Enes Sevim

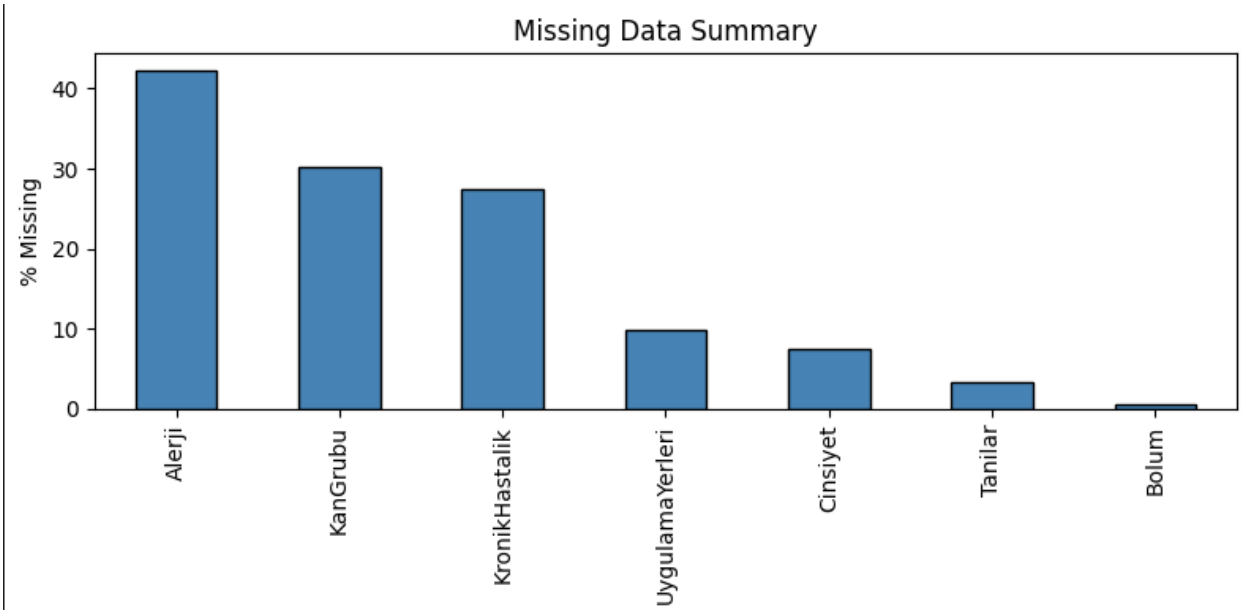
sevim.enes06@gmail.com

Pusula Academy Data Science Case Study

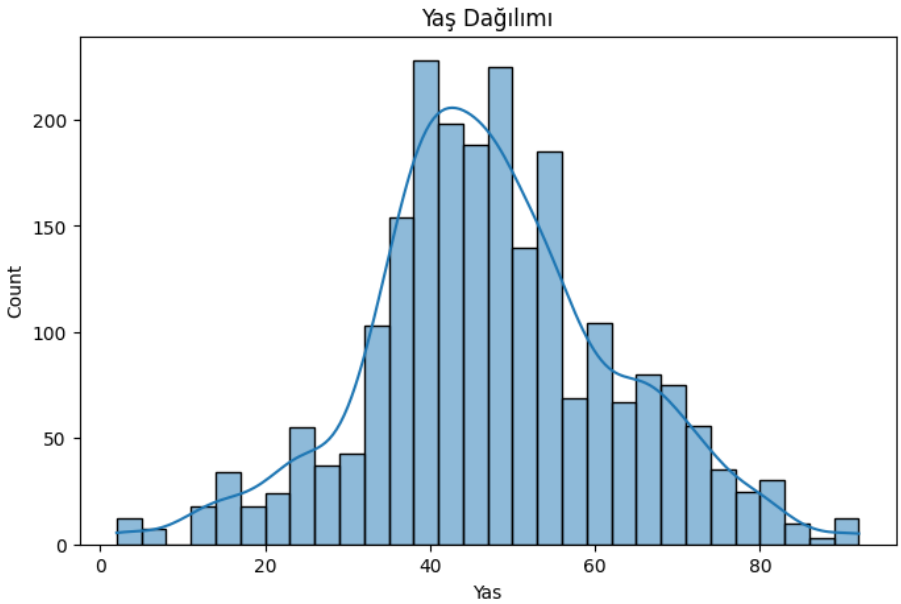
Explanatory Data Anaylsis eda.ipynb dosyasında, preprocess işlemi preprocess.py dosyasında gerçekleşmiştir.

