4.2 K-En Yakın Komşu

K-En Yakın Komşu (KNN), Denetimli Öğrenmede sınıflandırma ve regresyon için kullanılan algoritmalardan biridir. En basit makine öğrenmesi algoritması olarak kabul edilir. Diğer Denetimli Öğrenme algoritmalarının aksine, eğitim aşamasına sahip değildir. Eğitim ve test hemen hemen aynı şeydir. Tembel bir öğrenme türüdür. Bu nedenle, kNN, geniş veri setini işlemek için gereken algoritma olarak ideal bir aday değildir.

KNN ile temelde yeni noktaya en yakın noktalar aranır. K, bilinmeyen noktanın en yakın komşularının miktarını temsil eder. Sonuçları tahmin etmek için algoritmanın k miktarını (genellikle bir tek sayı) seçeriz. Model tanımada , en yakın komşu algoritması (kNN), sınıflandırma ve regresyon için kullanılan parametrik olmayan bir yöntemdir. Her iki durumda da, girdi, özellik alanında k en yakın eğitim örneklerinden oluşur. Çıktı, kNN'nin sınıflandırma veya regresyon için kullanılıp kullanılmayacağına bağlıdır:

- K-NN sınıflandırmasında , çıktı sınıf üyeliğidir. Bir nesne, komşularının çoğunluk oyuyla sınıflandırılır; nesne, en yakın komşuları arasında en yaygın olan sınıfa verilir (k , küçük bir pozitif bir tam sayı). Eğer k = 1 ise, nesne basitçe o en yakın komşunun sınıfına atanır.
- K-NN regresyonda çıktı, cismin özellik değeridir. Bu değer, en yakın komşularının değerlerinin ortalamasıdır.

KNN, örüntü tabanlı öğrenme veya tembel öğrenme türüdür; burada işlev sadece yerel olarak yaklaştırılır ve tüm hesaplama, sınıflandırmaya kadar ertelenir.

Hem sınıflandırma hem de regresyon için, komşuların katkılarına ağırlık koymak, böylece yakın komşuların ortalamaya daha uzak olanlardan daha fazla katkıda bulunmaları yararlı olabilir. Örneğin, ortak bir ağırlıklandırma şeması, her komşuya 1 / d ağırlığının verilmesini içerir; burada d komşuya olan uzaklıktır.

Komşular, sınıfın (kNN sınıflaması için) veya nesne mülk değerinin (kNN regresyonu için) bilindiği bir takım nesnelerden alınır. Bu, algoritma için ayarlanmış eğitim olarak düşünülebilir, ancak açık bir eğitim basamağı gerekmemektedir.

KNN algoritmasının bir özelliği, verilerin yerel yapısına duyarlı olmasıdır. Algoritma, başka popüler bir makine öğrenme tekniği olan k- means ile karıştırılmamalıdır.

K komşuları, tüm mevcut vakaları depolayan ve bir benzerlik ölçüsüne (ör. Mesafe fonksiyonları) dayalı yeni vakaları sınıflandıran basit bir algoritmadır. KNN, 1970'lerin başında halihazırda parametrik olmayan bir teknik olarak istatistiksel tahmin ve örüntü tanımada kullanılmıştır.

Bir örnek, komşularının çoğunluk oyuyla sınıflandırılır; bu olay, bir mesafe fonksiyonuyla ölçülen en yakın komşuları arasında en yakın olan sınıfa atanır. K = 1 ise, örnek yalnızca en yakın komşusunun sınıfına atanır.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

