**MACHINE LEARNING**

Artifical Intelligence: İnsan davranışlarını taklit ederek oluşturulmuş sistem veya makineler diyebiliriz.

Machine Learning: Makinelere geçmiş datalar dan istatistiksel yöntemleri kullanarak öğrenme yeteneği kazandırma demektir. Structured (satır sütunlardan oluşmuş data) datalar ile uğraşırız. Küçük datalar ile daha iyi sonuç alırız.

Deep Learning: Unstructured datalar ile uğraşırız. Text, resim gibi dosyalar için. Kullanılır. Girilen metnin devamını getirir. Büyük datalar ile daha iyi sonuçlar alınır.

Traditional Programming’te baştan kurallar bellidir. Ona göre sonuç alınır.

Machine Learning’te ise kurallar baştan belli değildir, kuralı veriyi işleyerek kendisi bulur. Örneğin; 3, 5 ve 15i veririz. Çarpma ilişkisini kendisi bulur.

2020 den 2021 e geçişte yapay zeka kullanımı birçok alanda büyük oranda artmıştır.

Satır🡪observation, sütun🡪 features (independent variable (X)) , label🡪 target (dependent variables (y))

Machine Learning Te 3 ana kategori vardır:

1. Supervised Learning: Data da label varsa, labellenmiş gözlemlerden öğrenme sürecine denir.
2. Unsupervised Learning: Makinenin unlabeled datadan öğrendiği durumdur. Örneğin, Data analysis bölümünde yaptıklarımızı makineye otomatik yaptırmış olacağız. Unsupervised learning te label yoktur.

**Examples of supervised machine learning include:**

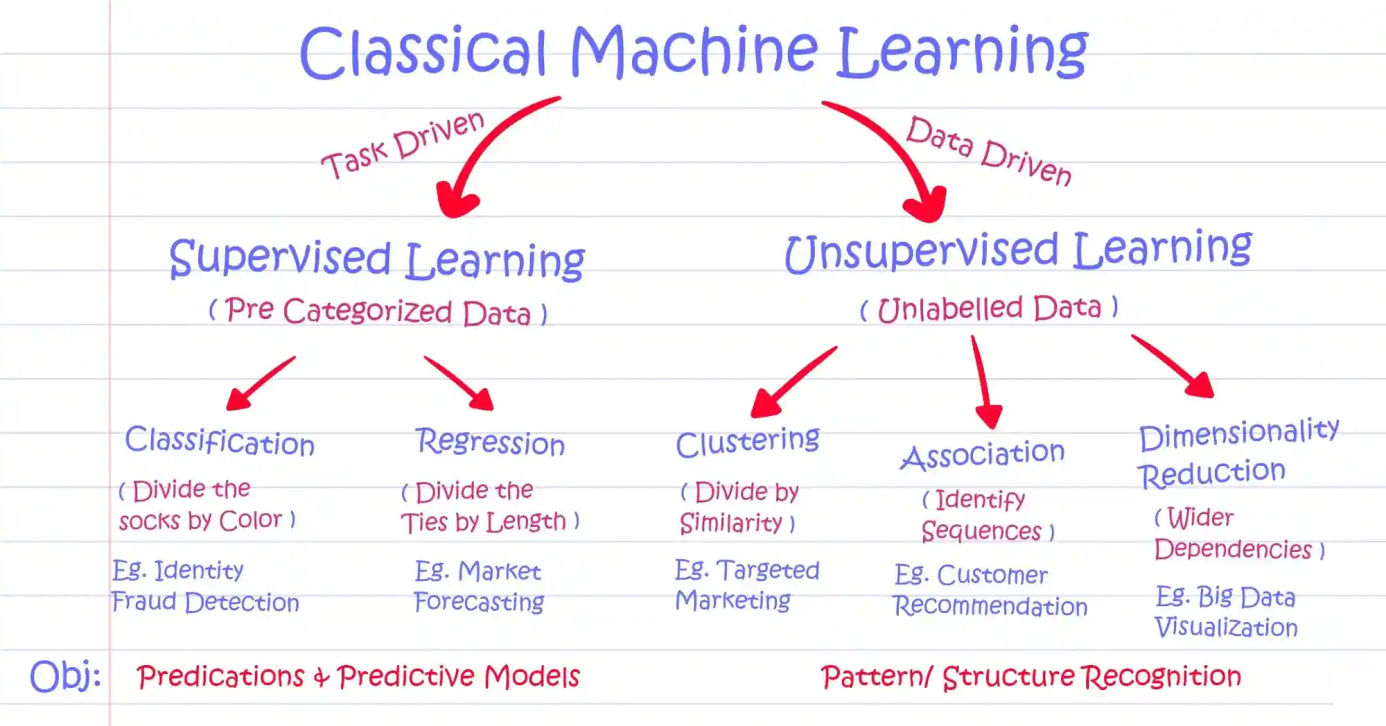
* Classification, identifying input data as part of a learned group.
* Regression, predicting outcomes from continuously changing data.

**Examples of unsupervised machine learning include:**

* Clustering, grouping together data points with similar data.
* Association, understanding how certain data features connect with other features.

Linear Regressions korelasyon ile ilişkilidir. Korelasyon iki nicel değişkenin arasındaki **ilişkinin yönü** ve **gücü** demektir. Korelasyon -1 ve +1 arasında değişir, r ile gösterilir.

Distance based modellerde scale etme çok daha önemlidir. Outlier ler varsa robust scaler daha etkili olabilir. Yine de hepsini deneyerek karar vermek daha doğru bir yaklaşım olacaktır. Distance-based modellerde veriler belirli bir aralıkta, yani bir intizam içinde ise scaling yapmadan daha iyi sonuç alınabilmektedir.



Correlation bize ilişkiyi açıklayan denklemi vermez, Regression Line ise bu ilişkiyi açıklar. (ÖNEMLİ)

Regression da ilişki line ile ifade edilir ( bağımlı ve bağımsız değişken arasındaki ilişkiyi açıklar), Correlation da ise noktalar ile ifade edilir.

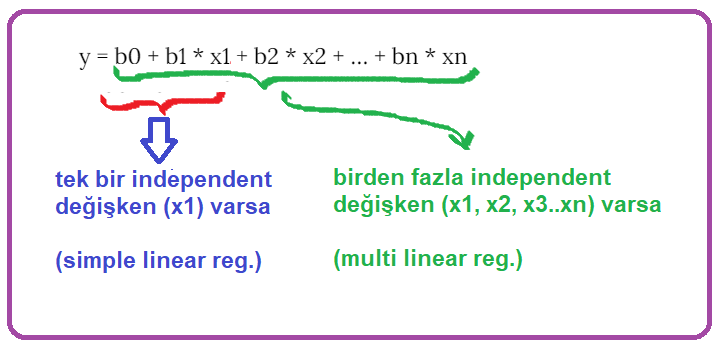
Linear regression dan söz edebilmmek için güçlü bir ilişki olması gerekir!

Tek future varsa **simple regression line**, çoklu future varsa **multi regression line** diyeceğiz.

Linear regression line da; Ortaya çıkan **model**, değişkenler arasındaki ilişkiyi açıklayan **matematiksel bir denklemdir.**Bu denklemi kullanan modelleme ile ne yapıyoruz? Bağımsız değişkenlerin değerlerine dayalı olarak bağımlı değişkenin değeri hakkında tahminlerde bulunmak için kullanıyoruz.

**Regression line** hatayı minimize edecek şekilde çizilir**. Line çizerken en küçük kareler yöntemi kullanır** bu süreçte. (Least Square of errors. Error = yi –yi^)

Birden fazla değişken (future) varsa multi **linear bir durum varsa Gradient Descent algoritması** ile regression line çizilir, bunu yaparken mean squared error kullanılır.



**Best fitting line;** data setindeki dağılımı mümkün olduğunca iyi bir şekilde özetlemelidir. Bunu lineer bir fonksiyonunun en uygun parametrelerini hesaplayarak yapar. Bu parametreler, eğim (**slope**) ve **intercept**olarak ifade edilir ve **en küçük kareler yöntemi** ile hesaplanır.**Gradient descent**yöntemi ise doğrunun en uygun eğim ve intercept'i hesaplamak için **iteratif**bir yaklaşım kullanır. Başlangıçta rastgele değerler atanır ve ardından hatanın azaltılması için **parametreler yavaş yavaş güncellenir**. Bu işlem, **hatanın minimum olduğu noktaya kadar devam eder.**Gradient descent yöntemi, büyük veri kümeleri için daha hızlı hesaplama yapabilme avantajına sahiptir. Ancak, doğru **hiperparametrelerin**seçimi ve **konverjansın**sağlanması için daha fazla dikkat gerektirir.**Konverjans**, bu güncelleme işleminin yavaşlaması veya durmasıyla gerçekleşir. Algoritmanın **konverjans sağladığı noktada, parametreler artık çok az değişiyor veya değişmiyor ve hedef değere yakın bir değere ulaşmıştır.**

Modelin başarısını ölçen

The coefficient of Determination – R^2: Bağımsız değişkenlerden tahmin edilebilen bağımlı değişkendeki değişimin oranı şeklinde tanımlanabilir.

y = b+aX deki X y’yi ne şekilde etkiliyor gibi düşünebiliriz.

R^2, 0 ile 1 arasında değer alır. Eğer bu değer 0 çıkarsa X ler y yi açıklayamıyor diyebiliriz. R^2 = 1 ise bağımsız değişkenler bağımlı değişkeni %100 oranında açıklıyor diyebiliriz. R^2 = 0.7 ise bağımsız değişkenler bağımlı değişkeni %70 oranında açıklıyor diyebiliriz.

Burada Gradient Descent'te güzel bir örnek var. Modelin öğrenme işlemini anlamada kolaylık sağlayacağını düşünüyorum.

**GradientDescent’eÖrnek;**  
Gradient descent’i gerçek bir problemmiş gibi düşünecek olursak;  
Bir dağın zirvesinden aşağı deniz seviyesine inmeniz gerektiğini hayal edin fakat görme yeteneğiniz çok az ve etraf da karanlık bu durumda aşağı nasıl inerdiniz?Öncelikle küçük adımlarla ilerlemeye gayret gösteririz, çok hızlı inmek tehlikeli olabilir ayağınız kayıp düşebilirsiniz amacınız genellikle küçük adımlarla inmek olur.Gradient Descent’te ise durum buna çok benzerdir, bir problemin çözümü için en uygun noktayı bulmak bunun içinse küçük adımlarla ilerleyerek hedefe varmaktır. Bu örnekte attığınız adım sayısı iterasyonun sayısına denk gelir. Aşağı inmek için kaç adım attınız? Hedefe ulaşmak için kaç iterasyona ihtiyacınız oldu ?Adımların büyüklüğü gradient descentte ki learning rate’e yani öğrenme oranına denk gelmektedir. Dağdan aşağıya doğru indiğiniz sırada belki 10-15 adım sonrasında daha rahat adım atmaya başladınız bunun nedeni artık bir sonraki adımınız için bir tahminleme yeteneği kazandınız 10 adımda az çok öğrenme gerçekleşti aslında.İşte gradient descentte de öğrenme işlemi bu şekilde gerçekleşmektedir. Birkaç adım iterasyona girdikten sonra öğrenme ve optimizasyon işlemleri yapılmaya başlanmaktadır.Bu örnek , Gradient Descent algoritmasını somut olarak düşünüp makine öğrenmesinde kullanılan bu algoritma ile optimize edilmeye çalışılan modelde bize en doğru sonucu verecek olan parametreleri bulmamız konusunda daha anlaşılırlık sağlıyor.

Not: Standart sapma mean dan büyükse outlier durum olabilir diye düşünürüz.

Not: Corelasyonun karesi, R^2 (Coefficient of determination) u verir nize simple regressionda.

Not: **Residuals**in a statistical or machine learning model are the **differences between observed and predicted values of data**. They are a diagnostic measure used when assessing the quality of a model. They are also known as **errors**. (edited). Residual: Tahmin edilen ile gerçek değer arasındaki farktır.

Train seti üzerinde residual toplamları daima 0 olur.

R^2 modelin başarı yüzdesini verir bize.

2.

**Residuals: (y-y^)**

🡪Linear regression modelinde hataların bağımsız olması ve normal dağılım sergilemesi yani pattern oluşturmaması gerekiyor.

🡪Orijin etrafında simetriklik olup orijinden uzakta olan yerlerde simetriklik olmaz. -> Homoscedasticity (costant variance)

🡪 Non linear data -> heteroscedasticity

🡪Varyans sabit ise pattern oluşturmamış oluyor.

**Regression Error Metrics:**

Daha önceden R^2 metriğimi görmüştük. Şimdi 3 metrik daha ekleyeceğiz.

Mean Absolute Error (MAE): Nu metrik hataları çok göz önüne çıkartmıyor. Hataları cezalandırmıyor. Model çok hata yaptı mı, yapmadı mı çok ortaya çıkarmıyor. ☹

Mean Squared Error (MSE): Hataların varyansı diyebiliriz. Küçük hataları çok fena cezalandırıyor. Hataları büyütüp, çok fazla göz önüne getiriyor. ☹

Root Mean Square Error (RMSE): Hataların standart sapması diyebiliriz. Hataları cezalandrıyor, küçük hataları ödüllendiriyor. RMSE’de önce karesini alıp sonra karekökünü aldığı için seviyeyi MAE ile eşitlemiş oluyor. İDEAL METRİK. 😊

->İyi bir modelleme yaptıysanız, MAE ile RMSE birbirine yakın olması gerekir.

