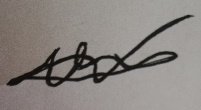
ÖRÜNTÜ TANIMA FİNAL SINAVI PYTHON PROJESİ

Volkan Tanışık

190106109049



YAPRAKLAR(LEAFS):

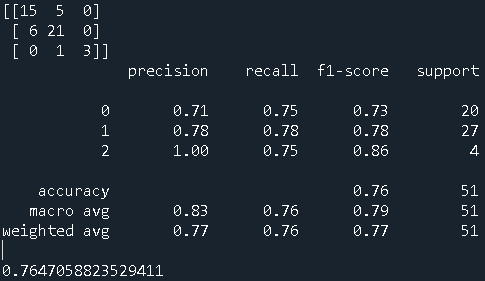
Veri setinin linki: <https://archive.ics.uci.edu/ml/datasets/Leaf>

Bu veri seti 40 farklı bitki türünün yapraklarını karşılaştırmaktadır.1’den 15’e ve 22’den 36’ya kadar basit yapraklar ve türleri sıralanırken 16’dan 21’e kadar ve 37’den 40’a kadar karmaşık yaprakları içermektedir.

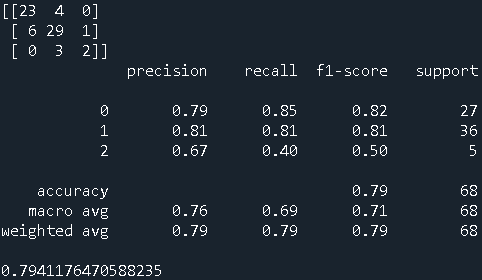
Veri setinde tür, numune numarası, eksantriklik, en boy oranı, uzama, sağlamlık, stokastik dışbükeylik, izoperimetrik faktör , maksimum girinti derinliği, lobluluk, ortalama yoğunluk, ortalama kontrast, pürüzsüzlük, üçüncü an, tekdüzelik, entropi özellikleri kullanılmıştır.

1.Uygulama K-en yakın komşu:

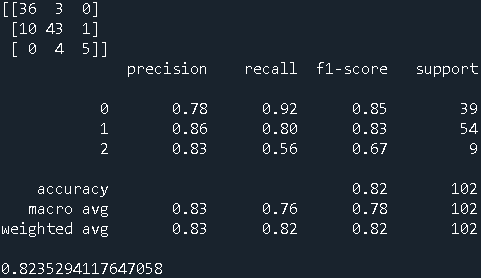
test size=0.15



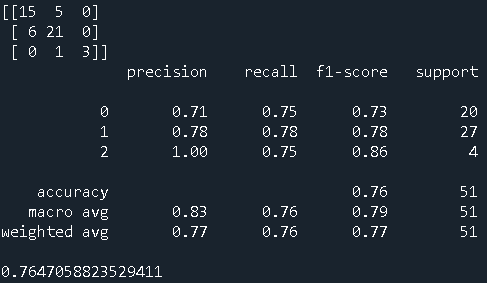
test size=0.2:



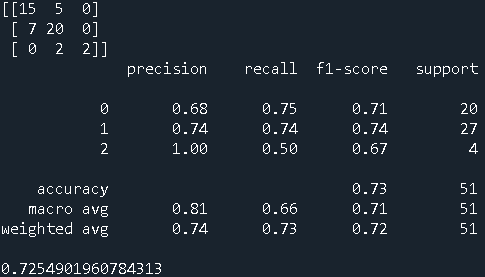
test size=0.3:



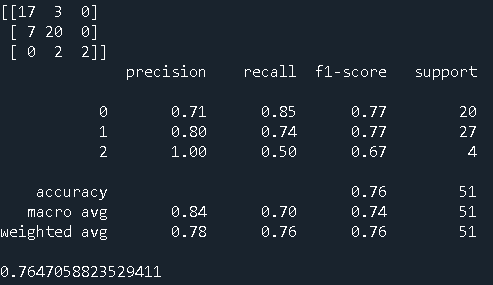
Komşuluk sayısı 3:



Komşuluk sayısı 5:

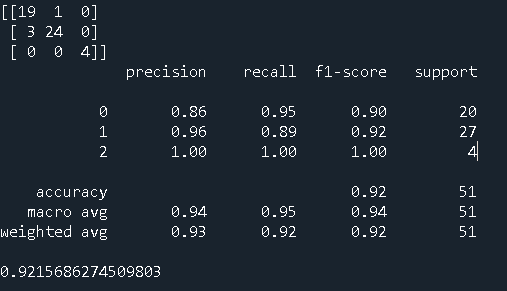


Komşuluk sayısı 7:

