İRİS DATASETİNİN KNN VE SVM ALGORİTMALARI İLE İNCELENMESİ

- ## KNN ALGORİTMASI (K –NEAREST NEIGHBORS)

KNN algoritması , sınıflandırma için kullanılan denetimli makine öğrenme algoritmasıdır. Komşulara bakarak tahmin yürütülür ve benzer olan sınıflar birbirlerine yakın olanlardır. KNN içinde tahmin edilecek değeri en yakın komşularından hangisinde yoğun olduğu bilgisine dayanarak sınıfı tahmin etmeye çalışılır.

Sınıflandırmanın yapılabilmesi için çeşitli parametrelere ihtiyaç duyulur: uzaklık,komşuluk sayısı(k) ve ağırlıklandırma ölçütleri .

Uzaklığı tahmin edilecek değerin diğer noktalara uzaklıkları hesaplanır.Bunun için Öklid,Minkowski,Manhattan vb gibi uzaklık hesaplamaları kullanılabilir.

Komşuluk sayısı(k) KNN classı için önemli bir parametredir. Bu parametreye dayalı olarak sınıflandırmalar yapılmaktadır. K=1 alındığında hedef değer en yakın komşuya atanır. K nin gittikçe hedef değere yaklaşması veya uzaklaşmasına bağlı olarak veri setindeki bilgiler dahilinde sınıf atamaları ve seçimleri yapılmaktadır.

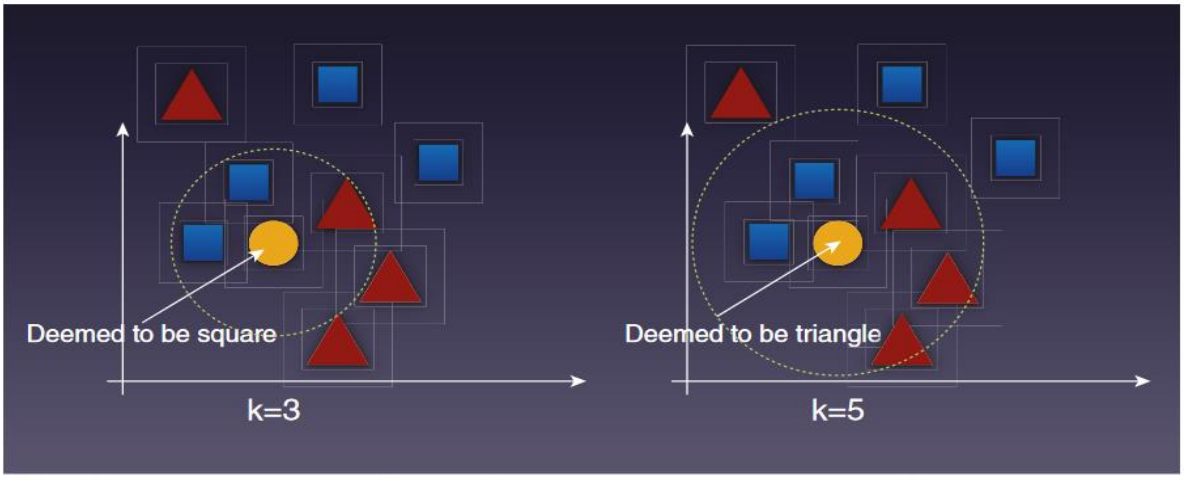


Figure 9.1: The classification of a point depends on the majority of its neighbors

K parametresinin çok düşük seçildiği durumlarda model overfit olabilmektedir. Yani model eğitim setindeki durumları ezberlemiştir ve test edilen veri setinde bu durumları aramaktadır.Bu da test veri setinde kötü tahmin skorlarına sebep olabilir.k değerinin çok yüksek seçildiği durumda ise model underfit olabilmektedir. Model eğitim verilerine uymaz ve yeni verilen için genelleştirme yapılamaz. Bu sebeple hem eğitim hem de test veri setinde problemlerle karşılaşabilinmektedir. Bu sebeplerle k değerinin olabilecek en olası optimum değer seçilmesi gerekmektedir.

