Ad-Soyad: Hüseyin Can Akkaya

E-posta: hcaakkaya@gmail.com

Fiziksel Tıp ve Rehabilitasyon Verisi Üzerinde EDA & Veri Hazırlama Dokümantasyonu

1. Proje Amacı

Bu çalışmada 2235 gözlem ve 13 özellikten oluşan Fiziksel Tıp ve Rehabilitasyon veri seti üzerinde Exploratory Data Analysis (EDA) yapılmış, veri temizlenmiş, dönüştürülmüş ve tahmine dayalı modellere hazır hale getirilmiştir.
Hedef değişken: TedaviSuresi

2. Gerekli Kütüphanaleri Yükleme

pandas, numpy, matplotlib.pyplot ve seaborn kütüphanalerini “import” komutuyla yüklüyoruz.

3. Veri Yükleme & Okuma

data adındaki değişkenimize = pd.read_excel(“your_dataset.xlsx”) komutu ile veriyi yüklüyoruz.

4. Veri hakkında bilgi alma

data.head() ile ilk 5 satıra bakıyoruz.

data.info() ile kaç satır kaç sütun, sütun isimleri ve type vb. gibi bilgileri öğreniyoruz.

data.describe() ile sütunlar arasındaki korelasyon değerlerini görüyoruz.

5. Boş değerleri bulma

data.isnull().sum() komutu ile hangi sütunda ne kadar eksik değerimiz var öğreniyoruz.

6. Veriyi kopyalama

Orijinal verimize dokunmamak için verimizi df içine data.copy() ile verimizin kopyasını atıyoruz.

7. String sütunları floata çevirme

TedaviSuresi ve UygulamaSuresi sütunlarını içindeki değerleri floata çeviriyoruz ve yeni değişkenlerine atıyoruz. (Aşağıda örnek bir değer ve nasıl yapıldığı yer almaktadır.)

- TedaviSuresi = 5 seans
- UygulamaSuresi = 20 dakika

bu sütunlardaki seans ve dakika kısımlarını çıkarıyoruz.

- df["TedaviSuresi_num_seans"] = df["TedaviSuresi"].str.extract("(\d+)").astype(float)
- df["UygulamaSuresi_num_dk"] = df["UygulamaSuresi"].str.extract("(\d+)").astype(float)

Daha sonra TedaviSuresi ve UygulamaSuresi sütunlarını siliyoruz.

- df = df.drop(columns = ["TedaviSuresi", "UygulamaSuresi"])

8. Veri hakkında daha fazla bilgi edinmek için groupby kullanma

Veri hakkında daha fazla bilgi öğrenmek için verimizi groupby komutu ile Cinsiyet, Yas, KanGrubu, Uyruk gibi sütunlar gruplara ayırıyoruz. (Aşağıda kodlar ve bir örnek mevcuttur.)

```
In [15]: df_grouped_gender = df.groupby("Cinsiyet")
```

```
In [16]: df_grouped_gender.count()
```

Out[16]:

Cinsiyet	HastaNo	Yas	KanGrubu	Uyruk	KronikHastalik	Bolum	Alerji	Tanilar	TedaviAdi	UygulamaYerleri	TedaviSuresi_num_seans	UygulamaSuresi_num_c
Erkek	792	792	503	792	601	792	447	775	792	691	792	79
Kadın	1274	1274	959	1274	907	1263	754	1216	1274	1162	1274	127

9. Yas grupları için sınırlar ve etiketler oluşturma

```
In [27]: bins = [0, 16, 26, 40, 60, 75, np.inf]
         labels = ["0-15", "16-25", "26-39", "40-59", "60-74","75+"]
```

- bins: yaş gruplarının sınırlarını belirliyor.
- labels: bu aralıkların isimlerini veriyor.

9.1. Yas grupları sütununu oluşturma

```
In [28]: df["age_group"] = pd.cut(
             df["Yas"],
             bins = bins,
             labels = labels,
             right=False, # sol dahil, sağ dahil değil [0,16) gibi
             include_lowest=True
         )
```

- pd.cut() -> sayısal bir değişkeni kategorik aralıklara ayırır.
- right=False -> aralıkların soldaki değerini alır ancak sağdaki değerini dahil etmez.

