Kan Transfüzyon Hizmet Merkezi Veri Setinde Karar Ağacı ve RNN Sınıflayıcıları

TBL332 Yazılım Geliştirme Laboratuvarı-II

Yunus Emre COŞKUN
Bilişim Sistemleri Mühendisliği
İstanbul, Türkiye
y.emrecoskun24@gmail.com

***Özet*—Makine öğrenmesi ve istatistikte ; sınıflandırma , bilgisayar programlarının verilen veri girişinden öğrendiği ve sonrasında yeni gözlemleri sınıflandırmak için bu öğrenmeyi kullandığı denetimli öğrenme yaklaşımıdır. Makine öğrenmesinde veri setlerinin sınıflandırma algoritmaları vardır. Bizim kullanıcağımız olan veri setimiz Kan Transfüzyon Hizmet Merkezi ve kullanıcağımız sınıflandırma algoritmaları Decision Tree (Karar Ağacı) ve Recurrent Neural Network(Tekrarlayan Sinir Ağları). Kan Transfüzyon Hizmet Merkezi Veri Seti bilgisi ise Tayvan'da yapılan bağışlar ve Mart 2007' de bağışın yapıldığını veritabanına kayıt eden bir veri setidir. Bu veri setimiz ile karar ağacı ile sınıflandırmada yapılacak işlem ağaç yapısı oluşturarak verisetinin daha anlaşılır ve optimum hale getirilir. Karar ağacı ile veri setinin yorumlanması kolaydır. Tekrarlayan Sinir Ağları ile yapılacak işlem ise önceki verilerin çıkışından bir sonraki verinin girişini tahmininde bulunmasıdır.**

Anahtar Kelimeler—Makine Öğrenmesinde sınıflandırma, Kan Transfüzyon Hizmet Merkezi, Decision Tree, Recurrent Neural Network

I. GİRİŞ

Makine öğrenimi, bilgisayarların algılayıcı verisi ya da veritabanları gibi veri türlerine dayalı öğrenimini olanaklı kılan algoritmaların tasarım ve geliştirme süreçlerini konu edinen bir bilim dalıdır. Makine öğrenmesinde veri setlerini sınıflandırma algoritma çeşitleri bulunmaktadır. Bu algoritmalardan ikisi Decision Tree (Karar Ağacı) ve Recurrent Neural Network (Tekrarlayan Sinir Ağları). Karar ağaçları, ağaç yapısı formunda sınıflandırma ya da regrasyon modeli oluşturur. İlişkili karar ağaçları kademeli olarak gelişir ve veri kümeleri küçük küçük kümelere ayrılır. Tekrarlayan bir sinir ağı, düğümler arasındaki bağlantıların geçici bir dizi boyunca yönlendirilmiş bir grafik oluşturduğu bir yapay sinir ağı sınıfıdır. Bu, zamansal dinamik davranış sergilemesini sağlar.

II. MATERYAL VE METOT

A. Veri Seti

TABLO 1. Veri Seti

Recency	Frequency	Monetary	Time	BloodinMarch2017
2	50	12500	98	1
0	13	3250	28	1
1	16	4000	35	1
2	20	5000	45	1
1	24	6000	77	0
…	…	…	…	…
72	1	250	72	0

Tayvan'daki Hsin-Chu şehrindeki Kan Transfüzyon Servis Merkezi'nden alınan kan bağışının 3 ayda bir yapıldığı veri setidir. Veri setimizde toplam 748 veri vardır. Değişkenleri ise; Recency, Frequency, Monetary, Time, BloodinMarch2007 olmak üzere 5 değişken bulunmaktadır.

Recency; Son bağıştan bu yana geçen ay sayısı.

Frequency; Toplam bağış sayısı.

Monetary; Toplam bağışlanan kan birimi(CC Para değerinde)

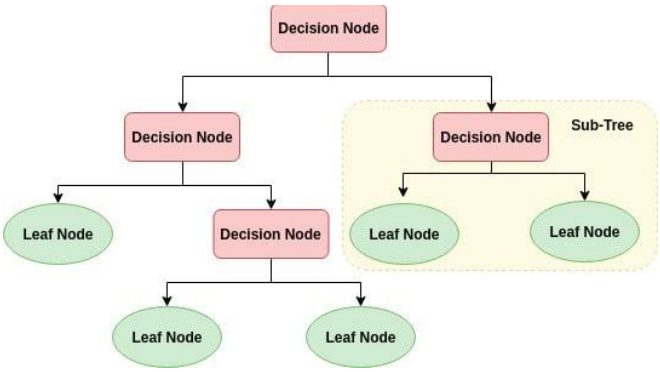
Time; İlk bağıştan bu yana geçen süre – ay

BloodinMarch2007; Mart 2007' de kan bağışı yapıldı mı?(1 evet, 0 hayır)

B. Karar Ağacı

Karar Ağacı algoritması, bir iç düğümün özelliği temsil ettiği, dal bir karar kuralını temsil ettiği ve her bir yaprak düğümünün sonucu temsil ettiği akış şeması benzeri bir ağaç yapısıdır. Karar Ağacında en üstteki düğüm hedef düğüm olarak bilinir. Hedef değişkenimizi entropi hesaplaması sonucu belirliyoruz. Ağacı akış şeması şekilde oluşturuyoruz. Bu ağaç İnsanların veri setlerini daha iyi anlayabilmesi ve daha iyi yorumlandırılması amaçlı yapılır.

Identify applicable funding agency here. If none, delete this text box.



