

**TÜRKİYE CUMHURİYETİ  
YILDIZ TEKNİK ÜNİVERSİTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



**AVUSTRALYA - MELBOURNE ŞEHİRİ İÇİN MAKİNE  
ÖĞRENMESİ ALGORİTMALARINI KULLANARAK EV  
FİYAT TAHMİNİ**

16011701 – Atakan TEKOĞLU

16011012 – Süleyman Ali Burak ÇINAR

**BİLGİSAYAR PROJESİ**

Danışman  
Prof. Dr. Oya KALIPSIZ

Nisan, 2020



## **TEŞEKKÜR**

---

Bize ayırdıkları kıymetli zaman ve sağladıkları destek için değerli öğretim üyelerimiz Prof. Dr. Oya KALIPSIZ ve Dr. Hamza Osman İLHAN'a teşekkür ederiz.

Atakan TEKOĞLU  
Süleyman Ali Burak ÇINAR

# İÇİNDEKİLER

---

KISALTMA LİSTESİ	v
ŞEKİL LİSTESİ	vi
TABLO LİSTESİ	viii
ÖZET	ix
ABSTRACT	x
<b>1 GİRİŞ</b>	<b>1</b>
<b>2 ÖN İNCELEME</b>	<b>2</b>
<b>3 FİZİBİLİTE</b>	<b>4</b>
3.1 Teknik Fizibilite . . . . .	4
3.1.1 Yazılım Fizibilitesi . . . . .	4
3.1.2 Donanım Fizibilitesi . . . . .	4
3.1.3 İş Gücü ve Zaman Planlaması . . . . .	5
3.1.4 Yasal Fizibilite . . . . .	7
3.1.5 Ekonomik Fizibilite . . . . .	7
<b>4 SİSTEM ANALİZİ</b>	<b>8</b>
4.1 Veri Seti Tanımlaması . . . . .	8
4.1.1 Melbourne Housing Veri Seti . . . . .	8
4.1.2 Kullanıcı Senaryosu Diyagramı . . . . .	9
4.1.3 Veri Akış Diyagramı . . . . .	9
<b>5 SİSTEM TASARIMI</b>	<b>10</b>
5.1 Veri Seti Analizi . . . . .	10
5.1.1 Melbourne Housing Veri Seti . . . . .	10
5.1.2 Veri Setinin Alanları . . . . .	11
5.2 Veri Seti Tasarımı . . . . .	12
5.2.1 Veri Temizleme . . . . .	12

5.2.2	Veri Setinin Model İçin Hazırlanması . . . . .	21
5.3	Makine Öğrenmesi Algoritmaları . . . . .	23
5.3.1	Linear Regression . . . . .	24
5.3.2	GridSearchCV . . . . .	25
5.3.3	Ridge Regressor . . . . .	26
5.4	Yapay Sinir Ağları . . . . .	29
5.4.1	ANN 2 Layers . . . . .	29
5.4.2	ANN 3 Layers . . . . .	32
5.4.3	ANN 5 Layers . . . . .	33
<b>6</b>	<b>UYGULAMA</b>	<b>36</b>
<b>Referanslar</b>		<b>39</b>
<b>Özgeçmiş</b>		<b>40</b>

## KISALTMA LİSTESİ

---

NaN	Not a Number
df	Data Frame
ANN	Artificial Neural Network

## ŞEKİL LİSTESİ

---

Şekil 2.1 Bivariate Analysis . . . . .	3
Şekil 3.1 İş Gücü ve Zaman Planlaması . . . . .	6
Şekil 3.2 Çalışan Giderleri . . . . .	7
Şekil 3.3 Donanım ve Yazılım Giderleri . . . . .	7
Şekil 4.1 Melbourne City Data Set . . . . .	8
Şekil 4.2 Kullanıcı Senaryosu Diyagramı . . . . .	9
Şekil 4.3 Veri Akış Diyagramı . . . . .	9
Şekil 5.1 Veri Setine Genel Bakış . . . . .	10
Şekil 5.2 Veri ve Özellik Sayısı İçin Çalışan Kod . . . . .	11
Şekil 5.3 Veri Tipleri . . . . .	12
Şekil 5.4 Veri Seti Üzerindeki NaN alanlar . . . . .	13
Şekil 5.5 Ev Fiyatının Histogram Görüntüsü . . . . .	13
Şekil 5.6 Correlation Matrix . . . . .	14
Şekil 5.7 Alanlar Silindikten Sonra Veri Setine Bakış . . . . .	15
Şekil 5.8 BuildingArea ve Landsize Alanlarının Temizlenmesi . . . . .	15
Şekil 5.9 Boş Alanların Doldurulduktan Sonraki Görüntüsü . . . . .	15
Şekil 5.10 Total Alanın Belirtilmesi . . . . .	16
Şekil 5.11 Landsize ve Buildingarea Silindikten Sonra Veri Seti . . . . .	16
Şekil 5.12 Distance Unique Alanlar . . . . .	17
Şekil 5.13 Fonksiyon Sonrası Distance Alanları . . . . .	17
Şekil 5.14 Price - Rooms İlişkisi . . . . .	18
Şekil 5.15 Price - Type İlişkisi . . . . .	18
Şekil 5.16 Price - Distance İlişkisi . . . . .	19
Şekil 5.17 Price - Total İlişkisi . . . . .	19
Şekil 5.18 Price - Bathroom İlişkisi . . . . .	20
Şekil 5.19 Price - Car İlişkisi . . . . .	20
Şekil 5.20 Price - CouncilArea İlişkisi . . . . .	21
Şekil 5.21 Veri Setinin Son Hali . . . . .	21
Şekil 5.22 CouncilArea Alanının Sayısal Alana Çevrilmesi . . . . .	21
Şekil 5.23 Type Alanının Sayısal Alana Çevrilmesi . . . . .	22
Şekil 5.24 Tüm Alanların Tek Bir Değişken İçinde Birleştirilmesi . . . . .	22

Şekil 5.25 Id - CouncilArea - Type Alanlarının Silinmesi . . . . .	23
Şekil 5.26 X ve y Değişkenleri . . . . .	23
Şekil 5.27 trainTestSplit Metodu . . . . .	24
Şekil 5.28 LinearRegression nesnesi ve RMSE Değeri . . . . .	24
Şekil 5.29 Tahmin Verilerinin Görselize Edilmesi İçin Fonksiyon . . . . .	24
Şekil 5.30 Tahmin Verilerinin Görselize Edilmesi . . . . .	25
Şekil 5.31 GridSearch İçin Fonksiyon . . . . .	25
Şekil 5.32 Ridge Regressor İçin Fonksiyon . . . . .	26
Şekil 5.33 alpha 0.01 . . . . .	27
Şekil 5.34 alpha 0.1 . . . . .	27
Şekil 5.35 alpha 1 . . . . .	28
Şekil 5.36 alpha 10 . . . . .	28
Şekil 5.37 alpha 100 . . . . .	29
Şekil 5.38 Train Data . . . . .	30
Şekil 5.39 Test Data . . . . .	30
Şekil 5.40 Veri Birleşimi ve Id Değerinin Saklanması . . . . .	30
Şekil 5.41 Verinin İkiye Ayrılması . . . . .	31
Şekil 5.42 Feature Scaling . . . . .	31
Şekil 5.43 Model Initialize ve Katman Ekleme ANN 2 Layers . . . . .	31
Şekil 5.44 ANN 2 Layers Fitting . . . . .	31
Şekil 5.45 ANN 2 Katmanlı Tahmin İşlemi . . . . .	32
Şekil 5.46 ANN 2 Katmanlı Tahmin Değerleri . . . . .	32
Şekil 5.47 Model Initialize ve Katman Ekleme ANN 3 Layers . . . . .	32
Şekil 5.48 ANN 3 Layers Fitting and Compile . . . . .	33
Şekil 5.49 ANN 3 Katmanlı Tahmin İşlemi . . . . .	33
Şekil 5.50 ANN 3 Katmanlı Tahmin Değerleri . . . . .	33
Şekil 5.51 Model Initialize ve Katman Ekleme ANN 5 Layers . . . . .	34
Şekil 5.52 ANN 5 Layers Fitting and Compile . . . . .	34
Şekil 5.53 ANN 3 Katmanlı Tahmin İşlemi . . . . .	34
Şekil 5.54 ANN 5 Katmanlı Tahmin Değerleri . . . . .	35
Şekil 6.1 Uygulama Görüntüsü . . . . .	36
Şekil 6.2 Uygulama Görüntüsü 1 . . . . .	37
Şekil 6.3 Uygulama Görüntüsü 2 . . . . .	38

## TABLO LİSTESİ

---

Tablo 5.1 GridSearchCV Best Decision . . . . .	26
--	----

## ÖZET

---

# Avustralya - Melbourne Şehri İçin Makine Öğrenmesi Algoritmalarını Kullanarak Ev Fiyat Tahmini

Atakan TEKOĞLU

Süleyman Ali Burak ÇINAR

Bilgisayar Mühendisliği Bölümü

Bilgisayar Projesi

Danışman: Prof. Dr. Oya KALIPSIZ

Bu raporda çeşitli makine öğrenmesi algoritmalarını kullanarak evlerin bazı özelliklerine dayanarak satış fiyatı tahmini yapmaya çalışıyoruz. Evlerin satış fiyatları oda sayısı, şehir merkezine olan uzaklık, evin tipi, garaj sayısı, toprak alanının büyüklüğü, evin bulunduğu muhite göre belirleniyor. Bunun için makine öğrenmesi algoritmaları kullanılmaktadır. Lineer regresyon, Lasso regresyon, karar ağaçları algoritması gibi algoritmalar tahmin için kullanılmaktadır. Veri temizleme işleminden sonra yaklaşık olarak 16.000 veri üzerinden tahmin işlemi yapılmaktadır. Lineer regresyon 0.600619 ,Lasso regresyon 0.600609, Decision Tree ise 0.459391 değere sahip olmaktadır.

**Anahtar Kelimeler:** Tahmin etmek, fiyatlandırma, makine öğrenmesi, makine öğrenmesi algoritmaları, istatistik, veri temizleme, lineer regresyon, decision tree, lasso,

## **ABSTRACT**

---

# **HOUSING PRICE PREDICTION USING MACHINE LEARNING ALGORITHMS: THE CASE OF MELBOURNE CITY AUSTRALIA**

Atakan TEKOĞLU  
Süleyman Ali Burak ÇINAR

Department of Computer Engineering  
Computer Project

Advisor: Prof. Dr. Oya KALIPSIZ

In this paper, we are predicting the sale price of the houses using various machine learning algorithms. Housing sales price are determined by numerous factors such as area of the property, location of the house, Rooms, garage, number of bathrooms, house type and so on. This paper uses machine learning algorithms to build the prediction model for houses. Here, machine learning algorithms such as Linear regression and Lasso Regression technique and Decision Tree are employed to build a predictive model. We have considered about housing data of 16.000 properties after data cleaning. Linear Regression, Lasso Regression and Decision Tree show the value of 0.600619, 0.600609 and 0.459391 respectively. This paper also represents significance of our approach and the methodology.

**Keywords:** Real Estate, Prediction Model, Linear Regression, Decision Tree, Lasso, Data Cleaning

# 1 GİRİŞ

---

Makine öğrenmesi modellerini kullanarak bir veri setine dayanarak tahminde bulunma işlemi günümüzde elektronik ve bilgisayar bilimcilerin araştırma yaptığı alanların başında gelmektedir. Bir insanın binlerce, milyonlarca bilgi bulunduran bir veri setine bakarak yüksek oranda tahminde bulunması mümkün değildir. Bunun için veriyi iyi okuyabilmeli ve yorumlayabilmek gereklidir. Bu noktada yardımımıza makine öğrenmesi teknikleri yetişmektedir. Veri seti üzerinde bilgi çıkarımı yapmak ve tahminde bulunmak için makine öğrenmesi teknikleri kullanılmalıdır.

Bir veri setine dayanarak tahminde bulunma işlemi, bu veri setinin düzenli hale getirilmesi prensibine dayanır. İlk olarak veri üzerinde genel bir gözlem yapmalı ve bu gözleme dayanarak çeşitli veri temizleme işlemleri uygulanmalıdır. Veri temizleme işlemi sırasında istatistiksel metotlardan faydalanan gereklidir.

Bu proje kapsamında Python dili ve bu dil ile kullanılan Pandas ve Numpy gibi yaygın kütüphaneler kullanılacak olup, verinin düzenlenmesi için çeşitli Python kütüphaneleri yazılacak olup, bu kütüphaneler sayesinde gerçekleşip, veri üzerinde tanımlanabilen ısı haritasına bakılarak ayrıca veri düzenlenmesi yapılacaktır.

