Python İle Veri Bilimi Nisan 2. Hafta Ödevi

🔍 Görev:

Kaggle sitesindeki veri setlerinden **birini** seçin ve aşağıdaki adımları izleyerek bir denetimli öğrenme modeli oluşturun.

🧠 Uygulanması Gereken Adımlar:

1. **Veri Setini Yükleyin ve İnceleyin**
    - Verinin yapısını anlayın (kaç örnek, kaç özellik, hedef değişken).
    - Eksik veri var mı? Varsa nasıl işlem yaptınız?

İlk olarak verimizi DataFrame formatına dönüştürelim, ardından ilerleyelim.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier  # Random Forest
→ Birden fazla karar ağacı (daha güçlü model)
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix, ConfusionMatrixDisplay

# --- DATAMIZI TANIYALIM ---
# Eğitim verisini oku
train_df = pd.read_csv(r"C:\Users\alper\Masaüstü\Masaüstü\ACUN MEDYA
VERİ BİLİMİ"
                       r"\ödevler\10.hafta ödev\archive\train.csv")

# Test verisini oku
test_df = pd.read_csv(r"C:\Users\alper\Masaüstü\Masaüstü\ACUN MEDYA
VERİ BİLİMİ\ödevler\10.hafta ödev\archive\test.csv")
```

Veri yapısını anlayalım ve eksik veri kontrolü yapalım.

```
  # Veriyi inceleyelim
print("\n", " " * 30, "İlk 5 veriye bakalım\n")
print(train_df.head())
print("\n", ("-" * 150), "\n")
  # Satır, sutün sayısını öğrenelim
print(" " * 30, "Kolonlar\n")
print(train_df.shape)
print("\n", ("-" * 150), "\n")

  # Verinin yapısını tanıyalım
print(" " * 30, "Veri yapısı\n")
print(train_df.info())
print("\n", ("-" * 150), "\n")

   # Eksik var mı yok mu kontrol edelim
print(" " * 30, "Eksik veri kontrolü\n")
print(train_df.isnull().sum())
```

Şimdi çıktılarımızı inceleyelim;

```
                          İlk 5 veriye bakalım

   battery_power  blue  clock_speed  ...  touch_screen  wifi  price_range
0            842     0          2.2  ...             0     1            1
1           1021     1          0.5  ...             1     0            2
2            563     1          0.5  ...             1     0            2
3            615     1          2.5  ...             0     0            2
4           1821     1          1.2  ...             1     0            1

[5 rows x 21 columns]

 --------------------------+-------------------------------------------------------------

                          Satır, Sutün Sayısı

(2000, 21)
```

```
                          Veri yapısı

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
 16  talk_time      2000 non-null   int64
 17  three_g        2000 non-null   int64
 18  touch_screen   2000 non-null   int64
 19  wifi           2000 non-null   int64
 20  price_range    2000 non-null   int64
dtypes: float64(2), int64(19)
```

```
Eksik veri kontrolü

battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

---

2. **Veriyi Temizleyin ve Hazırlayın**
   - Gerekirse kategorik değişkenleri sayısal forma çevirin (OneHotEncoder, LabelEncoder vs).
   - Giriş ve çıkış değişkenlerini ayırın.
   - Eğitim-test setlerine ayırın (%80-%20 gibi).

Burada test ve eğitim setine ayırdık

```
# --- HEDEFLERİ BELİRLEME ---
x = train_df.drop("price_range", axis=1)  # Bağımsız Değişkenler (Girdiler: battery_power, dual_sim vs.)
y = train_df["price_range"]  # Hedef Değişken (price_range)

# --- EĞİTİM & TEST SETİNE AYIRMA ---
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

---

3. **Model Seçin ve Eğitin**

   - Lojistik regresyon, karar ağaçları, k-en yakın komşu, SVM, rastgele orman gibi bir denetimli öğrenme algoritması seçin.
   - Modeli eğitin.

Burada model olarak denetimli öğrenme algoritması olan Random Forest'ı seçtik. Çünkü tek bir karar ağacı yerine **birden fazla (rastgele seçilen) karar ağacını** birlikte kullanır. Bu, modelin çeşitliliğini ve doğruluğunu artırır.

Şimdi modeli eğitmeye geçelim;

```
random_model = RandomForestClassifier(n_estimators=100, max_depth=10,
random_state=42)  # Model parametreleri
random_model.fit(x_train, y_train)
random_pred = random_model.predict(x_test)

random_accuracy = accuracy_score(y_test, random_pred)
print("\n", ("-" * 150), "\n")
print(f"Random Forest Doğruluğu: {random_accuracy:.4f}\n")
```

Modeli eğittik şimdi çıktımıza bakalım;

