MAKİNE ÖĞRENMESİ

Ödev ve Proje Dosyası

Zahide Nur ŞENSOY

22410082028

2024-2025 Eğitim Öğretim Yılı

MAKİNE ÖĞRENMESİ

Yapay Sinir Ağı (LSTM) Kullanarak IMDB Film Yorumları Üzerinden Duygu Analizi Proje Kısmı

Zahide Nur ŞENSOY

22410082028

2024-2025 Eğitim Öğretim Yılı

**Yapay Sinir Ağları (LSTM) Kullanarak IMDB Film Yorumlarından Duygu Analizi**

**Projenin Özeti**

Bu projede, IMDB film yorumlarını kullanarak duygu analizi gerçekleştirilmiştir. Projede kullanılan veri seti, 50.000 IMDB film yorumundan oluşmaktadır. Yorumlar pozitif ve negatif olarak sınıflandırılmıştır. Veri seti, eğitim ve test kümelerine ayrılarak modellenmiştir. Proje modellemesinde metin verilerini işlemek ve analiz etmek için doğal dil işleme (NPL) yöntemi kullanılmıştır. -LSTM- Long Short Term Memory (Uzun kısa dönemli bellek) Tabanlı bir yapay sinir ağı modeli geliştirerek, yorumların duygu durumunu tahmin eden bir model oluşturulmuş ve modelin performansı değerlendirilmiştir.

**Projede Kullanılan Algoritma**

Projede, kullanılan derin öğrenme modeli olan LSTM (Long Short- Term Memory); Yapay Sinir Ağında (Artificial Neural Network – ANN) bir yapay sinir ağı türü olan Tekrarlanan Sinir Ağı (Recurrent Neural Network – RNN) mimarisine dayalı bir yapay sinir ağı türüdür. Projeyi anlayabilmek için kısaca bu 3 terime bakalım;

**Yapay Sinir Ağı (Artificial Neural Network - ANN) :** Yapay sinir ağı, insan beynindeki nöronların çalışma prensibini taklit eden bir makine öğrenmesi algoritmasıdır. 3 Katmandan oluşur;

1. **Giriş Katmanı (Input Layer) :** Veriyi alır.
2. **Gizli Katmanlar (Hidden Layers):** Veriyi işler, öğrenmeyi sağlar.
3. **Çıkış Katmanı (Output Layers):** Tahmin edilen sonuçları üretir.

Karmaşık matematiksel ilişkilerden öğrenme yaparak tahmin ve sınıflandırma problemlerini çözer. Yapay sinir ağları genellikle bağımsız veriler üzerinde çalışır ve verilerin ardışık olması gerekmez.

Aşağıdaki görsel, ANN yapısına bir örnektir;

diyagram, daire, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Tekrarlanan Sinir Ağı (Reccurrent Neural Network – RNN) :** RNN, özellikle zaman bağımlı veya ardışık veri üzerinde çalışan bir yapay sinir ağı türüdür. Bir önceki adımdan elde edilen bilgiyi sonraki adıma taşıyarak geçmiş verileri hatırlayabilmektedir. Zaman serisi analizi (ör. hisse senedi tahmini), Doğal dil işleme (ör. metin analizi, dil modeli), Ses ve video analizi vs. gibi alanlarda kullanılır. Diğer yapay sinir ağlarından farkı, bir geri besleme mekanizmasıyla geçmiş adımlardan gelen bilgiyi işlemesidir.

Aşağıdaki görsel, RNN yapısına bir örnektir;diyagram, daire, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**-LSTM- (Long Short-Term Memory) :** Uzun kısa süreli bellek derin öğrenme alanında kullanılan yapay bir yinelemeli sinir ağı (RNN) mimarisidir. LSTM ağları, zaman serisi verilerine dayanarak sınıflandırmak, işlemek ve tahminler yapmak için çok uygundur. Çünkü bir zaman serisindeki önemli olaylar arasında bilinmeyen süreli gecikmeler olabilir. LSTM'ler, uzun dönemli bağımlılıkları öğrenmek için geliştirilmiştir. Avantajı; diğer RNN’lerin yaşadığı gradyan kaybı (vanishing gradient) problemini çözmesidir.

Anlık veriyi (resim gibi) değil, veri dizilerini (konuşma veya video gibi) de işleyebilir. Örneğin; Metin üretimi, el yazısı üretimi, el yazısı tanıma, müzik üretimi, dil çevirisi, ağ trafiğinde anomali veya IDS'lerde (saldırı tespit sistemleri) tespiti gibi görevleri de başarıyla işleyebilir.

