BURSA TEKNİK ÜNİVERSİTESİ

MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

RASSAL ORMAN YÖNTEMİ İLE KALP YETMEZLİĞİ TAHMİNİ VE ARAYÜZÜ PROGRAMLAMA

BİTİRME TEZİ

Muhammet Emin KAYIŞYAPAR

150109001

Elektrik-Elektronik Mühendisliği Bölümü

Danışman: Dr. Fatmatülzehra USLU

Haziran 2021

İNTİHAL BEYANI

Bu bitirme çalışmasında görsel, işitsel ve yazılı biçimde sunulan tüm bilgi ve sonuçların akademik ve etik kurallara uyularak tarafımdan elde edildiğini, bitirme çalışması içinde yer alan ancak bu çalışmaya özgü olmayan tüm sonuç ve bilgileri bitirme çalışmasında kaynak göstererek belgelediğimi, aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim.

Öğrencinin Adı Soyadı: Muhammet Emin Kayışyapar

İmzası: 

ÖZET

Kalp yetersizliği, kalp performansının azalması sonucu, kalbin doku ve organlara yeterli kanı gönderememesi sonucu ortaya çıkan bir rahatsızlıktır. Kalp damar hastalığı, kalp krizi, yüksek tansiyon (hipertansiyon), kalp kapak hastalığı gibi kalp performansını bozan ya da kalbin çalışmasını güçleştiren hastalıklar sonrasında ortaya çıkar. Kalp yetersizliği herhangi bir yaşta gelişebileceği gibi ilerleyen yaşla birlikte görülme sıklığı artar. Toplumda, <65 yaş kişilerde görülme sıklığı %1 iken, 75-84 yaş arasında görülme sıklığı %7'ye, >85 yaşta ise %15'e kadar çıkmaktadır. 65 yaş üstü kişilerde en sık hastaneye yatış nedenidir. [1]

Kalp yetersizliği hayat boyu devam eden kronik bir hastalık olduğu halde, erken teşhisle sebep olduğu hayati riskler azaltılabilmektedir. Hastalığın erken teşhisi ile ilaç tedavisi ve dikkat edilecek bir sosyal hayat ile hastanın hayat konforu arttırılarak yaşam süresi uzatılabilmektedir.

Yapay Zekâ ve Makine Öğrenmesi teknikleri günlük hayattaki basit veya karışık birçok probleme dahil olmaktadır. Bu yöntemler ile tecrübeye dayalı iş alanları, insan gücü ile uzun süren hesaplamalar ve analizler bu alandaki uzmanların eğittikleri makinelere devredilmektedir. Günlük hayatta sıklıkla kullandığımız teknolojik aletlerle birlikte (akıllı saat, akıllı telefon, kişisel bilgisayarlar) elde edebileceğimiz verileri değerlendirebilecek ve anlamlı sonuçlar çıkarabilecek uygulamalar insan hayatını olumlu yönde etkilecektir.

Bu çalışmada, kalp yetmezliği ile ilgili oluşturulan bir veri seti ile **Random Forest** yöntemi kullanılarak, veri setindeki parametreleri kullanıcıdan isteyerek kalp yetmezliği olasılığı hakkında riskli veya riskli değil şeklinde tahmin yapmak üzere eğitilen model, tasarlanan arayüze entegre edilmiştir.

Çalışmanın ana hedefi yapılan bir tahlil sonucunda ön tanı olarak kalp yetmezliği hakkında bir tahminde bulunmak ve bu aşamadan sonra yapılacak tetkiklerde yardımcı olarak kullanmaktır.

Anahtar Kelimeler: Kalp Yetmezliği**,** Makine Öğrenmesi, Rassal Orman, PyQt5

SUMMARY

Heart failure is a disease that occurs as a result of decreased heart performance and the inability of the heart to send blood enough to the tissues and organs. After that, some other diseases also occur such as cardiovascular disease, heart attack, high blood pressure (hypertension) and heart valve disease which impair heart performance or make it difficult for the heart to work. Besides, heart failure may develop at any age and its incidence increases with advancing age. In the society, while the incidence is 1% in people aged <65 years, the incidence increases to 7% between the ages of 75-84, and up to 15% in those ages >85 years. It is the most common reason for hospitalization in people over 65 years og age. [1]

Although heart failure is a lifelong chronic disease, the vital risks it causes can be reduced by early diagnosis. Lifespan of the patients can be extended by increasing the life comfort with the early diagnosis, drug treatment and a social life to be considered. Techniques of Artifical Intelligence and Machine Learning are involved lots of problems that are simple or complex in our daily life.

With these methods, same job-fields based on experience long-period calculations with human power and analyzes have been transferred to the machines that are trained by the experts.Applications that can evaluate the data we can obtain and draw meaningful results together with the technological devices we frequently use in our daily life (smart watch, smart phone, personal computers etc.) will positively affect human life. In this study, the model, which was trained to predict the probability of heart faliture as risky or not risky by asking the parameters in the dataset from the users, has been integrated into the designed interface by using Random Forest method with a data set that was created about heart faliture.

The main goal of this study is to make a prediction about heart faliture as an early diagnosis as a result of on analysis and to use as an assistant in the next analyzes.

Keywords: Machine Learning, Heart Failure, Random Forest, PyQt5

TEŞEKKÜR

Öncelikle tez konusunun belirlenmesi ve yazım süreci başta olmak üzere tezin her aşamasında bilgi ve tecrübesiyle bana yol gösteren tez danışmanım Dr. Öğr. Üyesi Fatmatülzehra USLU'ya teşekkür ederim.

Lisans eğitimim boyunca desteğini esirgemeyen bölüm hocalarım ve çalışma arkadaşlarıma, doğduğum andan itibaren tüm imkânları benim için seferber eden ve manevi destekleriyle hep yanımda olan aileme teşekkür ederim.

HAZİRAN 2021

MUHAMMET EMİN KAYIŞYAPAR

İÇİNDEKİLER

İNTİHAL BEYANI i
ÖZET ii
SUMMARY iii
TEŞEKKÜR iv
İÇİNDEKİLER v
ŞEKİLLER DİZİNİ vi
ÇİZELGELER DİZİNİ 7
1. GİRİŞ 8
1.1 TEZİN AMACI 8
1.2 LİTERATÜR TARAMASI 9
1.2.1 Makine Öğrenmesi 9
1.2.1.1 Öğretmeli Öğrenme (Supervised Learning) 9
1.2.1.2 Öğretmensiz Öğrenme (Unsupervised Learning) 9
1.2.1.3 Reinforcement (Pekiştirmeli) Learning 10
1.2.2 Sınıflandırma ve Regresyon 10
1.2.2.1 Naive Bayes 10
1.2.2.2 Karar Ağaçları 11
Gini ……..11
Entropy 12
1.2.2.3 Rassal Orman Yöntemi 12
1.2.3 Python ile Kullanıcı Arayüzü Tasarımı(GUI) 13
2. METOD 14
2.1 Kalp Yetmezliği Tahmini 14
2.2 Arayüz Tasarımı 19

2.3 Kalp Yetmezliği Tahmin Modelinin Arayüze Eklenmesi 20
3. SONUÇ 22
4. TARTIŞMA 24
KAYNAKLAR 25
EKLER 26

ŞEKİLLER DİZİNİ

Şekil 1.1 Gini Formülü [9] 11
Şekil 1.2 Entropy Formülü [9] 12
Şekil 1.3 Rassal Orman Yöntemi İşleyişi [11] 12
Şekil 2.1 Veri Setine Genel Bir Bakış 15
Şekil 2.2 Veri Seti Detayları 15
Şekil 2.3 Nümerik Değişkenlerin Dağılımı 16
Şekil 2.4 Kategorik Değişkenlerin Dağılımı 16
Şekil 2.5 Tukey's yöntemi ile anormallik tespiti 17
Şekil 2.6 SMOTE yöntemi ile Yeniden Örnekleme yapılması 17
Şekil 2.7 Box-cox yöntemi ile normalize edilen değişkenler 18
Şekil 2.8 Örnek bir RadioButton kullanımı 19
Şekil 2.9 LineEdit ve ToolTip 19
Şekil 2.10 PushButton kullanımı 20
Şekil 2.11 MessageBox kullanımı 20
Şekil 2.12 Random Forest Metodunu kullanma 21
Şekil 2.13 Program Çalışma Şekli ve Çıktılar 21
Şekil 3.1 Model Importance Sonuçları 22
Şekil 3.2 İlk Modeldeki başarı Değerleri 23
Şekil 3.3 İkinci Modeldeki Başarı Değerleri 23

