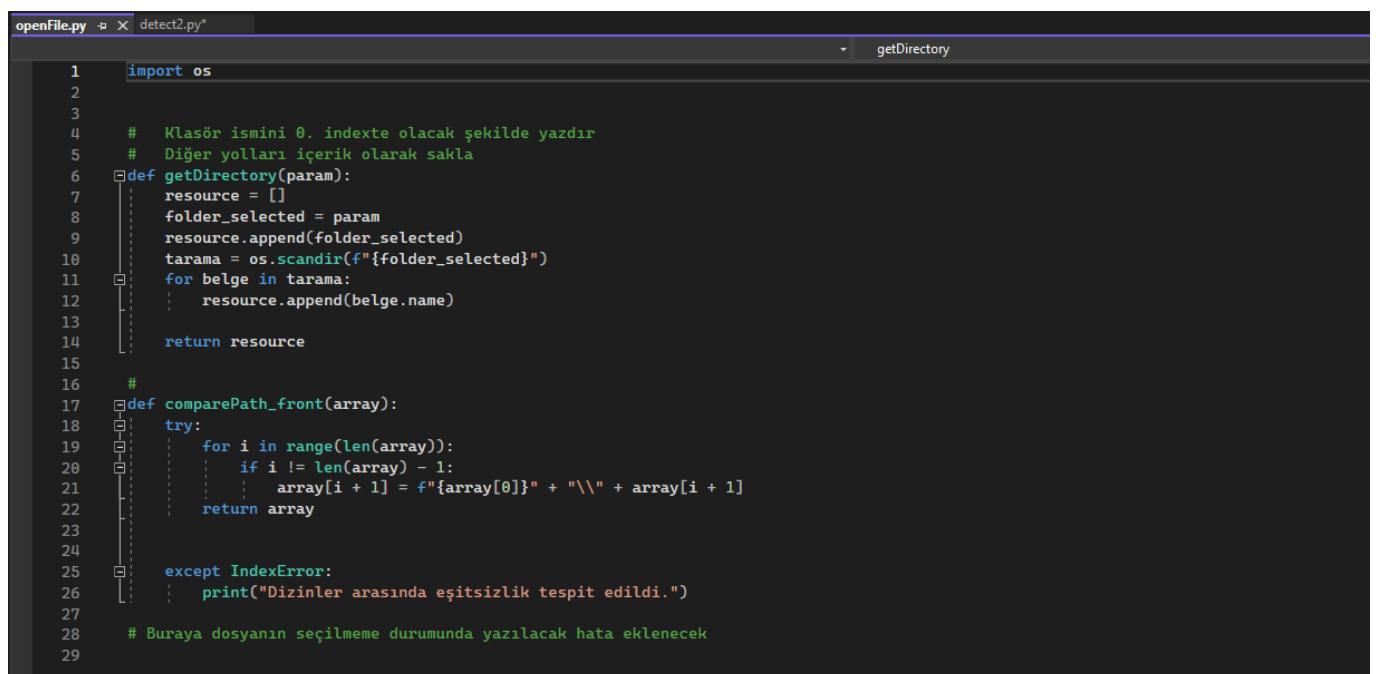
Bu kısımda yer alan kod bloğu sabit bir fonksiyon olup değişikliliğe ihtiyaç duyulmamaktadır. Input verisinin yolo için gerekli olan formata sokulması için yazılmış bir fonksiyon bloğudur.

```
if quantize == "-tpng":  
    whole_x1, other_x1 = [], []  
    whole_x2, other_x2 = [], []  
    whole_y1, other_y1 = [], []  
    whole_y2, other_y2 = [], []  
    where_counter = 0  
    my_dict = {'whole_question_box': [], 'question_number_box': [], 'answer_box': [], 'answer_a': [], 'answer_b': []},  
    'answer_c': [],  
    'answer_d': [], 'answer_e': [], 'page_number': [], 'answer_key1_box': [], 'answer_key2_box': [],  
    'answer_key3_box': [],  
    'answer_key4_box': [], 'answer_key5_box': [], 'question_box': [], 'explanation_box': [], 'question_kok': []}  
my_dict_copy = {'whole_question_box': [], 'question_number_box': [], 'answer_box': [], 'answer_a': [],  
    'answer_b': [], 'answer_c': [],  
    'answer_d': [], 'answer_e': [], 'page_number': [], 'answer_key1_box': [], 'answer_key2_box': [],  
    'answer_key3_box': [],  
    'answer_key4_box': [], 'answer_key5_box': [], 'question_box': [], 'explanation_box': [], 'question_kok': []}  
arl, ar2, ar3, ar4, ar5, ar6, ar7, ar8, ar9, ar10, ar11, ar12, ar13, ar14, ar15, ar16, ar17 = [], [], [], [], [], [], [], [], [],  
[], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], []  
soru_lokasyon = []  
  
compareXMLArray = []  
my_labels = []  
condition = True  
check1, check2, check3, check4, check5, check6, check7, check8, check9, check10, chech17 = False, False, False, False, False, False, False, False, False,  
# page_number_array = []  
question_number_array2 = []  
page_text_top = []  
image_size_xml = []  
dogrulama = False  
question_test2, question_test3 = 0, 0  
main_file = ""
```

Proje içerisinde kullanacağımız gerekli değişkenleri programın başında oluşturarak referans gösteriyoruz.

```
117         file_resource = of.getDirectory(param)  
118         jpg_files = [file for file in file_resource if file.endswith('.jpg')]  
119         png_files = [file for file in file_resource if file.endswith('.png')]  
120         jpg_files.sort(key=lambda x: int(re.search(r'\d+', x).group()))  
121         png_files.sort(key=lambda x: int(re.search(r'\d+', x).group()))  
122         main_file = file_resource[0]  
123         file_resource = []  
124         file_resource.append(main_file)  
125         file_resource.extend(jpg_files)  
126         file_resource.extend(png_files)  
127         print("Dosya Uzantınız =>", file_resource)  
128         file_resource = of.comparePath_front(file_resource)  
129  
130
```

Aşağıda openFile.py(of) yapısını görebilmektedir.



```
openFile.py ✘ × detect2.py*  
1  import os  
2  
3  
4  # Klasör ismini 0. indexte olacak şekilde yazdır  
5  # Diğer yolları içerik olarak sakla  
6  def getDirectory(param):  
7      resource = []  
8      folder_selected = param  
9      resource.append(folder_selected)  
10     tarama = os.scandir(f"{folder_selected}")  
11     for belge in tarama:  
12         resource.append(belge.name)  
13  
14     return resource  
15  
16     #  
17     def comparePath_front(array):  
18         try:  
19             for i in range(len(array)):  
20                 if i != len(array) - 1:  
21                     array[i + 1] = f"{array[0]}\" + "\\\" + array[i + 1]  
22             return array  
23  
24         except IndexError:  
25             print("Dizinler arasında eşitsizlik tespit edildi.")  
26  
27     # Buraya dosyanın seçilmeme durumunda yazılacak hata eklenecek  
28
```

Devam ettiğimizde burada **of** kütüphanesine başvurduğumuz bir fonksiyon yer alıyor. Bu fonksiyon sayesinde tek bir parametre olarak gelen path değeri içerisinde verileri parçalara ayıriyoruz ve bir diziye atarak ana

dosyamızı iletiyoruz.

Devamında belirtilen formatlarda dosya uzantılarına sahip dosyalrai ayıklıyoruz ve dosya biçimlerinin karışmaması ve anlık gelen formatlara göre şekillenen bir yapı oluşturuyoruz. Bu sayede .png veya .jpg formatında karmaşık bir dizi elde etmemiş oluyoruz.

```

131
132     while condition:
133         for k in range(1, len(file_resource)):
134             dogrula1, dogrula2, dogrula3, dogrula4, dogrula5 = 0, 0, 0, 0, 0
135             kokuoSorular = []
136
137             # Soruları son bir kontrole sokarak kalibrenin kontrolü yapılır.
138             lastCheck = 0
139             question_test1 = 0
140             question_check = 0
141             deneme = k
142             url1 = file_resource[k]
143             url1 = url1.replace('/', '\\\\')
144             print("Suanda okunan dosya adı =>", url1)
145             if url1.endswith(".png"):
146                 with open(url1, "rb") as f:
147                     data = f.read()
148                     img1 = cv2.imdecode(np.frombuffer(data, np.uint8), cv2.IMREAD_UNCHANGED)
149             else:
150                 img1 = cv2.imread(url1)
151             # img1 = cv2.imread(url1)
152             # cv2.imwrite("asd.jpg", img1)
153
154             new_width = img1.shape[1] # 2424
155             new_height = img1.shape[0] # 1701
156
157             dsize_h = (new_width, img1.shape[0])
158             dsize_v = (img1.shape[1], new_height)
159
160             img_sakla = cv2.resize(img1, dsize_h, interpolation=cv2.INTER_AREA)
161             img_sakla = cv2.resize(img1, dsize_v, interpolation=cv2.INTER_AREA)
162
163             img1 = cv2.resize(img1, dsize_h, interpolation=cv2.INTER_AREA)
164             img1 = cv2.resize(img1, dsize_v, interpolation=cv2.INTER_AREA)
165             imgsize = img1.shape
166             x1, y1, x2, y2 = int((imgsize[1] * 2) / 100), 0, int((imgsize[1] - (imgsize[1] * 2) / 100)), int(
167                 (imgsize[0] * 15) / 100)
168             cropped_image = img1[y1:y2, x1:x2]
169             text_ust = number_founder.test_algoritması(cropped_image)
170             page_text_top.append(text_ust)
171
172             Test veya Deneme Sayısının Tespiti İçin
173

```

Bu aşamada bir döngü oluşturuyoruz ve resimlerimizi sırasıyla işleme sokuyoruz. Gelen resimleri bir preprocessing uygulayarak stabil bir hale getiriyoruz.

Sonrasında yer alan **text_ust** değişkeni ile kitapların üst kısmında yer alan **test** veya **deneme** gibi yazıların tespit edilip sonrasında bu yazıların yanındaki sayıların çekilmesi üzerine yazılmış bir fonksiyondur. Bu fonksiyona parametre yollarken belirlenen ölçeklerde input resmi kesilir ve sonrasında yeni bir veri olarak kaydedilir.

Aşağıda gözüktüğü üzere PyTesseractOCR'in kullandığı ve bir regex algoritması yer alan bir fonksiyon görmekteyiz. Bu şekilde üst taraftaki yazılar tespit edilebilmektedir.

```

page_number_founder.py  openFile.py  detect2.py*  test_algoritması
68
69     def test_algoritması(image):
70         condition = True
71         counter = 0
72         integer_list, int_range, missing_ints, empty_string_index, next_missing = [], {}, [], [], 0
73
74
75         image = cv2.resize(image, None, fx=2, fy=2, interpolation=cv2.INTER_CUBIC)
76         #cv2.imwrite("usthesim.png",image)
77
78         # Renk düzeltmesi
79         gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
80         gray = cv2.medianBlur(gray, 3)
81         gray = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 31, 2)
82
83         # OCR işlemini gerçekleştir
84         # config="--tessdata-dir \"C:/Program Files/Tesseract-OCR/tessdata\""
85         text = pytesseract.image_to_string(gray, lang='eng')
86         text = text.lower()
87
88         # düzenli ifade oluşturma
89         regex = r"\b(?:deneme|test)\d*\b"
90
91         # düzenli ifade oluşturma
92         text = text.replace(" ", "")
93         text = re.sub(r'[\w\s]', '', text)
94
95         # düzenli ifadeyi uygulama
96         eslesmeler = re.findall(regex, text)
97
98         # egleşenleri yazdırma
99         for esleme in eslesmeler:
100             print(esleme)
101
102     return eslesmeler
103
104
105     def icindekilerTespit(image):
106
107         # Resmi gri tonlama dönüştür
108         gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
109
110         # Görüntüyü azalt ve medyan bulanıklaştırma uygula
111         denoised_img = cv2.fastNLMMeansDenoising(gray_img)
112         blurred_img = cv2.medianBlur(denoised_img, 3)
113
114
115
116
117
118
119
120
121
122
110 %  No Issues found

```

Bu kısmında bütün resimleri bu şekilde kontrol etmekteyiz fakat sadece **1.Test**'e ait olan kısımlar kesilerek alınmaktadır. XML kısmında bu ayrimı yapmaktayız.

Bunun nedeni test numarasının genelde testin ilk sayfasında yer almasıdır.