Şekil 1. Karar Ağacı

Değişken entropilerini hesaplamak için;

Öncelikle verisetinde bağımlı verinin entropisi hesaplanır(2).

$$Info(D) = - \sum^{m}_{i=1} pi \log_2(pi) \quad (1)$$

Her bir değişkenin entropisi hesaplanır(2).

$$Info_{Degisken}(D) = \sum^{v}_{j=1} |D_j|/|D| \times Info(Dj) \quad (2)$$

Son olarak değişkenin nitelik değeri Gain formülü ile bulunur.

$$Gain(Degisken) = Info(D) - Info_{Degisken}(D) \quad (3)$$

Karar ağacımızın entropi değerini en aza indirgeyen bölünmeler yapmasını isteriz. En iyi bölünmeyi belirlemek içinde bilgi kazancını kullanırız.

$$SplitInfo_{degisken}(D) = - \sum^{v}_{j=1} |D_j|/|D| \times \log_2(|D_j|/|D|) \quad (4)$$

$|D_j|/|D|$ j bölümünün ağırlığı olarak işlev görür.

v, degisken özelliğindeki ayrık değerlerin sayısıdır.

Bu durumda kazanç değeri şu şekilde tanımlanabilir:

$$GainRatio(Degisken) = Gain(Degisken)/SplitInfo_A(D) \quad (5)$$

Başka bir karar ağacı algoritması CARD, bölme noktaları oluşturmak için Gini yöntemini kullanır. (Gini Endeksi)

$$Gini(D) = 1 - \sum^{m}_{i=1} pi^2 \quad (6)$$

Gini Dizini, her özellik için bir ikili bölmeyi dikkate alır. Her bölümün ağırlıklı bir toplamını hesaplayabiliriz. Degisken niteliğindeki bir ikili bölünme D verilerini D1 ve D2'ye bölerse, D'nin Gini dizini şöyledir:

$$Gini_A(D) = |D_1|/|D| \; Gini(D_1) + |D_2|/|D| \; Gini(D_2) \quad (7)$$

Ayrık değerli bir öznitelik olması durumunda, seçilen için minimum gini dizinini veren altküme, ayırma özniteliği olarak seçilir. Sürekli değerli öznitelikler söz konusu olduğunda, strateji her bitişik değer çiftini olası bir ayrılma noktası olarak seçmek ve ayrılma noktası olarak seçilen daha küçük gini indeksi ile nokta yapmaktır.

$$\Delta Gini(A) = Gini(D) - Gini_A(D) \quad (8)$$

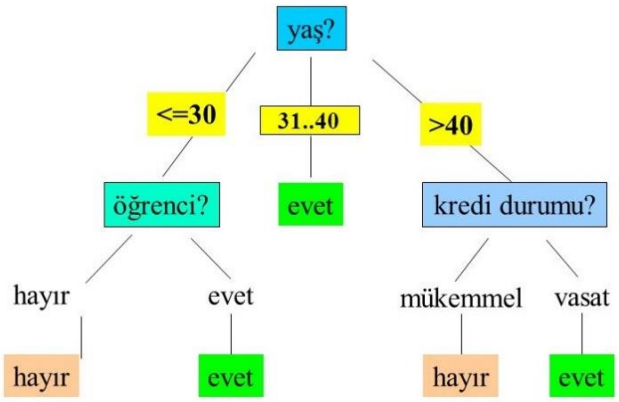
Minimum Gini dizinine sahip öznitelik, bölme özniteliği olarak seçilir.

Hedef değişkenimiz belli olduktan sonra ağaç yapısını oluşturmamız kolaylaşır. Küçük bir örnek verisetimiz ile karar ağacını gösterelim:

TABLO 2. Örnek Veri Seti

yaş	Gelir durumu	öğrenci	Kredi durumu	Biligsayarı varmı
<=30	yüksek	hayır	Vasat	hayır
<=30	yüksek	hayır	Mükemmel	hayır
31..40	yüksek	hayır	Vasat	evet
>40	orta	hayır	Vasat	evet
<=30	low	evet	Mükemmel	evet

Bilgisayara sahip olanları gösteren bir veri setinin Karar Ağacı Şekil2' deki gibidir.



Şekil 2. Karar Ağacı Örnek

En büyük nitelik değerini alan değişken yaş değişkeni olduğu için hedef değişken olmuştur. Şimdi karar ağacının artıları ve eksilerinden bahsedelim:

Artıları:

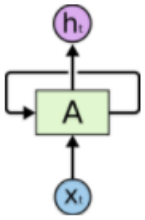
- Karar ağaçlarının yorumlanması ve görselleştirilmesi kolaydır.
- Doğrusal olmayan desenleri kolayca yakalayabilir.
- Kullanıcıdan daha az veri önişleme gerektirir.
- Değişken seçimi için uygun olan eksik değerleri tahmin etme gibi özellik kullanılabilir.
- Karar ağacının, algoritmanın parametrik olmayan doğası nedeniyle dağıtım hakkında herhangi bir varsayımı yoktur.

Eksileri:

- Gürültü verilere duyarlıdırç Gürültülü verileri değiştirebilir.
- Verilerdeki küçük değişiklik (veya varyans) farklı ağacıyla sonuçlanabilir.
- Karar ağaçları, dengesizlik veri kümesiyle önyargılı olduğundan karar ağacını oluşturmadan önce veri kümesini dengelemeniz önerilir.

C. *Tekrarlayan Sinir Ağları*

Tekrarlayan sinir ağı hafızalı bir sinir ağıdır. Bu hafızayı, önceki verilerden elde edilen bilgileri tahminlere dahil etmek için kullanılır.



Şekil 3. Tekrarlayan Sinir Ağı Yığını

Şekil 3' deki diyagramda, "A" girişi X_t, Çıkış değeri H_t.

