# Makine Öğrenmesi

Hakan DİNÇTÜRK
Kocaeli Üniversitesi,
Bilişim Sistemleri Mühendisliği
Kocaeli, İzmit
hakan.dincturkk@gmail.com

## I. GIRIŞ

Kuş türlerinin (arı kuşu, bülbül, ibibik, şahin, güvercin) sınıflandırılması.

## II. PROJE IŞLEYIŞI

a) Konu belirlendi,
b) Proje isterleri net bir şekilde anlaşıldı,
c)Araştırmalar yapıldı,
d) Veriler toplandı ve ön işleme yapıldı
e) CNN, YSA ve Decision Tree modelleri uygulandı.
f) Sayısal veriler çıkartıldı.
g) Rapor hazırlandı,
h) Proje Tamamlandı.

## III. KULLANILAN TEKNOLOJILER

- *Jupyter NoteBook*


Jupyter Notebook, çeşitli programlama dilleri için etkileşimli bir ortam sağlayan açık kaynak kodlu bir programdır. Adıyla müsemma bir defter gibidir. Tarif etmem gerekirse word içerisinde python kodları çalıştırabildiğinizi düşünün. İçerisinde markdown destekli notlar alabildiğiniz gibi bir alt satırda makine öğrenmesi kodlarını çalıştırabilirsiniz.

- *Python*

Python, nesne yönelimli, yorumlamalı, birimsel ve etkileşimli yüksek seviyeli bir programlama dilidir. Girintilere dayalı basit söz dizimi, dilin öğrenilmesini ve akılda kalmasını kolaylaştırır.

## IV. KULLANILAN YÖNTEMLER

- *Veri Seti Oluşturma*

Youtube üzerinden indirilecek ses dosyalarının linkleri toplandı ve JSON formatında depolandı. Verileri JSON türünde tutmamızın temel nedeni basit veri yapılarında JSON formatının çok başarılı işler çıkarmasıdır.

JSON dosyalarını okumak ve işlem yapmak içinse 'json' kütüphanesini projemize dâhil etmemiz gerek.

Youtube'dan veri indirmek için youtube_dl kütüphanesini kullandım. Bu kütüphane Youtube'dan istediğimiz videoyu istediğimiz formatta indirmemizi sağlar.

- *Veri Çekme*

```
def downloadAudio(link, folderName, counter):
    video_url = link
    video_info = YoutubeDL().extract_info(url = video_url,download=False)
    filename = "data2/" + folderName + "/" + folderName + str(counter) + ".wav"
    options={ 'format':'bestaudio/best', 'keepvideo':False, 'outttmpl':filename, }
    with YoutubeDL(options) as ydl:
        ydl.download([video_info['webpage_url']])
    print("Download complete... {}".format(filename))
```

Yukarıda ekran görüntüsü olan kodda verdiğimiz linklerde ki videoları .wav formatında istediğimiz isimde kaydedebiliyoruz.

- *JSON'dan veri okumak*

```
f = open('audioData.json')
data = json.load(f)
counter = 1
for i in data:
    print(i)
    counter = 1
    for j in data[i]:
        print(j['link'], str(counter))
        try:
            downloadAudio(j['link'], i, counter)
        except:
            print('HATA!!:' ,j['link'].split('=')[1], 'indirilemedi ', counter)
        counter= counter + 1
# Closing file
f.close()
```

*'audioData.json'* dosyasından verileri okuyoruz ve hazırladığımız JSON dosyasında ki linklerden fonksiyonumuza linkleri gönderiyoruz.

Verileri İşlemek

- Verileri işleme için kullandığım kütüphaneler

```
from pydub import AudioSegment
import librosa
import math
import pandas as pd
import numpy as np
import os
```

- Ses dosyalarını işlemek için kullandığımız fonksiyon

```
def features_extractor(file):
    audio, sample_rate = librosa.load(file, res_type='kaiser_fast')
    mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
    mfccs_scaled_features = np.mean(mfccs_features.T,axis=0)

    return mfccs_scaled_features
```

- Ses dosyalarını kaydetmek için kullandığımız fonksiyon

```
def auidoFileData(fileName):
    print(fileName)
    path = 'data/'+fileName+'/splitted/'
    audioNames = os.listdir(path)
    folderFileCount = len(audioNames)
    for i in range(0, folderFileCount - 1):
        print(path+audioNames[i])
        final_class_labels=fileName
        data=features_extractor(path+audioNames[i])
        print(i, ' - ', path+audioNames[i], ' - ', data)
        extracted_features.append([data,final_class_labels])
```

Ses dosyalarını dinamik olarak aldığımız için OS kütüphanesini kullandık.