## **2 ÖN İNCELEME**

---

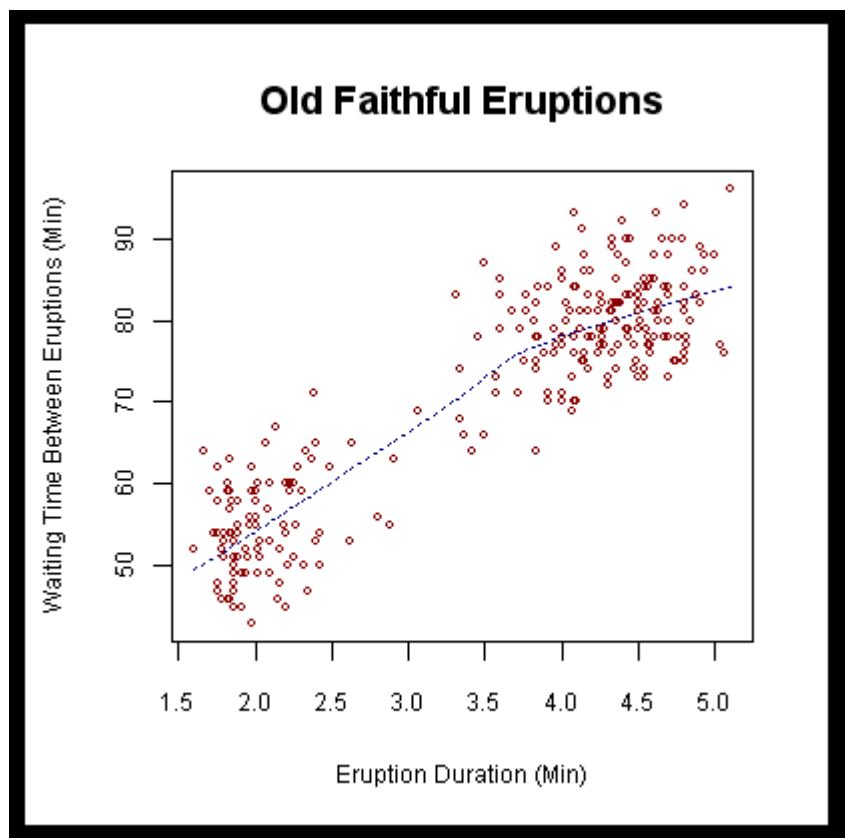
Projemizin sürekli ve gerçek değerli çıktılar üzerine kurulu olduğunu göz önünde bulundurarak bunun bir regresyon problemi olduğunu açık bir şekilde söylebiliriz. Buna bağlı olarak fiyat tahmini yapabilmek ve veri üzerinden modellerin çıkarım yapabilmesi için temizleme işlemi yapmak gerekmektedir. Temizleme işlemi internet üzerinde bulunan Weka gibi çeşitli yazılımlar aracılığı ile yapılabildiği gibi aynı zamanda Python kütüphaneleri kullanılarak da yapılmaktadır.

Veri analizi yaparken anormal değerleri saptamak, veri üzerindeki kayıp değerleri bulmak, düzensiz verileri ayıklamak, hem metinsel hem de sayısal veriler içeren alanları görebilmek için istatiksel ve grafiksel yaklaşımlar ile saptama yapılacaktır.

İki değişkene dayalı olarak işlem yapılan quantitative (statistical) analysis yöntemlerinden Bivariate analysis kullanılarak veri üzerinde daha iyi bir hakimiyet sağlanacak olup, aykırı değerler temizlecektir.

Sonuç olarak bu projede oda sayısı, evin bulunduğu konum, toprak alanı gibi parametrelere bağlı olarak ilgili ev için bir fiyat tahmini yapılması amaçlanmaktadır. Bunu yaparken çeşitli makine öğrenmesi modelleri karşılaştırıcalık olup, içlerinden en iyi sonuç veren kullanılacaktır.

Şekil-2.1 İki değişkenli olarak veri karşılaştırmasına örnektir.



Şekil 2.1 Bivariate Analysis

# **3**

## **FİZİBİLİTE**

---

Projenin, uygun yapıda tasarılanması için gerekli araştırmalar yapılmış ve teknik yapı hazırlanmaya başlanmıştır. Kullanımı kolaylaştıracak tasarım kararları belirlenmiştir

### **3.1 Teknik Fizibilite**

Sistemin yazılım ve donanım fizibilitesi Bölüm 3.1.1 ve Bölüm 3.1.2'de listelenmiştir

#### **3.1.1 Yazılım Fizibilitesi**

İstatistiksel veri analizi ve modellerin gerçeklemek için kullanılacak elementler aşağıda listelenmiştir.

- Windows 10
- Anaconda
- Jupyter Notebook
- Android Studio

İlgili işletim sistemi ve programların tamamı ücretsizdir.

#### **3.1.2 Donanım Fizibilitesi**

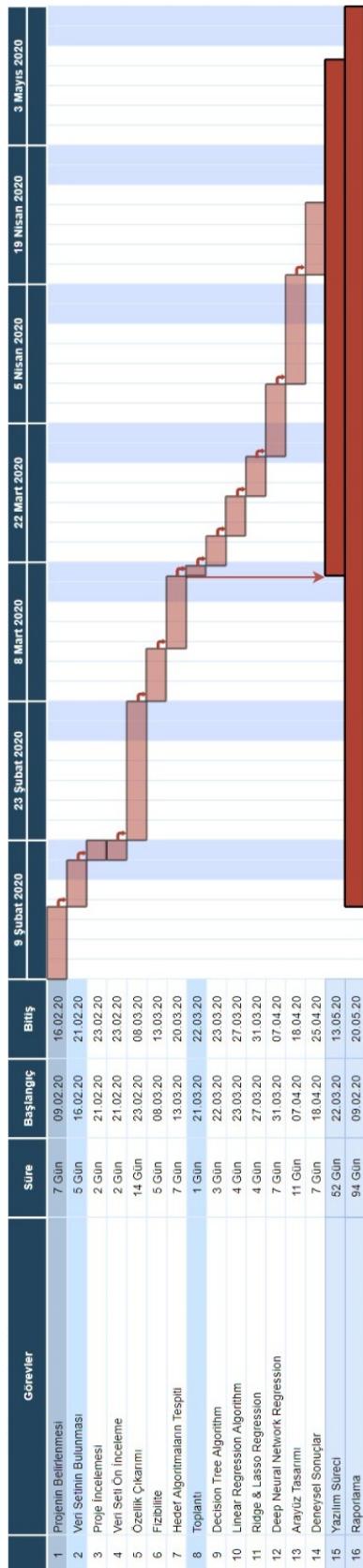
Yazılım fizibilitesinde kullanılması bahsedilen uygulamayı çalıştırılmak için kullanılacak bilgisayarlar minimum sistem gereksinimlerinin fazlasıyla üstündedir. Özellikler aşağıda listelenmiştir.

- 1.Bilgisayar

- Monster ABRA A5 V12.1
  - Intel Core i5-7300 HQ CPU 2.50GHz
  - 8192MB RAM
  - NVIDIA GeForce GTX 1050 4050MB
  - 1TB 7200RPM HDD
- 2.Bilgisayar
    - Monster ABRA A5 V15.2
    - Intel Coffee Lake Core i7-9750H
    - 8GB (1x8GB) DDR4 2666MHz
    - 4GB GDDR5 Nvidia GTX1650 128-Bit
    - 1TB 7200RPM HDD + 240GB M.2 SSD Sata 3

### **3.1.3 İş Gücü ve Zaman Planlaması**

Projenin tamamlanma süreci 101 gün olarak belirlenmiştir. Öncelikle uygun veri seti bulunup incelenecaktır. Veri seti bulunduktan sonra uygun hale getirmek için özellik çıkarımı yapılacaktır. Sonrasında model oluşturmada kullanılması gereken algoritmalar bulunup araştırılacaktır. Bu algoritmaların kullanım alanlarını görebilmek için literatür taraması yapılacak, konuya ilgili olan bilimsel çalışmalar incelenecaktır. Burda amaç genel işleyişi ve çalışma mantığını anlayıp uygun algoritmanın seçilmesi için gerekli donanımı kazanmaktır. Bu süreçte yazılım kısmına başlayıp projenin teslim tarihine yetişirilmesi sağlanacaktır. Kullanıcıya zahmetsız bir deneyim sunmak amacıyla arayüz tasarımlı yapılacaktır. Programın girdi çıktıları alınıp en iyi sonuçlara ulaşmak için karşılaşışmalar yapılacaktır. Projenin tamamlanmasında 2 adet Bilgisayar Mühendisliği 3. Sınıf Lisans öğrencisi görev alacaktır.



Şekil 3.1 İş Gücü ve Zaman Planlaması

### **3.1.4 Yasal Fizibilite**

Uygulamanın üzerinde gerçekleşeceği işletim sistemi Windows 10'un lisansı, Microsoft Imagine üzerinden ücretsiz olarak alınmıştır. Anaconda, Android Studio, Jupyter Notebook uygulamalarının ücretsiz lisans hakkı sağlayan Community versiyonu kullanılacaktır. Oluşturulacak platformun yasal uygunluğu araştırılarak herhangi bir hak ihlali bulunmadığı görülmüş ve sonuç olarak sistemin uygunluğuna karar verilmiştir.

### **3.1.5 Ekonomik Fizibilite**

Sistemin yazılım, donanım ve diğer tüm giderleri Şekil3.2 ve Şekil3.3 belirtilmiştir.

Çalışan Ünvanı	Çalışan Sayısı	Aylık Maaşı
<b>Yazılımcı</b>	2	3.500 TL
<b>Sistem Analisti</b>	1	4.500 TL
<b>Proje Yöneticisi</b>	1	5.500 TL
<b>Toplam</b>		<b>13.500 TL</b>

**Şekil 3.2 Çalışan Giderleri**

Geliştirme Bilgisayarları	Monster Abra A5 V12.1	4.000 TL
	Monster Abra A5 V15.2	6.500 TL
İşletim Sistemi	Windows 10	Ücretsiz
	Anaconda	Ücretsiz
IDE	Android Studio	Ücretsiz
	Jupyter Notebook	Ücretsiz
Toplam		<b>10.500 TL</b>

**Şekil 3.3 Donanım ve Yazılım Giderleri**

# 4 SİSTEM ANALİZİ

## 4.1 Veri Seti Tanımlaması

Veri seti; satırlarında gözlem birimleri, sütunlarında ise değişkenler bulunan iki boyutlu bir matristir. Satır ve sütunların kesişim bölgelerine hücre adı verilir. Her bir hücreye, gözlemlerin değerleri yani yapılan ölçmenin sonuçları sayı veya sembol olarak girilir. Bir hücrede herhangi bir sayı veya sembol olmaması, amaçlanan gözlemin bulunmadığını ifade eder. Bu bilgiler ışığında bizim veri setimizi tanıyalabiliriz.

### 4.1.1 Melbourne Housing Veri Seti

Projede kullanılacak veri seti Kaggle isimli siteden alınmıştır.[1] Bu veri seti 21 özellik ve 34.857 veriden oluşmaktadır. İçerisinde 8 kategorik ve 13 sayısal alan bulunmaktadır.

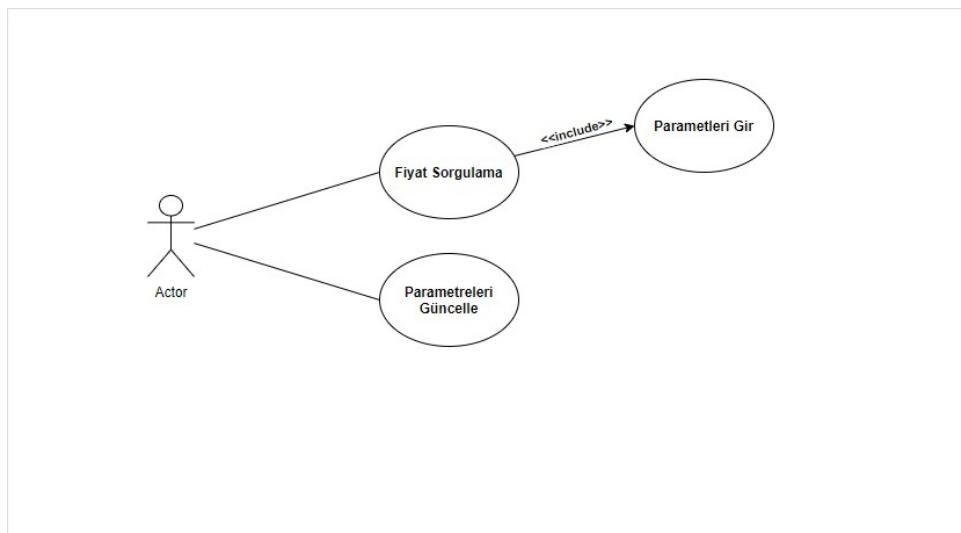
Aşağıda veri setimize genel olarak bir bakış yapılmıştır.

