Bölüm 3 : Regresyon

3.1. İlk Minik Makine Öğrenmesi Uygulaması

Bir önceki bölümde basit bir makine öğrenmesi uygulaması yapmıştık, ancak uygulamanın amacı bir makine öğrenmesi projesinde hangi adımların izlenmesi gerektiğini göstermekti. Bu sebeple kullanılan makine öğrenme algoritması için hazır modüllerden yararlandık ve nasıl çalıştığı üzerinde durmadık. Şimdi bir makine öğrenme projesinin hazırlık aşamalarında neler yapmamız gerektiğini ve bir model oluşturma, modeli eğitme ve test etme aşamalarının nasıl olacağı hakkında bilgi sahibiyiz.

Bu bölümde regresyonun en popüler uygulamalarından biri olan ev fiyatları tahmini sistemini, bir önceki bölümde öğrendiğimiz adımları uygulayarak ve lineer regresyon algoritmasını kendimiz yazarak geliştireceğiz. Başlayalım.

İlk olarak problem tanımını yapalım. Burada bir hikaye oluşturmaya aslında gerek yok. Ancak bu uygulamanın gerçek hayatta kullanılabileceğini düşünerek şöyle birkaç tane kullanım senaryosu oluşturabiliriz.

1. Yeni bir ev alacağız, bu evin fiyatının bize söylenen kadar edip etmediğini merak ediyoruz.
2. Elimizde bir sürü ev var ve biz bu evleri satacağız. Ancak her ev için en uygun fiyatın ne olduğundan emin değiliz.

Şimdi problem tanımımızın hatlarını biraz daha netleştirelim.

1. 232 metrekarelik bir ev bakıyoruz. Bunun fiyatının ne olacağını merak ediyoruz?
2. Bundan sonra bize gösterilen evlerin fiyatlarının ne olacağını önceden tahmin ederek kandırılıp kandırılmadığımızdan emin olmak istiyoruz.

Problemlerimizi yanıtlabilmek için evlerin büyüklükleri verildiğinde fiyatlarını tahmin edebileceğimiz bir model oluşturacağız. Bunun için evlerin büyüklüklerini ve fiyatlarını içeren bir veri setine ihtiyacımız var. Bu bilgileri içeren açık kaynaklı veri setlerini araştırdığınızda bu uygulama için kullanabileceğiniz birçok veri seti olduğunu görmüşsünüzdür.

Yazacağımız uygulamada kullanacağımız verileri https://wiki.csc.calpoly.edu/datasets/wiki/Houses buradan indirebilirsiniz. Önce

kullanacağımız kütüphaneleri içe aktaralım sonra veri setimizi çalışma alanına yükledikten sonra bir kopyasını oluşturalım.

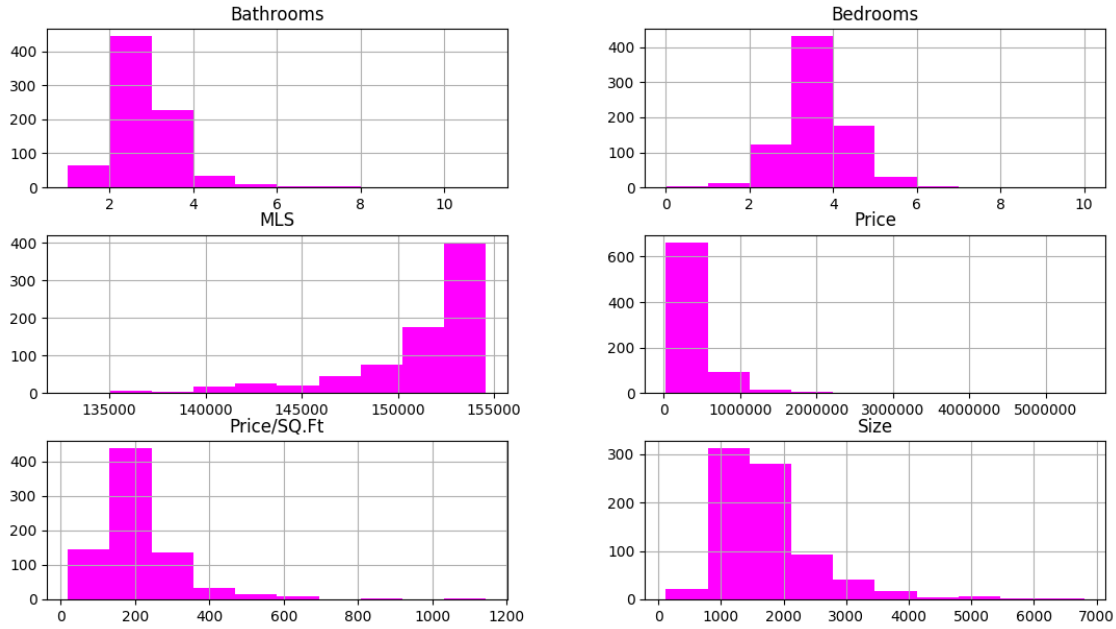
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# https://wiki.csc.calpoly.edu/datasets/wiki/Houses
print '1 - Veri Seti Yukleniyor...'
houses = pd.read_csv('RealEstate.csv')
veri_seti = houses.copy()
print 'HOUSES veri seti yuklendi. Bir kopyasi olusturuldu.'
```

Veri setini anlayabilmek için hızlıca içeriğine bir göz atalım.

