MUNZUR
ÜNİVERSİTESİ
2008

Uyku Bozukluğu Tahmini ve Sınıflandırma

Bu veri seti, uyku bozukluklarını tespit etmeye yönelik birden fazla modelin eğitimini ve performansını değerlendirmek amacıyla kullanılmıştır. Modelin başarısı, uyku bozukluğu olmayanları (None), uykusuzluğu olanları (Uykusuzluk) ve uyku apnesi olanları (Uyku Apnesi) doğru bir şekilde sınıflandırma yeteneği üzerine odaklanmaktadır.

Elde edilen sonuçlar, modelin her bir uyku bozukluğu türünü değerlendirmede ne kadar etkili olduğunu göstermektedir. Başarı oranı %92 olan bu model, farklı metrikler kullanılarak detaylı bir şekilde değerlendirilmiştir. Precision, recall ve f1-score gibi metrikler, modelin her bir sınıf için ne kadar doğru ve eksiksiz tahminler yaptığını ortaya koymaktadır. Bu değerler, modelin performansını anlamak ve geliştirmek için önemli ipuçları sağlamaktadır.

Bu dokümantasyon, modelin nasıl eğitildiğini, performansının nasıl değerlendirildiğini ve elde edilen sonuçları açıklamaktadır. Ayrıca, uyku bozukluklarıyla ilişkili faktörlerin anlaşılmasına yönelik faydalı bilgiler sunmaktadır.

Projenin adımları

- Kütüphanelerin eklenmesi

-Benzersiz değerlerin bulunması

-Veri ön temizleme

-Aykırı değerlerin kontrolü ve düzeltilmesi

-Kategorik değişkenleri sayısal değişkenlere dönüştürme

-Model oluşturma ve modeli eğitme

-Algoritmaların Kullanılması

-Verilerin görselleştirilmesi

-Model peroformansını değerlendirme

Hazırlayan: Medeni Aba

VERİ SETİ HAKKINDA

Veri Seti Genel Bakışı

Uyku Sağlığı ve Yaşam Tarzı Veri Seti, uyku ve günlük alışkanlıklarla ilgili çok çeşitli değişkenleri kapsayan 374 satır ve 13 sütundan oluşur. Cinsiyet, yaş, meslek, uyku süresi, uyku kalitesi, fiziksel aktivite seviyesi, stres seviyeleri, (Vücut Kitle İndeksi) BMI kategorisi, kan basıncı, kalp hızı, günlük adımlar ve uyku bozukluklarının varlığı veya yokluğu gibi ayrıntıları içerir.

Veri Seti Sütunları

Kişi Kimliği: Her bir birey için bir tanımlayıcı.

Cinsiyet: Kişinin cinsiyeti (Erkek/Kadın).

Yaş: Kişinin yıl cinsinden yaşı.

Meslek: Kişinin mesleği veya mesleği.

Uyku Süresi (saat): Kişinin günde uyuduğu saat sayısı.

Uyku Kalitesi (ölçek: 1-10): Uyku kalitesinin 1 ile 10 arasında değişen öznel bir değerlendirmesi.

Fiziksel Aktivite Seviyesi (dakika/gün): Kişinin günlük olarak fiziksel aktivitede bulunduğu dakika sayısı.

Stres Seviyesi (ölçek: 1-10): Kişinin deneyimlediği stres seviyesinin 1 ile 10 arasında değişen öznel bir değerlendirmesi.

Vücut Kitle İndeksi (BKİ) Kategorisi: Kişinin BKİ kategorisi (örn. Düşük Kilolu, Normal, Fazla Kilolu,Obez).

Kan Basıncı (sistolik/diyastolik): Kişinin kan basıncı ölçümü, sistolik basıncın diyastolik basınca oranı olarak gösterilir.

Kalp Hızı (bpm): Kişinin dinlenme kalp hızı, dakikadaki atış sayısı olarak.

Günlük Adım Sayısı: Kişinin günde attığı adım sayısı.

Uyku Bozukluğu: Kişide uyku bozukluğunun varlığı veya yokluğu (None, Uykusuzluk, Uyku Apnesi).

Uyku Bozukluğu Sütunu Hakkında Ayrıntılar;

None: Kişi herhangi bir spesifik uyku bozukluğu göstermez.

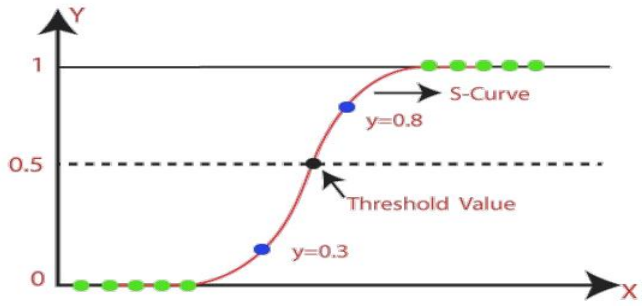
Uykusuzluk: Kişi uykuya dalmakta veya uykuda kalmakta zorluk çeker, bu da yetersiz veya kalitesiz uykuya yol açar.

Uyku Apnesi: Kişi uyku sırasında nefes almada duraklamalar yaşar, bu da uyku düzeninin bozulmasına ve potansiyel sağlık risklerine neden olur.

Kullanılan Algoritmalar

Logistic Regression

Lojistik regresyon, isminde “regresyon” geçmesine rağmen bir sınıflandırma algoritmasıdır. Yani görseldeki hayvanın kedi mi, köpek mi olduğu veya verilmiş olan bilgilerin bir erkeğe mi yoksa bir kadına mı ait olduğunu tahmin etme gibi iki sınıflı sınıflandırma problemlerinde sıkça kullanılır

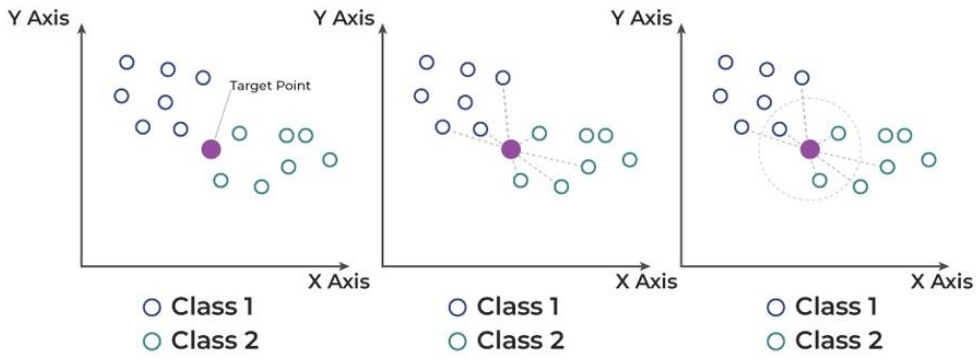


KNN-En Yakın Komşu Algoritması

KNN en basit anlamı ile içerisinde tahmin edilecek değerin bağımsız değişkenlerinin oluşturduğu vektörün en yakın komşularının hangi sınıfta yoğun olduğu bilgisi üzerinden sınıfını tahmin etmeye dayanır.KNN (K-Nearest Neighbors) Algoritması iki temel değer üzerinden tahmin yapar;

Distance (Uzaklık): Tahmin edilecek noktanın diğer noktalara uzaklığı hesaplanır. Bunun için Minkowski uzaklık hesaplama fonksiyonu kullanılır.

K (komşuluk sayısı): En yakın kaç komşu üzerinden hesaplama yapılacağını söyleriz. K değeri sonucu direkt etkileyecektir. K 1 olursa overfit etme olasılığı çok yüksek olacaktır. Çok büyük olursa da çok genel sonuçlar verecektir. Bu sebeple optimum K değerini tahmin etmek problemin asıl konusu olarak karşımızda durmaktadır.