```
 ------------------------------------------------------------------------------------------------------------

Random Forest Doğruluğu: 0.8925


 ------------------------------------------------------------------------------------------------------------
```

Çıktımıza göre modelin test verilerindeki örneklerin yaklaşık %89.25'ini doğru tahmin ettiği anlamına gelir. Bu, modelin öğrenme sürecinde veriye oldukça iyi uyum sağladığını ve doğru bir model seçimi yaptığımızı göstermektedir.

4. **Modeli Değerlendirin**
    - Doğruluk, F1 skoru, karışıklık matrisi gibi metriklerle değerlendirme yapın.
    - Gerekirse modelin başarısını artırmak için hiperparametre optimizasyonu deneyin.

Şimdi gelelim model değerlendirmeye;

```
target_names = ["Ekonomik Segment", "Orta Segment", "Üst Segment",
"Lüks Segment"]

# --- CLASSIFICATION REPORT ---
report = classification_report(y_test, random_pred,
target_names=target_names)
print("\n", ("-" * 150), "\n")
print("\nRandom Forest - Sınıflandırma Raporu:\n")
print(report)

# --- KARIŞIKLIK MATRİSİ (CONFUSION MATRIX) ---
conf_mat = confusion_matrix(y_test, random_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=conf_mat,
display_labels=target_names)
disp.plot(cmap=plt.cm.Oranges)
plt.title("Random Forest - Confusion Matrix")
plt.show()
```

Çıktılarımız;

1) Sınıflandırma Raporu

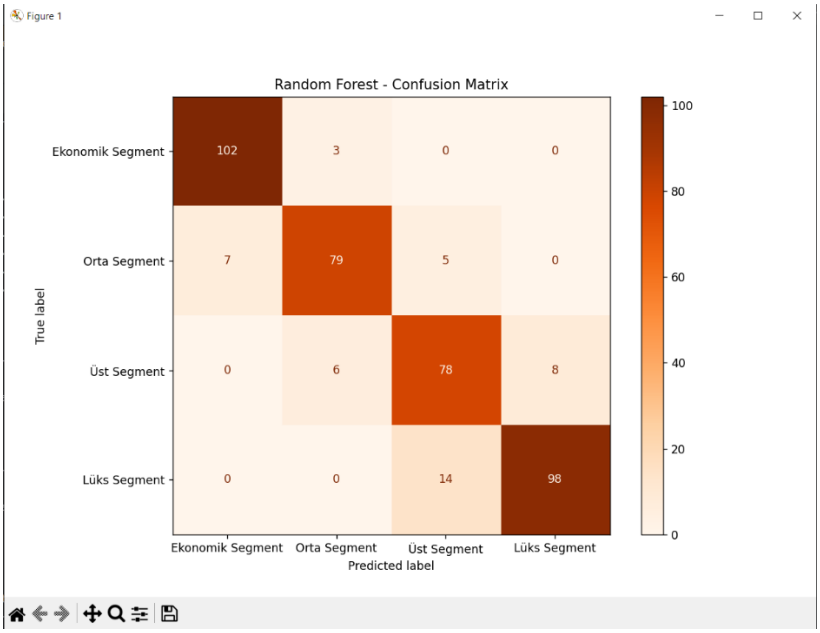
```
Random Forest - Sınıflandırma Raporu:

                precision    recall  f1-score   support

Ekonomik Segment       0.94      0.97      0.95       105
    Orta Segment       0.90      0.87      0.88        91
     Üst Segment       0.80      0.85      0.83        92
    Lüks Segment       0.92      0.88      0.90       112

