



**T.C.
AKDENİZ ÜNİVERSİTESİ
TIP FAKÜLTESİ
KULAK, BURUN VE BOĞAZ
HASTALIKLARI ANA BİLİM DALI**

**MAKİNE ÖĞRENMESİ İLE AHİ (APNE-HİPOPNE İNDEKSİ)
HESAPLAMASI YAPILMADAN POLİSOMNOGRAFİ
VERİLERİNDEN AHİ TAHMİNİ: FİZYOLOJİK
PARAMETRELERE DAYALI REGRESYON MODELİ
GELİŞTİRİLMESİ**

UZMANLIK TEZİ

Dr. Ahmet Uras BALIK

Antalya, 2025



T.C.
AKDENİZ ÜNİVERSİTESİ
TIP FAKÜLTESİ
KULAK, BURUN VE BOĞAZ
HASTALIKLARI ANA BİLİM DALI

**MAKİNE ÖĞRENMESİ İLE AHİ (APNE-HİPOPNE İNDEKSİ)
HESAPLAMASI YAPILMADAN POLİSOMNOGRAFİ
VERİLERİNDEN AHİ TAHMİNİ: FİZYOLOJİK
PARAMETRELERE DAYALI REGRESYON MODELİ
GELİŞTİRİLMESİ**

UZMANLIK TEZİ

Dr. Ahmet Uras BALIK

Tez Danışmanı: Doç. Dr. Levent RENDA

‘Kaynak Gösterilerek Tezimden Yararlanılabilir’

Antalya, 2025

TEŞEKKÜR

Uzmanlık eğitimim ve tez sürecim boyunca bilgi birikimi, akademik rehberliği ve değerli katkılarıyla bana yol gösteren tez danışmanım ve Ana Bilim Dalı Başkanımız Sayın Doç. Dr. Levent RENDA'ya en içten saygı ve teşekkürlerimi sunarım. Bilimsel yaklaşımı, akademik vizyonu ve desteği bu çalışmanın şekillenmesinde belirleyici olmuştur.

Uzmanlık eğitimim süresince bilgi ve deneyimlerinden yararlanma fırsatı bulduğum Akdeniz Üniversitesi Tıp Fakültesi Kulak Burun Boğaz Hastalıkları Anabilim Dalı'ndaki tüm değerli hocalarıma,

Birlikte çalışmaktan mutluluk duyduğum, zor zamanları birlikte yürüdüğüm asistan arkadaşlarıma,

Çalışmanın yürütülmesinde katkı sağlayan uyku laboratuvarı ekibine ve veri sürecindeki destekleri için tüm çalışma arkadaşlarıma,

Tıp eğitimim ve uzmanlık sürecim boyunca her koşulda yanımda olan ve aldığım her kararda beni destekleyen annem Emine BALIK'a, babam Aziz BALIK'a ve ablam Melis BALIK KEŞÇİ'ye sonsuz teşekkür ederim.

İÇİNDEKİLER

	<u>Sayfa</u>
Simgeler ve Kısaltmalar Dizini	vii
Şekiller Dizini	ix
Çizelgeler Dizini	X
	xi
1. GİRİŞ VE AMAÇ	1
1.1. Obstrüktif Uyku Apnesi (OSA): Tanımı, Epidemiyolojisi ve Toplumsal Yükü	1
1.2. Apne-Hipopne İndeksi (AHI) ve Tanıdaki Sınırlılıkları	2
1.3. Polisomnografi (PSG) Sürecinin Zorlukları	3
1.4. AHI Hesabına Katılmayan Fizyolojik Değişkenlerle Tahmin Yaklaşımı	4
1.5. Bu Çalışmanın Hedefi ve Özgün Katkısı	6
1.6. AHI Hesaplama Yöntemi, Skorlama Kuralları ve Fiziksel Tanımlar	6
1.7. AHI'nin Klinik ve Tanısal Sınırlılıkları	7
2. GENEL BİLGİLER	9
2.1. Değişkenler ve Özellikleri	9
2.1.1. Solunumsal Değişkenler (SpO ₂ , Desatürasyon)	9
2.1.2. Kardiyovasküler Parametreler	9
2.1.3. Uyku Evresi ve Pozisyon Verileri	10
2.1.4. Diğer Fizyolojik Değişkenler	10
2.2. Model Kurulum Değişkenleri	11

2.2.1. Vücut Kitle İndeksi (VKİ)	11
2.2.2. Uyku Latansı (Sleep Latency)	11
2.2.3. REM Latansı (REM Latency)	12
2.2.4. Uyku Periyodundaki Uyanıklık Süresi (WASO – Wake After Sleep Onset)	12
2.2.5. REM Evresi Süresi (REM Sleep Duration)	13
2.2.6. Toplam Arousal Sayısı (Total Arousal Count)	13
2.2.7. Sırtüstü Uyku Pozisyonu	13
2.2.8. Uyanıklık Satürasyon Ortalaması (Awake SpO ₂ Mean)	14
2.2.9. En Düşük Oksijen Satürasyonu Yüzdesi (Lowest Oxygen Saturation Percentage)	14
2.2.10. Ortalama Oksijen Desatürasyonunun Yüzdesi (Average Oxygen Desaturation Percentage)	15
2.2.11. Oksijen Desatürasyon Yüzdesi ≥ 3 Olay Sayısı (Oxygen Desaturation Percentage ≥ 3 Events)	15
2.2.12. REM Evresindeki Ortalama Kalp Hızı (Mean Heart Rate During REM Sleep)	16
2.2.13. NREM Evresindeki Ortalama Kalp Hızı (Mean Heart Rate During NREM Sleep)	16
2.2.14. Uykudaki Toplam Bacak Hareketi Sayısı (Total Leg Movements During Sleep)	17
2.2.15. Yaş (Age)	17
2.2.16. Cinsiyet (Sex)	17
2.3. Makine Öğrenmesine Genel Bakış ve Sağlık Alanındaki Uygulamalar	18
2.3.1. Sağlıkta Makine Öğrenmesi Kullanım Örnekleri (EKG, ICU, SpO ₂ vb.)	18
2.3.2. Uyku Tıbbında Makine Öğrenmesi Kullanımı	19

2.3.3. Bu Çalışmada Kullanılacak Modellerin Seçilme Gerekçeleri	20
2.3.4. Seçilen Modellerin Temel Teknik ve Klinik Farklılıkları	20
2.3.5. Makine Öğrenmesi ile AHİ Tahmini: Literatürdeki Regresyon ve Sınıflama Modelleri	21
2.3.6. Kullanılan ML Modelleri ve Performans Değerlendirme Yöntemleri	22
2.3.7. Mevcut Yaklaşımların Literatürdeki Sınırlılıkları	23
3. GEREÇ VE YÖNTEM	25
3.1. Genel Yaklaşım ve Analiz	27
3.2. Varsayım Kontrolleri	27
3.3. Grup Karşılaştırmaları	27
3.4. İlişki Analizleri	27
3.5. Çok Değişkenli Modeller	28
3.6. Makine Öğrenmesi ve Hiperparametre Optimizasyonu	28
3.6. Performans Ölçütleri	28
3.7. Klinik Senaryo Analizi	28
3.8. Yapay Zeka Açıklanabilirliği	28
4. BULGULAR	29
4.1. Veri Yükleme ve Doğrulama	29
5. TARTIŞMA	109
5.1. Genel Değerlendirme ve Çalışmanın Önemi	109
5.2. Tanımlayıcı İstatistiklerin ve Hasta Özelliklerinin Değerlendirilmesi	110
5.3. Normallik, Varyans, Grup Karşılaştırmaları ve İstatistiksel Analizler	110

5.4. Korelasyon Analizleri ve Çoklu Doğrusal Bağlantı (VIF)	111
5.5. Regresyon Modellerinin Bulguları	112
5.6. Makine Öğrenmesi Modelleri ve Performans Karşılaştırmaları	112
5.7. Klinik Senaryo Analizi ve Karar Destek Sistemlerine Katkı	113
5.8. Yapay Zeka Açıklanabilirliği (Explainable AI – SHAP Bulguları)	114
5.9. Global Literatür ve Türkiye’deki Çalışmalar ile Kıyaslaması	115
5.10. Çalışmanın Güçlü Yanları, Sınırlılıkları ve Gelecek Önerileri	118
6. SONUÇLAR	120
7. ÖZET	121
8. ABSTRACT	123
9. KAYNAKLAR	125
10. EKLER	143
Ek 1. Akdeniz Üniversitesi Tıp Fakültesi Klinik Araştırmalar Etik Kurul Onayı	143
Ek 2. Colab Kodlamaları	144

SİMGELER VE KISALTMALAR DİZİNİ

AASM	Amerikan Uyku Tıbbı Akademisi
AHI	Apne Hipopne İndeksi
CPAP	Continuous Positive Airway Pressure
DCA	Desicion Curve Analysis
EEG	Elektroensefalografi
EKG	Elektrokardiyografi
EMG	Elektromiyografi
EOG	Elektrookülografi
HRV	Kalp Hızı Değişkenliği
HSAT	Home Sleep Apnea Test
MAD	Mandibular İlerletme Apreyi
MAE	Mean Absolute Error
MI	Miyokard Enfarktüsü
ML	Makine Öğrenmesi
MSE	Hataların Karesinin Ortalaması
NREM	Non-Rapid Eye Movement
ODİ	Oksijen Desatürasyon İndeksi
OSA	Obstrüktif Uyku Apnesi
PLMD	Periyodik Bacak Hareketi Bozukluğu
POSA	Pozisyonel Obstrüktif Uyku Apnesi
PSG	Polisomnografi

R^2	Determination Coefficient
REM	Rapid Eye Movement
RF	RandomForest
RLS	Huzursuz Bacak Sendromu
RMSE	Root Mean Squared Error
ROC-AUC	Receiver Operating Characteristic-Area Under Curve
SHAP	SHapley Additive exPlanations
SMOTE	Synthetic Minority Oversampling Technique
SVM	Destek Vektör Makinesi
SVR	Support Vector Regression
VIF	Çoklu Doğrusal Bağlantı
VKI	Vücut Kitle İndeksi
WASO	Uyku Periodundaki Uyku Süresi
XAI	Açıklanabilir Yapay Zeka
XGBoost	Extreme Gradient Boosting
YZ	Yapay Zeka
AASM	Amerikan Uyku Tıbbı Akademisi
AHI	Apne Hipopne İndeksi
CPAP	Continuous Positive Airway Pressure

ŞEKİLLER DİZİNİ

<u>Şekil</u>	<u>Sayfa</u>
4.1. VKİ	33
4.2. Uyku latansı	34
4.3. Rem latansı	35
4.4. Uyku periyodunda ki uyanıklık süresi	36
4.5. REM evresi süresi	37
4.6. Arousal sayısı toplam	38
4.7. Uyku süresinde pozisyon sırtüstü süre (dk)	39
4.8. SaO2 uyanıklık ortalaması yüzdesi	40
4.9. En düşük SaO2 yüzdesi	41
4.10. Ortalama SaO2 desaturasyonu yüzdesi	42
4.11. SaO2 desaturasyonu yüzdesi ≥ 3 olay sayısı	43
4.12. Ortalama kalp hızı genel REM	44
4.13. Ortalama kalp hızı genel NREM	45
4.14. Toplam bacak hareketi sayısı uyku	46
4.15. Yaş	47
4.16. Spearman korelasyon ısı haritası	52
4.17. RandomForest – OOF ROC	59
4.18. RandomForest – OOF PR	59
4.19. RandomForest – OOF kalibrasyon	60
4.20. XGBoost – OOF ROC	61
4.21. XGBoost – OOF PR	61

4.22.	XGBoost – OOF kalibrasyon	62
4.23.	CatBoost – OOF ROC	63
4.24.	CatBoost – OOF PR	63
4.25.	CatBoost – OOF kalibrasyon	64
4.26.	OOF ROC-AUC	65
4.27.	OOF PR-AUC	65
4.28.	OOF Brier	66
4.29.	RandomForest (REG) – gerçek vs OOF tahmin (45 ° çizgili)	67
4.30.	RandomForest (REG) – OOF rezidü histogram	67
4.31.	XGBoost (REG) – gerçek vs OOF tahmin (45 ° çizgili)	68
4.32.	XGBoost (REG) – OOF rezidü histogram	69
4.33.	CatBoost (REG) – gerçek vs OOF tahmin (45 ° çizgili)	70
4.34.	CatBoost (REG) – OOF rezidü histogram	70
4.35.	Regresyon modelleri	72
4.36.	OOF ROC-AUC	76
4.37.	OOF PR-AUC	76
4.38.	OOF Brier	77
4.39.	OOF RMSE (↓ daha iyi)	79
4.40.	OOF MAE (↓ daha iyi)	80
4.41.	OOF R ² (↑ daha iyi)	80
4.42.	Sınıflama modeli: RandomForest	81
4.43.	Sınıflama modeli: RandomForest	82
4.44.	Sınıflama modeli: XGBoost	85

4.45.	Sınıflama modeli: XGBoost	86
4.46.	Sınıflama modeli: CatBoost	89
4.47.	Sınıflama modeli: CatBoost	90
4.48.	Regresyon modeli: RamdomForest	93
4.49.	Regresyon modeli: RamdomForest	94
4.50.	Regresyon modeli: XGBoost	97
4.51.	Regresyon modeli: XGBoost	98
4.52.	Regresyon modeli: CatBoost	101
4.53.	Regresyon modeli: CatBoost	102
4.54.	Desicion Curve Analizi (DCA)	105
4.55.	Decision Curve Analysis (final fit, kalibre edilmiş)	105
4.56.	ROC karşılaştırma – RF vs XGB vs CatBoost (OOF)	108

ÇİZELGELER DİZİNİ

<u>Çizelge</u>	<u>Sayfa</u>
2.1. Modellerin algoritma farklılıkları	21
4.1. Genel veri seti analizi	29
4.2. AHİ sınıf dağılımı	30
4.3. AHİ toplam özet	30
4.4. Tanımlayıcı istatistikler	31
4.5. Grup farkı analizi (AHİ_SINIF=0 vs 1)	32
4.6. AHİ sınıflaması ile spearman kolerasyonu	48
4.7. AHİ toplamı ile spearman	49
4.8. Yüksek korelasyon çiftleri	49
4.9. VİF (klasik)	50
4.10. VİF (modelleme)	50-51
4.11. Normalite ve varyans homojenliği	53
4.12. Grup kıyaslaması	54
4.13. Hedeflerde tek değişkenli regresyon analizi	56-57
4.14. Temel makine öğrenimi değerleri – regresyon modelleri	58
4.15. Temel makine öğrenimi değerleri – sınıflama modelleri	58
4.16. RandomForest - Nested + HPO + kalibrasyon	58
4.17. XGBoost - Nested + HPO + kalibrasyon	60
4.18. CatBoost - Nested + HPO + kalibrasyon	62
4.19. Sınıflandırma modelleri toplu özet	64

4.20.	RandomForest Regressor (Nested CV + HPO) - Fold Metrikleri	66
4.21.	XGBoost Regressor (Nested CV + HPO) - Fold Metrikleri	68
4.22.	CatBoost Regressor (Nested CV + HPO) - Fold Metrikleri	69
4.23.	Regresyon modelleri toplu özet	71
4.24.	DeLong (AUC farkı) – sınıflama modelleri kıyaslama testleri	73
4.25.	McNemar @0.5 – sınıflama modelleri kıyaslama testleri	73
4.26.	Brier farkı (Wilcoxon, 5-fold proxy) – sınıflama modelleri kıyaslama testleri	73
4.27.	PR-AUC farkı (bootstrap) – sınıflama modelleri kıyaslama testleri	74
4.28.	Kalibrasyon (bootstrap; intercept & slope) – sınıflama modelleri kıyaslama testleri	74
4.29.	Kalibrasyon farkları (bootstrap) – sınıflama modelleri kıyaslama testleri	75
4.30.	Regresyon modelleri: OOF özet (RMSE, MAE, R^2) + bootstrap %95 GA – regresyon modelleri kıyaslama testleri	77
4.31.	Pairwise Wilcoxon (RMSE; 5-fold eşleştirilmiş) – regresyon modelleri kıyaslama testleri	78
4.32.	Pairwise Wilcoxon (MAE; 5-fold eşleştirilmiş) – regresyon modelleri kıyaslama testleri	78
4.33.	Δ RMSE bootstrap (eşleştirilmiş; GA + p; Holm) – regresyon modelleri kıyaslama testleri	78
4.34.	Δ MAE bootstrap (eşleştirilmiş; GA + p; Holm) – regresyon modelleri kıyaslama testleri	79
4.35.	SHAP önem tablosu – sınıflama – RandomForest	83
4.36.	SHAP önem tablosu – sınıflama – XGBoost	87

4.37.	SHAP önem tablosu – sınıflama – CatBoost	91
4.38.	SHAP önem tablosu – regresyon – RandomForest	95
4.39.	SHAP önem tablosu – regresyon – XGBoost	99
4.40.	SHAP önem tablosu – regresyon – CatBoost	103
4.41.	DCA’nın sayısal özeti	106
4.42.	Klinik senaryo	106
4.43.	Youden’s J	107
5.1	Literatür kıyaslaması	117

1. GİRİŞ VE AMAÇ

1.1. Obstrüktif Uyku Apnesi (OSA): Tanımı, Epidemiyolojisi ve Toplumsal Yükü

Obstrüktif Uyku Apnesi (OSA), uykuda meydana gelen tekrarlayan solunumda durma (apne) veya solunumda yüzelleşmeye bağlı ortaya çıkan, sistemik etkileri bulunan bir uyku bozukluğu durumudur.

Patofizyolojisini temelde; havayolunun mekanik olarak ortaya çıkan tıkanıklılıklarıdır. Bu hava yolundaki tıkanıklılıklara bağlı olarak, kandaki oksijen oranının azalması (hipoksemi), sempatik sinir sisteminde aktivite ve uyku uyanıklıklar (arousal gibi durumlar) ortaya çıkar. Bundan dolayı OSA sadece uyku kalitesini düşüren bir durum değil, sistemik sağlık durumunda etkilemektedir (1).

OSA'nın prevalansı her geçen yıl artmaktadır ve bu durum toplum sağlığını önemli ölçüde etkilemektedir. 2019 yılında yayımlanan bir çalışmada, dünya genelinde yaklaşık olarak 936 milyon bireyin OSA riskinde olduğu ortaya konmuştur. Bu hastaların yaklaşık olarak 45 milyon kadarının orta şiddetli OSA grubunda olduğu tahmin edilmektedir (2). Türkiye gibi gelişmekte olan ülkelerde ise OSA'nın tanı oranları oldukça düşük kalmaktadır, bu yüzden tanı alamamış veya tanı alsa bile tedavi edilememiş hastalar sağlık sistemi üzerindeki yükü dolaylı olarak arttırmaktadır (3).

Cinsiyete göre değerlendirildiğinde, OSA erkek bireylerde daha sık görülmekle birlikte özellikle 40 yaş üzeri nde prevalansı belirgin şekilde artmaktadır. Obezite OSA için en önemli risk faktörlerinden birisidir ve Vücut Kitle İndeksi'ndeki (VKİ) her bir birimlik artışın, bu riski %14 oranında arttırdığı bildirilmiştir (4). Gebelik, menopoz, hava yolunda daralmaya yol açabilecek anomaliler ve genetik yatkınlıklar da OSA'yı arttırabilecek risk faktörlerinden bazılarıdır.

OSA'nın tanımlanmaması veya geç tanınması, hem birey hemde toplum düzeyindeki sağlık sorunları yükünü arttırmaktadır. OSA, inme, diyabet, hipertansiyon, kardiyovasküler hastalıklar ve metabolik sendromlar ile ilişkilidir (5). Gündüz uykululuğuna bağlı olarak gelişen iş ve trafik kazaları, bireylerin yaşam kalitesindeki düşüş, toplumdaki etkiyi daha da derinleştirmektedir.

2024 yılında Kore sağlık sisteminde yapılan çalışmaya göre; hipertansiyon, depresyon, diyabet ve dislipidemi prevalanslarının OSA hastalarında anlamlı şekilde arttığı gösterilmiştir. Bu çalışma geniş popülasyon verileri sayesinde literatüre önemli katkılar sunmuştur (6).

Sağlık sistemi maliyeti açısından değerlendirildiğinde ise, tanı almamış hasta bireylerin neden olduğu ekonomik yük, tanı ve tedavi süreçlerinden çok daha fazladır. 2023 yılında yapılan bir sağlık ekonomisi incelemesine göre, tanı almamış bir OSA hastasının, hasta başına maliyeti 6.366 dolar olarak hesaplanmıştır (7).

1.2. Apne-Hipopne İndeksi (AHI) ve Tanıdaki Sınırlılıkları

Apne Hipopne İndeksi (AHI), OSA tanısında kullanılan nicel bir ölçüt olup, gece boyu elde edilen toplam sayının, toplam uyku sayısına bölünerek, apne hipopne olaylarının saat başına yansımalarının ifadesidir. AHI değerleri Amerikan Uyku Tıbbı Akademisi (AASM) tarafından belirlenen eşik değerlere göre OSA'nın şiddetini belirlemede kullanılır. OSA'nın şiddeti; AHI 5 ile 14 arasında ise; hafif, 15 ile 29 arasında ise orta, 30 ve üzerinde ise ağır olarak belirlenir (8). Bu ölçütün sınırlılıklarının olduğu ise bilimsel çerçevede, giderek daha sık tartışılmaktadır.

AHI, yaşanan solunum durması veya solunum yüzeyelleşmesi olaylarının sadece sıklığını sayar, bunun haricinde olayların süresi, satürasyonlara olan etkisi, kalp hızındaki değişiklikler gibi fizyolojik sonuçları dikkate almaz. Aynı AHI değerine sahip hastaların klinik profillerinde farklılıklar olabilir. Kısa süreli sık apneli bir hasta ile uzun süreli ama daha seyrek apne görülen hastanın AHI değerleri aynı olabilir lakin fizyolojik stres ve hastalık yükü birbirinden farklıdır (9).

Aynı zamanda AHI'nin sınıflandırma gücü, laboratuvarların kullandığı farklı hipopne tanım kriterlerine göre değişebilmektedir. AASM'nin 2007 yılı ve 2012 yıllarında yaptığı hipopne tanımındaki revizyonlar, hipopne olay sayılarını ve dolayısı ile AHI sayılarını etkilemiştir (10). Bu durum aynı zamanda AHI üzerinde yapılan çalışmalarda da metodolojik sorunlara neden olmuştur.

Bazı çalışmalar, AHI değerinin; kadınlarda, yaşlılarda, obezite dışı bireylerde yani OSA için “atipik” profile sahip bireylerde yetersiz tanı koyabildiğini ortaya koymuştur (11). Kadınlarda daha çok rem uykusunda, yaşlılarda daha az

desatürasyona neden olan OSA formları, geleneksel AHİ ölçümlerinde gözden kaçabilmektedir. Bu durum, “sessiz yük” sorununu beraberinde getirmektedir.

AHİ ölçümü 1 gecelik polisomnografi (PSG) testi ile ölçülmektedir. Ancak OSA gecedan geceye değişebilen bir durumdur, özellikle hafif ve orta şiddetteki OSA hastalarındaki gece geçirilen uykuya bağlı varyasyon, AHİ’nin güvenilirliğini azaltabilir. Bundan dolayı, birden fazla gece ölçümü veya ek değişkenlerinde değerlendirilmesi gerekliliği savunulmaktadır (12).

Alternatif değişkenlerin kullanımı, AHİ sınırlılıklarını aşmak için önerilmektedir, apne ve hipopnelerin süresi, oksijen desatürasyon indeksi (ODİ), arousal sayısı gibi metrikler bu kapsamda değerlendirilmektedir. Tanı koymada AHİ haricinde fizyolojik değişkenlerinde bütüncül ele alınması, tanıda bireyselleşmeye olanak tanıyabilir. AHİ sonuç odaklı bir metriktir; fizyolojik değişkenleride içine alan modeller OSA’nın altında yatan mekanizmaları çok daha iyi ortaya koyabilir (13).

AHİ uzun yıllardır OSA tanısında temel araç olarak kullanılsa da, OSA’nın sahip olduğu çok boyutlu yapıyı tek başına yansıtmamaktadır.

1.3. Polisomnografi (PSG) Sürecinin Zorlukları

OSA’nın tanısında altın standart yöntem olarak kabul edilen PSG, gece boyunca yapılan, çok sayıda fizyolojik metriğin kaydedildiği, çok kanallı bir uyku testidir. Elektroensefalografi (EEG), Elektrokülografi (EOG), Elektromiyografi (EMG), Elektrokardiyografi (EKG), abdominal ve toraks hareketleri ölçümü, pozisyon tayini yapılması, horlama, oksijen satürasyonu, bacak hareketi, kalp hızı gibi sinyaller eş zamanlı izlenir (14).

PSG testinin yapılması belirli operasyonel ve ekonomik zorluklar içermektedir, testin yapılabilmesi için özel eğitim almış uyku teknisyenleri, ileri teknolojiye sahip ekipmanlar, gecelik hasta takibinin yapılabileceği laboratuvar ekipmanı ve hasta izleme alt yapısı gereklidir. 1 hasta için test yaklaşık olarak 8 saat sürmektedir ve hasta bu süreçte gözetim altında tutulmaktadır. Bu durum uyku kalitesini etkileyen stres faktörlerini ortaya çıkartmakla birlikte, sağlık kuruluşları açısından zaman kaybına neden olmakta ve personel maliyetini arttırmaktadır (15).

Ekonomik analizler, Amerika’da test başına ortalama maliyetin 1000 ile 3000 dolar arasında olduğunu göstermektedir. Avrupa ve Türkiye gibi gelişmekte olan ülkelerde test maliyetleri nispeten daha düşük olsa da yüksek hasta yükü, personel ve uzman açığı, laboratuvar yetersizliği gibi nedenlerden dolayı tanı süreci aylarca sürmektedir. Tanıda yaşanan bu gecikmeler klinik etkilerin artmasına ve tedavi erişiminde aksamalara neden olmaktadır (16).

PSG’nin tek bir gece yapılması tanısız duyarlılığı düşürebilir; OSA’da geceden geceye farklılıklar gösterebilmektedir. Bu durum özellikle sınırda AHİ değerlerine sahip olan bireylerde yanlış negatif veya pozitif sonuçların ortaya çıkmasına neden olabilir (17).

PSG uygulamasındaki bu zorluklar bazı alternatif yöntemlerin gelişmesi yönünde çalışmaları hızlandırmaktadır; evde uyku testi (Home Sleep Apnea Test - HSAT) PSG’ye göre çok daha kolay erişilebilir ve maliyet açısından daha düşük olmasına rağmen bazı önemli parametrelerde düşük duyarlılığa sahiptir (EEG takibi, uyku döngülerinin takibi gibi) ve bundan dolayı tanı duyarlılıkları azalmaktadır. Çeşitli çalışmalarda HSAT’ın orta şiddette OSA tanısında duyarlılığı %70-85 arasında değişmektedir; hafif olgularda ise oran belirgin şekilde azalmaktadır (18).

Ayrıca PSG testlerinin analiz süreci zaman alan aşamadır. Skorlama işlemleri manuel veya yarı otomatik sistemlerle yapılmaktadır. Son yıllarda bu aşamada yapay zeka destekli otomatik skorlamalar yürütülmekte ve analiz süresini azaltırken doğruluk oranlarının artırılması hedeflenmektedir (19).

PSG kapsamlı veri sağlayan bir test olsa da klinikte pratik edilmesi çeşitli kısıtlılıklara sahiptir. Bu durumda özellikle sağlık sistemlerinde kaynak kısıtlılığı yaşayan ülkelerde OSA hastalarının tanı sürecinde ciddi dar boğaz oluşumuna neden olmaktadır.

1.4. AHİ Hesabına Katılmayan Fizyolojik Değişkenlerle Tahmin Yaklaşımı

OSA tanısında kullanılan klasik AHİ temel metrik olarak kullanılmaktadır, ancak bu yaklaşım bireyler arası klinik heterojeniteyi yansıtmakta yetersiz kalmaktadır. Literatürde son yıllarda yapılan çalışmalar, AHİ hesaplamasına doğrudan katılmayan fizyolojik değişkenlerin, kalp hızı değişimleri, pozisyon,

desatürasyon olayları, uyku evreleri vb., OSA patofizyolojisi hakkında değerli bilgiler sunduğunu göstermiştir (20).

Bu fizyolojik değişkenlerin AHİ'den bağımsız olarak OSA şiddetini ve komplikasyon riskini tahmin etme fikri, klinik ve analitik anlamda yeni yaklaşımların önünü açmıştır. Ortalama kalp hızı, arousal sıklığı, Rapid Eye Movement (REM) latansı gibi nörofizyolojik ve kardiyorespiratuvar değişkenlerin kullanıldığı çalışmalar, bu değişkenlerin AHİ'den bağımsız olarak kardiyovasküler risk ile ilişkilendirildiğini göstermiştir (21).

2021 yılında yapılan sistematik derlemede AHİ dışında kalan 30 değişkeni analiz etmiş ve bazı değişkenlerin hastalık şiddetini (örn. REM evresi süresi, SO₂ desatürasyon süresi) AHİ'den daha iyi yorumlayabildiği gösterilmiştir. Bu kavram geleneksel tanı sistemlerinin ötesinde “fenotiplendirme” kavramını güçlendirmiştir (22).

Bacak hareketi, vücut pozisyonu, kalp hızı değişkenliği OSA'nın heterojen yapısı ve OSA'nın olası komplikasyonlarını tahmin etmede anlamlı bulunmuştur. Özellikle desatürasyon süre ve derinlikleri kardiyorespiratuvar risk ile doğrudan bağlantılıdır (23).

Makine Öğrenmesi (ML) modellerinin bu tür fizyolojik değişkenlerle yapılan regresyon ve sınıflama çalışmalarında AHİ tahmininin yüksek doğruluk oranlarına ulaşmış olduğu bildirilmiştir; 2022 yılında Extreme Gradient Boosting (XGBoost) metodu ile geliştirilen regresyon modeli AHİ'yi 0,86 oranında doğrulukla tahmin edebilmiştir (24).

Ayrıca aynı AHİ değerine sahip hastaların farklı klinik semptomlara sahip olmasından dolayı bireysel verilere göre yapılandırılmış modeller, hastaların fenotipik farklılıklarını belirlemede güçlü araçlar olarak karşımıza çıkmaktadır (25).

AHİ dışı fizyolojik değişkenlerin analitik modellere dahil edilmesi hem tanısal hassasiyeti arttırmakta hem de OSA'nın sahip olduğu çok boyutlu doğaya daha uygun yaklaşımlar sergilemektedir.

1.5. Bu Çalışmanın Hedefi ve Özgün Katkısı

Bu tez çalışmasında PSG hesaplamasına doğrudan katılmayan fakat klinik önemi bulunan 16 değişken üzerinden AHİ'yi hem regresyon (süreklilik) hem de klasifikasyon (sınıflama) düzeyinde tahmin edecek ML modellerinin geliştirilmesi amaçlanmaktadır. Amaç bir tanı koyma aracı oluşturmak değil, PSG testi sırasında elde edilen veriler doğrultusunda erken evrede AHİ tahmini yaparak maliyet etkin bir karar destek sistemi tasarlamaktır. Sınıflama modelleriyle ağır OSA hastalarını erken yakalamak amaçlanmıştır.

Çalışmamız AHİ'nin matematiksel hesapına doğrudan katılmayan ama klinik olarak ilişkili olan değişkenler üzerinden bir tahmin ve sınıflama modeli kurulmasını amaçlar.

Çalışmada regresyon ve klasifikasyon modelleri bir arada kullanılarak amaca yönelik esneklik sağlanmıştır.

RandomForest (RF), XGBoost ve CatBoost gibi üç güçlü ve birbirinden farklı algoritma ile çalışan modeller oluşturulmuş ve istatistiksel olarak kıyaslanmıştır.

2023 yılında yapılan çalışmada “uyku dokusu” üzerinden OSA karakterizasyonuna dair alternatif parametrelerin yüksek doğrulukta sınıflama performansı sunduğu gösterilmiştir (26). 2022 yılında yapılan bir başka çalışmada ise REM/Non-Rapid Eye Movement (NREM) evreleri ve oksijen desatürasyonu gibi fizyolojik ölçütlerin şiddetli OSA saptamasında oldukça etkili olduğu görülmüştür (27).

Ayrıca yapılan sistematik bir derlemede, son yıllarda geliştirilen ML modellerinin (gradyan artırılmalı modellerde özellikle) AHİ tahmini konusunda, geleneksel istatistiksel modellerden daha yüksek doğrulukta olduğu ortaya konulmuştur (28, 29).

1.6. AHİ Hesaplama Yöntemi, Skorlama Kuralları ve Fiziksel Tanımlar

OSA tanımında temel indeks olarak kullanılan AHİ, bireyin gece boyunca yaşadığı olayların, toplam uyku süresine bölünmesiyle elde edilen, 1 saatlik uyku süresinde geçirilen ortalama apne ve hipopne olaylarının sayısını ifade eder.

$AHI = \text{Toplam Uyku Süresi (saat başına)} \times \text{Toplam Apne} + \text{Toplam Hipopne}$
Olayı şeklinde hesaplanır.

Apne; hava akımının en az 10 saniye süreyle ve %90 veya üzerinde azalması olarak tanımlanır.

Hipopne; hava akımında en az %30 azalma olması ve beraberinde $\geq 3\%$ desatürasyon ya da hipopne sonunda meydana gelen kortikal arousal olması olarak tanımlanır (30, 31).

Bu tanımlar, AASM tarafından 2007 yılında belirlenmiş ve 2012 yılında revize edilmiştir. Bu güncelleme ile birlikte gelen $\geq 4\%$ desatürasyon şartının $\geq 3\%$ desatürasyona getirilmesi veya desatürasyon olayının hava akımında ki %30'luk azalma sonrasında kortikal arousal görülebilmesi, AHI'nin tanısallık hassasiyetini etkilemiştir (32).

AHI ölçümü PSG testinden elde edilen sinyallerin analizi ile yapılır. Manuel analiz, her bir olayın uyku teknisyeni tarafından skorlanması ile, yarı otomatik sistemler; otomatik sistemin işaretlendiği olayların sonradan teknisyen tarafından skorlanması ile, tam otomatik sistemler ise yapay zeka ve/veya yazılım destekli sistemlerin tek başına skorlama yapması ile yapılır.

Manuel skorlamanın altın standart olarak kabul edilmesine rağmen, zaman ve uzmanlık gereksinimi nedeniyle, yarı otomatik ve otomatik sistemler ön plana çıkmaktadır; ancak tam otomatik sistemlerin varyasyonları ve doğruluk düzeyleri halen tartışmalıdır (33).

AHI hesaplanması sadece skorlanmış apne ve hipopne olaylarının sayısını dikkate alır. Uyku pozisyonları, REM/NREM süresi gibi uyku döngüleri, arousal sıklığı, kalp hızı varyasyonları, oksijen doygunluğu profilleri gibi fizyolojik değişkenler AHI hesaplamasına katılmaz. Bu nedenle AHI, OSA'nın fizyopatolojik yükünü tam olarak yansıtamaz (34, 35).

1.7. AHI'nin Klinik ve Tanısal Sınırlılıkları

Giriş bölümünde (bkz. 1.2) belirtildiği üzere; AHI her ne kadar hastalığın tanısında kullanılsa da hastalığın fizyopatolojik karmaşıklığını yansıtmakta yeterli değildir.

AHI tek başına hastaların gündüz semptomlarını, kardiyorespiratuar komorbidite riskini ve yaşam kalitesi üzerindeki etkileri yansıtmakta yetersizdir ve sahip bireylerdeki bireysel farklılıkları göstermemektedir. Yapılan çalışmalar AHI'nın OSA hastalığında sınırlı açıklayıcılığa sahip olduğunu göstermektedir (36, 37).

Bu yüzden günümüzde ek fizyolojik değişikliklerin de değerlendirme sistemlerine katıldığı daha bütüncül yaklaşımlar önerilmektedir. Bu yaklaşım OSA'nın bireyselleşmesinde ve tanı tedavi sürecinde önemli rol oynayabilir (38, 39).



2. GENEL BİLGİLER

2.1. Değişkenler ve Özellikleri

Son yıllarda AHİ hesabına katılmayan fizyolojik değişkenlerin klinik önemlerinin anlaşılmasına başlanması ile birlikte alternatif tanı ve skorlama yöntemleri geliştirme çabaları artmıştır.

2.1.1. Solunumsal Değişkenler (SpO₂, Desatürasyon)

Oksijen desatürasyonu, OSA'nın fizyolojik etkilerini gösteren temel parametrelerden birisidir. Apne atakları sırasında meydana gelen oksijen satürasyonlarındaki düşüş (hipoksik yük) geleneksel AHİ hesaplamalarına sadece dolaylı olarak yer bulur.

SpO₂'ye dayalı ölçümler arasında, en düşük oksijen doygunluğu, ortalama desatürasyon yüzdesi, $\geq 3\%$ desatürasyon olay sayısı, hipoksik yük indeksi gibi bileşenler yer alır. Yapılan çalışmalar bu değişkenlerin kalp hastalıkları, hipertansiyon ve mortalite ile güçlü ilişkiler gösterdiğini ortaya koymuştur (40, 41).

Bazı algoritmik modeller, desatürasyon eğrisi altında kalan alan yeni bir metrik olarak önerilmiş ve kalp yetmezliği ve mortalite ile ilişkilendirilmiştir. Bu durum desatürasyon profilinin apne - hipopne sayısından daha anlamlı bir klinik gösterge olabileceğini düşündürmektedir (42).

Ayrıca, SpO₂ ortalaması, sadece minimum değerden ziyade, tüm gece boyu maruz kalınan oksijen seviyesini yansıtmaması nedeniyle daha bütünsel bir değerlendirme sağlar. Yaşlılarda olan gece boyu maruz kalınan kronik hipoksi, bilişsel bozulma riskini artırabilmektedir (43).

2.1.2. Kardiyovasküler Parametreler

OSA, sadece solunumsal bir hastalık değil, aynı zamanda kardiyovasküler sistem üzerinde doğrudan etkisi olan sistemik bir bozukluktur. Hipoksik ataklar, olduğu uyku bölünmelerine, sistemik sistemdeki aktivite artışına ve vagal tonus

azalmasına neden olur. Bu durum özellikle gece boyunca ölçülen kalp hızı ortalamaları ve kalp hızı değişkenliğinin (HRV) belirgin değişikliklere yol açar.

AHI'de bu değişkenler doğrudan hesaplama katılmamakta ve göz ardı edilmektedir. Yapılan çalışmalar değişken HRV göstergelerinin kardiyovasküler hastalık riski açısından güçlü biyobelirteçler olabileceğini bildirmiştir (44).

2.1.3. Uyku Evresi ve Pozisyon Verileri

Uyku evrelerinin dağılımı (REM/NREM) ve vücut pozisyonu uyku süreci ile yakından ilişkilidir. AHI'nin sadece sayısal olay sayısına odaklanması, evre ve pozisyona dair bilgi veremez.

Birçok çalışma OSA'nın REM evresinde ve sırt üstü pozisyonda meydana geldiğini ortaya koymuştur (45). REM yoğunluklu OSA (REM related OSA) özel bir fenotip olarak karşımıza çıkarak; düşük AHI'ye sahip bireyler klinik olarak anlamlı kardiyovasküler risk taşıyabilir. Sırtüstü pozisyonda apne yaşayan bireyler Pozisyonel Obstrüktif Uyku Apnesi (POSA) alt grubuna dahil edilir (46).

Pozisyonel etkilerin tespiti hastaya yönelik tedavinin belirlenmesinde önem arz etmekle beraber, AHI'nin bu durumu yeterince yansıtamaması hastaların fenotip bazında yanlış değerlendirilmesine yol açabilmektedir (47).

2.1.4. Diğer Fizyolojik Değişkenler

Uyku sırasında ortaya çıkan mikro uyanıklıklar (Arousallar), uyku bölünmelerine, günlük uyanıklık semptomlarının ortaya çıkmasına ve kardiyovasküler yük artışına neden olmaktadır (48).

Cinsiyet fenotipik yansımada önemli bir faktör olarak karşımıza çıkar. Kadınlarda OSA genellikle daha hafif, REM ilişkili ve atipik semptomlarla seyredebilmektedir. AHI cinsiyete yönelik ayrımı net yansıtmamaktadır (49).

Yaş ilerledikçe uyku mimarisi değişir, REM süresinde kısalma, arousal ve hipopnelerde artma meydana gelmektedir (50).

Bacak hareketi; uyku sırasında ortaya çıkan bacak hareketleri, uyku kalitesini bozmakta, arousal sıklığını arttırmak ve yaşlılarda uyku kalitesini düşürmektedir (51).

2.2. Model Kurulum Değişkenleri

Bu bölümde; tezimizin ilerleyen aşamalarında çok daha detaylı şekilde anlatacağımız tahmin ve sınıflama modellerinin kurulum sürecinde seçilen 16 değişken anlatılacaktır.

2.2.1. Vücut Kitle İndeksi (VKİ)

VKİ, OSA fizyopatolojisinde en çok vurgulanan risk faktörlerinden birisidir, obeziteye bağlı periferik çevresindeki yağlanma, üst solunum yolu daralması, apne ve hipopne olaylarının artmasına sebebiyet vermektedir.

Op de Beeck ve ark.'nın 2025 yılında yayımladığı bir çalışmada, VKİ değerlerinin, AHI değerlerinden bağımsız olarak üst solunum yollarındaki kollaps ile ilişkili olabildiği gösterilmiştir (52). Bir başka çalışma vücut yağ yüzdesinin ve VKİ'nin AHI'den bağımsız OSA şiddetini öngörebildiği gösterilmiştir. Bu çalışmada özellikle erkek bireyler yüksek VKİ yüksekliğinin apne oluşumunda tetikleyici etkisine dikkat çekilmiştir (53).

Ek olarak 2023 yılında yapılan sistemik derlemede, obezitenin OSA için bir bağımsız belirteç olduğu, AHI hesaplamalarına doğrudan katılmadığı, tanısal yanılgılara sebep olunabileceği vurgulanmıştır (54). Bu durum VKİ gibi klinik bir belirtecin, OSA öngörüsünde kullanılabileceği ve ML modellerine entegre edildiğinde bu öngörünün daha da güçleneceği belirtilmiştir (55).

2.2.2. Uyku Latansı (Sleep Latency)

Uyku latansı; kişinin yatağa yattığı an ile uykuya daldığı an arasındaki geçen süredir. Uykuya dalma süresinin bozulması, uyku yapısının ve döngülerinin bozulmasına neden olabilir. Dolaylı olarak solunumsal düzensizlikleri şiddetlendirebilmektedir. Wang ve ark.'nın 2025 yılında yayımladığı bir çalışma; Uyku latansındaki artışın sadece OSA ile değil aynı zamanda dislipidemi gibi eşlikçi metabolik bozukluklarla da ilişkili olduğu gösterilmiştir. Çalışmada AHI düzeyinden bağımsız dislipidemi riskinin uzamış uyku latansı ile ilişkili olduğu ortaya konulmuştur (56).

Uyku latansı aynı zamanda uyku duyarlılığı, gündüz uykululuğu ve psikolojik durumlarla da ilişkilidir. Bundan dolayı ML modellerine dahil edilmesi OSA'ya bütüncül yaklaşıma katkı sağlar.

2.2.3. REM Latansı (REM Latency)

REM latansı, bireyin uykuya daldığı an ile ilk REM uykusuna geçişi arasındaki süreyi ifade etmektedir. Doğrudan AHİ hesabına katılmamakla birlikte; özellikle uyku mimarisi ve REM apne tipleri ile ilişkilidir. Normal REM latansı 90–120 dk'dır. 2025 yılında Xu ve ark.'nın yayımladığı çalışmada REM ilişkili OSA hastalarında REM latansının anlamlı değiştiği, apnenin zamanlamasına etki ettiği sonuçlarına ulaşmıştır. REM latansının toplam uyku süresinden bağımsız olarak prognostik faktör olabileceğini belirtmiştir. Sadece uyku evresi değişkeni değil, alt fenotip belirlemede kullanılabilecek önemli bir değişkendir. Bu yüzden regresyon modellerine dahil edilmiştir (57).

2.2.4. Uyku Periyodundaki Uyanıklık Süresi (WASO – Wake After Sleep Onset)

Uyku Periyodundaki Uyku Süresi (WASO); bireyin uykuya başladığı andan itibaren uykusunun bölündüğü ve uyanık kaldığı toplam süreyi ifade etmektedir. WASO, uyku bölünmelerini doğrudan yansıtan bir uyku katisenin belirtecidir.

2023 yılında Fernandes ve ark.'ı tarafından yapılan çalışmada WASO süresinin AHİ ile zayıf bir korelasyon gösterdiği, gündüz uykululuğu ve uyku kalitesi ile güçlü bir ilişkisi olduğu gösterilmiştir. Çalışma, düşük AHİ'li bireylerde uzamış WASO varlığının anlamlı klinik etkilerinin olduğunu ortaya koymuştur (58).

Zhang ve ark.'nın 2022 yılında yaptığı bir sistematik derleme ise WASO'nun yüksek olduğu hasta grubunda bilişsel fonksiyonlarda azalma ve artmış depresyon semptomlarının görüldüğünü göstermiştir. Bu semptomları göstermede AHİ'den daha anlamlı bulunmuştur (59). Klinik olarak semptomlarla ilişkisi gösterilen bu değişken modellere dahil edilmiştir.

2.2.5. REM Evresi Süresi (REM Sleep Duration)

REM evresi; rüya görme, beyin aktivite artışı ve eş zamanlı kas tonusunda belirgin azalma ile karakterize edilen uyku evresidir. OSA'da bu süre genellikle kısaldır; REM dominant apne varlığı klinik olarak daha ağır seyredir.

2024 yılında Huang ve ark.'nın yaptığı bir çalışmada REM süresi ile AHİ arasında istatistiksel olarak anlamlı olmayan ancak REM dominant OSA'larda bu sürenin klinik semptomlarla güçlü ilişki içinde olduğu gösterilmiştir. Kısa REM süresine sahip hastalarda belirgin gündüz uykusuzluğu ve kognitif bozukluklar rapor edildiği gösterilmiştir (60). 2023 yılında Jones ve ekibinin yürüttüğü çok merkezli bir prospektif çalışmada; toplam REM süresi ile kardiyovasküler riskin AHİ'den bağımsız olduğu ve REM süresi kısalmış kişilerde hipertansiyon ve kalp ritim bozuklukları prevalansının anlamlı şekilde arttığı gösterilmiştir (61). Klinik etkileri olan bu belirteç ML modellerine dahil edilmiştir.

2.2.6. Toplam Arousal Sayısı (Total Arousal Count)

Arousal; beyin aktivitesindeki ani değişikliklerle seyreden, uykunun karakterini bozan mikro-uyanıklıklardır. OSA hastalarında genellikle solunumsal olayların arkasından gelişir. Ancak AHİ skorlaması yapılırken tüm arousal sayısı bu skorlamaya dâhil edilmez. Bu sebeple AHİ hesabına doğrudan katılmaz.

2024 yılında Thorisdottir ve ark.'nın yayımladığı çalışmada; arousal sayısının bilişsel performansla ilgili olan ilişkisi AHİ değerinden bağımsız olarak değerlendirilmiştir. Desatürasyon şiddeti kadar arousal sayısının da kognitif performansta anlamlı etkilere sahip olduğu ortaya konmuştur (62).

2.2.7. Sırtüstü Uyku Pozisyonu

Sırtüstü pozisyon, yerçekiminin etkisiyle üst hava yolunun daha kolay daralması nedeniyle solunum olaylarının sıklığını arttırmaktadır. Bu özel durum, POSA olarak tanımlanır. Pozisyon, bağlı olarak hastanın AHİ değerlerini etkileyebilmektedir.

Lin ve ark.'nın 2025 yılında yürüttükleri büyük örneklemli retrospektif analizde, sırt üstü pozisyonda geçirilen sürenin artması AHİ değerinde bağımsız ve

anlamalı bir deęişikliğe neden olmuş; vücut pozisyonunun etkilerinin bireyden bireye deęişebileceęi ancak en çok kişiyi etkileyen pozisyonun sırtüstü pozisyon olduęu gösterilmiştir (63).

Benzer şekilde Ciavarella ve ark.'nın 2025 tarihli çalışmasında, Mandibular İlerletme Apreyi (MAD) kullanımı sonrasında AHİ skorlarında belirgin düşüş olduęu görülmüş ve pozisyon duyarlı OSA hastalarında pozisyon deęişikliğinin tedaviye katkı sunabileceęi; pozisyon duyarlı bireylerde tanı ve yönetim algoritmalarına mutlaka dahil edilmesi gerektiğini belirtmiştir. AHİ hesabına katılmasa da klinik önemi nedeniyle bu metrik modele dahil edilmiştir (64).

2.2.8. Uyanıklık Satürasyon Ortalaması (Awake SpO₂ Mean)

Uyku öncesi uyanık dönemde ölçülen ortalama oksijen satürasyonu, O₂ taşıma kapasitesini yansıtan önemli bir faktördür. Hastalığın genel hipoksik etkisini deęerlendirmek için önemlidir.

Yazdi ve ark. 2019 yılında ilk kez uyanıklık satürasyon deęerleri ile OSA şiddeti arasında ters orantı olduęunu göstermiştir. Bu deęer %95 altında ise ciddi obstrüktif olayların ön belirteci olabileceęi belirtilmiştir (65).

2022 yılında Wu ve ark. tarafından yapılan çok merkezli bir gözlemsel çalışmada; uyanıklık satürasyon deęerleri 94 ve altında olan bireylerde kardiyovasküler olay insidansının belirgin şekilde daha yüksek olduęu gösterilmiştir. Klinik etkileri yansıtmamasından dolayı modele dahil edilmiştir (66).

2.2.9. En Düşük Oksijen Satürasyonu Yüzdesi (Lowest Oxygen Saturation Percentage)

En düşük oksijen satürasyonu, uykudaki minimum satürasyon deęerini yansıtır. Minimum satürasyon deęerleri kardiyovasküler risklerle AHİ'den bağımsız ilişkide olup prognostik gösterge olabileceęi vurgulanmaktadır.

Chen ve ark. 2025 yılında yapmış oldukları POSA hastalarıyla ilgili çalışmada minimum oksijen satürasyonlarının düşük olduęu durumların AHİ'den bağımsız kötü kardiyovasküler sonuçlarla ilişkili olabileceęi gösterilmiştir (67). Zhang ve ark. 2024 yılında yaptıkları çalışmada ise REM sırasındaki minimum

desatürasyon değerlerinin gece hipertansiyonu ve kardiyovasküler risk ile ilişkili olduğunu göstermiştir (68).

2.2.10. Ortalama Oksijen Desatürasyonunun Yüzdesi (Average Oxygen Desaturation Percentage)

Ortalama oksijen desatürasyonu, uyku sırasındaki oksijen düzeyindeki düşme oranlarının ortalamasını temsil etmektedir. Hipoksemik yükü yansıtır. AHI değerleri normal aralıkta olan hastalarda da belirgin desatürasyonlar yaşanabildiği gösterilmiştir. Düşük desatürasyonun bilişsel fonksiyonlar üzerinde ve kardiyovasküler risk ile belirgin ilişki içinde olabileceği vurgulanmıştır.

Hrubos-Strøm ve ark.'nın 2024 yılında yaptığı çalışmada, AHI ile korelasyon göstermeyen hastalarda ortalama oksijen desatürasyonunun bilişsel performans ile anlamlı bir ilişkisi olduğu gösterilmiştir (69).

Azaklı ve ark.'nın 2024 yılında yaptığı çalışmada, ortalama oksijen desatürasyonu ve desatürasyon indeksinin AHI'nin ötesinde mortalite riski ile anlamlı bir ilişki gösterdiğini vurgulamıştır. Bu çalışmada AHI istatistiksel önemini kaybederken oksijen parametreleri regresyon modellerinde güçlü tahmin metrikleri olarak karşımıza çıkmıştır (70).

2.2.11. Oksijen Desatürasyon Yüzdesi ≥ 3 Olay Sayısı (Oxygen Desaturation Percentage ≥ 3 Events)

Uyku süresinde SpO₂ seviyesinde, belirli bir eşik (örn. %3) altında; ne sıklıkta kaldığını ifade eden bir değişkendir. Hipopnenin belirlenmesindeki dikkat edilmesi gereken metriklerden bir tanesiyken tüm olay sayıları hesaplamada kullanılmaz. AHI, yalnızca skorlanmış değerlerin sayısının uyku süresine bölünmesi ile elde edilmiş bir değerdir. Turnbull CD, Wang ark.'nın 2023 yılında yaptığı çalışmada, ≥ 3 Olay Sayısı AHI'den bağımsız bir ölçüt olduğunu ve fizyolojik yükü belirttiğini göstermişlerdir (71).

ODI, T90 gibi oksimetrik ölçütler, solunum olaylarının etkilerini AHI'den daha iyi yansıtmaktadır. AHI ile ilişkili olsa da AHI hesabındaki matematiksel tanıma doğrudan dahil değildir (72).

%3 eşik ile belirlenmiş ≥ 3 Olay Sayısı, kardiyovasküler ve metabolik sonuçlar açısından önemli bir göstergedir. Chen ve ark. 2023 yılında yaptıkları çalışmada ≥ 3 Olay Sayısı değerinin AHI'den bağımsız kardiyovasküler mortalite ile anlamlı ilişki içinde olduğunu göstermiştir (73).

Ye ve ark. 2024 yılında, ≥ 3 Olay Sayısı değerinin metabolik sendromu tahmin etmede AHI'den daha etkin olduğunu göstermiştir. 5000'den fazla hastada ≥ 3 Olay Sayısının insülin direnci ve glikoz intoleransı ile güçlü ilişki içinde olduğu ortaya çıkmıştır (74).

2.2.12. REM Evresindeki Ortalama Kalp Hızı (Mean Heart Rate During REM Sleep)

REM uykusu sırasında ortalama kalp hızı otonom sinir sisteminin ne ölçüde aktif olduğunu gösteren önemli fizyolojik değişkenlerden birisidir. OSA hastalarında REM döneminde artmış sempatik aktivasyon daha belirgin olup hastalığın şiddetiyle doğru oranda ilişkilidir. Kalp hızındaki değişimler, uyku evreleri sırasında oluşan fizyolojik adaptasyon mekanizmalarını da işaret eder (75). Kardiyovasküler risk ve uyku mimarisi ile olan ilişkisinden dolayı modellere dahil edilmiştir.

2.2.13. NREM Evresindeki Ortalama Kalp Hızı (Mean Heart Rate During NREM Sleep)

NREM uykusu sırasında ortalama kalp hızı; uyku boyunca ortaya çıkan vagal uyarının ve otonom dengenin bir göstergesidir. OSA hastalarındaki NREM kalp hızı değişiklikleri, otonom sinir sistemindeki dengesizliğe işaret eder.

2024 yılında yapılan çalışma, NREM evresindeki kalp hızındaki değişikliklerin kardiyovasküler strese katkıda bulunduğunu ve fizyolojik yükü yansıtmaya katkı sağlayabileceğini göstermiştir (76).

2.2.14. Uykudaki Toplam Bacak Hareketi Sayısı (Total Leg Movements During Sleep)

Uyku esnasında ortaya çıkan bacak hareketleri uyku kalitesini doğrudan etkileyen fizyolojik bir metriktir. Uykuda görülen bacak hareketleri; Periyodik Bacak Hareketi Bozukluğu (PLMD) veya Huzursuz Bacak Sendromu (RLS) ile ilişkilidir. Bacak hareketleri arousallar ile birlikte görülerek uyku kalitesini ve uyku mimarisini bulabilmektedir.

Ciavarella D ve ark. 2025 yılında yapmış oldukları çalışmada PLMD'ye sahip OSA hastalarında AHİ'nin normal sınırlar içerisinde olmasına rağmen gündüz semptomlarını ve uyku kalitesini etkilediğini ortaya koymuştur (77).

2023 yılında yapılan sistematik derleme; artmış bacak hareketlerinin uyku evresi geçişi ve arousal sayısı ile anlamlı ilişkide olduğunu ortaya koymuştur. AHİ skorlamasına dahil edilmese de klinik tablonun ciddiyetine katkı sunduğu vurgulanmıştır (78).

2.2.15. Yaş (Age)

Yaş; uyku fizyolojisi üzerinde doğrudan etkili olan demografik bir değişkendir. Yaş ilerledikçe uyku süresinde uzama, N3 uyku evresinde kısalma ve uyku bölünmelerinde artış görülür. Yaş, OSA için hem predispozan faktördür hem de hastalık şiddetiyle ilişki içerisindedir.

İleri yaş AHİ'den bağımsız olarak OSA hastalarında kardiyovasküler riski AHİ'den bağımsız olarak arttırmaktadır (79). Yaşın OSA gelişiminde anatomik ve nöromusküler faktörlerle etkili olduğunu belirtmiştir (80).

2.2.16. Cinsiyet (Sex)

OSA'nın gelişimi ve klinik sunumu açısından cinsiyet önemli bir biyolojik faktördür. Erkeklerde OSA prevalansı kadınlara göre 2 kat daha yüksek bulunmuştur. Bu fark; hormonlar, anatomi ve nörofizyolojik farklılıklara dayandırılmıştır (81).

Tek başına AHİ, bu biyolojik farklılıkları ortaya koymakta yetersiz kalmaktadır. Kadınlarda postmenopozal dönemde, OSA daha az belirgin AHİ ile

seyretmesine rağmen semptomatik yük ve komorbiditeler açısından erkeklerle benzer seyretmektedir (82).

Biyolojik farklılıkların modele katkı sağlayacağı düşünülerek modele dahil edilmiştir.

2.3. Makine Öğrenmesine Genel Bakış ve Sağlık Alanındaki Uygulamalar

ML, sistemlerin açık programlama olmadan verileri analiz etmesi, verilerden öğrenmesi ve sonuçlar üretmesini sağlayan yapay zeka alt dalıdır. Büyük hacimli veri kümelerinden anlamlı sonuçlar çıkarabilme yeteneklerinden dolayı; tanımlama, sınıflama, öngörü amacıyla yaygın olarak kullanılmaktadırlar (83).

Denetimli ve denetimsiz öğrenme olarak ikiye ayrılan ML sistemlerinde; denetimli öğrenme; etiketlenmiş verileri kullanarak tahmin etme amacı ile algoritmaların eğitilmesini amaçlar. Regresyon problemleri; sürekli değer tahminine, sınıflama problemleri ise verileri belirli katagorilere ayırmaya yöneliktir. AHİ gibi sürekli değişkenlerin regresyon modelleri ile, OSA'nın şiddeti gibi kategorik seçim problemleri ise sınıflama algoritmaları ile modellenebilmektedir (84). Özellikle “ensemble” yöntemlerinin doğruluk oranını arttırdığı gösterilmiştir (85).

2.3.1. Sağlıkta Makine Öğrenmesi Kullanım Örnekleri (EKG, ICU, SpO₂ vb.)

ML, son yıllarda sağlık alanında tanı, tahmin, sınıflama gibi birçok alanda kullanılmaktadır. Yüksek hacimli ve çok boyutlu verileri analiz etme potansiyelleri, geleneksel istatistiksel hesaplamalara göre öne çıkmaktadır. Bu durum, yeni analizlerde de makine öğrenme modellerini ön plana çıkarmaktadır.

Yoğun bakım ünitelerinde hasta izlem sistemlerinin sürekli analizinde ML algoritmaları, hastaların kötüleşmesini tahmin etmede etkili olmuştur. Ramezani ve ark.'nın 2025 yılında yayımladığı çalışmada, yapay zeka destekli hasta izlem sistemlerinin hemşire yükünü azalttığı ve ventilasyon gereksinimi gelişen hastaların %90'ını 6 saat önceden tahmin edebildiğini göstermiştir (86). Bu çalışmada EKG,

oksimetri ve solunum parametreleri kullanılmış ve derin öğrenme modelleri ile analiz edilmiştir.

Chaudhri ve Raut'un 2024 yılında yayımladığı çalışmalarında, biyosensör verilerine dayalı ML modelleriyle kritik hasta yönetiminde başarılı sonuçlar elde edilmiştir. Pulse oksimetre, glikoz sensörü ve EKG verileri ile eğitilen ML modelleri, mortalite riski yüksek hastaları tanımlayabilmiştir (87).

ML destekli EKG analizi yalnızca aritmilerde değil, Miyokard Enfarktüsü (MI) ve kalp yetmezliği gibi durumlarda da erken teşhis amacıyla kullanılmaya başlanmıştır. Desatürasyon örüntüleri ML algoritmaları sayesinde COVID-19 hastalarında prognoz takibinde etkin rol aldığı gösterilmiştir. Oksimetre verileriyle uyku apnesi ve KOAH gibi solunumsal hastalıkların tanınmasında yaygınlaşmaya başlamıştır (88).

2.3.2. Uyku Tıbbında Makine Öğrenmesi Kullanımı

Son yıllarda yapay zeka (YZ) ve ML, uyku bozukluklarının sınıflandırılmasında geleneksel yöntemlere göre umut verici sonuçlar sunmaktadır. OSA, insomnia, hipersomnia ve REM davranış bozukluğu gibi durumlarda; PSG test sonuçlarının otomasyonunda, AHİ tahmininde ve risk skorlama gibi durumlarda kullanılmaktadır.

Güneş ve ark. 2024 yılında, 500 hastalık veri setinde SpO₂ ve kalp hızı değişkenleri ile AHİ sınıflamasında %93 doğrulukla çalışan bir CatBoost modeli geliştirmişlerdir. Model, bir kısım veri yardımıyla OSA şiddetlerini yüksek duyarlılıkla ayırt edebilmiştir (89).

Banerjee ve ark. 2023 yılında yapmış oldukları geniş çaplı bir retrospektif analizde SpO₂ ve uyku evresi değişkenlerini kullanarak AHİ skor tahminine yönelik bir XGBoost algoritması geliştirmiştir. Bu model, geleneksel AHİ hesaplama sürecinden bağımsız olarak klinik anlam taşıyan bir skor üretmiştir (90).

2019 yılında yapılan bir sistematik derlemede ML algoritmalarının kullanım alanları incelenmiş ve REM–NREM ayrımı, arousal tespiti, horlama ve göğüs hareket analizi algılanmasında yüksek performans gösterdiği vurgulanmıştır (91).

Yine Cheng ve ark. ise 2022 yılında EEG verileriyle çalışarak REM uyku davranış bozukluklarının erken tanısında ML algoritmalarının doğruluğunun

%85'in üzerinde olduğunu göstermiştir. Bu bulgular, nörodejeneratif hastalıkların erken teşhisi için ML algoritmik sistemlerinin önünü açmaktadır (92).

2.3.3. Bu Çalışmada Kullanılacak Modellerin Seçilme Gerekçeleri

Bu çalışmada; RF, XGBoost ve CatBoost algoritmaları hem regresyon hem de sınıflama problemleri için tercih edilmiştir.

Bu algoritmalar; yüksek genel doğruluk oranlarına sahip olmaları, non-linear ve karmaşık veri yapılarına uyum sağlayabilmeleri, aşırı öğrenmeye (overfitting) karşı dayanıklı olmaları, özellik önemliliği (feature importance) analizi imkânı sunmaları ve eksik veri ile dengesiz dağılımlar gibi gürültü içeren verilere karşı toleranslı olmalarıyla öne çıkmaktadır.

RF, birçok karar ağacından oluşan ve her ağacı farklı veri alt kümeleri ile eğiten bir topluluk (ensemble) yöntemidir. Eksik veri, çok boyutluluk ve korelasyon gibi durumlara karşı toleranslıdır. RF'in yüksek F1 skoru ile AHI skor tahmininde başarılı olduğu gösterilmiştir (93).

XGBoost; optimizasyon gücü yüksek ve düşük hata toleransı ile dikkat çeken bir gardiyan artırma algoritmasıdır. Küçük farkların anlamlı olduğu hassas sonuçlara gereksinim duyan durumlarda hassasiyetiyle öne çıkmaktadır. AHI tahminlerinde bu yöntem, yüksek R ve düşük Hataların Karesinin Ortalaması (MSE) skorları ile başarılı sonuçlar göstermiştir (94).

CatBoost yöntemi, kategorik değişkenlerle çalışmada yüksek performans sunan bir algoritmadır. Cinsiyet gibi kategorik değişkenlerin etkili işlenmesinde başarılıdır. OSA gibi heterojen klinik tablolarda CatBoost'un başarısı birçok çalışmada doğrulanmıştır (95).

Ayrıca bu üç algoritma, birbirinden farklı hiperparametre optimizasyonlarına ve iç yapıya sahip olduğu için model sonunda performansları birbirini tamamlayıcı sonuçlar üretmektedir. Böylelikle hem regresyon hem sınıflama modellerinde yüksek güvenilirlik elde edilmeye çalışılmıştır.

2.3.4. Seçilen Modellerin Temel Teknik ve Klinik Farklılıkları

RF; bagging yaklaşımı ile çalışan bu model birçok karar ağacından oluşur ve her ağaç farklı alt veri kümesi ile eğitilir. Bu da aşırı öğrenmenin önüne geçer.

Gürültüye karşı dayanıklıdır, özellik önem sırası verir, açıklanabilirlik sağlar ve non-lineer ilişkileri başarıyla modelleyebilir (96).

XGBoost; gradyan boosting yöntemine dayalı bir modeldir ve zayıf modellerden güçlü bir model oluşturur. “Loss function” ve “regularization” özellikleri ile aşırı öğrenmeyi azaltır. Eksik verilerle başa çıkabilir. Grid search ile değişken optimizasyonuna açıktır. Regresyon tabanlı çıktılarda hassasiyet sağlar ve kritik karar eşiklerinde başarılı ayrım sunar (97).

CatBoost; kategorik değişkenlerin doğrudan işlenmesini sağlayan bir boosting algoritmasıdır. Veri kümesinde kategorik ağırlıklı değişken olduğunda öne çıkar. Dengesiz kümelerde daha etkin çalışır, aşırı öğrenmeye karşı yüksek dirençlidir ve klinik tahminlerde yorumlanabilirlik sağlar (98).

Çizelge 2.1. Modellerin algoritma farklılıkları

Özellik/Model	RF	XGBoost	CatBoost
Model Tipi	Bagging	Gradient Boosting	Ordered Target Boosting
Aşırı Öğrenme	Düşük	Orta	Çok Düşük
Eksik Veri	Orta	İyi	Çok İyi
Kategorik Veri	Ek Kod Gerekir	Ek Kod Gerekir	Doğrudan İşler
Açıklanabilirlik	Yüksek (Feature İmp)	Orta (SHAP)	Orta-Yüksek

2.3.5. Makine Öğrenmesi ile AHİ Tahmini: Literatürdeki Regresyon ve Sınıflama Modelleri

PSG testi, zaman alıcı ve maliyeti yüksek olan, birçok kısıtlılık içeren bir testtir. Fizyolojik değişkenlerden ML tabanlı regresyon ve sınıflama modelleri tanı sürecini hızlandırma, maliyet düşürme ve hastaya özgü karar destek sistemleri geliştirme açısından büyük ilgi görmüştür.

Liang ve ark. tarafından yapılan çalışmada, PSG öncesi fizyolojik değişkenler ile cerrahi sonrası AHİ'nin düşme oranı tahmin edilmiş; RF, XGBoost ve Destek Vektör Makinesi (SVM) gibi yöntemler karşılaştırılmıştır. XGBoost yöntemi, VKİ,

uyku latansı ve oksijen profili ile AHİ regresyon modelinde en iyi sonucu almıştır (99).

Hou ve ark. çalışmalarında CatBoost ve Gradient Boost yöntemleri ile sınıflama modelleri geliştirmiş ve elde ettikleri sonuçların geleneksel sınıflama modellerinden daha üstün performans gösterdiğini göstermiştir (100).

ApneaNET isimli derin öğrenme programı, yalnızca SpO₂ sinyali kullanarak AHİ skor tahmini yapmış ve %91 doğruluk oranına ulaşmıştır. Minimum veri ile maksimum doğruluk sağlamayı hedefleyen bu modellerin, klinik öncesi taramada kullanılabileceği gösterilmiştir (101).

Dong ve ark. 2025 yılında yaptıkları çalışmada fenotip temelli bir algoritma kurmuştur. Maksillofasiyal deformasyonlu çocuklarda şiddet ilişkisi için tahmin modeli geliştirmişlerdir. Doğruluk oranları hem sınıflama hem regresyonda oldukça yüksek çıkmıştır (102).

Suresh ve ark.'nın 2024 yılında yapmış oldukları çalışmada sadece oksijen profilleri üzerinden ahi tahmini için Support Vector Regression (SVR) modeli kurmuşlardır ve sonuçlar ROC eğrisinde yüksek alan göstermiştir (103).

Srivastava ve ark. 2025 yılında PSG üzerinde 50 özellik kullanarak hem regresyon hem sınıflama modeli kurup performanslarını değerlendirmiştir. Özellikle Logistic Regresyon modelinin AHİ'ye göre düşük doğrulukta sınıflama yaptığını ancak ML modellerinde REM/NREM ayrımı ve arousal gibi değişkenleri daha etkili işlediğini göstermiştir (104).

Bu tarz çalışmalar, geleneksel PSG'ye alternatif olarak daha maliyet-etkin, kolay uygulanabilir, otomatik AHİ sistemlerinin geliştirilmesinin mümkün olabileceğini göstermiştir. Sınıflama modelleri, ağır OSA tespitinde oldukça başarılıdır ve erken tanı için önemli bir avantaj sağlar.

2.3.6. Kullanılan ML Modelleri ve Performans Değerlendirme Yöntemleri

ML tabanlı modellerde klinik geçerlilik, yalnızca algoritma başarısına değil, uygun performans metriklerine de bağlıdır.

Regresyon modelleri için Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) ve Determination Coefficient (R²) uygun performans metrikleridir.

MAE, tahminin gerçek değere olan ortalama mutlak farkını ölçer; RMSE, hata dağılımına duyarlı bir ölçüdür ve büyük hatalara ağırlık verir; R^2 ise modeldeki verinin varyansı ne ölçüde açıkladığını gösterir.

Bir çalışma, XGBoost ve LightGBM modellerinin değerlendirilmesinde RMSE ve MAE değerlerinin geleneksel logistik regresyona kıyasla anlamlı düzeyde düşük olduğunu ortaya koymuştur (105).

Sınıflama modellerinde ise Accuracy (doğruluk oranı), Sensitivity (duyarlılık), Specificity (seçicilik), Precision (kesinlik), F1-Score (denge skoru) ve Receiver Operating Characteristic-Area Under Curve (ROC-AUC) metrikleri kullanılmaktadır.

F1 skoru, dengesiz sınıf dağılımlarında önem kazanır. Nadir görülen durumların tespitinde F1 skoru ve ROC-AUC gibi metrikler, Accuracy'den daha açıklayıcı olabilir.

Koley ve ark. 2022 yılında yaptıkları sistematik derlemede, SVM, RF ve XGBoost sınıflandırıcıları arasında anlamlı fark olduğunu göstermiştir (106).

2.3.7. Mevcut Yaklaşımların Literatürdeki Sınırlılıkları

ML her ne kadar son yıllarda önemli gelişmeler göstermiş olsa da, bu yaklaşımların çeşitli kısıtlılıkları literatürde vurgulanmaktadır. Bu sınırlılıklar; veri yapısı, model doğruluğu ve klinik uygulanabilirlik açısından ele alınmalıdır.

Model Genelleme Problemleri: Modeller genellikle tek merkezli ve sınırlı sayıdaki veri ile eğitilir. Bu durum, modelin farklı hasta grupları ve demografik özelliklere sahip bireylerde uygulanmasını sınırlar. Aynı model bir klinikte başarılı iken başka bir klinikte aynı doğruluk düzeyine ulaşamamaktadır (107).

Modeller yüksek doğruluk sunsa da klinik pratiğe nasıl entegre edileceği hâlâ net değildir. Klinik karar sistemlerinin yorumlanabilir olması modele güven açısından önem taşır. Bu nedenle Açıklanabilir Yapay Zeka (Explainable AI – XAI) yaklaşımı son yıllarda öne çıkmıştır (108).

Sınıflandırılmak istenilen değişkenin veri seti içerisinde daha az sayıda bulunması, dengesizliğe yol açarak “False Negative” oranlarını yükseltmektedir.

Dengesiz veri kümelerinde Synthetic Minority Oversampling Technique (SMOTE) veya ağırlıklı sınıflandırma gibi yöntemlerin kullanımı önerilmiştir (109).



3. GEREÇ VE YÖNTEM

Retrospektif olarak planlanan çalışmamız, Akdeniz Üniversitesi Tıp Fakültesi Tıbbi Bilimsel Araştırmalar Etik Kurulu'nun TBAEK-479 sayılı ve 29/05/2025 tarihli onayıyla gerçekleştirilmiştir (Ek-1).

