**ÖZET**

**1.eksik veriler**

**2.**

1. **2.1.1 – eksik veriler**

Nan değerlere ortalama yazdırma

imputer = SimpleImputer(missing\_values=np.nan, strategy='mean')

1. **3.2.2 \_şablon\_veriler**

Veriler.csv Ülke yaş boy kilodan ciniyet tahmin eden lineer reg hale getirme

encoder: Kategorik to numeric

Sayısal olmayan verileri sayısal hale getirme

2 değişken varsa le yeterli-true/false 1/0

Ancak fazla değişken varsa ohe kullanmak gerekir. Çünkü aralarındaki ilişki doğrusal değil

le = preprocessing.LabelEncoder()

ohe = preprocessing.OneHotEncoder()

numpy dizileri dataframe donusumu-p*d.DataFrame*

dataframe birlestirme islemi -*concat*

#verilerin egitim ve test icin bolunmesi

ölçekleme-standar scaler- sc-veri benzeştirme

lineer reg

1. **6\_ECDF\_histogram**

ECDF histogram cizdirme datacamp

1. **7\_lineer reg**

Satışlar csv deki aylara göre satışlardan regresyon elde ettik

.Sort\_index()saçma değerlere gitmesin diye sıraladık

1. **8\_multireg**

Veriler.csv deki verilerden tahminler elde ettik model oluşturduk

Ve bunları karşılaştırarak sonuçlar elde ettik

import statsmodels.api as sm

#dummy variable

#model cinsiyet tahmin

# backward elemination,

# p value

model = sm.OLS(boy,X\_l).fit()

OLS tablosu

R2 değerleri

1. **9\_tenis v1\_v2**

Tenis verilerinden humudity i bulmaya çalıştık

-tüm tabloya le uygulama

veriler2=veriler.apply(preprocessing.LabelEncoder().fit\_transform)

genel uygulamalar

ikisi arasında ufak tefek farklar var incelenebilir.

1. **3.7.1 – şablon regresyon /3.3.3 polreg(silindi)** 🡪 **3.6.1 lin,pol, svr,dt, rf,silindi**🡪**3.7.1 ana şablon**

linner reg plot çizdirme

plynomal reg

lineer regresyonu poly ile girdilerin ^2 ^3 alarak daha doğru hale getirdik

tahminler yaptık single değer için

r2\_score

svr

decision tree

Z=X+0,5 – farklı değerlere ne sonuç veriyor model

Random forest

Model grafikleri

rbf-gausian

1. **4\_Titanic model evo sololearn**

Accuracy,TP TN gibi kapsamlı açıklamalar var

k-fold – train test spliti farklı farklı yerlerden ayırma

ikili girdili for döngüsü

accuracy

model.score

1. **3.5.1**\_svr support vektör regressiom 🡪 3.6.1 de devam ettiği için silindi

rbf-gausian

1. **10.3\_ödev8**

OLS grafik-p value

veriler.corr()-korelasyon

1. **4.9şablon sınıflandırma- sololearn model evo benzeri, ana şablon**

KNeighborsClassifier KNN

K değeri önemli, k 1 seçilirse tüm örneklerin yaşam sansıolur . k=n olursa %51 olan tüm uzayı domine eder.



Genel yaklaşım K değeri= karekök(eğitim boyutu(X\_train))/2

Örn: 1000 örneğimiz var ve%70 i eğitim içim ayırdık. K değeri kök 700/2

sklearn.svm import SVC

from sklearn.naive\_bayes import GaussianNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

# 7. ROC , TPR, FPR değerleri

Hangi veriler yanlış

#hangi veriler yanlış

df = pd.DataFrame(x\_test, columns=["0","1","2"])

df["actual"] = y\_test

df["predicted"] = y\_pred

incorrect = df[df["actual"] != df["predicted"]]

df["incorrect"]=df["actual"] != df["predicted"]

1. **13\_KMeans AgglomerativeClustering**

Gözetimsiz verilerde kullanılır. (belli bir sonuç yok. Survived-not survived gibi)

Rastgele noktalar belirleyerek cluster yapıyor.

Cluster sayısı belirleme önemli. Ona göre noktalar beliliyor. N\_cluster

Buna göre veriyi bölüyor.

For döngüsüyle en yüksek puanlı k nokta sayısı bulunabilir.

Agglomerative tüm clusterları ayrı ayrı bölüyor. Sonra en yakın 2 clusterı birleştiriyor. Sonra yakın clusterları birleştiriyor. Böyle böyle gruplar . çıkıyor.

Scatterla çizme

AgglomerativeClustering

dendogram

1. APRİORY 13.a

ECLAT

Equivalance class Transformation

Reinforced learning

1. **15\_Reinforced learning UCB**

**Dökümantasyon 14**

random.randrange(d)

1. 18\_NLP DDİ

Restoran yorumlarının iyi kötü olarak sınıflandırılması

Re.sub – cümle içinden istemediğimiz bölümü çıkarma

yorum = re.sub("[^a-zA-Z]"," ", yorumlar["Review"] [0])

karakter olmayan her şeyi çıkaraır

alfanumerik karakterleiri silme

büyük küçük harf değiştirme

stopwords – anlamsız kelimeler. Bunları bulma NLTK- tool kit

STEMMER-kelimeleri eklerinden ayırma

CountVectorizer – kelimeleri suffix matrix haline getirdi

0 1 lerden oluşan uzuuun matrix sonuçlarla train edilip sonuçlar elde edildi

1. **Sololearn 7\_neural Networks**

Creating Artificial Dataset- sistemden hazır database çekme

s= X[y==0][:,0] – y 0 olan tüm satırları seç

güzel bir ayırma var- (100,2) lik bir veri var. Bu veri içinde Y=0 olanlar ayrı x=0 olanlar ayrı olacak şekilde scatter yapıldı

MLP solver- yapay sinir ağlarındaki bir regresyon gibi bir yöntem

MNIST dataset- handwrite predicting

Drawing the Digits-matshow

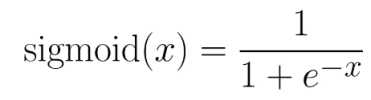
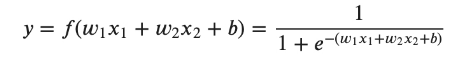
1x64 Bir data satırını çekip 8x8 hale getirip onu bastırdık. matshow

Yanlış tahmin ettiğimiz satırlar( incorrect = X\_test[y\_pred != y\_test])

Ve onların tek tek plotları

MLP weights

--------------------------------

Yukarıdaki formül bir sinir nöronun girdiye vereceği çıktı formülü

1. **19\_Yapay sinir ağları (YSA) artificial neural network(ann)**

ohe = ColumnTransformer – ohe yi direk data üzerinde uygulayarak işleri hızlandırıyor(?not: yalnız ayrı ayrı columlara dönüştümek yerine 2 lik sisteme dönüştürüyor.

Rectifier.

Optimizer – adam SGD skotastik gradient descending

Losses

classifier = Sequential()

dense

1. **20\_PCA Boyut indirgeme / LDA**

**Dökümantasyon 17**

Pca- Wine veri tabanını kullandık.

13 tane features olan bir kümeyi düzlem değiştirerek 2ye indiriyoruz

from sklearn.decomposition import PCA

başarı çok az düştü ama kullandığımız veri azaldı.

Pca gözetimsiz- tüm veriler aynı

LDA-gözetimli – data pointin hangi sınıfta olduğu önemli, sınıfları ayırır.

https://sebastianraschka.com/Articles/2014\_python\_lda.html

1. **21\_K-FOLD model değerlendirme grid search**

4\_titanc model evo da benzerini yaptık

k-fold

grid search

from sklearn.model\_selection import GridSearchCV

farklı farklı parametraleri denemek ve en başarılısını bulmak SVM parametrelerini tek tek bakmak

1. **22\_XGboost 19**

<https://xgboost.readthedocs.io/en/latest/>

yüksek verilerle iyi sonuç veriyor

hızlı çalışıyor

problem ve yorumun mümkün olması(çok önemli)

1. **23\_Model kaydetme ve çalıştırma.**

Urlden data indirme

Fit ettiğin veriyi kaydetme böyle tekrar tekrar fit etmeye gerek yok.

**DÖKÜMANTASYON**

Linearly seperable – tek bir doğru ile çözüm kümesi ayrılabiliyormu?