1. Exploratory Data Analysis

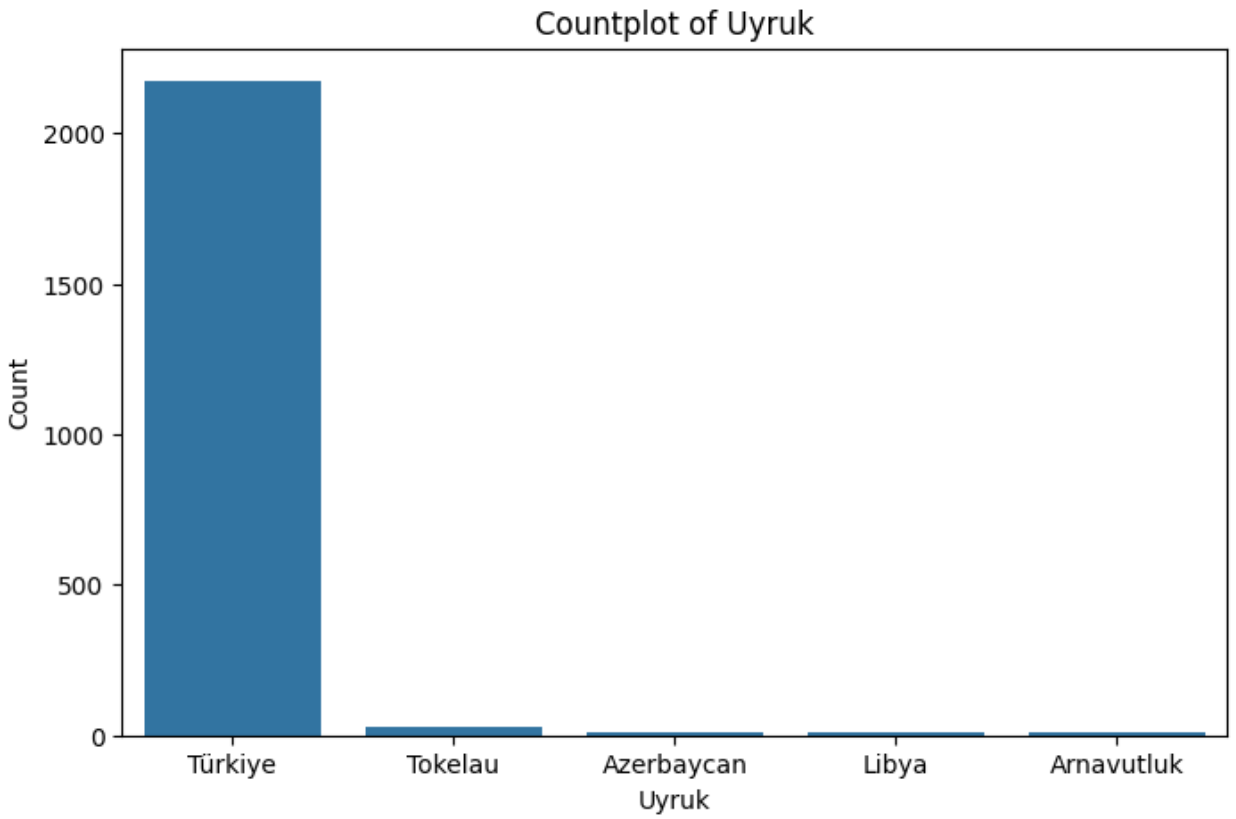
Verisetini incelediğimizde çoğu verinin object olduğunu ve eşsiz değerlerin fazla olduğunu gözlemliyoruz.



