T.C.
TOKAT GAZİOSMANPAŞA ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
ZOOTEKNİ ANABİLİM DALI
BİYOMETRİ VE GENETİK YÜKSEK LİSANS PROGRAMI

AĞIRLIK TAHMİNİ İÇİN BAZI FARKLI MAKİNE ÖĞRENMESİ ALGORİTMALARININ KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ

Ahmet Sinan GÜLER

Danışman: Doç. Dr. Yalçın TAHTALI

TOKAT- 2024

ETİK SÖZLEŞME

Tokat Gaziosmanpaşa Üniversitesi Lisansüstü Eğitim Enstitüsü tez yazım kılavuzuna göre, Doç. Dr. Yalçın Tahtalı danışmanlığında hazırlamış olduğum "Ağırlık Tahmini İçin Bazı Farklı Makine Öğrenmesi Algoritmalarının Karşılaştırılması" adlı Yüksek Lisans tezinin bilimsel etik değerlere ve kurallara uygun, özgün bir çalışma olduğunu, aksinin tespit edilmesi halinde her türlü yasal yaptırımı kabul edeceğimi beyan ederim.

…/…/2024

Ahmet Sinan Güler

TEŞEKKÜR

Çalışmamın tüm aşamalarında bilgi ve deneyimlerini benden esirgemeyen danışman hocam Sayın Doç. Dr. Yalçın TAHTALI'ya, araştırmalarımın başından beri desteğini benden esirgemeyen Sayın Dr. Öğr. Üyesi Lütfi BAYYURT'a, çalışmamı değerlendiren jüri üyesi Sayın Doç. Dr. Samet Hasan ABACI'ya, hayatım boyunca her koşulda desteklerini gördüğüm değerli aileme teşekkürlerimi sunarım.

Ahmet Sinan GÜLER

ÖZET

AĞIRLIK TAHMİNİ İÇİN BAZI FARKLI MAKİNE ÖĞRENMESİ ALGORİTMALARININ KARŞILAŞTIRILMASI

Güler, Ahmet Sinan
Yüksek Lisans, Tokat Gaziosmanpaşa Üniversitesi, Lisansüstü Eğitim Enstitüsü Zootekni Anabilim Dalı
Tez Danışmanı: Doç. Dr. Yalçın Tahtalı
Ağustos 2024, x + 71 sayfa

Bu çalışmada, Romanov kuzularında ağırlık tahmini için Random Forest (RF), Extreme Gradient Boost (XGBoost), LightGBM, Gradient Boosting Machine (GBM) ve Support Vector Machine (SVM) makine öğrenmesi algoritmalarının tahminleme performanslarının karşılaştırılması amaçlanmıştır. Bu amaçla Tokat ili Niksar ilçesinde yetiştirilen, 285 Romanov koyunundan doğan 50 Romanov kuzusunun büyüme dönemine ilişkin Cidago yüksekliği, Sağrı yüksekliği, Vücut uzunluğu, Göğüs derinliği, Göğüs çevresi, Kürekler arası göğüs genişliği ve canlı ağırlıkları kullanılmıştır. Yöntemlerin tahmin performanslarını karşılaştırmak için, Hata Kareler Ortalaması (HKO), Hata Kareler ortalamasının karekökü (HKOK) ve Belirtme Katsayısı (R^2) değerleri kullanılmıştır. Çalışma sonucunda Romanov kuzularının canlı ağırlık tahmininde Gradient Boosting Machine (GBM) algoritması, en düşük HKO (0.72801), HKOK (0.99534) ve en yüksek R^2 (0.98022) değerleri ile en iyi performansa ve doğruluğa sahip olduğu belirlenmiştir. Ayrıca, Random Forest (RF), Extreme Gradient Boost (XGBoost), LightGBM ve Support Vector Machine (SVM) algoritmaları da yüksek performans göstermiş ve canlı ağırlık tahmininde etkili sonuçlar sunmuştur.

Anahtar Kelimeler: Makina Öğrenmesi, XGBoost, LightGBM, GBM, Random Forest

ABSTRACT

COMPARISON OF DIFFERENT MACHINE LEARNING ALGORITHMS FOR WEIGHT PREDICTION

Güler, Ahmet Sinan
Master's Thesis, Tokat Gaziosmanpaşa University, Graduate Education Institute, Department of Animal Science
Advisör: Assoc. Prof. Dr. Yalçın Tahtalı
August 2024, x + 71 pages

In this study, it was aimed to compare the prediction performances of Random Forest (RF), Extreme Gradient Boost (XGBoost), LightGBM, Gradient Boosting Machine (GBM) and Support Vector Machine (SVM) machine learning algorithms for weight estimation in Romanov lambs. For this purpose, Cidago height, Rump height, Body length, Chest depth, Chest depth, Chest girth, Chest girth between the shoulder blades, Chest width between the shoulder blades and live weights of 50 Romanov lambs born from 285 Romanov ewes reared in Niksar district of Tokat province were used. Mean Square Error (MSE), Root Mean Squared Error (RMSE) and coefficient of determination (R^2) values were used to compare the prediction performance of the methods. As a result of the study, it was determined that Gradient Boosting Machine (GBM) algorithm had the best performance and accuracy with the lowest MSE (0.72801), RMSE (0.99534) and the highest R^2 (0.98022) values in the live weight estimation of Romanov lambs. Additionally, Random Forest (RF), Extreme Gradient Boost (XGBoost), LightGBM, and Support Vector Machine (SVM) algorithms also demonstrated high performance and provided effective results in predicting live weight.

Keywords: Machine Learning, XGBoost, LightGBM, GBM, Random Forest

İÇİNDEKİLER

Sayfa

KISALTMA VE SİMGELER

Simgeler	Açıklamalar
y_i	Gerçek değer (hedef değişken).
$\hat{y}_i$	Modelin tahmin ettiği değer.
$\hat{y}_i^{(m)}$	m-inci iterasyondaki tahmin değeri.
$r_i^{(m)}$	Residüel hata
γ_m	Öğrenme oranı
$h_m(x_i)$	m-inci iterasyonda kullanılan ağacın, i-inci örnek için yaptığı tahmin
$\ell(y_i, \hat{y}_i)$	Kayıp fonksiyonu
$\Omega(h_m)$	Düzenlileştirme terimi
f(x)	Tahmin fonksiyonu.
w	Ağırlık vektörü.
x	Girdi vektörü.
b	Sabit terim (bias).
y_i	i-inci veri noktasının gerçek etiketi.
α_i	Lagrange çarpanları.
K(x, x_i)	Çekirdek fonksiyonu
∑	Toplama işlemi.

Kısaltmalar	Açıklamalar
RF	Random Forest
SVM	Support Vector Machine
XGBoost	Extreme Gradient Boost
GBM	Gradient Boosting Machine
LGBM	LightGBM
HKO	Hata Kareler Ortalaması
R^2	Belirtme Katsayısı
R^2adj	Düzeltilmiş Belirtme Katsayısı
HKOK	Hata Kareler Ortalamasının Karekökü
OMH	Ortalama Mutlak Hata
OMYH	Ortalama Mutlak Yüzde Hata
CA	Canlı Ağırlık
CY	Cidago Yüksekliği
SY	Sağrı Yüksekliği
VU	Vücut Uzunluğu
GD	Göğüs Derinliği
GÇ	Göğüs Çevresi
KAGG	Kürekler Arası Göğüs Genişliği

ŞEKİLLER LİSTESİ

Şekiller No Sayfa

ÇİZELGELER LİSTESİ

1. GİRİŞ

Sağlıklı ve düzenli beslenme için insan vücudu, günlük olarak çeşitli besin öğelerine ihtiyaç duyar. Bu bağlamda, hayvansal ürünlerin beslenmedeki rolü hayati öneme sahiptir. Gelişmiş ülkelerde, bireylerin yaşam kalitesi ve refahı değerlendirilirken kişi başına düşen et tüketimi miktarı önemli bir gösterge olarak kullanılmaktadır. Ülkemizdeki kırmızı et tüketiminde, öncelikle sığır eti tercih edilmekte, ardından koyun ve keçi eti tüketimi gelmektedir (Karaoğlu ve Emsen, 2000).

Küçükbaş yetiştiriciliği sektörü, kuzu üretiminin ekonomik kazanç üzerindeki büyük payı nedeniyle önemli bir yer tutmaktadır. Bu sebeple, her bir koyundan elde edilen kuzu sayısı ve bu ırkların büyüme performansları, araştırmalarda temel ölçütler olarak kabul edilmektedir (Yılmaz ve Altınel, 2003; Kaymakçı ve Sönmez, 1996). Büyüme sürecinde, hayvanların sadece ağırlıkları değil, aynı zamanda vücut yapıları da değişim gösterir. Bu süreçte, ağırlık ve vücut yapısı arasındaki ilişkiyi anlamak, büyüme dönemindeki gelişimin daha net bir şekilde kavranmasını sağlar. Böylece, hayvanların büyüme, gelişme ve genel morfolojik özellikleri hakkında değerli bilgiler elde edilebilir (Altınel ve ark., 1998; Efe, 1990).

Besleme sürecinde, hayvanların büyüme ve gelişiminin izlenmesi, verimliliği artırmak ve mali kayıpları önlemek açısından kritik bir öneme sahiptir. Alınan vücut ölçümleri, en verimli hayvanların seçilmesi, besleme ve bakım sorunlarının tespiti, yemlik, otlatma alanları ve suluk gibi ekipmanların ihtiyaçlarının doğru bir şekilde belirlenmesi için temel bir araçtır (Ertuğrul, 1996). Optimal bir kuzu besi programı, adaptasyon sürecinin ardından kuzuların sütten kesilmesiyle başlar. Bununla birlikte, yem ve et fiyatlarındaki dalgalanmalar ve kötü işletme yönetimi, besi programının hem başlangıcını hem de sonunu etkileyebilir. Geleneksel koyun yetiştiricileri, tarımsal faaliyetlerin tamamlanması amacıyla kuzu besisine başlamayı erteleyebilirler. Bu tür durumlarda, maliyeti minimize eden üretim sistemlerine yönelmek, koyun yetiştiriciliğini daha karlı ve sürdürülebilir hale getirilebilir. Bu sebeple, bazen besleme sürecinin biraz daha geç başlaması uygun olabilir (Zimmermann ve ark., 2019).

Hayvanların büyüme ve gelişimini etkileyen birçok faktörü içeren besleme sürecinde, elde edilen verilerin doğru bir şekilde değerlendirilmesi başarılı bir yetiştiricilik için gereklidir. Bu noktada, modern teknolojilerin ve veri analiz yöntemlerinin kullanımı giderek önem kazanmaktadır. Hayvanların ağırlık, boy, vücut ölçüleri gibi fiziksel özelliklerinin düzenli olarak takip edilmesi, yalnızca verimliliği artırmakla kalmaz, aynı zamanda gelecekteki performanslarının tahmin edilmesini de mümkün kılabilir. İşte bu tür tahminlerde ve karar destek sistemlerinde, geleneksel yöntemlerin ötesine geçmek amacıyla makine öğrenmesi algoritmalarının kullanımı ön plana çıkmalıdır. Verilerin etkin bir şekilde işlenmesi ve anlamlı bilgiye dönüştürülmesi, besleme ve büyüme süreçlerinin daha iyi yönetilmesini sağlar. Bu bağlamda, makine öğrenmesi algoritmaları, hayvan yetiştiriciliğinde elde edilen verilerin analiz edilmesinde kritik bir rol oynar ve veriye dayalı kararların alınmasında önemli bir araç olarak kullanılabilir.

Makine öğrenmesi, veriden anlamlı bilgiler çıkarmak için kullanılan bir yöntemdir ve bu nedenle birçok bilimsel araştırma ve endüstriyel uygulama için kritik bir rol oynamaktadır (Géron, A., 2019). Makine öğrenmesi algoritmaları, büyük veri kümelerinden öğrenerek gelecekteki olayları tahmin edebilir ve karar verme süreçlerini iyileştirebilir. Özellikle hayvancılık ve tarım gibi alanlarda, hayvanların büyüme ve gelişim süreçlerini tahmin etmek, verimliliği artırmak ve kaynak kullanımını optimize etmek açısından makine öğrenmesi algoritmalarının kullanımı giderek yaygınlaşmaktadır. Bu algoritmalar, veriyi anlamlandırma ve doğru tahminlerde bulunma yetenekleri sayesinde, araştırmacılar ve uygulayıcılar için güçlü bir araçtır. Bu çalışmada kullanılan Random Forest (RF), Extreme Gradient Boost (XGBoost), LightGBM, Gradient Boosting Machine (GBM) ve Support Vector Machine (SVM) gibi algoritmalar, karmaşık verilerden en iyi sonuçları elde etmek için geliştirilmiş güçlü araçlar arasında yer almaktadır.

Bu çalışmada, Romanov kuzularının besi başlangıcındaki vücut özelliklerine ait ölçümler kullanarak, besi sonu canlı ağırlığın tahmin edilmesi için RF, XGBoost, LightGBM, GBM ve SVM veri madenciliği algoritmaları kullanılmış ve bu algoritmaların tahmin başarısı karşılaştırılmıştır.

2. LİTERATÜR KAYNAKLARI

Şengonca ve Gücük (1991), Yerli Merinos koyunlarının vücut ölçülerinden canlı ağırlık tahminini amaçlayan çalışmalarında, cidago yüksekliği, vücut uzunluğu, göğüs derinliği, göğüs genişliği, ön göğüs genişliği, göğüs çevresi ve ön incik çevresi gibi yedi farklı vücut ölçüsünün canlı ağırlık ile fenotipik ilişkilerini değerlendirmişlerdir. Araştırma sonucunda, göğüs çevresinin canlı ağırlıkla en yüksek fenotipik ilişki gösteren ölçü olduğu tespit edilmiştir ($r = 0.893$) ve bu ölçümün canlı ağırlık tahmininde güvenilir bir değişken olduğu belirlenmiştir. Ayrıca, tüm ölçümler arasında göğüs çevresinin canlı ağırlığın tahmin edilmesinde en etkili ve doğruluğu yüksek bir gösterge olduğu vurgulanmıştır.

Topal ve Macit (2004), Morkaraman koyunlarının vücut ölçülerinden canlı ağırlık tahmin etmek amacıyla çeşitli doğrusal regresyon modellerini değerlendirmişlerdir. Çalışmada, cidago yüksekliği, göğüs derinliği, göğüs çevresi, vücut uzunluğu, göğüs genişliği ve sağrı genişliği gibi değişkenler kullanılarak basit ve çoklu regresyon modelleri oluşturulmuştur. Bulgular, göğüs çevresi değişkeninin canlı ağırlığı tahmin etmek için en iyi bağımsız değişken olduğunu ve tahmin modellerinde yüksek doğruluk sağladığını ortaya koymuştur. En iyi performansı, göğüs çevresi ve sağrı genişliği değişkenlerinin birlikte kullanıldığı çoklu regresyon modeli göstermiştir ($R^2 = 0.784$, HKO = 9.678).

Koç ve Akman (2007), Siyah-Alaca tosunların farklı dönemlerdeki vücut ölçülerinden canlı ağırlık tahmini yapabilmek için stepwise-regresyon yöntemi ile çeşitli modeller geliştirmişlerdir. Çalışmada, 18 baş Siyah-Alaca tosunun göğüs çevresi, but çevresi, sağrı yüksekliği ve cidago yüksekliği gibi vücut ölçümleri kullanılmıştır. Sonuçlar, göğüs çevresi değişkeninin tek başına canlı ağırlık tahmini için yeterli olduğunu göstermiştir. Buna ek olarak, göğüs çevresi, but çevresi ve sağrı yüksekliğinin birlikte kullanılmasıyla tahmin doğruluğunun arttığı belirlenmiştir. Elde edilen modellerde, göğüs çevresinin canlı ağırlığı tahmin etmede en yüksek doğruluğa sahip olduğu ve bu değişkenin diğer vücut ölçüleri eklenmeden de kullanılabileceği vurgulanmıştır

Huma ve Iqbal (2019), Pakistan'ın Balochistan bölgesinde yetiştirilen Balochi koyunlarının canlı ağırlığını tahmin etmek amacıyla çeşitli makine öğrenimi algoritmalarının performansını incelemişlerdir. Çalışmada, 131 erkek koyundan elde edilen vücut uzunluğu, kalp çevresi, cidago yüksekliği, skrotal çevre, skrotal çap, skrotal uzunluk ve testis uzunluğu bağımsız değişkenler olarak kullanılmıştır. Araştırmada değerlendirilen algoritmalar arasında General Linear Model (GLM), Regression Trees (RT), RF ve SVM yer almaktadır. Model performansı OMH, OMYH, HKOK, gözlemlenen ve tahmin edilen değerler arasındaki korelasyon katsayısı (r) ve R^2 kriterlerini değerlendirilmiştir. Sonuçlar, RF algoritmasının hem eğitim hem de test veri setlerinde en yüksek doğruluk oranı ve en düşük hata değerlerine ulaştığını ortaya koymuştur.

Tırınk ve ark. (2022), Doğu Anadolu'da yaygın olarak yetiştirilen Morkaraman koyunlarının doğum ağırlığını bazı biyometrik ölçümlerden tahmin etmek amacıyla Classification and Regression Trees (CART) veri madenciliği algoritmasını kullanmışlardır. Çalışmada, 44 adet Morkaraman kuzusunun cidago yüksekliği, sağrı yüksekliği, göğüs çevresi ve vücut uzunluğu biyometrik ölçümlerine dayalı veriler değerlendirilmiştir. Doğum ağırlığını tahmin etmek amacıyla, CART algoritmasına 10 katlı çapraz doğrulama yöntemi uygulanmıştır. Sonuçlar, gerçek ve tahmini doğum ağırlığı arasındaki Pearson korelasyon katsayısının 0.935 olduğunu göstermiştir. Model performansını değerlendirmek için RMSE, rRMSE, Standard Deviation Ratio (SDratio), Akaike Information Criterion (AIC), Mean Absolute Deviation (MAD), Relative Absolute Error (RAE), Mean Absolute Percentage Error (MAPE), R^2 ve R^2adj gibi uyum iyiliği kriterleri kullanılmıştır. CART veri madenciliği algoritmalarıyla elde edilen model, R^2 (0.874) ile yüksek olduğu için kuzularda doğum ağırlığının tahmini için önemli olduğunu göstermiştir.

Çakmakçı (2022), Norduz koyunlarının canlı ağırlığını morfolojik ölçümlerden tahmin etmek amacıyla dört farklı makine öğrenme modelini karşılaştırmıştır. Kullanılan modeller arasında SVM, CART, RF ve Model Averaging Neural Networks (MANN) yer almaktadır. Çalışmada, göğüs çevresi, göğüs genişliği, göğüs derinliği, cidago yüksekliği, vücut uzunluğu, sağrı yüksekliği ve sağrı genişliği olmak üzere yedi farklı morfolojik

ölçüm değerlendirilmiştir. Tüm ölçümlerin canlı ağırlık ile pozitif korelasyon gösterdiği belirlenmiştir. Test veri seti ile yapılan tahminlerde, RF modelinin en düşük OMH, HKOK ve OMYH değerlerine sahip olduğu tespit edilmiştir.

Coşkun ve ark. (2022), Siyah Alaca sığırların farklı büyüme ve gelişme dönemlerindeki vücut ölçümlerini kullanarak canlı ağırlık tahmininde veri madenciliği algoritmalarının performanslarını karşılaştırmışlardır. Çalışmada, göğüs çevresi, cidago yüksekliği, vücut uzunluğu , göğüs derinliği ve sağrı yüksekliği gibi vücut ölçümleri kullanılarak, Multiple Linear Regression (MLR) , RF, Decision Tree (DT) ve K-Nearest Neighbors (KNN) algoritmalarının doğrulukları değerlendirilmiştir. Sonuçlar, MLR algoritmasının %93.9 R^2 ile en yüksek doğruluğa sahip olduğunu, RF'nin ise ikinci sırada yer aldığını göstermiştir. Araştırma bulguları, göğüs çevresi ve göğüs derinliği vücut ölçümlerinin tahmin için yeterli olduğunu ortaya koymaktadır.

Dang ve ark. (2022), Güney Kore'de yetiştirilen Hanwoo sığırlarının canlı ağırlığını tahmin etmek amacıyla çeşitli makine öğrenimi algoritmalarının kullanılabilirliğini araştırmışlardır. Çalışmada, Hanwoo sığırlarından elde edilen 33,536 örneğe ait on vücut ölçümü girdi değişkenleri olarak kullanılmıştır. Araştırmada değerlendirilen algoritmalar arasında Multilayer Perceptron (MP), KNN, LightGBM, TabNet ve FT-Transformer yer almaktadır. Yapılan analizler, on vücut ölçümünün canlı ağırlık ile yüksek korelasyon gösterdiğini ve LightGBM algoritmasının diğer algoritmalara kıyasla en iyi performansı sunduğunu ortaya koymuştur. Elde edilen bulgular, LightGBM algoritmasının model karmaşıklığı ve geliştirme süresi açısından üstün olduğunu ve Hanwoo sığırlarının canlı ağırlığını tahmin etmede etkili bir yöntem sunduğunu göstermektedir.

Hamadani ve Ganai (2023), koyun ağırlığının tahmini amacıyla çeşitli yapay zekâ algoritmalarını karşılaştırarak sıralamışlardır. Araştırmada, 11 yıllık koyun çiftliği verileri kullanılmış ve bu veriler PCA ve Feature Selection (FS) teknikleri ile üç farklı veri setine ayrılmıştır: PCA, PCA+FS ve FS çalışmada değerlendirilen 11 makine öğrenimi algoritması arasında, Multivariate Adaptive Regression Splines (MARS) algoritması, en yüksek korelasyon katsayısı ile en iyi performansı göstermiştir. Bunun

yanı sıra, Bayesian Ridge Regression (BRR), Ridge Regression (RR), SVM ve GBM algoritmaları da yüksek performans gösteren diğer yöntemler arasında yer almıştır.

Coşkun ve ark. (2023), Anadolu Merinos kuzularının besi başlangıcındaki vücut özellikleri ile nihai besi canlı ağırlıklarının tahmin edilmesinde veri madenciliği algoritmalarının performanslarını karşılaştırmışlardır. Çalışmada, başlangıçtaki vücut ölçümleri ve canlı ağırlıkları kullanılarak 54 erkek tek doğumlu Anadolu Merinos kuzusu değerlendirilmiştir. Araştırmada kullanılan algoritmalar arasında RF, XGBoost ve Bayesian Regularization Neural Network (BRNN) yer almaktadır. Sonuçlar, XGBoost algoritmasının diğer algoritmalara kıyasla daha yüksek bir doğruluk sağladığını ve nihai besi ağırlığını en iyi şekilde tahmin ettiğini ortaya koymuştur. Çalışmada, XGBoost algoritmasının toplam varyansın en büyük payını vücut uzunluğuna, en düşük payını ise göğüs derinliğine verdiği belirlenmiştir. Besi performansı açısından, XGBoost algoritmasının en düşük HKOK ve en yüksek R^2 değerlerine sahip olduğu tespit edilmiştir.

