4.Hafta

KARAR AĞAÇLARI VE ANSAMBL YÖNTEMLER

1) Decision Tree (Karar Ağacı) Classifier & Regressor

Karar Ağaçları, hem sınıflandırma hem de regresyon problemleri için kullanılabilen, akış şemasına benzeyen denetimli öğrenme algoritmalarıdır. Veriyi, sorular sorarak (özellik değerlerine göre) dallara ayırarak ve her dalda belirli bir karara (sınıf etiketi veya sayısal değer) ulaşarak öğrenirler.

- **Yapısı:**
  - **Kök Düğüm (Root Node):** Tüm veri setini temsil eden en üstteki düğümdür.
  - **Karar Düğümü (Decision Node):** Bir özelliğin değeri hakkında bir test yapılan düğümdür. ("Yaş > 30?").
  - **Yaprak Düğümü (Leaf Node):** Kararın verildiği veya sonucun tahmin edildiği son düğümdür. Daha fazla bölünme olmaz.
  - **Dallar (Branches):** Karar düğümlerinden çıkan ve testin sonucunu temsil eden yollardır.
- **Nasıl Çalışır?**
    1. Algoritma, veri setini en iyi bölen (yani hedef değişkeni en saf gruplara ayıran) özelliği ve eşik değerini bulur.
    2. Veri, bu özelliğe göre iki veya daha fazla alt gruba ayrılır.
    3. Bu işlem, her alt grup (düğüm) homojen hale gelinceye veya durdurma kriterleri (maksimum derinlik, minimum örnek sayısı vb.) karşılanana kadar özyinelemeli olarak devam eder.
  - **Sınıflandırma (Classifier):** Yaprak düğümlerdeki çoğunluk sınıf etiketi tahmin edilir. (Örn: "Müşteri 'evet' mi der 'hayır' mı?")
  - **Regresyon (Regressor):** Yaprak düğümlerdeki veri noktalarının ortalama değeri tahmin edilir. (Örn: "Bir evin fiyatı ne olur?")
- **Bölme Kriterleri (Sınıflandırma için):**
  - **Gini Saflığı (Gini Impurity):** Rastgele seçilen bir öğenin yanlış sınıflandırılma olasılığını ölçer. Amacı Gini'yi minimize etmektir.
  - **Enformasyon Kazancı (Information Gain) / Entropi:** Bir bölmenin veri setindeki belirsizliği (entropiyi) ne kadar azalttığını ölçer. Amacı bilgi kazancını maksimize etmektir.
- **Avantajları:**
  - Anlaşılması ve yorumlanması kolaydır (tıpkı bir akış şeması gibi).
  - Veri ön işlemeye (normalizasyon/ölçeklendirme) daha az ihtiyaç duyar.
  - Hem sayısal hem de kategorik verilerle çalışabilir.
- **Dezavantajları:**
  - **Aşırı Uyum (Overfitting)** eğilimi gösterirler (aşağıda detaylı açıklanacak).
  - Küçük veri değişikliklerine karşı hassas olabilirler (model yapısı değişebilir).
  - Optimum bir karar ağacı bulmak NP-hard bir problemdir; pratik algoritmalar genellikle açgözlü (greedy) yaklaşım kullanır.

2) Overfitting (Aşırı Uyum) Nedir? Nasıl Önlenir?

Aşırı Uyum (Overfitting), bir makine öğrenmesi modelinin **eğitim verisine çok iyi uyum sağladığı, ancak yeni ve daha önce görmediği (test) verilere genelleme yapmada başarısız olduğu** durumdur. Model, eğitim verisindeki gürültüyü ve rastlantısal desenleri bile ezberler, bu da test performansının düşmesine neden olur. Karar ağaçları, doğaları gereği aşırı uyuma çok yatkındır.

Nasıl Anlaşılır?

- Eğitim hatası (train error) çok düşükken, test hatası (test error) yüksekse.
- Model çok karmaşık (çok sayıda düğüm/derinlik) ve veri setindeki gürültüyü bile modellemişse.

Nasıl Önlenir?

1. **Daha Fazla Veri Toplama:** Daha fazla ve daha çeşitli eğitim verisi, modelin genelleme yeteneğini artırır.
2. **Özellik Seçimi/Mühendisliği:** Model için en alakalı ve önemli özellikleri seçmek veya yeni, daha açıklayıcı özellikler oluşturmak, gürültüyü azaltır.
3. **Model Karmaşıklığını Azaltma (Regularizasyon/Budama):**
    - **Ön-budama (Pre-pruning):** Ağacın büyümesini belirli bir noktada durdurmak için kriterler belirlemek (örneğin, maksimum derinlik, bir düğümdeki minimum örnek sayısı, minimum bilgi kazancı).
    - **Son-budama (Post-pruning):** Tam büyümüş bir ağacı oluşturduktan sonra, genelleme performansını artırmak için dalları budamak.
4. **Çapraz Doğrulama (Cross-Validation):** Modeli değerlendirmek için tek bir eğitim/test bölme yerine veriyi birden çok kez bölerek modelin performansını daha sağlam bir şekilde ölçmek. Bu, model seçimi ve hiperparametre ayarı için kritiktir.
5. **Ansambl Yöntemleri (Ensemble Methods):** Birden fazla temel modelin (örn. karar ağaçlarının) birleştirilerek daha güçlü ve genellenebilir bir model oluşturulması. Bu, aşırı uyumu önlemenin en etkili yollarından biridir.

