



Ömer Berke GÖKTÜRK 131180032

Umut Barış KORKUT 141180752

BM 496 BİLGİSAYAR PROJESİ II

FİNAL RAPORU

T.C

GAZİ ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ANKARA

2018

	2
ŞEKİLLERİN LİSTESİ	4
1. ÖZET	6
2. GİRİŞ	7
2.1. Proje Tanımı	7
2.2. Amaç	7
2.3. Kapsam	8
2.4. Literatür Taraması ve İlişkili Çalışmalar	8
3. KULLANILACAK ARAÇ VE YÖNTEMLER	13
3.1. Makine Öğrenmesi	13
3.1.1. KNN	13
3.1.2. Rassal Orman	14
3.1.3. Naive Bayes	14
3.1.4. Destek Vektör Makinesi (Support Vector Machine – SVM)	15
3.1.5. Lojistik Regresyon	16
3.1.6. Derin Öğrenme	16
3.2. Öznetelik Çıkarımı	17
3.2.1. Öğrenme Aktarımı (Inception V3 Modeli)	18
3.3. Kullanılan Araçlar	19
3.3.1. Orange	19
3.3.2. Windows 10 işletim sistemi	19
3.3.3. Python 3.6 programlama dili	19
3.3.4. Scikit – learn kütüphanesi	19
3.3.5. Jupyter – notebook geliştirme ortamı	20
3.3.6. Fatkun batch image downloader veri seti temin aracı	20
3.3.7. Linux işletim sistemi	20
3.3.8. Tensorflow kütüphanesi	21
3.3.9. Android studio bütünleşik geliştirme ortamı	21

	3
4. GERÇEKLEŐTİRİLEN UYGULAMA	23
4.1. Veri Kümesi (Yaprak Görselleri)	23
4.2. Sistem Tasarımı	29
4.3. Yaprak görsellerinin sınıflandırılması	31
4.3.1 Görüntülerin Vektörlere Dönüőtürölmesi	31
4.3.2 Algoritmaların Uygulanması	31
4.4 Sonuçların Karşılaőtırılması	37
4.5 Arayüz Tasarımı	39
5 SONUÇ VE DEĞERLENDİRME	40
6 REFERANSLAR	42

ŞEKİLLERİN LİSTESİ

Şekil 2.1. Türkiye’de yıllık tarımsal ilaç kullanımı

Şekil 2.2. Üzüm yaprağıyla ilgili sınıflandırmalar için yapılan işlemler

Şekil 2.3. Pamuk yaprağındaki hastalıkların tespiti için oluşturulan işlem basamakları[4].

Şekil 2.4. Pamuk yaprağındaki hastalıkların tespiti için uygulanan algoritmalar ve başarı oranları

Şekil 2.5. Önerilen yaklaşım için işlem basamakları[3].

Şekil 3.1. KNN algoritmasında kullanılan fonksiyonlar

Şekil 3.2. Rastsal orman algoritmasının görselleştirilmiş hali

Şekil 3.3. Naive bayes algoritması için fonksiyon

Şekil 3.4. SVM uzay bölmesi örneklendirmesi

Şekil 3.5. Logistic regresyon uzay bölme örneği

Şekil 3.6. Derin öğrenme katmanlarının görselleştirilmiş hali

Şekil 3.7. Öğrenme aktarımı örnek şeması

Şekil 3.8. Tensorflow sonuçları

Şekil 4.1. Orange üzerinde örnek şema

Şekil 4.2. Kriterlere uymayan yaprak görseli.

Şekil 4.3. Kriterlere uymayan yaprak görseli.

Şekil 4.4. Kriterlere uymayan yaprak görseli.

Şekil 4.5. Kriterlere uymayan yaprak görseli.

Şekil 4.6. Toplanan sağlıklı yaprak görsellerinden bazıları

Şekil 4.7. Toplanan bakteriyel lekeli yaprak görsellerinden bazıları

Şekil 4.8. Erken yanık hastalığına yakalanmış bitki görselleri

Şekil 4.9. Kurbağa göz yaprağı görselleri

Şekil 4.10. Mantar hastalıklarına ait yaprak görselleri

Şekil 4.11. Güneş yanığı hastalıklarına ait yaprak görselleri

Şekil 4.12. Büyük veri setindeki bazı görseller

Şekil 4.13. Kullanıcı uygulamayı başlatır.

Şekil 4.14. Kullanıcı, kamerayı yaprak görseline tutar.

Şekil 4.15. Sistem sonuç döndürür.

Şekil 4.16. Kullanıcı çıkış yapar.

Şekil 4.17. Öğrenme aktarımı yöntemiyle özellikleri çıkarılan görsellerin vektör değerleri

Şekil 4.18. Veri setinin eklenmesi

Şekil 4.19. SVM algoritması test sonuçları

Şekil 4.20. KNN algoritması test sonuçları

Şekil 4.21. Rassal Orman algoritması test sonuçları

Şekil 4.22. Yapay sinir ağı algoritması test sonuçları

Şekil 4.23. Lojistik regresyon algoritması test sonuçları

Şekil 4.24. Naive Bayes algoritması test sonuçları

Şekil 4.25. Büyük veri setinin yüklenmesi

Şekil 4.26. Büyük veri setinde rassal orman algoritmasının test sonuçları

Şekil 4.27. Büyük veri setinde SVM algoritmasının test sonuçları

Şekil 4.28. Büyük veri setinde yapay sinir ağı algoritmasının test sonuçları

Şekil 4.29. Büyük veri setinde lojistik regresyon algoritmasının test sonuçları

Şekil 4.30. Büyük veri setinde naive bayes algoritmasının test sonuçları

Şekil 4.31. Uygulama arayüzü ve yaprak görseliyle test

Şekil 5.1. Mobil uygulamanın test işlemi

1. ÖZET

Tarımsal ürünlerinin verim ve kalitesi için ilaçlama (pestisit) önemli bir yer tutmaktadır. Bununla birlikte bitki hastalıklarının doğru ve erken tespit edilmesi ilaçlamanın etkinliğini artıracaktır. Ancak bitki ve hastalık türleri de farklı farklı olabilmektedir. Dolayısıyla bitki ve hastalık türüne göre ilaçların doğru ve yeterli miktarda kullanılması önem arz etmektedir. Tarımsal ürünlerde gereksiz ilaç kullanımı maliyeti arttırdığı kadar toplum sağlığını da olumsuz etkilemektedir. Bitki hastalıklarının tespitinde uzmanlara ve ürün yetiştiricilerine önemli görevler düşmektedir. Ancak ürün yetiştiricilerin gerekli eğitime sahip olmaması ya da yeterli uzman bulunamaması bu alanda yaşanan en büyük sıkıntılardan biridir.

Bitkilerin ayırt edici birçok özelliği vardır. Bu özellikler, hastalık durumlarında da farklı şekillerde, kendilerini göstermektedir. Ancak belirtiler, her zaman açık değildir. Hatta bazı durumlarda gözle görülemeyen farklılıklar, hastalığın çok önemli belirtisi olabilir. Eğer bitkilerde, belirti özelliği taşıyan farklılıklar zamanında tespit edilebilirse, bitkiyi kurtarma ihtimali çok yüksektir.

Teknolojide yaşanan gelişmeler bilgisayar destekli tanı sistemlerinin de etkin bir şekilde kullanılmasına olanak sağlamıştır. Özellikle yapay zeka alanında yaşanan gelişmeler ile bilgisayarlar insanların ayırt edemeyeceği belirtileri net bir şekilde ortaya çıkarabilir.

Gerçekleştirilen bu projede bitki hastalıkları makine öğrenmesi yöntemleri kullanılarak otomatik tespit edilmiştir. Uluslararası konferansta yayınlanan makalede[1] bahsettiğimiz, bir bitkide hastalık olup olmadığını tespit eden iki sınıflı bir sınıflandırıcı sistemi, bu projede geliştirilmiş, hastalığın tam teşhisi yapılmış ve mobil uygulama geliştirilmiştir. Ayrıca geliştirilen uygulamayla, “2242 Tübitak Öncelikli Alanlarda Üniversite Öğrencileri Proje Yarışması” na başvurulmuştur.

Hastalık tespiti için Logistic regression, kNN, random forest, naive bayes, Destek Vektör Makinesi gibi algoritmaları kullanılmış. En yüksek performansı gösteren algoritma için Android tabanlı mobil bir uygulama geliştirilmiştir. Uygulama telefon kamerası ile alınan yaprak görseli üzerinden %97 doğrulukta hastalık tanısı yapabilmektedir.

2. GİRİŞ

2.1. Proje Tanımı

Bitki hastalıkları ile mücadele kalite ve ürün verimliliği için önemlidir. Ayrıca hastalık için uygun ve yeterli ilaç kullanımı hem toplum sağlığı hem de hastalıkla mücadele de önem arz etmektedir. Tedavi edilemeyen bitki hastalıklarından dolayı ciddi kayıplar yaşanmaktadır. Bu kayıplar bazen yanlış tespitlerden bazen de tespit edilememekten kaynaklanmaktadır. Yapılan yanlış tespitler sonucu, kullanılan yanlış ilaçların oluşturduğu maliyetler ve tedavi edilemediğinden dolayı kaybedilen bitkilerin oluşturduğu maliyetler, tarım sektörünü olumsuz etkilemektedir.

Tarımsal ilaç kullanımı Pesticides use							(Ton - Tonnes)
	İnsektisitler Insecticides	Fungusitler Fungicides	Herbisitler Herbicides	Akarisitler Acaricides	Rodentisitler Rodenticides	Diğer Other	Toplam Total
2006	7 628	19 900	6 956	902	3	9 987	45 376
2007	21 046	16 707	6 669	966	51	3 277	48 716
2008	9 251	16 707	6 177	737	351	5 613	38 836
2009	9 914	17 863	5 961	1 533	78	2 302	37 651
2010	7 176	17 396	7 452	1 040	147	5 344	38 555
2011	6 120	17 546	7 407	1 062	421	6 978	39 534
2012	7 264	18 124	7 351	859	247	8 766	42 611
2013	7 741	16 248	7 336	858	129	7 128	39 440
2014	7 586	16 674	7 794	1 513	149	6 007	39 723
2015	8 117	15 984	7 825	1 576	197	5 327	39 026
2016	10 425	20 485	10 025	2 025	259	6 835	50 054

Kaynak: Gıda, Tarım ve Hayvancılık Bakanlığı
Tablodaki rakamlar, yuvarlamadan dolayı toplamı vermeyebilir.

Source: Ministry of Food, Agriculture and Livestock
Figures in table may not add up to totals due to rounding.

Şekil 2.1. Türkiye’de yıllık tarımsal ilaç kullanımı[2]

Şekil 2.1’de görüldüğü gibi ilaç tüketimlerinin büyük bir çoğunluğu gereksiz tüketimi oluşturmaktadır. Bunun bir numaralı sebebi, bitki hastalıklarını tespit sistemlerinin yetersizliğidir.

Gerçekleştirilen bu projede bitki hastalıklarının otomatik tespitini gerçekleştiren Android tabanlı mobil bir uygulama geliştirmiştir. Hastalık tespiti makine öğrenmesi yöntemleri kullanılarak gerçekleştirilmiştir.

2.2. Amaç

Tarım alanının en önemli sorunu, bitki hastalıklarının sebep olduğu kayıplardır. Hastalıklar, verimliliği ciddi oranda düşürmektedir. Bunun sonucunda, ekonomik kayıplar oluşmaktadır. Bu nedenle hastalıklarla baş etmek önemli bir konudur.

Bitki hastalıklarıyla baş edebilmek için, hastalığın ne olduğu doğru bir şekilde tespit edilmelidir. Ayrıca tespit işlemi zamanında yapılmalıdır.

Bitki hastalıklarının erken tespit edilmesi, böylece tedavinin başarılı olması ve bütün bunların sonucunda ürün kalitesinin düşmesini önleyip verimliliği artırmak için yaprak görsellerinden hastalık tespiti yapan Android tabanlı mobil bir uygulama sunulmuştur.

