ÜSKÜDAR ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY ZEKÂ MÜHENDİSLİĞİ YÜKSEK LİSANS PROGRAMI

2024-2025

FİNAL PROJE RAPORU

ÖĞRENCİ NUMARASI: 234312026

ÖĞRENCİ ADI-SOYADI: BEŞİR ARSLAN

PROJE BAŞLIĞI: MAKİNE ÖĞRENMESİ KULLANILARAK SİROZ HASTA SONUÇLARININ ÇOK SINIFLI TAHMİNİ

İÇİNDEKİLER

Contents

1. ÖZET 2

2. **GİRİŞ** 2

3. MATERYALLER VE YÖNTEMLER 3

4. ARAÇLAR 5

5. KODLAR 5

6. SONUÇLAR 11

7. ÖNERİ VE TARTIŞMALAR 16

8. KAYNAKLAR 16

1. ÖZET

Bu çalışmada, siroz hastalarının sağlık durumlarını tahmin etmek için makine öğrenmesi teknikleri kullanılarak model geliştirilmiştir. Model, hastaların klinik verilerini analiz ederek üç olası sonuç sınıfını (hayatta kalma, ölüm ve karaciğer nakli sonrası yaşam) tahmin etmeyi hedeflemektedir. Geliştirilen karar destek sistemi, doktorların daha hızlı ve isabetli kararlar almasına yardımcı olurken, tedavi süreçlerinin hastalara özel olarak planlanmasına da olanak sağlamaktadır.

Çalışma kapsamında, veri setindeki sınıf dengesizliği problemi ele alınmış, farklı makine öğrenmesi modelleri değerlendirilmiş ve hiper parametre optimizasyonu yapılarak en iyi performansa sahip model seçilmiştir. Performans metrikleri detaylı bir şekilde analiz edilmiştir.

Bu proje, makine öğrenmesinin tıp alanındaki uygulamalarına değerli bir katkı sağlamayı hedeflemekte ve siroz hastalığının yönetiminde yenilikçi bir yaklaşım sunmaktadır.

2. GİRİŞ

Siroz, dünya genelinde ciddi bir sağlık sorunu olarak karşımıza çıkmakta ve karaciğerin kronik hastalıkları arasında önemli bir yer tutmaktadır. Karaciğer fonksiyonlarının kademeli olarak kaybına neden olan bu hastalık, bireylerin yaşam kalitesini önemli ölçüde düşürmekte ve tanı ile tedavi süreçlerini karmaşıklaştırmaktadır. Sirozun ilerlemesi genellikle uzun bir zaman dilimine yayılır ve bu süreç, hastalar üzerinde ağır sonuçlar doğurabilir. Bu bağlamda, hastalığın seyrini tahmin edebilecek güvenilir karar destek sistemlerine duyulan ihtiyaç her geçen gün artmaktadır. Bu tür sistemler, sağlık profesyonellerinin hızlı ve doğru kararlar almasına yardımcı olurken, tedavi süreçlerini hastaya özgü hale getirme imkânı sunmaktadır.

3. MATERYALLER VE YÖNTEMLER

Bu projede kullanılan veri seti, Kaggle platformunda yer alan Playground Series - Season 3, Episode 26 yarışmasından elde edilmiştir. Veri seti, **train.csv**, **test.csv** ve **submission.csv** olmak üzere üç farklı CSV formatında dosya içermektedir. Projede, **train.csv** veri seti üzerinde keşifsel veri analizi (EDA) gerçekleştirilmiştir.

3.1. Veri Setindeki Öznitelik Bilgileri

Veri dosyasındaki [train.csv] her bir satır bir hasta kaydını temsil etmektedir. Toplam 7905 hasta kaydı mevcuttur. Her bir hasta kaydı aşağıdaki özniteliklere sahiptir.

1	Patient_id	Hastaya özel kimlik numarası.
2	N_Days	Tanıdan bu yana geçen gün sayısı.
3	Drug	Kullanılan ilaç türü veya tedavi yöntemi.
4	Age	Hastanın yaşı.
5	Sex	Hastanın cinsiyeti (erkek/kadın).
6	Ascites	Karın bölgesinde sıvı birikimi (evet/hayır).
7	Hepatomegaly	Karaciğer büyümesi (evet/hayır).
8	Spiders	Deride örümcek benzeri damarların varlığı.
9	Edema	Ödem durumu (no/s/yes).
10	Bilirubin	Kandaki bilirubin seviyesi (karaciğer fonksiyon göstergesi).
11	Cholesterol	Kolesterol seviyesi.
12	Albumin	Kandaki albümin seviyesi (karaciğer fonksiyon göstergesi).
13	Copper	Kandaki bakır seviyesi.
14	Alk_Phos	Alkalen fosfataz enzimi seviyesi (karaciğer ve kemik sağlığı göstergesi).
15	SGOT	Aspartat aminotransferaz (AST) seviyesi (karaciğer fonksiyon testi).
16	Trygliceride s	Trigliserit seviyesi (kan yağları).
17	Platelets	Kan trombosit (platelet) sayısı.
18	Prothrombin	Protrombin zamanı (kan pıhtılaşma süresi).
19	Stage	Hastalığın evresi (1/2/3/4).
20	Status	Hastanın durumu (C/D/CL)

3.2.Veri Setinde Keşifsel Veri Analizi ve Ön İşleme

Keşifsel veri analizi kapsamında aşağıdaki işlemler uygulanmıştır:

- **Eksik değerlerin kontrolü** ve eksik değerlerin yönetimi.
- **Kategorik ve nümerik değişkenlerin sınıflandırılması.**
- Kategorik değişkenlerle hedef kolon (**Status**) arasındaki ilişkilerin incelenmesi.
- Nümerik değişkenlerin histogram görselleştirmesi ile dağılımının analizi.
- Veri setinin **korelasyon matrisi** ile ilişkilerinin değerlendirilmesi.
- **Aykırı değerlerin** IQR (interquartile range) kullanılarak hesaplanması
- **Aykırı değerlerin** kutu grafikleri (boxplot) kullanılarak tespiti.