ÇİZELGELER DİZİNİ

Tablo 1 Feature açıklama tablosu..... 14
Tablo 2 Importance Sonuçları için açıklama tablosu 22

1. GİRİŞ

1.1 TEZİN AMACI

Kalp yetersizliği, kalp performansının azalması sonucu, kalbin doku ve organlara yeterli kanı gönderememesi sonucu ortaya çıkan bir rahatsızlıktır. Kalp damar hastalığı, kalp krizi, yüksek tansiyon (hipertansiyon), kalp kapak hastalığı gibi kalp performansını bozan ya da kalbin çalışmasını güçleştiren hastalıklar sonrasında ortaya çıkar. Kalp yetersizliği konusunda hastaların hayati riskini azaltan en önemli faktör erken teşhis ve erken alınan tedbirlerdir. [1]

Günümüzde günlük hayatta herkesin kullandığı dijital cihazlar ile hayatımızda kritik rol oynayan uygulamalar vardır. Bu uygulamalara görüntülü konuşma uygulamaları, uzaktan erişim uygulamaları, analiz uygulamaları, sağlık uygulamaları gibi örnekler verilebilir. Bu uygulamaların hayatımızda bu kadar yaygın olmasındaki en önemli faktör grafiksel arayüzlerdir. Grafiksel kullanıcı arayüzü (GKA, a.k.a. GUI), uygulamaların kullanımını kolaylaştırmak için bilgisayarın grafik özelliklerini kullanan bir yazılım programları topluluğudur. Bir GUI, bilgisayarı ve yazılımı çalıştırmak için bir işaretçi, işaretleme aygıtı ve arabirim öğeleri kullanır.

Bu dijitalleşmenin yanında kullandığımız her uygulama ile dijital bir veri oluşturulmaktadır. Makine öğrenmesi algoritmaları bu verilerdeki kalıpları belirlemek ve işlemek için kullanılır. Bu çalışma kalp yetersizliği hakkında oluşturulan bir veri seti ile eğitilen modelle kalp yetmezliği tahmini yapılması hakkında olup eğitilen model, tasarlanan grafiksel bir ara yüz ile kullanıcıya sunulmuştur.

Bu ve benzeri çalışmalar sayesinde herhangi bir hastanın hastanede yaptırdığı bir tahlili veya veri tabanına alınan bilgileri ön işlemeden geçirilerek hastanın başvurduğu rahatsızlık ve başka rahatsızlıklar hakkında tahminler yapılarak hasta sağlığı hakkında doktor ve hastaya ön bilgi verilerek çoğu hastalıkta kritik rol oynayan erken teşhis konusunda ilerleme sağlanabileceği düşünülmektedir.

1.2 LİTERATÜR TARAMASI

1.2.1 Makine Öğrenmesi

Yapay Zekanın bir alt bilim dalı olan Makine Öğrenmesi, matematiksel ve istatistiksel işlemler ile veriler üzerinden çıkarımlar yaparak tahminlerde bulunan sistemlerin bilgisayarlar ile modellenmesidir. Günümüzde ML için birçok metodoloji ve algoritma mevcuttur. ML temelde öğrenme yöntemine göre üç gruba ayrılır; Öğretmenli(supervised), Öğretmensiz (Unsupervised) ve Reinforcement (Takviyeli) [2]

1.2.1.1 Öğretmeli Öğrenme (Supervised Learning)

Bu öğrenme tekniğinde modele verilen giriş verileri etiketlenmiştir. Modelin hangi girişe hangi çıkışı vereceği eğitim sırasında belirlenmiştir. Model, çalışırken bir bakıma verilen girişle eğitim verisi arasında eşleştirme yapar. [2]

Örneğin elimizde 10 fotoğraftan oluşan bir veri seti olsun. Bu fotoğraflardan 6 tanesi ördek fotoğrafı olsun ve "ördek" şeklinde etiketlensin. Kalan 4 fotoğrafı ise "ördek değil" şeklinde etiketleyerek modelimizi eğitelim. Modelimize bir ördek fotoğrafı verdiğimizde elindeki ördek fotoğrafları ile benzerlik bulacağından cevap olarak "ördek" değerini döndürür.

1.2.1.2 Öğretmensiz Öğrenme (Unsupervised Learning)

Bu öğrenme tekniğinde modele etiketsiz girdi görüntüleri verilerek modelin bu girdi görüntüleri arasında kendi kurallarına dayanarak sınıflandırma yapması beklenir. Mesela girdilerdeki verilerin birbirine olan uzaklığı, komşuluk ilişkileri, yoğunlukları, standart sapması bir kural olabilir. Eğitim için bir küme olmadığından model kendi kendine öğrenir. Daha çok sınıflandırma yapmak için kullanılır. [3]

Örneğin bir sosyal ağda tanıdığınız kişileri arkadaş ekliyorsunuz diyelim. Siz, sosyal ağ için girdilerden birisiniz. Sizin gibi birçok insan bu ağda arkadaşlarını kendi listesine ekliyor. Sosyal ağ sitesi kişiler arasında arkadaş grupları oluşturarak size "tanıyor olabileceğiniz kişiler" şeklinde arkadaş önerebilir.

1.2.1.3 Reinforcement (Pekiştirmeli) Learning

Pekiştirmeli öğrenme davranış psikolojisinden esinlenilen ve Ajan (Agent)'ların bir ortamda en yüksek ödül (Reward) miktarına ulaşabilmesi için hangi Politika (Policy)'i izlemesi gerektiğini araştıran bir makine öğrenmesi yaklaşımıdır. [4]

Örneğin bir bebek (Agent) sıcak bir şeye dokuyor ve eli yandığı için dokunduğu yerden elini çekiyor. Bu durumda bebek, çevresiyle olan etkileşimi ve çevreden geribildirim almasıyla öğrenmiş oluyor.

1.2.2 Sınıflandırma ve Regresyon

Sınıflama ve regresyon, önemli veri sınıflarını ortaya koyan veya gelecek veri eğilimlerini tahmin eden modelleri kurabilen analiz yöntemleridir. Sınıflama, kategorik değerleri tahmin ederken, regresyon, süreklilik gösteren değerlerin tahmin edilmesinde kullanılır. [5]

Örneğin, sınıflandırma modeli bir hastalığın varlığı veya yokluğunu sınıflandırırken, regresyon modeli ile bir evin, belirli bir dolar değeri için, belki de 100.000 ila 200.000 dolar aralığında satılacağı tahmin edilebilir. Sınıflandırma, veriyi, eğitim veri setindeki sınıflardan birisiyle eşlerken, regresyon gerçek değerli bir tahmini değişkene atar.

Sınıflama ve regresyon modellerinde kullanılan başlıca teknikler:

1. Yapay Sinir Ağları (Artificial Neural Networks)
2. Genetik Algoritmalar (Genetic Algorithms)
3. Random Forest (Rassal Orman)
4. Karar Ağaçları (Decision Trees)
5. Naive-Bayes sınıflayıcısı

1.2.2.1 Naive Bayes

Naive Bayes algoritması, belirli bir veri setindeki değerlerin frekansını ve kombinasyonlarını sayarak bir dizi olasılık hesaplayan basit bir olasılıksal sınıflandırıcıdır. Verilerdeki belirli bir özelliğin olasılığı, olasılıklar kümesinin bir üyesi olarak görünür ve

bir eğitim veri kümesinin bir sınıfı içindeki her bir özellik değerinin frekansı hesaplanarak türetilir. Eğitim veri kümesi, gelecekteki bilinmeyen değerleri tahmin etmek için bilinen değerleri kullanarak bir sınıflandırıcı algoritmasını eğitmek için kullanılan bir alt kümedir [6]

Algoritma Bayes teoremini kullanır ve sınıf değişkeninin değeri göz önüne alındığında tüm niteliklerin bağımsız olduğunu varsayar. Bu koşullu bağımsızlık varsayımı, gerçek dünya uygulamalarında nadiren geçerlidir. [7]

1.2.2.2 Karar Ağaçları

Karar ağaçları – sınıflama, özellik ve hedefe göre karar düğümleri (decision nodes) ve yaprak düğümlerinden (leaf nodes) oluşan ağaç yapısı formunda bir model oluşturan bir sınıflandırma yöntemidir. Karar ağacı algoritması, her bir dallanma ile veri setini daha küçük parçalara bölerek geliştirilir. Bir karar düğümü bir veya birden fazla dallanma içerebilir. İlk düğüme kök düğüm (root node) denir. Bir karar ağacı hem kategorik hem de sayısal verilerden oluşabilir. [8]