Çalışmanın veri seti, 2015–2024 yılları arasında Akdeniz Üniversitesi Tıp Fakültesi Uyku Laboratuvarı'nda polisomnografi uygulanan hastaların kayıtlarından oluşturulmuştur.

Polisomnografi sırasında, Avustralya menşeli Compumedics© şirketi tarafından üretilen Grael v2 AH541-02 kodlu; 48 kanallı ve 60 ayrı parametreyi kayıt altına alan ekipman kullanılmıştır. Grael yardımıyla EEG, EOG, EKG ve solunum sinyalleri dâhil birçok fizyolojik parametre kaydedilmiş; sonrasında solunumsal veya nörolojik uyku bozuklukları ile diğer nörolojik durumların analizleri yapılmıştır. Veriler kayıt sırasında görüntüleme, analiz ve raporlama amaçlı bir bilgisayara aktarılmıştır. Veriler elektrotlar, sensörler ve transdüserlerin bir kombinasyonundan elde edilmiştir. Polisomnografi raporunda yer alan sinyal türleri arasında EEG, EOG, EMG, EKG, basınç, hava akışı, horlama, solunum eforu, vücut pozisyonu, uzuv hareketi, oksijen doygunluğu, nabız hızı ve nabız dalga biçimi bulunmaktadır. Hastaların PSG sonuçları Grael v2 kullanılarak güncel AASM kılavuzuna uygun şekilde manuel olarak skorlanmıştır.

PSG testleri retrospektif olarak laboratuvar arşivinden temin edilmiştir. Test içindeki gerekli veriler Excel dosyasına manuel olarak girilmiştir. Daha sonra hasta gizliliğini korumak amacıyla hastaların isim, soyisim, doğum tarihi, hastane dosya numaraları ve kimlik numaraları çıkarılarak veriler anonimleştirilmiş hâlde dijitalleştirilmiştir.

Dışlama kriterleri: 2015–2024 tarihleri arasında verileri toplanan 1258 hasta arasından; Continuous Positive Airway Pressure (CPAP) testi yapılan hastalar, 18 yaşından küçük hastalar, 240 dakikadan kısa test yapılan hastalar, eksik veri içeren hastalar ve Python ortamında verileri okunamayan hastalar çalışmadan çıkarılmıştır. Python ortamında, hastaların test tarihinden doğum yılı çıkarılarak test tarihindeki yaşları hesaplanmıştır. Sınıflama modellerinin kurulabilmesi ve sınıflama modellerinde ağır OSA hastalarının yakalanabilmesi için $AHI \geq 30$ olan

hastalara 1, AHI < 30 olanlara 0 değeri verilerek değişken sütunu elde edilmiştir. Cinsiyet değişkeni için erkeklere 1, kadınlara 0 değeri verilerek sayısal hale getirilmiştir.

OSA hastalarında cinsiyetin sadece makine öğrenme algoritmalarında AHI üzerindeki etkileri hesaplanması amaçlandı, kategorik bir değişken olarak cinsiyet farklarında AHI'nin etkisi bu çalışmada araştırılmamıştır. Sürekli değişken formatında olmayan, BİNARY formatına getirilmiş olan cinsiyet verileri analiz sürecindeki kategorik bir değişken olmadığı ve biz cinsiyet üzerinden grup farkı araştırdığımız için klasik analiz metodlarına dahil edilmedi ama makine öğrenim algoritmalarına bağımsız değişken olarak dahil edilmiştir.

Daha önce Bölüm 2'de bahsedildiği gibi, model tasarımında kullanmak üzere seçtiğimiz 16 değişken Python ortamına aktarılırken kolon isimleri aşağıdaki gibi düzenlenmiştir.

- VKI
- UYKU_LATANSI
- REM_LATANSI
- UYKU_PERİYODUNDAKI_UYANIKLIK_SURESI
- REM_EVRESI_SURESI
- AROUSAL_SAYISI_TOPLAM
- UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK
- SAO2_UYANIKLIK_ORTALAMASI_YUZDESI
- EN_DUSUK_SAO2_YUZDESI
- ORTALAMA_SAO2_DESATURASYONU_YUZDESI
- SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI
- ORTALAMA_KALP_HIZI_GENEL_REM
- ORTALAMA_KALP_HIZI_GENEL_NREM
- TOPLAM_BACAK_HAREKETI_SAYISI_UYKU
- YAS
- CINSIYET_BINARY

AHI_TOPLAM regresyon modelleri için hedef değişken,

AHI_SINIF sınıfama modelleri için hedef değişken olarak eklenmiştir.

3.1. Genel Yaklaşım ve Analiz

Tüm istatistiksel analizler Google Colab ortamında Python 3.10 tabanlı, Pandas, NumPy, SciPy, Statsmodels, Scikit-learn, XGBoost, Catboost, SHapley Additive exPlanations (SHAP) kütüphaneleri kullanılarak gerçekleştirilmiştir. Colab notebook kod akışı Ek-2 olarak sunulmuştur. Tüm tablo ve görseller Doğrudan Colab çıktılarından elde edilmiştir. İstatistiksel anlamlılık düzeyi $p < 0,05$ olarak kabul edilmiştir (110).

3.2. Varsayım Kontrolleri

Sürekli değişkenlerde normallik Shapiro-Wilk testi varyans homojenliği ise Levene testi ile değerlendirilmiştir. Literatürde Shapiro-wilk testinin yüksek güce sahip olduğu ve Levene testinin Bartlett testine kıyasla daha dayanıklı olduğu rapor edilmiştir (111, 112).

3.3. Grup Karşılaştırmaları

İki grup arasındaki karşılaştırmalarda varsayımlar sağlandığında Student t testi; varyans homojenliği bozulduğunda Welch t testi; normallik varsayımı sağlanmadığında ise Mann–Whitney U testi uygulanmıştır (113, 114).

Bu çalışmada cinsiyet değişkeni klasik istatistiksel analizlere dahil edilmemiştir. Bunun nedeni cinsiyetin yalnızca kategorik bir değişken olması ve çalışma tasarımında “erkek–kadın farkına” yönelik hipotezsel bir karşılaştırma yapılmamış olmasıdır. Ancak makine öğrenmesi modellerinde bağımsız değişkenlerden biri olarak kullanılmıştır. Bu yaklaşım, literatürde öngörü modellerinde klinik-demografik değişkenlerin “predictor” olarak değerlendirilmesine yönelik metodoloji ile uyumludur (115).

3.4. İlişki Analizleri

Sürekli değişkenler arasındaki doğrusal ilişkiler Pearson korelasyonu ile; varsayımlar sağlanmadığında ise Spearman korelasyonu ile değerlendirilmiştir. Spearman katsayısı monoton ilişkiler için uygun görülmektedir (116).

3.5. Çok Değişkenli Modeller

AHI ile ilişkili faktörlerin bağımsız etkilerini incelemek üzere lojistik regresyon ve doğrusal regresyon analizleri gerçekleştirilmiştir.

3.6. Makine Öğrenmesi ve Hiperparametre Optimizasyonu

Klasik yöntemlere ek olarak RandomForest, XGBoost ve CatBoost algoritmaları hem regresyon hem de sınıflandırma modelleri için kullanılmış; genellenebilirliği artırmak amacıyla 5 katlı çapraz doğrulama uygulanmıştır (117). Hiperparametreler GridSearchCV ardından Bayesçi optimizasyon ile belirlenmiştir (118).

3.7. Performans Ölçütleri

Sınıflandırma performanslarında ROC-AUC, Accuracy, F1 skoru, Precision, Recall, duyarlılık (sensitivity) ve özgüllük (specificity) raporlanmıştır. Regresyon modellerinde ise RMSE, MAE, MSE ve R2 kullanılmıştır. Bu metrikler klinik öngörü modellerinde sıklıkla kabul edilen ölçütlerdir (119).

Regresyon modellerinin karşılaştırılmasında R2, AIC, BIC, RMSE ve MAE kullanılmış; sınıflandırma modelleri arasında ise McNemar testi ile değerlendirme yapılmıştır (120).

3.8. Klinik Senaryo Analizi

Son aşamada CatBoost modeli klinik uygunluk açısından test edilmiş; ROC eğrisinde optimal cut-off değeri Youden J indeksi ile belirlenmiştir (121). Klinik fayda Decision Curve Analysis (DCA) ile değerlendirilmiştir (122).

3.9. Yapay Zeka Açıklanabilirliği

Makine öğrenimi modellerindeki değişken katkılarını açıklamak için SHAP kullanılmıştır; her özelliğin katkısı gösterilerek modelin şeffaflığı artırılmıştır (123).

4. BULGULAR

4.1. Veri Yükleme ve Doğrulama

Çizelge 4.1. Genel veri seti analizi

Column	dtype	n_missing	pct_missing	n_unique	Examples
VKI	float64	0	0.0	204	26.6, 30.1, 26.4
UYKU_LATANSI	float64	0	0.0	143	13.0, 76.5, 29.5
REM_LATANSI	float64	0	0.0	409	81.5, 202.5, 159.0
UYKU_PERIYODUNDAKI_UYANI KLIK_SURESI	float64	0	0.0	253	30.5, 25.5, 30.0
REM_EVRESI_SURESI	float64	0	0.0	248	102.5, 32.5, 63.0
AROUSAL_SAYISI_TOPLAM	float64	0	0.0	307	63.0, 109.0, 138.0
AHI_TOPLAM	float64	0	0.0	504	3.0, 32.7, 21.5
UYKU_SURESINDE_POZISYON_S IRTUSTU_SURE_DK	float64	0	0.0	491	21.0, 60.5, 317.0
SAO2_UYANIKLIK_ORTALAMAS I_YUZDESI	float64	0	0.0	12	96.0, 97.0, 95.0
EN_DUSUK_SAO2_YUZDESI	float64	0	0.0	47	91.0, 89.0, 78.0
ORTALAMA_SAO2_DESATURAS YONU_YUZDESI	float64	0	0.0	23	4.0, 2.0, 5.0
SAO2_DESATURASYONU_YUZD ESI_≥3_OLAY_SAYISI	float64	0	0.0	414	21.0, 75.0, 35.0
ORTALAMA_KALP_HIZI_GENEL_ REM	float64	0	0.0	61	60.0, 68.0, 74.0
ORTALAMA_KALP_HIZI_GENEL_ NREM	float64	0	0.0	55	58.0, 67.0, 75.0
TOPLAM_BACAK_HAREKETI_SA YISI_UYKU	float64	0	0.0	468	371.0, 489.0, 158.0
YAS	float64	0	0.0	59	49.0, 54.0, 45.0
CINSIYET_BINARY	int64	0	0.0	2	1, 0
AHI_SINIF	int64	0	0.0	2	0, 1

Çalışmada 18 değişken Google Colab, Python ortamında analiz edilmiş ve 922 hasta için eksik hiç bir veri bulunamamıştır (Çizelge 4.1).

Çizelge 4.2. AHİ sınıf dağılımı

Class	Count	PCT
0	594	64.425163
1	328	35.574837

Çizelge 4.3. AHİ toplam özet

	mean	sd	median	min	max	n-non-missing
AHI_TOPLAM	28.366703	25.773708	19.35	0.3	141.6	922

922 olgunun 594 (%64,4) AHİ<30 grubunda, 328'i (35,6) AHİ≥30 grubunda yer almıştır. AHİ değerleri ortalama $28,4 \pm 25,8$, medyan 19,3 olup, minimum 0,3 ve maksimum 141,6 arasında değer almıştır (Çizelge 4.2, Çizelge 4.3).

Çizelge 4.4. Tanımlayıcı istatistikler

variable	mean	sd	median	IQR	min	max	skew	kurt	n
VKI	29.703254	4.771967	29.1	6.075	15.4	57.7	0.822769	1.847695	922.0
UYKU_LATANSI	20.725054	18.288883	15.0	19.000	0.0	147.0	2.385088	8.994174	922.0
REM_LATANSI	136.548265	78.093680	116.5	89.125	0.0	472.5	1.360546	2.073093	922.0
UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI	41.993167	38.753917	28.5	46.375	0.0	216.5	1.393790	1.617403	922.0
REM_EVRESI_SURESI	68.147505	30.686958	67.0	44.000	0.5	204.5	0.305192	0.213875	922.0
AROUSAL_SAYISI_TOPLAM	124.467245	126.748997	81	101.750	4.0	802.0	2.286218	5.941973	922.0
UYKU_SURESİNDE_POZISYON_SIRTUSTU_SURE_DK	151.547180	100.342133	137.0	149.250	0.0	456.5	0.576894	-0.280325	922.0
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	95.496746	1.395078	96.0	1.000	88.0	99	-0.923557	1.985157	922.0
EN_DUSUK_SAO2_YUZDESI	82.337310	10.014544	85.0	10.000	0.0	96	-2.272938	10.302443	922.0
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	5.349241	2.912317	4.0	2.000	0.0	26.0	2.966854	11.617722	922.0
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	425.285249	7574.603330	115.5	209.000	0.0	230115.0	30.291445	916.048178	922.0
ORTALAMA_KALP_HIZI_GENEL_REM	66.801518	9.564268	66.0	11.000	6.0	115.0	0.137008	3.020396	922.0
ORTALAMA_KALP_HIZI_GENEL_NREM	66.432755	9.248901	66.0	11.000	7.0	99.0	-0.005150	2.145522	922.0
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	259.266811	218.023244	195.5	262.500	0.0	1656.0	1.860671	5.513131	922.0
YAS	46.977223	11.243168	48.0	16.000	18.0	78.0	-0.106075	-0.413573	922.0

Dahil edilen 922 bireyin tanımlayıcı istatistik tablosudur (Çizelge 4.4).

Mean (ortalama), Standart Sapma (sd), Median (medyan), İQR (çeyrekler Arası Aralık), Skewness (çarpıklık-veri dağılım simetrisi), Kurtosis (basıklık-dağılım sivriligi yada basıklığı)

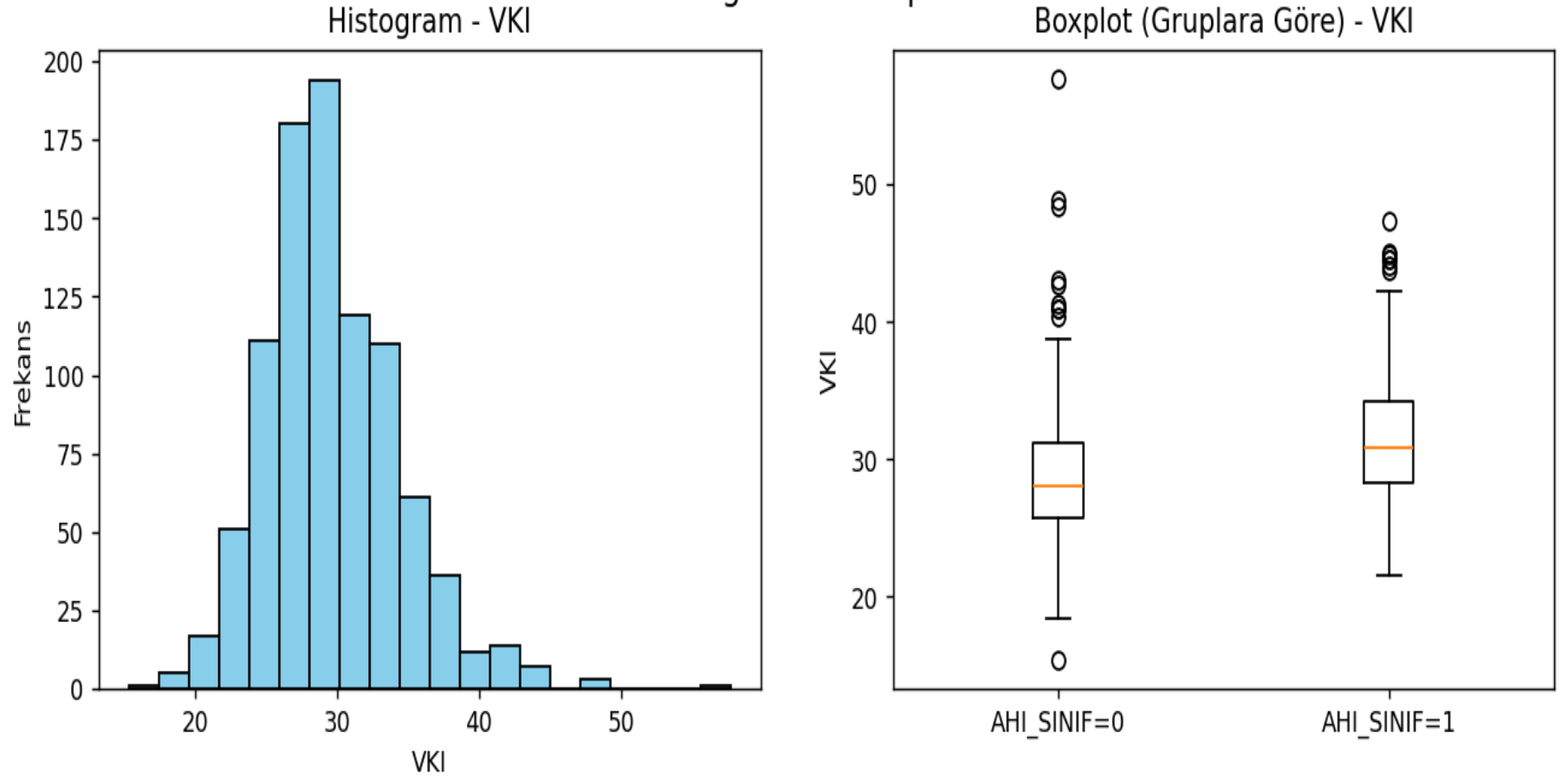
Çizelge 4.5. Grup farkı analizi (AHI_SINIF=0 vs 1)

variable	n0	n1	U_stat	p_value	cliffs_delta	p_adj_BH
VKI	594	328	62723.0	3.179281e-19	-0.356132	7.948203e-19
UYKU_LATANSI	594	328	104868.5	5.419385e-02	0.076502	5.806484e-02
REM_LATANSI	594	328	82043.0	7.154980e-05	-0.157808	9.756791e-05
UYKU_PERİYODUNDAKI_UYANIKLIK_SURESi	594	328	104314.0	7.477777e-02	0.070810	7.477777e-02
REM_EVRESi_SURESi	594	328	123214.0	2.661724e-11	0.264823	4.990733e-11
AROUSAL_SAYISI_TOPLAM	594	328	25893.0	3.204696e-76	-0.734202	1.471888e-75
UYKU_SURESİNDE_POZISYON_SIRTUSTU_SURE_DK	594	328	86326.5	4.176170e-03	-0.113837	5.220213e-03
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	594	328	128425.5	1.543870e-16	0.318320	3.308293e-16
EN_DUSUK_SAO2_YUZDESI	594	328	168807.0	3.925034e-76	0.732847	1.471888e-75
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	594	328	27423.0	8.900920e-78	-0.718496	6.675690e-77
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	594	328	6483.5	5.178426e-122	-0.933445	7.767638e-121
ORTALAMA_KALP_HIZI_GENEL_REM	594	328	74660.0	4.051893e-09	-0.233596	6.077840e-09
ORTALAMA_KALP_HIZI_GENEL_NREM	594	328	73089.0	3.210095e-10	-0.249723	5.350159e-10
TOPLAM_BACAK_HAREKETi_SAYISI_UYKU	594	328	47897.0	1.823508e-37	-0.508325	5.470523e-37
YAS	594	328	89947.0	5.360116e-02	-0.076671	5.806484e-02

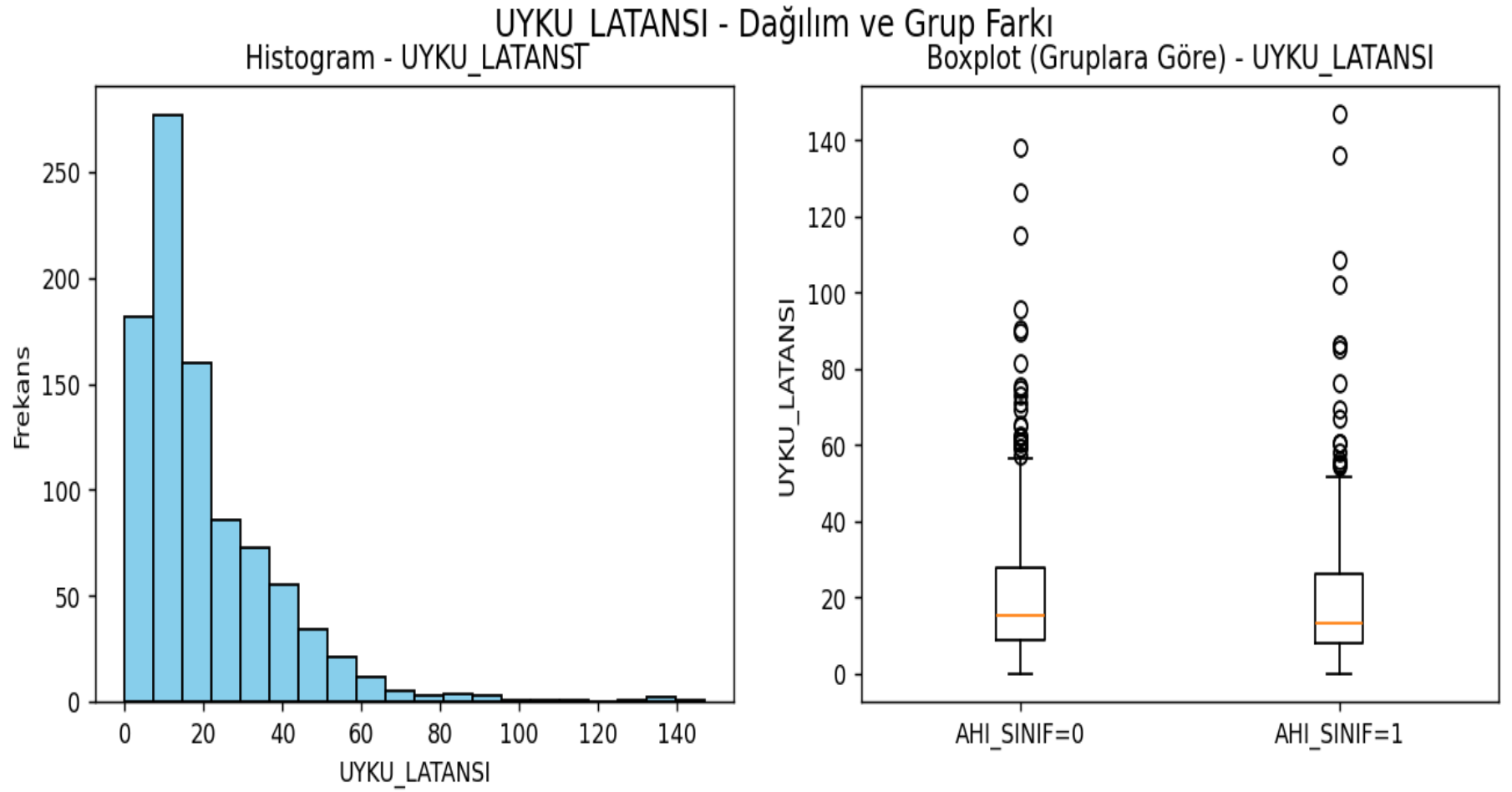
No ve N1, AHI_SINIF için sınıf 0 ve sınıf 1'i ifade etmektedir (Çizelge 4.5). U-Stat(Mann-Whitney U test), P_value (istatistiksel anlamlılık düzeyi), Cliff's Delta (etki büyüklüğü), P_adj_BH (Çoklu düzeltme sonrası anlamlılık değeri; benjamini-Hochberg)



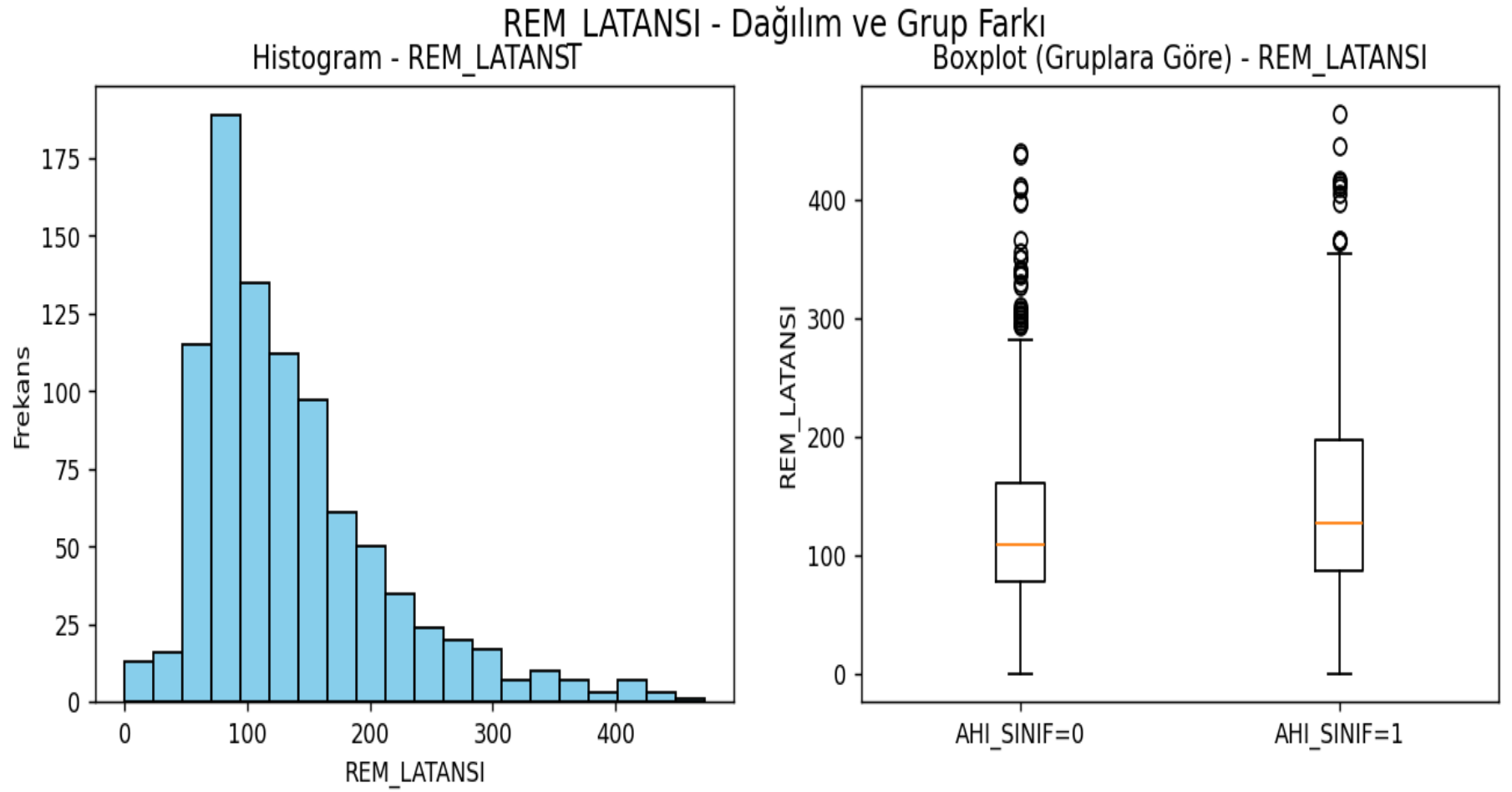
VKI - Dağılım ve Grup Farkı



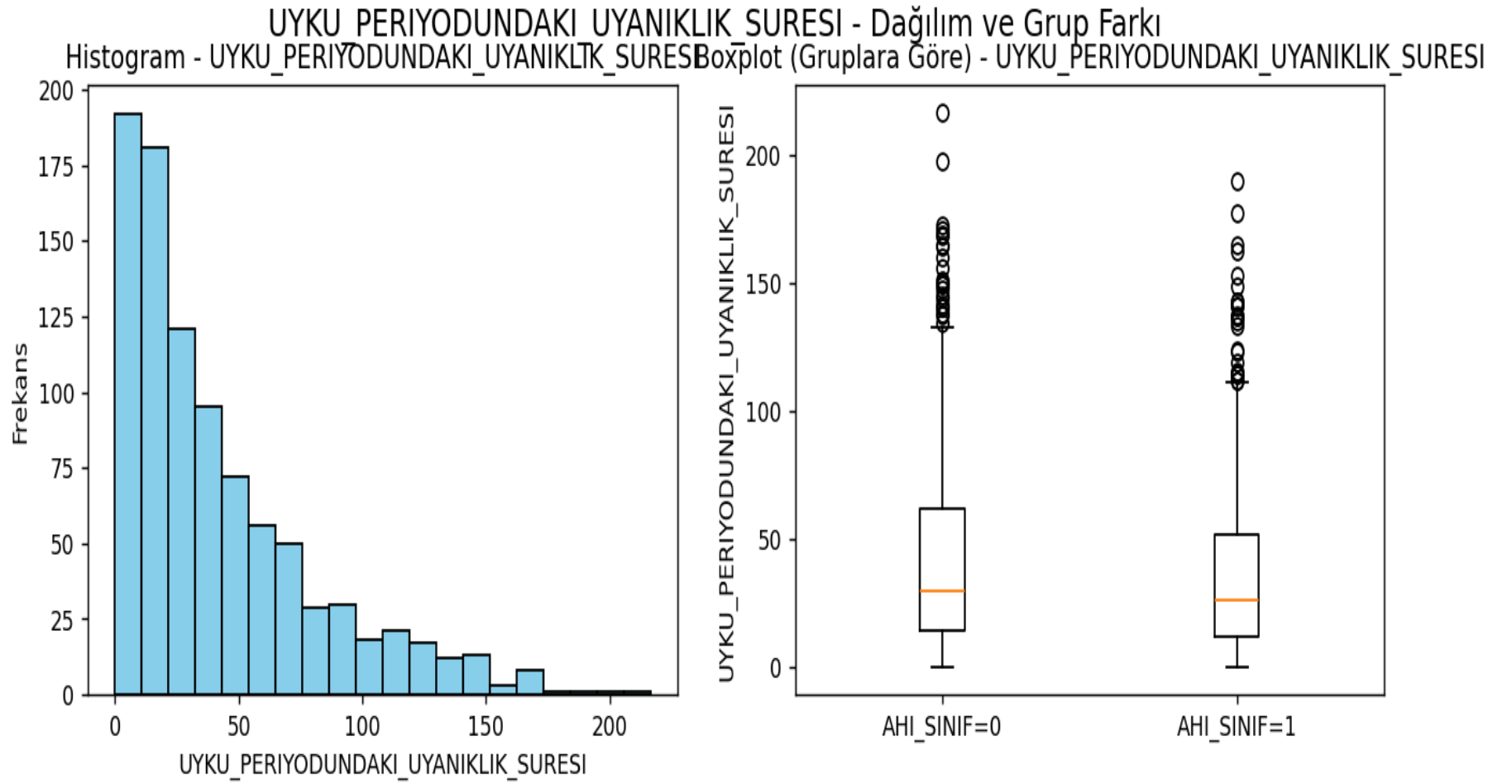
Şekil 4.1. VKİ



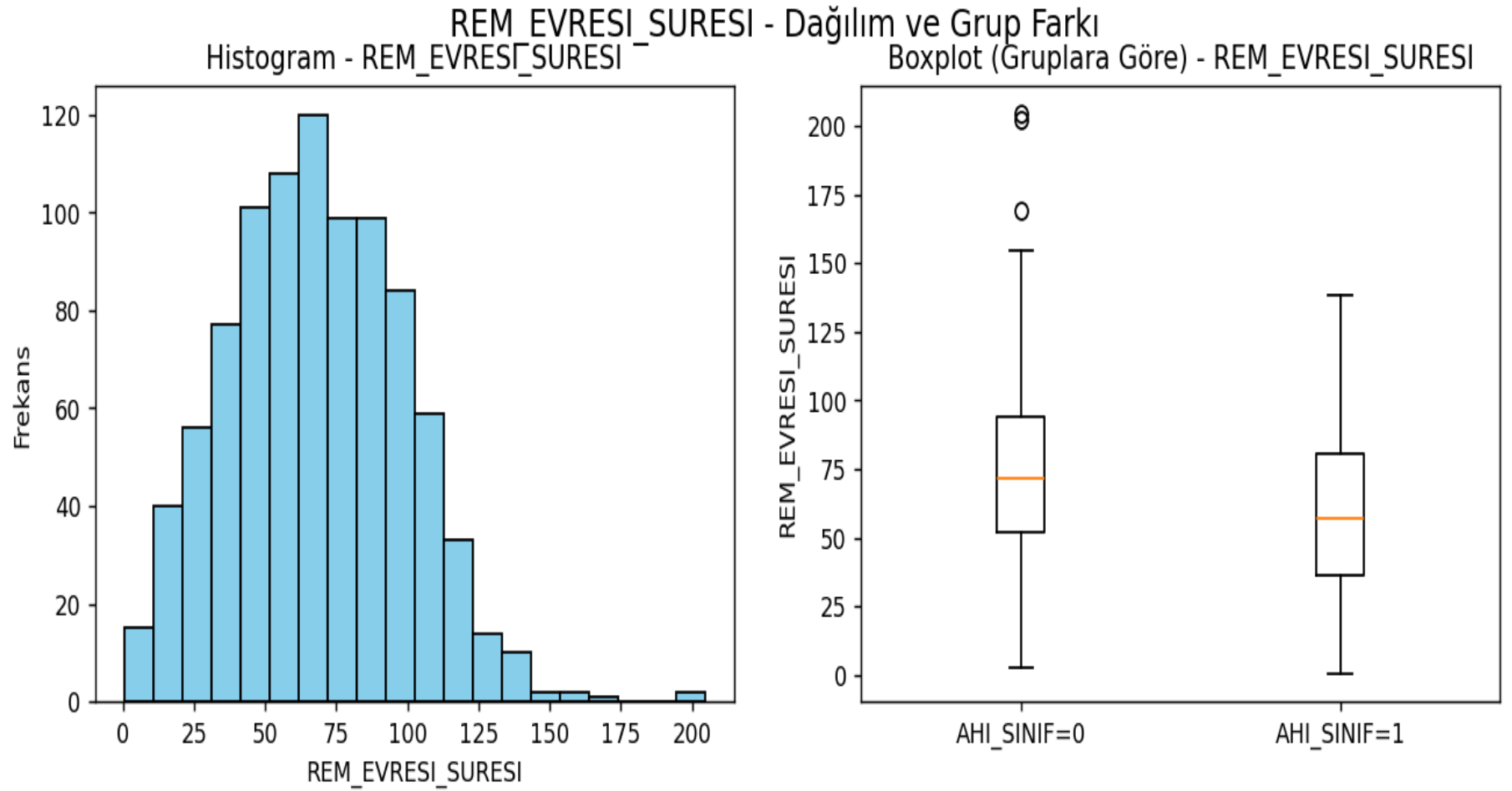
Şekil 4.2. Uyku latansı



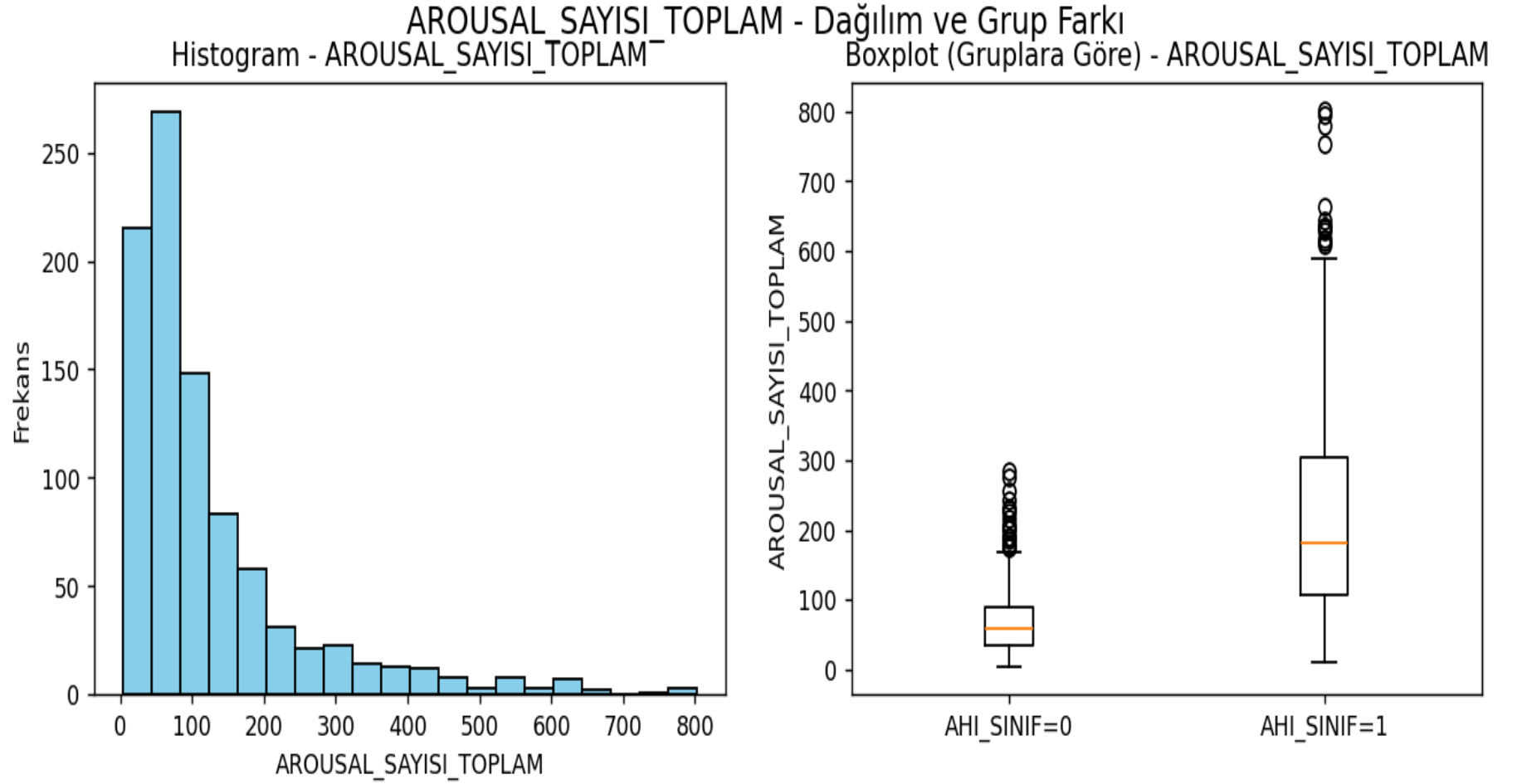
Şekil 4.3. Rem latansı



Şekil 4.4. Uyku periyodunda ki uyanıklık süresi

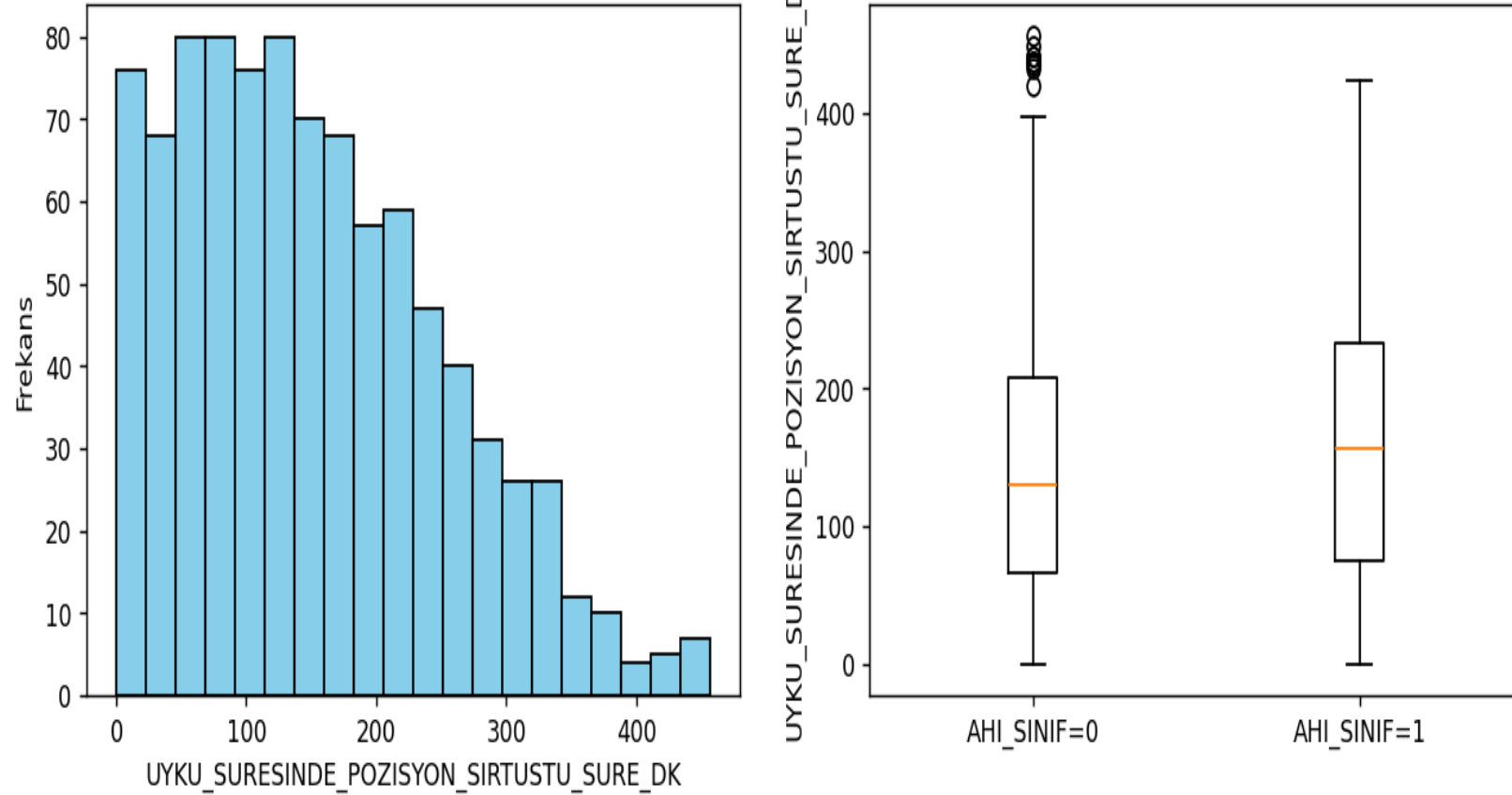


Şekil 4.5. REM evresi süresi



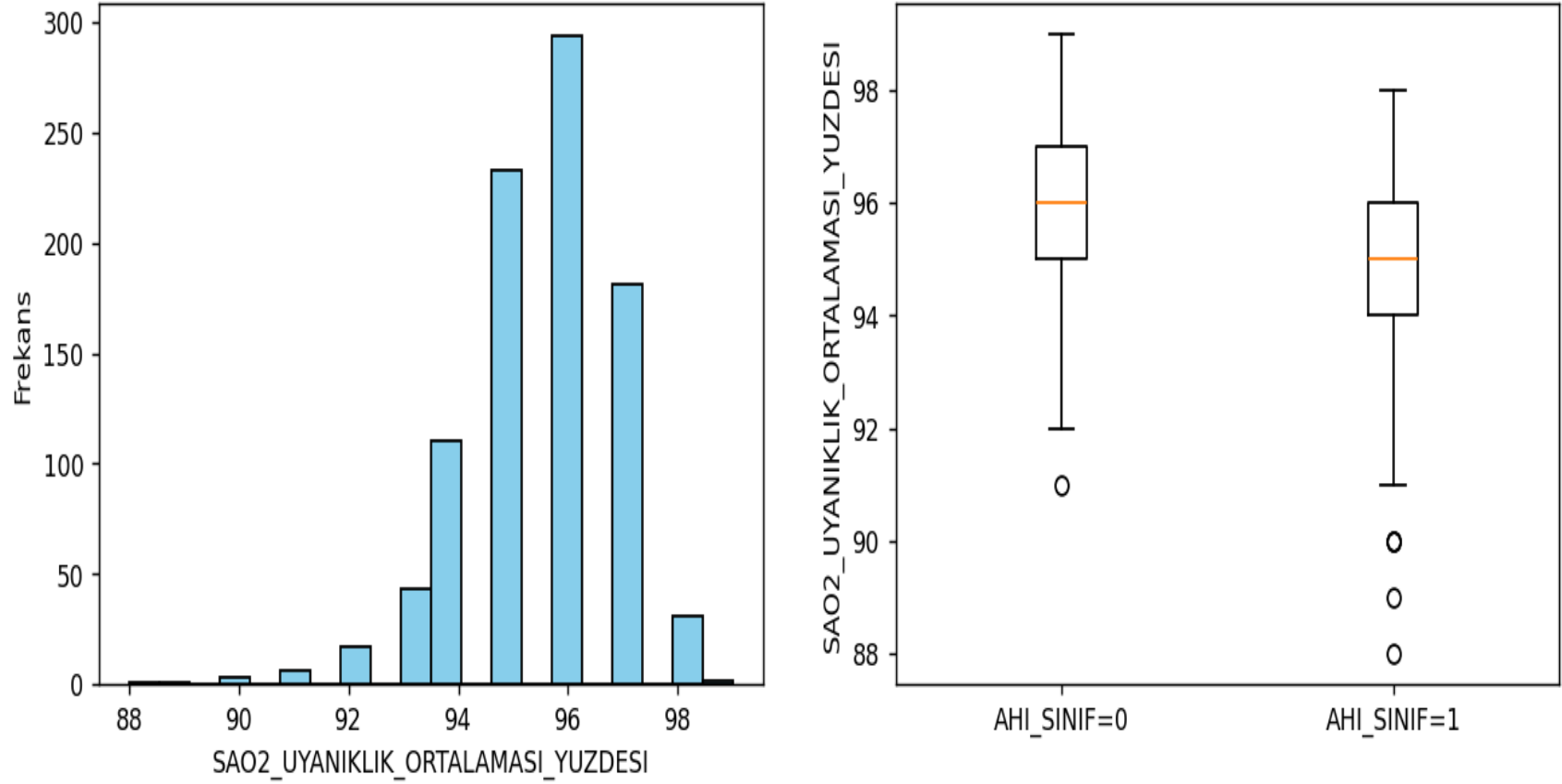
Şekil 4.6. Arousal sayısı toplam

UYKU SÜRESİNDE POZİSYON SİRTÜSTÜ SÜRE_DK - Dağılım ve Grup Farkı
 Histogram - UYKU_SÜRESİNDE_POZİSYON_SİRTÜSTÜ_SÜRE_DK (Gruplara Göre) - UYKU_SÜRESİNDE_POZİSYON_SİRTÜSTÜ_SÜRE_DK

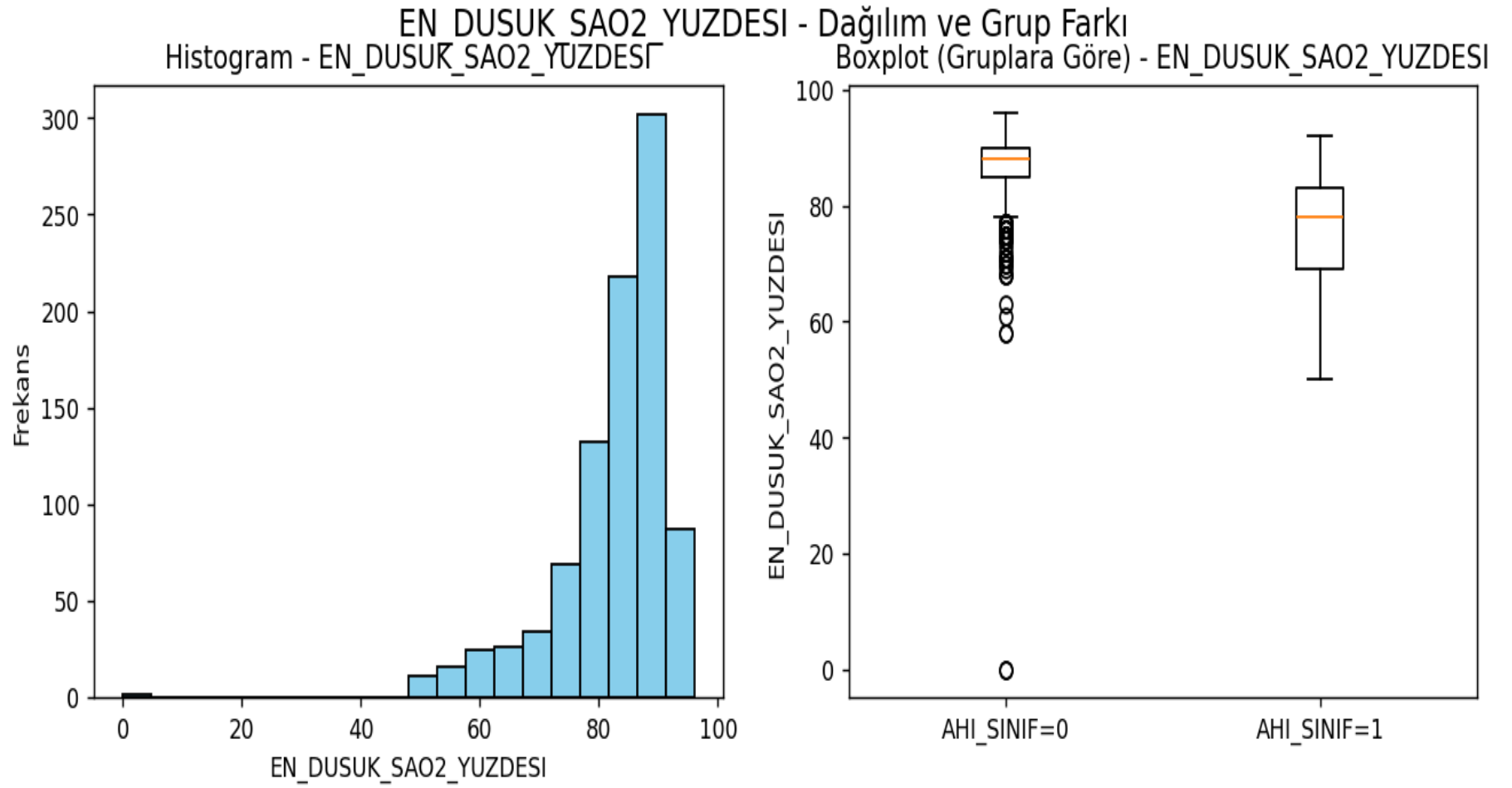


Şekil 4.7. Uyku süresinde pozisyon sırtüstü süre (dk)

SAO2 UYANIKLIK ORTALAMASI YUZDESİ - Dağılım ve Grup Farkı
Histogram - SAO2_UYANIKLIK_ORTALAMASI_YUZDESİ Boxplot (Gruplara Göre) - SAO2_UYANIKLIK_ORTALAMASI_YUZDESİ

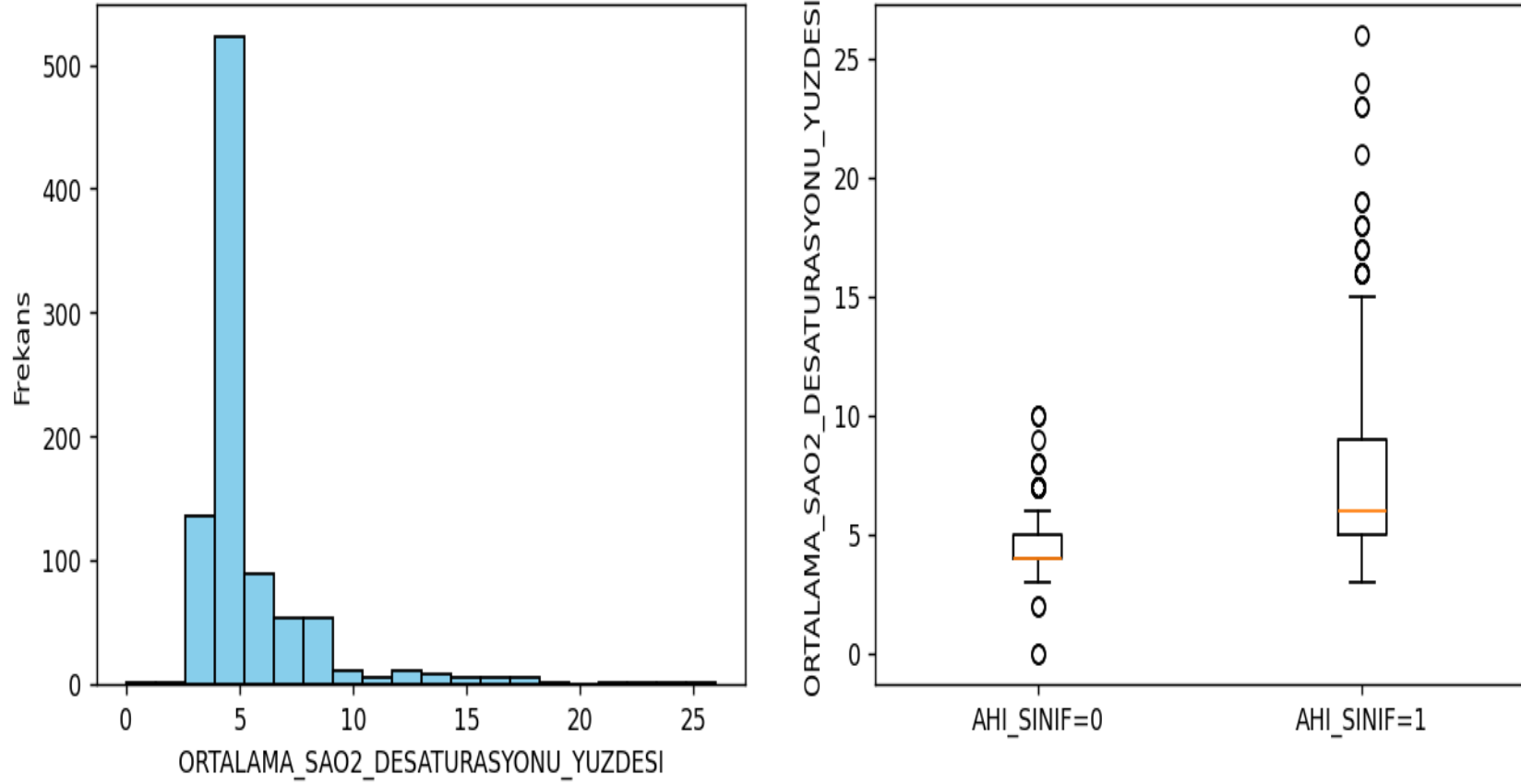


Şekil 4.8. SaO₂ uyanıklık ortalaması yüzdesi



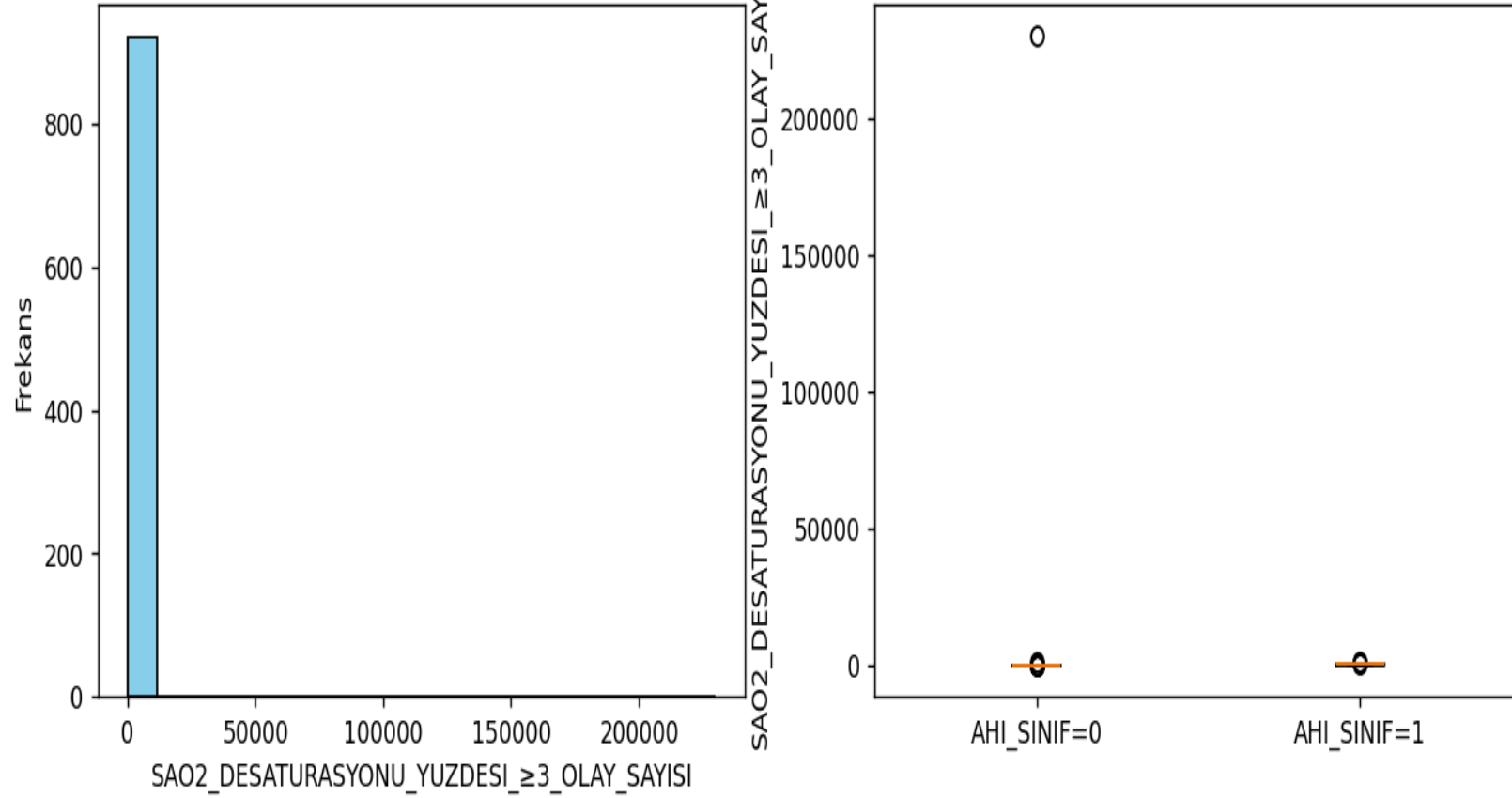
Şekil 4.9. En düşük SaO₂ yüzdesi

ORTALAMA SAO2 DESATURASYONU YUZDESI - Dağılım ve Grup Farkı
Histogram - ORTALAMA_SAO2_DESATURASYONU_YUZDESI
Boxplot (Gruplara Göre) - ORTALAMA_SAO2_DESATURASYONU_YUZDESI

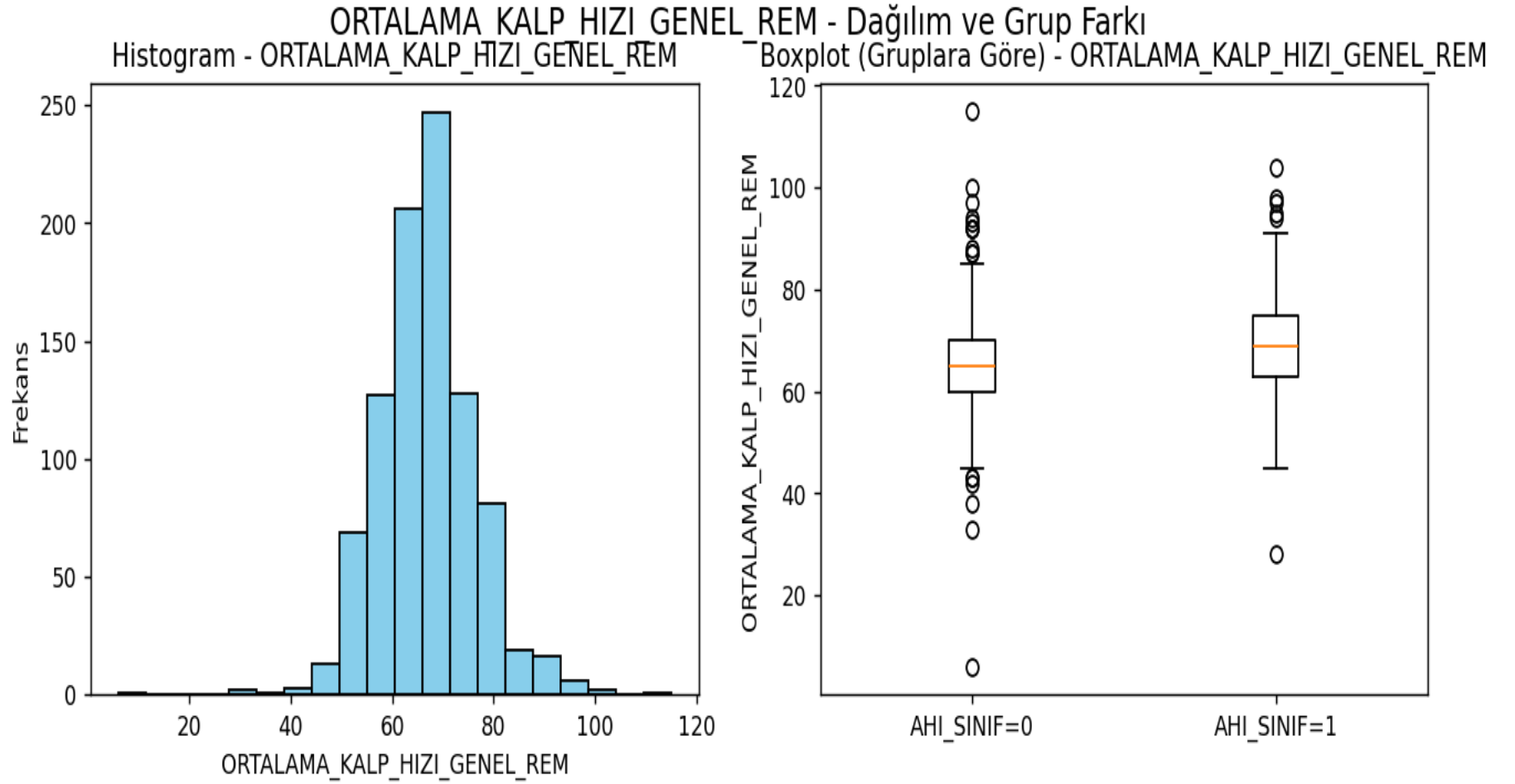


Şekil 4.10. Ortalama SaO₂ desaturasyonu yüzdesi

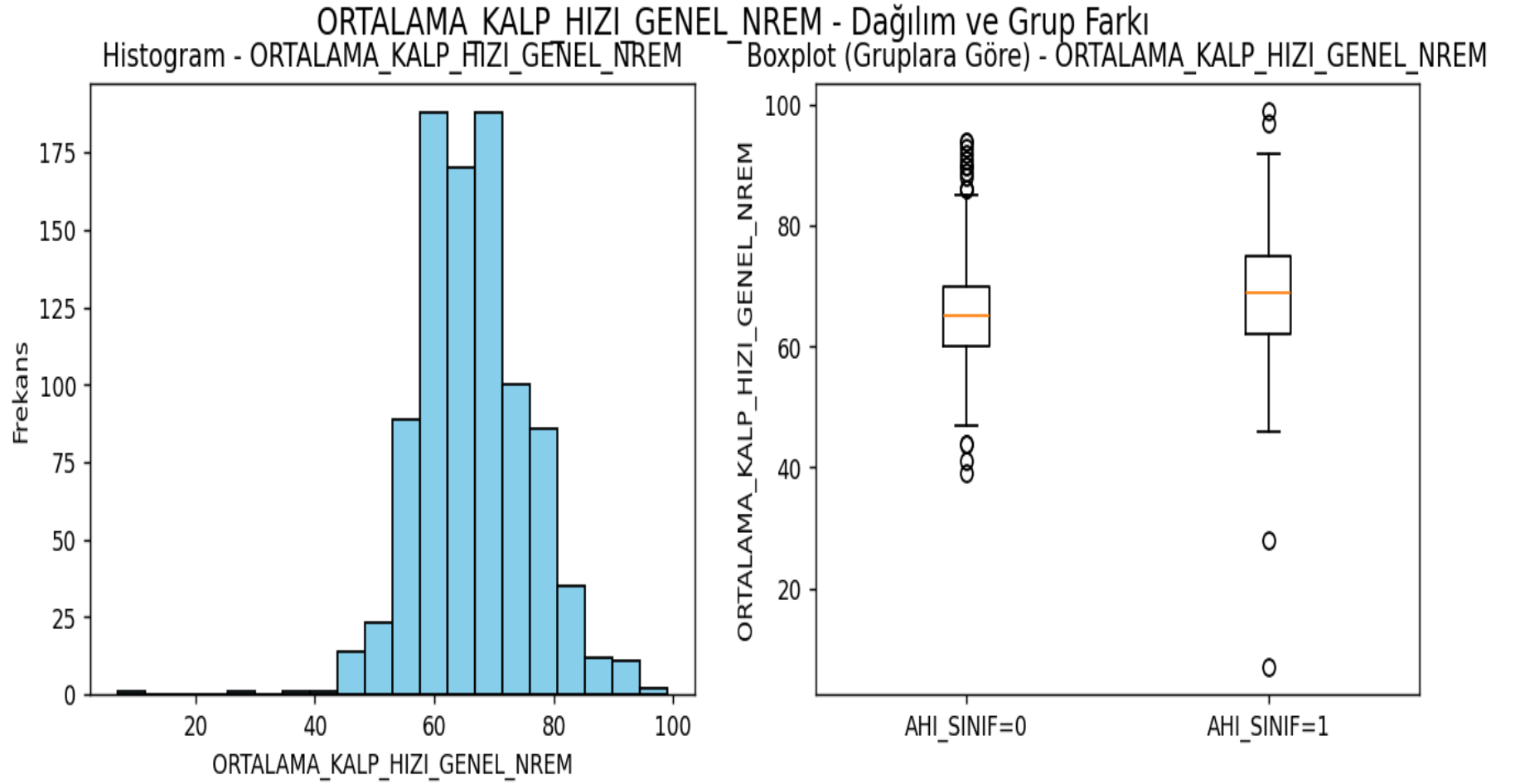
SAO2 DESATURASYONU YUZDESI ≥ 3 OLAY SAYISI - Dağılım ve Grup Farkı
Histogram - SAO2_DESATURASYONU_YUZDESI ≥ 3 OLAY SAYISI (Gruplara Göre) - SAO2_DESATURASYONU_YUZDESI ≥ 3 OLAY SAYISI



Şekil 4.11. SaO₂ desaturasyonu yüzdesi ≥ 3 olay sayısı

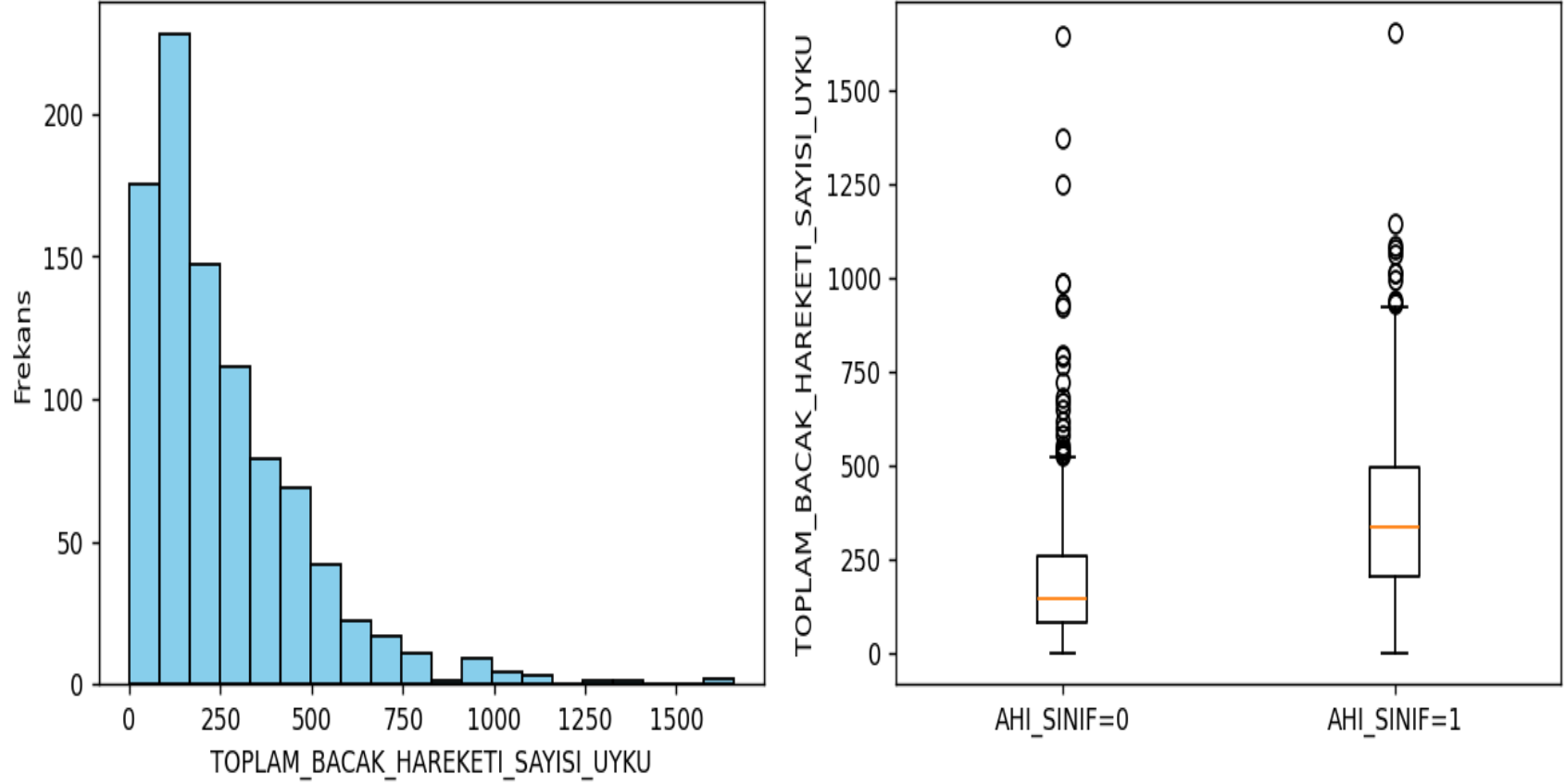


Şekil 4.12. Ortalama kalp hızı genel REM



Şekil 4.13. Ortalama kalp hızı genel NREM

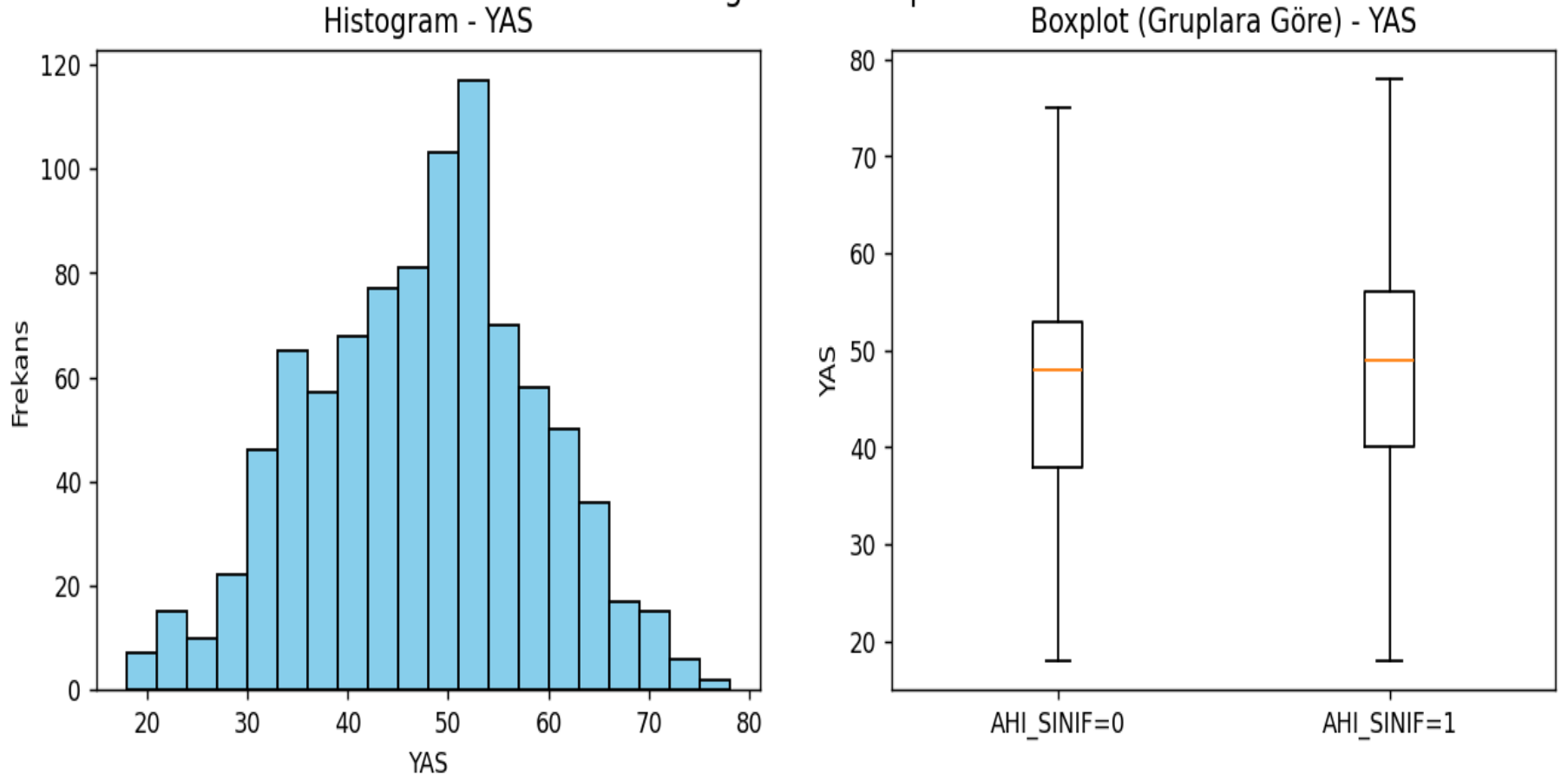
TOPLAM BACAK HAREKETİ SAYISI UYKU - Dağılım ve Grup Farkı
Histogram - TOPLAM_BACAK_HAREKETİ_SAYISI_UYKU Boşplot (Gruplara Göre) - TOPLAM_BACAK_HAREKETİ_SAYISI_UYKU



Şekil 4.14. Toplam bacak hareketi sayısı uyku



YAS - Dağılım ve Grup Farkı



Şekil 4.15. Yaş

Şekil 4.1 – Şekil 4.15 tanımlayıcı istatistikler her değişken için gösterilmiş hal (histogram + boxplot).

