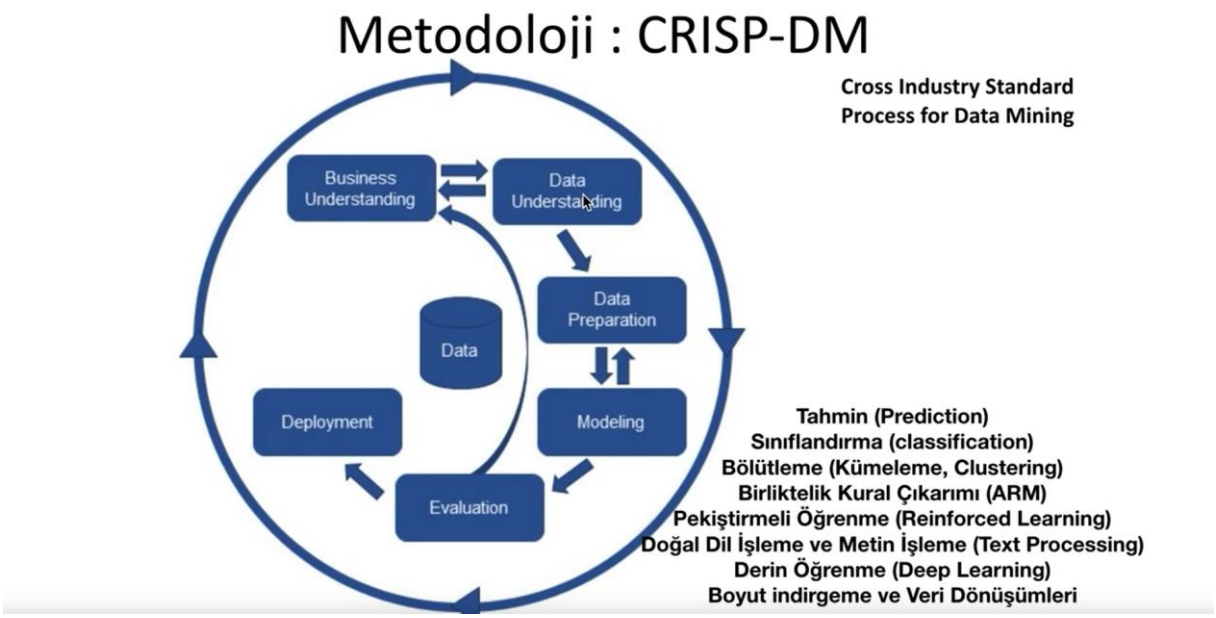
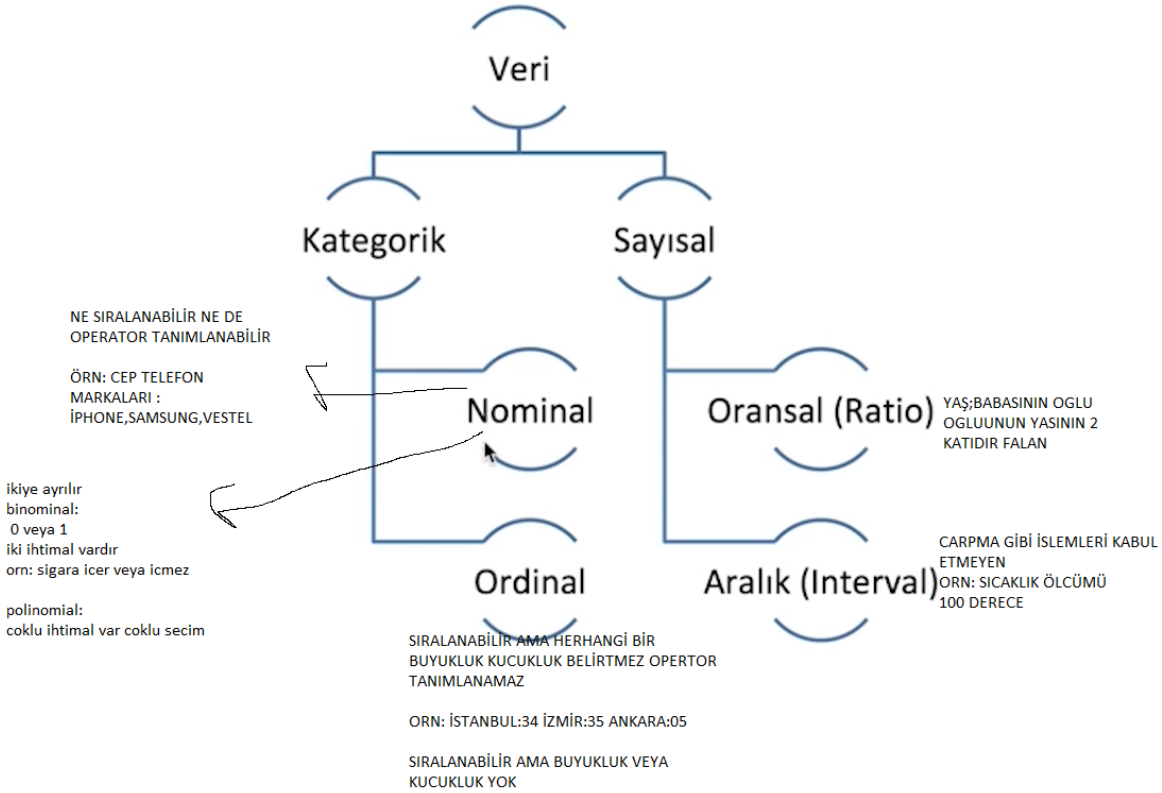
VERİ ÖNİŞLEME[DATA PREPROCESSING]



