Makine Öğrenmesi Algoritmaları Kullanılarak Sepsis Hastalığının Teşhisi

2024
BİLGİSAYAR MÜHENDİSLİĞİ
BİTİRME PROJESİ TEZİ

Yusuf ŞAHİN

Muhammet KAYA

Makine Öğrenmesi Algoritmaları Kullanılarak Sepsis Hastalığının Teşhisi

Yusuf ŞAHİN
Muhammet KAYA

Karabük Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümünde
Bitirme Projesi Tezi
Olarak Hazırlanmıştır.

KARABÜK
HAZİRAN 2024

Muhammet KAYA ve Yusuf ŞAHİN tarafından hazırlanan “Makine Öğrenmesi Algoritmaları Kullanılarak Sepsis Hastalığının Teşhisi” başlıklı bu projenin Bitirme Projesi Tezi olarak uygun olduğunu onaylarım.

Dr. Öğr. Üyesi Nesrin AYDIN ATASOY .....
Bitime Projesi Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

...../…../2024

Bilgisayar Mühendisliği bölümü, bu tez ile, Bitirme Projesi Tezini onamıştır

Prof. Dr. Oğuz FINDIK .....
Bölüm Başkanı

“Bu projedeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim.”

Yusuf ŞAHİN

Muhammet KAYA

ÖZET

Bitime Projesi Tezi

Makine Öğrenmesi Algoritmaları Kullanılarak Sepsis Hastalığının Teşhisi

Yusuf ŞAHİN
Muhammet KAYA

Karabük Üniversitesi
Bilgisayar Mühendisliği
Bilgisayar Mühendisliği Bölümü

Tez Danışmanı:
Dr. Öğr. Üyesi Nesrin AYDIN ATASOY
Haziran 2024

Kan zehirlenmesi olarak da bilinen sepsis hastalığı tek veya birden fazla organ yetmezliğine kadar ilerleyebilen ve hatta ölümle sonuçlanabilen ciddi bir hastalıktır. Makine öğrenmesi algoritmaları tıp alanında birçok hastalığın teşhisi için uzun yıllardır kullanılıyor olsa da sepsis teşhisi için daha önce yapılmış çalışma sayısı oldukça sınırlıdır. Oldukça büyük bir veri setinin kullanıldığı bu projede birçok güncel ve popüler makine öğrenimi algoritması kullanıldı ve sonuçları karşılaştırıldı. Bu projeyle birlikte, sepsis hastalığının erken teşhisi konusundaki bilimsel ve akademik anlayışın derinleşmesi ve hastalığın erken teşhisindeki iyileştirmeler sayesinde sağlık hizmetlerinin verimliliğinin artması öngörülmektedir.

Anahtar kelimeler: Makine Öğrenmesi, Sepsis

ABSTRACT

Senior Projcet Thesis

Diagnosis of Sepsis Disease Using Machine Learning Algorithms

Yusuf ŞAHİN
Muhammet KAYA

Karabük University
Faculty of Engineering
Department of Computer Engineering

Project Supervisor:
Assistant Professor Nesrin AYDIN ATASOY

June 2024

Sepsis, also known as blood poisoning, is a serious condition that can progress to single or multiple organ failure and even death. Although machine learning algorithms have been used in the medical field for the diagnosis of various diseases for many years, the number of studies specifically focused on sepsis diagnosis is quite limited. In this project, numerous current and popular machine learning algorithms were employed using a large dataset, and their results were compared. This project aims to deepen the scientific and academic understanding of early sepsis diagnosis and to improve healthcare efficiency through enhancements in the early detection of the disease.

Keywords: Machine Learning, Sepsis

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde, oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığımız, yönlendirme ve bilgilendirmeleriyle çalışmamızı bilimsel temeller ışığında şekillendiren sayın hocamız Dr. Öğr. Üyesi Nesrin AYDIN ATASOY'a sonsuz teşekkürlerimizi sunarız.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

BÖLÜM 1

SEPSİS HASTALIĞI

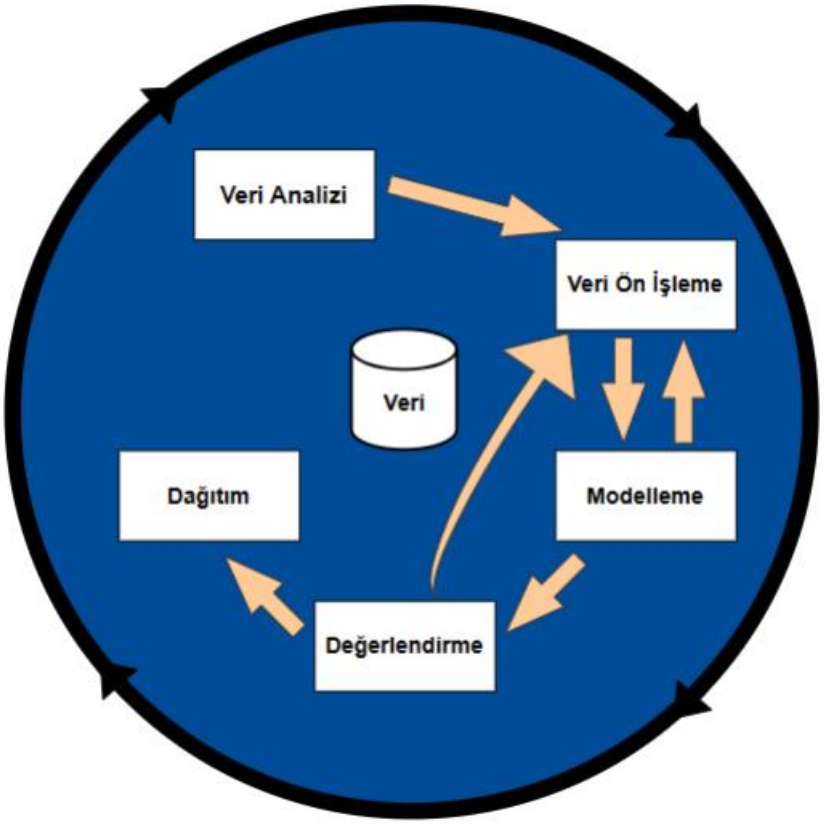
Vücutta meydana gelen şiddetli enfeksiyon sonucunda bağışıklık sisteminin aşırı tepkisiyle organ ve dokularda zararlar ortaya çıkabilir. Sepsis olarak bilinen bu ciddi durum, tek veya birden fazla organ yetmezliğine kadar ilerleyebilir ve hatta ölüme neden olabilir. Mikroorganizmaların kana karışmasıyla başlayan bu durum aynı zamanda kan zehirlenmesi olarak da adlandırılır. [1]

Sepsis teşhisi, genellikle klinik belirtiler ve laboratuvar testlerine göre konulur ve bu da doğru teşhisi zamanında koymayı zorlaştırabilir. Bu durum, hastalığın ciddiyetini ve tedaviye ne kadar hızlı başlanması gerektiğini göz ardı etmenin potansiyel risklerini beraberinde getirir.

Bu proje erken ve doğru teşhisi hızlandırmayı ve bu kritik durumu daha etkili bir şekilde yönetmeyi amaçlamaktadır. Sepsis tanısında mevcut yöntemlerin eksikliklerine odaklanılarak makine öğrenimi tekniklerinin kullanılmasıyla hastalığın erken teşhisi konusundaki sürecin iyileşmesi hedeflenmektedir

1.1. PROJE GENEL ŞEMA

Proje boyunca izlenecek olan yollar genel şema ile şöyle ifade edilebilir:



Şekil 1 Proje Genel Şema

1.2. LİTERATÜR ÖZETİ

Projemizde kullanılacak veri seti ve algoritmalara karar verilmesi amacıyla yaptığımız literatür çalışması sonucunda daha önce yapılmış benzer çalışmalar incelenmiştir. Bazı örnek çalışmalar şu şekildedir:

2023 yılında Ankara Üniversitesinde Gülperi Tunçyürek ve 3 ekip arkadaşı 3013 hastanın verileri kullanılarak bir proje geliştirmiştir. Bu projede bazı makine

öğrenmesi teknikleri ve yapay sinir ağları kullanılarak %83.7 başarı oranı elde edilmiştir. [2]

Pınar Kaya Aksoy 2022 yılında İzmir Bakırçay Üniversitesinde yapmış olduğu Lisansüstü Tez çalışmasında 977 hastanın verileri ile 32 farklı makine öğrenmesi modeli kullanmıştır. Bazı modeller %100 doğruluk oranı verse de k fold cross validation yöntemi ile yapılan doğrulamalar sonucunda en yüksek sınıflandırma başarısı %92.7'de kalmıştır.[3]

Literatürde bulunan bu çalışmalar ve diğer bazı benzer çalışmaların en büyük kısıtlılığı kullanılan veri setlerinin yetersizliğidir.

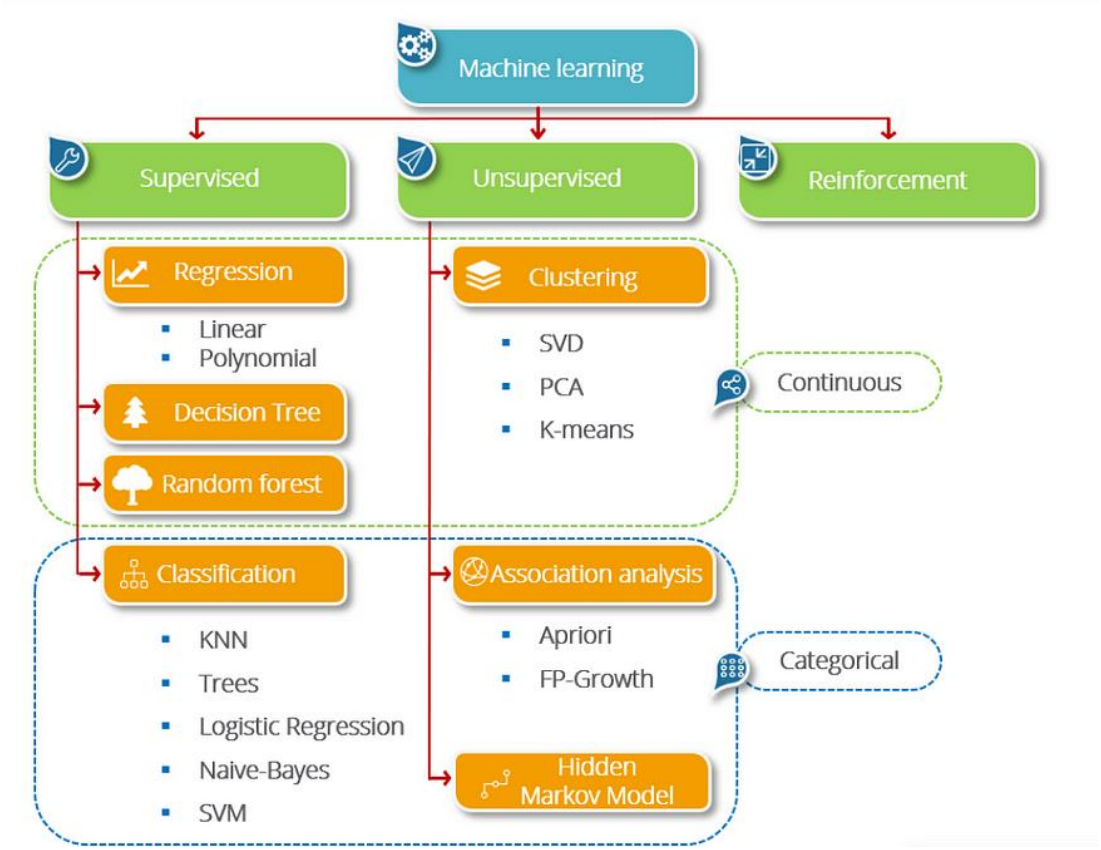
1.3. PROJENİN AMACI

Bu çalışmada makine öğrenmesi yöntemleri ile sepsis hastalığının hızlı ve güvenilir bir şekilde tespit edilmesi hedeflenmektedir. Bu projede, büyük bir veri seti üzerinde sepsis riskini belirlemek, klinik verileri analiz etmek, hasta durumunu izlemek ve doğru teşhis için öngörü modelleri oluşturmak yer almaktadır. Böylece, mevcut sağlık verileri kullanılarak sepsis teşhisinin doğruluğu artacak ve tedavi süreçleri hızlanacaktır. Ayrıca, elde edilen sonuç verileri klinik uygulamalara yol gösterecek ve sağlık sektöründe daha etkili sepsis teşhisini sağlayacaktır.

BÖLÜM 2

MAKİNE ÖĞRENMESİ

Makine öğrenimi (ML), bir veri setindeki verilere göre öğrenen ya da performansı iyileştiren sistemler oluşturmaya odaklanan bir yapay zeka (AI) alt kümesidir. Yapay zeka, insan zekasını taklit eden sistemler veya makineler anlamına gelen kapsamlı bir terimdir. Makine öğrenimi ve yapay zeka genellikle bir arada değerlendirilir ancak aynı anlama gelmezler. Tüm makine öğrenimi çözümleri yapay zeka iken tüm yapay zeka çözümlerinin makine öğrenimi olmaması önemli bir ayrımdır.[4]



Şekil 2 Makine Öğrenmesi Türleri [5]

2.1. MAKİNE ÖĞRENMESİ SÜRECİ HANGİ AŞAMALARI İÇERİR?

1- **Problem belirleme:** Neyi öngörmemiz gerektiğine ve bu tahminleri yapmak için ne tür gözlem verilerine sahip olmamız gerektiğine göre makine öğrenimi problemi netleştirilir.

2- **Veri toplama:** Yapılandırılmış (veri tabanından alınmış tablolu net veri gibi) veya yapılandırılmamış (düz paragraf metni gibi) verileri toplayarak bir veri seti oluşturulur.

3- **Veri hazırlama:** Verilerin makine öğrenimi için uygun şekilde hazırlanma aşamasıdır.

4- **Model seçimi:** Tahmin etmek için model seçmek önemli bir adımdır. Problemi en iyi şekilde temsil edecek ve en iyi sonucu verecek model seçilmelidir.

