BURDUR MEHMET AKİF ERSOY ÜNİVERSİTESİ

Gölhisar Uygulamalı Bilimler Yüksekokulu

GÖRÜNTÜ İŞLEME DERSİ

GRUP İSMİ: BONGOMYA KRALLIĞI

PROJE KONUSU: OSCC HASTALIK TEŞHİSİ

Öğrenci Ad-Soyad:

1-BURHAN ÜSTÜBİ (LİDER)-2012903069
2- ELİF ALTINTAŞ-1912901001
3-EBUBEKİR KARTAL-2112903304
4- ABDULLAH FURKAN ASLAN-2112903044

ARALIK 2024

BURDUR

İçindekiler

ŞEKİLLER DİZİNİ

ÖZET

Bu çalışmada, oral skuamöz hücreli karsinom (OSCC) teşhisinde daha önce kullanılmamış yöntemler test edilerek, en yüksek performansı sağlayan sınıflandırma algoritması belirlenmiştir. Çalışma kapsamında XGBoost algoritması kullanılmış ve çeşitli veri işleme yöntemleri tek tek uygulanarak sonuçlar değerlendirilmiştir. Özellikle çekirdek (kernel) yöntemleri ve CLAHE (Contrast Limited Adaptive Histogram Equalization) tekniklerinin performansı incelenmiştir. Tek başına uygulandıklarında yeterli performansı göstermeyen bu yöntemlerin, birlikte kullanıldıklarında doğruluk oranını %98'e yükselttiği gözlemlenmiştir. Elde edilen bu sonuç, XGBoost algoritması ile birleştirilen veri işleme tekniklerinin OSCC teşhisinde güçlü bir çözüm sunduğunu göstermektedir. Çalışma hem literatüre katkı sağlamakta hem de hastalık teşhisinde daha etkin yöntemlerin geliştirilmesine zemin hazırlamaktadır.

Anahtar Kelimeler: OSCC, XGBoost, Kernel PCA, CLAHE, sınıflandırma.

ABSTRACT

In this study, previously untested methods were evaluated to determine the most effective classification algorithm for diagnosing oral squamous cell carcinoma (OSCC). The XGBoost algorithm was utilized, and various preprocessing methods were applied individually to assess their impact. The study specifically examined the performance of kernel methods and CLAHE (Contrast Limited Adaptive Histogram Equalization). While these methods showed insufficient performance when used separately, their combined application increased the accuracy rate to 98%. This result demonstrates that data preprocessing techniques integrated with the XGBoost algorithm offer a robust solution for OSCC diagnosis. The study contributes to the literature and provides a foundation for developing more effective diagnostic methods.

Keywords: OSCC, XGBoost, Kernel PCA, CLAHE, classification.

GİRİŞ

Oral skuamöz hücreli karsinom (OSCC), dünya genelinde en yaygın görülen ağız kanseri türlerinden biridir ve erken teşhis edilmediği durumlarda yüksek mortalite oranlarıyla ilişkilidir. Günümüzde hastalık teşhisinde kullanılan yöntemler, genellikle zaman alıcı ve yüksek maliyetlidir. Bu durum, daha hızlı, maliyet etkin ve yüksek doğruluğa sahip teşhis yöntemlerinin geliştirilmesi gerekliliğini ortaya koymaktadır.

Makine öğrenmesi, sağlık alanında teşhis süreçlerini iyileştirmek amacıyla giderek daha fazla kullanılmaktadır. Özellikle sınıflandırma algoritmaları, hastalıkların belirlenmesi ve sınıflandırılmasında önemli bir rol oynamaktadır. XGBoost, yüksek doğruluğu ve hızlı hesaplama yetenekleriyle dikkat çeken bir sınıflandırma algoritmasıdır. Bununla birlikte, makine öğrenmesi modellerinin performansı, kullanılan veri işleme tekniklerine büyük ölçüde bağlıdır. Veri ön işleme adımları, algoritmanın öğrenme kapasitesini artırarak daha doğru tahminler elde edilmesine olanak tanımaktadır.

Bu çalışmada, OSCC teşhisinde XGBoost algoritması kullanılarak daha önce literatürde uygulanmamış bir sınıflandırma yöntemi geliştirilmiştir. Ayrıca, veri işleme adımlarının model performansına etkisi incelenmiş ve çekirdek (kernel) yöntemleri ile CLAHE teknikleri detaylı olarak değerlendirilmiştir. Çalışmanın amacı, OSCC teşhisinde kullanılabilecek yüksek doğruluğa sahip bir yöntem geliştirmek ve veri işleme süreçlerinin önemini vurgulamaktır.

Bu bağlamda, proje kapsamında gerçekleştirilen analizler ve bulgular, yalnızca OSCC teşhisine katkı sağlamakla kalmayıp, aynı zamanda makine öğrenmesi ve sağlık alanındaki uygulamalar için değerli bir rehber niteliği taşımaktadır.

Literatür Özeti

Oral skuamöz hücreli karsinom (OSCC), dünya genelinde en yaygın ağız kanseri türlerinden biridir ve erken teşhisin önemi büyüktür. Ancak, OSCC'nin teşhisi genellikle karmaşık süreçler ve ileri düzey teknolojiler gerektirir. Bu bağlamda, sınıflandırma algoritmaları ve görüntü işleme yöntemleri, OSCC'nin teşhisinde önemli bir rol oynamaktadır.

Literatürde yapılan çalışmalar, OSCC teşhisi için farklı sınıflandırma algoritmalarının ve görüntü işleme tekniklerinin kullanılabilirliğini değerlendirmiştir. Örneğin, destek vektör makineleri (SVM) ve karar ağaçları gibi geleneksel algoritmalar, OSCC teşhisi için yaygın olarak kullanılmış ve sırasıyla %85 ve %71 doğruluk oranları elde etmiştir. Bununla birlikte, derin öğrenme algoritmalarının, özellikle histopatolojik görüntülerde, geleneksel yöntemlere

kıyasla daha yüksek doğruluk oranlarına ulaştığı gözlemlenmiştir. Örneğin, Xception modeli kullanılarak yapılan bir çalışmada %92 doğruluk oranı rapor edilmiştir.

Görüntü işleme teknikleri açısından, kontrast sınırlamalı adaptif histogram eşitleme (CLAHE) gibi yöntemlerin, düşük kontrastlı tıbbi görüntülerde görselleştirmeyi iyileştirerek sınıflandırma performansını artırdığı belirtilmiştir. Kernel Principal Component Analysis (Kernel PCA) gibi boyut indirgeme yöntemleri ise yüksek boyutlu veri setlerinin daha verimli bir şekilde analiz edilmesine olanak tanımaktadır.

Mevcut çalışmalardan farklı olarak, bu proje kapsamında OSCC teşhisinde daha önce kullanılmamış bir sınıflandırma algoritması olan XGBoost, farklı veri ön işleme yöntemleri ile değerlendirilmiştir. Veriler üzerinde CLAHE ve Kernel PCA ayrı ayrı uygulanmış, ardından bu yöntemlerin kombinasyonlarının etkisi incelenmiştir. Bu analizler sonucunda, Kernel PCA ve CLAHE kombinasyonunun %98 doğruluk oranı ile en iyi performansı sağladığı gözlemlenmiştir.