1. **Hatalı satırları bulma**

incorrect = X\_test[y\_pred != y\_test]

incorrect\_true = y\_test[y\_pred != y\_test]

incorrect\_pred = y\_pred[y\_pred != y\_test]

bunu data frame ekleme

--------------------

df = pd.DataFrame(x\_test, columns=["0","1","2"])

df["actual"] = y\_test

df["predicted"] = y\_pred

incorrect = df[df["actual"] != df["predicted"]]

df["incorrect"]=df["actual"] != df["predicted"]

s= X[y==0][:,0] – y 0 olan tüm satırları seç

güzel bir ayırma var- (100,2) lik bir veri var. Bu veri içinde Y=0 olanlar ayrı x=0 olanlar ayrı olacak

0.5 ten büyükleri 1 küçükleri 0 yapma

y\_pred=(y\_pred>0.5)

1. **Belli s**ayıdan küçük df i ayırma y nin 3 ten küçük olduğu satırları ayırdık (7\_neural net)

X5 = X[y <= '3']

Neural network coeeficents

print(len(mlp.coefs\_))  
print(mlp.coefs\_[0].shape)  
print(mlp.coefs\_[1].shape)  
  
Output:  
2  
(10, 50)  
(50, 4)

10 inputs / 50 nodes in the hidden layer / 4 outputs

---

y\_pred=(y\_pred>0.5)

0.5 ten büyük olanlar 1 olmayanlar 0

1. **1 den 10 a kadar yazdırma**

range(1,10) -plot if içinde kullanılabilir

np.arrange

1. **Dizi işlemleri:**

**Dizi oluşturma**

oduller=[0]\*d 10 elemanlı her elemanı sıfır olan bir dizi

np.arange

**iloc işlemleri**

X= veriler.iloc [:,3:-1] 3. Den sondan 1 öncekine kadar satırlar

**İNDEXİNG**

geo =veriler.iloc[:,4:5].values --🡪size: (1000,)

geo =veriler.iloc[:,4].values-🡪 size(1000,1)

neden böyle oluyor bilmiyorum ama size (1000,) olunca işlemlerde hata veriyor

yorumlar.shape[0]- kaç row var

x\_train**.sort\_index() –** index numarasına göre sıralama

1. **GÖRSELLEŞTİRME**
   1. X kümesi 2 sutundan oluşsun. Y\_tahminde bu satırların ait olduğu sınıflar. Her sınıfı farklı renkte gösterme için kod(c ile color aynı değil, color=”b” blue yapar)

plt.scatter(X[:,0],X[:,1],c=Y\_tahmin)

yada

plt.scatter(X[Y\_tahmin==0,0],X[Y\_tahmin==0,1],s=100 ,color="red")

plt.scatter(X[Y\_tahmin==1,0],X[Y\_tahmin==1,1],s=100 ,color="blue")

plt.scatter(X[Y\_tahmin==2,0],X[Y\_tahmin==2,1],s=100 ,color="green")

ikiside aynı ikincide y nin true olduğu x satırlarını yazdırıyor. Burada c ile color aynı

--------------------------

X = cv.fit\_transform(derlem).toarray()

1. **Fit Train transform**

gnb.fit(X\_train,y\_train)- x le yi karşılaştırıp öğrenir

gnb.predict(X\_test) – x teste ilgili kuralları uygulayıp tahmin eder

gnb.fit\_transform—

Sınıflandırma yapılmış ama hangi tahmin değeriyle yapılmış onu anlamak için kullanılır.

y\_proba = rfc.predict\_proba(X\_test)

from sklearn import metrics

fpr , tpr , thold = metrics.roc\_curve(y\_test,y\_proba[:,0],pos\_label='e')

print(fpr)

print(tpr)

1. **Katogorik Nümerik**

Aşağıdaki kod kullanılabilir.

le = preprocessing.LabelEncoder()

ohe = preprocessing.OneHotEncoder()

geo = ohe.fit\_transform(veriler.iloc[:,4:5].values).toarray()

geo = pd.DataFrame(data=geo , index=range(geo.shape[0]), columns=["fr","ge","sp"])

gender=le.fit\_transform(veriler.iloc[:,5:6])

gender=pd.DataFrame(data=gender , index=range(gender.shape[0]), columns=["male"])

X = pd.concat([veriler.iloc[:,3:4],geo,gender,veriler.iloc[:,6:-1]], axis=1 )

Tüm veriye label encode uygulama

veriler2=veriler.apply(preprocessing.LabelEncoder().fit\_transform)

1. **Pandas numpy**

.values dataframe ->array of object ?

Pd.DataFrame array of object->dataframe



Add a scalar with operator version which return the same results. Tüm tadaframe bir sayı ekleme

df.add(1)

X=Z+ 1 buda çalışıyor

1. **Regresyon bilgileri(Tahmin Metodları)**

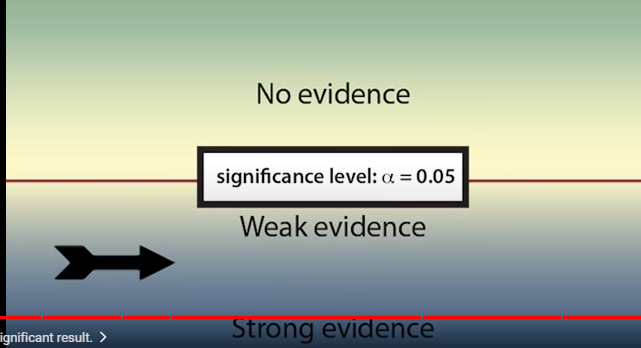
**Python kodu** 5 **8\_multi\_reg**

* 1. **P-value**

P değeri 0,05 altındaysa (küçüldükçe) H0 ın hatalı olma ihtimali artar(null hipotesis)

P değeri 0,05 üzerindeyse (büyüdükçe) H1 ın hatalı olma ihtimali artar(alternative hipotesis)

Backward eleminationda p 0,05 den büyük olanları sistemden atıyoruz. (tam tersi olması gerekmezmiydi?) tüm girdiler 0,05 ten küçük yada 0 olmalı

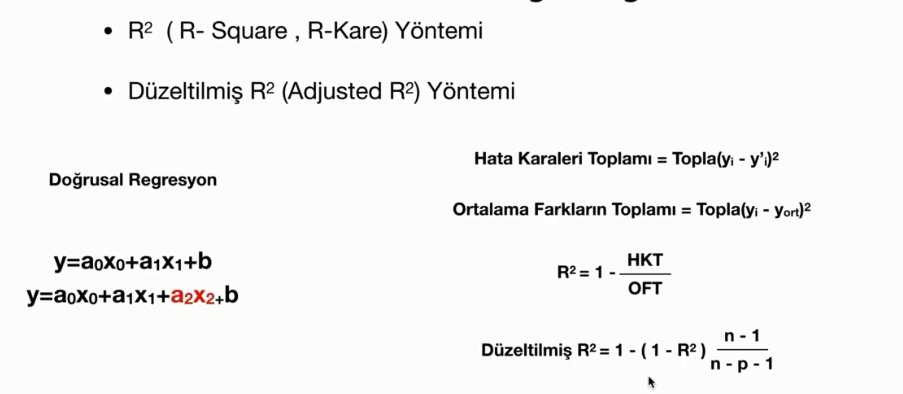
****

0.5 no evidance that H0 is wrong

0,05 weak evidance

0,001 strong evidance H0 is wrong

* 1. **R2 değeri (R square)**

****

Herkese ortalama değer yazsak R2=0 olur. En basit algoritma , R2 –sonsuza kadar gidebilir.

Mükemmel algoritma yazsak R2=1 olur. En iyi algoritma

Düzeltilmiş R2 yöntemi

Doğrusal regresyonda yeni bir feature eklendiğinde;

Eğer bu özellik olumlu bir etki yapıyorsa , R2 artar

Eğer bu özellik olumsuz bir etki yapacaksa , denklemdeki çarpanı 0 olur(neden?). bu nedenle R2 değeri değişmez.

Bu yüzden düzeltilmiş R2 var adjusted R2

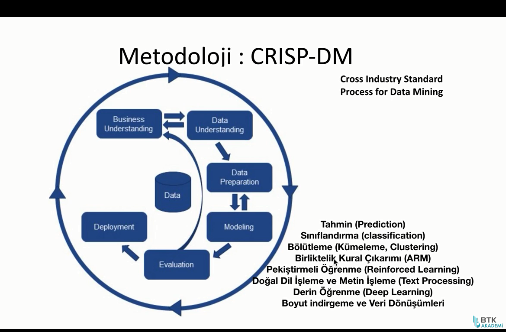
from sklearn.metrics import r2\_score

* 1. **Tahmin karşılaştırmaları**

**Model karşılaştırma**



1. **Tahmin Genel**





Ordinal- plaka numaraları, sıralanabilir ama büyüktür küçüktür ilişkisi yok

Nominal- araba markası, sıralama imkanı da yok

İnterval- aralıkta, çarpma anlamsız. Sıcaklık örneğin celcius cinsinden

Oransal - çarpılıp bölünebilir. Öreneğin yaş, oranlanabilir

Tahmin prediction- olup olmadığı ,aynı zamanda geçmiş verilerin tahmini

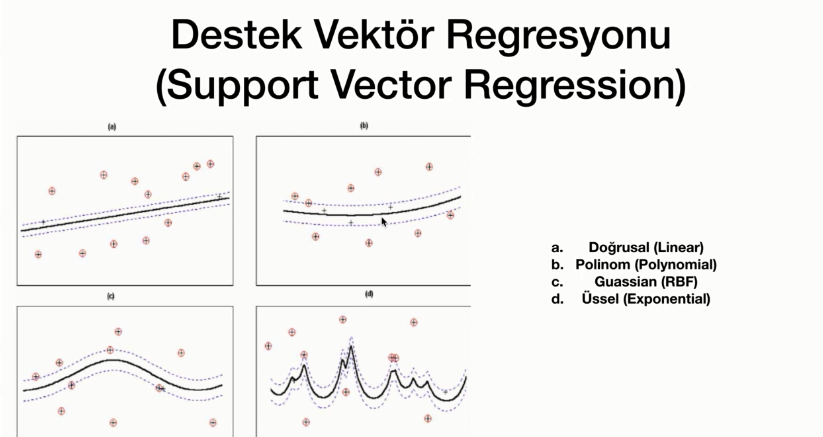
Forecasting – öngörü – geleceği tahmin etmek

