Grup 3 | 27/05/2024

Bu sunum nesne tabanlı programlama II dersi için hazırlanmıştır

Random Forest

Meme Kanseri Tespiti

Projenin adı

Breast Cancer Predicition

Hazırlayanlar

Adem Mert Sedef
Efe Abagünay,
Emirhan Duyan,
Furkan Kül,
Yunus Emre Kuru

İçindekiler

Giriş

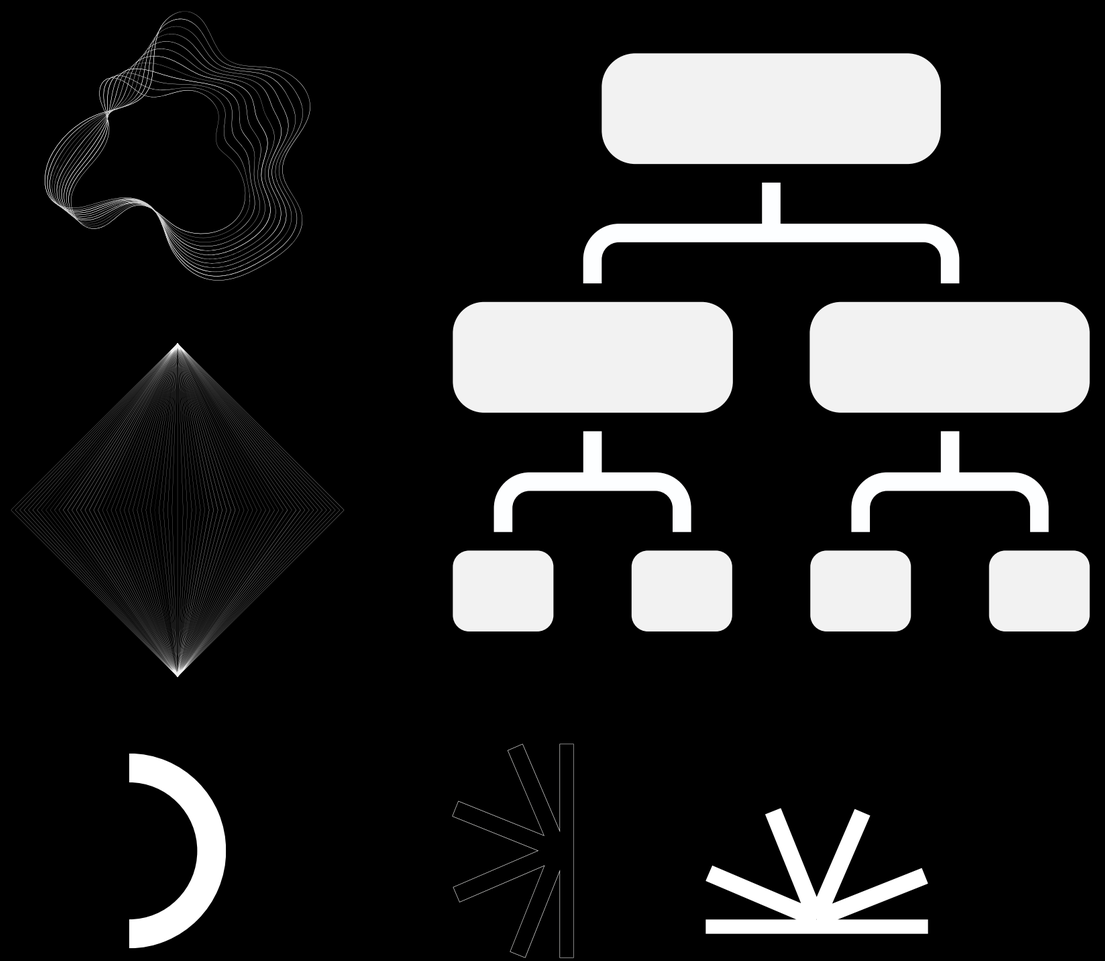
Bu projede, göğüs kanseri teşhisi ve sınıflandırması için makine öğrenmesi yöntemlerinden biri olan Random Forest (Rastgele Orman) algoritması kullanılmıştır. Amaç, verilen hasta özelliklerine dayanarak tümörün iyi huylu mu yoksa kötü huylu mu olduğunu tahmin etmektir.

Giriş

Veri kümesi olarak Wisconsin Üniversitesi tarafından toplanan "Wisconsin Göğüs Kanseri Veri Seti" kullanılmıştır. Bu veri seti, 569 hastadan alınan hücre çekirdeği özelliklerini içermektedir. Her bir hasta örneği için 30 farklı özellik (örneğin, hücre yarıçapı, doku kalınlığı, simetri) ve tümörün sınıfı (iyi huylu veya kötü huylu) bilgisi bulunmaktadır.

