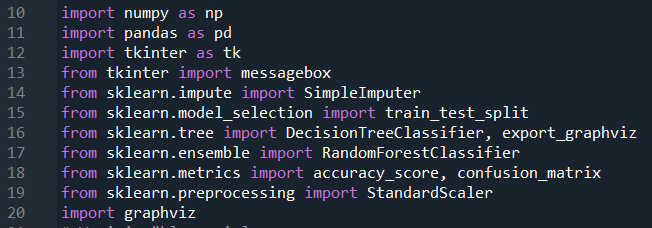
**Şeker Hastaları Tahmininde Karar Ağacı Algoritması Projesi Kod Analizi**



Numpy ve pandas: Sayısal işlemler ve veri manipülasyonu için kullanılan kütüphaneler.

tkinter: Python için bir GUI (Graphical User Interface - Grafiksel Kullanıcı Arayüzü) kütüphanesidir.

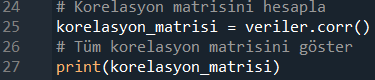
messagebox: Tkinter üzerinde mesaj kutuları oluşturmak için kullanılan bir alt modül.

SimpleImputer, train\_test\_split, DecisionTreeClassifier, RandomForestClassifier, accuracy\_score, confusion\_matrix, StandardScaler: Scikit-learn kütüphanesinden kullanılan sınıflar ve fonksiyonlardır.

graphviz: Karar ağacını görselleştirmek için kullanılır.



Diabetes verilerinin olduğu csv dosyasını çekip veriler değişkenine atıyoruz.



Burda corr() fonksiyonu kullanarak veriler dosyasının koralasyonunu hesapladık. Hesapladığımız matriksi korelasyon\_matrisi değikenine atadık ve bu değişkeni ekrana bastık.



Verideki kolanları seçerek 0 olan değerleri nan olarak değiştiriyoruz.



Kolonların ortalamasını alır ve nan olan yerlere o kolonun ortalamasını yazar. Bu sayede veri işlemesi tamamlanır ve boş olan verilerden kurtulur. Artık bu veriler üzerinde rahatça çalışılabilir.



Son kolon hariç tüm kolonları alır ve x değişkenine atadık. X değişkenleri sonucu etkileyen değişkenler.

Son kolonu alıp y değişkenine atadık. Son kolon sonuç değişkeni.



Verileri eğitim ve test olarak ayırdık.

A screen shot of a computer code

Description automatically generated

StandardScaler sınıfını kullanarak bir standartlaştırma nesnesi (sc) oluşturulur.

fit\_transform yöntemi, eğitim veri seti (x\_train) üzerinde çağrılarak, veri setinin ortalama ve standart sapma değerlerini hesaplar ve verileri standartlaştırır. Eğitim veri seti üzerinde bu işlem yapılır çünkü standartlaştırma işlemi, veri setinin istatistiksel özelliklerini kullanarak gerçekleştirilir.

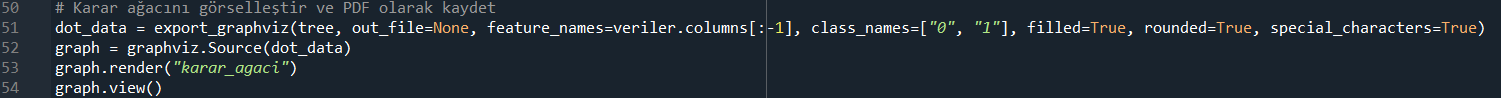
Ardından, aynı standartlaştırma nesnesi (sc) kullanılarak test veri seti (x\_test) standartlaştırılır. Ancak, fit\_transform yerine transform yöntemi kullanılır. Bu, test veri setinin sadece eğitim veri setinin istatistiksel özelliklerini kullanarak dönüştürülmesini sağlar, böylece eğitim ve test verileri aynı dönüşümü alır.



DecisionTreeClassifier sınıfından bir nesne (agac) oluşturulur. Bu, Karar Ağacı sınıflandırıcı modelini tanımlar.

random\_state=0 parametresi, modelin rastgeleliğini kontrol etmek için kullanılır. Her çalıştırıldığında aynı sonucu almak için belirli bir rastgele durumu sabitler.

fit yöntemi, oluşturulan Karar Ağacı modelini eğitmek için kullanılır. Eğitim veri seti (x\_train) ve doğru etiketlerin bulunduğu eğitim etiketleri (y\_train) bu yönteme verilir. Model, bu veri seti üzerinde eğitilerek, veri setindeki desenleri öğrenir ve gelecekteki tahminler için bu desenleri kullanır.