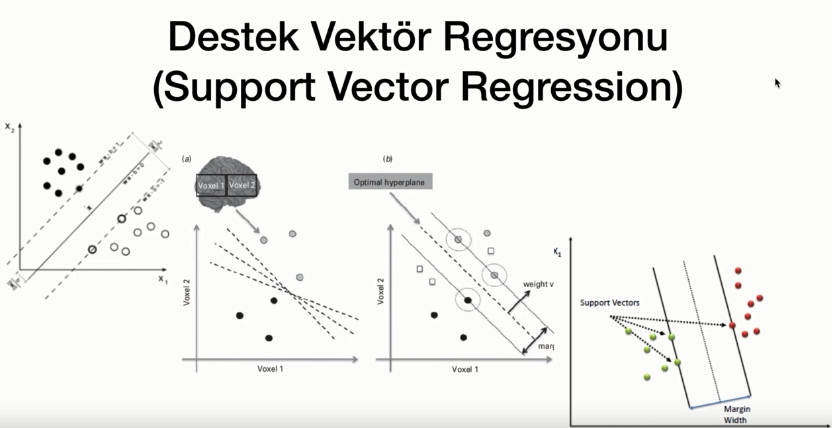
1. **Sınıflandırma –**

Python kodu - 11

Tahminden farkı: tahminde numeric bir değer tahmin ediliyor. Boy kaçtı gibi. Sınıflandırma nominal bir değer. Bilgisayar aldı yada almadı. Erkek kadın gibi.

* 1. Linear regeresyon - y = ax+b
  2. Support vektör regression 11





SVR

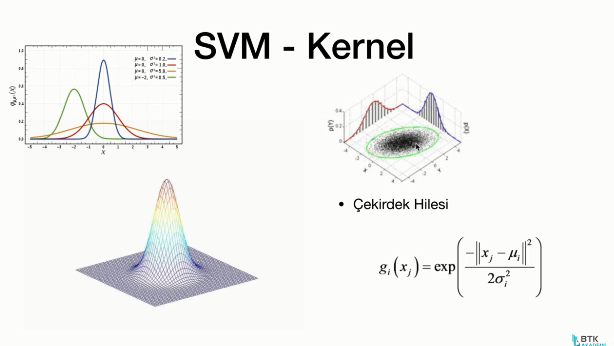
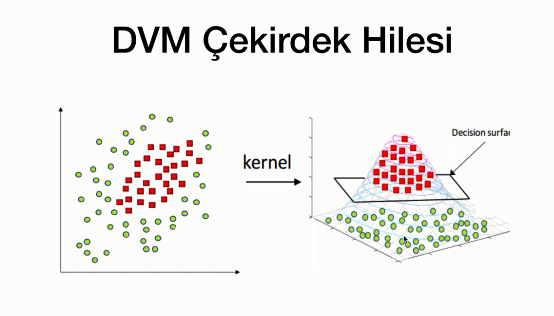
-Scaler ile kullanma zorunluluğu.

Random forest

-birçok decision tre enin birleştirilerek yapılmasıdır. Veri alt kümelere bölünür ve her biri için ayrı ayrı bir karar ağacı oluşturulur. İstenilen değer her ağacta denenir ve sonucun ortalaması alınır.

- farklı sınıfların bir **fonksiyon** ile ayrılabileceğini düşünür

* 1. Support Vektör machine
* Kernal trick

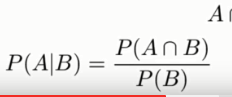
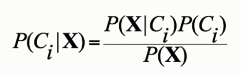


Doğrusal olarak ayrılamayan kümeleri ayırma. Boyut yükseltme. 3. Boyut ekledik. Merkez nokta berlirledik ve her noktayı uzaklığa böldük. 1 den fazla boyuta da bölünebilir. Nü değerleri önemli ve sigma

Sigma arttırarak çember çapı arttırabilir.

* 1. Naive Bayes

Python kodu - 11

bayes theorem

dengesiz veri kümelerinde çalışabilir.

3 yöntem var

-gausian naive bayes :tahmin edeceğimiz sınıf/kolon continious bir değerse (reel –ondalık)

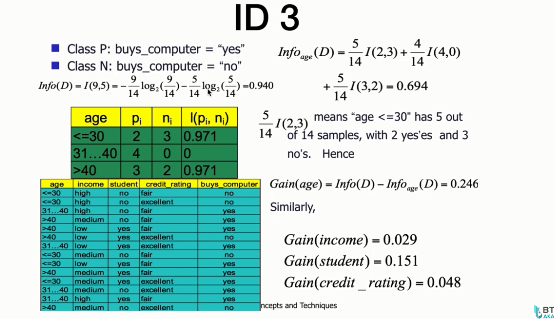
-multinomial naive bayes : tahmin edeceğimiz sınıf/kolon nominalse(erkek/kadın , üniversite)

-bernoulli naive bayes : ikili sistemlerde 1 veya 0 sa (sigara içiyor içmiyor)

* 1. **Decision tree**

Neyle başlayacığına karar vermek önemli. Burada entropi kavramı devreye girer. Aşağıda ID3 id3 algoritması





Age in informatiin değeri 0,694

Gaini en çok olan ile devam ediyoruz. Yes ve no lara gelinceye kadar soru sormaya devam ediyoruz.

-criterion “Gini“ de log2 tabanında değildir. Yukarıdaki entropy

* 1. Random forest
  2. Karmaşıklık matrisi

confusion matrix

A true positive (TP) is a datapoint we predicted positively that we were correct about.

A true negative (TN) is a datapoint we predicted negatively that we were correct about.

A false positive (FP) is a datapoint we predicted positively that we were incorrect about.

A false negative (FN) is a datapoint we predicted negatively that we were incorrect about.

(actual +) (actual -)

predicted positive TP FP

predicted negative FN TN

(total +) (total -)

skilearn reverse it (doğrulandı +1)

predicted - predicted +

(actual -) TN FP

(actual +) FN TP

(total +) (total -)

cm matrix te her satır toplamı gerçek değer,

sutun toplamı tahmin edilen değer

pred 0 pred 1 pred 2

actual 0 3 2 1

actual 1 0 5 2

actual 2 1 3 4

0 dan gerçekte 6 adet var

1 den gerçekte 7 adet

2 den gerçekte 8 adet

biz 4 adet 0 tahmin etmişiz, bunların 3 ü doğru 0(TN - 0 negatif) 1i yanlış 2

biz 10 adet 1 tahmin etmişiz bunların 5 i doğru (TP- 1 pozitif çünkü)2 tanesi 0 3 tanesi 2.toplam 5 yanlış

biz 7 adet 2 tahmin etmişiz bunların 4 ü doğru. 1 tanesi 0 ,2 tanesi 1 toplam 3 yanlış

"""

#calculating metrix\_score,

accuracy (.score , accuracy\_score(x,y) )- percent of predictions that re correct

Of 10000 credit card chards, we1 have 9900

legitimate charges and 100 fraud1ulent charges.

I could build a model that just predicts that

every single charge is legitimate and it would

get 9900/10000 (99%) of the predictions correct!