öğrenilen kısım : data preparation



A	B	C	D	E
ulke	boy	kilo	yas	cinsiyet
tr	130	30	10	e
tr	125	36	11	e
tr	135	34	10	k
tr	133	30	9	k
tr	129	38	12	e
tr	180	90	30	e
tr	190	80	25	e
tr	175	90	35	e
tr	177	60	22	k
us	185	105	33	e
us	165	55	27	k
us	155	50	44	k
us	160	58		k
us	162	59	41	k
us	167	62	55	k
fr	174	70	47	e
fr	193	90		e
fr	187	80	27	e
fr	183	88	28	e
fr	159	40	29	k
fr	164	66	32	k
fr	166	56	42	k
			28.45	

Buradaki ülke bilgilerini makinenin anlayacağı sayısal değerlere çevirmemiz gerekiyor; encode işlemi;

	TR	FR	US
TR	1	0	0
FR	0	1	0
US	0	0	1
TR	1	0	0
TR	1	0	0
FR	0	1	0
FR	0	1	0
US	0	0	1
US	0	0	1
TR	1	0	0
US	0	0	1
TR	1	0	0

Yukarıda yapılan islem one-hot encoding (encode-> kategorik verilerin sayısala dönüstürmek)

"Encode" terimi yapay zeka dünyasında genellikle kategorik verilerin sayısal değerlere dönüştürülmesini ifade eder. Kategorik veriler, genellikle metin tabanlı veya sembolik değerlerdir ve makine öğrenimi modelleri tarafından doğrudan işlenemezler. Bu nedenle, kategorik verilerin sayısal değerlere dönüştürülmesi, modelin bu verileri işleyebilmesi için önemlidir.

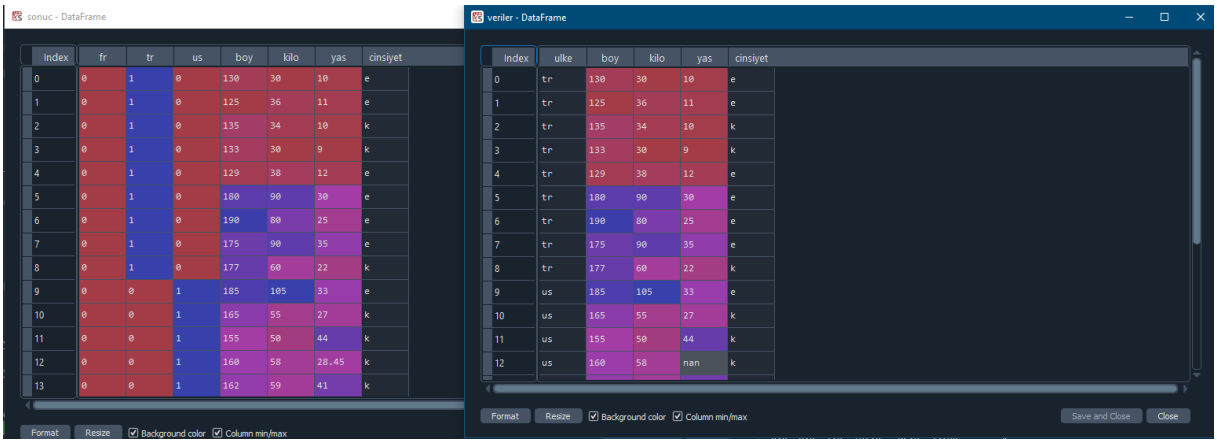
"Encode" işlemi, kategorik verilerin sayısal değerlere dönüştürülmesini sağlayan bir dönüşüm işlemidir. Örneğin, cinsiyet gibi bir kategorik değişkeni kodlamak için "One-Hot Encoding" veya "Label Encoding" gibi teknikler kullanılabilir.

