"Regresyon" terimi, istatistik ve matematikte, bir değişkenin diğer bir veya daha fazla değişkenle ilişkisini modellemek için kullanılan bir tekniktir. Bu ilişkiyi açıklamak için kullanılan modeller, genellikle gözlemlenen verileri temsil eden matematiksel denklemlerdir. Regresyon analizi, değişkenler arasındaki ilişkinin doğasını anlamak, tahmin yapmak ve sonuçları yorumlamak için kullanılır.

"Regresyon" terimi, İngiliz istatistikçi Francis Galton'un çalışmalarına dayanır. Galton, ebeveynlerin boy uzunluğunun çocukların boy uzunluğunu etkilediğini göstermek için çalışmıştır. Galton, çocukların boy uzunluğunun ebeveynlerin boy uzunluğunun ortalamasına "geri dönme eğilimi" gösterdiğini gözlemledi. Bu "geri dönme" veya "regresyon" terimi, Galton'un çalışmalarıyla popülerlik kazandı ve regresyon analizinin temelini oluşturdu.

Doğrusal regresyon, istatistiksel bir modelleme tekniğidir ve bir bağımlı değişken ile bir veya daha fazla bağımsız değişken arasındaki ilişkiyi açıklamak için kullanılır. Bu ilişkiyi bir doğru ile ifade eder. Genellikle, bir bağımsız değişkenin bağımlı değişken üzerindeki etkisini anlamak veya bir bağımlı değişkenin tahminini yapmak için kullanılır.

Doğrusal regresyon modeli, veri setindeki gözlemleri en iyi şekilde temsil eden bir doğruyu bulmaya çalışır. Bu doğru, bağımsız değişkenlerin katsayılarını (eğimleri) ve sabit bir terimi (kesme noktası) içerir. Model, bu katsayıları veri setine uygun olarak hesaplar.

Doğrusal regresyon, temelde iki tür veri arasındaki ilişkiyi anlamak için kullanılır. Örneğin, reklam harcamaları ile satışlar arasındaki ilişkiyi belirlemek veya bir kişinin yaşını ve kilosunu kullanarak sağlık durumunu tahmin etmek gibi durumlarda kullanılabilir.

MULTİPLE LİNEAR REGRESSİON

Çoklu Doğrusal Regresyon (Multiple Linear Regression)

Basit Doğrusal Regresyon

$$y = \alpha + \beta x,$$

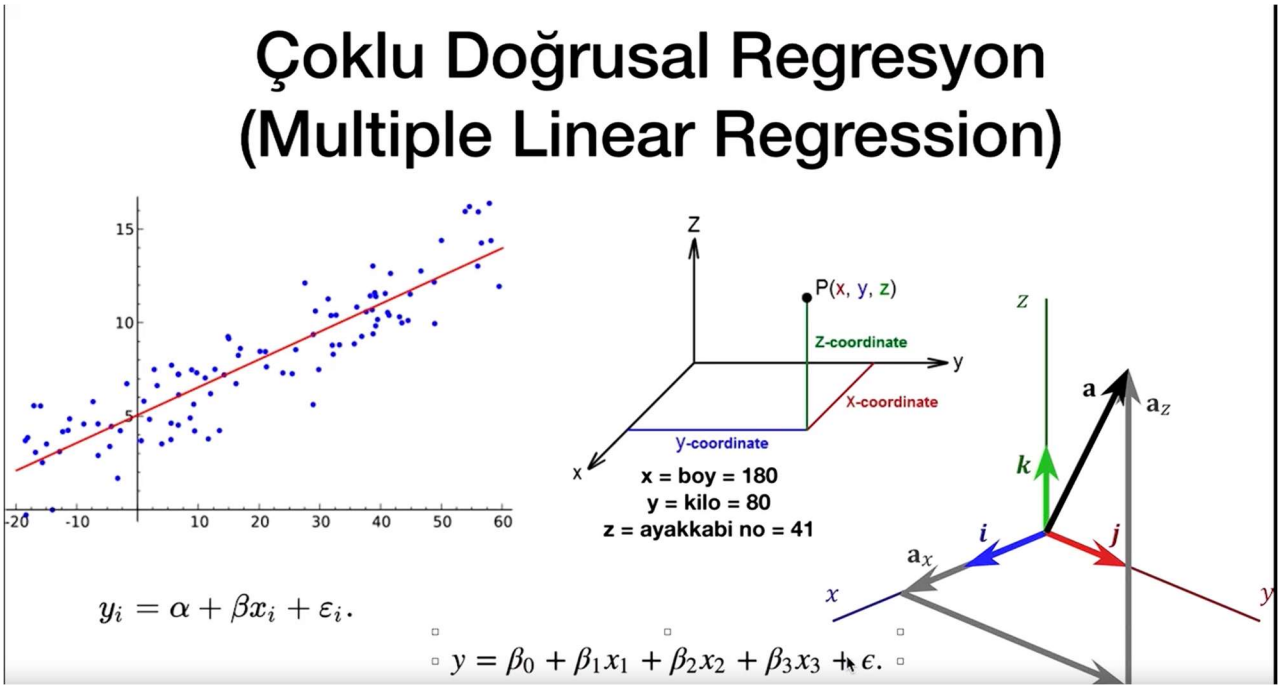
$$y_i = \alpha + \beta x_i + \varepsilon_i.$$

Satış = a + b (Ay) + e

Çoklu Doğrusal Regresyon

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon.$$

Boy = a + b (kilo) + c (yaş) + d (ayakkabı no) + e



Kukla Değişken (Dummy Variable)

Boy	Kilo	Yaş	Cinsiyet
130	30	10	e
129	38	12	e
135	34	10	k
133	30	9	k
175	90	35	E

Kukla Değişken Tuzağı
Dummy Variable Trap

E	K
1	0
1	0
0	1
0	1
1	0

dummy değişken tuzağına dikkat edilmeli; burada cinsiyet kolonu ile E-K [encoding işlemi uygulanmış cinsiyet kolonu] ayni aynda tabloda bulundurulursa öğrenme algoritmalari için verimli olmayabilir.

