**T.C. SAKARYA ÜNİVERSİTESİ**

**BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ**

**BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ BÖLÜMÜ**

**VERİ MADENCİLİĞİ DERSİ PROJE ÖDEVİ**

**Hazırlayan**

**6.grup**

**Dersin Öğretim Üyesi**

**Esin Ayşe ZAİMOĞLU**

[**SIRALI VE KİŞİ BAZLI, VERİ TEMİZLEMEDE YAPILANLAR 4**](#_33w2asclog7n)

[AYŞE 4](#_axynmhou3bru)

[BAHADIR 5](#_jyodq4yf0cme)

[DENİZ 5](#_mzh9xjvoxllc)

[ESRA & GÜLSEMİN 6](#_2kkhjhnz7v6y)

[**ALGORİTMALAR 10**](#_8lhz9bdprmdu)

[1.LİNEER REGRESYON (Gülsemin & Esra) 10](#_bkf5q7k38zn4)

[ÖNCESİ 14](#_9ccbiya74kq6)

[SONRASI 15](#_jiit0sszwvzy)

[**2. K-MEANS (Gülsemin&Esra) 16**](#_4l5dsg1s4x0r)

[**3. Apriori Algorithm (Can) 24**](#_hde6r9s2387k)

[3.1 Drive a Bağlanma 25](#_req7ug6sxaxy)

[3.2 Kütüphane İmport 25](#_n10m5i64ohe2)

[3.3 Veri Okuma 25](#_gx2inp2gv8xk)

[3.4 Kolon Seçimi ve Format Dönüşümü 26](#_cnu02zkduah3)

[3.5 Detaysız Veri Seti Okuma 26](#_cp1ki9r2hlw)

[3.6 Pandas Ayarları ve Veri Görüntüleme (Sık) 27](#_32j7syjj602v)

[3.7 Birliktelik Kurallarının Görüntülenmesi ve İndeksleme 27](#_dt8ieushq7qe)

[3.8 Sık Öğe Kümelerinin Sıralanması 27](#_8erd3613xmup)

[3.9 Spesifik Satır İnceleme ve Örnek Analiz 28](#_2nzw7lxbxt42)

[3.10 Özel Filtreleme Fonksiyonu ve Kullanımı 29](#_7bpxq2t1mq11)

[**4 Logistic Regression (Can) 30**](#_cil43bpispha)

[4.1 Drive a Bağlanma 30](#_d3bltg66p02e)

[4.2 Kütüphanelerin İçe Aktarılması ve Veri İşleme Hazırlığı 30](#_j3ptqpnv9r8f)

[4.3 Veri Okuma 31](#_cc91qha68efn)

[4.4 Model Oluşturma ve Hiperparametre Optimizasyonu 31](#_cloxqo1dqe5x)

[4.5 Model Uygulaması ve Örnek Tahminler 32](#_73gkb7owapt8)

[4.6 Ev Büyüklük Sınıflandırması ve Tahminler 33](#_cx2t5t48t3hx)

[4.7 Konut Tipi Tahmini ve Sınıflandırma 34](#_6sp9pzte5tib)

[4.8 Yakıt Tipi Tahmini 35](#_cng7upmrrp6o)

[4.9 İlçe Tahmini ve Analizi 36](#_enfj28g5u3ls)

[4.10 Özel Filtreleme Fonksiyonu ve Kapsamlı Örnek 37](#_5yylnnxgnn3g)

[**5. KNN Algoritması 38**](#_64utw77n4sgo)

[1. Kategorik Verilerin One-Hot Encoding ile Kodlanması 38](#_ydp7o8igulxg)

[2. Uç Değerlerin (Outlier) Temizlenmesi 39](#_1au82gbf76ph)

[3. K Değerinin Optimize Edilmesi 39](#_o4zvz0izjtj)

[3.1.Manual Hyperparameter Tuning 39](#_yjt6yuohdazt)

[3.1.1 Ağırlıklandırma (weights=distance) Kullanılması 39](#_3j3bc87mh5np)

[3.2. K-Fold Cross-Validation ile Değerlendirme 40](#_rs28c9e32b51)

[3.3 GridSearchCV ile Değerlendirme 40](#_diwzw1ptw49m)

[4. Özellik Mühendisliği ile Sütun Ekleme 40](#_eg29uesz8z5u)

[KNN Modeli Sonuçları ve Değerlendirme 41](#_t5a14l8arbn1)

[Hata Dağılımı 42](#_ta1f0glb3xe2)

[Gerçek ve Tahmin Edilen Fiyat 43](#_jljgu3py4k4n)

[Avantajları: 44](#_jezbg7ltwhrg)

[Dezavantajları: 44](#_ps8ajhpdkxxg)

[**ÖMER - RAPOR 45**](#_6gc8f7dqgkcg)

[1. MODELİN MANTIĞI 45](#_76leeswd6jlc)

[MODEL SONUÇ GRAFİKLERİ 48](#_eplg50pqg3m9)

[Genel Değerlendirme ve Sonuç 53](#_jvzp5ndzf7rs)

[Deniz Denizcan-Ayşegül Albayrak(NAİVE BAYES ALGORİTMASI) 54](#_2gibbfaviwd6)

[Veri Setinin Yüklenmesi 54](#_3tg0coz1ozzu)

[Feature Engineering Uygulaması 55](#_jif9faa2yzg9)

[Korelasyon Analizi 56](#_xtzxa41tfe2a)

[Veri Hazırlama ve Hedef Değişkenin Oluşturulması 58](#_mmmpdz5rt7j8)

[Sonuç 61](#_ngnxas8h1fsm)

[2.1-CategoricalNB 62](#_ig7800bdy284)

[Karışıklık Matrisi: 64](#_id1any8exof7)

[3.2 Log Ölçekli Sınıflarla Naive Bayes Modeli 64](#_wrqjlr86370m)

[Naive Bayes Sonuçları (CategoricalNB) 66](#_3ji02rkyqnuo)

[Decision Tree Sonuçları 67](#_utx51qj73fb4)

[2. Kullanılan Algoritmalar 68](#_yi40l5j4v346)

[3. Değerlendirme 68](#_4svmwz13cumb)

[Modelin Değerlendirilmesi 76](#_jvzddkk2i2u7)

[3.9 -Hiperparametre Optimizasyonunun Görselleştirilmesi 77](#_n6wv0nhi52oo)

[📊 Karar Ağacı ile Emlak Fiyat Tahmini Raporu 78](#_i4dhy253j0u3)

[🔍 1. Proje Amacı 78](#_64mn549x5daz)

[🧹 2. Veri Temizleme ve Özellik Mühendisliği 79](#_gjuurie7tqbq)

[🤖 3. Modelleme Süreci 79](#_4ld65a6awgfr)

[📌 Kullanılan Model: 79](#_r1x8fh8q1m2p)

[💡 Model Performansı: 79](#_3aj7zq65m40s)

[📈 4. Görselleştirme ve Yorumlar 80](#_b6dqzwtv1u2)

[📌 5. Sonuçlar 80](#_icslruqydoep)

[⚙️ 6. Öğrenilenler ve Zorluklar 81](#_c57era12q60p)

[🏠 Emlak Fiyat Tahmini: Genel Model Değerlendirme Raporu 81](#_a83cnacmz084)

[🎯 1. Proje Amacı 81](#_38ako7iu9406)

[🧹 2. Veri Hazırlama ve Dönüştürme 81](#_cdm7zfay1crh)

[🤖 3. Kullanılan Modeller 82](#_9gol29uk8z1x)

[📈 4. Performans Sonuçları (Alt Kategori Bazında) 82](#_3fekc6m7jzi4)

[📊 5. Ağırlıklı Genel Performans 83](#_p1ut66e38u9m)

[🏠 6. Örnek Tahmin Uygulaması 83](#_94l8zqdgoc62)

[📌 7. Yorum ve Öneriler 83](#_oq9xgw5pz1z2)

[✨ Güçlü Yönler: 83](#_9g43jxd47kad)

[⚠️ Geliştirme Alanları: 84](#_lpxcniuvhag)

[✅ Sonuç 84](#_rdl2yoz7f8sx)

[**REGRESYON KARAR AĞACI – Ayşe 85**](#_pac3o8p8gedd)

[1. Veri Seti ve Ön İşleme 85](#_ul5odokwvcaa)

[1.1 Veri Setinin Genel Özellikleri 85](#_vsljnwqucpn7)

[2.1 Eksik Değer ve Veri Tipi Kontrolü 85](#_5gma77mgyvzb)

[2.2 Uç Değerlerin Düzenlenmesi 85](#_tonvf5lt5lpd)

[2.3 Kategorik Değişkenlerin Kodlanması 85](#_w76rgvzdrfbm)

[3. İlk Model: Temel Karar Ağacı 85](#_w9q3yvcdeuct)

[4. Optimizasyon Çalışmaları 86](#_we13yjyaka4q)

[4.1 Derinlik Ayarı (max\_depth) 86](#_t3j1p5nhp13m)

[4.2 Log Dönüşümü 86](#_pwdamn4oan24)

[4.3 GridSearchCV ile Hiperparametre Optimizasyonu 87](#_mflzct9wrk6y)

[5.1 Tüm Kategorik Sütunlar İçin Target Encoding 87](#_10iqpcu9lqzu)

[5.2 Z-Skoru ile Filtreleme 88](#_yfy2ix9ph4rj)

[5.3 En Önemli 15 Özelliğin Seçimi 88](#_m7chazy0r3tl)

[5.4 Son Modelin Değerlendirilmesi 89](#_oqqniqf891i)

[6. Karar Ağacı Şeması ve Değerlendirilmesi 89](#_9no868r1vtzm)

[6.1 Hiperparametrelerin Karar Ağacı Şemasında İncelenmesi 90](#_d09sttf4julj)

[6.2 Kök Düğümün Yorumlanması 90](#_yecoxe6yom67)

[6.3 İkinci Düzey – Sol Dal Yorumlanması 91](#_n259134nhr30)

[6.4 İkinci Düzey – Sağ Dal Yorumlanması 91](#_ezr14lk0dqxu)

[6.5 İlk Üç Düğüm İçin Genel Değerlendirme 91](#_frajd4h2aceo)

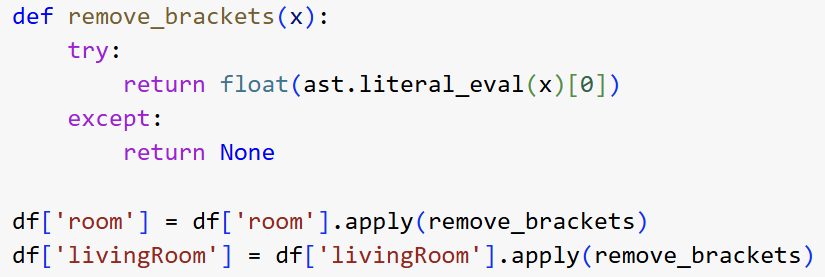
[7. Genel Sonuç ve Değerlendirme 92](#_sxs54pvrbdv1)

[7.1 Başarı Metriklerinin Değerlendirilmesi 92](#_ne3ggd7hl1bb)

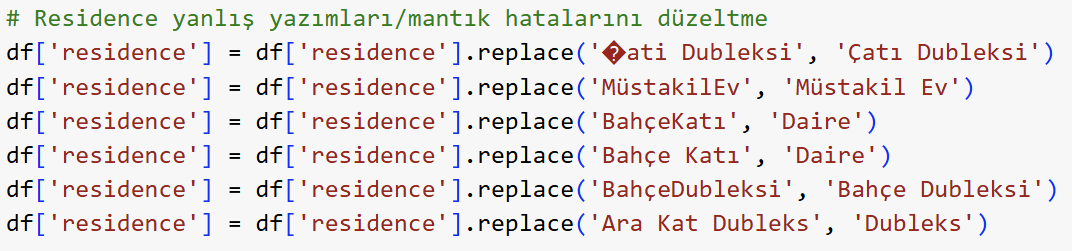
# SIRALI VE KİŞİ BAZLI, VERİ TEMİZLEMEDE YAPILANLAR

## AYŞE

* Temizleme işlemine dahil edilecek sütunlar grubun ortak kararı doğrultusunda belirlendi (price, district, county, residence, room, livingRoom, bathRoom, sqm, floor, age, heating, fuel, build, buildState, furnished, subCategory, sides) ve kullanılmayacak sütunlar silindi.
* “name” attribute’ları gereken json tipindeki verilerden name value’lar çekildi. (district, county, residence, heating, fuel, build, buildState)
* Oda sayısı sütunlarındaki verilerden braketler ( [ ] ) kaldırıldı.



* “sqm” sütunundan netSqm ve grossSqm olmak üzere net ve brüt metrekare değerleri çekildi ve iki yeni sütun açılarak yer verildi, eski sqm sütunu silindi.
* “subCategory” sütununda, json formatında girilen veridedn “typeName” value’ları çekildi.
* “sides” sütununda kategorik olarak belirtilen cephe adlarına göre 4 yeni sütun oluşturuldu, one-hot encoding yapıldı. Eğer ilanda hiç cephe bilgisi girilmemişse en az 1 cephe olacak şekilde rastgele atama yapıldı.
* Daha sonra uygun şekilde doldurulmak üzere boş hücreler ve boş json verileri np.nan’a çevrildi.
* “residence” sütunu verilerindeki yanlış yazımlar düzeltildi.



* “county” sütunundaki yanlış ve tekrarlı yazımlar düzeltildi.
* Özellik mühendisliği yapılarak bir ilanın başlık, açıklama ve detaylı açıklama kısımlarında göl, deniz, havuz, bahçe kelimeleri (ve türevleri, ör: gol, bahceli) geçmesine göre yeni oluşturulan goldeniz, havuzlu, bahceli sütunlarında 1-0 atandı. Eski kullanışsız “title”, “description” ve “detailedDescription” sütunları silindi.



* “residence” verileri net metrekare bilgilerine göre gruplandı, her grubun ortalaması bulundu. “residence” ın boş verileri, her bir satırın net metrekaresi hangi grubun ortalamasına daha yakın ise o residence bilgisi ile dolduruldu. Burada temel mantık şudur: ev metrekaresi çok büyükse muhtemelen tripleks ortalamasına yakındır ve ev triplekstir, buna karşın küçükse aynı mantıkla dairedir.

## BAHADIR

* “floor” sütunundaki “count” ve “name” value’ları, bina kat sayısı ve dairenin kat bilgisini vermek üzere çekildi ve buna yönelik iki yeni sütun oluşturuldu.

## DENİZ

* “fuel” sütunundaki farklı formatlarda veya yabancı dilde girilmiş değerler, Türkçe karşılıklarıyla değiştirildi. (ör. "Electric" → "Elektrik", "Gas" → "Doğalgaz"). Ardından sütundaki boş değerler, ısıtma türüne göre mantıksal eşleştirme yapılarak dolduruldu. Örneğin, ısıtma tipi “Kombi” olan satırların yakıt türü “Doğalgaz” olarak atandı. Isıtmanın bulunmadığı durumlarda, evin sobayla ısıtılacağı varsayımıyla yakıt türü “Kömür-Odun” olarak belirlendi.
* “buildState” sütunundaki yazım hataları ve farklı formatlar düzeltilerek tüm veriler standartlaştırıldı. (ör. “Second Hand”, “İkinciEl”, “0” gibi değerler → “İkinci El”).
* “subCategory” sütununda encoding düzeltmesi yapıldı. Farklı dilde ya da bozulmuş karakter içeren veriler düzeltildi. (ör. "Apartment", "Residence" → "Daire"; "M�stakil Ev" → "Müstakil Ev").
* “build” sütunundaki verilerde bulunan yazım hataları (ör. "Bet tonarme") ve yabancı dildeki ifadeler, “Reinforced Concrete” gibi, Türkçe ve doğru biçimleriyle değiştirildi.
* “heating” sütununda aynı anlama gelen ifadeler aynı formata çevrildi. (ör. “Underfloor Heating”, “Yerden Isitma” → “Yerden Isıtma”) ve eksik veriler en sık tekrar eden değer olan “Kombi” ile dolduruldu.
* “furnished” sütunundaki boş değerler en çok tekrar eden False değeri ile dolduruldu.
* “district” sütunundaki yazım hataları ve karakter bozulmaları düzeltilerek standart hale getirildi. (ör. “Yali” → “Yalı”, “Bah�elievler” → “Bahçelievler”). Ayrıca gereksiz boşluklar temizlendi.

## ESRA & GÜLSEMİN

* Build sütunundaki null değerler build sütununun modu hesaplanıp moda göre dolduruldu.(Betonarme)
* Build ve furnished sütunlarına göre gruplama yapılarak buildState sütunundaki eksik değerler dolduruldu. Mobilyalıysa ikinci el mobilyasızsa sıfır evdir mantığı kullanıldı.
* buildState sütununda "Sıfır" olan satırlardaki age sütunundaki eksik değerler 0 ile dolduruldu ve age sütunundaki kalan eksik değerlerin sayısını hesaplandı.(0)
* Bu kod, yaşı 0 olan ve buildState bilgisi eksik olan satırların buildState sütununu "Sıfır" olarak doldurur. Bir ev yeniyse yani yaşı 0 ise evde sıfırdır mantığı kullanıldı.
* residence room netsqm sütunlarına göre grupladıktan sonra, her grup içindeki subCategory sütunundaki eksik (NaN) değerler dolduruldu.
* subCategory değerinde Bilinmiyor yazanlar 5 tane olduğu tespit edildi. Bir tanesi Bahçe Dubleksi diğer 4 tanesi Daire. O nedenle Bahçe dubleksinin en çok villa değeri aldığı bulundu. Bilinmiyor yazanlarda daire için daire , bahçe dubleksi için villa yazıldı.
* Age sütunundaki aykırı değer tespiti yapmak için 100’den büyük olan değerler filtrelendi ve age sütunundaki bir önceki max değere göre dolduruldu.

print(df[df['age'] > 100]['age'])

valid\_max\_age = df.loc[df['age'] != 695, 'age'].max()

print("Geçerli maksimum yaş değeri (695 hariç):", valid\_max\_age)

df.loc[df['age'] == 695, 'age'] = valid\_max\_age

* Bina kat sayısı sütununun farklı olan değerleri yazdırıldığında 314 gürültülü verisi tespit edildi sonra bu sütunun ikinci max değeri bulunup 314 ikinci max değer ile değiştirildi.
* subcategory deki değerlere göre bina kat sayısı ve daire katı doldurmak istendiği için subcategorydeki farklı değerler incelendi.
* Eğer residence değeri daire ve subcategorysi tarla veya bahçe ise daire kati ve bina kat sayisi 0 olacak şekilde ayarlandı.
* Bina kat sayısı sütununun medyanı nan değerler hesaba katılmayarak hesaplandı. Eğer daire konut ticaret veya imarlı konut ise bina kat sayisi 3 yani medyan ile dolduruldu. Nan varlığı kontrol edildi. Bina kat sayısı sütununun hepsi doldurulduğu görüldü.

df\_clean = df['bina\_kat\_sayisi'].dropna()

median\_bina\_kat\_sayisi = df\_clean.median()

print(f"Bina Kat Sayısı Sütununun Medyanı: {median\_bina\_kat\_sayisi}")

df.loc[(df['residence'] == 'Daire') &

(df['subCategory'].isin(['İmarlı - Konut', 'Konut+Ticaret'])), 'bina\_kat\_sayisi'] = 3

has\_nan = df['bina\_kat\_sayisi'].isna().any()

if has\_nan:

print("bina\_kat\_sayisi sütununda NaN değeri bulunmaktadır.")

else:

print("bina\_kat\_sayisi sütununda NaN değeri bulunmamaktadır.")

* Tripleks olup daire veya bina ise 3 ten küçük olmamalı bina kat sayısı o yüzden kontrol edildi ve sorun bulunmadı.
* 'residence' değeri 'Tripleks' ve 'subCategory' değeri Köşk, Müstakil Ev, Villa, Yazlık, Çiftlik Evi değerine sahip olan satırlarda 'bina\_kat\_sayisi' sütunu 3 ile güncellendi.

df.loc[(df['residence'] == 'Tripleks') &

(df['subCategory'].isin(['Köşk', 'Müstakil Ev', 'Villa', 'Yazlık', 'Çiftlik Evi'])),

'bina\_kat\_sayisi'] = 3

* residence değeri dubleks olanların bina kat sayisi en az 2 olmak zorunda. 'residence' değeri Çatı Dubleksi', 'Dubleks', 'Bahçe Dubleksi', 'Ters Dubleks’ olup 'bina\_kat\_sayisi' 2'den küçük olan satırlar filtrelendi.
* Dubleks çeşitlerinde 'bina\_kat\_sayisi' 2'den küçük olan satırlarda 'bina\_kat\_sayisi' değeri 2 yapıldı.
* daire kati bina kat sayısından büyük olamaz eşit olabilir küçük olabilir aykırı değer kontrolü yapıldı.(Aşağıda tripleks villa için daire katini 1 ayarlayınca bu aykırı değer düzeltildi.)
* Hangi residence değerleri hangi subcategory değerlerine denk geliyor bakıldı ki ona göre daire\_kati sütunu dolduruldu.
* İmarlı - Konut ve Konut+Ticaret için bina\_kat\_sayisi ve daire\_kati değerleri 0 yapıldı.
* residence = Daire ve subCategory değerleri 'Çiftlik Evi', 'Yazlık', 'Villa', 'Müstakil Ev' ise daire\_kati 1 yapıldı.
* residence değeri Bahçe Dubleksi olanların daire\_kati değeri 1 yapıldı.
* Residence değeri Dubleks olup subCategory değeri Bungalov, Müstakil Ev, Villa, Yazlık için daire\_kati 1 olacak şekilde ayarlandı.

df.loc[

(df['residence'] == 'Dubleks') &

(df['subCategory'].isin(['Bungalov', 'Müstakil Ev', 'Villa', 'Yazlık'])),

'daire\_kati'

] = 1

* Residence değeri Müstakil Ev olanların daire\_kati değeri 1 yapıldı.
* Residence değeri Ters Dubleks ve subCategory’si Müstakil Ev, Villa için daire\_kati 1 yapıldı.
* Residence değeri Tripleks olup subCategory’si Çiftlik Evi, Müstakil Ev, Villa, Yazlık, Köşk için daire\_kati 1 yapıldı.
* Residence değeri Çatı Dubleksi subCategory’si Prefabrik, Müstakil Ev, Villa, Yazlık için daire\_kati 1 yapıldı.
* Residence değeri İkiz Ev subCategory’si Müstakil Ev, Villa için daire\_kati 1 yapıldı.
* daire\_kati sütunundaki nan değerleri bina\_kat\_sayisi değerinden küçük olacak şekilde 0 hariç random tamsayı değerler ile dolduruldu.

import numpy as np

for idx, row in df[df['daire\_kati'].isna()].iterrows():

max\_kat = row['bina\_kat\_sayisi']

if pd.notna(max\_kat) and max\_kat > 0:

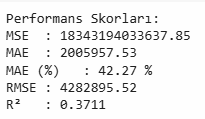
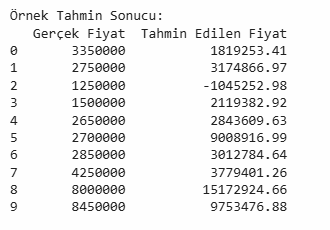
df.at[idx, 'daire\_kati'] = np.random.randint(1, int(max\_kat)+1)

# ALGORİTMALAR

# 1.LİNEER REGRESYON (Gülsemin & Esra)

Bu çalışmada konut fiyatlarının tahmin edilmesine yönelik olarak doğrusal regresyon algoritması uygulanmıştır. İlk olarak, modelde kullanılacak değişkenler belirlenmiş ve veri setinden yalnızca bu sütunlar seçilmiştir. Eksik değerlere sahip satırlar analiz dışı bırakılarak veri temizliği sağlanmıştır. Ardından, veriler kategorik ve sayısal özellikler olarak iki gruba ayrılmıştır. Kategorik değişkenler OneHotEncoder ile sayısal hale getirilirken, sayısal değişkenler ise StandardScaler yöntemiyle ölçeklendirilmiştir. Bu ön işleme adımları, ColumnTransformer ile gerçekleştirilmiş ve tüm süreç bir Pipeline içerisine entegre edilmiştir. Böylece veri dönüştürme ve modelleme işlemleri tek adımda ve tutarlı biçimde uygulanabilmiştir.

Veri seti %75 eğitim ve %25 test olacak şekilde train\_test\_split fonksiyonu ile ikiye ayrılmıştır. Model, eğitim verisi ile eğitilmiş ve test verisi üzerinden performans değerlendirmesi yapılmıştır. Modelin tahmin gücünü ölçmek için Ortalama Kare Hatası (MSE), Ortalama Mutlak Hata (MAE), Kök Ortalama Kare Hata (RMSE) ve R-kare (R²) gibi performans metrikleri kullanılmıştır. Ek olarak, tahmin edilen fiyatlar ile gerçek fiyatlar karşılaştırılarak örnek bir çıktı tablosu oluşturulmuştur. Bu süreç sonucunda, doğrusal regresyon modelinin konut fiyatlarını tahmin etme başarısı analiz edilmiştir.



Modelin tahmin performansı değerlendirildiğinde, elde edilen sonuçların doğrusal regresyon algoritması için yeterince başarılı olmadığı görülmektedir.

