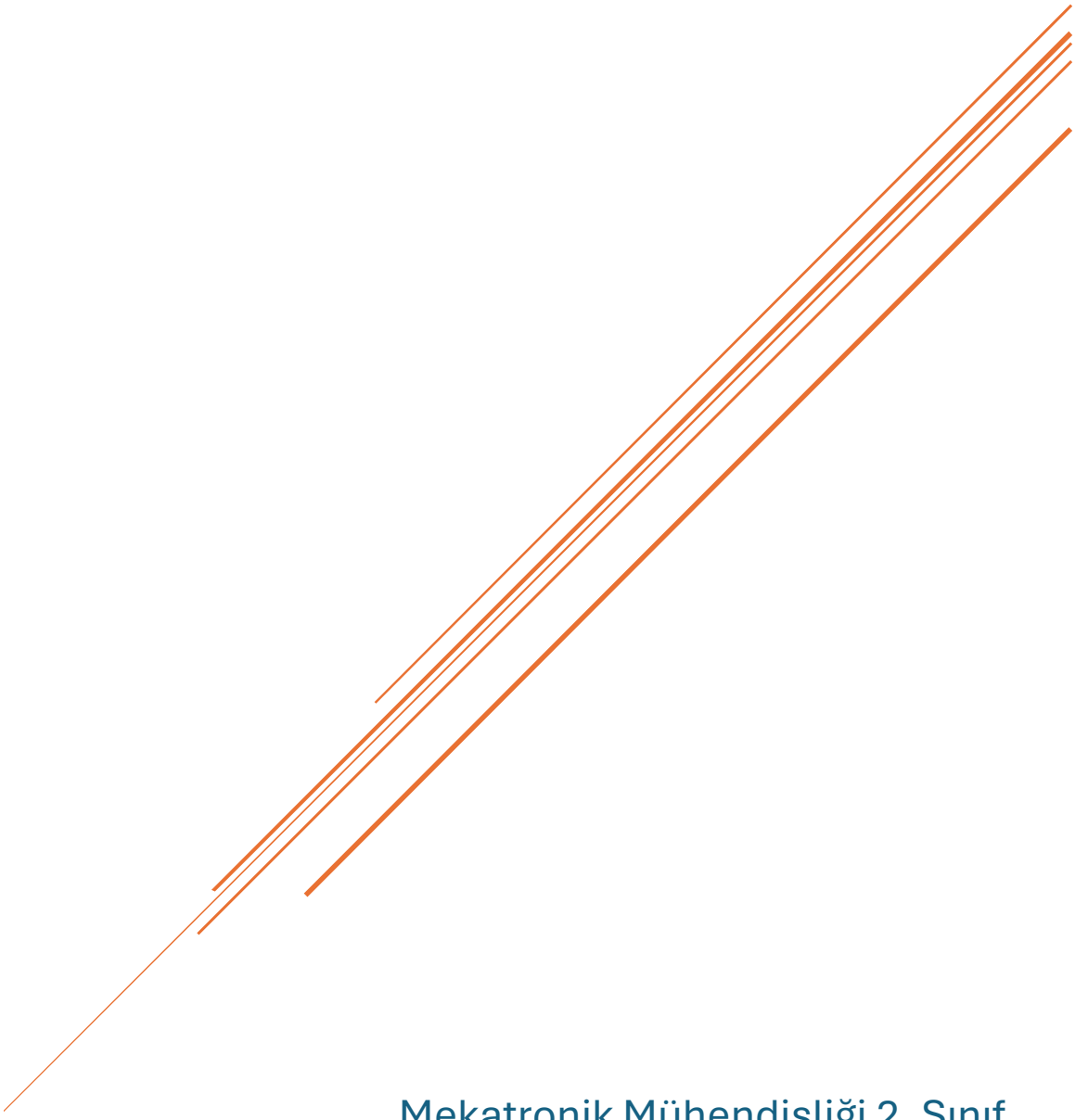
AIRLAB TASK

Eylül Beril Yurttaş



Mekatronik Mühendisliği 2. Sınıf

Contents

1. Temel Veri ve Makine Öğrenmesi Kavramları

1.1. Veri Analizi ve Hazırlığı

1.1.1. Veri analitiği nedir?

Veri analitiği, veri kümelerinden anlamlı bilgiler elde etmek için matematik, istatistik ve bilgisayar bilimi de dahil olmak üzere çok çeşitli analiz tekniklerini kullanan çok disiplinli bir alandır.
Veri analitiği, ham verileri eyleme dönüştürülebilir anlamlı bilgiye haline getirir. Veri analitiğinde amaç verilerden örüntüler, ilişkiler ve öngörüler çıkararak karar alma süreçlerini iyileştirmektir.

1.1.2. Bir makine öğrenmesi projesinde aykırı veriler (outliers) ile karşılaşıldığı zaman ne yapılabilir?

Bir aykırı değer, "oluştuğu örneklemdeki diğer üyelerden belirgin biçimde sapma gösteren" bir gözlemdir. [6] Aykırı değerler, veri kümelerinde genellikle önemli bilgiler barındırırlar. Aykırı verilerin öncelikle tespit edilmesi daha sonrasında ise bu verilerin yapısına göre verilere ne yapılacağının tespit edilmesi gerekir.

1.1.2.1. Aykırı Veri Tespiti (Outliers Detection)

Bu bölüm 3 aşamada incelenebilir:

İstatistiksel Yöntemler	• Z-Score • IQR (Interquartile Range)
Görsel Yöntemler	• Boxplot • Scatter Plot • Histagram
Model Tabanlı Yöntemler	• Isolation Forest • Local Outlier Factor (LOF)

Tablo 1- Aykırı Veri Tespit yöntemleri

1.1.2.2. Aykırı Veriyle Baş Etme (Handling Outliers)

Aykırı verilerle tespit edildikten sonra aykırı verinin tipine göre (univariate, multivariate, point, contextual, collective vb.) aykırı verilere ne yapılacağı belirlenir.

→ **Kaldırmak (Remove)**
Aykırılar hatalı veya anlamsız ise silinebilir. Ama çok fazla aykırı varsa veri kaybına yol açabilir.

→ **Dönüştürmek (Transform)**
Log, karekök ($\sqrt{}$), Box-Cox gibi dönüşümler aşırı uç değerlerin etkisini azaltır.

→ **Winsorizing (Sınırlandırma)**
Aykırı değerleri belli bir yüzdeye kadar sınırlandırılır. En yüksek %1'lik dilimi %99'uncu değere eşitlemek gibi.

→ **İyileştirmek (Impute)**
Hatalı ölçümse, ortalama veya medyan değerle değiştirilebilir.

→ **Model dayanıklılığını artırmak**
Bazı modeller aykırılara karşı daha dayanıklıdır. Örneğin, karar ağaçları (Decision Tree, Random Forest), Robust Regression, RANSAC gibi algoritmalar outlier'ların etkisini azaltır.

1.1.3. Data Leakage nedir ve makine öğrenmesi projelerinde neden tehlikelidir?

Makine öğreniminde veri sızıntısı, bir modelin eğitim sırasında yani tahmin anında mevcut olmayan bilgileri kullanması durumunda ortaya çıkar. Sızıntı, tahmin modelinin kullanım senaryosuna uygulanana kadar doğru görünmesine neden olur ancak daha sonra hatalı sonuçlar verir ve bu da zayıf karar alma süreçlerine ve yanlış içgörülere yol açar.

1.1.4. Eksik verilerle (Missing Data) başa çıkma yöntemlerinden 3 tanesini kısaca açıklayınız.

- **Silme Yöntemleri (Deletion)**

→ **Listwise Deletion (Complete Case Analysis)**
Herhangi bir sütunda eksik değer bulunan tüm satırlar analizden kaldırılır. Basit olmasına rağmen yalnızca veri kümesinin büyük, eksik veri oranının nispeten düşük olduğu ve Tamamen Rastgele Eksik (MCAR) durumunda kullanılması uygundur.