9.2. Yas gruplarına göre sayım ve özet istatistikler

```
In [29]: df_grouped_age = df.groupby("age_group", observed=True)["Yas"].count()
         ozet = df.groupby("age_group", observed=True).agg({
             "Yas": ["count", "mean", "min", "max"],
         })

         print(df_grouped_age)
         print("**************************************")
         print(ozet)
```

```
age_group
0-15       63
16-25     105
26-39     471
40-59    1157
60-74     346
75+        93
Name: Yas, dtype: int64
**************************************
            Yas
          count       mean min max
age_group
0-15         63  10.396825   2  15
16-25       105  21.866667  16  25
26-39       471  34.847134  26  39
40-59      1157  48.220398  40  59
60-74       346  66.690751  60  74
75+          93  81.139785  75  92
```

- groupby("age_group") -> yaş gruplarına göre grupla.
- .count() -> her yaş grubundaki kişi sayısını verir.

Her yaş grubu için:

- Kaç kişi (count)
- Ortalama yaş (mean)
- Minimum yaş (min)
- Maksimum yaş (max)

9.3. Yas kategorisi için fonksiyon ve uygulanması

```
In [30]: def age_category(yas):
             if yas <= 15:
                 return "Child"
             elif 16 <= yas <= 25:
                 return "Young Adult"
             elif 26 <= yas <= 39:
                 return "Early Adulthood"
             elif 40 <= yas <= 59:
                 return "middle age"
             elif 60 <= yas <= 74:
                 return "young old"
             else:
                 return "old age"
```

```
df["age_category"] = df["Yas"].apply(age_category)
```

- Her satırdaki Yas değerine fonksiyonu uygular.

9.4. Yeni kategorilerin gruplanması

```
In [32]: df_grouped_age_category = df.groupby("age_category")["age_category"].count()
         print(df_grouped_age_category)
```

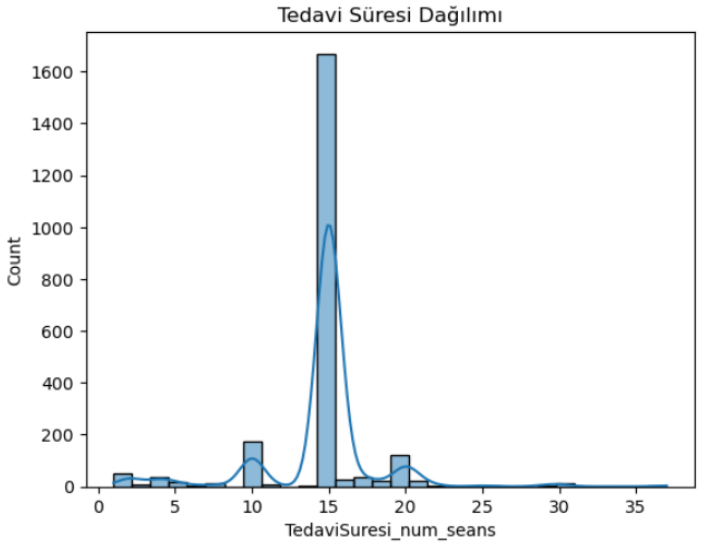
```
age_category
Child                63
Early Adulthood     471
Young Adult         105
middle age         1157
old age              93
young old           346
Name: age_category, dtype: int64
```

- age_category sütununa göre gruplar.
- Her kategoride kaç kişi olduğunu sayar.