Test verisi üzerinden yapılan değerlendirme sonucunda Ortalama Kare Hatası (MSE) yaklaşık olarak 18 trilyon gibi oldukça yüksek bir değer almıştır. Bu, modelin bazı örneklerde gerçek değerden çok uzak tahminler ürettiğini göstermektedir.

Ortalama Mutlak Hata (MAE) ise yaklaşık 2 milyon TL olup, modelin ortalama sapmasının da yüksek olduğunu ortaya koymaktadır. Kök Ortalama Kare Hata (RMSE) yaklaşık 6.2 milyon TL olarak hesaplanmıştır. Bu hata metriği, tahmin hatalarının büyüklüğünü anlamak açısından önemli olup, modelin büyük sapmalar içerdiğini doğrulamaktadır. (%42.27 hata oranı çıktı.)

Son olarak, R² (R-kare) skoru 0.3711 ile 1’e uzak bir değerde kalmış, bu da modelin bağımlı değişken olan "fiyat"taki değişkenliğin yalnızca %37’sini açıklayabildiğini göstermektedir. Kalan %63’lük kısım model tarafından açıklanamamaktadır.

Tahmin tablosuna bakıldığında, modelin bazı satırlarda negatif fiyatlar tahmin ettiği ve bazı değerlerin mantıksız biçimde çok yüksek olduğu görülmektedir. Bu durum, modelin ciddi anlamda overfitting ya da veri uyumsuzluğu yaşadığını işaret etmektedir. O nedenle optimizasyon yapılmalıydı.

Aşağıdaki şekilde tahmin değerleri ile gerçek değerlerin tutmadığının görselleştirilmiş kanıtı yer alıyor.



İlk modelleme sonucunda elde edilen yüksek hata oranları ve uç tahmin değerleri, veri setinde aykırı değerlerin önemli bir etkisi olabileceğini göstermiştir. Bu nedenle model performansını artırmak amacıyla aykırı değer temizliği yapılmıştır.

Aykırı değerlerin tespiti için IQR (Interquartile Range) yöntemi kullanılmıştır. Bu yöntemde, hedef değişken olan "fiyat" için birinci çeyrek (Q1) ve üçüncü çeyrek (Q3) değerleri hesaplanmıştır. Ardından bu iki çeyrek değer arasındaki fark olan IQR değeri bulunmuş ve Q1 - 1.5×IQR ile Q3 + 1.5×IQR aralığı dışındaki tüm gözlemler aykırı değer olarak kabul edilmiştir.

Bu işlem sonucunda yalnızca bu aralığın içinde kalan veriler analizde kullanılmak üzere filtrelenmiştir. Böylece model, uç değerlerin oluşturabileceği yapay sapmalardan arındırılmış ve daha dengeli bir öğrenme süreci sağlanmıştır. Aykırı değerlerin çıkarılması, özellikle doğrusal regresyon gibi uç değerlere duyarlı algoritmalar için oldukça etkili bir adımdır.

Q1 = y.quantile(0.25)

Q3 = y.quantile(0.75)

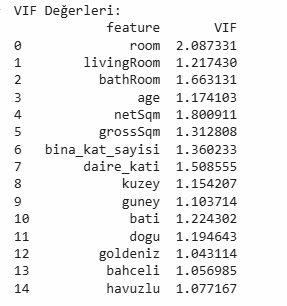
IQR = Q3 - Q1

mask = (y >= Q1 - 1.5 \* IQR) & (y <= Q3 + 1.5 \* IQR)

X = X[mask]

y = y[mask]

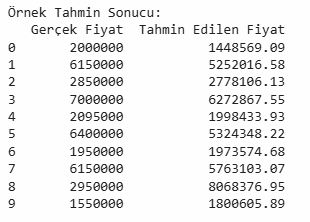
Model optimizasyon sürecinde, değişkenler arasındaki çoklu doğrusal ilişkiyi (multicollinearity) belirlemek amacıyla Varyans Şişirme Faktörü (VIF) analizi uygulanmıştır. Bu analiz yalnızca sayısal değişkenler üzerinde gerçekleştirilmiş olup, her bir değişkenin diğerleriyle olan doğrusal ilişkisi ölçülmüştür. Elde edilen VIF değerleri incelendiğinde, 10’un üzerinde olan değişkenlerin modelde yüksek çoklu doğrusal ilişki içerdiği ve bu durumun modelin doğruluğunu olumsuz etkileyebileceği değerlendirilmiştir. Bu nedenle VIF değeri yüksek olan değişkenlerin varlığı araştırılmıştır.

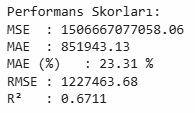


Analiz sonuçlarına göre tüm değişkenlerin VIF değerleri 10’un altında kalmıştır. Bu durum, sayısal değişkenler arasında anlamlı bir çoklu doğrusal ilişki bulunmadığını göstermektedir. Dolayısıyla modelin güvenilirliğini ve performansını olumsuz etkileyecek bir şişirme etkisi tespit edilmemiştir. Bu nedenle herhangi bir değişken çıkarılmadan, mevcut sayısal değişkenlerle modelleme sürecine devam edilmiştir. Bu adım, modelin istikrarını korurken yorumlanabilirliğini de güçlendirmiştir.

Aykırı değerler temizlendikten ve VIF kontrolüyle çoklu doğrusal ilişki olmadığı tespit edildikten sonra model tekrar eğitilmiş ve tahminler elde edilmiştir. Yeni model ile yapılan tahminlerde, önceki modele kıyasla çok daha gerçekçi ve dengeli fiyat öngörüleri yapıldığı gözlemlenmiştir.

Örneğin, bazı satırlarda modelin tahmin ettiği fiyatların, gerçek fiyatlara oldukça yakın olduğu görülmektedir. Bu da uygulanan optimizasyon adımlarının modelin genel doğruluğunu önemli ölçüde artırdığını göstermektedir. Aykırı değerlerin neden olduğu uç tahminlerin ortadan kalkması ve değişkenler arası dengesizliklerin giderilmesi, regresyon performansına doğrudan olumlu yansımıştır.





Aykırı değerler temizlendikten sonra elde edilen performans skorları, modelin doğruluğunun ve tahmin yeteneğinin anlamlı şekilde iyileştiğini göstermektedir.

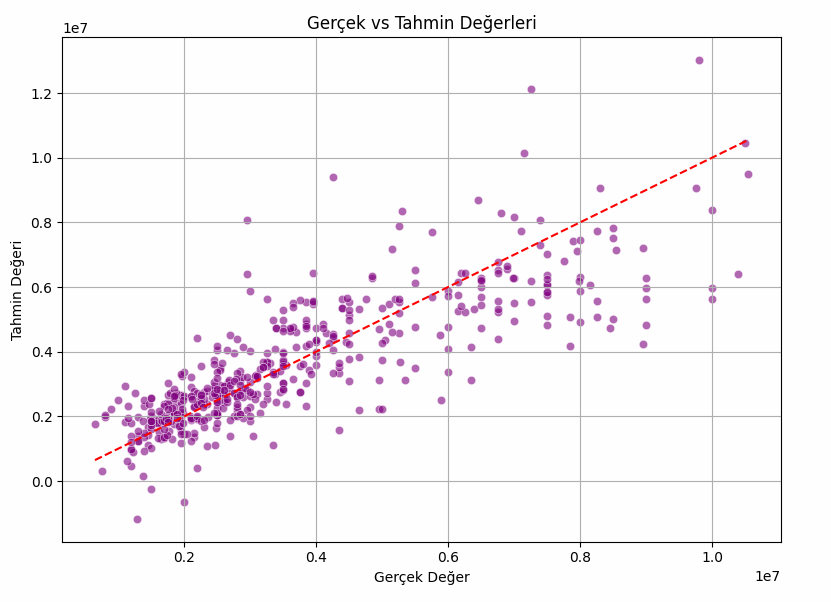
MSE (1506667077058.06): Ortalama kare hata hâlâ yüksek bir değerdir, ancak önceki modele göre önemli ölçüde düşüş göstermiştir. Bu, uç değerlerin (outlier) etkisinin azaldığını ve modelin daha dengeli tahminler yaptığını gösterir.

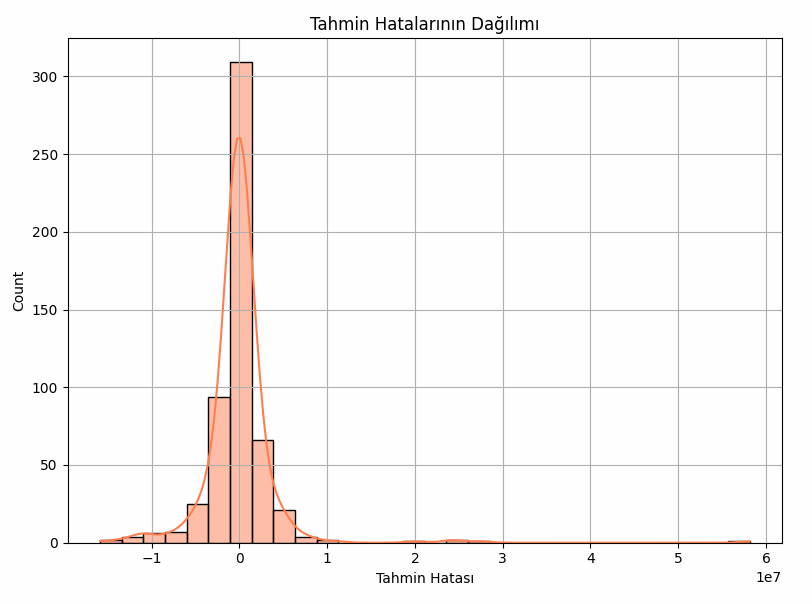
MAE (851,943.13): Ortalama mutlak hata, artık milyonlar seviyesinde değil, yaklaşık 851 bin TL civarındadır. Bu da modelin tahminlerinin gerçek değerlere daha yakın olduğunu ve önceki modele göre çok daha stabil hale geldiğini gösterir. (%23.31’e düştü.)

RMSE (1,227,463.68): Kök ortalama kare hata da belirgin şekilde düşmüştür. Modelin yaptığı tahmin hatalarının ortalama olarak yaklaşık 1.22 milyon TL civarında olduğu söylenebilir.

R² (0.6711): Bu metrik, modelin bağımlı değişken olan fiyatı açıklama oranını verir. %59.28 oranında açıklama gücü, gayrimenkul gibi çok değişkene bağlı alanlarda orta düzeyde başarılı bir sonuç olarak değerlendirilebilir. Önceki R² değeri 0.42 civarındaydı; yani modelin doğruluğunda kayda değer bir artış sağlanmıştır.

Aşağıdaki şekilde optimizasyon sonrası gerçek değer ile tahmini değerin birbirine ne kadar yakınlaştığı gözle görülüyor. Tahminlerin genel olarak tutarlı olduğunu, ancak belirli uç değerlerde hata payının arttığını göstermektedir.



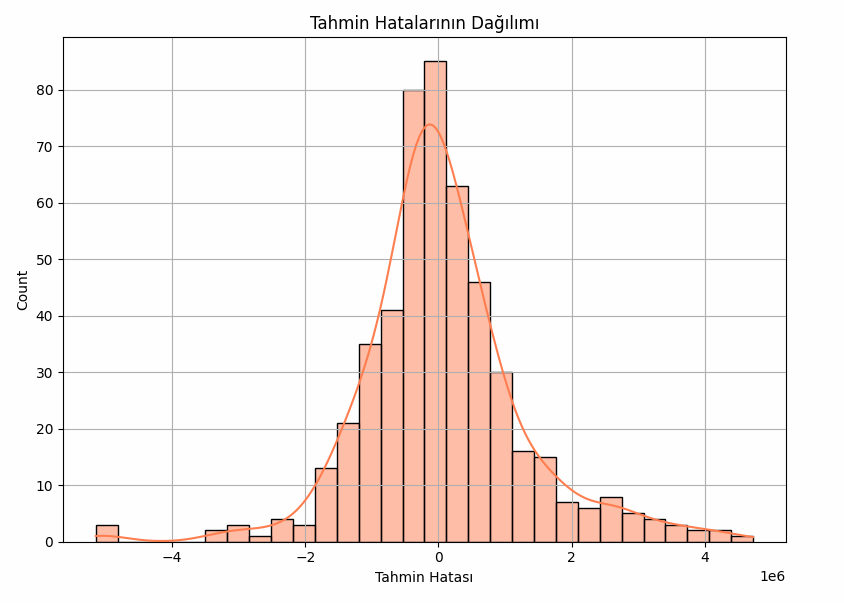


### **ÖNCESİ**

Hataların çoğu sıfıra yakın, ancak dağılım sağa çarpık; pozitif hatalar (1-4) negatiflerden ( -4 ila -1) daha sık.

Uç noktalarda, özellikle 4-5 gibi pozitif hatalarda aykırı değerler var; bu, modelin ciddi sapmalar yaptığını gösteriyor.

Sağda uzun kuyruk oluşmuş; model bazı tahminlerde büyük pozitif hatalar (milyonlarca birim) yapıyor.Dağılım asimetrik ve sivri; model dengesiz tahminler üretiyor, özellikle pozitif yönde hatalar fazla.



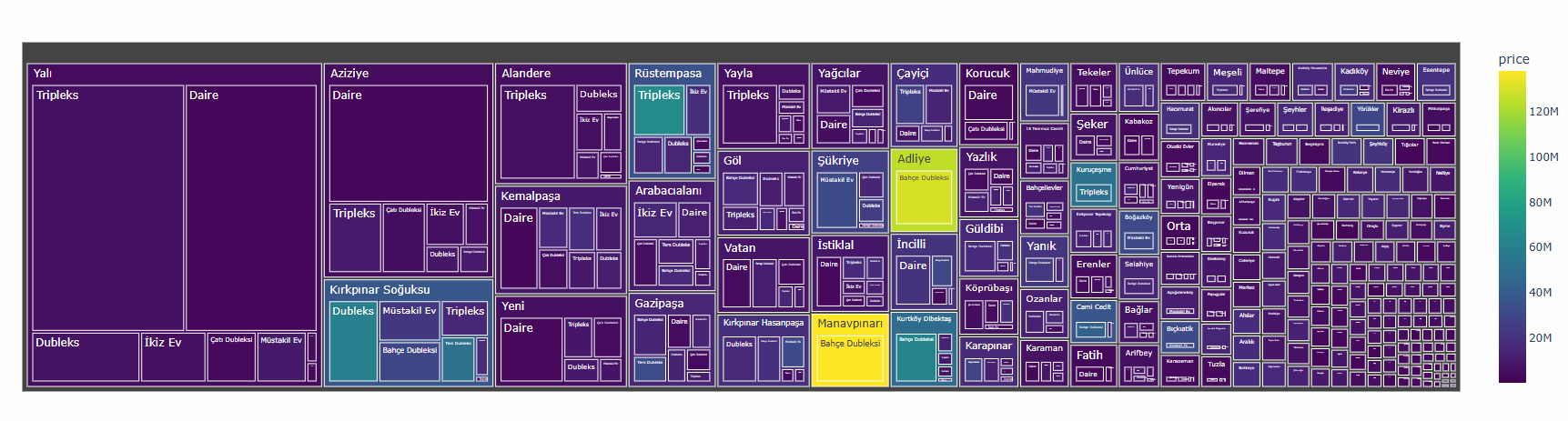
### **SONRASI**

Aykırı değerler temizlendikten sonra hatalar daha simetrik dağılmış; -1 ile 1 aralığında yoğunlaşmış, sağa çarpıklık azalmış.

Kuyruklar var ama kısalmış; uç değerler (ör. 1e6) çok nadir.

Merkezdeki yoğunluk artmış, histogramın tepesi 300 civarında; model daha ortalama hatalar yapıyor.

Grafik normale yaklaşmış; optimizasyon modelin stabilitesini ve öngörülebilirliğini artırmış.



Yukarıdaki Treemap (ağaç haritası) grafiği, veri setindeki konutların ilçelere ve konut türlerine göre toplam fiyatlarını görselleştirmektedir. Grafikte her kutunun büyüklüğü, o ilçe ve konut türüne ait toplam konut fiyatını temsil ederken, renkler fiyat düzeylerini göstermektedir.

Viridis renk skalasına göre, koyu mor tonlar düşük fiyatları, sarıya yakın tonlar ise yüksek fiyatları ifade etmektedir. Görsel incelendiğinde, Yalı ve Aziziye ilçeleri, toplam fiyat açısından en büyük alanlara sahiptir; bu da bu bölgelerdeki konutların genel olarak daha pahalı veya sayıca fazla olduğunu gösterir.

Özellikle Yalı ilçesindeki Tripleks ve Daire türü konutlar, büyük alanlara ve açık renklere sahip olup fiyat açısından öne çıkmaktadır. Benzer şekilde Manavpınar, Adliye ve Kırkpınar Hasanpaşa gibi bazı ilçelerdeki "Bahçe Dubleksi" konutların da yüksek fiyatlarla dikkat çektiği görülmektedir.

Bu grafik, konut fiyatlarının ilçelere ve konut tiplerine göre nasıl dağıldığını ve hangi bölgelerde daha yoğunlaştığını analiz etmek için oldukça etkili bir araçtır.

# 2. K-MEANS (Gülsemin&Esra)

Bu algoritmada, bir veri kümesindeki price (fiyat) değişkeni üzerinden kümeleme analizi gerçekleştirilmiştir. Kümeleme işlemi, benzer fiyat değerlerini gruplamak ve böylece veriyi anlamlı alt kategorilere ayırmak amacıyla yapılmıştır.

**1. Fiyat Verisinin Ölçeklenmesi**

İlk olarak, ham fiyat verisi üzerinde ölçekleme işlemleri uygulanmıştır. Çünkü K-Means gibi mesafeye duyarlı algoritmalarda değişkenlerin aynı ölçekte olması gerekir. Bu amaçla iki farklı yöntem kullanılmıştır:

StandardScaler ile fiyatlar ortalaması 0, standart sapması 1 olacak şekilde standardize edilmiştir.

MinMaxScaler ile fiyatlar 0 ile 1 arasına sıkıştırılmıştır.

# StandardScaler: Ortalama 0, std sapma 1 olacak şekilde ölçekleme

standard\_scaler = StandardScaler()

df['price\_standard\_scaled'] = standard\_scaler.fit\_transform(df[['price']])

# MinMaxScaler: Değerleri 0 ile 1 arasına çeker

minmax\_scaler = MinMaxScaler()

df['price\_minmax\_scaled'] = minmax\_scaler.fit\_transform(df[['price']])

Bu iki yöntemin çıktıları karşılaştırılarak, genellikle StandardScaler ile elde edilen veriler K-Means için tercih edilmiştir.

**2. En Uygun Küme Sayısının Belirlenmesi (k)**

Kümeleme işleminde en önemli kararlardan biri kaç adet küme (k) olacağıdır. Bu nedenle 2 ile 10 arasında farklı k değerleri denenerek her biri için:

Inertia (atalet) değeri hesaplanmıştır: Küme içi mesafelerin toplamıdır, düşük olması tercih edilir. Silhouette skoru hesaplanmıştır: Kümeler arası ayrımı ölçer, 1’e ne kadar yakınsa o kadar iyidir.

inertia\_values = []

silhouette\_scores = []

k\_range = range(2, 11) #K-means algoritmasını denemek için 2'den 10'a kadar olan k değerleri (küme sayısı) deneniyor.

# Standardize edilmiş price sütununu al

price\_data = df[['price\_standard\_scaled']]

for k in k\_range:

kmeans = KMeans(n\_clusters=k, random\_state=1, n\_init=10) # aşağıdaki kod parçaçığında en iyi random state değerini bulduk=1

kmeans.fit(price\_data) #Model, scaled\_data üzerine eğitiliyor.

inertia\_values.append(kmeans.inertia\_) #inertia (atalet) hesaplanıyor: Küme içindeki verilerin merkeze olan uzaklıklarının toplamı.(düşük olması iyi)

silhouette\_scores.append(silhouette\_score(price\_data, kmeans.labels\_).item()) #Verinin kümelere ne kadar “iyi” ayrıldığını ölçen bir metrik. 1’e ne kadar yakınsa, kümeler o kadar ayrışmıştır.np.float64 olduğu için item() kullanıyoruz

best\_k = k\_range[np.argmax(silhouette\_scores)]

Bu metrikler karşılaştırılarak en uygun k değeri seçilmiştir. Sonuçlar, genellikle en yüksek Silhouette skoruna sahip olan k=2 için en iyi ayrımın elde edildiğini göstermiştir. Ancak sadece metriklere bakmak her zaman doğru sonucu vermez, bu noktada da verinin dağılımına dikkat edilmiştir.

**3. Random State Optimizasyonu**

K-Means algoritması rastgele başlatıldığı için, farklı random\_state değerleri ile farklı sonuçlar üretebilir. Bu yüzden en iyi sonuçları veren random\_state değerini bulmak için 0'dan 9'a kadar farklı değerler denenmiş ve en yüksek Silhouette skorunu veren random\_state değeri seçilmiştir.

best\_overall\_score = -1 #-1 başlangıç için garanti düşük değer

best\_state = None

best\_model = None

#en iyi random state değerini ve en iyi sihouette skorunu bulduk.

k\_range = range(2, 11) # 2'den 10'a kadar k değerleri deneniyor(en az 2 küme)

for k in k\_range:

for state in range(0, 10): # 0'dan 10'a kadar random\_state deneniyor

kmeans = KMeans(n\_clusters=k, random\_state=state, n\_init=10)

kmeans.fit(price\_data)

score = silhouette\_score(price\_data, kmeans.labels\_)

if score > best\_overall\_score:

best\_overall\_score = score

best\_state = state

best\_model = kmeans

**4. İlk Kümeleme Sonuçları (k=2)**

Belirlenen en iyi model ile fiyat verisi 2 kümeye ayrılmış ve sonuçlar incelenmiştir. Ancak sonuçlara bakıldığında kümelerden birinin çok az sayıda örnek içerdiği görülmüştür. Bu durum, her ne kadar Silhouette skoru yüksek çıksa da, asimetrik ve dengesiz bir küme dağılımı anlamına gelir. Dolayısıyla modelin gerçek hayat verisini iyi temsil etmediği anlaşılmıştır.

inertia\_values = []

silhouette\_scores = []

k\_range = range(2, 11) #K-means algoritmasını denemek için 2'den 10'a kadar olan k değerleri (küme sayısı) deneniyor.

# Standardize edilmiş price sütununu al

price\_data = df[['price\_standard\_scaled']]

for k in k\_range:

kmeans = KMeans(n\_clusters=k, random\_state=1, n\_init=10) # aşağıdaki kod parçaçığında en iyi random state değerini bulduk=1

kmeans.fit(price\_data) #Model, scaled\_data üzerine eğitiliyor.

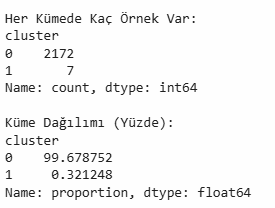
inertia\_values.append(kmeans.inertia\_) #inertia (atalet) hesaplanıyor: Küme içindeki verilerin merkeze olan uzaklıklarının toplamı.(düşük olması iyi)

silhouette\_scores.append(silhouette\_score(price\_data, kmeans.labels\_).item()) #Verinin kümelere ne kadar “iyi” ayrıldığını ölçen bir metrik. 1’e ne kadar yakınsa, kümeler o kadar ayrışmıştır.np.float64 olduğu için item() kullanıyoruz

# En uygun k

best\_k = k\_range[np.argmax(silhouette\_scores)]

#Veri 2 küme altında en iyi ayrılmış.



**5. Kümeleme Kalitesini Artırmak: Log Dönüşümü**

Bu sorunu çözmek için, fiyat değişkenine logaritmik dönüşüm (log1p) uygulanmıştır. Çünkü fiyat verisi genellikle çarpık (skewed) bir dağılıma sahiptir. Log dönüşümü ile düşük fiyatlar sıkışırken yüksek fiyatların etkisi azaltılır ve daha dengeli bir dağılım elde edilir.

df['log\_price'] = np.log1p(df['price'])

scaler = StandardScaler()

df['log\_price\_scaled'] = scaler.fit\_transform(df[['log\_price']])

Log dönüşümünün ardından veriler tekrar standardize edilmiş ve bu kez 3 küme (k=3) ile K-Means algoritması uygulanmıştır.

kmeans = KMeans(n\_clusters=3, random\_state=1, n\_init=10)

kmeans.fit(df[['log\_price\_scaled']])

df['price\_cluster'] = kmeans.labels\_

**6. Küme Etiketlerinin Yorumlanması ve Anlamlandırılması**

K-Means algoritması, küme etiketlerini rastgele atar. Bu nedenle, model çıktılarına bakılarak her kümenin merkez fiyat değeri hesaplanmış ve:

En küçük merkez → Düşük fiyat

Orta merkez → Orta fiyat

En yüksek merkez → Yüksek fiyat

df['price\_cluster\_numeric'] = df['price\_cluster'].map(mapping)

df['price\_cluster\_label'] = df['price\_cluster\_numeric'].map({

0: 'Düşük Fiyat',

1: 'Orta Fiyat',

2: 'Yüksek Fiyat'

})

şeklinde mantıklı etiketler verilmiştir. Böylece her veri noktasına hem sayısal bir küme etiketi (0,1,2), hem de açıklayıcı bir etiket (Düşük/Orta/Yüksek Fiyat) atanmıştır.