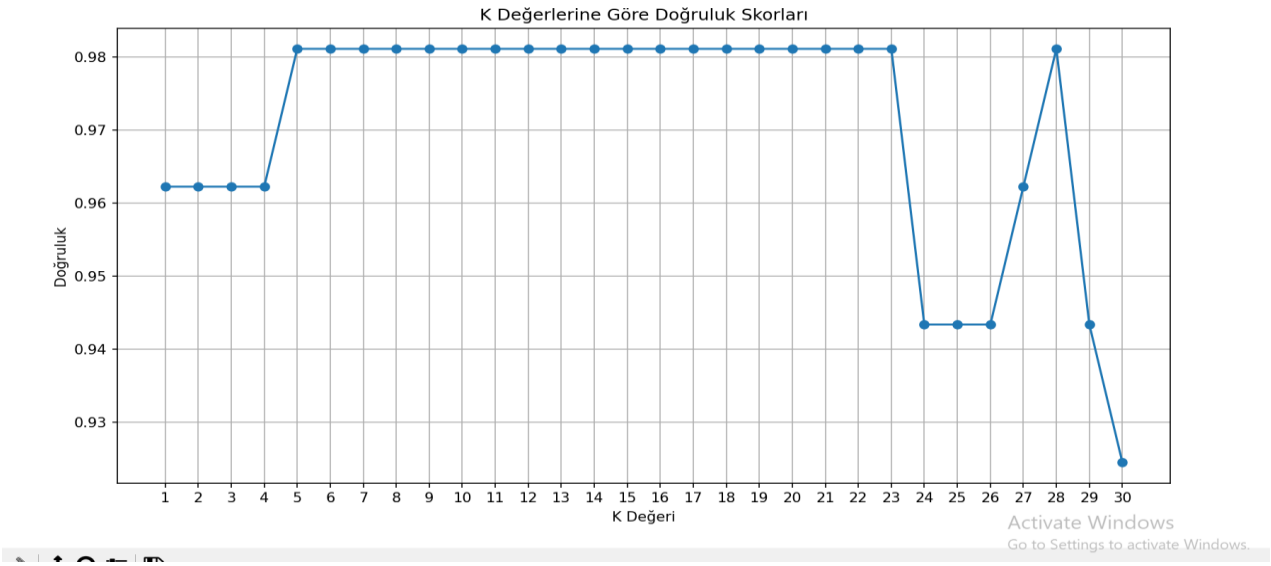
Diğer parametre olan ağırlık (weight) için uniform ve distance seçenekleri vardır. Ağırlığın uniform seçilmesi durumunda komşulukta bulunan tüm ağırlıklar eşit olacaktır.Distance seçilmesi durumunda ise komşulukta bulunan tüm komşuların ağırlığı dataya olan mesafe azaldıkça artacaktır. Bu hesaplama ise tüm komşuların ağırlığının $\frac{1}{d}$ veya $\frac{1}{d^2}$ şeklinde alınması ile yapılmaktadır. (d = hedef dataya mesafe)

KNN algoritmasının iris dataseti üzerinde eğitildiği ve test edildiğini gösteren kod aşağıdaki gibidir :

```
#ilgili kütüphanelerin yüklenmesi
import numpy as np
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

#İris veri setinin yüklenmesi
iris = load_iris()

x = iris.data    #veri setinin özellikleri x e atanır
y = iris.target  #veri setinin hedef değişkenleri y e atanır



#Veri Setleri eğitim ve test seti olarak ikiye ayrılır.X_train ve Y_train değişkenleri modelin
öğrenmesi için kullanılacak olan değişkenlerdir. X_test ve Y_test ise modelin performansını
değerlendirmek için kullanılan parametrelerdir.

X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.35, random_state=0)

#Veri setinin %35 ı test seti olarak ayrılmıştır.Bu da test_size parametresi ile belirlenmiştir.
#random_state parametresi belirlendiğinde train_test_split fonksiyonu veri setini aynı şekilde böler

# En iyi k değerini bulan fonksiyon
def find_best_k(accuracy_table):

    best_k = accuracy_table.argmax()+1
    best_accuracy = accuracy_table.max()
    print(f" The best K value is {best_k} with {best_accuracy:.2f} accuracy(max)")



def evaluate_knn(K_size, X_train, X_test, Y_train, Y_test):

    accuracy_table = np.zeros(K_size)  # K_size boyutunda dizi oluşturulur

    for i in range(1, K_size+1):  # 1'den K_size+1'e kadar olan tüm k değerleri için döngü
```

```
        neighbor = KNeighborsClassifier(n_neighbors= i)  # KNeighborsClassifier nesnesi içindeki
n_neighbors parametresi bu algoritmada kullanılacak olan komşu sayısını belirtir.

        neighbor.fit(X_train, Y_train) #Modelin eğitim verileri ile eğitilmesi

        y_pred = neighbor.predict(X_test) #Model test verisi üzerinde tahmin yapar

        accuracy_table[i-1] = accuracy_score(Y_test, y_pred) #her bir k değeri için hesaplanan
doğruluk değeri diziye atılır

    return accuracy_table


# K-NN alogritması ile farklı k değerlerini değerlendirerek en iyi k değeri bulunmasını sağlar
accuracy_table = evaluate_knn(20, X_train, X_test, Y_train, Y_test) #değerlendirilecek k değeri 20
alınımıştır.

for k,accuracy in enumerate(accuracy_table,start=1):
  print(f"Accuracy for K = {k} : {accuracy:.2f}")

find_best_k(accuracy_table) # max k değeri
```

Yukarıdaki kodda hespalanan doğruluk skorlarının gösterildiği grafik aşağıdaki gibidir.