```
print '2 - Veriye Hizli Bir Bakis...'
print 'Veri Setinin ilk 5 ornegi :\n', veri_seti.head()
print 'Veri Setinin Ozellikleri :\n', veri_seti.info()
print 'Sayisal Verilerin Ozellikleri :\n', veri_seti.describe()
print 'Sayisal Verilerin Dagilim Grafikleri Gosteriliyor...\n'
veri_seti.hist(color='magenta')
plt.show()
```

```
1 - Veri Seti Yukleniyor...
HOUSES veri seti yuklendi. Bir kopyasi olusturuldu.
2 - Veriye Hizli Bir Bakis...
Veri Setinin ilk 5 ornegi :
   MLS        Location    Price  Bedrooms  Bathrooms  Size \
0  132842     Arroyo Grande  795000.0       3       3  2371
1  134364       Paso Robles  399000.0       4       3  2818
2  135141       Paso Robles  545000.0       4       3  3032
3  135712        Morro Bay  909000.0       4       4  3540
4  136282  Santa Maria-Orcutt  109900.0       3       1  1249


  Price/SQ.Ft     Status
0      335.30  Short Sale
1      141.59  Short Sale
```

```
2     179.75  Short Sale
3     256.78  Short Sale
4      87.99  Short Sale
Veri Setinin Ozellikleri :
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 781 entries, 0 to 780
Data columns (total 8 columns):
MLS           781 non-null int64
Location      781 non-null object
Price         781 non-null float64
Bedrooms      781 non-null int64
Bathrooms     781 non-null int64
Size          781 non-null int64
Price/SQ.Ft   781 non-null float64
Status        781 non-null object
dtypes: float64(2), int64(4), object(2)
memory usage: 48.9+ KB
None
Sayisal Verilerin Ozellikleri :
                MLS         Price    Bedrooms   Bathrooms        Size \
count     781.000000  7.810000e+02  781.000000  781.000000   781.000000
mean   151224.550576  3.833291e+05    3.142125    2.355954  1755.058899
std      3936.122042  3.490381e+05    0.855768    0.846596   819.577603
min    132842.000000  2.650000e+04    0.000000    1.000000   120.000000
25%    149922.000000  1.990000e+05    3.000000    2.000000  1218.000000
50%    152581.000000  2.950000e+05    3.000000    2.000000  1550.000000
75%    154167.000000  4.290000e+05    4.000000    3.000000  2032.000000
max    154580.000000  5.499000e+06   10.000000   11.000000  6800.000000


