2.6 Modeli Seçmek ve Eğitmek

Sorunu tanımladınız, verileri aldınız, araştırdınız, bir eğitim seti ve bir test seti olarak ayırdınız ve makine öğrenme algoritmanız için verilerinizi temizleyip hazırladınız artık makine öğrenmesi modelini seçip eğitmeye hazırsınız. Başlamadan önce birkaç önemli noktaya göz atalım.

1. Veriler çok büyükse, çok sayıda farklı modeli makul bir süre içerisinde eğitmek için daha küçük eğitim setleri kullanabilirsiniz.
2. Standart parametreleri kullanarak farklı kategorilerdeki birçok hızlı modeli eğitip performanslarını ölçüp karşılaştırabilirsiniz.
3. Her bir model için, k katlamalı çapraz doğrulama kullanıp performans ölçümünün ortalama ve standart sapmasını hesaplayabilir, her bir algoritma için en önemli değişkenleri analiz edebilir ve modellerin yaptığı hataların türlerini analiz edebilirsiniz.

Örneğimizde iki ayrı probleme sahibiz. Bu çok basit bir uygulama olacağı için problemleri tek tek ele alacağız. İlk problem ile başlayalım:

- Kalp Hastalarının Maksimum Kalp Atış Hızının Tahmini **Yani veri setindeki özellikleri kullanarak bir kişinin maksimum kalp atış hızını bulacağız.**

Bu soruna yanıt ararken yaş, cinsiyet, dinlenme kan basıncı, kolesterol seviyeleri, açlık kan şekeri, dinlenme elektrokardiyografik sonuçlar, egzersize bağlı anjina durumu, ST segment eğimi gibi faktörleri kullanacağız. Aralarındaki ilişkiden ötürü bu bir regresyon problemiydi. Regresyonun tanımını hatırlayalım. Değişkenler arasındaki ilişkiyi ölçerek bir sonraki değeri tahmin ederken kullanıyorduk. Biz de elimizdeki eğitim verilerine uygun öyle bir model yaratacağız ki test veri setimizi kullanarak bundan sonra hesap ödeyenlerin bırakacakları bahşiş miktarını tahmin edebileceğiz. Bu tahminleri de test verimizdeki sonuçlarla karşılaştırıp modelimizinbaşarısını görebileceğiz. Programın tamamı baştan sonra sırasıyla aşağıda verilmiştir.

Paketleri içe aktararak başlayalım.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import ListedColormap
from sklearn.model_selection import train_test_split
from scipy.stats import boxcox
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.metrics import classification_report, accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

1. PROBLEM: Kalp Hastalarının Maksimum Kalp Atış Hızının Tahmini nasıl hesaplanır?

Veri setimizi çalışma alanımıza yükleyelim.

```
df = pd.read_csv('/kaggle/input/heartcsv/heart.csv')
```

Kullanmayacağımız özellikleri silelim.

```
df.drop(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'exang', 'oldpeak', 'slope', 'ca', 'thal'], axis=1, inplace=True)
```

Veri setimizi test ve eğitim seti olarak ikiye ayıracağız. Veri setini bölmeden önce modelin eğitiminde kullanabileceğimiz formata, vektör haline dönüştürüyoruz. Bu işlemleri yaparken ve bundan sonraki süreçlerde hazır kütüphanelerden yararlanacağız. Bu algoritmaların nasıl gerçekleştirildiğini ilerleyen bölümlerde detaylı bir şekilde inceleyeceğiz.

```
from sklearn.model_selection import train_test_split

# Veri setini eğitim ve test setlerine ayırma
X = df.drop('thalach', axis=1)  # Giriş özellikleri
y = df['thalach']  # Hedef değişken

# Veri setini eğitim ve test setlerine ayırma
X_egitim, X_test, y_egitim, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

}
```

Verilerimi böldükten sonra sırada modelimizi oluşturalım.

```
from sklearn.svm import SVR

# Destek Vektör Regresyon (SVR) modelini oluşturma
model = SVR(kernel='linear')

# Modeli eğitim verileriyle eğitme
model.fit(X_egitim, y_egitim)

# Modeli test verileriyle değerlendirme
dogruluk = model.score(X_test, y_test)
print("Model Doğruluğu:", dogruluk)
```

Modelimizi oluşturduğumuza göre eğitim setimizi kullanarak artık modeli eğitebiliriz.

```
from sklearn.svm import SVR

# Destek Vektör Regresyon (SVR) modelini oluşturma
model = SVR(kernel='linear')

# Modeli eğitim verileriyle eğitme
model.fit(X_egitim, y_egitim)
```

Modeli Test Etmek

Test verilerimizdeki ödenen hesap miktarlarını kullanarak ne kadar bahşiş bırakılacağını tahmin edelim.

```
# Modelin test seti üzerinde tahmin yapması
tahminler = model.predict(X_test)
print("Modelin Tahminleri:", tahminler)
```

Modelin başarısını ölçmek için genellikle kullanılan metrikler regresyon problemleri için R-kare (R-squared) değeri veya ortalama mutlak hata (Mean Absolute Error) gibi metriklerdir. Örneğin, R-kare değerini hesaplayarak modelin başarısını ölçebiliriz:

```
from sklearn.metrics import mean_absolute_error

# Modelin ortalama mutlak hata değerini hesaplama
mae = mean_absolute_error(y_test, tahminler)
print("Modelin Ortalama Mutlak Hata [illegible]eri:", mae)
```

Bu kod, modelin test seti üzerindeki tahminlerini ve gerçek hedef değişken değerlerini kullanarak ortalama mutlak hata değerini hesaplar. Ortalama mutlak hata, modelin tahminlerinin gerçek değerlerden ne kadar uzak olduğunu gösterir ve ne kadar düşük olursa, modelin o kadar iyi olduğunu belirtir.

2.7 Sonuçları Sunmak

Modelimizi eğittik ve test ettik. Şimdi sonuçlarımızı görselleştirelim. Burada unutmayın ki program her çalıştırıldığında veri setini test ve eğitim olarak ayırma işlemini farklı gerçekleştirecektir. Buna engel olmak için veri setini bölme işlemini aşağıdaki gibi gerçekleştirebilirsiniz.

```
from sklearn.model_selection import train_test_split

# Veri setini kopyalama
veri_kopya = df.copy()

# Veri setini test ve eğitim setlerine ayırma
X = veri_kopya.drop('thalach', axis=1)  # Giriş özellikleri
y = veri_kopya['thalach']  # Hedef değişken

X_egitim, X_test, y_egitim, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Önce modelimizin başarısına bakalım.

```
Modelin Ortalama Mutlak Hata Değeri: 3.2
```

Şimdi de modelimizin grafiğini inceleyelim.