Kök düğümü genel anlamda veri setini en iyi sınıflandıran(feature) özellik olarak seçilir. Mesela bir sınıflandırma probleminde en önemli sınıfla ilgili değerlendirme kök düğüm olarak seçilebilir. Kök düğüm seçmek için bazı değerlere bakılır. Bunlardan bazıları Gini ve Entropidir. Buradaki fark Entropy daha dengeli bir ağaç çıkarmaya meyilli iken Gini, frekansı fazla olan sınıfı ayrıştırmaya meyillidir. [9]

Gini

$$Gini = 1 - \sum_{j} p_j^2$$

Şekil 1.1 Gini Formülü [9]

Pj: j sınıfının olasılığının gerçekleşme olasılığı

Gini değeri her sınıf için ayrı ayrı hesaplandıktan sonra 0'a en yakın sınıf kök düğüm olarak seçilir. Örneğin kanser hastalığıyla ilgili bir sınıflandırma yaparken veri setinde "hasta sigara içiyor mu?" sorusu 500 evet 100 hayır almış olsun. Evet cevabını veren 500 kişiden 436'sı kanser hastası 64'ü sağlıklı, hayır cevabı veren 100 kişiden 17'si kanser hastası 83'ü ise sağlıklı olsun.

Evet cevabı için olasılık 1-(436/500)^2 –(64/500)^2 = 0.223,

hayır cevabı için olasılık 1- (17/100)^2 – (83/100)^2 = 0.712 ise,

$Gini_{sigara}$ = (500/600) x 0.223 + (100/600) x 0.712 = 0.638 olarak bulunur. Aynı şekilde sınıflandırma problemindeki diğer sınıflar için Gini değeri hesaplanırsa 0'a en yakın düğüm kök düğüm olarak seçilir.

Entropy

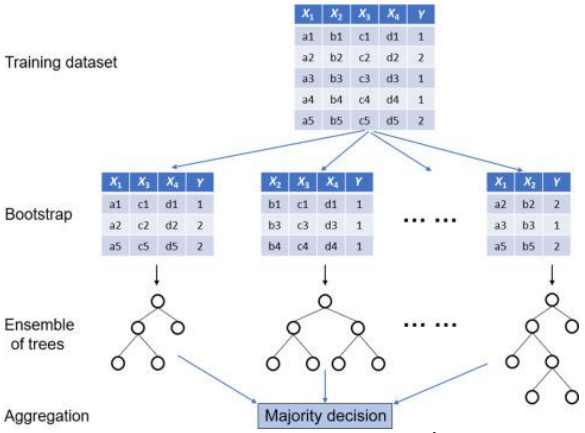
$$I_H = -\sum_{j=1}^{c} p_j log_2(p_j)$$

Şekil 1.2 Entropy Formülü [9]

Gini ile arasında çok bir fark olmayan entropy yöntemi neredeyse aynı işlemi log tabanında yapar.

1.2.2.3 Rassal Orman Yöntemi

Rassal Orman Yöntemi karar ağaçlarından topluluk oluşturan bir sınıflandırma ve regresyon metodudur. Rassal Orman Yöntemi, birçok karar ağacının bir araya geldiği toplulukta(orman), rastgele seçilen bir alt küme yardımı ile topluluklar oluşturmayı amaçlar. [10] Rassal Orman Yöntemi, farklı boyutlardaki ve tiplerdeki (Sınıflandırma ve Regresyon problemleri) veri setlerinde rahatlıkla kullanılabilmektedir. Şekil 1.3'de görüldüğü gibi oluşturulan alt kümelerden en çok oy küme sınıflandırma için seçilir.



Şekil 1.3 Rassal Orman Yöntemi İşleyişi [11]

Rassal Orman Sınıflandırıcı oluşturacağı alt toplulukların (karar ağaçlarının) sayısı (derinlik) ve bu karar ağaçlarında kullanılacak düğümlerin sayısını parametre alarak çalışır. [11]

Günlük hayattan bir örnek için elinizde "Yönetmen", "Başrol oyuncuları", "Görüntü Yönetmeni", "Sanat yönetmeni", "Müzikler", "Yapım yılı" vs.… Şeklinde bir film arşivi olsun ve bu arşivden film seçmek için arkadaşlarınızdan yardım istediğinizi düşünün. Her arkadaşınız bu arşivden belirli sorulara cevap vererek kendine göre bir film seçiyor olsun. Siz de arkadaşlarınız arasında en çok karar verilen yani en çok oy alan filmi seçtiğinizde Rassal Orman yapısına uygun bir seçim yapmış olursunuz. Burada kullanıcı olarak arkadaşlarınızın sayısını ve seçim yaparken seçtikleri kriter sayısı belirlemeniz gereken parametrelerdir.

1.2.3 Python ile Kullanıcı Arayüzü Tasarımı(GUI)

Python'un kullanıcı arayüzü tasarımı için kullanılabilecek kütüphaneleri Tkinter, kivy ve PyQt5'tir.Tkinter Python'un standart GUI geliştirme kütüphanelerinden birisidir. Genellikle başlangıç düzeyinde hızlı ve kolay uygulama geliştirmek için kullanılır. Basit uygulamalarda etkilidir. Kivy ise Android, Windows, Linux, Apple, Rasperry Pi gibi birçok işletim sistemini destekleyen bir diğer grafik arayüzü kütüphanesidir. Masaüstü ve mobil uygulamaları geliştirmekte kullanılır. Mobil uygulamalar konusunda diğer mobil uygulama geliştirme platformları kadar etkili değildir. Son olarak PyQt, çapraz platform uygulama geliştirmeye yarayan ve C++ ile yazılmış olan Qt kütüphanesinin Python bağlamasıdır. [12] Qt, grafik tasarıma yardımcı olarak Qt Designer'ı da içerisinde bulundurur. Qt Designer ile sürükle bırak şeklinde tasarlanan arayüzler Designer ile PyQt kodlarına çevrilir ve arayüz ile arkada işletilen kodlar birbirine entegre edilebilir.

2. METOD

2.1 Kalp Yetmezliği Tahmini

Bu çalışmada Rassal Orman'lar kullanılarak [13]'den alınan veri seti ile kalp yetmezliği olan hastalar tahmin edilmiştir. Tahminde kullanılan Rassal Orman modelinde karar ağaçlarının sayısı 100 olarak seçilmiştir. Her karar ağacı için maksimum derinlik 7 olarak belirlenmiştir. Hesaplama yöntemi olarak Gini seçilmiştir. Tahmin yapılırken kullanılan özellikler için aşağıdaki Tablo 1 incelenebilir.

Feature	**Açıklama**
age	Kalp yetmezliği olan kişinin yaşı
anaemia	Kırmızı kan hücrelerinin veya hemoglobinin azalması
creatinine_phosphokinase	Kandaki CPK enziminin seviyesi
diabetes	Hastanın diyabeti var mı
ejection_fraction	Her kasılmada kalpten çıkan kanın yüzdesi
high_blood_pressure	Hastanın yüksek tansiyonu var mı
platelets	Kandaki trombositler
serum_creatinine	Kandaki serum kreatinin düzeyi
serum_sodium	Kandaki serum sodyum seviyesi
sex	Hastanın cinsiyeti
smoking	Hasta sigara içiyor mu
Time	Hastanın gözlem süresi

Tablo 1 Feature açıklama tablosu

creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
582	0	20	1	265000.00	1.9	130	1	0	4	1
7861	0	38	0	263358.03	1.1	136	1	0	6	1
146	0	20	0	162000.00	1.3	129	1	1	7	1
111	0	20	0	210000.00	1.9	137	1	0	7	1
160	1	20	0	327000.00	2.7	116	0	0	8	1
...	...	...	...	...	...	...	...	...	...	...
133	0	60	1	219000.00	1.0	141	1	0	83	0
514	1	25	1	254000.00	1.3	134	1	0	83	0
59	0	60	0	255000.00	1.1	136	0	0	85	0
156	1	25	1	318000.00	1.2	137	0	0	85	0
61	1	40	0	221000.00	1.1	140	0	0	86	0