- **One-Hot Encoding:** Kategorik değişkenlerin her bir farklı kategorisi için yeni bir sütun oluşturulur ve ilgili kategoriye ait olan gözlemler için bu sütunlar 1 ile kodlanırken, diğerleri 0 ile kodlanır.
- **Label Encoding:** Kategorik değişkenler, her bir kategoriye benzersiz bir sayısal değerle eşlenir. Her kategoriye farklı bir sayı atanır ve bu sayılarla değişken kodlanır.

```
ulke = veriler.iloc[:,0:1].values
print(ulke)

from sklearn import preprocessing  # -> ön isleme

labelEnCode = preprocessing.LabelEncoder()

ulke[:,0]= labelEnCode.fit_transform(veriler.iloc[:,0]) #->fit_transform yöntemi, veriyi hem eğitir
                                                        # hem de dönüştürür bu nedenle genellikle
                                                        #eğitim verisi üzerinde kullanılır
print(ulke)

oneHotEncode = preprocessing.OneHotEncoder()
ulke= oneHotEncode.fit_transform(ulke).toarray()
print(ulke)
```

```
 [0]
 [0]]
[[0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]]
```

Veri eksikligi ve encode(kategorik verileri sayıya dondurma[one-hot_encoding]) işlemleri sonucu;



Verilerin test ve train olarak bolunmesi ;

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train , y_test = train_test_split(s,cinsiyetDF,test_size=0.33,random_state=0)
# x bağımssız y bağımlı degisken
# test_size ->%67 train %33 test
# hedef = bağımlı degisken
# x train ve x testleri kullarak y[bağımlı degsiskeni] tahmin ettircez
```

x_test - DataFrame

Index	fr	tr	us	boy	kilo	yas
20	1	0	0	164	66	32
10	0	0	1	165	55	27
14	0	0	1	167	62	55
13	0	0	1	162	59	41
1	0	1	0	125	36	11
21	1	0	0	166	56	42
11	0	0	1	155	50	44
19	1	0	0	159	40	29

y_test - DataFrame

Index	cinsiyet
20	k
10	k
14	k
13	k
1	e
21	k
11	k
19	k

x_train - DataFrame

Index	fr	tr	us	boy	kilo	yas
8	0	1	0	177	60	22
6	0	1	0	190	80	25
16	1	0	0	193	90	28.45
4	0	1	0	129	38	12
2	0	1	0	135	34	10
5	0	1	0	180	90	30
17	1	0	0	187	80	27
9	0	0	1	185	105	33
7	0	1	0	175	90	35
18	1	0	0	183	88	28
3	0	1	0	133	30	9
0	0	1	0	130	30	10
15	1	0	0	174	70	47
12	0	0	1	160	58	28.45

y_train - DataFrame

Index	cinsiyet
8	k
6	e
16	e
4	e
2	k
5	e
17	e
9	e
7	e
18	e
3	k
0	e
15	e
12	k

Evet, `train_test_split` fonksiyonunun genel kullanımı bu şekildedir. Bu fonksiyon, veri setini eğitim ve test setlerine bölmek için oldukça yaygın bir araçtır ve makine öğrenimi modelinin eğitilmesi ve değerlendirilmesi için veri bölme işlemini gerçekleştirmek için sıklıkla kullanılır.

Genel kullanım şu şekildedir:

python Copy code

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(s, cinsiyetDF, test_size=0.
```

Burada:

- `s`: Bağımsız değişkenleri (özellikleri) içeren veri setidir.
- `cinsiyetDF`: Bağımlı değişkeni (hedef değişkeni) içeren veri setidir.
- `test_size`: Veri setinin ne kadarının test seti olarak ayrılacağını belirten orandır. Genellikle 0 ile 1 arasında bir değerdir ve bu oran, test setinin toplam veri setine oranını ifade eder.
- `random_state`: Veri setinin rastgele bölünmesinde kullanılacak rastgele durumun belirlenmesini sağlayan bir değerdir. Bu, işlemi tekrarlandığında aynı bölünmenin elde edilmesini sağlar.

Bu satır, veri setini eğitim ve test setlerine böler ve sonuç olarak dört değişkene atanır: `x_train`, `x_test`, `y_train`, ve `y_test`. Bu değişkenler, sırasıyla eğitim setinin özellikleri, test setinin özellikleri, eğitim setinin hedefleri ve test setinin hedeflerini içerir. Bu şekilde, model eğitimi ve performans değerlendirmesi için gerekli veriler elde edilmiş olur.

