Makine Öğrenmesi
Giriş

Dr. Cahit Karakuş

“Balbiti”

İçindekiler

Kısaltmalar

AI: Artificial Intellegince
ML: Machine Learning
DL: Deep Learning
API: Application programming interfaces

Giriş

Fiziksel değişimlerin algılanması, veriye dönüştürülmesi, toplanması, sınıflandırılması, birleştirilmesi ve kıyaslama yapılarak otonom davranış elde edilmesi nasıl mümkün olabilecek? Otonom davranışda doğru karar verilmesinin nasıl yapılacağının iyi anlaşılması için, insan ve hayvanların bu işlevleri nasıl yaptığının çok iyi araştırılması gerekmektedir; Baykuşlar bir farenin ayak sesini 1km öteden algılar. Filler çok uzaklardan gelen sismik sinyalleri hisseder. Yaban arıları yuvalarını tahrip edebilecek tehditleri önceden fark ederek, topluca saldırıya geçerler ve düşmanlarını kilometrelerce kovalarlar. Hayvanların birbirleri ile bilgi paylaştıkları ve risk analizi yaptıkları da gözlenmiştir. Bunlara benzer yüzlerce örnek verebiliriz, burada önemli olan, hayvanların bu yetenekleri nasıl kazandıkları, nasıl algıladıkları, nasıl kestirim yaptıklarıdır.

Bilgisayar sistemleri öğrenmeye başladılar ve karar veriyorlar, robotlar dünyayı ele geçirecekler mi? Enerji problemi nasıl çözülecek? Yazılan komutlar ile robot sürüsü neler yapmaz ki? Robotlardan birşeyler yapılması isteniyorsa, kodların yazılması ve uygulamasının görülmesi gerekir.

Devasa veri yığınları ve makine öğremesi algoritmaları ile gelecekte hangi sektörlerde organize, otonom ve gezgin makineler yoğun olarak kullanılacaktır? İzlerimizden kimliğimizin belirlenmesine izin veriyoruz.

ML, verileri işleyerek görevlerin nasıl gerçekleştirileceğini "öğrenebilen" makineleri ifade eder. Öğrenme, deneyim yoluyla bilgi, anlayış ya da beceri kazanarak davranışsal eğilimin iyileştirilmesi, düzeltilmesi, güncellenmesidir. Öğrenmenin sözlük anlamı, bilgi edinmek, çalışmak, deneyimlemek suretiyle anlama, beceri kazanma gibi ifadelerin ve deneyimle **davranışsal eğilimlerin değiştirilmesini** içerir. Makine öğrenmesi alanında araştırmacılar tarafından keşfedilen kavram ve tekniklerin biyolojik öğrenmenin bazı yönlerini aydınlatabileceği de muhtemel görünüyor. Öte yandan biyolojik öğrenme metotları da makinlerin öğrenmesine inanılmaz katkı vereceği öngörülmektedir.

Yapay zekanın en aktif olarak kullanıldığı alan kuşkusuz robot teknolojileridir. Yapay zekanın gelişmesi robot teknolojilerinin gelişimini de doğrudan etkiledi. Robotlarda gerçekleşen performans problemlerini kolay bir şekilde algılayabilen yapay zeka, ihtiyaç halinde sorunları giderebiliyor. Böylece robotlar kendini yenileyebiliyor.

Son yıllarda taşımacılık alanında yaşanan en büyük gelişme sürücüsüz araçların geliştirilmesi oldu. Google, Tesla, Uber gibi büyük firmalar bu alana önemli yatırımlar yaptı ve sürücüsüz araç teknolojileri ivme kazandı. Sürücüsüz araç teknolojilerinin en büyük destekçisi ise yapay zeka teknolojisidir. Sürücüsüz araçların yanı sıra drone teknolojileri de yapay zekadan

faydalanıyor. Sürü drone teknolojilerinde, maden ocakları (Gelecekte göktaşlarını yağmalamada), savaş. Tıp teşhis, tedavi ve müdahalede yoğun olarak kullanılmaya başlanıldı.

Makine öğrenmesinde:

- Öğrenme Görevi: veri yığınından ne öğrenmek ya da tahmin etmek istiyoruz?
- Veriler ve varsayımlar: Elimizde hangi veriler var? Kaliteleri nedir? Verilen problem hakkında ne varsayabiliriz?
- Temsil: Sınıflandırılacak örneklerin uygun bir temsili nedir?
- Yöntem ve Tahmin: Olası hipotezler var mı? Tahminlerimizi verilen sonuçlara göre ayarlayabilir miyiz?
- Değerlendirme: Yöntem ne kadar iyi performans gösteriyor? Başka bir yaklaşım / model daha iyi performans gösterebilir mi?
- Sınıflandırma: Bir öğe sınıfının tahmini.
- Öngörü: Bir parametre değerinin tahmini.
- Karakterizasyon: Öğe gruplarını tanımlayan hipotezler bulun.
- Kümeleme: (Atanmamış) veri kümesinin ortak özelliklere sahip kümelere bölünmesi. (Denetimsiz öğrenme)

Bu derste, Matematiğe geri döneceğiz (Türev, integral, ikinci dereceden diferansiyel denklemler, lineer denklem sistem, matris) ve istatistikleri, olasılık yoğunluk fonksiyonlarını inceleyeceğiz ve veri kümelerine dayalı önemli sayıların nasıl hesaplanacağımızı, İhtiyacımız olan cevapları almak için çeşitli Python modüllerini nasıl kullanacağımızı, ve öğrendiklerimize dayanarak sonucu tahmin edebilen fonksiyonları nasıl yapacağımızı öğreneceğiz.

1. Makine Öğrenmesi Nedir?

İçinde bilgisayar sistemi bulunan makinelerin minimum insan müdahalesi ile bir görevi öğrenmesini sağlamak için verilere, algoritmalara ve matematiksel modellere odaklanan bir yapay zeka dalına Makine öğrenimi denir.

Makine öğrenimi, deneyimle veri yığınından otonom öğrenmek ve performansını geliştirmek için sistem programlamayla ilgilenen bir bilgisayar bilimi dalıdır. Örneğin: Robotlar, sensörlerden topladıkları verilere göre görevi yerine getirebilecek şekilde programlanır. Makine öğrenmesi ile verilerden programları otonom olarak öğrenir.

Makine Öğrenmesi, veri yığınından yorumlamaya, tahmin etmeye ya da karar vermeye yönelik otonom davranış paternleri geliştiren algoritmalar ve matematiksel modellerin oluşturulmasıdır. Veri yığınından kendi kendine öğrenen matematiksel modeller ve algoritmalar ile insandan bağımsız otonom davranış geliştirilmesidir.

Makine öğrenimini tanımlamanın birçok yolu vardır:

1) **Yapay zeka görünümü:** Öğrenme, insan bilgisi ve zekasının merkezinde yer alır ve aynı şekilde akıllı makineler inşa etmek için de gereklidir. Yapay zeka konusunda yıllarca süren çaba, tüm kuralları programlayarak akıllı bilgisayarlar oluşturmaya çalışmanın yapılamayacağını göstermiştir; otomatik öğrenme çok önemlidir. Örneğin, biz insanlar dili anlama becerisiyle doğmadık - onu öğreniyoruz - ve hepsini programlamaya çalışmak yerine bilgisayarların dili öğrenmesini sağlamaya çalışmak daha mantıklı.

2) **Yazılım mühendisliği görünümü:** Makine öğrenimi, bilgileri örnekleyerek programlamamıza olanak tanır; bu, geleneksel şekilde kod yazmaktan daha kolay olabilir.

3) **İstatistiksel olasılık görünümü:** Makine öğrenimi, bilgisayar bilimi ve istatistiğin birleşimidir: hesaplama teknikleri istatistiksel problemlere uygulanır. Makine öğrenimi, tipik istatistik problemlerinin ötesinde, birçok bağlamda çok sayıda soruna uygulanmıştır. Makine öğrenimi genellikle istatistiklerden farklı değerlendirmelerle tasarlanır *(Örneğin, hız genellikle doğruluktan daha önemlidir)*.

Bugün elimizde olan teknoloji, onun büyük bir ivme kazanmasına ve bildiğimiz ve güvendiğimiz aşağıdaki ürünler de dahil olmak üzere birçok günlük kullanımda uygulanmasına izin verdi:

- Kendi kendine giden sürücüz arabalar
- Netflix (nasıl yapıldığını görün) ve Amazon Prime Video Önerileri
- Dolandırıcılık tespiti

- Konuşma tanıma
- Çevrimiçi müşteri hizmetleri sohbet kutuları

Makine öğrenimi mühendisleri, yapay zeka sistemleri oluşturmak için veri bilimi ve yazılım mühendisliğini birleştirir. Makine öğrenimi mühendisi, veri bilimcisi veya derin öğrenme uzmanından yapay zeka modelini alır ve üretime uygular. Başka bir deyişle, veri modelleri ile yazılım arasındaki köprüdür, kendi kendine çalışan programlar ve daha fazla programlamaya ihtiyaç duymadan öğrenebilen algoritmalar oluştururlar. Bilgisayar mühendisleri öğrenebilen algoritmalar nasıl oluşturulur bilmek zorundalar.

Makine öğrenimi, kendi kendine öğrenen ve gelişen sistemler oluşturma sürecidir. Makine öğreniminin nihai amacı, bir sistemin otomatik olarak veri toplamasına ve bu verileri daha fazlasını öğrenmek için kullanmasına yardımcı olan algoritmalar tasarlamaktır. Sistemlerin, toplanan verilerdeki kalıpları araması ve bunları kendileri için hayati kararlar almak için kullanması beklenir. Genel olarak, makine öğrenimi, sistemlerin insanlar gibi düşünmesini ve hareket etmesini, insan benzeri zeka göstermesini ve onlara bir beyin vermesini sağlamaya çabalar.

Makine Öğrenimi Tanımları:
Makine öğrenimi, Yapay Zekanın heyecan verici bir alt dalıdır ve etrafımızdadır. Makine öğrenimi, verilere otomatik olarak erişebilen ve tahminler ve algılamalar yoluyla görevleri gerçekleştirebilen bilgisayar programları geliştirerek bilgisayar sistemlerinin deneyimlerden öğrenmesine ve gelişmesine yardımcı olur.

Makine Öğrenimi Nasıl Çalışır?
Makine Öğrenimi süreci, seçilen algoritmaya eğitim verilerinin girilmesiyle başlar. Algoritmanın doğru çalışıp çalışmadığını test etmek için farklı giriş verileri makine öğrenme algoritmasına beslenir. Tahmin ve sonuçlar eşleşmezse, veri bilimcisi istenen sonucu elde edene kadar algoritma birden çok kez yeniden eğitilir. Bu, makine öğrenimi algoritmasının sürekli olarak kendi kendine öğrenmesini ve zaman içinde doğruluğunu kademeli olarak artırarak en uygun yanıtı üretmesini sağlar.

Makine öğrenmesi yaklaşımları, öğrenme sistemi için mevcut olan "sinyal" veya "geri bildirim" in doğasına bağlı olarak geleneksel olarak üç geniş kategoriye ayrılır:

Makine öğrenmesi, bir öğrenme algoritmasının, veri yığını içinde öğrenme veri setini deneyimledikten sonra görevler üzerinde doğru bir şekilde performans gösterebilmesidir. Öğrenme örnekleri, genel olarak bilinmeyen bazı olasılık dağılımından yeni durumlarda yeterince doğru tahminler üretmesini sağlayan genel bir model oluşturmalıdır.

Makine öğrenmesi algoritmalarının hesaplamalı analizi ve performansları, **hesaplamalı öğrenme teoris**i olarak bilinen teorik bilgisayar biliminin bir dalıdır. Öğrenme setleri sonlu ve gelecek belirsiz olduğundan, öğrenme teorisi genellikle algoritmaların performansının garantisini vermez. Bunun yerine, performans üzerindeki olasılık sınırları oldukça yaygındır. Eğilim-varyans ayrışması, genelleme hatasını ölçmenin bir yoludur.

Genelleme bağlamında en iyi performans için, hipotezin karmaşıklığı, verilerin altında yatan fonksiyonun karmaşıklığıyla eşleşmelidir. Hipotez işlevden daha az karmaşıksa, model verileri altına yerleştirmiştir. Yanıt olarak modelin karmaşıklığı artarsa, öğrenme hatası azalır. Ancak hipotez çok karmaşıksa, model aşırı uyuma tabidir ve genelleme daha zayıf olacaktır.

Performans sınırlarına ek olarak, öğrenme kuramcıları öğrenmenin zaman karmaşıklığını ve fizibilitesini inceler. Hesaplamalı öğrenme teorisinde, bir hesaplama polinom zamanında yapılabiliyorsa uygulanabilir olarak kabul edilir. İki tür zaman karmaşıklığı sonucu vardır. Olumlu sonuçlar, polinom zamanında belirli bir fonksiyon sınıfının öğrenilebileceğini göstermektedir. Olumsuz sonuçlar, bazı sınıfların polinom zamanında öğrenilemeyeceğini göstermektedir.

Yapay Zeka, Makine Öğrenmesi ve Derin Öğrenme Arasındaki Temel Farklar

Profesör Andrew Moore tarafından tanımlanan yapay zeka (AI), yakın zamana kadar insan zekası gibi bilgisayarların davranış geliştirmesini sağlayan matematik ve mühendislik alanıdır.
Bunlar şunları içerir:

- Bilgisayar görüşü
- Dil İşleme
- Yaratıcılık
- Özetleme

Profesör Tom Mitchell tarafından tanımlandığı gibi, makine öğrenimi, bilgisayar programlarının deneyim yoluyla otomatik olarak öğrenmesine izin veren bilgisayar algoritmaları çalışmasına odaklanan yapay zekanın bilimsel bir dalını ifade eder.
Bunlar şunları içerir:

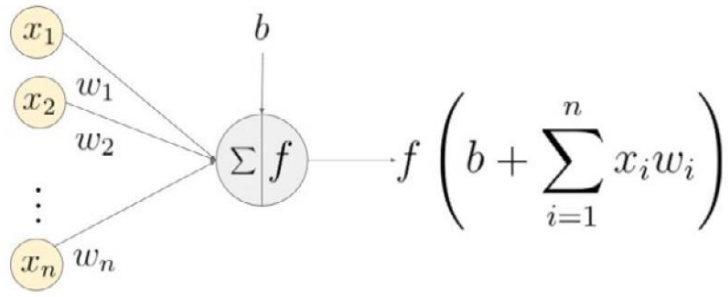
- Sınıflandırma
- Sinir ağı
- Kümeleme

Yapay zekâ, ister makine öğrenmesi kullansın ister kullanmasın herhangi bir tahmin veya karar işlemini gerçekleştiren teknolojilerin genel adıdır. Genel kanaatin aksine yapay zekâ makine öğrenmesi veya derin öğrenme algoritmaları olmaksızın da çalışan bir algoritma olabilir.

Makine öğrenmesinde algoritmalar tamamen veriden öğrenir. Makine öğrenmesini hard-coded olarak kodlanmış sembolik yapay zekâ algoritmalarından ayıran özellik algoritmanın tamamen veriden öğrenmesidir. Doğal olarak akla makine öğrenmesi algoritmalarının verilerden öğrenmesinin riskli tarafları gelebilir. Sorulara yanıt veren yazılımsal makine öğrenmelerine ahlaksız ve ırkçı söylemler sıkça sorulursa, günün sonunda yazılımın kendisi sapık ve ırkçı bir karaktere bürünebilmektedir.

Derin öğrenme modeli, verinin yapısına göre filtreleme yapmak amacıyla hangi parametrelere ne ağırlık verileceğini kendisinin keşfetmesidir. Derin öğrenme algoritması da veriye dayalı öğrenme gerçekleştirmekle birlikte, öğrenme süreci standart makine öğrenmesi algoritmalarında olduğu gibi tek bir matematiksel modele değil sinirsel ağ (neural network) olarak ifade edilen ağ diyagramlarına benzeyen yapıda geliştirilen hesaplamalarla çalışmaktadır. Her derin öğrenme algoritması bir makine öğrenmesi algoritmasıdır çünkü verilerden öğrenme gerçekleştirmektedir. Ancak her makine öğrenmesi algoritması derin öğrenme algoritması değildir; nitekim derin öğrenme, makine öğrenmesinin spesifik bir türüdür. Derin öğrenme, Yüksek bilgi işleme gücü ve büyük veri kümeleriyle birlikte katmanlı yapay sinir ağlarının güçlü matematiksel modelleri oluşturabildiği bir makine öğrenimi alt kümesidir.

Örnek: n giriş değerlerinin ağırlık değerleri (w) ile çarpılarak elde edilen fonksiyonun birde bağımsız değişken girişi olsun.



Bu algoritma öncelikle ağırlık değerleri (w)'yi tesadüfi olarak atar, bu değerlerin sonuçları ile gerçek sonuçlar arasındaki sayısal farka göre tıpkı kendini yeniler. Eldeki verilerle etiketler arasındaki ilişkiyi en iyi ifade eden konfigürasyon elde edildiğinde ise durur. Elde edilen konfigürasyonda artık w değerleri bilindiğinden algoritma belirli bir doğruluk gücünde tahmin yapar. Davranış analiz edilmiş oldu. Geleceğe yönelik öngörüde bulunurken w değerleri değişecektir.

Derin Öğrenme algoritmalarının standart makine öğrenmesi algoritmalarına göre en büyük üstünlüğü feature engineering adı verilen süreci gerektirmemesidir. Bu durum bir örnekle şöyle ifade edilebilir. Diyelim ki bir makineye ait 100 parametre ile karar verme tahmini yapan bir algoritma olsun. Standart makine öğrenmesi algoritmasında hangi parametrelerin

işe yarar olduğu ve modele konması gereken katsayının zarfı verilmek zorundadır. Bunun için teoriden ya da parametre seçim algoritmalarından faydalanabilir. Bu süreç feature engineering olarak adlandırılmaktadır. Derin öğrenmede ise bu süreç yoktur**. Derin öğrenme modeli, verinin yapısına göre hangi parametrelere ne ağırlık verileceğini kendisi keşfetmektedir.** Derin öğrenmenin bu özelliği bir makine öğrenmesi algotimasını daha esnek hale getirmekte ve ürüne dönüşümünde daha fazla kolaylık sağlamaktadır. Bugünlerde ürüne dönüşmüş olan insansız otomobiller gibi teknolojilerin ardında ise derin öğrenme vardır. Derin öğrenme algoritmalarının makine öğrenmesi algoritmalarına göre dezavantajı ise ihtiyaç duyulan donanım ve veri kapasitesidir.

Yapay zeka, Makine Öğrenimi ve Derin Öğrenme birbirinden nasıl farklıdır?

AI, ML ve Deep Learning arasındaki fark aşağıdaki tabloda verilmiştir:

Artificial Intelligence	Machine Learning	Deep Learning
Yapay zeka terimi ilk olarak 1956 yılında John McCarthy tarafından icat edildi.	ML terimi ilk olarak 1959 yılında Arthur Samuel tarafından kullanılmıştır.	DL terimi ilk olarak 2000 yılında Igor Aizenberg'de ortaya çıktı.
İnsan davranışını taklit edebilen akıllı makineler oluşturmak için kullanılan bir teknolojidir.	Geçmiş verilerden ve deneyimlerden öğrenen bir AI alt kümesidir.	Nöronlar olarak adlandırılan insan beyin hücrelerinden ilham alan ve insan beyninin çalışmasını taklit eden makine öğrenimi ve yapay zekanın alt kümesidir.
AI tamamen yapılandırılmış, yarı yapılandırılmış verilerle ilgilenir.	ML, yapılandırılmış ve yarı yapılandırılmış verilerle ilgilenir.	Derin öğrenme, yapılandırılmış ve yapılandırılmamış verilerle ilgilenir.
Çalışmak için büyük miktarda veri gerektirir.	Derin öğrenme ve yapay zekaya kıyasla daha az miktarda veri ile çalışabilir.	ML'ye kıyasla çok büyük miktarda veri gerektirir.
Yapay zekanın amacı, makinenin herhangi bir insan müdahalesi olmadan düşünmesini sağlamaktır.	Makine öğreniminin amacı, makinenin geçmiş deneyimlerden öğrenmesini sağlamaktır.	Derin öğrenmenin amacı, insan beyni gibi karmaşık problemleri çeşitli algoritmalar kullanarak ve filtre kullanarak çözmektir.

1.1. Veri Bilimcileri

Veri bilimcileri kısmen matematikçi, kısmen bilgisayar bilimcisi ve kısmen trend belirleyicidir. Hem iş hem de BT dünyasında çalışırlar ve bu da onları değerli bir çalışan yapar. Aslında, veri bilimcileri 2020'de Amerika'da bir numaralı işti ve bu iş yavaşlamıyor. Ve rekabet eksikliği sayesinde, bir veri bilimi kariyeri çok kazançlı bir seçenektir.

Bir veri bilimcisi kariyeri ilginizi çekiyorsa, izlemeniz gereken bir kılavuz:

1. Ortak programlama dilleri hakkında bir anlayış geliştirin: Python, C++, Java Script, Matlab, Asemble
2. Lisans derecenizi tamamlayın

İstatistiksel olasılık, bilgisayar bilimi, bilgi teknolojisi, matematik ve hatta varsa veri bilimidir.

3. Giriş seviyesi bir işte çalışın
4. Bir yüksek lisans veya doktora derecesi kazanın
5. Öğrenmeye ve çok çalışmaya devam edin

Yetenek gerekli:

- Olasılık ve istatistik
- Matematik ve cebirsel matematik
- Programlama
- Veritabanı Yönetimi
- Makine öğrenimi ve derin öğrenme
- Veri görüntüleme

1.2. Makine Öğrenimi Mühendisliği

Makine öğrenimi mühendisleri, sanal asistanlar, sohbet robotları, önerilen aramalar, çeviri uygulamaları, sürücüsüz arabalar ve diğer teknolojiler için tahmine dayalı modelleri otomatikleştirmek için yapay zeka sistemleri tasarlar. Dolayısıyla sorumlulukları, üzerinde çalıştıkları projenin türüne göre değişebilir. Örneğin, sanal asistanlarla çalışan bir makine öğrenimi uzmanı, temel olarak konuşma tanıma teknolojileri, doğal dil işleme ve makine çevirisine odaklanacaktır. Bu arada, kendi kendini süren araba teknolojisi üzerinde çalışan makine öğrenimi mühendisleri, bilgisayarla görme ve pekiştirmeli öğrenmeye odaklanır.

Makine öğrenimi mühendisinin ana rolleri ve sorumlulukları

- Akıllı Algoritmaları, veri yapılarını ve ayrıca bilgisayar mimarisini kullanarak makine öğrenimi sistemleri tasarlar ve geliştirir
- Hesaplamalar ve istatistiksel analiz gerçekleştirir
- Makine öğrenimi yazılımını devreye alır ve izler
- Makine öğrenimi algoritmalarını ve araçlarını uygular
- Veri oluşturur ve ardışık düzenleri modeller
- Kodu üretime getirmek için gereken veri kümelerini ve veri hatlarını yönetir
- Mevcut ML altyapısını iyileştirmek için en iyi uygulamaları araştırır ve uygular

2. Makine Öğrenmesi Uygulama Adımları

Makine ve Derin öğrenmenin en önemli özelliği, büyük miktarda veri ve bilgisayar gücünün kullanılmasıdır. Yapay zekanın birincil hedefi, öğrenen ve kendi kendini eğiten programlar, deneyimleri birbirleri ile paylaşan sistemler veya diğer dijital ürünler veya ek eğitim olmadan iş amaçlı kullanılabilecek modellerin oluşturulmasıdır. Veri bilimcileri, çeşitli biçimlerde çok sayıda yapılandırılmış veya yapılandırılmamış verileri işlemek için modeller uygular ve algoritmalar kullanırlar; böylece mühendisler, veri kümelerindeki kalıpları belirleyebilir ve pekiştirmeli öğrenme yoluyla yapay zekayı geliştirebilir ve "eğitebilir". Ayrıca, veri kümelerindeki belirli kayıtları kategorilere ayırabilir ve minimum insan müdahalesi ile bunu yapacak bir yapay zeka yazılımı oluşturabilirler.

Bilgi işlem sistemleri ve yapay zeka teknolojilerinin en yaygın pratik kullanım alanları:

- Bilgi işlem sistemlerindeki sunucularda biriken büyük veri yığının analiz edilme süreci yapay zeka uygulamalarındaki gelişmeleri hızlandırmıştır.
- Gerçek zamanlı olarak verinin kişiselleştirilmesi ile kendi kendine öğrenen ve yeni bilgilere uyum sağlayan ve akıllı "aracılar" (uygulamalar ve diğer yazılım ürünleri) kullanan **yapay zeka sistemleri (ML) geliştirilmiştir.**
- **Derin öğrenme:** Algoritmik kümelere (Filtreleme, sınırlandırma) dayanan ve temel mimarisi olarak sinir ağlarını kullanan teknolojidir.
- **Veri madenciliği:** Büyük veri tabanlarındaki bilgilerden eğilimleri ve diğer ilişkileri kuran istatistiksel yöntemlerdir.
- **Vaka temelli muhakeme:** Gelecekteki sorunları çözmek için geçmiş sonuçlardan gelen girdileri kullanan algoritmik yaklaşımlardır.
- **Bulanık mantık:** Verileri siyah-beyaz veya doğru-yanlış gibi ikili kalıplardan ziyade denge çizgisine göre kategorize edebilen sistemler.

Başlamak için Ne Gerekiyor?

İlk adım, yapay zeka yazılımını nasıl geliştireceğimizi değil, bunu neden yapacağımızı ve başlamak için neye ihtiyacımız olduğunu açıklığa kavuşturmaktır. Bu nedenle, ilk aşamada sunulan problem çözme fikrinden yola çıkılarak, hedeflenen ürün kullanıcılarının sorunlu noktaları ve değer önerileri belirlenir. Bu noktalar doğru bir şekilde çerçevelenirse, kullanılacak teknikler, tüm geliştirme adımları ve iş metrikleriyle bağlantılı yapay metrikleri belirlenebilir. Ayrıca bu adım, doğru veri hazırlığı için gereklidir.

Bir yapay zeka yazılımı oluşturmak için öncelikle bir veri yığınına ihtiyaç olunacaktır (Uygulama için veri). Bu aşamada en iyi bilinmesi gereken konu akıllı algoritma oluşturmadır. Müşterinin hazır bir projesi yoksa, proje oluşturulması, aranması ve bazen de etiketlenmesi için zaman sağlamalıdır. Veri kümesini etiketlemek için geliştirme etiketleme ekibine ihtiyaç duyulacaktır.

Herhangi bir yapay zeka uygulamasının temeli bir modeldir. Başlamadan önce yapılması gerekenler:

- Bir algoritma ve öğrenme türü seçilmelidir (denetimli, denetimsiz, pekiştirme).
- Verilerin organize edildiğinden, temizlendiğinden ve tutarlı olduğundan emin olunmalıdır.
- Kronolojik sıra tanımlanır, etiketler eklenir, vb.
- Verileri hazırladıktan sonra yapay zeka yazılımı oluşturmak için platform ve programlama dilleri belirlenir.

Platformlar

Yapay zeka uygulama geliştiricileri için ürün oluşturmaya yönelik hazır araçlar sağlayan birçok yapay zeka platformu vardır. Yapay zeka platformları akıllı karar verme algoritmalarını ve verilerini birleştirirler. Bazı platformların kullanımı kolaydır, bazıları ise derin kodlama uzmanlığı gerektirir.

Yapay zeka yazılımı geliştirmek için yaygın olarak kullanılan platformlar şunlardır:

- **Google platformu:** Yapay zeka Hub'dan (Yapay zeka sistemleri geliştirme kaynakları), AI Building Blocks adlı araçlardan ve fikirlerden lansmana kadar projeler oluşturmak için kod tabanlı bir veri bilimi ortamı olan AI Platform'dan oluşur.
- **Microsoft Azure:** AI yetenekleri arasında uygulamalar ve aracılar, bilgi madenciliği ve makine öğrenimi hizmetleri bulunur. Platform, modeller oluşturmaya, eğitmeye ve dağıtmaya yardımcı olur. Ayrıca içerik, duygu analizi veya anahtar kelime öbekleri çıkarmada kalıp tanımlamanın yerleşik AI yetenekleriyle bulut arama kullanılabilir.
- **Amazon Makine Öğrenimi:** Hizmetler, her türlü karmaşıklıkta ML modelleri oluşturmaya, eğitmeye ve dağıtmaya yardımcı olur. Amazon'un AWS'si, şirketler ve kuruluşlar için kullanıma hazır analitikler sunar ve uygulama geliştirmeyi basitleştirir. Hizmetler, ürünlerle sorunsuz bir şekilde bütünleşir ve rutin prosedürleri basitleştirir.

Programlama Dilleri, Kitaplıklar ve Çerçeveler

AI uygulamaları için yaygın olarak kullanılan programlama dilleri şunlardır:

- **Python:** Veri yapılarıyla kolayca bütünleşir, standart programlamanın ötesinde benzersiz algoritmalar sunar ve geliştiricinin NumPy, Pandas, Scikit, AIMA vb. gibi kitaplıklar ve araçlarla bilgilerini genişletmesine olanak tanır.
- **Java Script:** İstisna işlemeye yönelik düşünceli bir yaklaşım, çok iş parçacıklı uygulamalar geliştirmeye yönelik araçların kullanılabilirliği ve diziler, listeler ve yapılar için destek açısından farklılık gösteren nesne yönelimli bir dil.
- **C++:** Küresel olarak en hızlı derleme dillerinden biri, performans kaybı olmadan son derece karmaşık mantığı uygulamanıza olanak tanır. C++ paketleri, yüksek hızlı animasyona ve işleme motoruyla anında kullanıcı etkileşimine sahip uygulamalar içindir.

Algoritmaları denemek ve ince ayar yapmak ve yapay zeka yazılımı oluşturmak için farklı AI/ML çerçeveleri, uygulama programlama arayüzleri (Application programming interfaces: API'ler) ve altyapılar arasından seçim yapabilirsiniz.
Örneğin,

- Google, Android ve iOS ile uyumlu ve Node.js ve Cordova'yı destekleyen güçlü bir AI aracı olan API.AI'yi oluşturdu. Bir başka kullanışlı Google ürünü, derin makine öğrenimine dayalı uygulamalar geliştirmeye yardımcı olan açık kaynaklı TensorFlow kitaplığıdır.
- CNTK, Caffe, Keras, PyTorch, Accord.NET, scikit-learn ve Spark MLlib gibi çerçeveler;
- Geliştiricilerin IDE'ler ve Jupyter Notebook'lar gibi grafiksel kullanıcı arabirimlerini kullanmalarına olanak tanıyan Hizmet olarak ML platformları;
- REST uç noktaları olarak sunulan ve sonuçlarla birlikte JSON döndüren API'ler (ör. Azure Konu Algılama API'si).

Yapay Zeka ile Yazılım Nasıl Oluşturulur?

AI/ML çözümleri oluşturmak yinelemeli bir süreçtir. Geliştirme hattı, temel biçiminde şu şekilde temsil edilebilir:

- Araştırma, keşif ve ekip planlaması;
- Veri madenciliği;
- Modelleme;
- Minimum uygulanabilir ürün (Minimum viable product: MVP) ve iyileştirmeli ürün geliştirme;
- Başlatma ve destek.

Takım Planlaması

Rekabetçi bir ürün elde etmek için deneyimli bir ekip oluşturulması gerekir:

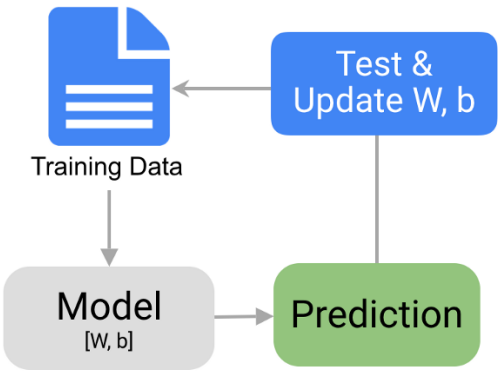
- Yönetim ve araştırma — proje yöneticisi, iş analisti;
- Veri analizi — veri bilimciler, veri kümesi biçimlendirme ekibi, makine öğrenimi mühendisleri;
- Geliştirme — çözüm mimarı, ön uç ve arka uç geliştiricileri, CV, NLP, MLOps, DevOps mühendisleri; ve
- Test etme — kalite güvence (QA) mühendisleri.
- Veri Madenciliği ve Modelleme

Gelişmiş algoritmalar bile uygun şekilde toplanmış ve hazırlanmış verilere ihtiyaç duyar. Çoğu zaman, mühendisler bir dizi adımdan oluşan Cross – Industry Standard Process for Data Mining (CRISP-DM) yöntemini kullanır:

- İş Anlayışı,
- Veri Anlama,
- Veri Hazırlama,
- Modelleme,
- Değerlendirme ve konuşlandırma.

Çoğu durumda, mühendisler adımları döngüsel olarak uygular ve bunları birkaç kez tekrarlar.

Makine Öğrenimi Uygulama Adımları

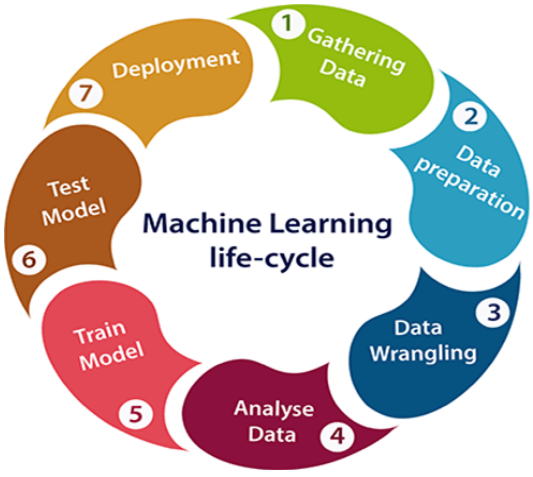


Makine öğrenimi yaşam döngüsü, aşağıda verilen yedi ana adımı içerir:

- Veri Toplama
- Veri hazırlama
- Veri Tartışması
- Verileri Analiz Et
- Modeli eğit
- Modeli test edin
- Uygulama

Tüm süreçteki en önemli şey sorunu anlamak ve sorunun amacını bilmektir. Bu nedenle, yaşam döngüsüne başlamadan önce sorunu anlamamız gerekir çünkü iyi sonuç, sorunun daha iyi anlaşılmasına bağlıdır.

Tüm yaşam döngüsü sürecinde bir problemi çözmek için "model" adı verilen bir makine öğrenme sistemi oluşturuyoruz ve bu model "eğitim" verilerek oluşturuluyor. Ancak bir modeli eğitmek için verilere ihtiyacımız var, dolayısıyla yaşam döngüsü veri toplamakla başlıyor.



Makinelere zeka aktarma görevi 7 ana adıma ayrılabilir:

1. Veri Toplama:

Bildiğiniz gibi, makineler önce onlara verdiğiniz verilerden öğrenirler. Makine öğrenimi modelinizin doğru kalıpları bulabilmesi için güvenilir verilerin toplanması son derece önemlidir. Makineye beslediğiniz verilerin kalitesi, modelinizin ne kadar doğru olduğunu belirleyecektir. Yanlış veya güncel olmayan verileriniz varsa, konuyla ilgili olmayan yanlış sonuçlara veya tahminlere sahip olursunuz.

Modelinizin sonucunu doğrudan etkileyeceğinden, güvenilir bir kaynaktan gelen verileri kullandığınızdan emin olun. İyi veriler alakalıdır, çok az eksik ve tekrarlanan değer içerir ve mevcut çeşitli alt kategorileri/sınıfları iyi bir şekilde temsil eder.

Veri Toplama, makine öğrenimi yaşam döngüsünün ilk adımıdır. Bu adımın amacı, verilerle ilgili tüm sorunları belirlemek ve elde etmektir. Bu adımda, veriler dosyalar, veritabanı, internet veya mobil cihazlar gibi çeşitli kaynaklardan toplanabileceğinden farklı veri kaynaklarını tanımlamamız gerekir. Yaşam döngüsünün en önemli adımlarından biridir. Toplanan verilerin niceliği ve kalitesi, çıktının verimliliğini belirleyecektir. Veriler ne kadar fazla olursa, tahmin o kadar doğru olur.
Bu adım aşağıdaki görevleri içerir:

- Çeşitli veri kaynaklarını tanımlayın
- Veri topla
- Farklı kaynaklardan elde edilen verileri entegre edin

Yukarıdaki görevi gerçekleştirerek, veri kümesi olarak da adlandırılan tutarlı bir veri kümesi elde ederiz. Daha sonraki adımlarda kullanılacaktır.

2. Verilerin Hazırlanması:

Verileri topladıktan sonra, sonraki adımlar için hazırlamamız gerekiyor. Veri hazırlama, verilerimizi uygun bir yere yerleştirip makine öğrenimi eğitimimizde kullanmak üzere hazırladığımız bir adımdır. Bu adımda, önce tüm veriler bir araya getirilir ve ardından verilerin sıralaması rastgele yapılır. Bu adım iki işleme ayrılabilir:

Veri keşfi: Çalışmamız gereken verilerin doğasını anlamak için kullanılır. Verilerin özelliklerini, biçimini ve kalitesini anlamamız gerekir. Verilerin daha iyi anlaşılması, etkili bir sonuca yol açar. Bunun içinde Korelasyonlar, genel eğilimler ve aykırı değerler buluyoruz.

Veri ön işleme: Şimdi bir sonraki adım, analizi için verilerin ön işlenmesidir.

Verilerinizi aldıktan sonra, hazırlamanız gerekir. Bunu şu şekilde yapabilirsiniz:

- Sahip olduğunuz tüm verileri bir araya getirmek ve rastgele hale getirmek. Bu, verilerin eşit olarak dağıtıldığından ve sıralamanın öğrenme sürecini etkilemediğinden emin olmaya yardımcı olur.
- İstenmeyen verileri, eksik değerleri, satırları ve sütunları, yinelenen değerleri, veri türü dönüştürmeyi vb. kaldırmak için verileri temizleme. Hatta veri kümesini yeniden yapılandırmanız ve satırları ve sütunları veya satır ve sütunların dizinini değiştirmeniz bile gerekebilir.
- Nasıl yapılandırıldığını anlamak ve çeşitli değişkenler ve mevcut sınıflar arasındaki ilişkiyi anlamak için verileri görselleştirin.
- Temizlenen verileri iki kümeye bölme - bir eğitim seti ve bir test seti. Eğitim seti, modelinizin öğrendiği settir. Eğitimden sonra modelinizin doğruluğunu kontrol etmek için bir test seti kullanılır.

