BC

Medikal Görüntü Sınıflandırması

Veri Madenciliği ve Analizi

Danışmanlar:

İÇİNDEKİLER

1- Giriş

Proje Amacı;

Hastalar aslen akciğer kanseridir.Ancak bu hastalık, hastaların beynine metastaz yapmıştır.Hastaların beyin MR verisetleri vardır.Bu verisetten hastaların akciğerindeki kanserinin küçük hücreli mi büyük hücreli mi olup olmadığını belirlemektir.

Veri Seti[1] Tanıtımı ve Yapılacak İşlemler:

- ❖ 925 tane MR görüntüleri vardır.
- Verisetteki orijinal resimler ; beyne metastaz yapmış tümörlere sahip kişilerin MR görüntülerdir. Bu hastaların sahip olduğu akciğer kanserinin iki sınıfı var; küçük hücreli ve büyük hücrelidir. Bu ilişkiden yola çıkılarak doktorlar tarafından f beyin MR resimleri üzerinden tümörler işaretlenerek maske veri setleri oluşturuldu.
- Orijinal görüntü ile maske görüntüleri üst üste gelecek şekilde yeni dataset oluşturuldu. Bu verisette '.csv' formatındaki dosyaya her pikseli temsil edecek şekilde 0 ila 255 sayıları arasında rakamsal ifadelere dönüştürülmüştür.
- Verisette kişilerin en belirgin tümörü gözüken 5 dilimi alınmıştır. Ancak kişiler 1 den fazla tümörü varsa 5 katı şeklinde görüntü artıyor.(2 tümör var ise 10 dilimi vardır)

'.csv' formatındaki verisetinin bilgileri:

- ➢ Dosyanın ilk sutun kişilerin numarasını temsil ediyor.
- ➢ 2. sutun görüntünün kaçıncı dilim oldugu temsil ediyor.
- ➢ 3.sutun sınıflandırma bilgilerini içeren sutunudur.Küçük hücreli=0 ve büyük hücreli=1 olduğunu temsil ediyor.
- ➢ 4. 'den sonuncuya kadarki sutunlar; kişilere ait tek bir dilime ait görüntülerin 224x224 lük araye dönüştürülüp daha sonra resminin 50176x1 arrayine dönüştürülmesini temsil ediyor.

- ❖ Verisetinde sınıf dengesizliği vardır. 600 tane büyük hücreli kanser hastası(sınıf=1), 325 tane küçük hücreli kanser hastası(sınıf=0) vardır.

Veriler üzerinden 2 farklı özellik çıkarımları yapıldı.Özellik çıkarımların yapılmasının sebebi ise resimdeki aradığımız bilgiler(pozitif) resim üzerindeki diğer aramadığımız bilgilerden(negatif) istatistiksel farklı olacaktır.Programa tanıtılmasında yardımcı olacaktır.

2 farklı veriseti üzerinde incelemeler yapılmıştır.Bu datasetler programa tanıtarak 9 farklı sınıflandırma ve daha sonra öğrenmesi test edilirken şu soruların cevapları aranacaktır:

1. Sınıf dengesizliği problemi orijinal ve diğer özellik çıkarımlarının sonuçlarını nasıl etkiler?Özellik çıkarımları performansı etkiler mi?
2. Sınıf dengesizliği hangi algoritmaları daha çok etkiler?Hangileri az etkilenir?
3. Aynı bilgilere sahip veriseti üzerinden çıkarılan özellik verisetlerinde aynı sınıflandırıcılar etkili olabilir mi?Ya da özellik çıkarımı türüne bağlı olarak farklı sınıflandırıcı algoritmaları mı kullanılmalı?

4. Hangi durumda hangi sınıflandırıcı kullanılmalı?
5. Sınıflandırma algoritmaları üzerinden torbalama ve güçlendirme teknikleri her zaman iyi sonuç verir mi? Belirgin iyileştirme yapabilirler mi? Hangi güçlendirme tekniği daha güçlüdür?
6. Sonuç değerlendirmede hangi değerlendirmeler daha doğru sonuç verir?

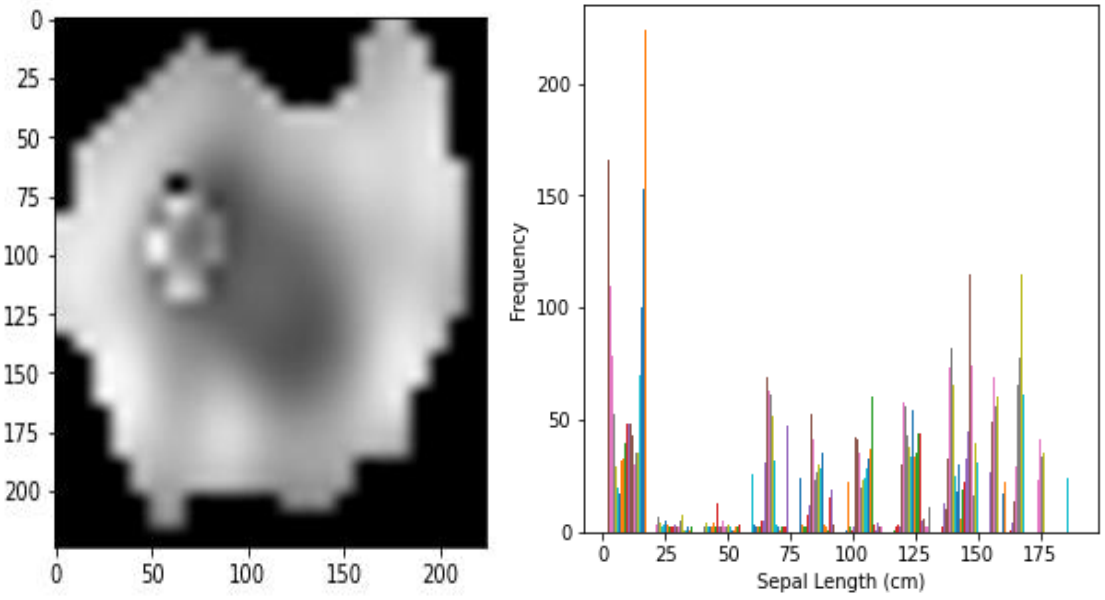
VeriSetini Tanıma:

Kişi Numarası girilerek dataset tanıtılmış ve hastaya ait resimler gösterilmiştir

```
6   """
7   import pandas
8   import numpy as np
9   from matplotlib import pyplot as plt
10  import time
11
12  adres='C:/Users/FbMmm/Downloads/khakDataDogru.csv'
13  x=pandas.read_csv(adres,header=None)
14  goruntu_matrisi=[]
15  goruntu_matrisi=x.iloc[:,3:50179]#x.drop([0,1,2],axis=1)
16  goruntu_matrisi=goruntu_matrisi/255.0
17  kisi=pandas.read_csv(adres,header=None,usecols=[0])
18  dilim_sayısı=pandas.read_csv(adres,header=None,usecols=[1])
19  sinif=pandas.read_csv(adres,header=None,usecols=[2])
20
21  #BILGI ALMA BOLUMU
22  j=int(input('Lutfen kisi numrası giriniz:'))
23
24
25  y=[]
26  #j.ci kişi kaç tane goruntu var
27  h=kisi[kisi[0]==j].size
28  s=int(h/5)
29  #hangi indexlerde
30  b=kisi[kisi[0]==j].index
31  g=int(b[3])
32  #indexlerden birincisi
33  a=int(np.float64(b[0]).item())
34  sınıf=int(sinif.iloc[a])
35  if sınıf==0:
36      l='Bunlar kücük hücreli  kanserlerdir'
37  else:
38      l='Bunlar büyük hücreli kanserlerdir'
39
40
```

```
#BILGILERİ YAZDIRMA

print(str(j)+' kişisi '+str(h)+'tane olan',end=" ")
for g in range(0,h):
    print(str(int(b[g])),end=". ")
print(' ci görüntü bölütlemelerinde görülen '+str(s)+' tane tümöre sahiptir.',end=" ")
print(l)
print('20 saniye sonra goruntuler gelecek')
time.sleep(20)

#TUM BOLUTLEMELERİ GORUNTULEME
for k in range (0,h):
    a=int(np.float64(b[k]).item())
    goruntu_matrisi2=np.resize(goruntu_matrisi.iloc[a,:],(224,224))
    y.append(goruntu_matrisi2)

y=np.array(y)

def display(im3d, cmap="gray", step=1):
    _, axes = plt.subplots(nrows=5, ncols=s, figsize=(16, 14))

    vmin = im3d.min()
    vmax = im3d.max()

    for ax, image in zip(axes.flatten(), im3d[::step]):
        ax.imshow(image, cmap=cmap, vmin=vmin, vmax=vmax)
        ax.set_xticks([])
        ax.set_yticks([])


display(y)
```

Orjinal Verisetinin ilk resmi ve histogramı(plt.hist):



Orjinal Datasetinin bilgileri;

Minimum değeri(np.min): 0

Max.değeri(np.max): 1192

Ortalama değeri(np.mean): 117.65824113175675

Ortanca değeri(np.median): 133.0

Standart sapması(np.std): 93.14906717239629

2.satırının convolusyonu: 4010.79447932

2.satırının mod değeri: 12948

Verisetin 2. Ve sonuncu resimlerin korelasyon matrisi: (farklı kişilere ait resimlerdir)

[[1. 0.3480866]

[0.3480866 1.]]

Verisetin 2. Ve 3. resimlerin korelasyon matrisi:(aynı kişiye ait resimlerdir)

[[1. 0.93572232]

[0.93572232 1.]]