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	YearBuilt	C
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	NaN	NaN	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	NaN	
2	Abbotsford	25 Bloomberg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	1900.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	NaN	NaN	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	...	2.0	0.0	134.0	150.0	1900.0	
5	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5	3067.0	...	2.0	1.0	94.0	NaN	NaN	
6	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5	3067.0	...	1.0	2.0	120.0	142.0	2014.0	
7	Abbotsford	16 Maugie St	4	h	NaN	SN	Nelson	6/08/2016	2.5	3067.0	...	2.0	2.0	400.0	220.0	2006.0	
8	Abbotsford	53 Turner St	2	h	NaN	S	Biggin	6/08/2016	2.5	3067.0	...	1.0	2.0	201.0	NaN	1900.0	
9	Abbotsford	99 Turner St	2	h	NaN	S	Collins	6/08/2016	2.5	3067.0	...	2.0	1.0	202.0	NaN	1900.0	

10 rows x 21 columns

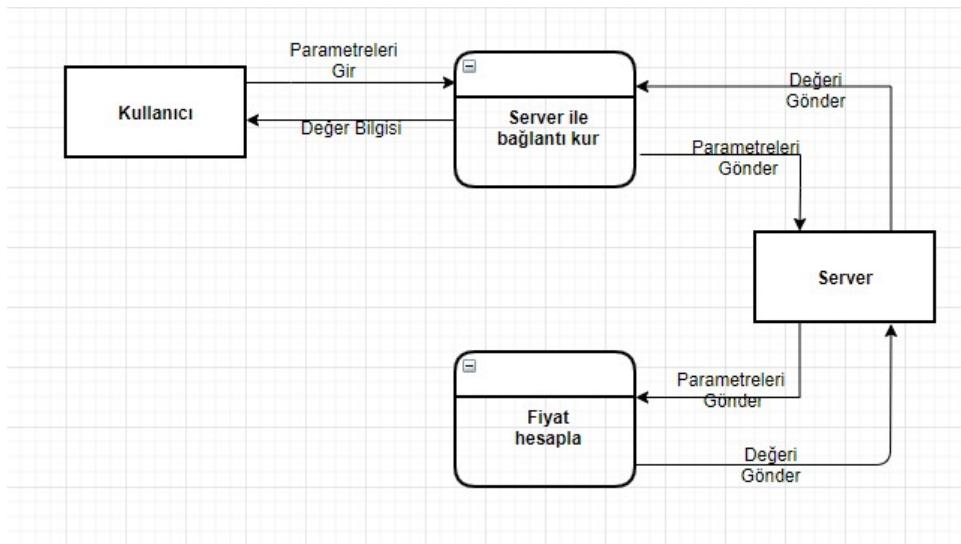
Şekil 4.1 Melbourne City Data Set

#### 4.1.2 Kullanıcı Senaryosu Diyagramı



Şekil 4.2 Kullanıcı Senaryosu Diyagrımı

#### 4.1.3 Veri Akış Diyagramı



Şekil 4.3 Veri Akış Diyagramı

# 5 SİSTEM TASARIMI

---

Gerçeklenmesi planlanan projede veri seti analizi yapılmış, bu analiz üzerinden ev fiyatını tahmin etmek için makine öğrenmesi modelleri kullanılacaktır. Bu modeller gelecek bölgelerde daha detaylı bir biçimde açıklanacaktır. Veri seti ham olarak 34.857 satır ve 21 sütundan oluşmaktadır. Veri seti içerisinde boş alanlar, değeri 0 olan anlamsız alanlar, kategorisi number olmasına rağmen string değer içeren değerler vardır. Böylece yüksek miktarda bilgi barındıran ve içinde çeşitli eksiklikler barındıran veri setini matplotlib, plotly, numpy ve pandas gibi popüler python temelli kütüphaneler ile temizleme işlemi yapılacaktır. Proje kapsamında model eğitiminden sonra deneme verilerinin girilmesi ile tahmin yapma işlemi yapılması düşünülmektedir.

## 5.1 Veri Seti Analizi

Bu bölümde Kaggle isimli siteden alınan[1] Melbourne şehrini verilerini barındıran veri seti incelenecektir.

### 5.1.1 Melbourne Housing Veri Seti

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	YearBuilt	C
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	NaN	NaN	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	NaN	
2	Abbotsford	25 Bloomberg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	1900.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	NaN	NaN	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	...	2.0	0.0	134.0	150.0	1900.0	
5	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5	3067.0	...	2.0	1.0	94.0	NaN	NaN	
6	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5	3067.0	...	1.0	2.0	120.0	142.0	2014.0	
7	Abbotsford	16 Maugie St	4	h	NaN	SN	Nelson	6/08/2016	2.5	3067.0	...	2.0	2.0	400.0	220.0	2006.0	
8	Abbotsford	53 Turner St	2	h	NaN	S	Biggin	6/08/2016	2.5	3067.0	...	1.0	2.0	201.0	NaN	1900.0	
9	Abbotsford	99 Turner St	2	h	NaN	S	Collins	6/08/2016	2.5	3067.0	...	2.0	1.0	202.0	NaN	1900.0	

10 rows x 21 columns

Şekil 5.1 Veri Setine Genel Bakış

### 5.1.2 Veri Setinin Alanları

- Sayısal Alanlar
  - Rooms
  - Price
  - Distance
  - Postcode
  - Bedroom2
  - Bathroom
  - Car
  - Landsize
  - BuildingArea
  - YearBuilt
  - Latitude
  - Longitude
  - Propertycount
- Kategorik Alanlar
  - Suburb
  - Address
  - Type
  - Method
  - SellerG
  - Date
  - CouncilArea
  - Regionname

Pandas kütüphanesi ile projeye dahil edilen veri seti üzerinde çalıştırılan 'shape' fonksiyonu ile kaç adet veri ve özellik olduğu öğrenilmiştir.

```
print("Veri Sayısı:{0} Özellik Sayısı:{1}".format(df.shape[0],df.shape[1]))  
Veri Sayısı:34857 Özellik Sayısı:21
```

**Şekil 5.2** Veri ve Özellik Sayısı İçin Çalışan Kod

- **Veri Seti Hakkında**

- Veri sayısı: 34.857
- Özellik Sayısı: 21

## 5.2 Veri Seti Tasarımı

### 5.2.1 Veri Temizleme

Bu noktadan sonra veri seti üzerine yorumlanabilirlik sağlanması gerekmektedir. Şekil 5.1 bizlere veri seti hakkında genel olarak bilgi verirken, Şekil 5.2 ise veri sayısı ve özellik bilgisini aktarıyor. Öte yandan alanlara göre veri tipi tanımlaması yapmaktadır. Bu bilgiyi ise Şekil 5.3 de görebiliriz.

```
df.info()  
  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 34857 entries, 0 to 34856  
Data columns (total 21 columns):  
 #   Column      Non-Null Count  Dtype     
---  --          --          --       --  
 0   Suburb      34857 non-null   object    
 1   Address     34857 non-null   object    
 2   Rooms       34857 non-null   int64     
 3   Type        34857 non-null   object    
 4   Price        27247 non-null   float64  
 5   Method       34857 non-null   object    
 6   SellerG     34857 non-null   object    
 7   Date         34857 non-null   object    
 8   Distance    34856 non-null   float64  
 9   Postcode    34856 non-null   float64  
 10  Bedroom2    26640 non-null   float64  
 11  Bathroom    26631 non-null   float64  
 12  Car          26129 non-null   float64  
 13  Landsize    23047 non-null   float64  
 14  BuildingArea 13742 non-null   float64  
 15  YearBuilt   15551 non-null   float64  
 16  CouncilArea 34854 non-null   object    
 17  Latitude     26881 non-null   float64  
 18  Longitude    26881 non-null   float64  
 19  Regionname   34854 non-null   object    
 20  Propertycount 34854 non-null   float64  
dtypes: float64(12), int64(1), object(8)  
memory usage: 5.6+ MB
```

**Şekil 5.3** Veri Tipleri

Veri setinde yüksek miktarda işe yaramaz alan bulunmaktadır. Bu alanlar veri seti üzerinde olumsuz etki yapmaktadır. Bu alanları Şekil 5.4 de büyükten küçüğe doğru sıralanmış vaziyette görmekteyiz.

Bizim için en fazla önem arz eden alan satış fiyatıdır. Satış fiyatı için histogram görünümü elde edelim ve yorumlayalım.

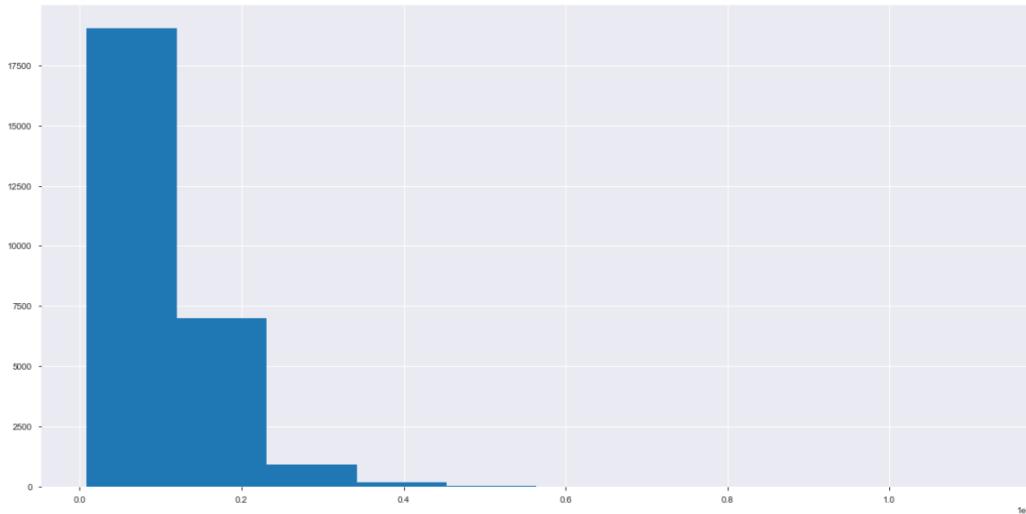
```
df.isnull().sum().sort_values(ascending=False)
```

BuildingArea	21115
YearBuilt	19306
Landsize	11810
Car	8728
Bathroom	8226
Bedroom2	8217
Longitude	7976
Latitude	7976
Price	7610
Regionname	3
Propertycount	3
CouncilArea	3
Postcode	1
Distance	1
Date	0
SellerG	0
Method	0
Type	0
Rooms	0
Address	0
Suburb	0
	dtype: int64

**Şekil 5.4** Veri Seti Üzerindeki NaN alanlar

Şekil 5.5 de görüleceği üzere histogram skew değeri 2.5889693410528607 ile right skew olarak görünüyor.Right skewed dağılıma sahip bir kümede mean değerinin median değerinden büyük olduğu söylenebilir.

```
plt.hist(df['Price'])
plt.show()
```



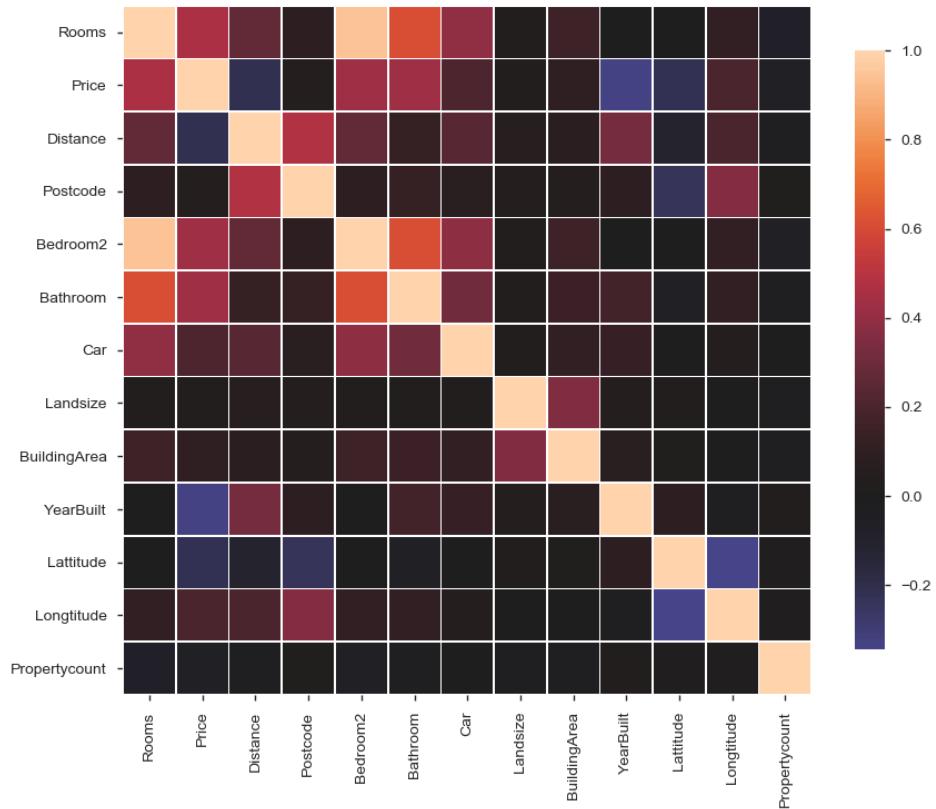
```
print("Skew of price:",df['Price'].skew())
Skew of price: 2.5889693410528607
```

**Şekil 5.5** Ev Fiyatının Histogram Görüntüsü

Daha iyi bir anlayış için Correlation Matrix ile görselize hale getirelim. Şekil 5.6 de görülen Correlation Matrix de iki özellik arasında doğrusal bir ilişki olup olmadığı, varsa bu ilişkinin şiddetini görüyoruz. Şekilde ayrıca korelasyon kat sayılarında görülmektedir. Korelasyon kat sayısı için aşağıdaki listede yorum yapılmıştır.[2]

- **Korelasyon katsayı( $r$ ) nın yorumu**

- $r < 0.2$  ise çok zayıf ilişki yada korelasyon yok
- $0.2-0.4$  arasında ise zayıf korelasyon
- $0.4-0.6$  arasında ise orta şiddette korelasyon
- $0.6-0.8$  arasında ise yüksek korelasyon
- $0.8 >$  ise çok yüksek korelasyon olduğu yorumu yapılır.



**Şekil 5.6 Correlation Matrix**

Şekil 5.4 de belirtildiği üzere veri setinin 6 alanındaki verilen çoğu kayıp verilerden oluşmaktadır.

Şekil 5.6 den yola çıkarak veri üzerindeki 'Address','SellerG','Suburb','YearBuilt','Latitude','Longitude','YearBuilt','Bedroom2','Date','Method','Propertycount','Regionname','Postcode' gibi

alanların bazlarının özellik olarak birbirine benzediği görülmeye. Bundan dolayı bu alanların silinmesi gerekmektedir.

Bu alanlar silindikten sonra veri seti Şekil 5.7 de belirtildiği gibi görülmektedir.

