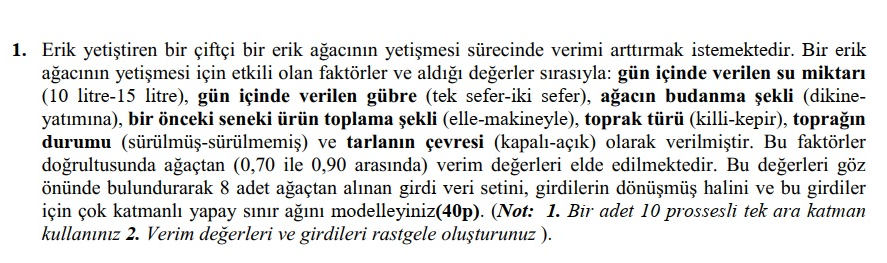
**Soru : 1**



Kodlar :

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

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

#Gerekli Kütüphaneleri Ekliyoruz

import numpy as np **#Liste ve array işlemleri için**

import pandas as pd **#Dataframeler için**

import random **#rastgele değer türetmek için**

**#Bizde soruda istenen rastgele atamaları yapalım**

sumiktarı = []

for i in range(8):

sumiktarı.append(random.randint(7,16))

gübre\_tekrar = []

for i in range(8):

gübre\_tekrar.append(random.randint(1,2))

budanma = []

for i in range(8):

budanma.append(random.randint(0,1))

toplanma = []

for i in range(8):

toplanma.append(random.randint(0,1))

toprak = []

for i in range(8):

toprak.append(random.randint(0,1))

toprakdurum = []

for i in range(8):

toprakdurum.append(random.randint(0,1))

tarlacevresi = []

for i in range(8):

tarlacevresi.append(random.randint(0,1))

verim = []

for i in range(8):

verim.append(float(random.uniform(0.7,0.9)))

index=["1.ağac","2.ağac","3.ağac","4.ağac","5.ağac","6.ağac","7.ağac","8.ağac"] **#Kullanmak istersek diye**

**#Altta verileri sözlüğe çevirdik dataframe e kolaylıkla dönüştürebilmek için**

datas = {"sumiktarı":sumiktarı,"gübre\_tekrar":gübre\_tekrar,"budanma":budanma,"toplanma":toplanma,"toprak":toprak,"toprakdurum":toprakdurum,"tarlacevresi":tarlacevresi,"verim":verim}

**#dataframe e çevirdik**

df = pd.DataFrame(datas)

X = df.iloc[:,:7].values#Özellikleri belirttik

y = df.iloc[:,7].values#Sonucları belirttik

**#Yapay Sinir Ağı işlemleri için keras ı ekledik**

import keras

from keras.models import Sequential

from keras.layers import Dense

predictier = Sequential() **#Katmanı başlattık**

**#10 processli bir hidden layer ekledik**

predictier.add(Dense(output\_dim=10,init="uniform",activation="relu",input\_dim=7))

**#Çıktı layerını ayarladık**

predictier.add(Dense(output\_dim = 1,init = "uniform",activation="sigmoid"))

**#Sinir ağımızı derledik**

predictier.compile(optimizer = "adam",loss = "binary\_crossentropy",metrics=["accuracy"])

**#Sinir ağını verilerimizle eğitelim**

predictier.fit(X,y,batch\_size=32,epochs=100)

**#Tahmin yapalım**

print("Örnek Tahmin : "+str(predictier.predict(np.array([13,2,0,1,0,0,0]).reshape(1,-1))))

**#Gerekli eklentiler**

**#Numpy 1.18.4**

**#pandas 1.0.3**

**#matplotlib**

**#keras 2.3.1**

**#random (default gelmekte)**

**--------------------------------------------------------------------------------------------------------------------------------------**

Örnek Ekran Çıktısı :

Epoch 1/100

8/8 [==============================] - 0s 18ms/step - loss: 0.6897 - accuracy: 0.0000e+00

