**MAKİNE ÖĞRENMESİ**

Makine öğrenmesi (ML), bir bilgisayarın doğrudan yönergeler olmadan öğrenmesine yardımcı olmak için matematiksel modelleri kullanma işlemidir. Bu, yapay zekanın (AI) bir alt kümesi olarak kabul edilir. Makine öğrenmesi, verilerdeki kalıpları belirlemek için algoritmaları kullanır. Tahmin yapabilen bir veri modeli oluşturmak için de bu kalıplar kullanılır. Tıpkı insanların daha fazla alıştırma yaptıkça gelişmesi gibi, veri ve deneyim miktarı arttıkça makine öğrenmesinin sonuçları da daha doğru hale gelir.

**Makine öğrenmesi algoritmaları**

Makine öğrenmesi algoritmaları, insanların karmaşık veri kümelerini keşfetmesine, analiz etmesine ve bunlarda anlam bulmasına yardımcı olan kod parçacıklardır. Her algoritma, bir makinenin belirli bir hedefi gerçekleştirmek için izleyebileceği sınırlı ve belirli adım adım ilerleyen yönerge kümesidir

**Makine öğrenmesi teknikleri**

**Supervized Learning (Denetimli Öğrenme)**

Denetimli öğrenmede, algoritmalar sağladığınız etiketli örnekleri temel alarak tahmin yapar. Bu teknik, sonucun nasıl görüneceğini bildiğiniz durumlarda faydalıdır.

Örneğin, son 100 yıla göre şehirlerin nüfuslarını içeren bir küme sağlayıp dört yıl sonra belirli bir şehrin nüfusunun ne olacağını öğrenmek istediğinizi varsayalım. Sonuç, veri kümelerinde mevcut olan etiketleri kullanır: nüfus, şehir ve yıl.

**Classification(Sınıflandırma):**

Sınıflandırma algoritmaları, verileri önceden ayarlanmış kategorilere atamak için tahmine dayalı hesaplamalar kullanır. Sınıflandırma algoritmaları giriş verileriyle eğitilir ve şöyle soruları yanıtlamak için kullanılır:

* Bu istenmeyen bir e-posta mı?

Verilen metnin yaklaşımı (olumlu, olumsuz, nötr) nedir?

Algoritmalar:

* **Support Vector Machines**
* Discriminant Analysis
* Naive Bayes
* **Nearest Neighbor**
* **Neural Networks**

**Regression Yöntemleri:**

Regresyon algoritmaları, geçmiş verileri temel alarak yeni bir veri noktasının değerini tahmin eder. Şu gibi soruları yanıtlamanıza yardımcı olurlar:

* Oturduğum şehirde iki yatak odalı bir evin ortalama fiyatı ne olacak?
* Salı günü kaç hasta kliniği ziyaret edecek?

**Algoritmalar:**

* **Linear Regression**
* SVR,GPR
* Ensemble Methods
* **Decisions Tree**
* **Neural Networks**

**Unsupervized Learning(Denetimsiz Öğrenme)**

Denetimsiz öğrenmede veri noktaları etiketlenmez. Algoritma, verileri düzenleyerek veya bunların yapısını açıklayarak veri noktalarını sizin için etiketler. Bu teknik, sonucun nasıl görüneceğini bilmediğiniz durumlarda faydalıdır.

Örneğin, müşteri verilerini sağlayıp benzer ürünlerden hoşlanan müşterilerin segmentlerini oluşturmak istediğinizi varsayalım. Sağladığınız veriler etiketlenmez ve sonuçtaki etiketler, veri noktalarında keşfedilen benzerlikler temel alınarak oluşturulur.

**Clustering(Kümeleme) Yöntemi:**

Kümeleme algoritmaları, veri noktaları arasındaki benzerlik düzeyini belirleyerek verileri birden fazla gruba böler. Kümeleme algoritmaları şunun gibi sorular için uygundur:

* Hangi izleyiciler aynı tür filmleri izlemekten hoşlanıyor?
* Hangi yazıcı modelleri aynı şekilde hatayla karşılaşıyor?

**Algoritmalar:**

* **K-means,**KMethods, Fuzzy
* Hierarchical
* Gaussian Mixture
* Hidden Markow Model
* **Neural Networks**

**Pekiştirmeye dayalı öğrenme**

Pekiştirmeye dayalı öğrenme, sonuçlardan öğrenen ve gerçekleştirilecek eylemi kararlaştıran algoritmaları kullanır. Algoritma, her eylemden sonra seçeneğin doğru mu, nötr mü yoksa yanlış mı olduğunu belirlemeye yardımcı olan geri bildirim alır. İnsan kılavuzluğu olmadan birçok küçük kararlar alması gereken otomatikleştirilmiş sistemler için kullanılabilecek iyi bir tekniktir.

Örneğin, sürücüsüz bir araç tasarlıyorsunuz ve bu aracın yasalara uyduğundan ve insan güvenliğini sağladığından emin olmak istiyorsunuz. Araç deneyim ve pekiştirme geçmişi kazandıkça şeritte kalmayı, hız limitini aşmamayı ve yayaları görünce fren yapmayı öğrenir.

**DATASETS**

Advertising.csv

Maaslar.csv(poly)

breast\_cancer(logic)

iris (tree)

veriler(svm)

**MAKİNE ÖĞRENMESİ YÖNTEMLERİNİN PYTHON UYGULAMALARI**

**Scikit-learn,** veri bilimi ve machine learning için en yaygın kullanılan Python paketlerinden biridir. Birçok işlemi gerçekleştirmenizi sağlar ve çeşitli algoritmalar sağlar. Scikit-learn ayrıca sınıfları, yöntemleri ve işlevleri ile kullanılan algoritmaların arka planıyla ilgili belgeler sunar.

**Regresyon b**ir bağımlı değişken ile diğer birkaç bağımsız değişken arasındaki ilişkiyi belirler. Regresyon analizi, bağımsız değişkenlerin bazıları değiştiğinde bağımlı değişkenin nasıl değiştiğini anlamaya yardımcı olmaktadır.

**https://miro.medium.com/max/297/1*3n_BKBPw0xeNlQFz91X3Mg.png**

**[RMSE] Kök Ortalama Kare Hata (Root Mean Square Error):**

Bir makine öğrenmesi modelinin, tahminleyicinin tahmin ettiği değerler ile gerçek değerleri arasındaki uzaklığın bulunmasında sıklıkla kullanılan, hatanın büyüklüğünü ölçen kuadratik bir metriktir.  RMSE değerinin sıfır olması modelin hiç hata yapmadığı anlamına gelir.

**1)Lineer Regresyon:**

**"""**

**Created on Thu May 6 01:27:27 2021**

**@author: Ayşe Özateş**

**"""**

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

df = pd.read\_csv("advertising.csv",usecols=[1,2,3,4])

print(df.head())

print("-------------------------------------------------------")

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, cross\_val\_predict

X = df.drop("sales", axis = 1)# tüm bağımsız değişkenleri seçiyorum

y = df["sales"] #bağımlı değişken olan sales i y olarak tanımlıyorum

#Eğitim ve test setlerimizi oluşturuyoruz

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20, random\_state= 42)

print(" X\_Train ",X\_train.shape)

print(" Y\_Train ",y\_train.shape)

print(" X\_Test ",X\_test.shape)

print(" y\_Test ",y\_test.shape)

training = df.copy()

print("Training:",training.shape)

reg = LinearRegression()

# %% fitting data

model = reg.fit(X\_train, y\_train)

print("model\_intercept ", model.intercept\_)

print("model\_coef: ",model.coef\_)

**# predict**

**# Örneğin 30 birim Tv harcaması , 10 birim radio harcaması, 40 birimde gazete harcaması olduğunda**

**#satışların tahmini değeri ne olur?**

yeni\_veri=[[30],[10],[40]]

yeni\_veri=pd.DataFrame(yeni\_veri).T

sales=model.predict(yeni\_veri)

print("Tahmin Edilen Sales:",sales**)**

**#Modelimizin Tahmin Başarısı için;**

#Eğitim Setinin Hatası

rmse = np.sqrt(mean\_squared\_error(y\_train, model.predict(X\_train)))

print("Eğitim Setinin Hatası:",rmse)

#Test Setinin Hatası

rmse = np.sqrt(mean\_squared\_error(y\_test, model.predict(X\_test)))

print("Test Setinin Hatası:",rmse)

**OUTPUT:**

TV radio newspaper sales

0 230.1 37.8 69.2 22.1

1 44.5 39.3 45.1 10.4

2 17.2 45.9 69.3 9.3

3 151.5 41.3 58.5 18.5

4 180.8 10.8 58.4 12.9

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

X\_Train (160, 3)

Y\_Train (160,)

X\_Test (40, 3)

y\_Test (40,)

Training: (200, 4)

model\_intercept 2.979067338122629

model\_coef: [0.04472952 0.18919505 0.00276111]

Tahmin Edilen Sales: [6.32334798]

Eğitim Setinin Hatası: 1.644727765644337

Test Setinin Hatası: 1.7815996615334508

**2)POLİNOM REGRESYON**

polynomial regression = y = b0 + b1\*x +b2\*x^2 + b3\*x^3 + ... + bn\*x^n

# -\*- coding: utf-8 -\*-

"""

Created on Thu May 6 03:37:44 2021

@author: ayşe özateş

"""

import numpy as np

import pandas as pd

**import matplotlib.pyplot as plt**

**#Veri Yükleme,**

df = pd.read\_csv("maaslar.csv")

**#DataFrame Oluşturma**

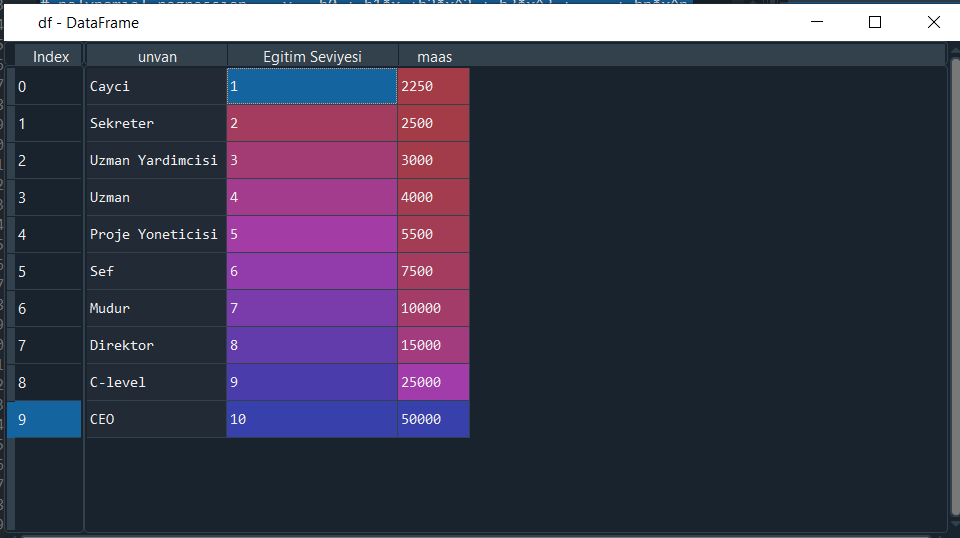
x=df.iloc[:,1:2] #Eğitim Seviyesi

y=df.iloc[:,2:] #Maaş

**#Numpy Array dönüşümü**

X=x.values

Y=y.values

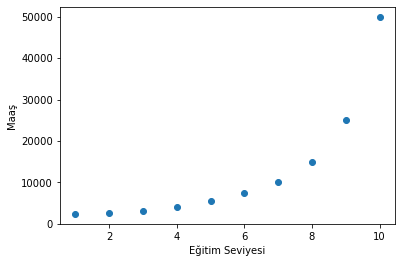


plt.scatter(X,Y)

plt.ylabel("Maaş")

plt.xlabel("Eğitim Seviyesi")

plt.show()