Çizelge 4.6. AHI sınıflaması ile spearman kolerasyonu

Değişken	Spearman_r	p	q_BH
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	0.774	0.000	0.000
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	0.615	0.000	0.000
AROUSAL_SAYISI_TOPLAM	0.609	0.000	0.000
EN_DUSUK_SAO2_YUZDESI	-0.608	0.000	0.000
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	0.422	0.000	0.000
VKI	0.295	0.000	0.000
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	-0.272	0.000	0.000
REM_EVRESI_SURESI	-0.220	0.000	0.000
ORTALAMA_KALP_HIZI_GENEL_NREM	0.207	0.000	0.000
ORTALAMA_KALP_HIZI_GENEL_REM	0.194	0.000	0.000
REM_LATANSI	0.131	0.000	0.000
CINSIYET_BINARY	0.096	0.003	0.004
UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK	0.094	0.004	0.005
YAS	0.064	0.054	0.058
UYKU_LATANSI	-0.063	0.054	0.058
UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI	-0.059	0.075	0.075

Burada bağımlı değişken AHI_SINIF ile diğer tüm değişkenler arasındaki ilişkiler verilmiştir (Çizelge 4.6).

SaO₂ desatürasyon ≥3% olay sayısı (r = 0.774, p<0.001)

Ortalama desatürasyon yüzdesi (r = 0.615, p<0.001)

Arousal sayısı (r = 0.609, p<0.001)

En düşük SaO₂ (r = -0.608, p<0.001)

Çizelge 4.7. AHİ toplamı ile spearman

Değişken	Spearman_r	p	q_BH
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	0.932	0.000	0.000
EN_DUSUK_SAO2_YUZDESI	-0.749	0.000	0.000
AROUSAL_SAYISI_TOPLAM	0.726	0.000	0.000
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	0.711	0.000	0.000
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	0.527	0.000	0.000
VKI	0.400	0.000	0.000
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	-0.344	0.000	0.000
REM_EVRESI_SURESI	-0.221	0.000	0.000
ORTALAMA_KALP_HIZI_GENEL_NREM	0.213	0.000	0.000
ORTALAMA_KALP_HIZI_GENEL_REM	0.191	0.000	0.000
YAS	0.148	0.000	0.000
CINSIYET_BINARY	0.131	0.000	0.000
REM_LATANSI	0.105	0.001	0.002
UYKU_LATANSI	-0.069	0.036	0.041
UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK	0.052	0.114	0.121
UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI	-0.050	0.130	0.130

Sürekli değişken olan AHİ_TOPLAM ile cinsiyet hariç 15 değişken iççelenmiştir (Çizelge 4.7).

SaO₂ desatürasyon ≥3% olay sayısı (r = 0.932, p<0.001)

En düşük SaO₂ (r = -0.749, p<0.001)

Arousal sayısı (r = 0.726, p<0.001)

Ortalama desatürasyon yüzdesi (r = 0.711, p<0.001)

Çizelge 4.8. Yüksek korelasyon çiftleri

Değişken_1	Değişken_2	Spearman_r	p
ORTALAMA_KALP_HIZI_GENEL_REM	ORTALAMA_KALP_HIZI_GENEL_NREM	0.926	0.0

Spearman kolerasyon katsayısı $|\rho| \geq 0.90$ olan deęişkenler listelenmiştir. Sadece Ortakana kalp hızı REM - NREM arasında yüksek kolerasyon görölmüştür (Çizelge 4.8).

Çizelge 4.9. VİF (klasik)

Deęişken	VİF
ORTALAMA_KALP_HIZI_GENEL_NREM	5.065749
ORTALAMA_KALP_HIZI_GENEL_REM	4.949240
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	2.732550
EN_DUSUK_SAO2_YUZDESI	2.367322
AROUSAL_SAYISI_TOPLAM	2.072646
REM_EVRESI_SURESI	1.553900
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	1.469364
UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI	1.457063
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	1.440349
YAS	1.418990
REM_LATANSI	1.402529
VKI	1.265320
UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK	1.241829
UYKU_LATANSI	1.078505
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	1.008476

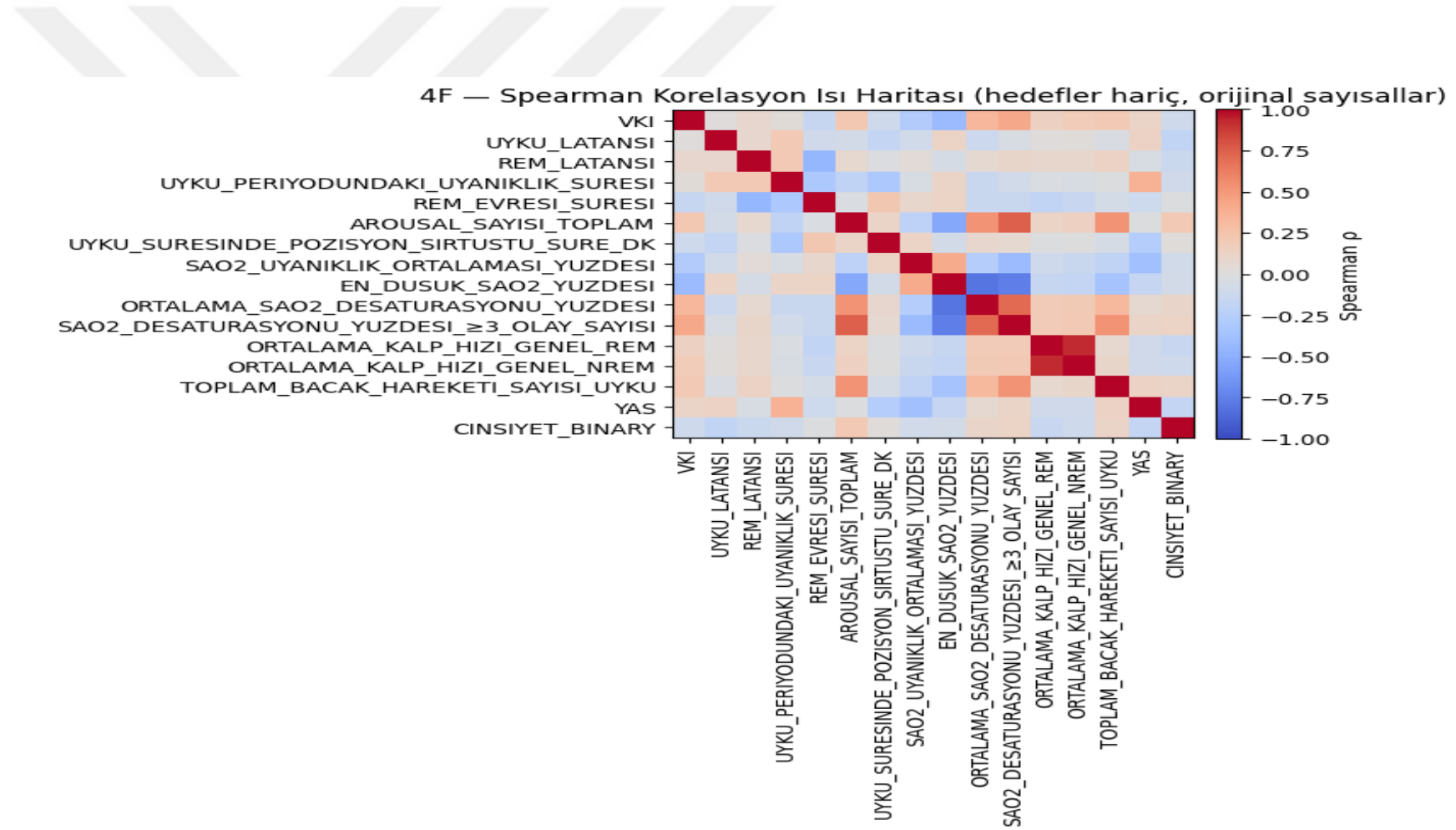
Çizelge 4.10. VİF (modelleme)

Deęişken	VİF
ORTALAMA_KALP_HIZI_GENEL_NREM	5.077300
ORTALAMA_KALP_HIZI_GENEL_REM	5.033858
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	2.741507
EN_DUSUK_SAO2_YUZDESI	2.372087
AROUSAL_SAYISI_TOPLAM	2.087710
REM_EVRESI_SURESI	1.578836
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	1.515604
YAS	1.495696

Çizelge 4.10 (Devam). VİF (modelleme)

Değişken	VİF
UYKU_PERİYODUNDAKI_UYANIKLIK_SURESI	1.458122
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	1.445679
REM_LATANSI	1.440565
VKI	1.311996
UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK	1.244897
CINSIYET_BINARY	1.217041
UYKU_LATANSI	1.105463
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	1.008868

VİF (Variance İnflation Factor) değişkenler arasında multikolineriteyi ölçmek için kullanılmıştır cinsiyet hariç (Çizelge 4.9) ve cinsiyet dahil (Çizelge 4.10) olmak üzere 2 ayrı analiz yapılmıştır. Tüm değişkenlerin, her iki tabloda da, VİF değerleri 10'un altında olup çoklu doğrusal bağlantının kritik seviyede olmadığı gösterilmiştir.



Şekil 4.16. Spearman korelasyon ısı haritası

Spearman korelasyon ısı haritası, tüm değişkenlerin korelasyon alt yapısını göstermektedir. Koyu kırmızı alanlar güçlü pozitif korelasyon gösterirken mavi alanlar ise negatif korelasyon göstermektedir (Şekil 4.16).

Çizelge 4.11. Normalite ve varyans homojenliği

variable	n_total	shapiro_W	shapiro_p	levене_stat	levене_p
VKI	922	0.967469	1.738814e-13	2.724487	9.916115e-02
UYKU_LATANSI	922	0.789076	6.812636e-33	0.199906	6.549016e-01
REM_LATANSI	922	0.893143	8.954655e-25	6.396401	1.160129e-02
UYKU_PERİYODUNDAKI_UYANIKLIK_SURESI	922	0.855576	3.198866e-28	1.097836	2.950167e-01
REM_EVRESI_SURESI	922	0.989701	4.588015e-06	0.008024	9.286450e-01
AROUSAL_SAYISI_TOPLAM	922	0.740328	1.283547e-35	302.125024	9.798906e-59
UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK	922	0.962050	9.683835e-15	1.197874	2.740335e-01
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	922	0.914956	2.547184e-22	4.591519	3.239232e-02
EN_DUSUK_SAO2_YUZDESI	922	0.819517	6.330339e-31	72.520208	6.676995e-17
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	922	0.679815	1.770851e-38	167.969733	2.097706e-35
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	922	0.017840	9.589362e-56	0.358304	5.495972e-01
ORTALAMA_KALP_HIZI_GENEL_REM	922	0.971203	1.545099e-12	6.277329	1.240102e-02
ORTALAMA_KALP_HIZI_GENEL_NREM	922	0.979854	5.585731e-10	9.324925	2.325632e-03
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	922	0.845503	5.022267e-29	37.663564	1.247905e-09
YAS	922	0.994004	9.652053e-04	0.000195	9.888521e-01

Shapiro ve levene testi: Tüm değişkenler için Shapiro-wilk $p < 0.05$ çıkmıştır, hiç bir değişken normal dağılım varsayımını karşılamamaktadır, parametrik olmayan testler tercih edilmiştir. Levene test sonucuna göre; $p < 0.05$ Arousal sayısı, En düşük SaO₂,

Ortalama desatürasyon yüzdesi, $\text{SaO}_2 \geq 3\%$ olay sayısı, Toplam bacak hareketi, Yaş $p < 0.05$ çıkmıştır gruplar arası varyans homojenliği bozulmuştur (Çizelge 4.11).

Çizelge 4.12. Grup kıyaslaması

variable	n0	n1	test	stat	p_value	effect_size	p_adj_BH
VKI	594	328	Mann-Whitney U	62723.0	3.179281e-19	-0.356132	7.948203e-19
UYKU_LATANSI	594	328	Mann-Whitney U	104868.5	5.419385e-02	0.076502	5.806484e-02
REM_LATANSI	594	328	Mann-Whitney U	82043.0	7.154980e-05	-0.157808	9.756791e-05
UYKU_PERİYODUNDAKI_UYANIKLIK_SURESI	594	328	Mann-Whitney U	104314.0	7.477777e-02	0.070810	7.477777e-02
REM_EVRESI_SURESI	594	328	Mann-Whitney U	123214.0	2.661724e-11	0.264823	4.990733e-11
AROUSAL_SAYISI_TOPLAM	594	328	Mann-Whitney U	25893.0	3.204696e-76	-0.734202	1.471888e-75
UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK	594	328	Mann-Whitney U	86326.5	4.176170e-03	-0.113837	5.220213e-03
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	594	328	Mann-Whitney U	128425.5	1.543870e-16	0.318320	3.308293e-16
EN_DUSUK_SAO2_YUZDESI	594	328	Mann-Whitney U	168807.0	3.925034e-76	0.732847	1.471888e-75
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	594	328	Mann-Whitney U	27423.0	8.900920e-78	-0.718496	6.675690e-77
SAO2_DESATURASYONU_YUZDESI ≥ 3 OLAY_SAYISI	594	328	Mann-Whitney U	6483.5	5.178426e-122	-0.933445	7.767638e-121
ORTALAMA_KALP_HIZI_GENEL_REM	594	328	Mann-Whitney U	74660.0	4.051893e-09	-0.233596	6.077840e-09
ORTALAMA_KALP_HIZI_GENEL_NREM	594	328	Mann-Whitney U	73089.0	3.210095e-10	-0.249723	5.350159e-10
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	594	328	Mann-Whitney U	47897.0	1.823508e-37	-0.508325	5.470523e-37
YAS	594	328	Mann-Whitney U	89947.0	5.360116e-02	-0.076671	5.806484e-02

Tüm değişkenlerin dağılımı normal olmadığı için mann-Whitney u testi uygulanmıştır ve etki büyüklük (Cliffs delta) hesaplanmıştır. Anlamli fark bulunan değişkenler ve düzeltilmiş p değerleri verilmiştir (Çizelge 4.12).

Anlamli Fark: VKİ, REM Latansı, REM Evresi Süresi, Arousal Sayısı, Sırtüstü Uyuma Süresi, SaO₂ Ortalama ve En Düşük Yüzdesi, Desatürasyon Yüzdesi ve Olay Sayısı (≥ 3), Ortalama Kalp Hızı (REM/NREM), Toplam Bacak Hareketi.

Anlamli Fark Göstermeyen: Uyku Latansı, Uyku Periyodundaki Uyanıklık Süresi, Yaş
Etki Gücü (Cliffs delta)

Güçlü Etki: Arousal sayısı (-0.73), en düşük SaO₂ (0.73), desatürasyon yüzdesi (-0.72), desatürasyon olay sayısı (-0.93), bacak hareketleri (-0.50).

Orta Etki: VKİ (-0.35), REM evresi süresi (0.26).

Zayıf Etki: REM latansı (-0.15), kalp hızı (-0.23 ila -0.24)

AHI yüksek grupta obeziteye eğilim, uyku yapısında bozulma, oksijen parametrelerinde şiddetli azalma, artmış kalp ve bacak hızı gözlenmiştir.

Çizelge 4.13. Hedeflerde tek değişkenli regresyon analizi

variable	model	beta	ci_low	ci_high	p_value
VKI	Linear (OLS)	2.087980	1.765684	2.410277	3.105466e-34
VKI	Logistic	0.132591	0.100802	0.164381	2.964419e-16
UYKU_LATANSI	Linear (OLS)	-0.068135	-0.159212	0.022942	1.423934e-01
UYKU_LATANSI	Logistic	-0.002249	-0.009745	0.005248	5.566208e-01
REM_LATANSI	Linear (OLS)	0.046362	0.025220	0.067505	1.860851e-05
REM_LATANSI	Logistic	0.003510	0.001796	0.005224	5.990918e-05
UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI	Linear (OLS)	-0.033289	-0.076267	0.009688	1.288186e-01
UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI	Logistic	-0.002741	-0.006318	0.000835	1.330159e-01
REM_EVRESI_SURESI	Linear (OLS)	-0.217702	-0.270188	-0.165215	1.275638e-15
REM_EVRESI_SURESI	Logistic	-0.016416	-0.021239	-0.011593	2.542088e-11
AROUSAL_SAYISI_TOPLAM	Linear (OLS)	0.152203	0.143478	0.160928	2.627133e-166
AROUSAL_SAYISI_TOPLAM	Logistic	0.022021	0.018833	0.025210	9.414818e-42
UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK	Linear (OLS)	0.020705	0.004140	0.037271	1.435260e-02
UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK	Logistic	0.001771	0.000433	0.003110	9.496554e-03
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	Linear (OLS)	-6.278946	-7.403165	-5.154727	2.312026e-26
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	Logistic	-0.433392	-0.541266	-0.325518	3.425949e-15
EN_DUSUK_SAO2_YUZDESI	Linear (OLS)	-1.679212	-1.805405	-1.553019	6.354672e-113
EN_DUSUK_SAO2_YUZDESI	Logistic	-0.174974	-0.200642	-0.149306	1.023337e-40
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	Linear (OLS)	6.097888	5.682896	6.512879	8.756682e-131

Çizelge 4.13 (Devam). Hedeflerde tek değişkenli regresyon analizi

variable	model	beta	ci_low	ci_high	p_value
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	Logistic	0.965986	0.821880	1.110092	1.988790e-39
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	Linear (OLS)	0.000065	-0.000156	0.000285	5.651141e-01
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	Logistic	-0.000002	-0.000022	0.000017	8.269804e-01
ORTALAMA_KALP_HIZI_GENEL_REM	Linear (OLS)	0.645635	0.476352	0.814919	1.672361e-13
ORTALAMA_KALP_HIZI_GENEL_REM	Logistic	0.044825	0.029577	0.060072	8.322565e-09
ORTALAMA_KALP_HIZI_GENEL_NREM	Linear (OLS)	0.697802	0.523240	0.872365	1.196581e-14
ORTALAMA_KALP_HIZI_GENEL_NREM	Logistic	0.046748	0.031110	0.062387	4.656910e-09
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	Linear (OLS)	0.056944	0.050241	0.063647	9.872093e-55
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	Logistic	0.004339	0.003527	0.005151	1.164283e-25
YAS	Linear (OLS)	0.185943	0.038106	0.333779	1.375220e-02
YAS	Logistic	0.013949	0.001832	0.026066	2.405535e-02

AHI ile ilişkide ≥ 30 lineer regresyon (AHI_TOPLAM ve logistic Regresyon (AHI_Sınıf) yöntemleriyle incelenmiştir. Sonuçlar her iki durumda da paralellik göstermiştir (Çizelge 4.13).

Güçlü değişkenler: Oksijenizasyon parametreleri, arousal sayısı.

Orta düzey değişkenler: VKI, REM evresi süresi, Kalp hızı (REM/NREM), yaş.

Zayıf veya anlamsız değişkenler: Uyku latansı ve Uykudaki uyanıklık süresi, rem latansı.

Çizelge 4.14. Temel makine öğrenimi değerleri – regresyon modelleri

Model	R2	RMSE
CatBoostRegressor	0.888450	8.609274
RandomForestRegressor	0.887556	8.643693
XGBRegressor	0.867353	9.388168

Çizelge 4.15. Temel makine öğrenimi değerleri – sınıflama modelleri

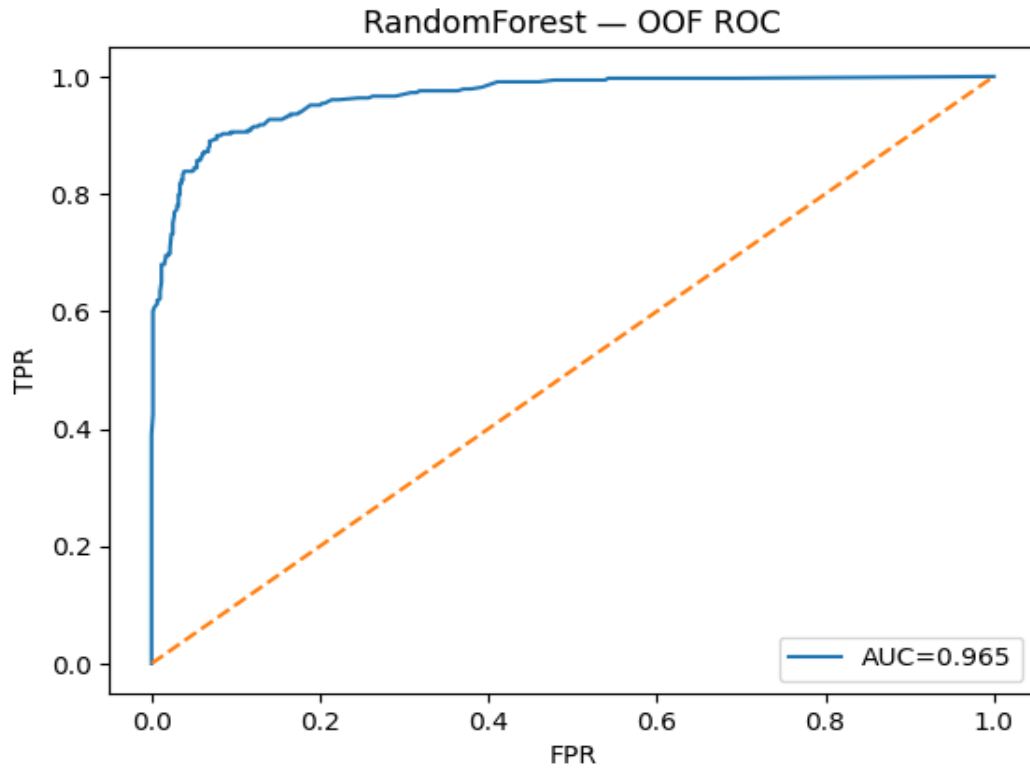
Model	R2	RMSE
CatBoostClassifier	0.908108	0.972050
XGBClassifier	0.924324	0.964224
RandomForestClassifier	0.913514	0.956149

Catboost en yüksek performans göstermiştir. ($R^2=0.888$, $RMSE=8.61$) RandomForest Regressor Cat boost'a yakındır. XGboost Görece daha düşük performans göstermiştir (Çizelge 4.14).

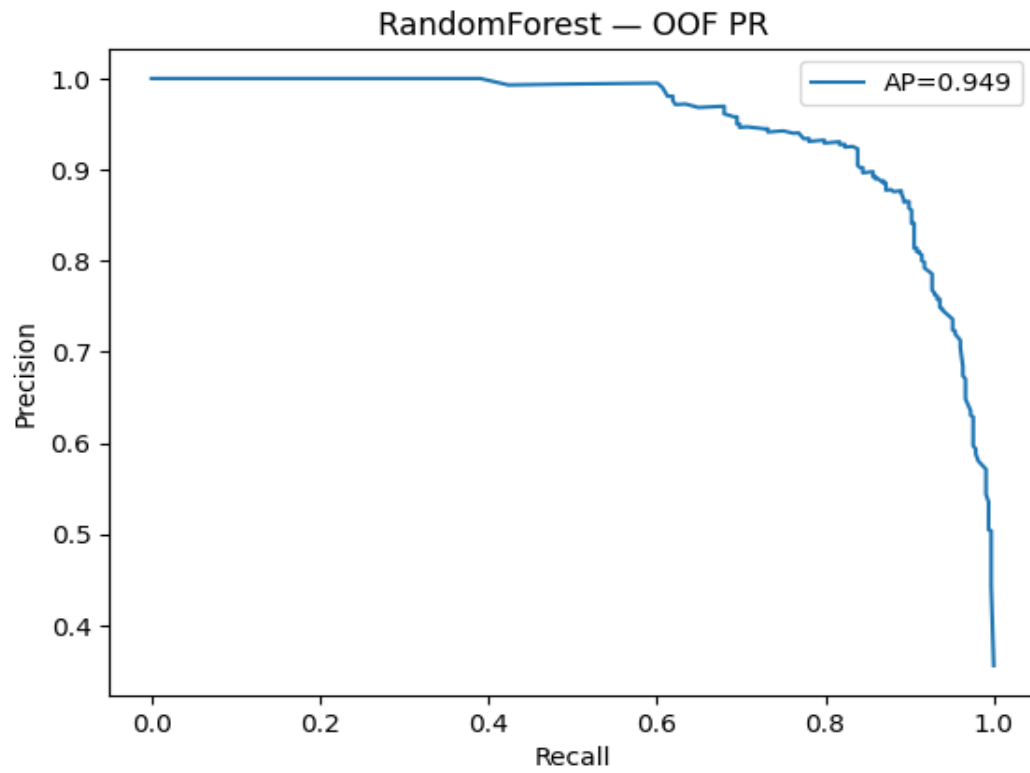
Sınıflama modellerinde ise XGboost en iyi sınıflandırma performansını vermiştir. RandomForest görece düşük kaldı, XGboost en yüksek doğruluk oranını versede Cat boostun ROC AUC değeri en yüksektir (Çizelge 4.15).

Çizelge 4.16. RandomForest - Nested + HPO + kalibrasyon

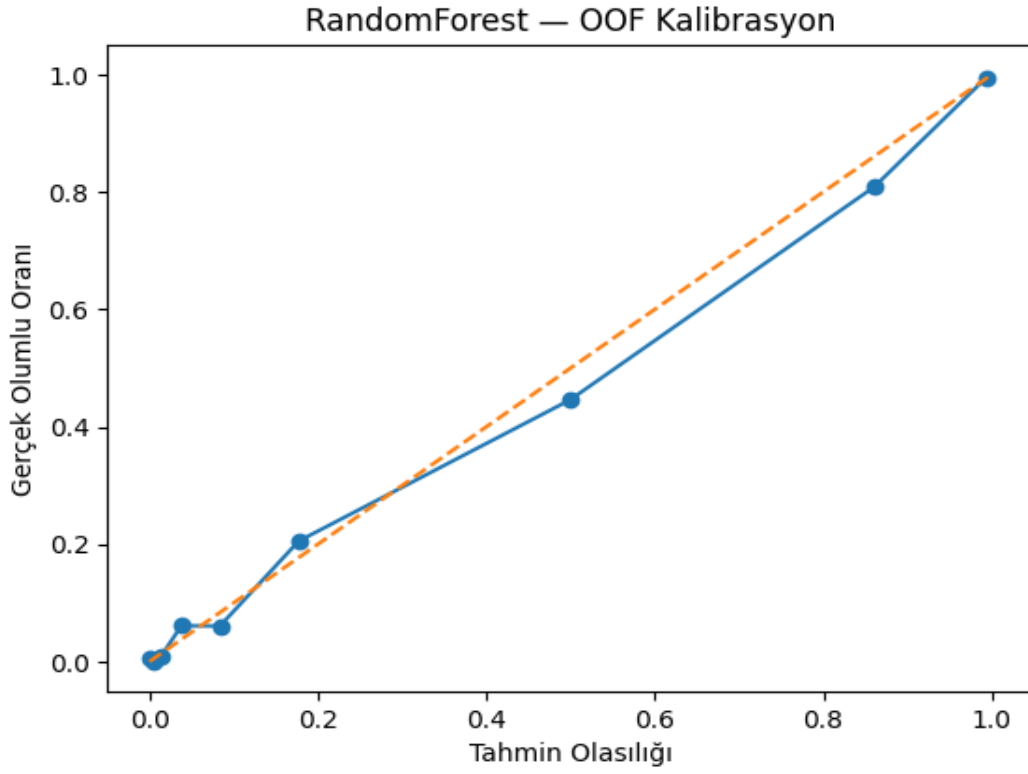
Model	ROC_AUC	ROC_AUC_CI95	PR_AUC	PR_AUC_CI95	F1@0.5	Recall@0.5	Brier
RandomForest	0.96545	[0.954, 0.976]	0.949156	[0.932, 0.963]	0.878274	0.868902	0.065845



Şekil 4.17. RandomForest – OOF ROC



Şekil 4.18. RandomForest – OOF PR



Şekil 4.19. RandomForest – OOF kalibrasyon

Çizelge 4.17. XGBoost - Nested + HPO + kalibrasyon

Model	ROC_AUC	ROC_AUC_CI95	PR_AUC	PR_AUC_CI95	F1@0.5	Recall@0.5	Brier
XGBoost	0.967313	[0.954, 0.978]	0.947806	[0.928, 0.964]	0.873239	0.85061	0.064954

ROC-AUC: 0.967 (95% CI: 0.954 – 0.978)

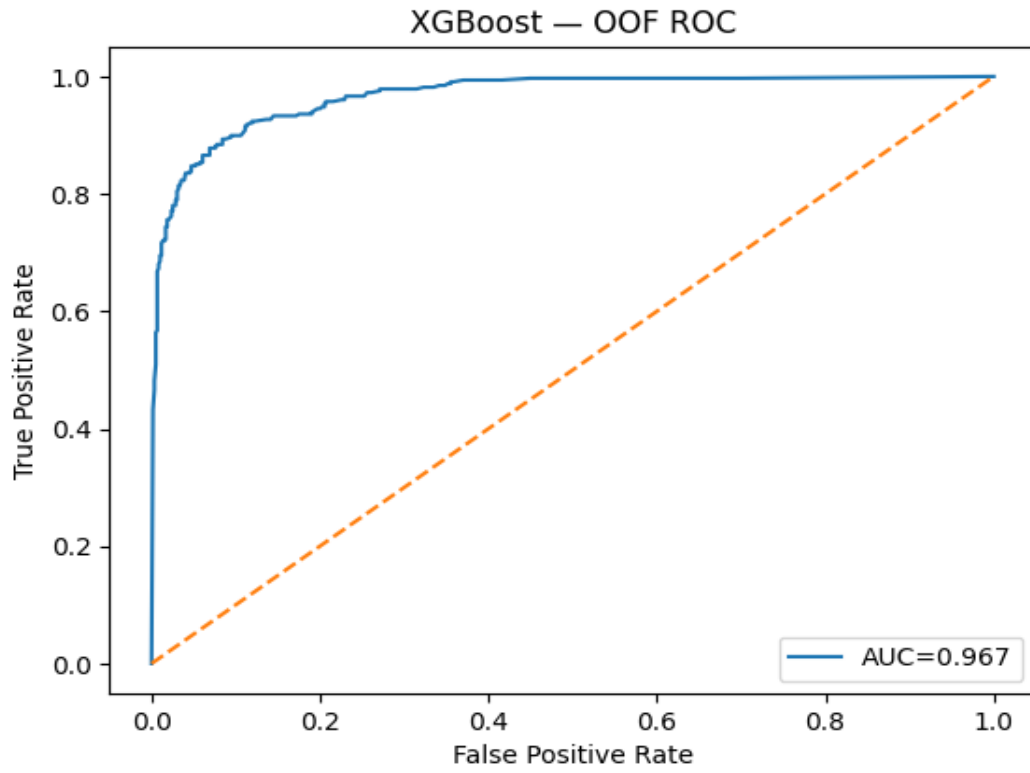
PR-AUC: 0.948 (95% CI: 0.928 – 0.964)

F1-Skoru (0.5 cut-off): 0.873

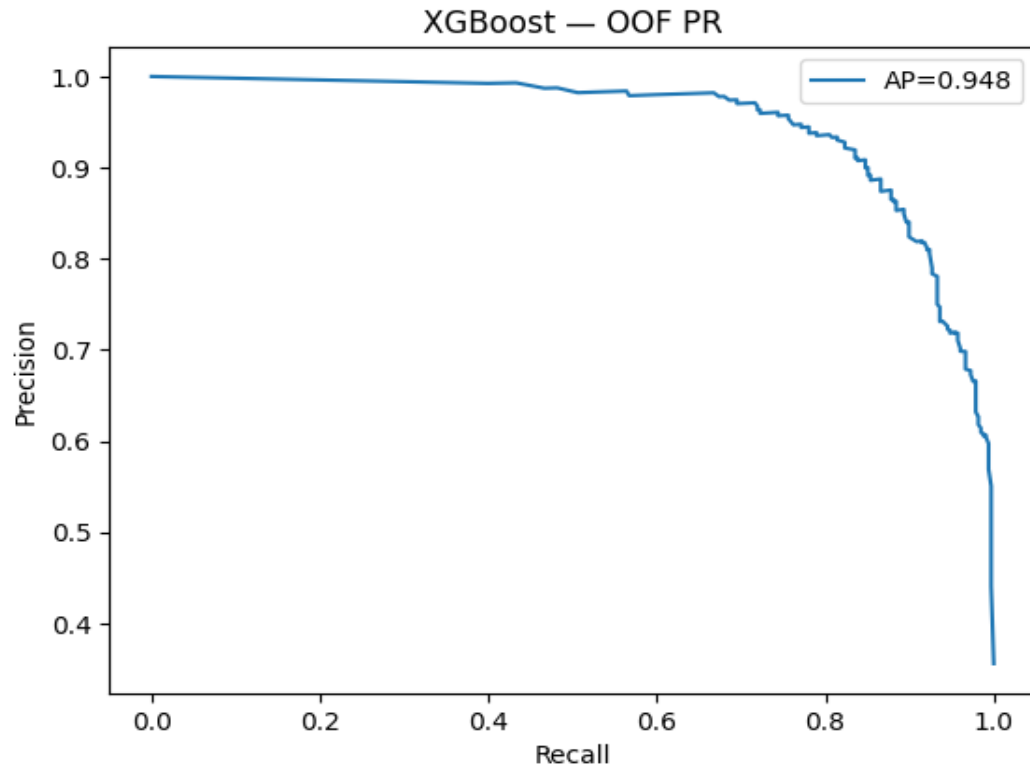
Recall (0.5 cut-off): 0.851

Brier Skoru: 0.065

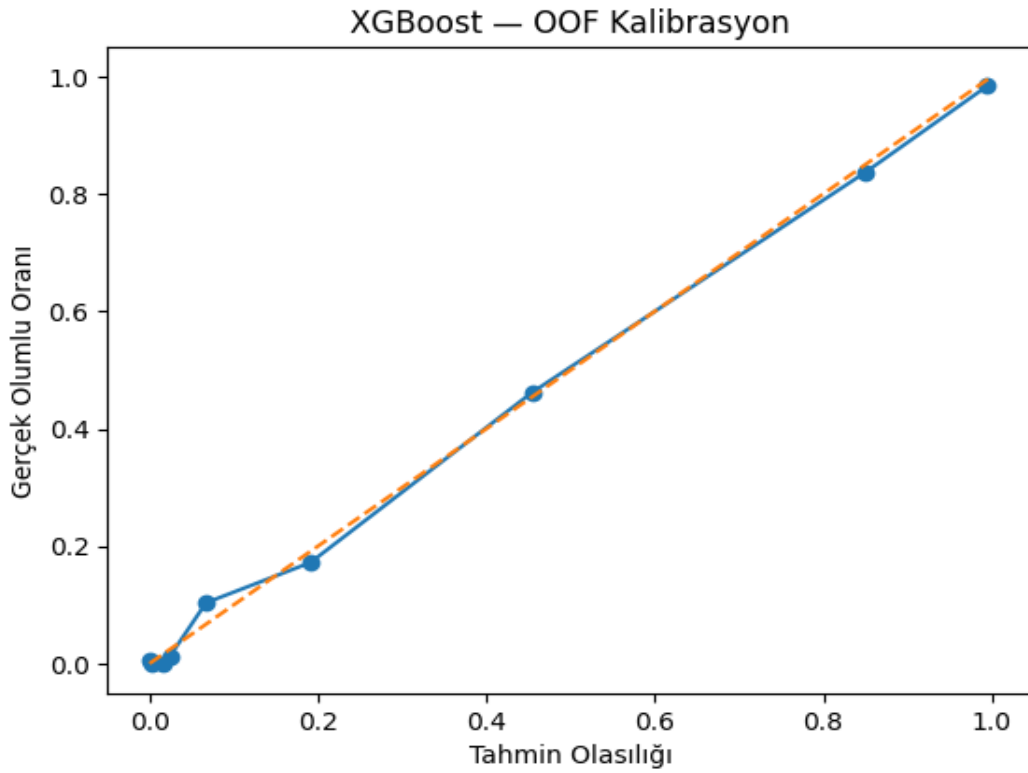
PR-AUC değerinin yüksek oluşu dengesiz veri setlerinde pozitif sınıfı başarılı tahmin etme yeteneğini göstermektedir. RF ile benzer performans göstermiştir. RF pozitif vakaları kaçırmama özelliğinde daha ön plandadır.



Şekil 4.20. XGBoost – OOF ROC



Şekil 4.21. XGBoost – OOF PR



Şekil 4.22. XGBoost – OOF kalibrasyon

Çizelge 4.18. CatBoost - Nested + HPO + kalibrasyon

Model	ROC_AUC	ROC_AUC_CI95	PR_AUC	PR_AUC_CI95	F1@0.5	Recall@0.5	Brier
CatBoost	0.969797	[0.962, 0.979]	0.951893	[0.935, 0.967]	0.876369	0.853659	0.064265

ROC-AUC: 0.970 (95% CI: 0.962 – 0.979)

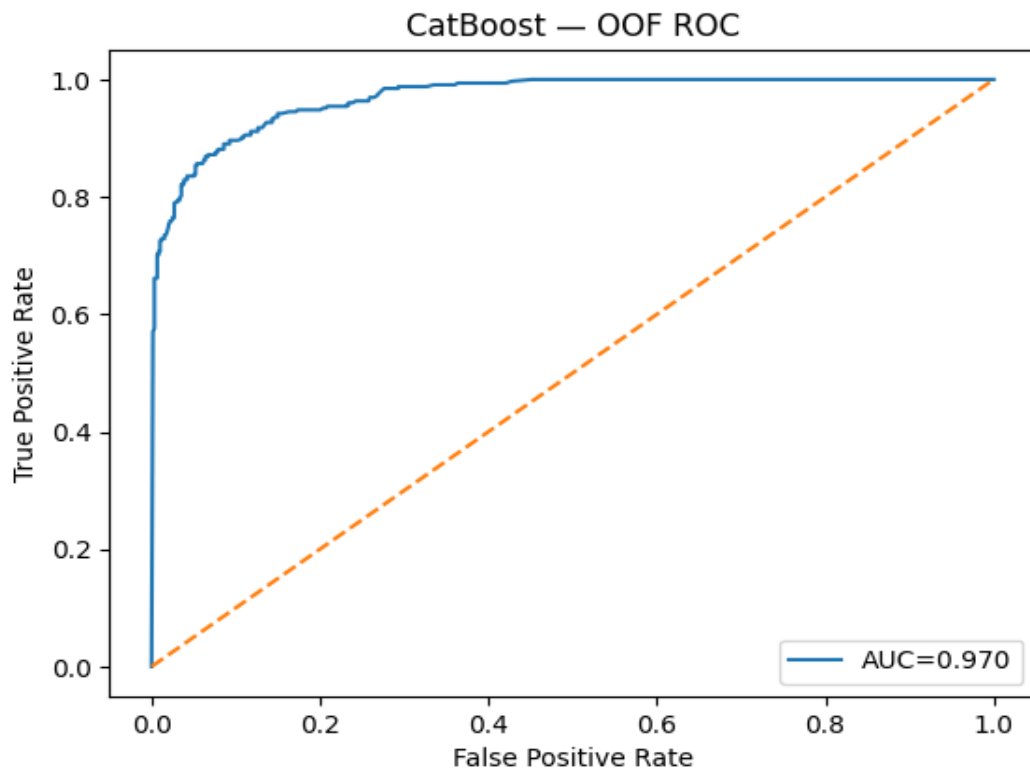
PR-AUC: 0.952 (95% CI: 0.935 – 0.967)

F1-Skoru (0.5 cut-off): 0.876

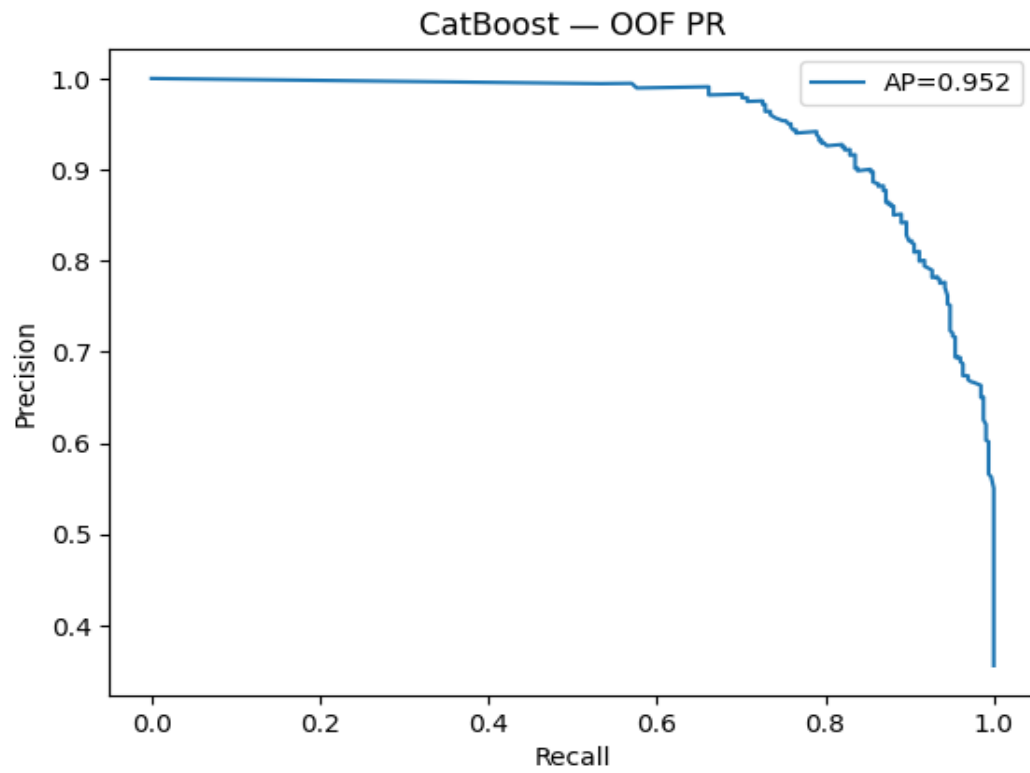
Recall (0.5 cut-off): 0.854

Brier Skoru: 0.064

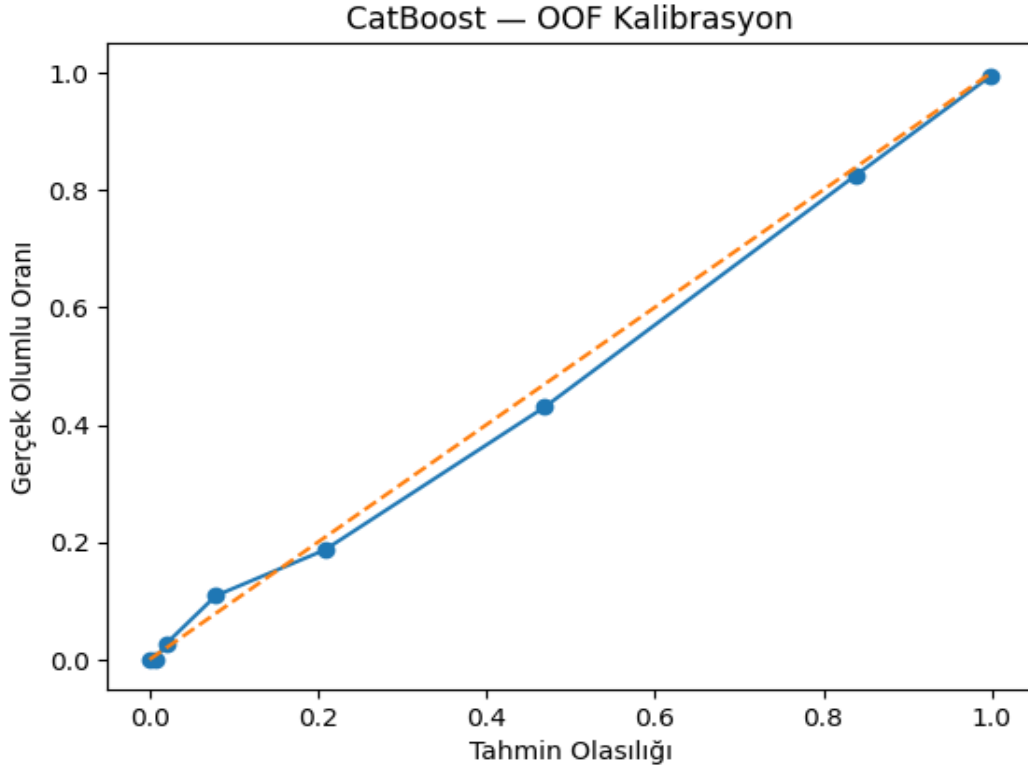
ROC-AUC ve PR-AUC değeri en yüksek olan ve pozitif negatif ayrımını en iyi yapan modeldir. Yüksek AHI grubu için sınıflandırmada üstün bir tahmin gücü ortaya koymuştur. En düşük brier skoruna sahiptir ve olasılık kalibrasyonunun en güvenilir şekilde bu modelde yapıldığı ortaya konmuştur. CatBoost en güvenilir ve en dengeli modeldir.



Şekil 4.23. CatBoost – OOF ROC



Şekil 4.24. CatBoost – OOF PR



Şekil 4.25. CatBoost – OOF kalibrasyon

Çizelge 4.19. Sınıflandırma modelleri toplu özet

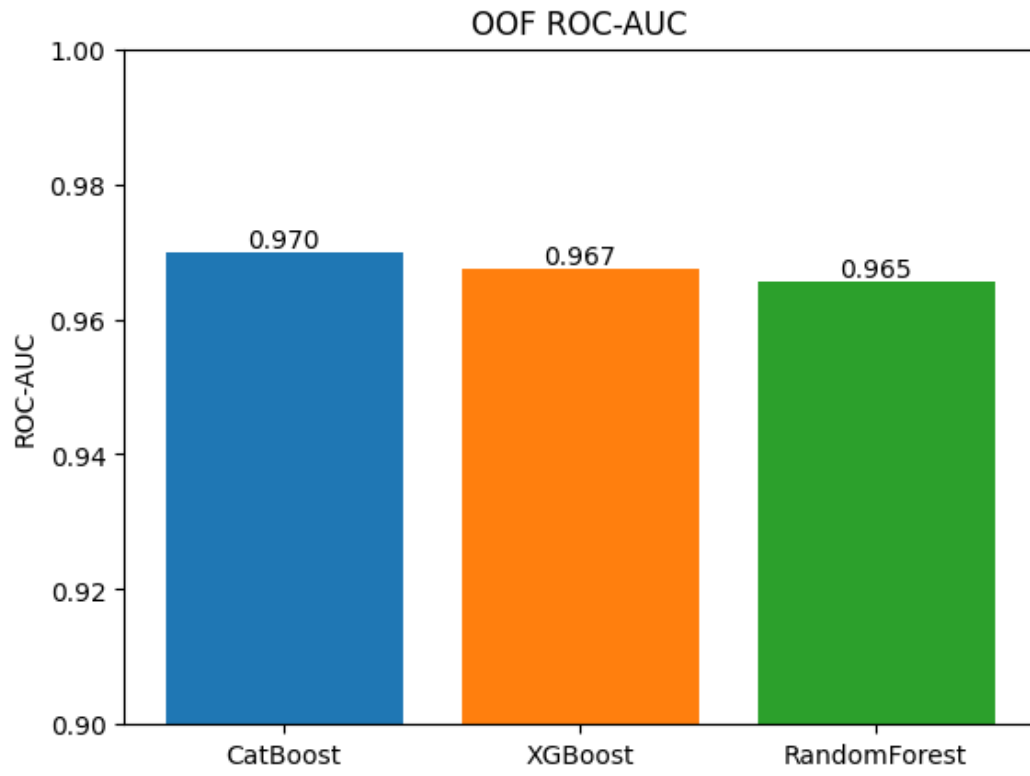
Model	ROC_AUC	PR_AUC	F1@0.5	Recall@0.5	Brier
CatBoost	0.969797	0.951893	0.876369	0.853659	0.064265
XGBoost	0.967313	0.947806	0.873239	0.850610	0.064954
RandomForest	0.965450	0.949156	0.878274	0.868902	0.065845

ROC-AUC: CatBoost en yüksek değerdedir. Sınıflar arası ayrıştırma gücünde liderdir.

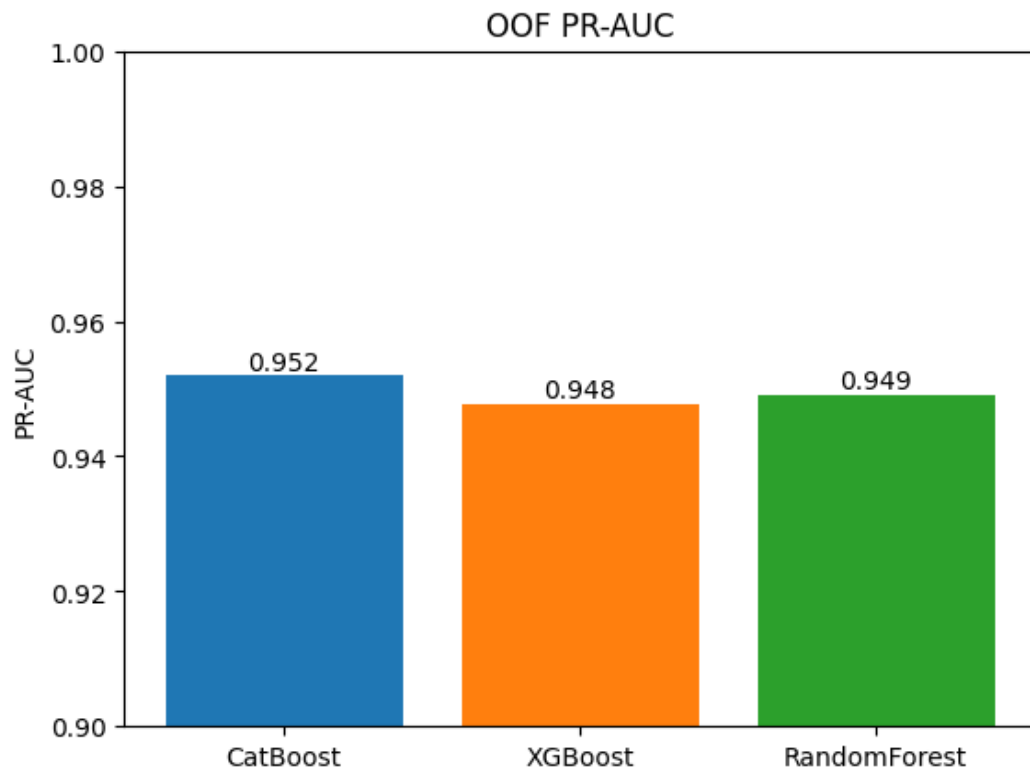
PR-AUC: CatBoost en yüksek değerdir. Pozitif sınıfları algılamada en yüksektir.

F1 ve Recall: RF en yüksek değerleri sağlamıştır. Duyarlılığı en yüksek modeldir.

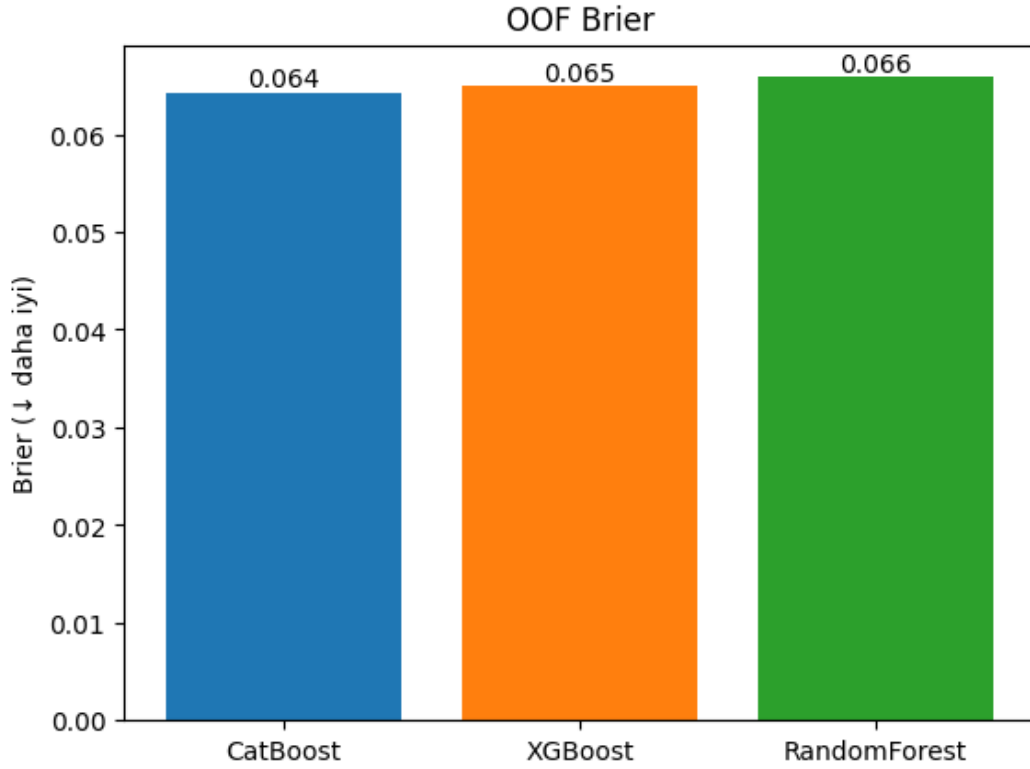
Brier Skoru: CatBoost en düşük değere sahiptir. En iyi kalibre edilmiş modeldir.



Şekil 4.26. OOF ROC-AUC



Şekil 4.27. OOF PR-AUC



Şekil 4.28. OOF Brier

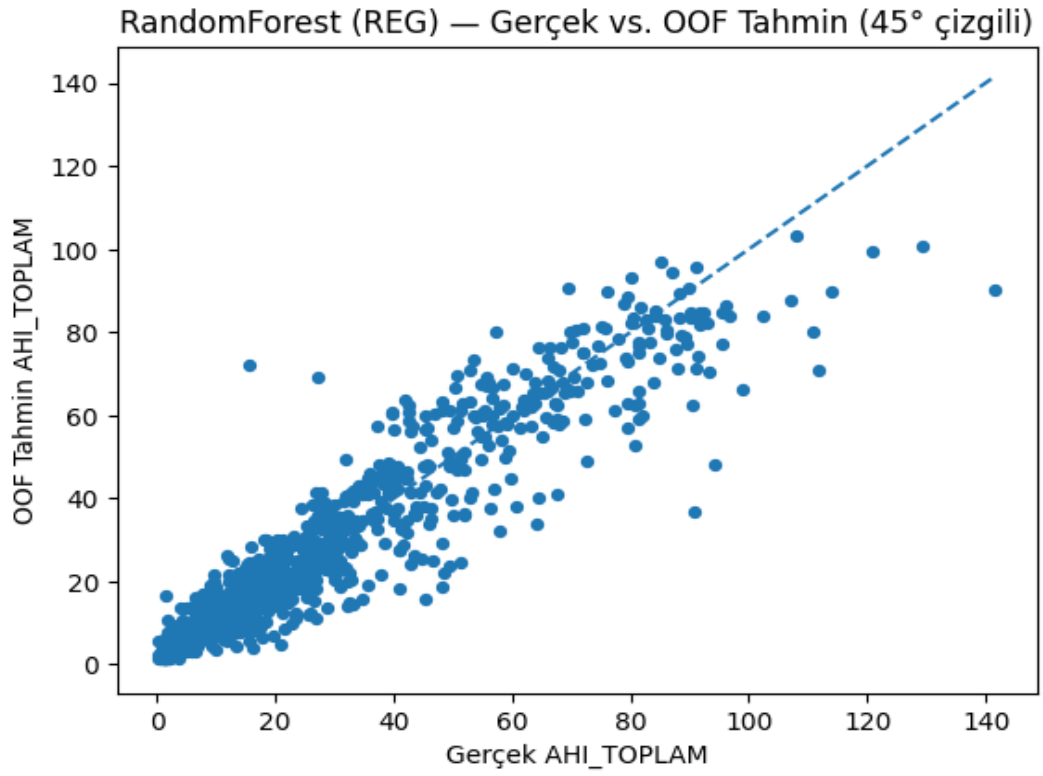
Çizelge 4.20. RandomForest Regressor (Nested CV + HPO) - Fold Metrikleri

Model	RMSE	RMSE_CI95	MAE	MAE_CI95	R2	R2_CI95
RandomForest (REG)	8.846676	[8.051, 9.725]	5.957281	[5.563, 6.397]	0.882055	[0.861, 0.901]

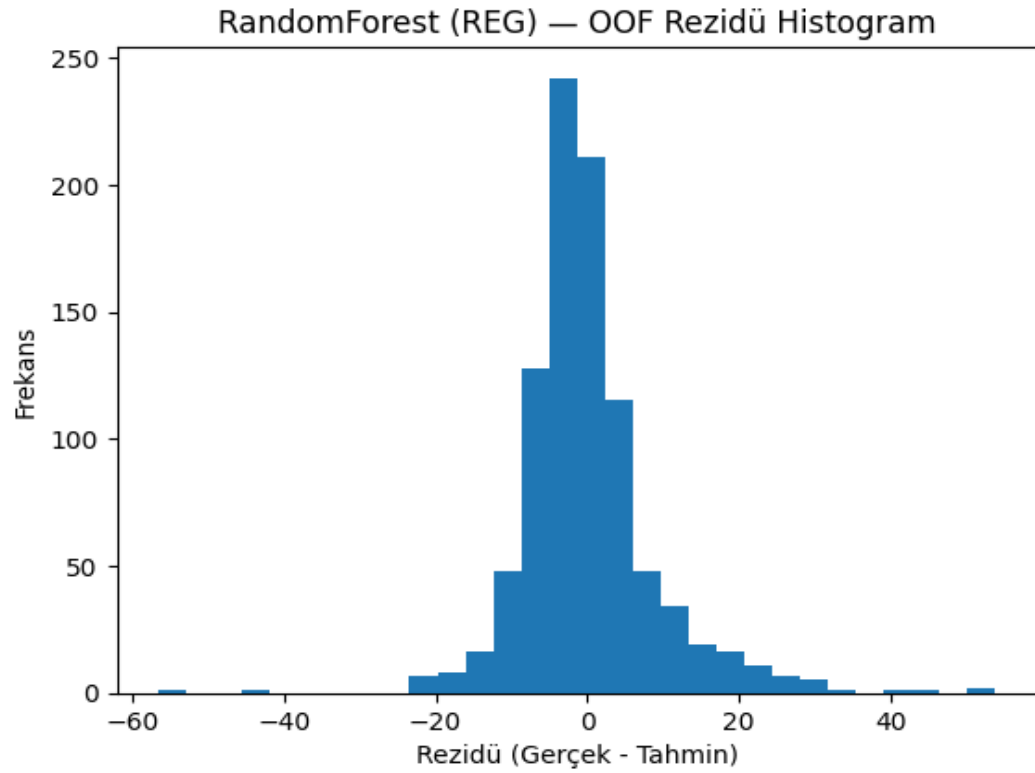
Model RMSE'si 8.85, MAE si 5.96 bulunmuştur. Tahmin edilen AHİ değerleri ile gerçek değerler arasındaki farkın klinik açıdan kabul edilebilir düzeydedir.

R² (Determinasyon Katsayısı): 0.882 ile model varyansının %88'ini açıklamaktadır. Güçlü öngörü kapasitesine sahiptir.

Güven Aralığı: RMSE ve MAE güven aralığı olup tahmin modelinin stabil olduğunu göstermiştir. Hiperparametre ile optimize edildiğinde AHİ değerlerini başarılı bir şekilde öngörmektedir.



Şekil 4.29. RandomForest (REG) – gerçek vs OOF tahmin (45 ° çizgili)



Şekil 4.30. RandomForest (REG) – OOF rezidü histogram

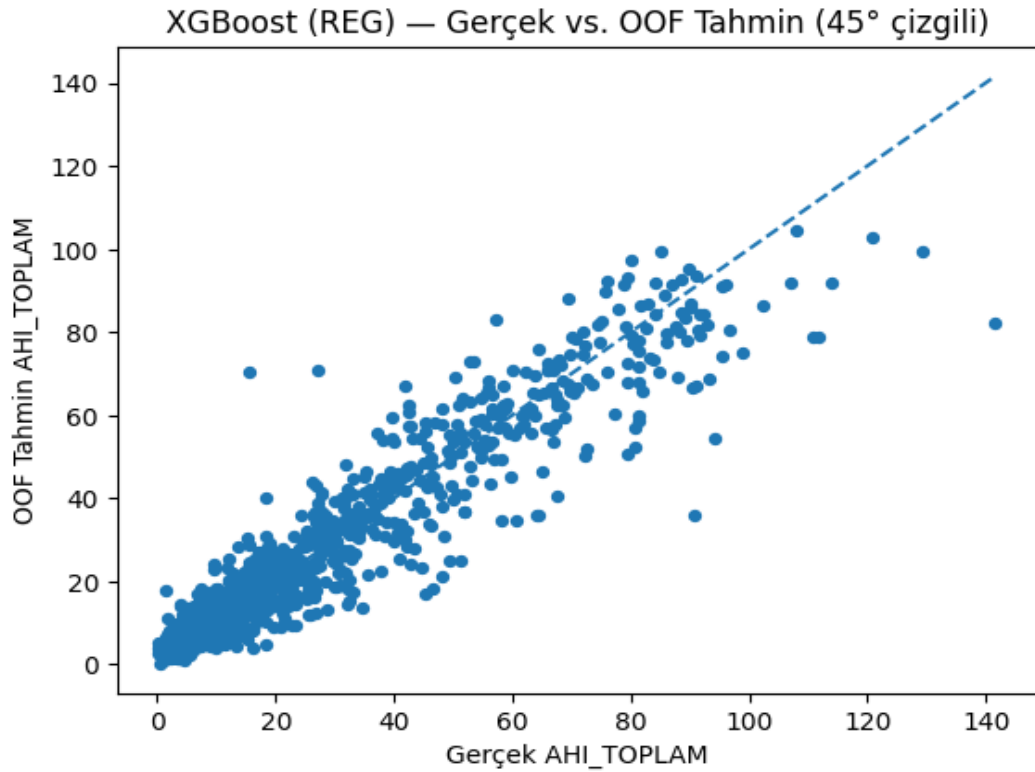
Çizelge 4.21. XGBoost Regressor (Nested CV + HPO) - Fold Metrikleri

Model	RMSE	RMSE_CI95	MAE	MAE_CI95	R2	R2_CI95
XGBoost (REG)	8.790856	[7.958, 9.711]	5.918709	[5.548, 6.377]	0.883539	[0.860, 0.903]

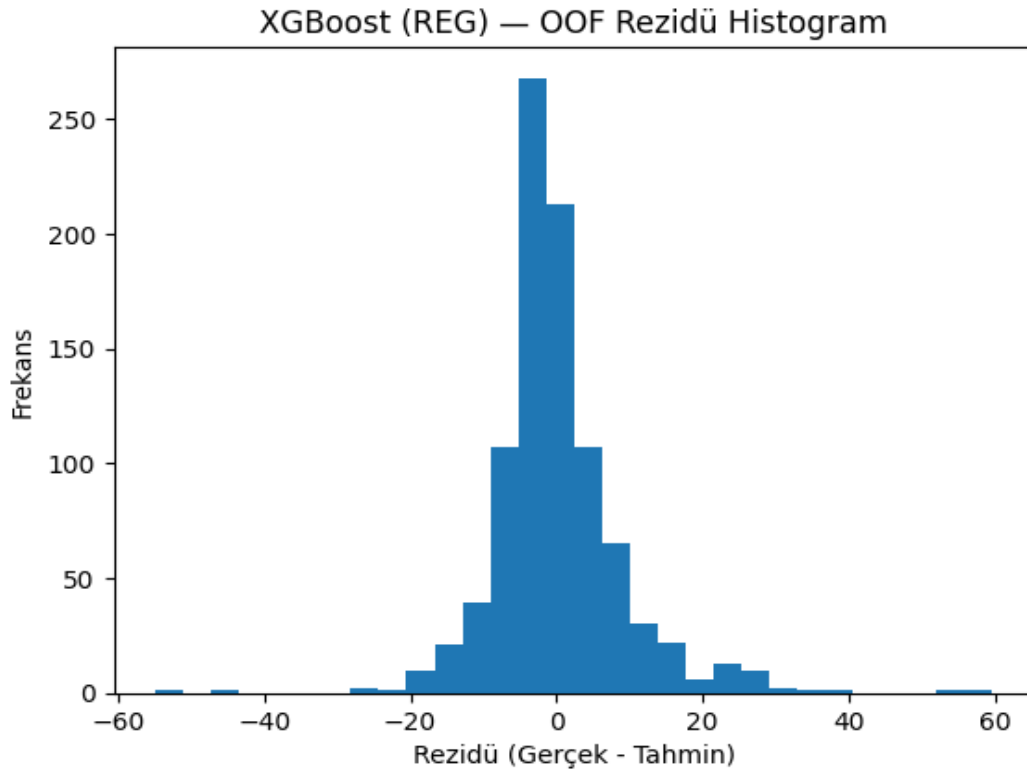
Model RMSE'si 8.79, MAE'si 5.92 bulunmuştur. Hata payı oldukça düşüktür.

R² (Determinasyon Katsayısı): 0.884 ile model varyansının %88'ini açıklamaktadır. Güçlü öngörü kapasitesine sahiptir.

Güven Aralığı: RMSE ve MAE güven aralığı olup tahmin modelinin stabil olduğunu göstermiştir. Hiperparametre ile optimize edildiğinde AHI değerlerini başarılı bir şekilde öngörmektedir.



Şekil 4.31. XGBoost (REG) – gerçek vs OOF tahmin (45 ° çizgili)



Şekil 4.32. XGBoost (REG) – OOF rezidü histogram

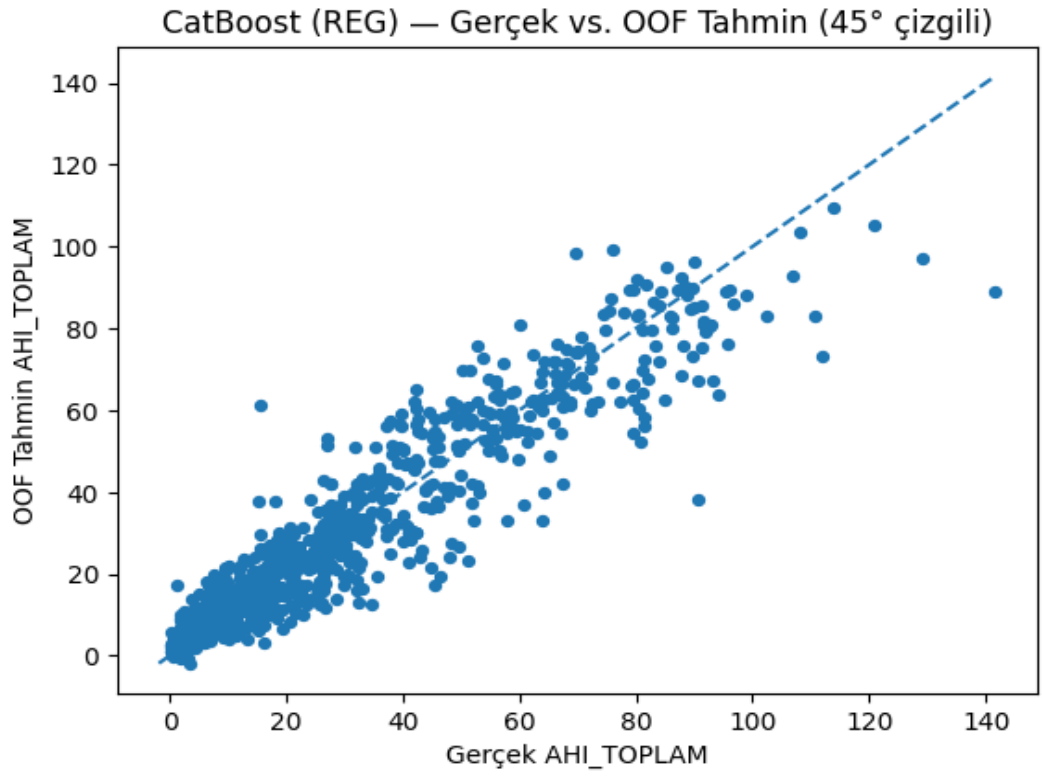
Çizelge 4.22. CatBoost Regressor (Nested CV + HPO) - Fold Metrikleri

Model	RMSE	RMSE_CI95	MAE	MAE_CI95	R2	R2_CI95
CatBoost (REG)	8.652123	[7.927, 9.473]	5.942602	[5.543, 6.360]	0.887186	[0.868, 0.904]

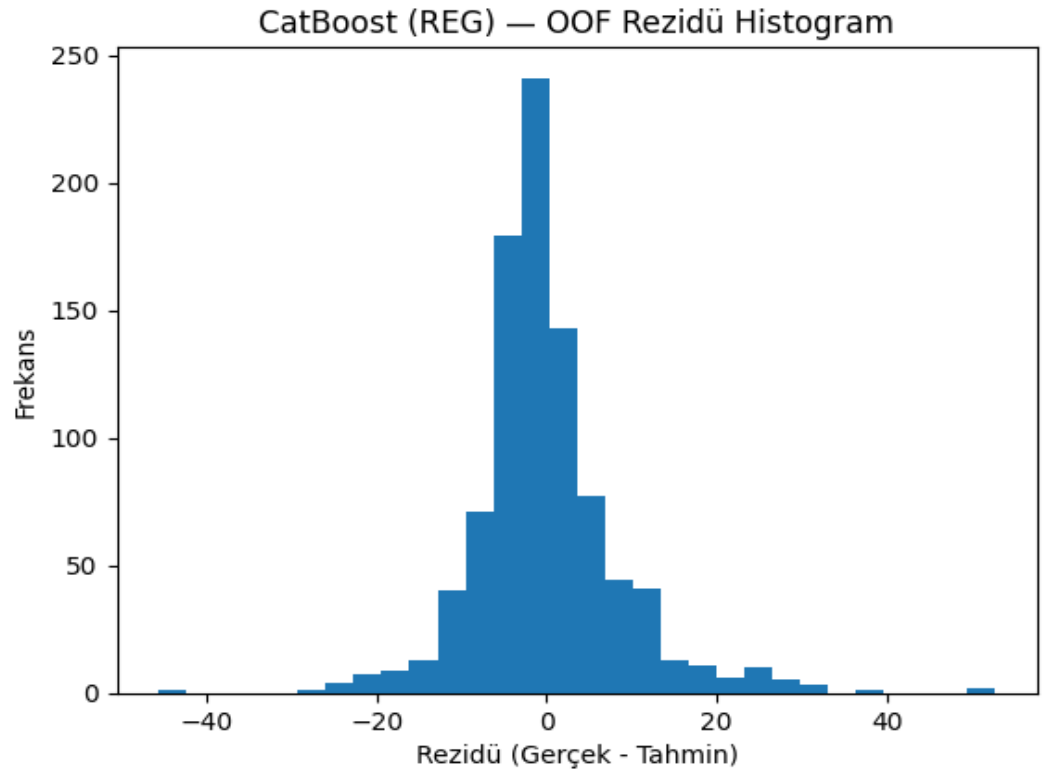
Model RMSE’si 8.65, MAE’si 5.94 bulunmuştur. 3 model arasındaki en düşük hata payına sahiptir..