5- **Eğitim-Doğrulama-Test verilerinin ayrılması:** Veriler, modelin çıktıyı öngörme yeteneğini kademeli olarak geliştirir. Veri setini eğitim, validasyon ve test verisi olarak ayrılır.

6- **Değerlendirme:** Değerlendirme, modelimizi eğitim için hiç kullanılmamış verilere karşı test etmemize olanak sağlar.

7- **Parametre ayarlama:** Elde edilen sonuçların değerlendirilmesinin ardından sonuçların daha da iyileştirilip iyileştirilemeyeceğine bakılması gerekmektedir. En iyi sonuçların elde edildiği uygun parametreleri ayarlanarak sonuçlar iyileştirilebilir.

8- **Tahmin:** Modelin görmediği veriler ile tahmin yapılır.[6]

2.2. GÖZETİMLİ ÖĞRENME

Bağımlı değişken ve bağımsız değişken, yani girdi ve çıktı bir aradaysa buna gözetimli öğrenme denir. Gözetimli öğrenmede, makineyi 'etiketlenmiş' verileri kullanarak eğitirsiniz. Bu, bazı verilerin zaten doğru yanıtla eşleştirildiği anlamına gelir.

Gözetimli öğrenme algoritması, etiketli eğitim verilerinden öğrenir, öngörülemeyen veriler için sonuçları tahmin etmenize yardımcı olur. Doğru öğrenme modelini başarıyla oluşturmak, son derece yetenekli veri bilimcilerinden oluşan bir ekip, zaman ve teknik uzmanlık gerektirir. Dahası, veri bilimci verilen girdiler değiştiğinde de sonucun doğru kalmasını sağlamak için modelleri yeniden oluşturmalıdır.

En sık kullanılan gözetimli öğrenme algoritmaları Karar Ağaçları, Lineer Regresyon, Destek Vektör Makineleri ve Lojistik Regresyondur.[7]

2.3. GÖZETİMSİZ ÖĞRENME

Gözetimsiz öğrenmede etiketli veriler bulunmaz. Gözlemlenen birimler benzer özelliklerine göre bir araya getirilir. Gözetimsiz öğrenme, etiketlenmemiş veri kümelerini analiz etmek ve kümelemek için yapay öğrenme algoritmalarını kullanır. Bu algoritmalar, insan müdahalesine ihtiyaç duymadan verileri sınıflandırırlar.

Gözetimsiz öğrenme modelleri üç ana görev için kullanılır: Kümeleme, İlişkilendirme ve Boyutsallık Azaltma.[8]

2.4 PROJEDE KULLANILACAK ALGORİTMALAR

Projede kullanılan algoritmalar şunlardır ; KNN, Logistic Regresyon, SVM, Random Forest, XGBoost, LightGBM

2.4.1 KNN (K En Yakın Komşu Algoritması)

En temel ve anlaşılması basit makine öğrenmesi algoritmalarından biridir. KNN algoritması hem regresyon hem de sınıflandırma için kullanılabilir.

K en yakın komşu (KNN) algoritması, adından da anlaşılacağı üzere, komşularına bakarak tahminlemede bulunan bir algoritmadır. KNN algoritmasında, benzer olan şeyler birbirine yakındır varsayımı geçerlidir. Bu varsayıma dayanarak tahminleme yapılacak örneğin K sayıdaki en yakın komşusuna bakılır ve buna göre sınıflandırma yapılır. Örneğin K değeri 5 ise ve en yakın 5 komşudan 3 tanesi A sınıfına, 2 tanesi B sınıfına aitse bu örnek için algoritmanın tahmin ettiği sınıf A sınıfı olacaktır.

KNN algoritmasında K değerinin tek sayıda seçilmesi önemlidir. Aksi durumda en yakın komşuların ait olduğu sınıflar eşit sayıda olabilir. (Örneğin; K 4 ise 2 komşu A sınıfına ait, diğer 2 komşu B sınıfına ait)

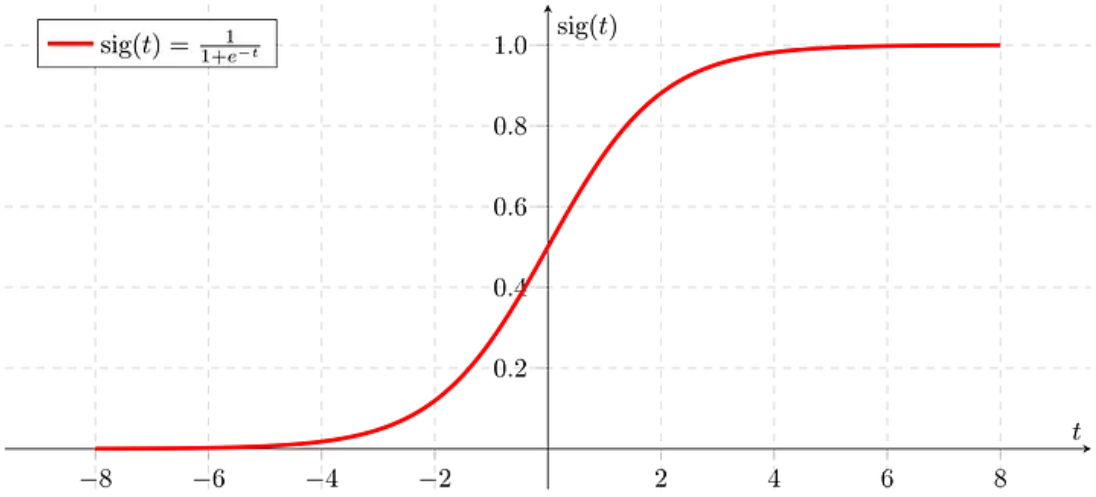
KNN algoritması için en yakın komşuları belirlerken farklı mesafe ölçme yöntemleri kullanılabilir. Bunlardan bazıları; Öklid, Manhattan, Minkowski.[9]

2.4.2 Logistic Regresyon

Lojistik regresyon, kategorik bağımlı değişkenlerin olasılıklarını tahmin etmek için kullanılan denetimli bir makine öğrenimi sınıflandırma algoritmasıdır. Lojistik regresyon, sürekli ve ayrık değişkenler ve doğrusal olmayan özellikler dahil olmak üzere çok sayıda özellikten yararlanabilir.[10]

Logistic Regresyon, isminde her ne kadar "regresyon" kelimesi geçse de aslında bir sınıflandırma algoritmasıdır.

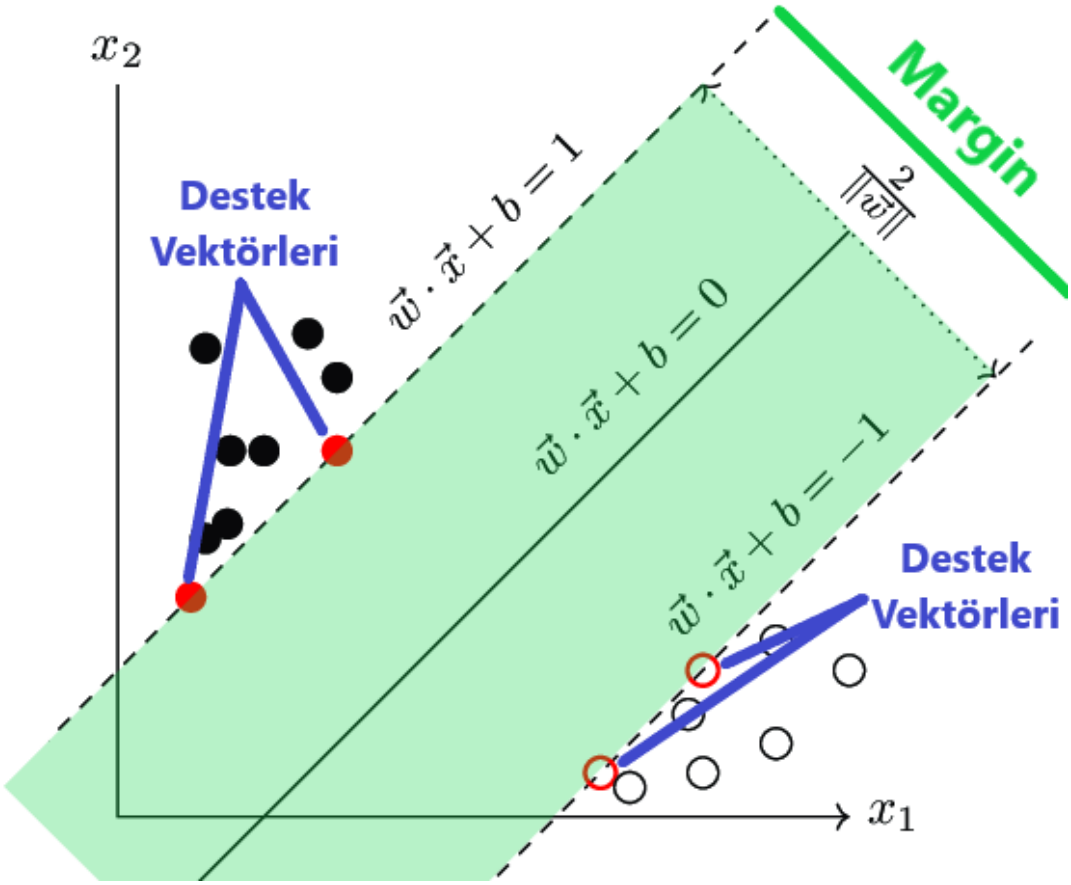
Lojistik regresyon, sınıflandırma yapmak için ***Sigmoid*** (Lojistik) Fonksiyonu kullanır. Sigmoid fonksiyonu "**S**" şeklinde bir eğridir. Sigmoid fonksiyonu derin öğrenmede aktivasyon fonksiyonu olarak da sıklıkla kullanılır.[11]



Şekil 3 Logistic Regression [12]

2.4.3 SVM (Support Vector Machine)

Destek Vektör Makineleri (Support Vector Machine), sınıflandırma problemlerinde yaygın olarak kullanılan bir gözetimli öğrenme yöntemidir. Bu yöntem, iki sınıfa ait noktaları ayırmak için bir düzlemde bir doğru çizer ve bu doğrunun, her iki sınıftaki noktalara da en uzak mesafede olmasını sağlamaya çalışır. [13]

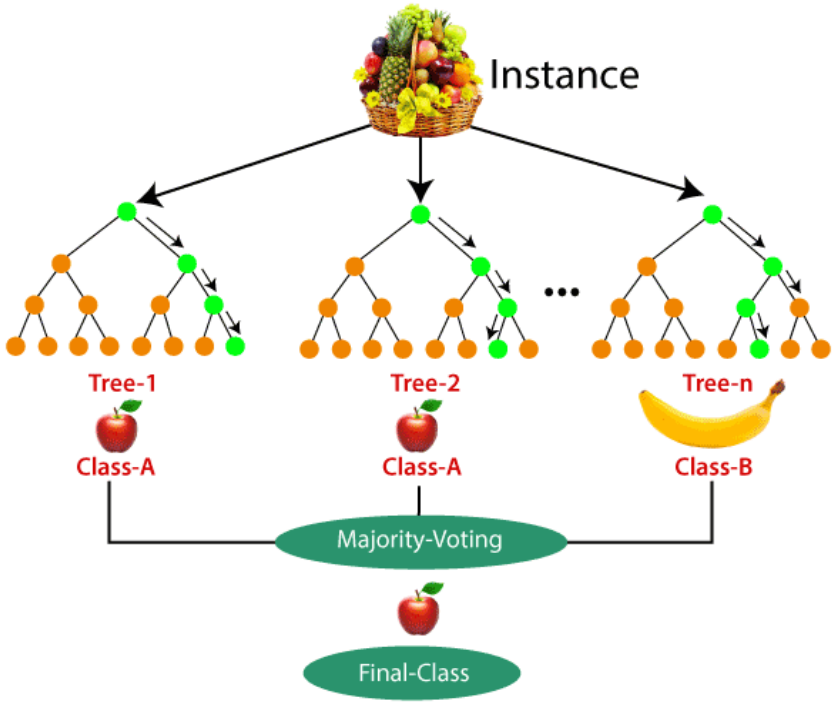


Şekil 4 SVM [13]

2.4.4 Random Forest

Random Forest algoritması, denetimli öğrenme teknikleri arasında yer alan güçlü bir algoritmadır. Hem sınıflandırma hem de regresyon problemlerinde kullanılarak, yüksek doğruluk oranlarına ulaşılmasını sağlar. Algoritma, birbirinden bağımsız birden fazla karar ağacı oluşturur ve bunları bir araya getirerek, en yüksek puan alan değerin seçilmesi ile sınıflandırma değerini yükseltmeyi hedefler. Bu sayede Random Forest, verilerin karmaşıklığını azaltarak daha iyi sonuçlar elde edilmesini sağlar. [14]

Random Forest, iki aşamada işlev görür: İlk olarak N sayıda bağımsız karar ağacı oluşturulur ve bunlar rastgele bir orman oluşturulmak üzere birleştirilir. Daha sonra her bir ağaç için ayrı ayrı tahminler yapılır ve en yüksek puan alan değer seçilir. [15]



Şekil 5 Random Forest

2.4.5 XGBoost

XGBoost, makine öğrenimi modellerinin eğitimi için optimize edilmiş bir dağıtık gradient boosting kütüphanesidir. Zayıf modellerin bir araya getirilmesi ile güçlü bir tahmin oluşturan bir topluluk öğrenme yöntemidir. Büyük veri kümelerinde üstün performans göstermesi ve birçok makine öğrenimi görevinde lider konumda olması nedeniyle XGBoost, en popüler makine öğrenimi algoritmalarından biri haline gelmiştir. [16]

XGBoost, gerçek verilerde sıklıkla karşılaştığımız eksik değerlerle başa çıkabilecek önemli bir özelliğe sahiptir. Bu özellik, önemli bir ön işleme gerektirmeden eksik değerlere sahip verileri kullanmamızı sağlar. [16]