3) Random Forest (Rastgele Orman) Yapısı

Random Forest, birden fazla karar ağacının bir araya gelerek oluşturduğu bir **ansambl (topluluk) öğrenme** algoritmasıdır. Özellikle sınıflandırma ve regresyon problemlerinde yüksek performans gösterir ve aşırı uyumu azaltmada çok etkilidir.

- **Temel Fikir:** "Birçok kötü sınıflandırıcının birleşimi, tek bir iyi sınıflandırıcıdan daha iyi olabilir." mantığına dayanır.
- **Nasıl Çalışır? (Bagging prensibi üzerine kurulu):**
    1. **Bootstrapping (Yeniden Örnekleme):** Orijinal eğitim veri setinden, yerine koymalı (yani aynı örnek birden fazla kez seçilebilir) olarak **birden çok (N adet) alt örneklem (subset)** oluşturulur. Her alt örneklem, orijinal veri setinin yaklaşık %63'ünü içerir (geri kalan %37'ye **"Out-of-Bag"** örnekler denir ve iç doğrulama için kullanılabilir).
    2. **Özellik Rastgeleliği:** Her bir alt örneklem üzerinde bir karar ağacı eğitilirken, düğümleri bölmek için **tüm özelliklerin rastgele bir alt kümesi** değerlendirilir. Bu, ağaçlar arasında çeşitlilik yaratır ve ağaçların aşırı korelasyonlu olmasını engeller.

3. **Ağaç Eğitimi:** Her bir alt örneklem ve rastgele özellik alt kümesi kullanılarak bağımsız bir karar ağacı eğitilir. Bu ağaçlar genellikle tam büyümeye bırakılır (budama yapılmaz), çünkü rastgelelik zaten aşırı uyumu azaltmaya yardımcı olur.
4. **Tahmin:**
    - **Sınıflandırma için:** Tüm ağaçlar tahmin yapar ve **çoğunluk oyu** (mode) ile son sınıflandırma belirlenir.
    - **Regresyon için:** Tüm ağaçların tahminlerinin **ortalaması** alınarak nihai tahmin belirlenir.

- **Avantajları:**
    - **Yüksek Doğruluk:** Genellikle yüksek doğruluk sağlar.
    - **Aşırı Uyuma Karşı Dirençli:** Çok sayıda ağacın ve rastgeleliğin kullanılması sayesinde aşırı uyumu büyük ölçüde azaltır.
    - **Kararlı:** Küçük veri değişikliklerine karşı daha az hassastır.
    - **Özellik Önemini (Feature Importance) Hesaplar:** Hangi özelliklerin model için daha önemli olduğunu belirleyebilir.
- **Dezavantajları:**
    - Karar ağacına göre daha az yorumlanabilirdir ("kara kutu" modeli gibi).
    - Daha fazla hesaplama gücü gerektirir (çok sayıda ağaç eğitilir).

4) Bagging vs Boosting

Ansambl yöntemleri, birden fazla öğrenme algoritmasının (temel öğreniciler) birleştirilerek daha iyi bir genel performans elde edilmesi ilkesine dayanır. İki ana yaklaşım vardır:

1. Bagging (Bootstrap Aggregating)

- **Mantık:** Temel öğrenicileri (genellikle yüksek varyanslı ama düşük biaslı modeller, yani Karar Ağaçları) **paralel olarak** eğitir ve her birinin tahminini birleştirir (sınıflandırma için çoğunluk oyu, regresyon için ortalama).
- **Nasıl Çalışır?**
    - Orijinal veri setinden **rastgele alt örnekler (bootstrap örnekleri)** oluşturulur.
    - Her bir alt örnek üzerinde **bağımsız bir temel öğrenici** eğitilir.
    - Eğitimden sonra, tüm temel öğrenicilerin tahminleri birleştirilerek nihai tahmin elde edilir.
- **Amacı:** Modelin **varyansını (variance)** azaltmaktır. Rastgele Orman, bagging'in en popüler örneğidir.

2. Boosting

- **Mantık:** Temel öğrenicileri (genellikle zayıf öğreniciler, yani düşük varyanslı ama yüksek biaslı modeller) **sıralı (ardışık)** bir şekilde eğitir. Her yeni temel öğrenici, önceki öğrenicinin yaptığı hatalara odaklanarak eğitilir.
- **Nasıl Çalışır?**
    - İlk temel öğrenici tüm veri seti üzerinde eğitilir.
    - Daha sonraki her öğrenici, önceki öğrenicinin **yanlış sınıflandırdığı veya kötü tahmin ettiği örneklere daha fazla ağırlık vererek** eğitilir.
    - Bu süreç, belirli bir sayıda temel öğreniciye ulaşılana veya performans artışı durana kadar devam eder.
    - Nihai tahmin, tüm temel öğrenicilerin **ağırlıklı toplamı** (weighted sum) ile elde edilir.
- **Amacı:** Modelin **önyargısını (bias)** ve bir miktar varyansını azaltmaktır**. Adaboost** ve **Gradient Boosting**, boosting'in popüler örnekleridir.