Epoch 2/100

8/8 [==============================] - 0s 125us/step - loss: 0.6884 - accuracy: 0.0000e+00

Epoch 3/100

8/8 [==============================] - 0s 125us/step - loss: 0.6873 - accuracy: 0.0000e+00

Epoch 4/100

8/8 [==============================] - 0s 125us/step - loss: 0.6861 - accuracy: 0.0000e+00

Epoch 5/100

8/8 [==============================] - 0s 249us/step - loss: 0.6849 - accuracy: 0.0000e+00

Epoch 6/100

8/8 [==============================] - 0s 124us/step - loss: 0.6836 - accuracy: 0.0000e+00

Epoch 7/100

8/8 [==============================] - 0s 125us/step - loss: 0.6822 - accuracy: 0.0000e+00

Epoch 8/100

8/8 [==============================] - 0s 125us/step - loss: 0.6808 - accuracy: 0.0000e+00

Epoch 9/100

8/8 [==============================] - 0s 125us/step - loss: 0.6793 - accuracy: 0.0000e+00

Epoch 10/100

8/8 [==============================] - 0s 125us/step - loss: 0.6778 - accuracy: 0.0000e+00

Epoch 11/100

8/8 [==============================] - 0s 124us/step - loss: 0.6763 - accuracy: 0.0000e+00

Epoch 12/100

8/8 [==============================] - 0s 125us/step - loss: 0.6747 - accuracy: 0.0000e+00

Epoch 13/100

8/8 [==============================] - 0s 125us/step - loss: 0.6731 - accuracy: 0.0000e+00

Epoch 14/100

8/8 [==============================] - 0s 125us/step - loss: 0.6714 - accuracy: 0.0000e+00

Epoch 15/100

W

8/8 [==============================] - 0s 125us/step - loss: 0.6697 - accuracy: 0.0000e+00

Epoch 16/100

8/8 [==============================] - 0s 125us/step - loss: 0.6680 - accuracy: 0.0000e+00

Epoch 17/100

8/8 [==============================] - 0s 250us/step - loss: 0.6662 - accuracy: 0.0000e+00

Epoch 18/100

8/8 [==============================] - 0s 125us/step - loss: 0.6644 - accuracy: 0.0000e+00

Epoch 19/100

8/8 [==============================] - 0s 250us/step - loss: 0.6626 - accuracy: 0.0000e+00

Epoch 20/100

8/8 [==============================] - 0s 249us/step - loss: 0.6607 - accuracy: 0.0000e+00

Epoch 21/100

8/8 [==============================] - 0s 249us/step - loss: 0.6588 - accuracy: 0.0000e+00

Epoch 22/100

8/8 [==============================] - 0s 250us/step - loss: 0.6568 - accuracy: 0.0000e+00

Epoch 23/100

8/8 [==============================] - 0s 125us/step - loss: 0.6548 - accuracy: 0.0000e+00

Epoch 24/100

8/8 [==============================] - 0s 125us/step - loss: 0.6527 - accuracy: 0.0000e+00

Epoch 25/100

8/8 [==============================] - 0s 250us/step - loss: 0.6506 - accuracy: 0.0000e+00

Epoch 26/100

8/8 [==============================] - 0s 124us/step - loss: 0.6485 - accuracy: 0.0000e+00

Epoch 27/100

8/8 [==============================] - 0s 125us/step - loss: 0.6463 - accuracy: 0.0000e+00

Epoch 28/100

8/8 [==============================] - 0s 125us/step - loss: 0.6441 - accuracy: 0.0000e+00

Epoch 29/100

8/8 [==============================] - 0s 250us/step - loss: 0.6418 - accuracy: 0.0000e+00

Epoch 30/100

8/8 [==============================] - 0s 249us/step - loss: 0.6395 - accuracy: 0.0000e+00