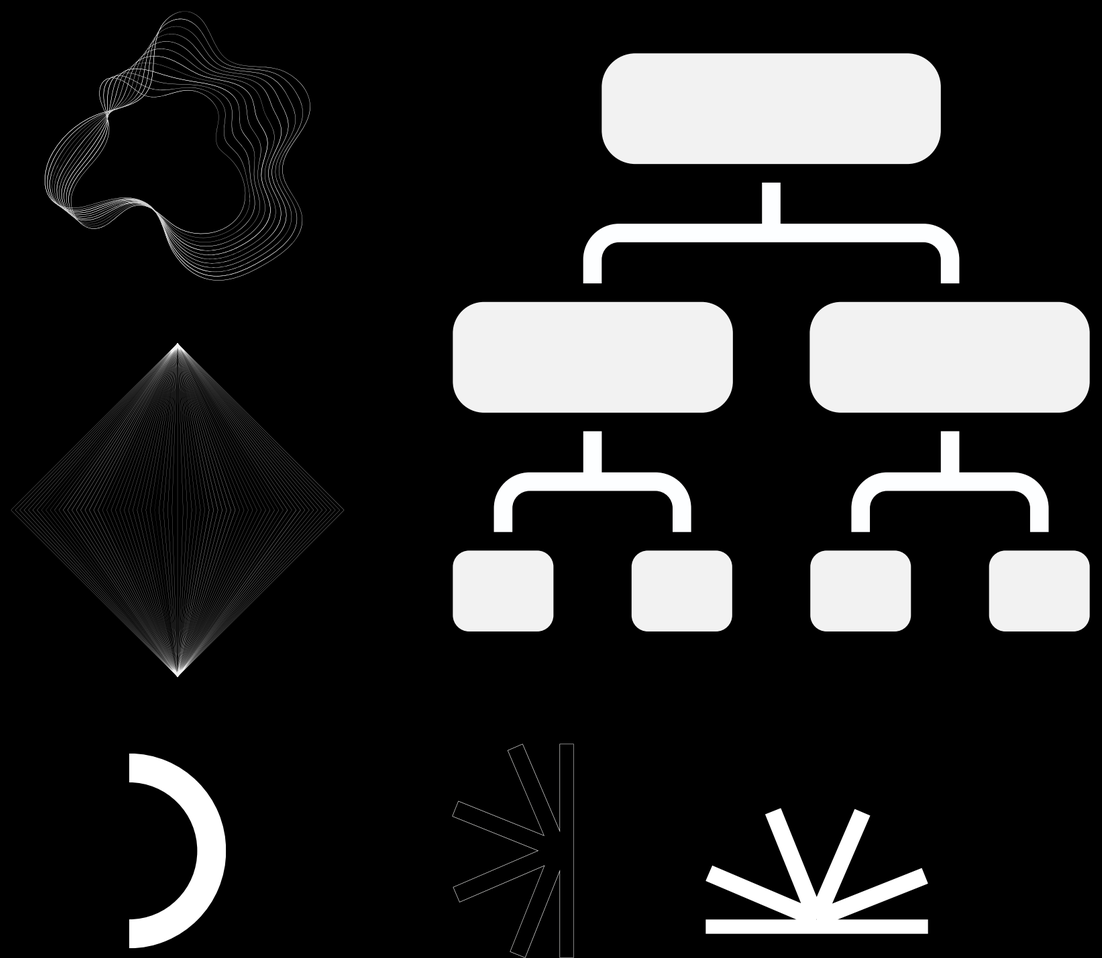
Random Forest Nedir?

Random forest, bir makine öğrenmesi tekniği olan ensemble learning (topluluk öğrenimi) altında bir karar ağacı algoritmasıdır. Karar ağacı algoritmaları, veri kümesini belirli koşullara dayanarak ağaç yapısında modellemeye dayanır.Ancak tek bir karar ağacı genellikle karmaşık veri setlerinde iyi performans göstermez ve aşırı uydurmayı (overfitting) teşvik edebilir.



Random Forest Nedir?

Random forest, bu durumu aşmak için bir çözüm sunar. Random forest, birden fazla karar ağacını bir araya getirir ve her bir ağacı rastgele alt örneklemeler ve rastgele alt özellikler kullanarak eğitir.



Random Forest Nerelerde Kullanılır?

Başlıca Kullanım Alanları :

1. Sınıflandırma (Classification)

Tıbbi Tanı: Hastalık teşhisi
Spam Filtreleme: E-posta sınıflandırma

2. Regresyon (Regression)

Emlak Değerleme: Ev fiyat tahmini
Hisse Senedi Tahmini: Piyasa fiyat tahmini

3. Özellik Seçimi (Feature Selection)

Veri setindeki kritik özellikleri belirleme

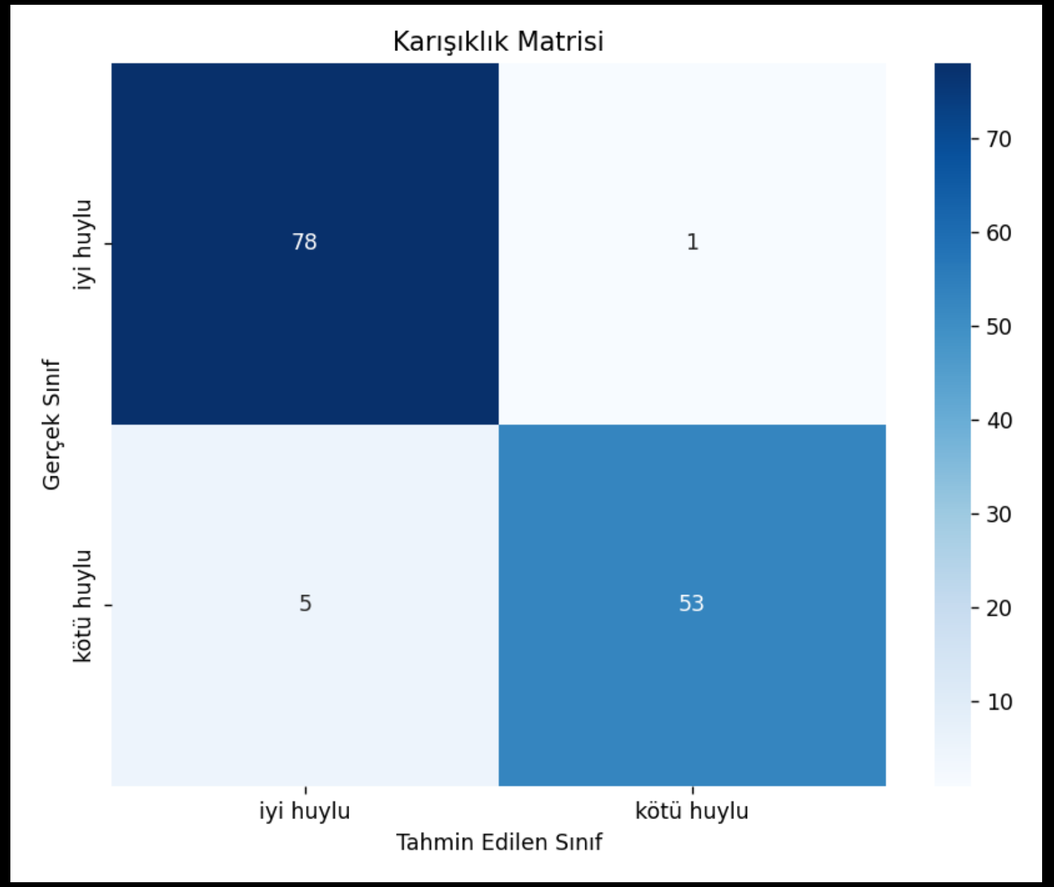
Kullandığımız Kütüphaneler

- import pandas as pd
- from sklearn.model_selection import train_test_split
- from sklearn.ensemble import RandomForestClassifier
- from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report, roc_curve, auc
- from sklearn.tree import plot_tree
- import matplotlib.pyplot as plt
- import seaborn as sns