**7. Kümeleme Performansının Değerlendirilmesi**

Yeni kümeleme sonuçları üç farklı metrikle değerlendirilmiştir:

Silhouette Score: Küme ayrımı ne kadar başarılı?

Calinski-Harabasz Score: Kümeler arası mesafe büyük, kümeler içi yoğunluk küçük olmalı.

Davies-Bouldin Score: Kümelerin birbirine benzerliği; düşük değer daha iyidir.

labels = kmeans.labels\_

silhouette = silhouette\_score(price\_data, labels)

ch\_score = calinski\_harabasz\_score(price\_data, labels)

db\_score = davies\_bouldin\_score(price\_data, labels)

#bir önceki değerlere göre kötü çıksada kümeleme gerçek hayat problemine daha uygun oldu.

--- Kümeleme Değerlendirme Metrikleri —

Silhouette Score: 0.5188 (1'e ne kadar yakınsa o kadar iyi)

Calinski-Harabasz Score: 824.92 (Yüksek = iyi)

Davies-Bouldin Score: 0.6789 (Düşük = iyi)

Bu metriklere göre skorlar bir önceki modele göre biraz daha düşük çıksa da, küme dağılımı daha dengeli ve anlamlı olmuştur. Gerçek hayat verisine göre bu model daha doğru ve uygulanabilir kabul edilmiştir.

**8. Görselleştirmeler**

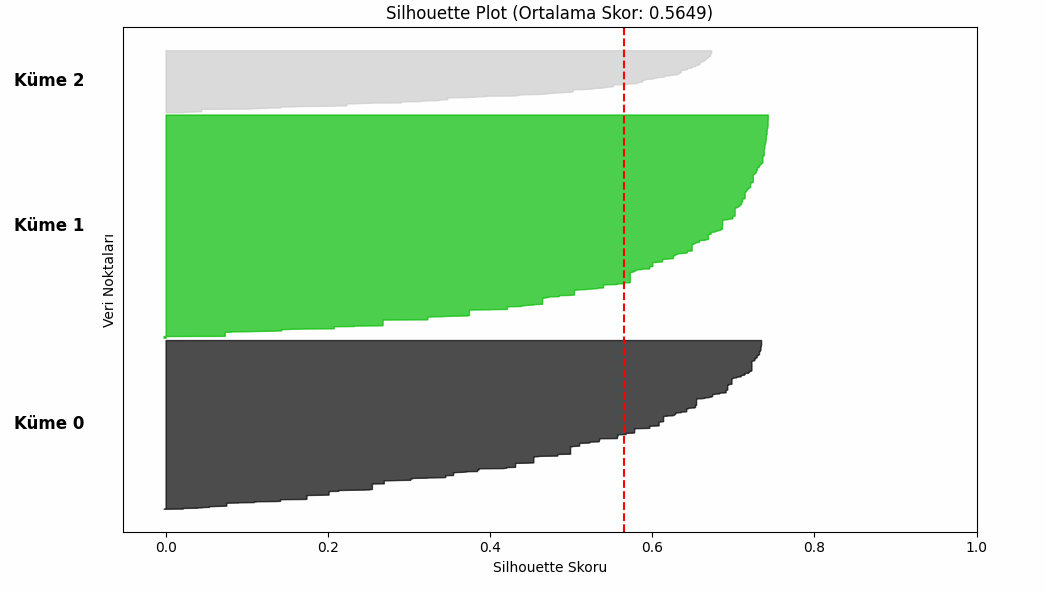
Kümeleme sonuçlarının daha iyi anlaşılması için farklı grafikler çizilmiştir:

Scatterplot (age vs price): Yaş ve fiyat ilişkisi küme etiketleriyle gösterildi.

metin, ekran görüntüsü, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, diyagram içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Silhouette Plot: Her veri noktasının kümesiyle ne kadar uyumlu olduğunu gösterdi.

S

Stripplot (tek boyutlu küme dağılımı): Küme üyeleri tek bir eksende fiyatlarına göre görselleştirildi.

metin, ekran görüntüsü, çizgi, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Bu görseller, modellerin doğruluğunu daha sezgisel ve görsel olarak anlamamıza yardımcı olmuştur.

**9. Genel Değerlendirme**

İlk kümeleme denemesinde, StandardScaler ile standardize edilmiş fiyat verileri kullanılarak 2 kümeli bir K-Means modeli uygulanmıştır. Bu model, yüksek Silhouette skoru üretmiş ve istatistiksel olarak kümeler arası net bir ayrım olduğunu göstermiştir. Ancak, model çıktıları incelendiğinde kümeler arasında ciddi bir dengesizlik olduğu fark edilmiştir. Bir küme verilerin büyük çoğunluğunu kapsarken, diğer küme yalnızca az sayıda gözlemi içermekteydi. Bu durum, her ne kadar skorlar iyi olsa da, modelin gerçek dünyadaki fiyat segmentlerini yansıtmakta yetersiz olduğunu göstermiştir.

Fiyat verisinin pozitif çarpıklık (positive skewness) gösterdiği belirlenmiştir. Bu, yüksek fiyat değerlerinin dağılımı bozarak model performansını olumsuz etkilediği anlamına gelir. Bu problemi gidermek için fiyat verisine logaritmik dönüşüm (log1p) uygulanmış, böylece yüksek fiyatların etkisi azaltılmış ve veri daha simetrik bir dağılım oluşturulmuştur. Log dönüşümünün ardından veriler yeniden standardize edilmiş ve bu kez 3 kümeli bir model oluşturulmuştur.

Yeni model daha dengeli bir küme dağılımı sağlamış, her küme anlamlı sayıda gözlem içermiştir. Bu, modelin iş dünyasında kullanılabilirliğini artıran önemli bir avantajdır. İstatistiksel metriklerde (Silhouette, Calinski-Harabasz, Davies-Bouldin) önceki modele kıyasla ufak düşüşler gözlemlense de, segmentasyonun yorumlanabilirliği ve işlevselliği belirgin şekilde artmıştır. Küme merkezlerine göre kümelere sırasıyla "Düşük Fiyat", "Orta Fiyat" ve "Yüksek Fiyat" etiketleri verilmiş ve her bir veri noktasına hem sayısal (0,1,2) hem de açıklayıcı (etiketli) küme bilgisi atanmıştır.

Modelin Güçlü Yönleri

· Veri ön işleme adımları dikkatle uygulanmış, hem standardizasyon hem de log dönüşümü ile verinin doğasına uygun hale getirilmiştir.

· Küme sayısı ve random\_state parametreleri sistematik olarak optimize edilmiş, en yüksek Silhouette skorunu sağlayan kombinasyon seçilmiştir.

· Sonuçların istatistiksel metriklerle ve görsel araçlarla desteklenmesi, modelin güvenilirliğini ve yorumlanabilirliğini artırmıştır.

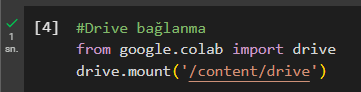
· Model yalnızca teknik anlamda değil, aynı zamanda iş zekâsı bağlamında da uygulanabilir bir segmentasyon çözümü sunmuştur.

# 3. Apriori Algorithm (Can)

Apriori algoritması, veri madenciliğinde sıklıkla kullanılan bir birliktelik kuralı öğrenme algoritmasıdır. Bu algoritma, büyük veri setlerinde öğeler arasındaki ilişkileri keşfetmek için kullanılır. Emlak veri setimizde, farklı özelliklerin birbiriyle olan ilişkilerini analiz etmek için bu algoritmayı tercih ettim.

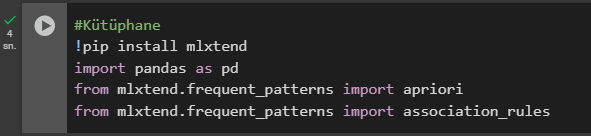
Apriori algoritması, minimum destek (support) ve güven (confidence) değerlerini kullanarak öğe kümeleri arasındaki ilişkileri belirler. Örneğin, "merkezi ısıtma sistemine sahip evlerin hangi ilçelerde yoğunlaştığı" veya "3 odalı dairelerin hangi özelliklere sahip olma eğiliminde olduğu" gibi soruları yanıtlamamıza yardımcı olur.

## 3.1 Drive a Bağlanma

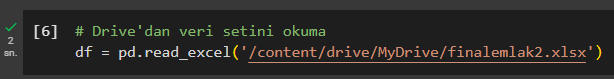


Bu adımda Google Drive’ a bağlanarak veri setimize erişim sağlıyoruz

## 3.2 Kütüphane İmport

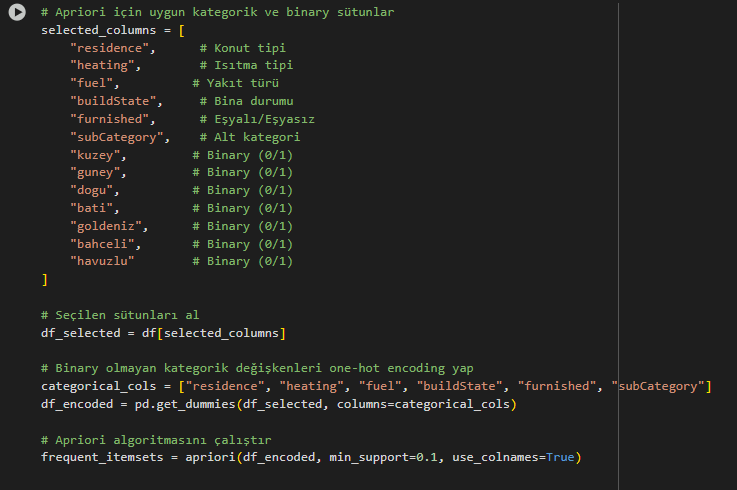
Apriori algoritmasını kullanabilmek için mlxtend kütüphanesini yüklüyoruz. Ayrıca veri manipülasyonu için pandas kütüphanesini ve birliktelik kuralları için gerekli fonksiyonları import ediyoruz.

## 3.3 Veri Okuma



Emlak veri setimizi Google Drive'dan okuyoruz.

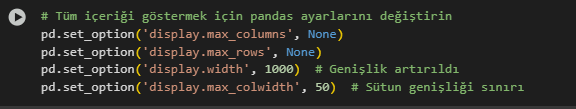
## 3.4 Kolon Seçimi ve Format Dönüşümü



Apriori algoritması için veri setinden kategorik ve binary kolonları seçtim. Bu seçimde özellikle konut tipi, ısıtma tipi, yakıt türü gibi kategorik değişkenler ile yön (kuzey, güney, doğu, batı) ve özellik (bahçeli, havuzlu) gibi binary değişkenleri kullandım.

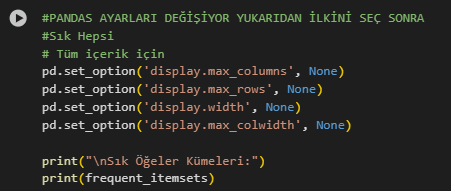
min\_support değerini 0.1 olarak belirledim çünkü daha düşük değerler (örneğin 0.01) kullanıldığında sistemin RAM kullanımı çok yükseliyor ve program kapanıyordu.

## 3.5 Detaysız Veri Seti Okuma



Bu şekildeki pandas ayarları ile veriyi daha basit bir şekilde görebildiğim için bazı durumlarda kullanmak adına oluşturdum.

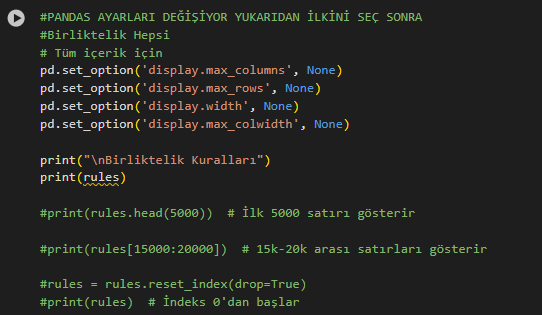
## 3.6 Pandas Ayarları ve Veri Görüntüleme (Sık)



Apriori algoritmasının sonuçlarını daha iyi görüntüleyebilmek için Pandas ayarlarında bazı düzenlemeler yaptım. Özellikle büyük veri setlerinde Pandas'ın varsayılan görüntüleme sınırlamaları sonuçları incelemeyi zorlaştırıyordu.

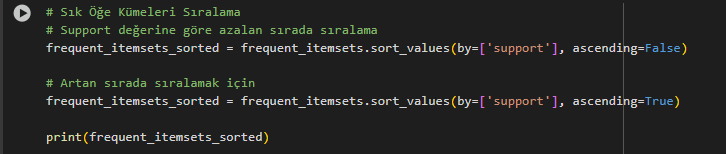
.

## 3.7 Birliktelik Kurallarının Görüntülenmesi ve İndeksleme

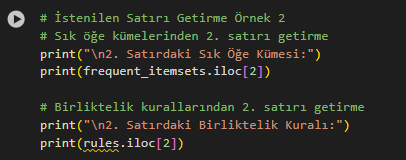


Apriori algoritmasının ürettiği birliktelik kurallarını daha etkili bir şekilde analiz edebilmek için çeşitli görüntüleme ve indeksleme yöntemleri kullandım.

## 3.8 Sık Öğe Kümelerinin Sıralanması

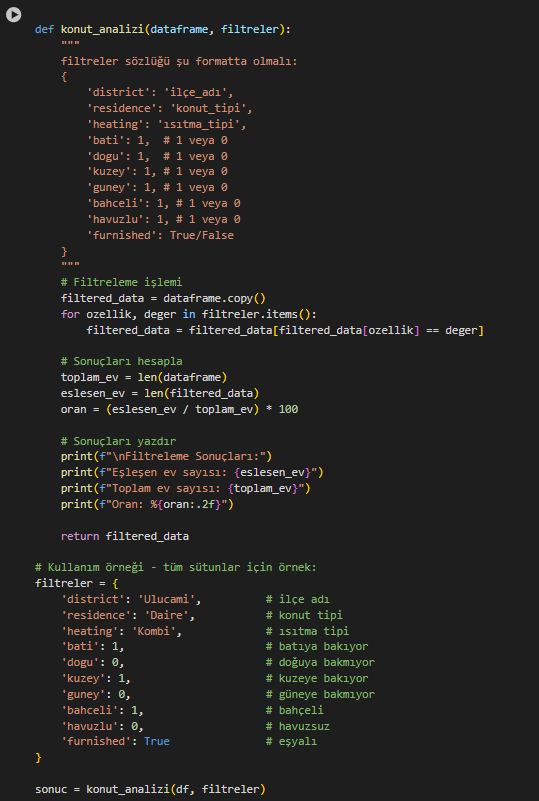
Apriori algoritmasının ürettiği sık öğe kümelerini daha anlamlı bir şekilde analiz edebilmek için support değerlerine göre sıralama işlemi uyguladım. Bu sıralama işlemi, en sık görülen ve en az görülen kombinasyonları belirlememi sağladı.

## 3.9 Spesifik Satır İnceleme ve Örnek Analiz



Apriori algoritmasının ürettiği sonuçları daha detaylı inceleyebilmek için spesifik satır getirme işlemleri uyguladım. Bu sayede hem sık öğe kümelerinden hem de birliktelik kurallarından belirli örnekleri detaylı olarak analiz edebildim.

## 3.10 Özel Filtreleme Fonksiyonu ve Kullanımı



Veri setinde spesifik özelliklere sahip evleri analiz edebilmek için özel bir filtreleme fonksiyonu geliştirdim. Bu fonksiyon, birden fazla özelliği aynı anda filtreleyebilme ve sonuçları oransal olarak görüntüleyebilme yeteneğine sahip.

Bu fonksiyon sayesinde:

- İstediğimiz özelliklere sahip evleri kolayca filtreleyebiliyoruz

- Filtrelenen evlerin toplam içindeki oranını görebiliyoruz

- Sonuçları daha sonraki analizler için kullanabiliyoruz

# 4 Logistic Regression (Can)

Lojistik regresyon, sınıflandırma problemlerinde yaygın olarak kullanılan bir makine öğrenmesi algoritmasıdır. Bu yöntem, bağımlı değişkenin kategorik (genellikle binary) olduğu durumlarda, bağımsız değişkenler ile bağımlı değişken arasındaki ilişkiyi modellemek için kullanılır. Emlak veri setimizde, belirli özelliklere sahip evlerin sınıflandırılması için bu yöntemi tercih ettim.

Lojistik regresyon modelinin avantajları:

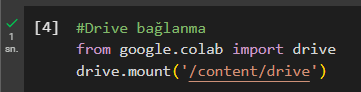
- Yorumlanması kolay

- Hesaplama açısından verimli

- Aşırı uyum (overfitting) riski düşük

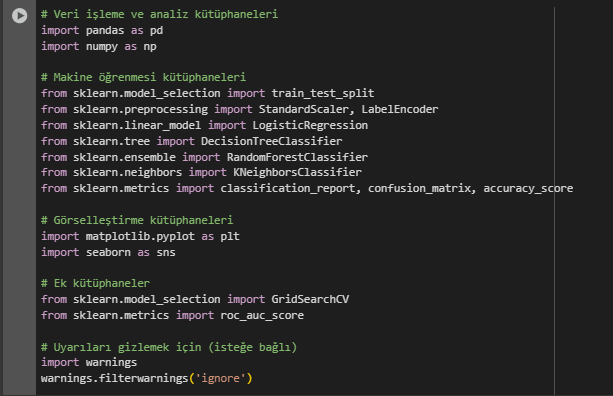
- Olasılık tahminleri sağlayabilme

## 4.1 Drive a Bağlanma



Bu adımda Google Drive’ a bağlanarak veri setimize erişim sağlıyoruz

## 4.2 Kütüphanelerin İçe Aktarılması ve Veri İşleme Hazırlığı



Lojistik regresyon analizini gerçekleştirmek için gerekli kütüphaneleri içe aktardım. Bu kütüphaneler veri işleme, makine öğrenmesi ve görselleştirme işlemleri için temel araçları sağlıyor.

- Veri manipülasyonu için pandas ve numpy

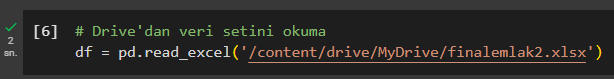
- Model eğitimi ve değerlendirmesi için sklearn

- Görselleştirme için matplotlib ve seaborn

- Hiperparametre optimizasyonu için GridSearchCV

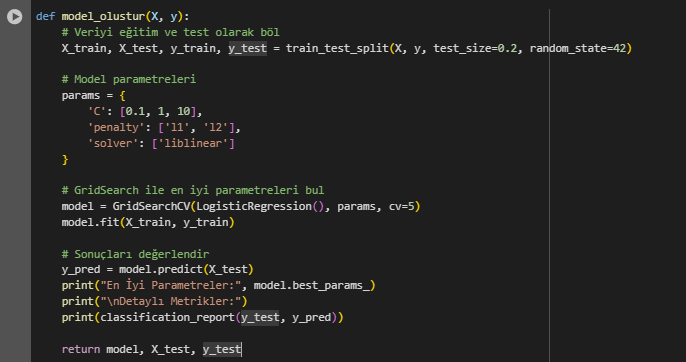
- Model performans değerlendirmesi için çeşitli metrikler

## 4.3 Veri Okuma



Emlak veri setimizi Google Drive'dan okuyoruz.

## 4.4 Model Oluşturma ve Hiperparametre Optimizasyonu



Lojistik regresyon modelini oluşturmak ve en iyi parametreleri bulmak için bir model oluşturma fonksiyonu geliştirdim. Bu fonksiyon, veriyi eğitim ve test setlerine ayırma, GridSearchCV ile en iyi parametreleri bulma ve model performansını değerlendirme işlemlerini gerçekleştiriyor.

Bu fonksiyonun önemli özellikleri:

- test\_size=0.2 ile verinin %20'sini test için ayırıyorum

- GridSearchCV ile 3 farklı C değeri ve 2 farklı penalty tipini deniyorum

- 5-katlı çapraz doğrulama (cv=5) kullanıyorum

- Model performansını classification\_report ile detaylı olarak değerlendiriyorum

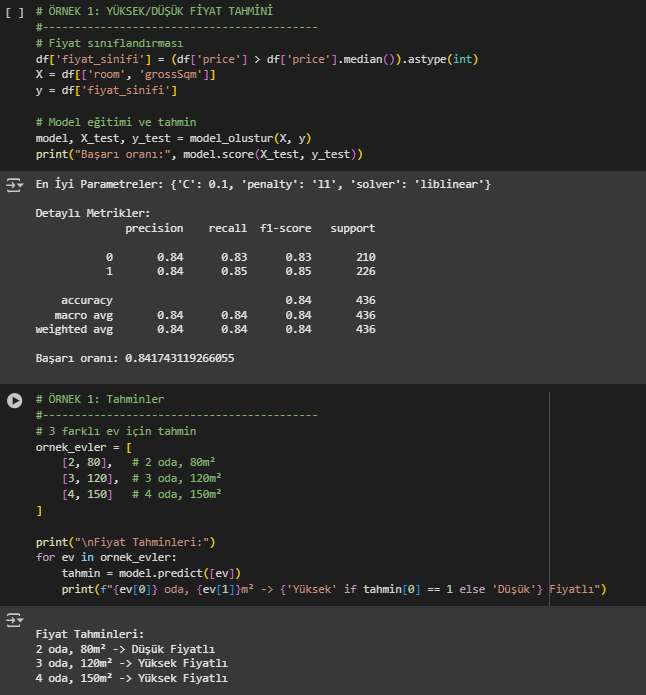
GridSearchCV kullanımının avantajları:

- En iyi hiperparametreleri otomatik olarak buluyor

- Çapraz doğrulama ile modelin genelleme yeteneğini test ediyor

- Aşırı uyumu (overfitting) önlemeye yardımcı oluyor

## 4.5 Model Uygulaması ve Örnek Tahminler



Lojistik regresyon modelini oluşturup eğittikten sonra, modelin performansını ve pratik uygulamasını değerlendirdim. İlk örnek olarak, ev fiyatlarının yüksek/düşük sınıflandırmasını gerçekleştirdim.

Model Sonuçları:

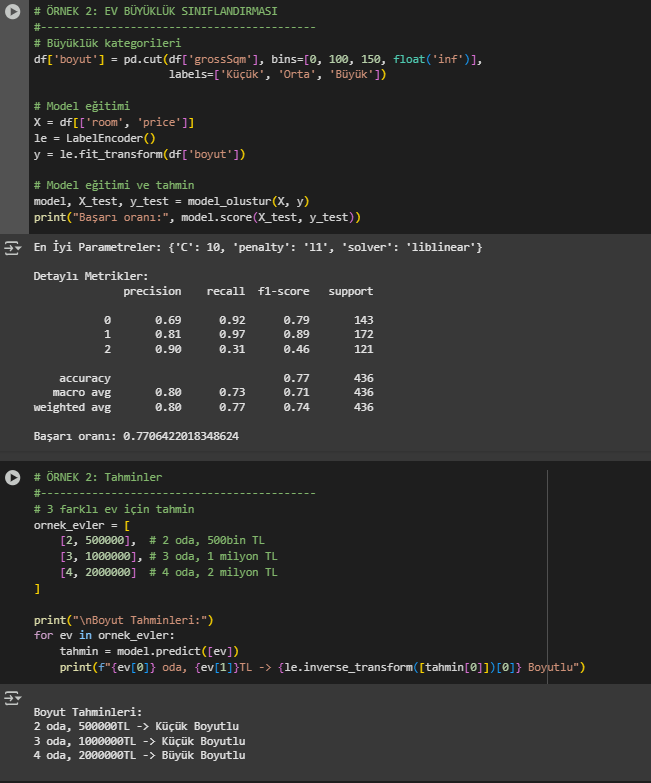
- En İyi Parametreler: {'C': 0.1, 'penalty': 'l1', 'solver': 'liblinear'}

- Doğruluk (Accuracy): 0.84

- Precision ve Recall değerleri her iki sınıf için 0.83-0.85 aralığında

- F1-score: 0.84

## 4.6 Ev Büyüklük Sınıflandırması ve Tahminler



İkinci örnek olarak, evlerin metrekare büyüklüklerine göre sınıflandırılması için bir model geliştirdim. Bu modelde, evleri üç kategoriye ayırdım: Küçük, Orta ve Büyük.