Epoch 31/100

8/8 [==============================] - 0s 126us/step - loss: 0.6372 - accuracy: 0.0000e+00

Epoch 32/100

8/8 [==============================] - 0s 249us/step - loss: 0.6348 - accuracy: 0.0000e+00

Epoch 33/100

8/8 [==============================] - 0s 249us/step - loss: 0.6324 - accuracy: 0.0000e+00

Epoch 34/100

8/8 [==============================] - 0s 125us/step - loss: 0.6300 - accuracy: 0.0000e+00

Epoch 35/100

8/8 [==============================] - 0s 125us/step - loss: 0.6275 - accuracy: 0.0000e+00

Epoch 36/100

8/8 [==============================] - 0s 125us/step - loss: 0.6250 - accuracy: 0.0000e+00

Epoch 37/100

8/8 [==============================] - 0s 249us/step - loss: 0.6224 - accuracy: 0.0000e+00

Epoch 38/100

8/8 [==============================] - 0s 250us/step - loss: 0.6199 - accuracy: 0.0000e+00

Epoch 39/100

8/8 [==============================] - 0s 249us/step - loss: 0.6173 - accuracy: 0.0000e+00

Epoch 40/100

8/8 [==============================] - 0s 250us/step - loss: 0.6147 - accuracy: 0.0000e+00

Epoch 41/100

8/8 [==============================] - 0s 249us/step - loss: 0.6120 - accuracy: 0.0000e+00

Epoch 42/100

8/8 [==============================] - 0s 124us/step - loss: 0.6094 - accuracy: 0.0000e+00

Epoch 43/100

8/8 [==============================] - 0s 249us/step - loss: 0.6067 - accuracy: 0.0000e+00

Epoch 44/100

8/8 [==============================] - 0s 249us/step - loss: 0.6040 - accuracy: 0.0000e+00

Epoch 45/100

8/8 [==============================] - 0s 124us/step - loss: 0.6013 - accuracy: 0.0000e+00

Epoch 46/100

8/8 [==============================] - 0s 250us/step - loss: 0.5986 - accuracy: 0.0000e+00

Epoch 47/100

8/8 [==============================] - 0s 125us/step - loss: 0.5958 - accuracy: 0.0000e+00

Epoch 48/100

8/8 [==============================] - 0s 125us/step - loss: 0.5931 - accuracy: 0.0000e+00

Epoch 49/100

8/8 [==============================] - 0s 124us/step - loss: 0.5904 - accuracy: 0.0000e+00

Epoch 50/100

8/8 [==============================] - 0s 248us/step - loss: 0.5877 - accuracy: 0.0000e+00

Epoch 51/100

8/8 [==============================] - 0s 250us/step - loss: 0.5850 - accuracy: 0.0000e+00

Epoch 52/100

8/8 [==============================] - 0s 250us/step - loss: 0.5823 - accuracy: 0.0000e+00

Epoch 53/100

8/8 [==============================] - 0s 125us/step - loss: 0.5796 - accuracy: 0.0000e+00

Epoch 54/100

8/8 [==============================] - 0s 249us/step - loss: 0.5770 - accuracy: 0.0000e+00

Epoch 55/100

8/8 [==============================] - 0s 249us/step - loss: 0.5743 - accuracy: 0.0000e+00

Epoch 56/100

8/8 [==============================] - 0s 125us/step - loss: 0.5717 - accuracy: 0.0000e+00

Epoch 57/100

8/8 [==============================] - 0s 124us/step - loss: 0.5691 - accuracy: 0.0000e+00

Epoch 58/100

8/8 [==============================] - 0s 125us/step - loss: 0.5665 - accuracy: 0.0000e+00

Epoch 59/100

8/8 [==============================] - 0s 125us/step - loss: 0.5640 - accuracy: 0.0000e+00

Epoch 60/100

8/8 [==============================] - 0s 125us/step - loss: 0.5615 - accuracy: 0.0000e+00

