PERCEPTRON ALGORİTMASI

Perceptron algoritması kullanılarak iris veri setinin sınıflandırılması sağlanmıştır. Uygulama içinde versi setinin iki boyutu seçilerek ilerlenmiştir.

Kısaca Perceptron en basit tek katmanlı sinir ağı modelidir. Tek katmanlı olduğu için sadece girdi ve çıktı katmanlarından oluşur.

Bir perceptron bir nöronun basit bir modelidir. x1 x2 … xn gibi farklı girişlere ve herbir giriş w1 w2 … wn gibi farklı ağırlıklara sahiptir.

Perceptron toplama(birleştirme) fonksiyonu ve aktivasyon fonksiyonu olmak üzere iki temel kısımdan oluşmaktadır.

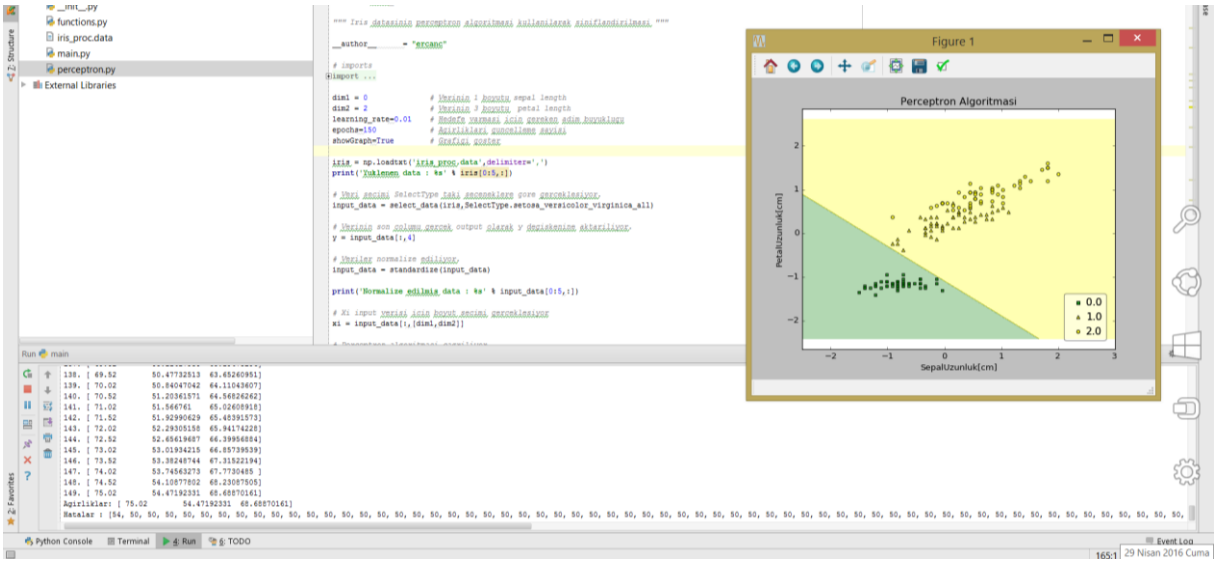
Toplama fonksiyonu perceptrona gelen net inputu hesaplar, girişler ve bias ın ağırlıkarı ile çarpımlarının toplamıdır. Aktivasyon fonksiyonu net inputu aşağıdaki gibi bir işlemden geçirerek çıkışı belirler. Bu iş için genellik sigmoid fonksiyonu türevi alınabildiğinden, sürekli olduğundan dolayı kullanılır.

$$f(s) = \begin{cases} 1 & \text{if } s \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