Tekrarlayan Sinir Ağının yapısı formülü:

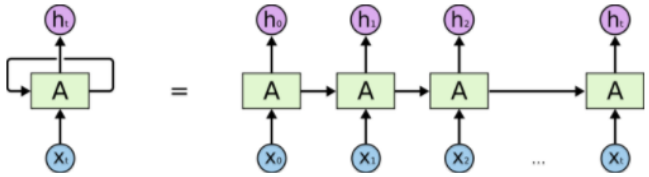
$$h_t = f_w(h_{t-1}, x_t) \quad (5)$$

4'deki formülde bulunan fonksiyonun açılımı :

$$h_t = \tanh(W_h h_{t-1} + W_{xh} x_t) \quad (6)$$

$$y_t = W_{hy} h_t \quad (7)$$

Tekrarlayan Sinir Ağlarında döngüler vardır. Bu döngüler bir adımdan diğer sinir ağına bilgi gitmesine izin verir.Tekrarlayan sinir ağı aynı ağın birden fazla kopyası olarak düşünülür ve her biri halefi bir mesaj gönderir. Döngüyü açtığımızda (Şekil 4);



Şekil 4. Tekrarlayan Sinir Ağı Yığını

Şekil 4' ü incelediğimizde bir tahminde bulunmak için önceki testten gelen girdileri ve verileri kullanırız. Bu veriler bağlam birimi olarak bilinir.

Bağlam birimi verileri ve tahmin hatası, bir sonraki tahminin yapılmasına yardımcı olmak için sisteme geri beslenir.

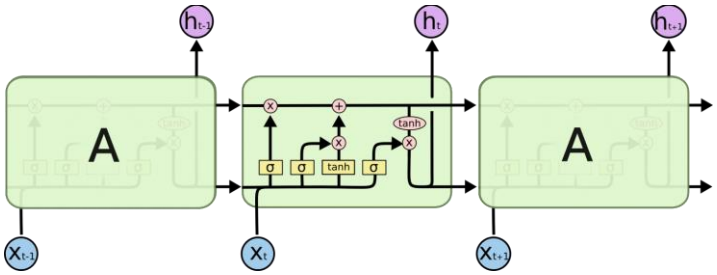
Her tahmin farklı bir zaman adımında yapılır.

RNN, önceki tahminden elde edilen bilgiyi alıp birsonraki hesaplamaya geçirmekte olan bir dinamik özelliği bulunmaktadır.

Örneğin, harf uygulamasında daha önce 'Hel' harflerini giridiğini düşünün. Ağ, bir sonraki harf tahminini yapmak için önceki harflerin bilgisini kullanabilir. Bu algoritma bir sonraki harfın 'I' olacağını makul bir şekilde tahmin edebilir. Önceki bilgiler olmazsa tahmini bulmak daha zor olur.

LSTM(UZUN KISA SÜRELİ BELLEK BİRİMLERİ)

Uzun kısa süreli bellek derin öğrenme alanında kullanılan yapay bir tekrarlayan sinir ağı mimarisidir. Standart ileri beslemeli sinir ağlarının aksine, LSTM'nin geri bildirim bağlantıları vardır. Yalnızca tek veri noktalarını değil, aynı zamanda tüm veri dizilerini işleyebilir.



Şekil 5. LSTM Yapısı

Sıradan bir LSTM ünitesi, bir hücre, bir giriş kapısı, bir çıkış kapısı ve bir unut kapısı oluşur Hücre, keyfi zaman aralıklarındaki değerleri hatırlar ve bu üç kapı, hücreye giren ve çıkan bilgi akışını düzenler.

III. DENEYSEL SONUÇLAR

Deneyler Python programlama dili ile yapılmıştır. Kan Transfüzyon Hizmet Merkezi Veri Setimizin sınıflandırma yöntemleri olan Karar Ağacı ve Tekrarlayan Sinir Ağacı algoritmalarıyla sınıflandırdık. İlk olarak Karar Ağacı ile sınıflandırma yönteminden bahsedelim.

Karar Ağacı ile Sınıflandırma (Python):

1- Önce gerekli kütüphaneleri yükledik.

2- Veri Setimizi yüklüyoruz. Veri setimizi CSV işlevini kullanarak yüklüyoruz.

3- Bağımlı ve bağımsız değişken olmak üzere iki tür değişkene bölüyoruz. Bağımlı değişkenimiz 'BloodinMarch2007' olur. Diğerleri bağımsız değişken.

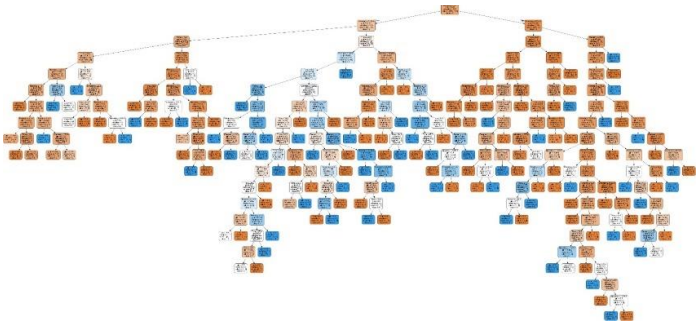
4- Model performansını anlamak için, veri kümesini bir eğitim setine ve bir test setine bölüyoruz. (train_test_split)

5- Scikit-learn'ı kullanarak bir karar ağacı modeli oluşturuyoruz. (Optimize edildi.)

6- Şimdi modeli değerlendirme aşamasında doğruluk testi yapıcağız ön görülen değer hesaplanılacak.