pandas
sklearn
matplotlib
seaborn

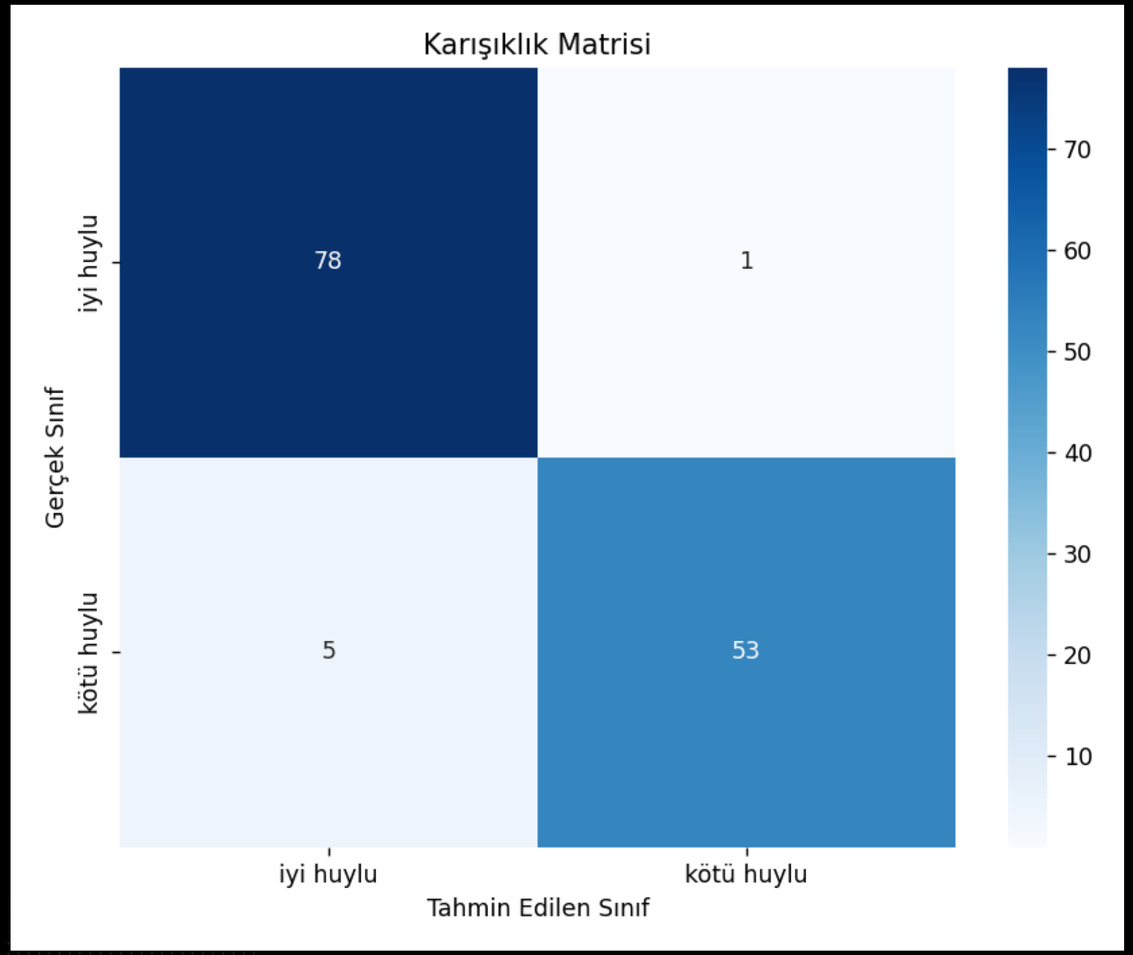
Karışıklık Matrisi nedir?

Karışıklık matrisi (confusion matrix), sınıflandırma modelinin performansını değerlendirmek için kullanılan bir tablodur. Karışıklık matrisi, gerçek sınıf değerleri ile modelin tahmin ettiği sınıf değerleri arasındaki ilişkiyi gösterir.



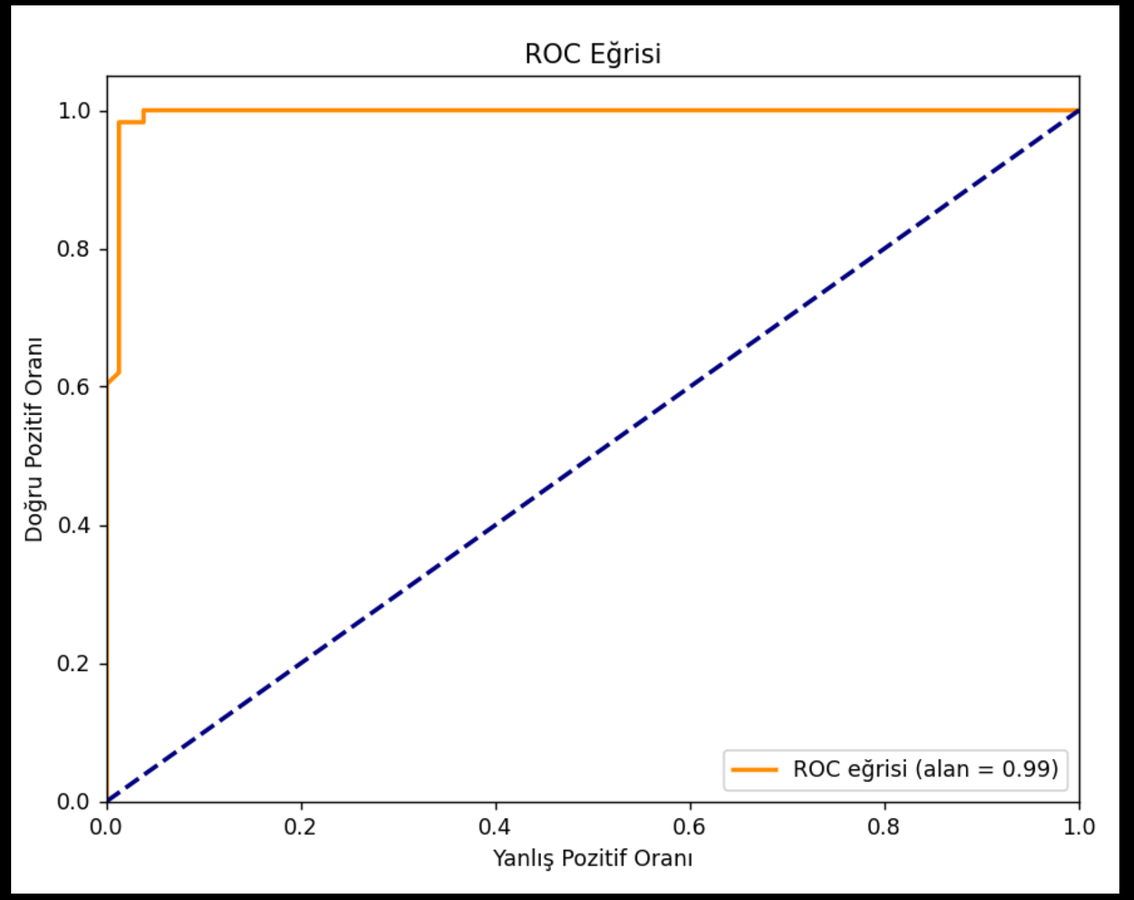
Karışıklık Matrisi Görselleştirme (Seaborn ile)

```
plt.figure(figsize=(8, 6))
sns.heatmap(karmasiklik_matrisi, annot=True,
fmt="d", cmap="Blues",
xticklabels=self.model.classes_,
yticklabels=self.model.classes_)
plt.xlabel("Tahmin Edilen Sınıf")
plt.ylabel("Gerçek Sınıf")
plt.title("Karışıklık Matrisi")
plt.show()
```