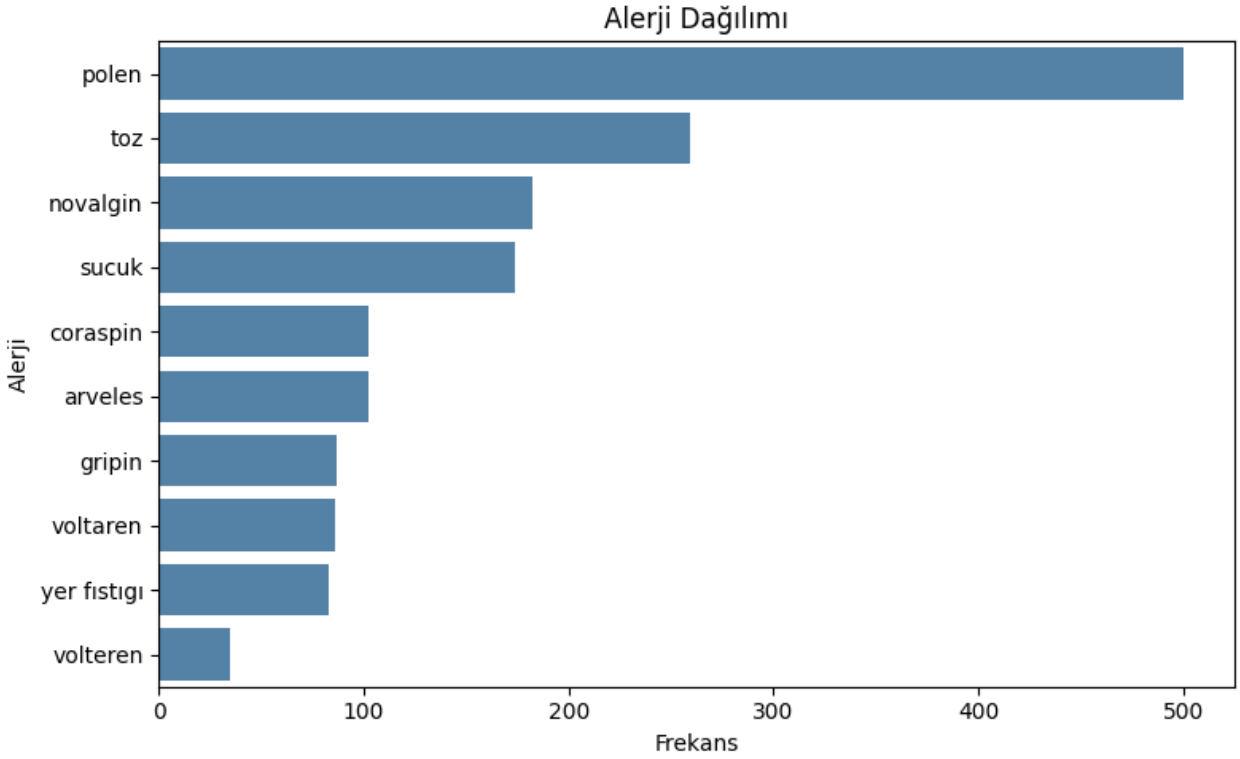
Eksik değerlere bakacak olursak Alerji ve Kronik hastalık hastada olmama ihtimali var bu değerleri "bilinmiyor" ile doldurabiliriz. Ayrıca görece az oranda olan bölüm ve Uygulama Yerleri verilerini TedaviAdi'na göre gruplayarak tahmin edebiliriz.