XGBoost'un diğer algoritmalardan daha iyi performans göstermesinin sebebi, gradyan iniş mimarilerini kullanarak zayıf öğrenenlere artırma ilkesi uygulanmasıdır.[17]

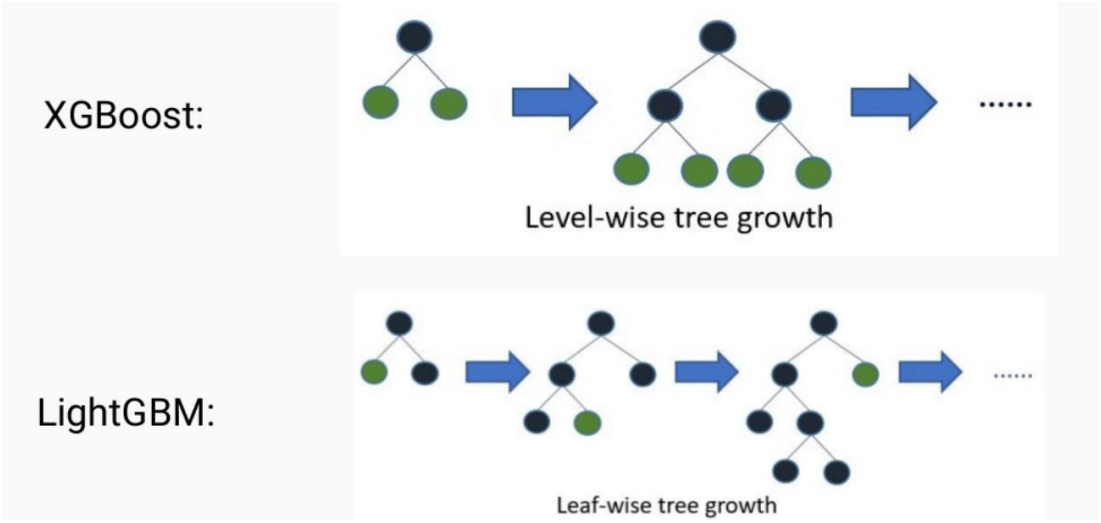
2.4.6 LightGBM

LightGBM, Microsoft DMTK (Distributed Machine Learning Toolkit) projesi kapsamında 2017 yılında geliştirilmiş bir boosting algoritmasıdır. Diğer boosting algoritmalarına kıyasla, bu algoritma yüksek işlem hızı, büyük veri kümelerini işleme kabiliyeti, az kaynak (RAM) kullanımı, yüksek tahmin oranı, paralel öğrenme ve GPU öğrenimine destek gibi bir dizi avantaj sunar. Yapılan bazı çalışmalarda LightGBM'in diğer modellere göre 20 kat daha hızlı olduğu bildiriliyor. [18]

LightGBM, histogram tabanlı çalışan bir algoritmadır.[18]

Algoritma, hedef değişkeni girdi özelliklerine göre tahmin eden tek bir karar ağacı oluşturarak başlar. Daha sonra modele daha fazla karar ağacı ekleyerek, her bir ağaç önceki ağacın hatalarını düzeltmeye çalışır. [19]

LightGBM'yi özellikle etkili kılan birkaç önemli özellik vardır: karar ağacı öğrenimi kullanımı, histogram tabanlı yaklaşım, yaprak bazlı ağaç büyümesi ve düzenleme (regularization).[19]



Şekil 6 LightGBM [20]

2.5. VERİ SETİ

Veri seti, belirli bir konunun sayılar veya değerler koleksiyonu olarak toplanıp saklanmasıyla oluşturulan dosyalara denir. Örneğin, belirli bir sınıftaki her öğrencinin test puanları; bir yoldan geçen araçların plakaları, renkleri, hızları gibi özelliklerin belli bir tablolama çerçevesinde toplanıp saklanmasıyla oluşan dosyalar birer veri setidir. Veri seti, tek bir veri tabanı tablosunun veya tek bir istatistiksel veri matrisinin içeriğine karşılık gelir; burada, tablonun her sütunu belirli bir değişkeni temsil eder ve her satır, söz konusu veri kümesinin belirli bir üyesine karşılık gelir.[21]

2.5.1. Seçilen Veri Seti

Seçtiğimiz veri seti, Kaggle platformu üzerinde bulunan Prediction of Sepsis (PhysioNet Computing in Cardiology Challenge 2019) veri setidir. Bu veri setinin daha önce yapılmış benzer çalışmalarda kullanılan veri setlerine göre en büyük avantajı çok büyük ve kapsamlı bir veri seti olmasıdır.

Seçilen veri setinin özellikleri şöyledir:

Prediction of Sepsis Data Set [22].

- Attribute Information:

  Vital signs (columns 1-8)

  HR: Heart rate (beats per minute)

  O2Sat: Pulse oximetry (%)

  Temp: Temperature (Deg C)

  SBP: Systolic BP (mm Hg)

  MAP: Mean arterial pressure (mm Hg)

DBP: Diastolic BP (mm Hg)

Resp: Respiration rate (breaths per minute)

EtCO2: End tidal carbon dioxide (mm Hg)

Laboratory values (columns 9-34)

BaseExcess: Measure of excess bicarbonate (mmol/L)

HCO3: Bicarbonate (mmol/L)

FiO2: Fraction of inspired oxygen (%)

pH: N/A

PaCO2: Partial pressure of carbon dioxide from arterial blood (mm Hg)

SaO2: Oxygen saturation from arterial blood (%)

AST: Aspartate transaminase (IU/L)

BUN: Blood urea nitrogen (mg/dL)

Alkalinephos: Alkaline phosphatase (IU/L)

Calcium: (mg/dL)

Chloride: (mmol/L)

Creatinine: (mg/dL)

Bilirubin_direct: Bilirubin direct (mg/dL)

Glucose: Serum glucose (mg/dL)

Lactate: Lactic acid (mg/dL)

Magnesium: (mmol/dL)

Phosphate: (mg/dL)

Potassium: (mmol/L)

Bilirubin_total: Total bilirubin (mg/dL)

TroponinI: Troponin I (ng/mL)

Hct: Hematocrit (%)

Hgb: Hemoglobin (g/dL)

PTT: partial thromboplastin time (seconds)

WBC: Leukocyte count (count*10^3/μL)

Fibrinogen: (mg/dL)

Platelets: (count*10^3/μL)

Demographics (columns 35-40)

Age: Years (100 for patients 90 or above)

Gender: Female (0) or Male (1)

Unit1: Administrative identifier for ICU unit (MICU)

Unit2: Administrative identifier for ICU unit (SICU)

HospAdmTime: Hours between hospital admit and ICU admit

ICULOS: ICU length-of-stay (hours since ICU admit)

Outcome (column 41)

SepsisLabel: For sepsis patients, SepsisLabel is 1 if $t \geq t_{sepsis} - 6$ and 0 if $t < t_{sepsis} - 6$. For non-sepsis patients, SepsisLabel is 0.

BÖLÜM 3
PROJENİN UYGULANMASI

3.1 Gerekli Kütüphanelerin İmport Edilmesi ve Veri Setinin Yüklenmesi

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import RobustScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_score
from sklearn.metrics import f1_score, confusion_matrix, make_scorer, fbeta_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from imblearn.under_sampling import RandomUnderSampler, NearMiss, TomekLinks
```

```
df = pd.read_csv("C:/Users/Yusuf/datasets/sepsis/Dataset.csv")
```

3.2 Veri Analizi

3.2.1 Veri Seti Hakkında Temel Bilgiler

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1552210 entries, 0 to 1552209
Data columns (total 44 columns):
 #   Column            Non-Null Count    Dtype
---  ------            --------------    -----
 0   Unnamed: 0        1552210 non-null  int64
 1   Hour              1552210 non-null  int64
 2   HR                1398811 non-null  float64
 3   O2Sat             1349474 non-null  float64
 4   Temp              525226 non-null   float64
 5   SBP               1325945 non-null  float64
 6   MAP               1358940 non-null  float64
 7   DBP               1065656 non-null  float64
 8   Resp              1313875 non-null  float64
 9   EtCO2             57636 non-null    float64
 10  BaseExcess        84145 non-null    float64
 11  HCO3              65028 non-null    float64
 12  FiO2              129365 non-null   float64
 13  pH                107573 non-null   float64
 14  PaCO2             86301 non-null    float64
 15  SaO2              53561 non-null    float64
 16  AST               25183 non-null    float64
 17  BUN               106568 non-null   float64
 18  Alkalinephos      24941 non-null    float64
 19  Calcium           91331 non-null    float64
 20  Chloride          70466 non-null    float64
 21  Creatinine        94616 non-null    float64
 22  Bilirubin_direct  2990 non-null     float64
 23  Glucose           265516 non-null   float64
 24  Lactate           41446 non-null    float64
 25  Magnesium         97951 non-null    float64
 26  Phosphate         62301 non-null    float64
 27  Potassium         144525 non-null   float64
 28  Bilirubin_total   23141 non-null    float64
 29  TroponinI         14781 non-null    float64
 30  Hct               137433 non-null   float64
 31  Hgb               114591 non-null   float64
 32  PTT               45699 non-null    float64
 33  WBC               99447 non-null    float64
 34  Fibrinogen        10242 non-null    float64
 35  Platelets         92209 non-null    float64
 36  Age               1552210 non-null  float64
 37  Gender            1552210 non-null  int64
 38  Unit1             940250 non-null   float64
 39  Unit2             940250 non-null   float64
 40  HospAdmTime       1552202 non-null  float64
 41  ICULOS            1552210 non-null  int64
 42  SepsisLabel       1552210 non-null  int64
 43  Patient_ID        1552210 non-null  int64
```

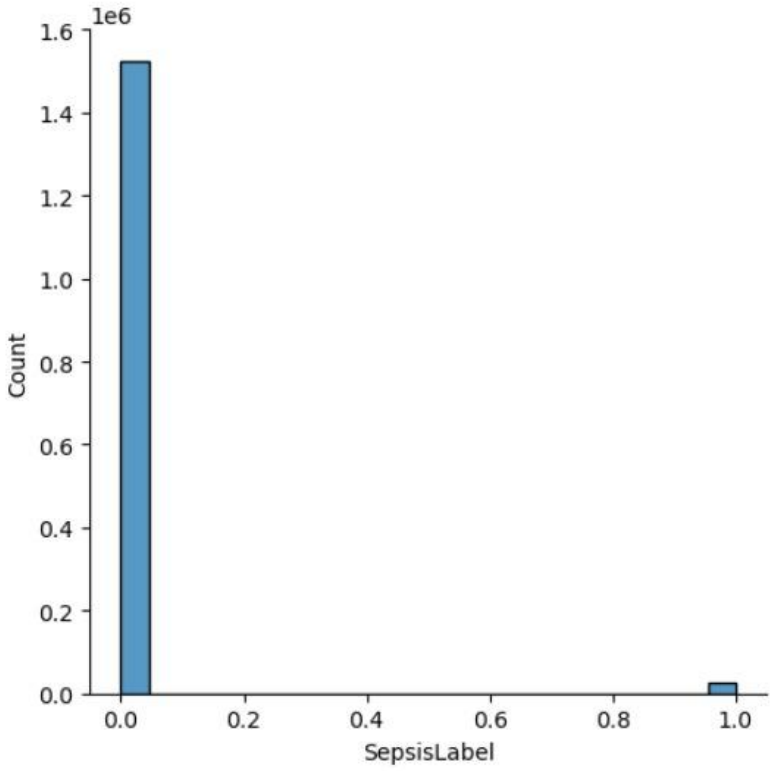
```
df['Patient_ID'].nunique()
```

```
40336
```

Veri seti 40336 hastanın farklı zaman aralıklarında alınmış tahlil verilerinden oluşuyor.

3.2.2 Sınıf Dağılımlarının Analizi

```
sns.displot(df["SepsisLabel"])
plt.show()
```




```
df["SepsisLabel"].value_counts()
```

```
0    1524294
1      27916
Name: SepsisLabel, dtype: int64
```

Görüldüğü gibi veri setinde dengesiz sınıf dağılımı problemi bulunuyor. Bu durum makine öğrenmesi modellerinin performansını olumsuz etkiler ve modelin azınlık sınıfını daha az öğrenmesine sebep olur. Bu durumu çözmek için veri ön işleme aşamasında farklı teknikler kullanılacaktır.

3.2.3 Eksik Veri Analizi

```
null_values = df.isnull().sum()
null_values = null_values / len(df)*100
null_values = null_values.sort_values(ascending=False)
null_values
```

Bilirubin_direct	99.807371	Hgb	92.617558
Fibrinogen	99.340167	FiO2	91.665754
TroponinI	99.047745	Hct	91.145979
Bilirubin_total	98.509158	Potassium	90.689082
Alkalinephos	98.393194	Glucose	82.894325
AST	98.377604	Temp	66.162697
Lactate	97.329872	Unit2	39.425078
PTT	97.055875	Unit1	39.425078
SaO2	96.549372	DBP	31.345887
EtCO2	96.286843	Resp	15.354559
Phosphate	95.986303	SBP	14.576958
HCO3	95.810618	O2Sat	13.061119
Chloride	95.460279	MAP	12.451279
BaseExcess	94.579020	HR	9.882619
PaCO2	94.440121	HospAdmTime	0.000515
Calcium	94.116067	SepsisLabel	0.000000
Platelets	94.059502	ICULOS	0.000000
Creatinine	93.904433	Unnamed: 0	0.000000
Magnesium	93.689578	Gender	0.000000
WBC	93.593199	Age	0.000000
BUN	93.134434	Hour	0.000000
pH	93.069688	Patient_ID	0.000000
		dtype: float64	

Veri setindeki özniteliklerin eksik veri oranları yüzdelik olarak yukarıda gösterilmiştir. Eksik veri oranlarının yüksekliği istenmeyen bir durumdur. Bu durum veri ön işleme adımında çözülecektir.