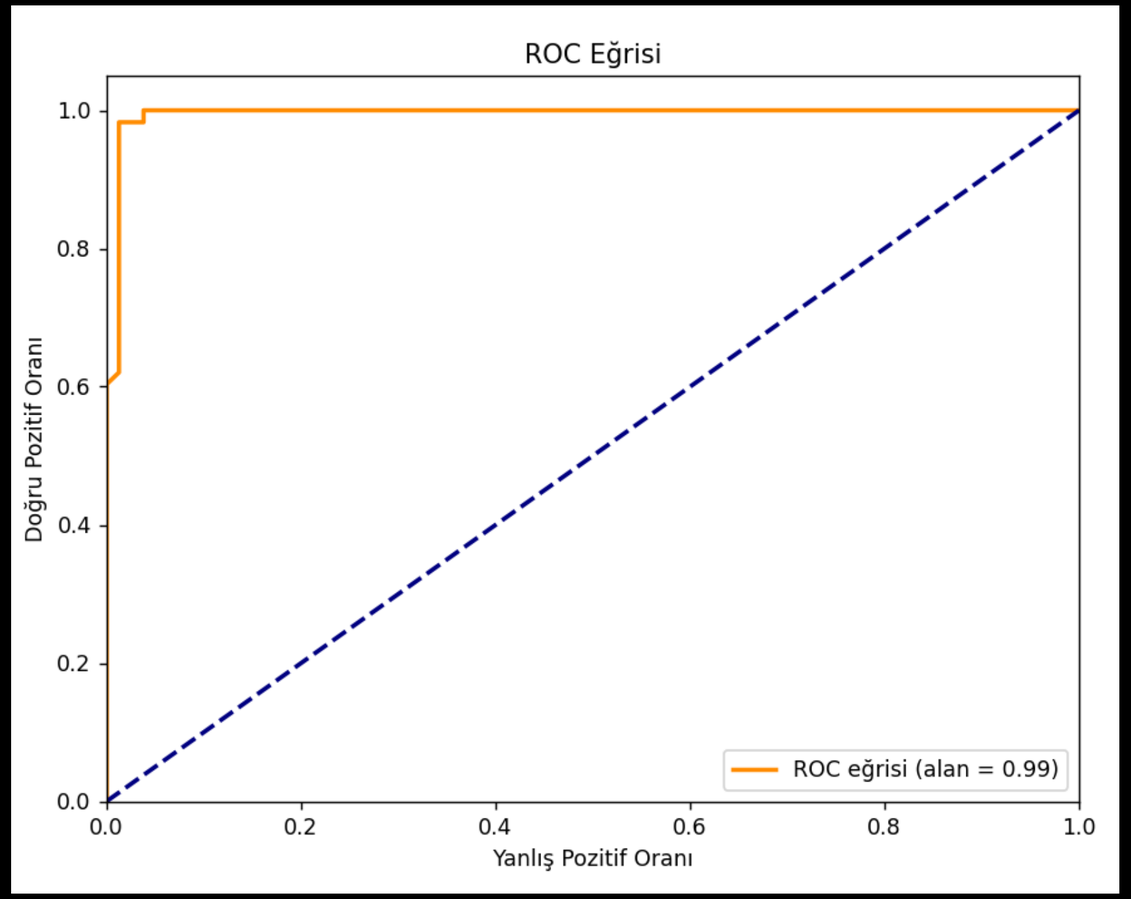
Roc Eğrisi nedir?

ROC (Receiver Operating Characteristic) eğrisi, sınıflandırma modellerinin performansını değerlendirmek için kullanılan bir grafiktir.