Aynı zamanda dummy değişkenin kendi içinde de bazı elemeler yapabilir;
buradaki örnekte dummy değişken binominal [iki ihtimalden biri] olduğu için sadece bir kolonu almamız yeterli olacaktır [kadın değilse erkektir]

Fakat bazı durumlarda dummy veriable ülke gibi degiskenleri içerisinde tutacaksa 2den fazla ülke olabilir : örn[tr,us,fr] o zaman kukla verinin tamamını tabloya eklemeliyiz.

---

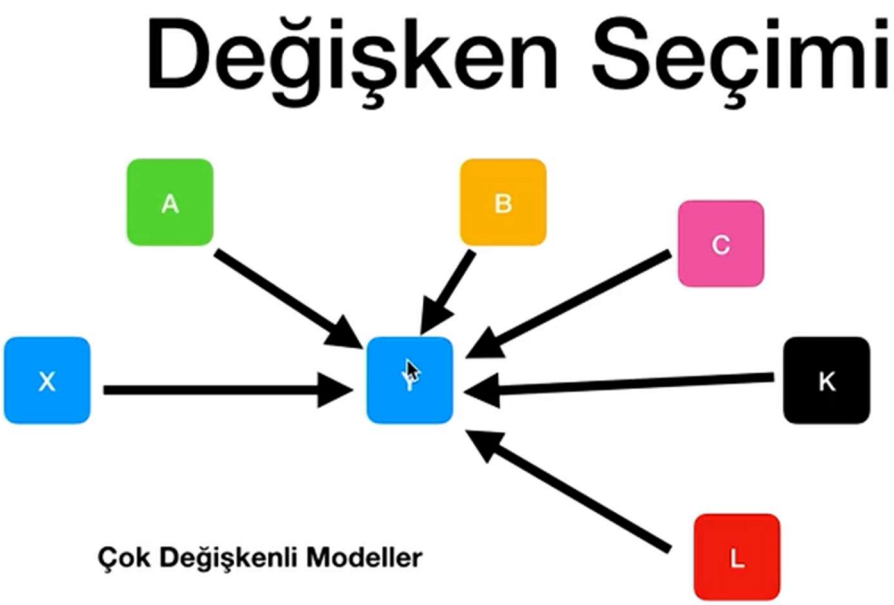
p-value (olasılık değeri)

- H0: null hypothesis : Farksızlık hipotezi, sıfır hipotezi, boş hipotez
- H1: Alternatif hipotez
- p-değeri: olasılık değeri (genelde 0.05 alınır)
- P-değeri küçüldükçe H0 hatalı olma ihtimali artar
- P-değeri büyüdükçe H1 hatalı olma ihtimali artar

H0:null hypothesis: her kutuda 100 tane kurabiye vardır

H1:alternatif: her kutuda 100 tane kurabiye yoktur

p-degeri : genelde 0.05 bu ornekte de 0.05 alındıgında 5 kutuda 100 tane kurabiye yoksa hipotezin yanlış olma olasılıgı artar. H1'in doğru olma olasılıgı artar



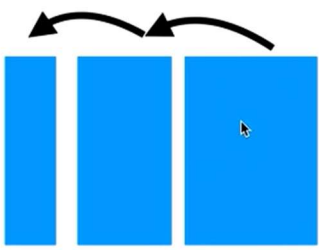
Burada önemli olan hangi bağımsız değişken bağımlı değişkeni daha fazla,daha az veya hic etkilemiyor bunun analizi.

Hangi bağımsız değişkenleri seçmeliyiz ?

Bütün Değişkenler

- Şayet değişken seçimi (selection) yapıldıysa ve değişkenlerden eminsek
- Zorunluluk varsa (örn. Bankadaki kredi skorları için geliştirilen modelin başarısının ölçülmesi)
- Keşif için (diğer 4 yöntemi kullanmadan önce bir ön fikir elde etmek için)

Geriye Eleme (Backward Elimination)



1.Significance Level (SL) seçilir (genelde 0.05)

2.Bütün değişkenler kullanılarak bir model inşa edilir.

3.En yüksek p-value değerine sahip olan değişken ele alınır ve şayet P>SL ise 4. adıma, değilse son adıma (6. adım) gidilir

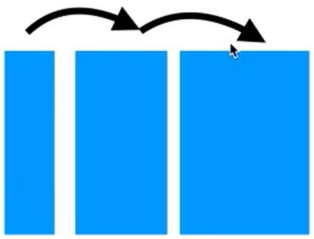
4.Bu aşamada, 3. adımda seçilen ve en yüksek p-değerine sahip değişken sistemden kaldırılır

5.Makine öğrenmesi güncellenir ve 3. adıma geri dönülür.

