İSTANBUL GELİŞİM ÜNİVERSİTESİ

İSTANBUL GELİŞİM MESLEK YÜKSEKOKULU

BİLGİSAYAR TEKNOLOJİLERİ BÖLÜMÜ

BİLGİSAYAR PROGRAMCILIĞI PROGRAMI

EMLAK FİYAT TAHMİN

FİNAL PROJE ÖDEVİ

Hazırlayan

220111583 – Ersin UÇAR

220111623 – Mevlütcan MERCAN

220111559 – Efe KÜRKÇÜ

220111621 – Mert Ali GENÇTÜRK

210111251 – Ayşenur Ceren ÖZTÜRK

Ödev Danışmanı

Öğr. Gör.

Tuğba Saray

Çetinkaya

İSTANBUL – 2023

ÖDEV TANITIM FORMU

YAZAR ADI SOYADI	: Ayşenur Ceren ÖZTÜRK
ÖDEVİN DİLİ	: Türkçe
ÖDEVİN ADI	: Emlak Fiyat Tahmini
BÖLÜM	: Bilgisayar Teknolojileri
PROGRAM	: Bilgisayar Programcılığı
ÖDEVİN TÜRÜ	: Final
ÖDEVİN TES. TARİHİ	:29.05.2024
SAYFA SAYISI	: 22
ÖDEV DANIŞMANI	: Öğr. Gör. Tuğba Saray Çetinkaya

BEYAN

Bu ödevin/projenin hazırlanmasında bilimsel ahlak kurallarına uyulduğu, başkalarının ederlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğu, kullanılan verilerde herhangi tahrifat yapılmadığını, ödevin/projenin herhangi bir kısmının bu üniversite veya başka bir üniversitedeki başka bir ödev/proje olarak sunulmadığını beyan eder, aksi durumda karşılaşacağım cezai ve/veya hukuki durumu kabul eder; ayrıca üniversitenin ilgili yasa, yönerge ve metinlerini okuduğumu beyan ederim.

29.05.2024

Mevlütcan MERCAN

Efe KÜRKÇÜ

Ersin UÇAR

Mert Ali GENÇTÜRK

Ayşenur Ceren ÖZTÜRK

KABUL VE ONAY SAYFASI

210111251 numaralı **Ayşenur Ceren ÖZTÜRK'ün** "Emlak Fiyat Tahmin" adlı çalışması, benim tarafımdan Final ödevi olarak kabul edilmiştir.

Öğretim Görevlisi

Tuğba Saray ÇETİNKAYA

ÖZET

Bu projede, çeşitli emlak özelliklerine dayalı olarak emlak fiyatlarını tahmin etmek amaçlanmıştır. Bu amaç doğrultusunda, farklı makine öğrenmesi algoritmaları kullanılmış ve bu modellerin performansları karşılaştırılmıştır. Veri seti, emlak ilanlarından derlenmiş olup oda sayısı, bulunduğu kat, ısıtma tipi, krediye uygunluk vb. sütunlardan oluşmaktadır. Projede tüm regresyon modelleri denenerek performansı en iyi olan model seçilmiştir bu sayede verimi en üst düzeye taşımış olduk. Projenin devamında ise performans iyileştirmeleri yaparak fiyat tahminin başarı oranını arttırdık.

İçindekiler

ÖNSÖZ

Bu proje, üniversite final ödevim için geliştirilmiştir. Efe KÜRKÇÜ, Mevlütcan MERCAN, Mert Ali GENÇTÜRK, Ayşenur Ceren ÖZTÜRK ile birlikte çalışarak bu projeyi hayata geçirdik.

Ödevin hazırlanması sürecinde karşılaştığımız zorlukları aşmak için birlikte çalışmak, fikir alışverişinde bulunmak ve sorunları birlikte çözmek büyük önem taşımıştır. Projenin farklı aşamalarında değerli katkılarda bulunan arkadaşlarıma teşekkür ederim.

Aynı zamanda projenin hazırlanmasında bize destek olan öğretmenlerimize ve Üniversitemize teşekkür etmek istiyorum. Onların rehberliği ve yönlendirmeleri sayesinde projemizi tamamlamak ve sunmak için gereken motivasyonu bulduk.