Sıradan bir LSTM ünitesi, bir hücre, bir giriş kapısı (Önemli yeni bilgiyi ekler), bir çıkış kapısı (Çıktıyı üretir) ve bir unutma kapısından (Gereksiz bilgiyi siler) oluşur. Hücre, değişken uzunlukta zaman aralıklarındaki değerleri hatırlar ve bu üç kapı, hücreye giren ve çıkan bilgi akışını düzenler.

Aşağıdaki görsel, LSTM yapısına bir örnektir;diyagram, ekran görüntüsü, plan içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Verilerden Örnek Satır Veri ve Verideki Değişkenlerin Açıklaması**

Proje, Kaggle'dan alınan IMDB film yorumları veri seti üzerinden gerçekleştirilmiştir. Yorumlar iki sınıfa ayrılmıştır;

1. **Positive :** Olumlu yorumlar.
2. **Negative :** Olumsuz yorumlar.

Kullanılan veri setinin ilk 5 satırından örnekler;

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Review:** Filmler hakkında kullanıcıların yazdığı yorumlar.

**Sentiment:** Yorumun duygu durumu (pozitif veya negatif).

**Eğitim Süreci ve Grafikleri**

**Eğitim ve Doğrulama Doğruluğu:** Eğitim doğruluğu, modelin eğitim verileri üzerindeki performansını gösterir. Doğrulama doğruluğu ise modelin daha önce görmediği doğrulama verileri üzerindeki performansını gösterir. Grafiklerde doğrulama doğruluğunun belirli bir noktada sabitlenmesi veya düşmesi, modelin aşırı öğrenme (overfitting) yapabileceğini gösterebilir. Buna göre veri setimiz üstünde bir yorum yapacaksak, Eğitim sırasında doğruluk oranının sürekli arttığı, doğrulama doğruluğunun ise belli bir noktadan sonra sabit veya azaldığı gözlemleniyor. Eğitim doğruluğunda, 1. epoch'ta %60,59 iken 4. epoch sonunda %88,45'e yükselmiş. Bu, modelin eğitim verisi üzerinde öğrenme gerçekleştirdiğini ve doğruluğun sürekli arttığını gösteriyor.

Doğrulama doğruluğunda ise 1. epoch'ta %76,49 iken 4. epoch'ta %76,57'de kalmış ve fazla bir iyileşme göstermemiş. Bu durum, modelin doğrulama verisine uyum sağlayamadığını ve aşırı öğrenme (overfitting) riskini işaret ediyor olabilir.

**metin, çizgi, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Eğitim ve Doğrulama Kayıpları :** Eğitim kaybı, modelin eğitim verileri üzerindeki hatasını gösterir. Doğrulama kaybı ise doğrulama verileri üzerindeki hatayı gösterir. Eğitim kaybının azalması, modelin eğitim verileri üzerinde daha iyi performans gösterdiğini gösterir. Ancak, doğrulama kaybının artması, modelin aşırı öğrenme yapabileceğini gösterebilir. Buna göre eğitim setimiz üstünde yorum yapacaksak, eğitim kaybı sürekli azalırken, doğrulama kaybının 3. epoch'tan sonra artmaya başladığı dikkat çekiyor. Bu durum aşırı öğrenme (overfitting) ihtimalini işaret ediyor. Eğitim kaybı, sürekli azalıyor ve modelin eğitim verisini daha iyi öğrenmesini gösteriyor. Bu, beklenen bir durum.

Doğrulama Kaybında ise dalgalı bir seyir izliyor. 1. epoch'ta düşük başlarken, 3. ve 4. epoch'ta artış gösteriyor. Bu, modelin doğrulama verisinde başarısız olmaya başladığını ve genelleştirme problemi yaşadığını gösteriyor.

**metin, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Karışıklık Matrisi**

Karışıklık matrisi, modelin doğru ve yanlış sınıflandırmalarını sayısal olarak özetleyen bir tablodur. Karışıklık matrisine göre;

**\*Doğru Pozitifler (gerçek pozitif olup doğru tahmin edilen):** Model, pozitif yorumları büyük oranda doğru tahmin ediyor.

**\*Doğru Negatifler** : Negatif yorumları da oldukça iyi ayırt edebiliyor.

metin, ekran görüntüsü, diyagram, kare içeren bir resim

Açıklama otomatik olarak oluşturuldu

**ROC Eğrisi ve AUC**

ROC Eğrisi, modelin farklı sınıflandırma eşik değerlerinde performansını görselleştiren bir grafiksel araçtır. AUC ise ROC eğrisinin altında kalan alanı ifade eder. Modelin genel performansını özetler. ROC eğrisine bakıldığında, AUC değeri yaklaşık 0.80 gibi görünüyor. Bu, modelin pozitif ve negatif sınıfları ayırmada makul bir performansa sahip olduğunu gösteriyor. ROC eğrisinin eğimi, düşük yanlış pozitif oranlarında bile modelin doğru tahmin yapabildiğini gösteriyor.

