# 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.

$$y' = b + w_1 x_1 + w_2 x_2 + w_3 x_3$$

**[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

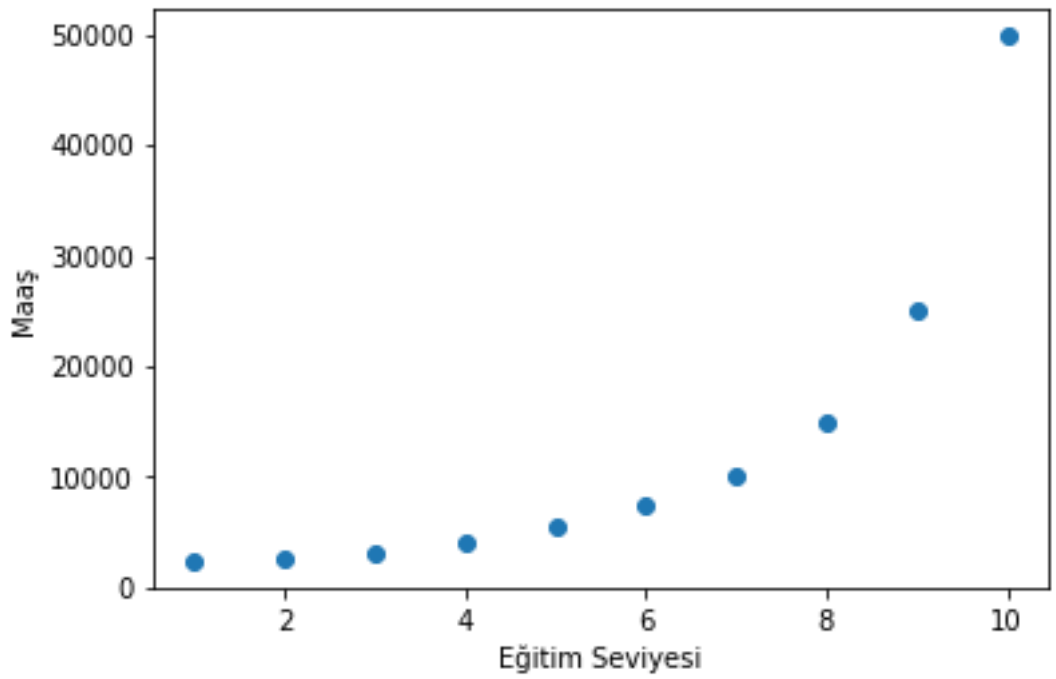

| Index | unvan | Egitim Seviyesi | maas |
|---|---|---|---|
| 0 | Cayci | 1 | 2250 |
| 1 | Sekreter | 2 | 2500 |
| 2 | Uzman Yardimcisi | 3 | 3000 |
| 3 | Uzman | 4 | 4000 |
| 4 | Proje Yoneticisi | 5 | 5500 |
| 5 | Sef | 6 | 7500 |
| 6 | Mudur | 7 | 10000 |
| 7 | Direktor | 8 | 15000 |
| 8 | C-level | 9 | 25000 |
| 9 | CEO | 10 | 50000 |

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)

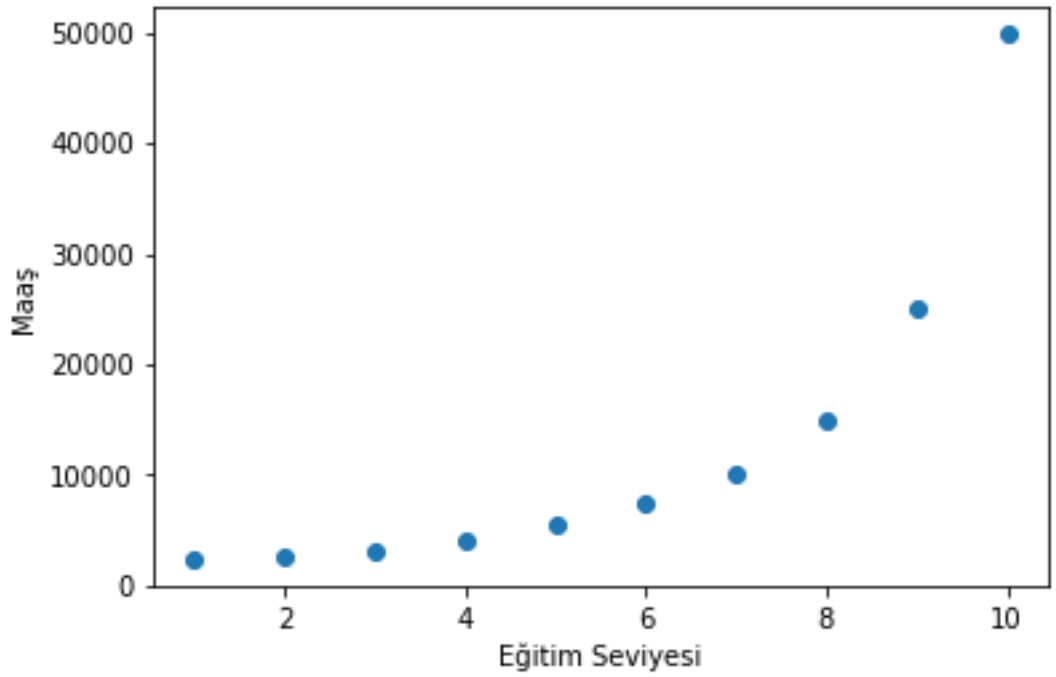
**#GÖRSELLEŞTİRME**

plt.scatter(X,Y)

plt.ylabel("Maaş")

plt.xlabel("Eğitim Seviyesi")

plt.show()

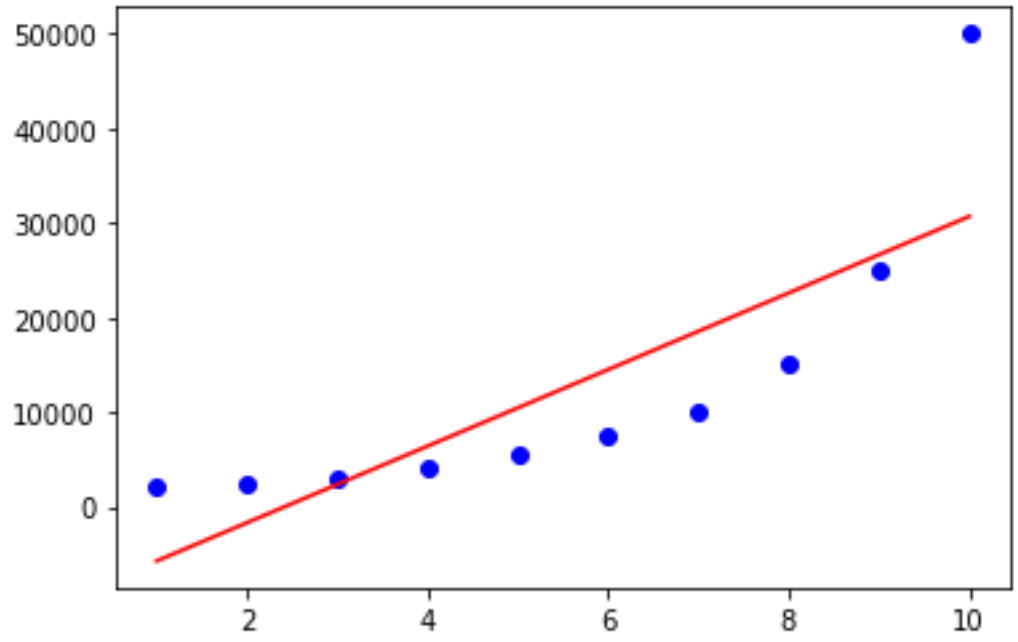



**#Linear Regresyon**

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

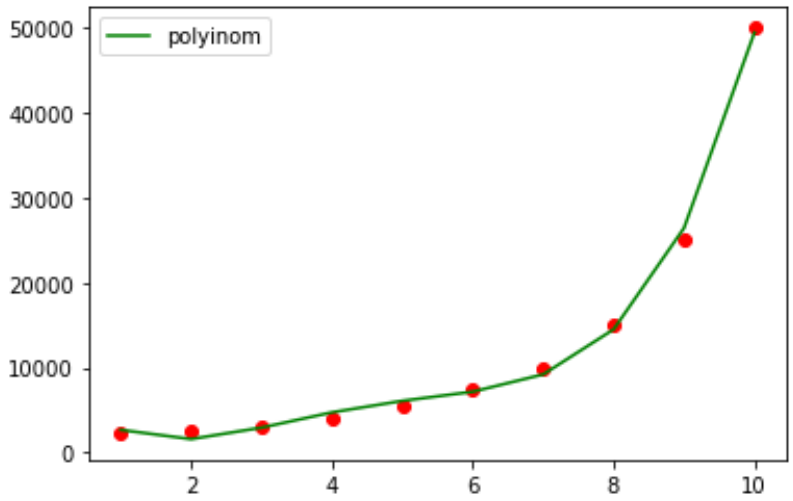
plt.plot(x,linear_reg.predict(X),color="red")

