MAKİNE ÖĞRENMESİ FİNAL PROJESİ

HAZIRLAYAN

ADI SOYADI : FURKAN DOĞAN
ÖĞRENCİ NUMARASI : 180303026
TESLİM TARİHİ : 06/01/2023

DERS ADI : MAKİNE ÖĞRENMESİ
DERS YÜRÜTÜCÜSÜ : DR. SAEİD AGAHİAN

Rapor/Ödev Başlığı: BİL 411 / Makine Öğrenmesi Final Projesi
Hazırlayanın Adı Soyadı: Furkan Doğan

BÖLÜM – 1 : VERİ SETİNE GENEL BAKIŞ

a. **Problemin Tanımı**

b. **Sınıf ve Örnek Sayısı**

BÖLÜM - 2 : ALGORİTMA HAKKINDA TEORİK BİLGİ

BÖLÜM – 3 : ÇAPRAZ GEÇERLİLİK (CROSS-VALİDATİON) DEĞERLENDİRME ÖLÇEĞİ

BÖLÜM – 4 : CONFUSİON MATRİX İLE DEĞERLENDİRME

---

1 - VERİ SETİNE GENEL BAKIŞ

Ses veri setinden Random Forest algoritmasını kullanarak bir Cinsiyet Tanıma sınıflandırıcısı oluşturacağız. Fikir, sesin ve konuşmanın akustik özelliklerine dayalı olarak bir sesi erkek veya dişi olarak tanımlamaktır. Veri seti, erkek ve kadın konuşmacılardan toplanan 3.168 kayıtlı ses örneğinden oluşmaktadır. Ses örnekleri, 0hz-280hz analiz frekans aralığı ile Seewave ve tuneR paketleri kullanılarak R'de akustik analiz ile önceden işlenmiştir. **Veri seti Kaggle' den alınmıştır.**

Problemin Tanımı

Amaç, bir Karar ağacı ve Rastgele Orman Ağacı sınıflandırıcısı oluşturmak ve her iki modelin doğruluğunu karşılaştırmaktır. 'label' sütunu, kişinin cinsiyetini tahmin etmemiz gereken hedef değişkendir.

Sınıf ve Örnek Sayısı

- **meanfreq**: ortalama frekans (kHz **cinsinden)**
- **sd**: frekansın standart sapması
- **median**: medyan frekans (kHz cinsinden)
- **Q25**: birinci nicelik (kHz cinsinden)
- **Q75**: üçüncü nicelik (kHz cinsinden)
- **IQR**: çeyrekler arası aralık (kHz cinsinden)
- **skew**: çarpıklık
- **kurt**: basıklık
- **sp.ent**: spektral entropi
- **sfm**: spektral düzlük
- **mode**: mod frekansı
- **centroid**: frekans merkezi
- **peakf**: tepe frekansı (en yüksek enerjiye sahip frekans)
- **meanfun**: akustik bir sinyal boyunca ölçülen temel frekansın ortalaması
- **minfun**: akustik bir sinyal boyunca ölçülen minimum temel frekans
- **maxfun**: akustik bir sinyal boyunca ölçülen maksimum temel frekans
- **meandom**: bir akustik sinyal boyunca ölçülen baskın frekansın ortalaması
- **mindom**: bir akustik sinyal boyunca ölçülen minimum baskın frekans
- **maxdom**: bir akustik sinyal boyunca ölçülen maksimum baskın frekans
- **dfrange**: bir akustik sinyal boyunca ölçülen baskın frekans aralığı
- **modindx**: frekans aralığına bölünen temel frekansların bitişik ölçümleri arasındaki birikmiş mutlak fark olarak hesaplanan modülasyon indeksi
- **label**: bay veya bayan

21 tane sınıf bulunmaktadır.

2 - ALGORİTMA HAKKINDA TEORİK BİLGİ

Decision Tree Classification

Ağaç tabanlı öğrenme algoritmaları, en çok kullanılan ve denetimli öğrenme yöntemlerinden biri olarak düşünülmektedir. Ağaç tabanlı yöntemler, yüksek doğruluk, kararlılık ve yorumlanma kolaylığına sahiptir. **Decision Tree** yapısında, ağaç üzerindeki her bir karar noktasında veri kümesinin o anki dağılımına göre bir "ölçüt" kullanılır. Bu ölçüt, ağaç yapısının ne şekilde büyüyeceğini belirler. Bu ölçütlerden ikisi, **entropy** ve **gini** indeksleridir.