AYŞENUR CEREN ÖZTÜRK

1. Konu Seçimi:

Makine öğrenmesi, birçok alanda devrim niteliğinde yenilikler getirmiştir ve bu alanlardan biri de emlak fiyat tahminidir. Emlak fiyat tahmini, gayrimenkul sektöründe oldukça önemli bir uygulama alanıdır. Bu konu, doğru fiyatlandırmanın kritik olduğu alıcılar, satıcılar ve yatırımcılar için büyük bir değer taşır. Emlak fiyat tahmininin doğru yapılabilmesi, piyasa analizlerinde, yatırım stratejilerinde ve bireysel gayrimenkul alım-satım kararlarında hayati rol oynar.

2. Veri Seti Seçimi:

Emlak fiyat tahmini çalışmamızda, uygun veri setini bulmak ve karar vermek için çeşitli veri platformlarını inceledik. İlk olarak "Kaggle" platformunu araştırdık. Kaggle, dünya genelinde veri bilimi ve makine öğrenmesi yarışmalarının yapıldığı, geniş bir veri seti havuzuna sahip popüler bir platformdur. Bu platformda emlak fiyat tahmini için birçok veri seti bulunmaktaydı. Bu platformda özellikle "Boston Housing" veri seti öne çıkıyordu ancak biz Türkiye'ye ait bir veri seti bulmayı amaçlıyorduk. Yapılan kapsamlı araştırmalar sonucunda, emlak jet firmasına ait bir veri setini kullanmaya karar verdik. Bu veri seti, Türkiye'deki emlak piyasasına dair kapsamlı ve güncel veriler sunduğundan, çalışmamız için en uygun seçenek olarak belirlendi.

3. Veri Setinin Yapısı:

Veri seti, emlak fiyatlarını tahmin etmek için çeşitli özellikleri içermektedir. Bu özellikler arasında "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", "il", "ilçe" ve "mahalle" bulunmaktadır. Her bir özellik, bir emlak ilanının belirli bir yönünü ifade eder ve çoğunlukla sayısal verilerden oluşur. Hedef değişken "fiyat" olup, diğer özellikler bu fiyatı tahmin etmek için bağımsız değişkenler olarak kullanılmıştır. Bu yapı, emlak piyasasındaki fiyatları daha iyi anlamak için kapsamlı

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	ilce	mahalle
39970000	9	Belirtilmemiş	Kiimalı	Krediye Uygun	İkinci El	Boş	Hayır	Müstakil Ev	550	21 Ve Üzeri	3	Boş	Bilinmiyor	Adalar	Nizam Mahallesi
8500000	5	Belirtilmemiş	Doğalgaz Sobalı	Krediye Uygun Değil	Sıfır	Boş	Hayır	Köşk	300	0	3	Boş	Bilinmiyor	Adalar	Heybeliada Mahallesi
8750000	4	Belirtilmemiş	Kombi Doğalgaz	Krediye Uygun	İkinci El	Boş	Hayır	Müstakil Ev	120	21 Ve Üzeri	2	Boş	Yatırıma Uygun	Adalar	Heybeliada Mahallesi
2950000	3	Giriş	Kombi Doğalgaz	Krediye Uygun	İkinci El	Boş	Hayır	Daire	85	21 Ve Üzeri	4	Boş	Bilinmiyor	Adalar	Heybeliada Mahallesi
9450000	6	2	Kombi Doğalgaz	Krediye Uygun	İkinci El	Eşyalı	Hayır	Daire	170	21 Ve Üzeri	3	Mülk Sahibi Oturuyor	Bilinmiyor	Adalar	Maden Mahallesi
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Şekil 1

bir veri analizi yapılmasını sağlar. (**Şekil 1**)

4. Veri Setinin Hazırlanması:

Yapmak istediğimiz model daha genel bir tahmin modeli olmasından dolayı değerlendirmeye almayacağımız birkaç sütunu veri setimizden çıkartıyoruz. Veri setinin hazırlanması aşamaları şu şekildedir;

4.1 Adres Sütunu ve İlçe Seçimi

```
# 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('')
```

Şekil 2

Veri Setimizde bulunan Adres sütununu modelimizde kullanabilmek için "Şehir", "İlçe" ve "Mahalle" olarak üçe böldük. Bu işlem için **split** komutunu kullandık. Bu sütunlardaki boş değerleri **fillna** komutu ile boş string olarak dolduruyoruz. Daha sonra birleşik olarak kalan Adres sütununu siliyoruz.