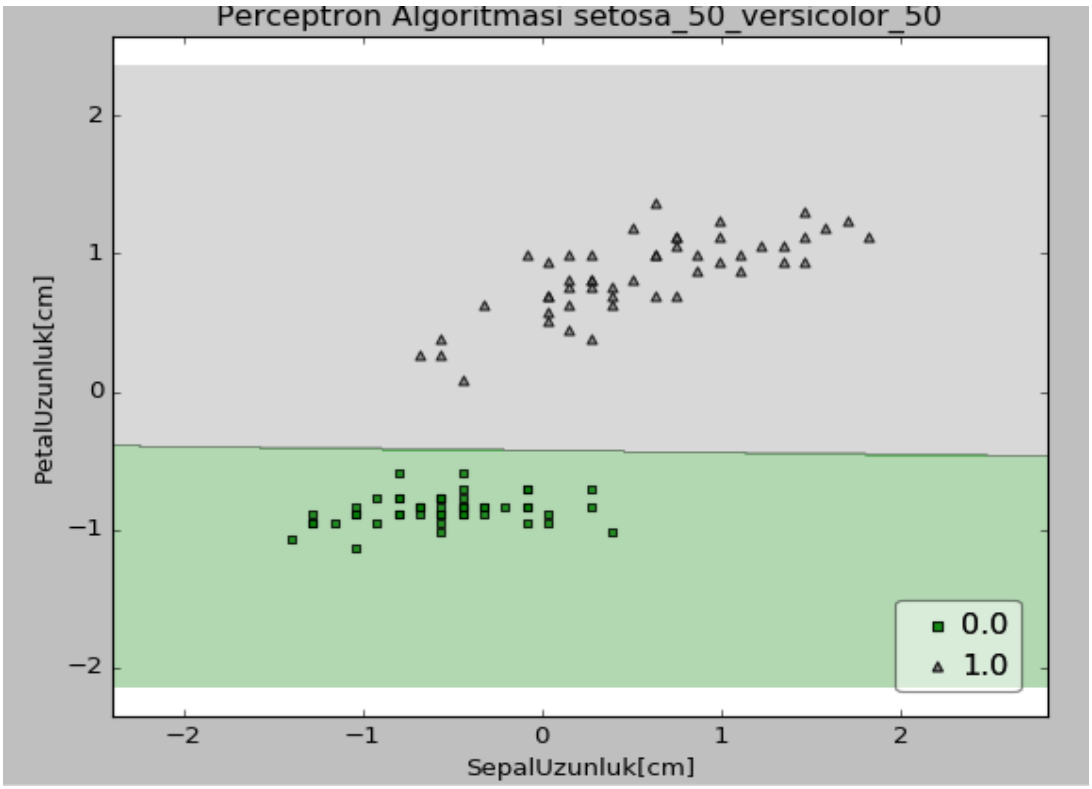
İris data setinin boyutlarının tanımı aşağıdaki gibidir. En son boyut verinin sınıfını göstermektedir.

