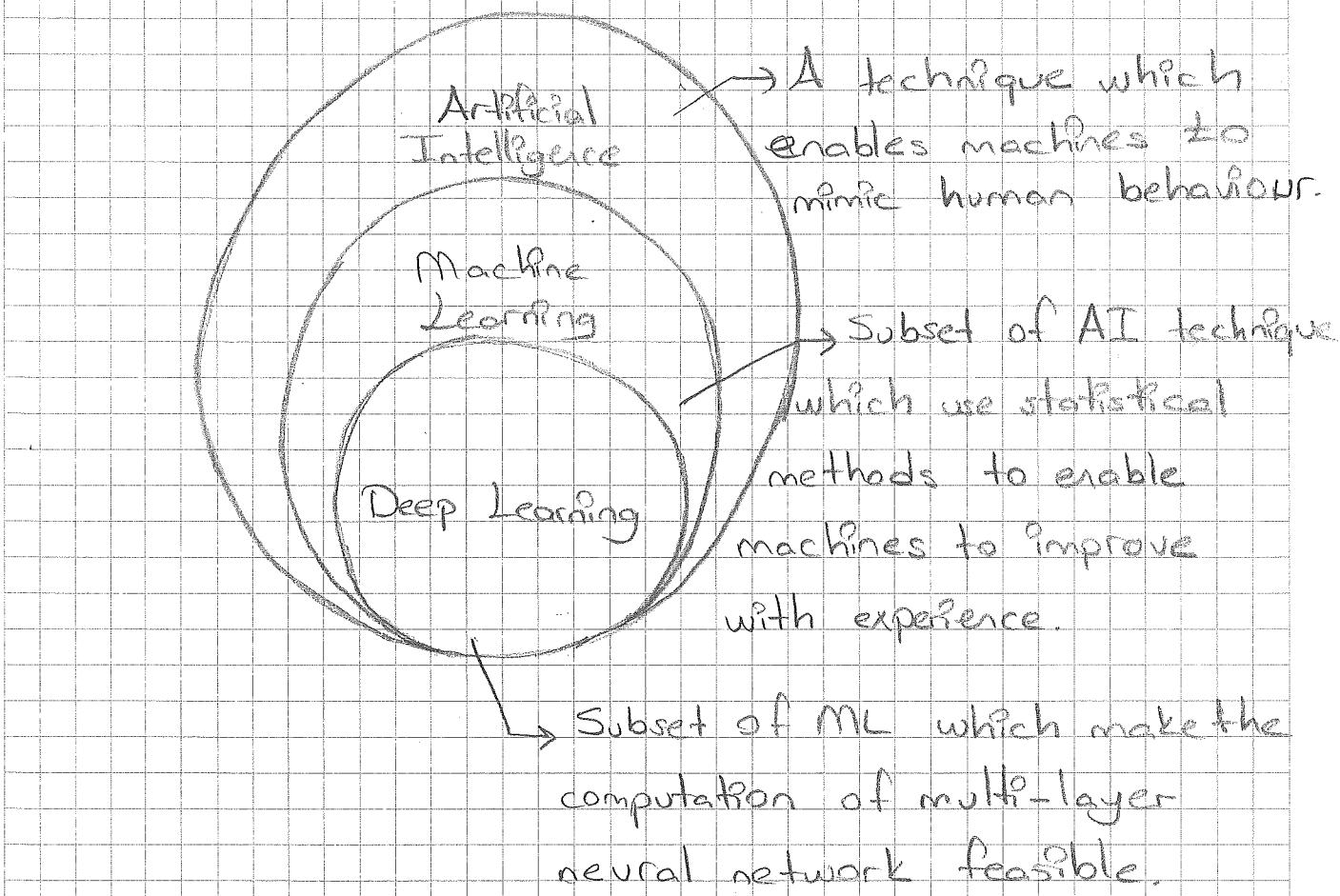


PRE-CLASS

Machine Learning 1959 yılında Arthur Samuel tarafından tanımlanmıştır.



ML Types

Supervised Learning

① Classification

② Regression

Unsupervised L.

① Clustering

② Dimensionality Reduction

Reinforcement L.

Kursta okşını göreceğiz.

Supervised ⇒ Labeled

Train data var

Unsupervised ⇒ Unlabeled
Train data yok

Machine Learning Terminology

Observation \Rightarrow Rows (E-mail)

Features \Rightarrow Columns (zincir, yes, coniugat...)

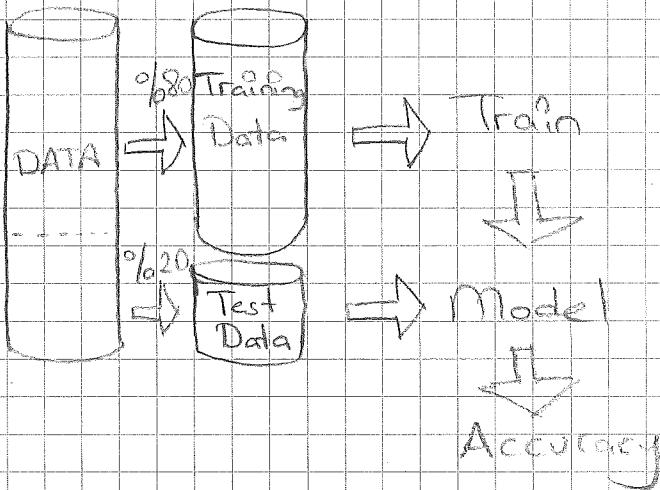
"
Variables = Attributes

Labels \Rightarrow Target column (spam-eposta, ev fikri)

\hookrightarrow The thing we're predicting.

Training Data \Rightarrow Bir modeli eğitmek için kullanılan tüm data setinin alt kumesi.

Test Data \Rightarrow Modeli test etmek için kullanılan tüm data setinin alt kumesi.



Training :

$$y = m * x + b$$

output slope input y-Intercept

7 Steps of ML

- Gathering data
- Preparing that data
- Choosing a model
- Training
- Evaluation
- Hyperparameter Tuning
- Prediction

SUPERVISED LEARNING

It is the process of learning from labeled observations.

Label'lar algoritmayla observation'ları nasıl eğiteceğini öğretir.

Supervised Learning $\xrightarrow{\text{Classification (Discrete)}}$

$\xrightarrow{\text{Regression (Continuous)}}$

Classification: Her observation bir category/class atar.

(Spam / not spam gibi)

Class'lar discrete (ayırlık) 'tir.

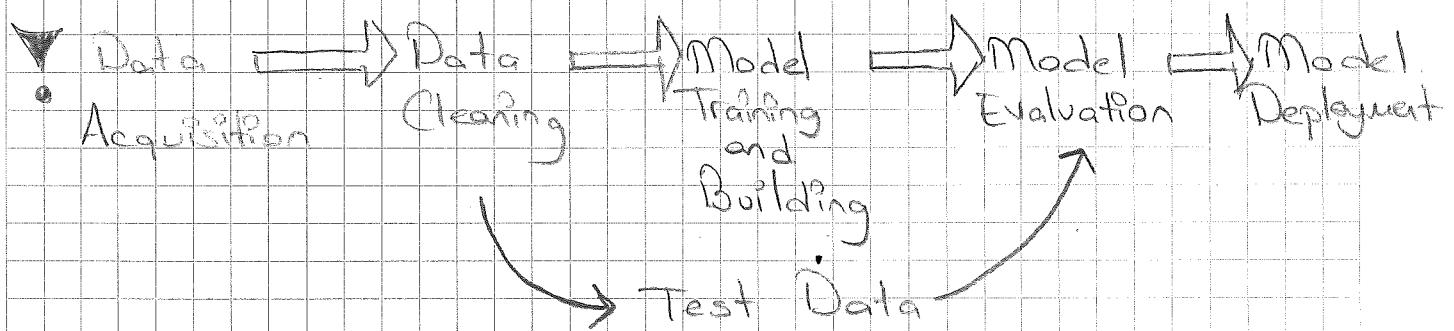
Regression: Observation'lardan "öğrendiklerine göre"

bir değer tahmin eder.

Bir feature'in
diğer feature'i
nasıl etkilediğine
bakılmış.

("4 yatak odası ve 3 banyosu olan bir ev
245.000 \$ olmalıdır." gibi)

! Pratik ML modellerinin çoğu Supervised Learning
kullanır



Correlation: İki nicel değişken arasındaki ilişkinin yönünü
ve eğimin gücünü verir.

-1 ile +1 arası da değişir.

STRONG
-1

NO RELATION
0

STRONG
+1

Direction	→ Positive → Negative
Strength	→ Weak → Strong

Linear Regression,

Simple Linear Regression : A linear regression model with a single explanatory variable.

OLS (Ordinary Least Squares) : A type of linear least squares method for estimating the unknown parameters in a linear regression model.

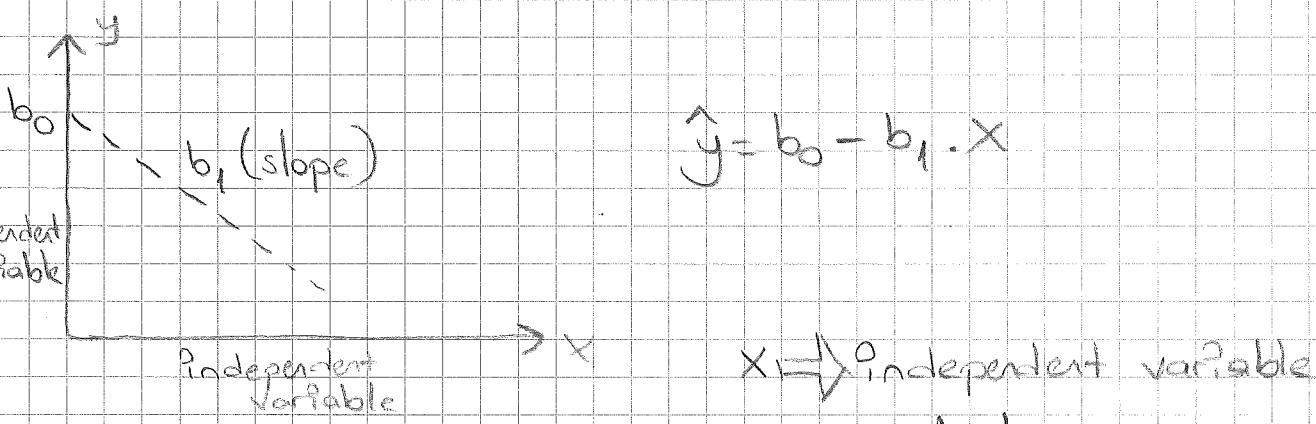
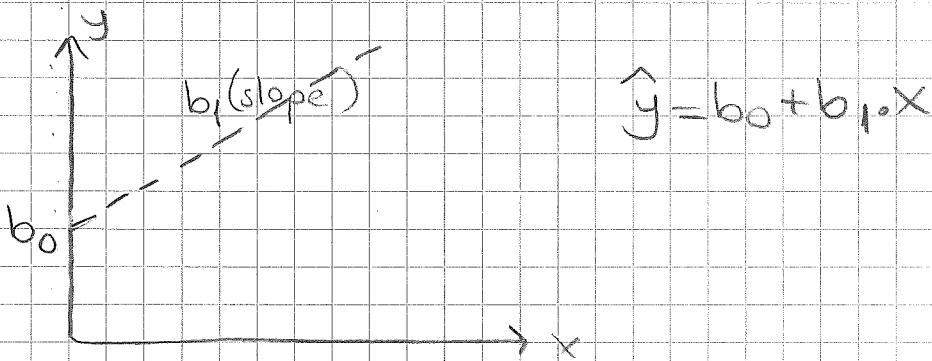
Residuals : The difference between the observed value and the estimated value of the quantity of interest (for example, sample mean)

Cost Function : Some function of the difference between estimated and true values for an instance of data.

Gradient Descent : A first-order iterative optimization algorithm for finding a local minimum of a differentiable function.

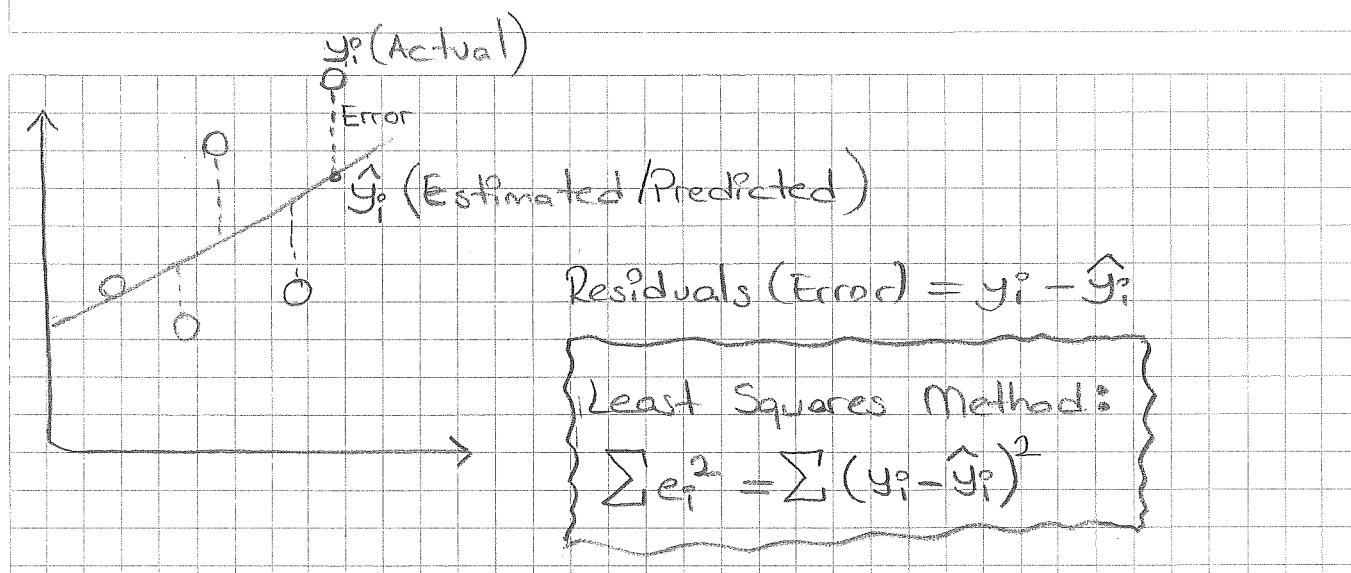
Linear Regression nedir?

Bağımlı ve bağımsız değişkenler arasındaki en iyi doğrusal line'ı bulma yöntemidir. Hangi line'i male uygun olduğunu bulmak için Least Squares Method kullanılır. (Continuous veriler arasında olur.)



$x \Rightarrow$ independent variable
control
manipulate
change

$y \Rightarrow$ dependent variable
outcome



MEAN ABSOLUTE ERROR (MAE)

Difference between the model predictions and the true (actual) values

$$MAE = \frac{1}{n} \sum_{i=0}^n |y_i - \hat{y}_i|$$

- ① Calculate the residual of every data point.
- ② Calculate the absolute value (to get rid of the sign)
- ③ Calculate the average of all residuals.

Eğer $MAE = 0 \Rightarrow$ Model tahmini mükemmel.

MEAN SQUARED ERROR , (mse)

The average squared difference between the estimated values and the actual value.

$$MSE = \frac{1}{n} \sum_{i=0}^n (y_i - \hat{y}_i)^2$$

- ① Calculate the residual for every data point.
- ② " the squared value of the residuals.
- ③ " the average of results from step 2

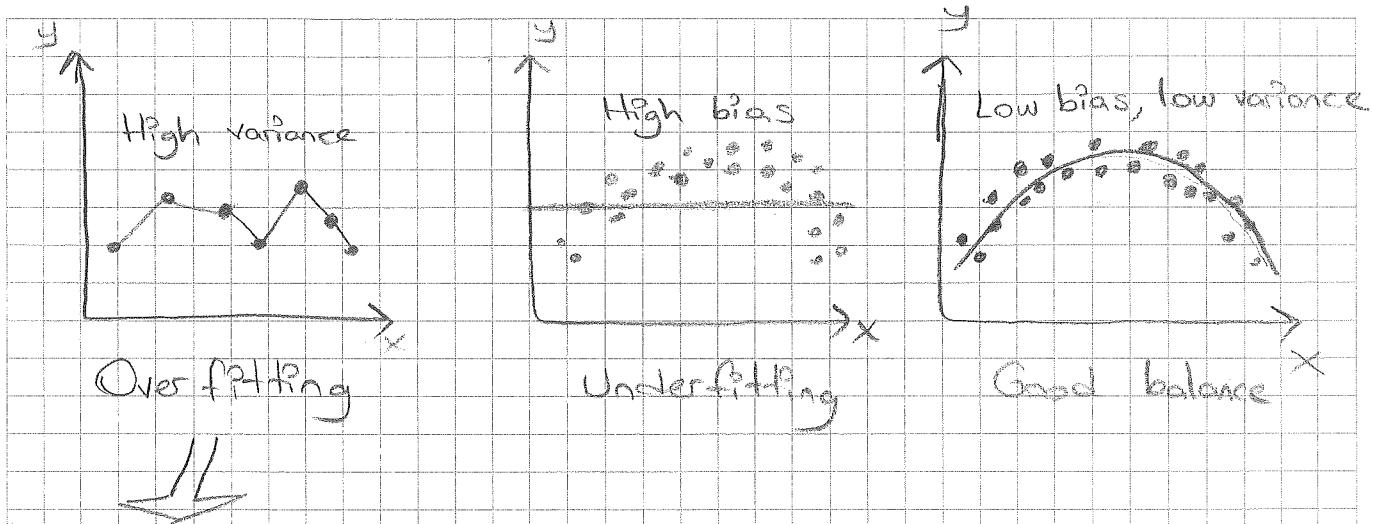
Root mean square error (rmse)

Rmse , represents the std of the residuals.

(Difference between the model predictions and the true values

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=0}^n (y_i - \hat{y}_i)^2}$$

- ① Calculate the residual for every data point.
 - ② " the squared value of the residual.
 - ③ " the average of the squared residuals.
 - ④ Obtain the square root of the result
- mynote



Model, veriyi öğrenmek yerine ezberler.

Bu nedenle sadexe bu dataya

özel bir model olur.

Genel bir model olmaz.

(Bir öğrencinin konuyu anlamak yerine

sinava sıkıca soruları ezberlemesi gibi)

Training Errors vs. Validation Error (Test Error)

Hata oranlarını azaltmak için modelimizin parametrelerinde ve özelliklerinde değişiklik yapabiliyoruz.

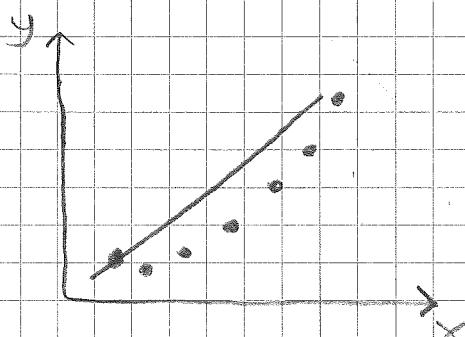
Bu şekilde hata oranı azalır ama model karmaşık bir hale gider. Bu da modelimizi **high variance (over fitting)** riskine atar.

! bias \Rightarrow Average distance between predictions and the truth.

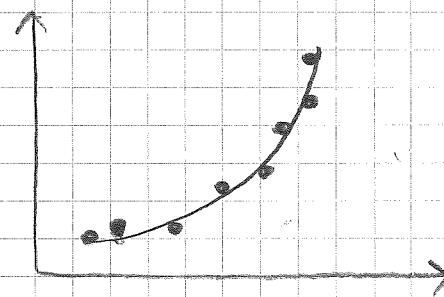
Overly Simple Model \Rightarrow Bias \rightarrow High } Under fit
 Variance \rightarrow Low

Overly Complex Model \Rightarrow Bias \rightarrow Low } Over fit
 Variance \rightarrow High

Polynomial Regression

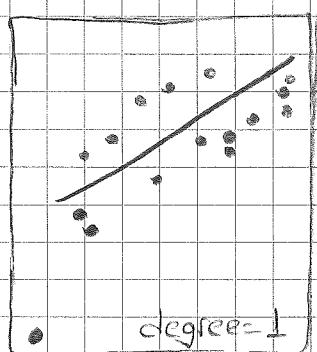


$$y = b_0 + b_1 x$$



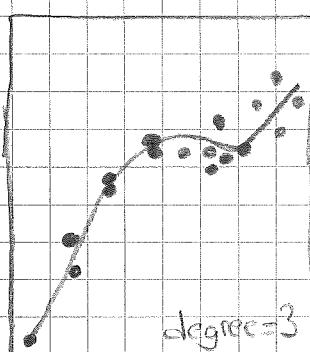
$$y = b_0 + b_1 x_1 + b_2 x_2^2 + \dots$$

Bağımlı ve bağımsız değişkenler arasındaki bağlantıının en iyi tahliliini verir. Degree=1 olursa linear regresyon oluruz.