**accuracy**=(TP+TN)/total

eror = 1 - accuracy

precision refers to the percentage of positive results which are relevant

TP/predicted yes

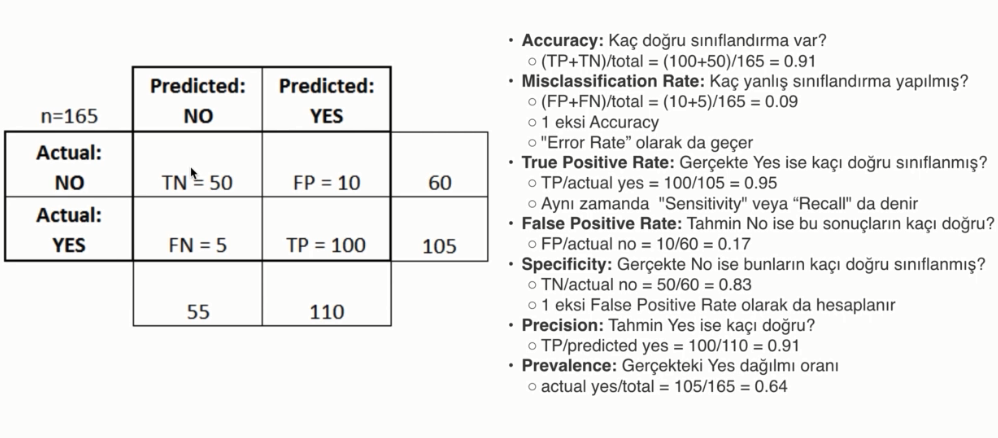
**Recall(sensitivity) TPR- (true positive rate)** is the percent of positive cases that the model predicts correctly.

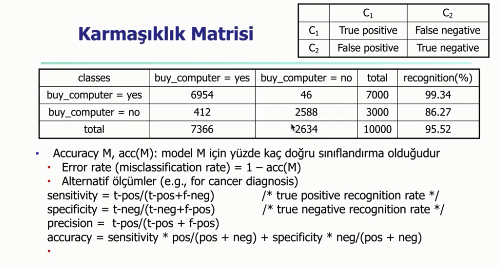
=TPR=TP/(TP+FN)= TP/Actual yes

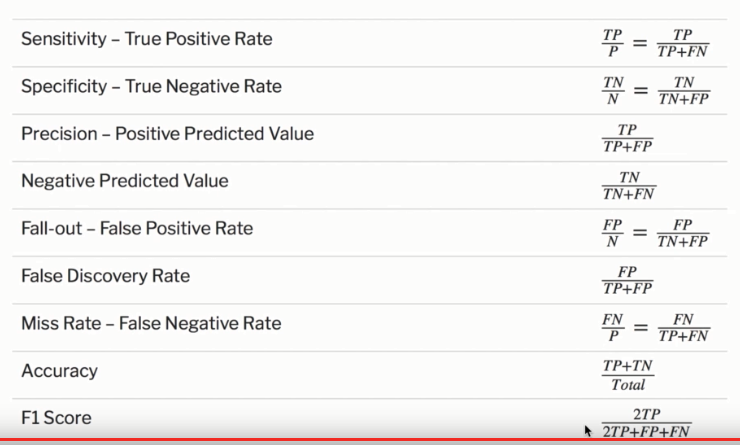
**False positive rate**, specisify, FPR = FP/Actual no

**prevalence** : actual yes/total – 0,5 demek dengeli bir dağılım olduğunu gösteriri. 0 yada 1 e giderse bir grup daha baskın demektir.

F1 - 2\*(precision\*recall) / (precision+recall)







* 1. **Accuracy paradoxu**

ZeroR algoritması : hangi sınıf daha çoksa tüm tahminlerin, o sınıfa ait olarak yapıyor. Nadir hastalığı tahmin ederken “hasta değil” demesi gibi.

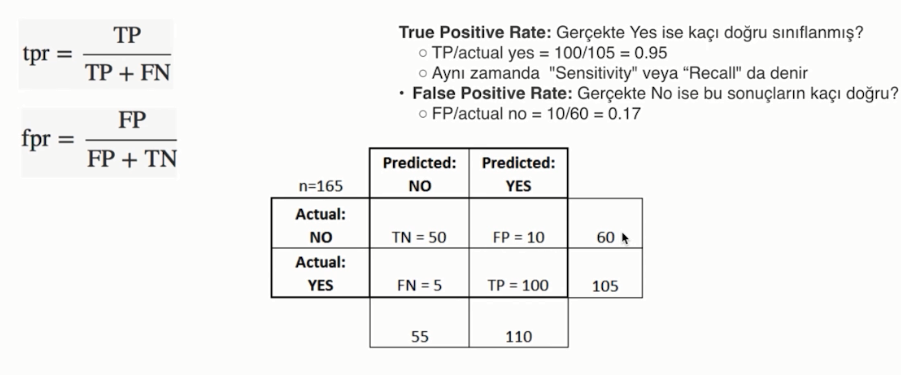
Aslında accuracy %99 ama algoritma akılsız

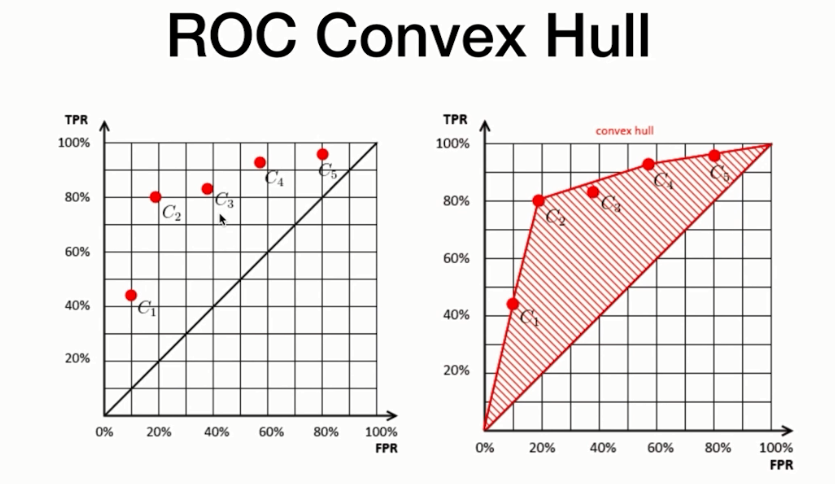
Accurcy hersey demek değil.

Algoritmanın zeroR ı geçmesi olması lazım.

* 1. **ROC eğrisi**

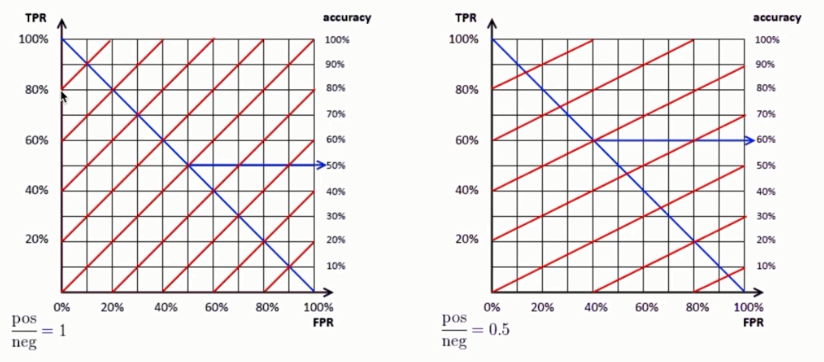
Receiver operating characteristic





Roc eğrisi- çizginin altına inmesi mantıksız.

C3 algoritması avantaj sağlamaz c2 veya c4 tarafondan domine edilir.

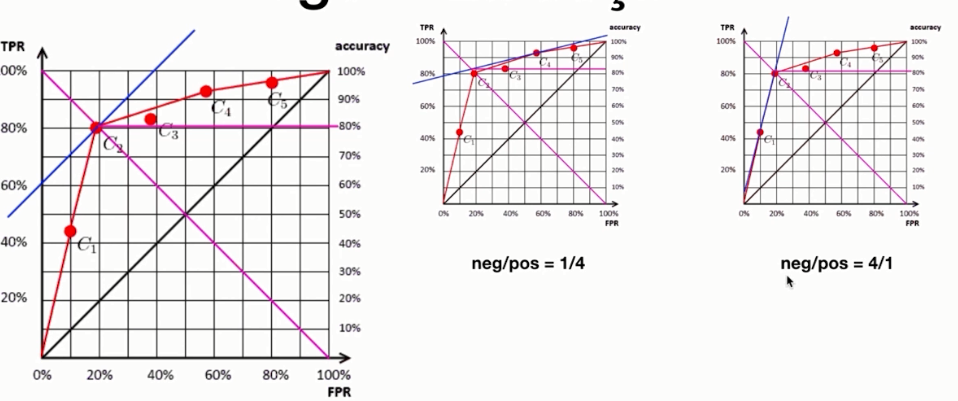


Prevelance değerlerinin faklı olması kırmızı eğrilerinin açılarını değiştirir.

Kırmızı eğrinin mavi çizgiyi kestiği nokta accurucydir.

Dolayısıyla;

Kırmızı çizgi üzerinde accuracyler aynıdır.