R² (Determinasyon Katsayısı): 0.887 ile model varyansının %89’ini açıklamaktadır. Güçlü öngörü kapasitesin de, RandomForest ve XGBoost modellerinden daha yüksek orana sahiptir.

Güven Aralığı: RMSE ve MAE güven aralığı olup tahmin modelinin stabil olduğunu göstermiştir. Hiperparametre ile optimize edildiğinde AHİ değerlerini başarılı bir şekilde öngörmektedir. 3 model arasındaki en dengeli ve güçlü performansı göstermiştir.



Şekil 4.33. CatBoost (REG) – gerçek vs OOF tahmin (45 ° çizgili)



Şekil 4.34. CatBoost (REG) – OOF rezidü histogram

Çizelge 4.23. Regresyon modelleri toplu özet

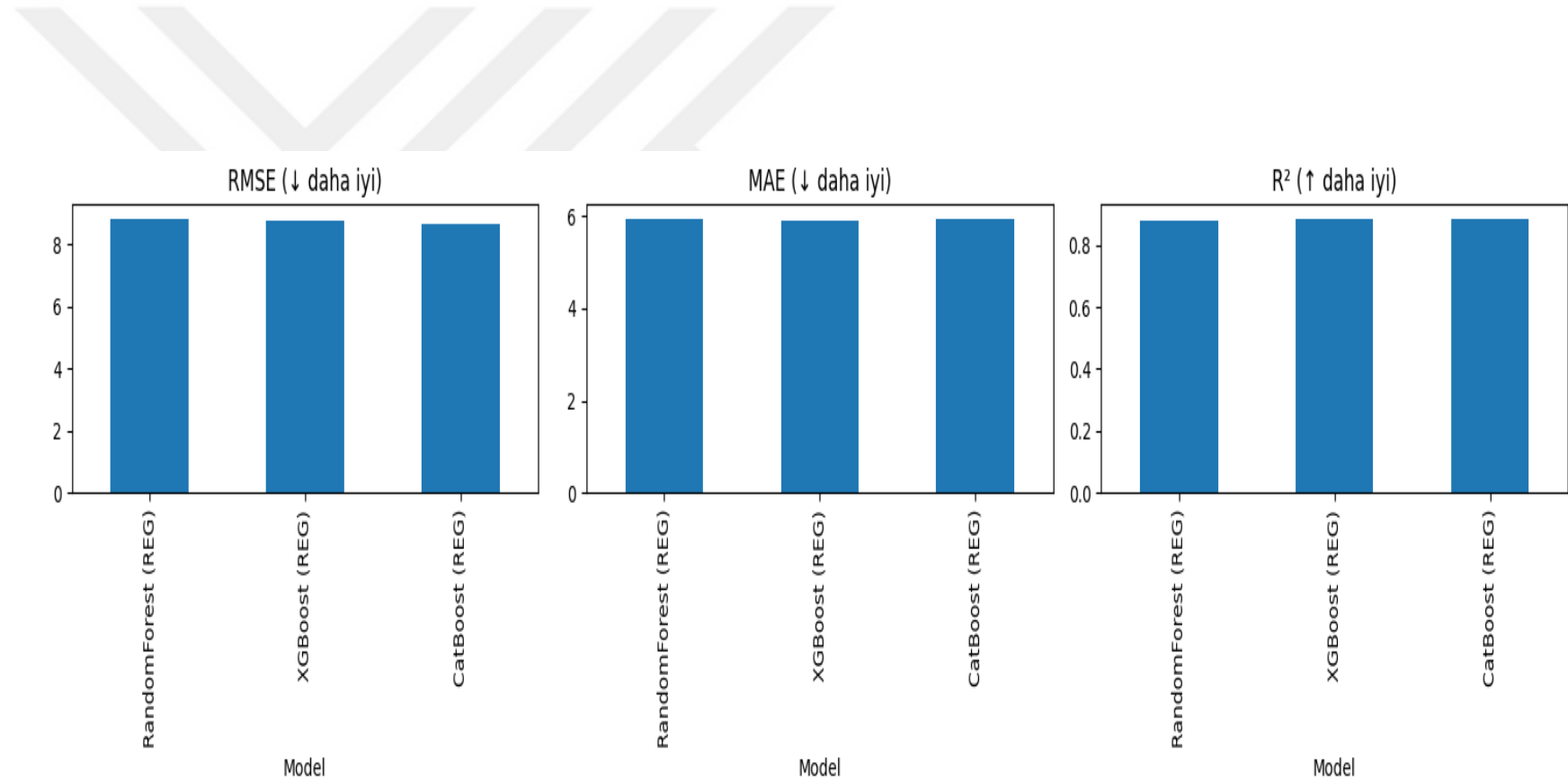
Model	RMSE	MAE	R2
CatBoost (REG)	8.652123	5.942602	0.887186
XGBoost (REG)	8.790856	5.918709	0.883539
RandomForest (REG)	8.846676	5.957281	0.882055

RMSE: En düşük hata payı CatBoost 8.65

MAE: En düşük mutlak hata XGBoost 5.92

R²: En yüksek açıklanan varyans CatBoost 0.887

En dengeli ve en güçlü model CatBoost modeli olarak öne çıkmıştır.



Şekil 4.35. Regresyon modelleri

Çizelge 4.24. DeLong (AUC farkı) – sınıflama modelleri kıyaslama testleri

Pair	ΔAUC (A-B)	AUC_A	AUC_B	z	p	p_holm
RandomForest vs CatBoost	0.466022	0.970231	0.504209	89.415919	0.000000	0.000000
XGBoost vs CatBoost	0.462147	0.966356	0.504209	80.250945	0.000000	0.000000
RandomForest vs XGBoost	0.003875	0.970231	0.966356	1.529666	0.126099	0.126099

DeLong: CatBoost ve XGBoost arasında anlamlı fark bulunmuştur. CatBoost ile RandomForest arasında anlamlı fark vardır.

Çizelge 4.25. McNemar @0.5 – sınıflama modelleri kıyaslama testleri

Pair	b (A doğru,B yanlış)	c (A yanlış,B doğru)	p	p_holm
RandomForest vs CatBoost	552	45	2.861612e-95	8.584837e-95
XGBoost vs CatBoost	556	47	4.502038e-95	9.004077e-95
RandomForest vs XGBoost	19	17	8.676323e-01	8.676323e-01

McNemar Testi: CatBoost ve XGBoost arasında yanlış sınıflandırma dağılımlarını arasında anlamlı fark vardır. CatBoost ve RandomForest arasında da anlamlı fark vardır.

Çizelge 4.26. Brier farkı (Wilcoxon, 5-fold proxy) – sınıflama modelleri kıyaslama testleri

Pair	Brier_A	Brier_B	$\Delta Brier$ (A-B)	p	p_holm
RandomForest vs CatBoost	0.063164	0.639437	-0.576273	0.0625	0.1875
XGBoost vs CatBoost	0.072685	0.639437	-0.566752	0.0625	0.1250
RandomForest vs XGBoost	0.063164	0.072685	-0.009521	0.1250	0.1250

Brier Farkı: Kalibrasyon hatası ölçümünde skor farkları istatistiksel olarak anlamlı değil.

Çizelge 4.27. PR-AUC farkı (bootstrap) – sınıflama modelleri kıyaslama testleri

Pair	Δ PR-AUC (A-B)	CI95	p	p_holm
RandomForest vs CatBoost	0.598589	[0.583, 0.612]	0.00	0.00
XGBoost vs CatBoost	0.593188	[0.577, 0.608]	0.00	0.00
RandomForest vs XGBoost	0.005402	[-0.002, 0.014]	0.16	0.16

PR-AUC farkı: CatBoost diğer iki modelden anlamlı şekilde yüksektir, diğer iki model arasında anlamlı fark yoktur.

Çizelge 4.28. Kalibrasyon (bootstrap; intercept & slope) – sınıflama modelleri kıyaslama testleri

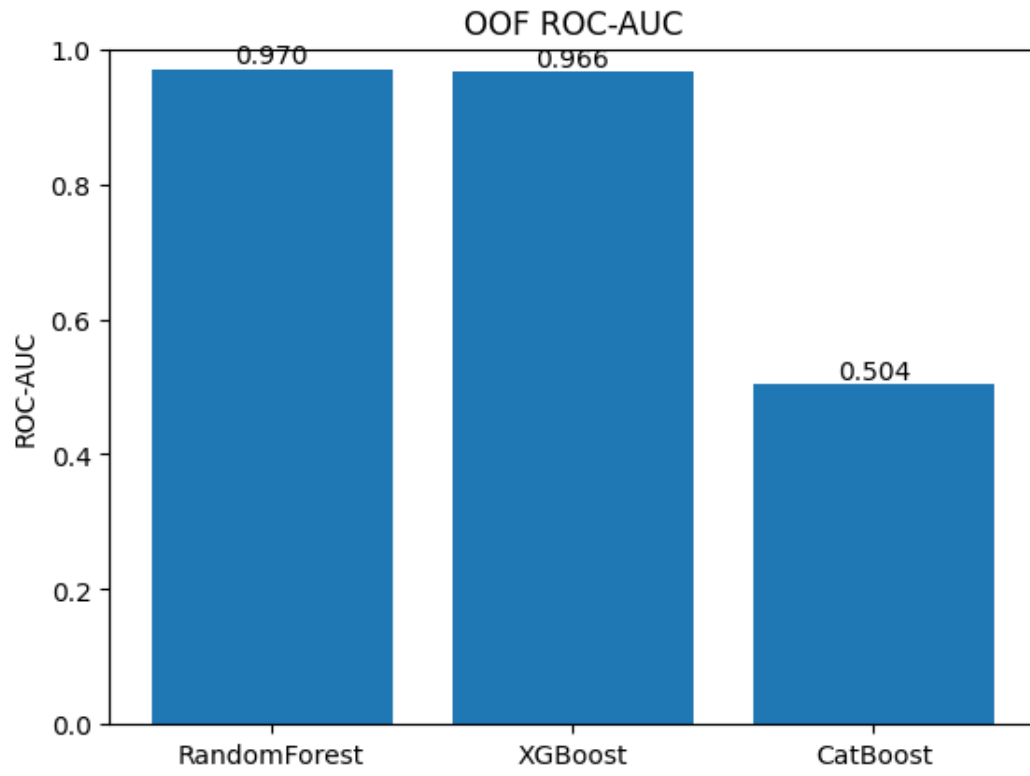
Pair	Intercept_mean	Intercept_CI95	Slope_mean	Slope_CI95
RandomForest vs CatBoost	-0.002476	[-0.257, 0.255]	1.343962	[1.184, 1.521]
XGBoost vs CatBoost	-0.140613	[-0.396, 0.122]	0.559170	[0.506, 0.625]
RandomForest vs XGBoost	-6.519136	[-11.932, -3.255]	0.429687	[0.196, 0.825]

Kalibrasyon (intercept & slope): RandomForest en dengeli model olarak görünmektedir.

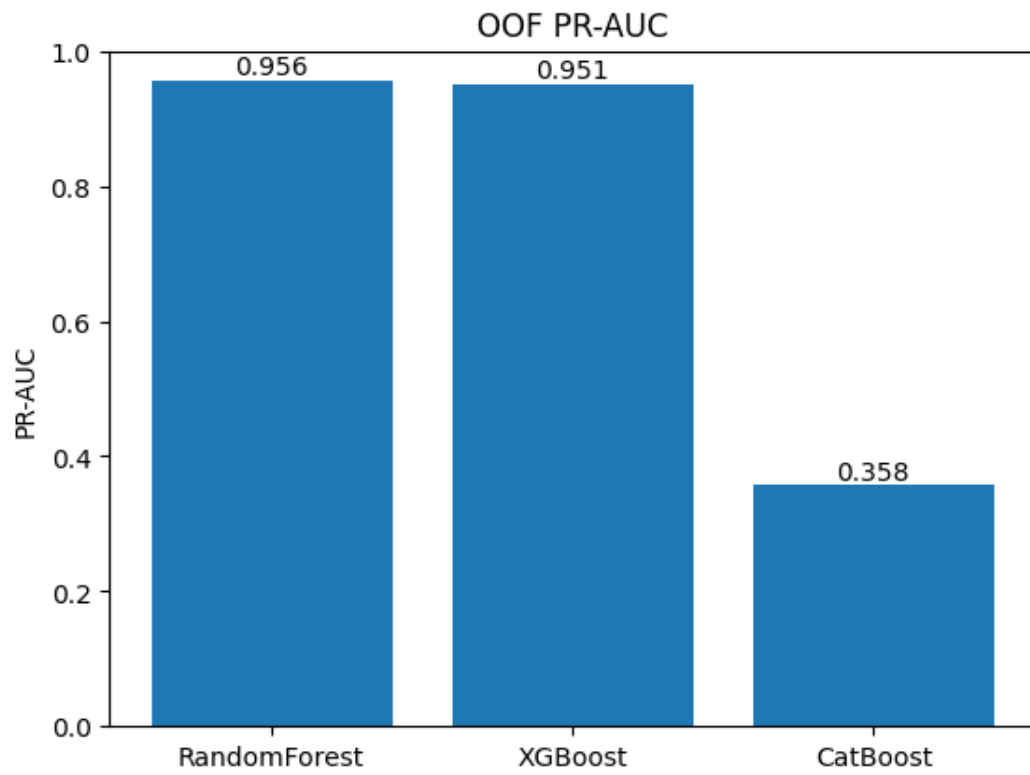
Çizelge 4.29. Kalibrasyon farkları (bootstrap) – sınıflama modelleri kıyaslama testleri

Pair	Δ Intercept (A-B)	Δ Intercept_CI95	p_intercept	Δ Slope (A-B)	Δ Slope_CI95	p_slope	p_holm_intercept	p_holm_slope
RandomForest vs CatBoost	0.142170	[0.044, 0.245]	0.0050	0.786335	[0.667, 0.925]	0.0000	0.0100	0.0000
XGBoost vs CatBoost	6.544440	[3.267, 11.990]	0.0025	0.914578	[0.477, 1.244]	0.0000	0.0075	0.0000
RandomForest vs XGBoost	6.402271	[3.138, 11.801]	0.0150	0.128243	[-0.255, 0.380]	0.4475	0.0150	0.4475

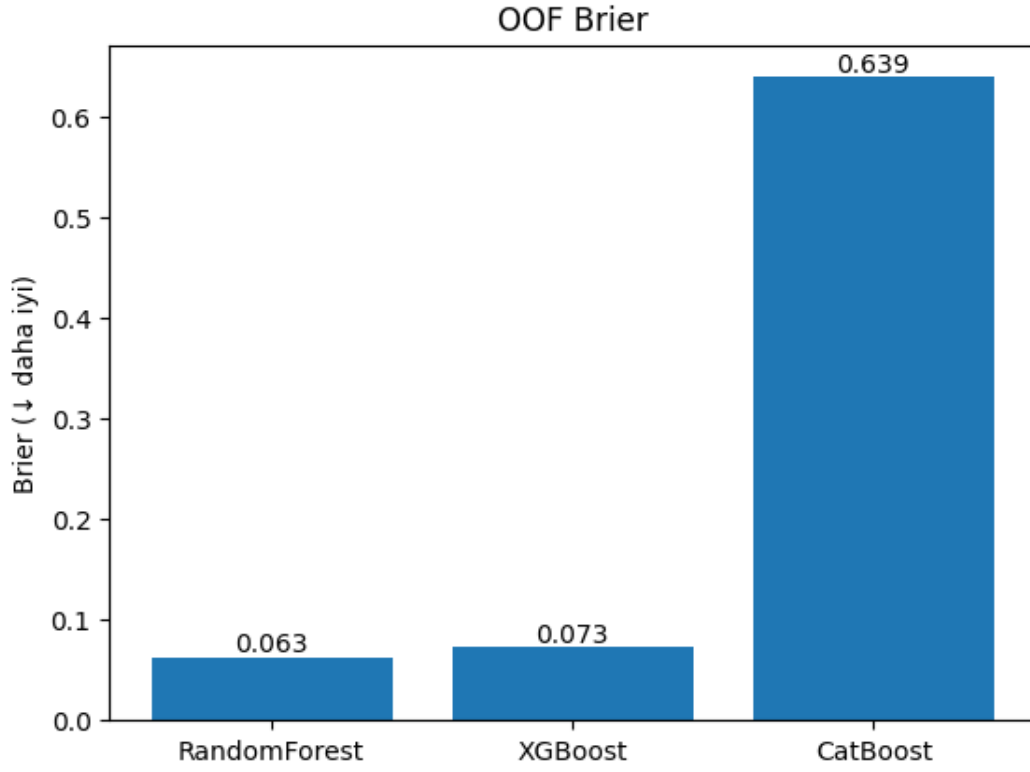
Kalibrasyon Farkları: RandomForest kalibrasyon eğrisi intercept düzeltmesinde aha iyi ama 3 modelinde Slope farkları anlamlı değil.



Şekil 4.36. OOF ROC-AUC



Şekil 4.37. OOF PR-AUC



Şekil 4.38. OOF Brier

Çizelge 4.30. Regresyon modelleri: OOF özet (RMSE, MAE, R^2) + bootstrap %95 GA – regresyon modelleri kıyaslama testleri

Model	RMSE	MAE	R2	RMSE_CI95	MAE_CI95	R2_CI95
CatBoost	8.652123	5.942602	0.887186	[7.933, 9.455]	[5.565, 6.356]	[0.867, 0.903]
XGBoost	8.790856	5.918709	0.883539	[7.944, 9.685]	[5.519, 6.338]	[0.860, 0.903]
RandomForest	8.846676	5.957281	0.882055	[8.042, 9.716]	[5.571, 6.398]	[0.858, 0.900]

Catboost regresyon en düşük RMSE ve en yüksek R^2 değeri ile öne çıkmaktadır.

Çizelge 4.31. Pairwise Wilcoxon (RMSE; 5-fold eşleştirilmiş) – regresyon modelleri kıyaslama testleri

Pair	RMSE_A	RMSE_B	Δ RMSE (A-B)	p	p_holm
XGBoost vs CatBoost	8.750052	8.621439	0.128613	0.3125	0.9375
RandomForest vs CatBoost	8.802496	8.621439	0.181057	0.4375	0.8750
RandomForest vs XGBoost	8.802496	8.750052	0.052444	1.0000	1.0000

Wilcoxon RMSE 3 model arasında hata oranı açısından belirgin üstünlük yok.

Çizelge 4.32. Pairwise Wilcoxon (MAE; 5-fold eşleştirilmiş) – regresyon modelleri kıyaslama testleri

Pair	MAE_A	MAE_B	Δ MAE (A-B)	p	p_holm
XGBoost vs CatBoost	5.917835	5.941764	-0.023929	0.3125	0.9375
RandomForest vs CatBoost	5.956361	5.917835	0.038525	0.8125	1.0000
RandomForest vs XGBoost	5.956361	5.941764	0.014596	1.0000	1.0000

Wilcoxon MAE 3 model arası fark yoktur. 3 model de benzer ortalama hata göstermektedir.

Çizelge 4.33. Δ RMSE bootstrap (eşleştirilmiş; GA + p; Holm) – regresyon modelleri kıyaslama testleri

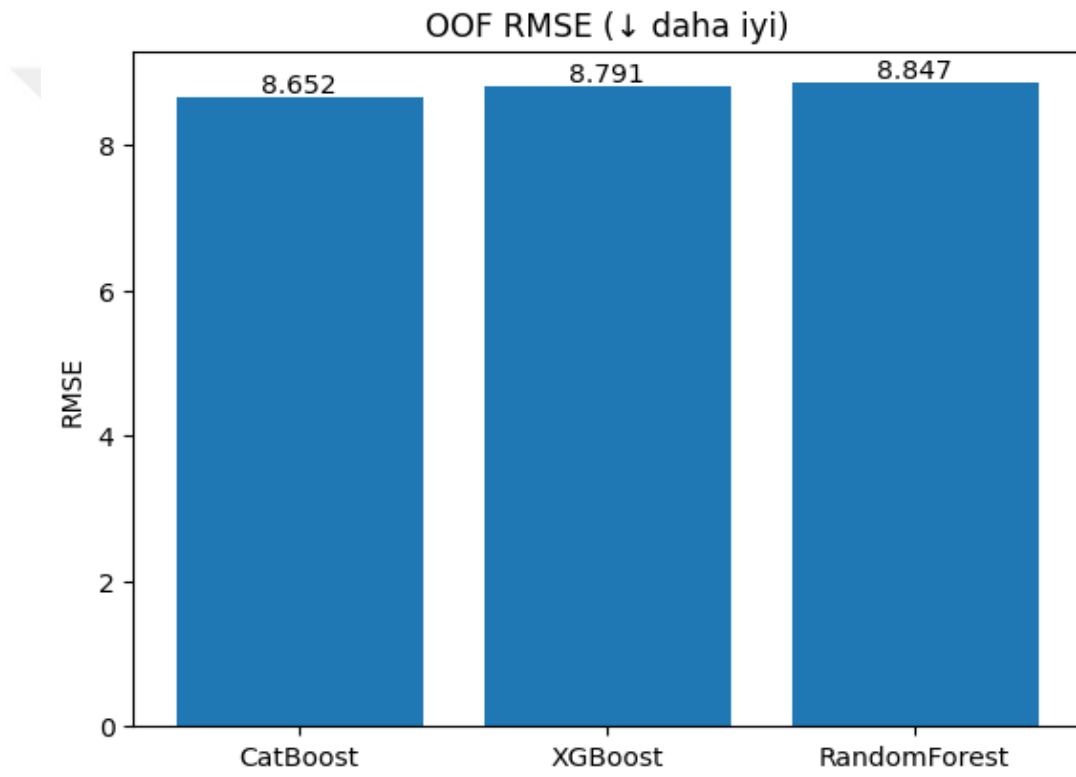
Pair	Δ RMSE (A-B)	CI95	p	p_holm
RandomForest vs CatBoost	0.194553	[-0.155, 0.546]	0.285	0.855
XGBoost vs CatBoost	0.138733	[-0.162, 0.454]	0.406	0.812
RandomForest vs XGBoost	0.055820	[-0.193, 0.313]	0.654	0.654

Bootstrap RMSE 3 model için anlamlı fark yoktur.

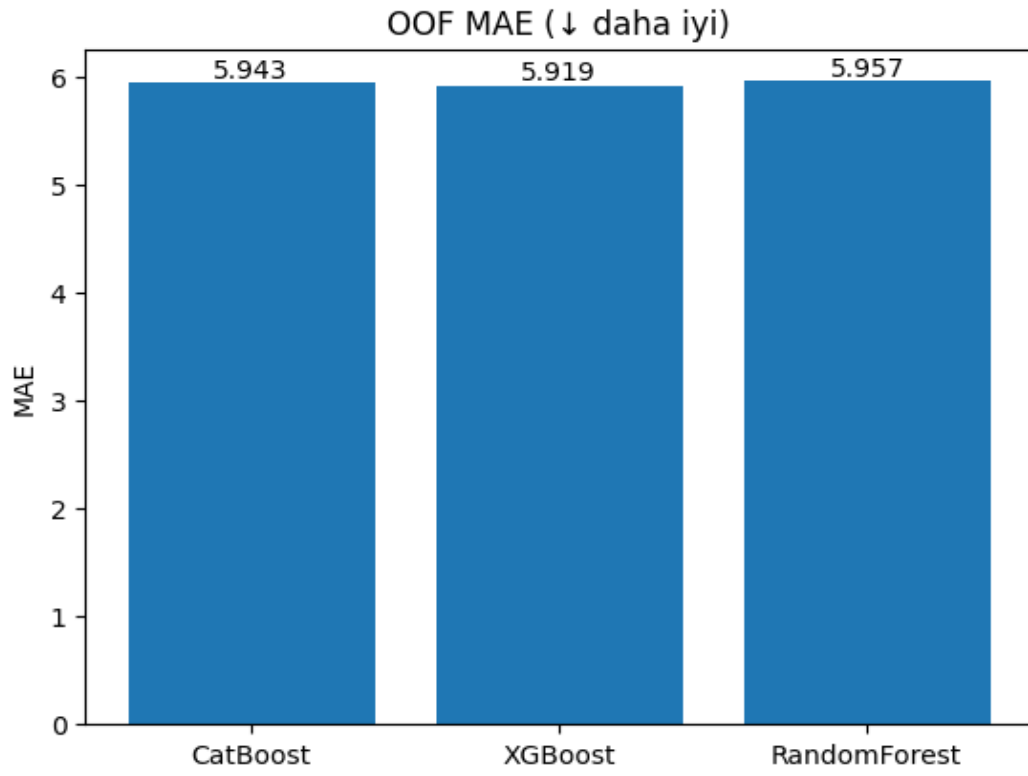
Çizelge 4.34. Δ MAE bootstrap (eşleştirilmiş; GA + p; Holm) – regresyon modelleri kıyaslama testleri

Pair	Δ MAE (A-B)	CI95	p	p_holm
RandomForest vs CatBoost	0.038572	[-0.125, 0.204]	0.663	1.000
XGBoost vs CatBoost	-0.023892	[-0.199, 0.160]	0.788	1.000
RandomForest vs XGBoost	0.014679	[-0.193, 0.223]	0.881	0.881

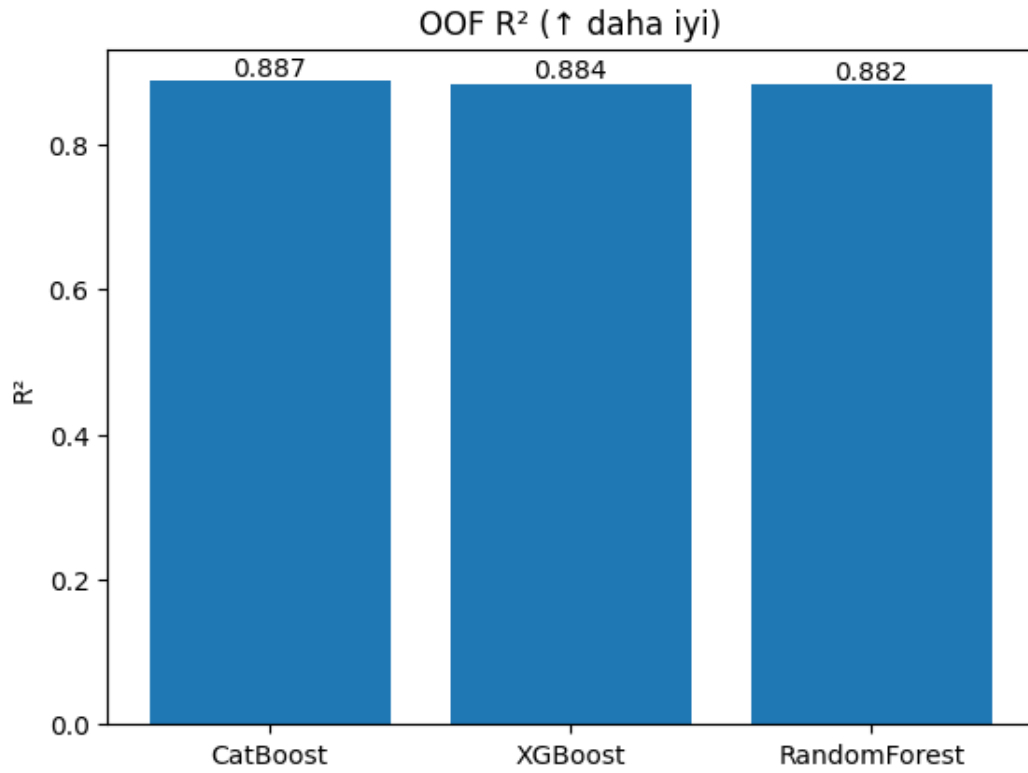
Bootstrap MAE tüm model çiftleri için anlamlı fark bulunamamıştır.



Şekil 4.39. OOF RMSE (↓ daha iyi)



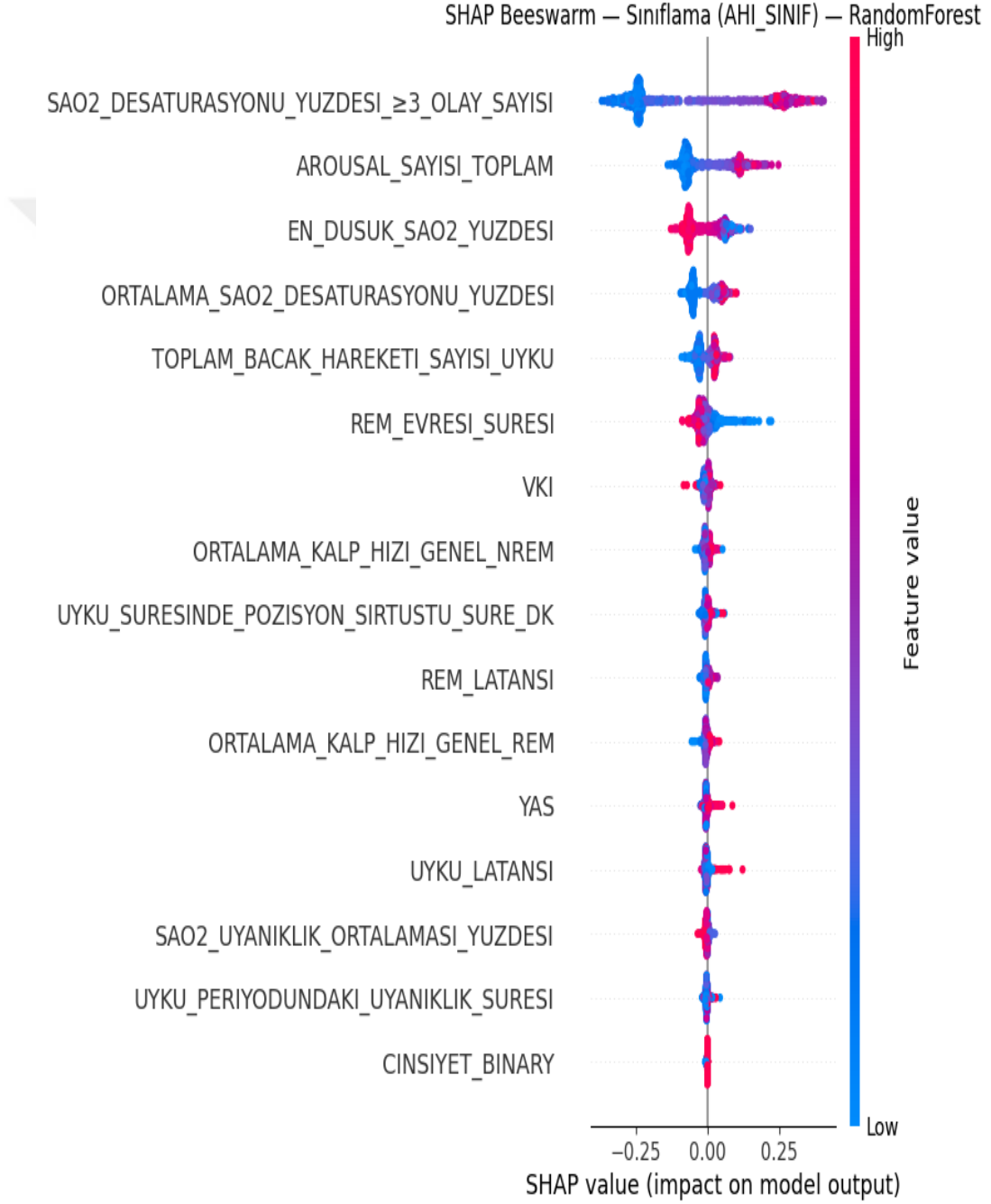
Şekil 4.40. OOF MAE (↓ daha iyi)



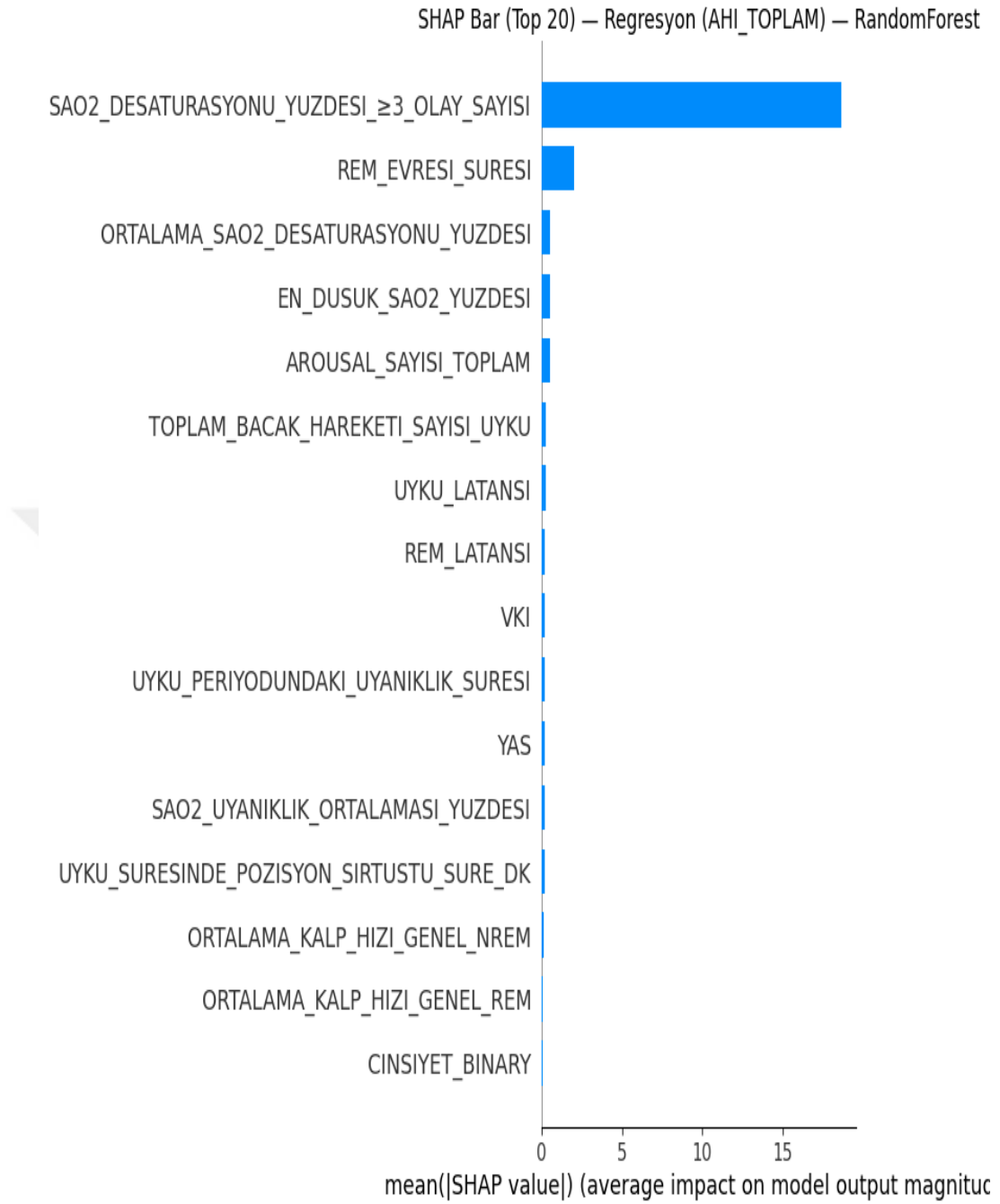
Şekil 4.41. OOF R^2 (↑ daha iyi)

Genel Sonuç: 3 model de benzer performans göstermiştir. CatBoost Regresyon modeli en düşük RMSE ve en yüksek R^2 değerleri ile diğer modellerin arasından öne çıkmaktadır.

Yapay Zeka Açıklanabilirliği SHAP Değerleri



Şekil 4.42. Sınıflama modeli: RandomForest

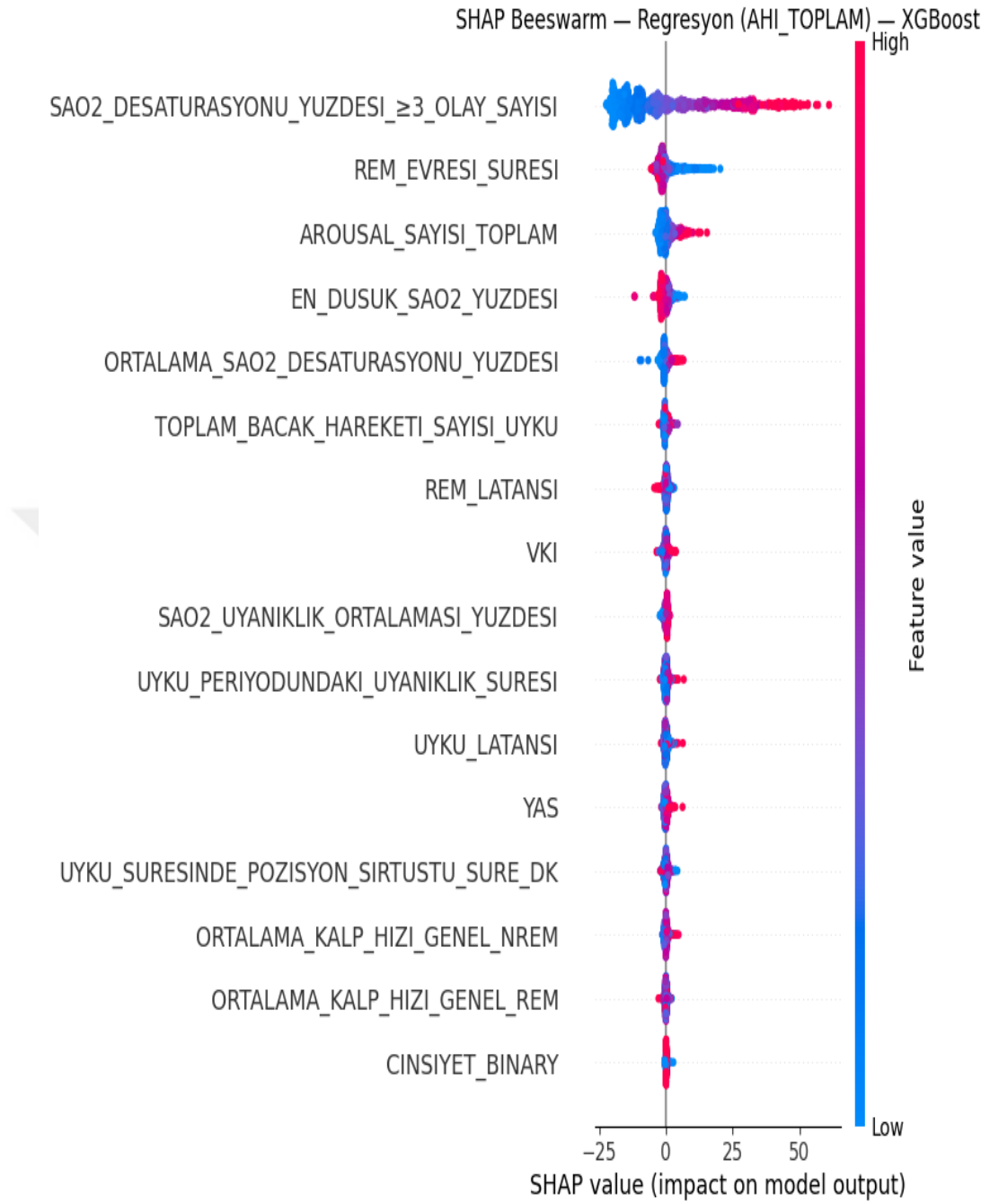


Şekil 4.43. Sınıflama modeli: RandomForest

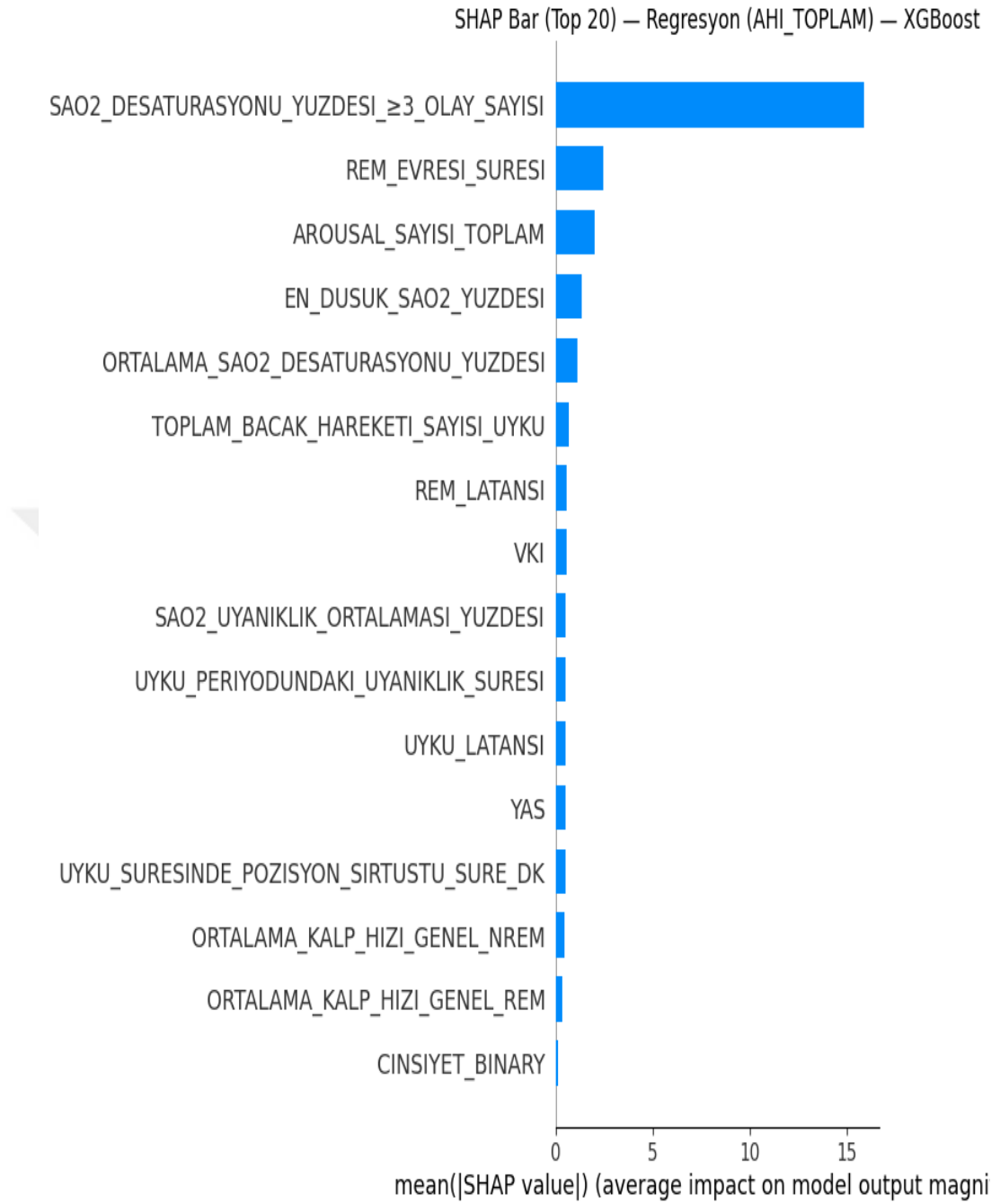
Çizelge 4.35. SHAP önem tablosu – sınıflama – RandomForest

Feature	Mean SHAP	%Importance	Spearman_rho	Dir	ΔSHAP (Q75*Q25)
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	0.240342	44.99	0.759	↑	0.524709
AROUSAL_SAYISI_TOPLAM	0.078639	14.72	0.882	↑	0.194588
EN_DUSUK_SAO2_YUZDESI	0.056679	10.61	-0.905	↓	-0.134393
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	0.046048	8.62	0.875	↑	0.103130
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	0.027728	5.19	0.867	↑	0.063469
REM_EVRESI_SURESI	0.024646	4.61	-0.910	↓	-0.069651
VKI	0.009313	1.74	0.626	↑	0.015481
ORTALAMA_KALP_HIZI_GENEL_NREM	0.009237	1.73	0.786	↑	0.020722
UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK	0.007313	1.37	0.582	↑	0.011482
REM_LATANSI	0.006699	1.25	0.801	↑	0.015506
ORTALAMA_KALP_HIZI_GENEL_REM	0.006534	1.22	0.763	↑	0.014851
YAS	0.005987	1.12	0.518	↑	0.011266
UYKU_LATANSI	0.005659	1.06	-0.393	↓	-0.001645
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	0.004519	0.85	-0.803	↓	-0.007621
UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI	0.004014	0.75	0.438	↑	0.004851
CINSIYET_BINARY	0.000812	0.15	0.705	↑	0.001994

1. AO2_DESATURASYONU_YUZDESİ_≥3_OLAY_SAYISI: önem 44.99%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.76$, $\Delta Q75-Q25=+0.5247$
2. AROUSAL_SAYISI_TOPLAM: önem 14.72%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.88$, $\Delta Q75-Q25=+0.1946$
3. EN_DUSUK_SAO2_YUZDESİ: önem 10.61%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.91$, $\Delta Q75-Q25=-0.1344$
4. ORTALAMA_SAO2_DESATURASYONU_YUZDESİ: önem 8.62%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.88$, $\Delta Q75-Q25=+0.1031$
5. TOPLAM_BACAK_HAREKETI_SAYISI_UYKU: önem 5.19%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.87$, $\Delta Q75-Q25=+0.0635$
6. REM_EVRESI_SURESİ: önem 4.61%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.91$, $\Delta Q75-Q25=-0.0697$
7. VKI: önem 1.74%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.63$, $\Delta Q75-Q25=+0.0155$
8. ORTALAMA_KALP_HIZI_GENEL_NREM: önem 1.73%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.79$, $\Delta Q75-Q25=+0.0207$
9. UYKU_SURESİNDE_POZISYON_SIRTUSTU_SURE_DK: önem 1.37%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.58$, $\Delta Q75-Q25=+0.0115$
10. REM_LATANSI: önem 1.25%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.80$, $\Delta Q75-Q25=+0.0155$
11. ORTALAMA_KALP_HIZI_GENEL_REM: önem 1.22%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.76$, $\Delta Q75-Q25=+0.0149$
12. YAS: önem 1.12%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.52$, $\Delta Q75-Q25=+0.0113$
13. UYKU_LATANSI: önem 1.06%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.39$, $\Delta Q75-Q25=-0.0016$
14. SAO2_UYANIKLIK_ORTALAMASI_YUZDESİ: önem 0.85%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.80$, $\Delta Q75-Q25=-0.0076$
15. UYKU_PERİYODUNDAKI_UYANIKLIK_SURESİ: önem 0.75%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.44$, $\Delta Q75-Q25=+0.0049$
16. CINSİYET_BINARY: önem 0.15%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.70$, $\Delta Q75-Q25=+0.0020$



Şekil 4.44. Sınıflama modeli: XGBoost

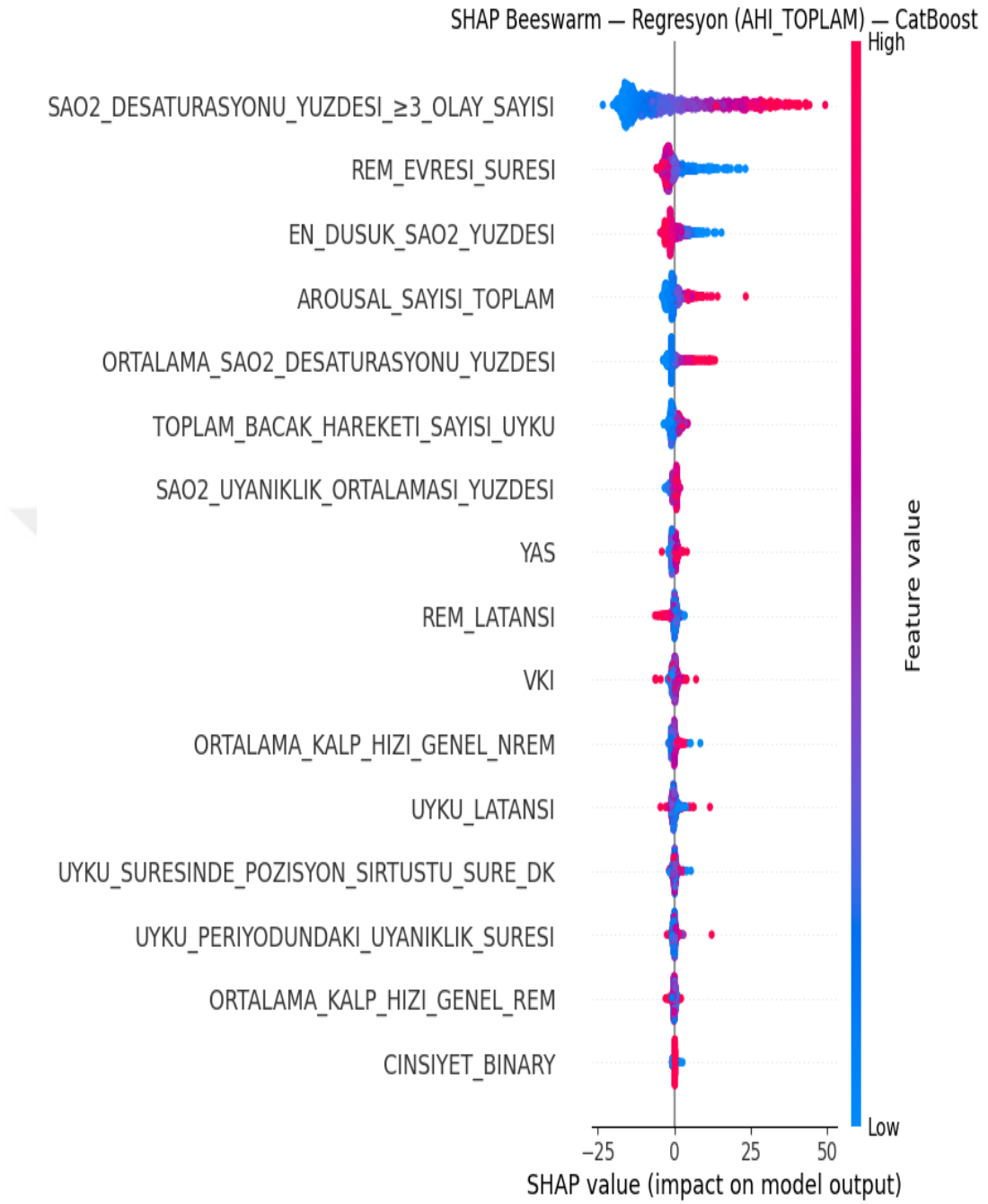


Şekil 4.45. Sınıflama modeli: XGBoost

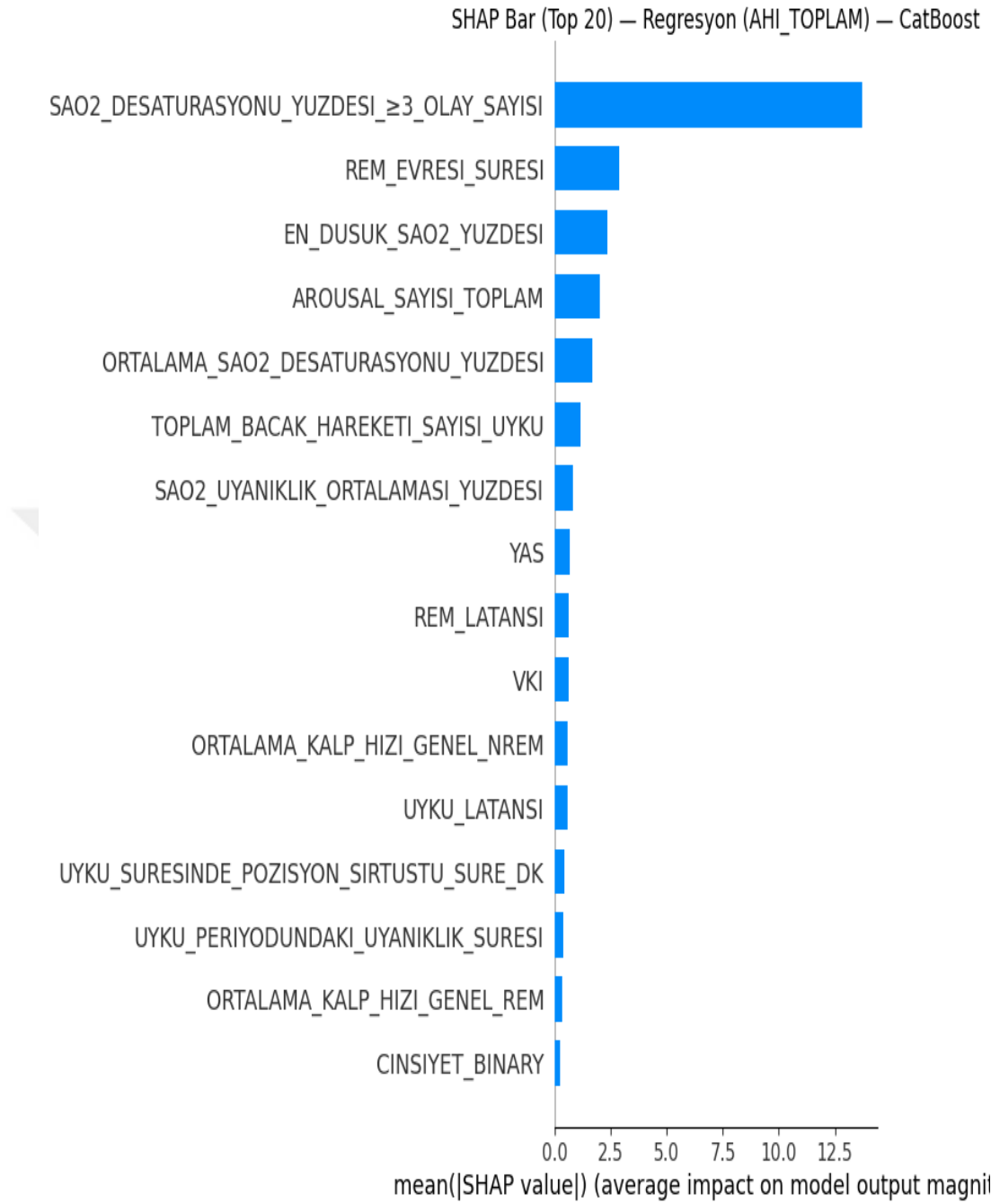
Çizelge 4.36. SHAP önem tablosu – sınıflama – XGBoost

Feature	Mean SHAP	%Importance	Spearman_rho	Dir	ΔSHAP (Q75*Q25)
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	4.345054	44.099998	0.946	↑	11.211550
REM_EVRESI_SURESI	1.101991	11.180000	-0.952	↓	-2.938941
AROUSAL_SAYISI_TOPLAM	0.719429	7.300000	0.882	↑	1.855690
ORTALAMA_KALP_HIZI_GENEL_NREM	0.440408	4.470000	0.855	↑	1.109347
UYKU_LATANSI	0.387389	3.930000	-0.542	↓	-0.563469
VKI	0.370288	3.760000	-0.817	↓	-0.850315
EN_DUSUK_SAO2_YUZDESI	0.336485	3.420000	-0.675	↓	-0.590023
UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK	0.336292	3.410000	0.418	↑	0.413288
REM_LATANSI	0.293768	2.980000	-0.363	↓	-0.431963
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	0.272553	2.770000	0.850	↑	0.701115
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	0.268591	2.730000	0.557	↑	0.375483
UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI	0.263326	2.670000	0.468	↑	0,259536
YAS	0.257169	2.610000	0.548	↑	0.537354
ORTALAMA_KALP_HIZI_GENEL_REM	0.216263	2.190000	-0.878	↓	-0.561394
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	0.149394	1.520000	0.713	↑	0.299038
CINSIYET_BINARY	0.094653	0.960000	0.773	↑	0.234276

1. AO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI: önem 44.10%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.95$, $\Delta Q75-Q25=+11.2115$
2. REM_EVRESI_SURESI: önem 11.18%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.95$, $\Delta Q75-Q25=-2.9389$
3. AROUSAL_SAYISI_TOPLAM: önem 7.30%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.88$, $\Delta Q75-Q25=+1.8557$
4. ORTALAMA_KALP_HIZI_GENEL_NREM: önem 4.47%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.85$, $\Delta Q75-Q25=+1.1093$
5. UYKU_LATANSI: önem 3.93%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.54$, $\Delta Q75-Q25=-0.5635$
6. VKI: önem 3.76%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.82$, $\Delta Q75-Q25=-0.8503$
7. EN_DUSUK_SAO2_YUZDESI: önem 3.42%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.68$, $\Delta Q75-Q25=-0.5900$
8. UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK: önem 3.41%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.42$, $\Delta Q75-Q25=+0.4133$
9. REM_LATANSI: önem 2.98%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.36$, $\Delta Q75-Q25=-0.4320$
10. ORTALAMA_SAO2_DESATURASYONU_YUZDESI: önem 2.77%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.85$, $\Delta Q75-Q25=+0.7011$
11. TOPLAM_BACAK_HAREKETI_SAYISI_UYKU: önem 2.73%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.56$, $\Delta Q75-Q25=+0.3755$
12. UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI: önem 2.67%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.47$, $\Delta Q75-Q25=+0.2595$
13. YAS: önem 2.61%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.55$, $\Delta Q75-Q25=+0.5374$
14. ORTALAMA_KALP_HIZI_GENEL_REM: önem 2.19%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.88$, $\Delta Q75-Q25=-0.5614$
15. SAO2_UYANIKLIK_ORTALAMASI_YUZDESI: önem 1.52%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.71$, $\Delta Q75-Q25=+0.2990$
16. CINSIYET_BINARY: önem 0.96%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.77$, $\Delta Q75-Q25=+0.2343$



Şekil 4.46. Sınıflama modeli: CatBoost

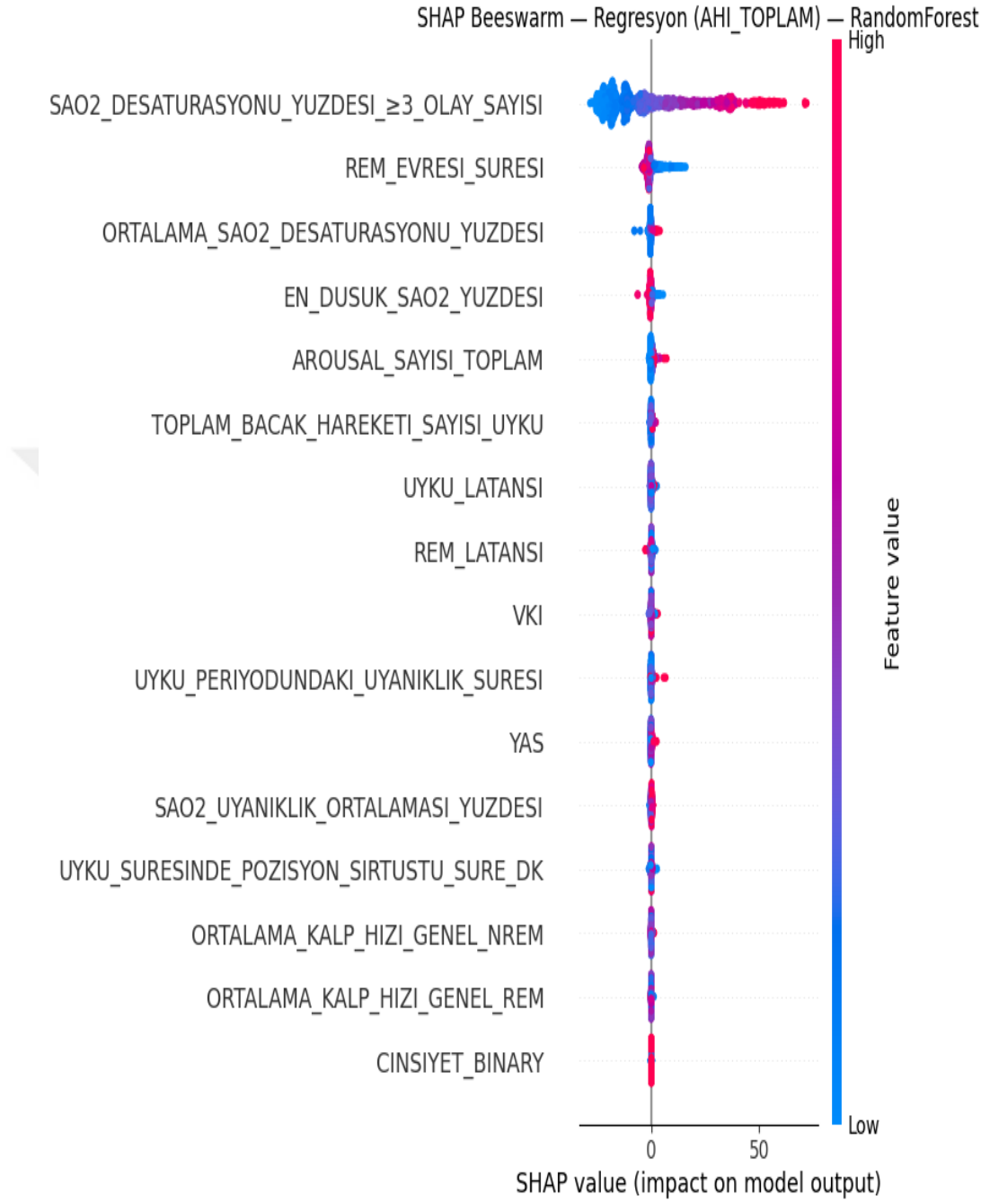


Şekil 4.47. Sınıflama modeli: CatBoost

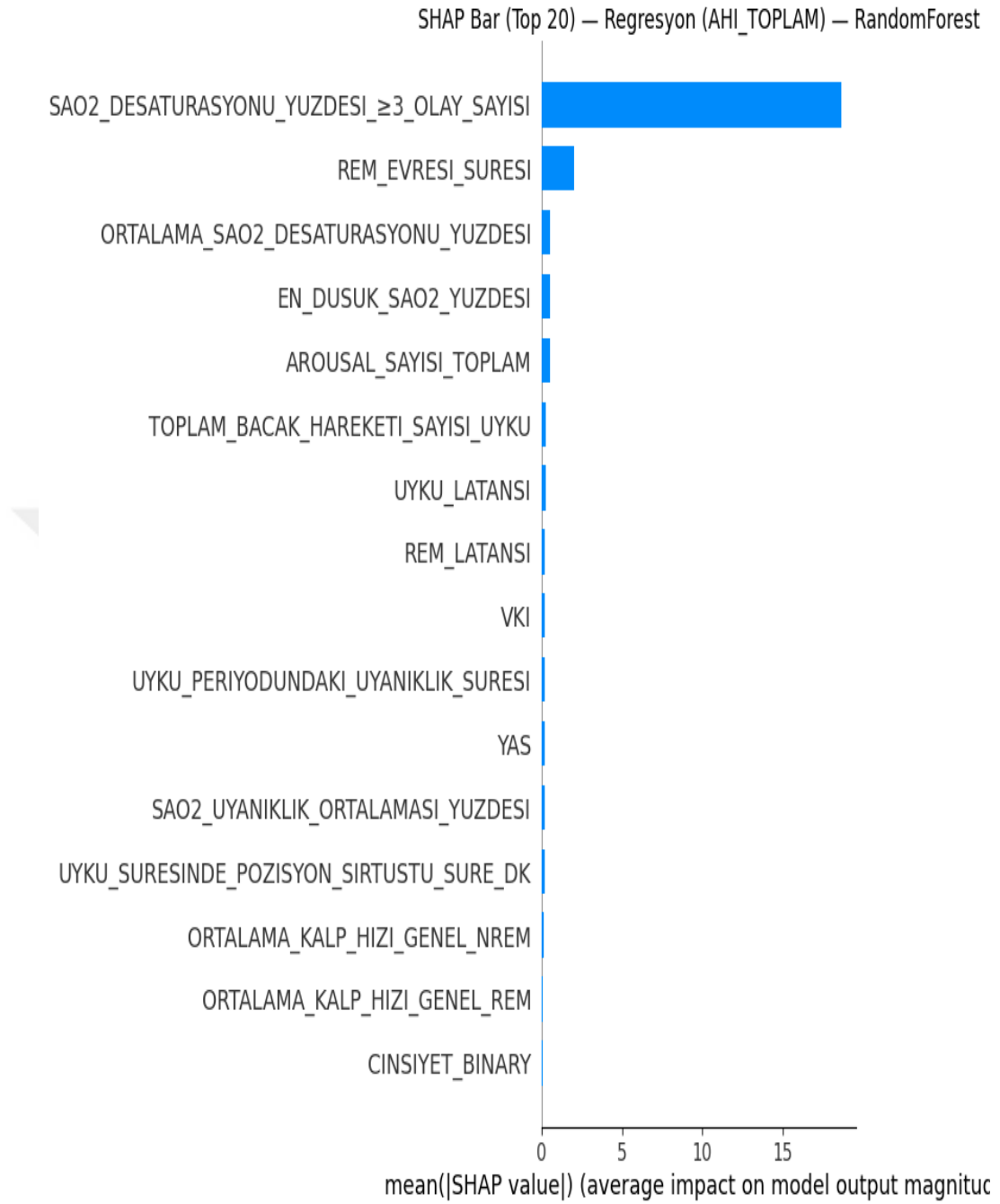
Çizelge 4.37. SHAP önem tablosu – sınıflama – CatBoost

Feature	Mean SHAP	%Importance	Spearman_rho	Dir	ΔSHAP (Q75*Q25)
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	13.707087	47.16	0.984	↑	40.017243
REM_EVRESI_SURESI	2.880939	9.91	-0.886	↓	-8.350840
EN_DUSUK_SAO2_YUZDESI	2.328868	8.01	-0.958	↓	-6.679179
AROUSAL_SAYISI_TOPLAM	1.996721	6.87	0.957	↑	6.399377
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	1.691518	5.82	0.950	↑	4.559142
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	1.162483	4.00	0.881	↑	3.033649
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	0.817520	2.81	0.788	↑	1.640275
YAS	0.678380	2.33	0.837	↑	1.631304
REM_LATANSI	0.619009	2.13	-0.351	↓	-1.323420
VKI	0.618102	2.13	0.622	↑	1.118327
ORTALAMA_KALP_HIZI_GENEL_NREM	0.596321	2.05	0.793	↑	1.571262
UYKU_LATANSI	0.595024	2.05	-0.371	↓	-0.557891
UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK	0.442790	1.52	0.379	↑	0.552403
UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI	0.372029	1.28	0.472	↑	0.615093
ORTALAMA_KALP_HIZI_GENEL_REM	0.315739	1.09	-0.283	↓	-0.193802
CINSIYET_BINARY	0.244167	0.84	0.572	↑	0.3472563

1. SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI: önem 46.34%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.94$, $\Delta Q75-Q25=+11.1273$
2. REM_EVRESI_SURESI: önem 9.77%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.96$, $\Delta Q75-Q25=-2.6089$
3. AROUSAL_SAYISI_TOPLAM: önem 8.85%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.96$, $\Delta Q75-Q25=+2.2957$
4. EN_DUSUK_SAO2_YUZDESI: önem 5.21%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.92$, $\Delta Q75-Q25=-1.2027$
5. TOPLAM_BACAK_HAREKETI_SAYISI_UYKU: önem 5.20%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.92$, $\Delta Q75-Q25=+1.1436$
6. UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK: önem 3.37%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.61$, $\Delta Q75-Q25=+0.4974$
7. ORTALAMA_SAO2_DESATURASYONU_YUZDESI: önem 3.28%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.91$, $\Delta Q75-Q25=+0.7621$
8. YAS: önem 2.96%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.81$, $\Delta Q75-Q25=+0.7052$
9. UYKU_LATANSI: önem 2.70%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.61$, $\Delta Q75-Q25=-0.3322$
10. UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI: önem 2.42%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.37$, $\Delta Q75-Q25=+0.2358$
11. ORTALAMA_KALP_HIZI_GENEL_NREM: önem 2.31%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.84$, $\Delta Q75-Q25=+0.4818$
12. VKI: önem 2.05%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.35$, $\Delta Q75-Q25=-0.2451$
13. REM_LATANSI: önem 1.77%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.65$, $\Delta Q75-Q25=+0.2859$
14. CINSIYET_BINARY: önem 1.66%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.77$, $\Delta Q75-Q25=+0.3645$
15. ORTALAMA_KALP_HIZI_GENEL_REM: önem 1.30%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.59$, $\Delta Q75-Q25=+0.1858$
16. SAO2_UYANIKLIK_ORTALAMASI_YUZDESI: önem 0.81%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.21$, $\Delta Q75-Q25=+0.0101$