       Price/SQ.Ft
count   781.000000
mean    213.131293
std     115.082146
min      19.330000
25%     142.140000
50%     188.360000
75%     245.420000
max    1144.640000
Sayisal Verilerin Dagilim Grafikleri Gosteriliyor...
```



Houses veri seti içerisinde özellikleri verilen evler San Luis Obispo (city in the U.S. state of California) ve çevresindeki bölgelere ait. Houses veri setinde 8 özellik bulunuyor. Bunlar, *MLS, Location, Price, Bedrooms, Bathrooms, Size, Price/SQ.Ft, Status.* Toplam 781 tane örnek içeriyor ve hiç eksik değer bulunmuyor. Location ve Status özellikleri dışındakilerin hepsi sayısal veriler içeriyor. Bizim sistemimizi geliştirirken kullanacağımız özellikler **Size** ve **Price**. Fiyatlar dolar cinsinden ve ev büyüklükleri de square foot türünden verilmiş. Şimdilik bunlarla ilgili bir sorun yaşamayacağız. Square foot olan değerleri metrekareye çevirebiliriz ve dolar kullanmak yerine lira cinsinden ifade edebiliriz. İşin özünü kavramamızda bunlar engel değil. Sayısal verilerin özelliklerine baktığımızda, evlerin ortalama 3 yatak odası ve 2 banyosu bulunduğunu ve ortalama 1755 square foot yani 163 metre kare büyüklüğünde olduğunu söyleyebiliriz.

Verilerimizi makine öğrenme algoritmamıza hazırlamaya başlayalım ve bundan sonra veri keşfine çıkalım. Kullanmayacağımız veriler üzerinde araştırma yapmamıza gerek yok.

```
print '3 - Veriyi Hazirlayalim...'
# kullanmayacagimiz ozellikleri temizleyelim.
del veri_seti['MLS']
del veri_seti['Location']
del veri_seti['Bedrooms']
del veri_seti['Bathrooms']
del veri_seti['Price/SQ.Ft']
del veri_seti['Status']
print 'Kullanilmayacak Ozellikler Silindi.'
```

Amerikan ölçülerine alışkın olmadığımız için ev büyüklüklerini metrekare cinsine çevirelim. Burada dönüşümü şöyle yapacağız: 1 square foot = 0.09290304 metrekare.

```
# veri donusumu islemini yapalim.
# 1 square foot = 0.09290304 m2
veri_seti['Size'] = veri_seti['Size'] * 0.09290304
print 'Square Foot degeri Metrekareye Donusturuldu.'