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

Şekil 3

Veri setinde aşırı veri olması modeli öğretmemizi zorlaştıracağı için veri setini küçültmeye karar verdik. Bu işlem ile birlikte artık sadece "İstanbul" ilçesini kullanıyoruz.

4.2 Veri Uyumsuzluğunun Giderilmesi

```python
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)


    def brut(value):
        return int(value.replace(".", "").split(" ")[0])

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

    return dataframe
```

Şekil 4

Veri setimizdeki bazı anlamsız verileri temizlemek için "**data_prep**" adında bir dataframe oluşturduk, burada veri setimizdeki gereksiz şeyleri temizliyoruz veya değiştiriyoruz. Burada "Türü", "Balkon Sayısı", "WC Sayısı" gibi gereksiz gördüğümüz sütunları siliyoruz. Fiyat sütunundaki virgülleri, TL ibaresini kaldırırken değişkenini de object'den int64'e çeviriyoruz. Oda sayısındaki gereksiz işaretleri kaldırıyoruz. Bina yaşı sütunundaki yanlış girilmiş verileri düzeltiyoruz. Brüt Metrekare bölümünde "**M2**" ibaresini boşluktan sonraki indeksi silerek düzenliyoruz ve int olarak döndürüyoruz.

```
# 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())
```

```
yapı_durumu
İkinci El          22978
Sıfır               8148
Yapım Aşamasında      10
Name: count, dtype: int64
```

```
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
```

Şekil 5

```
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"
```

Şekil 6

Veri setimize bina yaşı sıfır olan satırlar “0 (Yeni)” olarak girilmişti, bunu düzeltmek için sadece “0” kalacak şekilde düzenledik. Bina yaşı verilerini düzenlemek için bir fonksiyon tanımladık. Banyo sayılarını da düzenledik, projenin ilerleyen kısmında bu sütunu kaldıracağız. Binanın bulunduğu kat değerlerini de daha anlaşılır olması ve int’e çevirdiğimizde kontrol edilebilir olması açısından değiştirdik.

```
# 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 kayd
# index=False parametresi, satır numaralarını kaydetmemeyi sağlar
istanbul_df2.to_csv('istanbul2.csv', index=False)
```

Şekil 7

```
df_3.drop("mahalle", axis=1, inplace=True)
```

Şekil 8

Modeli eğitme kısmında fiyat değişkeni üzerinde olamaması gereken kadar yüksek etkisi olduğunu fark ettiğimiz için burada banyo_sayısı sütununu siliyoruz, ardından yaptığımız değişiklikleri kopya olarak başka bir tablo adıyla kaydediyoruz. Daha sonra Mahalle sütununu kaldırıyoruz.

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
```

Şekil 9

Burada bütün sütunları LabelEncoder ile kategorik verileri sayısal değere

```
df_3["ilce"] = le.fit_transform(df_2.ilce)
le.classes_
df_3.ilce.unique()
le = preprocessing.LabelEncoder()


df_3["yatırıma_uygunluk"] = le.fit_transform(df_2.yatırıma_uygunluk)
le.classes_
df_3.yatırıma_uygunluk.unique()
```

Şekil 10

```
df_3["kullanım_durumu"] = le.fit_transform(df_2.kullanım_durumu)
le.classes_
df_3.kullanım_durumu.unique()
le = preprocessing.LabelEncoder()
```

dönüştürüyoruz.

```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.11.4)
```

```
import numpy as np
import pandas as pd
import xgboost as xgb
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
```

Şekil 11

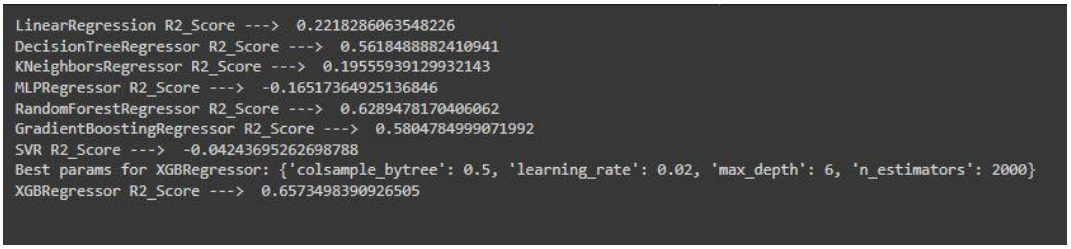
5. Veri Setimiz İçin En Uygun Modeli Bulma

Veri setimize en uygun modeli bulmak için birçok model ile test yapıyoruz, bu modelleri kullanabilmek için gerekli kütüphaneleri import ediyoruz.