-> MAE ile RMSE arasında fark büyük ise modeliznizde harbi büyük hatalar vardır.

Bu metrikleri modellerimizin performanslarını değerlendirmek için kullanacağız.

MSE genelde iki regression modelini karşılaştırmada kullanılır hataları insanın gözüne soktuğu için. Yani iki modeli birbirinden ayırmaksa amacımız, MSE bunun için biçilmiş kaftan

**Hangi lineer regresyonu ne zaman kullanırız?**

* Amaç bağımsız değişkenlerin değerlerine bağlı olarak bağımlı değişkenin (target) değeri hakkında tahmin yapmaksa, simple (basit) veya multi linear regresyon uygun olabilir.
* Fakat dikkate alınması gereken bir diğer faktör; bağımlı ve bağımsız değişkenler arasındaki ilişkinin gücüdür. İlişki güçlü ve lineer (doğrusal) ise, simple veya multi linear regresyon uygun olabilir. Fakat ilişki daha karmaşıksa, **polinom regresyon**daha iyi bir seçim olacaktır.
* Ayrıca; verileriniz tek bağımsız değişkenli bir veri ise simple linear regresyon, birden çok bağımsız değişkenli veriler için multi linear regresyon kullanılır.
* Bunların dışında; iki regresyon daha var. **Polinom regresyon**, düz bir çizgiden daha karmaşık bir ilişkiyi (eğri büğrü) izleyen veriler için uygundur ve **lojistik regresyon**, *ikili sonucu*olan veriler için uygundur.

**Scikit Learn Library**

Scikit Learn bir machine learning kütüphanesidir. Modeli eğitmekte bu kütüphaneyi kullanırız. Kullanacağımız modelleri bu küütphaneden import edeceğiz. Şöyle import edeceğiz:

conda install scikit-learn

pip install scikit-learn (anaconda kullanıldığında buna gerek yok)

**Random state** için bir sayı belirliyoruz. Genelde 42 seçilir. Ama farklı da olabilir. Her çalıştırdığınızda aynı dataları train ve test datasına ayırsın ve böylece aynı model sonuçlarını görelim diye kullanıyoruz. pandas taki seed metodu ile aynı işi yapıyor gibi düşünebilirsiniz. Bir nevi random durumu sabitliyorsunuz.

Model eğitim işlemi sırasında, verilerinizi kullanarak nesneyi "**fit**" etmeniz gerekiyor, yani **verilerinizi modele uydurmanız** gerekiyor. Bu nedenle, **LinearRegression() sınıfı**, modeli oluşturmak ve eğitmek için kullanılan bir araçtır. Ancak bu işlemleri gerçekleştirmek için **bir nesne üzerinden çağrılması**gerekmektedir.  
Bu nedenle LinearRegression() sınıfını bir değişkene atamak suretiyle bir nesne oluşturuyoruz. Ve işlemleri hep o nesne üzerinden yürütüyoruz.

**Bir modelde yuzde kac hataya kadar iyidir diyebiliyoruz?**

Datadan dataya değişir. Problemden probleme değişir. Standart bir değeri yoktur. problem vardır 1 birim büyük hatadır, problem vardır 1000 normaldir.

**R2 score:**

* Regresyon **modelinin verileri ne kadar iyi açıkladığını**ölçmek için kullanılan bir istatistiksel ölçüttür.
* R2 score, gerçek değerlerin değişkenliği ile tahmin edilen değerlerin değişkenliği arasındaki oranı ifade eder.
* 0 ile 1 arasında değer alır.
* R2 score 1'e yaklaştıkça, model gerçek verileri daha iyi açıklıyor demektir.
* Eğer R2 score değeri 1 ise, model tüm varyasyonu açıklar, yani gerçek veriler ile tahmini veriler arasında tam bir uyum vardır. Ancak, R2 score değeri 0'a yaklaştıkça, model gerçek verileri açıklamada başarısız olur.

R2 score, bir regresyon modelinin performansını ölçmek için kullanılan en yaygın ölçümlerden biridir.

R2 score değeri, modelin doğru şekilde yapılandırılıp yapılandırılmadığını, kullanılan değişkenlerin uygun olup olmadığını ve outlierların varlığı gibi faktörleri değerlendirmeye yardımcı olur.

R2 score değeri, farklı regresyon modelleri arasındaki performans farklarını da karşılaştırmak için kullanılabilir. Ancak, R2 score'nun yalnızca bir regresyon modelinin performansını ölçmek için kullanılabileceği unutulmamalıdır ve modelin diğer özellikleri de göz önünde bulundurulmalıdır.

**R2 Score negatif çıkar mı?**

R2 skoru, bir regresyon modelinin ne kadar iyi uyduğunu gösteren bir ölçüttür. R2 skoru, 1'e yaklaştıkça modelin iyiliği artar ve 0'a yaklaştıkça modelin kötüleşir. Negatif bir R2 skoru, modelin gerçek değerlerden daha kötü tahminler yaptığını gösterir ve bu, gerçek hayatta nadiren kullanışlı bir durumdur.

Bir örnek vermek gerekirse; bir ev fiyatlarını tahmin etmek için bir regresyon modeli oluşturduğunuzu varsayalım. Modeliniz, ev fiyatlarını tahmin etmek için çeşitli faktörleri kullanır, örneğin evin konumu, büyüklüğü ve özellikleri gibi. Ancak modeliniz yanlışlıkla tüm girdileri tersine çevirir ve tahminler yapar. Yani, modeliniz büyük evlerin ucuz olduğunu, küçük evlerin pahalı olduğunu ve iyi bir konumdaki evlerin kötü olduğunu tahmin eder. Bu durumda, R2 skoru negatif olacaktır çünkü modeliniz gerçek değerlerden çok daha kötü tahminler yapıyor.

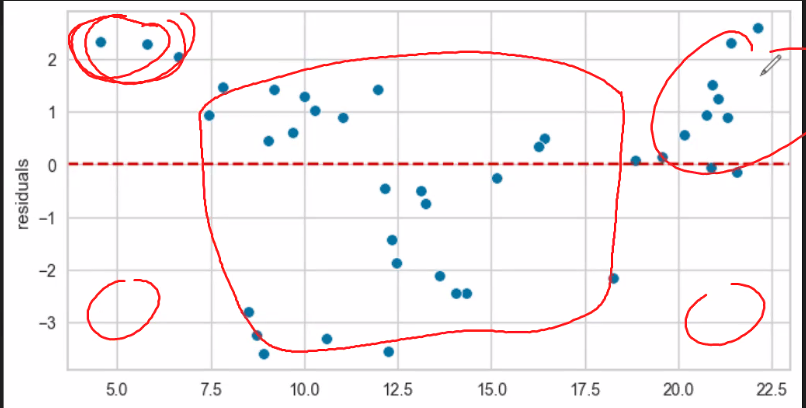
Tabii ki, bu örnek gerçek hayatta nadiren karşılaşılacak bir durumdur ve genellikle negatif R2 skorları, modelin doğru olmayan bir şekilde uygulanmasından veya veri setinin yanlış şekilde işlenmesinden kaynaklanır. Bu nedenle, modelinizin sonuçlarını yorumlamadan önce, özellikle negatif bir R2 skoru gibi anormal sonuçları dikkate almanız gerektiğini unutmayın

**Regresyon modelindeki tv radio newspaper in katsayilarinin istatistiksel anlamliligina bakmamiz gerekiyor mu?**

İstatistik ile ML farklı. Regresyon aynı ama biz datayı açıklamaya çalışmıyoruz. ML için prediction önemli

**İki uctaki degerlerde karsilik olmadigindan outliera yakin degerlerde hata var diyebilir miyiz?**

Evet diyebiliriz.



Sol ve sağ üstteki uç değerlerin line’ın altında karşılıkları olmadığından o değerlerin outlier olduğunu söyleyebiliriz.

**Residualler uzerinde shapiro testi vb uygulanabilir mi yani gerek var mı?**

Gerekli değil. Siz predictionlar ve metriclere yoğunlaşın

**regresyona ne demiştik x in y e etkisini açıklar demiştik değil mi? x bir birim artartsa y coef kadar artar..**

Session3

Bias-Variance Trade-Off

**Bias**, modelin gerçek değerden ne kadar sapma gösterdiğini ifade eder.**Yüksek bias, modelin aşırı basit olduğu ve verilerin karmaşıklığını yeterince yansıtamadığı anlamına gelir**.**Buna karşılık, düşük bias, modelin verileri iyi yansıttığı anlamına gelir.**

Overfitting: modelin eğitim verilerine **aşırı uyduğu** ve **farklı veri kümesinde kötü performans gösterebileceği**anlamına geliyor.

Underfitting🡪 High training error and high test error. Basit modellerde daha çok karşımıza çıkar. Hem classification da hem macihne learning te karşımıza çıkabilmektedir.

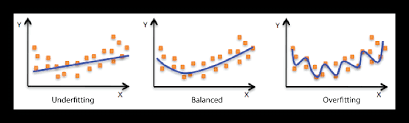
Overfitting 🡪 Low training error and high test error

total error = bias + variance + irreducible error  
formulasyonunda ie değiştirilemezdir, totral erorde sabit olduğundan bias artarsa varyans düşer, bias azalırsa varyans  artar

**Varyans**, modelin aynı veri kümesindeki farklı örnekler üzerinde nasıl performans gösterdiğini ifade eder.Yüksek varyans, **modelin eğitim verilerine aşırı uyduğu ve farklı veri kümesinde kötü performans gösterebileceği anlamına gelir.**Düşük varyans, **modelin farklı veri kümesinde iyi performans gösterebileceği anlamına gelir.**

**Bias-variance trade off**(*Bias-Varyans Dengelemesi)*, bu iki faktör arasındaki dengeyi bulmayı amaçlıyor.Modelin yüksek bias ve düşük varyansı olması,**aşırı basit**bir modeli ifade ederken,düşük bias ve yüksek varyans**, aşırı karmaşık**bir modeli ifade eder.

İdeal durumda, modelin hem düşük bias hem de düşük varyansı olmalıdır. Fakat gerçek hayattaki datalarda genellikle bu mümkün olamayacağından modeller arasında bir seçim yapmak gerekebilir. Bu nedenle, model seçiminde **bias-variance trade off**udikkate alıyoruz.



Underfitting ile mücadele: Daha komplex bir model bulmalı. Trade-off noktasına getirmeye çalışıyoruz.

Overfitting ile mücadele: Train datasına bir miktar hata ekleyerek train data score ile test data score arasındaki farkı azaltma yoluna gidiyoruz.

Daha fazla data ile train etme de bir diğer yoldur. Eğer complexiteyi azaltamıyor isek bu yolu tercih ederiz. Trade-off noktasına getirmeye çalışıyoruz.

Model complexity future sayısı ile orantılıdır. Ayrıca katsayılar (parametreler) ile alakalıdır. Parametrelerin büyüklüğü ve sayısı da etkendir.

Polynomial Regression

Y= c+ ax+bx^2 gibi birden fazla future var ve hangi futurein etkisi daha büyükse onları yansıtma imkanı verir.

Bağımsız değişken x ile bağımlı değişken y arasındaki ilişkinin n’ inci dereceden bir polinom olarak modellendiği bir regresyon türüdür. Doğrusal regresyon modellerinden farklı olarak polinom regresyon modelleri matematiksel dönüşümler yoluyla yapay olarak yeni özellikler oluşturarak verilerdeki karmaşık doğrusal olmayan ilişkileri yakalar.

Trades-off variance derecesini yakalayacak polinom derecesini belirleyebilmek önemlidir.



Aslında tüm mesele, modelin train datası ile eğitiminde öyle bir ayrıntı seviyesinde kalmalı ki eğitim sonunda dışardan gelecek her türlü gerçek datayı da kabul edilebilir optimum bir hata ile tanımlayabilmeli.

Polinomiyal regresyon, datasetindeki değişkenlerin arasındaki karmaşık ilişkileri keşfetmek için kullanılıyor.

**NOT:**

AYNI MODELE göre(linear) train data ve test data metriklerini karşılaştırdığımızda fark büyük ise high variance deriz.( VARIANCE)

FARKLI MODELLERE göre (lineer ve polinomial gibi) test metriklerini karşılaştırdığımızda fark büyük ise high bias deriz. (BIAS).

High variance + low bias 🡪 overfitting (1. Lineer model 2. ye göre overfitting deriz)

Low variance + high bias 🡪 underfitting olduğunu söyleyebiliriz. ((1. Lineer model 2. ye göre underfitting deriz).

Polinomial regresyon model ile sentetik veri üretiliyor gibi oluyor ve alakasız bir datadan bile predict yapabiliyor. Açıklaması zordur. Arabanın lastik basıncından bile fiyatını tahmin edebilir. O nedenle projelerde kullanılması tavsiye edilmez.



Not:

y-test, test datasındaki y değerleri (gerçek değerler)  
y-pred, eğitilen modelin predict ettiği y değeri. Bu iki değer arasında bir miktar hata olmasını bekleriz.

şimdi actual (gerçek) değerler ile pred (predict edilmiş) değerleri ilgili fonksiyonlara girip mae, mse, rmse ve R2\_score'u bulacağız.

**ML AŞAMALARI**

Data clean

Gerekirse scale /encoding yapılır.