print 'Veri Setinin Yeni Hali :'
print veri_seti.head()
```

```
Square Foot degeri Metrekareye Donusturuldu.
Veri Setinin Yeni Hali :
   Price      Size
0  795000.0  220.273108
1  399000.0  261.800767
2  545000.0  281.682017
3  909000.0  328.876762
4  109900.0  116.035897
```

Verilerimiz artık keşif için hazır. Şimdilik verilerin üzerinde herhangi bir özellik ölçeklendirmesi yapmayacağız. Ev fiyatları ve büyüklükleri arasındaki ilişkiyi inceleyelim. Tabi bunun için aralarında bir ilişki olduğundan emin olmalıyız. Bunun için görselleştirme teknikleri kullanılabilir veya değişkenler arası ilişkiler (korelasyon) incelenebilir. Biz önce korelasyona bakalım sonra da grafikleri inceleyelim. Özelliklerin isimlerini türkçeye çevirelim sonra bir önceki bölümde kullandığımız *corr()* fonksiyonu ile değişkenler arasındaki ilişkiye bakalım.

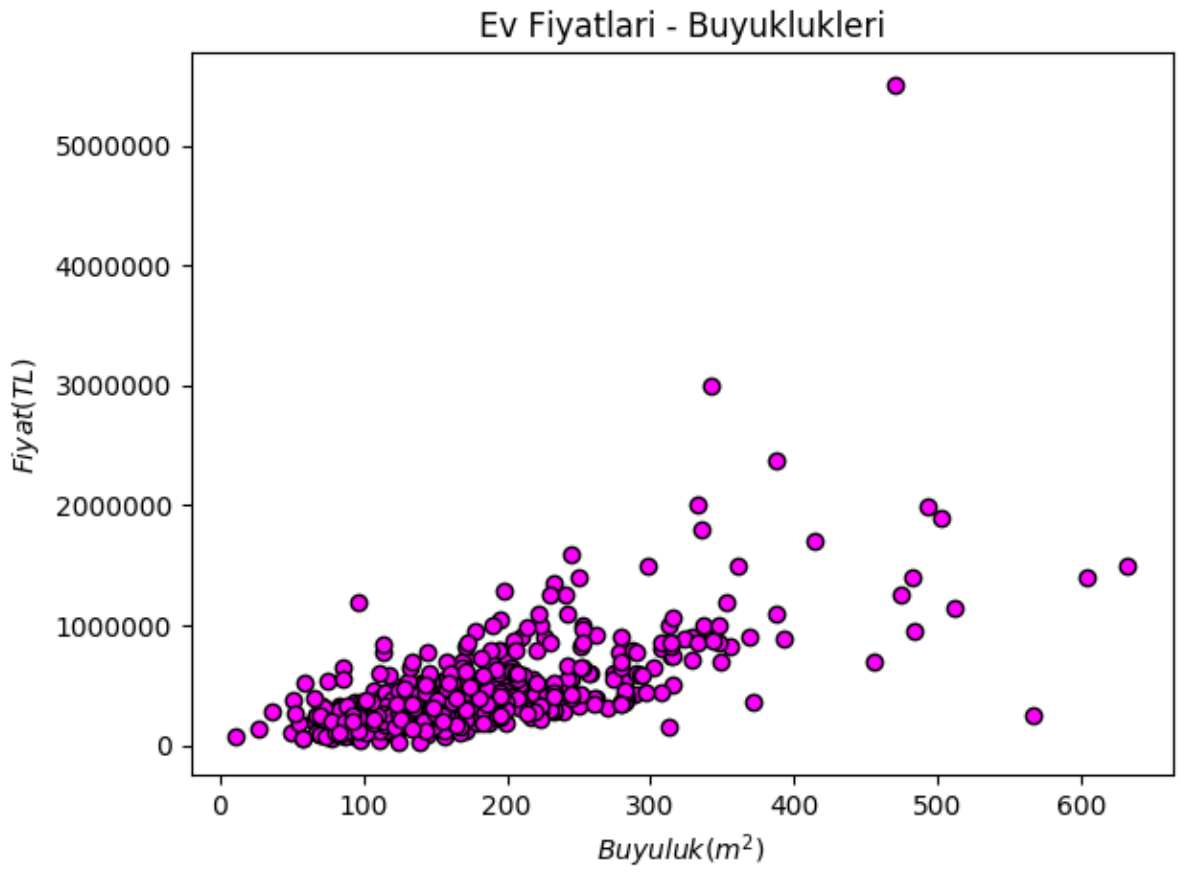
```
# ozelliklerin isimlerini turkceye cevirelim
veri_seti.rename(columns={'Price': 'Fiyat', 'Size': 'Buyukluk'}, inplace=True)


print veri_seti.corr(method='pearson')
```

```
Square Foot degeri Metrekareye Donusturuldu.
          Fiyat  Buyukluk
Fiyat     1.000000  0.664724
```

```
Buyukluk  0.664724  1.000000
```

Fiyat - büyüklük arasındaki katsayıya baktığımızda 0.67 olduğunu görüyoruz bir önceki bölümden hatırlarsanız bu aralarında güçlü bir pozitif ilişki olduğu anlamına gelmekteydi. Aralarında bir ilişki olduğundan artık eminiz.
Şimdi de aralarındaki ilişkiyi görselleştirelim.

```
plt.scatter(x=veri_seti.Buyukluk, y=veri_seti.Fiyat, edgecolors='black',
c='magenta')
plt.title('Ev Fiyatlari - Buyuklukleri')
plt.xlabel(r'$Buyuluk (m^2)$')
plt.ylabel(r'$Fiyat (TL)$')
plt.tight_layout()
plt.show()
```



Fiyat büyüklük dağılım grafiğine baktığımızda evin büyüklüğü arttıkça fiyatında da bir artış olduğunu rahatlıkla görebiliyoruz.

Veri setimiz üzerinde gerekli hazırlık aşamalarını yaptık ve artık verimizi makine öğrenme algoritmamıza hazır hale getirdik. Veri setini test setinin oranını 0.2 olarak ayarlayarak eğitim ve test seti olarak bölelim.