3.3 Veri Ön İşleme

3.3.1 Eksik Verilerin Doldurulması

Veri seti zaman serisinden oluştuğu için eksik verileri ortalama değerler ile doldurmak doğru bir yaklaşım olmayacaktır. Hasta verilerinin birbirlerine karışmaması için

“Patient_ID” sütunu ile hasta bazında imputasyon yapılmalı ve veri seti “bfill (backward fill)”, “ffill (forward fill)” metotları ile doldurulmalıdır.

bfill, geriye doğru doldurma yöntemidir. Eksik veriler, kendisinden sonra gelen geçerli bir veri ile doldurulur.

ffill, ileriye doğru doldurma yöntemidir. Eksik veriler, kendisinden önceki geçerli bir veri ile doldurulur.

a	b	c	d	e
1	8	abc	14	4
2	null	def	null	null
null	null	null	null	9
null	9	null	null	9
7	9	null	18	null
8	9	xyz	18	null

Şekil 7 Orijinal Veri

a	b	c	d	e
1	8	abc	14	4
2	9	def	18	9
7	9	xyz	18	9
7	9	xyz	18	9
7	9	xyz	18	null
8	9	xyz	18	null

Şekil 8 Bfill

a	b	c	d	e
1	8	abc	14	4
2	8	def	14	4
2	8	def	14	9
2	9	def	14	9
7	9	def	18	9
8	9	xyz	18	9

Şekil 9 Ffill

```
df[df.columns] = df.groupby('Patient_ID')[df.columns].transform(lambda x: x.fillna(method='bfill').fillna(method='ffill'))
```

3.3.2 Eksik Veri Oranı Yüksek Özniteliklerin Kaldırılması

```
null_values = df.isnull().sum()
null_values = null_values / len(df)*100
null_values = null_values.sort_values(ascending=False)
null_values
```

```
Bilirubin_direct    93.266633
EtCO2               89.938861
Fibrinogen          86.219455
TroponinI           81.905412
BaseExcess          63.709743
Lactate             63.533607
SaO2                62.980589
Alkalinephos        60.012756
Bilirubin_total     59.799447
AST                 59.567520
FiO2                50.224390
PaCO2               48.813691
HCO3                48.313952
pH                  47.427217
PTT                 45.201745
Chloride            44.905329
Unit2               39.425078
Unit1               39.425078
Phosphate           24.518847
DBP                 16.701993
Calcium              9.776705
Magnesium            8.787277
WBC                  4.227456
Platelets            4.150598
Hgb                  3.963317
Hct                  3.786472
Creatinine           3.275459
BUN                  3.233583
Potassium            3.072845
Glucose              2.708783
SBP                  0.707121
Temp                 0.478672
MAP                  0.173817
Resp                 0.128784
O2Sat                0.024095
HR                   0.009148
HospAdmTime          0.000515
SepsisLabel          0.000000
ICULOS               0.000000
Unnamed: 0           0.000000
Gender               0.000000
Age                  0.000000
Hour                 0.000000
Patient_ID           0.000000
dtype: float64
```

```
df = df.drop(['Bilirubin_direct', 'EtCO2', 'Fibrinogen', 'TroponinI',
'BaseExcess', 'Lactate', 'SaO2', 'Alkalinephos','Bilirubin_total',
'AST', 'FiO2', 'PaCO2', 'HCO3', 'pH', 'PTT', 'Chloride', 'Unit2',
'Unit1', 'Unnamed: 0', 'Patient_ID'], axis=1)

df.columns
```

```
Index(['Hour', 'HR', 'O2Sat', 'Temp', 'SBP', 'MAP', 'DBP', 'Resp', 'BUN',
       'Calcium', 'Creatinine', 'Glucose', 'Magnesium', 'Phosphate',
       'Potassium', 'Hct', 'Hgb', 'WBC', 'Platelets', 'Age', 'Gender',
       'HospAdmTime', 'ICULOS', 'SepsisLabel'],
      dtype='object')
```

%30'dan fazla eksik değer bulunan sütunlar ile Patient_ID ve Unnamed: 0 sütunları kaldırıldı.

3.3.3 Eksik Veri Bulunan Satırların Silinmesi

```
df.shape
```

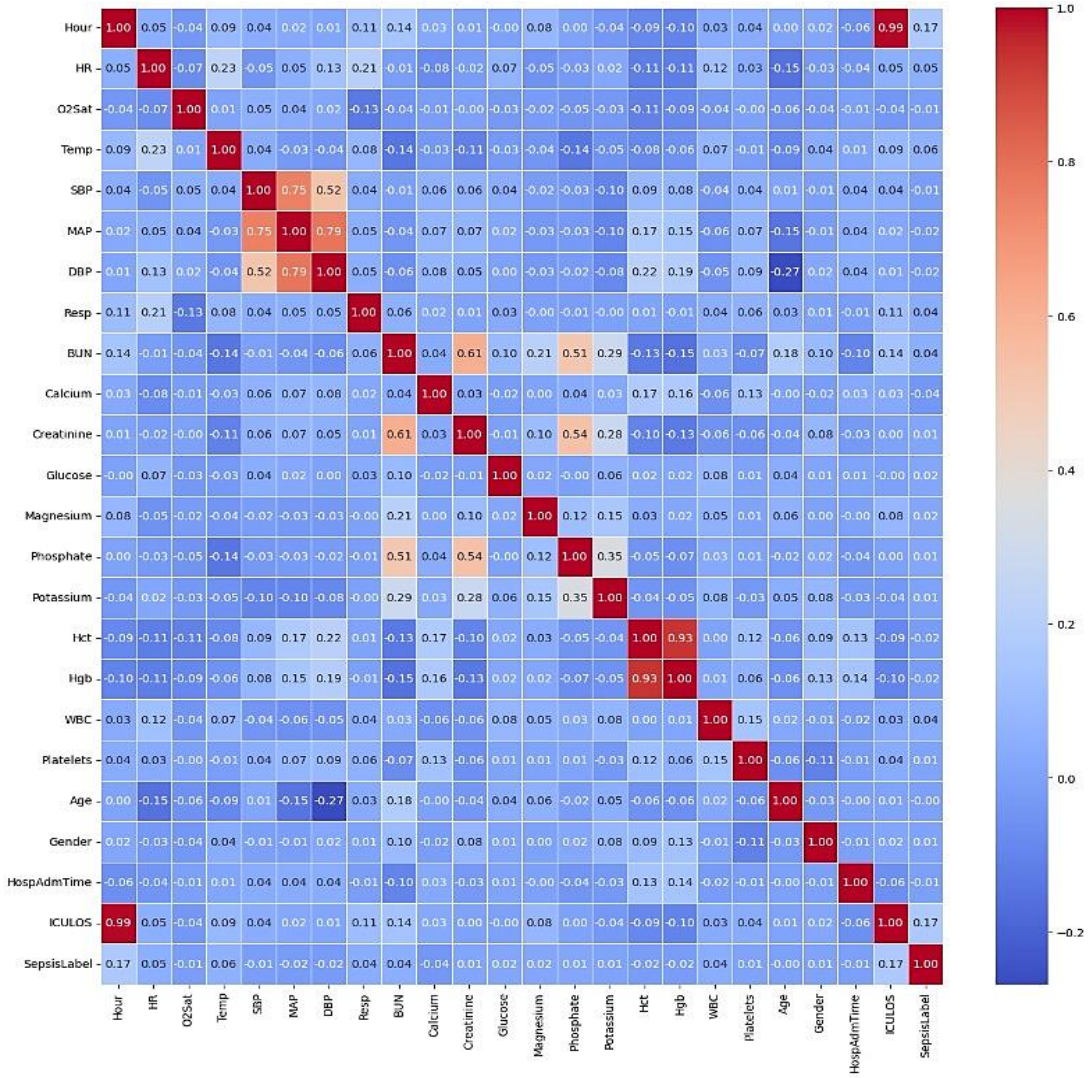
```
(1552210, 24)
```

```
df = df.dropna()
df.shape
```

```
(902045, 24)
```

3.3.4 Öznitelik Seçimi (Feature Selection)

```
plt.figure(figsize=(16, 16))
sns.heatmap(df.corr(), annot = True, fmt = ".2f", linewidths = .5, cmap='coolwarm')
plt.show()
```



Korelasyon matrisi için Seaborn kütüphanesinden faydalanarak ısı haritası(heatmap) oluşturulmuştur.

```
# Map ya da SBP ve DBP, Hour ya da ICULOS, Hgb ya da Hct
df = df.drop(['MAP', 'Hour', 'Hct'], axis=1)
df.columns
```

```
Index(['HR', 'O2Sat', 'Temp', 'SBP', 'DBP', 'Resp', 'BUN', 'Calcium',
       'Creatinine', 'Glucose', 'Magnesium', 'Phosphate', 'Potassium', 'Hgb',
       'WBC', 'Platelets', 'Age', 'Gender', 'HospAdmTime', 'ICULOS',
       'SepsisLabel'],
      dtype='object')
```

Yüksek korelasyonlu öznitelik çiftleri arasında biri diğerini açıklayabileceği için bazı öznitelikler kaldırılmıştır. İki öznitelik arasında %75'ten yüksek korelasyon varsa bu özniteliklerin her ikisini birden kullanmak mantıklı değildir.

3.3.5 StandardScaler ile Öznitelik Ölçeklendirme

StandardScaler, verileri standart normal dağılıma dönüştürmek için kullanılır. Bu yöntemde, her öznitelik için ortalama değer çıkarılır ve standart sapmaya bölünür. Sonuç olarak, ortalama değeri 0 ve standart sapması 1 olan bir normal dağılım elde edilir. Formülü ise şu şekildedir: [24]

$$\frac{X - \text{mean}(X)}{\text{std}(X)}$$

Şekil 10 StandardScaler [24]

```
scaler = StandardScaler()
scaling_columns = ['ICULOS', 'HR', 'O2Sat', 'Temp', 'SBP', 'DBP', 'Resp', 'BUN', 'Calcium',
                   'Creatinine', 'Glucose', 'Magnesium', 'Phosphate', 'Potassium', 'Hgb',
                   'WBC', 'Platelets', 'Age', 'HospAdmTime']
df[scaling_columns] = scaler.fit_transform(df[scaling_columns])
```

```
df.head()
```

	HR	O2Sat	Temp	SBP	DBP	Resp	BUN	Calcium	Creatinine	Glucose	...	Phosphate	Potassium	Hgb	WBC	Platele
0	-1.137495	0.881709	-1.499975	0.154014	0.328838	-0.392351	-0.00631	0.932934	-0.378237	0.618819	...	-1.04953	-1.460567	-0.45815	-0.01321	1.2597
1	-1.137495	0.881709	-1.499975	0.154014	0.328838	-0.392351	-0.00631	0.932934	-0.378237	0.618819	...	-1.04953	-1.460567	-0.45815	-0.01321	1.2597
2	-0.399727	0.881709	-1.499975	0.154014	0.328838	-0.297394	-0.00631	0.932934	-0.378237	0.618819	...	-1.04953	-1.460567	-0.45815	-0.01321	1.2597
3	-0.683484	0.881709	-1.499975	0.154014	0.328838	-0.297394	-0.00631	0.932934	-0.378237	0.618819	...	-1.04953	-1.460567	-0.45815	-0.01321	1.2597
4	-0.853738	0.881709	-1.499975	0.154014	0.328838	-0.867134	-0.00631	0.932934	-0.378237	0.618819	...	-1.04953	-1.460567	-0.45815	-0.01321	1.2597

5 rows × 21 columns

3.3.6 Veri Setinin Eğitim ve Test Seti Olarak Ayrılması

```
X = df.drop(['SepsisLabel'], axis=1)
y = df.SepsisLabel
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size = 0.25, random_state=42)
```

test_size, 0.25 olarak seçilmiştir. Bu durumda veri setinin %75'i eğitim için %25'i ise test için kullanılacaktır.

stratify=y parametresi, y'nin dağılımını eğitim ve test setlerinde koruyarak orantılı bir şekilde bölünmesini sağlar. Yani, sınıfların oranı eğitim ve test setlerinde aynı kalır. Dengesiz veri setleri için önemli bir parametredir.

random_state=42 parametresi, veri setinin her seferinde aynı rastgelelikte bölünmesini sağlar.

3.3.7 Dengesiz Veri Seti Probleminin Çözümü

Bölüm 3.2.2'de açıklandığı gibi, veri seti dengesiz sınıf dağılımına sahiptir. Bu durumun üstesinden gelmek için iki farklı strateji denenmiştir.

Birincisi, çoğunluk sınıfını 'undersampling' yöntemleri ile yeniden örnekleyerek sınıf dağılımlarını eşitlemektir. Burada dikkat edilmesi gereken husus, yeniden örnekleme işleminin tüm veri setine değil, sadece train setine uygulanması gerektiğidir. Aksi durumda, test setinin gerçek dünya verileri ile uyumu azalacak ve ortaya güvenilir olmayan sonuçlar çıkacaktır.

Undersampling için farklı metotlar denenmiş ve bu veri setinde en iyi sonucu veren, imbalanced learn kütüphanesindeki RandomUnderSampler yöntemi kullanılmıştır.