Under fit
High Bias

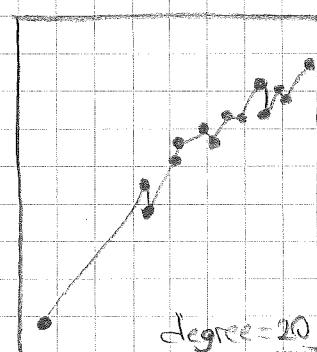
Low Variance



Correct fit

Low Bias

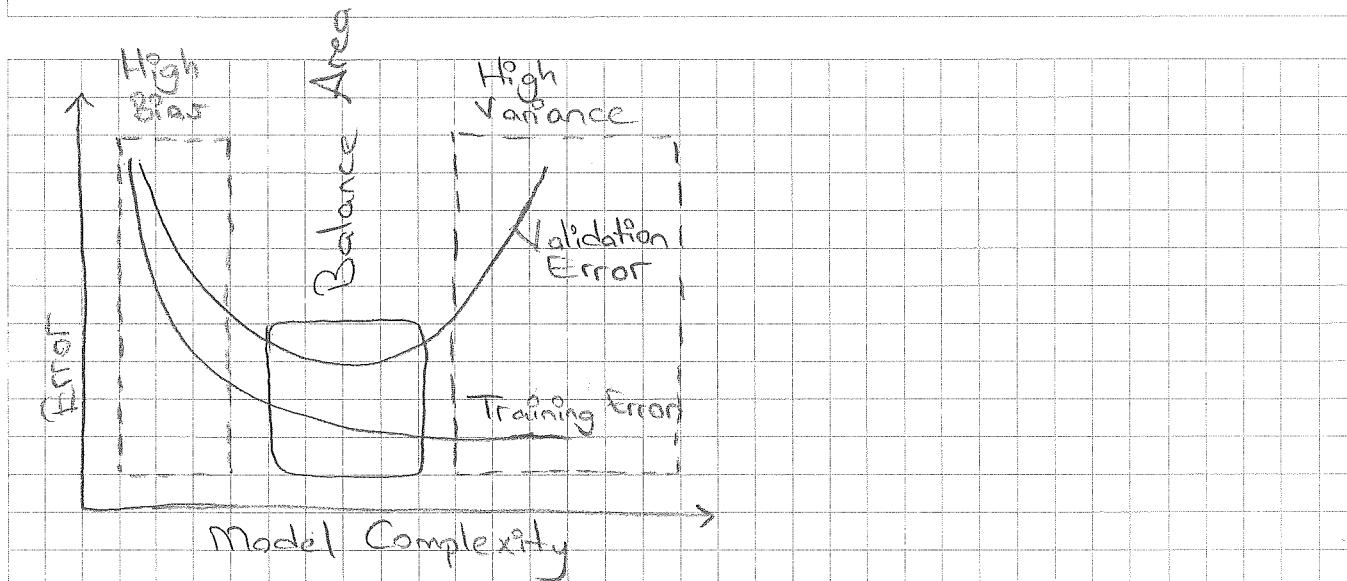
Low Variance



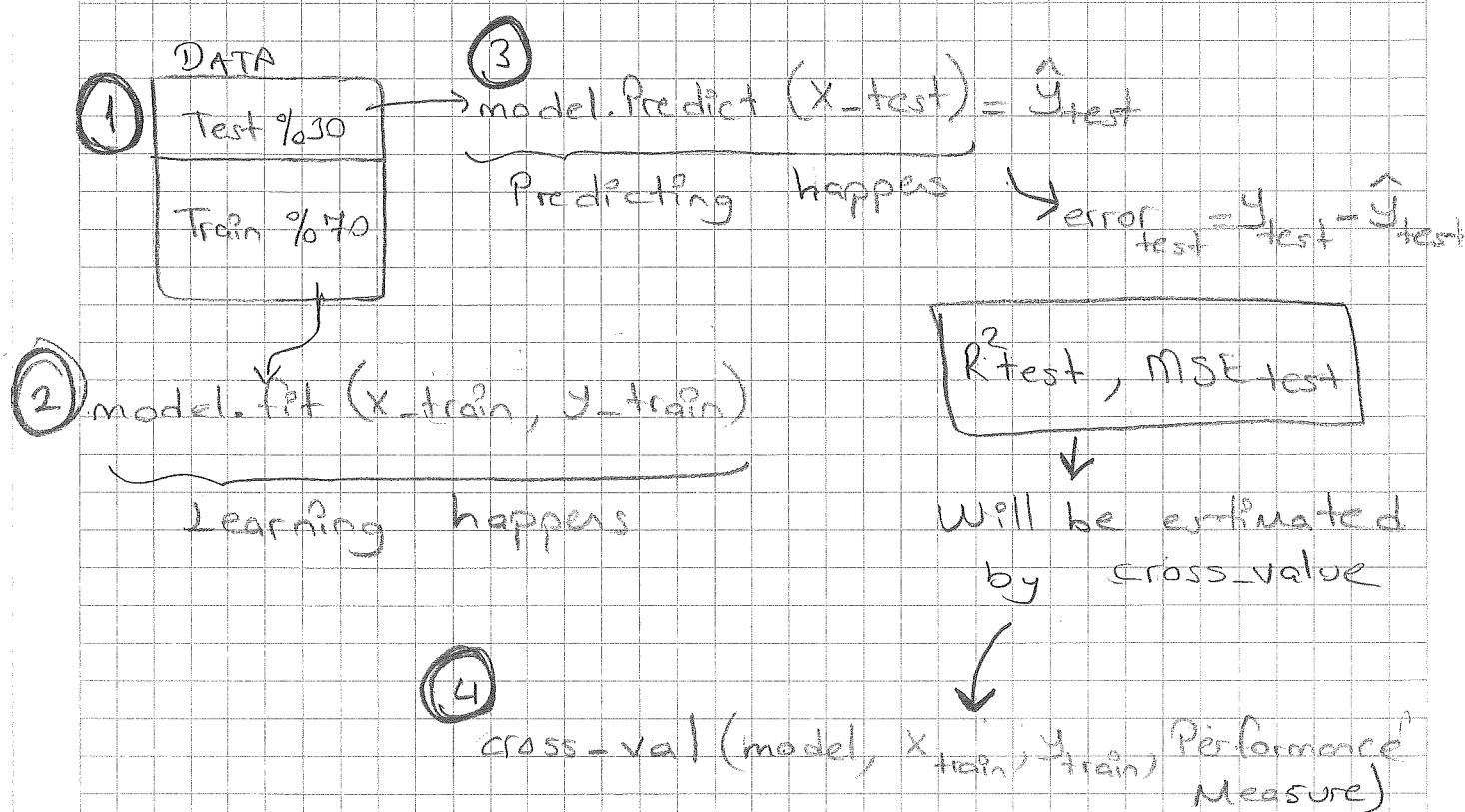
Over fit

Low Bias

High Variance



Pre-Class Video Plan



`model = LinearRegression()`

Polynomial Regression Steps:

- ① Convert X features with PolynomialFeatures() method.
- ② Explore the poly features dataset.
- ③ Perform Train, Test, Split on polynomial features.
- ④ Create polynomial regression model.
- ⑤ Evaluate the performance metrics and analyze coefficients.
- ⑥ Differentiate the degree of the polynomial model and examine train test errors.

Check Yourself - 2

- * Our goal with linear regression is to minimize the vertical distance between all the data points and our line.
- * Function to create predictions \Rightarrow .predict()
- * from sklearn.model_selection import train_test_split

```
x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.25, random_state=10)
```

↑
test_size in anti-clockwise direction

Datanin $\frac{1}{4}$ or 25% in test $\frac{3}{4}$ in train

(*) Least Squares Method \Rightarrow for minimizing the sum of squares.

(*) In machine learning algorithms;

Independent variables as $\Rightarrow x = df[["\text{Feature 1}", "Feature 2", ...]]$

Dependent variable as $\Rightarrow y = df[["\text{Target}"]]$

IN-CLASS

"y"

(Independent variables)

↑
features

Target "X"

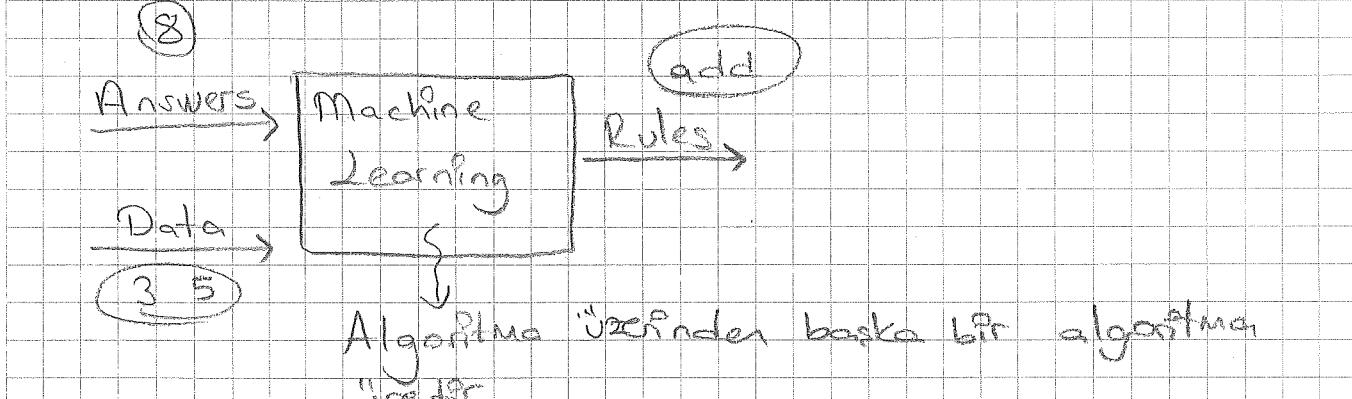
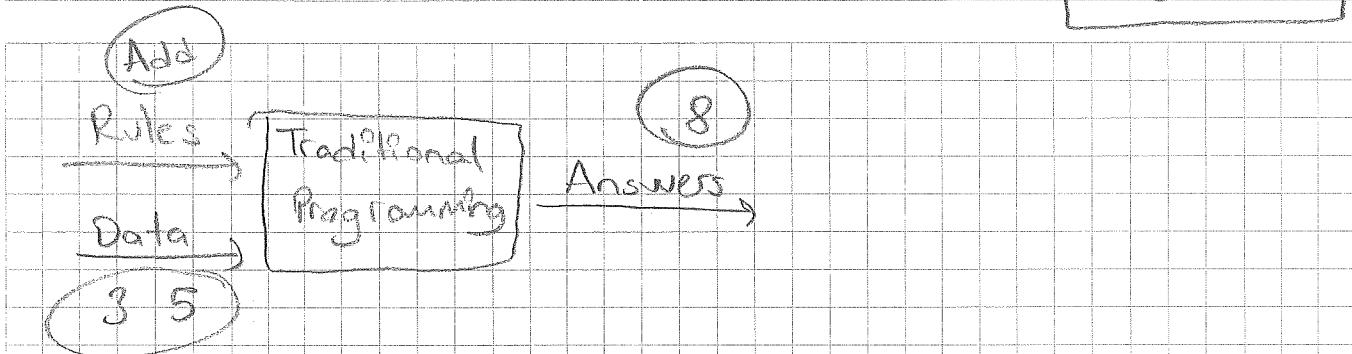
(Dependent variables)

Label

Train
data

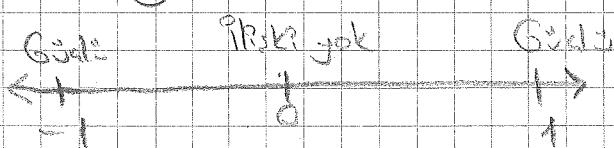
Test
data

Randomly
split.



Correlation :

- İki özelliklerin arasındaki ilişkilerin gücünü verir.
- $+1$ ile gösterilir. -1 ile $+1$ arasında gösterilir.



Correlation yoksa linear regresyon da yok.

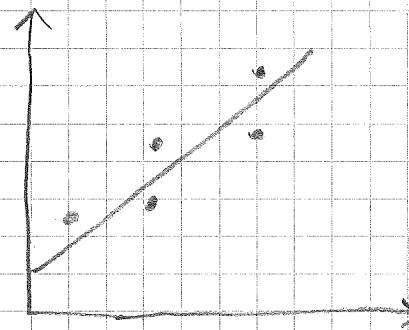
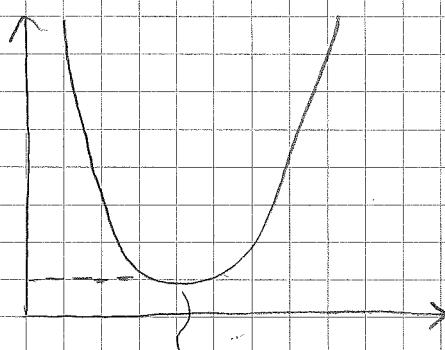
Regression :

- Bir feature' da bir değişiklik olurken diğer feature' in nasıl etkilendigine bakar.

Correlation, (r)

Regression

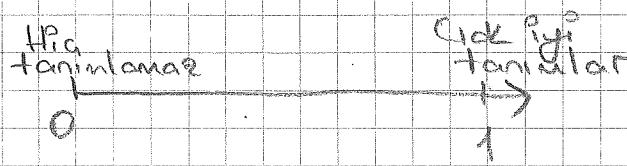
- (*) İki veri arasındaki ilişkiye bakar.
 - (*) Variable'lar birlikte hareket eder. (Scatterplot ortaya çıkar.)
 - (*) Data -l ile +l arasında bir rakamla ifade edilir.
 - (*) Is there a relationship between X and Y ?
- (*) Bir feature'in diğer feature'ları nasıl etkilediğine bakar.
 - (*) Cause and effect & senine gelse, (Etki - tepki)
 - (*) Data bir line ile ifade edilir.
 - (*) What is the relationship between X and Y ?
- Y Correlation yoksa regression de yok.
- Gradient Descent, :
- Gradient descent is an algorithm that finds best fit line for given training dataset.



Errorlerin θ 'a yaklaştırılmaya çalışılır. Bir kırık oldugu yerde durur ve line'i çizer

The Coefficient of Determination (R^2)

Elinizdeki data ile tahmin ettiğiniz kadarını karşıyalı?

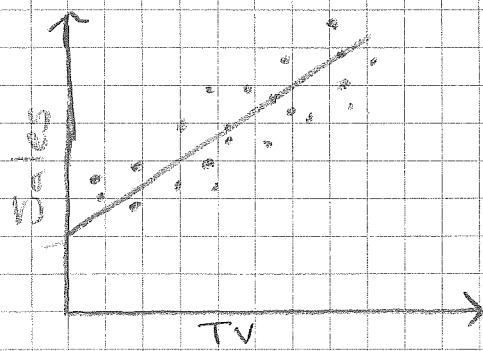


"Modeliminizi \hat{y}_i mi kotsu ays?" Bunun için kullanılır.

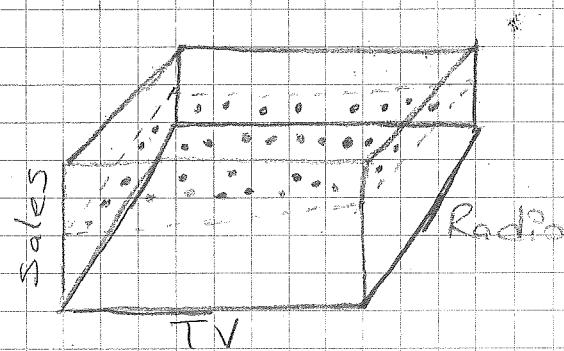
$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y}_i)^2}$$

$\hat{y}_i \rightarrow y_i$ 'nin tahmini
 $\bar{y}_i \rightarrow y_i$ 'nin ortalaması

Simple Linear Regression Multiple Linear Regression



$$y = b_0 + b_1 \cdot X$$



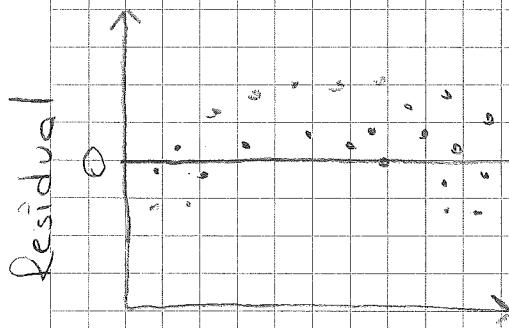
$$y = b_0 + b_1 \cdot X_1 + b_2 \cdot X_2 + \dots + b_n \cdot X_n$$

Dependent
Independent
 X 'in ağırlığı Paris

Residuals (Error): Gerçek değer ile tahmin değeri arasındaki farktır.

$$e = y - \hat{y}$$

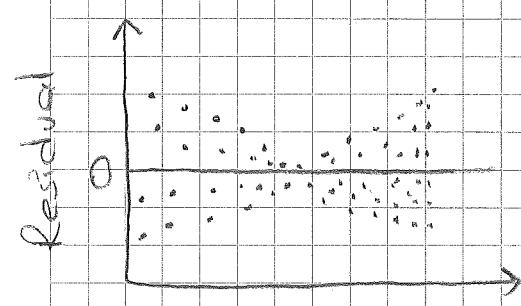
Best fit line'de errorların toplamı ve ortalaması her zaman 0 olur.



Heteroscedasticity

Non-linear data

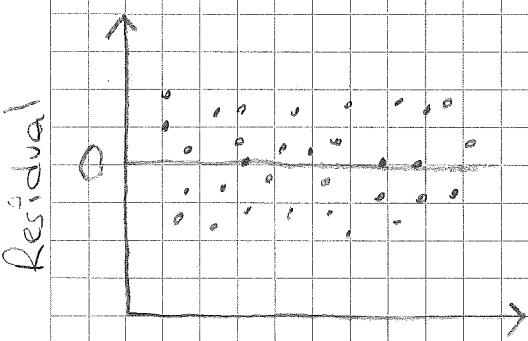
Linear regresyona uygun değil



Heteroscedasticity

Normal dağılıyor ama linear

regresyona uyumaz. Ama uydurulabilir.
(Log alınarak yapılmış.)



Homoscedasticity

Linear data

Linear regresyona uygun



Linear regresyona uyumuz isn'ti;

① Pattern göstermeyecek

② Normal dağılımımız

(2. örnekte sadece bir sort sağlanıyor.)

Regression Error Methods

① Mean Absolute Error (MAE) :

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

Desavantajı : Outlier hatalarını minimize etmez (Cezalandırır.)

② Mean Squared Error (MSE) : \rightarrow Hatanın varyansı

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

Hataların karesi alınıldığı için cezalandırması iyi

Desavantajı : Açıklaması zor

Örneğin target sütunu km'de ise bu, km².

③ Root Mean Squared Error (RMSE) : \rightarrow Hatanın standart sapması

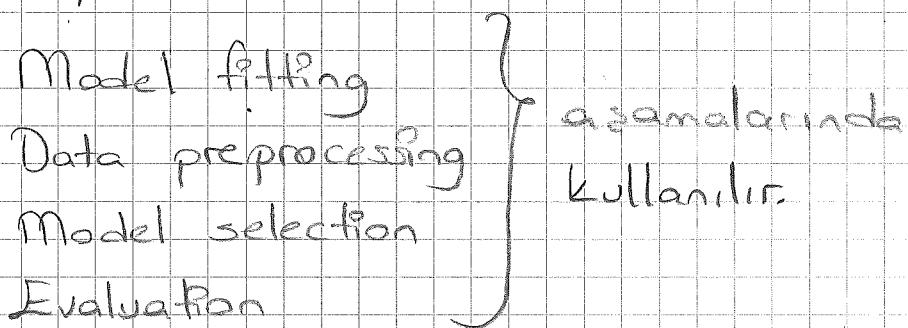
$$\sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Avantajı : Hem hataları cezalandırır.

Hem açıklaması kolaydır.

Scikit - Learn Kütüphanesi

Supervised ve Unsupervised Learning'i destekler.



ML Asamaları

1 Import asamaları

2 Splitting Data (Datayı böölme)

$x_train, x_test, y_train, y_test =$
 $\text{train_test_split}(X, y, \text{test_size} = 30,$
 $\text{random_state} = 42)$

3 Fitting Data (model kurma)

model.fit(x_train, y_train)

logmodel = LogisticRegression() \rightarrow "Once model olusturulur.
logmodel.fit(x_train, y_train) \rightarrow Sonra bu modelin içine
 x_train, y_train atılır.

(4)

Predicting Data (fit'te öğrendiklerine göre tahmin yapar.)

predictions = model.predict(X_test)

(5)

Probability of Predicted Data

model.predict_proba(X_test)

(Bu adımı, classification problemlerinde olasılık istenirken kullanacaktır.)

(6)

Evaluation (Değerlendirme)

MAE, MSE, RMSE kullanılarak evaluate istenir.

MULTIPLE LINEAR REGRESSION

① Datayı okuma ve inceleme:

`df = pd.read_csv("Advertising.csv")`

`df.shape`

`df.info()` → Missing value yok.

`df.describe()` → std ile mean birbirine yakınsa,
std > mean ise outlier sorunu var demektir.

Target sütun = "sales"

② Distribution of features:

`sns.pairplot(df);`

! sales - TV sütunları arasında kuvvetli bir ilişkisi var.

`sns.heatmap(df.corr(), annot=True);`



Target ile feature'lar arasında yüksek corr olabilir.
Varsa linear regresyon dan bahsedebiliriz.

Bir den fazla feature olduğu için linear regresyon uyumluğunu için residual plot'a bakmalıyız.

(3) Train - Test Split

pip install scikit-learn

$X = df.drop(columns='sales')$ ➔ Sadice feature'ları aldı.
(Bağımsız değişkenler)

$y = df.sales$ ➔ Sadice target'i, aldı.
(Bağımlı değişken)

from sklearn.model_selection import train-test-split

$X\text{-train}$, $X\text{-test}$, $y\text{-train}$, $y\text{-test} =$

train-test-split (X , y , test-size=0.3,
random-state=42)



Bu kod önce datayı karıştırdı.

Sonra %70'ini train için ayırdı.

%30'unu test için ayırdı.

(Data kütüphanesi test dataları %20, %10'a da dağıtabilir.)

Features	Target
$X\text{-train}$	$y\text{-train}$
$X\text{-test}$	$y\text{-test}$

! random-state=42'nin anlamı?

Ekip çalışmasında herkes aynı random değerleri alınsın gibi
(Train-test grupları aynı olsun.)

④ Model fitting :

from sklearn.linear_model import LinearRegression

model = LinearRegression() \rightarrow Model kurduk.

model.fit(X-train, y-train) \rightarrow X-train ve y-train ile eğitimin tamamla

⑤ Predicting Data :

y-pred = model.predict(X-test)



y-pred değişkenini oluşturup 'ine X-test'i attık.

X-train ile aldığı eğitimlere göre bir tahmin yap.

Sonra bu sonuçları y-test ile karşılaştırıp tahminleri doğru mu değil mi bakacağız.

Arka planda yaptığı işlem su:

model.coef_ \rightarrow Bu tür feature'lar için belirlediği array([..., ..., ...]) katsayıları verir.

model.intercept_ \rightarrow $x=0$ iken y 'yi kestigi nokta (b_0)

model.coef_ de bulduğu katsayılarla o süzünün değerini çarpacak, "uzerine de b_0 " ekleyerek tahminin değerlerini bulacak.

$$\text{mynote} \quad (y = b_0 + b_1 \cdot x)$$

⑥ Comparing :

```
my_dict = {"Actual": y_test,  
           "Pred": y_pred,  
           "Residual": y_test - y_pred}
```

Actual → Test için ayrılan target sütunu (Gerçek değerler)

Pred → Eğitim sonrası tahmin edilen değerler

Residual → Gerçek değer ile tahmin arasındaki fark. (Error)
("Bilgilerin bu kadariна sahip değilim.")

Comparing = pd.DataFrame(my_dict)

↓
Dataframe'e çevirdik.

result_sample = comparing.head(25)

↓
Görselleştirme için ilk 25'i aldık

result_sample.plot(kind="bar", figsize=(15, 3))

↓
Gerçek, tahmin değerleri ve residual'ları görsel birinde karşıladık.

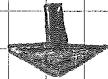
H) Error Matrix :

from sklearn.metrics import mean_absolute_error,
mean_squared_error,
r2_score

• RMSE = \sqrt{MSE} oldugu için bir fonksiyon yok.
Manuel olarak hesaplanır.

• R2-score = r2_score(y-test, y-pred)

→ 0.86



"Doğru tahmin yapabilmek için, gerekli bilginin %86'sına sahibiz." demek.

(Bağımlı değişkendeki varyansı açıklayabilece oranı.)

• mae = mean_absolute_error(y-test, y-pred)

→ 1,51

• mse = mean_squared_error(y-test, y-pred)

→ 3,49

• rmse = np.sqrt(mse)

→ 1,94

! MSE'de kare alındığı için binimlerde sonun yasası. Mesela doların karesi oluruz.

Bu yüzden karekökü'nü aldırmamız yöntemi olan RMSE'i yi abha çok kullanacağız ki binimler aynı olsun.

! MAE ile RMSE arasında ciddi bir uçurum varsa modeldeki kötü tahmin sayısı fazla demektir ve modele tekrar bakılmalıdır.

MAE ile RMSE birbirine yakın ise de cızalandırmaya az, bu yüzden tahminler iyiyapılmış demektir.

(8) Error Yorumu :

`sales_mean = df.sales.mean()`

, 14.02 \Rightarrow Gerçek df'in ortalamasını bulduk.

`mae / sales_mean`

$\rightarrow 0.10$ \Rightarrow Bulduğumuz error'u gerçek ortalaşa böldük.
Modelimiz ort. %10 yanlış tahmin yapıyor.

`rmse / sales_mean`

$\rightarrow 0.14$ \Rightarrow RMSE'ye göre; modelimiz ort. %14 yanlış tahmin yapıyor.

Bunların ardından müsteriye:

"Bu"ün modellemi denedim. En iyi sonuc linear regresyon da aldı. Tahmindeki hata oranım da "% 14" dir ve müsteri kabul ediyorsa model testime edilir.

⑨ Adjusted R² score

Feature sayısı arttıkça R² score da artar. R² score'ın scikit learn kütüphanesinde bir fonksiyonu yok. Manuel olarak yazılacak.

def adj_r2(y-test, y-pred, df):

$$r^2 = r^2\text{-score}(y\text{-test}, y\text{-pred})$$

n=df.shape[0] (Datadaki gözlen sayısi)

p=df.shape[1]-1 (Feature sayısi)

$$\text{adj}^2 = 1 - \frac{(1-r^2) * (n-1)}{(n-p-1)}$$

return adj_r2

adj_r2(y-test, y-pred, df)

Adjusted R2 score'ın anlamı:

Eğer dataya yeni feature'lar ekleyerek, datanın yeni bilgiler öğrenebilmesi için yeni gözlemler ilave etmemiz gereklidir ki data yeni seyler öğrenebilsin.

Feature sayısı arttığında gözlem sayısını artırarak R2 score'da yalnızca bir iyileşme olur ama MAE ve RMSE'de herhangi bir değişiklik olmasa veya daha çok kötüleşebilir. Bunun önüne geçmek için "Adjusted R2 score" fonksiyonunu kullanıyoruz.

Bu score, gözlem sayısı ile feature sayısını dengeler.

Kaynaklarda; "Eklenen her feature'in 10-15 gözlem gereklidir." olarak kabul edilir.