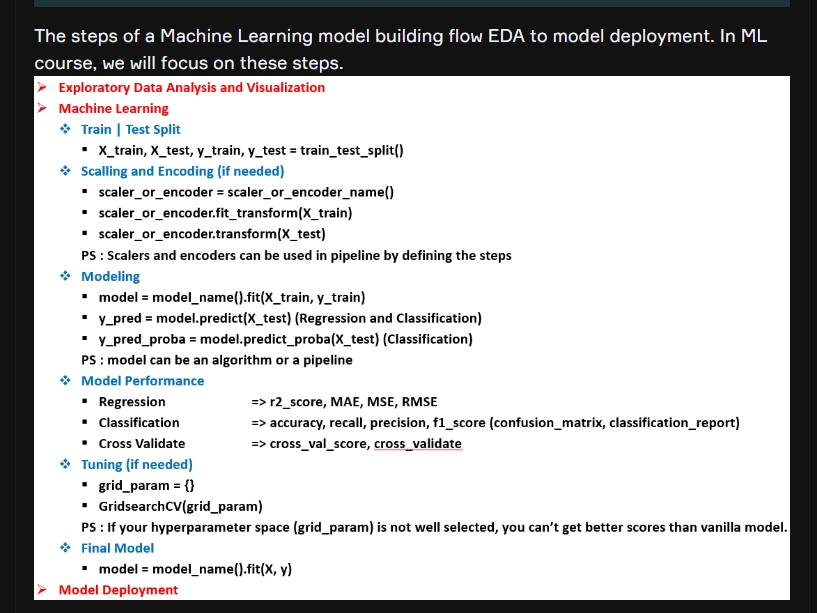
Train test data ayır.

Model oluştur, model building

Modeli eğitiyoruz. Training yapıyoruz train datası ile

Prediction alıyoruz Test datası ile

Model evaluaiton Test data ve train datasından metrik alıp karşılaştırıyoruzz. Bias variance durumlarına bakıyoruz. Overfitting, underfitting var mı buna bakıyoruz.



Overfitting te train ve test data varyansları birbirinden çok farklıdır.

**Session 4**

Regularization: Modele hata eklemek demek. Bunu **Ridge Regression, Lasso, Elastic-Net**, Cross Validation and Grid Search yöntemlerinden biri ile yaparız.

Regularization is a technique used in machine learning to prevent overfitting. It allows you to train a model that is simpler, more interpretable and generalizes better to new unseen data.

Multicollinearity: Independent feautrelerin yüksek korele olması durumudur (%90 üzeri). Bu durumda hata ekleyerek overfitiing i düşürmeye çalışacağız.

Lasso Regression (L1) (Lamda \*slope^2). Lambda modelin içindeki parametredir. Modele ince ayar yaparken kullanırız. Katsayı=parametre. Lamda ile modele eklediğimiz hata miktarını ayarlıyoruz diyebiliriz. Lambdanın değerini doğru belirlemek çok önemlidir.

Ridge Regression (L2) (Lamda \*|slope|).

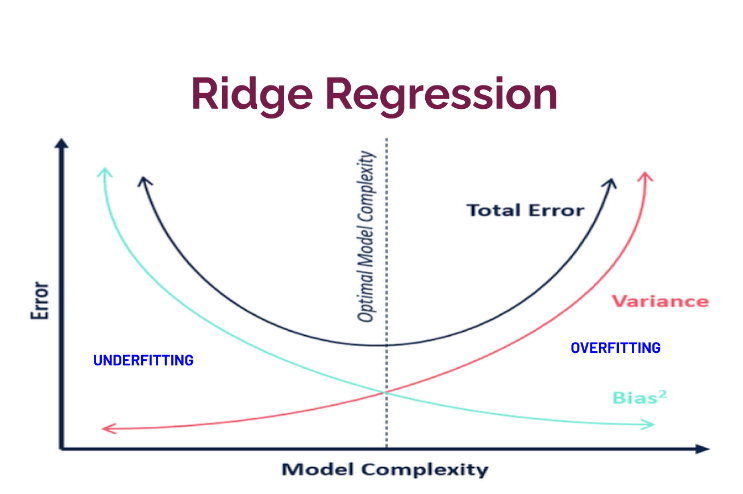
parametre = modelin kendi bulduğu (coef)

[11:25](https://ds1322eu.slack.com/archives/C044UUZD6EM/p1679732750548989)

hyperparametre = benim ayarladığım (lambda)

**Ridge Regression:** Train setinde hiç hata yapmazken regularization ile (hata eklediğimizde) Train setinde hatalı ama test data da ki hatayı azaltmış oluyoruz. Overfitting pronlemini bu şekilde çözmüş oluyoruz.

Underfitting durumunda da hata (bias) çok yüksek, variance çok düşüktür.

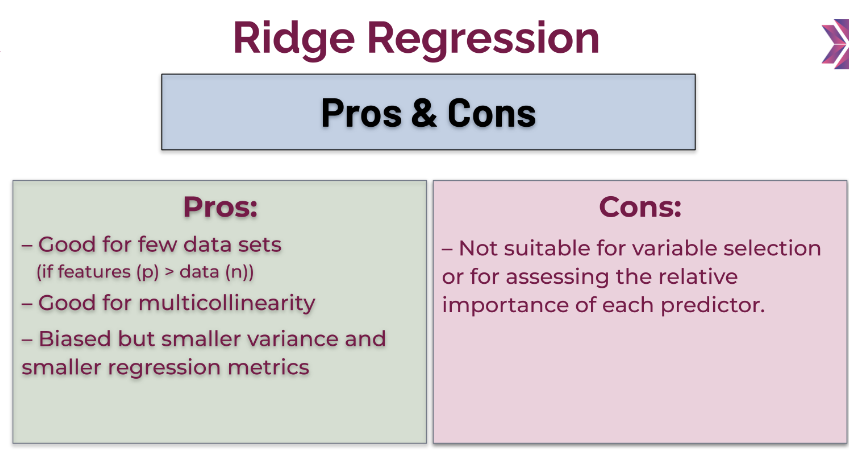


total errr = bias + variance + ie (ie değişmez total err değişmez)

* bias artarsa variance azalır tam terside geçerlidir.

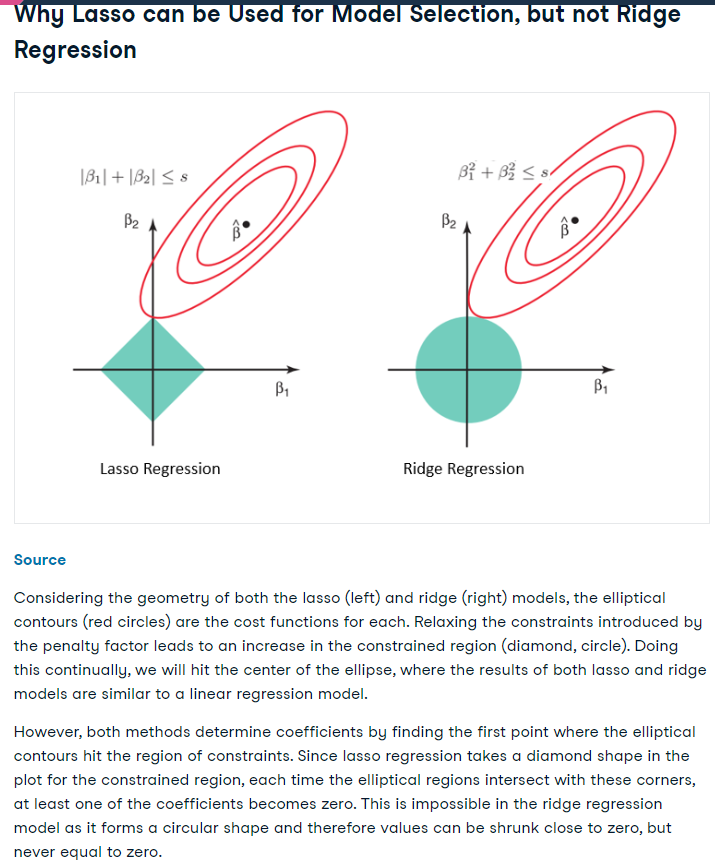
Tersten düşünürsek; yüksek hatada (**undefitting**durumu) modelin hatasını azalta azalta giderken (modelin complexcity'ini arttırırken) bir noktada optimum seviyeyi yakalıyoruz (trade off)Bu noktadan sonra hatayı azaltmaya devam edersek (mesela feature sayısını arttırırsak) bu sefer **overfitting**'e kayıyor..

Ridge katsayıların hepsini belirli oranda düşürür.

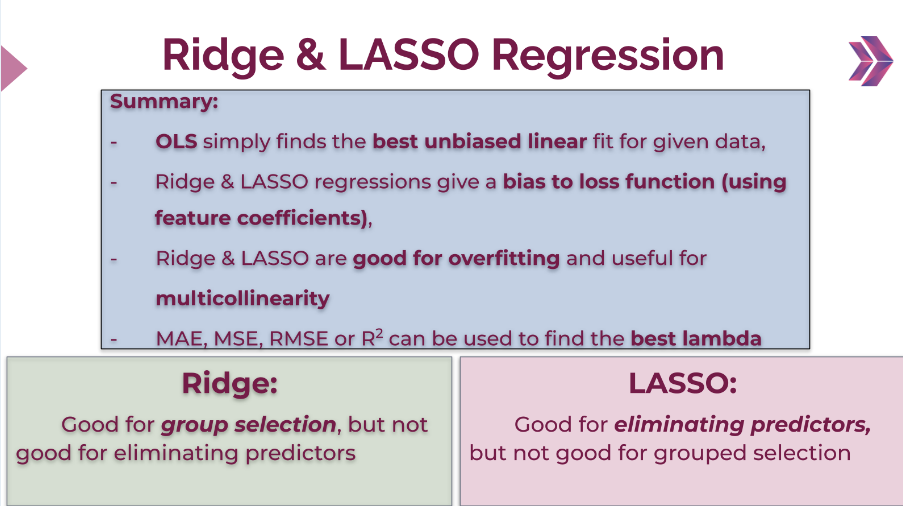


**Lasso Regression: (**Lamda \*slope^2): Lasso tüm katsayıları düşürürür. Bazılarını 0 yaparak öldürür. Önemsiz gördüğü future ların katsayılarını 0 yapar. Yüksek korele future lardan birini seçer. Hangü future daha önemli, bununLasso ile anlayabiliriz.



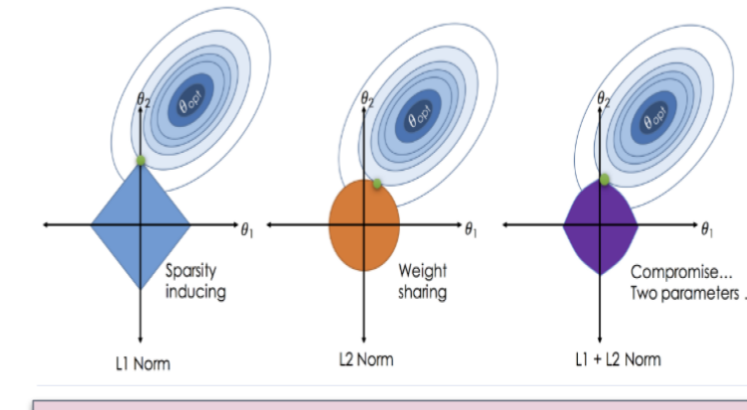


Ridge süründürür, Lasso öldürür.



DAHA ÇOK RİDGE KULLANILMALI, AMA BİR FUTURE SEÇİMİ YAPACAKSAK LASSO KULLNILMALI.

**Elastic Net: Lassoya yanlı bir modeldir.**





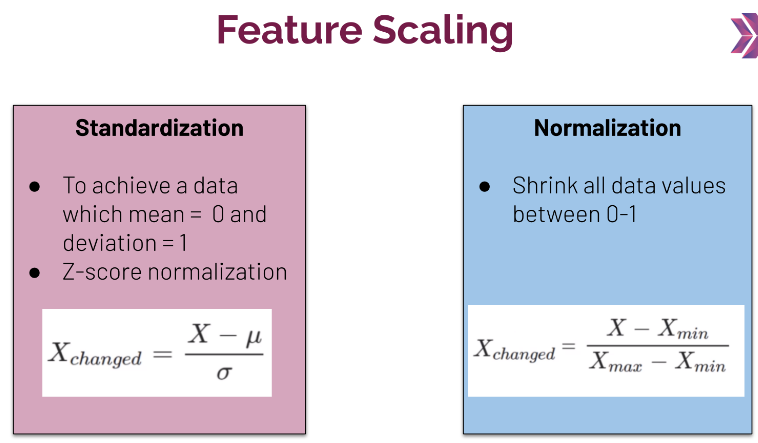
(1-alfa)/2 de alfa ya vereceğimiz değer, ne kadar lasso ne kadar Ridge gibi davranacağını belirleriz. Eğer alfa=1 verirsek Lasso olmuş olur.