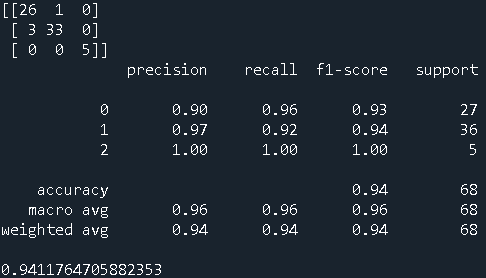


2. Uygulama Lojistik Regresyon:

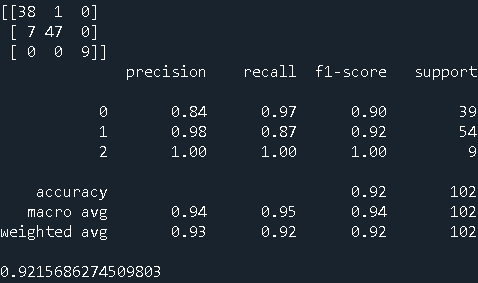
Test size 0.15:



Test size 0.2:

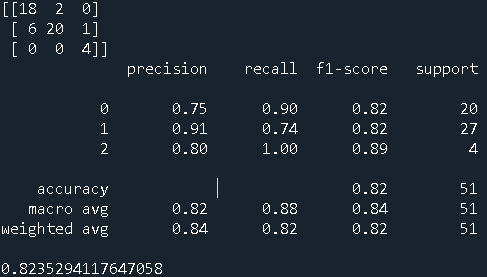


Test size 0.3:

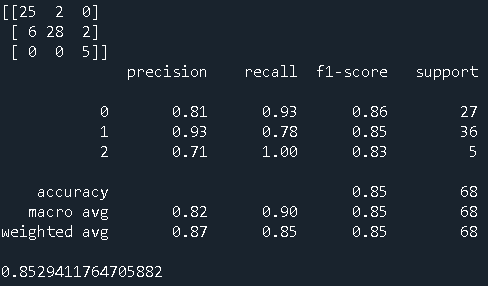


3. Uygulama Naive-bayes:

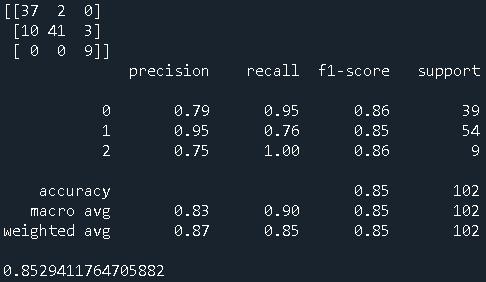
Test size 0.15:



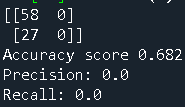
Test size 0.2:

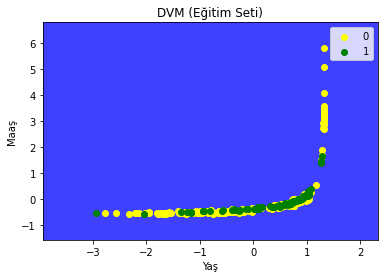


Test size 0.3:



4.Uygulama SVM(Destek Vektör Makinesi) Kernel lineer iken:





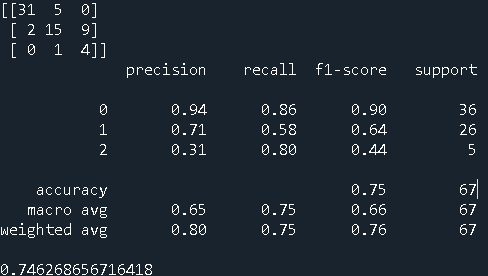
Kernel RBF iken:

Verilen giriş çıkış verileriyle Sonuç değişmedi fakat farklı giriş çıkış verileriyle fark gözlemlendi.