Setiawan ve ark. (2024), Sığır ağırlığını tahmin etmek için Linear Regression (LR), RF, SVM, KNN, Multi-layer Perceptron (MLP), GBM, LightGBM ve XGBoost gibi algoritmalar kullanılmıştır. Araştırma bulguları, SVM yönteminin en düşük OMH değeri 0.09, OMYH değeri 0.02, HKOK değeri 0.08 ve %97'lik R^2 değeri ile diğer algoritmalar arasında en iyi performansı sergilediğini göstermiştir.

Gomez-Vazquez ve ark. (2024), Güneydoğu Meksika'da yetiştirilen mandaların (Bubalus bubalis) vücut ağırlığını tahmin etmek amacıyla Principal Component Analysis (PCA) destekli GBM ve RF algoritmalarını incelemişlerdir. Çalışmanın ilk aşamasında, veri boyutunu azaltmak ve en bilgilendirici değişkenleri belirlemek için PCA uygulanmıştır. İkinci aşamada ise, PCA ile elde edilen bileşenler kullanılarak GBM ve RF algoritmaları ile iki farklı tahmin modeli geliştirilmiştir. Modellerin performansları R^2, HKOK ve OMH gibi kriterler kullanılarak karşılaştırılmıştır. GBM algoritmasının daha yüksek R^2 değeri ve daha düşük hata oranları ile daha iyi bir tahmin performansı sunduğunu ortaya koymuştur.

3. GENEL BİLGİLER

3.1. Romanov Irkı

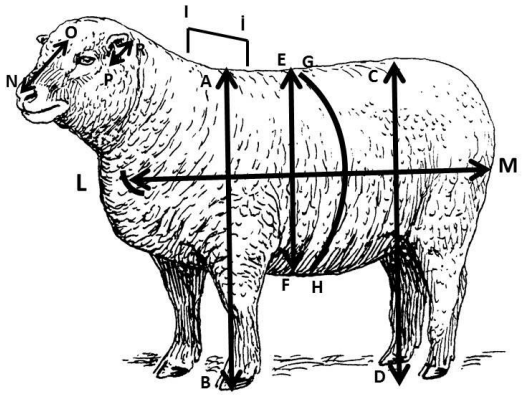
Romanov koyunları, Rusya'nın Volga bölgesinde özellikle Tutayev kesiminin dağlık alanlarında yetiştirilen ve yüksek doğurganlık yeteneğine sahip bir ırk olarak tanımlanmaktadır. Vücut siyah-gri renkte olup, baş, bacaklar ve kuyruk siyah parlak ve kısa kıllarla kaplıdır. Vücut orta büyüklükte, yetişkin canlı ağırlık ortalama 45-48 kg aralığındadır. Süt verimi, ikiz, üçüz ve dördüz doğum oranı oldukça yüksektir (Akçapınar, 2000).

3.1.1. Doğum ağırlığı

Doğum ağırlığı, kuzuların doğum sonrası büyüme hızını etkilemesi bakımından önemli unsurlardan biridir. Kuzuların doğum ağırlığı 3,5-5,5 kg arasında olduğunda normal kabul edilir (Dalton ve ark., 1980).

3.1.2. Vücut ölçüleri

Bu çalışmada, Şekil 3.1'de gösterilmiş olan vucüt ölçüm değerleri kullanılmıştır. Çalışmada kullanılan vucüt ölçülerine ait ölçüm değerleri şu şekildedir; A-B: Cidago yüksekliği, C-D: Sağrı yüksekliği, L-M: Vücut uzunluğu, E-F: Göğüs derinliği, G-H: Göğüs çevresi, I-İ: Kürekler arası göğüs genişliği



Şekil 3.1. Vücut ölçüm yöntemleri (Yaldızbaş, 2016)

Tahtalı ve Yaldızbaş (2020) tarafından elde edilen verilerden çalışma için kullanılan vücut ölçümleri ve tanımları aşağıdaki gibidir:

- Cidago Yüksekliği (CY): Cidagonun en yüksek noktasından yere olan düşey uzaklıktır. Ölçme bastonu ile ölçülmüştür.
- Sağrı Yüksekliği (SY): Tuber coxae'ları birleştiren bölgenin yere olan uzaklığıdır.Ölçme bastonu ile ölçülmüştür.
- Vücut Uzunluğu (VU): Omuz ucundan (articulus humeri) oturak yumrusuna (tuberischii) kadar olan uzaklıktır. Ölçme bastonu ile ölçülmüştür.
- Göğüs Derinliği (GD): Cidago ile göğüs kemiği (sternum) arasındaki düşeyuzaklıktır. Ölçme bastonu ile ölçülmüştür.
- Göğüs Çevresi (GÇ): Cidago ve sternum dan geçen ve göğsü tamamen çevreleyen ölçüdür. Ölçme şeridi ile ölçülmüştür.
- Kürekler Arası Göğüs Genişliği (KAGG): Kürekler arasında bulunan çukurluklar arası uzaklıktır. Ölçme bastonu ile ölçülmüştür.
- Canlı Ağırlık (CA): Tartım hassasiyeti 50 gr olan tartı ile ölçüm yapılmıştır. (Yaldızbaş, 2016)

4. MATERYAL VE YÖNTEM

4.1. Materyal

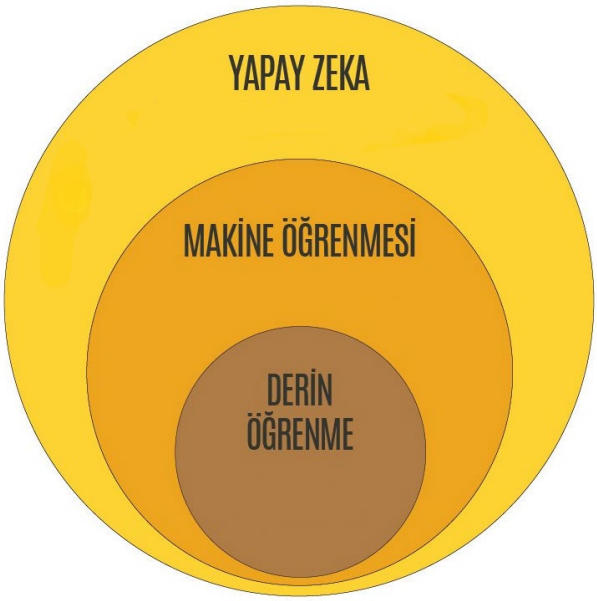
4.1.1. Veri seti

Bu çalışmada, Tahtalı ve Yaldızbaş (2020) tarafından Tokat ili Niksar ilçesindeki 50 Romanov kuzusundan alınan canlı ağırlık ve bazı vücut ölçülerine ait veriler kullanılmıştır. Veriler, kuzuların doğumundan itibaren her 15 günlük periyotlarla 180. güne kadar kaydedilen canlı ağırlık (CA), cidago yüksekliği (CY), sağrı yüksekliği (SY), vücut uzunluğu (VU), göğüs derinliği (GD), göğüs çevresi (GÇ) ve kürekler arası göğüs genişliği (KAGG) ölçümlerinden oluşmakta olup, araştırmacıların izniyle değerlendirilmiştir.

4.1.2. Makine öğrenmesi

Günümüzde, teknolojinin sürekli olarak gelişmesi ve güncellenmesiyle birlikte işlem kapasitesi çok yüksek bilgisayarlar geliştirilmiştir. Bu gelişme, büyük veri setlerinin oluşmasına ve bu verilerin işlenmesi, analiz edilmesi gereksinimini doğurmuştur. Bu gereksinimlerin ortaya çıkması ile makine öğrenmesi ve farklı metodların kullanımının yaygınlaşmasına yol açmıştır (Budak ve Erpolat, 2012).

Makine öğrenmesi, insan öğrenme sürecinden ilham alınarak geliştirilmiş bir teknolojidir. Bu alan, matematiksel ve istatistiksel yöntemlere dayanarak çıkarımlar yapmaktadır ve bu çıkarımları tahminlerde kullanılmaktadır. Şekil 4.1'de gösterildiği üzere yapay zekanın bir alt dalı olan makine öğrenmesi, bu yeteneklerini sürekli geliştirmektedir (Géron, A., 2019).



Şekil 4.1. Makine öğrenmesi alt ve üst dalı

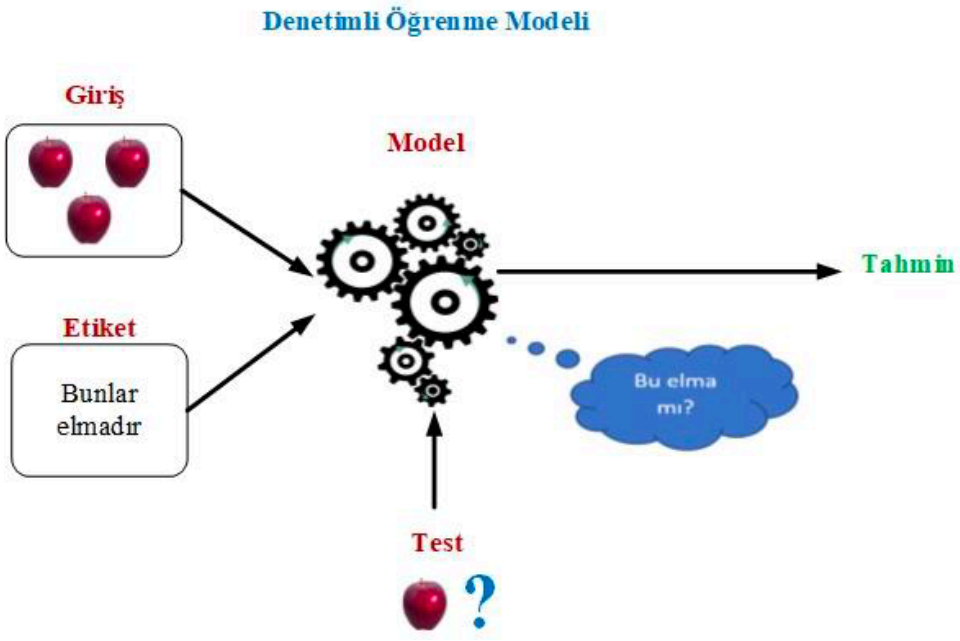
Makine öğrenmesi, geleneksel veri analiz yöntemlerinin yetersiz kaldığı karmaşık ve zorlu sorunların çözümünde kullanılır. Bu yaklaşım, mevcut veri kümelerinden öğrenerek ve bu bilgileri kullanarak modeller geliştirmeye odaklanır (Rebala ve ark., 2019).

Makine öğrenmesi, günümüzde birçok sektörde ve uygulama alanında temel bir rol oynamaktadır. Sağlık hizmetlerinde, hastalıkların teşhisi ve tedavi yöntemlerinin geliştirilmesi, tıp, doğal dil işleme, finansal risk değerlendirmesi, robot sistemlerinin kontrolü, görüntü işleme, siber güvenlik, IoT sistemleri, akıllı şehirler, tarım ve hayvancılık, hava tahmini, reklam sektörü gibi birçok sınıflandırma, kümeleme ve regresyon problemleri uygulamalarında kullanılmaktadır (Géron, A., 2019).

Makine öğrenmesi, öğrenme türlerine göre Denetimli Öğrenme (Supervised Learning), Denetimsiz Öğrenme (Unsupervised Learning) ve Pekiştirmeli Öğrenme (Reinforcement Learning) olmak üzere 3 farklı kategoriye ayrılmaktadır (Géron, A., 2019).

4.1.2.1. Denetimli öğrenme

Denetimli öğrenme, makine öğrenme uygulamalarında sıklıkla tercih edilen bir yöntemdir ve Şekil 4.2'de gösterildiği üzere, giriş ve çıkış verileri arasında açık bir ilişki bulunan durumlar için idealdir. Bu yöntemde, kullanılan verilerin tamamının etiketlenmiş olması şarttır. Eğitim sürecinde, bu etiketlenmiş verilerin büyük bir kısmı modeli eğitmek için, geri kalanı ise test etmek için kullanılır. Eğitimde kullanılan verilerin, taşıdığı etiketlerle örtüşen özelliklere sahip olması beklenir, çünkü oluşturulan denetimli öğrenme modeli, bu etiketlenmiş verileri temel alarak öğrenme işlemini yapar. Model, test verilerini kullanarak tahminlerde veya sınıflandırmalarda bulunur. Eğitim sürecinin sonunda, modelin ürettiği değerlerin, gerçekte üretmesi gereken değerlerle uyumlu olup olmadığı kontrol edilir. Bu sebeplerle, bu tür modellere literatürde 'Denetimli Öğrenme' modeli adı verilir. Denetimli öğrenme modellerinde, genellikle giriş ve çıkış verileri arasında doğrusal bir ilişki bulunmaz. Bu yüzden, eğitim metodolojilerini geliştirerek modelin daha iyi sonuçlar üretmesi amaçlanır (Metlek ve Kayaalp, 2020).



Şekil 4.2. Denetimli öğrenme modeli (Metlek ve Kayaalp, 2020)

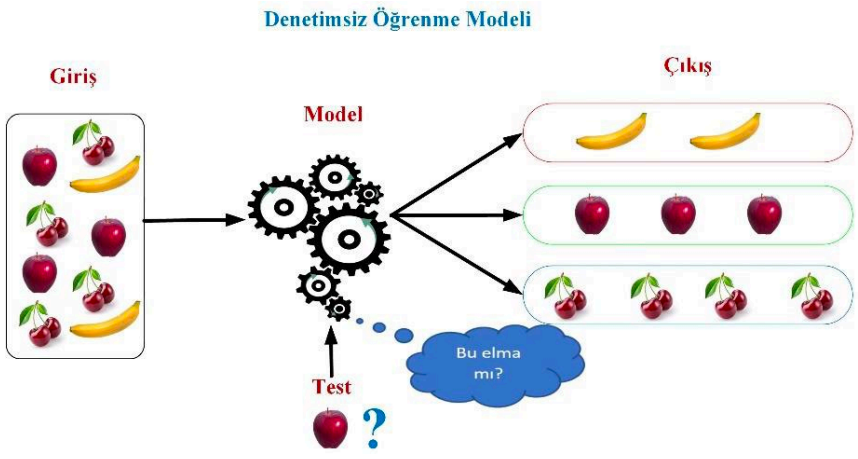
Sistemin öğrenme başarısı, temsil edilen sınıfı doğru şekilde yansıtan örneklerin sayısı ile doğru orantılıdır. Denetimli öğrenmenin ana özellikleri şu şekildedir:

- Kullanılan eğitim setinde hem giriş hem de çıkış verilerinin bulunması gerekmektedir.
- Eğitim verileri, tüm veri seti için genel bir model oluşturmak üzere kullanılır.
- Eğitim görmüş model, daha önce karşılaşmadığı giriş verilerine dayanarak çıkış değerlerini tahmin eder.

Denetimli öğrenme modelleri, esas olarak sınıflandırma ve regresyon alanlarında kullanılır. Sınıflandırma uygulamalarında genellikle SVM, Decision Tree, Naive Bayes ve K-Nearest Neighbors gibi yöntemler tercih edilirken, regresyon uygulamalarında Linear Regression, Principal Component Regression, Random Forest ve Artificial Neural Networks gibi modeller kullanılmaktadır (Metlek ve Kayaalp, 2020).

4.1.2.2. Denetimsiz öğrenme

Denetimsiz öğrenme algoritmalarının temel prensibi, etiketsiz veriler üzerinde çalışmaktır. Bu tür modeller, elimizdeki verilerin içindeki ilişkileri keşfederek, verileri doğal gruplarına ayırır. Şekil 4.3'de gösterildiği gibi, denetimsiz öğrenme algoritmaları verileri benzerliklerine göre; örneğin benzer bölümlere, yapısal özelliklere, yoğunluklarına ve diğer ortak özelliklerine göre sınıflandırır.



Şekil 4.3. Denetimsiz öğrenme modeli(Metlek ve Kayaalp, 2020)

Denetimsiz öğrenme algoritmaları, Hebbian Öğrenme kuralına dayalıdır. Bu kural, iki nöron birlikte aktif olduğunda aralarındaki bağın güçlendiğini, aktif olmadıklarında ise bu bağın zayıfladığını belirtir. Denetimsiz öğrenmenin temel ilkeleri şunlardır:

- Kullanılan veri setindeki örnekler bilinmektedir, ancak bu örneklerin hangi kümeye ait oldukları bilinmemektedir, bu da büyük veri uygulamalarında denetimsiz öğrenmeyi popüler kılar.
- Denetimsiz öğrenme modellerinin temel amacı, veri setindeki verilerden bir model veya örüntü çıkarmaktır.
- Denetimsiz öğrenme algoritmaları, veri setindeki verileri sınıflara ayırmak için kullanılır; bu süreç 'kümeleme' olarak adlandırılır ve bu süreç sonucunda oluşan sınıflar 'kümeler' olarak isimlendirilir.

Denetimsiz öğrenmede sıkça kullanılan algoritmalar ise K-mean Clustering, Hierarchical Clustering, Principal Component Analysis (Metlek ve Kayaalp, 2020).

4.1.2.3 Pekiştirmeli öğrenme

Pekiştirmeli öğrenme, makine öğrenmesinin temel bir metodudur ve deneme yanılma sürecine dayanır. Bu yaklaşımda, eğitim gören sistem çevresiyle etkileşim kurarak, hedeflere ulaşmak için en uygun yolu bulmak üzere algoritmalar geliştirir ve böylece öğrenme sürecini gerçekleştirir (Cruz ve ark., 2015).

4.1.3. Makine öğrenmesi algoritmaları

4.1.3.1 Karar ağaçları(KA)

KA, belirli bir hedef değişkenin, çeşitli açıklayıcı değişkenler doğrultusunda nasıl değiştiğini tahmin etmek için kullanılan etkili yöntemler arasında yer alır. Bu algoritma, veriyi belirli kurallara göre dallandırarak hiyerarşik bir yapı oluşturur ve bu yapı aracılığıyla hedef değişkenin olası değerlerini tahmin eder. Karar ağaçları regresyon tekniğinde, veri setinin farklı özellikleri ve bu özelliklerin belirli aralıklara ayrılması ile

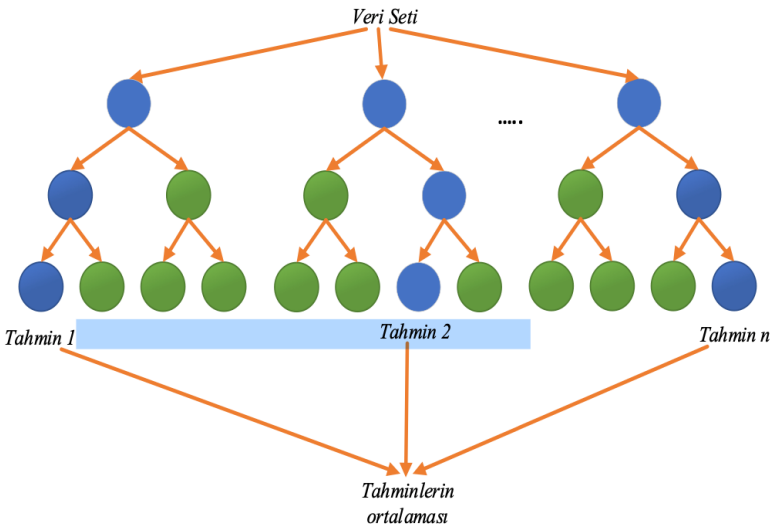
dallar ve yapraklar oluşur; böylece her bir karar aşaması veri setini daha küçük ve anlamlı gruplara ayırarak sonuca ulaşılmasını sağlar (Breiman ve ark., 1984).

Modelin en büyük avantajlarından biri, sürecin basit ve anlaşılır bir şekilde görselleştirilebilmesidir. Her adımda, belirli bir koşul doğrultusunda veri grupları daha küçük parçalara bölünerek analiz edilir ve nihai tahmin elde edilir. Bu nedenle, karar ağaçları, finansal analizden tıbbi araştırmalara ve mühendislik uygulamalarına kadar birçok farklı alanda hem sınıflandırma hem de regresyon problemleri için sıkça tercih edilmektedir (Quinlan, 1986).

Bununla birlikte, karar ağaçlarının bazı sınırlamaları da bulunmaktadır. Büyük veri kümeleri üzerinde çalışırken aşırı öğrenme problemi ortaya çıkabilir ve bu durum modelin yeni verilere karşı genelleme yeteneğini olumsuz etkileyebilir. Bu tür zorlukların üstesinden gelmek amacıyla budama teknikleri ya da topluluk öğrenmesi yaklaşımları, örneğin "RF" gibi daha gelişmiş modeller kullanılabilir (Hastie ve ark., 2009).

4.1.3.2. Random forest (RF)

RF algoritması, çok sayıda karar ağacının sonuçlarını bir araya getirerek çalışan ve hem sınıflandırma hem de regresyon sorunları için uygulanabilen bir topluluk öğrenme yöntemidir. Bu yöntem, birden fazla karar ağacının birleştirilmesiyle oluşturulan bir model üzerinden, daha kesin ve güvenilir tahminler elde etmeyi hedefler. Şekil 4.4'de test edilen veri seti, algoritma içerisindeki her bir karar ağacı tarafından ayrı ayrı değerlendirilir; sonrasında ise her bir ağaç bir tahminde bulunarak "oy kullanır". En yüksek oy toplayan sonuç, nihai tahmin olarak kabul edilir. Bu metodoloji, genel olarak daha kuvvetli ve hataya daha az yatkın bir model oluşturur (Liang ve ark., 2020).



Şekil 4.4. Random forest örnek şeması (Uyanık, 2021)

RF algoritması, özellik seçimindeki rastgelelik ilkesiyle çalışır; bu yöntem, karar ağaçları arasındaki korelasyonu düşürür ve böylece algoritmanın genel tahmin gücünü artırarak performansını iyileştirir. RF algoritmasının sağladığı avantajlar şunlardır:

- Aşırı uyum (overfitting) problemini azaltır, böylece modelin genelleştirme kabiliyeti artar.
- Eğitim veri setindeki aykırı değerlerin olumsuz etkilerine karşı dayanıklıdır, modelin sağlamlığını artırır.
- Modelin parametrelerinin ayarlanması nispeten basittir, bu sayede ağaç budama işlemine daha az ihtiyaç duyulur. Bu özellikler, RF algoritmasını hem sınıflandırma hem de regresyon problemleri için cazip bir seçenek haline getirir.