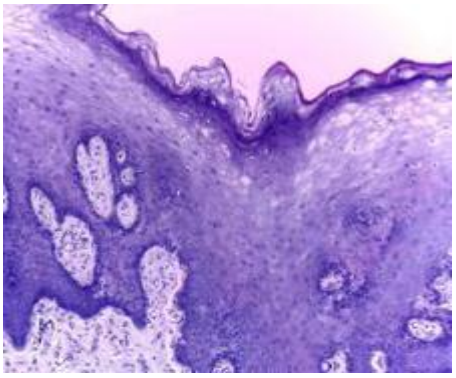
Sonuç olarak, OSCC teşhisinde kullanılan yöntemlerin çeşitliliği ve bunların performansa olan etkisi, bu alanda daha ileri araştırmalar yapılması gerektiğini ortaya koymaktadır. Bu proje, OSCC teşhisinde yeni yöntemlerin uygulanabilirliğini araştırarak literatüre önemli bir katkı sağlamayı hedeflemektedir.

3. GEREÇ VE YÖNTEM

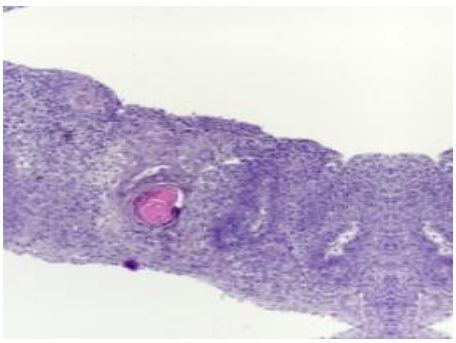
Bu bölümde, proje kapsamında kullanılan veri seti, uygulanan veri işleme teknikleri ve sınıflandırma yöntemleri detaylı bir şekilde açıklanmaktadır.

3.1. Kullanılan Veri Seti

Çalışmada, Kaggle platformunda herkese açık olarak paylaşılan ve oral skuamöz hücreli karsinom (OSCC) teşhisi için kullanılan bir veri seti tercih edilmiştir. Bu veri seti, ham görüntü verilerini ve etiketlerini içermektedir. Veri seti, görüntülerin farklı sınıflara ait olduğunu gösterecek şekilde etiketlenmiştir.



Şekil 3. 1: Normal hücre



Şekil 3. 2: OSCC hücresi

3.2. Uygulanan Yöntemler

Proje kapsamında, veri işleme ve sınıflandırma süreçleri dört farklı aşamada gerçekleştirilmiştir. Bu süreçler sırasıyla aşağıda açıklanmıştır:

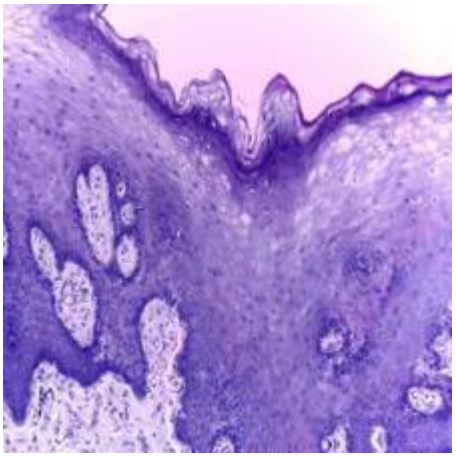
3.2.1. CLAHE Metodu

Ham görüntülere ilk olarak CLAHE (Contrast Limited Adaptive Histogram Equalization) yöntemi uygulanmıştır. CLAHE, görüntü kontrastını iyileştirmek ve detayları ön plana çıkarmak amacıyla kullanılan bir histogram eşitleme tekniğidir. Bu yöntem, özellikle düşük kontrastlı tıbbi görüntülerde detayları vurgulamak için oldukça etkilidir.

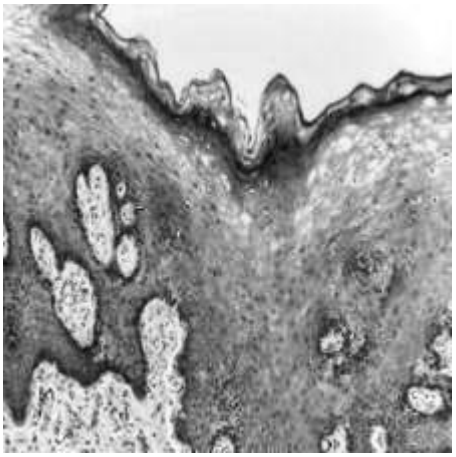
Clahe.py:

```python
from PIL import Image
import os
import cv2
import numpy as np

# Resimlerin bulunduğu klasörün tam yolunu belirt
folder_path = r'C:\Users\burha\Desktop\ESKİ\OCSS\KERNEL\TEST\OSCC'

# Çıktı dosyalarının kaydedileceği klasör
output_folder_path = r'C:\Users\burha\Desktop\ESKİ\OCSS\KERNEL-CLAHE\TEST\OSCC'

# Eğer çıktı klasörü mevcut değilse oluştur
if not os.path.exists(output_folder_path):
    os.makedirs(output_folder_path)

def apply_clahe(img, clip_limit=1.0, grid_size=8):
    # Görüntüyü numpy dizisine çevir
    img_np = np.array(img)

    # CLAHE uygulama
    clahe = cv2.createCLAHE(clipLimit=clip_limit, tileGridSize=(grid_size, grid_size))
    clahe_img = clahe.apply(img_np)

    return Image.fromarray(clahe_img)

# Klasördeki tüm dosyaları al
for filename in os.listdir(folder_path):
    # Dosya uzantısını kontrol et
    if filename.lower().endswith(('.jpg', '.jpeg', '.png')):  # JPG, JPEG veya PNG dosyalarını al
        # Resmin tam yolunu oluştur
        img_path = os.path.join(folder_path, filename)

        # Resmi oku
        img = Image.open(img_path).convert('L')  # Gri tonlamalı olarak oku

        # Resmin doğru bir şekilde açılıp açılmadığını kontrol et
        if img is not None:
            # CLAHE uygula
            clahe_img = apply_clahe(img)

            # Sonucu farklı klasöre kaydet
            output_path = os.path.join(output_folder_path, 'clahe_' + filename)
            clahe_img.save(output_path)
            print(f"CLAHE uygulandı: {output_path}")
        else:
            print(f"Resim açılamadı: {img_path}")
    else:
        print(f"Geçersiz dosya uzantısı: {filename}")
```