Öznitelik ölçekleme; [verileri aynı dünyaya çevirmek birbirine yakın sayılar elde etmek][birbirine göre ölçeklenmiş oldu standart scaler kullanarak]

X_test - NumPy object array

	0	1	2	3	4	5
0	1.29099	-0.377964	-1	0.4724	1.32854	-0.249913
1	-0.774597	-0.377964	1	0.549527	0.20439	-0.649773
2	-0.774597	-0.377964	1	0.70378	0.919757	1.58944
3	-0.774597	-0.377964	1	0.318147	0.613171	0.469836
4	-0.774597	2.64575	-1	-2.53554	-1.73732	-1.92932
5	1.29099	-0.377964	-1	0.626653	0.306586	0.549808
6	-0.774597	-0.377964	1	-0.221739	-0.306586	0.709752
7	1.29099	-0.377964	-1	0.0867674	-1.32854	-0.489829

X_train - NumPy object array

	0	1	2	3	4	5
0	-0.632456	0.866025	-0.408248	0.450494	-0.296579	-0.247171
1	-0.632456	0.866025	-0.408248	1.00825	0.509655	0.0341619
2	1.58114	-1.1547	-0.408248	1.13696	0.912772	0.357695
3	-0.632456	0.866025	-0.408248	-1.60891	-1.18344	-1.18495
4	-0.632456	0.866025	-0.408248	-1.35148	-1.34468	-1.3725
5	-0.632456	0.866025	-0.408248	0.579207	0.912772	0.503051
6	1.58114	-1.1547	-0.408248	0.879537	0.509655	0.221717
7	-0.632456	-1.1547	2.44949	0.793728	1.51745	0.784384
8	-0.632456	0.866025	-0.408248	0.364686	0.912772	0.971939
9	1.58114	-1.1547	-0.408248	0.70792	0.832148	0.315495
10	-0.632456	0.866025	-0.408248	-1.43729	-1.50593	-1.46628
11	-0.632456	0.866025	-0.408248	-1.566	-1.50593	-1.3725
12	1.58114	-1.1547	-0.408248	0.321782	0.106538	2.09727
13	-0.632456	-1.1547	2.44949	-0.278878	-0.377202	0.357695

```
from sklearn.preprocessing import StandardScaler # -> verileri standartlastırmaya yarar

sc=StandardScaler()

X_train = sc.fit_transform(x_train)
X_test = sc.fit_transform(x_test)
#Öznitelik ölçekleme; [verileri aynı dünyaya çevirmek birbirine yakın sayılar elde etmek]
#[birbirine göre ölçeklenmiş oldu]
# VERİLER STANDARTLAŞTIRILDI [StandartScaler]
```

Veri Önişleme - ADIMLAR:

1- Veri yükleme: [pd.read_csv('eksikveriler.csv')]

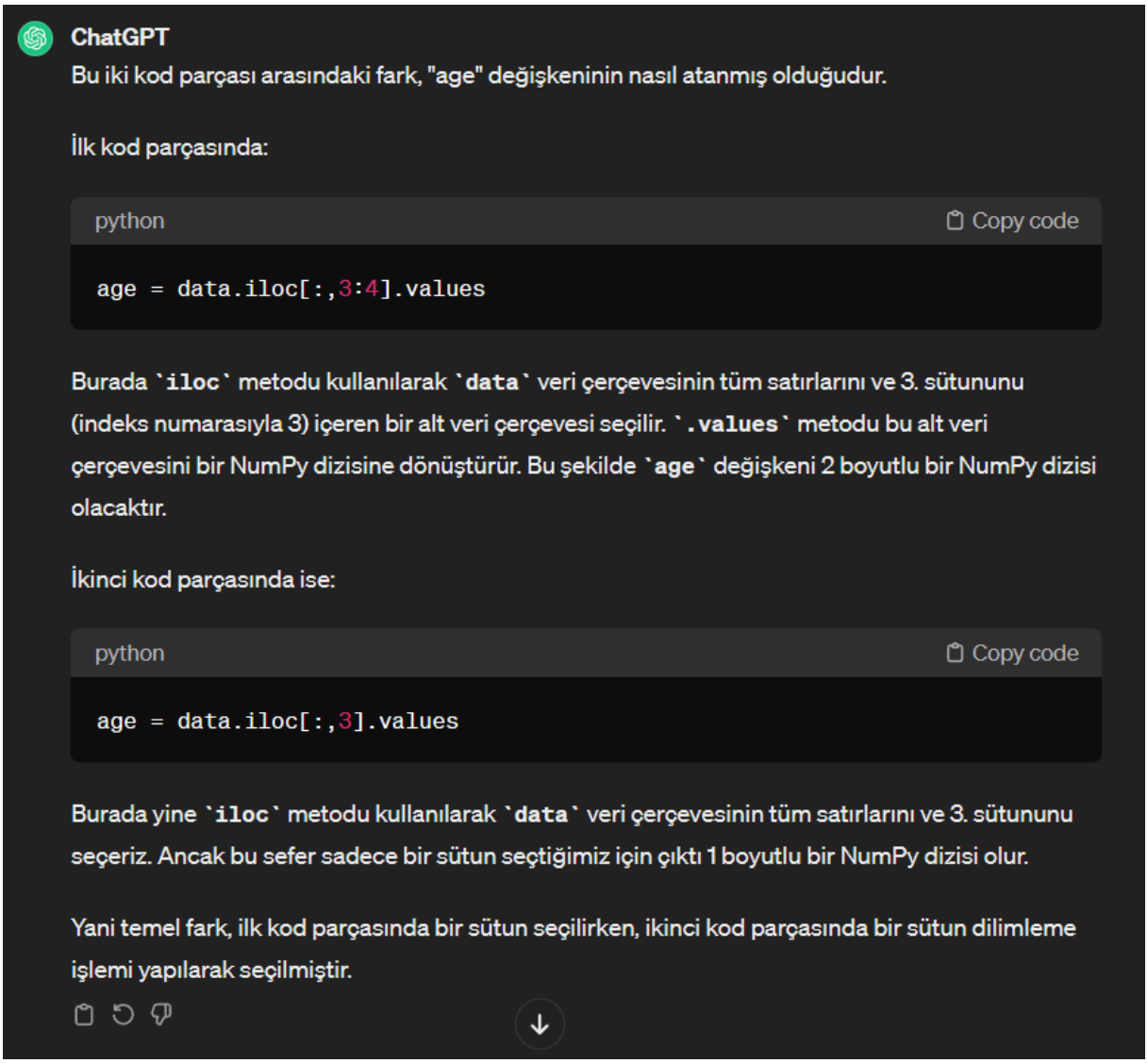
2- Eksik veri tamamlama: [imputer= SimpleImputer(missing_values=np.nan, strategy='mean')]