→ **Pairwise Deletion**
Eksik olmayan değerlere sahip değişken çiftleri kullanılarak ve istatistikler hesaplanarak mevcut tüm verilerin analiz için kullanılmasını içerir. Liste bazında silmenin aksine bu yaklaşım, her bir korelasyonu veya istatistiği ilgili değişken çifti için mevcut verilerle hesaplayarak mümkün olduğunca fazla veriyi korur.

- **Basit Atama Yöntemleri (Simple Imputation)**

Eksik değerler, aynı sütundaki diğer değerlerin ortalaması, medyanı veya modu ile doldurulur. Veriyi korur ve uygulanması kolaydır. Fakat özellikle de çok sayıda eksik veri varsa verilerin doğal dağılımını bozabilir.

Örneğin: Yaş sütunundaki eksik değerler ortalama yaş ile doldurulabilir

- **Gelişmiş Atama Yöntemleri (Advanced Imputation)**

→ **K-Nearest Neighbors (KNN) Imputation**
En yakın K satırı bulur ve eksik değerleri doldurmak için değerlerinin ortalamasını alır. Bu yöntem, eksik değerlerin benzer özelliklere sahip yakın örneklerden çıkarılabildiği durumlarda kullanışlıdır.

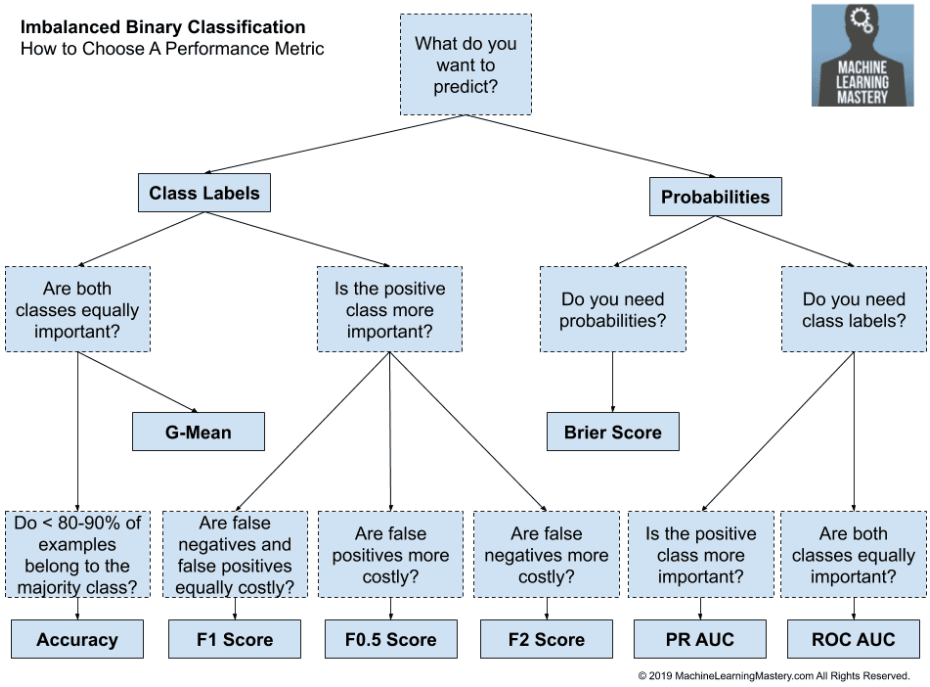
→ **Multiple Imputation by Chained Equations (MICE)**
Birden fazla veri kümesi oluşturup sonuçları birleştiren daha gelişmiş bir yaklaşımdır. MICE, birden fazla veri kümesi oluşturur her biri üzerinde analiz gerçekleştirir ve ardından daha sağlam bir veri yüklemesi için sonuçları birleştirir.

1.1.5. "Özellik Mühendisliği (Feature Engineering)" nedir ve bir modelin performansına nasıl etki eder?

Özellik mühendisliği, makine öğrenmesi sürecinde ham verilerin modelin öğrenme kapasitesini artıracak biçimde dönüştürülmesi, seçilmesi veya yeni özelliklerin türetilmesi işlemidir.
Doğru şekilde seçilmiş ve dönüştürülmüş özellikler, modelin veri içindeki örüntüleri (patterns) daha net biçimde algılamasını sağlayarak modelin doğruluğunu (accuracy) artırırken genelleme kabiliyetini (generalization ability) de güçlendirir yani model yalnızca eğitim verisinde değil daha önce karşılaşmadığı yeni veriler üzerinde de başarılı tahminlerde bulunabilir. Ayrıca, anlamlı ve sadeleştirilmiş özelliklerin kullanılması modelin öğrenme süresini kısaltır ve hesaplama maliyetini düşürür çünkü model gereksiz veya yinelenen değişkenlerle vakit kaybetmez.
Verideki gürültüyü (noise) ve karmaşıklığı (complexity) azaltarak modelin istikrarlı sonuçlar üretmesini sağlar. Fakat hatalı veya yetersiz bir özellik mühendisliği uygulandığında model, verideki önemli ilişkileri ve bağıntıları yakalayamaz. Bu da modelin yanlış genellemeler yapmasına, düşük performans göstermesine ve tahmin hatalarının artmasına neden olur.

1.1.6. Veri kümesi dengesizliği (Imbalanced Dataset) durumunda kullanılabilecek 2 farklı değerlendirme metriği (evaluation metric) nedir ve neden sadece doğruluk (accuracy) yeterli değildir?

Veri kümesi dengesizliği (Imbalanced Dataset), bir sınıfın diğerine göre çok daha fazla örneğe sahip olduğu durumlarda ortaya çıkar. Bu tür veri setlerinde, model çoğunluk sınıfını tahmin ederek yüksek doğruluk (accuracy) elde edebilir ancak bu sonuç yanıltıcıdır çünkü model azınlık sınıfını (genellikle asıl ilgilenilen sınıfı) doğru şekilde öğrenmemiş olabilir.
Bu nedenle yalnızca doğruluk metriği, dengesiz veri setlerinde modelin gerçek performansını yansıtmakta yetersizdir.



Resim 1- Dengesiz İkili Sınıflandırma

F-Beta Skoru

F-beta puanı, hem dengeli hem de dengesiz veri kullanım durumlarını puanlamak için oldukça güçlü bir puanlama mekanizmasıdır. Yaygın olarak kullanılan F1 puanının, β değeri 1 olarak ayarlanmış genelleştirilmiş halidir. F-Beta, hem Precision hem de Recall'u hesaba katar ve ikisi arasında ağırlıklı bir Harmonik ortalama alır.

Precision (Kesinlik): Modelin "pozitif" olarak tahmin ettiği örneklerin ne kadarının gerçekten pozitif olduğunu gösterir.
Recall (Duyarlılık): Gerçek pozitif örneklerin ne kadarının model tarafından doğru tahmin edildiğini gösterir.

F-Beta skoru, bu iki metriği dengeleyerek modelin azınlık sınıfını ne kadar iyi tespit ettiğini ölçer. Bu nedenle dengesiz veri setlerinde, özellikle yanlış negatiflerin (örneğin hastalığı olup da "sağlıklı" olarak sınıflandırılan bireylerin) maliyeti yüksek olduğunda F-Beta skoru tercih edilir.

ROC-AUC (Receiver Operating Characteristic – Area Under Curve)

ROC-AUC, modelin sınıflar arasında ayrım yapma kabiliyetini değerlendirir. ROC eğrisi, True Positive Rate (TPR) ile False Positive Rate (FPR) arasındaki ilişkiyi gösterir; AUC (eğri altındaki alan) değeri ne kadar yüksekse, model o kadar iyi ayırma yapıyor demektir.
AUC değeri 1'e yakınsa model iki sınıfı çok iyi ayırıyor, 0.5 civarındaysa model rastgele tahmin yapıyor anlamına gelir. Bu metrik, sınıf dengesizliği durumlarında modelin azınlık sınıfını tanıma gücünü doğruluktan çok daha sağlıklı biçimde ölçer.

