```
# pandas kütüphanesini yüklemek için kullanılır
pip install pandas
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Google Colab ile Google Drive'ı bağlamak için gerekli
kütüphaneyi içe aktarıyoruz
from google.colab import drive

# Google Colab ile Google Drive'ı bağlamak için gerekli
kütüphaneyi içe aktarıyoruz
drive.mount('/content/drive/')
```

```
# os kütüphanesini içe aktarıyoruz, bu kütüphane işletim sistemi
ile ilgili işlemleri yapmamızı sağlar
import os

# Çalışma dizinini değiştiriyoruz, bu dizin Google Drive'daki
belirli bir klasöre ayarlanıyor
os.chdir('/content/drive/My Drive/MakineSon')

# Şu anki çalışma dizinini yazdırmak için kullanılır, böylece
dizinin doğru şekilde değiştirildiğini doğrulayabiliriz
!pwd
```

```
# Veri setini oku
df = pd.read_csv('dataset.csv')  #verisetini okur ve "df"
değişkeninin içine atar
```

```
# Adres sütununu "Sehir", "Ilce" ve "Mahalle" olarak üçe böl ve
eksik değerleri doldur
adres_split = df['Adres'].str.split(' - ', expand=True)

# Her satırın üç parçaya bölünmesini sağla
df['Sehir'] = adres_split[0]
df['Ilce'] = adres_split[1]
df['Mahalle'] = adres_split[2]

# Eksik değerleri doldur (örneğin, boş string ile)
df['Ilce'] = df['Ilce'].fillna('')
df['Mahalle'] = df['Mahalle'].fillna('')
```

```
df.head()
```

```
#Adres Sütununu Siliyoruz
df.drop(columns=['Adres'], inplace=True)
```

```
df.head()
```

```
#Veri seti çok büyük olduğu için sadece İstanbul'u kullanmaya
karar verdik.
istanbul_df = (df[df['Sehir'] == 'İstanbul'])
```

```
istanbul_df
```

```
istanbul_df.drop(columns=['Sehir'], inplace=True)
```

```
istanbul_df
```

```
#Tapu durumu sütununu gereksiz bulduğumuz için veri setinden
kaldırıyoruz.
#Tahminleri olumsuz yönde etkileyebilir
istanbul_df = istanbul_df.drop(columns=['Tapu Durumu'])
```

```
def data_prep(dataframe):

    # Tipi bina olup oda sayısı 3+1, 2+1 gibi olanlar vardı. O
yüzden kaldırıldı
    dataframe.drop(dataframe[dataframe["Tipi"] == "Bina"].index,
axis=0, inplace=True)

    # Hepsi konut olduğu için bu sütunu kaldırıldı
    dataframe = dataframe.drop("Türü",axis=1)
    dataframe = dataframe.drop("Balkon Sayısı",axis=1)
    dataframe = dataframe.drop("WC Sayısı",axis=1)

    # Fiyat değişkeni object tipinden integer tipine çevrildi
    dataframe["Fiyat"] = dataframe["Fiyat"].str.replace(",","")
    dataframe["Fiyat"] = dataframe["Fiyat"].str.split("T").str[0]
    dataframe["Fiyat"] = dataframe["Fiyat"].astype(np.int64)

    # Oda sayısı evdeki toplam oda sayısı şeklinde düzeltildi
    def oda_sayisi(oda):
        if oda == "9+ Oda":
            return 9
        elif oda == "Stüdyo" or oda == "1 Oda":
            return 1
        else:
            sayi1, sayi2 = oda.split("+", 1)
            return int(float(sayi1)) + int(sayi2)

    dataframe["Oda Sayısı"] = dataframe["Oda
Sayısı"].apply(oda_sayisi)

    # Binanın yaşı 5-10 ve 11-15 olan değerler "10-May" ve "15-
Nov" şeklinde alındığı için düzeltildi
    dataframe["Binanın Yaşı"] = dataframe["Binanın
Yaşı"].str.replace("10-May","5-10")
    dataframe["Binanın Yaşı"] = dataframe["Binanın
Yaşı"].str.replace("15-Nov","11-15")


