YAPAY SİNİR AĞLARI DERSİ VİZE UYGULAMA SINAVI DENEY GÖZLEM RAPORU

YSA İLE KALP YETMEZLİĞİ TAHMİNİ

ENES YÜKSEL
4. SINIF / 194410044
Bilgisayar Mühendisliği

Kalp Yetmezliği Tahmini için "heart_failure_clinical_records_dataset" veri seti kullanılır ve kalp yetmezliğine bağlı ölümleri tahmin etmek için kullanılabilecek 12 özellik içerir: age,anaemia,creatinine_phosphokinase,diabetes,ejection_fraction,high_blood_pressure,platelets, serum_creatinine,serum_sodium,sex,smoking,time(Şekil 1)

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4	1
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6	1
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7	1
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7	1
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8	1
5	90.0	1	47	0	40	1	204000.00	2.1	132	1	1	8	1
6	75.0	1	246	0	15	0	127000.00	1.2	137	1	0	10	1
7	60.0	1	315	1	60	0	454000.00	1.1	131	1	1	10	1
8	65.0	0	157	0	65	0	263358.03	1.5	138	0	0	10	1
9	80.0	1	123	0	35	1	388000.00	9.4	133	1	1	10	1

Şekil 1 *Tahmin için Nitelikli 12 Özellik*

Gerekli kütüphaneler tanımlanır, özellikler matrisi oluşturulur, bağımlı değişken vektörü belirlenir, veri seti bölünür, Standardizasyon kullanılır ve YSA başlatılır. İki gizli katman oluşturulur ve sigmoid aktivasyon fonksiyonu tanımlanır.

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=100)
```

“optimizer” parametresi, w değerlerinin iyileştirilmesi için kullanılan optimizasyon algoritmalarının kullanılmasını sağlamaktadır. “adam” (Adaptive Moment Estimation) algoritması, her bir parametre için gerçek zamanlı olarak öğrenme oranını günceller.

Mini batch size(boyutu), parametre güncellemesinin gerçekleştiği ağa verilen alt örneklerin sayısıdır. Epoch(döngü) sayısı, eğitim sırasında tüm eğitim verilerinin ağa gösterilme sayısıdır.

“batch_size” ve “epochs” değerlerinde değişikler yapılır

Adım 11: Deney sonuçları için kritik öneme sahip "optimizer", "loss", "metrics" girdileri ayarlanır ve YSA derlenir

```
ann.compile(optimizer="adam",loss="binary_crossentropy",metrics=['accuracy'])
```

Optimizer parametresinin “adam” olarak alındığı senaryo için;

İlk 3 deney için Epoch(döngü) sayısı 100 ile sabit tutulmuştur ve deneylere başlanır:

DENEY 1

```
In [75]: ann.fit(X_train,y_train,batch_size=128,epochs=100)
         Epoch 92/100
         2/2 [==============================] - 0s 3ms/step - loss: 0.5338 - accuracy: 0.7703
         Epoch 93/100
         2/2 [==============================] - 0s 3ms/step - loss: 0.5313 - accuracy: 0.7656
         Epoch 94/100
         2/2 [==============================] - 0s 2ms/step - loss: 0.5285 - accuracy: 0.7656
         Epoch 95/100
         2/2 [==============================] - 0s 3ms/step - loss: 0.5259 - accuracy: 0.7656
         Epoch 96/100
         2/2 [==============================] - 0s 2ms/step - loss: 0.5233 - accuracy: 0.7656
         Epoch 97/100
         2/2 [==============================] - 0s 3ms/step - loss: 0.5206 - accuracy: 0.7656
         Epoch 98/100
         2/2 [==============================] - 0s 2ms/step - loss: 0.5179 - accuracy: 0.7656
         Epoch 99/100
         2/2 [==============================] - 0s 2ms/step - loss: 0.5153 - accuracy: 0.7656
         Epoch 100/100
         2/2 [==============================] - 0s 3ms/step - loss: 0.5124 - accuracy: 0.7703

Out[75]: <keras.callbacks.History at 0x1e6988a3190>
```

Senaryo 2 *batch_size=128,epochs=100*

Yorum 1: Mini batch size(boyutu) için varsayılan olarak nitelendirilen 32 değeri yerine ilk olarak 128 değeri atanmıştır. Bunun sonucunda 51 loss, 77 accuracy değeri elde edilmiştir. Eğitim sonucu oluşan değerler, kötü sayılabilecek kadar düşüktür.

DENEY 2