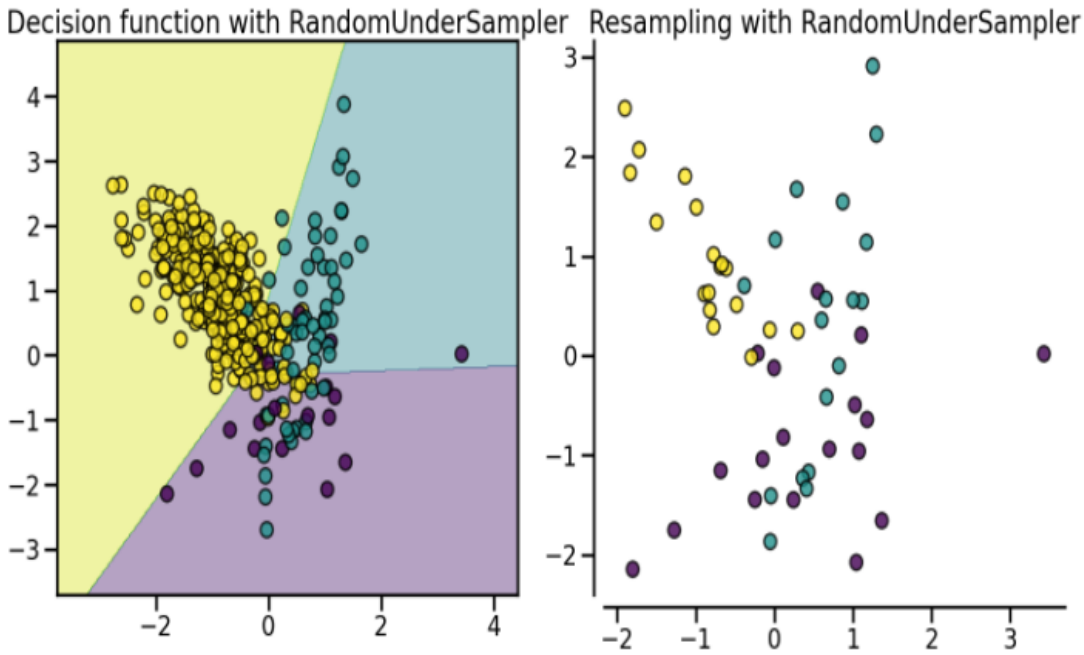
RandomUnderSampler, dengesiz veri setlerinde çoğunluk sınıfından rastgele örnekler çıkararak sınıf dengesi sağlamaya çalışan bir örnekleme tekniğidir. Bu yöntem, sınıflar arası dengesizliğin neden olabileceği öğrenme problemlerini azaltmak için kullanılır.

Uygulanan ikinci yöntem ise, veri seti üzerinde herhangi bir değişiklik yapmayıp makine öğrenmesi algoritmalarının bu konudaki özel parametrelerini kullanmaktır. Özellikle XGBoost algoritmasının 'scale_pos_weight' ve lightgbm algoritmasının 'is_unbalance' parametrelerinin çok iyi sonuçlar verdiği görülmüştür. Her iki yöntemin de karşılaştırılması bir sonraki bölümde yapılacaktır.

```
rus = RandomUnderSampler(random_state=42)
X_res, y_res = rus.fit_resample(X_train, y_train)
print(y_train.value_counts())
print(y_test.value_counts())
print(y_res.value_counts())
```

```
0    662829
1     13704
Name: SepsisLabel, dtype: int64
0    220944
1      4568
Name: SepsisLabel, dtype: int64
0    13704
1    13704
Name: SepsisLabel, dtype: int64
```

RandomUnderSampler yöntemi uygulandıktan sonra elde edilen yeniden örneklenmiş veri setinde sınıf dağılımları dengelenmiştir.



Şekil 11 Random UnderSampler [25]

3.4 MODELLEME

Model performansını ölçmek için kullanılan metrikler; Accuracy(Doğruluk), Precision(Kesinlik), Recall(Hassasiyet), F1 Score ve AUC-ROC metrikleridir.

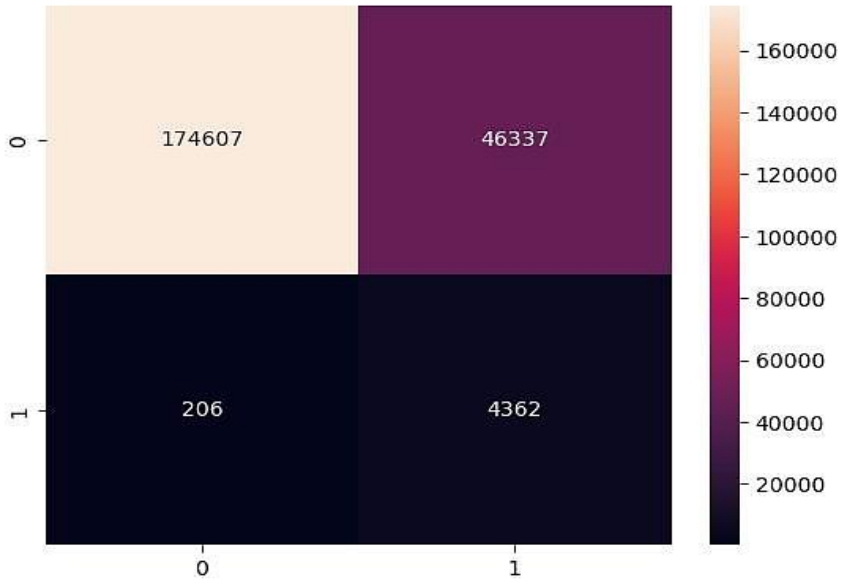
Accuracy: Modelin genel performansı
Precision: Doğru tahminlerin ne kadar kesin olduğu
Recall: Gerçek pozitif örneklerin oranı
F1 Score: Dengesiz sınıflar için yararlı hibrit metrik
AUC-ROC: Modelin iki sınıf arasında ayrım yapma yeteneği [26]

3.4.1 KNN (K – Nearest Neighbor)

```
knn_model = KNeighborsClassifier()

#knn_model.fit(X_train, y_train)
knn_model.fit(X_res, y_res)
knn_predictions = knn_model.predict(X_test)
evaluation(y_test,knn_predictions)
```

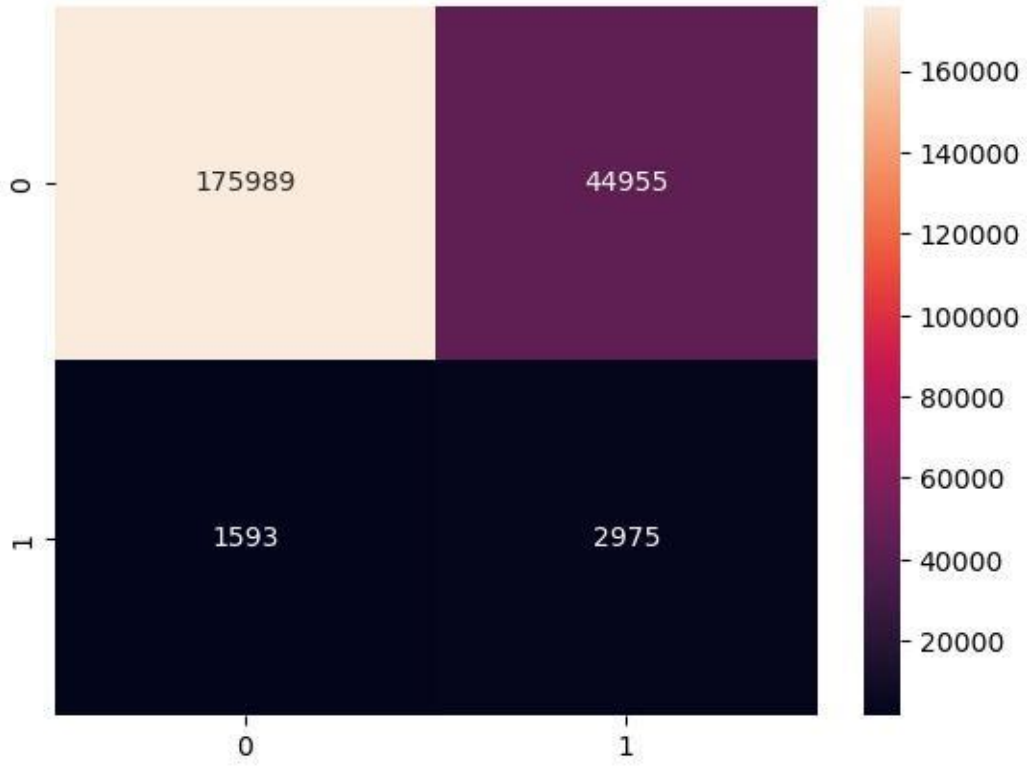
```
Accuracy:0.7936
Precision:0.0860
Recall:0.9549
F1 Score:0.1579
AUC-ROC:0.8726
```



3.4.2 Lojistik Regresyon

```
lgr_model = LogisticRegression()
#lgr_model.fit(X_train, y_train)
lgr_model.fit(X_res, y_res)
lgr_predictions = lgr_model.predict(X_test)
evaluation(y_test,lgr_predictions)
```

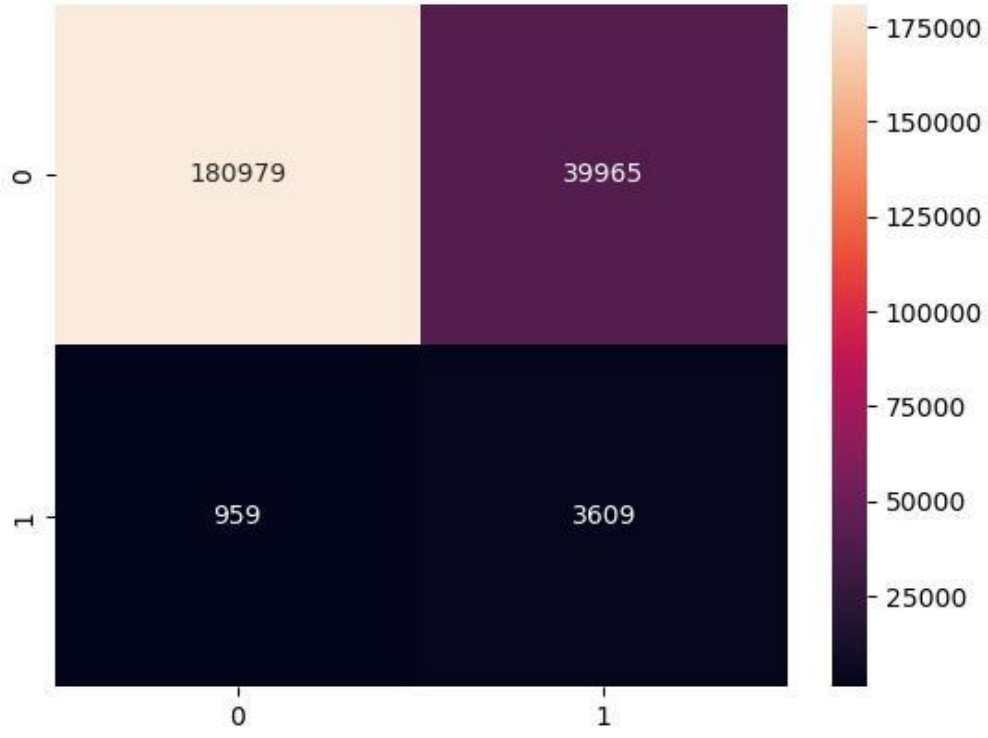
```
Accuracy:0.7936
Precision:0.0621
Recall:0.6513
F1 Score:0.1133
AUC-ROC:0.7239
```



3.4.3 Support Vector Machine

```
#svc_model = SVC()
#svc_model = SVC(kernel="linear")
#svc_model = SVC(kernel="poly")
#svc_model = SVC(kernel="sigmoid")
svc_model = SVC(kernel="rbf")
#svc_model.fit(X_train, y_train)
svc_model.fit(X_res, y_res)
svc_predictions = svc_model.predict(X_test)
evaluation(y_test,svc_predictions)
```

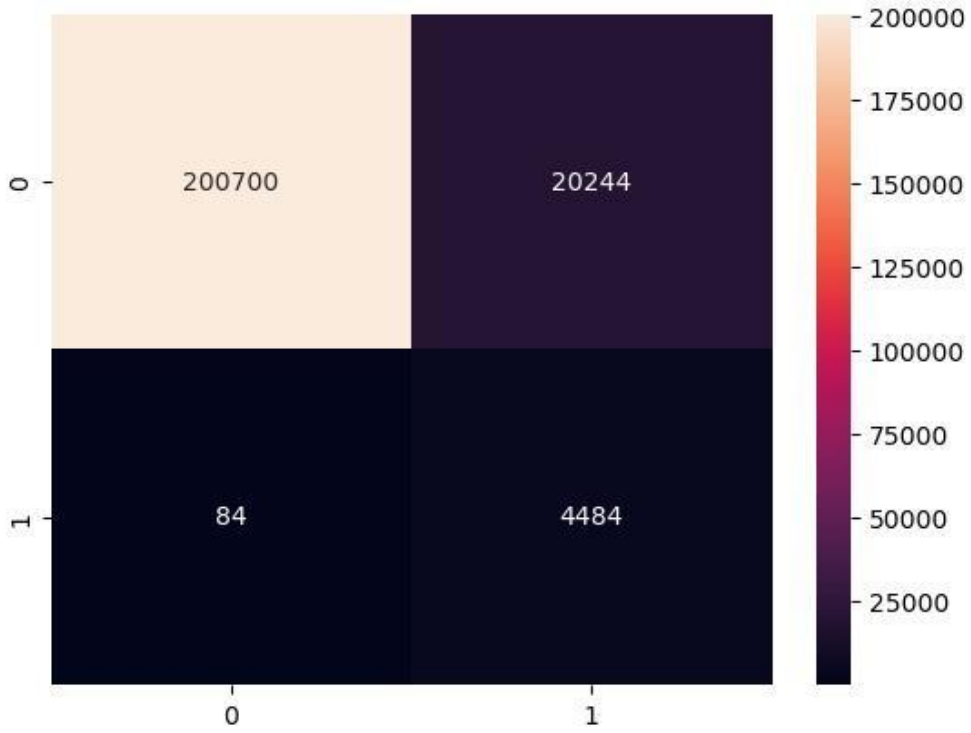
```
Accuracy:0.8185
Precision:0.0828
Recall:0.7901
F1 Score:0.1499
AUC-ROC:0.8046
```



