



Bitki Hastalıkları Projesi

Genel Bilgilendirme Raporu

Ahmet efe Y-

27 Ocak 2026 Salı

Program hakkında detaylı bilgilere geçmeden önce iki tane veri setinden bahsedicem;

Bu veri setleri programın yapı taşlarını oluşturan veri setleridir ve iki tane yapay zeka modelide bu iki veri seti ile eğitilmiş durumdadır;

*Bu veri setlerinden bir tanesi **Plant Village** Veri setidir. Bu veri seti ile **Softmax** çıktı veren **Çoklu sınıf ile sınıflandırma yapan** modeli eğittim ve **Plant Doc** isimli Dataset ilede **0-1 Arası çıktı veren Sigmoid modelini eğittim***

Veri setlerinin linkleri;

<https://github.com/pratikkayal/PlantDoc-Dataset>

<https://www.kaggle.com/datasets/arjuntejaswi/plant-village>

ÖNEMLİ NOT: Çoklu hastalık sınıflandırma modelindeki veri seti yüzünden validation seti %5 lik bir oranda gelmiş o yüzden o seti Plant Village Veri setinin testleri ile birleştirmek zorunda kaldım buda modelin validation loss değerlerinin 1.x 2.x gibi aralıklarda yerinde saymasına sebep oldu test verisi tutarsızlığı sebepli olarak validation

loss ve validation accuracy değerleri yaniltıcı olabilir bu yüzden o kısımda accuracy + loss ve daha farklı test yöntemleri ile modeli test etmek zorunda kaldım

ÖNEMLİ NOT-2: Bu programda bitkilerin fotoğrafını programa verirken lütfen aşağıdaki kriterlere uyunuz;

*Damar yapıları belli olmalı
Kadrajda sadece o bitki olmalı
Işıklandırma düzgün olmalı
Aşağıdaki kriterlere uymalı*



*Bitki hastalıkları projesi için geliştirmekte olduğum programın kabaca %65 i tamamlanmış durumdadır; bu projede **Temelde hastalık tespiti-hasta/sağlıklı ayrimı** olmak üzere iki özellik bulunmakta bu özellikler için iki ayrı model bulunmakta bu modeller temel alt yapısında **CNN** yani **Conventional Neural Network'leri** kullanmaktadır **Hasta-Sağlıklı ayrimı** yapan*

*model bizlere 0-1 arası bir değer verir bu değer 1 e ne kadar yakınsa ve 0 dan ne kadar uzak ise bitki o kadar sağlıklıdır metriğine göre programdaki karar mekanizmaları sayesinde ölçülür ve 0.5 ten büyük olan çıktılar sağlıklı (0.4) + 0 ve (0.4) + 0,05 arası gelen değerler ise **Kısmen sağlıklı** yani **Half Healthy** kategorisinde değerlendirilir ve **Hasta** yani **Diseased** olarak değerlendirilenler ise (0.4) + 0 ve (0.4) + 0,05 arasındaki değerlerinde altında olan çıktılardır;*

Model çıktılarının program tarafından yorumlanmasındaki metrikler bu şekilde dile getirilebilir ve teknik olarak açıklanabilir şimdi ise henüz eğitilmemiş olan Hastalık sınıflandırma modeline bir göz atalım;

BU modelde bitki hasta ise ve bu hastalık bilinmiyor ise gelen görseli X hastalık sayısınca tahmin eder ve X tane modele gösterilen hastalık türleri ile en uyumlu eşleşen sınıfı 0-1 arası en 1 e yakın değeri verir örneğin 4 sınıflı bir yapı olsa çıktı şu şekilde olur

$[[0.05616^{\sim}, 0.03043^{\sim}, 0.20510^{\sim}, 0.50331]]$

Şeklinde bir çıktı gelir burada aslında sınıflar şu şekildedir

**{'X hastalığı': 0 , 'Y hastalığı': 1, 'Z hastalığı': 2,
'Q hastalığı': 3}**

yani burada

X hastalığı : 0.05616~

Y hastalığı : 0.03043~

Z hastalığı : 0.20510~

Q hastalığı : 0.50331~

*Şeklinde bir karar verilir ve Q hastalığı tahminler yani **Prediction**'lar arasından en maksimum olanı olarak seçilir yani kısaca karar verilirken çok sınıflı yapılarda model bu şekilde her sınıfa skorlar biçer ve programda bunları metriklere göre analiz eder*

Bu projedeki 0-1 arası sınıflandırma yapan model ile çok sınıflı bir şekilde hastalık tespiti yapan modeli karıştırmayın ikisi farklı modeller ve şimdi

ise 0-1 arası sınıflandırma yapan modelin girdi olarak neler aldığına bakalım;

*Temel olarak bu model bizden girdi olarak görselin 0. axisine batch dimension **Batch boyutu** eklenmiş ve 224x224 olarak yeniden boyutlandırılmış RGB renk uzayına sahip bir görsel ister görseller genel olarak farklı boyutlarda ve çoğunlukla BGR2 renk uzaylarında gelirler ayrıca görüntü matrisleri içinde INT yani tam sayılar bulunur 0-256 arası yani 0 dan 255 e kadar rakamlardan oluşan büyük matrislerdir ve modele bunları vermeden önce ilk başta belirttiğim işlermeli uygulamamız gereklidir bu işlemleri python'da istersek **Tensorflow** un kendi fonksiyonları ile halledebilir veya istersekte harici olarak **opencv-python** ile halledebiliriz bu işleme **preprocess** yani ön işleme denir ön işleme esnasında modelin isteyeceği formatlara görüntü çevrilir.*

*ve bunun bir farklısı ise **Augomention**'dır bu ise hedef dizindeki klasörlerin hepsini bir sınıf olarak kabul eden ve o sınıfların içinden bir akış ile görselleri çekmek ve bir **generator** oluşturmaktır bu generatöre görselleri depolar akış esnasında istenilen görüntü işleme işlemleri uygulanıp modelin besleme verisini daha gerçek dünya*

verisine benzetip modeli zorlar ve öğrenmesini & genelleme yapmasına dahada olanak tanır modelimizin eğitimi ise tam olarak aşağıdaki anlattığım gibi gerçekleşiyor..

Görüntüler generator'de depolanmışken modele bu generator verilir ve model ise bu generator'deki veriler ile kendini besler modele verilir ve besleme aşaması gerçekleşmiş olur ve modelin tahmin yapacağında ise bu aşama tekrarlanır ve tahmin edeceğİ görsel modele verilmeden bu aşamalardan geçirilir ve modele o şekilde veriliki model sapıtmazsa saçma tahminler yapmasın veri tutarlı olsun. Ancak bu aşamada yani model eğitildikten sonra tahmin yapması gerekiğinde tek bir görseli modele vermek için bir generatore sokmak yerine daha çok **skimage opencv** ve **tensorflow** un görüntü fonksiyonları kullanılarak **Augomention** da hangi preprocess işlemleri uygulandı ise aynen yine o işlemler uygulanır

*Temelde bu programda **ResNet50** modeli kullanıldı her iki modelde aslında **ResNet** yani **Residual Network 50** olarak programda mevcuttur burada bu modeli kullanma sebebi ise şu; normalde Sequential gibi modeller sıralı modellerdir ve*

önceden eğitilmemiş 0 yani hazır modellerdir
gerçek dünya problemlerinde Sequential genelde
peki tercih edilmez Sequential daha çok kısa vadeli
modellerde ve bazı benzer küçük datasetlerde güzel
sonuçlar veren **Tensorflow.Keras** in güzel hazır
modellerinden birisidir ancak **ResNet** önceden
yaklaşık 2.3 MİLYON görsel ile beslenmiştir bu
görseller **Araba Bisiklet çiçek bardak vb. vb.** bir
cok görselden oluşur buda modeli çok ağır ve
büyük bir model yapıyor ve buda şunu getiriyor
Sizin modeli eğiteceğiniz verileri model zaten
önce **gördü** ancak sizin verdiğiniz kadar
detaylı görmedi sizde burada modele **Fine Tuning**
yani **ince ayar yaparak layerları**
donduruyorsunuz ve trainableını false
yapıyorsunuz buda tek seferde 50 layeri birden(50
layer içine weight + b vb değerler dahil edilmedi)
eğitmıyoruz sadece sondan n tane layeri trainable
yapıp onların üzerine eğitimi gerçekleştiriyoruz ve
modeli kompile ediyoruz kompile edilmiş modeli
sonraisnda tekrardan eğitiyoruz ve bunu comp - fit
-comp - fit -comp -fit şeklinde tekrarlıyoruz ve buda
modelin ilk başta kabaca öğrenememini sonrasında
ise detaylıca öğrenebilmesini sağlayan bir mimariyi
doğuruyor kısacası ResNet i kullanmanın temel
mutlak sebebi budur daha fazla anlatmaya pekte

gerek yok zaten bu model ile anlatılabilcek çok şey var

Bu aşamalarıda temel olarak anlattığımı göre artık programa geçebilirim;

Programda temel olarak model ile alakalı bir özgürlük mevcut; bahsettiğim şey şu; Yukarıda bahsettiğim 2 tane model ile çalışmak zorunda değil program eğer programın girdisini destekleyen türde bir model bulur iseniz o modeli indirip programa attığınızda program modeli tanıyor ve model ile senkronize çalışabiliyor bu özellik daha geliştirme aşamasındadır bilindiği üzere her modelin farklı girdi boyutları farklı girdi metrikleri ve farklı farklı istekleri bağımlılıkları mevcuttur şu anlık sadece programın girdisine göre tahmin yapabilen modeller program tarafından destekleniyor ancak bunları kullanmak zorunda değilsiniz bu sadece bir alternatif zaten temelde sağlam iki tane model mevcut ancak modellerde hala geliştirme aşamasında;