XGBoost Algoritması

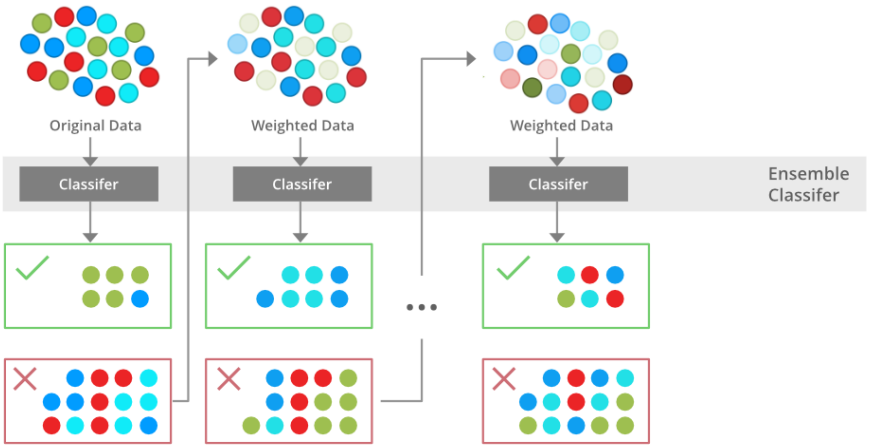
XGBoost (Extreme Gradient Boosting), makine öğrenmesinde sıklıkla kullanılan güçlü bir **gradient boosting** algoritmasıdır. Basitçe söylemek gerekirse, birçok zayıf öğrenciyi (genellikle karar ağaçları) bir araya getirerek çok daha güçlü bir tahmin modeli oluşturmayı hedefler.

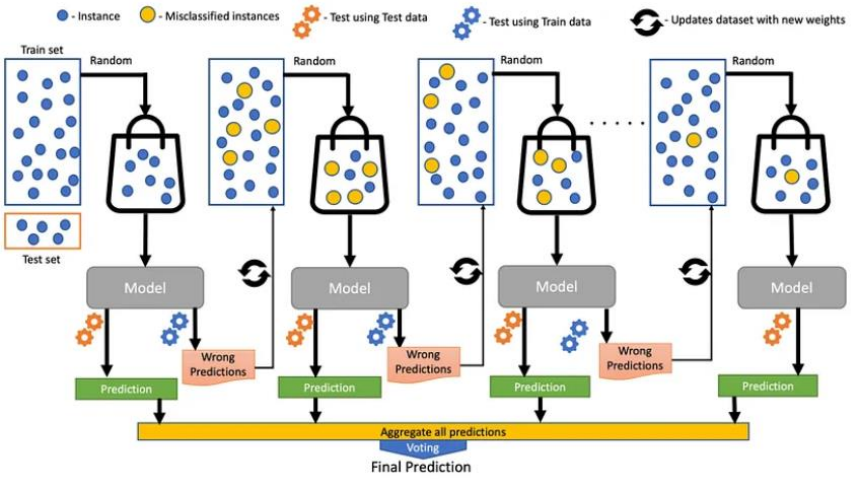
Nasıl Çalışır?

- **İteratif Öğrenme:** XGBoost, her adımda yeni bir model ekleyerek önceki modelin hatalarını düzeltmeye çalışır. Bu süreç, bir merdiven çıkmak gibi düşünebilirsiniz. Her basamak, modeli biraz daha iyi hale getirir.
- **Gradyan İniş:** Yeni bir model eklerken, hangi yönde ilerleyeceğimize karar vermek için gradyan iniş yöntemini kullanır. Bu, en hızlı şekilde en iyi çözüme ulaşmamızı sağlar.
- **Karar Ağaçları:** XGBoost, temel olarak karar ağaçlarını kullanır. Bu ağaçlar, verileri bölerek ve sınıflandırarak kararlar verir.

Neden XGBoost?

- **Yüksek Performans:** Büyük ve karmaşık veri setlerinde bile çok iyi sonuçlar verir.
- **Esneklik:** Sınıflandırma, regresyon gibi birçok farklı problem türüne uygulanabilir.
- **Düzenlileştirme:** Modelin aşırı karmaşıklaşmasını önler, böylece yeni verilere daha iyi genelleme yapabilir.





Özellik	KNN (En Yakın Komşular)	XGBoost (Extreme Gradient Boosting)	Lojistik Regresyon
Temel Fikir	Yeni bir veri noktasının sınıfını, en yakın komşularının çoğunluk sınıfına göre belirler.	Birçok zayıf öğrenciyi (genellikle karar ağaçları) birleştirerek güçlü bir model oluşturur.	Bir olayın gerçekleşme olasılığını lojistik bir fonksiyon kullanarak tahmin eder.
Model Eğitimi	Eğitim verisi üzerinde tüm hesaplamaları yapar. Yeni bir veri geldiğinde, tekrar en yakın komşuları bulur.	İteratif bir süreçle, her adımda yeni bir model ekleyerek modeli geliştirir.	Tüm eğitim verisi kullanılarak tek seferde model eğitimi yapılır.
Model Karmaşıklığı	K değeri (komşu sayısı) ile kontrol edilir. K küçükse model daha karmaşık, büyükse daha basit olur.	Düzenlileştirme parametreleriyle kontrol edilir.	Modelin karmaşıklığı, kullanılan özellik sayısı ve düzenlileştirme terimleriyle ilişkilidir.
Ölçeklenebilirlik	Büyük veri setlerinde yavaş olabilir, özellikle yüksek boyutlu verilerde.	Büyük veri setlerinde iyi çalışır.	Genellikle büyük veri setlerinde iyi çalışır.
Yorumlanabilirlik	Modelin kararlarını anlamak kolaydır (hangi komşular etkiledi).	Modelin kararlarını anlamak daha zordur, çünkü birçok karar ağacının birleşimiyle oluşur.	Modelin kararlarını anlamak daha kolaydır, lojistik regresyon katsayıları özelliklerin önemini gösterir.
Hiperparametreler	K değeri, uzaklık metriği gibi.	Öğrenme oranı, ağaç derinliği, düzenlileştirme parametreleri gibi birçok hiperparametre.	Düzenlileştirme parametreleri ve özellikleri seçme gibi.
Uygulama Alanları	Sınıflandırma, regresyon.	Sınıflandırma, regresyon, sıralama gibi birçok alanda kullanılır.	Sınıflandırma, özellikle ikili sınıflandırma problemlerinde kullanılır.

Kullanılan Yöntemler

Veri Ön İşlemede Kullanılan Yöntemler

1-Robust Scaler Nedir?

Robust Scaler, veri setindeki **aşırı uç değerlerin** etkisini en aza indirmek için kullanılan bir normalizasyon tekniğidir. Bu yöntem, verileri **ortanca** ve **çeyrekler arası aralığa** göre ölçeklendirir. Standart Scaler veya Min-Max Scaler gibi yöntemlerin aksine, aşırı uç değerlerden fazla etkilenmez.

Robust Scaler Nasıl Çalışır?