Veri tartışması, ham verileri temizleme ve kullanılabilir bir biçime dönüştürme işlemidir. Verilerin temizlenmesi, kullanılacak değişkenin seçilmesi ve bir sonraki adımda analize daha uygun hale getirilmesi için verinin uygun formata dönüştürülmesi işlemidir. Tüm sürecin en önemli adımlarından biridir. Kalite sorunlarını çözmek için verilerin temizlenmesi gerekir.
Bazı veriler yararlı olmayabileceğinden, topladığımız verilerin her zaman bizim kullanımımız için olması gerekli değildir. Gerçek dünya uygulamalarında, toplanan verilerin aşağıdakiler dahil çeşitli sorunları olabilir:

- Eksik Değerler
- Yinelenen veriler
- Geçersiz veri
- Gürültü

Bu nedenle, verileri temizlemek için çeşitli filtreleme teknikleri kullanıyoruz. Sonucun kalitesini olumsuz etkileyebileceğinden yukarıdaki hususların tespit edilip ortadan kaldırılması zorunludur.

Veri Analizi: Artık temizlenen ve hazırlanan veriler analiz aşamasına geçilir. Bu adım şunları içerir:

- Analitik tekniklerin seçimi
- Bina modelleri
- Sonucu gözden geçirin

Bu adımın amacı, çeşitli analitik teknikleri kullanarak verileri analiz etmek ve sonucu gözden geçirmek için bir makine öğrenimi modeli oluşturmaktır. Sınıflandırma, Regresyon, Kümeleme analizi, İlişkilendirme vb. gibi makine öğrenmesi tekniklerini seçtiğimiz, ardından hazırlanan verileri kullanarak modeli oluşturduğumuz ve modeli değerlendirdiğimiz problemlerin türünün belirlenmesi ile başlar. Bu nedenle, bu adımda verileri alıyoruz ve modeli oluşturmak için makine öğrenme algoritmalarını kullanıyoruz.

3. Model Seçimi:
Veri kümesinin yapısına ve problemin türüne göre algoritmalar belirlenir. Problemin türüne göre hangi makine öğrenmesi algoritmasının seçileceğine karar veren deneyim ve yetenek kazanma gereksinimine ihtiyaç vardır. Bir makine öğrenimi modeli, toplanan veriler üzerinde bir makine öğrenimi algoritması çalıştırıldıktan sonra elde ettiğiniz çıktıyı belirler. Eldeki görevle ilgili bir model seçmek önemlidir. Yıllar içinde bilim adamları ve mühendisler konuşma tanıma, görüntü tanıma, tahmin vb. gibi farklı görevlere uygun çeşitli modeller geliştirdiler. Bunun dışında modelinizin sayısal mı yoksa kategorik veriler için mi uygun olduğunu görmeli ve buna göre seçim yapmalısınız.

Eğitim Modeli: Şimdi bir sonraki adım modeli eğitmek, bu adımda modelimizi, problemin daha iyi sonuçlanması için performansını geliştirmek üzere eğitiyoruz. Modeli çeşitli makine öğrenimi algoritmaları kullanarak eğitmek için veri kümelerini kullanıyoruz. Çeşitli kalıpları, kuralları ve özellikleri anlayabilmesi için bir modelin eğitimi gereklidir.

Test Modeli: Makine öğrenimi modelimiz belirli bir veri kümesinde eğitildikten sonra modeli test ederiz. Bu adımda, modelimize bir test veri seti sağlayarak modelimizin doğruluğunu kontrol ederiz.
Modelin test edilmesi, proje veya problemin ihtiyacına göre modelin yüzde doğruluğunu belirler.

4. Modelin Eğitimi:
Eğitim, makine öğreniminde en önemli adımdır. Eğitimde, kalıpları bulmak ve tahminler yapmak için hazırlanan verileri makine öğrenme modelinize iletirsiniz. Görev setini gerçekleştirebilmesi için modelin verilerden öğrenmesiyle sonuçlanır. Zamanla, eğitimle model tahmin etmede daha iyi hale gelir.

5. Modelin Değerlendirilmesi:
Modelinizi eğittikten sonra, nasıl performans gösterdiğini kontrol etmeniz gerekir. Bu, modelin performansını daha önce görülmemiş veriler üzerinde test ederek yapılır. Kullanılan görünmeyen veriler, verilerimizi daha önce böldüğünüz test kümesidir. Eğitim için kullanılan aynı veriler üzerinde test yapılırsa, model verilere zaten alıştığından ve daha önce yaptığı gibi aynı kalıpları bulduğundan doğru bir ölçüm elde edemezsiniz. Bu size orantısız bir şekilde yüksek doğruluk sağlayacaktır. Test verileri üzerinde kullanıldığında, modelinizin nasıl performans göstereceğine ve hızına ilişkin doğru bir ölçüm elde edersiniz.

Dağıtım: Makine öğrenimi yaşam döngüsünün son adımı, modeli gerçek dünya sistemine yerleştirdiğimiz dağıtımdır.
Hazırlanan model, kabul edilebilir bir hızda ihtiyacımıza göre doğru bir sonuç üretiyorsa, modeli gerçek sisteme yerleştiririz. Ancak projeyi dağıtmadan önce, mevcut verileri

kullanarak performansını iyileştirip iyileştirmediğini kontrol edeceğiz. Dağıtım aşaması, bir proje için nihai raporun hazırlanmasına benzer.

6. Parametre Ayarı:
Modelinizi oluşturup değerlendirdikten sonra, doğruluğunun herhangi bir şekilde geliştirilip geliştirilemeyeceğine bakın. Bu, modelinizde bulunan parametreleri ayarlayarak yapılır. Parametreler, programcının genel olarak karar verdiği modeldeki değişkenlerdir. Parametrenizin belirli bir değerinde doğruluk maksimum olacaktır. Parametre ayarlama, bu değerleri bulmayı ifade eder. En iyileme yapılacak parametrelerin aralığı belirlenir.

7. Tahmin Yapmak
Sonunda, doğru tahminler yapmak için modelinizi öngörülemeyen, beklenen veriler üzerinde işlem yapılabilir.

2.1. Model Eğitimi

Model eğitimi, veri bilimi ekibinin yapılacak tahmin aralığı üzerindeki kayıp işlevini en aza indirmek için bir makine öğrenmesi algoritmasına en iyi ağırlıkları ve önyargıları uydurmak işlemidir.

Kayıp işlevleri, makine öğrenimi algoritmalarının nasıl optimize edileceğini tanımlar. Bir veri bilimi ekibi, proje hedeflerine, kullanılan veri türüne ve algoritma türüne bağlı olarak farklı türde kayıp işlevleri kullanabilir.

Denetimli öğrenme tekniği kullanıldığında, model eğitimi, veri özellikleri ile hedef etiket arasındaki ilişkinin matematiksel bir temsilini oluşturur. Denetimsiz öğrenmede, veri özellikleri arasında benzerler arasındaki ilişkiği bulmaya yönelik matematiksel bir temsil oluşturur.

Model Eğitiminin Önemi
Model eğitimi, makine öğreniminde birincil adımdır ve daha sonra doğrulanabilen, test edilebilen ve optimize edilebilen bir modelle sonuçlanır. Modelin eğitim sırasındaki performansı, nihai olarak gerçek dünyada uygulamaya konduğunda ne kadar iyi çalışacağını belirleyecektir.

Hem eğitim verilerinin kalitesi hem de algoritma seçimi, model eğitim aşamasının merkezinde yer alır. Çoğu durumda, eğitim verileri eğitim ve ardından doğrulama ve test için iki kümeye bölünür.

Model eğitiminde algoritma-model karmaşıklığı, performans, yorumlanabilirlik, bilgisayar kaynak gereksinimleri ve hız gibi her zaman dikkate alınması gereken ek faktörlerden

etkilenirr. Bu çeşitli gereksinimlerin dengelenmesi, algoritma seçmeyi ilgili ve karmaşık bir süreç haline getirebilir.

Bir Makine Öğrenimi Modeli Nasıl Eğitilir

Bir modeli eğitmek, mevcut eğitim verilerinizin kullanımını ve veri bilimi ekibinizin zamanını en üst düzeye çıkaran sistematik, tekrarlanabilir bir süreç gerektirir. Eğitim aşamasına başlamadan önce problemin belirlenmesi, veri setinize erişilmesi ve modele sunulacak verilerin temizlenmesi gerekir. Buna ek olarak hangi algoritmaları kullanacağınızı ve hangi parametrelerle (hiperparametreler) çalışacağını belirlemeniz gerekir. Tüm bunları yaptıktan sonra, veri kümesi bir eğitim seti ve bir test seti olarak ayrılabilir, ardından da model algoritmaları eğitim için hazırlanabilir.

Veri Kümesinin Bölünmesi

İlk eğitim verileriniz, dikkatli bir şekilde tahsis edilmesi gereken sınırlı bir kaynaktır. Bazıları modelinizi eğitmek için kullanılabilir ve bazıları modelinizi test etmek için kullanılabilir – ancak her adım için aynı verileri kullanamazsınız. Daha önce karşılaşmadığı yeni bir veri seti vermediğiniz sürece bir modeli doğru şekilde test edemezsiniz. Eğitim verilerini iki veya daha fazla kümeye bölmek, tek bir veri kaynağı kullanarak modeli eğitmenize ve ardından doğrulamanıza olanak tanır. Bu, modelin fazla uyumlu olup olmadığını görmenizi sağlar, yani eğitim verileriyle iyi performans gösterirken test verileriyle zayıf performans gösterir.

Eğitim verilerini bölmenin yaygın bir yolu, çapraz doğrulama kullanmaktır. Örneğin, 10 kat çapraz doğrulamada, veriler on kümeye bölünerek verileri on kez eğitmenize ve test etmenize olanak tanır. Bunu yapmak için:

1. Verileri on eşit parçaya veya katlara bölün.
2. Bir katı, uzatılmış kat olarak atayın.
3. Modeli diğer dokuz kat üzerinde eğitin.
4. Modeli uzatılmış kat üzerinde test edin.

Bu işlemi on kez tekrarlayın, her seferinde ayrı kat olarak farklı bir kat seçin. On uzatma katmanındaki ortalama performans, çapraz doğrulanmış puan olarak adlandırılan performans tahmininizdir.

Test Edilecek Algoritmaların Seçilmesi

Makine öğreniminde, aralarından seçim yapabileceğiniz binlerce algoritma vardır ve bir belirli model için hangisinin en iyi olacağını belirlemenin kesin bir yolu yoktur. Çoğu durumda, doğru bir çalışma modeliyle sonuçlananı bulmak için muhtemelen yüzlerce değilse de düzinelerce algoritma deneyeceksiniz. Aday algoritmaların seçilmesi genellikle aşağıdaki koşullara bağlı olacaktır:

- Eğitim verilerinin boyutu.
- Gerekli çıktının doğruluğu ve yorumlanabilirliği.

- Doğrulukla ters orantılı olan gerekli eğitim süresinin hızı.
- Eğitim verilerinin doğrusallığı.
- Veri setindeki özelliklerin sayısı.

Hiperparametrelerin Ayarlanması

Hiperparametreler, model kurulmadan ve eğitilmeden önce veri bilimi ekibi tarafından belirlenen üst düzey niteliklerdir. Eğitim verilerinden birçok öznitelik öğrenilebilirken, kendi hiperparametrelerini öğrenemezler.

Örnek olarak, bir regresyon algoritması kullanıyorsanız, model verileri analiz ederek regresyon katsayılarını kendisi belirleyebilir. Örnek: f(x)=ax2+bx+c ifadesinde a, b, c katsayıları. Bu katsayılardan birindeki hassasiyet durumuna bağl aşırı sapmaya neden olan parametreler. Ancak, değişkenlerin fazlalığını düzenlemek için kullanması gereken cezanın gücünü belirleyemez. Başka bir örnek olarak, rastgele orman tekniğini kullanan bir model, karar ağaçlarının nereye bölüneceğini belirleyebilir, ancak kullanılacak ağaç sayısının önceden ayarlanması gerekir.

Modellerin Sığdırılması ve Ayarlanması

Artık veriler hazırlandığına ve modelin hiperparametreleri belirlendiğine göre, sıra modelleri eğitmeye başlamaya geldi. İşlem, esasen, keşfetmeye karar verdiğiniz her bir hiperparametre değeri kümesini kullanarak farklı algoritmalar arasında geçiş yapmaktır.

Bunu yapmak için:
1. Verileri bölün.
2. Bir algoritma seçin.
3. Hiperparametre değerlerini ayarlayın.
4. Modeli eğitin.
5. Başka bir algoritma seçin ve 3. ve 4. adımları tekrarlayın.

Ardından, aynı algoritma için denemek istediğiniz başka bir hiperparametre değeri kümesi seçin, tekrar çapraz doğrulayın ve yeni puanı hesaplayın. Her bir hiperparametre değerini denedikten sonra, ek algoritmalar için aynı adımları tekrarlayabilirsiniz.

Bu denemeleri atletizm yarışları olarak düşünün. Her algoritma, farklı hiperparametre değerleriyle neler yapabileceğini göstermiştir. Artık her algoritmadan en iyi versiyonu seçip onları final yarışmasına gönderebilirsiniz.

En İyi Modelin Seçilmesi

Şimdi, size genel olarak en iyi modeli vereni belirlemek için her bir algoritmanın en iyi sürümlerini test etme zamanı.
1. Test verileriniz üzerinde tahminler yapın.

2. Bu modelin eğitimi sırasında hedef değişkeniniz için temel gerçeği belirleyin.
3. Tahminlerinizden ve kesinlik hedefi değişkeninden performans ölçütlerini belirleyin.
4. Her finalist modeli test verileriyle çalıştırın.

Test tamamlandıktan sonra, hangi modellerin daha iyi olduğunu belirlemek için performanslarını karşılaştırabilirsiniz. Genel kazanan, testte olduğu kadar eğitimde de (en iyisi değilse) iyi performans göstermiş olmalıdır. Aynı zamanda diğer performans ölçütleriniz üzerinde (hız ve deneysel kayıp gibi) iyi performans göstermeli ve – nihayetinde – problem ifadenizde ortaya atılan soruyu yeterince çözmeli veya yanıtlamalıdır.

Model Eğitimine Sistematik Yaklaşım

Sistematik ve tekrarlanabilir bir model eğitim süreci kullanmak, ölçekte başarılı ML/AI modelleri oluşturmayı planlayan herhangi bir kuruluş için büyük önem taşır. Bunun merkezinde, tüm kaynaklarınızın, araçlarınızın, kitaplıklarınızın ve belgelerinizin, işbirliğini engellemek yerine işbirliğini teşvik edecek tek bir kurumsal platformda bulunması yer alır.

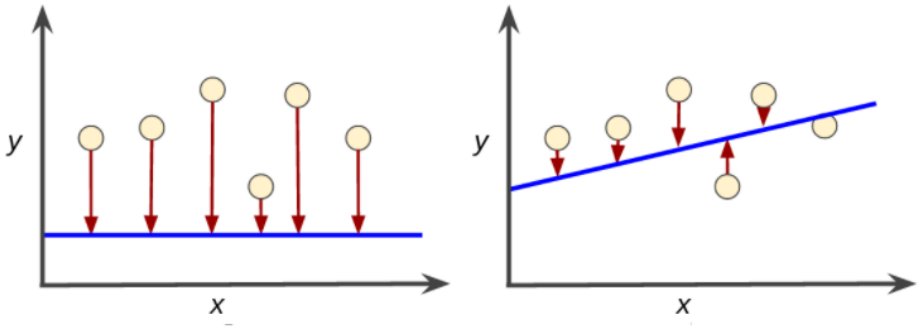
Eğitim Modelleri

Veri kümelerini tanımlamak için kullanılan farklı matematiksel modeller arasından seçme işlemi "Eğitim Model Seçimi" olarak bilinir. Model seçimi istatistik, makine öğrenimi ve veri madenciliği alanlarına uygulanmaktadır.

Makine öğrenmesi modelleri iyi performans gösterebilmeleri için çok fazla veri gerektirir. Bir makine öğrenmesi modeli eğitilirken, temsili veri örneklerinin toplanması gerekir. Eğitim setindeki veriler, bir metin topluluğuna, bir resim koleksiyonuna ve bir hizmetin tek tek kullanıcılarından toplanan verilerine kadar değişebilir. Bir modeli eğitmek, tüm ağırlıklar ve etiketli örneklerden eşik seviye değeri bulabilmek için iyi değerleri öğrenmek (belirlemek) anlamına gelir.

Denetimli öğrenmede, bir makine öğrenimi algoritması birçok örneği inceleyerek ve kaybı en aza indiren bir model bulmaya çalışarak bir model oluşturur; bu sürece ampirik risk minimizasyonu denir.

Kayıp, modelin tahmininin ne kadar kötü olduğunu gösteren bir sayıdır. Modelin tahminin performansı yüksekse, kayıp minumumdur; aksi takdirde kayıp daha büyüktür. Makine öğrenmesinde bir modeli eğitmenin amacı, tüm örneklerde ortalama olarak düşük kayıplı bir eşik değer bulmaktır. Örneğin, aşağıdaki şekilde, solda yüksek kayıplı bir modeli ve sağda düşük kayıplı bir modeli gösterilmektedir. Şekilde okların uzunluğu kaybı temsil eder. Mavi çizgiler tahminleri temsil eder.



Şekil. Sol modelde yüksek kayıp; doğru modelde düşük kayıp.

Soldaki grafikteki okların sağdaki grafikteki benzerlerinden çok daha uzun olduğuna dikkat edin. Açıkçası, sağdaki çizimdeki çizgi, soldaki grafikteki çizgiden çok daha iyi bir tahmin modelidir. Kayıpları anlamlı bir şekilde bir araya getirecek bir matematiksel fonksiyon - bir kayıp fonksiyonu – kullanılır.

Sayısal modeller için karesel kayıp adı verilen bir kayıp fonksiyonu kullanır. -1 ile 1 arasındaki farkların karelerinin toplamının toplam eleman sayısına bölümü ile küçük değer elde edilir.

Tek bir örnek için kayıpların karesi = *etiket ve tahmin arasındaki farkın karesi*
= (gözlem - tahmin (x))^2
= (y - y ') ^{2}

Ortalama kareler hatası (MSE), tüm veri kümesi boyunca örnek başına düşen ortalama kare kayıptır. MSE'yi hesaplamak için, tek tek örnekler için tüm kayıpların karesi toplanır ve ardından örneklerin sayısına bölünür:

$$MSE = \frac{1}{N}\sum_{(x,y)\in D}(y - yt)^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - yt_i)^2$$

burada:
(x,y) bir örnektir
x, modelin tahminlerde bulunmak için kullandığı özellikler kümesidir.
y, örneğin etiketidir.
Tahmin(x), yt: özellikler kümesi ile birlikte ağırlıkların ve yanlılığın bir fonksiyonudur.
D, (x,y) çiftler olan birçok etiketli örneği içeren bir veri kümesidir.
N, D içindeki örneklerin sayısıdır.

MSE, makine öğreniminde yaygın olarak kullanılmasına rağmen, ne tek pratik kayıp işlevi ne de tüm koşullar için en iyi kayıp işlevi değildir.

Örnek:
Makine öğrenmesinde bir modeli eğitmenin amacı, tüm örneklerde ortalama olarak düşük kayıplı bir eşik değer bulmaktır. Bu eşik değer ortalama kareler hatası ile bulunur.

i=1...5
örnek değerler, yi= 1, 2, 3, 4, 5 ;
modelin örnekleme tahmin değerleri yti=1, 2, 7, 2, 3
MSE=((1-1)^2+(2-2)^2+(3-7)^2+ (4-2)^2+(5-3)^2)/5= (0+0+16+4+4)/5=24/5=4.8

örnek değerler, yi= 1, 2, 3, 4, 5 ;
modelin örnekleme tahmin değerleri yti=1, 2, 4, 3, 5
MSE=((1-1)^2+(2-2)^2+(3-4)^2+ (4-3)^2+(5-5)^2)/5= (0+0+1+1+0)/5=2/5=0.4
İkinci örnek ortalama düşük kayıplı olduğu görülmektedir.

Toplu öğrenme:
Belirli bir hesaplama programını çözmek için sınıflandırıcılar veya uzmanlar gibi birden çok model stratejik olarak oluşturulur ve birleştirilir. Bu süreç toplu öğrenme olarak bilinir. Toplu öğrenme, bir modelin sınıflandırmasını, tahminini, işlev yaklaşımını geliştirmek için kullanılır. Topluluk öğrenme, daha doğru ve birbirinden bağımsız bileşen sınıflandırıcılar oluşturduğunuzda kullanılır.

Toplu yöntemler:
Topluluk yöntemlerinin iki paradigması şunlardır:

- Sıralı topluluk yöntemleri
- Paralel topluluk yöntemleri

Bir topluluk yönteminin genel ilkesi, tek bir modele göre sağlamlığı artırmak için belirli bir öğrenme algoritması ile oluşturulmuş birkaç modelin tahminlerini birleştirmektir. Torbalama, istikrarsız tahmin veya sınıflandırma şemalarını iyileştirmek için topluluk içinde bir yöntemdir. Arttırma yöntemi, kombine modelin yanlılığını azaltmak için sırayla kullanılır. Artırma ve Torbalama, varyans terimini azaltarak hataları azaltabilir.

Bir öğrenme algoritmasının beklenen hatası, yanlılık ve varyansa ayrıştırılabilir. Bir yanlılık terimi, öğrenme algoritması tarafından üretilen ortalama sınıflandırıcının hedef işlevle ne kadar yakından eşleştiğini ölçer. Varyans terimi, öğrenme algoritmasının tahmininin farklı eğitim setleri için ne kadar dalgalandığını ölçer.

Toplulukta Artımlı Öğrenme algoritması, bir algoritmanın, sınıflandırıcı halihazırda mevcut olan veri kümesinden oluşturulduktan sonra mevcut olabilecek yeni verilerden öğrenme yeteneğidir.

2.2. Tahmine Dayalı Modelleme Analitiği

Birçok kuruluş için, büyük veriler (inanılmaz hacimlerde ham yapılandırılmış, yarı yapılandırılmış ve yapılandırılmamış veriler) iş kararlarını destekleyebilen ve operasyonları geliştirebilen, kullanılmayan bir zeka kaynağıdır. Veriler çeşitlenmeye ve değişmeye devam ettikçe, giderek daha fazla kuruluş bu kaynaktan yararlanmak ve verilerden uygun ölçekte yararlanmak için tahmine dayalı analitiği benimsiyor.

Tahmine dayalı analitik nedir?

Yaygın bir yanlış anlama, tahmine dayalı analitik ve makine öğreniminin aynı şey olduğudur. Durum bu değil. Özünde, tahmine dayalı analitik, çeşitli istatistiksel teknikleri (makine öğrenimi, tahmine dayalı modelleme ve veri madenciliği dahil) kapsar ve gelecekteki sonuçları tahmin etmek veya "tahmin etmek" için istatistikleri (hem geçmiş hem de mevcut) kullanır. Bu sonuçlar, örneğin bir müşterinin sergilemesi muhtemel davranışlar veya pazardaki olası değişiklikler olabilir. Tahmine dayalı analitik, geçmişi analiz ederek gelecekteki olası olayları anlamamıza yardımcı olur.

Öte yandan makine öğrenimi, Arthur Samuel'in 1959'daki tanımına göre, "bilgisayarlara açıkça programlanmadan öğrenme yeteneği" veren bilgisayar biliminin bir alt alanıdır. Makine öğrenimi, örüntü tanıma çalışmasından gelişti ve algoritmaların verilerden öğrenebileceği ve veriler üzerinde tahminlerde bulunabileceği fikrini araştırıyor. Ve daha "akıllı" hale gelmeye başladıklarında, bu algoritmalar son derece doğru, veriye dayalı kararlar vermek için program talimatlarının üstesinden gelebilir.

Tahmine dayalı analitik nasıl çalışır?

Tahmine dayalı analitik, tahmine dayalı modelleme tarafından yönlendirilir. Bir süreçten çok bir yaklaşımdır. Tahmine dayalı analitik ve makine öğrenimi el ele gider, çünkü tahmine dayalı modeller tipik olarak bir makine öğrenimi algoritması içerir. Bu modeller, yeni verilere veya değerlere yanıt vermek için zamanla eğitilebilir ve işletmenin ihtiyaç duyduğu sonuçları sağlayabilir. Tahmine dayalı modelleme, makine öğrenimi alanıyla büyük ölçüde örtüşmektedir.

İki tür tahmin modeli vardır. Bunlar, sınıf üyeliğini tahmin eden Sınıflandırma modelleri ve bir sayıyı tahmin eden Regresyon modelleridir. Bu modeller daha sonra algoritmalardan oluşur. Algoritmalar, verilerdeki eğilimleri ve kalıpları belirleyerek veri madenciliği ve istatistiksel analiz gerçekleştirir. Tahmine dayalı analitik yazılım çözümleri, tahmine dayalı modeller oluşturmak için kullanılabilecek yerleşik algoritmalara sahip olacaktır. Algoritmalar, verilerin hangi kategorilere ait olduğunu belirleyen "sınıflandırıcılar" olarak tanımlanır.

Makine Öürenmesinde en yaygın olarak kullanılan tahmin modelleri şunlardır:

- **Karar ağaçları:** Karar ağaçları, çok değişkenli analizin basit ama güçlü bir şeklidir. Verileri dal benzeri bölümlere ayırmanın çeşitli yollarını tanımlayan algoritmalar tarafından üretilirler. Karar ağaçları, verileri girdi değişkenlerinin kategorilerine göre alt kümelere bölerek birinin karar yolunu anlamanıza yardımcı olur.

- **Regresyon (doğrusal ve lojistik):** Regresyon, istatistikte en popüler yöntemlerden biridir. Regresyon analizi, değişkenler arasındaki ilişkileri tahmin eder, büyük ve çeşitli veri kümelerinde anahtar kalıpları ve bunların birbirleriyle nasıl ilişkili olduğunu bulur.

- **Nöral ağlar:** İnsan beynindeki nöronların işleyişinden sonra modellenen sinir ağları (yapay sinir ağları olarak da adlandırılır), çeşitli derin öğrenme teknolojileridir. Genellikle karmaşık örüntü tanıma sorunlarını çözmek için kullanılırlar ve büyük veri kümelerini analiz etmek için inanılmaz derecede faydalıdırlar. Verilerdeki doğrusal olmayan ilişkileri yönetmede harikadırlar ve belirli değişkenler bilinmediğinde iyi çalışırlar.

Diğer sınıflandırıcılar:

- **Zaman Serisi Algoritmaları:** Zaman serisi algoritmaları verileri sırayla çizer ve zaman içindeki sürekli değerleri tahmin etmek için kullanışlıdır.
- **Kümeleme Algoritmaları:** Kümeleme algoritmaları, verileri benzer üyeleri olan gruplar halinde düzenler.
- **Aykırı Değer Algılama Algoritmaları:** Aykırı değer algılama algoritmaları, bir veri kümesi içinde beklenen bir modele veya standarda uymayan öğeleri, olayları veya gözlemleri belirleyerek anormallik algılamaya odaklanır.
- **Topluluk Modelleri:** Topluluk modelleri, tek bir algoritmadan elde edilebilecek olandan daha iyi tahmin performansı elde etmek için birden çok makine öğrenimi algoritması kullanır.
- **Faktör Analizi:** Faktör analizi, değişkenliği tanımlamak için kullanılan ve bağımsız gizil değişkenleri bulmayı amaçlayan bir yöntemdir.
- **Naïve Bayes:** Naïve Bayes sınıflandırıcısı, olasılığı kullanarak belirli bir özellik kümesine dayalı bir sınıf/kategori tahmin etmemizi sağlar.
- **Destek vektör makineleri:** Destek vektör makineleri, verileri analiz etmek ve kalıpları tanımak için ilişkili öğrenme algoritmalarını kullanan denetimli makine öğrenimi teknikleridir.

Her sınıflandırıcı verilere farklı bir şekilde yaklaşır, bu nedenle kuruluşların ihtiyaç duydukları sonuçları alabilmeleri için doğru sınıflandırıcıları ve modelleri seçmeleri gerekir.

Tahmine dayalı analitik ve makine öğrenimi uygulamaları

Verilerle dolup taşan ancak bunları yararlı içgörülere dönüştürmekte zorlanan kuruluşlar için tahmine dayalı analitik ve makine öğrenimi çözüm sağlayabilir. Bir kuruluş ne kadar veriye sahip olursa olsun, bu verileri iç ve dış süreçleri geliştirmek ve hedeflere ulaşmak için kullanamıyorsa, veriler işe yaramaz bir kaynak haline gelir.

Tahmine dayalı analitik, en yaygın olarak güvenlik, pazarlama, operasyonlar, risk ve dolandırıcılık tespiti için kullanılır. Tahmine dayalı analitik ve makine öğreniminin farklı sektörlerde nasıl kullanıldığına dair birkaç örnek:

- Bankacılık ve finansal hizmetler sektöründe, dolandırıcılığı tespit etmek ve azaltmak, piyasa riskini ölçmek, fırsatları belirlemek ve çok daha fazlası için tahmine dayalı analitik ve makine öğrenimi birlikte kullanılır.

- Siber güvenliğin günümüzde her işletmenin gündeminin başında yer almasıyla tahmine dayalı analitik ve makine öğreniminin güvenlikte önemli bir rol oynaması şaşırtıcı olmamalı. Güvenlik kurumları, genellikle hizmetleri ve performansı iyileştirmek için tahmine dayalı analitiği kullanır, aynı zamanda anormallikleri, sahtekarlığı tespit etmek, tüketici davranışını anlamak ve veri güvenliğini geliştirmek için de kullanır.

- Perakendeciler, tüketici davranışını daha iyi anlamak için tahmine dayalı analitik ve makine öğrenimi kullanıyor; kim neyi nereden satın alıyor? Bu sorular, doğru tahmine dayalı modeller ve veri setleri ile kolayca yanıtlanabilir, perakendecilerin önceden plan yapmasına ve mevsimsellik ve tüketici eğilimlerine göre ürün stoklamasına yardımcı olarak yatırım getirisini önemli ölçüde artırır.

Doğru ortamı geliştirmek

Makine öğrenimi ve tahmine dayalı analitik, herhangi bir kuruluş için bir nimet olabilirken, bu çözümleri, günlük operasyonlara nasıl uyacağını düşünmeden gelişigüzel uygulamak, kuruluşun ihtiyaç duyduğu içgörüleri sağlama yeteneklerini büyük ölçüde engeller.

Tahmine dayalı analitik ve makine öğreniminden en iyi şekilde yararlanmak için kuruluşların, bu çözümleri destekleyecek mimariye ve bunları besleyecek ve öğrenmelerine yardımcı olacak yüksek kaliteli verilere sahip olduklarından emin olmaları gerekir. Veri hazırlama ve kalite, tahmine dayalı analitik için kilit unsurlardır. Birden çok platforma yayılabilen ve birden çok büyük veri kaynağı içerebilen girdi verileri, merkezileştirilmiş, birleşik ve tutarlı bir biçimde olmalıdır.

Bunu başarmak için kuruluşlar, verilerin genel yönetimini denetlemek ve yalnızca yüksek kaliteli verilerin yakalanmasını ve kaydedilmesini sağlamak için sağlam bir veri yönetişim programı geliştirmelidir. İkinci olarak, kuruluşların işin her noktasında verimliliği artırmalarını sağlayacağından, tahmine dayalı analitik ve makine öğrenimini içerecek şekilde mevcut

süreçlerin değiştirilmesi gerekecektir. Son olarak, kuruluşların hangi sorunları çözmeye çalıştıklarını bilmeleri gerekir, çünkü bu, kullanacakları en iyi ve en uygun modeli belirlemelerine yardımcı olacaktır.

Tahmine dayalı modelleri anlama

Tipik olarak, bir kuruluşun veri bilimcileri ve BT uzmanları, doğru tahmine dayalı modelleri seçme veya kuruluşun ihtiyaçlarını karşılamak için kendi modellerini oluşturma konusunda görevlendirilir. Ancak bugün, tahmine dayalı analitik ve makine öğrenimi artık yalnızca matematikçilerin, istatistikçilerin ve veri bilimcilerin değil, aynı zamanda iş analistlerinin ve danışmanların da alanıdır. Giderek daha fazla şirket çalışanı bunu içgörüler geliştirmek ve iş operasyonlarını iyileştirmek için kullanıyor - ancak çalışanlar hangi modeli kullanacaklarını, nasıl uygulayacaklarını bilmediklerinde veya hemen bilgiye ihtiyaç duyduklarında sorunlar ortaya çıkıyor.

SAS'ta, kuruluşları veri yönetişimi ve analitiğiyle desteklemek için gelişmiş yazılımlar geliştiriyoruz. Veri yönetişimi çözümlerimiz, kuruluşların yüksek kaliteli verileri korumasına ve aynı ortamda işletme genelindeki operasyonları uyumlu hale getirmesine ve veri sorunlarını saptamasına yardımcı olur. . Bu tahmine dayalı analitik çözümleri, her tür kullanıcının ihtiyaçlarını karşılamak üzere tasarlanmıştır ve tahmine dayalı modelleri hızla dağıtmalarını sağlar.

2.3. Öngörülü Modelleme: Türler, Yararlar ve Algoritmalar

Tahmine dayalı modelleme, veri modellemeyi kullanarak gelecekteki sonuçları tahmin etme yöntemidir. Bir işletmenin ileriye dönük yolunu görmesinin ve buna göre planlar yapmasının önde gelen yollarından biridir. Kusursuz olmasa da, bu yöntem yüksek doğruluk oranlarına sahip olma eğilimindedir, bu yüzden bu kadar yaygın olarak kullanılır.

Öngörülü Modelleme Nedir?

Kısacası, tahmine dayalı modelleme, geçmiş ve mevcut verilerin yardımıyla gelecekteki olası sonuçları tahmin etmek ve tahmin etmek için makine öğrenimi ve veri madenciliğini kullanan istatistiksel bir tekniktir. Mevcut ve geçmiş verileri analiz ederek ve öğrendiklerini olası sonuçları tahmin etmek için oluşturulan bir modele yansıtarak çalışır. Tahmine dayalı modelleme, TV reytinglerinden ve bir müşterinin bir sonraki satın alımından kredi risklerine ve kurumsal kazançlara kadar hemen hemen her şeyi tahmin etmek için kullanılabilir.

Tahmine dayalı bir model sabit değildir; temel verilerdeki değişiklikleri dahil etmek için düzenli olarak doğrulanır veya revize edilir. Başka bir deyişle, tek başına yapılan bir tahmin değil. Tahmine dayalı modeller, geçmişte ne olduğuna ve şimdi ne olduğuna bağlı olarak

varsayımlarda bulunur. Gelen, yeni veriler şu anda olanlarda değişiklikler gösteriyorsa, gelecekteki olası sonuç üzerindeki etkisi de yeniden hesaplanmalıdır. Örneğin, bir yazılım şirketi, pazarlama harcamasının etkisine dayalı olarak gelecekteki gelir için bir model oluşturmak için birden fazla bölgedeki pazarlama harcamalarına karşı geçmiş satış verilerini modelleyebilir.

Tahmine dayalı modellerin çoğu hızlı çalışır ve genellikle hesaplamalarını gerçek zamanlı olarak tamamlar. Bu nedenle bankalar ve perakendeciler, örneğin bir çevrimiçi ipotek veya kredi kartı başvurusunun riskini hesaplayabilir ve bu tahmine dayanarak talebi neredeyse anında kabul edebilir veya reddedebilir.

Hesaplamalı biyoloji ve kuantum hesaplamada kullanılanlar gibi bazı kestirimci modeller daha karmaşıktır; ortaya çıkan çıktıların hesaplanması bir kredi kartı uygulamasından daha uzun sürer, ancak bilgi işlem gücü de dahil olmak üzere teknolojik yeteneklerdeki gelişmeler sayesinde geçmişte mümkün olandan çok daha hızlı yapılır.

Tahmine Dayalı Modeller

Neyse ki, tahmine dayalı modellerin her uygulama için sıfırdan oluşturulması gerekmez. Tahmine dayalı analitik araçları, çok çeşitli kullanım durumlarına uygulanabilecek çeşitli incelenmiş modeller ve algoritmalar kullanır.

Tahmine dayalı modelleme teknikleri zamanla mükemmelleştirildi. Daha fazla veri ekledikçe, daha güçlü bilgi işlem, yapay zeka ve makine öğrenimi ekledikçe ve analitikteki genel gelişmeleri gördükçe, bu modellerle daha fazlasını yapabiliyoruz.

İlk beş tahmine dayalı analitik modeli şunlardır:

1) **Sınıflandırma modeli:** En basit model olarak kabul edilir, verileri basit ve doğrudan sorgu yanıtı için sınıflandırır. Örnek bir kullanım durumu, “Bu bir hileli işlem mi?” Sorusuna cevap vermek olabilir.
2) **Kümeleme modeli:** Bu model, verileri ortak niteliklere göre iç içe yerleştirir. Ortak özelliklere veya davranışlara sahip şeyleri veya insanları gruplayarak çalışır ve her grup için daha büyük ölçekte stratejiler planlar. Bir örnek, aynı veya benzer durumdaki diğer kişilerin geçmişte yaptıklarına dayalı olarak bir kredi başvurusunda bulunan kişinin kredi riskinin belirlenmesidir.
3) **Tahmin modeli:** Bu çok popüler bir modeldir ve tarihsel verilerden öğrenmeye dayalı sayısal değeri olan her şey üzerinde çalışır. Örneğin, bir restoranın gelecek hafta ne kadar marul sipariş etmesi gerektiğini veya bir müşteri destek temsilcisinin günde veya haftada kaç aramayı idare edebileceğini yanıtlarken, sistem geçmiş verilere bakar.
4) **Aykırı Değerler modeli:** Bu model, anormal veya aykırı veri noktalarını analiz ederek çalışır. Örneğin, bir banka, bir işlemin müşterinin normal satın alma alışkanlıklarının dışında olup olmadığını veya belirli bir kategorideki harcamanın normal olup olmadığını

sorarak dolandırıcılığı belirlemek için aykırı bir model kullanabilir. Örneğin, kart sahibinin tercih ettiği büyük marketteki bir çamaşır ve kurutma makinesi için kredi kartından 1000$'lık bir ücret endişe verici olmayacaktır, ancak müşterinin başka hiçbir üründen ücret almadığı bir yerde tasarımcı kıyafetlerine harcanan 1.000$, bir hesabın ihlal edildiğinin göstergesi olabilir.

5) **Zaman serisi modeli:** Bu model, zamana dayalı olarak bir dizi veri noktasını değerlendirir. Örneğin, son dört ayda hastaneye kabul edilen felçli hasta sayısı, hastanenin önümüzdeki hafta, gelecek ay veya yılın geri kalanında kaç hasta kabul etmeyi bekleyebileceğini tahmin etmek için kullanılır. Bu nedenle, zaman içinde ölçülen ve karşılaştırılan tek bir metrik, basit bir ortalamadan daha anlamlıdır.

Tahmin Algoritmaları

Tahmine dayalı algoritmalar iki şeyden birini kullanır: makine öğrenimi veya derin öğrenme. Her ikisi de yapay zekanın (AI) alt kümeleridir. Makine öğrenimi (ML), elektronik tablo veya makine verileri gibi yapılandırılmış verileri içerir. Derin öğrenme (DL), video, ses, metin, sosyal medya gönderileri ve görüntüler gibi yapılandırılmamış verilerle ilgilenir; esasen insanların iletişim kurduğu şeyler, sayılar veya metrik okumalar değildir.