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)
```

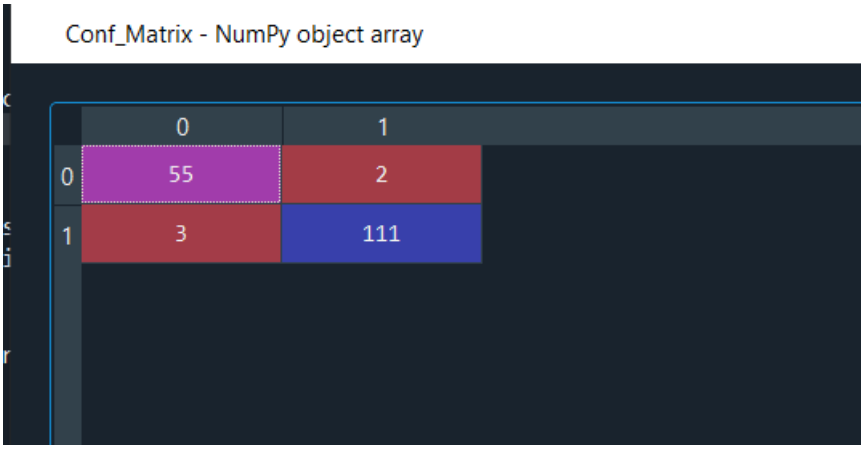
**OUTPUT:**

Text editor - Class_rep

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.95 | 0.96 | 0.96 | 57 |
| 1 | 0.98 | 0.97 | 0.98 | 114 |
| accuracy | | | 0.97 | 171 |
| macro avg | 0.97 | 0.97 | 0.97 | 171 |
| weighted avg | 0.97 | 0.97 | 0.97 | 171 |

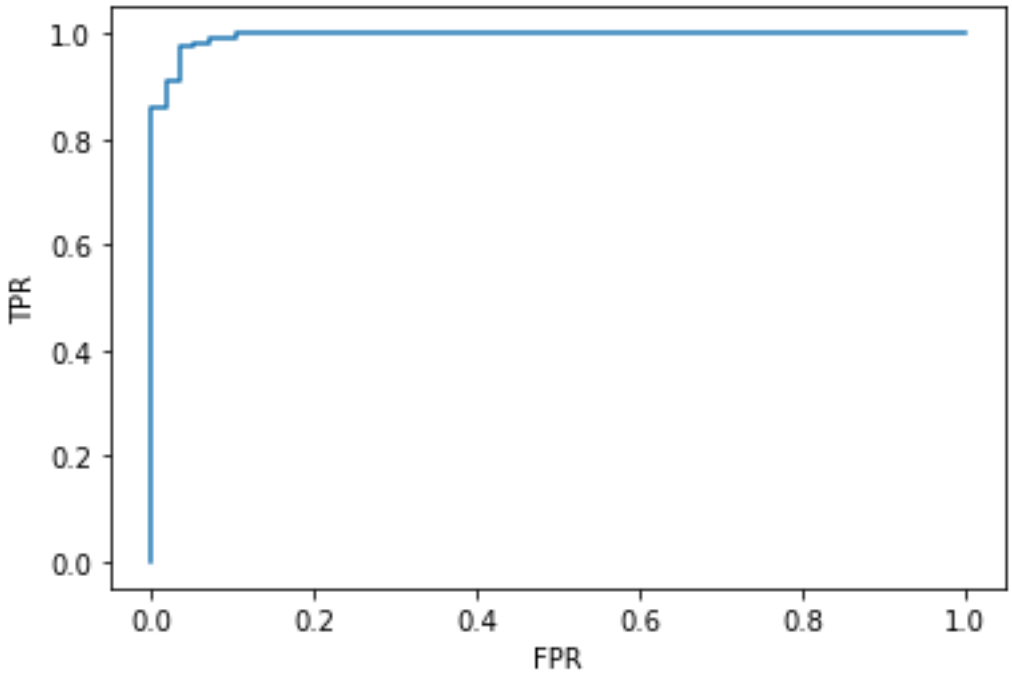
## Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | True Positives (TPs) | False Positives (FPs) |
| Predicted Negative (0) | False Negatives (FNs) | True Negatives (TNs) |




**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]]

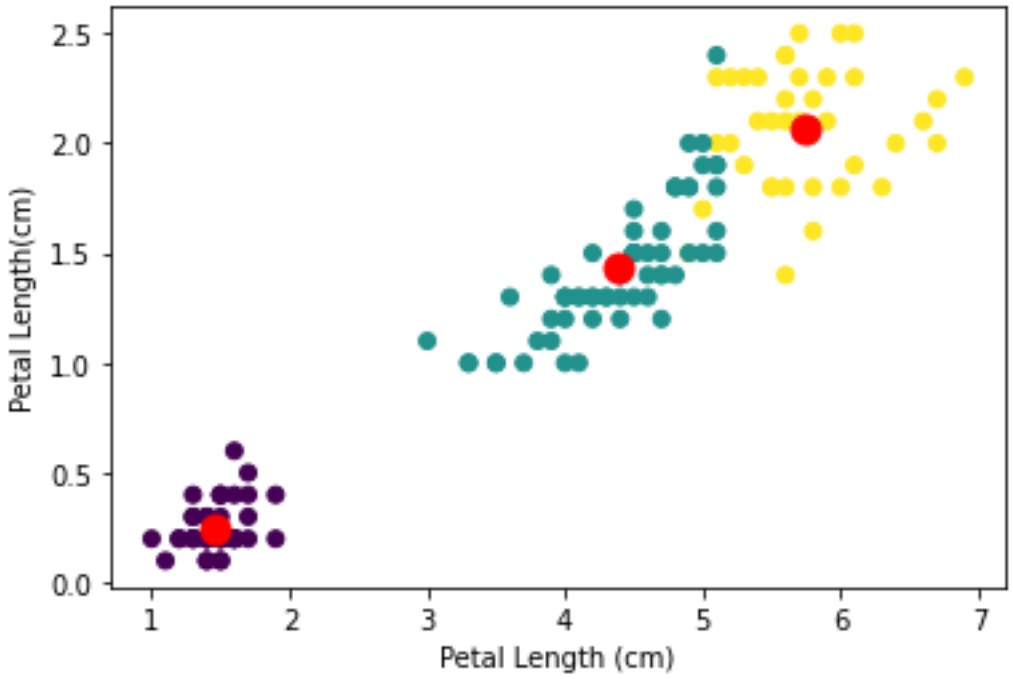
**6)K—MEANS**

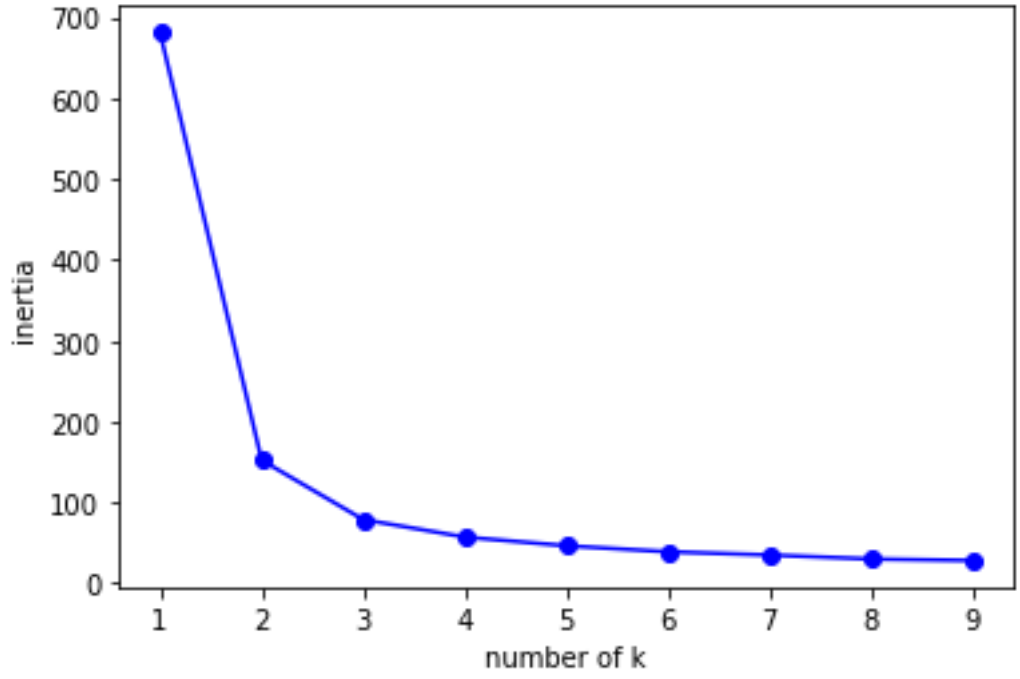
**K-Means Algoritması**, etiketlenmemiş verileri, yani tanımlanmış kategoriler veya gruplar içermeyen verileri kategorilere ayırmak için kullanılan denetlenmeyen bir öğrenme türüdür. Algoritma, K değişkeni tarafından temsil edilen grupların sayısıyla veri içindeki grupları bularak çalışır. Ardından, 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()
```


Petal Length(cm)
Petal Length (cm)
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7