Bu kod, oluşturulan Karar Ağacı modelini görselleştirmek ve PDF olarak kaydetmek için kullanılır. Görselleştirme, Karar Ağacı'nın nasıl yapılandırıldığını ve hangi özelliklerin hangi karar noktalarında kullanıldığını görsel olarak gösterir



random\_state=0: Rastgele sayı üreticisinin başlangıç durumunu belirler. Bu, modelin her çalıştırıldığında aynı sonuçları üretmesini sağlar, böylece sonuçlar tekrarlanabilir.

n\_estimators=100: Ormanda kaç tane karar ağacı (decision tree) bulunacağını belirtir. Bu örnekte, orman 100 ağaçtan oluşacaktır.

max\_depth=10: Her bir karar ağacının maksimum derinliğini belirtir. Ağaçların derinliği 10 ile sınırlıdır. Bu, modelin aşırı uyum yapmasını (overfitting) önlemeye yardımcı olabilir.

criterion='gini': Hangi kriterin ağaçların dallanması sırasında kullanılacağını belirtir. Burada gini kullanılmıştır, yani Gini safsızlık ölçütü kullanılacaktır. Alternatif olarak entropy de kullanılabilir.

model.fit(x\_train, y\_train):Bu satır, modeli eğitim verisi (x\_train) ve etiketleri (y\_train) kullanarak eğitir. fit yöntemi, rastgele orman modelini verilere göre optimize eder ve modeli oluştur

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bir GUI (Grafiksel Kullanıcı Arayüzü) olşturmak için ;

1. kullanici\_girdisi\_al adında bir fonksiyon tanımlıyoruz.
2. Fonksiyon, kullanıcının girdi kutularındaki değerleri almak için bir liste döngüsü kullanır. girdi.get() metoduyla her bir girdi kutusundaki değer alınır.
3. Aldığı değerler, her bir girdi için float (ondalıklı sayı) türüne dönüştürülür ve bir liste oluşturulur.
4. Bu liste, NumPy dizisine dönüştürülür ve tek bir veri noktasını içerecek şekilde yeniden şekillendirilir (reshape).
5. Eksik değerlerin doldurulması için önceden belirlenmiş imputer nesnesi kullanılır. Eksik değerler doldurulduktan sonra, veriler standartlaştırılır (sc nesnesi ile).
6. Standartlaştırılmış girdiler, eğitilmiş makine öğrenimi modeline iletilir ve tahminde bulunmak için kullanılır (model.predict()).
7. Elde edilen tahmin sonucuna göre bir mesaj oluşturulur. Eğer tahmin edilen değer 1 ise, "Şeker hastası olabilirsiniz." mesajı oluşturulur, aksi takdirde "Şeker hastası değilsiniz." mesajı oluşturulur.
8. Sonuç mesajı bir ileti kutusu aracılığıyla (messagebox.showinfo veya messagebox.showerror) kullanıcıya gösterilir.
9. Eğer kullanıcı geçersiz bir giriş yaparsa (örneğin, sayısal olmayan bir değer girerse), ValueError hatası alınır ve bir hata ileti kutusu ile kullanıcıya "Lütfen tüm alanlara geçerli sayılar giriniz." şeklinde bir hata mesajı gösterilir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu fonksiyon, bir karmaşıklık matrisi üzerinde çalışarak sınıf oranlarını hesaplamak için tasarlanmıştır. Karmaşıklık matrisi, bir sınıflandırma modelinin performansını değerlendirmek için kullanılan bir araçtır. Genellikle doğru tahminlerin, yanlış tahminlerin ve her bir sınıfın gerçek örnek sayısını içerir.