1.2. Makine Öğrenmesi Temelleri

1.2.1. Makine öğrenmesinin alt dalları nelerdir? Birkaç cümle ile açıklar mısınız?

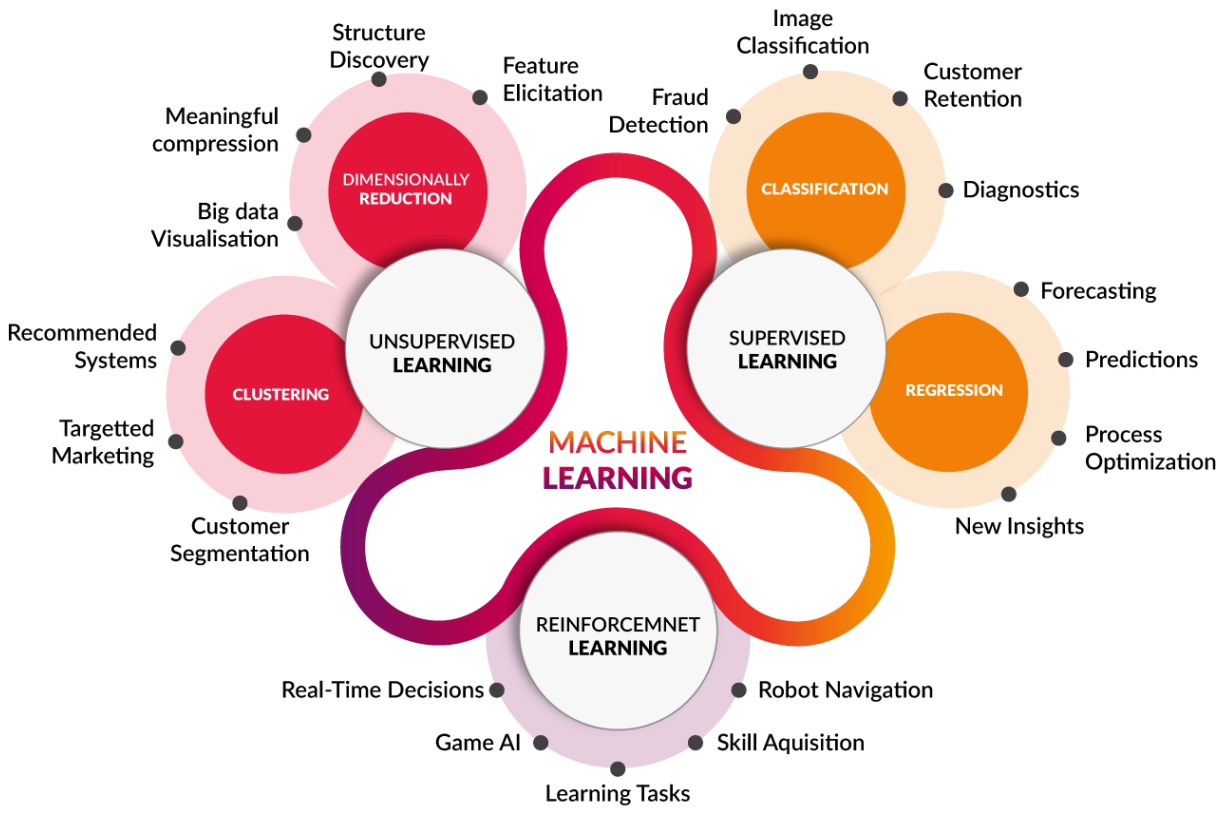
Makine öğrenmesi dört alt başlıkta incelenebilir.

→ **Unsupervised (Denetimsiz)**
Etiketlenmemiş veriler üzerinde gizli yapıları veya örüntüleri bulmayı amaçlar.

→ **Semi-Supervised (Yarı denetimli)**
Az miktarda etiketli ve çok miktarda etiketsiz veri birlikte kullanılarak modelin eğitildiği yöntemdir.

→ **Supervised (Denetimli)**
Girdi ve buna karşılık gelen doğru çıktılar (etiketler) kullanılarak modelin eğitildiği yöntemdir.

→ **Reinforcement (Pekiştirilmiş)**
Denetlenmeyen makine öğrenimi gibi etiketlenmemiş veri kümeleri kullanır ve algoritmaların verileri değerlendirmesine olanak tanır.



Resim 2- Makine Öğrenmesi Alanları

1.2.2. Cost function (Maliyet Fonksiyonu) nedir?

Makine öğrenmesinde bir modelin ne kadar hata yaptığını ölçen matematiksel bir ifadedir. Maliyet fonksiyonunun türleri vardır.

Mean Squared Error (MSE)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i{}^2)$$

Binary Cross-Entropy/Logloss

$$Log\ Loss = \frac{1}{N}\sum_{i=1}^{N} -(y_i * log(p_i) + (1 - y_i) * log\ (1 - p_i)$$

Categorical Cross-Entropy

$$CCE = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij}^{(actual)} * log\ (y_{ij}^{(predict)})$$

Hinge Loss

$$HL = max(0, 1 - y_{actual} * y_{predict})$$

Kullback-Leibler Divergence

$$KL = \sum_{i=1}^{k} y_i^{(actual)} * log\frac{y_i^{(actual)}}{y_i^{(predict)}}$$

Root Mean Squared Error (RMSE)

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

Mean Absolute Error (MAE)

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

1.2.3. Gradient descent (Gradyan İnişi) nedir?

Makine öğrenmesinde bir optimizasyon algoritmasıdır ve modelin cost function (maliyet fonksiyonu) değerini en küçük (minimum) hale getirmek için kullanılır.

$$\theta = \theta - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta}$$

$\boldsymbol{\theta}$: Öğrenilen parametre (örneğin model katsayısı veya ağırlık)
$\boldsymbol{J(\theta)}$: Cost function
$\frac{\partial J(\theta)}{\partial \theta}$: Cost fonksiyonunun gradyanı (eğimi)
$\boldsymbol{\alpha}$: Öğrenme oranı (learning rate) → adımın büyüklüğünü belirler

1.2.4. Learning rate (Öğrenme Oranı) fazla küçük ve fazla büyük seçilirse ne olur?

Düşük öğrenme oranı, modelin yavaş yakınsamasına neden olur ve daha fazla eğitim döngüsü (epoch) gerektirir. Bu durum, doğruluğu artırabilir ancak aynı zamanda hesaplama süresini de uzatır. Yüksek öğrenme oranı ise eğitimi hızlandırır fakat optimal ağırlıkların aşılmasına, dolayısıyla modelin kararsız hale gelmesine veya kayıp fonksiyonunun sapmasına neden olabilir. Bu nedenle, optimal öğrenme oranı eğitim hızını ve model doğruluğunu dengeleyen, aşırı eğitim süresine gerek kalmadan kararlı bir yakınsama sağlayan değerdir.

1.2.5. Over-fit (Aşırı Uyum) ve under-fit (Eksik Uyum) nedir, hangi sebeplerden meydana gelir?

Aşırı uyum, bir modelin eğitim verilerinden çok fazla şey öğrenmesi durumunda ortaya çıkar. Gürültü veya aykırı değerler gibi önemli olmayan ayrıntılar da içerir. Over-fit durumunun sebepleri vardır:

→ Yüksek varyans ve düşük bias.
→ Modelin kompleks olması
→ Eğitim verilerinin boyutunun büyüklüğü

Yetersiz uyum, aşırı uyumun tam tersidir. Bir modelin verilerde olup biteni yakalamak için çok basit olması durumunda ortaya çıkar. Under-fit durumunun sebepleri vardır:

→ Model çok basittir, bu yüzden verideki karmaşıklıkları temsil edemeyebilir.
→ Modeli eğitmek için kullanılan giriş özellikleri (input features), hedef değişkeni etkileyen temel faktörleri yeterince temsil etmemektedir.
→ Kullanılan eğitim veri setinin boyutu yeterli değildir.
→ Aşırı düzenlileştirme (regularization) uygulanmıştır; bu da modelin veriyi yeterince iyi öğrenmesini kısıtlar.
→ Özellikler (features) uygun şekilde ölçeklendirilmemiştir.