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  0
## 2          4.9         3.0          1.4         0.2  0
## 3          4.7         3.2          1.3         0.2  0
## .          .           .            .           .
## .          .           .            .           .
## 51         7.0         3.2          4.7         1.4  1
## 52         6.4         3.2          4.5         1.5  1
## 53         6.9         3.1          4.9         1.5  1
## .          .           .            .           .
## .          .           .            .           .
## 51         6.3         3.3          6.0         2.5  2
## 52         5.8         2.7          5.1         1.9  2
## 53         7.1         3.0          5.9         2.1  2
```

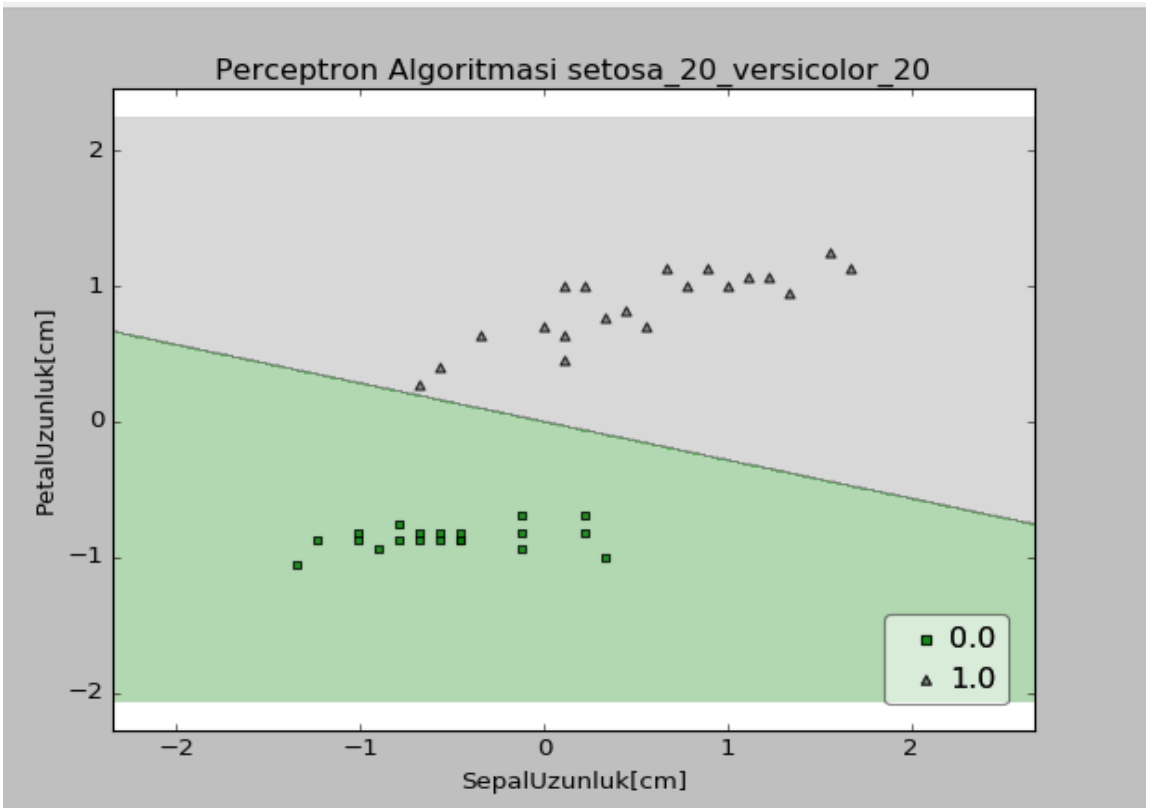
Perceptron algoritması pycharm geliştirme ortamı üzerinden geliştirildi.



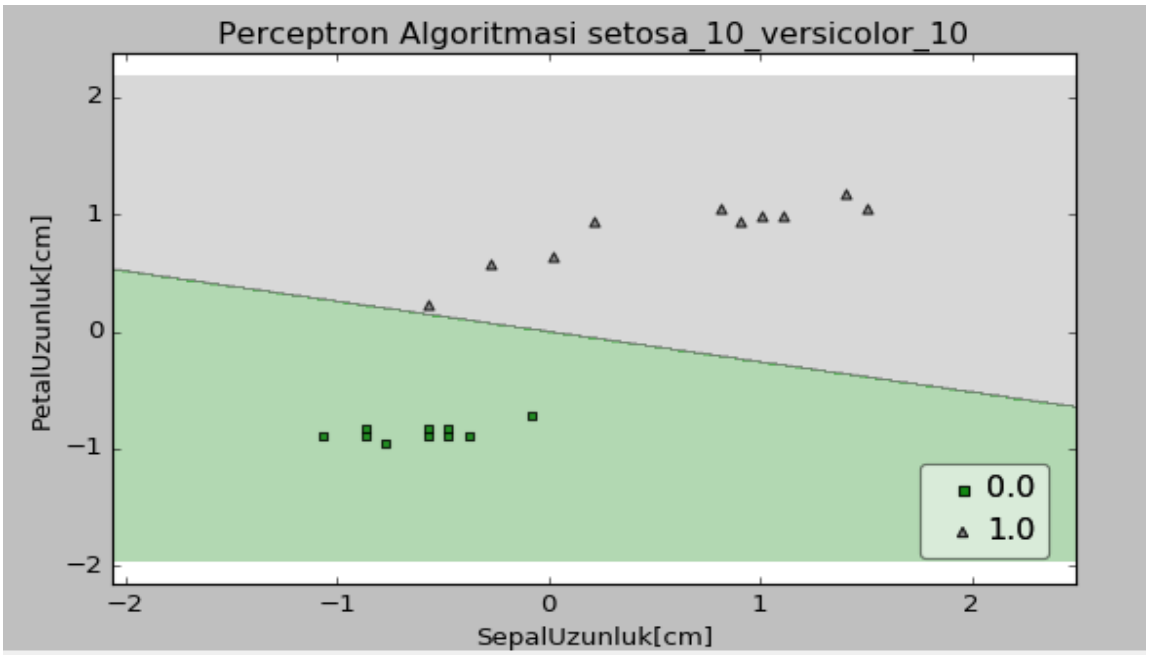
Verinin iki boyutu seçilerek aşağıdaki şekilde grafikler elde edilmitir. Farklı sayıda kayıtlar ile çalışılmıştır.



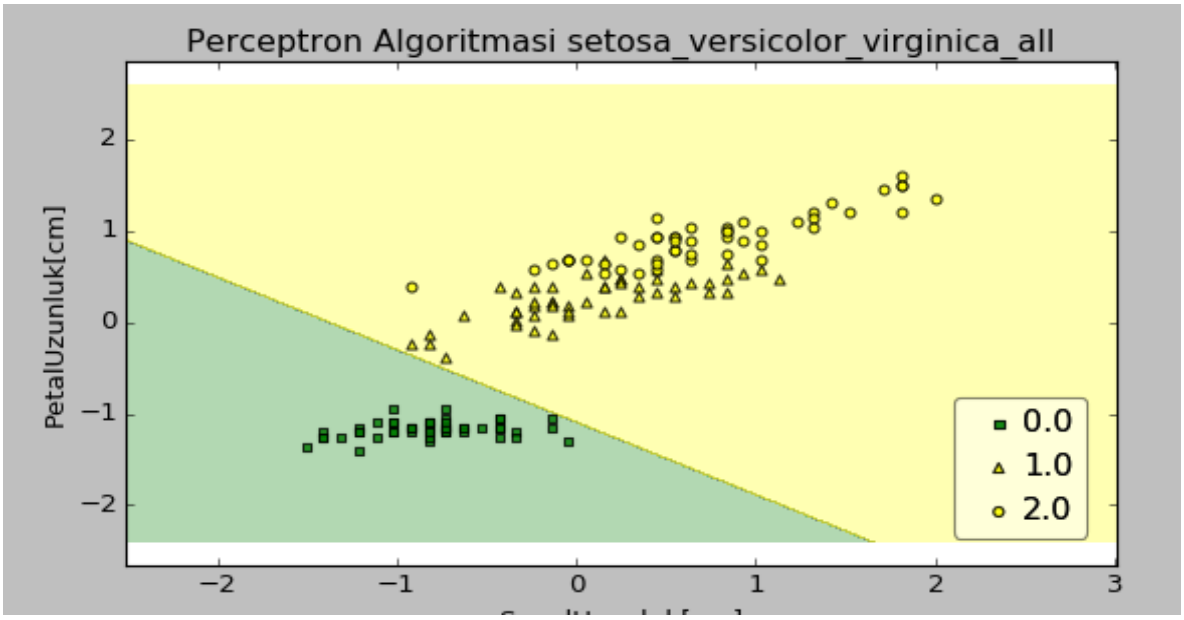
Birinci şekilde 50 setosa, 50 versicolor seçildi.



İkinci şekilde 20 setosa 20 versicolor seçildi.



Üçüncü şekilde 10 setosa ve 10 versicolor seçildi.



Dördüncü şekilde tüm veri seti kullanıldı. Aktivasyon fonksiyonu karakterinde kaynaklı çıkış iki sınıf olarak bulundu. Virgica ları error olarak hesaplayıp sarı alana versicolor ı olduğu yerlere veri özelliğinden kaynaklı bir şekilde ekledi.

main.py