1. Fonksiyon, karmaşıklık matrisini (confusion matrix) alır.
2. Karmaşıklık matrisinin satır sayısını (yani sınıf sayısını) shape[0] ile alır ve bu değeri sınıf\_sayısı değişkenine atar.
3. Sınıf oranlarını depolamak için karmaşıklık matrisi ile aynı boyutta bir dizi oluşturulur. Bu diziye sınıf oranlarını saklamak için kullanılacak ve her bir sınıfın oranını içerecek
4. Her bir sınıf için aşağıdaki işlemler gerçekleştirilir: a. Sınıfın toplam örnek sayısı hesaplanır. Bu, karmaşıklık matrisinin ilgili satırının toplamını alarak elde edilir (np.sum(karmaşıklık\_matrisi[i])). b. Eğer sınıfın toplam örnek sayısı 0'dan büyükse (yani sınıfın hiç örneği yoksa bu işlem yapılmaz), sınıfın oranı hesaplanır. Bu, karmaşıklık matrisinin ilgili satırının her bir elemanının toplam örnek sayısına bölünmesiyle elde edilir (karmaşıklık\_matrisi[i] / toplam\_örnekler). c. Elde edilen oranlar, sınıf\_oranları dizisinin ilgili satırına atanır.
5. Son olarak, hesaplanan sınıf oranları içeren dizi (sınıf\_oranları) geri döndürülür.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu kod, iki farklı makine öğrenimi modelinin performansını değerlendirmek için tasarlanmıştır. İşlevi şu adımları içerir:

1. İlk olarak, Karar Ağacı (agac) ve Random Forest (model) modelleri üzerinde test veri kümesi (x\_test) üzerinde tahminler yapılır.
2. accuracy\_score fonksiyonu kullanılarak her iki modelin doğruluk (accuracy) değeri hesaplanır. Doğruluk, doğru tahminlerin toplam tahminlere oranıdır.
3. Hesaplanan doğruluk değerleri ekrana yazdırılır.

Bu kod bloğunun amacı, her iki modelin test veri kümesi üzerinde ne kadar iyi performans gösterdiğini sayısal olarak değerlendirmektir. Doğruluk, genel model performansını ölçen bir metrik olup, değerlendirme sürecinde kullanıcıya objektif bir değer sunar. Örneğin, bir modelin %85 doğruluk değeri, test veri kümesindeki gözlemlerin %85'inde doğru tahmin yapabildiğini gösterir. Bu değerler, modellerin karşılaştırılması ve seçilmesi için önemli bir kriter olabilir.



confusion\_matrix(y\_test, agac\_tahmini): Bu satır, y\_test ve agac\_tahmini adlı iki sınıf etiketini alır ve bu etiketler arasındaki karşılaştırmaya dayalı bir karmaşıklık matrisi oluşturur. y\_test, test veri setindeki gerçek sınıf etiketlerini içerirken, agac\_tahmini, karar ağacı modelinin test veri seti üzerindeki tahminlerini içerir.

print("Karar Ağacı Karmaşıklık Matrisi:"): Bu satır, ekrana "Karar Ağacı Karmaşıklık Matrisi:" metnini yazdırır.

print(agac\_km): Bu satır, hesaplanan karmaşıklık matrisini ekrana yazdırır.

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

confusion\_matrix(y\_test, rf\_tahmini): Bu adımda, test setindeki gerçek sınıf etiketleri olan y\_test ile Random Forest modeli tarafından yapılan tahminler olan rf\_tahmini arasında bir karmaşıklık matrisi oluşturulur.

print("Random Forest Karmaşıklık Matrisi:"): Bu satır, ekrana "Random Forest Karmaşıklık Matrisi:" metnini yazdırır. Bu metin, karmaşıklık matrisinin sonuçlarını tanımlayan bir başlık olarak kullanılır.

print(rf\_km): Bu satır, önceden hesaplanmış olan karmaşıklık matrisini ekrana yazdırır. rf\_km değişkeni, Random Forest modelinin performansını değerlendirmek için kullanılan karmaşıklık matrisini temsil eder.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

karmaşıklık\_matrisi\_oranlarını\_hesapla(agac\_km): Bu fonksiyon, karmaşıklık matrisindeki değerleri alır ve bu değerleri sınıf oranlarına dönüştürür. Örneğin, bir hücredeki değer, gerçek pozitiflerin (TP) toplam gerçek pozitifler içindeki oranı olabilir.