	Rooms	Type	Price	Distance	Bathroom	Car	Landsize	BuildingArea	CouncilArea
0	2	h	NaN	2.5	1.0	1.0	126.0	NaN	Yarra City Council
1	2	h	1480000.0	2.5	1.0	1.0	202.0	NaN	Yarra City Council
2	2	h	1035000.0	2.5	1.0	0.0	156.0	79.0	Yarra City Council
3	3	u	NaN	2.5	2.0	1.0	0.0	NaN	Yarra City Council
4	3	h	1465000.0	2.5	2.0	0.0	134.0	150.0	Yarra City Council

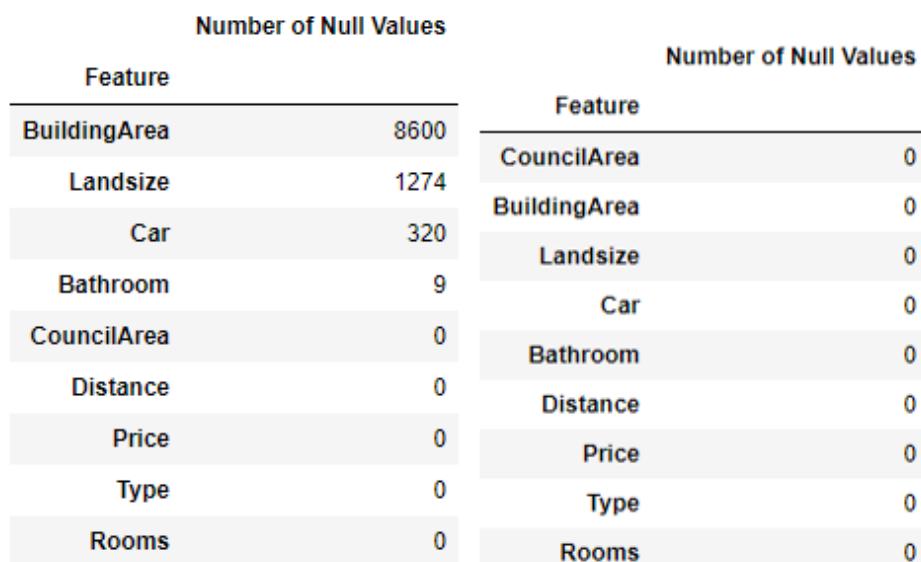
Şekil 5.7 Alanlar Silindikten Sonra Veri Setine Bakış

BuildingArea ve Landsize alanlarında bulunan NaN değerlerini kaldırmak için Şekil 5.8 de belirtilen fonksiyon yazılmıştır. Veri sayımız bu işleminden sonra 19.256 olmuştur.

```
df4=df3.dropna(subset=['BuildingArea','Landsize'], thresh = 1)
df4.shape
(19256, 9)
```

Şekil 5.8 BuildingArea ve Landsize Alanlarının Temizlenmesi

Not a Number alanları temizlendikten sonra hiçbir alanın boş kalmaması için ilgili konumlar bulunup 0 değeri ile doldurulur. Şekil 5.9 da öncesi ve sonrası görülmektedir.



Şekil 5.9 Boş Alanların Dolduruluktan Sonraki Görüntüsü

BuildingArea ve Landsize alanlarının benzerliği nedeniyle iki alanın değerleri toplanım Total isimli yeni bir değişken içerisinde değerlendirilmiştir. Bu sayede modelin performansının artırılması amaçlanmıştır. Şekil 5.10 de yapılan işlem belirtilmiştir.

```
df6 = df5.copy()
df6['Total'] = df6['Landsize'] + df6['BuildingArea']
df6.head()
```

	Rooms	Type	Price	Distance	Bathroom	Car	Landsize	BuildingArea	CouncilArea	Total
1	2	h	1480000.0	2.5	1.0	1.0	202.0	0.0	Yarra City Council	202.0
2	2	h	1035000.0	2.5	1.0	0.0	156.0	79.0	Yarra City Council	235.0
4	3	h	1465000.0	2.5	2.0	0.0	134.0	150.0	Yarra City Council	284.0
5	3	h	850000.0	2.5	2.0	1.0	94.0	0.0	Yarra City Council	94.0
6	4	h	1600000.0	2.5	1.0	2.0	120.0	142.0	Yarra City Council	262.0

Şekil 5.10 Total Alanın Belirtilmesi

Landsize ve Buildingarea bir düşünüldüğü için bu iki alana gerek kalmamıştır. Bu iki alanı attıktan sonra aşağıda veri setinin alanları görülmektedir.

```
df8 = df7.drop(['Landsize','BuildingArea'],axis = 'columns')
df8.head()
```

	Id	Rooms	Type	Price	Distance	Bathroom	Car	CouncilArea	Total	
1	2	2	h	1480000.0		1	1.0	1.0	Yarra City Council	202.0
2	3	2	h	1035000.0		1	1.0	0.0	Yarra City Council	235.0
4	5	3	h	1465000.0		1	2.0	0.0	Yarra City Council	284.0
5	6	3	h	850000.0		1	2.0	1.0	Yarra City Council	94.0
6	7	4	h	1600000.0		1	1.0	2.0	Yarra City Council	262.0

Şekil 5.11 Landsize ve Buildingarea Silindiğinden Sonra Veri Seti

Distance alanını incelediğimizde içerisindeki değerlerin birbirine çok yakın olduğu görülmektedir. Bu yakınlık distance özelliğinin parametrik olarak girilmesine olanak tanımıyor. Şekil 5.12 de birbirinden farklı olan tüm distance değerlerini görebiliyoruz.

```

df.Distance.unique()

array([ 2.5, 13.5, 3.3, 6.4, 13.8, 11.1, 6.3, 5.9, 11. , 12.2, 10.5,
       6.6, 9.7, 9.2, 13. , 13.9, 13.1, 10.8, 11.2, 10.7, 5.2, 11.8,
      11.7, 7.8, 9. , 3.2, 11.4, 8.9, 8.1, 9.3, 13.6, 3.4, 1.6,
     10.3, 8.5, 7.7, 8. , 9.4, 5.8, 3.5, 4.4, 12.1, 13.7, 14.5,
     4.6, 7.9, 15. , 12.8, 4.2, 5.6, 7.5, 6.2, 7.4, 8.7, 2.8,
     4.1, 6.9, 14.6, 8.4, 2.3, 5.5, 11.5, 14.7, 2.6, 9.9, 3.8,
     4.5, 8.8, 2.1, 1.2, 6.1, 12.6, 13.3, 6.5, 14.9, 1.9, 5.1,
     7. , 9.1, 9.5, 1.8, 1.5, 9.8, 12.4, 10.6, 8.2, 10.9, 10.4,
    14. , 12.5, 5.3, 12.7, 3.7, 3. , 5.7, 15.5, 4.3, 10.2, 16.5,
   23.2, 17.3, 35.2, 13.4, 25. , 16.1, 4. , 14.8, 10.1, 17.9, 17.5,
   16.7, 3.6, 43.4, 6.7, 20.6, 34.7, 23. , 24.7, 21.8, 22.2, 14.3,
   25.9, 1.3, 27. , 18. , 7.2, 36.9, 19.6, 28.8, 24.8, 15.4, 38. ,
  45.9, 12.9, 7.3, 20.4, 32.3, 45.2, 21.3, 16. , 18.4, 12.3, 5.4,
  25.2, 16.3, 33.8, 31.7, 29.8, 20. , 17.2, 16.2, 21.5, 14.2, 18.8,
  34.1, 22.7, 12. , 2.4, 19.9, 23.8, 23.5, 15.2, 35.4, 20.5, 2.7,
  20.8, 5. , 30.4, 15.3, 3.1, 17.6, 25.5, 27.2, 8.6, 31.2, 2. ,
  1.4, 0. , 26.5, 47.3, 0.7, 21.1, 23.6, 6.8, 34.9, 18.7, 26. ,
  41. , 47.4, 39.8, 27.1, 17.4, 37.5, 33.3, 31.6, 34.6, 44.2, 27.7,
  23.3, 29.3, 26.1, 29.9, 28.5, 43.3, 30.6, 39. , 31.4, 20.1, 35.5,
  48.1, 16.6, 22.9, 33. , nan, 29.5, 32.6])

```

**Şekil 5.12** Distance Unique Alanlar

Bir fonksiyon yazarak bu alanları 4 kategoride birleştirirsek parametre olarak girebiliriz.

- Distance 0: Değer 0
- Distance 0 ile 5 arasında: Değer 1
- Distance 5 ile 25 arasında: Değer 2
- Diğer Durumlar: 3

Şekil 5.13 de kullanılan fonksiyon ve elde edilen çıktı görülmektedir.

```

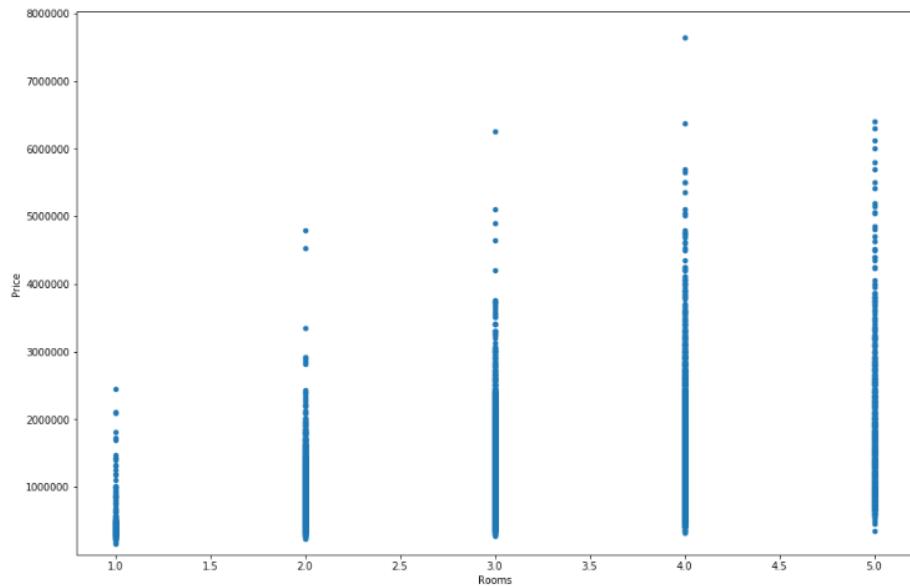
def set_distance(x):
    if x == 0:
        return 0
    elif 0 < x < 5:
        return 1
    elif 5<= x <= 25 :
        return 2
    else:
        return 3
df7 = df6.copy()
df7['Distance'] = df7['Distance'].apply(set_distance)
df7.Distance.unique()

array([1, 2, 3, 0], dtype=int64)

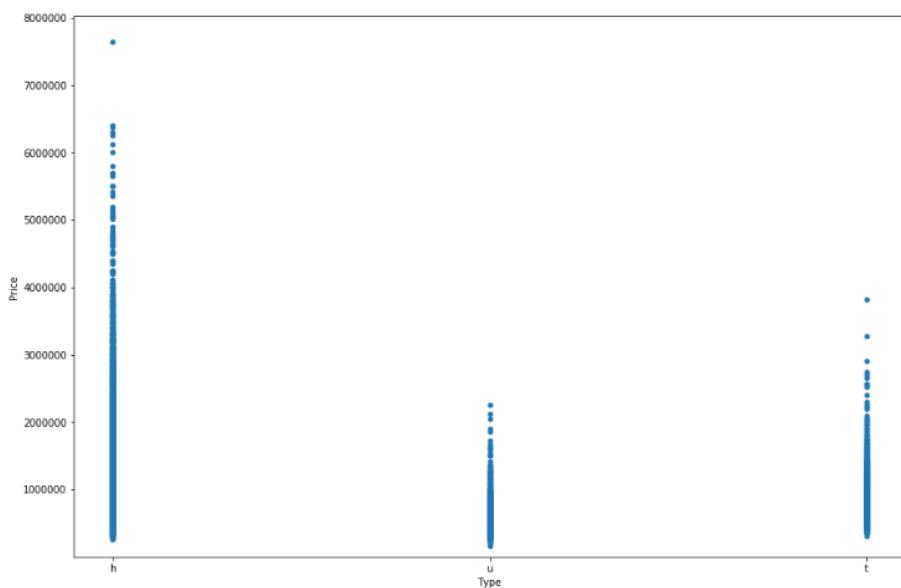
```

**Şekil 5.13** Fonksiyon Sonrası Distance Alanları

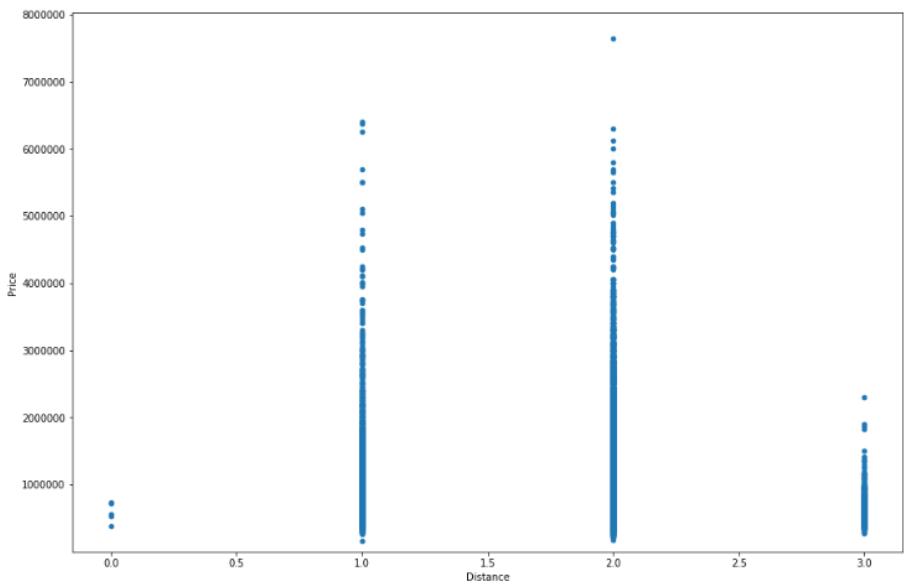
Oluşan son durumdan sonra veri setimizde bulunan alanların fiyat ile olan dağılımlarını ilişkisel olarak görebilmek için plot çizdirelim. Takip eden kısımlarda Rooms, Type, Distance, Total, Bathroom, Car ve CouncilArea alanları için Price alanı ile olan ilişki grafiksel olarak gösterilmiştir.