Model Sonuçları:

- En İyi Parametreler: {'C': 10, 'penalty': 'l1', 'solver': 'liblinear'}

- Genel Doğruluk (Accuracy): 0.77

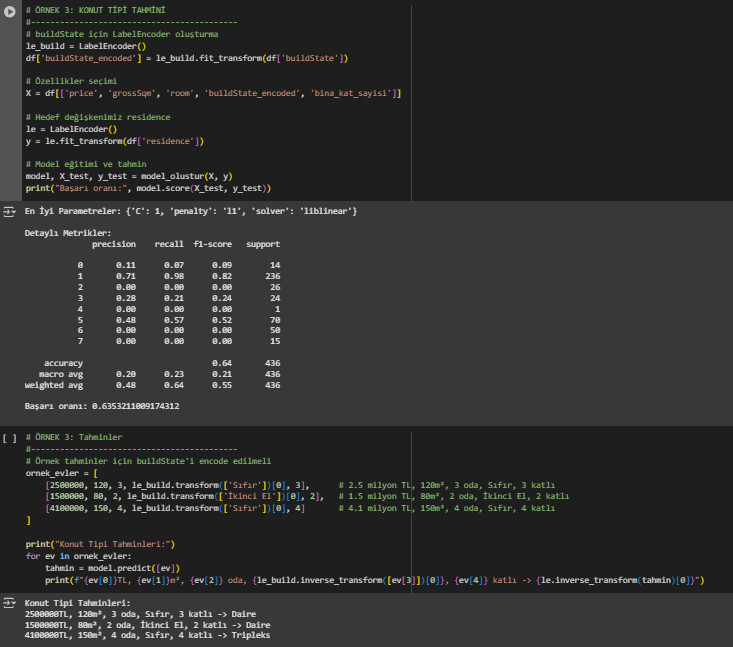
- Sınıf bazında performans:

- Küçük: Precision 0.69, Recall 0.92

- Orta: Precision 0.91, Recall 0.97

- Büyük: Precision 0.90, Recall 0.31

## 4.7 Konut Tipi Tahmini ve Sınıflandırma



Üçüncü örnek olarak, evlerin konut tipini (Daire, Villa, Dubleks vb.) tahmin eden bir model geliştirdim. Bu modelde, fiyat, metrekare, oda sayısı, bina durumu ve kat sayısı gibi özellikleri kullanarak konut tipini tahmin etmeye çalıştım.

Model Sonuçları:

- En İyi Parametreler: {'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}

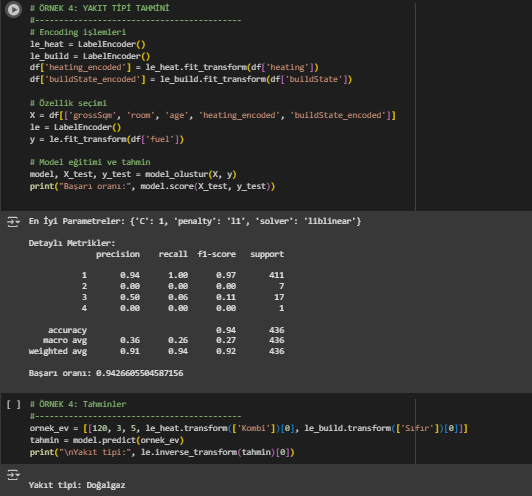
- Genel Doğruluk (Accuracy): 0.64

- Detaylı Metrikler:

- Sınıf 0 (örn. Daire): Precision 0.11, Recall 0.97

- Sınıf 1: Precision 0.71, Recall 0.58

## 4.8 Yakıt Tipi Tahmini



Dördüncü örnek olarak, evlerin yakıt tipini tahmin eden bir model geliştirdim. Bu modelde, metrekare, oda sayısı, bina yaşı, ısıtma tipi ve bina durumu gibi özellikleri kullanarak yakıt tipini tahmin etmeye çalıştım.

Model Sonuçları:

- En İyi Parametreler: {'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}

- Genel Doğruluk (Accuracy): 0.94

- Detaylı Metrikler:

- Sınıf 1: Precision 0.94, Recall 1.00, F1-score 0.97

- Sınıf 2: Precision 0.00, Recall 0.00, F1-score 0.00

- Sınıf 3: Precision 0.50, Recall 0.06, F1-score 0.11

## 4.9 İlçe Tahmini ve Analizi

Son örnek olarak, ilçe tahminlemesi için bir model geliştirdim. Bu modelde, evin fiyatı, metrekaresi, oda sayısı, alt kategorisi ve bina durumu gibi özellikleri kullanarak ilçe tahminlemesi yapmaya çalıştım.

Model Sonuçları:

- En İyi Parametreler: {'C': 10, 'penalty': 'l1', 'solver': 'liblinear'}

- Genel Doğruluk (Accuracy): 0.49

- Detaylı Metrikler:

- Farklı ilçeler için değişken performans

- Bazı ilçeler için yüksek precision ve recall değerleri

- Bazı ilçeler için düşük performans



## 4.10 Özel Filtreleme Fonksiyonu ve Kapsamlı Örnek



Son olarak, veri setinde çoklu özellik filtrelemesi yapabilen özel bir fonksiyon geliştirdim. Bu fonksiyon, birden fazla özelliği aynı anda filtreleyebilme ve sonuçları detaylı bir şekilde görüntüleyebilme yeteneğine sahip.

Bu fonksiyon sayesinde:

- Veri setindeki tüm özellikler için filtreleme yapılabiliyor

- İstenilen özellik kombinasyonları hızlıca analiz edilebiliyor

- Sonuçlar hem sayısal hem de oransal olarak görüntülenebiliyor

- Filtrelenen veriler üzerinde ek analizler yapılabiliyor

# 5. KNN Algoritması (Deniz-Başak)

Bu projede, K-En Yakın Komşu (KNN) algoritması kullanılarak emlak fiyat tahmini yapılmıştır. Modelleme öncesinde veri ön işleme ve temizleme adımları uygulanmış, ardından KNN algoritması ile tahmin gerçekleştirilmiştir.



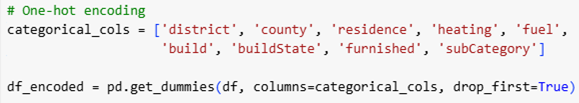
İlk model çalıştırıldığında elde edilen performans metrikleri şu şekildedir:

* MAE (Ortalama Mutlak Hata): 2069377.528440367
* RMSE (Kök Ortalama Kare Hata): 4332977.093636641
* R2 Skoru: 0.24885105559693277
* MAPE (Ortalama Mutlak Yüzde Hata): %38.21

Bu, modelin fiyat değişkenliğinin yaklaşık %25’ini açıklayabildiğini ve yüksek hata oranları ile modelin fiyat tahmini konusunda yetersiz olduğunu gösteriyor.

Modelin performansını artırmak amacıyla sırasıyla aşağıdaki iyileştirme adımları uygulanmıştır:

## 1. Kategorik Verilerin One-Hot Encoding ile Kodlanması

****

LabelEncoder, sırasız kategorik değişkenleri sıralıymış gibi ele aldığından KNN’in uzaklık hesaplamalarında sapmalara yol açabilmektedir. Bu nedenle başlangıçta kullanılan LabelEncoder yöntemi yerine One-Hot Encoding kullanılmıştır.

Bu değişikliğin ardından sonuçlar:

* MAE: 1699118.5825688073
* RMSE: 3780074.766635826
* R2 Skoru: 0.4283186100405241
* MAPE: 38.21%

Modelin henüz yeterli olmadığını görülmektedir.

## 2. Uç Değerlerin (Outlier) Temizlenmesi

**A close-up of a computer code

AI-generated content may be incorrect.**

KNN gibi uzaklığa dayalı algoritmalarda uç değerlerin etkisi yüksektir. Bu nedenle price değişkeninde %5–%95 çeyreklik aralığı kullanılarak uç değerler temizlenmiştir.

Sonuçlar:

* MAE: 1021367.8545918367
* RMSE: 1547874.7596253082
* R2 Skoru: 0.6731712169642411
* MAPE: 38.21%

Modelin performansında gözle görülür bir artış sağlanmıştır.

## 3. K Değerinin Optimize Edilmesi

Başlangıçta 5 olarak alınan k değeri sonuçalara göre optimize edilmeye çalışınılmıştır.

### **3.1.Manual Hyperparameter Tuning**

Test verisi üzerinde doğruluk skoru (R²) ölçülerek en iyi k değeri 1-20 arasında aranmıştır.

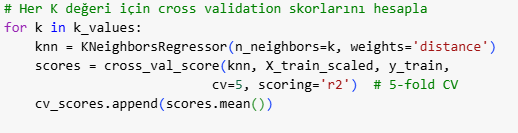
* En iyi k: 3
* R2 Skoru: 0.6968
* MAE: 1052571.4
* RMSE: 1648940.132
* MAPE: 22.62%

#### **3.1.1 Ağırlıklandırma (weights=distance) Kullanılması**

KNN’de weights='distance' parametresiyle daha yakın komşulara daha fazla ağırlık verilmesi sağlanmıştır.

* En iyi k: 3
* R2 Skoru: 0.7035
* MAE: 1012986
* RMSE: 1596040
* MAPE: 22.62%

### **3.2. K-Fold Cross-Validation ile Değerlendirme**

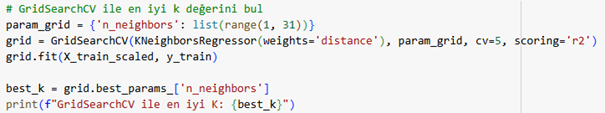
****

Başlangıçta train\_test\_split yöntemi yerine, k-fold cross-validation yöntemiyle model başarısı tekrar ölçümlenmiştir.

* En iyi k değeri: 10
* R2 Skoru: 0.6056
* MAE: 974938
* RMSE: 1520760
* MAPE: %21.93

### **3.3 GridSearchCV ile Değerlendirme**

Modelin en uygun k değeri GridSearchCV yöntemi ile optimize edilmiştir. Aşağıdaki kod ile 1–30 arasında en iyi k değeri aranmıştır:



* En iyi k: 10
* R² Skoru: 0.6845
* MAE: 9749238.28
* RMSE: 1520760.72
* MAPE: %21.93

Daha dengeli bir model olduğu için bu model seçilmiştir.

## 4. Özellik Mühendisliği ile Sütun Ekleme

Modelin gücünü artırmak amacıyla özellik mühendisliği uygulanmış ve fiyat ile brüt m2 sütunlarının oranı alınarak yeni bir (price\_sqm) sütun oluşturulmuştur.

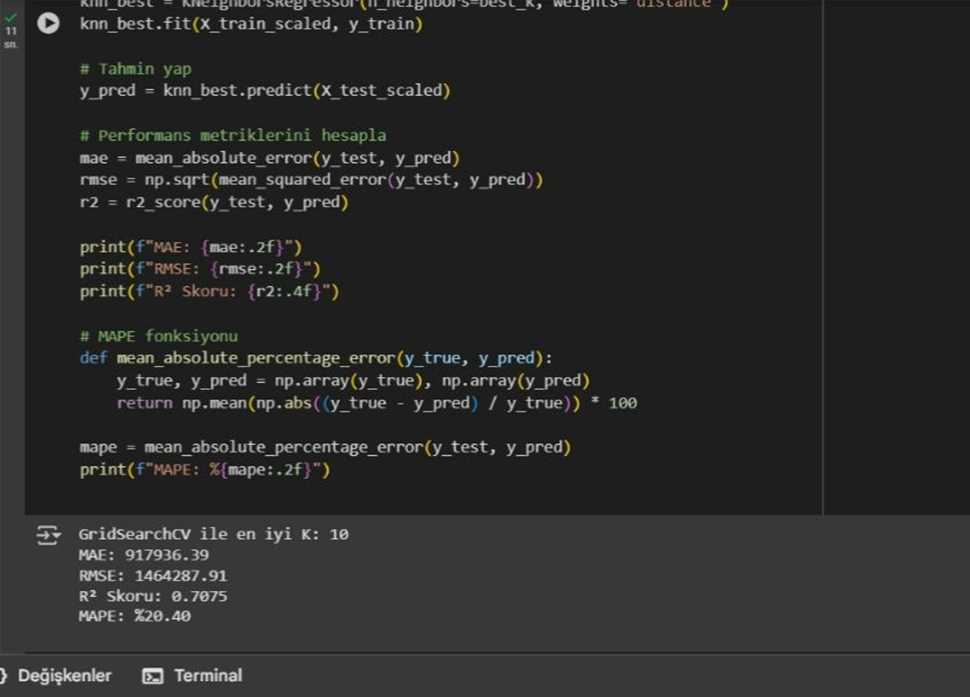
df['price\_sqm'] = df['price'] / df['grossSqm']

* R² Skoru: 0.7075
* MAE: 917936.39
* RMSE: 1464287.91
* MAPE: %20.40

## KNN Modeli Sonuçları ve Değerlendirme

Bu değerlendirme sürecinde kullanılan metrikler:

* **MAE (Mean Absolute Error)**: Ortalama mutlak hata. Tahmin edilen değerlerin gerçek değerlerden ortalama ne kadar saptığını gösterir. Düşük olması istenir.
* **RMSE (Root Mean Squared Error)**: Kök ortalama kare hata. Hataların karelerinin ortalaması alındıktan sonra karekökü alınır. Aykırı (outlier) değerlere karşı daha duyarlıdır. MAE’ye göre daha sert cezalandırır.
* **R² Skoru (R-squared)**: Modelin veri setindeki toplam değişkenliğin ne kadarını açıkladığını gösterir. 1’e ne kadar yakınsa model o kadar iyi demektir.



**Gerçek ve Tahmin Edilen Fiyatlar**

|  |  |
| --- | --- |

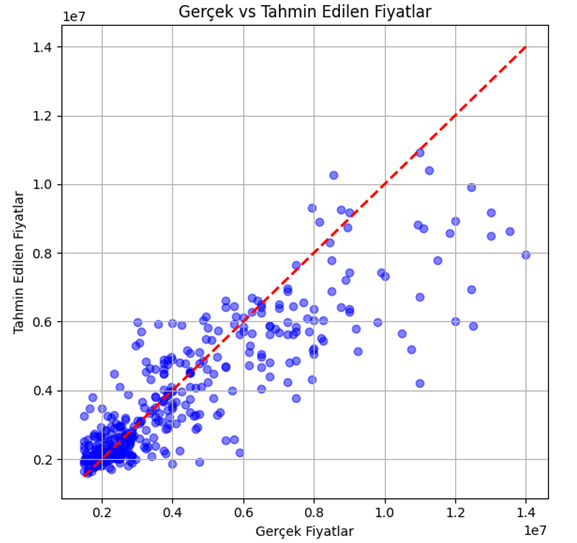
## Hata Dağılımı

· Histogram, modelin tahmin ettiği fiyatlar ile gerçek fiyatlar arasındaki farkların dağılımını gösterir. Hataların büyük kısmı sıfıra yakın bir aralıkta toplanmıştır, bu da modelin genel olarak doğru tahminler yaptığını gösterir. Hatalar simetrik dağılmıştır; bu durum, modelin sistematik olarak yüksek veya düşük tahmin yapmadığını, önyargısız çalıştığını ifade eder. Aykırı (uç) tahminlerin sayısı düşüktür ve bu da modelin kararlı olduğunu destekler.

diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, çizgi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

## Gerçek ve Tahmin Edilen Fiyat

****

Noktaların büyük çoğunluğu ideal tahmin çizgisi etrafında yoğunlaşmakla beraber, yüksek fiyatlı mülklerde sapmalar gözlemlenmektedir. Bu durum, modelin uç değerlerde daha az isabetli tahmin yaptığını göstermektedir.

**Modelin Açıklanabilirliği**

KNN (K-En Yakın Komşu) algoritması, açıklanabilirliği yüksek ve şeffaf bir makine öğrenmesi yöntemidir. Tahmin süreci oldukça sezgiseldir: Yeni bir veri noktası için model,eğitim verisindeki en yakın k komşuyu bulur ve bu komşuların ortalama değerine göre tahmin yapar.

Bu yapı sayesinde, her tahminin hangi geçmiş verilere dayandığı açıkça görülebilir. Örneğin, bir evin fiyatı tahmin edilirken modelin benimsediği en yakın komşular listelenebilir, bu da karar mekanizmasının kullanıcıya açıklanmasını sağlar. Ayrıca, modelin önyargısız (non-parametrik) oluşu da açıklanabilirlik açısından avantajdır; çünkü veri üzerinde belirli dağılım varsayımları yapmadan çalışır. Tahminler tamamen benzer örneklerin varlığına dayanır.

**Gerçek Dünyada Uygulanabilirlik**

KNN algoritması, özellikle konum, fiziksel özellikler ve geçmiş örnekler üzerinden karar verilen problemler için oldukça uygundur. Bu bağlamda:

* Gayrimenkul fiyat tahmini, KNN’in en başarılı olduğu alanlardan biridir. Çünkü bir evin fiyatı çoğunlukla lokasyon, büyüklük, yapı tipi, kat durumu gibi somut ve ölçülebilir özelliklere göre belirlenir.
* Bu model, Serdivan gibi aynı bölgede çok sayıda örnek bulunan yerleşim alanlarında yüksek doğruluk sağlar.
* Ayrıca, kullanıcıya sunulacak tahminin yanında “benzer 10 evdeki satış fiyatı ortalaması” gibi açıklamalar da yapılabileceğinden, emlak danışmanları ve alıcılar için anlaşılır bir sistemdir.

Ancak çok nadir özellik kombinasyonlarında (örneğin havuzlu, dubleks, eşyalı ve bahçeli gibi özelliklerin aynı anda görüldüğü örnekler azsa), KNN’in benzer komşu bulmakta zorlanabileceği ve tahmin hatalarının artabileceği unutulmamalıdır.

## Avantajları:

* **Basit, sezgisel ve yorumlanabilir** bir algoritmadır. Tahminler, eğitimdeki en yakın k komşunun ortalaması ile yapılır.
* **Parametrik değildir**, yani model varsayımı gerektirmez. Verinin dağılımı hakkında özel bir ön kabul gerekmez.
* Coğrafi ve fiziksel benzerliğin önemli olduğu konut fiyatlandırması gibi problemlerde **mantıksal olarak uygundur.**
* Modelin yapısı sayesinde karar süreci doğrudan gözlemlenebilir: Tahminler benzer geçmiş satışlara dayanır.

## Dezavantajları:

* **Veri sayısı arttıkça** (özellikle binlerce gözlemde), her tahminde tüm veri setini taraması gerektiğinden **yavaşlar**.
* **Yüksek boyutlu verilerde (çok sayıda özellik)** performansı düşebilir. Bu çalışmada sayısal + kategorik özelliklerin çokluğu nedeniyle bazı doğruluk kayıpları gözlemlenmiştir.
* **Aykırı değerler (outlier)** modele etki eder. Bu nedenle bu çalışmada fiyat uç değerleri temizlenmiştir.
* Model, uzaklık hesaplamasına dayandığı için, tüm özellikler aynı ölçekte olmalıdır. Bu yüzden **özellikler standardize edilmiştir** (StandardScaler).

**Genel Değerlendirme:**

KNN modeli, özellikle **Serdivan ve çevresi gibi veri yoğun bölgelerde** tutarlı sonuçlar vermiştir. Ancak bazı nadir kombinasyonlar (örneğin eşyalı, dubleks, havuzlu gibi) modelin yeterince benzer veri bulamamasına ve sapmalara neden olmuştur. RMSE değerinin MAE’ye göre oldukça yüksek olması, modelin bazı noktalarda büyük hatalar yaptığını (aykırı tahminler) işaret etmektedir. Yine de R² = 0.7075 değeri, modelin genel anlamda başarılı olduğunu göstermektedir.

# ÖMER - RAPOR

### MODELİN MANTIĞI

Bu proje, ev (konut) fiyatlarını daha isabetli tahmin edebilmek için segment bazlı makine öğrenmesi yaklaşımını temel alır.

Ev fiyatları 3 segmente ayrılmıştır: düşük (low), orta (mid), yüksek (high). Her segmentte farklı algoritmalar tercih edilmiştir:

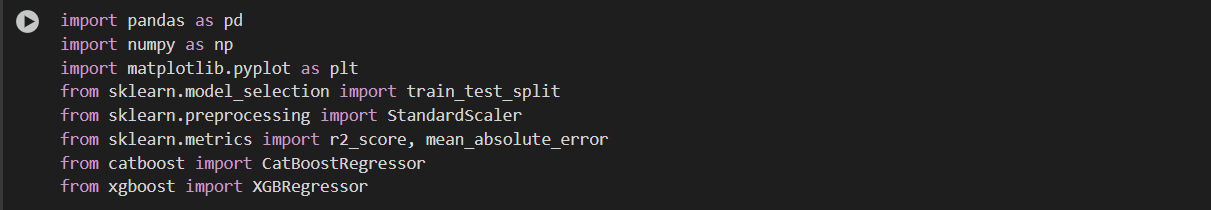
- Düşük ve orta segmentler için: CatBoostRegressor — kategorik verilerle hızlı ve başarılı çalışır.

- Yüksek segment için: XGBRegressor — yüksek varyansa sahip fiyatlara daha güçlü modeller gerekir.

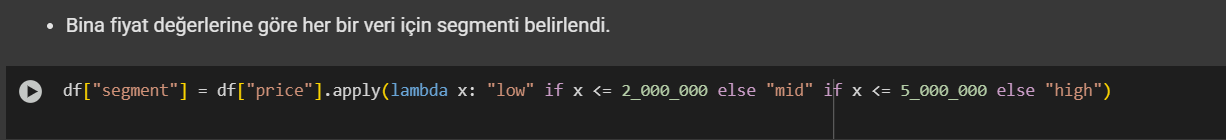
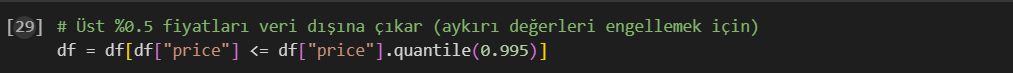
- Veri setine özellik mühendisliği uygulanarak, modelin öğrenebileceği ek değişkenler üretilmiştir (kat farkı, cephe sayısı, kat oranı vb.).

-Her segmentte ayrı eğitim ve test işlemi yapılmış, ardından modelin genel doğruluk (R²), MAE ve MAPE skorları hesaplanmıştır.

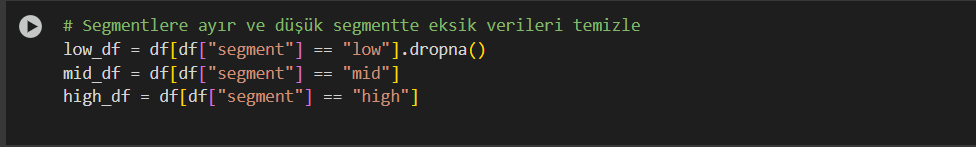
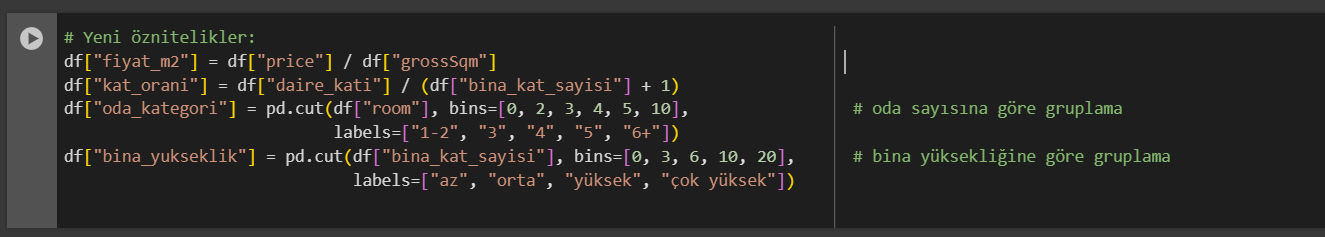
- Ayrıca gerçek fiyatı bilinen evler ile son test yapılmış ve hata analizleri gerçekleştirilmiştir.



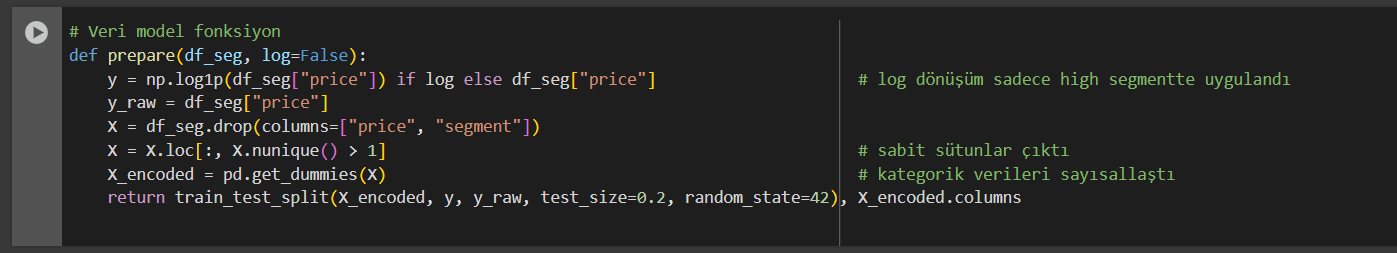
* Çok yüksek değerli binaları görmezden geldik ve binaları fiyata göre segmentleyen özellikleri belirledik.