Accuracy: 0.7111111111111111

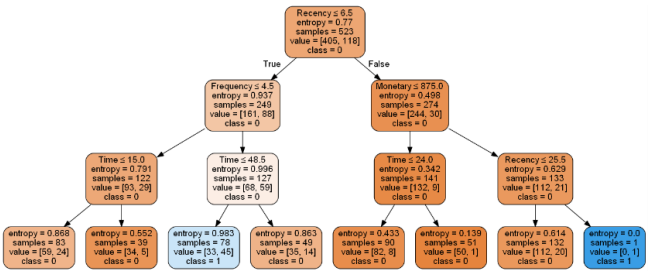
7- Karar Ağacını görselleştirme aşamasındayız. Karmaşık bir karar ağacı görseli oluştu.



Şekil 6. Karar Ağacı Görsel 1

8- Karar ağacı performansını optimize etmek için gerekli kod parçacıkları eklenmiştir. Doğruluk değeri %8 yükseldi ve karmaşık oluşan karar ağacımız daha optimize edilmiştir.

Accuracy: 0.7911111111111111



Şekil 7. Karar Ağacı Görsel 2

Bu ağaç modeli, önceki karar ağacı model grafiğinden daha az karmaşık, açıklanabilir ve anlaşılması kolay olmuştur.

9- Şimdiye kadar uyguladığımız deneyde test(%30) değerine göre hesap edilmiştir. Test değerlerini 20-30-50-80 değerleri ile sonucu hesaplayalım.

TABLO3. Karar Ağacı Sonuç Tablosu

Test Size	accuracy	precision	recall	f1-score
20	0.79	0.82	0.91	0.86
30	0.79	0.82	0.92	0.87
50	0.76	0.77	0.98	0.86
80	0.76	0.77	0.98	0.86

Tekrarlayan Sinir Ağı ile Sınıflandırma(Python):

1- Öncelikle gerekli kütüphaneleri yükledik. (LSTM ve RNN)

2- Verileri CSV işlevini kullanarak yüklüyoruz ve verileri optimize ediyoruz.

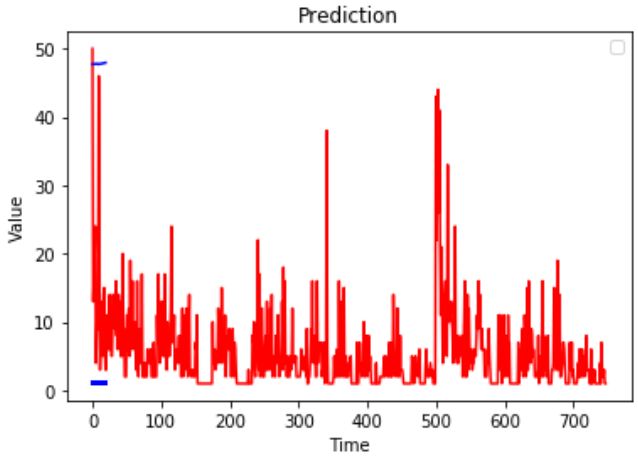
3- Tekrarlayan sinir ağlarında zaman adımlarına bakılır. Verilerin RNN tarafından anlaşılması için doğru forma girmesi için zaman aralığı ile döngü yapıyoruz.

4- Tekrarlayan sinir ağını kurma, derleme ve takma aşamasındayız; Başlat>Giriş ve gizli katmanları oluştur> Çıktı katmanını oluştur>RNN'i derle>RNN'i test setine tak

5- Tahminleri yapma aşamasındayız; tahmin ve doğruluk sonuçları ekrana yazdırıyoruz. Fakat doğruluk sonucumuz 0.50'nin altında olduğundan doğru sonuçları verdiğimize emin olamıyoruz.

TABLO4. RNN Tahmin Sonucu

Epoch 1/3 688/688 [==========] - 2s 3ms/step - loss: 3.2602 - accuracy: 0.2151
Epoch 2/3 688/688 [==========] - 2s 3ms/step - loss: 3.1427 - accuracy: 0.2297
Epoch 3/3 688/688 [==========] - 2s 3ms/step - loss: 2.8029 - accuracy: 0.2297

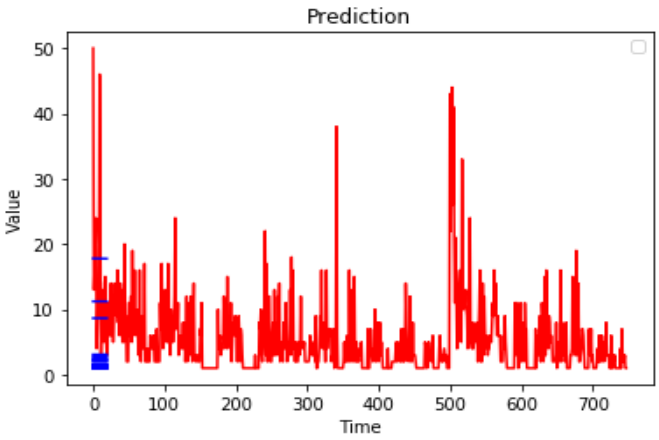


Şekil 9. Tekrarlayan Sinir Ağı Veri Görseli 2

6- Doğruluk değeri düşük aldığımızdan dolayı verileri encode ederek doğruluk değerini arttırmayı denedim. Yeni Doğruluk değeri 0.2307 olmuştur. Doğruluk değerini arttırmak için elimden gelenini yaptım fakat alabildiğim en yüksek değer 0.2307 olmuştur.

