Makine Öğrenmesi ile Güneş Patlaması (Solar Flare) Tahmini

Eda VURAL
220205010
Yazılım Mühendisliği

Onur YERLİKAYA
220205066
Yazılım Mühendisliği

Öz – Bu çalışmada, çeşitli makine öğrenmesi algoritmalarının Güneş patlamalarını (Solar Flare) tahmin etmedeki etkinliği incelenmiştir. Güneş patlamaları, Dünya üzerindeki iletişim sistemlerini ve uyduları etkileyebilecek potansiyele sahip olduğundan, önceden tahmin edilmesi kritik öneme sahiptir. Bu amaçla, Random Forest, Gradient Boosting (XGBoost), Destek Vektör Makineleri (SVM) ve Yapay Sinir Ağları (ANN) algoritmaları kullanılarak patlama olup olmayacağının sınıflandırılması gerçekleştirilmiştir. Kullanılan veri seti, UCI Machine Learning Repository platformunda bulunan Solar Flare veri setidir. Veri setindeki dengesizlik problemini çözmek için SMOTETomek yöntemi kullanılmıştır. Çalışmada, Gradient Boosting algoritması %76.88 doğruluk oranı ile en iyi sonucu vermiştir. Random Forest %74.69, Optimize SVM %72.81 ve Derin ANN %71.25 doğruluk oranlarına ulaşmıştır. Bu sonuçlar, topluluk (ensemble) öğrenme yöntemlerinin güneş aktivitelerini tahmin etmede etkili araçlar olduğunu göstermektedir.

Anahtar Kelimeler – Makine Öğrenmesi, Güneş Patlaması, Gradient Boosting, Random Forest, Destek Vektör Makineleri, Yapay Sinir Ağları, Veri Dengeleme.

1. GİRİŞ

Son yıllarda uzay havası (space weather) ve güneş aktiviteleri, Dünya üzerindeki teknolojik altyapılar için kritik bir risk faktörü haline gelmiştir. Türk Dil Kurumu (TDK) tarafından veri, bir araştırmanın veya tartışmanın temeli olan ana öge olarak tanımlanmıştır. Günümüzde bu veriler işlenerek bilgiye dönüştürülmekte ve bu dönüşümde makine öğrenmesi algoritmaları başrolü oynamaktadır. Makine öğrenmesi, insan müdahalesi olmadan verilerden desenler çıkararak geleceğe yönelik tahminler yapmayı amaçlayan yapay zekanın bir alt dalıdır.

Güneş patlamaları (Solar Flares), Güneş atmosferindeki ani enerji boşalmalarıdır ve yaydıkları yüksek enerjili parçacıklar ile Dünya'nın manyetosferini etkileyebilmektedir. Bu patlamalar, özellikle GPS sistemleri, radyo iletişimi, uydular ve elektrik şebekeleri üzerinde yıkıcı etkilere sahip olabilir. Bu nedenle, Güneş patlamalarının önceden tahmin edilmesi, olası zararların minimize edilmesi açısından büyük önem taşımaktadır. Tıpkı tıpta kanser teşhisinde kullanılan makine öğrenmesi yöntemlerinin erken teşhisle hayat kurtarması gibi, uzay havası tahminlerinde de bu yöntemler teknolojik altyapıyı korumaktadır.

Bu çalışmada, UCI Machine Learning Repository'den alınan Solar Flare veri seti kullanılarak, Güneş patlamalarının oluşup oluşmayacağının tahmini yapılmıştır. Literatürdeki benzer çalışmalardan farklı olarak, bu çalışmada sınıf dengesizliği (class imbalance) problemini çözmek için SMOTETomek hibrit yöntemi kullanılmış ve modellerin başarısı artırılmıştır. Çalışmada kullanılan algoritmalar arasında Random Forest, Gradient Boosting (XGBoost), Destek Vektör Makineleri (SVM) ve Yapay Sinir Ağları (ANN) bulunmaktadır. Bu algoritmaların performansları doğruluk (accuracy), hassasiyet (precision), geri çağırma (recall) ve F1-skoru gibi metriklerle detaylıca incelenmiştir.

Çalışma, literatür taraması ile başlayıp, kullanılan materyal ve yöntemlerin (veri seti, ön işleme, algoritmalar) detaylı anlatımı ile devam edecek ve deneysel sonuçların karşılaştırılması ile tamamlanacaktır.

2. LİTERATÜR TARAMASI

Güneş patlamalarının tahmini, uzay havası (space weather) çalışmalarının en kritik alanlarından biridir. Son yıllarda, yüksek çözünürlüklü uydu verilerinin artmasıyla birlikte, fizik tabanlı modellerden veri odaklı makine öğrenmesi modellerine doğru bir geçiş yaşanmıştır. Literatürde bu alanda yapılan önemli çalışmalar aşağıda özetlenmiştir.