```
In [136]: ann.fit(X_train,y_train,batch_size=64,epochs=100)
          Epoch 92/100
          4/4 [==============================] - 0s 1ms/step - loss: 0.4018 - accuracy: 0.8182
          Epoch 93/100
          4/4 [==============================] - 0s 2ms/step - loss: 0.4005 - accuracy: 0.8182
          Epoch 94/100
          4/4 [==============================] - 0s 1ms/step - loss: 0.3994 - accuracy: 0.8134
          Epoch 95/100
          4/4 [==============================] - 0s 2ms/step - loss: 0.3981 - accuracy: 0.8086
          Epoch 96/100
          4/4 [==============================] - 0s 2ms/step - loss: 0.3968 - accuracy: 0.8134
          Epoch 97/100
          4/4 [==============================] - 0s 2ms/step - loss: 0.3956 - accuracy: 0.8134
          Epoch 98/100
          4/4 [==============================] - 0s 1ms/step - loss: 0.3945 - accuracy: 0.8134
          Epoch 99/100
          4/4 [==============================] - 0s 2ms/step - loss: 0.3931 - accuracy: 0.8134
          Epoch 100/100
          4/4 [==============================] - 0s 2ms/step - loss: 0.3919 - accuracy: 0.8182

Out[136]: <keras.callbacks.History at 0x1e69ace40d0>
```

Senaryo 2 *batch_size=64,epochs=100*

Yorum 2: Mini batch size(boyutu) için varsayılan olarak nitelendirilen 32 değeri yerine 64 değeri atanmıştır. Bunun sonucunda 39 loss, 81 accuracy değeri elde edilmiştir. Eğitim sonucu oluşan değerler, ortanın altında sayılabilecek seviyededir.

DENEY 3

```
In [151]: ann.fit(X_train,y_train,batch_size=32,epochs=100)
          Epoch 92/100
          7/7 [==============================] - 0s 1ms/step - loss: 0.3554 - accuracy: 0.8517
          Epoch 93/100
          7/7 [==============================] - 0s 1ms/step - loss: 0.3546 - accuracy: 0.8517
          Epoch 94/100
          7/7 [==============================] - 0s 1ms/step - loss: 0.3535 - accuracy: 0.8469
          Epoch 95/100
          7/7 [==============================] - 0s 2ms/step - loss: 0.3529 - accuracy: 0.8469
          Epoch 96/100
          7/7 [==============================] - 0s 1ms/step - loss: 0.3519 - accuracy: 0.8469
          Epoch 97/100
          7/7 [==============================] - 0s 1ms/step - loss: 0.3510 - accuracy: 0.8469
          Epoch 98/100
          7/7 [==============================] - 0s 1ms/step - loss: 0.3502 - accuracy: 0.8469
          Epoch 99/100
          7/7 [==============================] - 0s 1ms/step - loss: 0.3496 - accuracy: 0.8469
          Epoch 100/100
          7/7 [==============================] - 0s 1ms/step - loss: 0.3488 - accuracy: 0.8469
Out[151]: <keras.callbacks.History at 0x1e6977b7820>
```

Senaryo 3 *batch_size=32,epochs=100*

Yorum 3: Mini batch size(boyutu) için varsayılan olarak nitelendirilen 32 değeri atanmıştır. Bunun sonucunda 34 loss, 84 accuracy değeri elde edilmiştir. Eğitim sonucu oluşan değerler, orta sayılabilecek seviyededir.

Çıkarım: Epoch sayısı sabit tutulup, batch_size için varsayılan olarak nitelendirilen 32 değeri kullanıldığında accuracy değeri artıp loss değeri azalmaktadır. Bu durum, tahminlerdeki verimliliği artırmaktadır.

Sonraki 3 deney için batch_size değeri 32 ile sabit tutulmuştur ve deneylere başlanır:

DENEY 4