RF algoritmasının çalışma prensibi, aşağıdaki adımlardan oluşur: Her bir karar ağacı için, orijinal veri setinden rastgele seçilen örnekler kullanılarak bir bootstrap örnek oluşturulur. Bu örnekler, orijinal veri seti ile aynı boyutta olacak şekilde, yerine koyma yöntemi ile seçilir. Karar ağaçları, maksimum büyüklüğe erişene kadar, budama yapılmaksızın bu ön yükleme örnekleri üzerinde büyütülür. Karar ağacı öğrenme süreci, her düğümde kullanılacak özelliklerin tamamı yerine rastgele seçilen bir alt küme üzerinden en iyi bölünmeyi belirlemek üzere modifiye edilmiştir. Bu yaklaşım, en iyi bölünmeyi bulma sürecini hızlandırarak, ağaçların öğrenme sürecini daha verimli hale getirir. Son olarak,

oluşturulan tüm ağaçların tahminleri bir araya getirilir ve ortalama alınarak algoritmanın nihai tahminleri üretilir (Nagalla ve ark., 2017).

RF'nin tahmin süreci belirli adımlar ve matematiksel formüller üzerinden ilerler. Bu süreç aşamaları;

1.Karar Ağaçları Oluşturulması: RF, topluluk (ensemble) öğrenme yöntemlerinden biridir ve birden fazla karar ağacını bir araya getirerek nihai tahmini gerçekleştirir. Bu modelde, her bir karar ağacı Ti(x) farklı bir alt küme veri üzerinde bağımsız olarak eğitilir. Bu alt kümeler, orijinal veri setinden rasgele örneklem alma yöntemiyle oluşturulur. Böylece her ağaç, veri setinin farklı bir perspektifini öğrenir ve çeşitliliği sağlar. Bu, modelin genel performansını ve doğruluğunu artırırken aynı zamanda aşırı öğrenme (overfitting) riskini azaltır (Breiman, 2001).

$$T_{i(x)} \tag{4.1}$$

Bu ifade, i-inci karar ağacının bir girdi olan x verisi üzerindeki tahminini göstermektedir.

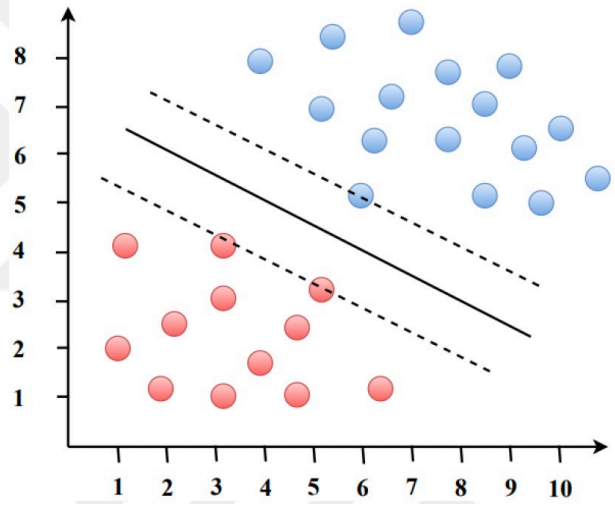
2.Öngörülerin Birleştirilmesi: Eğitim süreci tamamlandıktan sonra, her bir karar ağacının girdi verisi x için öngörüde bulunması sağlanır. RF, her ağacın yaptığı tahminlerin birleştirilmesiyle son tahmini elde eder. Bu birleştirme işlemi, sınıflandırma ve regresyon problemleri için farklılık gösterir:

- Sınıflandırma için: Her bir ağacın sınıf tahmini yapılır ve bu tahminler arasında çoğunluk oyu yöntemiyle (majority voting) en sık öngörülen sınıf, RF'nin nihai sınıf tahmini olarak kabul edilir.
- Regresyon için: Her bir ağacın verdiği sürekli değerler (örneğin ağırlık veya fiyat tahmini gibi) alınır ve bu değerlerin aritmetik ortalaması hesaplanır. Bu ortalama, modelin son tahmini olarak kabul edilir.

$$\hat{y} = \left(\frac{1}{n}\right) \sum_{\{i=1\}}^{n} T_{i(x)} \tag{4.2}$$

4.1.3.3. Support vector machine (SVM)

SVM, denetimli öğrenme kapsamında yer alan ve istatistiksel bir temele dayanan bir makine öğrenmesi yöntemidir. Bu yöntem, başlangıçta iki ya da daha fazla sınıfın doğrusal özelliklerini ayırt etmek üzere tasarlanmış, zamanla doğrusal olmayan veri özelliklerini de sınıflandırabilme yeteneğine sahip olmuştur. SVM, en etkin ayrımı sağlayacak karar sınırını belirleyerek iki farklı sınıfı birbirinden ayırma yeteneği üzerine kuruludur (Vapnik, 1999). Bu algoritma, çeşitli disiplinlerde tahminleme görevlerinde başarıyla uygulanmıştır (Cristianini ve Taylor, 2000).



Şekil 4.5. Doğrusal ayrılabilen veriler

SVM'nin tahmin süreci belirli adımlar ve matematiksel formüller üzerinden ilerler. Bu süreç aşamaları;

1. Sınıflandırma İçin: SVM, veriyi sınıflandırmak için iki sınıfı en iyi şekilde ayıran hiper düzlemi bulmayı amaçlar. SVM, sınıflar arasındaki mesafeyi maksimum yapan, yani iki sınıfa ait veri noktalarına en uzak mesafedeki düzlemi (hiper düzlem) bulur. Bu hiper düzlem, aşağıdaki formülle ifade edilir:

$$f(x) = w^T x + b \tag{4.3}$$

Bu formül, hiper düzlemi tanımlar ve sınıflandırma amacıyla kullanılır. SVM, iki sınıfı ayıran hiper düzlemi bulmak için en büyük marjı sağlamaya çalışır (Cortes ve Vapnik, 1995).

2. Sınıflandırma Kısıtlamaları: Sınıflandırmada SVM'nin amacı, veriyi doğru sınıflandıran bir hiper düzlem elde etmektir. Bunun için veri noktalarının sınıflandırılmasını sağlayan aşağıdaki kısıtlamalar uygulanır:

$$y_i(w^T x_i + b) \geq 1 \tag{4.4}$$

Bu koşul, her veri noktasının doğru sınıflandırılması için gereklidir (Hastie ve ark., 2009).

3. Regresyon İçin: SVM algoritmasının regresyon problemlerine uyarlanmış halidir. SVM, belirli bir marj içinde hatalara tolerans göstererek veri noktalarının tahmin edilmesini sağlar. SVM'nin temel amacı, bir marj bandı içinde kalacak şekilde doğrusal olmayan bir fonksiyon öğrenmektir. Bu modelde tahmin fonksiyonu şu şekilde ifade edilir:

$$f(x) = \sum_{\{i=1\}}^{n} \alpha_i K(x, x_i) + b \tag{4.5}$$

SVM, bir hata marjı belirleyerek tahminler yapar ve böylece bazı veri noktalarının marj içinde kalmasına izin verir (Smola ve Schölkopf, 2004).

4.1.3.4. Gradient boosting machine (GBM)

GBM, sınıflandırma ve regresyon gibi farklı alanlarda etkili olduğu kanıtlanmış ve bu nedenle popülerlik kazanmış bir makine öğrenmesi algoritmasıdır. Bu yöntem, ilk defa 1999 yılında Friedman tarafından tanıtılmış ve 2002'de sabit boyutlu karar ağaçlarını kullanarak yapılan iyileştirmelerle geliştirilmiştir. GBM, özellikle düşük performans gösteren öğrenme modellerini bir araya getirerek genel performansı artırma fikrine

dayanır. Bu süreç, birbiri ardına modeller kurarak ve her yeni modelle bir öncekinin hatalarını düzeltecek şekilde tahmin başarısını artırarak ilerler (Alpaydın, 2014).

GBM algoritmasının adımları, bir dizi düzenli prosedürü takip eder ve modelin performansını adım adım artırır. Bu sürecin ana hatları:

- İlk olarak, veri seti üzerinden bir karar ağacı kurulur ve bu ağacın tahminleri ile gerçek değerler arasındaki fark, yani kalıntı hata hesaplanır.
- Daha sonra, bu hata veri setinin yeni çıktıları olarak kabul edilir ve ikinci bir karar ağacı, bu hataları etiketler olarak kullanarak eğitilir. Bu, ilk ağacın yapamadığı tahminleri iyileştirmeyi amaçlar.
- Bu süreç, her yeni karar ağacının önceki ağaçların yapmış olduğu hataları düzeltmeye odaklanacak şekilde devam eder. Her adımda, yeni bir ağaç eğitilir ve bu ağaç, mevcut toplam tahminin hatalarını minimize etmek için tasarlanır.
- Bu adımlar, hata oranı belirlenen bir eşiğe düşene ya da başka bir durdurma kriteri karşılanana kadar devam eder.

Bu yaklaşımın temel avantajı, her adımda modelin performansını kademeli olarak iyileştirmesi ve özellikle zorlu veri setlerinde dahi etkili sonuçlar üretebilmesidir (Dixit, 2017).

Büyük veri setlerinin artan varlığıyla birlikte yeni algoritmalar geliştirilmiştir. XGBoost, CatBoost ve LightGBM, bu gelişmeler arasında öne çıkan ve popüler hale gelen modern gradyan artırma yöntemleridir. Bu algoritmalar, GBM algoritmasının temel prensiplerini benimseyerek, işlemlerin hızlanması ve tahmin başarısının artırılması yönünde özel optimizasyonlar sunar.

GBM'in tahmin süreci belirli adımlar ve matematiksel formüller üzerinden ilerler. Bu süreç aşamaları;

1.Başlangıç Tahmini: GBM, başlangıçta hedef değişkenin (örneğin, y) ortalama değeri üzerinden bir başlangıç tahmini yaparak işe başlar. Bu, ilk tahmin olarak kullanılacak değerdir ve aşağıdaki formülle gösterilir:

$$\hat{y}_i^{(0)} = \bar{y} \tag{4.6}$$

Bu adım, modelin kalibrasyonunu başlatır ve algoritmanın daha ileri tahminler yapması için bir temel sağlar (Friedman, 2001).

2. Kalıntı Hata Hesaplama: Her bir iterasyonda, mevcut tahminlerle gerçek değerler arasındaki fark, yani kalıntı hatalar hesaplanır. Kalıntı hatalar, bir sonraki tahminin hangi yönde ve ne kadar iyileştirilmesi gerektiğini belirlemek için kullanılır. Kalıntı hata $r_i^{(m)}$, aşağıdaki gibi ifade edilir:

$$r_i^{(m)} = y_i - \hat{y}_i^{(m-1)} \tag{4.7}$$

Bu adım, modelin hata oranını sürekli azaltarak tahminlerin doğruluğunu artırmayı amaçlar (Hastie ve ark., 2009).

3. Yeni Ağaç Ekleme: Her iterasyonda, mevcut kalıntı hataları modellemek için yeni bir karar ağacı eklenir. Bu ağaç, kalıntı hataları en aza indirecek şekilde eğitilir. Yeni tahminler, önceki tahminlere eklenerek iyileştirilir. Tahminlerin güncellenmiş hali aşağıdaki gibi verilir:

$$\hat{y}_i^{(m)} = \hat{y}_i^{(m-1)} + \gamma_m h_m(x_i) \tag{4.8}$$

Burada γ_m ağırlık katsayısını ve $h_m(x_i)$ yeni ağacın çıktısını temsil eder. Her yeni ağaç, modelin doğruluğunu artırır (Zhang ve Ma, 2012).

4. Toplam Model: GBM'in tüm iterasyonlardaki tahminleri toplamasıyla nihai model elde edilir. Tüm ağaçların toplamı, GBM'in son tahminini verir. Bu formül şu şekildedir:

$$\hat{y}_i = \sum_{m}^{=1M} \gamma_m h_m(x_i) \quad (4.9)$$

Bu, GBM modelinin tahmin gücünü artırmak için her iterasyondaki güncellemelerin birikimli etkisini temsil eder.

5. Kayıp Fonksiyonu ve Düzenleme: GBM algoritması, modelin karmaşık hale gelmesini kontrol etmek için düzenleme terimleri ekleyerek aşırı uyumu (overfitting) önler. Bu düzenlemeler, modelin hata oranını düşüren bir kayıp fonksiyonuna eklenir. GBM'de kayıp fonksiyonu şu şekilde tanımlanır:

$$\mathcal{L}(\Theta) = \sum_{i}^{=1n} \ell(y_i, \hat{y}_i) + \sum_{m}^{=1M} \Omega(h_m) \quad (4.10)$$

Bu formülde, $\boldsymbol{\Omega(h_m)}$ modelin karmaşıklığını kontrol eden düzenleme terimlerini ifade etmektedir (Natekin ve Knoll, 2013). Kayıp fonksiyonundaki düzenleme, modelin daha dengeli ve genellenebilir sonuçlar üretmesine katkıda bulunur.

4.1.3.5. Extreme gradient boost (XGBoost)

XGBoost, Chen ve Guestrin tarafından 2016'da geliştirilen, denetimli makine öğrenme algoritmaları arasında yer alan ve GBM algoritmasının temel ilkelerini daha ileriye taşıyan bir yöntemdir. XGBoost, hem sınıflandırma hem de regresyon problemlerini çözebilen, bir dizi regresyon karar ağacından oluşan modelleri bir araya getirerek güçlü bir öğrenme mekanizması oluşturmayı hedefler. Bu algoritma, birbiri ardına eklenen zayıf öğrenme modelleri, yani regresyon ağaçlarını kullanır; her bir iterasyonda, algoritmanın kalıntı hata hesaplanır ve bir sonraki ağacın daha iyi bir tahminde bulunabilmesi için yönlendirme amacı taşır. (Chen ve Guestrin, 2016)

XGBoost, GBM ile benzer çalışma prensiplerine sahip olsa da, özellikle büyük veri setlerinde daha etkili çalışabilmesi, hız ve modelin genel performansını iyileştiren optimizasyonları içermesi bakımından ayrılır. Bu özellikler, XGBoost'u karmaşık veri

setleri üzerinde çalışırken, diğer tahmin modelleri ve geleneksel gradient boosting yöntemlerine göre daha üstün kılar.

Regülarizasyon, XGBoost algoritmasında karmaşık modeller üzerinde kontrol sağlamak ve modelin aşırı öğrenmesini (overfitting) önlemek için kritik bir mekanizma sunar. Bu süreçte, gama parametresi kullanılarak, her iterasyonda hesaplanan kalıntı hata değerleri ölçeklendirilir. Bu ölçeklendirme, modelin karmaşıklığını ve dolayısıyla aşırı uyum riskini azaltır (Chen ve Guestrin, 2016).

XGBoost'un diğer bir önemli özelliği ise eksik verilerle başa çıkabilme kabiliyetidir. Birçok makine öğrenmesi algoritmasında eksik değerlerin doldurulması ya da kaldırılması gereken bir ön işleme adımı olmasına rağmen, XGBoost eksik değerleri modelleme sürecine dahil ederek bu sorunu çözer. Algoritma, bir ağacın her bir dalında eksik değerlerle karşılaştığında, bu değerleri en iyi şekilde yönlendirecek bir yol oluşturarak eksik veri problemine otomatik bir çözüm sunar. Bu yetenek, XGBoost'un veri ön işleme gereksinimlerini azaltır ve doğrudan çeşitli veri setleri üzerinde etkili bir şekilde çalışabilmesine olanak tanır (Chen ve Guestrin, 2016).

XGBoost ayrıca, ağaç budaması işlemini de etkili bir şekilde gerçekleştirir. Algoritma, belirlenen maksimum derinliğe (max_depth) kadar dallanma (bölme) işlemini gerçekleştirir. Bu sürecin ardından, model geriye doğru budamaya (pruning) başlar ve modelin genel performansına olumlu katkıda bulunmayan dalları kaldırır. Bu budama işlemi, modelin daha verimli olmasını sağlarken, aynı zamanda aşırı öğrenme riskini de azaltır. Bu iki özellik, XGBoost'un geniş çapta kullanılmasının ve farklı veri setleri üzerinde yüksek performans sergilemesinin ana nedenlerindendir (Singh, 2019).

XGBoost algoritmasının tercih edilmesinin ardındaki en güçlü nedenlerden biri, sunduğu hızlı işlem kabiliyetidir. GBM gibi geleneksel yöntemler ve diğer tahmin tekniklerine kıyasla, XGBoost önemli ölçüde daha hızlı sonuçlar elde etme avantajı sağlar. Bu hız, algoritmanın Windows ve Linux gibi farklı işletim sistemlerinde paralel işlemleri etkin bir şekilde kullanabilmesinden kaynaklanır; bu durum XGBoost'un GBM'e göre yaklaşık 10 kat daha hızlı performans sergilemesini sağlar. Ayrıca, XGBoost'un hafıza kullanımı

da optimize edilmiştir, bu sayede daha az hafıza tüketimi ile çalışır. Düşük donanım gereksinimleri, algoritmayı hem bireysel kullanıcılar hem de büyük ölçekli uygulamalar için cazip kılar, çünkü daha az kaynakla daha hızlı ve başarılı sonuçlar üretebilmektedir. Bu özellikler, XGBoost'u geniş bir uygulama yelpazesinde, özellikle büyük veri setlerini işlerken ve zaman kısıtlı projelerde tercih edilen bir yöntem haline getirir (Chen ve Guestrin, 2016).

XGBoost modelinin eğitimi sırasında elde edilecek başarının önemli bir parçası, parametre ayarlamasıdır. Etkili bir XGBoost modeli oluşturmak, genellikle iki temel adımdan geçer. Bunlar;

1. Temel model geliştirme: İlk olarak, XGBoost'un veri seti üzerindeki genel performansını değerlendirmek için varsayılan ayarlarla bir temel model geliştirilir. Bu aşama, modelin doğrudan performansını görmek ve iyileştirmeye açık alanları tespit etmek için önemlidir.
2. Parametre ayarlama ile ikinci model eğitimi: İkinci adımda, temel modelin sonuçları analiz edilerek parametrelerin optimize edilmesi sürecine geçilir. Bu süreç, temel modelin tahmin doğruluğunu artırmayı hedefler ve XGBoost modelinin veri setinin özelliklerine ve modelleme amacına daha uygun hale getirilmesine olanak tanır.

Parametre ayarlama süreci, genel olarak GBM ile benzer bir yaklaşımı izler. Başlangıçta, eğitim sürecini hızlandırmak adına yüksek bir öğrenme oranı seçilir. Daha sonra, modelin daha iyi performans göstermesini sağlamak için ağaca özgü parametreler ayarlanır. En son olarak, öğrenme oranı düşürülerek, modelin performansı en uygun düzeye çıkarılıncaya kadar tekrar eğitilir. Bu süreçte, modelin en uygun sınıflandırıcı sayısını belirlemek önemlidir. Bu aşamalı yaklaşım, modelin veri setine en uygun şekilde uyum sağlamasını ve dolayısıyla daha yüksek bir performans sergilemesini sağlar (Patrous, 2018).

XGBoost, her bir iterasyonda otomatik çapraz doğrulama yapabilme özelliği sayesinde, en ideal iterasyon sayısına tek seferde ulaşmayı kolaylaştırır (Singh, 2019). Bu özellik, modelin test hatasını daha doğru tahmin etmesine ve optimum model seçimine katkıda bulunur. Ayrıca, XGBoost'un eksik değerlerle başa çıkabilme yeteneği, saha verilerinde sık karşılaşılan eksiklikleri etkili bir şekilde yönetir. Bu sayede eksik veriler nedeniyle bilgi kaybı yaşanmaz ve veri setinin bütünlüğü korunur (Dixit, 2017).

XGBoost, veriyi işlerken "derin öncelikli arama" adı verilen bir yöntem kullanır. Bu yöntemde algoritma, karar ağacının en alt seviyesine kadar inerek karar noktalarına öncelik verir. En alt seviyedeki karar noktasına ulaştığında, daha ileri gidilemeyecek bir durumda geri dönerek diğer alt seviyelerdeki karar noktalarını araştırmaya devam eder. Bu yaklaşım, XGBoost'un veri içindeki karmaşık ilişkileri daha etkili bir şekilde modellemesine ve böylece daha isabetli tahminler yapmasına katkıda bulunur. XGBoost'un avantajları Şekil 4.6'da basitçe gösterilmiştir (Chen ve Guestrin, 2016).

Şekil 4.6. XGBoost avantaj şeması

XGBoost'un tahmin süreci belirli adımlar ve matematiksel formüller üzerinden ilerler. Bu süreç aşamaları;

1. Başlangıç Tahmini: XGBoost, tahmin sürecine hedef değişkenin ortalamasını temel alan bir başlangıç tahmini ile başlar. Bu başlangıç tahmini, ilk iterasyonda tüm veri noktaları için aynı değeri sağlar ve aşağıdaki formülle gösterilir:

$$\hat{y}_i^{(0)} = \bar{y} \tag{4.11}$$

Bu adım, algoritmanın başlangıçtaki tahmin doğruluğunu belirleyen bir referans noktası sunar (Chen ve Guestrin, 2016).

3. Kalıntı Hata Hesaplama: Her iterasyonda, mevcut tahmin ile gerçek değer arasındaki fark, yani kalıntı hata hesaplanır. Bu hata, modelin mevcut iterasyonda ne kadar iyileştirilmesi gerektiğine dair bilgi sağlar. Her bir veri noktası için kalıntı hata, şu formülle ifade edilir:

$$r_{\mathrm{i}}^{(\mathrm{m})} = y_{\mathrm{i}} - \hat{\mathrm{y}}_{\mathrm{i}}^{(\mathrm{m}-1)} \quad (4.12)$$

Bu aşama, hataların yönünü ve büyüklüğünü hesaplayarak bir sonraki iterasyonda modelin daha iyi tahmin yapmasını sağlar (Chen ve He, 2015).

