Proje-1: Regresyon

Not: Kod Dosyalarının Açıklamaları Kodun İçerisinde Yorum Satırları Şeklinde Açıklanmıştır.

Soru 1

```python
import pandas as pd
import matplotlib.pyplot as plt
# Gerekli olan kütüphaneleri ekliyoruz.
# Pandas kütüphanesini ile veri setimizi python kodunda kullanılacak şekilde içe aktarılıyor.
# Matplotlib kütüphanesi ile verilerin grafik ile görselleştirilmesi sağlanıyor.



# data_fd değişkenine veri setini atıyoruz.
data_fd = pd.read_csv('kc_house_data.csv')


# x1 değişkenine veri setindeki 'sqft_living' sütunu atanıyor.
x1 = data_fd['sqft_living'].values
# x2 değişkenine veri setindeki 'sqft_lot' sütunu atanıyor.
x2 = data_fd['sqft_lot'].values
# y değişkenine 'price' sütunu atanıyor.
y = data_fd['price'].values


# Bulunacak olan teta değişkenlerini tanımlıyoruz ve 0 değerini atıyoruz.
teta0 = 0
teta1 = 0
teta2 = 0


# Algoritmanın hızını belirleyen alfa değişkenini oluşturup 0.0000001 değeri atanıyor.
alfa = 0.000001
# Veri setindeki veri sayısı çekilip ornekSayisi değişkenine atanıyor.
ornekSayisi = len(x1)
```

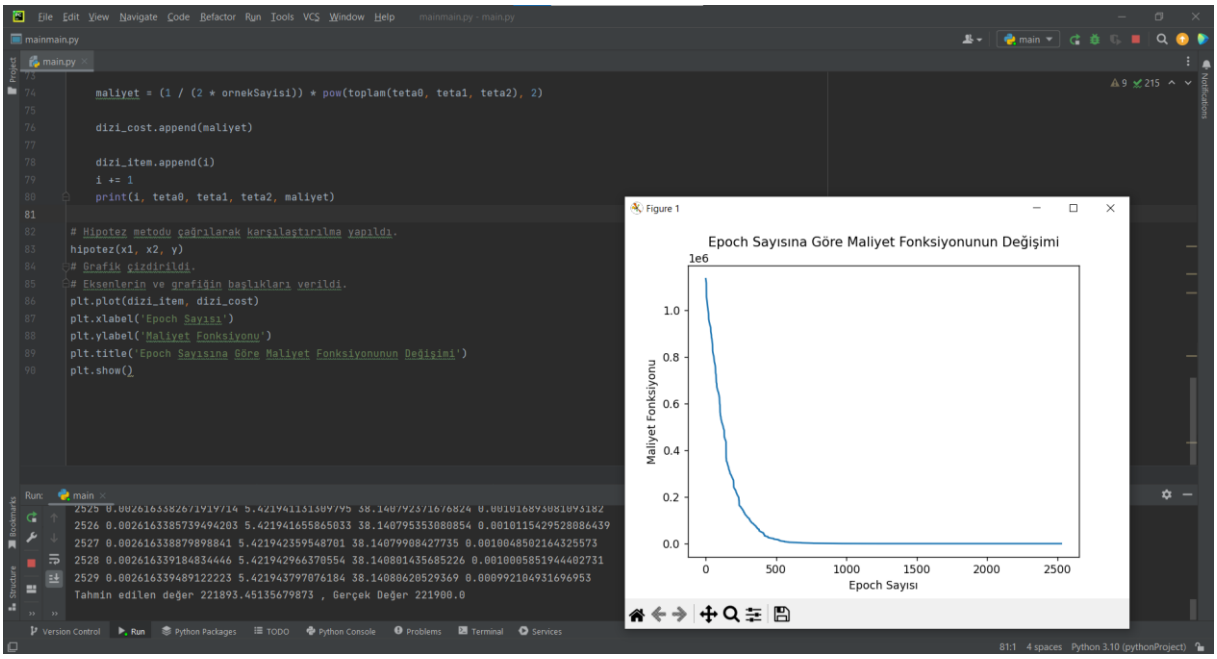
```python
# Çizdirilen grafikteki değerleri oluşturduğumuz dizi_cost ve dizi_item dizilerinden alıyoruz.
dizi_cost = []
dizi_item = []


# Gradient Descent algoritmasındaki toplam sembolünün bulunduğu kısmı ayırıp for döngüsü içerisinde
# toplam methodu içerisinde hesaplayıp return ile döndürüyoruz.
def toplam(t0, t1, t2):
    topla = 0
    for i in range(0, ornekSayisi, 1):
        topla += (t0 + t1*x1[i] + t2*x2[i] - y[i])

        return topla
```

```
# Maliyet değerimizin bulunduğu değişkeni tanımlayıp varsayılan olarak 1.0 değeri atanıyor.
# Burada maliyet değişkenine 1.0 değerini atamamızın sebebi 0 olduğunda döngünün içine girmemesidir.
maliyet =1.0

# Aşağıda hipotezin doğruluğunu test etmek için hipotez metodu oluşturuldu.
# Bulunan tetalar ile hipotez formulü oluşturulup x değerlerine veri setinden
# veriler çekilerek test edildi ve gerçek değer ile karşılaştırılması yapıldı ve ekrana yazdırıldı.
def hipotez(x1, x2, y):
    h = teta0 + teta1 * x1[0] + teta2 * x2[0]
    print("Tahmin edilen değer {} , Gerçek Değer {}".format(h, y[0]))
    return h
```

```
# while döngüsü oluşturuldu.
# Durma değeri olarak belli bir aralık verildi.
# Elimizdeki özellik sayısı ile aynı sayıda çıktı alınabilmesi için 69. satırdaki mod yapısı oluşturuldu.
# Bu yapı ile index in alacağı değerler sınırlandırılarak özellik sayı kadar döndürülmesi sağlandı.
# Gradient Descent algoritması manuel olarak yazıldı ve hesaplandı.
# Her döngüde maliyet fonksiyou güncellendi.
# dizi_cost dizisinin içerisine eklendi.
# İterasyon sayısı içinde bir dizi oluşturuldu ve döngünün her dönüşünde eleman eklenmesi sağlandı.
# Veriler ekrana yazdırıldı.
i = 0
while(maliyet > 0.001 or maliyet < -0.001):
    index = i % ornekSayisi
    teta0 = (teta0 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2))
    teta1 = (teta1 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2) * x1[index])
    teta2 = (teta2 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2) * x2[index])

    maliyet = (1 / (2 * ornekSayisi)) * pow(toplam(teta0, teta1, teta2), 2)

    dizi_cost.append(maliyet)

    dizi_item.append(i)
    i += 1
    print(i, teta0, teta1, teta2, maliyet)
```

```
# Hipotez metodu çağrılarak karşılaştırılma yapıldı.
hipotez(x1, x2, y)
# Grafik çizdirildi.
# Eksenlerin ve grafiğin başlıkları verildi.
plt.plot(dizi_item, dizi_cost)
plt.xlabel('Epoch Sayısı')
plt.ylabel('Maliyet Fonksiyonu')
plt.title('Epoch Sayısına Göre Maliyet Fonksiyonunun Değişimi')
plt.show()
```