Altta yer alan kod bloğunda YOLOV7'ye ait config düzenlemelerini görebilmekteyiz. Açıklama satırlarında gerekli bilgiler verilmiştir.

```

page_number_founder.py  openFile.py  detect2.py*  X
171     classes_to_filter = [] # İsm'e göre filtrelemek için sınıf listesi verebilirsin, sınıf numarası girmenize gerek yok. ['train','person']
172     coordinatesArray = []
173
174     # ****
175     # conf-thres (confidence threshold),
176     # algoritmanın tespit edilme olasılığının ne kadar yüksek olması gerektiğini belirler. Bu değer, 0 ile 1 arasında olmalıdır ve varsayılan değer 0.70'dir.
177     # Bu değer, algoritmanın tespit ettiği nesnelerin gerçekten var olduğundan emin olmak için kullanılır.
178     # Örneğin, eğer conf-thres değeri 0.70 olarak ayarlanılmışsa, algoritma sadece 0.70 veya daha yüksek olasılıkta gerçekten bir nesne olduğunu inandığı nesneleri tespit edecektir.
179
180     # iou-thres (Intersection over Union threshold),
181     # algoritmanın nesneleri arasındaki örtüşme oranını belirler.
182     # Bu değer, 0 ile 1 arasında olmalıdır ve varsayılan değer 0.70'dir.
183     # Bu değer, algoritma tarafından tespit edilen birden fazla nesnenin aynı gerçek nesne olduğunu gösterir.
184     # Örneğin, eğer iou-thres değeri 0.70 olarak ayarlanılmışsa, algoritma sadece 0.70 veya daha yüksek bir örtüşme oranına sahip olan nesneleri gerçek nesne olarak kabul edecektir. """
185
186     opt = {
187         # best_dene aktifliği en son durdurduğun bi onu
188         # test_model/best.pt
189         # test_model/best_320.pt
190         # best_320.pt
191         # teslim edildikten best_406 olarak geçicek
192         # model_testkesim.pt
193         # SON HALİ best278.pt
194         # "weights": "best278.pt", # Path to weights file default weights are for nano model
195         # "yaml": "data/coco.yaml",
196         # "img_size": 640, # default image size
197         # "conf_thres": 0.68, # confidence threshold for inference.
198         # "iou_thres": 0.18, # NMS IoU threshold for inference. #Algılamazsa .30 Üzerinde çalış !!!!!!!1
199         # "device": "cpu", # device to run our model i.e. 0 or 0,1,2,3 or cpu
200         # "classes": classes_to_filter # list of classes to filter or None
201     }
202

```

Devamında da etiketlenmiş sınıfların ayrimını yapıyoruz. Bu kısmında tespit edilmek istenen sınıflar açıklama satırlarıyla ekran çıktısıyla belirtilmiştir.

Tespit edilmesini istediğimiz tipleri **non_max_suppression()** içerisinde yer alan classes içerisinde belirtiyoruz

NOT: Projenin güncel halinde aşağıda belirtilen sınıflar kullanılıp, ilerleyen zamanlarda diğerleri de aktif edilip fonksiyonlara uyarlanıp kullanılabilir hale getirebilir.

NOT2: Diğer sınıfların aktif bir şekilde kullanılması için datasetinin içeriğinin komple gözden geçirilmesi ve eksik etiketlemelerin tamamlanması gerekmektedir.

```

239         if classes:
240             classes = [i for i in range(len(names)) if i not in classes]
241             # print(classes)
242             # 0-1,3-5 arası abcde
243             # 2 answer_Box
244             # 6-10 arası answer_key12345_box
245             # 11 explanation_box
246             # 12 question_box
247             # 13 question_kok
248             # 14 question_number_box
249             # 15 whole_question_box
250             # istenen classes=[0, 1, 3, 4, 5, 11, 12, 13, 14]
251
252     pre = non_max_suppression(pre, opt['conf-thres'], opt['iou-thres'],
253                               classes=[0, 1, 3, 4, 5, 13, 14,15],
254                               agnostic=False)
255
256

```

Config ayarlamalarımız ve inputumuz hazırladığımıza göre sınıflandırma işlemine geçebiliriz.

Bu aşamadan sonra ilk olarak gelen tespit bilgilerinden yola çıkararak soru ayımlarını yapmamız gerekmektedir. Bu ayımları yaparken ilk olarak:

- Sorunun kalibi(whole_question_box)
- answer_a
- answer_b
- answer_c
- question_number_box

Denetimini gerçekleştirerek yapmaktayız. Bunun nedeni bir sorunun unsurlarının belirlenmesidir.

Burada değişkenleri kullanmaktaız gerekli arttirma durumlarını yaparak ilerleyen kısımlarda problemin kontrolünü buradan baz alacağız.

Farklı kaynaklarda tespit edilmek istenilen soruya benzer klasik sorular çıkabilmekte, bu hata kontrolü sayesinde bu problemin önüne geçebilmektedir.

```

page_number_founder.py    openFile.py      detect2.py ⌂
258     counter, whole_counter, counter_whole = 0, 0, 0
259     question_check_1, question_check_2, question_check_3, question_check_4 = 0, 0, 0, 0
260     found_question_number = 0
261     check1_sayac, check2_sayac = 0, 0
262     for i, det in enumerate(pre):
263         s = ''
264         s += '%gx%g ' % img.shape[2:] # print string
265         gn = torch.tensor(img0.shape)[[1, 0, 1, 0]]
266         if len(det):
267             det[:, :4] = scale_coords(img.shape[2:], det[:, :4], img0.shape).round()
268
269             for c in det[:, -1].unique():
270                 n = (det[:, -1] == c).sum() # detections per class
271                 s += f'{n} {names[int(c)]}'*('n > 1'), " # add to string
272
273             for *xyxy2, conf2, cls2 in reversed(det):
274                 image_size_xnl.append(imgsize[0], imgsize[1])
275                 label = f'{names[int(cls2)]} {conf2:.2f}'
276                 plot_one_box(xyxy2, img0, label=label, color=colors[int(cls2)], line_thickness=2)
277                 c1, c2 = (int(xyxy2[0]), int(xyxy2[1])), (int(xyxy2[2]), int(xyxy2[3]))
278
279                 if names[int(cls2)] == "whole_question_box":
280                     question_test1 = question_test1 + 1
281
282
283                 for *xyxy2, conf2, cls2 in reversed(det):
284                     label = f'{names[int(cls2)]} {conf2:.2f}'
285                     plot_one_box(xyxy2, img0, label=label, color=colors[int(cls2)], line_thickness=2)
286                     c1, c2 = (int(xyxy2[0]), int(xyxy2[1])), (int(xyxy2[2]), int(xyxy2[3]))
287
288                     if names[int(cls2)] == "answer_a":
289                         question_check_1 = 1
290                         question_test2 = question_test2 + 1
291                     if names[int(cls2)] == "answer_b":
292                         question_check_2 = 1
293                         question_test3 = question_test3 + 1
294                     check1_sayac = check1_sayac + 1
295                     if names[int(cls2)] == "answer_c":
296                         question_check_3 = 1
297                     if names[int(cls2)] == "question_number_box":
298                         check2_sayac = check2_sayac + 1
299                         question_check_4 = 1
300

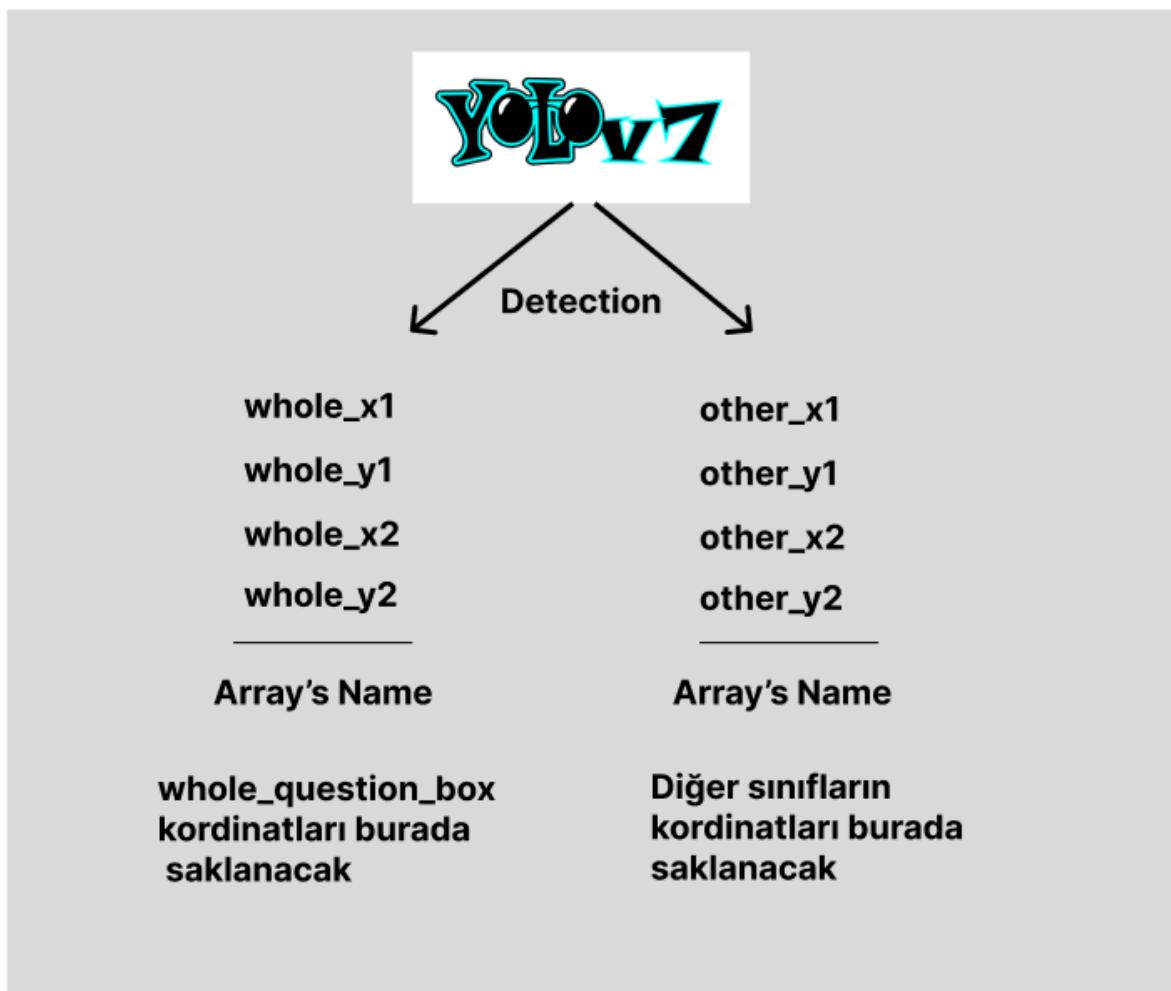
```

Kodların devamında ise aşamalı olarak nasıl tespit edildiğini konu alacağız.

```
page_number_founder.py    openFile.py    detect2.py ✘ ×  
508 question_check_4 = 0  
509  
510  
511 if ( question_check_1 == 1 and question_check_2 == 1 and question_check_4 == 1 and question_test1 != 0) or ( question_check_1 == 1 and question_check_3 == 1 and question_check_4 == 1 and question_test1 != 0):  
512 if (question_test1 == question_test2) or (question_test1 == question_test3):  
513 for *xxyy, conf, cls in reversed(det):  
514     label = f'{names[int(cls)]} {conf:.2f}'  
515     plot_one_box(xxyy, img0, label=label, color=colors[int(cls)], line_thickness=2)  
516     #cv2.imwrite("denemebak.jpg", img0)  
517     c1, c2 = (int(xxyy[0]), int(xxyy[1])), (int(xxyy[2]), int(xxyy[3]))  
518     if names[int(cls)] == "question_kok":  
519         ar17.append((int(xxyy[0]), int(xxyy[1]), int(xxyy[2]), int(xxyy[3])))  
520  
521     if names[int(cls)] == "whole_question_box":  
522         whole_x1.append(int(xxyy[0]))  
523         whole_y1.append(int(xxyy[1]))  
524         whole_x2.append(int(xxyy[2]))  
525         whole_y2.append(int(xxyy[3]))  
526  
527         if whole_x1[0] is not None:  
528             counter_whole = counter_whole + 1  
529             if counter_whole == 1:  
530                 page_number = founder.readPathname(url1)  
531  
532 # Sayfa numarasını ekleerek eşleştiriyoruz  
533 my_dict['page_number'].append(page_number)  
534 ar1.append(page_number)  
535  
536 # Verileri sözlüğe ekle, sonrasında ayırm kolay olacak  
537 my_dict["whole_question_box"].append(whole_x1[whole_counter])  
538 my_dict["whole_question_box"].append(whole_y1[whole_counter])  
539 my_dict["whole_question_box"].append(whole_x2[whole_counter])  
540 my_dict["whole_question_box"].append(whole_y2[whole_counter])  
541 ar2.append((whole_x1[whole_counter], whole_y1[whole_counter],  
542            whole_x2[whole_counter], whole_y2[whole_counter]))  
543  
544 whole_counter = whole_counter + 1  
545 else:  
546     counter = counter + 1  
547     other_x1.append(int(xxyy[0]))  
548     other_y1.append(int(xxyy[1]))  
549     other_x2.append(int(xxyy[2]))  
550     other_y2.append(int(xxyy[3]))  
551     my_labels.append(names[int(cls)])  
552     if names[int(cls)] == "question_number_box":  
553         soru_lokasyon.append((int(xxyy[0]), int(xxyy[1]), int(xxyy[2]), int(xxyy[3])))  
554  
555 value = 0  
556 croptheedge, croptheedge2, croptheedge3 = 0, 0, 0  
557 soru_numarasi_iptal = 0  
558  
559  
560  
561
```

Yukarıdaki kod çıktısını **Aşama 1** olarak ele alalım burayı aşağıdaki grafikten daha iyi anlayabiliriz.