**Feauture Scaling:**

* Modelin verimli ve hızlı çalışabilmesi için önemlidir. Hesaplama maliyetini düşürmek için scaling yapmak önemlidir.
* Bu sayede gradient descent hızlı çalışmış olur.
* Feature scaleing yapıldı ise, katsayısı büyük olan future nin daha önemli future olduğunu söyleyebiliriz.
* Modeliniz scale edilmiş data ile beslediyseniz, Xtest i scale yaarak predictionlarınız yapmanız gerekir.
* **Siz modelinizi scale edilmiş veri ile eğitmişseniz, predict için dışardan modele sokacağınız verinin de scale edilmiş olması gerekir. Bu, modelin featurelar arasındaki ölçek farklarını dikkate alarak doğru tahminler yapmasını sağlar. Yoksa modelin tahminleri yanıltıcı olabilir ve model performansı düşebilir. Bu yüzden modelin kullanımı için dışarıdan yeni veriler girilmeden önce bu verilerin de ölçeklendirilmesi gerekiyor.**

Scaling neden onemli:

* Gradient Descent algortihmasinin verimli ve hizli calisabilmesi icin scaling onemli. zaman kazandirir
* Model coefficients: scale yapildiysa rahatlikla en yuksek coef olan feature en onemli etkieye sahip diyebiliriz. scaling yapilmadiysa soylenemz bu net.
* distance base lagoritmalar (mesafeye dayali calisan algoritmalar)in iyi calisabilmesi icin mutlaka scaling yapmali



Normalization a min-max scale de denebiliyor.

**Robust scaler:** Standardization ve normalizationdan sonra 3. Bir alternatif scale etme yoludur. Outlier değerler olduğunda işlevsel olabilmektedir.

* + Data da get dummy ile yapılmış 0 ve 1 lerden oluşan data çoksa normalization tercih edilebilir.
  + Ama en iyisi deneme yanılma ile en uygun scale yöntemini seçmektir.

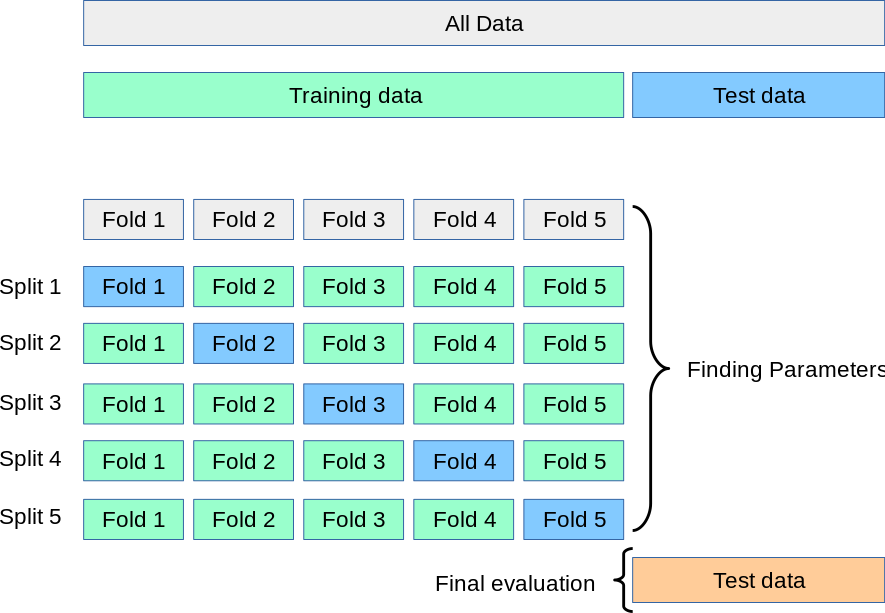
**Not:** Scaleing işlemini training dataya uygulamış oluyoruz. Tüm dataya yapmıyoruz. Tüm fit işlemlerini training dataya uyguluyoruz.

**Cross Validation (çapraz doğrulama)**

Birden çok kez train-test datayı ayırıp modeli farklı alt datalar ile ile test eder. Adeti k ya vereceğimiz değer ile biz belirleriz. Tüm skorların ortalaması alınır en son.

Test data🡪 hold out –tutulan moedele gösterilmeyen data- olarak da isimlendirilir.

Lambda ya biz alpha diyeceğiz. Alph aarttıkça regularization da artar. Regularization hata ekleme demektir. Sayı belirtmez isek alphanın default değeri 1 dir.



**Grid Search**

İdeal lambdayı bulmak için kullanırız.

ÖZET OLARAK;

1. Sptlit the data.
2. Feature Scaling.
3. Modelleme
4. Cross Validation, Grid Search
5. Evaluation

aşamalarını ugulamış oluyoruz.

Session 5

**Multicollinarity'nin** **ML modellerinin performansını olumsuz etkileri:**

1. sıkıntı; multi lineer regresyon gibi modellerde parametre tahmini ile ilgili. Multicolinerite durumunda model, bağımsız değişkenler arasındaki korelasyon nedeniyle parametre tahminlerinde yüksek bir varyans gösterir. Bu nedenle, modelin tahmin performansı düşük olur.
2. sıkıntı; modelin genelleştirme performansıyla ilgilidir. Eğer model, eğitim verilerindeki yüksek corelasyonlu featurelar arasındaki ilişkileri öğrenirse ve bu ilişkiler test verilerinde de geçerli değilse, modelin genelleştirme performansı düşük olabilir.

**train skor ile test skorunun birbirine yakın ve yüksek olmasından anlıyoruz ki**; train data ile yapılan eğitim sonucunda, dışardan modelin görmediği data girdiğinde de eğitimdeki kadar başarılı predict değerlerine ulaşmışız.  Demek ki eğitim genelleme konusunda başarılı olmuş.

**Multikolinerite ile başa çıkmak için birkaç yaklaşım vardır**:İlk olarak, *lineer regresyon* gibi modellerde, **değişkenleri *çıkarma***veya ***birleştirme***gibi yöntemler kullanılabilir.Diğer bir yöntem, **regülarizasyon**teknikleri kullanarak ***model karmaşıklığını azaltmaktır.***Ayrıca, ***feature selection*** yöntemleri de kullanılabilir.

**Grid search**bizim için ne yapıyordu arkadaşlar?  
Tüm kombinasyonların test edilmesiyle en iyi hiperparametre değerlerini bulmayı amaçlıyordu. Peki **bu değerler mutlak en iyi değerler diyebilir miyiz? Hayır!**En iyi hiperparametre değerleri, datasete ve probleme bağlı olarak değişebilir. Grid search, **sadece belirli bir hiperparametre aralığını kapsayan**bir arama yaptığından en iyi hiperparametre değerlerini bulmak için **yeterli olmayabilir**.Ayrıca **hiperparametreler arasındaki etkileşimleri dikkate almaz**. *Bu nedenle, belirli bir hiperparametrenin en iyi değerini bulduktan sonra, diğer hiperparametrelerin en iyi değerleri farklı olabilir*!Bununla birlikte, belirli bir hiperparametre aralığını kapsayan tüm kombinasyonların test ettiği için **en azından bize iyi bir hiperparametre seti sağlar**. Bu da iyi bir başlangıç noktası olarak işimizi kolaylaştırır.

**Why scale our data?**

* Many models use some form of distance to inform them
* Features on larger scales can disproportionately influence model
* We want features to be on a similar scale
* Normalizing or standardizing (scaling and centering)

[9:31 PM] **How to scale our data**

* Subtract the mean and divide by variance. All features are centered around zero and have a variance of one. This is called standardization
* Can also subtract the minimum and divide by the range. Minimum zero and maximum one.
* Can also normalize so the data ranges from -1 to +1

[10:30](https://ds1322eu.slack.com/archives/C044UUZD6EM/p1680204611820939)

**Cross Validation**

* We do cross-validation to check whether the one-time scores we receive are consistent or not, in other words to measure the estimator performance. Cross validation is only applied to the train set.
* Cross validation uses different data samples from train set and calculates scores for each data sample. We can calculate the final performance of estimator by averaging these scores.

Data lekeage nin önüne geçebilmenin en güzel ve etkili yolu cross validation yapmaktır.

Regülezation (ridge/lasso/elastic net) etmek multicollinearity sorununu çözmede de etkilidir; multicollinear olanlardan birinin katsayısını 0 yaparak onu öldürür. Linear regresyon kullanacaksak her zaman regüle hallerinden birini (ridge,lasso) tercih etmek en iyisidir. Böylece coefler daha stable olur.

Alpha regülarizasyonu kontrol eden bir parametredir. Alpha büyüdükçe modele eklenen hata artar, bununla birlikte coef ler düşer.

Grid search te yaptığı ara işlemlerin detaylarını göstermemesi için n\_jobs = -1 diyoruz.

**Session 6**

LOGISTIC REGRESSİON

Classification yapan bir modeldir.

Data yı modele soktuğumuzda kategorik ama hem de discrete değerler verir. Ankette 5 yıldız şeklinde değerlendirme ölçeği buna örnektir. Kanser mi, değil mi” şeklindeki ölçeklendirme de bir classification dur.

Lineer regresyonda continious bir değer bulaya çalışırken, Classification da discrete veya kategorik bir değer bulmaya çalışıyoruz.

Classification aslında bunu arka planda lineer regresyon ile yapar. Logistic regressionda classification yaparken lineer bir line ile bunu yapar. Yine kullanacağımız farklı metrikler olacak(lineerde rmse, mse gibi).

Disease diagnosos, Fraud detection, predicting the customer churn result gibi durumlarda kullanacağız.

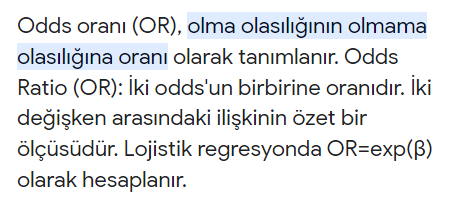
Logistic regressionda S-line ile prediction yapacağız. Line🡪 S-line a geçiş Sigmoid fonksiyonu ile olmaktadır. Sigmoid fonksiyonu bize datadaki her değer için bir olasılık hesabı belirler. Biz de classification için bir eşik değer belirleriz. Default olarak bu fonksiyonda eşik değeri %50 değeri olarak kabul edilir. Bu olasılıkları hesaplarken gradient descent algortması ile çalışır. S-line hesaplanan olasılık değerleri kullanılarak çizilmiş oluyor.

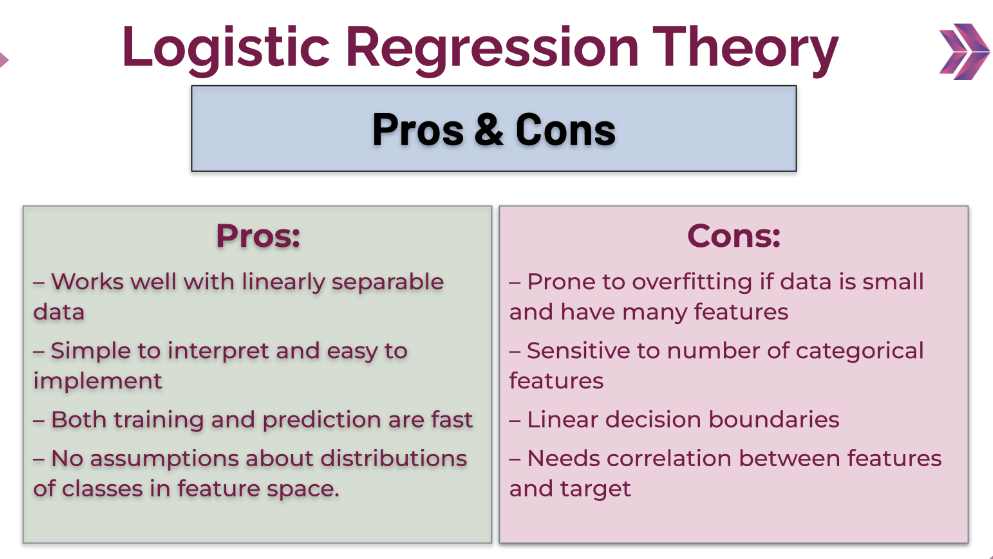
Olasılık hesaplarken her zaman 1 classı üzerinden hesapalma yapar.

Normalde lineer regresyonla datamızı modellediğimizde line'ımız -sonsuz ile +sonsuz arasında değer alabilir. Logistic regression ile classification yaparken lineer olarak oluşturduğu line'ın denklemini sigmoid fonksiyondan geçirerek değerleri 0-1 arasında sıkıştırmış olur. Böylece datamızı sınıflandırmış oluruz.

Lineer regresyonda LSE metodu kullanılırken logistic regresyonda MLE(Maximum Likelihood Estimation) metodu kullanılmaktadır.