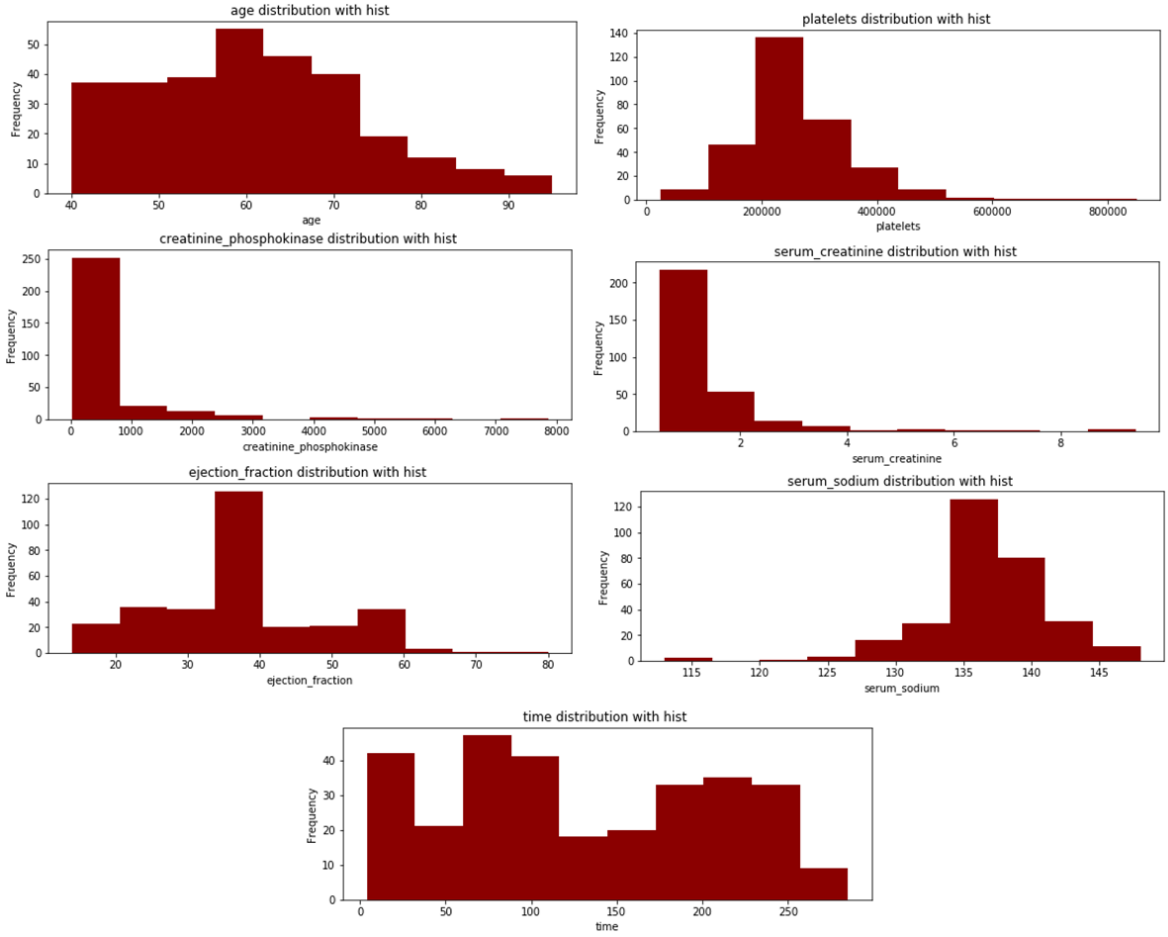
Şekil 2.1 Veri Setine Genel Bir Bakış

Şekil 2.1'de bir kısmı gözüken veri seti ile eğiteceğimiz modelde 12 farklı değişken ve bir tane de hedef değişkenimiz mevcut. (DEATH_EVENT) Veri setiyle ilgili detaylı bilgileri şekil 2.2'de görebiliriz. Şekil 2.3 ve 2.4'de kategorik ve nümerik verilerin dağılımını görebiliriz.

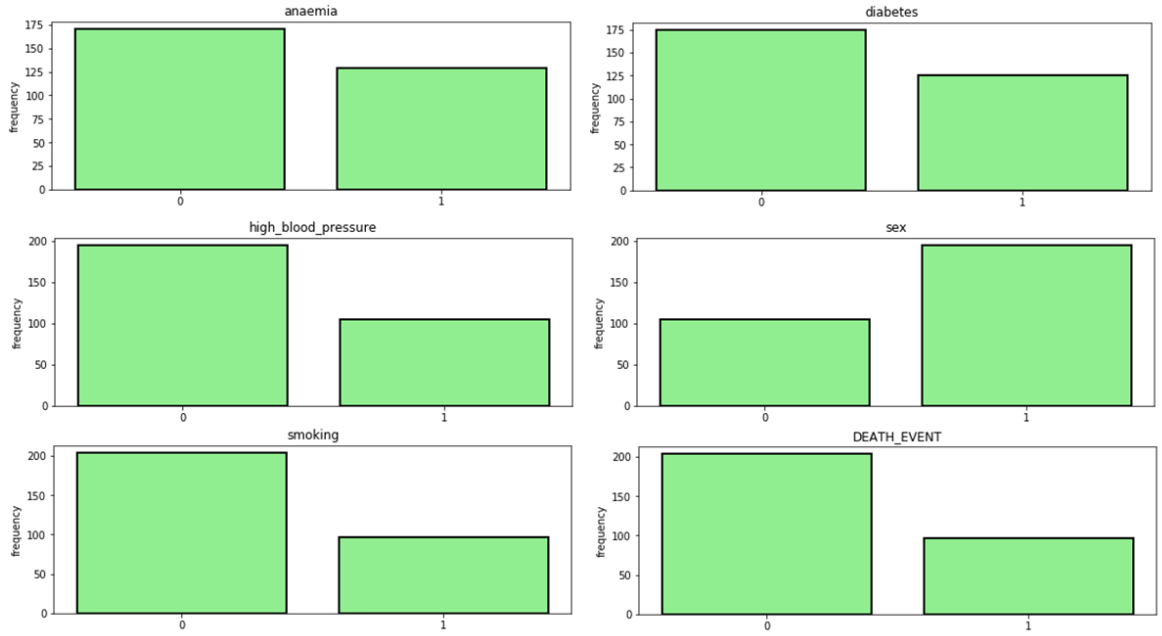
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       299 non-null    float64
 1   anaemia                   299 non-null    int64
 2   creatinine_phosphokinase  299 non-null    int64
 3   diabetes                  299 non-null    int64
 4   ejection_fraction         299 non-null    int64
 5   high_blood_pressure       299 non-null    int64
 6   platelets                 299 non-null    float64
 7   serum_creatinine          299 non-null    float64
 8   serum_sodium              299 non-null    int64
 9   sex                       299 non-null    int64
 10  smoking                   299 non-null    int64
 11  time                      299 non-null    int64
 12  DEATH_EVENT               299 non-null    int64
dtypes: float64(3), int64(10)
memory usage: 30.5 KB
```

Şekil 2.2 Veri Seti Detayları



Şekil 2.3 Nümerik Değişkenlerin Dağılımı



Şekil 2.4 Kategorik Değişkenlerin Dağılımı

Şekil 2.5 incelendiğinde nümerik değişkenlerin bazılarında verinin normal dağılımından aykırı değerler görülmektedir. Bu aykırı değerler veri setindeki değerler Tukey's yöntemi ile taranarak veri setinden çıkarılmıştır.

```
def detect_outliers(df,features):
    outlier_indices = []

    for c in features:
        # 1st quartile
        Q1 = np.percentile(df[c],25)
        # 3st quartile
        Q3 = np.percentile(df[c],75)
        # IQR
        IQR = Q3 - Q1
        # Outlier Step
        outlier_step = IQR * 1.5
        # detect outlier and their indeces
        outlier_list_col = df[(df[c] < Q1 - outlier_step) | (df[c] > Q3 + outlier_step)].index
        # store indeces
        outlier_indices.extend(outlier_list_col)

    outlier_indices = Counter(outlier_indices)
    multiple_outliers = list(i for i, v in outlier_indices.items() if v > 1)

    return multiple_outliers
```

```
data.loc[detect_outliers(data,["age","creatinine_phosphokinase","ejection_fraction",
                                                                "platelets","serum_creatinine","serum_sodium","time"])]
```

```
data = data.drop(detect_outliers(data,["age","creatinine_phosphokinase","ejection_fraction",
                                    "platelets","serum_creatinine","serum_sodium","time"]),axis = 0).reset_index(drop=True)
```

Şekil 2.5 Tukey's yöntemi ile anormallik tespiti

Son olarak normal dağılmayan değişkenler box-cox metodu kullanırak normalize edilmiştir. Öncesi ve sonrası şeklinde normalize edilen veriler şekil 2.7'de gösterilmiştir. Şekil 2.6'da görüldüğü gibi hedef değişkendeki dengesizlik yeniden örnekleme yöntemlerinden SMOTE ile giderilmiştir.