2-Önişleme

2.1.Özellik Çıkarımı

2.1.1. Odaklı Gradyanların Histogramı (HOG)

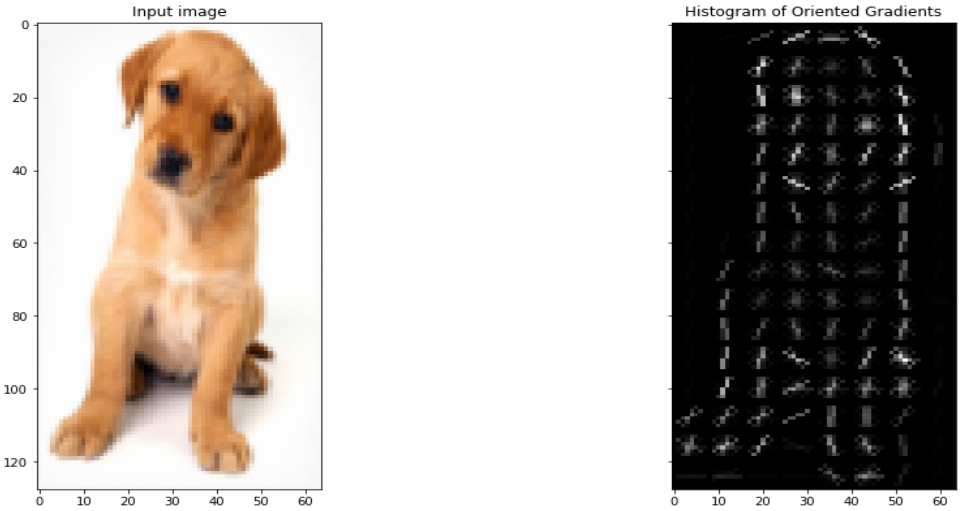
HOG, genellikle görüntü verilerinden özellikleri çıkarmak için kullanılan bir özellik tanımlayıcıdır. Nesne tespiti için bilgisayarla görme görevlerinde yaygın olarak kullanılır.

HOG tanımlayıcısı, bir nesnenin yapısına veya şekline odaklanır.

Bir bölge kordinaatlarıyla seçilirBu Her pikselin gradyanına ve yönüne (veya büyüklüğü ve yönünü) odaklanır., tüm görüntünün daha küçük bölgelere ayrıldığı ve her bölge için gradyanların ve yönelimin hesaplandığı anlamına gelir.HOG, seçili bölgelerin her bir pikseli için ayrı ayrı bir Histogram oluşturacaktır. Histogramlar, piksel değerlerinin gradyanları ve odakları kullanılarak oluşturulur,(Şekil 1 deki gibi)

Programlama Formulasyonu:

1- Verilerin boyut arraylerini 8'e yada 16'ya bölünebilecek sekilde ayarlama
2- Görüntüdeki her piksel için gradyanı hesaplama (**Gradyanlar, x ve y yönlerindeki küçük değişikliklerdir.)**
3- Gradyanları kullanarak her piksel değeri için büyüklük ve yön belirleme
4- Secilen m×m hücrelerde Gradyanların Histogramını hesaplama
5- Hücrelerde gradyanları normalleştirme
6- Görüntü için HOG özellikleri oluşturma(Her piksel için yapılan hesaplamaları resim üzerinde birleştirme)



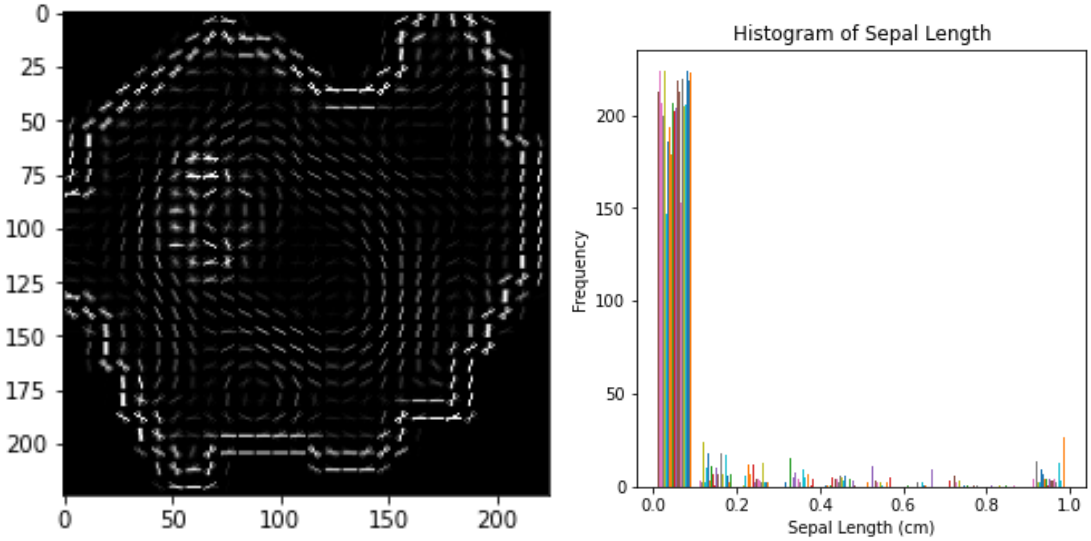
Şekil 1:Orjinal Resim ve Histogramı oluşturulmuş Resim

Python ile HOG FEATURE EXTRACTION

Yeni dataset oluşturuldu.Orjinal Datasetdeki gibi sınıf,kişi ve dilim sayısı bilgisi girildi ve yeni özellikli matris yerleştirildi.

```
"""
import pandas as pd
import pandas
import numpy as np
from skimage.io import imread, imshow
from skimage.transform import resize
from skimage.feature import hog
from skimage import exposure
import matplotlib.pyplot as plt
adres='C:/Users/FbMmm/Downloads/khakDataDogru.csv'
kisi=pandas.read_csv(adres,header=None,usecols=[0])
dilim_sayısı=pandas.read_csv(adres,header=None,usecols=[1])
sinif=pandas.read_csv(adres,header=None,usecols=[2])
x=pandas.read_csv(adres,header=None)
goruntu_matrisi=x.iloc[:,3:50179]#x.drop([0,1,2],axis=1)



y=np.zeros((925,50179))
for i in range(0,925):
    y[i,0]=kisi.iloc[i,0]
    y[i,1]=dilim_sayısı.iloc[i,0]
    y[i,2]=sinif.iloc[i,0]
for i in range(0,925):
    image = np.resize(goruntu_matrisi.iloc[i],(224,224))
    fd, hog_image = hog(image, orientations=9, pixels_per_cell=(8, 8),
                    cells_per_block=(2, 2), visualize=True)
    hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))
    hog_image_rescaled=np.resize(hog_image_rescaled,(1,50176))

    for j in range (0,50176):
                y[i,3+j]=hog_image_rescaled[0,j]

DF = pd.DataFrame(y)
DF.to_csv("C:/Users/FbMmm/OneDrive/Masaüstü/Yeni klasör/hog.csv")
```

HOG Verisetinin ilk resmi ve histogramı:



2.1.2. Düzey Birlikte Oluşum Matrisi(GLCM)

Bu bölümde; bir görüntüde belirli değerlere ve belirli bir uzamsal ilişkiye sahip piksel çiftlerinin ne sıklıkla meydana geldiğini hesaplayarak, bir GLCM matrisi oluşturarak bu matristen istatistiksel ölçümler çıkararak bir görüntünün dokusunu karakterize edilmiştir

GLCM, her pikseldeki yoğunluk varyasyonunu hesaplar.Burada referans ve komşu piksel olarak adlandırılan bir seferde iki piksel arasındaki ilişkiyi dikkate alır.

Programlama Formulasyonu:

1- Görüntü verilerini nicelleştirme
2- GLCM'yi yani. N x N boyutunda matris olusturma(GLCM'yi simetrik yapma, GLCM'yi normalleştirme)
3- Seçilen Özelliği hesaplama(Homojenliğine,enerjisine veya entropisine vs. göre)

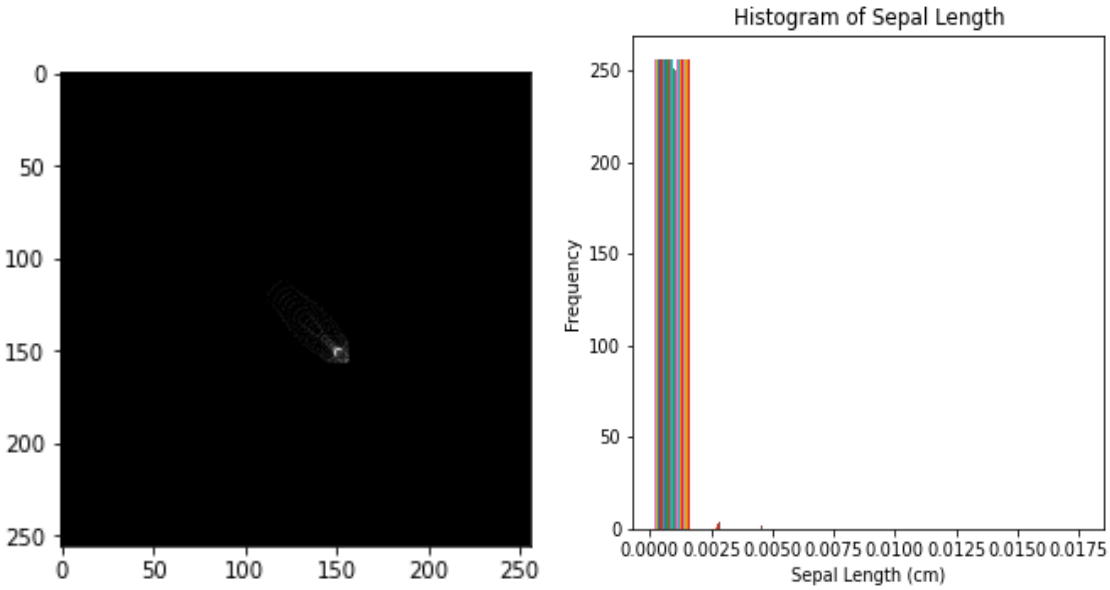
Python ile GLCM Feature Extraction:

Yeni dataset oluşturuldu.Orjinal Datasetdeki gibi sınıf,kişi ve dilim sayısı bilgisi girildi ve yeni özellikli matris yerleştirildi.

```
"""
import matplotlib.pyplot as plt
import pandas as pd
import pandas
from skimage.feature import greycomatrix, greycoprops
import numpy as np
adres='C:/Users/FbMmm/Downloads/khakDataDogru.csv'
kisi=pandas.read_csv(adres,header=None,usecols=[0])
dilim_sayısı=pandas.read_csv(adres,header=None,usecols=[1])
sinif=pandas.read_csv(adres,header=None,usecols=[2])
x=pandas.read_csv(adres,header=None)
goruntu_matrisi=x.iloc[:,3:50179]#x.drop([0,1,2],axis=1)
y=np.zeros((925,262147))

for i in range(0,925):
    y[i,0]=kisi.iloc[i,0]
    y[i,1]=dilim_sayısı.iloc[i,0]
    y[i,2]=sinif.iloc[i,0]
```

```


 for i in range(0,925):
     image = np.resize(goruntu_matrisi.iloc[i],(224,224))

     PATCH_SIZE = 21

     # select some patches from grassy areas of the image
     grass_locations = [(90, 54), (42, 63), (70, 92), (105, 105)]
     grass_patches = []
     for loc in grass_locations:
         grass_patches.append(image[loc[0]:loc[0] + PATCH_SIZE,
                                    loc[1]:loc[1] + PATCH_SIZE])

     # select some patches from sky areas of the image
     sky_locations = [(38, 14), (49, 28), (57, 57), (75, 39)]
     sky_patches = []
     for loc in sky_locations:
         sky_patches.append(image[loc[0]:loc[0] + PATCH_SIZE,
                                  loc[1]:loc[1] + PATCH_SIZE])


     for patch in (grass_patches + sky_patches):
         glcm = greycomatrix(patch, distances=[5], angles=[0], levels=512,
                             symmetric=True, normed=True)
         glcm=np.resize(glcm,(1,262144))
         for j in range (0,262144):
             y[i,3+j]=glcm[0,j]

 DF = pd.DataFrame(y)
 DF.to_csv("C:/Users/FbMmm/OneDrive/Masaüstü/Yeni klasör/glcm.csv")
```

GLCM Verisetinin ilk resmi ve histogramı:



HOG VE GLCM datasetlerini okuma ve verilerini standartlaştırma

```
29
30  adres="C:/Users/FbMmm/OneDrive/Masaüstü/Yeni klasör/glcm.csv"
31  #adres="C:/Users/FbMmm/OneDrive/Masaüstü/Yeni klasör/hog.csv"
32  x=pandas.read_csv(adres,header=None)
33
34  x=x.drop([0],axis=0)
35  x=x.drop([0],axis=1)
36
37
38  x_array=[]
39  y_array=[]
40  goruntu_matrisi=x.iloc[:,3:].values
41  sinif=x.iloc[:,2].values
42  x_array=np.array(goruntu_matrisi)
43  y_array=np.array(sinif)
44  x_train, x_test,y_train, y_test = train_test_split(x_array,y_array, test_size=0.20,
45                                                     random_state=20,stratify=y_array,shuffle=True)
46
47
48
49  sc = StandardScaler()
50  x_train = sc.fit_transform(x_train)
51  x_test = sc.transform(x_test)
52  #-------------------------------------------------
53
```

Eğitimde hastaların görüntülerinin bilgilerini içeren görüntü matrisi x'i, sınıflandırma bilgilerine sahip sınıf matriside y'yi temsil eder.

3-METOT

3.1.Sınıflandırma Türleri

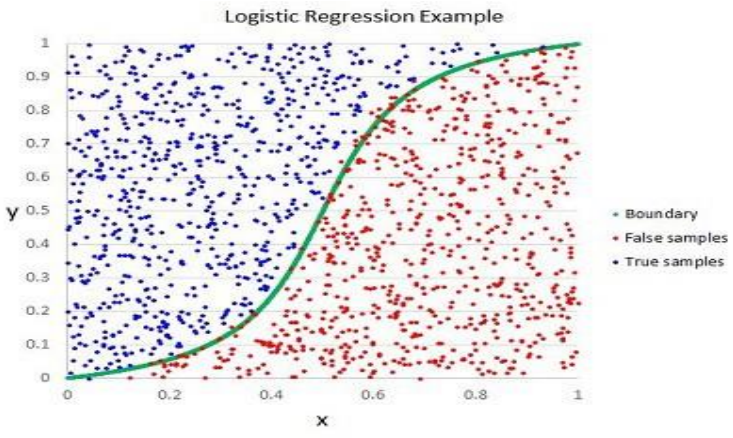
Kategorik verilerin tahmini i çin ise Classification (Sınıflandırma) kullanılır.Örneğin sınıflandırma ile bir kişinin erkek mi yoksa kadın mı yada hasta mı yoksa sağlıklı mı olduğunu tahmin edebiliriz.

Bizim datasetimizde ‘’Büyük Hücreli’’=1 ve ‘’Küçük hücreli’’=2 sınıflandırıp tahmin ettik.

```
import numpy as np
import pandas as pd
import pandas
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from skimage.transform import resize
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import confusion_matrix
from keras.models import Sequential
import keras
from keras.layers import Dense
```

3.1.1.Lojistik Regresyon(Logistic Regression)

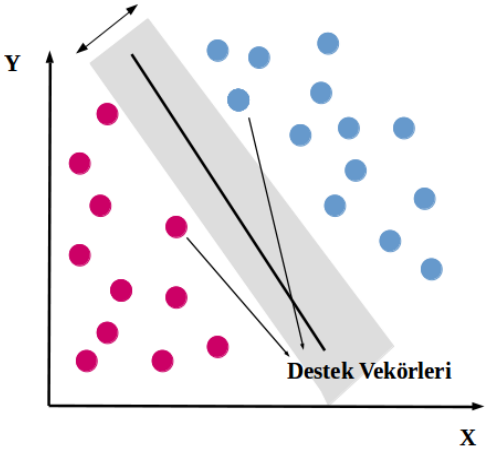
Lojistik regresyonun amacı, iki yönlü karakteristiği (bağımlı değişken = yanıt veya sonuç değişkeni) ile ilgili bir dizi bağımsız (öngörücü veya açıklayıcı) değişken arasındaki ilişkiyi tanımlamak için en uygun (henüz biyolojik olarak makul) modeli bulmaktır. (Şekil 2 deki gibi)



Şekil 2

3.1.2.Destek Vektör Makinesi(Support Vector Machine)

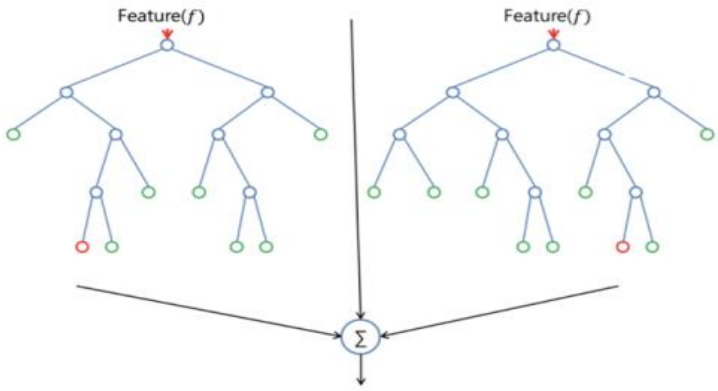
Her bir veri maddesini belirli bir koordinatın değeri olan her özelliğin değeri ile birlikte n-boyutlu boşluğa (burada n sahip olduğunuz özelliklerin sayısı) bir nokta olarak çizilir. Ardından, iki sınıftan oldukça iyi ayrım yapan hiper-düzlemi bularak sınıflandırma gerçekleştirilir.(Şekil 3 gibi)



Şekil 3

3.1.3. Rassal orman (Random Forest)

Birden fazla karar ağacını oluşturup daha doğru ve istikrarlı bir tahmin elde etmek için onları birleştirmesidir.(Sekil 5 deki gibi)



Sekil 5

3.1.4. Oylama Sınıflandırıcısı,(Voting Classification)

Oylama Sınıflandırıcısına iletilen her sınıflandırıcının bulgularını basitçe toplar ve oylamanın en yüksek çoğunluğuna dayalı olarak çıktı sınıfını tahmin eder. Buradaki fikir, ayrı özel modeller oluşturmak ve her biri için doğruluğu bulmak yerine, bu modeller tarafından eğitilen ve her bir çıktı sınıfı için birleşik oy çoğunluğuna dayalı çıktıyı tahmin eden tek bir model oluşturmaktır.

Logistik regresyon,RandomForest,SVC ve Voting sınıflandırma

```
#-------------------------------------------------

log=LogisticRegression(solver='lbfgs',max_iter=10000).fit(x_train,np.ravel(y_train,order='C'))
print(log.score(x_test,np.ravel(y_test,order='C')))

rnd=RandomForestClassifier(n_estimators=10).fit(x_train,np.ravel(y_train,order='C'))
print(rnd.score(x_test,np.ravel(y_test,order='C')))

svm=SVC(gamma='auto',probability=True,kernel='linear').fit(x_train,np.ravel(y_train,order='C'))
print(svm.score(x_test,np.ravel(y_test,order='C')))

voting=VotingClassifier(estimators=[('lr',log),('rf',rnd),
                                    ('svc',svm)],voting='hard').fit(x_train,np.ravel(y_train,order='C'))
print(voting.score(x_test,np.ravel(y_test,order='C')))


#-------------------------------------------------
```

3.1.4.Karar Ağaçları(Decision Tree Classification)

Bu yöntem;, özellik ve hedefe göre karar düğümleri (decision nodes) ve yaprak düğümlerinden (leaf nodes) oluşan ağaç yapısı formunda bir model oluşturan bir sınıflandırma yöntemidir. Karar ağacı algoritması, veri setini küçük ve hatta daha küçük parçalara bölerek geliştirilir.(Sekil 4 deki gibi)

Sekil 4

DesicionTree sınıflandırma

```
tree=DecisionTreeClassifier(criterion="entropy", max_depth=3).fit(x_train,np.ravel(y_train,order='C'))
print(tree.score(x_test,np.ravel(y_test,order='C')))
```

3.1.6. Topluluk Öğrenme(Ensemble Classification)

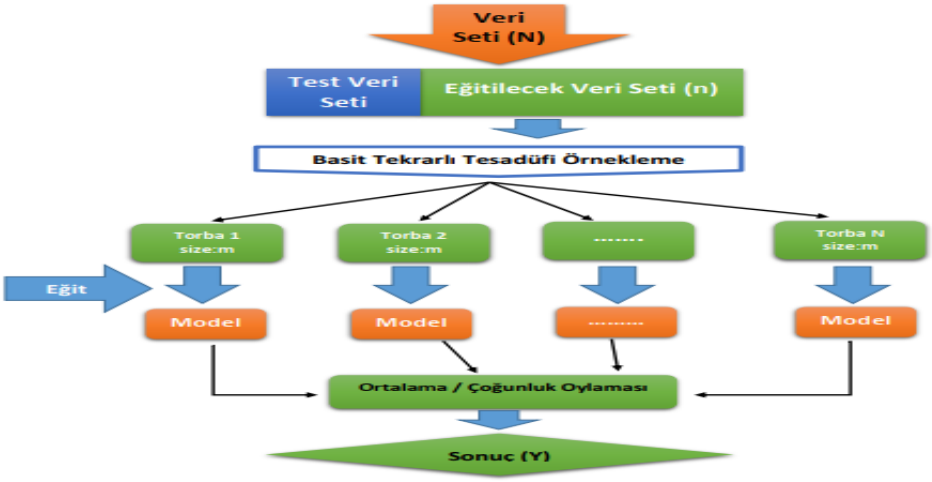
Bu sınıflandırmada Aynı sınıflandırma görevinde birden fazla sınıflandırıcı kullanılıyor, ortak akıl sağlanmış oluyor. Bu yöntemde farklı doğruluk skorlarına sahip sınıflandırıcıların sonuçları farklı yöntemlerle (oylama, ortalama vb.) birleştiriliyor. Böylelikle tek bir sınıflandırıcıdan daha iyi sonuçlar elde etme imkanı bulunuyor.