inertia
number of k
0
100
200
300
400
500
600
700
1
2
3
4
5
6
7
8
9

## 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,  0.9572
0244,
         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_laye
r_sizes': (100, 100), 'solver': 'adam'}
mlpc_tuned = MLPClassifier(activation = "logistic", alpha = 0.1, hidden_lay
er_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
```

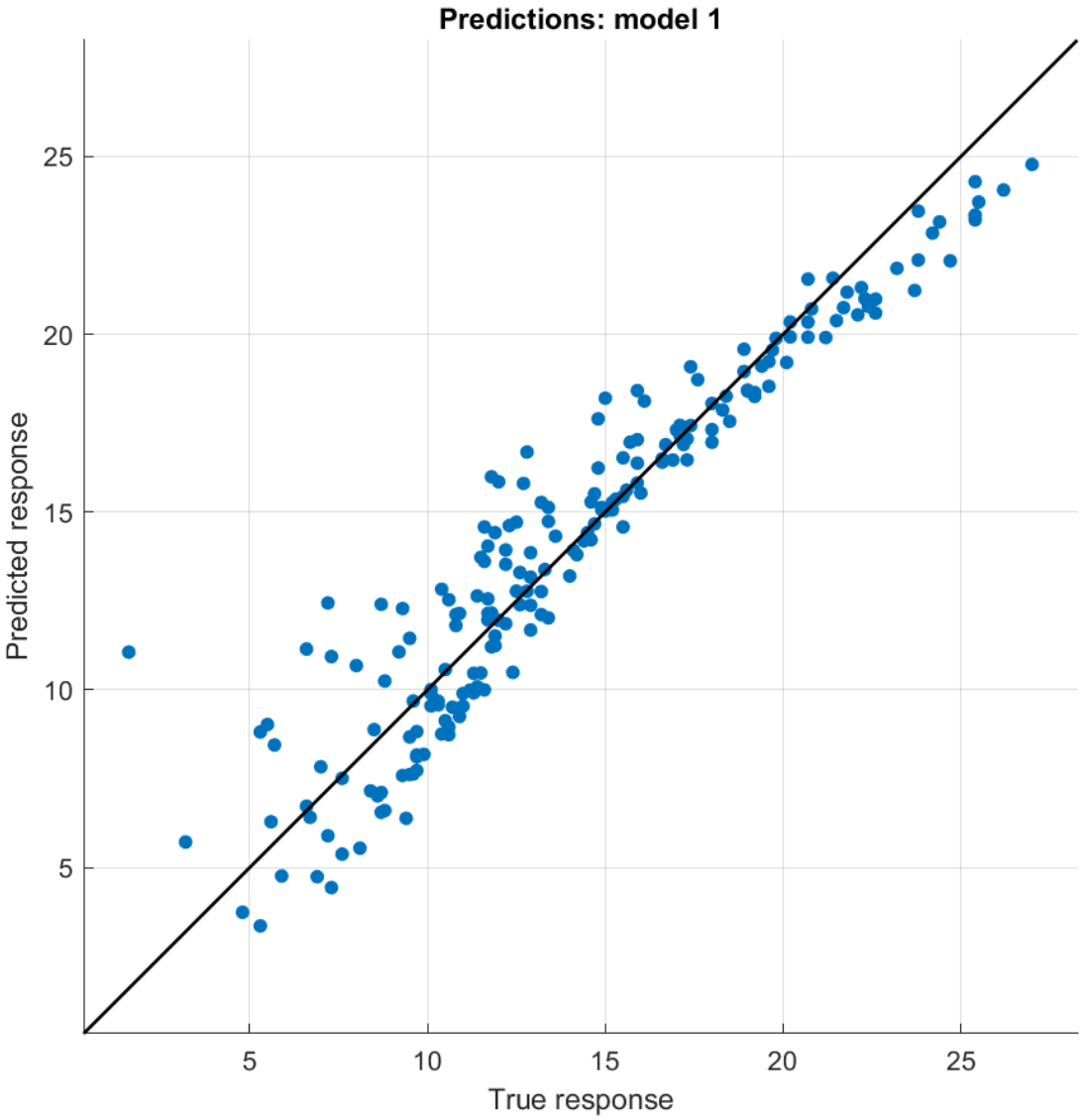
## MAKİNE ÖĞRENMESİ YÖNTEMLERİNİN MATLAB UYGULAMALARI

Statistics and Machine Learning Toolbox kullanılarak makine öğrenmesi algoritmaları uygulandı

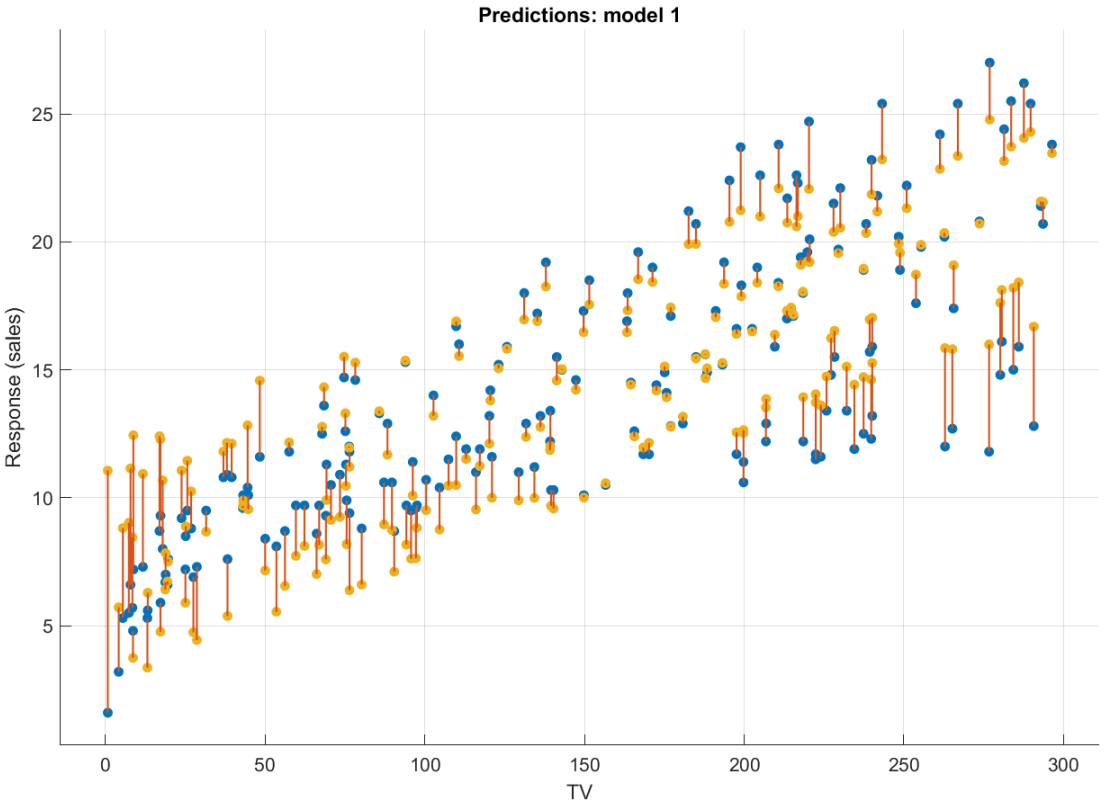
