3.3 Tek Değişkenli Lineer Regresyon

Elimizde bir eğitim veri seti var. X girdisinden y'yi tahmin etmek istiyoruz. Örnek olarak evin büyüklüğünü x olarak, tahmin etmeye çalıştığımız y'yi de evin fiyatı olarak düşünebiliriz.

- x : Girdi (Evin Büyüklüğü)
- y : Çıktı (Evin Fiyatı)

Şimdi x değerinden y değerini bulmaya, tahmin etmeye çalışacağız. Eğer elimizdeki veriler şu şekilde olsaydı:

Evin Büyüklüğü (metrekare)	Evin Fiyatı (TL)
100	100.000
200	200.000
300	300.000
400	???
500	500.000
600	600.000

400 metrekarelik evin fiyatını hiç tereddüt etmeden 400.000 TL derdik. Bunun nedenini anlayalım:

x : Evin Büyüklüğü (metrekare)
y : Evin Fiyatı (TL)

Bu iki bileşen arasında şöyle bir bağlantı kurduk.

$$y = 1000 * x$$

Bu bulduğumuz denkleme, makine öğrenmesinde hipotez denir. Daha resmi bir şekilde ifade edecek olursak:

- **Hipotez:** x girdilerini y çıktılarına eşler.

Makine Öğrenmesi problemlerinin hepsinde hipoteze ihtiyaç duyulur bunun sebebi var olan verilerden bir sonuca ulaşmak istenilmesidir.Yani, elimizdeki verilerden(x), sonuca(y) ulaşmak için var olan verilerden bizi sonuca götürecek bir fonksiyona ihtiyaç duyarız, yani *Hipotez*'e.

Karşılaşılan problemler her seferinde ev fiyatı tahmin etmek kadar kolay olmayacak ve bunu böyle hemen zihnimizden yapamayacağız. Makine öğrenmesinde hipoteze

ulaşmak için bir *Öğrenme Algoritmasına* ihtiyaç duyarız.

- **Öğrenme Algoritması:** Hipotez'e ulaşmak için kullanılan algoritmalar

Ele aldığımız ev fiyatı tahmini probleminde yanıt değişkenine ulaşmamızı sağlayan tek bir öngörücü değişken olduğu için tek değişkenli bir lineer regresyon problemidir.

Hipotez'i Nasıl Gösteririz?

Ele aldığımız problemde Hipotezi aşağıdaki gibi gösterebiliriz:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

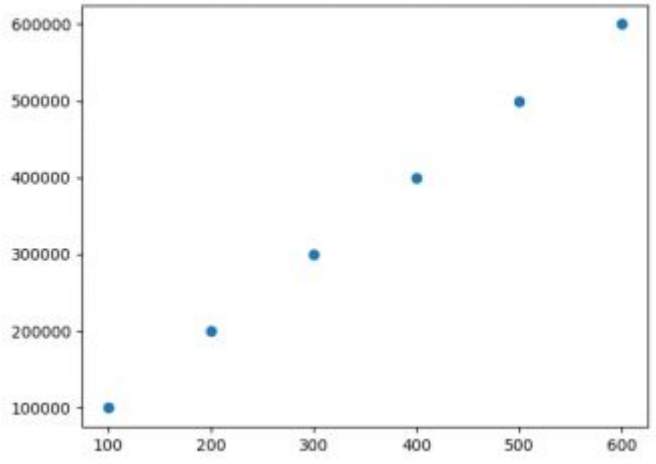
Burada gözüktüğü gibi, parametreler:

$$\theta_0, \theta_1$$

Devam etmeden önce bazı hatırlatmalar yapalım.

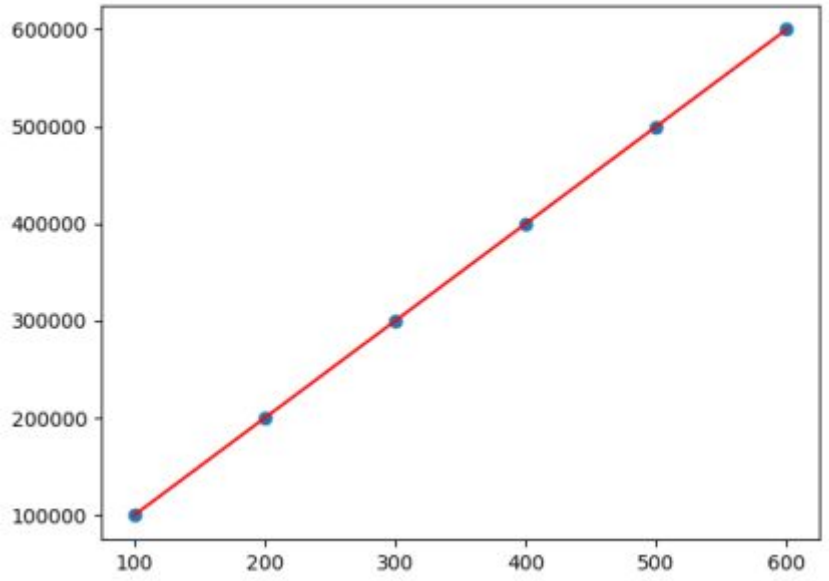
Örneğin bir insan ya hastadır 1 değerini alabilir, ya da sağlıklıdır 0 değerini alabilir. Yani bir insanın hem hasta hem sağlıklı olma durumu yoktur, 0.5 değerini alamaz. Ev fiyatlarına bakacak olursak, evin fiyatları pozitif tüm değerleri alabileceği gibi evin büyüklüğünü de ifade ederken tüm pozitif değerleri kullanabiliriz. Yani bu duruma sürekli diyebiliriz. Oysa, bir kişinin sağlıklı veya hasta olma durumu ayrıktır.

Hipotez bizi x değerlerinden y'ye götüren fonksiyondu. Neden iki tane parametre olduğunu daha iyi anlayabilmek için ev fiyatları ile ev büyüklüğüne ait verileri bir grafik üzerinde inceleyelim.



Yukarıdaki grafikte x ekseninde girdilerimiz(x) yani ev büyüklükleri yer alırken, y

ekseninde çıktılarımız(y) yani ev fiyatları yer almaktadır. Burada verilerimize en iyi uyan eğriyi bulmak isteseydik bu da aşağıdaki gibi olacaktı:



Önceden de belirttiğimiz gibi, bu problemde verilerimize en iyi uyan eğri aslında bir doğrudur. Bu nedenle hipotezimiz bir doğru denklemi şeklindedir ve iki tane parametresi vardır. Hatırlarsanız verilere en iyi uyacak fonksiyonu tahmin etmeye çalıştığımızda y = 1000 * x olarak bulmuştuk.