1 nci örnekte prevelance 1 olduğu için en yüksek accurcy c2 algoritması

2nci örnekte c4

3 n cü de c1

* 1. **AUC Curve**

Area under curve TPR-FPR eğrisi altında kalan alan

Ne kadar hızlı öğrendiğimizle alakalı

Hesaplaması zor bir yöntem

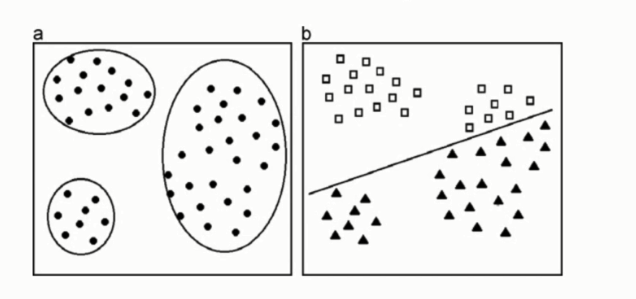
Ne kadar büyükse başarı o kadar büyüktür

1. **Bölütleme / kümeleme / Clustring**

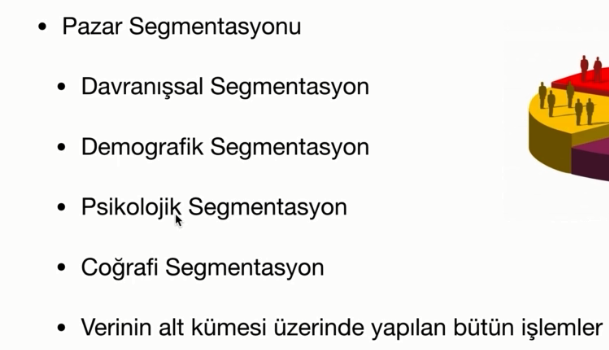
Sınıfladırma ile en temel fark Gözetimsiz bir veriye sahip olmasıdır.- **Gözetimsiz Öğrenme**

Sınıflandırmada veriler bellidir. Erkek kadın, ügen daire, gibi.

Clustringde verilerin sınıfları belli değil;  
örneğin müşteri segmentasyonu(ön tanımlı veri yok). Permium müşteri , orta gelirli müsteri diye ayırıp, Premium müsteriye x ürününü satabilirim. Metroya giden müsteri.sahtekarlık







Bunları bir şekilde makine kendisi öğrenebiliyor. Görüntü işleme. Video işleme

Temel clustering yöntemleri

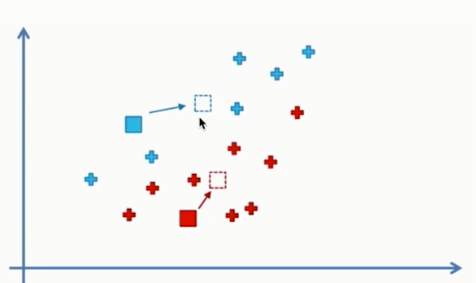
* 1. **K-Means**

**Python kodu 13\_Kmeans--- 12**

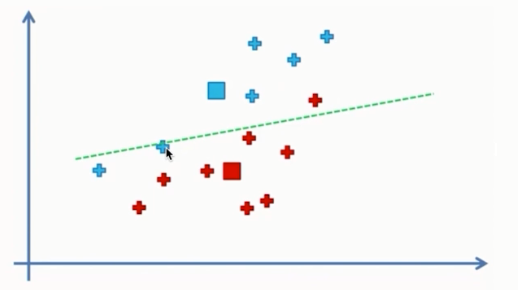
Kaç küme olacağını seçmek önemli

Rastgele iki nokta belirledi ve tam ortasından doğru çekilir ve ilgili kümeye atanır

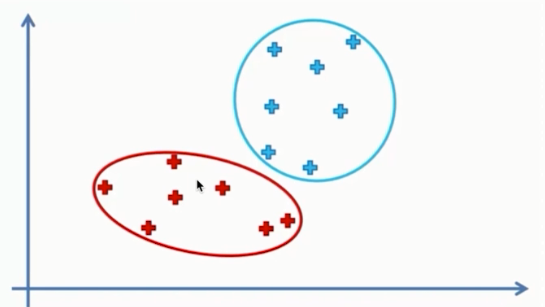
İki sınıfın ağırlık merkezine(cluster orta noktası) doğru kaydırılır ve kümeleme yapılır

fg

Tekrar yeni kümeleme oluşur. Sonra tekrar merkeze kaydırılır.

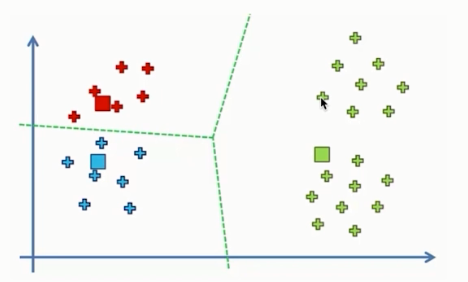
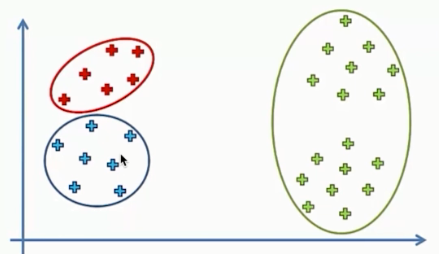


En stabil hale gelinceye kadar kaydırılır ve clustering biter



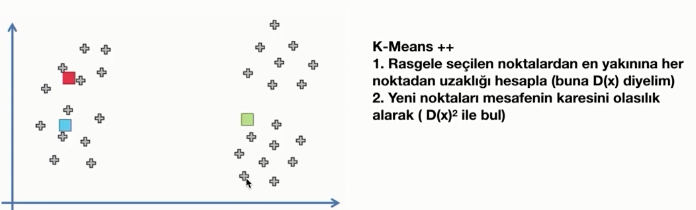
* + 1. Rastgele başlangıç tuzağı

Noktalar başlangıçta rastgele atanınca hatalar oluşuyor. Yanlış bölüyor. Gözle bile daha iyi ayırabiliriz bazen.

Çözüm algoritması var

K-Means+ algoritması:



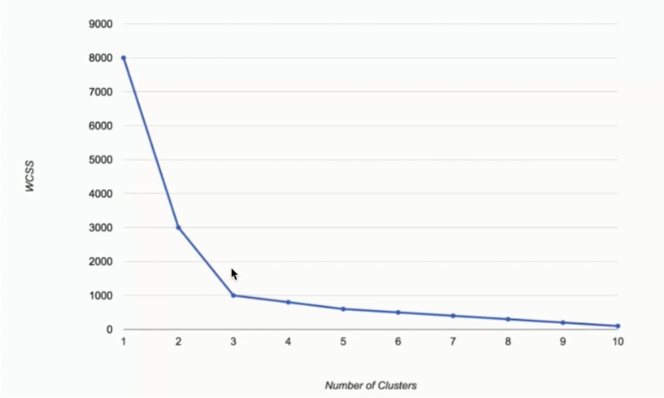
Ağırlık noktasını kaydırırken kendisine uzak olan noktaya kayma ihtimali daha yüksek çünkü uzaklığın karesini alarak değerlendiriyor.

Böylece cluster arasında atlama olabiliyor. K means atlayamaz çünkü kendi clusterın ağırlık noktasına gidiyor. K means+ ta uzaklığın karesine gittiğinden gidebilir.

* + 1. paremetreler

Kaç kluster olacağını belirlemek önemli. Bu formulle cluster başarısı ölçülebilir. Within-cluster sums of squares.her noktanın ağırlık merkezine olan uzaklığının karesi toplanıyor. 1 cluster için hesaplanıyor. Sonra 2 cluster için hesaplanıyor. We aşağıdaki tablo oluşuyor





WCSS cluster arttıkça düşüyor doğal olarak çünkü mesafeler küçülüyor. Cluster sayısı data point ile eşit olursa 0 olur doğal olarak. En optimum nokta dirsek yaptığı yer.

<https://scikit-learn.org/stable/modules/clustering.html>

wcss in başka bir hesaplama yöntemide inertia(beklide aynısıdır)

inert=[]

for i in range (10):

KM2=KMeans(n\_clusters=(i+1))

KM2.fit(X\_scaled[:,0:2])

inert.append(KM2.inertia\_)

plt.plot(inert)

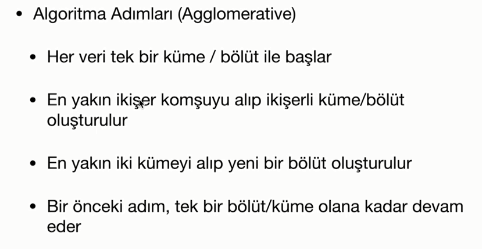
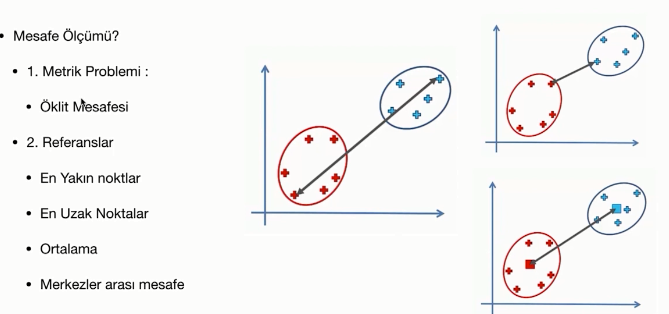
bu şekilde bulunan inertilar dirsek yaptığı yerden uygun şekilde bölütleme yapılabilir.