Buraya eklemeye yapabileceğimiz bir nokta da **my_labels** dizisidir. İlerideki kontrol algoritmalarımızda other dizilerinin içerisindeki kordinatların hangi sınıflara ait olduğunu bu sayede daha iyi bir şekilde anlayabilmekteyiz.

- AŞAMA 1-

Tespit aşamasında bir sayfa üzerinde kordinatlar karışık gelmektedir.

Burada amacımız sınıflandırma kordinatlarını bir çatı altında tutmak istediğimiz için bir diziye atıyoruz. Bu sayede sınıf karmaşasının önüne geçmiş oluyoruz.

Bunu 'whole_question_box' sınıfını ayırtarak yapmaktayız.

Bunun nedeni her bir 'whole_question_box' kordinatı içerisinde düşen diğer sınıfların tespit edilip soru bileşenlerinin birleştirilmesinden kaynaklıdır.

4. Bu sureden;

- I. Allah (c.c.) evrenin idaresini yürütmektedir.
- II. Kulluk görevinin dışında dayanışmaya da önem verilmelidir.
- III. İnsanlara yaşadıkları olaylar karşısında sabır tavsiye edilmelidir.
- IV. İbadetleri özüne uygun ihlasl yapmamız gereklidir.
mesajlarından hangileri çıkarılabilir?

A) I ve II

B) I ve III

C) II ve IV

D) III ve IV

Kodun devamında ise programın çoğu yerinde rastlayabileceğimiz bir kontrol algoritması yer almaktadır, bu gibi metodların amacı hata riskini en aza indirmek ve kullanıcıya kesintisiz bir kullanım sunmaktadır.

```

361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
      # Eğer soru numarası blok içerisinde değilse bunu iptal ediyoruz
      dogrulama = False
      # Eğer sayfada bir bozukluk varsa o sayfayı atlaması gerektiğini söylüyoruz.
      if soru_lokasyon:
          for cntor in range(len(whole_x1)):
              for cntor2 in range(len(soru_lokasyon)):
                  if whole_x1[cntor] - 20 < soru_lokasyon[cntor2][0] and whole_y1[cntor] - 20 < \
                     soru_lokasyon[cntor2][1] and whole_x2[
                     cntor] + 15 > soru_lokasyon[cntor2][2] and whole_y2[cntor] + 5 > soru_lokasyon[cntor2][3]:
                      dogrulama = True
              else:
                  continue

```

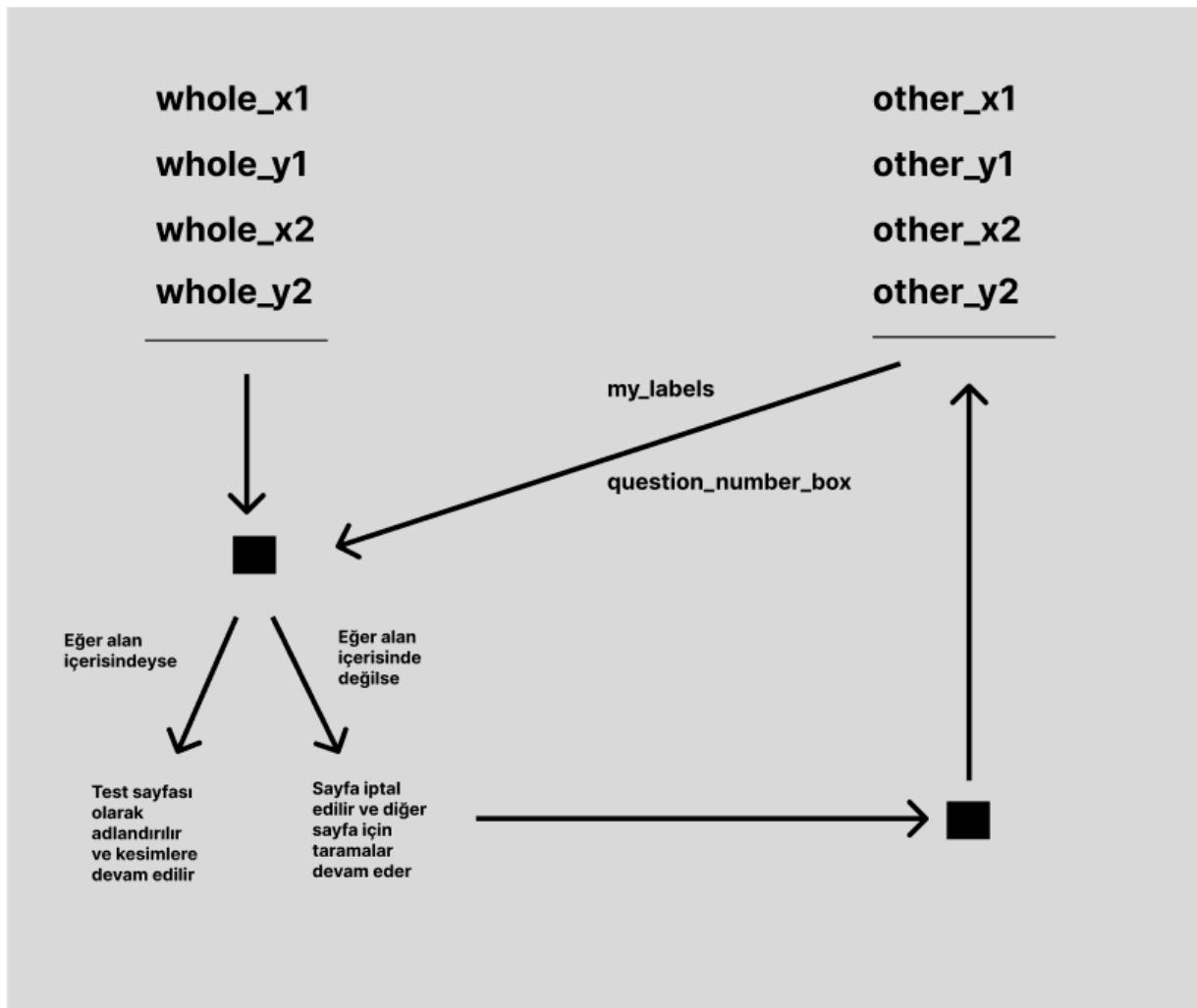
Burada da **Aşama 2** olarak adlandırılabilir bölüm bulunmaktadır. Bu bölüm tarif eden şablon aşağıda yer almaktadır.

```

396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
      label = f'{names[int(cls)]} {conf:.2f}'
      c1, c2 = (int(xxy[0]), int(xxy[1])), (int(xxy[2]), int(xxy[3]))
      if names[int(cls)] == "whole_question_box" and dogrulama == True:
          for l in range(whole_counter):
              explanation_condition, explanation_condition2, explanation_condition3, explanation_condition4, check1, check2, check3, check4, check5, check6, check7, check8, check9 = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
              dogrulam1, dogrulam2, dogrulam3, dogrulam4, dogrulam5 = 0, 0, 0, 0, 0
              for b in range(len(my_labels)):
                  # Açıklaşım satırını ve soru box'ını içerecek için küçült
                  if my_labels[b] == "question_box" or my_labels[b] == "explanation_box" or \
                     my_labels[b] == "question_number_box":
                      my_labels[b] = "question_box"
                  croptheadge = -10
                  croptheadge2 = 5
                  if my_labels[b] == "answer_box":
                      croptheadge3 = -10
                  if other_y1[b] != 0:
                      if my_labels[b] == "question_number_box":
                          degisken = True if whole_x1[l] - 60 < other_x1[b] + croptheadge2 and \
                           whole_y1[l] - 60 < \
                           other_y1[b] + croptheadge2 and whole_x2[l] + 15 > other_x2[b] + croptheadge and \
                           whole_y2[l] + 5 > other_y2[b] + croptheadge3 else False
                          degisken2 = True if whole_x1[l] - 60 < other_x2[
                           b] + croptheadge2 and \
                           whole_y1[l] - 60 < \
                           other_y2[b] + croptheadge2 and whole_x2[l] + 15 > other_x2[b] + croptheadge and \
                           whole_y2[l] + 5 > other_y2[b] + croptheadge3 else False
                      if degisken == True or degisken2 == True:
                          if whole_x1[l] < other_x1[b] or other_x2[b] < whole_x1[l] or \
                             whole_y1[l] > other_y1[b] or other_y2[b] > whole_y1[l]:
                              whole_x1[l] = other_x1[b] - 10
                              whole_y1[l] = other_y1[b] - 10
                          #if degisken == False or degisken2 == False:
                          #    soru_numarası_iptal = 1
                          # if whole_y1[l] > other_y1[b] or other_y2[b]>whole_y1[l]:x
                          # Doğruluk oranlarında sıkıntı
                          # çıktıgı zaman ilk bakacağı yerlerden biri burası olsun

```

- AŞAMA 2-



Yukarıdaki talimatlarda belirtildiği gibi bir soruya soru yapan unsurlar bulunmaktadır. Bunlardan biriside question_number_box'tır.

Burada amaç açık uçlu sorularda ve türevlerinde yer alan soru numarasının tespit edilip kesimi bozmasını engellemek ve programın sorunsuz bir .xml çıktısı vermesini sağlamaktır.

Aşama 3'e geçtiğimiz zaman bu kısımda filtreleme algoritmalarını uyguladıktan sonraki ayıklama algoritmasıyla karşılaşmaktayız.

Bütün olası durumları elediğimiz zaman elimizde kalan saf verileri **whole_question_box** içerisinde tekrardan aratıyoruz. Ve düzenli bir şekilde dizilere yerleştiriyoruz.

Burada dizi verileri şu şekilde gözükmektedir.

- array = [(x1,y1,x2,y2),(x1,y1,x2,y2)..n(veri)..(x1,y1,x2,y2)]

Burada dikkat etmemiz gereken nokta veri akışına herhangi bir aykırı müdahalenin kesim yapılan kitaba negatif etki bırakacağıdır.