- Verinin merkezi eğilimi, **ortalama** yerine **ortanca** ile ifade edilir. Ortanca, aşırı uç değerlerden etkilenmez ve veri setinin tam ortasındaki değeri temsil eder.
- Verinin yayılımını ölçmek için **çeyrekler arası aralık (IQR)** kullanılır.
- IQR, verinin 1. çeyrek (Q1, %25) ve 3. çeyrek (Q3, %75) arasındaki farkını ifade eder.

$$IQR = Q3 - Q1$$

Her bir veri noktası şu şekilde ölçeklendirilir:

$$X_{scaled} = \frac{X - \text{Median}(X)}{\text{IQR}(X)}$$

Burada:

- X: Orijinal veri
- $\text{Median}(X)$: Verinin ortanca değeri
- $\text{IQR}(X)$: Çeyrekler arası aralık

2-OneHotEncoder

Kategorik (niteliksel) verileri, makine öğrenmesi algoritmalarının işleyebilmesi için **sayısal (niceliksel)** bir forma dönüştüren bir kodlama yöntemidir. Her kategori, **0** ve **1**'den oluşan ayrı bir sütun olarak temsil edilir.

Her kategori için ayrı bir sütun oluşturulur ve gözlemde bulunan kategori sütununda 1, diğer sütunlarda 0 yazılır.

Neden Kullanılır?

- **Makine öğrenmesi modellerinin** çoğu, kategorik verilerle doğrudan çalışamaz. OneHotEncoder, bu kategorik verileri sayısal bir forma çevirerek modellerin anlamasına olanak tanır.
- Özellikle kategoriler arasında **sıralama ilişkisi olmayan** durumlar için uygundur. (Örneğin: "Renk" kategorisinin kırmızı, mavi ve yeşil arasında sıralama ilişkisi yoktur.)

Dengesizlik Verilerin İşlenmesi

SMOTE (Synthetic Minority Oversampling Technique)

Makine öğreniminde dengesiz veri setleri ile çalışırken kullanılan bir yeniden örnekleme (resampling) yöntemidir. Özellikle, sınıf dengesizliği olan veri setlerinde azınlık sınıfındaki örneklerin sayısını artırmak için kullanılır.

Neden SMOTE Kullanılır?

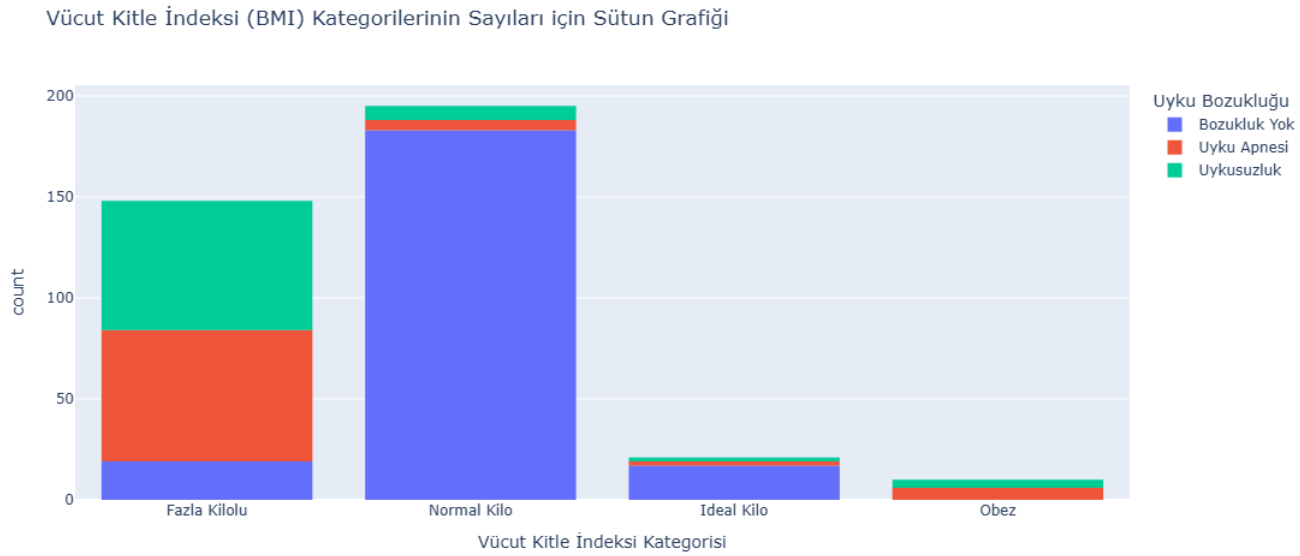
Sınıf dengesizliği, bir sınıfın diğer sınıflara kıyasla çok daha az sayıda örneğe sahip olduğu durumlarda ortaya çıkar. Bu durum, makine öğrenimi algoritmalarının çoğunluk sınıfa daha fazla odaklanmasına ve azınlık sınıfını ihmal etmesine yol açabilir.

SMOTE, azınlık sınıfındaki örneklerin sayısını artırarak bu sorunu çözmeyi hedefler. Ancak bunu **orijinal örnekleri kopyalamak yerine, yeni örnekler oluşturarak** yapar.

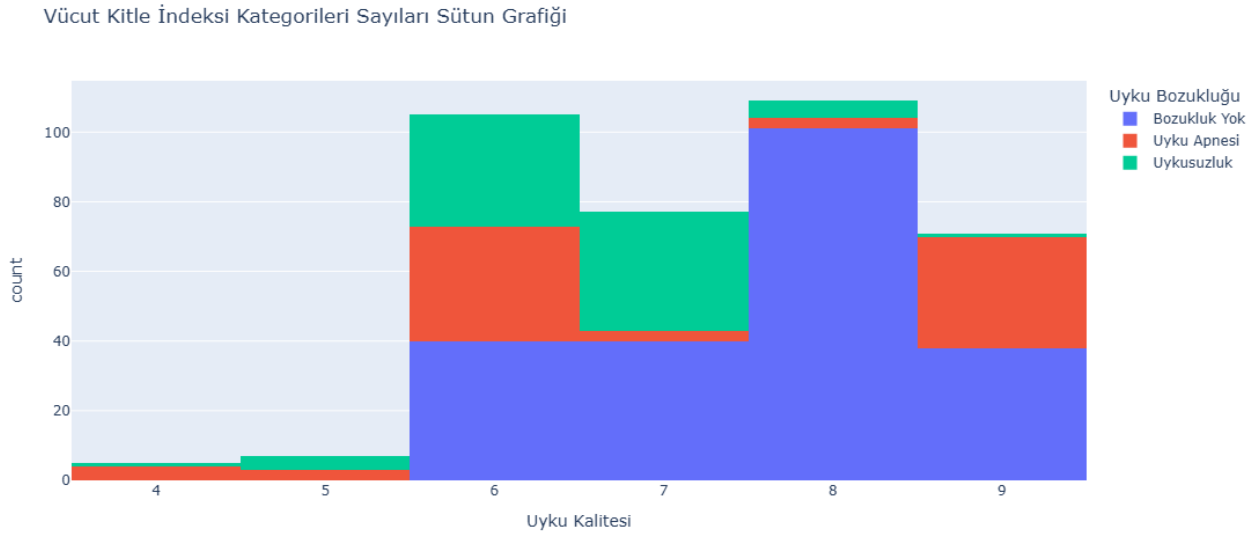
SMOTE'un Avantajları

1. **Sınıf Dengesizliğini Giderir:**
    - Azınlık sınıfının öğrenilmesini kolaylaştırır ve modelin performansını artırabilir.
2. **Overfitting'i Azaltır:**
    - Azınlık sınıfındaki örnekleri kopyalamak yerine sentetik örnekler oluşturduğu için aşırı öğrenme (overfitting) riskini azaltır.
3. **Esnek ve Genel Amaçlı:**
    - Birçok makine öğrenimi algoritmasıyla kolayca entegre edilebilir.