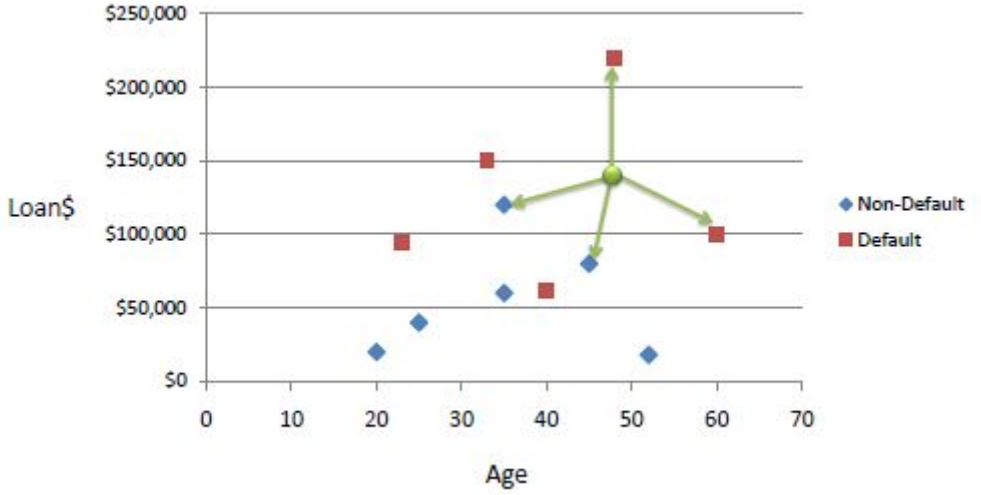
Ayrıca, üç mesafenin de yalnızca sürekli değişkenler için geçerli olduğuna dikkat edilmelidir. Kategorik değişkenler söz konusu olduğunda, Hamming mesafesi kullanılmalıdır. Ayrıca, veri kümesinde sayısal ve kategorik değişkenlerin bir karışımı olduğunda 0 ile 1 arasındaki sayısal değişkenlerin standardizasyonu meselesini ortaya çıkarmaktadır.

Hamming Distance

$$D_H = \sum_{i=1}^{k}|x_i - y_i|$$

K için en uygun değeri seçmek için önce verileri incelemek gerekir. Genel olarak, büyük bir K değeri, genel gürültüyü düşürdüğü için daha hassasdır, ancak garantisi yoktur. Çapraz doğrulama, K değerini doğrulamak için bağımsız bir veri kümesi kullanarak, geriye dönük olarak iyi bir K değerini belirlemenin başka bir yoludur. Tarihsel olarak, çoğu veri kümesi için optimal K, 3-10 arasında olmuştur. Bu, 1NN'den çok daha iyi sonuçlar üretir.

Örnek:
Kredi ile ilgili aşağıdaki verileri göz önünde bulundurun. Yaş(age) ve kredi(loan) iki sayısal değişken(tahmini) ve borç ödememek(default) de hedeftir(sonuç – yanıt).



Artık eğitim setini, Öklid uzaklığını kullanarak bilinmeyen bir durumu (Yaş = 48 ve Kredi = 142.000 ABD Doları) sınıflandırmak için kullanabiliriz. K = 1 ise, en yakın komşu, Varsayılan = Y ile eğitim setindeki son durumdur.

Age	Loan	Default	Distance	
25	$40,000	N	102000	
35	$60,000	N	82000	
45	$80,000	N	62000	
20	$20,000	N	122000	
35	$120,000	N	22000	2
52	$18,000	N	124000	
23	$95,000	Y	47000	
40	$62,000	Y	80000	
60	$100,000	Y	42000	3
48	$220,000	Y	78000	
33	$150,000	Y	8000	1
48	**$142,000**	**?**		

Euclidean Distance

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Doğrudan eğitim setinden mesafe ölçümlerinin hesaplanmasında bir büyük dezavantaj, değişkenlerin farklı ölçüm ölçeklerine sahip olması ya da sayısal ve kategorik değişkenlerin bir karışımı olması durumunda ortaya çıkmaktadır. Örneğin, bir değişken dolar cinsinden yıllık geliri temel alırken diğeri yıllara dayanıyorsa, gelirin hesaplanan mesafe üzerinde çok daha fazla etkisi olacaktır. Bir çözüm, aşağıda gösterildiği gibi eğitim setini standartlaştırmaktır.

Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	**0.61**	**?**	

Standardized Variable

$$X_s = \frac{X - Min}{Max - Min}$$

Aynı eğitim setindeki standart mesafeyi kullanarak, bilinmeyen verilerde şimdi farklı bir komşu bulmuş oldu.