3.3.1 Maliyet Fonksiyonu

Parametreleri yani tetaları 1000 ve 0 olarak zihnimizden tahmin ettik (y=1000*x+0) ve yukarıdaki grafikteki gibi verilerimize mükemmel uydu. Peki parametreleri yani bu tetaları nasıl bulacağız. Hipotezde yer alan parametreleri bulma işlemine *maliyet fonksiyonu* denir. Daha güzel bir şekilde ifade edersek:

- **Maliyet Fonksiyonu:** Kabaca diyebiliriz ki bulduğumuz hipotez ile var olan gerçek değerler arasındaki fark bizim maliyet fonksiyonumuzdur. Yani hipotezimizin bize olan maliyetini ölçmemizi sağlar.

Maliyet fonksiyonu olarak birçok metrik kullanılabilir: MSE, RMSE, MAE, MAPE gibi.

Biz bu problemde maliyet fonksiyonu olarak kare hata fonksiyonunu (Squared Error Function) kullanacağız. Makine öğrenmesinde gelenek olarak, hipotezi (h) ile maliyet fonksiyonunu da (J) ile gösterilir. Maliyet fonksiyonumuz şu şekildedir:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Makine öğrenmesi problemlerinin hepsinde asıl amaç bu maliyet fonksiyonunu

minimize etmektir. Doğal olarak maliyetimiz ne kadar az olursa gerçeğe en yakın değerleri tahmin edebiliriz. Amacımız:

$$\underset{\theta_0,\theta_1}{\text{minimize}}\ J(\theta_0,\theta_1)$$

Bu maliyet fonksiyonunu nasıl minimize edeceğimize geçmeden önce elimizde ne var bir bakalım.

Hypothesis:	$h_\theta(x) = \theta_0 + \theta_1 x$
Parameters:	θ_0, θ_1
Cost Function:	$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
Goal:	$\underset{\theta_0,\theta_1}{\text{minimize}}\ J(\theta_0,\theta_1)$

Şimdi bunu biraz basitleştirmek için tek parametreye indirgeyip ve şu şekle dönüştürelim.

Hypothesis:	$h_\theta(x) = \theta_1 x$
Parameters:	θ_1
Cost Function:	$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
Goal:	$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$

Parametre sayısını 1'e düşürdük, ve fark ettiyseniz yeni hipotezimiz, bizim problemimize tamamen uyuyor, yani şuna: 1000 * x. Devam edelim ve maliyet fonksiyonunu kendi problemimiz için çözelim.

Hipotezi Maliyet fonksiyonunda yerine koyalım:

$$J(\theta_1) = \frac{1}{2m}\sum_{i}^{m}((\theta_1 x^i) - y^i)^2$$

Buradaki m, örneklerimizin sayısı. Yani (x,y) çiftlerimizin. Bizim örneğimizde m = 5. Tahmin etmeye çalıştığımız örneği katmıyoruz. Onu tahmin edeceğiz.

Şimdi toplamı açalım:

$$J(\theta_1) = \frac{1}{2*5}[\ ((\theta_1 x^1) - y^1)^2 + \ ((\theta_1 x^2) - y^2)^2 + \ ((\theta_1 x^3) - y^3)^2 + \ ((\theta_1 x^5) - y^5)^2 + \ ((\theta_1 x^6) - y^6)^2]$$

x ve y değerlerimizi yerine yazalım:

$$J(\theta_1) = \frac{1}{10}[\ ((\theta_1 * 100) - 100000)^2 + \ ((\theta_1 * 200) - 200000)^2 + ((\theta_1 * 300) - 300000)^2 + \ ((\theta_1 * 500) - 500000)^2 + \ ((\theta_1 * 600) - 600000)^2$$

Evet, bu denklemi minimize etmek istiyoruz. Maliyet fonksiyonumuzu 0′ a eşitleyip devam edersek aslında en düşük maliyeti bulmuş olacağız:

$$\theta_1 = 1000$$

Hipotez fonksiyonumuz şimdi:

$$h_\theta(x) = \theta_1 * x = 1000 * x$$

Buna göre tahmin etmek istediğimiz ev büyüklüğünü yani 400 metrekareyi x yerine koyarak fiyatına ulaşabiliriz.

$$h_\theta(x) = \theta_1 * x = 1000 * 400 = 400000$$

Sonuç olarak, verilerimize en uygun eğri olan doğruyu çizmemizi sağlayan hipotez fonksiyonumuzu yazdık ve tahminimizi gerçekleştirdik.

Şimdi biraz da maliyet fonksiyonu ile tetanın arasındaki ilişkiye bir göz atalım. Maliyeti en az olan hipotez bizim için en iyi modeli, eğriyi vermektedir. Bunun için maliyet fonksiyonunu 0'a eşitleyerek, maliyetimizi en aza indirgeyecek teta'yı bulmaya çalıştık.

Şimdi bilgilerimizi bir gözden geçirelim. Kare hata fonksiyonunu maliyet fonksiyonumuz olarak kullandık. Amacımız, maliyet fonksiyonundan tetayı çekmekti. Teta değeri bizim hipotezimizde yer alan bir parametre ve biz öyle bir teta bulmak istedik ki elde ettiğimiz hipotez verilerimize tam uyum sağlasın. En iyi uyumu sağlayan teta için maliyet fonksiyonumuzu da en aza indirgemek istedik. Bu durumu biraz daha görsel olarak incelersek daha iyi anlayacağız.

Bunun için maliyet fonksiyonu olan J(θ) ile çeşitli hesaplamalar yapacağız.

θ'yı 0 seçseydik, maliyetimiz ne kadar olurdu?

$$\theta = 0$$
$$J(\theta) = J(0) = \frac{1}{10}[((0 * 100) - 100000)^2 + ((0 * 200) - 200000)^2 +$$
$$((0 * 300) - 300000)^2 + ((0 * 500) - 500000)^2 + ((0 * 600) - 600000)^2]$$
$$J(0) = 75 * 10^9$$

θ'yı 500 seçseydik, maliyetimiz ne kadar olurdu?