Şekil 3. 3: Clahe Metodu

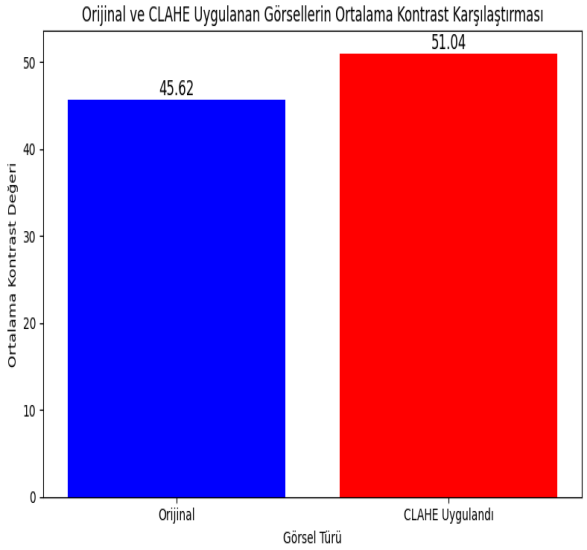


Şekil 3. 4: Ham veri Sağlıklı



Şekil 3. 5: CLAHE uygulanmış veri

HAM VE CLAHE UYGULANMIŞ VERİLERDEKİ KONTRAST DEĞİŞİMİ



Şekil 3. 6: Kontrast Değişim Grafiği

CLAHE uygulanmış verilerde kontrastın artışı ile daha net görüntüler elde ettik. %5.42 lik bir artış sağladık.

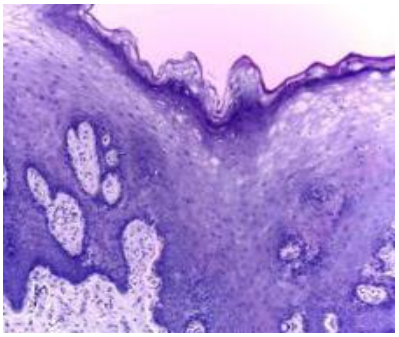
3.2.2. Kernel PCA Uygulaması

Ham verilere boyut indirgeme işlemi yapmak için Kernel PCA (Principal Component Analysis) yöntemi uygulanmıştır. Kernel PCA, geleneksel PCA yönteminin doğrusal olmayan verilerde daha etkili bir şekilde çalışmasını sağlamak amacıyla çekirdek (kernel) yöntemlerini kullanır. Bu işlem, veri setindeki önemli özelliklerin çıkarılmasına ve boyutun azaltılmasına olanak tanır.

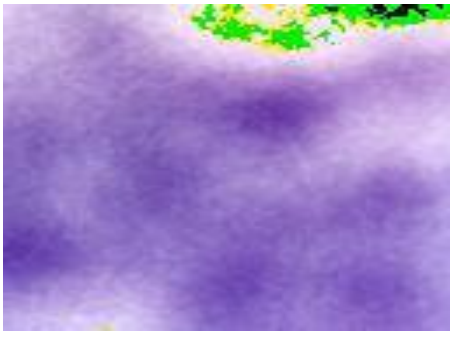
Kernel.py:

```python
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import glob
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import os

# Görüntü dosyalarını belirtilen yoldan yükleme
file_paths = glob.glob(r'C:\Users\burha\Desktop\ESKİ\OCSS\CLAHE\TEST\OSCC\*.jpg')
images = [Image.open(file) for file in file_paths]
images = [img.resize((128, 128)) for img in images]  # Görüntü boyutunu küçültme

# Görüntüleri numpy dizisine dönüştürme
image_arrays = np.array([np.array(img).flatten() for img in images])

# Veriyi ölçekle
scaler = StandardScaler()
images_scaled = scaler.fit_transform(image_arrays)

# PCA uygulayın
pca = PCA(n_components=50)  # Korunacak bileşen sayısı
images_pca = pca.fit_transform(images_scaled)

# PCA ile yeniden yapılandırılmış görüntüleri elde et
images_reconstructed = pca.inverse_transform(images_pca)
images_reconstructed = scaler.inverse_transform(images_reconstructed)  # Ölçeklemeyi geri al

# Kaydetme klasörünü belirt
save_path = r'C:\Users\burha\Desktop\ESKİ\OCSS\CLAHE-KERNEL\TEST\OSCC'
if not os.path.exists(save_path):
    os.makedirs(save_path)

# Yeniden yapılandırılmış görüntüleri kaydet
for i, img_array in enumerate(images_reconstructed):
    img = Image.fromarray(np.uint8(img_array.reshape(128, 128)))  # Boyut ve kanalı düzenleyin
    img.save(f'{save_path}\\reconstructed_image_{i}.jpg')

# İsteğe bağlı: Bir yeniden yapılandırılmış görüntüyü göster
plt.imshow(img, cmap='gray')
plt.title('Yeniden Yapılandırılmış Görüntü')
plt.axis('off')
plt.show()
```