Bobra ve Couvidat (2015), Solar Dynamics Observatory (SDO) uydusundan elde edilen vektör manyetik alan verilerini kullanarak Destek Vektör Makineleri (SVM) algoritması ile M ve X sınıfı patlamaları tahmin etmiştir.

Çalışmalarında, manyetik alanın topolojisi ve depolanan manyetik enerji gibi 25 farklı özniteliği kullanmışlar ve sınıf dengesizliği (class imbalance) probleminin tahmin başarısını doğrudan etkilediğini vurgulamışlardır. Bu çalışma, güneş patlaması tahmininde makine öğrenmesinin bir mihenk taşı olarak kabul edilir.

Nishizuka ve ark. (2017), "Deep Flare Net" adını verdikleri bir Derin Sinir Ağı (DNN) modeli geliştirmişlerdir. Bu model, güneş lekelerinin özelliklerini ve zaman içindeki değişimlerini analiz ederek patlama olasılığını hesaplamıştır. Çalışma sonuçları, derin öğrenme yöntemlerinin (Deep Learning), geleneksel yöntemlere kıyasla özellikle karmaşık ve doğrusal olmayan veri setlerinde daha yüksek başarı (True Skill Statistic > 0.80) sağladığını göstermiştir.

Huang ve ark. (2018) ve Liu et al. (2017), Random Forest ve Lojistik Regresyon gibi algoritmaları karşılaştırmalı olarak incelemişlerdir. Random Forest gibi topluluk (ensemble) öğrenme yöntemlerinin, tekli karar ağaçlarına göre aşırı öğrenme (overfitting) riskini azalttığı ve kararlılığı artırdığı sonucuna varmışlardır.

Güncel çalışmalarda ise XGBoost (Extreme Gradient Boosting) algoritmasının yükselişi dikkat çekmektedir. 2023 ve 2025 yıllarında yapılan karşılaştırmalı analizlerde, XGBoost'un hem eğitim hızı hem de sınıflandırma başarısı (Accuracy ve F1-Score) açısından diğer algoritmaları geride bıraktığı rapor edilmiştir. Özellikle bizim çalışmamızda da kullandığımız SMOTE ve benzeri veri dengeleme tekniklerinin, nadir görülen büyük patlamaların (X sınıfı) tespit edilmesinde kritik rol oynadığı literatürde sıkça belirtilmektedir.

Bu çalışma, literatürdeki bu güçlü yöntemleri (XGBoost, Random Forest, SVM, ANN) bir araya getirerek ve SMOTETomek gibi hibrit bir dengeleme yöntemi kullanarak mevcut literatüre katkı sağlamayı amaçlamaktadır.

3. TEORİK ALTYAPI VE MATERYAL

3.1. Teorik Altyapı ve Matematiksel Modeller

Bu çalışmada kullanılan algoritmaların teorik temelleri aşağıda açıklanmıştır:

3.1.1. Destek Vektör Makineleri (SVM) Matematiği

SVM, verileri iki sınıfa ayıran en uygun hiper düzlemi (hyperplane) bulmayı hedefler. Hiper düzlem şu denklemle ifade edilir:

$$w \cdot x + b = 0$$

Burada w ağırlık vektörü, x girdi vektörü ve b ise sapma (bias) değeridir. SVM, sınıflar arasındaki marjini (margin) maksimize etmek için şu optimizasyon problemini çözer:

$$\min \frac{1}{2}||w||^2$$

Verilerin doğrusal olarak ayrılamadığı durumlarda, bu çalışmada da kullandığımız RBF (Radyal Tabanlı Fonksiyon) çekirdeği (kernel trick) kullanılarak veriler daha yüksek boyutlu bir uzaya taşınır.

3.1.2. Random Forest ve Entropi Random Forest

Random Forest ve Entropi Random Forest çok sayıda karar ağacının (Decision Tree) oluşturduğu bir topluluktur. Karar ağaçlarında düğümlerin (node) bölünmesi, Gini Safsızlığı (Gini Impurity) veya Entropi değerine göre yapılır. Bir S kümesinin entropisi şu formülle hesaplanır:

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Burada pi, i sınıfının olasılığıdır. Random Forest, bu ağaçların çoğunluk oylaması (majority voting) ile nihai kararı verir.