6.Makine öğrenmesi sonlandırılır.

[literatürde sıkça kullanılır] ilk bütün değişkenler alanır eleye eleye gidilir

İleriye Seçim (Forward Selection)



1.Significance Level (SL) seçilir (genelde 0.05)

2.Bütün değişkenler kullanılarak bir model inşa edilir.

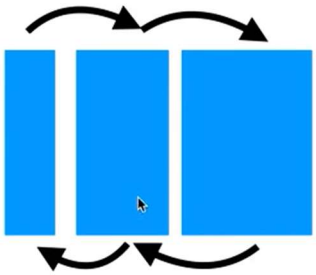
3.En **düşük** p-value değerine sahip olan değişken ele alınır

4.Bu aşamada, 3. adımda seçilen değişken sabit tutularak yeni bir değişken daha seçilir ve sisteme eklenir

5.Makine öğrenmesi güncellenir ve 3. adıma geri dönülür, şayet en düşük p-değere sahip değişken için p<SL şartı sağlanıyorsa 3. Adıma dönülür. Sağlanmıyorsa biter (6. Adıma geçilir)

6.Makine öğrenmesi sonlandırılır.

Çift Yönlü Eleme (Bidirecitonal Elimination)



1.Significance Level (SL) seçilir (genelde 0.05)

2.Bütün değişkenler kullanılarak bir model inşa edilir.

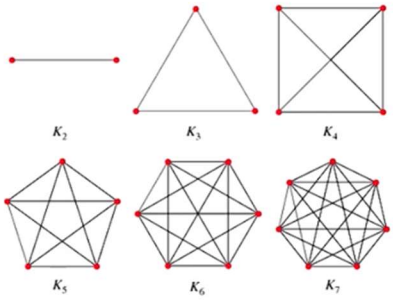
3.En **düşük** p-value değerine sahip olan değişken ele alınır

4.Bu aşamada, 3. adımda seçilen değişken sabit tutularak diğer bütün değişkenler sisteme dahil edilir ve en düşük p değerine sahip olan sistem de kalır

5.SL değerinin altında olan değişkenler sistemde kalır ve eski değişkenlerden hiçbirisi sistemden çıkarılamaz.

6.Makine öğrenmesi sonlandırılır.

Bütün Yöntemler



1.Başarı kriteri belirlenir.

2.Bütün olası regresyon modelleri inşa edilir (ikili seçim olur)

3.Başta belirlenen kriteri (1. adım) en iyi sağlayan yöntem seçilir

4.Makine öğrenmesi sonlandırılır.

Geriye eleme(backward elimination)

```python
# Geri Eleme (Backward Elimination)

import numpy as np
import statsmodels.api as sm # stat-> istatistik

X = np.append(arr=np.ones((22,1)).astype(int),  values=data, axis=1) # axis = 1 -> kolon olarak eklemek
# burada birlerden oluşan bir kolon ekliyoruz, denklemdeki çarpan değeri 1[etkisiz eleman]
#  y=b0 +b1x1 +b2x2.......+e.     aslında b0'i ekliyoruz

X_list = data.iloc[:,[0,1,2,3,4,5]].values # list olarak tanımlıyoruz ki
# analiz yapıldıgında hangisinin p degeri yuksek görelim ve onu çikaralim
X_list = np.array(X_list,dtype=float) # np arraya dönüstürmek

model = sm.OLS(genderDF.iloc[:,0:1],X_list).fit()
print(model.summary())
# bazı p degerleri 0.05'in üstünde çıktı öncelikle en buyuk cıkanı çıkartıp tekrar deneme yanılma
# yöntemiyle devam edilebilir, p degerleri dustukce sonuca yaklasiyo sayilabiliriz ;

# X_list = data.iloc[:,[3,4,]].values gibi
```

```
Console 1/A ×

No. Observations:                22   AIC:                            22.29
Df Residuals:                    16   BIC:                            28.83
Df Model:                         5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             2.2338      1.174      1.903      0.075      -0.254       4.722
x2             2.2461      1.075      2.089      0.053      -0.033       4.525
x3             1.8514      1.122      1.651      0.118      -0.526       4.229
x4            -0.0204      0.010     -2.098      0.052      -0.041       0.000
x5             0.0308      0.008      3.682      0.002       0.013       0.048
x6            -0.0077      0.010     -0.813      0.428      -0.028       0.012
==============================================================================
Omnibus:                        0.140   Durbin-Watson:                   1.516
Prob(Omnibus):                  0.932   Jarque-Bera (JB):                0.212
Skew:                          -0.158   Prob(JB):                        0.899
Kurtosis:                       2.637   Cond. No.                     4.53e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified
[2] The condition number is large, 4.53e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [55]:
```

POLYNOMİAL REGRESSION

Polinomal Regresyon
(Polynomial Regression)