10. Görselleştirmeler

Tedavi Süresi Dağılımı

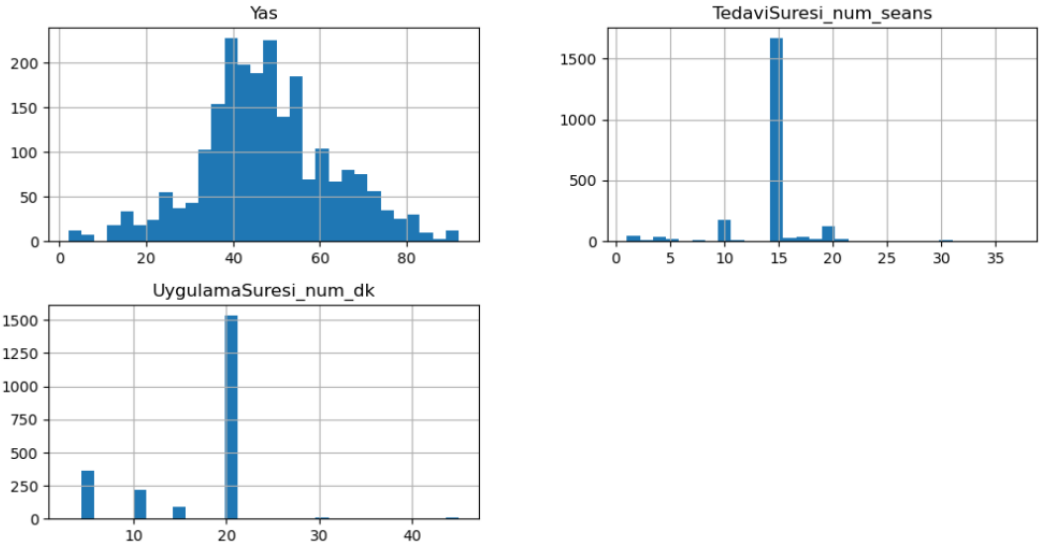
```
In [33]: sns.histplot(df["TedaviSuresi_num_seans"], kde=True, bins=30)
         plt.title("Tedavi Süresi Dağılımı")
         plt.show()
```



- histplot -> histogram çizer.
- bins=30 -> veriyi 30 eşit aralığa bölüyor.
- kde=True -> histogram üzerine bir yoğunluk eğrisi (Kernel Density Estimation) çiziyor.

Sayısal Değişkenlerin Dagılımı

```
In [34]: #Sayısal Değişkenlerin Dağılımı
         num_cols = ["Yas", "TedaviSuresi_num_seans", "UygulamaSuresi_num_dk"]

         df[num_cols].hist(bins=30, figsize=(12,6))
         plt.show()
```



- df[num_cols].hist() -> listedeki her sayısal değişken için ayrı histogram çizer.
- figsize=(12,6) -> grafiklerin boyutunu büyütüyor.

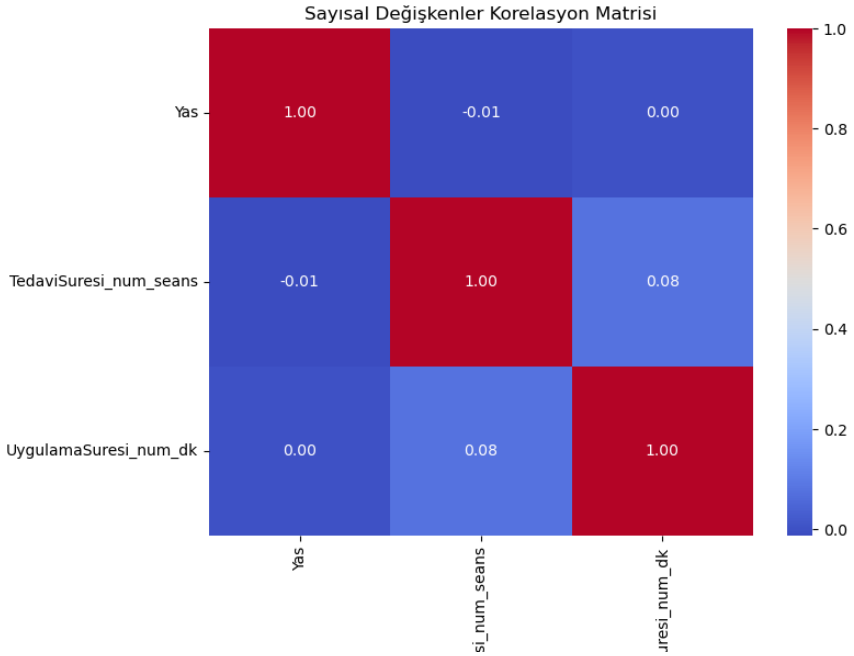
Kategorik Değişkenlerin Dağılımı

```
In [35]: #Kategorik değişkenlerin dağılımı

cat_cols = ["Cinsiyet","KanGrubu","Uyruk", "Bolum","Alerji", "UygulamaYerleri"]

for col in cat_cols:
    plt.figure(figsize=(12,6))
    sns.countplot(x=col, data=df, order=df[col].value_counts().index)
    plt.title(f"{col} Dağılımı")
    plt.xticks(rotation=90)
    plt.show()
```

- Döngü ile her kategorik değişkenin frekans grafiğini (bar chart) çiziyor.
- order=df[col].value_counts().index -> sütunları en çoktan aza doğru sıralıyor.
- rotation=90 -> etiketleri 90 derece döndürüp üst üste binmelerini engelliyor.