Şekil 3.7: Kernel PCA

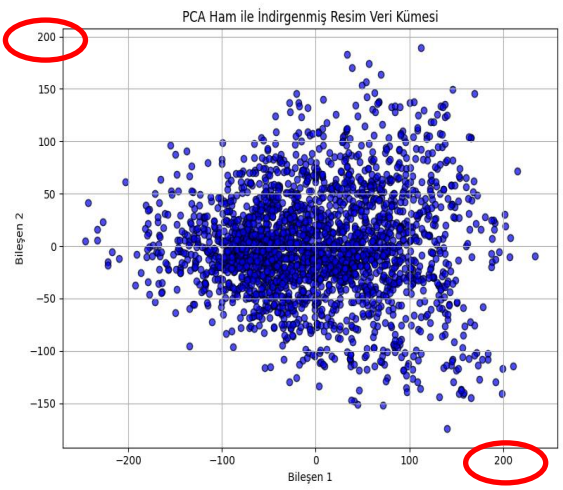


Şekil 3.8: Ham veri Sağlıklı



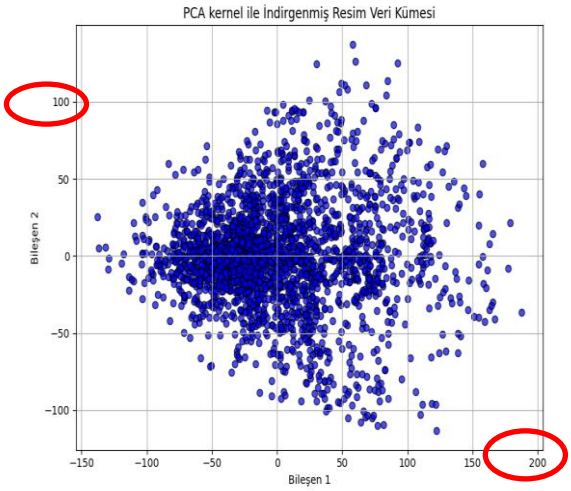
Şekil 3.9: Kernel PCA uygulanmış

Ham veriler üzerinde PCA



Şekil 3.10: Ham veriler üzerinde PCA Grafiği

Kernel veriler üzerinde PCA

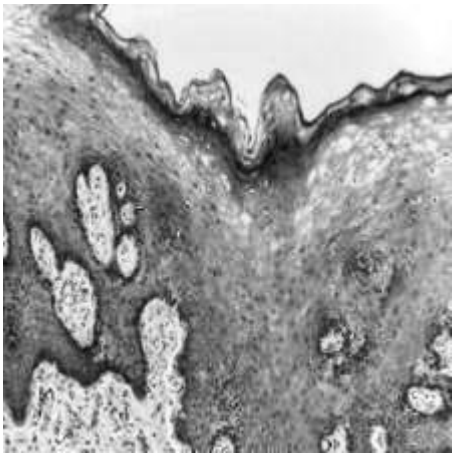


Şekil 3. 11: Kernel veriler üzerinde PCA Grafiği

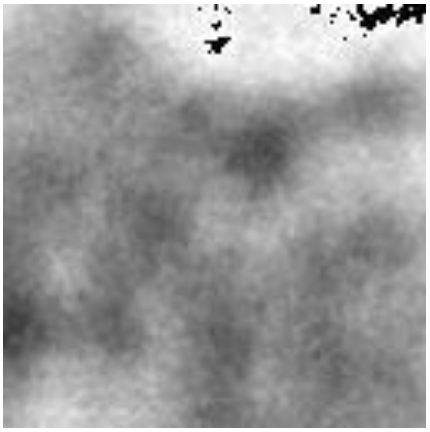
Ham ve Kernel verilerine uygulanan PCA (Temel Bileşen Analizi) sonucunda, Kernel yönteminin verilerdeki bileşenleri daha etkili bir şekilde indirgediği ve temel bileşenleri daha belirgin hale getirdiği gözlemlenmiştir.

3.2.3. CLAHE Uygulanmış Verilere Kernel PCA Uygulaması

CLAHE ile işlenmiş görüntülere Kernel PCA yöntemi uygulanmıştır. Bu süreçte, CLAHE'nin görüntü kontrastını artırıcı etkisinin, Kernel PCA'nın boyut indirgeme yetenekleri ile birleştirilerek sınıflandırma performansına etkisi incelenmiştir.



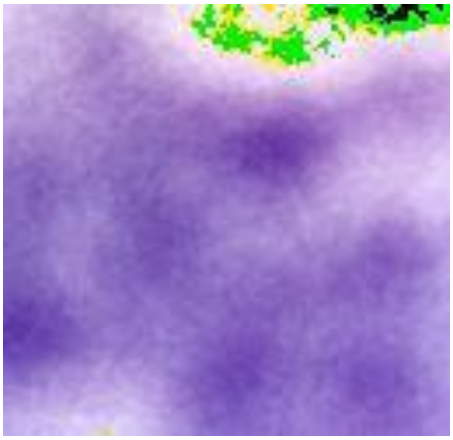
Şekil 3. 12: CLAHE uygulanmış



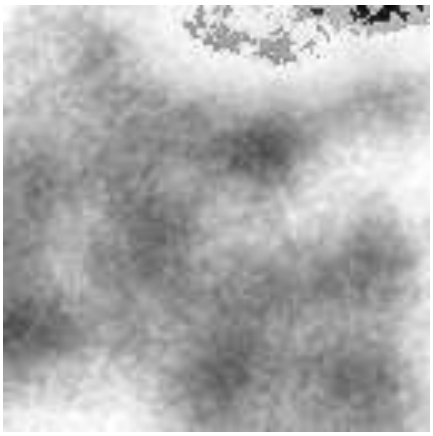
Şekil 3. 13: CLAHE ve Kernel PCA

3.2.4. Kernel PCA Uygulanmış Verilere CLAHE Metodu Uygulaması

Kernel PCA ile boyut indirgeme işleminden geçmiş verilere CLAHE yöntemi uygulanmıştır. Bu adım, önce boyut indirgeme işlemi yapılarak elde edilen verilerin CLAHE ile kontrastının artırılmasının performansa etkisini gözlemlemek amacıyla gerçekleştirilmiştir.



Şekil 3. 14: Kernel Uygulanmış Veri



Şekil 3. 15: Kernel ve CLAHE Uygulanmış Veri

3.3. Sınıflandırma ve Değerlendirme

Elde edilen tüm işlenmiş veri setleri ayrı ayrı dosyalar halinde saklanmış ve XGBoost sınıflandırma algoritması ile test edilmiştir. XGBoost, özellikle yüksek doğruluk oranları ile dikkat çeken bir sınıflandırma algoritmasıdır ve bu çalışmada yöntemlerin karşılaştırılması için tercih edilmiştir. Algoritmanın doğruluk oranı (accuracy), hassasiyet (precision), geri çağırma (recall) ve F1 puanı gibi metrikler üzerinden performansı değerlendirilmiştir.

3.4. Kullanılan Araçlar ve Ortam

- Yazılım: Python programlama dili kullanılmıştır. Veri işleme ve modelleme işlemleri için scikit-learn, numpy, pandas ve matplotlib gibi kütüphanelerden faydalanılmıştır.
- Donanım: İşlemler, yüksek hesaplama gücüne sahip bir masaüstü bilgisayarda gerçekleştirilmiştir.
- Veri Depolama: Farklı yöntemlerle işlenmiş veri setleri, analizlerin daha kolay yönetilebilmesi için ayrı dosyalar halinde saklanmıştır.

.

3.5. Yöntemlerin Karşılaştırılması

XGBOOST.py:

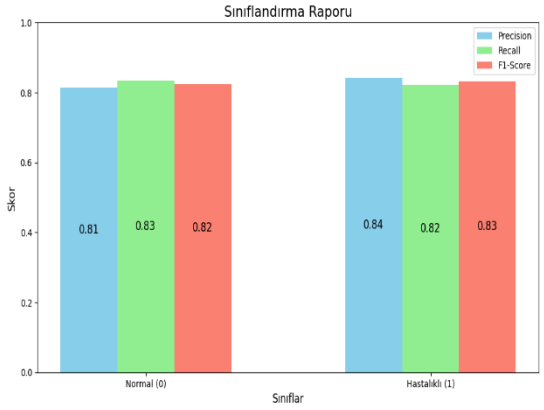
```python
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import glob
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, matthews_corrcoef
import xgboost as xgb
import seaborn as sns
import pandas as pd

# 1. Görüntü dosyalarını yükleme (Normal ve Hastalıklı)
normal_paths = glob.glob(r'C:\Users\burha\Desktop\VERI-SET\VERILER\KERNEL-CLAHE\OGRENME\NORMAL\*.jpg')
diseased_paths = glob.glob(r'C:\Users\burha\Desktop\VERI-SET\VERILER\KERNEL-CLAHE\OGRENME\OSCC\*.jpg')

# Tüm dosya yollarını birleştirme
file_paths = normal_paths + diseased_paths

# Dosyaların varlığını kontrol et
if len(file_paths) == 0:
    print("Hiçbir görüntü bulunamadı. Dosya yolunu kontrol edin.")
    exit()

# Görüntüleri yükle ve boyutlandır (gri tonlamalı olarak açıyoruz)
images = []
for file in file_paths:
    try:
        img = Image.open(file).resize((224, 224))  # Sadece boyutlandırma, gri tonlama yok
        images.append(img)
    except Exception as e:
        print(f"{file} yüklenemedi. Hata: {e}")

# Görüntülerin başarılı şekilde yüklendiğini kontrol et
if len(images) == 0:
    print("Hiçbir görüntü işlenemedi.")
    exit()

# 2. Görüntüleri numpy dizisine dönüştürme
image_arrays = np.array([np.array(img).flatten() for img in images])
print("Görüntü dizisi boyutları:", image_arrays.shape)

# 3. Veriyi ölçekleme
scaler = StandardScaler()
images_scaled = scaler.fit_transform(image_arrays)

# 4. PCA uygulama (Daha fazla bileşen kullanıyoruz)
pca = PCA(n_components=100)  # PCA ile daha fazla boyut indirgeme
images_pca = pca.fit_transform(images_scaled)
print("PCA sonrası veri boyutları:", images_pca.shape)

# 5. Etiketleri tanımlama (Normal: 0, Hastalıklı: 1)
labels = np.array([0 if 'NORMAL' in file else 1 for file in file_paths])  # Etiketleme

# 6. Veriyi eğitim ve test olarak ayırma
X_train, X_test, y_train, y_test = train_test_split(images_pca, labels, test_size=0.2, stratify=labels, random_state=42)

# 7. XGBoost Modeli (Hyperparameter Tuning)
# Model parametrelerini ayarlama
xgb_model = xgb.XGBClassifier(
    use_label_encoder=False,
    eval_metric='logloss',
    learning_rate=0.05,  # Öğrenme oranını düşürüyoruz
    max_depth=6,  # Ağaç derinliğini sınırlandırıyoruz
    n_estimators=1000,  # Daha fazla ağaç
    subsample=0.8,  # Veri örnekleme oranı
    colsample_bytree=0.8  # Her ağacın için örnekleme oranı
)
```