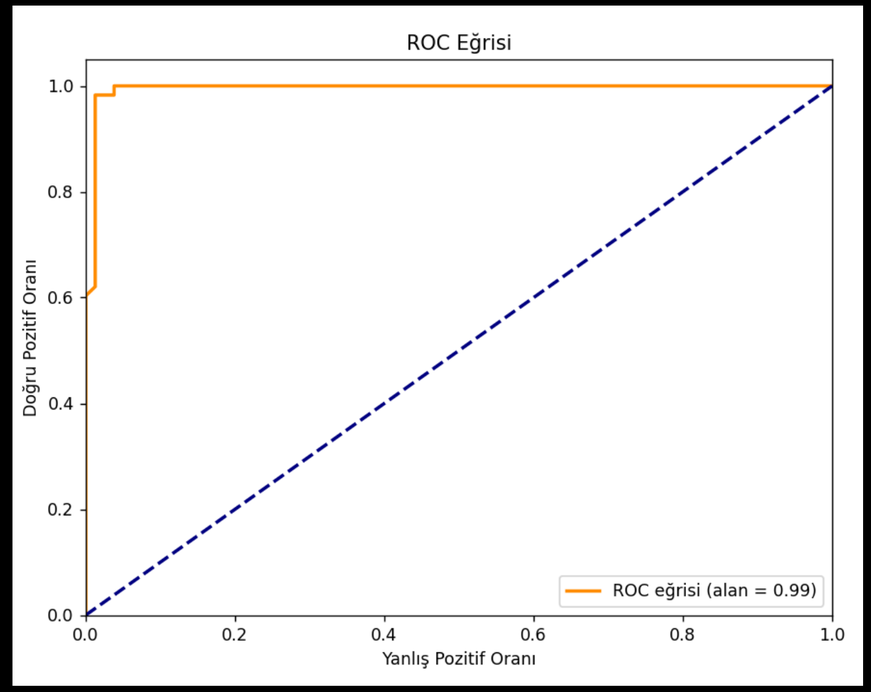
Roc Eğrisi nedir?

ROC eğrisi çizilirken, farklı bir eşik değeri (threshold) kullanarak modelin duyarlılık ve özgüllüğü hesaplanır. Eşik değeri, modelin pozitif olarak sınıflandıracağı örnekleri belirler. Eşik değeri arttıkça, model daha az örnek üzerinde pozitif tahminler yapar ve bu da genellikle özgüllüğü artırır ancak duyarlılığı azaltır.



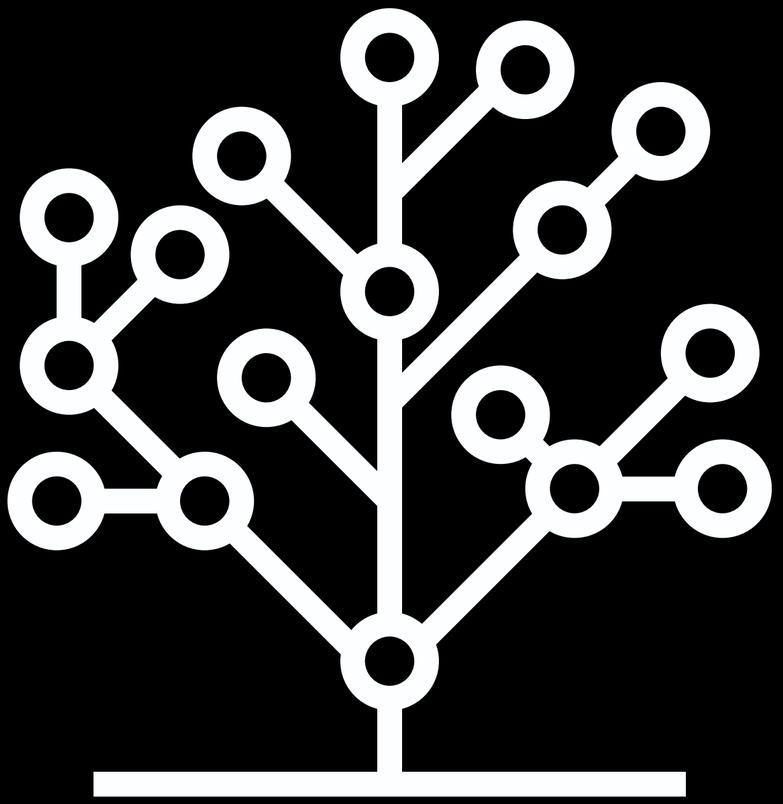
ROC Eğrisi Görselleştirme(matplotlib)

```
fpr, tpr, thresholds = roc_curve(self.y_test, y_olasilik, pos_label="kötü huylu")
        roc_auc = auc(fpr, tpr)
      plt.figure(figsize=(8, 6))
            plt.plot(fpr, tpr, color='darkorange', lw=2, label=
            f'ROC eğrisi (alan = {roc_auc:.2f})')
            plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--
    ')
            plt.xlim([0.0, 1.0])
            plt.ylim([0.0, 1.05])
            plt.xlabel('Yanlış Pozitif Oranı')
            plt.ylabel('Doğru Pozitif Oranı')
            plt.title('ROC Eğrisi')
            plt.legend(loc="lower right")
            plt.show()
```