## 1)LİNEAR REGRESSİON

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.

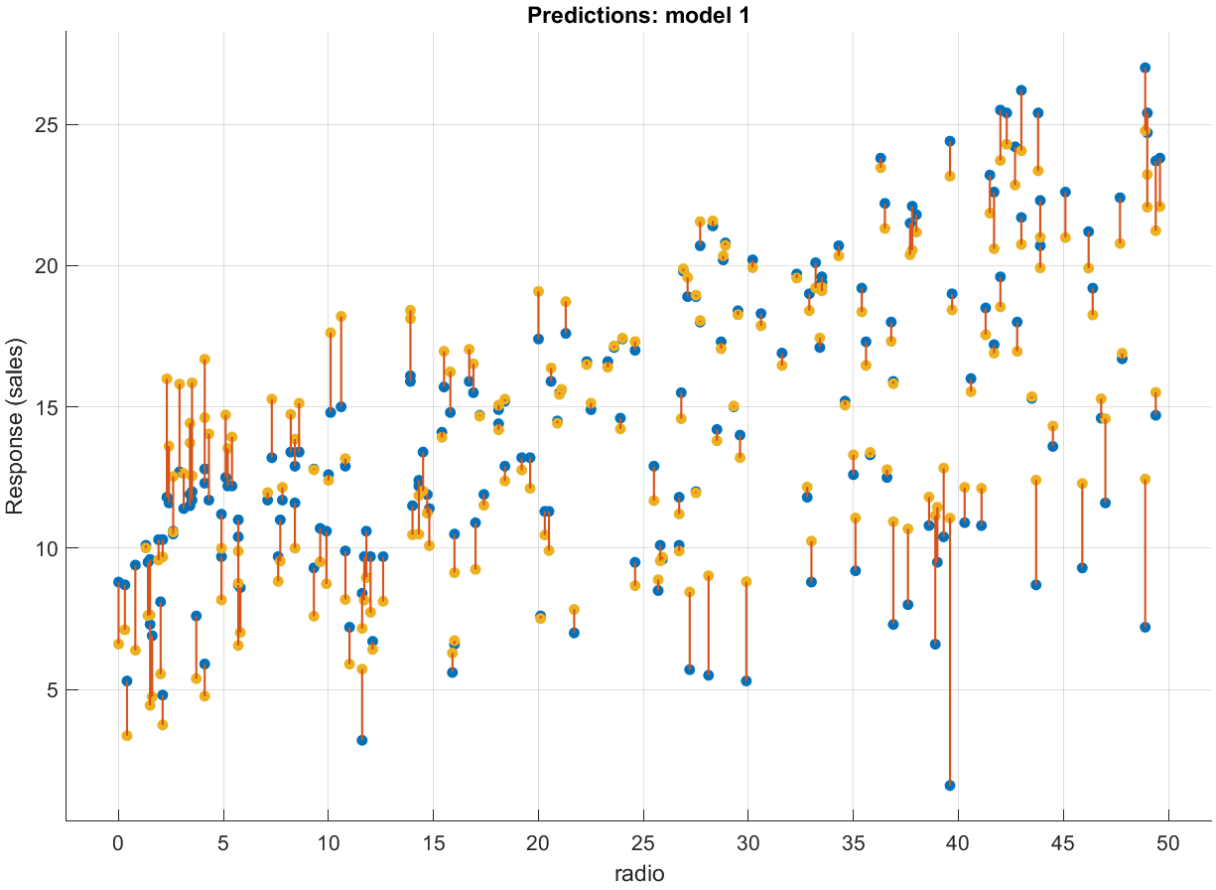
RMSE değeri ne kadar küçük olursa tahminlerimiz o kadar iyidir.


Predictions: model 1
Predicted response
True response
5
10
15
20
25

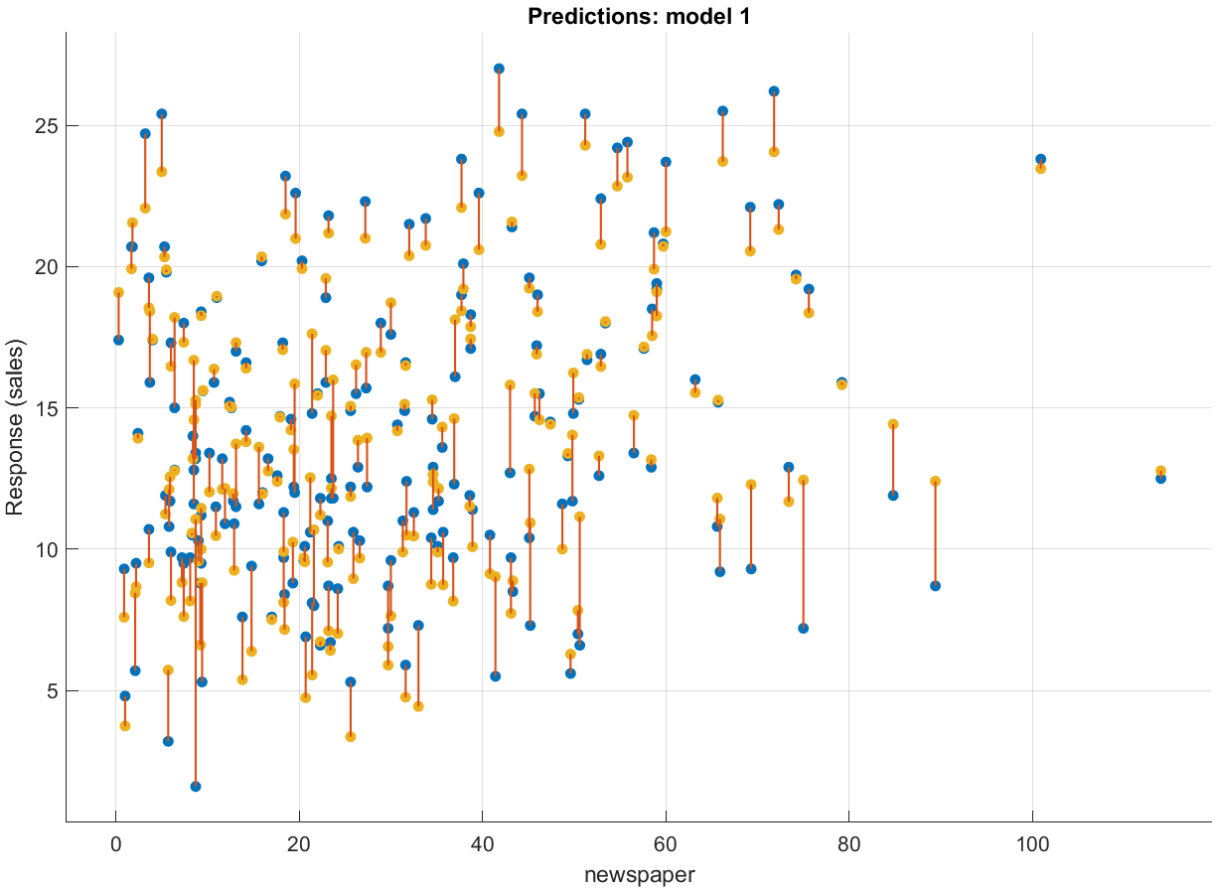
TV için;




**Radio için;**

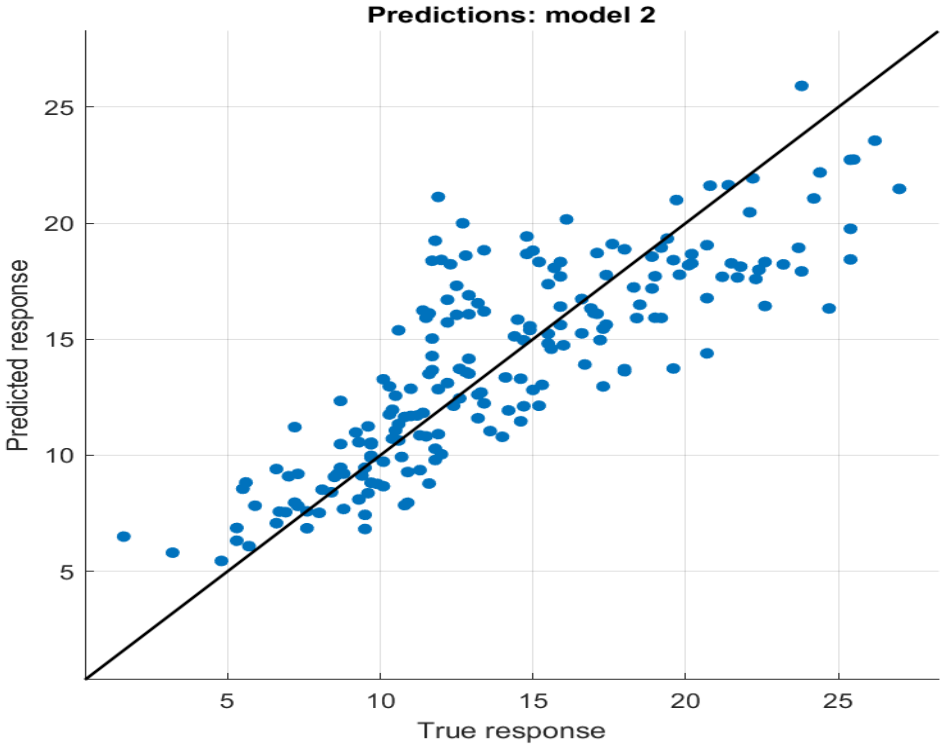



**Newspaper için;**




**Linear Regression with PCA**

**RMSE: 2.8989**




**POLYNOMİAL REGRESSİON**

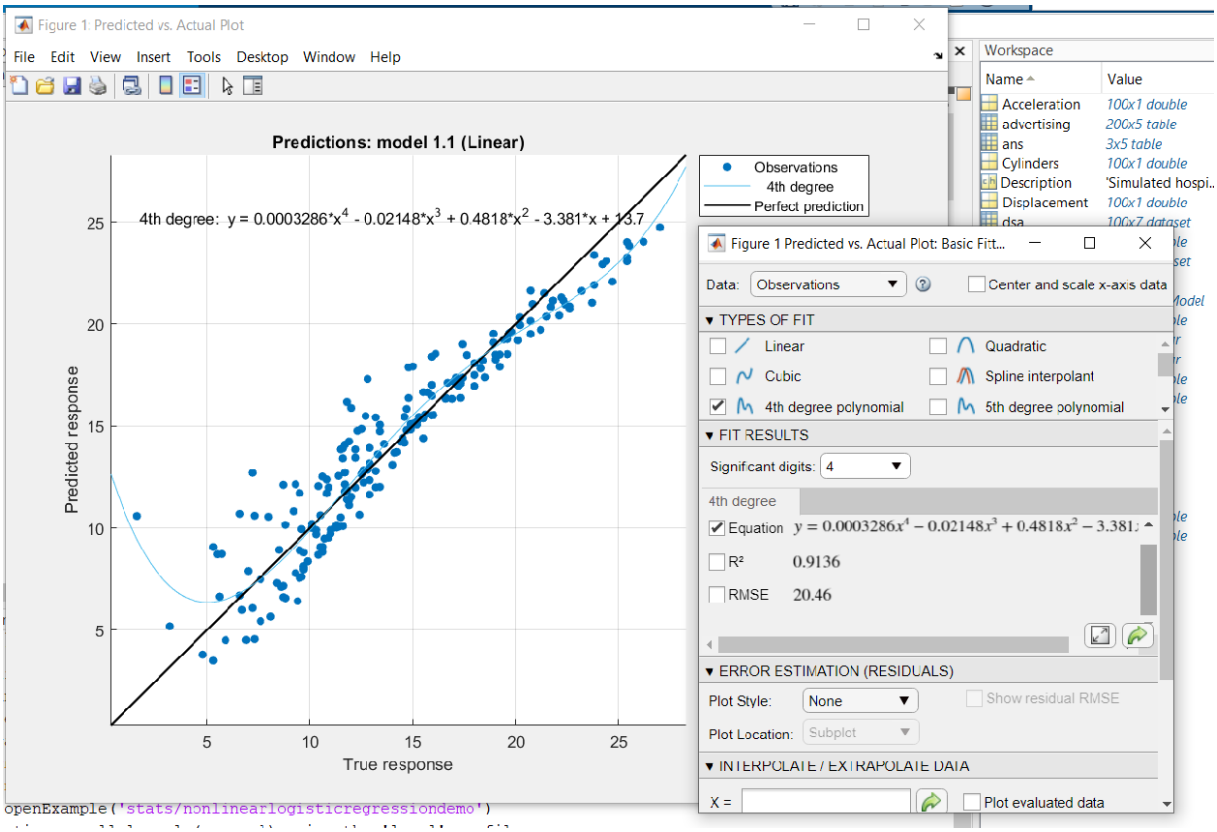


Figure 1: Predicted vs. Actual Plot
File Edit View Insert Tools Desktop Window Help
Predictions: model 1.1 (Linear)
4th degree: y = 0.0003286*x^4 - 0.02148*x^3 + 0.4818*x^2 - 3.381*x + 13.7