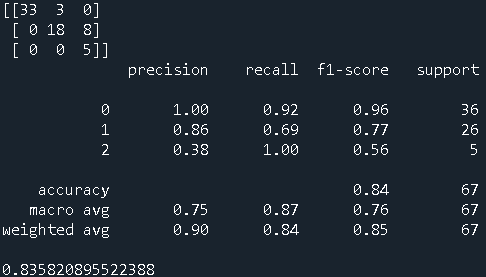
Çapraz Doğrulama Yöntemleri:

K-Fold yönteminde k-fold sayisi 5 iken:

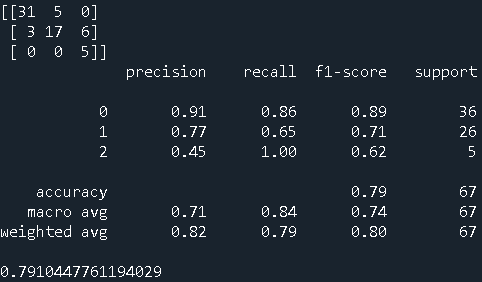
K-en yakın komşu(3):



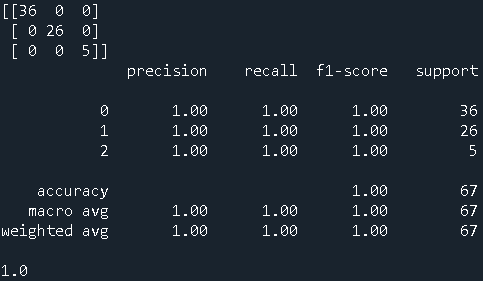
Lojistik Regresyon(0):



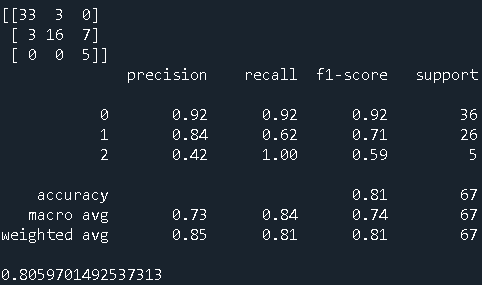
Naive Bayes:



Decision Tree:

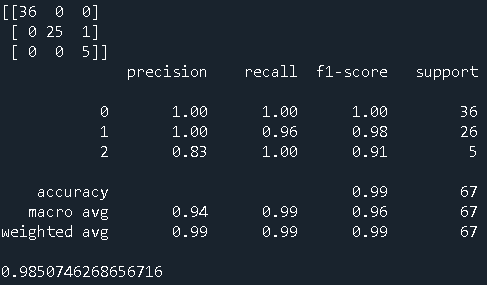


SVM:

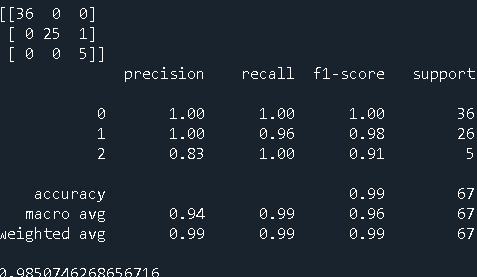


Random Forest:

Random Forest Classifier estimor sayisi 50:

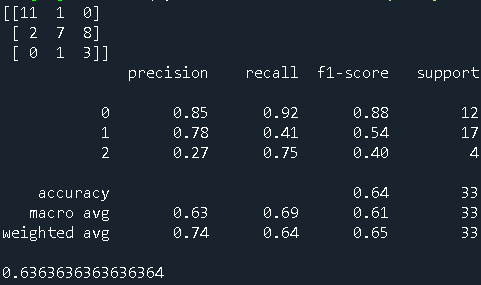


Random Forest Classifier estimor sayisi 100:

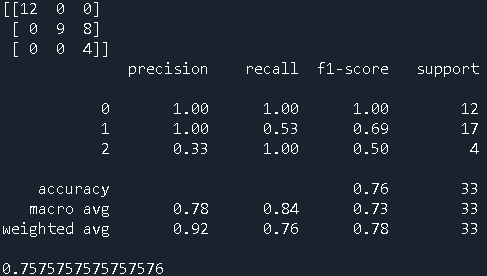


K-fold yönteminde k-fold sayisi 10 iken:

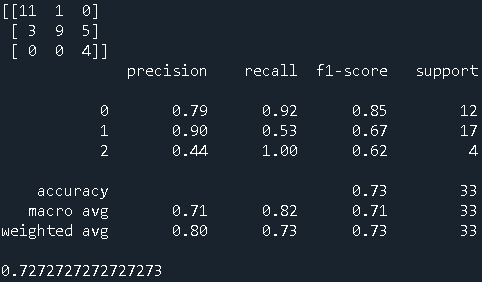
K en yakın komşu(3):