```
X = data.drop("DEATH_EVENT", axis = 1)
y = data.DEATH_EVENT
```

```
print("Before Smote")
y.value_counts()
```

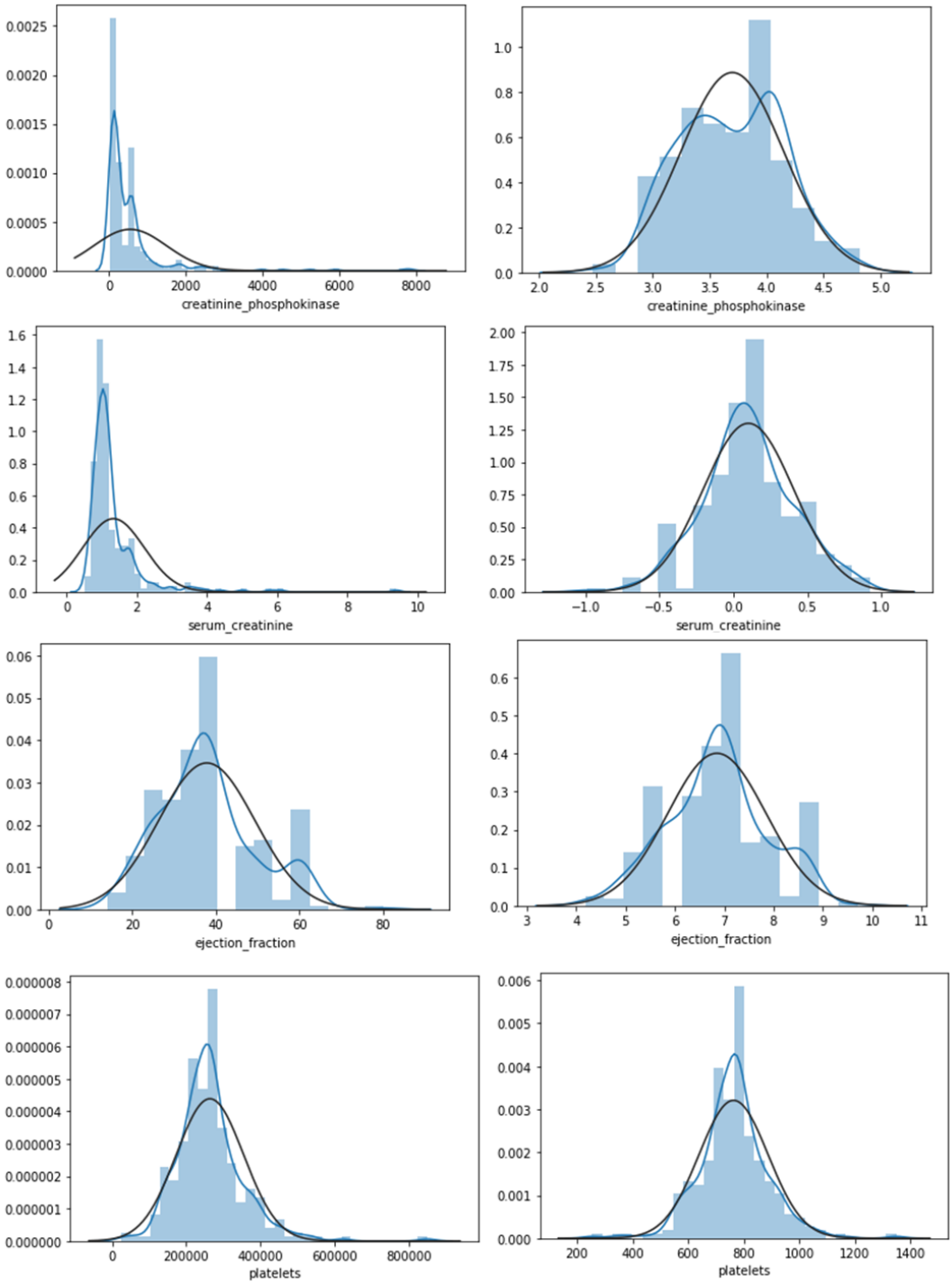
```
Before Smote

0    198
1     91
Name: DEATH_EVENT, dtype: int64
```

```
sm = SMOTE(random_state=42)
X_sm, y_sm = sm.fit_resample(X, y)
```

```
print("After Smote")
y_sm.value_counts()
```

```
After Smote

1    198
0    198
Name: DEATH_EVENT, dtype: int64
```

Şekil 2.6 SMOTE yöntemi ile Yeniden Örnekleme yapılması



Şekil 2.7 Box-cox yöntemi ile normalize edilen değişkenler

2.2 Arayüz Tasarımı

Çalışmada kullanıcıdan bir dizi değer istendiği için Qt sınıfından form objesi kullanılarak bir form objesi oluşturuldu. Bu form objesinde nümerik değerler Label'lar ile kategorik değerler ise radio button ile alındı. Her değişken için oluşturulan label ve radio button komponentleri form objesiyle birleştirildi. Şekil 2.8'de cinsiyet değişkeni için bir örnek verilmiştir.

```
self.Radmale   = QRadioButton("Male")
self.Radfemale = QRadioButton("Female")

self.sexBgroup = QButtonGroup()

self.sexBgroup.addButton(self.Radmale)
self.sexBgroup.addButton(self.Radfemale)

self.Radmale.toggled.connect(self.toggleRadio)
self.Radfemale.toggled.connect(self.toggleRadio)

sexLayout = QHBoxLayout()
sexLayout.addWidget(self.Radmale)
sexLayout.addStretch()
sexLayout.addWidget(self.Radfemale)

form.addRow(QLabel("Sex"),sexLayout)
```

Şekil 2.8 Örnek bir RadioButton kullanımı

Şekil 2.9'da görüldüğü gibi her Line Edit'e imleç üzerine getirildiğinde girilecek değer hakkında birim vermek ve açıklama yapmak için tooltip objesi eklendi.

```
self.age = QLineEdit()
self.Time = QLineEdit()
self.Time.setToolTip('Patient observation period: enter in days')
self.CPK = QLineEdit()
self.CPK.setToolTip('Level of the CPK enzyme in the blood (mcg/L)')
self.SerumSodium = QLineEdit()
self.SerumSodium.setToolTip(' Level of serum sodium in the blood (mEq/L)')
self.SerumCreatinine = QLineEdit()
self.SerumCreatinine.setToolTip('Level of serum creatinine in the blood (mg/dL)')
self.Platelets = QLineEdit()
self.Platelets.setToolTip('Platelets in the blood (kiloplatelets/mL)')
self.EjectionFraction = QLineEdit()
self.EjectionFraction.setToolTip('Percentage of blood leaving the heart at each contraction (percentage)')
```

Şekil 2.9 LineEdit ve ToolTip

```
Result = QPushButton(self)
Result.setText("Predict")
Result.clicked.connect(self.Prediction)
```

Şekil 2.10 PushButton kullanımı

Alınan değişkenleri eğitilen modelden geçirerek sonucu ekrana yazdırmak için "Predict" adında bir buton eklendi. Buton Event'i olarak Random Forest metodunu çalıştıran Prediction fonksiyonuna yönlendirildi. (Şekil 2.10)