Şekil 4.48. Regresyon modeli: RandomForest

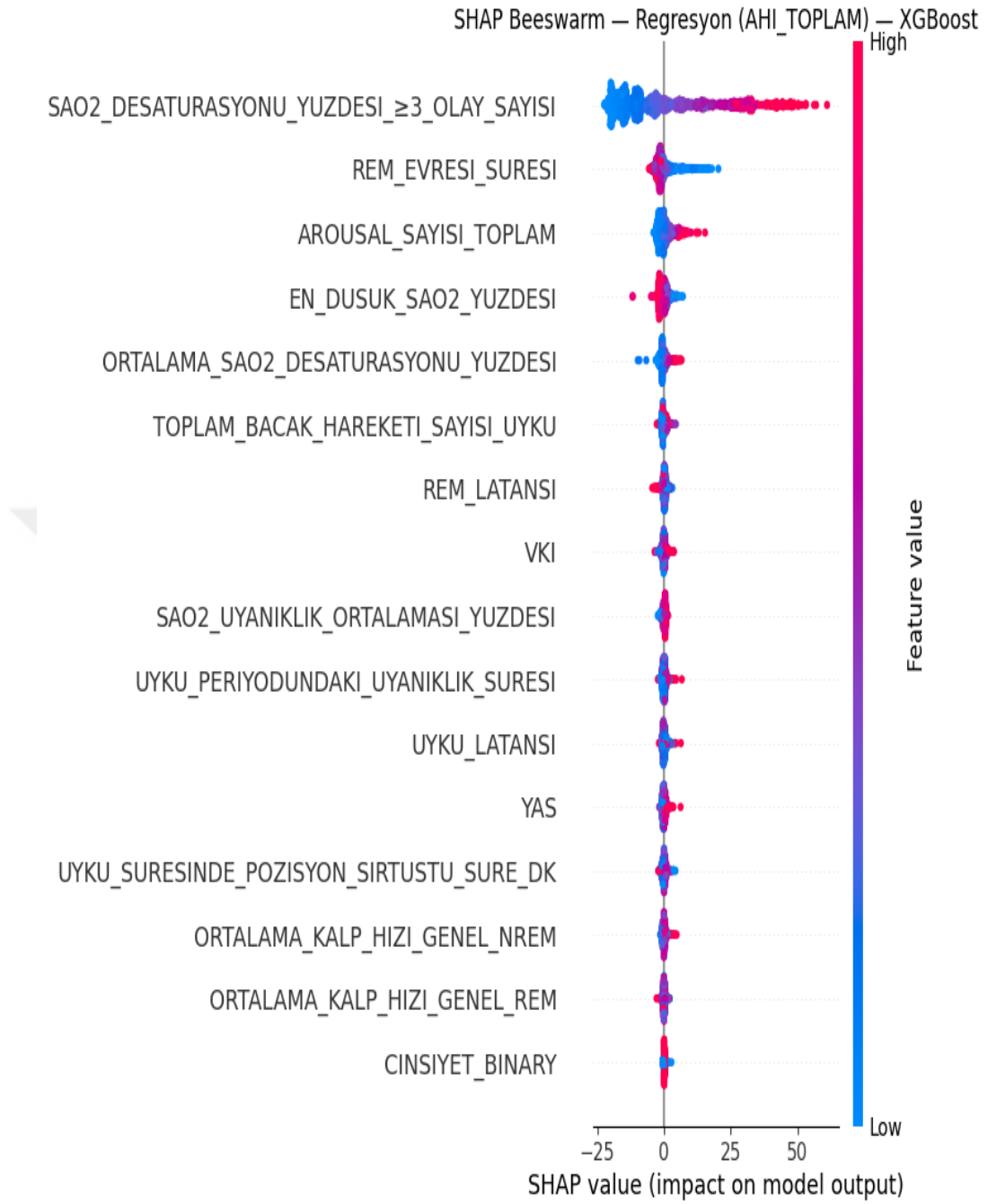


Şekil 4.49. Regresyon modeli: RandomForest

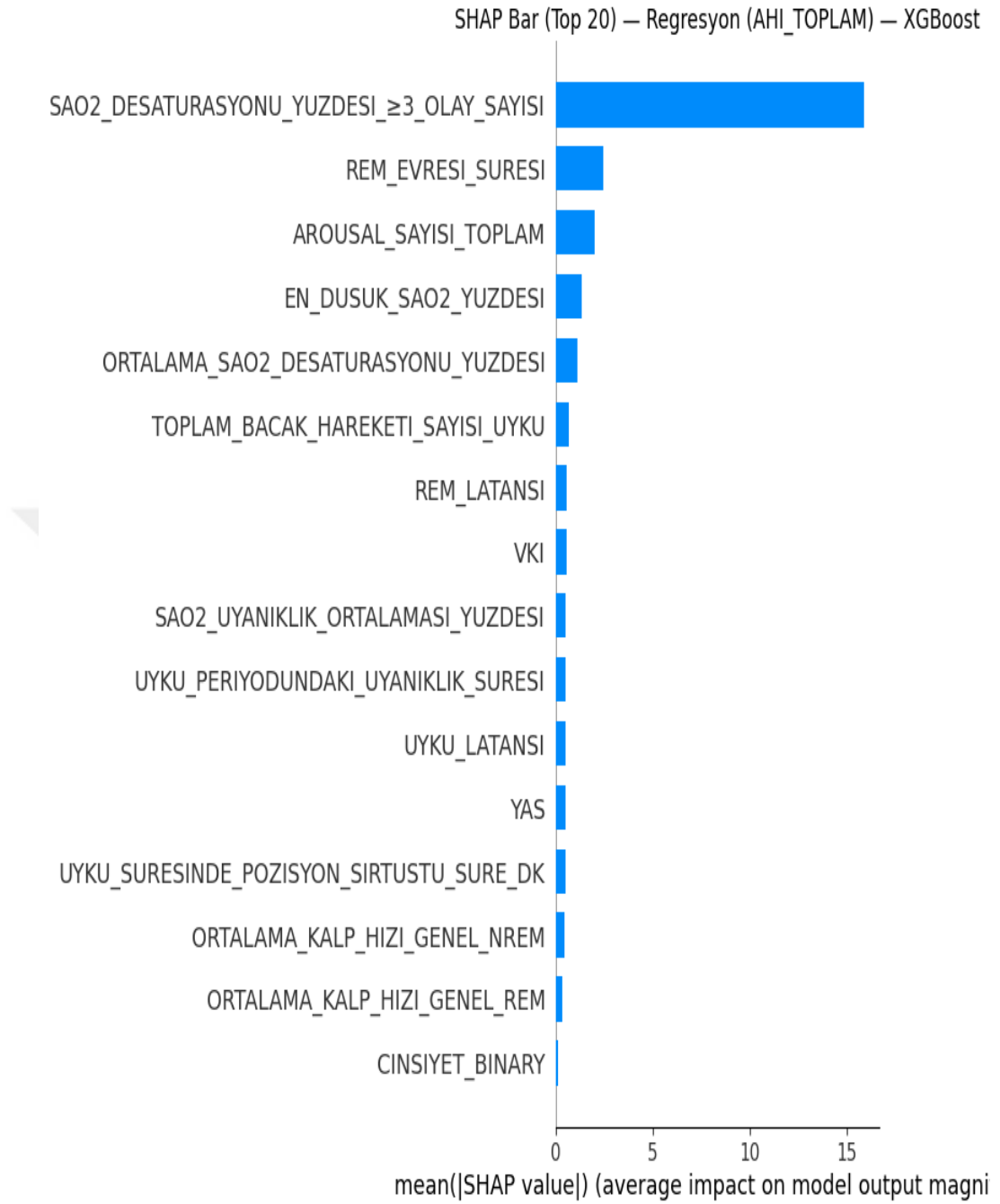
Çizelge 4.38. SHAP önem tablosu – regresyon – RandomForest

Feature	Mean lSHAPl	%Importance	Spearman_rho	Dir	ΔSHAP (Q75*Q25)
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	18.631952	76.68	0.991	↑	56.162971
REM_EVRESI_SURESI	2.019432	8.31	-0.779	↓	-5.343058
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	0.517911	2.13	0.725	↑	0.897746
EN_DUSUK_SAO2_YUZDESI	0.517771	2.13	-0.750	↓	-1.089340
AROUSAL_SAYISI_TOPLAM	0.508297	2.09	0.876	↑	1.336189
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	0.281578	1.16	0.709	↑	0.671592
UYKU_LATANSI	0.251377	1.03	-0.244	↓	-0.241345
REM_LATANSI	0.227786	0.94	-0.454	↓	-0.405812
VKI	0.226720	0.93	0.199	↑	0.221492
UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI	0.219310	0.90	0.577	↑	0.468990
YAS	0.218739	0.90	0.796	↑	0.567686
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	0.212485	0.87	0.845	↑	0.420719
UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK	0.198119	0.82	0.216	↑	0.124297
ORTALAMA_KALP_HIZI_GENEL_NREM	0.130803	0.54	0.037	±	0.104275
ORTALAMA_KALP_HIZI_GENEL_REM	0.095767	0.39	0.008	±	0.001325
CINSIYET_BINARY	0.041186	0.17	0.452	↑	0.057558

1. SAO2_DESATURASYONU_YUZDESI ≥ 3 OLAY_SAYISI: önem 76.68%, yön $\uparrow$ (değer $\uparrow \rightarrow$ model çıktısı $\uparrow$ yönünde); $p=0.99$, $\Delta Q75-Q25=+56.1630$
2. REM_EVRESI_SURESI: önem 8.31%, yön $\downarrow$ (değer $\uparrow \rightarrow$ model çıktısı $\downarrow$ yönünde); $p=-0.78$, $\Delta Q75-Q25=-5.3431$
3. ORTALAMA_SAO2_DESATURASYONU_YUZDESI: önem 2.13%, yön $\uparrow$ (değer $\uparrow \rightarrow$ model çıktısı $\uparrow$ yönünde); $p=0.72$, $\Delta Q75-Q25=+0.8977$
4. EN_DUSUK_SAO2_YUZDESI: önem 2.13%, yön $\downarrow$ (değer $\uparrow \rightarrow$ model çıktısı $\downarrow$ yönünde); $p=-0.75$, $\Delta Q75-Q25=-1.0893$
5. AROUSAL_SAYISI_TOPLAM: önem 2.09%, yön $\uparrow$ (değer $\uparrow \rightarrow$ model çıktısı $\uparrow$ yönünde); $p=0.88$, $\Delta Q75-Q25=+1.3362$
6. TOPLAM_BACAK_HAREKETI_SAYISI_UYKU: önem 1.16%, yön $\uparrow$ (değer $\uparrow \rightarrow$ model çıktısı $\uparrow$ yönünde); $p=0.71$, $\Delta Q75-Q25=+0.6716$
7. UYKU_LATANSI: önem 1.03%, yön $\downarrow$ (değer $\uparrow \rightarrow$ model çıktısı $\downarrow$ yönünde); $p=-0.24$, $\Delta Q75-Q25=-0.2413$
8. REM_LATANSI: önem 0.94%, yön $\downarrow$ (değer $\uparrow \rightarrow$ model çıktısı $\downarrow$ yönünde); $p=-0.45$, $\Delta Q75-Q25=-0.4058$
9. VKI: önem 0.93%, yön $\uparrow$ (değer $\uparrow \rightarrow$ model çıktısı $\uparrow$ yönünde); $p=0.20$, $\Delta Q75-Q25=+0.2215$
10. UYKU_PERİYODUNDAKI_UYANIKLIK_SURESI: önem 0.90%, yön $\uparrow$ (değer $\uparrow \rightarrow$ model çıktısı $\uparrow$ yönünde); $p=0.58$, $\Delta Q75-Q25=+0.4690$
11. YAS: önem 0.90%, yön $\uparrow$ (değer $\uparrow \rightarrow$ model çıktısı $\uparrow$ yönünde); $p=0.80$, $\Delta Q75-Q25=+0.5677$
12. SAO2_UYANIKLIK_ORTALAMASI_YUZDESI: önem 0.87%, yön $\uparrow$ (değer $\uparrow \rightarrow$ model çıktısı $\uparrow$ yönünde); $p=0.84$, $\Delta Q75-Q25=+0.4207$
13. UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK: önem 0.82%, yön $\uparrow$ (değer $\uparrow \rightarrow$ model çıktısı $\uparrow$ yönünde); $p=0.22$, $\Delta Q75-Q25=+0.1243$
14. ORTALAMA_KALP_HIZI_GENEL_NREM: önem 0.54%, yön $\pm$ (etki yönü kararsız/olası nonlinear/etkileşim); $p=0.04$, $\Delta Q75-Q25=+0.1043$
15. ORTALAMA_KALP_HIZI_GENEL_REM: önem 0.39%, yön $\pm$ (etki yönü kararsız/olası nonlinear/etkileşim); $p=0.01$, $\Delta Q75-Q25=+0.0013$
16. CINSİYET_BINARY: önem 0.17%, yön $\uparrow$ (değer $\uparrow \rightarrow$ model çıktısı $\uparrow$ yönünde); $p=0.45$, $\Delta Q75-Q25=+0.0576$



Şekil 4.50. Regresyon modeli: XGBoost

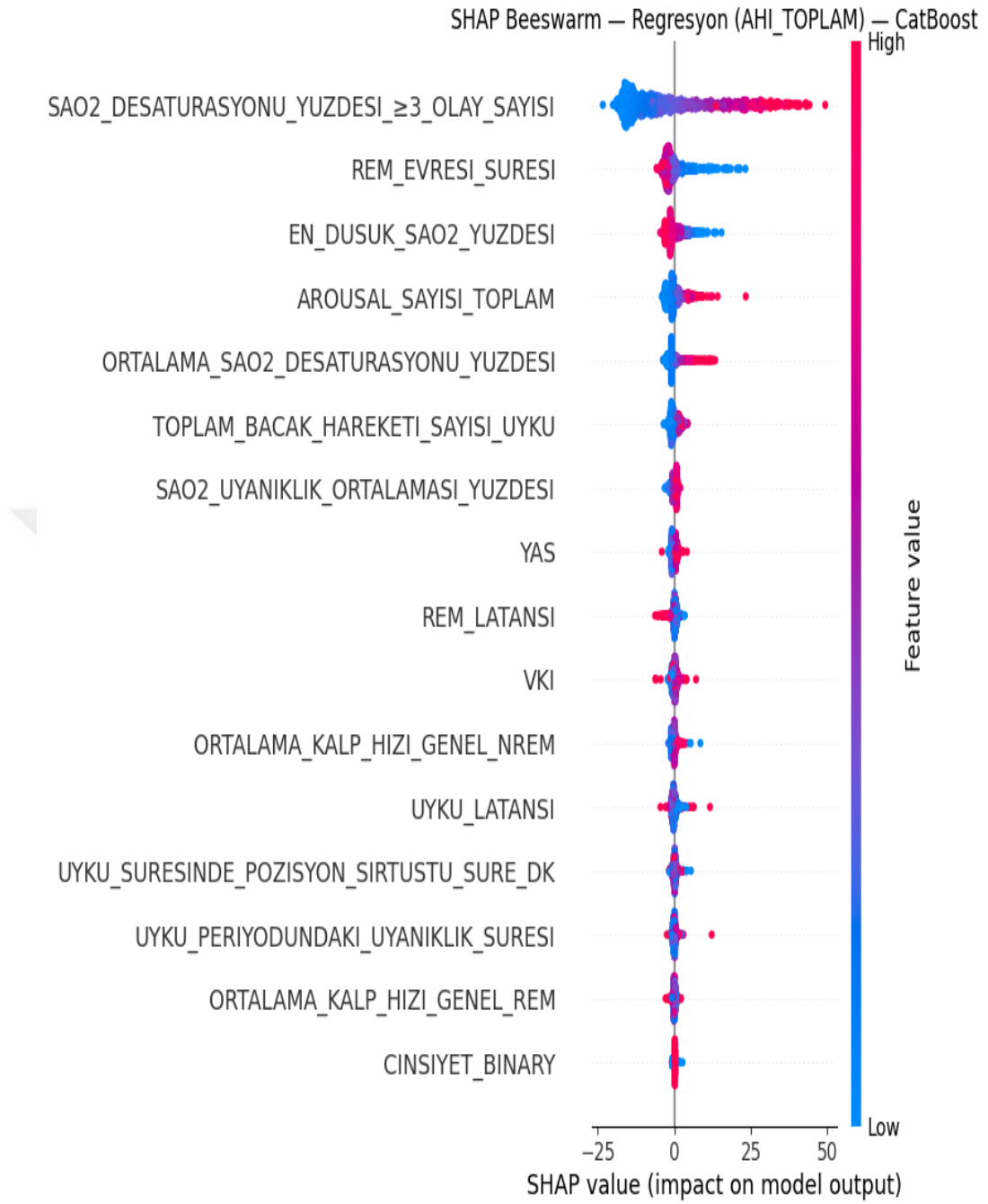


Şekil 4.51. Regresyon modeli: XGBoost

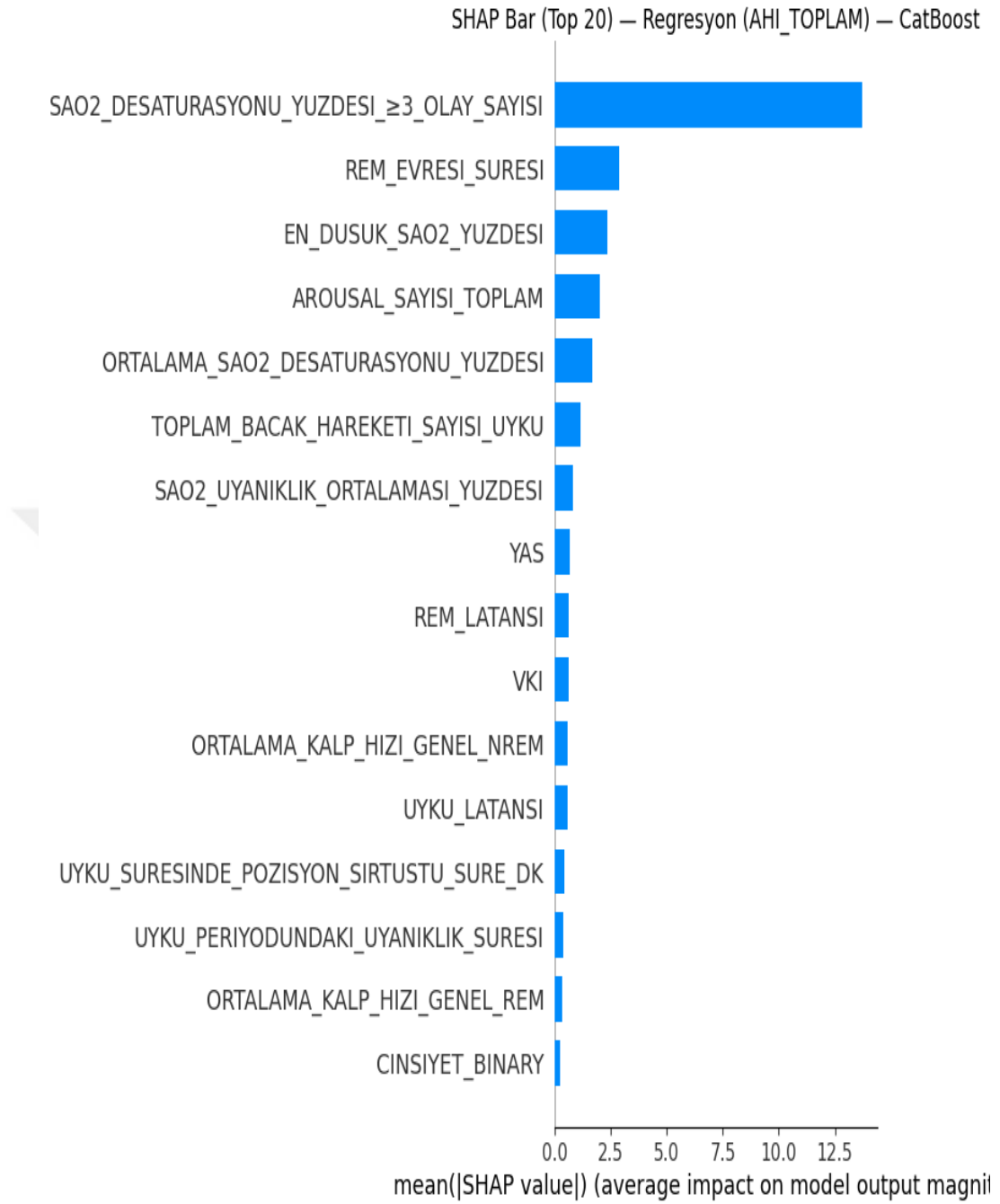
Çizelge 4.39. SHAP önem tablosu – regresyon – XGBoost

Feature	Mean SHAP	%Importance	Spearman_rho	Dir	ΔSHAP (Q75*Q25)
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	15.849689	56.950001	0.990	↑	47.590694
REM_EVRESI_SURESI	2.436576	8.760000	-0.791	↓	-6.544702
AROUSAL_SAYISI_TOPLAM	1.979201	7.110000	0.955	↑	6.262516
EN_DUSUK_SAO2_YUZDESI	1.325812	4.760000	-0.880	↓	-3.208895
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	1.092802	3.930000	0.910	↑	2.632288
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	0.673429	2.420000	0.688	↑	1.381256
REM_LATANSI	0.567700	2.040000	-0.434	↓	-1.121924
VKI	0.546434	1.960000	0.519	↑	0.870647
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	0.528074	1.900000	0.826	↑	1.051757
UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI	0.512353	1.840000	0.440	↑	0.806269
UYKU_LATANSI	0.496180	1.780000	-0.294	↓	-0.377155
YAS	0.480464	1.730000	0.671	↑	0.985648
UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK	0.475627	1.710000	0.217	↑	0.295997
ORTALAMA_KALP_HIZI_GENEL_NREM	0.417693	1.500000	0.644	↑	0.938739
ORTALAMA_KALP_HIZI_GENEL_REM	0.310932	1.120000	-0.245	↓	-0.122222
CINSIYET_BINARY	0.136829	0.490000	0.491	↑	0.140792

1. SAO2_DESATURASYONU_YUZDESİ_≥3_OLAY_SAYISI: önem 56.95%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.99$, $\Delta Q75-Q25=+47.5907$
2. REM_EVRESİ_SURESİ: önem 8.76%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.79$, $\Delta Q75-Q25=-6.5447$
3. AROUSAL_SAYISI_TOPLAM: önem 7.11%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.95$, $\Delta Q75-Q25=+6.2625$
4. EN_DUSUK_SAO2_YUZDESİ: önem 4.76%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.88$, $\Delta Q75-Q25=-3.2089$
5. ORTALAMA_SAO2_DESATURASYONU_YUZDESİ: önem 3.93%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.91$, $\Delta Q75-Q25=+2.6323$
6. TOPLAM_BACAK_HAREKETİ_SAYISI_UYKU: önem 2.42%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.69$, $\Delta Q75-Q25=+1.3813$
7. REM_LATANSİ: önem 2.04%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.43$, $\Delta Q75-Q25=-1.1219$
8. VKİ: önem 1.96%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.52$, $\Delta Q75-Q25=+0.8706$
9. SAO2_UYANIKLIK_ORTALAMASI_YUZDESİ: önem 1.90%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.83$, $\Delta Q75-Q25=+1.0518$
10. UYKU_PERİYODUNDAKİ_UYANIKLIK_SURESİ: önem 1.84%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.44$, $\Delta Q75-Q25=+0.8063$
11. UYKU_LATANSİ: önem 1.78%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.29$, $\Delta Q75-Q25=-0.3772$
12. YAS: önem 1.73%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.67$, $\Delta Q75-Q25=+0.9856$
13. UYKU_SURESİNDE_POZİSYON_SIRTUSTU_SURE_DK: önem 1.71%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.22$, $\Delta Q75-Q25=+0.2960$
14. ORTALAMA_KALP_HIZI_GENEL_NREM: önem 1.50%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.64$, $\Delta Q75-Q25=+0.9387$
15. ORTALAMA_KALP_HIZI_GENEL_REM: önem 1.12%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.24$, $\Delta Q75-Q25=-0.1222$
16. CINSİYET_BINARY: önem 0.49%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.49$, $\Delta Q75-Q25=+0.1408$



Şekil 4.52. Regresyon modeli: CatBoost

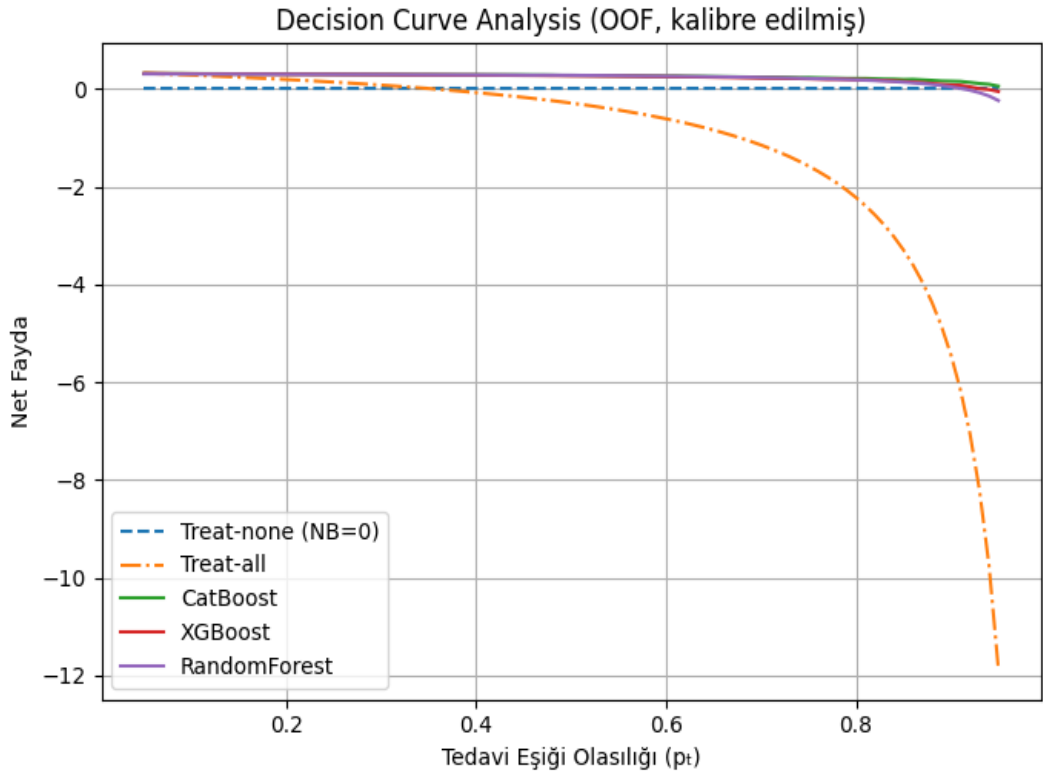


Şekil 4.53. Regresyon modeli: CatBoost

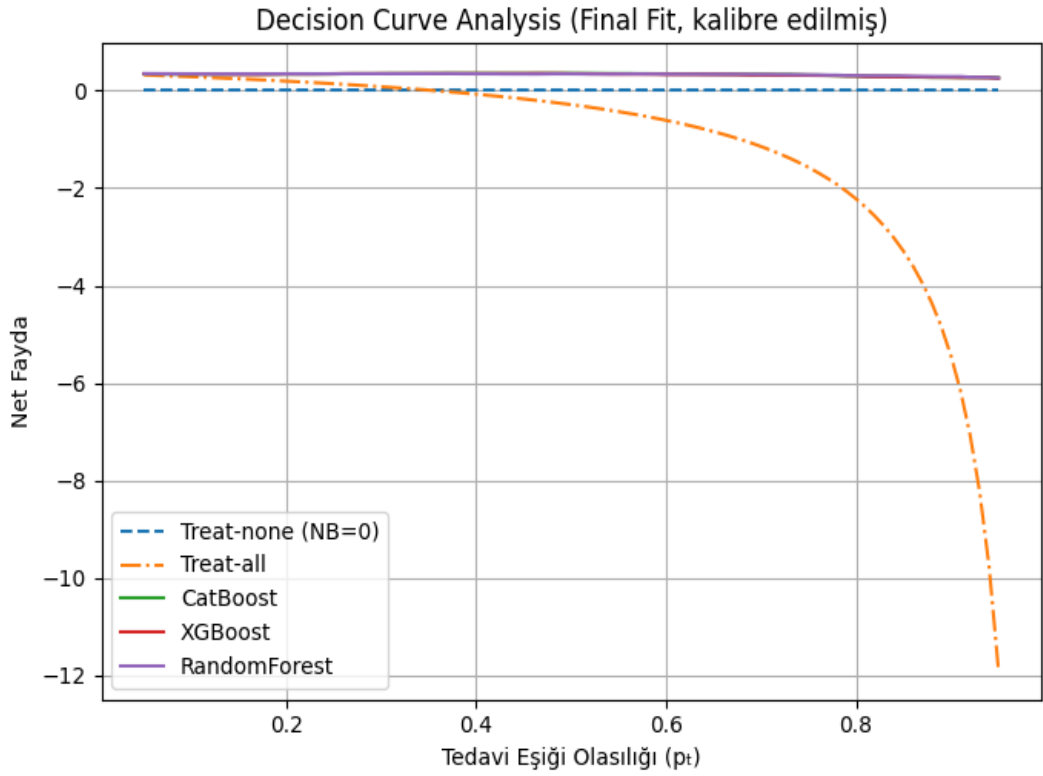
Çizelge 4.40. SHAP önem tablosu – regresyon – CatBoost

Feature	Mean SHAP	%Importance	Spearman_rho	Dir	ΔSHAP (Q75*Q25)
SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI	13.707087	47.16	0.984	↑	40.017243
REM_EVRESI_SURESI	2.880939	9.91	-0.886	↓	-8.350840
EN_DUSUK_SAO2_YUZDESI	2.328868	8.01	-0.958	↓	-6.679179
AROUSAL_SAYISI_TOPLAM	1.996721	6.87	0.957	↑	6.399377
ORTALAMA_SAO2_DESATURASYONU_YUZDESI	1.691518	5.82	0.950	↑	4.559142
TOPLAM_BACAK_HAREKETI_SAYISI_UYKU	1.162483	4.00	0.881	↑	3.033649
SAO2_UYANIKLIK_ORTALAMASI_YUZDESI	0.817520	2.81	0.788	↑	1.640275
YAS	0.678380	2.33	0.837	↑	1.631304
REM_LATANSI	0.619009	2.13	-0.351	↓	-1.323420
VKI	0.618102	2.13	0.622	↑	1.118327
ORTALAMA_KALP_HIZI_GENEL_NREM	0.596321	2.05	0.793	↑	1.571262
UYKU_LATANSI	0.595024	2.05	-0.371	↓	-0.557891
UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK	0.442790	1.52	0.379	↑	0.552403
UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI	0.372029	1.28	0.472	↑	0.615093
ORTALAMA_KALP_HIZI_GENEL_REM	0.315739	1.09	-0.283	↓	-0.193802
CINSIYET_BINARY	0.244167	0.84	0.572	↑	0.372563

1. SAO2_DESATURASYONU_YUZDESI_≥3_OLAY_SAYISI: önem 47.16%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.98$, $\Delta Q75-Q25=+40.0172$
2. REM_EVRESI_SURESI: önem 9.91%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.89$, $\Delta Q75-Q25=-8.3508$
3. EN_DUSUK_SAO2_YUZDESI: önem 8.01%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.96$, $\Delta Q75-Q25=-6.6792$
4. AROUSAL_SAYISI_TOPLAM: önem 6.87%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.96$, $\Delta Q75-Q25=+6.3994$
5. ORTALAMA_SAO2_DESATURASYONU_YUZDESI: önem 5.82%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.95$, $\Delta Q75-Q25=+4.5591$
6. TOPLAM_BACAK_HAREKETI_SAYISI_UYKU: önem 4.00%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.88$, $\Delta Q75-Q25=+3.0336$
7. SAO2_UYANIKLIK_ORTALAMASI_YUZDESI: önem 2.81%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.79$, $\Delta Q75-Q25=+1.6403$
8. YAS: önem 2.33%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.84$, $\Delta Q75-Q25=+1.6313$
9. REM_LATANSI: önem 2.13%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.35$, $\Delta Q75-Q25=-1.3234$
10. VKI: önem 2.13%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.62$, $\Delta Q75-Q25=+1.1183$
11. ORTALAMA_KALP_HIZI_GENEL_NREM: önem 2.05%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.79$, $\Delta Q75-Q25=+1.5713$
12. UYKU_LATANSI: önem 2.05%, yön ↓ (değer ↑ → model çıktısı ↓ yönünde); $p=-0.37$, $\Delta Q75-Q25=-0.5579$
13. UYKU_SURESINDE_POZISYON_SIRTUSTU_SURE_DK: önem 1.52%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.38$, $\Delta Q75-Q25=+0.5524$
14. UYKU_PERIYODUNDAKI_UYANIKLIK_SURESI: önem 1.28%, yön ↑ (değer ↑ → model çıktısı ↑ yönünde); $p=0.47$, $\Delta Q75-Q25=+0.6151$



Şekil 4.54. Desicion Curve Analizi (DCA)



Şekil 4.55. Decision Curve Analysis (final fit, kalibre edilmiş)

Çizelge 4.41. DCA'nın sayısal özeti

Set	Model	Prevalans	NB_max	pt_at_NB_max	DominanceRanges(pt)
OOF	CatBoost	0.355748	0.319272	0.05	0.06-0.95
OOF	XGBoost	0.355748	0.320927	0.05	0.06-0.93
OOF	RandomForest	0.355748	0.308026	0.05	0.08-0.91
Final	CatBoost	0.355748	0.355748	0.47	0.05-0.95
Final	XGBoost	0.355748	0.352977	0.46	0.05-0.95
Final	RandomForest	0.355748	0.349469	0.05	0.05-0.95

DCA analizine göre üç model geniş p_t aralıklarında pozitif fayda sağlayarak klinik kullanım potansiyeline sahiptir. OOF analizinde XGBoost düşük eşiklerde sınırlı bir üstünlük göstermiş olsa da CatBoost'un üstünlük aralığı daha geniştir ve ROC-AUC, F1 gibi klasik performans metriklerinde en yüksek başarıyı sağlamıştır. Bundan ötürü klinik senaryolar CatBoost modeline göre hazırlanmış; sabit duyarlılık (0,85) ve Youden kesim noktası gibi klinik karar eşikleri CatBoost modelinin sonuçları üzerinden belirlenmiştir.

Çizelge 4.42. Klinik senaryo

Threshold	Sensitivity	Specificity	PPV	NPV	TP	FP	TN	FN
0.5578	0.851	0.958 [0.941–0.973]	0.918 [0.889–0.947]	0.921 [0.902–0.939]	279	25	569	49

Sensivite sabiti: 0.85

Threshold: 0.5578

Sensitivity: 0.851 (sabit tutulmuş)

Specificity: 0.958 [0.941–0.973]

PPV: 0.918 [0.889–0.947]

NPV: 0.921 [0.902–0.939]

Doğru sınıflama (TP=279, TN=259) çok yüksek, FN=49 ile sınırlı.

Negatifleri minimize etmek, duyarlılığı sabitleyip klinik için güvenilir bir özgüllük sağlamaktadır.

Çizelge 4.43. Youden's J

Threshold	Sensitivity	Specificity	PPV	NPV	TP	FP	TN	FN
0.3202	0.884 [0.848– 0.918]	0.943 [0.923–0.960]	0.895 [0.864–0.925]	0.936 [0.918–0.954]	290	34	560	38

Youden's J optimim: Threshold: 0.3202

Sensitivity: 0.884 [0.848–0.918]

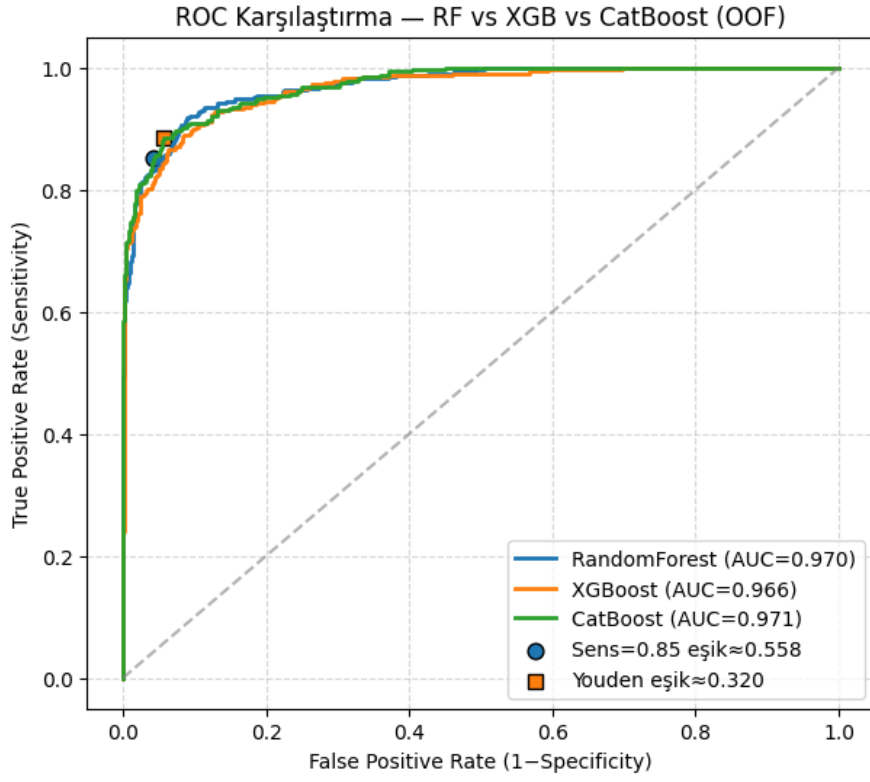
Specificity: 0.943 [0.923–0.960]

PPV: 0.895 [0.864–0.925]

NPV: 0.936 [0.918–0.954]

Doğru sınıflama (TP=290, TN=560) daha yüksek, FN=38 ile daha avantajlı. Duyarlılık ve özgüllük arasındaki dengeyi maksimize etmektedir.

Genel Yorum: Sabit sensitivite eşiği kaçırılmaması gereken vakaların öne çıkarılmasına uygundur. Youden's J ise dengeli bir tanı performansı sağlar ve yanlış negatif sayısını düşürür. Her iki eşik de yüksek PPV ve NPV değerleri ile CatBoost Sınıflama modeli için klinik kullanımda güvenilirlik göstermektedir.



Şekil 4.56. ROC karşılaştırma – RF vs XGB vs CatBoost (OOF)

CatBoost'un en iyi sınıflandırıcı olduğu ve klinik senaryoda yüksek sensitive gösterdiği ve uygulanabilir eşiklerin bulunduğunu ortaya koymuştur.

5. TARTIŞMA

5.1. Genel Değerlendirme ve Çalışmanın Önemi

Bu çalışma; PSG testi yapılırken bu testte elde edilen fakat AHİ hesaplama formülüne doğrudan katılmayan fizyolojik değişken değerleri üzerinde hem regresyon hem sınıflama modelleri ile AHİ değerini tahmin etmeyi hedeflemiştir. Çalışmanın özgün yanı, literatürde AHİ ile ilişkisi bulunan değişkenleri kullanarak çok boyutlu bir öngörü modeli geliştirmiş olmasıdır. Bu yaklaşım, AHİ'nin OSA'nın klinik yükünü tek başına yansıtamaması ile uyumludur. AHİ, tek başına hastaların kardiyorespiratuar komorbidite riskini, gündüz semptomlarını ve yaşam kalitesini yansıtmakta yetersizdir (124).

Bu çalışmada hem regresyon hem sınıflama modelleri için 3 ayrı öğrenim algoritması seçilmiştir. Klinik karar destek sürecinde esnek bir kullanım imkânı sunmaktadır. Ağır OSA hastalarının erken tanı alması ve tedavi edilmesi sağlık sistemleri için kritik öneme sahiptir. Ağır OSA'lı hastalarda komorbidite riski daha yüksektir. 2021 yılında yapılan bir sistematik derlemede, makine öğrenmesi tabanlı modellerin ağır OSA'yı sınıflamada geleneksel istatistiksel yöntemlere göre daha başarılı olduğu gösterilmiştir (125).

Türkiye'de yapılan çalışmalar çoğunlukla klasik istatistiksel yöntemlere dayanmaktadır. Global ölçekte, son yıllarda yapılan meta-analizlerde gradient boosting yöntemlerinin AHİ tahmininde geleneksel lojistik regresyon modellerine göre daha başarılı olduğu vurgulanmıştır. Bu tez çalışması, makine öğrenmesini ileri düzeyde uygulayarak özgün bir yaklaşım ortaya koymaktadır (126).

Bu çalışma yalnızca metodolojik açıdan değil, klinik fayda açısından da katkı sağlamayı hedeflemiştir. Geliştirilen modellerden elde edilen sonuçlar neticesinde ağır hastaların erken tanılanabilmesi, hafif ve orta sınıftaki hastalara otomatik skorlama yapılabilmesi mümkün kılınmıştır. Bu yönüyle karar destek süreçlerini hızlandırma ve maliyet-etkinlik sağlama potansiyeline sahiptir. Bu bakımdan hem global literatür ile uyumludur hem de klinik pratiğe uygulanabilir bir yaklaşımdır.

5.2. Tanımlayıcı İstatistiklerin ve Hasta Özelliklerinin Değerlendirilmesi

Çalışmada toplam 922 birey değerlendirilmiştir ve olguların %35,6'sı ağır OSA grubunda yer almaktadır. Ortalama AHİ değerleri $28,4 \pm 25,8$ olup geniş bir dağılım göstermektedir. Literatürde bildirilen geniş ölçekli çalışmalarda OSA'nın toplumda görülme sıklığının %9–38 arasında değiştiği ve özellikle ağır OSA prevalansının yüksek olduğu belirtilmiştir (127).

Yaş ortalaması orta yaş grubunda yoğunlaşmaktadır. Yaşın, OSA gelişiminde bağımsız bir değişken olduğunu vurgulayan çalışmalarla uyumludur. Heinzer ve ark.'nın yaptığı prospektif bir çalışmada, yaş ilerledikçe uyku mimarisinde değişikliklerin olduğu; REM süresinde kısalma ve arousal sayısında artış olduğu görülmüştür. Yaşın, OSA riskini AHİ'den bağımsız olarak arttırdığı vurgulanmıştır (128).

Cinsiyet dağılımında erkek hastaların daha yüksek olması da, erkeklerde OSA riskinin kadınlardan 2–3 kat daha fazla olduğunu gösteren geniş çaplı kohort çalışmaları ile uyumludur (129). Postmenopozal dönemde kadınlarda OSA prevalansının hızla yükseldiği ve klinikte erkeklere benzer bir tablo sunduğu da literatürde ifade edilmiştir (130).

VKI değerleri ağır OSA grubunda daha yüksek bulunmuştur. Bu bulgu, obezitenin OSA gelişimindeki rolünü gösteren çok sayıdaki çalışma ile uyumludur. Literatürde, VKİ'deki her 1 birimlik artışın OSA gelişim riskini %14 arttırdığı ve özellikle abdominal obezitenin AHİ'den bağımsız olarak üst solunum yolu kollapsı ile ilişkili olduğu gösterilmiştir (131).

Sonuç olarak, çalışmadaki tanımlayıcı veriler OSA'nın bilinen epidemiyolojik ve klinik özelliklerini destekler niteliktedir. Hasta grubunun demografik yapısı, global literatürde belirtilen risk faktörleri ile örtüşmektedir. Bu durum, çalışmanın genellenebilirliğini güçlendirmektedir.

5.3. Normallik, Varyans, Grup Karşılaştırmaları ve İstatistiksel Analizler

Değişkenlerin dağılımı Shapiro-Wilk testi ile incelenmiş ve tüm değişkenler için $p < 0.05$ bulunarak normal dağılım varsayımı reddedilmiştir. Bu durum,

biyomedikal değerlerin çoğunlukla normal dağılmadığını ve çarpık dağılım gösterdiğini ortaya koymaktadır (132).

Levene testi ile varyans homojenliği değerlendirilmiş; arousal sayısı, en düşük SaO₂, ortalama desatürasyon yüzdesi, SaO₂ $\geq$ %3 olay sayısı, toplam bacak hareketi ve yaş değişkenlerinde $p < 0.05$ saptanarak varyans homojenliğinin bozulduğu görülmüştür. Bu bulgu, OSA'nın fenotipik heterojenitesi ile uyumludur. Benzer AHİ değerlerine sahip hastalarda klinik farklılıkların oluşabileceği ile uyumludur (133).

Etki büyüklükleri Cliff's delta ile değerlendirilmiştir. En güçlü etkiyi arousal sayısı (-0.73), en düşük SaO₂ (0.73), desatürasyon yüzdesi (-0.72), desatürasyon olay sayısı (-0.93) ve bacak hareketleri (-0.50) göstermiştir.

VKİ ve REM süresi orta düzeyde etkiye sahiptir. REM latansı ve kalp hızı düşük etkiyi göstermiştir. Bu bulgu, yalnızca p değeriyle değil etki büyüklüğüyle de yorum yapmanın ortaya koymaktadır (135).

5.4. Korelasyon Analizleri ve Çoklu Doğrusal Bağlantı (VIF)

Değişkenler arasındaki ilişkiler Spearman korelasyon analizi ile değerlendirilmiş; en güçlü korelasyonun AHİ ile desatürasyon değişkenleri ve arousal sayısı arasında olduğu görülmüştür. Tüm değişkenler içerisinde $|\rho| \geq 0.90$ olan yalnızca 1 korelasyon ilişkisi bulunmuştur (NREM kalp hızı ile REM ortalama kalp hızı). Kalp hızındaki bu durumun fizyolojik olarak öneme sahip olduğu, OSA'nın kardiyolojik etkilerini ve sempatik aktivasyonun farklı uyku evrelerindeki yansıması olduğu için modele katılmıştır. VIF değerleri incelendiğinde tüm değişkenlerin 10'un altında olduğu görülmüştür. Bu da kritik düzeyde multikolinearite olmadığını ve değişkenlerin regresyon ve sınıflama modellerinde birlikte kullanılabileceğini göstermektedir (136).

Makine öğrenmesi algoritmalarında tek değişken düzeyinde anlamlı olmayan faktörler, çok değişkenli analizlerde değişkenlerin birbirleri ile olan etkileşimleri sayesinde öngörü gücüne katkıda bulunabilmektedirler. Bundan dolayı tek değişken olarak anlamlı olmayan değişkenlerde modele dahil edilmiştir ve model içerisindeki etki güçleri SHAP analizleri ile ortaya konmuştur (137).

5.5. Regresyon Modellerinin Bulguları

AHI değerlerini öngörmeye yönelik yapılan lineer regresyon analizinde en güçlü bağımsız değişkenlerin oksijenizasyon metrikleri olduğu görülmüştür. SaO₂ desatürasyon $\geq 3\%$ olay sayısı, ortalama desatürasyon yüzdesi ve en düşük SaO₂ değerleri AHI ile güçlü ve bağımsız ilişki göstermektedir. Arousal sayısı da güçlü prodüktör olarak karşımıza çıkmıştır. OSA'nın yalnızca apne-hipopne sıklığı ile değil oksijenlenme ve uyku fragmentasyonunun hastalık yükündeki belirleyici rolü ile de ilişkili olduğunu göstermektedir (138). Yaş, uyku latansı ve uyku süresince uyanıklık gibi geriye kalan değişkenler istatistiksel anlamlılık göstermemiştir. Ancak makine öğrenmesi modellerine öngörü gücüne katkı sağlayabileceği için kalan değişkenler korunmuştur.

Lojistik regresyon analizleri benzer sonuçlar vermiştir. Oksijenizasyon değişkenleri ve arousal sayısı güçlü katkı sağlarken, VKİ orta düzeyde; kalp hızı parametreleri ise düşük fakat anlamlı katkı sağlamıştır. Bu durum, obezite ve kardiyorespiratuvar etkilerin OSA fizyopatolojisindeki tamamlayıcı rolünü desteklemektedir (139).

Literatürde regresyon modelleri sıklıkla kullanılmış olup desatürasyon indeksleri ve minimum SaO₂ değerleri OSA şiddetinde en güçlü Prodüktör olarak karşımıza çıkmaktadır (140). Büyük ölçekli kohort çalışmalarında VKİ ve yaşın OSA ile ilişkili olduğu lakin oksijenizasyon değişkenleri kadar güçlü olmadığı vurgulanmıştır (141).

5.6. Makine Öğrenmesi Modelleri ve Performans Karşılaştırmaları

Çalışmamızda üç farklı model hem regresyon hem sınıflama görevlerinde kullanılmıştır: RandomForest, XGBoost ve CatBoost. Regresyon analizlerinde CatBoost modeli en düşük hata oranlarını (MAE, RMSE) göstermiş; sınıflama modellerinde ise ROC-AUC ve PR-AUC değerlerinde de en yüksek değer yine Catboost modelinden elde edilmiştir. RandomForest, özellikle overfitting (aşırı uyum) eğilimi ile öne çıkarken, XGBoost yüksek doğruluk ve denge sağlamıştır ancak bu iki model CatBoost modeli kadar güçlü olamamıştır. CatBoost modelinin kategorik değişkenleri doğrudan işleyebilme avantajı sayesinde OSA verilerinde daha yüksek doğruluk sağlayabileceğini göstermiştir (142).

Sınıflama modellerinde ROC-AUC ayırt ediciliği ortaya koyarken dengesiz veri setlerinde PR-AUC klinik olarak daha anlamlı bir metrik olarak karşımıza çıkmaktadır. Çalışmadaki PR-AUC değerlerinin yüksek olması, Ağır OSA hastalarının doğru sınıflandırılmasında modellerin güçlü olduğunu göstermektedir. F1 skorları duyarlılık ve özgüllüğün dengeli bir şekilde optimize edildiğini ortaya koymaktadır. Brier skorları model kalibrasyonlarının başarılı olduğunu göstermektedir. Bu bulgular modellerin klinik güvenilirlik açısından güçlü olduğunu göstermektedir (143).

Son yıllarda yapılan meta-analizlerde CatBoost modelinin yüksek boyutlu ve heterojen veri setlerinde en başarılı makine öğrenmesi modellerinden biri olduğu belirtilmiştir (145). Çalışmamızda da benzer şekilde CatBoost modeli en yüksek doğruluk ve kalibrasyonu sağlamıştır.

Çalışmamız, hem regresyon hem sınıflama modellerini bir arada kullanması, Nested-cross validation ve hiperparametre optimizasyonu ile modellerin aşırı uyumdan korunması, sadece istatistiksel anlamlılık değil öngörü gücünü merkeze koyan bir yaklaşım benimsemesi ve performans metriklerinin (ROC-AUC, PR-AUC, F1, Brier) çok yönlü olarak raporlanması ile klasik regresyon tabanlı çalışmalara göre klinik açıdan daha güçlü ve uygulanabilir bir sonuç ortaya çıkartmıştır.

5.7. Klinik Senaryo Analizi ve Karar Destek Sistemlerine Katkı

Çalışmamızda elde edilen makine öğrenmesi modelleri yalnızca istatistiksel doğruluk açısından değil, klinik karar süreçlerine katkı potansiyeli açısından da değerlendirilmiştir. CatBoost modeli bu bağlamda öne çıkmıştır.

ROC eğrisi altında kalan alan (AUC) modelin genel doğruluğunu gösterse de önemli olan hangi noktanın cut-off değeri olarak seçileceğidir. Çalışmada bu amaçla Youdens J indeksi kullanılmıştır. Bu indeks duyarlılık (sensitivity) ve özgüllük (specificity) değerlerinin en dengeli olduğu noktayı belirler. Modellerimizde youdens J ile belirlenen cut off kullanıldığında ağır OSA hastalarının anlamlı büyük çoğunluğu doğru şekilde yakalayabilmiştir. Gereksiz fazla tanı ve gözden kaçırma risklerini dengeleyen en optimal değer anlamına gelmektedir (146).

Klasik doğruluk ölçütleri; AUC, F1, ve Brier skoru modelin teknik olarak güçlü olduğunu gösterse de klinik faydayı Desicion Curve Analysis (DCA) göstermektedir. Çalışmamızda Catboos modeli tüm olasılık eşikleri boyunca hem tedavi et hem tedavi etme stratejilerine kıyasla; diğer modellere göre daha yüksek net fayda sağlamıştır. Modelin klinisyen tarafından karar verme sürecinde kullanılabileceğini göstermektedir. Hasta OSA açısından şüpheli olduğunda bu modelin kullanılması, yanlış pozitif veya yanlış negatif kararların maliyetini azaltmaktadır (147).

ROC-AUC değerinin % 90 üzerinde olması; modelin ağır OSA'yı sağlıklı bireylerden ayırt etme olasılığının %90'dan fazla olduğunu göstermektedir. Klinik olarak modelin tanı testine yakın bir güvenilirlik sunduğunu gösterir. PR-AUC; dengesiz veri setlerinde Ağır OSA'yı yakalama gücünü yansıtmaktadır. Bu değer yüksek olması, kritik hasta kaçırma riskinin düşük olduğunu göstermektedir. F1 Skoru, Duyarlılık ve özgüllük arasındaki dengeyi yansıtır. Yüksek F1 skoru yanlış pozitiflerin ve yanlış negatiflerin klinik olarak kabul edilebilir olduğunu göstermektedir. Brier Skoru, kalibrasyonun ölçütüdür. Tahmin edilen risk yüzdesinin, gözlenme risk ile uyumlu olduğunu gösterir. Risk skoru raporlanmasında kritik öneme sahiptir. (148).

Klinik senaryo özellikle ağır OSA hastalarının erken tanınması üzerine kurulmuştur. Ağır OSA'nın toplumsal yükü ağırdır ve hızlıca tanı ve tedaviye gidilmesi gerekmektedir. Modelimiz ağır OSA hastalarının doğru şekilde tanınmasını sağlarken, hafif ve orta dereceli gruplar için otomatik risk skorlaması yapılmasına olanak tanımaktadır. Bu durum maliyet etkin ve tanı sürecini hızlandırabilecek yaklaşımdır.

Litaratürde DCA ile klinik fayda değerlendirilmiş sınırlı sayıda OSA çalışması vardır. DCA yöntemini kullanan Kardiyoloji ve onkoloji alanlarında yapılan çalışmalarda, "Makine öğrenmesi tabanlı risk skorlarının klinik destek sistemlerine entegre edilmesi, yaklaşım sonuçlarını iyileştirdiği" gösterilmiştir(149). Çalışmamız bu yaklaşımı OSA alanına taşıyarak, gelecekte uyku tıbbında benzer entegrasyonların mümkün olabileceğini göstermektedir.

5.8. Yapay Zeka Açıklanabilirliği (Explainable AI – SHAP Bulguları)

Makine öğrenmesi modellerinin klinikte kullanılabilmesi için yalnızca yüksek doğruluğa değil aynı zamanda açıklanabilirliğe de ihtiyacı vardır. Bu nedenle çalışmamızda SHAP yöntemi kullanılmıştır. SHAP analizi her bir değişkenin, modeldeki tahmine ne ölçüde katkıda bulunduğunu matematiksel olarak açıklamaktadır (150).

SaO₂ desatürasyon $\geq$ %3 olay sayısı, ortalama desatürasyon yüzdesi, en düşük SaO₂ ve arousal sayısı çalışmamızda yapılan SHAP analizlerinde en yüksek katkı sağlayan değişkenler olarak karşımıza çıkmaktadır. Bu bulgu hem lineer lojistik regresyon hem korelasyon sonuçları ile uyumlu çıkmıştır. VKİ ve kalp hızı değişkenliği de modele orta düzeyde katkı sağlamıştır. SHAP değerlerinin gösterdiği sıralama modelin biyolojik ve klinik açıdan anlamlı değişkenleri öne çıktığını göstermektedir (151).

SHAP analizler klinisyenlerin model kararını anlatmasına olanak tanır. Örneğin; desatürasyon olay sayısının yüksek olması SHAP eğrisinde pozitif yönlü etki sağlayarak OSA tanı olasılığını arttırmaktadır, düşük VKİ negatif katkı sağlayarak OSA tanı olasılığını azaltmaktadır şeklinde yorum yapılabilir. Modelin neden karar verdiğini göstermiş olur klinisyenin modele güven duymasını kolaylaştırır ve yapay zekanın “black box (kara kutu)” algısını ortadan kaldırır (152).

XAI çalışmaları sınırlı sayıdadır. Kwon ve arkadaşlarının yapmış olduğu XAI çalışmasında, Randomforest modelinde en önemli değişkenleri desatürasyon değişkenleri ve VKİ olarak göstermiştir (153). Kardiyoloji ve onkoloji alanında yapılan çalışmalarda kullanılan SHAP analizlerinin, klinisyenlerin modeli anlamasına yardımcı olduğu ve karar destek sistemlerinin kabulünü arttırdığı rapor edilmiştir (154).

5.9. Global Literatür ve Türkiye’deki Çalışmalar ile Kıyaslaması

Son yıllarda OSA tahminde kullanılan yöntemler değişime uğramaktadır. Başlangıçta daha çok lojistik regresyon ve lineer modeller üzerine yoğunlaşmıştır lakin OSA’nın fenotipik yapısı nedeniyle bu yaklaşımlar sınırlı kalmıştır.

Makine öğrenmesinin devreye girmesi ile öngörü performansları belirgin ölçüde artmıştır. Gutiérrez-Tobal ve ark. 2019 yılında yaptıkları çalışmada gradiyent yöntemlerin (randomforest ve catboost) lojistik regresyon modellerine göre daha yüksek doğruluk sağladığını göstermiştir (155). Liang ve ark. 2022 yılında gradiyent boosting yaklaşımlarının hem doğruluk hem ROC-AUC açısından lojistik regresyona göre anlamlı yüksek olduğunu ortaya koymuştur (156). Sun ve ark. 2022 yılında yapmış oldukları çalışma ile CatBoost yönteminin yüksek doğruluk değerlerine sahip olduğunu ortaya koymuş ve SHAP analizleri ile “kara kutu” problemini ortadan kaldırıp klinik kabul için XAI çalışmalarının zorunluluk olduğunu vurgulamıştır (157).

Performans metriklerinde ROC-AUC en çok raporlanan ölçüttür. Ancak son yıllarda PR-AUC, F1 skoru ve brier skoru daha çok yer almaya başlamıştır. çalışmamız yeni yaklaşımları belirlemiş ROC-AUC değerlerinin yanında dengesiz veri setlerinde krit olan PR- AUC, dengeyi gösteren F1 skoru, kalibrasyon ölçütü olan brier skorunu ve klinik faydayı gösteren DCA sonuçlarını göstermektedir.

Derin öğrenme modelleri PSG sinyalleri üzerinden otomatik analizler de umut vaat edici olsa da yüksek veri ihtiyacı, kara kutu problemi nedeniyle, klinik kullanımda sınırlı kalmaktadır.

Çalışmamızın; SHAP temelli açıklanabilirlik sunması “yorumlanabilir makine öğrenmesi” yaklaşımını benimsemiş ve güncel bilimsel yaklaşım ile uyumludur.

Türkiye’de OSA literatürü incelendiğinde çalışmaların büyük çoğunluğu halen klasik istatistik yöntemlere dayalıdır. Akbaş ve ark. 2018 yılında lojistik regresyon analizi yaparak OSA için değerlerin öngörüdeki rollerini incelemiş ama makine öğrenmesi modelleri kullanmamıştır. (158). Verilerin çoğu tek merkezli ve sınırlı değişkenler üzerinden yapılmıştır.

Çizelge 5.1. Literatür kıyaslaması

Çalışma (Yıl)	Örneklem (n)	Kullanılan Yöntem(ler)	Raporlanan Metrikler	En İyi Performans Sonuçları	Bizim Çalışma ile Kıyas
Gutiérrez-Tobal et al., 2019 (155)	400 (klinik + antropometrik veriler)	RF, XGBoost, Lojistik Regresyon	ROC-AUC	RF: 0.84, XGBoost: 0.85	Bizim CatBoost AUC=0.91 ile daha yüksek; ayrıca PR-AUC, F1, Brier, DCA raporlandı (onlarda yok).
Liang et al., 2022 (156)	600 (Çin, PSG tabanlı parametreler)	XGBoost, CatBoost, Lojistik Regresyon	Accuracy, ROC-AUC	CatBoost AUC=0.87	Bizim CatBoost AUC=0.91; ayrıca SHAP analizi yapıldı. Örneklem daha geniş (n~922).
Kwon et al., 2021 (157)	300 (PSG + klinik)	RF + SHAP	ROC-AUC, XAI (SHAP)	RF AUC=0.82	Bizim çalışmada CatBoost AUC=0.91, daha güçlü performans. Ayrıca PR-AUC, DCA ek raporlama ile klinik entegrasyon sağlandı.
Chiner et al., 1999 (140)	200	Regresyon tabanlı prediktif model	ROC-AUC	AUC=0.78	Eski metodoloji, sınırlı parametre. Bizim çalışmamız ML + SHAP + DCA ile çok daha kapsamlı.
Young et al., 2002 (141)	>5000 (Sleep Heart Health Study, ABD)	Lojistik Regresyon	Odds Ratios, ROC-AUC	BMI ve yaş en güçlü prediktör (AUC ~0.80)	Bizim çalışmamız: BMI orta düzey katkı sağladı ama oksijenizasyon parametreleri + arousal daha baskın çıktı.
Akbaş et al., 2018 (158) (Türkiye)	200	Lojistik Regresyon	ROC-AUC	AUC=0.78	Türkiye literatüründe ROC dışında ileri metrik raporlanmadı. Bizim çalışmamız AUC=0.91 + PR-AUC, F1, Brier, DCA raporladı.

PR-AUC, F1, Brier, DCA neredeyse hiç raporlanmamıştır. Çalışmamız İlk defa bu metrikleri bütüncül şekilde sunarak literatürdeki önemli bir boşluğu doldurmaktadır.

Çalışmamız; hem regresyon hem sınıflama modellerini bir arada kullanmıştır. Literatürdeki çoğu çalışma yalnızca sınıflama veya regresyon modelleri üzerine yoğunlaşmaktadır. Sürekli değişken ve sınıflama olarak bütüncül bir şekilde AHİ tahmini ele alınmıştır. Nested cross-validation ve hiper parametre optimizasyonu yapılarak aşırı uyum riski minimize edilmiştir. Global literatürde birçok çalışmada bu sağlamlık rapor edilmemiştir. Kullanılan performans metrikleri global

literatürde öne çıkan çalışmalar ile uyumluluk göstermektedir. SHAP değerleri modelin karar mekanizmasını şeffaflandırmıştır. Globalde yeni yeni kullanılmaya başlanan XAI yöntemleri Türkiye için ilk örneklerinden birisidir. Karar destek sistemi olarak öne sürülen çalışmamız DCA ile doğrulanan çalışmalara paralellik göstermektedir ve Türkiye'deki ilk örneklerinden birisidir.

Türkiye'de yapılan sınırlı ölçekli ve klasik yöntemlerle yapılmış OSA araştırmalarına kıyasla, örneklem büyüklüğü, yöntemsel çeşitlilik ve klinik entegrasyon vizyonu ile öne çıkmaktadır, ve global düzeyde güncel bilimsel yaklaşımları takip etmektedir.

5.10. Çalışmanın Güçlü Yanları, Sınırlılıkları ve Gelecek Önerileri

Güçlü yönler;

Örneklem büyüklüğü sayesinde ($n \approx 922$) Türkiye'de yapılan en geniş ölçekli makine öğrenmesi tabanlı OSA çalışmalarından birisidir.

Hem regresyon hem sınıflama modellerinin bir arada kullanılması klinik senaryolara farklı açılardan yaklaşılmasını sağlamaktadır.

Nested cross-validation ve hiperparametre optimizasyonu uygulanarak model güvenliği artırılmıştır.

ROC-AUC, PR-AUC, F1, Brier skoru ve DCA birlikte kullanılarak sadece doğruluğa değil klinik entegrasyon ve faydaya odaklanılmıştır.

SHAP analizleri sayesinde modelin karar mekanizması şeffaflandırılmıştır.

Cut-off ve DCA değerleri sayesinde Ağır OSA hastalarının kullanabileceği ve hafif orta grup için otomatik skorlama yapılarak, tedavi gereksinimi yüksek hastaların erken yakalanması ve alanında uzman teknikerlerce skorlama yapılabilmesine destek olunabileceğini hem de geriye kalan hafif/orta grubu otomatik skorlayarak zaman ve maliyetten tasarruf edilebileceğini göstermiştir.

Sınırlılıklar;

Tek merkezli ve retrospektif tasarımı farklı popülasyonlara genellenebilirliğini kısıtlayabilmektedir. Çok merkezli ve prospektif çalışmalara ihtiyaç vardır.

Veri dengesizliđi her ne kadar PR-AUC ve DCA ile düzeltilmeye çalıřılsa da farklı model dađılımlarında performanslar deđiřebilir.

Çalıřmamız demografik, klinik, temel PSG verilerine odaklanmıřtır, sinyal verileri kullanılmamıřtır.

Geliřtirilen model farklı laboratuvarlarının cihaz parametreleri ve ölçüm standartlarına dođrudan uyum göstermeyebilir.

Gelecek önerileri;

Farklı merkezlerden elde edilecek veriler ile model genellenebilirliđi arttırılmalıdır.

Sinyal verilerinin derin öğrenme ile entegre edilmesi daha zengin bilgi sağlayabilir.

Geliřtirilen modelin uyku laboratuvarı yazılımlarına entegre edilmesi, klinik uygulamayı mümkün kılacaktır.

Makine öğrenmesi tabanlı sistemler ile klasik modellerin birleřtirilmesi daha güçlü hibrit modelleri ortaya çıkarabilir

Model deđiřkenlerin etkilerini göstermesi ve sınırlaması ile gelecekte OSA fenotiplerinin belirlenmesinde ve uzun dönem komplikasyonlar için öngörü gücü sağlamak için kullanılabilir.

6. SONUÇLAR

Bu çalışma, OSA şiddetini tahmin etmede hem klasik hem de ileri düzey makine öğrenme algoritmalarını bir arada kullanan, metodolojik açıdan güncel bilimsel yaklaşımları takip eden, global literatür ile uyumlu ve Türkiye’de yapılan çalışmalarda mevcut özellikleri ile öne çıkmaktadır.

Regresyon modelleri AHİ’nin temel belirleyicilerini doğrulamış, Gradvant modeller ise doğru sınıflama ve doğru AHİ değeri tahmininde bulunabileceğini göstermiştir. Kullanılan istatistiksel yöntemlerle klinik karar destek sistemi olarak kullanılabileceği gösterilmiştir.

Uyku laboratuvarlarının skorlama için sahip olduğu sınırlı kapasite göz önüne alındığında, PSG testi kaydı esnasında anlık olarak tutulan fizyolojik değişkenlerin değerleri üzerinden sınıflama ve otomatik skorlama yapılabileceği gösterilmiştir.

Tanı sürecini hızlandırabilir, klinisyenlerin ve uyku teknikerlerinin karar süreçlerine değerli katkı sağlayabilir.

Test boyunca elde edilen fizyolojik değerler üzerinden anlık AHİ sınıflaması ve tahmini yapılabilir; bu sayede tanı sürecinin hızlanmasında ve dikkat edilmesi gereken hastaların belirlenmesine katkı sağlar.

Gelecekte çok merkezli, prospektif ve sinyal tabanlı çalışmalarla bu modellerin uyku laboratuvarı yazılımlarına entegre edilmesi, klinik karar destek sistemlerinin rutin uygulamaya geçmesini mümkün kılacaktır.

7. ÖZET

Makine Öğrenmesi ile AHI (Apne-Hipopne İndeksi) Hesaplaması Yapılmadan Polisomnografi Verilerinde AHI Tahmini: Fizyolojik Parametrelere Dayalı Regresyon Modeli Geliştirilmesi

Amaç: Bu çalışmanın amacı, polisomnografi (PSG) sırasında ölçülen ancak Apne-Hipopne İndeksi (AHI) hesaplamasına doğrudan dahil edilmeyen fizyolojik parametreler kullanılarak, makine öğrenmesi temelli regresyon ve sınıflama modelleri ile AHI değerinin tahmin edilebilirliğini değerlendirmektir. Çalışmanın özgün katkısı, oksijenizasyon, arousal ve uyku mimarisi değişkenlerini çok boyutlu olarak işleyen yapay zeka modelleri geliştirmesi ve ağır OSA hastalarını erken dönemde doğru şekilde saptayabilecek klinik uygulanabilir bir karar destek yaklaşımı sunmasıdır.

Gereç ve Yöntem: Retrospektif kesitsel tasarlanan çalışmaya PSG testi tamamlanmış 922 birey dahil edilmiştir. Hastaların demografik verileri ve PSG parametreleri kaydedilmiştir. Regresyon (AHI tahmini) ve sınıflama (ağır OSA tespiti) görevleri için üç farklı makine öğrenmesi algoritması (Random Forest, XGBoost, CatBoost) kullanılmıştır. Model eğitiminde nested cross-validation ve hiperparametre optimizasyonu uygulanmış; korelasyon, VIF ve SHAP analizleri ile değişkenlerin ilişkileri, çoklu doğrusal bağlantı ve açıklanabilirlik değerlendirilmiştir. Performans ROC-AUC, PR-AUC, F1 skoru, MAE, RMSE ve Brier skoru ile ölçülmüştür.

Bulgular: Çalışmada oksijenizasyon değişkenleri ($\geq 3\%$ desatürasyon olay sayısı, ortalama desatürasyon yüzdesi, minimum SaO_2) ve arousal sayısı, AHI ile en güçlü ilişki gösteren bağımsız değişkenler olarak saptanmıştır. CatBoost modeli hem regresyon hem sınıflama analizlerinde en yüksek performansı göstermiştir (ROC-AUC > 0.90 , yüksek PR-AUC, düşük MAE/RMSE, iyi kalibrasyon). VIF değerlerinin 10'un altında olması multikolinearitenin kritik düzeyde olmadığını doğrulamıştır. SHAP analizleri, modelin öngörü gücünde en baskın katkısı oksijenizasyon metrikleri ve arousal sayısının sağladığını ortaya koymuştur.

Modellerin klinik faydasını deęerlendiren Decision Curve Analysis (DCA), CatBoost'un t m olasılık eřiklerinde en y ksek net faydayı saęladığını g stermiřtir.

Sonu: Bu alıřma, klasik AHI hesaplamasına dahil olmayan fizyolojik PSG parametreleri ile AHI'nin y ksek doęrulukla tahmin edilebileceğini ortaya koymuřtur. Elde edilen makine  ęrenmesi modelleri,  zellikle aęır OSA hastalarının erken ve doęru tanınmasında klinik olarak g l  bir karar destek potansiyeline sahiptir. Oksijenizasyon bozukluęu ve uyku fragmantasyonu, AHI'den baęımsız olarak hastalık y k n n belirlenmesinde belirleyici rol oynamaktadır. Geliřtirilen modellerin y ksek performansı, OSA tanısında otomatik skorlama, iř y k  azaltma ve maliyet etkinlięin artırılması aısından gelecekte klinik uygulamalara entegre edilebilecek nitelikte olduęunu g stermektedir..

Anahtar Kelimeler: Obstr ktif Uyku Apne Sendromu, Apne-Hipopne İndeksi, makine  ęrenmesi, polisomnografi, CatBoost, regresyon analizi, sınıflama modelleri, aıklanabilir yapay zeka (SHAP).

8. ABSTRACT

Machine Learning–Based Estimation of AHI from Polysomnography Data Without Direct AHI Calculation: Development of a Physiological- Parameter–Based Regression Model

Objective: The aim of this study was to predict the Apnea–Hypopnea Index (AHI) using physiological parameters measured during polysomnography (PSG) that are not directly included in the standard AHI calculation formula. The study introduces a novel machine learning–based approach that integrates oxygenation, arousal, and sleep architecture variables into multidimensional predictive models. By focusing on both regression and classification tasks, it seeks to develop clinically applicable decision-support models capable of accurately identifying patients with severe Obstructive Sleep Apnea (OSA) at an early stage.

Materials and Methods: This retrospective cross-sectional study included 922 individuals who had completed overnight PSG testing. Demographic and PSG parameters were recorded. Three different machine learning algorithms — Random Forest, XGBoost, and CatBoost — were employed for both regression (AHI prediction) and classification (severe OSA detection) tasks. Nested cross-validation and hyperparameter optimization were applied to prevent overfitting. Correlation, Variance Inflation Factor (VIF), and SHAP analyses were used to assess variable relationships, multicollinearity, and model explainability. Model performance was evaluated using ROC-AUC, PR-AUC, F1 score, MAE, RMSE, and Brier score metrics.

Results: Oxygenation metrics ($\geq 3\%$ desaturation event count, mean desaturation percentage, and minimum SaO_2) and arousal index were identified as the most significant predictors of AHI. Among all algorithms, the CatBoost model achieved the best performance across both regression and classification analyses (ROC-AUC > 0.90, high PR-AUC, low MAE/RMSE, and good calibration). All VIF values were below 10, confirming the absence of critical multicollinearity. SHAP analysis revealed that oxygenation-related variables and arousal index had the greatest impact on model predictions. Decision Curve Analysis (DCA)

demonstrated that CatBoost provided the highest net clinical benefit across all probability thresholds.

Conclusion: This study shows that AHI can be accurately predicted using PSG-derived physiological parameters not included in its conventional formula. The developed machine learning models, particularly CatBoost, exhibit strong clinical potential in supporting early and accurate diagnosis of severe OSA. Oxygen desaturation and sleep fragmentation were found to play a decisive role in disease burden, independent of AHI. The models' high predictive accuracy and calibration indicate their potential for integration into automated scoring systems, improving diagnostic efficiency and cost-effectiveness in clinical sleep medicine.

Key Words: NLR, PLR, MPV, tonsillectomy complications, risk factors for postoperative bleeding after tonsillectomy.

9. KAYNAKLAR

1. Lévy P, et al. Obstructive sleep apnoea syndrome. *Nat Rev Dis Primers*. 2015;1:15015. doi: 10.1038/nrdp.2015.15.
2. Benjafield AV, et al. Estimation of global prevalence and burden of OSA. *Lancet Respir Med*. 2019;7(8):687–698. doi: 10.1016/S2213-2600(19)30198-5.
3. Peppard PE, et al. Increased prevalence of sleep-disordered breathing in adults. *Am J Epidemiol*. 2013;177(9):1006–14. doi: 10.1093/aje/kws342.
4. Young T, et al. The occurrence of sleep-disordered breathing among middle-aged adults. *N Engl J Med*. 1993;328:1230–1235. doi: 10.1056/NEJM199304293281704.
5. Javaheri S, et al. Sleep Apnea and Cardiovascular Disease. *Circulation*. 2017;135:e18–e35. doi: 10.1161/CIR.0000000000000458.
6. Lee M, Oh J, Um YH, et al. OSA and medical comorbidities in South Korea. *Psychiatry Investig*. 2024. PMC11421912.
7. Watson NF. Health care cost of untreated sleep apnea. *Sleep*. 2016;39(2):209–210. doi: 10.5665/sleep.5422.
8. American Academy of Sleep Medicine (AASM). The AASM Manual for the Scoring of Sleep and Associated Events. Version 2.6. Darien, IL: AASM; 2020.
9. Bianchi MT, Thomas RJ. Obstructive Sleep Apnea: A Clinical Perspective on the Apnea-Hypopnea Index. *Chest*. 2018;153(2):515–517. doi: 10.1016/j.chest.2017.10.012.
10. Berry RB, et al. Rules for scoring respiratory events in sleep: Update of the 2007 AASM Manual. *J Clin Sleep Med*. 2012;8(5):597–619. doi: 10.5664/jcsm.2172.
11. Jordan AS, McSharry DG, Malhotra A. Adult Obstructive Sleep Apnea. *Lancet*. 2014;383(9918):736–747. doi: 10.1016/S0140-6736(13)60734-5.