NOT:

Genel kod yapısında bir değişiklik yapıldıktan sonra atılan commitlerde detaylıca yapılan değişikliliğin ayrıntıları paylaşılmalıdır. Aksi takdirde kod hata çözümü süreci uzayacaktır.

```

437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480

    if whole_x1[l] - 20 < other_x1[b] + croptheedge2 and whole_y1[l] - 20 < \
        other_y1[b] + croptheedge2 and whole_x2[\
        l] + 15 > other_x2[b] + croptheedge and whole_y2[l] + 5 > other_y2[\
        b] + croptheedge3:
        # Whole question'in ayrimi yukarıda yapılmıştı o sıraya göre şimdide diğer verileri ekle
        if my_labels[b] == "answer_a":
            if dogrula1 == 0:
                ar3.append((other_x1[b], other_y1[b], other_x2[b], other_y2[b]))
            check2 = True
            dogrula1 = 1

        elif my_labels[b] == "answer_b":
            ar4.append((other_x1[b], other_y1[b], other_x2[b], other_y2[b]))
            check3 = True
            dogrula2 = 1

        elif my_labels[b] == "answer_c":
            if dogrula3 == 0:
                ar5.append((other_x1[b], other_y1[b], other_x2[b], other_y2[b]))
            check4 = True
            dogrula3 = 1

        elif my_labels[b] == "answer_d":
            if dogrula4 == 0:
                ar6.append((other_x1[b], other_y1[b], other_x2[b], other_y2[b]))
            check5 = True
            dogrula4 = 1

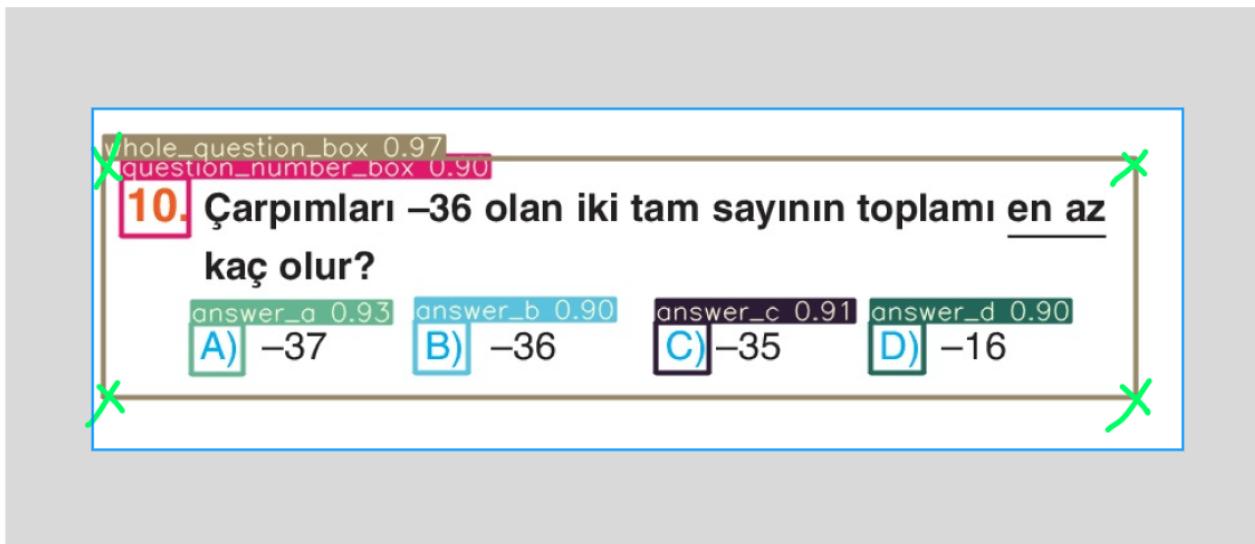
        elif my_labels[b] == "answer_e":
            if dogrula5 == 0:
                ar7.append((other_x1[b], other_y1[b], other_x2[b], other_y2[b]))
            check1 = True
            dogrula5 = 1

        elif my_labels[b] == "answer_box":
            ar8.append((other_x1[b], other_y1[b], other_x2[b], other_y2[b]))
            check6 = True

        elif my_labels[b] == "question_number_box": ... devamında da aynı işlemler diğer sınıflara uygulanmaktadır

```

- AŞAMA 3-



Yukarıdaki **Aşama 3** tablosunda gözüktüğü üzere yeşil işaretli alan içerişi taranmaktadır. Burada other dizisine ait verilere tolerans hakkı alınarak birkaç pixel küçültülmekte ve whole_question_box birkaç pixel büyütülmektedir.

Burada yapılan işlemin amacı etiketleme sırasında yapılan hataların egale edilmesini sağlamaktır.

```

        croptheedge, croptheedge2, croptheedge3 = 0, 0, 0
s1 = '0'
if not explanation_condition:
    ar16.append((s1, s1, s1, s1))
if not explanation_condition2:
    ar13.append((s1, s1, s1, s1))

if not explanation_condition3:
    ar14.append((s1, s1, s1, s1))
if not explanation_condition4:
    ar15.append((s1, s1, s1, s1))
if not check1:
    ar7.append((s1, s1, s1, s1))
if not check2:
    ar3.append((s1, s1, s1, s1))
if not check3:
    ar4.append((s1, s1, s1, s1))
if not check4:
    ar5.append((s1, s1, s1, s1))
if not check5:
    ar6.append((s1, s1, s1, s1))
if not check6:
    ar8.append((s1, s1, s1, s1))
if not check7:
    ar9.append((s1, s1, s1, s1))
if not check8:
    ar10.append((s1, s1, s1, s1))
if not check9:
    ar11.append((s1, s1, s1, s1))
if not check10:
    ar12.append((s1, s1, s1, s1))
#if not check17:
#    ar17.append((s1, s1, s1, s1))

if l == whole_counter - 1:
    b = 0
    value = 1

```

```

552
553
554     for i in range(len(ar9)):
555         try:
556             if len(ar9) == 1:
557                 if ar9[0][0] == '0':
558                     question_check_4 = 0
559             if len(ar9) == 2:
560                 if ar9[0][0] == '0' and ar9[0][1] == '0':
561                     question_check_4 = 0
562             except:
563                 print("soru iptali")
564

```

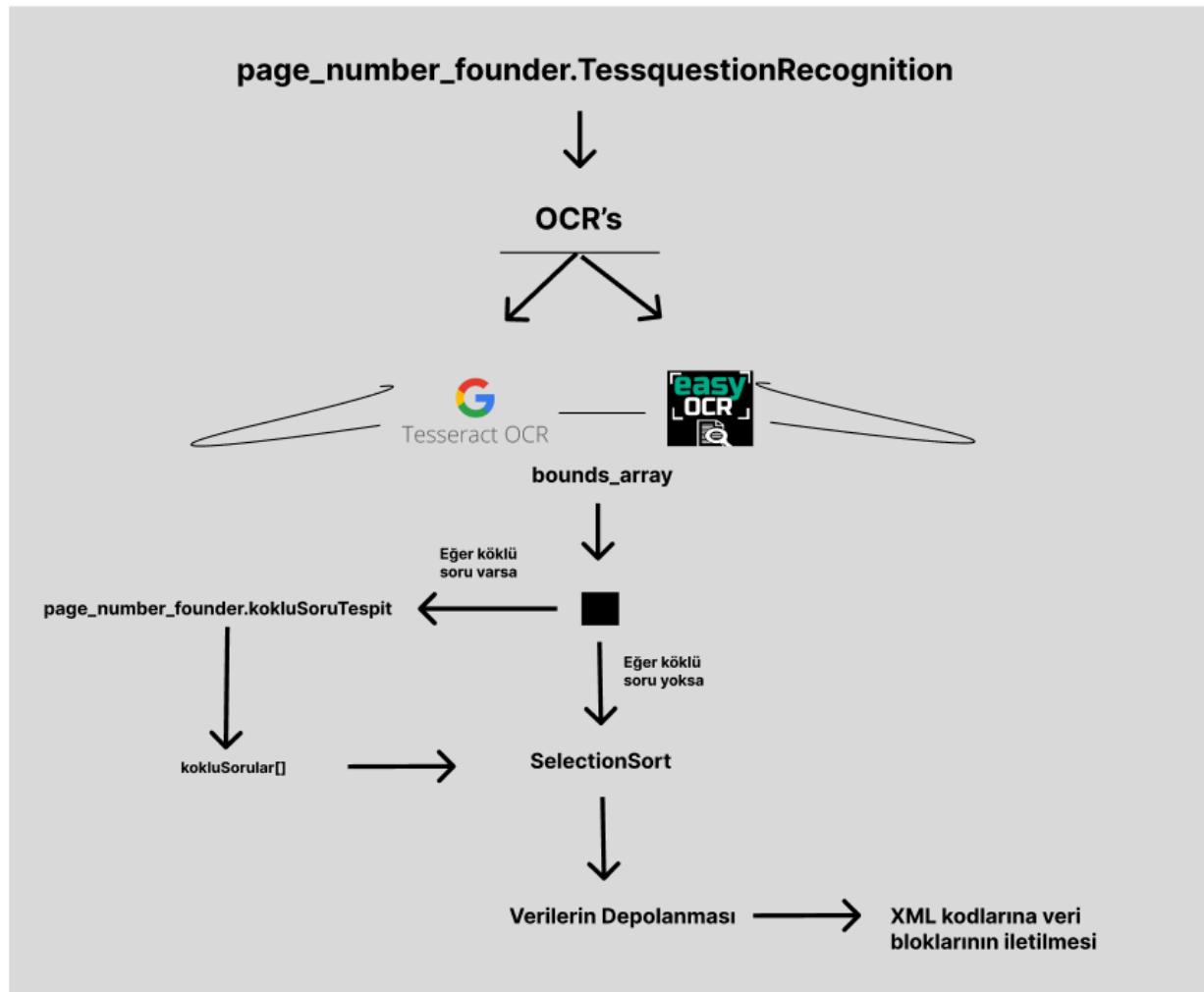
Xml'e aktarılan sınıflandırmalarda herhangi bir veri eksikliği tespit edilirse (Diziye ekleme yapılmaması) gelecek verinin yerini doldurması için '0' eklenmektedir.

Bu sayede veri kaymalarının önüne geçilmektedir.

Son aşama olan **Aşama 4** geçtiğimizde ise OCR ve sıralama algoritmalarını göreceğiz.

Detaylara geçmeden önce aşağıdaki grafikle son bölümün özetiğini görüntüleyelim.

- AŞAMA 4-



İlk olarak veri lokasyonlarını tespit edilmesi üzerine OCR algoritmasına yolluyoruz.

Gelen lokasyonlar sayfa üzerinde kesilerek küçük bir hale getiriliyor, sonrasında resim üzerinde farklı manipülasyon işlemleri uygulanıyor. İlk olarak bu veri TesseractOCR'a girmekte eğer başarısız olursa EasyOCR'a teslim edilmektedir.

Eğer tespit yine başarısız ise farklı bir manipülasyon işlemi uygulanır ve bu adımlar tekrarlanır.

Hala tespit edilememiş ise '0' olarak veri kümesine eklenerek dizi teslim edilir.

Sonrasında sayfa üzerinde köklü sorunun varlığı kontrol edilerek, var ise algoritmaya yönlendirme yapılır ve iki durum sonunda da SelectionSort algoritmasına bütün veri kümeleri iletilir.

Burada karmaşık gelen lokasyonları düzelterek soru numarasına göre sıralamaktayız. Sonrasında verilerimizi ortak bir dict kümesinde toplayarak geçici dict.'leri sıfırlıyoruz. Bu işlem kitap sayfaları bitene kadar devam etmektedir. Bahsedilen konu üzerine kodlara geçiyoruz.

Eğer belirtilen şartları sağlayarak fonksiyona girmezse hatalı veri olduğunu göstermek için "-1" olarak girilir.

```

if (question_check_1 == 1 and question_check_2 == 1 and question_check_4 == 1 and dogrulama == True) or (
    question_check_1 == 1 and question_check_3 == 1 and question_check_4 == 1 and dogrulama == True):

    #cv2.imwrite("nasiya.jpg", img0)

    found_question_number = number_founder.TessquestionRecognition(img_sakla,
                                                                    ar9)

else:
    for z in range(len(ar9)):
        question_number_array2.append('-1')
# OCR ile (varsayı) köklü soruların tespit edilmesi

```

Yukarıda bahsedilen manipülasyon işlemlerinin uygulandığı kısım aşağıdaki fonksiyon altında yer almaktadır. Bu kısma müdahale için .xml kısmındaki hatalardan yola çıkarak düzeltmeler yapılmalıdır.

```

page_number_founder.py  ✘ detect2.py
241     def TessquestionRecognition(image, dict_pagenumber):
242         condition = True
243         counter = 0
244         bounds_array = []
245         integer_list, int_range, missing_ints, empty_string_index, next_missing = [], {}, [], [], 0
246         img = image
247         print(img.shape[:2])
248
249         # Normalde burası kullandığım kısım fakat yeni bir yöntem deniyorum
250         # Şuanda kullanım olan test versiyonu
251         """
252             img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
253             thresh = 100 # 155 >= x
254             im_bw = cv2.threshold(img, thresh, 255, cv2.ADAPTIVE_THRESH_MEAN_C)[1]
255             thresh = im_bw
256             """
257
258         elamansayisi = len(dict_pagenumber)
259         while condition:
260             i = 0
261             for i in range(len(dict_pagenumber)):
262                 if dict_pagenumber[i][0] == '0':
263                     bounds_array.append(')')
264                     if i == len(dict_pagenumber) - 1:
265                         condition = False
266
267                 else:
268                     #x1 0 , y1 1 , x2 2 , y2 3
269                     final = img[dict_pagenumber[i][1] - 10:14 + dict_pagenumber[i][1] + (
270                         dict_pagenumber[i][3]-5 - dict_pagenumber[i][1]),
271                         dict_pagenumber[i][0] - 8:8 + dict_pagenumber[i][0] + (
272                             dict_pagenumber[i][2] - dict_pagenumber[i][0] - 3)]
273                     cv2.imwrite("yontem1.png",final)
274
275                     test = final
276
277                     gray = cv2.cvtColor(test, cv2.COLOR_BGR2GRAY)
278
279                     # ThresholdToRegion_Dynamic kullanarak karakterlerin bulunduğu bölgeleri tespit et
280                     regions = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]
281                     regions = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, \
282                                         cv2.THRESH_BINARY, 9, 2)
283
284                     # EasyOcr'da kullanılması için ham veriyi saklıyorum bu kısımda
285

```

Farklı manipülasyon tekniklerine ulaşmak ve daha fazlasını ekleyerek varyasyonları manipulasyon.py dosyasında yönetebiliriz. Buradan farklı teknikleri ön adım olarak deneyimlerebilir ve testini gerçekleştirebiliriz.

