KARABÜK ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSİTÜSÜ

YÜKSEK LİSANS BİYOMEDİKAL MÜHENDİSLİĞİ BÖLÜMÜ

DERSİN ADI:BMM723 Biyomedikal Mühendisliğinde Yapay Sinir Ağı Uygulamaları

Proje Adı: Regression ve Classification Uygulamaları

Öğrencinin Adı:TUĞBA TAŞBAŞI

Öğretim Elemanı: Dr.Öğr.Üyesi HAKAN YILMAZ

1) REGRESSİON

1.1 SEÇİLEN VERİ SETİ

Predicting Grades for the School Year (Math Subject)

Öğretim yılı için notları tahmin etme(Matematik Dersi)

Veri Seti Bilgileri

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
school 395 non-null object
sex 395 non-null object
age 395 non-null int64
address 395 non-null object
famsize 395 non-null object
Pstatus 395 non-null object
Medu 395 non-null int64
Fedu 395 non-null int64
Mjob 395 non-null object
Fjob 395 non-null object
reason 395 non-null object
guardian 395 non-null object
traveltime 395 non-null int64
studytime 395 non-null int64
failures 395 non-null int64
schoolsup 395 non-null object
famsup 395 non-null object
paid 395 non-null object
activities 395 non-null object
nursery 395 non-null object
higher 395 non-null object
internet 395 non-null object
romantic 395 non-null object
famrel 395 non-null int64
freetime 395 non-null int64
goout 395 non-null int64
Dalc 395 non-null int64
Walc 395 non-null int64
health 395 non-null int64
absences 395 non-null int64
G1 395 non-null int64
G2 395 non-null int64
G3 395 non-null int64
dtypes: int64(16), object(17)
memory usage: 102.0+ KB
None

1.2 RAPOR

Projede, bir öğrencinin yıl sonunda alacağı puanını tahmin eden linear regression,decision tree,random forest algoritmalarına r2 score,MSE,MAE değerlendirme skorları uygulanıldı. "G3" veya son sınıf etiketimiz (çıktı) olacak ve sütunların geri kalanı özelliklerimiz (girişler) olacaktır.

1.3 VERİ SETİ HAZIRLIK AŞAMASI

Literatur Taraması

Veri seti seçiminde genellikle sayısal girdileri ve bir çıktısı olan veri setleri araştırıldı. Çünkü **Linear regression** (Doğrusal regresyon), temel ve yaygın olarak kullanılan bir tahmin analizidir. Sayısal girdi ve çıktı değerleri (Kategorik verilerde kullanılamaz) arasında ilişki kurmayı sağlar yani kısaca amacımız karmaşık biçimde bulunan gerçek değerleri tek bir doğru şeklinde göstermektir.

Decision Tree insan düşünce yapısına en uygun olan yapılardan biridir. Karar ağacı öğrenmesinde, ağacın öğrenilmesi sırasında, üzerinde eğitim yapılan küme, çeşitli özelliklere göre alt kümelere bölünür, bu işlem, özyineli olarak (recursive) tekrarlanır ve tekrarlama işleminin tahmin üzerinde bir etkisi kalmayana kadar sürer. Bu işleme özyineli parçalama (recursive partitioning) ismi verilir.
Genelde veri madenciliği sırasında verinin gelme şekli aşağıdaki gibidir:
(x,Y) = (x1,x2,x3,x4,…, Y)
Bu gösterime göre x1'den xn'e kadar olan değerler, sistemin girdileri iken, Y değeri sistemin çıktısı olarak elde edilmesi istenen değerdir. Veri madenciliğinde karar ağacı öğrenmesi (decision tree learning) iki temel amaç için kullanılır. Bu amaçlar ve karar ağacı öğrenmesinin bu amaca yönelik özel isimleri aşağıdaki şekildedir:

- Sınıflandırma problemleri : Sınıflandırma ağaçları **(Classification Tree)** : Bir kişinin harcamalarından eğitim düzeyinin tahmini gibi, hedef kümeyi çeşitli sınıflardan birisine yerleştirmeyi amaçlayan ve sınıf tanımı yapan problemler.
- İlkelleme problemleri : İlkelleme ağaçları **(Regression Trees):** Sonuçta bir sınıf yerine sayısal bir değer döndüren veri madenciliği problemleri .Decision Tree algoritmasını kullanmak daha kullanışlıdır.Çünkü basit olduğu için çok veri kullanıldığında bu veriler özellikle sayılar girdi ve çıktıysa işlemi daha kısa sürede yapacaktır.

Random Forest karar ağaçlarının en büyük problemlerinden biri aşırı öğrenme-veriyi ezberlemedir (overfitting). Rassal orman modeli bu problemi çözmek için hem veri setinden hem de öznitelik setinden rassal olarak 10'larca 100'lerce farklı alt setler seçiyor ve bunları eğitiyor. Bu yöntemle 100'lerce karar ağacı oluşturuluyor ve her bir karar ağacı bireysel olarak tahminde bulunuyor. Günün sonunda problemimiz regresyonsa karar ağaçlarının tahminlerinin ortalamasını problemimiz sınıflandırmaysa tahminler arasında en çok oy alanı seçiyoruz.
Bütün bunları basit bir örnekle açıklamaya çalışalım: Örneğin bu akşam güzel bir film izlemek istiyorsunuz ve kafanız karışık. Bir arkadaşınızı ararsanız ve o size tercih ettiğiniz film türü, süre, yıl, oyuncu-yönetmen, hollywood-alternatif vs. soru setinden çeşitli sorularla daha önce izlediğiniz filmlere (training set) göre bir tahminde bulunursa bu karar ağacı olur. Eğer 20 arkadaşınız bu soru setinden farklı sorular seçip verdiğiniz cevaplara göre tavsiyede bulunursa ve siz en çok tavsiye edilen filmi seçerseniz bu rassal orman olur.

Rassal orman modelinde farklı veri setleri üzerinde eğitim gerçekleştiği için varyans, diğer bir deyişle karar ağaçlarının en büyük problemlerinden olan overfitting azalır. Ayrıca bootstrap yöntemiyle oluşturduğumuz alt-veri kümelerinde outlier bulunma şansını da düşürmüş oluruz.
Random forest modelinin diğer bir özelliği bize özniteliklerin ne kadar önemli olduğunu vermesi. (Bir özniteliğin önemli olması demek o özniteliğin bağımlı değişkendeki varyansın açıklanmasına ne kadar katkı yaptığıyla alakalı.) Random forest algoritmasına x sayıda öznitelik verip en faydalı y tanesini seçmesini isteyebiliriz ve istersek bu bilgiyi istediğimiz başka bir modelde kullanabiliriz.

Regression Değerlendirme Yöntemleri

a)R2 Score

Residual=v-v_{head}

Square residual=(residual)^2

Sum square residual=sum((y-y_head)^2^{SSR}

Sum square total=sum((y-y_avg)^2)^{SST}

R^2=1-(SSR/SST)

R^2 =1'e ne kadar yakın ise doğruluk o kadar iyi anlamına gelir.SST ve SSR değerleri birbirine yakın olması da r^2'i 1'e yaklaştırır.

b) [MSE] Ortalama Kare Hata (Mean Squared Error)

Note 1. Legend: A_j – actual values; $\bar{A}$ – the mean of the actual values; P_j – predicted values; $e_j = A_j - P_j$ – error; n – size of the data set