Sigmoid fonksiyonu nonlinear bir fonksiyon olduğundan total error eğrisi convex bir yapıda değildir.  çözümün convex bir yapıya dönmesi için ara işlem olarak **odds işlemi** kullanılır. log loss min olursa MLE maximum olur



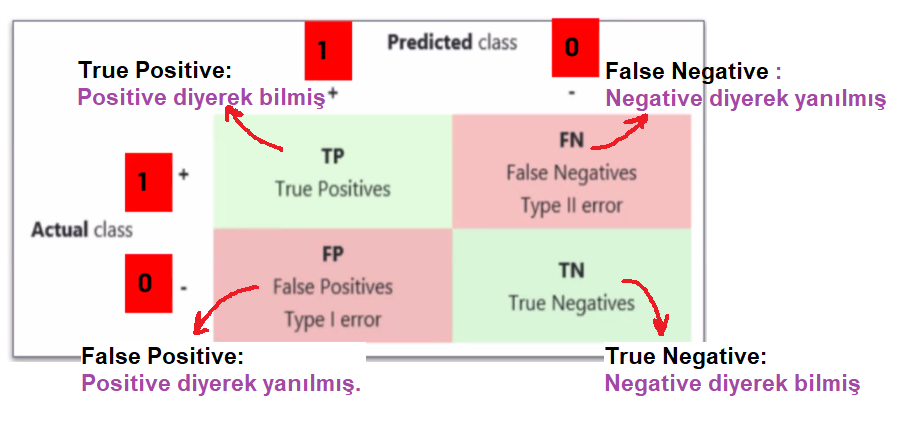


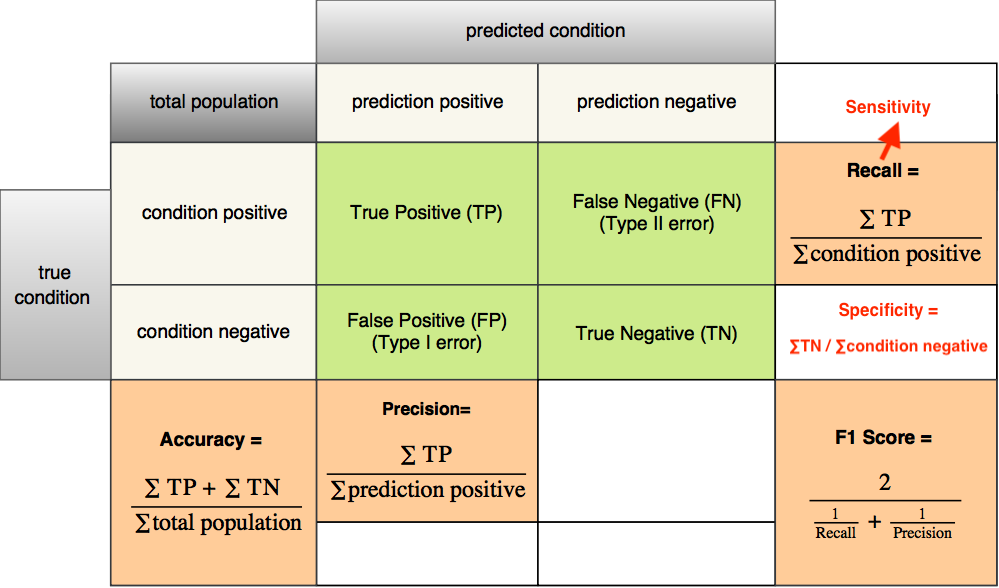
Linear regression ve logistic regression stable parametrik modellerdir. Parametrik modeller sayı sever, kategori sevmez.

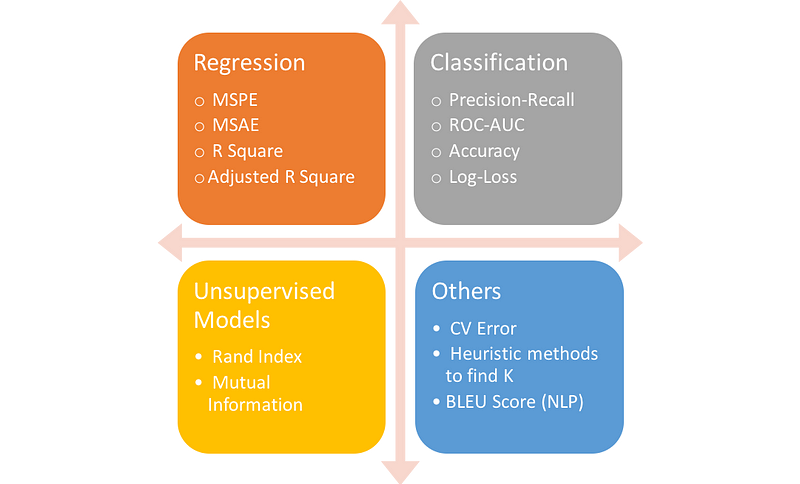
Scikit-learn kütüphanesi içindeki LogisticRegression sınıfının bazı önemli parametreleri şunlardır:

* penalty: "l1" veya "l2" değerlerinden biri olabilir. L1 ceza türü, modeldeki katsayıları sıfıra yakınlaştırarak özellik seçimi yapar. L2 ceza türü, katsayıları büyük olmayan ama birbirlerine yakın olan değerlere yakınlaştırarak overfitting'i önler.
* C: ceza parametresi, sıfıra yaklaştıkça cezalandırma artar, büyüdükçe cezalandırma azalır. Varsayılan değeri 1'dir.
* solver: "newton-cg", "lbfgs", "liblinear", "sag" veya "saga" değerlerinden biri olabilir. Bu parametre, eniyileme algoritmasını seçer. "liblinear" özellikle küçük veri kümeleri için iyidir. "sag" ve "saga" büyük veri kümeleri için daha hızlıdır.
* max\_iter: eniyileme algoritmasının maksimum iterasyon sayısıdır. Varsayılan değeri 100'dür.
* class\_weight: dengesiz veri kümeleri için sınıf ağırlıklarını belirlemek için kullanılır. Varsayılan değeri yoktur.

**Classification Error Metrics-Confusion Matrix**







**Metrik 1:** **Accuracy:** **Tüm doğru isabet edilen değer adedi / Tüm tahmin adedi ile hesaplar.**

Accuracy karar verme için her zaman iyi bir metrik olmaz, yanlı sonuç verebilir. Özellikle sınıflar arasında dengesiz bir dağılım olduğu zaman sağlıksız bir sonuç vermemize sebep olabilir. 0-1 arasında değer döner ve 1’e yaklaştıkça başarımız artar. Sınıf dağılımlarının dengesiz olması durumunda ***accuracy*** yerine ***precision*** ve ***recall*** metrikleri incelenir.

**Metrik 2 : Precision(Kesinlik): TP/(TP+FP) .** Tahmin edilen pozitif sınıfların gerçekte ne kadar pozitif olduğunu gösterir.

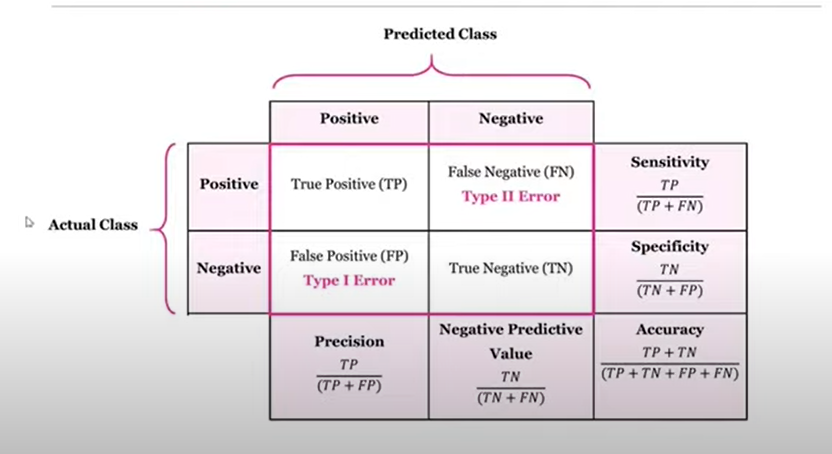
**Metrik 3: Recall(Sensitivity):** Kanseri yakalama oranıdır. 1 classı için isabet etme oranıdır.1 sınıfını yakalama oranıdır. TP/ (1 classı), yani **TP/(TP +FN).** Gerçektepozitif olan sınıfların ne kadarını pozitif olarak tahmin ettiğimizi gösterir.

🡪 Recall arttıkça precision düşer.

**Metrik 4: F1 Score:** Precision ve Recall un harmonik ortalamasıdır. Dengeli bir dağılım belirlemeye çalışır.

**Metrik 5: (Specificity):** 0 classı için isabet etme oranıdır.0 sınıfını yakalama oranıdır. **TN/(TN+FP)**

**Metrik 6: ROC/AUC**: Modelin 0 ve 1 classlarını ne kadar ayrıştırdığının oranını veriyor. **TP/(TP+FN)**

****

🡪 Balanced datalarda ROC/AUC (accuracy) veya precision kullanılabilir ama daha çok ROC/AUC tercih edilir.

🡪 Inbalanced datalarda precision, recall, f1 scorlarına bakacağız. Precision tercih edilir.

🡪 **Threshold değeri(eşik değeri**): Sınıflandırma modellerinin başarı değerlendirme süreçlerinde, başarı ölçüm metriklerine geçmeden önce, bağımlı değişkenin tahmin edilen değerlerinin elde edilmesi gerekmektedir. Bu kapsamda Sigmoid fonksiyonu kullanılarak ulaşılan olasılık değerlerinin belirlenen bir eşik değerin üstünde veya altında olması durumuna göre tahmin edilen sınıflar belirlenir. Bu eşik değere **classification threshold** denir.

🡪 Amaç cost effective değerlendirme yapmaktır. Precision recall arası dengeyi iyi belirlemek önemlidir bu nedenle.

Multiclass datalarda cross validationda precision, recall ve f1 metrikelrini tek başına kullanamayız. Bunun yerine weighted ve macro formlarını kullanmalıyız.

Multiclass datalarda average parametresi için none seçmeliyiz.

Multiclass datalarda threshold 0.5 olarak atanmaz. Hangi classın probability değeri yüksek ise o sınıfa atanır.

**KNN Theory (K Nearest Neighbours)**

Lazy learner bir modeldir, datayı ezberleme yaklaşımındadır, öğrenme (fit) süreci yoktur.

Küçük datalarda daha iyi çalışır, Scaling yapmaya ihtiyaç duyar (özellikle büyük datalarda).

Mesafe (distance) önemlidir. Belirlenen bir komşuluk değerine (k değerine) uygun olarak seçim yapar. k seçimi önemlidir, optimal olarak seçilmelidir. k değeri optimal değerinden küçük seçilirse overfit, büyük seçilirse de underfit olur. k’nın default değeri 5 tir.

KNN lineer bir model değildir.

Outlier değerlerden etkilenir. Inbalance değerlerden etkilenir.

Optimal k yı belirlerken 3 yöntem vardır:

* + 1. Elbow : Daha çok kullanacağımız bir metottur. Hesaplama maliyeti düşüktür.
    2. Grid search: Hatanın her zaman minimum olduğu noktayı bulur. Bu nedenle ciddi bir hesaplama maliyeti olabilmektedir. Eğer insan hayatını ilgilendiren ciddi konulu bir data ile çalışıyorsak bu metot tercih edilebilir. Yöntem tercihi context ten context e değişebilmektedir.

Yani, GridsearchCV herzaman en düşük error scorunu verecek olan maximum metric scorunu döndürür. GridsearchCV sonrası bulacağımız n\_neighbors sayısı çok maaliyetli olabileceğinden elbow metoduyla seçmek best practicedir.

**Not:** Optimum K değerini seçmek için öncelikle datamız dengeli ise **accuracy** metricini, dengesiz ise hedef classımızın **recall** metriğini kullanabiliriz.

**Not:** weight parameter of KNN tercihi: default = Minkowski dir. Burada p seçimi yapılması gerekir.

P=1 ise Manhattan, distance formülü ile, p = 2 ise Euclidean distance formülü ile hesaplama yapar.

P=1 seçiminde outlierlardan etkilenmez. Çünkü mesafeyi dolanarak (dik kenarlar toplamı şeklinde) ölçmüş olur.

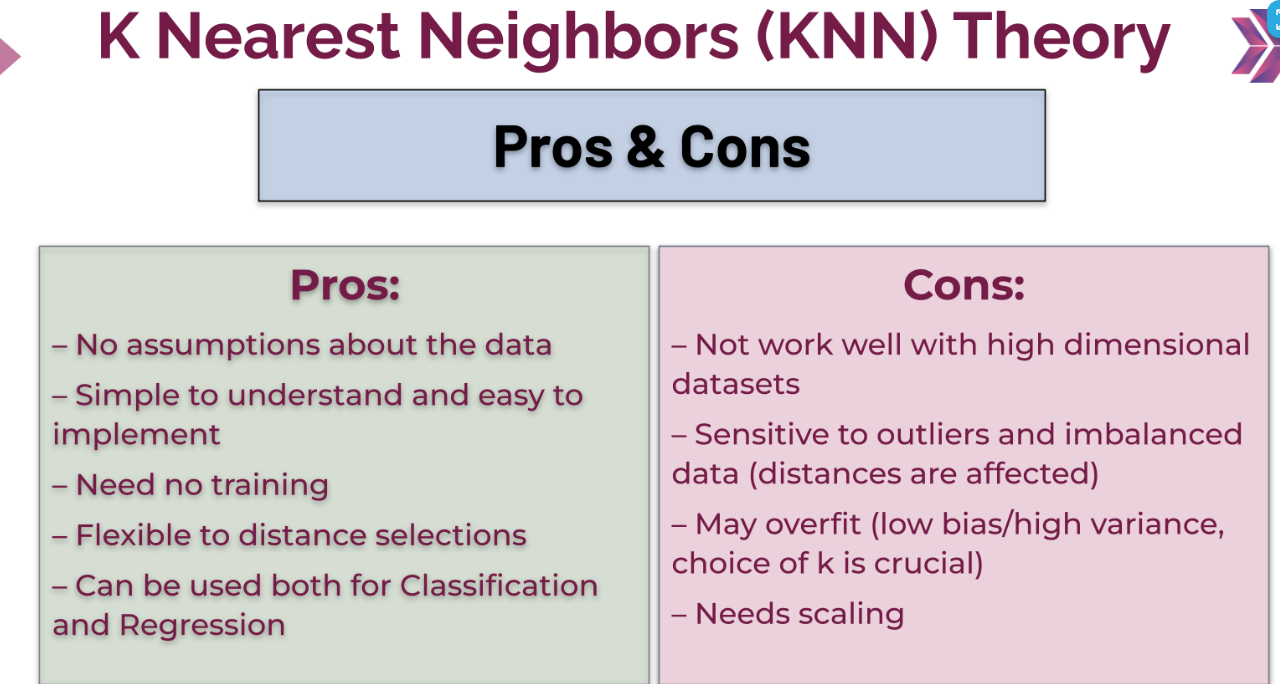
P=2 seçiminde outlierlardan dah afazla etkilenir. Hipotenüs üzerinden seçer.

