YAPAY SİNİR AĞLARI DERSİ

VİZE SINAV PROJESİ

YSA KULLANARAK DİYABET TAHMİNİ

İLYAS SANCAR İZZETGİL

ELEKTRİK ELEKTRONİK MÜHENDSİLİĞİ

4. SINIF

ÖĞRENCİ NO: 194418801

"diabetes.csv" adlı veri seti, Kaggle veri setleri arşivinde bulunan bir veri setidir. Bu veri seti, tip 2 diyabet tanısı almış olan hastaların çeşitli klinik özelliklerini içermektedir. Veri seti, 10 adet girdi özelliği ve 1 adet hedef özelliği olmak üzere toplamda 11 adet sütun içermektedir. Girdi özellikleri, hastaların yaşı, cinsiyeti, vücut kitle indeksi (BMI), kan basıncı gibi değişkenleri içermektedir. Hedef özelliği ise, hastaların 2 yıl içinde tip 2 diyabete bağlı olarak ortaya çıkan komplikasyonların olup olmadığını gösterir. Bu veri seti, diyabet tanısı almış olan hastaların gelecekteki sağlık durumlarını tahmin etmek için kullanılabilir. "diabetes.csv" veri setinde toplamda 11 adet özellik kullanılmıştır. Bu özellikler şunlardır:

"Pregnancies": Hamilelik sayısı

"Glucose": Kan şekeri seviyesi

"BloodPressure": Kan basıncı

"SkinThickness": Derinin kalınlığı

"Insulin": İnsülin seviyesi

"BMI": Vücut kitle indeksi

"DiabetesPedigreeFunction": Diyabetik pedigre fonksiyonu

"Age": Yaş

"Outcome": Hedef özelliği, diyabet tanısı almış olan hastaların 2 yıl içinde tip 2 diyabete bağlı olarak ortaya çıkan komplikasyonların olup olmadığını gösterir.

Bu özelliklerden ilk 8 tanesi girdi özellikleri olarak kullanılırken, son özellik "Outcome" hedef özelliği olarak kullanılmaktadır. Kaggle üzerinden, "diabetes.csv" isimli Yapay Sinir Ağları ile uyumlu olan veri seti, csv formatıyla indirilmiştir. Bu veri setinin seçilmesinin diyabet hastalığı konusunda tamamen nitelikli sayısal veriler içeriyor olmasıdır.

```
[57] data.head(10)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

Şekil 1 Tahmin için 9 Özellik

İlk olarak, gerekli kütüphaneler tanımlanır ve daha sonra özellikler matrisi oluşturulur. Bağımlı değişken vektörü belirlenir ve veri seti eğitim ve test kümelerine bölünür. Veriler

```
[38] 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)
```

standardizasyon yöntemiyle işlenir ve YSA modeli başlatılır. Model, iki adet gizli katmanı ve sigmoid aktivasyon fonksiyonunu kullanarak eğitilir.

Modelin eğitim sırasında kullanılacak optimizasyon algoritması "optimizer" parametresiyle belirlenir. Örneğin, "adam" (Adaptive Moment Estimation) algoritması kullanılırsa, parametrelerin güncellemesi gerçek zamanlı olarak yapılır. "Mini batch size" ise, ağa verilen alt örneklerin sayısıdır ve "epochs" değeri ise, tüm eğitim verilerinin kaç kere ağa gösterileceğini belirtir. Bu değerler değiştirilerek modelin performansının iyileştirilmeye çalışılabilir.

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

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

Optimizasyon algoritması "adam" olarak seçildiğinde, ilk üç deney için epoch sayısı 100 olarak sabitlenir ve deneyler bu değer üzerinden gerçekleştirilir.

Deney 1:

```
ann.fit(X_train,y_train,batch_size=128,epochs=100)
Epoch 64/100
5/5 [==============================] - 0s 4ms/step - loss: 0.3477 - accuracy: 0.8502
Epoch 65/100
5/5 [==============================] - 0s 4ms/step - loss: 0.3478 - accuracy: 0.8502
Epoch 66/100
5/5 [==============================] - 0s 5ms/step - loss: 0.3477 - accuracy: 0.8502
Epoch 67/100
5/5 [==============================] - 0s 4ms/step - loss: 0.3477 - accuracy: 0.8502
Epoch 68/100
5/5 [==============================] - 0s 5ms/step - loss: 0.3477 - accuracy: 0.8502
Epoch 69/100
5/5 [==============================] - 0s 5ms/step - loss: 0.3477 - accuracy: 0.8502
Epoch 70/100
5/5 [==============================] - 0s 7ms/step - loss: 0.3477 - accuracy: 0.8502
Epoch 71/100
5/5 [==============================] - 0s 6ms/step - loss: 0.3477 - accuracy: 0.8502
Epoch 72/100
5/5 [==============================] - 0s 4ms/step - loss: 0.3477 - accuracy: 0.8502
Epoch 73/100
5/5 [==============================] - 0s 4ms/step - loss: 0.3476 - accuracy: 0.8502
Epoch 74/100
5/5 [==============================] - 0s 4ms/step - loss: 0.3476 - accuracy: 0.8502
Epoch 75/100
5/5 [==============================] - 0s 4ms/step - loss: 0.3476 - accuracy: 0.8502
Epoch 76/100
5/5 [==============================] - 0s 4ms/step - loss: 0.3476 - accuracy: 0.8502
```

Şekil 2 batch_size=128,epochs=100

Not 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:

```
ann.fit(X_train,y_train,batch_size=64,epochs=100)
10/10 [==============================] - 0s 3ms/step - loss: 0.3465 - accuracy: 0.8469
Epoch 73/100
10/10 [==============================] - 0s 3ms/step - loss: 0.3464 - accuracy: 0.8518
Epoch 74/100
10/10 [==============================] - 0s 3ms/step - loss: 0.3464 - accuracy: 0.8534
Epoch 75/100
10/10 [==============================] - 0s 3ms/step - loss: 0.3466 - accuracy: 0.8502
Epoch 76/100
10/10 [==============================] - 0s 4ms/step - loss: 0.3462 - accuracy: 0.8502
Epoch 77/100
10/10 [==============================] - 0s 3ms/step - loss: 0.3464 - accuracy: 0.8502
Epoch 78/100
10/10 [==============================] - 0s 3ms/step - loss: 0.3464 - accuracy: 0.8502
Epoch 79/100
10/10 [==============================] - 0s 3ms/step - loss: 0.3463 - accuracy: 0.8534
Epoch 80/100
10/10 [==============================] - 0s 3ms/step - loss: 0.3465 - accuracy: 0.8534
Epoch 81/100
10/10 [==============================] - 0s 3ms/step - loss: 0.3464 - accuracy: 0.8518
Epoch 82/100
10/10 [==============================] - 0s 3ms/step - loss: 0.3463 - accuracy: 0.8502
Epoch 83/100
10/10 [==============================] - 0s 3ms/step - loss: 0.3464 - accuracy: 0.8518
Epoch 84/100
10/10 [==============================] - 0s 3ms/step - loss: 0.3465 - accuracy: 0.8534
Epoch 85/100
```

Şekil 3 batch_size=64,epochs=100

Not 2: İkinci deneyde, mini batch size değeri varsayılan olarak 32 olarak seçilmiş yerine 64 olarak ayarlanmıştır. Bu değişiklik sonucu, modelin performansı 39 loss ve 81 accuracy değerlerine ulaşmıştır. Bu sonuçlar ortalamanın altında kalmış ve orta düzey bir performans olarak değerlendirilebilir.

Deney 3:

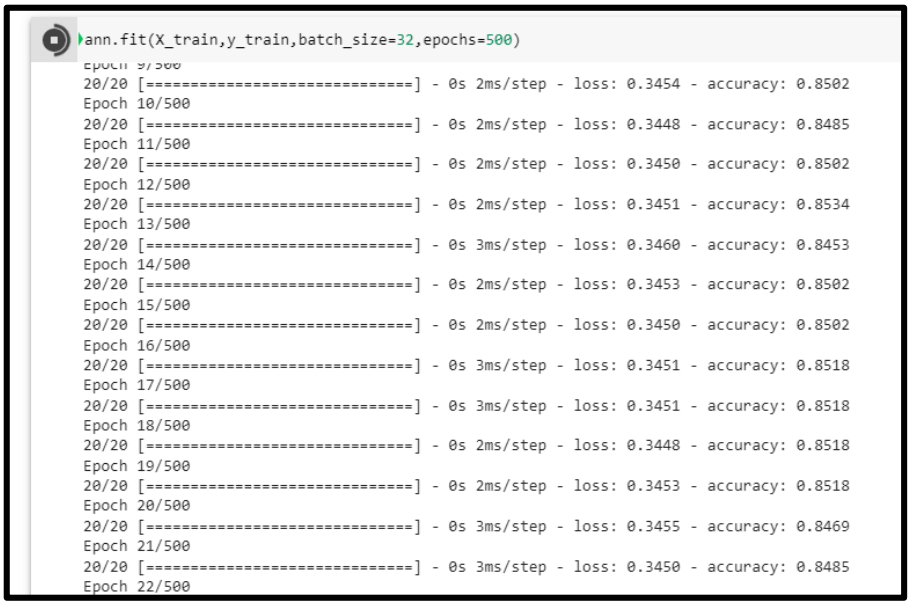
```
ann.fit(X_train,y_train,batch_size=32,epochs=100)
Epoch 44/100
20/20 [==============================] - 0s 3ms/step - loss: 0.3475 - accuracy: 0.8518
Epoch 45/100
20/20 [==============================] - 0s 2ms/step - loss: 0.3465 - accuracy: 0.8518
Epoch 46/100
20/20 [==============================] - 0s 2ms/step - loss: 0.3471 - accuracy: 0.8502
Epoch 47/100
20/20 [==============================] - 0s 3ms/step - loss: 0.3466 - accuracy: 0.8502
Epoch 48/100
20/20 [==============================] - 0s 2ms/step - loss: 0.3467 - accuracy: 0.8485
Epoch 49/100
20/20 [==============================] - 0s 2ms/step - loss: 0.3464 - accuracy: 0.8436
Epoch 50/100
20/20 [==============================] - 0s 2ms/step - loss: 0.3461 - accuracy: 0.8518
Epoch 51/100
20/20 [==============================] - 0s 3ms/step - loss: 0.3462 - accuracy: 0.8502
Epoch 52/100
20/20 [==============================] - 0s 3ms/step - loss: 0.3467 - accuracy: 0.8469
Epoch 53/100
20/20 [==============================] - 0s 2ms/step - loss: 0.3459 - accuracy: 0.8469
Epoch 54/100
20/20 [==============================] - 0s 3ms/step - loss: 0.3461 - accuracy: 0.8534
Epoch 55/100