Çoklu Doğrusal Regresyon

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon.$$

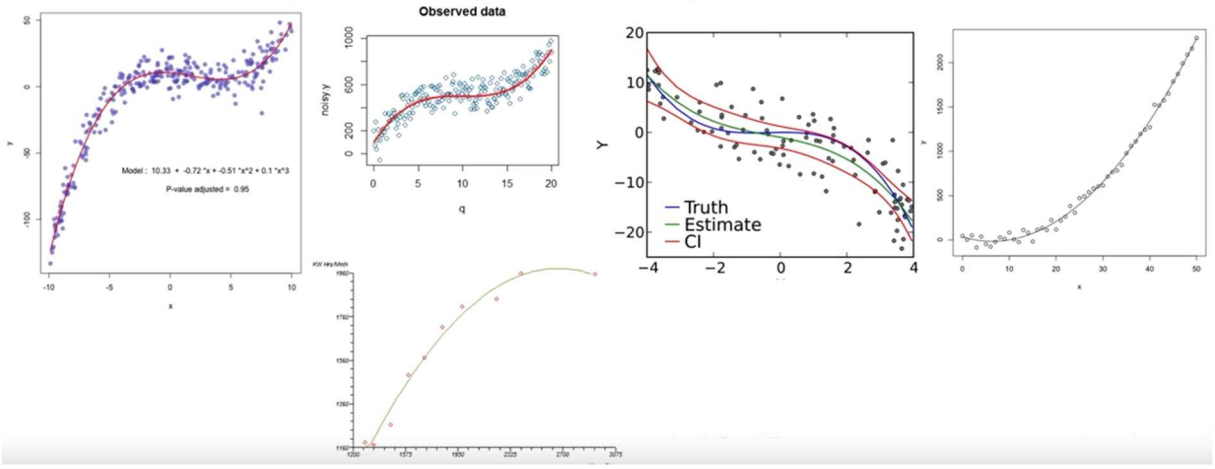
Polinomal Regresyon

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \ldots + \beta_h X^h + \epsilon,$$

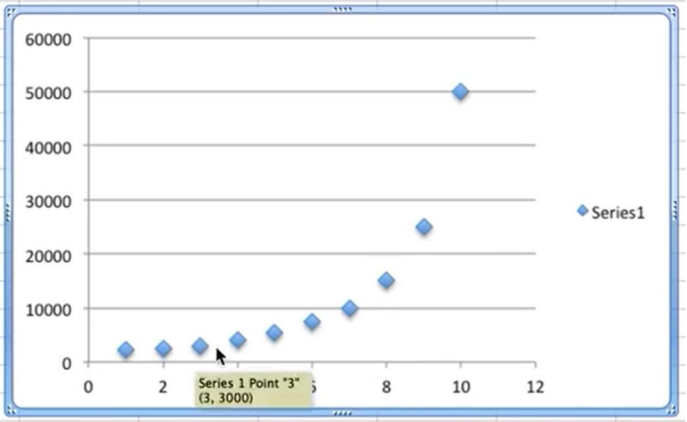
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{12} x_1 x_2 + \varepsilon$$

Boy = a + b (kilo) + c (yaş) + d (ayakkabı no) + e

Polinomal Regresyon
(Polinomial Regression)



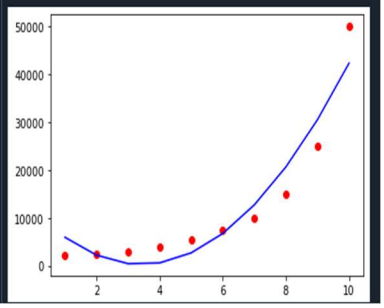
	A	B	C
1	unvan	Egitim Seviyesi	maas
2	Cayci	1	2250
3	Sekreter	2	2500
4	Uzman Yardi	3	3000
5	Uzman	4	4000
6	Proje Yoneti	5	5500
7	Sef	6	7500
8	Mudur	7	10000
9	Direktor	8	15000
10	C-level	9	25000
11	CEO	10	50000



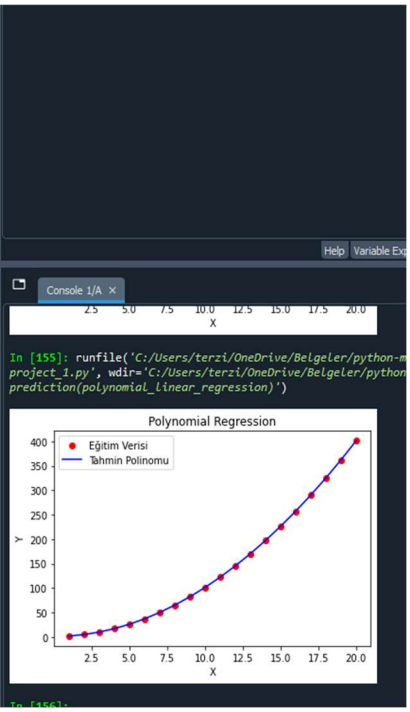

```python
# linear regression
lin_reg = LinearRegression().fit(X,Y)

plt.scatter(X,Y, color='red')
plt.plot(X,lin_reg.predict(X),color='blue')
plt.show()

# polynomial regression
from sklearn.preprocessing import PolynomialFeatures

poly_reg = PolynomialFeatures(degree = 2) #2. dereceden polynomial obje olustur
x_poly = poly_reg.fit_transform(X) # X değerini polinomal dünyaya çevirdi
print(x_poly)

lin_reg2 = lin_reg = LinearRegression().fit(x_poly,Y) # 3 tane kolonun carpanlarını öğreniyo
                                        #x^0 x^1 x^2

plt.scatter(X,Y, color='red')
plt.plot(X,lin_reg2.predict(poly_reg.fit_transform(X)),color='blue')
```

```
[[ 1.   1.   1.]
 [ 1.   2.   4.]
 [ 1.   3.   9.]
 [ 1.   4.  16.]
 [ 1.   5.  25.]
 [ 1.   6.  36.]
 [ 1.   7.  49.]
 [ 1.   8.  64.]
 [ 1.   9.  81.]
 [ 1.  10. 100.]]
```