3- Kategorik[nominal] veri donuşumu: [preprocessing.OneHotEncoder()]

4- Veri kümelerini birleştirme: [pd.concat([s,cinsiyetDF],axis=1)]

5- Verileri bölme: [train and test]
[x_train, x_test,y_train ,y_test =train_test_split(s,cinsiyetDF,test_size=0.33,random_state=0)]

6- Öznitelik ölcekleme: [StandardScaler().fit_transform(x_train)]



One-hot encoding : ülke gibi herhangi bir sıralaması olmayan - cinsiyet

Label encoding : sıralamasi olacak

"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

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

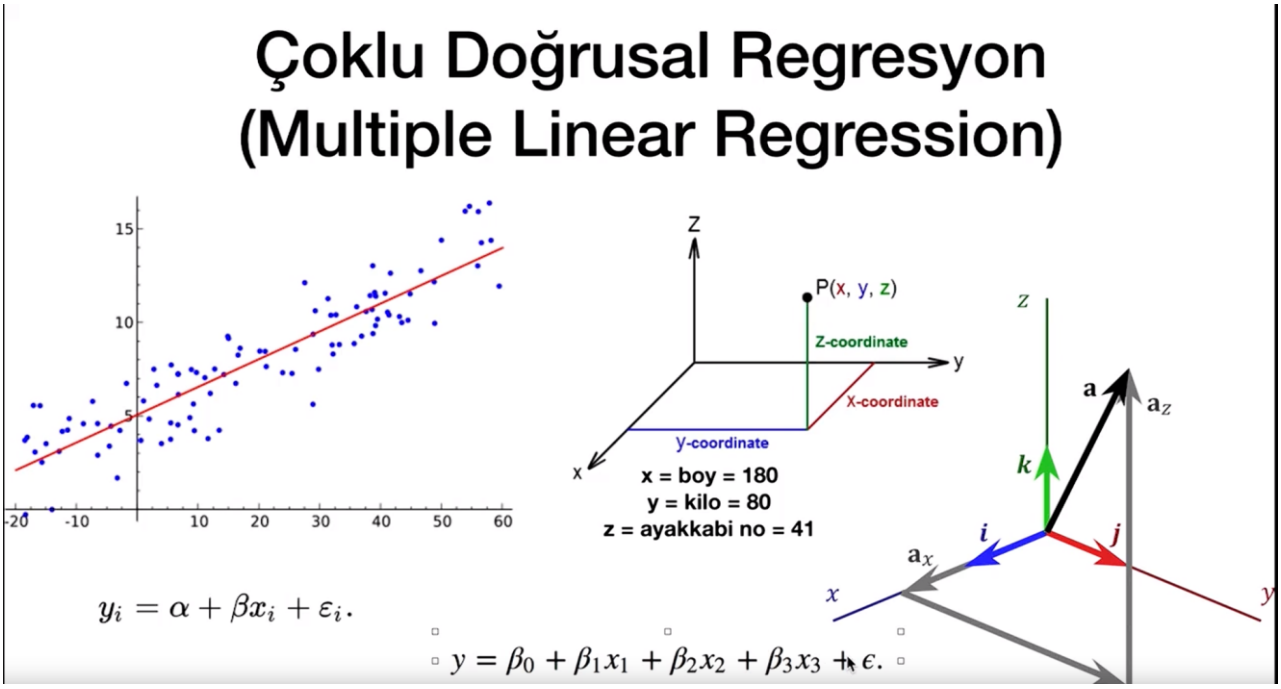
$$y_i = \alpha + \beta x_i + \varepsilon_i.$$