Daha yaygın tahmin algoritmalarından bazıları şunlardır:

1) **Rastgele Orman (Random Forest):** Bu algoritma, hiçbiri birbiriyle ilişkili olmayan karar ağaçlarının bir kombinasyonundan türetilmiştir ve büyük miktarda veriyi sınıflandırmak için hem sınıflandırma hem de regresyon kullanabilir.
2) **İki Değer için Genelleştirilmiş Doğrusal Model (GLM: Generalized Linear Model):** Bu algoritma, “en uygun”u bulmak için değişken listesini daraltır. "En uygun" sonucu belirlemek için devrilme noktalarını çözebilir ve veri yakalamayı ve kategorik tahminciler gibi diğer etkileri değiştirebilir, böylece diğer modellerdeki düzenli doğrusal regresyon gibi dezavantajların üstesinden gelebilir.
3) **Gradyan Artırılmış Model (Gradient Boosted Model):** Bu algoritma aynı zamanda birkaç birleşik karar ağacı kullanır, ancak Rastgele Ormanın aksine ağaçlar ilişkilidir. Her seferinde bir ağaç oluşturur, böylece bir sonraki ağacın önceki ağaçtaki kusurları düzeltmesini sağlar. Genellikle arama motoru çıktıları gibi sıralamalarda kullanılır.
4) **K-Means:** Popüler ve hızlı bir algoritma olan K-Means, veri noktalarını benzerliklere göre gruplandırır ve bu nedenle kümeleme modeli için sıklıkla kullanılır. Benzer şekilde astarlı kırmızı yün paltolardan hoşlanan bir milyon veya daha fazla müşteri gibi büyük bir grup içindeki bireylere kişiselleştirilmiş perakende teklifleri gibi şeyleri hızlı bir şekilde sunabilir.
5) **Prophet:** Bu algoritma, envanter ihtiyaçları, satış kotaları ve kaynak tahsisleri gibi kapasite planlaması için zaman serisi veya tahmin modellerinde kullanılır. Oldukça esnektir ve buluşsal yöntemleri ve bir dizi faydalı varsayımı kolayca barındırabilir.

Tahmine Dayalı Modelleme ve Veri Analitiği

Tahmine dayalı modelleme, tahmine dayalı analitik olarak da bilinir. Genel olarak, "tahmini modelleme" terimi akademik ortamlarda tercih edilirken, "tahmini analitik", tahmine dayalı modellemenin ticari uygulamaları için tercih edilen terimdir.

Tahmine dayalı analitiklerin başarılı kullanımı büyük ölçüde yeterli miktarda doğru, temiz ve alakalı verilere sınırsız erişime bağlıdır. Tahmine dayalı modeller, karar ağaçları ve k-ortalama kümeleme kullananlar gibi olağanüstü derecede karmaşık olabilse de, en karmaşık kısım her zaman sinir ağıdır; yani, bilgisayarların sonuçları tahmin etmek için eğitildiği model. Makine öğrenimi, son derece büyük veri kümelerindeki korelasyonları bulmak ve verilerdeki kalıpları "öğrenmek" ve tanımlamak için bir sinir ağı kullanır.

Öngörülü Modellemenin Faydaları

Özetle, tahmine dayalı analitik, iş sonuçlarını tahmin etmede zaman, çaba ve maliyetleri azaltır. Çevresel faktörler, rekabetçi zeka, düzenleme değişiklikleri ve piyasa koşulları gibi değişkenler, nispeten düşük maliyetlerle daha eksiksiz görünümler elde etmek için matematiksel hesaplamaya dahil edilebilir.

İşletmelere fayda sağlayabilecek belirli tahmin türlerinin örnekleri arasında talep tahmini, personel sayısı planlaması, kayıp analizi, dış faktörler, rekabet analizi, filo ve BT (IT) donanım bakımı ve finansal riskler yer alır.

Öngörülü Modellemenin Zorlukları

Tahmine dayalı analitiğin faydalı iş içgörüleri üretmeye odaklanmasını sağlamak önemlidir çünkü bu teknolojinin çıkardığı her şey yararlı değildir. Bazı mayınlı bilgiler, yalnızca meraklı bir zihni tatmin etmede değerlidir ve çok az veya hiç ticari etkisi yoktur. Taraflardan takip edilmek, birkaç işletmenin karşılayabileceği bir dikkat dağıtıcıdır.

Ayrıca, tahmine dayalı modellemede daha fazla veri kullanabilmek sadece bir noktaya kadar bir avantajdır. Çok fazla veri hesaplamayı bozabilir ve anlamsız veya hatalı sonuçlara yol açabilir. Örneğin, dış sıcaklık düştükçe daha fazla kat satılır. Ama sadece bir noktaya kadar. İnsanlar dışarısı -20 derece Fahrenhayt olduğunda, donma noktasının -5 derece altında olduğundan daha fazla palto almazlar. Belli bir noktada, soğuk, palto satın almaya teşvik edecek kadar soğuktur ve daha fazla soğuk sıcaklıklar artık bu kalıbı önemli ölçüde değiştirmez.

Tahmine dayalı modellemede yer alan devasa veri hacimleri ile güvenlik ve mahremiyeti korumak da zor olacaktır. Diğer zorluklar, makine öğreniminin sınırlamalarında yatmaktadır.

Öngörülü Modellemenin Sınırlamaları

Yaygın sınırlamalar ve bunların "en iyi düzeltmeleri" şunları içerir:

1) **Veri etiketlemedeki hatalar:** Bunlar, pekiştirmeli öğrenme veya üretken çekişmeli ağlar (GAN'lar) ile aşılabilir.
2) **Makine öğrenimini eğitmek için gereken çok büyük veri kümelerinin eksikliği:** Olası düzeltme, bir makinenin çok büyük bir veri kümesi yerine az sayıda gösterimden öğrendiği "tek adımda öğrenme"dir.
3) **Makinenin neyi neden yaptığını açıklayamaması:** Makineler insanlar gibi “düşünmez” veya “öğrenmez”. Aynı şekilde, hesaplamaları o kadar olağanüstü karmaşık olabilir ki, insanlar mantığı takip etmek şöyle dursun, bulmakta bile güçlük çekerler. Bütün bunlar, bir makinenin işini açıklamasını veya insanların bunu yapmasını zorlaştırıyor. Yine de, insan güvenliği başta olmak üzere birçok nedenden dolayı model şeffaflığı gereklidir. Umut verici potansiyel düzeltmeler: yerel-yorumlanabilir-modelden bağımsız açıklamalar (LIME: local-interpretable-model-agnostic explanations) ve dikkat teknikleri.
4) **Öğrenmenin genellenebilirliği veya daha doğrusu eksikliği:** İnsanlardan farklı olarak, makineler öğrendiklerini ileriye taşımakta zorluk çekerler. Başka bir deyişle, öğrendiklerini yeni koşullara uygulamakta zorlanırlar. Öğrendiği her şey yalnızca bir kullanım durumu için geçerlidir. Bu, büyük ölçüde, yakında herhangi bir zamanda AI derebeylerinin yükselişi konusunda endişelenmemize gerek yok. Yeniden kullanılabilir olması, yani birden fazla kullanım durumunda yararlı olması için makine öğrenimini kullanan tahmine dayalı modelleme için olası bir düzeltme, aktarım öğrenimidir.
5) **Veri ve algoritmalarda önyargı:** Temsil etmeme, sonuçları çarpıtabilir ve büyük insan gruplarına kötü muamele yapılmasına yol açabilir. Ayrıca, yerleşik önyargıları bulmak ve daha sonra temizlemek zordur. Başka bir deyişle, önyargılar kendi kendini sürdürme eğilimindedir. Bu hareketli bir hedeftir ve henüz net bir düzeltme tespit edilmemiştir.

Öngörülü Modellemenin Geleceği

Tahmine dayalı analitik olarak da bilinen tahmine dayalı modelleme ve makine öğrenimi hala genç ve gelişmekte olan teknolojiler, yani çok daha fazlası var. Teknikler, yöntemler, araçlar ve teknolojiler geliştikçe, işletmelere ve toplumlara faydaları da artacaktır.

Ancak bunlar, işletmelerin daha sonra, teknoloji olgunlaştıktan ve tüm pürüzler çözüldükten sonra benimsemeye gücü yetecek teknolojiler değil. Kısa vadeli avantajlar, geç benimseyen bir kişinin üstesinden gelmesi ve rekabetçi kalması için çok güçlüdür.

Öneri: Teknolojiyi şimdi anlayın ve devreye alın ve ardından teknolojilerdeki müteakip gelişmelerin yanı sıra iş avantajlarını büyütün.

Platformlarda Öngörülü Modelleme

En büyük şirketler dışındaki herkes için, tahmine dayalı analitikten yararlanmak, en kolay şekilde, yerleşik teknolojilere sahip ve önceden eğitilmiş makine öğrenimi içeren ERP sistemleri kullanılarak elde edilir. Örneğin, planlama, tahmin ve bütçeleme özellikleri, değişen piyasa koşullarıyla ilgilenen birden çok senaryoyu hızla modellemek için istatistiksel bir model motoru sağlayabilir.

Başka bir örnek olarak, bir tedarik planlama veya tedarik kapasitesi işlevi, potansiyel olarak geç teslimatları, satın alma veya satış siparişlerini ve diğer riskleri veya etkileri benzer şekilde tahmin edebilir. Şirketlerin üretim veya dağıtım gereksinimlerini karşılamak için dönmelerini sağlamak için alternatif tedarikçiler de gösterge tablosunda temsil edilebilir.

Finansal modelleme, planlama ve bütçeleme, ekibinizi bunaltmadan bu ileri teknolojileri kullanmanın birçok avantajını elde etmek için kilit alanlardır.

2.4. Optimizasyon (Eniyileme)

Optimizasyon: Olası tasarım varyasyonları arasından daha iyi veya daha uygun bir tasarım örneği bulma sürecidir. Parametrelerin duyarlılığına odaklanılması gerekir. İstatistiksel veri analiz yöntemlerini kullanır. Akort – Uygun olana ayarlama işlevine odaklanır.

Matematiksel modellemede, bir gerçel fonksiyonu minimize ya da maksimize etmek amacı ile gerçek ya da tam sayı değerlerini tanımlı bir aralıkta seçip fonksiyona yerleştirerek sistematik olarak bir problemi incelemek ya da çözmek işlemlerini ifade eder. Optimizasyon, algoritmaları bir öğrenme setindeki kaybı en aza indirirken, makine öğrenmesi görünmeyen numunelerdeki kaybı en aza indiren algoritma kullanılan paremetrelerdeki en iyileme yöntemidir.

Hız ve ölçeklenebilirlik nasıl optimize edilir?

- Parametrik taramalar otomatikleştirilir.
- Analitik türevleri kullanılarak gerçek zamanlı parametre ayarları gerçekleştirilir.
- Optimize etmek için performans özellikleri belirlenir.
- Optimize edilmiş model üzerinde hassasiyet ve istatistiksel analiz gerçekleştirilir.
- Çözüm kurulumu
- İşlem sonrası miktarlar

Parametreler:

- Geometriksel boyutlar, şekiller, yön, miktar vb.
- Malzemeler: kayıpsız, karmaşık, iletken, yalıtka, yarıiletken, anizotropik vb.

- Sınırlar: empedans ya da iletkenlik sınırları, bağlantılı sınır tarama açıları, simetri veya mod durumları, vb.

Parametrik analiz yapılırken:

- Model paremetrelerine atanan değişkenlerle nominal tasarım oluşturulur.
- Bir veya daha fazla değişken tarama tanımı yapılır. Her biri, bir aralıktaki değişken değerleri dizisini belirtir.
- Her varyasyon için tasarım çözülür. Her bir varyasyonun performansı nasıl etkilediğini belirlemek için sonuçlar karşılaştırılır.
- Yalnızca bilgi işlem kaynaklarıyla sınırlanan varyasyon sayısı belirlenir.
- Parametrik analizler, makul aralıktaki değişken değerlerin belirlenmesine yardımcı oldukları için genellikle optimizasyonun öncüleri olarak kullanılır.

Optimizasyon analiz parametreleri:

- Maliyet fonksiyonu: Maliyet işlevini en aza indirir. Bunu, minimum konumun da optimum konum olacak şekilde tanımlanır.
- Kabul Edilebilir Maliyet: Optimizasyonun durduğu maliyet fonksiyonunun değeri (negatif olabilir).
- Hedef Ağırlığı: Maliyet birden çok hedefle çalışıyorsa, her bir hedefe farklı ağırlık verebilirsiniz. Maliyet hesaplamasında daha ağır olan hedefe daha çok önem verilmelidir.
- Adım Boyutu: Aramayı mantıklı kılmak için algoritma, bireysel optimizasyon değişkenleri için minimum ve maksimum adım sınırlarını sınırlar.
- Maliyet Fonksiyonu Gürültü: Elemanlar ağındaki değişiklikler nedeniyle maliyet fonksiyonuna çeşitli gürültü kaynakları getirir. Gürültü, değişimin maliyet fonksiyonunun gerçekleştirilmesini desteklemek için yeterince önemli olup olmadığını gösterir.

Kullanılabilir Optimize Ediciler:

- Quasi Newton
- Sequential Non-Linear Programming (SNLP)
- Sequential Mixed Integer Non-Linear Programming
- Pattern Search
- Genetic Algorithm

Duyarlılık analizi:

- Belirli parametrelerdeki küçük değişikliklere karşı çıkış sinyalerin hassasiyetini belirlemek için kullanılır.
- Tasarım hedeflerinin karşılandığından emin olmak için bir tasarımı tolere edebilir. Örneğin, empedansın iyi eşleştiğinden emin olmak için mikroşerit hattının substrat geçirgenliği için maksimum kabul edilebilir sapmayı belirlenir.

- Bu amaçla parametrik analize tercih edilir. Duyarlılığı belirlemek ve bunu sayısal ağ gürültüsünden ve parametre değerlerindeki büyük ölçekli varyasyonlardan ayırmak için gereken parametre varyasyonlarının dikkatli seçimi.
- Optimetrikler, tasarım noktası ile ilgili parametreleri değiştirir ve ikinci dereceden polinomu istenen çıktıya otomatik olarak uyarlar.

İstatistiksel analiz:

- Parametre değerlerinde rastgele değişikliklere maruz kaldığında bileşenin verimini tahmin etmek için kullanılır.
- Optimetrics, tek tip ve Gauss dağılımı işlevlerini içerir.
  Tekdüze dağılım: Kullanıcı, analiz için değişkenin başlangıç değerinden yüzde toleransını belirtir. Bu tolerans aralığı içindeki değerleri tekdüze olasılık varsayarak çözer.
  Gauss dağılımı: Kullanıcı, dağılımın standart sapmasını ve üst ve alt limitleri belirtir. Gauss olasılığını varsayarak üst ve alt sınırlar arasındaki değerleri çözer.

2.5. Simülasyon

Simülasyon, gerçek sistemin yapısı ve davranışını anlayabilmek için mantıksal ve matematiksel modelleme kullanılarak deney yapma olanağı sağlayan bir modeldir. Teknik anlamda gerçek bir sisteminin işletilmesinin zaman üzerinden taklit edilmesidir.

Bilgisayar simülasyonu, gerçek veya teorik bir fiziksel sistemin bir matematiksel modelini tasarlama, gerçek dünyada gibi işlevlerini yerine getirme ve çıktısını analiz etme disiplinidir. Günümüzde uygulamalı eğitim alanlarında çok yoğun olarak kullanılmaya başlanılmıştır.

Simülasyon ve makine öğreniminin neredeyse zıt olduğu belirtilse de simülasyon modeli ile, rastgele değişken girdileri tam olarak bilinmez, ancak model genellikle tam olarak bilinir. Makine öğrenimi ile girdiler tam olarak bilinir, ancak eğitimden önce model bilinmez. Her ikisi de bir çıktı verir, ancak belirsizliğin kaynağı farklıdır. Simülasyonda, belirsizliğin ana kaynağı girdilerdedir. Bir dizi olası çıktı elde etmek ve sonuç olasılıkları hakkında açıklamalar yapmak için tekrar tekrar simülasyon yapmamız gerekiyor. Makine öğreniminde, belirsizliğin ana kaynağı modeldedir. Bir tahmin yaparken, model genellikle tahminden %100 emin değildir. Performans artımı ile doğru modele erişilmeye çalışılır.

Çıktı ile ilgili olarak, farklılıklar daha incedir. Her ikisi de bir çıktı verir, ancak belirsizliğin kaynağı farklıdır. Simülasyonda, belirsizliğin ana kaynağı girdilerdedir. Bir dizi olası çıktı elde etmek ve sonuç olasılıkları hakkında açıklamalar yapmak için tekrar tekrar simülasyon yapmamız gerekiyor. Makine öğreniminde, belirsizliğin ana kaynağı modeldedir. Bir tahmin yaparken, model genellikle tahminden %100 emin değildir.

Simülasyon, “gerçek uygulamaları sanal ortamlarda yaparak öğrenme”' ilkesini bünyesinde barındırır - sistem hakkında bilgi edinmek için önce bir çeşit model oluşturmalı ve ardından modeli çalıştırmalıyız.

Simülasyonun Özellikleri:

- Sistem davranışlarını gözler ve tanımlar.
- Gözlenen davranışlar için geçerli olan teori ve hipotezleri kurar.
- Sistem davranışlarını öngörür.
- Sistemi kontrol etme olanağı sağlar.
- Simülasyon, karmaşık sistemlerin tasarımı ve analizinde kullanılır.

Simülasyon aşağıda verilen amaçlardan birisini veya bir kaçını gerçekleştirmek için kullanılır:

- Değerlendirme: Belirlenen kriterlere göre önerilen sistemin ne kadar iyi çalıştığının gösterilmesi
- Karşılaştırma: Önerilen sistem tasarımlarının veya politikaların karşılaştırılması
- Tahmin : Önerilen koşullar altında sistemin performansının tahmin edilmesi
- Duyarlılık Analizi: Sistemin performansı üzerinde hangi faktörlerin etkili olduğunu belirlenmesi
- Optimizasyon: En iyi performans değerini veren faktör düzeylerinin bir kombinasyonunun belirlenmesidir.
- Darboğaz Analizi: Bir sistemde darboğazların belirlenmesi amacıyla simülasyon kullanılır.

İyi bir simülasyon çalışması için aşağıdaki koşulların sağlanması gerekir:

- Kullanıcılar tarafından kolay anlaşılmalıdır.
- Amaçlar veya hedefler doğru tanımlanmalıdır.
- Güçlü ve güvenilir olmalı, model hatalı sonuçlar vermemelidir.
- Kolayca dönüştürülebilir ve kontrol edilebilir olmalıdır.
- Kolayca değişikliklere uyarlanabilmeli kontrol edilebilmeli ve güncelleştirilebilmelidir.
- Hiç bir model tam olarak gerçek sistemin aynısı olamaz. Dolayısıyla simülasyon modeli kurulurken unutulan veya göz ardı edilen bazı ayrıntılar veya yapılan bazı varsayımlar simülasyon sonuçlarını etkileyebilir.
- Simülasyon çalışmasında insan faktörünün olması simülasyon sonuçlarını etkileyebilir.
- Tüm analiz yöntemlerinde olduğu gibi simülasyon çalışmasında da hatalı verilerin kullanılması, parametrelerin yanlış belirlenmesi ve soyutlamanın yanlış yapılması hatalı sonuçlara neden olur.

Simülasyon Ne Zaman Kullanılır? Bazı problemlerin çözümünde simülasyonun kullanılması zorunluluğu ortaya çıkabilir.

Aşağıdaki durumlarda simülasyon kullanılmasında yarar vardır:

- Problemin tam matematiksel modelinin olmaması.
- Matematiksel modelin analitik yaklaşımla çözülememesi.
- Analitik çözümün mümkün fakat bu çözümün matematiksel olarak çok karmaşık olması.
- Analitik çözümün maliyetleri artırması.
- Sistem henüz tasarım aşamasında ise
- Sistemin davranış analizi yapılacaksa
- Pahalı ve riskler içeren işlerde uzman eleman yetiştirmek amacıyla simülasyon kullanılır

Simülasyonun Avantajları:

- Devam eden operasyonları aksatmadan mevcut sistemleri incelemek için kullanılabilir.
- Önerilen sistemler, kaynaklar kullanılmadan önce "test edilebilir".
- Zamanı kontrol etmemize izin verir.
- Darboğazları belirlememize izin verir.
- Sistem performansı için hangi değişkenlerin en önemli olduğuna dair fikir edinmemizi sağlar.

Simülasyonun dezavantajları:

- Model inşa etmek bir bilim olduğu kadar bir sanattır. Analizin kalitesi, modelin kalitesine ve modelleyicinin becerisine bağlıdır.
- Simülasyon sonuçlarının yorumlanması bazen zordur.
- Simülasyon analizi zaman alıcı ve pahalı olabilir. Analitik bir yöntem daha hızlı sonuçlar sağlayacaksa kullanılmamalıdır.
- Analitik yöntemlere göre daha fazla uzmanlık isteyen bir yöntemdir.
- Simülasyon, incelenen sistemin etkinliği hakkında sadece sayısal bilgiler verebilir. Sebepsonuç ilişkileri hakkında sayısal verilerden görülebilecek ipuçları dışında bilgi vermez.
- Uygun olmayan varsayımlar, modeli gerçek sistemden uzaklaştırır ve yanlış kararlar alınmasına neden olur.

2.6. Kalibrasyon ve Kesinlik

Tüm analitik metotlar kantitatif analiz amacıyla kullanıldıklarında kalibrasyona gereksinim vardır. Kalibrasyon, bir ölçüm yapan sistemin çıkışında ölçülen analitik sinyalin doğru olarak saptanması amacıyla yapılan bir işlemdir: Sinyalin (veya yanıtın) kalibrasyonu yapılmadan bir örnek için alınan verilerle doğru kararlar verilemez. Bir kalibrasyon metodunun özgünlüğü kesinlik, doğruluk, eik değer, hassasiyet, algılama sınırları, seçicilik ve uygulanabilir oadaklanma kapasitesine ya da yeteneğine bağlıdır.

Kesinlik (Precision): Metodun tekrar üretilebilirliğinin (reproduciblility) bir ölçüsüdür; aynı şekilde elde edilen sonuçlar birbirlerine ne kadar yakın değerlerdedir? Rastgele (random) hatalar standart sapmayla izlenebilir; genellikle % relatif sapma değeriyle ifade edilir.

Analitik Metotların Kesinlik (Precision) Tanımları

Terim	Tanım
Mutlak standart sapma, s	$s = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \mathbf{x})^2}{N-1}}$
Relatif standart sapma, RSD	$RSD = \frac{s}{\mathbf{x}}$
Ortalamanın standart hatası, s_m	$s_m = \frac{s}{\sqrt{N}}$
Değişme katsayısı, CV	$CV = \frac{s}{x} 100\ (\%)$
Değişme (variance)	s^2

x_i = i'nci ölçmenin sayısal değeri

$\mathbf{x}$ (veya, $x_{ort.}$) = N ölçmenin ortalaması $\quad \mathbf{x} = \frac{\sum_{i=1}^{N} x_i}{N}$

3. Makine Öğrenmesi Türleri

Makine öğrenmesi türleri, veri yığını özeelliklerine ve problemin çözümüne göre farklılık gösterirler.

- Denetimli Öğrenme (Supervised Learning)
- Denetimsiz Öğrenme (Unsupervised Learning)
- Yarı denetimli Öğrenme
- Takviyeli Öğrenme

Genellikle, makine öğreniminde veri yığınından dört adım işlev gerçekleştirilir:

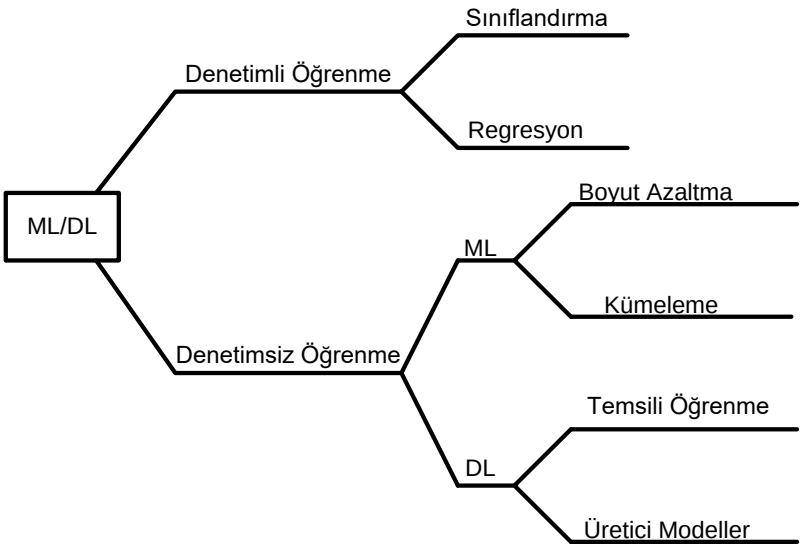
1) Veri tabanı yönetimi
2) Veri hazırlama: eksik, hatalı, normlara ve standartlara uymayan
3) Eğitim modeli: Verilerin bir koleksiyonundan öğrenilir. Eğitim veri yığınından sınıflandırma, matematiksel ifadenin katsayaları (regresyon), kümeleme
4) Uygulama modeli: Yeni test verileri hakkında kararlar almak için kullanılır.

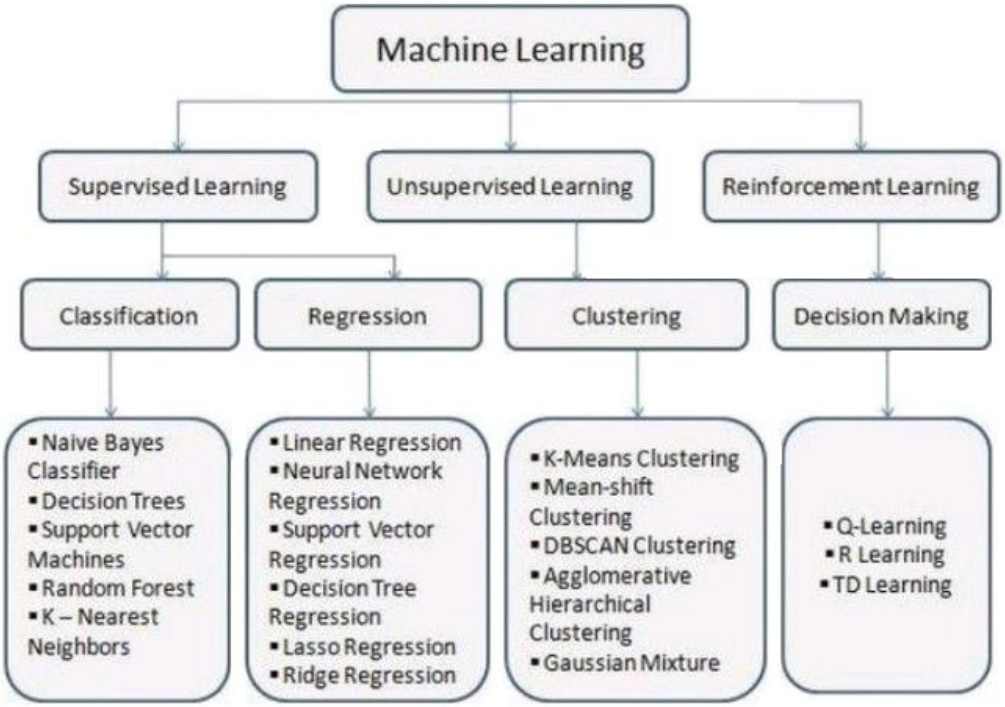
Makine öğrenimi türlerinden bazıları şunlardır:

- Eğitim verilerinin doğru yanıtlarla etiketlendiği denetimli öğrenim. En yaygın iki denetimli öğrenme türü, sınıflandırma ve regresyondur. Sınıflandırma, bir sandık dolusu meyvelerin her birinin özelliği etiketlendikten sonra giriş yapan bir meyvenin hangi sınıfa dahil olduğunun belirlenmesidir.
- Analiz etmek ve keşfetmek istediğimiz kalıpları, etiketlenmemiş verilerden oluşan bir koleksiyondan öğrenen denetimsiz öğrenme. En önemli iki örnek, boyut küçültme ve kümelenmedir. Bir sandık meyvenin benzerlik özelliklerine göre sınıflandırılmanın etiketleme olmadan yapılması.
- Robot veya kontrolör gibi bir temsilcinin geçmişteki eylemlerin sonuçlarına dayalı olarak uygun eylemleri öğrenmeye çalıştığı pekiştirmeli öğrenme.
- Eğitim verilerinin yalnızca bir alt kümesinin etiketlendiği yarı denetimli öğrenme.
- Mali piyasalarda olduğu gibi zaman serisi tahmini
- Fabrikalarda ve gözetimde arıza tespiti için kullanılanlar gibi anormallik tespiti.
- Verilerin elde edilmesinin pahalı olduğu aktif öğrenme

Bu nedenle bir algoritmanın hangi eğitim verilerinden elde edileceğinin belirlenmesi tecrübeye dayanmaktadır.

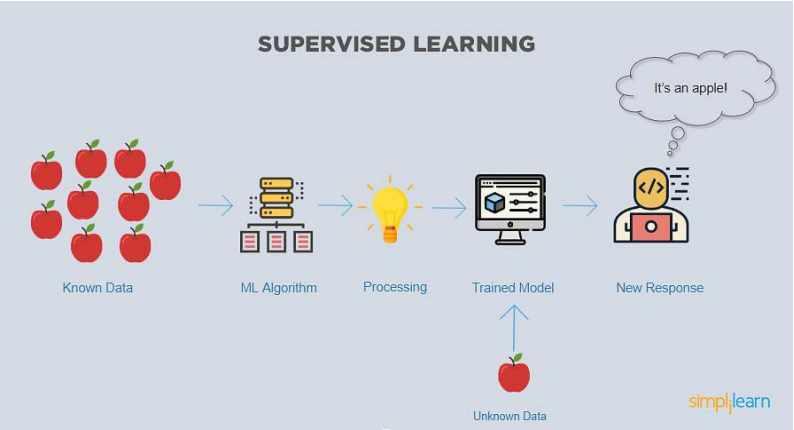
Makine öğrenimi modeli nedir?





1. Denetimli Öğrenme

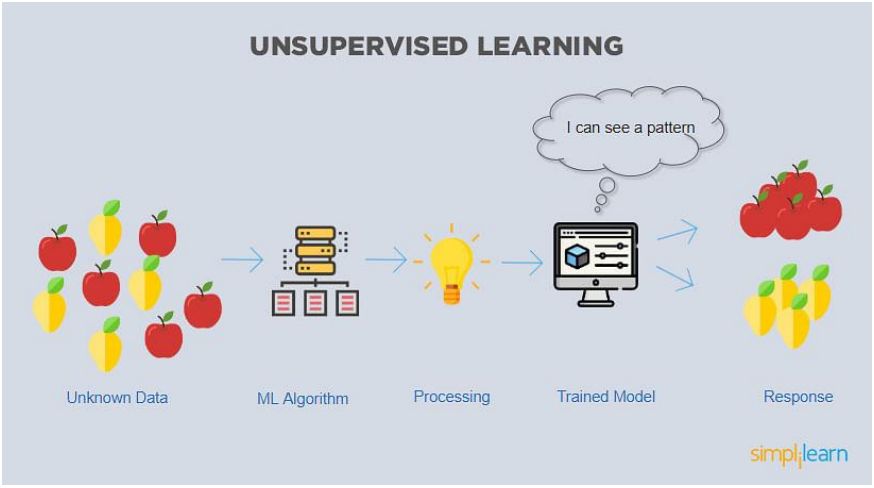
Denetimli öğrenmede, eğitim verileri için bilinen veya etiketlenmiş veriler kullanılır. Veriler bilindiği için, öğrenme denetlenir, yani başarılı uygulamaya yönlendirilir.



Model iyi eğitildikten sonra, verinin bir elma olduğunu belirleyecek ve istenen yanıtı verecektir.

2. Denetimsiz Öğrenme

Denetimsiz öğrenmede, eğitim verileri bilinmez ve etiketlenmez - bu, daha önce hiç kimsenin verilere bakmadığı anlamına gelir. Denetimsiz öğrenmeye ait eğitilmiş modelde, benzerleri bir araya getirmeye çalışır. Filtreleme yapılır. Bilinen verilerin yönü olmadan girdi, denetimsiz terimin kaynaklandığı algoritmaya yönlendirilemez.



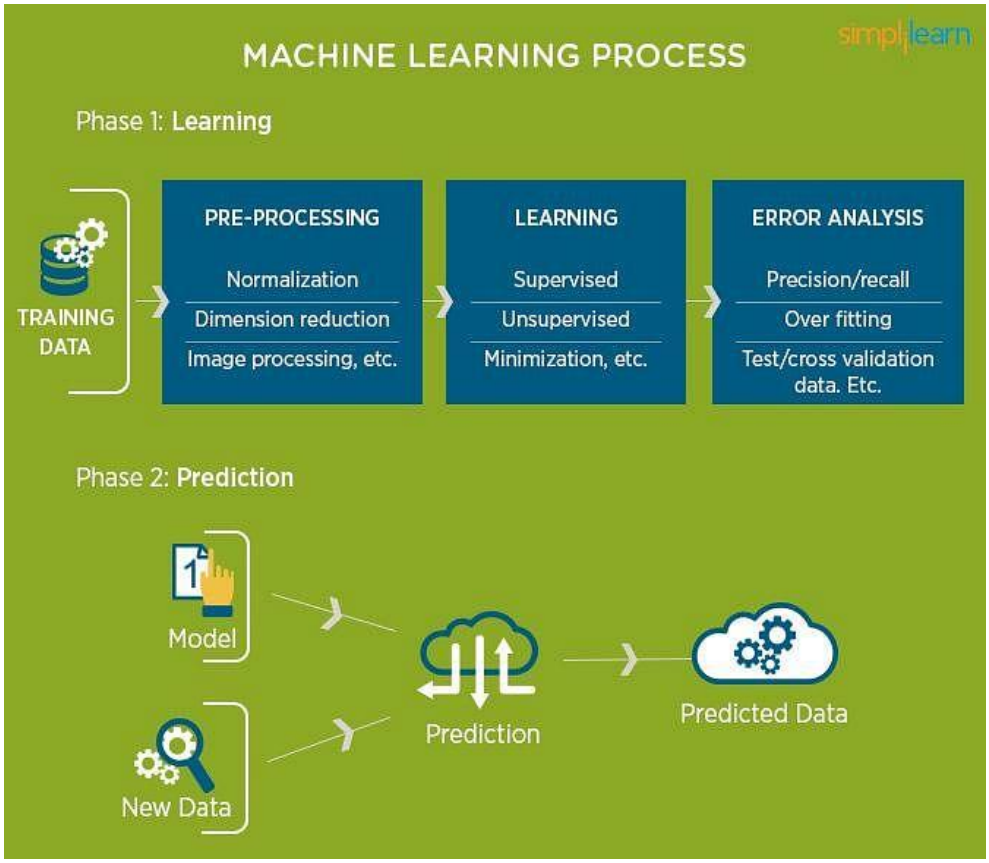
Yukarıdaki örnekte bilinmeyen veriler birbirine benzeyen elma ve armutlardan oluşmaktadır.

3. Pekiştirmeli Öğrenme

Geleneksel veri analizi türleri gibi, algoritma bir deneme yanılma süreci yoluyla verileri keşfeder ve ardından hangi eylemin daha yüksek ödüllerle sonuçlanacağına karar verir. Kümeleme yapılırken başarı oranına bakılır. Üç ana bileşen pekiştirmeli öğrenmeyi oluşturur: etmen, çevre ve eylemler. Etmen, öğrenen veya karar vericidir, çevre, aracının etkileşimde bulunduğu her şeyi içerir ve eylemler, aracının yaptığı şeydir.

Destekleyici öğrenme, aracı belirli bir süre içinde beklenen ödülü en üst düzeye çıkaran eylemleri seçtiğinde gerçekleşir. Temsilci sağlam bir politika çerçevesi içinde çalışırken bunu başarmak en kolay yoldur.

Aşağıda gösterilen süreç akışı, Makine Öğreniminin nasıl çalıştığını gösterir:



Makine Öğrenimi, matematiksel modellerdeki katsayıları veya algoritmaları otomatikleştirerek ya da otonomlaştırılarak veri çıkarma ve yorumlama şeklini almıştır ve böylece geleneksel istatistiksel ve olasık teknikleri yerini almıştır.

Hangi makine öğrenimi algoritmasının kullanılacağına nasıl karar verilir?
Aralarından seçim yapabilecek düzinelerce farklı algoritma var, ancak en iyi seçenek veya her duruma uyan bir seçenek yok. Çoğu durumda, deneme yanılmaya başvurmanız gerekir.

Ancak, seçimlerinizi daraltmanıza yardımcı olabilecek sorabileceğiniz bazı sorular var.

- Üzerinde çalışacağınız verilerin boyutu nedir?
- Üzerinde çalışacağınız veri türü nedir?
- Verilerden ne tür içgörüler arıyorsunuz?
- Bu içgörüler nasıl kullanılacak, karar vermeme nasıl yardım edecek?

Makine Öğrenimi için En İyi Programlama Dili Nedir?
Matlab, Python, Java Script, C++ mevcut birçok kütüphanenin yanı sıra veri analizi ve veri madenciliği için idealdir ve birçok algoritmayı (sınıflandırma, kümeleme, regresyon ve boyut azaltma için) ve makine öğrenimi modellerini destekler.

Bazı Makine Öğrenimi Algoritmalarına ve Süreçlerine Bir Bakış
Makine Öğreniminin ne olduğunu öğreniyorsanız, standart Makine Öğrenimi algoritmalarına ve süreçlerine aşina olmalısınız. Bunlara sinir ağları, karar ağaçları, rastgele ormanlar, ilişkiler ve dizi keşfi, gradyan artırma ve torbalama, destek vektör makineleri, kendi kendini organize eden haritalar, k-ortalama kümeleme, Bayes ağları, Gauss karışım modelleri ve daha fazlası dahildir.

Büyük verilerden en fazla değeri elde etmek için çeşitli algoritmaları kıyaslayan makine öğrenimi araçları ve süreçleri vardır:

- Kapsamlı veri kalitesi ve yönetimi
- Algoritmalarda modeller ve süreç akışları oluşturmak
- Etkileşimli veri keşfi ve model sonuçlarının görselleştirilmesi
- Uygulamaya yönelik en iyisini hızlı bir şekilde belirlemek için farklı Makine Öğrenimi modellerinin karşılaştırmaları
- En iyi performans gösterenleri belirlemek için otomatikleştirilmiş topluluk modeli değerlendirmesi
- Hızlı bir şekilde tekrarlanabilir, güvenilir sonuçlar alabilmeniz için kolay model dağıtımı
- Veriden karara kadar sürecin otomasyonu için entegre bir uçtan uca platform

Makine öğrenimi (ML) için ön koşullar

Makine Öğrenimi alanında başarılı olmak için birkaç gereksinimin karşılanması gerekir:

1. Matlab, Python, R, Java, JavaScript vb. programlama dilleri hakkında temel bilgi
2. Orta düzeyde istatistik ve olasılık bilgisi
3. Temel lineer cebir bilgisi. Polinom ve denklemlerde katsayıların belirlenmesi. Doğrusal regresyon modelinde, tüm veri noktalarından bir doğru çizilir ve bu doğru yeni değerleri hesaplamak için kullanılır.
4. Hesaplamalarda yorumların anlaşılması (Türev, Integral, Limit)
5. Karar verme için harcanan süreyi azaltmak için ham verilerin nasıl temizleneceği ve istenen formatta yapılandırılacağı bilgisi, veri hazırlama.