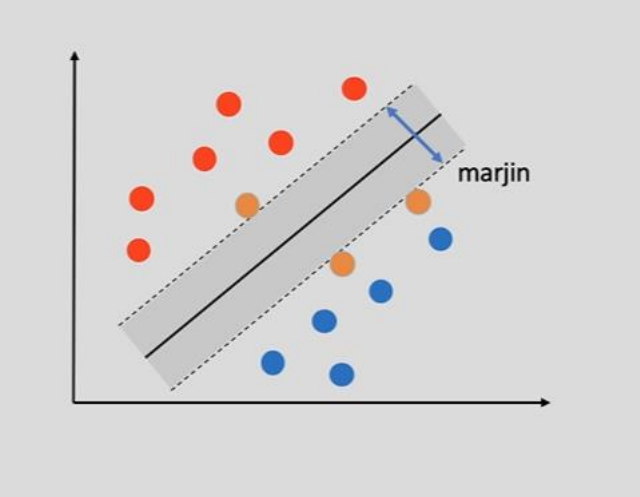


Grafiktede görüldüğü gibi K değerinin 5 olduğu durumda ilk maximum başarı değerine ulaşılmıştır.

Kullanım açısından KNN algoritması en yaklaşık değeri vermesine rağmen bellek kullanımı , veri seti ,maliyette artış ve birçok parametreye bağlı olarak etkilenmesi dezavantajlarına da sahiptir.

- SVM ALGORİTMASI (Support Vector Machine)

SVM algoritması iki sınıf arasındaki en ayırt edici sınırın olmasını sağlayan hiper-düzlemin bulunması ile yürütülen sınıflandırma algoritmasıdır.



İki ya da daha fazla sınıflı veri kümelerinde , sınıfları ayırma işlemi SVM modellemesiyle yapılabilir. Ayrım içi belirlenmesi gereken karar sınırı ve en yakın veri noktası arasındaki mesafeye marjin denir. SVM iki sınıfı ayırabilmek için marjini en iyi olacak şekilde ayarlayarak bir hiper-düzlem oluşturur.Hiper-düzlem oluştururken karar sınırına en yakın ve destek olabilecek destek vektörleri olan veri noktaları kullanılır.

SVC sınıflandırılması için kullanılan önemli parametreler şu şekildedir:

C = Düzenleştirme parametresi.Bu parametre SVM modelinin karmaşıklığı kontrol etmesine ve overfitting problemlerini giderilmesine yardımcı olur.C değerinin artmasıyla SVM modelinin eğitilmesi de artacaktır. C değeri marjin genişliğinin etkilediği için C değerinin artması marjin değerinde küçülmeye sebep olacaktır.

Kernel = Veri noktalarını ,düzlemsel olarak işlevsel hale getirmek için kullanılan matematiksel işlemlerdir.Verilerin ayrılma şekillerine göre farklı değerler almaktadır. Veri doğrusal olarak ayrılabiliyorsa *'linear'* değerini almalıdır. Doğrusal olarak ayrılmayan bir veri seti varsa *'poly', 'rbf', 'sigmoid', 'precomputed'* değerleri kullanılır.

Degree = Eğer baz alınan yöntem polinomsal yöntemse , bu parametre polinomun derecesini belirtir.