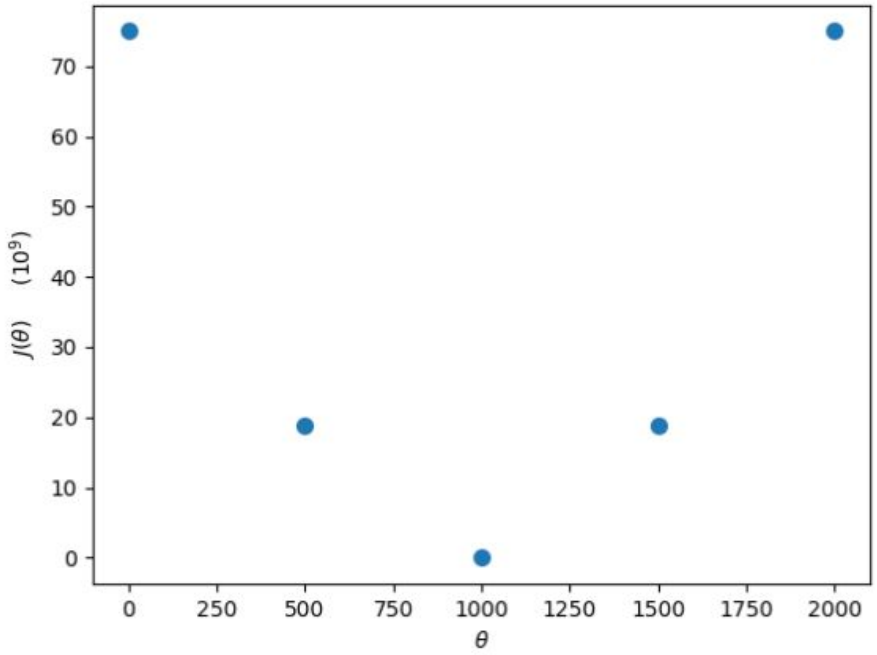
$$\theta = 500$$
$$J(\theta) = J(500) = \frac{1}{10}[((500 * 100) - 100000)^2 + ((500 * 200) - 200000)^2 +$$
$$((500 * 300) - 300000)^2 + ((500 * 500) - 500000)^2 + ((500 * 600) - 600000)^2]$$
$$J(500) = 18,75 * 10^9$$

θ'yı 1000 seçseydik, maliyetimiz ne kadar olurdu?
Bunu zaten hesaplamıştık, maliyet 0 olur.

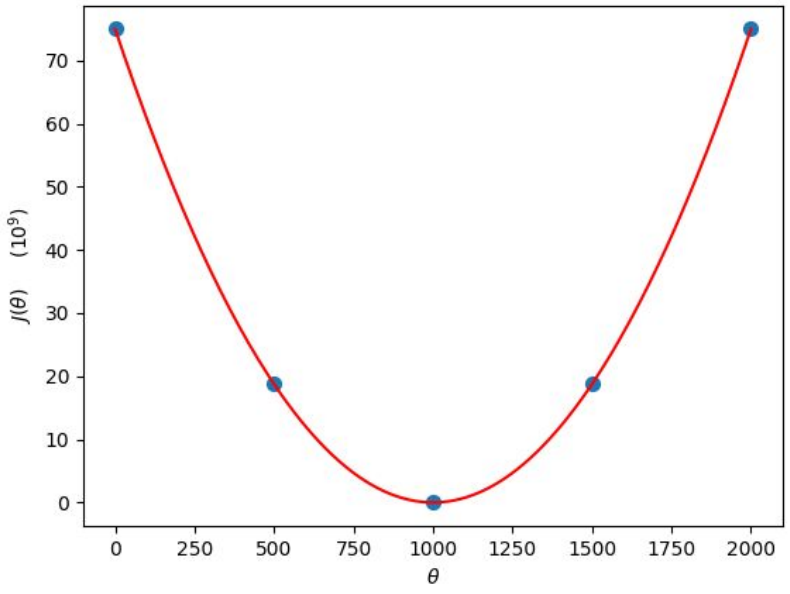
$$J(1000) = 0$$

Daha fazla hesaplama yapmaya gerek yok aslında, şimdi maliyet fonksiyonu ve teta arasındaki ilişkiyi grafik üzerinde bakarsak tahminen şöyle gözükecek:

J(1500) ile J(2000)' i ve diğerlerini isterseniz hesaplayabilirsiniz.



Diğer J(θ) 'ların hepsini hesaplarsanız, grafiğimiz şöyle gözükür:

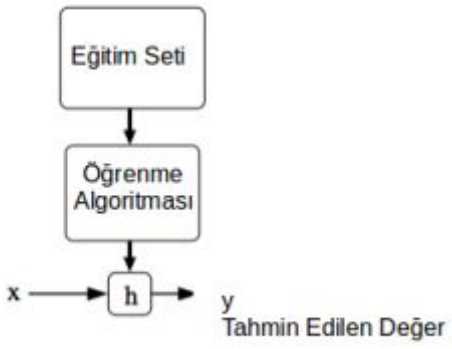


Bu grafikten ne anlıyoruz peki?

Ele aldığımız problemde maliyet fonksiyonunu minimize eden tek bir θ değeri mevcut olduğunu ve bulduğumuz θ değerinin bize en iyi hipotezi vermiş olduğunu gördük.

Özetle bu problemde, tek değişkenli bir lineer regresyon problemini ele aldık ve tahmin etmeye çalıştığımız değeri kare hata fonksiyonunu maliyet fonksiyonu olarak

kullanarak, hipotezimizi hesapladık.



Hipotezimizi basitleştirmeden önceki haline geri dönelim.

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Hatırlarsanız, makine öğrenmesinde asıl yapmak istediğimiz, amacımız, verilere en uygun eğriyi bulmaktı. Verilerin bir dağılım grafiğini çizdiğimizde, veriler eğer düz bir çizgiye düşüyorsa, yukarıdaki hipotezi kullanabiliriz, çünkü hipotez fonksiyonumuz düz bir çizgiyi tanımlar, doğru denklemidir. Bunu biraz daha net anlayalım. θ_0, θ_1 değerlerini değiştirerek düzlem üzerine çizebileceğimiz sonsuz sayıda düz çizgi vardır. Ancak, dağılım grafiğimizde verilerin konumuna baktığımızda, verilerimize en iyi uyabilecek yalnızca bir düz çizgi vardır ve şimdi yapmak istediğimiz bu en iyi uyan çizgiyi çizebilmektir.