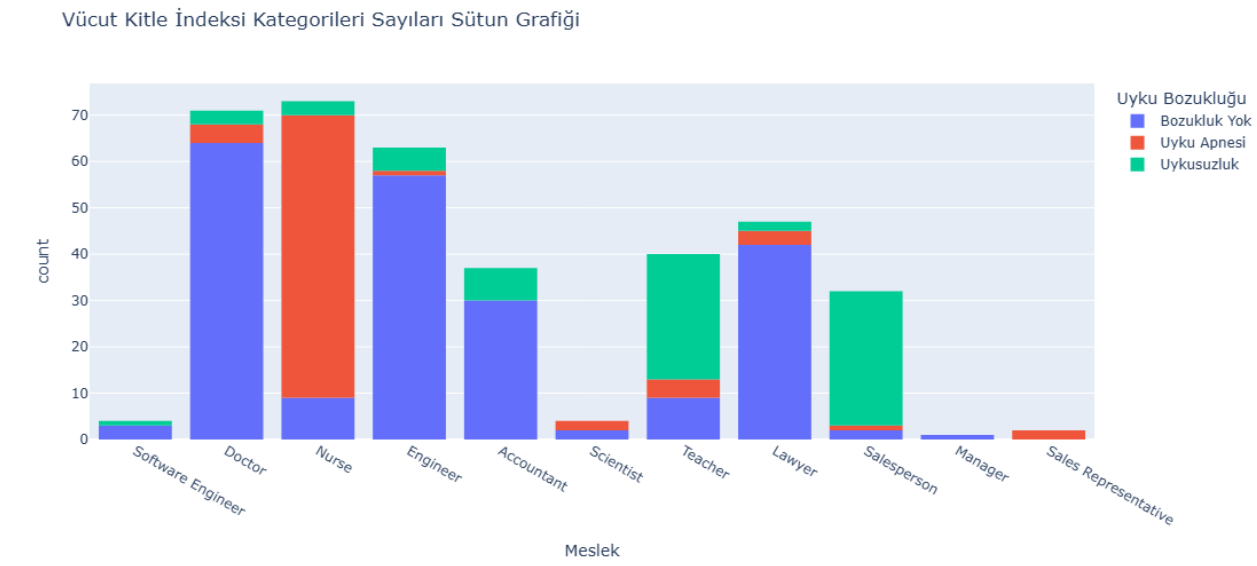
GRAFİKLER



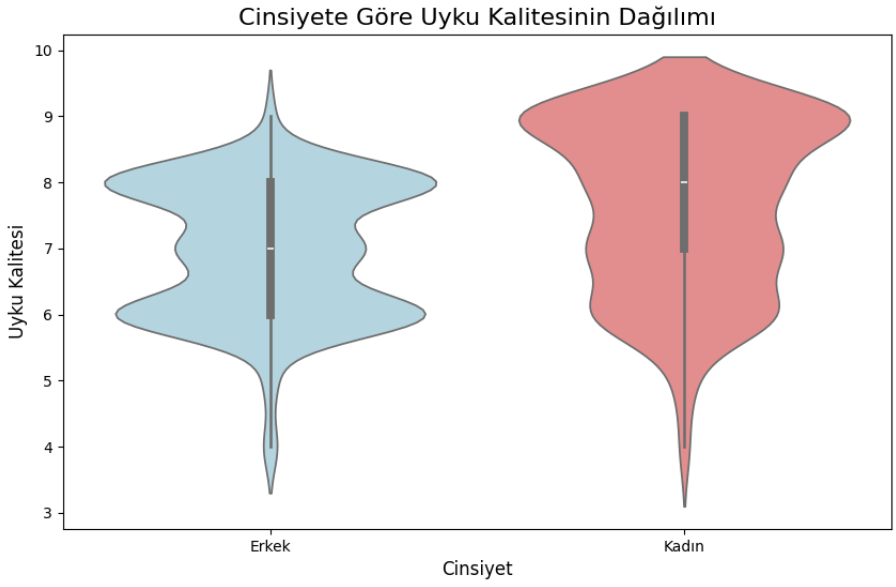
Şekil 1.1 Farklı vücut kitle indeksi (BMI) kategorilerinde uyku bozukluğu türlerinin dağılımını gösteren bir yığılmış sütun grafiğidir.



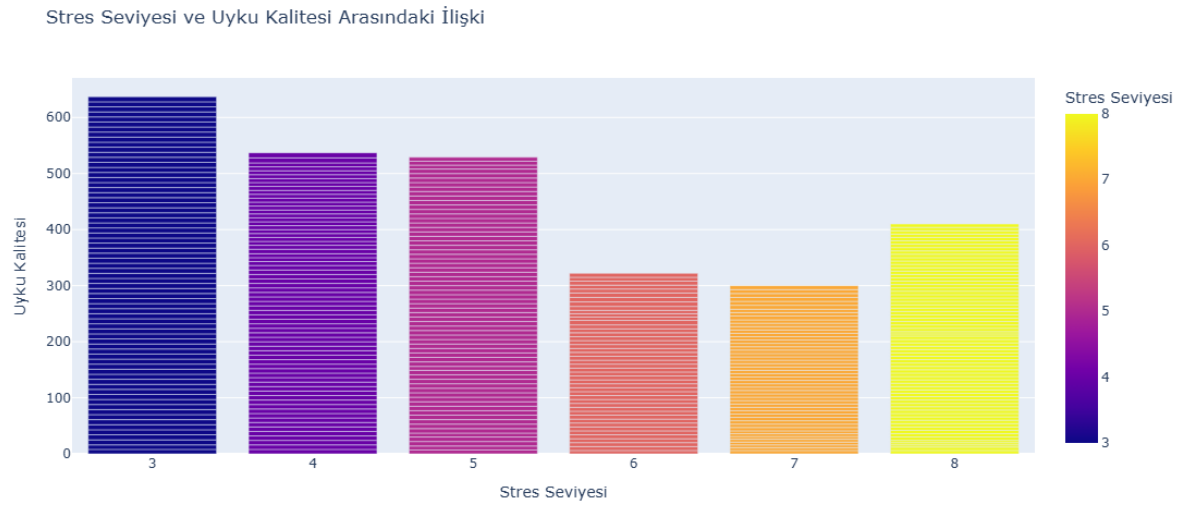
Şekil 1.2 Uyku kalitesiyle uyku bozukluklarının ilişkilendirilmesini ve kategorilere göre sayısal dağılım grafiği