3.4.4 Random Forest

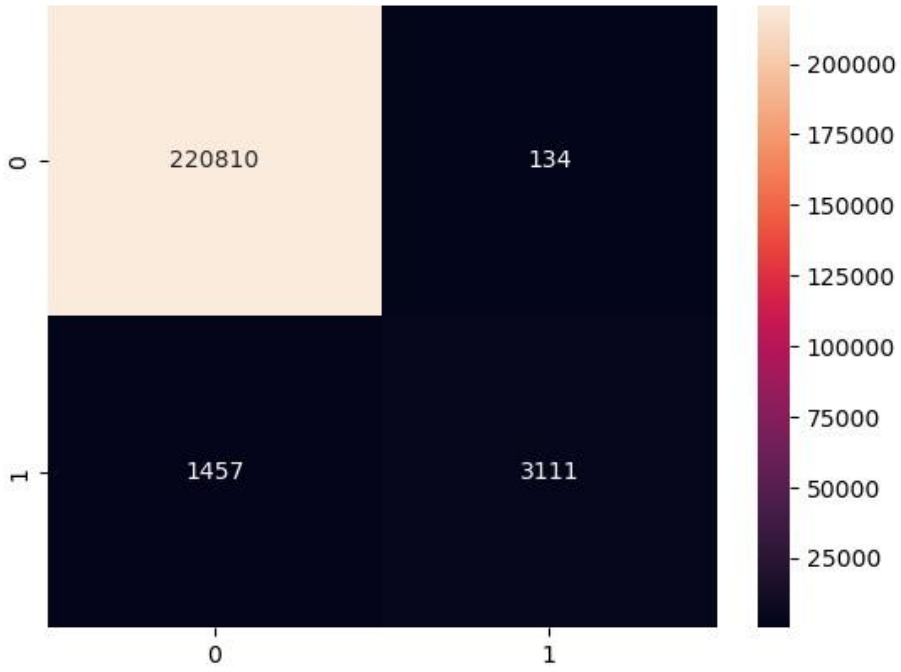
```
rfc_model = RandomForestClassifier(random_state=42)
#rfc_model.fit(X_train, y_train)
rfc_model.fit(X_res, y_res)
rfc_predictions = rfc_model.predict(X_test)
evaluation(y_test,rfc_predictions)
```

```
Accuracy:0.9099
Precision:0.1813
Recall:0.9816
F1 Score:0.3061
AUC-ROC:0.9450
```



```
rfc_model2 = RandomForestClassifier(random_state=42, class_weight='balanced')
rfc_model2.fit(X_train, y_train)
rfc2_predictions = rfc_model2.predict(X_test)
evaluation(y_test,rfc2_predictions)
```

```
Accuracy:0.9929
Precision:0.9587
Recall:0.6810
F1 Score:0.7964
AUC-ROC:0.8402
```

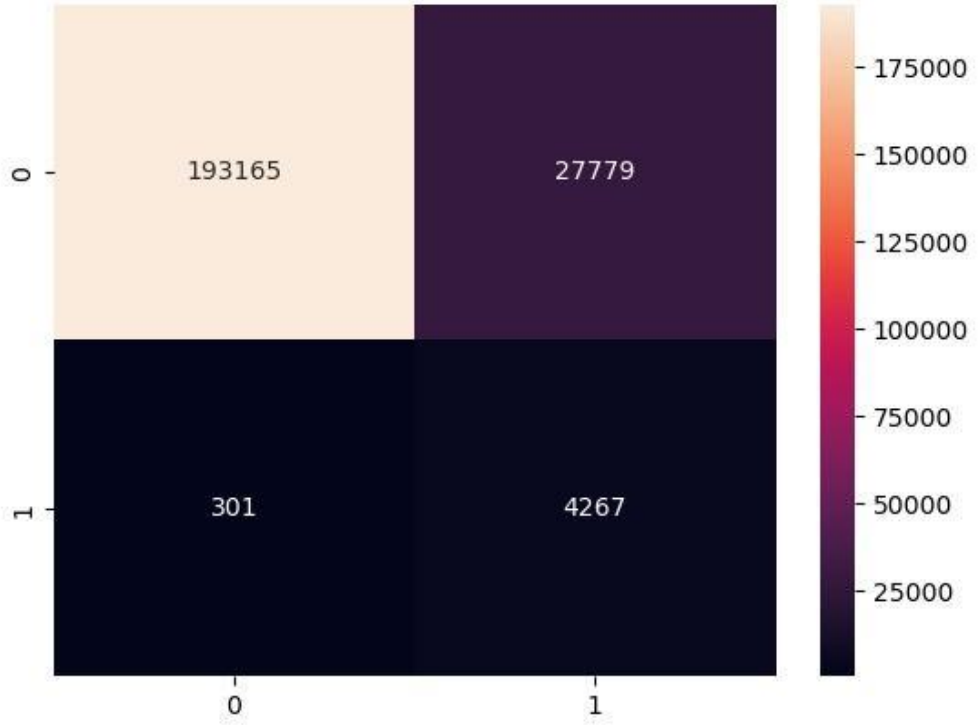


‘class_weght’ parametresi ile tüm veri seti üzerinde yapılan model, undersampling kullanılarak dengelenmiş veri seti ile yapılan ilk modele göre daha kötü performans göstermiş ve recall sonucu düşmüştür. Bu yüzden parametre optimizasyonu kısmında ilk model kullanılacaktır.

3.4.5 XGBoost Algoritması

```
xgb_model = XGBClassifier()
#xgb_model.fit(X_train, y_train)
xgb_model.fit(X_res, y_res)
xgb_predictions = xgb_model.predict(X_test)
evaluation(y_test,xgb_predictions)
```

```
Accuracy:0.8755
Precision:0.1332
Recall:0.9341
F1 Score:0.2331
AUC-ROC:0.9042
```



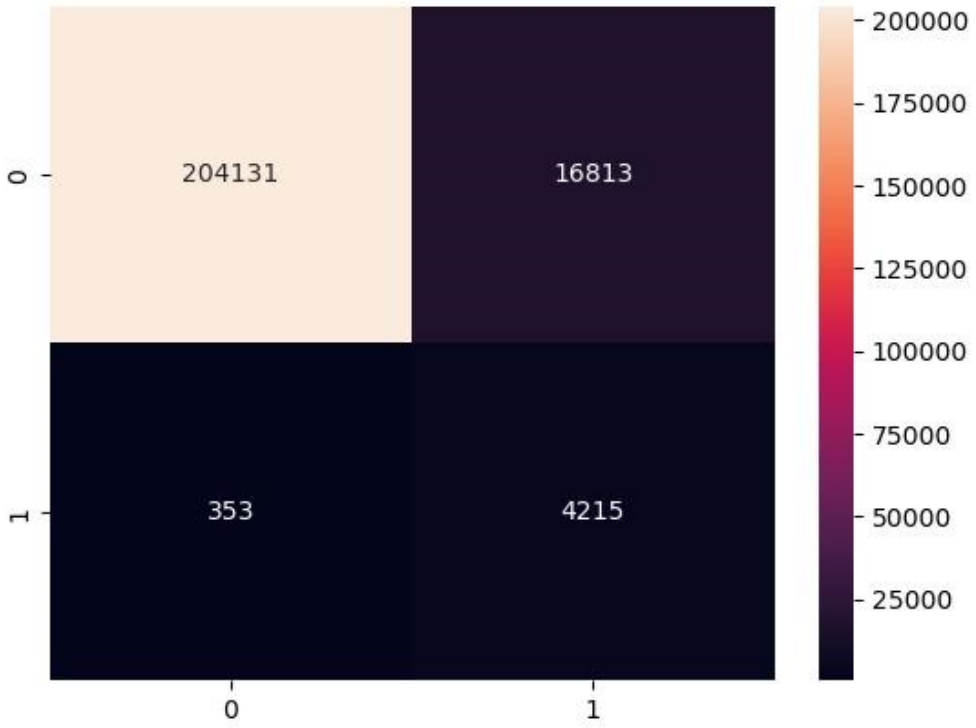

```
scale_pos_weight = sum(y_train == 0) / sum(y_train == 1)
scale_pos_weight
```

```
48.36755691768827
```

```
xgb_model2 = XGBClassifier(scale_pos_weight=scale_pos_weight)
xgb_model2.fit(X_train, y_train)
xgb2_predictions = xgb_model2.predict(X_test)
evaluation(y_test,xgb2_predictions)

# no scale_pos_weight
#Accuracy:0.9833
#Precision:0.8722
#Recall:0.2077
#F1 Score:0.3356
#AUC-ROC:0.6036
```

```
Accuracy:0.9239
Precision:0.2004
Recall:0.9227
F1 Score:0.3293
AUC-ROC:0.9233
```

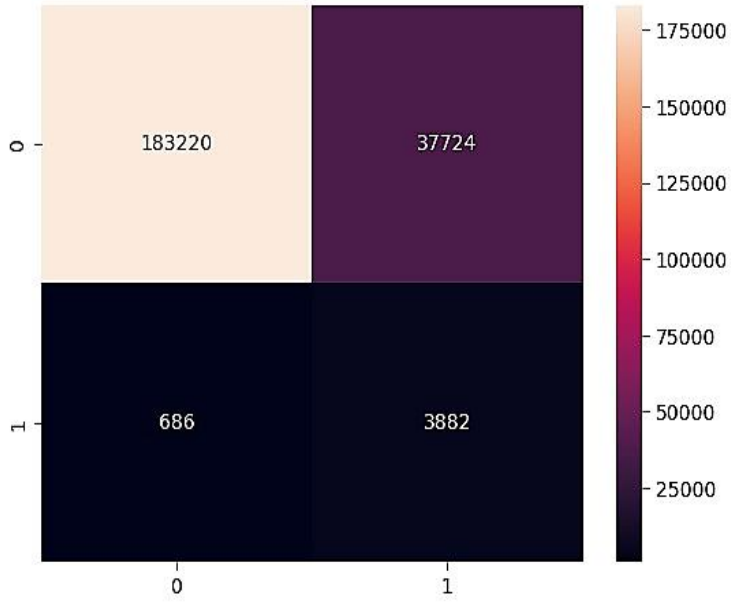


‘scale_pos_weight’ parametresi ile tüm veri seti kullanılarak oluşturulan model, undersamling ile dengelenmiş ilk modelden daha iyi bir performans göstermiştir. Bu yüzden parametre optimizasyonu kısmında ikinci model kullanılacaktır.

3.4.6 LightGBM

```
lgbm_model = LGBMClassifier()
#lgbm_model.fit(X_train, y_train)
lgbm_model.fit(X_res, y_res)
lgbm_predictions = lgbm_model.predict(X_test)
evaluation(y_test, lgbm_predictions)
```

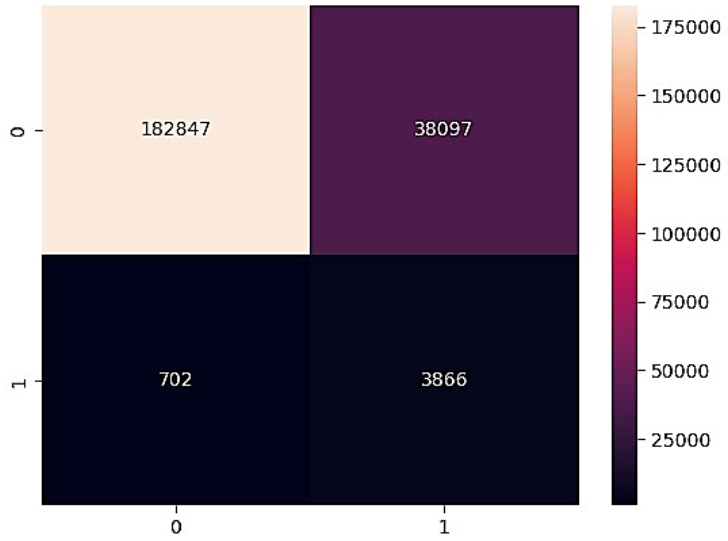
```
[LightGBM] [Info] Number of positive: 13704, number of negative: 13704
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001407 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 3499
[LightGBM] [Info] Number of data points in the train set: 27408, number of used features: 20
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000
Accuracy:0.8297
Precision:0.0933
Recall:0.8498
F1 Score:0.1681
AUC-ROC:0.8395
```



```
lgbm_model2 = LGBMClassifier(is_unbalance=True)
lgbm_model2.fit(X_train, y_train)
lgbm2_predictions = lgbm_model2.predict(X_test)
evaluation(y_test, lgbm2_predictions)

# no is_unbalance
#Accuracy:0.9806
#Precision:0.6279
#Recall:0.1075
#F1 Score:0.1836
#AUC-ROC:0.5531
```

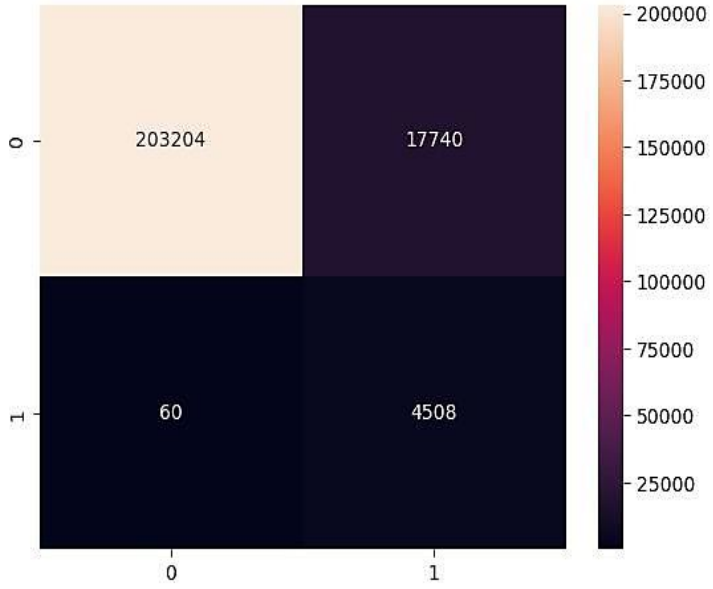
```
[LightGBM] [Info] Number of positive: 13704, number of negative: 662829
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.025413 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 4049
[LightGBM] [Info] Number of data points in the train set: 676533, number of used features: 20
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.020256 -> initscore=-3.878829
[LightGBM] [Info] Start training from score -3.878829
Accuracy:0.8280
Precision:0.0921
Recall:0.8463
F1 Score:0.1662
AUC-ROC:0.8369
```