3. Yeni Ağaç Ekleme: Bir sonraki aşamada, model yeni bir ağaç ekleyerek kalıntı hataları iyileştirmeye çalışır. Bu ağaç, hataların minimize edilmesine yönelik eğitilir ve mevcut tahminlere eklenir. Güncellenen tahminler şu formülle ifade edilir:

$$\hat{\mathrm{y}}_{\mathrm{i}}^{(\mathrm{m})} = \hat{\mathrm{y}}_{\mathrm{i}}^{(\mathrm{m}-1)} + \gamma_m\, h_m(x_{\mathrm{i}}) \quad (4.13)$$

Bu işlem, modelin tahmin doğruluğunu artırmaya yardımcı olur ve her iterasyonda modele eklenen ağaçlarla birlikte hataların daha da azaltılmasını sağlar (Zhang ve Ma, 2012).

4. Toplam Model: Tüm iterasyonlar tamamlandığında, modelin nihai tahmini tüm iterasyonlarda elde edilen tahminlerin toplamıyla elde edilir. XGBoost modeli için toplam tahmin şu şekilde ifade edilir:

$$\hat{\mathrm{y}}_{\mathrm{i}} = \sum_{m}^{=1\mathrm{M}} \gamma_m\, h_m(x_{\mathrm{i}}) \quad (4.14)$$

Burada:

- γ_m : Bu iterasyondaki ağaç için ağırlık katsayısıdır.
- $h_m(x_i)$: Yeni eklenen ağacın tahmin ettiği değerdir.

Bu aşama, XGBoost'un birikimli öğrenme sürecini tamamlayarak her bir ağacın sağladığı bilgiyi toplar ve modelin nihai tahminini oluşturur (Chen ve Guestrin, 2016).

5. Kayıp Fonksiyonu ve Düzenleme: XGBoost, modelin aşırı uyumunu (overfitting) kontrol altında tutmak için düzenleme terimleri kullanır. Bu düzenleme terimleri, modelin karmaşıklığını yönetmek amacıyla kayıp fonksiyonuna eklenir. XGBoost'un kayıp fonksiyonu aşağıdaki gibi formüle edilir:

$$\mathcal{L}(\Theta) = \sum_{i}^{=1n} \ell(y_i, \hat{y}_i) + \sum_{m}^{=1M} \Omega(h_m) \quad (4.15)$$

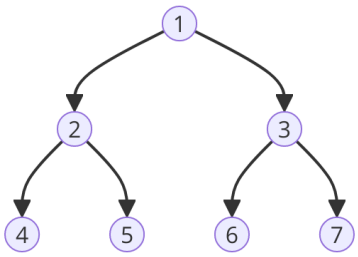
Burada:

- $\boldsymbol{\ell}(\mathbf{y}_i, \hat{\mathbf{y}}_i)$: Tahmin edilen değerlerle gerçek değerler arasındaki kaybı ifade eder.
- $\boldsymbol{\Omega}(\mathbf{h}_m)$: Modelin karmaşıklık seviyesini kontrol eden düzenleme terimidir.

Düzenleme terimleri, modelin aşırı öğrenmesini engelleyerek daha güvenilir ve genellenebilir bir model oluşturulmasına katkı sağlar (Natekin ve Knoll, 2013).

4.1.3.6. LightGBM

LGBM (LightGBM), adındaki "Light" ifadesiyle, yüksek hız ve verimliliğini vurgular; bu, Türkçeye "hafif" veya "çok hızlı" olarak çevrilebilir. Bir gradient boosting framework olan LightGBM, karar ağaçlarına dayalı bir yapı kullanır ve özellikle büyük veri setleri üzerinde çalışırken gösterdiği yüksek performansla öne çıkar. LGBM, derin öncelikli arama (Depth-First Search, DFS) yaklaşımını benimser. Bu yaklaşım ile bir karar ağacında dolaşırken, en derin düğümlere öncelik verilerek ilerlenir; örneğin Şekil 4.7'de belirtilen bir ağaç dolaşım sıralamasında düğümler 4-5-2-6-7-3-1 şeklinde ziyaret edilir. Bu strateji, LightGBM'in hızlı ve etkili bir şekilde eğitilmesine katkıda bulunur (Arslan, 2020).



Şekil 4.7. LightGBM ağaç dolaşım

LGBM, büyük veri setleriyle verimli şekilde çalışabilen ve az miktarda RAM kullanımı gerektiren bir algoritmadır. En iyi performansı elde etmek için en az 10.000 satırlık veri setleriyle kullanılması önerilir, çünkü küçük veri setlerinde performans sorunları yaşayabilir. LGBM, doğruluk odaklı bir yaklaşıma sahiptir ve 100'den fazla ayarlanabilir parametre ile yüksek derecede esneklik sunar. Bu, kullanıcıların spesifik ihtiyaçlarına uygun modeller oluşturmasına olanak tanır. Diğer karar ağacı temelli yöntemlere göre daha yüksek performans göstermesi ve büyük veri setlerinde hızlı sonuçlar üretebilmesi nedeniyle sıkça tercih edilir (Ke ve ark., 2017).

LightGBM'nin tahmin süreci belirli adımlar ve matematiksel formüller üzerinden ilerler. Bu süreç aşamaları;

1. Başlangıç Tahmini: LightGBM algoritması, tahmin sürecine hedef değişkenin (y) ortalamasını temel alarak bir başlangıç tahmini ile başlar. Bu, modelin ilk adımda tüm veri noktaları için aynı başlangıç tahminine sahip olmasını sağlar ve aşağıdaki formülle gösterilir:

$$\hat{y}_i^{(0)} = \bar{y} \tag{4.16}$$

Bu adım, algoritmanın başlangıçtaki tahmin doğruluğunu belirleyen bir referans noktası sunar (Chen ve Guestrin, 2016)

2.Kalıntı Hata Hesaplama: Her iterasyonda, mevcut tahmin ile gerçek değer arasındaki fark, yani kalıntı hata hesaplanır. Bu hata, modelin mevcut iterasyonda ne kadar

iyileştirilmesi gerektiğine dair bilgi sağlar. Her bir veri noktası için kalıntı hata, şu formülle ifade edilir:

$$r_i^{(m)} = y_i - \hat{y}_i^{(m-1)} \quad (4.17)$$

Bu aşama, hataların yönünü ve büyüklüğünü hesaplayarak bir sonraki iterasyonda modelin daha iyi tahmin yapmasını sağlar (Chen ve He, 2015).

3. Yeni Ağaç Ekleme: Bir sonraki aşamada, model yeni bir ağaç ekleyerek kalıntı hataları iyileştirmeye çalışır. Bu ağaç, hataların minimize edilmesine yönelik eğitilir ve mevcut tahminlere eklenir. Güncellenen tahminler şu formülle ifade edilir:

$$\hat{y}_i^{(m)} = \hat{y}_i^{(m-1)} + \gamma_m h_m(x_i) \quad (4.18)$$

Bu işlem, modelin tahmin doğruluğunu artırmaya yardımcı olur ve her iterasyonda modele eklenen ağaçlarla birlikte hataların daha da azaltılmasını sağlar (Zhang ve Ma, 2012).

4. Toplam Model: Tüm iterasyonlar tamamlandığında, modelin nihai tahmini tüm iterasyonlarda elde edilen tahminlerin toplamıyla elde edilir. LightGBM modeli için toplam tahmin şu şekilde ifade edilir:

$$\hat{y}_i = \sum_{m}^{=1M} \gamma_m h_m(x_i) \quad (4.19)$$

Burada:

- γ_m : Bu iterasyondaki ağaç için ağırlık katsayısıdır.
- $h_m(x_i)$: Yeni eklenen ağacın tahmin ettiği değerdir.

Bu aşama, tüm ağaçların katkısının birikimli olarak eklenmesiyle modelin nihai tahminini oluşturur (Guolin ve ark., 2017).

5. Kayıp Fonksiyonu ve Düzenleme: LightGBM, modelin aşırı uyumunu (overfitting) kontrol altında tutmak için düzenleme terimlerini kayıp fonksiyonuna ekler. Bu düzenlemeler, modelin karmaşıklığını yönetmek için kullanılır. LightGBM'de kayıp fonksiyonu şu şekilde tanımlanır:

$$\mathcal{L}(\Theta) = \sum_{i}^{=1n} \ell(y_i, \hat{y}_i) + \sum_{m}^{=1M} \Omega(h_m) \quad (4.20)$$

Burada:

- $\boldsymbol{\ell}(\mathbf{y}_i, \hat{y}_i)$: Tahmin edilen değerlerle gerçek değerler arasındaki kaybı ifade eder.
- $\boldsymbol{\Omega}(\mathbf{h_m})$: Modelin karmaşıklık seviyesini kontrol eden düzenleme terimidir.

Düzenleme terimleri, modelin daha genellenebilir olmasını sağlayarak aşırı öğrenmeyi önler (Guolin ve ark., 2017).

4.1.4. Veri işleme araçları

4.1.4.1. Python

Python, Hollandalı bir yazılımcı olan Guido Van Rossum tarafından 1990'lı yıllarda geliştirilmeye başlanmış kolay yazım biçimiyle öne çıkan bir programlama dilidir. Diğer dillere kıyasla daha basit olması ve herhangi bir derleyici gerektirmemesiyle dikkat çeker (Özgül, 2013). Açık kaynak kodlu ve ücretsiz olması, Python'u özellikle popüler kılar. TensorFlow, Keras ve Scikit-learn gibi kütüphaneler, Python'u makine öğrenimi ve veri bilimi için tercih edilen diller arasına sokar. Pandas, NumPy ve Matplotlib gibi veri analizi ve görselleştirme kütüphaneleriyle Python, veri bilimcileri ve geliştiriciler için vazgeçilmez bir araç haline gelmiştir (Arslan, 2019).

4.1.4.2. Pandas kütüphanesi

Pandas kütüphanesi, Python programlama dilinde geliştirilmiş, veri analizi ve veri işleme için güçlü araçlar sunan bir kütüphanedir. Pandas, özellikle zaman serileri ve sayısal dizilerle çalışmayı kolaylaştırırken, çeşitli veri yapılarını da destekler. Bu, kullanıcıların

veri setlerini kolayca oluşturmasına, düzenlemesine ve analiz etmesine olanak tanır (Chen, 2017). Pandas'ı bir Python uygulamasına dahil etmek için yalnızca *import pandas* komutunun kullanılması gerekir. Bu komut, kütüphanenin tüm fonksiyonlarını ve sınıflarını kullanıma sunar, böylece veri üzerinde çeşitli işlemler yapılabilmektedir.

4.1.4.3. NumPy kütüphanesi

Numpy, çok boyutlu diziler üzerinde matematiksel işlemleri kolaylıkla ve hızla yapmak için tasarlanmış bir Python kütüphanesidir. NumPy'nin kökeni, 1995 yılında Jim Hugunin tarafından geliştirilen Numeric kütüphanesine dayanmaktadır. Daha sonra, 2005 yılında Travis Oliphant, Numeric ve Numarray kütüphanelerinin özelliklerini birleştirerek NumPy'yi oluşturmuştur. Numpy'ı bir uygulamada kullanabilmek için, import numpy komutu kullanılır. Bu, Numpy kütüphanesinin tüm fonksiyonlarını uygulamanıza entegre eder ve veri analizi, doğrusal cebir işlemleri, dört işlem gibi matematiksel işlemleri büyük veri setleri üzerinde hızlı bir şekilde gerçekleştirmenizi sağlar.

4.1.4.4. Scikit-Learn kütüphanesi

Scikit-learn, Python programlama dilinde makine öğrenmesi ve veri bilimi projeleri için tasarlanmış ücretsiz ve açık kaynaklı bir kütüphanedir. Bu kütüphane, çeşitlilik gösteren algoritmaları bünyesinde barındırması sayesinde, sınıflandırma, regresyon, kümeleme gibi birçok farklı makine öğrenmesi işlemini destekler. Kullanıcı dostu arayüzü ve geniş dökümantasyonu ile hem deneyimli geliştiricilerin hem de bu alana yeni başlayanların kolayca kullanabileceği bir yapı sunar. Scikit-learn'ün zengin algoritma kütüphanesi, veri ön işleme, model seçimi ve değerlendirme gibi makine öğrenmesinin temel süreçlerini kapsar, bu da onu veri bilimi projeleri için vazgeçilmez bir araç haline getirir.

4.1.4.5. Matplotlib kütüphanesi

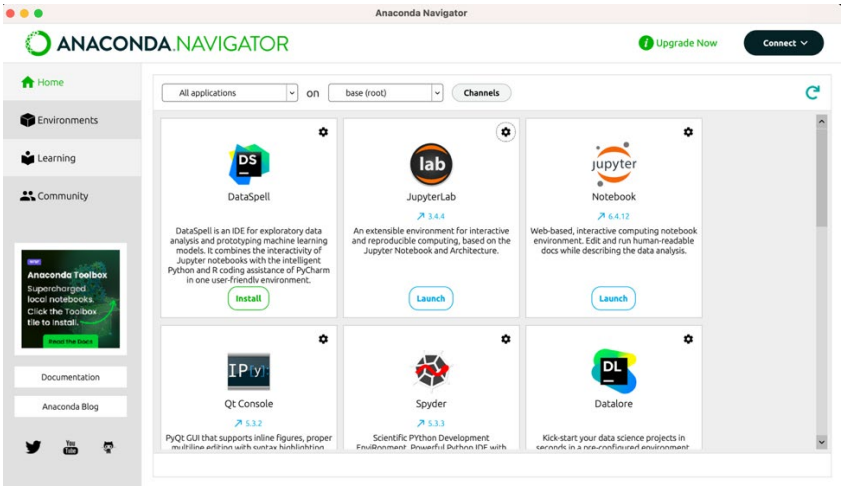
Matplotlib, verileri çeşitli formatlarda ve grafik türlerinde görselleştirmek için kullanılan bir Python kütüphanesidir. Durağan ve etkileşimli görseller oluşturabilme yeteneğiyle, kullanıcıların veri analizi sonuçlarını kolayca yorumlamasına yardımcı olur. Grafikler

üzerinde düzenleme yapma, yakınlaştırma ve uzaklaştırma gibi işlevler sunarak veri görselleştirme sürecini zenginleştirir. Ayrıca, yüksek kaliteli çıktılar alınmasını sağlayarak sunumlar ve bilimsel çalışmalar için ideal bir araçtır. Matplotlib, veri bilimcileri ve araştırmacılar arasında popüler bir seçimdir, çünkü karmaşık veri setlerini anlaşılır ve estetik görseller haline getirme gücüne sahiptir.

4.1.4.6. Anaconda

Anaconda, makine öğrenmesi, veri analizi ve veri bilimi gibi alanlarda çalışmalar yapmak için tasarlanmış, pek çok popüler uygulama yazılımını ve kütüphaneyi içinde barındıran bir Python ekosistemidir. Bu platform, Python ve R dilleri için geliştirilmiş kapsamlı bir paket yöneticisi ve ortam yönetimi sistemine sahiptir. Anaconda, özellikle bilim, mühendislik ve finans alanlarında veri işleme, veri analizi ve makine öğrenimi projeleri için tercih edilen bir platformdur.

Anaconda'nın içerdiği araçlardan biri de Jupyter Notebook'tur. Jupyter Notebook, canlı kod, denklemler, görselleştirmeler ve açıklayıcı metinleri bir arada sunabilen, interaktif bir web uygulamasıdır. Araştırmacılar ve veri bilimcileri tarafından veri temizleme, istatistiksel modelleme, görselleştirme ve makine öğrenimi gibi çeşitli görevlerde yaygın olarak kullanılmaktadır. Anaconda'nın bu kapsamlı ekosistemi sayesinde, kullanıcılar projeleri üzerinde hızlı ve etkili bir şekilde çalışabilirler.



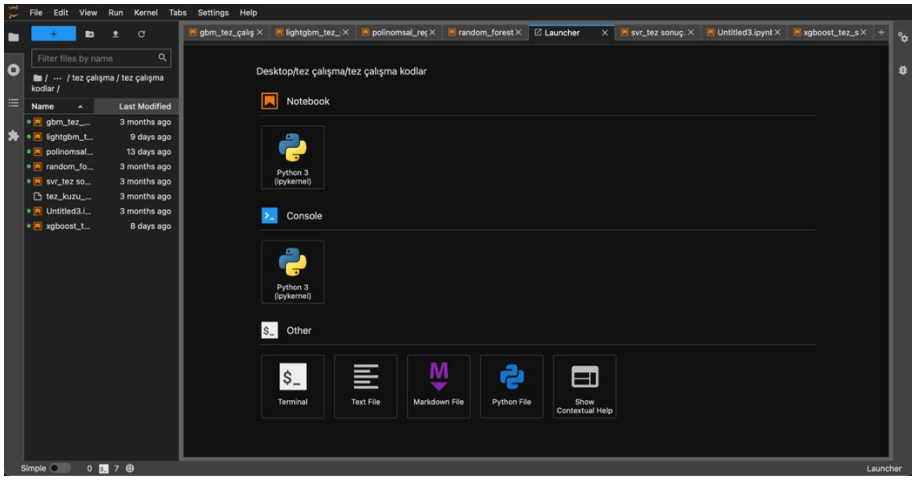
Şekil 4.8. Anaconda platformu

4.1.4.7. Jupyter notebook

Jupyter Notebook, Python başta olmak üzere Julia, R gibi diğer programlama dillerini de destekleyen interaktif bir web uygulamasıdır. Kullanıcı dostu arayüzü, programlama öğrenim sürecini kolaylaştırır ve kodların yazılmasını, okunmasını ve derlenmesini sadeleştirir. Bu basitlik, özellikle Python programlama dilini öğrenmek isteyenler için Jupyter Notebook'u tercih edilen bir platform yapar (Tuzcu, 2020). Kodların yanı sıra, açıklayıcı metinler, denklemler ve görselleştirmeler içerebilen bu araç, eğitim materyalleri, bilimsel çalışmalar ve veri analizi projeleri için idealdir.

4.1.4.8. JupyterLab

JupyterLab, Jupyter Notebook'un gelişmiş bir sürümü olarak görülebilecek, interaktif bir geliştirme ortamıdır. Bilim insanları, araştırmacılar ve veri analistleri tarafından geniş bir yelpazede kullanılan bu ortam, kod yazma, veri işleme, sonuçların görselleştirilmesi ve bilimsel makalelerin hazırlanması gibi işlemleri tek bir arayüz üzerinden yapmayı mümkün kılar. JupyterLab, kullanıcıların not defterleri, kod editörleri, terminaller ve özel bileşenleri bir arada kullanabilmelerine olanak tanıyan modüler bir yapıya sahiptir.



Şekil 4.9. JupyterLab arayüzü

4.2.Yöntem

Bu çalışmada, Romanov kuzularına ait veriler Python programlama dili ve çeşitli makine öğrenmesi algoritmaları kullanılarak Jupyter Notebook ortamında, Pandas ve Numpy kütüphaneleri ile düzenlenmiştir. Ayrıca, Scikit-learn kütüphanesi kullanılarak tahminleme için modeller oluşturulmuştur. Daha öncede belirtildiği gibi kullanılan algoritmalar şunlardır:

1. Random Forest (RF)
2. Extreme Gradient Boost (XGBoost)
3. LightGBM
4. Support Vector Machine (SVM)
5. Gradient Boosting Machine (GBM)

Çalışmada, makine öğrenmesi algoritmalar kullanılarak öncelikle veri seti %75 eğitim, %25 test olacak şekilde ayrılmıştır ve sonrasında modellerin başarı oranları değerlendirilmiştir. Ayrıca, her algoritma ve metot için çapraz doğrulama yöntemi ile yeniden başarı oranları hesaplanmış , böylelikle en etkili model belirlenmiştir.

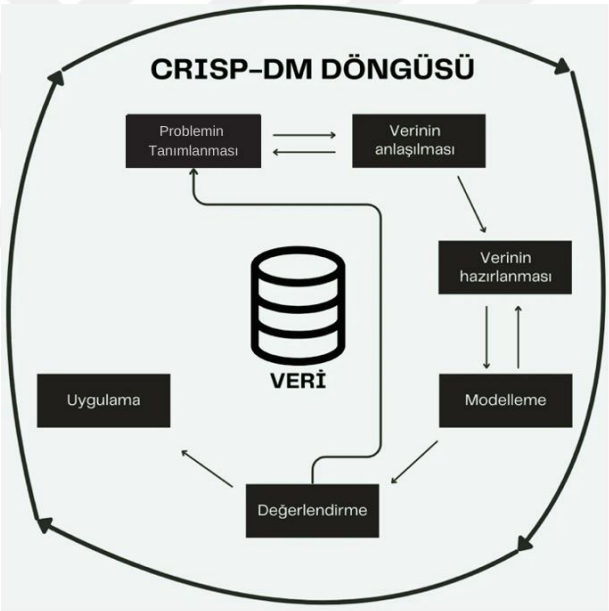
4.2.1. Cross-Industry Standard Process for Data Mining (CRISP-DM)

Cross-Industry Standard Process for Data Mining (CRISP-DM), veri madenciliği projelerinin yönetimi ve uygulanması için geniş çapta kabul gören bir standart süreç modelidir. Bu model, projenin döngüsü boyunca izlenmesi gereken aşamaları, bu aşamalardaki görevleri ve görevler arası ilişkileri kapsamlı bir şekilde tanımlar. CRISP-DM, projenin amaçlarına, uygulayıcının tecrübesine ve özellikle çalışılan verinin niteliğine göre esnek uygulamalar sunar, bu nedenle veri madenciliği çalışmalarında baştan sona bir rehber olarak işlev görür. Bu süreç, projenin hedeflerine uygun en etkili sonuçların elde edilmesi için veri madenciliği görevlerinin her birinin nasıl planlanıp gerçekleştirileceği konusunda detaylı bir yol haritası sunar. Projede karşılaşılan her bir görev ve bu görevlerin birbirleriyle olan ilişkisi, projenin başarısında önemli bir rol oynar. CRISP-DM, veri madenciliği alanında bir standart olarak kabul edilir ve çeşitli endüstrilerdeki veri madenciliği projelerinde uygulanabilirliği ile bilinir (Şeker, 2018). Bu modelin esnek yapısı, farklı türdeki veri setleri ve hedeflerle çalışılabilmesine olanak

tanır, bu da onu veri madenciliği alanında yaygın olarak kullanılan bir yöntem haline getirir.