3.2.1. Keşifsel Veri Analizi Sonucu Çıkarım

Yapılan analizler sonucunda, hedef değişkenin **CL (karaciğer nakli sonrası yaşam)** sınıfı için dengesiz bir sınıf dağılımına sahip olduğu tespit edilmiştir. Bu dengesizlik, modelin CL sınıfını doğru bir şekilde ayırt etme performansını olumsuz etkileyebileceği öngörülmüştür. Bu sorunu ele almak amacıyla, model eğitimi sırasında **sınıf ağırlıkları** hesaplanarak denge sağlanmıştır. Sınıf ağırlıkları, her sınıfın veri setindeki temsil oranına ters orantılı olacak şekilde belirlenmiş ve modelin eğitimi sırasında kayıp fonksiyonuna dahil edilmiştir. Bu yöntem, dengesiz veri seti üzerinde modelin tüm sınıfları dengeli bir şekilde öğrenmesini amaçlamıştır.

	precision	recall	f1-score	support
cat	0.67	0.31	0.42	13
fish	0.20	0.67	0.31	3
dog	0.67	0.67	0.67	9
accuracy			0.48	25
macro avg	0.51	0.55	0.47	25
weighted avg	0.61	0.48	0.50	25

Fig.1

3.2.2. Değerlendirme Metrik Seçimi ve Önemi

Model performansının değerlendirilmesinde F1-macro skoru temel metrik olarak kullanılmıştır. F1-macro, her bir sınıf için F1-skorunu ayrı ayrı hesaplar ve ardından bu skorların basit aritmetik ortalamasını alır. Bu metrik, sınıflar arasında bir denge sağladığı ve özellikle dengesiz veri setlerinde küçük sınıfların performansını ihmal etmediği için tercih edilmiştir. F1-weighted ise sınıfların veri setindeki temsil oranlarına göre ağırlıklı bir ortalama alarak genel performansı ölçer. Ancak, bu projede küçük sınıf olan CL'nin doğru tahmin edilmesi kritik olduğundan, tüm sınıfları eşit önemde değerlendiren F1-macro daha uygun bir seçim olarak belirlenmiştir.

3.2.3. Model Eğitim Öncesi Veriyi Bölme

Model eğitimine başlamadan önce train.csv ve test.csv dosyaları incelendi. İnceleme sırasında, hedef sütun olan Status etiketlerinin test.csv veri setinde bulunmadığı fark edildi. Bu nedenle, train.csv dosyası sınıf dengesizliği göz önünde bulundurularak eğitim (train), test ve doğrulama (validation) setlerine ayrıldı. Model, önce eğitim setinde öğrenim yapacak, ardından performansını test setinde değerlendirecek ve en son, hiç görmediği doğrulama seti üzerinde kendini doğrulayarak genelleme başarısını ölçmüş olacak.

3.2.4. Özellik Mühendisliği Adımlar

Özellik mühendisliği, ham verilerden anlamlı özellikler çıkararak modelin performansını artırır. Doğru özellikler, modelin veriyi daha iyi öğrenmesini sağlar, genelleme yeteneğini artırır ve hesaplama maliyetini düşürür. Siroz gibi medikal veri setlerinde, doğru özellik mühendisliği teşhis doğruluğunu ve model başarısını doğrudan etkiler. Projede yeni özellikler türetilmiş ve modelin başarısını nasıl etkilediği gösterilmiştir.

3.2.5. Dengesiz Veri setinde Örneklem Arttırımı

Modelimizin başarısını artırmak için, eğitim veri setindeki dengesiz sınıf dağılımını dengelemek amacıyla **RandomOverSampler** kullanılmıştır. Bu yöntem, azınlık sınıflarını rastgele örnekleyerek tüm sınıfların veri sayısını eşit hale getirmeyi hedeflemiştir.

Oversampling işlemi şu şekilde uygulanmıştır:

1. **Veri Bölme**: Veri seti, eğitim ve test olmak üzere ikiye ayrılmıştır.
2. **Oversampling**:
    - Yalnızca **eğitim veri seti** üzerinde gerçekleştirilmiştir.
    - Bu işlem, modelin öğrenme sürecinde sınıf dengesizliğinin olumsuz etkilerini azaltmayı sağlamıştır.
3. **Test Veri Seti**: Test setinde herhangi bir oversampling işlemi yapılmamış, veri setinin doğal sınıf dağılımı korunarak modelin gerçek dünyadaki performansı değerlendirilmiştir.

Bu süreç, modelin eğitim sırasında dengesiz veri setlerinden kaynaklanan sınıf ayrımına dayalı öğrenme problemlerini önlemekte ve adil bir test değerlendirmesi yapılmasına olanak tanımaktadır.

4. ARAÇLAR

Projede, **Python** temel araç olarak kullanılarak veri analizi, manipülasyonu ve ön işleme süreçleri başarılı bir şekilde yürütülmüştür. Bu süreçlerde, keşifsel veri analizi ile verinin temel yapısı anlaşılmış, gelişmiş modelleme teknikleriyle makine öğrenimi modelleri eğitilmiş ve hiperparametre optimizasyonu uygulanmıştır. Ayrıca, model performansı detaylı görselleştirmelerle analiz edilmiştir.

RapidMiner, veri setine hızlı bir genel bakış sağlamak, model eğitim süreçlerini desteklemek ve görselleştirmelerle sonuçları etkili bir şekilde sunmak için kullanılmıştır. Bu teknolojilerin bir arada kullanımı, projenin doğruluğunu, verimliliğini ve etkili sonuçlar üretme kapasitesini artırmıştır.

5. KODLAR

Rapid Miner ile veri hazırlama ve model eğitimi hızlı ve kolay bir şekilde uygulandı.

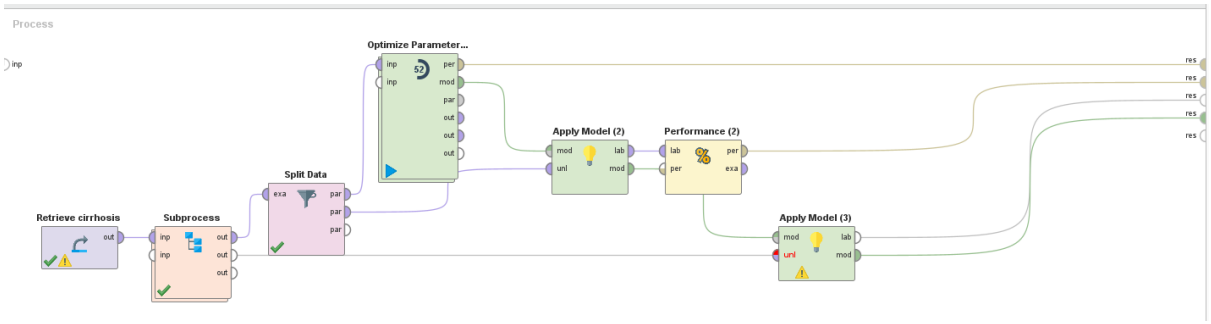

Fig.2. Rapid Miner Akış Görseli

RapidMiner kullanılarak veri seti yüklendikten sonra, **Subprocess** bir prosesin küçük bir birimi olarak düşünülebilir, proseste olduğu gibi, tüm operatörler ve operatör kombinasyonları bir alt proseste uygulanabilir. Bu alt birim aracıyla gerekli kolonlar seçilmiş, hedef kolon tanımlanmış ve nümerik kolonlar tanımlanmıştır.