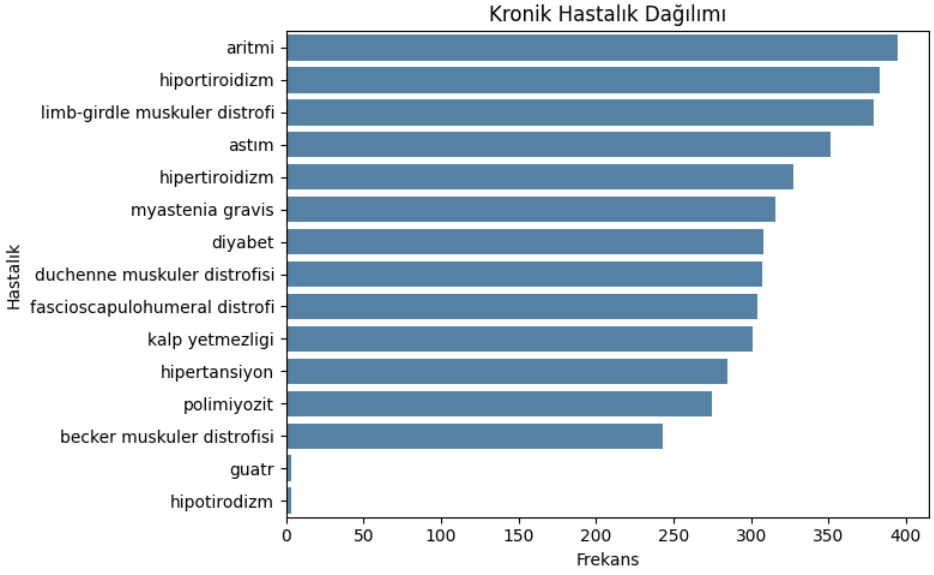
Yaş dağılımı standart dağılıma uygun gözüküyor. Yaş grupları halinde kullanılmaya uygun (Çocuk, Genç Yetişkin, Yaşlı).



Türkiye verisinin fazlalığından diğer verilerin etkisini azaltacaktır belki uyruk verisi kaldırılabilir.



Alerji verilerinde aynı sözcükler farklı şekillerde yazılmış, TOZ, toz gibi. Bunun için hepsini küçük yapabiliriz. Volteren sözcüğündeki yazım yanlışı yapılan veriler ise "voltaren" verilerine aktarılabilir.



Kronik Hastalık Dağılımı da benzer şekilde aynı sözcüklerin farklı yazımını içeriyor. Bu yüzden küçük yapabiliriz. Hipotirodizm ve hiportirodizm yine aynı veriyi ima ediyor.

```
diagnosis = df['Tanilar'].apply(normalize_text).explode().value_counts()
print("Total Diagnoses:", diagnosis.shape[0])
diagnosis.head(25)
```

✓ 0.0s

```
Total Diagnoses: 279

Tanilar
dorsalji                          800
diger                             769
tanımlanmamıs                     408
omuzun darbe sendromu             273
intervertebral disk bozuklukları  270
lumbosakral bolge                 266
servikotorasik bolge              211
servikal bolge                    148
eklem agrısı                      113
birden fazla yer                   75
fibromiyalji                       70
simdiki                            62
meniskus yırtıgı                   62
ekstremite agrısı                  61
```

Tanıları ve TedaviAdi verilerini incelediğimizce içerisinde çok fazla eşsiz değerin olduğunu gözlemliyoruz. Burada bir N değeri seçip en yaygın tanıları verisetine alıp diğerlerini “diğer” olarak işaretleyebiliriz. Ya da bir domain expert ile bu veriler incelenip sınıflandırılabilir.

```
treatment = df['TedaviAdi'].apply(normalize_text).explode().value_counts()
print("Total Treatments:", treatment.shape[0])
treatment.head(25)
```