Yaygın olarak kullanılan yöntemleri:

1- Bagging
2- Boosting

3.1.6.1.Torbalama(Bagging)

Bu Ensemble sınıflandırma yöntemlerindendir.Bu yöntem temel öğrenicilerin (base learner) herbiri, eğitim setinin rastgele seçilen farklı alt kümeleriyle eğitilir. Veri, önce eğitim ve test olarak ayrılır. Daha sonra eğitim için ayrılan veri setinden rastgele seçim yapılır ve her bir öğrenicinin çantasına konur. Torbadan çekilen topun torbaya tekrar konması gibi seçilenler tekrar seçilebilecek şekilde eğitim kümesinde kalmaya devam eder. Seçilen miktar eğitim için ayrılandan fazla değildir (genelde %60). Farklı eğitim setlerinin se çilmesindeki ama çkarar farklılıkları (model farklı eğitim setiyle oluşunca doğal olarak kararlarda da bir miktar farklılık oluşacaktır) elde ederek başarıyı yükseltmektir. Kararlar ağırlıklı oylama ile birleştirilir.



Sekil 6

Karar Ağaçları sınıflandırmasını Torbalama yapma

```
bag=BaggingClassifier(tree,n_estimators=10,max_samples=0.8,
                      n_jobs=-1,random_state=1).fit(x_train,np.ravel(y_train,order='C'))
print(bag.score(x_test,np.ravel(y_test,order='C')))
```

3.1.6.2.Güçlendirme(Boosting)

Torbalama yönteminin farklı bir versiyonudur. Fark; öğrenme sonuçlarının mütekip öğrenici için kullanılıyor olmasıdır. Diğer yöntemlere göre daha yaygındır, hızlı çalışır az bellek kullanır. Eğitim için ayrılan veri setinden bir temel öğrenici için rastgele seçim yapılır. Öğrenme gerçekleşir, model test edilir. Sonuçlardan yanlış sınıflandırılan örnekler belirlenir. Bunlar bir sonraki öğrenici için örnek seçiminde önceliklendirilir (seçilme olasılıkları arttırılır). Her seferinde bu bilgi güncellenir. Bagging yönteminde her bir örneğin seçilme şansı eşittir.

2 bilinen yöntemleri vardır: Adaboost ve Gradient Boosting

A-) AdaBoost

Yüksek düzeyde, AdaBoost, son sınıflandırmaya karar vermek için ormandaki her bir karar ağacının yaptığı tahminleri belirlemesi açısından Random Forest'a benzer. Bununla birlikte, bazı ince farklılıklar vardır. Örneğin AdaBoost'ta karar ağaçlarının derinliği 1'dir Ve her bir karar ağacının yaptığı tahminler, model tarafından yapılan nihai tahmin üzerinde farklı etkilere sahiptir.

SVC sınıfını AdaBoost ile güçlendirme

```
svc=SVC(probability=True, kernel='linear')
ada_boost=AdaBoostClassifier(n_estimators=1, base_estimator=svc,
                             learning_rate=0.001).fit(x_train,np.ravel(y_train,order='C'))

print(ada_boost.score(x_test,np.ravel(y_test,order='C')))
```

B-) Gradyan Güçlendirme(Gradient Boosting)

Gradyan artırma, önceki modeldeki hatayı hesaplayan yeni modellerin oluşturulduğu ve ardından son tahmini yapmak için arta kalanların eklendiği bir yöntemdir.

Gradient Boost algoritması AdaBoost gibi her ağaçtan sonra bir iyileştirme yapmak için düğüm oluşturmamaktadır. Bunun yerine Yaprak (Leaf) ile başlar. Bu yaprak tüm

ağırlıklar için bir ilk tahmini temsil eder. Buradaki ilk tahmin ortalama değerdir. Ardından Gradient Boost bir ağaç oluşturur.

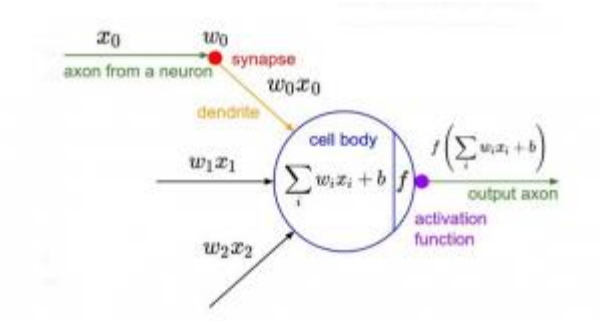
SVC sınıfını GradientBoosting ile güçlendirme

```
#--------------------------------
svc=SVC(probability=True, kernel='linear')
gbrt2=GradientBoostingClassifier(init=svc,
                                 random_state=0,n_estimators=10).fit(x_train,np.ravel(y_train,order='C'))
print(gbrt2.score(x_test,np.ravel(y_test,order='C')))


```

3.1.7.Yapay Sinir Ağları

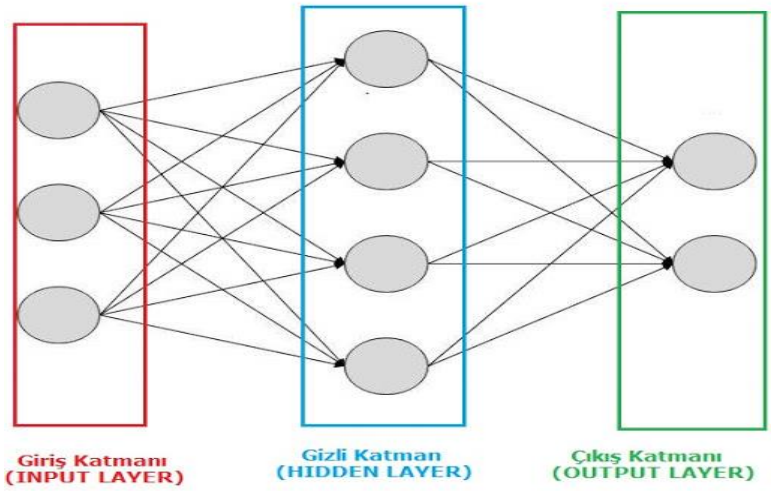
Yapay sinir ağları insan beyni örnek alınarak, öğrenme sürecinin matematiksel olarak modellenmesi sonucu ortaya çıkmıştır. Beyindeki biyolojik sinir ağlarının yapısını, öğrenme, hatırlama ve genelleme kabiliyetlerini taklit eder.

İnsandaki bir sinir hücresinin matematiksel modeli ise şu şekilde gösterilebilir:



Sekil 7

Yapay sinir ağları yapay sinir hücrelerinin birbirine bağlanmasıyla oluşan yapılardır.Yapay sinir ağları üç ana katmanda incelenir; Giriş Katmanı, Ara (Gizli) Katmanlar ve Çıkış Katmanı**.**Bilgiler ağa girdi katmanından iletilir. Ara katmanlarda işlenerek oradan çıktı katmanına gönderilirler. Bilgi işlemeden kasıt ağa gelen bilgilerin ağın ağırlık değerleri kullanılarak çıktıya dönüştürülmesidir. Ağın girdiler için doğru çıktıları üretebilmesi için ağırlıkların doğru değerlerinin olması gerekmektedir.