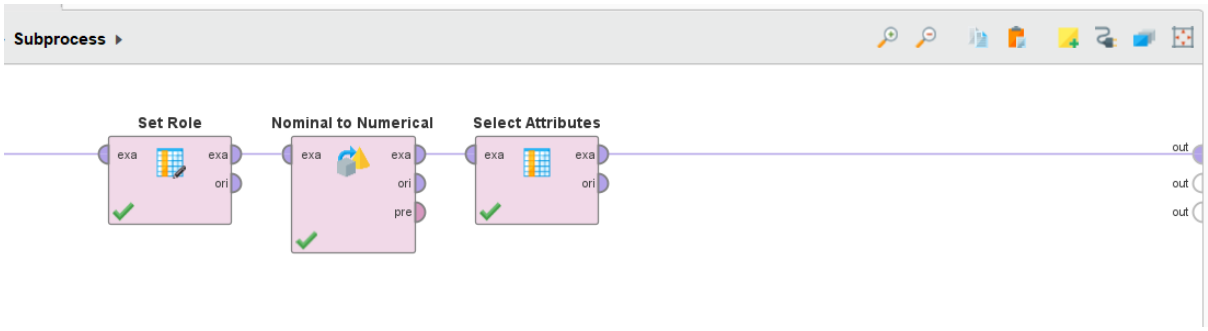

Fig.3. Subprocess Adımı

Bu adımda veri setinde hedef kolon belirlenir. Veri tipleri nümerik format haline getirilir ve modele dahil olacak kolonlar seçilir. **Split Data** fonksiyonu ile veri seti eğitim(%75) ve test(%25) olarak ayrılmıştır.

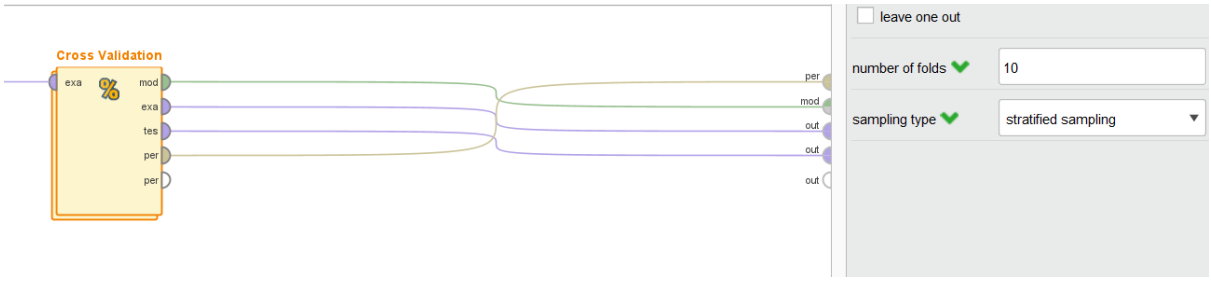

Fig.4. Optimizasyon Adımı

Optimize Paremeter ile cross validasyon işlemi 10 katlı bir şekilde gerçekleştiriliyor. Örneklem tipi dengesiz veri setleri için uygun olan stratified sampling kullanılmıştır. Model eğitiminde kullanılan modelimiz Random Forest olarak belirlenmiştir.

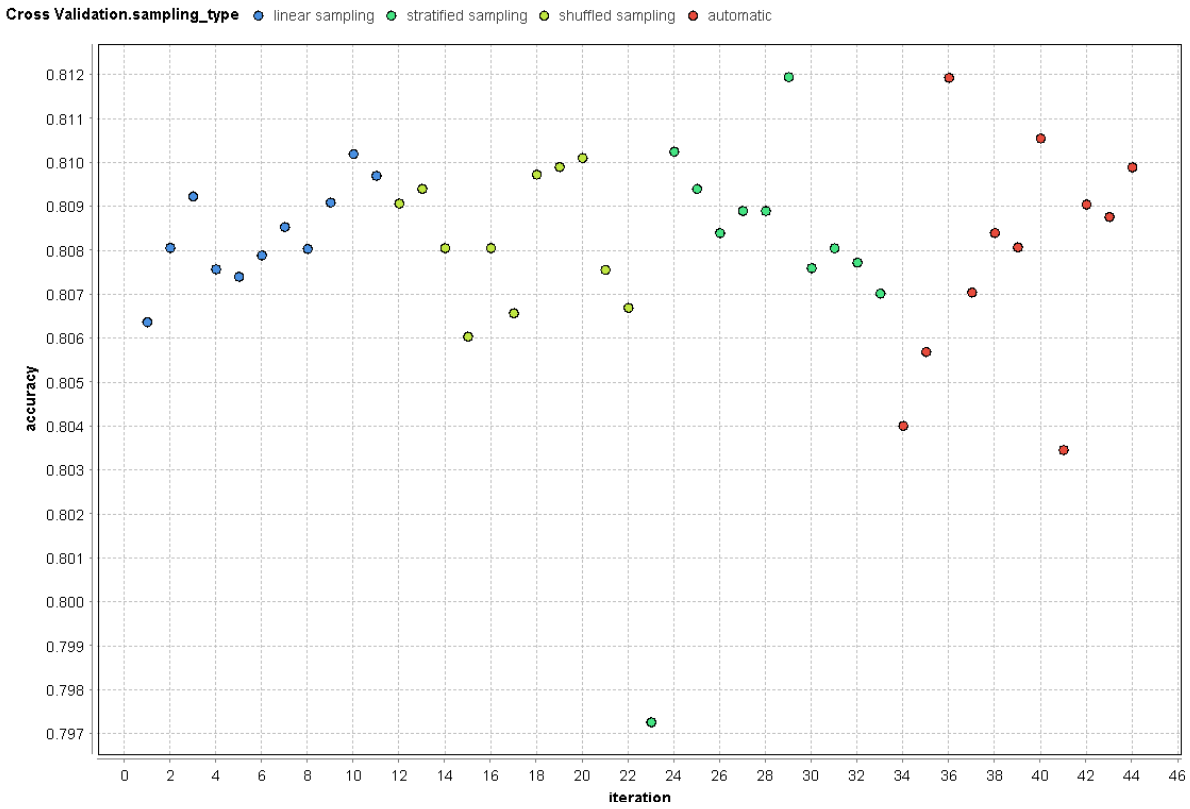

Fig.5. İterasyon başına Başarı Oranı

Her iterasyon ve örneklem tipi için modelimizin başarı oranı yukardaki grafikte çok net bir şekilde gösterilmiştir. Başarı değerlendirme metriği Accuracy olarak tanımlanmıştır. Ancak bu seçim dengesiz veri seti için çok da doğru olmadığı tespit edilmiştir.