Random Forest ve XGBoost algoritmalarında olduğu gibi dengesiz veri seti problemi sebebiyle iki farklı strateji ile modelleme yapılmış ve default parameteler ile her iki modelin de sonuçları birbirine yakın çıkmıştır.

```
lgbm_model = LGBMClassifier(n_estimators=300, num_leaves=100)
#lgbm_model.fit(X_train, y_train)
lgbm_model.fit(X_res, y_res)
lgbm_predictions = lgbm_model.predict(X_test)
evaluation(y_test, lgbm_predictions)
```

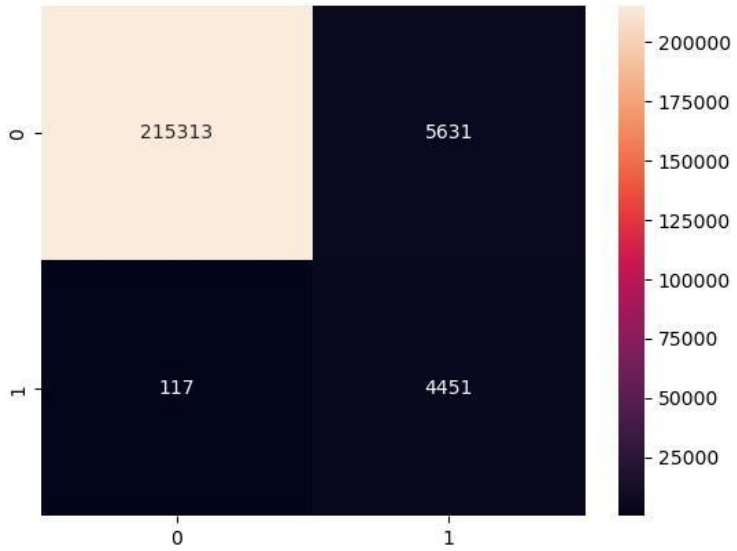
```
[LightGBM] [Info] Number of positive: 13704, number of negative: 13704
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.001239 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 3499
[LightGBM] [Info] Number of data points in the train set: 27408, number of used features: 20
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000
Accuracy:0.9211
Precision:0.2026
Recall:0.9869
F1 Score:0.3362
AUC-ROC:0.9533
```



```
lgbm_model2 = LGBMClassifier(is_unbalance=True, n_estimators=300, num_leaves=100)
lgbm_model2.fit(X_train, y_train)
lgbm2_predictions = lgbm_model2.predict(X_test)
evaluation(y_test, lgbm2_predictions)

# no is_unbalance
#Accuracy:0.9806
#Precision:0.6279
#Recall:0.1075
#F1 Score:0.1836
#AUC-ROC:0.5531
```

```
[LightGBM] [Info] Number of positive: 13704, number of negative: 662829
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.025149 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 4049
[LightGBM] [Info] Number of data points in the train set: 676533, number of used features: 20
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.020256 -> initscore=-3.878829
[LightGBM] [Info] Start training from score -3.878829
Accuracy:0.9745
Precision:0.4415
Recall:0.9744
F1 Score:0.6076
AUC-ROC:0.9745
```



Parametrelerde değişiklik yapıldığında ‘is_unbalance’ parametresi kullanılarak tüm veri seti üzerinde uygulanan modelin daha iyi performans verdiği görülmüştür. Yine de parametre optimizasyonunda her iki model de denenecektir.

3.5 HİPER PARAMETRE OPTİMİZASYONU

Parametre optimizasyonu için GridSearchCV algoritması kullanılmıştır. GridSearchCV ile modelde denenmesi istenen hiper parametreler ve değerleri için bütün kombinasyonlar ile ayrı ayrı model kurulur ve belirtilen metriğe göre en başarılı hiper parametre seti belirlenir. [27]

GridSearch içerisindeki iki önemli parametre 'cv' ve 'scoring' parametreleridir. Cv parametresi ile kaç katlı çapraz doğrulama yapılacağı belirlenir. Bu projede dengesiz veri seti kullanıldığı için stratified k-fold cross validation kullanılmıştır.

Stratified, özellikle sınıf dengesizliği olan veri kümeleri için kullanılır. Her katmanda sınıfların oranları eşitlenerek daha dengeli eğitim ve test setleri oluşturulur. Bu sayede modelin, tüm sınıfları daha doğru şekilde öğrenmesi sağlanır ve genelleme yeteneği artırılır. [28] Cross validation yöntemleri aşırı öğrenme (overfitting) riskini azaltmak için oldukça önemlidir.

```
sKFold = StratifiedKFold(n_splits=4, shuffle=False)#Default Parameters
```

GridSearch içerisinde kullanılacak scoring parametresi için klasik metrikler yerine "fbeta_score" metriğinin özel bir türü olan "f2_score" metriği kullanılmıştır. Scoring parametresinde recall kullanıldığından precision ve f1_score değerleri oldukça düşmekteydi. f1_score kullanıldığında ise aynı durum recall için geçerliydi. Bu proje için precision ve recall metriklerinin her ikisi de önemli olmakla birlikte recall metriği daha fazla öneme sahiptir. Çünkü false negatifler (FN), false pozitiflerden (FP) daha maliyetlidir. Yani modelin hasta olan birini sağlıklı olarak sınıflandırması, sağlıklı birini hasta olarak sınıflandırılmasından daha tehlikelidir.

```
f2 = make_scorer(fbeta_score, beta=2)
```

make_scorer metodu kullanılarak beta değeri 2 olan fbeta_score metriği oluşturulmuştur. Bu metriğin özel aldı f2_score olarak bilinir.

3.5.1 Random Forest

```
rfc_optimized_model = RandomForestClassifier()
```

```
rfc_param_grid = {
    'random_state':[42],
    'n_estimators': [100, 200, 300, 400],
    'min_samples_split': [2, 4, 5],
    'min_samples_leaf': [1, 2, 4],
    'max_depth': [None, 15, 30, 40],
    'max_features': ['auto', 'sqrt', 'log2']
}
```

random_state : Veri setinin her seferinde aynı rastgelelikte bölünmesini sağlar. (default=None)

n_estimators : Kullanılan ağaç sayısıdır. Daha fazla ağaç, daha iyi performans sağlar ancak eğitim süresini artırır ve aşırı öğrenme(overfitting) riski yükselir. (default=100)

min_samples_split : Bu parametre, bir düğümün bölünmesi için gerekli olan minimum örnek sayısıdır. Daha küçük bir değer, ağaçların daha derin olmasını sağlar ancak aşırı öğrenme(overfitting) riskini artırır. (default=2)

min_samples_leaf : Bu parametre, bir yaprak düğümünde bulunan minimum örnek sayısıdır. Daha küçük bir değer, ağaçların daha derin olmasını sağlar ancak eksik öğrenme (underfitting) riskini artırır. (default=1)

max_depth : Bu parametre, ağaçların maksimum derinliğidir. Daha derin ağaçlar, daha iyi bir performans sağlar ancak aşırı öğrenme (overfitting) riskini artırır. (default=None)

max_features : Bu parametre, her ağaçta kullanılan maksimum özellik sayısıdır. Daha fazla özellik, daha iyi bir performans sağlar ancak eğitim süresini artırır. (default=auto) [29]

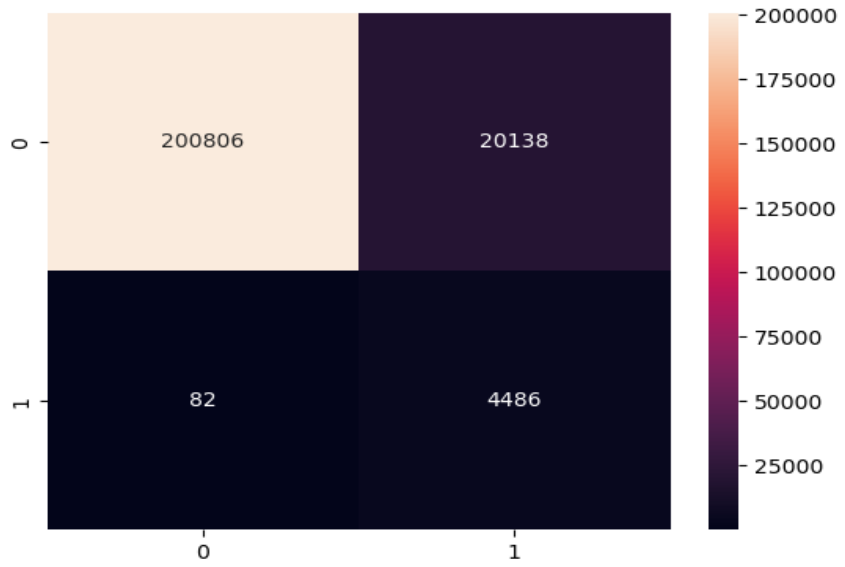
```
rfc_grid_search = GridSearchCV(estimator=rfc_optimized_model, param_grid=rfc_param_grid, cv=sKFold, scoring=f2, n_jobs=-1)
rfc_grid_search.fit(X_res, y_res)
```

```
print("Best parameters:", rfc_grid_search.best_params_)
print("Best Score (CV f2):", rfc_grid_search.best_score_)
```

```
Best parameters: {'max_depth': None, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 400, 'random_state': 42}
Best Score (CV f2): 0.9537859005377494
```

```
rfc_best_model = rfc_grid_search.best_estimator_
rfc_optimized_predictions = rfc_best_model.predict(X_test)
evaluation(y_test,rfc_optimized_predictions)
```

```
Accuracy:0.9103
Precision:0.1822
Recall:0.9820
F1 Score:0.3073
AUC-ROC:0.9455
```



3.5.2 XGBoost

```
xgb_optimized_model = XGBClassifier()
```

```
xgb_param_grid = {
        'scale_pos_weight' : [scale_pos_weight],
        'n_estimators': [100, 500, 750, 1000],
        'gamma': [0, 1, 5],
        'subsample': [0.7, 0.9],
        'max_depth': [6, 8, 10],
        'learning_rate': [0.2, 0.3],
        'min_child_weight': [1, 2]
}
```

scale_pos_weight: Pozitif sınıfın ağırlığını ayarlayan parametredir. Dengesiz veri setlerinde kullanılır. scale_pos_weight, genellikle pozitif sınıfın örnek sayısının negatif sınıfın örnek sayısına oranı olarak ayarlanır. (default=1)

n_estimators : Kullanılan ağaç sayısıdır. Daha fazla ağaç, daha iyi performans sağlar ancak eğitim süresini artırır ve aşırı öğrenme(overfitting) riski yükselir. (default=100)

gamma: Bir düğümün bölünmesinin minimum kayıp azalmasını kontrol eder. Gamma ne kadar büyükse model o kadar genelleyici olur ve aşırı öğrenme(overfitting) riski azalır.(default = 0)

subsample: Her ağacın eğitimi sırasında kullanılacak örneklerin oranını belirtir. Değeri 0 ile 1 arasında olmalıdır. Örneğin 0.7 değeri, her ağacın eğitimi için veri setinin %70'inin rastgele olarak seçileceği anlamına gelir. Bu, modelin aşırı uyum yapmasını engelleyebilir.(default=1)

max_depth: Her bir karar ağacının maksimum derinliğini belirtir. Daha büyük değerler modelin daha karmaşık hale gelmesine neden olabilir ve aşırı uyum riskini artırabilir. (default=6)

learning_rate: Her bir ağaç eklemesinden sonra yapılan düzeltmenin boyutunu belirler. Daha düşük bir öğrenme oranı daha doğru sonuçlar elde etmenizi sağlayabilir ancak daha fazla ağaç (n_estimators) gerektirebilir.(default=0.3)

min_child_weight: Bir yaprak düğümde bulunması gereken minimum örnek ağırlığını belirtir. Değer ne kadar yüksek olursa model o kadar düzenli hale gelir ve aşırı uyum yapma olasılığı azalır.(default=1) [30]

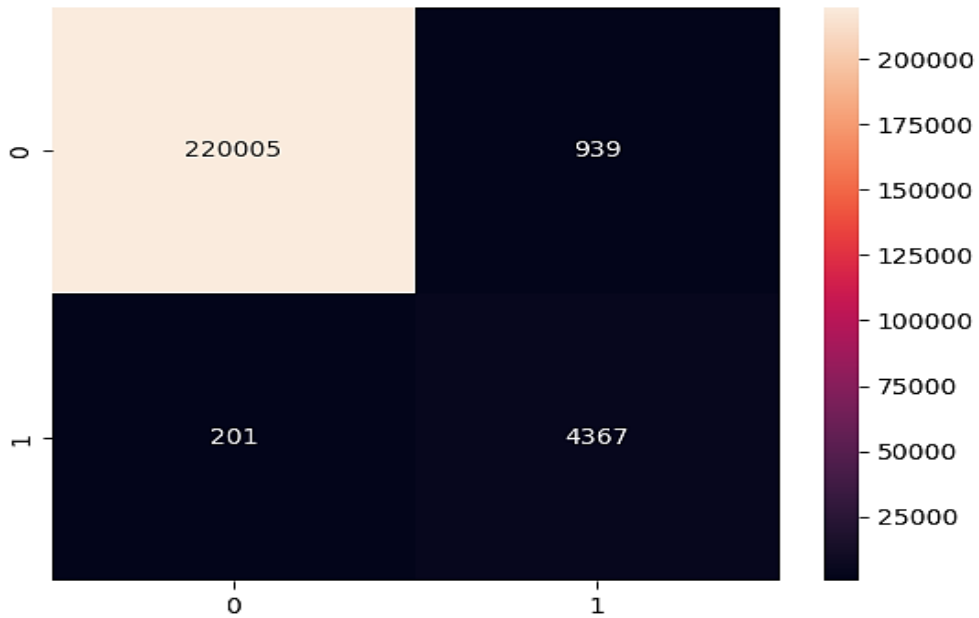
```
xgb_grid_search = GridSearchCV(estimator=xgb_optimized_model, param_grid=xgb_param_grid, cv=sKFold, scoring=f2, n_jobs=-1)
xgb_grid_search.fit(X_train, y_train)
```

```
print("Best parameters:", xgb_grid_search.best_params_)
print("Best Score (CV f2):", xgb_grid_search.best_score_)
```

```
Best parameters: {'gamma': 1, 'learning_rate': 0.2, 'max_depth': 8, 'min_child_weight': 2, 'n_estimators': 1000, 'scale_pos_wei
ght': 48.36755691768827, 'subsample': 0.9}
Best Score (CV f2): 0.8965412075202397
```

```
xgb_best_model = xgb_grid_search.best_estimator_
xgb_optimized_predictions = xgb_best_model.predict(X_test)
evaluation(y_test,xgb_optimized_predictions)
```

```
Accuracy:0.9949
Precision:0.8230
Recall:0.9560
F1 Score:0.8845
AUC-ROC:0.9759
```