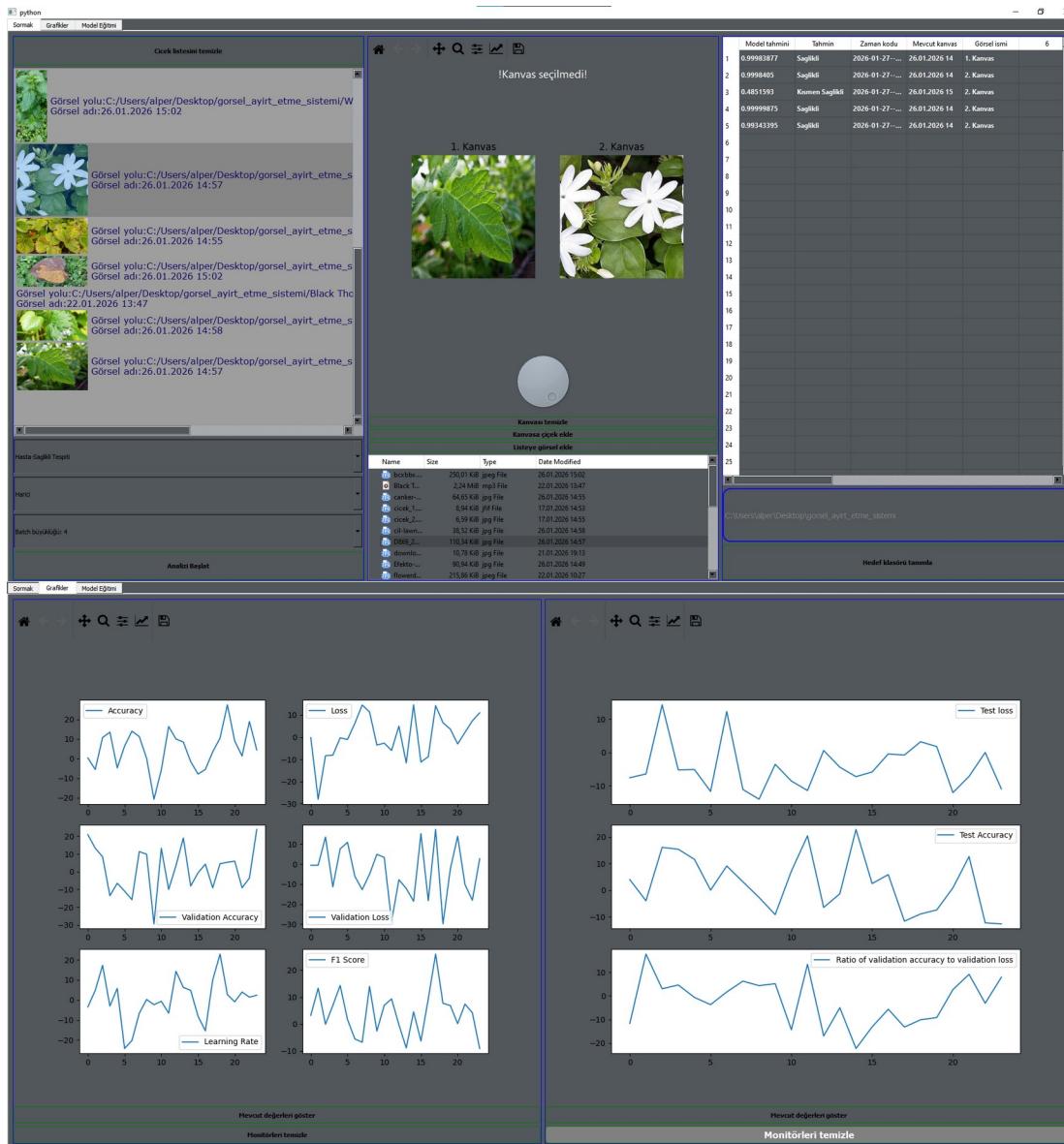
*Ve şimdi ise programdaki asıl özelliğe deşinelim; Programda **Transfer Learning** mevcut yani programı kendiniz eğitebiliyorsunuz ve yaptığı*

*hataları tekrarlamasını azaltabiliyorsunuz Şu anda bu özellik geliştirme aşamasında olsada programın bünyesi bunu kaldıracak mimariye sahip yani bu özellik eklenecek bir özellik ve teknik aksaklıklar çıkmaz ise programa kesin olarak eklenecek özelliklerden birisi bariz olarak budur; Progamda seçtiğiniz modelin **Summary** yani katman bilgilerini parametre sayılarını VB. bilgilerine erişmenizi sağlayan bir sistemde şu anda mevcut*

Ayrıca programdaki bir özellik ise model oluşturma sekmesi; evet bu sekme daha programda teknik olarak mevcut değil ancak en kısa zamanda eklenerek özelliklerden birisi ve teknik mimari bu özelliğe uygun tasarlandı

Ayrıca programdaki log sistemi sayesinde programı yazarken hazırladığım modeller dahil sizin eğiteceğiniz modellerinde tüm eğitim bilgilerinin loglandığı ve bu logların bir sistem aracılığı ile parse edilip sayısal formata dönüştükten sonra grafiklerinin çıkarılacağı bir sistemde programda Mevcuttur. Yani programda bir log + plotting sistemi mevcut olarak bulunmakta

Şimdi ise artık programın model ile olan bağlantıları ve özelliklerini anlatmayı bir kenara bırakıp programın kendisini sunayım;

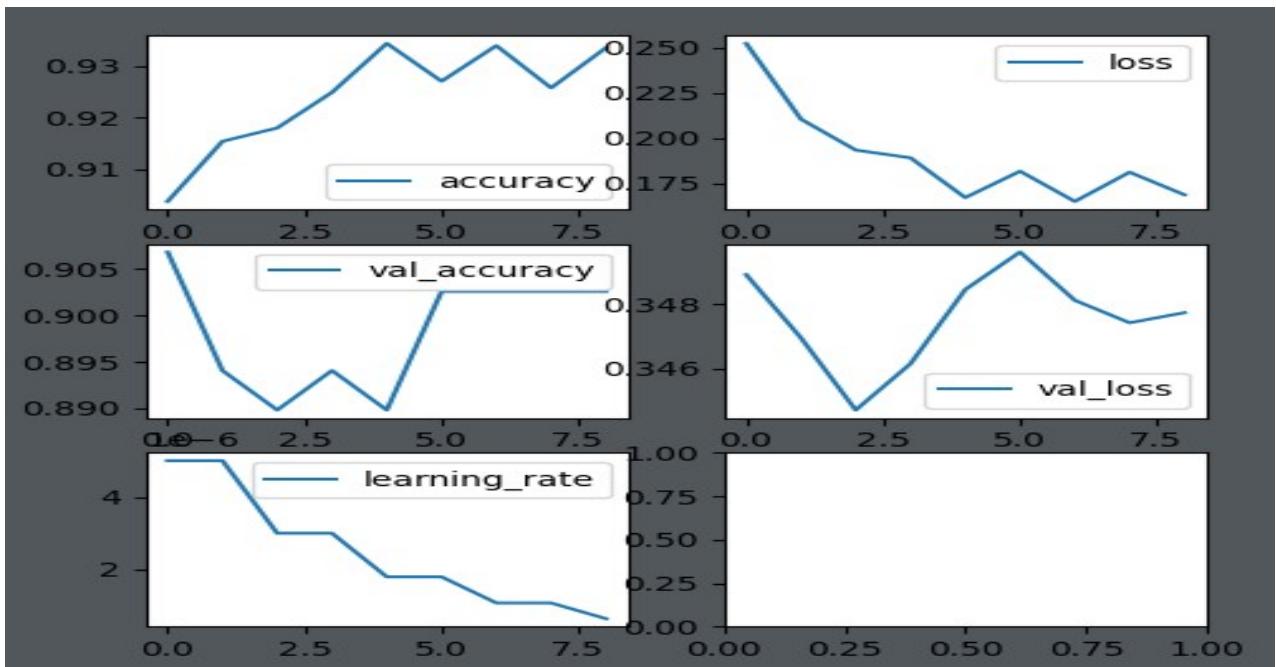




Görmüş olduğunuz üzere programda olduğumu belirttiğim özellikler bu şekilde gözükyor ayrıca eklemeyi unuttuğum şey ise şu;

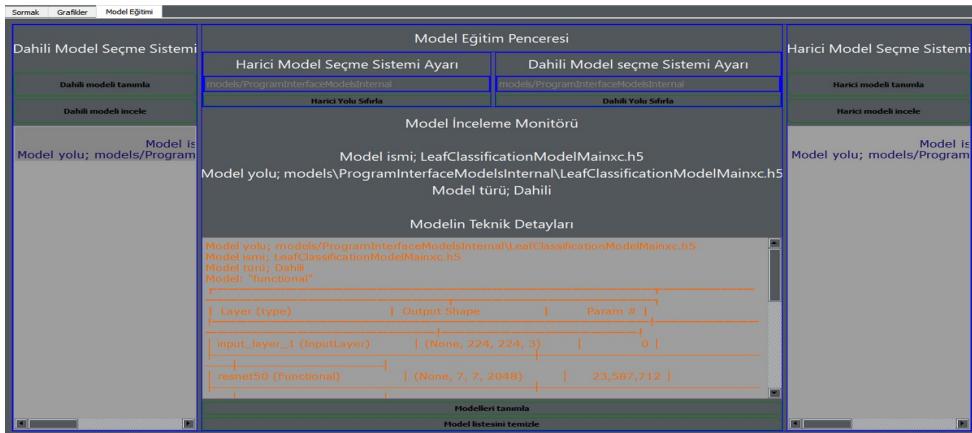
*Modele tahmin yaptırmadan önce **Batch büyüklüğü**, **Model türü**, **Model** gibi etkenleri seçip o yönde bir çıktı alabiliyorsunuz*