1.2.6. Aktivasyon fonksiyonu nedir ve ne gibi fonksiyonlar kullanılır?

Aktivasyon fonksiyonu (Activation Function), yapay sinir ağlarında bir nöronun çıktısını belirleyen matematiksel bir fonksiyondur. Temel görevi, ağın karmaşık ve doğrusal olmayan ilişkileri öğrenmesini sağlamaktır.

Eğer aktivasyon fonksiyonu kullanılmazsa sinir ağı yalnızca doğrusal bir model gibi davranır ve verideki karmaşık desenleri (örneğin görüntü, ses, metin verileri gibi) öğrenemez.

Sigmoid Fonksiyonu

$$f(x) = \frac{1}{1 + e^{-x}}$$

Çıkış aralığı: (0, 1)
Özellik: Küçük değerleri 0'a, büyük değerleri 1'e yaklaştırır.
Dezavantaj: Büyük pozitif veya negatif değerlerde gradyan sıfıra yaklaşır → vanishing gradient problemi.

Tanh (Hiperbolik Tanjant) Fonksiyonu

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Çıkış aralığı: (-1, 1)
Sigmoid'den daha iyidir, çünkü sıfır merkezlidir.
Ancak yine vanishing gradient problemi yaşayabilir.

ReLU (Rectified Linear Unit) Fonksiyonu

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

Negatif değerleri 0 yapar, pozitifleri aynen geçirir.
En çok kullanılan aktivasyon fonksiyonudur.
Avantaj: Hesaplama olarak çok hızlıdır.
Dezavantaj: Negatif değerlerde türev sıfır olduğu için bazı nöronlar tamamen "ölü" kalabilir (Dead ReLU problem).

Leaky ReLU

$$f(x) = \begin{cases} x, & x > 0 \\ 0.01x, & x \leq 0 \end{cases}$$

ReLU'nun "ölü nöron" problemini çözmek için geliştirilmiştir.
Negatif değerlerde küçük bir eğim bırakır.

Softmax Fonksiyonu

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Genellikle çok sınıflı sınıflandırma problemlerinin çıkış katmanında kullanılır.
Her sınıfa ait olasılık değerlerini verir (toplamı 1'dir).

2. Gelişmiş Makine Öğrenmesi Algoritmaları ve Derin Öğrenme

2.1. Karar Ağaçları ve Topluluk Öğrenmesi (Ensembles)

2.1.1. Information gain (Bilgi Kazancı) nedir?

Bir değişkenin, bir diğer değişkene bağlı olarak değişen koşullu dağılımı ile tek başına olan dağılımı arasındaki farkı (Kullback-Leibler ıraksaması) ölçen ve bu farkın ortalamasını alan bir değerdir. Temelde, bir özelliğin (değişkenin) hedef değişken hakkında ne kadar bilgi sağladığını belirtir.

2.1.2. Tree Ensembles (Ağaç Toplulukları) nedir?

Ağaç Toplulukları (Tree Ensemble), makine öğrenmesinde kullanılan bir denetimli öğrenme tekniğidir. Bu yöntem, tek başlarına yüksek performans göstermeyen ("zayıf öğrenici" veya "temel öğrenici" olarak adlandırılan) karar ağaçlarından oluşan bir koleksiyon bulundurur. Bu zayıf modellerin bir araya gelerek ortak bir tahmin oluşturmasıyla genellikle her birinden çok daha doğru sonuçlar veren güçlü ve yeni bir model meydana getirilir.
Topluluk öğrenmesinin en yaygın üç türü şunlardır:

- → **Torbalama (Bagging)**
  Her biri orijinal veriden rastgele ve yerine koymalı seçilen alt örneklemlerle (bootstrap) eğitilmiş zayıf modellerden (genellikle ağaçlar) oluşur. Bu modellerin tahminleri daha sonra (regresyonda ortalama alınarak gibi) birleştirilir. Random Forest, bu yöntemi temel alan çok başarılı bir ensemble algoritmasıdır.
- → **Yükseltme (Boosting)**
  Boosting (Yükseltme), modelleri sıralı olarak eğiten iteratif bir yöntemdir. Her yeni model, bir önceki modelin hatalı tahmin ettiği örneklere daha fazla odaklanarak (onların ağırlığını artırarak) eğitilir. Böylece her adımda hatalar düzeltilerek daha güçlü bir topluluk modeli oluşturulur. AdaBoost, bu fikri başarıyla uygulayan ilk algoritmalardandır.
- → **Gradyan Yükseltme (Gradient Boosting)**
  Gradient Boosting (Gradyan Yükseltme), boosting'in bir türüdür ancak ağırlık yerine "gradyanlara" veya "sözde artıklara" dayanır. Her yeni model, bir önceki modelin tahmin hatasının (artığının) kendisini tahmin etmeye çalışır. Tüm modeller toplandığında hata en aza indirilmiş olur. XGBoost, LightGBM gibi modern ve güçlü algoritmalar bu yöntemi temel alır.

2.1.3. Random Forest nedir ve Karar Ağacına göre temel avantajı nedir?

Özellik	Karar Ağacı	Random Forest	Avantajın Anlamı
Varyans (Overfitting)	**Çok Yüksektir.** Ağaç, eğitim verisindeki en küçük detayları ve gürültüyü (noise) bile öğrenerek çok karmaşık hale gelir. Eğitimde mükemmel sonuç verirken yeni verilerde kötü performans gösterir.	**Düşüktür.** Çok sayıda ağaç, birbirlerinin hatalarını "ortalama"lar. Bir ağacın öğrendiği gürültü, diğerleri tarafından dengelenir. (Bagging'in etkisi)	Random Forest, eğitim verisi dışındaki, daha önce görmediği verilerde **çok daha kararlı ve güvenilir** tahminler yapar.
Genelleme	**Zayıftır.** Eğitim verisine aşırı uyum sağladığı için genelleme kabiliyeti düşüktür.	**Güçlüdür.** Farklı veri ve özellik alt kümeleriyle eğitilen ağaçlar, verinin altında yatan gerçek kalıbı daha iyi yakalar.	Pratikte, Random Forest neredeyse her zaman tek bir Karar Ağacı'ndan daha yüksek test doğruluğuna sahiptir.
Kararlılık	**Düşüktür.** Eğitim verisindeki küçük bir değişiklik, tamamen farklı bir ağaç yapısı oluşturabilir.	**Yüksektir.** Tek bir ağaç yerine yüzlercesinin ortak kararı kullanıldığı için model çok daha kararlıdır.	Modelin tutarlılığı çok daha yüksektir.
Özellik Önemi	Doğrudan hesaplanabilir ama güvenilir değildir çünkü model kararsızdır.	**Çok daha güvenilir** bir şekilde hesaplanabilir. Model, hangi özelliklerin tahmin için gerçekten önemli olduğunu istatistiksel olarak daha sağlam gösterir.	Veri keşfi ve feature engineering için değerli bir çıktı sağlar.

Tablo 2- Karşılaştırma Tablosu

2.2. Doğal Dil İşleme (NLP) ve Derin Öğrenme

2.2.1. Doğal dil işleme (NLP) nedir, kullanım alanları nelerdir?

Doğal dil işleme (NLP), bilgisayarların insan dilini yorumlamasına, manipüle etmesine ve anlamasına izin veren teknolojidir. Doğal dil işleme, yapay zeka destekli otomasyonun önemli bir özelliğidir ve gerçek zamanlı makine-insan iletişimini destekler.

