|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import pandas as pd  from sklearn.metrics import r2\_score  # veri yukleme  veriler = pd.read\_csv('maaslar.csv')  x = veriler.iloc[:,2:4]  y = veriler.iloc[:,4:]  X = x.values  Y = y.values  #linear regression  from sklearn.linear\_model import LinearRegression  lin\_reg = LinearRegression()  lin\_reg.fit(X,Y)  plt.scatter(X,Y,color='red')  plt.plot(x,lin\_reg.predict(X), color = 'blue')  plt.show()  print('Linear R2 degeri')  print(r2\_score(Y, lin\_reg.predict(X)))  #polynomial regression  from sklearn.preprocessing import PolynomialFeatures  poly\_reg = PolynomialFeatures(degree = 2)  x\_poly = poly\_reg.fit\_transform(X)  print(x\_poly)  lin\_reg2 = LinearRegression()  lin\_reg2.fit(x\_poly,y)  plt.scatter(X,Y,color = 'red')  plt.plot(X,lin\_reg2.predict(poly\_reg.fit\_transform(X)), color = 'blue')  plt.show()  from sklearn.preprocessing import PolynomialFeatures  poly\_reg = PolynomialFeatures(degree = 4)  x\_poly = poly\_reg.fit\_transform(X)  print(x\_poly)  lin\_reg2 = LinearRegression()  lin\_reg2.fit(x\_poly,y)  plt.scatter(X,Y,color = 'red')  plt.plot(X,lin\_reg2.predict(poly\_reg.fit\_transform(X)), color = 'blue')  plt.show() |

|  |
| --- |
| #tahminler  print(lin\_reg.predict([[11]]))  print(lin\_reg.predict([[6.6]]))  print(lin\_reg2.predict(poly\_reg.fit\_transform([[6.6]])))  print(lin\_reg2.predict(poly\_reg.fit\_transform([[11]])))  print('Polynomial R2 degeri')  print(r2\_score(Y, lin\_reg2.predict(poly\_reg.fit\_transform(X))))  #verilerin olceklenmesi  from sklearn.preprocessing import StandardScaler  sc1=StandardScaler()  x\_olcekli = sc1.fit\_transform(X)  sc2=StandardScaler()  y\_olcekli = np.ravel(sc2.fit\_transform(Y.reshape(-1,1)))  from sklearn.svm import SVR  svr\_reg = SVR(kernel='rbf')  svr\_reg.fit(x\_olcekli,y\_olcekli)  plt.scatter(x\_olcekli,y\_olcekli,color='red')  plt.plot(x\_olcekli,svr\_reg.predict(x\_olcekli),color='blue')  plt.show()  print(svr\_reg.predict([[11]]))  print(svr\_reg.predict([[6.6]]))  print('SVR R2 degeri')  print(r2\_score(y\_olcekli, svr\_reg.predict(x\_olcekli)))  #Decision Tree Regresyon  from sklearn.tree import DecisionTreeRegressor  r\_dt = DecisionTreeRegressor(random\_state=0)  r\_dt.fit(X,Y)  Z = X + 0.5  K = X - 0.4  plt.scatter(X,Y, color='red')  plt.plot(x,r\_dt.predict(X), color='blue')  plt.plot(x,r\_dt.predict(Z),color='green')  plt.plot(x,r\_dt.predict(K),color='yellow')  plt.show()  print(r\_dt.predict([[11]]))  print(r\_dt.predict([[6.6]]))  print('Decision Tree R2 degeri')  print(r2\_score(Y, r\_dt.predict(X))) |