3.1. Denetimli Öğrenme

Denetimli öğrenme, etiketli eğitim verilerinden fonksiyonlar oluşturulması ile ilgili makine öğreniminin bir dalıdır. Belki de şimdilik makine ya da derin öğrenmenin ana akımıdır. Denetimli öğrenmede, eğitim verileri bir dizi giriş ve hedef çiftinden oluşur; burada giriş, özelliklerin bir vektörü (Öznitelik Vektörü) olabilir ve hedef, işlevin çıktı vermesi için ne istediğimizi belirtir. Hedef, sınıfın veya değer etiketinin tahmin edilmesidir.

Hedefin türüne bağlı olarak, denetimli öğrenimi kabaca iki kategoriye ayrılır: Sınıflandırma ve regresyon. (Kategori: Aralarında herhangi yönden benzerlik, bağ ya da ilgi bulunması)

- Sınıflandırma, aralarında herhangi yönden benzerlik, bağ ya da ilgi bulunan hedefleri içerir; Görüntü sınıflandırması gibi bazı basit durumlardan makine çevirileri ve resim yazısı gibi bazı gelişmiş konulara kadar değişen örnekler.
- Regresyon, nicel (sayısal) değişkenler arasındaki ilişkilerin belirlendiği hedefleri içerir. Uygulamaların tümü bu kategoriye girer. Örneğin, stok tahmini, görüntü maskeleme ve diğerlerini içerir.

Denetimli öğrenmeye yönelik standart yaklaşım, veri yığınını eğitim kümesi ve test kümesi olarak bölmektir. Eğitim seti, test setinden farklıdır.

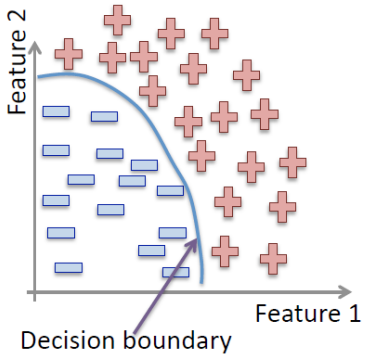
Soru: Meyvelerden oluşan torbayı taşıyan robotun taşıdığı torba parçalanır ve tüm meyveler birbirine karışır. Robot, topladığı meyveleri önceden etiketlediğinden hemen ayrıştırır. Bu örnekteki robotta hangi öğrenme algoritması bulunmaktadır?

Soru: Bir çocuğa içinde on aslan, on maymun, on fil ve diğerleri gibi her türden on hayvanın bulunduğu 100 oyuncak hayvan veriyoruz. Daha sonra, çocuğa bir hayvanın farklı

özelliklerine (özelliklerine) dayalı olarak farklı hayvan türlerini tanımayı öğretiyoruz. Rengi turuncuysa, o zaman bir aslan olabilir. Gövdesi büyük bir hayvansa, fil olabilir gibi.

Çocuğa hayvanları nasıl ayırt edeceğini öğretiriz, bu nasıl öğrenmeye bir örnektir? Şimdi çocuğa farklı hayvanlar verdiğimizde, onları uygun bir hayvan grubuna ayırabilmelidir.

Aynısı bilgisayarlar için de geçerlidir. Onlara gerçek etiketli değerleri ile binlerce veri noktası sağlıyoruz (Etiketli veriler, özellik değerleriyle birlikte farklı gruplara sınıflandırılır). Daha sonra eğitim döneminde farklı özelliklerinden ders çıkarır. Eğitim dönemi bittikten sonra eğitimli modelimizi tahmin yapmak için kullanabiliriz. Makineyi zaten etiketli verilerle beslediğimizi, bu nedenle tahmin algoritmasının denetimli öğrenmeye dayandığını unutmayın. Kısaca bu örnekteki tahminlerin etiketli verilere dayandığını söyleyebiliriz.



Denetimli öğrenme algoritmaları:

- K-En Yakın Komşular
- Doğrusal Regresyon
- Lojistik regresyon
- Rastgele Orman
- Gradyan Güçlendirilmiş Ağaçlar
- Destek Vektör Makineleri (SVM)
- Naive Bayes
- Nöral ağlar
- Karar ağaçları

Matematiksel modelde, her öğrenme örneği bazen özellik vektörü olarak adlandırılan bir dizi veya vektörle temsil edilir veya öğrenme verileri bir matrisle temsil edilir. Nesnel bir fonksiyonun yinelemeli optimizasyonu yoluyla, denetimli öğrenme algoritmaları yeni girdilerle ilişkili çıktıyı tahmin etmek için kullanılabilecek bir işlevi öğrenir. Zaman içinde çıktılarının veya tahminlerinin doğruluğunu arttıran bir algoritmanın bu görevi yerine getirmeyi öğrendiği söylenir.

Sıralı Denetimli Öğrenme problemlerini çözmek için farklı yöntemler şunlardır:

- Sürgülü pencere yöntemleri
- Tekrarlayan sürgülü pencereler
- Gizli Markow modelleri
- Maksimum entropi Markow modelleri
- Koşullu rastgele alanlar
- Grafik trafo ağları

3.2. Denetimsiz Öğrenme

Öğrenme algoritmasına hiçbir etiket verilmez ve girdisinde yapı bulmak için tek başına bırakılır. Denetimsiz öğrenmeye ait eğitilmiş modelde, benzer gruplarda aynı şeyleri elde edebilmeniz için benzerlerin ve farklı olanların özelliklerini belirlemeye çalışır. Filtreleme yapılır. Denetimsiz öğrenme algoritmaları, yalnızca girdileri içeren bir veri yığınını alır ve verileri yapısal olarak gruplar veya kümeler. Bu nedenle algoritmalar, etiketlenmemiş, sınıflandırılmamış veya kategorize edilmemiş test verilerinden öğrenilir.

Geri bildirime yanıt vermek yerine, denetimsiz öğrenme algoritmaları verilerdeki ortaklıkları ve farklılıkları tanımlar ve her yeni veri parçasında bu tür ortak özelliklerin varlığına veya yokluğuna bağlı olarak tepki verir. Denetimsiz öğrenmenin merkezi bir uygulaması, olasılık yoğunluk işlevini bulmak gibi istatistiklerde yoğunluk tahmini alanındadır.

Denetimsiz Öğrenmenin işlevleri:

- Veri kümeleri bulunur,
- Verilerin düşük boyutlu temsilleri bulunur,
- Verilerde ilginç benzer ve farklı yönler bulunur,
- İlginç koordinatlar ve korelasyonlar elde edilir,
- Yeni gözlemler ya da veritabanı elde edilir.

Soru: Elmalardan oluşan torbayı taşıyan robotun taşıdığı torba parçalanır ve tüm elmalar (Çürük, büyük, küçük, ort boy, ...) birbirine karışır. Robot, topladığı elmaları önceden etiketlemediğinden hemen ayrıştıramaz. Tek başına farklılıkları öğrenerek ayrıştırma işleme yapar. Bu örnekteki robotta hangi öğrenme algoritması bulunmaktadır?

Denetimli öğrenmenin aksine, denetimsiz öğrenme, benzerlikleri ve benzerler arasındaki ilişkileri belirlemek için verilerdeki gizli yapıları tanımlayan bir işlev olan etiketlenmemiş verilerden kaynaklanır.

Belki de denetimsiz öğrenmenin en temel türü, boyut azaltma yöntemleridir; PCA genellikle veri ön işlemede kullanılır ve t-SNE genellikle veri görselleştirmede kullanılır.

Daha gelişmiş bir dal, verilerdeki gizli kalıpları araştıran ve daha sonra bunlar hakkında tahminlerde bulunan kümelemedir; örnekler arasında K-ortalama kümeleme, Gauss karışım modelleri, gizli Markov modelleri ve diğerleri bulunur.

Denetimsiz öğrenme algoritmalarına örnekler:

- Boyut Küçültme
- Yoğunluk Tahmini
- Pazar Sepeti Analizi
- Üretken düşmanlık ağları (GAN'lar)
- Kümeleme

3.3. Yarı denetimli öğrenme

Yarı denetimli öğrenme, denetimsiz öğrenme (herhangi bir etiketlenmiş öğrenme verisi olmadan) ve denetimli öğrenme (tamamen etiketlenmiş öğrenme verisi ile) arasındadır. Bazı öğrenme örnekleri öğrenme etiketlerinin eksik olmasına rağmen, birçok makine öğrenmesi araştırmacısı, etiketlenmemiş verilerin, az miktarda etiketlenmiş verilerle birlikte kullanıldığında, öğrenme doğruluğunda önemli bir gelişme sağlayabildiğini bulmuştur.

Zayıf denetimli öğrenmede, öğrenme etiketleri gürültülü, sınırlı veya kesin değildir; bununla birlikte, bu etiketlerin elde edilmesi genellikle daha ucuzdur, bu da daha büyük etkili öğrenme setleriyle sonuçlanır.

3.4. Takviyeli Öğrenme

Bir bilgisayar programı, belirli bir hedefi (araç kullanmak veya rakibe karşı oyun oynamak gibi) gerçekleştirmesi gereken **dinamik bir ortamla etkileşime girer**. Sorun alanı içinde ilerledikçe, program ödüllendirmeye benzeyen ve performansı en üst düzeye çıkarmaya çalışan geri bildirim sağlar.

Takviye öğrenmesi, yazılım temsilcilerinin kümülatif ödül kavramını en üst düzeye çıkarmak için bir ortamda nasıl işlem yapmaları gerektiğiyle ilgili bir makine öğrenmesi alanıdır. Genelliği nedeniyle, oyun teorisi, kontrol teorisi, yöneylem araştırması, bilgi teorisi, simülasyon tabanlı optimizasyon, çok etmenli sistemler, sürü zekası, istatistikler ve genetik algoritmalar gibi birçok disiplinde çalışılmaktadır. Makine öğrenmesinde, ortam tipik olarak bir Markov Karar Süreci (MDP) olarak temsil edilir. Pek çok takviye öğrenme algoritması dinamik programlama teknikleri kullanır. Takviye öğrenme algoritmaları, MDP'nin kesin bir

matematiksel modeli hakkında bilgi sahibi değildir ve kesin modeller mümkün olmadığında kullanılır. Takviye öğrenme algoritmaları otonom araçlarda veya bir insan rakibe karşı bir oyun oynamayı öğrenmek için kullanılır.

- Q-Learning
- Temporal Difference (TD)
- Monte-Carlo Tree Search (MCTS)
- Asynchronous Actor-Critic Agents (A3C)

3.5. Pekiştirmeli öğrenme

Pekiştirmeli öğrenme, davranışçılıktan esinlenen, öznelerin bir ortamda en yüksek ödül miktarına ulaşabilmesi için hangi eylemleri yapması gerektiğiyle ilgilenen bir makine öğrenmesi yaklaşımıdır. Bu problem, genelliğinden ötürü oyun kuramı, kontrol kuramı, yöneylem araştırması, bilgi kuramı, benzetim tabanlı eniyileme ve istatistik gibi birçok diğer dalda da çalışılmaktadır.

Makine öğrenmesinde, ortam genellikle bir Markov karar süreci (MKS) olarak modellenir, bu bağlamda birçok pekiştirmeli öğrenme algoritması dinamik programlama tekniklerini kullanır. Pekiştirmeli öğrenme algoritmalarının klasik tekniklerden farkı, MKS hakkında ön bilgiye ihtiyaç duymamaları ve kesin yöntemlerin verimsiz kaldığı büyük MKS'ler için kullanılmalarıdır.

Pekiştirmeli öğrenme, doğru girdi/çıktı eşleşmelerinin verilmemesi ve optimal olmayan eylemlerin dışarıdan düzeltilmemesi yönleriyle gözetimli öğrenmeden ayrışır. Dahası, pekiştirmeli öğrenmede bilinmeyen uzayda keşif (İngilizce: exploration) ile mevcut bilgiden istifade (İngilizce: exploitation) arasında bir denge kurma söz konusudur.

3.6. Özellik Öğrenme

Öğrenme algoritmaları, genellikle eğitim sırasında sağlanan girdilerin daha iyi temsilini keşfetmeyi amaçlamaktadır. Klasik örnekler temel bileşenler analizi ve küme analizini içerir. Temsili öğrenme algoritmaları olarak da adlandırılan özellik öğrenme algoritmaları, genellikle girdilerindeki bilgileri korumaya çalışır, ancak sınıflandırma veya tahminler gerçekleştirmeden önce genellikle bir ön işleme adımı olarak yararlı hale getirecek şekilde dönüştürür. Bu teknik, bilinmeyen veri üreten dağıtımdan gelen girdilerin yeniden yapılandırılmasına izin verirken, bu dağıtım altında mantıksız olan yapılandırmalara mutlaka sadık kalmaz. Bu, manuel özellik mühendisliğinin yerini alır ve bir makinenin hem özellikleri öğrenmesini hem de belirli bir görevi gerçekleştirmek için kullanmasını sağlar.

Denetimli özellik öğrenmede, özellikler etiketli giriş verileri kullanılarak öğrenilir. Örnekler arasında yapay sinir ağları, çok katmanlı algılayıcılar ve denetimli sözlük öğrenmesi sayılabilir. Denetimsiz özellik öğrenmede, özellikler etiketlenmemiş girdi verileriyle benzerlikler ve benzerler arası ilişkiler öğrenilir. Örnekler arasında sözlük öğrenmesi, bağımsız bileşen analizi, otomatik kodlayıcılar, matris çarpanlarına ayırma ve çeşitli kümeleme biçimleri bulunmaktadır.

Manifold öğrenme algoritmaları, öğrenilen sunumun düşük boyutlu olması kısıtlaması altında bunu yapmaya çalışır. Seyrek kodlama algoritmaları, öğrenilen sunumun seyrek olduğu, yani matematiksel modelin çok sayıda sıfır olduğu kısıtlaması altında bunu yapmaya çalışır. Çok satırlı altuzay öğrenme algoritmaları, düşük boyutlu gösterimleri, çok boyutlu veriler için tensör gösterimlerinden, daha yüksek boyutlu vektörlere dönüştürmeden doğrudan öğrenmeyi amaçlamaktadır. Derin öğrenme algoritmaları birden çok temsil düzeyini ya da bir özellik hiyerarşisini keşfeder; daha düşük düzey özellikler (ya da üretme) açısından daha yüksek düzey, daha soyut özellikler tanımlanır. Akıllı bir makinenin, gözlemlenen verileri açıklayan temel varyasyon faktörlerini çözen bir temsili öğrenen bir makine olduğu ileri sürülmüştür.

Özellik öğrenme, makine öğrenmesinin sınıflandırma algoritmalarında, görevlerin genellikle matematiksel ve hesaplamaya uygun olarak işlenmesi için girdi gereksinimi gerçeğiyle motive edilir ve performansı yükseltir. Bununla birlikte, görüntüler, video ve duyusal veriler gibi gerçek dünya verileri, belirli özellikleri algoritmik olarak tanımlama girişimlerine yol açmamıştır. Bir alternatif, açık algoritmalara dayanmadan, bu özellikleri veya gösterimleri muayene yoluyla keşfetmektir.

3.7. Diğer Öğrenme Yöntemleri

Kendi kendine öğrenme:

Makine öğrenmesi paradigması olarak kendi kendine öğrenme 1982'de Crossbar Adaptive Array (CAA) adı verilen kendi kendine öğrenebilen bir sinir ağı ile tanıtıldı. Dış ödüller ve dış öğretmen tavsiyeleri olmayan bir öğrenmedir. CAA kendi kendine öğrenme algoritması, çapraz çubuk şeklinde, sonuç durumlarıyla ilgili eylemler ve duygular (duygular) hakkındaki kararları hesaplar. Sistem, biliş ve duygu arasındaki etkileşim tarafından yönlendirilir. Kendi kendine öğrenme algoritması W = || w (a, s) || bellek matrisini günceller böylece her yinelemede aşağıdaki makine öğrenme rutini yürütülür.

Seyrek sözlük öğrenme:

Seyrek sözlük öğrenmesi, bir eğitim örneğinin temel işlevlerin doğrusal bir kombinasyonu olarak temsil edildiği ve seyrek bir matris olduğu varsayılan bir özellik öğrenme yöntemidir.

Yöntem güçlü bir şekilde NP-zordur (Nondeterministic polynomial) ve yaklaşık olarak çözülmesi zordur. Seyrek sözlük öğrenmesi için popüler bir sezgisel yöntem K-SVD algoritmasıdır. Seyrek sözlük öğrenmesi çeşitli bağlamlarda uygulanmıştır. Sınıflandırmada sorun, daha önce görülmemiş bir eğitim örneğinin ait olduğu sınıfı belirlemektir. Her sınıfın önceden oluşturulduğu bir sözlük için, sınıfla ilgili sözlük tarafından en iyi şekilde temsil edilen yeni bir eğitim örneği ilişkilendirilir. Görüntü parazitlenmesinde seyrek sözlük öğrenmesi de uygulanmıştır. Ana fikir, temiz bir görüntü yamasının bir görüntü sözlüğü ile seyrek olarak temsil edilebileceğidir, ancak gürültü olamaz.

Robot öğrenme:

Gelişimsel robot biliminde, robot öğrenme algoritmaları, öz rehberli keşif ve insanlarla sosyal etkileşim yoluyla kümülatif olarak yeni beceriler kazanmak için müfredat olarak da bilinen kendi öğrenme deneyimleri dizilerini oluşturur. Bu robotlar aktif öğrenme, olgunlaşma, motor sinerjileri ve taklit gibi rehberlik mekanizmalarını kullanır.

Birleşik öğrenme:

Birleşik öğrenme, eğitim sürecini ademi merkeziyetçi hale getiren ve verilerini merkezi bir sunucuya göndermeye gerek kalmadan kullanıcıların gizliliğinin korunmasına izin veren eğitim makinesi öğrenme modellerine uyarlanmış bir Dağıtılmış Yapay Zeka biçimidir. Bu, eğitim sürecini birçok cihaza dağıtarak verimliliği de arttırır. Örneğin, Gboard, bireysel aramaları Google'a geri göndermek zorunda kalmadan kullanıcıların cep telefonlarında arama sorgusu tahmin modellerini eğitmek için birleşik makine öğrenmesi kullanır.

Sıralı öğrenme:

Sıralı öğrenme, mantıklı bir şekilde öğretme ve öğrenme yöntemidir.
Sıralı öğrenme sürecini kategorize edebileceğiniz farklı kategoriler:

- Sıra tahmini
- Sıra oluşturma
- Sıra tanıma
- Sıralı karar

Toplu istatistiksel öğrenme:

İstatistiksel öğrenme teknikleri, görünmeyen veya gelecekteki veriler hakkında tahminlerde bulunabilen bir dizi gözlemlenen veriden bir işlevi veya öngörücüyü öğrenmeye izin verir. Bu teknikler, öğrenilen tahmincinin gelecekteki görünmeyen veriler üzerindeki performansı hakkında, veri oluşturma sürecine ilişkin istatistiksel bir varsayıma dayalı olarak garantiler sağlar.

PAC öğrenimi:

PAC (Probably Approximately Correct) öğrenme, öğrenme algoritmalarını ve bunların istatistiksel verimliliklerini analiz etmek için tanıtılan bir öğrenme çerçevesidir.

PCA (Temel Bileşenler Analizi), KPCA (Çekirdek Tabanlı Temel Bileşen Analizi) ve ICA (Bağımsız Bileşen Analizi), boyut azaltma için kullanılan önemli özellik çıkarma teknikleridir.

Endüktif (Tümevarım) makine öğrenimi:

Tümevarımlı makine öğrenimi, bir sistemin gözlemlenen bir dizi örnekten genel bir kural oluşturmaya çalıştığı örneklerle öğrenme sürecini içerir.

Endüktif Mantık Programlama (ILP), arka plan bilgisini ve örnekleri temsil eden mantıksal programlamayı kullanan bir makine öğrenimi alt alanıdır.

Endüktif mantık programlama (ILP), giriş örnekleri, arka plan bilgisi ve hipotezler için tekdüze bir sunum olarak mantık programlamayı kullanarak kural öğrenmeye bir yaklaşımdır. Bilinen arka plan bilgisinin bir kodlaması ve gerçeklerin mantıksal bir veritabanı olarak temsil edilen bir dizi örnek göz önüne alındığında, bir ILP sistemi, tüm olumlu ve olumsuz örnekleri içeren varsayılmış bir mantık programı türetecektir. Endüktif programlama, fonksiyonel programlar gibi hipotezleri (ve sadece mantık programlamayı değil) temsil etmek için her türlü programlama dilini göz önünde bulunduran ilgili bir alandır.

Endüktif mantık programlama özellikle biyoinformatik ve doğal dil işlemede yararlıdır. Gordon Plotkin ve Ehud Shapiro, endüktif makine öğrenmesi için ilk teorik temeli mantıklı bir ortamda ortaya koydu. Shapiro ilk uygulamalarını (Model Çıkarım Sistemi) 1981'de kurdu: mantık programlarını pozitif ve negatif örneklerden indüktif olarak çıkartan bir Prolog programı. Buradaki tümevarım terimi, iyi düzenlenmiş bir kümenin tüm üyeleri için bir özellik kanıtlayan, matematiksel tümevarım yerine gözlemlenen gerçekleri açıklayan bir teori öneren felsefi tümevarım anlamına gelir.

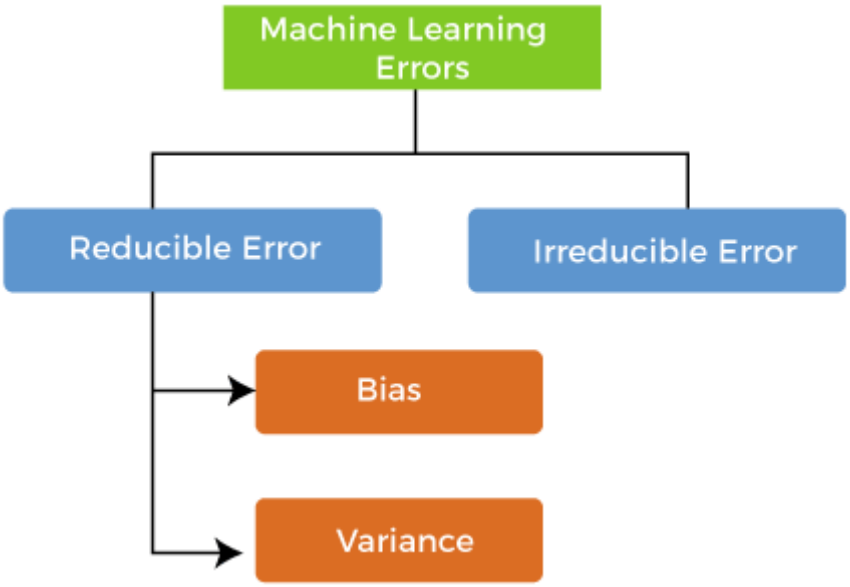
Tembel öğrenme algoritması:

Örnek tabanlı öğrenme algoritması, sınıflandırma gerçekleştirilinceye kadar indüksiyon veya genelleme sürecini geciktirdiği için Tembel öğrenme algoritması olarak da adlandırılır.

4. Makine Öğrenmesinde Performans Paremetreleri

Makine öğreniminde hata, bir algoritmanın önceden bilinmeyen veri kümesi için ne kadar doğru tahminlerde bulunabileceğinin bir ölçüsüdür. Bu hatalar temelinde, belirli veri kümesinde en iyi performansı gösterebilecek makine öğrenimi modeli seçilir. Makine öğreniminde başlıca iki tür hata vardır:

İndirgenebilir hatalar: Model doğruluğunu iyileştirmek için bu hatalar azaltılabilir. Bu tür hatalar ayrıca önyargı ve Varyans olarak sınıflandırılabilir.



İndirgenemez hatalar: Bu hatalar modelde her zaman mevcut olacaktır.
Hangi algoritmanın kullanıldığına bakılmaksızın. Bu hataların nedeni, değeri düşürülemeyen bilinmeyen değişkenlerdir**.**

Model performanslarının test edilmesi:

1) Performansı değerlendirme ihtiyacı.
2) Model değerlendirme teknikleri.
3) Sınıflandırma modeli değerlendirme ölçütleri.
4) Regresyon modeli değerlendirme metrikleri.

Makine öğrenimi modellerinin doğru ve güvenilir tahminler sunması beklenir. Tahminlerine güvenmek için makine öğrenimi modellerinin test verilerini nasıl genellediğini değerlendirmek önemlidir.

Veri kümesinin sınıflandırılması:

Eğitim seti: Tahmine dayalı modeller oluşturmak için kullanılan bir veri kümesinin parametrelerini ayarlayarak bir modeli eğitmek için kullanılacak bir dizi girdi örneği içerir.

Doğrulama seti: Eğitim aşamasında oluşturulan modelin performansını değerlendirmek olan bir veri kümesinin alt kümesidir. Bir modeli periyodik olarak değerlendirir ve model parametrelerinin ince ayarına izin verir. Tüm modelleme algoritmaları bir doğrulama kümesine ihtiyaç duymayabilirler.

Test seti: Aynı zamanda görünmeyen veriler olarak da bilinir. Bir modelin eğitim aşamasından sonra geçirdiği son değerlendirmedir. Bir test kümesi, bir modelin gelecekteki olası performansını değerlendirmek için kullanılan bir veri kümesinin alt kümesi olarak tanımlanır. Örneğin, bir model eğitime test setinden daha iyi uyuyorsa, büyük olasılıkla fazla uydurma mevcuttur.

Aşırı uydurma: Bir modelin veri kümesi tarafından açıklanabilecek olandan daha fazla parametre içerdiği anlamına gelir. Gürültülü veriler bozunmaya katkıda bulunur. Bu modellerin genelleştirilmesi, model veri kümesinden kastedilenden daha fazlasını öğrendiği için güvenilir değildir.

Her zaman bir modelin görünmeyen veriler üzerinde nasıl genellendiği test edilmelidir. Sınıflandırma problemleri için çok yaygın ve açık bir cevap, doğruluğunu ölçmektir. Model doğruluğunun model performansını ölçmek için etkili bir ölçüm olmadığı görülmektedir. Bir modelin performansını ölçmenin birçok yolu olduğu unutulmamalıdır.

Model değerlendirme teknikleri:

Bir modelin performansını değerlendirme teknikleri iki bölüme ayrılabilir: çapraz doğrulama ve bekleme. Bu tekniklerin her ikisi de model performansını değerlendirmek için bir test setinden yararlanır.

Hodout (bekletme) tekniği: Model performansının tarafsız bir tahminini elde etmek önemlidir. Holdout tekniğinin sunduğu şey tam olarak budur. Bu tarafsız tahmini elde etmek için, üzerinde eğittiğimiz verilerden farklı veriler üzerinde bir modeli test ederiz. Bu teknik, bir veri kümesini üç alt kümeye ayırır: eğitim, doğrulama ve test kümeleri.

Eğitim setinin modelin tahmin yapmasına yardımcı olduğunu ve test setinin modelin performansını değerlendirdiğini biliyoruz. Doğrulama seti, modelin parametrelerinde ince ayar yapmak için bir ortam sağlayarak modelin performansını değerlendirmeye de yardımcı olur. Buradan en iyi performans gösteren modeli seçiyoruz.

Holdout yöntemi, çok büyük bir veri kümesiyle uğraşırken idealdir, modelin fazla takılmasını önler ve daha düşük hesaplama maliyetlerine neden olur.

Bir işlev, bir dizi veri noktasına çok sıkı bir şekilde sığdığında, fazla uydurma olarak bilinen bir hata oluşur. Sonuç olarak, bir model görünmeyen veriler üzerinde düşük performans gösterir. Fazla uyumu tespit etmek için önce veri setimizi eğitim ve test setlerine ayırabiliriz. Daha sonra modelin performansını hem eğitim verileri hem de test verileri üzerinde izliyoruz.

Modelimiz, test seti ile karşılaştırıldığında eğitim setinde üstün performans sunuyorsa, fazla uydurma olma ihtimali yüksektir. Örneğin, bir model eğitim setinde %90 doğruluk sunarken test setinde %50 doğruluk sağlayabilir.

Model değerlendirme metrikleri:

Sınıflandırma sorunları için metrikler: Sınıflandırma problemlerine yönelik tahminler dört tür sonuç verir: gerçek pozitifler, gerçek negatifler, yanlış pozitifler ve yanlış negatifler.

Sınıflandırma doğruluğu: Sınıflandırma problemleri için en yaygın değerlendirme ölçütü doğruluktur. Yapılan tahminlerin (veya girdi örneklerinin) toplam sayısına karşı doğru tahminlerin sayısı olarak alınır. Ancak, bir modeli değerlendirmek için doğruluk kullanıldığı kadar, daha önce belirttiğimiz gibi model performansının net bir göstergesi değildir.

Sınıflandırma doğruluğu, her sınıfa ait örneklerin sayıca eşit olması durumunda en iyi sonucu verir. Bir eğitim setinde X sınıfından %97 ve Y sınıfından %3 örnek içeren bir senaryo düşünün. Bir model, X sınıfındaki her eğitim örneğini tahmin ederek çok kolay bir şekilde %97 eğitim doğruluğu elde edebilir.

Aynı model, %55 X numunesi ve %45 Y numunesi içeren bir test setinde test edildiğinde, test doğruluğu %55'e düşürülür. Bu nedenle sınıflandırma doğruluğu performansın açık bir göstergesi değildir. Yüksek düzeyde doğruluk elde etme konusunda yanlış bir his sağlar.

4.1. Modellerin Seçilmesi ve Değerlendirilmesi

Yüksek düzeyde, Makine Öğrenimi, istatistik ve hesaplamanın birleşimidir. Makine öğreniminin püf noktası, aslında istatistiksel tahminleri içeren algoritmalar veya matematiksel modeller kavramı etrafında döner. Bununla birlikte, herhangi bir modelin veri dağılımına bağlı olarak çeşitli sınırlamaları vardır. Hiçbiri tamamen doğru olamaz, çünkü bunlar sadece tahminlerdir. Bu sınırlamalar yaygın olarak önyargı ve varyans adıyla bilinir.

Yüksek önyargılı bir model, eğitim noktalarına fazla dikkat etmeyerek aşırı basitleştirecektir (örn.: Doğrusal Regresyonda, veri dağılımından bağımsız olarak, model her zaman doğrusal bir ilişki üstlenecektir).

Yüksek varyansa sahip bir model, daha önce görmediği test noktaları için genelleme yapmayarak kendisini eğitim verileriyle sınırlayacaktır (örneğin: Rastgele Orman ile maksimum_derinlik = Yok).

Sorun, rastgele bir orman algoritması ile bir gradyan artırma algoritması arasında veya aynı karar ağacı algoritmasının iki varyasyonu arasında seçim yapmamız gerektiğinde olduğu gibi, sınırlamalar ince olduğunda ortaya çıkar. Her ikisi de yüksek varyansa ve düşük önyargıya sahip olma eğiliminde olacaktır.

İşte burada model seçimi ve model değerlendirmesi devreye giriyor:

- Model seçimi ve model değerlendirmesi nedir?
- Etkili model seçim yöntemleri (yeniden örnekleme ve olasılıksal yaklaşımlar)
- Popüler model değerlendirme yöntemleri
- Önemli Makine Öğrenimi modeli ödünleşimleri

Model değerlendirmesi, test verileri üzerindeki modellerin doğruluğunu değerlendirme yöntemidir. Test verileri, model tarafından daha önce görülmeyen veri noktalarından oluşur.

Model seçimi, bireysel modeller gerekli kriterlere göre değerlendirildikten sonra en iyi modeli seçme tekniğidir.

Yeniden örnekleme yöntemleri, adından da anlaşılacağı gibi, modelin eğitim almadığı veri örneklerinde iyi performans gösterip göstermediğini kontrol etmek için veri örneklerini yeniden düzenlemenin basit teknikleridir. Başka bir deyişle, yeniden örnekleme, modelin iyi bir genelleme yapıp yapmayacağını anlamamıza yardımcı olur.

Rastgele Bölmeler, verilerin bir yüzdesini eğitim, test ve tercihen doğrulama kümelerine rastgele örneklemek için kullanılır. Bu yöntemin avantajı, orijinal popülasyonun her üç

kümede de iyi temsil edilme şansının yüksek olmasıdır. Daha resmi bir ifadeyle, rastgele bölme, verilerin önyargılı bir şekilde örneklenmesini önleyecektir.

Model seçiminde doğrulama setinin kullanımına dikkat etmek çok önemlidir. Doğrulama seti ikinci test setidir ve biri neden iki test seti var diye sorabilir. Özellik seçimi ve model ayarlama sürecinde, model değerlendirmesi için test seti kullanılır. Bu, model parametrelerinin ve özellik setinin, test setinde en uygun sonucu verecek şekilde seçildiği anlamına gelir. Böylece, son değerlendirme için tamamen görünmeyen veri noktalarına sahip (tuning ve özellik seçim modüllerinde kullanılmamış) doğrulama seti kullanılır.

Zamana Dayalı Bölme: Rastgele bölmelerin mümkün olmadığı bazı veri türleri vardır. Örneğin, hava tahmini için bir model eğitmemiz gerekiyorsa, verileri rastgele eğitim ve test setlerine bölemeyiz. Bu mevsimsel deseni karıştıracak! Bu tür veriler genellikle - Zaman Serileri terimi ile anılır.

Bu gibi durumlarda, zamana dayalı bir bölme kullanılır. Eğitim seti içinde bulunulan yılın son üç yılı ve 10 ayı için veri içerebilir. Son iki ay, test veya doğrulama seti için ayrılabilir.

Ayrıca, modelin belirli bir tarihe kadar eğitildiği ve eğitim penceresinin bir gün kaymaya devam edeceği şekilde ileriki tarihlerde yinelemeli olarak test edildiği bir pencere kümeleri kavramı da vardır (sonuç olarak, test seti de bir gün azalır). Bu yöntemin avantajı, modeli stabilize etmesi ve test seti çok küçük olduğunda (örneğin 3 ila 7 gün) fazla takmayı önlemesidir.

Ancak, zaman serisi verilerinin dezavantajı, olayların veya veri noktalarının karşılıklı olarak bağımsız olmamasıdır. Bir olay, ardından gelen her veri girişini etkileyebilir.

Örneğin, iktidar partisindeki bir değişiklik, önümüzdeki yıllarda nüfus istatistiklerini önemli ölçüde değiştirebilir. Veya kötü şöhretli koronavirüs pandemisi, önümüzdeki birkaç yıl için ekonomik veriler üzerinde büyük bir etkiye sahip olacak.

Böyle bir durumda hiçbir makine öğrenme modeli geçmiş verilerden öğrenemez çünkü olaydan önceki ve sonraki veri noktaları büyük farklılıklar gösterir.

K-Katlama Çapraz Doğrulama: Çapraz doğrulama tekniği, veri kümesini rastgele karıştırarak ve ardından onu k gruba bölerek çalışır. Daha sonra, her grup üzerinde yineleme yapıldığında, diğer tüm gruplar eğitim setinde bir araya getirilirken grubun bir test seti olarak düşünülmesi gerekir. Model test grubu üzerinde test edilir ve işlem k grupları için devam eder. Böylece, sürecin sonunda, k farklı test grubu üzerinde k farklı sonuç elde edilir. En iyi model daha sonra en yüksek puana sahip olanı seçerek kolayca seçilebilir.

Tabakalı K-Katlama (Stratified K-Fold): Tabakalı K-Katlama işlemi, tek bir fark noktasıyla K-Katlama çapraz doğrulama işlemine benzer – k-kat çapraz doğrulamadan farklı olarak, hedef değişkenin değerleri, katmanlı k-katlamada dikkate alınır. Örneğin, hedef değişken 2 sınıflı kategorik bir değişkense, tabakalı k-katlama, eğitim seti ile karşılaştırıldığında her bir test katının iki sınıfın eşit oranını almasını sağlar. Bu, model değerlendirmesini daha doğru ve model eğitimini daha az önyargılı hale getirir.

Önyükleme (Bootstrap): Bootstrap, kararlı bir model elde etmenin en güçlü yollarından biridir. Rastgele örnekleme kavramını takip ettiğinden rastgele bölme tekniğine yakındır.

İlk adım, bir örnek boyutu seçmektir (genellikle orijinal veri kümesinin boyutuna eşittir). Daha sonra, orijinal veri kümesinden rastgele bir örnek veri noktası seçilmeli ve önyükleme örneğine eklenmelidir. Ekleme işleminden sonra numunenin orijinal numuneye geri konması gerekir. Bu işlemin N kez tekrarlanması gerekir, burada N örnek boyutudur.

Bu nedenle, orijinal veri kümesinden değiştirme ile veri noktalarını örnekleyerek önyükleme örneğini oluşturan bir yeniden örnekleme tekniğidir. Bu, önyükleme örneğinin aynı veri noktasının birden çok örneğini içerebileceği anlamına gelir.

Model, önyükleme örneğinde eğitilir ve ardından önyükleme örneğine ulaşmayan tüm veri noktalarında değerlendirilir. Bunlara torba dışı örnekler denir.

Olasılık ölçüleri (Probabilistic measures): Olasılık Ölçütleri sadece model performansını değil, aynı zamanda model karmaşıklığını da hesaba katar. Model karmaşıklığı, modelin verilerdeki varyansı yakalama yeteneğinin ölçüsüdür. Örneğin, doğrusal regresyon algoritması gibi oldukça önyargılı bir model daha az karmaşıktır ve diğer yandan bir sinir ağı karmaşıklığı çok yüksektir. Burada dikkat edilmesi gereken bir diğer önemli nokta, olasılık ölçütlerinde dikkate alınan model performansının sadece eğitim setinden hesaplanmasıdır. Bir bekletme testi seti genellikle gerekli değildir. Bununla birlikte, biraz dezavantaj, olasılık ölçümlerinin modellerin belirsizliğini dikkate almaması ve karmaşık modeller yerine daha basit modelleri seçme şansına sahip olması gerçeğinde yatmaktadır.

Akaike Bilgi Kriteri (AIC: Akaike Information Criterion): Her modelin tamamen doğru olmadığı yaygın bir bilgidir. Her zaman KL bilgi metriği kullanılarak ölçülebilen bir miktar bilgi kaybı vardır. Kulback-Liebler veya KL sapması, iki değişkenin olasılık dağılımındaki farkın ölçüsüdür.

Bir istatistikçi olan Hirotugu Akaike, KL Bilgisi ile Maksimum Olabilirlik arasındaki ilişkiyi dikkate aldı (maksimum olasılıkta, parametreler ve belirli bir olasılık dağılımı veriliyken, bir X veri noktasını gözlemlemenin koşullu olasılığını maksimize etmek istenir) ve şu kavramı geliştirdi: Bilgi Kriteri (veya IC). Bu nedenle, Akaike'nin IC veya AIC'si bilgi kaybının ölçüsüdür.