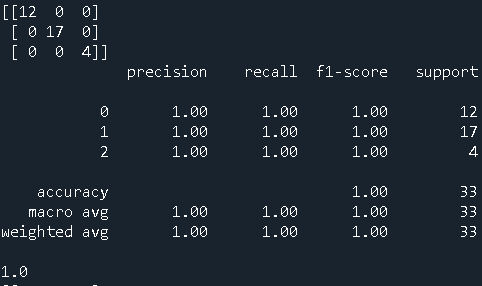
Lojistik Regresyon(0):



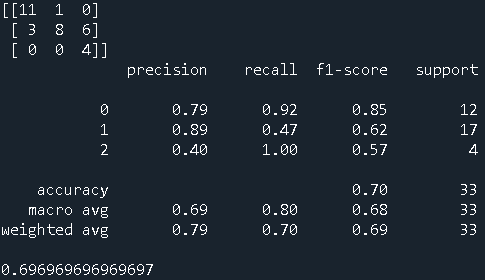
Naive Bayes:



Decision Tree:

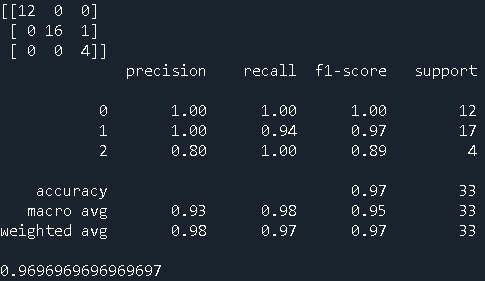


SVM:



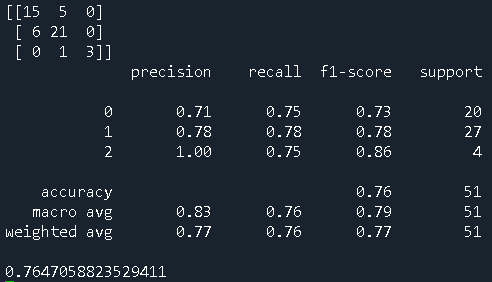
Random Forest:

Random Forest Classifier estimor sayisi 50:

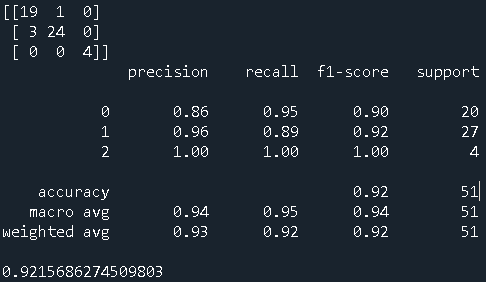


LeaveOneOut yöntemi test size 0.15 iken:

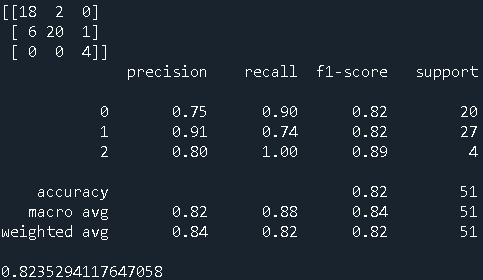
K-en yakın komşu(3):



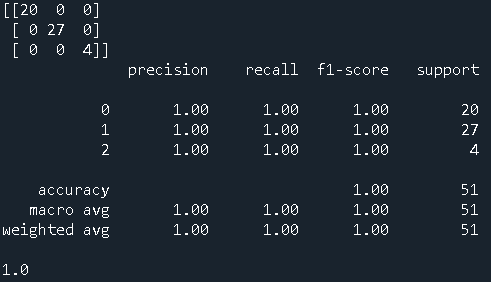
Lojistik Regresyon(0):



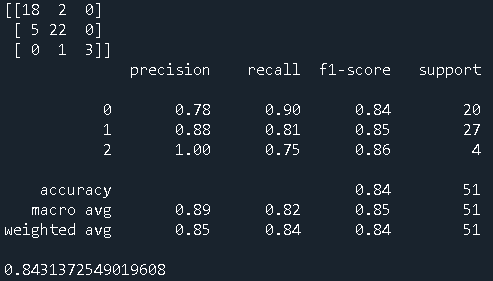
Naive Bayes:



Decision Tree:

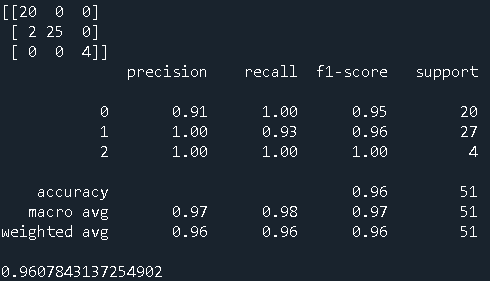


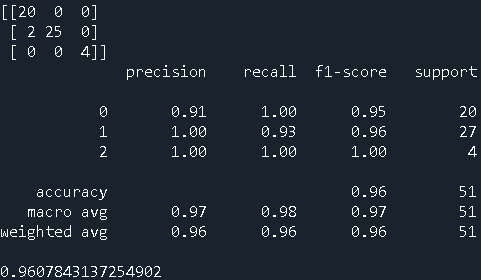
SVM:



Random Forest:

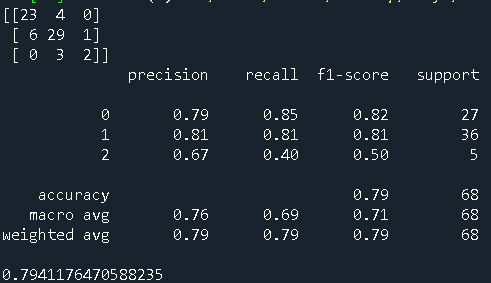
Random Forest Classifier estimor sayisi 100:



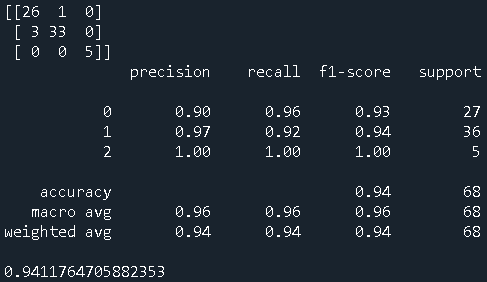
Random Forest Classifier estimor sayisi 50:

Leave One Out Yöntemi test size 0.2 iken:

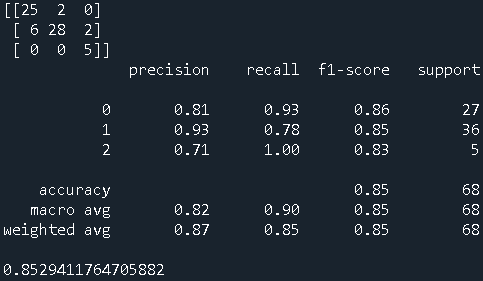
K En Yakın Nokta Yöntemi:



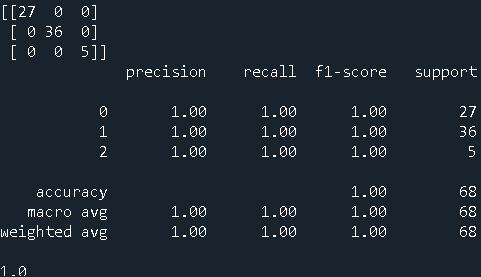
Lojistik Regresyon Yöntemi:



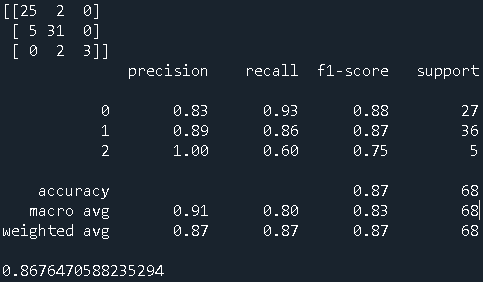
Naive Bayes Yöntemi:



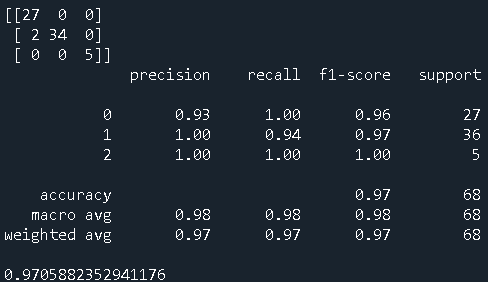
Decision Tree yöntemi:



SVC(Destek Vektör Makineleri) Yöntemi:

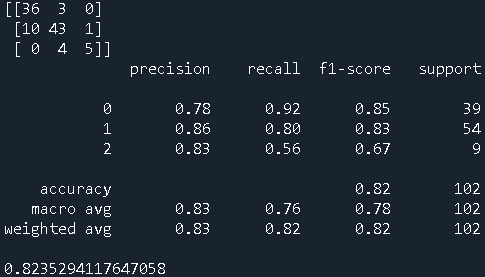


Random Forest Yöntemi estimator sayısı 50 iken:

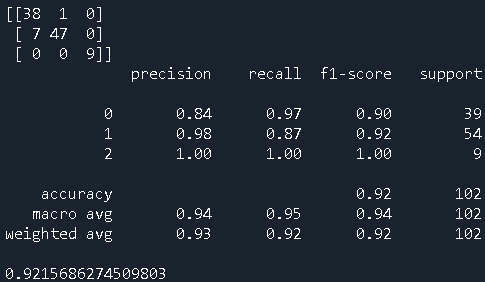


Leave One Out Yöntemi Test size 0.3 iken:

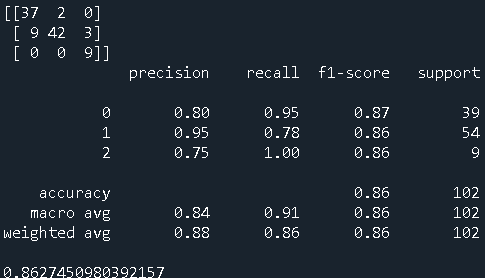
K En Yakın Yöntemi:

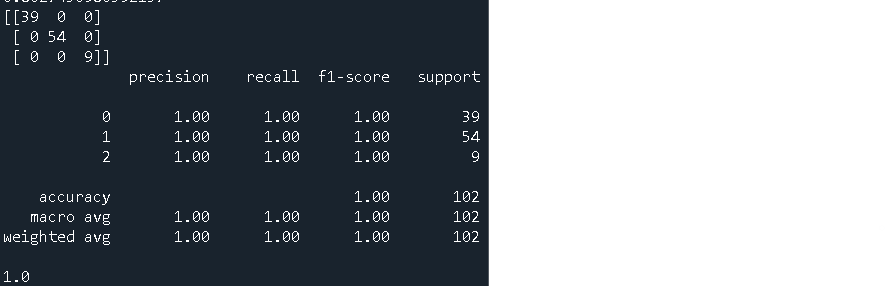


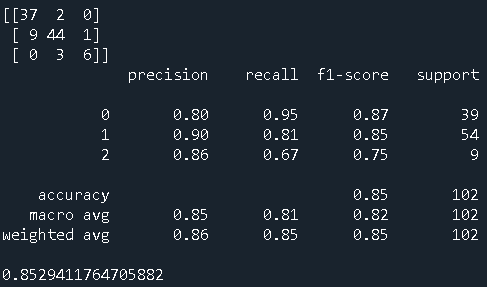
Logistic Regresyon Yöntemi:



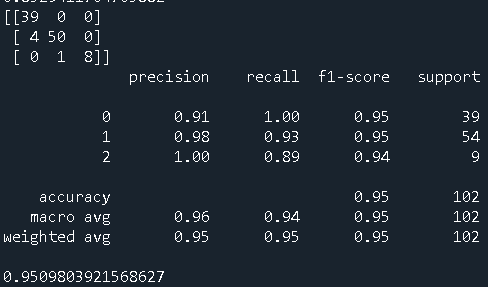
Naive Bayer Yöntemi:



Deicision Tree Yöntemi:

SVC(Destek Vektör Makineleri Yöntemi):

Random Forest Yöntemi:



KULLANILAN KODLAR:

K En Yakın Nokta:

#Kütüphaneleri yükle

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn.linear\_model import LogisticRegression

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

from sklearn.metrics import accuracy\_score

from sklearn.metrics import f1\_score

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

#Dosyayi Yukle

veri = pd.read\_csv('C:\\Users\\VOLKAN\\Desktop\\leafs\\leaf.csv')

ozellik\_sayisi = 16

#giris cikis belirle

giris\_verileri = veri.iloc[:,1:ozellik\_sayisi+1]

cikis = veri.iloc[:,-1].astype('int')

#Egitim ve test verilerini ayir

egitim\_giris, test\_giris,egitim\_cikis, test\_cikis = train\_test\_split(giris\_verileri,cikis, test\_size=0.15, random\_state=0)

#Standardizasyon

scaler = preprocessing.StandardScaler()

stdGiris = scaler.fit\_transform(egitim\_giris)

stdTest = scaler.transform(test\_giris)

