**KNN Uygulamaları – Ödev 3**

**Python Kodu**

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

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

import math

import pandas as pd

import seaborn as sn

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

iris = load\_iris()

def visualize\_dataset():

features = iris['data']

labels = iris['target']

names = iris.target\_names

colors = ['r', 'g', 'b']

\_, (sepal, petal) = plt.subplots(ncols=2, figsize=(12, 5))

for i, clr in enumerate(colors):

sepal\_x = features[:,1][labels==i]

sepal\_y = features[:,2][labels==i]

petal\_x = features[:,0][labels==i]

petal\_y = features[:,1][labels==i]

sepal.scatter(sepal\_x, sepal\_y, c = clr)

petal.scatter(petal\_x, petal\_y, c = clr)

sepal.legend(names)

petal.legend(names)

sepal.set\_xlabel('Sepal length')

sepal.set\_ylabel('Sepal width')

petal.set\_xlabel('Petal length')

petal.set\_ylabel('Petal width')

plt.show()

def visualize\_accuracy(conf\_matrix):

cm = pd.DataFrame(conf\_matrix, index = iris.target\_names, columns=iris.target\_names)

sn.set(font\_scale=1.2)

sn.heatmap(cm, annot=True, annot\_kws={'size': 16}, cmap="Blues")

plt.show()

class KNN:

def \_\_init\_\_(self, k\_value):

self.k = k\_value

def dist(self, row0, row1): # euclid

total = 0

for i, j in zip(row0, row1):

total += math.pow(i-j, 2)

return math.sqrt(total)

def get\_nearest\_neighbors(self, row\_to\_search):

distances, neighbors = [], []

for i, x\_row in enumerate(self.x\_train):

d = self.dist(row\_to\_search, x\_row)

distances.append([d, i]) # dist, index

distances.sort(key = lambda x: x[0])

for i in range(self.k):

neighbors.append(distances[i])

return neighbors

def predict(self, X\_test, X\_train, Y\_train):

self.x\_train, self.y\_train = X\_train, Y\_train

y\_predict = []

for x\_row in X\_test:

neighbors = self.get\_nearest\_neighbors(x\_row)

targets = []

for n in neighbors:

ind = n[1]

targets.append(self.y\_train[ind])

y\_predict.append(max(targets, key = targets.count))

return y\_predict

iris = load\_iris()

# visualize\_dataset()

x\_train, x\_test, y\_train, y\_test = train\_test\_split(

iris['data'],

iris['target'],

test\_size = 0.33,

random\_state = 0)

knn = KNN(k\_value=5)

y\_pred = knn.predict(x\_test, x\_train, y\_train)

print(r2\_score(y\_pred, y\_test))

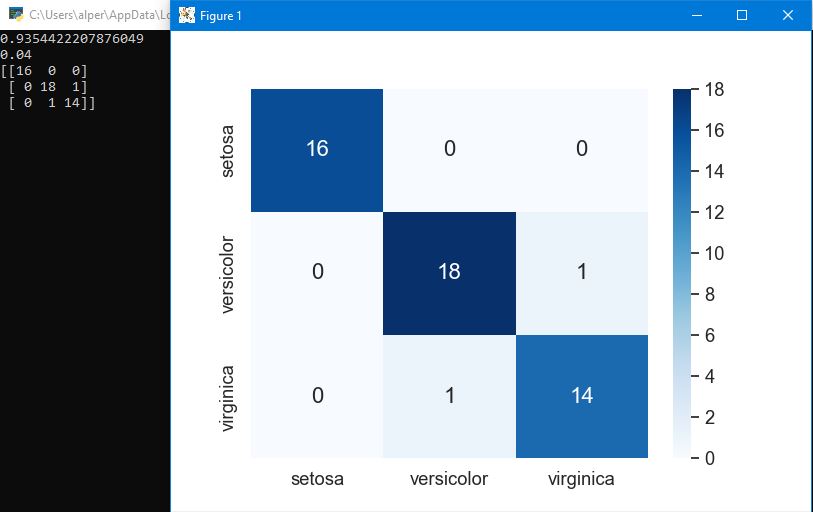
print(mean\_squared\_error(y\_pred, y\_test))

print(confusion\_matrix(y\_test, y\_pred))

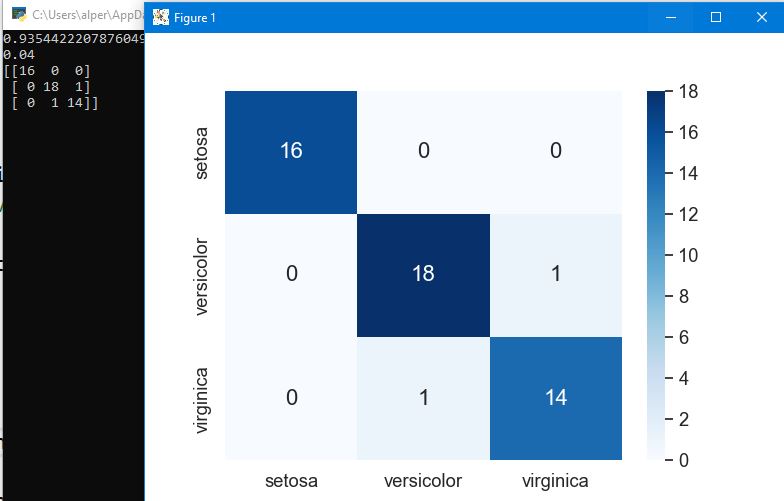
cm = confusion\_matrix(y\_test, y\_pred)

visualize\_accuracy(cm)