* 1. **Hiyerarşik kümeleme/bölütleme:**

iki yöntemi var

* + 1. Agglomerative

Her nokta ayrı küme, en yakınlar birleştiriliyor. Mesafeler ölçülüyor. Ona göre birliştiriliyor. Mesafe ölçümünün nasıl yapılacağı önemli.

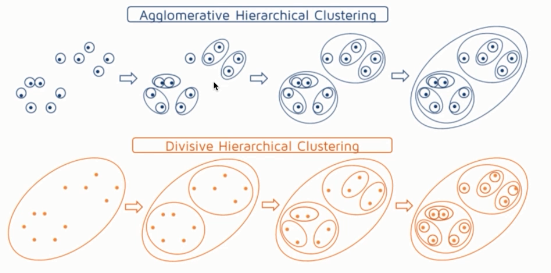
 

Affinity mesafenin nasıl ölçüleceği euclidean olabilir

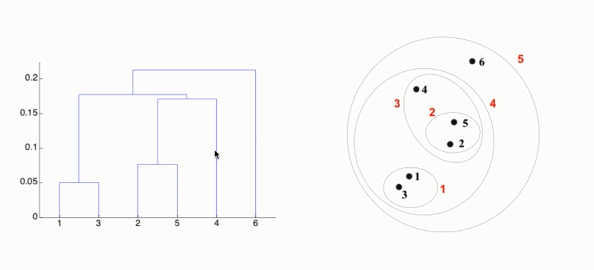
Linkage

* + 1. Divisive

bütün veri bir küme sonra bölünüyor.

****

* + 1. **Dendogram**

****

Dendograma bak , aradaki en uzun mesafeden böl.

En optimum yer mesafenin em uzun olduğu yer.

Uzaklıkm ölçme metodların önemli

* + 1. **Hiyareşik bölütleme genel**

Scipy kütüphanesinde kullanılabilir

1. **Birliktelik kural çıkarımı – association rule mining- ARM – ARL**

**Python kodu 13**

associacion rule model- bunu alan bunu alır(örn: bebek bezi alan, bira da alır) netflix de bu algoritma

Bebek bezi alanlar bira alır o zaman kampanya yapalım mantığı.

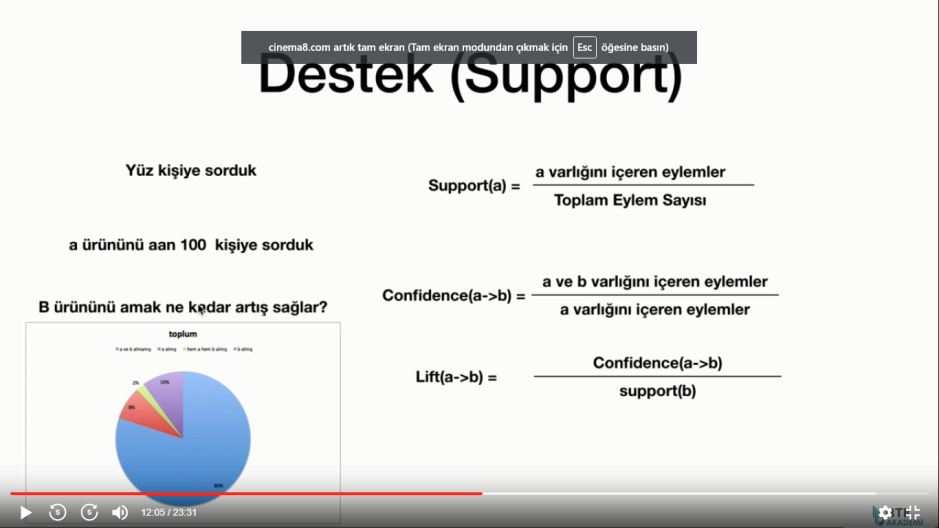
İlişkiyi bulmak önemli

corolation or causality

köpek balık saldırısı artınca dondurma satışı da artıyor (corolation)

havalar ısınınca dondurma satışı artıyor (causality)

**Destek (support)**

****

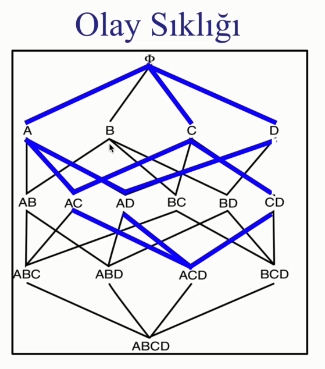
Lift 1 den büyükse- a ürünün üzerine odaklanmak b ürünü satışınıda arttırır. Bu örnekte lift=1,66

Lift 1 den küçükse- a ürününün alınması b ürününü olumsuz etkiliyor

Sklearnda yok

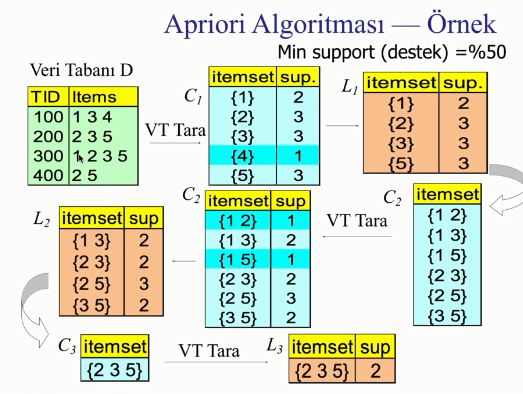
* 1. **Apriori**

**Python kodu 13**

****

En düşük olan b yi eledik

Frekansları sayar



Support değeri max/2 =1.5 yani frekans 3 iken c1 ve c2 de olduğu gibi frekansı 1 olanları eleyeceğiz

2 yi alan adam neyi alır-> 5 i alır çünkü l2 de en yüksek frekans

3 ü alana ne tavsiye ederiz-> 1,2,3 l2 de . 3 ü alanlara 4 ü tavsiye etmenin bir anlamı yok

****

* 1. **Eclat**

Eclat daha hızlı çalıyor

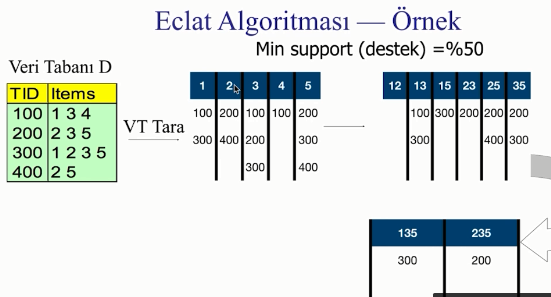
Eclat Derin öncelikli, apiori satır satır

Her ürünün hangi transectionlarda geçtiğini yazıyoruz. Mesala 1 numaralı ürün 1 ve 3 de geçmiş

Sonra 2 li 3 lü

Ortak elemanlar birleşiyor

1 3 ün heaplanması için 5 e ihtiyaç yok.



* 1. **fp growth**

1. **Reinforced learning**

**Python kodu : 14**

Makinanın yeni şeyler öğrenmesi. Kendi kendine . daha iyiye gitmesi

Alpha go – go şampiyonunu yenen makine bu mantıkla. En iyi oyunları analiz ederek

tek kollu canavar testi: jack pot makinaları var hangisi bana daha çok kazandırır ona bakıyoruz