TABLO5. LSTM-RNN Tahmin Sonucu 2(Epoch=3)

Epoch 1/3 688/688 [==========] - 3s 4ms/step - loss: 2.0589- accuracy: 0.2224
Epoch 2/3 688/688 [==========] - 2s 3ms/step - loss: 3.1427 - accuracy: 0.2307
Epoch 3/3 688/688 [==========] - 2s 3ms/step - loss: 2.8029 - accuracy: 0.2307

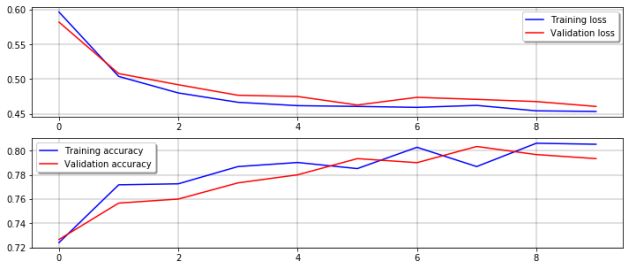


Şekil 9. Tekrarlayan Sinir Ağı Veri Görseli 2

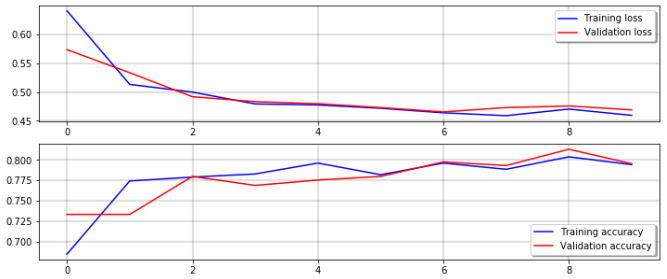
7- Doğruluk sonucu LSTM'de düşük doğruluk değeri aldığım için Simple RNN yöntemine geçiş yaptım. SimpleRNN deney sonuçları aşağıdaki tablolarda açıklanmıştır.
(NOT: Tabloda ve grafikte kullanılan terimleri açalım;
Epochs: Modellerin ağırlık değeri
Batch_size: Modele iletilecek örnek sayısı.
Loss: Modelin yaptığı tahminin, verinin gerçek değerinden farkı.
Accuracy: Doğruluk değeri.
Val_loss: Doğrulama kaybı.
VAl_acc: Doğrulama doğruluğu.)

TABLO6. Simple RNN Tablosu(Epochs=10,Batch_size=64)

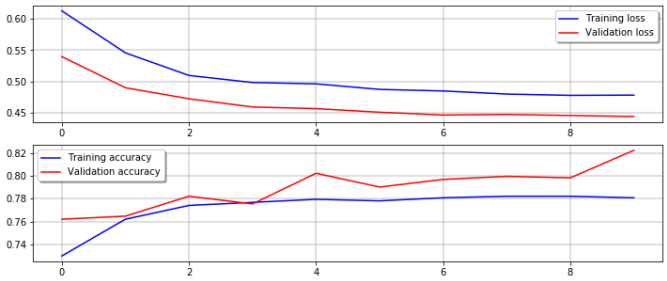
Test Size	accuracy	loss	val_loss	val_acc
20	0.8052	0.4538	0.4609	0.7933
30	0.7945	0.4594	0.4689	0.7956
50	0.7807	0.4782	0.4440	0.8222
80	0.7517	0.5089	0.4842	0.7646



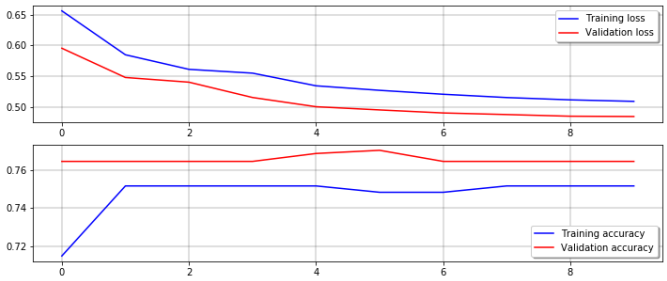
Şekil 9. RNN text=20 epochs=10 batchsize=64



Şekil 10. RNN text=30 epochs=10 batchsize=64



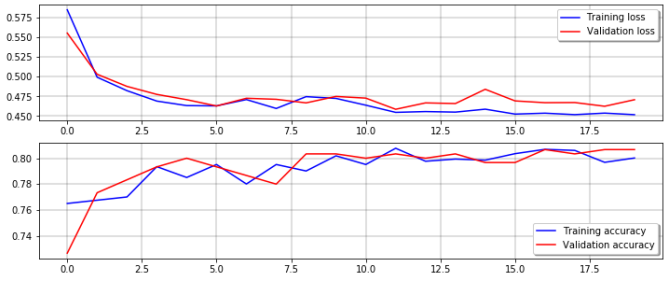
Şekil11. RNN text=50 epochs=10 bat chsize=64



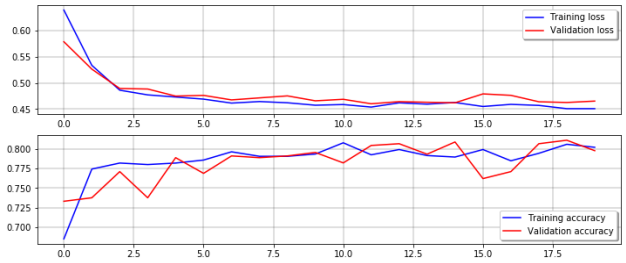
Şekil12. RNN text=80 epochs=10 bat chsize=64

TABLO7. Simple RNN Tablosu(Epochs=20,Batch_size=64)