Kullanım alanları:

- → Sınıflandırma
- → Duygu Analizi
- → Makine Çevirisi
- → Ses Tanıma
- → Yapay Metin Üretme
- → Benzerlik
- → Özetleme
- → İsimlendirilmiş Varlık Tespiti
- → Niyet Tespiti
- → Otomatik Yanıtlama

2.2.2. Yapay Zeka modelleri kelimeleri nasıl anlar, kısaca açıklayınız?

Yapay Zeka modelleri kelimeleri matematiksel vektörler olarak anlar. Kelimeleri sayısal forma dönüştürür ve bu sayılar arasındaki ilişkileri öğrenir. Bu vektörler, yüzlerce boyuttan oluşan soyut bir uzamda konumlanır ve bu uzayda benzer anlamlı kelimeler birbirine yakın, zıt anlamlılar uzak noktalarda bulunur. Örneğin, "kral" ve "kraliçe" vektörleri arasındaki matematiksel fark, "erkek" ve

"kadın" vektörleri arasındaki farkla benzerlik gösterir hatta "kral - erkek + kadın" işlemi "kraliçe" vektörüne çok yakın bir sonuç verir.

2.2.3. Transformers nedir ve NLP alanında neden devrim yaratmıştır?

Transformers, doğal dil işlemede (NLP) çığır açan bir sinir ağı mimarisidir. Önceki mimarilerden (RNN, LSTM) temel farkı, sadece "attention" (dikkat) mekanizmasına dayanması ve ardışık işlemi ortadan kaldırmasıdır.

Transformer modellerinin merkezindeki özellik, bir girdi dizisinin her bir parçası arasındaki ilişkileri (veya bağımlılıkları) tespit etme yeteneğini elde ettiği öz-dikkat (self-attention) mekanizmasıdır. Kendisinden önce gelen RNN ve CNN mimarilerinin aksine, transformer mimarisi sadece dikkat katmanları ve standart ileri beslemeli katmanlar kullanır.

Transformer modellerin ortaya çıkışından önce, çoğu Doğal Dil İşleme (NLP) görevi Özyinelemeli Sinir Ağlarına (RNN) dayanıyordu. RNN'lerin dizisel veriyi işleme şekli doğası gereği seri hâlinde olduğundan girdi dizisinin elemanlarını teker teker ve belirli bir sırayla işlerler.
Bu durum, RNN'lerin uzun menzilli bağımlılıkları yakalama yeteneğini kısıtlar. Bunun anlamı RNN'lerin sadece kısa metin dizilerini etkili bir şekilde işleyebileceğidir.

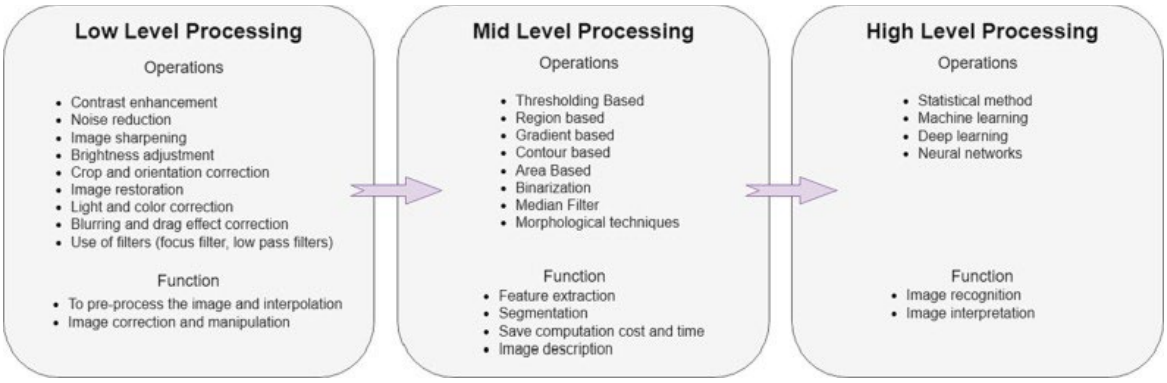
Dikkat mekanizmaları ise bir dizinin tamamını aynı anda inceleyebilir ve o dizinin belirli zaman adımlarına nasıl ve ne zaman odaklanacağına karar verebilir. Uzun menzilli bağımlılıkları anlama yeteneğini önemli ölçüde iyileştirir. Bu şekilde transformerlar, birçok hesaplama adımını seri bir şekilde değil aynı anda gerçekleştirme yeteneğine sahip olur. Paralelleştirmeye uygun olmak, transformer modellerin hem eğitim hem de çıkarım aşamalarında GPU'ların sağladığı güç ve hızdan tam anlamıyla faydalanmasını sağlar.

2.3. Görüntü İşleme ve Derin Öğrenme

2.3.1. Görüntü işleme nedir?

Girdisinin bir fotoğraf veya video karesi gibi bir görüntü olduğu her türlü sinyal işleme biçimidir. Görüntü işlemenin çıktısı ise bir görüntü veya görüntüyle ilgili bir dizi özellik veya parametre olabilir. Çoğu görüntü işleme tekniği, görüntüyü iki boyutlu bir sinyal olarak ele almayı ve standart sinyal işleme tekniklerini uygulamayı içerir.

Visualization - Resimde görünmeyen nesneler bulunur
Recognition - Görüntüdeki nesneler ayırt edilir veya tespit edilir
Sharpening and restoration - Orijinal görüntüden geliştirilmiş bir görüntü oluşturulur
Pattern recognition - Görüntüdeki nesnelerin etrafındaki çeşitli desenler ölçülür
Retrieval - Orijinal görüntüye benzer dijital görüntülerden oluşan geniş bir veritabanından görüntülere göz atılır ve aranır



Resim 3- Görüntü İşleme Seviyeleri

2.3.2. Görüntü işlemede kullanılan algoritmalar nelerdir? (En az 3 popüler algoritmadan bahsedebilirsiniz)

Canny Kenar Tespiti

1986'da John F. Canny tarafından geliştirilen ve birkaç aşamadan oluşan hassas bir kenar tespit algoritmasıdır. İlk olarak, kenar tespitini olumsuz etkileyen gürültüyü azaltmak için görüntüye 5x5'lik bir Gauss filtresi uygulanır. Ardından, yatay ve dikey yönde Sobel filtresi kullanılarak görüntünün yoğunluk gradyanı ve yönü hesaplanır. Sonraki aşamada, 'Non-maximum Suppression' (Maksimum Olmayanı Bastırma) ile kenar olmayan ve kalın görünen kenarlar inceltilerek sadece yerel maksimum gradyana sahip pikseller korunur. Genellikle ek bir adım olarak, histerezis eşikleme ile zayıf ve güçlü kenar pikselleri birleştirilerek kesintisiz ve gerçek kenarların tespit edilmesi sağlanır. Bu adımlar, algoritmanın hem gürültüye dayanıklı hem de yüksek doğrulukta kenarlar bulmasını mümkün kılar.

Histogram Eşitleme

Bir görüntünün kontrastını iyileştirmek ve detayları daha belirgin hale getirmek için kullanılan temel bir görüntü işleme tekniğidir. Bu yöntem, görüntünün mevcut piksel yoğunluklarının dağılımını temsil eden histogramını analiz eder ve bu dağılımı "eşitler". Temel prensibi, orijinal görüntünün yoğunluk değerlerini, birikimli dağılım fonksiyonu (CDF) kullanarak yeniden eşleyerek, çıktı görüntüsünün histogramının mümkün olduğunca düzgün (üniform) bir şekilde tüm yoğunluk aralığına yayılmasını sağlamaktır. Bu süreç, özellikle arka plan ve ön planın birbirine yakın olduğu veya aydınlatmanın yetersiz olduğu görüntülerde, kontrastı önemli ölçüde artırır ve daha önce görünmeyen detayları ortaya çıkarır. Sonuç olarak, görüntünün görsel kalitesi ve analiz edilebilirliği, piksellerin global olarak yeniden dağıtılmasıyla geliştirilmiş olur.