Bu şekilde iki farklı model arasındaki uyumsuzluk yakalanır ve en az bilgi kaybı olan model tercih edilen model olarak önerilir.

$$AIC = (2K - 2log(L))/N$$

- K = bağımsız değişkenlerin veya tahmin edicilerin sayısı
- L = modelin maksimum olasılığı
- N = eğitim setindeki veri noktalarının sayısı (özellikle küçük veri setleri durumunda faydalıdır)

AIC'nin sınırlaması, daha az eğitim bilgisi kaybeden karmaşık modelleri seçme eğiliminde olduğu için genelleme modelleri konusunda çok iyi olmamasıdır.

Bayes Bilgi Kriteri (BIC: Bayesian Information Criterion**):** BIC, Bayes olasılık kavramından türetilmiştir ve maksimum olabilirlik tahmini altında eğitilmiş modeller için uygundur.

$$BIC = K * log(N) - 2log(L)$$

• K = bağımsız değişken sayısı
• L = maksimum olasılık
• N = Eğitim setindeki örnekleyici/veri noktalarının sayısı

BIC, modeli karmaşıklığından dolayı cezalandırır ve tercihen veri kümesinin boyutu çok küçük olmadığında kullanılır (aksi takdirde çok basit modellere yerleşme eğilimi gösterir).

Minimum Açıklama Uzunluğu (MDL: Minimum Description Length): MDL, bir olayı bir olasılık dağılımından veya rastgele bir değişkenden temsil etmek için gereken ortalama bit sayısını ölçen entropi gibi niceliklerle ilgilenen Bilgi teorisinden türetilmiştir. MDL veya minimum açıklama uzunluğu, modeli temsil etmek için gereken bu tür bitlerin minimum sayısıdır.

$$MDL = L(h) + L(D|h)$$

- d = model
- D = model tarafından yapılan tahminler
- L(h) = modeli temsil etmek için gereken bit sayısı
- L(D | h) = modelden tahminleri temsil etmek için gereken bit sayısı

Yapısal Risk Minimizasyonu (SRM: Structural Risk Minimization)
Makine öğrenimi modelleri, bir dizi sonlu veriden genelleştirilmiş bir teori tanımlamanın kaçınılmaz sorunuyla karşı karşıyadır. Bu, modelin birincil öğrenme kaynağı olan eğitim verilerine önyargılı olduğu aşırı uyum durumlarına yol açar. SRM, modelin karmaşıklığını verilere uymadaki başarısına karşı dengelemeye çalışır.

ML modelleri nasıl değerlendirilir?
Modeller birden fazla metrik kullanılarak değerlendirilebilir. Ancak, bir değerlendirme metriğinin doğru seçimi çok önemlidir ve çoğu zaman çözülmekte olan soruna bağlıdır. Çok çeşitli metriklerin net bir şekilde anlaşılması, değerlendiricinin sorun ifadesi ile metrik arasında uygun bir eşleşme bulmasına yardımcı olabilir.

Sınıflandırma metrikleri:

Her sınıflandırma modeli tahmini için, doğru ve yanlış sınıflandırılan test senaryolarının sayısını gösteren, karışıklık matrisi adı verilen bir matris oluşturulabilir.

Şuna benziyor (1 -Positive ve 0 -Negative, hedef sınıflardır):
gerçek 0
gerçek 1
tahmin edilen 0
Gerçek Negatifler (TN)
Yanlış Negatifler (FN)

Tahmin edilen 1
Yanlış Pozitifler (FP)
Gerçek Pozitifler (TP)
• TN: Doğru sınıflandırılmış negatif vaka sayısı
• TP: Doğru sınıflandırılmış pozitif vaka sayısı
• FN: Yanlış bir şekilde negatif olarak sınıflandırılan pozitif vakaların sayısı
• FP: Doğru olarak pozitif olarak sınıflandırılan negatif vakaların sayısı

Doğruluk (Accuracy): Doğruluk en basit ölçüdür ve doğru sınıflandırılan test senaryolarının toplam test senaryosu sayısına bölünmesiyle tanımlanabilir.

$$Accuracy = (TP+TN)/(TP+TN+FP+FN)$$

Çoğu genel soruna uygulanabilir, ancak dengesiz veri kümeleri söz konusu olduğunda çok kullanışlı değildir.

Örneğin, banka verilerinde dolandırıcılık tespit ediyorsak, dolandırıcılığın dolandırıcılık dışı durumlara oranı 1:99 olabilir. Bu gibi durumlarda doğruluk kullanılırsa, tüm test durumları

dolandırıcılık dışı olarak tahmin edilerek modelin %99 doğru olduğu ortaya çıkacaktır. %99 doğru model tamamen işe yaramaz olacaktır.

Bir model, tüm 1000 (örneğin) veri noktasını sahtekarlık dışı olarak tahmin edecek şekilde kötü eğitilmişse, 10 sahtekarlık veri noktasında eksik olacaktır. Doğruluk ölçülürse, o modelin 990 veri noktasını doğru tahmin ettiğini gösterecek ve böylece (990/1000)*100 = %99 doğruluk oranına sahip olacaktır!

Bu nedenle doğruluk, modelin sağlığının yanlış bir göstergesidir.
Bu nedenle, böyle bir durum için, model tarafından tamamen gözden kaçırılan on dolandırıcılık veri noktasına odaklanabilen bir metrik gereklidir.

Kesinlik (Precision): Kesinlik, sınıflandırmanın doğruluğunu belirlemek için kullanılan ölçüdür.

$$Precision = TP/(TP + FP)$$

Sezgisel olarak, bu denklem, doğru pozitif sınıflandırmaların, tahmin edilen pozitif sınıflandırmaların toplam sayısına oranıdır. Kesir ne kadar büyükse, hassasiyet de o kadar yüksek olur, bu da modelin pozitif sınıfı doğru bir şekilde sınıflandırma yeteneğinin daha iyi olduğu anlamına gelir.

Öngörücü bakım probleminde (bir makinenin ne zaman onarılması gerektiğini önceden tahmin etmek gerektiği yerde), hassasiyet devreye girer. Bakım maliyeti genellikle yüksektir ve bu nedenle yanlış tahminler şirket için kayıplara neden olabilir. Bu gibi durumlarda, modelin pozitif sınıfı doğru bir şekilde sınıflandırma ve yanlış pozitiflerin sayısını azaltma yeteneği çok önemlidir!

Geriçağırma (Recall): Geri çağırma, toplam pozitif vaka sayısından doğru bir şekilde tespit edilen pozitif vakaların sayısını söyler.

$$Recall = TP/(TP + FN)$$

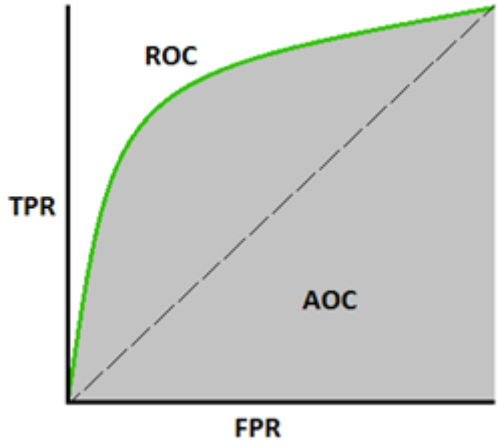
Hile sorununa geri dönersek, geri çağırma değeri hile davalarında çok faydalı olacaktır çünkü yüksek bir geri çekme değeri, toplam hile sayısından çok sayıda hile vakasının tespit edildiğini gösterecektir.

F1 Score: F1 puanı, Geri Çağırma ve Kesinliğin harmonik ortalamasıdır ve bu nedenle her birinin güçlü yönlerini dengeler.

Onarım gerektirebilecek uçak parçalarının tanımlanmasında olduğu gibi, hem hatırlamanın hem de kesinliğin değerli olabileceği durumlarda yararlıdır. Burada, şirketin maliyetinden tasarruf etmek için hassasiyet gerekecektir (çünkü uçak parçaları son derece pahalıdır) ve makinelerin stabil olmasını ve insan yaşamı için bir tehdit oluşturmamasını sağlamak için geri çağırma gerekecektir.

$$F1Score = 2 * ((precision * recall)/(precision + recall))$$

AUC-ROC: ROC eğrisi, yanlış pozitif hıza (TN / (TN+FP) karşı gerçek pozitif oranın (hatırlatma) bir grafiğidir. AUC-ROC, Alıcı Çalışma Karakteristikleri Altında Alan anlamına gelir ve alan ne kadar yüksek olursa, model performansı o kadar iyi olur. Eğri %50 çapraz çizgiye yakınsa, modelin çıktı değişkenini rastgele tahmin ettiğini gösterir.



Kayıp listesi **(Log loss)** çok etkili bir sınıflandırma metriğidir ve olabilirlik işlevinin, modelin gözlemlenen sonuç kümesinin ne kadar olası olduğunu düşündüğünü önerdiği -1* log'a (olasılık işlevi) eşdeğerdir.

Olabilirlik işlevi çok küçük değerler sağladığından, bunları yorumlamanın daha iyi bir yolu, değerleri günlüğe dönüştürmek ve daha düşük bir kayıp puanı daha iyi bir model önerecek şekilde ölçümün sırasını tersine çevirmek için negatif eklemektir.

Kazanç ve Artış Grafikleri: Kazanç ve artış çizelgeleri, tıpkı kafa karışıklığı matrisi gibi, ancak ince ancak önemli bir farkla model performansını değerlendiren araçlardır. Karışıklık matrisi, modelin tüm popülasyon veya tüm test seti üzerindeki performansını belirlerken, kazanç ve artış çizelgeleri, modeli tüm popülasyonun bölümleri üzerinde değerlendirir. Bu nedenle, popülasyonun her %'si (x ekseni) için bir puanımız (y ekseni) vardır.

Kaldırma çizelgeleri, bir modelin rastgele tahminlere kıyasla getirdiği gelişmeyi ölçer. İyileştirme, 'kaldırma' olarak adlandırılır.

K-S Grafiği: K-S şeması veya Kolmogorov-Smirnov şeması, iki dağılım arasındaki ayrım derecesini belirler – pozitif sınıf dağılımı ve negatif sınıf dağılımı. Fark ne kadar yüksek olursa, model olumlu ve olumsuz durumları ayırmada o kadar iyidir.

Regresyon metrikleri: Regresyon modelleri, ayrı çıktı değişkenlerine sahip sınıflandırma modellerinin aksine, sürekli bir çıktı değişkeni sağlar. Bu nedenle, regresyon modellerini değerlendirmek için metrikler buna göre tasarlanmıştır.

Ortalama Kare Hat (Mean Squared Error – MSE): MSE, gerçek değer ile tahmin edilen değer (hata) arasındaki farkı hesaplayan, karesini alan ve ardından tüm hataların ortalamasını sağlayan basit bir ölçümdür.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widetilde{y}_i)^2$$

MSE aykırı değerlere karşı çok hassastır ve aksi takdirde iyi donatılmış model tahminlerinde birkaç aykırı değer mevcut olsa bile çok yüksek bir hata değeri gösterecektir.

Kök Ortalama Kare Hatası (Root Mean Squared Error – RMSE): RMSE, MSE'nin köküdür ve hataların ölçeğini gerçek değerlere yaklaştırarak daha yorumlanabilir hale getirmeye yardımcı olduğu için faydalıdır.

Ortalama Mutlak Hata (Mean Absolute Error - MAE): MAE, mutlak hata değerlerinin ortalamasıdır (gerçekler – tahminler)

$$MAE = \frac{1}{n}\sum_{i=1}|x_i - x|$$

Aykırı değerlerin belirli bir dereceye kadar yok sayılması isteniyorsa, kare terimlerin kaldırılmasıyla aykırı değerlerin cezasını önemli ölçüde azalttığı için MAE seçimdir.

Kök Ortalama Kare Kayıtlı Hata Listesi (RMSLE Root Mean Squared Log Error): RMSLE'de, gerçek ve tahmin edilen değerlerle birlikte eklenen bir günlük işlevi dışında RMSE ile aynı denklem izlenir.

$$RMSLE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(log(x_i+1)) - (log(y_i+1))^2}$$

x gerçek değerdir ve y tahmin edilen değerdir. Bu, log işleviyle daha yüksek hata oranlarını küçümseyerek aykırı değerlerin etkisini azaltmaya yardımcı olur. Ayrıca, RMSLE, günlükleri kullanarak (tüm hata değerlerini karşılaştırarak) göreli bir hatayı yakalamaya yardımcı olur.

R-Squared: R-Square, bağımsız değişkenler veya tahmin ediciler yardımıyla yakalanabilen hedef değişkenin varyans oranını belirlemeye yardımcı olur.

$$R - Squared = 1 - \frac{Unexplained\ Variation}{Total\ Variation} = 1 - \frac{Variance\ (Model)}{Variance\ (Average)}$$

Ancak R-karenin devasa bir sorunu var. Diyelim ki, atanan ağırlığı w olan bir modele ilgisiz yeni bir özellik eklendi. Model, yeni tahmin edici ile hedef değişken arasında kesinlikle hiçbir korelasyon bulamazsa, w 0'dır. Bununla birlikte, rastgelelik nedeniyle küçük bir pozitif ağırlık (w>0) ekleyen ve yeni bir minimum kayıp elde edilen hemen hemen her zaman küçük bir korelasyon vardır. fazla takılma nedeniyle.

Bu nedenle, herhangi bir yeni özellik eklenmesiyle R-kare artar. Bu nedenle, yeni özellikler eklendiğinde değerinde azalma olmaması, modelin daha az özellikle daha iyi performans gösterip göstermediğini belirleme yeteneğini sınırlar.

Düzeltilmiş R-Kare: Düzeltilmiş R-Kare, eklenen özelliklerle değerin azaltılamaması sorununu ortadan kaldırarak R-Kare sorununu çözer. Daha fazla özellik eklendikçe puanı cezalandırır.

$$adjR_2 = 1 - (1 - R^2)\frac{n-1}{n-m-1}$$

Buradaki payda, öznitelik sayısı arttıkça artan sihirli unsurdur. Bu nedenle, genel değeri artırmak için R2'de önemli bir artış gereklidir.

Kümeleme metrikleri: Kümeleme algoritmaları, veri noktası gruplarını tahmin eder ve bu nedenle mesafeye dayalı metrikler en etkilidir.

Dunn Index: Dunn Index, düşük varyansa sahip (kümedeki tüm üyeler arasında) ve kompakt olan kümeleri belirlemeye odaklanır. Farklı kümelerin ortalama değerlerinin de birbirinden uzak olması gerekir.

• δ(Xi, Yj) kümeler arası mesafedir, yani Xi ve Xj arasındaki mesafe

• Δ(Xk) Xk kümesinin kümeler arası uzaklığıdır, yani Xk kümesi içindeki uzaklık

Ancak Dunn indeksinin dezavantajı, daha fazla sayıda küme ve daha fazla boyut ile hesaplama maliyetinin artmasıdır.

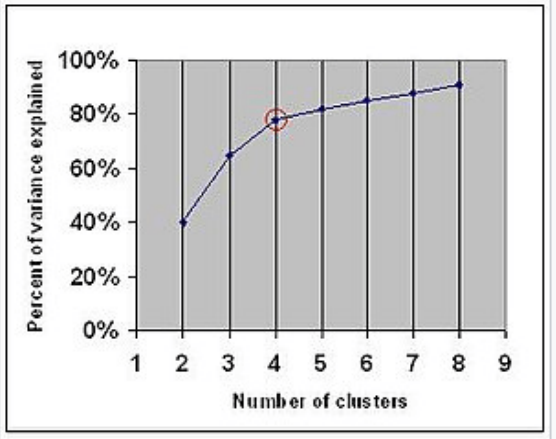
Siluet Katsayısı: Siluet Katsayısı, bir kümedeki her noktanın diğer kümelerdeki her noktaya -1 ile +1 aralığında ne kadar yakın olduğunu izler:

• Daha yüksek Silhouette değerleri (+1'e yakın) iki farklı kümeden alınan örnek noktaların çok uzakta olduğunu gösterir.

• 0, noktaların karar sınırına yakın olduğunu gösterir

• ve -1'e yakın değerler, noktaların kümeye yanlış atandığını gösterir.

Dirsek yöntemi (Elbow method)**:** Dirsek yöntemi, bir veri kümesindeki küme sayısını, x eksenindeki küme sayısını y ekseninde açıklanan varyans yüzdesine karşı çizerek belirlemek için kullanılır. Eğrinin aniden büküldüğü x eksenindeki noktanın (dirsek) optimal küme sayısını önerdiği kabul edilir.



Explained variance. The "elbow" is indicated by the red circle. The number of clusters chosen should therefore be 4.

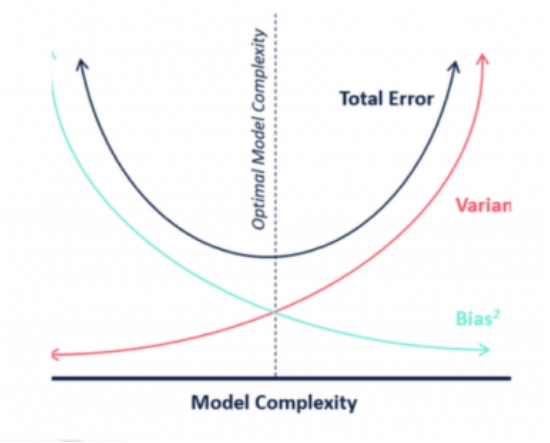
Takaslar

Önyargıya karşı varyans: Bu makalenin başında, model değerlendirmesi ve seçimi ihtiyacını anlamak için yanlılık ve varyanstan bahsetmek önemliydi. Ancak bunu daha ayrıntılı anlamak, değerlendirme ve seçme kavramlarını iyi kavramak için gereklidir.

Önyargı, bir model kesinlikle varsayımlarla yönetildiğinde ortaya çıkar – doğrusal regresyon modelinin çıktı değişkeninin bağımsız değişkenlerle ilişkisinin düz bir çizgi olduğunu varsayması gibi. Bu, gerçek değerler bağımsız değişkenlerle doğrusal olmayan bir şekilde ilişkili olduğunda eksik uydurmaya yol açar.

Bir model eğitim setine çok fazla odaklandığında ve varyasyonları çok yakından öğrendiğinde ve genellemeden ödün verdiğinde varyans yüksektir. Bu, aşırı uyumaya yol açar.

Optimal bir model, en düşük yanlılığa ve varyansa sahip olandır ve bu iki özellik dolaylı olarak orantılı olduğundan, bunu başarmanın tek yolu ikisi arasında bir ödünleşmedir. Bu nedenle, model seçimi, aşağıdaki resimdeki gibi yanlılık ve varyans kesişecek şekilde olmalıdır.



Bu, kullanımdaki modelin hiperparametrelerini yinelemeli olarak ayarlayarak başarılabilir (Hiperparametreler, model işlevlerine beslenen giriş parametreleridir). Her iterasyondan sonra model değerlendirmesi uygun bir metrik kullanılarak yapılmalıdır.
Öğrenme eğrileri

Model eğitiminin veya oluşturmanın ilerlemesini izlemenin en iyi yolu, öğrenme eğrilerini kullanmaktır. Bu eğriler, bir dizi hiperparametre kombinasyonundaki optimal noktaları belirlemeye yardımcı olur ve model seçimi ve model değerlendirme sürecinde büyük ölçüde yardımcı olur.
Tipik olarak, bir öğrenme eğrisi, y ekseninde model performansındaki öğrenmeyi veya gelişmeyi ve x ekseninde zaman veya deneyimi izlemenin bir yoludur.

En popüler iki öğrenme eğrisi şunlardır:

• Eğitim Öğrenme Eğrisi – Bir eğitim süreci sırasında fazla mesai değerlendirme metrik puanını etkin bir şekilde çizer ve böylece eğitim sırasında modelin öğrenimini veya ilerlemesini izlemeye yardımcı olur.
• Doğrulama Öğrenme Eğrisi – Bu eğride, değerlendirme metrik puanı doğrulama setinde zamana karşı çizilir.
•
Bazen, eğitim eğrisinin bir gelişme gösterdiği, ancak doğrulama eğrisinin bodur performans gösterdiği durumlar olabilir.

Bu, modelin fazla uyumlu olduğu ve önceki yinelemelere geri döndürülmesi gerektiği gerçeğinin göstergesidir. Başka bir deyişle, doğrulama öğrenme eğrisi, modelin ne kadar iyi genelleme yaptığını tanımlar.

Bu nedenle, eğitim öğrenme eğrisi ile doğrulama öğrenme eğrisi arasında bir ödünleşim vardır ve model seçim tekniği, hem eğrilerin kesiştiği hem de en düşük olduğu noktaya dayanmalıdır.

Bir makine öğrenimi algoritması değerlendirin ve seçin
Farklı makine öğrenimi algoritmaları, farklı eğilimler ve kalıplar arar. Tek bir algoritma, tüm veri kümeleri veya tüm kullanım durumları için en iyisi değildir. En iyi çözümü bulmak için birçok deney yapmanız, makine öğrenimi algoritmalarını değerlendirmeniz ve hiperparametrelerini ayarlamanız gerekir.

En iyi çözüm nasıl bulunur
İlk olarak, modelinizi değerlendirmek ve bir algoritma seçimini doğrulamak için bir model performans göstergesi seçer, gerekçelendirir ve uygularsınız. Model performans göstergelerinin örnekleri arasında F1 puanı, gerçek pozitif oran ve küme içi hata karesi toplamı yer alır.
Algoritmanızı en az bir derin öğrenme ve en az bir derin olmayan öğrenme algoritmasında uygulayın. Ardından, model performansını karşılaştırın ve belgeleyin.
En azından özellik oluşturma görevini içeren süreç modelinde en az bir yineleme daha uygulayın. Veri normalleştirme ve temel bileşen analizi (PCA) gibi model performansı üzerindeki etkiyi kaydedin. Algoritma sınıfına ve veri kümesi boyutuna bağlı olarak, sorununuzu çözmek için belirli teknolojileri veya çerçeveleri seçebilirsiniz.

Algoritmaları değerlendirirken ilgili terminolojiyi bilmek faydalıdır:

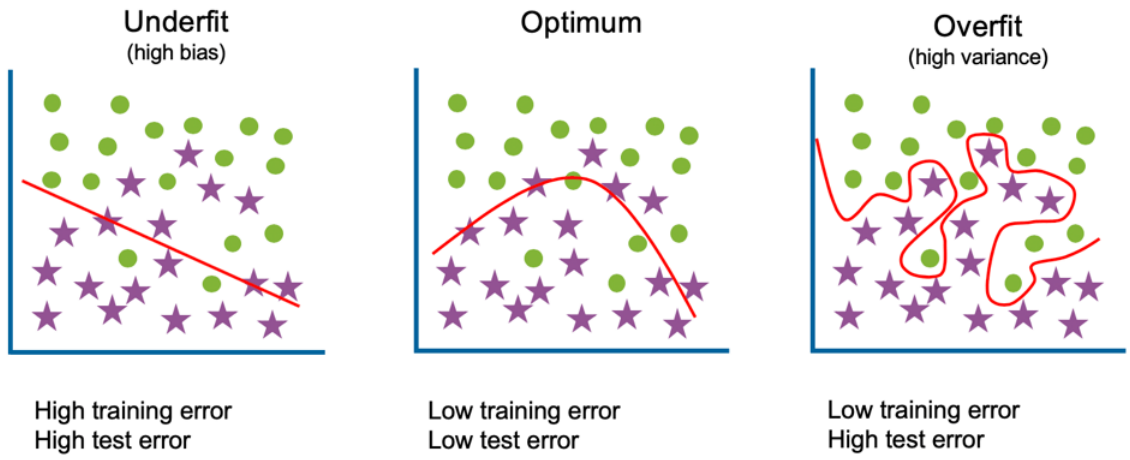
Term	Tanım
Açıklayıcı analitik	Açıklayıcı analitik, muhtemelen panolar ve raporlar oluşturmak için kullanılan en yaygın analiz türüdür. Daha önce meydana gelen olayları tanımlar ve özetler. Bir örnek, son beş yılda bir bölgedeki tüm mağazalarda her bir ürünün ürünlerinin nasıl satıldığını bilmek isteyen bir bakkal sahibidir.
Tahmine dayalı analitik	Tahmine dayalı analitik, gelecekteki sonuçları tahmin etmek için matematiksel ve istatistiksel yöntemlerin kullanılması anlamına gelir. Bakkal sahibi, stok seviyelerine karar verebilmek için önümüzdeki birkaç ay içinde kaç ürünün satılabileceğini anlamak istiyor.
Kuralcı analitik	Kuralcı analitik, bir dizi kısıtlamaya dayalı senaryoları simüle ederek iş kararlarını optimize etmek için kullanılır. Market sahibi, çalışanlar için bir personel programı oluşturmak istiyor, ancak bunu yapmak için uygunluk, tatil süresi, çalışma saatleri ve olası acil durumlar (kısıtlamalar) gibi faktörleri hesaba katması gerekiyor. Ayrıca herkes için çalışan bir program oluşturmalı ve işinin günlük olarak işlemesini sağlamalıdır.
Denetimli öğrenme	Denetimli öğrenme, etiketlenmiş eğitim verilerinden bir model öğretir ve görünmeyen veya gelecekteki veriler hakkında tahminler yapmanıza yardımcı olur. Eğitim sırasında algoritmaya doğru cevapları içeren bir veri seti verirsiniz (y etiketi). Ardından, doğru cevapları olan bir test veri seti ile model doğruluğunu doğrularsınız. Bir veri seti eğitim ve test setlerine bölünmelidir.
Sınıflandırma	Sınıflandırma ile, az sayıda ayrık değerli çıktıdan birini tahmin etmeye çalışıyorsunuz. Örneğin, etiketinizin ikili (ikili sınıflandırma) veya kategorik (çok sınıflı sınıflandırma) olup olmadığını tahmin etmeye çalışabilirsiniz.
regresyon	Regresyonda, öğrenme probleminin amacı sürekli değer çıktısını tahmin etmektir.
denetimsiz öğrenme	Bir veri seti verildiğinde, kümeleme gibi teknikleri kullanarak verideki benzer eğilimleri ve ilişkileri bulmaya çalışır.
Özellik	Özellik, modelin eğitilmesi için girdi olarak kullanılan bir niteliktir. Diğer adlar, boyut veya sütun içerir.

Bias	Önyargı, verilerinize mükemmel şekilde uyan bir modelin parametreleri ile algoritmanızın öğrendiği parametreler arasındaki beklenen farktır. Karar Ağaçları, K-En Yakın Komşular ve Destek Vektör Makineleri gibi düşük sapmalı algoritmalar, yüksek sapmalı algoritmalardan daha karmaşık desenler bulma eğilimindedir.
Variance	Varyans, algoritmanın eğitim verilerinden ne kadar etkilendiği ve yeni eğitim verileriyle parametrelerin ne kadar değiştiğidir. Doğrusal Regresyon, Lojistik Regresyon ve Naive Bayes gibi düşük varyanslı algoritmalar, yüksek varyanslı algoritmalardan daha az karmaşık modeller bulma eğilimindedir.
Underfitting	Model, veri içindeki kalıpları yakalamak için çok basittir. Model, eğitildiği verilerde ve görünmeyen verilerde düşük performans gösteriyor. Yüksek önyargı, düşük varyans. Yüksek eğitim hatası ve yüksek test hatası.
Overfitting	Model çok karmaşık veya çok spesifik, genellemeyen eğilimleri yakalıyor. Model, üzerinde eğitildiği verileri doğru bir şekilde tahmin eder, ancak görünmeyen verileri doğru bir şekilde tahmin etmez. Düşük önyargı, yüksek varyans. Düşük eğitim hatası ve yüksek test hatası.
Bias-Variance Trade-off	Bias-Varyance Trade-off, hem tren hem de test hatasını en aza indirerek doğru karmaşıklığa sahip bir model bulma anlamına gelir.

Algoritmaları eğitin, doğrulayın ve test edin

Makine öğrenimi algoritmaları örneklerden öğrenir. İyi verileriniz varsa, ne kadar çok örnek verirseniz, model verilerdeki kalıpları bulmada o kadar iyi olur. Ancak, aşırı takma konusunda dikkatli olun. İlk tabloda açıklandığı gibi, fazla uydurma, bir modelin eğitildiği veriler için doğru tahminlerde bulunabildiği ancak diğer verilere genelleyemediği yerdir.

Fazla uydurma, verilerinizi eğitim verileri, doğrulama verileri ve test verileri olarak ayırmanızın nedenidir. Doğrulama verileri ve test verileri genellikle birbirinin yerine kullanılır, ancak farklı amaçları vardır.



Verilerin eğitimi, test edilmesi ve doğrulanmasıyla ilgili terminolojiye aşina olun:

Terim	Tanım
Eğitim verileri	Bu veriler, modeli eğitmek ve model parametrelerine uydurmak için kullanılır. Modelin mümkün olduğunca çok örnek görmesini istediğiniz için en büyük veri oranını oluşturur.
Doğrulama verileri	Bu veriler, hiperparametreleri sığdırmak ve özellik seçimi için kullanılır. Model, eğitim sırasında bu verileri hiçbir zaman görmese de, bu verilere dayalı belirli özellikleri veya hiperparametreleri seçerek, sapma ve aşırı uyum riskini ortaya çıkarırsınız.
Test verisi	Bu veriler, ayarlanmış modellerinizi değerlendirmek ve karşılaştırmak için kullanılır. Bu veriler eğitim veya ayarlama sırasında görülmediğinden, modellerinizin görünmeyen verilere iyi genellenip genellenmediği konusunda fikir verebilir.

Çapraz doğrulama	Çapraz doğrulamanın amacı, belirli bir algoritmanın verileriniz ve kullanım durumunuz için uygun olup olmadığını değerlendirmektir. Çapraz doğrulama, hiperparametre ayarlama ve özellik seçimi için de kullanılır. Veriler, tren ve doğrulama kümelerine bölünür, ancak test verilerinin de bir tarafa yerleştirilmesi gerekir. Veri dilimlerinin her biri ile bir model oluşturursunuz. Algoritmanın son değerlendirmesi, modellerin her birinin ortalama performansıdır.
Bekletme yöntemi	Çapraz doğrulamanın en basit versiyonu, verilerinizi rastgele iki kümeye böldüğünüz hold-out yöntemidir: bir eğitim seti ve bir doğrulama seti. Bu yöntem en hızlıdır çünkü bir modelin yalnızca bir kez oluşturulmasını gerektirir. Ancak, yalnızca bir doğrulama veri seti ile, tahmin edilmesi kolay veya zor gözlemler içerme riskini taşırsınız. Bu nedenle, bu doğrulama verilerine gereğinden fazla uyduğunuzu ve test setinde kötü performans gösterdiğini fark edebilirsiniz.
K-kat çapraz doğrulama	Bu yöntem, verileri k-alt kümelere bölmeyi içerir. Daha sonra, doğrulama verileri olarak k-alt kümelerden birini kullanarak her seferinde bir modeli k kez eğitirsiniz. Eğitim verileri, doğrulama kümesinde olmayan diğer tüm gözlemlerdir. Son değerlendirmeniz, tüm k kıvrımların ortalamasıdır.
Birini dışarıda bırak çapraz doğrulama	Bu yöntem, k-kat çapraz doğrulamanın en uç versiyonudur. İçinde, k'niz N'dir (veri kümenizdeki gözlem sayısı). Bir gözlem dışında tüm verileri kullanarak bir modeli N ayrı zamanlarda eğitirsiniz ve ardından bu gözlem için tahmini ile doğruluğunu doğrularsınız. Bu algoritmanın veri kümenizle ne kadar iyi çalıştığını kapsamlı bir şekilde değerlendiriyor olsanız da, bu yöntem pahalıdır çünkü N model oluşturmanızı gerektirir.
Stratified cross-validation	Bu yöntem, k-katlı kümelerin, kategorik özelliklerde veya etikette her sınıf için benzer gözlem oranlarına sahip olmasını zorlar.

Algoritma keşfi

Keşfettiğiniz algoritmalar ile önce neyi başarmaya çalıştığınızı belirleyerek, çözüm arama kapsamını daraltabilirsiniz. Olası yöntemler arasında regresyon, sınıflandırma, kümeleme, öneriler ve anormallik tespiti için algoritmalar bulunur ancak bunlarla sınırlı değildir. Algoritma seçimini yönlendirmek için Algoritma Gezgini'ni de kullanabilirsiniz.

Regresyon: Regresyon algoritmaları, sürekli sayısal değerleri tahmin etmeye yönelik makine öğrenme teknikleridir. Denetimli öğrenme görevleridir, bu nedenle etiketli eğitim örnekleri gerektirirler.

Sınıflandırma: Sınıflandırma algoritmaları, girdi verilerinin hangi kategoriye ait olduğunu tahmin etmeye yönelik makine öğrenme teknikleridir. Denetimli öğrenme görevleridir, bu nedenle etiketli eğitim örnekleri gerektirirler.

Kümeleme: Kümeleme algoritmaları, verileri gruplardaki noktaların benzer özelliklere sahip olduğu birkaç gruba bölmek için makine öğrenimi teknikleridir. Denetimsiz öğrenme görevleridir ve etiketli eğitim örnekleri gerektirmezler.

Öneri motorları: Öneri motorları, kullanıcının bir öğeye veya ürüne olan ilgisini gösteren bir tercihi veya derecelendirmeyi tahmin etmek için oluşturulur. Bu sistemi oluşturmak için kullanılan algoritmalar, kullanıcılar, öğeler veya her ikisi arasında benzerlikler bulur.

Anomali tespiti: Anormallik tespiti, beklenen davranışa uymayan olağandışı olayları veya kalıpları belirlemek için kullanılan bir tekniktir. Tanımlanan öğelere genellikle anormallikler veya aykırı değerler denir.

Hiperparametre optimizasyonu: Parametreler ve hiperparametreler bazen birbirinin yerine kullanılsa da, aralarında ayrımlar vardır. Parametreler, algoritmanın eğitim sırasında öğrendiği özelliklerdir. Doğrusal regresyon için bu parametreler ağırlıklar ve önyargılardır. Rastgele ormanlar için, her düğümdeki değişkenler ve eşiklerdir.
Hiperparametreler, eğitimden önce ayarlanması gereken özelliklerdir. k-ortalama kümeleme için, k değerini tanımlamanız gerekir. Sinir ağları için bir örnek, öğrenme oranıdır. Hiperparametre optimizasyonu, en yüksek doğruluk ve en düşük RMSE (kök-ortalama-kare hatası) gibi performans ölçütünüzü optimize etmek için hiperparametreler için mümkün olan en iyi değerleri bulma sürecidir. Farklı değer kombinasyonları için bir model eğitirsiniz ve hangi modellerin en iyi çözümü bulduğunu değerlendirirsiniz. En iyi kombinasyonları arama yöntemleri arasında grid araması, rastgele arama, kaba-ince ve bayes optimizasyonu yer alır.

Grid arama: Grid aramasında, her bir hiperparametre için değerler belirtirsiniz ve bu değerlerin tüm kombinasyonları değerlendirilir. Örneğin, rastgele orman için hiperparametreleri değerlendirmek için, ağaç hiperparametresi sayısı için üç seçenek

sağlayabilirsiniz: 10, 20 ve 50. Her ağacın maksimum derinliği için ayrıca üç seçenek sağlarsınız: limit yok, 10 ve 20. Bu seçim sonuçları, dokuz olası kombinasyonun her biri için rastgele bir orman modelinin oluşturulmasıyla sonuçlanır: (10, limitsiz), (10, 10), (10, 20), (20, limitsiz), (20, 10), (20, 20), (50, limitsiz), (50, 10) ve (50, 20).

En iyi performansı sağlayan kombinasyon, son modeliniz için kullandığınız kombinasyondur. Bu yöntemin kullanımı basittir. Sağladığınız değerlerin en iyi kombinasyonunu bulabilir ve deneylerin her birini paralel olarak çalıştırabilirsiniz. Bununla birlikte, pek çok model üretildiği için hesaplama açısından da pahalıdır. Bir hiper parametre önemli değilse, gereksiz yere farklı olasılıkları keşfedebilirsiniz.

Rastgele arama: Rastgele aramada, her bir hiperparametre için aralıklar veya seçenekler belirlersiniz ve her birinin rastgele değerleri seçilir. Rastgele orman örneğinden devam ederek, ağaç sayısının 10 - 50 ve maksimum_derinlik için limitsiz, 10 veya 20 olmasını sağlayabilirsiniz. Bu sefer tüm permütasyonları hesaplamak yerine, sayıyı belirtebilirsiniz. çalıştırmak istediğiniz yinelemelerin sayısı. Sadece beş tane istiyorsun. (19, 20), (32 limitsiz), (40, limitsiz), (10, 20), (27, 10) gibi bir şeyi test edebilirsiniz.

Bu yöntemin kullanımı kolaydır, verimlidir ve genel performansı yalnızca birkaç hiperparametre etkilediğinde grid aramasından daha iyi performans gösterebilir. Deneylerin her birini paralel olarak çalıştırabilirsiniz. Ancak, rastgele örnekleme içerir, bu nedenle yalnızca o alanı ararsa en iyi kombinasyonu bulur.

Kaba-ince (Coarse-to-fine): Hem grid araması hem de rastgele arama için kaba-ince tekniğini de kullanabilirsiniz. Bu teknik, geniş aralıklarla veya tüm olası seçeneklerle daha geniş bir değişken yelpazesini keşfetmeyi içerir. İlk aramanızdan elde ettiğiniz sonuçları aldıktan sonra, umut verici görünen kalıpları veya bölgeleri bulmak için sonuçları keşfedersiniz. Eğer öyleyse, işlemi tekrarlayabilir, ancak aramanızı daraltabilirsiniz.

Rastgele orman örneği için, maksimum derinlik sınırı olmadığında ve ağaç hiperparametresi sayısı 10 veya 20 olduğunda sonuçların umut verici olduğunu fark edebilirsiniz. Arama işlemi tekrarlanır ancak maksimum derinlik hiperparametresini sabit tutar ve ayrıntı düzeyini artırır. ağaç seçeneklerinin sayısı, test değerleri 12, 14, 16, 18, daha iyi bir sonuç bulup bulamayacağınızı görmek için.

Bu yöntem, performans metriğini geliştirerek daha fazla optimize edilmiş hiperparametreler bulabilir. Ancak, keşfedilecek en iyi bölgeleri bulmak için sonuçların değerlendirilmesi zahmetli olabilir.

Bayes optimizasyonu: Bayes optimizasyonu, bir sonraki en iyiyi seçmek için hiperparametre kombinasyonları ile önceden elde edilen başarı bilgisini kullanır. Bu teknik,

hiperparametrelerin özellikler ve performansın hedef değişken olduğu bir model oluşturarak bir makine öğrenimi yaklaşımı kullanır. Her deneyden sonra yeni bir veri noktası eklenir ve yeni bir model oluşturulur. Benzer kombinasyonların benzer sonuçlara sahip olduğunu varsayar ve umut verici sonuçların bulunduğu bölgeleri keşfetmeye öncelik verir.