Şimdi ise modelin mevcut öğrenme grafiklerine gelelim

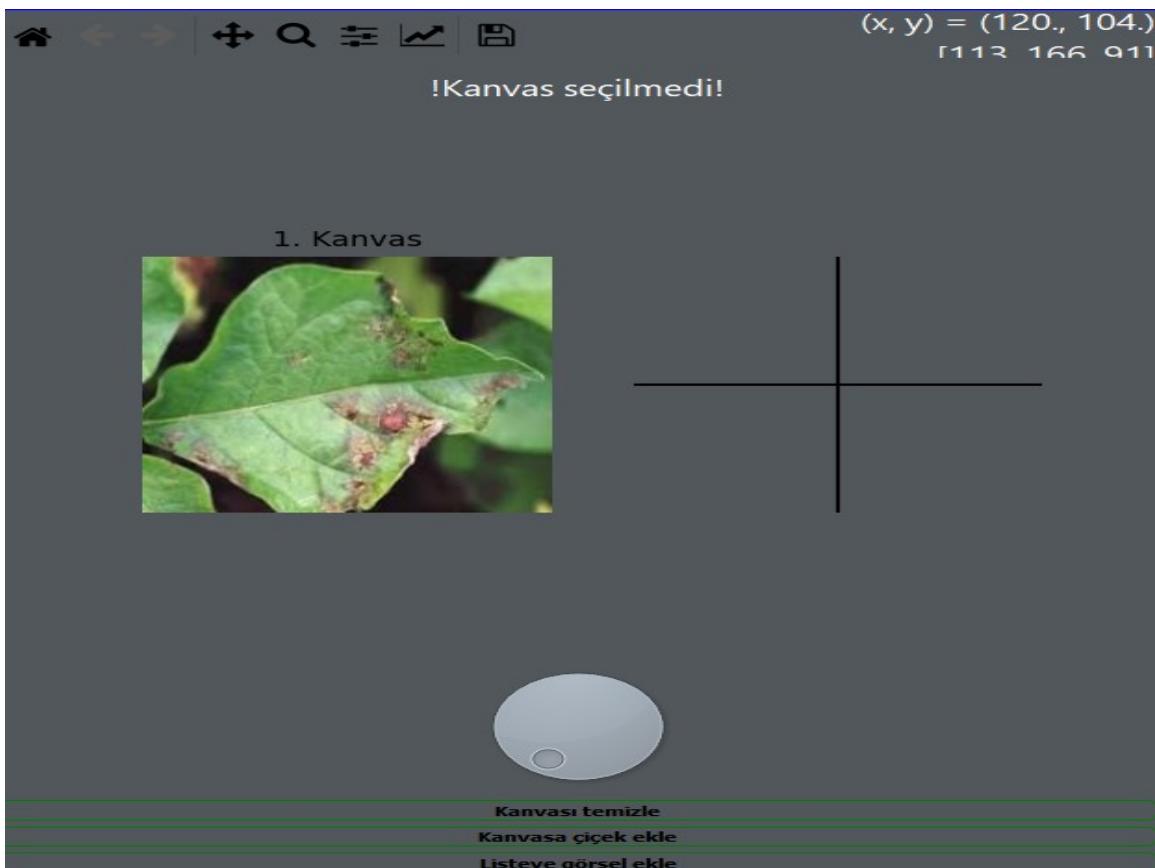


Grafiklerdede görmüş olduğunuz üzere **accuracy ile loss** birbirine ters orantılı ancak göze çarpan bir etken var **Validation Accuracy** ve **Validation Loss** grafikleri bu grafikler normalden farklı olarak bir birine yakın değerlerde başlamış(ki bu iç açıcı bir durum değil) validation accuracy bu grafikte başlarda düşen tarafta ancak accuracy düşerken validation loss ta düşüyor buda grafiğe tekrar bakıldığında iyi bir etken olarak göze

çarpiyor; işte tamda burada devreye learning rate giriyor burada learning rate 2. epoks civarlarında devreye girmiş ve 2. epokstan sonrasında accuracy ufak bir zıplama gerçekleştirmiş aynı şekilde loss ta düşüşte kalmış 4. epoksa yaklaşılrken learning rate tekrardan düşüse geçmiş ve bu seferde validation accuracy artık yükseliş göstermeye başlamış ve validation accuracy yükselirken validation loss ta onunla beraber yükselmeye başlamışki tamda 5. epoksta learning rate tekrar düşmüş ve loss dalgalı bir şekilde düşüş göstermiş accuracy ve loss dengesiz gibi gözüksede teknik olarak şu anlık düşük bir overfit riski taşıyor model bu test verilerine bakıldığında genellemesi orta seviyelerde duruyor model son epoksta EarlyStopping callback'i tarafından durdurulmuştur daha kısa kesmek gerekirse **val loss maksimum 0.348** göstermiş iken **val accuracy maksimum olarak 0.900** civarları göstermiş aradaki fark gözle görülür şekilde belirgin kısacısı model şu anki hali ile **mükemmel seviyede değil ancak karar verme aşamalarında %70 doğruluk** ile çalışıyor örnek olarak bir test yapalım



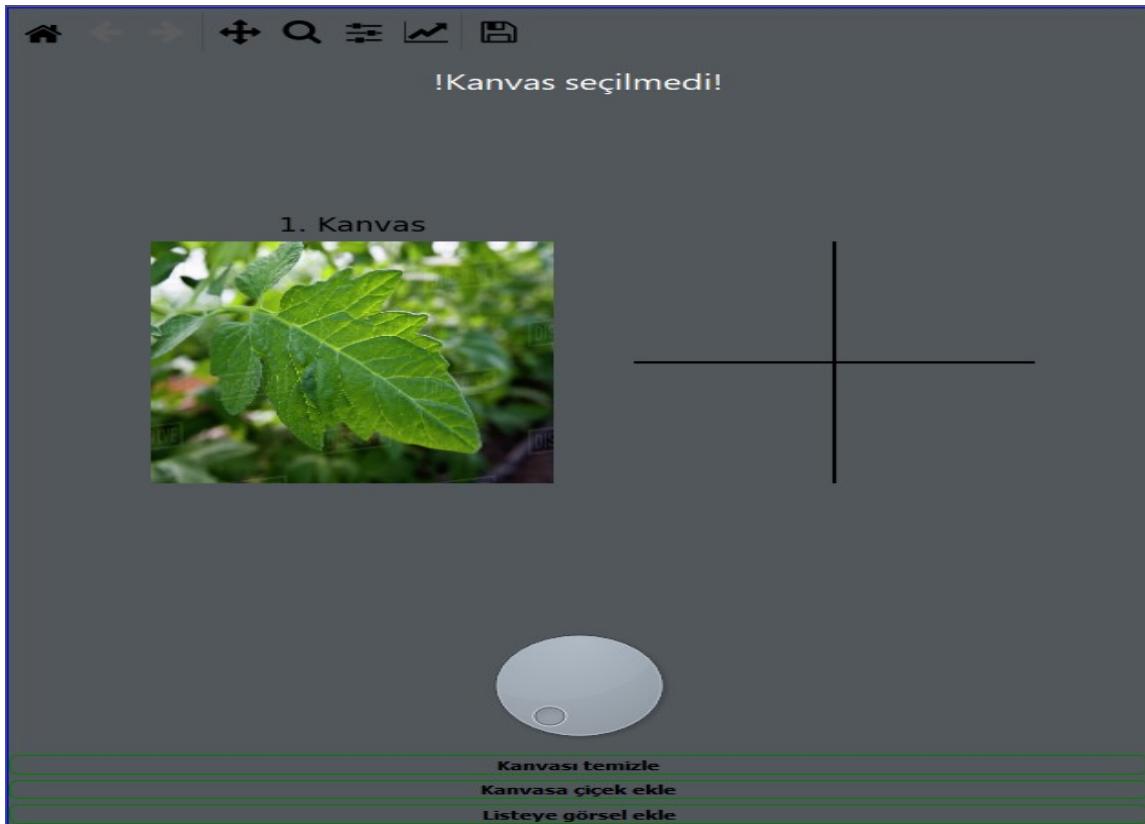
modelimizi tanımladık şimdi ise görselimizi tanımlayalım



*görselimizde tanımladık şimdi ise **analizi başlat** butonuna tıklayalım ve modelin yapacağı tahmine bakalım*

	Model tahmini	Tahmin	Zaman kodu	Mevcut kanvas	Görsel ismi	6
1	0.00031713338	Bitki hasta	2026-02-06--....	indir (1).jfif	1. Kanvas	
2						
3						
4						
5						
6						
7						

*Modelin belirtilen kanvasta görmüş olduğunuz görselle verdiği **sağlıklılık olasılığı** veya diğer bir değişim ile **sağlıklılık skoru** yaklaşık olarak **0.000317** civarında buda bitkinin hasta olduğunu anlayabildiğini gösteriyor şimdi ise sağlıklı bir çiçek ile testimizi tekrarlayalım*



Şimdi ise görselimiz ekranada göründüğü gibidir

*ve modelimizin bu görsele verdiği tahmin ise aynen
su şekildedir*

	Model tahmini	Tahmin	Zaman kodu	Mevcut kanvas	Görsel ismi	6
1	0.00031713338	Bitki hasta	2026-02-06--...	indir (1).jfif	1. Kanvas	
2	0.5132431	Saglikli	2026-02-06--...	26.01.2026 ...	1. Kanvas	
3						
4						
5						
6						
7						
8						

*Yani 1. kanvastaki görselimiz modelimiz **0.5132** civarında bir skor biçti ve sağlıklı etiketini belirtti ancak burada bir husus var*

*Fark etmiş olduğunuz gibi model **0.513** civarı bir değer verdi aslında model full sağlıklı olan bir çiçeğe kaba haesap en az **0.780** civarında bir skor vermeli işte burada o az önce yorumladığımız grafikler devreye giriyor*

O grafiklerdede görüldüğü üzere model ortalama bir başarı gösteriyor ve bunuda testlerimiz ile doğrulamış olduk

not: teknik yorumlardan ek olarak programda model ismi ve seçili kanvasın column etiketlerinde bir kayma olmuş o yüzden seçili kanvasta görsel isimleri görsel isimlerinde ise seçili kanvas yazıyor yani teknik bir aksaklık değil sadece küçük bir dikkatsizlik .)

Model yolu; models/ProgramInterfaceModels/Internal/LeafClassificationModelMainxc.h5

Model ismi; LeafClassificationModelMainxc.h5

Model türü; Dahili

Model; "functional"

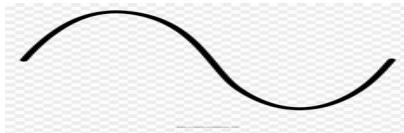
<i>/ Layer (type)</i>	<i>/ Output Shape</i>	<i>/ Param # /</i>
<i>/ input_layer_1 (InputLayer)</i>	<i>/ (None, 224, 224, 3)</i>	<i>0</i>
<i>/ resnet50 (Functional)</i>	<i>/ (None, 7, 7, 2048)</i>	<i>23,587,712</i>
<i>/ global_average_pooling2d (GlobalAveragePooling2D)</i>	<i>/ (None, 2048)</i>	<i>0</i>
<i>/ batch_normalization (BatchNormalization)</i>	<i>/ (None, 2048)</i>	<i>8,192</i>
<i>/ dense (Dense)</i>	<i>/ (None, 256)</i>	<i>524,544</i>
<i>/ dropout (Dropout)</i>	<i>/ (None, 256)</i>	<i>0</i>
<i>/ dense_1 (Dense)</i>	<i>/ (None, 1)</i>	<i>257</i>

Total params: 24,120,707 (92.01 MB)

Trainable params: 16,360,705 (62.41 MB)

Non-trainable params: 7,760,000 (29.60 MB)

Optimizer params: 2 (12.00 B)