```
# veri setini test ve eğitim olarak ikiye ayiralim.
X_egitim, X_test, y_egitim, y_test = train_test_split(X, y, test_size=0.2,
random_state=False)
```

Veri setimize uygun bir model yaratacağız. Modelimiz aslında çok basit, veri setini en iyi karşılayacak çizgiyi çizeceğiz. Yani, modelimiz eğitim setimize uyan en iyi doğru olacak. Bu sebeple modelimi oluştururken aslında doğru denkleminden yararlanacağız. Doğru denklemini yazarken ihtiyacımız olan veriler, doğrunun eğimi ve y eksenini kesen noktasıdır. Bu iki değeri N veri noktasına sahip bir veri kümesi göz önüne alındığında, eğim ve y'yi kesme noktasını aşağıdakileri kullanarak bulabiliriz:

$$m = \frac{n\sum(xy) - \sum x \sum y}{n\sum(x^2) - (\sum x)^2}$$

$$b = \frac{\sum y - m\sum x}{n}$$

```
# makine ogrenmesi
# modeli olusturalim
# m = [ nE(xy) - ExEy) ]  / [ nE(x^2)-(Ex)^2 ]
# b = ( Ey - mEx ) / n
class lineer_regresyon:
  def __init__(self):
    self.m = 0
    self.b = 0


  def egit(self, _x_, _y_):
    _x_ = np.asarray(_x_)
```

```
    _y_ = np.asarray(_y_)
    n = len(np.asarray(_x_))
    self.m = (n * sum(_x_ * _y_) - sum(_x_) * sum(_y_)) / (n *   sum(_x_ * _x_) - sum(_x_) * sum(_x_))
    self.b = (sum(_y_) - self.m * sum(_x_)) / n