Entropy: Veri kümesindeki örneklerin sınıf dağılımına göre bir ölçüttür. Eğer veri kümesinde sınıflar eşit dağılım gösteriyorsa, **entropy** değeri yüksek olur. Eğer veri kümesinde bir sınıf aşırı derecede öne çıkıyorsa, **entropy** değeri düşük olur.

Gini indeksi: **Gini** indeksi de veri kümesindeki örneklerin sınıf dağılımına göre bir ölçüttür. Eğer veri kümesinde sınıflar eşit dağılım gösteriyorsa, **gini** indeksi değeri yüksek olur. Eğer veri kümesinde bir sınıf aşırı derecede öne çıkıyorsa, **gini** indeksi değeri düşük olur. Hangisi kullanılacağı, veri kümesine göre değişebilir. Örneğin, eğer veri kümesinde sınıflar eşit dağılım gösteriyorsa, **entropy** değeri yüksek olacağı için **entropy** tercih edilebilir. Eğer veri kümesinde bir sınıf aşırı derecede öne çıkıyorsa, **gini** indeksi değeri düşük olacağı için **gini** indeksi tercih edilebilir.

Gini indeksi(dizini) veya Gini katsayısı : İtalyan istatistikçi Corrado Gini tarafından 1912'de geliştirilen istatistiksel bir ölçüdür. Katsayı, 0 (%0) ile 1 (%100) aralığındadır; 0, mükemmel eşitliği temsil eder ve 1, mükemmel eşitsizliği temsil eder.

Entropi : rasgele bir değişkenin belirsizliğinin ölçüsüdür, örneklerin keyfi bir koleksiyonunun saf olmayanlığını karakterize eder. Entropi ne kadar yüksek olursa elde edilen bilgi de o kadar fazla olur. Sezgisel olarak, belirli bir olayın öngörülebilirliğinden bahseder.

Entropi, tipik olarak, eğitim örneklerini daha küçük alt gruplara bölmek için bir karar ağacında bir düğümü kullandığımızda değişir. Bilgi kazancı, entropideki bu değişimin bir ölçüsüdür.

Random Forest Classification

Rastgele ormanlar veya rastgele karar ormanları, sınıflandırma, regresyon ve diğer görevler için, eğitim aşamasında çok sayıda karar ağacı oluşturarak problemin tipine göre sınıf veya sayı tahmini yapan bir toplu öğrenme yöntemidir. Random forest, birden fazla karar ağacını kullanarak daha uyumlu modeller üreterek daha isabetli sınıflandırma yapmaya çalışır. Bu yüzden öncelikle DT Sınıflandırmasını ardından RF sınıflandırmasını kullandım. Çalışma prensibi ise ;

[DT = Decision Trees - RF = Random Forest] adından da anlaşılacağı gibi, bir orman oluşturur ve bunu bir şekilde rastgele yapar. Kurduğu "orman", çoğu zaman "bagging" yöntemiyle eğitilen karar ağaçları topluluğudur. Bagging yönteminin genel fikri, öğrenme modellerinin bir kombinasyonunun genel sonucu arttırmasıdır.

K-Nearest Neighbors(KNN)

Diğer Denetimli Öğrenme algoritmalarının aksine, eğitim aşamasına sahip değildir. Eğitim ve test hemen hemen aynı şeydir. Tembel bir öğrenme türüdür. Bu nedenle, kNN, geniş veri setini işlemek için gereken algoritma olarak ideal bir aday değildir. KNN ile temelde yeni noktaya en yakın noktalar aranır. K, bilinmeyen noktanın en yakın komşularının miktarını temsil eder. Sonuçları tahmin etmek için algoritmanın k miktarını (biz k=10 kabul edeceğiz) seçeriz. KNN algoritmasının bir özelliği, verilerin yerel yapısına duyarlı olmasıdır.