Bu çalışmada izlenecek ana yöntem olan CRISP-DM adımları akış şemasına göre, Şekil 4.10. da gösterilmiştir. CRISP-DM adımları akışı oluşturmaktır şöyledir;

• Problemin tanımlanması
• Verinin Anlaşılması
• Verinin Hazırlanması
• Modelleme
• Değerlendirme
• Uygulama



Şekil 4.10. CRISP-DM adımları akışı

4.2.1.1. Problemin tanımlanması

CRISP-DM modelinin "Problemin tanımlanması" aşaması, projenin başlangıcında yer alır ve projenin temelini oluşturur. Bu aşama, projenin genel hedeflerinin, kapsamının ve önceliklerinin tanımlanması ile ilgilenir. Problemin tanımlanması, bir veri madenciliği projesinin başarısını doğrudan etkileyebilecek temel sorunları ve fırsatları tanımlamaya

odaklanır. Durum değerlendirmesi, mevcut iş ortamı, süreçler, veri kaynakları ve ilgili taraflar hakkında bir incelemedir. Bu adım, projenin hangi problemlerini çözebileceğini, hangi verilerin kullanılabilir olduğunu ve projenin gerçekleştirilebilirliğini anlamak için önemlidir. Proje hedeflerinden yola çıkarak, bu adımda spesifik veri madenciliği hedefleri belirlenir. Bu hedefler, proje hedeflerine ulaşmak için veri üzerinde ne tür analizlerin yapılacağını açıkça ifade eder. Plan, projenin kapsamını, takvimini, gerekli kaynakları, riskleri ve başarı kriterlerini içerir. Proje planı, projenin tüm süreci boyunca takip edilir ve gerektiğinde güncellenir. Bu aşama, projenin amacını ve yönünü netleştirir, projenin başarıyla tamamlanmasına zemin hazırlar (Chapman ve ark., 2000).

4.2.1.2. Verinin anlaşılması

Modelleme sürecinde modelin tasarımı ve kurulumu sırasında ortaya çıkabilecek sorunları önlemek için, verinin anlaşılması aşaması büyük önem taşır. Verinin anlaşılması ve analizi ne kadar iyi yapılırsa, modelleme aşaması da o kadar başarılı bir şekilde gerçekleştirilebilir (Doğan ve Arslantekin, 2023).

4.2.1.3. Verinin hazırlanması

Bu aşama, veri setindeki ham veriler ve uygulamada kullanılacak verilerin belirlenmesi, test ve eğitim verilerinin seçimi ve analiz edilmesi işlemlerini kapsar. Bu süreç, uygulamayı gerçekleştirecek olan analistin en fazla zaman harcadığı ve en önemli aşamalardan biridir (Doğan ve Arslantekin, 2023).

4.2.1.4. Modelleme

Bu aşamada, uygulamada kullanılacak gerçek modeli tasarlama ve oluşturma süreci başlar. Bu süreçte, analist modelde kullanılacak verilerin ne kadarının test, ne kadarının eğitim verisi olacağını belirler ve bu verileri hazırlamaya geçer. Model hazır hale getirildikten sonra, uygulamada kullanılacak tekniklerin uygulanmasına geçilir. Bu teknikler, eğitim verisi setinin sınıflandırılması, eğitim verileri ile ilişkilerin kurulması ve kümeleme tekniklerini içerir. Model, eğitim ve test verileri kullanılarak karşılaştırmalı bir

şekilde test edilir ve elde edilen sonuçlar analiz edilir. Analiz aşamasından sonra, elde edilen sonuçlara göre veri hazırlama kısmı tekrar gözden geçirilebilir (Doğan ve Arslantekin, 2023).

4.2.1.5. Değerlendirme aşaması

Değerlendirme aşamasında, modelin iş hedeflerine ulaşma düzeyi ve beklenen performans kriterlerine uygunluğu değerlendirilir. Bu süreçte, modelin test veri seti üzerindeki performansı gözden geçirilir ve iş problemini çözme yeteneği analiz edilir. Ayrıca, projenin başlangıcından itibaren gerçekleştirilen işlemler incelenerek, gelecekteki projeler için iyileştirme fırsatları belirlenir (Chapman ve ark., 2000).

4.2.1.6. Uygulama aşaması

Uygulama aşaması, modelin iş süreçlerine entegrasyonunu ve elde edilen bilginin kullanıma sunulmasını kapsar. Bu aşamada, model üretim ortamına alınır ve kullanıcılar modelin işleyişi hakkında bilgilendirilir. Modelin performansı sürekli izlenir ve gerektiğinde güncellemeler yapılır. Böylece model, sürekli olarak iş değeri yaratır. Ayrıca, bu aşama elde edilen sonuçların iş süreçlerinde etkin bir şekilde kullanılmasını içerir (Chapman ve ark., 2000).

4.2.2 Modellerin performans değerlendirmesi

Bu çalışmada, algoritmalar JupyterLab ortamında Python programlama dili kullanılarak kodlanmıştır. Makine öğrenimi algoritmalarında, açık kaynaklı bir makine öğrenimi kitaplığı olan Scikit_learn kitaplığı kullanılmıştır. Grafikleri çizmek için Matplotlib kitaplığı kullanılmıştır. Performansın değerlendirilmesi Scikit-learn metrikleri kullanılarak yapılmıştır. Tüm hiper parametreler, beş kat çapraz doğrulama yöntemi ve Grid Search işlevi kullanılarak ayarlanmıştır.

4.2.3. Yöntem karşılaştırma kriterleri

Tahmin modelinin veri üzerindeki performansını değerlendirmek ve model tarafından öngörülen değer ile gerçek değer arasındaki ilişkiyi ölçmek için çeşitli hata metrikleri kullanılır. Bu çalışmada, tahmin modellerinin değerlendirilmesi için HKOK, HKO ve R^2 kullanılmıştır. Bu hata ölçütleri arasında, tahmin modelinin doğruluk oranı R^2 yöntemi ile belirlenirken, algoritmanın hata ölçüsü HKOK ve HKO ile değerlendirilir. Bu metrikler 0 ile ∞ arasında değerler alabilir. Ancak, metriklerden elde edilen değerler küçüldükçe tahmin algoritmasının daha başarılı olduğu kabul edilir. HKOK, kare işlevinden önceki ortalama nedeniyle büyük veri kümelerinde daha avantajlıdır. Ayrıca, R^2 ile ölçülen doğruluk oranı 1'e yaklaştıkça, tahmin modelinin başarısının daha yüksek olduğu anlaşılır (Wang ve Xu, 2004).

4.2.3.1. Hata kareler ortalaması (HKO)

HKO, gerçek değer ile tahmin edilen değer arasındaki mutlak hataya karşılık gelir ve herhangi bir yön dikkate alınmadan verideki (tahminler) hataların ortalama büyüklüğünü ölçmek için kullanılır. Düşük bir HKO, modelin tahmininin gerçek değere daha yakın olduğunu gösterir.

$$HKO = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.21}$$

4.2.3.2. Belirtme katsayısı (R^2)

Belirme katsayısı R^2, tahminin doğruluğunu temsil eder. 1'e yakın R^2 değeri daha fazla tahmin doğruluğu anlamına gelir. Diğer bir ifade ile 1' e yakın değerler modelin bağımlı değişkenindeki varyansı iyi açıkladığını gösterir.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{4.22}$$

4.2.3.3. Hata kareler ortalamasının karekökü (HKOK)

Regresyon eğrisinde tahmin edilen değerler ile gerçek değerleri arasındaki mesafenin ölçüsü artıkları ifade eder ve artıkların standart sapması, hatanın ortalama karekökünü verir. Düşük bir HKOK değeri, modelin tahminlerinin gerçek değerlere daha yakın olduğunu gösterir.

$$HKOK = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{4.23}$$

5. BULGULAR ve TARTIŞMA

Bu çalışmada, Romanov kuzularının büyüme verileri üzerinde çeşitli makine öğrenmesi algoritmalarının performansını değerlendirmek için veri seti üzerinde RF, GBM, XGBoost, SVM ve LightGBM algoritmaları uygulanmıştır. Algoritmalar için en uygun parametre değerleri, Python programlama dilinde Grid Search yöntemi ile beş katlı çapraz doğrulama tekniği kullanılarak belirlenmiştir. Bu yöntemle, modelin hiperparametreleri optimize edilmiş ve her algoritma için en iyi sonuçları veren parametre kombinasyonları seçilmiştir.

Çalışmada kullanılan algoritmalar Romanov kuzularının ilk 180 günlük büyüme verileri üzerinde eğitimden geçirilmiş ve ardından tahmin performansları belirlenmiştir. Modelleme sürecinde, sağrı yüksekliği (SY), cidago yüksekliği (CY), vücut uzunluğu (VU), göğüs derinliği (GD), göğüs çevresi (GÇ) ve kürekler arası göğüs genişliği (KAGG) bağımsız değişkenlerinin ağırlık tahminleri üzerindeki önem düzeyleri belirlenmiştir. Analizler sonucunda, hangi bağımsız değişkenlerin tahminlerde daha etkili olduğunu göstermek için grafiklerle desteklenmiştir.

Çalışmada son aşamada, algoritmaların tahmin ettiği değerlerle gerçek ölçüm değerleri karşılaştırılmış ve her algoritmanın tahmin doğruluğu görselleştirilmiştir.

5.1. Makine Öğrenmesi Algoritmaları Hata Metrikleri

Romanov kuzularının büyüme verilerini temel alan bir veri seti üzerinde RF, GBM, XGBoost, SVM ve LightGBM algoritmaları tahmin performansını değerlendirmektedir. Performans değerlendirmesinde HKOK, HKO ve R^2 kullanılmıştır. Çizelge 5.1'de sunulan sonuçlar, her bir algoritmanın veri seti üzerindeki performansını göstererek en iyi modeli belirlemeye yönelik bir karşılaştırma yapmaktadır.

Çizelge 5.1. Değişkenlere ait tanımlayıcı istatistikler

Algorithms	HKOK	HKO	R^2
Random Forest (RF)	1.01365	1.02749	0.97948
Support Vector Machine (SVM)	1.09043	1.18903	0.97626
Extreme Gradient Boost (XGBoost)	1.03964	1.08085	0.97842
LightGBM	1.08735	1.18233	0.97639
Gradient Boosting Machine (GBM)	0.99534	0.99131	0.98022

Çizelge 5.1 incelendiği zaman en iyi performansı GBM algoritmasının gösterdiği görülmektedir. GBM, en düşük HKOK (0.99534) ve HKO (0.99131) değerleri ile en yüksek R^2 (0.98022) değerlerine sahiptir. Bu bulgu, GBM algoritmasının tahminlerde en düşük hata oranına sahip olduğunu ve veri setine en iyi uyumu sağladığını göstermektedir.

RF algoritması, GBM'e yakın bir performans göstererek HKOK değeri 1.01365 ve R^2 değeri 0.97948 ile öne çıkmıştır. RF algoritmasının da Romanov kuzularının ağırlık tahmini için güvenilir bir model olduğu görülmüştür. XGBoost algoritması da iyi bir performans göstermiş olup HKOK (1.03964) ve R^2 (0.97842) değerleri ile güvenilir bir model olduğu görülmüştür.

SVM ve LightGBM algoritmaları ise diğer algoritmalara kıyasla daha yüksek hata oranlarına sahiptir. SVM, 1.09043 HKOK ve 1.18903 HKO değeri ile en yüksek hata oranlarına sahipken, LightGBM 1.08735 HKOK ve 1.18233 HKO değeri ile benzer sonuçlar vermiştir. Her iki algoritmanın da veri setine uyumu yüksek olsa da ($R^2 > 0.97$), diğer algoritmalara kıyasla daha az başarılı oldukları sonucuna varılmıştır.

5.2 Random Forest (RF) Sonuçları

RF modellemesinin optimum değerleri Python'daki Grid Search işlevi kullanılarak beş kat çapraz doğrulama tekniği ile elde edilmiştir. RF algoritması için kullanılan parametre değerleri ve optimum sonuçlar Çizelge 5.2'de gösterilmiştir. Model için en iyi kombinasyonun max_dept parametresinin 9, max_features parametresinin 3 ve n_estimators parametresinin 1000 olduğu durumda bulunmuştur.

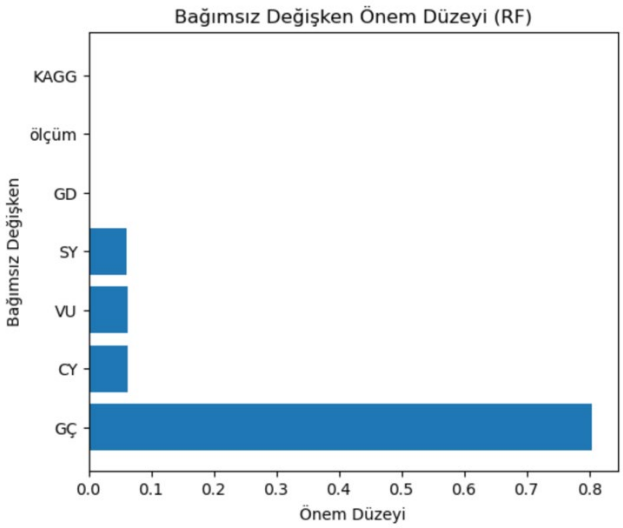
Çizelge 5.2. RF algoritması seçilen hiperparametre değerleri

Parametreler	max_depth	max_features	n_estimators
Aralıklar	[1, 2, 3, 4, 5, 6, 7, 8, 9]	[3, 5, 10, 15]	[1000, 2000, 3000]
Seçilen Değerler	9	3	1000

Aşağıda verilen Şekil 5.1 RF algoritmasının bağımsız değişkenler arasındaki önem düzeylerini göstermektedir. RF algoritması kullanılarak yapılan analiz sonucunda bağımsız değişkenlerin önem düzeyleri gösterilmiştir. Her bir bağımsız değişkenin, modelin tahmin sonuçlarına olan katkısı, grafikteki barların uzunluğu ile temsil edilmektedir.

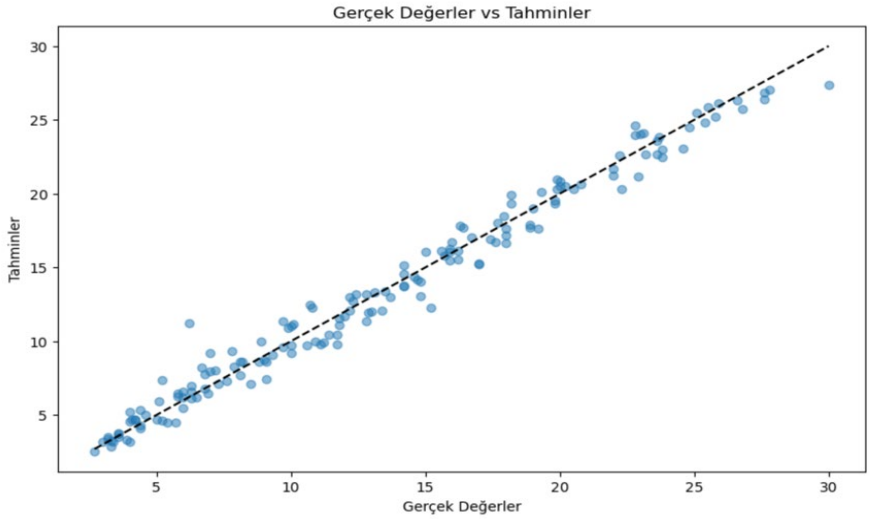
Göğüs çevresi değişkeni, yaklaşık %80'lik bir önem düzeyi ile modelde tahmin edilen ağırlık üzerinde en yüksek etkiye sahip değişken olarak öne çıkmaktadır. Bu, GÇ, modelin ağırlık tahminlerinde belirleyici bir rol oynadığını ve diğer bağımsız değişkenlere kıyasla çok daha yüksek bir katkı sağladığını göstermektedir.

Diğer değişkenler ise oldukça düşük önem düzeylerine sahiptir. Özellikle CY, VU ve SY bağımsız değişkenleri, modelin tahminlerinde düşük bir rol oynamaktadır. GD, ölçüm (Romanov kuzuların doğumdan itibaren 180. güne kadar her 15 günlük periyot zamanları) ve KAGG değişkenleri ise önemli bir katkı sağlamadığı görülmektedir.



Şekil 5.1. RF bağımlı değişkenlerin önem düzeyleri

Şekil 5.2 RF algoritmasının Romanov kuzularının büyüme ölçümlerine dayalı tahminleri ile gerçek değerler arasındaki ilişkiyi göstermektedir.



Şekil 5.2. RF algoritması kullanılarak tahmin değerlerinin gerçek ölçüm değerlerine göre dağılımı

Grafikte görülen noktalar, tahmin edilen değerlerin büyük bir kısmının gerçek değerlerle uyumlu olduğunu ortaya koymaktadır. Özellikle noktaların doğrusal bir eğri etrafında yoğunlaşması, algoritmanın tahmin doğruluğunun oldukça yüksek olduğunu ve verilerin genel eğilimini iyi bir şekilde yakaladığını göstermektedir.

Grafikteki eğim, RF algoritmasının verileri başarılı bir şekilde modellediğini ve düşük hata oranına sahip olduğunu kanıtlamaktadır. Bu durum, HKOK ve HKO gibi performans ölçütlerinin de desteklediği bir bulgudur. Algoritmanın R^2 değeri ise, tahminlerin %98 oranında doğru olduğunu göstermektedir.

Romanov kuzularının ilk 180 gün boyunca 15 gün aralıklarla çeşitli vücut ölçümleri ve RF algoritması kullanılarak yapılan tahmin değerleri gösterilmektedir. Bu ölçümler arasında SY, CY, VU, GD, GÇ ve KAGG yer almaktadır. Ayrıca, her ölçüm periyodunda gerçek ölçüm değerleri ve tahmin edilen değerler Çizelge 5.3'de verilmiştir.

Çizelge 5.3. RF algoritması ile Romanov kuzuları ilk 180 gün tahmin değerleri

ölçüm	SY	CY	VU	GD	GÇ	KAGG	Ölçüm Değerleri	Tahmin Değerleri
1	26.20	25.10	23.80	8.00	27.50	5.30	1.90	2.00
15	34.10	33.40	30.40	11.90	38.60	9.00	4.40	4.70
30	39.30	38.80	33.30	14.00	45.10	10.20	5.20	7.34
45	42.10	41.80	36.50	15.50	50.20	11.30	6.00	7.40
60	42.80	42.60	37.20	15.80	51.80	11.70	7.20	8.16
75	43.50	43.30	37.90	16.40	53.00	11.90	8.70	9.25
90	44.00	43.80	38.20	17.00	54.00	12.00	9.70	11.38
105	45.20	45.00	40.40	17.40	57.30	12.10	12.20	12.99
120	46.40	46.20	41.70	19.30	61.60	12.30	13.60	14.21
135	47.80	47.60	42.60	20.40	64.40	12.30	15.80	16.05
150	49.10	49.00	43.10	20.30	68.20	12.60	17.90	18.50
165	50.80	50.50	44.60	22.40	71.00	12.90	20.20	20.74
180	53.90	53.60	49.80	23.00	74.20	13.30	22.90	23.59

Sonuç olarak, RF algoritmasının, Romanov kuzularının büyüme sürecini tahmin etmede etkili bir araç olduğu görülmektedir. Bu bulgular, algoritmanın genel olarak yüksek bir doğrulukla çalıştığını ve çalışmada elde edilen veriler üzerinde güvenilir sonuçlar verdiğini ortaya koymaktadır. Modelin yüksek başarı oranı, RF algoritmasının bu tür verilerle yapılacak tahminlerde güvenilir bir seçenek olduğunu göstermektedir.

5.3. Support Vector Machine (SVM) Sonuçları

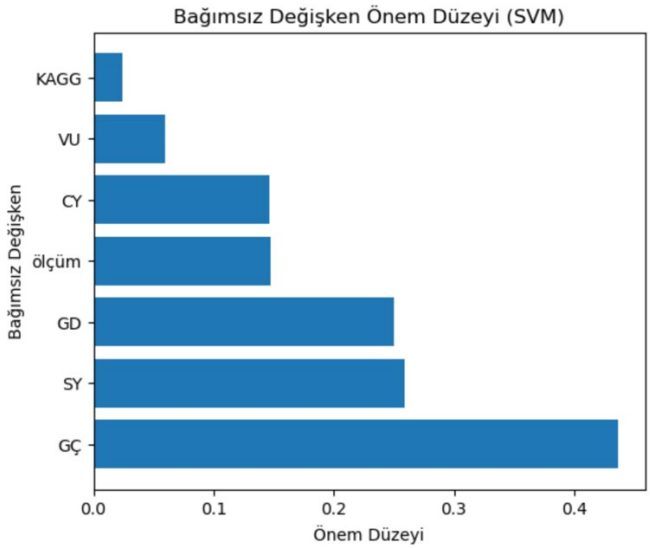
SVM modellemesinin optimum değerleri Python'daki Grid Search işlevi kullanılarak beş kat çapraz doğrulama tekniği ile elde edilmiştir. SVM algoritması için kullanılan parametre değerleri ve optimum sonuçlar Çizelge 5.4'de gösterilmiştir. Deneysel sonuçlar, SVM modeli için en iyi kombinasyonun C parametresinin 100, gamma parametresinin 0.1 ve epsilon parametresinin 0.1 olduğu durumda bulunmuştur.

Çizelge 5.4. SVM algoritması seçilen hiperparametre değerleri

Parametreler	C	gamma	epsilon
Aralıklar	[0.1, 1, 10, 100]	['scale', 'auto', 0.01, 0.1, 1, 10]	[0.01, 0.1, 1]
Seçilen Değerler	100	0.1	0.1

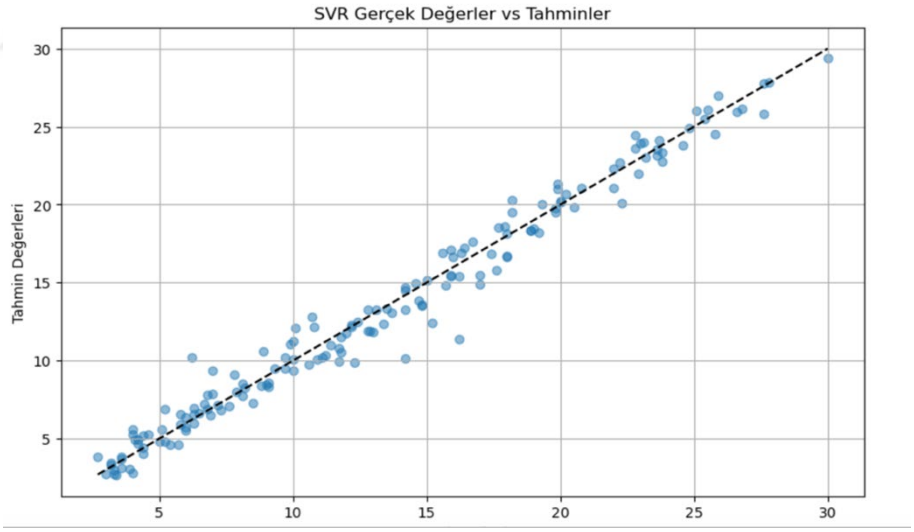
Aşağıda verilen Şekil 5.3 SVM algoritması ile bağımsız değişkenlerin ağırlık tahminleri üzerindeki önem düzeyleri gösterilmektedir. Şekil 5.3. model tahmininde hangi bağımsız değişkenlerin daha fazla katkı sağladığını görsel olarak ifade etmektedir. Bu analizde GÇ, yaklaşık %40'lık bir önem düzeyi ile SVM modelinde en belirleyici değişken olarak öne çıkmış ve ağırlık tahminlerinde kritik bir rol oynamıştır.