# linear regression = y = b0 + b1\*x

# multiple linear regression y = b0 + b1\*x1 + b2\*x2

**# %% linear regression oluşturma**

from sklearn.linear\_model import LinearRegression

linear\_reg = LinearRegression()

linear\_reg.fit(X,Y)

**# polynomial regression = y = b0 + b1\*x +b2\*x^2 + b3\*x^3 + ... + bn\*x^n**

from sklearn.preprocessing import PolynomialFeatures

polynomial\_reg = PolynomialFeatures(degree = 4) #Dördüncü dereceden bir polinom

x\_polynomial = polynomial\_reg.fit\_transform(X)

print(x\_polynomial)

linear\_reg2 = LinearRegression()

linear\_reg2.fit(x\_polynomial,y)

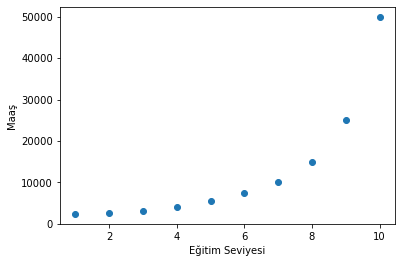
**#GÖRSELLEŞTİRME**

plt.scatter(X,Y)

plt.ylabel("Maaş")

plt.xlabel("Eğitim Seviyesi")

plt.show()

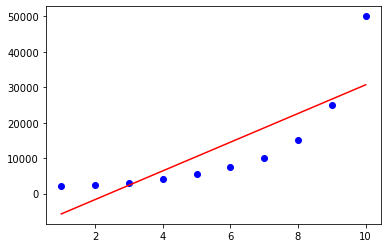


**#Linear Regresyon**

plt.scatter(X,Y,color="blue")

plt.plot(x,linear\_reg.predict(X),color="red")

plt.show()



**# %% Polinom Regresyon**

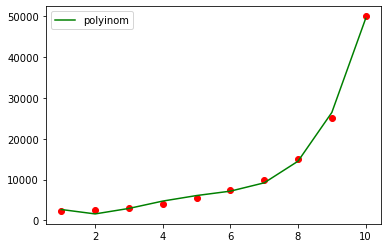
plt.scatter(X,Y,color="red")

y\_head2 = linear\_reg2.predict(x\_polynomial)

plt.plot(X,y\_head2,color= "green",label = "polyinom")

plt.legend()

plt.show()

****

**#Tahminler**

print("Eğitim Seviyesi 12 Olanın Maaşı:",linear\_reg.predict([[12]]))

print("Eğitim Seviyesi 6.6 Olanın Maaşı:",linear\_reg.predict([[6.6]]))

print("Eğitim Seviyesi 12 Olanın Maaşı:",linear\_reg2.predict(polynomial\_reg.fit\_transform([[12]])))

print("Eğitim Seviyesi 6.6 Olanın Maaşı:",linear\_reg2.predict(polynomial\_reg.fit\_transform([[6.6]])))

**OUTPUT**

[[1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00]

[1.000e+00 2.000e+00 4.000e+00 8.000e+00 1.600e+01]

[1.000e+00 3.000e+00 9.000e+00 2.700e+01 8.100e+01]

[1.000e+00 4.000e+00 1.600e+01 6.400e+01 2.560e+02]

[1.000e+00 5.000e+00 2.500e+01 1.250e+02 6.250e+02]

[1.000e+00 6.000e+00 3.600e+01 2.160e+02 1.296e+03]

[1.000e+00 7.000e+00 4.900e+01 3.430e+02 2.401e+03]

[1.000e+00 8.000e+00 6.400e+01 5.120e+02 4.096e+03]

[1.000e+00 9.000e+00 8.100e+01 7.290e+02 6.561e+03]

[1.000e+00 1.000e+01 1.000e+02 1.000e+03 1.000e+04]]

Eğitim Seviyesi 12 Olanın Maaşı: [[38760.60606061]]

Eğitim Seviyesi 6.6 Olanın Maaşı: [[16923.33333333]]

Eğitim Seviyesi 12 Olanın Maaşı: [[151799.24242426]]

Eğitim Seviyesi 6.6 Olanın Maaşı: [[8146.9948718]]

**3)LOGİSTİC REGRESSİON**

Lojistik, bağımlı değişkenin kategorik olduğu bir regresyon yöntemidir. Diğer bir ifade bağımlı değişkenlerin sürekli çıkış değerleri yerine sınıfları tahmin edilir. Lojistik regresyon, s, bağımsız x değişkeninin -∞ ile +∞ arasında değerler alabilen doğrusal işlevi olmak üzere,

f(s) = ℯ 𝑠 1 + ℯ 𝑠 = 1 1 + ℯ −𝑠 2.2 işlevi ile ifade edilir [12]

"""

Created on Fri May 7 03:06:50 2021

@author: Ayşe Özateş

"""Logistic Regression"""

import matplotlib.pyplot as plt

from sklearn.datasets import load\_breast\_cancer

veriler=load\_breast\_cancer()

x=veriler.data

y=veriler.target

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.3,

train\_size=0.7,random\_state=88)

from sklearn.linear\_model import LogisticRegression

Lr=LogisticRegression(max\_iter=3000)

Lr.fit(X\_train,y\_train)

predicted\_classes\_Lr=Lr.predict(X\_test)

**# Değerlendirme Ölçütleri- Confusion Matrix**

from sklearn.metrics import confusion\_matrix,classification\_report

Conf\_Matrix=confusion\_matrix(y\_test,predicted\_classes\_Lr)

Class\_rep=classification\_report(y\_test,predicted\_classes\_Lr)

**#ROC\_CURVE**

from sklearn.metrics import roc\_curve

y\_prob=Lr.predict\_proba(X\_test)

y\_prob=y\_prob[:,1]

FPR,TPR,Thresholds=roc\_curve(y\_test,y\_prob)

plt.plot(FPR,TPR)

plt.xlabel("FPR")

plt.ylabel("TPR")

plt.show()

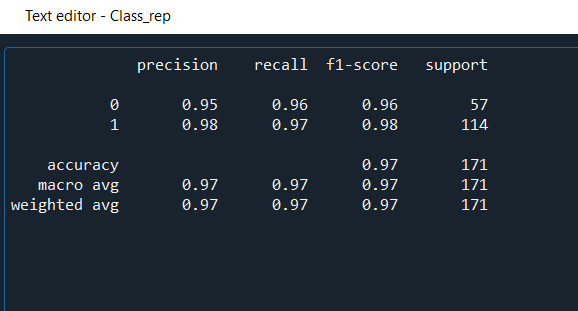
**#ROC\_AUC\_SCORE**

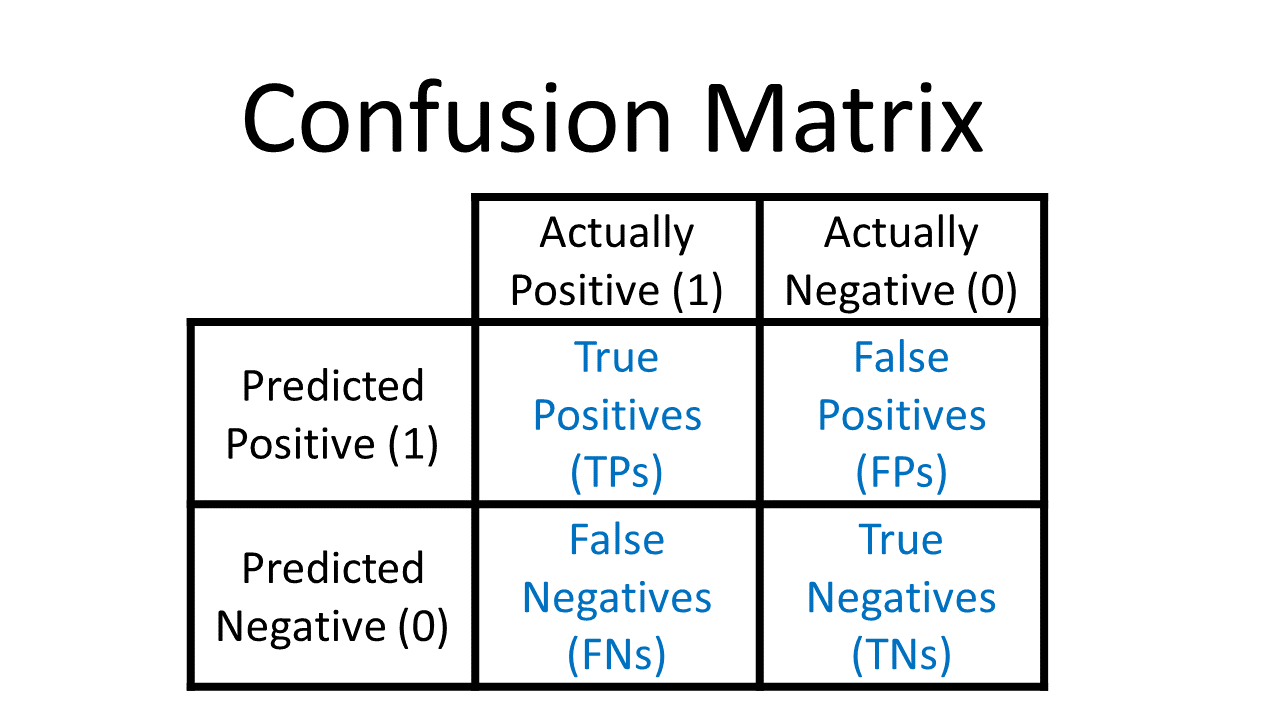
from sklearn.metrics import roc\_auc\_score

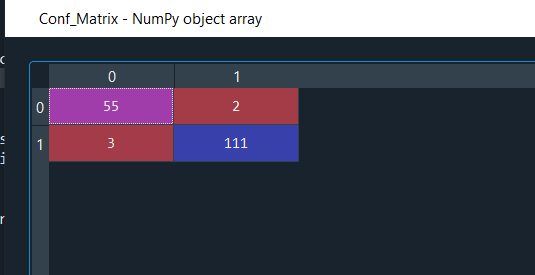
roc\_auc=roc\_auc\_score(y\_test,y\_prob)

print(roc\_auc)

**OUTPUT:**

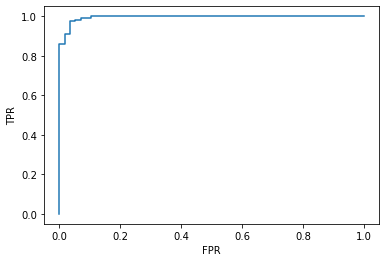
****



****

**ROC\_AUC\_SCORE: 0.994921514312096**

**ROC\_CURVE:**



**4)KARARAĞACI**

Verileri iki veya daha fazla homojen kümeye ayıran karar ağacı algoritmaları. Verileri, veri noktaları arasındaki en önemli fark yaratan öğeyi temel alarak ayırmak için if-then kurallarını kullanırlar.