Sekil 8

```
classifier = Sequential()
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu',
                     input_dim = 50176))
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['acc'])
classifier.fit(x_train, y_train, batch_size = 8, nb_epoch = 20)

print(classifier.evaluate(x_test, y_test))

y_pred = classifier.predict(x_test)
y_pred = (y_pred > 0.5)
```

```
plt.plot(classifier.history.history['acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
# Plot training & validation loss values
plt.plot(classifier.history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

3.2.Sonuç Değerlendirme

3.2.1.Karmaşıklık Matrisi(Confusion Matrisi)

Karmaşıklık matrisi test verisi ile tahmin değerlerinin karşılaştırmamızı ve yaptığımız modelimizin performansını ölçmemizi sağlar.(Sekil 7 deki gibi)

TP (True Positive)ve TN (True Negative)değerleri doğru değer sayısını verir.

FP (False Positive)ve FN (False Negative)değerleri yanlış değer sayısını verir.

Doğru değer sayısının tüm değer sayılarına oranı Doğru Tahmin Oranını gösterir.

Yanlış değer sayısının tüm değer sayılarına oranı Yanlış Tahmin Oranını gösterir.

Confusion Matrix and ROC Curve

		Predicted Class	
		No	Yes
Observed Class	No	TN	FP
	Yes	FN	TP

TN	True Negative
FP	False Positive
FN	False Negative
TP	True Positive

Model Performance

Accuracy	= (TN+TP)/(TN+FP+FN+TP)
Precision	= TP/(FP+TP)
Sensitivity	= TP/(TP+FN)
Specificity	= TN/(TN+FP)

Şekil 9

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

3.2.2. Ortalama Kare Hata (Mean Squared Error):

Ortalama kare hata bir regresyon eğrisinin bir dizi noktaya ne kadar yakın olduğunu söyler. MSE, bir makine öğrenmesi modelinin, tahminleyicinin performansını ölçer, her zaman pozitif değerlidir ve MSE değeri sıfıra yakın olan tahminleyicilerin daha iyi bir performans gösterdiği söylenebilir.

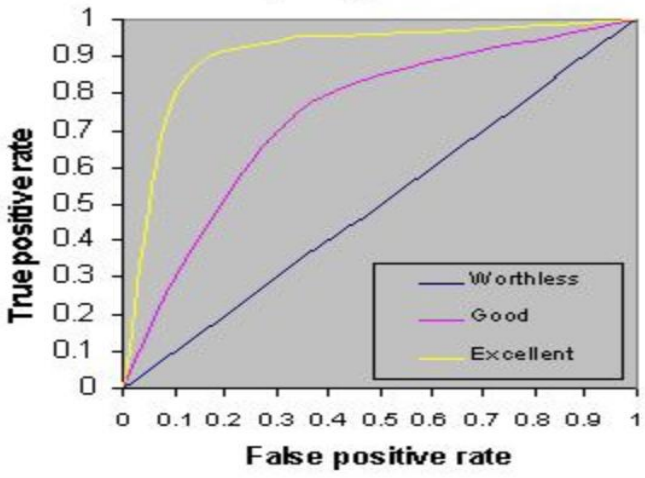
```
y_pred=model.predict(x_test)
y_pred = (y_pred > 0.5)
np.sqrt(mean_squared_error(y_test,y_pred))
```

3.2.3.ROC eğrisi(ROC curve)

ROC bir olasılık eğrisidir. altında kalan alan olan AUC ayrılabilirliğin derecesini veya ölçüsünü temsil eder. AUC Modelin sınıfları ne kadar başarılı ayırt edebildiğini anlatır. AUC arttıkça, model 0'ları 0 ve 1'leri 1 olarak tahmin etmede daha iyidir. AUC değeri 0.5 olan modelin performansının kötü olduğunu ve rastgele tahminleme yaptığı söylenebilir.

ROC eğrisinde X ekseninde FPR(Yanlış Pozitif Oran) ve Y ekseninde ise TPR (Gerçek Pozitif Oranı) bulunmaktadır.

Eğrinin altında kalan arttıkça sınıflar arasında ayırt etme performansı artmaktadır.(Sekil 8 deki gibi)



Sekil 10

```
false_positive_rate, true_positive_rate, threshold = roc_curve(y_test, y_pred)
print('roc_auc_score for MODEL: ', roc_auc_score(y_test, y_pred))
#X-axis and True Positive Rate on Y-axis.
plt.subplots(1, figsize=(10,10))
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate)
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

4.UYGULAMA

SINIFLANDIRMA:

Logistik regresyon,RandomForest,SVC ve Voting sınıflandırma performans sonuçları

OUTPUT: (HOG için)

Lojistik Regresyon: 0.8216216216216217

Rassal Orman: 0.7405405405405405

Destek Vektör Makinesi: 0.7783783783783784

Oylama sınıflandırması :0.7783783783783784

OUTPUT: (GLCM için)

Lojistik Regresyon: 0.654054054054054

Rassal Orman: 0.7351351351351352

Destek Vektör Makinesi: 0.654054054054054

Oylama sınıflandırması : 0.7351351351351352

DesicionTree sınıflandırma ve bu sınıfı Bagging yapma performans sonuçları

OUTPUT: (HOG için)

Karar Ağaçları: 0.6972972972972973

Torbalama: 0.7081081081081081

OUTPUT: (GLCM için)

Karar Ağaçları: 0.654054054054054

Torbalama: 0.6594594594594595

SVC sınıfını GradientBoosting ile arttırma performans sonuçları

OUTPUT: (HOG için)

Gradyan Güçlendirme :0.7945945945945946

OUTPUT: (GLCM için)

Gradyan Güçlendirme: 0.7081081081081081

SVC sınıfını AdaBoost ile arttırma performans sonuçları

OUTPUT: (HOG için)

Adaboost: 0.8

OUTPUT: (GLCM için)

Adaboost: 0.7189189189189189

Ort.Kare Hata, ROC eğrisi(AUC değeri) ve Karmaşıklık Matrisi Görüntüleme

Çıktının Tablosu

Ortalama Kare hata değeri, ROC eğrisinin altında kalan alanın(AUC) **skoru,Karmasıklık Matrisinde yer verilen Test verilerinin üzerinden** (185 tane veri vardır) **doğru bilinen verilerin sayısı(**Matriste 1.satır 1.sutundaki ve 2.satır 2.sutundaki değerlerin toplamı**) ve yanlış bilinen verilerin sayısı (**Matriste 1.satır 2.sutundaki ve 2.satır 1.sutundaki değerlerin toplamı**) tabloda sınıflandırmalara göre yer verilmiştir.**

HOG için:

	Ort.Kare_ Hata	AUC_score	Doğru-(Konf.Matris)	Yanlış-(Konf.Matris)
Lojistik Reg.	0.5742209763929995	0.5307692307692308	124	61
Rassal Orman	0.5927489783638191	0.5	120	65
Destek.V.M.	0.4707670566443893	0.748076923076923	144	41
Oylama Sın.	0.4707670566443893	0.748076923076923	144	41
Karar Ağacı	0.5501842443243016	0.5868589743589744	129	56
Torbalama	0.5402702026688978	0.5846153846153846	131	54
Gradyan Güç	0.4532167311622613	0.7464743589743591	147	38
AdaBoost	0.4472135954999579	0.7506410256410257	148	37

GLCM i çin:

	Ort.Kare_ Hata	AUC_score	Doğru- (Konf.Matris)	Yanlış- (Konf.Matris)
Lojistik Reg.	0.5881716976750462	0.6064102564102564	121	64
Rassal Orman	0.5146502354656655	0.6653846153846154	136	49
Destek.V.M.	0.5881716976750462	0.6134615384615385	121	64
Oylama Sın.	0.5146502354656655	0.6653846153846154	136	49
Karar Agacı	0.5881716976750462	0.5112179487179487	121	64
Torbalama	0.5835585150955648	0.5224358974358975	122	63
Gradyan Güç	0.5402702026688978	0.6269230769230769	131	53
AdaBoost	0.5301708036860207	0.6317307692307693	133	52

Yapay Sinir Ağlar Sınıflandırması

Test Değerlendirme sonuçları

Orjinal Veriseti i çin:

Kayıp=0.6449844547220178

Doğruluk=0.6486486196517944

HOG i çin:

Kayıp=0.6449844547220178

Doğruluk=0.7621621489524841

GLCM i çin:

Kayıp=1.4406425025012042

Doğruluk=0.6972972750663757

Ortalama Kare hata değeri, ROC eğrisinin altında kalan alanın(AUC) **skoru,Karmasıklık Matrisinde yer verilen Test verilerinin üzerinden** (185 tane veri vardır) **doğru bilinen verilerin sayısı(**Matriste 1.satır 1.sutundaki ve 2.satır 2.sutundaki değerlerin toplamı**) ve yanlış bilinen verilerin sayısı (**Matriste 1.satır 2.sutundaki ve 2.satır 1.sutundaki değerlerin toplamı**) tabloda ANN sınıflandırmasına göre yer verilmiştir.**

Orjnal Veriseti için:

	Ort.Kare_ Hata	AUC_score	Doğru- (Konf.Matris)	Yanlış- (Konf.Matris)
ANN	0.5927489783638191	0.5	120	65

HOG için:

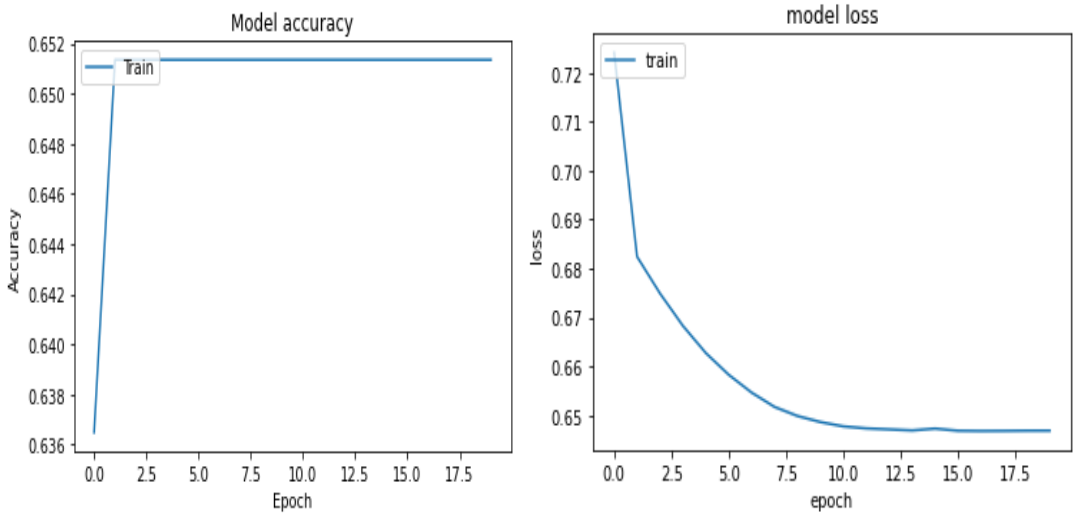
	Ort.Kare_ Hata	AUC_score	Doğru- (Konf.Matris)	Yanlış- (Konf.Matris)
ANN	0.4223486455268661	0.7919871794871794	152	33

GLCM için:

	Ort.Kare_ Hata	AUC_score	Doğru- (Konf.Matris)	Yanlış- (Konf.Matris)
ANN	0.5501842443243016	0.6503205128205128	129	56