|  |  |
| --- | --- |
| #Random Forest Regresyonu  from sklearn.ensemble import RandomForestRegressor  rf\_reg=RandomForestRegressor(n\_estimators = 10,random\_state=0)  rf\_reg.fit(X,Y.ravel())  print(rf\_reg.predict([[6.6]]))  plt.scatter(X,Y,color='red')  plt.plot(X,rf\_reg.predict(X),color='blue')  plt.plot(X,rf\_reg.predict(Z),color='green')  plt.plot(x,r\_dt.predict(K),color='yellow')  plt.show()  print('Random Forest R2 degeri')  print(r2\_score(Y, rf\_reg.predict(X)))  print(r2\_score(Y, rf\_reg.predict(K)))  print(r2\_score(Y, rf\_reg.predict(Z)))  #Ozet R2 değerleri  print('-----------------------')  print('Linear R2 degeri')  print(r2\_score(Y, lin\_reg.predict(X)))  print('Polynomial R2 degeri')  print(r2\_score(Y, lin\_reg2.predict(poly\_reg.fit\_transform(X))))  print('SVR R2 degeri')  print(r2\_score(y\_olcekli, svr\_reg.predict(x\_olcekli))  print('Decision Tree R2 degeri')  print(r2\_score(Y, r\_dt.predict(X)))  print('Random Forest R2 degeri')  print(r2\_score(Y, rf\_reg.predict(X))) | Linear R2 degeri  0.6690412331929894  [[ 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.]]  [[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]]  [[34716.66666667]]  [[16923.33333333]]  [[8146.9948718]]  [[89041.66666667]]  Polynomial R2 degeri  0.9973922891706613  [0.01150915]  [0.01150915]  SVR R2 degeri  0.7513836788854973  [50000.]  [10000.]  Decision Tree R2 degeri  1.0  [10500.]  Random Forest R2 degeri  0.9704434230386582  0.9704434230386582  0.8820523231127324  -----------------------  Linear R2 degeri  0.6690412331929894  Polynomial R2 degeri  0.9973922891706613  SVR R2 degeri  0.7513836788854973  Decision Tree R2 degeri  1.0 |

|  |  |
| --- | --- |
| import numpy as np  import matplotlib.pyplot as plt  import pandas as pd  veriler = pd.read\_csv('veriler.csv')  #pd.read\_csv("veriler.csv")  #test  #print(veriler)  x = veriler.iloc[:,1:4].values #bağımsız değişkenler  y = veriler.iloc[:,4:].values #bağımlı değişken  #verilerin egitim ve test icin bolunmesi  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, random\_state=0)  #verilerin olceklenmesi  from sklearn.preprocessing import StandardScaler  sc=StandardScaler()  X\_train = sc.fit\_transform(x\_train)  X\_test = sc.transform(x\_test)  from sklearn.linear\_model import LogisticRegression  logr = LogisticRegression(random\_state=0)  logr.fit(X\_train,y\_train)  y\_pred = logr.predict(X\_test)  print(y\_pred)  print(y\_test)  from sklearn.metrics import confusion\_matrix  cm = confusion\_matrix(y\_test,y\_pred)  print(cm)  from sklearn.neighbors import KNeighborsClassifier  knn = KNeighborsClassifier(n\_neighbors=1, metric='minkowski')  knn.fit(X\_train,y\_train)  y\_pred = knn.predict(X\_test)  cm = confusion\_matrix(y\_test,y\_pred)  print(cm)  from sklearn.svm import SVC  svc = SVC(kernel='rbf')  svc.fit(X\_train,y\_train)  y\_pred = svc.predict(X\_test)  cm = confusion\_matrix(y\_test,y\_pred)  print('SVC')  print(cm)  from sklearn.naive\_bayes import GaussianNB  gnb = GaussianNB()  gnb.fit(X\_train, y\_train) | y\_pred = gnb.predict(X\_test)  cm = confusion\_matrix(y\_test,y\_pred)  print('GNB')  print(cm)  from sklearn.tree import DecisionTreeClassifier  dtc = DecisionTreeClassifier(criterion = 'entropy')  dtc.fit(X\_train,y\_train)  y\_pred = dtc.predict(X\_test)  cm = confusion\_matrix(y\_test,y\_pred)  print('DTC')  print(cm)  from sklearn.ensemble import RandomForestClassifier  rfc = RandomForestClassifier(n\_estimators=10, criterion = 'entropy')  rfc.fit(X\_train,y\_train)  y\_pred = rfc.predict(X\_test)  cm = confusion\_matrix(y\_test,y\_pred)  print('RFC')  print(cm)  y\_proba = rfc.predict\_proba(X\_test)  print(y\_test)  print(y\_proba[:,0])  from sklearn import metrics  fpr , tpr , thold = metrics.roc\_curve(y\_test,y\_proba[:,0],pos\_label='e')  print(fpr)  print(tpr)  ['e' 'e' 'e' 'e' 'k' 'e' 'k']  [['k']['k']['k']['k'] ['e']['k']['k']]  [[0 1][5 1]]  [[1 0][1 5]]  SVC  [[1 0][1 5]]  GNB  [[0 1][5 1]]  DTC  [[1 0][1 5]]  RFC  [[1 0][0 6]]  [['k']['k']['k']['k']['e']['k']['k']]  [0. 0. 0.3 0. 0.7 0. 0.2]  [0. 0. 0.33333333 1. ]  [0. 1. 1. 1.] |