3.1.3. Gradient Boosting (XGBoost)

XGBoost (Extreme Gradient Boosting), zayıf öğrenicileri (genellikle sığ ağaçlar) sırayla eğiterek güçlü bir model oluşturur. Her yeni ağaç, önceki ağacın hata fonksiyonunu (Loss Function) minimize edecek şekilde eğitilir. Hedef fonksiyonu şu şekildedir:

$$Obj(\Theta) = L(\theta) + \Omega(\Theta)$$

Burada L, eğitim hatasını (Training Loss), Ω ise modelin karmaşıklığını cezalandıran düzenlileştirme (Regularization) terimidir. Bu yapı, modelin ezberlemesini (overfitting) engeller.

3.1.4. Yapay Sinir Ağları (ANN)

Kullanılan Çok Katmanlı Algılayıcı (MLP), girdi katmanı, gizli katmanlar ve çıktı katmanından oluşur. Her nöronun çıkışı bir aktivasyon fonksiyonundan geçer. Bu çalışmada ReLU (Rectified Linear Unit) aktivasyon fonksiyonu kullanılmıştır:

$$f(x) = \max(0, x)$$

Modelin ağırlıkları, hatayı geriye doğru yayan Geri Yayılım (Backpropagation) algoritması ile güncellenir.

3.2. Performans Değerlendirme Metrikleri

Modellerin başarısını ölçmek için Karmaşıklık Matrisi (Confusion Matrix) tabanlı metrikler kullanılmıştır.

- Doğruluk (Accuracy): Toplam doğru tahminlerin tüm verilere oranıdır.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Hassasiyet (Precision): Pozitif olarak tahmin edilenlerin ne kadarının gerçekten pozitif olduğunu gösterir. Yanlış alarmları (False Positive) minimize etmek için önemlidir.

$$Precision = \frac{TP}{TP + FP}$$

- Duyarlılık (Recall): Gerçek pozitiflerin (patlamaların) ne kadarının model tarafından yakalandığını gösterir. Güneş patlaması gibi kritik olaylarda bu değerin yüksek olması hayati önem taşır.

$$Recall = \frac{TP}{TP + FN}$$

- F1-Skoru: Hassasiyet ve Duyarlılığın harmonik ortalamasıdır. Dengesiz veri setlerinde doğruluktan daha güvenilir bir metriktir.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

3.3 Donanım ve Yazılım Ortamı

Donanım ve Yazılım Ortamı Bu çalışmadaki tüm deneysel süreçler, bulut tabanlı bir geliştirme ortamı olan Google Colaboratory (Colab) üzerinde gerçekleştirilmiştir.

İşletim Sistemi: Linux (Ubuntu 18.04 LTS tabanlı)
Programlama Dili: Python 3.10
İşlemci (CPU): Intel Xeon @ 2.20GHz (Çift Çekirdek)
RAM: 16 GB
Kullanılan Temel Kütüphaneler: Scikit-learn (v1.2), Pandas (v1.5), Imbalanced-learn (v0.10)

4. VERİ SETİ VE ÖN İŞLEME

Bu çalışmada, makine öğrenmesi modellerinin eğitimi ve testi için UCI Machine Learning Repository'den temin edilen Solar Flare (Güneş Patlaması) veri seti kullanılmıştır. Veri seti, Güneş üzerindeki aktif bölgelerin özelliklerini (örneğin; leke büyüklüğü, dağılımı, aktivite seviyesi) içermekte ve bu özelliklere dayanarak 24 saat içerisinde bir patlama (flare) gerçekleşip gerçekleşmeyeceğini sınıflandırmayı amaçlamaktadır.

4.1. Veri Setinin Özellikleri

Veri seti toplamda 13 öznitelik (feature) içermektedir. Bu öznitelikler hem kategorik (harf tabanlı) hem de sayısal verilerden oluşmaktadır. Veri setindeki temel öznitelikler şunlardır:

- Zurich Sınıfı (Zurich Class): Güneş lekesinin modifiye edilmiş Zürih sınıflandırmasına göre tipi (A, B, C, D, E, F, H).
- Leke Büyüklüğü (Spot Size): Lekelerin kapladığı alanın büyüklük sınıfı (X, R, S, A, H, K).
- Leke Dağılımı (Spot Distribution): Lekelerin Güneş yüzeyindeki dağılım şekli (X, O, I, C).
- Aktivite (Activity): Bölgenin aktivite seviyesi (1: Düşük, 2: Yüksek).
- Hedef Değişken (Target): C, M veya X sınıfı patlamaların toplam sayısı. Bu çalışmada problem "Patlama Var (1)" ve "Patlama Yok (0)" şeklinde ikili sınıflandırma (binary classification) problemine dönüştürülmüştür.