#K En Yakın Komşu

from sklearn.neighbors import KNeighborsClassifier

siniflandirici = KNeighborsClassifier(n\_neighbors=3)

siniflandirici.fit(stdGiris, egitim\_cikis)

cikis\_tahmin=siniflandirici.predict(stdTest)

#Başarıyı belirle

basari = accuracy\_score(test\_cikis, cikis\_tahmin)

fSkor = f1\_score(test\_cikis, cikis\_tahmin, labels=None, pos\_label=1, average='micro', sample\_weight=None)

print(confusion\_matrix(test\_cikis, cikis\_tahmin))

print(classification\_report(test\_cikis, cikis\_tahmin))

print(accuracy\_score(test\_cikis, cikis\_tahmin))

Lojistik Regresyon:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn.linear\_model import LogisticRegression

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

from sklearn.metrics import accuracy\_score

from sklearn.metrics import f1\_score

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

#Dosyayi Yukle

veri = pd.read\_csv('C:\\Users\\VOLKAN\\Desktop\\leafs\\leaf.csv')

ozellik\_sayisi = 16

#giris cikis belirle

giris\_verileri = veri.iloc[:,1:ozellik\_sayisi+1]

cikis = veri.iloc[:,-1].astype('int')

#Egitim ve test verilerini ayir

egitim\_giris, test\_giris,egitim\_cikis, test\_cikis = train\_test\_split(giris\_verileri,cikis, test\_size=0.15, random\_state=0)

#Standardizasyon

scaler = preprocessing.StandardScaler()

stdGiris = scaler.fit\_transform(egitim\_giris)

stdTest = scaler.transform(test\_giris)

#Logistic Regression

log\_reg = LogisticRegression(random\_state=0)

log\_reg.fit(stdGiris,egitim\_cikis)

cikis\_tahmin=log\_reg.predict(stdTest)

#Başarıyı belirle

basari = accuracy\_score(test\_cikis, cikis\_tahmin)

fSkor = f1\_score(test\_cikis, cikis\_tahmin, labels=None, pos\_label=1, average='weighted', sample\_weight=None)

print(confusion\_matrix(test\_cikis, cikis\_tahmin))

print(classification\_report(test\_cikis, cikis\_tahmin))

print(accuracy\_score(test\_cikis, cikis\_tahmin))

Naive Bayes Yöntemi:

import pandas as pd

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

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import f1\_score

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

#Dosyayi Yukle

veri = pd.read\_csv('C:\\Users\\VOLKAN\\Desktop\\leafs\\leaf.csv')

ozellik\_sayisi = 16

#giris cikis belirle

giris\_verileri = veri.iloc[:,1:ozellik\_sayisi+1]

cikis = veri.iloc[:,-1].astype('int')

#Egitim ve test verilerini ayir

egitim\_giris, test\_giris,egitim\_cikis, test\_cikis = train\_test\_split(giris\_verileri,cikis, test\_size=0.15, random\_state=0)

#Gaussian NB (Naive Bayes)

olasilik\_modeli = GaussianNB()

olasilik\_modeli.fit(egitim\_giris, egitim\_cikis)

cikis\_tahmin = olasilik\_modeli.predict(test\_giris)

#Başarıyı belirle

basari = accuracy\_score(test\_cikis, cikis\_tahmin)

fSkor = f1\_score(test\_cikis, cikis\_tahmin, labels=None, pos\_label=1, average='weighted', sample\_weight=None)

print(confusion\_matrix(test\_cikis, cikis\_tahmin))

print(classification\_report(test\_cikis, cikis\_tahmin))

print(accuracy\_score(test\_cikis, cikis\_tahmin))

SVM(Destek Vektör Makinesi) Yöntemi:

#Kütüphaneleri yükle

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import os

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

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn import metrics

from sklearn.metrics import confusion\_matrix

from matplotlib.colors import ListedColormap

#Dosyayi Yukle

os.chdir('C:\\Users\\VOLKAN\\Desktop\\leafs')

dataset = pd.read\_csv('C:\\Users\\VOLKAN\\Desktop\\leafs\\leaf.csv')

#Giris cikis belirle

X = dataset.iloc[:, [5,10]].values

y = dataset.iloc[:, 6].values.astype('int')

#Egitim ve test verilerini ayir

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

#Standardizasyon

sc\_X = StandardScaler()

X\_train = sc\_X.fit\_transform(X\_train)

X\_test = sc\_X.transform(X\_test)