✓ 0.0s

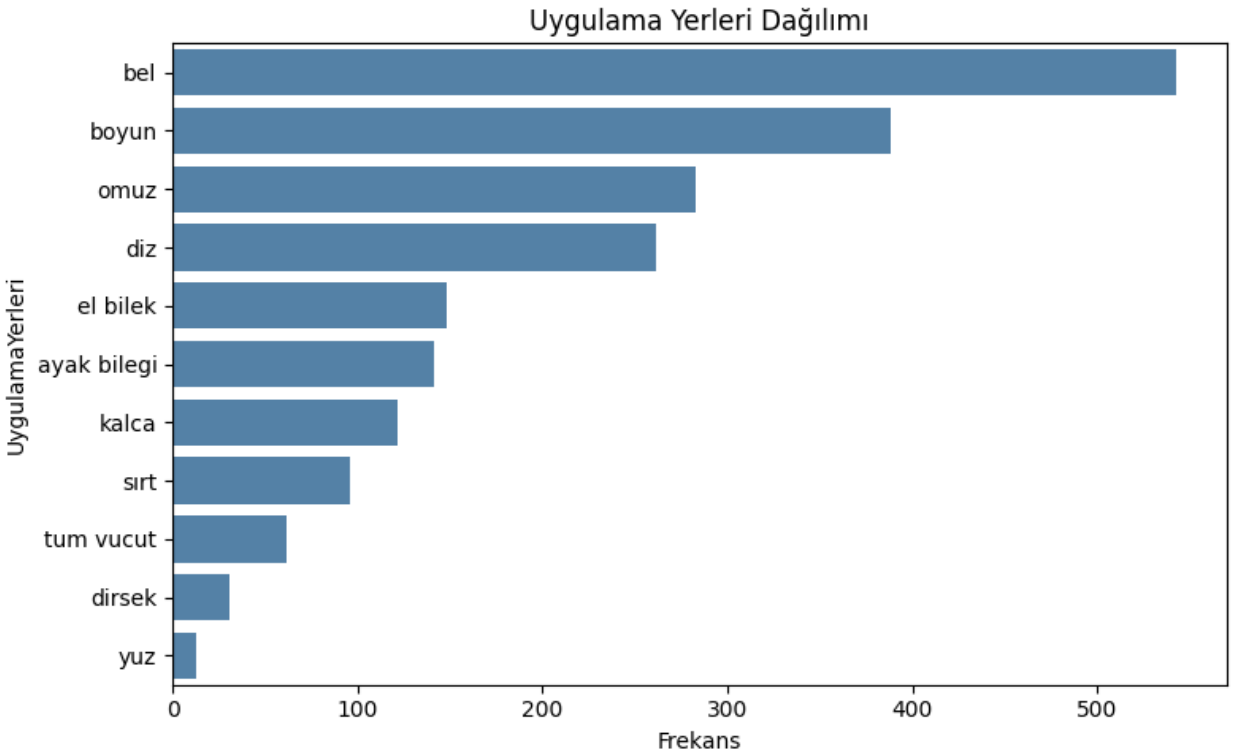
```
Total Treatments: 233

TedaviAdi
dorsalji -boyun+trapez            231
iv disk bozuklugu-bel             200
dorsalji 1                        140
dorsalji-bel                      120
sol omuz impingement              105
gonartroz-meniskopati              95
sag omuz impingement               80
boyun-trapezz                      60
dorsalji-dorsal                    56
alt ekstremite atrofi-bilateral    37
dorsalji boyun 1                   35
impingement sag                    32
sol diz implantı reh               20
```