```
In [226]: ann.fit(X_train,y_train,batch_size=32,epochs=500)
          7/7 [==============================] - 0s 1ms/step - loss: 0.2306 - accuracy: 0.9234
          Epoch 493/500
          7/7 [==============================] - 0s 1ms/step - loss: 0.2305 - accuracy: 0.9234
          Epoch 494/500
          7/7 [==============================] - 0s 1ms/step - loss: 0.2304 - accuracy: 0.9234
          Epoch 495/500
          7/7 [==============================] - 0s 1ms/step - loss: 0.2302 - accuracy: 0.9234
          Epoch 496/500
          7/7 [==============================] - 0s 1ms/step - loss: 0.2301 - accuracy: 0.9234
          Epoch 497/500
          7/7 [==============================] - 0s 1ms/step - loss: 0.2300 - accuracy: 0.9234
          Epoch 498/500
          7/7 [==============================] - 0s 1ms/step - loss: 0.2296 - accuracy: 0.9234
          Epoch 499/500
          7/7 [==============================] - 0s 1ms/step - loss: 0.2295 - accuracy: 0.9234
          Epoch 500/500
          7/7 [==============================] - 0s 1ms/step - loss: 0.2293 - accuracy: 0.9234
Out[226]: <keras.callbacks.History at 0x1e69e23daf0>
```

Senaryo 4 *batch_size=32,epochs=500*

Yorum 4: Epoch sayısı için ilk değer olan 100 yerine 500 değeri alınmıştır. Bunun sonucunda 22 loss, 92 accuracy değeri elde edilmiştir. Eğitim sonucu oluşan değerler, iyi düzeydedir.

DENEY 5

```
In [241]: ann.fit(X_train,y_train,batch_size=32,epochs=1000)
          Epoch 992/1000
          7/7 [==============================] - 0s 1ms/step - loss: 0.0691 - accuracy: 0.9761
          Epoch 993/1000
          7/7 [==============================] - 0s 1ms/step - loss: 0.0688 - accuracy: 0.9761
          Epoch 994/1000
          7/7 [==============================] - 0s 1ms/step - loss: 0.0683 - accuracy: 0.9761
          Epoch 995/1000
          7/7 [==============================] - 0s 1ms/step - loss: 0.0679 - accuracy: 0.9761
          Epoch 996/1000
          7/7 [==============================] - 0s 1ms/step - loss: 0.0675 - accuracy: 0.9713
          Epoch 997/1000
          7/7 [==============================] - 0s 1000us/step - loss: 0.0672 - accuracy: 0.9761
          Epoch 998/1000
          7/7 [==============================] - 0s 1ms/step - loss: 0.0668 - accuracy: 0.9761
          Epoch 999/1000
          7/7 [==============================] - 0s 2ms/step - loss: 0.0668 - accuracy: 0.9761
          Epoch 1000/1000
          7/7 [==============================] - 0s 1ms/step - loss: 0.0662 - accuracy: 0.9761
Out[241]: <keras.callbacks.History at 0x1e69f31e970>
```

Senaryo 5 *batch_size=32,epochs=1000*

Yorum 5: Epoch sayısı için 500 yerine 1000 değeri alınmıştır. Bunun sonucunda 06 loss, 97 accuracy değeri elde edilmiştir. Eğitim sonucu oluşan değerler, mükemmel düzeydedir.

RMSprop, karelerin ortalamasının karekökü yayılımı(Root Mean Square Propagation) nın ana fikri, gradyanın karelerin ortalamasının karekökü ile normalize edilmesidir.

DENEY 6

Adım 11: Deney sonuçları için kritik öneme sahip "optimizer", "loss", "metrics" girdileri ayarlanır ve YSA derlenir

```
#ann.compile(optimizer="adam",loss="binary_crossentropy",metrics=['accuracy'])
```

#Adım 12: Alternatif "rmsprop" optimizer değeri ayarlanır ve YSA tekrar derlenir

```
ann.compile(optimizer="rmsprop",loss="binary_crossentropy",metrics=['accuracy'])
```

Optimizer parametresinin "rmsprop" olarak alındığı senaryo için;

```
ann.fit(X_train,y_train,batch_size=32,epochs=1000)
Epoch 992/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1029 - accuracy: 0.9617
Epoch 993/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1021 - accuracy: 0.9617
Epoch 994/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1023 - accuracy: 0.9617
Epoch 995/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1022 - accuracy: 0.9617
Epoch 996/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1027 - accuracy: 0.9617
Epoch 997/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1019 - accuracy: 0.9617
Epoch 998/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1020 - accuracy: 0.9617
Epoch 999/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1015 - accuracy: 0.9617
Epoch 1000/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1016 - accuracy: 0.9617

<keras.callbacks.History at 0x1e6a0555f70>
```

Senaryo 6 *optimizer="rmsprop"*

Yorum 6: Optimizer parametresi için “adam” yerine "rmsprop" kullanılmıştır. Bunun sonucunda 10 loss, 96 accuracy değeri elde edilmiştir. Eğitim sonucu oluşan değerler, yine mükemmele yakın düzeydedir.

Çıkarım: Son deneyde optimizer parametresi için, epoch sayısı ve batch_size değerleri sabit tutulmuştur. “adam” yerine "rmsprop" kullanılmıştır. Bunun sonucunda “adam”a göre loss değeri artıp, “accuracy” değeri azalmaktadır.

Sonraki deney için optimizer parametresi için verimli olan “adam” ile sabit tutulmuştur ve deneylere devam edilir:

DENEY 7

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=100)
```