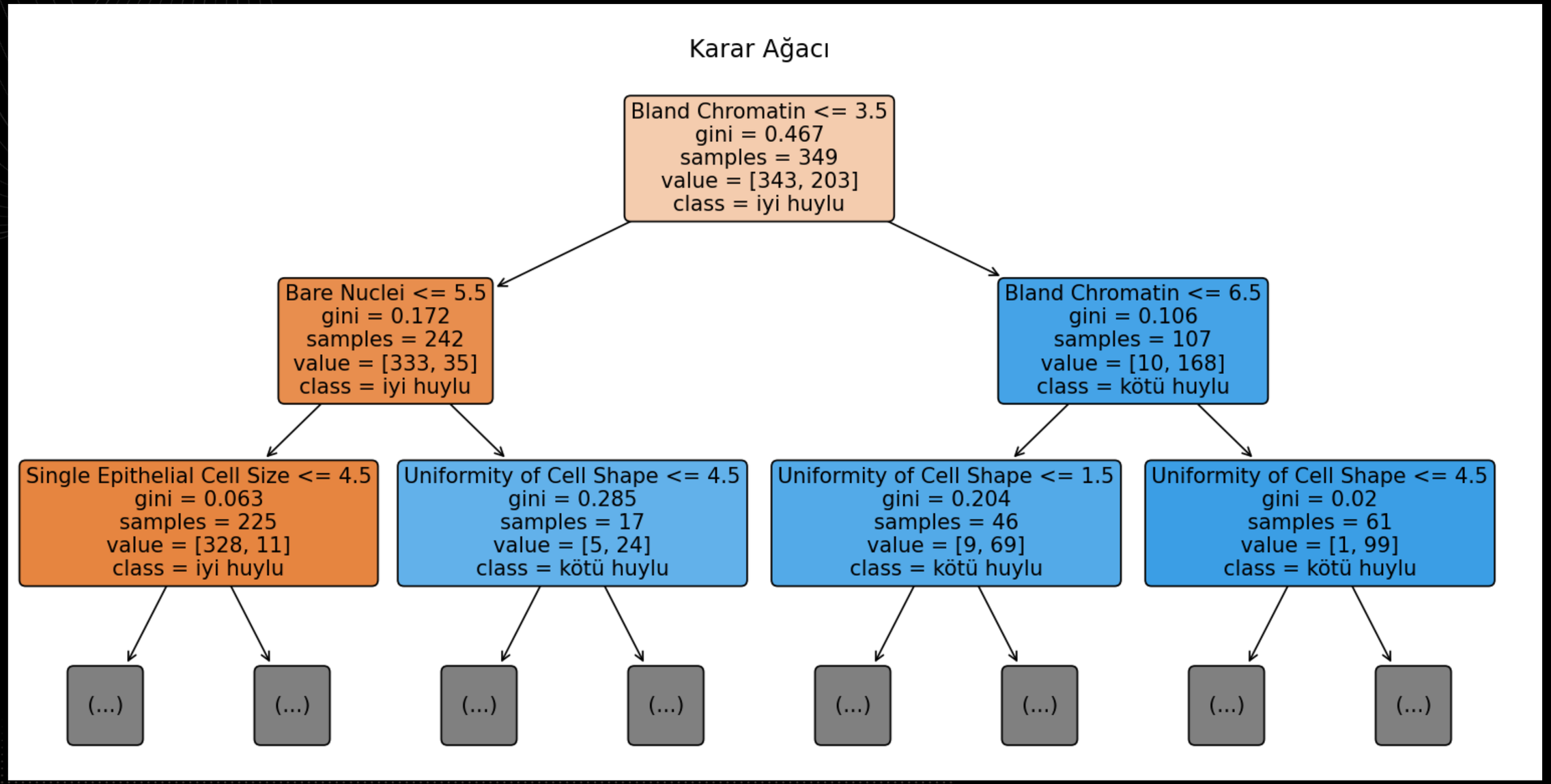
Karar Ağacı nedir?

Karar Ağacı, makine öğrenimi alanında sınıflandırma ve regresyon problemleri için kullanılan bir modelleme tekniğidir. Temel olarak, bir veri kümesindeki özelliklerin değerlerine göre kararlar vermek için ağaç yapısını kullanır.



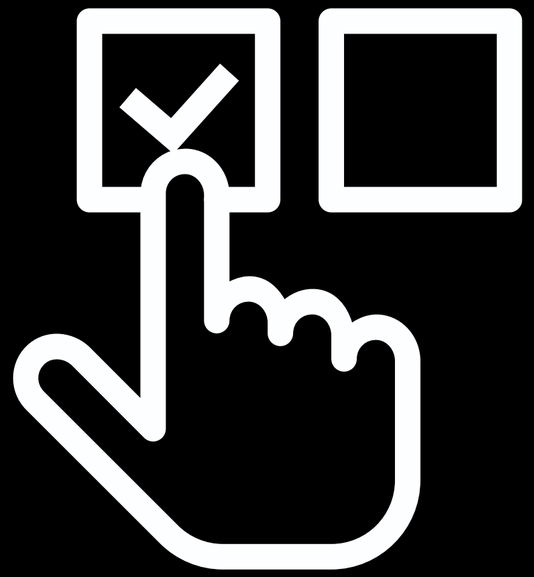
Karar Ağacı Görselleştirme (Matplotlib,scikitlearn)

```
plt.figure(figsize=(20, 15))
 agac = self.model.estimators_[0]
 plot_tree(agac,
feature_names=self.X_egitim.columns,
class_names=self.model.classes_,
 filled=True, rounded=True, fontsize=12,
max_depth=2)
 plt.title("Karar Ağacı", fontsize=14)
 plt.show()
```


Karar Ağacı
Bland Chromatin <= 3.5
gini = 0.467
samples = 349
value = [343, 203]
class = iyi huylu
Bare Nuclei <= 5.5
gini = 0.172
samples = 242
value = [333, 35]
class = iyi huylu
Bland Chromatin <= 6.5
gini = 0.106
samples = 107
value = [10, 168]
class = kötü huylu
Single Epithelial Cell Size <= 4.5
gini = 0.063
samples = 225
value = [328, 11]
class = iyi huylu
Uniformity of Cell Shape <= 4.5
gini = 0.285
samples = 17
value = [5, 24]
class = kötü huylu
Uniformity of Cell Shape <= 1.5
gini = 0.204
samples = 46
value = [9, 69]
class = kötü huylu
Uniformity of Cell Shape <= 4.5
gini = 0.02
samples = 61
value = [1, 99]
class = kötü huylu
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)