NOT: Farklı bir teknik deneyerek elde edilen başarılı olmuş manipülasyonları hemen kullanıma almayıp diğer kitaplarda da teste sokmalsınız, bunun nedeni diğer kitaplar için risk oluşturma ihtimali olmasından kaynaklıdır.

```

1 import cv2
2 import numpy as np
3 import pytesseract
4
5 # Görüntüyü oku ve gri tonlamaya dönüştür
6 image = cv2.imread('C:/pagesTest/denenecek.png')
7 test = image
8
9 """
10 gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
11
12 # Görüntüyü azaltmak için Gaussian bulanıklık uygula
13 blurred_image = cv2.GaussianBlur(gray_image, (5, 5), 0)
14
15 # Kenarları belirginleştirmek için Canny kenar algılama uygula
16 canny_image = cv2.Canny(blurred_image, 30, 150)
17
18 # Görüntüyü dilate etme ve erozyon uygulayarak sayıları daha belirgin hale getir
19 kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
20 dilated_image = cv2.dilate(canny_image, kernel, iterations=1)
21 eroded_image = cv2.erode(dilated_image, kernel, iterations=1)
22
23 cv2.imwrite("yontem1.png", eroded_image)
24 """
25
26 #####
27 final = cv2.cvtColor(test, cv2.COLOR_BGR2GRAY)
28 thresh = 180 # 155 >= x
29 im_bw = cv2.threshold(final, thresh, 255, cv2.ADAPTIVE_THRESH_MEAN_C)[1]
30
31
32 # Hafifçe bulanıklaştır ve Otsu'nun eşliğini uygula
33 blur = cv2.GaussianBlur(im_bw, (3, 3), 0)
34 thresh = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]
35
36 cv2.imwrite("yontem2.png", thresh)
37
38 #####
39 """
40 gray_image = cv2.cvtColor(test, cv2.COLOR_BGR2GRAY)
41 filtered = cv2.bilateralFilter(gray_image, 9, 75, 75)
42 thresh = cv2.adaptiveThreshold(filtered, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, 15, 3)
43
44 cv2.imwrite("yontem3.png", thresh)
45 """

```

Soru numarasının saylaştırılmasından sonra ocr'ı eğer köklü soru varsa tekrardan aktifleştiriyoruz.Aşağıda belirtilen fonksiyonun içerisinde giderek detayları inceleyelim.

```

# OCR ile (varsı) köklü soruların tespit edilmesi
try:
    if ar17[0][0]:
        # Buradan donecek olan ((numaralar),sayfa_no)
        kokluSorular = number_founder.kokluSoruTespit(img_sakla, ar17,page_number)
except:
    print("köklüsorubulunamadi")

```

Bu kısımda ilk olarak köklü sorunun çerçevesi kesilerek üstündeki metin tesseract vasıtasıyla metine çevrilir. Çevrilen bu metinin içerisinde **kelimeler** dizisi içerisinde kelimelelerden biri aranır. Bunun nedeni kök tiplerinin sorularla bağlantı olması ve .xml kısmına işaretleme yapılmasından kaynaklıdır. Bu kelimelelerden biri tespit edildikten sonra öncesinde yer alan 5 kelimelek alan kesilir ve içerisinde sayılar aranmaya başlanır.

Dikkat edilmesi gereken noktalardan bazıları "-" işaretini ile "ve" kısımlarıdır.

"-" gelen kısımlarda aralığı alırken, "ve" ile belirtilen yerlerde belirli soruları kapsamış olmaktadır.

Geri kalan kısımlarda sadece "1,2,3" gibi unsurlara yer verilmekte ve "," burada ayrıstırılarak ham sayılarla ulaşabilmekteyiz.

Aşağıda yer alan kodda yukarıda dikkat etmemiz gereken kısımlara ait fonksiyonlar yer almaktadır.

```
manipulasyon.py * page_number_founder.py detect2.py kokluSoruTespit
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
# Resimdeki metni oku
text = pytesseract.image_to_string(final)

kelimeler = ["soru", "according", "Soru", "Sorular", "soru"]
tut = []
# Noktalama işaretlerini kaldırma
text = ''.join(c for c in text if c not in ('.,;?!'))
for kelime in kelimeler:
    if kelime.lower() in text.lower():
        kelimeler_once = text.lower().split(kelime.lower())[0].split()[-5:]
        sayilar = []
        aralik = []
        for k in kelimeler_once[::-1]:
            if k.isdigit() or k.replace("ve", "").isdigit():
                if k.isdigit():
                    sayilar.append(int(k))
                elif k.replace("ve", "").isdigit():
                    sayilar.append(int(k.replace("ve", "")))
            if len(sayilar) == 2:
                break
            elif '-' in k:
                aralik = k.split('-')
                if aralik[0].isdigit() and aralik[1].isdigit():
                    sayilar += list(range(int(aralik[0]), int(aralik[1]) + 1))

        if len(sayilar) == 1:
            tut.append(sayilar[0])
        elif len(sayilar) > 1:
            tut += sayilar

tut = list(set(tut)) # tekrar edenleri sil

tut = list(map(int, tut))
tut.sort()
is_ardisik = True
for i in range(1, len(tut)):
    if tut[i] - tut[i - 1] != 1:
        is_ardisik = False
        break
if not is_ardisik:
    ara_degerler = []
    for i in range(1, len(tut)):
        if tut[i] - tut[i - 1] != 1:
            ara_degerler += list(range(tut[i - 1] + 1, tut[i]))
    tut += ara_degerler
    tut = sorted(tut)

#numbers.sort(reverse=False)
calculator1 calculator2 calculator3 calculator4 = 0 1 2 3
```

Sonrasında ise SelectionSort algoritmasının uygulanmasına geçilmektedir ve bu şekilde veri düzeneği sağlanmıştır.

```

manipulasyon.py* page_number_founder.py detect2.py ✘
 586     found_question_number = list(map(int, found_question_number))
 587     # Soruları sıralama
 588     size = len(found_question_number)
 589     # cv2.imwrite("bak.jpg", img0)
 590     for s in range(size):
 591         min_idx = s
 592         for i in range(s + 1, size):
 593             if found_question_number[i] < found_question_number[min_idx]:
 594                 min_idx = i
 595
 596     # Doğru Sıraya Alıyorum Hepsini Dağıtık Hali Bu Kısmında Düzeltiliyor
 597     (found_question_number[s], found_question_number[min_idx]) = (
 598         found_question_number[min_idx], found_question_number[s])
 599
 600
 601     try:
 602         (ar3[s], ar3[min_idx]) = (ar3[min_idx], ar3[s])
 603         (ar4[s], ar4[min_idx]) = (ar4[min_idx], ar4[s])
 604         (ar5[s], ar5[min_idx]) = (ar5[min_idx], ar5[s])
 605         (ar6[s], ar6[min_idx]) = (ar6[min_idx], ar6[s])
 606         (ar7[s], ar7[min_idx]) = (ar7[min_idx], ar7[s])
 607         (ar8[s], ar8[min_idx]) = (ar8[min_idx], ar8[s])
 608         (ar9[s], ar9[min_idx]) = (ar9[min_idx], ar9[s])
 609         (ar10[s], ar10[min_idx]) = (ar10[min_idx], ar10[s])
 610         (ar11[s], ar11[min_idx]) = (ar11[min_idx], ar11[s])
 611         (ar12[s], ar12[min_idx]) = (ar12[min_idx], ar12[s])
 612         (ar13[s], ar13[min_idx]) = (ar13[min_idx], ar13[s])
 613         (ar14[s], ar14[min_idx]) = (ar14[min_idx], ar14[s])
 614         (ar15[s], ar15[min_idx]) = (ar15[min_idx], ar15[s])
 615         (ar16[s], ar16[min_idx]) = (ar16[min_idx], ar16[s])
 616         (ar2[s], ar2[min_idx]) = (ar2[min_idx], ar2[s])
 617
 618     except:
 619         print("soru hata tespiti, geçildi")
 620
 621     for u in range(len(found_question_number)):
 622         question_number_array2.append(found_question_number[u])
 623
 624     # Yeni Gelecek Numaralar İçin Dizimizi Sıfırlıyoruz

```

Son olarak kitabın sayfalarının verilerini tutması için oluşturulan **my_dict_copy** dict'ine verilerimizi atarak geçici sayfa dict'ımızı sıfırlıyoruz. Eğer kitabın sonuna gelindiye uygulamanın .xml çıktısı vermesi için `create_XML.saltGenerateXML()` fonksiyona yönelmektediyiz. Burada yukarıda yer alan verilerimizi parametre olarak vermektediyiz.

```

623
624     # Yeni Gelecek Numaralar İçin Dizimizi Sıfırlıyoruz
625     found_question_number = []
626
627     if dogrulama == True:
628         # Veri Arraylerimizi Tek Çatı Altında Array'de yineden topluyoruz
629         my_dict_copy['page_number'].append(ar1)
630         my_dict_copy['whole_question_box'].append(ar2)
631         my_dict_copy['answer_a'].append(ar3)
632         my_dict_copy['answer_b'].append(ar4)
633         my_dict_copy['answer_c'].append(ar5)
634         my_dict_copy['answer_d'].append(ar6)
635         my_dict_copy['answer_e'].append(ar7)
636         my_dict_copy['answer_box'].append(ar8)
637         my_dict_copy['question_number_box'].append(ar9)
638         my_dict_copy['answer_key1_box'].append(ar10)
639         my_dict_copy['answer_key2_box'].append(ar11)
640         my_dict_copy['answer_key3_box'].append(ar12)
641         my_dict_copy['answer_key4_box'].append(ar13)
642         my_dict_copy['answer_key5_box'].append(ar14)
643         my_dict_copy['question_box'].append(ar15)
644         my_dict_copy['explanation_box'].append(ar16)
645         my_dict_copy['question_kok'].append(koklusorular)
646
647     # Genel Verilerimizi Sıfırlıyoruz Ve Farklı Sayfaya Geçiş Yapıyoruz
648
649     whole_x1, other_x1 = [], []
650     whole_x2, other_x2 = [], []
651     whole_y1, other_y1 = [], []
652     whole_y2, other_y2 = [], []
653     my_labels = []
654     ar1, ar2, ar3, ar4, ar5, ar6, ar7, ar8, ar9, ar10, ar11, ar12, ar13, ar14, ar15, ar16, ar17 = [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], []
655     soru_lokasyon = []
656     koklusorular = []
657
658     #birlestir = "program_ciktilari//cikti_" + str(k) + ".jpg"
659     # Execute ederken burası
660     #cv2.imwrite(birlestir, img0)
661
662     # print(c1,c2)
663
664     if len(file_resource) == k + 1:
665         condition = False
666
667     create_XML.saltGenerateXML(question_number_array2, my_dict_copy, image_size_xml, param3, page_text_top)

```

XML YAPISI Bu fonksiyonu açıklamadan önce istenilen formatı paylaşıyor olacağım.



