 **ERCİYES ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**ÖRÜNTÜ TANIMAYA GİRİŞ DERSİ**

**ÖDEV RAPORU**

**KAAN ÇOLAK - 1030516475**

Python bilgimin üzerine bir çok kütüphane araştırması yaptım. Türkçe , İngilizce ve Almanca kaynakların veri bilimi ile ilgili blog , medium ve stackoverflow üzerindeki paylaşımları inceledim. Hatalarımın alternatiflerini de yine bu alanlarda bulabildim.

**Kodlar:**

import pandas as pd

import numpy as np

import pandas as pd

#pandas kullanarak excelde daha önceden transpose ettiğim verilerimi okuttum:

datas = pd.read\_excel('otu.xlsx','Sheet1')

#Bağımsız niteliği X, Bağımlıyı ise y değişkenine atadım,

#Eğitim ve test setlerini ayırdım.%20 test için ayrılırken, %80lik kısmını ise veri için ayırdım:

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

datas\_train, datas\_test= train\_test\_split(datas,test\_size=0.20)

X\_train=datas\_train.iloc[:,1:]

X\_test=datas\_test.iloc[:,1:]

y\_train=datas\_train.iloc[:,0]

y\_test=datas\_test.iloc[:,0]

#Verileri normalize/standardize edelim:

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

sc.fit(X\_train)

X\_train=sc.transform(X\_train)

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

#Model için k en yakın komşu algoritmasını kullandım ve bununla alakalı scikit-learn sınıfını kullandım,

#Ayrıca parametre de verdim :

from sklearn.neighbors import KNeighborsClassifier

enyakin\_komsu = KNeighborsClassifier(n\_neighbors=5)

#Modeli eğittim:

enyakin\_komsu.fit(X\_train, y\_train)

#Test seti ile hedef sınıfları tahmin ettim:

y\_pred=enyakin\_komsu.predict(X\_test)

#Hata matrisi(confusion matrix) ile modelimin başarısını ölçtüm,

#Tahmin sonuçlarımı test sonuçları ile karşılaştırdım:

from sklearn.metrics import classification\_report, confusion\_matrix

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

print(classification\_report(y\_test,y\_pred))

import numpy as np

#sınıftaki her bir farkli eleman icin (left/right) hesaplanir:

for j in np.unique(datas.iloc[:,0]):

#o sınıf değeri için gözlem sayısı:

summ=len([i for i in range(0,len(y\_test)) if y\_test.iloc[i]==j])

# o sınıf değeri için yapılan başarılı tahmin sayısı:

pred\_success=len([i for i in range(0,len(y\_test)) if y\_test.iloc[i]==j and y\_test.iloc[i]==y\_pred[i]])

if j=="left":

# eğer ki sınıfı "left" ise ([birinci sınıf başarılı tahmin sayısı]/[birinci sınıf toplam gözlem sayısı]) sensitivity olarak sonuçlar yazilir:

print("sensitivity: "+str(round(pred\_success\*100/summ))+"%.")

else:

# eğer ki sınıf "right" ise ([ikinci sınıf başarılı tahmin sayısı]/[ikinci sınıf toplam gözlem sayısı]) specifity olarak sonuçlar yazılır:

print("specifity: "+str(round(pred\_success\*100/summ))+"%.")

**Çıktı:**

runfile('D:/2020-2021/Lessons/Introduction to Pattern Recognition/Minterm/untitled6.py', wdir='D:/2020-2021/Lessons/Introduction to Pattern Recognition/Minterm')

[[30 2]

[17 6]]

precision recall f1-score support

left 0.64 0.94 0.76 32

right 0.75 0.26 0.39 23

accuracy 0.65 55

macro avg 0.69 0.60 0.57 55

weighted avg 0.69 0.65 0.60 55

sensitivity: 94%.

specifity: 26%.