Error (magnitude of error): $\mathbb{D}1 = A_j - P_j = e_j$

Absolute error: $\mathbb{D}2 = |A_j - P_j| = |e_j|$

Squared error: $\mathbb{D}3 = (A_j - P_j)^2 = e_j^2$

$$MSE = \frac{1}{n}\sum_{j=1}^{n} e_j^2$$

Ortalama kare hata bir regresyon eğrisinin bir dizi noktaya ne kadar yakın olduğunu söyler. MSE, bir makine öğrenmesi modelinin, tahminleyicinin performansını ölçer, her zaman pozitif değerlidir ve MSE değeri sıfıra yakın olan tahminleyicilerin daha iyi bir performans gösterdiği söylenebilir.

c)MAE] Ortalama Mutlak Hata (Mean Absolute Error)

$$MAE = \frac{1}{n}\sum_{j=1}^{n} |e_j|$$

Ortalama mutlak hata iki sürekli değişken arasındaki farkın ölçüsüdür. MAE, her gerçek değer ile veriye en iyi uyan çizgi arasındaki ortalama dikey mesafedir. MAE aynı zamanda her veri noktası ile en iyi uyan çizgi arasındaki ortalama yatay mesafedir. MAE değeri kolay

yorumlanabilir olduğu için regresyon ve zaman serisi problemlerinde sıkça kullanılmaktadır. MAE, yönlerini dikkate almadan bir dizi tahmindeki hataların ortalama büyüklüğünü ölçen, tüm tekil hataların ortalamada eşit olarak ağırlıklandırıldığı doğrusal bir skordur. MAE değeri 0'dan ∞'a kadar değişebilir. Negatif yönelimli puanlar yani daha düşük değerlere sahip tahminleyiciler daha iyi performans gösterir.

PHYTON ALGORİTMA

#Gerekli Kütüphaneler import edildi
#DataFrame oluşturuldu csv dosyası okundu
#Veri setimizin bilgileri görüntülendi
#G3 drop edildi(uçuruldu) x adında bir DataFrame oluşturuldu
#x DataFrame dummy variable(0-1) olarak dönüştürülüp axis=1'de concat ile birleştirildi
#object olan veriler numerical seçildi
kopyaları silindi
#y-NumPy array oluşturuldu G3 sütunu içerikli
#%% normalization sonuçları yükseltmek için denendi normalizer uygulandı
#%% Linear Regression
#LinearRegression sınıfı import edildi
#mean_squared_error sınıfı import edildi
#mean_absolute_error sınıfı import edildi
#r2_score sınıfı import edildi
#X,y fit edildi
#predict
r2 score linear doğruluk hesaplandı
#ortalama karasel hata hesaplandı
#ortalama mutlak hata hesaplandı
#hesaplanan değerleri görüntüle
#%% RanfomForest Regression
#train_test_split sınıfı import edildi
RandomForestRegressor sınıfı import edildi
Veri kümemizi test ve train şekinde bölüyoruz
test_size = 0.35 Veri
x_train,y_train = X,y'nin eğitim kümesi
x_test, y_test = X,y'nin test kümesi
test_size = veri kümesinin %35 i eğitim %65'i test için ayrılacak .
Random Forest algoritmasını kullanarak eğitilen makinenin egitim değerlerine göre bir tahmin yapılması sağlandı
#r2 score RandomForestRegressor'a göre hesaplandı
#ortalama karasel hata RandomForestRegressor'a göre hesaplandı
#ortalama mutlak hata RandomForestRegressor'a göre hesaplandı
#hesaplanan değerleri görüntüle

1.4 NORMALİZASYON ÖNCESİ SONUÇLAR

1.4.1Linear Regression Sonuçlar

Linear Accuracy: 0.8457661446216241
Linear MSE: 3.2291137398043763
Linear MAE: 1.1846834206882912

1.4.2 Desicion Tree Sonuçlar

```
DecisionTree Accuracy:  1.0
DecisionTree MSE:  0.0
DecisionTree MAE:  0.0
```

1.4.3 Random Forest Sonuçlar

```
RanfomForest Accuracy:  0.8431471592192503
RanfomForest MSE:  3.665928776978417
RanfomForest MAE:  1.1642446043165469
```