# -\*- coding: utf-8 -\*-

"""

Created on Fri May 7 04:12:39 2021

@author: Ayşe Özateş

"""

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

from sklearn.datasets import load\_iris

iris=load\_iris()

iris.feature\_names

Data\_iris=iris.data

Data\_iris=pd.DataFrame(Data\_iris,columns=iris.feature\_names)

Data\_iris['label']=iris.target

plt.scatter(Data\_iris.iloc[:,2],Data\_iris.iloc[:,3],c=iris.target)

plt.xlabel("Petal Length (cm)")

plt.ylabel("Petal width(cm)")

plt.show()

x=Data\_iris.iloc[:,0:4]

y=Data\_iris.iloc[:,4]

X\_train, X\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,train\_size=0.8,

random\_state=88,shuffle=True,stratify=y)

Dt=DecisionTreeClassifier()

Dt.fit(X\_train,y\_train)

Predicted\_types\_Dt=Dt.predict(X\_test)

accuracy\_score(y\_test,Predicted\_types\_Dt)

from sklearn.model\_selection import cross\_val\_score

Scores\_Dt=cross\_val\_score(Dt,x,y,cv=10)

print("Scores\_Dt:",Scores\_Dt)

**5)** **SVM sınıflandırma**

# -\*- coding: utf-8 -\*-

"""

Created on Fri May 7 05:20:14 2021

@author: Ayşe Özateş

**"""**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

**# veri yükleme**

veriler=pd.read\_csv('veriler.csv')

print(veriler)

x=veriler.iloc[:,1:4].values #bağımsız değişkenler

**y=veriler.iloc[:,4:].values.flatten()#bağımlı değişken**

**#verilerin eğitim ve test için bölünmesi**

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size = 0.33,

random\_state=0)

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

X\_train=sc.fit\_transform(x\_train)

X\_test=sc.fit\_transform(x\_test)

from sklearn.svm import SVC

svc=SVC(kernel='poly')

svc.fit(X\_train,y\_train)

y\_pred=svc.predict(X\_test)

cm=confusion\_matrix(y\_test,y\_pred)

print("SVC")

print(cm)

**OUTPUT:**

ulke boy kilo yas cinsiyet

0 tr 130 30 10 e

1 tr 125 36 11 e

2 tr 135 34 10 k

3 tr 133 30 9 k

4 tr 129 38 12 e

5 tr 180 90 30 e

6 tr 190 80 25 e

7 tr 175 90 35 e

8 tr 177 60 22 k

9 us 185 105 33 e

10 us 165 55 27 k

11 us 155 50 44 k

12 us 160 58 39 k

13 us 162 59 41 k

14 us 167 62 55 k

15 fr 174 70 47 e

16 fr 193 90 23 e

17 fr 187 80 27 e

18 fr 183 88 28 e

19 fr 159 40 29 k

20 fr 164 66 32 k

21 fr 166 56 42 k

SVC

[[0 1]

[7 0]]

**6)K—MEANS**

**K-Means Algoritması**, etiketlenmemiş verileri, yani tanımlanmış kategoriler veya gruplar içermeyen verileri kategorilere ayırmak için kullanılan denetlenmeyen bir öğrenme türüdür. Algoritma, K değişkeni tarafından temsil edilen grupların sayısıyla veri içindeki grupları bularak çalışır. Ardından, verilen özelliklere dayanarak her bir veri noktasını K gruplarından birine atamak için tekrarlanır.

**# -\*- coding: utf-8 -\*-**

**"""**

**Created on Fri May 7 10:17:10 2021**

**@author: ayşe özateş**

**"""**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.datasets import load\_iris**

**iris=load\_iris()**

**Data\_iris=iris.data**

**"""K-Mean Kümeleme"""**

**from sklearn.cluster import KMeans**

**KMNS=KMeans(n\_clusters=3)**

**KMNS.fit(Data\_iris)**

**Labels=KMNS.predict(Data\_iris)**

**Ctn=KMNS.cluster\_centers\_**

**"""Grafik Görünümü"""**

**plt.scatter(Data\_iris[:,2],Data\_iris[:,3],c=Labels)**

**plt.scatter(Ctn[:,2],Ctn[:,3],marker="o",color="red",s=120)**

**plt.xlabel("Petal Length (cm)")**

**plt.ylabel("Petal Length(cm)")**

**plt.show()**

**KMNS.inertia\_**

**K\_inertia=[]**

**for i in range(1,10):**

**KMNS=KMeans(n\_clusters=i, random\_state=44)**

**KMNS.fit(Data\_iris)**

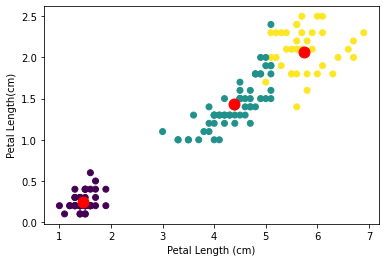
**K\_inertia.append(KMNS.inertia\_)**

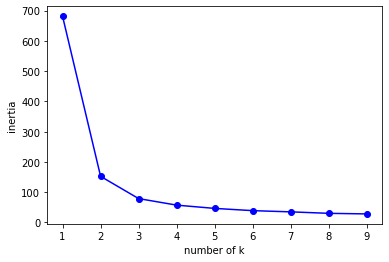
**plt.plot(range(1,10),K\_inertia,color="blue",marker="o")**

**plt.xlabel("number of k")**

**plt.ylabel("inertia")**

**plt.show()**





**7) Yapay Sinir Ağları**

**Model ve Tahmin**

df = diabetes.copy()

df = df.dropna()

y = df["Outcome"]

X = df.drop(['Outcome'], axis=1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30, random\_state=42)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X\_train)

X\_train\_scaled = scaler.transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

**X\_test\_scaled[0:5]**

**Out**: array([[ 0.69748316, -0.70719864, -0.64639893, 0.81207927, 0.95720244,

0.26575953, -0.11680393, 0.85019217],

[-0.52953881, -0.27388818, 0.29399563, 0.74746428, -0.6936878 ,

0.488933 , -0.94192338, -1.03426754],

[-0.52953881, -0.39769117, -0.31449497, -1.3202154 , -0.6936878 ,

-0.1543317 , -0.91266382, -1.03426754],

[ 1.31099414, -0.42864191, 0.57058226, -1.3202154 , -0.6936878 ,

-0.96825847, 1.12965312, 0.07927683],

[ 1.00423865, 0.46892976, 1.12375553, -1.3202154 , -0.6936878 ,

-0.27248236, -0.76051413, 1.44979298]])

from sklearn.neural\_network import MLPClassifier

mlpc = MLPClassifier().fit(X\_train\_scaled, y\_train)

y\_pred = mlpc.predict(X\_test\_scaled)

accuracy\_score(y\_test, y\_pred)

**Out:** 0.7359307359307359

## Model Tuning

Mlpc

mlpc\_params = {"alpha": [0.1, 0.01, 0.02, 0.005, 0.0001,0.00001],

"hidden\_layer\_sizes": [(10,10,10),

(100,100,100),

(100,100),

(3,5),

(5, 3)],

"solver" : ["lbfgs","adam","sgd"],

"activation": ["relu","logistic"]}

mlpc = MLPClassifier()

mlpc\_cv\_model = GridSearchCV(mlpc, mlpc\_params,

cv = 10,

n\_jobs = -1,

verbose = 2)

mlpc\_cv\_model.fit(X\_train\_scaled, y\_train)

**Out:** Fitting 10 folds for each of 180 candidates, totalling 1800 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 25 tasks | elapsed: 40.1s

[Parallel(n\_jobs=-1)]: Done 146 tasks | elapsed: 2.8min

[Parallel(n\_jobs=-1)]: Done 349 tasks | elapsed: 7.4min

[Parallel(n\_jobs=-1)]: Done 632 tasks | elapsed: 12.3min

[Parallel(n\_jobs=-1)]: Done 997 tasks | elapsed: 17.6min

[Parallel(n\_jobs=-1)]: Done 1442 tasks | elapsed: 22.0min

[Parallel(n\_jobs=-1)]: Done 1800 out of 1800 | elapsed: 25.2min finished

GridSearchCV(cv=10, estimator=MLPClassifier(), n\_jobs=-1,

param\_grid={'activation': ['relu', 'logistic'],

'alpha': [0.1, 0.01, 0.02, 0.005, 0.0001, 1e-05],

'hidden\_layer\_sizes': [(10, 10, 10), (100, 100, 100),

(100, 100), (3, 5), (5, 3)],

'solver': ['lbfgs', 'adam', 'sgd']},

verbose=2)

**print("En iyi parametreler: " + str(mlpc\_cv\_model.best\_params\_))**

En iyi parametreler: {'activation': 'logistic', 'alpha': 0.01, 'hidden\_layer\_sizes': (100, 100), 'solver': 'adam'}

mlpc\_tuned = MLPClassifier(activation = "logistic", alpha = 0.1, hidden\_layer\_sizes = (100, 100, 100), solver = "adam")

**mlpc\_tuned.fit(X\_train\_scaled, y\_train)**

**Out:**MLPClassifier(activation='logistic', alpha=0.1, hidden\_layer\_sizes=(100, 100, 100))

y\_pred = mlpc\_tuned.predict(X\_test\_scaled)

accuracy\_score(y\_test, y\_pred)