weighted_mean_recall: 53.65% +/- 3.16% (micro average: 53.65%), weights: 1, 1, 1

	true D	true C	true CL	class precision
pred. D	1367	276	95	78.65%
pred. C	632	3447	111	82.27%
pred. CL	0	1	0	0.00%
class recall	68.38%	92.56%	0.00%	

Fig.6. Performans Sonuçları

Dengesiz veri setlerinde değerlendirme metrikleri genelde recall ve precision olarak belirlenmektedir. CL sınıfı için modelimiz doğru sonuçlar verememiş hatta burada underfitting olduğunu söylememiz daha doğru olacaktır. CL sınıfı çok azınlıkta olduğu için böyle bir sonuçla karşılaşmamızın normal olduğu tespit edilmiştir.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split,
StratifiedKFold, cross_validate, RandomizedSearchCV
from sklearn.metrics import classification_report, roc_auc_score,
make_scorer, confusion_matrix, ConfusionMatrixDisplay
from sklearn.preprocessing import label_binarize
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
from imblearn.ensemble import BalancedRandomForestClassifier, EasyEnsembleClassifier
from imblearn.pipeline import Pipeline
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
import warnings
warnings.filterwarnings("ignore")
```

Fig.7. Gerekli Kütüphanelerin Yüklenmesi

Makine öğrenmesi projelerinde veri analizi, modelleme, optimizasyon ve değerlendirme için çeşitli kütüphaneler kullanılmıştır. **pandas** ve **numpy**, veri manipülasyonu ve hesaplama için; **matplotlib** ve **seaborn**, görselleştirme için kullanılmıştır. Veri setini bölmek için **train_test_split**, sınıf dengesini korumak için **StratifiedKFold**; çapraz doğrulama ve hiperparametre aramaları için **cross_validate** uygulanmıştır.

Model performansı **classification_report**, **roc_auc_score**, ve **confusion_matrix** ile ölçülmüş; sınıf dengesizliği için **RandomOverSampler**, **BalancedRandomForestClassifier** ve **EasyEnsembleClassifier** kullanılmıştır. Gradyan artırma algoritmaları **XGBClassifier** ve **LGBMClassifier**, hiperparametre optimizasyonu için **RandomizedSearchCV** tercih edilmiştir. Veri işleme adımları **Pipeline** ile düzenlenmiş ve gereksiz uyarılar **warnings.filterwarnings** ile engellenmiştir. Bu araçlar, projeyi daha verimli ve etkili hale getirmiştir.

```python
# 1. Veri Setinin Yüklenmesi
df = pd.read_csv("df_train_prep.csv")
df["Age"] = df["Age"].apply(lambda x: round(x / 365))
df['NEW_ALBI'] = (0.66 * np.log10(df['Bilirubin'] * 17.104) - 0.085 * df['Albumin'] * 10)
df['NEW_MAYORISKSCORE'] = (0.0394 * (df['Age']) + 0.8707 * np.log(df['Bilirubin']) +
                           2.380 * np.log(df['Platelets']) +0.8592 * df['Edema'] +2.533 * np.log(df['Albumin'] * 10))

# 2. Özellik ve Hedef Değişkenlerin Ayrılması
X = df.drop("Status", axis=1)  # Özellikler
y = df["Status"]  # Hedef değişken

# 3. Veriyi Train, Test ve Validasyon olarak Bölme
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.4, stratify=y, random_state=42)
X_test, X_val, y_test, y_val = train_test_split(X_temp, y_temp, test_size=0.5, stratify=y_temp, random_state=42)
```

Fig.8. Veri Yüklenme, Değişken Ayrımı ve Veri Bölme

Veri seti, df_train_prep.csv dosyasından okunarak yüklenmiştir. Yaş değişkeni, gün cinsinden olduğu için 365'e bölünerek yıllık yaş hesaplanmıştır. Ek olarak, ALBI (Albumin-Bilirubin) ve Mayor Risk Skoru gibi yeni özellikler türetilmiştir. ALBI hesaplamasında bilirubin ve albümin değerleri logaritmik dönüşümle birleştirilmiş, Mayor Risk Skoru ise yaş,

bilirubin, platelet, ödem ve albümin değişkenlerinin belirli katsayılarla ağırlıklandırılmasıyla oluşturulmuştur.

Hedef değişken (Status) ile bağımsız değişkenler ayrılmış ve veri seti eğitim, test ve doğrulama olmak üzere üç alt gruba bölünmüştür. İlk aşamada, veri seti train_test_split kullanılarak %60 eğitim ve %40 geçici veri seti olarak ayrılmıştır. Daha sonra geçici veri seti %50-%50 oranında bölünerek test ve doğrulama veri setleri elde edilmiştir. Bölme işlemlerinde sınıf dengesizliğini önlemek amacıyla stratify parametresi kullanılmıştır.

```
# 4. Base Modelleri Tanımlama
models = {
    "Random Forest": BalancedRandomForestClassifier(random_state=42),
    "Easy Ensemble": EasyEnsembleClassifier(random_state=42),
    "XGBoost": XGBClassifier(use_label_encoder=False, eval_metric='mlogloss', random_state=42),
    "LightGBM": LGBMClassifier(force_col_wise=True, random_state=42)
}
```

Fig.9. Base modellerin Tanımlanması

Dengesiz veri setleri için özel tasarlanmış ve gradyan artırma algoritmaları kullanılan dört model tanımlanmıştır. **BalancedRandomForestClassifier** ve **EasyEnsembleClassifier** azınlık sınıfını daha iyi öğrenmek için dengesiz veriye odaklanır. Ancak bu yöntemler daha fazla hesaplama gücü gerektirir.

XGBClassifier ve **LGBMClassifier**, güçlü gradyan artırma algoritmalarıdır. XGBClassifier, log kayıp metriğiyle genelleme yapar, LGBMClassifier ise büyük veri setlerinde daha verimli çalışır. Her iki model de daha uzun eğitim süreleri alabilir. Tüm modellerde, tutarlılık için random_state parametresi kullanılmıştır.

