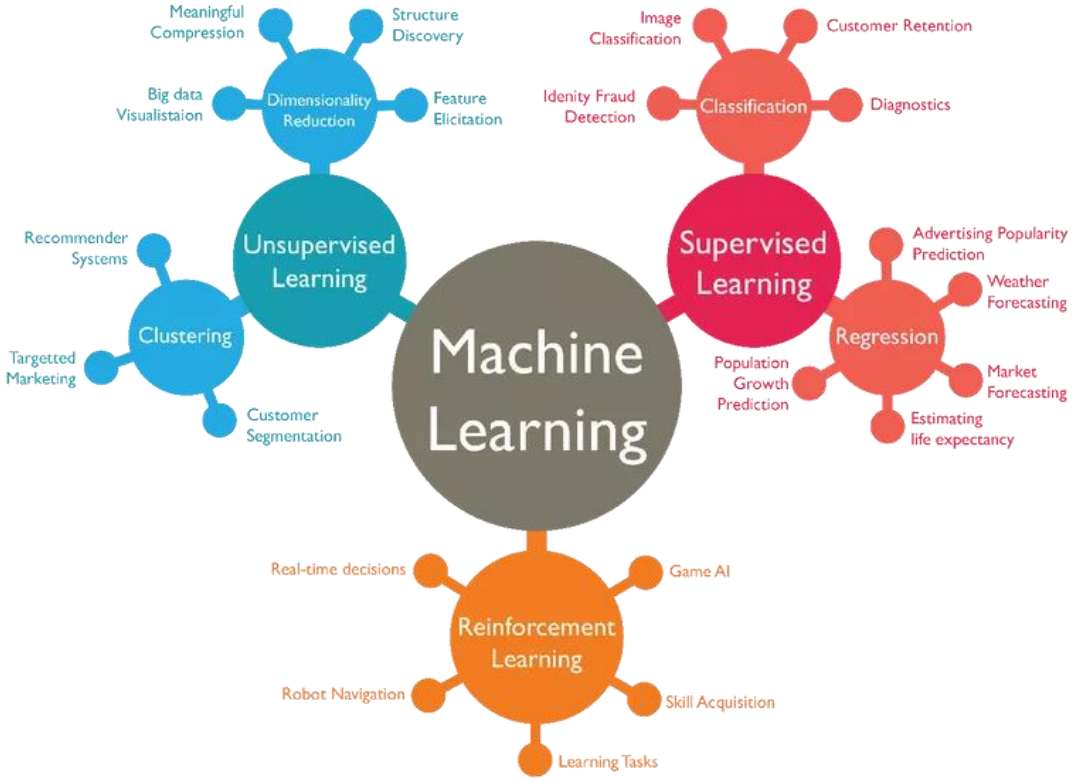
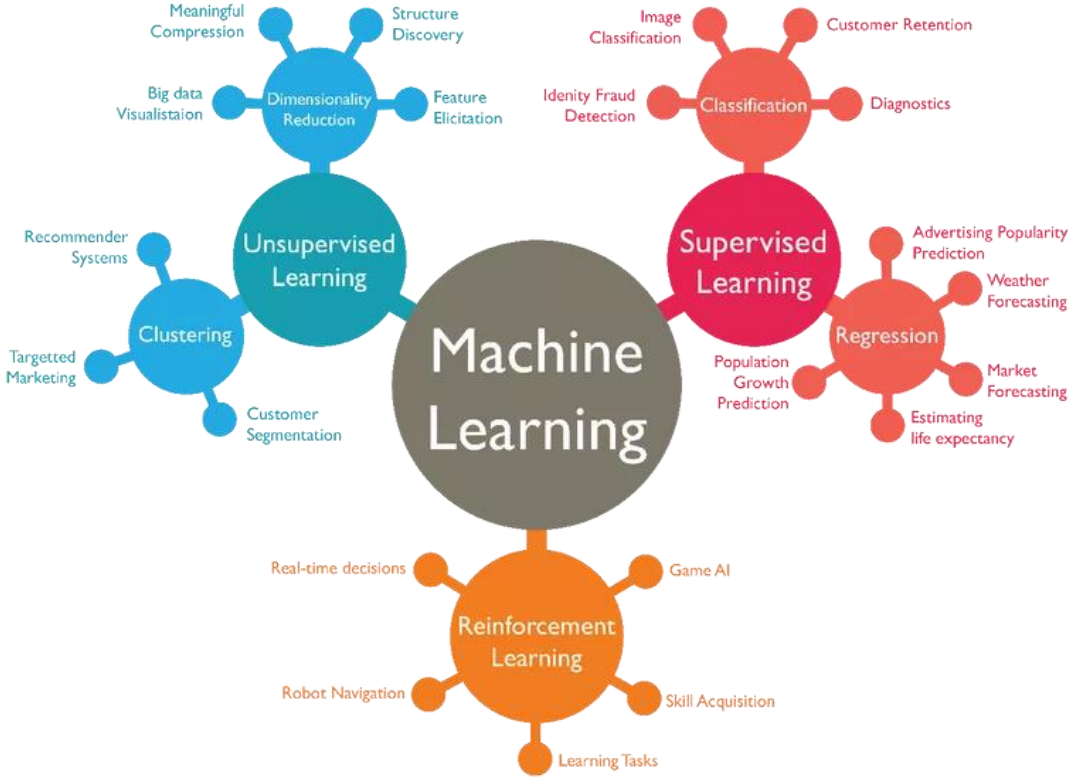
Makine öğrenmesi algoritmaları genellikle üç şekilde sınıflandırılır.