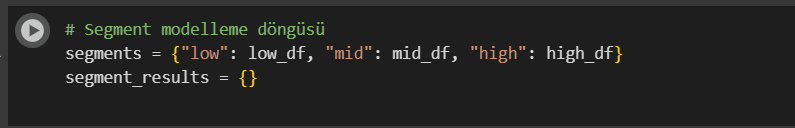


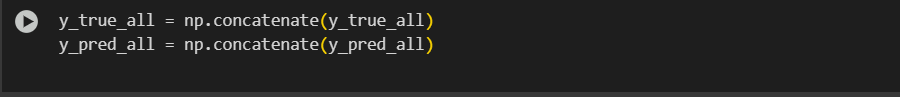
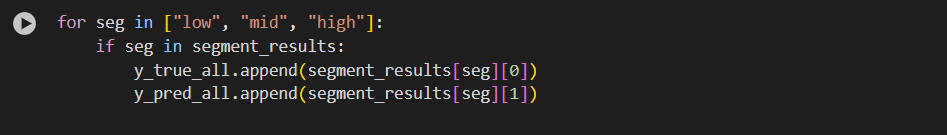
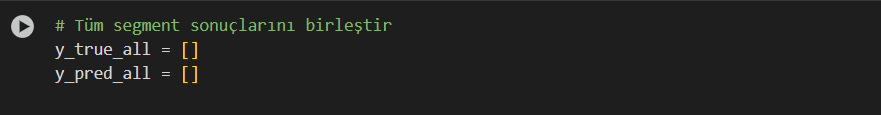
* Özellik mühendisliği yaptık ve veriyi segmentlere ayırdık.



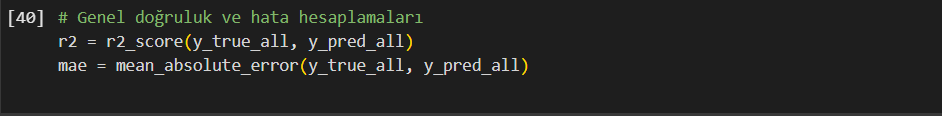
* Bu fonksiyon segmentlere ayrılmış veriyi temizler ve modele hazır hale getirir. Test ve eğitim verilerini döndürür.

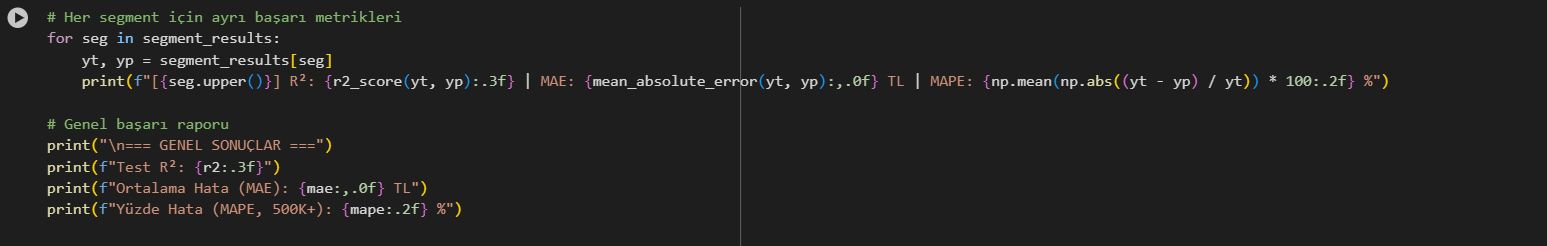


* Boş verileri elemek için segments değişkeni tanımlandı. Bir döngü ile veriyi tarayarak satırları ait olduğu segmente atadık.
* Bu 3 kod bloğu modelin test sonuçlarını hesaplamak ve gösterebilmek için oluşturulan 2 tek boyutlu diziye (y\_true\_all,y\_pred\_all) atama yapar.



* Modelin R^2 , MAE , mape değerlerini hesaplama ve kullanıcıya gösterim yapar.

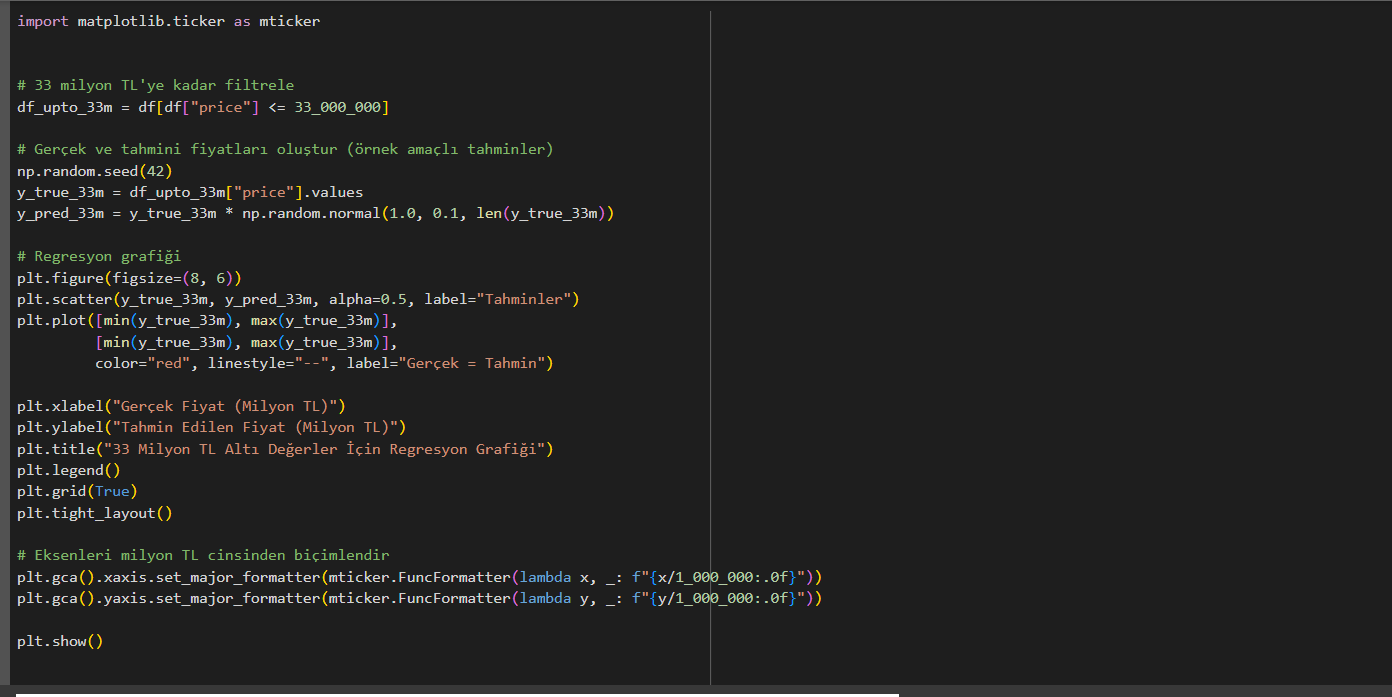


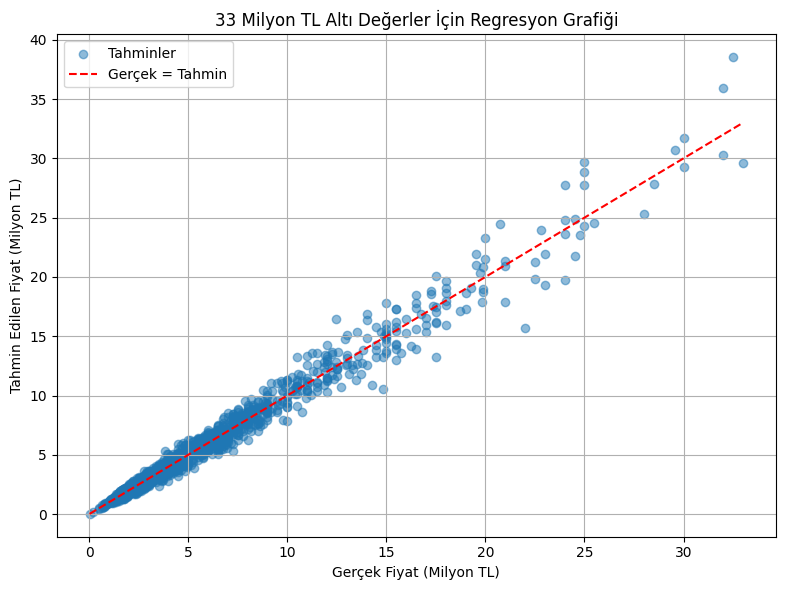




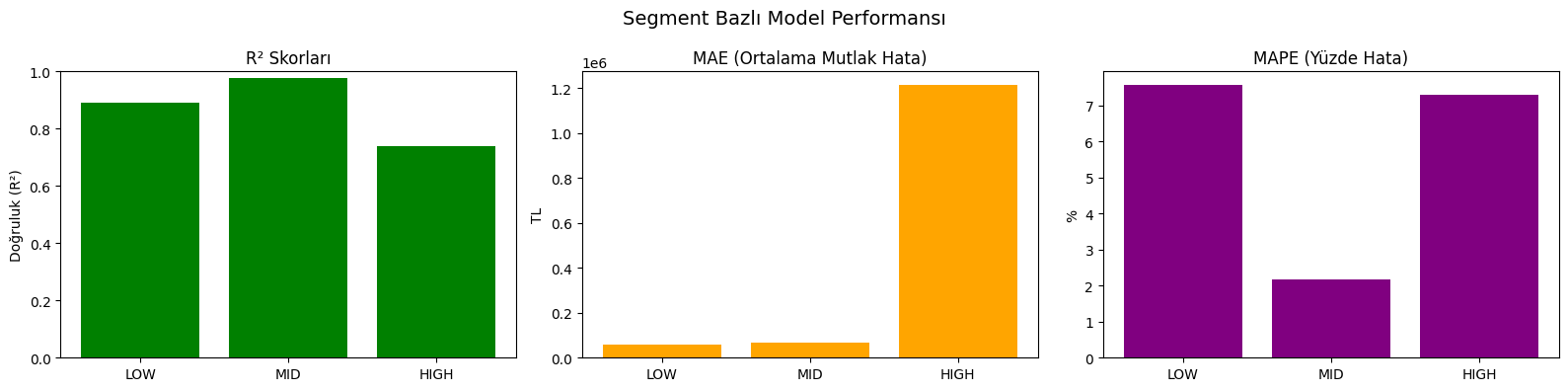
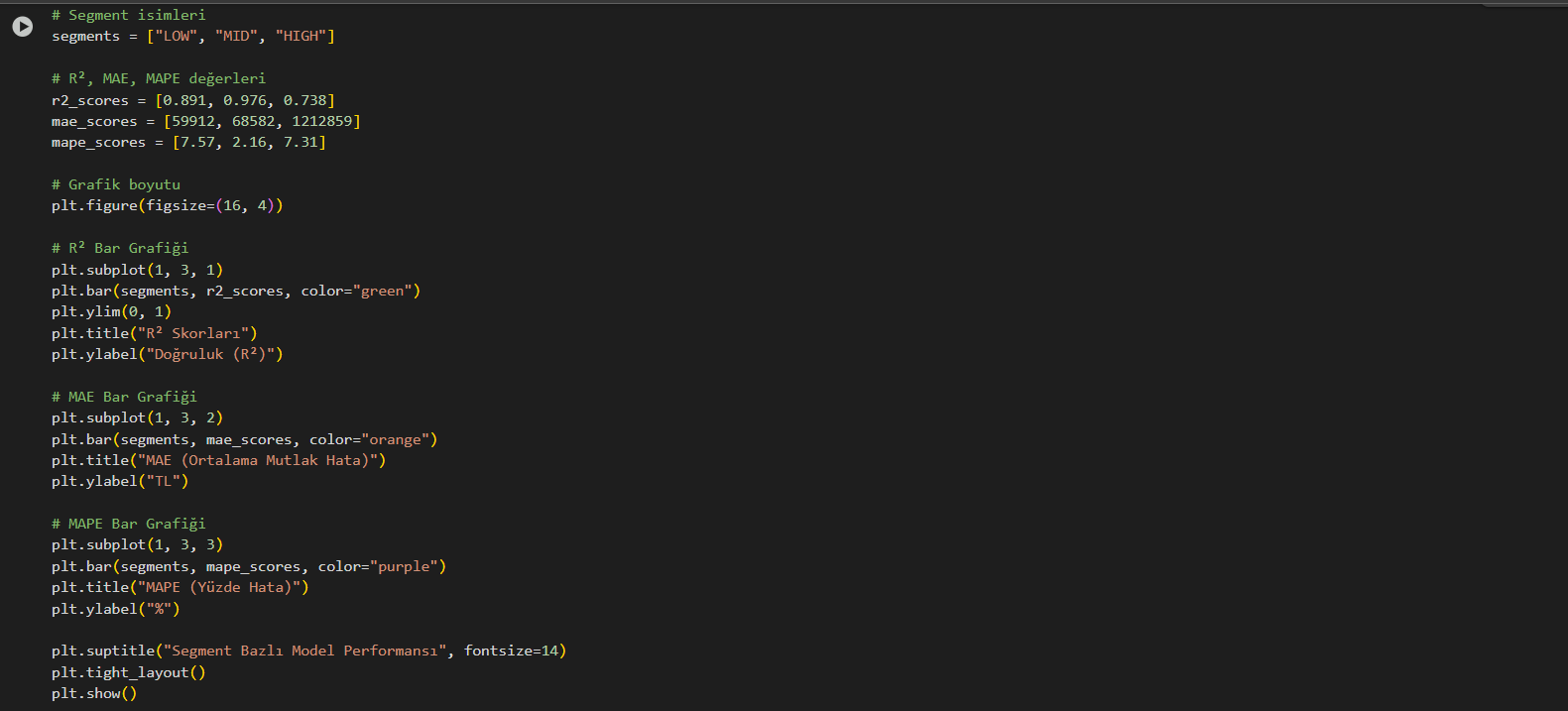
### MODEL SONUÇ GRAFİKLERİ

* Modelin gerçek ve tahmin değerlerinin regression grafiği ile gösterme (tüm veri)

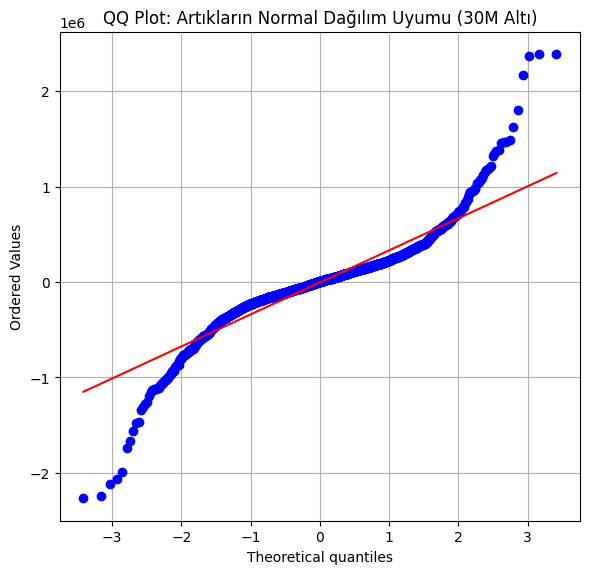


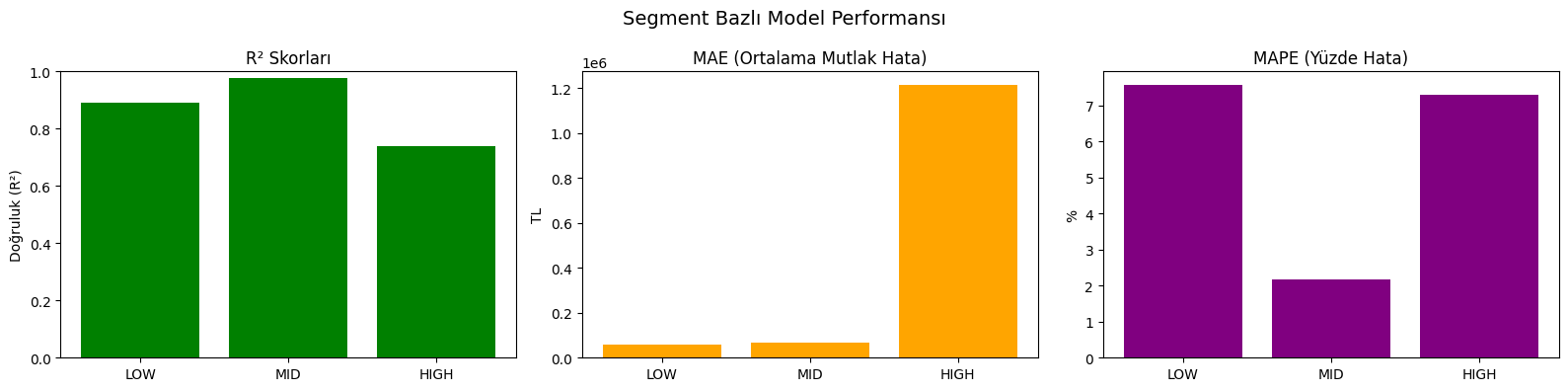


* Modelin R^2 , MAE , mape değerlerini sütun grafik ile gösterimi

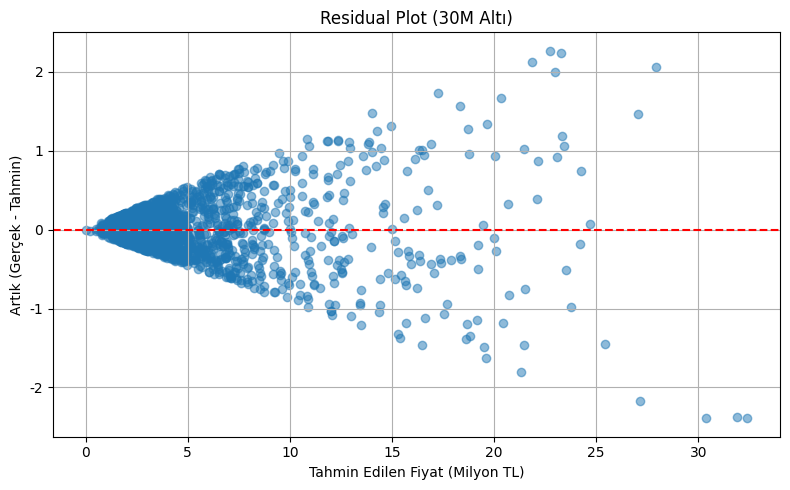


* Model hata sapma performansını QQ Plot grafiği ile gösterimi





* Model tahmin hatalarının sapma paylarının grafiğini Artık Değer Grafiği ile gösterimi



## 

## 

## 

## 

## 

## 

### Genel Değerlendirme ve Sonuç

Bu çalışmada başlangıçta kullanılan ev fiyat tahmin modeli, segment ayrımı ve özellik mühendisliği olmaksızın eğitildiğinde, tahmin başarısı oldukça düşüktü. Özellikle yüksek fiyatlı örneklerde model büyük sapmalar gösteriyor, R² skoru düşüyor ve ortalama hatalar yüksek seyrediyordu.

Modelin doğruluk ve genelleme gücünü artırmak amacıyla aşağıdaki iyileştirme adımları uygulandı:

* **Veri segmentasyonu**: Veri üç segmentte (LOW, MID, HIGH) gruplandı. Her segment için ayrı model eğitildi.Özellikle MID segmentinde yani orta fiyatlı örneklerde yüksek başarı oranı sağlandı. HIGH segment örneklerinde verilerin uçuk değerleri ve örnek veri sayısının az olması sebebiyle bu segment düzeyinde fiyat tahminlerinde hata oranı fazla çıkmasına sebep oldu.
* **Özellik mühendisliği**: Sabit sütunlar elendi, kategorik veriler sayısallaştırıldı (one-hot encoding), log dönüşümü yüksek segmentte kullanıldı,0 değere sahip sütunlar elenerek olası veri manipülasyonları önlenmeye çalışıldı.
* **Aykırı değer temizliği**: Üst %0.5 fiyat dışlandı, 695 gibi veri gürültüsü oluşturan yaş ve bina katı gibi aykırı değerler düzeltildi.

Bu sayede modelin genel başarısı önemli ölçüde arttı.

## **Deniz Denizcan-Ayşegül Albayrak(NAİVE BAYES ALGORİTMASI)**

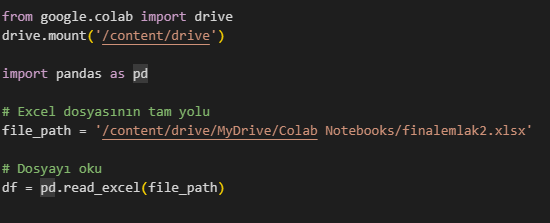
Naive Bayes algoritması, makine öğrenmesinde sıkça kullanılan **olasılıksal bir sınıflandırma yöntemidir**. Bayes teoremi temeline dayanır ve özelliklerin birbirinden bağımsız olduğu varsayımıyla çalışır. Bu projede, emlak veri setimizdeki evlerin çeşitli özelliklerine göre **fiyat aralıklarının sınıflandırılmasını** sağlamak amacıyla Naive Bayes algoritmasını kullandım.

Naive Bayes, özellikle büyük ve çok boyutlu veri setlerinde **hızlı ve etkili sınıflandırma** yapabilmesiyle bilinir. Model, her bir özelliğin sınıfa ait olma olasılığını hesaplayarak en yüksek olasılığa sahip sınıfı tahmin eder. Bu yaklaşım, şu gibi sorulara cevap ararken oldukça faydalıdır:

* "Belli bir metrekareye, oda sayısına ve konuma sahip evlerin hangi fiyat kategorisine ait olma olasılığı yüksektir?"
* "Düşük fiyatlı evler genellikle hangi özelliklere sahiptir?"

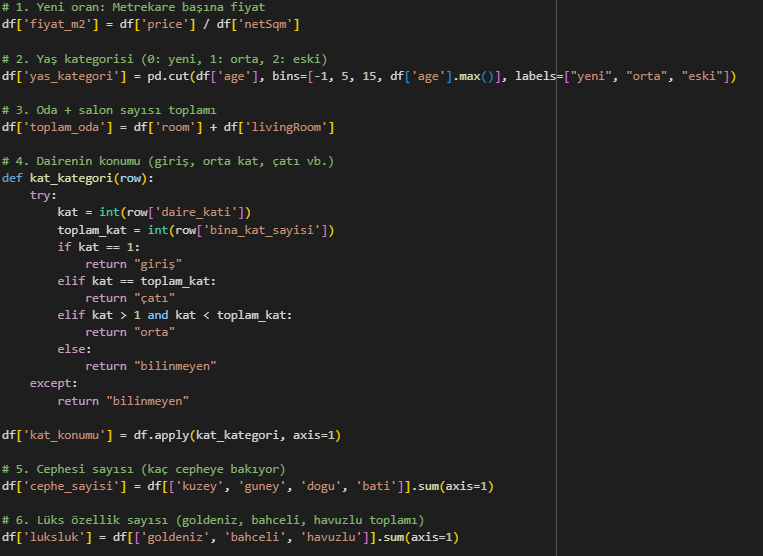
Bu bağlamda, emlak veri setindeki özellikleri kullanarak fiyatları **düşük, orta ve yüksek** gibi kategorilere ayırdım ve modelin bu sınıfları ne kadar başarılı tahmin edebileceğini test ettim.

### **Veri Setinin Yüklenmesi**



Yukarıda gösterilen kodla Google Drive’da yer alan finalemlak2.xlsx adlı Excel dosyasına erişilip üzerinde işlem yapmaya hazır hale getirilmiştir.

### **Feature Engineering Uygulaması**



Veri seti üzerinde yapılan feature engineering (özellik mühendisliği) işlemleriyle, modelin öğrenme kapasitesini artıracak yeni değişkenler türetilmiştir. İlk olarak, konutların birim alan maliyetini yansıtmak amacıyla fiyatın metrekareye oranı alınarak fiyat\_m2 değişkeni oluşturulmuştur. Ardından, bina yaşı age değişkeni kategorilere ayrılarak yas\_kategori sütunu ile yaş grupları (yeni, orta, eski) tanımlanmıştır. Konuttaki toplam yaşam alanını yansıtmak için oda ve salon sayıları toplanarak toplam\_oda değişkeni elde edilmiştir. Dairenin binadaki fiziksel konumu (giriş, çatı, ara kat) kat\_konumu adlı yeni bir kategorik değişken ile sınıflandırılmıştır. Dairenin kaç farklı cepheye baktığını ifade eden cephe\_sayisi sütunu, kuzey, güney, doğu ve batı yönlerine bakan cephelerin sayısal toplamıyla hesaplanmıştır. Son olarak, evin sahip olduğu lüks özelliklerin (deniz manzarası, bahçe, havuz) toplamı alınarak luksluk adlı değişken oluşturulmuş ve bu sayede konutun yaşam konforu ile ilişkili nitelikleri tek bir gösterge altında toplanmıştır. Bu işlemlerle, modelin daha anlamlı ilişkiler öğrenmesine katkı sağlanması hedeflenmiştir.



Bu satırda, sürekli (sayısal) bir değişken olan price (fiyat) sütunu, sınıflandırma problemi için üç kategorili bir değişkene dönüştürülmüştür.

pd.qcut() fonksiyonu, verileriniceliksel olarak (quantiles) üç eşit parçaya böler.

* Yani fiyat dağılımı %33, %33, %33 olacak şekilde üç sınıfa ayrılır.