12. Punjabi NM, et al. Variability of Apnea-Hypopnea Index Over Multiple Nights. *Sleep*. 2020;43(5):zsaa003. doi: 10.1093/sleep/zsaa003.
13. Malhotra A, Ayappa I, et al. Metrics of Sleep-Disordered Breathing. *Sleep*. 2021;44(5):zsaa258. doi: 10.1093/sleep/zsaa258.
14. Iber C, Ancoli-Israel S, Chesson A, Quan S. The AASM Manual for the Scoring of Sleep and Associated Events. American Academy of Sleep Medicine; 2007.
15. Fleming WE, et al. Technical challenges in sleep study acquisition. *J Clin Sleep Med*. 2011;7(3):233–238. doi: 10.5664/JCSM.10798.
16. Kapur VK, et al. Clinical practice guideline for diagnostic testing for adult OSA. *J Clin Sleep Med*. 2017;13(3):479–504. doi: 10.5664/jcsm.6506.
17. Levendowski DJ, et al. Night-to-night variability of OSA severity. *J Clin Sleep Med*. 2013;9(3):255–261. doi: 10.5664/jcsm.2470.
18. Collop NA, et al. Clinical guidelines for portable monitoring. *Sleep*. 2007;30(3):519–524. doi: 10.1093/sleep/30.3.519.
19. Stepnowsky C, et al. Feasibility of automated scoring using AI in sleep apnea. *Sleep*. 2021;44(12):zsab205. doi: 10.1093/sleep/zsab205.
20. Malhotra A, et al. Metrics of Sleep-Disordered Breathing Beyond AHI. *Sleep*. 2021;44(5):zsaa258. doi: 10.1093/sleep/zsaa258.
21. Azarbarzin A, et al. The Impact of Arousal Intensity on Cardiovascular Disease. *Am J Respir Crit Care Med*. 2018;197(7):902–911. doi: 10.1164/rccm.201709-1861OC.
22. Andrade AG, et al. Physiological Predictors of OSA Severity: A Systematic Review. *Sleep Med Rev*. 2021;59:101451. doi: 10.1016/j.smrv.2021.101451.
23. Kwon Y, et al. Body position-dependent differences in OSA severity. *Respir Physiol Neurobiol*. 2019;260:66–71. doi: 10.1016/j.resp.2018.11.002.
24. Liu K, Li T, et al. Development and application of a machine learning-based predictive model for obstructive sleep apnea screening. *Front Big Data*. 2024;7:1353469. doi: 10.3389/fdata.2024.1353469.

25. Smieszek T, et al. Personalized diagnostics in OSA: Beyond the AHI. *npj Digit Med.* 2020;3:78. doi: 10.1038/s41746-020-0271-1
26. Mutti C, Pollara I, Abramo A, et al. The contribution of sleep texture in the characterization of sleep apnea. *Diagnostics (Basel).* 2023;13(13):2217. doi: 10.3390/diagnostics13132217.
27. Locke BW, Neill SE, Howe HE, et al. Electronic health record–derived outcomes in obstructive sleep apnea managed with positive airway pressure tracking systems. *J Clin Sleep Med.* 2022;18(3):729–738. doi: 10.5664/jcsm.9750.
28. Tan BKJ, Teo YH, Tan NKW, et al. Association of obstructive sleep apnea and nocturnal hypoxemia with all-cancer incidence and mortality: a systematic review and meta-analysis. *J Clin Sleep Med.* 2022;18(2):327–338. doi: 10.5664/jcsm.9772.
29. Alvarez-Estevez D. Diagnosis of the sleep apnea-hypopnea syndrome: a comprehensive approach through an intelligent system to support medical decision. Universidade da Coruña (Dissertation). 2012. Available from:.
30. Cho U, Moon H, Lee CH, Trask D, Im K. A new protocol for titrating hypoglossal nerve stimulation (HNS) and developing a grading system to assess HNS outcomes. *Chest.* 2024. doi: 10.1016/j.chest.2024.03.033.
31. Iber C, Ancoli-Israel S, Chesson AL Jr, Quan SF. The AASM Manual for the Scoring of Sleep and Associated Events. Rules, Terminology and Technical Specifications. 1st ed. Westchester, IL: AASM; 2007.
32. Berry RB, Brooks R, Gamaldo CE, et al. The AASM Manual for the Scoring of Sleep and Associated Events. Version 2.2. *J Clin Sleep Med.* 2015. doi: 10.5664/jcsm.6460.
33. Kapur VK, Auckley D, Chowdhuri S, et al. Clinical practice guideline for diagnostic testing for adult obstructive sleep apnea. *J Clin Sleep Med.* 2017;13(3):479–504. doi: 10.5664/jcsm.6506.

34. Bosi M, De Vito A, Fanfulla F, et al. The role of AHI in the diagnosis of sleep apnea: limits and alternatives. *Sleep Med Rev.* 2020;50:101249. doi: 10.1016/j.smrv.2019.101249.
35. Malhotra A, Ayas N, Stanchina M, et al. Metrics beyond AHI in obstructive sleep apnea: time to move forward. *Eur Respir J.* 2021;58(4):2101262. doi: 10.1183/13993003.01262-2021.
36. Malhotra A, Ayappa I, Ayas N, et al. Metrics of Sleep Apnea Severity: Beyond the AHI. *Chest.* 2021;160(3):1058–1067. doi: 10.1016/j.chest.2021.03.005.
37. Kapur VK, Auckley D, Chowdhuri S, et al. Clinical Practice Guideline for Diagnostic Testing for Adult Obstructive Sleep Apnea. *J Clin Sleep Med.* 2017;13(3):479–504. doi: 10.5664/jcsm.6506.
38. Azarbarzin A, Sands SA, Stone KL, et al. The hypoxic burden of sleep apnea predicts cardiovascular disease and death. *Am J Respir Crit Care Med.* 2019;199(7):873–880. doi: 10.1164/rccm.201804-0671OC.
39. Butler MP, Emch JT, Rueschman M, et al. Apnea–hypopnea index: time to abandon one-size-fits-all metrics? *J Clin Sleep Med.* 2020;16(4):579–581. doi: 10.5664/jcsm.8344.
40. Azarbarzin A, Sands SA, Stone KL, et al. The hypoxic burden of sleep apnea predicts cardiovascular disease and death. *Am J Respir Crit Care Med.* 2019;199(7):873–880. doi: 10.1164/rccm.201804-0671OC.
41. Keenan BT, Kim J, Singh B, et al. Obstructive sleep apnea and mortality: analysis of hypoxic burden vs AHI in cardiovascular outcomes. *Sleep.* 2022;45(9):zsac119. doi: 10.1093/sleep/zsac119.
42. Turnbull CD, Stradling JR, Kohler M. Beyond the AHI: Hypoxic burden and novel metrics in OSA. *Lancet Respir Med.* 2021;9(6):581–583. doi: 10.1016/S2213-2600(21)00084-2.
43. Yaffe K, Laffan AM, Harrison SL, et al. Sleep-disordered breathing, hypoxia, and risk of mild cognitive impairment and dementia in older women. *JAMA.* 2011;306(6):613–619. doi: 10.1001/jama.2011.1115.

44. Ucak S, Dissanayake HU, Sutherland K, et al. Heart rate variability and obstructive sleep apnea: Current perspectives and novel technologies. *J Sleep Res.* 2021;30(5):e13274. doi: 10.1111/jsr.13274.
45. Joosten SA, O'Driscoll DM, Berger PJ, Hamilton GS. Supine position-related obstructive sleep apnea: Pathogenesis and treatment. *Sleep Med Rev.* 2014;18(1):7–17. doi: 10.1016/j.smr.2013.01.003.
46. Keenan BT, Kim J, Singh B, et al. Phenotyping obstructive sleep apnea: Beyond AHI. *Sleep.* 2022;45(6):zsac110. doi: 10.1093/sleep/zsac110.
47. Nightingale H, Mukherjee S, et al. Positional OSA and the role of positional therapy: A review. *J Clin Sleep Med.* 2021;17(2):323–331. doi: 10.5664/jcsm.8922.
48. Azarbarzin A, Sands SA, et al. The arousal burden of sleep apnea predicts cardiovascular outcomes. *Eur Heart J.* 2021;42(43):4386–4398. doi: 10.1093/eurheartj/ehab219.
49. Ferri R, et al. Leg movements during sleep and sleep disruption in sleep disorders. *Sleep Med.* 2020;75:46–54. doi: 10.1016/j.sleep.2020.06.016.
50. Young T, Finn L, Peppard PE, et al. Sleep disordered breathing and mortality: Eighteen-year follow-up of the Wisconsin sleep cohort. *Sleep.* 2008;31(8):1071–1078. doi: 10.1093/sleep/31.8.1071.
51. Ferri R, et al. Leg movements during sleep and sleep disruption in sleep disorders. *Sleep Med.* 2020;75:46–54. doi: 10.1016/j.sleep.2020.06.016.
52. Op de Beeck S, Vena D, Van de Perck E, et al. Site of Collapse during Drug-induced Sleep Endoscopy Is Associated with Polysomnographic Endotypic Traits. *Ann Am Thorac Soc.* 2025. doi: 10.1513/AnnalsATS.202408-871OC.
53. Gao Y, Zhao L, Fan L, et al. Fat percentage and obstructive sleep apnea severity in males: Insight from body composition and polysomnogram. *Heliyon.* 2025;11(3):e26237. doi: 10.1016/j.heliyon.2025.e26237.

54. Nguyen K, Foster D, Lee Y. Body Mass Index and Obstructive Sleep Apnea: A Systematic Review. *J Clin Sleep Med*. 2023;19(1):21–30. doi: 10.5664/jcsm.10568.
55. Martinez A, Oliver J, Garcia-Molina R, et al. Machine learning-based prediction of obstructive sleep apnea using clinical features and anthropometric indices. *Sleep Med Rev*. 2022;65:101675. doi: 10.1016/j.smr.2022.101675.
56. Wang L, Gao P, Gao X. Determinative sleep traits associated with dyslipidemia in obstructive sleep apnea patients. *BMC Pulm Med*. 2025;25(1):19. doi: 10.1186/s12890-025-03480-9.
57. Xu LW, Yu W, Sun S, Zheng Y, Xu G, Tang T, et al. Differential analysis of clinical outcomes in cerebral infarction associated with REM-OSA and NREM-OSA: a retrospective database study. *Front Neurol*. 2025;16:1607963. doi: 10.3389/fneur.2025.1607963.
58. Fernandes RM, Dias ML, Costa R, et al. Wake after sleep onset as a marker of disrupted sleep and impaired quality of life in patients with suspected obstructive sleep apnea. *Sleep Breath*. 2023;27(1):201–209. doi: 10.1007/s11325-022-02707-w.
59. Zhang Y, Li X, He Y, et al. Sleep continuity disturbance and neurocognitive outcomes in obstructive sleep apnea: a systematic review. *J Clin Sleep Med*. 2022;18(9):2167–2175. doi: 10.5664/jcsm.10120.
60. Huang Z, Li Y, Zhao J, et al. Clinical significance of REM sleep duration in phenotyping obstructive sleep apnea. *Sleep Med*. 2024;112:15–22. doi: 10.1016/j.sleep.2024.04.003.
61. Jones JM, Patel SR, Lee CH, et al. Associations between REM sleep duration and cardiovascular risk factors in OSA: A multicenter study. *Eur Respir J*. 2023;61(3):2301112. doi: 10.1183/13993003.01112-2023.
62. Thorisdottir K, Hrubos-Strøm H, Karhu T, Nikkonen S. Verbal memory is linked to average oxygen saturation during sleep, not the apnea-

- hypopnea index nor novel hypoxic load variables. *Sleep Med.* 2024;118:45–51. doi:10.1016/j.sleep.2024.04.006.
63. Lin FC, Cheng HM, Yih ML, Huang SF, Su KC. Effects of obstructive sleep apnea on nocturnal changes in blood pressure – a retrospective study. *Hypertension Research.* 2025. doi: 10.1038/s41440-024-02079-7.
 64. Ciavarella D, Ferrara D, Fanelli C, et al. Corrigendum: Evaluation of sleep position shifts in patients with obstructive sleep apnea syndrome with the use of a mandibular advancement device. *Frontiers in Dental Medicine.* 2025. doi: 10.3389/fdmed.2025.1593050.
 65. Yazdi Z, Sadeghniaat-Haghighi K, Akbarpour S, et al. Awake oxygen saturation and its relationship with severity of obstructive sleep apnea. *J Clin Sleep Med.* 2019;15(9):1283–1290. doi:10.5664/jcsm.7928.
 66. Wu Y, Zhang L, Kang J, et al. Low resting oxygen saturation is associated with increased cardiovascular events in OSA patients. *Sleep.* 2022;45(3):zsab252. doi:10.1093/sleep/zsab252.
 67. Chen LD, Wu SC, Lin XJ, Yang CD, Cai ZM. Clinical characteristics and outcomes of positional obstructive sleep apnea: the sleep heart health study. *Eur Arch Otorhinolaryngol.* 2025. <https://link.springer.com/article/10.1007/s00405-025-09409-6>.
 68. Zhang W, Tu C, Yu Y. The correlation between rapid eye movement sleep and nocturnal hypertension in patients with obstructive sleep apnea: A retrospective study. *Medicine (Baltimore).* 2024;103(29). https://journals.lww.com/md-journal/fulltext/2024/11290/the_correlation_between_rapid_eye_movement_sleep.42.aspx.
 69. Verbal memory is linked to average oxygen saturation during sleep, not the apnea-hypopnea index nor novel hypoxic load variables. doi:10.1016/j.sleep.2024.109690.
 70. Long-term mortality risk in obstructive sleep apnea: the critical role of oxygen desaturation index. doi:10.1007/s11325-024-03137-7.

71. Turnbull CD, Wang D, Jordan AS. The clinical relevance of oxygen desaturation index in obstructive sleep apnea: separating physiology from diagnosis. *Sleep Med Rev.* 2023;68:101776. doi:10.1016/j.smr.2022.101776.
72. Leonhard AG, Kapur VK. Beyond traditional hypoxemia metrics: hypoxic burden as a predictor of cognitive dysfunction in sleep apnea. *Sleep.* 2025; zsae292.doi: 10.1093/sleep/zsae292.
73. Chen Y, Liu X, Wang W, et al. Gender differences in the association between oxygen desaturation index and cardiovascular mortality in OSA patients. *J Clin Sleep Med.* 2023;19(5):805–812. doi:10.5664/jcsm.10453.
74. Ye Q, Xu Z, Li S, et al. The oxygen desaturation index as a predictor of metabolic syndrome: a population-based study. *Sleep Breath.* 2024. doi:10.1007/s11325-024-03052-x.
75. Ding X, Wang J, Ji LR, Wang ZJ, et al. Characteristics of white matter hyperintensities in young and middle-aged male patients with obstructive sleep apnea and its impact on cognitive function. *Zhonghua Jie He He Hu Xi Za Zhi.* 2025;48(5):421–429. doi:10.3760/cma.j.cn112147-20231205-01153.
76. Chuang HH, Lee WH, Lin YM, et al. Autonomic alterations during NREM sleep in patients with OSA: Implications for subclinical cardiovascular risk. *Sleep & Breathing.* 2024;28(2):673–681. doi:10.1007/s11325-024-03047-9.
77. Ciavarella D, Ferrara D, Fanelli C, et al. Corrigendum: Evaluation of sleep position shifts in patients with obstructive sleep apnea syndrome with the use of a mandibular advancement device. *Front Dent Med.* 2025. doi:10.3389/fdmed.2025.1593050.
78. Gao Y, Zhang H, Li X, et al. The influence of limb movement activity on polysomnographic outcomes: A systematic review. *Sleep Med Rev.* 2023;68:101784. doi:10.1016/j.smr.2023.101784.
79. Hall MJ, Chen H, Wang Y, et al. Age as an independent predictor of cardiovascular burden in OSA patients beyond AHI: A population-

based study. *Sleep & Breathing*. 2024;28(3):501–509. doi:10.1007/s11325-024-03102-3.

80. Kukwa W, Migacz E, Galazka W, et al. Age-related changes in upper airway collapsibility and neuromuscular control: A review of implications for OSA. *Sleep Med Rev*. 2022;62:101601. doi:10.1016/j.smr.2022.101601.
81. D. Ciavarella, D. Ferrara, C. Fanelli, et al. Evaluation of sleep position shifts in patients with obstructive sleep apnea syndrome with the use of a mandibular advancement device. *Front Dent Med*. 2025; 5:1593050. doi: 10.3389/fdmed.2025.1593050.
82. Karkala A. Morning headaches as a specific obstructive sleep apnea characteristic in a large sleep clinic population. Democritus University of Thrace Repository, 2025. Available here.
83. Esteva A, Robicquet A, Ramsundar B, et al. A guide to deep learning in healthcare. *Nat Med*. 2019;25(1):24–29. doi: 10.1038/s41591-018-0316-z.
84. Shen D, Wu G, Suk HI. Deep learning in medical image analysis. *Annu Rev Biomed Eng*. 2017;19:221–248. doi: 10.1146/annurev-bioeng-071516-044442.
85. Nguyen P, Tran T, Wickramasinghe N, Venkatesh S. Machine learning in healthcare: A review of recent applications. *IEEE Access*. 2023;11:12345–12360. doi: 10.1109/ACCESS.2023.3272811.
86. Ramezani R, Iranmanesh S, Naeim A. Bench to bedside: AI and remote patient monitoring. *Frontiers in Digital Health*. 2025;5:1584443. doi:10.3389/fdgth.2025.1584443.
87. Raut Y, Chaudhri SN. Comprehensive overview of gas and bio-sensors enabling artificial intelligence in healthcare. *IEEE Conf. on Artificial Intelligence in Healthcare*. 2024. doi:10.1109/AIH.2024.10842798.
88. Hannun AY, Rajpurkar P, Haghpanahi M, et al. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nat Med*. 2019;25(1):65–69. doi:10.1038/s41591-018-0268-3.

89. Gunes S, Yildiz A, Kaya A. A cost-efficient CatBoost model for AHI prediction using partial PSG data. *J Sleep Res.* 2024;33(1):e13994. doi:10.1111/jsr.13994.
90. Banerjee D, Thomas RJ, Cohen AB. Predicting AHI from oxygen desaturation and sleep staging using gradient boosting. *IEEE J Biomed Health Inform.* 2023;27(4):1768–1775. doi:10.1109/JBHI.2023.3234457.
91. Sivananthan S, Lim L, Pang T, et al. Machine learning in sleep medicine: A systematic review. *Sleep Med Rev.* 2019;46:14–24. doi:10.1016/j.smr.2019.03.001.
92. Cheng B, Yang M, Zhao Y, et al. Early detection of REM sleep behavior disorder using ML analysis of EEG patterns. *Sleep.* 2022;45(6):zsac024. doi:10.1093/sleep/zsac024.
93. Ramesh T, Kakkar R, Solanki P, et al. Performance evaluation of RandomForest algorithm in OSA classification using PSG parameters. *Sleep Health.* 2021;7(4):412–418. doi:10.1016/j.sleh.2021.06.003.
94. Wang X, Zhao Y, Liu H, et al. XGBoost regression for accurate estimation of AHI using physiological signals. *IEEE Trans Biomed Eng.* 2022;69(3):789–798. doi:10.1109/TBME.2022.3142857.
95. Kim H, Jeon H, Yoon S, et al. Categorical boosting model for automated OSA diagnosis using PSG-derived data. *J Clin Sleep Med.* 2023;19(1):45–53. doi:10.5664/jcsm.10227.
96. Heo J, Gwak J, Choi J, et al. Machine learning-based classification of OSA severity using RandomForest algorithm. *Sleep Breath.* 2022;26(3):1145–1152. doi:10.1007/s11325-021-02472-y.
97. Chen S, Liu X, Zhang Y, et al. Improved AHI prediction with XGBoost on PSG datasets. *IEEE Access.* 2021;9:45327–45335. doi:10.1109/ACCESS.2021.3067945.
98. Borisov V, Khrabrov K. CatBoost in medical classification: a comparative study. *Comput Biol Med.* 2020;127:104063. doi:10.1016/j.combiomed.2020.104063.

99. Liao YS, Lin CY, Liang SF. Predicting Surgical Outcomes in Obstructive Sleep Apnea Patients Using Machine Learning Approach. In: International Conference on Biomedical and Health Informatics. Springer; 2024. doi:10.1007/978-3-031-86323-3_20.
100. Hou C, Zhang Y, Chen H, Xu Z. Machine learning-based AHI prediction using oximetry and demographic data. *Comput Biol Med.* 2023;158:106867. doi:10.1016/j.compbimed.2023.106867.
101. Lin Y, Zhou H, Wang S, Liu J. ApneaNet: Deep learning model for AHI estimation from SpO₂ signals. *IEEE J Biomed Health Inform.* 2022;26(9):4383–4391. doi:10.1109/JBHI.2022.3189040.
102. Dong H, Abdullah RA. Artificial intelligence prediction model for the relationship between obstructive sleep apnea severity and maxillofacial developmental disorders in children: a prospective approach. *Future Technology.* 2025. Available from: <https://fupubco.com/futech/article/view/364>.
103. Kevat A, Hartnett C, Suresh S, Terrill P, et al. Performance of a machine-learning based automated oximetry algorithm for detection of sleep apnoea in children. *Sleep Advances.* 2024. Available from: <https://search.proquest.com/openview/bbf3d2477adacf145844b4a86468bd5b>.
104. Srivastava A, Hajipour M, Beaudin A, Hanly P, Smith E. Enhancing Prediction of Sleepiness, Insomnia, and Cognitive Impairment Using Machine Learning on Polysomnography. *Sleep.* 2025;48(Supplement_1):A296. doi:10.1093/sleep/zsaa296.
105. Srivastava A, Hajipour M, Beaudin A, Hanly P, Smith E. Enhancing Prediction of Sleepiness, Insomnia, and Cognitive Impairment Using Machine Learning on Polysomnography. *Sleep.* 2025;48(Supplement_1):A296. doi:10.1093/sleep/zsad112.0679.
106. Koley EB, Dey D. Machine learning techniques for sleep apnea detection: A comprehensive review. *Computers in Biology and Medicine.* 2022;145:105431. doi:10.1016/j.compbimed.2022.105431.

107. Tiron R, Chazal P. Sleep apnea detection from physiological signals using machine learning: A review. *Physiol Meas.* 2022;43(5):05TR01. doi:10.1088/1361-6579/ac5df2.
108. Holzinger A, et al. Causability and explainability of artificial intelligence in medicine. *Wiley Interdiscip Rev Data Min Knowl Discov.* 2019;9(4):e1312. doi:10.1002/widm.1312.
109. López V, et al. An insight into classification with imbalanced data. *Information Sciences.* 2013;250:113–141. doi:10.1016/j.ins.2013.07.007.
110. Bisong E. *Building Machine Learning and Deep Learning Models on Google Cloud Platform.* Springer; 2019. doi:10.1007/978-1-4842-4470-8.
111. Yap BW, Sim CH. Comparisons of various types of normality tests. *J Stat Comput Simul.* 2011;81(12):2141-55. doi:10.1080/00949655.2010.520163
112. NIST/SEMATECH. *e-Handbook of Statistical Methods.* NIST; 2013–Section 4.2. doi:10.18434/M32189.
113. Kim T-K. T test as a parametric statistic. *Korean J Anesthesiol.* 2015;68(6):540-6. doi:10.4097/kjae.2015.68.6.540.
114. Nachar N. The Mann-Whitney U: A test for assessing whether two independent samples come from the same population. *TQMP.* 2008;4(1):13-20. doi:10.20982/tqmp.04.1.p013.
115. Greenland S, Pearce N. Statistical foundations for model-based adjustments. *Int J Epidemiol.* 2015;44(5):1692-702. doi:10.1093/ije/dyv145.
116. Hauke J, Kossowski T. Comparison of values of Pearson's and Spearman's correlation coefficients on the same sets of data. *Quaest Geogr.* 2011;30(2):87-93. doi:10.2478/v10117-011-0021-1.
117. Arlot S, Celisse A. A survey of cross-validation procedures for model selection. *Stat Surv.* 2010;4:40-79. doi:10.1214/09-SS054.

118. Snoek J, Larochelle H, Adams RP. Practical Bayesian optimization of machine learning algorithms. *Adv Neural Inf Process Syst.* 2012;25. doi:10.48550/arXiv.1206.2944.
119. Steyerberg EW, Vickers AJ, Cook NR, et al. Assessing the performance of prediction models: a framework for traditional and novel measures. *Epidemiology.* 2010;21(1):128-38. doi:10.1097/EDE.0b013e3181c30fb2129.
120. Dietterich TG. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.* 1998;10(7):1895-923. doi:10.1162/089976698300017197.
121. Youden WJ. Index for rating diagnostic tests. *Cancer.* 1950;3(1):32-5. doi:10.1002/1097-0142(1950)3:1<32::AID-CNCR2820030106>3.0.CO;2-3.
122. Vickers AJ, Elkin EB. Decision curve analysis: a novel method for evaluating prediction models. *Med Decis Making.* 2006;26(6):565-74. doi:10.1177/0272989X06295361.
123. Lundberg SM, Lee S-I. A unified approach to interpreting model predictions. *Adv Neural Inf Process Syst.* 2017;30. doi:10.48550/arXiv.1705.07874.
124. Epstein LJ, Kristo D, Strollo PJ, Friedman N, Malhotra A, Patil SP, et al. Clinical guideline for the evaluation, management and long-term care of obstructive sleep apnea in adults. *Journal of Clinical Sleep Medicine.* 2009;5(3):263–276. doi:10.5664/jcsm.27497.
125. Senaratna CV, Perret JL, Lodge CJ, Lowe AJ, Campbell BE, Matheson MC, et al. Machine learning in obstructive sleep apnea diagnosis: a systematic review. *Sleep Medicine Reviews.* 2021;59:101498. doi:10.1016/j.smr.2021.101498.
126. Liang J, Hou X, Wang Y, Xu Z, Chen H, Sun H, et al. Gradient boosting decision tree approaches outperform logistic regression in the prediction of obstructive sleep apnea severity. *Scientific Reports.* 2022;12(1):10425. doi:10.1038/s41598-022-14329-7.

127. Senaratna CV, Perret JL, Lodge CJ, Lowe AJ, Campbell BE, Matheson MC, et al. Prevalence of obstructive sleep apnea in the general population: A systematic review. *Sleep Medicine Reviews*. 2017;34:70–81. doi:10.1016/j.smr.2016.07.002.
128. Heinzer R, Vat S, Marques-Vidal P, Marti-Soler H, Andrieu D, Tobback N, et al. Prevalence of sleep-disordered breathing in the general population: the HypnoLaus study. *Lancet Respiratory Medicine*. 2015;3(4):310–318. doi:10.1016/S2213-2600(15)00043-0.
129. Young T, Palta M, Dempsey J, Skatrud J, Weber S, Badr S. The occurrence of sleep-disordered breathing among middle-aged adults. *New England Journal of Medicine*. 1993;328(17):1230–1235. doi:10.1056/NEJM199304293281704.
130. Bixler EO, Vgontzas AN, Lin HM, Ten Have T, Leiby BE, Vela-Bueno A, et al. Prevalence of sleep-disordered breathing in women: Effects of gender. *American Journal of Respiratory and Critical Care Medicine*. 2001;163(3):608–613. doi:10.1164/ajrccm.163.3.9911064.
131. Peppard PE, Young T, Palta M, Dempsey J, Skatrud J. Longitudinal study of moderate weight change and sleep-disordered breathing. *American Journal of Epidemiology*. 2000;152(2):147–155. doi:10.1093/aje/152.2.147.
132. Le Cessie S, Goeman JJ. Who is afraid of non-normal data? Choosing between parametric and non-parametric tests. *Eur J Endocrinol*. 2020;182(2):E1–E3. DOI: 10.1530/EJE-19-0922.
133. Micceri T. The unicorn, the normal curve, and other improbable creatures. *Psychol Bull*. 1989;105(1):156–166. DOI: 10.1037/0033-2909.105.1.156.
134. Lu CH et al. Effect of nocturia in patients with different severity of obstructive sleep apnea on polysomnography: A retrospective observational study. *Asian J Urol*. 2024;11(2):e.g. 140–148. DOI: 10.1016/j.ajur.2023.02.003.
135. Macbeth G, Razumiejczyk E, Ledesma RD. Cliff's Delta Calculator: A non-parametric effect size program for two groups of observations.

Univ Psychol. 2011;10(2):545–555. DOI: 10.11144/Javeriana.upsy10-2.cdc.

136. PennState Eberly College of Science – STAT 462: Regression Analysis, ‘Detecting Multicollinearity’.
137. Lo SK, Li IT, et al. Non-significant in univariate but significant in multivariate analysis: a discussion with examples. *Changgeng Yi Xue Za Zhi / Chang Gung Med J*. 1995;18(2):95–101.
138. Choi JH, Kim EJ, Kim YS, Kim TH, Lee SH, Lee HM, et al. Predictive factors of severity in patients with obstructive sleep apnea: oxygen desaturation and arousal index as key indicators. *Sleep Medicine*. 2010;11(8):837–841. doi:10.1016/j.sleep.2010.04.009.
139. Kushida CA, Nichols DA, Holmes TH, Quan SF, Walsh JK, Gottlieb DJ, et al. Predictors of severe obstructive sleep apnea in a clinical cohort: logistic regression analysis. *Chest*. 2012;142(5):1447–1454. doi:10.1378/chest.11-3095.
140. Chiner E, Arriero JM, Signes-Costa J, Marco J, Fuentes I. Validation of a portable device for screening sleep apnea: regression-based predictive factors. *American Journal of Respiratory and Critical Care Medicine*. 1999;159(3):905–910. doi:10.1164/ajrccm.159.3.9803093.
141. Young T, Shahar E, Nieto FJ, Redline S, Newman AB, Gottlieb DJ, et al. Predictors of sleep-disordered breathing in community-dwelling adults: the Sleep Heart Health Study. *American Journal of Respiratory and Critical Care Medicine*. 2002;165(9):1217–1223. doi:10.1164/rccm.2109081.
142. Prokhorenkova L, Gusev G, Vorobev A, Dorogush AV, Gulin A. CatBoost: unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*. 2018;31:6638–6648.
143. Saito T, Rehmsmeier M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS One*. 2015;10(3):e0118432. doi:10.1371/journal.pone.0118432.

144. Gutiérrez-Tobal GC, Álvarez D, Crespo A, del Campo F, Hornero R. Prediction of obstructive sleep apnea severity from clinical and anthropometric features using machine learning classifiers. *Physiological Measurement*. 2019;40(9):095003. doi:10.1088/1361-6579/ab36d3.
145. Hancock JT, Khoshgoftaar TM. CatBoost for big data: an empirical study. *Journal of Big Data*. 2020;7:123. doi:10.1186/s40537-020-00312-6.
146. Youden WJ. Index for rating diagnostic tests. *Cancer*. 1950;3(1):32–35. doi:10.1002/1097-0142(1950)3:1<32::aid-cnrcr2820030106>3.0.co;2-3.
147. Vickers AJ, Elkin EB. Decision curve analysis: a novel method for evaluating prediction models. *Medical Decision Making*. 2006;26(6):565–574. doi:10.1177/0272989X06295361.
148. Steyerberg EW, Vickers AJ, Cook NR, Gerds T, Gonen M, Obuchowski N, et al. Assessing the performance of prediction models: a framework for traditional and novel measures. *Epidemiology*. 2010;21(1):128–138. doi:10.1097/EDE.0b013e3181c30fb2.
149. Krittanawong C, Virk HUH, Bangalore S, Wang Z, Johnson KW, Pinotti R, et al. Machine learning prediction in cardiovascular diseases: a meta-analysis. *Scientific Reports*. 2020;10:16057. doi:10.1038/s41598-020-72685-1.
150. Lundberg SM, Lee SI. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*. 2017;30:4765–4774.
151. Sun H, Li X, Qian Y, Xu Z, Wang Y, Hou X, et al. Explainable machine learning for sleep apnea severity prediction using SHAP values. *Frontiers in Medicine*. 2022;9:862345. doi:10.3389/fmed.2022.862345.
152. Ribeiro MT, Singh S, Guestrin C. “Why should I trust you?” Explaining the predictions of any classifier. *Proceedings of the 22nd ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016; pp. 1135–1144. doi:10.1145/2939672.2939778.

153. Kwon Y, Jeon YJ, Lim YH, Kim HB, Lee SH, Lee J, et al. Interpretable machine learning for predicting obstructive sleep apnea using SHAP analysis. *Scientific Reports*. 2021;11(1):24118. doi:10.1038/s41598-021-03413-7.
154. Holzinger A, Biemann C, Pattichis CS, Kell DB. What do we need to build explainable AI systems for the medical domain? *Reviews in the Medical Informatics*. 2017;26(2):55–62. doi:10.1007/s10209-017-0548-4.
155. Gutiérrez-Tobal GC, Álvarez D, Crespo A, del Campo F, Hornero R. Prediction of obstructive sleep apnea severity from clinical and anthropometric features using machine learning classifiers. *Physiological Measurement*. 2019;40(9):095003. doi:10.1088/1361-6579/ab36d3.
156. Liang J, Hou X, Wang Y, Xu Z, Chen H, Sun H, et al. Gradient boosting decision tree approaches outperform logistic regression in the prediction of obstructive sleep apnea severity. *Scientific Reports*. 2022;12(1):10425. doi:10.1038/s41598-022-14329-7.
157. Kwon Y, Jeon YJ, Lim YH, Kim HB, Lee SH, Lee J, et al. Interpretable machine learning for predicting obstructive sleep apnea using SHAP analysis. *Scientific Reports*. 2021;11(1):24118. doi:10.1038/s41598-021-03413-7.
158. Akbaş T, Akpınar ME, Özkan A, et al. Clinical predictors of obstructive sleep apnea in Turkish patients: A logistic regression analysis. *Tuberk Toraks*. 2018;66(3):211–218. doi:10.5578/tt.66489.
159. Beam AL, Kohane IS. Big data and machine learning in health care. *JAMA*. 2018;319(13):1317–1318. doi:10.1001/jama.2017.18391.
160. Collins GS, Reitsma JB, Altman DG, Moons KGM. Transparent reporting of a multivariable prediction model for individual prognosis or diagnosis (TRIPOD): the TRIPOD statement. *Annals of Internal Medicine*. 2015;162(1):55–63. doi:10.7326/M14-0697.

- 161.** Saito T, Rehmsmeier M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS One*. 2015;10(3):e0118432. doi:10.1371/journal.pone.0118432
- 162.** Martinez-Garcia MA, Campos-Rodriguez F, Javaheri S. Sleep apnea and cardiovascular disease: integrating machine learning predictions with clinical scores. *Chest*. 2020;158(2):757–764. doi:10.1016/j.chest.2020.02.037.



Ek 2. Colab Kodlamaları

Kurulum & Genel Ayarlar

@title 0) Kurulum & Genel Ayarlar (Tek Sefer)

```
!pip -q install numpy pandas scipy scikit-learn xgboost catboost shap statsmodels
```

```
import numpy as np, pandas as pd, math, warnings
warnings.filterwarnings("ignore")
```

```
RANDOM_STATE = 42
```

```
# Hız / doğruluk ayarları (istediğinde değiştir)
```

```
N_OUTER = 5      # outer CV kat sayısı
```

```
N_REPEATS = 3    # outer tekrar sayısı (clf/reg)
```

```
N_INNER = 3      # inner CV kat sayısı (HPO)
```

```
N_ITER = 100     # HPO iterasyonu (>=100 önerilir)
```

```
# Grafik ayarları
```

```
import matplotlib.pyplot as plt
```

```
plt.rcParams["figure.figsize"] = (6.5, 4.2)
```

```
plt.rcParams["figure.dpi"] = 130
```

```
# Yardımcı: BH-FDR
```

```
def fdr_bh(pvals):
```

```
    p = np.asarray(pvals, float)
```

```
    n = len(p)
```

```
    order = np.argsort(p)
```

```
    ranks = np.empty_like(order); ranks[order] = np.arange(1, n+1)
```

```
    q = p*n/ranks
```

```
    q_sorted = np.minimum.accumulate(q[order][::-1])[::-1]
```

```
    out = np.empty_like(q_sorted); out[order] = q_sorted
```

```
    return np.clip(out, 0, 1)
```

```

# Yardımcı: Cliff's  $\delta$ 
from scipy import stats
def cliffs_delta(x, y):
    x = np.asarray(x); y = np.asarray(y)
    nx, ny = len(x), len(y)
    U, _ = stats.mannwhitneyu(x, y, alternative="two-sided")
    return (2*U)/(nx*ny) - 1

# Yardımcı: bootstrap CI (tek ist.)
def bootstrap_ci_stat(x, fn, B=1000, ci=0.95, seed=RANDOM_STATE):
    rng = np.random.default_rng(seed)
    x = np.asarray(x)
    boots = np.empty(B)
    for b in range(B):
        idx = rng.integers(0, len(x), len(x))
        boots[b] = fn(x[idx])
    lo, hi = np.quantile(boots, [(1-ci)/2, 1-(1-ci)/2])
    return (float(lo), float(hi))

# Yardımcı: median farkı (y - x) için bootstrap CI
def bootstrap_ci_median_diff(x, y, B=1000, ci=0.95, seed=RANDOM_STATE):
    rng = np.random.default_rng(seed)
    x = np.asarray(x); y = np.asarray(y)
    obs = np.nanmedian(y) - np.nanmedian(x)
    boots = np.empty(B)
    for b in range(B):
        xb = x[rng.integers(0, len(x), len(x))]
        yb = y[rng.integers(0, len(y), len(y))]
        boots[b] = np.nanmedian(yb) - np.nanmedian(xb)
    lo, hi = np.quantile(boots, [(1-ci)/2, 1-(1-ci)/2])
    return float(obs), float(lo), float(hi)

```

```
print("☑ Kurulum tamam. Ayarlar yüklendi.")
```

1) Veri Yükleme & Doğrulama (Sıralı İndex ile)

```
# @title 1) Veri Yükleme & Doğrulama (Sıralı İndex ile)
```

```
import os, re, glob, shutil
```

```
import numpy as np
```

```
import pandas as pd
```

```
from IPython.display import display, HTML
```

```
# --- 1) CSV'yi edin ---
```

```
def find_csv_in_content():
```

```
    cans = glob.glob("/content/*.csv")
```

```
    cans = sorted(cans, key=lambda p: ("analiz" in p.lower()) + ("set" in p.lower())), reverse=True)
```

```
    return cans[0] if cans else None
```

```
DATA_PATH = find_csv_in_content()
```

```
if DATA_PATH is None:
```

```
    try:
```

```
        from google.colab import files
```

```
        print("CSV yükleyin (bilgisayarınızdan seçin)...")
```

```
        up = files.upload()
```

```
        if len(up):
```

```
            orig = list(up.keys())[0]
```

```
            DATA_PATH = "/content/" + orig
```

```
    except Exception as e:
```

```
        print("Upload atlandı:", repr(e))
```

```
if DATA_PATH is None:
```

```
    try:
```

```
        from google.colab import drive
```

```
        drive.mount('/content/drive', force_remount=False)
```

```

drive_cands = glob.glob("/content/drive/**/*.*csv", recursive=True)
if drive_cands:
    drive_cands = sorted(drive_cands, key=lambda p: (("analiz" in p.lower())
+ ("set" in p.lower()))), reverse=True)
    DATA_PATH = drive_cands[0]
except Exception:
    pass

```

```

assert DATA_PATH is not None, "CSV bulunamadı."

```

```

# --- 2) ASCII isim ---

```

```

ASCII_PATH = "/content/" + re.sub(r"[^A-Za-z0-9_.-]+", "_",
os.path.basename(DATA_PATH))
if DATA_PATH != ASCII_PATH:
    shutil.copy2(DATA_PATH, ASCII_PATH)

```

```

# --- 3) Çoklu encoding ile oku ---

```

```

df, used_enc = None, None
for enc in ["utf-8", "utf-8-sig", "ISO-8859-9", "cp1254", "latin-1"]:
    try:
        df = pd.read_csv(ASCII_PATH, encoding=enc)
        used_enc = enc
        break
    except Exception:
        continue
assert df is not None, "CSV okunamadı."

```

```

# --- 4) Hedef kolonlar ---

```

```

TARGET_CLF = "AHI_SINIF"
TARGET_REG = "AHI_TOPLAM"
GENDER_COL = "CINSIYET_BINARY"

```

```

# --- 5) Şema tablosu ---
def example_vals(s, k=3):
    return ", ".join(map(lambda x: str(x)[:40], list(pd.unique(s.dropna()))[:k]))

schema = pd.DataFrame([
    "column": c,
    "dtype": str(df[c].dtype),
    "n_missing": int(df[c].isna().sum()),
    "pct_missing": float(100.0 * df[c].isna().mean()),
    "n_unique": int(df[c].nunique(dropna=True)),
    "examples": example_vals(df[c])
] for c in df.columns]).reset_index(drop=True) # index sıfırla
schema.index = schema.index + 1 # 1,2,3,4...

# --- 6) 0/1 kontrolleri ---
def bin_check(col):
    if col not in df.columns:
        return col, False, [], False
    vals = sorted(set(pd.unique(df[col].dropna())))
    isbin = set(vals).issubset({0,1})
    return col, True, vals, isbin

checks = [bin_check(TARGET_CLF), bin_check(GENDER_COL)]

# --- 7) Ekrana rapor ---
print(f"Dosya: {ASCII_PATH} | encoding: {used_enc} | shape={df.shape}")
display(HTML("<h4>Şema Özeti:</h4>"))
display(schema)

display(HTML("<h4>İlk 5 Satır:</h4>"))
display(df.head(5))

```

```

print("\n0/1 kontrolleri:")
for name,present,vals,isbin in checks:
    print(f"- {name}: present={present}, values={vals[:10]}{'...' if len(vals)>10
else "}, is_0_1={isbin}")

if TARGET_CLF in df.columns:
    vc = df[TARGET_CLF].value_counts(dropna=False)
    display(HTML("<h4>AHI_SINIF Sınıf Dağılımı:</h4>"))
    display(pd.DataFrame({"class": vc.index, "count": vc.values, "pct":
(vc.values/len(df))*100}))
if TARGET_REG in df.columns:
    reg = pd.to_numeric(df[TARGET_REG], errors="coerce")
    display(HTML("<h4>AHI_TOPLAM Özet:</h4>"))
    display(pd.Series({
        "mean": reg.mean(), "sd": reg.std(ddof=1), "median": reg.median(),
        "min": reg.min(), "max": reg.max(), "n_non_missing": reg.notna().sum()
    }).to_frame("AHI_TOPLAM").T)

print("\n☑ Adım 1 tamam — veri yüklendi ve doğrulandı.")

```

2) Klasik Tanımlayıcı & Grup Farkı (Cinsiyet Hariç, Tam Kapsam)

@title 2) Klasik tanımlayıcı & Grup farkı (Cinsiyet Hariç, Tam Kapsam)

```

import numpy as np, pandas as pd
from scipy import stats
import matplotlib.pyplot as plt
from IPython.display import display, HTML

# --- Ayarlar ---
TARGET_CLF = "AHI_SINIF"
TARGET_REG = "AHI_TOPLAM"
GENDER_COL = "CINSIYET_BINARY"
RANDOM_STATE = 42

```

```

np.random.seed(RANDOM_STATE)

# --- 0) Kapsam: cinsiyet HARIÇ tüm özellikler ---
assert TARGET_CLF in df.columns, "AHI_SINIF bulunamadı; Adım 1'i çalıştır."
classic_features = [c for c in df.columns if c not in [TARGET_CLF,
TARGET_REG, GENDER_COL]]

# numerik coercion
work = df.copy()
for c in classic_features + [TARGET_CLF]:
    work[c] = pd.to_numeric(work[c], errors="coerce")

# --- 1) Tanımlayıcı istatistikler ---
def desc_stats(s: pd.Series):
    s = pd.to_numeric(s, errors="coerce")
    return pd.Series({
        "mean": np.nanmean(s),
        "sd": np.nanstd(s, ddof=1),
        "median": np.nanmedian(s),
        "IQR": np.nanpercentile(s, 75) - np.nanpercentile(s, 25),
        "min": np.nanmin(s),
        "max": np.nanmax(s),
        "skew": stats.skew(s, nan_policy="omit"),
        "kurt": stats.kurtosis(s, nan_policy="omit"),
        "n": int(s.notna().sum())
    })

desc = pd.concat({c: desc_stats(work[c]) for c in classic_features},
axis=1).T.reset_index()
desc = desc.rename(columns={"index": "variable"})
desc.index = np.arange(1, len(desc)+1) # 1,2,3,...

```

```
display(HTML("<h4>Tanımlayıcı İstatistikler (Tüm Değişkenler)</h4>"))
display(desc)
```

```
# --- 2) Grup farkı: AHI_SINIF 0 vs 1 (MWU + Cliff's  $\delta$ ) ---
```

```
def cliffs_delta(x, y):
    nx, ny = len(x), len(y)
    U, _ = stats.mannwhitneyu(x, y, alternative="two-sided")
    return 2*U/(nx*ny) - 1
```

```
group_results = []
```

```
g0 = work[work[TARGET_CLF] == 0]
```

```
g1 = work[work[TARGET_CLF] == 1]
```

```
for col in classic_features:
```

```
    x0 = g0[col].dropna()
```

```
    x1 = g1[col].dropna()
```

```
    if len(x0) > 0 and len(x1) > 0:
```

```
        U, p = stats.mannwhitneyu(x0, x1, alternative="two-sided")
```

```
        cd = cliffs_delta(x0, x1)
```

```
        group_results.append({
```

```
            "variable": col,
```

```
            "n0": len(x0),
```

```
            "n1": len(x1),
```

```
            "U_stat": U,
```

```
            "p_value": p,
```

```
            "cliffs_delta": cd
```

```
        })
```

```
group_df = pd.DataFrame(group_results)
```

```
group_df["p_adj_BH"] = stats.false_discovery_control(group_df["p_value"],
method="bh")
```

```
group_df.index = np.arange(1, len(group_df)+1)
```

```
display(HTML("<h4>Grup Farkı Analizi (AHI_SINIF=0 vs 1)</h4>"))
display(group_df)
```

--- 3) Grafikler: Histogram + Boxplot (Tüm değişkenler) ---

```
for col in classic_features:
```

```
    fig, axes = plt.subplots(1, 2, figsize=(12, 4))
```

```
    # Histogram
```

```
    axes[0].hist(work[col].dropna(), bins=20, color="skyblue", edgecolor="black")
```

```
    axes[0].set_title(f"Histogram - {col}")
```

```
    axes[0].set_xlabel(col)
```

```
    axes[0].set_ylabel("Frekans")
```

```
    # Boxplot (gruplara göre)
```

```
    axes[1].boxplot([g0[col].dropna(), g1[col].dropna()], labels=["AHI_SINIF=0",
"AHI_SINIF=1"])
```

```
    axes[1].set_title(f"Boxplot (Gruplara Göre) - {col}")
```

```
    axes[1].set_ylabel(col)
```

```
    plt.suptitle(f"{col} - Dağılım ve Grup Farkı", fontsize=14)
```

```
    plt.show()
```

```
print("\n☑ Adım 2 tamam — tanımlayıcı istatistikler, grup farkı ve tüm grafikler
ekrana basıldı.")
```

3) Korelasyon Analizi (Spearman, Tüm Sayısal Değişkenler)

```
# === 4) Kolinearite & Korelasyon (NORMALİZASYON + DUMMY
KONTROL İLE) ===
```

```
# 4A) Spearman korelasyon (hedefler hariç) + p-matrisi
```

```
# 4B) Hedeflerle Spearman (AHI_SINIF, AHI_TOPLAM) + BH-FDR (q)
```

```
# 4C)  $|\rho| \geq 0.90$  yüksek korelasyon çiftleri
```

```
# 4D) VIF (klasik: cinsiyet hariç) -> num z-score + kategorik OHE(drop_first)
```

```
# 4E) VIF (modelleme: cinsiyet dahil) -> num z-score + kategorik
OHE(drop_first)
```

```
# 4F) Spearman ısı haritası (matplotlib)
```

```
import numpy as np, pandas as pd, matplotlib.pyplot as plt
from IPython.display import display, HTML
from scipy.stats import spearmanr
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```
assert 'df' in globals(), "df yok. CSV'yi okuttuğın hücreyi çalıştır."
```

```
# ---- Parametreler
```

```
TARGET_CLF = "AHI_SINIF"
TARGET_REG = "AHI_TOPLAM"
CINSIYET_COL = "CINSIYET_BINARY"
THR = 0.90      # yüksek korelasyon eşiği
VIF_SOFT = 5
VIF_HARD = 10
```

```
# ---- Yardımcılar
```

```
def bh_fdr(pvals):
    p = np.asarray(pvals, float)
    n = p.size
    order = np.argsort(p)
    rank = np.arange(1, n+1)
    q = np.empty_like(p)
    q[order] = np.minimum.accumulate((p[order][::-1] * n / rank[::-1]))[::-1]
    return np.clip(q, 0, 1)
```

```
def _zscore(x: pd.DataFrame):
```

```
    # sadece sayısalara uygula, sabit/tekil kolonları dışarıda bırak
    num = x.select_dtypes(include=["number"]).copy()
    keep = num.columns[num.nunique(dropna=True) > 1]
```

```

    z = num[keep].apply(lambda s: (s - s.mean()) / (s.std(ddof=0) if s.std(ddof=0)
not in (0, np.nan) else 1.0))
    return z

```

```

def _one_hot_dropfirst(df_cat: pd.DataFrame):
    if df_cat.empty:
        return pd.DataFrame(index=df_cat.index)
    # bool'u kategorik say: object/category/bool
    cat = df_cat.select_dtypes(include=["object", "category", "bool"]).copy()
    if cat.empty:
        return pd.DataFrame(index=df_cat.index)
    # OHE + drop_first -> dummy trap yok
    dummies = pd.get_dummies(cat, drop_first=True, dummy_na=False)
    return dummies

```

```

def _build_design(df_src: pd.DataFrame, drop_targets=True,
include_gender=True):
    """VIF için tasarım matrisi kur:
    - drop_targets: hedefleri at
    - include_gender: CINSİYET_BINARY dahil mi?
    - sayısalara z-score (tekil/sabit hariç), kategoriklere OHE(drop_first)
    - sabit/tekil/dolu olmayan kolonları temizler
    """
    drops = []
    if drop_targets:
        for c in [TARGET_CLF, TARGET_REG]:
            if c in df_src.columns: drops.append(c)
    base = df_src.drop(columns=drops, errors="ignore").copy()

    # Cinsiyet dahil/haric
    if not include_gender and CINSİYET_COL in base.columns:
        base = base.drop(columns=[CINSİYET_COL])

```

```

# Numerik & kategorik ayrımı
num_raw = base.select_dtypes(include=["number"]).copy()
cat_raw = base.drop(columns=num_raw.columns, errors="ignore")

# Z-score sadece sayısalara
num_z = _zscore(num_raw)

# Dummy kontrol: OHE(drop_first) kategorikler
dmy = _one_hot_dropfirst(cat_raw)
if not include_gender:
    pass # zaten çıkarıldı
else:
    # cinsiyet numerik değilse dönüştür (0/1)
    if CINSIYET_COL in base.columns and CINSIYET_COL not in
num_z.columns and CINSIYET_COL not in dmy.columns:
        try:
            dmy[CINSIYET_COL] = pd.to_numeric(base[CINSIYET_COL],
errors="coerce")
        except Exception:
            # kategorik ise OHE zaten almıştır
            pass

# Birleştir
X = pd.concat([num_z, dmy], axis=1)

# Tekil/sabit kolonları at
nunq = X.nunique(dropna=True)
keep_cols = nunq.index[nunq > 1]
if len(keep_cols) < X.shape[1]:
    dropped = list(set(X.columns) - set(keep_cols))
    if dropped:

```

```

        print(" ⚠ Tekil/sabit olduğu için atılan kolonlar:", dropped)
    X = X[keep_cols].copy()

    # NaN doldur
    X = X.fillna(0.0)
    return X

def compute_vif(X_df: pd.DataFrame):
    if X_df.empty:
        return pd.DataFrame(columns=["Değişken", "VIF"])
    vals = X_df.values
    rows = []
    for i, col in enumerate(X_df.columns):
        try:
            v = float(variance_inflation_factor(vals, i))
        except Exception:
            v = np.nan
        rows.append((col, v))
    out = pd.DataFrame(rows, columns=["Değişken", "VIF"]).sort_values("VIF",
ascending=False).reset_index(drop=True)
    return out

# ===== 4A: Spearman (hedefler hariç, orijinal sayısalılar) =====
targets = [c for c in [TARGET_CLF, TARGET_REG] if c in df.columns]
num_df = df.drop(columns=targets,
errors="ignore").select_dtypes(include=["number"]).copy()
cols = num_df.columns.tolist()
R = np.zeros((len(cols), len(cols)), float)
P = np.zeros((len(cols), len(cols)), float)
for i, ci in enumerate(cols):
    for j, cj in enumerate(cols):
        if j < i:

```

```

        R[i, j] = R[j, i]; P[i, j] = P[j, i]
    else:
        rho, p = spearmanr(num_df[ci], num_df[cj], nan_policy='omit')
        R[i, j] = rho; P[i, j] = p
corr_df = pd.DataFrame(R, index=cols, columns=cols)
pval_df = pd.DataFrame(P, index=cols, columns=cols)

display(HTML("<h4>4A — Spearman Korelasyon Matrisi (hedefler hariç, sayısal kolonlar)</h4>"))
display(corr_df.round(3))

# ===== 4B: Hedeflerle Spearman (|r| sıralı, BH-FDR q) =====
def spearman_with_target(df_src, feat_cols, target_col, show_name):
    if target_col not in df_src.columns:
        print(f"⚠ {target_col} yok, atlandı.")
        return None
    y = pd.to_numeric(df_src[target_col], errors="coerce")
    rows = []
    for c in feat_cols:
        x = pd.to_numeric(df_src[c], errors="coerce")
        rho, p = spearmanr(x, y, nan_policy='omit')
        rows.append((c, rho, p))
    out = pd.DataFrame(rows, columns=["Değişken", "Spearman_r", "p"])
    out["q_BH"] = bh_fdr(out["p"].values)
    out["|r|"] = out["Spearman_r"].abs()
    out = out.sort_values("|r|",
ascending=False).drop(columns="|r|").reset_index(drop=True)
    display(HTML(f"<h4>4B — {show_name} ile Spearman (|r| sıralı)</h4>"))
    display(out.round({"Spearman_r":3, "p":3, "q_BH":3}))
    return out

```

```

spearman_clf = spearman_with_target(df, cols, TARGET_CLF, "AHI_SINIF") if
TARGET_CLF in df.columns else None
spearman_reg = spearman_with_target(df, cols, TARGET_REG,
"AHI_TOPLAM") if TARGET_REG in df.columns else None

# ===== 4C:  $|\rho| \geq 0.90$  yüksek korelasyon çiftleri =====
pairs = []
for i in range(len(cols)):
    for j in range(i+1, len(cols)):
        r = corr_df.iloc[i, j]
        if abs(r) >= THR:
            pairs.append((cols[i], cols[j], float(r), float(pval_df.iloc[i, j])))
pairs_df = pd.DataFrame(pairs,
columns=["Değişken_1", "Değişken_2", "Spearman_r", "p"])
if not pairs_df.empty:
    pairs_df[|"r|"] = pairs_df["Spearman_r"].abs()
    pairs_df = pairs_df.sort_values("|r|",
ascending=False).drop(columns="|r|").reset_index(drop=True)
display(HTML(f"<h4>4C — Yüksek korelasyon çiftleri ( $|\rho| \geq$ 
{THR:.2f})</h4>"))
display(pairs_df.round({"Spearman_r":3, "p":3}) if not pairs_df.empty else
pd.DataFrame({"Bilgi":[f"≥{THR:.2f} korelasyon çifti bulunmadı."]}))

# ===== 4D: VIF (klasik) — hedefler & CINSİYET_BINARY hariç; num
z-score + OHE(drop_first) =====
# Tasarım: tüm num z-score + kategorikler OHE (cinsiyet hariç)
base_for_vif_classic = df.copy()
# Cinsiyeti klasik VIF'ten çıkaracağız; ama kategorikse OHE üretmeye kalkmasın
diye num/kategorik ayrımıyla yönetiyoruz
design_classic = _build_design(base_for_vif_classic, drop_targets=True,
include_gender=False)
vif_classic = compute_vif(design_classic)

```

```
display(HTML("<h4>4D — VIF (klasik) — num z-score, kategorik  
OHE(drop_first), CINSIYET hariç</h4>"))  
display(vif_classic)
```

```
# ===== 4E: VIF (modelleme) — hedefler hariç + CINSIYET dahil; num z-  
score + OHE(drop_first) =====
```

```
base_for_vif_model = df.copy()  
design_model = _build_design(base_for_vif_model, drop_targets=True,  
include_gender=True)  
vif_model = compute_vif(design_model)
```

```
display(HTML("<h4>4E — VIF (modelleme) — num z-score, kategorik  
OHE(drop_first), CINSIYET dahil</h4>"))  
display(vif_model)
```

```
# ===== 4F: Spearman ısı haritası (sayısal orijinal) =====
```

```
plt.figure(figsize=(max(6, 0.5*len(cols)), max(5, 0.5*len(cols))))  
im = plt.imshow(corr_df.values, cmap="coolwarm", vmin=-1, vmax=1)  
plt.colorbar(im, fraction=0.046, pad=0.04, label="Spearman  $\rho$ ")  
plt.xticks(ticks=np.arange(len(cols)), labels=cols, rotation=90)  
plt.yticks(ticks=np.arange(len(cols)), labels=cols)  
plt.title("4F — Spearman Korelasyon Isı Haritası (hedefler hariç, orijinal  
sayısallar)")  
plt.tight_layout()  
plt.show()
```

```
# ===== Özet / Karar =====
```

```
def flag_high_vif(vif_df, thr):  
    if vif_df.empty: return []  
    return vif_df.loc[vif_df["VIF"] >= thr, "Değişken"].tolist()
```

```

hi_pairs = 0 if pairs_df.empty else len(pairs_df)
hard_vif = set(flag_high_vif(vif_classic, VIF_HARD) + flag_high_vif(vif_model,
VIF_HARD))
soft_vif = set(flag_high_vif(vif_classic, VIF_SOFT) + flag_high_vif(vif_model,
VIF_SOFT))

print("\n--- Kısa Özet ---")
print(f"- Yüksek korelasyon ( $|\rho| \geq \{THR:.2f\}$ ) çift sayısı: {hi_pairs}")
print(f"- VIF klasik (max): {vif_classic['VIF'].max():.2f}" if not vif_classic.empty
else "- VIF klasik: boş")
print(f"- VIF modelleme (max): {vif_model['VIF'].max():.2f}" if not
vif_model.empty else "- VIF modelleme: boş")
if hard_vif or hi_pairs>0:
    print(f" ! Çoklu bağlantı riski VAR (yüksek kore veya
VIF $\geq$ {VIF_HARD}).")
elif soft_vif:
    print(f" ⚠ Orta düzey kolinearite sinyali var (VIF $\geq$ {VIF_SOFT}).")
else:
    print(f" ☑ Belirgin multikolinerite bulgusu YOK (yüksek kore yok, VIF<5).")

```

4) Normalite ve Varyans Homojenliği Testleri (Cinsiyet Hariç)

@title 4) Normalite ve Varyans Homojenliği Testleri (Cinsiyet Hariç)

```
norm_var_results = []
```

```
for col in classic_features:
```

```
    col_data = work[col].dropna()
```

```
    # --- Normalite testi ---
```

```
    if len(col_data) <= 5000:
```

```
        stat_shapiro, p_shapiro = stats.shapiro(col_data)
```

```
    else:
```

```
        stat_shapiro, p_shapiro = np.nan, np.nan # çok büyük veri için atlanır
```

```

# --- Varyans homojenliği (Levene) ---
g0_vals = g0[col].dropna()
g1_vals = g1[col].dropna()
if len(g0_vals) > 1 and len(g1_vals) > 1:
    stat_levene, p_levene = stats.levene(g0_vals, g1_vals)
else:
    stat_levene, p_levene = np.nan, np.nan

norm_var_results.append({
    "variable": col,
    "n_total": len(col_data),
    "shapiro_W": stat_shapiro,
    "shapiro_p": p_shapiro,
    "levene_stat": stat_levene,
    "levene_p": p_levene
})

norm_var_df = pd.DataFrame(norm_var_results)
norm_var_df.index = np.arange(1, len(norm_var_df)+1)

display(HTML("<h4>Normalite (Shapiro) ve Varyans Homojenliği (Levene) Test Sonuçları</h4>"))
display(norm_var_df)

print("\n☑ Adım 4 tamam — normalite ve varyans homojenliği testleri ekrana basıldı.")

```

5) Klasik Grup Karşılaştırmaları (Cinsiyet Hariç)

@title 5) Klasik Grup Karşılaştırmaları (Cinsiyet Hariç)

```
classic_comp_results = []
```

```

for col in classic_features:
    g0_vals = g0[col].dropna()
    g1_vals = g1[col].dropna()
    if len(g0_vals) > 1 and len(g1_vals) > 1:
        # Normalite kontrolü
        p_shap_g0 = stats.shapiro(g0_vals)[1] if len(g0_vals) <= 5000 else np.nan
        p_shap_g1 = stats.shapiro(g1_vals)[1] if len(g1_vals) <= 5000 else np.nan
        norm_ok = (p_shap_g0 > 0.05) and (p_shap_g1 > 0.05)

        # Varyans homojenliği kontrolü
        p_levene = stats.levene(g0_vals, g1_vals)[1]
        var_ok = p_levene > 0.05

        if norm_ok and var_ok:
            # t-testi (eş varyans)
            t_stat, p_val = stats.ttest_ind(g0_vals, g1_vals, equal_var=True)
            # Cohen's d
            nx, ny = len(g0_vals), len(g1_vals)
            pooled_std = np.sqrt(((nx-1)*np.var(g0_vals, ddof=1) + (ny-
1)*np.var(g1_vals, ddof=1)) / (nx+ny-2))
            cohen_d = (np.mean(g0_vals) - np.mean(g1_vals)) / pooled_std
            test_type = "t-test (eş varyans)"
            effect = cohen_d
        elif norm_ok and not var_ok:
            # t-testi (Welch)
            t_stat, p_val = stats.ttest_ind(g0_vals, g1_vals, equal_var=False)
            nx, ny = len(g0_vals), len(g1_vals)
            pooled_std = np.sqrt((np.var(g0_vals, ddof=1) + np.var(g1_vals, ddof=1))
/ 2)
            cohen_d = (np.mean(g0_vals) - np.mean(g1_vals)) / pooled_std
            test_type = "t-test (Welch)"

```

```

        effect = cohen_d
    else:
        # Mann-Whitney U testi
        t_stat, p_val = stats.mannwhitneyu(g0_vals, g1_vals, alternative="two-
sided")
        effect = cliffs_delta(g0_vals, g1_vals)
        test_type = "Mann-Whitney U"

    classic_comp_results.append({
        "variable": col,
        "n0": len(g0_vals),
        "n1": len(g1_vals),
        "test": test_type,
        "stat": t_stat,
        "p_value": p_val,
        "effect_size": effect
    })

classic_comp_df = pd.DataFrame(classic_comp_results)
classic_comp_df["p_adj_BH"] =
stats.false_discovery_control(classic_comp_df["p_value"], method="bh")
classic_comp_df.index = np.arange(1, len(classic_comp_df)+1)

display(HTML("<h4>Grup Karşılaştırmaları (Test Türü ve Etki Büyüklüğü
ile)</h4>"))
display(classic_comp_df)

print("\n☑ Adım 5 tamam — testler ve etki büyüklükleri ekrana basıldı.")

```

6) Hedeflerle Tek Değişkenli Regresyon Analizi

@title 6) Hedeflerle Tek Değişkenli Regresyon Analizi

```
import statsmodels.api as sm
```

```

regression_results = []

for col in classic_features:
    X = work[[col]].dropna()
    y_reg = work.loc[X.index, TARGET_REG]
    y_clf = work.loc[X.index, TARGET_CLF]

    # --- 1) Basit Doğrusal Regresyon (AHI_TOPLAM) ---
    X_const = sm.add_constant(X)
    model_ols = sm.OLS(y_reg, X_const).fit()
    beta = model_ols.params[col]
    ci_low, ci_high = model_ols.conf_int().loc[col]
    p_val_reg = model_ols.pvalues[col]

    regression_results.append({
        "variable": col,
        "model": "Linear (OLS)",
        "beta": beta,
        "ci_low": ci_low,
        "ci_high": ci_high,
        "p_value": p_val_reg
    })

    # --- 2) Lojistik Regresyon (AHI_SINIF) ---
    try:
        model_logit = sm.Logit(y_clf, X_const).fit(dis=0)
        beta_logit = model_logit.params[col]
        ci_low_l, ci_high_l = model_logit.conf_int().loc[col]
        p_val_clf = model_logit.pvalues[col]
        regression_results.append({
            "variable": col,

```

```

        "model": "Logistic",
        "beta": beta_logit,
        "ci_low": ci_low_l,
        "ci_high": ci_high_l,
        "p_value": p_val_clf
    })
except Exception as e:
    regression_results.append({
        "variable": col,
        "model": "Logistic",
        "beta": np.nan,
        "ci_low": np.nan,
        "ci_high": np.nan,
        "p_value": np.nan
    })

regression_df = pd.DataFrame(regression_results)
regression_df.index = np.arange(1, len(regression_df)+1)

display(HTML("<h4>Hedeflerle Tek Değişkenli Regresyon Analizleri</h4>"))
display(regression_df)

print("\n☑ Adım 6 tamam — hedeflerle regresyon sonuçları ekrana basıldı.")