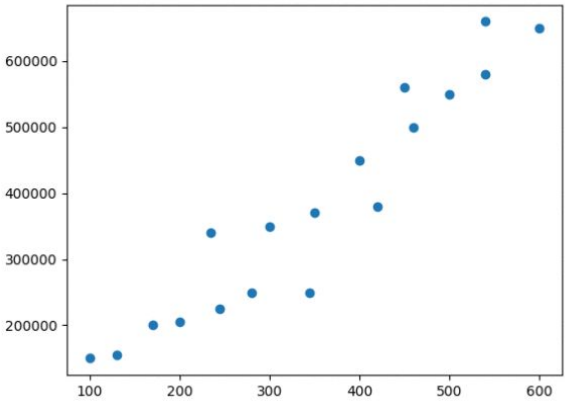
Maliyet fonksiyonu (J) bizim istediğimiz bu amaca algoritmik olarak hizmet eder. Eğer θ_0, θ_1 parametrelerinin doğru hipotezden çok uzakta olduğunu bulursak bize çok büyük bir maliyet verir. Ancak, doğru bir hipotezin maliyeti çok düşüktür. Bunu genelde θ_0, θ_1 'ların maliyet fonksiyonunu en aza indirgeyerek yaparız.

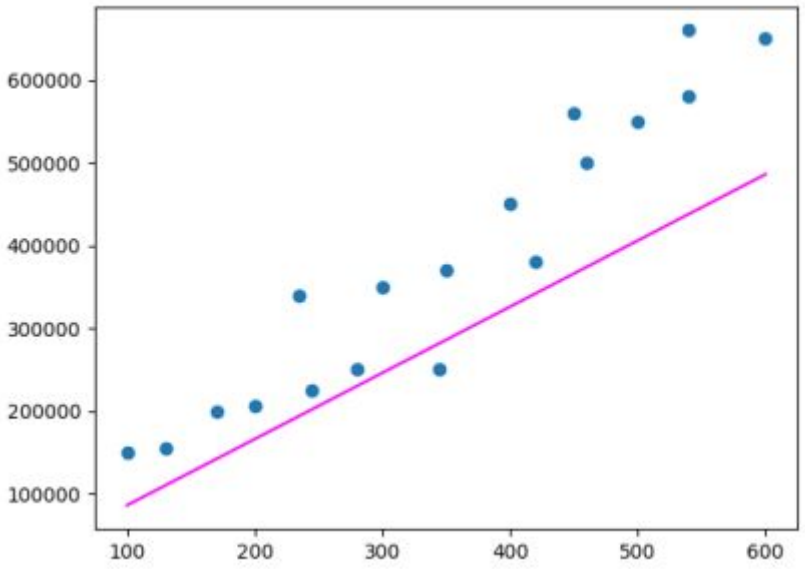
Şimdi konuyu biraz daha rahat anlayabilmek için elimizdeki veri setine bir göz atalım.

Evin Büyüklüğü (metrekare)	Evin Fiyatı (TL)
100	150000

130	155000
170	200000
200	206000
235	340000
245	225000
280	250000
300	250000
300	350000
345	250000
350	370000
400	450000
420	380000
450	560000
460	500000
500	550000
540	580000
540	660000
600	650000

Dağılım grafiğinde verilerimizi inceleyelim





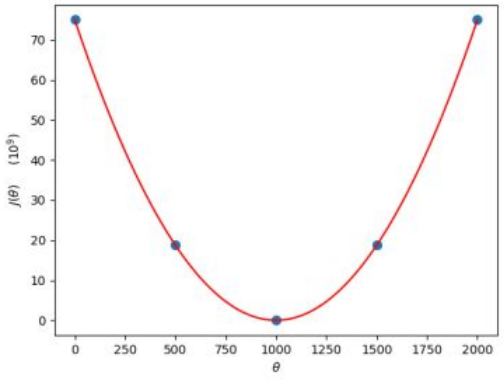
Eğer tetaları 800, 6000 olarak seçseydik, hipotezimiz şöyle olacaktı,

$$h_\theta(x) = 6000 + 800 * x$$

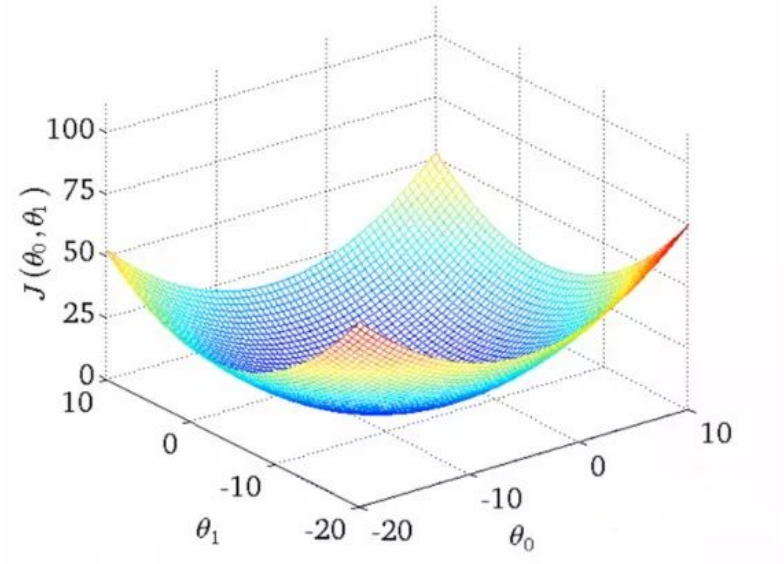
Bu hipotezin eğrisi de yukarıdaki gibi olacaktı. Rastgele seçtiğimiz teta değerlerinin verilerimize mükemmel uymadığını görüyoruz.

Tekrar hatırlamak gerekirse, hipotez (h) öğrenmeye çalıştığımız modeldir. Hipotez fonksiyonunu "öğrenmek" için θ'ları bulmamız gerekir.