**Out:** 0.7359307359307359

**MAKİNE ÖĞRENMESİ YÖNTEMLERİNİN MATLAB UYGULAMALARI**

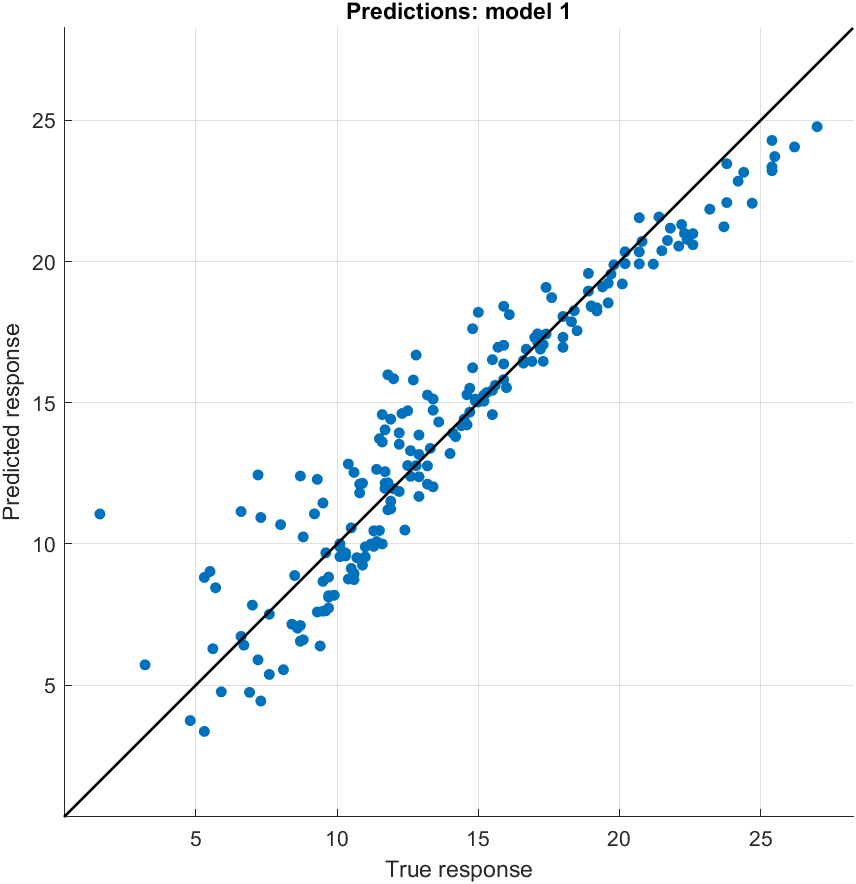
Statistics and Machine Learning Toolbox kullanılarak makine öğrenmesi algoritmaları uygulandı

**1)LİNEAR REGRESSİON**

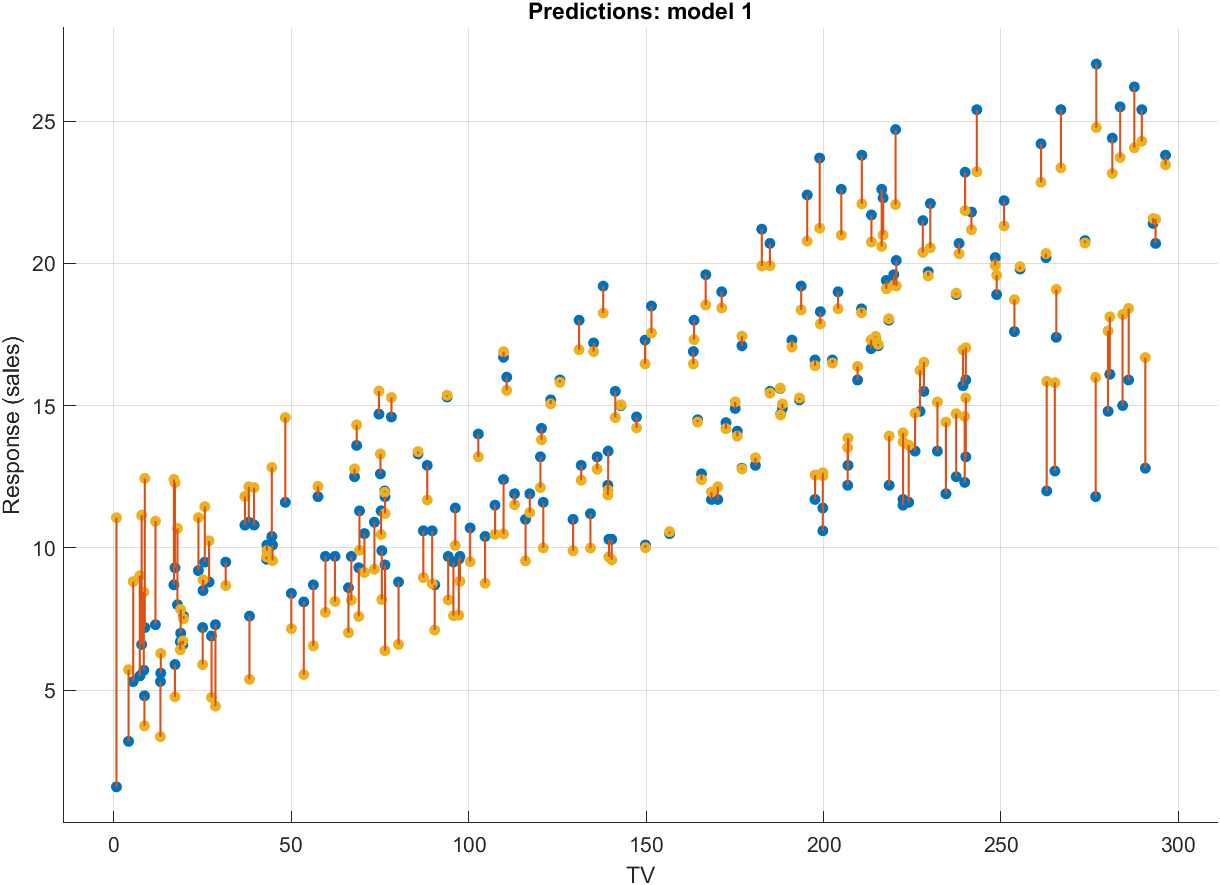
Advertising data seti ile matlabın Makine Öğrenmesi Tools kullanılarak lineer regression öğrenme yöntemi ile Tv,Radio ve Newspaper reklam harcamalarının satışlara olan etkisi tahmin edildi.

RMSE değeri ne kadar küçük olursa tahminlerimiz o kadar iyidir.