-Gözetimli Öğrenme(Supervised Learning)
-Gözetimsiz Öğrenme (Unsupervised Learning)
-Takviyeli Öğrenme (Reinforcement Learning)





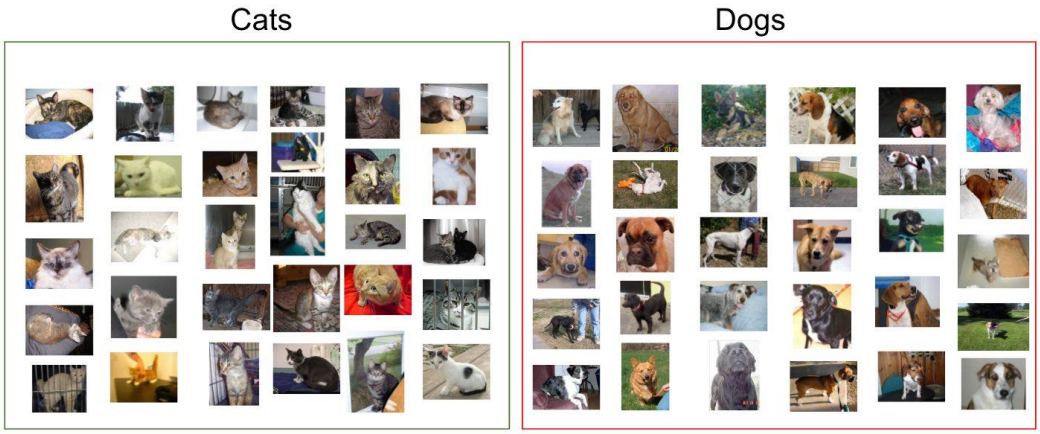
Gözetimli Öğrenme (Supervised Learning)

Bu sınıftaki algoritmalar, öğrendiklerinden yola çıkarak tahminleme yapmak için etiketli (labeled) verileri kullanır.
Yani eğitimde kullanılacak veri ve veriye ait sınıflar (kategoriler/etiketler) önceden bilinir. Bu bilgi ile sistem öğrenir ve yeni gelen datayı bu öğrendikleriyle yorumlar.

Gözetimli öğrenmedeki en zaman alıcı aşama eğitim verisinin hazırlanması aşamasıdır. Titizlikle hazırlanmamış bir eğitim verisi ile eğitilmiş sistem kötü tahminler yapacaktır.

Gözetimli Öğrenmeyi bir örnek ile anlatalım,
Bir sistem tasarlayacağınızı varsayalım. Bu sistem sizin gösterdiğiniz resimde kedi veya köpek olup olmadığını size söyleyecek. Böyle bir sistem için gözetimli öğrenme kategorisindeki makine öğrenmesi algorima(larına)sına ihtiyaç duyabilirsiniz.

Bu tarz algoritmaların çalışma prensibi bu örnekte şöyle olacaktır;
Resimde bir kedi olduğunu sistemin tahmin etmesini istiyorsak, sisteme önce bir miktar kedi resmi içeren veri seti vermemiz gerekiyor.
Bu veri setine eğitim verisi adı veriliyor. (train data)



Sample of cats & dogs images from Kaggle Dataset

Bu eğitim versindeki fotoğrafların “kedi fotoğrafı” olduğunu eğitim verisinde belirtmemiz yani veriyi etiketlememiz (label) gerekiyor. Bu eğitim verisi ile sistem/algoritma eğitiliyor. Aynı işlem köpek resimleri için de yapılıyor.

Sistemi eğitmek için kullandığımız veriden farklı bir veri seti daha oluşturuyoruz (test data), bu veri setinin içine bazı kedi ve köpek resimleri ekliyoruz fakat bu sefer hangilerinin kedi hangilerinin köpek resmi olduğunu söylemiyoruz. Bu test verisi ile sistemi test ediyoruz.
Bu sayede sistem, yeni resmin içindeki hayvanın kedi veya köpek olup olmadığını tahmin edebiliyor.

Bu örnekte olduğu gibi bir eğitim veri seti ile eğittiğimiz makine öğrenmesi yöntemine gözetimli öğrenme yöntemi deniyor.