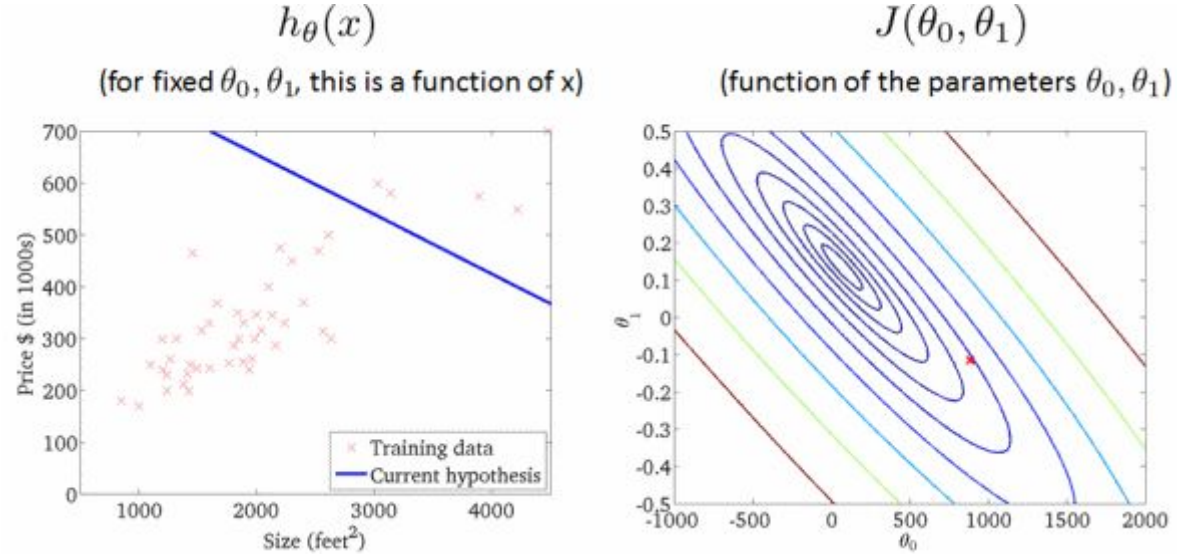
Hatırlarsanız tek bir teta ile gerçekleştirdiğimiz bir önceki örnekte maliyet fonksiyonu ile teta arasında şöyle bir ilişki olduğunu görmüştük:



Ancak şu an elimizde iki tane parametre var, θ_0, θ_1 ve tahmin edersiniz ki bunların maliyet fonksiyonu ile aralarındaki ilişki tek bir parametreye sahip hipotezdeki gibi basit olmayacaktır. Burada daha rahat görselleştirebilmemiz için devreye kontur grafikleri giriyor. Kontur grafiğini bir önceki grafiğin üç boyutlu hali gibi düşünebilirsiniz. Tabi bu grafiği isterseniz iki boyutlu uzayda da temsil edebilirsiniz. Aşağıda bu grafiğe ait örnekler yer alıyor.



Bundan sonra yapacaklarımızı daha iyi anlamak için aşağıdaki görseli dikkatlice inceleyin.



Şimdi yukarıdaki görseli birlikte yorumlayalım.

- Sağ tarafta kontur grafiğinin iki boyutlu düzlemde temsili verilmiş. Ortaya doğru giderek küçülen halkaya doğru baktığımızda tetas.png değerleri ile hesaplanan maliyet fonksiyonunun değerinin 0'a doğru yaklaştığını görüyoruz.
- Maliyet fonksiyonunun sıfıra yaklaşması bize en iyi hipotezi verecektir. En iyi hipotez ise veri setimize uyan en iyi eğriyi verir.
- Sağ tarafta tetas.png değerlerini kırmızı ile seçiyor ve sıfıra doğru yaklaştırıyor.
- Bu sırada sol tarafta verilen verilerimizin dağılım grafiğine baktığımızda,
- Seçilen tetalara göre hesaplanan hipotezin, maliyet fonksiyonunu minimize ettiğimiz anda verilere en iyi uyum sağladığını görebiliyoruz.

3.3.2 Dereceli Azalma

Bundan sonra ne yapacağımızı kısaca anlamışsınızdır. Amaç bir önceki örnekte olduğu gibi ve az önceki görselde gözüktüğü gibi maliyet fonksiyonumuzu minimize eden en iyi θ_0, θ_1 değerlerini seçerek hipotezimizi oluşturacağız. Burada *Dereceli Azalma*'yı kullanacağız. Dereceli azalmanın o meşhur karışık tanımlamaları yerine daha basitçe anlamaya çalışalım.

Gradient Descent ya da türkçe olarak Dereceli Azalma aslında bir algoritmadır. Sadece makine öğrenmesi problemlerinde değil, bir çok optimizasyon probleminde de kullanılır. Maliyet fonksiyonunu en aza indirgemek de bir optimizasyon problemidir ve bizim makine öğrenmesinde dereceli azalmayı kullanmamızın da en temel sebebi budur.

- **Dereceli Azalma Algoritması:** Basitçe, bir fonksiyonun minimumunu bulmamızı sağlar. Fonksiyonun minimumunu bulmak için birinci dereceden yinelemeli bir optimizasyon algoritmasıdır.

Biz dereceli azalmayı maliyet fonksiyonunu en aza indirgemek için bize gereken θ_0, θ_1 değerlerini bulmak için kullanacağız.

- Dereceli azalmada kullanacağımız fonksiyon : $J(\theta_0, \theta_1)$
- Amacımız : $\underset{\theta_0,\theta_1}{\text{minimize}}\, J(\theta_0, \theta_1)$

Dereceli Azalma Algoritmasının Basitçe Çalışma Şekli:

- θ_0, θ_1 'lara herhangi bir değer verin (teta değerlerine başlangıç değeri olarak 0 verebiliriz.).
- $J(\theta_0, \theta_1)$ değerini minimuma indirgeyene kadar θ_0, θ_1 değerlerini değiştirin.

Dereceli Azalmanın Algoritması:

$$
\begin{gathered}
Yakınsayana\ Kadar\ Devam\ \{ \\
\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\
\} \\
j = 0\ ve\ j = 1\ için
\end{gathered}
$$

$J(\theta_0, \theta_1)$ fonksiyonunun maliyet fonksiyonu olduğunu biliyoruz. θ_j nin de sırayla j=0 ve j=1 için θ_0, θ_1 değerlerini bize vereceğini biliyoruz.

- α : Öğrenme Oranı (bir katsayı)
- $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$: Maliyet Fonksiyonunun Kısmi Türevi
- θ_j = θ_0, θ_1 : Parametreler (tetalar)
- $:=$: eşzamanlı olarak güncellemek (θ_0, θ_1 'ları)

Eşzamanlı Olarak Tetaları güncellemek şu demek:

$$yeni\_teta\_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$yeni\_teta\_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := yeni\_teta\_0$$
$$\theta_1 := yeni\_teta\_1$$

Yani, önce tetaları bulacağız, sonra teta değişkenine atayacağız. Fark ettiyseniz, maliyet fonksiyonunun kısmi türevini aldığımız kısımda her iki tetayı da kullanıyoruz. Önce θ_0 'ı bulup θ_0 'a atarsak bir sonraki adımda θ_1 'i hesaplarken kullanacağımız θ_0 değerini değiştirmiş olduğumuz halde görürüz.