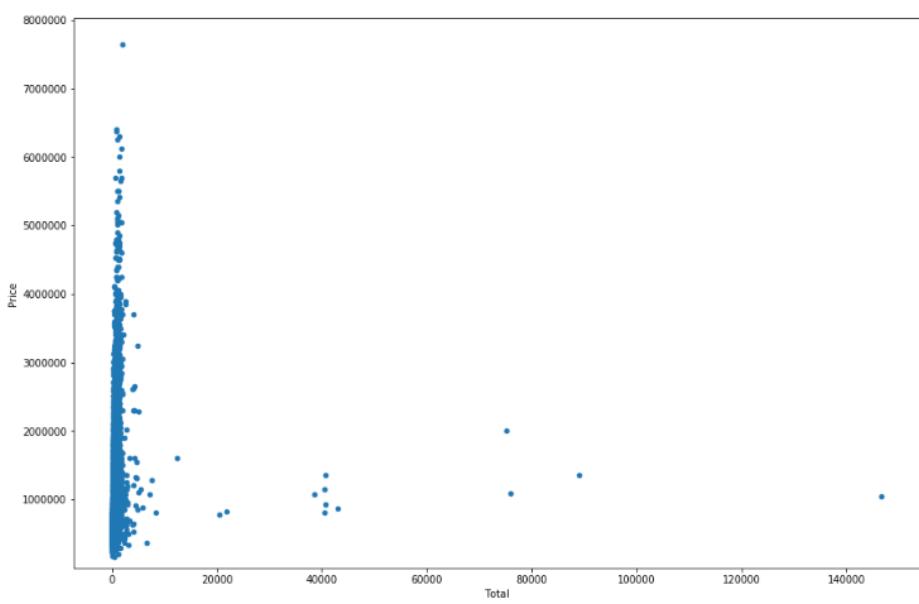
**Şekil 5.14** Price - Rooms İlişkisi

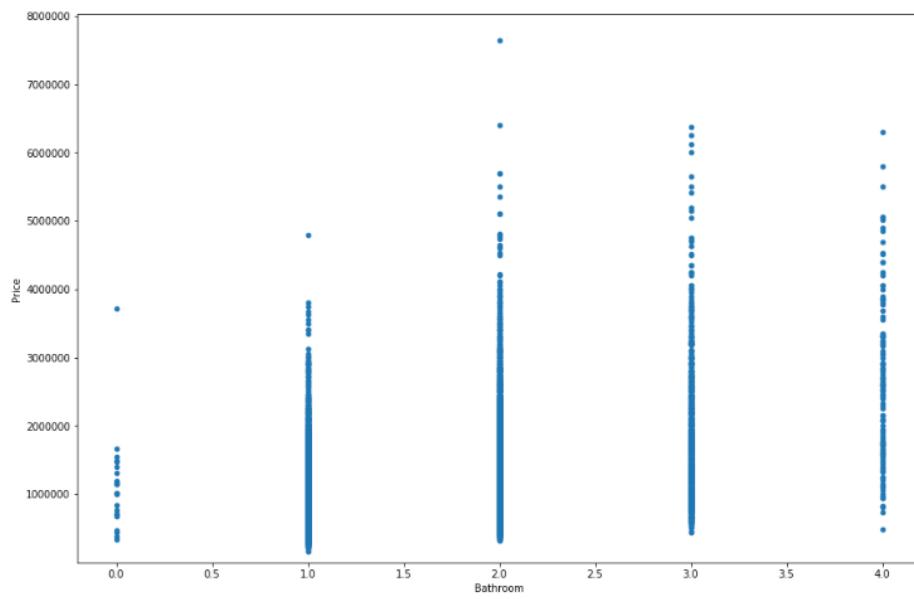


**Şekil 5.15** Price - Type İlişkisi

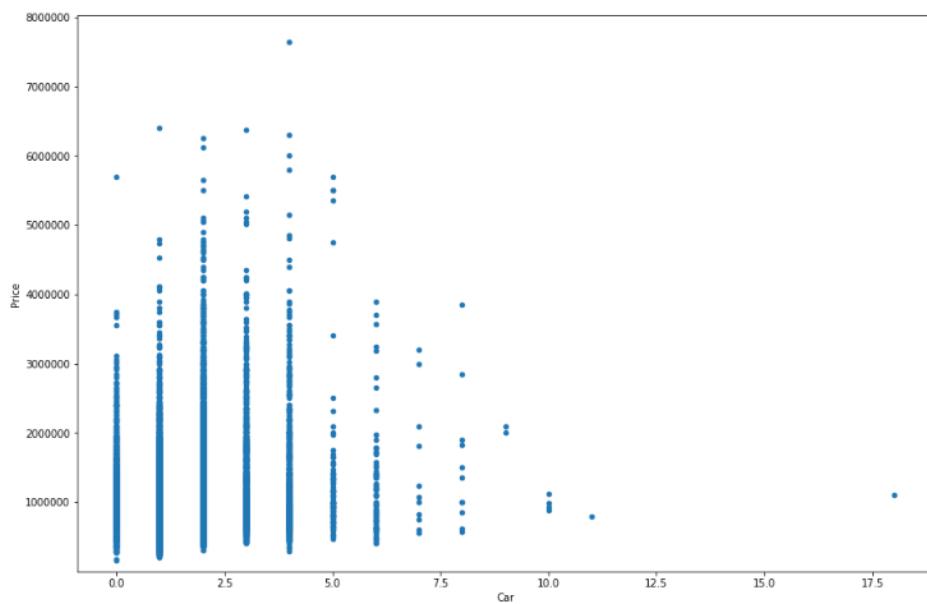


Şekil 5.16 Price - Distance İlişkisi

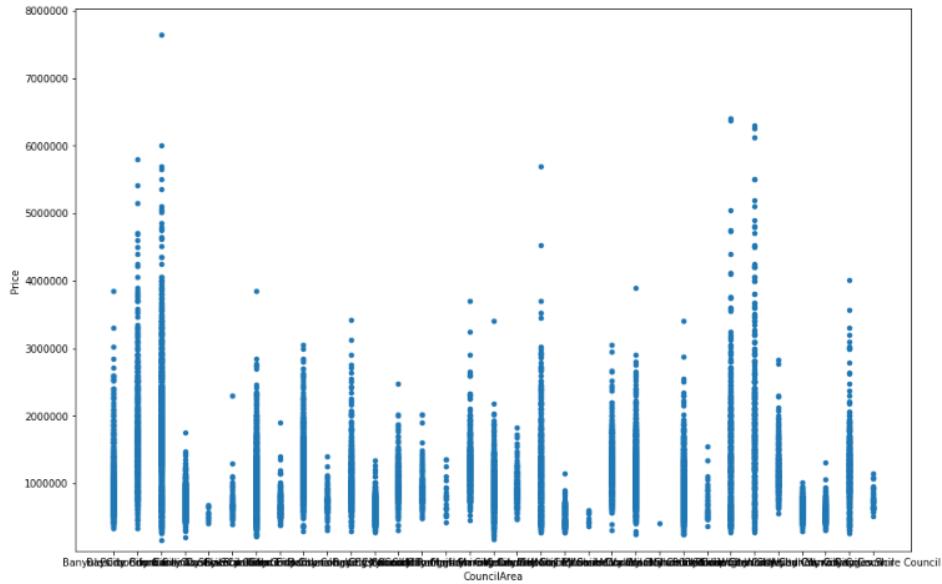




**Şekil 5.18** Price - Bathroom İlişkisi



**Şekil 5.19** Price - Car İlişkisi



**Şekil 5.20** Price - CouncilArea İlişkisi

Veri setimizin son halini Şekil 5.21

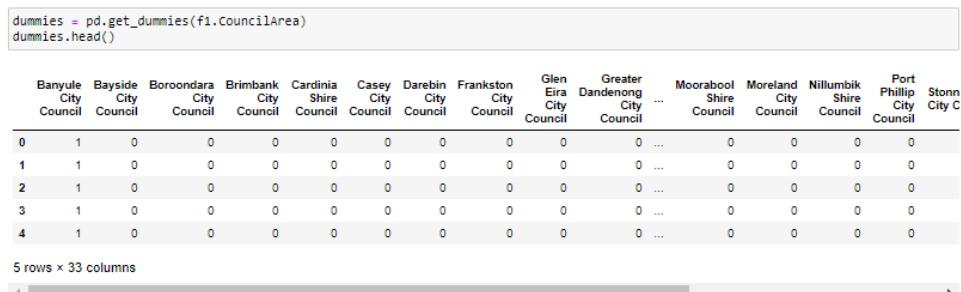
Out[335]:

	Id	Rooms	Type	Price	Distance	Bathroom	Car	CouncilArea	Total
0	3719	5	h	1901000.0	2	2.0	2.0	Banyule City Council	1188.0
1	3721	4	h	1650000.0	2	3.0	2.0	Banyule City Council	1051.0
2	3726	3	h	1816000.0	2	1.0	2.0	Banyule City Council	1026.0
3	3727	2	u	525000.0	2	1.0	1.0	Banyule City Council	149.0
4	3728	4	h	2050000.0	2	3.0	2.0	Banyule City Council	1013.0

**Şekil 5.21** Veri Setinin Son Hali

### 5.2.2 Veri Setinin Model İçin Hazırlanması

Veri setinin son halinde elimizde kategorik veriler bulunmaktadır. Bu kategorik verileri getdummies [3] adlı pandas fonksiyonu ile random göstergelerle çevireceğiz. İlk olarak CouncilArea alanını Şekil 5.22 görüldüğü üzere dummies isimli değişkende tutacağımız.



**Şekil 5.22** CouncilArea Alanının Sayısal Alana Çevrilmesi

Daha sonra Şekil 5.23 de görüldüğü üzere Type alanını dummies2 isimli değişkende tutuyor olacağız.

```
: dummies2 = pd.get_dummies(f1.Type)
dummies2.head()
```

	<b>h</b>	<b>t</b>	<b>u</b>
0	1	0	0
1	1	0	0
2	1	0	0
3	0	0	1
4	1	0	0

**Şekil 5.23** Type Alanının Sayısal Alana Çevrilmesi

İki farklı değişken içindeki bu alanlar ve asıl veri setimizi tek bir değişkende toplamak için ise yine bir Pandas fonksiyonu olan concat isimli fonksiyonu kullanacağız.

```
f2 = pd.concat([f1,dummies2],axis='columns')
f2.head()
```

Type	Price	Distance	Bathroom	Car	CouncilArea	Total	Banyule City Council	... Port Phillip City Council	Stonnington City Council	Whitehorse City Council	Whittlesea City Council	Wyndham City Council	Yarra City Council	Yarra Ranges Shire Council	<b>h</b>	<b>t</b>	<b>u</b>	
h 1901000.0	2	2.0	2.0	Banyule City Council	1188.0	1	...	0	0	0	0	0	0	0	0	1	0	0
h 1650000.0	2	3.0	2.0	Banyule City Council	1051.0	1	...	0	0	0	0	0	0	0	0	1	0	0
h 1816000.0	2	1.0	2.0	Banyule City Council	1028.0	1	...	0	0	0	0	0	0	0	0	1	0	0
u 525000.0	2	1.0	1.0	Banyule City Council	149.0	1	...	0	0	0	0	0	0	0	0	0	0	1
h 2050000.0	2	3.0	2.0	Banyule City Council	1013.0	1	...	0	0	0	0	0	0	0	0	1	0	0

**Şekil 5.24** Tüm Alanların Tek Bir Değişken İçinde Birleştirilmesi

Artık CouncilArea ve Type alanlarını veri setimizden silebiliriz. Ayrıca alaksız özellik olan Id değerininide siliyoruz. temp isimli değişkende ise sonradan kullanılmak üzere ayrıca bir kopya alınıyor.

```

temp = f2.drop(['Type','CouncilArea'],axis = 'columns')
f3 = f2.drop(['Id','Type','CouncilArea'],axis = 'columns')
f3.head()

```

	Rooms	Price	Distance	Bathroom	Car	Total	Banyule City Council	Bayside City Council	Boroondara City Council	Brimbank City Council	...	Port Phillip City Council	Stonnington City Council	Whitehorse City Council	Whittlesea City Council	Wyndham City Council
0	5	1901000.0	2	2.0	2.0	1188.0	1	0	0	0	...	0	0	0	0	0
1	4	1650000.0	2	3.0	2.0	1051.0	1	0	0	0	...	0	0	0	0	0
2	3	1816000.0	2	1.0	2.0	1026.0	1	0	0	0	...	0	0	0	0	0
3	2	5250000.0	2	1.0	1.0	149.0	1	0	0	0	...	0	0	0	0	0
4	4	2050000.0	2	3.0	2.0	1013.0	1	0	0	0	...	0	0	0	0	0

5 rows × 42 columns

Şekil 5.25 Id - CouncilArea - Type Alanlarının Silinmesi

### 5.3 Makine Öğrenmesi Algoritmaları

Yaygın olarak kullanılan açık kaynak kodlu makine öğrenmesi kütüphanesi olan Scikit-Learn(sklearn) [4] ile işlemlerimize devam edeceğiz.