Veri setimizi, eğitim aşamasında verinin yüzde 70'i yerine yüzde 80’i oranında olacak ve verinin yüzde 30'u yerine yüzde 20’si test aşamasında olacak şekilde eğitim ve test veri setlerine tekrar ayırırız:

```
ann.fit(X_train,y_train,batch_size=32,epochs=1000)
Epoch 992/1000
8/8 [==============================] - 0s 1ms/step - loss: 0.0801 - accuracy: 0.9791
Epoch 993/1000
8/8 [==============================] - 0s 1ms/step - loss: 0.0803 - accuracy: 0.9791
Epoch 994/1000
8/8 [==============================] - 0s 1000us/step - loss: 0.0801 - accuracy: 0.9791
Epoch 995/1000
8/8 [==============================] - 0s 1ms/step - loss: 0.0798 - accuracy: 0.9791
Epoch 996/1000
8/8 [==============================] - 0s 1ms/step - loss: 0.0798 - accuracy: 0.9791
Epoch 997/1000
8/8 [==============================] - 0s 1ms/step - loss: 0.0799 - accuracy: 0.9833
Epoch 998/1000
8/8 [==============================] - 0s 2ms/step - loss: 0.0798 - accuracy: 0.9791
Epoch 999/1000
8/8 [==============================] - 0s 1ms/step - loss: 0.0797 - accuracy: 0.9791
Epoch 1000/1000
8/8 [==============================] - 0s 1ms/step - loss: 0.0793 - accuracy: 0.9833

<keras.callbacks.History at 0x1e6a0576be0>
```

Senaryo 7 *test_size=0.2,random_state=100*

Yorum 7: Veri setini, eğitim aşamasında verinin yüzde 80’i oranında olacak ve verinin yüzde 20’si test aşamasında olacak şekilde ayırdığımızda; 07 loss, 98 accuracy değerleri elde ederiz. Bu değerler daha da iyi tahminler almamızı sağlar.

DENEY 8

Python ayrılmış veri setindeki verileri her seferinde farklı yerlerinden böler. “random_state”, değeri her seferinde belirlenen değere göre bölme işlemi yapar.

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)
```

“random_state” parametresi, önceki deneylerde olan 100 değeri yerine 0 değeri aldığında oluşan sonuç:

```
In [316]: ann.fit(X_train,y_train,batch_size=32,epochs=1000)
          Epoch 992/1000
          8/8 [==============================] - 0s 1ms/step - loss: 0.0966 - accuracy: 0.9540
          Epoch 993/1000
          8/8 [==============================] - 0s 1ms/step - loss: 0.0963 - accuracy: 0.9540
          Epoch 994/1000
          8/8 [==============================] - 0s 1ms/step - loss: 0.0961 - accuracy: 0.9540
          Epoch 995/1000
          8/8 [==============================] - 0s 1ms/step - loss: 0.0965 - accuracy: 0.9498
          Epoch 996/1000
          8/8 [==============================] - 0s 1ms/step - loss: 0.0963 - accuracy: 0.9540
          Epoch 997/1000
          8/8 [==============================] - 0s 1ms/step - loss: 0.0962 - accuracy: 0.9540
          Epoch 998/1000
          8/8 [==============================] - 0s 1ms/step - loss: 0.0959 - accuracy: 0.9540
          Epoch 999/1000
          8/8 [==============================] - 0s 1ms/step - loss: 0.0960 - accuracy: 0.9540
          Epoch 1000/1000
          8/8 [==============================] - 0s 1ms/step - loss: 0.0957 - accuracy: 0.9540
Out[316]: <keras.callbacks.History at 0x1e695193070>
```

Senaryo 8 *test_size=0.2, random_state=0*

Yorum 8: Belirlenen değere göre bölme işlemi yapan “random_state” parametresi, 100 değeri yerine 0 değeri aldığında 09 loss, 95 accuracy değerleri elde edilmektedir.

Çıkarım: “random_state” parametresi, 100 değerini aldığında en verimli sonucu vermektedir.

GENEL YORUM

“test_size” parametresi 0.2, “random_state” parametresi 100, “optimizer” parametresi “adam", “batch_size” parametresi 32, “epochs” parametresi 1000 değerini aldığında en yüksek accuracy değerine ve en düşük loss değerine ulaşılmıştır.