Gözetimli Öğrenme yöntemleri iki grupta incelenir;
– *Sınıflandırma Yöntemi (Classification Method)*
– *Regresyon Yöntemi (Regression Method)*

Sınıflandırma Yöntemi (Classification Method)

Sınıflandırma problemi için geliştirilen algoritmalar, adından da anlaşılacağı üzere verileri belli özelliklerine göre sınıflandırırlar.

Sınıflandırma yapısal (structure) veya yapısal olmayan (unstructure) veriler üzerinde yapılabilir.

Eğer sistem, hangi verinin, hangi koşullarda, hangi sınıfa ait olacağı bilgisi ile sınıflandırılarak eğitilirse, yeni veri setindeki veriyi de öğrendiklerine benzer biçimde sınıflandırabilir.

Yukarıdaki kedi-köpek resimleri örneği bir sınıflandırma problemine örnektir. Orada da kedi resimleri, kedi sınıfı ile ilişkilendirilmiş sorasında yeni resimdeki figürün o sınıfa ait olup olmadığının tahmin etmesi beklenmiştir.

Bir diğer örnek,
0-17 yaş aralığındaki kişileri çocuk,
18-25 yaş aralığındaki kişileri genç,
26 yaş ve üstündeki kişileri yetişkin sınıfıyla sınıflandırmak,
bu yöntemi gözünüzde canlandırabilmeniz adına bir diğer basit örnek olarak kabul edilebilir. Ancak gerçek hayatta bu yöntemlerin kullanıldığı problemlerin çok daha karmaşık olduğu bilinmelidir.

Sınıflandırma Yöntemleri de aşağıdaki gibi kategorize edilir;
–**Binary Classification (İkili Sınıflandırma):** İki olası sonuç ile sınıflandırma. Örn: Cinsiyet sınıflandırması (Erkek / Kadın)

–**Multi Class Classification (Çoklu Sınıf Sınıflandırma):** İkiden fazla sınıfı sınıflandırma. Bir sınıfa ait birden fazla farklı veri varsa bu farklı veriler tespit edilir ve her biri tek bir etikete atanır. Örn: Bir hayvan sınıfında kedi ya da köpek olabilir ancak ikisi birlikte bir sınıfta olamaz kendi içinde sınıflara bölünmelidir.

–**Multi Label Classification (Çoklu Etiket Sınıflandırma):** Bir veri birden fazla sınıfla ilişkilendirilebilir. Örn: Bir makale hem sağlık hem spor hem de insan ile ilgili olabilir.

Sınıflandırma yönteminde en çok kullanılan algoritmalar;

- **Naive Bayes :** Verileri olasılık ilkeleri ile hasaplayarak sınıflandıran bir sınıflandırma algoritmasıdır. Basit bir ifadeyle, bir Naive Bayes sınıflandırıcı, bir sınıftaki belirli bir özelliğin varlığının başka herhangi bir özelliğin varlığına bağlı olmadığını varsayar. Örneğin, bir meyve kırmızı, yuvarlak ve çapı yaklaşık 3 inç ise bir elma olarak düşünülebilir. Bu özellikler birbirlerine veya diğer özelliklerin varlığına bağlı olsa bile, bu özelliklerin tümü, bu meyvenin bir elma olması olasılığına bağımsız olarak katkıda bulunur ve bu yüzden “Naif” olarak bilinir.
- Eğitilmiş veriler üzerinde olasılık işlemleri yapılır ve sisteme sunular yeni verinin önceki olasılık değerine göre sınıflandırılması sağlanır. Başka bir örnek ile ifade edilecek olursa: binlerce makalenin hangi alanda yazıldıklarına göre kategorize etmek istiyorsunuz. (tıp, teknoloji, edebiyat) Bunun için belli makalelerde geçen belli kelimelerin olasılık değerlerinin, diğerlerine oranla fazla olması durumuna göre o

makalenin hangi kategoriye ait olduğunu öğrenmek isterseniz(Sağlık kelimesinin çok geçtiği makale tıp ile ilgili bir makaledir gibi) bu algoritma işinize yarayabilir.

- **K-Nearest Neighbours (En Yakın Komşu) :** Bu tip sınıflandırma, her bir noktanın en yakın komşularının basit çoğunluk oyu ile hesaplanması ile elde edilen sınıflandırmadır. Veri hangi veriye en çok yakındır? mantığı ile dallanır.
  Bu algoritmanın uygulanması kolaydır, gürültülü eğitim verisine (noisy training data) dayanıklıdır ve eğitim verileri büyükse oldukça etkilidir.
- **Decision Tree (Karar Ağacı) :** Veriler, sınıfları ile birlikte bu algoritmaya verildiğinde, algoritma verileri sınıflandırmak için kullanılabilecek bir dizi kural üretir. Karar düğümleri(decision node) ve yaprak düğümleri(leaf node) olan bir ağaç yapısına sahiptir. Hem sınıflandırma hem de regresyon yönteminde kullanılabilir.
- **Random Forest :** Sınıflandırma işlemi sırasında birden fazla decision-tree kullanılarak sınıflandırma değerinin yükseltilmesi hedefleyen, sınıflama veya regresyon yönteminde kullanılabilen algoritmadır.
- **Support Vector Machine (Destekçi Vektör Makinesi) :** Veri setinde birbirine benzeyen gruplar arasına birbirinden en uzak olan noktalardan sınırlar çizmeye yarayan algoritmadır.