2.3. Kapsam

Bitki hastalıkları tespit edilirken, bitkilerde karşılaştırılacak birçok farklı özellik bulunmaktadır. Hasta olan ve sağlıklı bitkiler arasında, bazı özel farklılıklar görülmektedir. Hastalık tespiti için; bitkilerin yaprak şekilleri, kök şekilleri, gövde, çeşitli kısımların renkleri gibi özellikler incelenerek, farklılıklar tespit edilir. Bu yöntemi yürütecek uzman kişi, gözle algılayabileceği farklılıkları kolayca tespit edebildiği durumda, hastalık için net bir ifadeyle bulunabilir. Ancak duyu organlarıyla algılanamayacak farkları, insanların tespit etmesi mümkün değildir. Hatta farklı bir koku yaymayan ve görüntü olarak farklı algılanmayan bitkilerde hastalık tespiti mümkün olmamaktadır. Olumsuz bir durumdan şüphelenilemeyecek bu gibi anlarda, bitki ölümleri gerçekleşmeden, sorun algılamak imkânsızdır.

Sensörler, görüntü işleme cihazları, boyut ölçüm yöntemleri gibi araçlar, insanın fark edemeyeceği hususları, kolaylıkla tespit edebilir. Bahsedilen, insanların ölçüm yapamayacağı durumlar için, bu tarz sistemler geliştirilebilir. Ancak ölçüm sonuçlarının insanlar tarafından yorumlanması, yine bir takım hatalar doğuracaktır. Ölçüm ve değerlendirme aşaması bir bütün olarak ele alınıp, bilgisayar sistemleri aracılığıyla gerçekleştirilirse, tespit ve çözüm işlemlerinin başarı yüzdesi artırılabilir. Ölçüm için kullanılacak mekanik veya elektronik sistemlere yorum mekanizması eklemek, makine öğrenmesi alanının sınırları içindedir. Yani makine öğrenmesi, bitkilerle ilgili çeşitli ölçüm değerlerini karşılaştırıp, problemin türüne göre – hastalıklı veya hastalısız yorum yapabilen bir daldır.

2.4. Literatür Taraması ve İlişkili Çalışmalar

Makine öğrenmesi yöntemleriyle; bitki türlerinin, hastalıklarının tespit edilmesiyle ilgili daha önce yapılan bir takım çalışmalar vardır [8,9,10].

Dünyada bulunan birçok üzüm türü vardır. Bu üzüm türlerinin yaprakları incelenerek, üzüm türlerini tespit etmeye yönelik bir çalışma yapılmıştır. Bu tespit işleminde yapay sinir ağıları kullanılmıştır[3].

Hindistan'da üzüm yetiştiriciliği, Hindistan tarım faaliyetlerinin önemli bir kısmını oluşturmaktadır. Avrupa'dan ve Batı Asya'dan gelen üzümün yetiştiriciliği için, üzümün yetiştirme şartlarının bilinmesi gerekir. Üzümün meyve verebilecek elverişlilikte yetiştirilmesi için, sıcak ve kuru hava gereklidir. Üzüm sağlığını etkileyen en önemli faktörlerden ikisi; toz ve küftür. Onun dışında bitki hastalıkları da üzüm yetiştiriciliğini ciddi oranda etkilemektedir. Sağlıklı üzüm yetiştirebilmek için; elverişli hava koşullarının sağlanması, toz ve küfün uzaklaştırılması, bitkinin doğru zamanda budanması ve bitki hastalıklarının zamanında tespit edilmesi son derece önemlidir[4].

Hindistan'daki üzüm yetiştiriciliği için yapılan literatür çalışmasında, sağlıklı koşulların düzeltilmesi için kullanılabilecek teknolojik sistemler; desen tanıma, resim işleme şeklinde belirlenmiştir. Bu tarz teknolojik bir sistemden beklenti; zararlı otların tespit edilmesi, meyve ve sebzelerin ayırt edilmesi ve bitki hastalıklarının tespit edilmesidir.

Bitki hastalıklarının otomatik olarak saptanması, önemli bir araştırma konusudur. Böyle bir çalışma başarıyla yürütüldüğü durumda, oluşturulan bitki tarlalarının verimliliği artabilir ve bitki yapraklarında belirtiler olduğu anda, bitki hastalıkları algılanabilir. Bu nedenle, bitki hastalık vakaları, hızlı ve doğru bir şekilde zamanında tespit edilebilmelidir. Daha önce bahsedildiği üzere, yanlış tespitler kadar zamanında yapılamayan tespitler de verimlilik açısından sıkıntılar doğurmaktadır. Bu durum, en büyük sorunlardan biri olan, bitki ilaçlarının gereksiz kullanımını beraberinde getirmektedir. Bitki hastalıkları için kullanılan böcek ilaçları, eğer gereğinden fazla kullanılırsa hem maliyeti artırır, hem de toksik kalıntılar, sağlık açısından sorunlar yaratır. Bu nedenle, bitki hastalıklarının tespiti için, düzgün ve etkili modeller geliştirmek gerekir.

Al-Bashish, D., M. Braik ve S. Bani-Ahmad, yapraktaki hastalıkların tespiti ve sınıflandırılması için K-means algoritmasını önermişlerdir. Aynı ekip, yaprak doku özelliklerinin sınıflandırılması için CCM' yi, hastalıkları 6 kategoride sınıflandırmak için de BPNN'yi önermiştir[3].

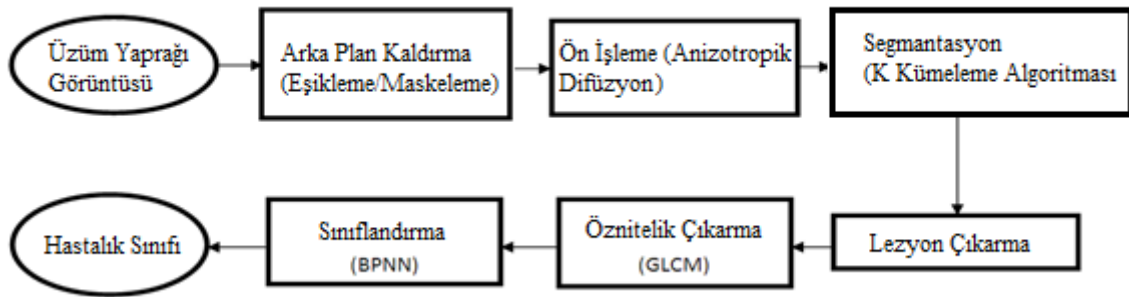
A. Camargo, J.S. Smith, bitkilerin ya da bitkinin yapraklarının RGB renklerini; H, I3a ve I3b' ye dönüştürmeyi tartışmışlardır. Dönüştürülen görüntü, daha sonra histogram kullanılarak analiz edilen yoğunluk dağılımlarına göre bölünür. Çıkarılan bölge; hedef bölgenin bir parçası olmayan piksel bölgesi kaldırılarak işlenir. Daha sonra her düğümün ve pikselin bütün komşuları incelenir[18].

S. S. Sannakki, V. S. Rajpurohit, V. B. Nargund ve diğer ekip üyeleri, hastalık tespiti ve derecelendirmesi için, K-means kümeleme algoritmasını önermişlerdir [4].

H. Al-Hiary, S. Bani-Ahmad, M. Reyalat, özellik çıkarımından önce yeşil piksellerin maskelenmesini önermiş ve daha doğru bir sınıflandırma elde etmişlerdir [13].

Bütün bu araştırmacılar, yaprak örneklerine ait görüntüleri; sabit bir mesafeden, düzgün bir arka plana sahip olan ve güzel ışıklandırılmalı olarak çekilmiş resimlerden toplamışlardır.

A. Meunkaewjinda, P. Kumsawat ve ekibi; üzüm yaprağı rengini tanımak için SOFM ve BPNN'yi, segmentasyon için MSOFM'yi, sınıflandırma için GA ve SVM algoritmalarını ve karmaşık birçok hesap için başka algoritmaları kullanmayı önermiştir [17].



Şekil 2.2. Üzüm yaprağıyla ilgili sınıflandırmalar için yapılan işlemler[4].

Yine Hindistan'da yapılan bir başka çalışma, pamuk yetiştiriciliğinde verimi artırmaya yöneliktir. Hindistan bir tarım ülkesidir ve nüfusunun büyük çoğunluğu tarımla geçimini sağlamaktadır. Bu nedenle tarım verimliliği, ülke ekonomisi için son derece önemlidir. Ancak verimliliği artırmaya yönelik hamleler, teknik bilgi gerektirmektedir. Bu nedenle Hindistan tarımı için, teknolojik bilgisayar sistemlerinin etkin kullanılabilmesi önemlidir. Beyaz altın olarak nitelendirilen pamuk, Hindistan'da 60 milyon insana geçim kaynağı oluşturmaktadır. Bu açılardan ele alınan pamuk, verimliliğin artırılması gereken önemli sorunlardan birini oluşturmaktadır[6].

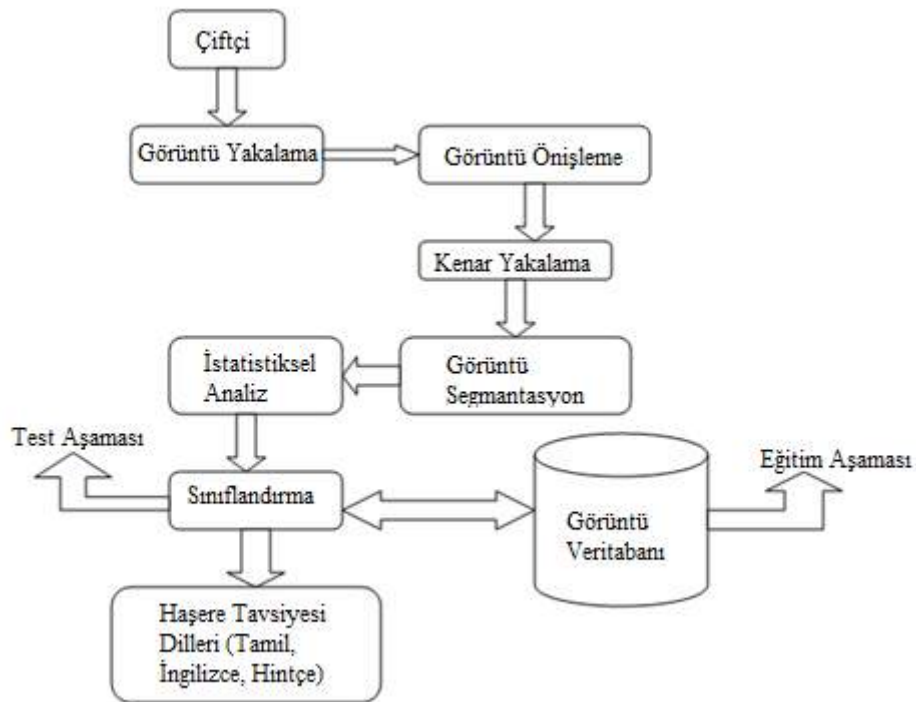
Yapılan çalışmada, pamuk yetiştiriciliği için tarım alanında görüntü işleme yöntemlerinin kullanılması düşünülmüştür. Görüntü RGB özelliği ile piksel sayma tekniği, tarım alanında yoğun bir şekilde uygulanmaktadır. Görüntü işlemenin tarım alanındaki bir takım amaçları şu şekildedir:

- 1- Hastalıklı kök, yaprak, meyve tespiti

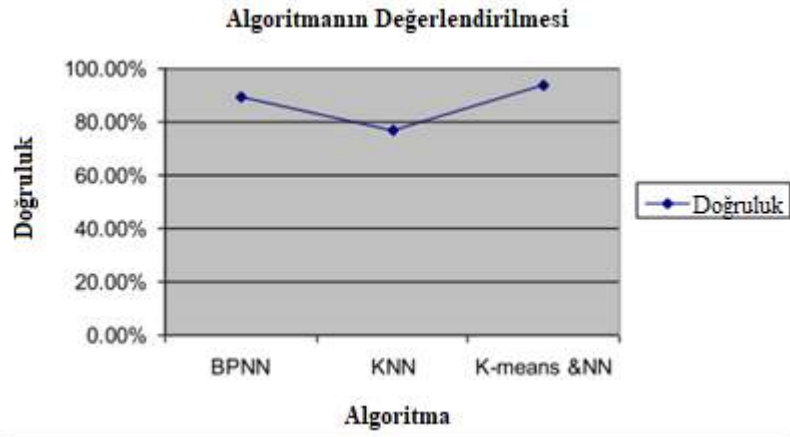
- 2- Etkilenen bölgenin hastalığa göre ölçülmesi
- 3- Etkilenen bölgenin sınırlarının tespiti
- 4- Etkilenen bölgenin renginin belirlenmesi
- 5- Meyvelerin boyut ve şeklinin belirlenmesi
- 6- Nesnenin doğru tespiti

Literatür taramasında bulunan, bu amaçlara yönelik geliştirilmiş olan çalışmada, görüntüler dijital kamera tarafından elde edilir, işlenir ve daha sonra algoritmalar uygulanarak başarı oranı tespit edilir[6].