Örnek olarak kullanacagimiz veri seti yapay ögrenme alaninin en popüler veri setlerinden “Iris” veri seti. Iris veri seti 3 Iris bitki türüne (Iris setosa, Iris virginica and Iris versicolor) ait, her bir türden 50 örnek olmak üzere toplam 150 örnek sayisina sahip bir veri setidir. Her bir örnek için 4 özellik tanimlanmistir: taç yaprak uzunlugu, taç yaprak genisligi, çanak yaprak genisligi, çanak yaprak uzunluğu. Yani veri setimizde, her bir bitki örnegi ayri bir gözlemi (örnegi) ifade ederken; bitki tür ismi bagimli degisken, bitkilerin ölçülen 4 temel özelligi ise bagimsiz degiskenleri ifade eder.

Biz sadece bitkilere ait ilk iki temel özelliği kullanacağız. Programın tamamı aşağıdadır.

calistir.py

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

```
reload(sys)
sys.setdefaultencoding('utf-8')

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from ciz import dagilim_grafigi, hipotez_grafigi
from k_en_yakin_komsu import KEnYakinKomsu

u""" 1. Veri Setinin Yuklenmesi """
print '1. Veri Seti Yukleniyor ...'
# Veri Setini Csv Dosyasindan Okuma
iris = pd.read_csv('iris.csv', header=None, names=['SepalLength',
'SepalWidth', 'PetalLength', 'PetalWidth', 'Class'])

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

# sadece sepal length ve sepal width ozellikleri kullanarak knn
algoritmasini kullanacagiz
# bu nedenle kullanmayacagimiz verileri silelim
del veri_seti['PetalLength']
del veri_seti['PetalWidth']

u""" 2. Dagilim Grafigi"""
print '2. Dagilim Grafigi Gosteriliyor ...'
dagilim_grafigi(veri_seti)

# Tahmin Etmeye Calisacagimiz Verileri Okuma
y = veri_seti['Class'].as_matrix()
del veri_seti['Class']

# Toplam Girdi Sayisi
m, n = veri_seti.shape

# Ongorucu Veriler, ilk iki degisken: sepal length ve sepal width
X = veri_seti.as_matrix()

u""" 3. Egitim-Test Setinin Bolunmesi"""
print '3. Egitim-Test Seti Bolunmesi Yapiliyor ...'
X_egitim, X_test, y_egitim, y_test = train_test_split(X, y,
test_size=0.30, random_state=False)

print ' Egitim Seti: ', len(X_egitim)
print ' Test Seti: ', len(X_test)

# X,y setleri birlestiriliyor, ve type=liste
```

```
y_egitim = np.reshape(y_egitim, (-1, len(y_egitim)))
egitim = np.concatenate((X_egitim, y_egitim.T), axis=1).tolist()

y_test = np.reshape(y_test, (-1, len(y_test)))
test = np.concatenate((X_test, y_test.T), axis=1).tolist()

u""" 4. KNN Algoritmasinin Calistirilmasi"""
print '4. KNN Algoritmasinin Calistiriliyor ...'

k_NN = KEnYakinKomsu(k=3)
tahminler = list()
for t in test:
    komsular = k_NN.komsulari_bul(egitim, t)
    sonuc = k_NN.tahmin_et(komsular)
    tahminler.append(sonuc)

u""" 5. KNN Algoritmasinin Basarisi Hesaplaniyor..."""
print '5. KNN Algoritmasinin Basarisi Hesaplaniyor ...'

dogru_tahmin_sayisi = 0
for i in range(len(test)):
    if test[i][-1] == tahminler[i]:
        dogru_tahmin_sayisi += 1

print ' Basarisi : ', dogru_tahmin_sayisi / float(len(test)) *
100.0, '%'

u""" 6. Test Setinin Grafigi ..."""
print '6. Test Setinin Grafigi Gosteriliyor...'

egitim_seti = pd.DataFrame(egitim, columns=['SepalLength',
'SepalWidth', 'Class'])
tahminler = np.reshape(np.array(tahminler), (-1, len(tahminler)))
test = np.concatenate((test, (tahminler).T), axis=1)
test_seti = pd.DataFrame(test, columns=['SepalLength',
'SepalWidth', 'Class', 'Prediction'])

hipotez_grafigi(egitim_seti, test_seti)
```