Regresyon Yöntemi (Regression Method)

Diğer gözetimli öğrenme yöntemlerinden biri de Regresyon Yöntemidir. Regresyon problemleri, üretilen çıktının sürekli sayılardan oluştuğu durumlar için kullanılıyor. Örnek, bir çalışanın işe geldiği gün sayısı, gün içinde ürettiği ürün adedine göre ona sayısal bir verimlilik puanı oluşturmak isterseniz regresyon algoritmalarını kullanabilirsiniz. Regresyon algoritmaları gözetimli öğrenme kategorisinde olmasının yanı sıra anomaly detection (anormal durum yakalama) gibi durumlarda hem gözetimli (supervised) hem gözetimsiz (unsupervised) öğrenme yöntemleri ile kullanılabilir.

En çok kullanılan Regresyon algoritmalarına bir bakalım;

- **Linear Regression:** Sayısal girdi ve çıktılar arasındaki doğrusal ilişkiyi tespit etmeyi sağlar. Düzlemde yayılmış verinin modelini en iyi biçimde doğrusal olarak çıkartmaya çalışan yöntemdir.

- **Logistic Regression:** Bir sonucu belirleyen bir veya daha fazla bağımsız değişken bulunan veri kümesini analiz etmek için düzlemde en iyi eğriyi yakalamaya çalışan istatistiksel bir yöntemdir. Sonuç, ikiye bölünmüş bir değişkenle ölçülür (sadece iki olası sonuç vardır).

- **Multiple Linear Regression:** Birden fazla tahminleyici (predictor) değişken kullanarak tahminlemeye çalışılan doğrusal regresyonun adıdır.

- **Polynomial Regression:** Veriler arası ilişki her zaman doğrusal olmayabilir. Optimum ilişkiyi bulmak için bir eğri gerekebilir. Tıpkı polinom fonksiyonlarında olduğu gibi bu yöntemde de bir terimin karesi veya küpü(veya terimin üssü herhangi bir sayı

olabilir) alınarak doğrusal olmayan bir regresyon modeli oluşturulmak istenebilir. Bu gibi durumlarda kullanılabilen bir algoritmadır.

- **Support Vector Regression:** Algoritmayı karakterize eden tüm ana özellikleri (maksimal marjı) koruyan bir regresyon yöntemi olarak da kullanılabilir. Support Vector Machine ile aynı ilkeleri kullanır. Ana fikir, hatanın en üst düzeye çıkarıldığı hiper düzlemi bireyselleştirerek hatayı en aza indirgemek, hatanın bir kısmının tolere edildiğini göz önünde bulundurmak. Örn: Bir personelin eğitim seviyesine göre maaşını tahmin eden model geliştirmek.
- **Decision Tree:** Sınıflandırma yönteminde de kullanılan decision tree, regresyon yönteminde de aynı şekilde kullanılabilir. Bu algoritma, kök düğümden başlayarak, yukarıdan aşağıya inşa edilen node'lar (düğüm) ile verilerin, kendi içlerinde benzer değerlere (homojen) sahip olanlarının alt kümelere ayrılmasını sağlayan algoritmadır.

Gözetimsiz Öğrenme(Unsupervised Learning):

Gözetimli öğrenme yönteminin aksine herhangi bir kategorize edilmiş, etiketlenmiş eğitim verisi kullanılarak eğitilmez. Gözetimsiz öğrenme yöntemi, önceden eğitilmemiş veriler üzerinde çalışarak veriler arasında bağıntılar bulup birbirine yakın anlamda/içerikte/değerde olan verilerin kendi içinde kümelenmesi mantığıyda çalışır.

Girdi verisinin hangi sınıfa ait olduğu önceden bilinmez. Bu sınıflandırma işlemleri veriye bakılarak algoritmalar tarafından öğrenilir. Yeni gelen veriler de algoritmanın oluşturduğu gruplara uygun olarak en yakın gruba atanır.

Gözetimsiz öğrenmeye örnek verecek olursak;
Bir market işlettiğinizi varsayalım müşterilerinizin ürün satın alırkenki davranışlarını incelemek ve ona uygun şekilde ürünlerinizin stoklarını güncellemek veya raf düzenlemesi yapmak istediğinizi varsayalım.
Her gün yüzlerce müşterinizin geldiğinizi hesap edersek, bu durumda sistemi "bu satılırsa şu da satılır" şeklinde bir eğitim verisi ile eğitebilmeniz olası değildir.

Bu yüzden gözetimli öğrenme algoritmaları kullanmak yerine gözetimsiz öğrenme algoritmaları kullanarak birbiri ile yakın alışveriş alışkanlıkları olan müşteri gruplandırabilir. Çıkan gruplara göre stoklarınızı güncelleyebilir veya raf düzenlemenizi buna uygun yapabilirsiniz.