The screenshot shows a code editor with an XML file named 'deneme_matali.xml'. The code is color-coded for syntax highlighting. A vertical orange line and several red rectangular boxes highlight specific parts of the XML structure. The XML content describes two test items, each with a list of options (A, B, C) and their coordinates.

```
<testler>
    <test ID="0"
        ADI="[]"
        BASARIORANI="50"
        SECENEKSAYISI="3"
        BaslangicSayfasi="9"
        BitisSayfasi="16"
        testCerceve="37,0,1833,119">
        <item>
            <cevap>-</cevap>
            <testID>0</testID>
            <imgsize>2381,1871</imgsize>
            <soruNo>7</soruNo>
            <page>9</page>
            <rect>204,140,1800,2381</rect>
            <secenekler secenek="A">
                <secenek secenek="A">
                    X="220"
                    Y="687"/>
                <secenek secenek="B">
                    X="802"
                    Y="688"/>
                <secenek secenek="C">
                    X="0"
                    Y="0"/>
            </secenekler>
        </item>
        <item>
            <cevap>-</cevap>
            <testID>0</testID>
            <imgsize>2381,1871</imgsize>
            <soruNo>8</soruNo>
            <page>10</page>
            <rect>202,0,1746,1588</rect>
            <secenekler secenek="A">
                <secenek secenek="A">
                    X="179"
                    Y="913"/>
                <secenek secenek="B">
                    X="177"
                    Y="1248"/>
                <secenek secenek="C">
                    X="0"
                    Y="0"/>
            </secenekler>
        </item>
    </test>
</testler>
```

Yukarıda görüldüğü üzere genel yapı "testler" altında toplanmakta ve her test "test" altında listelenmektedir, bu yapının altındaki her soru ise "item" olarak eklenmektedir. Item dışındaki testin hem yanında yer alan verileri parametre olarak gelen verileri kullanarak elde etmekteyiz. İlgili detayları aşağıda bulabilirsiniz.

Gelen verilerimizi tek düzeye ederek işlenebilir bir hale getiriyoruz.

```

265
266     def SaltGenerateXML(question_no,mydict,imgsize,outputxml,pageustyazi):
267         #Salt Hale Getiriyorum Verilerimi
268         #print("soru numaraları=>",question_no)
269
270         dict = {'whole_question_box': [], 'question_number_box': [], 'answer_box': [], 'answer_a': [], 'answer_b': [],
271                 'answer_c': [], 'answer_d': [], 'answer_e': [], 'page_number': [], 'answer_key1_box': [], 'answer_key2_box': [],
272                 'answer_key3_box': [], 'answer_key4_box': [], 'answer_key5_box': [], 'question_box': [], 'explanation_box': [], 'question_kok': []}
273
274         dict['whole_question_box'].append(makeArray(mydict['whole_question_box']))
275         dict['question_number_box'].append(makeArray(mydict['question_number_box']))
276         dict['answer_box'].append(makeArray(mydict['answer_box']))
277         dict['answer_a'].append(makeArray(mydict['answer_a']))
278         dict['answer_b'].append(makeArray(mydict['answer_b']))
279         dict['answer_c'].append(makeArray(mydict['answer_c']))
280         dict['answer_d'].append(makeArray(mydict['answer_d']))
281         dict['answer_e'].append(makeArray(mydict['answer_e']))
282
283         #Sadece PageNumber'da fonksiyona sokmuyorum burası biraz istisna kalıyor, farklı bir yapıya sahip
284         ust_yazi_donestec = 0
285         for i in range(len(mydict['page_number'])):
286             for k in range(len(mydict['page_number'][i])):
287                 dict['page_number'].append(mydict['page_number'][i][k])
288         liste = []
289
290         dict['answer_key1_box'].append(makeArray(mydict['answer_key1_box']))
291         dict['answer_key2_box'].append(makeArray(mydict['answer_key2_box']))
292         dict['answer_key3_box'].append(makeArray(mydict['answer_key3_box']))
293         dict['answer_key4_box'].append(makeArray(mydict['answer_key4_box']))
294         dict['answer_key5_box'].append(makeArray(mydict['answer_key5_box']))
295         dict['question_box'].append(makeArray(mydict['question_box']))
296         dict['explanation_box'].append(makeArray(mydict['explanation_box']))
297         dict['question_kok'].append(makeArray(mydict['question_kok']))
298
299         fileName = f"{outputxml}"
300         root = gfg.Element("testler")
301         calculator,calculator2,calculator3,calculator4 = 0,1,2,3
302         test_counter, test_counter2 = 0,0
303         condition= True
304         secenek_kontrol = 4
305         kok_counter = 0
306         ic_sayac = 0
307         k = 0
308         kok_soru_arttir = 0

```

Eğer köklü sorumuz var ise bunu bir diziye ekleyerek ileride kullanıma alacağız.

```

309         for part in dict['question_kok']:
310             # tüm sayfa sayıları için indeksleri 2'şer aralıklarla gezip yazdırma
311
312             for b in range(1, len(part), 3):
313                 page_number = part[b]
314                 liste.append(page_number)

```

Burada tekdüze edilmiş verilerimizi kullanarak değiştirmelerimizi gerçekleştirmekteyiz.

Verilerimiz şu şekilde gözükmektedir:

- $(x_1, y_1, x_2, y_2, x_1, y_1, x_2, y_2 \dots n \dots, x_1, y_1, x_2, y_2)$ şeklindedir.

Her 4 veride bir tekrar ettiği için veri kümemizi 4'e bölgerek soru adeti kadar algoritmamızı işleme sokuyoruz.

XML'in üst kısmında yer alan verilerin doldurulması için ve gerekli işlemleri try bloğu içerisinde gerçekleştiriyoruz ve aksi durumda yapması gerekeni except bloğunda gerçekleştiriyoruz.

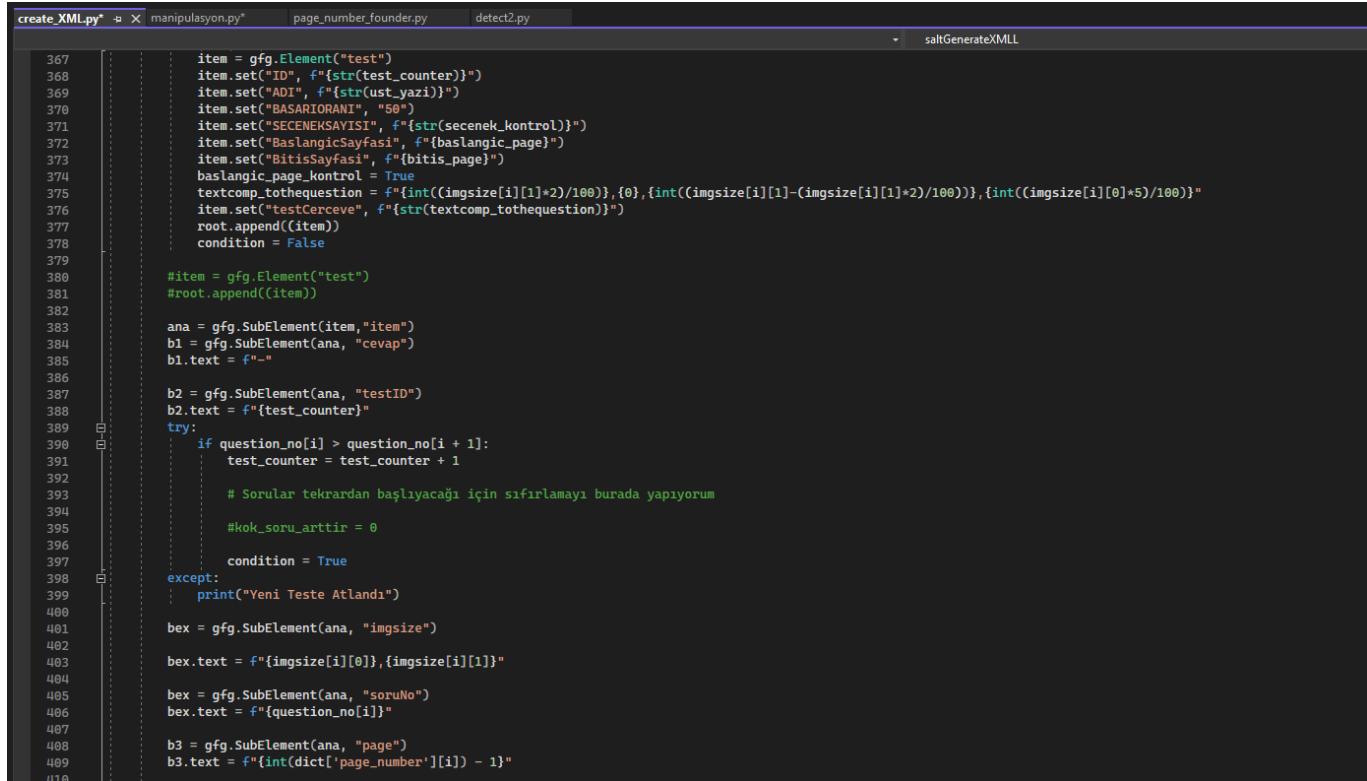
```

316     for i in range(int(len(dict['whole_question_box'][0])/4)):
317         baslangic_page_kontrol = True
318         baslangic_page, bitis_page = "", ""
319         condition2 = True
320         if baslangic_page_kontrol == True:
321             baslangic_page = int(dict['page_number'][i]) - 1
322             ust_yazi = pageustyazi[0]
323
324         if dict['answer_e'][0][calculator] != '0' and dict['answer_e'][0][calculator2] != '0':
325             try:
326                 if dict['answer_e'][0][calculator+4] != '0' and dict['answer_e'][0][calculator2+4] != '0':
327                     secenek_kontrol = 5
328                 except:
329                     if dict['answer_e'][0][calculator-4] != '0' and dict['answer_e'][0][calculator2-4] != '0':
330                         secenek_kontrol = 5
331                     else:
332                         secenek_kontrol = 3
333                         # her bir parça için döngü
334
335
336         if condition == True:
337             c = i
338             while condition2:
339
340                 try:
341                     if question_no[c] > question_no[c + 1]:
342                         bitis_page = int(dict['page_number'][c]) - 1
343                         ust_yazi = pageustyazi[ust_yazi_donence]
344                         baslangic_page_kontrol = False
345                         ust_yazi_donence = ust_yazi_donence + 1
346                         test_counter2 = test_counter2 + 1
347                         condition2 = False
348                     else:
349                         tut = dict['page_number'][c]
350                         c = c + 1
351                         if tut != dict['page_number'][c]:
352                             ust_yazi_donence = ust_yazi_donence + 1
353                             # Bitiş sayfasında sıkıntı çıkarsa burayı kaldırımlırsın.
354                             # Zarfiyat: Aynı sayfa da başlayıp biten testler sıkıntıya girebiliyor.
355                             bitis_page = int(dict['page_number'][c]) - 1
356
357             except:
358                 try:
359                     bitis_page = int(dict['page_number'][c]) - 1
360                     test_counter2 = test_counter2 + 1
361                     condition2 = False
362                 except:
363                     print("atlandı")
364                     test_counter2 = test_counter2 + 1
365

```

Test sayısının artışının takibini:

n dizi sırasındaki index değerine ait soru numarasını göstermektedir; n>n+1 olduğunda yani 5,6,1 gibi bir sıralama geldiğinde "1"'e geçilirse test sayısının artacağı anlamına gelmektedir. Ve while döngüsü kırılarak diğer testte işlemlerin yapılması için gerekli işlemler uygulanır.