(10) Evaluation Model :

```
def eval_metric(actual, pred):
```

```
    mae = mean_absolute_error(actual, pred)
```

```
    mse = mean_squared_error(actual, pred)
```

```
    rmse = np.sqrt(mse(actual, pred))
```

R2, MAE,

MSE, RMSE

için

toplu

fonsiyon

yazdık.

$$R2_score = r2_score(actual, pred)$$

```
print(R2_score)
```

```
print(mae)
```

```
print(mse)
```

```
print(rmse)
```

mynote

`eval_metric(y-test, y-pred)`

→ R2-score : 0.86

MAE : 1.51

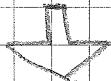
MSE : 3.79

RMSE : 1.94

"Aldığım bu skorlar gerçekten modelin yapabildiği en iyi tahminler mi?"

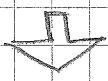
Bunu anlamak için modeli eğittiğimiz dataya kendisine test edereliğiz.

$y\text{-train-pred} = \text{model.predict}(X\text{-train})$



Bu sefer tahmin için $y\text{-test}'i$ değil, $X\text{-train}'i$ kullandık.

gerçek
değer
↑
 $\text{eval_metric}(y\text{-train}, y\text{-train-pred})$
tahmin
↑



Oluşturduğumuz def fonksiyonun place
 $y\text{-test}$ yerine $train'$ leri atıyoruz.

Anaç: Train setinde datanın genelleme yapabilmesini istiyoruz, ebeveyn yapmasını istemiyoruz.

Train skorları ile test skorları birbirine yakın olursa; "Model güzel bir genelleme yapmış."

Skorlar arasında büyük farklar olursa;

"Model öğrenmemiş, eberlemiş. Eğitim yaptığı datada skorlar çok iyi, hiç görmediği datada skorlar düşük."

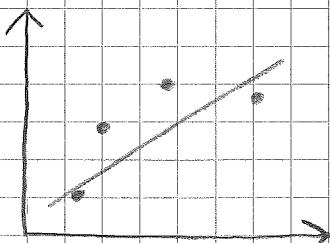
Yani, overfitting kontrolü yapıyor.

"Eberleme, öğren!"

Biz bu data örneğinde sadece Linear Regression'a baktık. Bundan sonraki datalarda tüm modelleri deneyip en iyi skor aldığımiza yola devam edeceğiz.

BIAS

Modelin dataları çok fazla genellemesi dir.



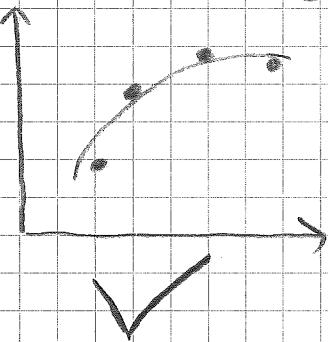
UNDERFIT

Az parametre

High bias

Low variance

Data → Simple

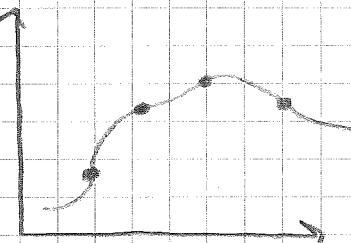


Ideal

Low bias

Low variance

Data → Complex



OVERFIT

Çok parametre

Low bias

High variance

Data → Complex

→ Modelin daha eğitime ihtiyacı var. Skorları artır. Egit beni :)

UNDERFIT : Datayı çok fazla geneller.

Hem eğitlim hem de test datasında büyük hatalar yapar.

Bu yüzden variance düşük çıkar.

Model hiç bir şey öğrenmemis olur.

OVERFIT : Çok parametre olunca bütün noktalarına gitir.

Datayı esberler.

Train datasında variance \rightarrow Düşük

Test datasında variance \rightarrow Yüksek

Günlük

esberledi.

HOW TO RECOGNIZE UNDERFITTING - OVERFITTING?

Underfitting,

High training error

High test error

Overfitting,

Low training error

High test error

En çok bununla karşılaşıyoruz.

Hem train hem test datasında düşük hatalar almayı bekliyoruz.

! Variance : Train ile test datası arasındaki fark

Complexity çok düşük olursa model underfit (çok karşılaşıyor), çok yüksek olursa overfit olur. Bir ne çok yüksek ne de çok düşük olsun istiyorum.

! Bias : Train setindeki gerçek değerle tahmin edilen değer arasındaki fark \rightarrow Low \rightarrow Underfitting ✓

High \rightarrow Overfitting ✗

Under fitting ve Over fitting ile mücadele için;

Under fitting \Rightarrow feature eksikir. (Complexity artar.)

Over fitting \Rightarrow Feature azaltılır. (Complexity düşer.)

Datanın outside'dan kaynaklanan yorum
data artırlar. Ya da degree düşerler.

	Bias	Variance	Complexity	Flexibility	Generalizability
Under fitting (Simple)	High	Low	Low	Low	High
Over fitting (Complex)	Low	High	High	High	Low

Over fitting 'de "Daha iyi bir sonuc alır mıymos?"
amaçyla Cross Validation yapılabilir.

Ya da regularization (Lasso & Ridge) yapılabilir.

TYPES OF REGRESSION MODELS

Simple Linear Regression

Multi Linear "

Polynomial Regression

Balanci (Gesetz
datalar da
top vullen
het)

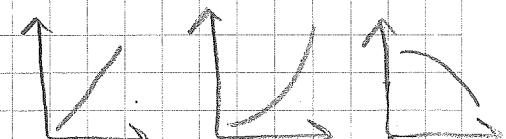
Simple Linear

1. Multi Linear

Polynomial

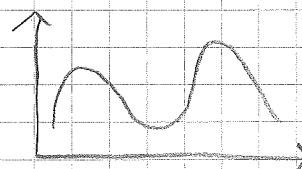
- 1 feature
 - Bir den fazla feature
 - Multi'nin özel birimi
 - 1 label
 - 1 label
 - Non-linear datalarak
öğretilicek.
 - Polynomial degree'sini
seçmek çok önemli.
 - Degree=1 \rightarrow Linear
 - Degree=Düşük \rightarrow Underfit
 - Degree=Yüksek \rightarrow Overfit

Linear Regression $\Rightarrow y = b_0 + b_1 x$
 $y = b_0 + b_1 x^2$

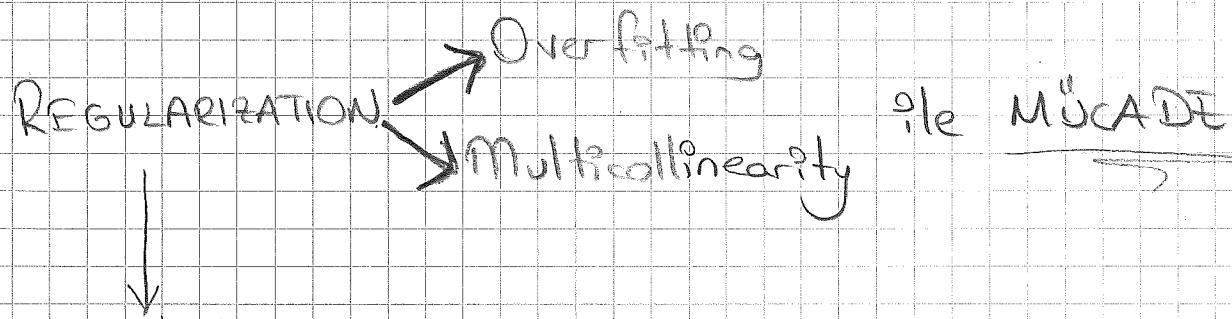


Nonlinear Regression $\Rightarrow y = b_0 + (b_2 - b_1) \cdot e^{-\frac{b_3}{X}}^{b_4}$

$$y = b_1 \cdot \cos(x - b_2) + b_3 \cdot \cos(2x - b_4)$$



▼ Polynomial regression'da Bias Variance Trade-off (Bias-Variance dengesi)'ni ayarlamak çok önemli.



Performansı yükseltir.

Train'de de test'te de makul sonuçlar verir.

Multicollinearity, linear modellerde yaşanan bir sorundur. Advance modellerde bunların arkasında çatışan parametreler var ama linear modellerde yok.

Cor. ols
başlıyor.

15.01.2022

Multicollinearity: Bir modeldeki değişkenlerin en az ikisiin birbiryle çok iyi ilişti olması durum.

Bu durum, öne sürülen sırasının karışmasına neden olur.

Modelin kalası karışır. Herhangi birisi atılabilir. Atilmasa da regularization yapılır. Alma sistemi yapılarak feature'lerden herhangi biri atılabılır, farketmez.

REGULARIZATION:

$$\sum e_i^2 = \sum (y - \hat{y})^2 + \text{Penalty} \quad (\text{Least Squared Method's cost function})$$

Cost function'a penalty (cera) ekleyerek regularization yapılır.

Regularization için 3 yöntem var:

Ridge Regression (L2) ← Penalty → Lasso Regression (L1)

Elastic-Net

λ. coef'in (slope) karesi
kadar hata eklenir.

λ. coef'in mutlak
değeri kadar hata
eklenir.

→ Çezaın seviyesini belirler.

! λ : Lambda (Hyper-parameter) :

Modelin içinde sabit olmayan, elle değiştirebileğimiz, modelin kalitesini, repliklerini iyileştirebileğimiz parametredir.

Lambda arttıkça iyileşme gönülür. Ama bir noktadan sonra tekrar kötüleşme başlar.

① Ridge Regression :

Datamız train datasında çok iyi sonuc verip test datasında çok kötü sonuc vermiş olsun. (Overfit \Rightarrow Ezber)

Biz train datasına ridge regression ile penalty eklersek ($\lambda \sum_{j=1}^p B_j^2$) p'ne'imiz train ve test datasında makul bir noktaya gelir. (Train datasına hata ekleyince, test datasındaki hata düşer.)

(\downarrow
Bias ekledik, variance düşü)

! Lambda Python'da fonksiyon gibi olduğu için λ diyeceğiz.

Lambda

Ridge Regression Avantajları

- ① Eğer feature fasta, otlu olsaydı;
Regularization ile bazı feature'ların etkisini azaltarak complexity'ı aşağı seker. (Yani feature'ları azaltır.) Az feature -az row gerektir. Böylece model daha düzgün çalışır.
- Çünkü modelin düzgün çalışması için; data sağa doğru bütükse (feature) sağa doğru da (observation) büyük olmalı. Bu durumu değiştirmeyen yakalı Ridge Regression ile bazı feature'ları öne çıkarırız böylece observation - feature dengelemiş olur.
- ② Multicollinearity için çok iyi.
Feature'lar arasında önemvisi yapar, en iyi feature'ları seçer.
- ③ Modelde bias (hata) eklenir, bunun karşılığında variance düşer. Böylece regression metric'lerinde iyileşme olur.

Ridge Regression Dezavantajları

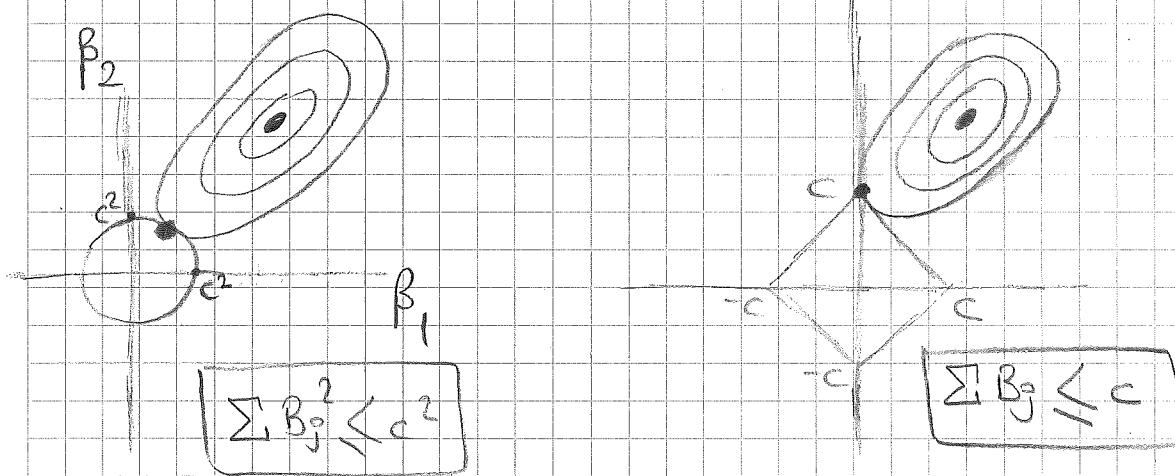
Feature selection yapamaz. İyisi kötüyü gösterir ama kullanma düşüncesinde feature'ları atamaz. Mesela 100 feature'ın ilk 20'si sıfıra yakın değerler çıktı (çok şıkkı), geri kalan 80'yi atarsak stability bozulur. Çünkü, genetik olarak feature'ların birbirleriyle ilişkili olduğu düşünülmeli.

Desmos Graphing Calc. Programı yükle

Zoplarda mutlaka bir etki vardır.

(2) LASSO Regression :

Calisma matrisi Ridge ile aynı. Cost function'a hata eklenir. Ama karesi değil, mutlak deger eklenir. ($\lambda \sum_{j=1}^p |\beta_j|$) \rightarrow Egimin mutlak degeri



-Ridge Regression -

Feature'ları etki sirasına göre sıralar, ama atmaz.

(X ve Y eksenindeki degerler (X eksenindeki deger 0)

0'a yaklasa bile hic bir zaman 0'a dusmez.)

\downarrow
0'a yaklastirir.

-LASSO Regression -

Önemli feature'ları alır, gari balanı direkt atar.

\downarrow
0 yapar.

LASSO Regression Avantajları:

- ① feature fazla, data az ise "öneşit feature" ları direkt atar. Complexity azaltır. Ama hata artabilir.
- ② Feature selection yapabiliç.
- ③ Bias (hata) eklenir, variance düşer. Regression metric'leri iyileşir.

LASSO Regression Dezavantajları:

Yüksek corr olabilecek olan feature'ların birini alıp diğerini attığı için gruplama yapamaz. Çünkü feature'ları yok eder.

Feature azaltacağı için bu durum performansı etkiler ve R² skorları biraz düşer.

LASSO feature sayısını düşürdükten sonra bit deha feature feature düşürmek istesek:

30 feature $\rightarrow R^2 = \% 93$, RMSE = 2

\downarrow Lasso ile
20 feature $\rightarrow R^2 = \% 90$, RMSE = 3

\downarrow
10 feature $\rightarrow R^2 = \% 80$, RMSE = 4

Lassonun sonra kendini düşündürerek, performans çok düşer.

Summary:

- ④ OLS (Ordinary Least Squares) hatalı bir model ortaya koymaya çalışır.
- ④ Ridge & LASSO, öne çıkarmak istedikleri feature'lara bias ekler.
↓ ↓
Süründür Oldür.
- ④ Ridge & LASSO, over fitting ile mücadele için kullanılır.
- ④ Multicollinearity' den Kurtuluş için kullanılır.
Ridge \Rightarrow Group selection için kullanılır.
feature'ların hepsi "Önemliyse, atanıysak" bu yöntem
- LASSO \Rightarrow Eliminating predictors için kullanılır. (iyileştirilmesi)
kötülen at.)
Feature sayısı düşer

3)

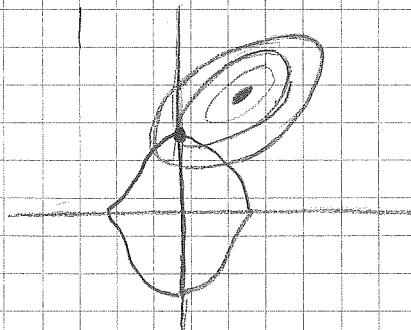
Elastik - NET

Ne Ridge olsun, ne de LASSO olsun. İkisinden de olsun.

Yani $\alpha \rightarrow 0$ ise Ridge gibi davranır.

$\alpha \rightarrow 1$ ise LASSO gibi davranır.

Ama genellikle LASSO'yu seçer.

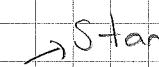
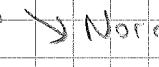


Elastic - NET'i pek kullanmayacagız.

Regularization yapabilmek için 2 sey yapılmalı:

① Feature Scaling (Bütün coefficient'ları aynı seviyeYE getirir.) 

② Cross Validation and Grid Search (λ 'yi bulmak için)

① Feature Scaling,  

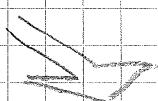
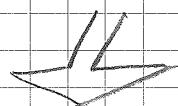
Ridge veya LASSO uygulanmadan önce coef'leri aynı aralığa çekmeniz lazımlı ki coefficient'ları aynı kefeye kayabilelim.

Sayılar çok büyükse coef küçük, düşüksese coef büyük olur. Bu büyük olan da her zaman kazańır. Bunu istemiyorum. Aynı range'de olsunlar ki kıyaslama yapabileym.

! Scale sadece train datasinge uygulanır.

Regularization'dan önce scale'leriz ki coef'lerin karşılaştırılabilmesi.

Range'ler aynı aralıkta mı değil mi? Karar verebilirsen scale uygularız.



Standardisation

Normalisation

Standardization: (Z-score)

mean = 0 } feature değerlerini bu formata getirir.
 std = 1 } standart dağılım.

$$X_{\text{changed}} = \frac{X - \mu}{\sigma}$$

Normalization:

Feature değerlerini 0-1 aralığına getirir.

En küçük değere 0, en büyük değere 1 venir, aralığı buna göre belirler.

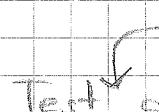
$$X_{\text{changed}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$



Hangi iyi sonuc varisse onu sececeğit.

► Feature scaling sadece train datasına uygulanır.

• Tüm dataya uygulanırsa "Data Leakage" ortaya çıkar.



Test datanın kopye çekmesi

Data Leakage:

Test datanın görmemesi gerekken yerleri görüp kopye etmesidir. Bu durumda overfitting ortaya çıkar.
Model güvenililiği ortadan kalkar.

② Cross Validation and Grid Search:

"Train için acaba datanın doğru yerini mi aldım?"

Her seferinde datanın farklı yerlerini train ve test olarak ayırır.

k-fold Cross Validation:

Iteration 1 accuracy = %92

Iteration 2 accuracy = %95

Iteration 3 accuracy = %89

⋮

Iteration k accuracy = %90

Default k=5

Final accuracy \Rightarrow Tüm iteration'ların ortalaması

"Datanın her yerinde modelin düzgün çalışıyor mu?"

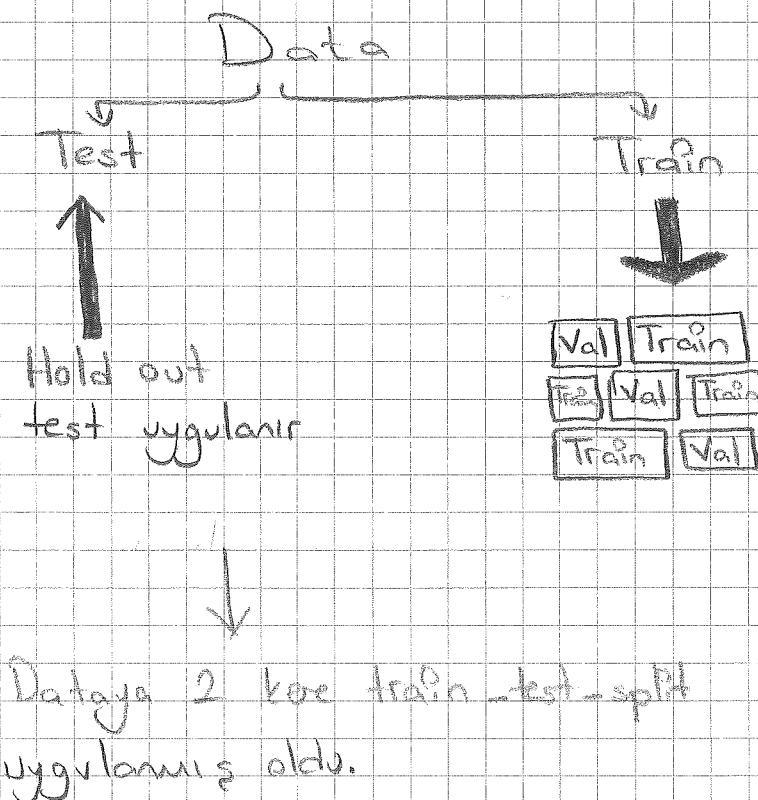
Bunu bulmak oluyoruz.

LOO Cross Validation :

Bir datasetin sadece 1 observation'ini teste ayırtır. Genel kalan hepsi train'e ayırtır. Bunu her satır için tek tek yapar. Satır sayısı kadar iteration olur.

İş yükü coktur. Bilgisayar çok iyi değilse çok uzun sürer.

Cross Validation uyguladık ve acaba Data leakage oldu mu bundan şüpheleniyorum. O zaman "Hold Out Test" yapılır.



Train data'sı tekrar bölünür.
Cross validation buraya uygulanır.
Sonra data önce hiç
görmemişti test datasına
hold out test uygulanır.
Sonuçlar iyiye yola devam.

Grid Search :

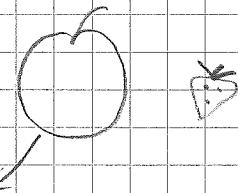
Modeli test ettiğten sonra "hyper-parameter" ların en optimumunu bulmak için yapılır.

Scikit learn ; verdığınız her hyper-parameter'i iterasyon üzerinde deneysen (grid search) hem de cross validation yapan bir fonksiyona sahiptir.
Buna "grid search cross validation" denir.

Hem iterasyon hem hyper metre'leri tek tek dener.

Best model, best parameter ortaya çıkar.

! Scaling yapmadığı zaman :



① Model buna çok daha fazla ağırlık verir. Bu da feature'ı nemli bile olsa bit plaşa çıkarır.

② Gradient Descent tabanlı (Linear Reg. Logistic Reg.) modellerde çok daha hızlı olabilir.

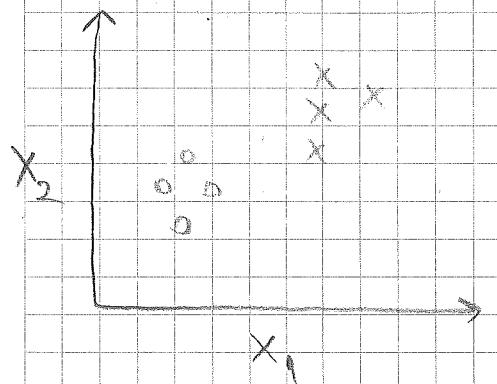
Distance tabanlı (mesafe tabanlı) modellerde büyük olan farklılıklar çok daha büyük ağırlık vermeye.

20.01.2022

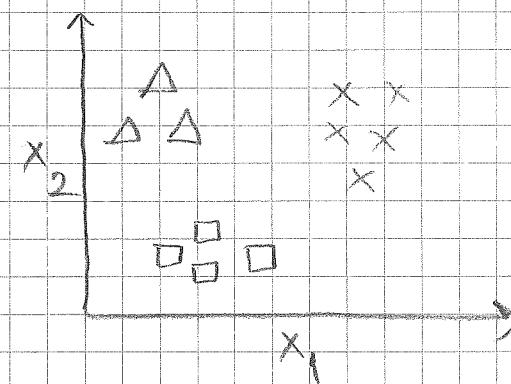
Logistic Regression Theory

Bir classification algoritmasıdır.

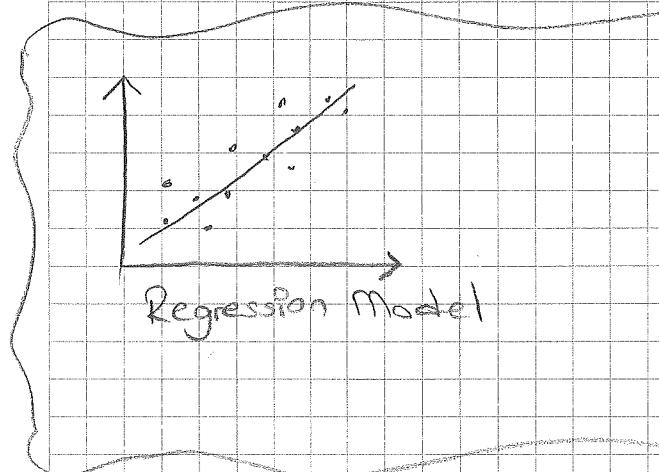
Hasta, hasta değil gibi.