Korelasyon Matrisi (Sayısallar Arası)

```
In [36]: plt.figure(figsize=(8,6))
sns.heatmap(df[num_cols].corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Sayısal Değişkenler Korelasyon Matrisi")
plt.show()
```

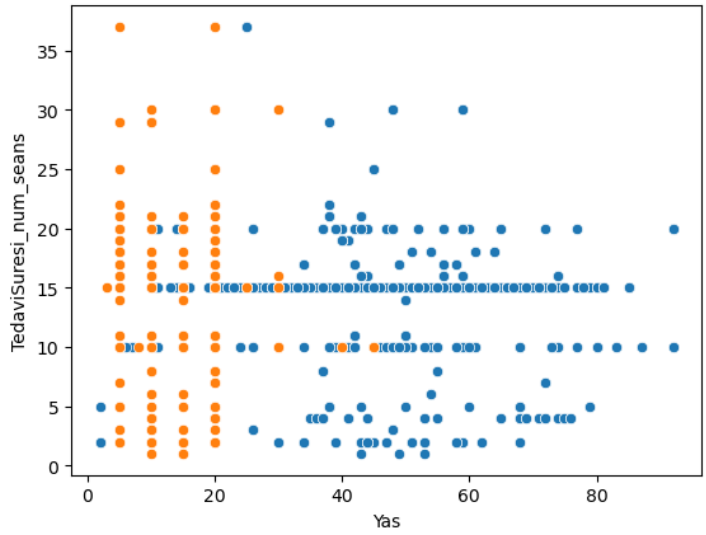


- df[num_cols].corr() -> sayısal sütunlar arası korelasyonu hesaplar.
- heatmap(..., annot=True) -> her hücreye korelasyon değerini yazar.
- cmap="coolwarm" -> renk paleti seçimi (kırmızı = negatif, mavi = pozitif).

Scatter Plotlar

```
In [37]: sns.scatterplot(x="Yas", y="TedaviSuresi_num_seans", data=df)
         sns.scatterplot(x="UygulamaSuresi_num_dk", y="TedaviSuresi_num_seans", data=df)
```

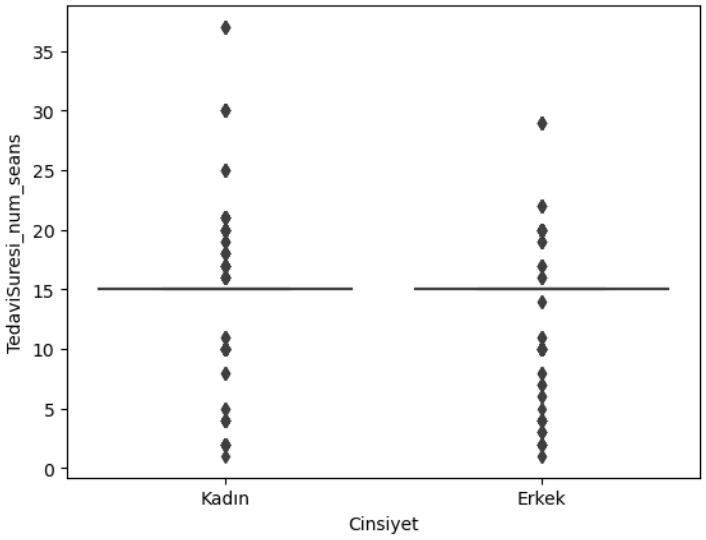
```
Out[37]: <Axes: xlabel='Yas', ylabel='TedaviSuresi_num_seans'>
```



Boxplot (Kategorik vs Sayısal)

```
In [38]: sns.boxplot(x="Cinsiyet", y="TedaviSuresi_num_seans", data=df)
```

```
Out[38]: <Axes: xlabel='Cinsiyet', ylabel='TedaviSuresi_num_seans'>
```



- Kategorik (Cinsiyet) ile sayısal (Tedavi süresi) arasındaki dağılımı gösteriyor.
- Boxplot: ortanca (median), çeyrekler, aykırı değerler.
- Amaç: Erkek/Kadın arasında tedavi süresi farklı mı görmek.