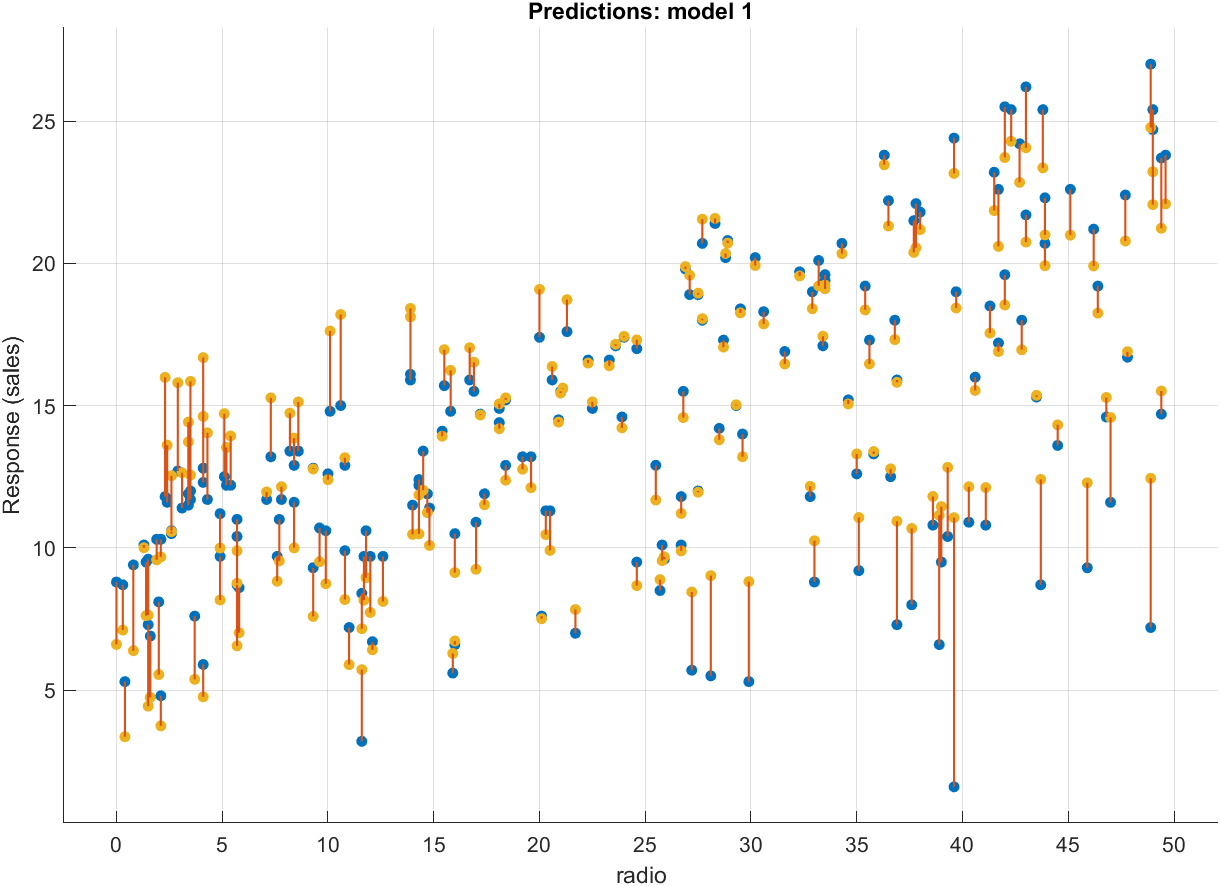




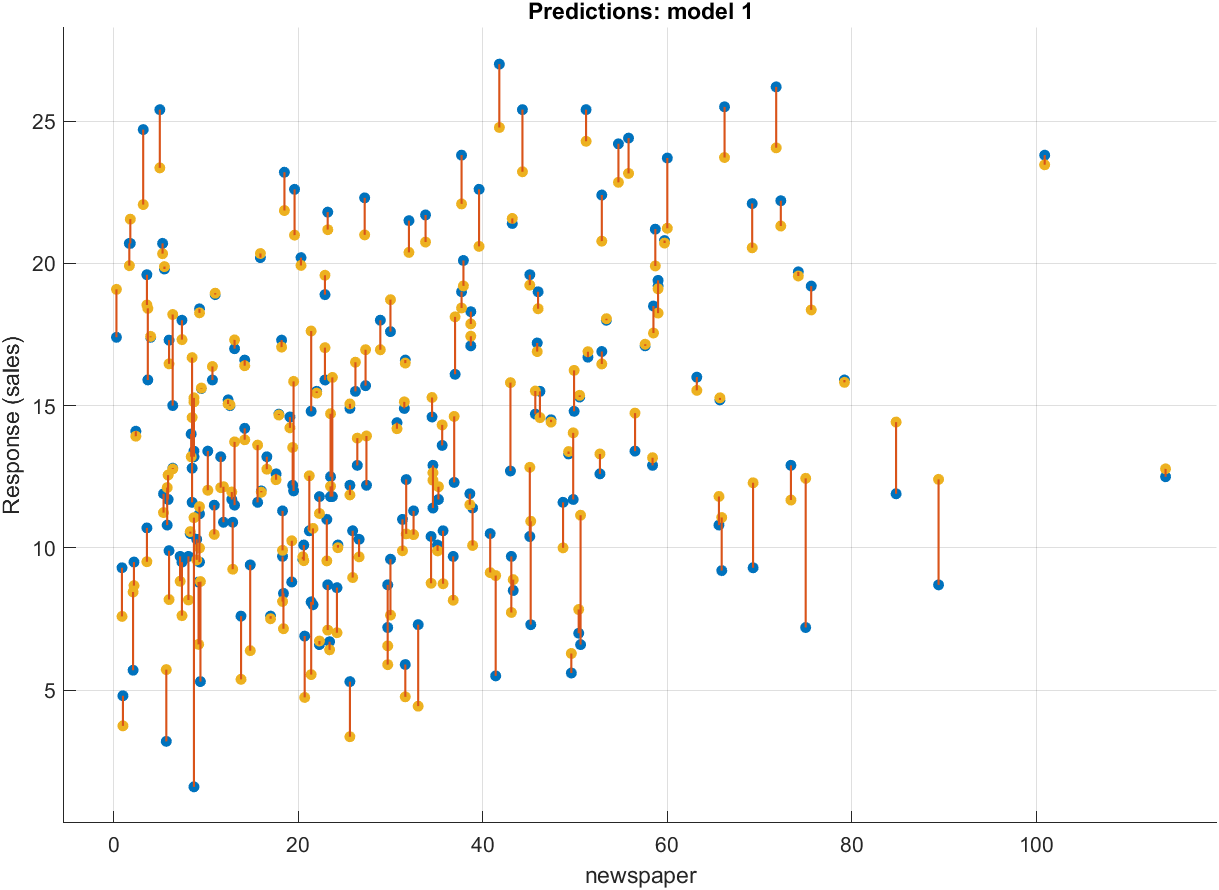
TV için;