k_en_yakin_komsu.py

```
# coding=utf-8
""" k en yakin komsu
k nearest neighbor"""
```

```
import math

import operator


def oklit_uzakligi(x1, x2, uzunluk):
    mesafe = 0
    for i in range(uzunluk):
        mesafe += pow((x1[i] - x2[i]), 2)
    return math.sqrt(mesafe)


class KEnYakinKomsu():

    def __init__(self, k):
        self.k = k

    def komsulari_bul(self, egitim, test):

        uzakliklar = list()
        ozellik_sayisi = len(test) - 1
        for e in egitim:
            uzaklik = oklit_uzakligi(test, e, ozellik_sayisi)
            uzakliklar.append((e, uzaklik))
        uzakliklar.sort(key=operator.itemgetter(1))
        komsular = list()
        for i in range(self.k):
            komsular.append(uzakliklar[i][0])
        return komsular

    def tahmin_et(self, komsular):
        siniflandirma_oyu = dict()
        for x in range(len(komsular)):
            oy = komsular[x][-1]
            if oy in siniflandirma_oyu:
                siniflandirma_oyu[oy] += 1
            else:
                siniflandirma_oyu[oy] = 1
        sirali_oylar = sorted(siniflandirma_oyu.iteritems(),
key=operator.itemgetter(1), reverse=True)
        return sirali_oylar[0][0]
```

ciz.py

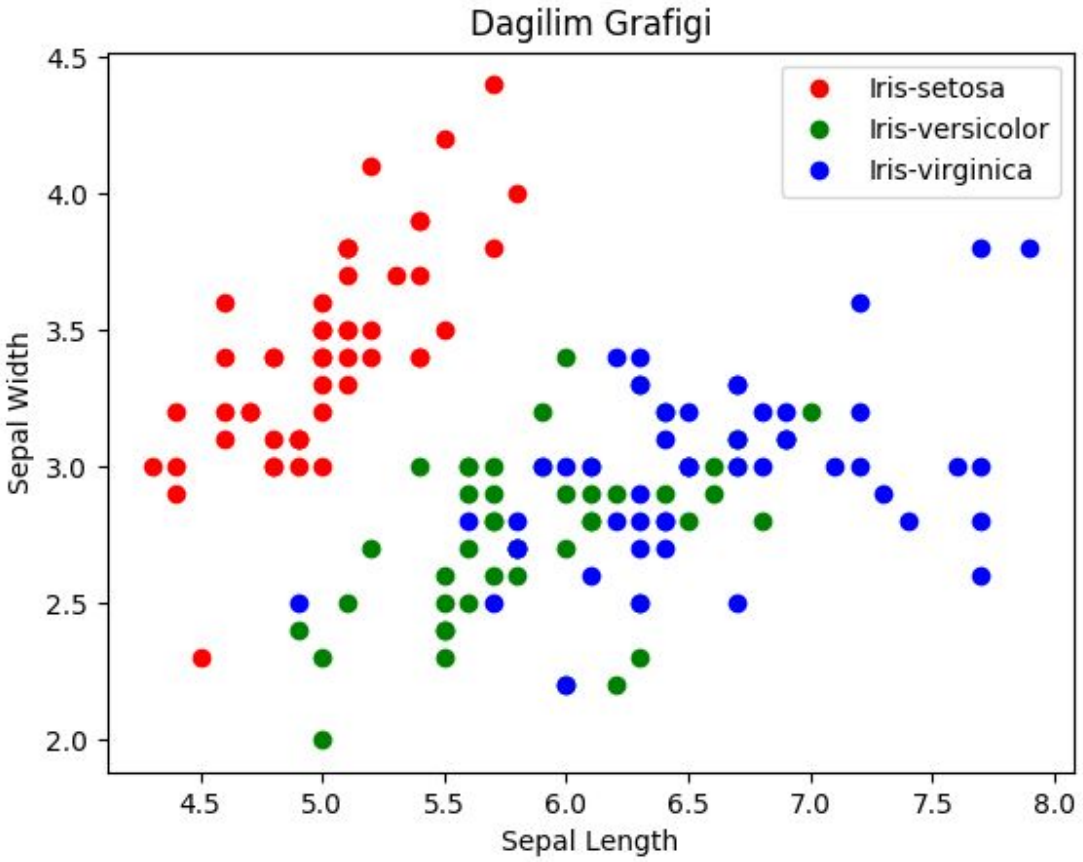
```
# coding=utf-8
""" Plotting Data
```