**Hastalık Sınıflandırma Modeli(CNN)
ResNet50**

BU modelin teknik olarak tespit edebildiği hastalıklar

Domates bakteriyel leke hastalığı

Domates erken yanıklığı

Domates geç yanıklığı

Domates septorya yaprak lekesi

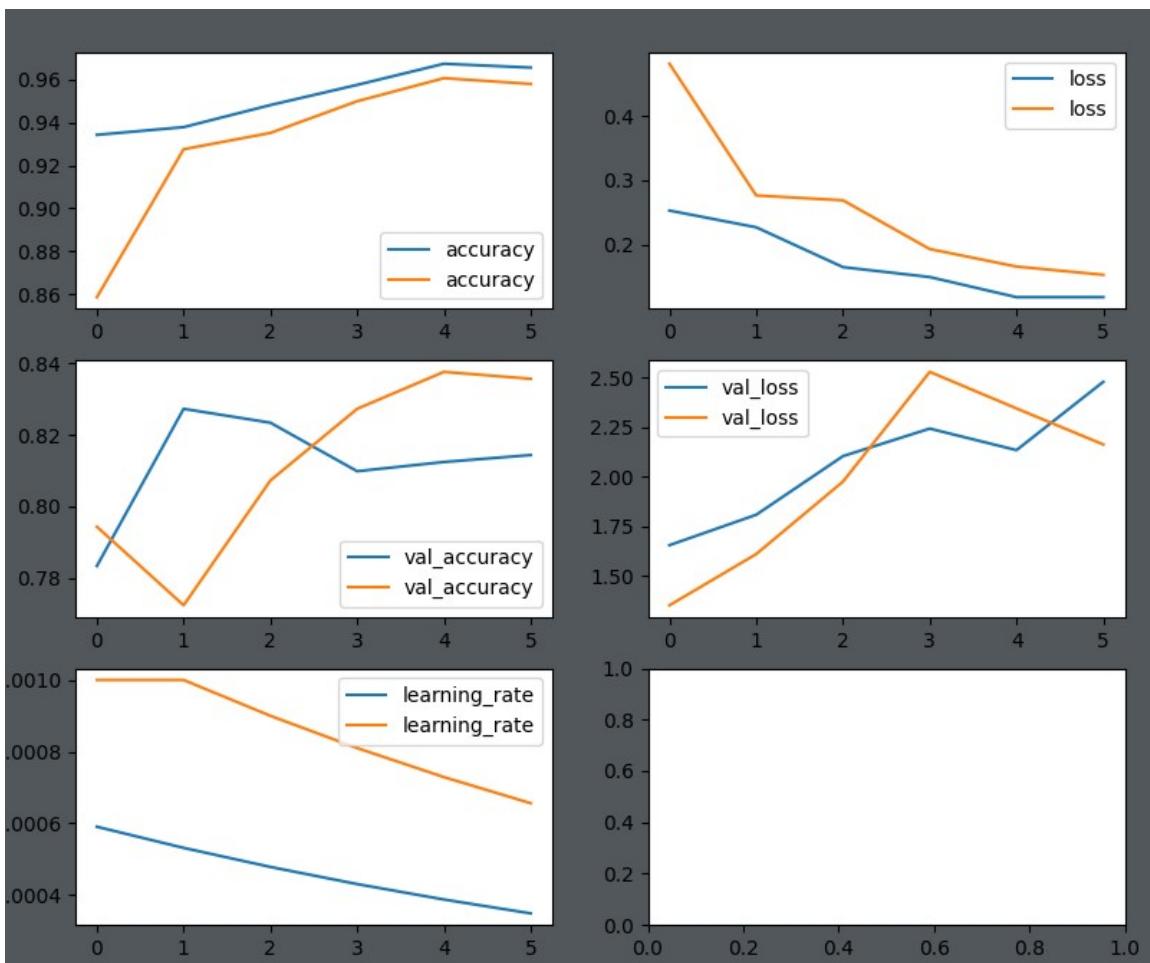
*Model yukarıda belirtmiş olduğum hastalıkları teknik olarak tespit edebiliyor 1. dökümanda belirttiğim gibi burada **Softmax** aktivasyonu kullanıldı yani bize modelin yaptığı tahminden bir vektör geliyor ve sınıf sayısı kadar uzunlukta*

*değerler geliyor maksimum argümanın anahtarı(dict veri yapısında) bize hangi sınıfın en çok olasılık ile model tarafından seçildiğini belirtir yani daha temel bir anlatış ile; **Bize gelen vektörde en büyük olaslığın denk geldiği sınıf ismi bize modelin görseli analiz ederek vermiş olduğu tahmini sınıfı verir ve buda modelin bize görsele dayalı yaptığı olasılık tahminini temsil eder...***
şeklinde temel olarak açıkalanabilir

*Şimdi ise Hangi modeli kullandığımı gelelim; Bunun için kullandığım model **ResNet50 Modeli** olarak geçen Görüntüler üzerinde temel olarak iyi sonuçlar veren bir CNN modelidir*

*Bu modeli biraz daha anlatmam gerekirse; BU model önceden eğitilmiş **Pretrained** modeldir ve biz burada bu modelin katmanlarını yani **Layerlarını** dondurarak sondan 30 tane layer'i tekrar eğittik yani bir sürü layeri tekrar tekrar eğitmek yerine sondan 30 tane layer'i eğittik Buna **Transfer Learning** denir yani kısaca önceden eğitilmiş bir modeli kendi verilerin ile tekrardan eğitmektir*

***Model** eğitilirken 16 epoks ve 16 batch büyüğünü ile eğitilmiştir model toplamda net olarak **4 sınıfı 1546 görselli** Test verisi ve **4 sınıfı ve 2236 görselli** bir **Training** yani **Eğitim** verisi ile eğitilmiştir Modelin eğitim grafikleri ise aşağıda verildiği gibidir*



Grafikler pek iç açıcı durmuyor bu konuda sizlerle beraber bende hem fikirim ancak bunun mutak sebebi şu;

İlk baştada belirttiğim übere Validation seti yani test seti **Plant Doc** Veri setinde çok tutarsız bir şekilde hazırlanmış yani neredeyse %5 lik bir oranda bu nedenle benimmde **Validation** setine yama yapmam yani teknik bir tabir ile **Plant village**'ın test setleri ile birleştirmem gerekti burada her ne kadar çaba göstersemde veri seti bir kez tutarsızdı o yüzden validation loss en az 1.50 Ki bu çok büyük ve kötü bir değer bunu kabul ediyorum ama şimdi bu grafikleri gerçekten yorumlayalım **Çünkü bazen grafikler yanlışlıbilir**

öncelikle accuracy ile loss hem modelin **Fine tuning öncesi** hemde **Fine tuning sonrası** 'nda pozitif yönde bir birine ters orantılı gitmiş bu modelin öğrendiğini gösteriyor Fine tuning öncesi model eğitildikten sonra model tekrar kompile edilip tekrardan eğitilmiştir bu sonra tekrar kompile edillip tekrarlanmıştır... Burada **Mavi çizgi** fine tuning öncesi **turuncu çizgi** fine tuning sonrası temsil etmektedir

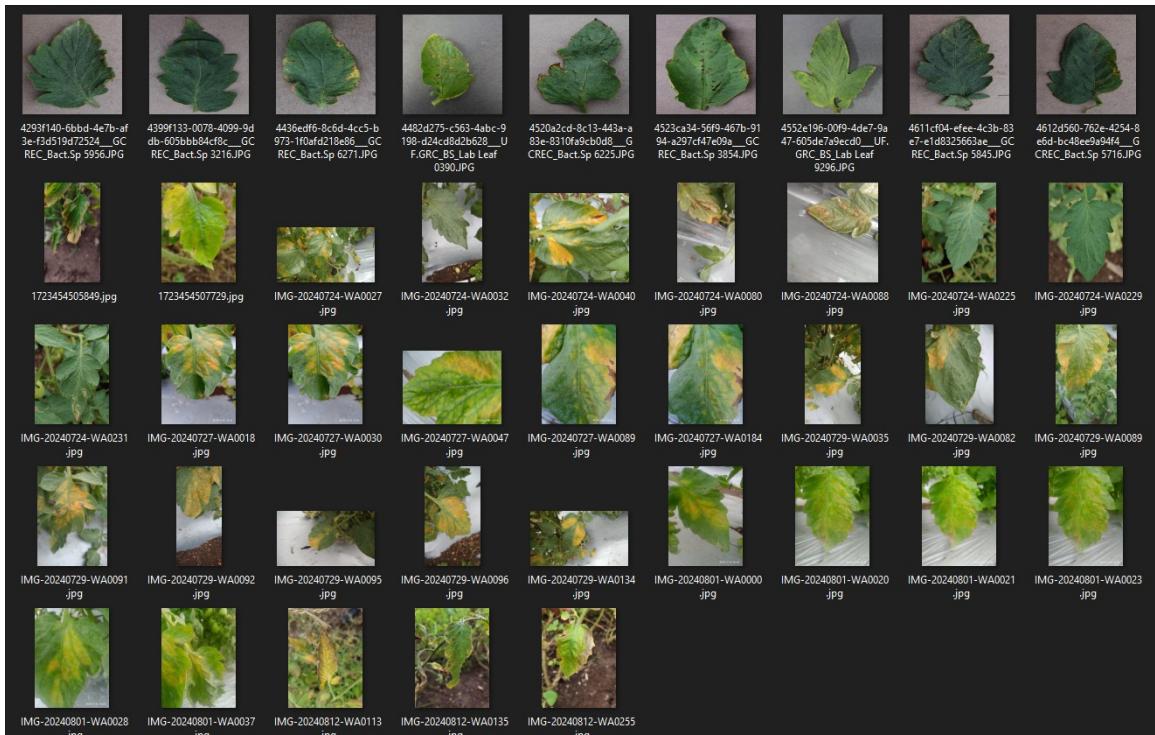
İlk olarak teknik bir terimle başlamam gerek bu grafikleri yorumlamaya çünkü aksi halde çok muğlak bir yorumlama olacak

Bu grafikler normal şartlarda Makine öğrenmesi ve Deep learning'te **Overfitting** olarak kabul edilen modelin **Training** veri setini ezberleyip test veri setinde duvara çarpması yani teknik bir tabir ile çok yüksek validation loss değerleri göstermesi durumudur **ANCAK** şu anda benim eğittiğim modeli bu olgudan ayıran temel bir unsur var **Overfitting** olduğunu kanıtlayabilmemiz için ilk olarak **Test** yani **Validation** veri setinin **Training** yani **Train** yani **Eğitim** veri seti ile orantılı olması & uyumlu olması yani iki setinde bir birinden alakasız / çok farklı olmaması gereklidir eğer dediğim koşul sağlandığı halde bu grafikler bu şekilde olsa idi; Bu bir klasik **Overfitting** senaryosu olmaktan öteye gidemezdi burası tartışmasız bir gerçek ancak bu modelde eğitim ve train setinde sayısal olarak her ne kadar orantılı olsada içerik verileri çokta uyumlu değil bunu kanitlamak için veri Train veri setinden bir kaç görsel ile Test setinden bir kaç görsel vermeme izin verin;