```python
X = df.drop(["fiyat"], axis=1)  # 'fiyat' sütununu df DataFrame'inden çıkar ve geriye kalan sütunları X adlı değişkene ata
y = df["fiyat"]  # 'fiyat' sütununu df DataFrame'inden al ve y adlı değişkene ata


def compML(df, target, alg, params=None):
    # Veriyi hedef değişken ve özellikler olmak üzere ayır
    y = df[target].values
    X = df.drop([target], axis=1)

    # Veriyi MinMaxScaler ile ölçeklendir
    scaler = MinMaxScaler()
    X = scaler.fit_transform(X)

    # Eğitim ve test setlerine ayır
    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=144, shuffle=True)

    if params:
        # Eğer hiperparametreler belirtilmişse, GridSearchCV kullanarak en iyi parametreleri bul
        grid_search = GridSearchCV(alg(), param_grid=params, cv=5, scoring='r2')
        grid_search.fit(x_train, y_train)
        model = grid_search.best_estimator_  # En iyi modeli seç
        print(f"Best params for {alg.__name__}: {grid_search.best_params_}")  # En iyi parametreleri yazdır
    else:
        # Hiperparametreler belirtilmemişse, algoritmayı varsayılan parametrelerle eğit
        model = alg().fit(x_train, y_train)

    # Modeli kullanarak tahmin yap
    y_pred = model.predict(x_test)
    r2 = r2_score(y_test, y_pred)  # R2 skorunu hesapla
    print(alg.__name__, "R2_Score ---> ", r2)  # Modelin adı ve R2 skorunu yazdır

# Modellerin listesini güncelle ve örnek hiperparametreler tanımla
models = [LinearRegression, DecisionTreeRegressor, KNeighborsRegressor, MLPRegressor, RandomForestRegressor, GradientBoostingRegressor, SVR, XGBRegressor]
xgb_params = {
    "colsample_bytree":[0.4,0.5,0.6],
    "learning_rate":[0.01,0.02,0.09],
    "max_depth":[2,3,4,5,6],
    "n_estimators":[100,200,500,2000]
}

# Her model için compML fonksiyonunu çağırarak performansı karşılaştır
for i in models:
    if i == XGBRegressor:
        compML(df, "fiyat", i, params=xgb_params)  # XGBRegressor için özel hiperparametrelerle çağır
    else:
        compML(df, "fiyat", i)  # Diğer modeller için varsayılan hiperparametrelerle çağır
```

Şekil 12

İlk önce fiyat sütunu ile diğer sütunları birbirinden ayırmak için fiyat sütununu tablodan atıyoruz, ardından fiyat'ı y değişkenine atıyoruz. Daha sonra oluşturduğumuz fonksiyon ile eğitim ve test setlerini düzenliyoruz, burada **test size** değerini **0.25** olarak oluşturduk. Daha sonra her modeli karşılaştırıyoruz ve sonuçları ekrana yazdırıyoruz.



Şekil 13

6. En İyi Değeri Bulma:

Regresyon modelimize karar verdikten sonra en iyi başarı skorunu elde etmek için regresyon modeline vermemiz gereken en iyi değerleri denememiz gerekir. Fiyatı her zamanki gibi hedef değişken olarak belirliyoruz. Veri seti, eğitim ve test setlerine bölünüyor; bu adımda 75% eğitim, 25%'i test seti olarak ayırıyoruz. Daha sonra, XGBoost için denenecek parametre aralıklarını tanımlıyoruz. Bu parametreler modelin başarısını etkileyen faktördür.

RandomizedSearchCV kullanılarak en iyi parametreler bulunuyor. Çapraz doğrulama yöntemiyle birlikte belirli bir sayıda rastgele parametre kombinasyonu oluşturuluyor. En iyi parametreler belirlendikten sonra, bu parametrelerle eğitilen modelin performansı ölçülüyor. Bu ölçümde R2 skorunu kullanıyoruz ve sonucu ekrana yazdırıyoruz.

Son olarak, en iyi parametrelerle eğitilen model test seti üzerinde kullanılarak tahminler yapılıyor. Tahminlerin gerçek değerlerle olan R2 skoru tekrar hesaplanıyor ve sonuç ekrana yazdırılıyor. Bu adımlar, XGboost regresyon modelinin en iyi parametrelerini belirleme ve bu parametrelerle modelin eğitilmesi sürecini özetliyor. Bu şekilde modelin en iyi tahminleri yapmasını sağlayabiliriz.

Yazdığımız kodlar ve aldığımız başarı skorları şu şekildedir: (**Şekil 5**.bkz)