**Radio için;**

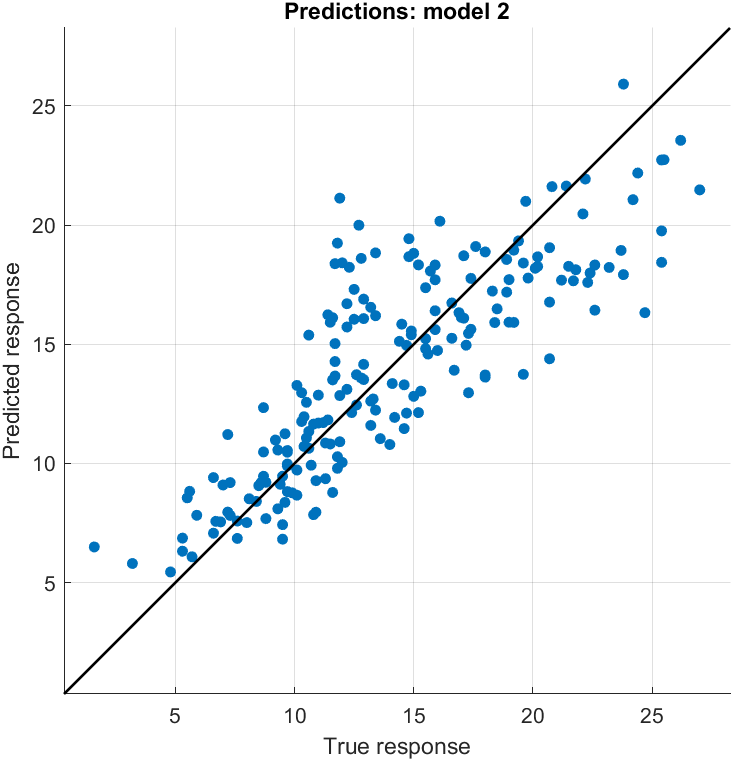


**Newspaper için;**

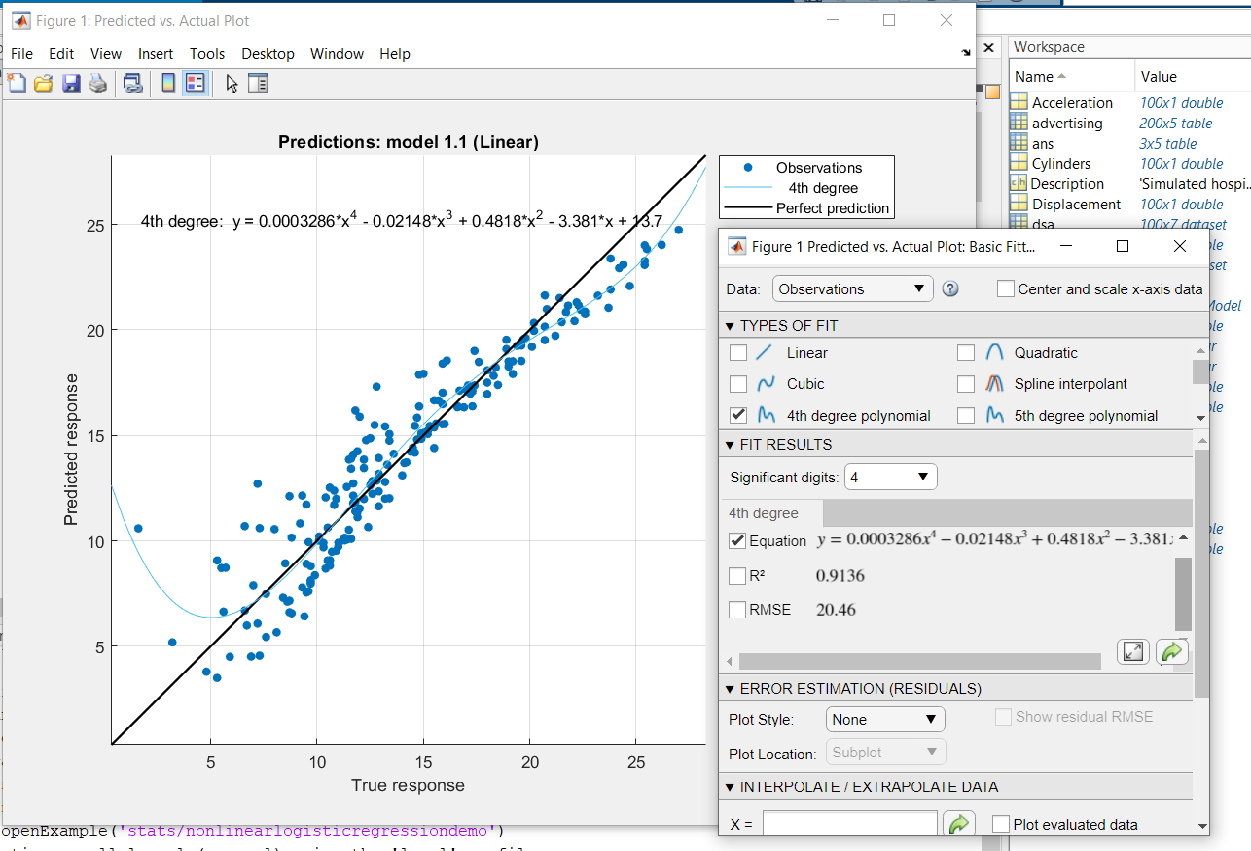
****

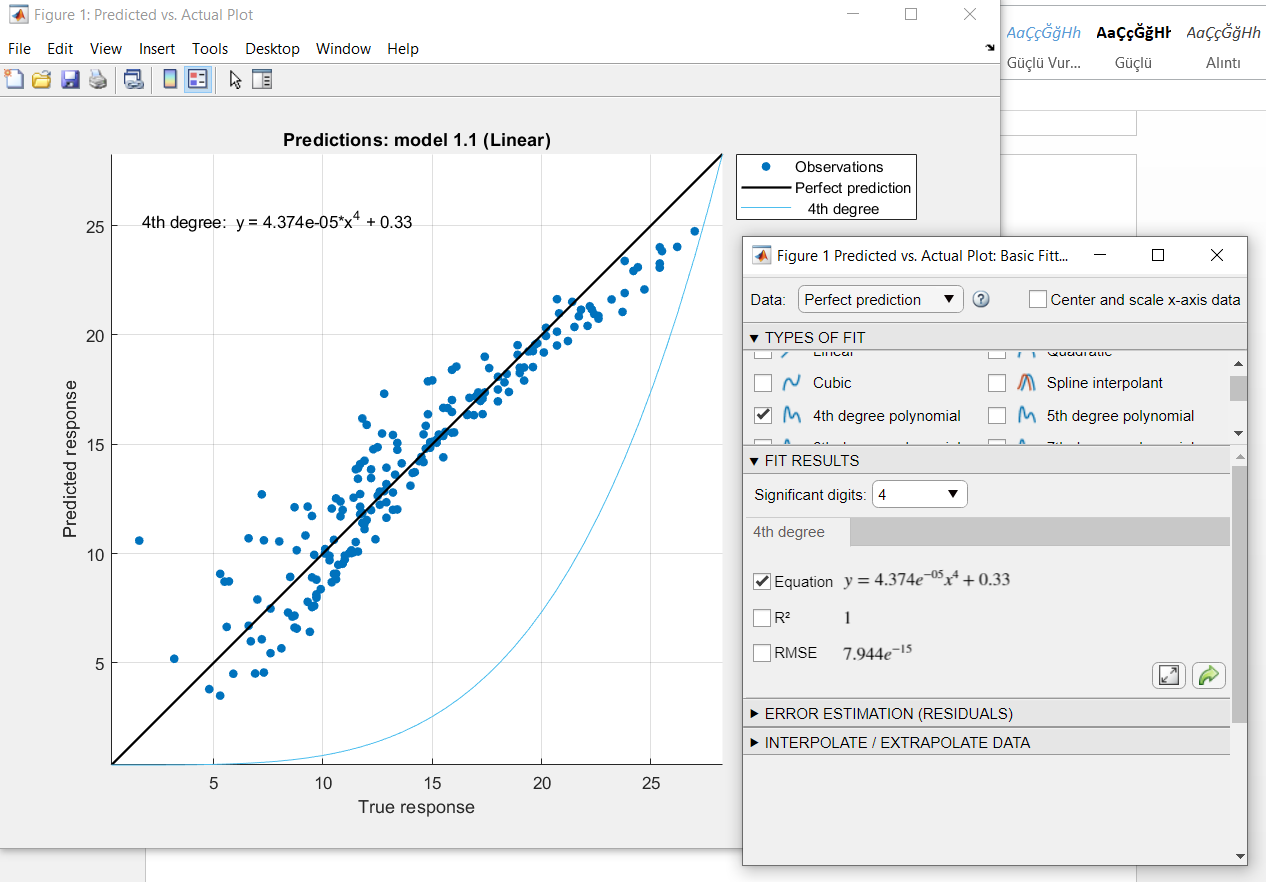
**Linear Regression with PCA**

**RMSE: 2.8989**



**POLYNOMİAL REGRESSİON**

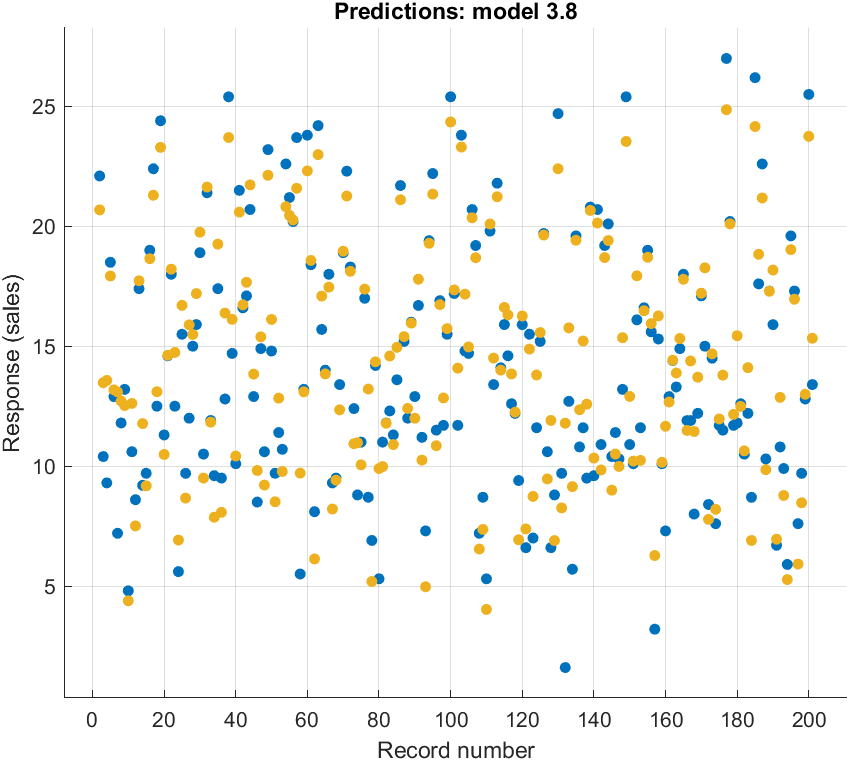
****

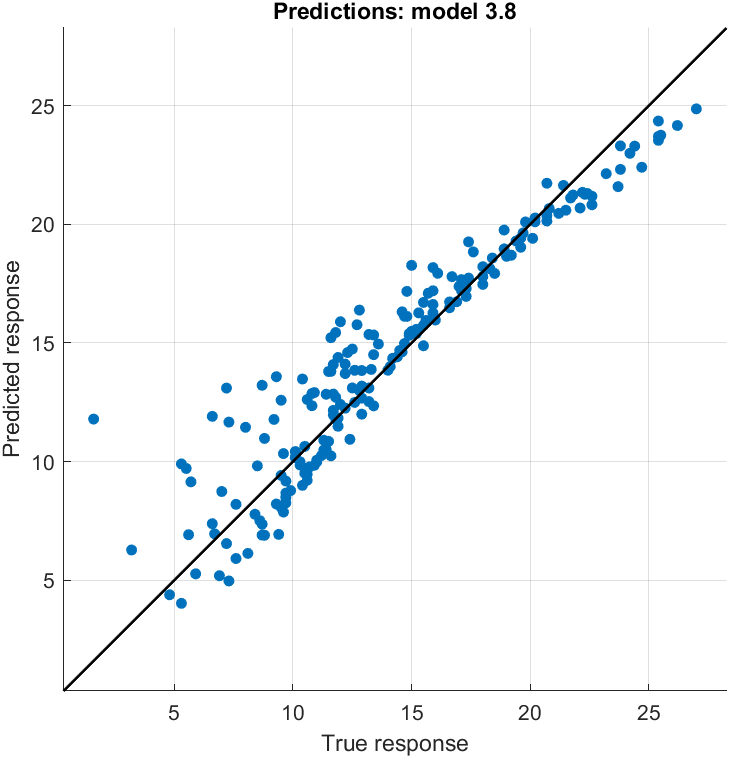