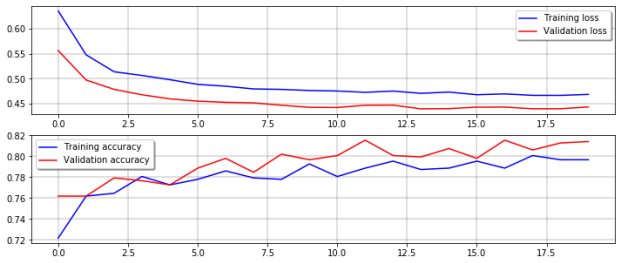
Test Size	accuracy	loss	val_loss	val_acc
20	0.8002	0.4518	0.4708	0.8067
30	0.8021	0.4509	0.4654	0.7978
50	0.7968	0.4686	0.4435	0.8142
80	0.7517	0.4884	0.4957	0.7646



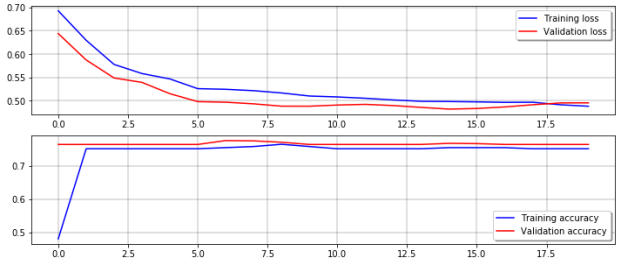
Şekil13. RNN text=20 epochs=20 bat chsize=64



Şekil14. RNN text=30 epochs=20 bat chsize=64



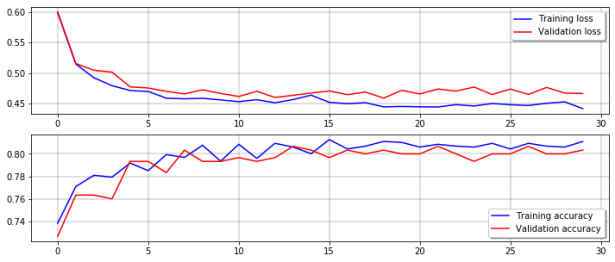
Şekil15. RNN text=50 epochs=20 bat chsize=64



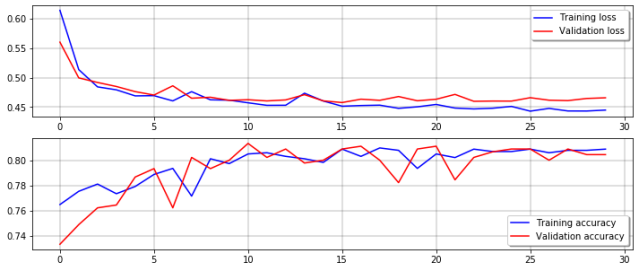
Şekil16. RNN text=80 epochs=20 bat chsize=64

TABLO8. Simple RNN Tablosu(Epochs=30,Batch_size=64)

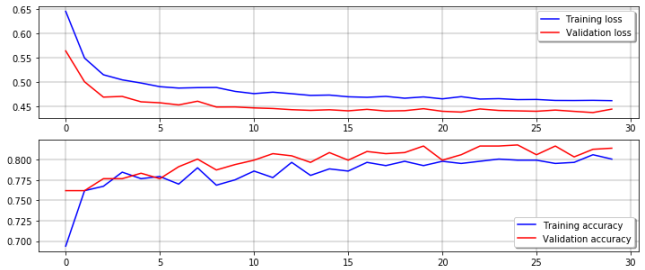
Test Size	accuracy	loss	val_loss	val_acc
20	0.8116	0.4416	0.4664	0.8033
30	0.8088	0.4451	0.4658	0.8044
50	0.8008	0.4622	0.4448	0.8142
80	0.7718	0.4657	0.4907	0.7663



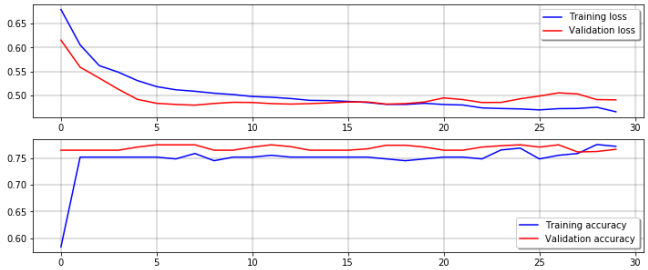
Şekil17. RNN text=20 epochs=20 bat chsize=64



Şekil18. RNN text=30 epochs=30 bat chsize=64



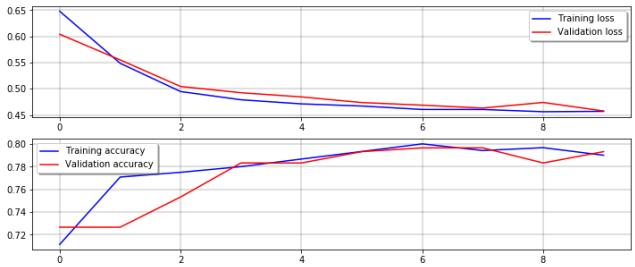
Şekil19. RNN text=50 epochs=30 bat chsize=64



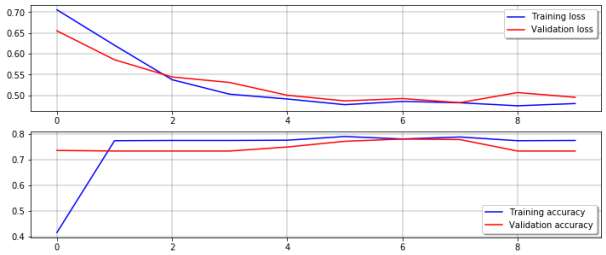
Şekil20. RNN text=80 epochs=30 bat chsize=64