Thresholding (Eşikleme)

Bir görüntüyü ikili (binary) bir forma dönüştürmek için kullanılan temel ve yaygın bir segmentasyon yöntemidir. Amacı, bir görüntüdeki nesneleri arka plandan ayırmaktır. Temel çalışma prensibi, önceden belirlenmiş bir yoğunluk değeri olan eşik değerini (T) kullanmaya dayanır. İşlem, her bir pikselin yoğunluk değerini bu eşik değer ile karşılaştırır. Eşiğin üzerindeki pikseller beyaz (255), altındakiler ise siyah (0) yapılır. Bu, basit global eşikleme için geçerlidir ve formülle ` hedef_piksel =

(piksel > T) ? 255 : 0` şeklinde ifade edilebilir. Ancak aydınlatmanın düzensiz olduğu durumlarda, uyarlamalı eşikleme (adaptive thresholding) gibi daha gelişmiş yöntemler kullanılır. Bu yöntem, eşik değerini pikselin yerel komşuluğuna göre hesaplayarak daha sağlam sonuçlar verir. Eşikleme; uygulamasının basit, hızlı ve hesaplama açısından verimli olması nedeniyle özellikle OCR (Optik Karakter Tanıma), nesne sayma ve endüstriyel kalite kontrol gibi alanlarda sıklıkla bir ön işlem adımı olarak kullanılır.

2.3.3. Auto-Encoder (Oto-Kodlayıcı) nedir ve kullanım amacı nedir?

Auto-Encoder (Oto-Kodlayıcı), denetimsiz öğrenme için kullanılan bir yapay sinir ağı mimarisidir. Veriyi (genellikle görüntü veya özellik vektörlerini) verimli bir şekilde sıkıştırır (kodlar) ve ardından bu sıkıştırılmış halinden orijinaline mümkün olduğunca yakın bir şekilde geri oluşturur (kodunu çöze). Ağ, bir Kodlayıcı (Encoder) ve bir Kod Çözücü (Decoder) olmak üzere iki ana bileşenden oluşur:

Kodlayıcı, yüksek boyutlu giriş verisini alır ve onu çok daha düşük boyutlu, öz bir temsil olan "latent uzay gösterimi"ne (bottleneck) indirger.

Kod Çözücü, bu sıkıştırılmış bilgiyi alarak orijinal giriş boyutunda bir çıktı oluşturmaya çalışır.

Oto-Kodlayıcıların asıl kullanım amacı, veri boyutundan bağımsız olarak verinin en önemli ve anlamlı özelliklerini çıkarmaktır. Bu sayede başlıca dört alanda kullanılırlar:

→ **Boyut İndirgeme:** PCA'ye daha güçlü bir alternatif olarak
→ **Gürültü Giderme:** Gürültülü veriyi temizlenmiş haline dönüştürmek için,
→ **Özellik Çıkarımı:** Latent uzaydaki temsilleri diğer makine öğrenmesi modellerinde girdi olarak kullanmak için
→ **Anomali Tespiti:** Normal verileri öğrenen modelin, farklı yapıdaki anormal verileri kötü bir şekilde yeniden oluşturması prensibiyle çalışır.

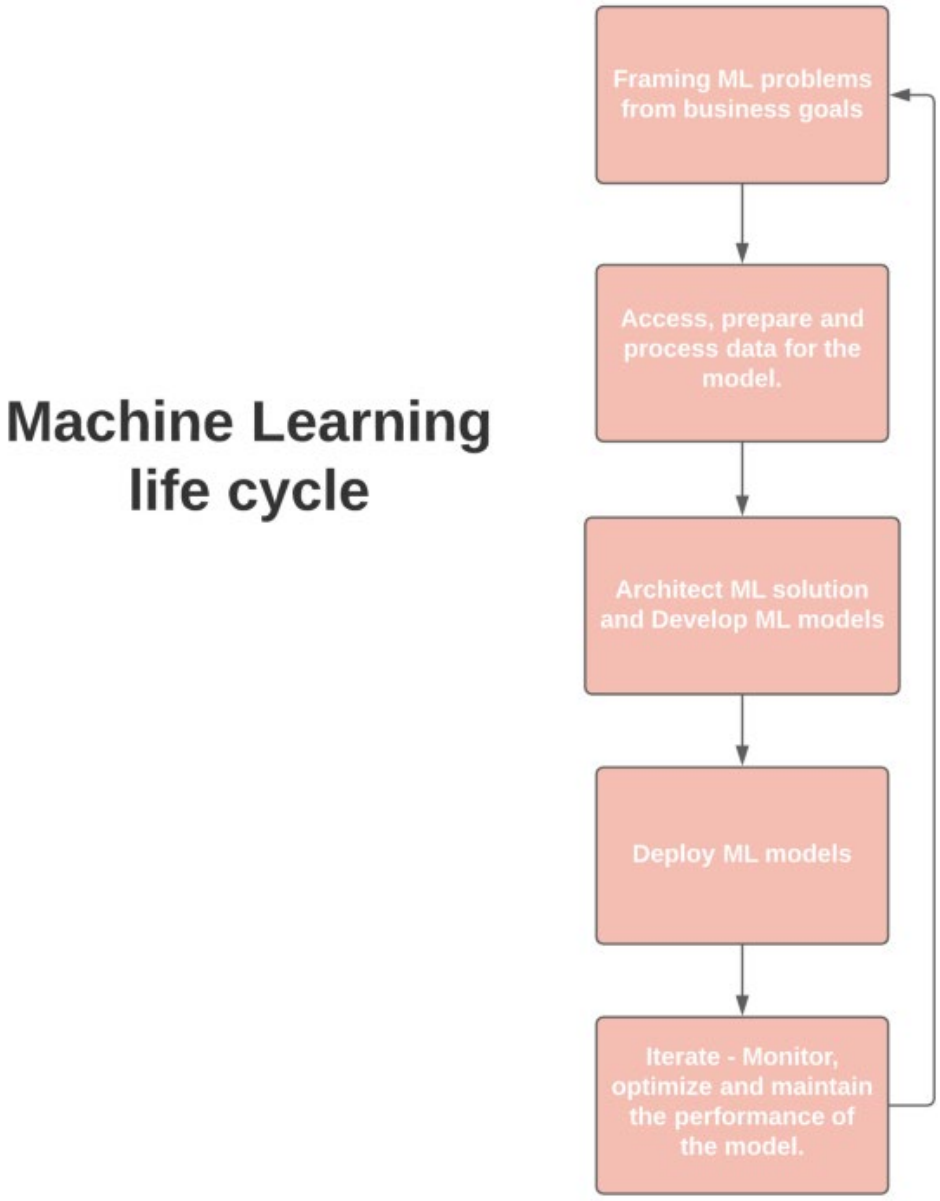
3. MLOps ve Proje Yönetimi

3.1. MLOps Temelleri

3.1.1. MLOps nedir? Kısaca açıklayınız.

MLOps (Machine Learning Operations), makine öğrenmesi modellerinin yaşam döngüsünü yönetmek için kullanılan bir uygulama ve kültür bütünleşmesidir.