Yanlış Şekilde Tetaları Güncellemek de aşağıdaki şekilde olur:

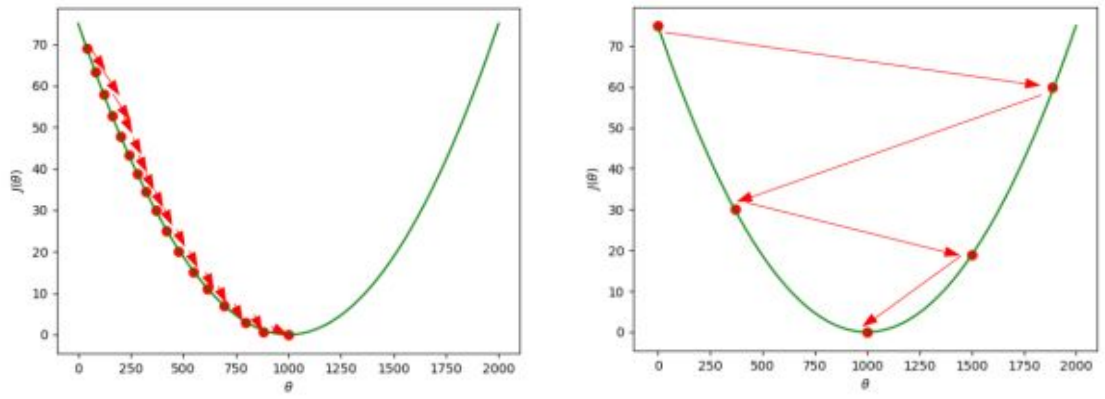
$$yeni\_teta\_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_0 := yeni\_teta\_0$$
$$yeni\_teta\_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_1 := yeni\_teta\_1$$

3.3.3 Öğrenme Oranı

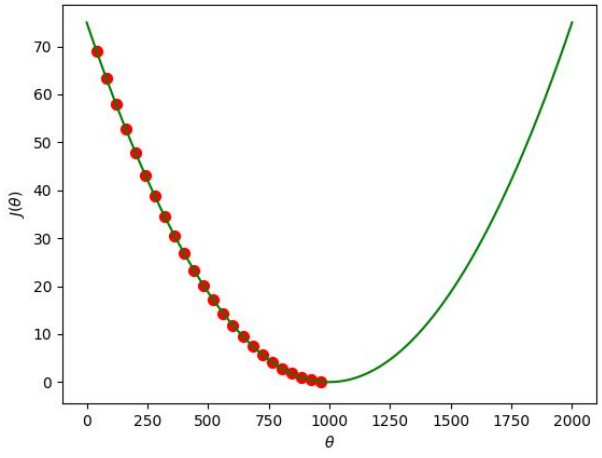
Şimdi öğrenme oranından α biraz bahsedelim.

- Öğrenme oranı, zamanın azalan bir fonksiyonudur. Bu parametre, optimum ağırlığa doğru ne kadar hızlı veya yavaş hareket edeceğimizi belirler.
- α çok büyükse optimal çözümü atlayacağız. Çok küçükse, en iyi değerleri birleştirmek için çok fazla yinelemeye ihtiyacımız olacaktır. Bu yüzden iyi bir α kullanmak çok önemlidir.

Görerek daha iyi anlarız. O yüzden dereceli azalma ile öğrenme oranı arasındaki ilişkiye bakalım. Sol tarafta alfayı çok küçük seçersek, sağ tarafta da alfayı çok büyük seçersek maliyeti nasıl minimize ettiğimize bakın.



Dereceli azalma(Gradyan inişi), öğrenme oranı α sabit olsa bile, yerel bir minimuma yaklaşabilir. Yerel bir minimuma yaklaştığımızda dereceli azalma otomatik olarak daha küçük adımlar atacaktır. Yani, zamanla α'nın azaltılmasına gerek yoktur. Yerel minimuma yaklaştıkça aslında grafiğimiz şöyle olacaktır:



Unutmayın, lineer regresyonun amacı maliyet fonksiyonunu en aza indirmektir.

Böylece en iyi θ_j değerlerini bularak veri setine en iyi uyan eğriyi çizebilirsiniz.

Şimdi tek değişkenli lineer regresyon örneğimizle devam edelim. Çalıştır programımız ile başlayalım. Çalıştıracağımız ana program sadece bu.

Önce gerekli paketleri içe aktaralım.

```
# coding=utf-8
""" Main
Calistir """

from ciz import dagilim_grafigi, hipotez_grafigi,
maliyet_devir_grafigi
from dereceli_azalma import dereceli_azalma
import pandas as pd
import numpy as np
```

Şimdi veri setini yükleyelim.

```
u""" 1- Veri Setinin Yuklenmesi """
print '1. Veri Seti Yukleniyor...'
regresyon_veri_1 = pd.read_csv('regresyon_veri_1.txt',
header=None, names=['X', 'Y'])

# orjinal veri setini bozmamak icin kopya olusturalim.
veri_seti = regresyon_veri_1.copy()
```

Veri setimizin dağılım grafiğine bir bakalım.

```
print '2. Verilerin Dagilim Grafigi Gosteriliyor ...'
dagilim_grafigi(veri_seti)
```