TRAİN SETİ



TEST SETİ



Bu domates bakteriyel enfeksiyon sınıfının train ve test klasöründeki bir kısmının karşılığı ve bakıldığı zaman bir birinin **Işık & Açı & Ortam** farklı olmak üzere train ve test seti dağınık zaten **Plant village** labaratuvar ortamında yaprakları çekerken **Plant docs** veri seti gerçek dünya örneklerinden çekilmiştir buda **Train** ve **Test** veri setlerinin neden bu denli tutarsız olduğunu açıklaya unsurlardan biridir

Şimdi ise bu bilgileri artık biliyoruz şimdi gerçekten bu sınıflandırma modelinin çalıştığını kanıtlayalım ve ortaya attığımız tezin gerçek olduğu kanıtlanmış olsun;

bu modelin sınıflandırma yapısına gelelim yani sınıf yapısına;

```
{'Tomato_Bacterial_spot': 0,
'Tomato_Early_blight': 1,
'Tomato_Late_blight': 2,
'Tomato_Septoria_leaf_spot': 3}
```

Peki ya? Model bu sınıflandırmayı yaptıktan sonra hangisini seçtiğini nereden bilecek? Cevap aslında çok basit

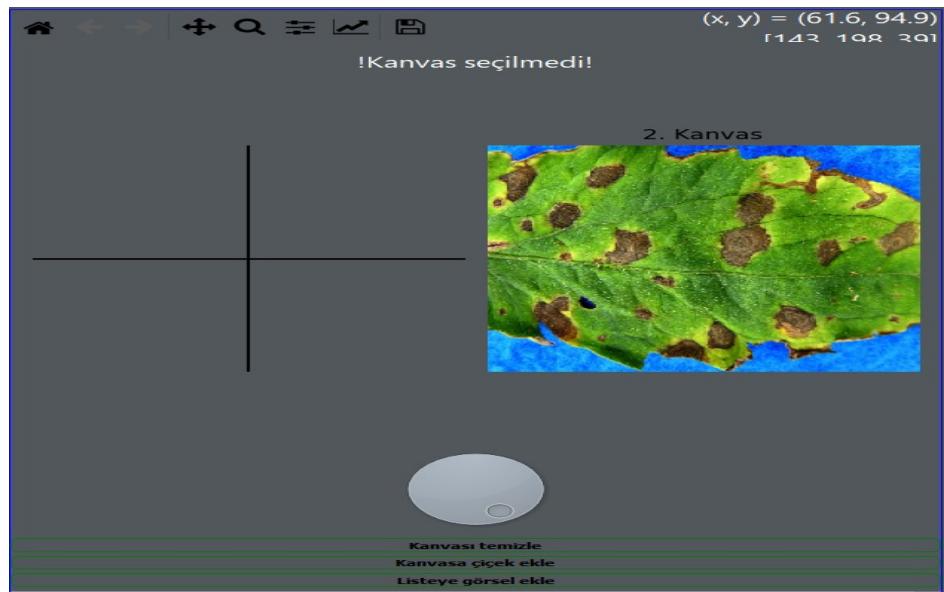
```
{0: 'Tomato_Bacterial_spot',
1: 'Tomato_Early_blight',
2: 'Tomato_Late_blight',
3: 'Tomato_Septoria_leaf_spot'}
```

Artık görmüş olduğunuz üzere sınıflar numaralandırıldı peki şimdi ne yapıcaz?

modelin verdiği çıktıda ilk baştada anlattığım üzere hepsine bir skor biçilir ve en büyük skorun indeksinin bu dictionary(sözlük) teki karşılığı = **Modelin tahmini** olarak anlatmak doğru olur

Bunun nasıl yapıldığını aslında anlatmak gereksiz bir şeydir ancak anlatmaktanda zarar gelmeyeceği için ben anlatma taraftarıyım

İlk olarak programımıza bir hasta çiçek verelim 1. hastalık türümüz **Domates Erken Yanıklığı** olsun yani aşağıdaki görsel;

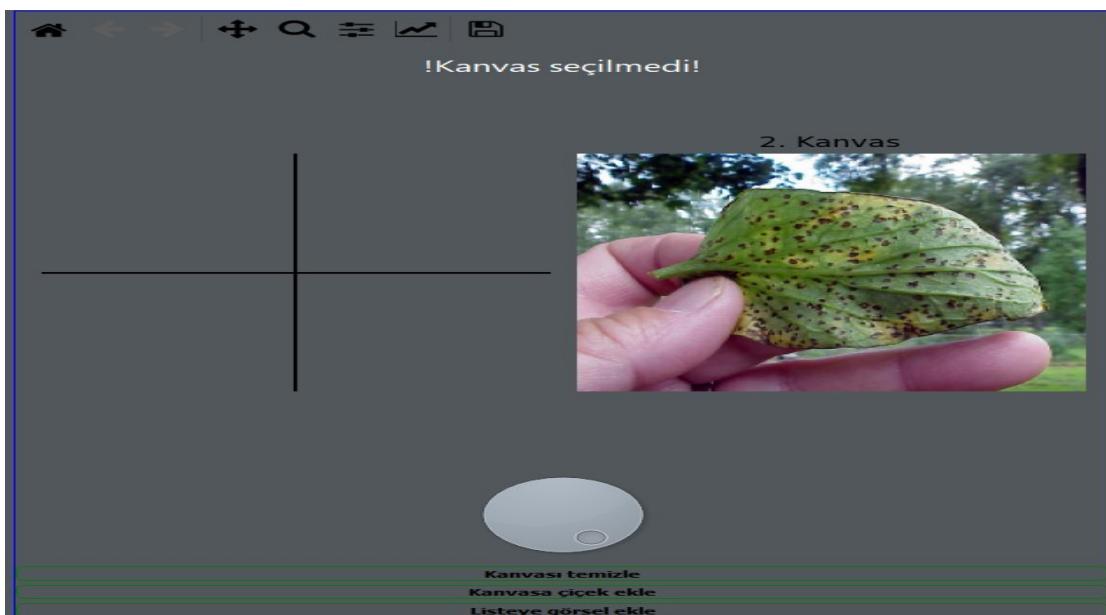


Bu görsele modelimizin tahmini aşağıdaki gibi olmuş;

	Model tahmini	Tahmin	Zaman kodu	Mevcut kanvas	Görsel ismi
1	0.34415853	Domates Erke...	2026-02-07--...	2. Kanvas	6.02.2026 20:24
2					
3					
4					

0.3441 civarı bir olasılık ile Domates erken yanıklığını bulmuş ancak bu olasılık 4 sınıf arasındaki olasılıkları temsil eder;

*Şimdi ise farklı bir hastalık olan **Domates bakteri lekesi** hastalığını programa verelim ve modelimizden tahmin bekleyelim bakalım bu sefer bize ne vericek; görsel aşağıdaki gibidir;*



Ve modelimizin tahmini ise aşağıdaki gibi olmuştur;

	Model tahmini	Tahmin	Zaman kodu	Mevcut kanvas	Görsel ismi
1	0.34415853	Domates Erke...	2026-02-07--...	2. Kanvas	6.02.2026 20:24
2	0.5896346	domates bakt...	2026-02-07--...	2. Kanvas	7.02.2026 03:48
3					
4					

*0.5896 civarında bir olasılık ile 4 olasılık arasından
bu seçilmiş ve cevabımız yine **Domates bakteri
lekesi** olmuş*

***NOT:** Yukarıdaki kanvas seçilemedi uyarısının
sebebi kanvas seçilince **Label'a** sinyal gönderen
sistemin daha kurulmamış olmasıdır yani aslında
kanvas seçilidir.*

*Evet hastalık sınıflandırma modelimizde
tanıttığımıza göre artık yavaş yavaş sona gelelim*

*Bu program geliştirmeye açık bir program
olmakla beraber bu programa yakında mutlaka
gelecek özelliklerden bir kaçını buraya bırakıyorum
veya şu anda bile gelmiş olabilir;*

***Transfer Learning ile modelinizi sürekli
olarak eğitebilmeniz**
Kendi modelinizi oluşturan bir sekme
**Google gömülü görsel aktarım
sistemi(EKLENDİ)**
Kameradan fotoğraf çekip analiz edebilme*

Google gömülü görsel aktarım sistemini biraz açacak olursam;

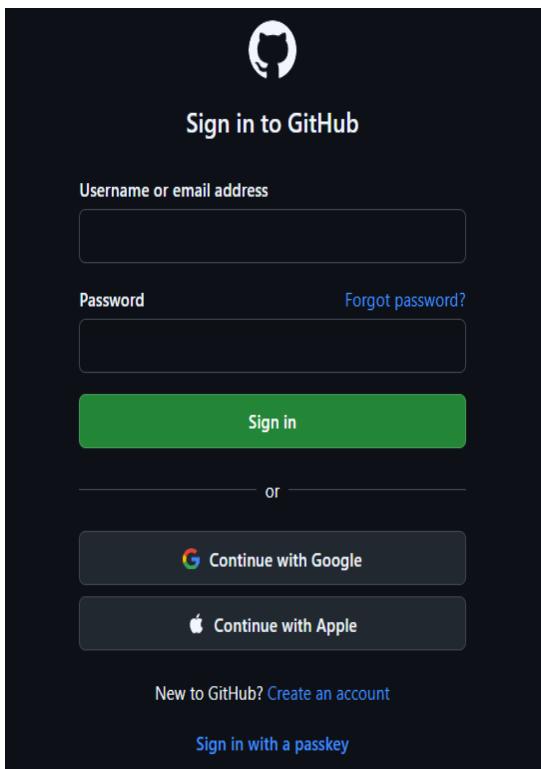
*Bu sistem profesyonel bir **Google Embed** sistem değildir sadece ihtiyaçları karşılaması için geliştirilmiş olan fonksiyonel bir internetten görsel paylaşım / aktarımı için ufak bir sistemdir hatta sistem demek bile çok olur bir kaç satırlık script ve 3 fonksiyondan oluşan bi sistem demek daha doğru olur bu kısımda web siteleri açılamadığı için şunları yapmanız gereklidir;*

*Eğer bir görsel bilgisayarınızda yok ise ve bu görseli aktarmak istiyorsanız en güvenilir çözüm olarak bir **Github** hesabı açmalısınız açığınız **Github** hesabına istediğiniz görseli yükleyip sonrasında görselin olduğu **Repo'yu Public** yapmalısınız sonrasında ise linkini kopyalayıp bu sistemde **Url tanımla** kısmında o linki girip görseli gördükten sonra **Sürükle-Bırak** ile kendi dosya sisteminize almanız önerilir*

Bunu nasıl yapacaksınız?