GÇ'yi takiben, SY, GD, ölçüm (Romanov kuzuların doğumdan itibaren 180. güne kadar 15 günlük zaman periyotları) ve CY değişkenleri de modele anlamlı katkı sağlamaktadır; ancak bunların önem düzeyi GÇ'ne kıyasla daha düşüktür. VU ve KAGG bağımsız değişkenleri modelde en düşük öneme sahiptir. Bu değişkenlerin katkısının sınırlı olması, SVM algoritmasında ağırlık tahminlerinde daha düşük bir etkiye sahip olduklarını göstermektedir.



Şekil 5.3. SVM bağımlı değişkenlerin önem düzeyleri

Şekil 5.4 SVM algoritması kullanılarak yapılan tahminler ile gerçek değerlerin karşılaştırılmasını göstermektedir.



Şekil 5.4. SVM algoritması kullanılarak tahmin değerlerinin gerçek ölçüm değerlerine göre dağılımı

Dağılım grafiği incelendiğinde, noktaların genel olarak doğrusal bir eğri etrafında yoğunlaştığı ve genel olarak tahmin edilen değerlerin gerçek değerlere oldukça yakın olduğu gözlemlenmektedir. Bu, SVM algoritmasının tahmin performansının güçlü olduğunu ve verileri başarılı bir şekilde modellediğini ortaya koymaktadır.

Ancak, grafikte bazı veri noktalarının gerçek değerlerden sapma gösterdiği, özellikle 10-15 birim aralığında daha fazla hata payının oluştuğu gözlemlenmiştir. Bu durum, algoritmanın belirli aralıklardaki veri noktalarında daha az doğru tahminler yaptığını göstermektedir. Tablodaki HKOK ve HKO değerleri incelendiğinde de bu sapmaların modelin hata oranını artırdığı görülmektedir.

SVM, genel olarak kabul edilebilir bir performans sergilemekle birlikte, diğer algoritmalara kıyasla (örneğin, RF ve GBM) biraz daha yüksek hata oranına sahiptir. Bu nedenle, modelin genel uyumu yüksek olsa da bazı veri noktalarında modelin daha düşük performans gösterdiği ve bu sapmaların tahmin doğruluğunu etkilediği söylenebilir.

Çizelge 5.5'de Romanov kuzularının ilk 180 gün boyunca 15 gün aralıklarla çeşitli vücut ölçümleri ve SVM algoritması kullanılarak yapılan tahmin değerleri gösterilmektedir.

Çizelge 5.5. SVM algoritması ile Romanov kuzuları ilk 180 gün tahmin değerleri

ölçüm	SY	CY	VU	GD	GÇ	KAGG	Ölçüm Değerleri	Tahmin Değerleri
1	26.20	25.10	23.80	8.00	27.50	5.30	1.90	1.80
15	34.10	33.40	30.40	11.90	38.60	9.00	4.40	4.89
30	39.30	38.80	33.30	14.00	45.10	10.20	5.20	6.88
45	42.10	41.80	36.50	15.50	50.20	11.30	6.00	8.61
60	42.80	42.60	37.20	15.80	51.80	11.70	7.20	9.21
75	43.50	43.30	37.90	16.40	53.00	11.90	8.70	9.82
90	44.00	43.80	38.20	17.00	54.00	12.00	9.70	10.18
105	45.20	45.00	40.40	17.40	57.30	12.10	12.20	12.30
120	46.40	46.20	41.70	19.30	61.60	12.30	13.60	14.20
135	47.80	47.60	42.60	20.40	64.40	12.30	15.80	16.22
150	49.10	49.00	43.10	20.30	68.20	12.60	17.90	18.58
165	50.80	50.50	44.60	22.40	71.00	12.90	20.20	21.01
180	53.90	53.60	49.80	23.00	74.20	13.30	22.90	24.82

SVM algoritması Romanov kuzularının büyüme süreçlerini tahmin etmede iyi bir model olarak kullanılabilir, ancak diğer algoritmalarla kıyaslandığında daha yüksek bir hata payına sahiptir. Bu yüzden, tahmin performansını artırmak için algoritmanın optimizasyonu göz önünde bulundurulmalı ya da farklı algoritmalarla kıyaslandığında daha uygun bir model seçimi yapılmalıdır.

5.4. Gradient Boosting Machine (GBM) Sonuçları

GBM modellemesinin optimum değerleri Python'daki Grid Search işlevi kullanılarak beş kat çapraz doğrulama tekniği ile elde edilmiştir. GBM algoritması için kullanılan parametre değerleri ve optimum sonuçlar Çizelge 5.6'da gösterilmiştir. Deneysel sonuçlar, GBM modeli için en iyi kombinasyonun learning_rate parametresinin 0.01, max_depth parametresinin 8, n_estimators parametresinin 1000 ve subsample parametresinin 0.5 olduğu durumda bulunmuştur.

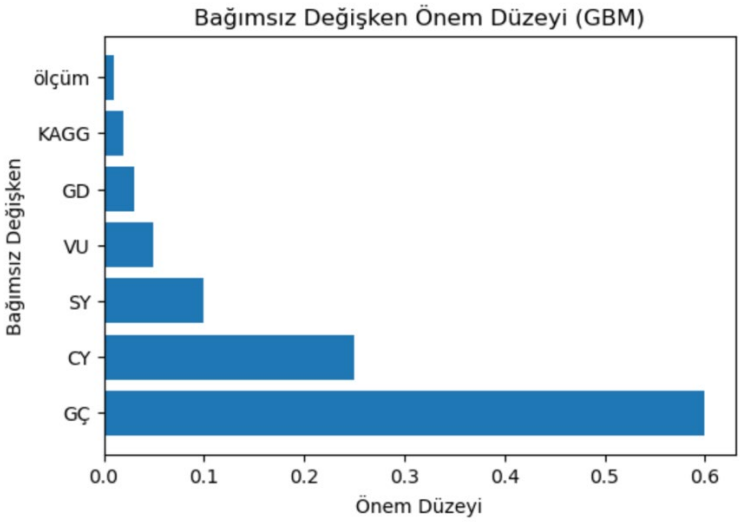
Çizelge 5.6. GBM algoritması seçilen hiperparametre değerleri

Parametreler	learning_rate	max_depth	n_estimators	subsample
Aralıklar	[0.001, 0.01, 0.1, 0.2]	[3, 5, 8, 50, 100]	[200, 500, 1000, 2000]	[1, 0.5, 0.75]
Seçilen Değerler	0.01	8	1000	0.5

Şekil 5.5'de GBM algoritması ile bağımsız değişkenlerin ağırlık tahminleri üzerindeki önem düzeyleri analiz edilerek görselleştirilmiştir. Değişkenlerin modele katkısı, barların uzunlukları ile ifade edilmiştir.

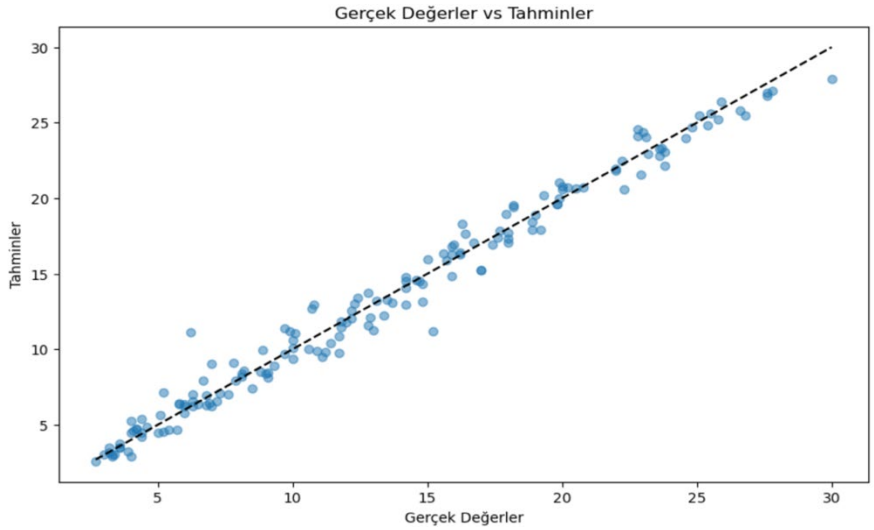
GBM modelinde göğüs çevresi, yaklaşık %60'lık bir önem düzeyi ile en kritik değişken olarak öne çıkmaktadır. Bu, Romanov kuzularının ağırlık tahmininde GÇ'nin belirleyici bir faktör olduğunu ve tahminlere en fazla katkıyı sağladığını göstermektedir. Bu bulgu, diğer algoritmalarda da benzer bir şekilde gözlemlenmiş olup, GÇ'nin tüm modellerde önemli bir değişken olduğunu doğrulamaktadır.

Diğer önemli değişkenler arasında CY öne çıkmaktadır, ancak GÇ'ye kıyasla çok daha düşük bir katkı sağladığı görülmektedir. SY ve VU değişkenleri ise daha düşük bir önem düzeyine sahiptir. Özellikle KAGG ve ölçüm (kuzuların doğumdan itibaren 180. güne kadar 15 günlük zaman periyotları) değişkenleri model üzerindeki katkısı oldukça düşük olup, modelin tahminlerinde etkili olmamaktadır.



Şekil 5.5. GBM bağımlı değişkenlerin önem düzeyleri

Şekil 5.6'de GBM algoritması kullanılarak yapılan tahminler ile gerçek değerlerin karşılaştırılmasını göstermektedir.



Şekil 5.6. GBM algoritması kullanılarak tahmin değerlerinin gerçek ölçüm değerlerine göre dağılımı

Dağılım grafiği incelendiğinde, noktalar genel olarak doğrusal bir çizgi etrafında toplanmış olup, tahmin edilen değerlerin büyük çoğunluğu gerçek değerlere oldukça yakındır. Bu durum, GBM algoritmasının güçlü bir modelleme performansına sahip olduğunu ve tahminlerinin yüksek doğrulukta olduğunu göstermektedir.

GBM algoritmasının diğer algoritmalarla karşılaştırıldığında en düşük HKOK ve HKO değerlerine sahip olduğu görülmektedir. Bu, modelin tahminlerinin hata oranının diğer modellere göre daha düşük olduğunu ve gerçek değerlere daha yakın sonuçlar ürettiğini gösterir. R^2 değeri ise modelin %98 oranında bağımlı değişkeni açıkladığını, yani modelin oldukça başarılı bir uyum sağladığını ortaya koymaktadır

Çizelge 5.7.'de Romanov kuzularının ilk 180 gün boyunca 15 gün aralıklarla çeşitli vücut ölçümleri ve GBM algoritması kullanılarak yapılan tahmin değerleri gösterilmektedir.

Çizelge 5.7. GBM algoritması ile Romanov kuzuları ilk 180 gün tahmin değerleri

ölçüm	SY	CY	VU	GD	GÇ	KAGG	Ölçüm Değerleri	Tahmin Değerleri
1	26.20	25.10	23.80	8.00	27.50	5.30	1.90	2.00
15	34.10	33.40	30.40	11.90	38.60	9.00	4.40	4.70
30	39.30	38.80	33.30	14.00	45.10	10.20	5.20	7.34
45	42.10	41.80	36.50	15.50	50.20	11.30	6.00	7.40
60	42.80	42.60	37.20	15.80	51.80	11.70	7.20	8.16
75	43.50	43.30	37.90	16.40	53.00	11.90	8.70	9.25
90	44.00	43.80	38.20	17.00	54.00	12.00	9.70	11.38
105	45.20	45.00	40.40	17.40	57.30	12.10	12.20	12.99
120	46.40	46.20	41.70	19.30	61.60	12.30	13.60	14.21
135	47.80	47.60	42.60	20.40	64.40	12.30	15.80	16.05
150	49.10	49.00	43.10	20.30	68.20	12.60	17.90	18.50
165	50.80	50.50	44.60	22.40	71.00	12.90	20.20	20.74
180	53.90	53.60	49.80	23.00	74.20	13.30	22.90	23.59

GBM algoritması hem düşük hata oranı hem de yüksek açıklayıcılık ile en başarılı model olarak öne çıkmaktadır. Özellikle Romanov kuzularının büyüme ölçümlerine dayalı tahminlerde yüksek doğrulukta sonuçlar vermektedir. Bu nedenle GBM, çalışmanızda tahmin performansını optimize etmek amacıyla güvenilir bir algoritma olarak kullanılabilir.

5.5. Extreme Gradient Boost (XGBoost) Sonuçları

XGBoost modellemesinin optimum değerleri Python'daki Grid Search işlevi kullanılarak beş kat çapraz doğrulama tekniği ile elde edilmiştir. XGBoost algoritması için kullanılan parametre değerleri ve optimum sonuçlar Çizelge 5.8'de gösterilmiştir. Deneysel sonuçlar, XGBoost modeli için en iyi kombinasyonun colsample_bytree 0.5, n_estimators parametresinin 800, max_depth parametresinin 4 ve learning_rate parametresinin 0.02 olduğu durumda bulunmuştur.

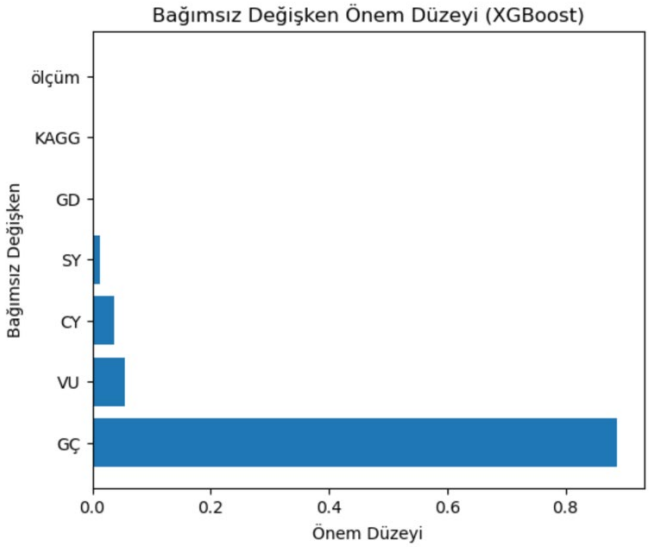
Çizelge 5.8. Xgboost algoritması seçilen hiperparametre değerleri

Parametreler	colsample_bytree	n_estimators	max_depth	learning_rate
Aralıklar	[0.4, 0.5, 0.6, 0.9, 1]	[800, 900, 1000, 2000, 3000]	[2, 3, 4, 5, 6, 7]	[0.1, 0.01, 0.02, 0.05]
Seçilen Değerler	0.5	800	4	0.02

Şekil 5.7'de Romanov kuzularının ilk 180 günlük büyüme verileri temel alınarak, XGBoost algoritması ile bağımsız değişkenlerin ağırlık tahminleri üzerindeki önem düzeyleri analiz edilerek görselleştirilmiştir. Değişkenlerin modele katkısı, barların uzunlukları ile ifade edilmiştir.

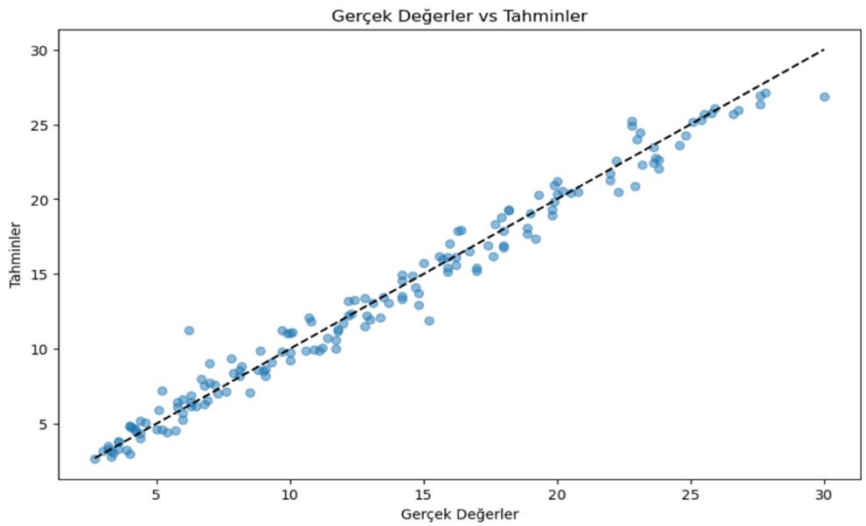
XGBoost algoritmasında, GÇ en önemli değişken olarak öne çıkmış ve diğer değişkenlere göre çok daha yüksek bir önem düzeyine sahip olduğu gözlemlenmiştir. Yaklaşık %80'lik bir önem düzeyi ile GÇ, Romanov kuzularının ağırlık tahmininde kritik bir rol oynamaktadır. Bu, modelin ağırlık tahminlerinde GÇ ölçümünün belirleyici bir faktör olduğunu bir kez daha kanıtlamaktadır.

Diğer değişkenler arasında ise VU, CY ve SY parametreleri daha düşük önem düzeylerine sahiptir. Bu değişkenlerin ağırlık tahminlerinde etkisi minimal düzeyde kalmıştır. GD, KAGG ve ölçüm (kuzuların doğumdan itibaren 180. güne kadar 15 günlük zaman periyotları) değişkenlerinin model üzerindeki etkisi çok düşüktür.



Şekil 5.7. XGBoost bağımlı değişkenlerin önem düzeyleri

Şekil 5.8.'de XGBoost algoritması kullanılarak yapılan tahminler ile gerçek değerlerin karşılaştırılmasını göstermektedir



Şekil 5.8. XGBoost algoritması kullanılarak tahmin değerlerinin gerçek ölçüm değerlerine göre dağılımı

Şekil 5.8'de tahmin edilen değerler ile gerçek değerlerin doğrusal bir eğri etrafında toplandığı ve modelin çoğunlukla başarılı tahminler yaptığı görülmektedir. Ancak bazı noktalar, gerçek değerlerden biraz sapma göstermektedir. Bu sapmalar, belirli verilerde XGBoost algoritmasının tahminlerinin tam olarak isabet etmediğini gösteriyor.

Çizelge 5.9'de Romanov kuzularının ilk 180 gün boyunca 15 gün aralıklarla çeşitli vücut ölçümleri ve XGBoost algoritması kullanılarak yapılan tahmin değerleri gösterilmektedir.

Çizelge 5.9. XGBoost algoritması ile Romanov kuzuları ilk 180 gün tahmin değerleri

ölçüm	SY	CY	VU	GD	GÇ	KAGG	Ölçüm Değerleri	Tahmin Değerleri
1	26.20	25.10	23.80	8.00	27.50	5.30	1.90	2.06
15	34.10	33.40	30.40	11.90	38.60	9.00	4.40	4.79
30	39.30	38.80	33.30	14.00	45.10	10.20	5.20	7.21
45	42.10	41.80	36.50	15.50	50.20	11.30	6.00	7.43
60	42.80	42.60	37.20	15.80	51.80	11.70	7.20	8.41
75	43.50	43.30	37.90	16.40	53.00	11.90	8.70	9.60
90	44.00	43.80	38.20	17.00	54.00	12.00	9.70	11.25
105	45.20	45.00	40.40	17.40	57.30	12.10	12.20	13.19
120	46.40	46.20	41.70	19.30	61.60	12.30	13.60	14.40
135	47.80	47.60	42.60	20.40	64.40	12.30	15.80	16.07
150	49.10	49.00	43.10	20.30	68.20	12.60	17.90	18.77
165	50.80	50.50	44.60	22.40	71.00	12.90	20.20	20.77
180	53.90	53.60	49.80	23.00	74.20	13.30	22.90	23.47

XGBoost, Romanov kuzularının büyüme ölçümlerine dayalı tahminlerde yüksek bir doğruluk oranı sergileyen, güçlü bir modelleme algoritmasıdır. Ancak diğer algoritmalarla (örneğin GBM ve RF) kıyaslandığında, hata oranı biraz daha yüksek çıkmaktadır. Bu nedenle XGBoost algoritması, kabul edilebilir bir performansa sahip olsa da, en iyi modelleme performansını elde etmek için diğer algoritmaların sonuçları da dikkate alınmalıdır.

5.6. LightGBM Sonuçları

LightGBM modellemesinin optimum değerleri Python'daki Grid Search işlevi kullanılarak beş kat çapraz doğrulama tekniği ile elde edilmiştir. LightGBM algoritması için kullanılan parametre değerleri ve optimum sonuçlar Çizelge 5.10'da gösterilmiştir. Deneysel sonuçlar, LightGBM modeli için en iyi kombinasyonun colsample_bytree 0.9, n_estimators parametresinin 100, max_depth parametresinin 3 ve learning_rate parametresinin 0.01 olduğu durumda bulunmuştur.

Çizelge 5.10. LightGBM algoritması seçilen hiperparametre değerleri

Parametreler	colsample_bytree	learning_rate	n_estimators	max_depth
Aralıklar	[0.4, 0.5, 0.6, 0.9, 1]	[0.01, 0.1, 0.5, 1]	[20, 40, 100, 200, 500, 1000, 2000]	[3, 4, 5, 6, 7, 8, 9, 10, 11]
Seçilen Değerler	0.9	0.1	100	3

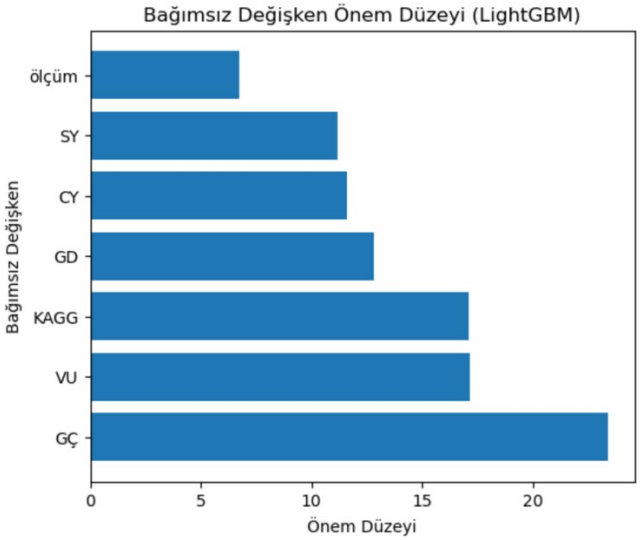
Şekil 5.9'da Romanov kuzularının ilk 180 günlük büyüme verileri temel alınarak, LightGBM algoritması ile bağımsız değişkenlerin ağırlık tahminleri üzerindeki önem düzeyleri analiz edilerek görselleştirilmiştir.