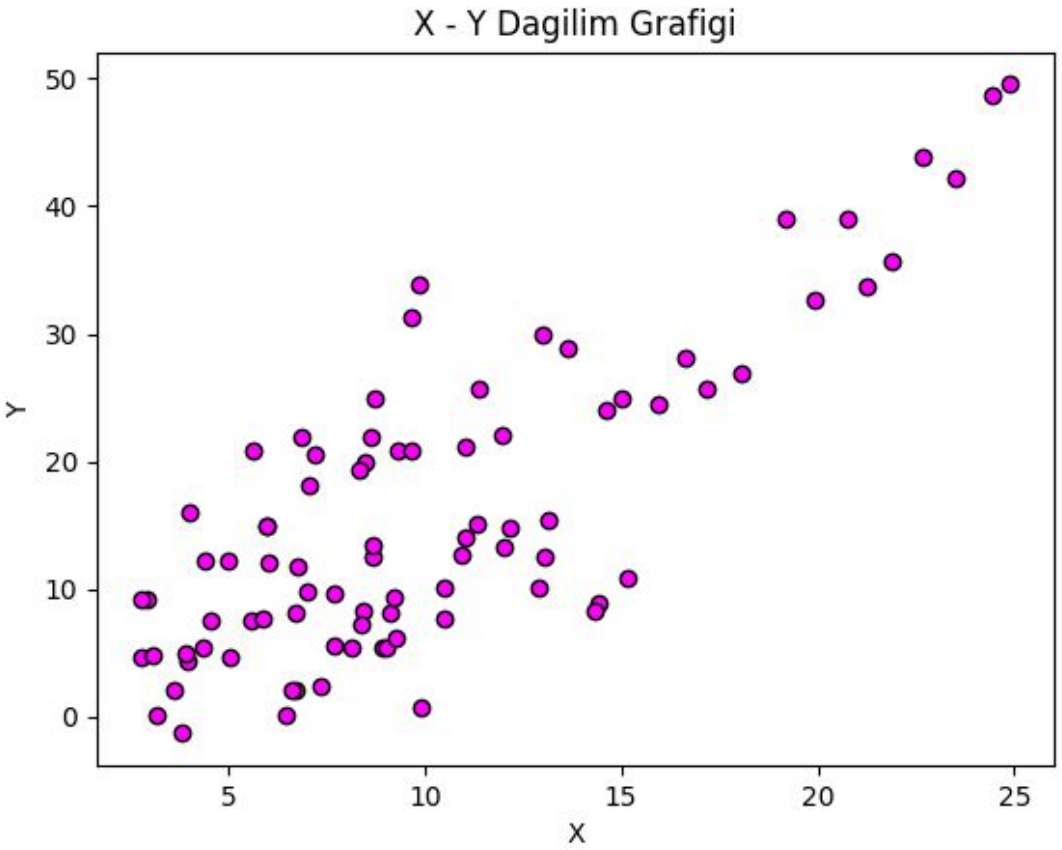
dagilim_grafigi(veri_seti) metoduna bakalım. Burada parametre olarak gelen veri setinden x ve y'e ekseni için gerekli bilgileri okuduktan sonra bir dağılım grafiği olarak gösteriyoruz.

```
# coding=utf-8
""" Plotting Data
Veri Setinin Grafiginin Cizilmesi"""

import numpy as np
import matplotlib.pyplot as plt
```

```
def dagilim_grafigi(veri_seti):
    """

    :param veri_seti: Dagilim Grafiginin Yapilacagi Veri Seti
    """
    x = np.asarray(veri_seti['X'])
    y = np.asarray(veri_seti['Y'])
    plt.figure()
    plt.scatter(x, y, c='magenta', edgecolors='black')
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.title('Dagilim Grafigi')
    plt.show()
```

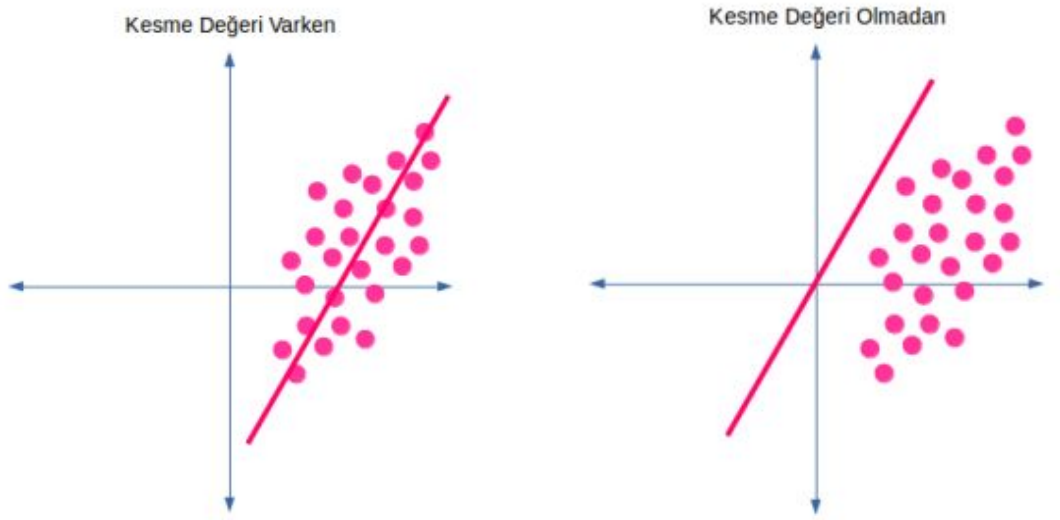


Verilerin dağılım grafiğine baktığımızda aralarında bir ilişki olduğunu açıkça görebiliyoruz, bir de bunu pearson momentler çarpımı ile bakalım.

```
print '3. Veri Arastirmasi ...'
print 'Verilerin Arasindaki Iliski, Pearson Momentler Carpimi :'
print veri_seti.corr()
```

3.3.4 Kesme Terimi

Lineer Regresyonda, **kesme değeri** yani **önyargı terimi** olmadan çözümümüz orijin noktasından geçmek zorundadır. Yani, tüm özellikler sıfır olduğunda tahmin edilen değerimizin de sıfır olması gerekir. Bununla birlikte, eğitim verilerimizin öne sürdüğü cevap bu olmayabilir. Herhangi bir özellikten bağımsız olarak bir önyargı ağırlığı eklenmesi, öğrendiğimiz ağırlıklarla açıklanan aşırı düzlemin (hiper düzlemin), başlangıç noktasından geçmeyen verilere daha kolay uymasını sağlar.



Öngürücü verimize kesme değeri ekleyelim ve verilerimizi öngörücü veri seti (X), ve hedef veri seti (y) olarak ikiye ayıralım.

```
u""" 4. Kesme Degerinin Eklenmesi"""
print '4. Kesme Degeri Ekleniyor...'
veri_seti.insert(0, 'Kesme Degeri', 1)
kesme_degeri_kolonu = veri_seti.shape[1]
X = np.matrix(veri_seti.iloc[:, 0:kesme_degeri_kolonu - 1])
y = np.matrix(veri_seti.iloc[:, kesme_degeri_kolonu -
1:kesme_degeri_kolonu])
```

Dereceli azalma ile teta değerlerini bulurken, teta değerlerine önceden ilk değerlerini atamamız gerekir, bu sebeple parametrelerin ilk değerlerini sıfır olarak ayarlayalım.

```
teta = np.matrix(np.array([0, 0]))
```

Öğrenme oranını ve devir sayısını *Dereceli Azalmayı* çalıştırdığımızda kullanacağız. Devir sayısı kaç iterasyon (döngü) yapacağımızı belirtirken, öğrenme oranı da ne kadar hızla hareket etmemiz gerektiğini belirtiyor.

```
u""" 5. Ogrenme Orani ve Devir Sayisinin Ayarlanmasi"""
print '5. Ogrenme Orani ve Devir Sayisi Ayarlaniyor ...'
alpha = 0.0001
toplam_devir = 1000

print 'Devir Sayisi ', toplam_devir
print 'Alfa ', alpha
```