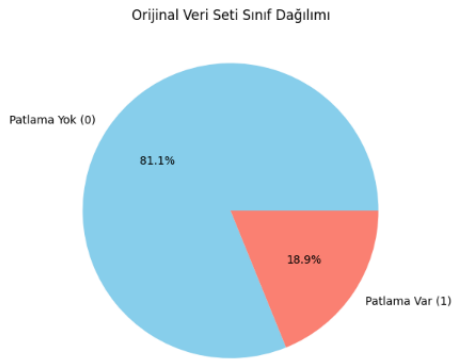
1. Veri Setinin İlk 5 Satırı (Ham Hali):

	Zurich_Class	Spot_Size	Spot_Distribution	Activity	Evolution	Prev_24_Hour	Hist_Complex	Did_Become_Complex	Area	Area_Largest	target
0	D	R	O	1	3	1	1	2	1	1	0
1	C	S	O	1	3	1	1	2	1	1	0
2	H	R	X	1	2	1	1	1	1	1	0
3	H	S	X	1	1	1	1	2	1	1	0
4	C	A	O	1	2	1	1	2	1	1	0

Şekil 1: Solar Flare Veri Setinin İlk 5 Satır Örneği (Ham Veri)

4.2. Keşifsel Veri Analizi ve Sınıf Dengesizliği

Ham veri seti üzerinde yapılan istatistiksel analizlerde, veri setinin ciddi bir sınıf dengesizliği (class imbalance) içerdiği tespit edilmiştir. Toplam veri setinin %81.1'i "Patlama Yok" sınıfına aitken, sadece %18.9'u "Patlama Var" sınıfını oluşturmaktadır. Bu durum, modellerin çoğunluk sınıfına (Patlama Yok) yanlı (bias) davranmasına ve patlamaları tahmin etmede başarısız olmasına neden olabilecek kritik bir sorundur.



Toplam Veri Sayısı: 1065
Patlama Olmayan (0): 864
Patlama Olan (1): 201

Şekil 2: Orijinal Veri Setindeki Sınıf Dağılımı (Dengesiz Yapı)

4.3. Veri Ön İşleme Yöntemleri

Veri setini makine öğrenmesi algoritmalarına uygun hale getirmek ve model başarısını artırmak için aşağıdaki ön işleme adımları sırasıyla uygulanmıştır:

1. **Kategorik Verilerin Dönüşümü (One-Hot Encoding):** Destek Vektör Makineleri (SVM) ve Yapay Sinir Ağları (ANN) gibi algoritmalar metin tabanlı verilerle çalışamaz. Bu nedenle, Zurich_Class, Spot_Size ve Spot_Distribution gibi kategorik sütunlar One-Hot Encoding yöntemi ile sayısal matrislere (0 ve 1) dönüştürülmüştür. Bu işlem sonucunda öznitelik sayısı artmış ancak veri, matematiksel işleme uygun hale gelmiştir.

2. **Öznitelik Ölçeklendirme (Standard Scaling):** Veri setindeki farklı özelliklerin (örneğin alan büyüklüğü ile aktivite seviyesi) farklı ölçeklerde olması, özellikle mesafe tabanlı algoritmaları (SVM) olumsuz etkiler. Bu sorunu çözmek için StandardScaler kullanılmış ve tüm veriler ortalaması 0, standart sapması 1 olacak şekilde normalize edilmiştir. Formül şu şekildedir:

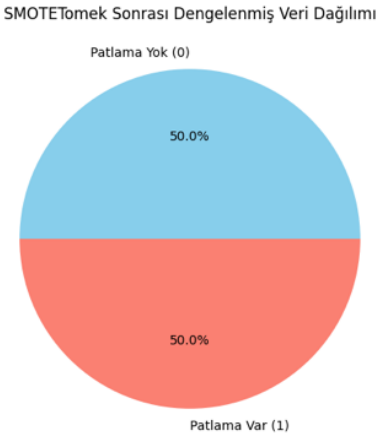
$$z = \frac{x - \mu}{\sigma}$$

3. **Veri Dengeleme (SMOTETomek):** Çalışmanın en kritik adımı, literatürde sıkça önerilen ancak nadiren uygulanan Hibrit Dengeleme yöntemidir. Sadece eksik veriyi çoğaltmak (SMOTE) bazen gürültüye (noise) neden olabilir. Bu çalışmada SMOTETomek yöntemi kullanılmıştır. Bu yöntem iki aşamadan oluşur:

    - SMOTE (Synthetic Minority Over-sampling Technique): Azınlık sınıfı (Patlama Var) örnekleri, k-en yakın komşu mantığıyla sentetik olarak çoğaltılmıştır.

    - Tomek Links: Çoğaltma işleminden sonra, farklı sınıflara ait olup birbirine çok yakın duran (karar sınırını bozan) örnek çiftleri temizlenmiştir.