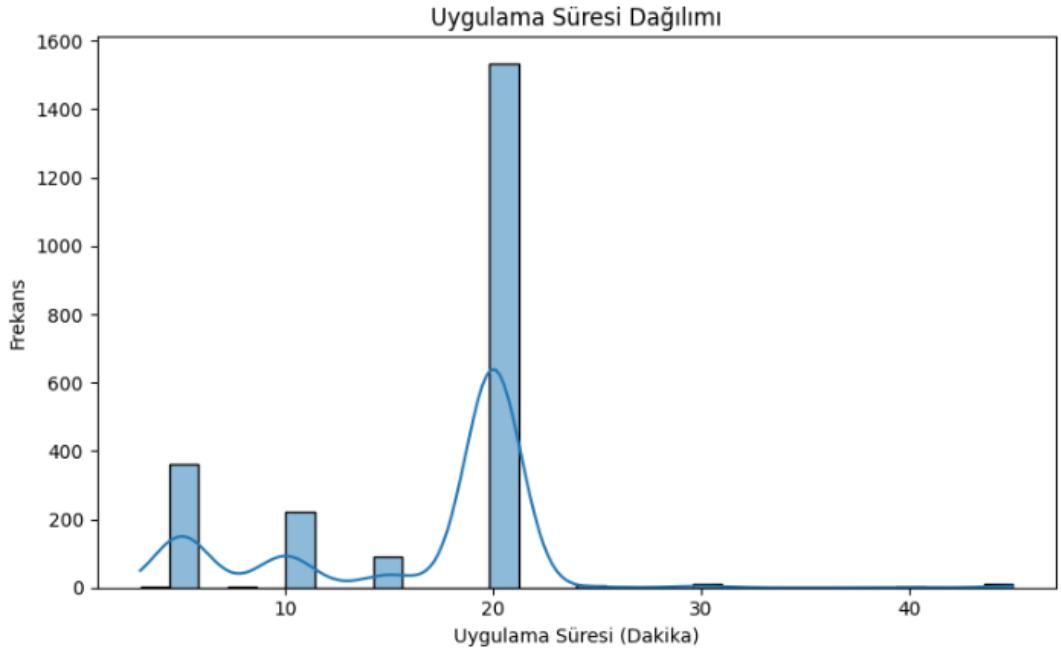
```
df['TedaviSuresi'].value_counts()
✓ 0.0s
```

```
TedaviSuresi
15 Seans    1670
10 Seans     175
20 Seans     113
2 Seans       45
17 Seans      36
4 Seans       35
16 Seans      27
18 Seans      20
21 Seans      20
5 Seans       17
30 Seans      12
19 Seans      10
11 Seans       9
3 Seans        7
8 Seans        6
37 Seans       5
```

Hedef verimiz TedaviSuresi'ni incelediğimizde 15 Seans'da bir yığılma olduğunu görüyoruz. Seans verisi kesikli ("discrete") bir değişken olması ve 37 gibi outlier değerler içermesi nedeniyle bu değişken olduğu gibi kullanılacaktır. Eğer bu veri bir "feature" olsaydı Kısa-Orta-Uzun gibi bir sınıflandırma yapılabilirdi.



Uygulama Yerleri verilerini incelediğimizde sol ya da sağ olmasının TedaviSuresi'ne etkisi olmayacağından sol sağ ve bölgesi sözcüklerini çıkarma kararı alınmıştır.



Uygulama Suresini incelediğimizde right skewed olduğunu ve sağ tarafta outlier'lara sahip olduğunu görüyoruz.

2. Data Preprocess

```python
def main():
    df = pd.read_excel(DATA_PATH)
    preprocessor = DataPreprocessor(df)
    preprocessor.fill_missing_values()
    preprocessor.fix_typos()
    preprocessor.feature_cleaning()
    preprocessor.encode_categorical()
    preprocessor.encode_numeric()
    preprocessor.info()
    preprocessor.save("cleaned_data.xlsx")
```

Preprocess işlemi 5 adımdan oluşmaktadır.

Eksik verilerin tamamlanması için KanGrubu, Cinsiyet ve KronikHastalıklar değişkenleri hasta bazında sabit kaldığı için eksik veriler aynı HastaNo'ya sahip diğer kayıtlardan forward/backward fill yöntemiyle doldurulmuştur.

```python
columns_to_fill = ["KanGrubu", "Cinsiyet", "KronikHastalik"]
for column in columns_to_fill:
    self._fill_missing_values_by_patient(column)
```

```python
# Fill categorical columns with "bilinmiyor"
self.df["Cinsiyet"].fillna("bilinmiyor", inplace=True)
self.df["KanGrubu"].fillna("bilinmiyor", inplace=True)
self.df["Alerji"].fillna("bilinmiyor", inplace=True)
self.df["KronikHastalik"].fillna("bilinmiyor", inplace=True)

# Group-based filling
self._fill_groupby(target_col="Bolum", group_col="TedaviAdi")
self._fill_groupby(target_col="UygulamaYerleri", group_col="TedaviAdi")

# Fill remaining missing values in "Tanilar" with overall most frequent value
simple_imputer = SimpleImputer(strategy="most_frequent")
self.df["Tanilar"] = simple_imputer.fit_transform(self.df[["Tanilar"]]).ravel()
```

Ardından eksik kalan Cinsiyet, KanGrubu, Alerji ve KronikHastalıklar bilinmiyor ile doldurulmuştur.