3 - ÇAPRAZ GEÇERLİLİK (CROSS-VALİDATİON) DEĞERLENDİRME ÖLÇEĞİ

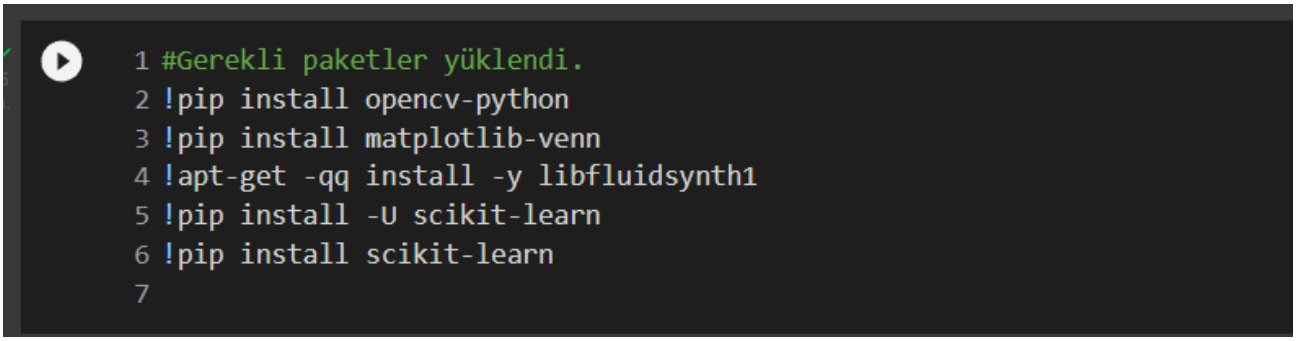
Cross-validation, makine öğrenmesi modelinin görmediği veriler üzerindeki performansını mümkün olduğunca objektif ve doğru bir şekilde değerlendirmek için kullanılan istatistiksel bir yeniden

örnekleme(resampling) yöntemidir. İkinci bir kullanım alanı ise modelde hiper parametre optimizasyonu yapmaktır. K-Fold CV yöntemi üzerinden çalışma mantığı ise şudur :

1. Veri seti rastgele olacak şekilde karıştırılır. (opsiyonel)
2. Veri seti k gruba ayrılır.
3. Her grup icin aşağıdaki adımlar uygulanır :
   - Seçilen grup validasyon seti olarak kullanılır.
   - Diğer tüm gruplar (k-1 grup) train seti olarak kullanılır.
   - Train seti kullanılarak model kurulur ve validasyon seti ile değerlendirilir.
   - Modelin değerlendirme puanı bir listede saklanır.

Şimdi projeye ait kısımları inceleyelim..

Yorum satırlarında yapılan her şey açıklanmıştır.

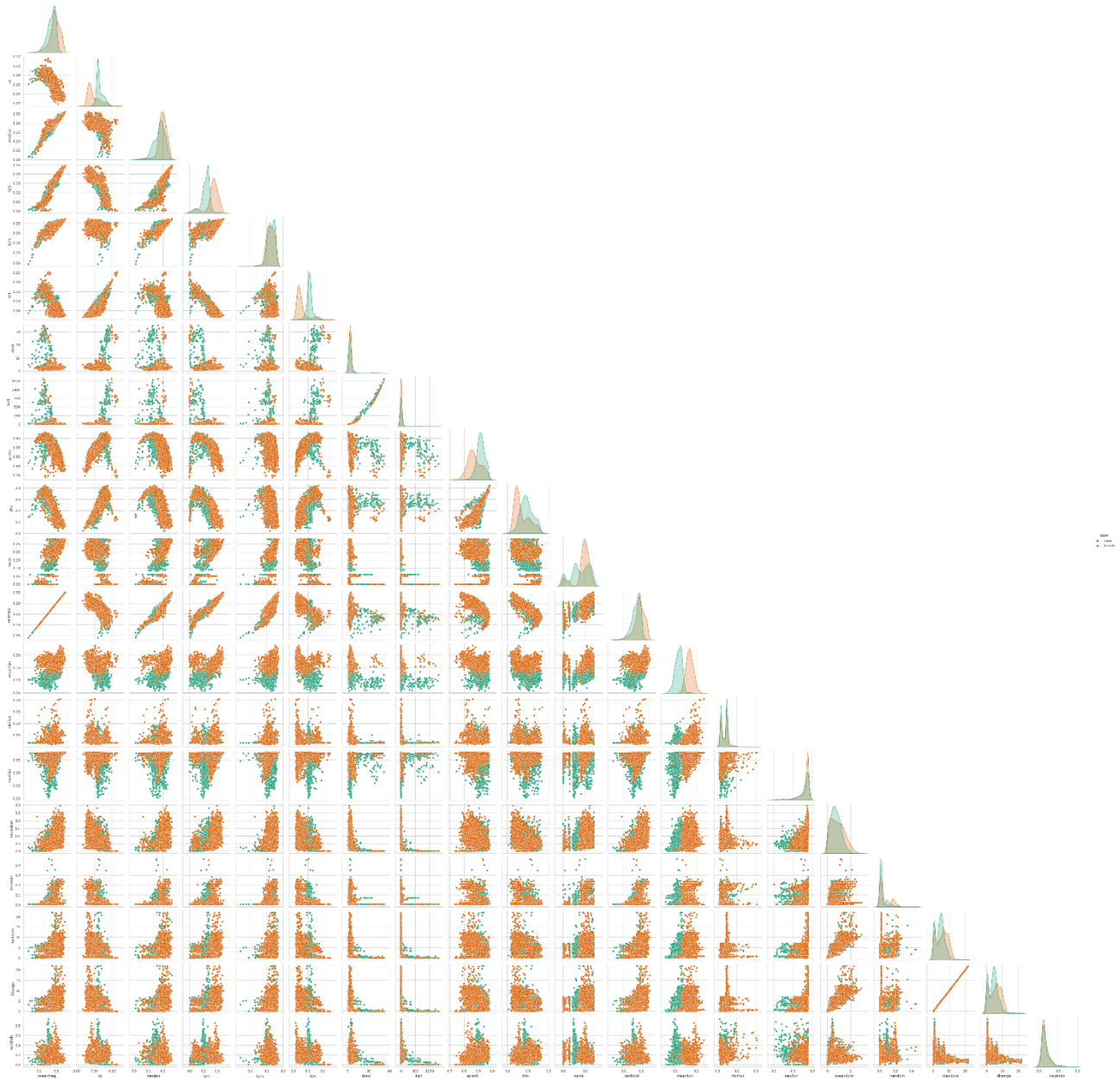


```
#Gerekli paketler yüklendi.
!pip install opencv-python
!pip install matplotlib-venn
!apt-get -qq install -y libfluidsynth1
!pip install -U scikit-learn
!pip install scikit-learn