#Destek Vektör Makinesi

classifier = SVC(kernel='linear', random\_state = 0)

classifier.fit(X\_train, y\_train)

#Başarıyı belirle

y\_pred = classifier.predict(X\_test)

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

print(cm)

print("Accuracy score %.3f" %metrics.accuracy\_score(y\_test, y\_pred))

print("Precision:",metrics.precision\_score(y\_test, y\_pred))

print("Recall:",metrics.recall\_score(y\_test, y\_pred))

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('blue', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('yellow', 'green'))(i), label = j)

plt.title('DVM (Eğitim Seti)')

plt.xlabel('Yaş')

plt.ylabel('Maaş')

plt.legend()

plt.show()

Leave One Out Yöntemi:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

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

from sklearn.metrics import accuracy\_score

from sklearn.metrics import f1\_score

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import LeaveOneOut

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report

#Dosyayi Yukle

veri = pd.read\_csv('C:\\Users\\VOLKAN\\Desktop\\leafs\\leaf.csv')

ozellik\_sayisi = 16

#giris cikis belirle

giris\_verileri = veri.iloc[:,1:ozellik\_sayisi+1]

cikis = veri.iloc[:,-1].astype('int')

#Egitim ve test verilerini ayir

egitim\_giris, test\_giris,egitim\_cikis, test\_cikis = train\_test\_split(giris\_verileri,cikis, test\_size=0.15, random\_state=0)

#Standardizasyon

scaler = preprocessing.StandardScaler()

stdGiris = scaler.fit\_transform(egitim\_giris)

stdTest = scaler.transform(test\_giris)

siniflandiricilar=[KNeighborsClassifier(n\_neighbors=3), LogisticRegression(random\_state=0), GaussianNB(), DecisionTreeClassifier(), SVC(), RandomForestClassifier(n\_estimators=100,)]

basari=list()

fSkor = list()

for i in range(6):

siniflandiricilar[i].fit(stdGiris, egitim\_cikis)

cikis\_tahmin = siniflandiricilar[i].predict(stdTest)

print(confusion\_matrix(test\_cikis, cikis\_tahmin))

basari.append(accuracy\_score(test\_cikis, cikis\_tahmin))

fSkor.append( f1\_score(test\_cikis, cikis\_tahmin, labels=None, pos\_label=1, average='weighted', sample\_weight=None))

print(classification\_report(test\_cikis, cikis\_tahmin))

print(accuracy\_score(test\_cikis, cikis\_tahmin))

K-fold Yöntemi:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

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

from sklearn.metrics import accuracy\_score

from sklearn.metrics import f1\_score

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import KFold

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report

#Dosyayi Yukle

veri = pd.read\_csv('C:\\Users\\VOLKAN\\Desktop\\leafs\\leaf.csv')

ozellik\_sayisi = 16

#giris cikis belirle

giris\_verileri = veri.iloc[:,1:ozellik\_sayisi+1]

cikis = veri.iloc[:,-1].astype('int')

siniflandiricilar=[KNeighborsClassifier(n\_neighbors=3), LogisticRegression(random\_state=0), GaussianNB(), DecisionTreeClassifier(), SVC(), RandomForestClassifier(n\_estimators=50,)]

basari=list()

kSayisi = 10

kf = KFold(n\_splits=kSayisi)

fSkor = list()

for i in range(6):

toplamBasari = 0;

toplamfSkor = 0;

for egitim\_index, test\_index in kf.split(giris\_verileri):

#Standardizasyon

scaler = preprocessing.StandardScaler()

stdGiris = scaler.fit\_transform(giris\_verileri.iloc[egitim\_index,:])

stdTest = scaler.transform(giris\_verileri.iloc[test\_index,:])

siniflandiricilar[i].fit(stdGiris, cikis[egitim\_index] )

cikis\_tahmin = siniflandiricilar[i].predict(stdTest)

toplamBasari += (accuracy\_score(cikis[test\_index], cikis\_tahmin))

toplamfSkor += ( f1\_score(cikis[test\_index], cikis\_tahmin, labels=None, pos\_label=1, average='weighted', sample\_weight=None))

print(confusion\_matrix(cikis[test\_index], cikis\_tahmin))

basari.append(toplamBasari/kSayisi)

fSkor.append(toplamfSkor/kSayisi)

print(classification\_report(cikis[test\_index], cikis\_tahmin))

print(accuracy\_score(cikis[test\_index], cikis\_tahmin))