****

**SVM**

**yfit = svmModel.predictFcn(T)**

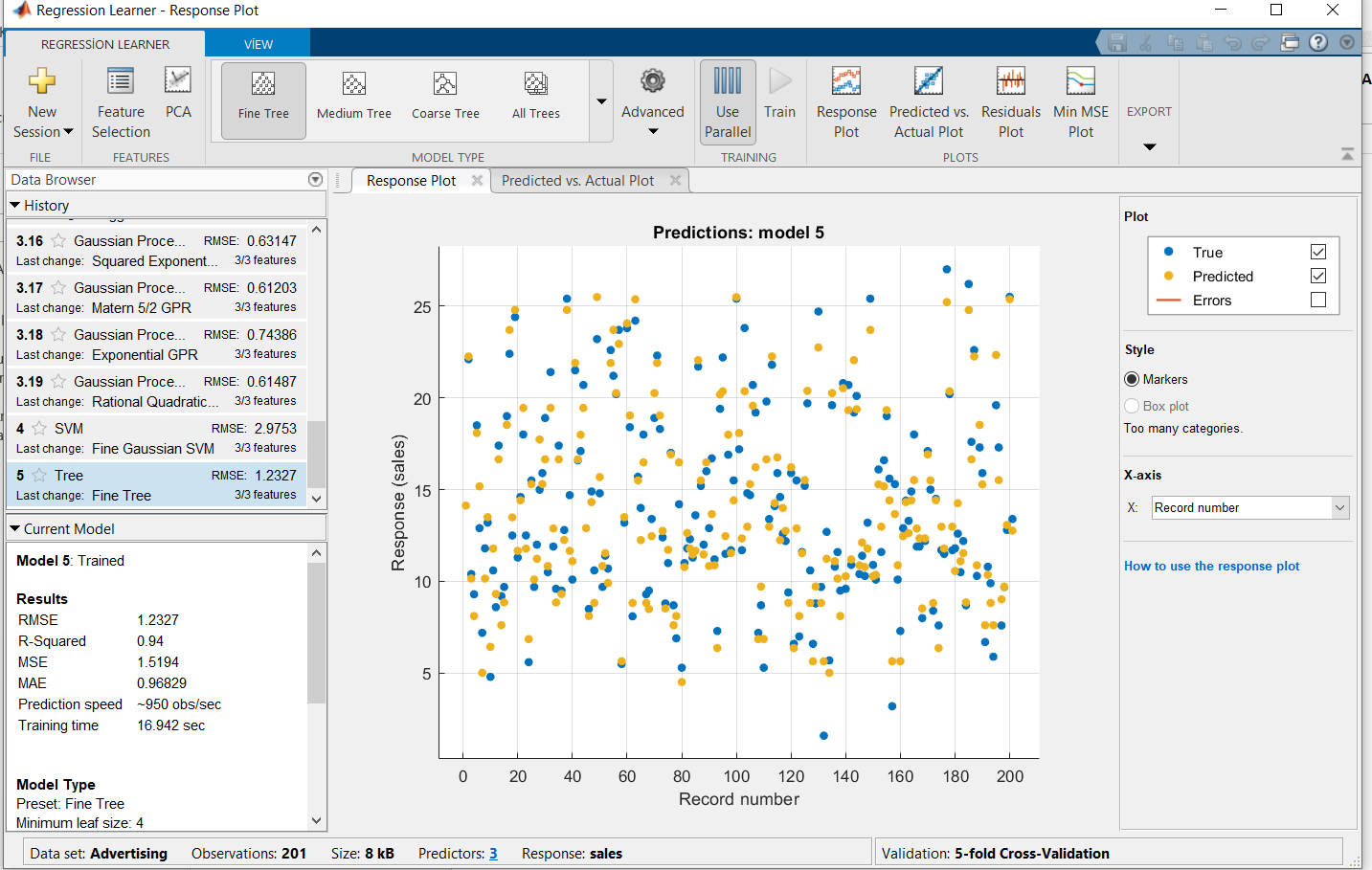
**RMSE:**

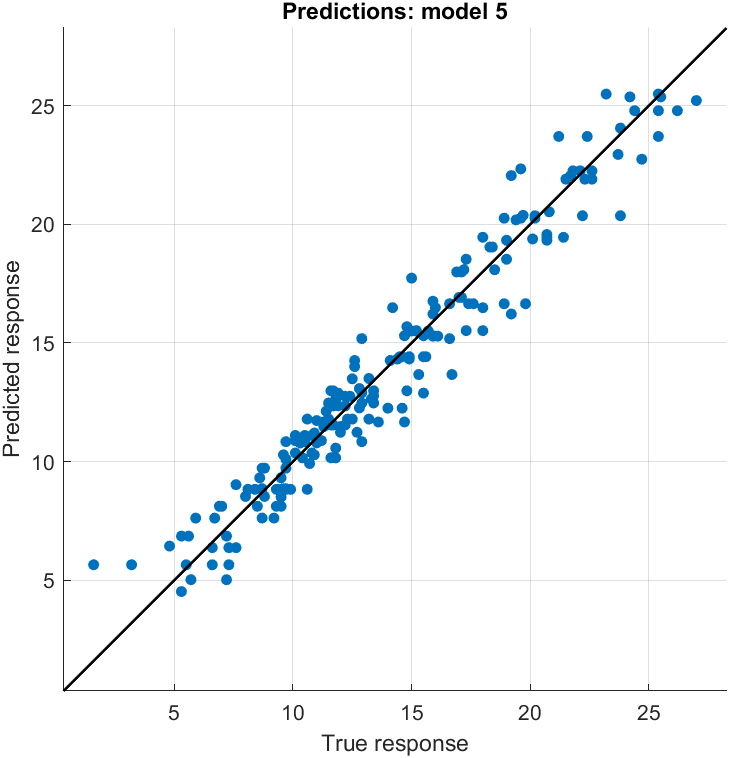
****

****

**KARAR AĞAÇI**

**yfit = treepredictModel.predictFcn(T)**

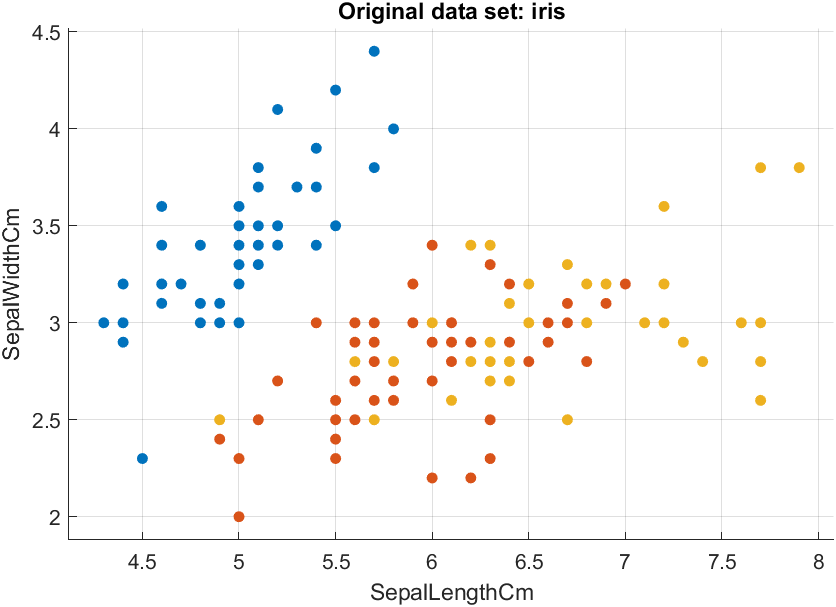
****

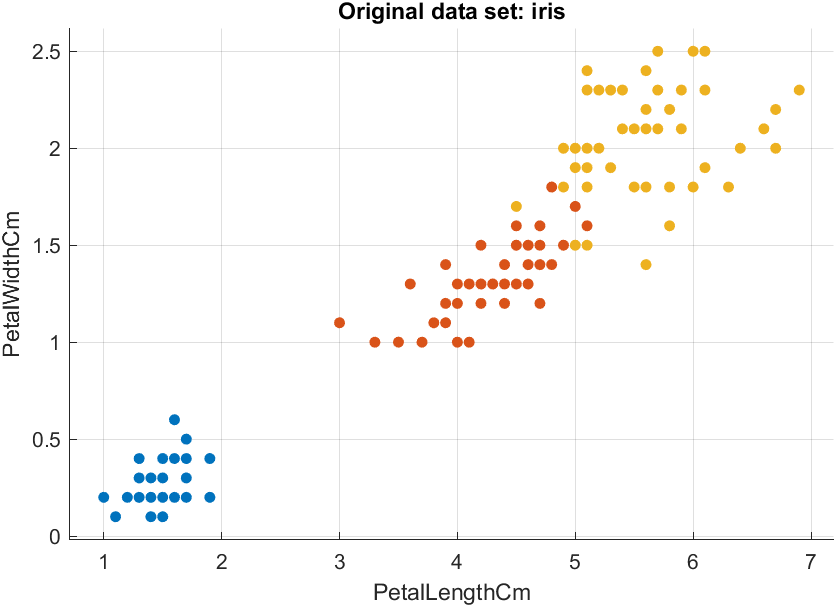
****

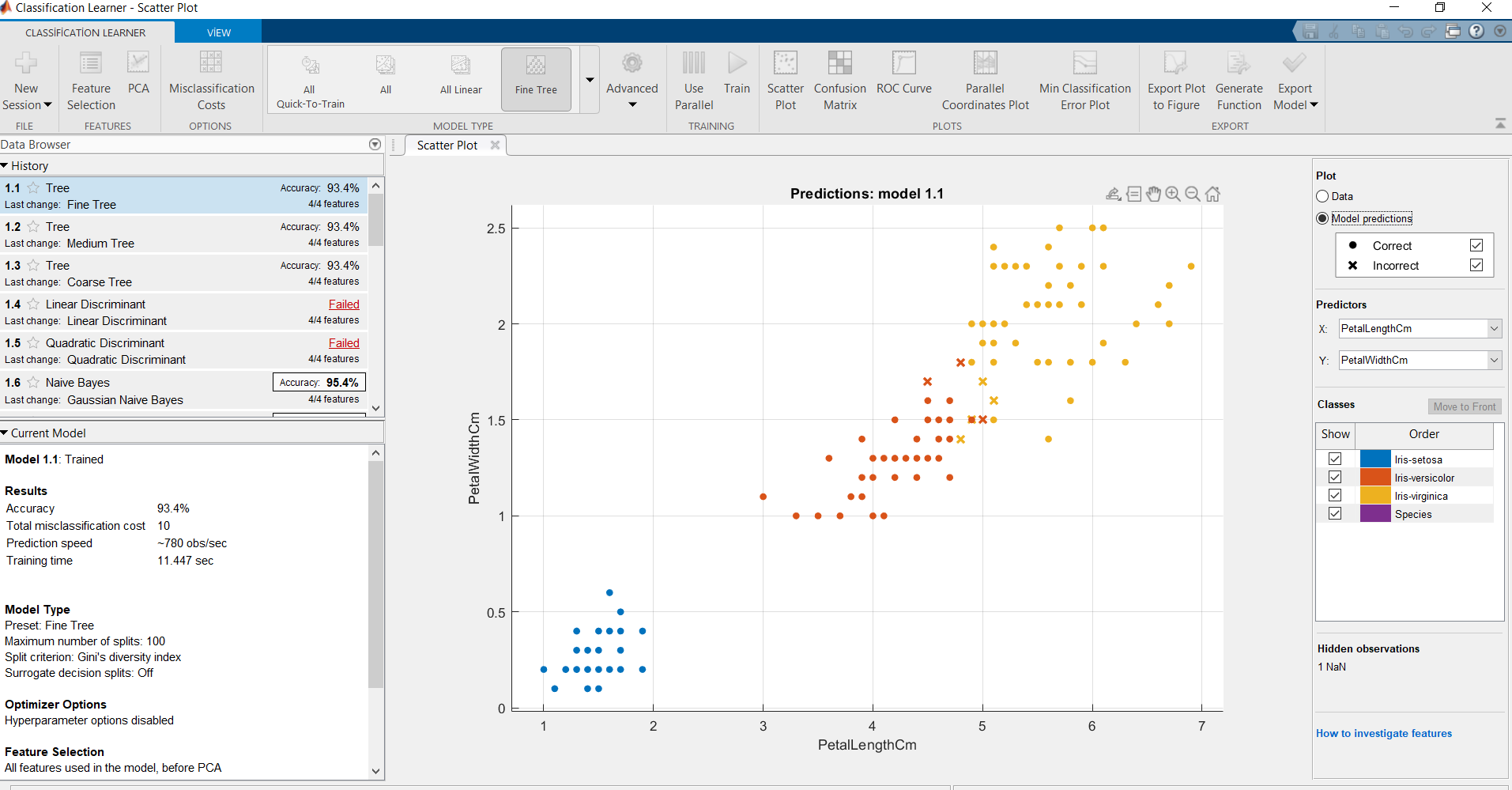