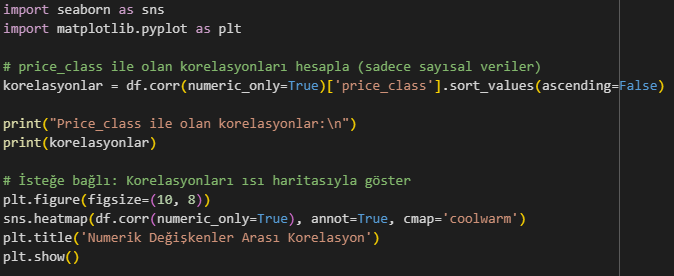
labels=[0, 1, 2] ifadesiyle bu üç sınıfa sırasıyla etiketler atanır:

* **0**: En düşük fiyatlı konutlar
* **1**: Orta seviye fiyatlı konutlar
* **2**: En yüksek fiyatlı konutlar



Önceki adımda pd.qcut(..., labels=[0, 1, 2]) kullanıldığında, etiketler görünüşte sayılar olsa da, bu sütun aslında category (kategorik) veri tipinde olur.  
astype(int) komutu, bu kategorik etiketleri gerçek tam sayılara (örn. int64) dönüştürür.

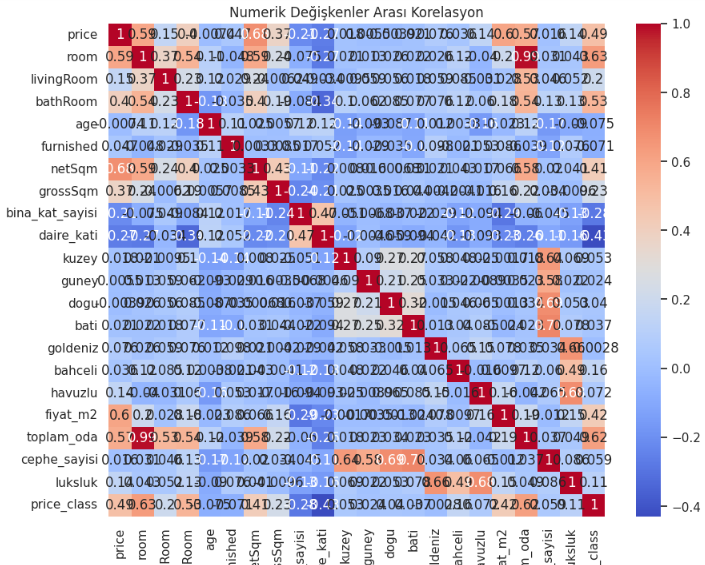
### **Korelasyon Analizi**



Veri setindeki sayısal değişkenler arasındaki ilişkileri incelemek amacıyla korelasyon analizi yapılmıştır. Öncelikle, hedef değişken olan price\_class ile diğer sayısal sütunlar arasındaki korelasyon katsayıları hesaplanarak, bu değişkenlerle olan pozitif ve negatif ilişkiler sıralanmıştır. Böylece, price\_class üzerinde en fazla etkisi olan özellikler tespit edilmiştir. Ardından, veri setindeki tüm sayısal değişkenler arasındaki korelasyonlar, hem değerleri hem de yönleriyle birlikte seaborn kütüphanesi kullanılarak ısı haritası şeklinde görselleştirilmiştir. Bu görsel sayesinde değişkenler arasındaki güçlü pozitif veya negatif ilişkiler kolayca anlaşılmış, veri yapısının genel görünümü ortaya konmuştur. Yapılan bu analiz, modelleme sürecinde hangi değişkenlerin daha belirleyici olduğunu belirlemeye ve veri setindeki çoklu doğrusal bağlantıları fark etmeye olanak sağlamıştır.

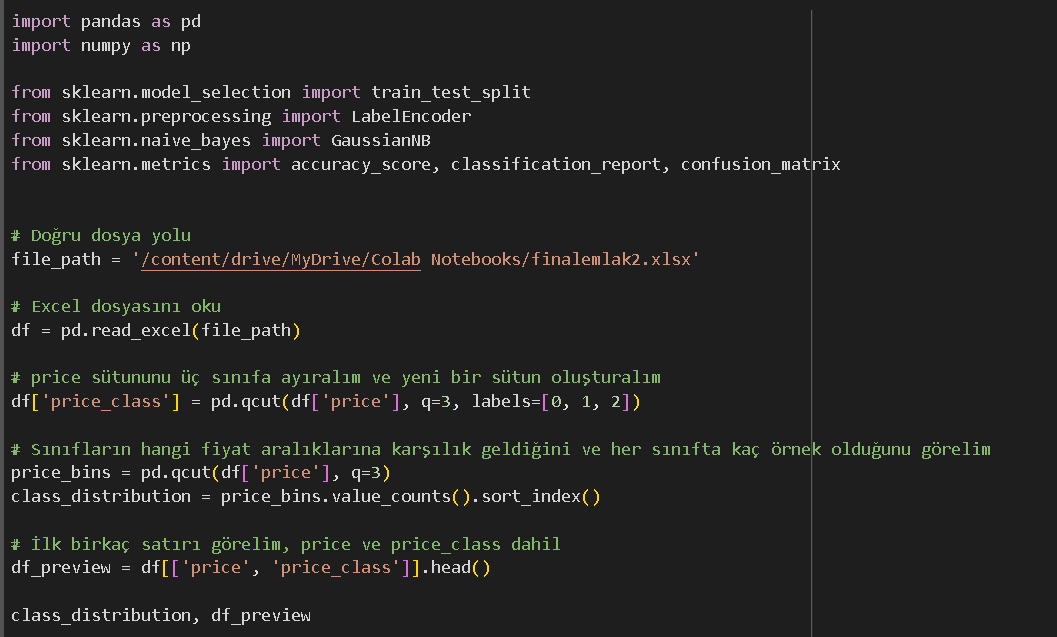
Çıktı:





### 

### **Veri Hazırlama ve Hedef Değişkenin Oluşturulması**



**1.1 Hedef Değişkenin Oluşturulması: Fiyat Sınıflandırması**

Veri seti yüklendikten sonra, bağımlı değişken olan **fiyat (price)** sütunu sınıflandırma problemi haline getirilmiştir. Bu amaçla, fiyatlar üç eşit gruba ayrılarak her bir gruba **0**, **1** ve **2** etiketleri atanmıştır. Bu işlem pd.qcut() fonksiyonu ile gerçekleştirilmiş, böylece her sınıfta yaklaşık olarak eşit sayıda örnek bulunması sağlanmıştır. Bu sınıflar modelin tahminleyeceği hedef değişken olan price\_class sütununda tutulmuştur. Ayrıca, sınıf dağılımı ve örnek veri görünümü elde edilerek veri setinin ilk yapısı gözlemlenmiştir.

Çıktı;

(18999.999, 2350000.0] 732 örnek → Sınıf 0

(2350000.0, 4350000.0] 725 örnek → Sınıf 1

(4350000.0, 138000000.0] 722 örnek → Sınıf 2

**1.2- Kategorik sütunlar belirlenir.**

Veri ön işleme aşamasında, modelleme için kullanılacak kategorik değişkenler otomatik olarak tespit edilmiştir. Bu işlem, veri çerçevesindeki object ve bool veri tipine sahip sütunlar filtrelenerek yapılmıştır. Hedef değişken olan price\_class bu listeye yanlışlıkla dahil edilmişse, elle çıkarılmıştır. Bu adım, ileride yapılacak etiketleme (Label Encoding) işlemi için gereklidir.

Çıktı;

Kategorik sütunlar: ['district', 'county', 'residence', 'heating', 'fuel', 'build', 'buildState', 'furnished', 'subCategory']

**1.3- Label Encoding yapılır.**

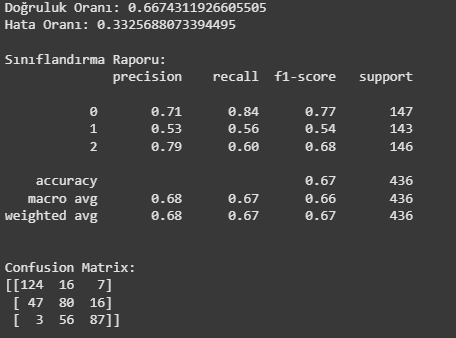


Makine öğrenmesi algoritmaları sayısal girdilerle çalıştığı için, veri setindeki tüm kategorik değişkenler sayısal forma dönüştürülmüştür. Bu amaçla, LabelEncoder kullanılarak her bir kategorik sütun içerisindeki benzersiz kategoriler 0, 1, 2**, ...** gibi tam sayılara çevrilmiştir. Bu işlem modelin kategorik değişkenlerle çalışmasını mümkün kılmıştır.

| **ÖRNEK**  **room\_count (orjinal)** | **room\_count (encode sonrası)** |
| --- | --- |
| 2+1 | 0 |
| 3+1 | 1 |
| 1+1 | 2 |

**1.4 Gaussian Naive Bayes ile Fiyat Sınıflandırma Modeli**

Çıktı;



Veri setindeki özellikler (X) ve hedef değişken (y) ayrıştırılmıştır; burada price\_class hedef değişken olarak kullanılmış, price ve price\_class sütunları modelde kullanılmamak üzere çıkarılmıştır. Veri seti, model eğitimi için %80, test için %20 oranında rastgele bölünmüştür. Sınıflandırma algoritması olarak Gaussian Naive Bayes seçilmiş ve eğitim verisi ile model eğitilmiştir. Daha sonra test verisi kullanılarak tahminler yapılmış, modelin doğruluk oranı hesaplanmış ve ek olarak hata oranı çıkarılmıştır. Model performansını detaylı incelemek için sınıflandırma raporu ve karışıklık matrisi oluşturularak ekrana yazdırılmıştır.

Çıktılar incelendiğinde, modelin test verisi üzerindeki doğruluk oranı %66.7 olarak bulunmuş; düşük fiyat aralığındaki sınıf için yüksek başarı sağlanırken, orta fiyat aralığında modelin karışıklık ve hata oranının daha yüksek olduğu gözlemlenmiştir. Yüksek fiyat sınıfında model dengeli bir performans göstermiştir. Karışıklık matrisi, orta sınıf örneklerin diğer sınıflarla karıştığını ve bunun genel model başarımını etkilediğini ortaya koymuştur.

Bu değerlendirmeler, modelin hangi sınıflarda güçlü, hangi sınıflarda geliştirilmesi gerektiğine dair önemli bilgiler sunmaktadır.

**1.5 GaussianNB ve Random Forest Modellerinin Karşılaştırılması**

Bu aşamada, sadece Naive Bayes değil, aynı zamanda daha güçlü ve yaygın olarak kullanılan **Random Forest** algoritması da modele dahil edilmiştir. Her iki model aynı eğitim ve test verisi ile eğitilmiş ve karşılaştırılmıştır. Aşağıdaki çıktılar sayesinde hangi modelin daha başarılı olduğunu değerlendirebiliriz.

Çıktı;

PERFORMANS ÖZETİ

| **Model** | **Doğruluk (Accuracy)** | **F1 Skoru (Macro Avg.)** |
| --- | --- | --- |
| Gaussian Naive Bayes | 0.667 | 0.665 |
| Random Forest | 0.782 | 0.784 |

Random Forest modeli, hem doğruluk hem de F1 skoru açısından daha yüksek başarı göstermiştir.

SINIF BAZLI F1 SKORLARI

| **Sınıf** | **GaussianNB** | **Random Forest** |
| --- | --- | --- |
| 0 | 0.77 | 0.83 |
| 1 | 0.54 | 0.70 |
| 2 | 0.68 | 0.82 |

Orta fiyat sınıfı (sınıf 1) hem modeller hem de genel olarak en zor sınıflandırılan grup olmuştur.

### **Sonuç**

Random Forest modeli, veri kümesindeki karmaşık ilişkileri daha iyi yakalayarak fiyat sınıflandırmasında daha başarılı olmuştur. Bu nedenle, bu gibi heterojen ve çok boyutlu veri kümelerinde ağaç tabanlı algoritmalar daha güçlü bir seçenek olabilir.

### **2.1-CategoricalNB**

Veri setimizde yer alan bağımsız değişkenlerin büyük çoğunluğu kategorik yapıda olduğundan, klasik Naive Bayes algoritmalarına göre daha uygun olan **Categorical Naive Bayes (CategoricalNB)** sınıflandırma modeli tercih edilmiştir. Bu model, kategorik veriler için sınıf koşullu olasılıkların tahmin edilmesini kolaylaştırır ve genellikle Ordinal veya One-Hot Encoder ile ön işlem görmüş verilerle başarılı sonuçlar verir.

Ön İşlem Aşamaları

* Bağımlı değişken olan price sütunu, pd.qcut() fonksiyonu kullanılarak üç eşit frekanslı sınıfa ayrılmıştır:  
  + low
  + medium
  + high
* Bağımsız değişken olarak kullanılan kategorik sütunlar:  
  + district, county, residence, heating, fuel, build, buildState, furnished, subCategory
* Bu sütunlar OrdinalEncoder ile sayısal forma dönüştürülmüştür.
* Veriler, %80 eğitim ve %20 test olacak şekilde ayrılmıştır.

Model Eğitimi ve Tahmin

Model aşağıdaki gibi eğitilmiş ve test verisi üzerinde tahmin yapılmıştır.

model = CategoricalNB()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

Model Performansı

Aşağıda, modelin test verisi üzerindeki başarı oranı ve sınıflandırma raporu yer almaktadır.

**Accuracy (Doğruluk Oranı)**: 0.674

**Error Rate(Hata Oranı):**0.325



Değerlendirme

* Model, genel doğruluk açısından %67.4 başarı sağlamıştır.
* low sınıfında en yüksek **recall** değeri (%86) elde edilmiştir.
* medium sınıfı için modelin performansı düşüktür (f1-score: 0.44); bu durum modelin bu sınıfı ayırmakta zorlandığını göstermektedir.
* high ve low sınıflarında dengeli bir başarı gözlenmiştir.
* **Random Forest**, doğruluk ve f1-score açısından en iyi sonuçları vermiştir.
* **CategoricalNB**, kategorik verilerle çalışmasına rağmen medium sınıfında zorlanmıştır.
* **GaussianNB**, sürekli veriler için daha uygun olup bu veri setinde en düşük başarıyı göstermiştir.

**3.1 KBinsDiscretizer + OrdinalEncoder + CategoricalNB**

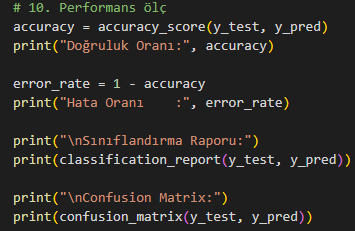
3.1.1 Veri Hazırlama Adımları

**Hedef değişken**: price sütunu qcut fonksiyonu ile 3 eşit parçaya bölünerek price\_class (0: düşük, 1: orta, 2: yüksek) sınıf etiketi oluşturulmuştur.

**Sayısal sütunlar**: KBinsDiscretizer kullanılarak 5 eşit frekanslı gruba ayrılmıştır. Böylece Naive Bayes için uygun hale getirilmiştir.

**Kategorik sütunlar**: OrdinalEncoder ile sayısal formata çevrilmiştir.

**Performans Değerlendirmesi**



Başarı Sonuçları:

* **Doğruluk Oranı**: 0.729 (yaklaşık %73)
* **Hata Oranı:0.270**
* Model, özellikle **düşük fiyatlı (0)** ve **yüksek fiyatlı (2)** konutları iyi sınıflandırmıştır.
* **Orta fiyatlı konutlarda (1)** başarı düşüktür. Çünkü bu sınıf, hem yukarı hem aşağı gruba benzer özellikler taşıyabilir (sınıf ayrımı daha belirsiz olabilir).

Detaylı Yorumlar:

* **Sınıf 0 (düşük fiyat)**: Precision %75, recall %85 → Model bu sınıfı oldukça iyi tanıyor.
* **Sınıf 1 (orta fiyat)**: Precision %61, recall %52 → Belirsiz bir grup olduğu için karıştırılıyor.
* **Sınıf 2 (yüksek fiyat)**: Precision %80, recall %81 → Yüksek başarı.

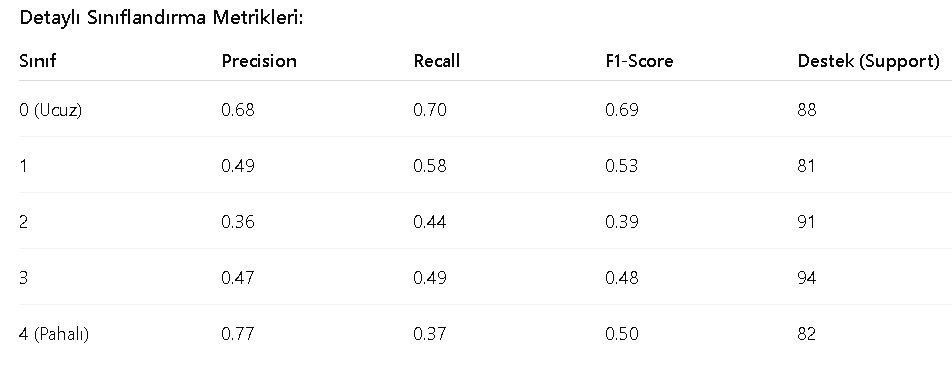
### **Karışıklık Matrisi:**

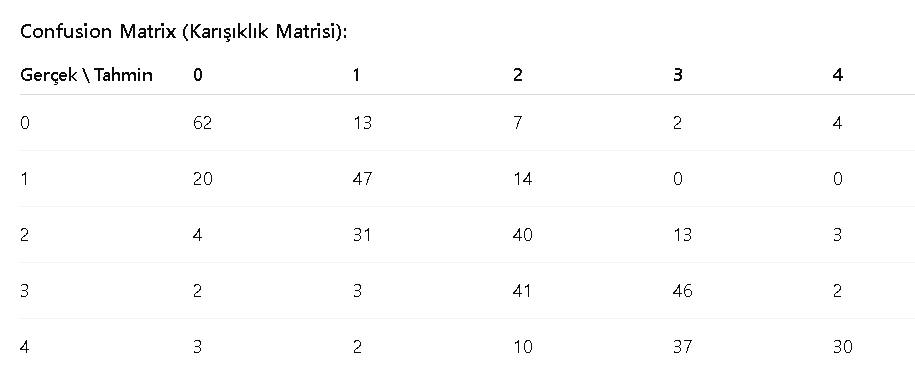
* **Sınıf 1**, sıklıkla sınıf 0 ve 2 ile karıştırılmış (özellikle sınıf 2 ile).
* **Sınıf 0 ve 2**, genellikle doğru sınıflandırılmıştır.

### **3.2 Log Ölçekli Sınıflarla Naive Bayes Modeli**

**Veri ve Model Bilgisi:**

* Veri setindeki fiyatlar logaritmik (ln) ölçeğe dönüştürülmüş ve 5 sınıfa bölünmüştür.Fiyat kategorisi (0: en ucuz, 4: en pahalı)
* Kategorik değişkenler sayısal forma çevrilmiş (Label Encoding).
* Özellikler arasında hem kategorik hem sayısal değişkenler yer almaktadır.
* Model olarak Gaussian Naive Bayes kullanılmıştır.





Model özellikle en ucuz (0) ve en pahalı (4) kategorilerinde karışıklık yaşamakta, 4. sınıfın recall’u düşük (0.37), yani en pahalı kategorideki örneklerin büyük bir kısmı doğru tahmin edilemiyor.

Orta sınıflar (1, 2, 3) için genel performans düşüyor; bu sınıflar arasında çok karışıklık var.

**Değerlendirme:**

* Modelin genel doğruluğu %52 civarında, bu da sınıflandırma probleminin zor olduğunu gösteriyor.
* Fiyatı 5 sınıfa bölmek, özellikle orta sınıflarda karışıklığı artırmış olabilir.
* GaussianNB modelinin varsayımları (özelliklerin normal dağıldığı) veri için tam karşılanmıyor olabilir.

**3.3 -Naive Bayes ve Decision Tree Karşılaştırma**

Veri Hazırlık Süreci

Sayısal değişkenlerde 0 ve altı değerler NaN olarak işaretlenip sütun medyanıyla dolduruldu.

Hedef değişken olan price, logaritmik ölçekte (log1p) dönüştürüldü ve qcut ile 3 sınıfa ayrılarak price\_class oluşturuldu:

* 0 = Ucuz, 1 = Orta, 2 = Pahalı

Kategorik değişkenler OrdinalEncoder ile sayısallaştırıldı.

Sayısal değişkenler Naive Bayes için KBinsDiscretizer ile kategorik yapılara dönüştürüldü.

### **Naive Bayes Sonuçları (CategoricalNB)**

**Kullanılan Yöntem**

* Sayısal sütunlar, farklı **n\_bins** (3, 5, 7) değerleriyle kategorik yapılara dönüştürülmüştür.
* Ardından, bu kategorik veriler kullanılarak **Categorical Naive Bayes (CategoricalNB)** modeli eğitilmiştir.
* Model parametrelerinden **alpha** için 0.01, 0.1, 0.5, 1.0 ve 2.0 değerleri denenmiştir.

**Genel Gözlemler**

#### 

| n\_bins | Alpha | Doğruluk (Accuracy) |
| --- | --- | --- |
| 3 | 0.01–0.5 | 0.7385 (En yüksek) |
| 5 | — | 0.7225 |
| 7 | 0.5–1.0 | 0.7110 (En düşük) |

**En yüksek doğruluk**: n\_bins=3, alpha=0.01–0.5 aralığında elde edilmiştir (yaklaşık %73.85).

**En düşük doğruluk**: n\_bins=7, alpha=0.5 ve 1.0 değerlerinde gözlemlenmiştir (yaklaşık %71.10).

En iyi yapılandırma (n\_bins=3, alpha=0.01–0.5) için:

* Doğruluk: %73.85
* **Class 2** için F1-score: ~0.83 (yüksek başarı)
* **Class 1** için F1-score: ~0.61 (daha düşük, zor sınıf)

Karışıklık matrisinde **Class 2** çoğunlukla doğru tahmin edilmiştir.  
**Class 1** için model performansı zayıftır; precision ve recall değerleri 0.58–0.63 aralığındadır.  
Bu durum, **Class 1’in diğer sınıflarla karıştığını** ve modelin bu sınıfı ayırt etmekte zorlandığını gösterir.

**n\_bins arttıkça doğruluk oranı azalmaktadır:**

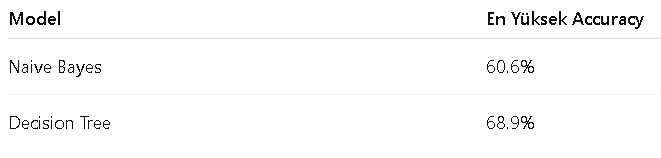
* n\_bins=3 → %73.85
* n\_bins=5 → %72.25
* n\_bins=7 → %71.10

Sebebi, yüksek bin sayısının veriyi fazla parçalayıp bazı bilgileri bozmasıdır.  
Özellikle Gaussian olmayan Naive Bayes modellerinde çok fazla ayrık kategori oluşturmak performansı düşürebilir.

### **Decision Tree Sonuçları**

**Kullanılan Yöntem:**

* Sayısallaştırılmış orijinal veri (binleme yapılmamış) ile
* max\_depth=5 kullanılarak Decision Tree modeli eğitildi.



**Naive Bayes**, özellikle kategorik verilerle hızlı ve etkili çalışsa da, karmaşık ilişkileri modellemede sınırlı kalmıştır.

**Decision Tree**, daha yüksek başarı oranı ile bu veri setinde **daha uygun bir model** olarak öne çıkmıştır.

Ancak Naive Bayes, daha düşük işlem maliyeti ve basitliğiyle avantajlı olabilir.

**3.4 Naive Bayes ve Decision Tree Modellerinin Doğruluk (Accuracy) Karşılaştırması**

**1- Fiyatı Sınıflandırma (Binning)**

Sürekli (sayısal) olan price değişkenini kategorilere ayırdık.(n\_bins parametresi).

Bu sayede fiyat,

* 0: Düşük fiyat
* 1: Orta fiyat
* 2: Yüksek fiyat  
   şeklinde sınıflandırıldı.

#### **2. Kullanılan Algoritmalar**

* **Naive Bayes**

alpha (smoothing parametresi) değiştirildi.

n\_bins (fiyat kategorisi sayısı) farklı şekillerde denendi.  
**Karşılaştırma için:** Decision Tree gibi başka modeller de kullanılmış.

#### **3. Değerlendirme**

Başarı metrikleri: Accuracy (doğruluk) baz alındı.

Sonuçlar results\_df isimli bir veri çerçevesinde tutulup grafikle görselleştiridi.