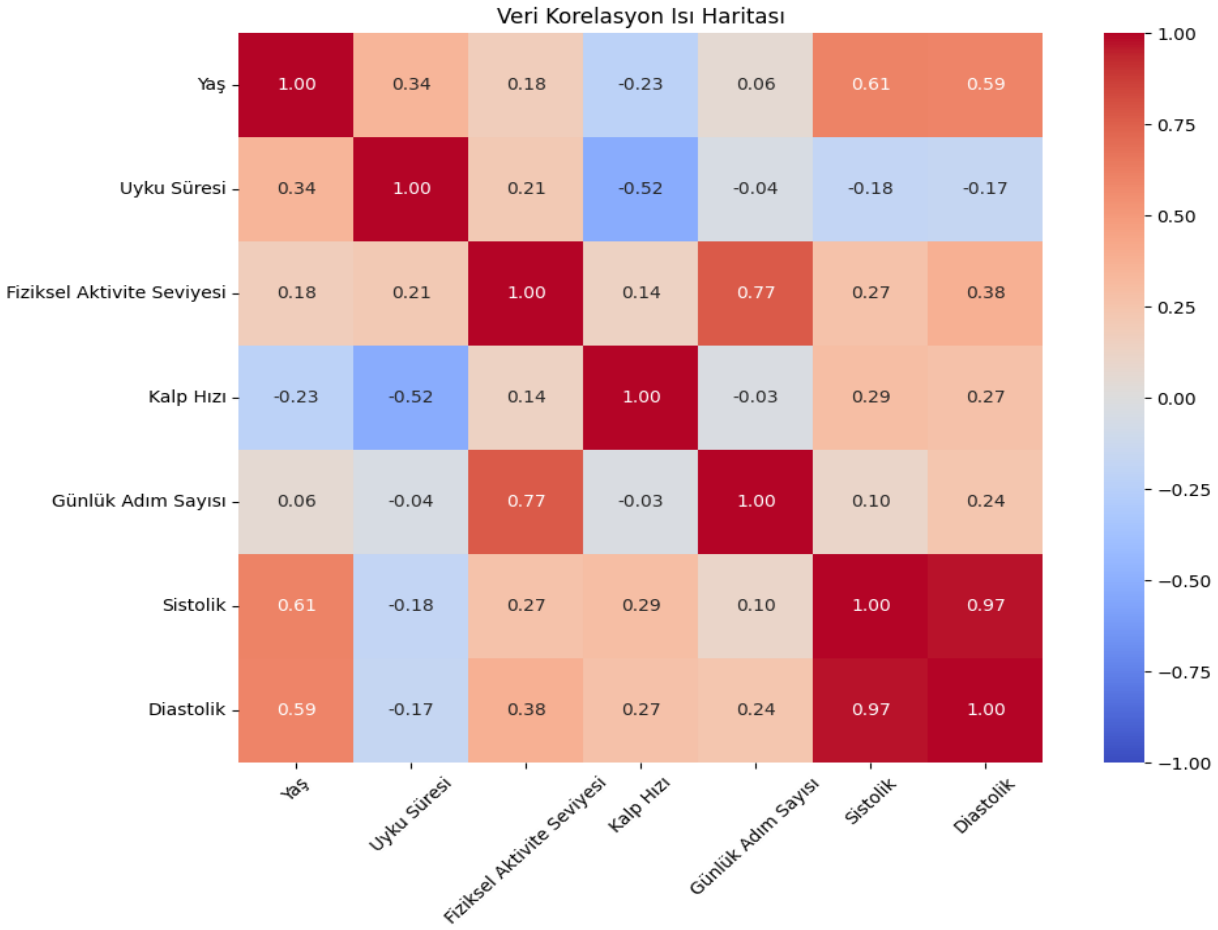
Şekil 1.3 Farklı meslek gruplarındaki bireylerin vücut kitle indeksi kategorilerine göre uyku bozukluğu durumlarını karşılaştıran bir sütun grafiğidir.



Şekil 1.4- Cinsiyete göre uyku kalitesinin dağılımını gösteren bir viyol grafiği

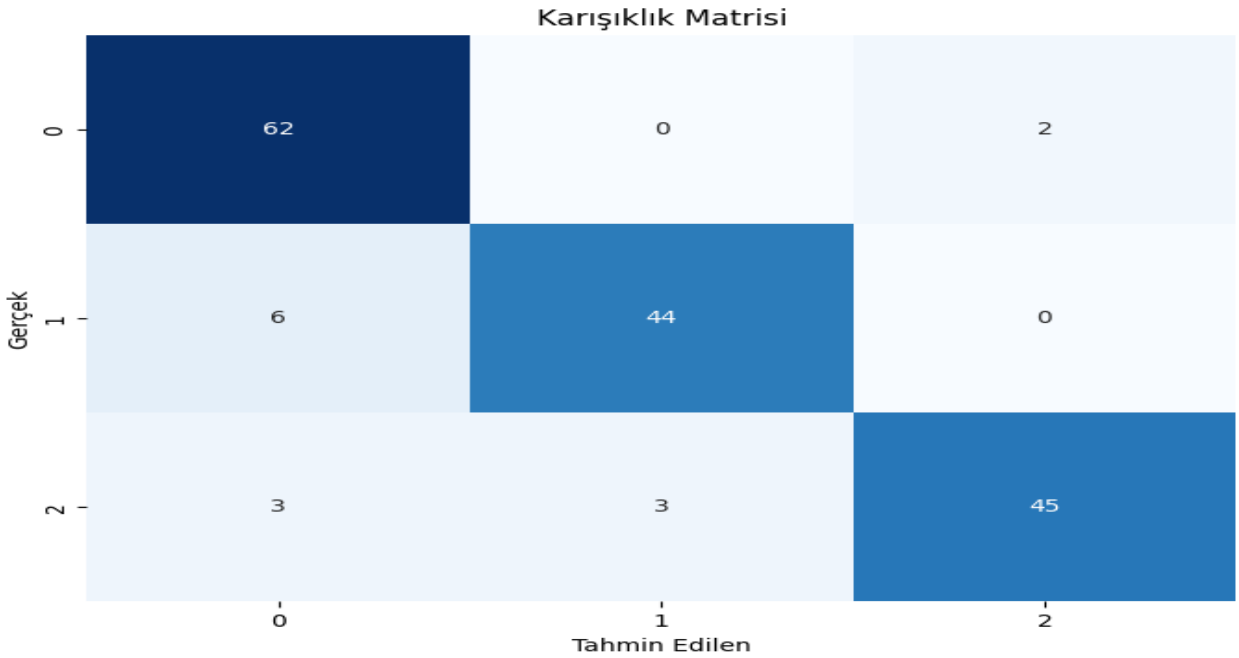


Şekil 2.1 - Stres seviyesi ile uyku kalitesi arasındaki ilişkiyi gösteren bir çubuk grafiği



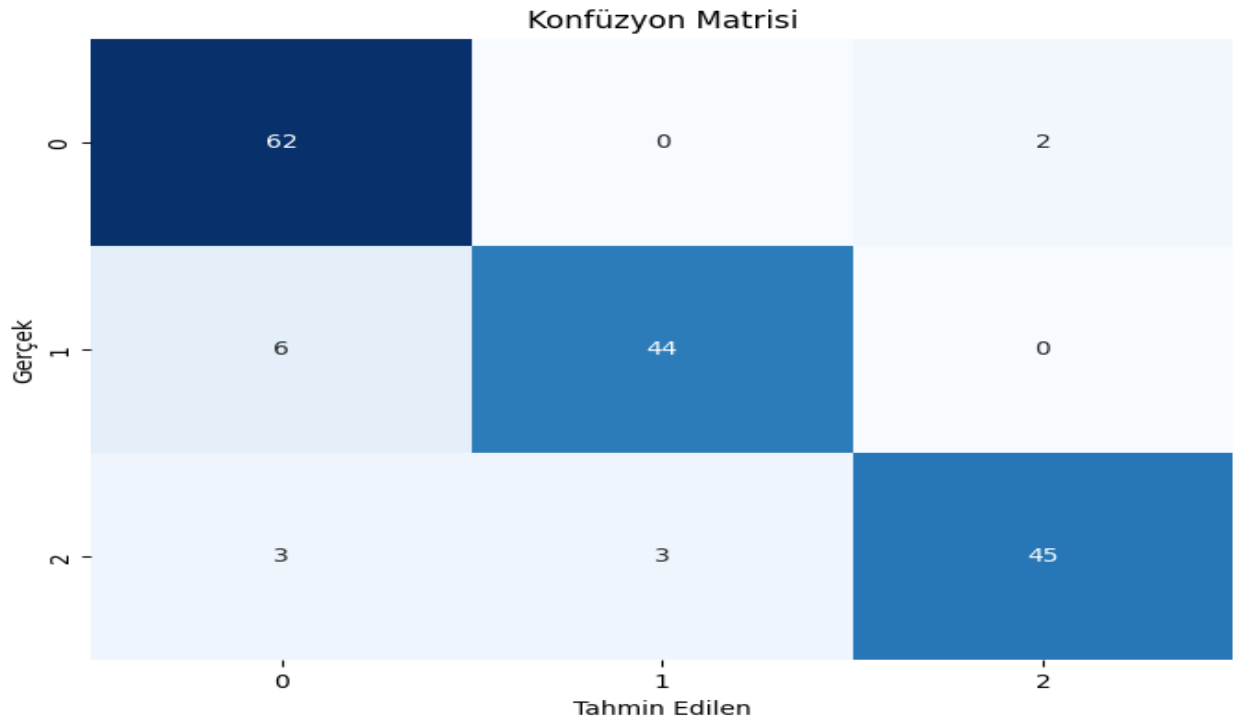
Şekil 2.2- Farklı özellikler arasındaki korelasyonları gösteren bir ısı haritası grafiği