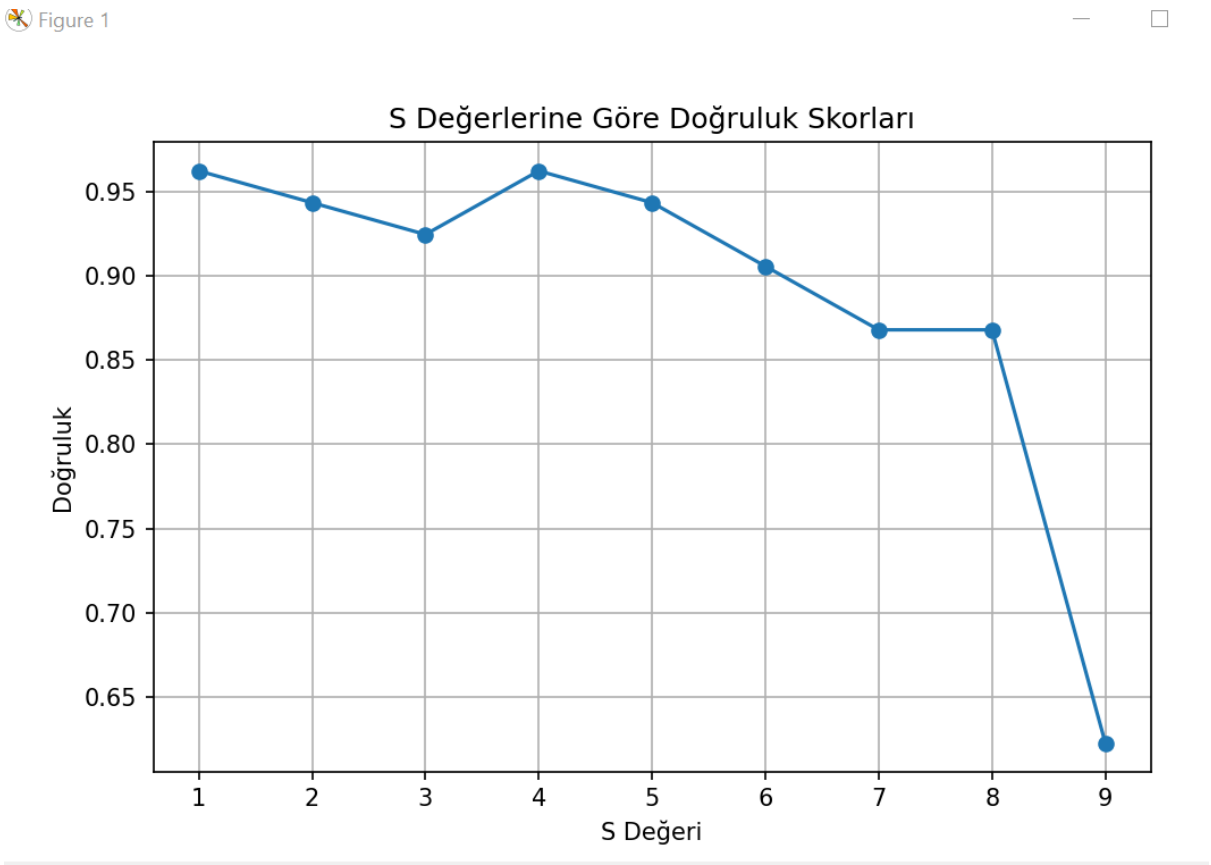
Gamma = Eğer baz alınan yöntem RBF ise gamma RBF in genişliğini belirler ve karar sınırlarının esnekliğini ayarlar.

SVM modelleri yüksek eğitim süreleri sebebiyle büyük veri kümeleri için uygun değildir.Sınıflararası çakışma durumlarında da çalışma durumu iyi değildir. Fakat diğer modellemelere göre daha az bellek tüketimi ve hızlı tahmin görüsü sayesinde iyi çalışmalar gösterir.

SVM modelinin iris dataseti üzerinde eğitim ve test sürecini gösteren kod parçası aşağıdadır :

```
#ilgili kütüphanelerin yüklenmesi
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn import metrics

iris = load_iris() #iris veri setinin yüklemesi
x,y = iris.data,iris.target
k_size = range(1,11) #k aralığı
c = 1.0 #C parametresi Margin genişliğini belirlemektedir.
accuracy= np.zeros(len(k_size)) #başarı oranlarının tutulduğu dizi
oluşturulur

X_train,X_test,Y_train,Y_test =
train_test_split(x,y,test_size=0.35,random_state=109) #%35 u test %65 i
eğitim


for i in k_size:
  #Svm classifier
  clf = SVC(kernel='linear', C=c ) #Veriler doğrusal şekilde ayrılacağı
için kernel lineer alınmıştır.
  clf.fit(X_train,Y_train) # sınıfın model üzerinde eğitilmesi
  y_pred = clf.predict(X_test) #Model test verisi üzerinde tahmin yapar
  accuracy[i-1] = metrics.accuracy_score(Y_test, y_pred) # Doğruluk skoru
hesaplanması
  print(f"{i}.iteration C value: {c}, Accuracy Score: {accuracy[i-1]}")
  c /= 2 # C değerini azaltma


best_accuracy  = 0 #en iyi başarı oranının tutulacağı değer
best_c_value = 0 #başarı oranının en iyi olduğu yerdeki C değerinin
tutulacağı değer

best_accuracy = accuracy.max()
best_c_value = accuracy.argmax()+1
print(f"Best accuracy : {best_accuracy} , Best C value  : {best_c_value}")

#C değeri azaldığı süre boyunca başarı değerinin de genelde azaldığı
gözlemlenmiştir.
```

Yukarıdaki kodda hesaplanan Doğruluk değerlerinin grafikte gösterimi şağıdaki gibidir.



(S değeri iterasyon değeridir)