Bolum, UygulamaYerleri ve Tanilar TedaviAdi ile ilişkili olduğundan eksik veriler her TedaviAdi grubu içindeki en sık görülen değerlerle (mode) tamamlanmıştır.

```
def fix_typos(self):
    """
    Fix common typos in specific columns.
    """

    self.df["Alerji"] = self.df["Alerji"].str.replace(
        "volteren", "voltaren", case=False, regex=False
    )

    self.df["KronikHastalik"] = self.df["KronikHastalik"].str.replace(
        "hipotirodizm", "hiportiroidizm", case=False, regex=False
    )
```

Sonraki adımda EDA bölümünde bahsettiğim yazım yanlışları giderilmiştir.

```
def feature_cleaning(self):
    """
    Perform feature cleaning tasks such as dropping unnecessary columns,
    creating age groups, extracting body parts, and numeric extraction.
    """
    self._create_age_groups()
    self._extract_body_parts()
    self._numeric_extraction()
```

Bu aşamada yaş değişkeni "Çocuk", "Genç", "Genç Yetişkin", "Yetişkin", "Yaşlı" kategorilerine gruplandırılmıştır. UygulamaYerleri değişkeninden "sol", "sağ" ve "bölgesi" açıklamaları temizlenmiştir. Son olarak TedaviSuresi ve UygulamaSuresi değişkenlerinden sırasıyla "Seans" ve "Dakika" ifadeleri çıkarılarak sadece sayısal değerler bırakılmıştır."

```python
def encode_categorical(self):
    """
    Encode categorical features using one-hot encoding and multi-label binarization.
    """

    # YaşGrubu encoding
    self._encode_simple_categorical("YaşGrubu")

    # Cinsiyet encoding
    self.df["Erkek"] = np.where(self.df["Cinsiyet"] == "Erkek", 1, 0)
    self.df["Kadin"] = np.where(self.df["Cinsiyet"] == "Kadın", 1, 0)
    self.df.drop(columns=["Cinsiyet"], inplace=True)

    # KanGrubu encoding
    self._encode_simple_categorical("KanGrubu")

    # Uyruk encoding
    self._encode_simple_categorical("Uyruk")

    # KronikHastalik encoding
    self._encode_multivalue_categorical("KronikHastalik")

    # Bolum encoding
    self._encode_simple_categorical("Bolum")

    # Alerji encoding
    self._encode_multivalue_categorical("Alerji")

    # Tanilar encoding
    self._encode_multivalue_categorical("Tanilar", top_n=10)

    # TedaviAdi encoding
    self._encode_multivalue_categorical("TedaviAdi", top_n=10)

    # UygulamaYerleri encoding
    self._encode_multivalue_categorical("UygulamaYerleri")
```

YaşGrubu, Cinsiyet, KanGrubu, Uyruk ve Bolum değişkenleri az sayıda eşsiz değer içerdiğinden tamamına One-Hot encoding uygulanmıştır. Alerji ve UygulamaYerleri değişkenlerinde birden fazla değer tek hücrede bulunduğundan, bu değerler ile ayrıştırılarak her biri için ayrı One-Hot encoding yapılmıştır. Son olarak Tanılar ve TedaviAdi değişkenleri çok fazla eşsiz değer içerdiğinden, en sık görülen ilk 10 değer için One-Hot encoding uygulanmış, geri kalan değerler 'Diğer' kategorisi altında toplanmıştır.

```python
def encode_numeric(self):
    """
    Encode numeric features using Min-Max scaling.
    """
    scaler = MinMaxScaler()
    self.df["UygulamaSuresi"] = scaler.fit_transform(self.df[["UygulamaSuresi"]])
```

Son olarak UygulamaSuresi verilerinde MinMax scaler kullanılmıştır.