```
msgBox = QMessageBox()
msgBox.setText(Result)
msgBox.setWindowTitle("Result")
msgBox.setDetailedText(Detail)
restartBtn = msgBox.addButton('Retry', QMessageBox.ActionRole)
ExitBtn    = msgBox.addButton('Exit', QMessageBox.ActionRole)


ret = msgBox.exec()

if ret == QMessageBox.Close:
          QCloseEvent()
elif msgBox.clickedButton() == restartBtn:
    self.age.clear()
    self.Time.clear()
    self.CPK.clear()
    self.SerumSodium.clear()
    self.SerumCreatinine.clear()
    self.Platelets.clear()
    self.EjectionFraction.clear()
elif msgBox.clickedButton() == ExitBtn:
    self.close()
```

Şekil 2.11 MessageBox kullanımı

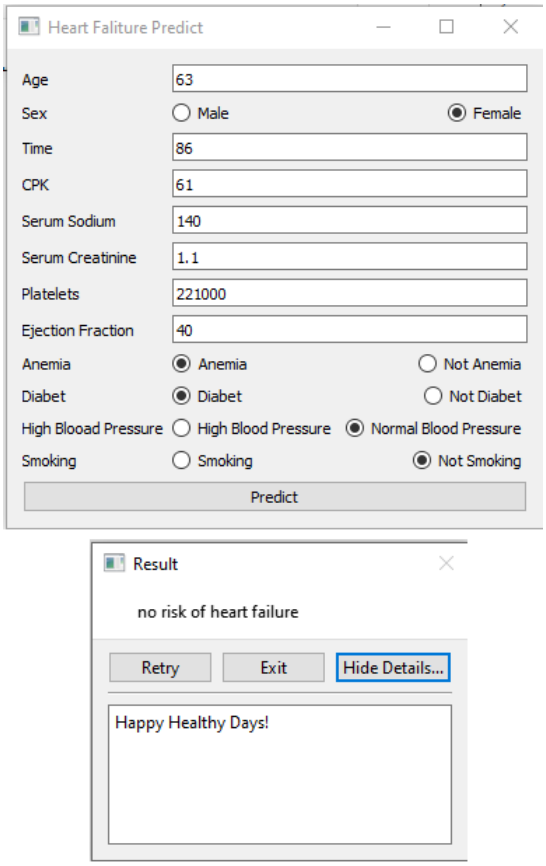
Program tahmini yaptıktan sonra ekrana yazdırmak için QMessageBox sınıfı kullanıldı ve Programı tekrarlamak için "Retry", Çıkmak için "Exit" yapılan tahmin hakkında detay öğrenmek için ise "Show Details" butonları eklendi. Şekil 3.4'te butonlara atanan Event'ler gösterilmektedir. (Şekil 2.11)

2.3 Kalp Yetmezliği Tahmin Modelinin Arayüze Eklenmesi

Hazırlanan veri seti, Rassal Orman Metodu ile %20'si test %80'i eğitim verisi olmak üzere ayrılarak eğitildi. Eğitim sonunda modelin eğitiminde etkili olan değişkenler belirlendi ve bu değişkenler eğitim ve test verisinden seçilerek ikinci bir eğitimle modelin başarısı saptandı. İlk modelde accuracy score %91 bulunurken, ikinci modelde %97 olarak saptandı. (Şekil 3.2 ve 3.3) Eğitilen son model pickle kütüphanesi aracılığı ile dışarı aktarıldı ve PyQt5 ile hazırlanan arayüze eklendi. (Şekil 2.12). Şekil 2.13'te programın son hali ile çalışma şeklini görebiliriz.

```
X_train, X_test, Y_train, Y_test = train_test_split(X_sm, y_sm, test_size = 0.2, random_state = 42)


model_rnd = RandomForestClassifier()
model_rnd.fit(X_train, Y_train)
importance = model_rnd.feature_importances_


x_train_random_forest = X_train[["age","creatinine_phosphokinase","ejection_fraction","serum_creatinine","time"]]
x_test_random_forest = X_test[["age","creatinine_phosphokinase","ejection_fraction","serum_creatinine","time"]]


random_forest_model = RandomForestClassifier(max_depth=7, random_state=25)
random_forest_model.fit(x_train_random_forest, Y_train)
y_pred_random_forest = random_forest_model.predict(x_test_random_forest)
cm_random_forest = confusion_matrix(y_pred_random_forest, Y_test)
acc_random_forest = accuracy_score(Y_test, y_pred_random_forest)

pickle.dump(model_rnd, open('model.pkl','wb'))

model = pickle.load(open('model.pkl','rb'))

DEATH_EVENT=model.predict([[float(self.age.text()),
                            self.anemia,
                          float(self.CPK.text()),
                            self.diabet,
                          float(self.EjectionFraction.text()),
                            self.highbloodpressure,
                          float(self.Platelets.text()),
                          float(self.SerumCreatinine.text()),
                          float(self.SerumSodium.text()),
                            self.sex,
                            self.smoking,
                          float(self.Time.text()) ]])