XGB Classifier Konfüzyon Matrisi



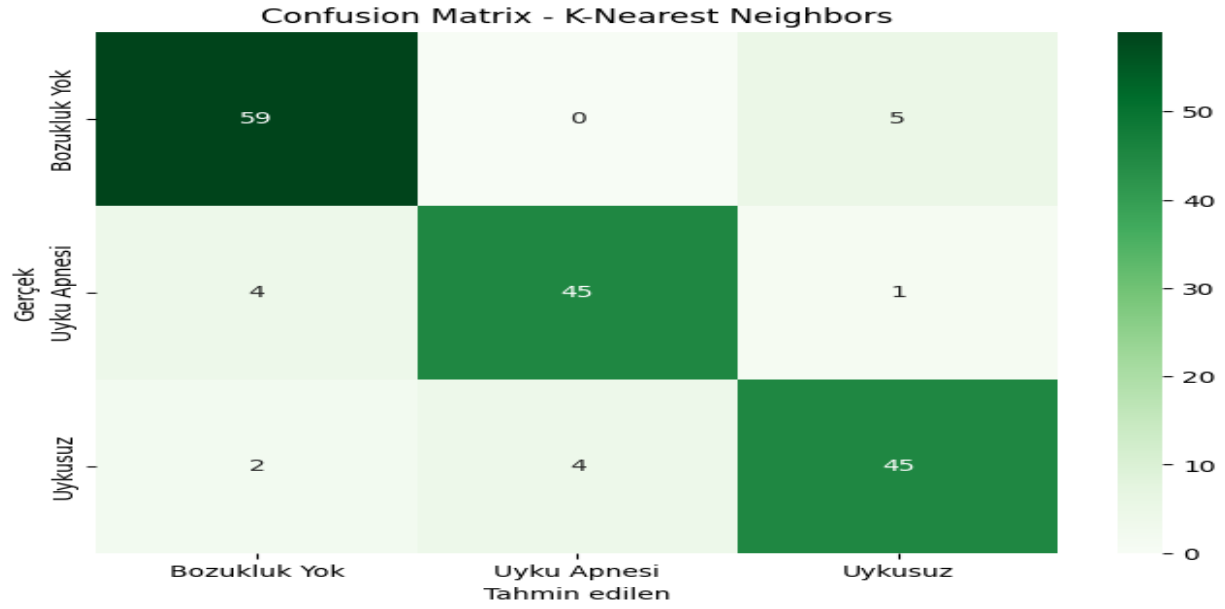
Şekil 3.1 - Makine öğrenmesi modelinin performansını gösteren bir karışıklık matrisi; modelin doğru ve yanlış tahminlerini, her sınıf için gerçekteki değerlerle karşılaştırarak özetler.

Lojistik Regresyon Confüzyon Matrisi

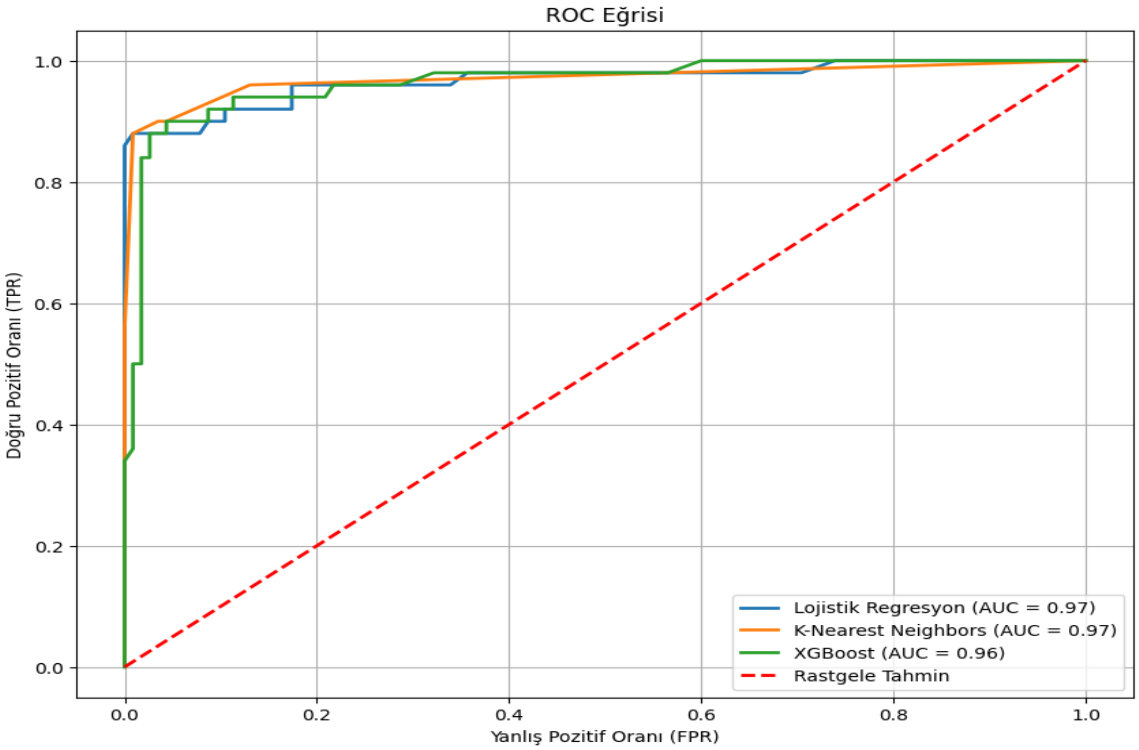


Şekil 3.2 - Tahmin başarı oranını gösteren bir konfüzyon matrisi, gerçek sınıflar ile modelin tahmin ettiği sınıflar arasındaki birebir karşılaştırmayı sunar.

KNN Konfüzyon Matrisi



Şekil 3.3 K-En Yakın Komşu (KNN) algoritması kullanılarak yapılan bir sınıflandırma modelinin performansını gösteren bir karışıklık matrisi, modelin gerçek ve tahmin edilen sınıflar arasındaki doğruluk oranlarını özetler.



Şekil 4 Doğru pozitif oran (TPR) ile yanlış pozitif oran (FPR) arasındaki ilişkiyi gösterir ve farklı algoritmaların başarılarını karşılaştırmak için AUC (Eğri Altındaki Alan) değerlerini içerir.

Lojistik Regresyon ve **KNN** modellerinin AUC değerleri (0.97 , 0.97) oldukça iyi. Bu modeller, sınıfları çok iyi ayırt edebiliyor. **XGBoost** AUC değeri (0.96) biraz daha düşük, ancak yine de çok iyi bir performansa sahip.