**CLASSİFİCATİON**

**KARAR AĞACI**

**Data Set is iris.csv**

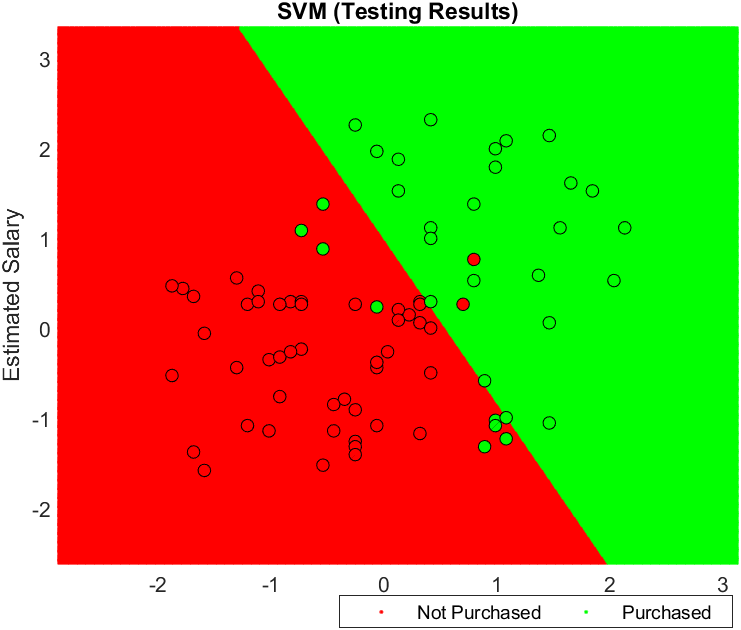
****

****

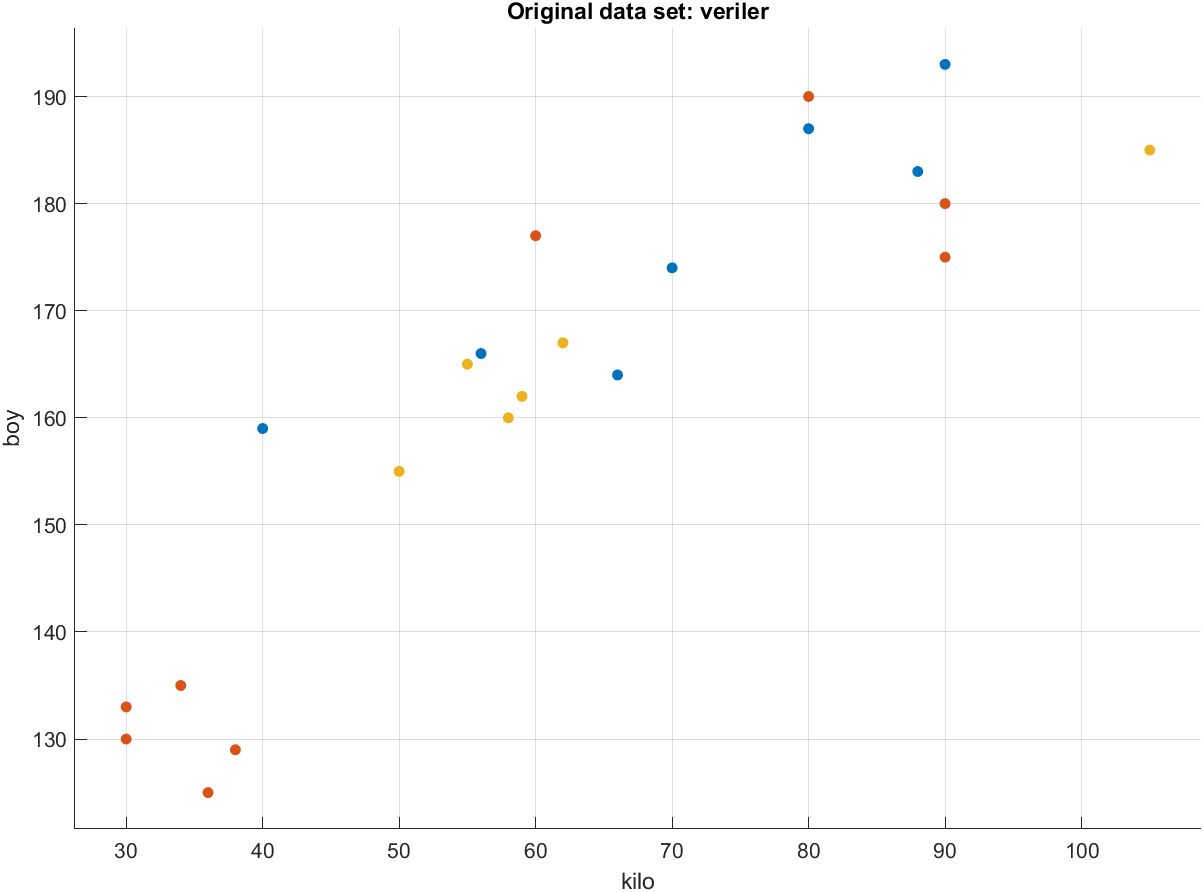
****

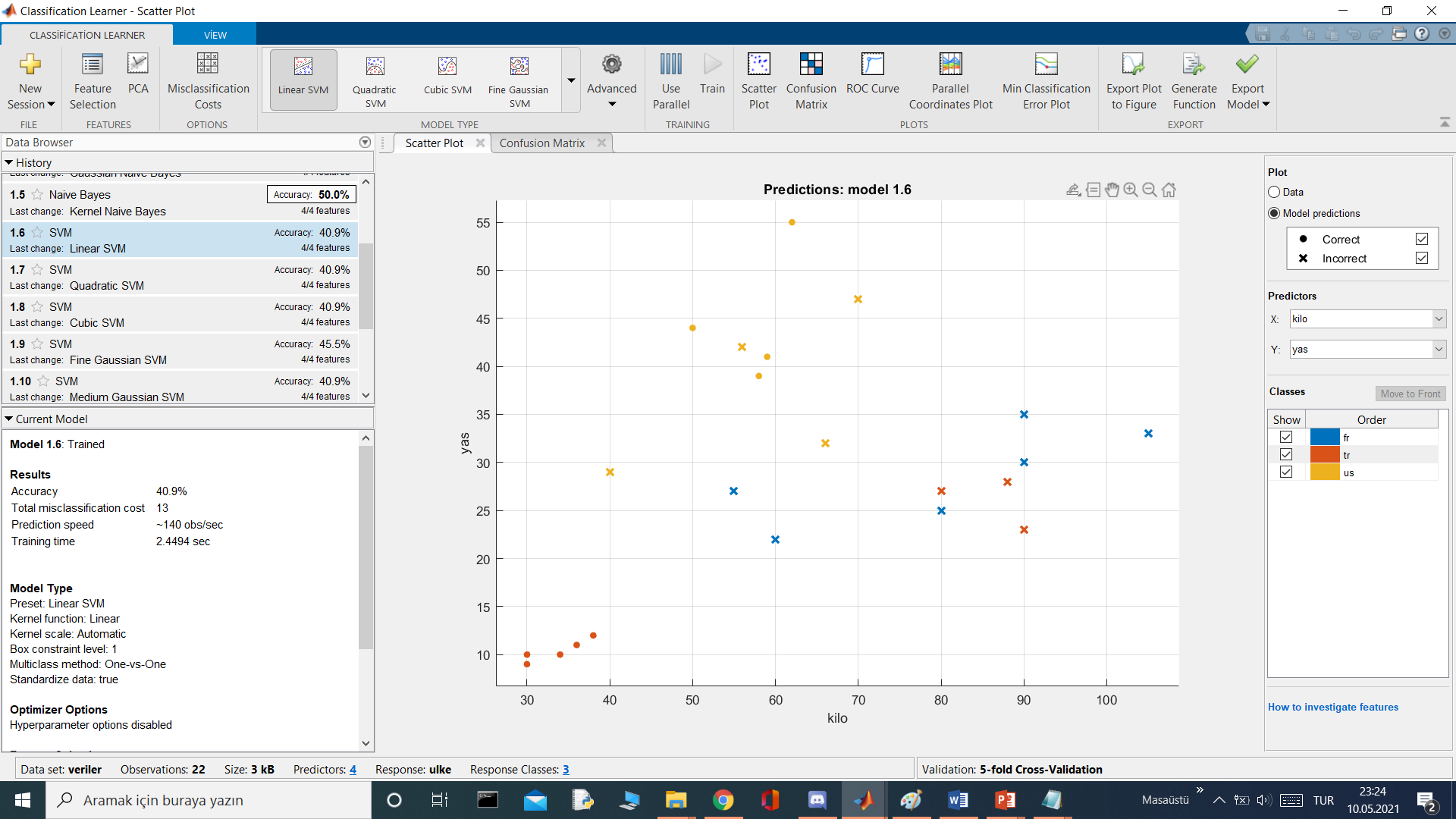
**SVM sınıflandırma**

****

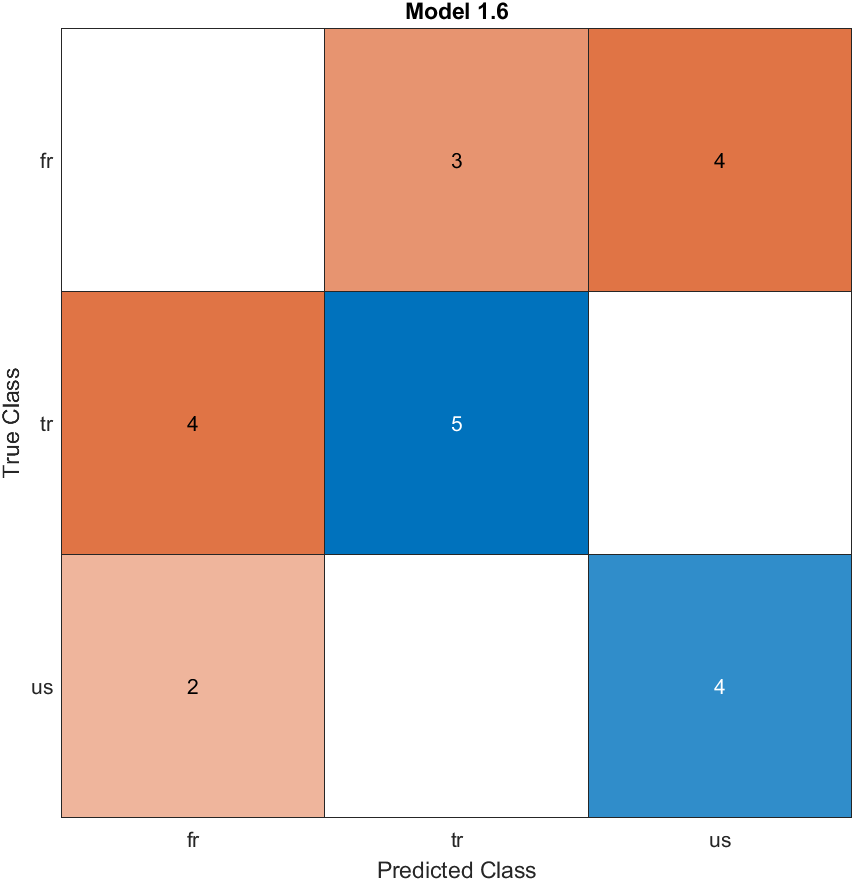
****

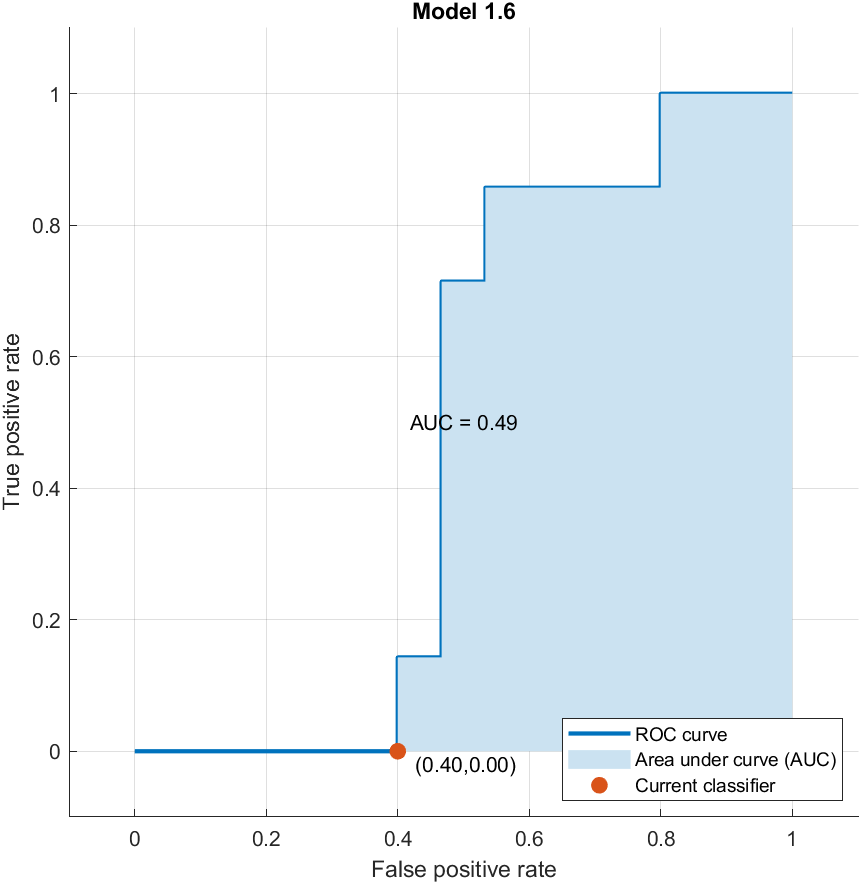
**Dataset: veriler.csv**

****

****

**Confusion Matrix**

****

****