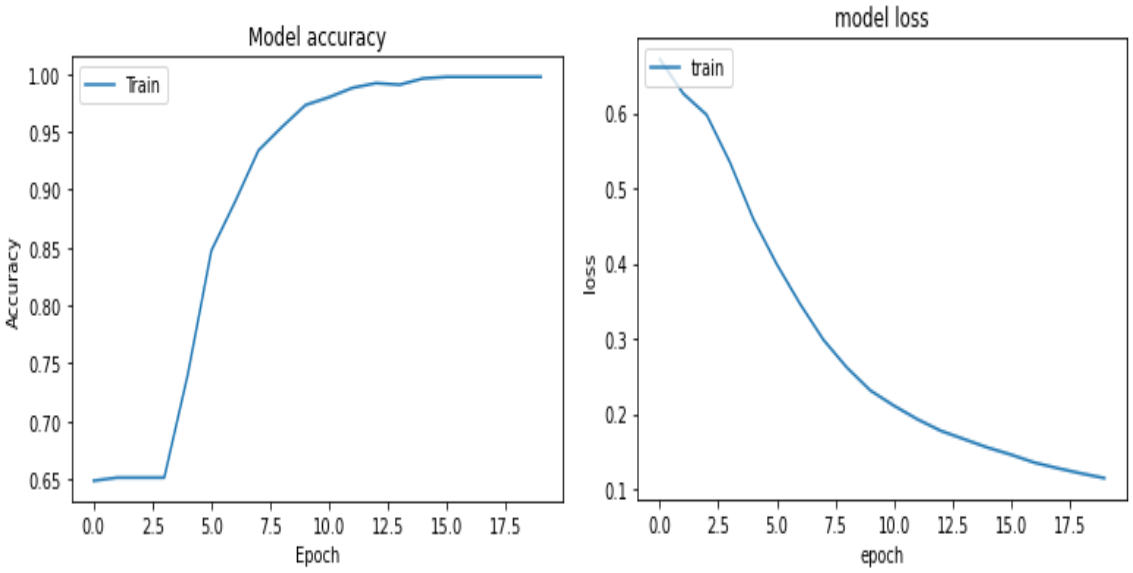
Her eğitim devirinde(epoch) Eğitim sırasında Model doğruluğu ve kaybı

Orjinal Veriseti için:

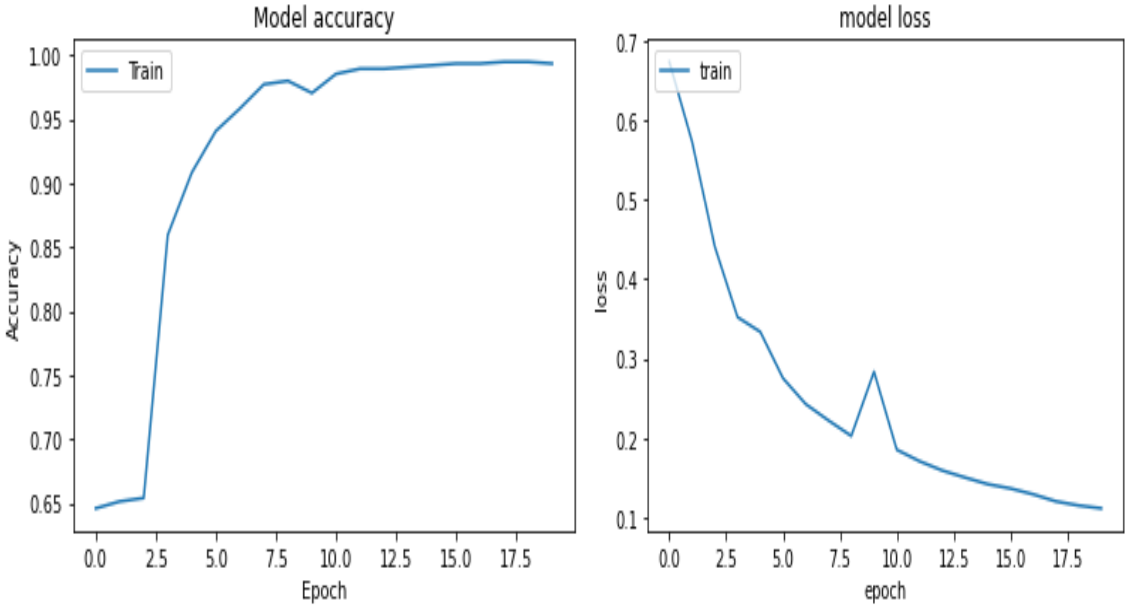


Model doğruluğu 0.7’i geçememiştir.Kayıp 0.6’da kalmıştır.

Hog için:



GLCM için:



Model doğruluğu 0.9’u geçmiştir.Kayıp 0.1 altına düşmüştür

5.TARTIŞMA

AUC skorlar ve normal skorlar;

Pozitif 120 test verisi var iken 65 negatif test veriseti vardır.Test Verisetinde dengesizlik söz konusudur. Bu yüzden normal skorlara ek olarak ROC eğrisi ve AUC değerine göre bir değerlendirme yapılmıştır.

HOG için Normal skorlara bakarsak Lojistik regresyon birincidir, ancak normal skor sınıf dengesizliğine dikkat etmeyip doğru sonuç vermez.Eğer verisetinde Sınıf dengesizliği var ise ROC eğrisine göre AUC skorlar aynı olmayacaktır.

Genel olarak özellikle sınıf dengesizliğiden daha çok etkilenen sınıflar(Karar Ağaçları vs.) normal skorlar ile AUC skorları arasında uçurum vardır.

AUC Skorlar HOG için ;

Lojistik Regresyon,Rassal Orman ve Destek Vektör Makinesi arasında en iyi sonuç veren Destek Vektör Makinesi oldu.(Bu yüzden Oylama sınıflandırması bu sınıf değerini seçti)

Destek Vektör Makinesi sınıflandırmada dördüncü oldu.

En kötü sonucu Rassal Orman verdi.Rassal Orman AUC değeri 0.5'tir. Modelin performansı kötüdür ve rastgele tahminleme yaptmıştır

İkinci en kötü sonucu Lojistik Regresyon vermiştir. AUC değeri 0.5'e yakındır. Modelin performansı fena değil ve rastgele tahminleme yaptmış olabilir.

Karar Ağaçları sınıflandırmaların sondan üçüncü kötü sonucunu verdi. AUC değeri 0.5'e yakındır. Modelin performansı fena değil ve rastgele tahminleme yaptmış olabilir. Torbalama yöntemi bu sınıfın sıralamasını değiştiremedi. Yine AUC değeri 0.5'e yakındır.

Destek Vektör Sınıflandırmasını Güçlendirmek için Adaboost ve Gradyan Güçlendirme yöntemleri Destek Vektör Sınıflandırması sonucu iyi yönde biraz değiştirerek, Adaboost birinci sınıflandırıcı oldu. Ancak DVM sınıflandırıcısını güçlendirmesinde belirgin iyileştirme yoktur.

Gradyan Güçlendirme, sınıflandırma yarışında üçüncü olmuştur.DVM sınıflandırıcısını güçlendirmesinde belirgin iyileştirme yoktur.

Medikal görüntü sınıflandırma alanında genellikle tercih edilen çoğu zaman iyi sonuç verdiği gözlenen Yapay Sinir Ağları birinci olmuştur.AUC değeri 0.8'e yakındır. İyi bir sonuçtur. Öngörüleceği üzere gizli katmanlar arttırılarak sonuç daha iyi sonuçlar elde edilebilir.

AUC skorları GLCM için;

Lojistik Regresyon,Rassal Orman ve Destek Vektör Makinesi arasında en iyi sonuç veren Rassal Orman oldu.(Bu yüzden Oylama sınıflandırması bu sınıf değerini seçti)Rastgele Orman, özelliklerin rastgele alt kümeleri oluşturarak ve bu alt kümeleri kullanarak daha küçük ağaçlar oluşturarak, çoğu zaman aşırı uydurmayı(overfitting) engeller. Muhtemelen eğitimde aşırı öğrenme oldu ve Rassal Orman en şanslı çıkarak sınıflandırmanın birincisi oldu.Ve HOG özellik çıkarımında kullanılan Rassal Ormandan daha iyi sonuç vermiştir.

Destek Vektör Makinesi sondan dördüncü oldu.DVM ikili sınıflandırmada iyi olsada özellik sayısı örnek sayısından fazla olduğunda DVM iyi performans göstermez. Bu yüzden bu sınıflandırma iyi performans gösterememiş olabilir. HOG özellik çıkarımında kullanılan özelliklerden dolayı muhtemelen GLCM özellik çıkarımlarında daha fazla özellik sayısı oluştu.Buda DVM sınıfını kötü etkiledi.

Lojistik Regresyon sondan üçüncü oldu.Bu sınıfın varsayımından dolayı kestirim değeri olarak birbirine çok yakın olduğu halde, sınır çizgisinden dolayı değerine göre atama yapıldığında, neredeyse birbirine eşit olan bu iki değer farklı gruplara atanabilecektir.Muhtemelen negatif ile pozitif örnekler iyi sınıflandırılamadı.Yinede HOG özellik çıkarımında kullanılan Lojistik Regresyon'dan daha iyi sonuç vermiştir.

Karar Ağaçları sınıflandırmaların en kötü sonucunu verdi ve Torbalama yöntemi bu sınıfın durumunu düzeltemedi. Veriyi iyi bir şekilde açıklamayan aşırı karmaşık ağaçlar üretmiş olabilir. Bu durumda ağaç dallanması takip edilememiş olabilir.Aşırı öğrenme yaşanmış olabilir .(Bu problemin çözümü için model parametrelere kısıtlamalar ve budama gibi yöntemler kullanılabilir.) AUC değeri 0.5'e yakındır. Modelin performansı kötü ve rastgele tahminleme yapmış olabilir.

Destek Vektör Sınıflandırmasını Güçlendirmek için Adaboost ve Gradyan Güçlendirme yöntemleri Destek Vektör Sınıflandırması sonucu iyi yönde biraz değiştirerek, Adaboost birinci ikinci iyi sınıflandırıcı oldu. Gradyan Güçlendirme, sınıflandırma yarışında üçüncü olmuştur. Ancak ikiside DVM sınıflandırıcısını güçlendirmesinde belirgin iyileştirme yapamamıştır.