- Fonksiyona ses dosyalarını göndermek için kullandığımız kod kısmı

```
path = 'data2/'
#otherPath = path + 'splittedAllAudioFiles/'
fileNames = os.listdir(path)
counter=1
for i in range(0, len(fileNames)-1):
    if fileNames[i] != 'diger':
        if fileNames[i] != 'splittedAllAudioFiles':
            counter = 1
            audioPath = path+fileNames[i]
            audioFileNames = os.listdir(audioPath)
            for j in range(0, len(audioFileNames)-1):
                if audioFileNames[j] != 'splitted':
                    #print(audioPath+'/splitted/'+audioFileNames[j])

                    audioFileBadName = audioFileNames[j].split('.')[0]
                    size = len(audioFileBadName)
                    if fileNames[i] != 'beeBird':
                        if j > 9:
                            audioFile = audioFileBadName[:size - 2]
                        else:
                            audioFile = audioFileBadName[:size - 1]
                    else :
                        audioFile = audioFileBadName[:size - 1]
                    song = AudioSegment.from_wav(audioPath+'/'+audioFileBadName+".wav")
                    duration = song.duration_seconds
                    for k in range(int(duration/5)):
                        splitted_audio = song[(5*k)*1000:5*(k+1)*1000]
                        splitted_audio.export(audioPath+'/splitted/'+audioFile+str(counter)+'.wav', format="wav")
                        splitted_audio.export(otherPath+'/'+audioFile+str(counter)+'.wav', format="wav")
                        #print(audioPath+'/splitted/'+audioFile+str(counter)+'.wav',)

                        counter=counter+1
            print('-----------------')
            auidoFileData(fileNames[i])
            print('-----------------')
```

- CNN

Convonlutional Neural Networ, derin öğrenmenin iki temel algoritmasından birisidir. Resim sınıflandırma, nesne tanıma, image ve captioning problemlerinde daha çok tercih edilir.

```
xtrain, xtest, ytrain, ytest = train_test_split(newX, y, test_size=0.1, random_state=0)
```

```
y.shape
```

```
(***
```

```
num_labels=9
```

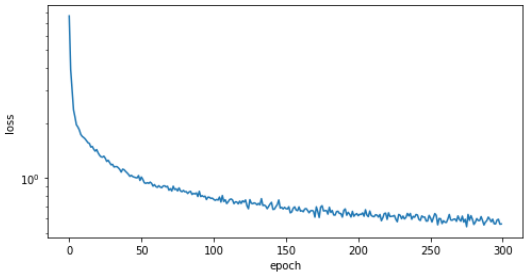
```
model=Sequential()
model.add(Dense(125, input_shape=(40, )))
model.add(Activation('relu'))
model.add(Dropout(0.5))

#2. gizli
model.add(Dense(250))
model.add(Activation('relu'))
model.add(Dropout(0.5))

#3. gizli
model.add(Dense(125))
model.add(Activation('relu'))
model.add(Dropout(0.5))

#output
model.add(Dense(num_labels))
model.add(Activation('softmax'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 125)               5125

 activation (Activation)     (None, 125)               0

 dropout (Dropout)           (None, 125)               0

 dense_1 (Dense)             (None, 250)               31500

 activation_1 (Activation)   (None, 250)               0

 dropout_1 (Dropout)         (None, 250)               0

 dense_2 (Dense)             (None, 125)               31375

 activation_2 (Activation)   (None, 125)               0
```

```
# Plot loss
plt.figure(figsize=(8,4))
plt.semilogy(history.history['loss'])
plt.xlabel('epoch'); plt.ylabel('loss')
```

```
Text(0, 0.5, 'loss')
```




- YSA

Yapay sinir ağları başlıca teşhis, sınıflandırma, tahmin, kontrol, veri ilişkilendirme, veri filtreleme, yorumlama gibi alanlarda kullanılmaktadır.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(newX, label, test_size = 0.1, random_state=1)
print(x_train)
```

```
[[3.8265237 1.8915022 4.6298759 ... 5.6298345 4.3177918 1.4206719]
 [3.2413019 8.0953789 1.8785106 ... 3.2329597 5.7430787 3.266638 ]
 [3.9254413 1.7591204 3.7156158 ... 2.2072949 2.5309236 1.6333632]
 ...
 [4.684386  7.4814568 1.5321209 ... 6.0879362 1.2897229 2.9496333]
 [3.2620801 7.2486328 9.2928375 ... 4.5290666 1.7248689 3.001303 ]
 [3.0348978 1.8658041 8.109832  ... 2.1282499 3.1664484 1.2965281]]
```

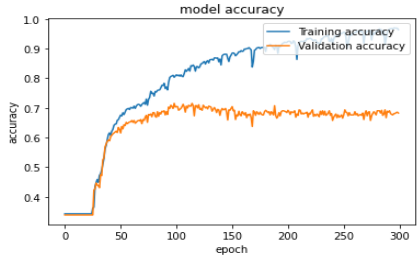
```
#input_dim = features, units= yapay sinir hücresi, init=yapay sinir hücresi içerisindeki ağırlıkla
# BinaryCrossentropy: Gerçek etiketler ile tahmin edilen etiketler arasındaki çapraz entropi kaybı

label.shape
```

```
(2153, 1)
```

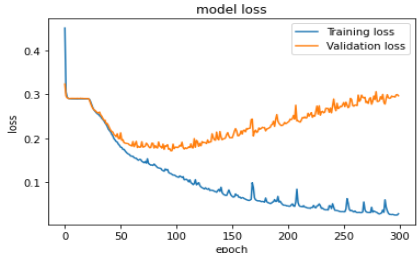
```
model = Sequential()