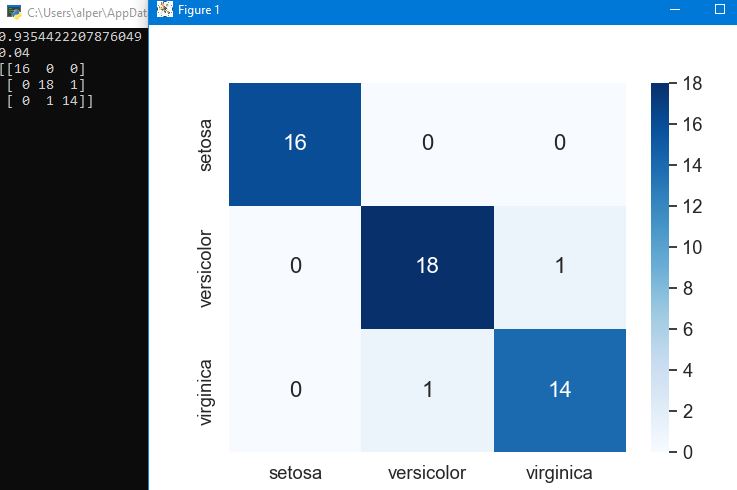
**K Değeri 1 iken**



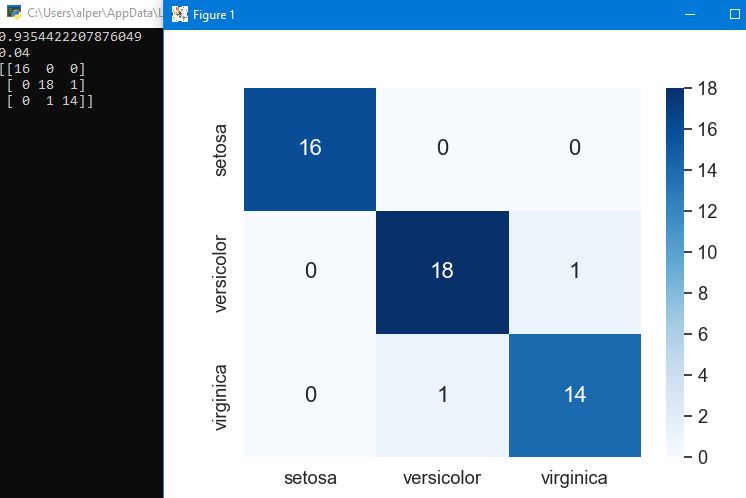
**K Değeri 2 iken**



**K Değeri 3 iken**



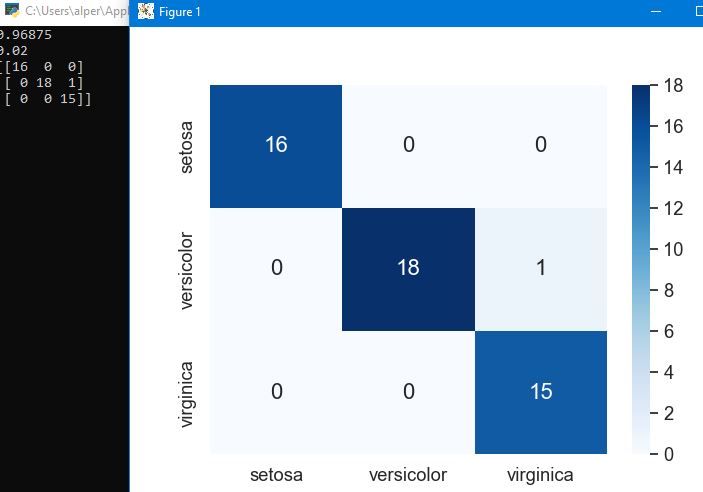
**K Değeri 4 iken**



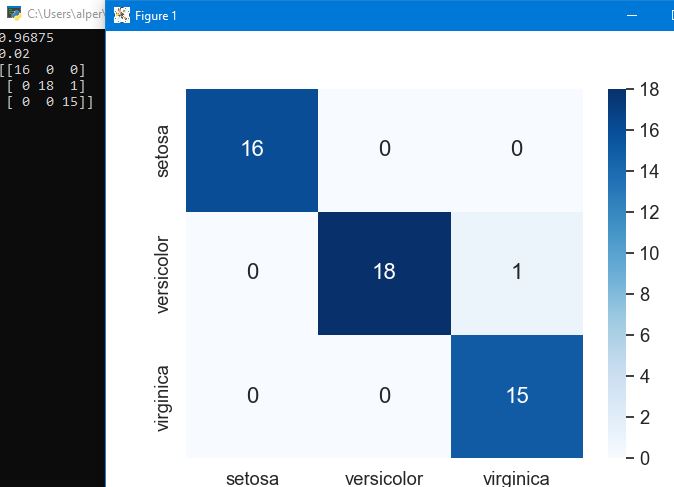
**K Değeri 5 iken**



**K Değeri 6 iken**



**K Değeri 7 iken**



**K Değeri 8 iken**



**K Değeri 9 iken**



**K Değeri 10 iken**



**K Değeri 11 iken**