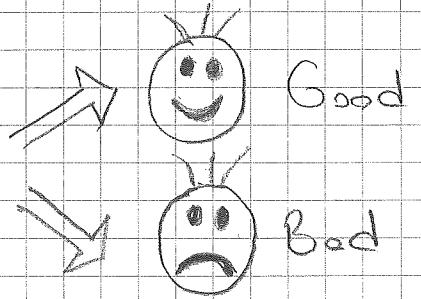
Binary Classification



Multi-class Classification

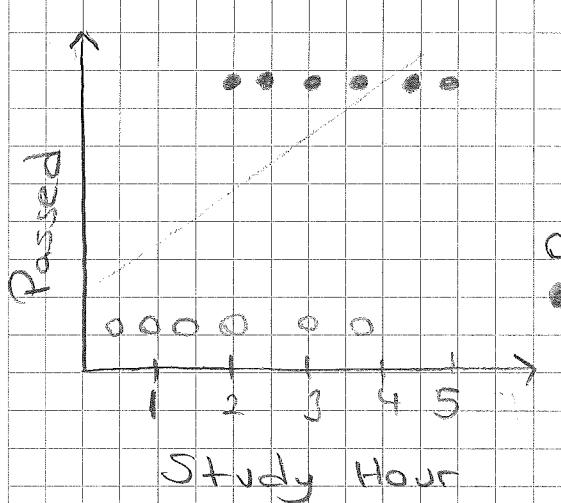


Regression Model



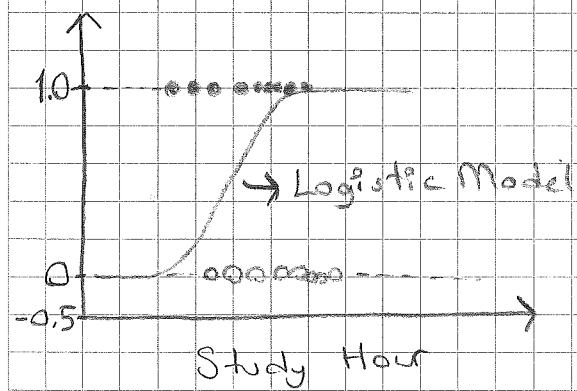
Classification Model

! $\log \rightarrow -\infty$ ile $+\infty$ arasındaki sayılar
0 ile 1 arasındaki sayıler



Buraya Linear Line uygulanır ama mantıksız olur.

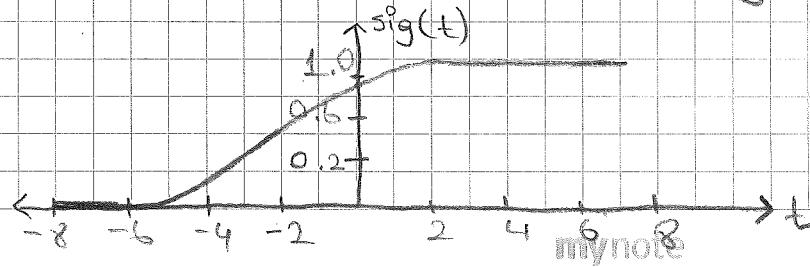
Bu yüzden bu tür verilerde Logistic Regression'i kullanılır.



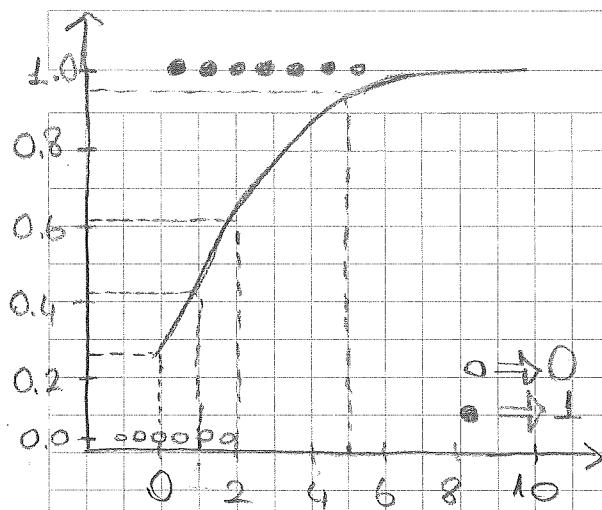
\Rightarrow Aldığınız bütün değerler 0-1 arasında olacak. Yani olasılıkları olacak.

Logistic Model \Rightarrow Bütün değerler 0-1 arasına getirir.
Her bir değer için probability verir.
Bunu yapan fonksiyon \Rightarrow Sigmoid Function

! Study hour'a ne verirsek verelim Sigmoid Function
 size 0 ile 1 arasında bir değer döndürür.



\rightarrow t ye göre her zaman
 0-1 arası değer
 döndürür.



Hıç çalışmadi \Rightarrow Geçme şansı $\approx 62\%$

1 saat çalıştı \Rightarrow " " $\Rightarrow 41\%$

2 " " " \Rightarrow " " $\Rightarrow 61\%$

5 " " " \Rightarrow " " $\Rightarrow 95\%$

Logistic Regression;

- ① Her feature için coef değerlerini bulur.
- ② Sigmoid function'ı w'ye göre değerleri 0-1 arasına sokar. Bize bir probability verir.
- ③ Bu işin bittiğimiz threshold'a göre "gesti-kaldı" der.

Threshold : > 0.5 "gesti"
 < 0.5 "kaldı" gibi

Sigmoid FUNCTION

$$P(z) = \frac{1}{1 + e^{-z}}$$

\Rightarrow def sigmoid(z):
 return 1 / (1 + np.exp(-z))

$z = np.dot(x, weight)$

$h = sigmoid(z)$

Probability'ı bulmak için coef'lere ihtiyacımız var. Bu yüzden yine 'Linear Regression', kullanmak zorundayız.

$$\text{Income} = b_0 + b_1 \cdot X$$



$$P(\text{Income} > 4000) = \frac{1}{1 + e^{-(b_0 + b_1 \cdot X)}}$$



Degeri 4000'den
yüksek olantasin
olasılığı.

Linear Reg.

$$z = b_0 + b_1 \cdot X$$



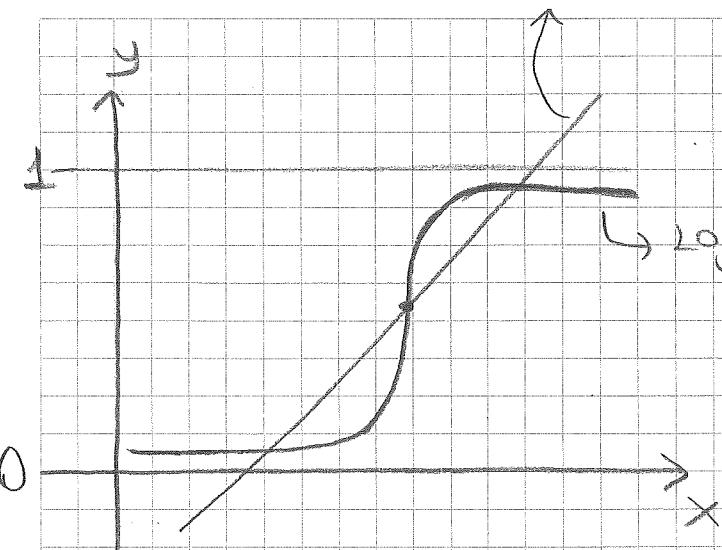
Böylece yine bir intercept ve slope üzerinden elimizde bir eğrini elde ediyoruz. Böylece elimizde her bir deger için karşılıkta bir deger (b_1) j sıfır noktası için sabit bir deger (b_0) oluyor.

Böylece Linear Regression ile galisabiliyoruz.

Linear Reg.'dan Logistic Reg.'a geçeceğiz.

Linear Model

$$y = b_0 + b_1 \cdot x$$



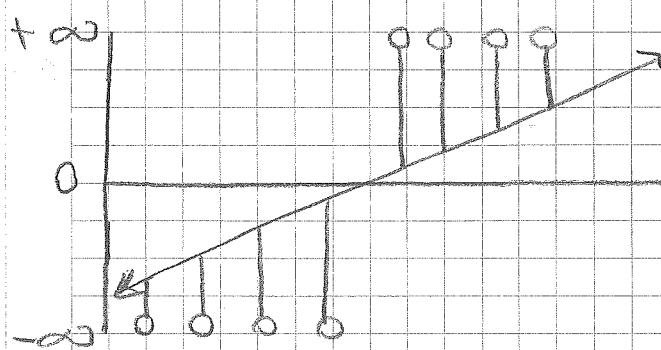
↳ Logistic Model

$$P = \frac{1}{1 + e^{-(b_0 + b_1 x)}}$$

(*) "Once değerlerini probability'e (0-1 arası) getiriyoruz.

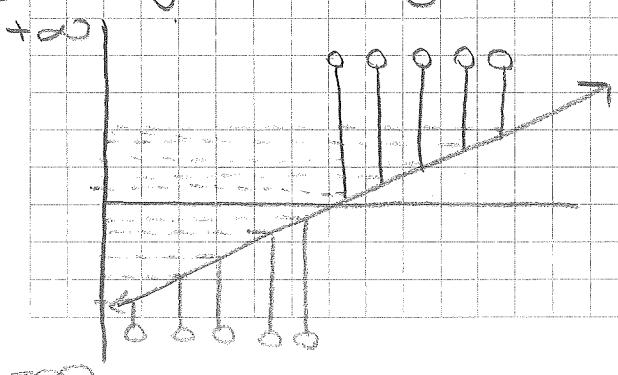
(*) 1 gözlemleneni $+ \infty$ 'a

0 gözlemlerini $- \infty$ 'a kaydırıyoruz.



(*) Burda residual'lar, bulmanız mümkün değil. Çünkü residual'lar sonsuz. Yani burda bir "best fit line" çizmeniz mümkün değil.

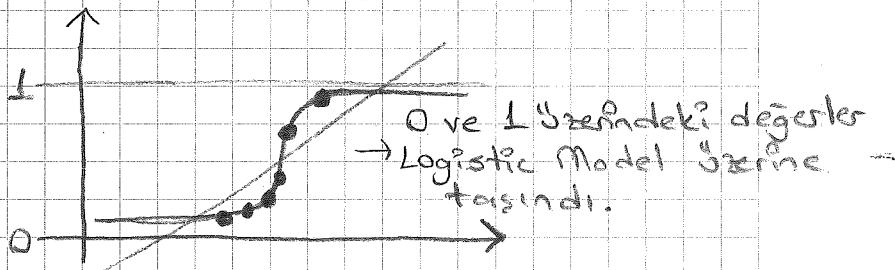
(*) Bu yüzden rastgele bir çizgi çiziyoruz.



İzdihamların y ekseniindeki ln karşılıklarını alıyoruz.

(*) In karsılıkları kullanarak probability'ler hesaplanır.

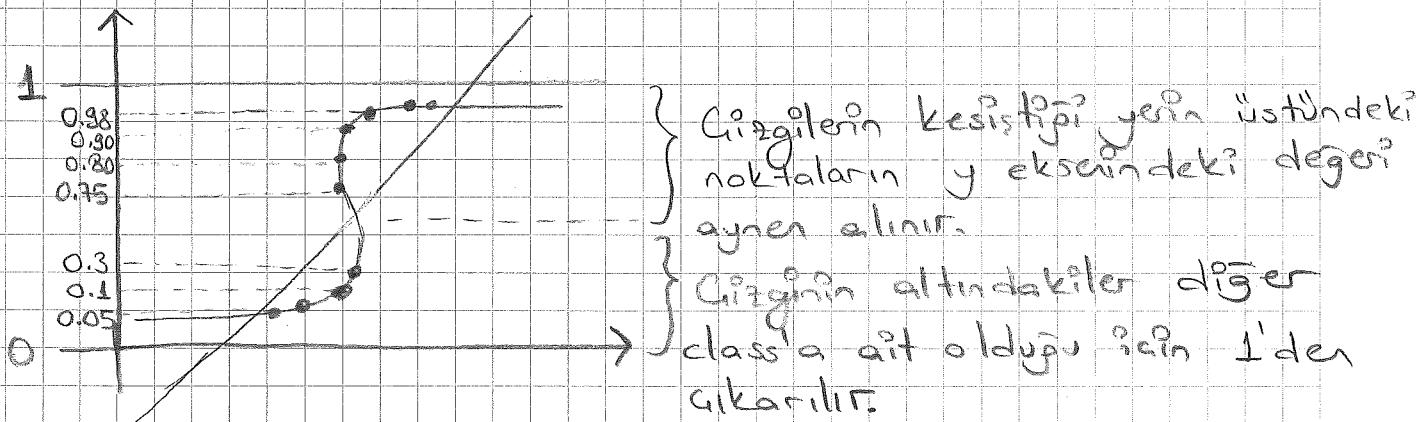
$$P = \frac{e^{\ln(\text{odds})}}{1 + e^{\ln(\text{odds})}}$$



Likelihood = 1 class'ına ait olma olasılığı.

Bunu maximize etmek istiyoruz ki başarımla artırsın.

MLE \Rightarrow Maximum Likelihood Estimation



$$\begin{aligned} \text{Likelihood} &= 0.98 \times 0.9 \times 0.8 \times 0.7 \times (1-0.3) \times (1-0.1) \times (1-0.05) \\ &= 0.31 \end{aligned}$$

Maximize etmek istedigimiz değer.

Bu değer 1'e ne kadar yakın olursa, model o kadar başarılı demek.

Bunun anlamı:

Linear Reg ile Logistic Reg. cizgilerinin kesistikleri noktanın "üstündeki değerlerin büyümecini, altındaki değerlerin küçülmesini istiyoruz ki Likelihood büyüsün."

$$\text{Likelihood} = P \cdot (1-P)$$

Yukarıda kalanlar



Likelihood' da değerlerimizi artırmak istiyorsak
ki başarımız artırsın.

Ana elinizde "Gradient Descent Algoritması" var.

Bu algoritma değerlerini minimize etmeye çalışır.

Likelihood' da değerlerini maximize etmeye çalıştığımız için bu algoritma Likelihood için kullanılır.



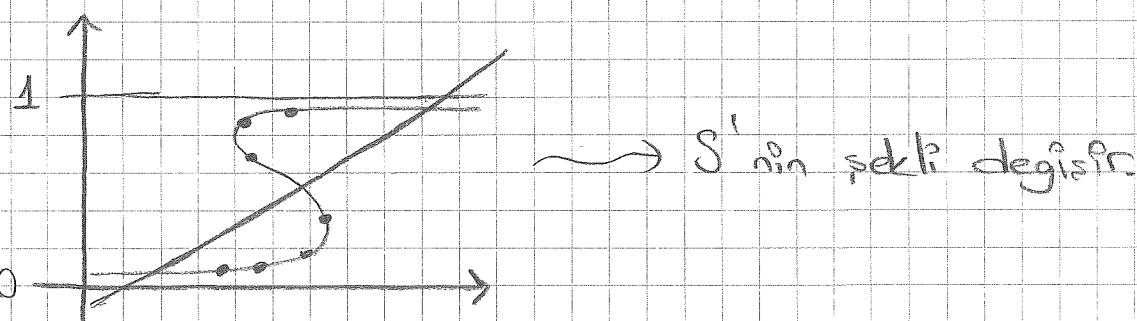
Bu yüzden "Cost function (Log Loss)" fonksiyonu
oluşturulmuş.

Cost function minimize edilince Likelihood
maximize edilmiş olur.

Cost function her zaman negatiftir. Onu minimize
edince en yüksek skor elde edilmiş olur.

(*) Log Loss function nasıl değişir?

Gradient Descent Algoritması yeni bir line çizer.
Bu line'a göre logistic line' da değişir. Böylece
izdüşümler de (y eksenine karşılık gelen nokta) değişir.

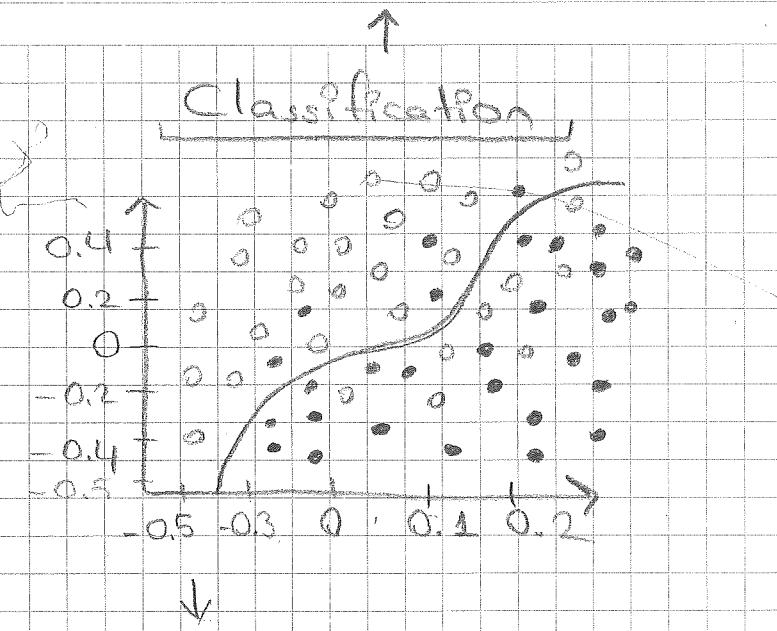
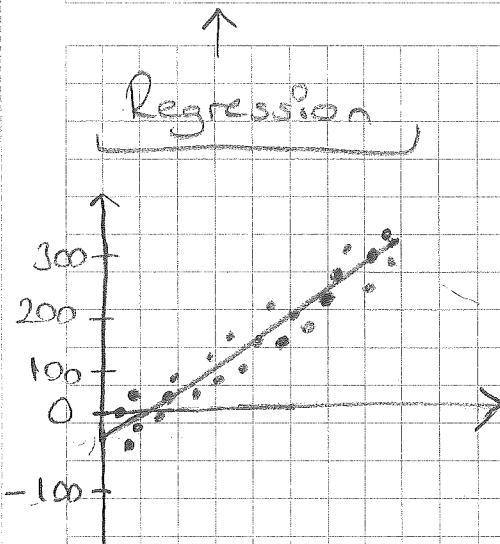


Böylece Likelihood değerleri de değişir.

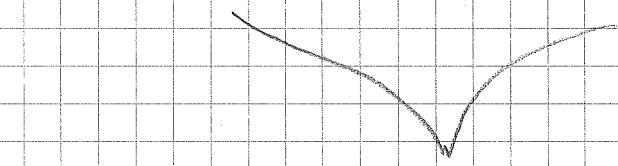
"OZETLE :

- ① 0-1 noktalarını probability'e çevir (0-1 arasına)
- ② Değerleri log ile $+\infty$ ile $-\infty$ arasına taşı.
- ③ Araya bir line çiz.
- ④ Noktaların izdüşümelerini bul.
- ⑤ 0 ve 1 class'ının izdüşümelerini teksir probability'e çevir.
- ⑥ Yani 0 ve 1 olan değerleri logistic line üzerine taşıdık.
- ⑦ Line değerini iyileştirmek için Likelihood'ı最大化 etmek gereklidir fakat Gradient Descent Algoritması minimum etmek üzerine çalıştığı için "Log Loss Cost Function"ı kullan.

"Hava sıcaklığı ne olacak?" "Hava sıcak mı soğuk mu?"



Araya bir decision boundary
koyarak ikinci kategoriyi birbirinden
ayırmak amacları.
(Sigmoid function ile)



Aynı soru ikinci şekilde de sorulabilir:

Regression ile \Rightarrow Sıcaklığı tahmini yapılıp regression hə-
sabı yapılır.

Classification ile \Rightarrow Threshold belirlenir.

(20°C üstü sıcak, altı soğuk gibi)

Classification Error Metrics

① Confusion Matrix :

		Predict Class	
		+	-
Actual Class	+	TP	FN Type II Error
	-	FP Type I Error	TN

In Python:

```
confusion_matrix(y, y-pred)
```

True-Positive : Gerçekte (+)

dogrular tahmin

Actual \rightarrow 1

Predicted \rightarrow 1

False - Positive : Gerçekte (-),

yanlış tahmin

Actual \rightarrow 0

Predicted \rightarrow 1

True - Negative : Gerçekte (-),

dogrular tahmin

Actual \rightarrow 0

Predicted \rightarrow 0

False - Negative : Gerçekte (+), yanlış tahmin.

Actual \rightarrow 1

Predicted \rightarrow 0

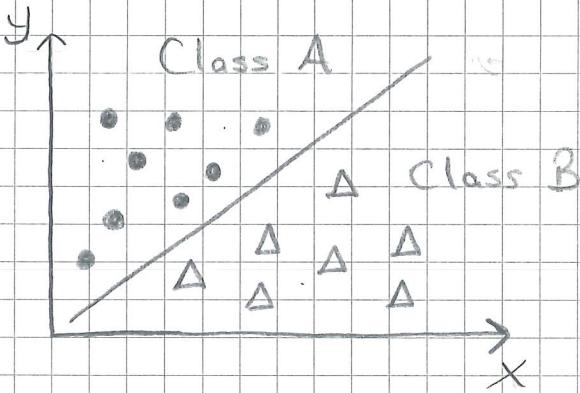
Actual	1	1	0	0
Predicted	0	1	1	0
	FN	TP	FP	TN

Negatif tahmin (N)
Yanlış (F)

Pozitif tahmin (P)
Doğru (T)

Pozitif tahmin (P)
Yanlış (F)

Negatif tahmin (N)
Doğru (T)



Confusion Matrix Üzerinden tahmin ettiklerimiz:

① Accuracy

② Recall

③ Precision

④ f1-Score

Class'ları düzgün ayırabildi mi
ayıramadı mı?

① Accuracy :

+	TP	FN
-	FP	TN

→ Accuracy

$$\frac{TP+TN}{TP+FN+FP+TN}$$

Actual : 1 1 0 1 0 0 0 1 1 0

Predicted : 0 1 1 1 0 0 0 1 1 1

X ✓ X ✓ ✓ ✓ ✓ ✓ ✓ ✓ X

$$\text{Accuracy} = \frac{\text{Doğru bilinen tahminter}}{\text{Top tahminter}} \times 100$$

$$= \frac{4}{10} \cdot 100 \rightarrow \text{Accuracy} = \% 40$$

Accuracy Tercih edilmez. Nedeni?

Mesela 63 tane hasta var.

Gercekte hastalarдан \rightarrow 15 Kanser
 \rightarrow 58 saglıklı

Tahmin \rightarrow 60 doğru
 \rightarrow 3 yanlis

$$\text{Accuracy} = \frac{60}{63} \cdot 100 = \% 95$$

Data'daki kanser hastalarının sayısı çok az iken, sağlıklı bireylerin sayısı çok fazla. Yani aslında Accuracy'ın başarısı daha çok sağlıklı bireyler üzerinde geliyor. Ama biz 5 kanser hastamızı tespit etmek istiyoruz.

Yani datadaki class'lardaki observation sayıları arasında çok fark varsa Accuracy bu işi yanıltır. Kanser olan bireyler üzerinde iyi bir tahmin yapısını diyeceğiz.

$\text{Recall}(x)$
var mıdır?

2) RECALL :

(Sensitivity)

TP	FN
FP	TN

→ Recall

"Gercekten kanser olan hastalarin kacını bildim?"

Actual : 1 1 0 0 1 0 0 1 0 0
Predicted : 0 1 1 0 1 0 0 0 1 0

Gercekte kanser olanlar : 4 kişi

dogru tahmin : 2 kişi

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$= \frac{2}{4} \Rightarrow \text{Sensitivity} = \% 50$$

3

Precision :

+ TP	FN
FP	TN

Tahmin etti
ve bildi

Precision

Sadece pozitif tahminlere odaklanıyoruz.

"Yaptığım pozitif tahminlerin ne kadarı doğru?"

Actual : 1 1 0 0 1 0 0 0 0 1

Predicted : 0 1 1 0 1 0 0 0 0 0

Pozitif tahmin edilen : 3 kişi

Gercekten pozitif olan : 2 kişi

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$= \frac{2}{3} \rightarrow \text{Precision} = \% 67$$

! Recall'ın tersi

• ↑
④ SPECIFICITY :

TP	FN
FP	TN

→ Specificity

"Kanser hastası olmayanların kaçını bildim?"

Actual : 1 1 0 0 1 0 0 0 0 0 1

Predicted : 0 1 1 0 1 0 0 0 0 0 0

FN TP FP TN TP TN TN TN TN FN

TN ⇒ 5 Kişi

FP ⇒ 1 Kişi

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$= \frac{5}{5+1} = \frac{5}{6} \Rightarrow \text{Specificity} = \% 98$$

! Recall ⇒ Kanser hastalarına odaklandı. (1'e bakar.)