metin, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Prediction Sonuçları ve Analizin Yorumu**

**Eğitim ve Doğrulama Performansı Özet Tablosu**

**Eğitim Doğruluğu ve Kaybı :** İlk epoch'ta eğitim doğruluğu %60,59 iken, 4. epoch sonunda %88,45 seviyesine yükselmiş. Bu, modelin eğitim verisini öğrenme sürecinin başarılı olduğunu gösteriyor. Ancak doğruluk oranındaki bu hızlı artış, modelin aşırı öğrenme **(overfitting)** riski taşıdığını düşündürebilir. Eğitim kaybı, 0.6352'den başlayarak 4. epoch sonunda 0.2874 seviyesine düşmüş. Eğitim kaybındaki azalma, modelin eğitim verisini iyi bir şekilde öğrenmekte olduğunu gösterir.

**Doğrulama Doğruluğu ve Kaybı :** Doğrulama doğruluğu ilk epoch'ta %76,49 olarak başlamış ve 3. epoch'ta %77,66 ile en yüksek seviyeye ulaşmıştır. Ancak sonrasında %76,57'ye düşmüştür. Bu durum, modelin doğrulama verisinde genelleme yapmada zorlandığını ve aşırı öğrenme riski taşıdığını gösterebilir. Doğrulama kaybı, başlangıçta 0.4801 seviyesindeyken, son epoch'ta 0.5387 seviyesine yükselmiştir. Eğitim kaybı azalırken doğrulama kaybının artması, modelin aşırı öğrenme yaptığının göstergesi.

**metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Karışıklık Matrisine bakarsak Yanlış pozitif ve yanlış negatif oranlarının dengelenmesi gerekebilir. Bu, sınıflar arasındaki dengesizlik veya modelin yeterince öğrenemediği durumların bir sonucu olabilir.

ROC eğrisi ve AUC bakarsak, AUC değeri modelin pozitif ve negatif sınıfları ayırma kabiliyetinin makul olduğunu gösteriyor. Ancak daha yüksek bir değer (örneğin 0.90+) daha iyi bir sınıflandırma performansı anlamına gelir. Pozitif ve negatif sınıflar arasındaki ayırım yeterince iyi, ancak iyileştirme yapılabilir. Model, düşük yanlış pozitif oranlarında makul bir doğruluk sağlıyor. Bu da belirli eşik değerlerinde modelimizin işe yarar olabileceğini gösteriyor.

**SONUÇ**

**Test Doğruluğu:** %76.28

**Test Kaybı:** 0.5440

Modelin son test doğruluğu %76,28 olarak ölçülmüş. Bu oran, modelin temel düzeyde kabul edilebilir bir performansa sahip olduğunu ancak iyileştirebileceğini gösterir. Genelleme performansı olarak ise Doğrulama ve test doğruluğu, eğitim doğruluğunun gerisinde kalmıştır. Bu durum, modelin genelleme kapasitesinin sınırlı olduğunu ve doğrulama verisi ile test verisinde daha iyi sonuçlar alabilmesi için düzenlileştirme (regularization - Dropout, L2 regularization gibi yöntemler-) yöntemlerinin uygulanması gerektiğini gösterir.

**Kullanılan Kaynaklar**

Bilgin,M.Makine Öğrenmesi.(2018).Papatya Yayınevi.

Confusion Matrix (Karışıklık Matrisi) (2024 Jan 11).Medium. <https://medium.com/@serapozden922/confusion-matrix-kar%C4%B1%C5%9F%C4%B1kl%C4%B1k-matrisi-62c43b8ad2b0#:~:text=Kar%C4%B1%C5%9F%C4%B1kl%C4%B1k%20matrisi%2C%20bir%20modelin%20performans%C4%B1n%C4%B1,%C5%9Feklinde%20temsil%20edilen%20bir%20ara%C3%A7t%C4%B1r>.

IMDB Dataset of 50K Movie Reviews (2019).Kaggle. <https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>

Kaynak Yazımı Kılavuzu.(?)

<https://halic.edu.tr/tr/s-arastirma/Documents/etikkurul/basvuruformlari/etik-kurul-kaynak-yazim-kilavuzu.pdf>

ROC ve AUC (2020, Jan 12).Medium. <https://medium.com/@gulcanogundur/roc-ve-auc-1fefcfc71a14>

Sentiment Analysis on IMDB Movie Reviews: A Beginner’s Guide (2023, Oct 20).Medium.