Değerlendirme

Desicion Tree istenilen bir sonucu vermiştir %100 doğruluk hata oranları ise 0 dır.Hata oranlarının 0'a yakın olması (yani MSE,MAE'nin) tercih edilir.Ne kadar yakınsa istenilene o kadar ulaşılmıştır.Accury yani başarı oranı ise 1'e yaklaşmalıdır. Linear regression random foresttan daha iyi başarı oranı vermiştir Linear MSE RandomForest MSE'den daha daha düşüktür.Fakat RandomForest MAE Linear MAE den daha düşüktür.

1.5 NORMALİZASYON SONRASI SONUÇLAR

1.5.1 Linear Regression

```
Linear Accuracy:  0.8498970503208036
Linear MSE:  3.142627122982587
Linear MAE:  1.218890755813371
```

1.5.2 Deicion Tree

```
DecisionTree Accuracy:  1.0
DecisionTree MSE:  0.0
DecisionTree MAE:  0.0
```

1.5.3 Random Forest

```
RanfomForest Accuracy:  0.8552356829344991
RanfomForest MSE:  3.3833985611510786
RanfomForest MAE:  1.2218705035971225
```

Değerlendirme

Doğruluk oranı hepsinde de artış gösterdi. Linear MSE 3.22 den 3.14 e düştü.Linear MAE 1.18 den 1.21 e arttı.Random forest MSE 3.66 dan 3.38 e düştü.Random forest MAE 1.16 dan 1.22 ye çıktı. Normalizasyon doğruluk oranında ve MSE'lerde istenildiği gibi iyileştirme yaptı. Fakat MAE oranlarını artırdı.

2)CLASSİFİCATİON

2.1) SEÇİLEN VERİ SETİ

Diabetes

Veri seti birkaç tıbbi öngörücü (bağımsız) değişken ve bir hedef (bağımlı) değişkenden oluşur. Bağımsız değişkenler, hastanın sahip olduğu gebeliklerin sayısını, BMI'larını, insülin seviyesini, yaşını vb. İçerir. Teşhis önlemlerine dayanarak diyabetin başlangıcını tahmin etmeye çalışıldı.

Veri Seti Bilgileri

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies 768 non-null int64
Glucose 768 non-null int64
BloodPressure 768 non-null int64
SkinThickness 768 non-null int64
Insulin 768 non-null int64
BMI 768 non-null float64
DiabetesPedigreeFunction 768 non-null float64
Age 768 non-null int64
Case 768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None

2.2) Rapor

Bu veri seti aslen Ulusal Diyabet ve Sindirim ve Böbrek Hastalıkları Enstitüsü'nden alınmıştır. Veri kümesinin amacı, veri kümesine dahil edilen belirli tanı ölçümlerine dayanarak bir hastanın diyabet olup olmadığını teşhis amaçlı olarak tahmin etmektir. Bu örneklerin seçiminde daha büyük bir veritabanından çeşitli kısıtlamalar getirildi. Classification için knn, svm, random forest çalışılıp r2 score(accury) ve f1 skoru hesaplandı.

2.3) VERİ SETİ HAZIRLIK AŞAMASI

Literatur Taraması

KNN (K nearest neighborhood, en yakın k komşu)

Sınıflandırmada (classification) kullanılan bu algoritmaya göre sınıflandırma sırasında çıkarılan özelliklerden (feature extraction), sınıflandırılmak istenen yeni bireyin daha önceki bireylerden k tanesine yakınlığına bakılmasıdır.

SVM (Support Vector Machine, Destekçi Vektör Makinesi)

Sınıflandırma (Classification) konusunda kullanılan oldukça etkili ve basit yöntemlerden birisidir. Sınıflandırma için bir düzlemde bulunan iki grup arasında bir sınır çizilerek iki grubu ayırmak mümkündür. Bu sınırın çizileceği yer ise iki grubun da üyelerine en uzak olan yer olmalıdır. İşte SVM bu sınırın nasıl çizileceğini belirler.