    # Sıkıntılı ilanlardı. Net m2 = 1 ?
    dataframe.drop(dataframe[dataframe["Bulunduğu
Kat"]==44].index, axis=0, inplace=True)

    # Boş olan değerler Null olarak atandı
    #dataframe =
dataframe.replace(["Belirtilmemiş","Bilinmiyor"], np.nan)

    # Brüt metrekare sütunu düzeltilerek sayısal değere
dönüştürüldü
    def brut(value):
```

```
        return int(value.replace(".", "").split(" ")[0])

    dataframe["Brüt Metrekare"] = dataframe["Brüt
Metrekare"].apply(brut)

    return dataframe
```

```
# istanbul_df veri çerçevesini data_prep fonksiyonuna
geçiriyoruz,
# bu fonksiyon verileri ön işleme tabi tutar ve dönüşümlerin
ardından geri döner
istanbul_df = data_prep(istanbul_df)

# Veri çerçevesinin ilk birkaç satırını görüntülemek için
kullanılır,
# bu sayede verilerin doğru şekilde yüklendiğini ve işlendiğini
doğrulayabiliriz
istanbul_df.head()
```

```
# Grafiğin boyutunu belirlemek için kullanılır (15 genişlik, 5
yükseklik)
plt.figure(figsize=(15,5))

# Seaborn kütüphanesi ile "Ilce" sütunundaki değerlerin sayısını
sayarak bir çubuk grafiği oluşturur
ax = sns.countplot(x="Ilce",data=df_copy)

# Çubuk grafikteki her çubuğun üstüne sayısal etiket ekler
ax.bar_label(ax.containers[0])

# X eksenindeki etiketleri 90 derece döndürür, böylece daha
okunaklı hale gelir
plt.xticks(rotation=90)

# Grafiği ekranda gösterir
plt.show()
```

```
# Grafiğin boyutunu belirlemek için kullanılır (15 genişlik, 5 yükseklik)
plt.figure(figsize=(15,5))

# Seaborn kütüphanesi ile "Tipi" sütunundaki değerlerin sayısını sayarak bir çubuk grafiği oluşturur
ax = sns.countplot(x='Tipi',data=df)

# Çubuk grafikteki her çubuğun üstüne sayısal etiket ekler
ax.bar_label(ax.containers[0])

# Grafiği ekranda gösterir
plt.show()
```

```
# Grafiğin boyutunu belirlemek için kullanılır (15 genişlik, 20 yükseklik)
plt.figure(figsize=(15,20))

# Seaborn kütüphanesi ile "Ilce" sütunundaki her bir ilçenin "Fiyat" ortalamasını yatay çubuk grafiği olarak oluşturur
ax = sns.barplot(x="Fiyat", y="Ilce", data=df_copy,orient='h', ci = 0)

# Çubuk grafikteki her çubuğun üstüne sayısal etiket ekler,
# fmt parametresi ile sayısal etiket formatını belirler (%.2f: ondalık olarak 2 basamak gösterir),
# fontsize parametresi ile yazı boyutunu ayarlar, padding parametresi ile etiketi çubuktan uzaklaştırır
ax.bar_label(ax.containers[0], fmt='%.2f', fontsize=10, padding=5)

# X eksenindeki etiketleri 90 derece döndürür, böylece daha okunaklı hale gelir
plt.xticks(rotation=90)

# Grafiği ekranda gösterir
plt.show()
```

```
# istanbul_df veri çerçevesinin sütun isimlerini yeniden
adlandırıyoruz. Yeni isimler sırasıyla aşağıdaki listede
belirtilmiştir:

df_ist = istanbul_df.columns = ["fiyat", "oda_sayısı",
"bulunduğu_kat", "ısıtma_tipi",
                        "krediye_uygunluk", "yapı_durumu",
"eşya_durumu", "site_içerisinde",
                        "tipi", "brüt_metrekare", "binanın_yaşı",
"binanın_kat_sayısı",
                        "kullanım_durumu", "yatırıma_uygunluk",
"banyo_sayısı",
                        "ilce", "mahalle"]
```

```
istanbul_df
```

```
# "yapı_durumu" sütunundaki her bir değerin kaç kez geçtiğini
sayar ve sonuçları gösterir
istanbul_df.yapı_durumu.value_counts()
```

```
# binanın_yaşı sütununu düzenleyelim, 0 (Yeni) olanları 0 olarak
düzelteceğiz