<https://medium.com/@AMustafa4983/sentiment-analysis-on-imdb-movie-reviews-a-beginners-guide-d5136ec74e56>

Uzun kısa süreli bellek (2014).Wikipedia. <https://tr.wikipedia.org/wiki/Uzun_k%C4%B1sa_s%C3%BCreli_bellek>

Yapay Sinir Ağları (2021).Wikipedia. <https://tr.wikipedia.org/wiki/Yapay_sinir_a%C4%9Flar%C4%B1>

Yinelemeli Sinir ağı (2021).Wikipedia. <https://tr.wikipedia.org/wiki/Yinelemeli_sinir_a%C4%9F%C4%B1#:~:text=Yinelemeli%20sinir%20a%C4%9F%C4%B1%2C%20d%C3%BC%C4%9F%C3%BCmler%20aras%C4%B1,Recurrent%20neural%20network)%20olarak%20an%C4%B1l%C4%B1r>.

|  |  |
| --- | --- |
| Öğrenci No: | 22410082028 |
| İsim soyisim: | Zahide Nur ŞENSOY |
| Ödev numarası: | 051224K-6 |
| Tarih: | 05.12.2024 |
| Ödev sorusu : | Projede kullanılan Kodları yazınız. |
| **Gerekli kütüphaneleri içe aktaralım;**  #Kütüphaneler.  import pandas as pd  import numpy as np  from sklearn.preprocessing import OneHotEncoder  from sklearn.model\_selection import train\_test\_split  from tensorflow.keras.preprocessing.text import Tokenizer  from tensorflow.keras.callbacks import EarlyStopping  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Embedding, LSTM, Dropout, Dense  **Daha sonra veri setimizi yükleyelim;**  #Veri Setini Yükle.  data = pd.read\_csv("IMDB Dataset.csv") #Veri setinin yolunu buraya yazıyoruz.  **Yüklediğimiz veri setimize göz atmak için ilk 5 satırını ve son 5 satırını okutalım;**  data.head() #İlk 5 satır.    data.tail() #Son 5 satır.    **Data verimiz hakkında bilgi sahibi olmak, kısaca bir özetini görmek için; (Satır sayısı, Her sütunun veri tipi, Her sütunda null (boş) değerlerin sayısı, Toplam sütun sayısı)**  #Veri setimiz hakkında bilgi edinelim.  data.info()  <class 'pandas.core.frame.DataFrame'>  RangeIndex: 50000 entries, 0 to 49999  Data columns (total 2 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 review 50000 non-null object  1 sentiment 50000 non-null object  dtypes: object(2)  memory usage: 781.4+ KB  **Fikir edinmek amacıyla veri setimizde sayısal sütunlar için temel istatistiksel bilgileri (ortalama, standart sapma, min, max, çeyrekler vb.) özetleyelim.Bu sayede genel eğilimleri hızlıca görebileceğiz;**  #Veri setimiz hakkında fikir sahibi olalım.  data.describe()    **Veri setimizde sentiment (duygu) sütunundaki her değerin kaç kez tekrarlandığına bakıyoruz;**  data["sentiment"].value\_counts()    **Veri setimizde eksik değer var mı kontrol ediyoruz;**  data.isna().sum()    **Veri setimizde yinelenen değerleri kontrol ediyoruz;**  data.duplicated().sum()  **418**  **Veri setimiz hakkında biraz bilgi sahibi olduk.Şimdi verilerilerimizi salt hale getirmeliyiz.İlk olarak veri setimizdeki durdurma sözcüklerini kaldıralım;**  #Durdurma sözcüklerini kaldırma.  def remove\_stopwords(text):      stop\_words = stopwords.words('english')      words = text.split()      filtered\_sentence = ''      for word in words:          if word not in stop\_words:              filtered\_sentence = filtered\_sentence + word + ' '      return filtered\_sentence  **İkinci olarak veri setimizdeki özel karakterleri işleyelim;**  #Özel işaretleri işleme.  def normalize\_text(text):      text = text.lower()      #URL'leleri kaldırma.      text = re.sub('https?://\S+|www\.\S+', '', text)      #Kelimeler ve fazla boşlukları kaldırma.      text = re.sub('\\W', ' ', text)      text = re.sub('\n', '', text)      text = re.sub(' +', ' ', text)      text = re.sub('^ ', '', text)      text = re.sub(' $', '', text)      return text  **Üçüncü olarak veri setimizdeki noktalama işaretlerini kaldıralım;**  #Noktalama işaretleri kaldırma.  def remove\_punctuation(text):      table = str.maketrans('','',string.punctuation)      words = text.split()      filtered\_sentence = ''      for word in words:          word = word.translate(table)          filtered\_sentence = filtered\_sentence + word + ' '      return filtered\_sentence  **Dördüncü olarak veri setimizdeki kelimelerimizi köklerine indirgeyelim;**  #Kelimeleri köklerine indirgeme.  def stemming(text):      ps = PorterStemmer()      words = text.split()      filtered\_sentence = ''      for word in words:          word = ps.stem(word)          filtered\_sentence = filtered\_sentence + word + ' '      return filtered\_sentence  **Veri setimizdeki verileri analize geçmeden önce data temiz ve kolay okunabilir bir hale getirmeye çalıştık.Şimdi tüm fonksiyonları bir yerde çağıralım;**  #Tüm fonksiyonlar.  def clean\_text(text):      text = text.lower()      text = text.replace(',',' , ')      text = text.replace('.',' . ')      text = text.replace('/',' / ')      text = text.replace('@',' @ ')      text = text.replace('#',' # ')      text = text.replace('?',' ? ')      text = normalize\_text(text)      text = remove\_punctuation(text)      text = remove\_stopwords(text)      text = stemming(text)      return text  **Şimdi veri setimizi eğitim ve test setlerine böleceğiz.Daha sonra ise, sinir ağlarıyla uyumluluk için tek bir sıcak kodlayıcı kullanarak etiketlerimizi sayısal biçime kodlayacağız;**  #Review = X, Sentiment = Y değişkenlerine atadık.  