Daha önce yapılmış olan bu çalışmada, bilgisayar sistemleri tarafından tespit edilmesi beklenen hastalıklar; Fusarium solgunluğu, Verticillium solgunluğu, kök çürüklüğü, kemik çürüklüğü, gri küf, mor lekeler, yaprak dökme, bakteriyel bulantı, yaprak kıvrıkcılığıdır.

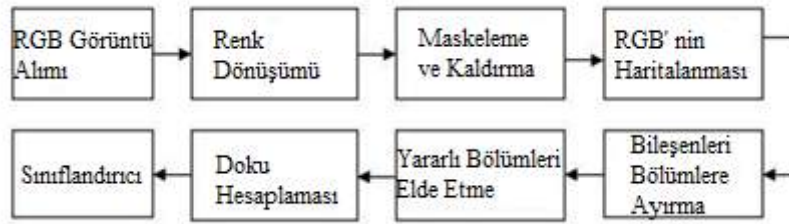


Şekil 2.3. Pamuk yaprağındaki hastalıkların tespiti için oluşturulan işlem basamakları[6].



Şekil 2.4. Pamuk yaprağındaki hastalıkların tespiti için uygulanan algoritmalar ve başarı oranları[6]

Bitki yapraklarının sağlıklı bölgelerinin saptanması ve hastalıkların doku özelliklerine göre sınıflandırılması üzerine yapılan bir başka çalışmada, dijital kamera vasıtasıyla çeşitli yaprak görüntüleri toplanmıştır. Toplanan resimlere algoritmalar uygulamak için öncelikle, tespit için yararlı olabilecek yaprak özellikleri, görüntü işleme teknikleriyle belirlenmiştir[5].



Şekil 2.5. Önerilen yaklaşım için işlem basamakları[5].

3. KULLANILACAK ARAÇ VE YÖNTEMLER

Raporun bu bölümünde hangi araçların ve yöntemlerin kullanıldığı sebebiyle birlikte açıklanıp kullanılan araç ve yöntemler hakkında kısaca bilgiler verilmiştir.

3.1. Makine Öğrenmesi

Görsel sınıflandırma için yöntem olarak makine öğrenmesinin dalları olan KNN (K Nearest Neighbours, En yakın K komşu), Random Forest (Rastsal Orman), Naive Bayes, SVM (Support Vector Machine, Destek Vektör Makinesi), Derin Öğrenme (Deep Learning) ve Lineer Regresyon gibi bir çok yöntem uyguladık. Derin öğrenme yöntemi sınıflandırmanın doğruluğu konusunda en başarılı yöntem oldu fakat daha çok bilgisayar gücü gerektirmektedir, lineer regresyon ise daha az bilgisayar gücü gerektirmekteyken sınıflandırma doğruluğu konusunda daha az başarılıdır.

Android uygulaması için makine öğrenmesi olarak yapay sinir ağları ile derin öğrenme kullandık.

3.1.1. KNN

KNN algoritması her veriyi uzayda bir nokta olarak düşünür ve her özelliği (feature) uzayın bir boyutu olarak düşünür. Verilerin birbirlerine uzaklığını çeşitli uzaklık fonksiyonları ile hesaplayarak sınıflandırma yapar.

Devamlı (continious) verilerde kullanılan üç temel uzaklık fonksiyonu vardır. Bunlardan en yaygın olarak kullanılanları Öklid ve Manhattan uzaklık fonksiyonlarıdır. Devamlı olmayan yani kategorisel verilerde ise Hamming uzaklık fonksiyonu kullanılır.

UZAKLIK FONKSİYONU

ÖKLİD	$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$
MANHATTAN	$\sum_{i=1}^k x_i - y_i $
MİNKOVSKI	$\left(\sum_{i=1}^k (x_i - y_i)^q \right)^{1/q}$

Hamming Uzaklığı

$$D_H = \sum_{i=1}^k |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

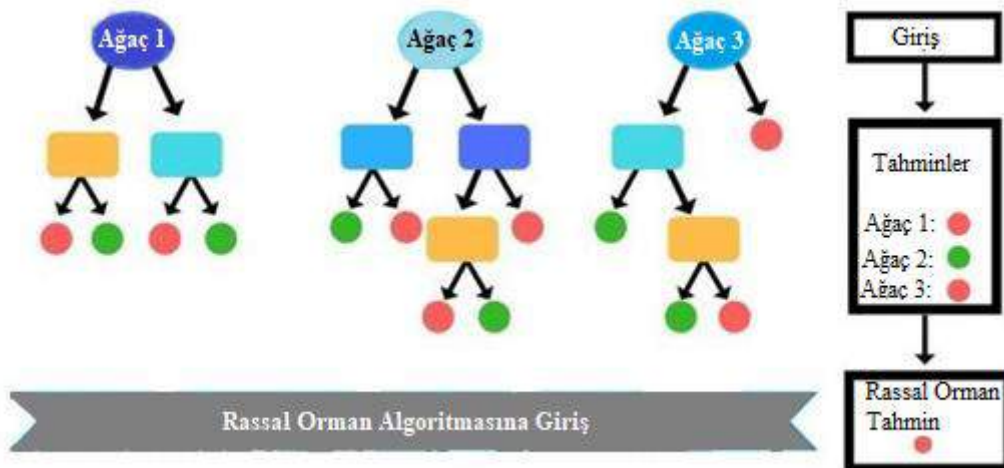
$$x \neq y \Rightarrow D = 1$$

Şekil 3.1. KNN algoritmasında kullanılan fonksiyonlar

3.1.2. Rassal Orman

Rassal orman (Random Forrest) bir diğer deyişle Rastsal Karar Ormanları (Random Decision Forests) makine öğrenimi modelinin eğitimi sırasında verilerin özelliklerine bakarak çok sayıda karar ağacı oluşturur. Karar ağaçlarının ortalamasını alarak verinin hangi sınıfa ait olduğuna belirler.

Karar ağacı derken, bir ağaç yapısı oluşturularak ağacın yaprakları seviyesinde sınıf etiketleri ve bu yapraklara giden ve başlangıçtan çıkan kollar ile de özellikler üzerindeki işlemler ifade edilmektedir.



Şekil 3.2. Rastsal orman algoritmasının görselleştirilmiş hali

3.1.3. Naive Bayes

Naive Bayes sınıflandırma yöntemi, özellikler (features) arasındaki bağıntılara bakarak Bayes Teoreminin uygulanmasına dayanır. Bayes teoremi koşullu olasılık hesaplaması yapar. Naive Bayes yönteminde girilen verinin olasılığı en yüksek olan sınıfa ait olduğu tahmin

edilir. Veri setinin boyutunun çok olduğu durumlarda etkilidir.

$$P(c | x) = \frac{P(x | c)P(c)}{P(x)}$$

Olasılık
Önceki Sınıf Olasılığı

Sonraki Olasılık
Tahmincinin Önceki Olasılığı

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \cdots \times P(x_n | c) \times P(c)$$

Şekil 3.3. Naive bayes algoritması için fonksiyon

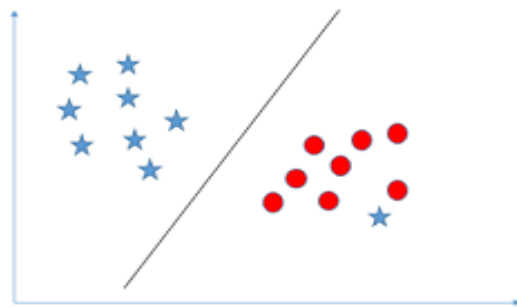
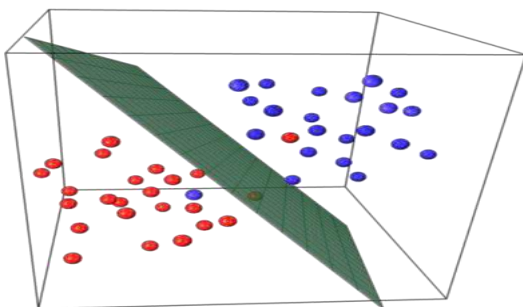
Şekil 3.3. de görülen teorem, bir rastsal değişken için koşullu olasılıklar ile önsel(marjinal) olasılıklar arasındaki ilişkiyi göstermektedir.

$P(c|x)$: B olayı gerçekleştiği durumda x olayının meydana gelme olasılığı $P(x|c)$; A olayı gerçekleştiği durumda x olayının meydana gelme olasılığı $P(c)$ ve $P(x)$: c ve x olaylarının önsel olasılıklarıdır.

3.1.4. Destek Vektör Makinesi (Support Vector Machine – SVM)

SVM (Support Vector Machine) sınıflandırma için uzayda noktalar haline düşünülen verileri ikiye bölerek ayırır. Uzayın her boyutu bir özellik olarak düşünülür. (İki özellik için düzlem doğru ile ayrılır, üç özellik için uzay düzlem ile ayrılır vb.) Uzayı ikiye bölerken farklı sınıfların birbirine olan en uzak noktaları esas alınır.

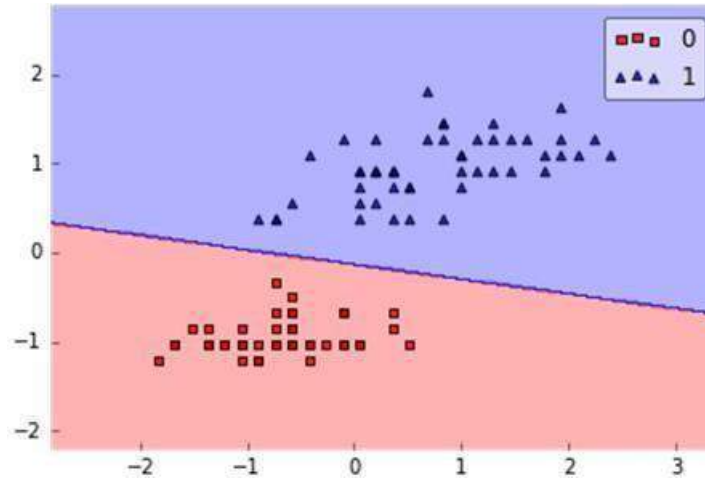
Örneğin en uzak doğrunun bulunabilmesi için yapılan yöntem şöyledir: iki gruba da yakın ve birbirine paralel iki sınır çizgisi çizilir ve bu çizgilerin aralarındaki orta noktalardan geçen doğru ayırım doğrusu olarak bulunur.



Şekil 3.4. SVM uzay bölmesi örneklendirmesi

3.1.5. Lojistik Regresyon

Logistic regression, SVM gibi lineer bir metottur. Sınıflandırma için uzayda noktalar haline düşünülen verileri ikiye bölerek ayırır. Uzayın her boyutu bir özellik olarak düşünülür (İki özellik için düzlem doğru ile ayrılır, üç özellik için uzay düzlem ile ayrılır vb.). Uzayı ikiye bölerken her sınıfa ait olma olasılığını hesaplar.



Şekil 3.5. Logistic regresyon uzay bölme örneği

3.1.6. Derin Öğrenme

Derin Öğrenme, verilerin işlenmesinde insan beyninin işleyişini taklit eden ve karar vermede kullanılacak modeller yaratan bir yapay zeka yöntemidir. Derin öğrenme, yapılandırılmamış veya etiketlenmemiş verilerden denetimsiz öğrenebilen ağlara sahiptir. Yapay Zeka'nın ve makine öğreniminin bir alt kümesidir.