Observations
4th degree
Perfect prediction
Predicted response
True response
Workspace
Name
Value
Acceleration 100x1 double
advertising 200x5 table
ans 3x5 table
Cylinders 100x1 double
Description 'Simulated hospi..
Displacement 100x1 double
Figure 1 Predicted vs. Actual Plot: Basic Fitt...
Data: Observations
Center and scale x-axis data
TYPES OF FIT
Linear
Quadratic
Cubic
Spline interpolant
4th degree polynomial
5th degree polynomial
FIT RESULTS
Significant digits: 4
4th degree
Equation y = 0.0003286x^4 − 0.02148x^3 + 0.4818x^2 − 3.381
R²
0.9136
RMSE
20.46
ERROR ESTIMATION (RESIDUALS)
Plot Style: None
Show residual RMSE
Plot Location: Subplot
INTERPOLATE / EXTRAPOLATE DATA
X =
Plot evaluated data
openExample('stats/nonlinearlogisticregressiondemo')

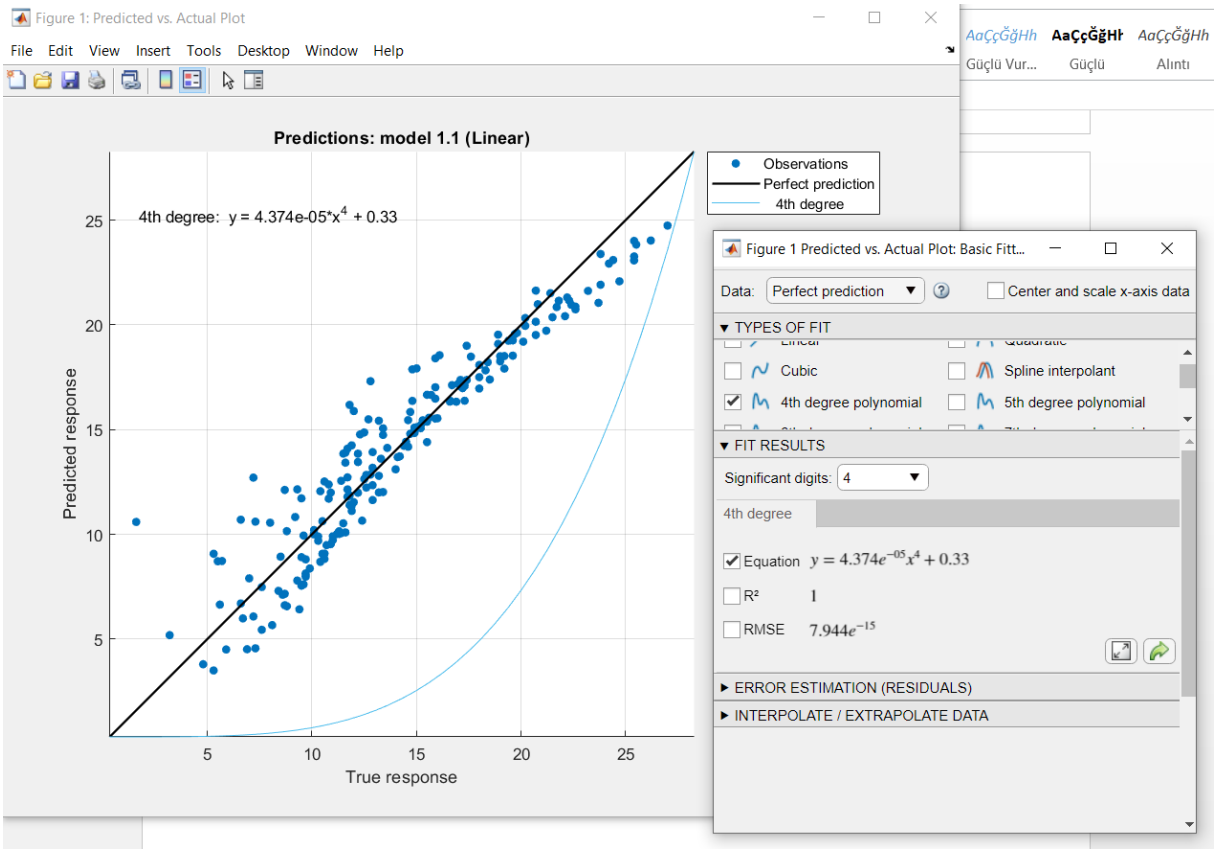
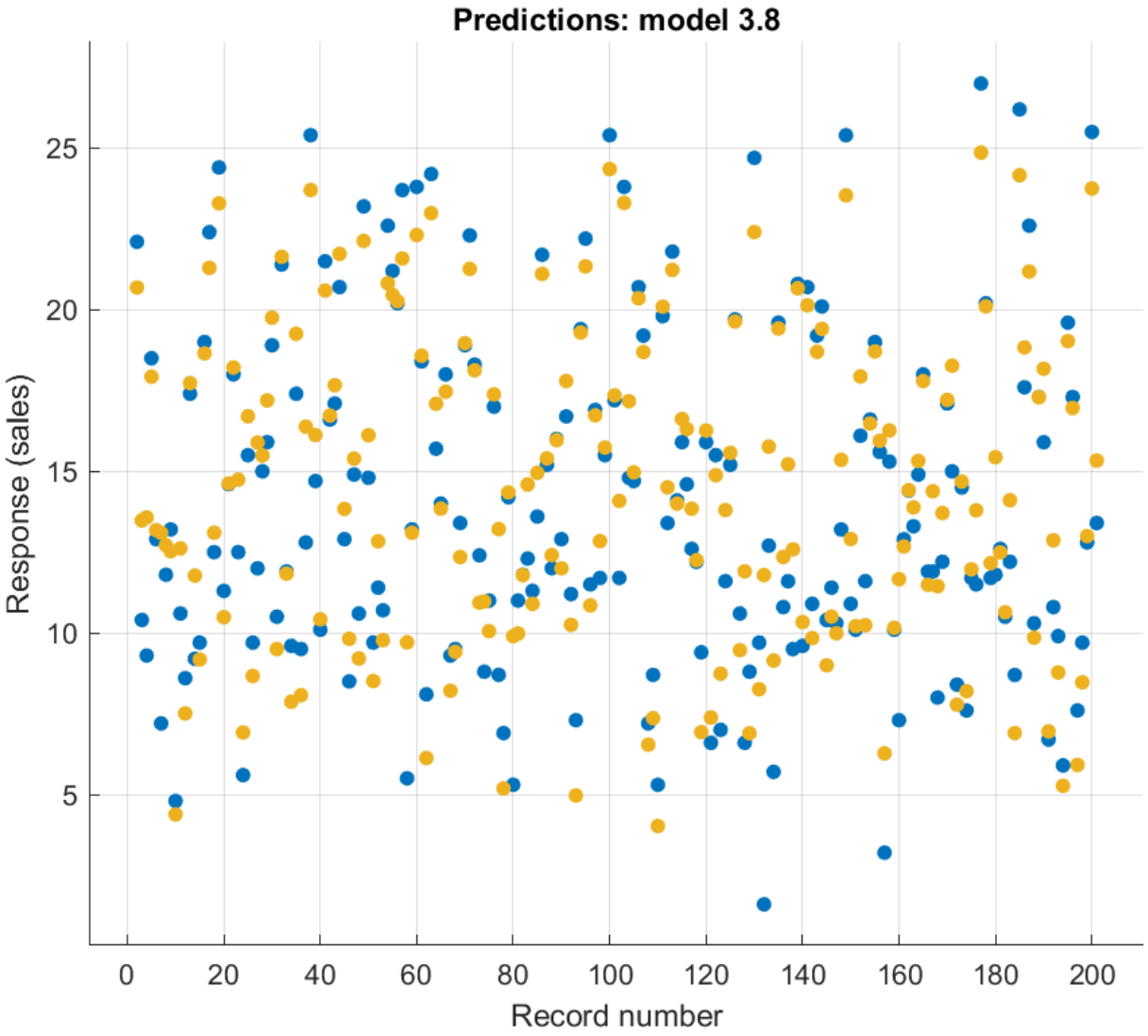


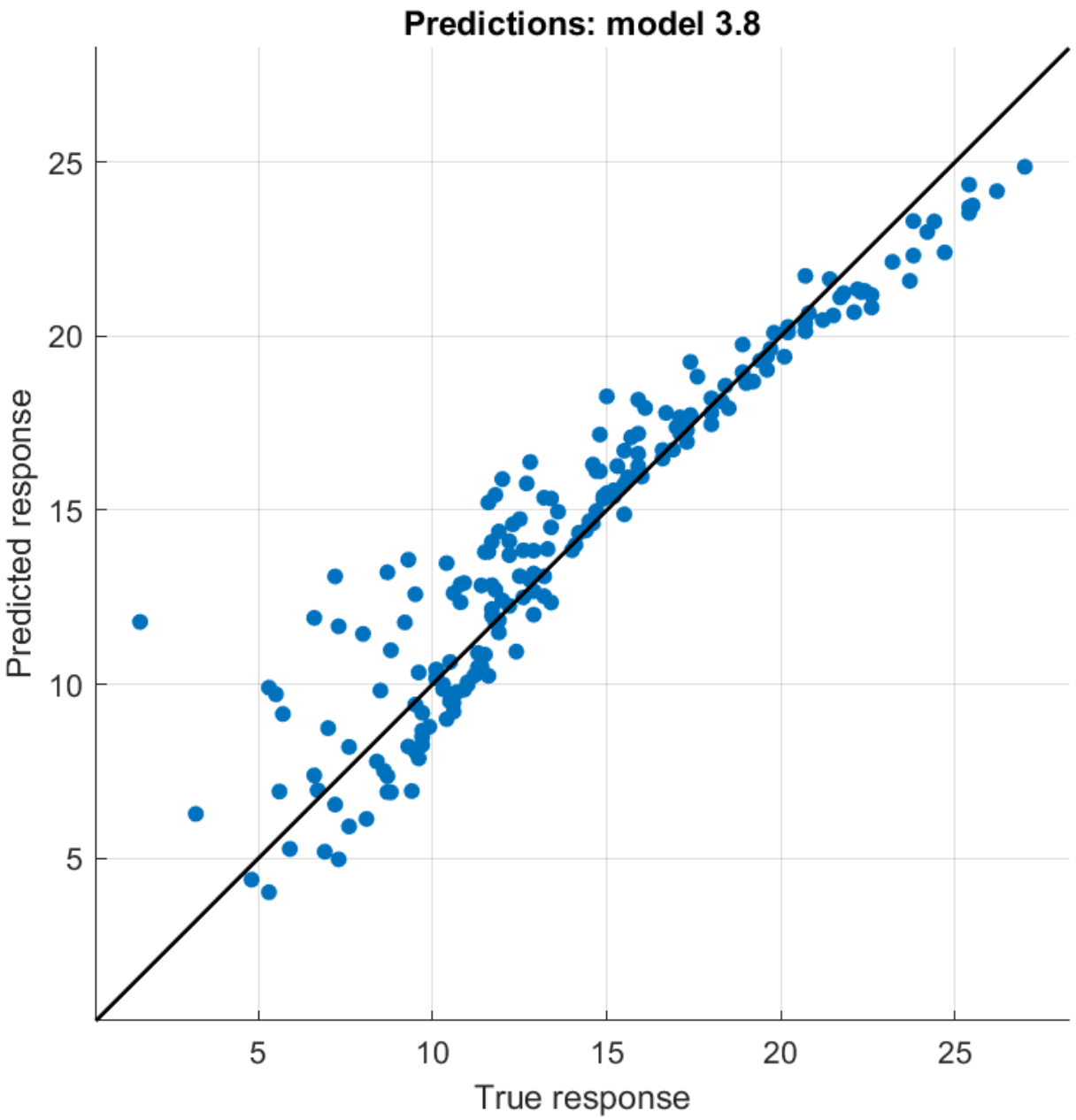
Figure 1: Predicted vs. Actual Plot
File Edit View Insert Tools Desktop Window Help
Predictions: model 1.1 (Linear)
4th degree: y = 4.374e-05*x^4 + 0.33
Observations
Perfect prediction
4th degree
Predicted response
True response
Figure 1 Predicted vs. Actual Plot: Basic Fitt...
Data: Perfect prediction
Center and scale x-axis data
TYPES OF FIT
Cubic
Spline interpolant