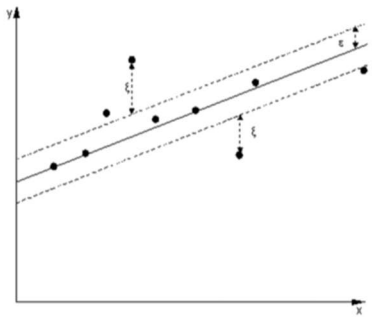
print(x_poly)



```python
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt

data = pd.read_csv('data1.csv') # x^2 + 1 fonksiyonu    - > 1-> 2 , 2->5 , 3->10

x = data[['x']]
y = data[['y']]
X = x.values
Y = y.values


poly_reg = PolynomialFeatures(degree=2)
x_poly = poly_reg.fit_transform(X)

lin_reg = LinearRegression().fit(x_poly, y)

plt.scatter(X, Y, color='red', label='Eğitim Verisi')
plt.plot(X, lin_reg.predict(poly_reg.fit_transform(X)), color='blue',label='Tahmin Polinomu')

plt.xlabel('X')
plt.ylabel('Y')
plt.title('Polynomial Regression')
plt.legend()
plt.show()
```



Destek Vektör Regresyonu (Support Vector Regression)



$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot \langle x_i, x \rangle + b$$ **Doğrusal SVR**

$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot \langle \varphi(x_i), \varphi(x) \rangle + b$$ **Doğrusal olmayan SVR (non-linear SVR)**

$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot K(x_i, x) + b$$

Polynomial

$$k(x_i, x_j) = (x_i . x_j)^d$$

Gaussian Radial Basis function

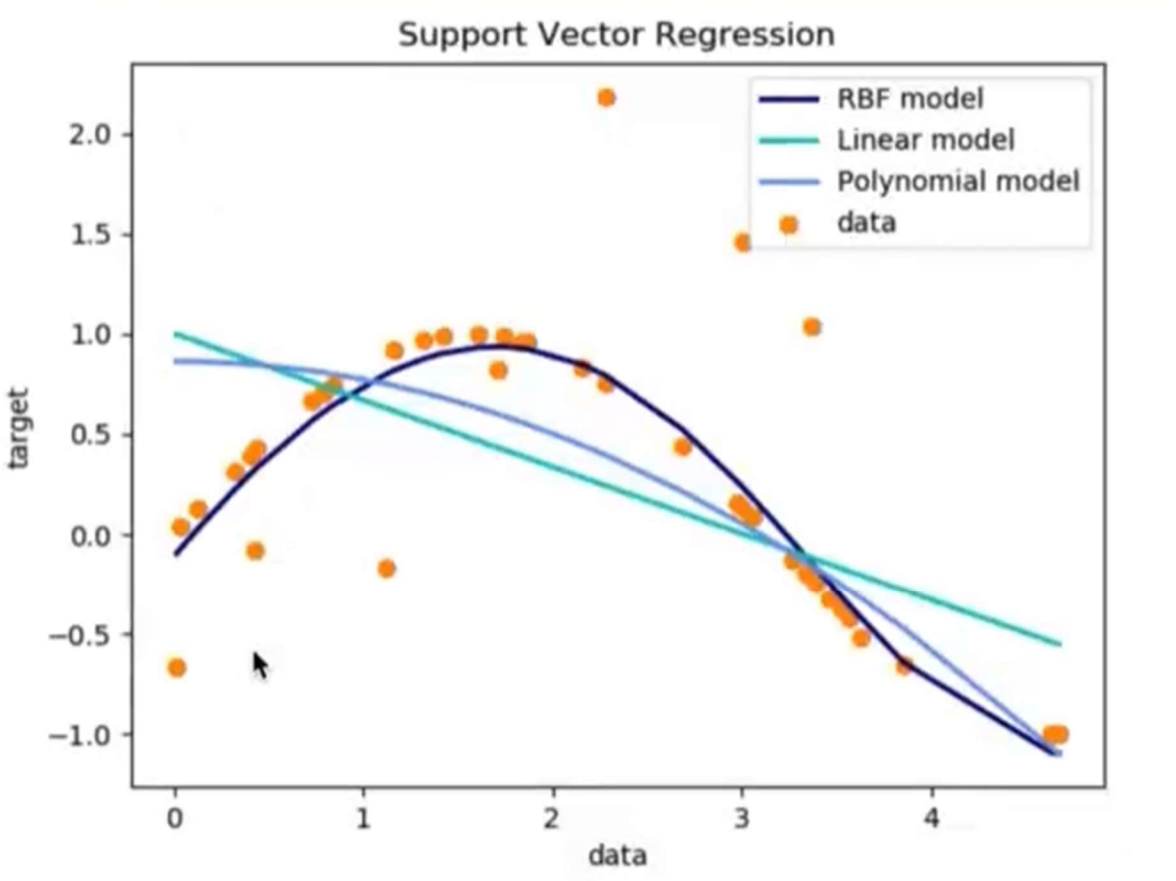
$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Support vector regression margin değerlerini tanımlıyor ve bu margin değerlerine giren maksimum noktayi elde edebileceği minimum margin değerine sahip fonksiyonu almayı amaçlıyor