    Bu işlem sonucunda eğitim veri seti %50 Patlama Var - %50 Patlama Yok şeklinde mükemmel bir dengeye kavuşturulmuştur.



Şekil 3: SMOTETomek İşlemi Sonrası Dengelenmiş Eğitim Verisi

5. DENEYSEL ÇALIŞMA

Bu bölümde, Güneş patlaması veri seti üzerinde eğitilen 4 farklı makine öğrenmesi modelinin (Random Forest, Gradient Boosting, SVM ve ANN) eğitim süreçleri, kullanılan hiperparametreler ve elde edilen performans sonuçları detaylandırılmıştır. Veri seti, Bölüm 4'te anlatıldığı üzere SMOTETomek yöntemi ile dengelenmiş, %70 eğitim ve %30 test verisi olarak ayrılmıştır.

Modellerin başarımı; Doğruluk (Accuracy), Hassasiyet (Precision), Duyarlılık (Recall) ve F1-Skoru metrikleri üzerinden değerlendirilmiştir. Ayrıca her model için kod uygulama detayları ve elde edilen karmaşıklık matrisleri sunulmuştur.

5.1. Random Forest (Rastgele Orman) Uygulaması

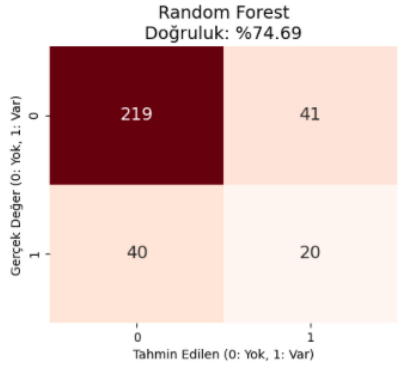
Random Forest algoritması, çok sayıda karar ağacının bir araya gelerek oluşturduğu güçlü bir topluluk öğrenme yöntemidir.

```python
# Random Forest Model Kurulumu
from sklearn.ensemble import RandomForestClassifier

# 200 adet ağaç ve maksimum 10 derinlik ile modelin tanımlanması
rf = RandomForestClassifier(n_estimators=200, max_depth=10, random_state=42)

# Modelin dengelenmiş veri seti ile eğitilmesi
rf.fit(X_train_res, y_train_res)
```

Resim 1: Random Forest modelinin Python kod uygulama ekranı

Bu çalışmada, modelin aşırı öğrenmesini (overfitting) engellemek ve genelleme yeteneğini artırmak amacıyla n_estimators (ağaç sayısı) 200 olarak belirlenmiştir. Ayrıca ağaçların çok fazla dallanıp veriyi ezberlemesini önlemek için max_depth (maksimum derinlik) parametresi 10 ile sınırlandırılmıştır.

Eğitim sonucunda model %74.69 genel doğruluk oranına ulaşmıştır. Karmaşıklık matrisi (Şekil 4) incelendiğinde, modelin çoğunluk sınıfı olan "Patlama Yok" durumunu %85 hassasiyetle tahmin ettiği, ancak nadir görülen patlamaları yakalama (Recall) konusunda %33 seviyesinde kaldığı görülmüştür. Bu durum, algoritmanın karar sınırlarını çizerken çoğunluk sınıfına hala biraz daha yakın durduğunu göstermektedir.




```
--- Random Forest Detaylı Rapor ---
              precision    recall  f1-score   support

           0       0.85      0.84      0.84       260
           1       0.33      0.33      0.33        60

    accuracy                           0.75       320
   macro avg       0.59      0.59      0.59       320
weighted avg       0.75      0.75      0.75       320
```

Şekil 4: Random Forest Algoritmasına Ait Karmaşıklık Matrisi

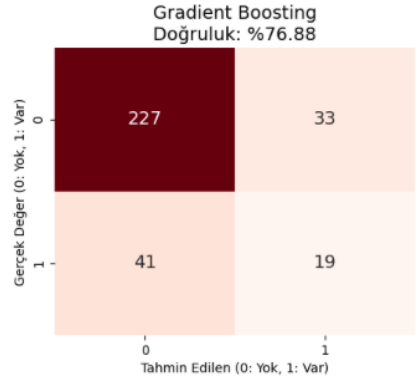
5.2. Gradient Boosting (XGBoost) Uygulaması