Bu ve bunun gibi örnekleri çoğaltmak mümkün.
Kısacası eğitim verisi hazırlanamayacak karmaşıklıktaki veriler için gözetimsiz öğrenme yöntemi kullanılmalıdır.

Gözetimsiz öğrenme yöntemleri genelde üç başlık altında incelenir;

- *Kümeleme (Clustering)*
- *Birliktelik Kuralı (Association Rule Mining)*
- *Boyut Azaltma (Dimensionality Reduction)*

Kümeleme (Clustering)

Kümeleme yöntemi, veri setindeki her bir verinin birbirlerine benzerlik durumlarına göre gruplara ayrılması işlemine denir.

Her bir grup birer küme anlamına gelir. Her kümede birbirine en yakın veriler olmalı ve birbiriyle benzerlik göstermeyen veriler de mümkün olduğunca farklı kümelerde olmalıdır yani kümeler arası benzerliğin az olması gerekmektedir.

Örneğin, elimizde renk kodlarından oluşan bir veri seti var. Ve biz bu veri setindeki birbirine yakın renkleri gruplamak istiyoruz.

Bu durumda şöyle bir soru ortaya çıkıyor;
Küme(bu örnek için renk sayısı) sayısını biz mi belirleyeceğiz? Yoksa algoritma kendisi mi bulacak?
Bu sorunun cevabı her ikisi de mümkün.

Kümeleme algorimaları birbirine yakın renkleri bir araya getirerek kümeler oluşturabilir. (5 küme, 10 küme vs.) Ancak biz bu renklerden sadece “kırmızı”,”yeşil”, “mavi” renklerin kümelenmelerini istiyorsak küme sayısını 3 belirtebiliriz.
Bu sayede veri setimizdeki verilerden
kırmızı alt tonlu olanlarını bir kümeye,
yeşil alt tonlu olanlarını başka bir kümeye,
mavi alt tonlu olanlarını bir başka kümeye ayırabiliriz.

Elimizdeki veriye en uygun küme sayısını belirlemek istememizdeki temel sebep;
1- Küme içindeki değerlerin birbirine en çok benzemesi
2- Kümelerinse birbirinden olabildiğince farklı olmasıdır.

Çok fazla küme ile çalışmak birbirine çok benzeyen kümelerin oluşmasına sebep olabilir. Bu yüzden optimal bir küme sayısı belirtmek gerekir. Ancak küme sayısını kolayca belirtmek her veri seti için mümkün olmaz.
Bu yüzden **K-Means Algoritması** kullanılır.
Bu algoritma, kullanacağınız veri setinde oluşturabileceğiniz optimal küme sayısını size söyleyebilir.

Birliktelik Kuralı (Association Rule Mining)

Bu kural kümeleme yönteminden farklı olarak değişkenler arası ilginç ilişkileri keşfetmek için kurallar arayan bir yöntemdir.

Veri seti içindeki geçmiş tarihli hareketlerin örüntülerini analiz eden ve birlikte gerçekleşme durumlarını çözümleyen veri madenciliği yöntemidir. Bu örüntülerden hareketle gelecekteki veriler için tahminleme yapabilir.

Birliktelik kuralı (Association Rule) algoritmaları ile sepet analizi yapabilirsiniz.

Hangi müşteri hangi ürünle birlikte neleri satın almış bunları inceleyebilirsiniz. Bu sayede geliri arttırmak ve verimi yükseltmek adına her müşteri için özel ürün tavsiyelerinde

bulunabilir, promosyonlar yapabilir, stoğunuzu bu bilgiler ile güncel tutabilir, raf düzenlemesinde bu analizden faydalanabilir ilişkili ürünleri birbirine yakın dizebilirsiniz.

Association Rule ile ilgili en çok kullanılan algoritmalardan bazıları şunlardır;
-Apriori algorithm
– Eclat algorithm
-FP-growth algorithm

Yukarıdaki algoritmaların hepsinin amacı veriler arasındaki bağıntıyı ortaya çıkartmaktır. Bu amaç için her biri kendi içinde farklı matematiksel işlemleri barındırır.

Not: Kümeleme ve Birliktelik Kuralı yöntemleri arasındaki temel fark; Kümeleme, veri noktaları ile ilgilidir, Birliktelik Kuralı ise bu veri noktalarının nitelikleri arasındaki ilişkileri bulmakla ilgilidir.

Boyut Azaltma (Dimensionality Reduction)

Boyut azaltma, amaç doğrultusunda en iyi sonucu verecek olan öznitelikler(feature) ile çalışmak için kullanışsız, gereksiz olan öznitelikleri çıkartmak, veri boyutunu azaltmak olarak nitelendirilebilir.