```

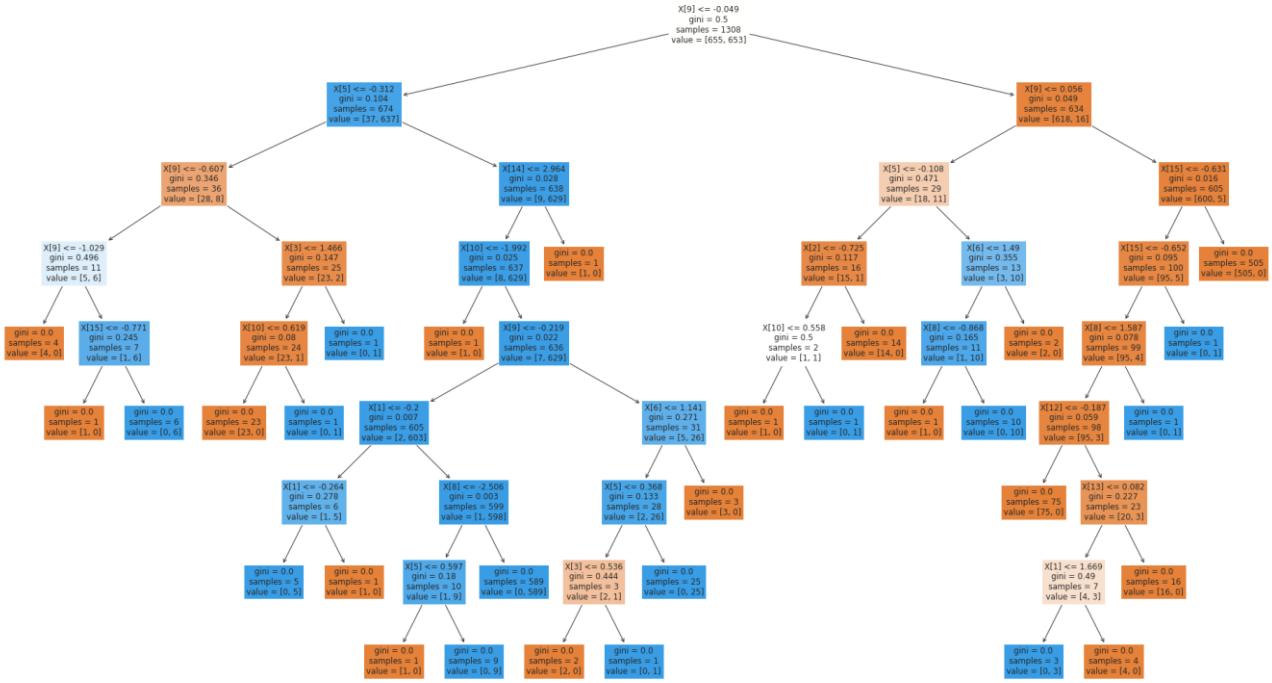
```
train.head().style.background_gradient(cmap='coolwarm') # Veri kümesinin ilk beş satırını göster ve değerleri renklerle belirt.
```