A/B testi- reklam tıklama testi

THY reklam tıklaması, hangi reklam daha başarılı

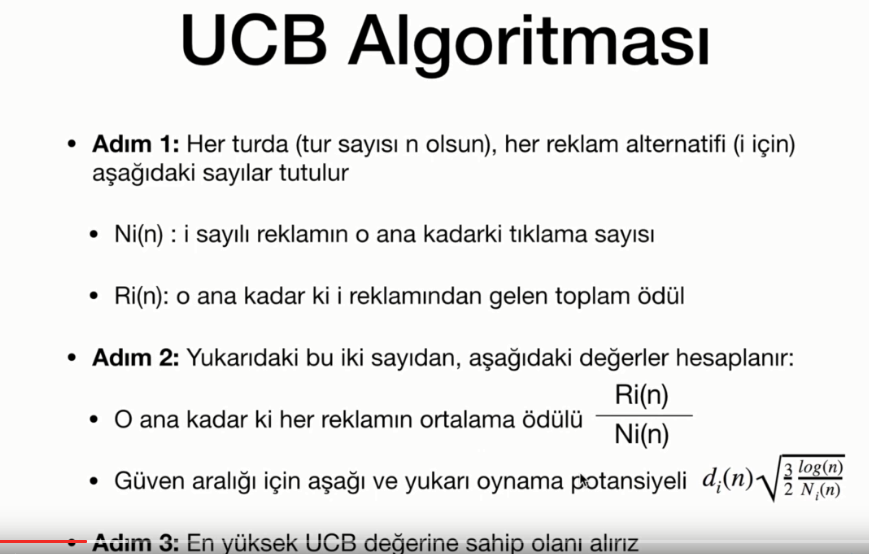
* 1. **UCB Upper confidance** selection

UCB( upper confidance) –ödül kazanma, kazanamama

5 farklı reklam var 1 tanesi daha başarılı başarılı olanı göstermeye çalışıypruz

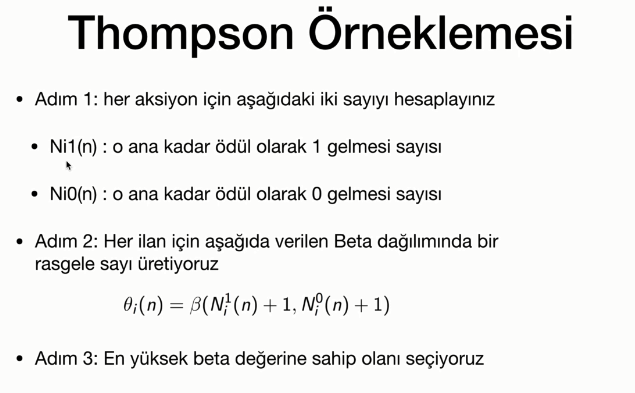
En fazla tıklanan reklamı bulmak

Random selection rastgele seçim

****

* 1. **Thompson Algoritması**

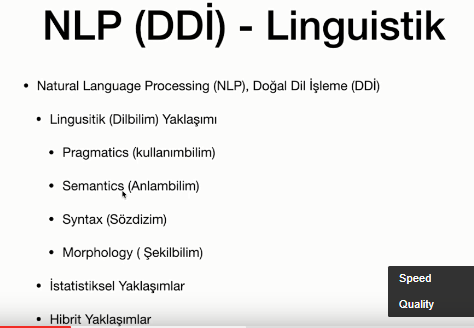
UCB ye çok benziyor,

****

1. **Doğal Dil İşleme NLP Natural Language Processing**

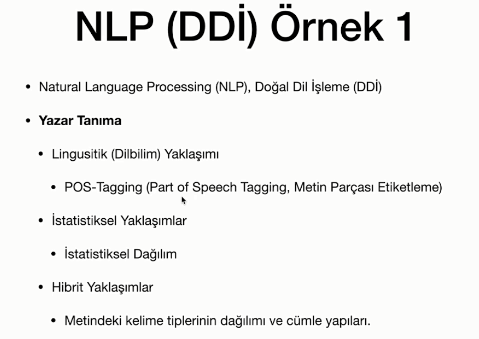
**Python kodu 15**

****

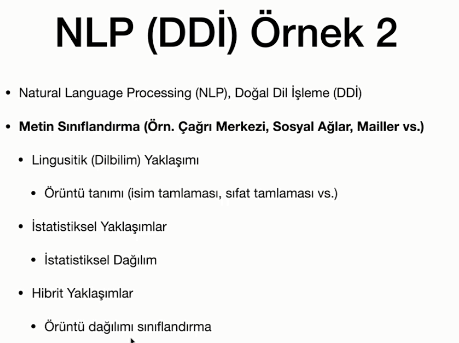
****

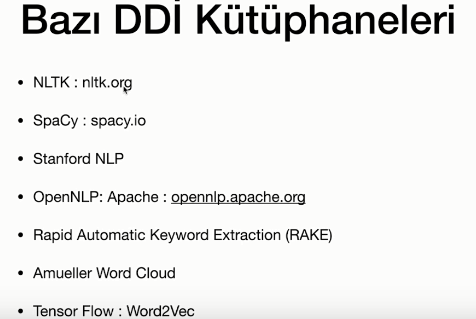
****

**TF-IDF :** yaygın bir kullanım.

****

Metin içindeki sıfat filleri sayma, yazarın imla hataları olabilir.

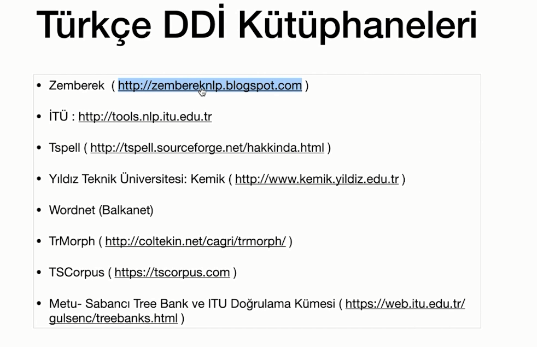




NLTK-

NLP- apache kullanılıyorsa

Tenserflow-



Zemberek- morfoloji seviyesinde

İTÜ- hazır tool kaynak var

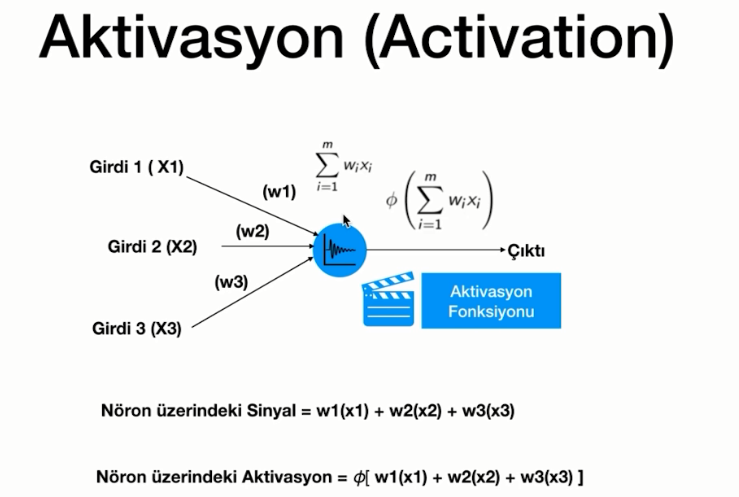
* 1. **Genel DDİ Problemleri**

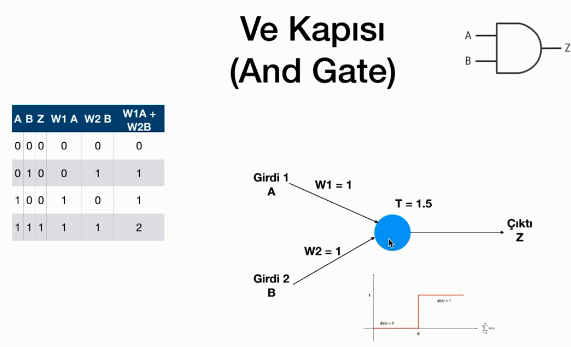
Diğerlerinden farklı olarak feature extraction yapılmalı. Çünkü veriler hazır değil

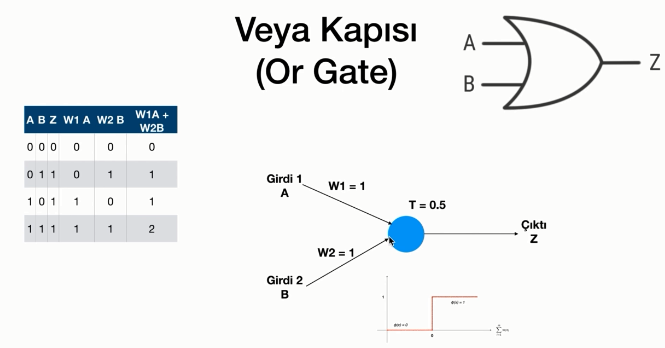
Veri ön işleme önemli çünkü gerisi zaten benzer. Öznitelik çıkarımları.

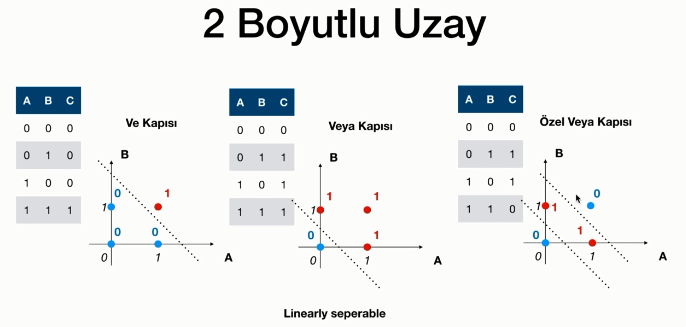
* 1. **Py**

1. **Yapay Sinir Ağları**





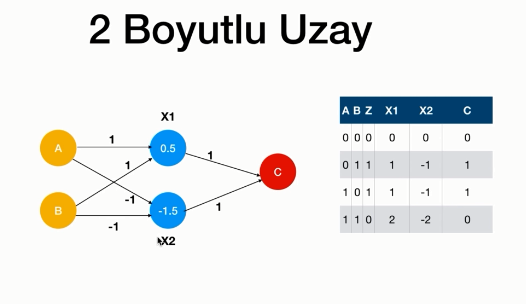




Tek bir doğru ile ayırılabiliyorsa lineear seperable

Tek bir doğru ile ayrılamıyorsa linearly seperable değil

Bu yüzden xor 1 noronla ayrılmaz çözüm aşağıda.