Öznitelik (feature): Veriye ait her bir özelliğe verilen isimdir. (Örn: ad, soyad, yaş, doğum yeri, kan grubu vs. bilgisi)

Örneğin, elimizde kişilere ait kan değerlerinin ve kişisel bilgilerin yer aldığı bir veri seti aşağıdaki gibi olsun;
Adı,
Soyadı,
Kan grubu,
Hemogram değeri,
Lökosit sayısı,
Eritrosit sayısı,
Kişinin kullandığı ilaçlar,
Kişinin saç rengi,
Göz rengi,
Ten rengi

Bir kişinin kan değerlerine bakılarak bir hastalık tespiti yapmak istiyorsak, buradaki kan değerlerinin her birine ihtiyacımız olacak.
Ancak kişinin saç, göz ve ten rengi bilgisine ihtiyacımız yok. Bu yüzden bu öznitelikleri veri setinden çıkartıp çok daha rahat analiz yapmayı sağlayabiliriz bu işlemin bir diğer adı **Feature Selection**'dır (aşağıda bir daha üzerinden geçiliyor).

Peki veri boyutu çok büyüdüğünde, gereksiz öznitelikleri nasıl tespit edeceğiz?
Bunun için **PCA** (Principal Component Analysis) isminde bir algoritma var. Hangi özniteliğin çıkartılması konusunda PCA yardımcı olmaktadır.

Boyut azaltma konusu başlığı altında incelenmesi gerek diğer konular ise;
1 – Öznitelik Çıkarma ve Öznitelik Mühendisliği (Feature Extraction and Feature Engineering): Ham verilerin modelleme için uygun özniteliklere dönüştürülmesi, ham veri içinden uygun olanların ham haldeyken işleme alınmasıdır. Örneğin, bir kişinin parmak izlerini ve yüzünü tarattığı verilerden oluşan bir veri setiniz var. Siz de parmak izi verileri ile ilgili bir çalışma yapmak isiyorsunuz. O halde yüz tanıma verilerini kullanmanıza gerek yoktur. Veri ham haldeyken yüz tanıma verilerini yok sayabilirisiniz.

2 – Öznitelik Dönüşümü (Feature Transformation): Algoritmanın doğruluğunu artırmak için verilerin dönüştürülmesidir. Bir veya birden fazla öznitelik birleştirilerek elde edilebilir. Ya da var olan özniteliğe yeni bir şey eklenerek (bir sayı, bir harf vs.) yeni öznitelik yaratılabilir. Bu yöntemlere öznitelik dönüşümü adı verilir. Dönüştürülen yeni öznitelik, eski özniteliklerin özelliklerini taşır veya taşımayabilir.

3 – Öznitelik Seçimi (Feature Selection): Gereksiz özelliklerin kaldırılması (Yukarıda anlatılan kan değerlerine bağlı hastalık tespiti" öznitelik seçimine örnektir.).

Takviyeli Öğrenme (Reinforcement Learning)

Bu öğrenme biçimi diğerlerinden biraz farklıdır. Temelinde canlıların davranış psikolojisine dayandırılır. Bu yöntem öğrenme işlemini çevreden aldığı geri bildirim (feed-back) ile gerçekleştirmektedir.

Bu yöntemle olası durumların, hedef olup olmadığının kontrol edilir. Denemelerin sonucunda hedefe ulaşılamadığında ceza (penalty), ulaştığında ise ödül (reward) sinyali alınır ve sistem ceza sinyali aldığı hamleyi bir daha tekrarlamaz. Ödül sinyali aldığı deneyimden faydalanarak öğrenmeye devam eder ve hep maksimum ödülü amaçlayarak sürekli öğrenmeye işlevini sürdürür.

Bu sebeple mükemmele erişme, öğrenmeyi durdurma gibi bir durum söz konusu olmaz. Algoritma sürekli öğrenmeye devam eder.

Örneğin, 2017 yılında Google'ın geliştirdiği yapay zeka AlphaGo, Go oyunu dünya şampiyonu Ke Jie'yi yenmişti. AlphaGo'nun öğrenme mantığının arkasında Reinforcement Learning vardı.