Specificity ⇒ Kanser olmayanlara odaklandı. (0'a bakar.)

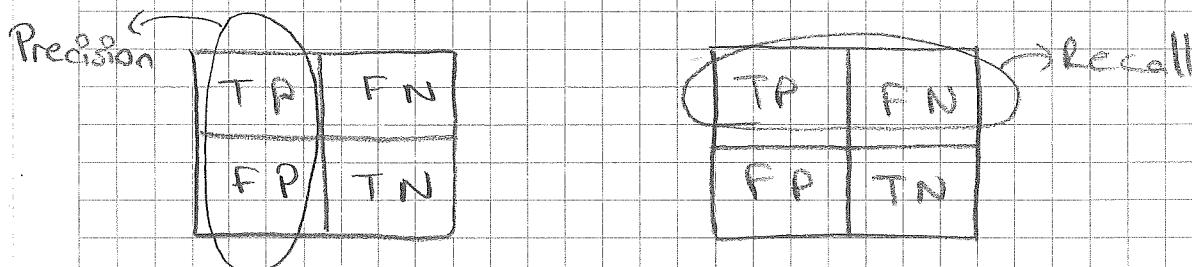
(5)

F1-Score

Precision ve Recall'ın harmonik ortalamasıdır.

Harmonik ort. alınır çünkü eğer eşit dağılmayan bir veri seti varsa, sonuçlardan biri 0 çıkarsa F1-score da 0 çıkar ve modelin başarısızlığı ortaya atılmış olur. Unbalance durumda yaniltıcı sonuç vermez.

$$F_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$



Actual : 1 1 0 0 1 0 0 0 0 1

Predicted : 0 1 1 0 1 0 0 0 0 0

FN TP FP TN TP TN TN TN FN

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{2}{3}$$

$$F_1 = 2 \cdot \frac{\frac{2}{3} \cdot \frac{2}{4}}{\frac{2}{3} + \frac{2}{4}}$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{2}{4}$$

$$F_1 = \% 57$$

F_1 -Score formülüne göre Recall ve Precision ters orantılıdır. Bu yüzden F_1 -Score'ı artırmak için hangisi iyiysse o yukarı çekilir, digeri de düşer.

Model Karşılaştırması Yeterlikleri:

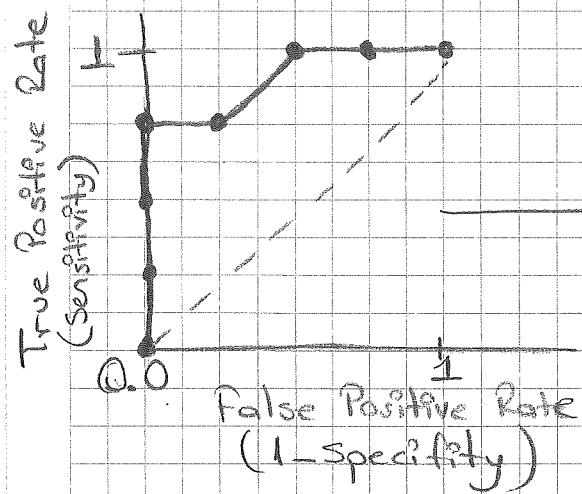
- ① ROC / AUC (Balanced datalarab)
- ② Precision-Recall Curve (Imbalance datalarab)

Receiver Operator Characteristics → ROC

Area Under Curve → AUC

ROC / AUC

Model gücünü değerlendirmek için kullanılır.



True Positive Rate yüksektikçe, $\{$ iyi
False Positive Rate düşükse $\{$ tahmin yapar.

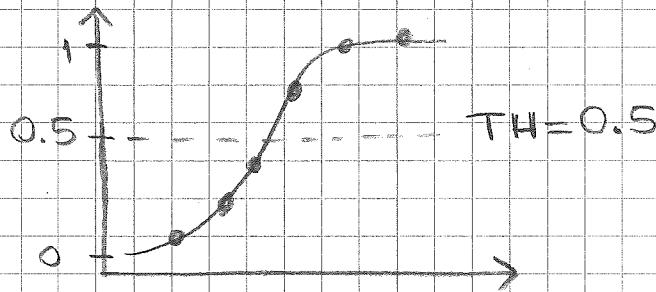
$$\text{True Positive Rate} = \frac{\text{TP}}{\text{(Sensitivity)}} \downarrow \text{Recall}$$

$$\text{False Positive Rate} = \frac{\text{FP}}{\text{(1 - Specificity)}} \quad \text{FP} + \text{TN}$$

- ▼ "Ölustürdüğünüz modellerden hangisi daha güclü?"
- Bu nedenle için ROC / AUC kullanılır.

ROC / AUC Nasıl Gelir?

Threshold değerini default $\rightarrow 0.5$



Gesilli? TH değerlerini deney. Mesela

TH = 0

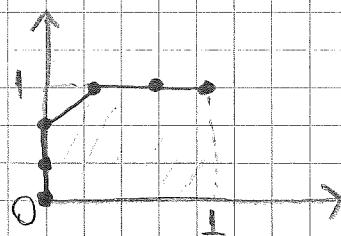
TH = 0.2

TH = 0.8

TH = 1

Threshold değerlerine göre yeni logistic Prc', cizer ve ona göre TP, FP, FN, TN değerlerini hesaplar

Bulduğum değerleri formülde yerine koyar ve grafiği cizer. Altta kalan alan ne kadar büyükse model o kadar başarılıdır.

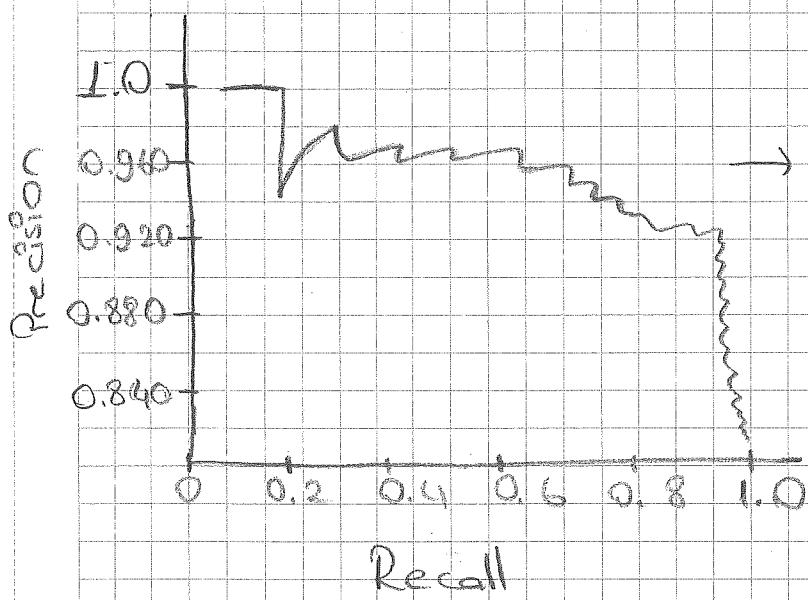


⚠️ Imbalance datalarında (class sayıları uyumsuz),

ROC/AUC' a bakılmaz.

Precision - Recall Curve' e bakılır.

Precision - Recall Curve



ROC / AUC'den farklı
sap 'ı tıpkı doğru etan
antitik model ile-
siyor denekti.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

K. Nearest Neighbors Theory (KNN)

- (*) 'A lazy learner' bir algoritmadır.
 - (*) Training data yoktur.
 - (*) Bu tür algoritma hafızasında储存 eder, yeni gelen data nereye gelirse en yakınındaki komşularına göre o datayı sınıflandırır.
 - (*) Linear Regression ve Logistic Regression parametric algoritmalarıdır. Fakat KNN, non-parametric bir algoritmadır. Yani b_1 ve b_0 gibi katsayılar yoktur. Hiçbir hesaplama yapmaz.
 - (*) KNN genelde az verik data setlerinde kullanılır. Büyük data setleri için uygun değildir.
 - (*) Low dimensional datasets, } gibi alanlarda fault detection } kullanılır.
Recommender systems ... }

$k=5$
Hyper Parameters
weights = 'uniform'
metric = 'minkowski'
 P

① "K" \Rightarrow Sample'ın hangi class'a ait olduğunu K

Konsuluk sayısına göre karar verir. Bu yüzden K seçimi çok önemlidir. Örneğin;

$k=5$ verdiğinde en yakın 5 komşuya bakar, en çok hangi class'tan etmen var ise sample'i o class'a atar.

↑ Eğer K değeri çok büyük seçilirse "large bias" durumu ortaya çıkar. Underfit durumu ortaya çıkar.

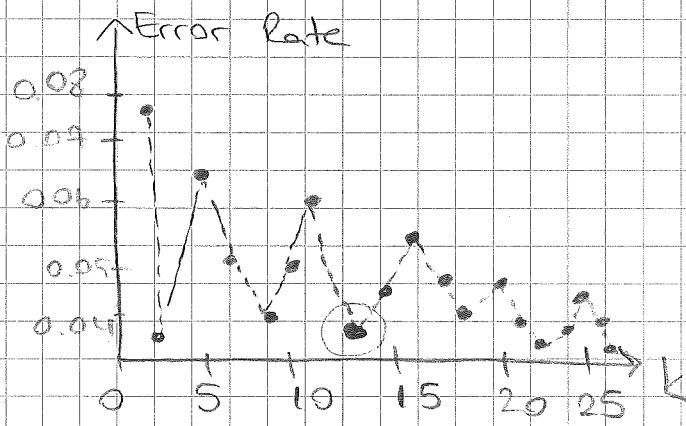
↓ Eğer K değeri çok küçük seçilirse overfit durumu ortaya çıkar. Train set'te çok iyi bir başarı elde eder ama test datalarında çok kötü tahminler yapar.

Bu yüzden optimal K değeri bulmak önemlidir.

Optimal K yi bulabilmek için; train-test dataları bölgür. K için bir aralık verilir ve bu her K değeri için Cross Validate işlemi yapılarak en az hata veren K değeri seçilir.

Elbow Method

k için verilen analitikteki her k değerini için error metrikleri hesaplanır. Bu error metriklerine göre bir grafik elde edilir.



K 'nin çok yüksek olmasına isteniyorsa, C_{Jaku} ve der fit tehlikesi var. Error'un çok çok yüksek olmasına isteniyorsa, Error'un daha düşük olan k değerlerini almaması râgmen arada çok fazla bir error farkı olmadığı için, k 'yı kucuk tutmak adına $k=13$ olarak seçilebilir.

KNN metodunda Grid Search sistemi yerine Elbow method ile çizilen grafique göre karar vermek daha mantıklıdır.

GridSearch sistemi k 'yı 20-25 değerleri arasında götürür.

(2) weights $\Rightarrow \{ \text{'uniform'}, \text{'distance'} \}$

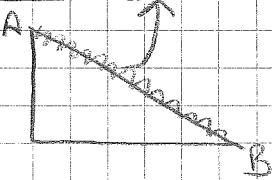
uniform \rightarrow Her bir komşunun 1 oyu var.
↳ (Default değer)

distance \rightarrow Yakin olan komşunun biraz daha çok olana
üstlenir. Yakin olanın oyu artar. Bu şekilde
tek bir komşu, bir den fazla komşudan daha azı
olduğu için üstlenlik kazanabilir.

(3) metric \Rightarrow Mesafe hesabı yapın parametre.
(Hem kategorik, hem continuous datalarda kullanılır.)

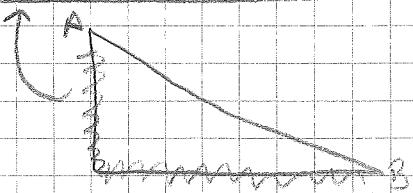
Euclidean Distance } Bu pikrinin varlığı olan
Manhattan Distance } "Minkowski" default parametre.

Euclidean Distance \Rightarrow Küçük yaşlı mesafe



Minkowski bu
iki yaşlı hesa-
bını da kullanır.

Manhattan Distance



"p" parametresi ile
minkowski'ye karar
verilir.

(4)

p \rightarrow Minkowski parametresi için kullanılan bir parametre.

p = 1 \rightarrow Manhattan distance gibi davranışır.



Hataları cezalandırdığı için outlier'larla karşı dirençli.

Daha çok multidimensional data setlerinde kullanılır.
(Feature sayısı > 5 olan data setlerinde)

p = 2 \rightarrow Euclidean distance gibi davranışır.



Outlier'lar ile mücadelede iyi değil.

Daha çok küçük data setlerinde kullanılır.

(Feature sayısı < 5 olan data setlerinde)

! KNN'de scale işlemi zorlu. !

Cümlü mesafe tabanlı bir algoritma.

Olumlu Yönler

(*) Lazy learner oldugu için assumption'lar ile ilgilenmez. Bu yuzeden her data da kula-

karilabilir.

(*) Anlamasi ve uygulamasi kolaydir.

(*) Training yoktur.

(*) Distance metrikleri degistiripse farklı sonuçlar elde edilebilir.

(*) Hem classification hem de regression analizlerinde kullanilabilir.

Olumsuz Yönler

(*) Böyük datalar da kullanis-

sız. (Böüm noktalar tek tek

hesaplandigi için uzun süren.)

(*) Outlier'lardan çok etki-

lenir. (Dengesiz data setleri için uygun degil.)

(*) Over fit tehlikesinden dolayı

k seçimi çok önemli.

(*) Scaling işlemi zoruldur.

Regression analizlerinde de yine konsularına bakar.

Bunların ortalamasına bakarak sample, bir class'a atar.

Classification \Rightarrow Konsularin modunu bakar.

Regression \Rightarrow Konsularin ortalamasina bakar.

Support Vector Machines Model (SVM)

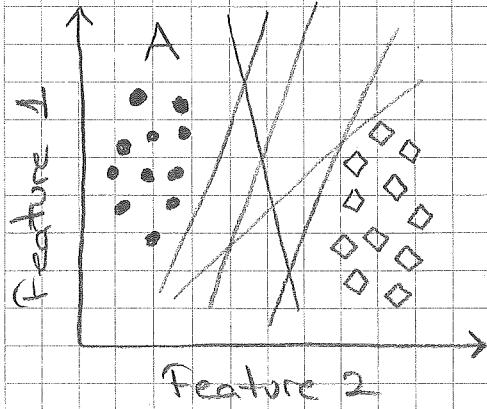
Text Classification

Image Recognition

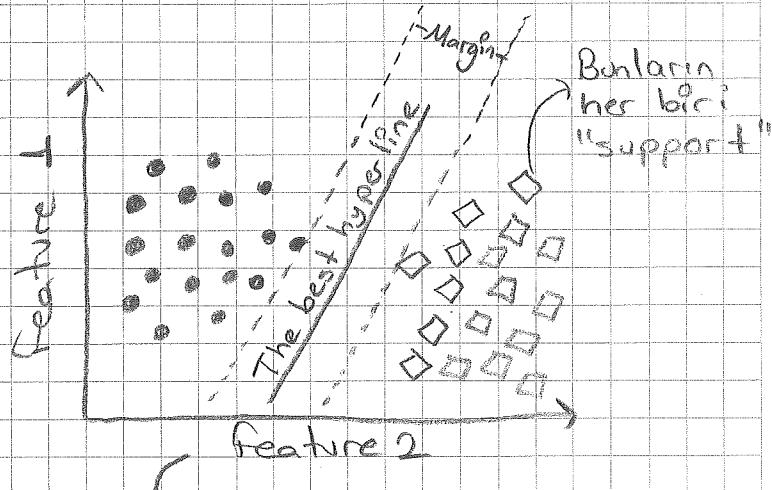
Image-based Gender Detection

gibi bütün classification sonuçlarında kullanılır.

Linear olarak ayrılabilen datalar da mümkün.
sonuçlar venir.



iki class arasında sınırlı
çapılı çözülebilir.

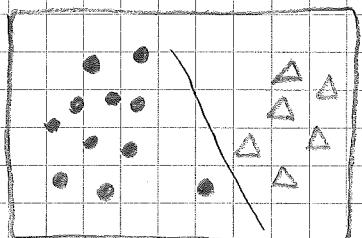


Ama ben böyle 2 support
seçeyim ki, ortalarının en
en uzaklı iki support'a
en uzak olsun.

Margin \Rightarrow Seçilen supportlar arası uzaklık.

3 boyutlu olduğunda ise, araya line yerine hyperline
olar. Class'ları birbirinden ayırt ve 3 boyutlu bir görüntü
segler.

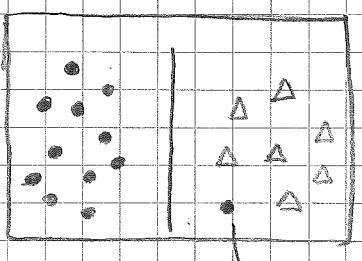
Hard Margin - Soft Margin



Hard Margin

Classları tam ayırmak için böyle bir çizgi olursa "over fitting" denir. Hard margin olur.

Test data'sında ayrimi tam yapamaz.



Soft Margin

Bunu hata olarak kabul etmiş ve line'ı diğer sapt noktalarına göre güncellmiş. Train data'sında biraz hata yapmış olsa da test data'sında daha iyi tahminler yapacak.

Over fitting olmasından sonra bazı hatalara izin verir.

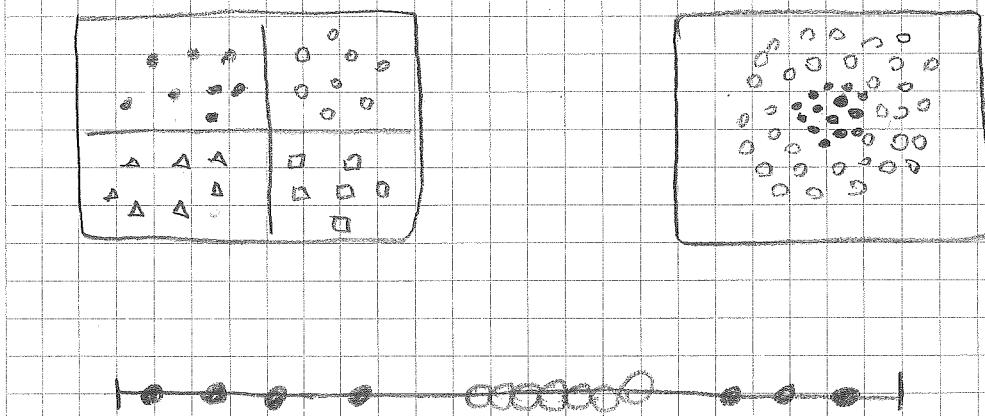
Bu işlem "C" parametresi ile regularization uygulayarak yapılır.

Yani margin, C parametresi ile ayarlanır.

► Regularization \Rightarrow Hatalarla mücadele etmek için hata eklenerek over fit sorunundan kurtulma işlemi.

C parametresi \Rightarrow Hataya ne kadar tolerans göstericeğim? (Over fit - under fitler kurtarma)

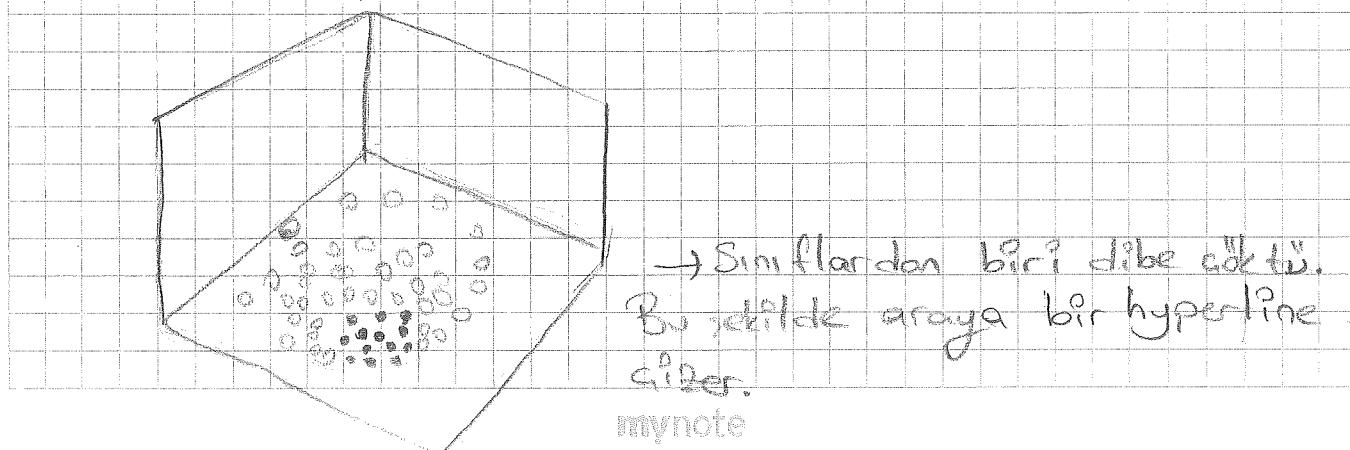
KERNEL TRICK \rightarrow 2D'den 3D'ye



Yukarıdaki datalacımında lineer bir line çizilemiyor. İst. işe gemiciz bir durum var. Böyle durumlarda kernel trick ile boyut değiştirmeye işlemi uygulanır.

Kernel'in yaptığı iş:

- ① Boyutu bir üst boyuta çevir. (2D'den 3D'ye)
- ② Bu işlem computational power gerekliliğinden hallede.



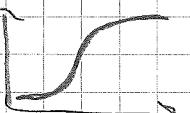
\rightarrow Sınıflardan biri dibe çıktı.
Bu tekilde araya bir hyperline
gizlenir.

Böyle bir durada 1D'ler 2D'ye çevirir ve araya hyperline'ler gizler.

SVM özetle \Rightarrow Birbirine en yakın supportlar arasındaki marginı belirleyip ortaya hyperPlane oluştur.

Eğer data linear olarsa ayrılmayorsa
Kernel ile boyutu değiştirir, sonra hyperline
dizesi.

Kernel Parameters

- ① Linear Kernel  Datada birer bir g鰐k『n』de yoksas GridSearch'e sokulur ve en iyi skor veren yöntem seçilir.
 - ② Polynomial Kernel 
 - ③ rbf  ! Kernel'in default'ı 'rbf' tır. En iyi sonucları, genelde bu
 - ④ sigmoid (Deep learning'de kullanılıc.) verir.
 - ⑤ precomputed (Kendine güveniyorsan kendi kernel'ini kendin mynote hesapla, buraya at :)

Linear \Rightarrow Feature sayısı fazla, satır sayısı az ise önerilir.

rbf \Rightarrow Complex ve nonlinear datalarda tercih edilir. (Lineer datalarda bile kullanılabilir.)

Feature sayısı az, satır sayısı fazla ise önerilir.

C Parameter, (default = 1)

Margin'in genişliğini ayarlar.

C artıca \uparrow Margin hard' lasır. (Overfitting)

Gamma Parameter, (default = 1)

Modelin complexity'sini ayarlar.

! Gamma sadece nonlinear kernel'larda kullanılır.
(rbf, poly, sigmoid)

Gamma arttıkça \uparrow complexity artar. \uparrow

Yani linear kernel \Rightarrow C parametresi?

nonlinear kernel \Rightarrow C parametresi, gamma,

NonLinear Kernel'da C ve gamma'nın optimum değeri bulunur. (Grid Search ile)

C ve gamma arttıkça overfit durumu artar.

SVM Olumlu Yönleri

Olumsuz Yönleri

- (*) Computer Vision sorunlarını çözer. (*) Kernel ve parametre (2D'den 3D'ye taşıma gibi) seçimi çok hassas.
- (*) Generalization error'u düşür. (*) Outlier veriler bu modelde sorun çıkarabilir. Çünkü
- (*) C ve gamma parametreleri ile onlar yüzünden hyperplane'yi ayırayarak complexity çözüm yapısını yanlış能得到.

Decision Tree Theory (DTT)

Hem regression,

Hem classification da kullanılır.