Soru1 Çıktı


maliyet = (1 / (2 * ornekSayisi)) * pow(toplam(teta0, teta1, teta2), 2)
dizi_cost.append(maliyet)
dizi_item.append(i)
i += 1
print(i, teta0, teta1, teta2, maliyet)
# Hipotez metodu çağrılarak karşılaştırılma yapıldı.
hipotez(x1, x2, y)
# Grafik çizdirildi.
# Eksenlerin ve grafiğin başlıkları verildi.
plt.plot(dizi_item, dizi_cost)
plt.xlabel('Epoch Sayisi')
plt.ylabel('Maliyet Fonksiyonu')
plt.title('Epoch Sayısına Göre Maliyet Fonksiyonunun Değişimi')
plt.show()
2526 0.0026163385739494203 5.421941655865033 38.140795353080854 0.0010115429528086439
2527 0.0026163388798988 41 5.421942359548701 38.140799084277735 0.0010048502164325573
2528 0.0026163391848344 46 5.421942966370554 38.140801435685226 0.0010005851944402731
2529 0.0026163394891222 23 5.421943797076184 38.14080620529369 0.000992104931696953
Tahmin edilen değer 221893.45135679873 , Gerçek Değer 221900.0
Figure 1
Epoch Sayısına Göre Maliyet Fonksiyonunun Değişimi
1e6
Maliyet Fonksiyonu
Epoch Sayısı