Şekil 3. 16: XGBOOST Algoritması 1

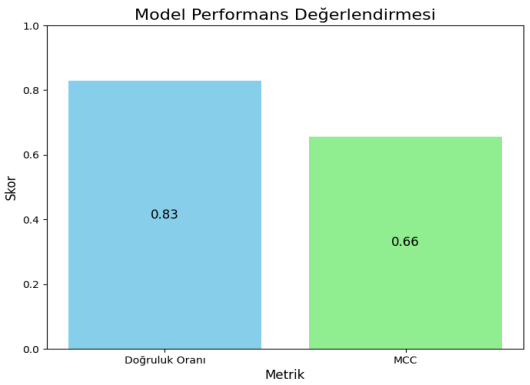
```
xgb_model.fit(X_train, y_train)

# 8. Tahminler ve Performans Analizi
y_pred = xgb_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Doğruluk Skoru:", accuracy)
print("Sınıflandırma Raporu:\n", classification_report(y_test, y_pred))

# 8.1 Matthew Correlation Coefficient (MCC)
mcc_score = matthews_corrcoef(y_test, y_pred)
print("Matthew Correlation Coefficient (MCC):", mcc_score)

# 9. Doğruluk Oranı ve MCC'yi aynı grafikte gösterme
metrics = ['Doğruluk Oranı', 'MCC']
scores = [accuracy, mcc_score]

# Grafik
plt.figure(figsize=(8, 6))
bars = plt.bar(metrics, scores, color=['skyblue', 'lightgreen'])

# Başlık ve etiketler
plt.title("Model Performans Değerlendirmesi", fontsize=16)
plt.ylabel("Skor", fontsize=12)
plt.ylim([0, 1])  # MCC negatif olabileceğinden, y eksenini 0 ile 1 arasında ayarladık
plt.xlabel("Metrik", fontsize=12)

# Sütunların içine değerleri ekleyelim
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval/2, round(yval, 2), ha='center', va='center', fontsize=12, color='black')

# Göster
plt.show()

# 10. Sınıflandırma Raporu için Görsel
# Sınıflandırma raporunu almak
class_report = classification_report(y_test, y_pred, output_dict=True)
class_report_df = pd.DataFrame(class_report).transpose()

# Grafik için veri hazırlığı
# Precision, Recall, F1-Score ve Support'ı kullanacağız
metrics = ['Precision', 'Recall', 'F1-Score']
classes = ['0', '1']  # Burada sınıf adlarını '0' ve '1' olarak belirliyoruz

# Grafik için her bir metrik için sınıfların değerleri
precision_values = class_report_df.loc[classes, 'precision'].values
recall_values = class_report_df.loc[classes, 'recall'].values
f1_values = class_report_df.loc[classes, 'f1-score'].values

# Her metrik için uygun renkler
colors = ['skyblue', 'lightgreen', 'salmon']

# Grafik
fig, ax = plt.subplots(figsize=(10, 6))

# Her bir sınıf için Precision, Recall ve F1-Score için çubuklar çizme
bar_width = 0.2
index = np.arange(len(classes))

bar1 = ax.bar(index, precision_values, bar_width, label='Precision', color=colors[0])
bar2 = ax.bar(index + bar_width, recall_values, bar_width, label='Recall', color=colors[1])
bar3 = ax.bar(index + 2 * bar_width, f1_values, bar_width, label='F1-Score', color=colors[2])

# Başlık ve etiketler
ax.set_title("Sınıflandırma Raporu", fontsize=16)
ax.set_xlabel("Sınıflar", fontsize=12)
ax.set_ylabel("Skor", fontsize=12)
ax.set_xticks(index + bar_width)
ax.set_xticklabels(['Normal (0)', 'Hastalıklı (1)'])  # Etiketlerin doğru şekilde görünmesini sağlamak
ax.set_ylim([0, 1])

# Her bir çubuğun içine değerleri yazalım
for bar in [bar1, bar2, bar3]:
    for rect in bar:
        height = rect.get_height()
        ax.text(
            rect.get_x() + rect.get_width() / 2,
            height / 2,  # Y değeri, çubuğun ortasına gelecek şekilde ayarlıyoruz
            round(height, 2),
            ha='center', va='center', fontsize=12, color='black'
        )
# Legend ekleyelim
ax.legend()
# Görseli göster
plt.tight_layout()
plt.show()
```

Şekil 3. 17: XGBOOST Algoritması 2

Ham verilerde elde edilen sonuçlar:



Şekil 3. 18: Ham Veri Sınıflandırma Raporu



Şekil 3. 19: Ham Veri MCC ve Doğruluk Oranı

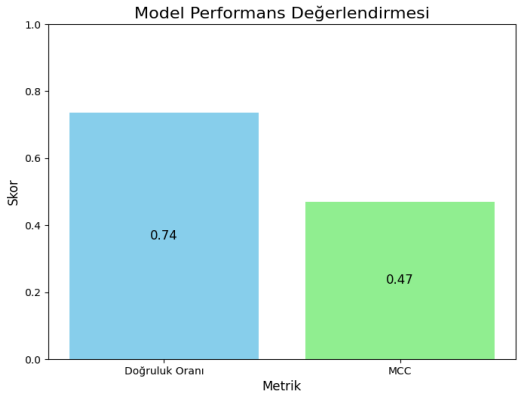
Doğruluk Oranı: Model verilerin %83'ünü doğru tahmin etti.