4th degree polynomial
5th degree polynomial
FIT RESULTS
Significant digits: 4
4th degree
Equation y = 4.374e^-05 x^4 + 0.33
R²
1
RMSE
7.944e^-15
ERROR ESTIMATION (RESIDUALS)
INTERPOLATE / EXTRAPOLATE DATA

**SVM**

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

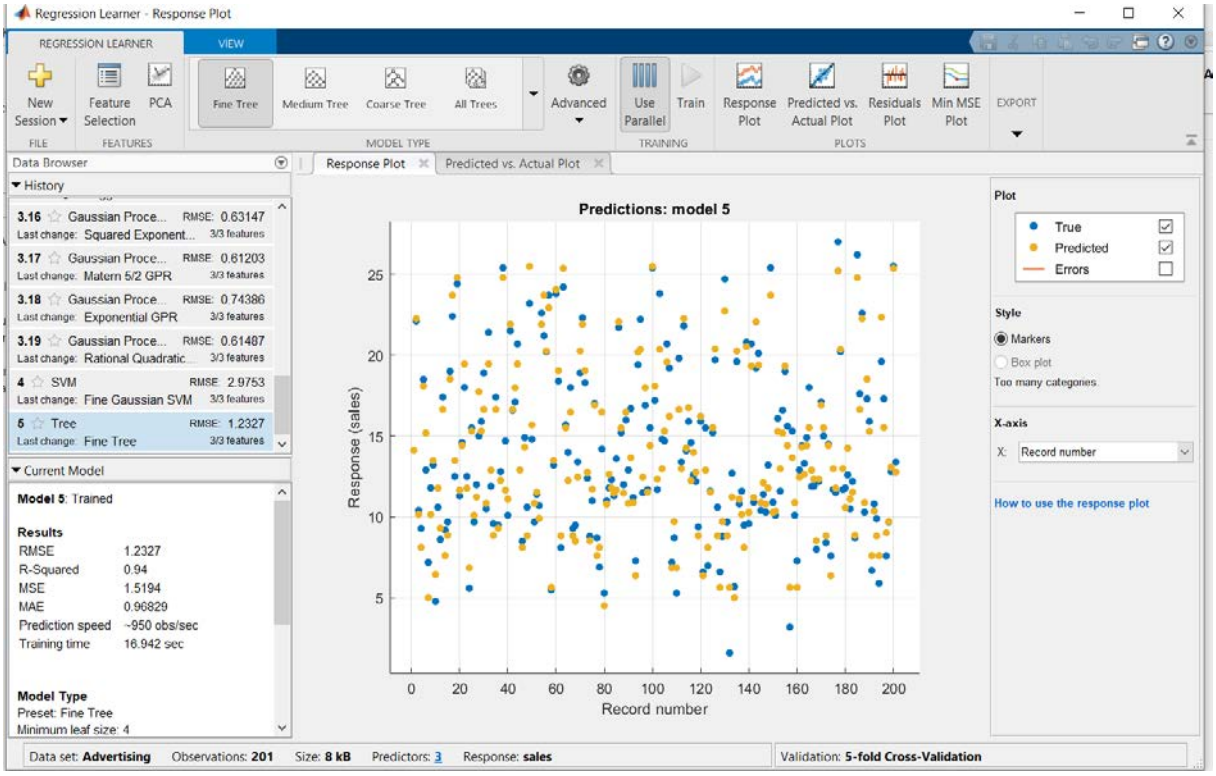
**RMSE:**

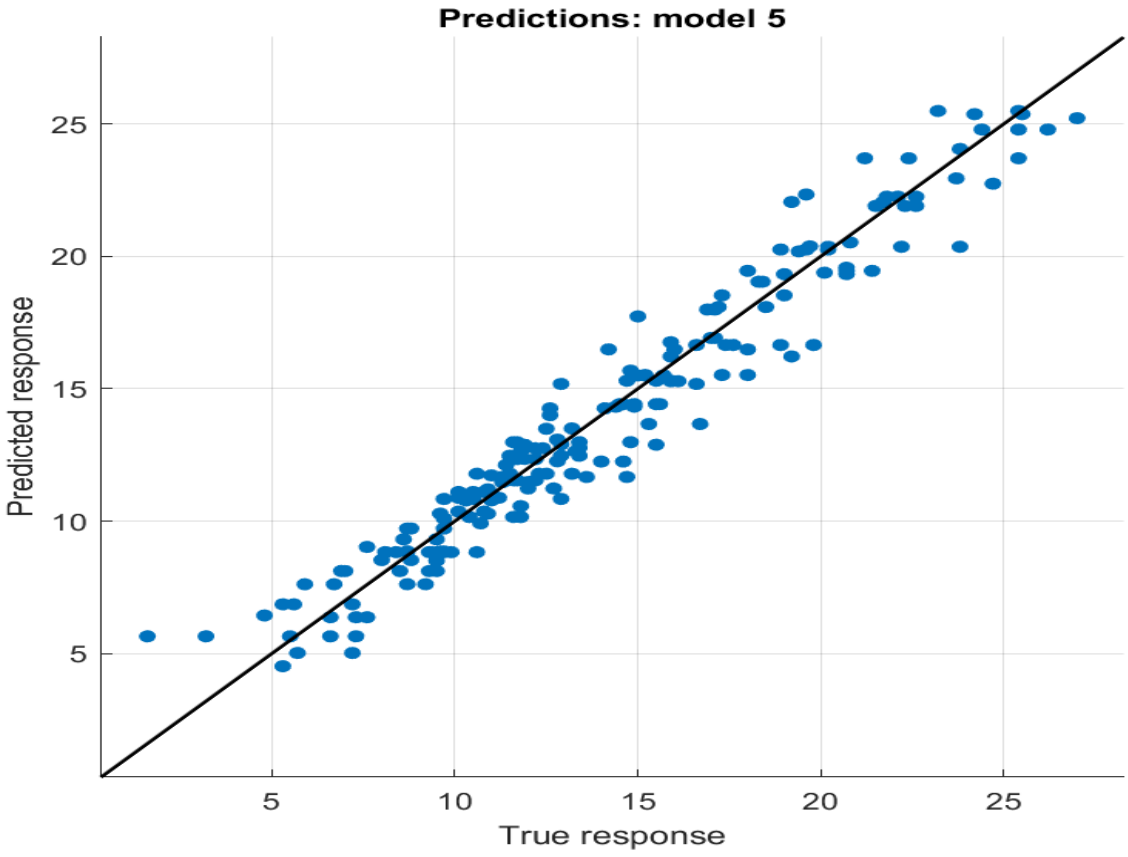



Predictions: model 3.8
Predicted response
True response
5
10
15
20
25
5
10
15
20
25

## KARAR AĞAÇI

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

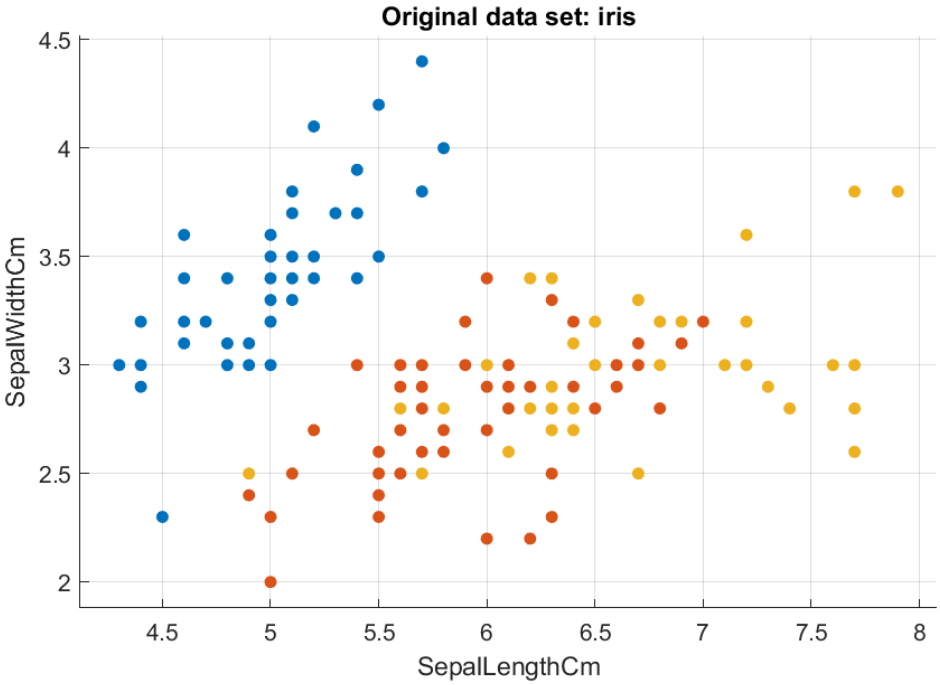


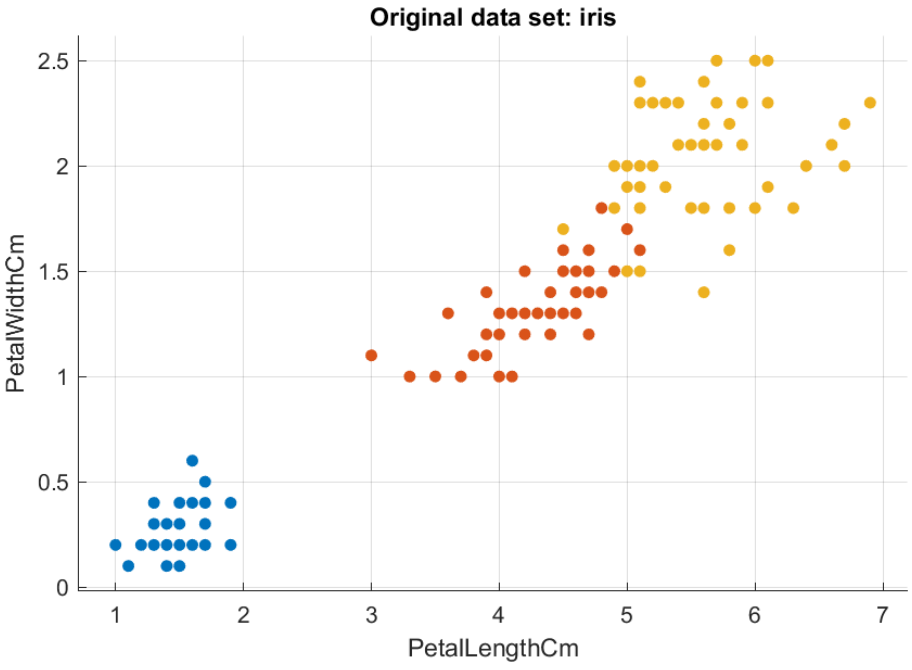



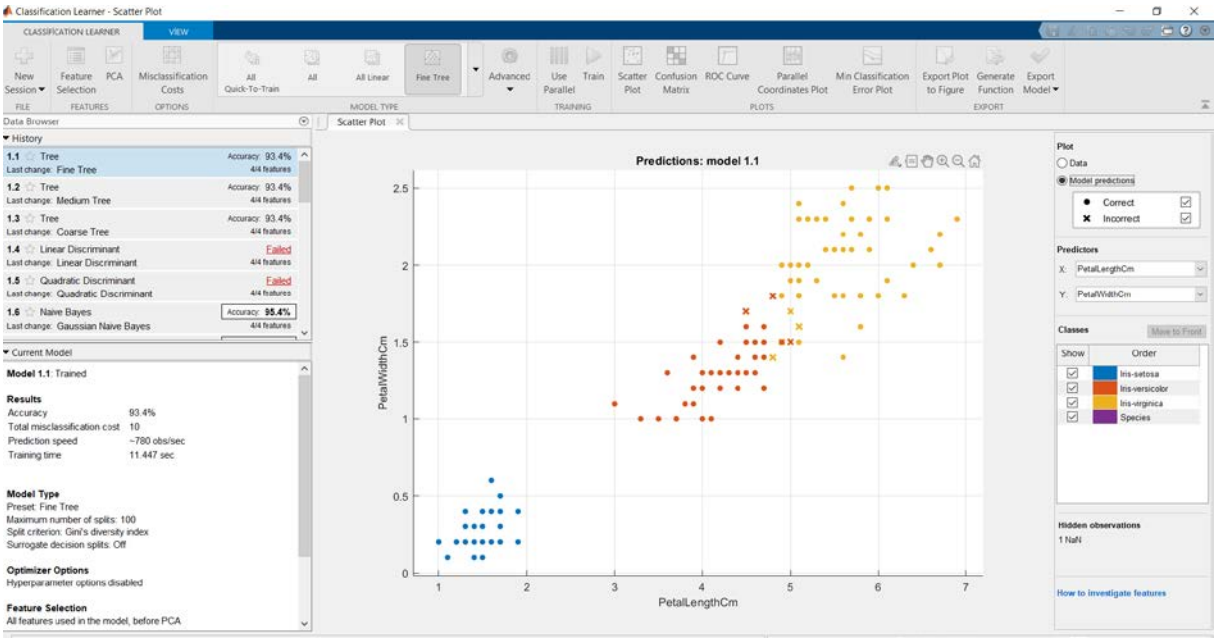