```python
from sklearn.model_selection import RandomizedSearchCV  # RandomizedSearchCV'yi içe aktar, hiperparametre araması için kullanılır
from sklearn.preprocessing import MinMaxScaler  # MinMaxScaler'ı içe aktar, verileri ölçeklendirmek için kullanılır
from sklearn.model_selection import train_test_split  # train_test_split'i içe aktar, veriyi eğitim ve test setlerine bölmek için kullanılır
import xgboost as xgb  # xgboost kütüphanesini içe aktar
from sklearn.metrics import r2_score  # r2_score metriğini içe aktar, modelin performansını değerlendirmek için kullanılır

target = 'fiyat'  # Hedef değişkeni tanımla
y = df[target].values  # Hedef değişkeni y'ye ata
X = df.drop([target], axis=1)  # Hedef değişken hariç tüm değişkenleri X'e ata

# Veriyi ölçeklendirme
scaler = MinMaxScaler()  # MinMaxScaler nesnesini oluştur
X = scaler.fit_transform(X)  # X verilerini MinMaxScaler ile ölçeklendir

# Eğitim ve test setlerine bölme
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=144, shuffle=True)  # Veriyi %75 eğitim ve %25 test setlerine böl

# XGBoost için hiperparametreler
params = {
    'colsample_bytree': [0.4, 0.5, 0.6, 0.7],  # Her ağaç için kullanılacak sütun örnekleme oranları
    'learning_rate': [0.01, 0.03, 0.05, 0.07, 0.09],  # Öğrenme oranları
    'max_depth': [3, 4, 5, 6, 7],  # Ağaçların maksimum derinlikleri
    'n_estimators': [500, 1000, 1500, 2000, 2500],  # Ağaç sayıları
    'subsample': [0.6, 0.7, 0.8, 0.9, 1.0],  # Örnekleme oranları
    'gamma': [0, 0.1, 0.2, 0.3]  # Ağaç yapılandırmasında düğüm bölünmesinin minimum kayıp azaltımı
}
xgb_model = xgb.XGBRegressor(early_stopping_rounds=10)  # XGBRegressor modelini tanımla ve erken durdurma turunu ayarla

# RandomizedSearchCV ile en iyi parametreleri bulma
random_search = RandomizedSearchCV(
    xgb_model, param_distributions=params, n_iter=50, cv=5, scoring='r2', verbose=1, n_jobs=-1, random_state=42
)  # RandomizedSearchCV'yi tanımla, n_iter=50 ile 50 farklı parametre kombinasyonunu dene
random_search.fit(x_train, y_train, eval_set=[(x_test, y_test)], verbose=False)  # Modeli eğitim seti ile eğit ve test seti ile doğrula

# En iyi parametreleri ve en iyi skoru yazdırma
print(f"Best params for XGBRegressor: {random_search.best_params_}")  # En iyi parametreleri yazdır
print(f"Best R2 score: {random_search.best_score_}")  # En iyi R2 skorunu yazdır

# En iyi parametrelerle eğitilen modeli kullanarak tahmin yapma
best_xgb_model = random_search.best_estimator_  # En iyi modeli al
y_pred = best_xgb_model.predict(x_test)  # Test seti üzerinde tahmin yap
print(f"XGBRegressor R2_Score ---> {r2_score(y_test, y_pred)}")  # Tahminlerin R2 skorunu yazdır
```

Şekil 5

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
Best params for XGBRegressor: {'subsample': 0.8, 'n_estimators': 2500, 'max_depth': 7, 'learning_rate': 0.03, 'gamma': 0, 'colsample_bytree': 0.5}
Best R2 score: 0.6257340237958682
XGBRegressor R2_Score ---> 0.6511093836983544
```