Classification problemlerinden:

Medical Diagnosis,

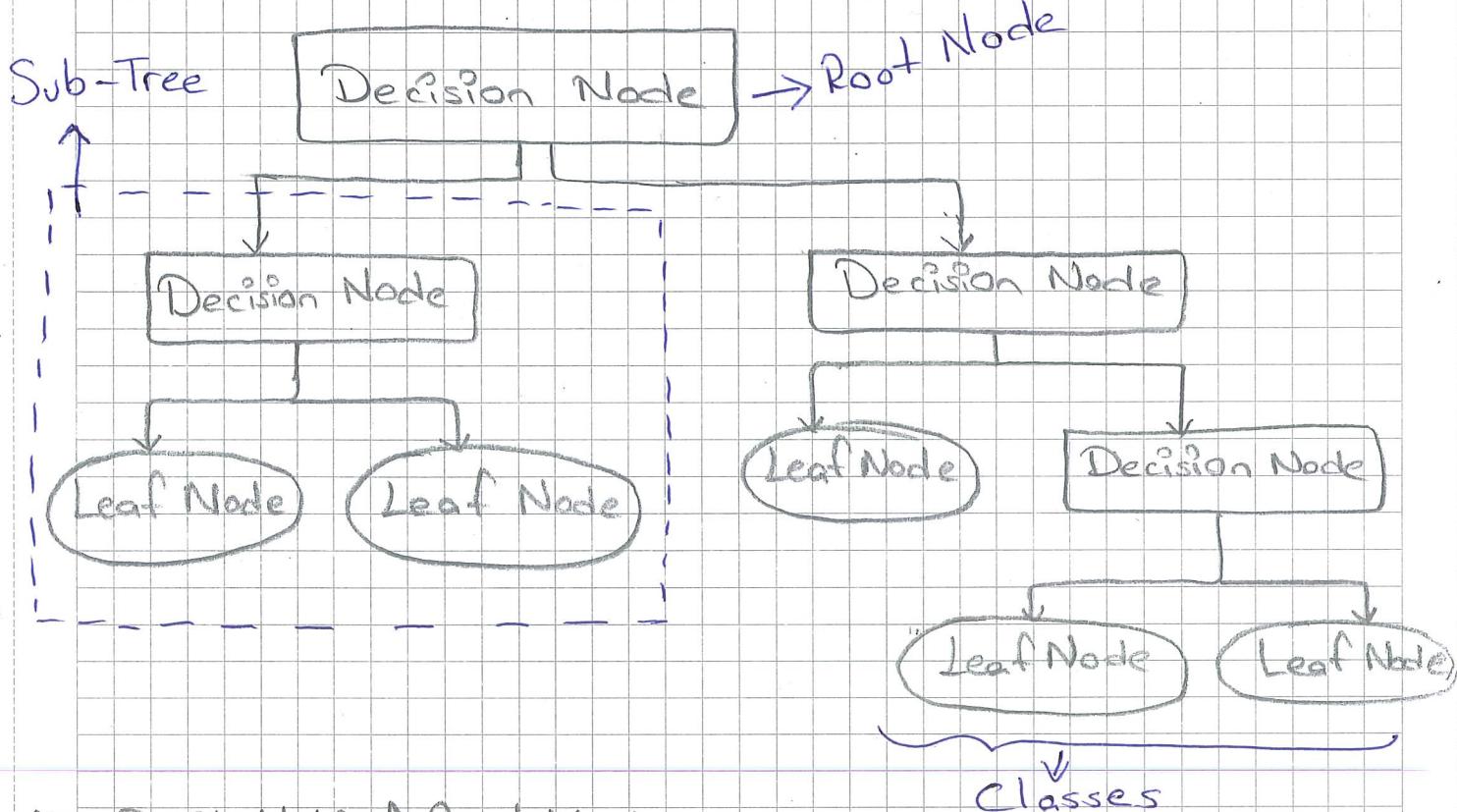
Text Classification,

Credit Risk Analysis'te de kullanılır.

Linear Regression gibi araba tahminlerinde de kullanılır.

Ters ağac

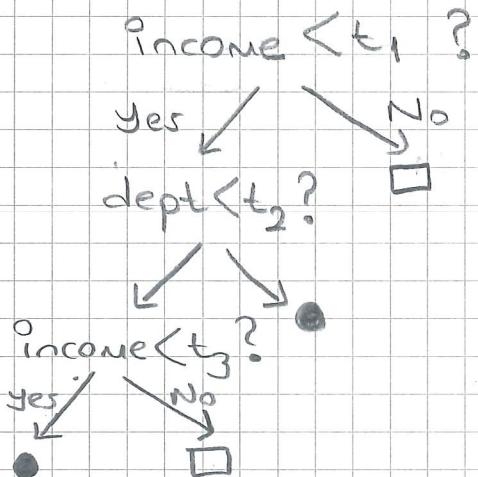
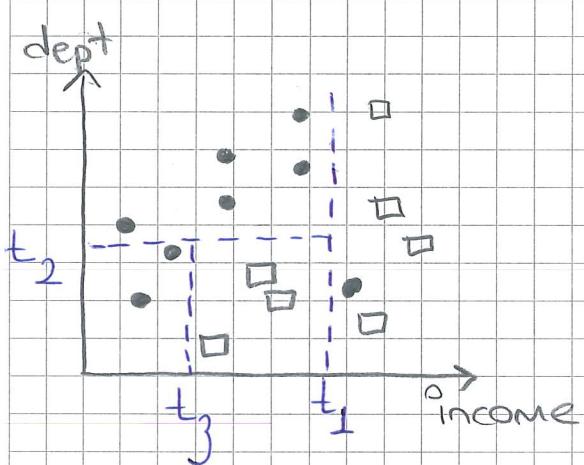
Decision Tree Diagram



En üstteli \Rightarrow Root Node

En alttakı \Rightarrow Leaf Node

Araďakilerin hepsi \Rightarrow Decision Node



Once ayırabildiği en iyi yerden böler, yani income ekseniindeki t_1 'den.

t_1 'den büyük olanlar pure olarak ayrıldı.

Bu sefer en iyi ayımı dept süzününden t_2 'den yapar, en son t_3 'ten böler ve sınıflandırma bitir.

Amaçımız ; highest homogeneity. Yani bölüğümlü bölgelerde tek bir sınıfın kalması.

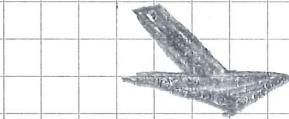
Decision root'u hangi attribute ile seçeceğim?

Sectığım feature'i neresinden ayıracığım?

Ağacın büyümeye ne kadar izin vereceğim?



Gini Index
(Gini Impurity Index)



Information Gain / Entropy

1

Gini Index

0 bölgedeki yanlış hesaplamaların değerini minimize etmeye çalışır.

$$Gini = 1 - \sum_{i=1}^n p^2(x_i)$$

$1 - (i \text{ class'ında olma olasılığı})$



Amaçımız Gini'yi küçütmek.

Eğer ayrılan sınıf "pure" ise 1 olma olasılığı

$1 - 1 = 0$ olur, yani basarı %100 olur.

Gini Index Nasıl Çalışır?

Diyelim ki 3 feature ve 1 target limiz var.

3 feature için Gini formülü uygulanır ve Gini değerleri bulunur.

column-1 = 0.364 column-2 = 0.360 column-3 = 0.381

Gini index, "root node" olarak column-2'yi seçer.
Çünkü en düşük Gini'yi arıyoruz. En iyi ayrim Gini Index yapmış.

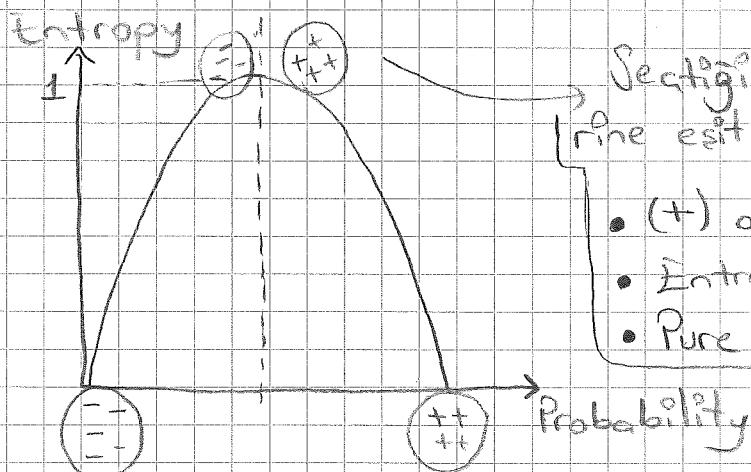
(2)

Entropy Formula

Entropy = Karsı demek.

Entropy'nin en az olmasının istiyoruz.

$$H = - \sum_{i=1}^n P(x_i) \log_2 P(x_i)$$



Sectiği bölgeler (+) ve (-)'ler birbirine eşit olursa "max entropy" olur.

- (+) olma olasılığı $\Rightarrow 50\%$
- Entropy $\Rightarrow 1$
- Pure değil

Sectiği bölgeler "pure" olursa entropy 0 olur.

- (+) olma olasılığı $\Rightarrow 100\%$
- Entropy $\Rightarrow 0$
- Pure

Information Gain (IG)

En fazla bilgiyi hangi node üzerinde hesaplıyoruz?
(Root node nedir?)

Bize entropy'yi IG verecek.

Gini Impurity \Rightarrow Düsürmeye çalışıyoruz.

Information Gain \Rightarrow Yükseltmeye çalışıyoruz.

Bu \Rightarrow
analırsa
diger
yükselir.

Information Gain Nasıl Gelsin?

Mango, apple, banana class'larına ait ağırlık ve
seçimluk feature'ları olsun:

Height(cm)	Width(cm)	Class	Target
1	1	Mango	
1	2	Banana	
1	3	Apple	
2	1		
2	2		
2	3		
3	1		
3	2		
3	3		
4	1		
4	2		
4	3		
5	1		
5	2		
5	3		
5	4		
6	1		
6	2		
6	3		
6	4		
7	1		
7	2		
7	3		
7	4		
8	1		
8	2		
8	3		
8	4		

Mesela Height > 10 üzerinden entropy'yi hesaplar ve;

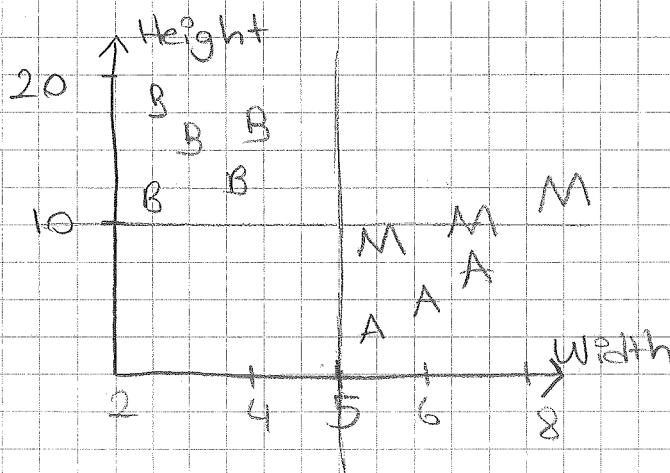
$$IG(\text{Height}) = 0.696$$

Width > 5 üzerinden entropy'yi hesaplar ve;

$$IG(\text{Width}) = 0.97$$



Bu ikisinden büyük olanı seçer. (Gini'nin aksine)



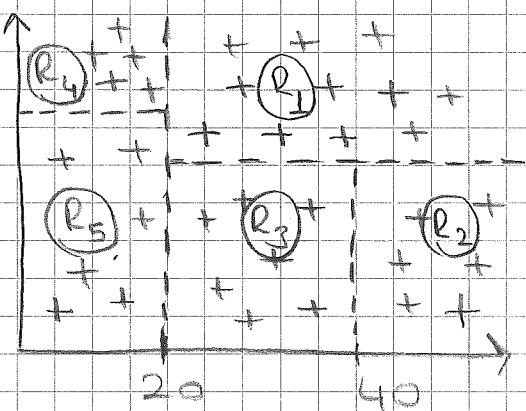
\rightarrow Width 5'ten cizgide
Banana için purity elde
edilir.

Decision Tree ile Regression - Variance

CART \Rightarrow Classification and Regression Tree
ilk bu amaca kullanılmış.

Gini Impurity ve Entropy burda yok. Sadece classification da var.

Bu yöntem, Variance üzerinden çalışır.



Variance'a göre sınıfları ayırmak

Mean'e göre prediction.

"ilk nerden bölerssem ki? taraftaki variance'lar nın olur?" Dijital ayırm yapıyor.

Hangi bölgeler arası variance'i en azı indirmeye çalışıyoruz. (Hangi sayılar arasındaki fark en az aynı sınıfta)

Tahminlen ve mean üzerinden yapar.

Hangi bölgelerdeki noktaların mean prediction olarak gen döner?

! Variance düşük olsun istiyorsa ama hataya hiç neden vermezsek datayı ezberter overfitting oluştur.

Bagging'in özel bir hali.

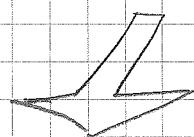
29.01.22

Random Forest, (RF)

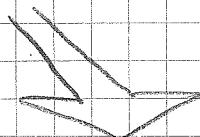
ENSEMBLE METHODS:

Bir çok base model kullanarak ortaya bir meta-model çıkarmaktır.

Yani; bir çok ağac oluşturacak. Buradan gelen sonuçların hepsi birleştirilerek bir tane sonuc bulunur.



Bagging
(Random Forest)



Boosting
(AdaBoost)

Bagging ve Boosting:

Ortak yönler \Rightarrow ikisi de voting üzerine çalışır.

Yani mesela 100 ağac gelir, hepsinin bir orandı oyu olur.

İkisinde de tek bir algoritma var.

Farklılıklar \Rightarrow Bagging'de bütün modeller ayrı ayrı oluşturular. (Bütün decision tree'ler birbirinden bağımsız)

Boosting'de ise bir modelin her seferinde diğer model haber alardır.

Bu yüzden bagging'de her bir base modelin meta-model üzerinde 1 oyu vardır. Boosting'de iki modelin oyu fazla olur.

Yani Ensemble Methods' da bir sonraki adımlarda base model oluştururlar, bunlar bir araya toplanırlar, aralarından bir karar çıkar. Meta model bu toplantıdan çıkan karara "göre" son karara varır.

Bagging ile Boosting'in bu karara gitme şekilleri farklıdır.

Bagging \Rightarrow Base modeller birbirinden habersiz. (Her model 1 oy)

Boosting \Rightarrow Base-modellerin birbirinden haber var.
(içinde modelin oyu fazla)

! Bagging \Rightarrow Strong algoritmalar kullanır.

Boosting \Rightarrow Weak algoritmalar kullanır.

Tarihsel Gelişimi :

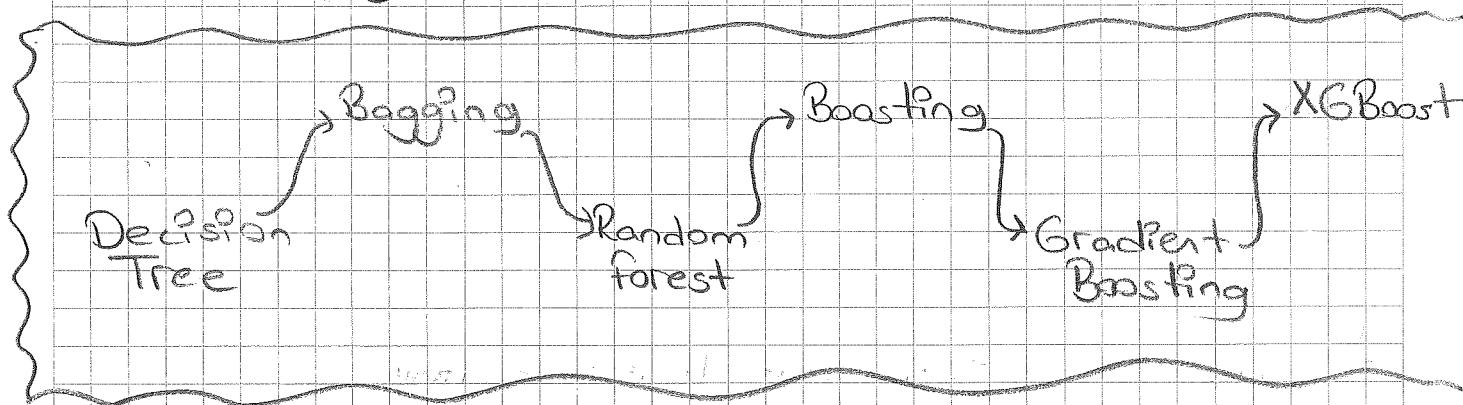
Decision Tree Model'ün variance ilgili problemlerinden dolayı variance sorunu ile başa çıkmak için "Bagging" ortaya atılmış ve bunun ile variance kontrol edilebilmiş.

Daha sonra Random Forest gelmiştir. Random Forest, tabanında Bagging kullanır ama Decision Tree'ün de bazı özelliklerini değiştirecek modelin variance'sını o da oyndar ve daha güzel bir sonuç elde eder.

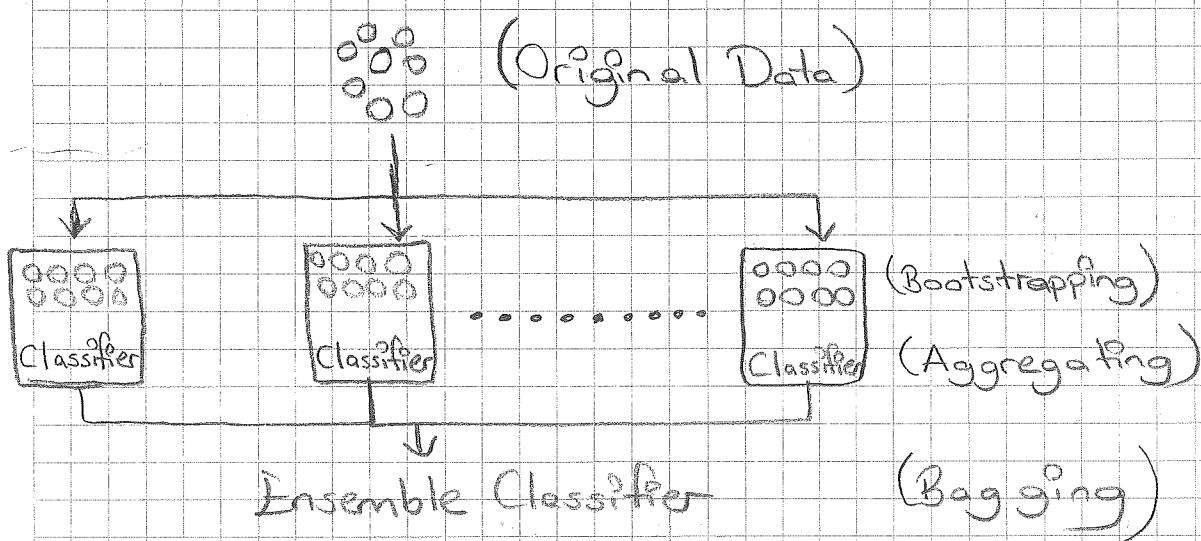
Daha sonra Boosting algoritmları ortaya çıkmıştır.

Daha sonra Gradient Boosting Algoritması ortaya çıkmış. Bunun arkasında Gradient Descent Algoritması çalışır.

En son olarak da, bunların en iyi hali olan XGBoost Algoritması geliştirilmiştir.



Bagging (Bootstrap AGGREGATING) :



Yeni data setleri oluşturma sistemi?

Bootstrapping : Datadan 'n' tane, datanın boyutunda
subsample'lar oluşturur.

Datadan bir örnek çeker, geri atar; çeker, geri
atar. (Duplicate de olabilir) (Data ile aynı boyutta)

Bu şekilde subsample'lar oluşturur. (Classifier)

Oluşan subsample'lar aynı büyüklüktedirler ama
farklı şekilde dirler. Bu da variance', kontrol etmenizi
sağlar.

Aggregating kısmında gelir ve hepsine aynı
"classifier", verir. Oluşan base classifier'lar meta
classifier üzerindeki oylarını kullanır.

Base classifier'lar ile dataya farklı açılardan bakmış
olarız. ○

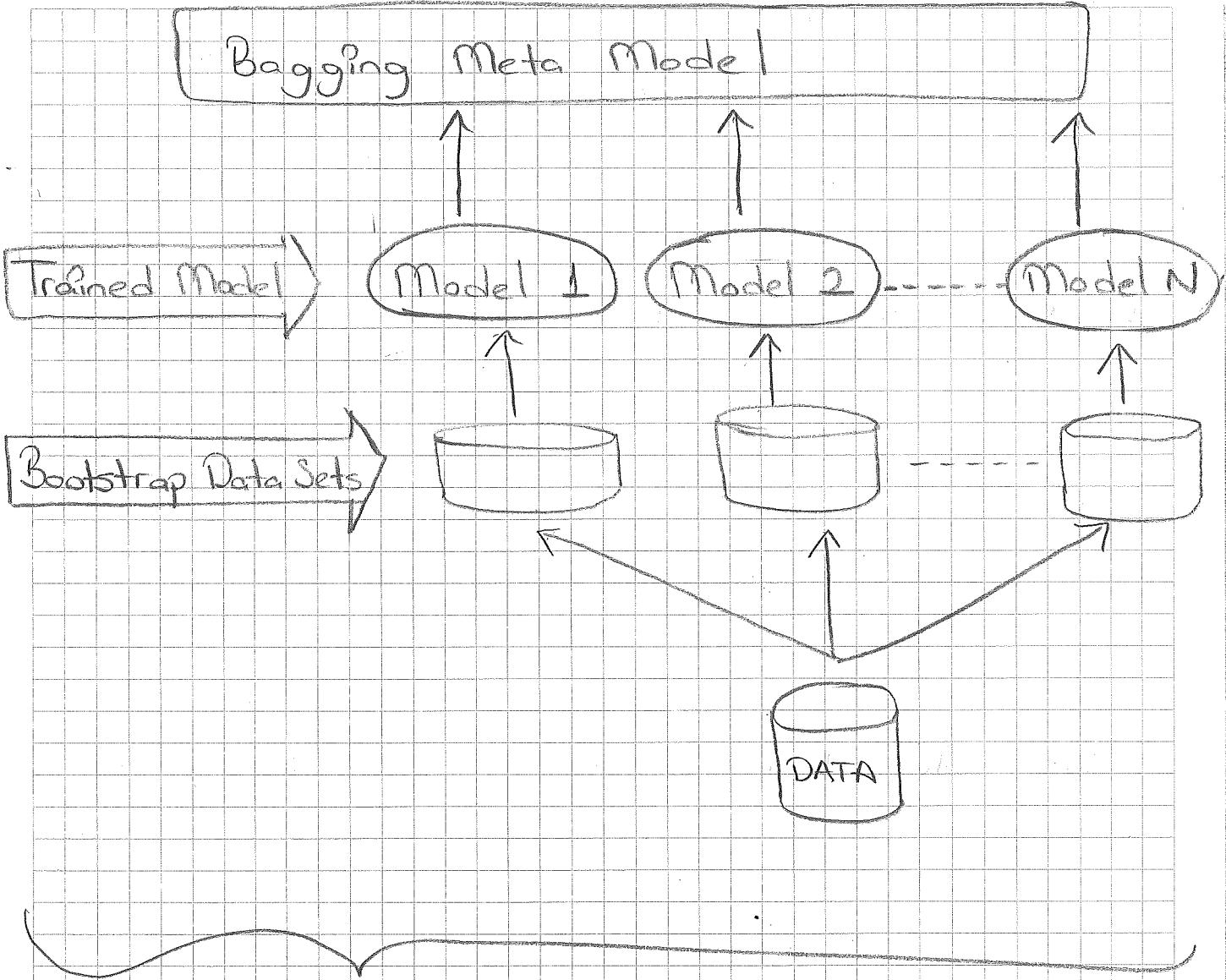
- ① Modellerin n'inci subsample oluşturular.
- ② Bu subsample'ları classifier'lara ya da regression
algoritmalarına verilir.
- ③ Burdan çıkan sonuçlar

Classification ise

VOTING (mode) Üzerinden
Karar verir.

Regression ise

AVERAGE (MEAN) Üzerinden
Karar verir.



Decision Tree

Logistic Regression

SVM

Linear Regression ... gibi istediğiniz modeli
model kismına koyabiliriz.

Random FOREST

Sımdıye kadar Bagging metodundan bahsettiğimizde
Random Forest' da Bagging üzerinde kurulmuştur.
Fakat aralarında farklılar vardır.