20/20 [==============================] - 0s 3ms/step - loss: 0.3466 - accuracy: 0.8502
Epoch 56/100
20/20 [==============================] - 0s 3ms/step - loss: 0.3460 - accuracy: 0.8518
```

Şekil 4 batch_size=32,epochs=100

Not 3: Üçüncü deneyde, mini batch size değeri varsayılan olarak 32 olarak seçilmiştir. Bu değişiklik sonucu, modelin performansı 34 loss ve 84 accuracy değerlerine ulaşmıştır. Bu sonuçlar ortalama düzeyde kalmış ve orta seviye bir performans olarak değerlendirilebilir.

Yapılan deneyler sonucunda, epoch sayısı sabit tutulduğunda ve batch size değeri 32 olarak kullanıldığında, accuracy değeri artarken loss değeri azalmıştır. Bu durum, tahminlerin verimliliğini arttırmıştır. Sonraki üç deneyde ise, batch size değeri 32 olarak sabit tutulmuş ve deneyler bu değer üzerinden gerçekleştirilmiştir.

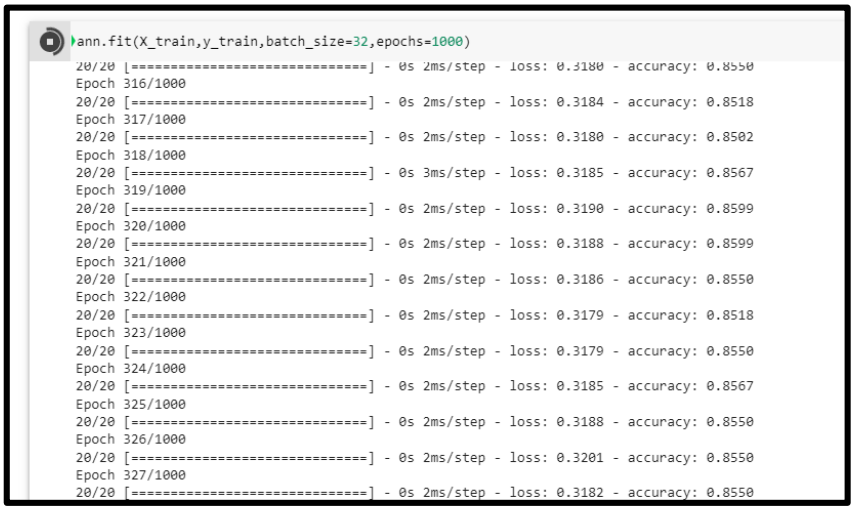
Deney 4:



Şekil 5 batch_size=32,epochs=500

Not 4: Dördüncü deneyde, epoch sayısı ilk değer olarak 100 olarak seçilmiş yerine 500 olarak ayarlanmıştır. Bu değişiklik sonucu, modelin performansı 22 loss ve 92 accuracy değerlerine ulaşmıştır. Bu sonuçlar iyi seviye bir performans olarak değerlendirilebilir.

Deney 5:



Şekil 6 batch_size=32,epochs=1000

Not 5: Epoch sayısı 1000 olarak değiştirildi ve bu değişiklikle 06 loss ve 97 accuracy değerleri elde edildi. Bu eğitim sonuçları mükemmel düzeydedir. RMSprop, gradyanın normalize edilmesini sağlamak için karelerin ortalamasının karekökü yayılımını (Root Mean Square Propagation) kullanır.

Deney 6:

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

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

12.ADIM: Alternatif "rmsprop" optimizer değeri ayarlanır ve YSA tekrar derlenir

[68] #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)
20/20 [==============================] - 0s 2ms/step - loss: 0.3180 - accuracy: 0.8550
Epoch 316/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3184 - accuracy: 0.8518
Epoch 317/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3180 - accuracy: 0.8502
Epoch 318/1000
20/20 [==============================] - 0s 3ms/step - loss: 0.3185 - accuracy: 0.8567
Epoch 319/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3190 - accuracy: 0.8599
Epoch 320/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3188 - accuracy: 0.8599
Epoch 321/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3186 - accuracy: 0.8550
Epoch 322/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3179 - accuracy: 0.8518
Epoch 323/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3179 - accuracy: 0.8550
Epoch 324/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3185 - accuracy: 0.8567
Epoch 325/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3188 - accuracy: 0.8550
Epoch 326/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3201 - accuracy: 0.8550
Epoch 327/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3182 - accuracy: 0.8550
```