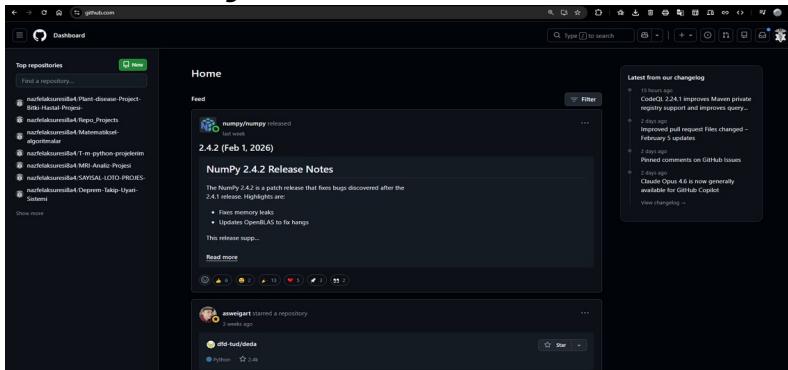
1. Adım Github hesabı oluşturma

*https://github.com/login adresinden bir hesap açın
bu sayfa şöyle görünür*



2. Adım Github hesabınızda repo açmak

Hesap açıldıktan sonra sizi aşağıdaki gibi bir ekran karşılar



Burada ise New seçeneğinden bir repo açınız

3. Adım Repo açmak

Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

1 General

Owner * / Repository name *
Gorsel_yukleme_deposu is available.

Great repository names are short and memorable. How about [supreme-octo-adventure](#)?

Description

0 / 350 characters

2 Configuration

Choose visibility *

Choose who can see and commit to this repository

Add README

READMEs can be used as longer descriptions. [About READMEs](#)

Add .gitignore

.gitignore tells git which files not to track. [About ignoring files](#)

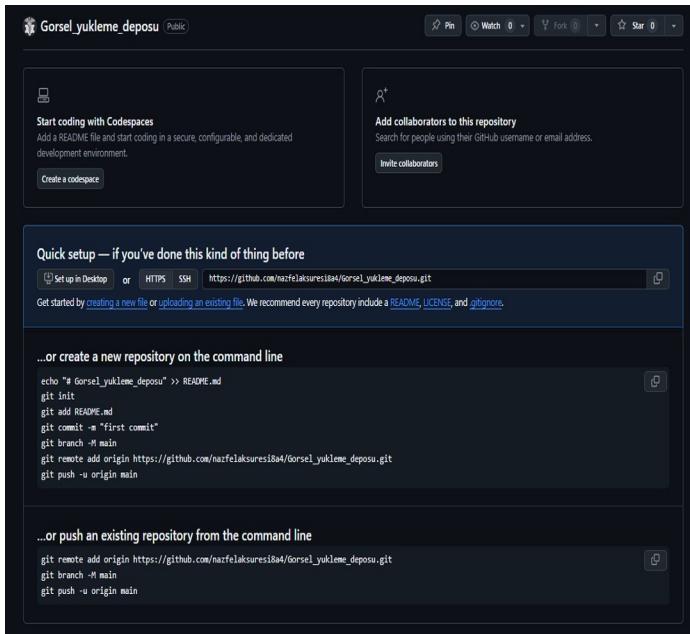
Add license

Licenses explain how others can use your code. [About licenses](#)

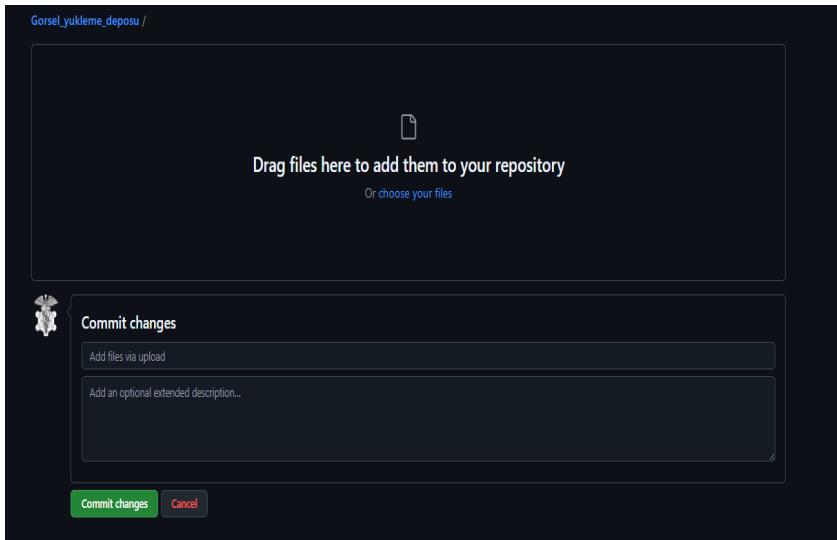
Create repository

*Burada ise ekranada görmüş olduğunuz gibi reponuza isim verin ve en önemlisi benimde yaptığım gibi reponuzu **Public** görünümüne alın ve en sonundada **Create Repository** butonuna basıp reponuzu açın*

4. Adım Repoya ilk görseli eklemek

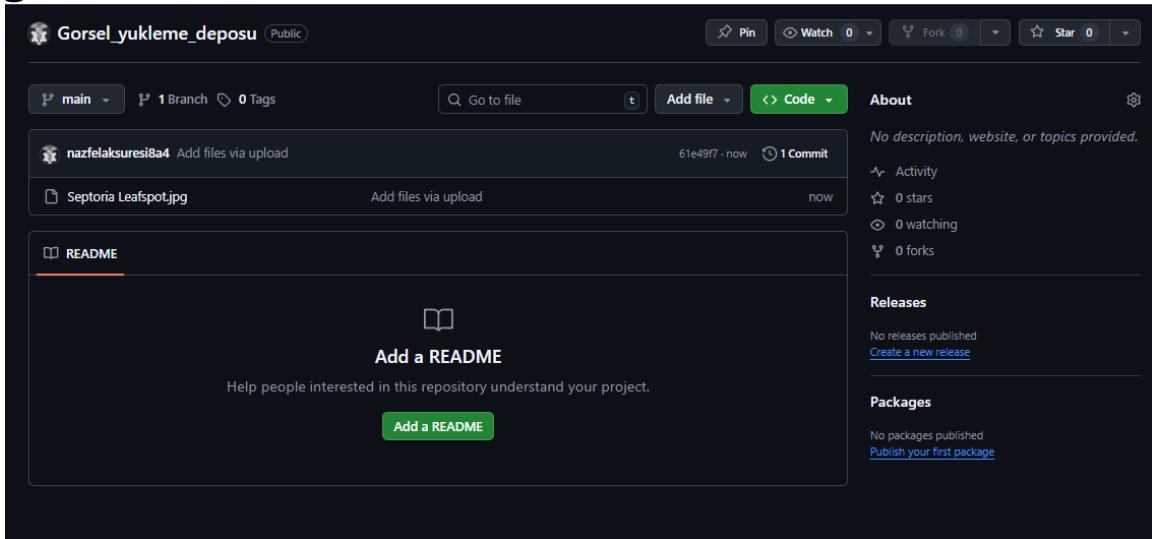


*Ekranda sizi böyle bir alan karşılayacak burada ise ortadaki **uploading an existing file** yazısına tıklayın tıkladığınızda şöyle bir ekran sizi karşılar*

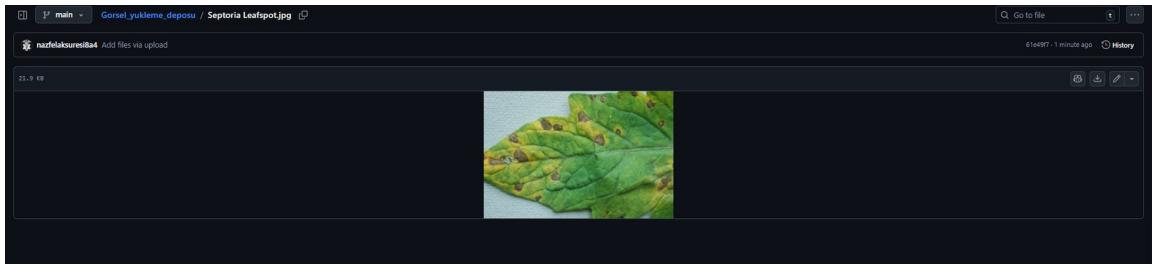


*Buraya istediğiniz görseli **sürükle-bırak yaparak** atın ve **Commit changes** butonuna tıklayın artık reponuzda bir görsel olacak **Görsel yükleme***