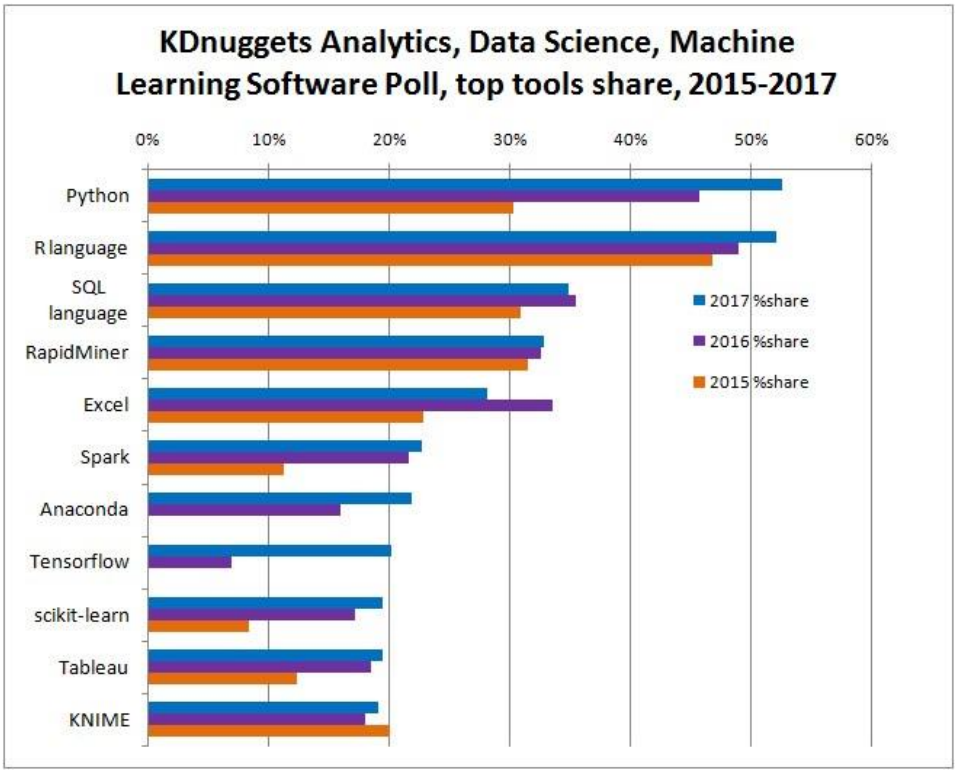
Bu yöntemde esas amaç, insan beynine ve algılarına yakın (veya çok daha iyi düzeyde) işleyen algoritmalar geliştirmektir.
Deep Learning algoritmaları takviyeli öğrenme mantığı ile geliştirilmiştir.

Nereden başlamalıyım?

Makine öğrenmesine nereden başlayacağınızı düşünüyorsanız, aşağıdaki adımları takip etmek size yardımcı olacaktır;

1- Geliştirme yapmayı istediğiniz bir programlama dili seçin
Makine öğrenmesinde sıkça kullanılan programlama dilleri arasından bir dil seçin.

Aşağıdaki grafikte 2015- 2017 yılları arasında makine öğrenmesinde kullanılan popüler dilleri göreceksiniz. Bunlardan birini seçebilirsiniz.



2015- 2017 yılları arasında makine öğrenmesinde kullanılan popüler dillerin grafiği

2- Seçtiğiniz dile özel geliştirilmiş makine öğrenmesi kütüphaneleri hakkında bilgi sahibi olun

Diyelim ki bu diller içinden Python'ı seçtiniz. Burada makine öğrenmesi için yazılmış olan kütüphaneleri ve işe yaradıklarını öğrenin.

2.1 – **Python'da kullanılan Core Kütüphaneleri:**

NumPy: NumPy, bilimsel/matematiksel/mantıksal/istatistiksel hesaplama için oluşturulmuş bir python kütüphanesidir. NumPy kullanılarak istatistik işlemleri ve simülasyonlarda yapılabilir.

SciPy: Sık kullanılan matematiksel ve fiziksel problemlerinin bilgisayar ortamında ifade edilmesine yönelik fonksiyonları barındırmaktadır.

Pandas: "etiketli" ve "ilişkisel" verilerle çalışmak üzere tasarlanmış, yapısal olmayan verinizi yapısal veritabanlarındaki gibi çalıştırmanızı sağlayan kütüphanedir. Pandas, hızlı ve kolay veri işleme, toplama ve görselleştirme için tasarlanmıştır. İki temel veri yapısına sahiptir;

Serie: tek boyutlu tablo yapısına sahiptir.
DataFrame: iki boyutludur tablo yapısına sahiptir.

2.2 – Python'da kullanılan Analiz Görüntüleme (Visualize) Kütüphaneleri:

Matplotlib: Grafik çizimi için kullanılır. Bilimsel programlamanın en önemli araçlarından birisidir. Verilerin etkileşimini görselleştirebilir ve görsel raporlar oluşturulabilinir. İki boyutlu ya da üç boyutlu grafikler üretilebilir. Grafik çeşitleri aşağıdaki gibidir.
Line plots,
Scatter plots,
Bar charts and Histograms,
Pie charts,
Stem plots,
Contour plots,
Quiver plots,
Spectrograms

Seaborn: Daha çok istatistiksel modellerin görselleştirilmesine odaklanmıştır; Bu tür görselleştirmeler, ısı haritalarını, verileri özetleyen ama yine de genel dağılımları tasvir eden grafikler için kullanılır. Seaborn, temelinde Matplotlib'e bağımlıdır.

Bokeh: İnteraktif görselleştirmeleri amaçlayan Bokeh, Matplotlib'den bağımsızdır. Data-Driven Document yani veriye dayalı dökümanların görselleştirmesini yapan yüksek özellikli görselleştirme imkanı sağlayan bir Python kütüphanesidir.

Plotly: Bu kütüphane web tabanlı görselleştirmeler oluşturabilen bir kütüphanedir.