istanbul_df['binanın_yaşı'] =
istanbul_df['binanın_yaşı'].replace('0 (Yeni)', '0')
```

```
# Koşullu dönüşüm fonksiyonunu tanımlayalım
def convert_yapı_durumu(row):
    if row['binanın_yaşı'] == '0':
        return 'Sıfır'
    elif row['yapı_durumu'] == 'Yapım Aşamasında':
        return 'Yapım Aşamasında'
    else:
        return 'İkinci El'  # Varsayılan olarak İkinci El
döndürelim

# apply fonksiyonunu kullanarak dönüşümü uygulayalım
istanbul_df['yapı_durumu'] =
istanbul_df.apply(convert_yapı_durumu, axis=1)

# Sonuçları kontrol edelim
print(istanbul_df['yapı_durumu'].value_counts())
```

```
istanbul_df.loc[istanbul_df.banyo_sayısı == "Yok",
"banyo_sayısı"] = 0
istanbul_df.loc[istanbul_df.banyo_sayısı == "5", "banyo_sayısı"]
= 5
istanbul_df.loc[istanbul_df.banyo_sayısı == "2", "banyo_sayısı"]
= 2
istanbul_df.loc[istanbul_df.banyo_sayısı == "1", "banyo_sayısı"]
= 1
istanbul_df.loc[istanbul_df.banyo_sayısı == "3", "banyo_sayısı"]
= 3
istanbul_df.loc[istanbul_df.banyo_sayısı == "4", "banyo_sayısı"]
= 4
```

```
# "eşya_durumu" sütununda "Belirtilmemiş" olan değerleri "Boş"
olarak değiştirir
istanbul_df.loc[istanbul_df.eşya_durumu == "Belirtilmemiş",
"eşya_durumu"] = "Boş"

# "eşya_durumu" sütunundaki her bir değerin kaç kez geçtiğini
sayar ve sonuçları gösterir
istanbul_df.eşya_durumu.value_counts()
```

```
# "yatırıma_uygunluk" sütununda "Belirtilmemiş" olan değerleri
"Bilinmiyor" olarak değiştirir
istanbul_df.loc[istanbul_df.yatırıma_uygunluk == "Belirtilmemiş",
"yatırıma_uygunluk"] = "Bilinmiyor"

# "yatırıma_uygunluk" sütunundaki her bir değerin kaç kez
geçtiğini sayar ve sonuçları gösterir

istanbul_df.yatırıma_uygunluk.value_counts()
```

```
# "tipi" sütunundaki her bir değerin kaç kez geçtiğini sayar ve
sonuçları gösterir
istanbul_df.tipi.value_counts()
```

```
# "bulunduğu_kat" sütunundaki her bir değerin sonundaki ".Kat"
ifadesini kaldırır
istanbul_df['bulunduğu_kat'] =
istanbul_df['bulunduğu_kat'].str.replace('.Kat', '')

# "bulunduğu_kat" sütunundaki her bir değerin kaç kez geçtiğini
sayar ve sonuçları gösterir
istanbul_df.bulunduğu_kat.value_counts()
```

```
istanbul_df.loc[istanbul_df.bulunduğu_kat == "Yüksek Giriş",
"bulunduğu_kat"] = "Giriş"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "Düz Giriş",
"bulunduğu_kat"] = "Giriş"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "Yüksek Bodrum",
"bulunduğu_kat"] = "Bodrum"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "Tam Bodrum",
"bulunduğu_kat"] = "Bodrum"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "Yarı Bodrum",
"bulunduğu_kat"] = "Bodrum"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "10",
"bulunduğu_kat"] = "10-20"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "11",
"bulunduğu_kat"] = "10-20"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "12",
"bulunduğu_kat"] = "10-20"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "13",
"bulunduğu_kat"] = "10-20"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "14",
"bulunduğu_kat"] = "10-20"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "15",
"bulunduğu_kat"] = "10-20"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "16",
"bulunduğu_kat"] = "10-20"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "17",