Uniform🡪 majority voting (eşit ağırlıklandırır hepsini) distance🡪 uzaklığa göre ağırlıklandırır.

K yı tek sayı seçmek best practise açısından kullanışlıdır.

Balance olması: 0 ve 1 adedinin eşit olmasında direkt balance diyebiliriz. Eğer eşit değilse hemen imbalance diyemeyiz, skorlara bakmamız gerekir.

**ÇOK ÖNEMLİ NOT**: Test datasında (hold out set) üzerinden bulacağımız en optimum K değeri data leakage sorununa neden olabileceğinden train datamızı CV'da verip burdaki validation setinin ortalama error scorlarına göre karar vereceğiz.



Özetle;

KNN için en önemli olan hyper\_parametrelerimiz:

1. n\_neighbors (binary modellerde classlar arasında eşitlik olmaması için tek sayı olmasına dikkat edilmelidir. Multiclass datalarda tek veya çift olması farketmez.)

2. weights

3. p'dir.

**Support Vector Machines**

1960larda geliştirilmiştir. Lineer veya non lineer datalarda kullanılabiliyor.

Yüksek accuracy veren classification problemlerinde kullanılır.

Usage areas: Text Classifciation, Image Recognition, Image-based Gender Detection.

SVN algoritmasının temel amacı bir hyperplane bulmaktır.

* SVM, makine öğrenmesinde sınıflandırma ve regresyon analizleri için kullanılan bir algoritmadır.
* SVM, öğrenme için veri setindeki örnekleri kullanır ve bu örnekleri bir uzayda birbirine yakınlık açısından konumlandırır.
* SVM, sınıflandırma yapmak için, iki sınıf arasındaki ayrımı en iyi şekilde sağlayan bir hiperdüzlemi (hyperplane) bulmaya çalışır.
* SVM, doğrusal veya doğrusal olmayan veri kümeleri için kullanılabilir. Doğrusal olmayan veri kümelerinde, verilerin yüksek boyutlu uzaya projekte edilmesi (kernel trick) kullanılabilir.
* SVM, verilerin sınıflarının belirlenmesinde, örneklerin uzaklıklarını dikkate alır.
* SVM, margin (kenarlık) kavramını kullanarak, sınıflar arasındaki ayrımın ne kadar iyi olduğunu ölçer. Margin, bir hiperdüzlemle en yakın örnekler arasındaki mesafeye eşittir.
* SVM, veri setindeki aykırı örneklerin etkisini azaltır, böylece daha genelleştirilebilir modeller oluşturabilir.
* SVM, öğrenme sürecinde genelleştirme hatasını minimize etmeye çalışır. Bu, aşırı öğrenme (overfitting) problemini önlemeye yardımcı olur.
* SVM, bir sınıflandırma problemi ile karşı karşıya olduğumuzda kullanılır. Örneğin, bir nesnenin bir sınıfa ait olup olmadığını belirlemek istediğimizde, SVM algoritması kullanılabilir.

C parametesi büyüdükçe margin küçülür ve overfittinge yaklaşır, çok küçüldükçe margin büyüyerek underfittinge gider. Optimum C yi bulmamız gerekir.

Geniş margin🡪 soft margin, dar margin 🡪hard margin

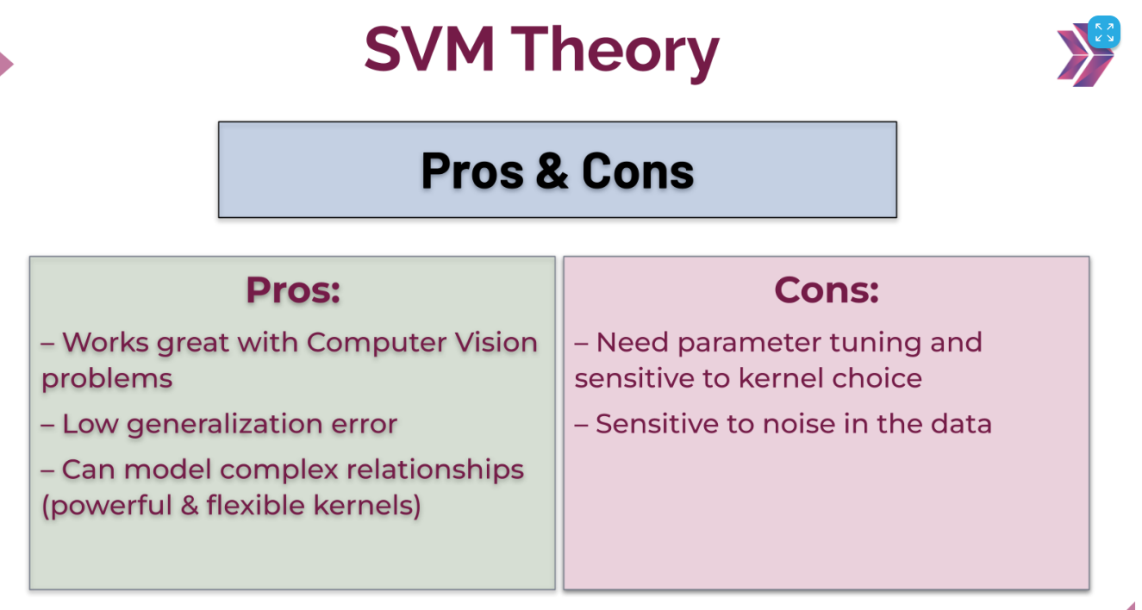
Kernel Trick ile nonlineer datalarda dataya boyut kazandırarak class larımızı daha iyi ayrılabilir hale getiriyor ve hyperplane ile ayrılabilir hale geliyor.

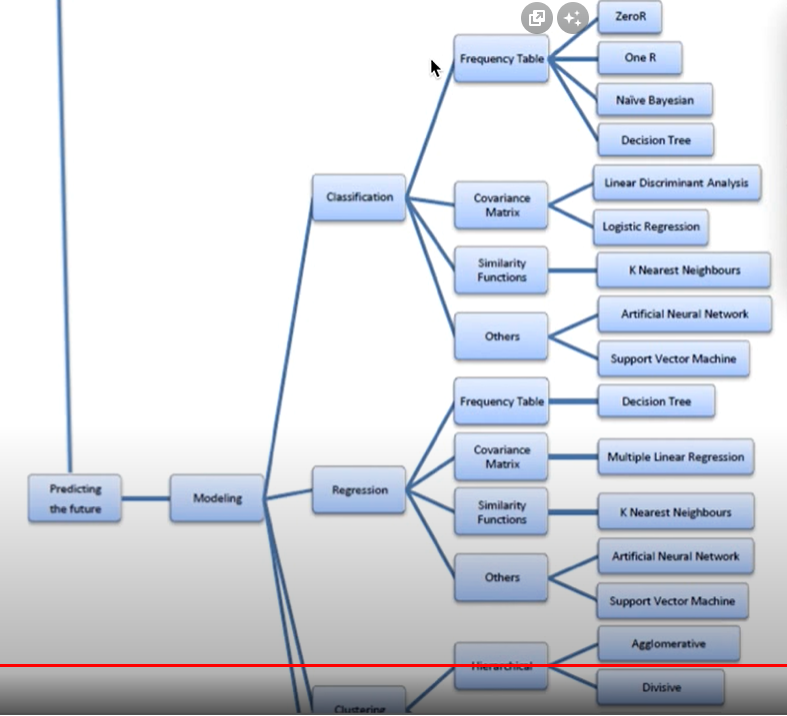
Kernel parametresinin default u rbf tir. Alternatifleri “linear, poly, sigmoid, precomputed” dır. Poly de degree de seçmek gerekir. İhtiyaca uygun seçilir.

Feature sayımız çok olduğu ve gözlem sayımız as ise 🡪 linear

Feature sayımız çok olduğu ve gözlem sayımız çok ise 🡪 rbf (gauss)

Kernel rbf, poly, sigmoid kullanıldığında “gamma” parametresi kullanılır, defaultu 1 dir. Gamma küçüldükçe soft hale gelir, büyüdükçe hard hale gelir vr overfit e sebep olur. Gammayı datayı çevreleyen kıvrım detayını belirleyen parametre gibi düşünebilirsin.





Desicion Tree Theory

Explainable bir modeldir.

Medical diagnosis, Text classification, credit risk analysis vb çok durumda kullanılır.

* + 1. Desicion node (Root Node)
    2. Sub-Tree
    3. Leaf node

Her bir yaraktaki homojenlik sağlanana kadar dallanma devam eder.

Gini impurity index mini olmalı.

Information Gain entropy (düzensizlik/belirsizlik) tabanlıdır, büyük olması istenir. Tüm data aynı kategoride ise entropy 0 olur, tamamen purely dir; 0 ve 1 class dataları eşit ise entropy değeri 1 olur.

Desicion theory entropy üzerinden hesaplama yapar.

Entropy = kaos gibi düşünülebilir. Düzen arttıkça kaos azalır.

Desicion Regresyonda çeşitliliğin(varyansın) az olmasına yönelik çalışıyor. O leafteki değerlerin ortalamasını döndürüyor.

NOT: criterion🡪 gini / entropi den biri seçilir. Max\_depth: derinlik belirlemesidir.

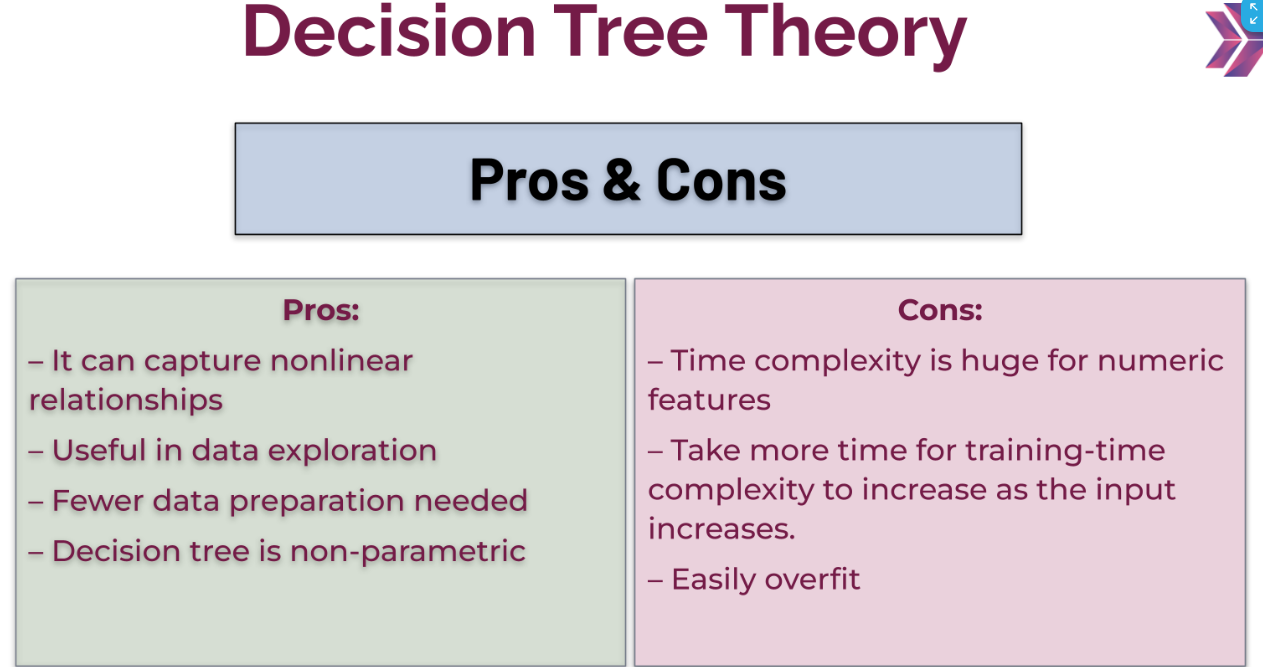
Splitter: best/ random. Best seçilirse en önemli feautureye göre seçim yapar

Max\_features: dikkate almasını istediğimiz feature sayısıdır.

Min\_samples\_leaf: bölünebilmesi için mininmum gözlem sayısıdır. Defaultu 1 dir. Bunu değiştireceğiz aralıklar vererek.