İlk olarak modele gireceğimiz olan veri setinin içinde Price değeri olmamalı. Price değerini göz ardı edip X isimli değişken içinde geri kalan tüm verileri depoluyoruz. Ardından Price değerini y isimli değişkende tutuyoruz. Şekil 5.26

```

X = f3.drop(['Price'],axis='columns')
X.head(3)

```

	Rooms	Distance	Bathroom	Car	Total	Banyule City Council	Bayside City Council	Boroondara City Council	Brimbank City Council	Port Phillip City Council	Stonnington City Council	Whitehorse City Council	Whittlesea City Council	Wyndham City Council
0	5	2	2.0	2.0	1188.0	1	0	0	0	0	0	0	0	0
1	4	2	3.0	2.0	1051.0	1	0	0	0	0	0	0	0	0
2	3	2	1.0	2.0	1026.0	1	0	0	0	0	0	0	0	0

3 rows × 41 columns

```

y = f3.Price
y.head(3)

```

```

0    1901000.0
1    1650000.0
2    1816000.0
Name: Price, dtype: float64

```

Şekil 5.26 X ve y Değişkenleri

Şimdi tahmin oluşturmak için sklearn tarafından bize sunulan traintestsplit methodu ile test verilerimizi oluşturalım.

Burda traintestsplit işleminden kısaca bahsedicek olursak; dizilerimizi veya matrislerimizi rastgele subsetlere ve test datalarına dönüştürüyoruz.

```
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=10)
```

Şekil 5.27 traintestsplit Metodu

### 5.3.1 Linear Regression

LinearRegression nesnesi ile modelimizi oluşturalım ve train datamız ile de modelimizi besleyelim. Ardından score değerimizin, hata payımızın RMSE [5] olarak öğrenelim.

```
from sklearn import linear_model  
from sklearn import ensemble  
lr = linear_model.LinearRegression()  
model = lr.fit(X_train, y_train)  
  
print ("R^2 is: \n", model.score(X_test, y_test))  
predictions = model.predict(X_test)  
  
R^2 is:  
0.6117386364935725
```

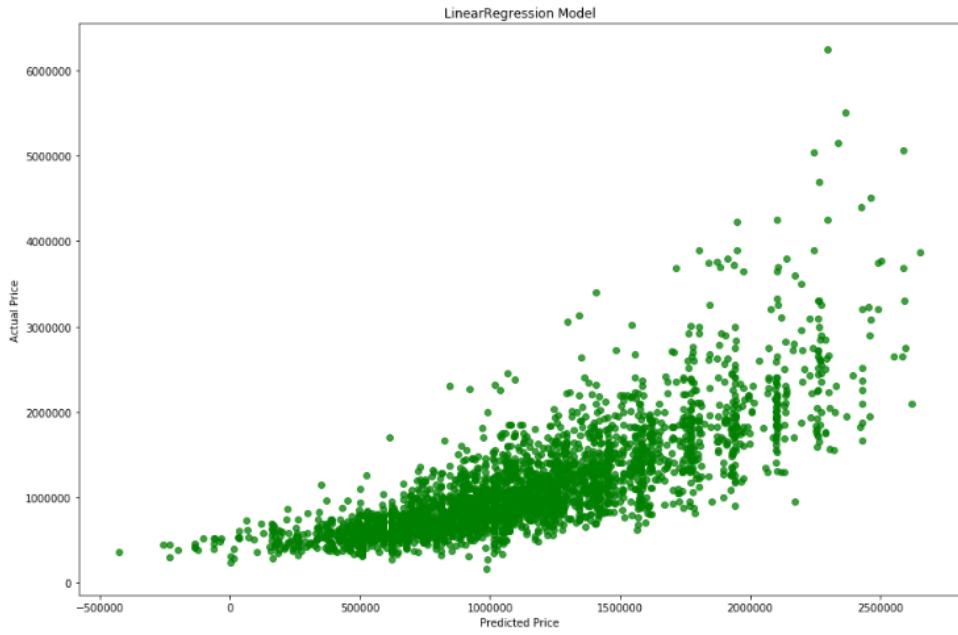
Şekil 5.28 LinearRegression nesnesi ve RMSE Değeri

**RMSE:** Bir makine öğrenmesi modelinin, tahminleyicinin tahmin ettiği değerler ile gerçek değerleri arasındaki uzaklığın bulunmasında sıkılıkla kullanılan, hatanın büyüklüğünü ölçen kuadratik bir metriktir. RMSE tahmin hatalarının (kalıntıların) standart sapmasıdır. Yani, kalıntılar, regresyon hattının veri noktalarından ne kadar uzakta olduğunun bir ölçüsüdür.

Şimdi bir grafik üzerinde nasıl bir sonuç alındığını görüntüleyelim.

```
actual_values = y_test  
plt.scatter(predictions, actual_values, alpha=.75,  
            color='g') #alpha helps to show overlapping data  
plt.xlabel('Predicted Price')  
plt.ylabel('Actual Price')  
plt.title('LinearRegression Model')
```

Şekil 5.29 Tahmin Verilerinin Görselize Edilmesi İçin Fonksiyon



**Şekil 5.30 Tahmin Verilerinin Görselize Edilmesi**

### 5.3.2 GridSearchCV

GridSearchCV üzerinden parametrelere göre en uygun model için bir seçme işlemi yapalım. Linear Regression, Lasso Regression ve DecisionTreeRegressor arasından en iyi seçim için bir fonksiyon oluşturalım.

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor
def find_best_model_using_gridsearchcv(X,y):
    algos = {
        'linear_regression' : {
            'model': LinearRegression(),
            'params': {
                'normalize': [True, False]
            }
        },
        'lasso': {
            'model': Lasso(),
            'params': {
                'alpha': [1,2],
                'selection': ['random', 'cyclic']
            }
        },
        'decision_tree': {
            'model': DecisionTreeRegressor(),
            'params': {
                'criterion' : ['mse','friedman_mse'],
                'splitter': ['best','random']
            }
        }
    }
    scores = []
    cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
    for algo_name, config in algos.items():
        gs = GridSearchCV(config['model'], config['params'], cv=cv, return_train_score=False)
        gs.fit(X,y)
        scores.append({
            'model': algo_name,
            'best_score': gs.best_score_,
            'best_params': gs.best_params_
        })
    return pd.DataFrame(scores,columns=['model','best_score','best_params'])

find_best_model_using_gridsearchcv(X,y)
```

**Şekil 5.31 GridSearch İçin Fonksiyon**

Tablo 5.1 de en farklı modellerin scoreları görülmektedir.

model	best_score	best_param
linear_regression	0.600619	{'normalize': False}
lasson	0.600605	{'alpha': 2, 'selection': 'cyclic'}
decision_tree	0.459135	{'criterion': 'friedman_mse', 'splitter': 'ran...}

**Tablo 5.1** GridSearchCV Best Decision

### 5.3.3 Ridge Regressor

Şimdi veri setimizi Ridge Regressor modele giydirelim. Ve neticesinde RMSE değerini görelim. 5 farklı öğrenme değeri üzerinden işlem yürüteceğiz.

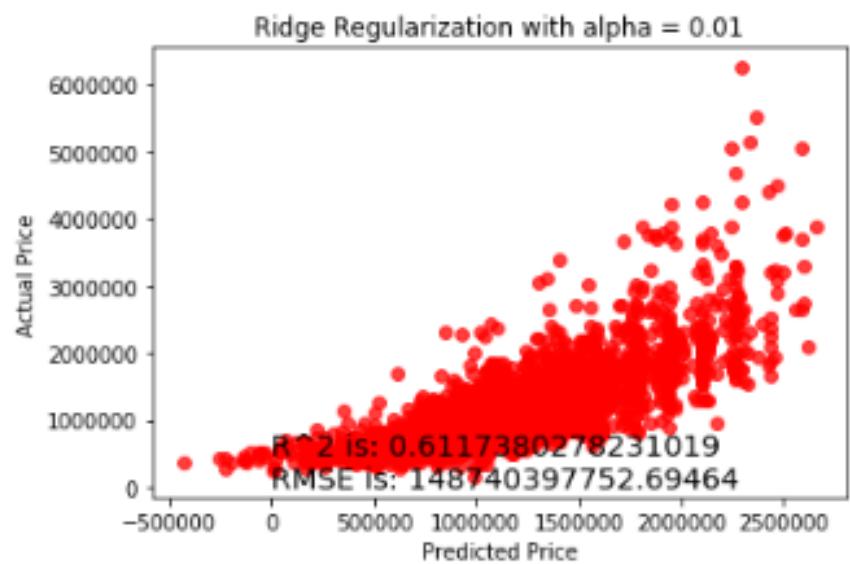
- alpha 0.01
- alpha 0.1
- alpha 1
- alpha 0.10
- alpha 0.100

Şimdi linear model'i sklearn kütüphanesinden import edip Ridge Regressor model için fonksiyon oluşturuluyor.

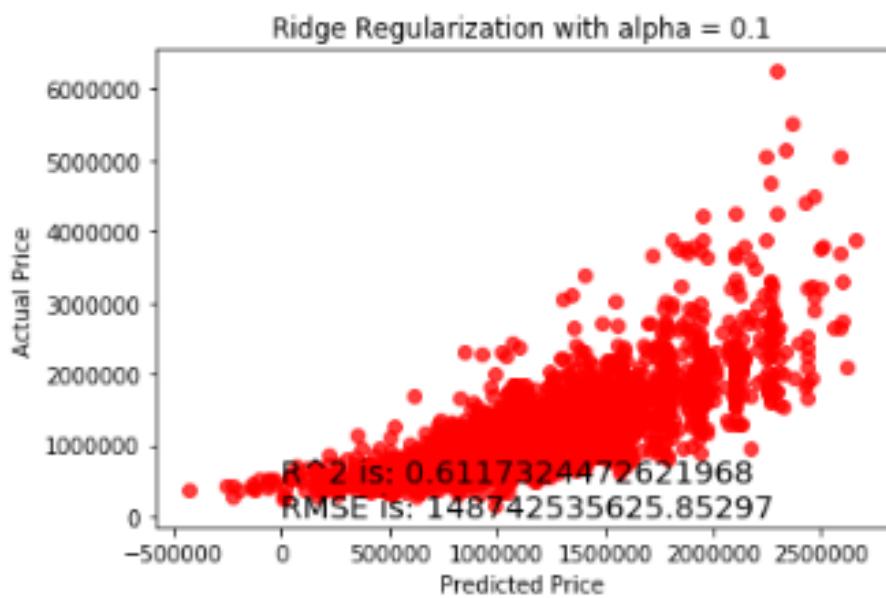
```
from sklearn import linear_model
%matplotlib inline
for i in range (-2, 3):
    alpha = 10**i
    rm = linear_model.Ridge(alpha=alpha)
    ridge_model = rm.fit(X_train, y_train)
    preds_ridge = ridge_model.predict(X_test)
    actual_values = y_test
    plt.scatter(preds_ridge, actual_values, alpha=.75, color='r')
    plt.xlabel('Predicted Price')
    plt.ylabel('Actual Price')
    plt.title('Ridge Regularization with alpha = {}'.format(alpha))
    overlay = 'R^2 is: {}\nRMSE is: {}'.format(
        ridge_model.score(X_test, y_test),
        mean_squared_error(y_test, preds_ridge))
    plt.annotate(s=overlay, xy=(12.1, 10.6), size='x-large')
plt.show()
```

**Şekil 5.32** Ridge Regressor İçin Fonksiyon

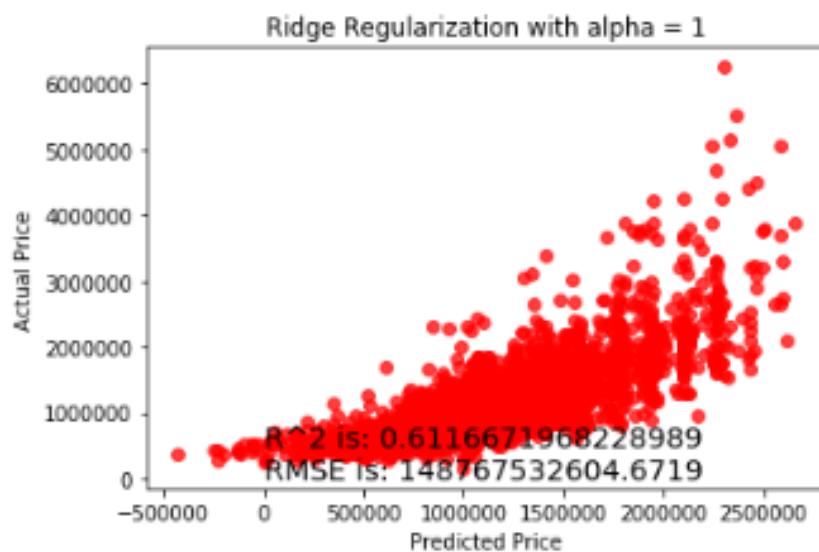
Bu fonksiyon neticesinde alpha değerlerine göre aşağıdaki çıktıları görselize bir biçimde görüyoruz.



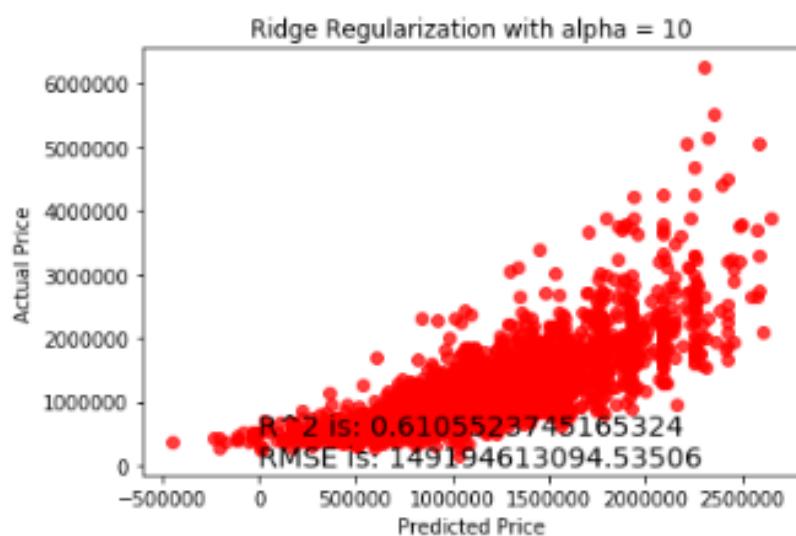
Şekil 5.33 alpha 0.01



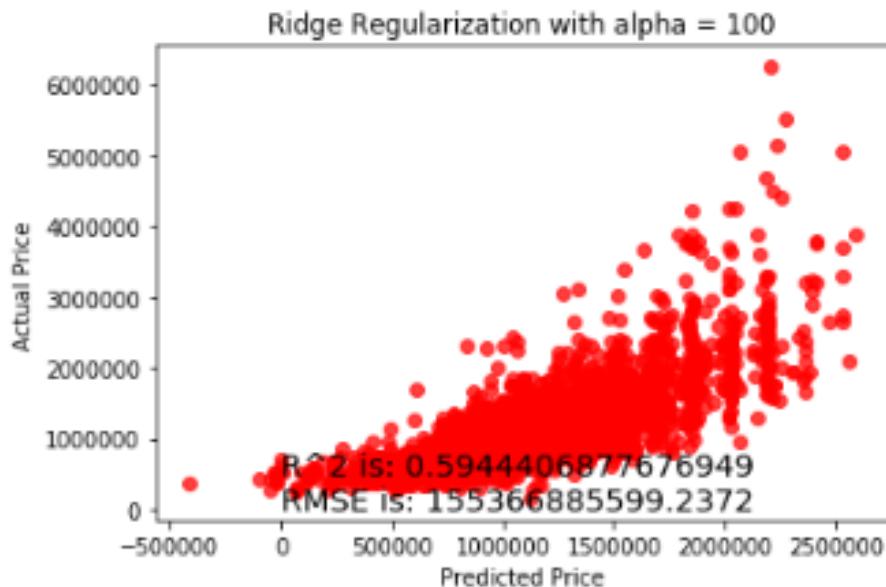
Şekil 5.34 alpha 0.1



Şekil 5.35 alpha 1



Şekil 5.36 alpha 10



**Şekil 5.37 alpha 100**

## 5.4 Yapay Sinir Ağları

### 5.4.1 ANN 2 Layers

Makine öğrenmesi teknilerinden biri olan derin öğrenmeyi projemize entegre etmeye çalışalım. Kastedilen şey yapay sinir ağlarından başka bir şey değil. Yapacağımız şey Keras adlı açık kaynak kodlu kütüphaneyi dahil etmek.

Şu an zaten elimizde düzenlenmiş bir veri seti var. Bu veri setini tekrar ayarlamak gerekecek. Veri setini ilk olarak iki parçaya bölüyoruz. Şekil 5.38 de train data ve Şekil 5.39 de test data görülmektedir. Bu data elimizde olan düzenlenmiş veri setinden alınarak 2 parçaya bölünmüştür.

Id	Rooms	Price	Distance	Bathroom	Car	Total	Banyule C	Bayside C	Boroonda	Brimbank	Cardinia S	Casey S	Darev
							1	0	0	0	0	0	0
3719	5	1901000	2	2	2	1188	1	0	0	0	0	0	0
3721	4	1650000	2	3	2	1051	1	0	0	0	0	0	0
3726	3	1816000	2	1	2	1026	1	0	0	0	0	0	0
3727	2	525000	2	1	1	149	1	0	0	0	0	0	0
3728	4	2050000	2	3	2	1013	1	0	0	0	0	0	0
3729	5	1930000	2	2	1	1175	1	0	0	0	0	0	0
3730	2	2850000	2	1	1	1777	1	0	0	0	0	0	0
3731	4	2600000	2	2	3	1193	1	0	0	0	0	0	0
3733	2	600000	2	1	1	207	1	0	0	0	0	0	0
3735	3	1601000	2	1	1	1008	1	0	0	0	0	0	0
3736	3	1100000	2	2	1	855	1	0	0	0	0	0	0
3738	4	2550000	2	2	2	1962	1	0	0	0	0	0	0
3741	4	2012000	2	3	2	842	1	0	0	0	0	0	0
3744	4	2309000	2	3	2	1468	1	0	0	0	0	0	0
5365	2	815000	2	1	2	709	1	0	0	0	0	0	0
5368	3	720000	2	1	0	505	1	0	0	0	0	0	0
5369	4	1370000	2	2	4	481	1	0	0	0	0	0	0
5370	2	757000	2	2	2	275	1	0	0	0	0	0	0
5371	3	740600	2	2	2	337	1	0	0	0	0	0	0
5372	3	580000	2	1	1	577	1	0	0	0	0	0	0
5373	2	565000	2	2	1	308	1	0	0	0	0	0	0
5301	4	750000	2	2	2	510	1	0	0	0	0	0	0

Şekil 5.38 Train Data

Id	Rooms	Distance	Bathroom	Car	Total	Banyule C	Bayside C	Boroonda	Brimbank	Cardinia S	Casey S	Darev
						0	0	0	0	0	0	0
19636	3	2	1	2	520	0	0	0	0	0	0	0
19638	4	2	1	2	870	0	0	0	0	0	0	0
19640	2	2	1	2	519	0	0	0	0	0	0	0
19641	3	2	2	1	536	0	0	0	0	0	0	0
19642	3	2	1	1	540	0	0	0	0	0	0	0
19643	2	2	1	1	363	0	0	0	0	0	0	0
19708	3	2	2	2	333	0	0	0	0	0	0	0
19709	3	2	2	2	339	0	0	0	0	0	0	0
19747	4	2	3	2	720	0	0	0	0	0	0	0
19878	4	2	1	1	560	0	0	0	0	0	0	0
19879	4	2	1	1	452	0	0	0	0	0	0	0
19882	3	2	1	4	601	0	0	0	0	0	0	0
19884	4	2	1	5	565	0	0	0	0	0	0	0
19957	3	2	2	1	409	0	0	0	0	0	0	0
20156	2	2	1	0	270	0	0	0	0	0	0	0
20157	4	2	3	4	422	0	0	0	0	0	0	0
20193	4	2	1	6	953	0	0	0	0	0	0	0
20194	3	2	2	2	150	0	0	0	0	0	0	0
20195	3	2	2	3	623	0	0	0	0	0	0	0
20197	3	2	1	1	298	0	0	0	0	0	0	0
20198	3	2	1	2	741	0	0	0	0	0	0	0
20100	2	2	1	1	550	0	0	0	0	0	0	0

Şekil 5.39 Test Data

Şekil 5.40 de görüldüğü gibi daha sonra bu iki veri setini birleştirip, id alanını sonradan çıktı alırken sıra bozulmaması için bir değişken içerisinde depolayacağız.