Bu işlemin yapılması için iki gruba da yakın ve birbirine paralel iki sınır çizgisi çizilir ve bu sınır çizgileri birbirine yaklaştırılarak ortak sınır çizgisi üretilir.

Classification Değerlendirme Yöntemleri

F1 skoru

sklearn.metrics.f1_score

Dengeli F-puanı veya F-ölçüsü olarak da bilinen F1 puanını hesaplama

F1 skoru, hassasiyet ve hatırlamanın ağırlıklı ortalaması olarak yorumlanabilir, burada F1 skoru 1'de en iyi değerine ve en kötü skoru 0'da olur. Hassasiyet ve hatırlamanın F1 skoruna göreceli katkısı eşittir. F1 puanı için formül:

F1 = 2 * (precision * recall) / (precision + recall)

Çok sınıflı ve çok etiketli durumda, bu, average parametreye bağlı olarak her sınıfın F1 puanının ortalamasıdır.

Hamming Kayıp Metriği(Hamming Loss Metric)

from sklearn.metrics import hamming_loss

Hamming Loss, doğru sınıflandırılmış veri örneğini saymak yerine, tahmin sırasında sınıf etiketlerinin bit dizesinde üretilen kaybı hesaplar. Veri örneği için sınıf etiketlerinin orijinal ikili dizesi ile öngörülen sınıf etiketleri arasında XOR işlemi yapar ve veri kümesindeki ortalamayı hesaplar. İfadesi

$$hamming\,loss = \frac{1}{|N|.|L|}\sum_{i=1}^{|N|}\sum_{j=1}^{|L|} XOR(y_{i,j},\, \hat{y_{i,j}})$$

| N | = veri örneği sayısı

| L | = sınıf boşluğunun önemliliği

$y_{i,j}$ = veri örneği i'de j sınıf etiketi gerçek biti

$\hat{y}_{i,j}$ = veri örneği i'de j sınıf etiketi j'nin tahmini biti

'Hamming loss' değeri 0 ile 1 arasında değişmektedir. Bir kayıp ölçütü olduğundan yorumu normal doğruluk oranından farklı olarak tersidir. Darbe kaybının daha düşük değeri daha iyi bir sınıflandırıcıyı gösterir.

Phyton Algoritma

#Gerekli Kütüphaneler import edildi

#DataFrame oluşturuldu csv dosyası okundu

#Outcome sütun adı case olarak değiştirildi

#DataFrame bilgileri görüntüle

#y-NumPy array oluşturuldu case sütunu için

#X adında DataFrame oluşturuldu Case drop edilip

#x=(X-np.min(X))/(np.max(X)-np.min(X)) (Min-Max normalizasyon yapıldı)

#train_test_split sınıfı import edildi

Veri kümemizi test ve train şekinde bölüyoruz

test_size = 0.3 Veri random_state=1

#%% KNN

#KNeighborsClassifier import edildi

#score hesaplandı(accury,f1)

#skor sonuçları gösterildi

#hyper parametre en iyi k değerini bulmak için bir for döngüsü oluşturuldu doğruluğun en üst noktasının n_neighbors'un en iyi değerinin neolduğu görüldü.

#%% SVM

SVC import edildi.

#gama değer 'auto' seçildi.En iyi değeri bulup kendisi bulsun diye.

#skorlar hesaplandı(accury,f1)

#skor sonuçları gösterildi

#%% DT Classification

#DecisionTreeClassifier import edildi.

#score hesaplandı(accury,f1)

#skor sonuçları gösterildi

#%% RF Classification

#RandomForestClassifier import edildi.