X = data["review"]  y = data['sentiment']  #Değişkenleri sayısal vektörlere dönüştürme.(Sıcak kodlayıcı yöntemi)  one = OneHotEncoder()  y = one.fit\_transform(np.asarray(y).reshape(-1,1)).toarray()  **Veri setimizden eğitim (train) ve test (test) verilerini ayıralım;**  #Eğitim(train) ve test(test) verisi ayıralım.  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,                                                     y,                                                     test\_size=0.3, #Verilerin %30 test, %70'i eğitim için kullanılacak.                                                     random\_state=42, #Her seferinde aynı sonuçları elde etmeyi sağlar.                                                     shuffle=True) #Veri setinin karıştırılmasını sağlar.  #Eğitim setimiz üzerinde modelimizi eğitip, ardından test setimizi kullanarak modelimizin başarımını değerlendireceğiz.  **Şimdi veri setimizden metin analizi yapabilmek için metinlerimizi anlamlı birimlere ayırmalıyız.Bunun için bazı özellikler tanımlayalım;**  #Özellikler.  vocab\_size = 10000 #Kelime için Maximum sayı.  max\_length = 50 #Dizi için Maximum uzunluk.  trunc\_type = 'post' #Uzun metinleri kesme.  padding\_type = 'post' #Kısa metinlere ekleme.  oov\_tok = '<OOV>' #Olmayan kelimeler için yer tutucu.  **Tokenizerimizi (Metni, sinir ağları için uygun sayısal bir biçime dönüştüren bir araç) tanımlarız;**  #Tokenizer tanımlanır.  tokenizer = Tokenizer(num\_words=vocab\_size, oov\_token=oov\_tok) #Yukarıdaki tanımlanan özelliklerle Tokenizer tanımlama.  tokenizer.fit\_on\_texts(X\_train) #Eğitim metni verilerine uyarlama.  **Veri setimizdeki kullanıma hazır hale getirdiğimiz metin verilerini tokenleştici ile bir sinir ağı tarafından işlenebilecek bir biçime dönüştürmeye hazırlıyoruz;**  #Metinleri tokenizer edip doldurma.  X\_train = tokenizer.texts\_to\_sequences(X\_train) #Eğitim kümesinde metni, daha önce tanımladığımız belirteçleri kullanarak sayısal değere dönüştürür.  X\_test = tokenizer.texts\_to\_sequences(X\_test) #Yukarıdakinin test kümesine uygulanması.  X\_train = pad\_sequences(X\_train, maxlen=max\_length,                           padding=padding\_type,                           truncating=trunc\_type)     #Eğitim kümesinde Tüm dizelerin aynı uzunlukta olmasını sağlar.(Maximum 50)  X\_test = pad\_sequences(X\_test, maxlen=max\_length,                           padding=padding\_type,                           truncating=trunc\_type)     #Yukarıdakinin aynısının Tets kümesine uygulanması.  **Sinir ağı mimarimizi oluştururuz;**  def build\_model():      model = Sequential()      model.add(Embedding(vocab\_size, 64, input\_length=max\_length))      model.add(LSTM(64, return\_sequences=True))      model.add(Dropout(0.2))  #Bırakma düzenlemesi ekle.      model.add(LSTM(32, return\_sequences=True))      model.add(Dense(32, activation='relu'))      model.add(Dropout(0.2))  #Bırakma düzenlemesi ekle.      model.add(LSTM(32))      model.add(Dense(32, activation='relu'))      model.add(Dropout(0.2))  #Bırakma düzenlemesi ekle.      model.add(Dense(2, activation='softmax'))      model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])      early\_stopping = EarlyStopping(monitor='val\_loss', patience=3)  #Erken Durdurma (3 kez izler herhangi bir iyileşme olmazsa durdurur.)      return model, early\_stopping  **Son olarak Model eğitimi yapıyoruz;**  model, early\_stopping = build\_model()  batch\_size = 300  history = model.fit(X\_train,y\_train,epochs=20,batch\_size=batch\_size,validation\_split=.2,callbacks=[early\_stopping])  **Epoch 1/20**  **[1m94/94[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m42s[0m 361ms/step - accuracy: 0.6059 - loss: 0.6352 - val\_accuracy: 0.7649 - val\_loss: 0.4801**  **Epoch 2/20**  **[1m94/94[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m45s[0m 409ms/step - accuracy: 0.8237 - loss: 0.4043 - val\_accuracy: 0.7663 - val\_loss: 0.4833**  **Epoch 3/20**  **[1m94/94[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m44s[0m 438ms/step - accuracy: 0.8462 - loss: 0.3543 - val\_accuracy: 0.7766 - val\_loss: 0.4949**  **Epoch 4/20**  **[1m94/94[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m74s[0m 351ms/step - accuracy: 0.8845 - loss: 0.2874 - val\_accuracy: 0.7657 - val\_loss: 0.5387**  **Şimdi, genelleme yetenekleri hakkında kapsamlı bir anlayış elde etmek için test bölünmesinde nasıl performans gösterdiğini inceleyelim;**  score = model.evaluate(np.asarray(X\_test),np.asarray(y\_test))  **[1m469/469[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m13s[0m 28ms/step - accuracy: 0.7628 - loss: 0.5440**  **Sonuç olarak, modelimiz %76,28 doğruluk gösteriyor.** | |