Programın buraya kadar çıktısı aşağıdaki gibi olacaktır.

```
1. Veri Seti Yukleniyor...
2. Verilerin Dagilim Grafigi Gosteriliyor ...
3. Veri Arastirmasi ...
Verilerin Arasindaki Iliski, Pearson Momentler Carpimi :
          X         Y
X  1.000000  0.797074
Y  0.797074  1.000000
Aralarinda Guclu Pozitif Korelasyon var, r : 0.797073934649
4. Kesme Degeri Ekleniyor...
5. Ogrenme Orani ve Devir Sayisi Ayarlaniyor ...
Devir Sayisi 1000
Alfa  0.0001
```

Artık Dereceli Azalmayı çalıştırabiliriz ve dereceli_azalma(x, y, theta, alpha, epoch) fonksiyonunu çağırıp, sonra da nasıl çalıştığına bakalım.

```
u""" 6. Dereceli Azalmanin Hesaplanmasi"""
print '6. Dereceli Azalma Hesaplaniyor ...'
teta, maliyet = dereceli_azalma(X, y, teta, alpha, toplam_devir)
```

Dereceli azalma ve maliyet hesapları 3. bölümün başlarında anlatılmıştır.

```
# coding=utf-8
""" Gradient Descent
Dereceli Azalma """
import numpy as np
from maliyeti_hesapla import maliyet_hesapla


def dereceli_azalma(x, y, theta, alpha, epoch):
    """

    :param x: Ongorucu Degisken
    :param y: Hedef Degisken
    :param theta: Parametreler
    :param alpha: Ogrenme Orani
```

```
    :param devir_sayisi: Iterasyon Sayisi
    :return: theta: bulunan parametreler ve maliyet
    """
    yeni_theta = np.matrix(np.zeros(theta.shape))
    parametreler = int(theta.ravel().shape[1])

    maliyet_J = np.zeros(shape=(epoch, 1))

    for i in range(epoch):
        hata = (x * theta.T) - y

        for j in range(parametreler):
            term = np.multiply(hata, x[:, j])
            ara_islem = (np.divide(alpha, len(x)) * np.sum(term))
            yeni_theta[0, j] = np.subtract(theta[0, j], ara_islem)

            theta = yeni_theta
            maliyet_J[i, 0] = maliyet_hesapla(x, y, theta)

    return theta, maliyet_J
```

```
# coding=utf-8
""" Compute Cost Function
Maliyet Fonksiyonunu Hesapla """
import numpy as np


def maliyet_hesapla(x, y, theta):
    """

    :param x: Ongorucu Degisken
    :param y: Hedef Degisken
    :param theta: parametre
    :return: maliyet
    """
    m = len(x)
    toplamin_acilimi = np.power(((x * theta.T) - y), 2)
    maliyet = np.sum(toplamin_acilimi) / (2 * m)
    return maliyet
```

Şimdi dereceli azalma algoritmamızın doğrulunu ölçebilmek için hesaplanan teta değerlerimize, maliyet-devir ve hipotez grafiklerimize bakalım.

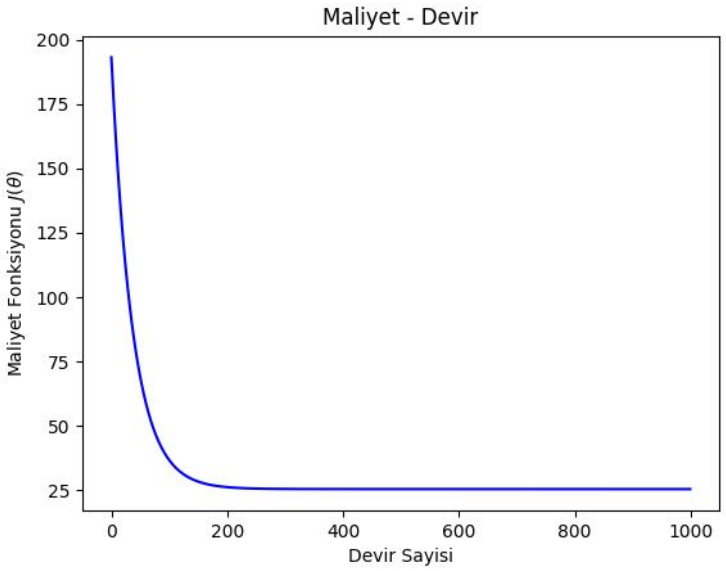
```
u""" 5- Sonuc"""
print '5- [Sonuc]: Grafikler ...'
```

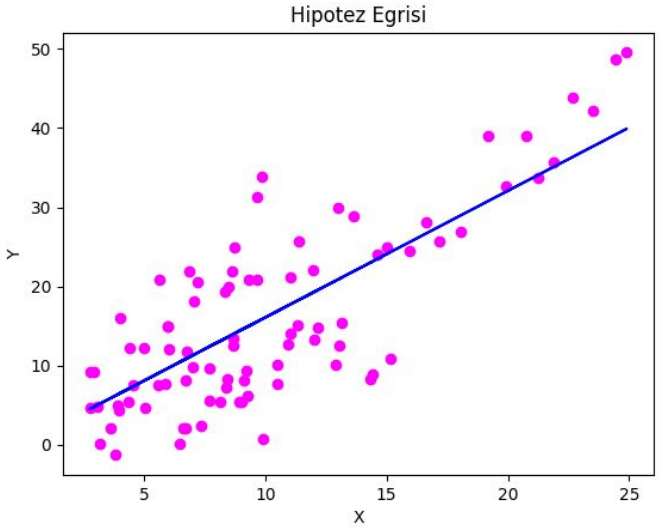
```
print 'Hesaplanan Teta ', teta
maliyet_devir_grafigi(toplam_devir, maliyet)
hipotez_grafigi(veri_seti['X'], veri_seti['Y'], teta)

print '[BILGI] : Program Basari ile Sonlandi.'
```

Programının çıktısı şu şekildedir:

```
4- Dereceli Azalma Hesaplaniyor ...
5- [Sonuc]: Grafikler ...
Hesaplanan Teta  [[0.08665226 1.59912138]]
[BILGI] : Program Basari ile Sonlandi.
```





Grafiklerde gözüktüğü gibi veri setimize en uygun hipotez eğrisini bulduk ve dereceli azalma fonksiyonumuz doğru çalışmıştır.