```

train=pd.read_csv("train.csv")
test=pd.read_csv("test.csv")

data = pd.concat([train, test],axis=0, sort='False', ignore_index = True)

data = data[data.columns.difference(['Id'])]

ids = test["Id"]

```

Şekil 5.40 Veri Birleşimi ve Id Değerinin Saklanması

Daha sonra veriyi train ve test data olmak üzere ikiye ayırıyoruz.

```
train = data.iloc[:4000,:] #Upto 4000 rows from first
test = data.iloc[4000:,:] # From 4000 row to last
X_train = train[train.columns.difference(['Price'])].values
y_train = train[['Price']].values
X_test = test[test.columns.difference(['Price'])].values
```

Şekil 5.41 Verinin İkiye Ayrılması

Değerlerin aynı aralığa veya aynı ölçüye yerleştirilmesi ve böylece hiçbir değişkenin diğerine baskın olmaması için feature scaling yapıyoruz.

```
from sklearn.preprocessing import StandardScaler, PowerTransformer
pt_X = PowerTransformer(method='yeo-johnson', standardize=False)
sc_y = StandardScaler()
sc_X = StandardScaler()
y_train = sc_y.fit_transform(y_train)
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

Şekil 5.42 Feature Scaling

Sırada modelimizi başlatmak var.Daha sonra katmanlarını ekliyoruz.(5.43)

```
#Initialising the ANN
model = model()
#Adding the input Layer and first hidden Layer
model.add(Dense(units =400, kernel_initializer='random_uniform', activation= 'tanh',
                input_dim=X_train.shape[1]))
#Add the second hidden Layer
model.add(Dense(units =400, kernel_initializer='random_uniform', activation= 'tanh'))
#Add the second hidden Layer
model.add(Dense(units =10, kernel_initializer='random_uniform', activation= 'relu'))
#The output Layer
model.add(Dense(units =1, kernel_initializer='random_uniform', activation= 'elu'))
```

Şekil 5.43 Model Initialize ve Katman Ekleme ANN 2 Layers

Daha sonra ağırlık güncelleme değeri epoch 150 yapılarak işleme başlıyoruz.

```
#Compiling the ANN
opt = keras.optimizers.Adam(lr=0.0015, beta_1=0.9, beta_2=0.999, decay=0.0, amsgrad=False)
model.compile(optimizer=opt, loss='mean_squared_logarithmic_error', metrics=['mse'])
#fitting the ANN to the training set
model_filepath = 'min_v1_model.h5'
checkpoint = ModelCheckpoint(model_filepath, monitor = 'val_loss', verbose=1, save_best_only = True, mode='min')
model.fit(X_train,y_train, validation_split=0.07, batch_size=32, nb_epoch=150, callbacks=[checkpoint])
model.load_weights(model_filepath)
```

Şekil 5.44 ANN 2 Layers Fitting

Tahmin yapma işlemi Şekil 5.45 de uygulanıyor. Daha sonra Şekil 5.46 da tahmin edilen değerler görülüyor.

```

y_pred = model.predict(X_test)

y_pred = sc_y.inverse_transform(y_pred)

```

**Şekil 5.45 ANN 2 Katmanlı Tahmin İşlemi**

```
y_pred.head()
```

	<b>Id</b>	<b>SalePrice</b>
<b>0</b>	19636.0	2908567.50
<b>1</b>	19638.0	3139891.25
<b>2</b>	19640.0	3411094.50
<b>3</b>	19641.0	2307300.25
<b>4</b>	19642.0	3184729.25

**Şekil 5.46 ANN 2 Katmanlı Tahmin Değerleri**

#### 5.4.2 ANN 3 Layers

3 Katmanlı olarak aynı işlemleri tekrar edelim.

```

from keras.initializers import he_normal
from keras.layers.normalization import BatchNormalization

#Initialising the ANN
model = Sequential()
#Adding the input Layer and first hidden Layer
model.add(Dense(units =300, kernel_initializer=he_normal(seed=None), activation= 'tanh',
                input_dim=X_train.shape[1]))

model.add(Dropout(0.2))

#Add the first hidden Layer
model.add(Dense(units =100, kernel_initializer=he_normal(seed=None), activation= 'tanh'))
model.add(Dropout(0.2))

#Add the second hidden Layer
model.add(Dense(units =80, kernel_initializer=he_normal(seed=None), activation= 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

#Add the third hidden Layer
model.add(Dense(units =50, kernel_initializer=he_normal(seed=None), activation= 'relu'))
model.add(BatchNormalization())

#The output Layer
model.add(Dense(units =1, kernel_initializer=he_normal(seed=None), activation= 'elu'))

```

**Şekil 5.47 Model Initialize ve Katman Ekleme ANN 3 Layers**

```

#Compiling the ANN
opt = keras.optimizers.Adam(lr=0.0015, beta_1=0.9, beta_2=0.999, decay=0.0, amsgrad=False)
model.compile(optimizer=opt, loss='mean_squared_logarithmic_error', metrics=['mse'])
#Fitting the ANN to the training set
model_filepath = 'min_v1_model3.h5'
checkpoint = ModelCheckpoint(model_filepath, monitor = 'val_loss', verbose=1, save_best_only = True, mode='min' )
model.fit(X_train,y_train, validation_split=0.07, batch_size=32, nb_epoch=200, callbacks=[checkpoint])
model.load_weights(model_filepath)

Train on 3719 samples, validate on 281 samples
Epoch 1/200
3719/3719 [=====] - 2s 471us/step - loss: 0.1018 - mse: 0.6270 - val_loss: 4.9888e-04 - val_mse: 0.532

Epoch 00001: val_loss improved from inf to 0.00050, saving model to min_v1_model3.h5
Epoch 2/200
3719/3719 [=====] - 0s 91us/step - loss: 0.0710 - mse: 0.4060 - val_loss: 4.9888e-04 - val_mse: 0.635

Epoch 00002: val_loss did not improve from 0.00050
Epoch 3/200
3719/3719 [=====] - 0s 100us/step - loss: 0.0588 - mse: 0.3595 - val_loss: 4.9888e-04 - val_mse: 0.635

Epoch 00003: val_loss did not improve from 0.00050
Epoch 4/200
3719/3719 [=====] - 0s 131us/step - loss: 0.0534 - mse: 0.3314 - val_loss: 4.9888e-04 - val_mse: 0.635

```

**Şekil 5.48** ANN 3 Layers Fitting and Compile

```

y_pred = model.predict(X_test)

y_pred = sc_y.inverse_transform(y_pred)

```

**Şekil 5.49** ANN 3 Katmanlı Tahmin İşlemi

y_pred.head()	
	Id      SalePrice
0	19636.0 8.815904e+05
1	19638.0 7.723262e+05
2	19640.0 8.312768e+05
3	19641.0 1.143470e+06
4	19642.0 7.975538e+05