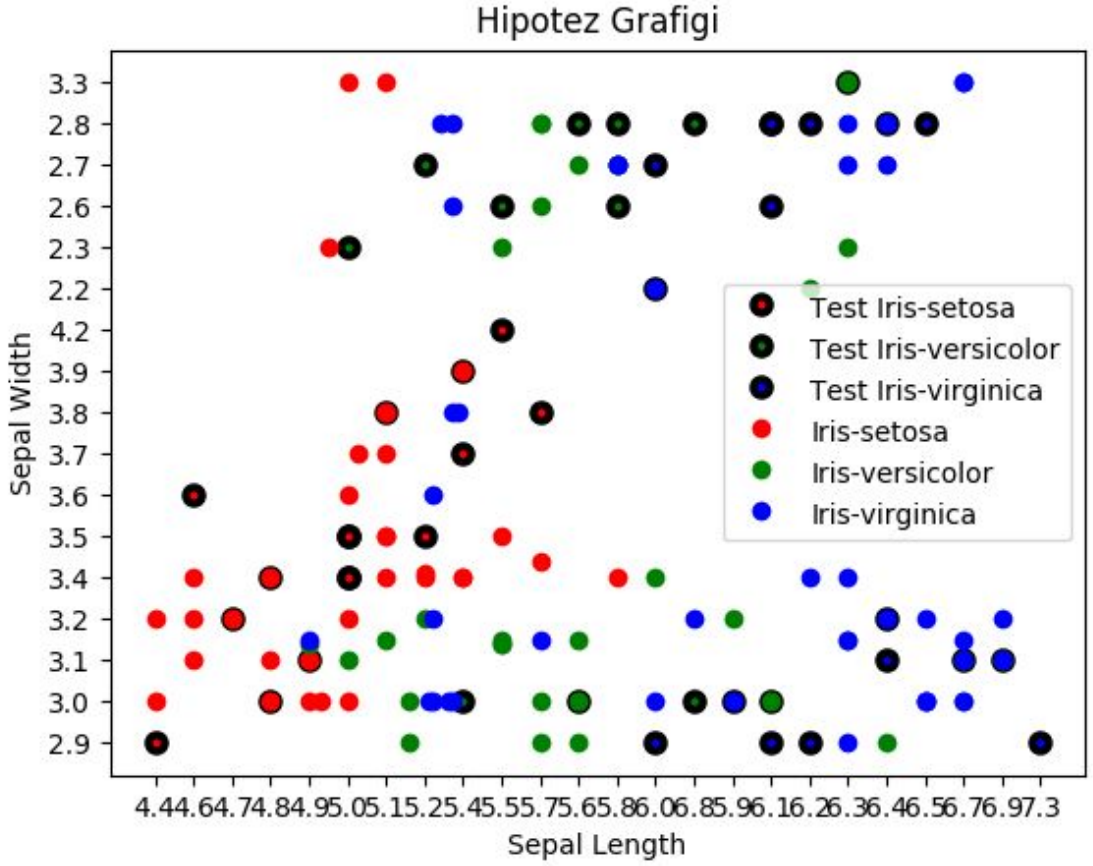
```
Veri Setinin Grafiginin Cizilmesi"""

import matplotlib.pyplot as plt


def dagilim_grafigi(veri_seti):
    """

    :param veri_seti: Dagilim Grafiginin Yapilacagi Veri Seti
    """

    setosa = veri_seti[veri_seti['Class'] == 'Iris-setosa']
    versi = veri_seti[veri_seti['Class'] == 'Iris-versicolor']
    virginica = veri_seti[veri_seti['Class'] == 'Iris-virginica']

    plt.figure()

    plt.plot(setosa['SepalLength'], setosa['SepalWidth'], 'o',
markeredgecolor='red', markerfacecolor='red')
    plt.plot(versi['SepalLength'], versi['SepalWidth'], 'o',
markeredgecolor='green', markerfacecolor='green')
    plt.plot(virginica['SepalLength'], virginica['SepalWidth'],
'o', markeredgecolor='blue', markerfacecolor='blue')
    plt.xlabel('Sepal Length')
    plt.ylabel('Sepal Width')
    plt.title('Dagilim Grafigi')
    plt.legend(['Iris-setosa', 'Iris-versicolor',
'Iris-virginica'])
    plt.show()


def hipotez_grafigi(egitim_seti, test_seti):
    """

    :param egitim_seti: Dagilim Grafiginin Yapilacagi Veri Seti
    """
    plt.figure()

    # test setinin gorsellestirilmesi
    setosa_ = test_seti[test_seti['Prediction'] == 'Iris-setosa']
    versi_ = test_seti[test_seti['Prediction'] ==
'Iris-versicolor']
    virginica_ = test_seti[test_seti['Prediction'] ==
'Iris-virginica']

    plt.plot(setosa_['SepalLength'], setosa_['SepalWidth'], 'o',
markeredgecolor='black', markerfacecolor='red',
             markeredgewidth=3)
```

```
    plt.plot(versi_['SepalLength'], versi_['SepalWidth'], 'o',
markeredgecolor='black', markerfacecolor='green',
             markeredgewidth=3)
    plt.plot(virginica_['SepalLength'], virginica_['SepalWidth'],