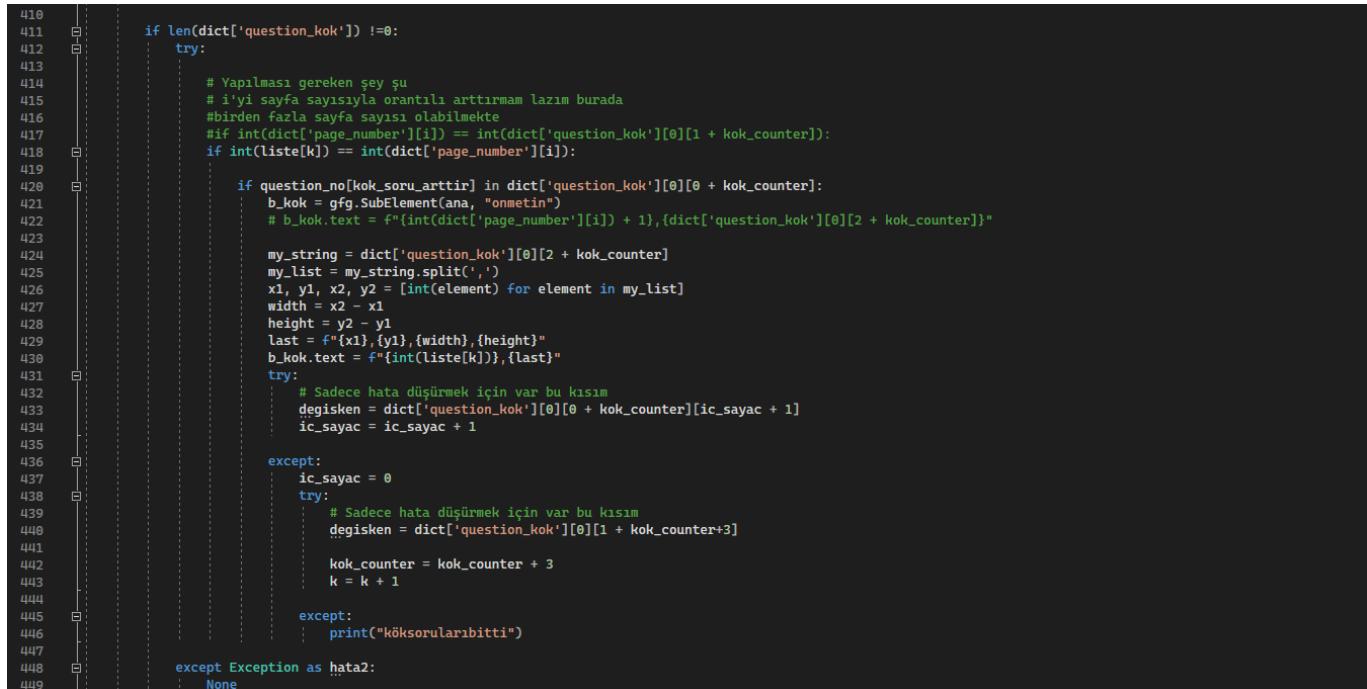


```

367     item = gfg.Element("test")
368     item.set("ID", f"{str(test_counter)}")
369     item.set("ADIN", f"{str(ust_yazi)}")
370     item.set("BASARIORANT", "50")
371     item.set("SECENEKSAYISI", f"{str(secenek_kontrol)}")
372     item.set("BaslangicSayfasi", f"{baslangic_page}")
373     item.set("BitisSayfasi", f"{bitis_page}")
374     baslangic_page_kontrol = True
375     textcomp_tothequestion = f'{int((imgsize[i][1]*2)/100)},{0},{int((imgsize[i][1]-(imgsize[i][1]*2)/100))},{int((imgsize[i][0]*5)/100)}'
376     item.set("testCerceve", f'{str(textcomp_tothequestion)}')
377     root.append(item)
378   condition = False
379
380   #item = gfg.Element("test")
381   #root.append(item)
382
383   ana = gfg.SubElement(item,"item")
384   b1 = gfg.SubElement(ana, "cevap")
385   b1.text = f"-"
386
387   b2 = gfg.SubElement(ana, "testID")
388   b2.text = f"{test_counter}"
389   try:
390     if question_no[i] > question_no[i + 1]:
391       test_counter = test_counter + 1
392
393     # Sorular tekrardan başlıyacağı için sıfırlamayı burada yapıyorum
394
395     #kok_soru_arttir = 0
396
397     condition = True
398   except:
399     print("Yeni Teste Atlandı")
400
401   bex = gfg.SubElement(ana, "imgsize")
402
403   bex.text = f'{imgsize[i][0]},{imgsize[i][1]}'
404
405   bex = gfg.SubElement(ana, "soruNo")
406   bex.text = f'{question_no[i]}'
407
408   b3 = gfg.SubElement(ana, "page")
409   b3.text = f'{int(dict["page_number"][i]) - 1}'
410

```

Eğer ilk kısımda köklü soruyu tespit edip veri eklediysek bu kısıma gelmekteyiz, xml çıktısına köklü soruları kapsayan tag'leri buradan eklemekte ve yönetmekteyiz.



```

410
411   if len(dict['question_kok']) !=0:
412     try:
413
414       # Yapılması gereken şey şu
415       # i'yi sayfa sayısıyla orantılı artırmam lazımlı burada
416       # birden fazla sayfa sayısı olabilmekte
417       #if int(dict['page_number'][i]) == int(dict['question_kok'][0][1 + kok_counter]):
418       if int(liste[k]) == int(dict['page_number'][i]):
419
420         if question_no[kok_soru_arttir] in dict['question_kok'][0][0 + kok_counter]:
421           b_kok = gfg.SubElement(ana, "onmetin")
422           # b_kok.text = f'{int(dict['page_number'][i]) + 1},{dict['question_kok'][0][2 + kok_counter]}'
423
424           my_string = dict['question_kok'][0][2 + kok_counter]
425           my_list = my_string.split(',')
426           x1, y1, x2, y2 = [int(element) for element in my_list]
427           width = x2 - x1
428           height = y2 - y1
429           last = f'{x1},{y1},{width},{height}'
430           b_kok.text = f'{int(liste[k])},{last}'
431         try:
432           # Sadece hata düşürmek için var bu kısım
433           degisken = dict['question_kok'][0][0 + kok_counter][ic_sayac + 1]
434           ic_sayac = ic_sayac + 1
435
436         except:
437           ic_sayac = 0
438         try:
439           # Sadece hata düşürmek için var bu kısım
440           degisken = dict['question_kok'][0][1 + kok_counter+3]
441
442           kok_counter = kok_counter + 3
443           k = k + 1
444
445         except:
446           print("köksorularıbitti")
447
448       except Exception as hata2:
449         None

```

Devamında da .xml'i yönetmemizi sağlayan kodları kullanarak genel çerçeveyemi oluşturuyoruz.

```

463     #ic_sayac = ic_sayac + 1
464
465     # Soru numarası hizasından kesilmiş soru
466     if dict['question_number_box'][0][calculator3] == '9':
467         textcomp_tothequestion = f'{dict["whole_question_box"][0][calculator]}, {dict["whole_question_box"][0][calculator2]}, {dict["whole_question_box"][0][calculator3]}, {dict["whole_question_box"][0][calculator4]}'
468         tothequestion = gfg.SubElement(ana, "rect")
469         tothequestion.text = f'{textcomp_tothequestion}'
470
471     else:
472         textcomp_tothequestion = f'{dict["question_number_box"][0][calculator3]}, {dict["whole_question_box"][0][calculator2]}, {dict["whole_question_box"][0][calculator3]}, {dict["whole_question_box"][0][calculator4]}'
473         tothequestion = gfg.SubElement(ana, "rect")
474         tothequestion.text = f'{textcomp_tothequestion}'
475
476     b4_extra = gfg.SubElement(ana, "secenekler")
477     b4_extra.set("secenek", "A")
478     b4_extra.set("X", f'{dict["answer_a"][0][calculator]}')
479     b4_extra.set("V", f'{dict["answer_a"][0][calculator + 1]}')
480
481     b5_extra = gfg.SubElement(ana, "secenekler")
482     b5_extra.set("secenek", "B")
483     b5_extra.set("X", f'{dict["answer_b"][0][calculator]}')
484     b5_extra.set("V", f'{dict["answer_b"][0][calculator + 1]}')
485
486     b6_extra = gfg.SubElement(ana, "secenekler")
487     b6_extra.set("secenek", "C")
488     b6_extra.set("X", f'{dict["answer_c"][0][calculator]}')
489     b6_extra.set("V", f'{dict["answer_c"][0][calculator + 1]}')
490
491     if len(dict["answer_d"][0]) > 1 and dict["answer_d"][0][calculator] != "0":
492         b7_extra = gfg.SubElement(ana, "secenekler")
493         b7_extra.set("secenek", "D")
494         b7_extra.set("X", f'{dict["answer_d"][0][calculator]}')
495         b7_extra.set("V", f'{dict["answer_d"][0][calculator + 1]}')
496     if secenek_kontrol == 5:
497         if dict["answer_e"][0][calculator] != "-" or dict["answer_key_box"][0][calculator] != "0":
498             b8_extra = gfg.SubElement(ana, "secenekler")
499             b8_extra.set("secenek", "E")
500             b8_extra.set("X", f'{dict["answer_e"][0][calculator]}')
501             b8_extra.set("V", f'{dict["answer_e"][0][calculator + 1]}')
502
503     calculator = calculator + 4
504     calculator2 = calculator2 + 4
505     calculator3 = calculator3 + 4
506     calculator4 = calculator4 + 4
507
508
509     tree = gfg.ElementTree(root)
510     with open(fileName, "wb") as files:
511         tree.write(files)
512
513
514

```

Böylelikle kod aşaması tamamlanmış bulunmaktadır.

5. Uygulamanın Execute Edilmesi

Bu işlem için sanal bir anaconda ortamı kurulması gerekmektedir. Burada sadece uygulamanın gerektirdiği kütüphaneler kurulu olup fazla veri kalabalığına engel olmak amaçlanmaktadır.

İlk olarak anaconda sisteminizde yüklü olmak zorundadır, bunun için aşağıdaki anaconda yazısına tıklayabilirsiniz.

Erişmek İçin =>[Anaconda](#)

Sonrasında cmd'yi açarak gerekli işlem adımlarına başlıyoruz.

```
conda create -n ortam_adi python=3.9.12
```

Komut satırını kullanarak kendimize yeni bir ortam oluşturuyoruz.

```
C:\Users\Ctnkaya>conda create -n ortam_adi python=3.9.12
Collecting package metadata (current_repodata.json): done
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.3.1
  latest version: 23.5.0

Please update conda by running

  $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

  conda install conda=23.5.0

## Package Plan ##

environment location: C:\Users\Ctnkaya\anaconda3\envs\ortam_adi

added / updated specs:
- python=3.9.12

The following packages will be downloaded:

package          | build
-----|-----
ca-certificates-2023.05.30 | haa95532_0
openssl-1.1.1u   | h2bbff1b_0
pip-23.1.2      | py39haa95532_0
python-3.9.12   | h6244533_0
setuptools-67.8.0 | py39haa95532_0
sqlite-3.41.2    | h2bbff1b_0
tzdata-2023c    | h04d1e81_0
wheel-0.38.4     | py39haa95532_0
Total:           27.6 MB

The following NEW packages will be INSTALLED:

ca-certificates      pkgs/main/win-64::ca-certificates-2023.05.30-haa95532_0
openssl              pkgs/main/win-64::openssl-1.1.1u-h2bbff1b_0
pip                  pkgs/main/win-64::pip-23.1.2-py39haa95532_0
python               pkgs/main/win-64::python-3.9.12-h6244533_0
setuptools            pkgs/main/win-64::setuptools-67.8.0-py39haa95532_0
sqlite               pkgs/main/win-64::sqlite-3.41.2-h2bbff1b_0
tzdata                pkgs/main/noarch::tzdata-2023c-h04d1e81_0
vc                   pkgs/main/win-64::vc-14.2-h21ff451_1
vs2015_runtime        pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
wheel                 pkgs/main/win-64::wheel-0.38.4-py39haa95532_0

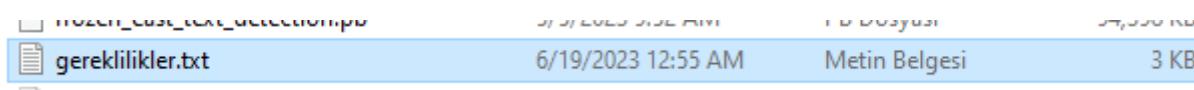
Proceed ([y]/n)? Y olarak girelim

Downloading and Extracting Packages
```

Oluşturulan ortamı aktif ederek içerisinde paket yüklenebilir hale getiriyoruz aksi takdirde paket kurulumları localde kalacaktır.

```
C:\Users\Ctnkaya>conda activate ortam_adi
(ortam_adi) C:\Users\Ctnkaya>
```

Sonrasında yolov7 içerisinde yer alan **gereklikler.txt** dosyasının konumuna ulaşmamız gerekmektedir.