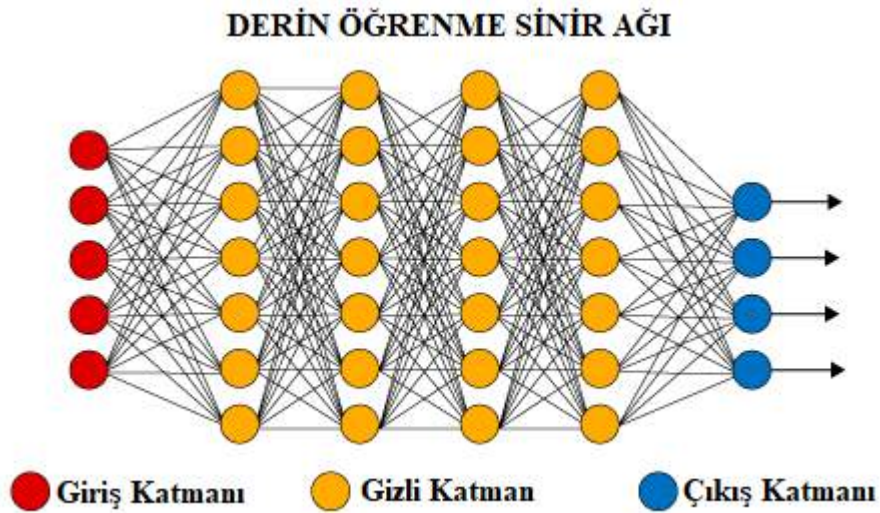
Derin öğrenme, bilgisayarların deneyimlerden öğrenmelerini ve dünyayı kavramların hiyerarşisi olarak anlamalarını sağlayan bir makine öğrenimidir. Bilgisayar deneyimlerden bilgi topladığından, bilgisayarın ihtiyaç duyduğu etiketleme gibi bilgileri sağlaması için bir insana ihtiyaç yoktur. Kavramların hiyerarşisi, bilgisayarın daha basit olanlardan daha karmaşık yapıları öğrenmesini sağlar; Bu hiyerarşilerden oluşan bir grafik, çok sayıda katmandan oluşur.

Derin Öğrenme, dünyanın her bölgesinden çok farklı çeşitlerde oluşan bir veri patlamasına

yol açan dijital çağ sonucunda gelişmiştir. Büyük Veri olarak bilinen bu veri, sosyal medya, internet arama motorları, e-ticaret platformları, çevrim içi sinemalar ve benzeri kaynaklardan alınır. Bu muazzam miktarda veri kolayca erişilebilir ve bulut bilişim gibi teknolojiler ile paylaşılabilir. Bununla birlikte, normalde yapılandırılmamış olan veriler, insanların bunu anlamaları, ilgili bilgileri çıkarmaları ve etiketlemeleri için on yıllar alabileceği kadar geniştir. Derin öğrenme gibi yöntemler, bu büyük veriyi kullanılabilir ve anlamlı bilgiye dönüştürmek için kullanılırlar.

Derin öğrenme algoritmaları çok sayıda veri ile çok başarılı sonuçlar verirler.

Şekil 3.6' da derin öğrenme algoritmasının çalışma mantığı, katmanlar üzerinden gösterilmiştir. Giriş katmanından sonra, birçok gizli katman birbiriyle iletişime geçmektedir. Bu bağlantılar sonucu, çıkış katmanına ulaşılmaktadır.



Şekil 3.6. Derin öğrenme katmanlarının görselleştirilmiş hali

3.2. Öznitelik Çıkarımı

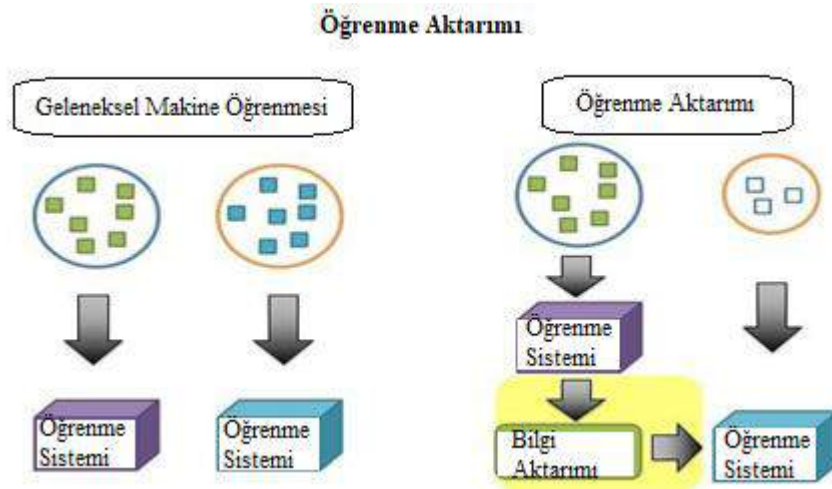
Görsellerden öznitelik çıkarımı (feature extraction) yöntemi olarak öğrenme aktarımı (transfer learning) kullanmayı tercih ettik. Öğrenme aktarımı daha önceden eğitilmiş bir yapay sinir ağının (neural network) çıkarttığı özellikleri kullanma yöntemine verilen addır. Önceden eğitilmiş model olarak VGG16, VGG19, ResNet50, Inception V3, Xception gibi modelleri araştırdık. Google'ın sunduğu Inception v3 modelini ve kullanmaya karar verdik. Bu modeli kullanmaya karar vermemizin sebebi olarak Inception v3 modelinin daha yüksek sınıflandırma doğruluk başarısına sahip olması söylenebilir.

3.2.1. Öğrenme Aktarımı (Inception V3 Modeli)

Görsellerden manuel olarak kullanışlı öznelik (features) çıkarmak çok zordur. Örneğin bir yüzeyin dokusunun (pürüzlü, sert vb.) nasıl olduğunu kod yazarak ayırt etmek çok zordur. Bu yöntemi manuel olarak yapmaktansa derin öğrenme kullanmak çok kullanışlıdır çünkü derin öğrenme ham verilerden özellik çıkarımı yapabilen bir yöntemdir.

Öğrenim aktarımı, bir tür problemde eğitim sırasında kazanılan bilginin, benzer bir problem türünde eğitim için kullanıldığı bir makine öğrenme tekniğidir. Derin öğrenmede, ilk katmanlar problemin özelliklerini belirlemek için kullanılır. İlk katmanlarda elde edilen bilgiler farklı görsellerden öznelik çıkarımı yapmak için de kullanılabilir. Bu, inceptionv3 modelinin daha önceden öğrenmiş olduğu bazı parametreleri yeniden kullanmamıza izin verir, böylece çok daha az eğitim verisiyle yeni bir yüksek sınıflandırma başarısına sahip sınıflandırıcı oluşturabiliriz. Öğrenim aktarımı sırasında, önceden eğitilmiş ağıın son birkaç katmanı kaldırılabilir ve yeni iş için yeni katmanlarla yeniden eğitilebilir.

Şekil 3.7’de önceden eğitilmiş bir sistemin elde ettiği bilgilerden yararlanarak, daha az veri ile nasıl yeni sistem oluşturulabileceğini şematik olarak göstermektedir.



Şekil 3.7. Öğrenme aktarımı örnek şeması

Projenin birinci aşamasında öğrenme aktarımı gerçekleştirmek için arka planda scikit-learn kütüphanesini kullanan Orange3 aracı kullanıldı. Orange3 aracını kullanarak yüklenen görselleri inceptionv3 modeline tabi tutulup, görsellerden elde edilen 2048 adet öznelik, vektör halinde csv formatında kaydedildi.

Projenin ikinci kısmında öğrenme aktarımı gerçekleştirmek için tensorflow kütüphanesi kullanıldı. Tensorflow kütüphanesi kullanılarak yüklenen görselleri inceptionv3 modeline

tabi tutulup, görsellerden elde edilen 2048 adet öznitelik, vektör halinde oluşturuldu.

3.3. Kullanılan Araçlar

3.3.1. Orange

Orange3 veri analizi ve makine öğrenmesi uygulamaları için görsel arayüzlü bir araçtır. İçeriğinde makine öğrenimi ve veri analizini kolaylaştıran çeşitli algoritmalar bulunmaktadır. Orange, Anaconda[19] programının sunduğu bir platformdur. Anaconda yüklendikten sonra, kolayca Anaconda arayüzünden çalıştırılabilmektedir. Anaconda programının, windows işletim sistemi için minimum sistem gereksinimleri aşağıdaki gibidir:

- Microsoft® Windows® 7/8/10 (32- ya da 64-bit), macOS ya da Linux
- Python 2.7, 3.4, 3.5 ya da 3.6,
- Minimum 3gb kullanılabilir disk alanı,

Orange programı Anaconda'yla birlikte geldiği için, gereken sistem gereksinimleri Orange için de geçerlidir.

3.3.2. Windows 10 işletim sistemi

Projenin birinci aşamasında; geliştirdiğimiz uygulama için işletim sistemi olarak Windows 10'u tercih etme sebebimiz olarak önceden Windows kullanıcısı olmamız, Windows'a aşina olmamız, windowsun Python dilini desteklemesi söylenebilir.

Projenin ikinci aşamasında; Android Studio bütünleşik geliştirme ortamı Windows 10 işletim sistemine kurulmuştur.

3.3.3. Python 3.6 programlama dili

Geliştirdiğimiz uygulama için programlama dili olarak Python'ın şu anki en son sürümü olan 3.6 yı kullanmayı tercih ettik. Python tercih etme sebebimiz olarak güncel makine öğrenmesi kütüphanelerinin Python desteğinin fazla olması söylenebilir. Python 3.6 sürümünü tercih etme sebebimiz olarak en güncel sürüm olması ve güncel makine öğrenmesi kütüphaneleri ile uyumluluk sorunu içermemesi söylenebilir.

Python 3.6, internet aracılığıyla kendi sitesinden ücretsiz olarak indirilebilir [20].

3.3.4. Scikit – learn kütüphanesi

Görsel sınıflandırma için derin öğrenme yöntemi kullanan kütüphaneleri araştırdık. Caffé,

Caffe2, Chainer, CNTK(Microsoft Cognitive Toolkit), Deeplearning4j, Keras, MATLAB, MxNet, TensorFlow, Theano, Torch/PyTorch kütüphanelerini araştırdık.

Projenin başlarında uygulamada Google şirketinin geliştirdiği TensorFlow makine öğrenmesi kütüphanesini temel olarak kullanmayı tercih ettik.

Daha sonra makine öğrenmesi kütüphanelerinden Scikit-Image, Scikit-Learn gibi kütüphaneleri araştırdık. Scikit-Learn kütüphanesini kullanmaya karar verdik. Bu kütüphaneyi kullanmayı tercih etme sebebimiz olarak Scikit-Learn kütüphanesine aşina olmamız, proje kapsamında yapacağımız işler için yeterli olması ve göreceli olarak kolay olması söylenebilir.

3.3.5. Jupyter – notebook geliştirme ortamı

Geliştirme ortamı olarak Notepad++, Jupyter Notebook, atom, PyCharm gibi bazı uygulamaları araştırdık. Başlarda geliştirme ortamı olamadan Microsoft Command Line (Komut İstemi) kullandık. Sonradan proje büyüdükçe Jupyter-Notebook kullanmaya karar verdik. Bu kararı vermemize sebep olarak Jupyter-Notebook’a aşina olmamız ve kullanıcı dostu arayüzü olması söylenebilir.

Anaconda programı, Jupyter Notebook platformunu içermektedir.

3.3.6. Fatkun batch image downloader veri seti temin aracı

Projenin birinci aşamasında sınıflandırma tanımları ile ilgili veri setlerinin temini için Google Görseller sitesinden gerekli görselleri Chrome web tarayıcısının bir eklentisi olan Fatkun Batch Image Downloader kullanarak indirdik [21].