```
# 5. Çapraz Doğrulama Sonuçlarının Depolanması
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
scoring = {
    'accuracy': 'accuracy',
    'f1_macro': 'f1_macro',
    'roc_auc_ovr': make_scorer(roc_auc_score, needs_proba=True, average='macro', multi_class='ovr')
}
cv_results_summary = []
```

Fig.10. Çapraz Doğrulama Tanımlanması

Modelin genel performansını değerlendirmek için çapraz doğrulama (cross-validation) kullanılmıştır. **StratifiedKFold** metodu, her bir katmanda hedef değişkenin sınıf dağılımını koruyarak veriyi böler. Bu, dengesiz veri setlerinde daha doğru sonuçlar elde edilmesini sağlar. Değerlendirme metrikleri olarak **accuracy**, **f1_macro** ve **roc_auc_ovr** seçilmiştir. **Accuracy** genel doğru sınıflandırma oranını ölçerken, **f1_macro** her sınıf için F1 skorlarının ortalamasını alarak dengesiz sınıflarda daha iyi performans gösterir. **roc_auc_ovr** metriği ise çok sınıflı durumlar için ROC AUC skorunu hesaplayarak, modelin her sınıf için ayrım gücünü değerlendirir. Bu şekilde, modelin performansı daha kapsamlı bir şekilde değerlendirilir ve farklı metriklerin kullanımı, sınıf dengesizliğinden kaynaklanabilecek olumsuz etkileri azaltmaya yardımcı olur.

```python
for name, model in models.items():
    # RandomOverSampler'ı tanımla
    oversampler = RandomOverSampler(random_state=42)

    # Oversampling işlemini uygula
    X_train_resampled, y_train_resampled = oversampler.fit_resample(X_train, y_train)

    # Oversampled verilerin boyutlarını ve sınıf dağılımını yazdır
    print(f"\nModel: {name}")
    print("Oversampled X_train shape:", X_train_resampled.shape)
    print("Oversampled y_train shape:", y_train_resampled.shape)
    print("Original class distribution:", Counter(y_train))
    print("Oversampled class distribution:", Counter(y_train_resampled))

    pipeline = Pipeline([
        ('oversampler', RandomOverSampler(random_state=42)),
        ('model', model)
    ])
    cv_results = cross_validate(pipeline, X_train, y_train, cv=cv, scoring=scoring)
    cv_results_summary.append({
        "Model": name,
        "Accuracy": np.mean(cv_results['test_accuracy']),
        "F1 Macro": np.mean(cv_results['test_f1_macro']),
        "ROC AUC": np.mean(cv_results['test_roc_auc_ovr'])
    })

cv_results_df = pd.DataFrame(cv_results_summary)
```

Fig.11. Eğitim Verisinde Sampling İşlemi ve Model Tanımı

Her model için **RandomOverSampler** kullanılarak **eğitim** veri setindeki azınlık sınıfın sayısı artırılmıştır. **RandomOverSampler**, azınlık sınıfı örneklerini çoğaltarak veri setindeki dengesizliği gidermeyi amaçlar. Bu işlem, eğitim setinin boyutlarını büyütür ve sınıf dağılımını daha dengeli hale getirir.

Her model için önce bu oversampling işlemi uygulanır ve ardından **Pipeline** oluşturularak modelin eğitimi yapılır. Pipeline, iki adımdan oluşur: **RandomOverSampler** ile veri setinin dengelenmesi ve modelin eğitilmesi. Çapraz doğrulama (**cross_validate**) kullanılarak her modelin **accuracy**, **f1_macro** ve **roc_auc_ovr** metrikleri hesaplanır.

Sonuçlar, her modelin performansını özetleyen bir liste (`cv_results_summary`) olarak saklanır ve bu liste sonunda bir **DataFrame** (`cv_results_df`) olarak derlenir. Bu işlem, her modelin oversampling sonrası performansını karşılaştırmak için kullanılır.

```python
# 6. Test Verisi Üzerinde Performans Değerlendirmesi
initial_test_results = []

for name, model in models.items():
    pipeline = Pipeline([
        ('oversampler', RandomOverSampler(random_state=42)),
        ('model', model)
    ])
    pipeline.fit(X_train, y_train)
    y_test_pred = pipeline.predict(X_test)
    y_test_binarized = label_binarize(y_test, classes=[0, 1, 2])
    roc_auc = roc_auc_score(y_test_binarized, pipeline.predict_proba(X_test), average="macro", multi_class="ovr")
    f1_macro = classification_report(y_test, y_test_pred, output_dict=True)['macro avg']['f1-score']
    initial_test_results.append({
        "Model": name,
        "F1 Macro": f1_macro,
        "ROC AUC": roc_auc
    })