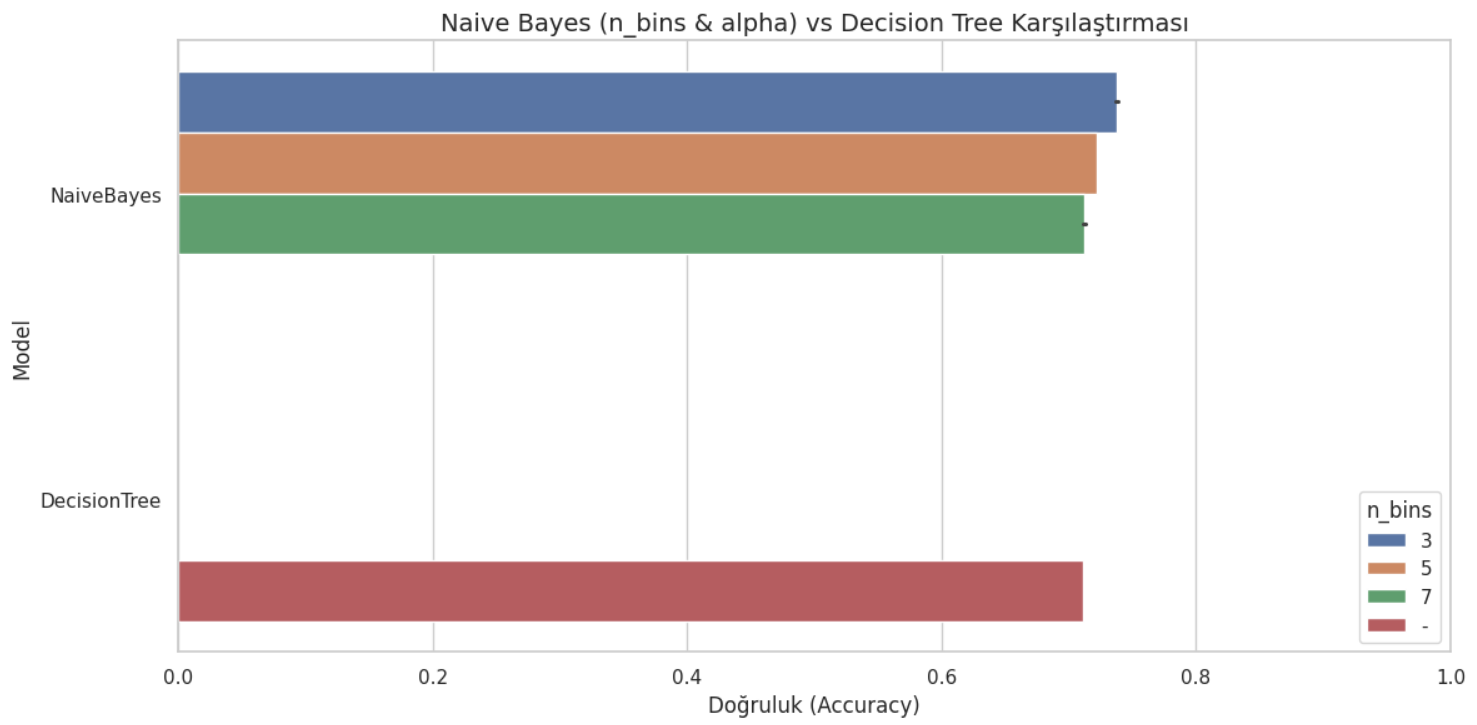
Modelleme Yaklaşımı

Bu çalışmada, ev fiyatlarının doğrudan sayısal değer olarak tahmini yerine, fiyat aralıklarına göre sınıflandırılması amaçlanmıştır. Sürekli olan price değişkeni, **eşit frekanslı sınıflara** (n\_bins) ayrılarak **kategorik bir hedef değişken** (price\_cat) elde edilmiştir.

Sınıflandırma işlemi için **Naive Bayes algoritması** uygulanmıştır. Modelin performansı, farklı alpha (Laplace smoothing) ve n\_bins değerleri denenerek değerlendirilmiştir. Ayrıca, kıyaslama amacıyla **Decision Tree sınıflandırma algoritması** da kullanılmıştır

Model Performansları

Model doğruluk (accuracy) skorları görselleştirilerek karşılaştırılmıştır. Aşağıdaki grafikte, farklı n\_bins değerlerinde elde edilen başarılar gözlemlenmektedir. Naive Bayes'in sade yapısı nedeniyle bazı durumlarda Decision Tree'ye kıyasla daha düşük doğruluk verdiği görülse de, uygun parametrelerle rekabetçi sonuçlar elde edilmiştir.



**1. Grafik Yorumlaması**

Yukarıdaki grafikte, **Naive Bayes** algoritması farklı n\_bins (3, 5, 7) ve alpha parametreleri ile test edilmiştir. Ayrıca karşılaştırma amacıyla bir **Decision Tree** modeli de dahil edilmiştir.

* **Naive Bayes** modelleri benzer doğruluk (accuracy) oranlarına sahiptir ve %70 civarında performans göstermektedir.
* **Decision Tree** algoritması tek bir doğruluk değeri ile (n\_bins parametresi olmadan) karşılaştırmaya dahil edilmiştir ve bu model de yaklaşık olarak benzer başarıda sonuç vermiştir.
* n\_bins=3 ile çalışan Naive Bayes modeli, diğer iki Naive Bayes varyantına göre çok az farkla en iyi doğruluğu vermiştir.
* Grafik üzerinde görülen küçük hata çubukları (error bar), doğruluk skorlarının güven aralıklarını yansıtmaktadır. Bu farkların istatistiksel olarak anlamlı olmayabileceğini de düşündürmektedir.

2. Sonuç

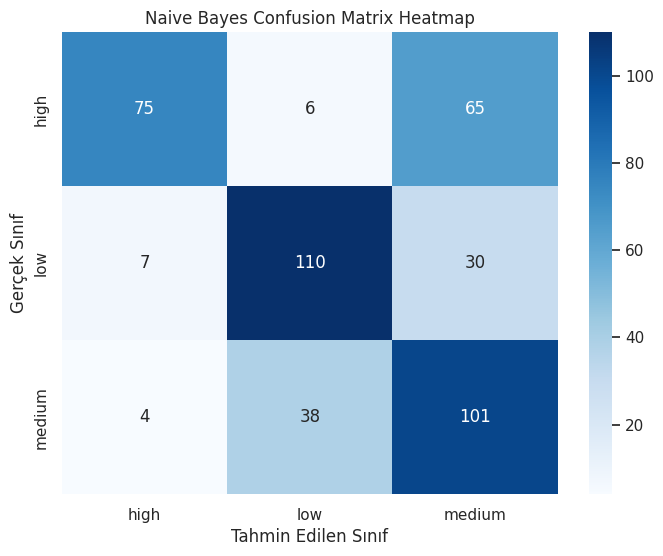
Bu analiz sonucunda, **Naive Bayes algoritması farklı parametrelerle tutarlı bir doğruluk sergilemiş**, bu da modelin sağlamlığını göstermektedir. Ancak Decision Tree modeli de çok yakın bir performans sergilediği için **bu veri setinde hangi modelin tercih edileceği uygulama hedeflerine ve yorumlanabilirliğe bağlı olabilir.**

**3.5- Naive Bayes Modeli ile Fiyat Sınıflandırması ve Confusion Matrix Analizi**

Bu bölümde, elimizdeki emlak veri setinde yer alan fiyat değişkeni, dağılıma göre üç sınıfa (low, medium, high) ayrılarak sınıflandırma problemi haline getirilmiştir. Ardından, bu sınıfları tahmin etmek amacıyla Gaussian Naive Bayes algoritması kullanılmıştır.

Veri ön işleme adımlarında sayısal sütunlar seçilmiş ve hedef değişken olarak price\_class tanımlanmıştır. Eğitim ve test verileri ayrıldıktan sonra model eğitilmiş ve test verileri üzerinde tahminler yapılmıştır.

Modelin performansını değerlendirmek için confusion matrix (karmaşıklık matrisi) kullanılmıştır. Matris, gerçek sınıflar ile modelin tahmin ettiği sınıfların ne derece örtüştüğünü görsel olarak sunmaktadır. Bu sayede, modelin hangi sınıflarda başarılı olduğu, hangi sınıflarda hata yaptığı kolayca analiz edilebilmektedir.

****

Model Performansı Üzerine Yorumlar

Yukarıdaki confusion matrix, Naive Bayes sınıflandırıcısının low, medium ve high fiyat sınıflarını tahmin etme başarısını göstermektedir.

* Low sınıfı için model oldukça başarılıdır. Gerçekten "low" olan 147 örnekten 110'u doğru sınıflandırılmıştır.
* Medium sınıfında model 101 doğru tahmin yaparken, 38 örneği "low" ve 4 örneği "high" sınıfına yanlış tahmin etmiştir.
* High sınıfı, modelin en çok hata yaptığı sınıftır. 146 gerçek "high" örnekten yalnızca 75’i doğru tahmin edilmiştir. Model bu sınıftaki birçok örneği yanlışlıkla "medium" olarak etiketlemiştir (65 örnek).

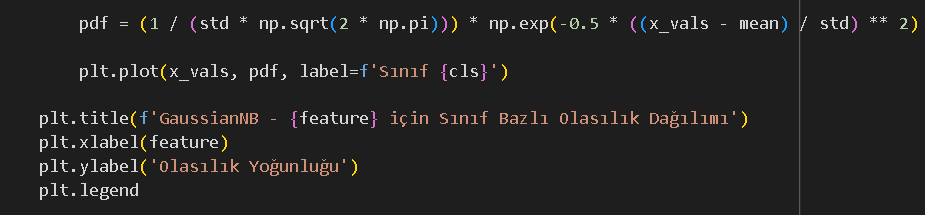
Bu durum, modelin özellikle "high" ve "medium" sınıfları arasında ayrım yapmakta zorlandığını göstermektedir. Bu karışıklık, bu iki sınıfın bazı ortak özelliklere sahip olmasından veya Gaussian Naive Bayes modelinin varsayımlarının bu veriye tam olarak uymamasından kaynaklanıyor olabilir.

**3.6- Modelin Gaussian Dağılımına Göre Özellik Bazlı Karar Verme Yapısını İnceleme**

Gaussian dağılımlarının hesaplanması:

Her sınıf için ilgili özelliğin ortalaması ve varyansı kullanılarak olasılık yoğunluğu şu şekilde hesaplanmıştır:

Bu satır, **normal dağılım fonksiyonunun** matematiksel formülünün kod karşılığıdır ve modelin temelini oluşturur.

****

Bu bölümde, Naive Bayes algoritmasının temelini oluşturan özelliklerin her sınıf için nasıl dağıldığı görselleştirilmiştir. GaussianNB modeli, her bir sayısal özelliğin (örneğin oda sayısı, bina yaşı vb.) her sınıf (low, medium, high) için normal (gauss) dağılıma uyduğunu varsayarak çalışır.

Modelin eğitimi tamamlandıktan sonra, elimizdeki sayısal özellikler kullanılarak her sınıf için aşağıdaki adımlarla olasılık yoğunluk fonksiyonları çizilmiştir:

Her bir özellik için:

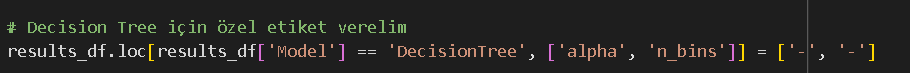
* + İlgili sınıfa ait verilerin ortalaması (mean) ve varyansı (variance) elde edilmiştir.
  + Bu istatistikler kullanılarak olasılık yoğunluk fonksiyonları hesaplanmış ve çizilmiştir.
  + Böylece, modelin hangi değer aralıklarında hangi sınıfa ait olma ihtimalini daha yüksek gördüğü ortaya konmuştur.

Yorumlanabilecek Noktalar:

* Eğer bir özelliğin dağılımları (low, medium, high) birbirinden net ayrılıyorsa, bu özellik sınıflandırmada etkili bir rol oynuyordur.
* Eğer dağılımlar birbirine çok yakınsa veya üst üste biniyorsa, model bu özelliği kullanarak sınıfları ayırt etmekte zorlanabilir.
* Bu grafikler, hangi değişkenlerin sınıflandırma açısından daha bilgilendirici olduğunu anlamamıza olanak sağlar.

Bu görselleştirme sayesinde, Naive Bayes modelinin özellikler üzerinden nasıl tahminde bulunduğu anlaşılmış ve modelin içsel karar mekanizması hakkında sezgisel bir fikir elde edilmiştir. Bu analiz, hem modelin performansını hem de yorumlanabilirliğini artırmak için oldukça değerlidir.

**3.7-Naive Bayes Modeli: Alpha ve n\_bins Parametrelerine Göre Performans Analizi**



Bu bölümde, Naive Bayes algoritmasının alpha (Laplace smoothing) ve n\_bins (özelliklerin kaç eşit aralığa bölündüğü) parametrelerinin model doğruluğu üzerindeki etkisi detaylı olarak analiz edilmiştir.

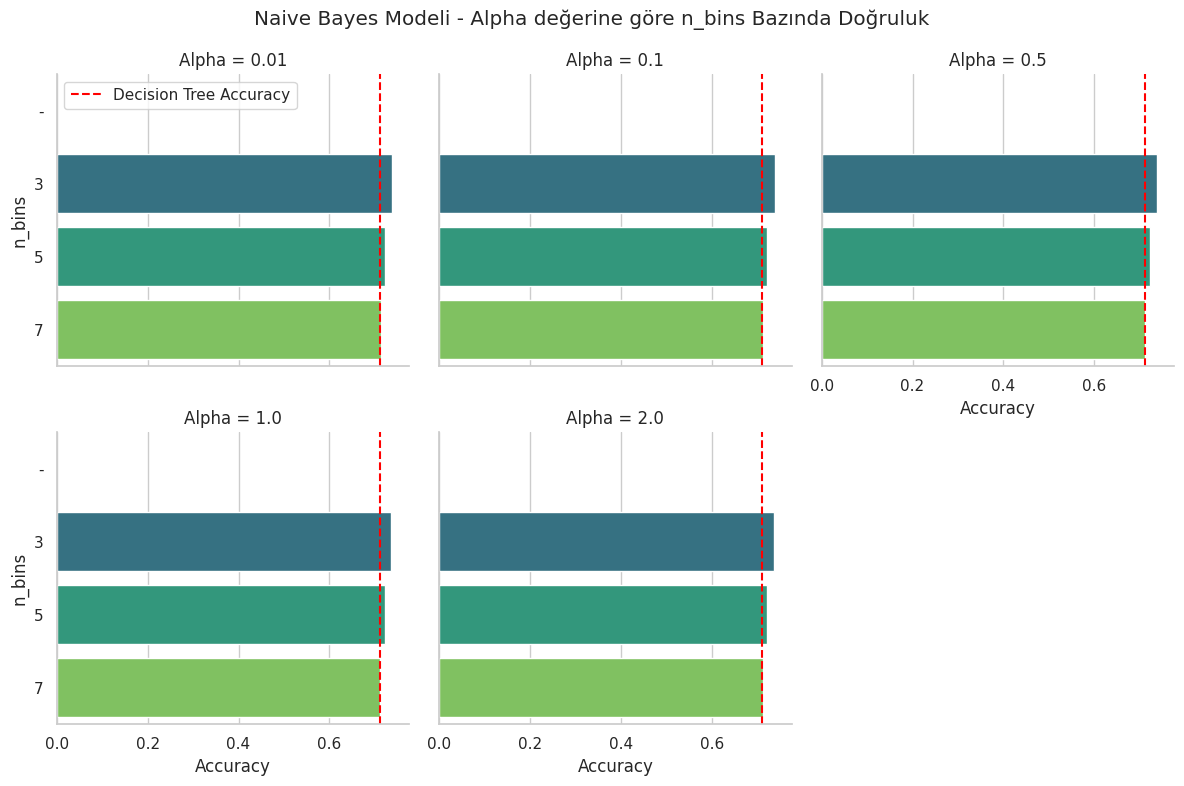
Her bir alpha değeri için, farklı n\_bins değerlerinde elde edilen doğruluklar ayrı grafikler halinde sunulmuştur. Bu sayede parametre kombinasyonlarının model performansına etkisi açık şekilde görülebilmektedir.

Grafiklerde referans noktası olarak, **Decision Tree modelinin doğruluk değeri** kırmızı kesikli çizgiyle gösterilmiştir. Böylece, Naive Bayes'in belirli parametre değerlerinde Decision Tree modelinden daha iyi performans gösterip göstermediği kolaylıkla gözlemlenebilmektedir.

Gözlemler:

* Belirli alpha değerlerinde, n\_bins değiştikçe doğrulukta belirgin dalgalanmalar gözlemlenmiştir.
* Bazı durumlarda Naive Bayes modeli, Decision Tree'yi geçememiştir. Ancak bazı alpha-n\_bins kombinasyonlarında daha yüksek doğruluk değerleri elde edilmiştir.
* En başarılı performansın elde edildiği alpha ve n\_bins kombinasyonları, modelin parametre optimizasyonu açısından yol gösterici olmuştur.

Bu analiz sonucunda, Naive Bayes modelinin hiperparametrelerine dikkat edildiğinde, özellikle basit veri setlerinde güçlü bir alternatif olabileceği anlaşılmıştır.



Grafik Yorumlaması

Grafikte, **Naive Bayes modeli** için farklı alpha değerlerinde ve çeşitli n\_bins (sınıf sayısı) ayarlarında elde edilen doğruluk (accuracy) değerleri gösterilmektedir. Her bir alt grafik, farklı bir alpha değerini temsil etmekte; yatay eksen doğruluk oranlarını göstermektedir. Ayrıca, her grafikte **kırmızı kesikli çizgi** ile **Decision Tree (Karar Ağacı)** modelinin sabit doğruluk değeri referans olarak belirtilmiştir.

Detaylı Yorumlar:

* Tüm alpha değerleri için Naive Bayes modeli, n\_bins değişse bile genellikle **Karar Ağacı modeline çok yakın veya onu aşan doğruluklar** göstermiştir.
* Özellikle alpha = 0.5 ve alpha = 2.0 için **n\_bins = 3** durumunda Naive Bayes modelinin doğruluğu, Decision Tree modelini net şekilde geçmiştir.
* alpha = 1.0 ve alpha = 2.0 alt grafiklerinde doğruluk oranları genel olarak daha stabil görünmektedir.
* **n\_bins değeri arttıkça** (örneğin 3'ten 7'ye), doğrulukta çok büyük oynamalar gözlemlenmemektedir. Bu durum, Naive Bayes modelinin farklı histogram bölme stratejilerine karşı **daha esnek** olduğunu göstermektedir.
* Karar Ağacı modeli için tek bir doğruluk değeri mevcut olduğundan, bu model **sabit bir referans noktası** olarak değerlendirilmiştir.

Yapılan karşılaştırmada, Naive Bayes modelinin farklı alpha ve n\_bins kombinasyonlarına rağmen oldukça tutarlı sonuçlar verdiği, bazı durumlarda Decision Tree modelinden daha yüksek doğruluk sağladığı gözlemlenmiştir. Bu durum, Naive Bayes'in parametrik doğasının ve esnekliğinin, sınırlı örnekleme sahip histogram tabanlı senaryolarda avantaj sağlayabileceğini göstermektedir.

**3.8-Naive Bayes ile Çok Sınıflı Sınıflandırma ve GridSearch ile Hiperparametre Optimizasyonu**

1. Özelliklerin Türlerine Göre Ayrılması

* Modelde hem sayısal hem de kategorik değişkenler bulunmaktadır.
* Sayısal değişkenler int64 ve float64 tipinde, kategorik değişkenler ise object ve category tipindedir.
* Bu ayrım, ön işleme sürecinde farklı dönüşümler uygulamak için gereklidir.

2. Veri Ön İşleme ve Dönüşümler

Sayısal özellikler, modelin doğrudan kullanabileceği biçime getirilebilmesi için KBinsDiscretizer ile kategorik hale dönüştürülmüştür.

strategy='uniform' parametresiyle sayısal değerler eşit aralıklara bölünür (örneğin, 3, 5 veya 7 bin).

encode='onehot-dense' seçeneğiyle bu bini kategorik one-hot vektörlerine dönüştürülür.

Kategorik özellikler ise OneHotEncoder kullanılarak binary sütunlara çevrilmiştir.

Bilinmeyen kategoriler için handle\_unknown='ignore' ayarı sayesinde eğitimde görülmeyen kategoriler hata oluşturmaz.

Bu dönüşümler ColumnTransformer ile sayısal ve kategorik sütunlara ayrı ayrı uygulanır.

3. Model Pipeline Kurulumu

* Ön işleme ve sınıflandırıcı adımları Pipeline ile ardışık olarak bağlanmıştır.
* Böylece veri, önce ön işleme tabi tutulur, ardından CategoricalNB sınıflandırıcısına gönderilir.
* Bu yapı, kodun okunabilirliğini artırır ve GridSearch gibi otomatik parametre ayarlama işlemlerine olanak sağlar.

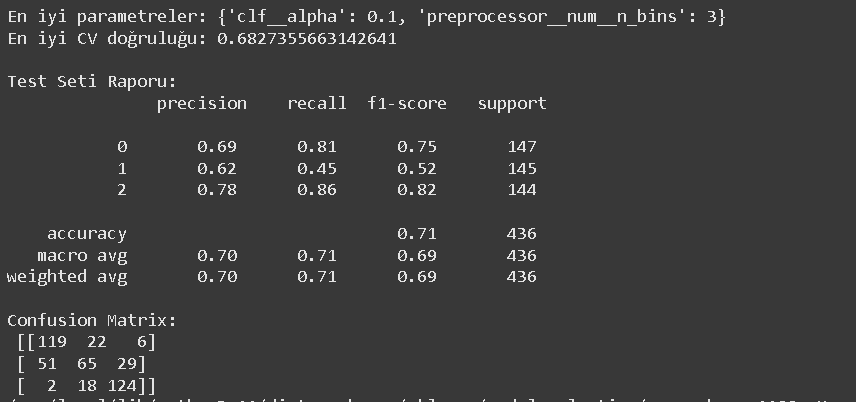
5. Hiperparametre Optimizasyonu

* Model performansını artırmak amacıyla GridSearchCV ile iki temel hiperparametre optimize edilmiştir:  
  + preprocessor\_\_num\_\_n\_bins: Sayısal özelliklerin kaç parçaya bölüneceği (3, 5, 7).
  + clf\_\_alpha: Naive Bayes modelinde Laplace düzeltme (smoothing) katsayısı ([0.01, 0.1, 0.5, 1.0, 2.0]).
* 5 katlı çapraz doğrulama (cross-validation) kullanılarak her parametre kombinasyonu test edilmiş ve doğruluk (accuracy) metriği ile değerlendirilmiştir.
* Paralel işlem (n\_jobs=-1) ile hesaplama süresi kısaltılmıştır.

6. Model Eğitimi ve Değerlendirme

* Veri seti %80 eğitim, %20 test olarak bölünmüştür ve sınıf dağılımını korumak için stratify=y parametresi kullanılmıştır.
* En iyi parametre kombinasyonu GridSearchCV tarafından belirlenmiş ve model bu parametrelerle eğitilmiştir.
* Test seti üzerinde tahminler yapılarak aşağıdaki metrikler raporlanmıştır:  
  + **Classification Report:** Her sınıf için doğruluk (accuracy), precision, recall ve F1-score değerleri.
  + **Confusion Matrix:** Gerçek ve tahmin edilen sınıfların dağılımı. Bu matris, hangi sınıflarda modelin daha başarılı ya da başarısız olduğunu gösterir.

KOD ÇIKTISI



### **Modelin Değerlendirilmesi**

En İyi Parametreler

GridSearchCV kullanılarak yapılan hiperparametre araması sonucunda en iyi performans aşağıdaki değerlerle elde edilmiştir:

* alpha: 0.1 (Naive Bayes modelinin düzgün çalışması için kullanılan parametre)
* n\_bins: 3 (Sayısal değişkenlerin 3 eşit aralığa bölünmesi)

Bu parametrelerle modelin eğitim verisi üzerindeki **en iyi doğruluk oranı (CV accuracy)**: **%68.27**

Test Seti Performansı

Model, test verisi üzerinde %71 doğruluk elde etmiştir. Sınıflara göre modelin performansı:

| **Sınıf** | **Açıklama** | **Doğruluk (F1-score)** |
| --- | --- | --- |
| 0 | Düşük fiyat | %75 |
| 1 | Orta fiyat | %52 |
| 2 | Yüksek fiyat | %82 |

Model, **yüksek ve düşük fiyatlı evleri** doğru tahmin etmede başarılıdır.

**Orta fiyatlı evlerde** hata oranı daha yüksektir. Bu da modelin bu sınıfı diğerleriyle karıştırdığını gösterir.

Karışıklık Matrisi

|  | **Tahmin 0** | **Tahmin 1** | **Tahmin 2** |
| --- | --- | --- | --- |
| Gerçek 0 (düşük) | 119 | 22 | 6 |
| Gerçek 1 (orta) | 51 | 65 | 29 |
| Gerçek 2 (yüksek) | 2 | 18 | 124 |

Orta sınıfa ait birçok veri diğer sınıflara kaymıştır.

Düşük ve yüksek fiyat sınıflarında model daha kararlıdır.