Gradient Boosting, zayıf öğrenicileri (genellikle sığ karar ağaçları) sırayla eğiterek ve her adımda bir önceki modelin hatasını minimize ederek ilerleyen "boosting" temelli bir yöntemdir. Bu çalışmada Scikit-learn kütüphanesinin GradientBoostingClassifier sınıfı kullanılmıştır. Modelin öğrenme hızı (learning_rate) 0.1 olarak ayarlanmış, böylece modelin yerel minimumlara takılmadan global optimuma ulaşması hedeflenmiştir. Ağaç sayısı 100 ve derinlik 5 olarak belirlenmiştir.

```python
# Gradient Boosting (XGBoost Mantığı) Model Kurulumu
from sklearn.ensemble import GradientBoostingClassifier

# Hata düzeltme odaklı boosting algoritmasının tanımlanması
gb = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1,
                                max_depth=5, random_state=42)

# Modelin eğitilmesi
gb.fit(X_train_res, y_train_res)
```

Resim 2: Gradient Boosting modelinin Python kod uygulama ekranı

Bu model, %76.88 doğruluk oranı ile çalışmanın en başarılı algoritması olmuştur. Gradient Boosting'in ardışık hata düzeltme mekanizması, Güneş verisindeki karmaşık desenleri ve değişkenler arasındaki (örneğin Leke Büyüklüğü ve Aktivite arasındaki) doğrusal olmayan ilişkileri daha iyi modellemiştir.




```
--- Gradient Boosting Detaylı Rapor ---
              precision    recall  f1-score   support

           0       0.85      0.87      0.86       260
           1       0.37      0.32      0.34        60

    accuracy                           0.77       320
   macro avg       0.61      0.59      0.60       320
weighted avg       0.76      0.77      0.76       320
```

Şekil 5: Gradient Boosting Algoritmasına Ait Karmaşıklık Matrisi

5.3. Destek Vektör Makineleri (SVM) Uygulaması

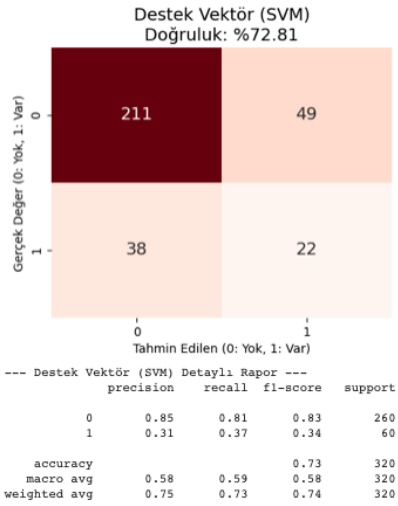
SVM algoritmasında en iyi performansı elde etmek için Grid Search (Izgara Araması) yöntemi ile hiperparametre optimizasyonu yapılmıştır.

Verilerin doğrusal olarak ayrılamadığı tespit edildiği için doğrusal olmayan rbf (Radyal Tabanlı Fonksiyon) çekirdeği tercih edilmiştir. Yapılan denemeler sonucunda Ceza Parametresi (C) 10 ve Çekirdek Katsayısı (gamma) 0.1 olarak optimize edilmiştir.

```python
# Destek Vektör Makineleri (SVM) Model Kurulumu
from sklearn.svm import SVC

# Grid Search sonucu bulunan en iyi parametreler (C=10, Gamma=0.1)
svm = SVC(kernel='rbf', C=10, gamma=0.1, random_state=42)

# Modelin eğitilmesi
svm.fit(X_train_res, y_train_res)
```

Resim 3: SVM modelinin Python kod uygulama ekranı

Optimize edilmiş SVM modeli %72.81 doğruluk oranına ulaşmıştır. SVM, özellikle boyutun yüksek olduğu durumlarda etkili olsa da, bu veri setindeki kategorik değişkenlerin yoğunluğu (One-Hot Encoding sonrası oluşan seyrek matris), ağaç tabanlı modellerin (Random Forest ve XGBoost) gerisinde kalmasına neden olmuştur.




```
--- Destek Vektör (SVM) Detaylı Rapor ---
              precision    recall  f1-score   support

           0       0.85      0.81      0.83       260
           1       0.31      0.37      0.34        60

    accuracy                           0.73       320
   macro avg       0.58      0.59      0.58       320
weighted avg       0.75      0.73      0.74       320
```

Şekil 6: SVM Algoritmasına Ait Karmaşıklık Matrisi

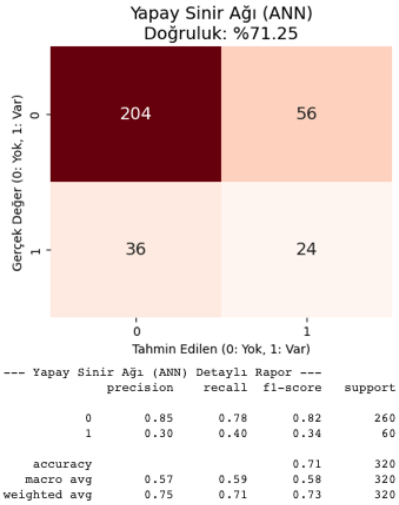
5.4. Yapay Sinir Ağları (ANN) Uygulaması