11.Eksik Değerlerin Doldurulması

```
# Eksik değerlerin doldurulması
df["Cinsiyet"].fillna(df["Cinsiyet"].mode()[0], inplace=True)
```

- Cinsiyet sütunundaki boş değerler, **en sık görülen değer** (mode) ile dolduruluyor.

```
# Çok fazla veri eksik olduğu için "Bilinmiyor" ile dolduruyoruz.
df["Alerji"].fillna("Bilinmiyor", inplace=True)
df["KanGrubu"].fillna("Bilinmiyor", inplace=True)
df["KronikHastalik"].fillna("Bilinmiyor", inplace=True)
df["UygulamaYerleri"].fillna("Bilinmiyor", inplace=True)
```

- Bu sütunlarda eksik değer oranı yüksek -> en mantıklı strateji “Bilinmiyor” kategorisi eklemek.

```
# Az veri eksik olduğu için en sık değer ile dolduruyoruz.
df["Bolum"].fillna(df["Bolum"].mode()[0], inplace=True)
df["Tanilar"].fillna(df["Tanilar"].mode()[0], inplace=True)
```

- Bu sütunlarda eksik oranı düşük -> mod ile doldurmak yeterli.

```
df.isnull().sum()
```

```
HastaNo                  0
Yas                      0
Cinsiyet                 0
KanGrubu                 0
Uyruk                    0
KronikHastalik           0
Bolum                    0
Alerji                   0
Tanilar                  0
TedaviAdi                0
UygulamaYerleri          0
TedaviSuresi_num_seans   0
UygulamaSuresi_num_dk    0
age_group                0
age_category             0
dtype: int64
```

- Eksik değer kalıp kalmadığını kontrol ediyor.

12. Feature Engineering ve One-Hot Encoding işlemleri

Kronik Hastalık, Tanı, Alerji, Uygulama Yeri Özellikleri ve Tedavi Yoğunluğu

```
# Kronik hastalık sayısı
df["KronikHastalikSayisi"] = df["KronikHastalik"].apply(lambda x: len(str(x).split(",")) if x!="Bilinmiyor" else 0)

# Örnek: Yaygın hastalıklar için flag
df["Diyabet"] = df["KronikHastalik"].str.contains("Diyabet", na=False).astype(int)
df["Hipertansiyon"] = df["KronikHastalik"].str.contains("Hipertansiyon", na=False).astype(int)
```

- KronikHastalik sütunu virgül ile ayrılmış liste → hastalık sayısını hesaplıyor.
- Eğer “Bilinmiyor” ise 0.
- Eğer satırda “Diyabet” geçiyorsa → 1, yoksa → 0.
- Aynısı Hipertansiyon için.

```
# Tani sayısı
df["TaniSayisi"] = df["Tanilar"].apply(lambda x: len(str(x).split(",")) if x!="Bilinmiyor" else 0)

# Örnek grup: Omurga rahatsızlığı
df["OmurgaSorunu"] = df["Tanilar"].str.contains("Bel|Boyun|Omurga", na=False).astype(int)
```

- Bir satırdaki tanıların sayısını hesaplıyor.
- Eğer tanılar içinde Bel, Boyun, Omurga gibi kelimeler geçiyorsa 1 geçmiyorsa 0 yazdırılıyor.

```
df["AlerjiVarMi"] = (df["Alerji"]!="Bilinmiyor").astype(int)
```

```
df["UygulamaYeriSayisi"] = df["UygulamaYerleri"].apply(lambda x: len(str(x).split(",")) if x!="Bilinmiyor" else 0)
```

```
df["TedaviYogunlugu"] = df["TedaviSuresi_num_seans"] * df["UygulamaSuresi_num_dk"]
```

```
df = pd.get_dummies(df, columns=["Cinsiyet","KanGrubu","Uyruk","Bolum","Alerji","UygulamaYerleri"], drop_first=True)
```

- Eğer Alerjisi varsa 1 yoksa 0 yazdırılıyor. Bilinmiyor yazıyorsa da 0 yazdırılıyor
- Bir satırda kaç farklı uygulama yeri olduğu sayılıyor.
- Seans sayısı × Seans başına uygulama süresi
- Böylece toplam tedavi yükünü temsil eden yeni bir sayısal özellik üretiliyor.
- Kategorik değişkenler **dummy sütunlara** çevriliyor.
- Örn. Cinsiyet -> Cinsiyet_Erkek (1/0)
- drop_first=True -> “dummy trap” (çoklu doğrusal bağımlılık) önleniyor.