Soru 2

```python
import pandas as pd
import matplotlib.pyplot as plt
# Gerekli olan kütüphaneleri ekliyoruz.
# Pandas kütüphanesini ile veri setimizi python kodunda kullanılacak şekilde içe aktarılıyor.
# Matplotlib kütüphanesi ile verilerin grafik ile görselleştirilmesi sağlanıyor.


# data_fd değişkenine veri setini atıyoruz.
data_fd = pd.read_csv('kc_house_data.csv')


# Değişkenlere özelliklerimizi aktarıyoruz.
x1 = data_fd['bedrooms'].values
x2 = data_fd['bathrooms'].values
x3 = data_fd['sqft_living'].values
x4 = data_fd['sqft_lot'].values
x5 = data_fd['waterfront'].values
x6 = data_fd['view'].values
x7 = data_fd['condition'].values
x8 = data_fd['grade'].values
x9 = data_fd['sqft_above'].values
x10 = data_fd['sqft_basement'].values
x11 = data_fd['yr_built'].values
x12 = data_fd['yr_renovated'].values
x13 = data_fd['lat'].values
x14 = data_fd['long'].values
x15 = data_fd['sqft_living15'].values
x16 = data_fd['sqft_lot15'].values
y = data_fd['price'].values
```

```python
# Bulunacak olan teta değişkenlerini tanımlıyoruz ve 0 değerini atıyoruz.
teta0 = 0.0
teta1 = 0.0
teta2 = 0.0
teta3 = 0.0
teta4 = 0.0
teta5 = 0.0
teta6 = 0.0
teta7 = 0.0
teta8 = 0.0
teta9 = 0.0
teta10 = 0.0
teta11 = 0.0
teta12 = 0.0
teta13 = 0.0
teta14 = 0.0
teta15 = 0.0
teta16 = 0.0


# Algoritmanın hızını belirleyen alfa değişkenini oluşturup 0.0000001 değeri atanıyor.
alfa = 0.000001
# Veri setindeki veri sayısı çekilip ornekSayisi değişkenine atanıyor.
ornekSayisi = len(x1)

# Çizdirilen grafikteki değerleri oluşturduğumuz dizi_cost ve dizi_item dizilerinden alıyoruz.
dizi_cost = []
dizi_item = []
```

```
# Gradient Descent algoritmasındaki toplam sembolünün bulunduğu kısmı ayırıp for döngüsü içerisinde
# toplam methodu içerisinde hesaplayıp return ile döndürüyoruz.
def toplam(t0, t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16):
    topla = 0.0
    for i in range(0, ornekSayisi, 1):
        topla += (t0 + t1 * x1[i] + t2 * x2[i] + t3 * x3[i] + t4 * x4[i] + t5 * x5[i] + t6 * x6[i] + t7 * x7[i] + t8 *
                  x8[i] + t9 * x9[i] +
                  t10 * x10[i] + t11 * x11[i] + t12 * x12[i] + t13 * x13[i] + t14 * x14[i] + t15 * x15[i] + t16 * x16[
                      i] - y[i])

        return topla

# Maliyet değerimizin bulunduğu değişkeni tanımlayıp varsayılan olarak 1.0 değeri atanıyor.
# Burada maliyet değişkenine 1.0 değerini atamamızın sebebi 0 olduğunda döngünün içine girmemesidir.
maliyet = 1.0
```

```
# Aşağıda hipotezin doğruluğunu test etmek için hipotez metodu oluşturuldu.
# Bulunan tetalar ile hipotez formulü oluşturulup x değerlerine veri setinden
# veriler çekilerek test edildi ve gerçek değer ile karşılaştırılması yapıldı ve ekrana yazdırıldı.
def hipotez(x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, y):
    h = teta0 + teta1 * x1[0] + teta2 * x2[0] + teta3 * x3[0] + teta4 * x4[0] + teta5 * x5[0] + teta6 * x6[0] + teta7 * \
        x7[0] + teta8 * x8[0] + teta9 * x9[0] + teta10 * x10[0] + teta11 * x11[0] + teta12 * x12[0] + teta13 * x13[
            0] + teta14 * x14[0] + teta15 * x15[0] + teta16 * x16[0]
    print("Tahmin edilen değer {} , Gerçek Değer {}".format(h, y[0]))
    return h
```

```
# while döngüsü oluşturuldu.
# Durma değeri olarak belli bir aralık verildi.
# Elimizdeki özellik sayısı ile aynı sayıda çıktı alınabilmesi için 101. satırdaki mod yapısı oluşturuldu.
# Bu yapı ile index in alacağı değerler sınırlandırılarak özellik sayı kadar döndürülmesi sağlandı.
# Gradient Descent algoritması manuel olarak yazıldı ve hesaplandı.
# Her döngüde maliyet fonksiyou güncellendi.
# dizi_cost dizisinin içerisine eklendi.
# İterasyon sayısı içinde bir dizi oluşturuldu ve döngünün her dönüşünde eleman eklenmesi sağlandı.
# Veriler ekrana yazdırıldı.
i = 0
while (maliyet > 0.001 or maliyet < -0.001):
    index = i % ornekSayisi

    teta0 = (teta0 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                        teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                        teta14, teta15, teta16))
    teta1 = (teta1 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                        teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                        teta14, teta15, teta16) * x1[index])