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.

---

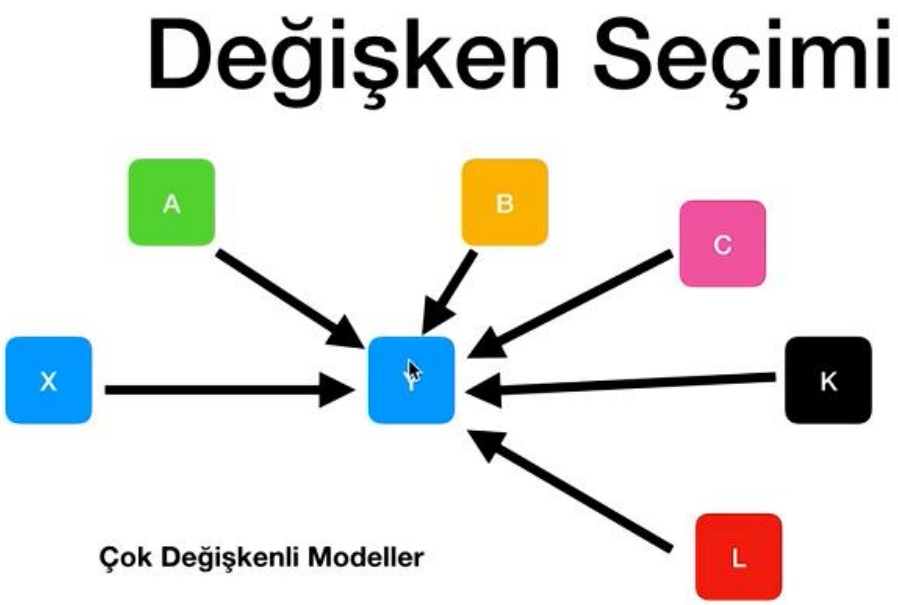
p-value (olasılık değeri)

- 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



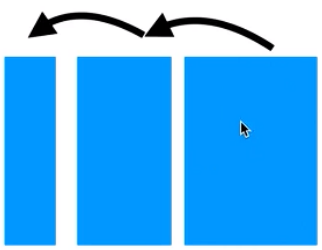
Burada önemli olan hangi bağımsız değişken bağımlı değişkeni daha fazla,daha az veya hic etkilemiyor bunun analizi.

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

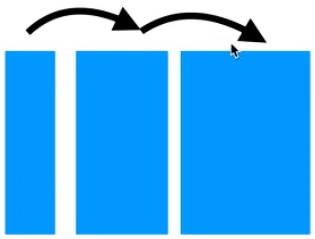
4.Bu aşamada, 3. adımda seçilen ve en yüksek p-değerine sahip değişken sistemden kaldırılır

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.

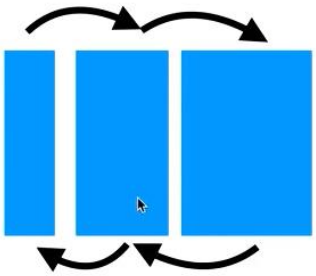
3.En **düşük** p-value değerine sahip olan değişken ele alınır

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.

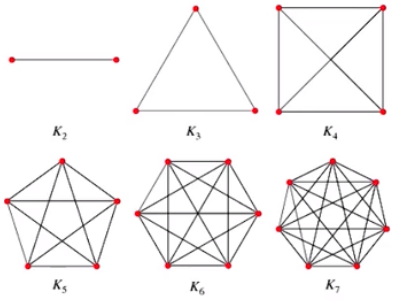
3.En **düşük** p-value değerine sahip olan değişken ele alınır