Epoch 61/100

8/8 [==============================] - 0s 249us/step - loss: 0.5591 - accuracy: 0.0000e+00

Epoch 62/100

8/8 [==============================] - 0s 249us/step - loss: 0.5567 - accuracy: 0.0000e+00

Epoch 63/100

8/8 [==============================] - 0s 124us/step - loss: 0.5543 - accuracy: 0.0000e+00

Epoch 64/100

8/8 [==============================] - 0s 250us/step - loss: 0.5520 - accuracy: 0.0000e+00

Epoch 65/100

8/8 [==============================] - 0s 125us/step - loss: 0.5498 - accuracy: 0.0000e+00

Epoch 66/100

8/8 [==============================] - 0s 250us/step - loss: 0.5476 - accuracy: 0.0000e+00

Epoch 67/100

8/8 [==============================] - 0s 124us/step - loss: 0.5455 - accuracy: 0.0000e+00

Epoch 68/100

8/8 [==============================] - 0s 125us/step - loss: 0.5434 - accuracy: 0.0000e+00

Epoch 69/100

8/8 [==============================] - 0s 125us/step - loss: 0.5414 - accuracy: 0.0000e+00

Epoch 70/100

8/8 [==============================] - 0s 249us/step - loss: 0.5395 - accuracy: 0.0000e+00

Epoch 71/100

8/8 [==============================] - 0s 125us/step - loss: 0.5376 - accuracy: 0.0000e+00

Epoch 72/100

8/8 [==============================] - 0s 249us/step - loss: 0.5358 - accuracy: 0.0000e+00

Epoch 73/100

8/8 [==============================] - 0s 249us/step - loss: 0.5340 - accuracy: 0.0000e+00

Epoch 74/100

8/8 [==============================] - 0s 250us/step - loss: 0.5324 - accuracy: 0.0000e+00

Epoch 75/100

8/8 [==============================] - 0s 125us/step - loss: 0.5308 - accuracy: 0.0000e+00

Epoch 76/100

8/8 [==============================] - 0s 125us/step - loss: 0.5292 - accuracy: 0.0000e+00

Epoch 77/100

8/8 [==============================] - 0s 125us/step - loss: 0.5278 - accuracy: 0.0000e+00

Epoch 78/100

8/8 [==============================] - 0s 250us/step - loss: 0.5264 - accuracy: 0.0000e+00

Epoch 79/100

8/8 [==============================] - 0s 249us/step - loss: 0.5251 - accuracy: 0.0000e+00

Epoch 80/100

8/8 [==============================] - 0s 125us/step - loss: 0.5238 - accuracy: 0.0000e+00

Epoch 81/100

8/8 [==============================] - 0s 250us/step - loss: 0.5226 - accuracy: 0.0000e+00

Epoch 82/100

8/8 [==============================] - 0s 250us/step - loss: 0.5215 - accuracy: 0.0000e+00

Epoch 83/100

8/8 [==============================] - 0s 125us/step - loss: 0.5205 - accuracy: 0.0000e+00

Epoch 84/100

8/8 [==============================] - 0s 249us/step - loss: 0.5195 - accuracy: 0.0000e+00

Epoch 85/100

8/8 [==============================] - 0s 249us/step - loss: 0.5186 - accuracy: 0.0000e+00

Epoch 86/100

8/8 [==============================] - 0s 124us/step - loss: 0.5177 - accuracy: 0.0000e+00

Epoch 87/100

8/8 [==============================] - 0s 249us/step - loss: 0.5170 - accuracy: 0.0000e+00

Epoch 88/100

8/8 [==============================] - 0s 247us/step - loss: 0.5162 - accuracy: 0.0000e+00

Epoch 89/100

8/8 [==============================] - 0s 124us/step - loss: 0.5155 - accuracy: 0.0000e+00