    teta2 = (teta2 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                        teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                        teta14, teta15, teta16) * x2[index])
    teta3 = (teta3 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                        teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                        teta14, teta15, teta16) * x3[index])
    teta4 = (teta4 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                        teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                        teta14, teta15, teta16) * x4[index])
    teta5 = (teta5 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                        teta7, teta8, teta9, teta10, teta11, teta12, teta13,
```

```
                                                    teta14, teta15, teta16) * x5[index])
    teta6 = (teta6 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                    teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                    teta14, teta15, teta16) * x6[index])
    teta7 = (teta7 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                    teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                    teta14, teta15, teta16) * x7[index])
    teta8 = (teta8 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                    teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                    teta14, teta15, teta16) * x8[index])
    teta9 = (teta9 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                    teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                    teta14, teta15, teta16) * x9[index])
    teta10 = (teta10 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                      teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                      teta14, teta15, teta16) * x10[index])
    teta11 = (teta11 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                      teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                      teta14, teta15, teta16) * x11[index])
    teta12 = (teta12 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                      teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                      teta14, teta15, teta16) * x12[index])
    teta13 = (teta13 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                      teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                      teta14, teta15, teta16) * x13[index])
    teta14 = (teta14 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                      teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                      teta14, teta15, teta16) * x14[index])
    teta15 = (teta15 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                      teta7, teta8, teta9, teta10, teta11, teta12, teta13,
```

```
                                                    teta14, teta15, teta16) * x15[index])
    teta16 = (teta16 - alfa * (1.0 / ornekSayisi) * toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6,
                                                    teta7, teta8, teta9, teta10, teta11, teta12, teta13,
                                                    teta14, teta15, teta16) * x16[index])

    maliyet = (1 / (2 * ornekSayisi)) * pow(
        toplam(teta0, teta1, teta2, teta3, teta4, teta5, teta6, teta7, teta8, teta9, teta10, teta11, teta12,
               teta13, teta14, teta15, teta16), 2)

    dizi_cost.append(maliyet)

    dizi_item.append(i)
    i += 1

    print(i, teta0, teta1, teta2, teta3, teta4, teta5, teta6, teta7, teta8, teta9, teta10, teta11, teta12, teta13,
          teta14, teta15, teta16, maliyet)
```

```
# Hipotez metodu çağrılarak karşılaştırılma yapıldı.
hipotez(x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, y)
# Grafik çizdirildi.
# Eksenlerin ve grafiğin başlıkları verildi.
plt.plot(dizi_item, dizi_cost)
plt.xlabel('Epoch Sayısı')
plt.ylabel('Maliyet Fonksiyonu')
plt.title('Epoch Sayısına Göre Maliyet Fonksiyonunun Değişimi')
plt.show()
```