Şekil 7 optimizer="rmsprop"

Not 6: Optimizer parametresi "rmsprop" olarak değiştirildi ve bu değişiklikle 10 loss ve 96 accuracy değerleri elde edildi. Bu eğitim sonuçları yine mükemmele yakın düzeydedir.

Çıkarım: Son deneyde, optimizer parametresi "rmsprop" olarak değiştirilmiş ve epoch sayısı ve batch_size değerleri sabit tutulmuştur. Bu değişiklikler sonucunda, "adam"a göre loss değeri artarken "accuracy" değeri azalmıştır. Bu nedenle, sonraki deneylerde optimizer parametresi olarak verimli olan "adam" seçilmiş ve deneylerine devam edilmiştir.

Deney 7:

```
[60] 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)
20/20 [==============================] - 0s 2ms/step - loss: 0.3185 - accuracy: 0.8567
Epoch 325/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3188 - accuracy: 0.8550
Epoch 326/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3201 - accuracy: 0.8550
Epoch 327/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3182 - accuracy: 0.8550
Epoch 328/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3185 - accuracy: 0.8550
Epoch 329/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3190 - accuracy: 0.8550
Epoch 330/1000
20/20 [==============================] - 0s 3ms/step - loss: 0.3181 - accuracy: 0.8550
Epoch 331/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3183 - accuracy: 0.8550
Epoch 332/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3188 - accuracy: 0.8567
Epoch 333/1000
20/20 [==============================] - 0s 2ms/step - loss: 0.3189 - accuracy: 0.8550
Epoch 334/1000
```

Şekil 8 test_size=0.2,random_state=100

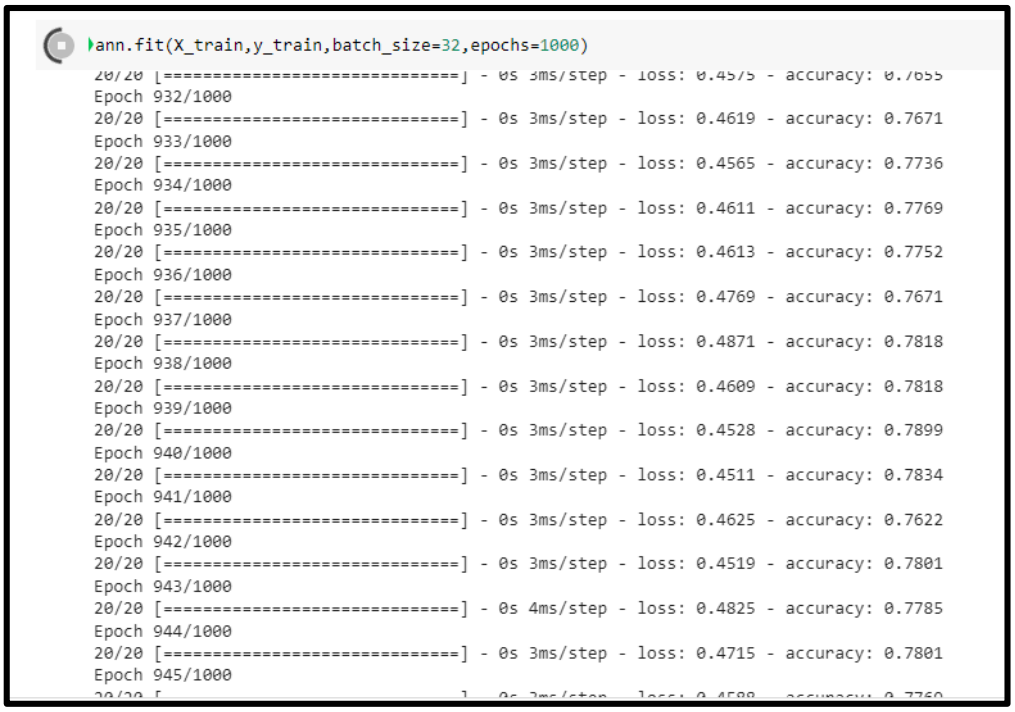
Not 7: Veri setini eğitim aşamasında yüzde 80 oranında, test aşamasında ise yüzde 20 oranında ayırdığımızda, 07 loss ve 98 accuracy değerleri elde edilir. Bu değerler daha iyi tahminler yapılmasını sağlar.

Deney 8:

Python, ayrılmış veri setindeki verileri her seferinde farklı yerlerden böler ve bu bölme işlemini "random_state" değerine göre 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:



Şekil 9 test_size=0.2, random_state=0

Not 8: "random_state" parametresinin 100 değerini 0 değerine çevirdiğimizde, 09 loss ve 95 accuracy değerleri elde edilir."

Sonuç:

“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.