#skorlar hesaplandı(accury,f1)

#skor sonuçları gösterildi

2.4 NORMALİZASYON ÖNCESİ SONUÇLARI(MAX-MİN)

2.4.1) KNN (K nearest neighborhood, en yakın k komşu)

knn accury: 0.7857142857142857
knn f1 score: 0.6373626373626373
knn hamming loss: 0.21428571428571427

2.4.2) SVM (Support Vector Machine, Destekçi Vektör Makinesi)

svm accuracy: 0.6428571428571429
svm f1 score: 0.782608695652174
svm hamming loss: 0.35714285714285715

2.4.3) DT(Desicion Tree)

dt accuracy: 0.6818181818181818
dt f1 score: 0.5504587155963303
dt hamming loss: 0.3181818181818182

2.4.4) RT(Random Forest Sonuçlar)

```
rf accuracy:  0.8051948051948052
rf f1 score:  0.7058823529411765
rf hamming loss:  0.19480519480519481
```

2.5) NORMALİZASYON SONRASI SONUÇLARI(MAX-MİN)

2.5.1) KNN (K nearest neighborhood, en yakın k komşu)

```
knn accury:  0.8116883116883117
knn f1 score:  0.7070707070707071
knn hamming loss:  0.18831168831168832
```

2.5.2) SVM (Support Vector Machine, Destekçi Vektör Makinesi)

```
svm accuracy:  0.7727272727272727
svm f1 score:  0.7545118843293285
svm hamming loss:  0.22727272727272727
```

2.5.3) DT(Desicion Tree)

```
dt accuracy:  0.7012987012987013
dt f1 score:  0.5576923076923076
dt hamming loss:  0.2987012987012987
```

2.5.4) RT(Random Forest Sonuçlar)

```
rf accuracy:  0.8116883116883117
rf f1 score:  0.7184466019417476
rf hamming loss:  0.18831168831168832
```

Değerlendirme

test_size=0.2 seçildiğinde en iyi sonuçların alındığı görüldü.
Normalizasyon öncesinde n_neighbors=16 seçildi.Daha iyi sonuçlar alındı.Normalizasyon sonrasında ise 7 seçildi ve sonuçlar yükseldi.

Normalizasyon öncesinde RF accury en yüksek değeri almıştır. F1 score 1'e en yakın değer SVM de alınmıştır. Hamming loss 0'a en yakın olan ise DT'dir.

Normalizasyon sonrasında RF accury ve KNN accury en yüksek değeri almıştır. F1 score 1'e en yakın değer SVM de alınmıştır. Hamming loss 0'a en yakın olan ise RF ve KNN'dir. Normalizasyon sonrasında genel olarak değerler artmıştır.

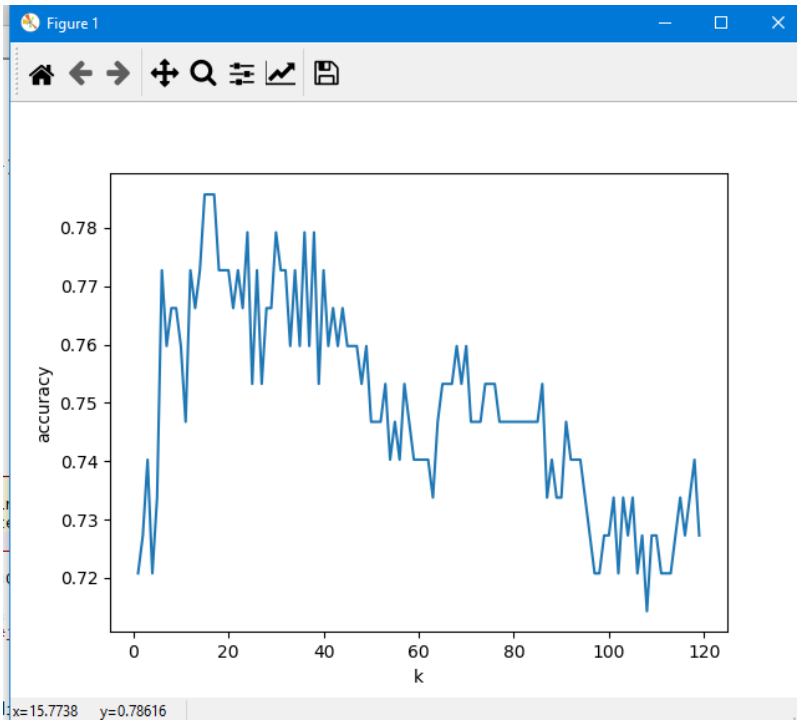
Figure 1
0.78
0.77
0.76
0.75
0.74
0.73
0.72
accuracy
0
20
40
60
80
100
120
k
x=15.7738 y=0.78616