Gerekli konuma ulaştıktan sonra cmd ekranına aşağıdaki komut satırını girelim.

```
(ortam_adi) F:\yolov7>pip install -r gereklilikler.txt
```

```
(ortam_adi) F:\yolov7>pip install -r gereklilikler.txt
Collecting aiohttp==3.8.4 (from -r gereklilikler.txt (line 1))
  Downloading aiohttp-3.8.4-cp39-cp39-win_amd64.whl (323 kB)
    ..... 323.6/323.6 kB 6.7 MB/s eta 0:00:00
Collecting aiosignal==1.3.1 (from -r gereklilikler.txt (line 2))
  Downloading aiosignal-1.3.1-py3-none-any.whl (7.6 kB)
Collecting altgraph==0.17.3 (from -r gereklilikler.txt (line 3))
  Downloading altgraph-0.17.3-py2.py3-none-any.whl (21 kB)
Collecting anyio==3.6.2 (from -r gereklilikler.txt (line 4))
  Downloading anyio-3.6.2-py3-none-any.whl (80 kB)
    ..... 80.6/80.6 kB 4.4 MB/s eta 0:00:00
Collecting astor==0.8.1 (from -r gereklilikler.txt (line 5))
  Downloading astor-0.8.1-py2.py3-none-any.whl (27 kB)
Collecting async-timeout==4.0.2 (from -r gereklilikler.txt (line 6))
  Using cached async_timeout-4.0.2-py3-none-any.whl (5.8 kB)
Collecting attrdict==2.0.1 (from -r gereklilikler.txt (line 7))
  Downloading attrdict-2.0.1-py2.py3-none-any.whl (9.9 kB)
Collecting attrs==22.2.0 (from -r gereklilikler.txt (line 8))
  Downloading attrs-22.2.0-py3-none-any.whl (60 kB)
    ..... 60.0/60.0 kB ? eta 0:00:00
Collecting Babel==2.12.1 (from -r gereklilikler.txt (line 9))
  Downloading Babel-2.12.1-py3-none-any.whl (10.1 kB)
    ..... 10.1/10.1 kB 10.9 MB/s eta 0:00:00
Collecting bce-python-sdk==0.8.83 (from -r gereklilikler.txt (line 10))
  Downloading bce_python_sdk-0.8.83-py3-none-any.whl (210 kB)
    ..... 210.5/210.5 kB 13.4 MB/s eta 0:00:00
Collecting bcrypt==4.0.1 (from -r gereklilikler.txt (line 11))
  Downloading bcrypt-4.0.1-cp36-ab13-win_amd64.whl (152 kB)
    ..... 152.9/152.9 kB 9.5 MB/s eta 0:00:00
Collecting beautifulsoup4==4.12.2 (from -r gereklilikler.txt (line 12))
  Downloading beautifulsoup4-4.12.2-py3-none-any.whl (142 kB)
    ..... 143.0/143.0 kB 8.8 MB/s eta 0:00:00
Collecting blinker==1.6.2 (from -r gereklilikler.txt (line 13))
  Downloading blinker-1.6.2-py3-none-any.whl (13 kB)
Collecting Brotli==1.0.9 (from -r gereklilikler.txt (line 14))
  Using cached Brotli-1.0.9-cp39-cp39-win_amd64.whl (383 kB)
Collecting cachetools==5.3.0 (from -r gereklilikler.txt (line 15))
  Using cached cachetools-5.3.0-py3-none-any.whl (9.3 kB)
Collecting certifi==2021.10.8 (from -r gereklilikler.txt (line 16))
  Downloading certifi-2021.10.8-py2.py3-none-any.whl (149 kB)
    ..... 149.2/149.2 kB 8.7 MB/s eta 0:00:00
Requirement already satisfied: cffi==1.15.1 in c:\users\ctnkaya\appdata\roaming\python\python39\site-packages (from -r gereklilikler.txt (line 17)) (1.15.1)
Collecting charset-normalizer==2.1.1 (from -r gereklilikler.txt (line 18))
  Downloading charset_normalizer-2.1.1-py3-none-any.whl (39 kB)
Collecting click==8.1.3 (from -r gereklilikler.txt (line 19))
  Using cached click-8.1.3-py3-none-any.whl (96 kB)
Collecting colorama==0.4.6 (from -r gereklilikler.txt (line 20))
  Using cached colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Collecting contourpy==1.0.6 (from -r gereklilikler.txt (line 21))
  Downloading contourpy-1.0.6-cp39-cp39-win_amd64.whl (161 kB)
    ..... 161.3/161.3 kB 18.1 MB/s eta 0:00:00
Collecting cryptography==40.0.2 (from -r gereklilikler.txt (line 22))
  Downloading cryptography-40.0.2-cp36-ab13-win_amd64.whl (2.6 MB)
    ..... 2.6/2.6 kB 11.1 MB/s eta 0:00:00
Collecting cssselect==1.2.0 (from -r gereklilikler.txt (line 23))
  Using cached cssselect-1.2.0-py3-none-any.whl (18 kB)
Collecting cssutils==2.6.0 (from -r gereklilikler.txt (line 24))
  Downloading cssutils-2.6.0-py3-none-any.whl (399 kB)
```

... paketler devam etmekte

Böylelikle uygulama paketleri kurulmuş olacaktır.

Uygulamanın execute edilmesi aşamasında **pyinstaller** kullanılması gerekmektedir. Çıktı kodunda yer alan lokasyonlar geliştiricinin kendi cihazına göre ayarlanması ve verilerin oradan çekilmesini sağlaması gerekmektedir.

```
pyinstaller --noconfirm --onedir --console --add-data
"C:/ham_dosya/yolov7/models;models/" --add-data
"C:/ham_dosya/yolov7/scripts;scripts/" --add-data
"C:/ham_dosya/yolov7/utils;utils/" --add-data
"C:/ham_dosya/yolov7/page_number_founder.py;." --add-data
"C:/ham_dosya/yolov7/goz_ugulamasi.py;." --add-data
"C:/ham_dosya/yolov7/openFile.py;." --add-data
"C:/ham_dosya/yolov7/create_XML.py;." --add-data
"C:/ham_dosya/yolov7/best278.pt;." --add-data
"C:/ham_dosya/yolov7/page_number_2_withtext.py;." --add-binary "C:/Program
Files/Tesseract-OCR/tesseract.exe;." --add-data "C:/Program Files/Tesseract-
OCR/iconv.dll;." --add-data "C:/Program Files/Tesseract-OCR/icudt64.dll;." --add-
data "C:/Program Files/Tesseract-OCR/icuin64.dll;." --add-data "C:/Program
Files/Tesseract-OCR/icuuc64.dll;." --add-data "C:/Program Files/Tesseract-
OCR/libarchive-13.dll;." --add-data "C:/Program Files/Tesseract-OCR/libbz2-
```

```
1.dll;." --add-data "C:/Program Files/Tesseract-OCR/libcairo-2.dll;." --add-data
"C:/Program Files/Tesseract-OCR/libcurl-4.dll;." --add-data "C:/Program
Files/Tesseract-OCR/libeay32.dll;." --add-data "C:/Program Files/Tesseract-
OCR/libexpat-1.dll;." --add-data "C:/Program Files/Tesseract-OCR/libffi-6.dll;." -
--add-data "C:/Program Files/Tesseract-OCR/libfontconfig-1.dll;." --add-data
"C:/Program Files/Tesseract-OCR/libfreetype-6.dll;." --add-data "C:/Program
Files/Tesseract-OCR/libgcc_s_seh-1.dll;." --add-data "C:/Program Files/Tesseract-
OCR/libgif-7.dll;." --add-data "C:/Program Files/Tesseract-OCR/libglib-2.0-
0.dll;." --add-data "C:/Program Files/Tesseract-OCR/libgobject-2.0-0.dll;." --add-
data "C:/Program Files/Tesseract-OCR/libgomp-1.dll;." --add-data "C:/Program
Files/Tesseract-OCR/libharfbuzz-0.dll;." --add-data "C:/Program Files/Tesseract-
OCR/libidn2-0.dll;." --add-data "C:/Program Files/Tesseract-OCR/libintl-8.dll;." -
--add-data "C:/Program Files/Tesseract-OCR/libjbig-2.dll;." --add-data "C:/Program
Files/Tesseract-OCR/libjpeg-8.dll;." --add-data "C:/Program Files/Tesseract-
OCR/liblept-5.dll;." --add-data "C:/Program Files/Tesseract-OCR/liblz4-1.dll;." --
add-data "C:/Program Files/Tesseract-OCR/liblzma-5.dll;." --add-data "C:/Program
Files/Tesseract-OCR/liblzo2-2.dll;." --add-data "C:/Program Files/Tesseract-
OCR/libnettle-6.dll;." --add-data "C:/Program Files/Tesseract-OCR/libnghttp2-
14.dll;." --add-data "C:/Program Files/Tesseract-OCR/libopenjp2.dll;." --add-data
"C:/Program Files/Tesseract-OCR/libpango-1.0-0.dll;." --add-data "C:/Program
Files/Tesseract-OCR/libpangocairo-1.0-0.dll;." --add-data "C:/Program
Files/Tesseract-OCR/libpangoft2-1.0-0.dll;." --add-data "C:/Program
Files/Tesseract-OCR/libpangowin32-1.0-0.dll;." --add-data "C:/Program
Files/Tesseract-OCR/libpcre-1.dll;." --add-data "C:/Program Files/Tesseract-
OCR/libpixman-1-0.dll;." --add-data "C:/Program Files/Tesseract-OCR/libpng16-
16.dll;." --add-data "C:/Program Files/Tesseract-OCR/libssh2-1.dll;." --add-data
"C:/Program Files/Tesseract-OCR/libstdc++-6.dll;." --add-data "C:/Program
Files/Tesseract-OCR/libtesseract-5.dll;." --add-data "C:/Program Files/Tesseract-
OCR/libtiff-5.dll;." --add-data "C:/Program Files/Tesseract-OCR/libunistring-
2.dll;." --add-data "C:/Program Files/Tesseract-OCR/libwebp-7.dll;." --add-data
"C:/Program Files/Tesseract-OCR/libwinpthread-1.dll;." --add-data "C:/Program
Files/Tesseract-OCR/libxml2-2.dll;." --add-data "C:/Program Files/Tesseract-
OCR/libzstd-1.dll;." --add-data "C:/Program Files/Tesseract-OCR/ssleay32.dll;." --
add-data "C:/Program Files/Tesseract-OCR/zlib1.dll;." --add-data "C:/Program
Files/Tesseract-OCR/tessdata;tessdata/" --collect-all "easyocr"
"C:/ham_dosya/yolov7/detect2.py"
```

Bu işlem sonrasında cmd'nin bulunduğu lokasyonda detect2 klasörü oluşacak ve içerisindeki dist klasöründe gerekli exe dosyalarıyla kütüphaneleri bulunacaktır.

Ad	Değiştirme tarihi	Tür
build	2/22/2023 12:17 AM	Dosya klasörü
dist	2/22/2023 12:18 AM	Dosya klasörü

Ad	Değiştirme tarihi	Tür	Boyut
.vs	3/15/2023 9:23 AM	Dosya klasörü	
certifi	2/22/2023 12:19 AM	Dosya klasörü	
contourpy	2/22/2023 12:18 AM	Dosya klasörü	
cv2	2/22/2023 12:19 AM	Dosya klasörü	
kiwisolver	2/22/2023 12:18 AM	Dosya klasörü	
matplotlib	2/22/2023 12:18 AM	Dosya klasörü	
models	2/22/2023 12:19 AM	Dosya klasörü	
numpy	2/22/2023 12:18 AM	Dosya klasörü	
pandas	2/22/2023 12:18 AM	Dosya klasörü	
PIL	2/22/2023 12:18 AM	Dosya klasörü	
pytz	2/22/2023 12:18 AM	Dosya klasörü	
scipy	2/22/2023 12:18 AM	Dosya klasörü	
scipy.libs	2/22/2023 12:19 AM	Dosya klasörü	
scripts	2/22/2023 12:19 AM	Dosya klasörü	
setuptools-65.5.0-py3.9.egg-info	2/22/2023 12:19 AM	Dosya klasörü	
skimage	2/22/2023 12:18 AM	Dosya klasörü	
tessdata	2/22/2023 12:19 AM	Dosya klasörü	
torch	2/22/2023 12:19 AM	Dosya klasörü	
torchvision	2/22/2023 12:18 AM	Dosya klasörü	
utils	2/22/2023 12:19 AM	Dosya klasörü	
wheel-0.37.1-py3.9.egg-info	2/22/2023 12:19 AM	Dosya klasörü	
yaml	2/22/2023 12:18 AM	Dosya klasörü	
_asyncio.pyd	2/22/2023 12:14 AM	Python Extension ...	56 KB
_bz2.pyd	2/22/2023 12:14 AM	Python Extension ...	75 KB
_ctypes.pyd	2/22/2023 12:14 AM	Python Extension ...	115 KB
_decimal.pyd	2/22/2023 12:14 AM	Python Extension ...	258 KB
_elementtree.pyd	2/22/2023 12:14 AM	Python Extension ...	176 KB
_hashlib.pyd	2/22/2023 12:14 AM	Python Extension ...	54 KB
_lzma.pyd	2/22/2023 12:14 AM	Python Extension ...	152 KB
_multiprocessing.pyd	2/22/2023 12:14 AM	Python Extension ...	22 KB
_overlapped.pyd	2/22/2023 12:14 AM	Python Extension ...	37 KB
_queue.pyd	2/22/2023 12:14 AM	Python Extension ...	21 KB
_socket.pyd	2/22/2023 12:14 AM	Python Extension ...	71 KB

Dist klasörünü teslim ederek bütün adımları tamamlamış olacaksınız.

Yazan: Yiğit Çetinkaya

Dokümantasyonun sonu