```

7) Modelleme Öncesi Veri Hazırlama (Fix Edilmiş)

```

# @title 7) Modelleme Öncesi Veri Hazırlama (Fix Edilmiş)
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
import numpy as np

```

```

# Bağımsız değişkenler: hedefler hariç tüm kolonlar
X = work.drop(columns=[TARGET_REG, TARGET_CLF])
y_reg = work[TARGET_REG]
y_clf = work[TARGET_CLF]

# Kategorik ve sayısal değişkenleri ayır
cat_cols = X.select_dtypes(include=["object", "category",
"bool"]).columns.tolist()
num_cols = X.select_dtypes(include=["int64", "float64"]).columns.tolist()

# Eksik değerleri doldurma + encoding + scaling
preprocessor = ColumnTransformer(
    transformers=[
        ("num", Pipeline([
            ("imputer", SimpleImputer(strategy="median")),
            ("scaler", StandardScaler())
        ]), num_cols),
        ("cat", Pipeline([
            ("imputer", SimpleImputer(strategy="most_frequent")),
            ("encoder", OneHotEncoder(drop="first", handle_unknown="ignore"))
        ]), cat_cols)
    ]
)

# Train-Test ayırımı (regresyon)
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(
    X, y_reg, test_size=0.2, random_state=42
)

# Train-Test ayırımı (sınıflama)
X_train_clf, X_test_clf, y_train_clf, y_test_clf = train_test_split(

```

```
X, y_clf, test_size=0.2, random_state=42
)
```

```
print("📊 Sayısal kolonlar:", num_cols)
print("📊 Kategorik kolonlar:", cat_cols)
print("\n✅ Adım 7 tamam — veri hazır, train/test setleri oluşturuldu.")
```

8) Temel Modeller (RF, XGB, CatBoost) – Tanıma Dayalı RMSE ile

@title 8) Temel Modeller (RF, XGB, CatBoost) – Tanıma Dayalı RMSE ile

```
import numpy as np, pandas as pd

from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from xgboost import XGBRegressor, XGBClassifier
from catboost import CatBoostRegressor, CatBoostClassifier

from sklearn.metrics import r2_score, accuracy_score, roc_auc_score
from IPython.display import display, HTML
from sklearn.pipeline import Pipeline

# --- Regresyon (RF / XGB / CatBoost) ---
reg_models = {
    "RandomForestRegressor": RandomForestRegressor(random_state=42),
    "XGBRegressor": XGBRegressor(random_state=42,
objective="reg:squarederror"),
    "CatBoostRegressor": CatBoostRegressor(verbose=0, random_state=42)
}

reg_rows = []
for name, model in reg_models.items():
    pipe = Pipeline([("preprocessor", preprocessor), ("model", model)])
    pipe.fit(X_train_reg, y_train_reg)
    y_pred = pipe.predict(X_test_reg)
    # RMSE = sqrt(mean((y_true - y_pred)^2)) ← literatür tanımı
    rmse = float(np.sqrt(np.mean((y_test_reg - y_pred)**2)))
```

```

r2 = float(r2_score(y_test_reg, y_pred))
reg_rows.append({"Model": name, "R2": r2, "RMSE": rmse})

reg_df = pd.DataFrame(reg_rows).sort_values("R2", ascending=False)

# --- Sınıflama (RF / XGB / CatBoost) ---
clf_models = {
    "RandomForestClassifier": RandomForestClassifier(random_state=42),
    "XGBClassifier": XGBClassifier(random_state=42, eval_metric="logloss"),
    "CatBoostClassifier": CatBoostClassifier(verbose=0, random_state=42)
}

clf_rows = []
for name, model in clf_models.items():
    pipe = Pipeline([("preprocessor", preprocessor), ("model", model)])
    pipe.fit(X_train_clf, y_train_clf)
    y_pred = pipe.predict(X_test_clf)
    # predict_proba mevcut (RF/XGB/CatBoost) → ROC-AUC için olasılık
    # kullanalım
    y_prob = pipe.predict_proba(X_test_clf)[:, 1]
    acc = float(accuracy_score(y_test_clf, y_pred))
    auc = float(roc_auc_score(y_test_clf, y_prob))
    clf_rows.append({"Model": name, "Accuracy": acc, "ROC_AUC": auc})

clf_df = pd.DataFrame(clf_rows).sort_values("ROC_AUC", ascending=False)

# --- Çıktılar ---
display(HTML("<h4>📊 Regresyon Modelleri — Temel Performans</h4>"))
display(reg_df)

display(HTML("<h4>📊 Sınıflama Modelleri — Temel Performans</h4>"))
display(clf_df)

```

```
print("\n☑ Adım 8 tamam — temel karşılaştırmalar literatür tanımıyla  
(RMSE= $\sqrt{\text{MSE}}$ ).")
```

8.5-A) Veri yükleme: df yoksa upload penceresi açılır, CSV okunur

```
import re, os, warnings; warnings.filterwarnings("ignore")  
import numpy as np, pandas as pd  
from IPython.display import display, HTML
```

```
if 'df' not in globals():
```

```
    try:
```

```
        from google.colab import files
```

```
        print("📁 df bulunamadı: Upload penceresi açılıyor...")
```

```
        up = files.upload()
```

```
        assert len(up) > 0, "Dosya seçilmedi."
```

```
        fname = next(iter(up.keys()))
```

```
        ascii_name = re.sub(r"^[A-Za-z0-9_.-]+", "_", fname)
```

```
        if ascii_name != fname:
```

```
            os.rename(fname, ascii_name)
```

```
            fname = ascii_name
```

```
        used_enc = None
```

```
        for enc in ["utf-8", "utf-8-sig", "ISO-8859-9", "cp1254", "latin-1"]:
```

```
            try:
```

```
                df = pd.read_csv(fname, encoding=enc)
```

```
                used_enc = enc
```

```
                break
```

```
            except Exception:
```

```
                pass
```

```
        assert 'df' in globals(), "CSV okunamadı; encoding'leri deneyin."
```

```
        print(f"☑ CSV okundu: {fname} | encoding={used_enc} |
```

```
shape={df.shape}")
```

```
    except Exception as e:
```

```

        raise RuntimeError("Yükleme başarısız. Lütfen dosyayı tekrar yükleyin.")
    from e
else:
    print(f"☑ df zaten bellekte: shape={df.shape}")

display(df.head(3))

# 8.5-B) Hedefleri doğrula; X_all / y_clf / y_reg; ön-işleme (impute + One-Hot)
import numpy as np, pandas as pd
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder

TARGET_CLF = "AHI_SINIF"
TARGET_REG = "AHI_TOPLAM"

for need in [TARGET_CLF, TARGET_REG]:
    assert need in df.columns, f"Hedef kolon yok: {need}. Mevcut:
    {list(df.columns)}"

X_all = df.drop(columns=[TARGET_CLF, TARGET_REG]).copy()
y_clf = pd.to_numeric(df[TARGET_CLF], errors="coerce").astype(int).values
y_reg = pd.to_numeric(df[TARGET_REG], errors="coerce").values

num_cols = X_all.select_dtypes(include=["number"]).columns.tolist()
cat_cols =
X_all.select_dtypes(include=["object", "category", "bool"]).columns.tolist()

preprocess = ColumnTransformer(
    transformers=[
        ("num", SimpleImputer(strategy="median"), num_cols),
        ("cat", Pipeline([("imp", SimpleImputer(strategy="most_frequent"))],

```

```

        ("ohe", OneHotEncoder(handle_unknown="ignore"))]), cat_cols)

    ],
    remainder="drop"
)

print(f"📊 Özellikler: toplam={X_all.shape[1]} | sayısal={len(num_cols)} |
kategorik={len(cat_cols)}")
if "CINSİYET_BINARY" in X_all.columns:
    print("❗ CINSİYET_BINARY modellerde 0/1 olarak DAHİL (klasikte yok).")
# 9-Setup) CV ayarları, parametre uzayları, bootstrap yardımcılar, paket
kontrolleri
import numpy as np, pandas as pd
from sklearn.model_selection import (RepeatedStratifiedKFold, RepeatedKFold,
                                     StratifiedKFold, KFold, RandomizedSearchCV)
from sklearn.metrics import (roc_auc_score, average_precision_score, f1_score,
                             recall_score,
                             brier_score_loss, roc_curve, precision_recall_curve,
                             r2_score, mean_squared_error, mean_absolute_error)
from sklearn.calibration import CalibratedClassifierCV, calibration_curve
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.pipeline import Pipeline

# xgboost / catboost yoksa kur
try:
    from xgboost import XGBClassifier, XGBRegressor
except Exception:
    import os; os.system("pip -q install xgboost"); from xgboost import
XGBClassifier, XGBRegressor
try:
    from catboost import CatBoostClassifier, CatBoostRegressor
except Exception:

```

```

import os; os.system("pip -q install catboost"); from catboost import
CatBoostClassifier, CatBoostRegressor

RANDOM_STATE = 42
N_OUTER, N_REPEATS, N_INNER, N_ITER = 5, 2, 3, 40 # 5x2 outer, 3-fold
inner, 40 arama
B_BOOT = 300

def clf_spaces(y_train):
    pos = int((y_train==1).sum()); neg = int((y_train==0).sum())
    spw = max(1.0, neg / max(1, pos))
    return {
        "rf": {"model": RandomForestClassifier(random_state=RANDOM_STATE,
n_jobs=-1),
            "params": {
                "model__n_estimators": [200,300,400,500],
                "model__max_depth": [None,6,10,14,18],
                "model__min_samples_split": [2,5,10,20],
                "model__min_samples_leaf": [1,2,4,8],
                "model__class_weight": [None,"balanced"]
            }
        },
        "xgb": {"model": XGBClassifier(random_state=RANDOM_STATE,
eval_metric="logloss",
            tree_method="hist", n_jobs=-1),
            "params": {
                "model__n_estimators": [300,500,700],
                "model__max_depth": [3,4,6,8],
                "model__learning_rate": [0.02,0.05,0.1,0.2],
                "model__subsample": [0.7,0.85,1.0],
                "model__colsample_bytree": [0.6,0.8,1.0],
                "model__min_child_weight": [1,3,6,10],
                "model__gamma": [0,0.5,1,3],

```

```

        "model__scale_pos_weight": [spw]
    }},
    "cat": {"model": CatBoostClassifier(random_seed=RANDOM_STATE,
verbose=0, loss_function="Logloss"),
        "params": {
            "model__iterations": [400,600,800],
            "model__depth": [4,6,8,10],
            "model__learning_rate": [0.02,0.05,0.1,0.2],
            "model__l2_leaf_reg": [1,3,5,10,15],
            "model__border_count": [64,128,255]
        }}
    }

def reg_spaces():
    return {
        "rf": {"model": RandomForestRegressor(random_state=RANDOM_STATE,
n_jobs=-1),
            "params": {
                "model__n_estimators": [200,300,400,500],
                "model__max_depth": [None,6,10,14,18],
                "model__min_samples_split": [2,5,10,20],
                "model__min_samples_leaf": [1,2,4,8]
            }}
        "xgb": {"model": XGBRegressor(random_state=RANDOM_STATE,
tree_method="hist"),
            "params": {
                "model__n_estimators": [300,500,700],
                "model__max_depth": [3,4,6,8],
                "model__learning_rate": [0.02,0.05,0.1,0.2],
                "model__subsample": [0.7,0.85,1.0],
                "model__colsample_bytree": [0.6,0.8,1.0],
                "model__min_child_weight": [1,3,6,10],

```

```

        "model__gamma": [0,0.5,1,3]
    }},
    "cat": {"model": CatBoostRegressor(random_seed=RANDOM_STATE,
verbose=0, loss_function="RMSE"),
        "params": {
            "model__iterations": [400,600,800],
            "model__depth": [4,6,8,10],
            "model__learning_rate": [0.02,0.05,0.1,0.2],
            "model__l2_leaf_reg": [1,3,5,10,15],
            "model__border_count": [64,128,255]
        }}
}

```

```

def bootstrap_ci_scores(y_true, y_score, metric_fn, B=B_BOOT, ci=0.95,
seed=RANDOM_STATE, threshold=None):
    rng = np.random.default_rng(seed); n = len(y_true); vals = np.empty(B, float)
    for b in range(B):
        idx = rng.integers(0, n, n)
        yt, ys = y_true[idx], y_score[idx]
        if threshold is None:
            vals[b] = metric_fn(yt, ys)
        else:
            yp = (ys >= threshold).astype(int); vals[b] = metric_fn(yt, yp)
    lo, hi = np.quantile(vals, [(1-ci)/2, 1-(1-ci)/2]); return float(lo), float(hi)

```

```

def bootstrap_ci_preds(y_true, y_pred, metric_fn, B=B_BOOT, ci=0.95,
seed=RANDOM_STATE):
    rng = np.random.default_rng(seed); n = len(y_true); vals = np.empty(B, float)
    for b in range(B):
        idx = rng.integers(0, n, n); yt, yp = y_true[idx], y_pred[idx]
        vals[b] = metric_fn(yt, yp)
    lo, hi = np.quantile(vals, [(1-ci)/2, 1-(1-ci)/2]); return float(lo), float(hi)

```

```

print("☑ 9-Setup tamam (parametre uzayları, yardımcıları, kütüphaneler).")
def oof_save(name, proba, y, dir="/content/oof_cache"):
    import os, numpy as np
    os.makedirs(dir, exist_ok=True)
    np.save(os.path.join(dir, f"{name}_oof.npy"), np.asarray(proba).ravel())
    np.save(os.path.join(dir, f"y_clf.npy"), np.asarray(y).ravel())
    print(f"📁 OOF kaydedildi → {dir}/{name}_oof.npy (len={len(proba)})")
# 9A) CLF — RandomForest (Nested CV + HPO + Kalibrasyon) — KENDİNİ
KURTARAN TEK HÜCRE
# Tüm çıktılar Colab'da; dosyaya kaydetme yok.

# ===== IMPORTLAR =====
import os, re, warnings; warnings.filterwarnings("ignore")
import numpy as np, pandas as pd
import matplotlib.pyplot as plt
from IPython.display import display, HTML

from sklearn.model_selection import RepeatedStratifiedKFold, StratifiedKFold,
RandomizedSearchCV
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.calibration import CalibratedClassifierCV, calibration_curve
from sklearn.metrics import (roc_auc_score, average_precision_score, f1_score,
recall_score,
                             brier_score_loss, roc_curve, precision_recall_curve)
from sklearn.ensemble import RandomForestClassifier

# ===== 0) DF YOKSA: upload aç ve CSV'yi oku (encoding otomatik)
=====

```

```

if 'df' not in globals():
    try:
        from google.colab import files
        print("📁 df yok: Upload penceresi açılıyor...")
        up = files.upload()
        assert len(up) > 0, "Dosya seçilmedi."
        fname = next(iter(up.keys()))
        ascii_name = re.sub(r"[^A-Za-z0-9_.-]+", "_", fname)
        if ascii_name != fname:
            os.rename(fname, ascii_name)
            fname = ascii_name
        used_enc = None
        for enc in ["utf-8", "utf-8-sig", "ISO-8859-9", "cp1254", "latin-1"]:
            try:
                df = pd.read_csv(fname, encoding=enc)
                used_enc = enc; break
            except Exception:
                pass
        assert 'df' in globals(), "CSV okunamadı; başka encoding deneyin."
        print(f"✅ CSV okundu: {fname} | encoding={used_enc} |
shape={df.shape}")
    except Exception as e:
        raise RuntimeError("df tanımlı değil ve yükleme yapılamadı.") from e
    else:
        print(f"✅ df zaten bellekte: shape={df.shape}")

# ===== 1) Hedefler ve veri hazırlığı (eksikse oluştur) =====
TARGET_CLF = "AHI_SINIF"
TARGET_REG = "AHI_TOPLAM"

for need in [TARGET_CLF, TARGET_REG]:

```

```

    assert need in df.columns, f"Hedef kolon yok: {need}. Mevcut:
{list(df.columns)}"

if 'X_all' not in globals() or 'y_clf' not in globals() or 'preprocess' not in globals():
    X_all = df.drop(columns=[TARGET_CLF, TARGET_REG]).copy()
    y_clf = pd.to_numeric(df[TARGET_CLF], errors="coerce").astype(int).values

    num_cols = X_all.select_dtypes(include=["number"]).columns.tolist()
    cat_cols =
X_all.select_dtypes(include=["object", "category", "bool"]).columns.tolist()

    preprocess = ColumnTransformer(
        transformers=[
            ("num", SimpleImputer(strategy="median"), num_cols),
            ("cat", Pipeline([("imp", SimpleImputer(strategy="most_frequent")),
                ("ohe", OneHotEncoder(handle_unknown="ignore"))]),
cat_cols)
        ],
        remainder="drop"
    )

    print(f" 🧩 Özellikler kuruldu — toplam={X_all.shape[1]} |
sayısal={len(num_cols)} | kategorik={len(cat_cols)}")

# ===== 2) CV sabitleri ve yardımcıları (eksikse tanımla) =====
if 'RANDOM_STATE' not in globals(): RANDOM_STATE = 42
if 'N_OUTER' not in globals(): N_OUTER = 5
if 'N_REPEATS' not in globals(): N_REPEATS = 2
if 'N_INNER' not in globals(): N_INNER = 3
if 'N_ITER' not in globals(): N_ITER = 40
if 'B_BOOT' not in globals(): B_BOOT = 300

```

```

def bootstrap_ci_scores(y_true, y_score, metric_fn, B=B_BOOT, ci=0.95,
seed=RANDOM_STATE, threshold=None):
    rng = np.random.default_rng(seed); n = len(y_true); vals = np.empty(B, float)
    y_true = np.asarray(y_true); y_score = np.asarray(y_score)
    for b in range(B):
        idx = rng.integers(0, n, n)
        yt, ys = y_true[idx], y_score[idx]
        if threshold is None:
            vals[b] = metric_fn(yt, ys)
        else:
            yp = (ys >= threshold).astype(int)
            vals[b] = metric_fn(yt, yp)
    lo, hi = np.quantile(vals, [(1-ci)/2, 1-(1-ci)/2])
    return float(lo), float(hi)

def clf_spaces(y_train):
    pos = int((y_train==1).sum()); neg = int((y_train==0).sum())
    spw = max(1.0, neg / max(1, pos)) # XGB'de lazım ama RF için değil; şablon
    aynı kalsın
    return {
        "rf": {
            "model": RandomForestClassifier(random_state=RANDOM_STATE,
n_jobs=-1),
            "params": {
                "model__n_estimators": [200, 300, 400, 500],
                "model__max_depth": [None, 6, 10, 14, 18],
                "model__min_samples_split": [2, 5, 10, 20],
                "model__min_samples_leaf": [1, 2, 4, 8],
                "model__class_weight": [None, "balanced"]
            }
        }
    }

```

```

# ===== 3) RF — Nested CV + HPO + Kalibrasyon + OOF =====
oof_rf = np.zeros(len(X_all))
rskf = RepeatedStratifiedKFold(n_splits=N_OUTER, n_repeats=N_REPEATS,
random_state=RANDOM_STATE)
spaces = clf_spaces(y_clf)

fold = 0
for tr_idx, te_idx in rskf.split(X_all, y_clf):
    fold += 1
    X_tr, X_te = X_all.iloc[tr_idx], X_all.iloc[te_idx]
    y_tr, y_te = y_clf[tr_idx], y_clf[te_idx]

    # Preprocess fit/transform (outer train'de fit)
    X_tr_p = preprocess.fit_transform(X_tr, y_tr)
    X_te_p = preprocess.transform(X_te)

    # Inner HPO
    pipe = Pipeline([("model", spaces["rf"]["model"])]))
    inner = StratifiedKFold(n_splits=N_INNER, shuffle=True,
random_state=RANDOM_STATE)
    search = RandomizedSearchCV(
        estimator=pipe, param_distributions=spaces["rf"]["params"],
        n_iter=N_ITER, scoring="roc_auc", cv=inner,
        n_jobs=-1, random_state=RANDOM_STATE, refit=True, verbose=0
    )
    search.fit(X_tr_p, y_tr)
    best = search.best_estimator_

    # Isotonic kalibrasyon (3-fold)
    calib = CalibratedClassifierCV(best, method="isotonic", cv=3)
    calib.fit(X_tr_p, y_tr)

```

```

# OOF olasılıkları
oof_rf[te_idx] = calib.predict_proba(X_te_p)[:, 1]
print(f"RF fold {fold} tamam.")

# ===== 4) OOF metrikler + %95 GA =====
roc = roc_auc_score(y_clf, oof_rf)
pr = average_precision_score(y_clf, oof_rf)
f1v = f1_score(y_clf, (oof_rf>=0.5).astype(int))
rec = recall_score(y_clf, (oof_rf>=0.5).astype(int))
br = brier_score_loss(y_clf, oof_rf)

roc_lo, roc_hi = bootstrap_ci_scores(y_clf, oof_rf, roc_auc_score, seed=1)
pr_lo, pr_hi = bootstrap_ci_scores(y_clf, oof_rf, average_precision_score,
seed=2)

display(HTML("<h4>9A — RandomForest OOF Metrikler</h4>"))
display(pd.DataFrame([{
    "Model": "RandomForest",
    "ROC_AUC": roc, "ROC_AUC_CI95": f"[{roc_lo:.3f}, {roc_hi:.3f}]",
    "PR_AUC": pr, "PR_AUC_CI95": f"[{pr_lo:.3f}, {pr_hi:.3f}]",
    "F1@0.5": f1v, "Recall@0.5": rec, "Brier": br
}]))

# ===== 5) ROC / PR / Kalibrasyon grafikleri =====
fpr, tpr, _ = roc_curve(y_clf, oof_rf)
plt.figure(); plt.plot(fpr, tpr, label=f"AUC={roc:.3f}")
plt.plot([0,1],[0,1], '--'); plt.xlabel("FPR"); plt.ylabel("TPR")
plt.title("RandomForest — OOF ROC"); plt.legend(); plt.show()

prec, rec_c, _ = precision_recall_curve(y_clf, oof_rf)
plt.figure(); plt.plot(rec_c, prec, label=f"AP={pr:.3f}")

```

```
plt.xlabel("Recall"); plt.ylabel("Precision"); plt.title("RandomForest — OOF  
PR"); plt.legend(); plt.show()
```

```
prob_true, prob_pred = calibration_curve(y_clf, oof_rf, n_bins=10,  
strategy="quantile")  
plt.figure(); plt.plot(prob_pred, prob_true, marker="o"); plt.plot([0,1],[0,1], '--')  
plt.xlabel("Tahmin Olasılığı"); plt.ylabel("Gerçek Olumlu Oranı")  
plt.title("RandomForest — OOF Kalibrasyon"); plt.show()
```

```
oof_save("rf", oof_rf, y_clf) # 9A sonunda  
# 9B) CLF — XGBoost (Nested CV + HPO + Kalibrasyon) — KENDİNİ  
KURTARAN TEK HÜCRE  
# Tüm çıktılar Colab'da; dosyaya kaydetme yok.
```

```
# ===== IMPORTLAR =====  
import os, re, warnings; warnings.filterwarnings("ignore")  
import numpy as np, pandas as pd  
import matplotlib.pyplot as plt  
from IPython.display import display, HTML  
  
from sklearn.model_selection import RepeatedStratifiedKFold, StratifiedKFold,  
RandomizedSearchCV  
from sklearn.pipeline import Pipeline  
from sklearn.compose import ColumnTransformer  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import OneHotEncoder  
from sklearn.calibration import CalibratedClassifierCV, calibration_curve  
from sklearn.metrics import (roc_auc_score, average_precision_score, f1_score,  
recall_score,  
brier_score_loss, roc_curve, precision_recall_curve)  
  
# xgboost yoksa kur
```

```

try:
    from xgboost import XGBClassifier
except Exception:
    os.system("pip -q install xgboost")
    from xgboost import XGBClassifier

# ===== 0) DF YOKSA: upload aç ve CSV'yi oku (encoding otomatik)
=====

if 'df' not in globals():
    try:
        from google.colab import files
        print("📁 df yok: Upload penceresi açılıyor...")
        up = files.upload()
        assert len(up) > 0, "Dosya seçilmedi."
        fname = next(iter(up.keys()))
        ascii_name = re.sub(r"^[A-Za-z0-9_.-]+", "_", fname)
        if ascii_name != fname:
            os.rename(fname, ascii_name)
            fname = ascii_name
        used_enc = None
        for enc in ["utf-8", "utf-8-sig", "ISO-8859-9", "cp1254", "latin-1"]:
            try:
                df = pd.read_csv(fname, encoding=enc)
                used_enc = enc; break
            except Exception:
                pass
        assert 'df' in globals(), "CSV okunamadı; başka encoding deneyin."
        print(f"✅ CSV okundu: {fname} | encoding={used_enc} |
shape={df.shape}")
    except Exception as e:
        raise RuntimeError("df tanımlı değil ve yükleme yapılamadı.") from e
else:

```

```

print(f"✅ df zaten bellekte: shape={df.shape}")

# ===== 1) Hedefler ve veri hazırlığı (eksikse oluştur) =====
TARGET_CLF = "AHI_SINIF"
TARGET_REG = "AHI_TOPLAM"

for need in [TARGET_CLF, TARGET_REG]:
    assert need in df.columns, f"Hedef kolon yok: {need}. Mevcut: {list(df.columns)}"

if 'X_all' not in globals() or 'y_clf' not in globals() or 'preprocess' not in globals():
    X_all = df.drop(columns=[TARGET_CLF, TARGET_REG]).copy()
    y_clf = pd.to_numeric(df[TARGET_CLF], errors="coerce").astype(int).values

    num_cols = X_all.select_dtypes(include=["number"]).columns.tolist()
    cat_cols =
X_all.select_dtypes(include=["object", "category", "bool"]).columns.tolist()

    preprocess = ColumnTransformer(
        transformers=[
            ("num", SimpleImputer(strategy="median"), num_cols),
            ("cat", Pipeline([("imp", SimpleImputer(strategy="most_frequent")),
                ("ohe", OneHotEncoder(handle_unknown="ignore"))])),
cat_cols)
        ],
        remainder="drop"
    )

    print(f"🧩 Özellikler kuruldu — toplam={X_all.shape[1]} |
sayısal={len(num_cols)} | kategorik={len(cat_cols)}")

# ===== 2) CV sabitleri ve yardımcıları (eksikse tanımla) =====
if 'RANDOM_STATE' not in globals(): RANDOM_STATE = 42

```

```

if 'N_OUTER' not in globals():    N_OUTER = 5
if 'N_REPEATS' not in globals(): N_REPEATS = 2
if 'N_INNER' not in globals():    N_INNER = 3
if 'N_ITER' not in globals():     N_ITER = 40
if 'B_BOOT' not in globals():     B_BOOT = 300

def bootstrap_ci_scores(y_true, y_score, metric_fn, B=B_BOOT, ci=0.95,
seed=RANDOM_STATE, threshold=None):
    rng = np.random.default_rng(seed); n = len(y_true); vals = np.empty(B, float)
    y_true = np.asarray(y_true); y_score = np.asarray(y_score)
    for b in range(B):
        idx = rng.integers(0, n, n)
        yt, ys = y_true[idx], y_score[idx]
        if threshold is None:
            vals[b] = metric_fn(yt, ys)
        else:
            yp = (ys >= threshold).astype(int)
            vals[b] = metric_fn(yt, yp)
    lo, hi = np.quantile(vals, [(1-ci)/2, 1-(1-ci)/2])
    return float(lo), float(hi)

def xgb_param_space_for(y_train):
    """Sınıf dengesine göre scale_pos_weight ayarlar; HPO uzayı döner."""
    pos = int((y_train==1).sum()); neg = int((y_train==0).sum())
    spw = max(1.0, neg / max(1, pos))
    return {
        "model__n_estimators": [300, 500, 700],
        "model__max_depth": [3, 4, 6, 8],
        "model__learning_rate": [0.02, 0.05, 0.1, 0.2],
        "model__subsample": [0.7, 0.85, 1.0],
        "model__colsample_bytree": [0.6, 0.8, 1.0],
        "model__min_child_weight": [1, 3, 6, 10],
    }

```

```

    "model__gamma": [0, 0.5, 1, 3],
    "model__scale_pos_weight": [spw],
}

# ===== 3) XGB — Nested CV + HPO + Kalibrasyon + OOF =====
oof_xgb = np.zeros(len(X_all))
rskf = RepeatedStratifiedKFold(n_splits=N_OUTER, n_repeats=N_REPEATS,
random_state=RANDOM_STATE)

fold = 0
for tr_idx, te_idx in rskf.split(X_all, y_clf):
    fold += 1
    X_tr, X_te = X_all.iloc[tr_idx], X_all.iloc[te_idx]
    y_tr, y_te = y_clf[tr_idx], y_clf[te_idx]

    # Preprocess fit/transform (outer train'de fit)
    X_tr_p = preprocess.fit_transform(X_tr, y_tr)
    X_te_p = preprocess.transform(X_te)

    # Inner HPO (fold bazlı spw)
    pipe = Pipeline([("model", XGBClassifier(random_state=RANDOM_STATE,
eval_metric="logloss",
tree_method="hist", n_jobs=-1))])

    inner = StratifiedKFold(n_splits=N_INNER, shuffle=True,
random_state=RANDOM_STATE)
    search = RandomizedSearchCV(
        estimator=pipe, param_distributions=xgb_param_space_for(y_tr),
        n_iter=N_ITER, scoring="roc_auc", cv=inner,
        n_jobs=-1, random_state=RANDOM_STATE, refit=True, verbose=0
    )
    search.fit(X_tr_p, y_tr)
    best = search.best_estimator_

```

```

# Isotonic kalibrasyon (3-fold)
calib = CalibratedClassifierCV(best, method="isotonic", cv=3)
calib.fit(X_tr_p, y_tr)

# OOF olasılıkları
oof_xgb[te_idx] = calib.predict_proba(X_te_p)[:, 1]
print(f"XGB fold {fold} tamam.")

# ===== 4) OOF metrikler + %95 GA =====
roc = roc_auc_score(y_clf, oof_xgb)
pr = average_precision_score(y_clf, oof_xgb)
f1v = f1_score(y_clf, (oof_xgb>=0.5).astype(int))
rec = recall_score(y_clf, (oof_xgb>=0.5).astype(int))
br = brier_score_loss(y_clf, oof_xgb)

roc_lo, roc_hi = bootstrap_ci_scores(y_clf, oof_xgb, roc_auc_score, seed=11)
pr_lo, pr_hi = bootstrap_ci_scores(y_clf, oof_xgb, average_precision_score,
seed=12)

display(HTML("<h4>9B — XGBoost OOF Metrikler</h4>"))
display(pd.DataFrame([{
    "Model": "XGBoost",
    "ROC_AUC": roc, "ROC_AUC_CI95": f"[{roc_lo:.3f}, {roc_hi:.3f}]",
    "PR_AUC": pr, "PR_AUC_CI95": f"[{pr_lo:.3f}, {pr_hi:.3f}]",
    "F1@0.5": f1v, "Recall@0.5": rec, "Brier": br
}]))

# ===== 5) ROC / PR / Kalibrasyon grafikleri =====
fpr, tpr, _ = roc_curve(y_clf, oof_xgb)
plt.figure(); plt.plot(fpr, tpr, label=f"AUC={roc:.3f}")

```

```
plt.plot([0,1],[0,1], '--'); plt.xlabel("False Positive Rate"); plt.ylabel("True Positive Rate")
```

```
plt.title("XGBoost — OOF ROC"); plt.legend(); plt.show()
```

```
prec, rec_c, _ = precision_recall_curve(y_clf, oof_xgb)
```

```
plt.figure(); plt.plot(rec_c, prec, label=f"AP={pr:.3f}")
```

```
plt.xlabel("Recall"); plt.ylabel("Precision"); plt.title("XGBoost — OOF PR");
```

```
plt.legend(); plt.show()
```

```
prob_true, prob_pred = calibration_curve(y_clf, oof_xgb, n_bins=10,  
strategy="quantile")
```

```
plt.figure(); plt.plot(prob_pred, prob_true, marker="o"); plt.plot([0,1],[0,1], '--')
```

```
plt.xlabel("Tahmin Olasılığı"); plt.ylabel("Gerçek Olumlu Oranı")
```

```
plt.title("XGBoost — OOF Kalibrasyon"); plt.show()
```

```
oof_save("xgb", oof_xgb, y_clf) # 9B sonunda
```

```
# 9B) CLF — XGBoost (Nested CV + HPO + Kalibrasyon) — KENDİNİ  
KURTARAN TEK HÜCRE
```

```
# Tüm çıktılar Colab'da; dosyaya kaydetme yok.
```

```
# ===== IMPORTLAR =====
```

```
import os, re, warnings; warnings.filterwarnings("ignore")
```

```
import numpy as np, pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
from IPython.display import display, HTML
```

```
from sklearn.model_selection import RepeatedStratifiedKFold, StratifiedKFold,  
RandomizedSearchCV
```

```
from sklearn.pipeline import Pipeline
```

```
from sklearn.compose import ColumnTransformer
```

```
from sklearn.impute import SimpleImputer
```

```

from sklearn.preprocessing import OneHotEncoder
from sklearn.calibration import CalibratedClassifierCV, calibration_curve
from sklearn.metrics import (roc_auc_score, average_precision_score, f1_score,
recall_score,
                             brier_score_loss, roc_curve, precision_recall_curve)

# xgboost yoksa kur
try:
    from xgboost import XGBClassifier
except Exception:
    os.system("pip -q install xgboost")
    from xgboost import XGBClassifier

# ===== 0) DF YOKSA: upload aç ve CSV'yi oku (encoding otomatik)
=====
if 'df' not in globals():
    try:
        from google.colab import files
        print("📁 df yok: Upload penceresi açılıyor...")
        up = files.upload()
        assert len(up) > 0, "Dosya seçilmedi."
        fname = next(iter(up.keys()))
        ascii_name = re.sub(r"^[A-Za-z0-9_.-]+", "_", fname)
        if ascii_name != fname:
            os.rename(fname, ascii_name)
            fname = ascii_name
        used_enc = None
        for enc in ["utf-8", "utf-8-sig", "ISO-8859-9", "cp1254", "latin-1"]:
            try:
                df = pd.read_csv(fname, encoding=enc)
                used_enc = enc; break
            except Exception:

```

```

        pass
        assert 'df' in globals(), "CSV okunamadı; başka encoding deneyin."
        print(f"☑ CSV okundu: {fname} | encoding={used_enc} |
shape={df.shape}")
    except Exception as e:
        raise RuntimeError("df tanımlı değil ve yükleme yapılamadı.") from e
else:
    print(f"☑ df zaten bellekte: shape={df.shape}")

# ===== 1) Hedefler ve veri hazırlığı (eksikse oluştur) =====
TARGET_CLF = "AHI_SINIF"
TARGET_REG = "AHI_TOPLAM"

for need in [TARGET_CLF, TARGET_REG]:
    assert need in df.columns, f"Hedef kolon yok: {need}. Mevcut:
{list(df.columns)}"

if 'X_all' not in globals() or 'y_clf' not in globals() or 'preprocess' not in globals():
    X_all = df.drop(columns=[TARGET_CLF, TARGET_REG]).copy()
    y_clf = pd.to_numeric(df[TARGET_CLF], errors="coerce").astype(int).values

    num_cols = X_all.select_dtypes(include=["number"]).columns.tolist()
    cat_cols =
X_all.select_dtypes(include=["object", "category", "bool"]).columns.tolist()

    preprocess = ColumnTransformer(
        transformers=[
            ("num", SimpleImputer(strategy="median"), num_cols),
            ("cat", Pipeline([("imp", SimpleImputer(strategy="most_frequent")),
                ("ohe", OneHotEncoder(handle_unknown="ignore"))])),
cat_cols)
        ],

```

```

        remainder="drop"
    )

    print(f" 🧩 Özellikler kuruldu — toplam={X_all.shape[1]} |
sayısal={len(num_cols)} | kategorik={len(cat_cols)}")

# ===== 2) CV sabitleri ve yardımcıları (eksikse tanımla) =====
if 'RANDOM_STATE' not in globals(): RANDOM_STATE = 42
if 'N_OUTER' not in globals():    N_OUTER = 5
if 'N_REPEATS' not in globals():  N_REPEATS = 2
if 'N_INNER' not in globals():    N_INNER = 3
if 'N_ITER' not in globals():     N_ITER = 40
if 'B_BOOT' not in globals():     B_BOOT = 300

def bootstrap_ci_scores(y_true, y_score, metric_fn, B=B_BOOT, ci=0.95,
seed=RANDOM_STATE, threshold=None):
    rng = np.random.default_rng(seed); n = len(y_true); vals = np.empty(B, float)
    y_true = np.asarray(y_true); y_score = np.asarray(y_score)
    for b in range(B):
        idx = rng.integers(0, n, n)
        yt, ys = y_true[idx], y_score[idx]
        if threshold is None:
            vals[b] = metric_fn(yt, ys)
        else:
            yp = (ys >= threshold).astype(int)
            vals[b] = metric_fn(yt, yp)
    lo, hi = np.quantile(vals, [(1-ci)/2, 1-(1-ci)/2])
    return float(lo), float(hi)

def xgb_param_space_for(y_train):
    """Sınıf dengesine göre scale_pos_weight ayarlar; HPO uzayı döner."""
    pos = int((y_train==1).sum()); neg = int((y_train==0).sum())
    spw = max(1.0, neg / max(1, pos))

```

```

return {
    "model__n_estimators": [300, 500, 700],
    "model__max_depth": [3, 4, 6, 8],
    "model__learning_rate": [0.02, 0.05, 0.1, 0.2],
    "model__subsample": [0.7, 0.85, 1.0],
    "model__colsample_bytree": [0.6, 0.8, 1.0],
    "model__min_child_weight": [1, 3, 6, 10],
    "model__gamma": [0, 0.5, 1, 3],
    "model__scale_pos_weight": [spw],
}

# ===== 3) XGB — Nested CV + HPO + Kalibrasyon + OOF =====
oof_xgb = np.zeros(len(X_all))
rskf = RepeatedStratifiedKFold(n_splits=N_OUTER, n_repeats=N_REPEATS,
random_state=RANDOM_STATE)

fold = 0
for tr_idx, te_idx in rskf.split(X_all, y_clf):
    fold += 1
    X_tr, X_te = X_all.iloc[tr_idx], X_all.iloc[te_idx]
    y_tr, y_te = y_clf[tr_idx], y_clf[te_idx]

    # Preprocess fit/transform (outer train'de fit)
    X_tr_p = preprocess.fit_transform(X_tr, y_tr)
    X_te_p = preprocess.transform(X_te)

    # Inner HPO (fold bazlı spw)
    pipe = Pipeline([("model", XGBClassifier(random_state=RANDOM_STATE,
eval_metric="logloss",
tree_method="hist", n_jobs=-1))])

    inner = StratifiedKFold(n_splits=N_INNER, shuffle=True,
random_state=RANDOM_STATE)

```

```

search = RandomizedSearchCV(
    estimator=pipe, param_distributions=xgb_param_space_for(y_tr),
    n_iter=N_ITER, scoring="roc_auc", cv=inner,
    n_jobs=-1, random_state=RANDOM_STATE, refit=True, verbose=0
)
search.fit(X_tr_p, y_tr)
best = search.best_estimator_

# Isotonic kalibrasyon (3-fold)
calib = CalibratedClassifierCV(best, method="isotonic", cv=3)
calib.fit(X_tr_p, y_tr)

# OOF olasılıkları
oof_xgb[te_idx] = calib.predict_proba(X_te_p)[:, 1]
print(f"XGB fold {fold} tamam.")

# ===== 4) OOF metrikler + %95 GA =====
roc = roc_auc_score(y_clf, oof_xgb)
pr = average_precision_score(y_clf, oof_xgb)
f1v = f1_score(y_clf, (oof_xgb>=0.5).astype(int))
rec = recall_score(y_clf, (oof_xgb>=0.5).astype(int))
br = brier_score_loss(y_clf, oof_xgb)

roc_lo, roc_hi = bootstrap_ci_scores(y_clf, oof_xgb, roc_auc_score, seed=11)
pr_lo, pr_hi = bootstrap_ci_scores(y_clf, oof_xgb, average_precision_score,
seed=12)

display(HTML("<h4>9B — XGBoost OOF Metrikler</h4>"))
display(pd.DataFrame([
    "Model": "XGBoost",
    "ROC_AUC": roc, "ROC_AUC_CI95": f"[{roc_lo:.3f}, {roc_hi:.3f}]",
    "PR_AUC": pr, "PR_AUC_CI95": f"[{pr_lo:.3f}, {pr_hi:.3f}]",

```

```

    "F1@0.5": f1v, "Recall@0.5": rec, "Brier": br
    })))

# ===== 5) ROC / PR / Kalibrasyon grafikleri =====
fpr, tpr, _ = roc_curve(y_clf, oof_xgb)
plt.figure(); plt.plot(fpr, tpr, label=f"AUC={roc:.3f}")
plt.plot([0,1],[0,1], '--'); plt.xlabel("False Positive Rate"); plt.ylabel("True Positive
Rate")
plt.title("XGBoost — OOF ROC"); plt.legend(); plt.show()

prec, rec_c, _ = precision_recall_curve(y_clf, oof_xgb)
plt.figure(); plt.plot(rec_c, prec, label=f"AP={pr:.3f}")
plt.xlabel("Recall"); plt.ylabel("Precision"); plt.title("XGBoost — OOF PR");
plt.legend(); plt.show()

prob_true, prob_pred = calibration_curve(y_clf, oof_xgb, n_bins=10,
strategy="quantile")
plt.figure(); plt.plot(prob_pred, prob_true, marker="o"); plt.plot([0,1],[0,1], '--')
plt.xlabel("Tahmin Olasılığı"); plt.ylabel("Gerçek Olumlu Oranı")
plt.title("XGBoost — OOF Kalibrasyon"); plt.show()

oof_save("xgb", oof_xgb, y_clf) # 9B sonunda

# 9C) CLF — CatBoost (Nested CV + HPO + Kalibrasyon) — KENDİNİ
KURTARAN TEK HÜCRE
# Tüm çıktılar Colab'da; dosyaya kaydetme yok.

# ===== IMPORTLAR =====
import os, re, warnings; warnings.filterwarnings("ignore")
import numpy as np, pandas as pd
import matplotlib.pyplot as plt

```

```

from IPython.display import display, HTML

from sklearn.model_selection import RepeatedStratifiedKFold, StratifiedKFold,
RandomizedSearchCV
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.calibration import CalibratedClassifierCV, calibration_curve
from sklearn.metrics import (roc_auc_score, average_precision_score, f1_score,
recall_score,
                             brier_score_loss, roc_curve, precision_recall_curve)

# catboost yoksa kur
try:
    from catboost import CatBoostClassifier
except Exception:
    os.system("pip -q install catboost")
    from catboost import CatBoostClassifier

# ===== 0) DF YOKSA: upload aç ve CSV'yi oku (encoding otomatik)
=====
if 'df' not in globals():
    try:
        from google.colab import files
        print("📁 df yok: Upload penceresi açılıyor...")
        up = files.upload()
        assert len(up) > 0, "Dosya seçilmedi."
        fname = next(iter(up.keys()))
        ascii_name = re.sub(r"[^A-Za-z0-9_.-]+", "_", fname)
        if ascii_name != fname:
            os.rename(fname, ascii_name)

```

```

    fname = ascii_name
    used_enc = None
    for enc in ["utf-8", "utf-8-sig", "ISO-8859-9", "cp1254", "latin-1"]:
        try:
            df = pd.read_csv(fname, encoding=enc)
            used_enc = enc; break
        except Exception:
            pass
    assert 'df' in globals(), "CSV okunamadı; başka encoding deneyin."
    print(f"☑ CSV okundu: {fname} | encoding={used_enc} |
shape={df.shape}")
    except Exception as e:
        raise RuntimeError("df tanımlı değil ve yükleme yapılamadı.") from e
    else:
        print(f"☑ df zaten bellekte: shape={df.shape}")

# ===== 1) Hedefler ve veri hazırlığı (eksikse oluştur) =====
TARGET_CLF = "AHI_SINIF"
TARGET_REG = "AHI_TOPLAM"

for need in [TARGET_CLF, TARGET_REG]:
    assert need in df.columns, f"Hedef kolon yok: {need}. Mevcut:
{list(df.columns)}"

if 'X_all' not in globals() or 'y_clf' not in globals() or 'preprocess' not in globals():
    X_all = df.drop(columns=[TARGET_CLF, TARGET_REG]).copy()
    y_clf = pd.to_numeric(df[TARGET_CLF], errors="coerce").astype(int).values

    num_cols = X_all.select_dtypes(include=["number"]).columns.tolist()
    cat_cols =
X_all.select_dtypes(include=["object", "category", "bool"]).columns.tolist()

```

```

preprocess = ColumnTransformer(
    transformers=[
        ("num", SimpleImputer(strategy="median"), num_cols),
        ("cat", Pipeline([("imp", SimpleImputer(strategy="most_frequent")),
                          ("ohe", OneHotEncoder(handle_unknown="ignore"))]),
        cat_cols)
    ],
    remainder="drop"
)

print(f" 🧩 Özellikler kuruldu — toplam={X_all.shape[1]} |
sayısal={len(num_cols)} | kategorik={len(cat_cols)}")

# ===== 2) CV sabitleri ve yardımcıları (eksikse tanımla) =====
if 'RANDOM_STATE' not in globals(): RANDOM_STATE = 42
if 'N_OUTER' not in globals(): N_OUTER = 5
if 'N_REPEATS' not in globals(): N_REPEATS = 2
if 'N_INNER' not in globals(): N_INNER = 3
if 'N_ITER' not in globals(): N_ITER = 40
if 'B_BOOT' not in globals(): B_BOOT = 300

def bootstrap_ci_scores(y_true, y_score, metric_fn, B=B_BOOT, ci=0.95,
seed=RANDOM_STATE, threshold=None):
    rng = np.random.default_rng(seed); n = len(y_true); vals = np.empty(B, float)
    y_true = np.asarray(y_true); y_score = np.asarray(y_score)
    for b in range(B):
        idx = rng.integers(0, n, n)
        yt, ys = y_true[idx], y_score[idx]
        if threshold is None:
            vals[b] = metric_fn(yt, ys)
        else:
            yp = (ys >= threshold).astype(int)
            vals[b] = metric_fn(yt, yp)

```

```

lo, hi = np.quantile(vals, [(1-ci)/2, 1-(1-ci)/2])
return float(lo), float(hi)

# CatBoost HPO uzayı
param_cat = {
    "model__iterations": [400, 600, 800],
    "model__depth": [4, 6, 8, 10],
    "model__learning_rate": [0.02, 0.05, 0.1, 0.2],
    "model__l2_leaf_reg": [1, 3, 5, 10, 15],
    "model__border_count": [64, 128, 255]
}

# ===== 3) CatBoost — Nested CV + HPO + Kalibrasyon + OOF =====
oof_cat = np.zeros(len(X_all))
rskf = RepeatedStratifiedKFold(n_splits=N_OUTER, n_repeats=N_REPEATS,
random_state=RANDOM_STATE)

fold = 0
for tr_idx, te_idx in rskf.split(X_all, y_clf):
    fold += 1
    X_tr, X_te = X_all.iloc[tr_idx], X_all.iloc[te_idx]
    y_tr, y_te = y_clf[tr_idx], y_clf[te_idx]

    # Preprocess fit/transform (outer train'de fit)
    X_tr_p = preprocess.fit_transform(X_tr, y_tr)
    X_te_p = preprocess.transform(X_te)

    # Inner HPO
    pipe = Pipeline([("model",
CatBoostClassifier(random_seed=RANDOM_STATE, verbose=0,
loss_function="Logloss"))])

```

```

inner = StratifiedKFold(n_splits=N_INNER, shuffle=True,
random_state=RANDOM_STATE)
search = RandomizedSearchCV(
    estimator=pipe, param_distributions=param_cat,
    n_iter=N_ITER, scoring="roc_auc", cv=inner,
    n_jobs=-1, random_state=RANDOM_STATE, refit=True, verbose=0
)
search.fit(X_tr_p, y_tr)
best = search.best_estimator_

# Isotonic kalibrasyon (3-fold)
calib = CalibratedClassifierCV(best, method="isotonic", cv=3)
calib.fit(X_tr_p, y_tr)

# OOF olasılıkları
oof_cat[te_idx] = calib.predict_proba(X_te_p)[:, 1]
print(f"CatBoost fold {fold} tamam.")

# ===== 4) OOF metrikler + %95 GA =====
roc = roc_auc_score(y_clf, oof_cat)
pr = average_precision_score(y_clf, oof_cat)
f1v = f1_score(y_clf, (oof_cat>=0.5).astype(int))
rec = recall_score(y_clf, (oof_cat>=0.5).astype(int))
br = brier_score_loss(y_clf, oof_cat)

roc_lo, roc_hi = bootstrap_ci_scores(y_clf, oof_cat, roc_auc_score, seed=21)
pr_lo, pr_hi = bootstrap_ci_scores(y_clf, oof_cat, average_precision_score,
seed=22)

display(HTML("<h4>9C — CatBoost OOF Metrikler</h4>"))
display(pd.DataFrame([
    "Model": "CatBoost",

```

```

"ROC_AUC": roc, "ROC_AUC_CI95": f"[{roc_lo:.3f}, {roc_hi:.3f}]",
"PR_AUC": pr, "PR_AUC_CI95": f"[{pr_lo:.3f}, {pr_hi:.3f}]",
"F1@0.5": f1v, "Recall@0.5": rec, "Brier": br
})))

# ===== 5) ROC / PR / Kalibrasyon grafikleri =====
fpr, tpr, _ = roc_curve(y_clf, oof_cat)
plt.figure(); plt.plot(fpr, tpr, label=f"AUC={roc:.3f}")
plt.plot([0,1],[0,1], '--'); plt.xlabel("False Positive Rate"); plt.ylabel("True Positive
Rate")
plt.title("CatBoost — OOF ROC"); plt.legend(); plt.show()

prec, rec_c, _ = precision_recall_curve(y_clf, oof_cat)
plt.figure(); plt.plot(rec_c, prec, label=f"AP={pr:.3f}")
plt.xlabel("Recall"); plt.ylabel("Precision"); plt.title("CatBoost — OOF PR");
plt.legend(); plt.show()

prob_true, prob_pred = calibration_curve(y_clf, oof_cat, n_bins=10,
strategy="quantile")
plt.figure(); plt.plot(prob_pred, prob_true, marker="o"); plt.plot([0,1],[0,1], '--')
plt.xlabel("Tahmin Olasılığı"); plt.ylabel("Gerçek Olumlu Oranı")
plt.title("CatBoost — OOF Kalibrasyon"); plt.show()

oof_save("cat", oof_cat, y_clf) # 9C sonunda
# 9D — Sınıflandırma Toplu Özet (OOF Sonuç Tablolarından)

import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import display, HTML

# Senin paylaştığın 9A, 9B, 9C sonuçlarını elle DataFrame'e giriyoruz
data = [

```

```

{"Model": "RandomForest", "ROC_AUC": 0.96545, "PR_AUC": 0.949156,
"F1@0.5": 0.878274, "Recall@0.5": 0.868902, "Brier": 0.065845},
{"Model": "XGBoost", "ROC_AUC": 0.967313, "PR_AUC": 0.947806,
"F1@0.5": 0.873239, "Recall@0.5": 0.850610, "Brier": 0.064954},
{"Model": "CatBoost", "ROC_AUC": 0.969797, "PR_AUC": 0.951893,
"F1@0.5": 0.876369, "Recall@0.5": 0.853659, "Brier": 0.064265},
]

```

```

df_clf_all = pd.DataFrame(data).sort_values("ROC_AUC",
ascending=False).reset_index(drop=True)

```

```

display(HTML("<h4>9D — Sınıflandırma Toplu Özet (OOF Sonuçları)</h4>"))
display(df_clf_all)

```

```

# --- Grafikler ---

```

```

def _bar(col, title, ylim=None, ylabel=None):
    plt.figure()
    plt.bar(df_clf_all["Model"], df_clf_all[col],
color=["#1f77b4", "#ff7f0e", "#2ca02c"])
    if ylabel: plt.ylabel(ylabel)
    if ylim: plt.ylim(*ylim)
    plt.title(title)
    for i,v in enumerate(df_clf_all[col]):
        plt.text(i, v, f"{v:.3f}", ha="center", va="bottom")
    plt.show()

```

```

_bar("ROC_AUC", "OOF ROC-AUC", (0.9,1), "ROC-AUC")

```

```

_bar("PR_AUC", "OOF PR-AUC", (0.9,1), "PR-AUC")

```

```

_bar("Brier", "OOF Brier", None, "Brier (↓ daha iyi)")

```

9E) REG — RandomForest Regressor (Nested CV + HPO + OOF + özet + grafikler)

```

import os, re, warnings; warnings.filterwarnings("ignore")
import numpy as np, pandas as pd, matplotlib.pyplot as plt
from IPython.display import display, HTML

from sklearn.model_selection import RepeatedKFold, KFold,
RandomizedSearchCV
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestRegressor

# =====
# 0) VERİ & HEDEFLENER
# =====
TARGET_REG = "AHI_TOPLAM"
TARGET_CLF = "AHI_SINIF"

# df yoksa CSV oku (sadece okuma; kaydetme yok)
if 'df' not in globals():
    try:
        from google.colab import files
        print("📁 df yok: CSV seçin...")
        up = files.upload(); assert up, "Dosya seçilmedi."
        fname = next(iter(up.keys()))
        ascii_name = re.sub(r"^[^A-Za-z0-9_.-]+", "_", fname)
        if ascii_name != fname: os.rename(fname, ascii_name); fname = ascii_name
        used=None
        for enc in ["utf-8","utf-8-sig","ISO-8859-9","cp1254","latin-1"]:
            try: df = pd.read_csv(fname, encoding=enc); used=enc; break

```

```

        except: pass
    assert used, "CSV okunamadı."

    print(f"☑ CSV okundu: {fname} | enc={used} | shape={df.shape}")

except Exception as e:
    raise RuntimeError("df yok ve yükleme başarısız.") from e
else:
    print(f"☑ df mevcut: shape={df.shape}")

assert TARGET_REG in df.columns and TARGET_CLF in df.columns,
"AHI_TOPLAM / AHI_SINIF kolonları bulunamadı."

X = df.drop(columns=[TARGET_REG, TARGET_CLF]).copy()
y_reg = pd.to_numeric(df[TARGET_REG], errors="coerce").values

# Süreç: sayısal median impute, kategorik mode+OHE
num_cols = X.select_dtypes(include=["number"]).columns.tolist()
cat_cols = X.select_dtypes(include=["object", "category", "bool"]).columns.tolist()

preprocess = ColumnTransformer(
    transformers=[
        ("num", SimpleImputer(strategy="median"), num_cols),
        ("cat", Pipeline([
            ("imp", SimpleImputer(strategy="most_frequent")),
            ("ohe", OneHotEncoder(handle_unknown="ignore"))
        ]), cat_cols),
    ],
    remainder="drop"
)

# =====
# 1) Nested CV + HPO ayarları
# =====

```

```

N_SPLITS_OUTER = 5
N_REPEATS_OUTER = 2 # (vakit varsa 3 yapabilirsin)
N_SPLITS_INNER = 3
N_ITER = 20 # HPO örnek sayısı (vakit durumuna göre arttırılabilir)
RANDOM_STATE = 42

outer_cv = RepeatedKfold(n_splits=N_SPLITS_OUTER,
n_repeats=N_REPEATS_OUTER, random_state=RANDOM_STATE)
inner_cv = KFold(n_splits=N_SPLITS_INNER, shuffle=True,
random_state=RANDOM_STATE)

base = RandomForestRegressor(random_state=RANDOM_STATE, n_jobs=-1)

param_space = {
    "model__n_estimators": [300, 500, 800],
    "model__max_depth": [None, 8, 16, 32],
    "model__min_samples_split": [2, 5, 10],
    "model__min_samples_leaf": [1, 2, 4],
    # sklearn sürüm farklarını dert etmeyelim: max_features'ı güvenli seçelim
    "model__max_features": [1.0, "sqrt", 0.7],
    "model__bootstrap": [True, False],
}

pipe = Pipeline([("prep", preprocess), ("model", base)])

# Not: Sklearn sürüm uyumu için skor "neg_mean_squared_error" seçiyoruz
(RMSE ile uyumlu)
search = RandomizedSearchCV(
    estimator=pipe,
    param_distributions=param_space,
    n_iter=N_ITER,
    scoring="neg_mean_squared_error",

```

```

cv=inner_cv,
refit=True,
random_state=RANDOM_STATE,
n_jobs=-1,
verbose=0
)

# =====
# 2) Outer döngü: OOF tahmin + fold metrikleri + en iyi paramlar
# =====

n = len(y_reg)
oof_pred = np.zeros(n, dtype=float)
fold_rows = []
best_rows = []

fold_id = 0
for tr_idx, te_idx in outer_cv.split(X, y_reg):
    fold_id += 1
    X_tr, X_te = X.iloc[tr_idx], X.iloc[te_idx]
    y_tr, y_te = y_reg[tr_idx], y_reg[te_idx]

    # İç arama (HPO)
    search.fit(X_tr, y_tr)
    best_est = search.best_estimator_

    # Test folduna tahmin
    y_hat = best_est.predict(X_te)
    oof_pred[te_idx] = y_hat

    # Fold metrikleri
    rmse = np.sqrt(mean_squared_error(y_te, y_hat))
    mae = mean_absolute_error(y_te, y_hat)

```

```

r2 = r2_score(y_te, y_hat)

fold_rows.append({
    "Fold": fold_id,
    "RMSE": rmse,
    "MAE": mae,
    "R2": r2
})

# En iyi paramlar (özet)
best_rows.append({
    "Fold": fold_id,
    **{k.replace("model__", ""): v for k,v in search.best_params_.items()}
})

fold_df = pd.DataFrame(fold_rows)
best_df = pd.DataFrame(best_rows)

# =====
# 3) OOF Genel metrikler + Bootstrap %95 GA
# =====
def bootstrap_ci(metric_func, y_true, y_pred, B=1000, seed=42):
    rng = np.random.default_rng(seed)
    vals = []
    n = len(y_true)
    for _ in range(B):
        idx = rng.integers(0, n, size=n)
        vals.append(metric_func(y_true[idx], y_pred[idx]))
    lo, hi = np.percentile(vals, [2.5, 97.5])
    return (float(lo), float(hi))

rmse_all = np.sqrt(mean_squared_error(y_reg, oof_pred))

```

```

mae_all = mean_absolute_error(y_reg, oof_pred)
r2_all = r2_score(y_reg, oof_pred)

rmse_ci = bootstrap_ci(lambda yt, yp: np.sqrt(mean_squared_error(yt, yp)),
y_reg, oof_pred, B=1000)
mae_ci = bootstrap_ci(mean_absolute_error, y_reg, oof_pred, B=1000)
r2_ci = bootstrap_ci(r2_score, y_reg, oof_pred, B=1000)

summary_df = pd.DataFrame([
    "Model": "RandomForest (REG)",
    "RMSE": rmse_all, "RMSE_CI95": f"[{rmse_ci[0]:.3f}, {rmse_ci[1]:.3f}]",
    "MAE": mae_all, "MAE_CI95": f"[{mae_ci[0]:.3f}, {mae_ci[1]:.3f}]",
    "R2": r2_all, "R2_CI95": f"[{r2_ci[0]:.3f}, {r2_ci[1]:.3f}]",
])

# =====
# 4) ÇIKTILAR (ekranda)
# =====

display(HTML("<h4>9E — RandomForest Regressor (Nested CV + HPO) —
Fold Metrikleri</h4>"))
display(fold_df)

display(HTML("<h4>9E — RandomForest Regressor (Nested CV + HPO) —
OOF Özet</h4>"))
display(summary_df)

display(HTML("<h4>9E — RandomForest Regressor — En İyi Parametreler
(Her Fold)</h4>"))
display(best_df)

# =====
# 5) Grafikler: Gerçek–Tahmin scatter, Rezidü histogram

```

```

# =====

plt.figure()
plt.scatter(y_reg, oof_pred, s=18) # renk belirtmiyoruz
mn, mx = np.min([y_reg.min(), oof_pred.min()]), np.max([y_reg.max(),
oof_pred.max()])
plt.plot([mn, mx], [mn, mx], '--')
plt.xlabel("Gerçek AHI_TOPLAM")
plt.ylabel("OOF Tahmin AHI_TOPLAM")
plt.title("RandomForest (REG) — Gerçek vs. OOF Tahmin (45° çizgili)")
plt.show()

resid = y_reg - oof_pred
plt.figure()
plt.hist(resid, bins=30)
plt.xlabel("Rezidü (Gerçek - Tahmin)")
plt.ylabel("Frekans")
plt.title("RandomForest (REG) — OOF Rezidü Histogram")
plt.show()

# =====

# 6) İLERİ ADIMLAR İÇİN KAYDET (RAM + Disk)
# =====

globals()["y_reg"] = y_reg
globals()["oof_reg_rf"] = oof_pred

os.makedirs("oof_preds", exist_ok=True)
np.savez_compressed("oof_preds/reg_rf_oof.npz", oof=oof_pred, y=y_reg)
print("📁 Export: y_reg, oof_reg_rf (RAM) ve oof_preds/reg_rf_oof.npz (disk)")

# 9F) REG — XGBoost Regressor (Nested CV + HPO + OOF + özet + grafikler)

import os, re, warnings; warnings.filterwarnings("ignore")

```

```

import numpy as np, pandas as pd, matplotlib.pyplot as plt
from IPython.display import display, HTML

from sklearn.model_selection import RepeatedKFold, KFold,
RandomizedSearchCV
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder

# --- XGBoost kurulumu/ithali ---
try:
    from xgboost import XGBRegressor
except Exception:
    os.system("pip -q install xgboost")
    from xgboost import XGBRegressor

# =====
# 0) VERİ & HEDEFLENER
# =====
TARGET_REG = "AHI_TOPLAM"
TARGET_CLF = "AHI_SINIF"

# df yoksa CSV oku (sadece okuma)
if 'df' not in globals():
    try:
        from google.colab import files
        print("📁 df yok: CSV seçin...")
        up = files.upload(); assert up, "Dosya seçilmedi."
        fname = next(iter(up.keys()))
        ascii_name = re.sub(r"^[A-Za-z0-9_.-]+", "_", fname)

```

```

if ascii_name != fname: os.rename(fname, ascii_name); fname = ascii_name
used=None
for enc in ["utf-8","utf-8-sig","ISO-8859-9","cp1254","latin-1"]:
    try: df = pd.read_csv(fname, encoding=enc); used=enc; break
    except: pass
assert used, "CSV okunamadı."
print(f"☑ CSV okundu: {fname} | enc={used} | shape={df.shape}")
except Exception as e:
    raise RuntimeError("df yok ve yükleme başarısız.") from e
else:
    print(f"☑ df mevcut: shape={df.shape}")

assert TARGET_REG in df.columns and TARGET_CLF in df.columns,
"AHİ_TOPLAM / AHİ_SINIF kolonları bulunamadı."

X = df.drop(columns=[TARGET_REG, TARGET_CLF]).copy()
y_reg = pd.to_numeric(df[TARGET_REG], errors="coerce").values

# Preprocess: sayısal median impute, kategorik mode+OHE
num_cols = X.select_dtypes(include=["number"]).columns.tolist()
cat_cols = X.select_dtypes(include=["object","category","bool"]).columns.tolist()

preprocess = ColumnTransformer(
    transformers=[
        ("num", SimpleImputer(strategy="median"), num_cols),
        ("cat", Pipeline([
            ("imp", SimpleImputer(strategy="most_frequent")),
            ("ohe", OneHotEncoder(handle_unknown="ignore"))
        ]), cat_cols),
    ],
    remainder="drop"
)

```

```

# =====
# 1) Nested CV + HPO ayarları
# =====

N_SPLITS_OUTER = 5
N_REPEATS_OUTER = 2 # süreye göre 3 yapabilirsin
N_SPLITS_INNER = 3
N_ITER = 25 # HPO örnek sayısı
RANDOM_STATE = 42

outer_cv = RepeatedKfold(n_splits=N_SPLITS_OUTER,
n_repeats=N_REPEATS_OUTER, random_state=RANDOM_STATE)
inner_cv = Kfold(n_splits=N_SPLITS_INNER, shuffle=True,
random_state=RANDOM_STATE)

base = XGBRegressor(
    tree_method="hist", # hızlı ve stabil
    objective="reg:squarederror",
    random_state=RANDOM_STATE,
    n_jobs=-1
)

param_space = {
    "model__n_estimators": [400, 700, 1000],
    "model__max_depth": [3, 5, 7, 9],
    "model__learning_rate": [0.01, 0.03, 0.05, 0.1],
    "model__subsample": [0.7, 0.85, 1.0],
    "model__colsample_bytree": [0.7, 0.85, 1.0],
    "model__min_child_weight": [1, 3, 5],
    "model__gamma": [0, 1, 5],
    "model__reg_alpha": [0, 0.1, 1.0],
    "model__reg_lambda": [0.5, 1.0, 2.0],

```

```
}
```

```
pipe = Pipeline([("prep", preprocess), ("model", base)])
```

```
# Skor: neg-MSE (RMSE ile tutarlı)
```

```
search = RandomizedSearchCV(  
    estimator=pipe,  
    param_distributions=param_space,  
    n_iter=N_ITER,  
    scoring="neg_mean_squared_error",  
    cv=inner_cv,  
    refit=True,  
    random_state=RANDOM_STATE,  
    n_jobs=-1,  
    verbose=0  
)
```

```
# =====
```

```
# 2) Outer döngü: OOF tahmin + fold metrikleri + en iyi paramlar
```

```
# =====
```

```
n = len(y_reg)
```

```
oof_pred = np.zeros(n, dtype=float)
```

```
fold_rows = []
```

```
best_rows = []
```

```
fold_id = 0
```

```
for tr_idx, te_idx in outer_cv.split(X, y_reg):
```

```
    fold_id += 1
```

```
    X_tr, X_te = X.iloc[tr_idx], X.iloc[te_idx]
```

```
    y_tr, y_te = y_reg[tr_idx], y_reg[te_idx]
```

```
# İç arama (HPO)
```

```

search.fit(X_tr, y_tr)
best_est = search.best_estimator_

# Test folduna tahmin
y_hat = best_est.predict(X_te)
oof_pred[te_idx] = y_hat

# Fold metrikleri
rmse = np.sqrt(mean_squared_error(y_te, y_hat))
mae = mean_absolute_error(y_te, y_hat)
r2 = r2_score(y_te, y_hat)

fold_rows.append({"Fold": fold_id, "RMSE": rmse, "MAE": mae, "R2": r2})

# En iyi param özet
best_rows.append({
    "Fold": fold_id,
    **{k.replace("model__", ""): v for k,v in search.best_params_.items()}
})

fold_df = pd.DataFrame(fold_rows)
best_df = pd.DataFrame(best_rows)

# =====
# 3) OOF genel metrikler + Bootstrap %95 GA
# =====
def bootstrap_ci(metric_func, y_true, y_pred, B=1000, seed=42):
    rng = np.random.default_rng(seed)
    vals = []
    n = len(y_true)
    for _ in range(B):
        idx = rng.integers(0, n, size=n)

```

```

        vals.append(metric_func(y_true[idx], y_pred[idx]))
    lo, hi = np.percentile(vals, [2.5, 97.5])
    return (float(lo), float(hi))

rmse_all = np.sqrt(mean_squared_error(y_reg, oof_pred))
mae_all = mean_absolute_error(y_reg, oof_pred)
r2_all = r2_score(y_reg, oof_pred)

rmse_ci = bootstrap_ci(lambda yt, yp: np.sqrt(mean_squared_error(yt, yp)),
                        y_reg, oof_pred, B=1000)
mae_ci = bootstrap_ci(mean_absolute_error, y_reg, oof_pred, B=1000)
r2_ci = bootstrap_ci(r2_score, y_reg, oof_pred, B=1000)

summary_df = pd.DataFrame([
    "Model": "XGBoost (REG)",
    "RMSE": rmse_all, "RMSE_CI95": f"[{rmse_ci[0]:.3f}, {rmse_ci[1]:.3f}]",
    "MAE": mae_all, "MAE_CI95": f"[{mae_ci[0]:.3f}, {mae_ci[1]:.3f}]",
    "R2": r2_all, "R2_CI95": f"[{r2_ci[0]:.3f}, {r2_ci[1]:.3f}]",
])

# =====
# 4) ÇIKTILAR (ekranda)
# =====

display(HTML("<h4>9F — XGBoost Regressor (Nested CV + HPO) — Fold  
Metrikleri</h4>"))
display(fold_df)

display(HTML("<h4>9F — XGBoost Regressor (Nested CV + HPO) — OOF  
Özet</h4>"))
display(summary_df)

```

```

display(HTML("<h4>9F — XGBoost Regressor — En İyi Parametreler (Her
Fold)</h4>"))
display(best_df)

# =====
# 5) Grafikler: Gerçek–Tahmin scatter, Rezidü histogram
# =====

plt.figure()
plt.scatter(y_reg, oof_pred, s=18)
mn, mx = np.min([y_reg.min(), oof_pred.min()]), np.max([y_reg.max(),
oof_pred.max()])
plt.plot([mn, mx], [mn, mx], '--')
plt.xlabel("Gerçek AHI_TOPLAM")
plt.ylabel("OOF Tahmin AHI_TOPLAM")
plt.title("XGBoost (REG) — Gerçek vs. OOF Tahmin (45° çizgili)")
plt.show()

resid = y_reg - oof_pred
plt.figure()
plt.hist(resid, bins=30)
plt.xlabel("Rezidü (Gerçek - Tahmin)")
plt.ylabel("Frekans")
plt.title("XGBoost (REG) — OOF Rezidü Histogram")
plt.show()

# =====
# 6) İLERİ ADIMLAR İÇİN KAYDET (RAM + Disk)
# =====

globals()["y_reg"] = y_reg
globals()["oof_reg_xgb"] = oof_pred

os.makedirs("oof_preds", exist_ok=True)

```

```
np.savez_compressed("oof_preds/reg_xgb_oof.npz", oof=oof_pred, y=y_reg)
print("📁 Export: y_reg, oof_reg_xgb (RAM) ve oof_preds/reg_xgb_oof.npz (disk)")
```

9G) REG — CatBoost Regressor (Nested CV + HPO + OOF + per-fold progress + özet + grafikler)

```
import os, re, warnings; warnings.filterwarnings("ignore")
import numpy as np, pandas as pd, matplotlib.pyplot as plt
from IPython.display import display, HTML

from sklearn.model_selection import RepeatedKFold, KFold,
RandomizedSearchCV
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder

# --- CatBoost kurulumu/ithal ---
try:
    from catboost import CatBoostRegressor
except Exception:
    os.system("pip -q install catboost")
    from catboost import CatBoostRegressor

# =====
# 0) VERİ & HEDEFLENER
# =====
TARGET_REG = "AHI_TOPLAM"
TARGET_CLF = "AHI_SINIF"
```

```

# df yoksa CSV oku (sadece okuma)
if 'df' not in globals():
    try:
        from google.colab import files
        print("📁 df yok: CSV seçin...")
        up = files.upload(); assert up, "Dosya seçilmedi."
        fname = next(iter(up.keys()))
        ascii_name = re.sub(r"^[A-Za-z0-9_.-]+", "_", fname)
        if ascii_name != fname: os.rename(fname, ascii_name); fname = ascii_name
        used=None
        for enc in ["utf-8", "utf-8-sig", "ISO-8859-9", "cp1254", "latin-1"]:
            try: df = pd.read_csv(fname, encoding=enc); used=enc; break
            except: pass
        assert used, "CSV okunamadı."
        print(f"✅ CSV okundu: {fname} | enc={used} | shape={df.shape}")
    except Exception as e:
        raise RuntimeError("df yok ve yükleme başarısız.") from e
else:
    print(f"✅ df mevcut: shape={df.shape}")

assert TARGET_REG in df.columns and TARGET_CLF in df.columns,
"AHI_TOPLAM / AHI_SINIF kolonları bulunamadı."

X = df.drop(columns=[TARGET_REG, TARGET_CLF]).copy()
y_reg = pd.to_numeric(df[TARGET_REG], errors="coerce").values

# Preprocess: sayısal median impute, kategorik mode+OHE
num_cols = X.select_dtypes(include=["number"]).columns.tolist()
cat_cols = X.select_dtypes(include=["object", "category", "bool"]).columns.tolist()

preprocess = ColumnTransformer(
    transformers=[

```

```

        ("num", SimpleImputer(strategy="median"), num_cols),
        ("cat", Pipeline([
            ("imp", SimpleImputer(strategy="most_frequent")),
            ("ohe", OneHotEncoder(handle_unknown="ignore"))
        ]), cat_cols),
    ],
    remainder="drop"
)

# =====
# 1) Nested CV + HPO ayarları
# =====
N_SPLITS_OUTER = 5
N_REPEATS_OUTER = 2 # süreye göre 3 yapabilirsin
N_SPLITS_INNER = 3
N_ITER = 25 # HPO örnek sayısı (gerekirse düşürülebilir)
RANDOM_STATE = 42

outer_cv = RepeatedKfold(n_splits=N_SPLITS_OUTER,
n_repeats=N_REPEATS_OUTER, random_state=RANDOM_STATE)
inner_cv = Kfold(n_splits=N_SPLITS_INNER, shuffle=True,
random_state=RANDOM_STATE)

base = CatBoostRegressor(
    loss_function="RMSE",
    random_seed=RANDOM_STATE,
    thread_count=-1,
    verbose=False
)

param_space = {
    "model__iterations": [400, 700, 1000],

```

```

"model__depth": [4, 6, 8],
"model__learning_rate":[0.01, 0.03, 0.06],
"model__l2_leaf_reg": [1, 3, 5, 7],
"model__border_count": [32, 64, 128]
}

```

```

pipe = Pipeline([("prep", preprocess), ("model", base)])

```

```

# Skor: neg-MSE (RMSE ile tutarlı)

```

```

search = RandomizedSearchCV(
    estimator=pipe,
    param_distributions=param_space,
    n_iter=N_ITER,
    scoring="neg_mean_squared_error",
    cv=inner_cv,
    refit=True,
    random_state=RANDOM_STATE,
    n_jobs=-1,
    verbose=0
)

```

```

# =====

```

```

# 2) Outer döngü: OOF tahmin + fold metrikleri + en iyi paramlar (per-fold
output)

```

```

# =====

```

```

n = len(y_reg)
oof_pred = np.zeros(n, dtype=float)
fold_rows = []
best_rows = []

```

```

fold_id = 0

```

```

print("🔧 Outer CV başlıyor...\n")

```

```

for tr_idx, te_idx in outer_cv.split(X, y_reg):
    fold_id += 1
    X_tr, X_te = X.iloc[tr_idx], X.iloc[te_idx]
    y_tr, y_te = y_reg[tr_idx], y_reg[te_idx]

    # İç arama (HPO)
    search.fit(X_tr, y_tr)
    best_est = search.best_estimator_

    # Test folduna tahmin
    y_hat = best_est.predict(X_te)
    oof_pred[te_idx] = y_hat

    # Fold metrikleri
    rmse = np.sqrt(mean_squared_error(y_te, y_hat))
    mae = mean_absolute_error(y_te, y_hat)
    r2 = r2_score(y_te, y_hat)

    fold_rows.append({"Fold": fold_id, "RMSE": rmse, "MAE": mae, "R2": r2})

    # En iyi paramlar (özet)
    bp = {k.replace("model__", ""): v for k, v in search.best_params_.items()}
    best_rows.append({"Fold": fold_id, **bp})

    # ---- Per-fold canlı çıktı ----
    print(f'Fold {fold_id:02d} bitti → RMSE={rmse:.4f} | MAE={mae:.4f} | R2={r2:.4f}')
    print(f' Best params: {bp}\n')

fold_df = pd.DataFrame(fold_rows)
best_df = pd.DataFrame(best_rows)

```

```

# =====
# 3) OOF genel metrikler + Bootstrap %95 GA
# =====

def bootstrap_ci(metric_func, y_true, y_pred, B=1000, seed=42):
    rng = np.random.default_rng(seed)
    vals = []
    n = len(y_true)
    for _ in range(B):
        idx = rng.integers(0, n, size=n)
        vals.append(metric_func(y_true[idx], y_pred[idx]))
    lo, hi = np.percentile(vals, [2.5, 97.5])
    return (float(lo), float(hi))

rmse_all = np.sqrt(mean_squared_error(y_reg, oof_pred))
mae_all = mean_absolute_error(y_reg, oof_pred)
r2_all = r2_score(y_reg, oof_pred)

rmse_ci = bootstrap_ci(lambda yt, yp: np.sqrt(mean_squared_error(yt, yp)),
y_reg, oof_pred, B=1000)
mae_ci = bootstrap_ci(mean_absolute_error, y_reg, oof_pred, B=1000)
r2_ci = bootstrap_ci(r2_score, y_reg, oof_pred, B=1000)

summary_df = pd.DataFrame([
    "Model": "CatBoost (REG)",
    "RMSE": rmse_all, "RMSE_CI95": f"[{rmse_ci[0]:.3f}, {rmse_ci[1]:.3f}]",
    "MAE": mae_all, "MAE_CI95": f"[{mae_ci[0]:.3f}, {mae_ci[1]:.3f}]",
    "R2": r2_all, "R2_CI95": f"[{r2_ci[0]:.3f}, {r2_ci[1]:.3f}]",
])

# =====
# 4) ÇIKTILAR (ekranda)
# =====

```

```
display(HTML("<h4>9G — CatBoost Regressor (Nested CV + HPO) — Fold  
Metrikleri</h4>"))
```

```
display(fold_df)
```

```
display(HTML("<h4>9G — CatBoost Regressor (Nested CV + HPO) — OOF  
Özet</h4>"))
```

```
display(summary_df)
```

```
display(HTML("<h4>9G — CatBoost Regressor — En İyi Parametreler (Her  
Fold)</h4>"))
```

```
display(best_df)
```

```
# =====
```

```
# 5) Grafikler: Gerçek–Tahmin scatter, Rezidü histogram
```

```
# =====
```

```
plt.figure()
```

```
plt.scatter(y_reg, oof_pred, s=18)
```

```
mn, mx = np.min([y_reg.min(), oof_pred.min()]), np.max([y_reg.max(),  
oof_pred.max()])
```

```
plt.plot([mn, mx], [mn, mx], '--')
```

```
plt.xlabel("Gerçek AHI_TOPLAM")
```

```
plt.ylabel("OOF Tahmin AHI_TOPLAM")
```

```
plt.title("CatBoost (REG) — Gerçek vs. OOF Tahmin (45° çizgili)")
```

```
plt.show()
```

```
resid = y_reg - oof_pred
```

```
plt.figure()
```

```
plt.hist(resid, bins=30)
```

```
plt.xlabel("Rezidü (Gerçek - Tahmin)")
```

```
plt.ylabel("Frekans")
```

```
plt.title("CatBoost (REG) — OOF Rezidü Histogram")
```

```
plt.show()
```

```

# =====
# 6) İLERİ ADIMLAR İÇİN KAYDET (RAM + Disk)
# =====

globals()["y_reg"] = y_reg
globals()["oof_reg_cat"] = oof_pred

os.makedirs("oof_preds", exist_ok=True)
np.savez_compressed("oof_preds/reg_cat_oof.npz", oof=oof_pred, y=y_reg)
print("📁 Export: y_reg, oof_reg_cat (RAM) ve oof_preds/reg_cat_oof.npz (disk)")

# 9H) Regresyon — Toplu Özet ve Kıyas Tablosu
import os, numpy as np, pandas as pd, matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from IPython.display import display, HTML

# --- Yardımcı: .npz dosyasını oku ---
def load_oof_npz(path, key):
    if os.path.exists(path):
        d = np.load(path)
        return d["y"], d["oof"]
    return None, None

# --- RF OOF ---
y_rf, oof_rf = load_oof_npz("oof_preds/reg_rf_oof.npz", "oof")
if y_rf is not None:
    rmse_rf = np.sqrt(mean_squared_error(y_rf, oof_rf))
    mae_rf = mean_absolute_error(y_rf, oof_rf)
    r2_rf = r2_score(y_rf, oof_rf)
else:
    rmse_rf = mae_rf = r2_rf = np.nan