Beklenmedik şekilde Yapay Sinir Ağları sınıflandırmada dördüncü olmuştur.Muhtemelen eğitim sırasında Aşırı öğrenme oldu , bu sınıflandırıcının Medikal görüntü işlemede önem arz eden resmi piksel piksel işleme avantajına rağmen birinci olamadı.

Ortalama Kare Hata

Ortalama Kare Hatası tahmin edilen sonuçlarınızın gerçek sayıdan ne kadar farklı olduğuna dair size mutlak bir sayı verir.Tüm sınıflandırma sonuçlarıyla karşılaştırmak için gerçek bir sayı verir ve en iyi sınıflandırma modelini seçmemize yardımcı olur.

Tüm sonuçlar ROC eğrisi-AUC skorları ile orantılıdır.

Karmaşıklık Matrisi değerlendirmesi;

HOG için;

Birinci olan Yapay Sinir Ağlarının Karmaşıklık Matrisine baktığımızda [[107 13],[20 45]], Pozitif 120 verisetinden 107'sini doğru bilmiş, 65 negatif verisetinden 45 tanesini doğru bilmiştir. Sonuca göre pozitif datasetler üzerinde etkili çalışmışken negatif test veriseti üzerinde de etkili olmuş ve sınıf dengesizliğinden çok etkilenmemiştir. Negatif testte en iyi sonuç bu sınıftatır.Adaboost'a göre negatif test veri setleri üzerinde daha iyi performans sergilemiştir.

İkinci olan Adaboost'un Karmaşıklık Matrisine baktığımızda [[110 10], [27 38]] pozitif 120 verisetinden 110'sini doğru bilmiş, 65 negatif verisetinden 38 tanesini doğru bilmiştir. Sonuca göre sınıf dengesizliğinden etkilenmiş olacak ki pozitif test verisetleri üzerinde daha etkili çalışmışken negatif test veriseti üzerinden yaklaşık yüzde 50 gibi bir performans vermiş.

Üçüncü olan Gradyan Güçlendirici Karmaşıklık Matrisine baktığımızda [[109 11],[27 38]] Pozitif 120 verisetinden 109'unu doğru bilmiş, 65 negatif verisetinden 38 tanesini doğru bilmiştir. Sonuca göre pozitif datasetler üzerinde daha etkili çalışmışken negatif test veriseti üzerinde etkili olamamış ve sınıf dengesizliğinden Adaboost gibi etkilenmiştir.

Dördüncü olan Destek Vektör Makinesi Güçlendirici Karmaşıklık Matrisine baktığımızda [[102 18],[23 42]] Pozitif 120 verisetinden 102'sini doğru bilmiş, 65 negatif verisetinden 42 tanesini doğru bilmiştir. Sonuca göre pozitif datasetler üzerinde etkili çalışmışken negatif test veriseti üzerinde de etkili olmuş ve sınıf dengesizliğinden çok etkilenmemiştir, Yapay Sinir Ağları gibi sınıf dengesizliğinden çok etkilenmemiştir. Adaboost'a göre negatif test veri setleri üzerinde daha iyi performans sergilemiştir. Negatif testte en iyi ikinci sonuç bu sınıftatır.

Beşinci olan Karar Ağaçları üzerinde Torbalama sınıflandırması Karmaşıklık Matrisine baktığımızda [[120 0] , [54 11]] Pozitif 120 verisetinden hepsini doğru bilmiş, 65 negatif verisetinden 11 tanesini doğru bilmiştir. Sonuca göre pozitif datasetler üzerinde daha etkili çalışmışken negatif test veriseti üzerinde etkili olamamış ve sınıf dengesizliğinden ciddi derecede etkilenmiştir.Pozitif bilinenlerin yüzdesi ile negatif bilinenlerin yüzdesi arasında uçurum vardır.Bu sınıf, Karar Ağaçlarının skorlarını iyileştirmesine rağmen Negatif test verileri üzerinde en kötü üçüncü sonucu veren sınıftır ve karar ağaçlarına göre negatif veriseti üzerinde kötü performans sergilemiştir.

Altıncı olan Karar Ağaçları sınıflandırması Karmaşıklık Matrisine baktığımızda [[115 5],[51 14]] Pozitif 120 verisetinden 115 tanesini doğru bilmiş, 65 negatif verisetinden 14 tanesini doğru bilmiştir. Sonuca göre pozitif datasetler üzerinde daha etkili çalışmışken negatif test veriseti üzerinde etkili olamamış ve sınıf dengesizliğinden ciddi derecede etkilenmiştir.

Yedinci olan Logistik Regresyon sınıflandırması Karmaşıklık Matrisine [[120 0], [61 4]] Pozitif 120 verisetinden hepsini doğru bilmiş, 65 negatif verisetinden sadece 4 tanesini doğru bilmiştir. Sonuca göre pozitif datasetler üzerinde daha etkili çalışmışken negatif test veriseti üzerinde etkili olamamış ve sınıf dengesizliğinden ciddi derecede etkilenmiştir.Pozitif bilinenlerin yüzdesi ile negatif bilinenlerin yüzdesi arasında uçurum vardır. Negatif test verileri üzerinde en kötü ikinci sonucu veren sınıftır.

Sonuncu olan Rassal Orman sınıflandırması Karmaşıklık Matrisine [[120 0],[65 0]]

Pozitif 120 verisetinden hepsini doğru bilmiş, 65 negatif verisetinden hiç doğru bilmiştir. Sonuca göre pozitif datasetler üzerinde daha etkili çalışmışken negatif test veriseti üzerinde hiç etkili olamamış ve sınıf dengesizliğinden aşırı derecede etkilenmiştir.Pozitif bilinenlerin yüzdesi yüzde 100 iken negatif bilinenlerin yüzdesi 0'dır . Negatif test verileri üzerinde en kötü sonucu veren sınıftır.

GLCM için;

Birinci olan Rassal Orman Karmaşıklık Matrisine baktığımızda [[108 12],[37 28]] Pozitif 120 verisetinden 108'sini doğru bilmiş, 65 negatif verisetinden 28 tanesini doğru bilmiştir. Sonuca göre sınıf dengesizliğinden etkilenmiş olacak ki pozitif test verisetleri üzerinde daha etkili çalışmışken negatif test veriseti üzerinden yaklaşık yüzde 50 gibi bir performans vermiş.

İkinci olan Adaboost'un Karmaşıklık Matrisine baktığımızda [[111 9],[43 22]] Pozitif 120 verisetinden 111'ini doğru bilmiş, 65 negatif verisetinden 22 tanesini doğru bilmiştir. Sonuca göre sınıf dengesizliğinden etkilenmiş olacak ki pozitif test verisetleri üzerinde daha etkili çalışmışken negatif test veriseti üzerinden yarısından 1/3 'ünü doğru bilerek negatif test setleri üzerinde kötü performans sergilemiştir.

Üçüncü olan Gradyan Güçlendirici Karmaşıklık Matrisine baktığımızda [[108 12]

[42 23]] Pozitif 120 verisetinden 108'ini doğru bilmiş, 65 negatif verisetinden 23 tanesini doğru bilmiştir. Sonuca göre pozitif datasetler üzerinde daha etkili çalışmışken negatif test veriseti üzerinde etkili olamamış ve sınıf dengesizliğinden Adaboost gibi etkilenmiştir.Ama 1 tane daha fazla Adaboost'tan negatif datasette doğru bilmiştir.

Dördüncü Yapay Sinir Ağları Karmaşıklık Matrisine baktığımızda [[97 23], [33 32]]

Pozitif 120 verisetinden 97'sini doğru bilmiş, 65 negatif verisetinden 32 tanesini doğru bilmiştir. Sonuca göre pozitif datasetler üzerinde etkili çalışmışken negatif test veriseti üzerinde de yüzde 50 etkili olmuş ve sınıf dengesizliğinden aşırı derecede etkilenmemiştir. Ancak HOG'daki YSA'daki iyi sonuç elde edememiştir.

Beşinci Destek Vektör Makinelerindeki Karmaşıklık Matrisine baktığımızda [[90 30], [34 31]] Pozitif 120 verisetinden 90'nını doğru bilmiş, 65 negatif verisetinden 31 tanesini doğru bilmiştir. Sonuca göre pozitif datasetler üzerinde etkili çalışmışken negatif test veriseti üzerinde de yüzde 50 etkili olmuş ve sınıf dengesizliğinden aşırı derecede etkilenmemiştir. Ancak HOG'daki DVM'den daha iyi sonuç elde etmiştir.

Altıncı olan Logisitik Regresyon sınıflandırması Karmaşıklık Matrisine [[92 28], [36 29]] Pozitif 120 verisetinden hepsini 92 tanesini bilmiş, 65 negatif verisetinden 29 tanesini doğru bilmiştir. Sonuca göre pozitif datasetler üzerinde daha etkili çalışmışken negatif test veriseti üzerinde yüzde 40 gibi etkili olmuş.Bu sonuç GLCM özellik çıkarımı HOG'dan daha iyi sonuç verdiğini gösterir. Ve bu sınıfın sınıf dengesizliğinden etkilenmeme olasılığı olduğunu gösterir.

Yedinci olan Beşinci olan Karar Ağaçları üzerinde Torbalama sınıflandırması Karmaşıklık Matrisine baktığımızda [[118 2], [61 4]] Pozitif 120 verisetinden 118 tanesini doğru bilmiş, 65 negatif verisetinden 4 tanesini doğru bilmiştir. Sonuca göre pozitif datasetler üzerinde daha etkili çalışmışken negatif test veriseti üzerinde etkili olamamış ve sınıf dengesizliğinden ciddi derecede etkilenmiştir.Pozitif bilinenlerin yüzdesi ile negatif bilinenlerin yüzdesi arasında uçurum vardır.Bu sınıf, Karar Ağaçlarının skorlarını iyileştirmesine rağmen Negatif test verileri üzerinde en kötü ikinci sonucu veren sınıftır ve karar ağaçlarına göre negatif veriseti üzerinde kötü performans sergilemiştir. Bu sınıfta HOG özellik çıkarımı GLCM'den daha iyi sonuç vermiştir.