2.3 – Python'da kullanılan Makine Öğrenmesi Kütüphaneleri:

SciKit-Learn: Scikits, görüntü işleme ve makine öğrenimi kolaylaştırma gibi belirli işlevler için tasarlanmış bir Python kütüphanesidir. Doğrusal regresyon, lojistik regresyon, karar ağaçları, rastgele orman gibi birçok temel yöntemi içerir.

Theano: Çok boyutlu diziler dahil matematik ifadelerini etkili bir şekilde tanımlamayı, en iyilemeyi ve değerlendirmeyi sağlayan bir Python kütüphanesidir. Derin öğrenme(Deep learning) için kullanılan kütüphanedir.

TensorFlow: Google'ın çıkarttığı bir makine öğrenmesi kütüphanesidir. Açık kaynak kodludur. Python ile geliştirilebilinir. Derin öğrenme için kullanılan bir kütüphanedir.

Keras: Modelleri tanımlamayı ve eğitmeyi kolaylaştırmayı sağlayan, Theano veya Tensorflow'u backend olarak kullanan ve python dilini kullanan bir wrapper.

2.4 – Python'da kullanılan Doğal Dil işleme Kütüphaneleri:

NLTK (Natural Language Toolkit): Doğal Dil İşleme için kullanılır.
NLTK, dilbilim, bilişsel bilim, yapay zeka, vb. gibi konuların öğretimini ve araştırmasını kolaylaştırmayı amaçlamıştır ve bugün bu konuya odaklanarak kullanılmaktadır.

Gensim: Ham ve yapılandırılmamış dijital metinlerle kullanılmak üzere tasarlanmıştır. Hem verimli hem de kullanımı kolaydır.

2.5 – Python'da kullanılan Veri madenciliği, İstatistik Kütüphaneleri:

Scrapy: Websitelerinden iletişim bilgileri veya URL'ler gibi yapılandırılmış verilerin alınması için, örümcek botlar olarak da bilinen tarama programları yapmak için bir kütüphanedir.

Statsmodels: İsminden de anlaşılacağı gibi bir istatistik kütüphanesidir. Kullanıcılarına istatistiksel modellerin tahmin edilmesini, çeşitli yöntemlerle istatistiksel analizlerin yapılmasını bu istatistiklere bağlı veri araştırması yapabilmeyi sağlar.

Bunlara ek olarak; **Lasagne, Caffe, Torch kütüphanelerini ve gerçek zamanlı işlemlerde kullanabileceğiniz PySpark**interface'ini de inceleyebilirisiniz.

3-Hangi Geliştirme Ortamı (IDE) kullanılmalı?

Bu ve bunun gibi kütüphaneleri çalıştırmak için bir Python geliştirme ortamı (veya hangi dili seçtiyseniz ona ait bir geliştirme ortamı) gerekecektir.
– Jupyter/IPython Notebook
– PyCharm
– Spyder
– Rodeo
– Geany gibi ortamları bu iş için rahatlıkla kullanabilirsiniz.

4- Anaconda Navigator'da bu iş için biçilmiş kaftan!

Yukarıda bahsettiğim geliştirme ortamlarını kullanmak yerine, veri bilimi için hazırlanmış, içersinde Python dilini kullanabileceğiniz ve makine öğrenmesine dair bir çok paket programı, kütüphaneyi içeren **Anaconda** isimli programı da bu linkten indirerek kurabilir ve çalışmalara başlayabilirsiniz.
Anaconda'nın içerisinde Jpyter, Spider, Vscode gibi makine öğrenmesi çalışmalarınızda size yardımcı olacak bir çok araç birlikte geliyor.

5- Bu adımları takip ederek ilk makine öğrenmesi projenizi gerçekleyebilirsiniz **ama hepsinden önce!**

Eklemek istediğim bir şey daha var, burada kullanılacak yöntem ve algoritmaların temelini anlamak için temel düzeyde matematik (Cebir, İstatistik, Kalkülüs vb.) bilginizin olması gerekmektedir. Ancak bu sizi korkutmasın, her algoritmanın çok detayına girmeden de rahatlıkla makine öğrenmesi projeleri yapabilirisiniz.

Sonuç

Yukarıda anlattığım gibi farklı şekillerde makine öğrenmesi yöntemleri, bu yöntemlere ait bir çok (buraya ekleyemediklerim dahil) algoritma ve bir çok araç mevcut.

Bunlardan hangisini ne zaman kullanacağız sorusunun yanıtı olarak şunu söylemek doğru olacaktır;

Veri madenciliği deneysel bir iştir. Veri setinize ve amacınıza bağlı olarak bu algoritmaların birini ya da bir kaçını birleştirerek kullanmanız gerekebilir. Amacınız maksimum verimliliği

sağlayacak bir sistem modeli elde etmek olmalıdır. Hangisi veya hangilerinin kombinasyonu ile en verimli sonuca ulaşacağınıza ancak deneyerek karar verebilirsiniz.

Umarım sizler için faydalı bir içerik olmuştur.