Kütüphaneleri Ekleme

```
import numpy as np  # lineer cebir
import pandas as pd  # veri işleme, CSV dosyalarını okuma
import os
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import OneHotEncoder,LabelEncoder ,RobustScaler,StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split,StratifiedShuffleSplit,StratifiedKFold,cross_val_score
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression
import xgboost as xgb
```

Veri Setini Okuma İlk 5 Satını Görüntüleme

```
[2] # Türkçe karakterler için 'cp1254' kodlamasıyla dosyayı okuma
    df = pd.read_csv('/content/Sleep_health_and_lifestyle_dataset.csv', encoding='cp1254')
```

```
[3] df.head(5)
```

	Kişi ID	Cinsiyet	Yaş	Meslek	Uyku Süresi	Uyku Kalitesi	Fiziksel Aktivite Seviyesi	Stres Seviyesi	Vücut Kitle İndeksi Kategorisi	Kan Basıncı	Kalp Hızı	Günlük Adım Sayısı	Uyku Bozukluğu
0	1	Erkek	27	Software Engineer	6.1	6	42	6	Fazla Kilolu	126/83	77	4200	NaN
1	2	Erkek	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	NaN
2	3	Erkek	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	NaN
3	4	Erkek	28	Sales Representative	5.9	4	30	8	Obez	140/90	85	3000	Uyku Apnesi
4	5	Erkek	28	Sales Representative	5.9	4	30	8	Obez	140/90	85	3000	Uyku Apnesi

Veri seti satır ve sütun görüntüleme

```
[4] df.shape #(satır,sütun)
```

```
(374, 13)
```

Veri seti sütun veri tiplerini görüntüleme

```
df.dtypes
```

	0
Kişi ID	int64
Cinsiyet	object
Yaş	int64
Meslek	object
Uyku Süresi	float64
Uyku Kalitesi	int64
Fiziksel Aktivite Seviyesi	int64
Stres Seviyesi	int64
Vücut Kitle İndeksi Kategorisi	object
Kan Basıncı	object
Kalp Hızı	int64
Günlük Adım Sayısı	int64
Uyku Bozukluğu	object

Veri seti temel istatistik değerleri getirme

```
[6] df.describe() # Temel İstatistik değerleri getirir
```

	Kişi ID	Yaş	Uyku Süresi	Uyku Kalitesi	Fiziksel Aktivite Seviyesi	Stres Seviyesi	Kalp Hızı	Günlük Adım Sayısı
count	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000
mean	187.500000	42.184492	7.132086	7.312834	59.171123	5.385027	70.165775	6816.844920
std	108.108742	8.673133	0.795657	1.196956	20.830804	1.774526	4.135676	1617.915679
min	1.000000	27.000000	5.800000	4.000000	30.000000	3.000000	65.000000	3000.000000
25%	94.250000	35.250000	6.400000	6.000000	45.000000	4.000000	68.000000	5600.000000
50%	187.500000	43.000000	7.200000	7.000000	60.000000	5.000000	70.000000	7000.000000
75%	280.750000	50.000000	7.800000	8.000000	75.000000	7.000000	72.000000	8000.000000
max	374.000000	59.000000	8.500000	9.000000	90.000000	8.000000	86.000000	10000.000000

Sütun düzenlemesi ve benzersiz kelimelerin bulunması

```
[7] columns = [column for column in df.columns if column!='Kişi ID'] #'Kişi ID' sütunu hariç sütun bilgilerini tuttuğumuz yeni liste
```

```
[8] for column in columns:   #Sütunlarda gezinip onlara ait benzersiz kelimeleri buluyoruz
        unique_values = df[column].unique()
        print(f"Unique values in '{column}': {unique_values}")
```

```
Unique values in 'Cinsiyet': ['Erkek' 'Kadın']
Unique values in 'Yaş': [27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 48 49 50 51 52
 53 54 55 56 57 58 59]
Unique values in 'Meslek': ['Software Engineer' 'Doctor' 'Sales Representative' 'Teacher' 'Nurse'
 'Engineer' 'Accountant' 'Scientist' 'Lawyer' 'Salesperson' 'Manager']
Unique values in 'Uyku Süresi': [6.1 6.2 5.9 6.3 7.8 6.  6.5 7.6 7.7 7.9 6.4 7.5 7.2 5.8 6.7 7.3 7.4 7.1
 6.6 6.9 8.  6.8 8.1 8.3 8.5 8.4 8.2]
Unique values in 'Uyku Kalitesi': [6 4 7 5 8 9]
Unique values in 'Fiziksel Aktivite Seviyesi': [42 60 30 40 75 35 45 50 32 70 80 55 90 47 65 85]
Unique values in 'Stres Seviyesi': [6 8 7 4 3 5]
Unique values in 'Vücut Kitle İndeksi Kategorisi': ['Fazla Kilolu' 'Normal' 'Obez' 'Ideal Kilo']
Unique values in 'Kan Basıncı': ['126/83' '125/80' '140/90' '120/80' '132/87' '130/86' '117/76' '118/76'
 '128/85' '131/86' '128/84' '115/75' '135/88' '129/84' '130/85' '115/78'
 '119/77' '121/79' '125/82' '135/90' '122/80' '142/92' '140/95' '139/91'
 '118/75']
Unique values in 'Kalp Hızı': [77 75 85 82 70 80 78 69 72 68 76 81 65 84 74 67 73 83 86]
Unique values in 'Günlük Adım Sayısı': [ 4200 10000  3000  3500  8000  4000  4100  6800  5000  7000  5500  5200
  5600  3300  4800  7500  7300  6200  6000  3700]
Unique values in 'Uyku Bozukluğu': [nan 'Uyku Apnesi' 'Uykusuzluk']
```

```
[9] # 'Uyku Bozukluğu' sütunundaki NaN değerlerini 'Bozukluk Yok' ile değiştir
    df['Uyku Bozukluğu'].fillna('Bozukluk Yok', inplace=True)
```

```
[10] # 'Uyku Bozukluğu' sütununun değer sayıları
     uyku_bozuklugu_sayilari = df['Uyku Bozukluğu'].value_counts()

     print("'Uyku Bozukluğu' Değer Sayıları:")
     print(uyku_bozuklugu_sayilari)  #Insomnia ( Uykusuzluk ) - Sleep Apnea ( Uyku Apnesi ) - No Disorder ( Uyku Sorunu Yok )
```

```
'Uyku Bozukluğu' Değer Sayıları:
Uyku Bozukluğu
Bozukluk Yok    219
Uyku Apnesi      78
Uykusuzluk       77
Name: count, dtype: int64
```

```
# 'Vücut Kitle İndeksi Kategorisi' sütunundaki 'Normal' değerini 'Normal Kilo' ile değiştir
df['Vücut Kitle İndeksi Kategorisi'] = df['Vücut Kitle İndeksi Kategorisi'].replace({'Normal': 'Normal Kilo'})

# 'Vücut Kitle İndeksi Kategorisi' değerlerinin sayısını yazdır
df['Vücut Kitle İndeksi Kategorisi'].value_counts()
```

	count
Vücut Kitle İndeksi Kategorisi	
Normal Kilo	195
Fazla Kilolu	148
Ideal Kilo	21
Obez	10

Kan basıncı değerinin model daha iyi öğrensin diye 2 sütuna ayırılması

```
[ ]  df = pd.concat([df, df['Kan Basıncı'].str.split('/', expand=True)], axis=1).drop('Kan Basıncı', axis=1)
     df = df.rename(columns={0: 'Sistolik', 1: 'Diastolik'}) #axis=1 sütun bazında verileri ayırmak için kullanırız 0 olsa satır olurdu
```

```
[ ]  df['Sistolik'] = df['Sistolik'].astype(float)
     df['Diastolik'] = df['Diastolik'].astype(float)
```