Sonuncu olan Karar Ağaçları [[119 1],[63 2]] Pozitif 120 verisetinden hepsini doğru bilmiş, 65 negatif verisetinden sadece 2 doğru bilmiştir. Sonuca göre pozitif datasetler üzerinde daha etkili çalışmışken negatif test veriseti üzerinde hiç etkili olamamış ve sınıf dengesizliğinden aşırı derecede etkilenmiştir.. Negatif test verileri üzerinde en kötü sonucu veren sınıftır. Bu sınıfta HOG özellik çıkarımı GLCM'den daha iyi sonuç vermiştir.

Özet olarak;

HOG için:

İkili sınıflandırmlarda Destek Vektör Makinesi daha avantajlıdır.DVM resme yüzeysel bakar. HOG filteresi sayesinde medikal resimler yüzeysel algılanabilecek şekilde görüntünün kenarlarları kesikli yol gibi çizildi.Bu yüzden Lojistik Regresyon,Rassal Orman ve Karar Ağaçlarına göre daha avantajlı oldu. Ancak YSA resimdeki özellikleri piksel piksel daha ayrıntılı inceler. Ve Genel olarak, çok sayıda eğitim örneği olduğunda YSA, DVM 'den daha iyi performans gösterecektir, Bu yüzden YSA daha başarılı olmuş olabilir.

Rassal Orman'ın en kötü sonucu vermesinin sebebi ormanda yeterince ağaç olması durumunda, sınıflandırıcının modele uymayacağı olabilir. Yeterli ağaç olmadığı için rastgele tahmin yapmıştır.

GLCM için:

İkili sınıflandırmada pozitif ve negatif test veri seti üzerinde YSA ve DVM daha iyi sonuç ortaya koyuyor.

GLCM özellik çıkarımı verisetini karmaşıklaştırmıştır. Eğitimin aşırı uydurulmasına neden olmuştur.

Veriler üzerindeki sınıf dengesizliği Karar Ağaçları sınıflandırmasını hem AUC skorunu hemde negatif test seti üzerindeki performansını ciddi derecede olumsuz etkiliyor. Torbalama bu durumu daha da kötüleştiriyor.

Aşırı öğrenme durumlarında Rassal Orman açık ara iyi sonuç veriyor.

Adaboost, Gradient Boosting'den daha iyi sonuç veriyor.YSA'da DVM'den daha iyi sonu çveriyor.

Orijinal,GLCM ve HOG verisetlerinin YSA sınıflandırmasında karşılaştırılması

Orijinal verisetinde eğitim sırasında Model doğruluğu 0.7'i geçememiştir.Kayıp 0.6'da kalmıştır .Orijinal verisettinde YSA sınıflandırma yapılırken rastgele tahminleme yaplmış.AUC değeri 0.5'tir. Karmaşıklık Matrisi [[120 0], [65 0]]' dir. Pozitif test verisetleri üzerinde iyi performans sergilerken negatif dataset üzerinde hiç öğrenmemiştir ve sınıf dengesizliğinden ciddi etkilenmiştir.Test verisetindeki değerlendirme de Kayıp değeri=0.6449844547220178 Doğruluk değeri=0.6486486196517944'dür.(eğitim sırasındaki değerlerle benzerdir.)

HOG'de eğitim sırasında Model doğruluğu 0.9'u geçmiştir.Kayıp 0.1 altına düşmüştür ve iyi performans sergilemiştir.Verisetteki sınıflandırma tüm çalışmaların en iyi sonucunu vermiştir. Karmaşıklık Matrisine baktığımızda 185 veriden 152'sini bilmiştir. Hem dengeli şekilde bilmiş hemde en iyi performansı sergilemiştir.Test verisetindeki değerlendirme kayıp değeri değişmemiştir, Doğruluk değeri 0.7621621489524841 olarak pozitif yönde Orijinal verisetine göre ilerleme olmuştur.

GLCM'de eğitim sırasında Model doğruluğu 0.9'u geçmiştir.Kayıp 0.1 altına düşmüştür ve eğitimde iyi performans sergilemiştir.Ancak sınıflandırmada HOG sonuçlarına göre kötüdür. Test verisetindeki değerlendirme kayıp değeri 1.4406425025012042 Doğruluk değeri 0.6972972750663757'dir. Kayıp değerinin aşırı yüksektir. Sebebi ise; Loss fonksiyonu temelde modelin yaptığı tahminin, gerçek değerden (ground truth) ne kadar farklı olduğunu hesaplamaktadır. Bu nedenle iyi tahmin eden bir model oluşturmamışsa, gerçek değer (ground turth) ile tahmin edilen değer arasındaki fark yüksek olacak dolayısıyla loss değeri yüksek olacaktır.Modelin tahmin edilen değerler aşırı uydurulmuş olabilir.Aşırı uydurulmanın sebebi ise; Overfitting problemi olan modellerde yüksek varyans, düşük bias durumu görülmektedir. Bu genellikle model çok karmaşık olduğunda (yani gözlem sayısına kıyasla çok fazla özellik / değişken varsa) gerçekleşir. Bu tip modeller verilerdeki değişkenler arasındaki gerçek ilişkiler yerine eğitim verilerindeki "gürültüyü" öğrenir veya açıklar.

Kısacası;Orjinal Dataset üzerinden özellik çıkarımlar, sınıf dengesizliği söz konusu olunca performans açısından faydalı olabiliyor.Ama modelin aşırı uydurulmasına da neden olabiliyor.

6.SONUÇLAR

Bu proje özetle yapılmış olanlar;

Beyin MR Veriseti üzerinden HOG ve GLCM algoritmaları ile özellik çıkarımları yapıldı.HOG verisetini uzaktan bakıldığında dahi belirginleştirirken, GLCM verisetinin özelliklerini arttırdı ve görüntü kalitesini arttırarak modeli kompleksleştirdi.

9 farklı sınıflandırma normal skor, ROC eğrisi ile AUC skor,ortalama kare hata ve karmaşıklık matrisi ile değerlendirildi. Verisetinde sınıf dengesizliği olduğu için normal skorların sonuçları doğru vermediği gözlendi.Özellikle sınıf dengesizliğinden etkilenen sınıflarda AUC skoru ile normal skor arasında uçurum oluştu. Ortalama kare hatalar , AUC skoru ile doğru orantılı değerler verdi.Karmaşıklık matrisi ile sınıflandırmalar sınıf dengesizliğinden nasıl etkilendiği gözlemlendi.

Sınıflandırmaları ayrıntılı incelersek;

Datasetler üzerinde Lojistik Regresyon,DVM ve Rassal Orman sınıflandırılmaları kullanıldı.Bu sınıflandırmalardan en iyisini oylama yoluyla seçildi.HOG filtresinde DVM kazanırken, GLCM filtresinde Rassal Orman kazandı.

Karar Ağaçları kullanıldı ve bu algoritma üzerinden Torbalama tekniği kullanıldı.Ancak verisetinde sınıf dengesizliği olduğu için ciddi derecede etkilendi ve torbalama tekniği az olan sınıfta daha kötü performans sergiledi.

DVM algoritmasını güçlendiren tekniklerden olan Gradient Boosting ve Adaboost algoritmaları karşılaştırmalı olarak kullanıldı.Adaboost daha iyi sonuç verdiği gözlendi.

Medikal görüntü işlemlerde insan beynini örnek alan ayrıntılı şekilde resimleri inceleyen YSA sınıflandırması kullanıldı. 19 sınıflandırma işleminin en başarılısı HOG filtreli YSA modeli , AUC 0.8 yakın değeri ile birinci oldu.

YSA modeli, HOG ve GLCM filtrelerinin yararlı olup olmadığının kontrolü için orijinal , HOG ve GLCM verisetleri üzerinde sınıflandırıcı olarak kullanıldı. Sınıf dengesizliği olduğu için kesinlikle filtreleme işleminin yapılması gerektiğini ispatladı. Orijinal verisetinin eğitimi bittikten sonra her eğitim devrindeki(epoch) doğruluk değeri 0.7'ye çıkamadı. Ancak HOG ve GLCM veri setinde doğruluk değeri 0.9 üstüne çıktı. Ancak GLCM filterisinde model kompleksleştiği için aşırı uydurma oluştu.(Rassal Orman bu durumda daha iyi performans sergiledi)

Kısacası; hangi sınıfın en iyi performans sergileyeceğini önceden kestirilemez, veri seti üzerinden deneme yanılma yoluyla bulmak biraz zaman kaybettirse de en iyi sonuç elde edilen yol budur.Ve performans değerlendirmesi AUC skoru ve karmaşıklık matrisi ile en iyi sonuç vermektedir.

REFERANS:

[1]https://drive.google.com/drive/folders/1e6BShhgeW7qYqPUadG66tBxAT4XZa6l?usp=sharing

1- https://medium.com/@k.ulgen90/makine-%C3%B6%C4%9Frenimi-b%C3%B6l%C3%BCm-5-karar-a%C4%9Fa%C3%A7lar%C4%B1-c90bd7593010
2- https://medium.com/@gulcanogundur/roc-ve-auc-1fefcfc71a14
3- https://tr.linkedin.com/pulse/derin-%C3%B6%C4%9Frenme-uygulamalar%C4%B1nda-temel-kavramlar-skor-ve-%C3%A7arkac%C4%B1
4- https://www.veribilimiokulu.com/yapay-sinir-agiartificial-neural-network-nedir/
5- https://medium.com/@ekrem.hatipoglu/machine-learning-classification-logistic-regression-part-8-b77d2a61aae1
6- https://ichi.pro/tr/makine-ogrenimi-siniflandirmasi-icin-bir-konsensus-yontemi-olarak-yumusak-oylama-siniflandiricisi-40595306617155
7- https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn/#:~:text=A%20Voting%20Classifier%20is%20a,chosen%20class%20as%20the%20output.&text=Voting%20Classifier%20supports%20two%20types%20of%20votings.
8- https://dergipark.org.tr/tr/download/article-file/30026
9- https://bilkav.com/wp-content/uploads/2020/03/udemy_siniflandirma_karsilastirma.pdf
10- https://medium.com/yaz%C4%B1l%C4%B1m-ve-bili%C5%9Fim-kul%C3%BCb%C3%BC/model-performans%C4%B1n%C4%B1-de%C4%9Ferlendirmek-regresyon-48b4afec8664
11- https://scikit-learn.org/stable/modules/model_evaluation.html

ÖZGEÇMİŞ