# 1. Katman - Input
model.add(Dense(40,input_dim=40,kernel_initializer='uniform',activation='relu'))

# 2. Katman
model.add(Dense(36,kernel_initializer='uniform',activation='relu'))

# 3. Katman
model.add(Dense(20,kernel_initializer='uniform',activation='sigmoid'))

# 4. Katman - Output
model.add(Dense(9,kernel_initializer='uniform',activation='sigmoid'))
```

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

```
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Training accuracy', 'Validation accuracy'], loc='upper right
plt.show()
```




```
]: # loss history
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['Training loss', 'Validation loss'], loc='upper right')
plt.show()
```




- Confusion Matrix

Özellikle istatistiksel sınıflandırma probleminde, bir hata matrisi olarakda bilinen bir karışıklık matrisi, tipik olarak denetimli bir öğrenme algoritması olan bir algoritmanın performansını görselleştirmesine izin veren özel bir tablo düzenidir.

```
from sklearn.metrics import confusion_matrix
confmat = confusion_matrix(y_test,result)
print(confmat)
```

```
[[17  0  0  1  2  1  3  0  7]
 [ 2  2  2  1  0  0  1  0  0]
 [ 5  0  2  0  0  0  1  0  1]
 [ 2  0  0  1  0  0  1  0  0]
 [ 3  0  0  0  2  0  0  0  0]
 [ 1  1  1  1  0  1  1  0  0]
 [ 5  1  1  1  0  0 53  1 20]
 [ 2  0  1  0  0  0  8  2  1]
 [ 3  1  0  1  2  0 12  1 37]]
```

Yaşanılan Sorunlar:

Bu aşamaya kadar karşılaştığım zorlukların en başında kaliteli ses verileri toplamaktı. Olması gereken gibi bir ses verisi buluyorsun ama süresi kısa oluyordu, süresi uzun olanlarda ya arkada dip ses, ya insan sesi gibi etkenler oluyordu.

Karşılaştığım bir diğer zorluk ise python kullanımında oldu, pythonda az çok syntax bilgim vardı fakat bu proje için yeterli değildi. İşimi çok kolaylaştıracak ama bilmediğim fonksiyonlar vardı. Özellikle bu fonksiyonları aramak için uzunca bir vakit harcadım.

Bir diğer zorluk ise ses verisinin spektrumlarını çıkarmak oldu. İlk başlarda sadece ses verisini almamın yeterli olduğunu düşünüyordum daha sonra araştırma yapmaya devam ettikçe yukarıda da belirttiğim gibi makine öğrenmesinde kullanmak için mfcc fonksiyonunda tekrar işlem yapmak gerekiyormuş. Bunu yaparken tamamen dinamik bir şekilde yapılması zorlu bir işlemdi.

Bir diğer zorluk ise youtube_dl kütüphanesi ile indirme yaptığımız için işlem sırasında konsolu sürekli canlı tutmak zorundaydım. Ve indirme işlemi çok yavaş ilerliyor, kullandığım internet 100mbit olmasına rağmen 80kbit ile indiriyor.

Bir diğer sorun ise ses işlemeye uygun bir ses işleme algoritması bulmaktaydı. Diğer modellere için geliştirilmiş algoritmaları ses işleme için kullanmaya çalıştım. Bu da algoritmanın amacına uygun olmadığı için yeteri kadar doğru bir sonuç vermedi. Yani doğruluk yüzdesi düşüktü.

Sorunlardan diğeri ise benim kullandığım veri setini oluştururken genelde kullanılan veri setinden farklı bir yöntemle oluşturdum bu yüzden bu veri setini başka bir dosyada kullanmak istediğimde sayısal verilerde problem çıkartıyor. Bunu bütün verileri tek tek float değerine çevirerek çözdüm.

KAYNAKÇA


[1]. https://intellipaat.com/community/65014/download-only-audio-from-youtube-video-using-youtube-dl-in-python-script
[2]. https://www.bogotobogo.com/VideoStreaming/YouTube/youtube-dl-embedding.php
[3]. https://github.com/ytdl-org/youtube-dl
[4]. https://itsfoss.com/youtube-dl-audio-only/
[5]. https://campus.datacamp.com/courses/spoken-language-processing-in-python/manipulating-audio-files-with-pydub?ex=2#:~:text=To%20import%20an%20audio%20file,PyDub%20works%20with%20.
[6]. https://www.reddit.com/r/learnpython/comments/j9jhwn/pydub_cant_find_file/
[7]. https://stackoverflow.com/questions/68247327/cannot-find-a-solution-to-winerror-2-file-not-found-by-audiosegment-from-mp3