3.5.3 LightGBM

İlk olarak, undersampling ile dengelenmiş veri seti kullanılarak optimizasyon yapılmıştır.

```
lgbm_optimized_model = LGBMClassifier()
```

```
lgbm_param_grid = {
    'n_estimators': [100, 250, 400],
    'num_leaves': [31, 100, 200],
    'learning_rate': [0.05, 0.1],
    'subsample' : [0.7, 0.9],
    'max_depth': [-1, 10, 20]
}
```

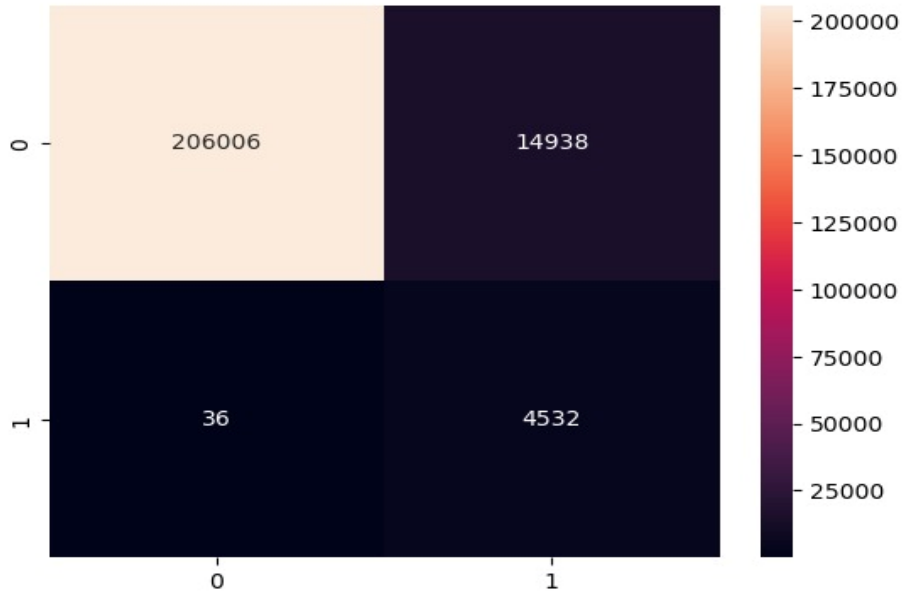
```
lgbm_grid_search = GridSearchCV(estimator=lgbm_model, param_grid=lgbm_param_grid, cv=sKFold, scoring=f2, n_jobs=-1)
lgbm_grid_search.fit(X_res, y_res)
```

```
print("Best parameters:", lgbm_grid_search.best_params_)
print("Best Score (CV f2):", lgbm_grid_search.best_score_)
```

```
Best parameters: {'learning_rate': 0.1, 'max_depth': -1, 'n_estimators': 400, 'num_leaves': 200, 'subsample': 0.7}
Best Score (CV f2): 0.9711370714513625
```

```
lgbm_best_model = lgbm_grid_search.best_estimator_
lgbm_predictions = lgbm_best_model.predict(X_test)
evaluation(y_test, lgbm_predictions)
```

```
Accuracy:0.9336
Precision:0.2328
Recall:0.9921
F1 Score:0.3771
AUC-ROC:0.9623
```



Bir sonraki adımda ‘is_unbalance' parametresi ile tüm veri seti kullanılarak optimizasyon yapılmıştır.

```
lgbm_optimized_model2 = LGBMClassifier()
```

```
lgbm_param_grid2 = {
    'n_estimators': [100, 500, 750, 1000],
    'num_leaves': [31, 100, 250, 500],
    'learning_rate': [0.05, 0.1],
    'subsample' : [0.7, 0.9],
    'max_depth': [-1, 15, 30],
    'is_unbalance': [True]
}
```

n_estimators : Kullanılan ağaç sayısıdır. Daha fazla ağaç, daha iyi performans sağlar ancak eğitim süresini artırır ve aşırı öğrenme(overfitting) riski yükselir. (default=100)

num_leaves : Her bir karar ağacındaki yaprakların maksimum sayısını belirler. Daha yüksek sayıda yaprak modelin karmaşıklığını artırır. Aşırı öğrenmeden kaçınmak için 2^(max_depth)'den küçük olması gerekmektedir. (default=31)

learning_rate : Her bir ağaç eklemesinden sonra yapılan düzeltmenin boyutunu belirler. Daha düşük bir öğrenme oranı , daha doğru sonuçlar elde etmenizi sağlayabilir ancak daha fazla ağaç (n_estimators) gerektirebilir. (default=0.1)

subsample: Her ağacın eğitimi sırasında kullanılacak örneklerin oranını belirtir. Değeri 0 ile 1 arasında olmalıdır. Örneğin, 0.7 değeri, her ağacın eğitimi için veri setinin %70'inin rastgele olarak seçileceği anlamına gelir. (default=1)

max_depth: Her bir karar ağacının maksimum derinliğini belirtir. -1 değeri, ağacın maksimum derinliğinin sınırlandırılmadığını gösterir. (default=-1)

is_unbalance: Veri setinin dengesiz olduğunu ve pozitif ve negatif sınıflar arasında denge sağlanması gerektiğini belirtir. Bu parametre True olarak ayarlanırsa sınıf dengesizliği otomatik olarak ele alınır. (default=False)

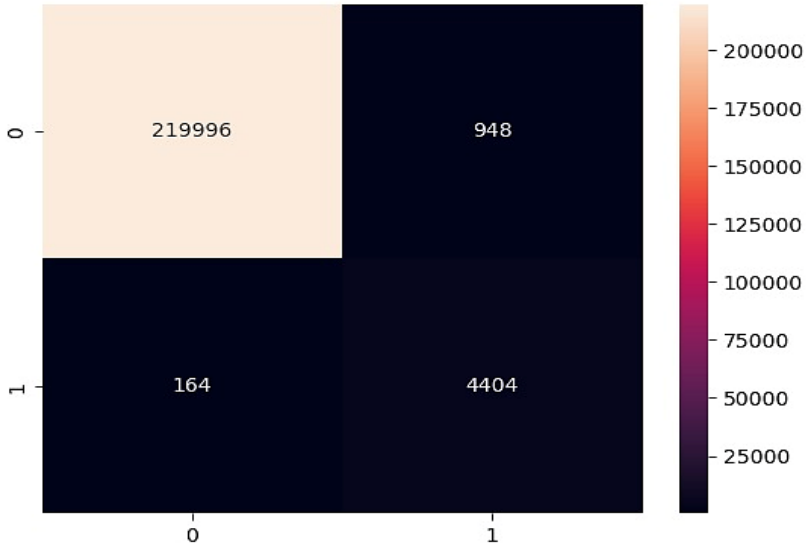
```
lgbm_grid_search2 = GridSearchCV(estimator=lgbm_optimized_model2, param_grid=lgbm_param_grid2, cv=sKFold, scoring=f2, n_jobs=-1)
lgbm_grid_search2.fit(X_train, y_train)
```

```
print("Best parameters:", lgbm_grid_search2.best_params_)
print("Best Score (CV f2):", lgbm_grid_search2.best_score_)
```

```
Best parameters: {'is_unbalance': True, 'learning_rate': 0.05, 'max_depth': 30, 'n_estimators': 750, 'num_leaves': 250, 'subsam
ple': 0.7}
Best Score (CV f2): 0.9111029253568768
```

```
lgbm_best_model2 = lgbm_grid_search2.best_estimator_
lgbm2_predictions = lgbm_best_model2.predict(X_test)
evaluation(y_test, lgbm2_predictions)
```

```
Accuracy:0.9951
Precision:0.8229
Recall:0.9641
F1 Score:0.8879
AUC-ROC:0.9799
```



BÖLÜM 4
SONUÇLAR

	Accuracy	Precision	Recall	F1-Score	AUC-ROC
KNN	%79.36	%8.60	%95.49	%15.79	%87.26
Lojistik Regresyon	%79.36	%6.21	%65.13	%11.33	%72.39
SVM	%81.85	%8.28	%79.01	%14.99	%80.46
Random Forest	%91.03	%18.22	%98.20	%30.73	%94.55
XGBoost	%99.49	**%82.30**	%95.60	%88.45	%97.59
LightGBM-1 (Undersampling)	%93.36	%23.28	**%99.21**	%37.71	%96.23
LightGBM-2 (is_unbalance)	**%99.51**	%82.29	%96.41	**%88.79**	**%97.99**

Şekil 12 Sonuçlar

Genel olarak geleneksel makine öğrenmesi algoritmaları, sınıf dengesizliğinden olumsuz etkilenmiş ve çok iyi sonuçlar verememiştir. Undersampling yöntemi, negatif sınıftaki bazı önemli bilgilerin kaybolmasına yol açtığından dolayı "precision" ve "F1_score" değerleri düşmüştür. Bununla birlikte KNN ve Random Forest algoritmalarının "recall" değeri oldukça yüksektir. Yani bu iki model pozitif sınıf için oldukça iyi sonuçlar vermektedir ancak çok sayıda false negatif (FN) değer içermektedir.

XGBoost ve LightGBM gibi boosting temelli modern algoritmalar, geleneksel algoritmalara oranla oldukça başarılı sonuçlar vermiştir. Her iki algoritmanın da sonuçları birbirine oldukça yakındır. Bu algoritmalar çok iyi "recall" değerine sahip olmakla birlikte "F1_score" değerleri de %90'a yakındır. Yani geleneksel algoritmalara oranla sonuçlar daha dengelidir. Bu algoritmaların accuracy (doğruluk) değeri de %99'un üzerindedir ancak dengesiz veri setleri için bu metrik yanıltıcı sonuçlar verebilir. Bu yüzden değerlendirme yapılırken diğer metrikler daha çok dikkate alınmıştır.

Bu çalışma ile birlikte makine öğrenmesi algoritmalarının sepsis hastalığını yüksek doğrulukla teşhis edebileceği ortaya konmuştur. Ayrıca, sağlık alanındaki birçok veri kümesi dengesiz sınıf dağılımı problemi içerdiğinden, bu çalışma diğer birçok hastalığın makine öğrenmesi ile teşhisi için de örnek teşkil etmektedir.

KAYNAKLAR

[1] https://www.medicalpark.com.tr/sepsis/hg-2169

[2] https://gbyf.org.tr/projects/20220123/

[3] https://acikerisim.bakircay.edu.tr/items/71ab0e88-40b0-496c-8ac1-cc2f023bc98a

[4] https://www.oracle.com/tr/artificial-intelligence/machine-learning/what-is-machine-learning/

[5] https://mervetatlidil.medium.com/makine-öğrenmesi-machine-learning-8960166d36d8

[6] https://miuul.com/not-defteri/ml101-makine-ogrenmesi-nedir

[7] https://medium.com/databulls/ makine-öğrenmesinde-gözetimli-ve-gözetimsiz-öğrenme-126a3dd41422

[8] https://medium.com/databulls/ makine-öğrenmesinde-gözetimli-ve-gözetimsiz-öğrenme-126a3dd41422

[9] https://medium.com/machine-learning-türkiye/knn-k-en-yakın-komşu-7a037f056116

[10] https://bulutistan.com/blog/lojistik-regresyon-nedir/

[11] https://mfakca.medium.com/ lojistik-regresyon-nedir-nasıl-çalışır-4e1d2951c5c1

[12] https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e

[13] https://medium.com/deep-learning-turkiye/ nedir-bu-destek-vektör-makineleri-makine-öğrenmesi-serisi-2-94e576e4223e

[14] https://ece-akdagli.medium.com/ makine-öğrenmesinde-random-forest-algoritması-a79b044bbb31

[15] https://www.javatpoint.com/machine-learning-random-forest-algorithm

[16] https://www.geeksforgeeks.org/xgboost/

[17] https://dergipark.org.tr/en/download/article-file/2259415

[18] https://www.veribilimiokulu.com/lightgbm/

[19] https://dataaspirant.com/lightgbm-algorithm/

[20] https://rohitgr7.github.io/lightgbm-another-gradient-boosting/

[21] https://www.odakarge.com/veri-seti-nedir-yapay-zeka-teknolojilerinde-nasil-kullanilir/

[22] https://www.kaggle.com/datasets/salikhussaini49/prediction-of-sepsis

[23] https://medium.com/@aman.gupta435433/performing-backward-filling-and-forward-filling-operations-in-pyspark-2065aafa9ff0

[24] https://medium.com/@mehmetcanduru/özellik-ölçeklendirme-standardscaler-robustscaler-ve-minmaxscaler-karşılaştırması -a2b40a56b550

[25] https://imbalanced-learn.org/stable/under_sampling.html#controlled-under-sampling

[26] Afshine Amidi and Shervine Amidi, Makine Öğrenmesi ipuçları ve püf noktaları El Kitabı VIP

[27] https://medium.com/bilişim-hareketi/hiperparametre-optimizasyonu-9ba0e7f32e6f

[28] https://miuul.com/blog/makine-ogrenmesinde-model-dogrulama-s%C3%BCrecleri-capraz-do%C4%9Frulama#:~:text=Stratified%20k-fold%20cross%20validation,sa%C4%9Flan%C4%B1r%20ve%20genelleme%20yetene%C4%9Fi%20art%C4%B1r%C4%B1l%C4%B1r.

[29] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[30] https://xgboost.readthedocs.io/en/stable/parameter.html