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	mode	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	modindx	label
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.893369	0.491918	0.000000	0.059781	0.084279	0.015702	0.275862	0.007812	0.007812	0.007812	0.000000	0.000000	male
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.892193	0.513724	0.000000	0.066009	0.107937	0.015826	0.250000	0.009014	0.007812	0.054688	0.046875	0.052632	male
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.846389	0.478905	0.000000	0.077316	0.098706	0.015656	0.271186	0.007990	0.007812	0.015625	0.007812	0.046512	male
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232	0.083878	0.151228	0.088965	0.017798	0.250000	0.201497	0.007812	0.562500	0.554688	0.247119	male
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568	0.104261	0.135120	0.106398	0.016931	0.266667	0.712812	0.007812	5.484375	5.476562	0.208274	male

Nitelikler arasındaki korelasyonu ve sınıf değişkenine göre dağılımı gösteren çoklu görsel.



Gini indexi;



ÇAPRAZ GEÇERLİLİK (CROSS-VALİDATİON) DEĞERLENDİRME ÖLÇEĞİ

```
1 #DecisionTree Sınıflandırmasına ait Cross Valudation kısmı.
2 # Karar ağacı sınıflandırma modeline ait k-katlı çapraz doğrulama (k-fold cross validation) işlemini gerçekleştir.
3 from sklearn.model_selection import cross_val_score
4 accuracies = cross_val_score(estimator=clf, X=x_train, y=y_train, cv=10)
5 # Elde edilen doğruluk skorlarını yazdır.
6 accuracies
7
```

```
array([0.99236641, 0.98473282, 0.95419847, 0.96183206, 0.96183206,
       0.98473282, 0.96183206, 0.94656489, 0.95384615, 0.96153846])
```

```
[270] 1 #Cross Valudation değerinin ortalaması.
      2 accuracies.mean()
```

```
0.9663476218438051
```

```
[271] 1 #Cross Valudation değerinin standart sapması.
      2 accuracies.std()
```

```
0.014603262857476538
```

CONFUSİON MATRİX İLE DEĞERLENDİRME

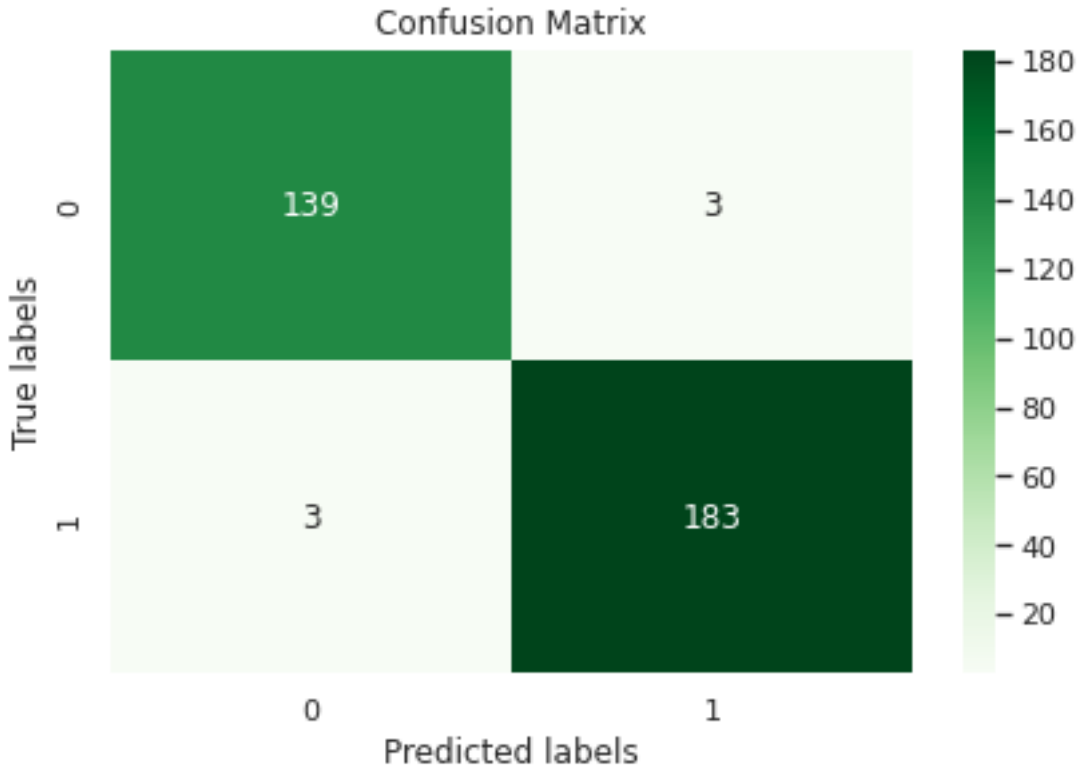
Makine öğrenmesinde kullanılan sınıflandırma modellerinin performansını değerlendirmek için hedef niteliğe ait tahminlerin ve gerçek değerlerin karşılaştırıldığı hata matrisinin görseli aşağıdadır. Her ne olursa olsun sınıflandırma tahminleri şu dört değerlendirmeden birine sahip olacaktır:

- Doğruya doğru demek (True Positive – TP) DOĞRU
- Yanlışa yanlış demek (True Negative – TN) DOĞRU
- Doğruya yanlış demek (Predicted Positive – FP) YANLIŞ
- Yanlışa doğru demek (Predicted Negative – FN) YANLIŞ

RANDOMFORESTCLASSİFİER CONFUSİON MATRİX

```
import matplotlib.pyplot as plt
import seaborn as sns

pred = clf_rf.predict(x_test) # modelden tahminler yap.
conf_matrix = confusion_matrix(y_test, pred) # confusion matrix oluştur.

#RandomForestClassifier confusion matrix'i grafik şeklinde gösterin
ax= plt.subplot()
# confusion matrix'i "Greens" renk paletiyle göster.
sns.heatmap(conf_matrix, annot=True, ax = ax, cmap='Greens', fmt='g')

# Eksen etiketlerini ayarla.
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(['0', '1'])
ax.yaxis.set_ticklabels(['0', '1'])

```

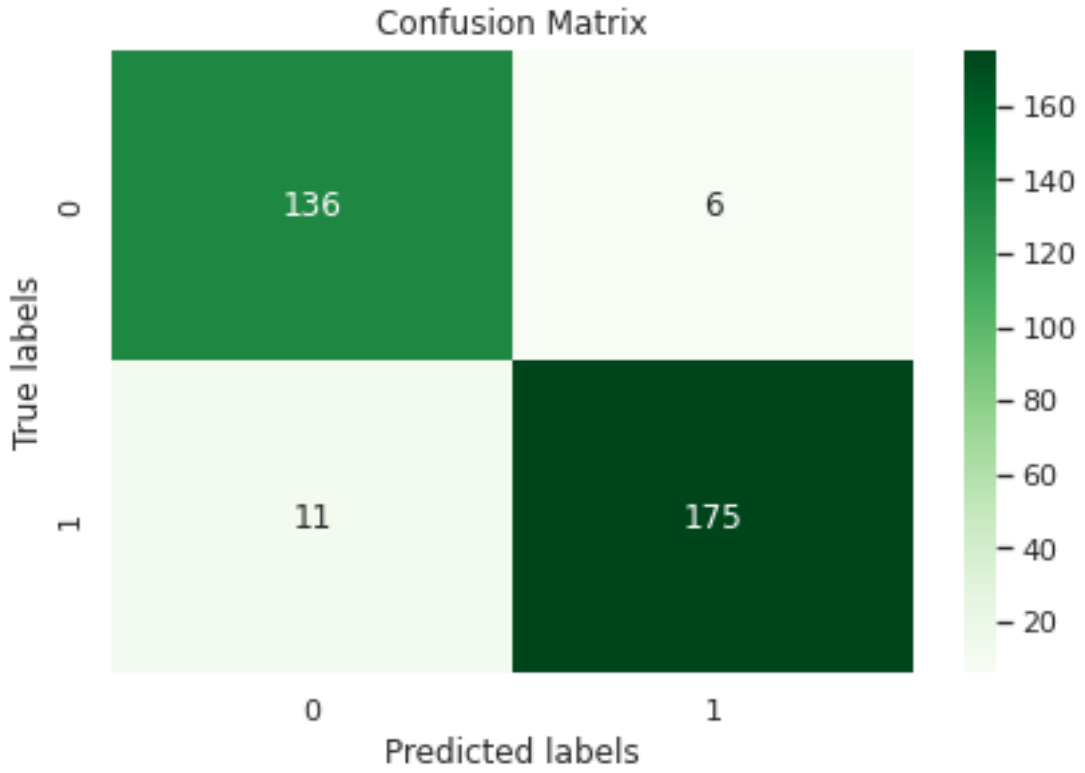
Yukarıdaki kodun çıktısı olarak aşağıdaki Confusion Matrix görüntüsü elde edilir.