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

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

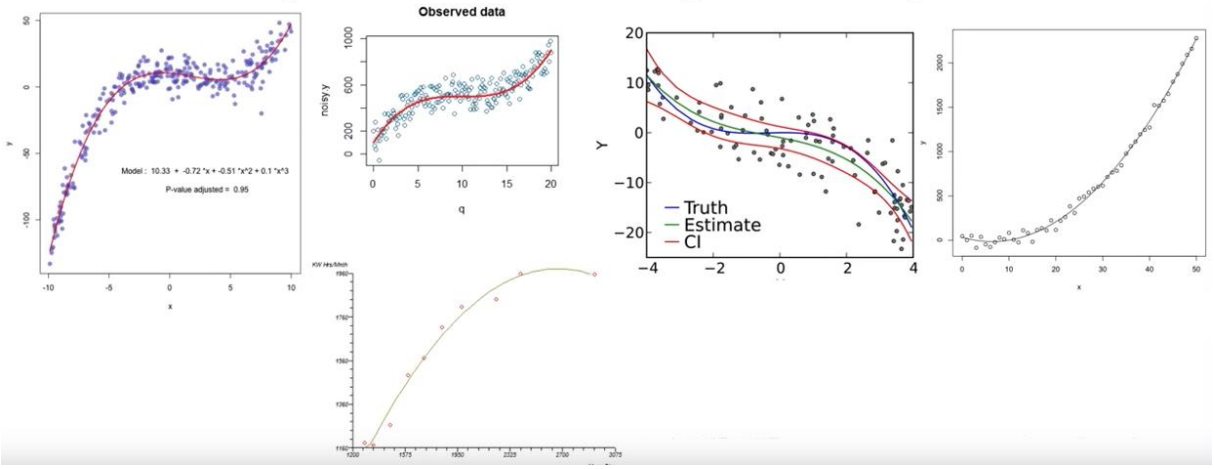
Polinomal Regresyon

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

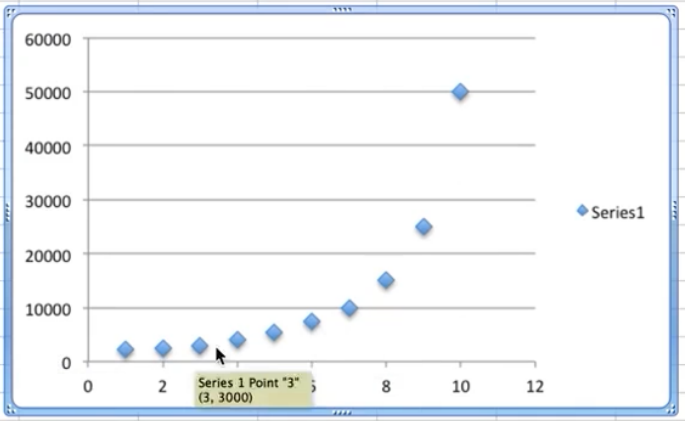
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{12} x_1 x_2 + \varepsilon$$

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

Polinomal Regresyon
(Polinomial Regression)



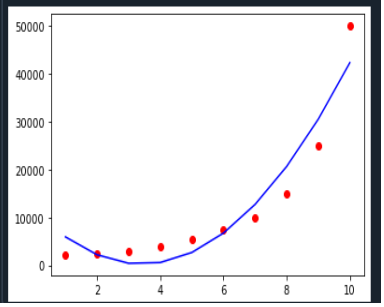
	A	B	C
1	unvan	Egitim Seviyesi	maas
2	Cayci	1	2250
3	Sekreter	2	2500
4	Uzman Yardi	3	3000
5	Uzman	4	4000
6	Proje Yoneti	5	5500
7	Sef	6	7500
8	Mudur	7	10000
9	Direktor	8	15000
10	C-level	9	25000
11	CEO	10	50000



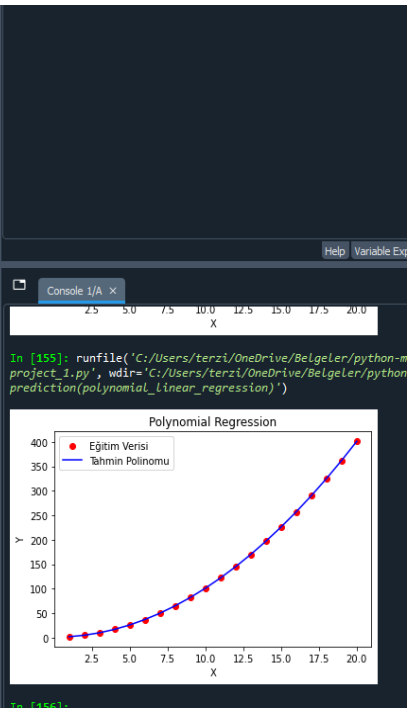

```
# linear regression
lin_reg = LinearRegression().fit(X,Y)

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')
```

```
Console 1/A
[[ 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.]]
```