Random forest, Decision Tree kullanıyor, bu Bagging olur; Random Forest olmas.

Bagging ile Random Forest farkı:

- ① Subsample oluştururken subsample boyutu $2/3$ 'ür.
(Bagging'de $3/3$ 'ü.)
- ② Feature'lar arasında da seçim yapılır, hepsi alınamaz.
"Verdiğim feature'lar arasında sadece kader feature olur" denir. Her seferinde farklı feature'lar seçiliyor.
İçin bütün ağaclar birbirinden farklı olur.

Böylece hem giren data değişir hem de datanın
üzerinde en fazla 3 feature'lar değişir.

Bütün ağacların Root Node'ları farklı olur.
Bu şekilde Decision Tree'nin sona kadar gitmesini
engellenmiş oluyoruz. (Variance sona kadar gitmez.)

Sonrasında bütün subsample'lar birer prediction yapar ve bunların ortalaması alınır.

"Özetle":

- ① Subsample olarak datanın $2/3$ 'ünü alır.
- ② Aynı Decision Tree'ler oluşturmasının diye bütünsel feature'ların belli bir kısmını random olarak alır.
- ③ Böylece her seferinde Root Node değişir.

▼ Bu tür subsample'lar birer oğan sahip.

- ④ Bu şekilde variancenin kontrol ederek Decision Tree'nin overfitting'e gitme problemini ortadan kaldırır. (Bazı feature'ların öne çıkması engellenir.)

Root Node hem subsample'larından hem de feature'ları "yazdırarak" değiştirmeye salıslıdır.

(Her bir oğan birbirinden farklı olur.)

SVM, Random Forest'a göre daha iyi mi?

HYPERPARAMETERS:

(1) n_estimators: "Kaç tane ağac kurulsun?
(default=100)

Cok ağac olursa işlem çok uzun sürer.

(2) max_depth: Dalları budama işlemi.
("Kaç adım gideyim?")
(default=None)

(3) max_features: "Feature'lardan rastgele kaçıni alayim?"

(4) min_samples_split: "Bölmek için bir dalda
kaç sample bulunur?"
(default=2)

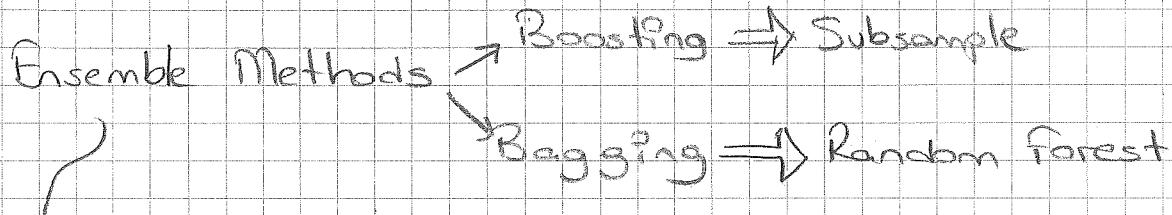
Olumlu Yönler:

- ① Multi-class sınıflarında çok iyi.
- ② Hızlı bir şekilde prediction olabilir.
- ③ Feature importance çok yüksektir.

Olumsuz Yönler:

- ① Küçük datalar için kötü
- ② Maliyeti yükseklik. Gıdu cihazı gerektirir.

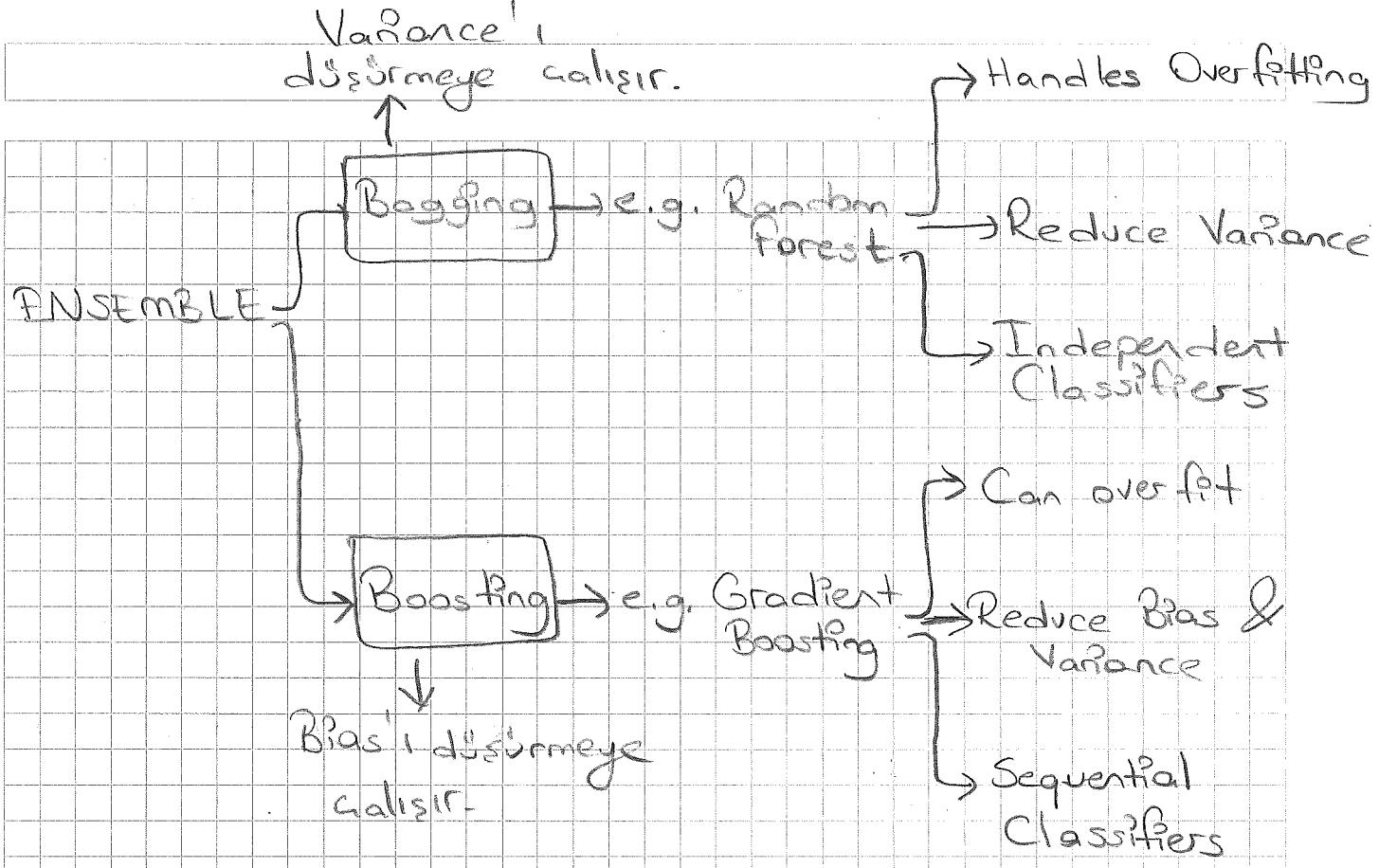
XGBOOST MODEL



Küçük küçük weak learner'lar oluşturarak, bunlardan meydana gelen meta model ile daha iyi prediction'lar elde etmek.

Bagging'de her bir ağaca yeri bir model gires ama Boosting'de aynı datayı ağaslar sırayla ele alır. Yani tek data üzerinde çalışır. (Ana data)
(Subsample olayı yok.)

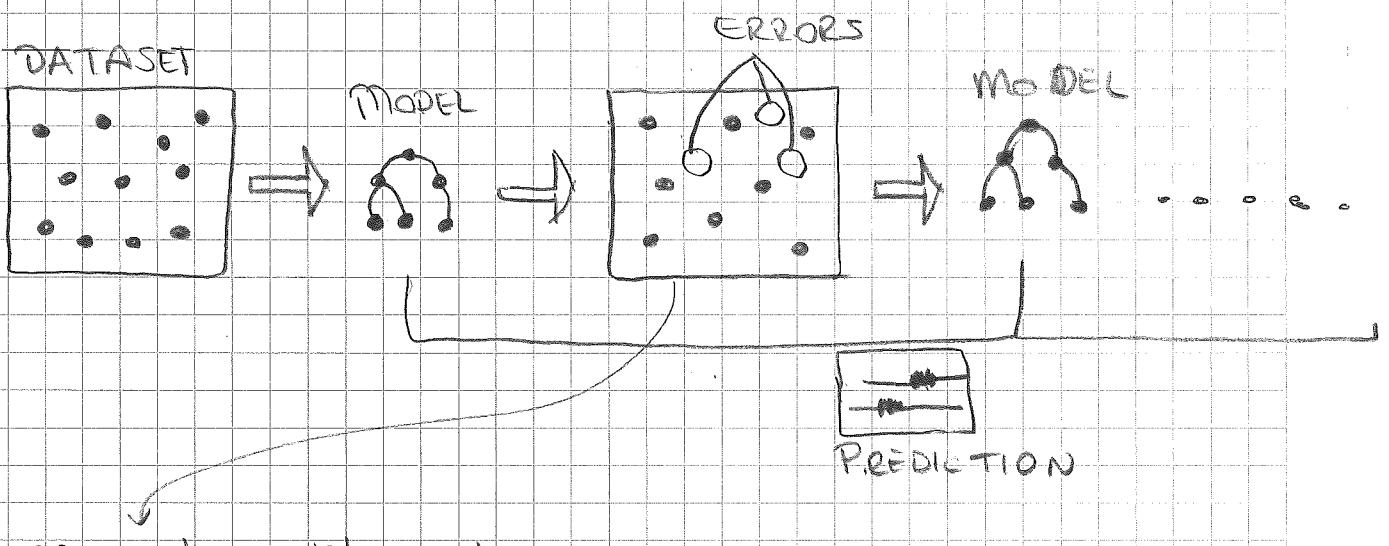
Bagging'de her bir model bağımlı olduğu için her birinin oyu eşit. Ama Boosting'de hepsi aynı datayı ele aldığı için ağırlıklı olanın oyu fazladır.



⚠️ Bagging → Variance'ı düşürür. (Piller paralel bağlı gibi)

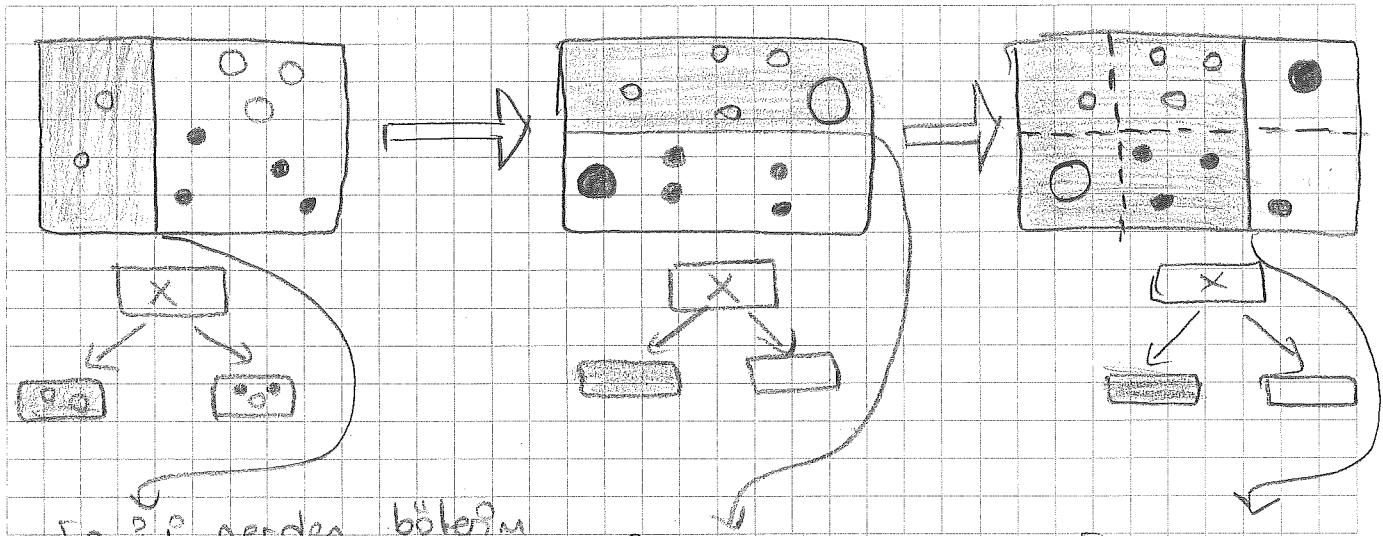
Boosting → Bias'ı düşürür. (Piller seri bağlı gibi)

Boosting'de bir modelden diğerine bilgi aktarımı vardır.



Manipule edilmiş data.

Errotlar ağırlıklandırılır. Bir sonraki adımda modelin bilene döklenen daha çok ağırlık verilen.



En iyi nerden böleme
değip buradan bölmüş.
Yanlış tahmin edilen
bölgeleri bir sonrakine
ağırlandırmak aktarır.

Bu model dataya
sıfırdan bakar. de bu şart ve
Buradan bölmüş sınıflara ayrılmış
tomurlanır.

Ağırlıklandırma'dan karıştı; mesela datada yanlış bilinen değer 9 olsun. Bir dahaki modelde 9 ağırlıklarıdır. Yani 9, 2 kere 3 kere yazılır.

① ADA BOOST Algorithms

HYPERPARAMETERS

base-estimator \Rightarrow default = None (DecisionTree kullanır.)



"Logistic" yazarsak logistic regressioni
bulsun.

n-estimator \Rightarrow Arka arkaya eklenecek ağac adedi.
(default = 50)

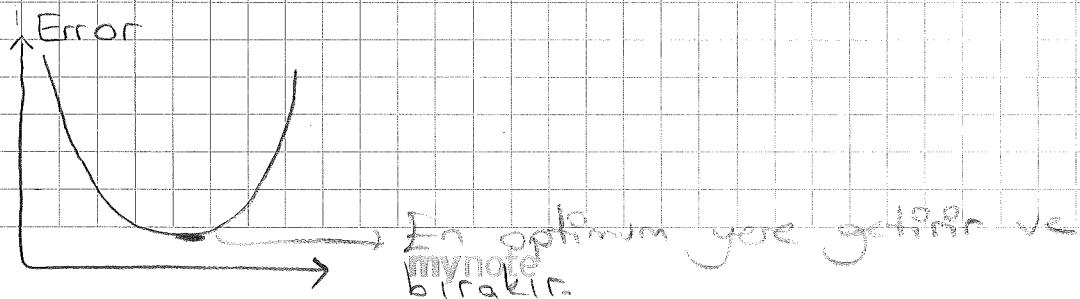
! Logistic, SVM... herhangi bir model
kullanılabilir.

② GRADIENT BOOSTING (GBM)

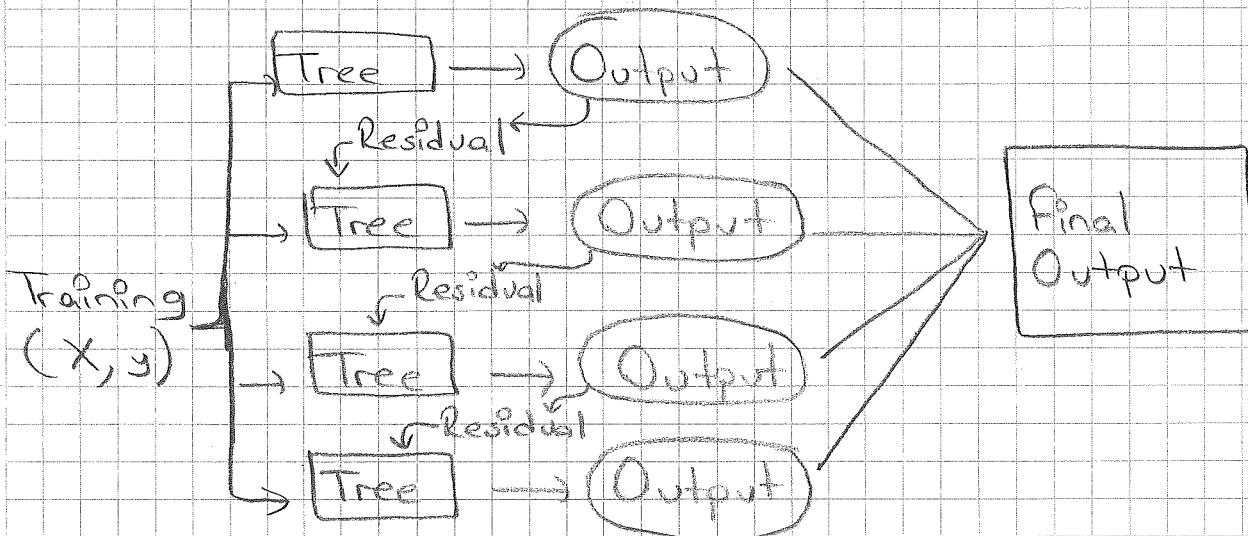
Ağırlıklandırma üzerinden çalışır. Bir önceki modelin yap-
lığı hataları iyileştirme üzerinden çalışır.

Gradient Descent gibi residual'ları aşağı sekmeye
çalışarak hataları aşırı doğrularak sıfıra yaklaş-
tmaya çalışır. Tanrımanı sıfır yaparsa over fitting olur.

Bu yüzden sıfır yapmak yerine sıfıra yaklaşmaya
çalışır.



Sıfır'a ne kadar yaklaştıracagına ağız sayısını ile karar verir.



X' 'i ilk Tree'ye sokar, burada residual olur. ($y - \hat{y}$)

Burdan sonra target (y) ile birisi kalmaz. Artık residual'ları bir dahaki ağaçta aktarır ve o residual'ları tahmin etmeye calısır. Bu şekilde yeri gikan residual'ı bir sonraki ağaçca ileterek, ağaç sayısını kadar isteme devam eder. Her ağaçta residual'ları araltıp sıfır yaklaştırmaya calısır ve optimum yerde bırakır.

(y' 'yi sadece ilk Tree'de residual hesabi için kullanır ve sonrasında residual'ları aşağı çekme hedefi ile devam eder. y' 'yi bir daha kullanmayı.)

Residual sıfır kadar giderse zaten y' 'yi bulmuş olur. Bu da overfitting'e sebepl olur ki bunu istemeyiz.

(3) XGBoost (Extreme Gradient Boosting)

Bir önceki Gradient Algoritması üzerine salıgarak XGBoost geliştirilmiştir.

* Önceden farklı olarak buna regularization eklenmiştir.

Diger modelde Tree Sayısı ayarlanması son adıma kadar gidiş overfitting'e sebep olabiliyordu.

XGBoost kendi içinde bir regularization uygular.
(Hata ekleme)

Diyelim ki data'de missing value'lar var ve çok hızlı sonuç almak istiyoruz. Missing value'ları temizlemeden XGBoost uygulanabilir.

Missing value'ları hata olarak kabul eder ve diğer ağızca atar. Ama missing value'lar bosтан temizlenip verilirse çok daha iyi sonuçlar verir. Bu yüzden model öncesi EDA işlemi mutlaka yapılmalıdır!

* Bir ağadan diğerine geçerken kendi içinde Cross Validate işlemini yapar. Böylece extra Cross Validate işlemine gerek kalmaz.

Biz gene de Cross Validate ve GridSearch yapmalıyız. (Hyperparametreler için.)

- (*) Bütün tree'ler paralelleştirilebilir. Böylece işlemeler hızlanır olur.
- (*) İşlemi cache seviyesinde yapılırsa bir hizda burdens kazanırız.
- (*) Overfit'e gitmeden kısa sürede yüksek skorlar elde edilebilir bir modeldir.
- (*) Normal Gradient Boosting ile arasında mükemmel skor farkları yoktur ama ciddi bir süre farkı vardır. (Random Forest ile de çok büyük bir skor farkı yoktur.)

! Logistic Regression, XGBoost'tan daha hızlıdır.

XGBoost Hyperparameters

n_estimators: default=100

subsample: default = 1.0

"⁵
Büyük bir kısım veriyi kullan."

max_depth: default=3 (Ağac ne kadar derin olmalı?)

learning_rate: default=0.1

"Her nümerik ne kadar katkida bulunur?"

XGBoost

Faydalari

- Büyük datolarda hızlı.
- FDA'sız bile hızlı güzel sonuçlar.
- Feature Importance yapar.
- Model performansı çok iyi.
- Çok fazla hyperparametresi var. Bunlarla optimizasyon yaparak model iyileştirilebilir.
- Negatif yönler
- Görselleştirme yok.
(Çünkü açıklaması zor.)
- Çok fazla hyperparametresi olduğu için yönetmesi zor.
- Bir bozulunca diğer de bozulur.

Ozette ;

Bagging \Rightarrow Menda Random forest modelde ; ana data-dan subsample'lar üretir. Bunların her birine bir classifier veya Tree'ler oluştur. Feature'ların %inden de subsample'lar olur. Böylece hiçbir ağas birbirine benzemez. Böylece varyansı azaltıp overfittinge mücadele eder.

Boosting \Rightarrow Bütün data modelde girer. (Subsample yok.)

ilk model datayı manipüle eder, bir karar verir. Burdan sonra manipüle edilmiş data sonraki modele gider, ikinci üzeinden bir karar verir. Bu şekilde sona kadar gidiip bir prediction yapılır. Her modeldeki olusan ağırlıklara göre Meta Model karara gider.

UNSUPERVISED LEARNING

Train dataesi yok, target label yok. Bu tür data algoritmeye verilir ve kümeler ortaya çıkar.

① Clustering

② Dimensionality Reduction

Unsupervised learning, domain knowledge gerektirir.

Clustering

④ Customer Segmentation: Customer'lar gruplandırılır.
Sonra bu kümelere bakılarak hangi reklamlar verileceğine karar verilir.

⑤ Targeted Marketing

⑥ Recommender Systems

Dimensionality Reduction

3 boyuttan sonra görselleştirme yapılmaz. 3 feature'dan fazla feature'sı olan big dataları görselleştirmek, daha anlamlı hale getirebilmek için kullanılır.

Clustering

Benzer özelligi olan seylerin aynı grubu atarak kümelenme işlemi yapar.

Eğer cluster sayisi çok artırsak bir-birine çok benzeren çok fazla cluster'lar olur ve kümelerin hiçbirisi bir-birinden ayırt edilemez.

Cluster sayisi çok az olursa da alınması gereken bilgi kaçırılmış olur. Fazla genellemeye yapar.

Bu yüzden cluster sayisi çok önemlidir.

K-means Algorithm

$k \rightarrow$ kaç cluster olusacagini belirtir.

(Supervised learning'de konusluğunu belirtiyor du.)

K-means algoritmasının amacı; her bir kume içindeki kiler birbirine benzesseler ve her kume birinden farklı olsun.

K-means algoritması iterasyon mantığı ile çalışır.

(*) k sayısı kadar rastgele "Centroids" atanır.

(*) Bu centroids'ler her iterasyonda hareket ederek en iyi yer bulurlar ve çevredeki dekiler kendisi grubu olarak belirleyip kümelerler.

Centroids \rightarrow Cluster'ların merkezi.

! İlk nokta random atandığı için, istenen sonuçlar alınmazsa k-means algoritması birkaç defa çalıştırılabilir.

k-means algoritmasını random olarak yoluna başladıkten sonra çevresini hesaplaması ve duracıgı yeni bilmesi gereklidir. Bu da "Distance Function" ve "Optimization Criteria" ile yapar.

$$d(p, q) = \sum_{i=1}^n (q_i - p_i)^2 \Rightarrow \text{Distance Function (Çevre hesabı)}$$

$$\left[\sum_{i=0}^n \min_{M_j \in C} (\|x_i - M_j\|^2) \right] \Rightarrow \text{Optimization Criteria (Algıtıma ne zaman duracak?)} \quad \downarrow$$

Bölgelerin varyansının sıfırda en yakın olduğu noktada durur. (Çevresindekilerin uzaklığının karesi min olduğunda)

k Sayısının Tespiti

① Domain Knowledge

② Data Driven Approach
(Elbow Method)

Domain Knowledge: Datayı biler birisi kaç kümeye olduğunu söyley ve k sayısı ona göre belirlenir.

Elbow Method: Optimal k sayısını matematiksel hesaplarla bulur.