print("Karar Ağacı Komşuluk Matrisi:"): Bu satır, ekrana "Karar Ağacı Komşuluk Matrisi:" metnini yazdırır. Bu metin, komşuluk matrisinin sonuçlarını tanımlayan bir başlık olarak kullanılır.

print(agac\_sınıf\_oranları): Bu satır, hesaplanan komşuluk matrisini ekrana yazdırır. agac\_sınıf\_oranları değişkeni, önceden hesaplanmış olan karmaşıklık matrisinin oranlarını temsil eder ve bu oranlar sınıf tahminlerinin ne kadar doğru olduğunu belirtir.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

karmaşıklık\_matrisi\_oranlarını\_hesapla(rf\_km): Bu fonksiyon, karmaşıklık matrisindeki değerleri alır ve bu değerleri sınıf oranlarına dönüştürür. Örneğin, bir hücredeki değer, gerçek pozitiflerin (TP) toplam gerçek pozitifler içindeki oranı olabilir.

print("Random Forest Komşuluk Matrisi:"): Bu satır, ekrana "Random Forest Komşuluk Matrisi:" metnini yazdırır. Bu metin, komşuluk matrisinin sonuçlarını tanımlayan bir başlık olarak kullanılır.

print(rf\_sınıf\_oranları): Bu satır, hesaplanan komşuluk matrisini ekrana yazdırır. rf\_sınıf\_oranları değişkeni, önceden hesaplanmış olan karmaşıklık matrisinin oranlarını temsil eder ve bu oranlar sınıf tahminlerinin ne kadar doğru olduğunu belirtir.



pencere = tk.Tk(): Bu satır, bir Tkinter uygulaması oluşturur ve bu uygulamaya ait bir ana pencereyi temsil eden bir nesne oluşturur. Tk() fonksiyonu, bir Tkinter uygulaması oluştururken kullanılır.

pencere.title("Şeker Hastalığı Kontrolü"): Bu satır, oluşturulan ana pencerenin başlığını ayarlar. Başlık, pencerenin üst kısmında görünen metni belirtir. Bu durumda, başlık "Şeker Hastalığı Kontrolü" olarak ayarlanmıştır.

metin, multimedya yazılımı, yazılım, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

etiketler = ["Hamilelik", "Glikoz", "Kan Basıncı", "Cilt Kalınlığı", "Insulin", "Vücut Kitle Endeksi", "Diyabet Soy Geçmişi", "Yaş"]: Bu satır, her bir veri girişi için bir etiket belirleyen bir liste oluşturur. Bu etiketler, her bir giriş kutusuyla ilişkilendirilecek olan veri türlerini veya girişin neyi temsil ettiğini belirtir.

girisler = []: Bu satır, oluşturulan giriş kutularını saklamak için boş bir liste oluşturur.

for i, etiket in enumerate(etiketler):: Bu döngü, etiketler listesindeki her bir öğe için bir döngü başlatır. enumerate() fonksiyonu, listedeki her bir öğenin dizinini (index) ve değerini (value) döndürür.

tk.Label(pencere, text=f"{etiket}:").grid(row=i, column=0): Bu satır, her bir etiket için bir etiket widget'i oluşturur ve bu etiketleri pencerenin sol sütununa yerleştirir. grid() yöntemi, etiketleri belirtilen satır ve sütuna yerleştirir.

girdi = tk.Entry(pencere): Bu satır, her bir giriş kutusu için bir Entry widget'i oluşturur.

girdi.grid(row=i, column=1): Bu satır, her bir giriş kutusunu pencerenin sağ sütununa yerleştirir. grid() yöntemi, giriş kutularını belirtilen satır ve sütuna yerleştirir.

girisler.append(girdi): Bu satır, her bir giriş kutusunu girisler adlı bir listeye ekler. Bu, daha sonra kullanıcı tarafından girilen verilere erişmek için kullanılabilir.