Bununla birlikte, belirsizliği büyük kazanç olasılığı olarak da dikkate alır. Büyük keşfedilmemiş alanlar varsa, o alanlara da öncelik verir. Yalnızca bir hiper parametreyi, yani ağaç sayısını alarak, algoritma önce 10'u deneyebilir ve iyi bir performans elde edebilir. Daha sonra 32'yi dener ve performans önemli ölçüde daha iyidir.

Bayes optimizasyonu, performansı tahmin etmek için ilk iki veri noktasına dayalı bir model oluşturur. Modelin yalnızca iki veri noktasıyla doğrusal olması muhtemeldir, bu nedenle seçtiği bir sonraki değer, ağaç sayısı arttıkça performansın artması beklentisiyle 40'tır. Öyle değil.

Şimdiye kadar en iyi sonucu gördüğü yerde 32 civarında bir gelişme olabileceğini öne süren başka bir model oluşturuyor. Bununla birlikte, 10 ile 32 arasında hala keşfedilmemiş büyük bir boşluk vardır ve büyük belirsizlik nedeniyle 21'i seçer. Yine, model bu yeni verilerle ince ayar yapılır ve başka bir değer seçilir.

Bu yöntem, performans metriğini geliştirerek daha fazla optimize edilmiş hiperparametreler bulabilir. Parametre sayısı yüksek olduğunda ve her deney hesaplama açısından pahalı olduğunda, optimum çözümü aramak için harcanan zamanı azaltabilir. Ancak, sonraki hiperparametre değerleri kombinasyonu önceki çalıştırmalar tarafından belirlendiği için her denemeyi paralel olarak çalıştıramazsınız. Ayrıca ayarlama gerektirir: her bir hiperparametre ve uygun bir çekirdek için bir ölçek seçme.

Toplu öğrenme (Ensemble learning): Topluluklar, daha doğru bir çözüm sağlamak için her biri veriler içinde farklı modeller bulan çeşitli makine öğrenimi modellerini birleştirir. Bu teknikler, daha fazla trend yakaladıkları için performansı artırabilir. Nihai tahmin birçok modelden bir fikir birliği olduğu için fazla takmayı da azaltabilirler.

Torbalama (Bagging): Torbalama veya önyükleme toplamaları, modelleri paralel olarak oluşturma ve son tahmin olarak tahminlerinin ortalamasını alma yöntemidir. Bu modeller aynı algoritma ile oluşturulabilir. Örneğin, rastgele orman algoritması birçok karar ağacı oluşturur. Doğrusal regresyon modeli ve destek vektör makinesi (SVM) modeli gibi farklı türde modeller de oluşturabilirsiniz.

Artırma (Boosting): Boosting, önceki modellerin başarısını değerlendirerek modelleri sırayla oluşturur. Sonraki model, mevcut modellerin kötü performans gösterdiği örnekleri tahmin

etmek için öğrenme eğilimlerine öncelik verir. Üç yaygın teknik; AdaBoost, Gradient Boosting ve XGBoosted'dir.

İstifleme (Stacking): İstifleme, modeller oluşturmayı ve çıktılarını özellikler olarak nihai bir modelde kullanmayı içerir. Örneğin, amacınız bir sınıflandırıcı oluşturmak ve bir KNN modeli ve bir Naïve Bayes modeli oluşturuyorsunuz. İkisi arasında seçim yapmak yerine, iki tahminini nihai bir lojistik regresyon modeline aktarabilirsiniz. Bu son model, iki ara modelden daha iyi sonuçlar verebilir.

4.2. Grid Arama

Makine öğrenimi modellerinin çoğu, modelin öğrenme şeklini değiştirecek şekilde ayarlanabilen parametreler içerir. Örneğin, sklearn'den gelen lojistik regresyon modeli, modelin karmaşıklığını etkileyen düzenlileştirmeyi kontrol eden bir C parametresine sahiptir. C için en iyi değeri nasıl seçeriz? En iyi değer, modeli eğitmek için kullanılan verilere bağlıdır.

Bir yöntem, farklı değerleri denemek ve ardından en iyi puanı veren değeri seçmektir. Bu teknik grid araması olarak bilinir. İki veya daha fazla parametre için değer seçmemiz gerekseydi, değer kümelerin tüm kombinasyonları değerlendirilir, böylece bir değerler gridsı oluşturulur.

Örneğe girmeden önce değiştirdiğimiz parametrenin ne işe yaradığını bilmekte fayda var. Daha yüksek C değerleri modele, eğitim verilerinin gerçek dünya bilgisine benzediğini ve eğitim verilerine daha fazla ağırlık verdiği söylenir. C'nin daha düşük değerleri ise tam tersini yapar.

Varsayılan Parametreleri Kullanma:
Önce sadece temel parametreleri kullanarak grid araması yapmadan ne tür sonuçlar üretebileceğimize bir bakalım. Başlamak için önce birlikte çalışacağımız veri setini yüklemeliyiz.

```
from sklearn import datasets
iris = datasets.load_iris()
```

Modeli oluşturmak için bir dizi bağımsız değişken X ve bir bağımlı değişken y olmalıdır.

```
X = iris['data']
y = iris['target']
```

Şimdi iris çiçeklerini sınıflandırmak için lojistik modeli yükleyeceğiz.

```
from sklearn.linear_model import LogisticRegression
```

Modelin oluşturulması, modelin bir sonuç bulmasını sağlamak için max_iter değerini daha yüksek bir değere ayarlamak gerekir. Lojistik regresyon modelinde C için varsayılan değerin 1 olduğunu unutmayın, bunu daha sonra karşılaştıracağız. Aşağıdaki örnekte, iris veri setine bakılır ve lojistik regresyonda C için değişen değerlere sahip bir model eğitmeye çalışılır.

```
logit = LogisticRegression(max_iter = 10000)
```

Modeli oluşturduktan sonra model verilere uydurulur.

```
print(logit.fit(X,y))
```

Modeli değerlendirmek için skor yöntemini çalıştırıyoruz.
print(logit.score(X,y))

Python kod örneği:
```
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
iris = datasets.load_iris()
X = iris['data']
y = iris['target']
logit = LogisticRegression(max_iter = 10000)
print(logit.fit(X,y))
print(logit.score(X,y))
```

Varsayılan ayar olan C = 1 ile 0,973 puan elde ettik. Farklı 0,973 değerlerine sahip bir grid araması uygulayarak daha iyisini yapıp yapamayacağımızı görelim.

Grid Arama Uygulaması:
Daha önceki adımların aynısını izlenir, ancak bu sefer C için bir değer aralığı belirlenir. Aranan parametreler için hangi değerlerin ayarlanacağını bilmek, alan bilgisi ve pratiğin bir kombinasyonunu alacaktır.

C için varsayılan değer 1 olduğundan, onu çevreleyen bir dizi değer ayarlanacaktır.
C = [0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2]

Daha sonra C'nin değerlerini değiştirmek için bir for döngüsü oluşturulur ve her değişiklikte model değerlendirilir.

İlk önce skoru içinde saklamak için boş bir liste oluşturacağız.
```
scores = []
```

C'nin değerlerini değiştirmek için, değerler aralığında döngüye girmeli ve her seferinde parametre güncellenmelidir.

```
for choice in C:
  logit.set_params(C=choice)
  logit.fit(X, y)
  scores.append(logit.score(X, y))
```

Bir listede saklanan puanlarla, en iyi C seçiminin ne olduğunu değerlendirilebilir.
print(scores)

Python kod örneği:

```
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
iris = datasets.load_iris()
X = iris['data']
y = iris['target']
logit = LogisticRegression(max_iter = 10000)
C = [0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2]
scores = []
for choice in C:
  logit.set_params(C=choice)
  logit.fit(X, y)
  scores.append(logit.score(X, y))
print(scores)
```

Sonuçların açıklanması:
C'nin düşük değerlerinin temel parametre 1'den daha kötü performans gösterdiğini görebiliriz. Ancak, C değerini 1,75'e yükselttikçe modelin doğruluğu arttı. C'yi bu miktarın üzerine çıkarmanın model doğruluğunu artırmaya yardımcı olmadığı görülüyor.

Denetimli öğrenmenin amacı, yeni veriler üzerinde iyi performans gösteren bir model oluşturmaktır. Yeni verileriniz varsa, modelinizin bu veriler üzerinde nasıl performans gösterdiğini görmek iyi bir fikirdir. Sorun şu ki, yeni verileriniz olmayabilir, ancak bu deneyimi veri kümesinde eğitim ve test bölme gibi bir prosedürle simüle edebilirsiniz.

En İyi Uygulamalar Üzerine Not:
Lojistik regresyon modelimizi, onu eğitmek için kullanılan aynı verileri kullanarak puanladık. Model bu verilere çok yakınsa, görünmeyen verileri tahmin etmede iyi olmayabilir. Bu istatistiksel hata, aşırı uydurma olarak bilinir.

Eğitim verilerinin puanları tarafından yanıltılmamak için, verilerimizin bir kısmını bir kenara ayırabilir ve özellikle modeli test etmek amacıyla kullanabiliriz. Yanlış yönlendirilmeyi ve fazla uydurmayı önlemek için eğitim/test ayırma işlevinin iyi bilinmesi gerekmektedir.

4.3. AUC - ROC Curve

Sınıflandırmada birçok farklı değerlendirme ölçütü vardır. En popüler olanı, modelin ne sıklıkla doğru olduğunu ölçen doğruluktur. Bu harika bir ölçümdür çünkü anlaşılması kolaydır ve en doğru tahminleri almak genellikle istenir. Başka bir değerlendirme metriği kullanmayı düşünebileceğiniz bazı durumlar vardır.

Diğer bir yaygın metrik, alıcı çalışma karakteristiği (ROC: receiver operating characteristic) eğrisinin altındaki alan olan AUC'dir. Alıcı işletim karakteristik eğrisi, farklı sınıflandırma eşiklerinde yanlış pozitif (FP: false positive) oranına karşı gerçek pozitif (TP: true positive) oranı gösterir. Eşikler, ikili sınıflandırmada iki sınıfı ayıran farklı olasılık kesimleridir. Bir modelin sınıfları ne kadar iyi ayırdığını bize söylemek için olasılığı kullanır.

Imbalanced Data

Verilerimizin çoğunun tek bir değerde olduğu dengesiz bir veri setimiz olduğunu varsayalım. Çoğunluk sınıfını tahmin ederek model için yüksek doğruluk elde edebiliriz.

Kod örneği:

```
import numpy as np
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, roc_curve

n = 10000
ratio = .95
n_0 = int((1-ratio) * n)
n_1 = int(ratio * n)

y = np.array([0] * n_0 + [1] * n_1)
# below are the probabilities obtained from a hypothetical model that always predicts the majority class
# probability of predicting class 1 is going to be 100%
y_proba = np.array([1]*n)
y_pred = y_proba > .5

print(f'accuracy score: {accuracy_score(y, y_pred)}')
cf_mat = confusion_matrix(y, y_pred)
print('Confusion matrix')
print(cf_mat)
print(f'class 0 accuracy: {cf_mat[0][0]/n_0}')
print(f'class 1 accuracy: {cf_mat[1][1]/n_1}')
```

Çok yüksek bir doğruluk elde etmemize rağmen, model veriler hakkında hiçbir bilgi sağlamadı, bu yüzden kullanışlı değil. Sınıf 1'i zamanın %100'ünde doğru bir şekilde tahmin ederken, sınıf 0'ı zamanın %0'ında yanlış tahmin ederiz. Doğruluk pahasına, iki sınıfı bir şekilde ayırabilen bir modele sahip olmak daha iyi olabilir.

Kod örneği:

```
# below are the probabilities obtained from a hypothetical model that doesn't always predict the mode
y_proba_2 = np.array(
    np.random.uniform(0, .7, n_0).tolist() +
    np.random.uniform(.3, 1, n_1).tolist()
)
y_pred_2 = y_proba_2 > .5

print(f'accuracy score: {accuracy_score(y, y_pred_2)}')
cf_mat = confusion_matrix(y, y_pred_2)
print('Confusion matrix')
print(cf_mat)
print(f'class 0 accuracy: {cf_mat[0][0]/n_0}')
print(f'class 1 accuracy: {cf_mat[1][1]/n_1}')
```

İkinci tahmin seti için birincisi kadar yüksek bir doğruluk puanımız yok ama her sınıf için doğruluk daha dengeli. Doğruluğu bir değerlendirme metriği olarak kullanarak, bize veriler hakkında hiçbir şey söylemese de ilk modeli ikinciden daha yüksek değerlendiririz.

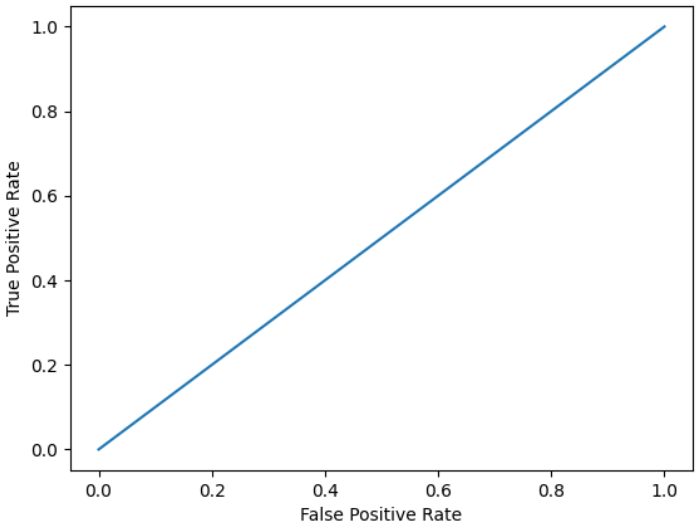
Bu gibi durumlarda, AUC gibi başka bir değerlendirme metriğinin kullanılması tercih edilir.

```
import matplotlib.pyplot as plt

def plot_roc_curve(true_y, y_prob):
    """
    plots the roc curve based of the probabilities
    """

    fpr, tpr, thresholds = roc_curve(true_y, y_prob)
    plt.plot(fpr, tpr)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
```

Example: Model 1

plot_roc_curve(y, y_proba)

print(f'model 1 AUC score: {roc_auc_score(y, y_proba)}')

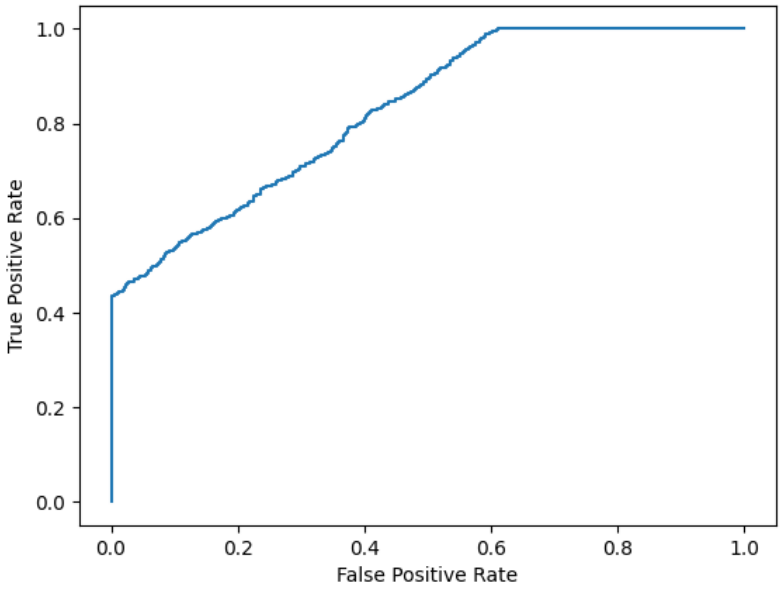


model 1 AUC score: 0.5

Example: Model 2

plot_roc_curve(y, y_proba_2)

print(f'model 2 AUC score: {roc_auc_score(y, y_proba_2)}')



model 2 AUC score: 0.8270551578947367

Yaklaşık 0,5'lik bir AUC puanı, modelin iki sınıf arasında bir ayrım yapamadığı ve eğrinin 1 eğimli bir çizgi gibi görüneceği anlamına gelir. 1'e yakın bir AUC puanı, modelin, iki sınıfı ayırın ve eğri grafiğin sol üst köşesine yaklaşacaktır.

Olasılıklar

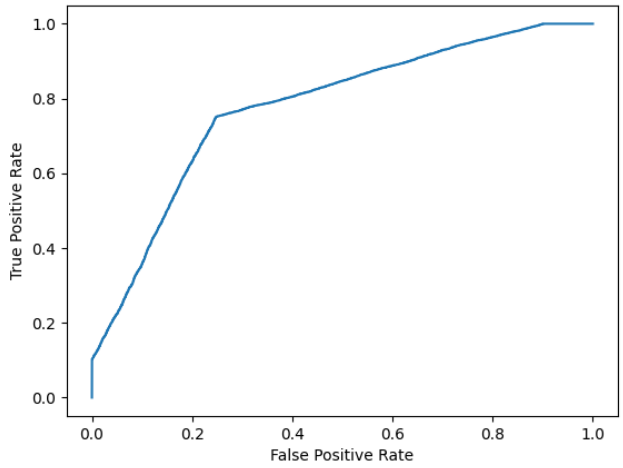
AUC, sınıf tahminlerinin olasılıklarını kullanan bir metrik olduğundan, benzer doğruluklara sahip olsalar bile, daha yüksek AUC puanına sahip bir modelde, daha düşük puana sahip bir modelden daha emin olabiliriz. Aşağıdaki verilerde, varsayımsal modellerden iki olasılık grubumuz var. Birincisi, iki sınıfı tahmin ederken "güvenli" olmayan olasılıklara sahiptir (olasılıklar .5'e yakındır). İkincisi, iki sınıfı tahmin ederken daha "güvenli" olan olasılıklara sahiptir (olasılıklar 0 veya 1'in uç noktalarına yakındır).

Kod örneği:

```
import numpy as np

n = 10000
y = np.array([0] * n + [1] * n)
#
y_prob_1 = np.array(
   np.random.uniform(.25, .5, n//2).tolist() +
   np.random.uniform(.3, .7, n).tolist() +
   np.random.uniform(.5, .75, n//2).tolist()
)
y_prob_2 = np.array(
   np.random.uniform(0, .4, n//2).tolist() +
   np.random.uniform(.3, .7, n).tolist() +
   np.random.uniform(.6, 1, n//2).tolist()
)

print(f'model 1 accuracy score: {accuracy_score(y, y_prob_1>.5)}')
print(f'model 2 accuracy score: {accuracy_score(y, y_prob_2>.5)}')

print(f'model 1 AUC score: {roc_auc_score(y, y_prob_1)}')
print(f'model 2 AUC score: {roc_auc_score(y, y_prob_2)}')
```

Kod örneği: Plot model 1

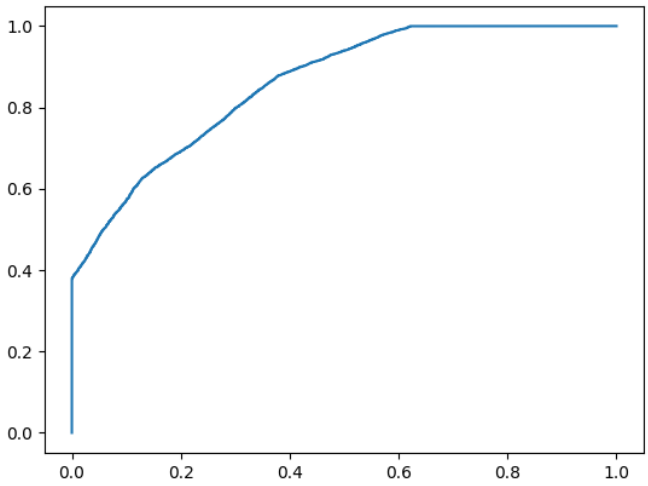
```
plot_roc_curve(y, y_prob_1)
```



Kod örneği: Plot model 2

fpr, tpr, thresholds = roc_curve(y, y_prob_2)

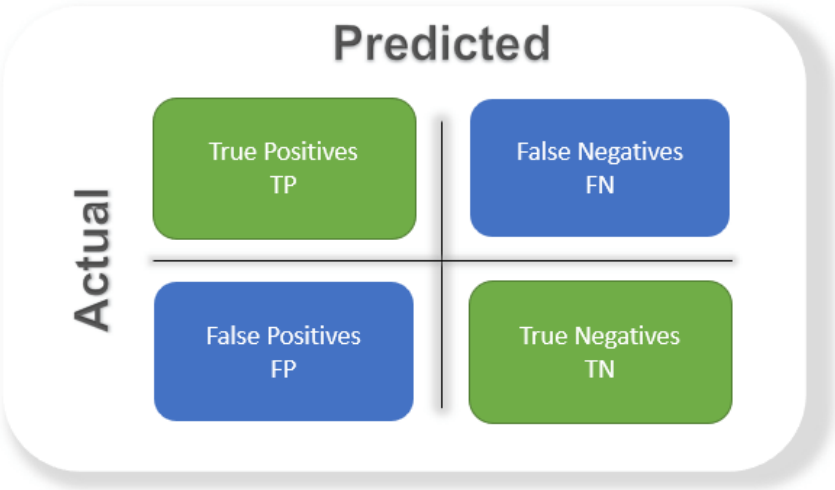
plt.plot(fpr, tpr)



İki modelin doğrulukları benzer olsa da, AUC puanı daha yüksek olan model, tahmin edilen olasılığı hesaba kattığı için daha güvenilir olacaktır. Gelecekteki verileri tahmin ederken size daha yüksek doğruluk vermesi daha olasıdır.

4.4. Karışıklık matrisi

Karışıklık matrisi, diğer sınıflandırma ölçütleri için temel oluşturur. Modelin performansını tam olarak tanımlayan bir matristir. Bir karışıklık matrisi, her sınıfın doğru ve yanlış sınıflandırmalarının derinlemesine bir dökümünü verir.



Gerçek pozitifler – pozitif tahminlerin aslında pozitif olduğu bir senaryo.
Gerçek olumsuzlar – olumsuz tahminler aslında olumsuzdur.
Yanlış pozitifler – pozitif tahminler aslında negatiftir.
Yanlış negatifler – negatif tahminlerin aslında pozitif olduğu bir senaryo.

Yukarıdaki dört terimin tanımından çıkarılacak sonuç, gerçek pozitifleri ve gerçek negatifleri çoğaltmanın önemli olduğudur. Yanlış pozitifler ve yanlış negatifler, gerçek dünya uygulamalarında maliyetli olabilecek yanlış sınıflandırmayı temsil eder. Bir modelin tıbbi dağıtımında yanlış teşhis örneklerini düşünün.

Bir model, sağlıklı bir kişinin kanser olduğunu yanlış tahmin edebilir. Ayrıca, aslında kanserli birini kansersiz olarak sınıflandırabilir. Her iki sonuç da hastaların teşhis konulduktan (veya yanlış teşhis konulduktan sonra) sağlıkları, tedavi planları ve masraflar açısından hoş olmayan sonuçlar doğuracaktır. Bu nedenle, yanlış negatifleri ve yanlış pozitifleri en aza indirmek önemlidir.

Görüntüdeki yeşil şekiller, modelin doğru tahmini yaptığı zamanı temsil eder. Mavi olanlar, modelin yanlış tahminlerde bulunduğu senaryoları temsil ediyor. Matrisin satırları gerçek sınıfları, sütunlar ise tahmin edilen sınıfları temsil eder.

Doğruluğu karışıklık matrisinden hesaplayabiliriz. Doğruluk, "doğru" diyagonaldeki değerlerin ortalaması alınarak verilir.

Doğruluk = (Gerçek Pozitif + Gerçek Negatif) / Toplam Örnek
Bu şu anlama gelir: Doğruluk = Toplam Doğru Tahmin Sayısı / Toplam Sayı

Gözlemler
Karışıklık matrisi, yukarıda bahsedilen sınıflandırmanın dört olası sonucunu görselleştirdiği için, doğruluğun yanı sıra, kesinlik, hatırlama (Geri Çağırma) ve nihayetinde F-skoru hakkında fikir sahibiyiz. Matristen kolayca hesaplanabilirler. Kesinlik, geri çağırma ve F-skoru aşağıdaki bölümde tanımlanmıştır.

F-puanı
F puanı, puanı belirlemek için bir testin hem kesinliğini hem de geri çağrılmasını içeren bir ölçümdür. Bu gönderi, onu hatırlama ve kesinliğin harmonik ortalaması olarak tanımlar.

F-skoru, F-ölçü veya F1 skoru olarak da bilinir.

Kesinlik, bir sınıflandırıcı tarafından tahmin edilen toplam pozitif sonuçlara bölünen gerçek pozitiflerin sayısını ifade eder. Basitçe söylemek gerekirse, kesinlik, tüm olumlu tahminlerin gerçekte ne kadarının doğru olduğunu anlamayı amaçlar.

Kesinlik = Gerçek Pozitifler / (Doğru Pozitifler + Yanlış Pozitifler)

Öte yandan hatırlama (Geri çağırma), pozitif olarak tahmin edilmesi gereken tüm örneklere bölünen gerçek pozitiflerin sayısıdır. Hatırlama, gerçek pozitif tahminlerin ne kadarının doğru olarak tanımlandığını algılama amacına sahiptir.

Geri Çağırma = Doğru Pozitifler / (Doğru Pozitifler + Yanlış Negatifler)

Karışıklı matrisi, modeldeki hataların nerede yapıldığını değerlendirmek için sınıflandırma problemlerinde kullanılan bir tablodur.

Satırlar, sonuçların olması gereken gerçek sınıfları temsil eder. Sütunlar yaptığımız tahminleri temsil eder. Bu tabloyu kullanarak hangi tahminlerin yanlış olduğunu görmek kolaydır.

Örnek: Bir Karışıklık Matrisi Oluşturma
Karışıklık matrisleri, lojistik bir regresyondan yapılan tahminlerle oluşturulabilir.

Şimdilik NumPy kullanılarak gerçek ve tahmin edilen değerler üretilir.
import numpy

Daha sonra "gerçek" ve "tahmin edilen" değerler için sayıların üretilmesi gerekecektir.
actual = numpy.random.binomial(1, 0.9, size = 1000)
predicted = numpy.random.binomial(1, 0.9, size = 1000)

Karışıklık matrisini oluşturmak için sklearn modülünden metriklerin içe aktarılması gerekiyor.
from sklearn import metrics.

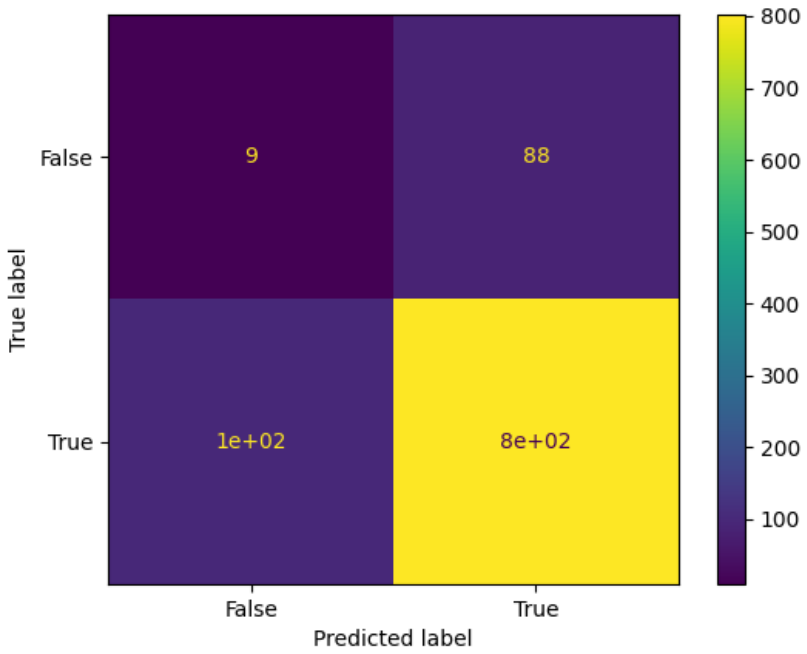
Metrikler içe aktarıldığında, gerçek ve tahmin edilen değerler üzerinde karışıklık matrisi işlevi kullanılabilir.
confusion_matrix = metrics.confusion_matrix(actual, predicted)

Daha yorumlanabilir bir görsel görüntü oluşturmak için tabloyu bir karışıklık matrisi görüntüsüne dönüştürilmesi gerekiyor.
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])

Ekranı görselleştirmek, pyplot'un matplotlib'den içe aktarılması gerektirr.
import matplotlib.pyplot as plt

Son olarak grafiği görüntülemek için pyplot'tan plot() ve show() fonksiyonları kullanılır.
cm_display.plot()
plt.show()

Tüm örneğin çalıştırılması:
```
import matplotlib.pyplot as plt
import numpy
from sklearn import metrics
actual = numpy.random.binomial(1,.9,size = 1000)
predicted = numpy.random.binomial(1,.9,size = 1000)
confusion_matrix = metrics.confusion_matrix(actual, predicted)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =
confusion_matrix, display_labels = [False, True])
cm_display.plot()
plt.show()
```



Sonuçların Açıklanması: Oluşturulan Karışıklık Matrisi dört farklı kadrana sahiptir.

- True Negative (Top-Left Quadrant)
- False Negative (Top-Right Quadrant)
- False Positive (Bottom-Left Quadrant)
- True Positive (Bottom-Right Quadrant)

Doğru, değerlerin doğru bir şekilde tahmin edildiği anlamına gelir; Yanlış, bir hata veya yanlış tahmin olduğu anlamına gelir.
Artık bir Karışıklık Matrisi yaptığımıza göre, modelin kalitesini ölçmek için farklı ölçüler hesaplayabiliriz. İlk olarak, Doğruluk'a bakalım.

Oluşturulan Metrikler
Matris, sınıflandırma modelini değerlendirmemize yardımcı olan birçok yararlı ölçüm sağlar. Farklı ölçüler şunları içerir: Doğruluk, Kesinlik, Duyarlılık (Geri Çağırma), Özgüllük ve aşağıda açıklanan F-skoru.

Kesinlik (Accuracy): Doğruluk, modelin ne sıklıkla doğru olduğunu ölçer.
Nasıl hesaplanır: (Doğru Olumlu + Doğru Olumsuz) / Toplam Tahminler
(True Positive + True Negative) / Total Predictions

Python Örneği: Accuracy = metrics.accuracy_score(actual, predicted)

Kesinlik (Precision): Öngörülen pozitiflerin yüzde kaçı gerçekten pozitiftir?
Nasıl Hesaplanır: True Positive / (True Positive + False Positive)
Kesinlik, doğru tahmin edilen olumsuz durumları değerlendirmez.
Pyhon kod örneği: Precision = metrics.precision_score(actual, predicted)

Sensitivity (Recall)
Of all the positive cases, what percentage are predicted positive?
Sensitivity (sometimes called Recall) measures how good the model is at predicting positives.
This means it looks at true positives and false negatives (which are positives that have been incorrectly predicted as negative).

Hassasiyet - Duyarlılık(Geri Çağırma - Hatırlama) (Sensitivity - Recall)
Tüm pozitif vakaların yüzde kaçının pozitif olduğu tahmin ediliyor?
Duyarlılık (bazen Geri Çağırma olarak da adlandırılır), modelin pozitifleri tahmin etmede ne kadar iyi olduğunu ölçer. Bu, gerçek pozitiflere ve yanlış negatiflere (yanlış bir şekilde negatif olarak tahmin edilen pozitiflere) baktığı anlamına gelir.

Nasıl Hesaplanır: True Positive / (True Positive + False Negative)
Duyarlılık, modelin bir şeyin olumlu olduğunu ne kadar iyi tahmin ettiğini anlamada iyidir.
Python kod örneği: *Sensitivity_recall = metrics.recall_score(actual, predicted)*

Özgüllük (Specificity)
Model, olumsuz sonuçları tahmin etmede ne kadar iyi? Özgüllük duyarlılığa benzer, ancak olumsuz sonuçlar açısından bakar.
Nasıl hesaplanır: True Negative / (True Negative + False Positive)
Geri Çağırmanın tam tersi olduğu için, ters konum etiketini alarak "recall_score function" işlevini kullanırız.
Python kod örneği: *Specificity = metrics.recall_score(actual, predicted, pos_label=0)*

F-score
F-score, kesinlik ve duyarlılığın "harmonik ortalamasıdır". Hem yanlış pozitif hem de yanlış negatif durumları dikkate alır ve dengesiz veri kümeleri için iyidir.
Nasıl hesaplanır : 2 * ((Precision * Sensitivity) / (Precision + Sensitivity))
Bu puan True Negative değerlerini dikkate almaz.
Python kod örneği: *F1_score = metrics.f1_score(actual, predicted)*

Bütün hesaplamalar tek bir Python komutunda:
```
#metrics
print({"Accuracy":Accuracy,"Precision":Precision,"Sensitivity_recall":Sensitivity_recall,"Specificity":Specificity,"F1_score":F1_score})
```

4.5. Çapraz Doğrulama tekniği

Çapraz doğrulama, bir eğitim veri kümesinin ve bağımsız bir veri kümesinin kullanımını içerir. Bu iki küme, orijinal veri kümesinin bölümlenmesinden kaynaklanır. Kümeler bir algoritmayı değerlendirmek için kullanılır.

İlk olarak, veri kümesi eşit büyüklükte örnek gruplarına bölünür. Bu gruplara katmanlar denir. Değerlendirilecek model, biri hariç tüm gruplarda eğitilir. Eğitimden sonra hariç tutulan üzerinde model test edilir. Bu işlem daha sonra kat sayısına bağlı olarak tekrar tekrar yapılır. Tekrarlamanın nedeni, her katın dışlanması ve test seti olarak hareket etmesidir. Son olarak, algoritmanın bir problem üzerinde ne kadar etkili olduğuna dair bir tahmin elde etmek için tüm katmanlardaki ortalama performans ölçülür.

Popüler bir çapraz doğrulama tekniği, k-katlı çapraz doğrulamadır. Yukarıda açıklanan aynı adımları kullanır. k, (kullanıcı tarafından belirlenen bir sayıdır), kat sayısı anlamına gelir. k değeri, veri kümesinin boyutuna göre değişebilir, ancak örnek olarak, 4 katlı çapraz doğrulama senaryosunu kullanalım.

Model dört kez eğitilecek ve test edilecektir. Diyelim ki ilk turda kıvrımlar 1,2 ve 3. katmanlarda olacak. Test 4. katmanda yapılacak. İkinci tur için 1,2 ve 4. Katmanlarında antrenman yapabilir ve 3. katmanda test yapabilir. 1,3 ve 4 numaralı katmanlarda antrenman yapabilir ve 2. katmanda test edebilir.

Son turda 2. ve 4. katmandada test yapılacak ve 1. katmandada test edilecek. Antrenman ve test verileri arasındaki değişim bu yöntemi çok etkili kılıyor. Ancak, holdout tekniğiyle karşılaştırıldığında, çapraz doğrulamanın çalışması daha uzun sürer ve daha fazla hesaplama kaynağı kullanır.

Modelleri ayarlarken, görünmeyen verilerde genel model performansını artırmayı hedefliyoruz. Hiperparametre ayarı, test setlerinde çok daha iyi performansa yol açabilir. Bununla birlikte, parametreleri test kümesine optimize etmek, modelin görünmeyen veriler üzerinde daha kötü performans göstermesine neden olan bilgi sızıntısına neden olabilir. Bunu düzeltmek için çapraz doğrulama yapabiliriz.
CV'yi daha iyi anlamak için iris veri seti üzerinde farklı yöntemler uygulayacağız. Önce verileri yükleyelim ve ayıralım.

```
from sklearn import datasets

X, y = datasets.load_iris(return_X_y=True)
```

Çapraz doğrulama için birçok yöntem vardır, k-kat çapraz doğrulamaya bakarak başlayacağız.