Epoch 90/100

8/8 [==============================] - 0s 125us/step - loss: 0.5149 - accuracy: 0.0000e+00

Epoch 91/100

8/8 [==============================] - 0s 250us/step - loss: 0.5143 - accuracy: 0.0000e+00

Epoch 92/100

8/8 [==============================] - 0s 125us/step - loss: 0.5138 - accuracy: 0.0000e+00

Epoch 93/100

8/8 [==============================] - 0s 249us/step - loss: 0.5133 - accuracy: 0.0000e+00

Epoch 94/100

8/8 [==============================] - 0s 125us/step - loss: 0.5129 - accuracy: 0.0000e+00

Epoch 95/100

8/8 [==============================] - 0s 124us/step - loss: 0.5125 - accuracy: 0.0000e+00

Epoch 96/100

8/8 [==============================] - 0s 125us/step - loss: 0.5121 - accuracy: 0.0000e+00

Epoch 97/100

8/8 [==============================] - 0s 124us/step - loss: 0.5118 - accuracy: 0.0000e+00

Epoch 98/100

8/8 [==============================] - 0s 249us/step - loss: 0.5115 - accuracy: 0.0000e+00

Epoch 99/100

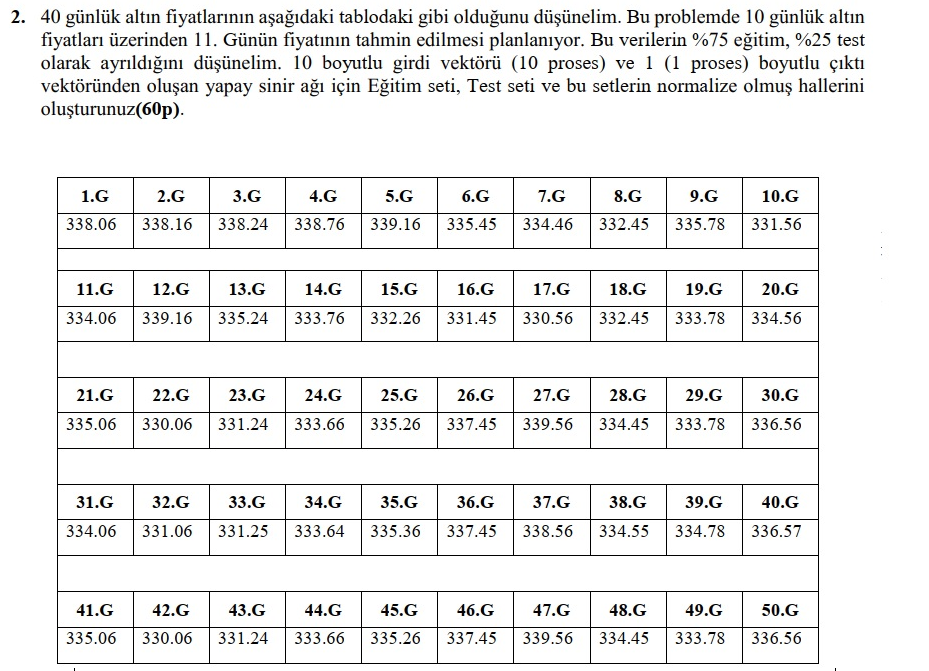
8/8 [==============================] - 0s 125us/step - loss: 0.5112 - accuracy: 0.0000e+00