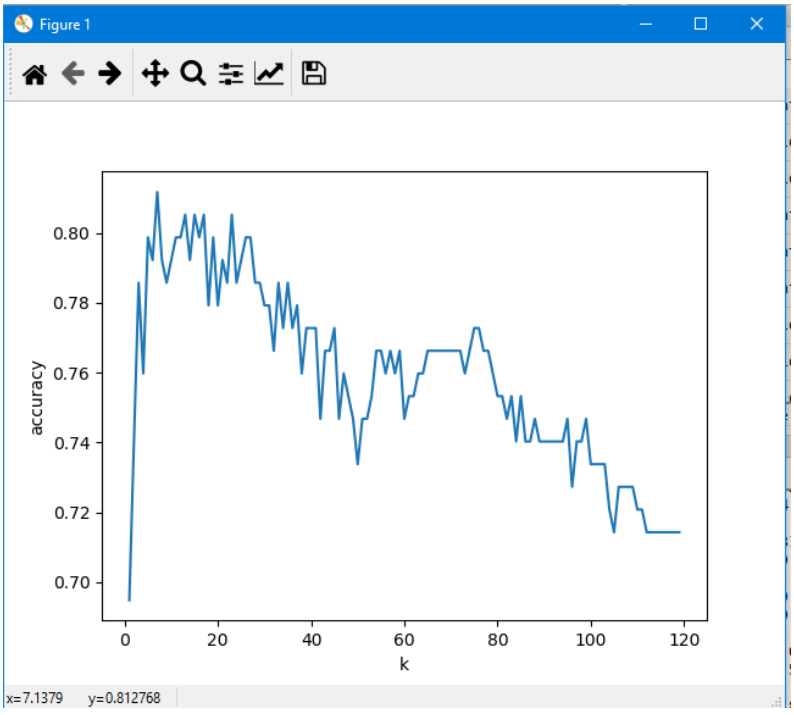
Figure 1
0.80
0.78
0.76
0.74
0.72
0.70
accuracy
0
20
40
60
80
100
120
k
x=7.1379 y=0.812768

KAYNAKÇA

- https://www.kaggle.com/janiobachmann/predicting-grades-for-the-school-year/data
- https://www.kaggle.com/saurabh00007/diabetescsv
- https://www.kaggle.com/archaeocharlie/a-beginner-s-approach-to-classification
- http://bilgisayarkavramlari.sadievrenseker.com/2013/03/31/siniflandirma-classification/
- https://www.veribilimiokulu.com/support-vector-machine-svm-ile-siniflandirma-python-ornek-uygulamasi/
- https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics
- Weisberg, S. (2005). *Applied linear regression* (Vol. 528). John Wiley & Sons.
- https://en.wikipedia.org/wiki/Linear_regression
- https://en.wikipedia.org/wiki/Decision_tree
- https://en.wikipedia.org/wiki/Random_forest
- https://www.kaggle.com/caneremec/s-n-fland-r-c-lar-n-kar-la-t-r-lmas
- https://www.researchgate.net/publication/322010741_Comparison_of_Random_Forest_k-Nearest_Neighbor_and_Support_Vector_Machine_Classifiers_for_Land_Cover_Classification_Using_Sentinel-2_Imagery