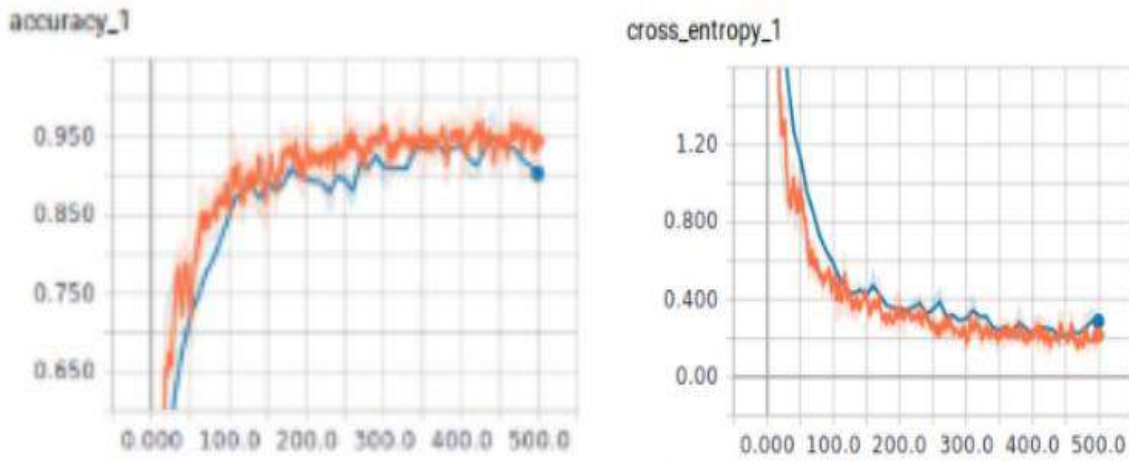
Fatkun Batch Image Downloader, Google görsellerden aranan fotoğrafları toplu olarak indirmeye yarayan bir programdır. İstenen fotoğraf kriterleri arama motoruna yazıldıktan sonra toplu indirme butonuna basılır ve indirilmesi istenmeyen fotoğraflar elenir. Elenmeyen fotoğraflar indirilir.

3.3.7. Linux işletim sistemi

Projenin ikinci aşamasında mobil uygulamada kullanılacak makine öğrenimi modelini oluşturmak için çoğunlukla ubuntu terminalinde linux kodları kullanılmıştır. Python programlama dilini hali hazırda desteklediği için işletim sistemi olarak bir Linux dağıtımı olan Ubuntu işletim sistemi tercih edilmiştir.

3.3.8. Tensorflow kütüphanesi

Projenin ikinci aşamasında; sci-kit kütüphanesi yerine tensorflow kütüphanesi kullanılmıştır. Mobil uygulamalara desteği, geniş makine öğrenmesi kütüphaneleri ve göreceli olarak kolay olması sebebiyle Google şirketinin geliştirdiği Tensorflow makine öğrenmesi kütüphaneleri tercih edilmiştir. Tensorflow, Virtualenv yöntemiyle ubuntu sistemine yüklenmiştir ve “source ~/tensorflow/bin/activate“ kodu ile terminalde çalıştırılmıştır.



Şekil 3.8. Tensorflow sonuçları

3.8 numaralı şekillerdeki grafikler Tensorflow kütüphanesinin içindeki Tensorboard aracı ile oluşturulmuştur. “tensorboard --logdir tf_files/training_summaries &” kodu ile makine öğrenmesi eğitim esnasındaki istatistikler “tf_files/training_summaries” konumunda saklanmıştır.

Makine öğrenmesi modeli Android uygulaması ile bütünleştirilirken Tensorflowun TFMobile aracı kullanılmıştır.

3.3.9. Android studio bütünleşik geliştirme ortamı

Mobil uygulama geliştirilirken, Google şirketinin Android uygulamaları geliştirmek için oluşturduğu bütünleşik geliştirme ortamı olan Android Studio kullanılmıştır. Android Studio “https://developer.android.com/studio/” adresinden temin edilir. Ekran yönergeleri takip edilerek kurulumu yapılır. Android Studio bütünleşik geliştirme ortamı, gerekli bağımlılık (dependency) dosyalarını otomatik olarak indirmektedir.

Android Studio geliştirme ortamının Windows işletim sistemi için sistem minimum gereksinimleri:

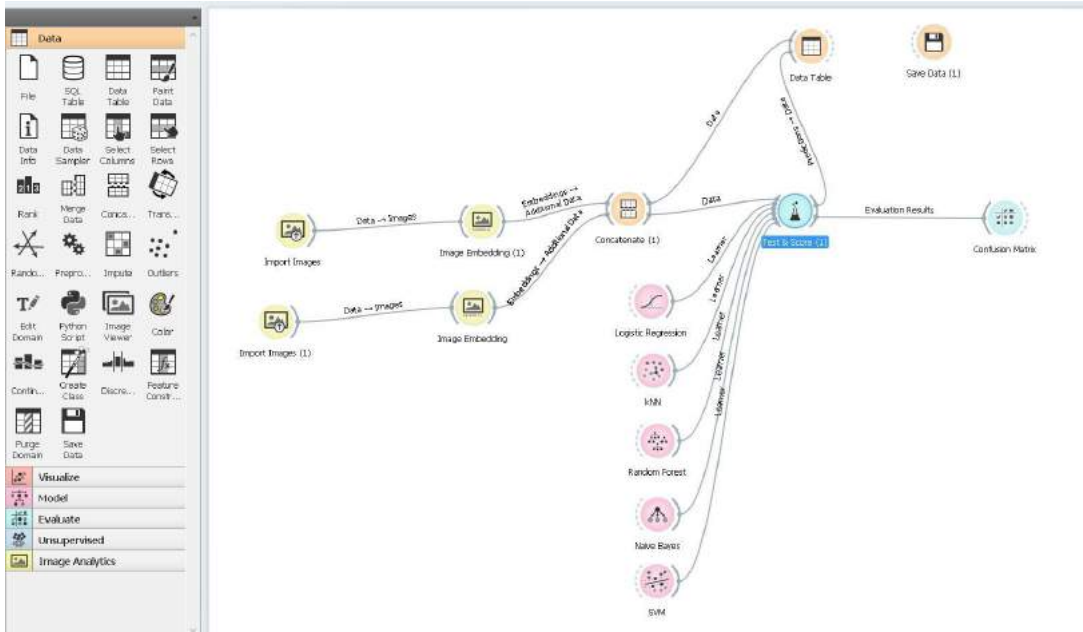
- Microsoft® Windows® 7/8/10 (32- ya da 64-bit)
- 3 GB RAM minimum, 8 GB RAM tavsiye edilen; ekstra 1 GB Android Emulator için,
- Minimum 2 GB kullanılabilir disk alanı,
- 4 GB tavsiye edilen (500 MB IDE için + 1.5 GB Android SDK ve 2 GB emulator sistemi görüntüsü için)
- 1280 x 800 minimum ekran ekran çözünürlüğü

Mobil uygulama Java programlama dilinde Android Studio ortamında gerçekleştirilmiştir. Makine öğrenimi modeli sonradan bütünleştirilmiştir.

4. GERÇEKLEŞTİRİLEN UYGULAMA

Literatür taramasında incelenen ve bahsedilen üç farklı örnekteki aşamalara benzer aşamalar içeren bir sistem geliştirildi. Bu sistem; görüntülerin toplanması, görüntü işleme yöntemleriyle analizi sonucunda özellik çıkarımı, oluşturulan modellere yönelik çeşitli algoritmaların uygulanması ve test sonuçlarının analizini içermektedir.

Projenin birinci aşamasında, şekil-4.1'de görüldüğü gibi orange aracı kullanılarak algoritmaların başarı sonuçları elde edilmiş ve analiz edilmiştir. Bu aşamada elde edilen deneyimler projenin ikinci aşamasında kullanılmıştır.



Şekil 4.1. Orange üzerinde örnek şema

4.1. Veri Kümesi (Yaprak Görselleri)

Daha önceki örnek çalışmalarda edinilen tecrübelerle göre, verimli bir görüntü işleme için, tutarlı ve belirlenen kriterlere uygun resim seçmek son derece önemlidir. Belirlenen bu kriterler; düzgün arka plan, düzgün ve yeterli ışıklandırma, yeterli çözünürlük, uygun fotoğraf mesafesi şeklindedir. Bu kriterlere uygun resimler, tek tek incelenerek, veri seti için toplanmıştır. Toplanan veri setindeki resimlerin vektörlere dönüştürülmesi, öznelite çıkarmı yapılması için Orange 3 kullanılmıştır.

Veri setindeki yaprak fotoğraflarını seçerken yaprak görsellerinin sahip olması gereken kriterler:

- İyi aydınlatılmış ortamda çekilmesi
- Görselde yalnızca tek yaprağın odakta olması
- Arka planın mümkün olduğunca sade ve beyaz olması, gerekmektedir.



Şekil 4.2. Kriterlere uymayan yaprak görseli.

Şekil 4.2.'deki görsel makine öğrenmesi için kötü bir veridir. Bunun sebebi odakta birden çok yaprak bulunması ve arka planın karışık olmasıdır.



Şekil 4.3. Kriterlere uymayan yaprak görseli.

Şekil 4.3.'deki görsel makine öğrenmesi için kötü bir veridir. Bunun sebebi odakta birden çok yaprak bulunması ve arka planın bulanık olmasıdır.



Şekil 4.4. Kriterlere uymayan yaprak görseli.

Şekil 4.4.'deki görsel makine öğrenmesi için kötü bir veridir. Bunun sebebi arka planın bulanık olmasıdır.



Şekil 4.5. Kriterlere uymayan yaprak görseli.

Şekil 4.5.'deki görsel makine öğrenmesi için iyi bir veridir bunun sebebi odakta yalnızca bir yaprak vardır ve arka plan sade, beyazdır.

Bitkilerin hastalığını tespit etmek için, yeşil yapraklı bitkilerin yaprak görsellerine ihtiyaç olduğu tespit edilmiştir. Bu nedenle, öncelikle beş hastalık belirlenmiş ve bu hastalıklara sahip olan bitkilerin yaprak görselleri ve sağlıklı bitkilerin yaprak görselleri manuel olarak toplanarak, altı sınıflı bir veri seti hazırlanmıştır. Manuel olarak hazırlanan veri setindeki bazı hastalık türleri ve içerdikleri görseller Şekil 4.6 - 4.11' de gösterilmiştir.



Şekil 4.6. Toplanan sağlıklı yaprak görsellerinden bazıları



Şekil 4.7. Toplanan bakteriyel lekeli yaprak görsellerinden bazıları



Şekil 4.8. Erken yanık hastalığına yakalanmış bitki görselleri



Şekil 4.9. Kurbağa göz yaprağı görselleri



Şekil 4.10. Mantar hastalıklarına ait yaprak görselleri



Şekil 4.11. Güneş yanığı hastalıklarına ait yaprak görselleri

Daha başarılı sınıflandırma için, ya toplanan görseller bir takım ön işlemlerden geçmeli ya da ön işlemler sonucu ulaşılmak istenen sonuca yakın görseller toplanmalıdır. Projenin bu aşamasında, gürültü azaltarak başarıyı artırma işlemi es geçilmiştir.

Yapılan çalışmalar sonucunda, veri setinin kalitesinin, başarı yüzdesi için çok önemli olduğu tespit edilmiştir. Manuel olarak, internet aracılığıyla, kapsamlı bir veri seti için, yaprak görsellerinin tek tek toplanması pek mümkün değildir. Bu nedenle, çalışmanın geleceğini değerlendirmek amacıyla, kapsamlı bir veri setine sahip olunduğu durumda, başarı yüzdesinin ne olacağı önemlidir. Bu durumu test etmek amacıyla, “Plantvillage disease classification challenge” isimli yarışmanın açık olarak sunduğu veri seti kullanılmıştır. 38 sınıfı içeren 21.917 yaprak görseli eğitim amacıyla kullanılmıştır.

Veri setindeki 38 sınıf; Elma uyuzu (*Venturia Inaequalis*), elma siyah çürüğü (*Botryosphaeria Obtusa*), elma sedir pası (*Gymnosporangium Juniperi- Virginianae*), sağlıklı elma, sağlıklı yaban mersini, sağlıklı kiraz, kiraz tozlu küf (*Podoshiera Clandestine*), mısır gri yaprak lekesi (*Cercospora Zeae- Maydis*), mısır ortak pası (*Puccinia Sorghi*), sağlıklı mısır, mısır kuzey yaprak yanığı (*Exserohilum Turcicum*), üzüm siyah çürüğü (*Guignardia Bidwellii*), üzüm siyah kızamığı (*Phaeomoniella Aleophilum*, *Phaeomoniella Chlamydospora*), sağlıklı üzüm, üzüm yaprak yanığı (*Pseudocercospora Vitis*), portakal Huanglongbing (*Candidatus Liberibacter spp*), şeftali bakteriyel noktası (*Xanthomonas Campestris*), sağlıklı şeftali, dolmalık biber bakteriyel nokta (*Xanthomonas Campestris*), sağlıklı dolmalık biber, patates erken yanığı, (*Alternaria Solani*), sağlıklı patates, patates geç yanığı (*Phytophthora Infestans*), sağlıklı ahududu, sağlıklı soya fasülyesi, kabak tozu küfü (*Erysiphe Cichoracearum*), sağlıklı çilek, çilek yaprağı alazı (*Diplocarpon Earlianum*), domates bakteriyel noktası (*Xanthomonas Campestris* pv. *Vesicatoria*), domates erken yanığı (*Alternaria Solani*), domates geç yanığı

(Phytophthora Infestans), domates yaprağı kalıbı (Passalora Fulva), domates septoria yaprağı lekesi (Septoria Lycopersici), domates iki benekli örümcek akarı (Tetranychus Urticae), domates hedef noktası (Corynespora Cassiicola), domates mozaik virüsü, domates sarı yaprak virüsü, sağlıklı domates şeklindedir.