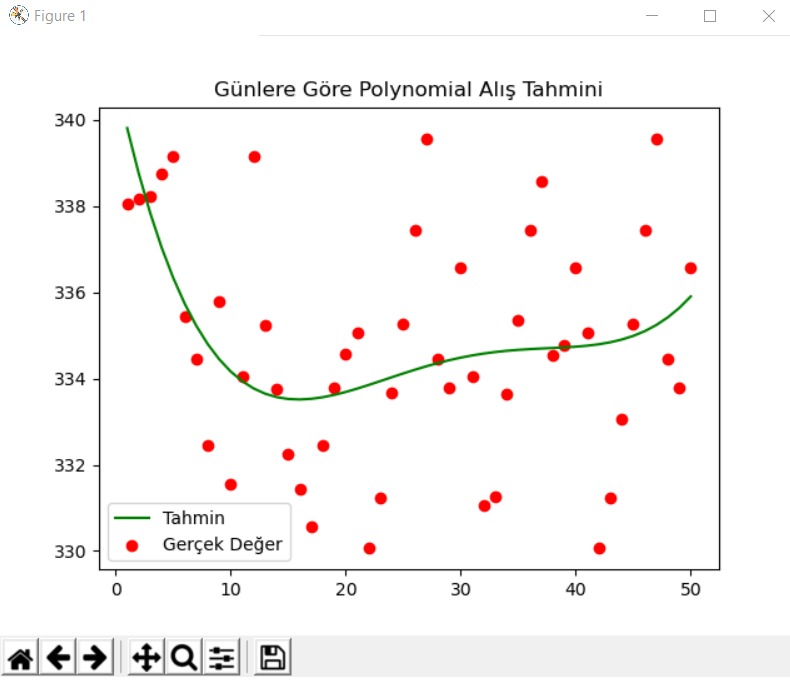
Epoch 100/100

8/8 [==============================] - 0s 249us/step - loss: 0.5110 - accuracy: 0.0000e+00

**Örnek Tahmin : [[0.7993987]]**

**Soru 2 :**





normalizasyon ve train test split görmüş hali buradaki fark normalizasyon işleminden kaynaklanıyor



Kaynak kodu :

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

**"""**

**Created on Mon Jun 15 14:24:36 2020**

**@author: Murat Çiçek**

**"""**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

veri = pd.read\_csv('C:\\Users\\Murat\\Desktop\\ysa\\data.csv') **#bilgisayarımdaki yer sizde neredeyse onu yapabilirsiniz.**

X = veri["Days"].values.reshape(-1,1)

y = veri["Prices"].values.reshape(-1,1)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size = 0.2,random\_state=0)

from sklearn.preprocessing import Normalizer

Sc = Normalizer()

X\_train = Sc.fit\_transform(X\_train)

X\_test = Sc.fit\_transform(X\_test)

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=4)

from sklearn.linear\_model import LinearRegression

lr = LinearRegression()

x\_poly = poly.fit\_transform(X\_train)

sample = y[:40]

lr.fit(x\_poly,sample)

plt.title("Günlere Göre Polynomial Alış Tahmini")

plt.scatter(X,y,color = "red", label = "Gerçek Değer")

plt.plot(X,lr.predict(poly.fit\_transform(X)), color = "green", label = "Tahmin")

plt.legend()

plt.show()

print("11.gün altın alış fiyatı : ")

print(lr.predict(poly.fit\_transform(np.array([11]).reshape(-1,1))))

**#Gerekli eklentiler**

**#Numpy 1.18.4**

**#pandas 1.0.3**

**#matplotlib**

**#örneğin pip install matplotlib şeklinde kurulabilir.**

Data.scv dosyası :

Days,Prices

1,338.06

2,338.16

3,338.24

4,338.76

5,339.16

6,335.45

7,334.46

8,332.45

9,335.78

10,331.56

11,334.06

12,339.16

13,335.24

14,333.76

15,332.26

16,331.45

17,330.56

18,332.45

19,333.78

20,334.56

21,335.06

22,330.06

23,331.24

24,333.66

25,335.26

26,337.45

27,339.56

28,334.45

29,333.78

30,336.56

31,334.06

32,331.06

33,331.25

34,333.64

35,335.36

36,337.45

37,338.56

38,334.55

39,334.78

40,336.57

41,335.06

42,330.06

43,331.24

44,333.06

45,335.26

46,337.45

47,339.56

48,334.45

49,333.78

50,336.56