TABLO9. Simple RNN Tablosu(Epochs=10,Batch_size=128)

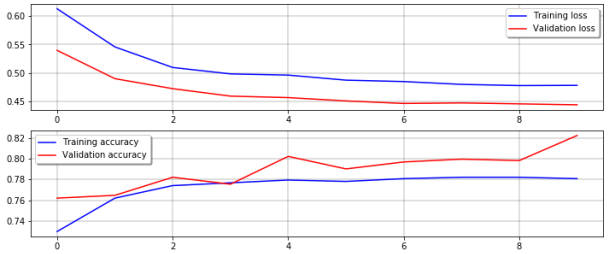
Test Size	accuracy	loss	val_loss	val_acc
20	0.7901	0.4570	0.4577	0.7933
30	0.7744	0.4805	0.4954	0.7333
50	0.7781	0.4890	0.4592	0.7674
80	0.7517	0.5220	0.4979	0.7663



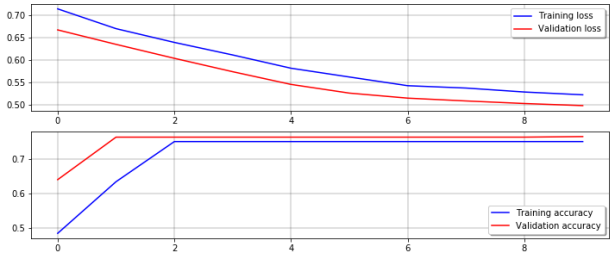
Şekil21. RNN text=20 epochs=10 bat chsize=128



Şekil22. RNN text=30 epochs=10 bat chsize=128



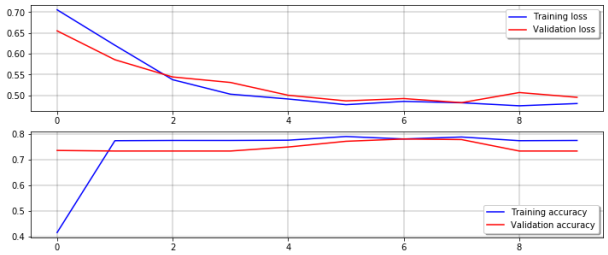
Şekil23. RNN text=50 epochs=10 batchsize=128



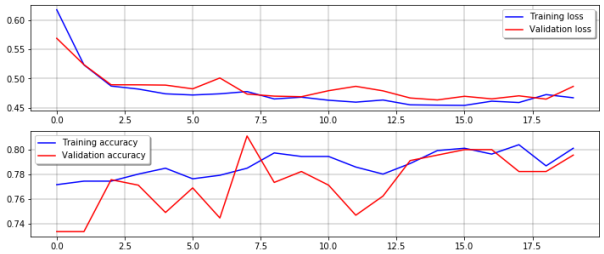
Şekil24. RNN text=80 epochs=10 batchsize=128

TABLO10. Simple RNN Tablosu(Epochs=20,Batch_size=128)

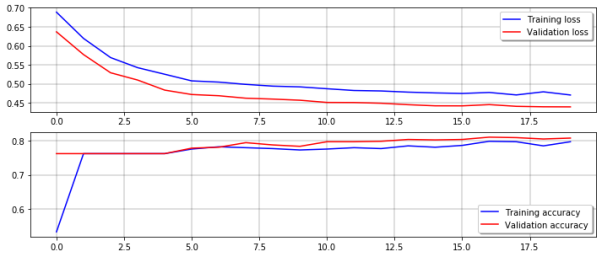
Test Size	accuracy	loss	val_loss	val_acc
20	0.8110	0.4482	0.4635	0.8000
30	0.8011	0.4669	0.4864	0.7956
50	0.7968	0.4707	0.4396	0.8075
80	0.7517	0.5220	0.4979	0.7663



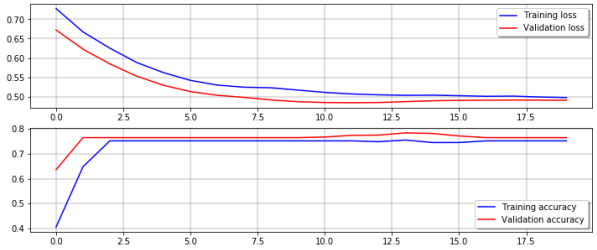
Şekil25. RNN text=20 epochs=20 batchsize=128



Şekil26. RNN text=30 epochs=20 ba chsize=128



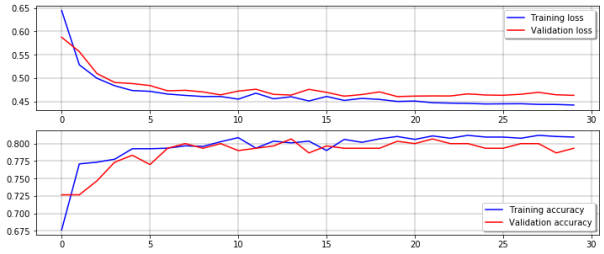
Şekil27. RNN text=50 epochs=20 batchsize=128



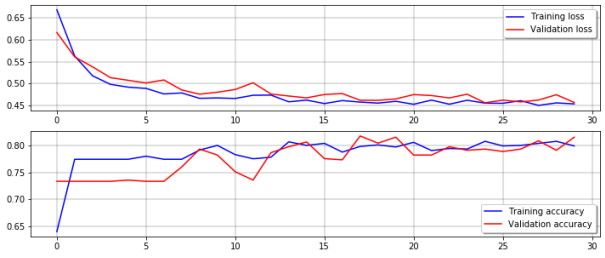
Şekil28. RNN text=80 epochs=20 batchsize=128