if DEATH_EVENT[0]== 0:
    Result =  "no risk of heart failure"
    Detail =  "Happy Healthy Days!"
else:
    Result =  "high risk of heart failure"
    Detail = InfoCPK + InfoEF + InfoAge + InfoSC + InfoTime
```

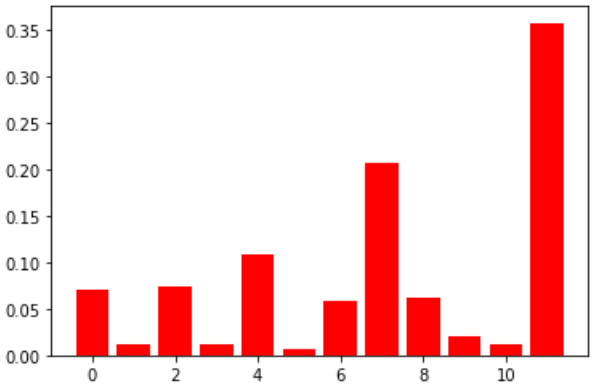
Şekil 2.12 Random Forest Metodunu kullanma



Şekil 2.13 Program Çalışma Şekli ve Çıktılar

3. SONUÇ

Çalışmada kalp yetmezliği tahmini için kullanılan veri setinde yaş, creatinine phosphokinase, ejection fraction, serum creatine ve zaman değişkenlerinin diğer değişkenlere nazaran kalp yetmezliğini daha çok etkilediği görülmüştür.

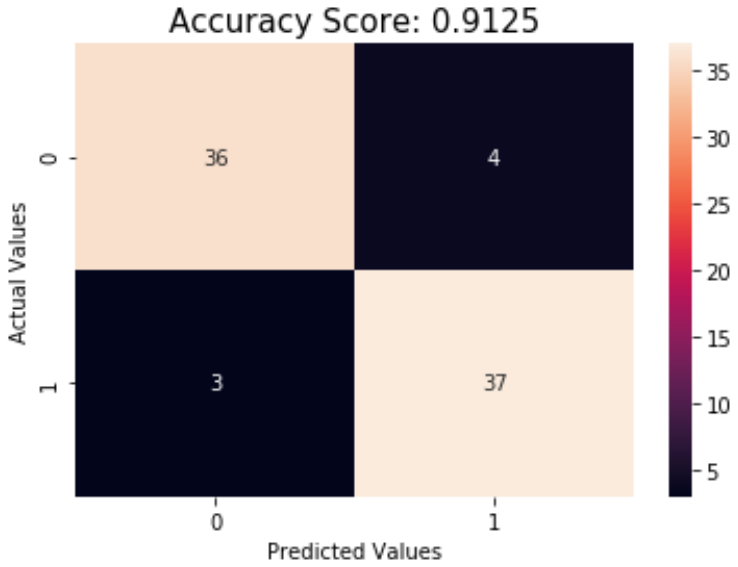


Şekil 3.1 Model Importance Sonuçları

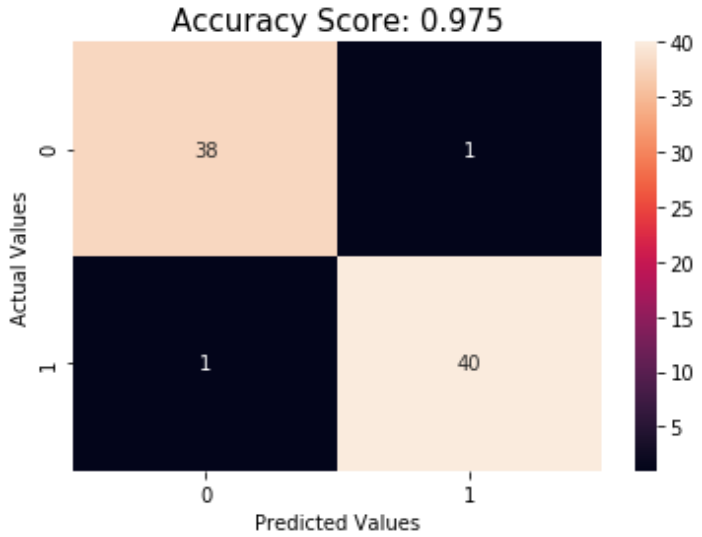
Numara	Feature
0	age
1	anaemia
2	creatinine_phosphokinase
3	diabetes
4	ejection_fraction
5	high_blood_pressure
6	platelets
7	serum_creatinine
8	serum_sodium
9	sex
10	smoking
11	Time

Tablo 2 Importance Sonuçları için açıklama tablosu

Model eğitilirken iki farklı eğitimden geçirilmiştir. İlk eğitimde elde edilen importance değerlerine göre (Şekil 3.1) ikinci eğitimde bu değişkenler baz alınarak eğitim başarısı arttırılmıştır.



Şekil 3.2 İlk Modeldeki başarı Değerleri



Şekil 3.3 İkinci Modeldeki Başarı Değerleri

4. TARTIŞMA

Bu çalışma hastanelerde her gün tedavi olan veya teşhis için başvuran binlerce hastadan elde edilen verilerin gerekli metodlarla işlenerek depolandığında ve bu veriler farklı amaçlara yönelik farklı algoritmalarla işlendiğinde doktorlar veya hastalar için anlamlı sonuçların ortaya çıkabileceğini göstermiştir.

Daha geniş kapsamda tedavi sonrasında riski bulunan hastalar için tahminler yapılarak riski olan hastaların taburcu edilmeden bir süre daha gözetim altında tutularak hastaların beklenmeyen komplikasyonları engellenebilir.

Bu tür makine öğrenmesi yöntemleri ile geliştirilen farklı algoritmalar hastanelerin veri tabanına gömülerek her yapılan tahlilden sonra ilgili algoritmaların sonuçları ile otomatik tahminler döndürülebilir. Örneğin herhangi bir hasta herhangi bir hastalık için kan tahlili yaptırdığında kanındaki değeler otomatik olarak değerlendirilip ilgili hastalığın yanında başka hastalıklar adına da hasta uyarılabilir. Zamanla elde edilen güvenilir algoritmalar ile bu yöntemler standart haline getirilerek erken teşhis ile çoğu hastalık etkisini göstermeden durdurulabilir.

KAYNAKLAR

[1] TÜRK KARDİYOLOJİ DERNEĞİ,Kalp Yetersizliği Çalışma Grubu, «KALP YETERSİZLİĞİ NEDİR?,» [Çevrimiçi]. Available: https://tkd.org.tr/kalp-yetersizligi-calisma-grubu/sayfa/toplum_icin_bilgiler#a4. [Erişildi: 25 Temmuz 2021].

[2] H. İ. ŞAFAK, «Makine Öğrenmesi Nedir ?,» 9 aralık 2017. [Çevrimiçi]. Available: https://medium.com/türkiye/makine-öğrenmesi-nedir-20dee450b56e. [Erişildi: 30 mart 2021].

[3] T. T. R. &. F. J. Hastie, Unsupervised learning. In The elements of statistical learning, New York: Springer, 2009.

[4] B. S. a. A. Z. O. Chapelle, «Semi-supervised learning,» *IEEE TRANSACTIONS ON NEURAL NETWORKS,* cilt 3, no. 20, pp. 542-542, 2009.

[5] H. L. Siddharth Misra, Machine Learning for Subsurface Characterization, 2020.

[6] J. A. A. a. C. M. J. George Dimitoglou, *Comparison of the C4.5 and a Naive Bayes Classifier for the Prediction of Lung Cancer Survivability,* 2012.

[7] M. S. S. S. Tina R. Patil, «Performance Analysis of Naive Bayes and J48 Classification Algorithm for Data Classification,» *International Journal Of Computer Science And Applications,* pp. 256-261, Nisan 2013.

[8] S. Sayad, «saedsayad.com,» 2021. [Çevrimiçi]. Available: http://www.saedsayad.com/decision_tree.htm.

[9] M. F. Akça, «Deep-learning-türkiye-Karar Ağaçları,» 7 ocak 2020. [Çevrimiçi]. Available: https://medium.com/deep-learning-turkiye/karar-ağaçları-makine-öğrenmesi-serisi-3-a03f3ff00ba5#:~:text=Karar%20ağaçları%2C%20Sınıflandırma%20ve%20Regresyon,Karmaşık%20veri%20setlerinde%20kullanılabilir..

[10] L. BREIMAN, «Random Forests,» *2001 Kluwer Academic Publishers,* pp. 5-32, 2001.

[11] S. K. T. Ç. Aslı ÇALIŞ, «VERİ MADENCİLİĞİNDE KARAR AĞACI ALGORİTMALARI İLE BİLGİSAYAR VE İNTERNET GÜVENLİĞİ ÜZERİNE BİR UYGULAMA,» *Endüstri Mühendisliği Dergisi,* pp. 2-19, 10 Eylül 2014.

[12] «cozumPedia,» 2021. [Çevrimiçi]. Available: https://cozumpedia.com/pyqt-nedir.

[13] D. C. &. G. Jurman, «Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone,» *Research Article,* 3 Şubat 2020.

EKLER

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
import scipy

from scipy import stats
from scipy.stats import norm, skew, boxcox
from collections import Counter

from sklearn.preprocessing import RobustScaler, StandardScaler
from sklearn.metrics import mean_squared_error, confusion_matrix, accuracy_score, plot_confusion_matrix, auc
from sklearn.base import BaseEstimator, TransformerMixin, RegressorMixin, clone

from sklearn.model_selection import train_test_split, StratifiedKFold, GridSearchCV
from sklearn.linear_model import LogisticRegression
from catboost import CatBoostClassifier, Pool
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_selection import RFE
from imblearn.over_sampling import SMOTE

import pickle

#XGBOOST
from xgboost import XGBClassifier

#warning
import warnings
warnings.filterwarnings('ignore')

import sys
from PyQt5.QtWidgets import *
from PyQt5.QtCore    import *
from PyQt5.QtGui     import *


class MainPage(QWidget):
    def __init__(self):
        super().__init__()
        self.setupUi()
        self.setWindowIcon(QIcon("ICON.png"))
        self.setWindowTitle("Heart Faliture Predict")


    def setupUi(self):

        Result = QPushButton(self)
        Result.setText("Predict")
        Result.clicked.connect(self.Prediction)

        form   = QFormLayout()
        window = QWidget()

        self.Radmale   = QRadioButton("Male")
        self.Radfemale = QRadioButton("Female")

        self.sexBgroup = QButtonGroup()

        self.sexBgroup.addButton(self.Radmale)
        self.sexBgroup.addButton(self.Radfemale)

        self.Radmale.toggled.connect(self.toggleRadio)
        self.Radfemale.toggled.connect(self.toggleRadio)

        sexLayout = QHBoxLayout()
        sexLayout.addWidget(self.Radmale)
        sexLayout.addStretch()
        sexLayout.addWidget(self.Radfemale)

        self.Raddiabet  = QRadioButton("Diabet")
        self.Radnotdiabet = QRadioButton("Not Diabet")

        self.diabetBgroup = QButtonGroup()

        self.diabetBgroup.addButton(self.Raddiabet)
        self.diabetBgroup.addButton(self.Radnotdiabet)

        self.Raddiabet.toggled.connect(self.toggleRadio)
        self.Radnotdiabet.toggled.connect(self.toggleRadio)

        diabetLayout = QHBoxLayout()
        diabetLayout.addWidget(self.Raddiabet)
        diabetLayout.addStretch()
        diabetLayout.addWidget(self.Radnotdiabet)

        self.RadHighBloodPressure = QRadioButton("High Blood Pressure")
        self.RadNormalBloodPressure  = QRadioButton("Normal Blood Pressure")

        self.HighBloodPressureBgroup = QButtonGroup()
```

```
        self.HighBloodPressureBgroup.addButton(self.RadHighBloodPressure)
        self.HighBloodPressureBgroup.addButton(self.RadNormalBloodPressure)

        self.RadHighBloodPressure.toggled.connect(self.toggleRadio)
        self.RadNormalBloodPressure.toggled.connect(self.toggleRadio)

        HighBloodPressureLayout = QHBoxLayout()
        HighBloodPressureLayout.addWidget(self.RadHighBloodPressure)
        HighBloodPressureLayout.addStretch()
        HighBloodPressureLayout.addWidget(self.RadNormalBloodPressure)

        self.RadSmoking  = QRadioButton("Smoking")
        self.RadnotSmoking = QRadioButton("Not Smoking")

        self.SmokingBgroup = QButtonGroup()

        self.SmokingBgroup.addButton(self.RadSmoking)
        self.SmokingBgroup.addButton(self.RadnotSmoking)

        self.RadSmoking.toggled.connect(self.toggleRadio)
        self.RadnotSmoking.toggled.connect(self.toggleRadio)