'o', markeredgecolor='black', markerfacecolor='blue',
             markeredgewidth=3)

    # egitim setinin gorsellestirilmesi
    setosa = egitim_seti[egitim_seti['Class'] == 'Iris-setosa']
    versi = egitim_seti[egitim_seti['Class'] == 'Iris-versicolor']
    virginica = egitim_seti[egitim_seti['Class'] ==
'Iris-virginica']

    plt.plot(setosa['SepalLength'], setosa['SepalWidth'], 'o',
markeredgecolor='red', markerfacecolor='red')
    plt.plot(versi['SepalLength'], versi['SepalWidth'], 'o',
markeredgecolor='green', markerfacecolor='green')
    plt.plot(virginica['SepalLength'], virginica['SepalWidth'],
'o', markeredgecolor='blue', markerfacecolor='blue')

    plt.xlabel('Sepal Length')
    plt.ylabel('Sepal Width')
    plt.title('Hipotez Grafigi')
    plt.legend(['Test Iris-setosa', 'Test Iris-versicolor', 'Test
Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
                'Iris-virginica'])

    plt.show()
```

Programın Çıktısı:

```
1. Veri Seti Yukleniyor ...
2. Dagilim Grafigi Gosteriliyor ...
3. Egitim-Test Seti Bolunmesi Yapiliyor ...
 Egitim Seti:  105
 Test Seti:  45
4. KNN Algoritmasinin Calistiriliyor ...
5. KNN Algoritmasinin Basarisi Hesaplaniyor ...
 Basarisi :  71.1111111111 %
6. Test Setinin Grafigi Gosteriliyor...
```


Dagilim Grafigi
Iris-setosa
Iris-versicolor
Iris-virginica
Sepal Width
Sepal Length
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Hipotez Grafigi
Test Iris-setosa
Test Iris-versicolor
Test Iris-virginica
Iris-setosa
Iris-versicolor
Iris-virginica
Sepal Width
Sepal Length
3.3
2.8
2.7
2.6
2.3
2.2
4.2
3.9
3.8
3.7
3.6
3.5
3.4
3.2
3.1
3.0
2.9
4.4 4.6 4.7 4.8 4.9 5.0 5.1 5.2 5.4 5.5 5.7 5.6 5.8 6.0 6.8 5.9 6.1 6.2 6.3 6.4 6.5 6.7 6.9 7.3