K-Fold
Modelde kullanılan eğitim verileri, modeli doğrulamak için kullanılmak üzere k adet daha küçük kümeye bölünür. Model daha sonra eğitim setinin k-1 katları üzerinde eğitilir. Kalan kat daha sonra modeli değerlendirmek için bir doğrulama seti olarak kullanılır.
Farklı iris çiçek türlerini sınıflandırmaya çalışacağımız için bir sınıflandırıcı model almamız gerekecek, bu alıştırma için bir DecisionTreeClassifier kullanacağız. Ayrıca sklearn'den CV modüllerini içe aktarmamız gerekecek.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold, cross_val_score
```

Yüklenen verilerle artık değerlendirme için bir model oluşturabilir ve sığdırabiliriz.
```
clf = DecisionTreeClassifier(random_state=42)
```

Şimdi modelimizi değerlendirelim ve her k-katlamada nasıl performans gösterdiğini görelim.
```
k_folds = KFold(n_splits = 5)

scores = cross_val_score(clf, X, y, cv = k_folds)
```
Tüm kıvrımlar için puanların ortalamasını alarak CV'nin genel olarak nasıl performans gösterdiğini görmek de iyi bir uygulamadır.

Kod örneği: Run k-fold CV:
```
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold, cross_val_score

X, y = datasets.load_iris(return_X_y=True)

clf = DecisionTreeClassifier(random_state=42)

k_folds = KFold(n_splits = 5)

scores = cross_val_score(clf, X, y, cv = k_folds)

print("Cross Validation Scores: ", scores)
print("Average CV Score: ", scores.mean())
print("Number of CV Scores used in Average: ", len(scores))
```

Tabakalı K-Fold (K-Katlama):
Sınıfların dengesiz olduğu durumlarda, hem tren hem de doğrulama setlerindeki dengesizliği hesaba katacak bir yola ihtiyacımız var. Bunu yapmak için hedef sınıfları katmanlaştırabiliriz, yani her iki küme de tüm sınıfların eşit bir oranına sahip olacaktır.

Kod örneği:

```
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import StratifiedKFold, cross_val_score

X, y = datasets.load_iris(return_X_y=True)

clf = DecisionTreeClassifier(random_state=42)

sk_folds = StratifiedKFold(n_splits = 5)

scores = cross_val_score(clf, X, y, cv = sk_folds)

print("Cross Validation Scores: ", scores)
print("Average CV Score: ", scores.mean())
print("Number of CV Scores used in Average: ", len(scores))
```

Kat sayısı aynı da, tabakalı sınıflar olsa emin olunmasından ortalama CV temel aynık-katından artar.
Leave-One-Out (LOO)

Eğitim veri setindeki k-kat LeaveOneOut gibi bölmelerin sayısını seçmek yerine, doğrulamak için 1 gözlem ve eğitmek için n-1 gözlem kullanın. Bu yöntem, kapsamlı bir tekniktir.
Kod örneği: Run LOO CV:

```
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import LeaveOneOut, cross_val_score

X, y = datasets.load_iris(return_X_y=True)

clf = DecisionTreeClassifier(random_state=42)

loo = LeaveOneOut()

scores = cross_val_score(clf, X, y, cv = loo)
```

```
print("Cross Validation Scores: ", scores)
print("Average CV Score: ", scores.mean())
print("Number of CV Scores used in Average: ", len(scores))
```

Yapılan çapraz doğrulama puanlarının sayısının veri setindeki gözlem sayısına eşit olduğunu görebiliriz. Bu durumda iris veri setinde 150 gözlem vardır.
Ortalama CV puanı %94'tür.

Leave-P-Out (LPO)
P-Out, doğrulama setimizde kullanmak üzere p sayısını seçebilmemiz açısından, Birini Bırakma (Leave-One-Out (fikrinden nüanslı bir farktır.

Kod örneği: Run LPO CV

```
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import LeavePOut, cross_val_score

X, y = datasets.load_iris(return_X_y=True)

clf = DecisionTreeClassifier(random_state=42)

lpo = LeavePOut(p=2)

scores = cross_val_score(clf, X, y, cv = lpo)

print("Cross Validation Scores: ", scores)
print("Average CV Score: ", scores.mean())
print("Number of CV Scores used in Average: ", len(scores))
```

Gördüğümüz gibi, bu kapsamlı bir yöntem, bir p = 2 olsa bile, Birini Bırakma'dan çok daha fazla puan hesaplıyoruz, ancak kabaca aynı ortalama CV puanına ulaşıyor.

Karışık Böl (Shuffle Split)

KFold'dan farklı olarak ShuffleSplit, trend veya doğrulama setlerinde kullanılmamak üzere verilerin bir yüzdesini dışarıda bırakır. Bunu yapmak için tren ve test boyutlarının yanı sıra bölmelerin sayısına karar vermeliyiz.

Kod örneği: Run Shuffle Split CV

```
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.model_selection import ShuffleSplit, cross_val_score

X, y = datasets.load_iris(return_X_y=True)

clf = DecisionTreeClassifier(random_state=42)

ss = ShuffleSplit(train_size=0.6, test_size=0.3, n_splits = 5)

scores = cross_val_score(clf, X, y, cv = ss)