3.1.2. Makine öğrenmesi yaşam döngüsünde (ML lifecycle) hangi aşamalar bulunur?



Resim 4- ML Life Cycle

3.1.3. Bir makine öğrenmesi modelini sadece eğitmek yeterli midir? Neden?

Hayır, bir makine öğrenmesi modelini sadece eğitmek kesinlikle yeterli değildir. Bunun başlıca nedenleri şunlardır:

→ **Modelin durağan dünyanın sürekli değişen ve dönüşen halde olması**

Bir makine öğrenmesi modelini sadece eğitmek yeterli değildir, çünkü modelin eğitildiği veriler zamanla gerçek dünyadaki değişimler nedeniyle geçerliliğini yitirebilir. Bu durum "Model Kayması" veya "Veri Kayması" olarak adlandırılır. Örneğin, ekonomik bir kriz döneminde veya tüketici

alışkanlıklarının hızla değiştiği bir periyotta eğitilmiş bir model, yeni koşullar altında güvenilir tahminler yapamaz. Model, artık mevcut gerçekliği temsil etmeyen tarihsel verilere "aşırı uyum sağlamış" duruma gelir ve performansı sessizce düşer.

→ **Deneysel gözlemler ve eğitim süreciyle gerçek hayat uygulamasının uyumsuzluğu**
Model eğitimi genellikle kontrollü ve temiz bir laboratuvar ortamında gerçekleşir. Ancak gerçek dünya, gürültülü, eksik ve tutarsız verilerle doludur. Bir modeli üretim ortamına dağıtmak, onu farklı yazılım sistemleri, donanım kısıtlamaları ve kullanıcı etkileşimleriyle bütünleştirmeyi gerektirir. Eğitim sırasında karşılaşılmayan bu teknik zorluklar, modelin dağıtılmadan önce kapsamlı bir şekilde test edilmesini ve güvenilir bir altyapıya entegre edilmesini zorunlu kılar.

→ **Modelin sürekli bakıma ve izlenmeye muhtaç olması**
Bir modeli dağıtmak, görevin sonu değil, başlangıcıdır. Modelin canlı sistemdeki performansı—tahmin doğruluğu, yanıt süresi ve kaynak tüketimi—sürekli olarak izlenmelidir. Zamanla performansında bir düşüş tespit edildiğinde, modelin yeni verilerle yeniden eğitilmesi veya ayarlanması gerekir. Aksi takdirde, hatalı veya verimsiz tahminler yapmaya devam ederek iş süreçlerine zarar verebilir.

→ **Etik, güvenlik ve regülasyonlara uyumlama sorumluluğu**
Model üretime alındığında, eğitim aşamasında ortaya çıkmamış etik ve güvenlik sorunları belirebilir. Model, gerçek dünya verileri üzerinde adil olmayan önyargılar sergileyebilir veya kötü niyetli saldırılara maruz kalabilir. Özellikle sağlık ve finans gibi sektörlerde, GDPR veya HIPAA gibi veri gizliliği düzenlemelerine uyum sağlamak için model kararlarının şeffaf ve denetlenebilir olması gerekir.

3.1.4. Model versiyonlama (model versioning) nedir ve neden önemlidir?

Model versiyonlama, yazılım değişikliklerini zaman içinde izleme ve kontrol etme sürecidir. İster bir uygulama ister bir makine öğrenimi modeli geliştiriliyor olsun, hataları düzeltmek ve anlaşmazlıkları önlemek için ekip üyeleri tarafından yapılan her değişiklik takip edilmelidir. Makine öğrenmesi projelerinin kaotik bir deney sürecinden, kontrollü ve güvenilir bir üretim sistemine dönüşmesini sağladığından dolayı model versiyonlama son derce önemlidir. Bir modelin tek seferlik bir kod parçası değil sürekli evrim geçiren bir varlık olduğu gerçeği versiyonlamayı zorunlu kılar. Bu sayede hangi modelin hangi veri ve parametrelerle eğitildiği, kimin tarafından değiştirildiği ve üretimdeki performans metriklerei tam olarak izlenebilir. Bu izlenebilirlik olmadan performansı düşen bir modelin neden performanısının düştüğünü tespit etmek veya başarılı bir deneyi tekrarlamak imkansızlaşır. En kritik nokta ise, bir hata tespit edildiğinde sistemin bir önceki kararlı versiyona anında geri dönebilmesidir.

3.2. Proje Yönetimi ve Takım Çalışması

3.2.1. Bir yapay zeka projesinde takım içinde hangi roller bulunur? (örnek: data scientist, data engineer...)

→ Veri Bilimci (Data Scientist)
→ Makine Öğrenmesi Mühendisi (ML Engineer)
→ Veri Mühendisi (Data Engineer)
→ Yazılım Mühendisi (Software Engineer)
→ MLOps Mühendisi (MLOps Engineer)
→ Ürün Yöneticisi (Product Manager)
→ Prompt Mühendisi (Prompt Engineer)
→ Etik / Uyum Uzmanı (Ethics / Compliance Specialist)
→ Proje Yöneticisi (Project Manager)
→ YZ Tasarımcısı (AI Designer)
→ DevOps Mühendisi (DevOps Engineer)
→ Test/UAT Mühendisi (Test/UAT (**User Acceptance Testing**)Engineer)

3.2.2. Bir AI projesi başarısız oluyorsa sence en yaygın sebepler neler olabilir?

Daha önce hiç bir AI projesi yapmadım ve içinde bulunmadım ama şu iki olasılık büyük ihtimalle projeyi batırıyordur. İlki teorik bilgi eksikliği ya da deneyimsizliktir diye düşünüyorum. Eğer proje neyi, ne kadar bildiğini bilmeyen bir kitleyle yapılıyorsa veya neyi, nereye kadar, ne kadar zamanda öğrenip de yapacağını bilmeyen ya da direkt olarak yapmaktan kaçan, projenin çok yüksek ihtimalle sonunu bu getirir. Yeteri kadar bilmese bile öğrenecek kapasitede kişilerle ancak işler bitirilebilir. Başarılı iş/ kişiler görüyorsanız ya zenginlerdir ya dolandırıcılardır ya da yeterince mesai harcamışlardır ve sonunu görmek istemişlerdir.
İkinci en olası gördüğüm ihtimalse iyi bir fikirle yola çıkılmamasıdır. Yani projeyi yapan kişilerin saçma bir fikirle ya da hevesle projenin çok iyi bir amacının olduğu yanılgısına düşmesidir. Zaten yapılmış/ var olan ya da kimsenin kullanmaya ihtiyacının olmadığı ya da yakın gelecekte ölmeye mahkum bir alanda proje yapmaya çalışmak veya sırf havalı olduğu için ve 'Ben yaptım.' diyebilmek için kişilerin boylarını aşan işlere kalkışması ölümcül olacaktır.

3.3. Uygulamalı Python ve Programlama Bilgisi

3.3.1. Bildiğin Python kütüphaneleri nelerdir? (Makine öğrenmesi/veri bilimi alanındaki kütüphaneleri listeleyiniz)

NumPy, Pandas, Matplotlib kullandım ve bunları biliyorum.

3.3.2. Bildiğin yazılım dilleri nelerdir?

C, C++ ve Python orta üst seviyede biliyorum. 2 sene bilgisayar mühendisliği okudum aynı dersleri iki kere almak zorunda kaldım. Derste Java gördüm ama hoşlanmıyorum. İHA yaparken Mission Planner kullandım. Raspberry Pi OS arayüzüne hakim sayılırım. ROS2 Humble ve SLAM ile LIDAR verilerini alarak haritalama yapmayı deniyorum.

3.3.3. Nesneye yönelik programlama (Object-Oriented Programming - OOP) biliyor musun? Biliyorsan temel 4 prensibini (Encapsulation, Inheritance, Polymorphism, Abstraction) kısaca açıklar mısın?

Evet; Java, Python ve C++ öğrenirken nesne yönelimli kısımlardan çokça bahsedildi. İlk NYP dersimi Java üzerinden almıştım şu an Python üzerinden nesne yönelimli programlama dersi alıyorum.

Nesne yönelimli programlama gerçek hayattaki nesneleri ve onların birbirleriyle olan ilişkilerini bilgisayar programlarına modellemek için kullanılan bir programlama paradigması veya modelidir.