Sonuç

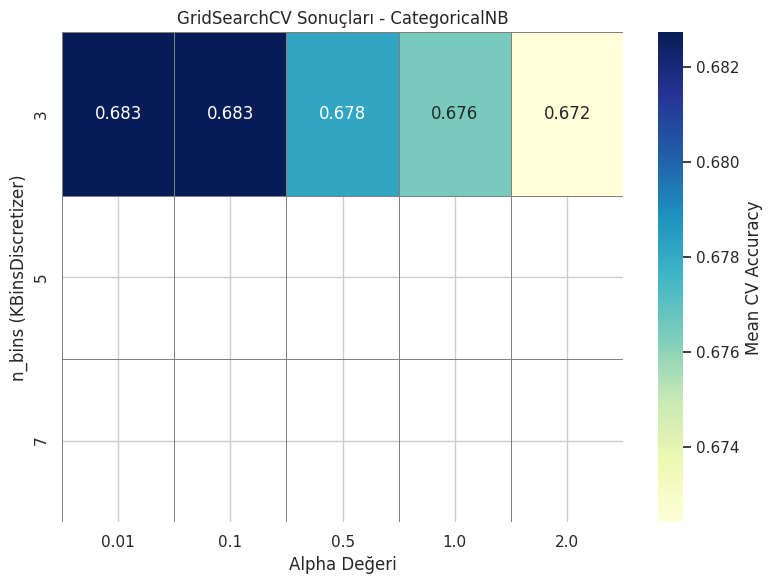
Model genel olarak iyi sonuçlar vermiştir. Ancak özellikle **orta fiyatlı evleri ayırt etmede zorlanmaktadır.**

### **3.9 -Hiperparametre Optimizasyonunun Görselleştirilmesi**

Modelin performansını artırmak amacıyla, GridSearchCV kullanılarak farklı hiperparametre kombinasyonlarının doğruluk skorları hesaplanmıştır. Bu kod bloğu, yapılan bu arama sonuçlarını analiz etmek ve yorumlamak için kullanılmıştır.

**Yapılan işlemler:**

1. **GridSearch Sonuçlarının Alınması:** grid\_search.cv\_results\_ ile her hiperparametre kombinasyonu için çapraz doğrulama (cross-validation) doğruluk skorları elde edilmiştir.
2. **Sonuçların Tablo Haline Getirilmesi:** Elde edilen sonuçlardan, sayısal özellikler için kullanılan n\_bins (KBinsDiscretizer’daki aralık sayısı) ve Naive Bayes modelindeki alpha parametreleri ile bu kombinasyonların ortalama doğruluk skorları (mean\_test\_score) ayrı bir DataFrame’e aktarılmıştır.
3. **Pivot Tablo Oluşturulması:** Bu DataFrame, her n\_bins değeri satır, her alpha değeri sütun olacak şekilde düzenlenmiştir. Böylece farklı parametre kombinasyonlarındaki doğruluk sonuçları kolayca karşılaştırılabilir.
4. **Isı Haritası (Heatmap) ile Görselleştirme:** seaborn kütüphanesi kullanılarak oluşturulan ısı haritası, farklı n\_bins ve alpha parametrelerinin model doğruluğuna etkisini renk yoğunluğu ve sayısal değerler ile görsel olarak ortaya koyar.  
   * Renkler doğruluk oranını gösterir, koyu mavi daha düşük, açık yeşil daha yüksek doğruluğa karşılık gelir.
   * Böylece en iyi parametre kombinasyonları kolayca tespit edilir.



Haritaya göre:

* En yüksek doğruluk skoru **%68.3 (0.683)** ile n\_bins=3 ve alpha=0.01 ile alpha=0.1 kombinasyonlarında elde edilmiştir.
* n\_bins değeri arttıkça (5 ve 7 olduğunda), model doğruluğu hesaplanamamış olabilir veya veriler GridSearchCV’ye ulaşmamış olabilir (beyaz alanlar bu yüzden boş görünüyor).
* alpha değeri arttıkça modelin doğruluğu azalmıştır. Bu da çok büyük alpha değerlerinin aşırı düzeltme yaparak modele zarar verdiğini gösterir.

Sonuç olarak, **n\_bins=3 ve alpha=0.01 ya da 0.1** kombinasyonları model için en uygun hiperparametreler olarak öne çıkmaktadır.

-BAHADIR-

## 📊 Karar Ağacı ile Emlak Fiyat Tahmini Raporu

### 🔍 1. Proje Amacı

Bu proje, emlak verileri üzerinde karar ağacı regresyon modeli kullanılarak ev fiyatlarını tahmin etmeyi amaçlamaktadır. Çalışmada veri ön işleme, özellik mühendisliği, modelleme ve model iyileştirme adımları gerçekleştirilmiştir.

### 🧹 2. Veri Temizleme ve Özellik Mühendisliği

* Veriler finalemlak2.xlsx dosyasından okunmuştur.
* Bazı kategoriler ("İmarlı - Konut", "Konut+Ticaret") "Tarla" etiketiyle birleştirilmiştir.
* Net m²’ye göre birim fiyat (netSqm\_div\_price) hesaplanmış, aykırı değerler (1000-60000 arası) filtrelenmiştir.
* Yeni bir özellik olarak kat\_orani = daire\_kati / bina\_kat\_sayisi eklenmiştir.
* Kategorik sütunlar LabelEncoder ile sayısal hale getirilmiştir.

### 🤖 3. Modelleme Süreci

#### 📌 Kullanılan Model:

* DecisionTreeRegressor
* En iyi model parametreleri:  
  + max\_depth=7,
  + min\_samples\_split=15,
  + min\_samples\_leaf=1

#### 💡 Model Performansı:

| Metodoloji | MAE | RMSE | R² Skoru | Ortalama Hata Oranı |
| --- | --- | --- | --- | --- |
| İlk Model | 804,035 TL | 1,602,084 TL | 0.6990 | - |
| Aykırı Değer Temizliği | 847,583 TL | 1,440,816 TL | 0.7248 | 21.56% |
| GridSearchCV ile Optimizasyon | 809,777 TL | 1,413,492 TL | 0.7352 | 21.24% |
| Önemsiz Özelliklerin Atılması | 759,849 TL | 1,402,490 TL | 0.7393 | 20.39% |
| Log Dönüşümlü Model | 821,251 TL | 1,405,653 TL | 0.7381 | 22.11% |
| Final Model (En İyi) | 741,720 TL | 1,219,581 TL | 0.7486 | - |

### 📈 4. Görselleştirme ve Yorumlar

* Gerçek vs Tahmin Edilen Fiyatlar: Nokta grafiği oluşturulmuş, çoğu tahmin doğrunun yakınında yer almaktadır.
* Hata Dağılımı: Hatalar genelde ±2 milyon TL aralığındadır.
* Özellik Önem Skoru:  
  + En önemli özellik: netSqm (Ağırlık: %78)
  + Diğerleri: grossSqm, county, subCategory, room, age, vs.
* Korelasyon Matrisi ve Kısmi Bağımlılık Grafiği de analiz edilmiştir.

### 📌 5. Sonuçlar

* Modelin R² değeri 0.74 seviyelerinde olup, veri setinin yaklaşık %74’ünü açıklayabilmektedir.
* En iyi sonuçlar, önemsiz özelliklerin çıkarılması ve hiperparametre optimizasyonu ile elde edilmiştir.
* Fiyat tahminleri bazı örneklerde %5’in altında hata verirken, bazı yüksek sapmalı örneklerde %50+ oranında sapma gözlenmiştir.

### ⚙️ 6. Öğrenilenler ve Zorluklar

* Veriyi temizlemek ve dönüştürmek, model başarısı için kritik önemdedir.
* Aykırı değerler modelin performansını düşürebilir.
* GridSearchCV zaman alıcıdır ancak hiperparametre seçiminde oldukça etkilidir.
* Bazı sınıflarda düşük veri yoğunluğu, modelin genel başarısını etkileyebilir.

## 🏠 Emlak Fiyat Tahmini: Genel Model Değerlendirme Raporu

### **🎯 1. Proje Amacı**

**Bu projenin amacı, farklı gayrimenkul alt kategorileri için çeşitli makine öğrenmesi algoritmalarını kullanarak fiyat tahmini yapmaktır. Alt kategoriler bazında (Daire, Villa, Müstakil Ev, Tarla) en iyi performansı veren modeller belirlenmiş ve karşılaştırmalı analiz gerçekleştirilmiştir.**

### **🧹 2. Veri Hazırlama ve Dönüştürme**

* **Veri Kaynağı: finalemlak2.xlsx**
* **Kategori Dönüşümü:**
  + **Farklı alt kategoriler sadeleştirilmiş (ör. "İmarlı - Konut" → "Tarla", "Köşk" → "Villa").**
  + **Bazı kategoriler fiyat eşiğine göre yeniden etiketlenmiştir (ör. Yazlıklar 2.5M üstüyse → "Villa").**
* **Yeni Özellik: toplam\_oda = room + livingRoom + bathRoom**
* **Aykırı Değer Temizliği: IQR yöntemi ile uç değerler filtrelendi.**
* **Kategorilere Göre Bölme: Veriler 4 alt kategoriye ayrılarak modeller ayrı ayrı eğitildi.**

### **🤖 3. Kullanılan Modeller**

**Aşağıdaki 4 farklı regresyon algoritması her alt kategoriye uygulanmıştır:**

* **DecisionTreeRegressor**
* **RandomForestRegressor**
* **LinearRegression**
* **KNeighborsRegressor (KNN)**

**Her alt kategori için en uygun model belirlendi ve final analiz bu modele göre yapıldı.**

### **📈 4. Performans Sonuçları (Alt Kategori Bazında)**

| **AltKategori** | **Model** | **R² Skoru** | **MAE (Milyon)** | **MAE (%)** |
| --- | --- | --- | --- | --- |
| **Daire** | **RandomForest** | **0.7711** | **0.3836** | **14.51** |
| **Villa** | **RandomForest** | **0.6681** | **1.0841** | **18.15** |
| **Müstakil Ev** | **RandomForest** | **0.3280** | **1.7870** | **28.87** |
| **Tarla** | **LinearRegression** | **0.7463** | **0.4194** | **15.53** |

**✅ *Not:* R² skoru ne kadar 1’e yakınsa model o kadar başarılıdır. MAE (Mean Absolute Error) ve MAE% ise tahmin hatalarını gösterir; düşük olması istenir.**

### **📊 5. Ağırlıklı Genel Performans**

**Kategori büyüklüklerine göre hesaplanan ağırlıklı genel skorlar:**

* **R² (Açıklanan Varyans): 0.7170**
* **Ortalama MAE (Milyon): 0.6424**
* **Ortalama MAE (%): 16.34**

**Bu sonuçlar, modelin genel anlamda güvenilir bir tahmin sunduğunu göstermektedir.**

### **🏠 6. Örnek Tahmin Uygulaması**

**Özellikleri verilen bir “Daire” için model, fiyat tahmini yapmıştır:**

**Özellikler:**

**- İl: Sakarya**

**- İlçe: Serdivan**

**- Yapı Tipi: Apartman**

**- Isınma: Doğalgaz (Kombi)**

**- m²: 85 (net) / 100 (brüt)**

**- Oda Sayısı: 3+1**

**- Yaş: 5**

**🔍 Tahmini Fiyat: 2,304,600 TL**

### **📌 7. Yorum ve Öneriler**

#### **✨ Güçlü Yönler:**

* **Kategorilere özel model seçimi doğruluğu artırmıştır.**
* **Aykırı değer temizliği modelin genelleme gücünü yükseltmiştir.**
* **RandomForest modelleri, genelde en iyi performansı göstermiştir.**

#### **⚠️ Geliştirme Alanları:**

* **“Müstakil Ev” ve “Tarla” kategorilerinde örnek sayısı az ve hata oranları yüksektir.**
* **Fiyat tahminlerinde konumun koordinat bazlı etkisi dahil edilmemiştir.**
* **Model açıklanabilirliği için SHAP, LIME gibi analizler eklenebilir.**

### **✅ Sonuç**

**Bu proje kapsamında yapılan çalışma; farklı model türlerinin kıyaslanması, alt kategoriye göre optimize edilmesi ve tahmin doğruluğunun istatistiksel olarak analiz edilmesini içermektedir. Özellikle “Daire” ve “Villa” kategorilerinde R² > 0.70 başarısı ile güvenilir fiyat tahminleri yapılabilmiştir.**

# REGRESYON KARAR AĞACI – Ayşe

Karar ağaçları, verileri dallara ayırarak karar verme süreçlerini görselleştirmeye ve yorumlamaya yardımcı olan algoritmalardır. Regresyon karar ağacı ise sürekli sayısal verilerin tahmininde kullanılır.

## 1. Veri Seti ve Ön İşleme

## 1.1 Veri Setinin Genel Özellikleri

Veri setinde her satır bir emlak ilanını temsil etmektedir. Toplam 25 sütun, hem sayısal hem de kategorik niteliklerden oluşmaktadır. Hedef değişken konutun satış fiyatıdır (price).

## 2.1 Eksik Değer ve Veri Tipi Kontrolü

Veri setinde eksik değer bulunmamaktadır.

Boolean değişkenler (örneğin furnished) sayısal olarak 0 ve 1'e çevrilmiştir.

## 2.2 Uç Değerlerin Düzenlenmesi

Fiyat değişkeninde uç değerler, modeli yanıltmaması için Winsorization yöntemiyle alt ve üst %2’lik dilimlerle sınırlandırılmıştır.

Daha sonra, alternatif olarak Z-skoru yöntemiyle uç değerler filtrelenmiş ve bu yöntemle daha iyi sonuçlar elde edilmiştir.

## 2.3 Kategorik Değişkenlerin Kodlanması

İki aşamalı bir strateji uygulanmıştır:

Ordinal Encoding: buildState gibi sıralı ilişki içeren kategorik sütunlara uygulanmıştır.

Target Encoding (Mean Encoding): İlçe (county) ve mahalle (district) sütunları için ilk olarak ortalama fiyatlar hesaplanmış ve bu ortalamalara göre encode edilmiştir. Daha sonra bu yöntem diğer tüm kategorik değişkenlere genişletilmiştir.

## 3. İlk Model: Temel Karar Ağacı

Eğitim ve test veri seti %80-%20 oranında ayrılmıştır.

Varsayılan parametrelerle bir DecisionTreeRegressor modeli eğitilmiştir ve sonuçları şu şekilde::

metin, yazı tipi, beyaz, cebir içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

## 4. Optimizasyon Çalışmaları

## 4.1 Derinlik Ayarı (max\_depth)

Aşırı öğrenmenin önüne geçmek için modelin derinliği 2 ila 20 arasında denenmiş, en düşük hata 6 derinlikte elde edilmiştir.

En iyi ortalama RMSE: 2,068,805 TL (max\_depth = 6)

## 4.2 Log Dönüşümü

Log Dönüşümü ile fiyat değerlerini sıkıştırıp çarpıklığın keskinliğini yumuşatmaya çalıştım. Model daha küçük verilerle daha anlamlı çalışacak.

metin, diyagram, ekran görüntüsü, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, çizgi, ekran görüntüsü, diyagram içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Çarpıklık (skewness) orijinal: 2.26 → log dönüşümü sonrası: 0.72

Bu dönüşüm modeli daha dengeli hale getirmiştir.

## 4.3 GridSearchCV ile Hiperparametre Optimizasyonu

GridSearch ile model için en iyi hiperparametreler test edilip model bu parametrelerle tekrar eğitildiğinde başarı metrikleri şu şekilde değişmiştir.

metin, yazı tipi, makbuz, beyaz içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Maalesef ki dişe dokunur bir iyileşme görülemedi.

**5. Yeni Bir Model Yaklaşımı**

One hot encoding’lerden dolayı önceki modelde sütun sayısı 60 olmuştu, bu sefer önceki modelimden farklı olarak target label encoding ve Z-Skoru ile filtreleme yöntemi kullandım.

## 5.1 Tüm Kategorik Sütunlar İçin Target Encoding

İlk modelde yalnızca ilçe ve mahalleye uygulanan target encoding, ikinci aşamada tüm kategorik sütunlara uygulanmıştır. Bu yöntem, rastgele sayısal atamalar yerine fiyatla orantılı olarak daha anlamlı dönüşümler sağlamıştır.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

## 5.2 Z-Skoru ile Filtreleme

Price sütunundaki uç değerler modelin başarısını olumsuz etkiliyordu, bu sebeple z-skoru kullanarak ortalamadan 2.5 standart sapma uzağında olan değerleri filtreledim. Filtreleme sonrası minimum price değeri 19b iken maksimum price değeri 21m oldu.

Minimum değer çok küçük ve yanıltıcı olduğundan minimum benzersiz fiyatları sıraladım ve diğer verilerle orantısız şekilde çok küçük olan iki değeri tekrar filtreledim.

Verimin son haline, ilk modelimdeki gibi log dönüşümü ve Gridsearch ile en iyi parametrelerin belirlenmesi adımlarını uyguladım.

## 5.3 En Önemli 15 Özelliğin Seçimi

Karar ağacı modelinden elde edilen değişken önem puanlarına göre ilk 15 özellik seçilerek model yeniden kuruldu.

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Bu grafik beni çok şaşırttı açıkçası ama Regresyon Karar Ağaçları için normal bir durummuş. Çok güçlü korelasyon gösteren bir iki değişkene göre ilk düğümler oluşur ve böyle dengesiz bir grafik oluşabilirmiş.

Karar Ağacı şemasında da göreceğiz ki ağacın verdiği ilk karar subCategory sorgulamasıyla gerçekleşiyor. Kararı en çok etkileyen değişken de doğal olarak kendisi.

Sonraki düğümleri de bu grafiğe uygun şekilde netSqm ve district takip ediyor.

## 5.4 Son Modelin Değerlendirilmesi

Özellik seçimi sonrasında modelin performansı öncekine göre gözle görülür iyileşme gösterdi.

metin, makbuz, yazı tipi, beyaz içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Bu model hem daha az değişkenle hem de daha yüksek doğrulukla tahminler üretebilmektedir.

metin, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, diyagram içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Gerçek ve tahmin edilen fiyat grafiği mükemmel olmasa da oldukça iyi.

## 6. Karar Ağacı Şeması ve Değerlendirilmesi

Modelin eğitiminde kullanılan en iyi hiperparametrelerle oluşturulan regresyon karar ağacı aşağıdaki gibidir. Bu ağaç, fiyatların logaritmik dönüşümüne göre çalışmaktadır ve dallanmalar, varyansı en çok azaltan özellik ve eşiklere göre yapılmaktadır.

metin, plan, harita, diyagram içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

## 6.1 Hiperparametrelerin Karar Ağacı Şemasında İncelenmesi

**6.1.1 max\_depth=10 – Maksimum Derinlik:**

* Karar ağacının kökten yapraklara kadar olan yolunun maksimum 10 seviye derinliğe sahip olmasına izin verir.
* Görselde, dallanma yapısı 10 seviyeyi geçmiyor gibi görünüyor (tam saymak gerekebilir ama bu düzeyde yoğunluk, genellikle derinlik sınırına ulaşıldığını gösterir).
* Modelin daha fazla derinleşmemesi max\_depth=10 sınırıyla uyumludur.

**6.1.2 min\_samples\_split=20 – Dallanma için Minimum Örnek Sayısı:**

* Bir iç düğümde dallanma (split) yapılabilmesi için en az 20 örnek olmalıdır.
* Görselde yer alan iç düğümlerdeki samples sayısına baktığımızda, dallanan düğümlerin neredeyse tamamı 20 ve üzeri örnek içermektedir.
* Bu durum, bu parametrenin uygulandığını gösterir.

**6.1.3 min\_samples\_leaf=5 – Yaprak Düğümde Minimum Örnek:**

* Bir yaprak (son) düğümde en az 5 örnek olması gerekir.
* Görseldeki yaprak düğümlerinde genellikle samples değeri 5 veya daha fazla. Daha az örnek içeren düğüm görünmüyor.
* Bu da parametrenin etkin biçimde uygulandığını gösterir.

## 6.2 Kök Düğümün Yorumlanması

**metin, plan, harita, diyagram içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**Koşul:** subCategory\_encoded <= 7.5  
 **value = 15.044**, **samples = 1709**, **squared\_error = 0.434**

* Bu düğüm, modelin ilk ayrım yaptığı yerdir. %80 oranında ayrım yaptığımız için başlangıçta 1709 adet veri vardır.
* subCategory\_encoded sütunu, emlak ilanının türünü (örneğin daire, villa, müstakil ev, bina vb.) temsil eder.
* 7.5 değeri, target encoding ile verilen alt tür kodlamaları üzerinden gelir.
* Bu noktada model, emlak ilanlarını daha uygun fiyatlı olan türler (örneğin: daire, yazlık, tarla vs.) ile daha pahalı türler (villa, köşk, çiftlik evi gibi) arasında ayırmaktadır.
* Sol dala gidenler (<= 7.5) daha uygun fiyatlı ilanları, sağ dala gidenler ise genellikle daha lüks ve pahalı ilanları temsil eder.
* Value değeri o düğümdeki ortalama fiyatları temsil eder. Log dönüşümlü hali 15.044 iken gerçek fiyatı 3,444,109 TL’dir.

## 6.3 İkinci Düzey – Sol Dal Yorumlanması

**Koşul:** room <= 1.5  
 **value = 14.524**, **samples = 715**, **squared\_error = 0.088**

* Bu düğümde model, oda sayısını dikkate alır.
* 1.5 oda ve altı olan ilanlar ile daha fazla odaya sahip olanlar ayrıştırılır.
* Oda sayısı arttıkça genellikle fiyat da artacağından, bu mantıklı bir bölünmedir.
* value = 14.524 olduğuna göre log fiyat yaklaşık exp(14.524) ≈ 2,030,000 TL civarındadır.

## 6.4 İkinci Düzey – Sağ Dal Yorumlanması

**Koşul:** netSqm <= 94.5  
 **value = 14.687**, **samples = 1096**, **squared\_error = 0.162**

* Bu düğümde model, net metrekare bilgisini değerlendirir.
* 94.5 m² altı evlerle, daha geniş evler ayrıştırılmıştır.
* Net metrekare fiyat üzerinde önemli bir etkendir ve bu noktadaki ayrım da varyansı azaltmaya yönelik oldukça anlamlıdır.
* value = 14.687 → exp(14.687) ≈ 2,412,000 TL olarak yorumlanabilir.

## 6.5 İlk Üç Düğüm İçin Genel Değerlendirme

Modelin ilk üç düğümde yaptığı kararlar şu şekilde sıralanabilir:

1. **Ev türü** → lüks mü değil mi?
2. **Oda sayısı veya net metrekare** → büyüklük ölçütü
3. Varyansı azaltan özelliklere göre daha küçük gruplara bölerek log fiyatı tahmin etmeye çalışır.

Bu yapı, modelin önce konut tipine göre genel fiyat seviyesini belirlediğini, ardından evin büyüklüğü (oda sayısı, metrekare) gibi değişkenlerle daha isabetli tahminler yapmaya çalıştığını göstermektedir.

## 7. Genel Sonuç ve Değerlendirme

Bu çalışmada, Sakarya iline ait konut verileri kullanılarak konut fiyatlarını tahmin edebilen bir regresyon karar ağacı modeli geliştirilmiştir. Modelin başarımı, çeşitli veri ön işleme yöntemleri, encoding stratejileri, log dönüşümü ve hiperparametre optimizasyonları ile sistematik olarak artırılmaya çalışılmıştır.

Öne çıkan adımlar şunlardır:

* Fiyat verisinin uç değerlerini Winsorization ve Z-skoru ile filtreleyerek veri seti dengelenmiştir.
* Kategorik değişkenler için, hedef değişken (fiyat) ile ilişkili olacak şekilde **target encoding** yöntemi kullanılmıştır.
* Fiyat değişkeni, dağılımın çarpıklığını azaltmak için **log dönüşümüne** tabi tutulmuştur.
* Hiperparametreler (max\_depth, min\_samples\_leaf, min\_samples\_split) GridSearchCV ile en iyi kombinasyon olacak şekilde belirlenmiştir.
* Özellik önemi analizi sonucunda en etkili 15 değişken seçilerek, model sadeleştirilmiş ve başarı korunmuştur.

## 7.1 Başarı Metriklerinin Değerlendirilmesi

**Mean Absolute Error (MAE): 993.105 TL**

* Bu değer, modelin tahmin ettiği fiyatlar ile gerçek fiyatlar arasındaki mutlak farkların ortalamasını verir.
* Ortalama olarak model, yaklaşık **993 bin TL** sapma ile tahmin yapmaktadır.
* Konut fiyatlarının milyon TL mertebesinde olduğu düşünüldüğünde, bu sapma tolerans sınırları içinde kabul edilebilir düzeydedir.
* Ancak üst düzey doğruluk isteyen uygulamalarda daha düşük MAE hedeflenmelidir.

**Root Mean Squared Error (RMSE): 1,708,198 TL**

* RMSE, hata karelerinin ortalamasının kareköküdür. Büyük hataları daha fazla cezalandırdığı için duyarlıdır.
* RMSE'nin MAE’den daha yüksek çıkması, bazı örneklerde modelin yüksek sapmalar yapabildiğini gösterir.
* Yine de bu değer, verideki uç fiyatlara rağmen oldukça makul bir düzeydedir.

**Mean Absolute Percentage Error (MAPE): %22.74**

* MAPE, ortalama hata yüzdesini verir. Gerçek değerin yüzde kaçına hata yapıldığını gösterir.
* Bu çalışmada modelin tahminleri, gerçek değerlerden ortalama **%22.74** sapmıştır.
* %20-25 bandı, emlak gibi fiyat aralıkları geniş olan sektörlerde genellikle kabul edilebilir kabul edilir.
* Ancak uygulama senaryosuna göre iyileştirme yapılabilir.

**R-kare (R²): 0.7340**

* R², modelin hedef değişkendeki toplam varyansın ne kadarını açıkladığını gösterir.
* Bu değerin **0.73** olması, modelin fiyat değişimlerinin %73’ünü açıklayabildiğini ifade eder.
* Regresyon problemlerinde 0.70 üzeri R² değerleri genellikle başarılı olarak değerlendirilir.