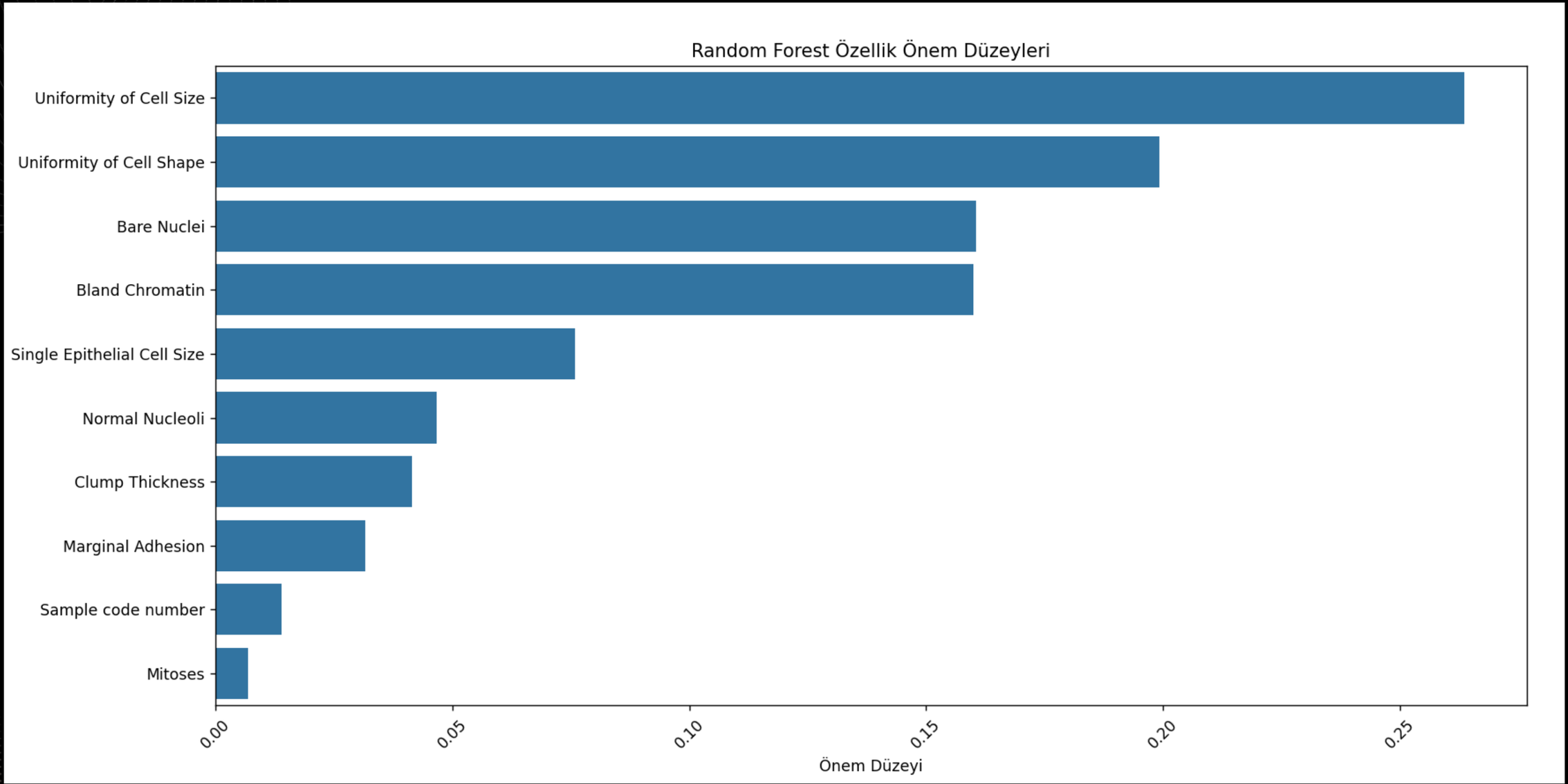
Özellik Önem Düzeyleri nedir?

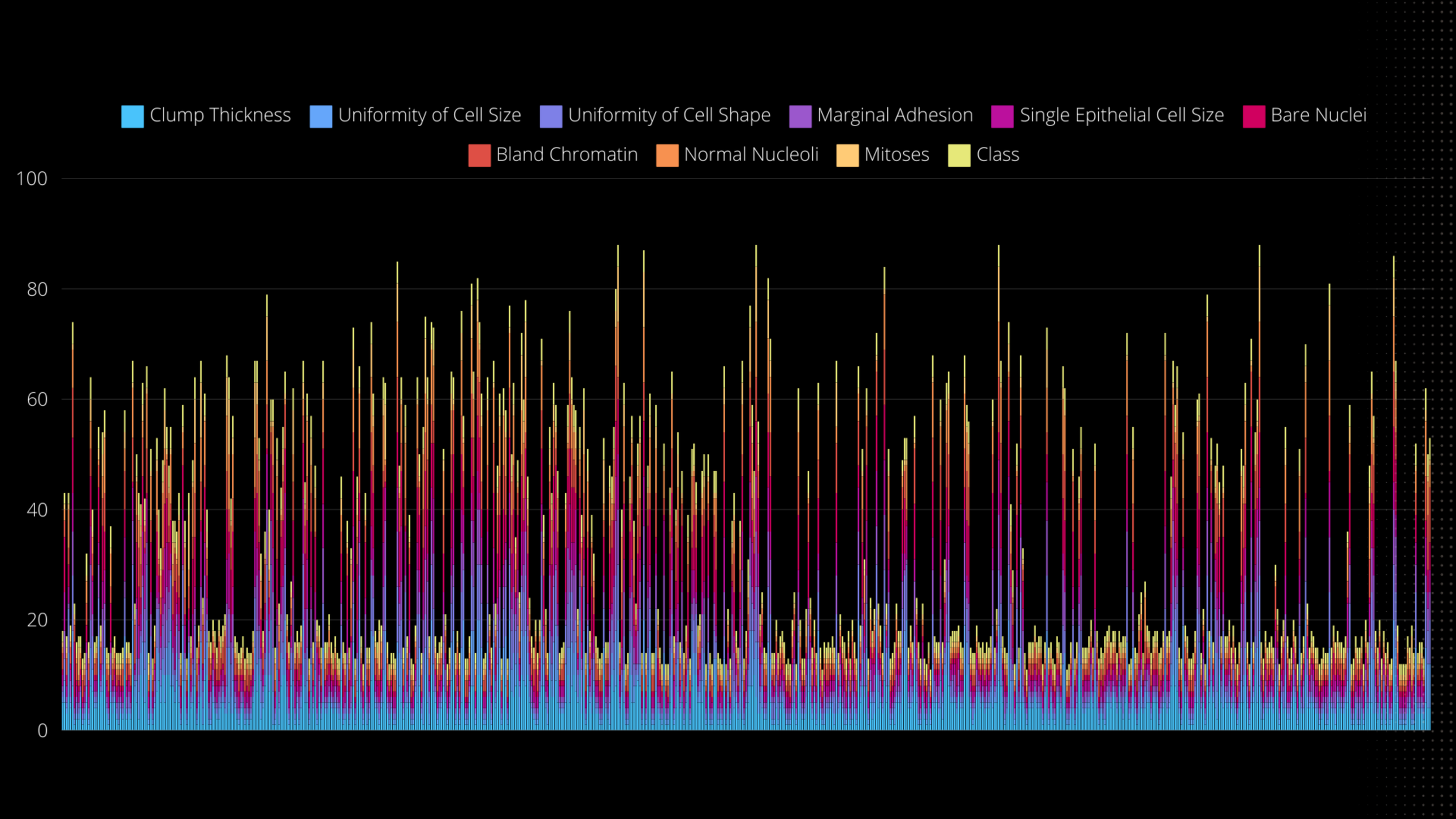
Özellik önem düzeyleri, makine öğrenimi modellerinde her bir özelliğin (değişkenin) modelin sonuçlarına olan katkısını ölçen bir metriktir. Özellik önem düzeyleri, bir modelin davranışını ve nasıl kararlar aldığını anlamak için önemli bir araç sağlar.