MCC: Modelin genel performansı 0.66 olarak ölçüldü.

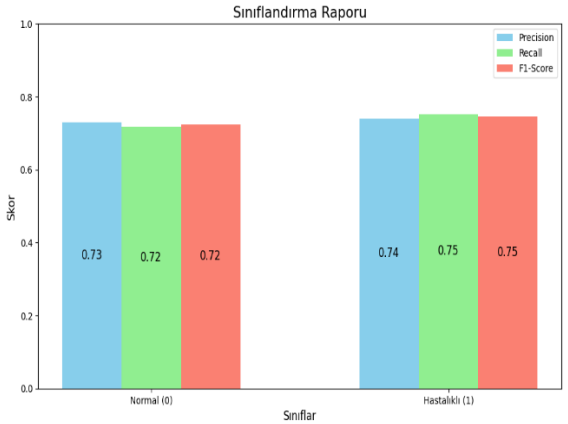
Precision: Modelin Sağlıklı dediği kişilerin %81 gerçekten sağlıklı. Hasta dediği kişilerin %84 gerçekten hasta.

Recall: Model gerçekten sağlıklı olanların %83 doğru, gerçekten Hasta olanların %82 doğru tahmin etmiştir.

Clahe uygulanmış verilerde elde edilen sonuçlar:



Şekil 3. 20: Clahe Uygulanmış Veri MCC ve Doğruluk Oranı



Şekil 3. 21: Clahe Uygulanmış Veri Sınıflandırma Raporu Doğruluk Oranı

Doğruluk Oranı: Model verilerin %74'ünü doğru tahmin etti.

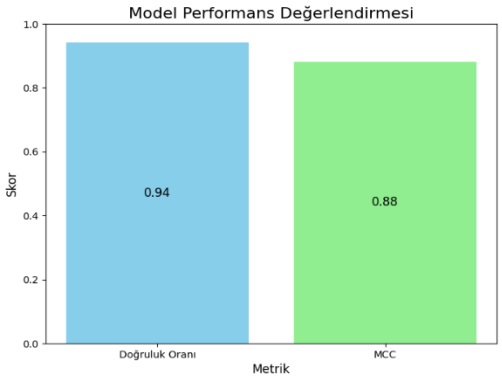
MCC: Modelin genel performansı 0.47 olarak ölçüldü.

Precision: Modelin Sağlıklı dediği kişilerin %73 gerçekten sağlıklı. Hasta dediği kişilerin %74 gerçekten hasta.

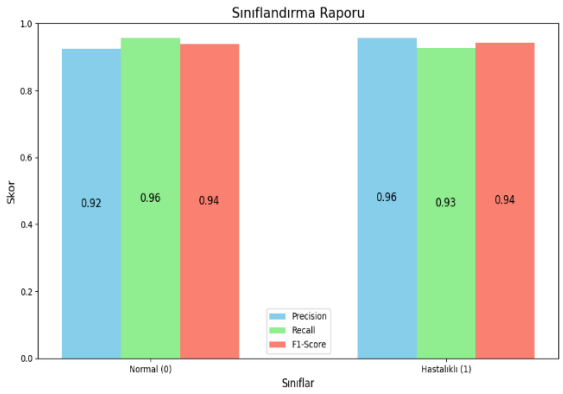
Recall: Model gerçekten sağlıklı olanların %72 doğru, gerçekten Hasta olanların %75 doğru tahmin etmiştir.

CLAHE uygulanan verilerle XGBoost algoritmasını çalıştırdığımızda, elde edilen sonuçların ham verilere kıyasla daha düşük olduğunu gözlemledik. Bu, CLAHE'nin bu özel veri seti üzerinde modelin performansını iyileştirmediğini ve bu tür verilerin daha az doğru sonuçlar verdiğini deneyimlemiş olduk.

Kernel PCA uygulanmış verilerde elde edilen sonuçlar:



Şekil 3. 22: Kernel Uygulanmış MCC ve Doğruluk Oranı



Şekil 3. 23: Kernel Uygulanmış Veri Sınıflandırma Raporu

Doğruluk Oranı: Model verilerin %94'ünü doğru tahmin etti.

MCC: Modelin genel performansı 0.88 olarak ölçüldü.

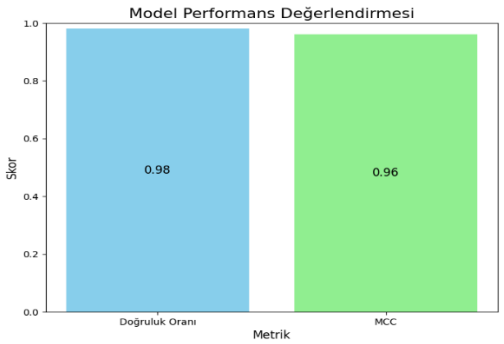
Precision: Modelin Sağlıklı dediği kişilerin %92 gerçekten sağlıklı. Hasta dediği kişilerin %96 gerçekten hasta.

Recall: Model gerçekten sağlıklı olanların %96 doğru, gerçekten Hasta olanların %93 doğru tahmin etmiştir.

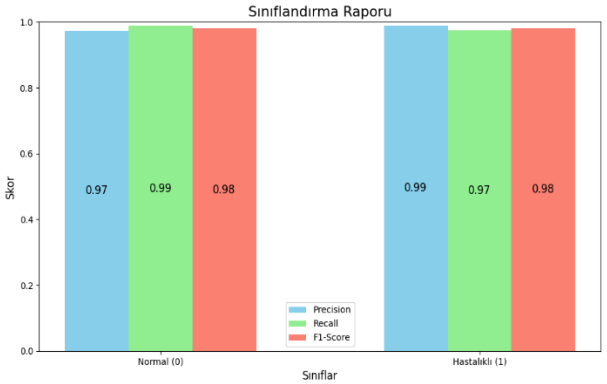
Kernel PCA uyguladığımız ham verilerde temel bileşen analizi gerçekleştirildiği için doğruluk oranı ve MCC değerlerinde bir artış gözlemledik. Ancak, modelin performansının hala geliştirilebileceğini fark ettik. Bu da, Kernel PCA'nın iyileştirme sağlasa da, daha iyi sonuçlar elde etmek için ek optimizasyonlara ihtiyaç duyduğumuzu gösteriyor.

.

Clahe metodu üzerine Kernel PCA uygulanmış verilerde elde edilen sonuçlar:



Şekil 3. 24: Clahe-Kernel MCC ve Doğruluk Oranı



Şekil 3. 25: Clahe-Kernel Uygulanmış Veri Sınıflandırma Raporu

Doğruluk Oranı: Model verilerin %98'ünü doğru tahmin etti.