Diğer modellerle karşılaştırıldığı zaman, LightGBM algoritmasında değişkenler arasında daha dengeli bir dağılım gözlemlenmektedir. Ancak GÇ değişkeni, diğer değişkenlere göre en önemli katkıyı sağlayan faktör olarak öne çıkmaktadır ve yaklaşık %25'lik bir önem düzeyine sahiptir.

Diğer değişkenler arasında VU, KAGG ve GD değişkenleri de modelin tahminlerine anlamlı bir katkı sağlamaktadır. Bu değişkenler, LightGBM modelinde ağırlık tahminlerinde nispeten daha etkili olmuşlardır. CY ve SY değişkenleri de tahminlerde önemli bir rol oynamaktadır.

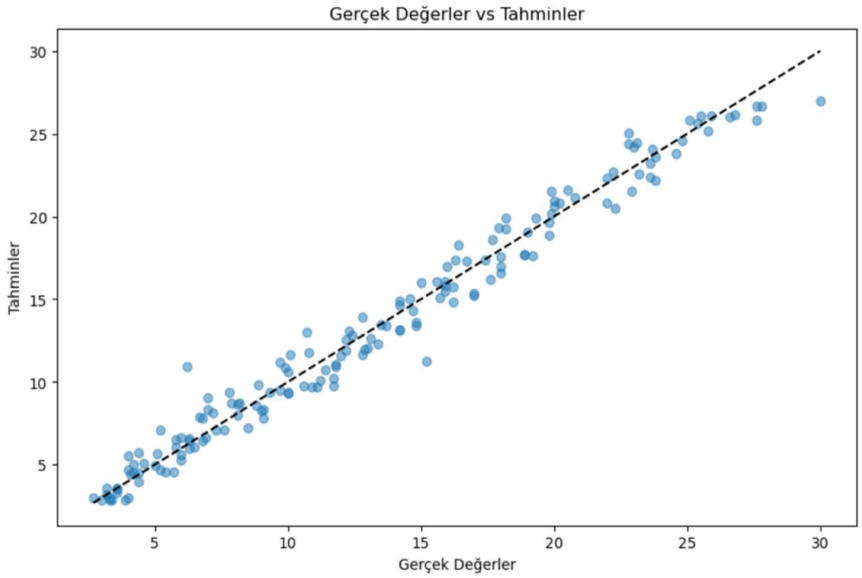
Bununla birlikte, ölçüm (kuzuların doğumdan itibaren 180. güne kadar 15 günlük zaman periyotları) değişkeni en düşük katkıya sahiptir, ancak yine de model üzerinde bir etkisi

bulunmaktadır. Bu durum, diğer algoritmalara kıyasla LightGBM modelinin daha fazla değişkenin tahminlerde etkili olmasına olanak tanıdığını göstermektedir.



Şekil 5.9. LightGBM bağımlı değişkenlerin önem düzeyleri

Şekil 5.10 LightGBM algoritması kullanılarak yapılan tahminler ile gerçek değerlerin karşılaştırılmasını göstermektedir



Şekil 5.10. LightGBM algoritması kullanılarak tahmin değerlerinin gerçek ölçüm değerlerine göre dağılımı

Şekil 5.10 incelendiğinde, tahmin edilen değerlerin büyük çoğunluğunun gerçek değerlerle uyumlu olduğu ve noktaların doğrusal bir eğri etrafında toplandığı gözlemlenmektedir. Bu durum, modelin genel tahmin başarısının iyi olduğunu ancak bazı veri noktalarında sapmaların bulunduğunu gösterir. Bu sapmalar, modelin belirli veri aralıklarında daha fazla hata yaptığını göstermektedir.

Çizelge 5.11'de Romanov kuzularının ilk 180 gün boyunca 15 gün aralıklarla çeşitli vücut ölçümleri ve LightGBM algoritması kullanılarak yapılan tahmin değerleri gösterilmektedir..

Çizelge 5.11. LightGBM algoritması ile Romanov kuzuları ilk 180 gün tahmin değerleri

ölçüm	SY	CY	VU	GD	GÇ	KAGG	Ölçüm Değerleri	Tahmin Değerleri
1	26.20	25.10	23.80	8.00	27.50	5.30	1.90	2.88
15	34.10	33.40	30.40	11.90	38.60	9.00	4.40	5.18
30	39.30	38.80	33.30	14.00	45.10	10.20	5.20	7.12
45	42.10	41.80	36.50	15.50	50.20	11.30	6.00	8.91
60	42.80	42.60	37.20	15.80	51.80	11.70	7.20	9.16
75	43.50	43.30	37.90	16.40	53.00	11.90	8.70	9.76
90	44.00	43.80	38.20	17.00	54.00	12.00	9.70	11.17
105	45.20	45.00	40.40	17.40	57.30	12.10	12.20	12.58
120	46.40	46.20	41.70	19.30	61.60	12.30	13.60	15.01
135	47.80	47.60	42.60	20.40	64.40	12.30	15.80	16.60
150	49.10	49.00	43.10	20.30	68.20	12.60	17.90	19.31
165	50.80	50.50	44.60	22.40	71.00	12.90	20.20	21.25
180	53.90	53.60	49.80	23.00	74.20	13.30	22.90	23.96

LightGBM, Romanov kuzularının büyüme ölçümlerine dayalı tahminlerde genel olarak iyi bir performans sergilemektedir. Ancak diğer algoritmalarla kıyaslandığında (örneğin GBM ve RF), hata oranının biraz daha yüksek olduğu ve tahmin doğruluğunun biraz daha düşük olduğu gözlemlenmektedir.

Yukarda verilen çalışmanın sonuçlarına bakılarak, literatürdeki benzer çalışmalar incelendiğinde, Setiawan ve ark (2024) büyükbaş hayvan ağırlığı tahminine yönelik çalışmalarında XGBoost algoritması Ortalama Mutlak Hata (OMH) = 1.08 ve R^2 = 0.99 gibi yüksek doğruluk değerleriyle en iyi sonuçları vermiştir. Bu çalışmasında ise

XGBoost, HKOK = 1.03964, OMH = 1.08085 ve R^2 = 0.97842 değerleri ile başarılı bir performans sergilemekle birlikte, Setiawan ve ark (2024) bulgularına kıyasla daha düşük bir R^2 değerine sahiptir. Bu fark, veri setindeki farklılıklar ve hayvan türüne (küçükbaş vs. büyükbaş) bağlı olabilir. Bu çalışmada GBM algoritması ise HKOK = 0.99534, OMH = 0.99131 ve R^2 = 0.98022 ile XGBoost'tan daha iyi sonuçlar vermiştir.

Literatürde başka bir benzer çalışma olan Coşkun ve ark (2023) Anadolu Merinos kuzuları üzerine yaptıkları araştırmada, XGBoost algoritması HKOK = 1.492 ve R^2 = 0.946 değerleri ile en başarılı algoritma olarak öne çıkmıştır. Bu çalışmasında ise XGBoost, HKOK = 1.03964, HKO = 1.08085 ve R^2 = 0.97842 değerleri ile daha yüksek bir doğruluk oranı sağlamış ve Coşkun ve arkadaşlarının çalışmasına kıyasla daha iyi sonuçlar vermiştir fakat bu çalışmada en iyi performansı sergileyen algoritma XGBoost değil HKOK = 0.99534, HKO = 0.99131 ve R^2 = 0.98022 değerleri ile GBM algoritması olmuştur. Sonuç olarak, Coşkun ve arkadaşlarının (2023) çalışmasında ve bu tez çalışmasında performanları bakımından XGBoost önemli sonuçlar sunmuştur fakat bu çalışmada GBM'nin en başarılı model olduğu görülmektedir.

Literatürdeki çalışmalara bakıldığı zaman XGBoost güçlü bir algoritma olarak öne çıksa da, GBM'in de belirli veri setlerinde etkili bir algoritma olduğu görülmektedir; bu durum, her iki algoritmanın da hayvan ağırlığı tahmininde hayvan türü ve veri setinin özelliklerine bağlı olarak algoritmaların performanslarının değişebileceğini göstermektedir.

6.SONUÇ VE ÖNERİLER

Bu çalışma, Romanov kuzularının büyüme ölçümleri ve ağırlıkları kullanılarak, farklı makine öğrenmesi algoritmalarının ağırlık tahmini üzerindeki performanslarını incelemiştir. Elde edilen bulguların sonucunda GBM algoritmasının en düşük hata oranına (HKOK: 0.99534, HKO: 0.99131) ve en yüksek doğruluk oranına (R^2: 0.98022) sahip olduğunu göstermektedir. Bu bulgu, GBM algoritmasının Romanov kuzularının ağırlık tahmininde en başarılı model olduğunu ortaya koymaktadır.

Çalışmada kullanılan algoritmalar arasında, RF algoritması dikkate değer bir doğrulukla tahmin yapmış, HKOK değeri 1.01365, HKO değeri 1.02749 ve R^2 değeri 0.97948 olarak hesaplanmıştır. RF algoritması, özellikle göğüs çevresi ve cidago yüksekliği gibi bağımsız değişkenlere duyarlılık göstererek modelin tahmin doğruluğuna anlamlı bir katkı sağlamıştır.

XGBoost algoritması ise HKOK değeri 1.03964, HKO değeri 1.08085 ve R^2 değeri 0.97842 ile RF algoritmasına yakın bir performans sergilemiş, GBM'e göre daha yüksek hata oranına sahip olmasına rağmen güvenilir bir tahmin modeli olarak öne çıkmıştır. XGBoost'un hayvan ağırlık tahmini uygulamalarında güçlü bir modelleme kapasitesine sahip olduğu, diğer çalışmalarda da benzer şekilde gözlemlenmiştir.

SVM ve LightGBM algoritmaları, diğer modellere göre daha yüksek hata oranlarıyla tahmin yapmışlardır. SVM algoritması, HKOK değeri 1.09043 ve HKO değeri 1.18903 ile en yüksek hata oranına sahip olup, R^2 değeri 0.97626 olarak hesaplanmıştır. LightGBM algoritması da benzer bir hata oranına sahip olup, HKOK değeri 1.08735, HKO değeri 1.18233 ve R^2 değeri 0.97639 olarak bulunmuştur. Bu algoritmalar, daha düşük doğruluk oranlarıyla tahmin yapmalarına rağmen, göğüs çevresi gibi belirli değişkenlerin ağırlık tahminindeki önemini vurgulamakta ve model uyumunda diğer algoritmalara göre daha az başarı göstermektedirler.

Çalışmada kullanılan bağımsız değişkenler arasında, göğüs çevresi, tüm modellerde ağırlık tahmini üzerinde en yüksek öneme sahip değişken olarak öne çıkmıştır. Göğüs çevresini takiben, cidago yüksekliği ve sağrı yüksekliği gibi değişkenlerin de tahmin doğruluğuna katkı sağladığı görülmüştür. Bu durum, makine öğrenmesi algoritmalarının her birinin bağımsız değişkenlere duyarlılığını doğrulamakta olup, modellerin tahmin doğruluğunu artırmak için bu değişkenlerin dikkate alınması gerektiğini ortaya koymaktadır

Makine öğrenmesi algoritmalarının hayvan ağırlık tahminindeki üstün performansı dikkate alındığında, bu teknolojilerin tarım ve hayvancılık sektöründe daha geniş çapta uygulanması önerilmektedir. Özellikle, veri toplama ve analiz süreçlerinin standardizasyonu büyük önem arz etmektedir. Yetiştiricilik faaliyetlerinde, büyüme performansı ve diğer kritik parametrelerin sürekli izlenmesi için düzenli ve sistematik veri toplama süreçlerinin oluşturulması gerekmektedir. Bu verilerin etkin bir şekilde analiz edilmesi amacıyla gelişmiş yazılımlar ve algoritmaların kullanımı teşvik edilmelidir.

Ek olarak, bu algoritmaların entegre edildiği özel yazılım ve uygulamaların geliştirilmesiyle daha verimli ve faydalı sonuçlar elde edilebilir. Bu tür uygulamalar, yetiştiricilere gerçek zamanlı veri analizi ve hızlı karar alma yeteneği sunarak hayvan sağlığı ve verimliliğinin artırılmasına katkı sağlayacaktır. Özelleştirilmiş yazılımlar aracılığıyla, çiftlik yönetimi süreçleri otomatikleştirilebilir ve kaynak kullanımı optimize edilebilir.

Bu öneriler doğrultusunda, makine öğrenmesi algoritmalarının hayvan ağırlık tahmini üzerine kullanımı ile tarım ve hayvancılık sektöründe önemli ilerlemeler kaydedilebilecektir.

KAYNAKLAR

Akçapınar, H., 2000. Koyun Yetiştiriciliği (2. baskı). İsnat Matbaacılık, 278 s, Ankara.

Akçapınar, H. ve Özbeyaz, C., 1999. Hayvan Yetiştiriciliği Temel Bilgileri (1. baskı). Kariyer Matbaacılık, 312 s, Ankara.

Altınel, A., Evrim, M., Özcan, M., Başpınar, H. ve Deligözoğlu, F., 1998. Sakız, Kıvırcık ve Alman Siyah Başlı Koyun Irkları Arasındaki Melezlemeler ile Kaliteli Kesim Kuzuları Elde Etme Olanaklarının Araştırılması. Turkish Journal of Veterinary and Animal Sciences, 22 (3), 257-265.

Alpaydın, E., 2014. Introduction to Machine Learning. The MIT Press, 640 sayfa, Cambridge, MA, ABD

Arslan, İ., 2019. Python ile Veri Bilimi. Pusula Yayıncılık, 432 s, İstanbul.

Arslan, Y., 2020. LightGBM algoritması ile yeni bir satış tahmin modelinin oluşturulması ve perakende sektörüne uygulanması. (Yüksek Lisans Tezi), İstanbul Aydın Üniversitesi Lisansüstü Eğitim Enstitüsü, İstanbul.

Breiman, L., 2001. Random forests. Machine Learning, 45 (1), 5-32.

Breiman, L., Friedman, J., Stone, C. J. ve Olshen, R. A., 1984. Classification and Regression Trees. CRC Press, 368 s, ABD.

Budak, H. ve Erpolat, S., 2012. Kredi riski tahmininde yapay sinir ağları ve lojistik regresyon analizi karşılaştırılması. AJIT-e: Bilişim Teknolojileri Online Dergisi, 3 (9), 23-30.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. ve Wirth, R., 2000. CRISP-DM 1.0: Step-by-step data mining guide. SPSS Inc.

Chen, D. Y., 2017. Pandas for everyone: Python data analysis. Addison-Wesley Professional. https://pandas.pydata.org/getting_started.html (Erişim tarihi: 26 Kasım 2022).

Chen, T. ve 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.

Chen, T. ve He, T., 2015. Higgs boson discovery with boosted trees. JMLR Workshop and Conference Proceedings, 42, 69-80.

Cortes, C. ve Vapnik, V., 1995. Support-vector networks. Machine Learning, 20 (3), 273-297.

Coşkun, G., Şahin, Ö., Altay, Y. ve Aytekin, İ., 2023. Final fattening live weight prediction in Anatolian Merinos lambs from some body characteristics at the initial of fattening by using some data mining algorithms. Black Sea Journal of Agriculture, 6 (1), 47-53. https://doi.org/10.47115/bsagriculture.1181444.

Coşkun, G., Şahin, Ö., Altay, Y. ve Aytekin, İ., 2022. Farklı veri madenciliği algoritmaları kullanılarak Anadolu Merinos kuzularında besi başlangıcındaki bazı vücut özelliklerinden besi sonu canlı ağırlığın tahmini. Türk Tarım ve Doğa Bilimleri Dergisi, 9 (3), 470-476.

Cruz, A. A., Silva, B. B., & Oliveira, C. D., 2015. *Advanced Machine Learning Techniques*. Academic Press, 350 s, Londra, Birleşik Krallık.

Çakmakçı, C., 2022. Live weight prediction in Norduz sheep using machine learning algorithms. Turkish Journal of Agriculture - Food Science and Technology, 10(4), 587-594.

Cristianini, N. ve Taylor, J. S., 2000. An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, 189 s, Birleşik Krallık.

Dang, C., Choi, T., Lee, S., Lee, S., Alam, M., Park, M., Han, S., Lee, J., & Hoang, D., 2021. Machine learning-based live weight estimation for Hanwoo cow. Animals, 11(8), 2295. https://doi.org/10.3390/ani11082295

Dalton, D. C., Knight, T. W. ve Johnson, D. L., 1980. Lamb survival in sheep breeds on New Zealand hill country. New Zealand Journal of Agricultural Research, 23 (2), 167-173.

Dixit, A., 2017. Ensemble Machine Learning. Packt Publishing Ltd, 438 s, Birmingham, Birleşik Krallık.

Doğan, K., & Arslantekin, S., 2023. Veri madenciliğinin önemi ve kütüphanelerde kullanımı. Dil ve Tarih-Coğrafya Fakültesi Dergisi, 63(1), 503-541. https://doi.org/10.33171/dtcfjournal.2023.63.1.21

Efe, E., 1990. Büyüme Eğrileri. (Doktora Tezi), Çukurova Üniversitesi Fen Bilimleri Enstitüsü Zootekni Anabilim Dalı, Adana.

Friedman, J., 2002. Stochastic gradient boosting. Computational Statistics & Data Analysis, 38 (4), 367-378.

Friedman, J. H., 2001. Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29 (5), 1189-1232.

Guolin, K., et al., 2017. A scalable tree boosting system. Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 135-144.

Gomez-Vazquez, A., Tırınk, C., Cruz-Tamayo, A. A., Cruz-Hernandez, A., Camacho-Pérez, E., Okuyucu, İ. C., Şahin, H. A., Dzib-Cauich, D. A., Gülboy, Ö., Garcia-Herrera, R. A., & Chay-Canul, A. J., 2024. Prediction of body weight by using PCA-supported gradient boosting and random forest algorithms in water buffaloes (*Bubalus bubalis*) reared in south-eastern Mexico. Animals, 14(2), 293. https://doi.org/10.3390/ani14020293

Géron, A., 2019. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, 856 s, ABD.

Hamadani, A., & Ganai, N. A., 2023. Artificial intelligence algorithm comparison and ranking for weight prediction in sheep. *Scientific Reports*, 13, 13242. https://doi.org/10.1038/s41598-023-40528-4

Hastie, T., Tibshirani, R. ve Friedman, J., 2009. The Elements of Statistical Learning. Springer, 745 s, ABD.

Huma, Z. E. ve Iqbal, F., 2019. Predicting the body weight of Balochi sheep using a machine learning approach. Turkish Journal of Veterinary & Animal Sciences, 43 (4), 500-506. https://doi.org/10.3906/vet-1812-23.

Karaoğlu, M. ve Emsen, H., 2000. Zootekni Bölümü ve Doğu Anadolu Bölgesi'nde yapılan çalışmalar. Atatürk Üniversitesi Ziraat Fakültesi Dergisi, 31 (Özel Sayı), 37-47.

Kaymakçı, M. ve Sönmez, R., 1996. İleri Koyun Yetiştiriciliği. Ege Üniversitesi Basım Evi, 520 s, Bornova-İzmir.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. ve Liu, T.-Y., 2017. LightGBM: A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems, 30, 3146-3154.

Koç, A., ve Akman, N., 2007. Siyah-Alaca tosunların değişik dönemlerdeki vücut ölçüleri ve vücut ölçülerinden canlı ağırlığın tahmini. Adnan Menderes Üniversitesi Ziraat Fakültesi Dergisi, 4 (1/2), 21-25.

Liang, Y. C., Maimury, Y., Chen, A. H. L. ve Juarez, J. R. C., 2020. Machine learning-based prediction of air quality. Applied Sciences, 10 (24), 9151.

Metlek, S. ve Kayaalp, K., 2020. Makine Öğrenmesinde Teoriden Örnek Matlab Uygulamalarına Kadar Destek Vektör Makineleri. İksad Yayınevi, 104 s, Türkiye.

Morde, V., 2019. XGBoost algorithm: Long she may rein. Towards Data Science. https://towardsdatascience.com/https-mediumcom-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d (Erişim tarihi: 27 Mayıs 2019).

Natekin, A. ve Knoll, A., 2013. Gradient boosting machines, a tutorial. Frontiers in Neurorobotics, 7 (21), 1-21.

Nagalla, R., Pothuganti, P. ve Pawar, D. S., 2017. Analyzing gap acceptance behavior at unsignalized intersections using support vector machines, decision tree and random forests. Procedia Computer Science, 109, 474-481.

Patrous, Z. S., 2018. Evaluating XGBoost for user classification by using behavioral features extracted from smartphone sensors. (Yüksek Lisans Tezi), KTH Royal Institute of Technology, School of Computer Science and Communication, İsveç.

Rebala, G., Ravi, A., & Churiwala, S., 2019. An Introduction to Machine Learning. Springer, 263 s, Cham, İsviçre.

Quinlan, J. R., 1986. Induction of decision trees. Machine Learning, 1 (1), 81-106.

Setiawan, A., Utami, E. ve Ariatmanto, D., 2024. Cattle body weight prediction using regression machine learning. Jurnal Teknik Informatika (JUTIF), 5 (2), 509-518. https://doi.org/10.52436/1.jutif.2024.5.2.1521.

Singh, A., 2019. Comprehensive guide for ensemble models. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/ (Erişim tarihi: 1 Temmuz 2019).

Smola, A. J. ve Schölkopf, B., 2004. A tutorial on support vector regression. Statistics and Computing, 14 (3), 199-222.

Şengonca, M. ve Gücük, T., 1991. Yerli Merinos koyunlarında bazı vücut ölçülerinden canlı ağırlığın tahmini olanakları. Uludağ Üniversitesi Ziraat Fakültesi Dergisi, 8, 1-8.

Tahtalı, Y. ve Yaldızbaş, A. T., 2020. Romanov kuzularının vücut özelliklerinin tanımlanmasında doğrusal ve doğrusal olmayan modellerin karşılaştırılması. Turkish Journal of Agriculture - Food Science and Technology, 8 (2), 499-503. https://doi.org/10.24925/turjaf.v8i2.499-503.3342.