TABLO10. Simple RNN Tablosu(Epochs=30,Batch_size=128)

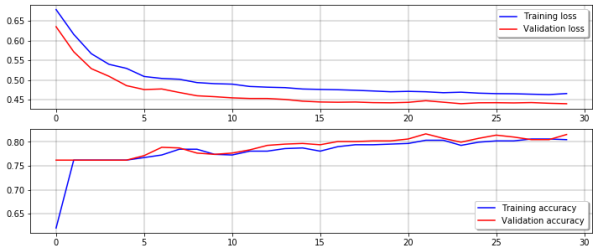
Test Size	accuracy	loss	val_loss	val_acc
20	0.8094	0.4423	0.4631	0.7933
30	0.7992	0.4534	0.4568	0.8156
50	0.8048	0.4655	0.4396	0.8155
80	0.7450	0.4905	0.4895	0.7688



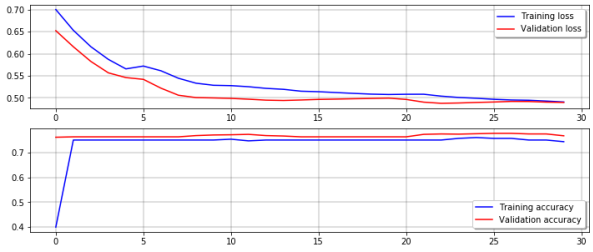
Şekil29. RNN text=20 epochs=30 batchsize=128



Şekil30. RNN text=30 epochs=30 batchsize=128



Şekil31. RNN text=50 epochs=30 batchsize=128



Şekil32. RNN text=80 epochs=30 batchsize=128

IV. SONUÇ

TABLO 3. SINIFLANDIRICILARIN DOĞRULUK SONUCU

Sınıflandırıcı	**Doğruluk Sonucu**
Karar Ağacı	0.79
Tekrarlayan Sinir Ağları(LSTM)	0.23
Tekrarlayan Sinir Ağları(SimpleRNN)	0.81

Sınıflandırma yönteminde doğruluk sonuçlarını yukarıdaki tablodaki gibidir. Karar ağacında deneysel sonuçta en yüksek değer Test(%20) ve Eğitim(%80) oranları ile 0,79 doğruluk değer sonucu alındı. LSTM yöntemi ile ise Test(%20) ve Eğitim(%80) oranları ile 0,23 doğruluk sonucu değeri aldık. Tekrarlayan Sinir Ağları LSTM yöntemi doğruluk değerlerini düşük aldığımızdan dolayı SimpleRNN yöntemini denedik ve Epochs=30, Batch_size=6 değerleri ile Test (%20) ve Eğitim(%80) oranları ile en yüksek doğruluk sonucunu 0,81 aldık.

KAYNAKLAR

[1] https://medium.com/@sengul_krdrl/maki%CC%87ne-%C3%B6%C4%9Frenmesi%CC%87nde-siniflandirma-algori%CC%87tmasi-t%C3%BCrleri%CC%87-5e0f32245889#:~:text=Makine%20%C3%B6%C4%9Frenmesi%20ve%20istatistikte%20%3B%20s%C4%B1n%C4%B1fland%C4%B1rma,%C3%B6%C4%9Frenmeyi%20kulland%C4%B1%C4%9F%C4%B1%20denetimli%20%C3%B6%C4%9Frenme%20yakla%C5%9F%C4%B1m%C4%B1d%C4%B1r%20.

[2] http://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center

[3] https://www.artificiallyintelligentclaire.com/recurrent-neural-networks-python/

[4] https://www.datacamp.com/community/tutorials/decision-tree-classification-python

[5] https://www.tensorflow.org/guide/keras/rnn

[6] https://www.datacamp.com/community/tutorials/lstm-python-stock-market?utm_source=adwords_ppc&utm_campaignid=10267161064&utm_adgroupid=102842301792&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=332602034361&utm_targetid=aud-299261629574:dsa-429603003980&utm_loc_interest_ms=&utm_loc_physical_ms=1012782&gclid=Cj0KCQjwudb3BRC9ARIsAEa-vUs7EtfWW7Ta0_ZZ2OvUU0ZBaOCiCPVHAHJWb2yZX5TqSIh08Kw1MsMaApMGEALw_wcB

[7] https://towardsdatascience.com/recurrent-neural-networks-by-example-in-python-ffd204f99470

[8] https://medium.com/@hamzaerguder/recurrent-neural-network-nedir-bdd3d0839120

[9] https://towardsdatascience.com/rnn-simplified-a-beginners-guide-cf3ae1a8895b

[10] https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb

[11] https://towardsdatascience.com/what-is-a-decision-tree-22975f00f3e1

[12] https://www.youtube.com/watch?v=BSpXCRTOLJA

[13] https://www.youtube.com/watch?v=z_PCJeDOFOk

[14] https://scikit-learn.org/stable/modules/tree.html

[15] https://towardsdatascience.com/visualizing-decision-trees-with-python-scikit-learn-graphviz-matplotlib-1c50b4aa68dc

[16] https://www.analyticsvidhya.com/blog/2019/01/fundamentals-deep-learning-recurrent-neural-networks-scratch-python/

[17] https://adventuresinmachinelearning.com/recurrent-neural-networks-lstm-tutorial-tensorflow/

[18] https://pythonprogramming.net/recurrent-neural-network-rnn-lstm-machine-learning-tutorial/