Clustering Evaluation

① Clustering Tendency (Hopkins Test)

② Optimal Number of Clusters (Elbow Method)

③ Clustering Quality (External Metrics (Domain Knowledge))

(Internal Metrics (No Domain Knowledge))

1. Clustering Test,

Hopkins Test'ı Bir datasetin cluster'ları ip cluster'ları arasında nasıl dağılıyor hakkında bilgi verir. İki hipotez oluşturur:

Null Hypothesis (H_0) : Data, non-random, uniform distribution'dır.

*	*	*	*	*
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*

(No meaningful clusters)

Alternative Hypothesis (H_a) : Data random dağılmıştır. (Presence of clusters)

```
{ from sklearn import datasets  
  from pycluster import hopkins  
  X = datasets.load_iris().data  
  hopkins(X, 150)}
```



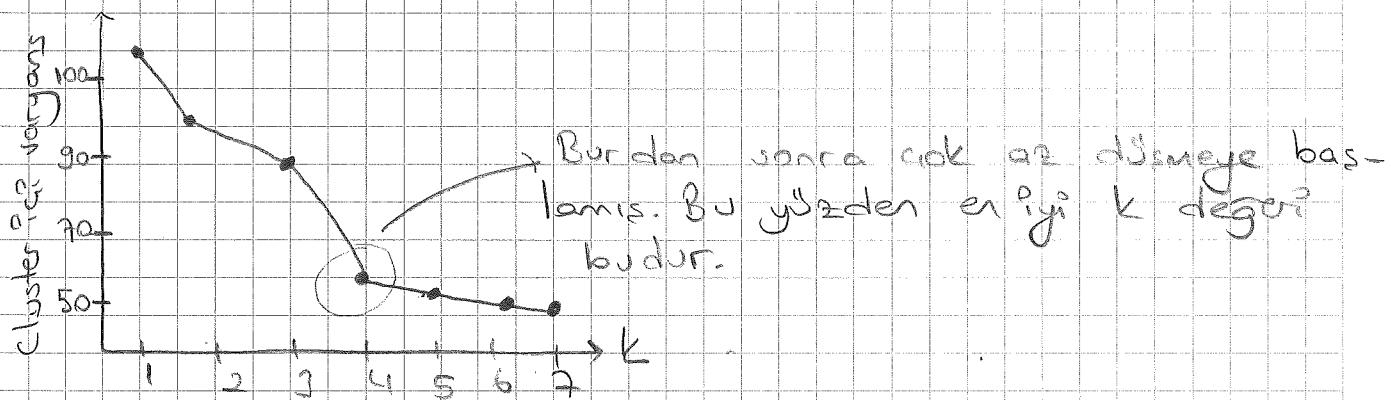
Burdan sikan değer 0.5^+ in üzerindeyse H_0 kabul.
 0.5^+ in altındaysa H_a kabul.

H_0 : Data random değil, clustering olmaz. (>0.5)

H_a : Data random, clustering tendency olabilir. (<0.5)

② Elbow Method

Domain Knowledge Pimkani yoksa bu metod kullanılır.



③ Clustering Quality

En iyi cluster \Rightarrow Cluster içindeki mesafe minimum,
Clusterlar arası mesafe max.

Bunun için 2 metrik kullanılır:

External Metrics: (Domain Knowledge)

* Adjusted Rand Index (Bir bunu kullanacaksınız.)

* Fowlkes - Mallows Index

* Jaccard Index / coefficient

Internal Metrics \circ (No Domain Knowledge)

- (*) Silhouette Coefficient (Bir kuru kullanacagiz.)
- (*) Davies - Bouldin Index
- (*) Dunn Index

→ Benzerligi Uzaklidan calisir

Adjusted Rand Index \circ

Cluster'larin birbirlerin ile benzerligi Uzaklidan bir sayi dondurur. (Kümeler birbirlerin ile benzer mi degil mi?)

Domain bilgisi olan birisi hepsi bilmese de datanin bir kismini label'lar. Bunun Sonunden cluster'larin birbirleri ile olan benzerligi octaya konur.



→ R2 score 0 veya negatif ise kötü clustering, 1'e yakin ise iyi clustering yapilmis demektir. Kume içi mesafe kisa, kümeler arası mesafe uzun demektir.

→ Mesafe ölçümde kullanılır.

Silhouette Coefficient, ?

Domin bilgisi olan bir yoksа bu metod kullanılır.

$$S = \frac{b - a}{\max(a, b)} \Rightarrow \begin{array}{l} \text{1'e yaklasirsa super,} \\ \text{-1'e yaklasirsa kötü.} \end{array}$$

a \Rightarrow Kime Pki ortalama mesafe

b \Rightarrow Diger cluster'lara olan ortalama mesafe

BSYÜK datalarında matris çok yüksek.

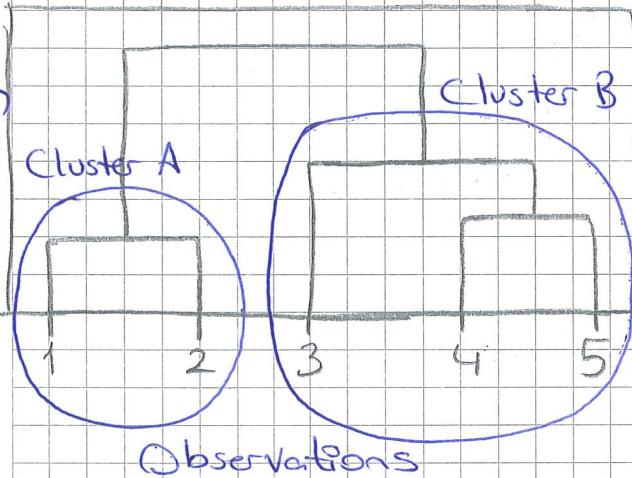
09.02.2022

Hierarchical Clustering Theory

K-means Algoritması gibi clustering için kullanılan bir algoritmadır. Önce bir dendrogram yapar, bu dendrogram üzerinde ne kadar cluster yapılacağını ortaya koyar.

Datanın ne kadar cluster'e ayrılacağı önceden belli olmaz. Dendrogram yapılır, buna göre karar verilir.

DENDROGRAM:



Similarity ne kadar azsa observation'lar birbirine o kadar yakındır. Kümeler arasındaki uzaklığın ne kadar olduğunu dendrogram üzerinde görebek kaç cluster olması gerektiğine karar verilir.

Den dogruların iki şekilde olusturulur :

Agglomerative
(En çok kullanılan)

Divisive

Agglomerative : Asagıdan yukarıya doğru çalışır. Önce bütün observationları birer cluster olarak kabul eder; yukarı doğru çıkararak en son datanın hepsini bir cluster olarak kabul eder.

Divisive : Yukarıdan aşağıya doğru çalışır. Tüm datayı alır, sağa doğru ikiye ayırarak gider. K-means algoritmasını kullanır. Çok kullanışlı degildir. Bunun yerine K-Mean algoritmasını kullanmak daha mantıklı.

K-means Algoritmasında olduğu gibi Hierarchical Clustering'de de amas :

Cluster içindeki mesafe min olsun.

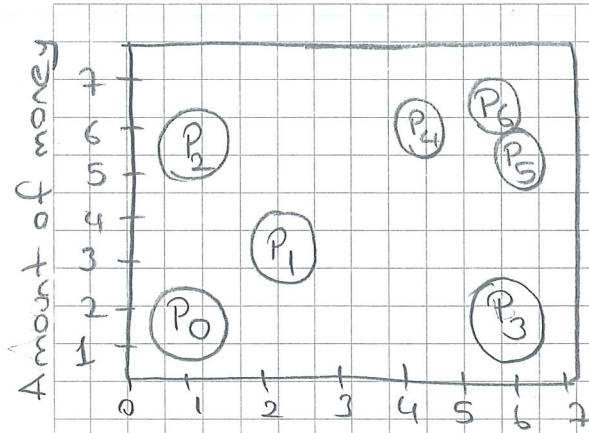
Clusterlar arasındaki mesafe max olsun.



Agglomerative



Divisive



Money mobility

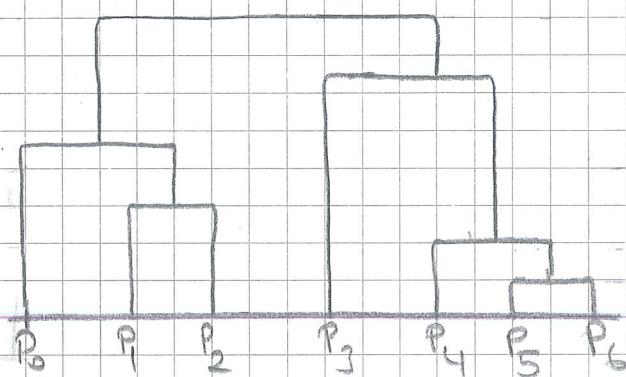
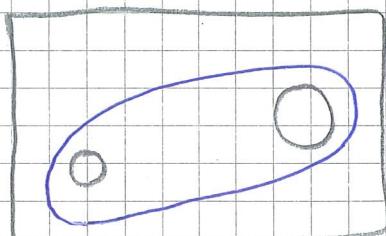
Dendogram, her bir data noktasını bir cluster olarak alır.

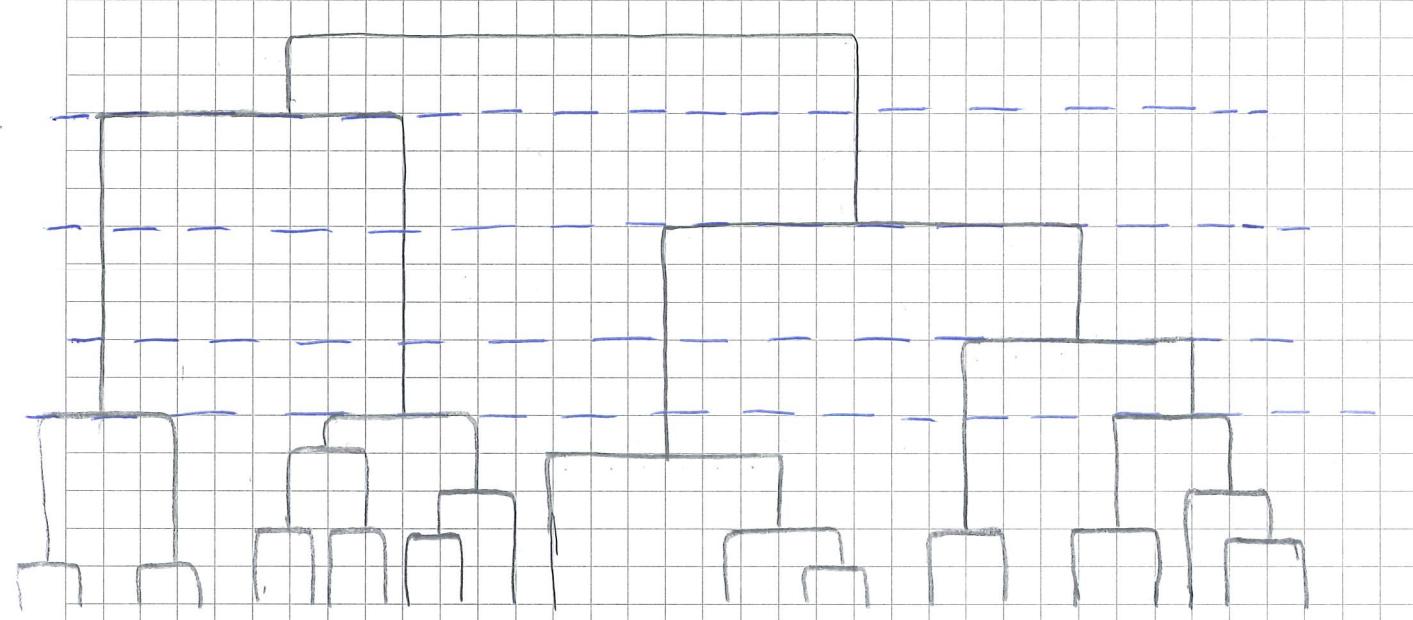
Sonra her birinin birbirlerine olan mesafelerini hesaplar. En yakın iki zanesinin orta noktasını bir cluster olarak alır. (P_5 ile P_6). Bu iki noktası yerine tek bir cluster alır.

Veinden işlem yapar. En yakındaki P_4 'ü alır ve orta noktasını bularak birleştirir. Sonra da P_3 'sini alır.

P_1 ve P_2 birbirine en yakın iki noktası olduğunu için bunları birleştirip ayrı bir cluster yapar. Sonra bunlara P_0 'ı katar.

En son asamada kalın son iki cluster'i birleştirir.

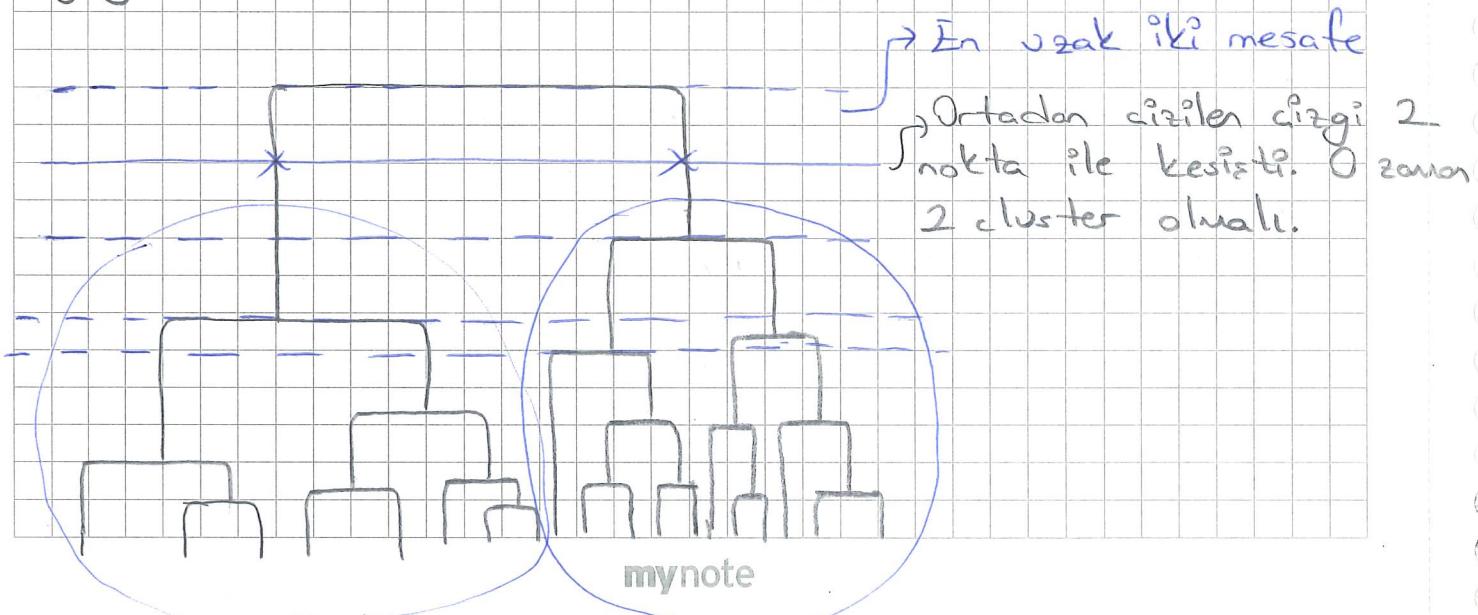




Cluster'lar arası mesafe yukarıdaki gibi belirlenir. Mesafe hangiinde daha büyük ise odağı cluster'lar birbirinden daha uzaktır.

Hangi iki çizgi arası mesafe en uzaksa, o iki çizgi arasındaki hatalı bir çizgi çizilir ve o çizgi üzerinde kesen dikey çizgiler sayılır. Bu sayı bize cluster sayısını verir.

Bu bilgi data'dan elde ettiğiniz bilgi, kesin doğruluğu yoktur. Bir zaman böyle bulduğumuzdan farklı bir sayı söyleyebilir. O zaman o kabul edilir.



Hierarchical Clustering'in k-means'den farklı
hicbir işlem yapmadan dendograma bakarak k sa-
yısını tahmin edebiliyor olmamızdır.

Hyperparameters :

sklearn.cluster.AgglomerativeClustering

① affinity (Default = "euclidean") :

Distance parametresidir.

{'euclidean', 'manhattan', 'cosine', 'precomputed'}

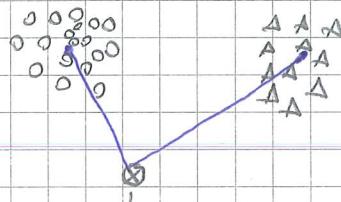
! 'euclidean' sadece 'ward' ile kullanılır.

② linkage (Default = "ward")

Datanın hangi cluster'a gideceğini belirler. (Cluster'lar arası mesafeleri ölçer.)

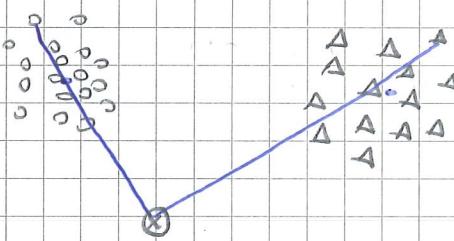
{'ward', 'complete', 'average', 'single'}

average :



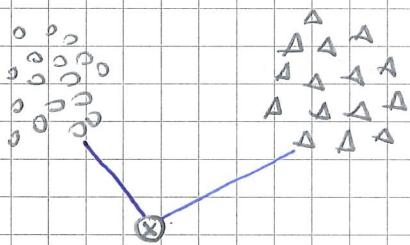
Bu data noktası hangi cluster'in mean'ine daha
yakınca ola atar.
mynote

complete :



Complete, cluster'ların en uzak noktalarını bulur.
Bunlar içinde en yakın olana atama yapar.

single :



Single, en yakın nokta hangisiyse ona atar.



Complete \Rightarrow Noise datalarda iyi çalışır fakat büyük cluster'lar olur.

Single \Rightarrow Grit datalarda kötü çalışır. (Complete'in aksine)

Ward : Kümeletin içinin varyansının en az olmasını sağlayacak şekilde çalışır. Dataya hangisine eklediğinde varyans az olacaksa ona ekler.

Principal Component Analysis (PCA)

The Curse of Dimensionality → Büyük datalarda boyut etme yolu.

- (*) Dataya eklenen her feature maliyet olarak gelir.
- (*) Feature sayısı arttıkça görselleştirme imkanı zorlaşır.
- (*) Data noktaları orttakca datanın eğilmesi zorlaşır.
- (*) Yani dataya eklenen her bir sütun, sonan, maliyet olarak bize gelir.

Bu sorunlardan kurtulmak için Dimension reduction tekniklerini uyguluyoruz.

- (*) FDA zorlukunu ortadan kaldırılmak için,
- (*) Datayı 2D ve 3D boyutta projeksiyon görselleştirmek için,
- (*) Sütun sayısı arttıkça; complexity, overfitting, multicollinearity sorunları ortaya çıkar. Bunları ortadan kaldırılmak için,

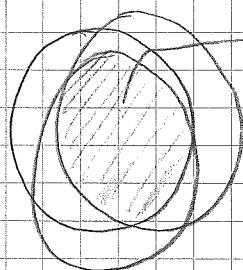
Dimension Reduction ile feature sayısı düşürüldüğünde bu sorunların hepsi ile mücadele edilir. Sütun sayısı az, satır sayısı fazla olduğunda modeller daha iyi öğrenir. Dimension Reduction yapmanın bir sebebi de budur.

sklearn.decomposition.PCA

Principal Components Analysis (PCA)

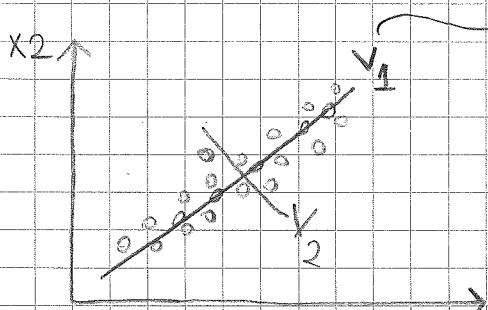
Dimension'ı düşürmek için kullanılan popüler bir tekniktir.

PCA; yüksek boyuttaki datalarda bir takım bilgilerden faydalayıarak, bir newi oluşturmak istediklerinde daha az feature (boyut) ile onları tanımlar.



→ 3 feature'lı bir datasetin sadece ortasındaki kisim alınca data, 1D boyuta iner. Bu tür bilginin sadece 96% alınılmış olur. (1 feature ile)

Teknik olarak PCA, "varyansı paylaşan değişken kümelerini" tanımlar ve bu varyansı temsil edecek bir değişim olusturur.



→ 2 feature'lı bir datasette V_1 , datasetin varyansını en iyi tanımlayan component olur. fakat V_2 'den uzakta kalan, tanımlanamayan bazı noktalar

X_1 katır. Daha fazla component istiyorsak bir de V_2 çiziniz, o ekseğindeki redüksiyonları alırız.

Amaçımız; en az feature ile en çok varyansı açıklamak.

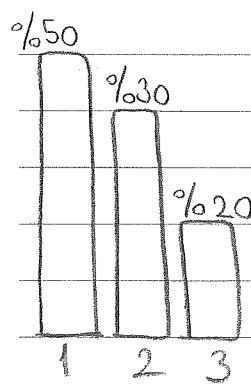
Olusan her bir componente Principal Component deñir.

Principal Component uyguladığınızda modele herhangi bir sınırla koymazsak, feature adedinde component oluşturur.

Bu componentlerde belki orantalar da total varyansı karşılar.

18 feature'lı bir datadan 18 component alınmışsa hiçbirinin birbir ile corr ilişkisi olmaz. Hepsinin birbiryle corr'u 0 olur. Bu componentler birer feature gibi davranışır.

→ Hani bir dataya PCA yapıldıktan sonra multicollinearity'den söz edilemez.



→ 3 feature'lı bir dataya PCA uygulandığında birbiryle corr ilişkisi olmayan 3 component olusur. (Her component içindeki feature'lar arası corr ilişkisi olabilir.)

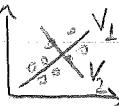
'Her component içinde her feature içindeki bilgi vardır.'

Steps of PCA

① Once data standardize edilir. (z score' a göre scale işlemi.) Böylece mean noktası bulunmuş olur.

② Mean noktası üzerinden her bir sample'in birbirle-
rince göre Covariance Matrix'ler ortaya çıkar.

③ Linear transformasyon ile Eigenvectors ve
Eigenvalues hesaplanır.



Eigenvector'den gelen değerlerde göre componentler sıralanır.

Eigenvector'den gelen değerler, toatal varyansın
kaçta kasını karşıladığı söyler.

Eğer bir sınırlı sayıda feature sayısı kadar component olusur.

④ Eger bir "k" sayısı veya "varyans" belirlenirse
o kadar component oluşturulur.

O zamanda $\%100$ varyans karşılanamaz. Seçilen component kadar varyans karşılanır.

! Görselleştirme için component sayısı 2 veya 3'e düşü-
lmek zorunda.

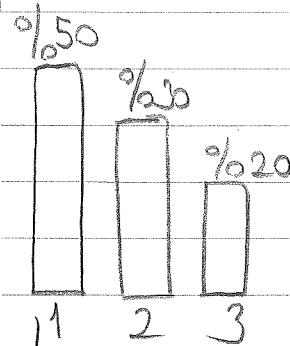
Eğer görselleştirme isteniyorsa, amaç overfitting veya
multicollinearity ile mücadele ise çok daha fazla compo-
nent seçilebilir.

① Standardization

② Covarian Matrix Computation

③ Compute Eigenvectors and Eigenvalues

④ Choose "k" eigenvectors with the largest eigenvalues.



Eigenvalue'su en yüksek vektör.

Her component içinde 3 feature bilgileri de belli oranlarında vardır.

PCA

Avantajları

(*) Feature'lar arası corr yok eder.

(*) Algoritmanın performansını artırır. (den bilgi var.)

(*) Overfitting'i azaltır.

(*) Görselleştirme yapılabilir.

Dezavantajları

(*) Yorumlaması zor.

(Her component'te her feature' den bilgi var.)

(*) Standardization gereklidir.
(Mean'e göre bir vektör çiziliyor ki covariance matrix' ten düzgün olun.)

(*) Bilgi kaybı.

(Feature'ların boyutunu düşürmek için bir takım bilgilerden vazgeçiliyor.)

PCA, total varyansın belli bir kısmını almayarak boyut düşürme istemidir.