initial_test_results_df = pd.DataFrame(initial_test_results)
```

Fig.12. Test verisi Performans Değerlendirilmesi

Bu kodda, **RandomOverSampler** yalnızca eğitim verisi üzerinde uygulanmaktadır. Her model için oluşturulan **Pipeline** içinde, eğitim verisi (**X_train, y_train**) oversampling işlemine tabi tutulur. Ancak, test verisi (**X_test**) üzerinde herhangi bir oversampling yapılmaz. Eğitim sırasında oversampling, modelin sınıf dengesizliklerini daha iyi öğrenmesini sağlar, fakat test verisi orijinal haliyle kullanılarak modelin genel performansı değerlendirilir. Kodda, her modelin eğitim süreci tamamlandıktan sonra test verisi üzerinde tahminler yapılır ve **F1 Macro** ile **ROC AUC** skorları hesaplanır. Bu metrikler, modelin test verisi üzerindeki genel başarısını ölçmek için kullanılır. Sonuçlar, her modelin **F1 Macro** ve **ROC AUC** skorlarını içeren bir listeye eklenir ve bu liste daha sonra bir **DataFrame**'e dönüştürülerek performans karşılaştırması yapılır.

6. SONUÇLAR

Train verisi üzerinde over sampling işlemi uygulanmıştır. Daha sonra veri, kendi içinde train-test split yöntemiyle bölünmüştür. Model eğitimi, 5 katlı çapraz doğrulama (n=5) ile gerçekleştirilerek daha genellenebilir sonuçlar elde edilmeye çalışılmıştır.

	Model	Accuracy	F1 Macro	ROC AUC
0	Random Forest	0.814255	0.625358	0.864922
1	Easy Ensemble	0.718742	0.575045	0.848143
2	XGBoost	0.814674	0.630651	0.869763
3	LightGBM	0.812568	0.639744	0.873289

Fig.13. Train verisi- Çapraz Doğrulama Sonuçları

Veri arttırma işlemi yalnızca eğitim verisi üzerinde uygulanmıştır. Test verisi üzerinde herhangi bir örnek arttırma işlemi yapılmamıştır. Bu yaklaşım, verinin doğallığını koruyarak daha güvenilir sonuçlar elde etmeyi hedeflemektedir.

	Model	F1 Macro	ROC AUC
0	Random Forest	0.610111	0.869294
1	Easy Ensemble	0.549522	0.830596
2	XGBoost	0.629649	0.852436
3	LightGBM	0.632730	0.860771

Fig.14. Test verisi- Performans Değerlendirmesi

Model sonuçlarının başarı metriklerinin görselleştirilmesi.

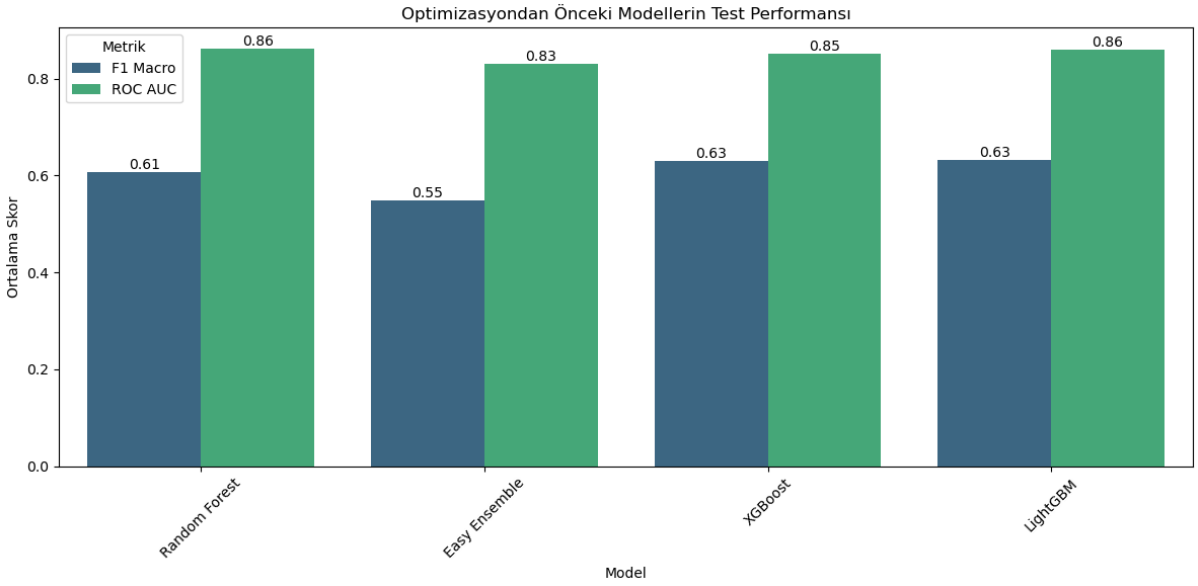

Fig.15. Test Performans Grafikleri