```
#encoding: utf-8
# module perceptron_project.main


"""
     Ercan Can
     Iris datasinin perceptron algoritmasi kullanilarak siniflandirilmasi
"""


__author__      = "ercanc"

# imports
import numpy as np
import matplotlib.pyplot as plt
from perceptron import Perceptron
from functions import plot_draw,standardize,select_data,SelectType

dim1 = 0              # Verinin 1 boyutu sepal length
dim2 = 2              # Verinin 3 boyutu  petal length
learning_rate=0.01    # Hedefe varmasi icin gereken adim buyuklugu
epochs=150            # Agirliklari guncelleme sayisi
showGraph=True        # Grafigi goster

iris = np.loadtxt('iris_proc.data',delimiter=',')
print('Yuklenen data : %s' % iris[0:5,:])

# Veri secimi SelectType taki seceneklere gore gerceklesiyor.
input_data =
select_data(iris,SelectType.setosa_versicolor_virginica_all)
```

```
# Verinin son columu gercek output olarak y degiskenine aktariliyor.
y = input_data[:,4]

# Veriler normalize ediliyor.
input_data = standardize(input_data)

print('Normalize edilmis data : %s' % input_data[0:5,:])

# Xi input verisi icin boyut secimi gerceklesiyor
xi = input_data[:,[dim1,dim2]]

# Perceptron algoritmasi cagriliyor
ppn = Perceptron(epochs=epochs, eta=learning_rate)

# Perceptron ogrenme metodu calistiriliyor.
ppn.train(xi, y)
print('Agirliklar: %s' % ppn.w_)
print('Hatalar : %s' % ppn.errors_)

if showGraph:
    plot_draw(X=xi, y=y, pcn=ppn)
    plt.title('Perceptron Algoritmasi')
    plt.xlabel('SepalUzunluk[cm]')
    plt.ylabel('PetalUzunluk[cm]')
    plt.show()
```

perceptron.py

```
# encoding: utf-8
# module perceptron_project.main

"""
     Ercan Can
     Perceptron algoritmasi
"""

__author__      = "ercanc"

# imports
import numpy as np

# classes

class Perceptron(object):
    def __init__(self, eta=0.01, epochs=50):
        self.eta = eta
        self.epochs = epochs

    def train(self, X, y):

        # +1 ile bias tanimi yapilir. X.shape[1] ile 2 w belirtilir.
```

```
        self.w_ = np.zeros(1 + X.shape[1])
        self.errors_ = []

        # Wij (t+1) = Wij (t) + eta . err (p)
        for i in range(self.epochs):
            errors = 0

            for xi, target in zip(X, y):
                err = target - self.predict(xi)
                calculation = self.eta * err

                # w1,w2 weights
                self.w_[1:] += calculation * xi

                # bias weight
                self.w_[0] += calculation

                errors += int(calculation != 0.0)

            self.print_weights(self.w_, i)

            self.errors_.append(errors)

        return self

    # net input hesaplama sigma(xi*wi)+bias
    def net_input(self, X):
        return np.dot(X, self.w_[1:]) + self.w_[0]

    def predict(self, X):
        return np.where(self.net_input(X) >= 0.0, 1, 0)

    def print_weights(self, w, i):
        print("%s. %s" % (i, w))
```

functions.py

```
# encoding: utf-8
# module perceptron_project.functions

"""
     Ercan Can
     Program icin gerekli olan fonksiyonlari icerir
"""


__author__ = "ercanc"

# imports
from itertools import cycle

import matplotlib.pyplot as plt
```

```
import numpy as np
from enum import Enum
from matplotlib.colors import ListedColormap