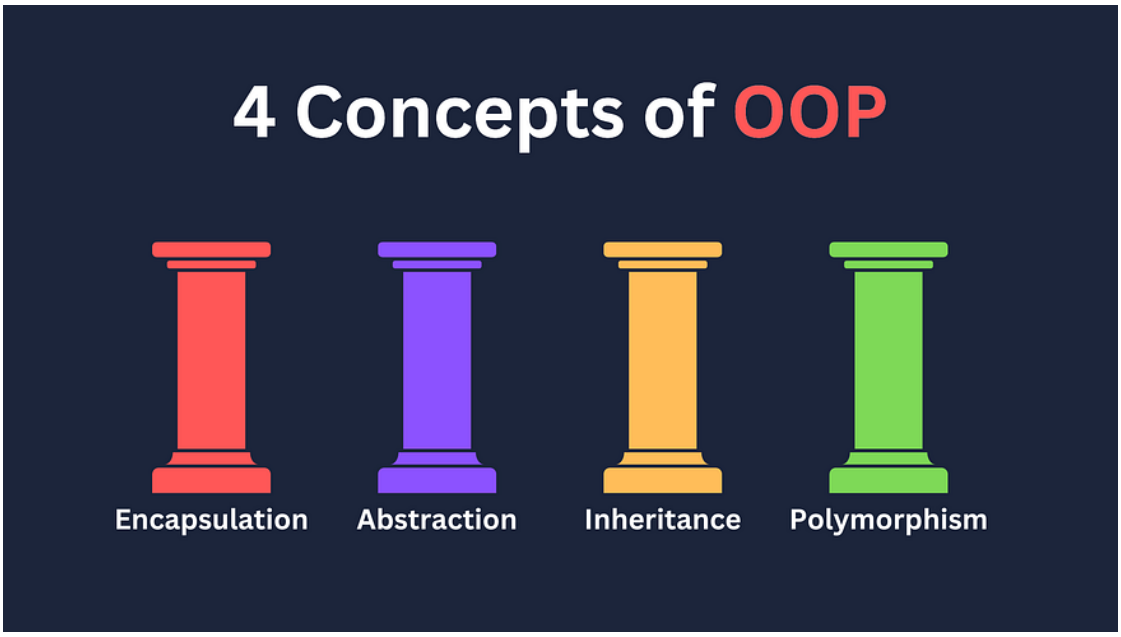
NYP'de temel fikir, programları birbiriyle etkileşim halinde olan "nesneler" topluluğu olarak düşünmektir. Bu, geleneksel, sırayla çalışan (prosedürel) programlamaya bir alternatiftir.
Bu dört temel prensibin basitçe amacı şunlardır:

Encapsulation, veri ve bu veriyi işleyen metodları bir arada tutarak ve veriye doğrudan erişimi kısıtlayarak sadece kontrollü yollarla değiştirilmesini sağlamak.

Inheritance, bir sınıfın başka bir sınıftan özellik ve davranışlarını miras alarak kod tekrarını önlemek.

Polymorphism, aynı arayüzün farklı nesneler tarafından farklı şekilde implemente edilmesini sağlamak.

Abstraction, karmaşık sistemi basitleştirmek, sadece gerekli detayları göstermek.



Resim 5- Nesne Yönelimli Programlamanın 4 Temel Konsepti

3.3.4. Aşağıdaki iki Python kütüphanesinin (Pandas ve NumPy) veri bilimindeki temel görevlerini ve birbirlerine göre avantajlarını karşılaştırınız.

Pandas ve NumPy, Python'un veri bilimi alanında kullanılan iki temel kütüphanedir. NumPy, öncelikle sayısal hesaplamalar ve bilimsel işlemler için tasarlanmış olup çok boyutlu diziler üzerinde yüksek performanslı matematiksel operasyonlar sunar. Matris işlemleri, lineer cebir uygulamaları ve istatistiksel hesaplamalarda yüksek performans gösterir. Pandas ise daha çok veri manipülasyonu ve analizi üzerine odaklanmıştır. DataFrame yapısı sayesinde tablosal verileri etkili bir şekilde işleyebilir, eksik verileri yönetebilir ve zaman serisi analizleri yapabilir. NumPy'nin aksine, Pandas heterojen veri tiplerini destekler ve veri temizleme, gruplama, filtreleme gibi karmaşık veri işleme görevlerinde daha esnektir. Performans açısından bakıldığında, NumPy saf matematiksel işlemlerde daha hızlıyken, Pandas veri manipülasyonu ve okunabilirlik açısından daha kullanıcı dostudur. Gerçek dünya veri bilimi projelerinde genellikle bu iki kütüphane birlikte kullanılır.

3.4. Deneyim ve Katılım Bilgileri

3.4.1. Daha önce yaptığın projelerden bahseder misin?

Bireysel olarak yaptığım projeler daha çok derslerden istenenlerdi. Adam asmaca oyunu web sitesi, planner uygulaması, çiftlik oyunu arayüzü vs. gibi şeylerdi. Yarışma özelinde çalıştım daha çok o yüzden kendi başıma yaptığım kayda değer bir projem yok.

3.4.2. Katıldığın yarışmalar varsa nelerdir, başarı sıralamaların nelerdir? (Başarı sıralaması ekibe alımda kriter olarak değerlendirilmeyecektir)

3 senedir Teknofest'e katılıyoruz. Bir sanayi aracımız bir de İHA'mız var. 2023'te Sanayide Dijital Teknolojiler Yarışması birincisi olan takımdaydım. 2024'te aynı yarışmada farklı takımla En Özgün

Tasarım ödülü aldık. 2025'te Uluslararası İHA Yarışması'nda takım lideriydim. Finalist olduk. Genel olarak görüntü ve video işleme işleriyle ilgileniyordum.

3.4.3. Her hafta Salı günleri saat 16.30'da Yıldız Teknik Üniversitesinde yüz yüze yapılacak olan toplantılara katılabilir misin?

Kampüsüm Davutpaşa'da olmadığı için emin değilim ama çoğunlukla katılmaya çalışırım.

4. Kaynakça

[1] https://www.coursera.org/articles/data-analytics
[2] https://aws.amazon.com/what-is/data-analytics/
[3] https://www.geeksforgeeks.org/machine-learning/what-are-outliers-in-data/
[4] https://www.kdnuggets.com/dealing-with-outliers-a-complete-guide
[5] https://thedocs.worldbank.org/en/doc/20f02031de132cc3d76b91b5ed8737d0-0050012017/related/lecture-12-1.pdf
[6] https://webspace.ship.edu/pgmarr/Geo441/Readings/Grubbs%201969%20-%20Detecting%20outlying%20observations%20in%20samples.pdf
[7] https://www.ibm.com/think/topics/data-leakage-machine-learning
[8] https://medium.com/@tyagi.lekhansh/handling-missing-values-in-machine-learning-strategies-for-imputation-and-model-robustness-4ba6287f1094
[9] https://chatgpt.com/
[10] https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/
[11] https://www.masraff.co/makine-ogrenmesi/
[12] https://www.oracle.com/tr/artificial-intelligence/machine-learning/what-is-machine-learning/
[13] https://medium.com/@anishnama20/understanding-cost-functions-in-machine-learning-types-and-applications-cd7d8cc4b47d
[14] https://www.geeksforgeeks.org/data-science/what-is-gradient-descent/
[15] https://www.geeksforgeeks.org/machine-learning/impact-of-learning-rate-on-a-model/
[16] https://en.wikipedia.org/wiki/Information_gain_(decision_tree)
[17] https://aws.amazon.com/tr/what-is/nlp/
[18] https://yapayzeka.itu.edu.tr/arastirma/dogal-dil-isleme
[19] https://www.sciencedirect.com/topics/computer-science/image-processing
[20] https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html
[21] https://medium.com/@murat_sivri/mlops-nedi%CC%87r-2356464eb93e