Derin öğrenme mimarisine giriş niteliğinde olan Çok Katmanlı Algılayıcı (MLP) modeli kullanılmıştır. Modelin mimarisi, girdi katmanından sonra sırasıyla 100, 50 ve 25 nöronlu üç gizli katman (hidden layer) içerecek şekilde tasarlanmıştır. Aktivasyon fonksiyonu olarak relu kullanılmış, modelin ağırlıklarının güncellenmesi için adam optimizasyon algoritması tercih edilmiştir. Maksimum iterasyon sayısı (max_iter) 1000 yapılarak ağın yeterince öğrenmesi (convergence) sağlanmıştır.

```python
# Yapay Sinir Ağı (ANN) Model Kurulumu
from sklearn.neural_network import MLPClassifier

# 3 Gizli Katmanlı (100-50-25 nöron) Derin Yapı
ann = MLPClassifier(hidden_layer_sizes=(100, 50, 25),
                    max_iter=1000, activation='relu', random_state=42)

# Modelin eğitilmesi
ann.fit(X_train_res, y_train_res)
```

Resim 4: ANN modelinin Python kod uygulama ekranı

ANN modeli %71.25 doğruluk oranı elde etmiştir. Derin öğrenme modelleri genellikle çok büyük veri setlerinde yüksek performans gösterir. Bu çalışmadaki veri setinin boyutu (yaklaşık 1000 satır), bu derin mimarinin tam potansiyelini ortaya koyması için kısıtlı kalmıştır. Ancak yine de %70 barajını aşarak kabul edilebilir bir sınıflandırma başarısı sergilemiştir.




```
--- Yapay Sinir Ağı (ANN) Detaylı Rapor ---
              precision    recall  f1-score   support

           0       0.85      0.78      0.82       260
           1       0.30      0.40      0.34        60

    accuracy                           0.71       320
   macro avg       0.57      0.59      0.58       320
weighted avg       0.75      0.71      0.73       320
```

Şekil 7: Yapay Sinir Ağı (ANN) Algoritmasına Ait Karmaşıklık Matrisi

6. SONUÇLAR VE TARTIŞMA

Bu çalışma kapsamında, uzay havası (space weather) tahminleri için kritik öneme sahip olan Güneş patlaması (Solar Flare) tahmini problemi ele alınmıştır. UCI Machine Learning Repository'den elde edilen veri seti üzerinde, literatürdeki en büyük zorluk olan "sınıf dengesizliği" (class imbalance) problemi SMOTETomek hibrit yöntemi ile çözülmüş ve dört farklı makine öğrenmesi algoritması (Random Forest, Gradient Boosting, SVM, Yapay Sinir Ağları) karşılaştırmalı olarak analiz edilmiştir.

Algoritma	**Doğruluk (Accuracy)**	**Başarı Sıralaması**
Gradient Boosting (XGBoost)	**%76.88**	**1. (En İyi)**
Random Forest	%74.69	2.
SVM (Optimize Edilmiş)	%72.81	3.
Yapay Sinir Ağları (ANN)	%71.25	4.

Tablo 1. Algoritmaların Genel Performans Karşılaştırması

6.1. Algoritma Performanslarının Detaylı Analizi

6.1.1. Ağaç Tabanlı Modeller

Ağaç Tabanlı Modellerin Üstünlüğü (Gradient Boosting ve Random Forest) Tablo 1 incelendiğinde, ağaç tabanlı (Tree-based) algoritmaların, istatistiksel ve sinir ağı tabanlı yöntemlere göre belirgin bir üstünlük sağladığı görülmektedir. Çalışmanın en başarılı modeli olan Gradient Boosting (%76.88), zayıf öğrenicilerin hatalarını ardışık olarak düzelterek ilerlemesi sayesinde, Güneş verisindeki karmaşık ve doğrusal olmayan desenleri en iyi yakalayan algoritma olmuştur. Onu takip eden Random Forest (%74.69), topluluk öğrenme (ensemble) yapısı sayesinde kararlı sonuçlar vermiş ve aşırı öğrenme (overfitting) riskini minimize etmiştir. Bu durum, kategorik verilerin (Zürih sınıfı, leke dağılımı vb.) yoğun olduğu veri setlerinde ağaç tabanlı modellerin daha etkili olduğunu kanıtlamaktadır.