Tırınk, C., Tosun, R., Saftan, M., Kaya, E. ve Atalay, A. İ., 2022. Morkaraman kuzularında vücut ölçümleriyle doğum ağırlığının CART algoritması ile tahmini. Large Animal Review, 28, 187-192.

Topal, M., & Macit, M., 2004. Prediction of body weight from body measurements in Morkaraman sheep. *Journal of Applied Animal Research*, 25(2), 97-100. https://doi.org/10.1080/09712119.2004.9706484

Tuzcu, S., 2020. Çevrimiçi kullanıcı yorumlarının duygu analizi ile sınıflandırılması. Eskişehir Türk Dünyası Uygulama ve Araştırma Merkezi Bilişim Dergisi, 1 (2), 1-5.

Uyanık, T., 2021. Ticari gemilerde operasyonel elektriksel gücün tahmininde makine öğrenmesi yaklaşımı: şaft jeneratörü güç tahmini uygulaması. Akıllı Ulaşım Sistemleri ve Uygulamaları Dergisi, 4(2), 165-174. https://doi.org/10.51513/jitsa.993058

Vapnik, V., 1999. The Nature of Statistical Learning Theory. Springer Science & Business Media, 314 s, ABD.

Wang, W. ve Xu, Z., 2004. A heuristic training for support vector regression. Neurocomputing, 61, 259-275. https://doi.org/10.1016/j.neucom.2003.11.012.

Yaldızbaş, A. T., 2016. Romanov kuzularında değişik vücut ölçüleri bakımından büyüme eğrisinin doğrusal ve doğrusal olmayan modeller ile belirlenmesi. (Yüksek Lisans Tezi), Gaziosmanpaşa Üniversitesi Fen Bilimleri Enstitüsü, Tokat, Türkiye.

Yılmaz, A. ve Altınel, A., 2003. Alman Siyah Başlı Etçi Irkının Baba Hattı Olarak Kullanılmasıyla Elde Edilen Üçlü Kullanma Melezi Kuzular ile Kıvırcık ve Türk Merinosu Kuzuların Büyüme Hızı ve Yaşama Gücü Özellikleri. İstanbul Üniversitesi Veteriner Fakültesi Dergisi, 29 (2), 213-219.

Zhang, C. ve Ma, Y. (Ed.), 2012. Ensemble Machine Learning: Methods and Applications. Springer, 332 s, ABD.

Zimmermann, M. J., Kuehn, L. A., Spangler, M. L., Thallman, R. M., Warren, M., Snelling, W. M. ve Lewis, M. L., 2019. Comparison of different functions to describe growth from weaning to maturity in crossbred beef cattle. Journal of Animal Science, 97 (4), 1523-1533.

Anonim, 2020. Scikit-learn Nedir? Karabay Yazılım. https://www.karabayyazilim.com/blog/python/scikit-learn-nedir-2020-02-12-062241 (Erişim tarihi: 27 Kasım 2022).

Anonim, 2022. Matplotlib: Visualization with Python. https://matplotlib.org/ (Erişim tarihi: 27 Kasım 2022).

EKLER

Ek 1. GBM Algoritma Kodları

[1]:
```
#Kütüphane içe aktarma
import numpy as np
import pandas as pd

from warnings import filterwarnings
filterwarnings('ignore')
```

[2]:
```
#veriyi içe aktarma
ad = pd.read_excel("tez_kuzu_verisi_kopyası.xlsx")
#veri kopyası oluşturma
df = ad.copy()
#veride ilk 5 gösterme
df.head()
```

•••

[3]:
```
# x(bağımsız değişken) ve y ( bağımlı değişken)
X = df.drop("agırlık", axis = 1 )
y = df["agırlık"]
```

[4]:
```
#makine  öğrenmesi test verisi ayırma
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split ( X, y , test_size = 0.25, random_state= 42)
```

Name: Untitled Folder
Path: Desktop/tez çalışma/tez çalışma
Created: 2024-06-28 17:21:43
Modified: 2024-06-28 17:21:42
Writable: true

[5]:
```
#xgboost kütüphane import
from sklearn.ensemble import GradientBoostingRegressor
#model kurma
gbm_model = GradientBoostingRegressor().fit(X_train, y_train)
```

[6]:
```
#Model tahmin
y_pred = gbm_model.predict(X_test)
```

[8]:
```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBRegressor

# XGBoost modeli için hiperparametre grid'i
gbm_grid = {
    'learning_rate': [0.001, 0.01, 0.1, 0.2],
    'max_depth': [3, 5, 8,50,100],
    'n_estimators': [200, 500, 1000, 2000],
    'subsample': [1,0.5,0.75],
}

# GridSearchCV objesini oluşturma
gbm = GradientBoostingRegressor()

gbm_cv = GridSearchCV(gbm,
                      param_grid = gbm_grid,
                      cv = 5,
                      n_jobs = -1,
                      verbose = 2)


# Modeli eğitim verisi üzerinde fit etme
gbm_cv.fit(X_train, y_train)
```

Jun 28, 2024 5:21 PM

•••

[9]:
```
gbm_cv.best_params_
```

•••

[10]:
```
y_pred = gbm_cv.predict(X_test)
```

[12]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
#hiper parametre ayarlanmış sonuçlar
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Ortalama Mutlak Hata (MAE): {mae:.5f}")
print(f"Ortalama Kare Hata (MSE): {mse:.5f}")
print(f"Kök Ortalama Kare Hata (RMSE): {rmse:.5f}")
print(f"R-kare: {r2:.5f}")
```

[14]:
```
# İstenen indeksleri seçme
index = [0,1,2,3,4,5,6,7,8,9,10,11,12]

# Seçilen indekslerin bağımsız değişken değerlerini alma
X_selected_1 = X.loc[index]
y_pred_selected_1 = gbm_cv.predict(X_selected_1)
comparison_table = X_selected_1.copy()
comparison_table['Ölçüm Değerleri'] = y.loc[index]
comparison_table['Tahmin Değerleri'] = y_pred_selected_1
display(comparison_table.style.set_precision(2))
```

[15]:
```
#hiperparametleri ayarlanmış gerçek tahmini değer grafik
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel('Gerçek Değerler')
plt.ylabel('Tahminler')
plt.title('Gerçek Değerler vs Tahminler')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--')  # Mükemmel tahmin hattı
plt.show()
```

[16]:
```
# Özellik önem düzeylerini alma
gbm_importance =gbm_model.feature_importances_

# Görselleştirme
gbm_importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': gbm_importance
}).sort_values(by='Importance', ascending=False)

plt.figure(figsize=(5, 4))
plt.barh(gbm_importance_df['Feature'], gbm_importance_df['Importance'])
plt.xlabel('Önem Düzeyi')
plt.ylabel('Bağımlı Değişken')
plt.title('Bağımlı Değişken Önem Düzeyi(GBM) ')
plt.show()

import ace_tools as tools; tools.display_dataframe_to_user(name="Comparison Table", dataframe=comparison_table)
```

Ek 2. LightGBM Algoritma Kodları

[1]:
```
#Kütüphane içe aktarma
import numpy as np
import pandas as pd

from warnings import filterwarnings
filterwarnings('ignore')
```

[2]:
```
#veriyi içe aktarma
ad = pd.read_excel("tez_kuzu_verisi_kopyası.xlsx")
#veri kopyası oluşturma
df = ad.copy()
#veride ilk 5 gösterme
df.head()
```

[3]:
```
# x(bağımsız değişken) ve y ( bağımlı değişken)
X = df.drop("agırlık", axis = 1 )
y = df["agırlık"]
```

[4]:
```
#makine  öğrenmesi test verisi ayırma
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split ( X, y , test_size = 0.25, random_state= 42)
```

Jun 28, 2024 5:21 PM

[5]:
```
#lightgbm import
from lightgbm import LGBMRegressor
#model kurma
light_model = LGBMRegressor().fit(X_train, y_train)
```

[6]:
```
#Model tahmin
y_pred = light_model.predict(X_test)
```

[7]:
```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBRegressor

# XGBoost modeli için hiperparametre grid'i
lgbm_grid = {
    'colsample_bytree': [0.4, 0.5,0.6,0.9,1],
    'learning_rate': [0.01, 0.1, 0.5,1],
    'n_estimators': [20, 40, 100, 200, 500,1000,2000],
    'max_depth': [3,4,5,6,7,8,9,10,11]
}

# GridSearchCV objesini oluşturma
lgbm = LGBMRegressor()

lgbm_cv = GridSearchCV(lgbm,
                       param_grid = lgbm_grid,
                       cv = 5,
                       n_jobs = -1,
                       verbose = 2)


# Modeli eğitim verisi üzerinde fit etme
lgbm_cv.fit(X_train, y_train)
```

•••

[8]:
```
lgbm_cv.best_params_
```

•••

[9]:
```
y_pred = lgbm_cv.predict(X_test)
```

[10]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
#hiper parametre ayarlanmış sonuçlar

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Ortalama Mutlak Hata (MAE): {mae:.5f}")
print(f"Ortalama Kare Hata (MSE): {mse:.5f}")
print(f"Kök Ortalama Kare Hata (RMSE): {rmse:.5f}")
print(f"R-kare: {r2:.5f}")
```

```
Ortalama Mutlak Hata (MAE): 0.85604
Ortalama Kare Hata (MSE): 1.18233
Kök Ortalama Kare Hata (RMSE): 1.08735
R-kare: 0.97639
```

[11]:
```
# İstenen indeksleri seçme
index = [0,1,2,3,4,5,6,7,8,9,10,11,12]

# Seçilen indekslerin bağımsız değişken değerlerini alma
X_selected_1 = X.loc[index]
y_pred_selected_1 = lgbm_cv.predict(X_selected_1)
comparison_table = X_selected_1.copy()
comparison_table['Ölçüm Değerleri'] = y.loc[index]
comparison_table['Tahmin Değerleri'] = y_pred_selected_1
display(comparison_table.style.set_precision(2))
```

•••

[15]:
```
import matplotlib.pyplot as plt
import pandas as pd

# Özellik isimleri
features = ['ölçüm', 'sağrı yük.', 'cidago yük', 'vücut uzun.', 'göğüs derinliği', 'göğ. Çevresi', 'kürekler.aras.gnş.']

# Önem puanları
importance = [119, 198, 206, 304, 227, 415, 303]

# Önem puanlarını toplamına bölerek yüzde hesaplama
total_importance = sum(importance)
importance_df = pd.DataFrame({
    'Feature': features,
    'Importance': importance
}).sort_values(by='Importance', ascending=False)

importance_df['Percentage Importance'] = (importance_df['Importance'] / total_importance) * 100 /100

plt.figure(figsize=(5, 4))
plt.barh(importance_df['Feature'], importance_df['Percentage Importance'])
plt.xlabel('Önem Düzeyi')
plt.ylabel('Bağımlı Değişken')
plt.title('Bağımlı Değişken Önem Düzeyi (LightGBM)')
plt.show()
```

•••

Ek 3. RF Algoritma Kodları

[10]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor

# RandomForestRegressor için hiperparametre grid'i
rf_params = {'max_depth': list(range(1,10)),
             'max_features': [3,5,10,15],
             'n_estimators' : [ 1000,2000,3000]}

rf_model = RandomForestRegressor(random_state = 42)

# GridSearchCV objesini oluşturma
rf_cv_model = GridSearchCV(rf_model,
                           rf_params,
                           cv = 5,
                            n_jobs = -1)

# Modeli eğitim verisi üzerinde fit etme
rf_cv_model.fit(X_train, y_train)
```

•••

[11]:
```
rf_cv_model.best_params_
```

•••

[20]:
```
y_pred = rf_cv_model.predict(X_test)
```

[14]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Ortalama Mutlak Hata (MAE): {mae:.5f}")
print(f"Ortalama Kare Hata (MSE): {mse:.5f}")
print(f"Kök Ortalama Kare Hata (RMSE): {rmse:.5f}")
print(f"R-kare: {r2:.5f}")
```

•••

[1]:
```
#Kütüphane içe aktarma
import numpy as np
import pandas as pd

from warnings import filterwarnings
filterwarnings('ignore')
```

[3]:
```
#veriyi içe aktarma
ad = pd.read_excel("tez_kuzu_verisi_kopyası.xlsx")
#veri kopyası oluşturma
df = ad.copy()
#veride ilk 5 gösterme
df.head()
```

•••

[4]:
```
# x(bağımsız değişken) ve y ( bağımlı değişken)
X = df.drop("agırlık", axis = 1 )
y = df["agırlık"]
```

[5]:
```
#veri görselleştirme kütüphane import
import matplotlib.pyplot as plt
```

[6]:
```
#makine  öğrenmesi test verisi ayırma
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split ( X, y , test_size = 0.25, random_state= 42)
```

[22]:
```
#makine öğrenmesi model import
from sklearn.ensemble import RandomForestRegressor
#model eğitme
rf_model = RandomForestRegressor(random_state = 42)
rf_model.fit(X_train, y_train)
```

•••

[23]:
```
#Model tahmin
y_pred = rf_model.predict(X_test)
```

[15]:
```
# İstenen indeksleri seçme
index = [0,1,2,3,4,5,6,7,8,9,10,11,12]

# Seçilen indekslerin bağımsız değişken değerlerini alma
X_selected_1 = X.loc[index]
y_pred_selected_1 = rf_cv_model.predict(X_selected_1)
comparison_table = X_selected_1.copy()
comparison_table['Ölçüm Değerleri'] = y.loc[index]
comparison_table['Tahmin Değerleri'] = y_pred_selected_1
display(comparison_table.style.set_precision(2))
```

•••

[16]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel('Gerçek Değerler')
plt.ylabel('Tahminler')
plt.title('Gerçek Değerler vs Tahminler')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--')  # Mükemmel tahmin hattı
plt.show()
```

•••

[24]:
```
# Özellik önem düzeylerini alma
rf_importance = rf_model.feature_importances_

# Görselleştirme
rf_importance_df = pd.DataFrame({
    'Feature': X_train.columns,
    'Importance': rf_importance
}).sort_values(by='Importance', ascending=False)

plt.figure(figsize=(5, 4))
plt.barh(rf_importance_df['Feature'], rf_importance_df['Importance'])
plt.xlabel('Önem Düzeyi')
plt.ylabel('Bağımlı Değişken')
plt.title('Bağımlı Değişken Önem Düzeyi (RF) ')
plt.show()
```

Name: Untitled Folder
Path: Desktop/tez çalışma/tez çalışma kodlar
Created: 2024-06-28 17:21:43
Modified: 2024-06-28 17:21:42
Writable: true

•••

Ek 4. SVM Algoritma Kodları

[1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
from IPython.display import display
import numpy as np
```

[2]:
```
#veriyi içe aktarma
ad = pd.read_excel("tez_kuzu_verisi_kopyası.xlsx")
#veri kopyası oluşturma
df = ad.copy()
#veride ilk 5 gösterme
df.head()
```

•••

[3]:
```
X = df.drop("agırlık", axis = 1 )
y = df["agırlık"]
```

[4]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
scaler = StandardScaler()
X_train_norm = scaler.fit_transform(X_train)
X_test_norm = scaler.transform(X_test)
```

[5]:
```
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': ['scale', 'auto', 0.01, 0.1, 1, 10],
    'epsilon': [0.01, 0.1, 1]
}
```

[6]:
```
#GridSearchCV objesini oluşturma
svr_cv = GridSearchCV(SVR(), param_grid, cv=5, scoring='neg_mean_squared_error', verbose=2, n_jobs=-1)

# Grid Search'i çalıştırma
svr_cv.fit(X_train_norm, y_train)

# En iyi hiperparametreleri alma
best_params = svr_cv.best_params_
print("En iyi hiperparametreler: ", best_params)

# En iyi parametrelerle SVR modelini oluşturma
best_svr_model = SVR(C=best_params['C'], gamma=best_params['gamma'], epsilon=best_params['epsilon'])
```

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
En iyi hiperparametreler:  {'C': 100, 'epsilon': 0.1, 'gamma': 0.1}
```

[7]:
```
# Modeli normalleştirilmiş eğitim verisi üzerinde eğitme
best_svr_model.fit(X_train_norm, y_train)

# Modelle test verisi üzerinde tahmin yapma
y_pred = best_svr_model.predict(X_test_norm)
```

[8]:
```
# Gerçek ve tahmin edilen değerleri içeren DataFrame oluşturma
comparison_df = pd.DataFrame({
    'Actual Weight': y_test,
    'Predicted Weight': y_pred
})

# Karşılaştırma tablosunu görselleştirme
display(comparison_df.head(10).style.set_precision(2).set_table_styles(
    [{'selector': 'th', 'props': [('font-size', '12pt'), ('text-align', 'center')]}]
))

# İstenen indeksleri seçme
index = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

# Seçilen indekslerin bağımsız değişken değerlerini alma
X_selected_1 = X.loc[index]
X_selected_1_norm = scaler.transform(X_selected_1)
y_pred_selected_1 = best_svr_model.predict(X_selected_1_norm)

comparison_table = X_selected_1.copy()
comparison_table['Ölçüm Değerleri'] = y.loc[index]
comparison_table['Tahmin Değerleri'] = y_pred_selected_1

# Sonuçları gösterme
display(comparison_table.style.set_precision(2))
```

•••

[9]:
```
import matplotlib.pyplot as plt

# Scatter plot oluşturma
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'k--')
plt.xlabel('Gerçek Değerler')
plt.ylabel('Tahmin Değerleri')
plt.title('SVR Gerçek Değerler vs Tahminler')
plt.grid(True)
plt.show()
```

•••

[10]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

#hiper parametre ayarlanmış sonuçlar
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Ortalama Mutlak Hata (MAE): {mae:.5f}")
print(f"Ortalama Kare Hata (MSE): {mse:.5f}")
print(f"Kök Ortalama Kare Hata (RMSE): {rmse:.5f}")
print(f"R-kare: {r2:.5f}")
```

```
Ortalama Mutlak Hata (MAE): 0.81139
Ortalama Kare Hata (MSE): 1.18903
Kök Ortalama Kare Hata (RMSE): 1.09043
R-kare: 0.97626
```

[11]:
```
from sklearn.inspection import permutation_importance
```

[12]:
```
# Compute permutation feature importance
perm_importance = permutation_importance(best_svr_model, X_test_norm, y_test, n_repeats=30, random_state=42)

# Create a DataFrame to display feature importances
feature_importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': perm_importance.importances_mean
}).sort_values(by='Importance', ascending=False)

# Visualize feature importances
plt.figure(figsize=(5, 4))
plt.barh(feature_importance_df['Feature'], feature_importance_df['Importance'])
plt.xlabel('Önem Düzeyi')
plt.ylabel('Bağımsız Değişken')
plt.title('Bağımsız Değişken Önem Düzeyi (SVR)')
plt.show()
```

•••

Ek 5. XGBoost Algoritma Kodları

[1]:
```
#Kütüphane içe aktarma
import numpy as np
import pandas as pd

from warnings import filterwarnings
filterwarnings('ignore')
```

[2]:
```
#veriyi içe aktarma
ad = pd.read_excel("tez_kuzu_verisi_kopyası.xlsx")
#veri kopyası oluşturma
df = ad.copy()
#veride ilk 5 gösterme
df.head()
```

•••

[3]:
```
# x(bağımsız değişken) ve y ( bağımlı değişken)
X = df.drop("agırlık", axis = 1 )
y = df["agırlık"]
```

[4]:
```
#makine  öğrenmesi test verisi ayırma
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split ( X, y , test_size = 0.25, random_state= 42)
```

[5]:
```
#xgboost kütüphane import
from xgboost import XGBRegressor
#model kurma
xgb_model = XGBRegressor().fit(X_train, y_train)
```

[6]:
```
#Model tahmin
y_pred = xgb_model.predict(X_test)
```

[7]:
```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBRegressor

# XGBoost modeli için hiperparametre grid'i
xgb_grid = {
     'colsample_bytree': [0.4, 0.5,0.6,0.9,1],
     'n_estimators':[800,900,1000,2000,3000],
     'max_depth': [2,3,4,5,6,7],
     'learning_rate': [0.1, 0.01, 0.02,0.05]
}

# GridSearchCV objesini oluşturma
xgb = XGBRegressor()

xgb_cv = GridSearchCV(xgb,
                      param_grid = xgb_grid,
                      cv = 5,
                      n_jobs = -1,
                      verbose = 2)


# Modeli eğitim verisi üzerinde fit etme
xgb_cv.fit(X_train, y_train)
```

•••

[8]:
```
xgb_cv.best_params_
```

•••

[9]:
```
y_pred = xgb_cv.predict(X_test)
```

[11]:
```python
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

#hiper parametre ayarlanmış sonuçlar
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Ortalama Mutlak Hata (MAE): {mae:.5f}")
print(f"Ortalama Kare Hata (MSE): {mse:.5f}")
print(f"Kök Ortalama Kare Hata (RMSE): {rmse:.5f}")
print(f"R-kare: {r2:.5f}")
```

•••

[12]:
```python
# İstenen indeksleri seçme
index = [0,1,2,3,4,5,6,7,8,9,10,11,12]

# Seçilen indekslerin bağımsız değişken değerlerini alma
X_selected_1 = X.loc[index]
y_pred_selected_1 = xgb_cv.predict(X_selected_1)
comparison_table = X_selected_1.copy()
comparison_table['Ölçüm Değerleri'] = y.loc[index]
comparison_table['Tahmin Değerleri'] = y_pred_selected_1
display(comparison_table.style.set_precision(2))
```

•••

[14]:
```python
import matplotlib.pyplot as plt

#hiperparametleri ayarlanmış gerçek tahmini değer grafik
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel('Gerçek Değerler')
plt.ylabel('Tahminler')
plt.title('Gerçek Değerler vs Tahminler')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--')  # Mükemmel tahmin hattı
plt.show()
```

•••

[15]:
```python
# Özellik önem düzeylerini alma
xgb_importance = xgb_model.feature_importances_

# Görselleştirme
xgb_importance_df = pd.DataFrame({
    'Feature': X_train.columns,
    'Importance': xgb_importance
}).sort_values(by='Importance', ascending=False)

plt.figure(figsize=(5, 4))
plt.barh(xgb_importance_df['Feature'], xgb_importance_df['Importance'])
plt.xlabel('Önem Düzeyi')
plt.ylabel('Bağımlı Değişken')
plt.title('Bağımlı Değişken Önem Düzeyi (XGBoost) ')
plt.show()
```

•••