print("Cross Validation Scores: ", scores)
print("Average CV Score: ", scores.mean())
print("Number of CV Scores used in Average: ", len(scores))
```

Bitiş Notları

Bunlar, modellere uygulanabilecek CV yöntemlerinden sadece birkaçıdır. Çoğu modelin kendi sınıfına sahip olduğu daha birçok çapraz doğrulama sınıfı vardır. Daha fazla CV seçeneği için sklearns çapraz doğrulamasına göz atın.

4.6. Özellik (Öznitelik) Vektörleri

Makine öğrenmesinin omurgasını oluşturan disiplinler:

- İstatistiksel hesaplama ile yakından ilgilidir.
- Matematiksel optimizasyon çalışması, makine öğrenmesi alanına yöntemler, teori ve uygulama alanları sağlar.
- Veri madenciliği, denetimsiz öğrenim yoluyla keşifsel veri analizine odaklanan ilgili bir çalışma alanıdır.
- Uygulamalı Matematik
- Bilgisayar sistemleri ve yazılımlar

Makine öğrenmesi (ML), yapay zekanın bir alt kümesi olarak görülür. Makine öğrenmesi algoritmalardan oluşur. Açık bir şekilde programlanmadan tahminler veya kararlar vermek için "öğrenme verileri" olarak bilinen örnek verilere dayalı bir matematiksel model oluşturulmasıdır. Makine öğrenmesi algoritmaları, gerekli görevleri yerine getirmek için geleneksel algoritmalar geliştirmenin zor veya mümkün olmadığı filtreleme ve katsayıları değişken fonksiyonların çok çeşitli uygulamalarında kullanılır.

Veri madenciliği, denetimsiz öğrenim yoluyla keşifsel veri analizine odaklanan ilgili bir çalışma alanıdır. Makine öğrenmesi, bilgisayarları açıkça programlanmaksızın görevleri nasıl gerçekleştirebileceklerini keşfetmeyi içerir. Belirli görevleri yerine getirmeleri için sağlanan verilerden öğrenen algoritmaları içerir. Makine öğrenmesi tam olarak tatmin edici bir algoritmanın bulunmadığı görevleri yerine getirmelerini öğrenmek için çeşitli yaklaşımlar kullanır. Çok sayıda potansiyel cevap bulunduğu durumlarda, doğru cevapların bazılarını geçerli olarak etiketlemektir.

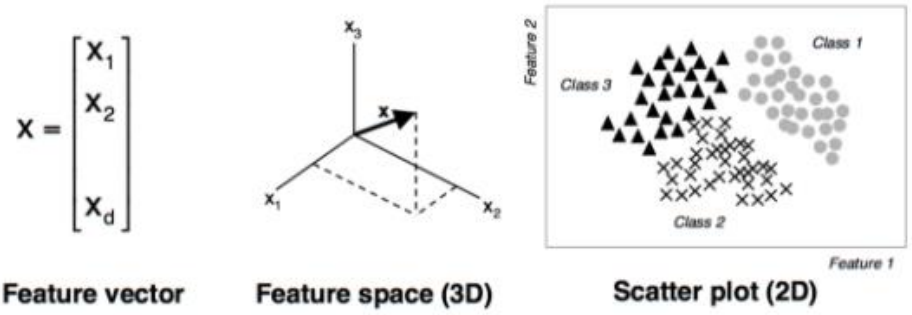
Makine öğrenimi karmaşık olarak algılanan problemlerin analizi için mükemmel bir yöntemdir. Genel anlamda makine öğrenimin görevi sorunları tespit etmek, gelecekte olabilecek senaryoları önceden tahmin etmek ve büyük veri yığınının genel olarak kalıplarını keşfedebilme yeteneğine sahiptir ancak her türlü teknolojide olduğu gibi makine öğrenimi de tamamen mükemmel değildir. Hata yapma olasılığını arttıran nedenler; verileri elde etmenin zorluğu gerekli hesaplama gücü getirilen karmaşıklık, bazı algoritmaların uzun süren eğitim süreleri vb. Bu nedenler makine öğrenimine geçişin engeli değil erteleyicisidir.

Makine öğrenmesinde, özellik vektörleri, bir nesnenin özellikler adı verilen sayısal veya sembolik özelliklerini matematiksel olarak, kolayca analiz edilebilir bir şekilde temsil etmek için kullanılır. Makine öğreniminin ve kalıp işlemenin birçok farklı alanı için önemlidirler. Makine öğrenimi algoritmaları, algoritmaların işleme ve istatistiksel analiz yapabilmesi için tipik olarak nesnelerin sayısal bir temsilini gerektirir. Özellik vektörleri, doğrusal regresyon gibi istatistiksel prosedürlerde kullanılan açıklayıcı değişkenlerin vektörlerinin eşdeğeridir.

Aşina olabileceğiniz bir özellik vektörüne örnek RGB (kırmızı-yeşil-mavi) renk açıklamalarıdır. Bir renk, içinde ne kadar kırmızı, mavi ve yeşil olduğu ile tanımlanabilir. Bunun için bir özellik vektörü, renk = [R, G, B] olacaktır.

Bir vektör, tek sütunlu ancak çok satırlı bir matris gibi, genellikle uzamsal olarak temsil edilebilen bir sayı dizisidir. Bir özellik, bir nesnenin bir yönünün sayısal veya sembolik bir özelliğidir. Özellik vektörü, bir nesne hakkında birden çok öğe içeren bir vektördür. Nesneler için özellik vektörlerini bir araya getirmek bir özellik alanı oluşturabilir.

Özellikler, bir bütün olarak, yalnızca bir piksel veya bütün bir görüntüyü temsil edebilir. Ayrıntı düzeyi, bir kişinin nesne hakkında ne öğrenmeye veya temsil etmeye çalıştığına bağlıdır. 3 boyutlu bir şekli, yüksekliğini, genişliğini, derinliğini vb. Gösteren bir özellik vektörüyle tanımlayabilirsiniz.



Özellik vektörleri, birçok analiz türüne yardımcı olmak için nesneleri sayısal bir şekilde temsil etmenin etkinliği ve pratikliği nedeniyle makine öğreniminde yaygın olarak kullanılmaktadır. Analiz için iyidirler çünkü öznitelik vektörlerini karşılaştırmak için birçok teknik vardır. İki nesnenin özellik vektörlerini karşılaştırmanın basit bir yolu Öklid mesafesini almaktır.

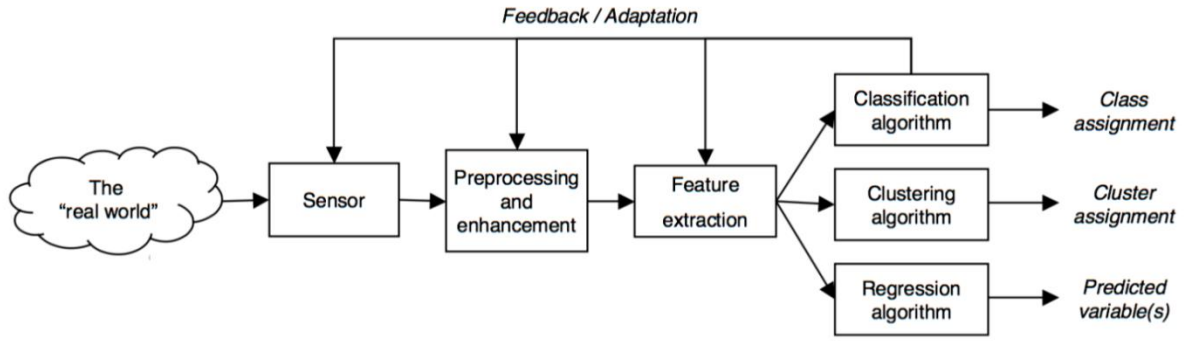
Görüntü işlemede, özellikler gradyan büyüklüğü, renk, gri tonlama yoğunluğu, kenarlar, alanlar ve daha fazlası olabilir. Özellik vektörleri, bir görüntü hakkındaki özniteliklerin, listelenen örnekler gibi, öznitelik vektörlerine yerleştirildikten sonra sayısal olarak karşılaştırılabilmesinin uygun yolu nedeniyle özellikle görüntü işlemedeki analizler için popülerdir.

Konuşma tanımada özellikler ses uzunlukları, gürültü seviyesi, gürültü oranları ve daha fazlası olabilir.

İstenmeyen e-postayla mücadele girişimlerinde özellikler bol miktarda bulunur. IP konumu, metin yapısı, belirli kelimelerin sıklığı veya belirli e-posta başlıkları olabilir.

Özellik vektörleri, makine öğrenmesinde sınıflandırma problemlerinde, yapay sinir ağlarında ve k en yakın komşu algoritmalarında kullanılır.

Örüntü tanıma süreçlerinde, özellik vektörleri, veri toplama ve verileri anlamlandırma arasında kullanılan araçlardır:



4.7. Parametreler ve Hiperparametreler

Yeni bir şey öğrenmeye başladığınızda, uğraştığınız şeylerden biri, içine girdiğiniz alanın dilidir. Bir alanda kullanılan terimleri (ve bazı durumlarda sembolleri ve kısaltmaları) açıkça anlamak, konunun kendisini anlamanın ilk ve en temel adımıdır. Makine Öğrenimi'ne başladığımızda, parametreler ve hiperparametreler kavramı çok karıştırıcıdır.

ML/DL'de bir model, model parametreleri tarafından tanımlanır veya temsil edilir. Bununla birlikte, bir modeli eğitme süreci, girdi özelliklerini (bağımsız değişkenler) etiketlere veya hedeflere (bağımlı değişken) doğru şekilde eşleyen optimal parametreleri öğrenmek için öğrenme algoritmasının kullanacağı optimal hiperparametreleri seçmeyi içerir.

Hiperparametreler

Hiperparametreler, değerleri öğrenme sürecini kontrol eden ve bir öğrenme algoritmasının öğrenmesiyle sonuçlanan model parametrelerinin değerlerini belirleyen parametrelerdir. 'Hyper_' öneki, bunların öğrenme sürecini ve bundan kaynaklanan model parametrelerini kontrol eden 'üst düzey' parametreler olduğunu gösterir.

Bir öğrenme modelini tasarlayan bir makine öğrenimi mühendisi olarak, daha modelin eğitimi başlamadan önce öğrenme algoritmanızın kullanacağı hiperparametre değerlerini seçer ve ayarlarsınız. Bu bağlamda, model öğrenme/eğitim sırasında değerlerini değiştiremediği için hiperparametrelerin modelin dışında olduğu söylenir.

Hiperparametreler, öğrenme algoritması tarafından öğrenirken kullanılır, ancak bunlar ortaya çıkan modelin bir parçası değildir. Öğrenme sürecinin sonunda, model olarak adlandırdığımız etkin bir şekilde eğitilmiş model parametrelerine sahibiz. Eğitim sırasında kullanılan hiperparametreler bu modelin parçası değildir. Örneğin, modelin kendisinden bir modeli eğitmek için hangi hiperparametre değerlerinin kullanıldığını bilemeyiz, sadece öğrenilen model parametrelerini biliyoruz.

Temel olarak, makine öğrenimi ve derin öğrenmede, eğitim başlamadan önce değerlerine karar verdiğiniz veya yapılandırmalarını seçtiğiniz ve eğitim bittiğinde değerleri veya yapılandırması aynı kalacak olan her şey bir hiper parametredir.

Yaygın hiperparemetre örnekleri:

- Eğitim testi bölme oranı
- Optimizasyon algoritmalarında öğrenme oranı (örn. gradyan inişi)
- Optimizasyon algoritması seçimi (örneğin, gradyan inişi, stokastik gradyan inişi)
- Bir sinir ağı (NN) katmanında etkinleştirme işlevi seçimi (örn. Sigmoid, ReLU, Tanh)
- Modelin kullanacağı maliyet veya kayıp fonksiyonu seçimi
- Bir NN'deki gizli katmanların sayısı

- Her katmandaki etkinleştirme birimi sayısı
- NN cinsinden bırakma oranı (bırakma olasılığı)
- Bir NN eğitimindeki yineleme (dönem) sayısı
- Bir kümeleme görevindeki küme sayısı
- Evrişimli katmanlarda çekirdek veya filtre boyutu
- Havuzlama boyutu
- Parti boyutu

Parametreler:
Öte yandan parametreler modelin içindedir. Yani, kullanılan algoritma girdi özellikleri ile etiketler veya hedefler arasındaki eşlemeyi öğrenmeye çalıştığından, eğitim sırasında tamamen verilerden öğrenilir veya tahmin edilirler.

Model eğitimi tipik olarak bazı değerlere (rastgele değerlere veya sıfırlara ayarlanmış) başlatılan parametrelerle başlar. Eğitim/öğrenme ilerledikçe, ilk değerler bir optimizasyon algoritması (örneğin gradyan inişi) kullanılarak güncellenir. Öğrenme algoritması, öğrenme ilerledikçe parametre değerlerini sürekli olarak günceller, ancak model tasarımcısı tarafından ayarlanan hiperparametre değerleri değişmeden kalır.

Öğrenme sürecinin sonunda, model parametreleri, modelin kendisini oluşturan şeydir.

Parametre örnekleri:
- Doğrusal ve lojistik regresyon modellerinin katsayıları (veya ağırlıkları).
- Bir NN'nin ağırlıkları ve yanlılık, meyinlenme, eğilim
- Kümelemedeki küme merkezleri

Basitçe söylemek gerekirse, makine öğrenimi ve derin öğrenmedeki parametreler, öğrenme algoritmanızın öğrenirken bağımsız olarak değiştirebileceği değerlerdir ve bu değerler, sağladığınız hiperparametrelerin seçiminden etkilenir. Böylece eğitim başlamadan önce hiperparametreleri ayarlarsınız ve öğrenme algoritması parametreleri öğrenmek için bunları kullanır. Eğitim sahnesinin arkasında parametreler sürekli olarak güncellenir ve eğitim sonundaki son olanlar modelinizi oluşturur.

Bu nedenle, doğru hiperparametre değerlerinin ayarlanması çok önemlidir çünkü model eğitimi sırasında bunların kullanılmasından kaynaklanacak modelin performansını doğrudan etkiler. Modeliniz için en iyi hiperparametreleri seçme sürecine hiperparametre ayarlama denir.

Makine Öğrenimi Modellerinde Hiperparametre Optimizasyonu:
Öğretici, bir makine öğrenimi modelinde bir parametrenin ve bir hiper parametrenin ne olduğunu ve modelinizin performansını artırmak için neden hayati olduğunu açıklar.

Makine öğrenimi, verileri tahmin etmeyi ve sınıflandırmayı içerir ve bunu yapmak için veri kümesine göre çeşitli makine öğrenimi modelleri kullanırsınız. Makine öğrenimi modelleri, davranışlarının belirli bir soruna göre ayarlanabilmesi için parametrelendirilir. Bu modeller birçok parametreye sahip olabilir ve en iyi parametre kombinasyonunu bulmak bir arama problemi olarak ele alınabilir. Ancak, uygulamalı makine öğreniminde yeniyseniz, parametre adı verilen bu terim size yabancı gelebilir. Ama endişelenme! Bu blogun ilk etapta bunu öğreneceksiniz ve ayrıca bir makine öğrenimi modelinin bir parametresi ile bir hiperparametresi arasındaki farkın ne olduğunu keşfedeceksiniz. Bu blog aşağıdaki bölümlerden oluşmaktadır:

- Bir makine öğrenimi modelinde parametre ve hiper parametre nedir?
- Modelinizin performansını artırmak için hiperparametre optimizasyonu/ayarlaması neden hayati önem taşıyor?
- Hiperparametreleri optimize etmek/ayarlamak için iki basit strateji
- Python'da iki stratejiyle basit bir vaka çalışması

Bir makine öğrenimi öğrenme modelinde parametre nedir?
Model parametresi, modele dahil olan ve değeri verilen verilerden tahmin edilebilen bir konfigürasyon değişkenidir.

- Tahminler yapılırken model tarafından gereklidirler.
- Değerleri, modelin probleminiz üzerindeki becerisini tanımlar.
- Tahmin edilirler veya verilerden öğrenilirler.
- Genellikle uygulayıcı tarafından manuel olarak ayarlanmazlar.
- Genellikle öğrenilen modelin bir parçası olarak kaydedilirler.

Bu nedenle, yukarıdaki noktalardan ana çıkarımınız, makine öğrenme algoritmaları için çok önemli olan parametreler olmalıdır. Ayrıca, geçmiş eğitim verilerinden öğrenilen modelin bir parçasıdırlar. Hadi biraz daha derine inelim. Genel olarak programlama yaparken kullandığınız fonksiyon parametrelerini düşünün. Bir fonksiyona bir parametre iletebilirsiniz. Bu durumda parametre, bir dizi değerden birine sahip olabilen bir işlev argümanıdır. Makine öğreniminde, kullandığınız belirli model işlevdir ve yeni veriler üzerinde bir tahminde bulunmak için parametreler gerektirir. Bir modelin sabit veya değişken sayıda parametreye sahip olup olmadığı, “parametrik” veya “parametrik olmayan” olarak adlandırılıp adlandırılamayacağını belirler.
Model parametrelerinin bazı örnekleri şunları içerir:

- Bir yapay sinir ağındaki ağırlıklar.
- Bir destek vektör makinesindeki destek vektörleri.
- Doğrusal bir regresyon veya lojistik regresyondaki katsayılar.

Bir makine öğrenimi öğrenme modelinde hiper parametre nedir?
Model hiperparametresi, modelin dışında olan ve değeri verilerden tahmin edilemeyen bir konfigürasyondur.
• Model parametrelerini tahmin etmeye yardımcı olmak için genellikle süreçlerde kullanılırlar.
• Genellikle pratisyen tarafından belirtilirler.
• Genellikle buluşsal yöntemler kullanılarak ayarlanabilirler.
• Genellikle belirli bir tahmine dayalı modelleme problemi için ayarlanırlar.

Belirli bir problemde bir model hiperparametresi için en iyi değeri bilemezsiniz. Pratik kuralları kullanabilir, diğer konularda kullanılan değerleri kopyalayabilir veya deneme yanılma yoluyla en iyi değeri arayabilirsiniz. Bir makine öğrenimi algoritması belirli bir problem için ayarlandığında, esasen, en yetenekli tahminlerle sonuçlanan modelin parametrelerini keşfetmek için modelin hiper parametrelerini ayarlarsınız.

"Applied Predictive Modelling" adlı çok popüler bir kitaba göre - "Birçok model, verilerden doğrudan tahmin edilemeyen önemli parametrelere sahiptir. Örneğin, K-en yakın komşu sınıflandırma modelinde… Uygun bir değeri hesaplamak için analitik bir formül bulunmadığından, bu tür model parametresi bir ayar parametresi olarak anılır."

Model hiperparametreleri, genellikle işleri kafa karıştırıcı hale getirebilecek model parametreleri olarak adlandırılır. Bu karışıklığın üstesinden gelmek için iyi bir kural şudur: "Bir model parametresini manuel olarak belirtmeniz gerekiyorsa, bu muhtemelen bir model hiperparametresidir. " Bazı model hiperparametre örnekleri şunları içerir:

- Bir sinir ağını eğitmek için öğrenme oranı.
- Destek vektör makineleri için C ve sigma hiperparametreleri.
- k'deki k-en yakın komşular.

Bir makine öğrenimi modelinde doğru hiper parametre değerlerinin önemi:
Hiperparametreler hakkında düşünmenin en iyi yolu, tıpkı bir AM radyonun düğmelerini net bir sinyal almak için çevirebildiğiniz gibi, performansı optimize etmek için ayarlanabilen bir algoritmanın ayarları gibidir. Bir makine öğrenimi modeli oluştururken, model mimarinizi nasıl tanımlayacağınıza ilişkin tasarım seçenekleri sunulur. Çoğu zaman, belirli bir model için en uygun model mimarisinin ne olması gerektiğini hemen bilemezsiniz ve bu nedenle bir dizi olasılığı keşfetmek istersiniz. Gerçek bir makine öğrenimi tarzında, ideal olarak makineden bu keşfi gerçekleştirmesini ve en uygun model mimarisini otomatik olarak seçmesini isteyeceksiniz.

Örnek olay bölümünde, doğru hiperparametre değerleri seçiminin bir makine öğrenimi modelinin performansını nasıl etkilediğini göreceksiniz. Bu bağlamda, doğru değer

kümesinin seçilmesi tipik olarak "Hiperparametre optimizasyonu" veya "Hiperparametre ayarı" olarak bilinir.

Hiperparametreleri optimize etmek/ayarlamak için iki basit strateji:
Modeller birçok hiperparametreye sahip olabilir ve en iyi parametre kombinasyonunu bulmak bir arama problemi olarak ele alınabilir.
Şu anda birçok hiperparametre optimizasyonu/ayar algoritması olmasına rağmen, bu gönderi iki basit stratejiyi tartışıyor: 1. grid araması ve 2. Rastgele Arama.

Hiperparametrelerin grid araması:
Grid araması, bir gridda belirtilen algoritma parametrelerinin her bir kombinasyonu için metodik olarak bir model oluşturacak ve değerlendirecek olan hiperparametre ayarlamaya yönelik bir yaklaşımdır.

Aşağıdaki örneği ele alalım:
Bir makine öğrenimi modeli X'in a1, a2 ve a3 hiper parametrelerini aldığını varsayalım. Grid aramada, önce a1, a2 ve a3 hiperparametrelerinin her biri için değer aralığını tanımlarsınız. Bunu, hiperparametrelerin her biri için bir dizi değer olarak düşünebilirsiniz. Şimdi grid arama tekniği, ilk başta tanımladığınız tüm olası hiperparametre (a1, a2 ve a3) değerleri kombinasyonlarıyla X'in birçok versiyonunu oluşturacaktır. Bu hiperparametre değerleri aralığına grid adı verilir.

Grid şu şekilde tanımladığınızı varsayalım:
a1 = [0,1,2,3,4,5]
a2 = [10,20,30,40,5,60]
a3 = [105,105,110,115,120,125]

Hiperparametreler için tanımladığınız değerler dizisinin, hiperparametre yalnızca Tamsayı değerleri alıyorsa diziye Kayan tip değerleri sağlayamayacağınız bir anlamda meşru olması gerektiğini unutmayın.

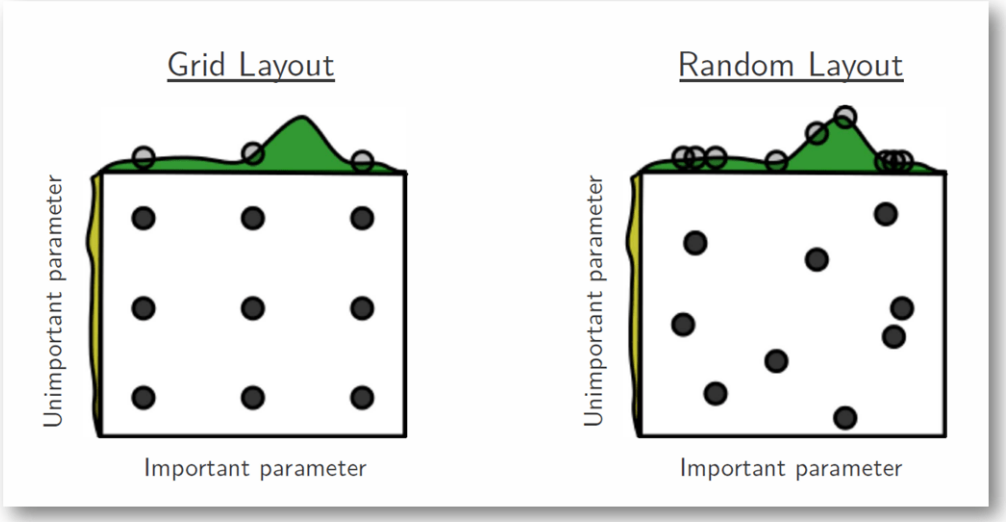
Şimdi, grid araması, az önce tanımladığınız grid ile X'in birkaç sürümünü oluşturma sürecine başlayacaktır.

[0,10,105] kombinasyonu ile başlayacak ve [5,60,125] ile bitecek. Bu ikisi arasındaki tüm ara kombinasyonlardan geçecek ve bu da grid aramasını hesaplama açısından çok pahalı hale getiriyor.

Hiperparametrelerin rastgele aranması:
İstatistikte, dağılımla, esasen, gözlemlenen veya teorik oluşum sıklığını gösteren bir değişkenin değerlerinin düzenlenmesi kastedilmektedir. Öte yandan, Örnekleme istatistikte kullanılan bir terimdir. Bir bütün olarak popülasyon hakkında bir şeyler anlamak için hedef popülasyondan temsili bir örnek seçme ve bu örnekten veri toplama sürecidir.

Her hiperparametre için bir örnekleme dağılımı tanımlayacaksınız. Optimum modeli ararken kaç yineleme oluşturmak istediğinizi de tanımlayabilirsiniz. Her yineleme için, modelin hiperparametre değerleri, tanımlanan dağılımların örneklenmesiyle belirlenecektir. Grid araması yerine rastgele arama kullanımını motive eden birincil teorik desteklerden biri, çoğu durumda hiperparametrelerin eşit derecede önemli olmadığı gerçeğidir. Çoğu veri kümesi için hiper parametrelerden yalnızca birkaçı gerçekten önemlidir, ancak farklı veri kümelerinde farklı hiper parametreler önemlidir. Bu fenomen, grid aramasını yeni veri kümeleri için algoritmaları yapılandırmak için kötü bir seçim haline getiriyor."

Bir hiperparametrenin model puanını optimize etme üzerinde önemli ölçüde daha fazla etkiye sahip olduğu bir hiperparametre alanı üzerinde arama yapıyoruz - her eksende gösterilen dağılımlar modelin puanını temsil ediyor. Her durumda, 9 farklı modeli değerlendiriyoruz. Şebeke arama stratejisi, optimal modeli açıkça gözden kaçırır ve önemsiz parametreyi keşfetmek için fazladan zaman harcar. Bu grid araması sırasında, her bir hiperparametreyi izole ettik ve diğer tüm hiperparametreleri sabit tutarken mümkün olan en iyi değeri aradık. İncelenmekte olan hiperparametrenin elde edilen model puanı üzerinde çok az etkisi olduğu durumlarda bu, boşa harcanan çabayla sonuçlanır. Tersine, rastgele arama çok daha gelişmiş keşif gücüne sahiptir ve kritik hiperparametre için en uygun değeri bulmaya odaklanabilir.



4.8. Metrikler

- Model ne kadar iyi gidiyor? Kullanışlı bir model mi?
- Modeli daha fazla veri üzerinde eğitmek performansını iyileştirecek mi?
- Daha fazla özellik eklenmesi gerekiyor mu?

Eğitim / Test / Onaylama ayrımı
Modeli doğru bir şekilde değerlendirmek için yapabileceğiniz en önemli şey, modeli tüm veri kümesi üzerinde eğitmemektir. Tekrar ediyorum: modeli tüm veri kümesi üzerinde eğitmeyin. Tipik bir eğitim/test ayrımı, verilerin %70'inin eğitim için ve %30'unun test için kullanılması olacaktır.

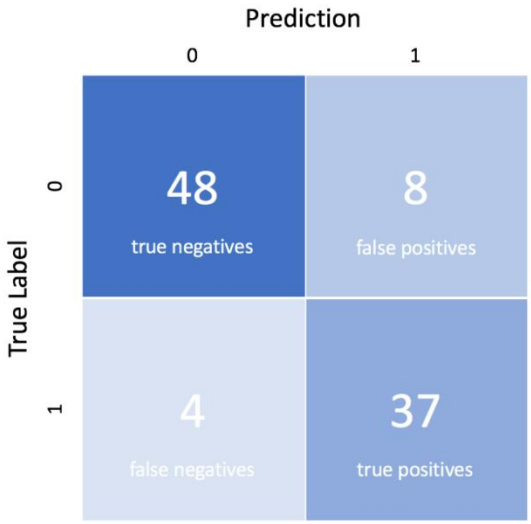
Eğitim setine fazla uyma olasılığını önlemek için model değerlendirirken yeni verileri kullanmak önemlidir. Bununla birlikte, bazen bir modelin en iyi parametrelerini bulmak için inşa ederken modeli değerlendirmek faydalı olabilir - ancak bu değerlendirme için test setini kullanamayız, aksi takdirde en iyi performansı gösteren parametreleri seçeriz. Belki de en iyi genelleştiren parametreler değil. Modeli oluşturmaya ve ayarlamaya devam ederken modeli değerlendirmek için, doğrulama kümesi olarak bilinen verilerin üçüncü bir alt kümesi oluşturulur. Tipik bir eğitim/test/doğrulama ayrımı, eğitim için verilerin %60'ını, doğrulama için verilerin %20'sini ve test için verilerin %20'sini kullanmak olacaktır.

Ayrıca, bu bölmeleri yapmadan önce verileri karıştırmanın çok önemli olduğunu, böylece her bölmenin veri kümesini doğru bir şekilde temsil etmesi sağlanacaktır.

Metrikler
Sınıflandırma metrikleri:
Sınıflandırma tahminleri gerçekleştirirken ortaya çıkabilecek dört tür sonuç vardır.

- Gerçek pozitifler, bir gözlemin bir sınıfa ait olduğunu ve aslında o sınıfa ait olduğunu tahmin ettiğiniz zamandır.
- Gerçek olumsuzluklar, bir gözlemin bir sınıfa ait olmadığını ve aslında o sınıfa ait olmadığını tahmin ettiğiniz zamandır.
- Bir gözlemin gerçekte öyle olmadığı halde bir sınıfa ait olduğunu tahmin ettiğinizde yanlış pozitifler meydana gelir.
- Yanlış negatifler, bir gözlemin bir sınıfa ait olmadığı halde aslında öyle olduğunu tahmin ettiğinizde ortaya çıkar.

Bu dört sonuç genellikle bir karışıklık matrisi üzerinde çizilir. Aşağıdaki karışıklık matrisi, ikili sınıflandırma durumuna bir örnektir. Bu matrisi, test verileriniz üzerinde tahminler yaptıktan ve ardından her bir tahmini yukarıda açıklanan dört olası sonuçtan biri olarak tanımladıktan sonra oluşturursunuz.



Bir sınıflandırma modelini değerlendirmek için kullanılan üç ana metrik doğruluk, kesinlik ve hatırlamadır.

Doğruluk, test verileri için doğru tahminlerin yüzdesi olarak tanımlanır. Doğru tahmin sayısı toplam tahmin sayısına bölünerek kolayca hesaplanabilir.
doğruluk=doğru tahminler /tüm tahminlerdoğruluk=doğru tahminler/tüm tahminler

Kesinlik, belirli bir sınıfa ait olduğu tahmin edilen tüm örnekler arasında ilgili örneklerin (gerçek pozitiflerin) oranı olarak tanımlanır.
kesinlik=doğru pozitifler / yanlış pozitifler
kesinlik=gerçek pozitifler / yanlış pozitifler

Geri çağırma, bir sınıfa ait olduğu tahmin edilen örneklerin, gerçekten sınıfa ait olan tüm örneklere göre oranı olarak tanımlanır.
geri çağırma=gerçek pozitifler / yanlış negatifler

Aşağıdaki grafik, kesinlik ve geri çağırma arasındaki farkı görselleştiren olağanüstü bir iş çıkarıyor.

Bir kişinin kanser olduğunu yanlış tahmin eden bir model istemezsiniz (kişi sahip olmadığı bir hastalık için acı verici ve pahalı bir tedavi sürecine girer) ama aynı zamanda bir kişiyi yanlış tahmin etmek istemezsiniz. Bu nedenle, bir modelin hem kesinliğini hem de geri çağrılmasını değerlendirmek önemlidir.

Kesinlik ve hatırlama arasındaki dengeyi açıklamak için başka örnekler ararken, intiharı tahmin etmek için makine öğrenimini kullanmayı tartışan aşağıdaki makaleye rastladım. Bu durumda, modelin kesinliğinden çok geri çağrılmasına daha fazla odaklanmak isteriz. Aslında intiharı düşünmeyen birine müdahale etmek, intiharı düşünen birini kaçırmaktan çok daha

az zararlı olacaktır. Bununla birlikte, kesinlik yine de önemlidir, çünkü modelinizin yanlış pozitifleri öngördüğü çok fazla örnek istemezsiniz.

Regresyon metrikleri

Regresyon modelleri için değerlendirme ölçüleri, sınıflandırma modelleri için tartıştığımız yukarıdaki ölçülerden oldukça farklıdır çünkü artık ayrık sayıda sınıf yerine sürekli bir aralıkta tahmin yapıyoruz. Regresyon modeliniz bir evin fiyatını 400 bin dolar olarak öngörüyorsa ve 405 bin dolara satıyorsa, bu oldukça iyi bir tahmindir. Ancak sınıflandırma örneklerinde sadece bir tahminin doğru mu yanlış mı olduğu ile ilgilendik, bir tahminin "oldukça iyi" olduğunu söyleme yeteneği yoktu. Bu nedenle, regresyon modelleri için farklı bir dizi değerlendirme ölçütüne sahibiz.

Açıklanan varyans, beklenen sonuçlar içindeki varyansı karşılaştırır ve bunu modelimizin hatasındaki varyansla karşılaştırır. Bu metrik esas olarak, modelimizin açıklayabildiği orijinal veri kümesindeki varyasyon miktarını temsil eder.

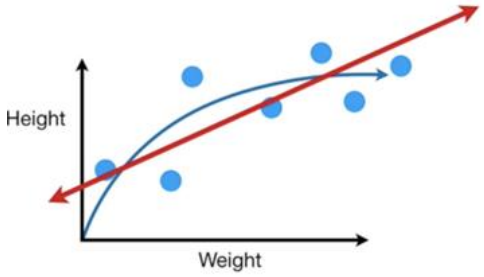
Ortalama kare hatası, tahmin edilen çıktı ile gerçek çıktı arasındaki kare farklarının ortalaması olarak basitçe tanımlanır. Kare hatası yaygın olarak kullanılır, çünkü tahminin çok yüksek veya çok düşük olup olmadığı konusunda agnostiktir, sadece tahminin yanlış olduğunu bildirir.

4.9. Yanlılık ve Varyans

Bir makine öğrenimi yönteminin gerçek ilişkiyi yakalayamamasına yanlılık denir. İstatistiksel tanımın aksine, varyans, verilerin yayılması değil, bir modelin doğruluğunun farklı veri kümelerine göre nasıl değiştiği anlamına gelir. Veri kümeleri arasındaki uyum farklarına varyans denir.

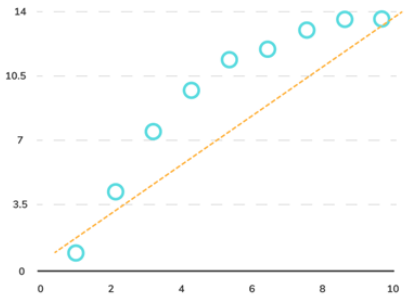
Varyans: Modelin tahmin ettiği verilerin, gerçek verilerin etrafında nasıl (ne kadar) saçıldığını ölçer.

Öyargı, yanlılık (Bias):
Yanlılık terimi, y-kesme noktası değil, modelin örneklerle aynı çizgide olan bir eğriyi oluşturmadaki başarısızlığının derecesidir. Bir makine öğrenimi yönteminin (doğrusal regresyon gibi) gerçek ilişkiyi yakalayamamasına yanlılık denir.



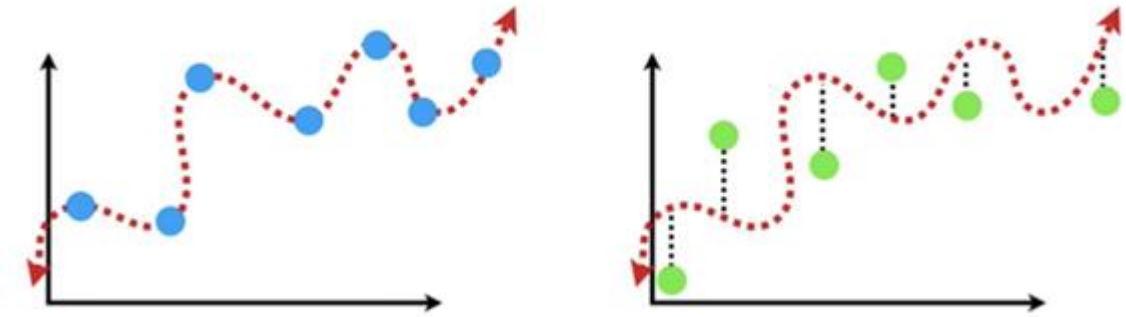
Yanlılık, ML modelinin tahmin edilen değerleri ile gerçek değeri arasındaki fark olarak tanımlanır. Yanlılık, hem eğitim hem de test verilerinde önemli bir yanlışlığa neden olur. Bir algoritmanın her zaman düşük yanlı olması tavsiye edilir.

Öngörülen veriler, önemli sapma nedeniyle düz bir çizgi biçimindedir ve bu nedenle veri kümesine tam olarak uymaz. Verilerin eksik takılması, bu tür bir uydurma için kullanılan terimdir. Bu, teori çok basit veya doğrusal olduğunda ortaya çıkar. Aşağıdaki grafiği bunun gibi bir duruma örnek olarak düşünün.

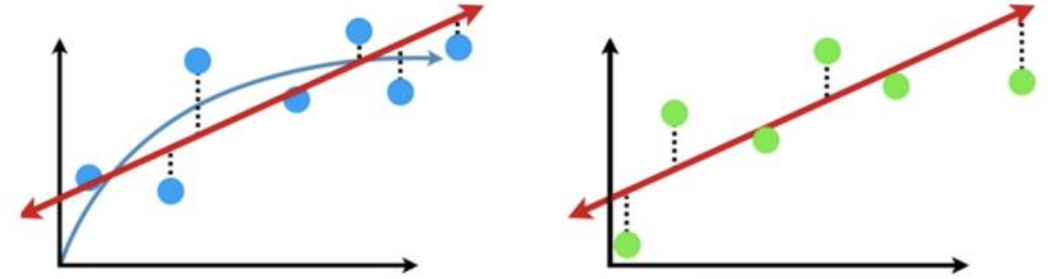


Varyans (değişim):

İstatistiksel tanımın aksine, varyans, verilerin yayılması değil, bir modelin doğruluğunun farklı veri kümelerine göre nasıl değiştiği anlamına gelir. Veri kümeleri arasındaki uyum farklarına varyans denir.



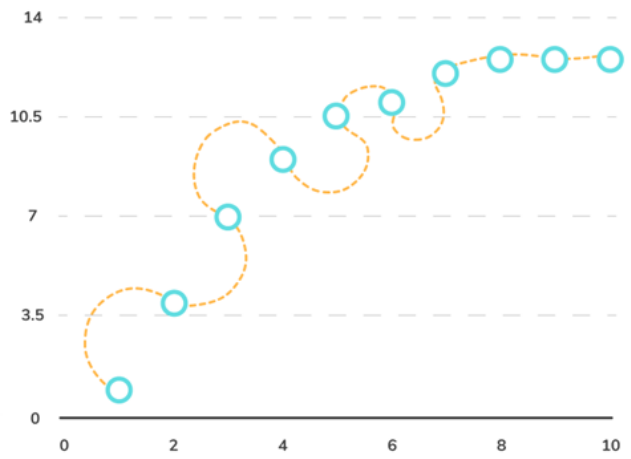
Önceki görüntüdeki dalgalı çizgi, diğer veri kümelerinde kökten farklı bir performans sergiliyor. Bu nedenle, yüksek bir varyansa sahip olduğunu söylüyoruz. Öte yandan, farklı veri kümeleri için karelerin toplamı benzer olduğu için düz çizgi nispeten düşük varyansa sahiptir.



Modelin varyansı, verilerimizin dağılımı hakkında bize bilgi veren belirli bir veri noktası için model tahmininin değişkenliğidir. Doğrulama hatası ile eğitim hatası arasındaki farktır. Yüksek varyansa sahip model, eğitim verilerine çok karmaşık bir uyum sağlar ve bu nedenle yeni verilere tam olarak uymaz. Sonuç olarak, bu tür modeller eğitim verileri üzerinde iyi performans gösterirken, verileri test ederken yüksek hata oranlarına sahiptir.

Bir modelin varyansı aşırı olduğunda, Verilerin Fazla Uyulması olarak adlandırılır. Karmaşık bir eğri ve yüksek mertebeden bir hipotez kullanarak eğitim setini doğru bir şekilde uydurmayı içeren fazla uydurma, bilinmeyen verilerle ilgili hata önemli olduğundan geçerli bir seçenek değildir.

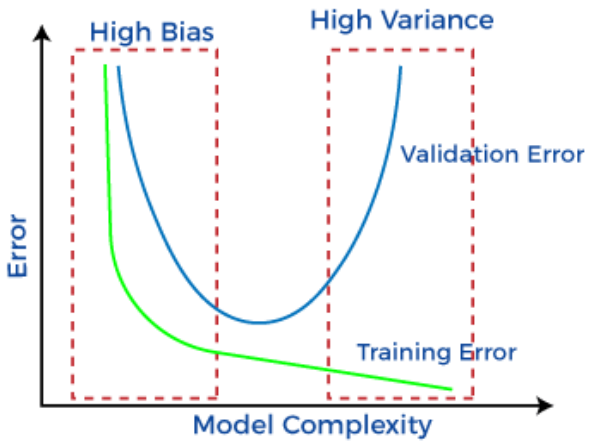
Bir veri modeli eğitilirken varyans minimumda tutulmalıdır.



Model, iki hata arasında en iyi dengeyi sağlamak için her zaman düşük bir sapma ve düşük bir varyansı hedeflemelidir.

Makine Öğreniminde Önyargı ve Varyans:

Makine öğrenimi, makinelerin veri analizi yapmasına ve tahminler yapmasına olanak tanıyan bir Yapay Zeka dalıdır. Ancak makine öğrenme modeli doğru değilse tahmin hataları yapabilir ve bu tahmin hataları genellikle Yanlılık ve Varyans olarak bilinir. Makine öğreniminde, model tahminleri ile gerçek tahminler arasında her zaman küçük bir fark olduğu için bu hatalar her zaman mevcut olacaktır. Makine öğrenimi/veri bilimi analistlerinin temel amacı, daha doğru sonuçlar elde etmek için bu hataları azaltmaktır. Bu başlıkta, önyargı ve varyans, Bias-varyans değiş tokuşu, Eksik Uyum ve Fazla Uyum tartışacağız. Ancak başlamadan önce, Makine öğrenimindeki hataların neler olduğunu anlayalım.



Önyargı (Sapma) nedir?

Genel olarak, bir makine öğrenimi modeli verileri analiz eder, içindeki kalıpları bulur ve tahminlerde bulunur. Eğitim sırasında model, veri kümesindeki bu kalıpları öğrenir ve bunları tahmin için test verilerine uygular. Tahminler yapılırken, model tarafından yapılan tahmin değerleri ile gerçek değerler/beklenen değerler arasında bir fark oluşur ve bu fark bias hataları veya bias kaynaklı hatalar olarak bilinir. Doğrusal Regresyon gibi makine öğrenme algoritmalarının veri noktaları arasındaki gerçek ilişkiyi yakalayamaması olarak tanımlanabilir. Her algoritma bir miktar sapma ile başlar, çünkü modeldeki varsayımlardan sapma oluşur, bu da hedef fonksiyonun öğrenilmesini kolaylaştırır. Bir modelde şunlar bulunur:

- **Düşük Sapma:** Düşük sapma modeli, hedef fonksiyonun biçimi hakkında daha az varsayımda bulunacaktır.
- **Yüksek Önyargı:** Yüksek önyargılı bir model daha fazla varsayımda bulunur ve model, veri setimizin önemli özelliklerini yakalayamaz hale gelir. Yüksek önyargılı bir model de yeni veriler üzerinde iyi performans gösteremez.

Genel olarak, doğrusal bir algoritma, hızlı öğrenmelerini sağladığı için yüksek bir önyargıya sahiptir. Algoritma ne kadar basit olursa, tanıtılması muhtemel önyargı o kadar yüksek olur. Oysa doğrusal olmayan bir algoritma genellikle düşük önyargıya sahiptir.

Düşük önyargılı makine öğrenimi algoritmalarının bazı örnekleri, Karar Ağaçları, k-En Yakın Komşular ve Destek Vektör Makineleridir. Aynı zamanda yüksek yanlılığa sahip bir algoritma Lineer Regresyon, Lineer Diskriminant Analizi ve Lojistik Regresyondur.

Yüksek Önyargıyı Azaltmanın Yolları:

Yüksek yanlılık esas olarak çok basit bir model nedeniyle oluşur. Aşağıda yüksek yanlılığı azaltmanın bazı yolları verilmiştir:

- Model eksik olduğundan girdi özelliklerini artırın.
- Düzenlileştirme süresini azaltın.
- Bazı polinom özelliklerini dahil etmek gibi daha karmaşık modeller kullanın.

Varyans Hatası nedir?

Varyans, farklı eğitim verilerinin kullanılması durumunda tahmindeki varyasyon miktarını belirtecektir. Basit bir deyişle, varyans, rastgele bir değişkenin beklenen değerinden ne kadar farklı olduğunu söyler. İdeal olarak, bir model bir eğitim veri kümesinden diğerine çok fazla farklılık göstermemelidir, bu da algoritmanın girdiler ve çıktı değişkenleri arasındaki gizli eşlemeyi anlamada iyi olması gerektiği anlamına gelir. Varyans hataları, düşük varyanslı veya yüksek varyanslıdır.

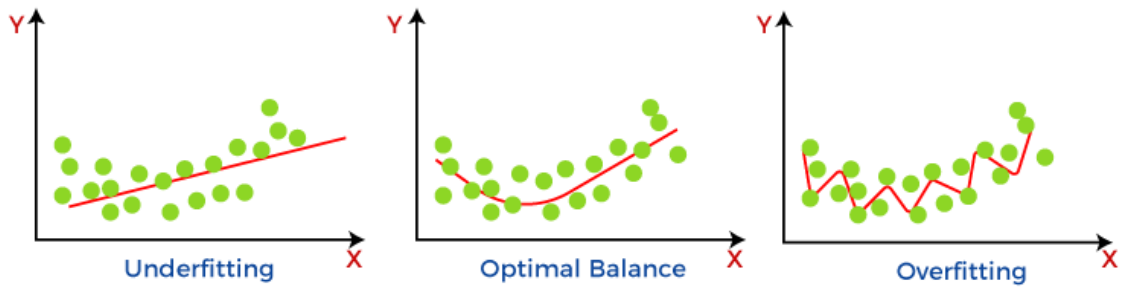
Düşük varyans, eğitim veri setindeki değişikliklerle hedef fonksiyonun tahmininde küçük bir değişiklik olduğu anlamına gelir. Aynı zamanda, Yüksek varyans, eğitim veri kümesindeki değişikliklerle hedef fonksiyonun tahmininde büyük bir değişiklik gösterir.

Yüksek varyans gösteren bir model, çok şey öğrenir ve eğitim veri kümesiyle iyi performans gösterir ve görünmeyen veri kümesiyle iyi genelleme yapmaz. Sonuç olarak böyle bir model eğitim veri seti ile iyi sonuçlar verir ancak test veri seti üzerinde yüksek hata oranları gösterir.

Yüksek varyansla model, veri setinden çok fazla şey öğrendiğinden modelin aşırı uyumuna yol açar. Yüksek varyansa sahip bir model aşağıdaki problemlere sahiptir:

- Yüksek bir varyans modeli, fazla uydurmaya yol açar.
- Model karmaşıklıklarını artırın.

Genellikle, doğrusal olmayan algoritmalar, modele uyması için çok fazla esnekliğe sahiptir, yüksek varyansa sahiptir.



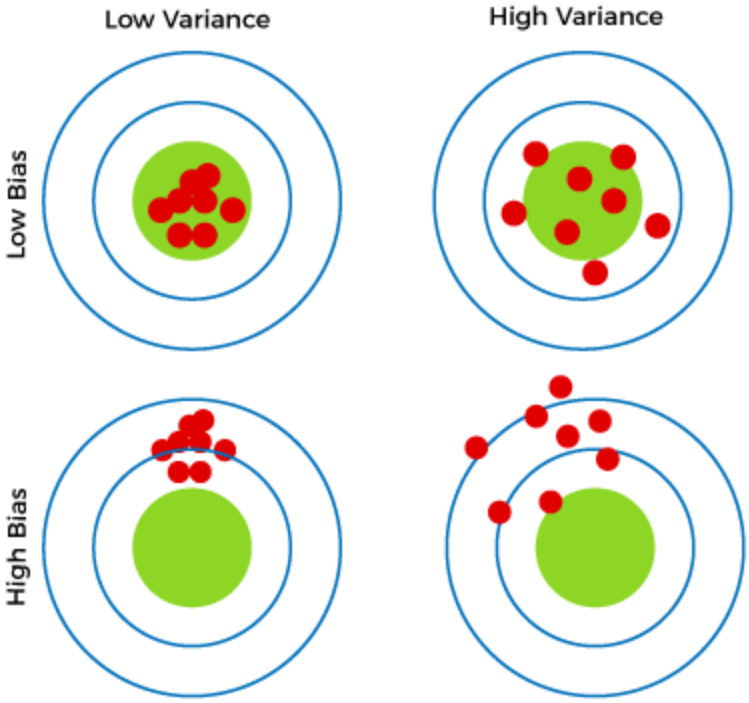
Düşük varyanslı makine öğrenimi algoritmalarının bazı örnekleri, Lineer Regresyon, Lojistik Regresyon ve Lineer diskriminant analizidir. Aynı zamanda, yüksek varyanslı algoritmalar karar ağacı, Destek Vektör Makinesi ve K-en yakın komşulardır.

Yüksek Varyansı Azaltmanın Yolları:

- Bir model fazla takıldığından giriş özelliklerini veya parametre sayısını azaltın.
- Çok karmaşık bir model kullanmayın.
- Eğitim verilerini artırın.
- Düzenleme süresini artırın.

Önyargı Varyansının Farklı Kombinasyonları

Aşağıdaki diyagramla temsil edilen dört olası yanlılık ve varyans kombinasyonu vardır:



1. Düşük Önyargı, Düşük Varyans: Düşük yanlılık ve düşük varyansın birleşimi, ideal bir makine öğrenimi modelini gösterir. Ancak, pratik olarak mümkün değildir.

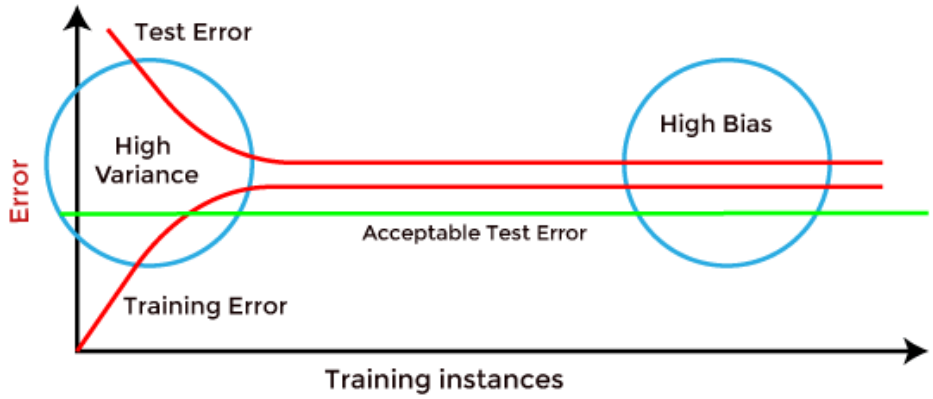
2. Düşük Sapma, Yüksek Varyans: Düşük sapma ve yüksek varyans ile model tahminleri ortalama olarak tutarsız ve doğrudur. Bu durum, model çok sayıda parametre ile öğrendiğinde ve dolayısıyla fazla uydurmaya yol açtığında ortaya çıkar.

3. Yüksek Önyargı, Düşük Varyans: Yüksek önyargı ve düşük varyans ile tahminler tutarlıdır ancak ortalama olarak yanlıştır. Bu durum, bir model eğitim veri kümesiyle iyi öğrenmediğinde veya az sayıda parametre kullandığında ortaya çıkar. Modelde yetersiz uyum sorunlarına yol açar.

4. Yüksek Önyargı, Yüksek Varyans: Yüksek yanlılık ve yüksek varyans ile tahminler tutarsız ve ortalama olarak da hatalıdır.

Yüksek Varyans veya Yüksek Önyargı nasıl belirlenir?

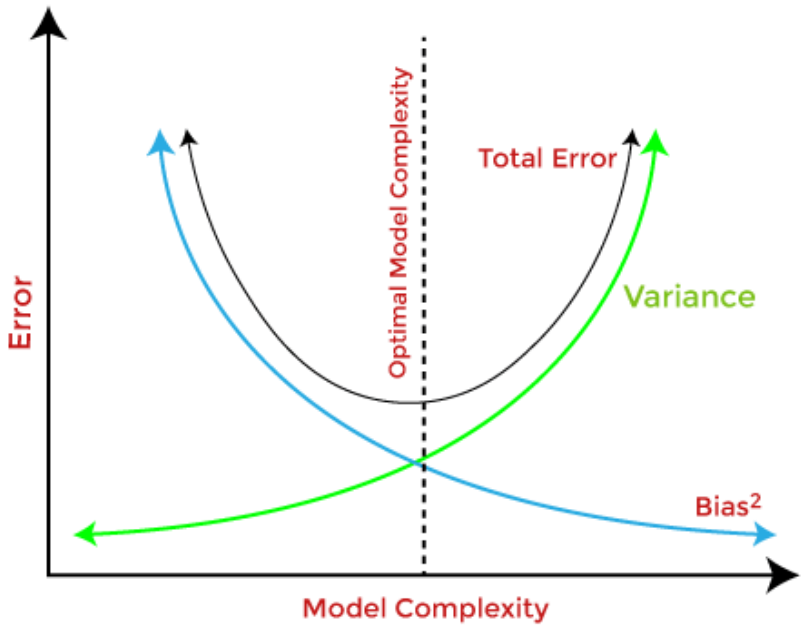
Modelin sahip olması durumunda yüksek varyans tespit edilebilir.: Düşük eğitim hatası ve yüksek test hatası.



Modelde şunlar varsa Yüksek Önyargı tanımlanabilir: Yüksek eğitim hatası ve test hatası neredeyse eğitim hatasına benzer.

Önyargı-Varyans Dengeleme

Makine öğrenimi modelini oluştururken, modele fazla ve eksik uymayı önlemek için önyargı ve varyansa dikkat etmek gerçekten önemlidir. Model daha az parametre ile çok basitse, düşük varyansa ve yüksek yanlılığa sahip olabilir. Oysa model çok sayıda parametreye sahipse, yüksek varyansa ve düşük yanlılığa sahip olacaktır. Bu nedenle, önyargı ve varyans hataları arasında bir denge yapılması gerekir ve önyargı hatası ile varyans hatası arasındaki bu denge, Önyargı-Varyans değiş tokuşu olarak bilinir.



Modelin doğru bir şekilde tahmin edilmesi için algoritmaların düşük bir varyansa ve düşük bir önyargıya ihtiyacı vardır. Ancak bu mümkün değildir çünkü yanlılık ve varyans birbiriyle ilişkilidir:

- Varyansı azaltırsak, yanlılığı artıracaktır.
- Önyargıyı azaltırsak, varyansı artıracaktır.

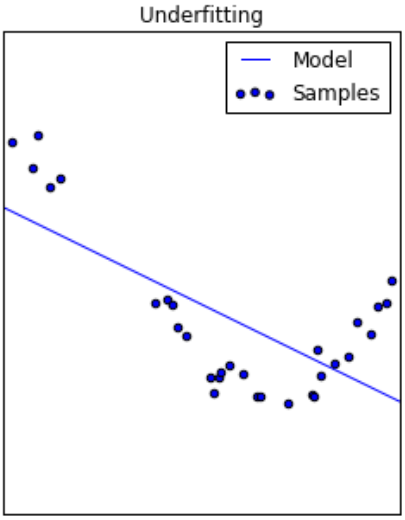
Önyargı-Varyans değiş tokuşu, denetimli öğrenmede merkezi bir konudur. İdeal olarak, eğitim verilerindeki düzenlilikleri doğru bir şekilde yakalayan ve aynı anda görünmeyen veri kümesiyle iyi bir şekilde genelleştiren bir modele ihtiyacımız var. Ne yazık ki, bunu aynı anda yapmak mümkün değildir. Çünkü yüksek varyanslı bir algoritma, eğitim verileriyle iyi performans gösterebilir, ancak gürültülü verilere fazla uymaya yol açabilir. Oysa yüksek önyargılı algoritma, verilerdeki önemli düzenlilikleri bile yakalayamayan çok basit bir model oluşturur. Bu nedenle, optimal bir model oluşturmak için önyargı ve varyans arasında tatlı bir nokta bulmamız gerekiyor.

Differentiate between bias and variance in the context of deep learning models. How can you achieve balance between the two?
Derin öğrenme modelleri bağlamında yanlılık ve varyans arasında ayrım yapıldığına ikisi arasındaki denge nasıl sağlanabilir?

Tahminleri anlamak söz konusu olduğunda, tahmin hatalarını anlamak çok önemlidir. İndirgenebilir (karesel sapma veya karesel varyanstan kaynaklanan hatalar) ve indirgenemez (bir sistemdeki rastgelelik veya doğal değişkenlik nedeniyle ortaya çıkan ve model değiştirilerek azaltılamayan hatalar) hatalar başlıca iki tür hatadır. İki tür indirgenebilir hata vardır: yanlılık ve varyans. Bu kusurların tam olarak kavranması, fazla ve eksik uyumu önleyerek doğru bir modelin oluşturulmasına yardımcı olur.

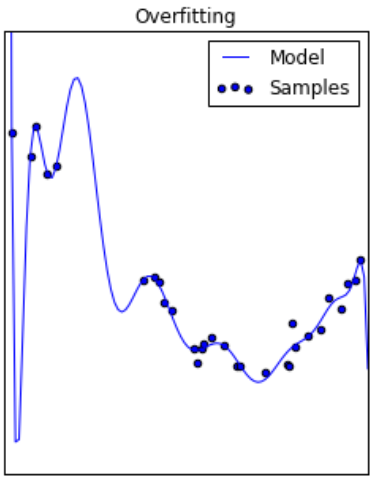
Herhangi bir makine öğrenimi modelinin nihai amacı, örneklerden öğrenmek ve onu gerçekleştirmesi için eğittiğimiz görevle ilgili bir dereceye kadar bilgiyi genelleştirmektir. Bazı makine öğrenimi modelleri, bu bilginin altında yatan yapıyı önererek genelleme için bir çerçeve sağlar. Örneğin, doğrusal bir regresyon modeli, onu beslediğimiz bilgiler arasındaki doğrusal ilişkileri öğrenmek için bir çerçeve uygular. Bununla birlikte, bazen çok fazla önceden oluşturulmuş yapıya sahip bir model sağlarız ve modelin örneklerden öğrenme yeteneğini sınırlarız - örneğin, üstel bir veri kümesi üzerinde doğrusal bir model eğittiğimiz durum gibi. Bu durumda, modelimiz önceden empoze edilen yapı ve ilişkiler tarafından önyargılıdır.

Yüksek önyargılı modeller sunulan verilere çok az dikkat eder; bu aynı zamanda yetersiz donatma olarak da bilinir.

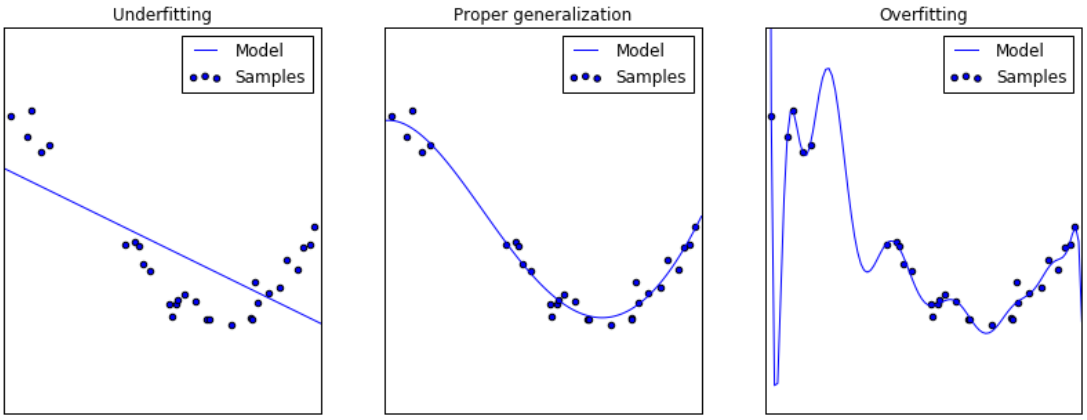


Ayrıca, gerekli tüm bilgileri sunmadan bir görevi gerçekleştirmeyi öğretmeye çalışarak bir modele önyargılı yaklaşmak da mümkündür. Modelin kısıtlamalarının modelin performansını etkilemediğini biliyorsanız, ancak yine de yetersiz uyum belirtileri gözlemliyorsanız, büyük olasılıkla modeli eğitmek için yeterli özellik kullanmıyorsunuzdur.

Diğer uçta, bazen modelimizi eğittiğimizde, eğitim verilerinden çok fazla şey öğrenir. Yani modelimiz sinyale ek olarak verilerdeki gürültüyü de yakalar. Bu, modelde gerçek eğilimi temsil etmeyen vahşi dalgalanmalara neden olabilir; bu durumda modelin yüksek varyansa sahip olduğunu söylüyoruz. Bu durumda modelimiz, yeni verilere genellemeyi dikkate almadan eğitim verilerine çok fazla önem verdiği için iyi genelleme yapamamaktadır. Başka bir deyişle, modeli eğitim verilerine fazla uydurduk.



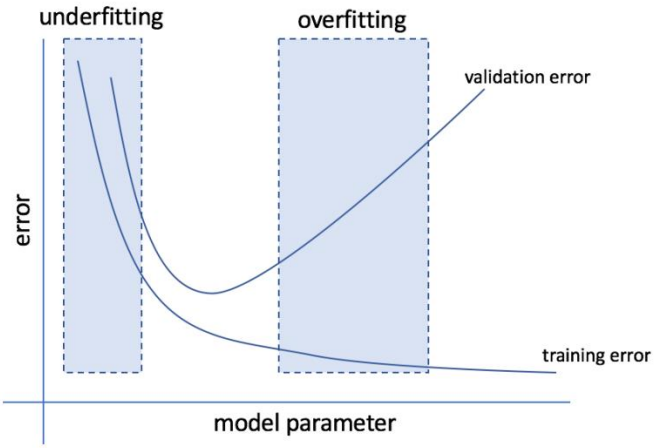
Özetle, yüksek yanlılığa sahip bir model, gerçek eğilimi öğrenmekle sınırlıdır ve verilere uymaz. Yüksek varyansa sahip bir model, eğitim verilerinden çok fazla şey öğrenir ve verilere gereğinden fazla uyar. En iyi model, iki uç noktanın ortasında bir yerde bulunur.



Doğrulama eğrileri

Herhangi bir makine öğrenimi modelinin amacı genellemedir. Doğrulama eğrileri, iyi genelleyen bir model oluşturmak için bir modeli eksik ve fazla uydurma arasındaki tatlı noktayı bulmamızı sağlar.

Tipik bir doğrulama eğrisi, modelin verilere fazla veya eksik uyma eğilimini kontrol eden bazı model hiperparametrelerinin bir fonksiyonu olarak model hatasının bir grafiğidir. Seçtiğiniz parametre, değerlendirdiğiniz belirli modele bağlıdır; örneğin, bir lineer regresyon modeli için polinom özelliklerinin derecesini (tipik olarak bu, bu dereceye kadar polinom özellikleriniz olduğu anlamına gelir) çizmeyi seçebilirsiniz. Genel olarak, seçilen parametre, modelin karmaşıklığı üzerinde bir dereceye kadar kontrole sahip olacaktır. Bu eğri üzerinde modelin hem eğitim hatasını hem de doğrulama hatasını çiziyoruz. Bu hataların her ikisini bir arada kullanarak, bir modelin yüksek yanlılıktan mı yoksa yüksek varyanstan mı muzdarip olduğunu teşhis edebiliriz.



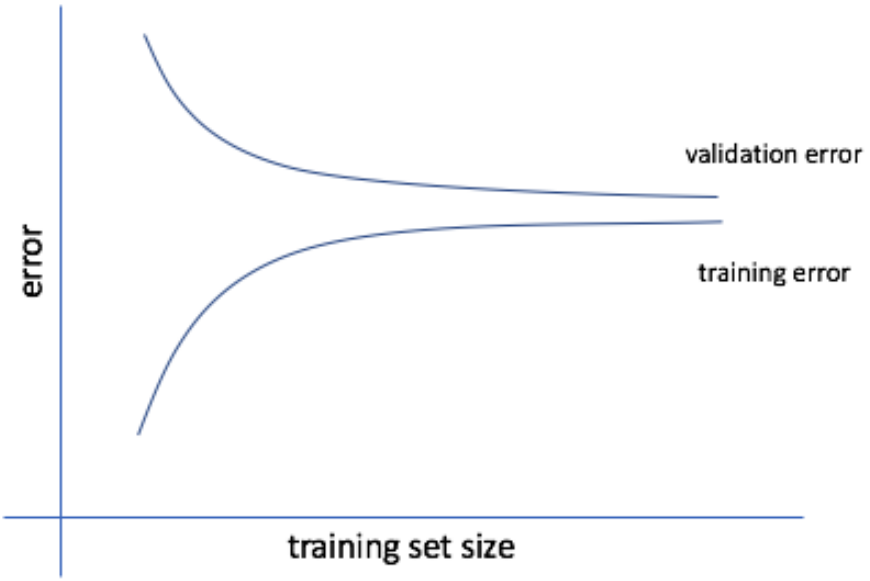
Hem eğitim hatasının hem de doğrulama hatasının yüksek olduğu bölgede, model yüksek yanlılığa tabidir. Burada verilerden ders çıkarılmaz ve kötü performans gösterir.

Eğitim hatasının ve doğrulama hatasının ayrıldığı bölgede, eğitim hatasının düşük kalması ve doğrulama hatasının artmasıyla yüksek varyansın etkilerini görmeye başlıyoruz. Modelimiz eğitim verilerinden yeni verilere genelleme yapamadığı için doğrulama hatası yüksek kalırken, verileri fazla uydurduğumuz ve eğitim örneklerinden çok fazla şey öğrendiğimiz için eğitim hatası düşüktür.

Öğrenme eğrileri

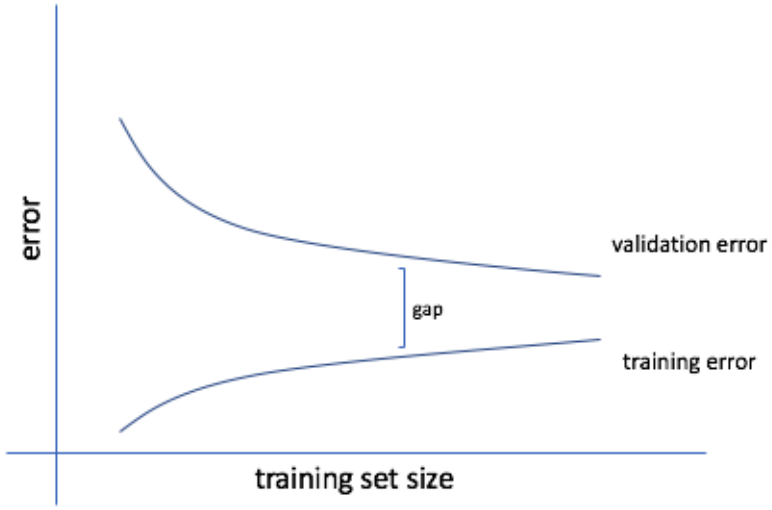
Bir modeldeki yanlılığı ve varyansı teşhis etmek için tartışacağımız ikinci araç, öğrenme eğrileridir. Burada, eğitim örneklerinin sayısının bir fonksiyonu olarak bir modelin hatasını çizeceğiz. Doğrulama eğrilerine benzer şekilde, hem eğitim verileri hem de doğrulama verileri için hatayı çizeceğiz.

Modelimiz yüksek önyargıya sahipse, doğrulama ve eğitim veri kümeleri için yüksek bir hataya oldukça hızlı yakınsama gözlemleyeceğiz. Model yüksek yanlılıktan muzdaripse, daha fazla veri üzerinde eğitim modeli geliştirmek için çok az şey yapacaktır. Bunun nedeni, verilere uymayan modellerin verilere çok az dikkat etmesidir, bu nedenle daha fazla veri beslemek işe yaramaz. Yüksek önyargıdan muzdarip modelleri iyileştirmeye yönelik daha iyi bir yaklaşım, modelin uygun ilişkileri öğrenmek için daha donanımlı olabilmesi için veri kümesine ek özellikler eklemeyi düşünmektir.



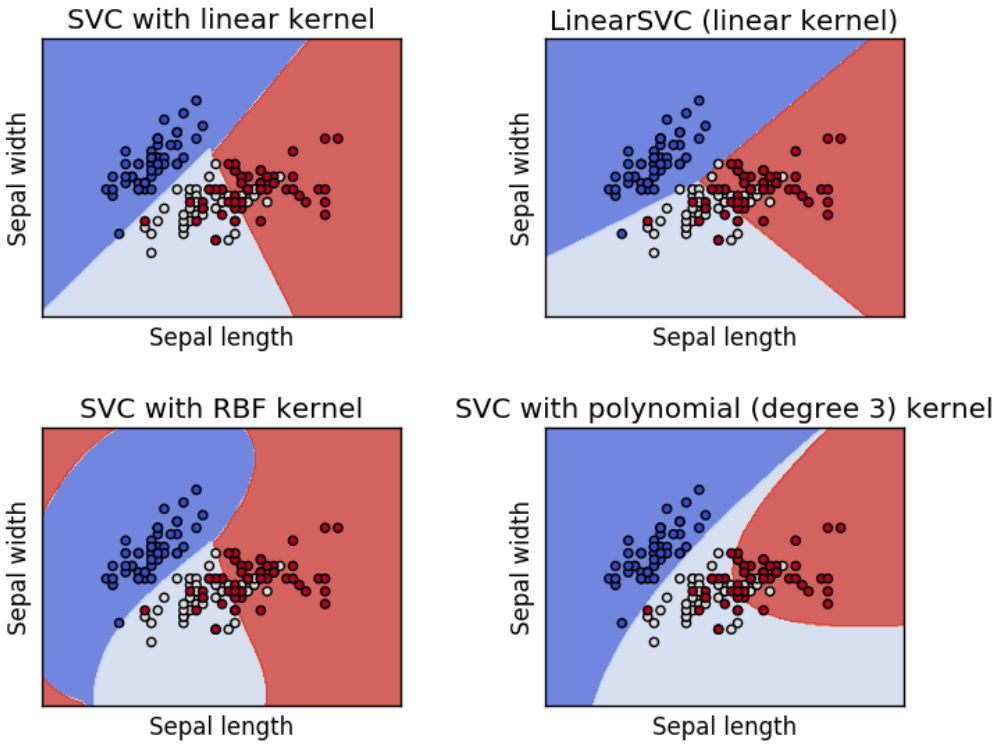
Modelimizin varyansı yüksekse, eğitim ve doğrulama hatası arasında bir boşluk görürüz. Bunun nedeni, modelin eğitim verileri için iyi performans göstermesidir, çünkü bu alt kümeye fazla uyarlanmıştır ve doğru ilişkileri genelleştiremediğinden doğrulama verileri için

düşük performans gösterir. Bu durumda, eğitim sırasında daha fazla veri beslemek, modelin performansını iyileştirmeye yardımcı olabilir.



Diğer pratik tavsiyeler

Sınıflandırıcı modellerini değerlendirirken yapacağım bir diğer yaygın şey, veri setini iki boyuta indirgemek ve ardından gözlemleri ve karar sınırını çizmektir. Bazen, performansını değerlendirirken verileri ve modelinizi görsel olarak incelemek faydalı olabilir.



4.10. Veri Gürültüsü

Makine öğrenimi veri setinde iki tür gürültü olabilir: tahmin özniteliklerinde (öznitelik gürültüsü) ve hedef öznitelikte (sınıf gürültüsü). Bir veri setinde gürültünün varlığı, öğrenme algoritmalarının performansını düşüren model karmaşıklığını ve öğrenme süresini artırabilir.

4.11. Kayıp Veri

Eğer veride bazı örneklerin bazı özellikleri kayıpsa izlenecek iki yol vardır:

- Kayıp özelliklere sahip örnekler veriden tamamen çıkartılır.
- Kayıp verilerle çalısabilecek sekilde algoritma düzenlenir.

Eğer kayıplı örneklerin sayısı birinci seçenek uygulanamayacak kadar çoksa ikinci seçenek uygulanmalıdır. Kayıp bilgiye sahip özellik vektörü için kazanç hesaplanırken kayıplı örnekler hariç tutularak bilgi kazancı normal sekilde hesaplanır ve daha sonra F katsayısıyla çarpılır. F, kayıpsız verinin tamamına oranıdır.

$IG(X) = F.(H(X) - H(V, X))$.

Kayıp bilgiye sahip özellik vektörü içinde en sık tekrarlanan değerin kayıp bilgi yerine yazılması da önerilen yöntemlerdendir.

Eksik değerler ortaya çıktığında veri noktalarını kolayca atamayız. Sınıflandırılacak bir test noktası da eksik değişkenlere sahip olabilir. Sınıflandırma ağaçlarının, eksik değerlerini tamamlamanın güzel bir yolu vardır. Sınıflandırma ağaçları, bir yedek ayrım bularak sorunu çözer. Başka bir değişkene dayalı başka bir ayrım bulmak için, sınıflandırma ağaçları diğer tüm değişkenleri kullanarak tüm bölünmelere bakar ve optimum bölünmeye en çok benzeyen eğitim veri noktaları bölümünü veren birini arar. En iyi bölünmenin sonucunu tahmin etmeye çalışırlar.

4.12. Maliyet hesabı

Zaman
Bellek
Hassasiyet
Karşılaştırma
Aşırı uyum
Korolasyon

4.13. Aşırı Uyum

Makine öğreniminde, istatistiksel bir model, temelde yatan ilişki yerine rastgele hata veya gürültüyü tanımladığında "aşırı uyum" ortaya çıkar. Bir model aşırı derecede karmaşık olduğunda, eğitim veri türlerinin sayısına göre çok fazla parametrenin olması nedeniyle normal olarak aşırı uyum gözlemlenir. Model, aşırı uyumlu olan zayıf performans sergiliyor.

Modeli eğitmek için kullanılan kriterler, bir modelin etkinliğini değerlendirmek için kullanılan kriterlerle aynı olmadığından, aşırı uyum olasılığı oluşur.

Çok fazla veri kullanarak aşırı uydurma önlenebilir, küçük bir veri kümeniz olduğundan ve siz ondan öğrenmeye çalıştığınız için aşırı uydurma nispeten gerçekleşir. Ancak küçük bir veritabanınız varsa ve buna dayalı bir modelle gelmek zorunda kalırsanız. Böyle bir durumda çapraz doğrulama olarak bilinen bir teknik kullanabilirsiniz. Bu yöntemde veri kümesi, test ve eğitim veri kümeleri olmak üzere iki bölüme ayrılır, test veri kümesi yalnızca modeli test ederken, eğitim veri kümesinde veri noktaları modelle birlikte gelir.

Bu teknikte, bir modele genellikle eğitimin (eğitim veri seti) çalıştırıldığı bilinen bir veri kümesi ve modelin test edildiği bilinmeyen verilerden oluşan bir veri kümesi verilir. Çapraz doğrulama fikri, eğitim aşamasında modeli "test etmek" için bir veri kümesi tanımlamaktır.

5. Yararlanılan Kaynaklar

1. Machine Learning (ML) Algorithms For Beginners with Code Examples in Python. https://medium.com/towards-artificial-intelligence/machine-learning-algorithms-for-beginners-with-python-code-examples-ml-19c6afd60daa
2. https://www.w3schools.com/python/python_ml_getting_started.asp
3. https://www.tutorialspoint.com/machine_learning_with_python/knn_algorithm_finding_nearest_neighbors.htm
4. https://brilliant.org/wiki/feature-vector/
5. https://stanford.edu/~shervine/l/tr/teaching/cs-229/
6. https://stanford.edu/~shervine/l/tr/teaching/cs-221/
7. https://stanford.edu/~shervine/l/tr/teaching/cs-230/