**Şekil 5.50** ANN 3 Katmnalı Tahmin Değerleri

### 5.4.3 ANN 5 Layers

5 Katmanlı olarak aynı işlemleri tekrar edelim.

```

from keras.initializers import he_normal
from keras.layers.normalization import BatchNormalization

#Initialising the ANN
model = Sequential()
#Adding the input layer and first hidden Layer
model.add(Dense(units =150, kernel_initializer=he_normal(seed=None), activation= 'tanh',
                input_dim=X_train.shape[1]))

#Add the first hidden Layer
model.add(Dense(units =100, kernel_initializer=he_normal(seed=None), activation= 'tanh'))
model.add(Dropout(0.2))

#Add the second hidden Layer
model.add(Dense(units =80, kernel_initializer=he_normal(seed=None), activation= 'tanh'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

#Add the third hidden Layer
model.add(Dense(units =50, kernel_initializer=he_normal(seed=None), activation= 'tanh'))

#Add the fourth hidden Layer
model.add(Dense(units =30, kernel_initializer=he_normal(seed=None), activation= 'relu'))
model.add(BatchNormalization())

#Add the fifth hidden Layer
model.add(Dense(units =10, kernel_initializer=he_normal(seed=None), activation= 'relu'))
model.add(BatchNormalization())

#The output Layer
model.add(Dense(units =1, kernel_initializer=he_normal(seed=None), activation= 'elu'))

```

**Şekil 5.51 Model Initialize ve Katman Ekleme ANN 5 Layers**

```

opt = keras.optimizers.Adam(lr=0.0015, beta_1=0.9, beta_2=0.999, decay=0.0, amsgrad=False)
model.compile(optimizer=opt, loss='mean_squared_logarithmic_error', metrics=['mse'])
#Fitting the ANN to the training set
model_filepath = 'min_vl_model4.h5'
checkpoint = ModelCheckpoint(model_filepath, monitor = 'val_loss', verbose=1, save_best_only = True, mode='min' )
history=model.fit(X_train,y_train, validation_split=0.07, batch_size=32, nb_epoch=400, callbacks=[checkpoint])
model.load_weights(model_filepath)

Train on 3719 samples, validate on 281 samples
Epoch 1/400
3719/3719 [=====] - 1s 225us/step - loss: 0.0738 - mse: 0.4496 - val_loss: 0.0018 - val_mse: 0.2010
Epoch 00001: val_loss improved from inf to 0.00177, saving model to min_vl_model4.h5
Epoch 2/400
3719/3719 [=====] - 0s 80us/step - loss: 0.0607 - mse: 0.3749 - val_loss: 0.0068 - val_mse: 0.4105
Epoch 00002: val_loss did not improve from 0.00177
Epoch 3/400
3719/3719 [=====] - 0s 78us/step - loss: 0.0521 - mse: 0.3478 - val_loss: 0.0085 - val_mse: 0.4233
Epoch 00003: val_loss did not improve from 0.00177
Epoch 4/400
3719/3719 [=====] - 0s 79us/step - loss: 0.0503 - mse: 0.3398 - val_loss: 0.0058 - val_mse: 0.3303
Epoch 00004: val_loss did not improve from 0.00177
Epoch 5/400
3719/3719 [=====] - 0s 80us/step - loss: 0.0486 - mse: 0.3365 - val_loss: 0.0095 - val_mse: 0.3969

```

**Şekil 5.52 ANN 5 Layers Fitting and Compile**

```

y_pred = model.predict(X_test)

y_pred = sc_y.inverse_transform(y_pred)

```

**Şekil 5.53 ANN 3 Katmanlı Tahmin İşlemi**

```
y_pred.head()
```

	<b>Id</b>	<b>SalePrice</b>
0	19636.0	1213643.00
1	19638.0	1044929.50
2	19640.0	1304359.00
3	19641.0	1411648.00
4	19642.0	1099935.25

**Sekil 5.54 ANN 5 Katmnalı Tahmin Değerleri**

# 6 UYGULAMA

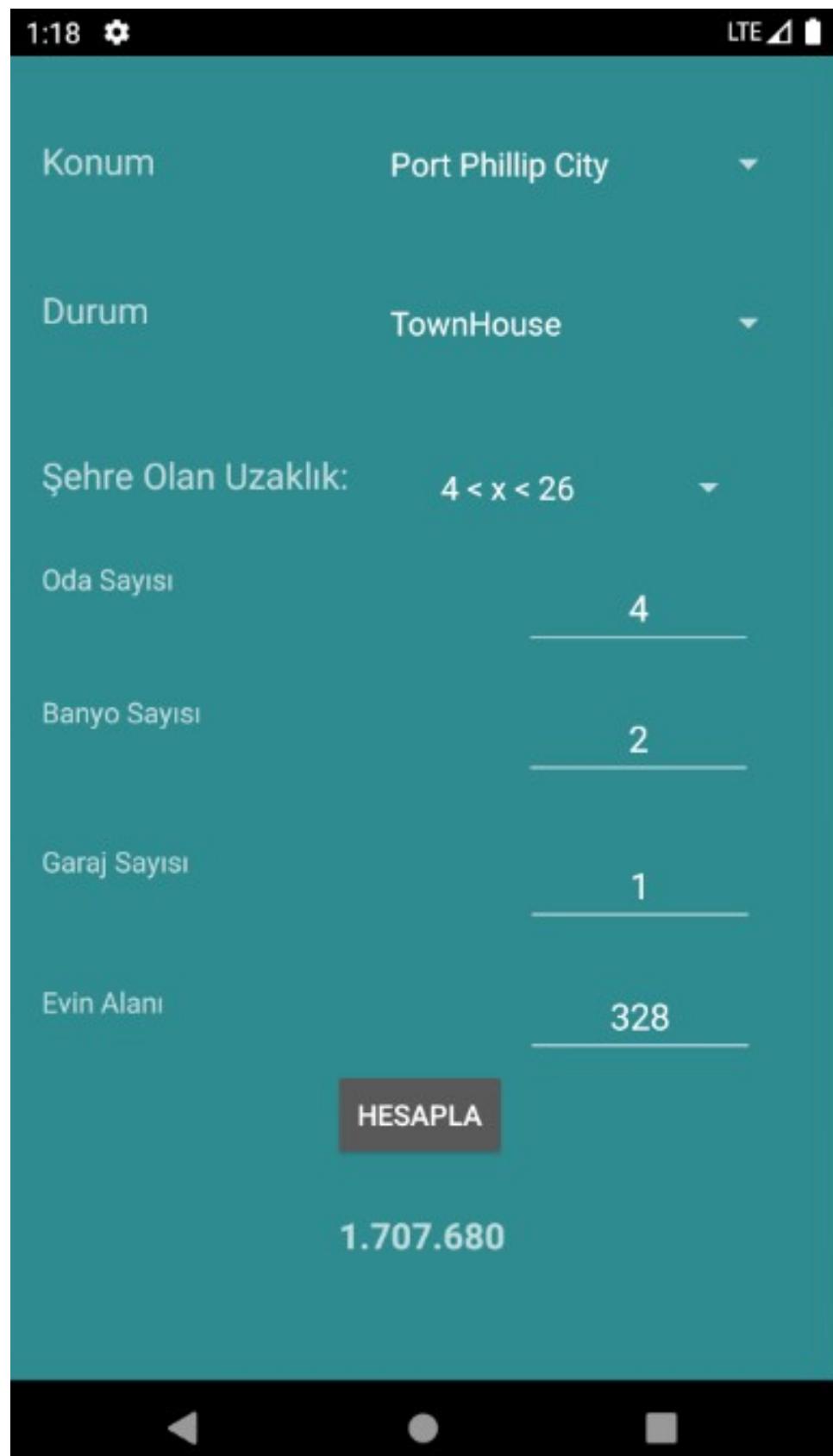
---

Projenin arayüzü kolay kullanım esasına dayalı olarak tasarlanmıştır. Kullanıcı evin konum, şehre olan uzaklık ve durum özelliklerine açılır liste şeklinde ulaşırken oda sayısı, banyo sayısı, garaj sayısı ve evin alanı gibi özellikleri ilgili alanlara girerek belirtecektir.

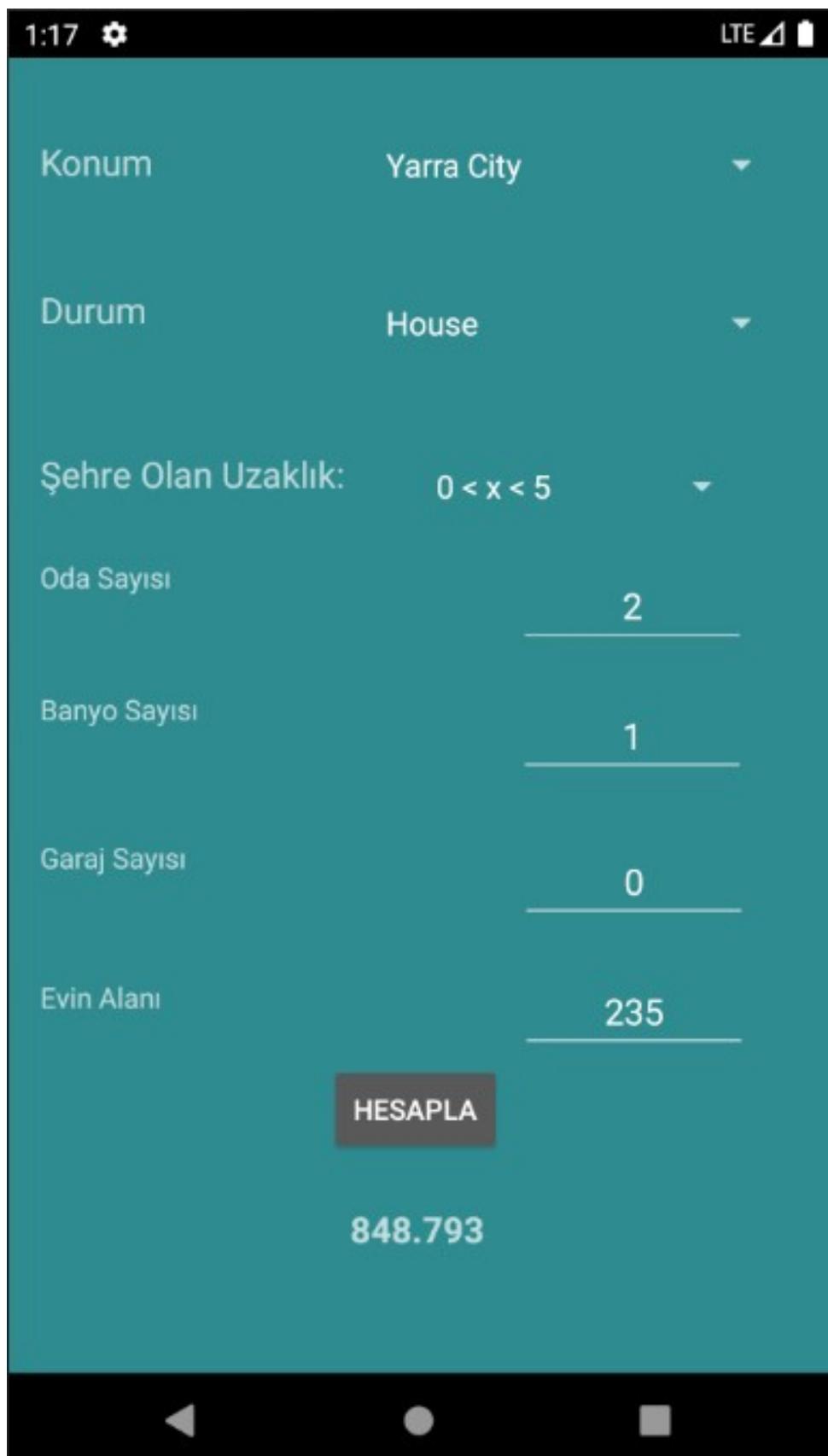


Şekil 6.1 Uygulama Görüntüsü

Kullanıcı arayüz üzerinde fiyatını merak ettiği evin özelliklerini girip hesapla dedikten sonra girdiği özelliklerdeki evin fiyat tahminini ekranında görecektir.



Şekil 6.2 Uygulama Görüntüsü 1



Şekil 6.3 Uygulama Görüntüsü 2

## Referanslar

---

- [1] T. Pino. (2018). Melbourne housing data set, [Online]. Available: <https://www.kaggle.com/anthonypino/melbourne-housing-market>.
- [2] P.D. S. Kul. (2018). Korelasyon analizi, [Online]. Available: <http://www.p005.net/analiz/korelasyon-analizi>.
- [3] R. Langford. (2017). Kategorik alanların değiştirilmesi, [Online]. Available: <https://towardsdatascience.com/the-dummys-guide-to-creating-dummy-variables-f21faddb1d40>.
- [4] (). Scikitlearn, [Online]. Available: <https://scikit-learn.org/stable/about.html>.
- [5] J. Moody. (2019). What does rmse really mean? [Online]. Available: <https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e>.

# Özgeçmiş

---

## BİRİNCİ ÜYE

**İsim-Soyisim:** Atakan TEKOĞLU  
**Doğum Tarihi ve Yeri:** 01.08.1995, İstanbul  
**E-mail:** atakanekoglu@gmail.com  
**Telefon:** 0537 295 82 82  
**Staj Tecrübeleri:** -

## İKİNCİ ÜYE

**İsim-Soyisim:** Süleyman Ali Burak ÇINAR  
**Doğum Tarihi ve Yeri:** 21.02.1998, Kocaeli  
**E-mail:** sabc4129@gmail.com  
**Telefon:** 0506 262 48 62  
**Staj Tecrübeleri:** -

## Proje Sistem Bilgileri

**Sistem ve Yazılım:** Windows İşletim Sistemi, Anaconda, Jupyter Notebook, Android Studio, Visual Studio Code  
**Gerekli RAM:** 8GB  
**Gerekli Disk:** 512MB