Elma Uyuzu			
Elma Siyah Çürüğü			
Elma Sedir Pası			
Sağlıklı Elma			
Sağlıklı Yaban Mersini			

Sağlıklı Kiraz			
Kiraz Tozlu Küf			
Mısır Gri Yaprak Lekesi			
Mısır Ortak Pası			

Şekil 4.12. Büyük veri setindeki bazı görseller

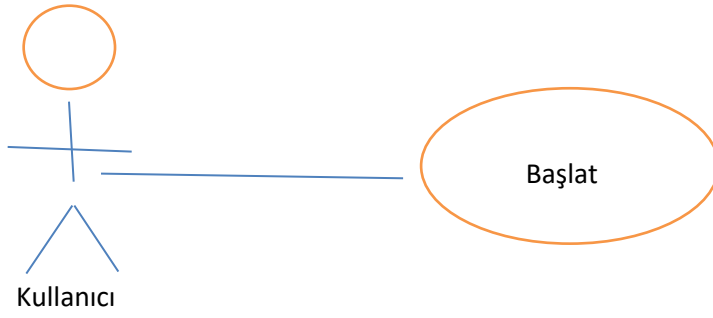
4.2. Sistem Tasarımı

Kullanıcı, uygulamayı başlatır. (Bkz. Şekil 4.13)

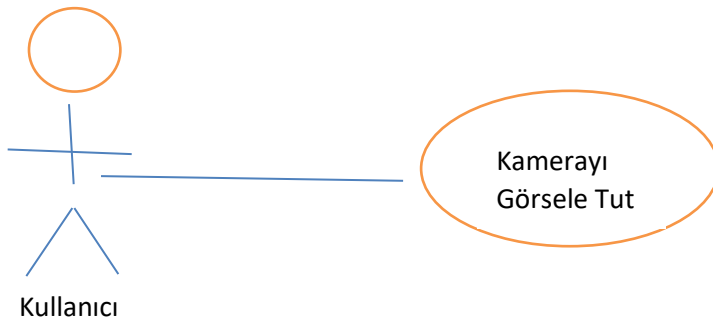
Kullanıcı, telefonun kamerasını, yaprak görseline yöneltir. (Bkz. Şekil 4.14)

Sistem, anlık ve dinamik olarak yaprak görselini sınıflandırır ve kullanıcıya sonucu döndürür. (Bkz. Şekil 4.15)

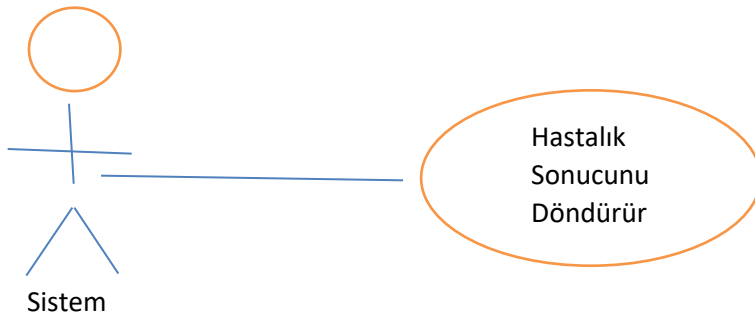
Kullanıcı uygulamadan çıkış yapabilir. (Bkz. Şekil 4.16)



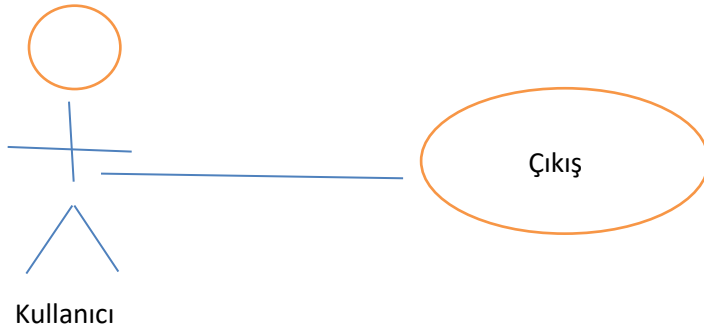
Şekil 4.13. Kullanıcı uygulamayı başlatır.



Şekil 4.14. Kullanıcı, kamerayı yaprak görseline tutar.



Şekil 4.15. Sistem sonuç döndürür.



Şekil 4.16. Kullanıcı çıkış yapar.

4.3. Yaprak görsellerinin sınıflandırılması

4.3.1 Görüntülerin Vektörlere Dönüştürülmesi

Toplanan görseller projenin birinci aşamasında, Orange3 aracı kullanılarak Inception v3 öğrenme aktarımı modeliyle öznitelikleri çıkarılıp, veri setine dönüştürülmüştür. Veri setine ait bazı öznitelik örnekleri şekil 4.17’de görülmektedir.

```
n0,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,n16,n17,n18,n19,n20,n21,n22,n23,n24,n25,n26,n27,n28,n29,n30,n31,n32,n33,n34,n35,n36,n37,n38,n39,n40,n41,n42,n43,n44,n45,n46,n47,n48,n49,
continuous,continuous,continuous,continuous,continuous,continuous,continuous,continuous,continuous,continuous,continuous,continuous,continuous,continuous,continuous,continuous,
0.169189453125,0.429931640625,0.3984375,0.189697265625,0.2705078125,0.0775146484375,0.161499023438,0.455078125,0.10888671875,0.69775390625,0.372802734375,0.2529296875,0.483642578125,0.629:
0.9853515625,0.60400390625,0.9638671875,0.5322265625,3.388671875,0.146728515625,0.219848632813,0.1826171875,0.3701171875,0.161865234375,0.0486145019531,0.070556640625,2.0234375,0.64990234:
0.3037109375,0.250244140625,0.149536132813,0.0277557373047,0.362548828125,0.0680541992188,0.3857421875,0.50390625,0.0161285400391,0.58203125,0.0516662597656,1.146484375,0.57861328125,0.32:
0.329833984375,0.188354492188,0.8828125,0.0244750976563,0.52001953125,0.100158691406,0.126098632813,0.33740234375,0.0628051757813,0.0386352539063,0.150390625,0.169311523438,0.6376953125,0.6:
0.62109375,0.0753173828125,2.744140625,0.148193359375,2.392578125,0.30078125,0.121765136719,0.371826171875,0.218994140625,0.402587890625,0.0279693603516,0.6025390625,2.193359375,0.4472656:
0.47802734375,0.465576171875,0.5048828125,0.91162109375,3.51171875,0.457763671875,0.55224609375,0.40087890625,0.114990234375,0.153198242188,2.11000442505e-05,0.17578125,2.255859375,1.4951:
0.98779296875,0.8876953125,0.47265625,0.238159179688,1.0771484375,0.45849609375,0.0556640625,0.80029296875,0.40478515625,0.0430603027344,0.239501953125,0.0109024047852,0.58056640625,0.813:
0.5146484375,0.0848388671875,0.62890625,0.209594726563,2.171875,1.396484375,0.0433349609375,0.415771484375,0.432373046875,0.193481445313,0.0333862304688,0.1806640625,0.72265625,0.31225585:
0.27392578125,0.24267578125,2.32421875,0.1533203125,0.4482421875,0.73291015625,0.125610351563,0.139770507813,0.285400390625,0.56689453125,0.312255859375,0.6396484375,0.77392578125,0.08129:
0.44091796875,0.182373046875,0.466796875,0.3447265625,0.9716796875,0.77685546875,1.1083984375,0.43017578125,0.282470703125,0.238037109375,0.0454711914063,0.0950927734375,0.0198211669922,0.3:
0.365966796875,0.798828125,0.183349609375,0.501953125,0.64306640625,0.77294921875,0.100158691406,0.175415039063,0.102661132813,0.255859375,0.0840454101563,0.276611328125,0.238403320313,0.:
0.52880859375,0.0424194335938,0.461181640625,0.0326232910156,2.107421875,0.106201171875,0.00484085083008,0.199096679688,0.00936126708984,0.106018066406,0.00652313232422,0.0187225341797,0.0:
0.0480651855469,0.0395812988281,0.3310546875,0.159912109375,0.0116348266602,0.0872192382813,0.0436096191406,0.118347167969,0.170166015625,0.254150390625,0.142333984375,0.32421875,0.11328:
0.136474609375,0.6767578125,0.490966796875,0.0234222412109,0.1533203125,0.51025390625,0.42138671875,0.34130859375,0.136108398438,0.7626953125,0.0235137939453,0.102294921875,0.7431640625,0.0:
0.0655517578125,0.0663452148438,1.009765625,0.0212097167969,0.00733184814453,0.135498046875,0.173217773438,0.21240234375,0.0382080078125,0.0382080078125,0.02712851563,0.026290895547,0.7783203125,0.127:
0.7265625,0.1298828125,0.61279296875,0.304931640625,0.32666015625,0.162231445313,0.15173398438,0.466796875,0.0385131835938,0.54833984375,0.093505859375,0.412353515625,0.5947265625,0.0497:
0.0802612304688,0.98193359375,0.80615234375,0.211059570313,1.3134765625,0.164184570313,0.0453796386719,0.0787963867188,0.0778198242188,0.126831054688,0.0123825073242,0.128540039063,0.917:
0.0753173828125,0.187866210938,1.025390625,0.00519943237305,0.7421875,0.398681640625,0.150268554688,0.48779296875,0.106811523438,0.311279296875,0.0770874023438,0.033203125,0.39697265625,0.0:
0.04833984375,0.152465820313,0.45166015625,0.173828125,0.176513671875,0.53662109375,0.5224609375,0.48193359375,0.139282226563,0.491943359375,0.0454711914063,0.7451171875,0.6923828125,0.22:
0.340087890625,0.120971679688,0.157348632813,0.64013671875,0.3564453125,0.498046875,0.049495117188,0.62939453125,0.39013671875,0.238891601563,0.25634765625,0.08984375,0.0165405273438,0.0:
0.84521484375,0.6865234375,1.267578125,0.0431518554688,3.013671875,1.0439453125,0.13818359375,0.445556640625,0.465576171875,0.33251953125,0.0277252197266,0.0597534179688,1.7412109375,0.98:
0.0323486328125,0.265625,0.31787109375,0.0500183105469,0.0,0.0946044921875,0.0326538085938,0.101257324219,0.0503540039063,0.0142517089844,0.51318359375,0.2919921875,0.10546875,0.236083984:
0.556640625,0.21875,0.247192382813,0.455810546875,1.208984375,0.228271484375,0.105834960938,0.2841796875,0.0814208984375,0.0932006835938,0.0377807617188,0.50048828125,1.091796875,0.014434:
0.689453125,0.16162109375,1.6875,0.56787109375,1.013671875,0.0245361328125,0.260986328125,0.595703125,0.130615234375,0.322998046875,0.117980957031,0.8759765625,2.001953125,0.119323730469,0.0:
0.121887207031,0.48046875,1.0888671875,0.0228729248047,0.122253417969,1.2021484375,0.77783203125,0.439697265625,0.0640258789063,0.434814453125,0.00351333618164,0.181030273438,1.114257812:
0.202880859375,0.3173828125,0.0985107421875,0.86865234375,0.9404296875,0.1826171875,0.0471801757813,0.00029850061035,0.0471496582031,0.41748046875,0.0227813720703,0.215454101563,0.10620:
```

Şekil 4.17. Öğrenme aktarımı yöntemiyle özellikleri çıkarılan görsellerin vektör değerleri

4.3.2 Algoritmaların Uygulanması

Projenin birinci aşamasında oluşturulan vektörel veri seti üzerinde bazı işlemler yapılmıştır. İlk iki satır silinmiş, ayrıca veri seti etiketlenmiştir. Data bölümü 2048 özelliğin her yaprak için bulunduğu veri setini ifade etmektedir. Etiket ise, altı adet sınıfın olduğu ve yaprakların hangi sınıfa ait olduğunu göstermektedir. Daha sonra, bu veri seti kullanılarak makine öğrenmesi algoritmalarıyla, test sonuçları karşılaştırılmıştır.

```
X=pd.read_csv("YaprakData.csv")
Y=pd.read_csv("YaprakTarget.csv")
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
random_state=0)
```

```
X_train.shape
```

```
(187, 2048)
```

```
X_test.shape
```

```
(63, 2048)
```

```
Y_train.shape
```

```
(187, 1)
```

```
Y_test.shape
```

```
(63, 1)
```

Şekil 4.18. Veri setinin eklenmesi

Şekil 4.18’ de görüldüğü gibi, veri seti, görsellere ait özniteliklerin bulunduğu “X” ve sınıf bilgilerinin bulunduğu “Y” bölümlerine ayrılmıştır. Veri seti ayrıca, %75’i eğitim ve %25’ i test olacak şekilde bölünmüştür. Şekil 4.18’ de bölümlere ayrılan veri seti, python programlama diliyle, sisteme yüklenmiştir. X ve Y bölümleri için, eğitim setinde 187 ve test setinde 63 örnek vardır. Toplam 2048 öznitelik bulunmaktadır. “Shape” komutu, bu özellikleri göstermek amacıyla kullanılmıştır.