Özet: Non linear ilişkileri çok güzel yakalar, data görselleştirme için kullanışlıdır.outlier değerlerden çok etkilenmez (regresyon hariç). Non-parametrik bir değerdir. 😊

Hesaplama maliyeti fazladır, kolaylıkla overfit olabilir. ☹

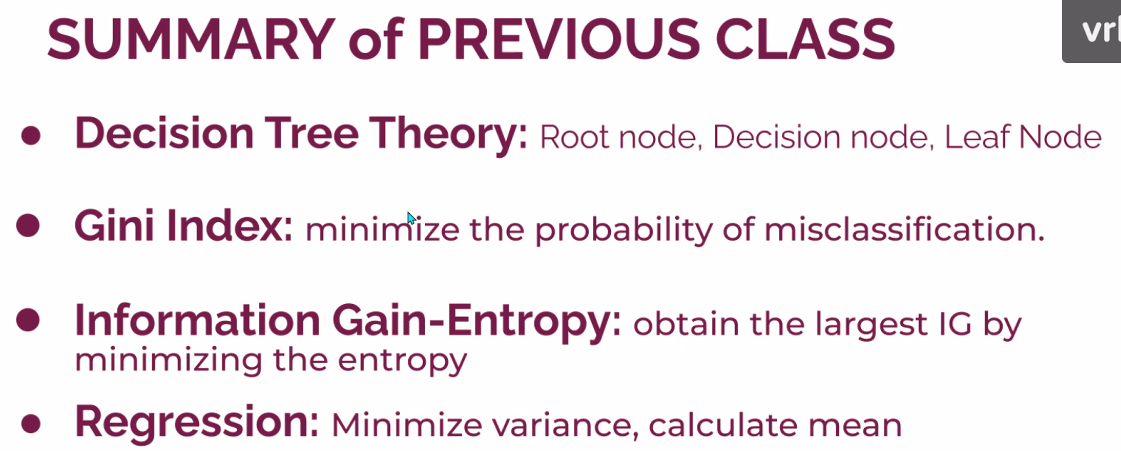


Tree based modellerde onehatencoder kullanmıyoruz, ordinal encoder kullanıyoruz. Çünkü

* + Skorların daha yüksek çıkmasına katkı sağlıyor
  + Modelin 2.5 kat daha hızlı çalışmasını sağlıyor
  + Doğru feature importance elde edebilmek için ordinal encoder yardımcı oluyor.
  + Hiyerarşi çok önemli olmuyor, if else ile çalıştığı için.

Tree based modeller her zaman en önemli future a odaklanma, diğerlerinin katsayısını sıfırlama eğilimindedir.

Tree based modeller outlierlara sensivity dir.



ROC-AUC, bir sınıflandırma modelinin performansını ölçmek için kullanılan bir metrikdir. Bu metrik, sınıflandırma modelinin doğruluğu, hassasiyeti ve spesifitesi gibi performans ölçütlerinin birleştirilmesiyle hesaplanır.

ROC-AUC, sınıflandırma modelinin birçok farklı kesme noktası kullanılarak elde edilen hassasiyet ve spesifite değerlerini temsil eden ROC eğrisinin altındaki alanı ifade eder. Bu değer, sınıflandırma modelinin performansını bir sayısal değerle ifade etmeyi sağlar ve farklı sınıflandırma modellerinin karşılaştırılması için kullanılabilir.

ROC-AUC'nin çizdirilmesi, sınıflandırma modelinin performansını görselleştirerek anlamak için faydalıdır. ROC eğrisi, sınıflandırma modelinin farklı hassasiyet ve spesifite seviyelerinde nasıl performans gösterdiğini gösterir. Eğrinin altındaki alanın büyüklüğü, sınıflandırma modelinin performansının ne kadar iyi olduğunu belirler.

Özetle, ROC-AUC'nin çizdirilmesi, sınıflandırma modelinin performansını ölçmek, karşılaştırmak ve görselleştirmek için kullanılır.

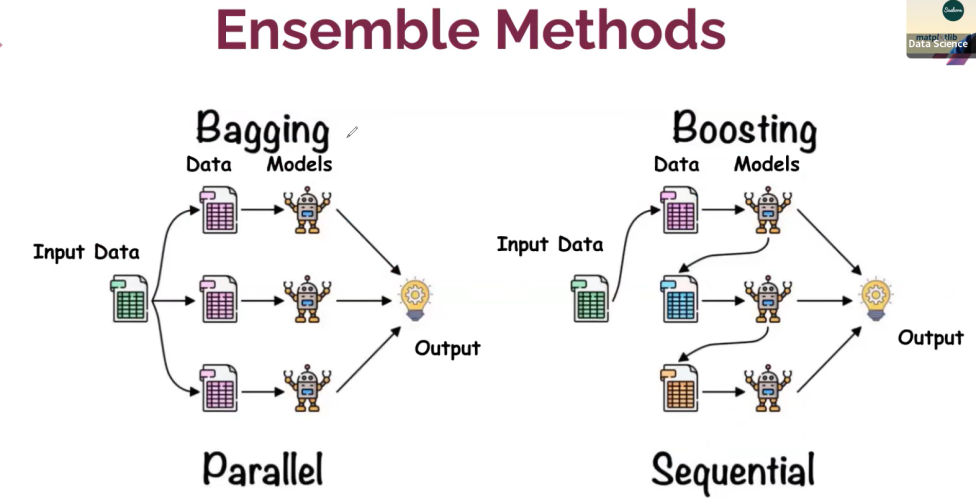
***Ensemble Metotları* Giriş: Bagging & Boosting (BootStrap Aggregating)**

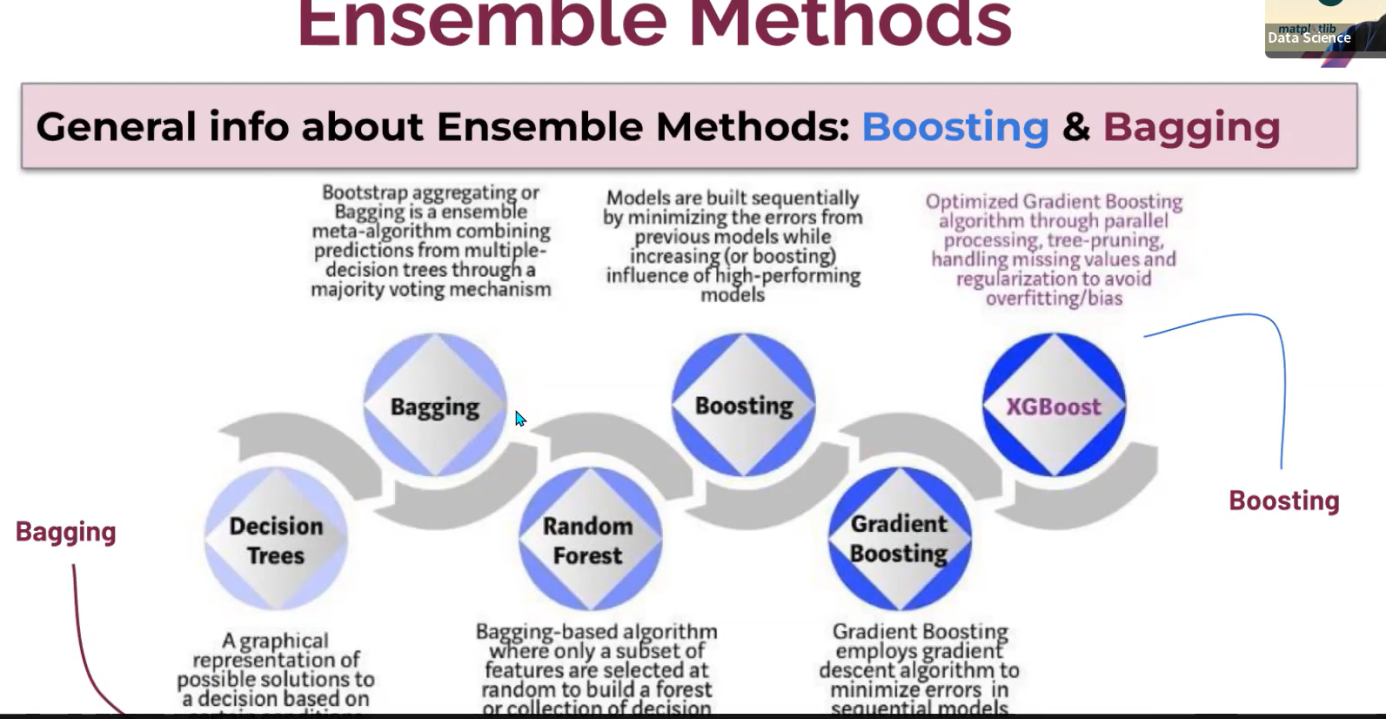
*Ensemble 2 yöntemi var: Bagging ve Boosting*

*Bagging: Birden fazla modelin aynı anda paralel biçimde çalışmasıdır. Her bir model ayrı ayrı oluşturulur ve her model kendi predictionu nu yapar. Bütün featurelerden sonuç döndürdüğü için daha güvenilir sonuç döndürmektedir.*

*Bagging methodları güçlü alt modelleri tercih ederken, bagging ve boosting metotları weak learnerslar ile çalışır. Boosting ileride detaylandırılacaktır.*

*Bagging🡪 her bir alt modelin modele katkısı eşittir, Boosting🡪 her bir alt modelin katkısı performansına göre değişir. Her ikisinde arka planda Decision Tree çalışır, ikisi de tree based modellerdir.*

**

**

**Random Forest Ensemble Method (Black Box model)**

Random forest, desicion tree algoritmasını kullanır. Bagging yöntemi kullanılır. Algoritmalar paralel çalışır. Ağaçların dallanması sırasında hem dataya hem featurelara random luk katar. Birden fazla ağaç oluşmaktadır.

Bunun için önce boostrap data deti oluşturur. Tüm data setlerinin gözlem sayısı aynı olmak zorundadır. Her bir modelden çıkan sonuca Desicion Tree uyguluyor. Datanın 2/3 ünü modele verir, 1/3 ünü out of bag e (teste) ayırır future importance için. Desicion tree de en önemli futures üzerinden dallanma yaparken Random Forest te ise farklı features üzerinden parçalama yapar ve farklı features üzerinden öğrenir, bütüncül bir değerlendirme imkanı sağlamış oluyor böylelikle.

Scale etme işlemine gerek duymuyor (+). Açıklanabiliriliği zordur (-), bu nedenle Black Box model de denir. Cardinality si yüksek future ler bu model uygulanmadan önce drop edilmelidir.

n-estimators (default = 100): The number of trees in the forest. The more trees the better accuracy.

Max-depths (default: None): The maximum depth of the tree.

Criterion( default: squared error): Diğer seçenekler absolute error, friedman\_mse, poisson dur. Dallanma kriterini neye göre belirleyeceğidir.

Max\_features (default: 1): Split yaparken seçeceği feature sayısıdır. Diğerleri log 2 gibi seçenekler var.

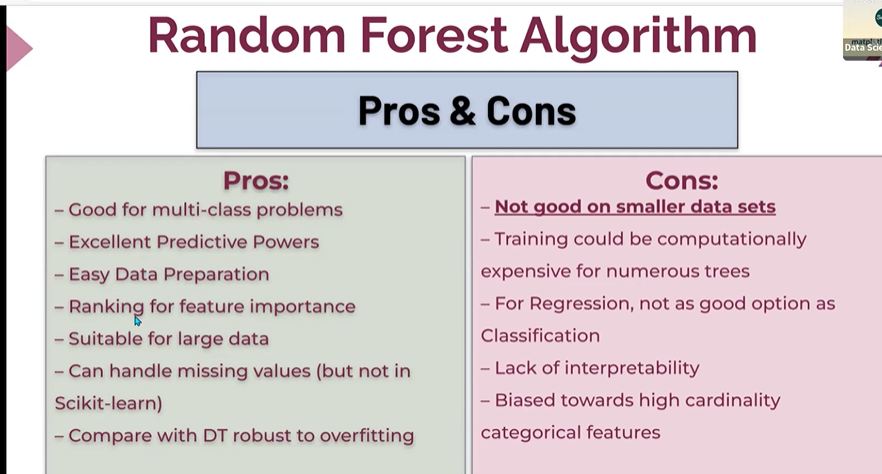
Max-feautres (default: sqrt): Number of features to consider when looking for the best split. Datada 16 features varsa karekökünü alır, 4 olarak devam eder.

Min\_samples\_leaf: En sonki bir yaprakta en az kaç gözlem olması gerektiğini belirten bir parametredir.