# functions

# 0:Iris-setosa [50], 1:Iris-versicolor[100], 2:Iris-virginica[150]
def select_data(X, type):
    if type is SelectType.setosa_50_versicolor_0:
        print "Verinin ilk 50 kaydi olan Iris-setosa [50]
secildi..."
        return X[0:50]
    elif type is SelectType.setosa_10_versicolor_10:
        print "Veri icinde Iris-setosa [10], Iris-versicolor[10]
adet secildi"
        return np.concatenate((X[0:10], X[50:60]), axis=0)
    elif type is SelectType.setosa_20_versicolor_20:
        print "Veri icinde Iris-setosa [20], Iris-versicolor[20]
adet secildi"
        return np.concatenate((X[0:20], X[50:70]), axis=0)
    elif type is SelectType.setosa_30_versicolor_30:
        print "Veri icinde Iris-setosa [30], Iris-versicolor[30]
adet secildi"
        return np.concatenate((X[0:30], X[50:80]), axis=0)
    elif type is SelectType.setosa_50_versicolor_50:
        print "Verinin ilk 50 kaydi olan Iris-setosa [50] ve  Iris-
versicolor[50]  secildi..."
        return X[0:100]
    elif type is SelectType.setosa_versicolor_virginica_all:
        print "Verinin tamami secildi..."
        return X[0:150]
    else:
        print "Veri secilmedi !!!"


# Iris datasinin standardize edilmesi
def standardize(X):
    mu = np.mean(X, axis=0)
    std = np.std(X, axis=0)
    return (X - mu) / (std + 0.2)


"""
    iris[:,0] = (iris[:,0] - iris[:,0].mean()) / iris[:,0].std()
    iris[:,1] = (iris[:,1] - iris[:,1].mean()) / iris[:,1].std()
    iris[:,2] = (iris[:,2] - iris[:,2].mean()) / iris[:,2].std()
    iris[:,3] = (iris[:,3] - iris[:,3].mean()) / iris[:,3].std()
"""

# Grafik cizme
def plot_draw(X, y, pcn):
    colors = 'green,gray,yellow'  # renk listesi
    markers = 's^oxv<>'  # plot markerlar
    res = 0.01  # plot hassasiyet parametresi
    ax = plt.gca()
```

```
    marker_gen = cycle(list(markers))

    # make color map
    n_classes = np.unique(y).shape[0]
    colors = colors.split(',')
    cmap = ListedColormap(colors[:n_classes])

    # plot the decision surface
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, res),
                         np.arange(y_min, y_max, res))

    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    Z = pcn.predict(np.array([xx.ravel(), yy.ravel()]).T)

    Z = Z.reshape(xx.shape)
    ax.contourf(xx, yy, Z, alpha=0.3, cmap=cmap)

    ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.min(),
y_max=yy.max())

    # plot class samples
    for c in np.unique(y):
        y_data = X[y == c, 1]

        ax.scatter(x=X[y == c, 0],
                   y=y_data,
                   alpha=0.8,
                   c=cmap(c),
                   marker=next(marker_gen),
                   label=c)

    legend = plt.legend(loc=4,
                        fancybox=True,
                        framealpha=0.3,
                        scatterpoints=1,
                        handletextpad=-0.25,
                        borderaxespad=0.9)
    ax.add_artist(legend)

    return ax


# classes

# Verinin secimi kriterleri
class SelectType(Enum):
    setosa_50_versicolor_0 = 1
    setosa_10_versicolor_10 = 2
    setosa_20_versicolor_20 = 3
    setosa_30_versicolor_30 = 4
    setosa_50_versicolor_50 = 5
    setosa_versicolor_virginica_all = 6
```

UYGULAMANIN OUTPUTU