Uyguladığımız modeller üzerinden hiper parametre optimizasyonları gerçekleştirilmiştir. Hiper parametre optimizasyonunun amacı, modelin performansını artırmak için en uygun parametre kombinasyonlarını bulmak ve böylece daha doğru ve genellenebilir sonuçlar elde

etmektir.

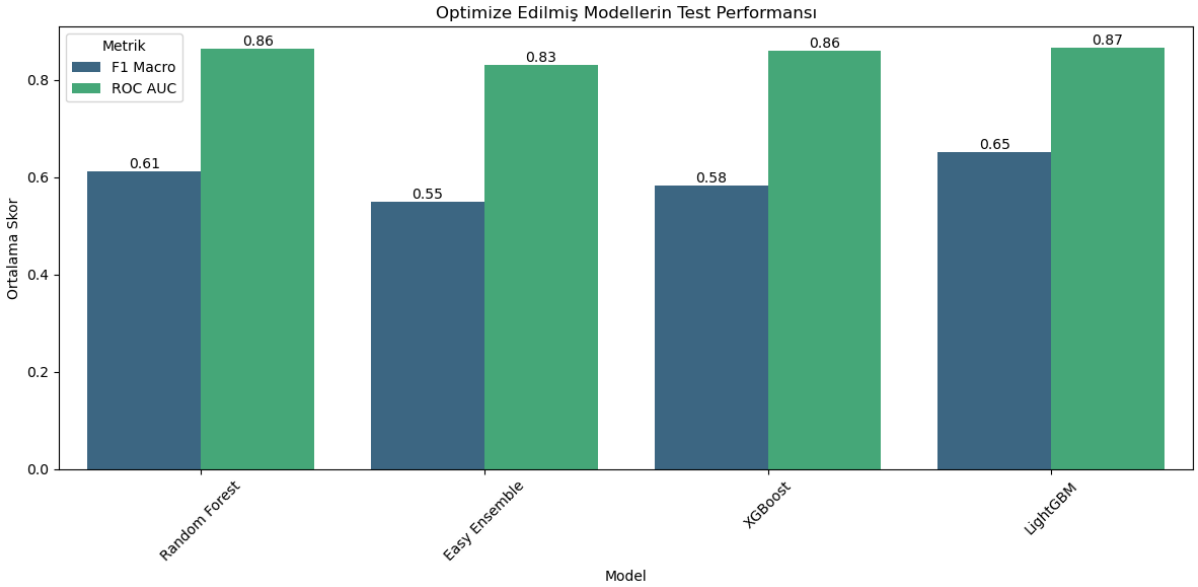

Fig.16. Optimizasyon sonrası Test Performans Grafikleri

Modellerin karmaşıklık matrisi incelenmiştir. Modellerin f1-macro ve ROC-AUC performans metriklerini karşılaştırarak, hangi modelin belirli bir veri seti üzerinde daha iyi performans gösterdiğini değerlendirmemize olanak tanır. En iyi iki modelin karmaşıklık matrisi çizdirilmiştir.

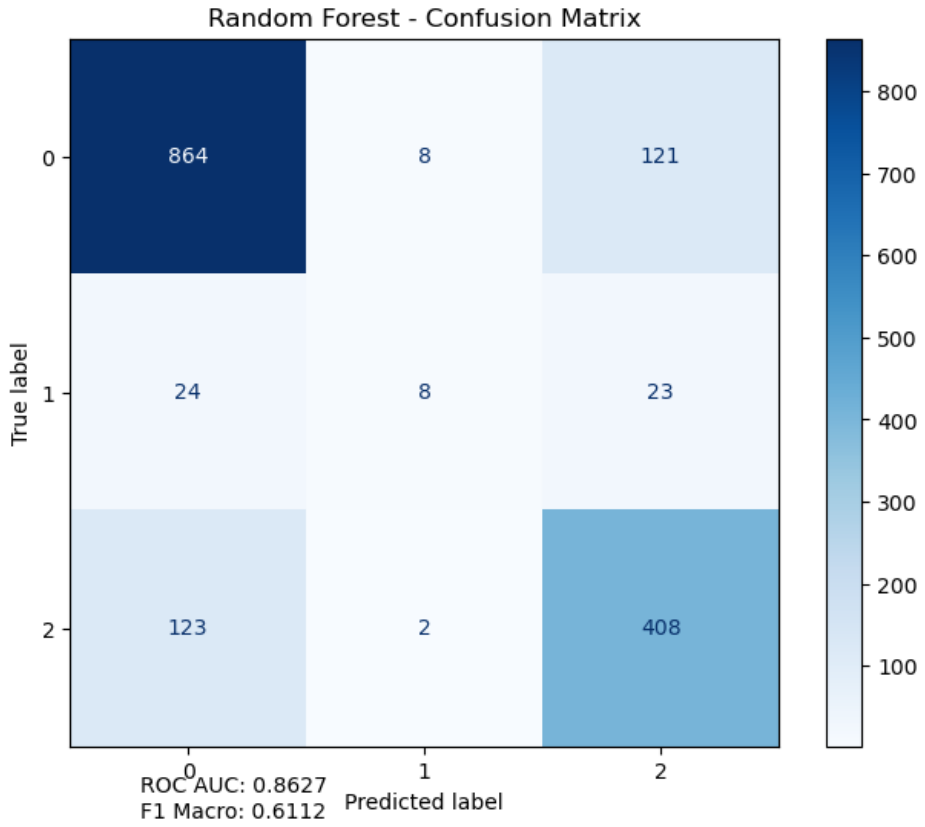

Fig.17. Random Forest Karmaşıklık Matriksi

• Sınıf 0'da yüksek doğrulukla tahmin yapmış (864 doğru, 121 yanlış).
• Sınıf 1 ve 2'de performans düşüyor, özellikle Sınıf 1'de tahminler zayıf (8 doğru).
• ROC AUC: 0.8627, F1 Macro: 0.6112 ile makul bir performans sergiliyor ancak dengeli değil.

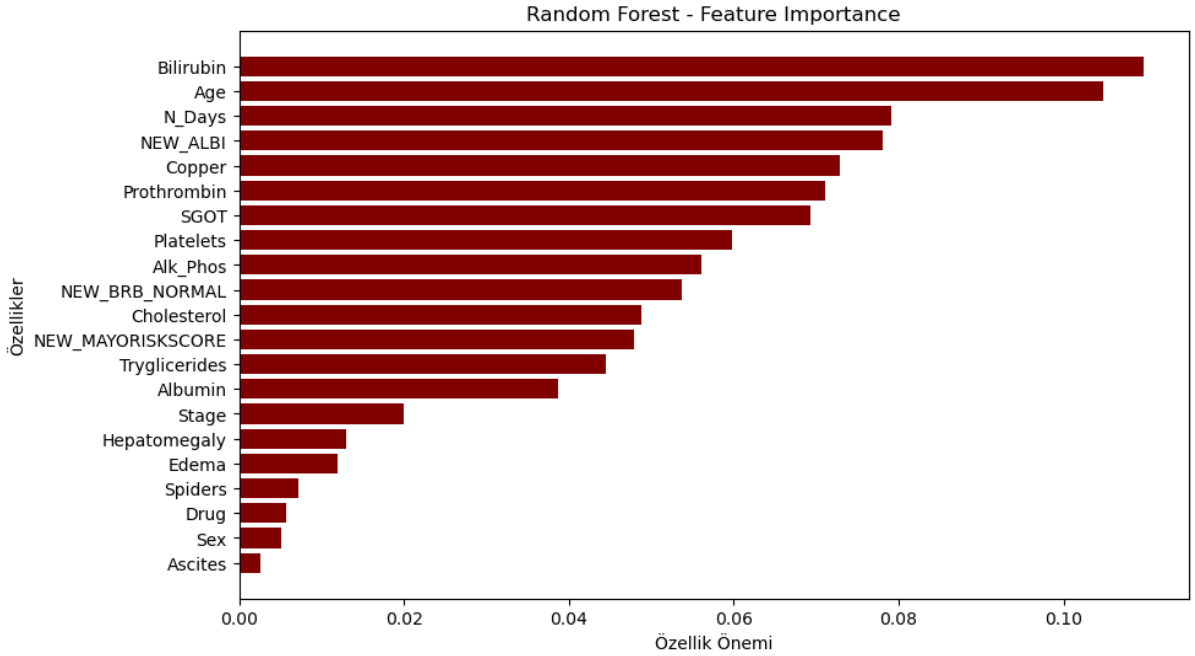

Fig.18. RF Özellik Önem Görselleştirilmesi

Random Forest modeli için feature importance grafiği, modelin karar verirken hangi özelliklere ne kadar ağırlık verdiğini gösterir.

- **En yüksek önem taşıyan özellikler** grafiğin üstünde, diğerleri aşağıda sıralanır.
- Önemli özellikler, modelin performansını artırmada daha etkili olanlardır.
- Eğer belirli bir özellik çok düşük bir değere sahipse, model için pek katkı sağlamıyor olabilir.

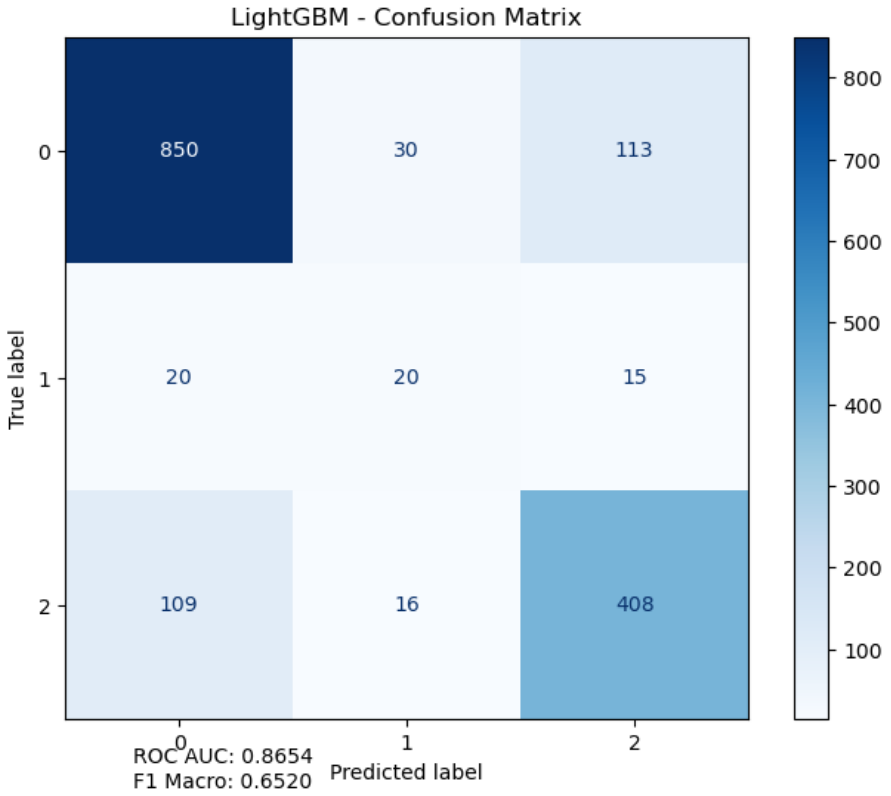

Fig.19. LightGBM Karmaşıklık Matriksi

• Sınıf 0'da benzer bir doğruluk var (850 doğru, 113 yanlış), ancak Sınıf 1 için performansı daha iyi (20 doğru, 30 yanlış).
• Sınıf 2 için tahminler yine zayıf (408 doğru, 109 yanlış).
• ROC AUC: 0.8654, F1 Macro: 0.6520 ile Random Forest'tan biraz daha iyi bir genel performans gösteriyor.

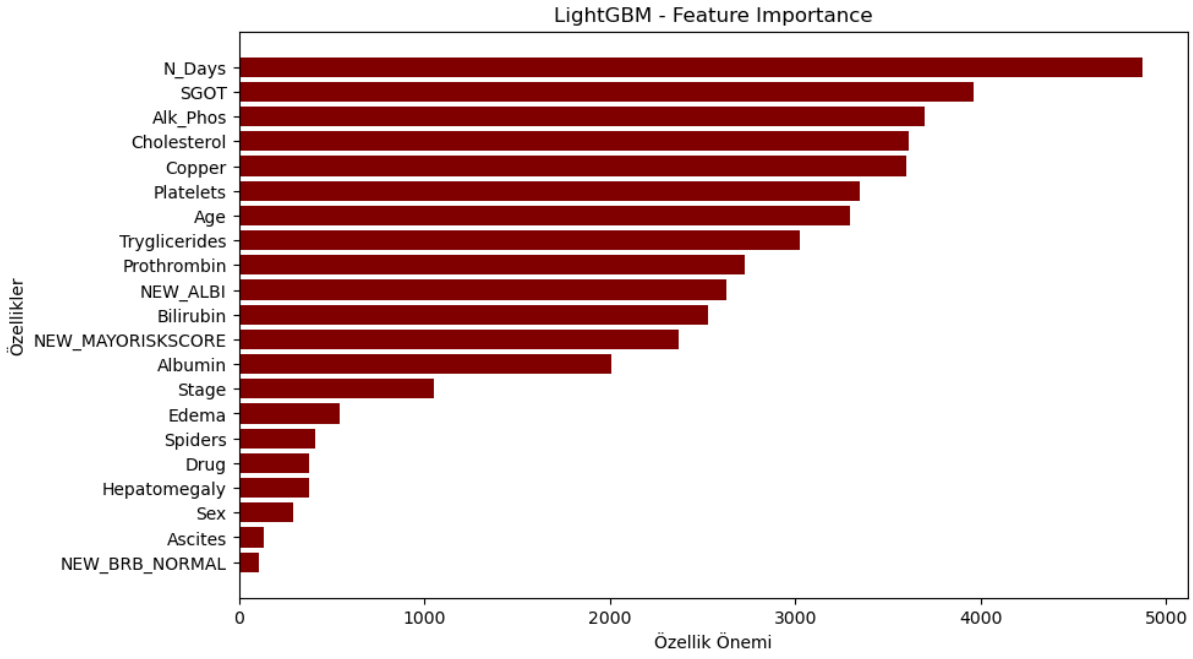

Fig.20. LightGBM Özellik Önem Görselleştirilmesi

Modelin tahmin yaparken hangi özelliklere daha fazla güvendiğini gösterir. LightGBM bu önemi, ağaçlardaki bölünmelerin sıklığı veya kazandırdığı bilgiye göre hesaplar.

- Üstteki özellikler: Kararları en çok etkileyenlerdir.
- Alttakiler: Daha az katkı sağlar, gerekirse çıkarılabilir.

7. ÖNERİ VE TARTIŞMALAR