Birden fazla doğru çizilebiliyosa minimum margin değerine sahip ayni noktalari içine alabiliecek değeri elde etmeye çalışıyor

Farkli fonksiyonlar kullanılabilir;

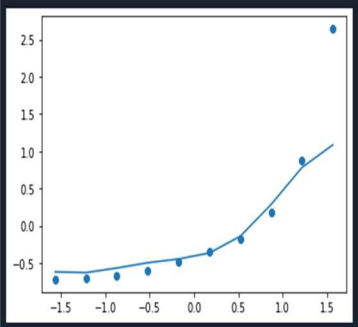
Doğrusal , doğrusal olmayan[polynomial, rbf(radial basis function)] , exp




```
svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)
svr_lin = SVR(kernel='linear', C=1e3)
svr_poly = SVR(kernel='poly', C=1e3, degree=2)
```

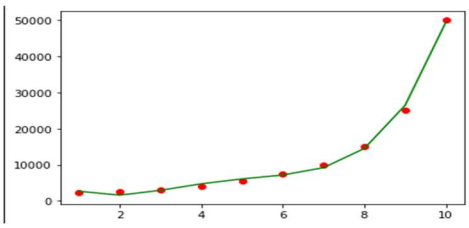
```
# support vector regression scaler ile kullanmamız gerekli veriler uzerindeki marjinal verilere duyarlı

x_sc = StandardScaler().fit_transform(x)
y_sc = StandardScaler().fit_transform(y)
plt.show()

# SVR        svm -> support vector machine
from sklearn.svm import SVR

svr_reg = SVR(kernel="rbf")
svr_reg.fit(x_sc , y_sc)

plt.scatter(x_sc , y_sc)
plt.plot(x_sc, svr_reg.predict(x_sc))
plt.show()
```

```
was passed when a 1d array was expected. Please change the
  y = column_or_1d(y, warn=True)
```



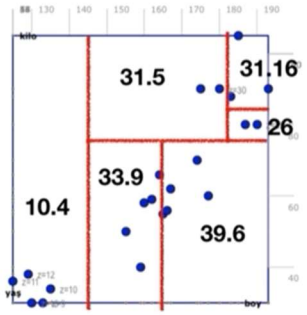

```
In [31]:
```

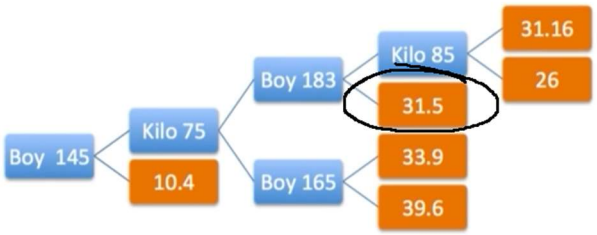
(rbf)



(polynomial regression)

Karar Ağacı ile Tahmin (Decision Tree)



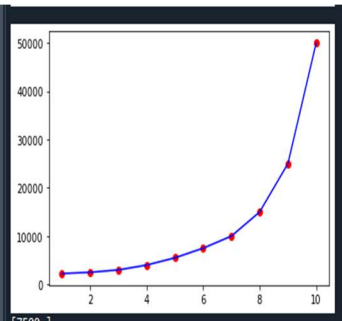


örnek, boy = 180, kilo = 80

Karar ağaçları genelde sınıflandırma için kullanılır fakat tahmin için de kullanılabilir.

Hassaslığı arttırmak için karar ağacını daha fazla dallandırmalıyız

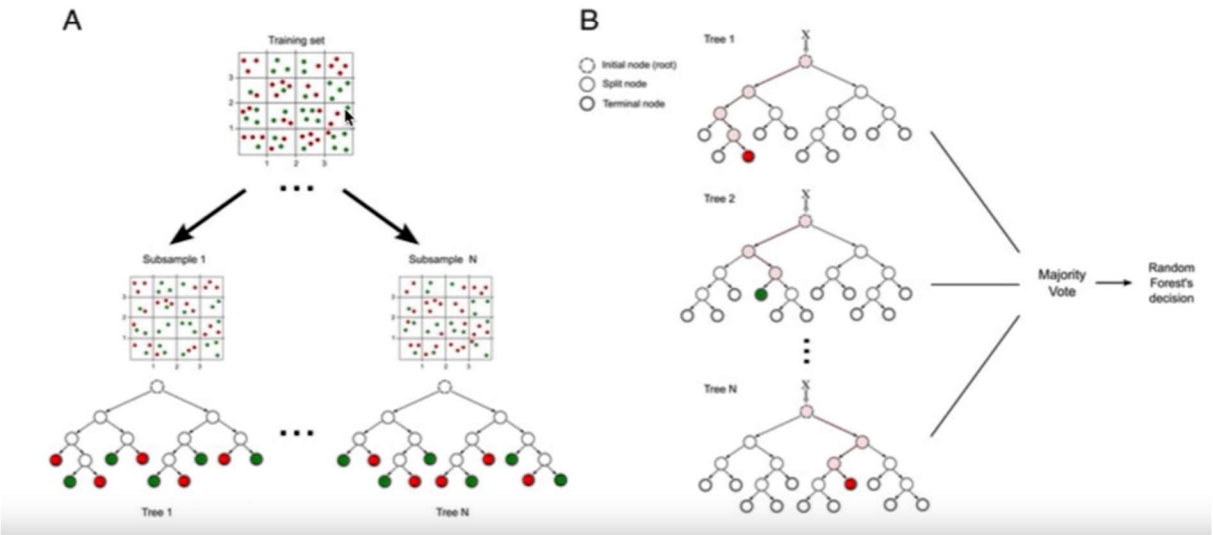
```
# decision tree
from sklearn.tree import DecisionTreeRegressor

dt_reg = DecisionTreeRegressor(random_state=0)

dt_reg.fit(x, y)

plt.scatter(x,y,color='red')
plt.plot(x,dt_reg.predict(x),color='blue')
plt.show()

print(dt_reg.predict([[5.6]]))
print(dt_reg.predict([[6]]))
print(dt_reg.predict([[6.5]]))
```




```
[7500.]
[7500.]
[7500.]
```