Perceptron- geri yansıtma

Önce ağırlıkları değiştiriypruz

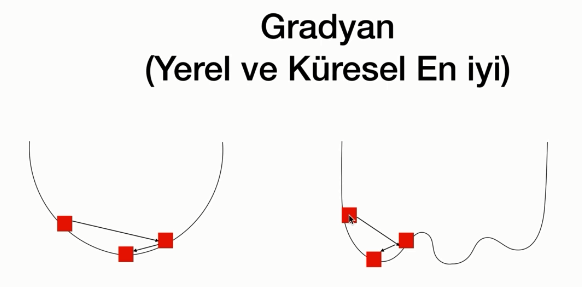
Amaç ağırlıkları bulmak, eğitmek. Bir yerden başlayıp zamanla kendimizi eğitrek başırayı ağırtmak.

Ağırlıklara ceza vererek katsayılar değişir

Gradyan alçalış (gradiend descendent)- belli adımlarlar la optimun-m noktaya iniyor. Ama çok daha iyi bir noktayı bulmamızı engeller.

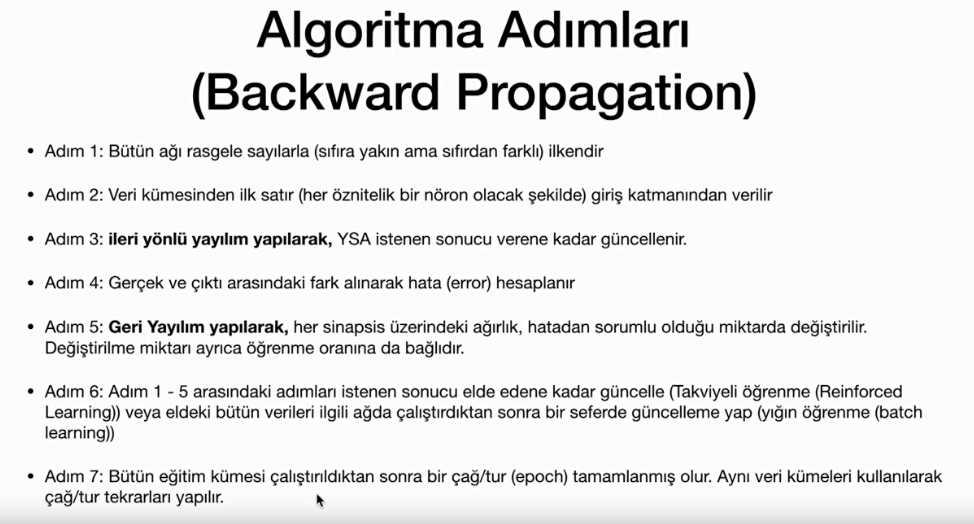
Stochastic gradient descendent- yukardaki kötüleşen durumu biraz daha iyi hala getirir.

Satır satır alıp değerleri güncellersek stochastic yaklaşım, 100 satırı okursak batch(yağın) yaklaşım





Back propagation: ( en önemli yaklaşım)- kaç tekrarın olacağını belirlemek önemli. Learning rate, ne kadar öğrendiği. Epox? Kaç defa de öğrenceği.



Derin öğrenme kütüphaneleri:

* Pytorch-
* Caffe-
* Keras- tensorflow üzerinde hızlı işlemler
* tensorFlow- google detaylı
* Theno – işlemleri gpu ya dağıtma
  1. **Kodlama**

Katman DENSE,

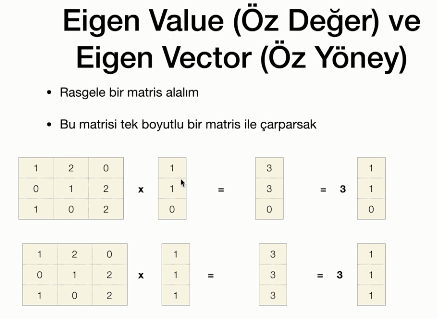
Keras dökümantasyon önemli

Gizli katman sayısını bulmak için giriş ile çıkışın ortası alınabilir.

Gizli katmanlarda lineer(relu) çıkış katmanında sigmoid

init=initilations

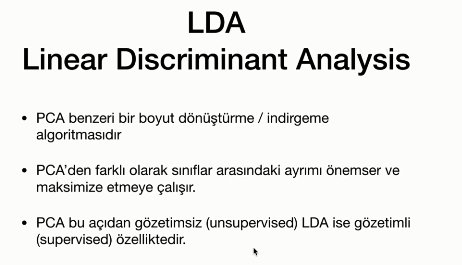
1. **Boyut indirgeme**
   1. **PCA principal component analysis**

****

* 1. **LDA linear discriminant analysis 18**

**LDA için** veriler sınıflandırılmış / etiketlenmiş olacak

LDA amaç, sınıflarları birbirinden ayıran en iyi boyutu bulmak

PCA amaç,verileri birbirinden ayıran en iyi boyut****

https://sebastianraschka.com/Articles/2014\_python\_lda.html

1. **Model Seçimi**
   1. **K\_fold-Cross Validation**

1. estimator : classifier (bizim durum)

2. X

3. Y

4. cv : kaç katlamalı

'''

basari = cross\_val\_score(estimator = classifier, X=X\_train, y=y\_train , cv = 4)

print(basari.mean())

print(basari.std())

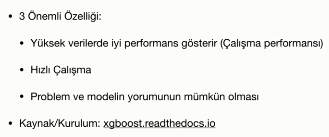
* 1. **Grid search**

Parametreleri optimize etmek için

Örneğin kodda svm in c: değerlerini vs hepsini deneyecek. Dahabaşarılı hala hale getirmek için

Accurcy biraz daha arttı . c=1 gamma ve rbf dedi. Svc parametreleri için

1. **XGBoost**

****

<https://xgboost.readthedocs.io/en/latest/>

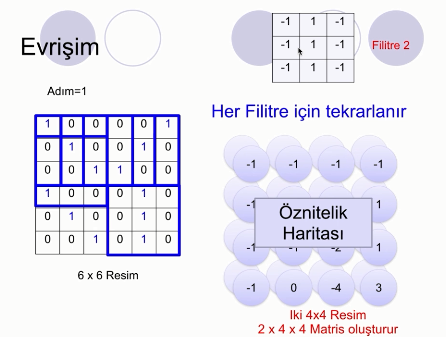
yüksek verilerle iyi sonuç veriyor

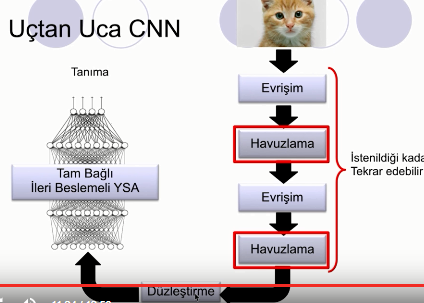
hızlı çalışıyor

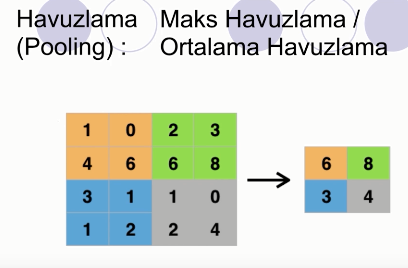
problem ve yorumun mümkün olması(çok önemli)

1. **CNN (Convolutional Neural Network)**

Keras üzerinden devam ettik

****

****

****

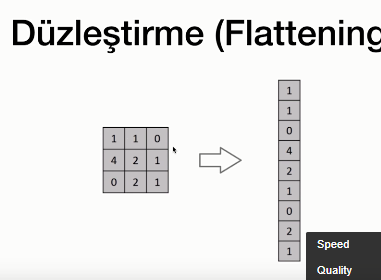
****

İmage data generatör

**BAŞVURULAR**

df.add(1) 8.b

R2 Value 9.b

Tahmin karşılaştırmaları 9.c

Support Vektor Machine 11.b, 7, 11

naive bayes 11.d

decision tree 11.e

accurucy paradoxu 11.h

ROC Curve 11.i

y\_proba = rfc.predict\_proba(X\_test) 6

hangi veriler yanlış 11

k-Means 12.b

Hiyerarşik kümeleme/bölütleme 12.b

Dendogram 12.b.ii