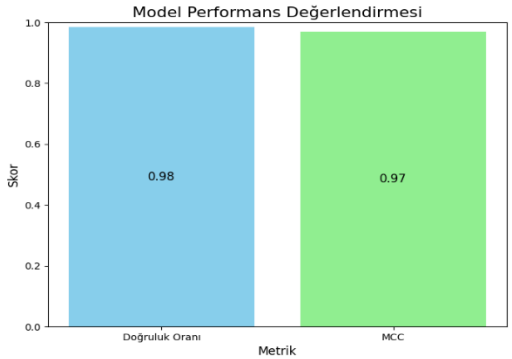
MCC: Modelin genel performansı 0.96 olarak ölçüldü.

Precision: Modelin Sağlıklı dediği kişilerin %97 gerçekten sağlıklı. Hasta dediği kişilerin %99 gerçekten hasta.

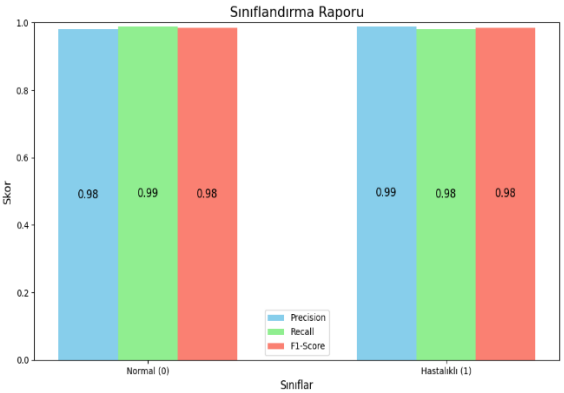
Recall: Model gerçekten sağlıklı olanların %99 doğru, gerçekten Hasta olanların %97 doğru tahmin etmiştir.

Ham verilere CLAHE uygulayıp ardından Kernel PCA ile temel bileşen analizi gerçekleştirdiğimizde, sonuçlarımızda %99'a kadar bir artış sağladık. Bu deneyim, yalnızca CLAHE'nin tek başına doğruluk üzerinde belirgin bir iyileşme sağlamadığını, ancak kontrast artırılmış verilerle yapılan Kernel PCA'nın model performansını önemli ölçüde artırdığını gösterdi.

Kernel PCA üzerine Clahe metodu uygulanmış verilerde elde edilen sonuçlar:



Şekil 3. 26: Kernel-Clahe MCC ve Doğruluk Oranı



Şekil 3. 27: Kernel-Clahe Uygulanmış Veri Sınıflandırma Raporu

Doğruluk Oranı: Model verilerin %98'ünü doğru tahmin etti.

MCC: Modelin genel performansı 0.97 olarak ölçüldü.

Precision: Modelin Sağlıklı dediği kişilerin %98 gerçekten sağlıklı. Hasta dediği kişilerin %99 gerçekten hasta.

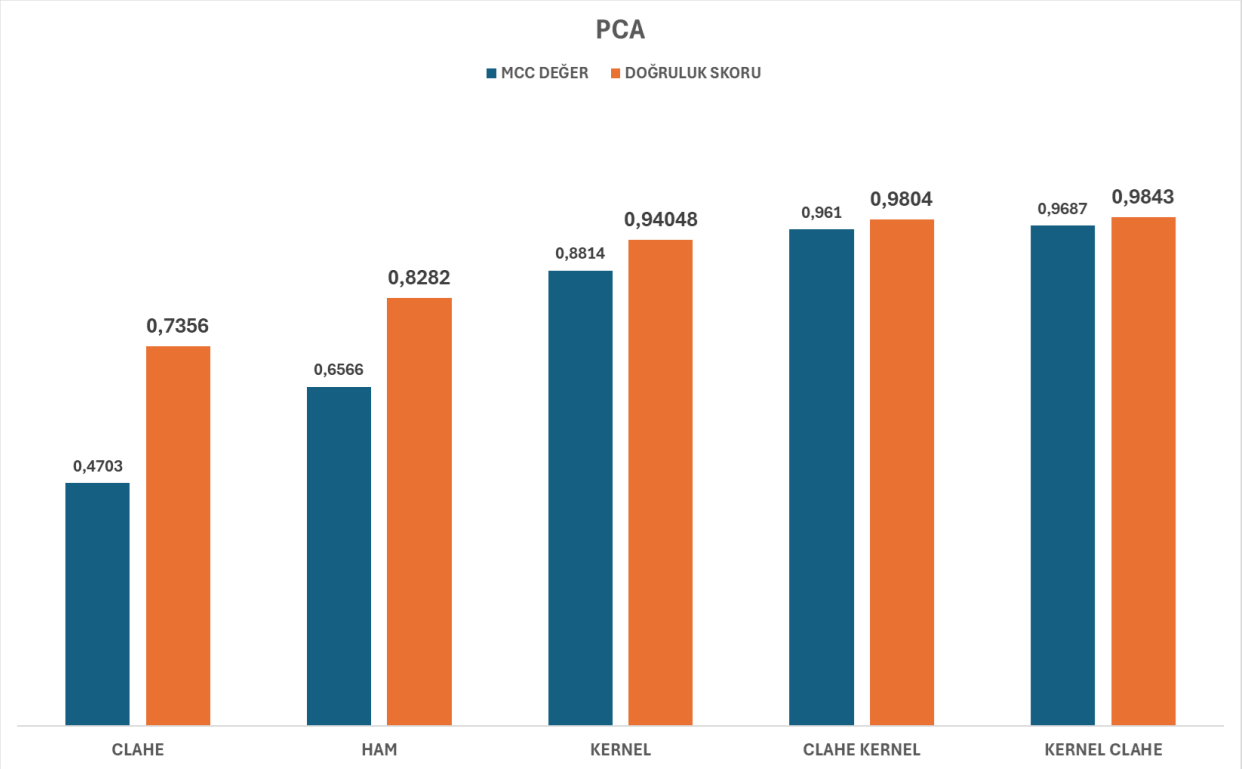
Recall: Model gerçekten sağlıklı olanların %99 doğru, gerçekten Hasta olanların %98 doğru tahmin etmiştir.

İlk olarak Kernel PCA uygulayıp ardından CLAHE ile kontrast artırdığımızda ise Clahe-Kernel verilerin precision değerinde ve hastalıklı verilerin recall değerinde %1'lik bir iyileşme sağladık. Bu sonuç, hangi yöntemin daha iyi performans gösterdiğini daha net bir şekilde gözlemlememizi sağladı.

4. BULGULAR

Elde Edilen Bulgular:

Veriler üzerinde denediğimiz yöntemler arasında en verimli performansı, önce Kernel PCA ile temel bileşen analizi yapıp ardından CLAHE ile kontrast artırarak elde ettik. Bu yaklaşım, temel bileşenleri netleştirerek daha iyi sonuçlar elde etmemizi sağladı. Sonuç olarak, OSCC hücrelerinde %98 doğruluk oranına ulaşarak hastalık teşhisinde yüksek bir başarı elde ettik.