Özellik Önem Düzeyleri Grafiği

```
onem = pd.Series(self.model.feature_importances_,
index=self.X_egitim.columns)
 onem_sirali = onem.sort_values(ascending=False)
 plt.figure(figsize=(10, 8))
 sns.barplot(x=onem_sirali, y=onem_sirali.index)
 plt.xlabel('Önem Düzeyi')
 plt.ylabel('Özellik')
 plt.title("Random Forest Özellik Önem Düzeyleri")
 plt.xticks(rotation=45)
 plt.show()
```


Random Forest Özellik Önem Düzeyleri
Uniformity of Cell Size
Uniformity of Cell Shape
Bare Nuclei
Bland Chromatin
Single Epithelial Cell Size
Normal Nucleoli
Clump Thickness
Marginal Adhesion
Sample code number
Mitoses
0.00
0.05
0.10
0.15
0.20
0.25
Önem Düzeyi


Clump Thickness
Uniformity of Cell Size
Uniformity of Cell Shape
Marginal Adhesion
Single Epithelial Cell Size
Bare Nuclei
Bland Chromatin
Normal Nucleoli
Mitoses
Class
100
80
60
40
20
0

Kod

```python
class MemeKanseriSiniflandirici:
    def __init__(self, veri_yolu="breast_cancer.csv", test_boyutu=0.2, rastgele_durum=42):
        try:
            self.veri = pd.read_csv(veri_yolu)
        except FileNotFoundError:
            raise FileNotFoundError(f"Veri dosyası bulunamadı: {veri_yolu}")

        self.veri.rename(columns={'Class': 'class'}, inplace=True)
        self.veri.dropna(inplace=True)
        self.veri.columns = self.veri.columns.str.strip()

        X = self.veri.drop("class", axis=1)
        y = self.veri["class"].replace({2: "iyi huylu", 4: "kötü huylu"})

        self.X_egitim, self.X_test, self.y_egitim, self.y_test = train_test_split(
            X, y, test_size=test_boyutu, random_state=rastgele_durum
        )

        self.model = RandomForestClassifier(n_estimators=100, random_state=rastgele_durum)
```

Kod

```python
def train(self):
    self.model.fit(self.X_egitim, self.y_egitim)

def tahmin_et(self, X):
    return self.model.predict(X)

def olasilik_tahmin_et(self, X):
    return self.model.predict_proba(X)[:, 1]

def degerlendir(self):
    y_tahmin = self.tahmin_et(self.X_test)
    y_olasilik = self.olasilik_tahmin_et(self.X_test)

    dogruluk = accuracy_score(self.y_test, y_tahmin)
    kesinlik = precision_score(self.y_test, y_tahmin, pos_label="kötü huylu")
    duyarlilik = recall_score(self.y_test, y_tahmin, pos_label="kötü huylu")
    f1 = f1_score(self.y_test, y_tahmin, pos_label="kötü huylu")
    karmasiklik_matrisi = confusion_matrix(self.y_test, y_tahmin)
    rapor = classification_report(self.y_test, y_tahmin)

    print("Accuracy (Doğruluk):", dogruluk)
    print("F1 Score (F1 Puanı):", f1)
    print("\nPrecision (Hassasiyet):", kesinlik)
    print("Recall (Geri Çağırma):", duyarlilik)
    print("\nClassification Report:\n", rapor)
```

Kod Çıktısı

```
Accuracy (Doğruluk): 0.9562043795620438
F1 Score (F1 Puanı): 0.9464285714285714

Precision (Hassasiyet): 0.9814814814814815
Recall (Geri Çağırma): 0.9137931034482759

Classification Report:
              precision    recall  f1-score   support

   iyi huylu       0.94      0.99      0.96        79
  kötü huylu       0.98      0.91      0.95        58

    accuracy                           0.96       137
   macro avg       0.96      0.95      0.95       137
weighted avg       0.96      0.96      0.96       137
```

Doğruluk(Accuracy) nedir?

Doğruluk (Accuracy), bir sınıflandırma modelinin doğru tahmin ettiği örneklerin toplam örnekler içindeki oranını ifade eden bir metriktir.

Matematiksel olarak, doğruluk yandaki formülle hesaplanır:

$$\text{Accuracy} = \frac{\text{Doğru Tahminler}}{\text{Toplam Örnekler}}$$

Modelimizin doğruluk oranı %95.62 olarak hesaplanmıştır.

F1 Skoru nedir?

"F1 skoru" terimi, genellikle bir makine öğrenimi modelinin performansını değerlendirmek için kullanılan bir ölçüdür.

F1 skoru, hassasiyet (precision) ve geri çağırma (recall) değerlerinin harmonik ortalaması olarak hesaplanır.

$$F1 = 2 \times \frac{\text{hassasiyet} \times \text{geri çağırma}}{\text{hassasiyet} + \text{geri çağırma}}$$

Modelimizin F1 skoru %94.64'tür.

Hassasiyet (Precision)

Hassasiyet, bir sınıflandırma modelinin pozitif olarak tahmin ettiği örneklerin ne kadarının gerçekten pozitif olduğunu ölçen bir metriktir.

Hassasiyet (P), yandaki formülle hesaplanır:

$$F1 = 2 \times \frac{\text{hassasiyet} \times \text{geri çağırma}}{\text{hassasiyet} + \text{geri çağırma}}$$

Modelimizin kötü huylu tümörleri doğru tespit etme oranı %98.0'dir.

Geri Çağırma(Recall) nedir?

Geri çağırma (Recall), bir sınıflandırma modelinin gerçek pozitiflerin ne kadarını doğru bir şekilde tanıdığını ölçen bir metriktir.

Geri çağırma (R), yandaki formülle hesaplanır:

$$\text{Geri Çağırma (Recall)} = \frac{\text{Doğru Pozitifler}}{\text{Doğru Pozitifler} + \text{Yanlış Negatifler}}$$

Modelimizin tüm kötü huylu tümörleri yakalama oranı %92.0'dır.

Dinlediğiniz için teşekkür ederiz.