"bulunduğu_kat"] = "10-20"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "18",
"bulunduğu_kat"] = "10-20"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "19",
"bulunduğu_kat"] = "10-20"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "20",
"bulunduğu_kat"] = "10-20"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "21",
"bulunduğu_kat"] = "21-29"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "22",
"bulunduğu_kat"] = "21-29"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "23",
"bulunduğu_kat"] = "21-29"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "24",
"bulunduğu_kat"] = "21-29"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "25",
"bulunduğu_kat"] = "21-29"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "26",
"bulunduğu_kat"] = "21-29"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "27",
"bulunduğu_kat"] = "21-29"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "28",
"bulunduğu_kat"] = "21-29"
```

```
istanbul_df.loc[istanbul_df.bulunduğu_kat == "29",
"bulunduğu_kat"] = "21-29"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "30",
"bulunduğu_kat"] = "30-39"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "31",
"bulunduğu_kat"] = "30-39"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "32",
"bulunduğu_kat"] = "30-39"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "33",
"bulunduğu_kat"] = "30-39"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "34",
"bulunduğu_kat"] = "30-39"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "35",
"bulunduğu_kat"] = "30-39"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "36",
"bulunduğu_kat"] = "30-39"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "37",
"bulunduğu_kat"] = "30-39"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "38",
"bulunduğu_kat"] = "30-39"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "39",
"bulunduğu_kat"] = "30-39"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "40+",
"bulunduğu_kat"] = "40"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "Kot 1 (-1)" ,
"bulunduğu_kat"] = "-1"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "Kot 2 (-2)" ,
"bulunduğu_kat"] = "-2"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "Kot 3 (-3)" ,
"bulunduğu_kat"] = "-3"
istanbul_df.loc[istanbul_df.bulunduğu_kat == "Kot 4 (-4)" ,
"bulunduğu_kat"] = "-3"
```

```
# "bulunduğu_kat" sütununda "Belirtilmemiş" olan satırları
filtreleyerek yeni bir veri çerçevesi oluşturur
istanbul_df2 = istanbul_df[istanbul_df['bulunduğu_kat'] !=
'Belirtilmemiş']
```

```
# "istanbul_df2" veri çerçevesindeki "bulunduğu_kat" sütunundaki
her bir değerin kaç kez geçtiğini sayar ve sonuçları gösterir
istanbul_df2.bulunduğu_kat.value_counts()
```

```
# "istanbul_df2" veri çerçevesindeki "eşya_durumu" sütunundaki
her bir değerin kaç kez geçtiğini sayar ve sonuçları gösterir
istanbul_df2.eşya_durumu.value_counts()
```

```
istanbul_df.head()
```

```
# Modeli eğitme kısmında verilerin dengesini bozduğunu gördüğümüz
için kaldırdık.
istanbul_df2 = istanbul_df.drop(columns=['banyo_sayısı'])
```

```
# "istanbul_df2" veri çerçevesini 'istanbul2.csv' adında bir CSV
dosyasına dönüştürür ve kaydeder
# index=False parametresi, satır numaralarını kaydetmemeyi sağlar
istanbul_df2.to_csv('istanbul2.csv', index=False)
```

```
istanbul_df2
```