5.6 – 6 – 6.5 degerlerini aynı değerde tahmin etti

Rassal Orman



Rassal ağaç (random forest) : birden fazla decision tree'nin ayni problem aynı veri kümesi üzerinde çizilmesi ve daha sonra problemin çözümünde hep birlikte kullanılmasına dayanıyor.

Olukça güçlü bir yöntemdir

Bu duruma Majority voted learning - > çoğunluğun oyu ismi verilir

Veri kümesini küçük parçalara bölüyor birden fazla decision tree'yi öğreniyor

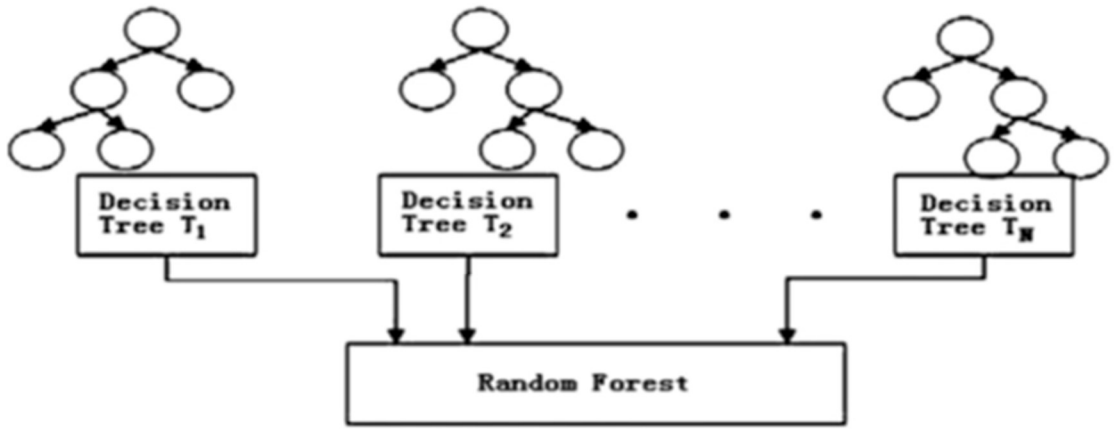
Bu şekilde birden fazla tahmin algoritmasi gelişiyor ve çoğunluğun oyuna(majority voted) göre *sınıflandırma* yapılıyor

Tahminde ise verilen kararlarin (majority voted) ortalaması sonuç oluyor

Karar ağaçlarinda verinin artması durumunda başarının düşmesi ile ilgili bir sonuç var

Bazı durumlarda verinin artması sonucunda karar ağacına etkisi ;

- Sonuçların yanlış çıkması
- Overfitting (aşırı öğrenme) -> ezberlemeye gidebilir
- Ağacın çok fazla dallanması durumunda hesaplama zamanı sıkıntısı çıkartabilir