        SmokingLayout = QHBoxLayout()
        SmokingLayout.addWidget(self.RadSmoking)
        SmokingLayout.addStretch()
        SmokingLayout.addWidget(self.RadnotSmoking)

        self.RadAnemia  = QRadioButton("Anemia")
        self.RadnotAnemia = QRadioButton("Not Anemia")

        self.AnemiaBgroup = QButtonGroup()

        self.AnemiaBgroup.addButton(self.RadAnemia)
        self.AnemiaBgroup.addButton(self.RadnotAnemia)

        self.RadAnemia.toggled.connect(self.toggleRadio)
        self.RadnotAnemia.toggled.connect(self.toggleRadio)

        AnemiaLayout = QHBoxLayout()
        AnemiaLayout.addWidget(self.RadAnemia)
        AnemiaLayout.addStretch()
        AnemiaLayout.addWidget(self.RadnotAnemia)

        self.age = QLineEdit()
        self.Time = QLineEdit()

        self.Time.setToolTip('Patient observation period: enter in days')
        self.CPK = QLineEdit()
        self.CPK.setToolTip('Level of the CPK enzyme in the blood (mcg/L)')
        self.SerumSodium = QLineEdit()
        self.SerumSodium.setToolTip(' Level of serum sodium in the blood (mEq/L)')
        self.SerumCreatinine = QLineEdit()
        self.SerumCreatinine.setToolTip('Level of serum creatinine in the blood (mg/dL)')
        self.Platelets = QLineEdit()
        self.Platelets.setToolTip('Platelets in the blood (kiloplatelets/mL)')
        self.EjectionFraction = QLineEdit()
        self.EjectionFraction.setToolTip('Percentage of blood leaving the heart at each contraction (percentage)')


        form.addRow(QLabel("Age"),self.age)
        form.addRow(QLabel("Sex"),sexLayout)
        form.addRow(QLabel("Time"),self.Time)
        form.addRow(QLabel("CPK"),self.CPK)
        form.addRow(QLabel("Serum Sodium"),self.SerumSodium)
        form.addRow(QLabel("Serum Creatinine"),self.SerumCreatinine)
        form.addRow(QLabel("Platelets"),self.Platelets)
        form.addRow(QLabel("Ejection Fraction"),self.EjectionFraction)
        form.addRow(QLabel("Anemia"),AnemiaLayout)
        form.addRow(QLabel("Diabet"),diabetLayout)
        form.addRow(QLabel("High Blood Pressure"),HighBloodPressureLayout)
        form.addRow(QLabel("Smoking"),SmokingLayout)
        form.addRow(Result)


        self.setLayout(form)

        self.show()

    def Prediction(self):
        InfoAge  = "zzz zzz zzz\n"
        InfoCPK  = "bla bla bla\n"
        InfoEF   = "tada tada\n"
        InfoSC   = "mrr mrr mrr mrr\n"
        InfoTime = "ring ring ring\n"

        def detect_outliers(df,features):

            outlier_indices = []

            for c in features:
                # 1st quartile
                Q1 = np.percentile(df[c],25)
```

```
        # 3st quartile
        Q3 = np.percentile(df[c],75)
        # IQR
        IQR = Q3 - Q1
        # Outlier Step
        outlier_step = IQR * 1.5
        # detect outlier and their indeces
        outlier_list_col = df[(df[c] < Q1 - outlier_step) | (df[c] > Q3 + outlier_step)].index
        # store indeces
        outlier_indices.extend(outlier_list_col)

    outlier_indices = Counter(outlier_indices)
    multiple_outliers = list(i for i, v in outlier_indices.items() if v > 1)

    return multiple_outliers

data = pd.read_csv("heart_failure_clinical_records_dataset.csv")

data.loc[detect_outliers(data,["age","creatinine_phosphokinase","ejection_fraction",
                                                    "platelets","serum_creatinine","serum_sodium","time"])]

data = data.drop(detect_outliers(data,["age","creatinine_phosphokinase","ejection_fraction",
                                "platelets","serum_creatinine","serum_sodium","time"]),axis = 0).reset_index(drop=True)

skewed_feats = data.apply(lambda x: skew(x.dropna())).sort_values(ascending = False) #Çarpıklık kontrolü
skewness = pd.DataFrame(skewed_feats, columns = ["skewed"])


data["creatinine_phosphokinase"], lam = boxcox(data["creatinine_phosphokinase"])
data["serum_creatinine"], lam_serum_creatine = boxcox(data["serum_creatinine"])
data["ejection_fraction"], lam_serum_creatine = boxcox(data["ejection_fraction"])
data["platelets"], lam_serum_creatine = boxcox(data["platelets"])


skewed_feats = data.apply(lambda x: skew(x.dropna())).sort_values(ascending = False)
skewness_new = pd.DataFrame(skewed_feats, columns = ["skewed"])

X = data.drop("DEATH_EVENT", axis = 1)
y = data.DEATH_EVENT

sm = SMOTE(random_state=42)
X_sm, y_sm = sm.fit_resample(X, y)


X_train, X_test, Y_train, Y_test = train_test_split(X_sm, y_sm, test_size = 0.2, random_state = 42)


model_rnd = RandomForestClassifier()
model_rnd.fit(X_train, Y_train)
importance = model_rnd.feature_importances_


x_train_random_forest = X_train[["age","creatinine_phosphokinase","ejection_fraction","serum_creatinine","time"]]
x_test_random_forest = X_test[["age","creatinine_phosphokinase","ejection_fraction","serum_creatinine","time"]]


random_forest_model = RandomForestClassifier(max_depth=7, random_state=25)
random_forest_model.fit(x_train_random_forest, Y_train)
y_pred_random_forest = random_forest_model.predict(x_test_random_forest)
cm_random_forest = confusion_matrix(y_pred_random_forest, Y_test)
acc_random_forest = accuracy_score(Y_test, y_pred_random_forest)

pickle.dump(model_rnd, open('model.pkl','wb'))

model = pickle.load(open('model.pkl','rb'))

DEATH_EVENT=model.predict([[float(self.age.text()),
                            self.anemia,
                        float(self.CPK.text()),
                            self.diabet,
                        float(self.EjectionFraction.text()),
                            self.highbloodpressure,
                        float(self.Platelets.text()),
                        float(self.SerumCreatinine.text()),
                        float(self.SerumSodium.text()),
                            self.sex,
                            self.smoking,
                        float(self.Time.text()) ]])

if DEATH_EVENT[0]== 0:
    Result =  "no risk of heart failure"
    Detail =  "Happy Healthy Days!"
else:
    Result =  "high risk of heart failure"
    Detail = InfoCPK + InfoEF + InfoAge + InfoSC + InfoTime



msgBox = QMessageBox()
msgBox.setText(Result)
```

```
        msgBox.setWindowTitle("Result")
        msgBox.setDetailedText(Detail)
        restartBtn = msgBox.addButton('Retry', QMessageBox.ActionRole)
        ExitBtn    = msgBox.addButton('Exit', QMessageBox.ActionRole)


        ret = msgBox.exec()

        if ret == QMessageBox.Close:
                QCloseEvent()
        elif msgBox.clickedButton() == restartBtn:
            self.age.clear()
            self.Time.clear()
            self.CPK.clear()
            self.SerumSodium.clear()
            self.SerumCreatinine.clear()
            self.Platelets.clear()
            self.EjectionFraction.clear()
        elif msgBox.clickedButton() == ExitBtn:
            self.close()



    def toggleRadio(self):

        rdButon=self.sender()

        if rdButon.isChecked():

            if rdButon.text() == 'Male':
                self.sex = 1

            elif rdButon.text() == 'Female':
                self.sex = 0

            if rdButon.text() == 'Diabet':
                self.diabet = 1

            elif rdButon.text() == 'Not Diabet':
                self.diabet = 0

            if rdButon.text() == 'High Blood Pressure':
                self.highbloodpressure = 1

            elif rdButon.text() == 'Normal Blood Pressure':
                self.highbloodpressure = 0

            if rdButon.text() == 'Smoking':
                self.smoking = 1

            elif rdButon.text() == 'Not Smoking':
                self.smoking = 0

            if rdButon.text() == 'Anemia':
                self.anemia = 1

            elif rdButon.text() == 'Not Anemia':
                self.anemia = 0


def main():
    app = QApplication(sys.argv)
    mainPage = MainPage()
    sys.exit(app.exec())

if __name__ == "__main__":
    main()
```