Soru 2 Çıktı

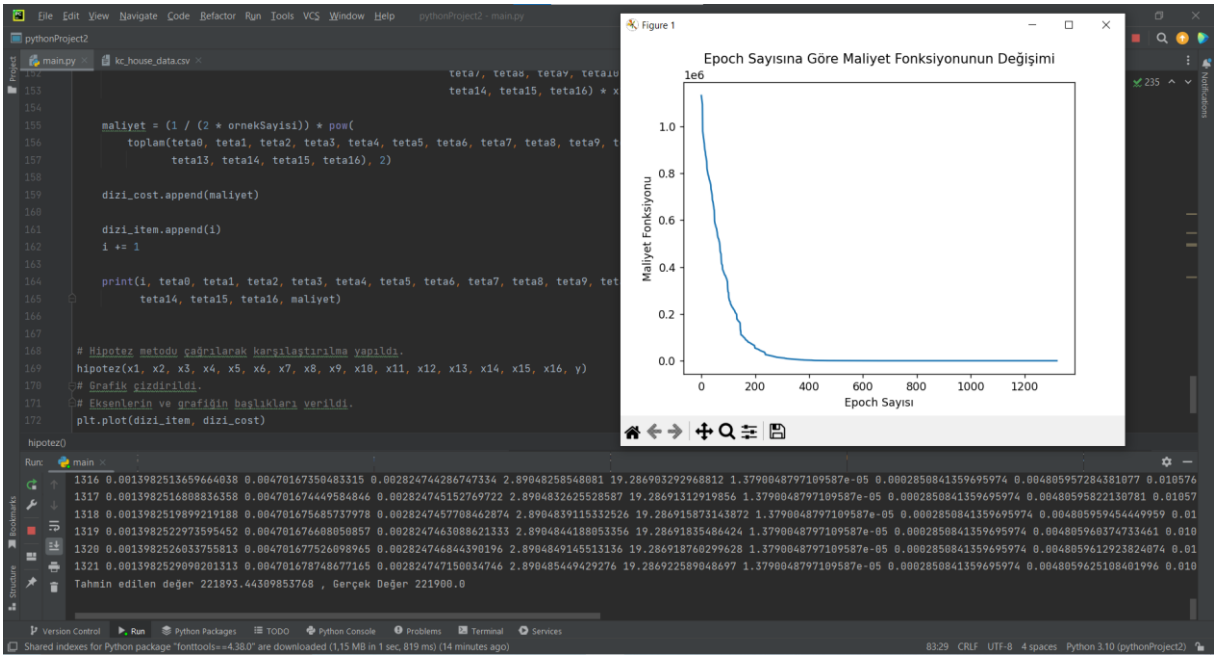


Figure 1
Epoch Sayısına Göre Maliyet Fonksiyonunun Değişimi
1e6
Maliyet Fonksiyonu
Epoch Sayısı
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
1200
1316 0.0013982513659664038 0.08470167350483315 0.002824744286747334 2.89048258548081 19.286903292968812 1.3790048797109587e-05 0.0002850841359695974 0.004805957284381077 0.010576
1317 0.0013982516808836358 0.08470167449584846 0.002824745152769722 2.8904832625528587 19.28691312919856 1.3790048797109587e-05 0.0002850841359695974 0.004805958221307 81 0.01057
1318 0.0013982519899219188 0.08470167568573797 8 0.002824745770846287 4 2.8904839115332526 19.286915873143872 1.3790048797109587e-05 0.0002850841359695974 0.004805959454449959 0.01
1319 0.0013982522973595452 0.08470167660880585 7 0.002824746308862133 3 2.8904844188853356 19.286918354864 24 1.3790048797109587e-05 0.0002850841359695974 0.004805960374733461 0.010
1320 0.0013982526033755813 0.08470167752609896 5 0.002824746844390196 2.8904849145513136 19.286918760299628 1.3790048797109587e-05 0.0002850841359695974 0.004805961292382 4074 0.01
1321 0.0013982529090201313 0.08470167874867716 5 0.002824747150034746 2.8904854494292 76 19.286922589048697 1.3790048797109587e-05 0.0002850841359695974 0.004805962510840 1996 0.010
Tahmin edilen değer 221893.44309853768 , Gerçek Değer 221900.0

Açıklamalar

Model matematiksel formüllerin python koduna çevrilmesi ile oluşturuldu. İlk olarak bir regrasyon modeli belirlendi. Kodun okunabilirliğini arttırmak için toplam sembolü ve sonrasındaki kısmı toplam fonksiyonu içerisinde hesaplandı ve ana fonksiyonda çağrıldı. Ana fonksiyonun içinde maliyet fonksiyonu oluşturuldu ve döngü her döndüğünde maliyet fonksiyonun değeri güncellendi. Her bir özellik için teta değeri bulundu. Maliyet fonksiyonu 0'a yaklaşana kadar bu işlem devam etti. En son hipotez fonksiyonu oluşturuldu ve içerisinde tahmin edilen ve gerçek fiyat karşılaştırıldı. İterasyon değerleri ve maliyet fonksiyonun değerleri dizi içerisine aktarıldı. Dizi içerisinde ki değerler grafikleştirildi.

Her iki soru için aynı işlemler uygulandı.