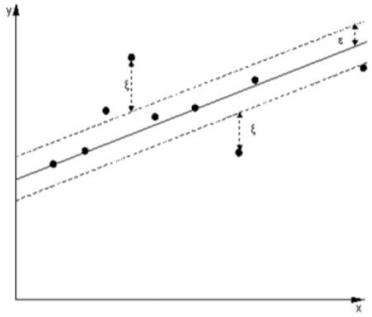
print(x_poly)



```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)



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

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

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

Polynomial

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

Gaussian Radial Basis function

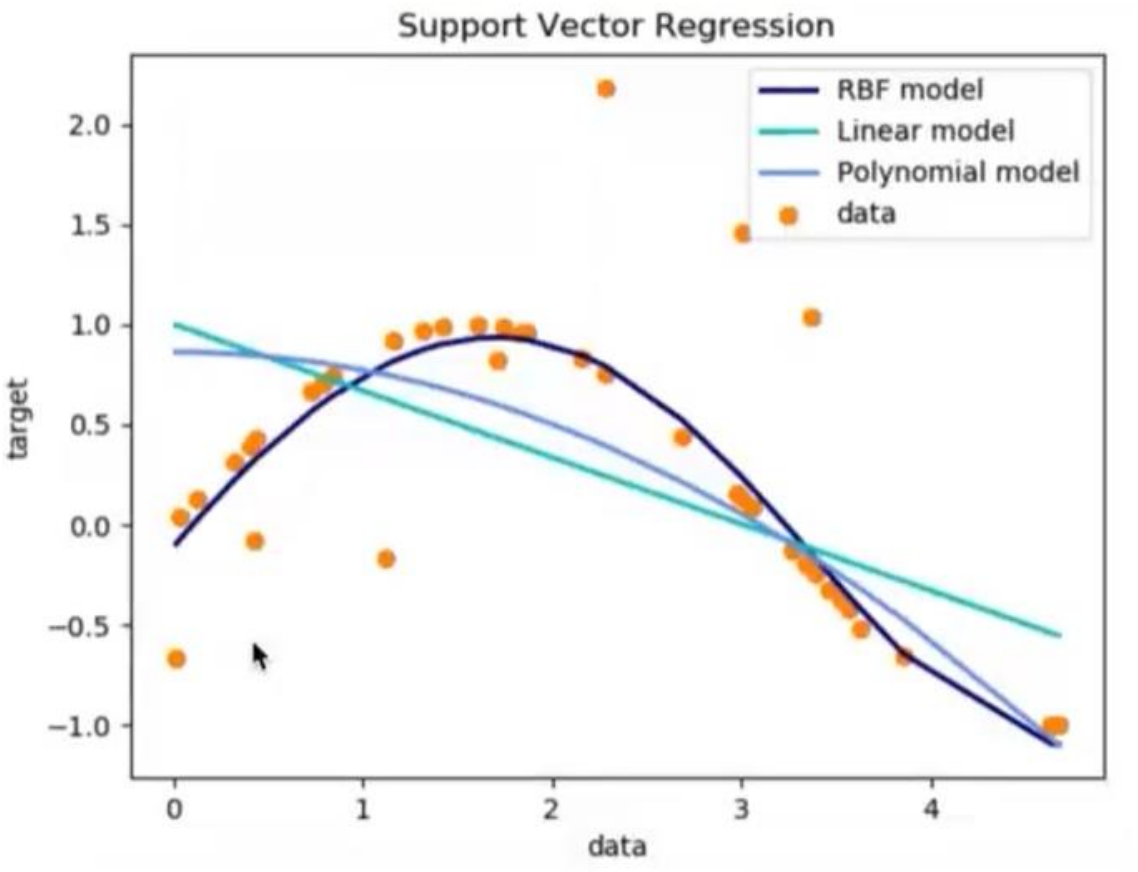
$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

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;

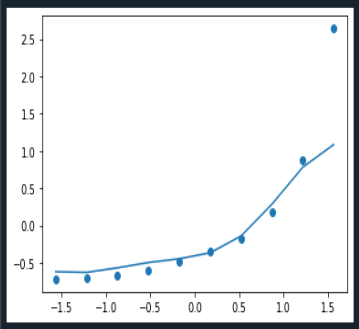
Doğrusal , doğrusal olmayan[polynomial, rbf(radial basis function)] , exp




```
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)
```

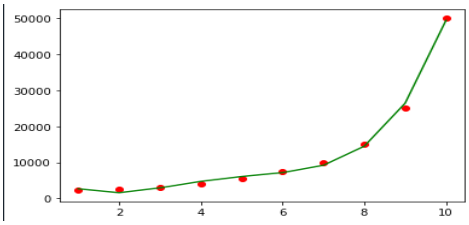
```
# support vector regression scaler ile kullanmamız gerekli veriler uzerindeki marjinal verilere duyarlı

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)