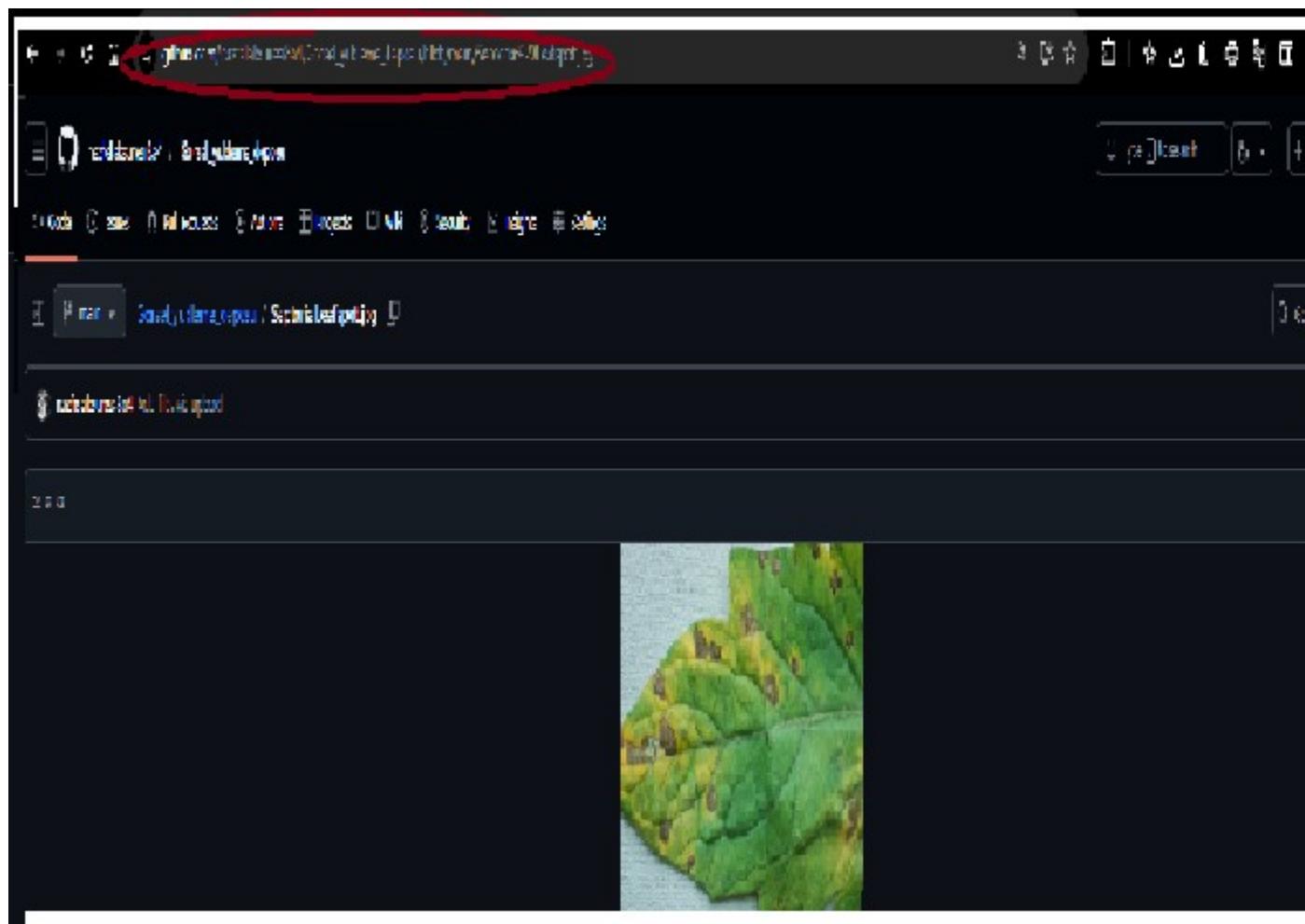
*işleminiz bittiğinden sonra reponuz şu şekilde
görünecektir*



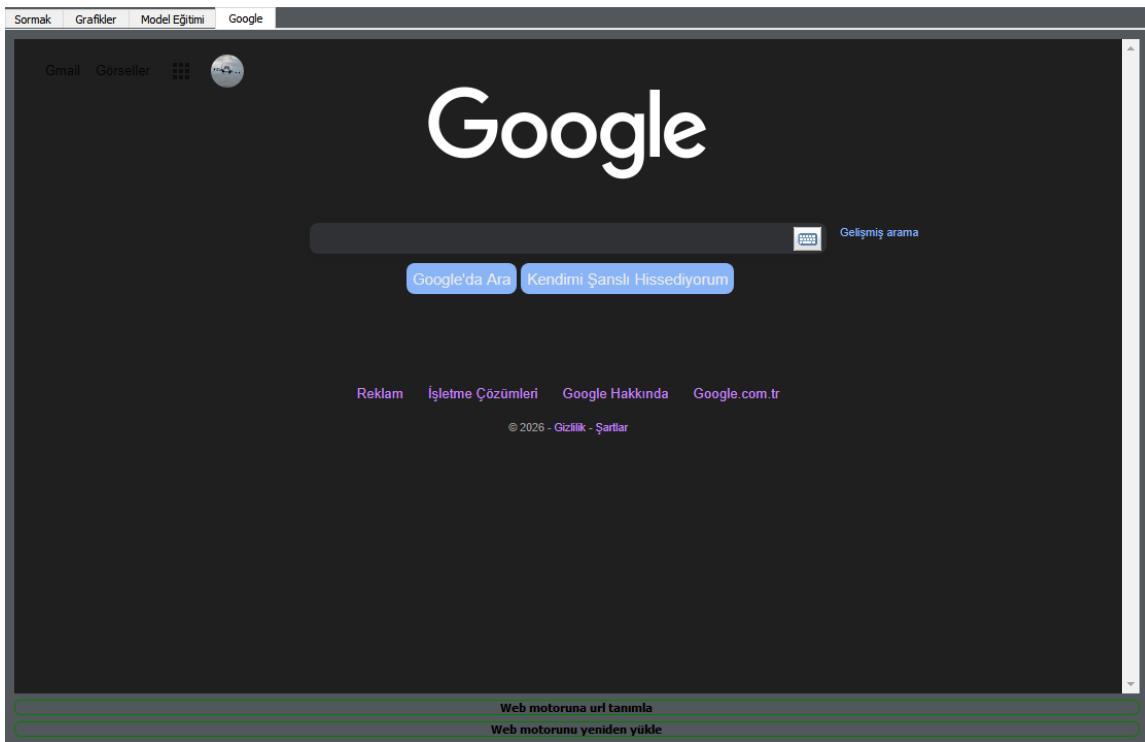
*Burada artık o dosyanın üzerine tıklayın ve şu
ekrana geliceksiniz*



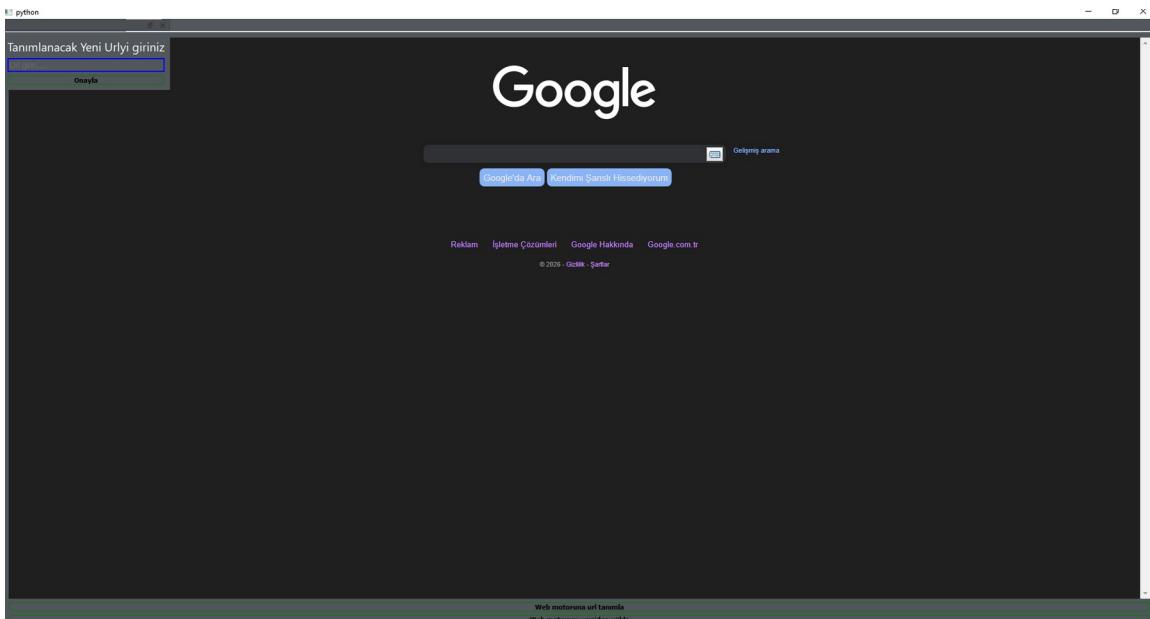
*Göründüğünüz gibi yüklediğiniz görsel ney ise artık
bu ekranda görünür durumda olarak karşımıza
çıkar şimdi ise son adıma geldik*



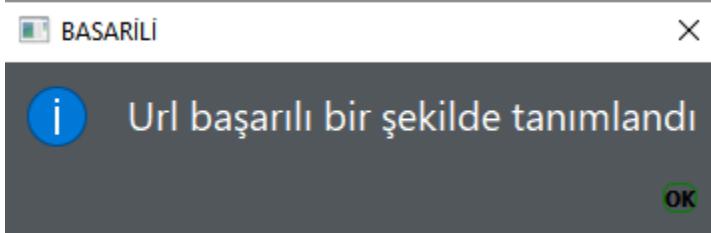
buradaki urlyi kopyalayın ve sonra programa gelin



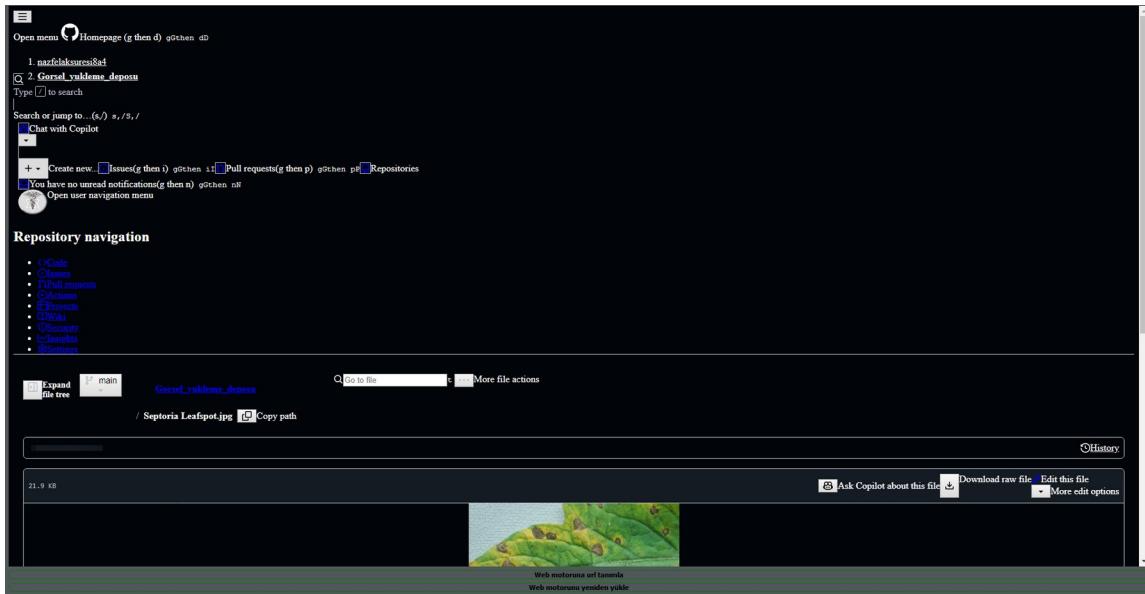
*Programın **Google** kısmına gelin ve burada alt
kısımındaki **Web motoruna url tanımla** butonuna
tıklayın*