```
C:\Users\ercanc\AppData\Local\Continuum\Anaconda2\python.exe
C:/Users/ercanc/Desktop/DERS/DOGA_TEMELLI_HESAPLAMA/perceptron_proje
ct/main.py
Yuklenen data : [[ 5.1  3.5  1.4  0.2  0. ]
 [ 4.9  3.   1.4  0.2  0. ]
 [ 4.7  3.2  1.3  0.2  0. ]
 [ 4.6  3.1  1.5  0.2  0. ]
 [ 5.   3.6  1.4  0.2  0. ]]
Verinin ilk 50 kaydi olan Iris-setosa [50] ve  Iris-versicolor[50]
secildi...
Normalize edilmis data : [[-0.44246638  0.60266827 -0.89075539 -
0.76625924 -0.71428571]
 [-0.68099272 -0.13953403 -0.89075539 -0.76625924 -0.71428571]
 [-0.91951907  0.15734689 -0.95168257 -0.76625924 -0.71428571]
 [-1.03878225  0.00890643 -0.8298282  -0.76625924 -0.71428571]
 [-0.56172955  0.75110873 -0.89075539 -0.76625924 -0.71428571]]
0. [ 0.01        0.00034586  0.01668186]
1. [ 0.01        0.00034586  0.02338385]
2. [ 0.01        0.00034586  0.02338385]
3. [ 0.01        0.00034586  0.02338385]
4. [ 0.01        0.00034586  0.02338385]
5. [ 0.01        0.00034586  0.02338385]
6. [ 0.01        0.00034586  0.02338385]
7. [ 0.01        0.00034586  0.02338385]
8. [ 0.01        0.00034586  0.02338385]
9. [ 0.01        0.00034586  0.02338385]
10. [ 0.01        0.00034586  0.02338385]
11. [ 0.01        0.00034586  0.02338385]
12. [ 0.01        0.00034586  0.02338385]
13. [ 0.01        0.00034586  0.02338385]
14. [ 0.01        0.00034586  0.02338385]
15. [ 0.01        0.00034586  0.02338385]
16. [ 0.01        0.00034586  0.02338385]
17. [ 0.01        0.00034586  0.02338385]
18. [ 0.01        0.00034586  0.02338385]
19. [ 0.01        0.00034586  0.02338385]
20. [ 0.01        0.00034586  0.02338385]
21. [ 0.01        0.00034586  0.02338385]
22. [ 0.01        0.00034586  0.02338385]
23. [ 0.01        0.00034586  0.02338385]
24. [ 0.01        0.00034586  0.02338385]
25. [ 0.01        0.00034586  0.02338385]
26. [ 0.01        0.00034586  0.02338385]
27. [ 0.01        0.00034586  0.02338385]
28. [ 0.01        0.00034586  0.02338385]
29. [ 0.01        0.00034586  0.02338385]
30. [ 0.01        0.00034586  0.02338385]
31. [ 0.01        0.00034586  0.02338385]
32. [ 0.01        0.00034586  0.02338385]
33. [ 0.01        0.00034586  0.02338385]
34. [ 0.01        0.00034586  0.02338385]
35. [ 0.01        0.00034586  0.02338385]
```

```
36. [ 0.01        0.00034586  0.02338385]
37. [ 0.01        0.00034586  0.02338385]
38. [ 0.01        0.00034586  0.02338385]
39. [ 0.01        0.00034586  0.02338385]
40. [ 0.01        0.00034586  0.02338385]
41. [ 0.01        0.00034586  0.02338385]
42. [ 0.01        0.00034586  0.02338385]
43. [ 0.01        0.00034586  0.02338385]
44. [ 0.01        0.00034586  0.02338385]
45. [ 0.01        0.00034586  0.02338385]
46. [ 0.01        0.00034586  0.02338385]
47. [ 0.01        0.00034586  0.02338385]
48. [ 0.01        0.00034586  0.02338385]
49. [ 0.01        0.00034586  0.02338385]
Agirliklar: [ 0.01        0.00034586  0.02338385]
Hatalar : [3, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Process finished with exit code 0
```