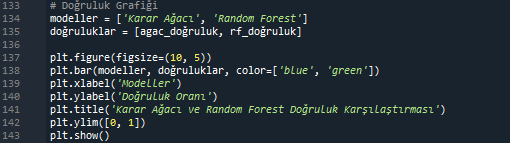


tk.Button(pencere, text="Kontrol Et", command=kullanici\_girdisi\_al): Bu satır, bir "Kontrol Et" adında bir buton oluşturur. Butonun bulunduğu pencere pencere olarak belirtilir. Butonun metni "Kontrol Et" olarak belirlenir. command parametresi, butona tıklandığında çağrılacak olan fonksiyonu belirtir. Burada kullanici\_girdisi\_al adlı bir fonksiyonun çağrılacağı belirtilmiştir.

.grid(row=len(etiketler), column=0, columnspan=2): Bu satır, butonun konumunu ve yerleşimini belirler. grid() yöntemi, butonu belirtilen satır ve sütuna yerleştirir. row ve column parametreleri, butonun konumunu belirler. columnspan=2, butonun iki sütuna yayılacağını belirtir. Bu sayede butonun genişliği, giriş kutularının genişliğine eşit olur.



mainloop() yöntemi, Tkinter uygulamasının ana döngüsünü başlatır. Bu döngü, kullanıcı etkileşimlerini algılar, olayları işler ve pencerenin güncellenmesini sağlar. Kullanıcı pencereyi kapattığında veya program sonlandırıldığında mainloop() yöntemi sona erer ve program kapanır.



modeller = ['Karar Ağacı', 'Random Forest']: Bu satır, karşılaştırılacak modellerin isimlerini içeren bir liste oluşturur.

doğruluklar = [agac\_doğruluk, rf\_doğruluk]: Bu satır, her bir modelin doğruluk oranlarını içeren bir liste oluşturur. agac\_doğruluk ve rf\_doğruluk değişkenleri, karar ağacı ve Random Forest modellerinin doğruluk oranlarını temsil eder.

plt.figure(figsize=(10, 5)): Bu satır, bir matplotlib figürü oluşturur ve figür boyutunu belirler.

plt.bar(modeller, doğruluklar, color=['blue', 'green']): Bu satır, çubuk grafiğini oluşturur. modeller listesi çubukların x eksenindeki konumlarını, doğruluklar listesi ise çubukların y eksenindeki yüksekliklerini belirler. color parametresi, her bir çubuğun rengini belirler.

plt.xlabel('Modeller') ve plt.ylabel('Doğruluk Oranı'): Bu satırlar, x ve y eksenlerine etiketler ekler.

plt.title('Karar Ağacı ve Random Forest Doğruluk Karşılaştırması'): Bu satır, grafiğin başlığını belirler.

plt.ylim([0, 1]): Bu satır, y ekseninin limitlerini belirler, bu durumda doğruluk oranlarının 0 ile 1 arasında olduğunu belirtir.

plt.show(): Bu satır, oluşturulan çubuk grafiğini görüntüler.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

plt.figure(figsize=(10, 8)): Bu satır, bir matplotlib figürü oluşturur ve figür boyutunu belirler.

plt.imshow(korelasyon\_matrisi, cmap='coolwarm', interpolation='nearest'): Bu satır, bir görüntü (image) oluşturur ve bu görüntüye korelasyon matrisini (korelasyon\_matrisi) yerleştirir. cmap parametresi, kullanılacak renk haritasını belirler, bu durumda 'coolwarm' renk haritası kullanılır. interpolation parametresi, görüntüyü nasıl ölçeklendireceğini belirler.

plt.colorbar(): Bu satır, renk skalasını (colorbar) görüntüye ekler.

plt.title('Özellikler Arasındaki Korelasyon Matrisi'): Bu satır, görüntünün başlığını belirler.

plt.xticks(range(len(korelasyon\_matrisi)), korelasyon\_matrisi.columns, rotation=90): Bu satır, x eksenindeki etiketleri (tick'leri) belirler. korelasyon\_matrisi.columns, korelasyon matrisindeki özelliklerin isimlerini içeren bir dizidir. rotation=90, etiketlerin dikey olarak yazılmasını sağlar.

plt.yticks(range(len(korelasyon\_matrisi)), korelasyon\_matrisi.columns): Bu satır, y eksenindeki etiketleri (tick'leri) belirler. korelasyon\_matrisi.columns, korelasyon matrisindeki özelliklerin isimlerini içeren bir dizidir.

plt.show(): Bu satır, oluşturulan ısı haritasını (heatmap) görüntüler.