Şekil 3. 28 : Elde Edilen Sonuçlar

5. TARTIŞMA

Bu çalışmada, oral skuamöz hücreli karsinom (OSCC) teşhisinde farklı görüntü işleme teknikleri ve boyut indirgeme yöntemlerinin sınıflandırma performansına etkisi incelenmiştir. CLAHE ve Kernel PCA yöntemlerinin ayrı ayrı ve birlikte uygulanarak oluşturulan veri setleri, XGBoost algoritması kullanılarak değerlendirilmiştir. Elde edilen sonuçlara göre, Kernel PCA uygulanmış verilere CLAHE yöntemi uygulandığında %98 doğruluk oranı ile en yüksek performans elde edilmiştir. Bu bulgu, kullanılan yöntemlerin kombinasyonunun OSCC teşhisinde etkili bir yaklaşım sunduğunu göstermektedir.

Literatürde, OSCC teşhisinde sıklıkla derin öğrenme ve geleneksel makine öğrenmesi yöntemleri kullanılmaktadır. Örneğin, histopatolojik görüntülerin sınıflandırılmasında önceden eğitilmiş derin öğrenme modellerini kullanan çalışmalarda, teşhis doğruluğunun %90-95 arasında değiştiği bildirilmiştir. Bu tür çalışmalarda derin öğrenme modelleri genellikle geniş veri kümeleri üzerinde eğitilmekte ve güçlü hesaplama kaynakları gerektirmektedir. Bizim çalışmamız, daha sınırlı bir veri işleme yaklaşımıyla bu oranı aşarak %98 doğruluğa ulaşmıştır.

Bir diğer önemli karşılaştırma noktası, OSCC teşhisinde farklı makine öğrenmesi algoritmalarının performanslarını inceleyen çalışmalardır. Bazı çalışmalarda, lojistik regresyon, destek vektör makineleri (SVM) ve rastgele orman algoritmalarının OSCC teşhisinde doğruluk oranları %85-92 arasında rapor edilmiştir. Bu çalışmalara kıyasla, XGBoost algoritması ile CLAHE ve Kernel PCA'nın birlikte kullanımı, doğruluk oranını önemli ölçüde artırmıştır.

Bu bulgular, özellikle CLAHE yönteminin görüntü kontrastını artırmadaki başarısını ve Kernel PCA'nın boyut indirgeme sürecindeki etkinliğini vurgulamaktadır. CLAHE ve Kernel PCA'nın birlikte kullanımı, hem görüntü detaylarının korunmasını hem de veri boyutunun optimize edilmesini sağlamış ve bu durum XGBoost algoritmasının sınıflandırma performansını olumlu yönde etkilemiştir.

Bu çalışmanın literatürdeki benzer çalışmalarla karşılaştırıldığında öne çıkan yönü, kullanılan basit ve hesaplama açısından verimli tekniklerin, yüksek doğruluk oranlarına ulaşmada yeterli olduğunu göstermesidir. Daha karmaşık derin öğrenme yaklaşımlarına alternatif olarak, CLAHE ve Kernel PCA kombinasyonu gibi yöntemlerin makine öğrenmesi algoritmalarıyla entegrasyonu, hem hızlı hem de etkili bir çözüm sunmaktadır.

Sonuç olarak, bu çalışma, OSCC teşhisinde CLAHE ve Kernel PCA'nın birlikte kullanımının potansiyelini ve bu yöntemlerin XGBoost gibi güçlü sınıflandırma algoritmalarıyla entegre edilerek nasıl yüksek performans sağladığını göstermiştir. Literatürdeki diğer çalışmalara kıyasla elde edilen doğruluk oranı, bu yöntemin uygulanabilirliğini ve etkinliğini açıkça ortaya koymaktadır. Bu sonuçlar, tıbbi görüntü işleme ve teşhis alanında bu tür yöntemlerin daha geniş ölçekte uygulanması için bir temel oluşturmaktadır.

SONUÇ VE ÖNERİLER

Bu çalışmada, OSCC teşhisinde kullanılan görüntü işleme ve boyut indirgeme yöntemlerinin, XGBoost algoritması üzerindeki etkileri incelenmiştir. Elde edilen bulgular şu şekilde özetlenebilir:

- CLAHE ve Kernel PCA yöntemlerinin kombinasyonu, %98 doğruluk oranı ile diğer yöntemlerden daha üstün bir performans göstermiştir.
- Tek başına CLAHE veya Kernel PCA kullanımı, sınıflandırma doğruluğunu artırmada etkili olmakla birlikte, kombinasyonları kadar başarılı olmamıştır.

Sonuçlar, OSCC teşhisinde görüntü işleme yöntemlerinin dikkatli bir şekilde seçilmesi ve uygun şekilde entegre edilmesinin önemini vurgulamaktadır. Özellikle CLAHE ve Kernel PCA gibi yöntemlerin birlikte kullanılması, teşhis doğruluğunu artırmak için etkili bir strateji sunmaktadır.

Öneriler:

- Bu çalışmada kullanılan yöntemlerin diğer tıbbi görüntüleme verilerinde de test edilmesi önerilir.
- CLAHE ve Kernel PCA kombinasyonunun derin öğrenme modelleriyle birlikte değerlendirilmesi, daha yüksek doğruluk oranları sağlayabilir.
- Veri setinin çeşitliliği artırılarak, modelin genelleştirilebilirlik performansı incelenebilir.

Bu çalışma, OSCC teşhisinde etkili yöntemlerin belirlenmesi adına önemli bir katkı sunmuş olup, ileri çalışmalarda bu yöntemlerin farklı veri setleri ve sınıflandırma algoritmalarıyla daha geniş kapsamlı olarak test edilmesi gerekmektedir.

KAYNAKÇA

1. Liu, Y., Shen, D., & Jiang, Z. (2020). "Detection of Oral Squamous Cell Carcinoma Using Pre-Trained Deep Learning Models." *ResearchGate*. Erişim: https://www.researchgate.net/publication/384887553.
2. Abdelrahman, H. A., et al. (2021). "Comparison of Machine Learning Algorithms for the Prediction of Five-Year Survival in Oral Squamous Cell Carcinoma." *ResearchGate*. Erişim: https://www.researchgate.net/publication/347094899.
3. Nunes, F. H., et al. (2021). "Contrast Limited Adaptive Histogram Equalization (CLAHE) Applied to X-Ray Images: An Analysis of the Effect on Image Quality." *Radiography*, 27(1), 174–179.
4. Scholkopf, B., Smola, A. J., & Muller, K. R. (1998). "Nonlinear Component Analysis as a Kernel Eigenvalue Problem." *Neural Computation*, 10(5), 1299-1319.
5. Singh, P., & Gupta, K. (2019). "Role of Machine Learning Algorithms in Oral Cancer Detection: A Review." *Journal of Oral Biology and Craniofacial Research*, 9(3), 190-196.
6. Dergipark (2023). "Makine Öğrenmesi Yöntemleri ile OSCC Teşhisinin İyileştirilmesi." *Dergipark Makaleleri*. Erişim: https://dergipark.org.tr/.
7. K.aggle. "Oral Squamous Cell Carcinoma Dataset." *Kaggle Public Datasets*. Erişim: https://www.kaggle.com/.