Veri Önişleme

```
[ ]  df.drop(columns=['Kişi ID'],inplace = True) #Modelin yanlış öğrenmesinden kaçınmak için 'Kişi Id' sildik
```

```
[ ]  label_encoder = LabelEncoder() #Modelin öğrenebilmesi adına kategorik verileri sayısal verilere dönüştürüyoruz
     df['Uyku Bozukluğu'] = label_encoder.fit_transform(df['Uyku Bozukluğu'])
```

```
[ ]  print(label_encoder.classes_)
```

```
['Bozukluk Yok' 'Uyku Apnesi' 'Uykusuzluk']
```

ÖN İŞLEME

```
[ ]  sayisal_ozellikler = ['Yaş', 'Uyku Süresi',
                           'Kalp Hızı', 'Günlük Adım Sayısı', 'Sistolik', 'Diastolik']

     kategorik_ozellikler = ['Meslek', 'Uyku Kalitesi', 'Cinsiyet',
                             'Fiziksel Aktivite Seviyesi', 'Stres Seviyesi', 'Vücut Kitle İndeksi Kategorisi']
```

```
[ ]  ön_işleme = ColumnTransformer( #Farklı sütunlara farklı dönüşümler uygulamak için Column Transformer uygulandı
         transformers=[
             ('sayisal', RobustScaler(), sayisal_ozellikler), #Sayısal değerleri RobutScaler ile ölçeklendirdik
             ('kategorik', OneHotEncoder(drop='first', sparse_output=False, handle_unknown='ignore'), kategorik_ozellikler)
         ])
              #Kategorik değerleri de onehot encoder 0-1 lere dönüştürdük
```

```
[ ]  X = df.drop(columns=['Uyku Bozukluğu']) #Uyku bozukluğu hariç diğer sütun bilgilerini X de tutuyoruz
     y = df['Uyku Bozukluğu'] #Ulaşmak istediğimiz yer uyku bozukluğu zaten o yüzden onu ayrı y de tutuyoruz , ögrenme öncesi önemli bir adım.
```

```
[ ]  X_islenmis = ön_işleme.fit_transform(X) #Makine öğrenmesinde verilecek işlenmiş verilerimiz
```

Dengesiz Verilerin İşlenmesi

Dengesizlik Verilerinin İşlenmesi

```
[ ]  # SMOTE'yi başlat
     smote = SMOTE(random_state=42) #Rastgelelik

     # Smote sonrası mevcut veri setinin dengelenmiş yeni hali
     X_smote, y_smote = smote.fit_resample(X_islenmis, y)
     X_smote.shape  #yeni oluşan dengelenmiş
```

```
(657, 45)
```

```
[ ]  # Veriyi eğitim ve test setlerine ayırma (örneğin, %75 eğitim, %25 test)
     X_train, X_test, y_train, y_test = train_test_split(X_smote, y_smote, test_size=0.25, random_state=42)
```

Model Eğitme

Logistic Regresyon

```
# Lojistik Regresyon sınıflandırıcıyı başlatma
model_lr = LogisticRegression()

# Modeli eğitim verisi üzerinde eğitme
model_lr.fit(X_train, y_train)

# Test verisi üzerinde tahmin yapma
y_pred_lr = model_lr.predict(X_test)

# Değerlendirme metriklerini hesaplama
doğruluk_lr = accuracy_score(y_test, y_pred_lr)
kesinlik_lr = precision_score(y_test, y_pred_lr, average='weighted')
geri_bulma_lr = recall_score(y_test, y_pred_lr, average='weighted')
f1_skoru_lr = f1_score(y_test, y_pred_lr, average='weighted')

# Metrikleri yazdırma
print(f'Doğruluk (Accuracy): {doğruluk_lr}')
print(f'Kesinlik (Precision): {kesinlik_lr}')
print(f'Geri Bulma (Recall): {geri_bulma_lr}')
print(f'F1-Skoru (F1-score): {f1_skoru_lr}')
print(" ")

# Sınıflandırma raporunu oluşturma
print(classification_report(y_test, y_pred_lr))
```

```
Doğruluk (Accuracy): 0.9151515151515152
Kesinlik (Precision): 0.918337101915166
Geri Bulma (Recall): 0.9151515151515152
F1-Skoru (F1-score): 0.9150469370989746

              precision    recall  f1-score   support

           0       0.87      0.97      0.92        64
           1       0.94      0.88      0.91        50
           2       0.96      0.88      0.92        51

    accuracy                           0.92       165
   macro avg       0.92      0.91      0.91       165
weighted avg       0.92      0.92      0.92       165
```

XGB

```python
# XGBoost sınıflandırıcı modelini başlat
model_xgb = xgb.XGBClassifier()

# Modeli eğitim verisiyle eğit
model_xgb.fit(X_train, y_train)

# Test verisi üzerinde tahmin yap
y_pred = model_xgb.predict(X_test)

# Modelin değerlendirme metriklerini hesapla
accuracy_xgb = accuracy_score(y_test, y_pred)  # Doğruluk
precision_xgb = precision_score(y_test, y_pred, average='weighted')  # Kesinlik
recall_xgb = recall_score(y_test, y_pred, average='weighted')  # Hatırlama (Recall)
f1_xgb = f1_score(y_test, y_pred, average='weighted')  # F1 Skoru

# Metrikleri yazdır
print(f'Doğruluk (Accuracy): {accuracy_xgb}')
print(f'Kesinlik (Precision): {precision_xgb}')
print(f'Hatırlama (Recall): {recall_xgb}')
print(f'F1 Skoru: {f1_xgb}')
print(" ")

# Sınıflandırma raporunu yazdır
print(classification_report(y_test, y_pred))
```

```
Doğruluk (Accuracy): 0.9151515151515152
Kesinlik (Precision): 0.918337101915166
Hatırlama (Recall): 0.9151515151515152
F1 Skoru: 0.9150469370989746

              precision    recall  f1-score   support

           0       0.87      0.97      0.92        64
           1       0.94      0.88      0.91        50
           2       0.96      0.88      0.92        51

    accuracy                           0.92       165
   macro avg       0.92      0.91      0.91       165
weighted avg       0.92      0.92      0.92       165
```

KNN

```
[40] from sklearn.neighbors import KNeighborsClassifier

     # Initialize KNN classifier (example using k=5)
     knn_clf = KNeighborsClassifier(n_neighbors=5)

     # Train the model
     knn_clf.fit(X_train, y_train)

     # Predictions
     y_pred = knn_clf.predict(X_test)

     # Calculate metrics
     accuracy_knn = accuracy_score(y_test, y_pred)
     precision_knn = precision_score(y_test, y_pred, average='weighted')
     recall_knn = recall_score(y_test, y_pred, average='weighted')
     f1_knn = f1_score(y_test, y_pred, average='weighted')

     # Classification report
     print("Classification Report:")
     print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.92      0.91        64
           1       0.92      0.90      0.91        50
           2       0.88      0.88      0.88        51

    accuracy                           0.90       165
   macro avg       0.90      0.90      0.90       165
weighted avg       0.90      0.90      0.90       165
```

Genel Doğruluk

```
[ ]  from sklearn.metrics import accuracy_score

     # Modellerin doğruluk değerlerini hesaplama ve yazdırma
     for idx, model in enumerate([model_lr, knn_clf, model_xgb]):
         y_pred = model.predict(X_test)
         accuracy = accuracy_score(y_test, y_pred)
         print(f"{modeller[idx]}: Doğruluk = {accuracy:.2f}")
```

```
Lojistik Regresyon: Doğruluk = 0.92
K-Nearest Neighbors: Doğruluk = 0.90
XGBoost: Doğruluk = 0.92
```