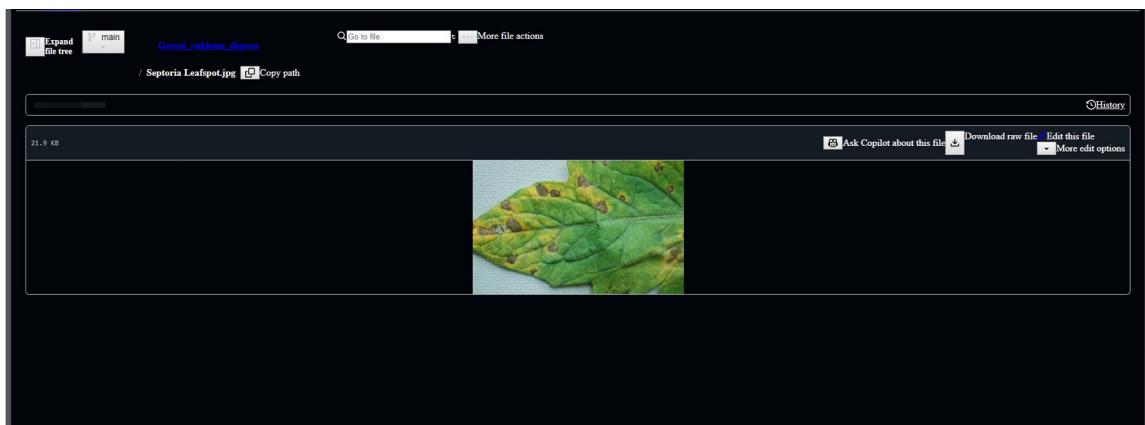
*Ekranınıza böyle bir **pop-up** gelecek buradaki url girme kısmına kopyaladığınız url'i yapıştırınız ve sonra **Onayla** butonuna tıklayın başarılı bir şekilde tanımlanır ise şu uyarı mesajını göreceksiniz*



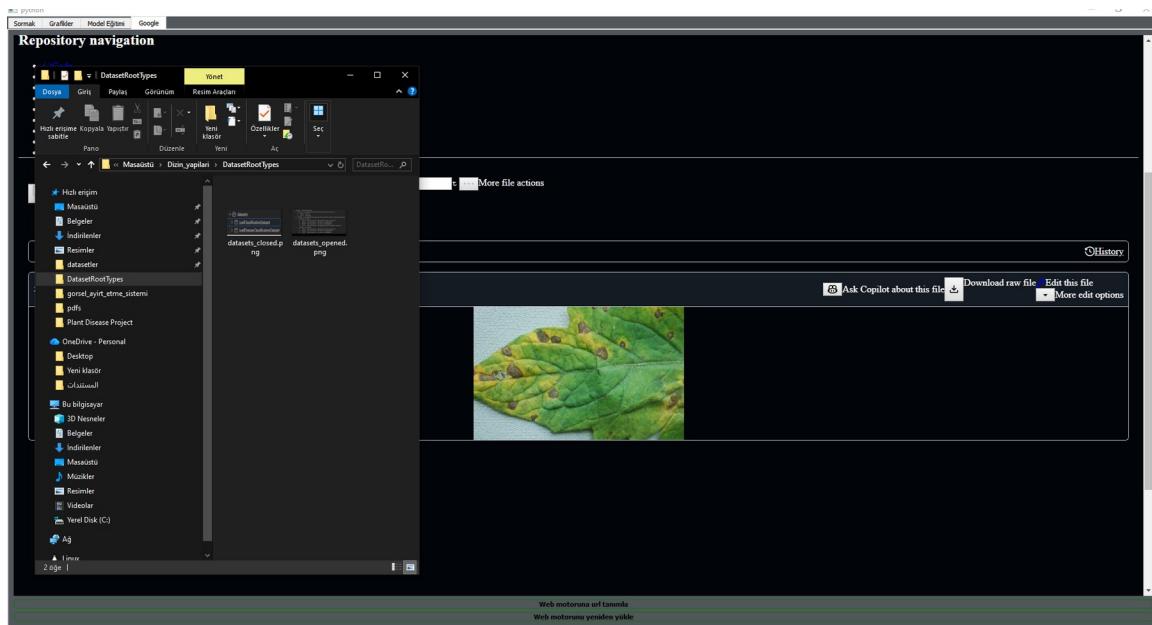
sonrasında şu ekranı görüceksiniz



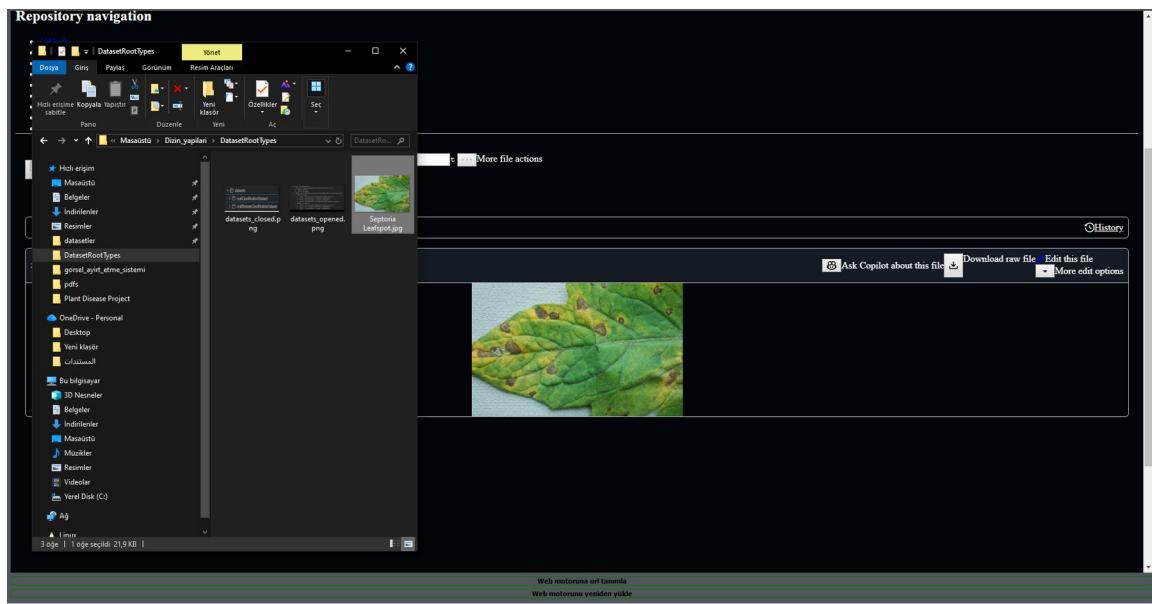
Bu ekranda aşağıya kaydırın ve buraya gelin



Sonrasında dosya sisteminizi açın

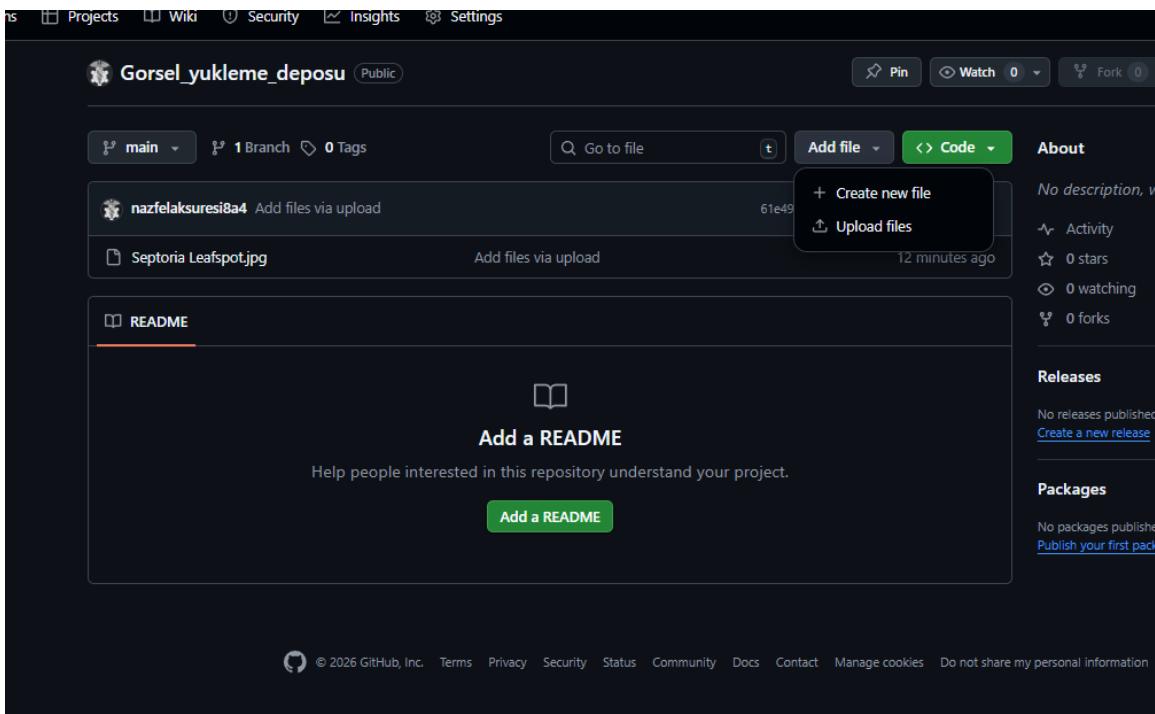


ve burada görselin üzerine basılı tutup dosya sisteminize atın



işte işleminiz bitti artık görsele ulaşabilirsiniz

Ve başka bir görsel eklemek isterseniz ise



*Buradan **Add file** ve sonra **Upload file** seçeneğine tıkmanız sonrasında sürükle bırak ile başta anlattığım gibi görseli yüklemeniz yeterlidir*

***NOT:** Biz githuba bilgisayarımızdan görsel yükledik ancak başka bir cihazdan yükleyseydikte işlem sırası değişmeyecekti*

Son Olarak;

*Beni dinlediğiniz için ve bu program ile
ilgilendığınız için teşekkür ederim Önerileriniz veya
benim gözümden kaçan herhangi bir unsur &
teknik detay var ise bunu bana lütfen bildiriniz*

*github: <https://github.com/nazfelaksuresi8a4>
discord: **kgt_141***

*Teknik bilgilendirme(**Kullanılan
modül/Frameworkler**)*





K Keras