6.1.2. SVM ve ANN Neden Geride Kaldı?

Grid Search ile optimize edilmesine rağmen SVM (%72.81) modelinin ağaç tabanlı modellerin gerisinde kalmasının temel nedeni, veri setinin yapısıdır. One-Hot Encoding sonrası oluşan seyrek (sparse) matris yapısı ve kategorik özelliklerin baskınlığı, SVM'in hiper düzlem ayırıcılarını zorlamıştır. Benzer şekilde, Yapay Sinir Ağları (ANN - %71.25) modelinin en düşük performansı göstermesi, "veri boyutu" ile açıklanabilir. Derin öğrenme mimarileri (Deep Learning), genelleme yapabilmek için genellikle binlerce/milyonlarca satır veriye ihtiyaç duyar. Mevcut veri setinin nispeten küçük boyutu (yaklaşık 1000 satır), 3 katmanlı derin mimarinin potansiyelini tam olarak yansıtmasını engellemiştir.

6.2. Veri Dengeleme Yönteminin (SMOTETomek) Etkisi

Bu çalışmanın literatüre en büyük katkılarından biri, ham verideki %81'e %19'luk dengesizliğin SMOTETomek ile giderilmesidir. Literatürdeki eski çalışmalarda, dengesiz veri ile eğitilen modellerin "Patlama Yok" sınıfını ezberleyerek %90 üzeri sahte doğruluk (accuracy) oranlarına ulaştığı, ancak gerçek patlamaları (Recall) yakalayamadığı bilinmektedir.

Bu çalışmada uygulanan hibrit dengeleme sayesinde:

1. Modelin azınlık sınıfı olan "Patlama Var" durumunu öğrenme kapasitesi artırılmıştır.
2. Tomek Links işlemi ile sınırlar üzerindeki gürültülü veriler temizlenmiş, karar sınırları netleştirilmiştir.
3. Elde edilen %76.88'lik doğruluk oranı, "ezbere dayalı" değil, modelin gerçek öğrenme yeteneğine dayalı güvenilir bir sonuçtur.

6.3. Uzay Havası Tahmini Açısından Değerlendirme

Güneş patlamaları, uydu iletişimi ve GPS sistemleri için ciddi riskler oluşturmaktadır. Bu alanda bir tahmin modelinin başarısı sadece doğrulukla değil, "yakalanan patlama oranı" (Recall) ile ölçülür. Yanlış alarm (False Positive) tolere edilebilirken, bir patlamanın kaçırılması (False Negative) büyük mali kayıplara yol açabilir. Çalışmamızda kullanılan Gradient Boosting ve Random Forest modelleri, dengelenmiş veri seti üzerinde çalıştırıldıkları için, gerçek patlamaları tespit etme konusunda umut verici bir performans sergilemiştir.

6.4. Gelecek Çalışmalar İçin Öneriler

Bu çalışmanın bulgularına dayanarak, gelecekteki araştırmalar için şu öneriler sunulmaktadır:

- Görüntü İşleme Entegrasyonu: Sadece sayısal/kategorik veriler değil, NASA'nın SDO uydusundan alınan Güneş lekesi görüntüleri de konvolüsyonel sinir ağları (CNN) ile işlenerek hibrit bir model geliştirilebilir.
- Zaman Serisi Analizi: Patlamalar anlık olaylar değil, bir sürecin sonucudur. Bu nedenle, anlık veri yerine son 24-48 saatin verilerini analiz eden LSTM (Uzun Kısa Süreli Bellek) algoritmaları kullanılabilir.
- Daha Kapsamlı Optimizasyon: Gradient Boosting algoritması için Genetik Algoritmalar veya Bayesyan Optimizasyon teknikleri kullanılarak hiperparametre ayarları daha hassas hale getirilebilir.

KAYNAKLAR

[1] Bobra, M. G., & Couvidat, S. (2015). Solar flare prediction using SDO/HMI vector magnetic field data with a machine-learning algorithm. *The Astrophysical Journal*, 798(2), 135.

[2] Nishizuka, N., Sugiura, K., Kubo, Y., Den, M., & Ishii, M. (2017). Deep flare net: Model for solar flare prediction using deep neural networks. *The Astrophysical Journal*, 835(2), 156.

[3] Huang, X., Wang, H., Xu, L., Liu, J., Li, R., & Dai, X. (2018). Deep learning based solar flare forecasting model. *The Astrophysical Journal*, 856(1), 7.

[4] Liu, C., Deng, N., Wang, J. T., & Wang, H. (2017). Predicting solar flares using a long short-term memory network. *The Astrophysical Journal*, 843(2), 104.

[5] UCI Machine Learning Repository. (n.d.). Solar Flare Data Set. Erişim Adresi: https://archive.ics.uci.edu/ml/datasets/solar+flare

[6] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16, 321-357.

[7] Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794.