Makine öğrenmesi algoritmalarının uygulanmasından sonra, test sonuçları karşılaştırılmıştır. Bunun sonucunda, en başarılı algoritmanın, Destek Vektör Makinesi (Support Vector Machine – SVM) olduğu tespit edilmiştir. (Bkz Şekil 4.19)

```

svm=SVC(C=10,gamma=0.001)
svm.fit(X_train,Y_train)
print("Training score: {:.3f}.\n".format(svm.score(X_train,Y_train)))
print("Test score: {:.3f}.\n".format(svm.score(X_test,Y_test)))

#grid search
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100],
              'gamma': [0.001, 0.01, 0.1, 1, 10, 100]}
grid_search = GridSearchCV(SVC(), param_grid, cv=5)
grid_search.fit(X_train, Y_train)
GridSearchCV(cv=5, error_score='raise', estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False),
fit_params={}, iid=True, n_jobs=1,
param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100], 'gamma': [0.001, 0.01, 0.1, 1, 10, 100]}),
pre_dispatch='2*n_jobs', refit=True, scoring=None, verbose=0)
print("Grid search SVC training score: {:.3f}.".format(grid_search.score(X_train,Y_train)))
print("Grid search SVC test score: {:.3f}.".format(grid_search.score(X_test,Y_test)))
print(grid_search.best_params_)
print(grid_search.best_score_)

Training score: 0.995.

Test score: 0.810.

Grid search SVC training score: 0.995
Grid search SVC test score: 0.810
{'C': 10, 'gamma': 0.001}

```

Şekil 4.19. SVM algoritması test sonuçları

```

#KNN k parameter 1 to 20
results_knn=[]
for i in range(1,20):
    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,Y_train)
    results_knn.append(knn.score(X_test,Y_test))
max_accuracy_knn=max(results_knn)
best_k=1+results_knn.index(max(results_knn))
print("Max accuracy is {:.3f} on test dataset with {} neighbors.\n".format(max_accuracy_knn,best_k))
plt.plot(np.arange(1,20),results_knn)
plt.xlabel("n neighbors")
plt.ylabel("Accuracy")
knn=KNeighborsClassifier(n_neighbors=best_k)
knn.fit(X_train,Y_train)
print("Training score: {:.3f}.".format(knn.score(X_train,Y_train)))
print("Test score: {:.3f}.".format(knn.score(X_test,Y_test)))
print(classification_report(Y_test, knn.predict(X_test)))

Max accuracy is 0.794 on test dataset with 9 neighbors.

Training score: 0.733
Test score: 0.794

```

Şekil 4.20. KNN algoritması test sonuçları

```
#random forest estimators 2 to 30
results_forest=[]
for i in range(2,30):
    forest=RandomForestClassifier(random_state=12,n_estimators=i)
    forest.fit(X_train,Y_train)
    results_forest.append(forest.score(X_test,Y_test))

max_accuracy_forest=max(results_forest)
best_n_est=2+results_forest.index(max(results_forest))
print("Max Accuracy is {:.3f} on test dataset with {} estimators.\n".format(max_accuracy_forest,best_n_est))

plt.plot(np.arange(2,30),results_forest)
plt.xlabel("n_estimators")
plt.ylabel("Accuracy")

forest=RandomForestClassifier(random_state=12,n_estimators=11)
forest.fit(X_train,Y_train)
print("Training score: {:.3f}".format(forest.score(X_train,Y_train)))
print("Test score: {:.3f}".format(forest.score(X_test,Y_test)))
```

Max Accuracy is 0.698 on test dataset with 23 estimators.

Training score: 0.984
Test score: 0.619

Şekil 4.21. Rassal Orman algoritması test sonuçları

```
neural=MLPClassifier(max_iter=400,random_state=0,hidden_layer_sizes=[40,40])
neural.fit(X_train,Y_train)
print("Training score: {:.3f} .\n".format(neural.score(X_train,Y_train)))
print("Test score: {:.3f} .\n".format(neural.score(X_test,Y_test)))

#metrics
print(classification_report(Y_test, neural.predict(X_test)))
```

Training score: 0.995 .

Test score: 0.794 .

Şekil 4.22. Yapay sinir ağı algoritması test sonuçları

```
#logistic c=1
logisticregression = LogisticRegression().fit(X_train, Y_train)
print("Training score: {:.3f}".format(logisticregression.score(X_train, Y_train)))
print("Test score: {:.3f}".format(logisticregression.score(X_test, Y_test)))

print(classification_report(Y_test, logisticregression.predict(X_test)))
```

Training score: 0.995
Test score: 0.794

Şekil 4.23. Lojistik regresyon algoritması test sonuçları

```

gnb = GaussianNB()
gnb.fit(X_train,Y_train)
print("Training score: {:.3f}".format(gnb.score(X_train,Y_train)))
print("Test score: {:.3f}".format(gnb.score(X_test,Y_test)))

#metrics
print(classification_report(Y_test, gnb.predict(X_test)))

Training score: 0.941
Test score: 0.714

```

Şekil 4.24. Naive Bayes algoritması test sonuçları

Manuel olarak toplanan veri setine, algoritmalar uygulandıktan sonra, büyük veri seti üzerinde çalışılmıştır. Şekil 4.14’ te büyük veri seti eklenmiştir.

```

X=pd.read_csv("dataD.csv")
Y=pd.read_csv("targetD.csv")
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X.drop([0,1]), Y.drop([0,1]),
random_state=0)

```

X_train.shape
(16437, 2048)
X_test.shape
(5480, 2048)
Y_train.shape
(16437, 1)
Y_test.shape
(5480, 1)

Şekil 4.25. Büyük veri setinin yüklenmesi

Şekil 4.14’ te görüldüğü üzere, veri seti X ve Y parçalarına ayrılmış, eğitim ve test olarak da iki parça halinde yüklenmiştir.

```
#random forest estimators 2 to 30
results_forest=[]
for i in range(2,30):
    forest=RandomForestClassifier(random_state=12,n_estimators=i)
    forest.fit(X_train,Y_train)
    results_forest.append(forest.score(X_test,Y_test))

max_accuracy_forest=max(results_forest)
best_n_est=2+results_forest.index(max(results_forest))
print("Max Accuracy is {:.3f} on test dataset with {} estimators.\n".format(max_accuracy_forest,best_n_est))

plt.plot(np.arange(2,30),results_forest)
plt.xlabel("n estimators")
plt.ylabel("Accuracy")

forest=RandomForestClassifier(random_state=12,n_estimators=11)
forest.fit(X_train,Y_train)
print("Training score: {:.3f}".format(forest.score(X_train,Y_train)))
print("Test score: {:.3f}".format(forest.score(X_test,Y_test)))
```

Max Accuracy is 0.799 on test dataset with 29 estimators.

Training score: 0.999
Test score: 0.713

Şekil 4.26. Büyük veri setinde rassal orman algoritmasının test sonuçları

```
svm=SVC(C=10,gamma=0.001)
svm.fit(X_train,Y_train)
print("Training score: {:.3f}.\n".format(svm.score(X_train,Y_train)))
print("Test score: {:.3f}.\n".format(svm.score(X_test,Y_test)))
```

Training score: 0.994.

Test score: 0.962.

Şekil 4.27. Büyük veri setinde SVM algoritmasının test sonuçları

```
neural=MLPClassifier(max_iter=400,random_state=0,hidden_layer_sizes=[40,40])
neural.fit(X_train,Y_train)
print("Training score: {:.3f} .\n".format(neural.score(X_train,Y_train)))
print("Test score: {:.3f} .\n".format(neural.score(X_test,Y_test)))

#metrics
print(classification_report(Y_test, neural.predict(X_test)))
```

Training score: 1.000

Test score: 0.964

Şekil 4.28. Büyük veri setinde yapay sinir ağı algoritmasının test sonuçları

```
#logistic c=1
logisticregression = LogisticRegression().fit(X_train, Y_train)
print("Training score: {:.3f}".format(logisticregression.score(X_train, Y_train)))
print("Test score: {:.3f}".format(logisticregression.score(X_test, Y_test)))

print(classification_report(Y_test, logisticregression.predict(X_test)))

Training score: 1.000 .

Test score: 0.948 .
```

Şekil 4.29. Büyük veri setinde lojistik regresyon algoritmasının test sonuçları

```
gnb = GaussianNB()
gnb.fit(X_train,Y_train)
print("Training score: {:.3f}".format(gnb.score(X_train,Y_train)))
print("Test score: {:.3f}".format(gnb.score(X_test,Y_test)))

#metrics
print(classification_report(Y_test, gnb.predict(X_test)))

Training score: 0.774
Test score: 0.752
```

Şekil 4.30. Büyük veri setinde naive bayes algoritmasının test sonuçları

4.4 Sonuçların Karşılaştırılması

Projenin birinci aşamasında orange3 aracını kullanarak elde edilen, 5 farklı algoritmaya ait test sonuçları tablo-1’de görülmektedir. Sınıflandırma başarısı (classification accuracy), CA başlığı altında gösterilmektedir.

Projenin ikinci aşamasında orange3 aracını kullanarak elde edilen, 5 farklı algoritmaya ait test sonuçları tablo-2’de görülmektedir. Mobil uygulamada kullanılacak algoritma olarak sinir ağı algoritmasının tercih edilmesine karar verilmiştir. Sınıflandırma başarısı (classification accuracy), CA başlığı altında gösterilmektedir.

Hassasiyet doğru pozitiflerin bütün pozitiflere oranıdır. Denklem (1)’ de hassasiyetin matematiksel ifadesi verilmiştir.

$$Hassasiyet = \frac{tp}{tp+fp} \quad (1)$$

Kesinlik doğru pozitiflerin, doğru positif ve hatalı negatiflerin toplamına oranıdır. Denklem (2)’ de kesinliğin matematiksel ifadesi verilmiştir.

$$Kesinlik = \frac{tp}{tp+fn} \quad (2)$$

F1 skoru istatistiksel terimler olan hassasiyet ve kesinliğin ağırlıklı ortalamalarıdır. Hatalı positif (false positive) ve hatalı negatif (false negative) sayıları hesaba katarak bulunur. Sınıf verileri dengesiz olduğunda kullanışlıdır. Denklem (3)' te F1 skorunun matematiksel ifadesi verilmiştir.

$$F1 = 2 \frac{Kesinlik * Hassasiyet}{Kesinlik + Hassasiyet} \quad (3)$$

CA değeri, model üzerinde uygulanan algoritmanın, test kümesiyle denenmesi sonucu, elde edilen doğruluğun, yüzdesel ifadesidir.