Şekil 6

Aldığımız çıktı ise şu şekildedir: (**Şekil 6**.bkz)

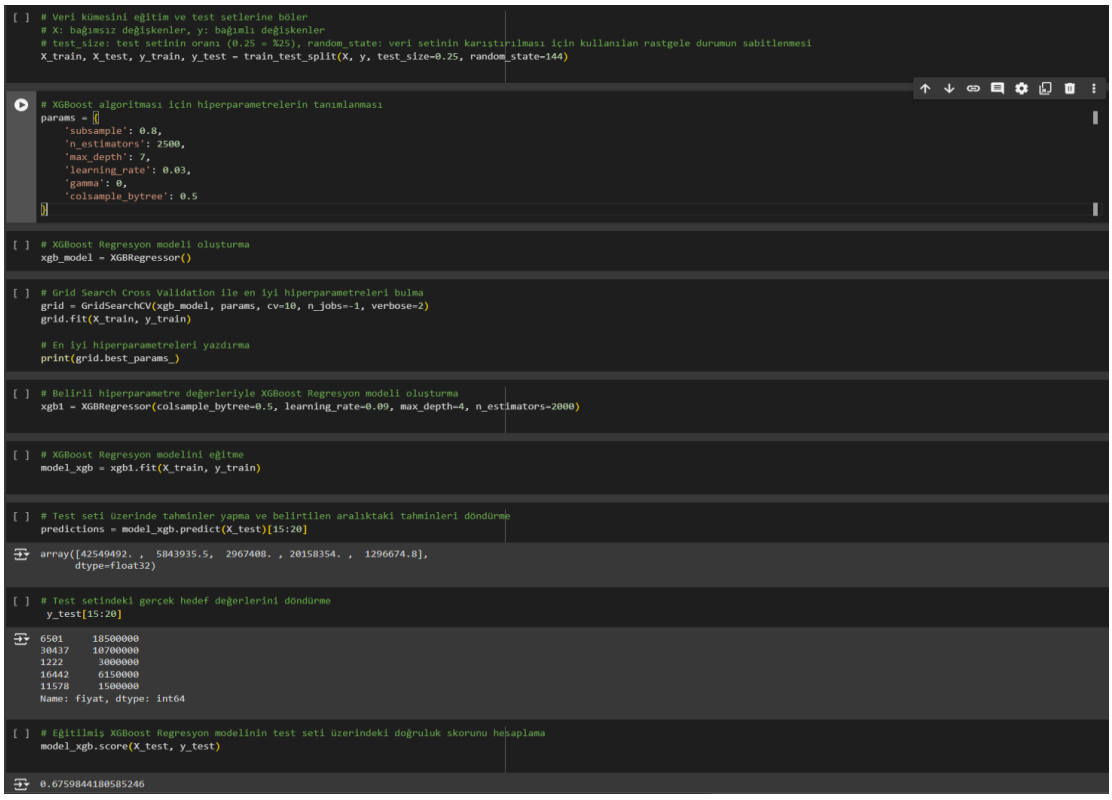
7. Belirlenen Parametrelere Göre XGBoost Modelinin Oluşturulması:

Bu aşamada, veri setimizi makine öğrenimi modellemesi için hazırlıyoruz. İlk adım olarak, veri setimizi yüklüyoruz ve bağımlı bağımsız değişkenleri belirliyoruz.Daha sonra, veri setimizi eğitim ve test setlerine ayırarak modelimizi doğru bir şekilde eğitmek ve değerlendirmek için gerekli adımları atıyoruz.XGBoost algoritması için en uygun parametreleri belirlemek için Grid Search Cross Validation tekniğini kullanıyoruz ve ardından belirlenen en iyi parametrelerle XGBoost

regresyon modelimizi oluşturuyoruz. Modelimizin performansını ölçmek için çeşitli metrikler kullanıyoruz ve tahminlerimizin gerçek değerlerle nasıl uyum sağladını görselleştirerek için hazırlamasını ve oluşturduğumuz modelin performansının değerlendirilmesini içeriyor.