  def tahmin_et(self, _x_test_):
    return self.m * np.asarray(_x_test_) + self.b
```

Veri setlerini *pandas.Series* tipinde alan bir model oluşturduk. Bu modelde yukarıda verilen formüllere göre eğim ve kesme noktası hesaplanıyor. Bu değerlere göre de modeli eğitiyor ve yeni gelen verileri test ediyor. Şimdi modelimi eğitim verilerimizle eğitip, test verilerimizle test etme zamanı. Performans ölçütümüzü yine Ortalama Kare Hata (MSE) olarak seçelim ve test verilerimizi kullanarak bulduğumuz tahminlerdeki hatayı ölçmek için ortalama_kare_hata fonksiyonunu yazalım.

MSE'nin formülünü hatırlayalım.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

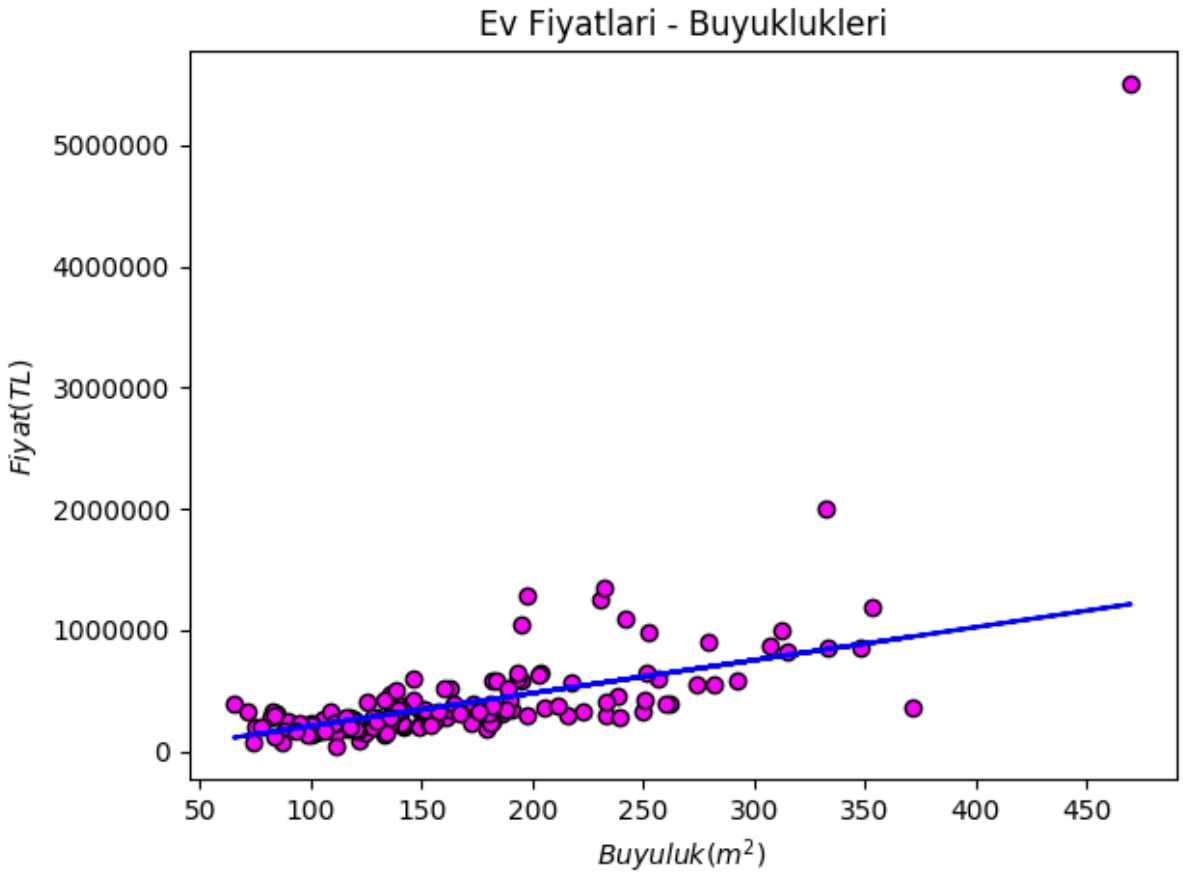
```
# MSE = ( 1 / n ) * ( E(y - yi)^2 )
def ortalama_kare_hata(_y_test_, _y_tahmin_):
  _y_test_ = np.asarray(_y_test_)
  return sum((_y_test_ - _y_tahmin_) ** 2) / len(_y_test_)
```

Modelimizi eğitelim ve test edelim.

```
model = lineer_regresyon()
model.egit(X_egitim, y_egitim)
y_tahmin = model.tahmin_et(X_test)


basari = ortalama_kare_hata(y_test, y_tahmin)
```

Test veri setimize modelimizin ne kadar uyduğunu anlayabilmek için görselleştirelim.

```
plt.scatter(X_test, y_test, c='magenta', edgecolors='black')
plt.plot(X_test, y_tahmin, c='blue')
plt.title('Ev Fiyatlari - Buyuklukleri')
plt.xlabel(r'$Buyuluk (m^2)$')
plt.ylabel(r'$Fiyat (TL)$')
plt.tight_layout()
plt.show()
```



232 metrekarelik bir ev bakıyoruz. Bu evin fiyatının ne olacağını merak ediyorduk, bunun cevabını da görsele bakarak cevaplayabiliriz. Hem tahminimizi gerçekleştirelim hem de tahminimize test verilerini kullanarak çizdiğimiz görselde modelimizin üstünde bakalım.

```
tahminimiz = model.tahmin_et(232)
print '232 metrekarelik evin fiyati : ', tahminimiz, 'TL'


plt.scatter(X_test, y_test, edgecolors='black', c='yellow')
plt.plot(X_test, y_tahmin, c='blue', zorder=1)
plt.scatter(232, tahminimiz, edgecolors='black', c='red', s=100, zorder=2)
plt.title('232 metrekarelik Evin Fiyati')
plt.xlabel(r'$Buyuluk (m^2)$')
```

```
plt.ylabel(r'$Fiyat (TL)$')
plt.legend(['Model', 'Veri', r'232 $(m^2)$lik evin fiyati'])
plt.tight_layout()
plt.show()
```

```
232 metrekarelik evin fiyati : 565469.02671 TL
```