DECİSİONTREECLASSİFİER CONFUSİON MATRİX

```
import matplotlib.pyplot as plt
import seaborn as sns

pred = clf_dt.predict(x_test) # modelinizden tahminler yap.
conf_matrix = confusion_matrix(y_test, pred) # confusion matrix oluştur.

#DecisionTreeClassifier confusion matrix'i grafik şeklinde göster.
ax= plt.subplot()
# confusion matrix'i "Greens" renk paletiyle göster.
sns.heatmap(conf_matrix, annot=True, ax = ax, cmap='Greens', fmt='g')

# Eksen etiketlerini ayarla.
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(['0', '1'])
ax.yaxis.set_ticklabels(['0', '1'])
```

Yukarıdaki kodun çıktısı olarak aşağıdaki Confusion Matrix görüntüsü elde edilir.



ACCURACY_SCORE İLE DOĞRULUK ORANI

Yüksek bir doğruluk oranı, modelin etkili bir şekilde sınıflandırma yaptığı anlamına gelebilir.

```python
from sklearn import tree


# Veri çerçevesinden etiket değişkenini düşürün ve değişkenleri standartlaştır.
scaler=StandardScaler()
scaled_df=scaler.fit_transform(temp_df.drop('label',axis=1))
X=scaled_df
Y=df['label']

# Eğitim ve test verilerini ayırın
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.20,random_state=42)

#Bu kod parçacığında, veri kümesinden bir karar ağacı ve rastgele ormanı kullanarak sınıflandırma modelleri eğitilmektedir.
# Karar ağacı modelini oluşturun ve eğit.
clf_dt=DecisionTreeClassifier()
clf_dt.fit(x_train,y_train)

# Karar ağacı modelini görsel olarak çizdir.
plt.figure(figsize=(30,18))
tree.plot_tree(clf,filled=True)
plt.show()

# Karar ağacı modelinin test verisi üzerinden tahminler yapın ve doğruluk skoru hesapla.
pred=clf_dt.predict(x_test)
print("Karar ağacı modeli doğruluk oranı: ",accuracy_score(pred,y_test))

# Rastgele orman modelini oluşturun ve eğit.
clf_rf=RandomForestClassifier()
clf_rf.fit(x_train,y_train)

# Rastgele orman modelinin test verisi üzerinden tahminler yapın ve doğruluk skoru hesapla.
pred=clf_rf.predict(x_test)
print("Rastgele orman modeli doğruluk oranı: ", accuracy_score(pred,y_test))
```

```
Karar ağacı modeli doğruluk oranı:  0.948170731707317
Rastgele orman modeli doğruluk oranı:  0.9817073170731707
```

"GridSearchCV()" fonksiyonu, modelinizin performansını optimize etmek için verilen parametrelerin değerlerini değiştirerek modeli eğitir ve test eder.

```
# En iyi skor ve parametreleri yazdır.
print("Best score : ",CV_rfc.best_score_)
print("Best Parameters : ",CV_rfc.best_params_)
# Test verisindeki tahminlerin doğruluk skorunu yazdır.
print("Precision Score : ", precision_score(CV_rfc.predict(x_test),y_test))
```

```
Best score :  0.9816472171039162
Best Parameters :  {'criterion': 'entropy', 'max_depth': 7, 'max_features': 'auto', 'n_estimators': 200}
Precision Score :  0.9838709677419355
```