5) Adaboost (Adaptive Boosting) Mantığı

AdaBoost (Adaptive Boosting), boosting ailesinin ilk ve en bilinen algoritmalarından biridir. Genellikle zayıf öğrenicileri (örneğin, sadece tek bir bölme yapan "karar sapı" - decision stump) kullanarak güçlü bir sınıflandırıcı oluşturur.

- **Nasıl Çalışır?**
    1. Tüm eğitim örneklerine başlangıçta eşit ağırlıklar atanır.
    2. Her bir iterasyonda:
        - Bir **zayıf öğrenici** (örneğin, bir karar sapı) veri seti üzerinde eğitilir.
        - Bu öğrenici tarafından **yanlış sınıflandırılan örneklerin ağırlıkları artırılır**, doğru sınıflandırılanların ağırlıkları azaltılır. Bu sayede, sonraki zayıf öğreniciler yanlış yapılan örneklere daha fazla odaklanır.
        - O anki zayıf öğreniciye, doğruluk performansına göre bir **ağırlık** atanır (daha iyi performans gösterenlere daha yüksek ağırlık).
    3. Bu adımlar, belirlenen sayıda iterasyon veya hata oranı eşiği karşılanana kadar tekrarlanır.
    4. Nihai sınıflandırma, tüm zayıf öğrenicilerin **ağırlıklı çoğunluk oyu** ile yapılır. Daha doğru tahmin yapan öğrenicilerin oyları daha fazla sayılır.
- **Avantajları:** Basit ve etkili, aşırı uyuma karşı nispeten dayanıklı.
- **Dezavantajları:** Gürültülü verilere ve aykırı değerlere karşı hassas olabilir, temel öğrenicilerin kalitesine bağlıdır.

6) Gradient Boosting

Gradient Boosting, Adaboost'tan daha genel ve güçlü bir boosting algoritmasıdır. Önceki öğrenicilerin hatalarına odaklanmak yerine, **kalan hataları (residuals) veya gradyanları** öğrenmeye çalışır.

- **Mantık:** Model, gerçek değerler ile mevcut modelin tahminleri arasındaki farkı (hata) minimize etmeye çalışır. Bunu yaparken, her yeni temel öğrenici (genellikle küçük karar ağaçları), önceki ağaçların **hatalarının gradyanını (yani hatanın yönünü ve büyüklüğünü)** öğrenerek bu hataları düzeltmeye odaklanır.
- **Nasıl Çalışır?**
    1. İlk temel öğrenici (genellikle bir ortalama veya basit bir ağaç), hedef değişkeni tahmin etmek için eğitilir.
    2. Her iterasyonda:
        - Mevcut modelin **tahmin hataları (residual'lar)** hesaplanır (Gerçek Değer - Tahmin Edilen Değer).
        - Yeni bir temel öğrenici, bu **residual'ları tahmin etmek** için eğitilir. Yani, model artık doğrudan hedefi değil, hedefin hata payını öğrenmeye çalışır.
        - Bu yeni öğrenicinin tahmini, mevcut modele eklenir (genellikle küçük bir "öğrenme oranı" ile çarparak, bu öğrenme oranı aşırı uyumu kontrol etmeye yardımcı olur).
    3. Bu süreç, belirlenen sayıda ağaca ulaşılana veya performans artışı durana kadar tekrarlanır.
    4. Nihai tahmin, tüm temel öğrenicilerin (artık tahmin edilen residual'lar) toplamıdır.
- **Avantajları:**
    - Çok yüksek doğruluk ve performans sunar, genellikle Kaggle yarışmalarında en üst sıralarda yer alır.
    - Hem sınıflandırma hem de regresyon için kullanılabilir.
    - Özellik önemini hesaplayabilir.
- **Dezavantajları:**
    - Eğitim süresi uzun olabilir ve hesaplama açısından yoğundur.
    - Hiperparametre ayarı karmaşık olabilir (çok sayıda ayarlanacak parametre vardır).
    - Ardışık yapısı nedeniyle paralelleştirilmesi zordur.
    - Aşırı uyuma Random Forest'tan daha yatkın olabilir (iyi hiperparametre ayarı gereklidir).

Gradient Boosting'in Gelişmiş Versiyonları:

- **XGBoost:** Hızlı ve performanslı, düzenlileştirme teknikleri içerir.
- **LightGBM:** Daha hızlı eğitim, daha az bellek kullanımı.
- **CatBoost:** Kategorik özelliklerle daha iyi başa çıkar.