MAKİNE ÖĞRENMESİ

Ödev Kısmı

Zahide Nur ŞENSOY

22410082028

2024-2025 Eğitim Öğretim Yılı

|  |  |
| --- | --- |
| Öğrenci No: | 22410082028 |
| İsim soyisim: | Zahide Nur ŞENSOY |
| Ödev numarası: | 231024K-1 |
| Tarih: | 23.10.2024 |
| Ödev sorusu : | Ekte verilen m1.txt ve m2.txt dosyasının textlerini karşılaştırınız. Benzerlik oranlarını bulunuz. |
| str1 = open('m1.txt', 'r').read()  str2 = open('m2.txt', 'r').read()  #Yukardaki kodlarda m1 ve m2 metinlerini str1 ve str2 olarak atıyoruz, okutuyoruz.  str1 = str1.replace(".","")  str1 = str1.replace(",","")  str1 = str1.replace("'","")  str2 = str2.replace(".","")  str2 = str2.replace(",","")  str2 = str2.replace("'","")  #Yukarıdaki kodlarda atadığımız metinlerin içerisindeki nokta, virgül ve üstten tırnak işaretlerini kaldırıyoruz.  durma = ["ve","veya","ile","çünkü","birkaç","böyle","falan","herkes","hiçbiri",            "gibi","hangi","kim","şu","şey","yada","zira","zaten","yine","neyse",            "ama","ancak","asla","az","bazı","bazısı","belki","birçok","çok","çoğu",            "daha","değil","diğer","elbette","hiç","ise","kendi","kime","niye","önce",           "ötürü","rağmen","şunu","şunlar","tümü","veya","yoksa","zaten","zira"]  for i in durma:      str1 = str1.replace(i,"")      str2 = str2.replace(i, "")  #Yukarıda Bağlaç ve yardımcı kelimeleri içeren bir liste oluşturuyoruz. Bu listedeki kelimeler metinden çıkarılıyor döngü ile kelimeler metinlerden temizleniyor.  l1 = list(str1.split(" "))  print(l1)  print(len(l1))  l2 = list(str2.split(" "))  print(l2)  print(len(l2))  #Yukarıda kodlarda iki metin de boşluklarından bölünerek ayrı ayrı kelime olarak ayrılıyor. Bu kelimeler l1 ve l2 listesine atayıp, yazdırıyoruz.  s1 = set(l1)  print(s1)  print(len(s1))  s2 = set(l2)  print(s2)  print(len(s2))  #Yukarıdaki kodlarda l1 ve l2 listesindeki tekrar eden kelimeleri kaldırıyoruz.(Sadece bir kez bulunuyor.)l1 ve l2 boyutlarını yazdırıyoruz.  st = set.union(s1,s2)  print(st)  print(len(st))  #Yukarıdaki kodlarda 2 metnin kümelerini birleştiriyoruz. Birleştikten sonraki küme boyutunu yazdırıyoruz.  ts = len(s1)+len(s2)  print("İki metin için tekil sözcük sayısı = ", ts)  tss = len(st)  print("Birleşim sonrası tekil sözcük sayısı = ",tss)  fark = ts - tss  print("Fark = ", fark)  benzeme = (fark\*100)/tss  print("Benzeme oranı = %",benzeme)  #Yukarıda 2 metindeki toplam tekil kelime sayısı ile birleşim sonrası tekil kelime sayısı karşılaştırdık.2 metindeki benzer kelimelere bakıp, Benzeme oranını hesaplıyoruz. | |