## CLASSİFİCATİON

## KARAR AĞACI

### Data Set is iris.csv

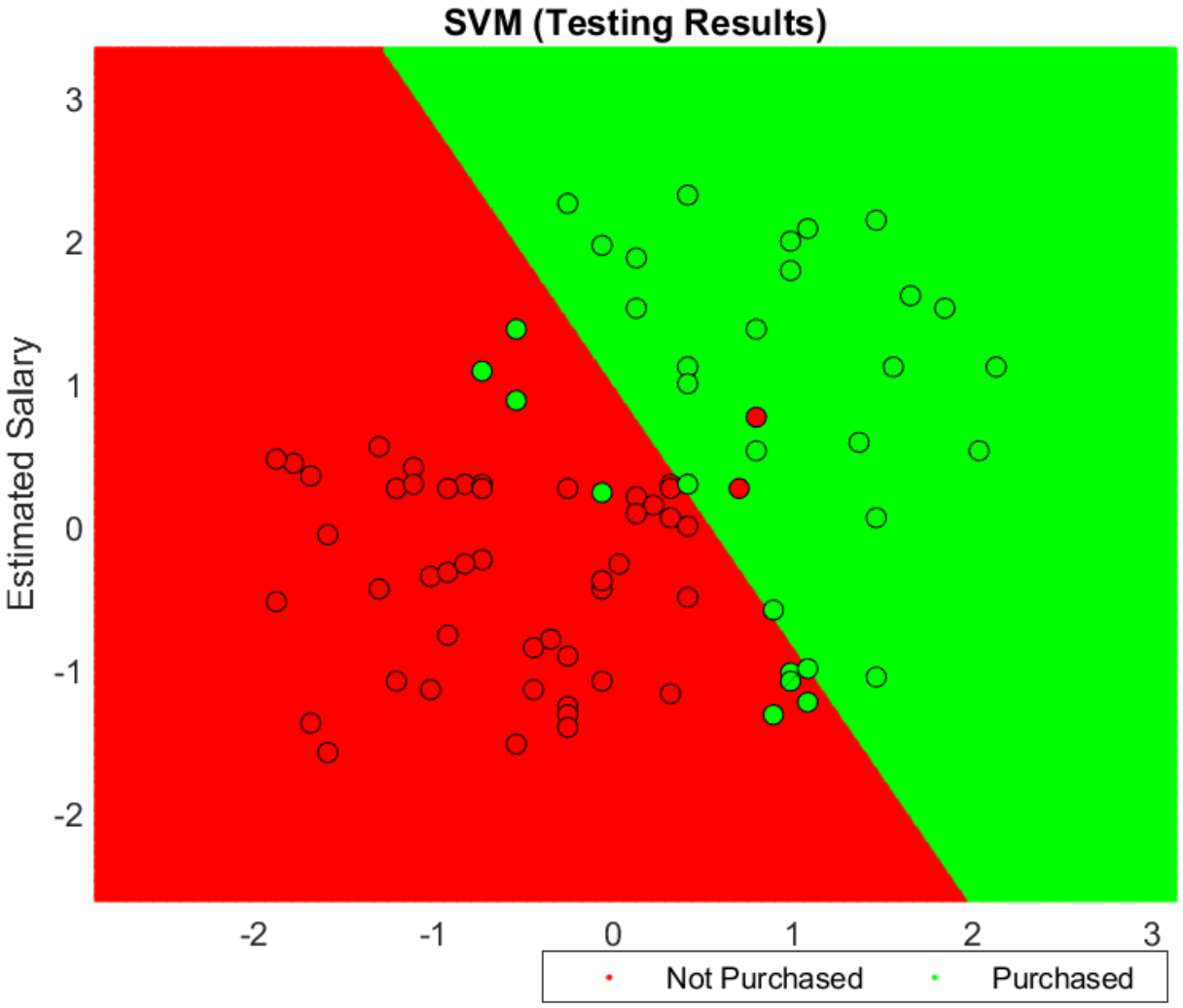


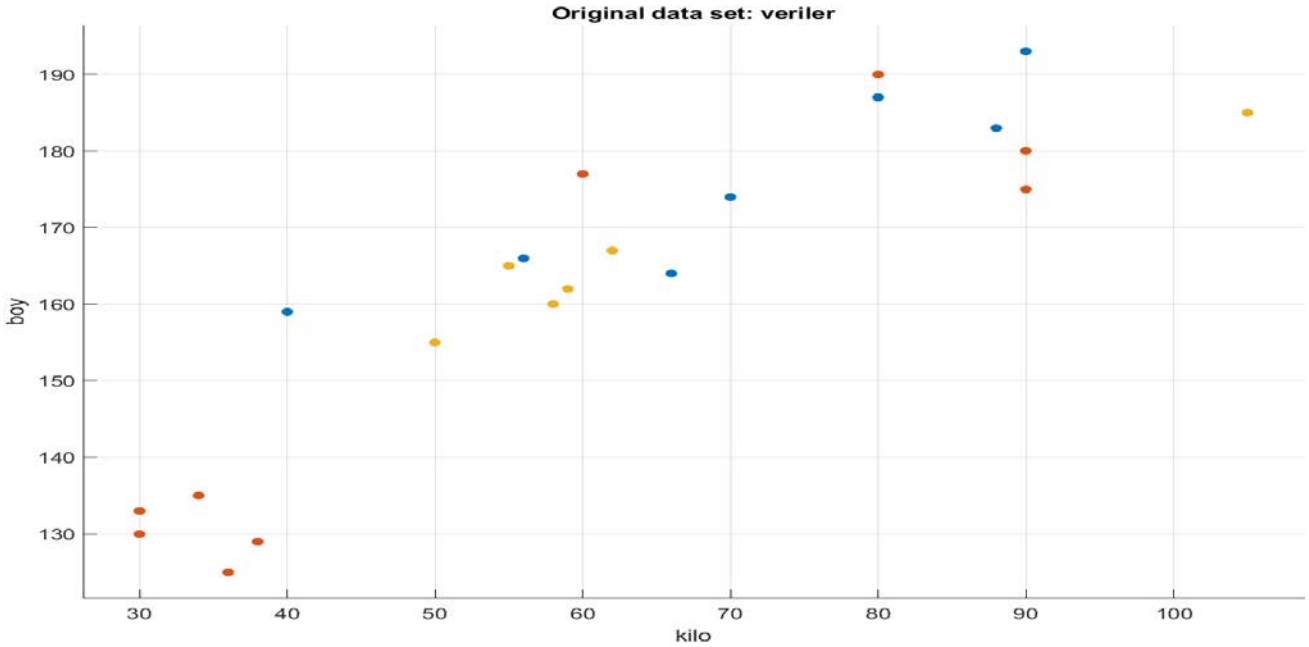


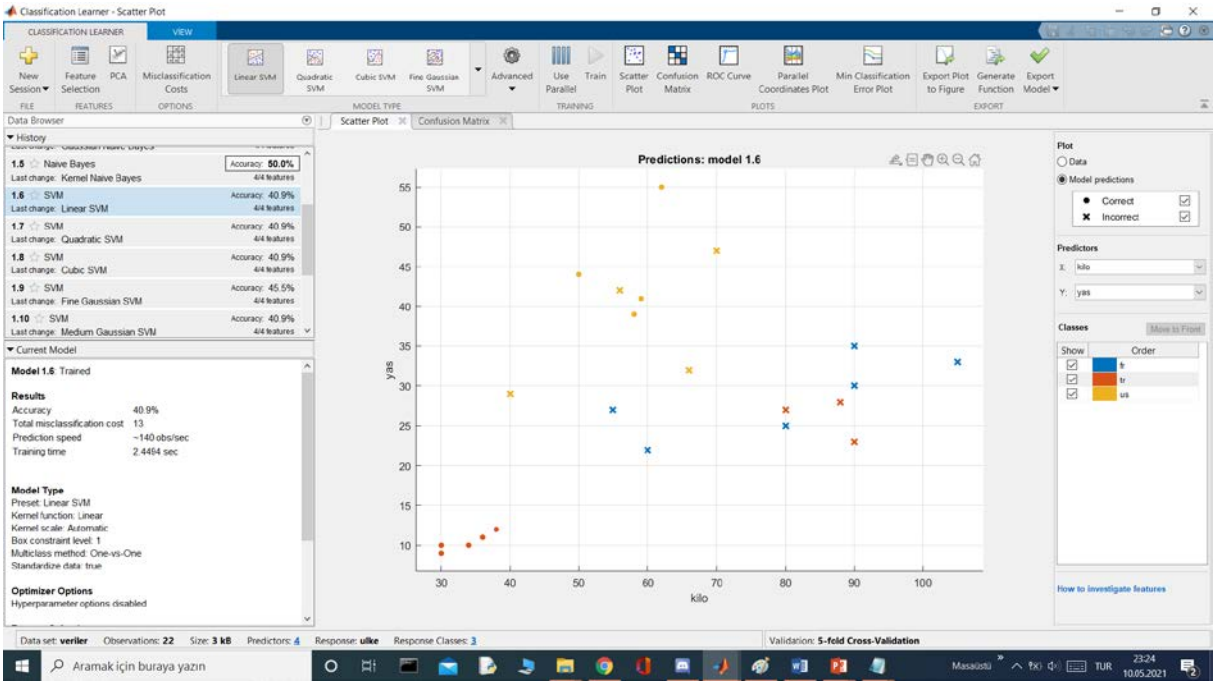

## SVM sınıflandırma


SVM (Training Results)
Estimated Salary
3
2
1
0
-1
-2
-2
-1
0
1
2
3
Not Purchased
Purchased

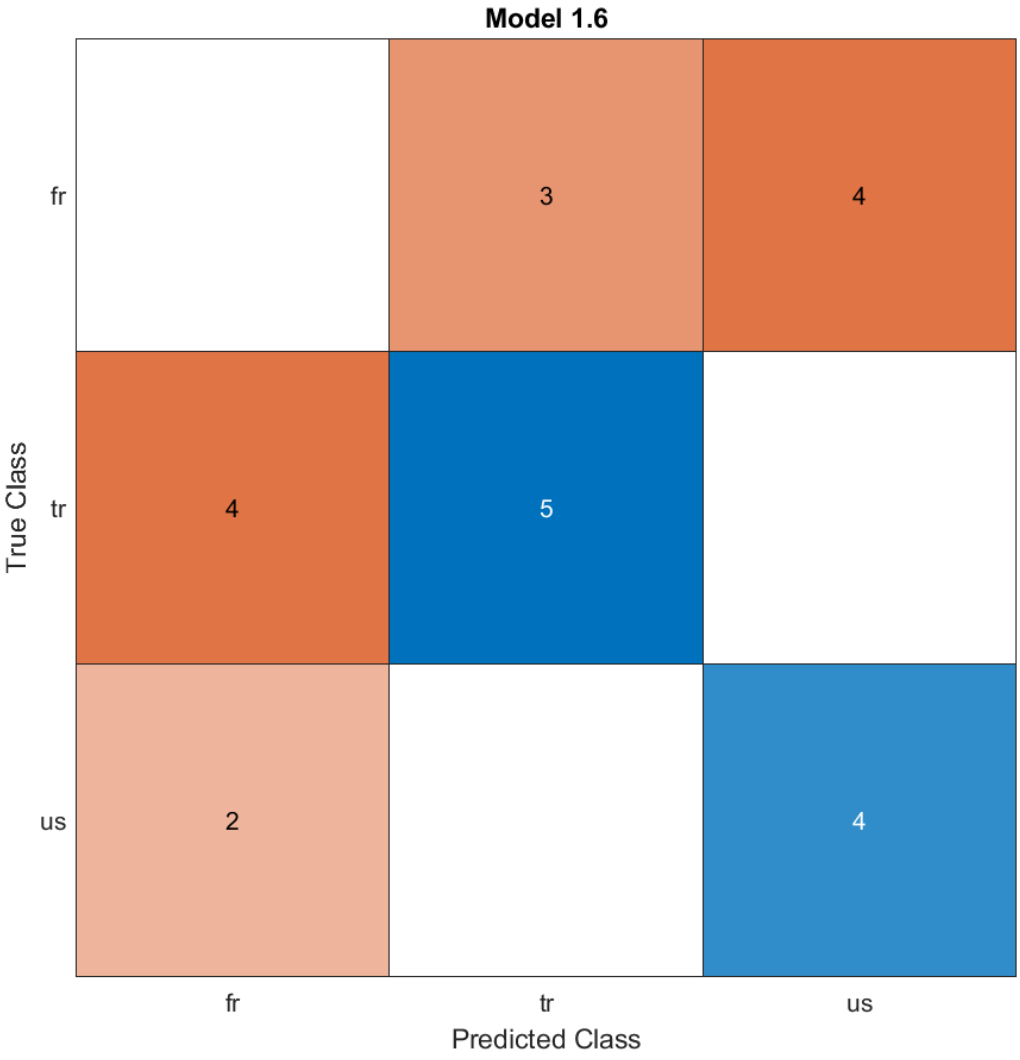


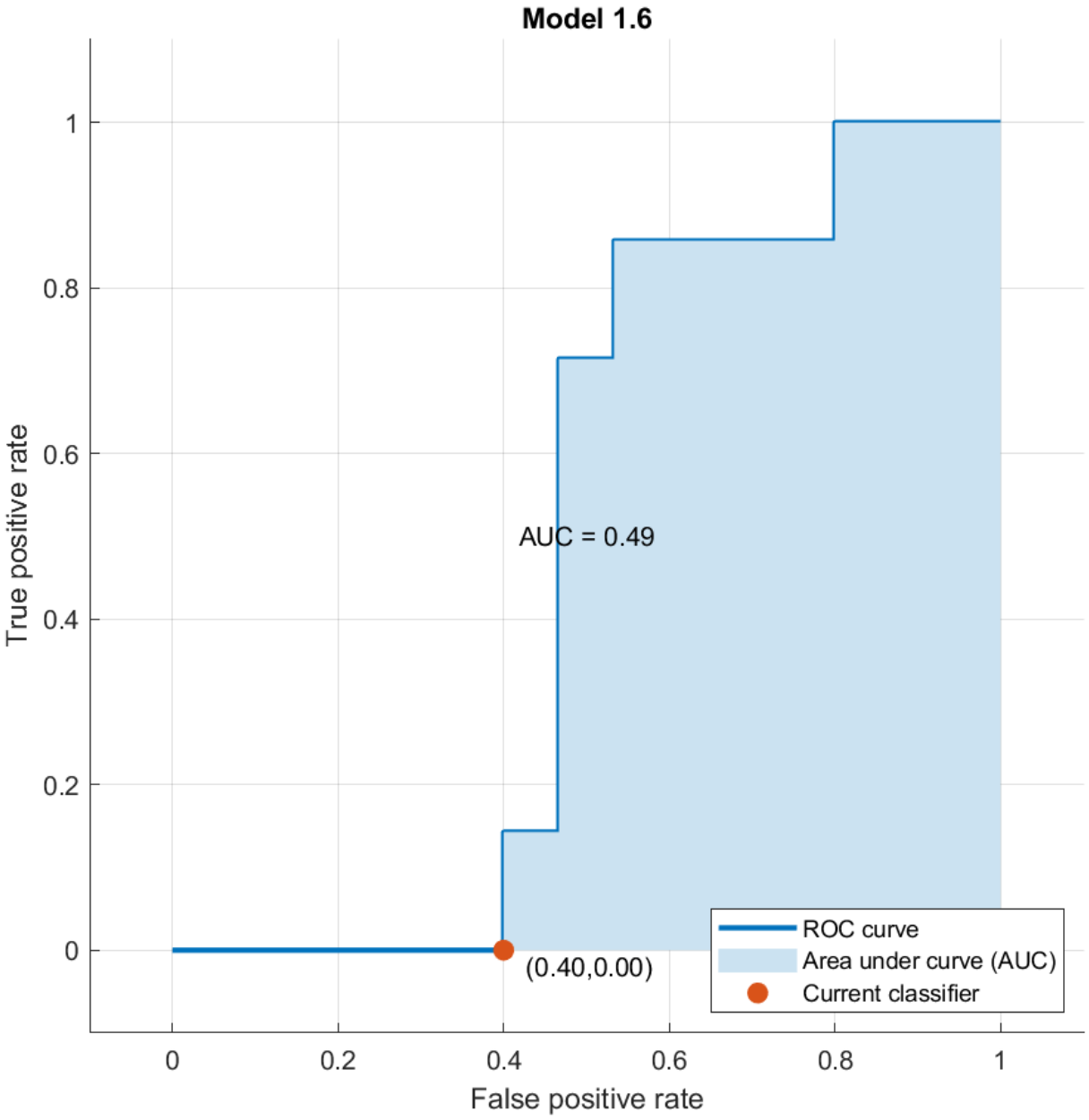

**Dataset: veriler.csv**




## Confusion Matrix


Model 1.6
True Class
fr
tr
us
3
4
4
5
2
4
fr
tr
us
Predicted Class


Model 1.6
True positive rate
False positive rate
AUC = 0.49
(0.40,0.00)
ROC curve
Area under curve (AUC)
Current classifier
0
0.2
0.4
0.6
0.8
1