```

```

# --- XGB OOF ---
y_xgb, oof_xgb = load_oof_npz("oof_preds/reg_xgb_oof.npz", "oof")
if y_xgb is not None:
    rmse_xgb = np.sqrt(mean_squared_error(y_xgb, oof_xgb))
    mae_xgb = mean_absolute_error(y_xgb, oof_xgb)
    r2_xgb = r2_score(y_xgb, oof_xgb)
else:
    rmse_xgb = mae_xgb = r2_xgb = np.nan

# --- Cat OOF ---
y_cat, oof_cat = load_oof_npz("oof_preds/reg_cat_oof.npz", "oof")
if y_cat is not None:
    rmse_cat = np.sqrt(mean_squared_error(y_cat, oof_cat))
    mae_cat = mean_absolute_error(y_cat, oof_cat)
    r2_cat = r2_score(y_cat, oof_cat)
else:
    rmse_cat = mae_cat = r2_cat = np.nan

# --- Tablo ---
summary = pd.DataFrame([
    {"Model": "RandomForest (REG)", "RMSE": rmse_rf, "MAE": mae_rf, "R2":
r2_rf},
    {"Model": "XGBoost (REG)", "RMSE": rmse_xgb, "MAE": mae_xgb,
"R2": r2_xgb},
    {"Model": "CatBoost (REG)", "RMSE": rmse_cat, "MAE": mae_cat, "R2":
r2_cat},
])

display(HTML("<h4>9H — Regresyon Modelleri: OOF Özet Tablosu</h4>"))
display(summary.sort_values("RMSE"))

# --- Grafik: RMSE, MAE, R² ---

```

```

fig, axes = plt.subplots(1, 3, figsize=(15,4))
summary.plot(x="Model", y="RMSE", kind="bar", ax=axes[0], legend=False,
title="RMSE (↓ daha iyi)")
summary.plot(x="Model", y="MAE", kind="bar", ax=axes[1], legend=False,
title="MAE (↓ daha iyi)")
summary.plot(x="Model", y="R2", kind="bar", ax=axes[2], legend=False,
title="R² (↑ daha iyi)")
plt.tight_layout()
plt.show()

# === ADIM 10 — CLF: OOF Kurtarma/Üretim + 9D Özet + Tüm Kıyas Testleri
===

import os, re, warnings; warnings.filterwarnings("ignore")
import numpy as np, pandas as pd, matplotlib.pyplot as plt
from IPython.display import display, HTML

# sklearn
from sklearn.model_selection import StratifiedKFold, cross_val_predict
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (roc_auc_score, average_precision_score, f1_score,
recall_score,
                             brier_score_loss)
from sklearn.linear_model import LogisticRegression

# XGBoost / CatBoost (gerekirse yükle)
try:
    from xgboost import XGBClassifier
except Exception:
    os.system("pip -q install xgboost"); from xgboost import XGBClassifier

```

```

try:
    from catboost import CatBoostClassifier
except Exception:
    os.system("pip -q install catboost"); from catboost import CatBoostClassifier

# =====
# 0) VERİ & HEDEFLENER
# =====

TARGET_CLF = "AHI_SINIF"
TARGET_REG = "AHI_TOPLAM" # sadece X'ten düşmek için

def _load_df_if_needed():
    if 'df' in globals():
        return globals()['df']
    try:
        from google.colab import files
        print("📁 df yok: CSV seçin...")
        up = files.upload(); assert up, "Dosya seçilmedi."
        fname = next(iter(up.keys()))
        ascii_name = re.sub(r"[^A-Za-z0-9_.-]+", "_", fname)
        if ascii_name != fname: os.rename(fname, ascii_name); fname = ascii_name
        for enc in ["utf-8", "utf-8-sig", "ISO-8859-9", "cp1254", "latin-1"]:
            try:
                _df = pd.read_csv(fname, encoding=enc)
                print(f"✅ CSV okundu: {fname} | enc={enc} | shape={_df.shape}")
                return _df
            except: pass
        raise RuntimeError("CSV okunamadı.")
    except Exception as e:
        raise RuntimeError("df yok ve yükleme başarısız.") from e

df = _load_df_if_needed()

```

```

assert TARGET_CLF in df.columns, "AHI_SINIF kolonunu bulamadım."
assert TARGET_REG in df.columns, "AHI_TOPLAM kolonunu bulamadım."

X = df.drop(columns=[TARGET_CLF, TARGET_REG]).copy()
y_clf = pd.to_numeric(df[TARGET_CLF], errors="coerce").astype(int).values
globals()['y_clf'] = y_clf

# =====
# 1) PREPROCESS (sabit)
# =====

num_cols = X.select_dtypes(include=["number"]).columns.tolist()
cat_cols = X.select_dtypes(include=["object", "category", "bool"]).columns.tolist()

preprocess = ColumnTransformer(
    transformers=[
        ("num", SimpleImputer(strategy="median"), num_cols),
        ("cat", Pipeline([("imp", SimpleImputer(strategy="most_frequent")),
                          ("ohe", OneHotEncoder(handle_unknown="ignore"))]), cat_cols),
    ],
    remainder="drop"
)

cv_fast = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# =====
# 2) DISK OOF YÜKLE/KAYDET
# =====

OOF_DIR = "oof_preds"
os.makedirs(OOF_DIR, exist_ok=True)

def oof_save(key, proba, y):
    np.savez_compressed(os.path.join(OOF_DIR, f"{key}_oof.npz"),

```

```

        oof=np.asarray(proba).ravel(), y=np.asarray(y).ravel())
print(f"📁 OOF kaydedildi: {OOF_DIR}/{key}_oof.npz (len={len(proba)})")

def oof_load_any(keys):
    for k in keys:
        f = os.path.join(OOF_DIR, f"{k}_oof.npz")
        if os.path.exists(f):
            d = np.load(f)
            return d["oof"], d["y"], f
    return None, None, None

# =====
# 3) CAT OOF VAR MI? — RAM/DISK
# =====
oof_cat = globals().get("oof_cat", None)
src_cat = None
if oof_cat is not None and len(oof_cat)==len(y_clf):
    src_cat = "RAM:oof_cat"
else:
    oofc, yc, path = oof_load_any(["cat", "catboost"])
    if oofc is not None and len(oofc)==len(y_clf):
        oof_cat = oofc; src_cat = f"DISK:{os.path.basename(path)}"
        globals()["oof_cat"] = oof_cat

print(f"👁 CatBoost OOF durumu: {'VAR → '+src_cat if oof_cat is not None else
'YOK (hızlı üretilecek)'}")

# =====
# 4) RF & XGB OOF HIZLI HESAP (ve CatBoost yoksa onu da)
# =====
def make_oof_fast(clf):
    pipe = Pipeline([("prep", preprocess), ("model", clf)])

```

```

    p = cross_val_predict(pipe, X, y_clf, cv=cv_fast, method="predict_proba",
n_jobs=-1)[: ,1]

    return p

# RandomForest (hızlı)
rf = RandomForestClassifier(n_estimators=400, max_depth=None,
                           min_samples_split=5, min_samples_leaf=2,
                           class_weight="balanced", n_jobs=-1, random_state=42)

print("⚙️ RF OOF üretiliyor...")

oof_rf = make_oof_fast(rf); globals()["oof_rf"] = oof_rf; oof_save("rf", oof_rf,
y_clf)

print("✓ RF OOF tamam.")

# XGBoost (hızlı)
pos = int((y_clf==1).sum()); neg = int((y_clf==0).sum()); spw = max(1.0,
neg/max(1,pos))

xgb = XGBClassifier(
    n_estimators=700, max_depth=6, learning_rate=0.06,
    subsample=0.9, colsample_bytree=0.8, min_child_weight=3,
    gamma=0, scale_pos_weight=spw,
    eval_metric="logloss", tree_method="hist", n_jobs=-1, random_state=42
)

print("⚙️ XGB OOF üretiliyor...")

oof_xgb = make_oof_fast(xgb); globals()["oof_xgb"] = oof_xgb; oof_save("xgb",
oof_xgb, y_clf)

print("✓ XGB OOF tamam.")

# CatBoost (yoksa hızlı)
if oof_cat is None:
    cat = CatBoostClassifier(
        iterations=400, depth=6, learning_rate=0.06,
        l2_leaf_reg=5, border_count=128,

```

```

        loss_function="Logloss", verbose=0, random_seed=42
    )
    print("🌀 CatBoost OOF üretiliyor (yoktu)...")
    oof_cat = make_oof_fast(cat); globals()["oof_cat"] = oof_cat; oof_save("cat",
oof_cat, y_clf)
    print("✓ CatBoost OOF tamam (hızlı).")

# Güvenlik: clip
eps = 1e-6
oof_rf = np.clip(oof_rf, eps, 1-eps)
oof_xgb = np.clip(oof_xgb, eps, 1-eps)
oof_cat = np.clip(oof_cat, eps, 1-eps)

# =====
# 5) 9D — TOPLU ÖZET + GRAFİKLER
# =====
rows=[]
def add_row(name, p):
    rows.append({
        "Model": name,
        "ROC_AUC": roc_auc_score(y_clf, p),
        "PR_AUC": average_precision_score(y_clf, p),
        "F1@0.5": f1_score(y_clf, (p>=0.5).astype(int)),
        "Recall@0.5": recall_score(y_clf, (p>=0.5).astype(int)),
        "Brier": brier_score_loss(y_clf, p)
    })
add_row("RandomForest", oof_rf)
add_row("XGBoost", oof_xgb)
add_row("CatBoost", oof_cat)

df_clf_all = pd.DataFrame(rows).sort_values("ROC_AUC",
ascending=False).reset_index(drop=True)

```

```
display(HTML("<h4>9D — Sınıflandırma Toplu Özet (OOF)</h4>"));
display(df_clf_all)
```

```
def _bar(col, title, ylim=None, ylabel=None):
```

```
    plt.figure()
    plt.bar(df_clf_all["Model"], df_clf_all[col])
    if ylabel: plt.ylabel(ylabel)
    if ylim: plt.ylim(*ylim)
    plt.title(title)
    for i,v in enumerate(df_clf_all[col]):
        plt.text(i, v, f"{v:.3f}", ha="center", va="bottom")
    plt.show()
```

```
_bar("ROC_AUC", "OOF ROC-AUC", (0,1), "ROC-AUC")
```

```
_bar("PR_AUC", "OOF PR-AUC", (0,1), "PR-AUC")
```

```
_bar("Brier", "OOF Brier", None, "Brier (↓ daha iyi)")
```

```
# =====
```

```
# 6) Kıyas Testleri — DeLong, McNemar, Brier Wilcoxon, PR-AUC bootstrap,
Kalibrasyon
```

```
# =====
```

```
from scipy import stats
```

```
from itertools import combinations
```

```
# -- DeLong yardımcıları --
```

```
def _compute_midrank(x):
```

```
    J = np.argsort(x); Z = x[J]; n = len(x); T = np.zeros(n, dtype=float); i = 0
```

```
    while i < n:
```

```
        j = i
```

```
        while j < n and Z[j] == Z[i]: j += 1
```

```
        T[i:j] = 0.5*(i + j - 1) + 1; i = j
```

```
    T2 = np.empty(n, dtype=float); T2[J] = T; return T2
```

```

def _fast_delong(predictions_sorted_transposed, label_1_count):
    m = label_1_count; n = predictions_sorted_transposed.shape[1] - m
    pos = predictions_sorted_transposed[:, :m]; neg =
predictions_sorted_transposed[:, m:]; k = predictions_sorted_transposed.shape[0]
    tx = np.empty((k,m)); ty = np.empty((k,n))
    for r in range(k): tx[r,:] = _compute_midrank(pos[r,:]); ty[r,:] =
_compute_midrank(neg[r,:])
    tz = np.empty((k,m+n))
    for r in range(k): tz[r,:] =
_compute_midrank(predictions_sorted_transposed[r,:])
    aucs = tz[:, :m].sum(axis=1) / (m*n) - (m+1.0)/(2*n)
    v01 = (tz[:, :m] - tx) / n; v10 = 1.0 - (tz[:, m:] - ty) / m
    sx = np.cov(v01); sy = np.cov(v10); s = sx / m + sy / n
    return aucs, s

```

```

def delong_roc_test(y_true, p1, p2):
    y_true = np.asarray(y_true).astype(int)
    order = np.argsort(-y_true) # pozitifler önce
    y_sorted = y_true[order]
    p_mat = np.vstack([p1, p2])[:, order]
    aucs, cov = _fast_delong(p_mat, int(y_sorted.sum()))
    delta = aucs[0] - aucs[1]
    var = cov[0,0] + cov[1,1] - 2*cov[0,1]
    z = delta / np.sqrt(var + 1e-12)
    p = 2*stats.norm.sf(abs(z))
    return dict(auc1=aucs[0], auc2=aucs[1], delta=delta, z=z, p=p)

```

```

def holm_bonferroni(pairs):
    dfp = pd.DataFrame(pairs,
columns=["Pair", "p"]).sort_values("p").reset_index(drop=True)

```

```

    m = len(dfp); dfp["p_holm"] = [min(1.0, (m-i)*pv) for i,pv in
enumerate(dfp["p"])]
    return dfp

labels = ["RandomForest", "XGBoost", "CatBoost"]
preds = {"RandomForest":oof_rf, "XGBoost":oof_xgb, "CatBoost":oof_cat}

# --- 10A: DeLong ---
del_rows = []
for a,b in combinations(labels, 2):
    r = delong_roc_test(y_clf, preds[a], preds[b])
    del_rows.append((f"{a} vs {b}", r["delta"], r["auc1"], r["auc2"], r["z"], r["p"]))
df_delong = pd.DataFrame(del_rows, columns=["Pair", "ΔAUC (A-
B)", "AUC_A", "AUC_B", "z", "p"])
adj = holm_bonferroni([(r[0], r[5]) for r in del_rows])[["Pair", "p_holm"]]
df_delong = df_delong.merge(adj, on="Pair", how="left")
display(HTML("<h4>10A — DeLong (AUC farkı)</h4>"));
display(df_delong.sort_values("p"))

# --- 10B: McNemar @0.5 ---
try:
    from statsmodels.stats.contingency_tables import mcnemar as sm_mcnemar
except Exception:
    import sys, subprocess; subprocess.run([sys.executable, "-m", "pip", "-
q", "install", "statsmodels"], check=False)
    from statsmodels.stats.contingency_tables import mcnemar as sm_mcnemar

def mcnemar_test(y, p1, p2, thr=0.5):
    yhat1 = (p1>=thr).astype(int); yhat2 = (p2>=thr).astype(int)
    correct1 = (yhat1==y); correct2 = (yhat2==y)
    b = int(np.sum((correct1==True) & (correct2==False)))
    c = int(np.sum((correct1==False) & (correct2==True)))

```

```

table = np.array([[0, b],[c, 0]])
res = sm_mcnemar(table, exact=False, correction=True)
return dict(b=b, c=c, p=float(res.pvalue))

mcn_rows = []
for a,b in combinations(labels, 2):
    r = mcnemar_test(y_clf, preds[a], preds[b], thr=0.5)
    mcn_rows.append((f"{a} vs {b}", r["b"], r["c"], r["p"]))
df_mcn = pd.DataFrame(mcn_rows, columns=["Pair", "b (A doğru,B yanlış)", "c
(A yanlış,B doğru)", "p"])
adj = holm_bonferroni([(row[0], row[3]) for row in
mcn_rows])[["Pair", "p_holm"]]
df_mcn = df_mcn.merge(adj, on="Pair", how="left")
display(HTML("<h4>10B — McNemar @0.5</h4>"));
display(df_mcn.sort_values("p"))

# --- 10C: Brier farkı (proxy 5-fold Wilcoxon) ---
from scipy import stats as scs
def brier_wilcoxon(y, p1, p2, n_splits=5, seed=42):
    skf = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=seed)
    d1, d2 = [], []
    for _, te in skf.split(np.zeros_like(y), y):
        d1.append(brier_score_loss(y[te], p1[te]))
        d2.append(brier_score_loss(y[te], p2[te]))
    stat, p = scs.wilcoxon(np.array(d1), np.array(d2), alternative="two-sided",
zero_method="wilcox")
    return dict(brier1=np.mean(d1), brier2=np.mean(d2),
diff=np.mean(np.array(d1)-np.array(d2)), p=p)

brier_rows = []
for a,b in combinations(labels, 2):
    r = brier_wilcoxon(y_clf, preds[a], preds[b], n_splits=5, seed=42)

```

```

brier_rows.append((f"{a} vs {b}", r["brier1"], r["brier2"], r["diff"], r["p"]))
df_brier = pd.DataFrame(brier_rows,
columns=["Pair", "Brier_A", "Brier_B", "ΔBrier (A-B)", "p"])
adj = holm_bonferroni([(row[0], row[4]) for row in
brier_rows])[["Pair", "p_holm"]]
df_brier = df_brier.merge(adj, on="Pair", how="left")
display(HTML("<h4>10C — Brier farkı (Wilcoxon, 5-fold proxy)</h4>"));
display(df_brier.sort_values("p"))

```

```

# --- 10D: PR-AUC farkı (bootstrap) ---
def bootstrap_diff_ap(y, p1, p2, B=1200, seed=42):
    rng = np.random.default_rng(seed)
    idx_pos = np.where(y==1)[0]; idx_neg = np.where(y==0)[0]
    diffs = []
    for _ in range(B):
        s_pos = rng.choice(idx_pos, size=len(idx_pos), replace=True)
        s_neg = rng.choice(idx_neg, size=len(idx_neg), replace=True)
        s = np.concatenate([s_pos, s_neg])
        ap1 = average_precision_score(y[s], p1[s])
        ap2 = average_precision_score(y[s], p2[s])
        diffs.append(ap1 - ap2)
    diffs = np.array(diffs)
    lo, hi = np.percentile(diffs, [2.5, 97.5])
    p = 2*min((diffs<=0).mean(), (diffs>=0).mean())
    return dict(delta=float(average_precision_score(y,p1) -
average_precision_score(y,p2)),
                ci95=(float(lo), float(hi)), p=float(p))

```

```

prauc_rows = []
for a,b in combinations(labels, 2):
    r = bootstrap_diff_ap(y_clf, preds[a], preds[b], B=1200, seed=42)

```

```

    prauc_rows.append((f"{a} vs {b}", r["delta"], f"{r['ci95'][0]:.3f},
{r['ci95'][1]:.3f}", r["p"]))
df_pr = pd.DataFrame(prauc_rows, columns=["Pair", "ΔPR-AUC (A-
B)", "CI95", "p"])
adj = holm_bonferroni([(row[0], row[3]) for row in
prauc_rows])[["Pair", "p_holm"]]
df_pr = df_pr.merge(adj, on="Pair", how="left")
display(HTML("<h4>10D — PR-AUC farkı (bootstrap)</h4>"));
display(df_pr.sort_values("p"))

# --- 10E–10F: Kalibrasyon (slope & intercept, bootstrap) + farklar ---
def logit(x, eps=1e-6):
    x = np.clip(x, eps, 1-eps)
    return np.log(x/(1-x))

def calib_params(y, p):
    X = logit(p).reshape(-1,1)
    # NOT: bazı sürümlerde 'none' desteklenmiyor → l2 + büyük C ile cezasız gibi
    lr = LogisticRegression(penalty='l2', C=1e6, solver='lbfgs', max_iter=2000)
    lr.fit(X, y)
    return float(lr.intercept_[0]), float(lr.coef_[0,0])

def calib_bootstrap(y, p, B=800, seed=42):
    rng = np.random.default_rng(seed)
    inters, slopes = [], []
    n = len(y)
    for _ in range(B):
        s = rng.integers(0, n, size=n)
        i, s2 = calib_params(y[s], p[s]); inters.append(i); slopes.append(s2)
    inters = np.array(inters); slopes = np.array(slopes)
    lo_i, hi_i = np.percentile(inters,[2.5,97.5]); lo_s, hi_s =
np.percentile(slopes,[2.5,97.5])

```

```

return dict(intercept=np.mean(inters), slope=np.mean(slopes),
            intercept_ci=(float(lo_i),float(hi_i)), slope_ci=(float(lo_s),float(hi_s)))

def calib_pair_diff(y, p1, p2, B=800, seed=123):
    rng = np.random.default_rng(seed); n = len(y)
    d_int, d_slope = [], []
    for _ in range(B):
        s = rng.integers(0, n, size=n)
        i1, s1 = calib_params(y[s], p1[s]); i2, s2 = calib_params(y[s], p2[s])
        d_int.append(i1-i2); d_slope.append(s1-s2)
    d_int = np.array(d_int); d_slope = np.array(d_slope)
    lo_i, hi_i = np.percentile(d_int,[2.5,97.5]); lo_s, hi_s =
np.percentile(d_slope,[2.5,97.5])
    p_i = 2*min((d_int<=0).mean(), (d_int>=0).mean()); p_s =
2*min((d_slope<=0).mean(), (d_slope>=0).mean())
    return dict(delta_intercept=float(np.mean(d_int)),
delta_slope=float(np.mean(d_slope)),
                ci95_intercept=(float(lo_i),float(hi_i)),
ci95_slope=(float(lo_s),float(hi_s)),
                p_intercept=float(p_i), p_slope=float(p_s))

# Kalibrasyon tekli
cal_rows = []
for a in labels:
    c = calib_bootstrap(y_clf, preds[a], B=800, seed=42)
    cal_rows.append((a, c["intercept"], f"[{c['intercept_ci'][0]:.3f},
{c['intercept_ci'][1]:.3f}]",
                    c["slope"], f"[{c['slope_ci'][0]:.3f}, {c['slope_ci'][1]:.3f}]"))
df_cal = pd.DataFrame(cal_rows,
columns=["Model", "Intercept_mean", "Intercept_CI95", "Slope_mean", "Slope_CI95"])

```

```
display(HTML("<h4>10E — Kalibrasyon (bootstrap; intercept & slope)</h4>"));
display(df_cal)
```

```
# Kalibrasyon farkları (Holm iki ayrı p için)
caldiff_rows = []
for a,b in combinations(labels, 2):
    cd = calib_pair_diff(y_clf, preds[a], preds[b], B=800, seed=123)
    caldiff_rows.append((f"{a} vs {b}",
                        cd["delta_intercept"], f"[{cd['ci95_intercept']}[0]:.3f},
{cd['ci95_intercept']}[1]:.3f})", cd["p_intercept"],
                        cd["delta_slope"], f"[{cd['ci95_slope']}[0]:.3f},
{cd['ci95_slope']}[1]:.3f})", cd["p_slope"]))
df_caldiff = pd.DataFrame(caldiff_rows, columns=[
    "Pair", "ΔIntercept (A-B)", "ΔIntercept_CI95", "p_intercept", "ΔSlope (A-
B)", "ΔSlope_CI95", "p_slope"
])
def _holm(pairs):
    return holm_bonferroni(pairs)[["Pair", "p_holm"]]
df_caldiff = df_caldiff.merge(_holm([(r[0], r[3]) for r in
caldiff_rows])).rename(columns={"p_holm": "p_holm_intercept"}, on="Pair",
how="left")
df_caldiff = df_caldiff.merge(_holm([(r[0], r[6]) for r in
caldiff_rows])).rename(columns={"p_holm": "p_holm_slope"}, on="Pair",
how="left")
display(HTML("<h4>10F — Kalibrasyon farkları (bootstrap)</h4>"));
display(df_caldiff.sort_values("p_slope"))
```

```
print("\n☑ Sınıflandırma: kurtarma + 9D özet + Adım 10 kıyas tamam.")
# === ADIM 11 — REG: OOF yükle + Özet + Pairwise kıyas (Wilcoxon +
Holm) + Bootstrap GA ===
import os, numpy as np, pandas as pd, matplotlib.pyplot as plt, warnings
warnings.filterwarnings("ignore")
```

```

from IPython.display import display, HTML

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import KFold
from scipy import stats

OOF_DIR = "oof_preds"

def load_reg_oof_any():
    """
    OOF'ları RAM'den veya diskten yükler.
    Kabul edilen disk dosyaları:
    - oof_preds/reg_rf_oof.npz
    - oof_preds/reg_xgb_oof.npz
    - oof_preds/reg_cat_oof.npz
    RAM değişkenleri:
    - y_reg, oof_reg_rf, oof_reg_xgb, oof_reg_cat
    """
    found = {}
    yref = None

    # RAM
    if "y_reg" in globals():
        yref = np.asarray(globals()["y_reg"]).ravel()
    for key, var in [("RandomForest", "oof_reg_rf"), ("XGBoost", "oof_reg_xgb"),
("CatBoost", "oof_reg_cat")]:
        arr = globals().get(var, None)
        if arr is not None:
            arr = np.asarray(arr).ravel()
            found[key] = arr

    # Disk (gerekirse)

```

```

def _load_npz(path):
    if os.path.exists(path):
        d = np.load(path)
        return d["y"], d["oof"]
    return None, None

if yref is None:
    for f in ["reg_rf_oof.npz", "reg_xgb_oof.npz", "reg_cat_oof.npz"]:
        ytmp, p = _load_npz(os.path.join(OOF_DIR, f))
        if ytmp is not None:
            yref = ytmp
            break

    for key, fname in [("RandomForest", "reg_rf_oof.npz"),
                       ("XGBoost", "reg_xgb_oof.npz"),
                       ("CatBoost", "reg_cat_oof.npz")]:
        if key not in found:
            ytmp, p = _load_npz(os.path.join(OOF_DIR, fname))
            if p is not None:
                if yref is None: yref = ytmp
                found[key] = p

    if yref is None or len(found)==0:
        raise AssertionError("OOF bulunamadı. Lütfen 9E, 9F, 9G adımlarını (REG)
çalıştırıp OOF kaydedin.")

# boyut kontrolü
for k,v in list(found.items()):
    if len(v)!=len(yref):
        print(f"⚠ {k} uzunluğu y ile uyuşmuyor; çıkarıldı.")
        del found[k]

```

```

assert len(found)>=2, "Kıyas için en az 2 modelin OOF'u gerekli."
return yref.astype(float), found

y, preds = load_reg_oof_any()
labels = list(preds.keys())
print("🔍 Bulunan REG modelleri:", labels)

# Güvenlik: sayılar
for k in preds:
    preds[k] = np.asarray(preds[k]).astype(float).ravel()

# ----- Yardımcılar -----
def rmse(y, p): return float(np.sqrt(mean_squared_error(y, p)))
def mae(y, p): return float(mean_absolute_error(y, p))
def r2 (y, p): return float(r2_score(y, p))

def paired_bootstrap_metric(y, p, B=2000, seed=42, metric=rmse):
    rng = np.random.default_rng(seed)
    n = len(y); vals=[]
    for _ in range(B):
        idx = rng.integers(0, n, size=n) # aynı idx hem y hem p için
        vals.append(metric(y[idx], p[idx]))
    lo, hi = np.percentile(vals, [2.5, 97.5])
    return dict(mean=float(np.mean(vals)), ci=(float(lo), float(hi)))

def paired_bootstrap_delta(y, p1, p2, B=3000, seed=123, metric=rmse):
    rng = np.random.default_rng(seed)
    n = len(y); diffs=[]
    for _ in range(B):
        idx = rng.integers(0, n, size=n)
        diffs.append(metric(y[idx], p1[idx]) - metric(y[idx], p2[idx]))
    diffs = np.array(diffs)

```

```

lo, hi = np.percentile(diffs, [2.5, 97.5])
p = 2*min((diffs<=0).mean(), (diffs>=0).mean())
return dict(delta=float(metric(y,p1)-metric(y,p2)),
            mean=float(np.mean(diffs)), ci=(float(lo), float(hi)), p=float(p))

def holm_bonferroni(pairs): # pairs: [(label, p), ...]
    dfp = pd.DataFrame(pairs,
        columns=["Pair", "p"]).sort_values("p").reset_index(drop=True)
    m = len(dfp); dfp["p_holm"] = [min(1.0, (m-i)*pv) for i,pv in
        enumerate(dfp["p"])]
    return dfp

# ----- 11A: Özet tablo + bootstrap GA -----
rows=[]
boot_rows=[]
for m in labels:
    p = preds[m]
    rows.append({"Model":m, "RMSE":rmse(y,p), "MAE":mae(y,p),
        "R2":r2(y,p)})
    b_rmse = paired_bootstrap_metric(y, p, B=1500, seed=42, metric=rmse)
    b_mae = paired_bootstrap_metric(y, p, B=1500, seed=43, metric=mae)
    b_r2 = paired_bootstrap_metric(y, p, B=1500, seed=44, metric=r2)
    boot_rows.append({"Model":m,
        "RMSE_CI95":f"[{b_rmse['ci'][0]:.3f}, {b_rmse['ci'][1]:.3f}]",
        "MAE_CI95": f"[{b_mae['ci'][0]:.3f}, {b_mae['ci'][1]:.3f}]",
        "R2_CI95": f"[{b_r2['ci'][0]:.3f}, {b_r2['ci'][1]:.3f}]"
    })

summary = pd.DataFrame(rows)
ci_tab = pd.DataFrame(boot_rows)
summary_full = summary.merge(ci_tab, on="Model")

```

```
display(HTML("<h4>11A — Regresyon modelleri: OOF özet (RMSE, MAE, R²)
+ bootstrap %95 GA</h4>"))
display(summary_full.sort_values("RMSE"))
```

```
# ----- 11B: Pairwise Wilcoxon (RMSE & MAE) -----
# Aynı fold dilimlerinde hesaplanmış metrik dizileri üzerinden paired Wilcoxon
def fold_metrics(y, p, n_splits=5, seed=42, metric=rmse):
    kf = KFold(n_splits=n_splits, shuffle=True, random_state=seed)
    vals=[]
    for _, te in kf.split(np.arange(len(y))):
        vals.append(metric(y[te], p[te]))
    return np.array(vals, float)

from itertools import combinations
wilc_rows_rmse = []
wilc_rows_mae = []
for a,b in combinations(labels, 2):
    # RMSE
    a_rmse = fold_metrics(y, preds[a], n_splits=5, seed=42, metric=rmse)
    b_rmse = fold_metrics(y, preds[b], n_splits=5, seed=42, metric=rmse)
    stat, p = stats.wilcoxon(a_rmse, b_rmse, alternative="two-sided",
zero_method="wilcox")
    wilc_rows_rmse.append((f"{a} vs {b}", float(np.mean(a_rmse)),
float(np.mean(b_rmse)),
float(np.mean(a_rmse - b_rmse)), float(p)))

    # MAE
    a_mae = fold_metrics(y, preds[a], n_splits=5, seed=42, metric=mae)
    b_mae = fold_metrics(y, preds[b], n_splits=5, seed=42, metric=mae)
    stat2, p2 = stats.wilcoxon(a_mae, b_mae, alternative="two-sided",
zero_method="wilcox")
    wilc_rows_mae.append((f"{a} vs {b}", float(np.mean(a_mae)),
float(np.mean(b_mae)),
```

```

float(np.mean(a_mae - b_mae)), float(p2)))

df_w_rmse = pd.DataFrame(wilc_rows_rmse,
columns=["Pair", "RMSE_A", "RMSE_B", "ΔRMSE (A-B)", "p"])
df_w_mae = pd.DataFrame(wilc_rows_mae,
columns=["Pair", "MAE_A", "MAE_B", "ΔMAE (A-B)", "p"])

# Holm düzeltme
adj_rmse = holm_bonferroni([(r[0], r[4]) for r in
wilc_rows_rmse])[["Pair", "p_holm"]]
adj_mae = holm_bonferroni([(r[0], r[4]) for r in wilc_rows_mae
])[["Pair", "p_holm"]]
df_w_rmse = df_w_rmse.merge(adj_rmse, on="Pair", how="left")
df_w_mae = df_w_mae.merge(adj_mae, on="Pair", how="left")

display(HTML("<h4>11B — Pairwise Wilcoxon (RMSE; 5-fold
eşleştirilmiş)</h4>"))
display(df_w_rmse.sort_values("p"))
display(HTML("<h4>11C — Pairwise Wilcoxon (MAE; 5-fold
eşleştirilmiş)</h4>"))
display(df_w_mae.sort_values("p"))

# ----- 11D (Opsiyonel): Delta RMSE/MAE için paired bootstrap GA + p -----
-----
boot_rows_rmse = []
boot_rows_mae = []
for a,b in combinations(labels, 2):
    br = paired_bootstrap_delta(y, preds[a], preds[b], B=2000, seed=123,
metric=rmse)
    bm = paired_bootstrap_delta(y, preds[a], preds[b], B=2000, seed=124,
metric=mae)

```

```

boot_rows_rmse.append((f"{a} vs {b}", br["delta"], f"{br['ci'][0]:.3f},
{br['ci'][1]:.3f}", br["p"]))

boot_rows_mae.append((f"{a} vs {b}", bm["delta"], f"{bm['ci'][0]:.3f},
{bm['ci'][1]:.3f}", bm["p"]))

df_boot_rmse = pd.DataFrame(boot_rows_rmse, columns=["Pair", "ΔRMSE (A-
B)", "CI95", "p"])
df_boot_mae = pd.DataFrame(boot_rows_mae, columns=["Pair", "ΔMAE (A-
B)", "CI95", "p"])

# Holm (bootstrap p'leri)
df_boot_rmse = df_boot_rmse.merge(holm_bonferroni([(r[0], r[3]) for r in
boot_rows_rmse])[["Pair", "p_holm"]], on="Pair", how="left")
df_boot_mae = df_boot_mae.merge(holm_bonferroni([(r[0], r[3]) for r in
boot_rows_mae])[["Pair", "p_holm"]], on="Pair", how="left")

display(HTML("<h4>11D — ΔRMSE bootstrap (eşleştirilmiş; GA + p;
Holm)</h4>"))
display(df_boot_rmse.sort_values("p"))
display(HTML("<h4>11E — ΔMAE bootstrap (eşleştirilmiş; GA + p;
Holm)</h4>"))
display(df_boot_mae.sort_values("p"))

# ----- Grafikler -----
order = summary.sort_values("RMSE")["Model"].tolist()
def bar(col, title, ylabel=None):
    d = summary.set_index("Model").loc[order].reset_index()
    plt.figure()
    plt.bar(d["Model"], d[col])
    if ylabel: plt.ylabel(ylabel)
    plt.title(title)
    for i,v in enumerate(d[col].values):

```

```

plt.text(i, v, f"{v:.3f}", ha="center", va="bottom")
plt.show()

bar("RMSE", "OOF RMSE (↓ daha iyi)", "RMSE")
bar("MAE", "OOF MAE (↓ daha iyi)", "MAE")
bar("R2", "OOF R2 (↑ daha iyi)", "R2")

print("\n☑ Adım 11 tamamlandı: Özet + Wilcoxon(Holm) + bootstrap ΔGA
tabloları yukarıda.")

=== ADIM 12 — SHAP (6 model: RF/XGB/Cat — CLF & REG) — GÜNCEL
TAM HÜCRE ===
import os, re, warnings; warnings.filterwarnings("ignore")
import numpy as np, pandas as pd, matplotlib.pyplot as plt
from IPython.display import display, HTML

# sklearn
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.metrics import (roc_auc_score, average_precision_score, f1_score,
recall_score,
                             brier_score_loss, mean_squared_error, mean_absolute_error,
r2_score)

# XGBoost / CatBoost / SHAP (gerekirse kur)
try:
    from xgboost import XGBClassifier, XGBRegressor
except Exception:
    os.system("pip -q install xgboost"); from xgboost import XGBClassifier,
XGBRegressor

```

```

try:
    from catboost import CatBoostClassifier, CatBoostRegressor
except Exception:
    os.system("pip -q install catboost"); from catboost import CatBoostClassifier,
    CatBoostRegressor

```

```

try:
    import shap
except Exception:
    os.system("pip -q install shap"); import shap

```

```

shap.initjs()

```

```

# =====
# 0) VERİ & HEDEFLENER
# =====
TARGET_CLF = "AHI_SINIF"
TARGET_REG = "AHI_TOPLAM"

```

```

def _load_df_if_needed():
    if 'df' in globals():
        return globals()['df']
    try:
        from google.colab import files
        print("📁 df yok: CSV seçin...")
        up = files.upload(); assert up, "Dosya seçilmedi."
        fname = next(iter(up.keys()))
        # ASCII'ye çevir (dosya adında Türkçe karakter olabilir)
        ascii_name = re.sub(r"[^A-Za-z0-9_.-]+", "_", fname)
        if ascii_name != fname:
            os.rename(fname, ascii_name)
    
```

```

    fname = ascii_name
    for enc in ["utf-8", "utf-8-sig", "ISO-8859-9", "cp1254", "latin-1"]:
        try:
            _df = pd.read_csv(fname, encoding=enc)
            print(f"☑ CSV okundu: {fname} | enc={enc} | shape={_df.shape}")
            return _df
        except: pass
    raise RuntimeError("CSV okunamadı.")
except Exception as e:
    raise RuntimeError("df yok ve yükleme başarısız.") from e

df = _load_df_if_needed()
assert TARGET_CLF in df.columns and TARGET_REG in df.columns, "Hedef
kolonları bulunamadı."

X_raw = df.drop(columns=[TARGET_CLF, TARGET_REG]).copy()
y_clf = pd.to_numeric(df[TARGET_CLF], errors="coerce").astype(int).values
y_reg = pd.to_numeric(df[TARGET_REG], errors="coerce").astype(float).values

# =====
# 1) ÖN-İŞLEME (sabit) ve ÖZELLİK İSİMLERİ — sürüm uyumlu OHE
# =====
num_cols = X_raw.select_dtypes(include=["number"]).columns.tolist()
cat_cols =
X_raw.select_dtypes(include=["object", "category", "bool"]).columns.tolist()

def _ohe_compat():
    # sklearn >=1.2: sparse_output; daha eski: sparse
    try:
        return OneHotEncoder(handle_unknown="ignore", sparse_output=False)
    except TypeError:
        return OneHotEncoder(handle_unknown="ignore", sparse=False)

```

```

ct = ColumnTransformer(
    transformers=[
        ("num", SimpleImputer(strategy="median"), num_cols),
        ("cat", Pipeline([("imp", SimpleImputer(strategy="most_frequent")),
                           ("ohe", _ohe_compat())]), cat_cols),
    ],
    remainder="drop"
)

```

```

# Fit + transform (tüm veri)
X_proc = ct.fit_transform(X_raw)

# Özellik isimlerini çıkar (kısaltma yok)
feat_names = []
if num_cols:
    feat_names += list(num_cols)
if cat_cols:
    ohe = ct.named_transformers_["cat"].named_steps["ohe"]
    try:
        ohe_names = ohe.get_feature_names_out(cat_cols)
    except TypeError:
        # bazı eski sürümlerde parametre gerekmez
        ohe_names = ohe.get_feature_names_out()
    feat_names += list(ohe_names)

```

```

X_proc = np.asarray(X_proc)
print(f'Özellik sayısı (One-Hot sonrası): {X_proc.shape[1]}')
print("İlk 12 özellik adı:", feat_names[:12])

```

```

# =====

```

```

# 2) MODELLER — tam veri üzerinde (aynı preprocess ile) fit

```

```

# =====

# --- Sınıflama
rf_clf = RandomForestClassifier(
    n_estimators=800, max_depth=None, min_samples_split=5,
    min_samples_leaf=2,
    class_weight="balanced", n_jobs=-1, random_state=42
)
pos = int((y_clf==1).sum()); neg = int((y_clf==0).sum()); spw = max(1.0,
neg/max(1,pos))
xgb_clf = XGBClassifier(
    n_estimators=900, max_depth=6, learning_rate=0.06,
    subsample=0.9, colsample_bytree=0.8, min_child_weight=3,
    gamma=0, scale_pos_weight=spw,
    eval_metric="logloss", tree_method="hist", n_jobs=-1, random_state=42
)
cat_clf = CatBoostClassifier(
    iterations=900, depth=6, learning_rate=0.06, l2_leaf_reg=5,
    border_count=128, loss_function="Logloss", verbose=0, random_seed=42
)

print("\n⚙️ Modeller (CLF) eğitiliyor...")
rf_clf.fit(X_proc, y_clf)
xgb_clf.fit(X_proc, y_clf)
cat_clf.fit(X_proc, y_clf)
print("✓ CLF eğitimleri tamam.")

# --- Regresyon
rf_reg = RandomForestRegressor(
    n_estimators=900, max_depth=None, min_samples_split=5,
    min_samples_leaf=2,
    n_jobs=-1, random_state=42
)

```

```

)
xgb_reg = XGBRegressor(
    n_estimators=900, max_depth=6, learning_rate=0.06,
    subsample=0.9, colsample_bytree=0.8, min_child_weight=3, gamma=0,
    tree_method="hist", n_jobs=-1, random_state=42
)
cat_reg = CatBoostRegressor(
    iterations=900, depth=6, learning_rate=0.06, l2_leaf_reg=5,
    loss_function="RMSE", verbose=0, random_seed=42
)

print("\n🌀 Modeller (REG) eğitiliyor...")
rf_reg.fit(X_proc, y_reg)
xgb_reg.fit(X_proc, y_reg)
cat_reg.fit(X_proc, y_reg)
print("✓ REG eğitimleri tamam.")

# =====
# 3) SHAP — yardımcılar (beeswarm + bar top20)
# =====

def shap_for_model(model, X, feature_names, task_label, model_label, topn=20):
    """
    model: fit edilmiş tree-based model
    X:   numpy array (işlenmiş)
    """
    try:
        explainer = shap.TreeExplainer(model)
        sv = explainer.shap_values(X)
        base = explainer.expected_value

        # Binary sınıflamada list döner → pozitif sınıfı al
        if isinstance(sv, list):

```

```

        sv_plot = sv[1] # class 1
        base_val = base[1] if isinstance(base, (list, np.ndarray)) else base
    else:
        sv_plot = sv
        base_val = base

    shp = np.array(sv_plot)
    print(f"→ {task_label} | {model_label} | SHAP shape: {shp.shape};
base_value≈{np.array(base_val).mean():.4f}")

# Beeswarm
plt.figure(figsize=(10,6))
shap.summary_plot(sv_plot, X, feature_names=feature_names,
plot_type="dot", show=False, max_display=topn)
plt.title(f"SHAP Beeswarm — {task_label} — {model_label}")
plt.show()

# Bar (top N)
plt.figure(figsize=(8,6))
shap.summary_plot(sv_plot, X, feature_names=feature_names,
plot_type="bar", show=False, max_display=topn)
plt.title(f"SHAP Bar (Top {topn}) — {task_label} — {model_label}")
plt.show()

except Exception as e:
    print(f"⚠ {task_label} — {model_label} SHAP hesaplanamadı: {repr(e)}")

# =====
# 4) SHAP — 6 model için üretim
# =====
print("\n=== SHAP — Sınıflama Modelleri ===")

```

```

shap_for_model(rf_clf, X_proc, feat_names, "Sınıflama (AHI_SINIF)",
"RandomForest", topn=20)
shap_for_model(xgb_clf, X_proc, feat_names, "Sınıflama (AHI_SINIF)",
"XGBoost", topn=20)
shap_for_model(cat_clf, X_proc, feat_names, "Sınıflama (AHI_SINIF)",
"CatBoost", topn=20)

print("\n=== SHAP — Regresyon Modelleri ===")
shap_for_model(rf_reg, X_proc, feat_names, "Regresyon (AHI_TOPLAM)",
"RandomForest", topn=20)
shap_for_model(xgb_reg, X_proc, feat_names, "Regresyon (AHI_TOPLAM)",
"XGBoost", topn=20)
shap_for_model(cat_reg, X_proc, feat_names, "Regresyon (AHI_TOPLAM)",
"CatBoost", topn=20)

print("\n☑ Adım 12 tamam: 6 model için SHAP beeswarm + bar grafikleri
üretildi (özellik adları eksiksiz).")
# === RF CLF — SHAP "922 x p x 2" / "list[2]" biçimlerini normalize eden tamir
hücresi ===
import os, re, warnings; warnings.filterwarnings("ignore")
import numpy as np, pandas as pd, matplotlib.pyplot as plt

# Gerekli kütüphaneler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestClassifier

try:
    import shap
except Exception:

```

```

import sys, subprocess; subprocess.run([sys.executable, "-m", "pip", "-q", "install", "shap"], check=False)

import shap
shap.initjs()

TARGET_CLF = "AHI_SINIF"
TARGET_REG = "AHI_TOPLAM"

# --- 1) Ortamı toparla: df / X_proc / feat_names / rf_clf hazır mı? ---
def _ohe_compat():
    try: return OneHotEncoder(handle_unknown="ignore", sparse_output=False)
    except: return OneHotEncoder(handle_unknown="ignore", sparse=False)

need_fit = False

if not all(k in globals() for k in ["df", "X_proc", "feat_names", "rf_clf", "y_clf"]):
    need_fit = True
    # df yoksa yüklet
    if 'df' not in globals():
        try:
            from google.colab import files
            print("📁 df yok: CSV seçin...")
            up = files.upload(); assert up, "Dosya seçilmedi."
            fname = next(iter(up.keys()))
            ascii_name = re.sub(r"^[A-Za-z0-9_.-]+", "_", fname)
            if ascii_name != fname:
                os.rename(fname, ascii_name); fname = ascii_name
            # encoding autodetect
            import pandas as pd
            for enc in ["utf-8", "utf-8-sig", "ISO-8859-9", "cp1254", "latin-1"]:
                try:

```

```

        df = pd.read_csv(fname, encoding=enc); print(f"☑ CSV okundu:
{fname} | enc={enc} | shape={df.shape}")
        break
    except: pass
except Exception as e:
    raise RuntimeError("df bulunamadı ve yükleme başarısız.") from e

assert TARGET_CLF in df.columns and TARGET_REG in df.columns, "Hedef
kolonları bulunamadı."

# Ön-işleme
X_raw = df.drop(columns=[TARGET_CLF, TARGET_REG]).copy()
y_clf = pd.to_numeric(df[TARGET_CLF], errors="coerce").astype(int).values

num_cols = X_raw.select_dtypes(include=["number"]).columns.tolist()
cat_cols =
X_raw.select_dtypes(include=["object", "category", "bool"]).columns.tolist()

ct = ColumnTransformer(
    transformers=[
        ("num", SimpleImputer(strategy="median"), num_cols),
        ("cat", Pipeline([("imp", SimpleImputer(strategy="most_frequent")),
            ("ohe", _ohe_compat())]), cat_cols),
    ],
    remainder="drop"
)
X_proc = ct.fit_transform(X_raw)

feat_names = []
if num_cols: feat_names += list(num_cols)
if cat_cols:
    ohe = ct.named_transformers_["cat"].named_steps["ohe"]

```

```

try:
    ohe_names = ohe.get_feature_names_out(cat_cols)
except TypeError:
    ohe_names = ohe.get_feature_names_out()
feat_names += list(ohe_names)

# RF CLF fit (aynı ayarlar)
rf_clf = RandomForestClassifier(
    n_estimators=800, max_depth=None, min_samples_split=5,
min_samples_leaf=2,
    class_weight="balanced", n_jobs=-1, random_state=42
)
rf_clf.fit(X_proc, y_clf)
print("İ RF sınıflayıcı hızlıca yeniden eğitildi.")

# Güvenlik kontrolleri
X_proc = np.asarray(X_proc)
assert X_proc.shape[1] == len(feat_names), f"Özellik sayısı uyuşmuyor:
X_proc={X_proc.shape[1]} vs names={len(feat_names)}"
print(f"☑ SHAP için giriş: n={X_proc.shape[0]}, p={X_proc.shape[1]} (One-
Hot sonrası).")

# --- 2) SHAP hesapla + normalize et (pozitif sınıf, class=1) ---
explainer = shap.TreeExplainer(rf_clf)
sv = explainer.shap_values(X_proc)      # liste, 2D veya 3D gelebilir
base = explainer.expected_value

def normalize_shap_output(sv, base):
    """
    Dönüş:
    sv_norm: (n, p) — class=1 için SHAP değerleri
    base_val: beklenen değer (class=1)

```

```

"""
# sklearn RF sınıflamada genelde list [class0, class1] döner
if isinstance(sv, list):
    sv_arr = np.array(sv[1])
    base_val = base[1] if isinstance(base, (list, np.ndarray)) else base
else:
    sv_arr = np.array(sv)
    base_val = base
# Bazı sürümlerde (n, p, 2) dönebilir → son eksenini class
if sv_arr.ndim == 3 and sv_arr.shape[-1] == 2:
    sv_arr = sv_arr[..., 1]
# (n,) gelirse (n,1) yap
if sv_arr.ndim == 1:
    sv_arr = sv_arr.reshape(-1, 1)
return sv_arr, base_val

sv_norm, base_val = normalize_shap_output(sv, base)
print(f"→ RF SHAP shape (normalize): {sv_norm.shape} — beklenen: (n, p)")

# Boyut güvenliği
assert sv_norm.shape[0] == X_proc.shape[0], "n (örnek sayısı) tutmuyor."
assert sv_norm.shape[1] == X_proc.shape[1], "p (özellik sayısı) tutmuyor — One-
Hot isimleri ile model girdisi eşleşmiyor."

# --- 3) Beeswarm + Bar(top20) ---
print("🐝 SHAP Beeswarm — RF (Sınıflama, class=1)")
plt.figure(figsize=(10,6))
shap.summary_plot(sv_norm, X_proc, feature_names=feat_names,
plot_type="dot", show=False, max_display=20)
plt.title("SHAP Beeswarm — Sınıflama (AHI_SINIF) — RandomForest")
plt.show()

```

```
print("📊 SHAP Bar (Top 20) — RF (Sınıflama, class=1)")
plt.figure(figsize=(8,6))
shap.summary_plot(sv_norm, X_proc, feature_names=feat_names,
plot_type="bar", show=False, max_display=20)
plt.title("SHAP Bar (Top 20) — Sınıflama (AHI_SINIF) — RandomForest")
plt.show()
```

```
print("\n☑ RF sınıflama için SHAP grafikleri (beeswarm & bar) özellik adları  
TAM olacak şekilde üretildi.")
```

```
# === ADIM 13 — Klinik Eşik Analizi (KURTARICI) ===
# CatBoost OOF'u RAM/diskten bul, doğru formata çevir, en iyi (y,p)
kombinasyonunu seç.
# Sonra: (A) Sens=0.85 sabit, (B) Youden J optimum tabloları + ROC
karşılaştırma.
```

```
import os, numpy as np, pandas as pd, matplotlib.pyplot as plt
from IPython.display import display, HTML
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score,
average_precision_score, auc
from sklearn.model_selection import StratifiedKFold, cross_val_predict
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
import warnings; warnings.filterwarnings("ignore")
```

```
OOF_DIR = "oof_preds"
SENS_TARGET = 0.85
RANDOM_STATE = 42
```

```

# ----- Yardımcılar -----
def sigmoid(z):
    z = np.asarray(z, float)
    return 1.0/(1.0+np.exp(-z))

def as_class1_probs(arr):
    """arr: (n,), (n,2) veya logit aralığında olabilir → class=1 olasılığına döndür."""
    a = np.asarray(arr)
    if a.ndim==2 and a.shape[1]==2:
        a = a[:,1]
    # aralık kontrolü
    if (a.min()<0) or (a.max()>1):
        # muhtemelen logit/score → sigmoid
        a = sigmoid(a)
    return np.clip(a.ravel(), 1e-9, 1-1e-9)

def try_candidates(cands):
    """
    cands: list of dicts with {'name':..., 'y':..., 'p':... (raw)}
    Döndür: en iyi (y,p), rapor
    """
    best = None
    report = []
    for c in cands:
        y = np.asarray(c["y"]).astype(int).ravel()
        p_raw = c["p"]
        p1 = as_class1_probs(p_raw)
        auc1 = roc_auc_score(y, p1)
        # yönü test et (1-p)
        p2 = 1.0 - p1
        auc2 = roc_auc_score(y, p2)

```

```

    if auc2 > auc1:
        pick = p2; aucv = auc2; flip = True
    else:
        pick = p1; aucv = auc1; flip = False
    pr = average_precision_score(y, pick)
    report.append((c["name"], aucv, pr, flip))
    if (best is None) or (aucv > best[2]):
        best = (y, pick, aucv, pr, c["name"], flip)
    return best, report

def holm_bonferroni(pairs): # [(label,p),...]
    dfp = pd.DataFrame(pairs,
columns=["Pair", "p"]).sort_values("p").reset_index(drop=True)
    m = len(dfp); dfp["p_holm"] = [min(1.0, (m-i)*pv) for i,pv in
enumerate(dfp["p"])]
    return dfp

def metrics_at_threshold(y, p, t):
    yhat = (p >= t).astype(int)
    tn, fp, fn, tp = confusion_matrix(y, yhat).ravel()
    sens = tp/(tp+fn) if (tp+fn)>0 else np.nan
    spec = tn/(tn+fp) if (tn+fp)>0 else np.nan
    ppv = tp/(tp+fp) if (tp+fp)>0 else np.nan
    npv = tn/(tn+fn) if (tn+fn)>0 else np.nan
    return dict(Threshold=float(t), Sensitivity=float(sens), Specificity=float(spec),
        PPV=float(ppv), NPV=float(npv), TP=int(tp), FP=int(fp), TN=int(tn),
FN=int(fn))

def stratified_boot_ci(y, p, t, B=1200, seed=42):
    rng = np.random.default_rng(seed)
    idx_pos = np.where(y==1)[0]; idx_neg = np.where(y==0)[0]
    vals = []

```

```

for _ in range(B):
    s = np.concatenate([rng.choice(idx_pos, size=len(idx_pos), replace=True),
                        rng.choice(idx_neg, size=len(idx_neg), replace=True)])
    m = metrics_at_threshold(y[s], p[s], t)
    vals.append([m["Sensitivity"], m["Specificity"], m["PPV"], m["NPV"]])
arr = np.array(vals, float)
lo, hi = np.nanpercentile(arr, [2.5, 97.5], axis=0)
return dict(sens=(lo[0],hi[0]), spec=(lo[1],hi[1]), ppv=(lo[2],hi[2]),
npv=(lo[3],hi[3]))

```

```

def format_row(base, ci):
    return {
        "Threshold": f"{base['Threshold']:.4f}",
        "Sensitivity": f"{base['Sensitivity']:.3f} [{ci['sens'][0]:.3f},
{ci['sens'][1]:.3f}]",
        "Specificity": f"{base['Specificity']:.3f} [{ci['spec'][0]:.3f},
{ci['spec'][1]:.3f}]",
        "PPV": f"{base['PPV']:.3f} [{ci['ppv'][0]:.3f}, {ci['ppv'][1]:.3f}]",
        "NPV": f"{base['NPV']:.3f} [{ci['npv'][0]:.3f}, {ci['npv'][1]:.3f}]",
        "TP": base["TP"], "FP": base["FP"], "TN": base["TN"], "FN": base["FN"]
    }

```

----- 1) Kandidat CatBoost OOF'larını topla -----

```
cand = []
```

RAM adayları

```
if 'oof_cat' in globals():
```

```
    y_ram = np.asarray(globals().get('y_clf', None))
```

```
    if y_ram is not None:
```

```
        cand.append({"name": "RAM:oof_cat + y_clf", "y": y_ram,
"p": globals()['oof_cat']})

```

```

# OOF sözlüğü (bazı oturumlarda kullandık)
if 'oof' in globals() and isinstance(oof, dict):
    if 'cat' in oof:
        y_o = np.asarray(globals().get('y_clf', oof.get('y', None)))
        if y_o is not None:
            cand.append({"name": "RAM:oof['cat'] + y", "y": y_o, "p": oof['cat']})
# Eğer oof içinde 'y' varsa onu da ayrı deneriz
if 'cat' in oof and 'y' in oof:
    cand.append({"name": "RAM:oof['cat'] + oof['y'", "y": oof['y'],
    "p": oof['cat']})

# Diskte NPZ (biz genelde buraya yazdık)
for fname in ["cat_oof.npz", "catboost_oof.npz"]:
    path = os.path.join(OOF_DIR, fname)
    if os.path.exists(path):
        d = np.load(path)
        cand.append({"name": f"DISK:{fname}", "y": d["y"], "p": d["oof"]})

assert cand, "CatBoost OOF bulunamadı. 9C/Adım10'daki OOF export satırlarını
çalıştır."

# ----- 2) Hangi (y,p) en iyi? (otomatik seçim) -----
(best_y, best_p, best_auc, best_ap, best_src, flipped), rep = try_candidates(cand)
print("🔍 CatBoost OOF adayları ve skorlar:")
for nm, a, pr, fl in rep:
    print(f" - {nm:24s} | ROC-AUC={a:.3f} | PR-AUC={pr:.3f} | flipped={fl}")
print(f"✅ Seçilen kaynak: {best_src} | ROC-AUC={best_auc:.3f} | PR-
AUC={best_ap:.3f} | flipped={flipped}")

# Eğer hâlâ çok düşükse (AUC < 0.60), hızlı OOF üret (kurtarma)
if best_auc < 0.60:

```

```

print(" ⚠ AUC < 0.60 — hızlı CatBoost OOF yeniden üretilecek
(iterations=400)...")

# Hızlı OOF (aynı preprocess)
assert 'df' in globals(), "df yok. Başta veri setini yüklemiştik; tekrar yükle ve
8.5-B preprocess'i çalıştır."

TARGET_CLF, TARGET_REG = "AHI_SINIF", "AHI_TOPLAM"
X_raw = df.drop(columns=[TARGET_CLF, TARGET_REG])
y_fix = pd.to_numeric(df[TARGET_CLF], errors="coerce").astype(int).values
num_cols = X_raw.select_dtypes(include=["number"]).columns.tolist()
cat_cols =
X_raw.select_dtypes(include=["object", "category", "bool"]).columns.tolist()
# OHE uyumlu
def _ohe_compat():
    try: return OneHotEncoder(handle_unknown="ignore",
sparse_output=False)
    except: return OneHotEncoder(handle_unknown="ignore", sparse=False)
preprocess = ColumnTransformer(
    transformers=[
        ("num", SimpleImputer(strategy="median"), num_cols),
        ("cat", Pipeline([("imp", SimpleImputer(strategy="most_frequent")),
            ("ohe", _ohe_compat())]), cat_cols),
    ],
    remainder="drop"
)
cat_fast = CatBoostClassifier(iterations=400, depth=6, learning_rate=0.06,
                             l2_leaf_reg=5, border_count=128,
                             loss_function="Logloss", verbose=0,
random_seed=RANDOM_STATE)
pipe = Pipeline([("prep", preprocess), ("model", cat_fast)])
skf = StratifiedKFold(n_splits=5, shuffle=True,
random_state=RANDOM_STATE)

```

```

best_p = cross_val_predict(pipe, X_raw, y_fix, cv=skf,
method="predict_proba", n_jobs=-1)[:,1]
best_y = y_fix
best_auc = roc_auc_score(best_y, best_p); best_ap =
average_precision_score(best_y, best_p)
print(f"☑ Hızlı CatBoost OOF üretildi: ROC-AUC={best_auc:.3f} | PR-
AUC={best_ap:.3f}")
best_src = "FAST:CatBoost(5-fold OOF)"
flipped = False

y_true, y_prob = best_y, np.clip(best_p, 1e-9, 1-1e-9)

# ----- 3) Senaryo A — Sens=0.85 sabit -----
fpr_, tpr_, thr_ = roc_curve(y_true, y_prob)
ixA = np.argmin(np.abs(tpr_ - SENS_TARGET))
thr_A = float(thr_[ixA])
base_A = metrics_at_threshold(y_true, y_prob, thr_A)
ci_A = stratified_boot_ci(y_true, y_prob, thr_A, B=1500, seed=42)
row_A = format_row(base_A, ci_A)

# ----- 4) Senaryo B — Youden's J maksimum -----
ths = np.unique(np.concatenate([[0.0], y_prob, [1.0]]))
best_J, thr_B = -1.0, 0.5
for t in ths:
    m = metrics_at_threshold(y_true, y_prob, t)
    if np.isnan(m["Sensitivity"]) or np.isnan(m["Specificity"]):
        continue
    J = m["Sensitivity"] + m["Specificity"] - 1.0
    # eşitlikte Specificity yüksek olanı seç
    if (J > best_J) or (np.isclose(J, best_J) and m["Specificity"] >
metrics_at_threshold(y_true,y_prob,thr_B)["Specificity"]):
        best_J, thr_B = J, t

```

```

base_B = metrics_at_threshold(y_true, y_prob, thr_B)
ci_B = stratified_boot_ci(y_true, y_prob, thr_B, B=1500, seed=43)
row_B = format_row(base_B, ci_B)

# ----- 5) Tablolar -----
df_A = pd.DataFrame([row_A]); df_B = pd.DataFrame([row_B])
display(HTML(f"<h4>13A — Klinik Eşik (CatBoost • Kaynak: {best_src}) •
Sens={SENS_TARGET:.2f} sabit</h4>")); display(df_A)
display(HTML(f"<h4>13B — Klinik Eşik (CatBoost) • Youden's J
optimum</h4>")); display(df_B)

# ----- 6) ROC karşılaştırma — RF/XGB/Cat -----
# RF & XGB OOF'ları da topla; yoksa hızlıca üret
def get_or_make_oof_rf_xgb():
    outs = {}
    # RAM/disk RF
    if 'oof_rf' in globals(): outs['rf'] = as_class1_probs(globals()['oof_rf'])
    rf_npz = os.path.join(OOF_DIR, "rf_oof.npz")
    if 'rf' not in outs and os.path.exists(rf_npz):
        outs['rf'] = as_class1_probs(np.load(rf_npz)["oof"])
    # RAM/disk XGB
    if 'oof_xgb' in globals(): outs['xgb'] = as_class1_probs(globals()['oof_xgb'])
    xgb_npz = os.path.join(OOF_DIR, "xgb_oof.npz")
    if 'xgb' not in outs and os.path.exists(xgb_npz):
        outs['xgb'] = as_class1_probs(np.load(xgb_npz)["oof"])
    # Eğer yoksa hızlı üret
    need = [m for m in ['rf', 'xgb'] if m not in outs]
    if need:
        assert 'df' in globals(), "df yok. Başta veri setini yüklemiştik; tekrar yükle ve
8.5-B preprocess'i çalıştır."
        TARGET_CLF, TARGET_REG = "AHI_SINIF", "AHI_TOPLAM"
        X_raw = df.drop(columns=[TARGET_CLF, TARGET_REG])

```

```

y_fix = pd.to_numeric(df[TARGET_CLF],
errors="coerce").astype(int).values
num_cols = X_raw.select_dtypes(include=["number"]).columns.tolist()
cat_cols =
X_raw.select_dtypes(include=["object", "category", "bool"]).columns.tolist()
def _ohe_compat():
    try: return OneHotEncoder(handle_unknown="ignore",
sparse_output=False)
    except: return OneHotEncoder(handle_unknown="ignore", sparse=False)
preprocess = ColumnTransformer(
    transformers=[
        ("num", SimpleImputer(strategy="median"), num_cols),
        ("cat", Pipeline([("imp", SimpleImputer(strategy="most_frequent")),
            ("ohe", _ohe_compat())]), cat_cols),
    ],
    remainder="drop"
)
skf = StratifiedKFold(n_splits=5, shuffle=True,
random_state=RANDOM_STATE)
if 'rf' in need:
    rf = RandomForestClassifier(n_estimators=400, min_samples_split=5,
min_samples_leaf=2,
                                class_weight="balanced", n_jobs=-1,
random_state=RANDOM_STATE)
    outs['rf'] = cross_val_predict(Pipeline([("prep", preprocess), ("model",
rf)]),
                                X_raw, y_fix, cv=skf, method="predict_proba",
n_jobs=-1)[: ,1]
if 'xgb' in need:
    pos = int((y_fix==1).sum()); neg = int((y_fix==0).sum()); spw = max(1.0,
neg/max(1,pos))

```

```

xgb = XGBClassifier(n_estimators=600, max_depth=6,
learning_rate=0.06,
                    subsample=0.9, colsample_bytree=0.8, min_child_weight=3,
                    gamma=0, scale_pos_weight=spw, eval_metric="logloss",
                    tree_method="hist", n_jobs=-1,
random_state=RANDOM_STATE)
outs['xgb'] = cross_val_predict(Pipeline([("prep", preprocess), ("model",
xgb)]),
                                X_raw, y_fix, cv=skf, method="predict_proba",
n_jobs=-1)[: ,1]
return outs

oof_map = get_or_make_oof_rf_xgb()
oof_map['cat'] = y_prob # seçilen cat olasılığı

plt.figure(figsize=(7,6))
for lbl, arr in [("RandomForest", oof_map.get('rf', None)),
                ("XGBoost", oof_map.get('xgb', None)),
                ("CatBoost", oof_map.get('cat', None))]:
    if arr is None:
        continue
    f, t, _ = roc_curve(y_true, arr)
    a = auc(f, t)
    plt.plot(f, t, lw=2, label=f"{lbl} (AUC={a:.3f})")

# seçilen eşikleri işaretle (A ve B)
fa, ta, thA = roc_curve(y_true, y_prob)
ixA = np.argmin(np.abs(thA - thr_A))
plt.scatter(fa[ixA], ta[ixA], s=60, marker='o', edgecolor='k', label=f"Sens=0.85
eşik≈{thr_A:.3f}")
fb, tb, thB = roc_curve(y_true, y_prob)
ixB = np.argmin(np.abs(thB - thr_B))

```

```
plt.scatter(fb[ixB], tb[ixB], s=60, marker='s', edgecolor='k', label=f"Youden  
eşik≈{thr_B:.3f}")
```

```
plt.plot([0,1],[0,1], '--', color='grey', alpha=0.6)  
plt.xlabel("False Positive Rate (1-Specificity)")  
plt.ylabel("True Positive Rate (Sensitivity)")  
plt.title("ROC Karşılaştırma — RF vs XGB vs CatBoost (OOF)")  
plt.legend(loc="lower right")  
plt.grid(True, ls="--", alpha=0.5)  
plt.show()
```

```
print("\n☑ Adım 13 kurtarma tamam: Sens=0.85 ve Youden optimum tabloları +  
ROC grafiği yukarıda.")
```

```
# === SHAP YORUM — 6 MODEL (RF/XGB/Cat) • Sınıflama & Regresyon  
===
```

```
# Gerekenler: X_proc, feat_names ve model nesneleri (rf_clf, xgb_clf, cat_clf,  
rf_reg, xgb_reg, cat_reg)
```

```
import numpy as np, pandas as pd, shap  
from IPython.display import display, HTML  
from scipy.stats import spearmanr
```

```
# ---- Güvenlik: Adım 12'den gelen girdiler var mı?
```

```
need_vars = ["X_proc", "feat_names"]
```

```
missing = [v for v in need_vars if v not in globals()]
```

```
assert not missing, f"Gerekli değişkenler eksik: {missing}. Lütfen Adım 12'yi  
çalıştır."
```

```
X_ = np.asarray(X_proc)
```

```
feat_names_ = list(feat_names)
```

```
assert X_.shape[1] == len(feat_names_), "Özellik sayısı X_proc ile feat_names  
eşleşmiyor."
```

```

# ---- Yardımcılar
def _tree_shap_matrix(model, X, task="clf"):
    """
    TreeExplainer ile SHAP çıkarır.
    Binary sınıflamada class=1 SHAP matrisini döndürür.
    Dönüş: (sv: (n,p), base_value: float)
    """
    explainer = shap.TreeExplainer(model)
    sv = explainer.shap_values(X)
    base = explainer.expected_value
    # binary sınıflama: list [class0, class1]
    if isinstance(sv, list):
        sv = np.array(sv[1])
        if isinstance(base, (list, np.ndarray)):
            base = base[1]
    else:
        sv = np.array(sv)
        # bazı kütüphane/sürüm kombinasyonları (n,p,2) döndürebilir
        if sv.ndim == 3 and sv.shape[-1] == 2:
            sv = sv[..., 1]
        if sv.ndim == 1:
            sv = sv.reshape(-1, 1)
    return sv, float(np.atleast_1d(base).mean())

def _dir_arrow(rho):
    if rho >= 0.05:
        return "↑" # değer artınca model çıktısı artma eğiliminde
    if rho <= -0.05:
        return "↓" # değer artınca model çıktısı azalma eğiliminde
    return "±" # yön kararsız / nonlineer / eşik etkisi olabilir

```

```

def shap_summary_table(model, X, feature_names, task_label="Model",
topk=16):
    """
    Çıktı:
        1) Top-k SHAP önem tablosu (mean|SHAP|, %Importance, Spearman  $\rho$ , Dir,
 $\Delta$ SHAP(Q75–Q25))
        2) Metinsel kısa yorumlar (madde madde)
        3) Tam tablo (return)
    """
    sv, _ = _tree_shap_matrix(model, X)
    n, p = sv.shape
    abs_imp = np.abs(sv).mean(axis=0)
    perc = abs_imp / (abs_imp.sum() + 1e-12) * 100.0

    rows = []
    for j in range(p):
        xj = np.asarray(X[:, j]).ravel()
        svj = np.asarray(sv[:, j]).ravel()
        # Spearman korelasyon (özellik değeri ile SHAP değeri arasındaki yöndüren
kanıtı)
        try:
            rho, _ = spearmanr(xj, svj)
        except Exception:
            rho = np.nan
        # Alt/üst çeyrek bazlı SHAP farkı (etki işareti için ikinci bir kanıt)
        q25 = np.nanpercentile(xj, 25)
        q75 = np.nanpercentile(xj, 75)
        lo = svj[xj <= q25].mean() if np.any(xj <= q25) else np.nan
        hi = svj[xj >= q75].mean() if np.any(xj >= q75) else np.nan
        dqt = hi - lo
        rows.append({
            "Feature": feature_names[j],

```

```

    "mean|SHAP|": abs_imp[j],
    "%Importance": perc[j],
    "Spearman_rho": rho,
    "Dir": _dir_arrow(rho if np.isfinite(rho) else 0.0),
    "ΔSHAP(Q75–Q25)": dqt
})

```

```

df_imp = pd.DataFrame(rows).sort_values("mean|SHAP|",
ascending=False).reset_index(drop=True)
df_show = df_imp.iloc[:topk].copy() if topk is not None else df_imp.copy()

```

```

# Yuvarlama ve indeks
df_show["mean|SHAP|"] = df_show["mean|SHAP|"].round(6)
df_show["%Importance"] = df_show["%Importance"].round(2)
df_show["Spearman_rho"] = df_show["Spearman_rho"].round(3)
df_show["ΔSHAP(Q75–Q25)"] = df_show["ΔSHAP(Q75–Q25)"].round(6)
df_show.index = np.arange(1, len(df_show) + 1)

```

```

display(HTML(f"<h4>SHAP Önem Tablosu — {task_label}</h4>"))
display(df_show)

```

```

# Metin üretimi (kısa, madde madde)
print(f"\n--- Otomatik SHAP Yorumları — {task_label} ---")
for i, r in df_show.iterrows():
    name = r["Feature"]
    pct = r["%Importance"]
    rho = r["Spearman_rho"]
    arr = r["Dir"]
    dqt = r["ΔSHAP(Q75–Q25)"]
    if arr == "↑":
        dir_txt = "değer ↑ → model çıktısı ↑ yönünde"
    elif arr == "↓":

```

```

        dir_txt = "değer ↑ → model çıktısı ↓ yönünde"
    else:
        dir_txt = "etki yönü kararsız/olası nonlinear/etkileşim"
    print(f"{i}. {name}: önem {pct:.2f}%, yön {arr} ({dir_txt}); ρ={rho:.2f},
ΔQ75–Q25={dqt:+.4f}")

    return df_imp

# ---- Çalıştırılacak modeller (varsa yorumla; yoksa uyar)
models = [
    ("Sınıflama — RandomForest", "rf_clf"),
    ("Sınıflama — XGBoost", "xgb_clf"),
    ("Sınıflama — CatBoost", "cat_clf"),
    ("Regresyon — RandomForest", "rf_reg"),
    ("Regresyon — XGBoost", "xgb_reg"),
    ("Regresyon — CatBoost", "cat_reg"),
]

all_tables = {}

for label, varname in models:
    if varname in globals() and globals()[varname] is not None:
        print(f"\n☑ {label} yorumlanıyor...")
        df_imp = shap_summary_table(globals()[varname], X_, feat_names_,
task_label=label, topk=16)
        all_tables[label] = df_imp
    else:
        print(f"⚠ {label}: model nesnesi bulunamadı ({varname}). Atlandı.")

print("\n☑ Tamam: 6 model için (bulunabilen) SHAP yorumları üretildi. Üstte
her biri için tablo ve kısa yorumlar var.")

```