|  |  |
| --- | --- |
| Öğrenci No: | 22410082028 |
| İsim soyisim: | Zahide Nur ŞENSOY |
| Ödev numarası: | 231024K-2 |
| Tarih: | 23.10.2024 |
| Ödev sorusu : | Adınızı ve soyadınızı Shannon Entropy ile ikili bitler halinde kodlayan programı yazınız. |
| import math  s = "Zahide Nur Şensoy"  s = s.replace(" ", "")  liste = []  for i in s:      liste.append(i)  print("Karakter Listesi:", liste)  k = set(liste)  print("Benzersiz Karakterler:", k)  d = {}  for i in k:      adet = liste.count(i)      oran = adet / len(liste)      d.update({i: oran})  print("Frekans Oranları:", d)  shannon = 0  for i in d:      v = d[i]      shannon += v \* math.log(v, 2)  shannon \*= -1  print("Shannon Entropisi:", shannon)  bs = math.ceil(shannon)  print("Bit sayısı =", bs)  dk = list(k)  b = []  for i in range(int(math.pow(2, bs))):      a = bin(i)[2:]      b.append(a)  dk.sort()  print("Sıralı Benzersiz Karakterler:", dk)  print("İkili Kodlar (Başlangıç):", b)  for i in range(len(b)):      for j in range(bs - len(b[i])):          b[i] = "0" + b[i]  print("İkili Kodlar (Doldurulmuş):", b)  coded = ""  for i in liste:      coded += b[dk.index(i)] + "-"  print("Kodlanmış Metin:", coded.strip("-")) | |

metin, el yazısı, doküman, belge içeren bir resim

Açıklama otomatik olarak oluşturuldu

|  |  |
| --- | --- |
| Öğrenci No: | 22410082028 |
| İsim soyisim: | Zahide Nur ŞENSOY |
| Ödev numarası: | 291024K-3 |
| Tarih: | 29.10.2024 |
| Ödev sorusu : | Covid-19 hastalağına yakalananların semptomlarını gösteren tablo verilmiştir. Covid-19 için en önemli değişken hangisidir? Shannon entropi ile hesaplayınız.  A4 kağıt üzerine tabloyu çizerek değişken entropilerini tek tek hesaplayınız ve en önemli değişkeni bulunuz. |
|  | |

|  |  |
| --- | --- |
| Öğrenci No: | 22410082028 |
| İsim soyisim: | Zahide Nur ŞENSOY |
| Ödev numarası: | 021124K-4 |
| Tarih: | 02.11.2024 |
| Ödev sorusu : | Diyabet hastalığı için en az 40 satırlık veriyi train için, eğitime alınmamış en az 10 satırı test için kullanınız ve sonuçları kod dosyasına açıklama satırı ile not ediniz. |
| **İlk olarak gerekli kütüphaneler içe aktarılır. Colab'da bunlar tanımlı olduğu için bu kısmı atlıyorum.**  import math  from sklearn.neighbors import KNeighborsClassifier  import pandas as pd  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import accuracy\_score, classification\_report  **İkinci olarak diyabet verilerimi tanımlıyorum. -40 eğitim verisi ve 10 test verisi- (Diabetes kısmında 1=Evet, 0=Hayır olarak yazdım.)**  data = {      'Glucose': [40, 40, 45, 45, 40, 45, 40, 30, 65, 45,                  35, 45 ,55, 40, 35, 50, 40, 40, 40, 40,                  40, 40, 40, 45, 40, 60, 40, 40, 45, 55,                  45, 50, 45, 40, 45, 45, 45, 40, 45, 45,                  40, 40, 45, 45, 45, 45, 40, 40, 45, 45],      'BloodPressure': [85, 92, 63, 80, 73, 82, 85, 63, 65, 82,                        73, 90, 68, 93, 80, 70, 73, 67, 75, 80,                        72, 88, 78, 98, 88, 67, 85, 88, 78, 73,                        77, 68, 77, 85, 70, 72, 90, 65, 88, 88,                        68, 73, 88, 78, 78, 85, 83, 63, 73, 90],      'Diabetes': ["hayır", "hayır", "evet", "hayır", "evet", "hayır", "hayır", "evet", "evet", "hayır",                   "evet", "hayır", "evet", "hayır", "evet", "evet", "evet", "evet", "evet", "evet",                   "evet", "hayır", "evet", "hayır", "hayır", "evet", "hayır", "hayır", "hayır", "evet",                   "evet", "evet", "evet", "hayır", "hayır", "evet", "hayır", "evet", "hayır", "evet",                   "evet", "hayır", "evet", "hayır", "evet", "hayır", "evet", "evet", "hayır", "hayır"]  }  **DataFrame'e dönüştürüyoruz.**  df = pd.DataFrame(data)  **Özellikler (x) ve hedef değişkeni (y) olarak ayıklıyoruz.**  x = df[['Glucose', 'BloodPressure']] #Özellikler  y = df['Diabetes'] #Hedef değişken  **Girdiğimiz verileri eğitim ve test verisi olarak ayırıyoruz. (40 eğitim, 10 test)**  x\_train = x[:40] #İlk 40 satır eğitim verisi  y\_train = y[:40] #İlk 40 satır hedef değişken  x\_test = x[40:]  #Sonraki 10 satır test verisi  y\_test = y[40:]  #Sonraki 10 satır hedef değişken  **KNN modelini tanımlıyoruz ve eğitiyoruz.**  knn = KNeighborsClassifier(n\_neighbors=3)  knn.fit(x\_train, y\_train)  **Test seti ile tahmin yapıyoruz.**  y\_pred = knn.predict(x\_test)  **Sonuçları değerlendiriyoruz.**  accuracy = accuracy\_score(y\_test, y\_pred)  print("Doğruluk Oranı:", accuracy)  print("Sınıflandırma Raporu:\n", classification\_report(y\_test, y\_pred))  **Doğruluk Oranı: 0.5**  **Sınıflandırma Raporu:**  **precision recall f1-score support**  **evet 0.50 0.60 0.55 5**  **hayır 0.50 0.40 0.44 5**  **accuracy 0.50 10**  **macro avg 0.50 0.50 0.49 10**  **weighted avg 0.50 0.50 0.49 10**  **Sonuçlar;**  **metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim  Açıklama otomatik olarak oluşturuldu**  **Bu kodlarda, Glucose (glikoz) ve BloodPressure (kan basıncı) özelliklerine göre diabet olup olmadığını tahmin etmek için**  **KNN [K-Nearest Neighbors] (K En Yakın Komşu) Algoritması kullanarak bir makine öğrenmesi modeli oluşturdum.Test verisi üzerinde modelin doğruluğunu değerlendirdim.** | |