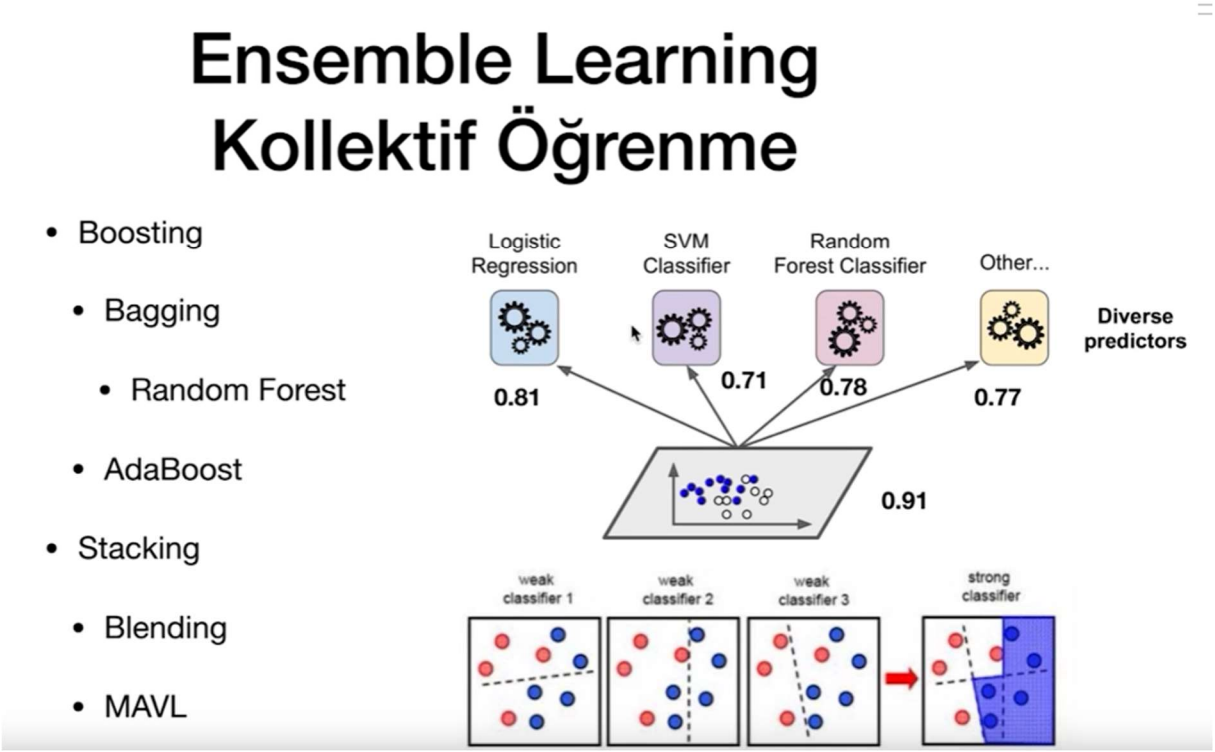


Ensemble Learning (Kollektif Öğrenme)

Birden fazla sınıflandırma veya birden fazla tahmin algoritmasi aynı anda kullanılarak daha basarili bir sonuç çıkartılabilir

Buna Ensemble Learning (kollektif öğrenme) denir

Random forest ile decision tree'nin ayni anda kullanılması ensemble learning'e örnektir



Tahmin Algoritmalarının Değerlendirilmesi (Evaluation of Predictions):

R2 Hesaplama;

- R2 (R- Square , R-Kare) Yöntemi

TOPLA (gerçek − tahimn)^2

Hata Karaleri Toplamı = Topla(y_i - y'_i)²

TOPLA (gerçek − tahminORT)^2

Ortalama Farkların Toplamı = Topla(y_i - y_{ort})²

$$R^2 = 1 - \frac{HKT}{OFT}$$

OFT = Olabilecek En Kötü Tahmin

R2 değeri 1 'e yaklaştıkça iyi sonuç 0 'a yaklaştıkça kötü sonuç negatif değerler çok kötü sonuç

boy	kilo	yas	tahmin	başarı	=(C2-D2)*(C2-D2)	=(C2-D$24)*(C2-D$24)
130	30	10	12	10	4	355.836777
125	36	11	11	11	0	319.109504
135	34	10	10	10	0	355.836777
133	30	9	10	9	1	394.56405
129	38	12	11	12	1	284.382231
180	90	30	35	30	25	1.29132231
190	80	25	26	25	1	14.927686
175	90	35	47	35	144	37.6549587
177	60	22	22	22	0	47.1095041
185	105	33	33	33	0	17.1095041
165	55	27	27	27	0	3.4731405
155	50	44	44	44	0	229.109504
160	58	28.5	27	28.5	2.25	0.1322314
162	59	41	40	41	1	147.291322
167	62	55	50	55	25	683.109504
174	70	47	47	47	0	328.927686
193	90	28.5	29	28.5	0.25	0.1322314
187	80	27	22	27	25	3.4731405
183	88	28	33	28	25	0.74586777
159	40	29	22	29	49	0.01859504
164	66	32	33	32	1	9.83677686
166	56	42	44	42	4	172.56405
			28.8636364		308.5	3406.63636	0.90944146

⇩

308.5	3406.63636	=1-F24/G24

Düzeltilmiş R2 (Adjusted R2) Yöntemi:

R2 üzerinden bir model değerlendirilirken eklenen yeni değişken sisteme asla *negatif* etki yapmaz, R2 değerini *asla düşürmez* bu sebepten dolayı R2 bazı durumlarda *yetersiz ve yanıltıcı* kalabilir

Yöntemleri Karşılaştırmak

- R^2 (R- Square , R-Kare) Yöntemi
- Düzeltilmiş R^2 (Adjusted R^2) Yöntemi

Doğrusal Regresyon

$$y=a_0x_0+a_1x_1+b$$

$$y=a_0x_0+a_1x_1+a_2x_2+b$$

Hata Karaleri Toplamı = $\text{Topla}(y_i - y'_i)^2$

Ortalama Farkların Toplamı = $\text{Topla}(y_i - y_{ort})^2$

$$R^2 = 1 - \frac{HKT}{OFT}$$

n = eleman sayısı
p = değişken sayısı

$$\text{Düzeltilmiş } R^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

```
>>> from sklearn.metrics import r2_score
>>> y_true = [3, -0.5, 2, 7]
>>> y_pred = [2.5, 0.0, 2, 8]
>>> r2_score(y_true, y_pred)
0.948...
```

-Desicion tree regresyon modelinde R2 SCORE kullanmak yanıltıcı olabilir.