Projede, ilk adım olarak veri analizi yapıldıktan sonra, herhangi bir veri artırımı uygulanmadan ve sınıf dengesizliği korunarak model eğitimi gerçekleştirilmiştir. Bu süreçte, temel odak noktamız olan **F1-Macro skoru**, en iyi modelde %52 olarak gözlemlenmiştir.

Başarı oranını artırmak amacıyla, yalnızca eğitim verisi üzerinden örnek artırımı uygulanmıştır. Bu yöntemle elde edilen sonuçlar, önce eğitim verisi üzerinde, ardından test verisi ile karşılaştırmalı olarak değerlendirilmiştir. Bu kapsamda, LightGBM modeli ile %65'lik bir F1-Macro skoru elde edilerek gözle görülür bir başarı artışı sağlanmıştır.

Projenin gelecekteki geliştirme aşamaları için şu öneriler sunulmaktadır:

1. **Hiperparametre optimizasyonu**: Daha iyi hiperparametre değerleri deneyerek modelin performansı artırılabilir.
2. **Aykırı değerlerin işlenmesi**: Outlier değerler baskılanarak modelin genel tahmin gücü iyileştirilebilir.
3. **Özellik seçimi**: Özellik önemi grafiğinde düşük katkıya sahip olan özellikler veri setinden çıkarılarak model sadeleştirilebilir.
4. **Yeni özelliklerin eklenmesi**: Veri setine yeni özellikler eklenerek model yeniden eğitilebilir ve başarı oranı test edilebilir.

Sonuç olarak, proje kapsamında gerçekleştirilen örnek artırımı ve diğer yöntemler, başarı oranında anlamlı bir iyileşme sağlarken, önerilen ek adımlar ile model performansı daha da geliştirilebilir.

8. KAYNAKLAR

1. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/
2. https://scikit-learn.org/stable/
3. https://pubmed.ncbi.nlm.nih.gov/37159031/
4. https://www.geeksforgeeks.org/confusion-matrix-machine-learning/
5. https://www.kaggle.com/competitions/playground-series-s3e26/data
6. https://altair.com/altair-rapidminer

KONTROL LİSTESİ

	EVET /HAYIR
Çalışmanızı hem bir araç kullanarak hem de kod yazarak mı hazırladınız?	EVET
Raporunuzu şablonda belirtildiği gibi hazırladınız mı?	EVET
Çalışmanızın sonuçlarını (print screen) rapora eklediniz mi?	EVET
Rapor dosyanızı şablondaki gibi yeniden adlandırdınız mı?	EVET
Projedeki tüm dosyaları (proje dokümanı, kodlar, veri seti vb.) "ogrencino_adsoyad.zip" ismi ile sıkıştırılmış halde tek bir dosya olarak yüklediniz mi?	EVET