|  |  |
| --- | --- |
| Öğrenci No: | 22410082028 |
| İsim soyisim: | Zahide Nur ŞENSOY |
| Ödev numarası: | 021124K-5 |
| Tarih: | 02.11.2024 |
| Ödev sorusu : | Veri setini Naive Bayes yöntemi ile sınıflandırınız, ardından listede olmayan farklı durumlar için predictionlar deneyiniz. |
| **\*\*A:\*\*** Accident  \* A(Yes) = 5/10  \* A(No) = 5/10  ---  \* A(Clear|Yes) = 2/5  \* A(Clear|No) = 2/5  \* A(Avarage|Yes) = 1/5  \* A(Avarage|No) = 1/5  \* A(Normal|Yes) = 1/5  \* A(Normal|No) = 2/5  \* A(No|Yes) = 2/5  \* A(Yes|Yes) = 3/5  ---  \* Oranları Çarpalım: "Yes" ;  \* P(X|Accident=Yes) = 2/5 \* 1/5 \* 1/5 \* 2/5 \* 5/10 = 0,0032  \*Oranları Çarpalım: "No" ;  \* P(X|Accident=No) = 2/5 \* 1/5 \* 2/5 \* 3/5 \* 5/10 = 0,0096  metin, yazı tipi, ekran görüntüsü, beyaz içeren bir resim  Açıklama otomatik olarak oluşturuldu  import numpy as np  from sklearn.naive\_bayes import GaussianNB  # Features  X = np.array([      [0, 0, 2, 0],      [1, 1, 1, 1],      [2, 0, 0, 0],      [2, 2, 0, 1],      [1, 2, 1, 0],      [0, 1, 0, 0],      [0, 2, 1, 0],      [1, 0, 2, 1],      [2, 2, 2, 1],      [2, 0, 2, 1]  ]) # Label  Y = np.array([1, 1, 0, 1, 0, 0, 0, 1, 0, 1]) # Label  model = GaussianNB()  model.fit(X, Y)  **Parantez içindeki sayıları istediğiniz gibi değiştirerek istediğiniz verilerin sonucuna bakabilirsiniz.**  print(model.predict([[2, 1, 2, 0]]))  **\*\*ÖRNEK1:\*\* [Clear(2), Avarage(1), Normal(1), No(0) => Sonuç: N0 (0)] Verilerinin sonucu asağıdaki Görseldedir.**    **\*\*ÖRNEK2:\*\* [Clear(2), Avarage(1), High(2), No(0) =] Verilerinin Sonucu asağıdaki görseldedir.** | |