        accuracy                           0.89       400
       macro avg       0.89      0.89      0.89       400
    weighted avg       0.89      0.89      0.89       400
```

2) KARIŞIKLIK MATRİSİ



1.) Sınıflandırma Raporu

1.1) Ekonomik Segment:

- **Precision**: 0.94
  - Model, "Ekonomik Segment" sınıfını doğru tahmin ettiğinde, %94 doğrulukla tahmin ediyor. Yani, modelin "Ekonomik Segment" tahminleri büyük ölçüde doğru.
- **Recall**: 0.97
  - Model, gerçek "Ekonomik Segment" örneklerinin %97'sini doğru şekilde tahmin etmiş. Yani, çoğu gerçek "Ekonomik Segment" örneği doğru şekilde yakalanmış.
- **F1-score**: 0.95
  - Precision ve recall arasındaki dengeyi yansıtan F1 skoru oldukça yüksek, %95. Bu, modelin "Ekonomik Segment"i doğru sınıflandırmada güçlü olduğunu gösteriyor.

1.2) Orta Segment:

- **Precision**: 0.90
  - Modelin "Orta Segment" için yaptığı tahminlerin %90'ı doğru.
- **Recall**: 0.87
  - Gerçek "Orta Segment" örneklerinin %87'si doğru tahmin edilmiş. Bu, modelin bu sınıfta biraz daha fazla hata yaptığını gösteriyor.
- **F1-score**: 0.88
  - F1 skoru, precision ve recall arasında bir dengeyi gösteriyor ve %88 gibi iyi bir değere sahip.

1.3) Üst Segment:

- **Precision**: 0.80
  - "Üst Segment" sınıfı için modelin tahmin doğruluğu biraz daha düşük, %80. Model, "Üst Segment" tahminlerinde %20'lik bir hata payı bırakıyor.
- **Recall**: 0.85
  - Gerçek "Üst Segment" örneklerinin %85'i doğru tahmin edilmiş.
- **F1-score**: 0.83
  - "Üst Segment" için F1 skoru biraz daha düşük (%83), çünkü hem precision hem de recall bu sınıf için biraz daha düşük seviyelerde.

1.4) Lüks Segment:

- **Precision**: 0.92
  - "Lüks Segment" için modelin tahmin doğruluğu oldukça yüksek, %92.
- **Recall**: 0.88
  - Gerçek "Lüks Segment" örneklerinin %88'i doğru tahmin edilmiş.
- **F1-score**: 0.90
  - F1 skoru oldukça iyi, %90.

1.5) Accuracy: %89

- Modelin genel doğruluğu %89. Bu, genellikle iyi bir performans gösteriyor.

1.6) Macro Average: %89

- **Macro average**: Her sınıfın metriklerinin ortalamasıdır. Burada precision, recall ve F1-score her sınıf için aynı şekilde %89 civarındadır.

1.7) Weighted Average: %89

- **Weighted average**: Her sınıfın örnek sayısına göre ağırlıklı ortalama. Bu da %89 civarında, yani modelin genel başarısı tutarlı.

- **Sonuç olarak** Model, **Ekonomik Segment** ve **Lüks Segment** üzerinde oldukça başarılı. **Orta Segment** ve **Üst Segment** için çokta başarılı denmez.

2) KARIŞIKLIK MATRİSİ

1. **Ekonomik Segment**:
   - **102 doğru tahmin** (doğru sınıflandırılan örnekler).
   - **3 yanlış tahmin** (orta segment olarak sınıflandırılanlar).
   - **0 yanlış tahmin** (diğer segmentlere.)
2. **Orta Segment**:
   - **79 doğru tahmin** (doğru sınıflandırılan örnekler).
   - **7 yanlış tahmin** (ekonomik segment olarak sınıflandırılanlar).
   - **5 yanlış tahmin** (üst segment olarak sınıflandırılanlar).
3. **Üst Segment**:
   - **78 doğru tahmin** (doğru sınıflandırılan örnekler).
   - **6 yanlış tahmin** (orta segment olarak sınıflandırılanlar).
   - **8 yanlış tahmin** (lüks segment olarak sınıflandırılanlar).
4. **Lüks Segment**:
   - **98 doğru tahmin** (doğru sınıflandırılan örnekler).
   - **14 yanlış tahmin** (üst segment olarak sınıflandırılanlar).
   - **0 yanlış tahmin** (diğer segmentlere.)

Kısaca Sonuç;

Model en iyi **Ekonomik** ve **Lüks Segment**'te başarılı tahminler yapmış.
Orta ve özellikle **Üst Segment**'te karışıklık daha fazla.
Yani model genel olarak iyi ama orta-üst gruplarda biraz zorlanıyor.

---

5. **Sonuçları Yorumlayın**
    - Model iyi mi performans gösterdi?
    - Veride hangi değişkenler önemliydi?
    - Daha iyi bir sonuç almak için ne yapılabilir?

Model iyi mi performans gösterdi?
Evet, %89 doğrulukla oldukça iyi performans gösterdi.

Veride hangi değişkenler önemliydi?
En önemli değişkenler: battery_power, ram, px_height, px_width.

Daha iyi bir sonuç almak için ne yapılabilir?
Hiperparametre optimizasyonu yapılabilir, daha fazla model denenebilir.