7.1. Modelin Oluşturulması ve Eğitilmesi:

(**Şekil 8**)**ve**(**Şekil 7**) **de** modelimizi oluşturduk ve eğittik.Veri setimizdeki sütunların fiyat üzerindeki etkisini, eğitim ve test üzerindeki doğruluk skoru değerlerini burada görebiliriz.



Şekil 8

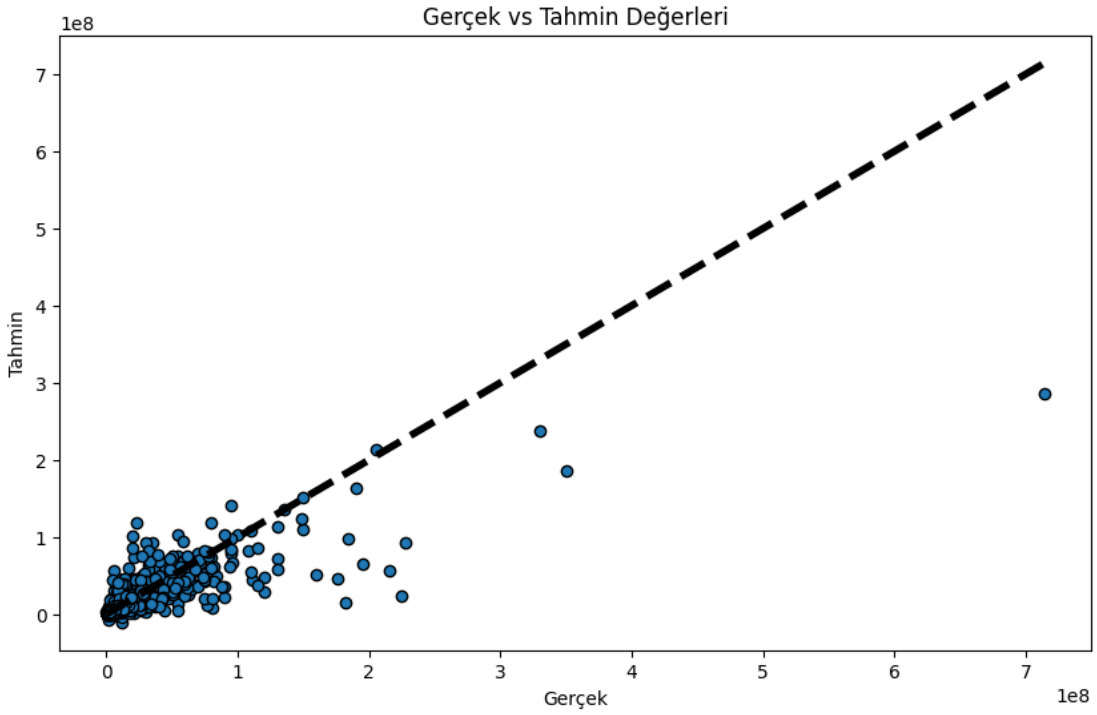
Şekil 7

7.2 Görselleştirme Aşamasında Ortaya Çıkan Grafikler:

7.2.1Gerçek VS Tahmin Değerleri:

Bu grafikte, noktaların çizgiye ne kadar yakın olduğu, modelin ne kadar doğru tahminler yaptığını gösterir. Eğer noktalar çizgiye yakınsa, modelin tahminleri gerçek değerlere daha yakındır ve modelin daha iyi performans gösterdiği söylenebilir.

Grafikte görüldüğü üzere noktaların çoğunluğunun çizgi üzerinde ve yakınlarında olması modelimizin gerçek değerlere yakın fiyat tahmini yaptığını gösterir.