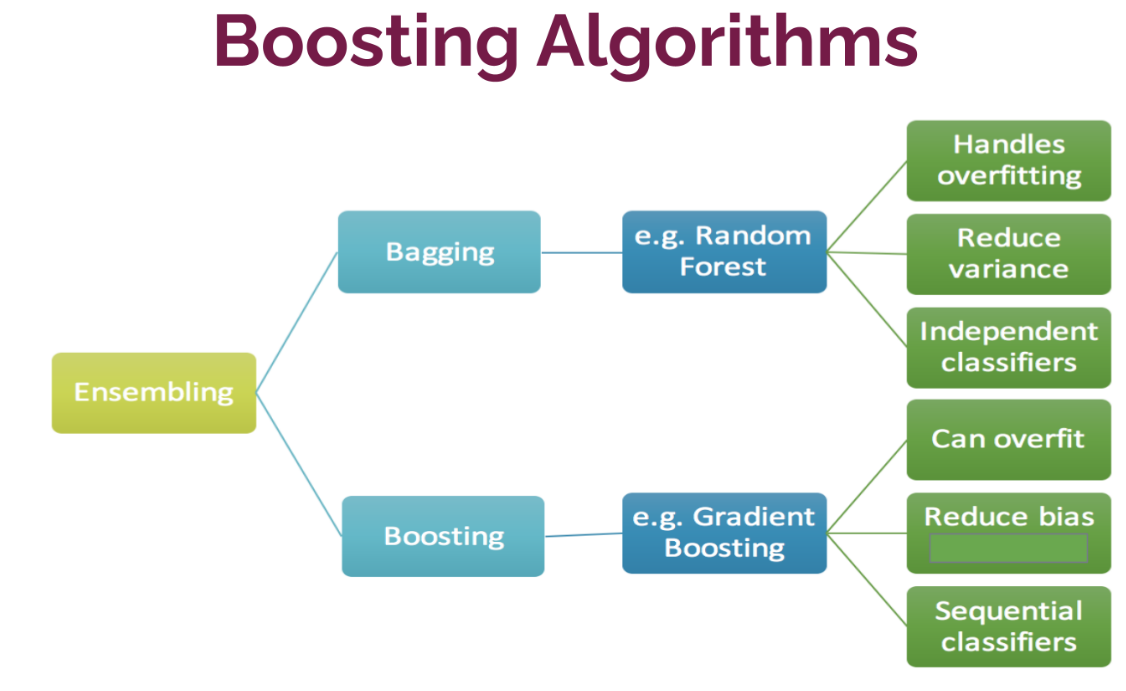
Min\_samples\_split: split yapması için olması gereken gözlem sayısıdır.

Bootstrapbool (default: True)

Oob\_scorebool (default: False)



**Boosting**



Bagging metodu varianceyi düşürür, bias ı artırır. Boosting metodu, bias ı düşürüp variance yi artırır. Boosting e week learner (zayıf ağaçlar gibi düşün) vardır.

Ada(ptive)Boost🡪 Gradient Boosting (GBM)🡪 XGBoost🡪 LightGBM🡪 CatBoost, …

**AdaptiveBoosting:** Hataya göre model ağırlıklandırılır. Hata yaptığı gözlemleri çoklar ve öğretmeye çalışır.

base\_estimator parametresinini default u DecisionTreeClassifier’dır.

**Gradient Boosting**: Bir önceki modelin yaptığı hataları düzeltme üzerine bir mantığı var. Adaboost a kısmen benziyor. Hatalı tahmin residual lerini minimize etmeye çalışıyor. Gradient Descent algoritmasını kullanılan tree-based bir modeldir.

Hem regression hem classification üzerinde düşünülebilir. Regressionda tüm label ların ortalamaısnı alarak bunu yapar. Classificationda 1 classı na ait olanları 1’e, 0 class ına ait olanları 0’a çekmeye çalışır. Yine residual ı minimize etmeye çalışır.

**Karşılaştırma**: AdaBoost ve GBM in her ikisinde weak learner var. AdaBoost ta bir ağırlıklandırma var, GBM de bu yoktur.

**XGBoost (eXtreme Gradient Boosting):** Mantığı GBM ile aynı olmakla birlikte GBM in gelişmişidir. Overfittingi gidermek için ağaçları budama vardır. Modelin çalışma hızı için cash (ön bellek) kullanılıyor. Regularization (Ridge-Lasso) parametreleri var. Missing valueler ile çalışıyor, sıkıntı çıkarmıyor. Cross validaton yapar kendi içinde. Hızlı çalışır ve güzel skorlar verir.

Feature importance önemlidir (RF deki gibi). Black box model de denir, görselleştirmesi, tune etmesi çok zor bir modeldir, çok fazla parametresi var. Ferrari gibidir, yönetmesi zordur.

n-estimator(default:100): çalışacak olan ağaç sayısıdır. Learning rate arasında bir denge vardır, biri arttıkça diğeri azalır.

Supsample (default: 1.0): Kullanmak istediğimiz data miktarını oransal olarak gireriz. 1.0 yazılıdığında tüm datayı kullanır.

Max\_depth (default:6): ağacın derinliğidir.

Learning\_rate (default: 0.3):

**UNSUPERVISED LEARNING**

Label (target feature) yoktur. Clustering (kümeleme) ve dimensionality reduction modelleri vardır.

Clustering; customer segmentation da, recomendation systems(film vb önerisi), hedef kitle belirlemede kullanılır.

Clustering; benzer örnekleri bir araya getirerek gruplandırır. Yorumlama meselesi bize kalmaktadır, yorum aşamasında class isimleri domain knowledge ile belirlenmektedir.

Kümeleme adedi çok fazla olursa anlamlı bir kümeleme olmamış olur.

Clustering Algoritmaları:

* + 1. **K-means Clustering Theory** 🡪 nümerik veriler ile çalışır, kategorik veriler ile çalışmaz. Encoding yapıp denesek bile çok kaliteli bir modelleme olmaz.

Centroid tabanlı bir modeldir. K kaç seçilirse datayı o kadar bölmeye çalışır. Kümeler arasında Maximal inter cluster, küme içinde minimal intra cluster (Minimum contact maximum benefit) olmalıdır. Distance based bir modeldir, bu yüzden ihtiyaç halinde scaling yapılabilir modelleme öncesinde.

Dunn Index =**min(Intra-Cluster distance)/max(Inter-Cluster distance**)

<https://www.naftaliharris.com/blog/visualizing-k-means-clustering/> 🡪 interaktif görülebiliyor.

K’nın seçimini biz yapıyoruz ve bunun belirlenmesinde domain knowledge önemlidir. Ayrıca driven metodundan (elbow metodu) yararlanılır.

Evaluation: Supervised modellerdeki gibi değerlendirme metrikleri burda yoktur. Bunun yerine Clustering Tendency, Optimal Number of Clusters, Clustering Quality gibi yaklaşımlar vardır.

Clustering Tendency: Hopkins test ile ölçülür. Eldeki datanın clusteringe ne kadar yatkın olduğunu söylüyor bize. İyi bir clustering olması için datanın pattern oluşturması, uniform dağılım göstermemesi lazım. Hopkins testinin sonucunun 0’a yakın olmasını isteriz iyi bir clustering için. 0.5 ten büyük ve 1’e yakınsa clustering tendency için kümelemeye yatkın değildir deriz.

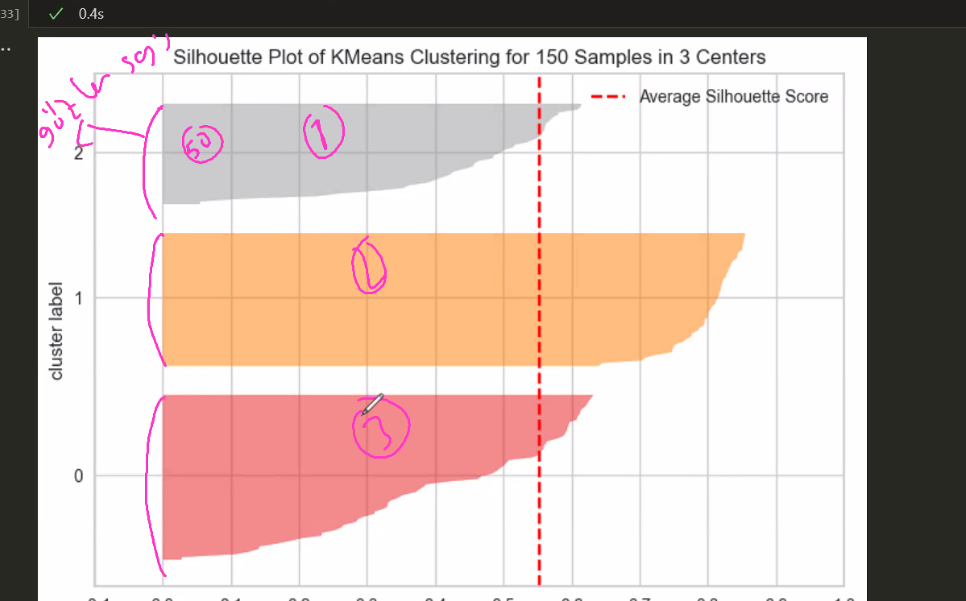
Optimal Number of Clusters: İdeal K nın seçiminde domanin knowledge çok önemlidir. Herhangi bir bilgimiz yoksa datadan yaptığımız çıkarımlar bize yol gösterecektir. Bu aşamada Elbow metodu kullanabileceğimiz metotlardan biridir. Grafiğin keskin düşüşün durduğu ilk noktayı K noktası olarak alacağız.

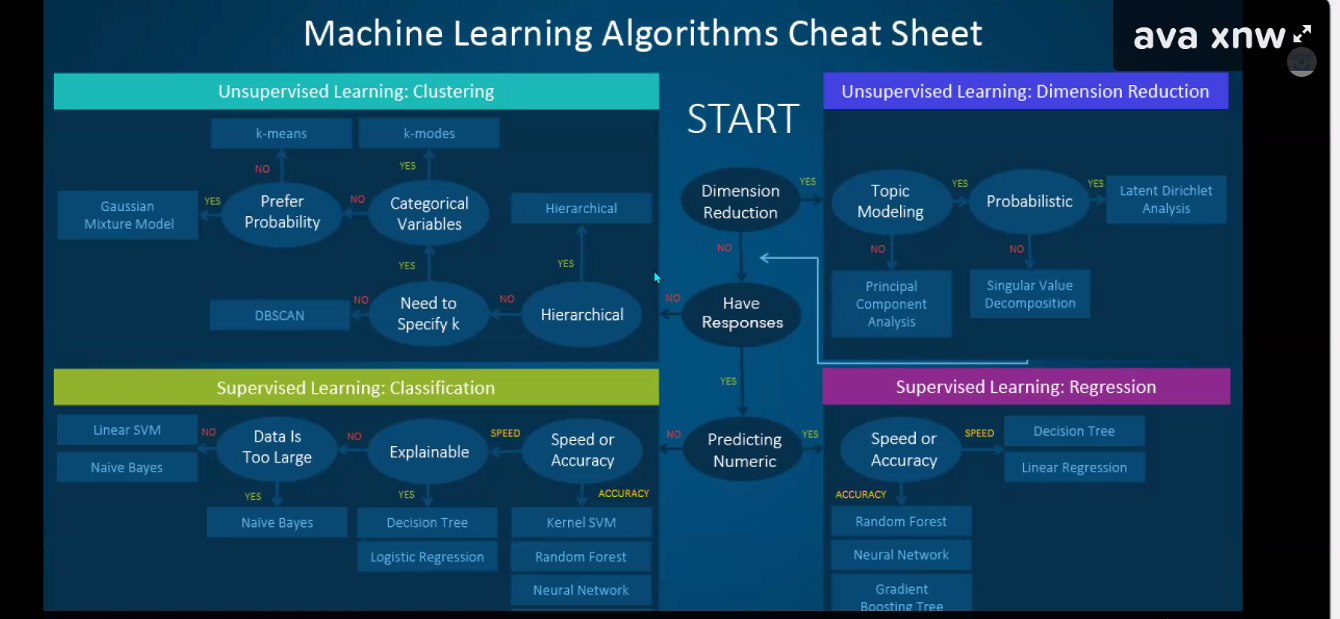
Clustering Quality: Label var (yani domain knowledge var ise) ve clustering isteniyorsa adjusted rand-index gibi metotlarla kümelememin kalitesini ölçeriz. Domain knowledge yoksa (label yoksa) Silhouette coefficient e bakarız. İnertia ya da bakarız.

Adjusted rand-index 1 e yakın olması iyidir. Silhouette coefficient in yüksek olması iyidir. Silhouette coefficient kümeler arası uzaklık ile ilişkilidir. İnertia nın (eylemsizlik) minimum olması beklenir, ne kadar küçükse centroidleri o kadar iyi çevrelemiştir.

Tüm hepsine ve pairplotlara bakıp hepsini harmanlayarak nihayetinde K yı belirleyeceğiz. Yaptığımız kümelemeleri açıklayabiliyor olmalıyız, ground truth olmamalı.

Feature sayısı az olduğunda modelleme daha sağlıklı olmaktadır, bu nedenle olabildiğince az feature kullanmayı tercih etmeliyiz.





* + 1. **Hierarchical Clustering** 🡪 cluster sayısı başta belirtilmez.

Aşamalar:

Datayı çok iyi tanımalıyız, Değerler continuous içeriyor mu, integer mi?--> model seçimi yapılır (hierarchical veya non-hierarchical). 🡪 Cluster number belirle. 🡪 Sonuçları karşılaştır. 🡪 Cluster çıktılarına göre modeli tekrar yorumla.

2 yaklaşım tipi vardır: Agglomerative (tümevarım- **Bottom-up approach**-daha çok kullanılır) ve Divisive (tümdengelim- **Top-down approach**).

Dendogram is a branching diagram that represents relaotionship similarity. Y ekseni kümeler arası distance yi gösterir.

Affinity (default: euclidian): Yakınlık demektir. Diğerleri manhattan, cosine, precomputed.

Linkage (default:ward - varyansa göre hesaplama yapıyor): Bağlantı demektir. Diğer parametreler: ward, complete, average, single. Bu önemlidir, buna dikkat edeceğiz.

K-means e göre daha yavaş çalışır. O nedenle küçük datalar için daha uygun denilebilir.

* + 1. **Principal Component Analysis Theory**

PCA futurelerin sıkıştırılarak yeni componentler oluşturur.

PCA future ler arası yüksek korelasyon ve lineer bir ilişki varsa iyi sonuç verir, düşük korelasyonda kötü sonuç verir.