AUC: normalize edilmiş üniteler kullanıldığında, eğri altındaki alan (area under curve,AUC), bir sınıflandırıcının rastgele seçilmiş bir negatif örnekden rastgele seçilmiş bir pozitif sırayı daha yüksek sıralayabileceği olasılığına eşittir.

Tablo 1. Manuel olarak toplanan veri setinin başarı değerleri

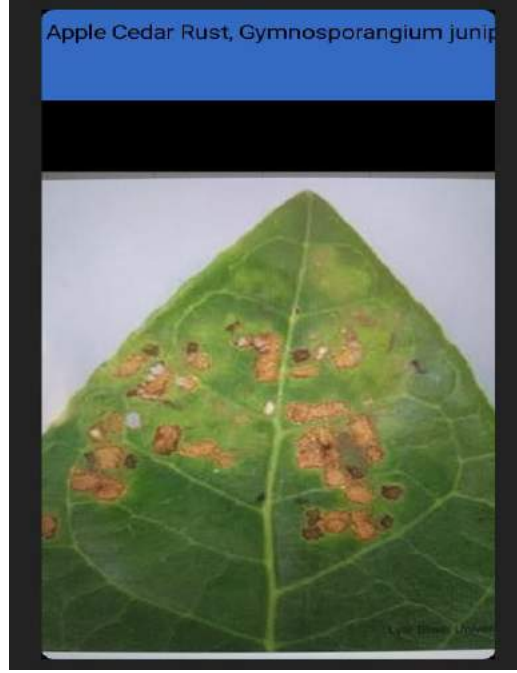
Metot	AUC	CA	F1	Hassasiyet	Kesinlik
kNN	0.877	0.794	0.770	0.800	0.791
SVM	0.917	0.810	0.800	0.810	0.810
Rassal Orman	0.771	0.619	0.581	0.611	0.620
Sinir Ağı	0.904	0.794	0.790	0.801	0.790
Naive Bayes	0.819	0.714	0.711	0.720	0.711
Lojistik Regresyon	0.916	0.794	0.790	0.810	0.790

Tablo 2. Büyük veri setinin başarı değerleri

Metot	AUC	CA	F1	Hassasiyet	Kesinlik
kNN	0.990	0.888	0.885	0.892	0.888
SVM	1.000	0.966	0.966	0.966	0.966
Rassal Orman	0.957	0.739	0.723	0.724	0.739
Sinir Ağı	0.999	0.970	0.970	0.970	0.970
Naive Bayes	0.920	0.749	0.757	0.788	0.749
Lojistik Regresyon	0.999	0.955	0.955	0.955	0.955

4.5 Arayüz Tasarımı

Android tabanlı mobil uygulama, Android Studio ortamında geliştirilmiştir. Şekil 4.31’ de görüldüğü şekilde bir arayüz geliştirilmiştir. Arayüz son derece sadedir. Kullanıcının yapması gereken, telefonunun kamerasını, yaprak görseline tutmak ve sonucu okumaktır.



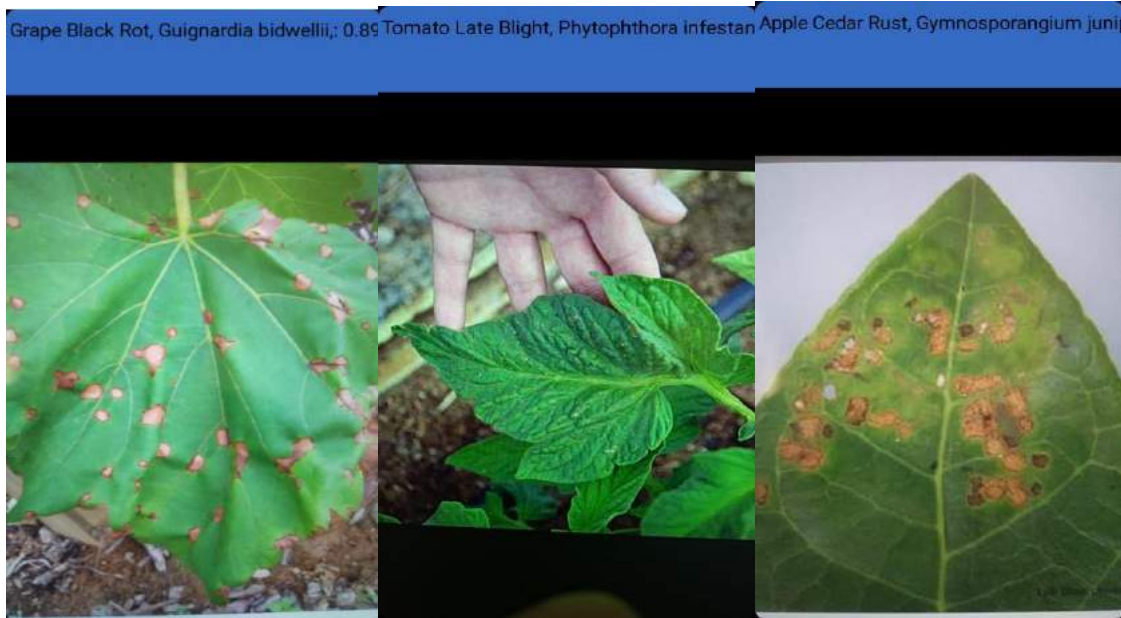
Şekil 4.31. Uygulama arayüzü ve yaprak görseliyle test

5 SONUÇ VE DEĞERLENDİRME

Gerçekleştirilen çalışmada, toplanan veri setinden, inception v3 modeliyle, öznitelikler çıkarılmıştır. En başarılı algoritmayı tespit etmek için, jupiter notebook ortamında python programlama dili kullanılarak, bir model oluşturulmuştur. Model üzerinde, birçok algoritma denenerek, en başarılı algoritma belirlenmiştir. Bu başarı yüzdesine göre seçilen algoritma kullanılarak, android tabanlı, sade arayüzlü bir mobil uygulama geliştirilmiştir.

Veri seti üzerinde uygulanan algoritmaların başarı sonuçları karşılaştırılmıştır. Buna göre en başarılı algoritma yapay sinir ağı ile derin öğrenme olmuştur.

Mobil uygulama birkaç resimle test edilmiştir. Bu testler Şekil 5.1’ de verilmiştir.



Şekil 5.1. Mobil uygulamanın test işlemi

Bitki hastalıklarının erken tespiti için, bir takım çalışmalar incelenmiştir. Elde edilen bulgularda, bu alanda bazı eksiklikler göze çarpmıştır. Özellikle Türkiye’de, bu tarzda bir mobil uygulama tespit edilememiştir. Bunun sonucunda, bu açığı gidermek üzere, Android tabanlı mobil bir hastalık tespit sistemi geliştirilmiştir.

Geliştirilen sistem, kullanılabilirlik açısından son derece kolaydır. Uygulama açıldıktan sonra, hiçbir kullanıcı aktivitesine gerek kalmadan, sadece kameranın ilgili görsele yöneltmesiyle, sonuç kullanıcıya sunulmaktadır. Yapılan testlerde, ortalama %97’ lik bir başarı elde edilmiştir.

Bu çalışmayla, tarım alanında önemli bir sorun olarak görülen, bitki hastalıklarından kaynaklanan zaiyatlara çözüm getirmek amaçlanmıştır.

Çalışmanın devamında, Türkiye'deki bütün bitkileri kapsayan, büyük bir veri seti toplanması hedefler arasındadır. Böylece, bitki türüne bakılmaksızın, herhangi bir hastalığın tespiti yapılması mümkün olacaktır.

Çalışmanın, pratikte ciddi bir ekonomik tasarruf sağlayacağı öngörülmektedir. Aynı zamanda, gereksiz ilaç kullanımından doğan, sağlık problemlerinin önüne geçilecek ve ekolojik dengeyi bozan etmenler önlenecektir. Bu açıdan bakıldığında, tarım sektöründe önemli bir milat olacağı düşünülmektedir.

6 REFERANSLAR

- [1] Göktürk Ö. B., Korkut U. B., Yıldız O., “Makine Öğrenmesi ile Bitki Hastalıklarının Tespiti”, *26th International Conference on Signal Processing and Communications Applications (SIU)*, 2018
- [2] İnternet: http://www.tuik.gov.tr/PreIstatistikTablo.do?istab_id=2288
- [3] Bashish Al D., Braik M., Bani – Ahmad S., “A Framework for Detection and Classification of Plant Leaf and Stem Diseases”, *International Conference on Signal and Image Processing (ICSIP)*, 2010
- [4] Sannakki S. S., Rajpurohit V. S., Nargund V. B., Kulkarni P., “Diagnosis and Classification of Grape Leaf Diseases Using Neural Networks”, *4th International Conference on Computing Communications and Networking Technologies (ICCCNT)*, 2013
- [5] Arivazhagan S., Shebiah R. N., Ananthi S., Varthini S. V., “Detection of Unhealthy Region of Plant Leaves and Classification of Plant Leaf Diseases Using Texture Features”, *Agricultural Engineering International The CIGR e-journal* 15(1) p: 211-217, 2013
- [6] Revathi P., Hemalatha M., “Classification of Cotton Leaf Spot Diseases Using Image Processing Edge Detection Techniques”, *International Conference on Emerging Trends in Science Engineering and Technology (INCOSET)*, 2012
- [7] İnternet: <https://agtechtr.wordpress.com/2017/09/19/makine-ogrenmesi-modern-tarimi-nasil-degistirir/>
- [8] Berikol, G., Kula, S., Yıldız, O., "Machine Learning Techniques in Diagnosis of Pulmonary Embolism", *Journal of Clinical and Analytical Medicine*, 2015
- [9] Arslan, A., Yıldız, O., "Automated Auscultative Diagnosis System for Evaluation of Phonocardiogram Signals Associated with Heart Murmur Diseases", *Gazi University Journal of Science*, 31(1): 112-124, 2018
- [10] Karakış, Rukiye, İnan Güler, and Ali Hakan Işık. "Feature selection in pulmonary function test data with machine learning methods." *Signal Processing and Communications Applications Conference (SIU)*, 2013 21st. IEEE, 2013.
- [11] James W. C., ``Assessment of Plant Diseases and Losses", *Annual Review of Phytopathology*, 27-48, 1974.
- [12] C. H. Bock, G. H. Poole, P. E. Parker, and T. R. Gottwald., “Plant disease severity estimated visually, by digital photography and imageanalysis, and by hyperspectral imaging”, *Critical Reviews in Plant Sciences*, March 2010, vol. 29, pp. 59–107.
- [13] H. Al-Hiary, S. Bani-Ahmad, M. Reyalat, M. Braik, and Z.ALrahamneh, “Fast and accurate detection and classification of plantdiseases”, *International Journal of Computer Applications*, 2001, vol. 17, pp.31–38.
- [14] D. W. Zhang and J. Wang, “Design on image features recognition system of cucumber downy mildew based on BP algorithm”, *Journal of Shenyang Jianzhu University (Natural Science)*, May 2009, vol. 25, pp. 574–578.
- [15] D. T. Zhao, Y. H. Chai, and C. L. Zhang, “Inspection of soybean frog-eye spot based on image procession”, *Journal of Northeast Agricultural University*, vol. 41, pp. 119–124, April 2010.
- [16] Y. L. Cui, P. F. Cheng, X. Z. Dong, Z. H. Liu, and S. X. Wang, “Image processing and extracting color features of greenhouse diseased leaf”, *Transactions of the CSAE*. vol. 21(Supp.), pp. 32–35, December 2005

- [17] Meunkaewjinda A., Kumsawat P., Attakitmongcol K., Srikaew A., “Grape Leaf Disease Detection from Color Imagery Using Hybrid Intelligent System”, *5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, 2008
- [18] Camargo A., Smith J. S., “An Image Pattern Classification for the Identification of Disease Causing Agents in Plants”, *Comput. Electron. Agric.* 66(2), 121-125, 2009
- [19] Internet: “Anaconda”, <https://anaconda.org/anaconda/python>
- [20] Internet: “Python 3.6”, <https://www.python.org/downloads/release/python-360/>
- [21] Internet: “Fatkun Batch Image Downloader”, <https://chrome.google.com/webstore/detail/fatkun-batch-download-ima/nnjjahliabnchcpehckpkdeckfgnohf?hl=tr>