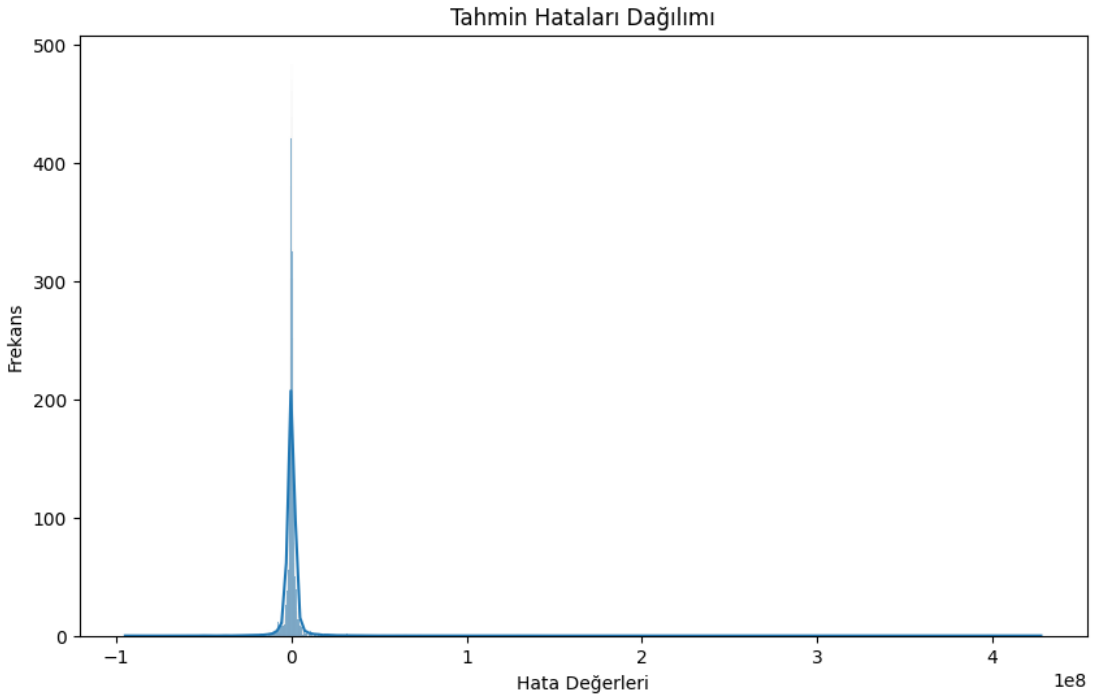


Şekil 9

7.2.2 Tahmin Hataları Dağılımı:

Bu grafik, modelin tahminlerinin gerçek değerlerden ne kadar sapma gösterdiğini gösterir. Özellikle, tahmin hatlarının dağılımını ve bu hataların sıklığını gösterir. Eğer bu dağılım normal bir dağılıma yakınsa, bu modelin genellikle iyi tahminler yaptığı ve hataların rastgele olduğu anlamına gelir.Ancak eğer dağılım düzensiz ise, bu modelin bazı durumlarda daha büyük hatalar yaptığı veya belirli bir eğilim veya desen içerdiği anlamına gelebilir. Tahmin hatalarının bu dağılımı, modelin performansını değerlendirmek ve iyileştirmek için önemli bir gösterge olabilir.

Grafikte görüldüğü üzere (**Şekil 10**), hata değeri 0 üzerindeki 400'e kadar çıkıyor, bu genellikle modelin belirli bir değeri doğru tahmin ettiği anlamına gelir. Yani tahmin edilen değerler gerçek değerlere oldukça yakın olmuştur ve bu durumda hataların büyük çoğunlu sıfıra yakın olacaktır.Bu modelin genel olarak başarılı olduğunu ve doğru tahminler yaptığını gösterir.



Şekil 10

KAYNAKÇA

- https://books.google.com.tr/books?hl=tr&lr=&id=C-yglCEcK0sC&oi=fnd&pg=PT12&dq=machine+learning+python&ots=2s71D3mgkr&sig=20EgxBH9DKGveQodhSe6vTpyEeQ&redir_esc=y#v=onepage&q=machine%20learning%20python&f=false
- https://books.google.com.tr/books?hl=tr&lr=&id=sKXIDwAAQBAJ&oi=fnd&pg=PP1&dq=machine+learning+python&ots=VaDolOVEGr&sig=UjkjhlL_sfDed6deoz5SJNLd1-Y&redir_esc=y#v=onepage&q=machine%20learning%20python&f=false
- https://github.com/senanurbalcioglu/ev_fiyat_tahmini
- https://youtu.be/Eo2en0bWN4c
- https://www.cell.com/heliyon/pdf/S2405-8440(20)32461-0.pdf