Yapay Zeka İle Kanser Tespiti

Proje Tanım

Bu proje yapay zeka uygulamalarında kullanılan bir algoritma olan kNN (k Nearest Neighborhood, En Yakın k Komşu) algoritması ile gerçeklenecektir. Gerçeklenen algoritma ile, UC Irvine Üniversitesi'nin sağlamış olduğu Göğüs Kanseri verileri işlenecektir. Hasta olup olmadığı belli olmayan bir kişinin verileri sisteme beslenerek, hastalık tahmini yapılacaktır.

Kullanılan Araçlar

- Microsoft Visual Studio Community

Kullanılan Veriler

- UC Irvine Üniversitesi veritabanından alınmış göğüs kanseri verileri

Veriler Hakkında Bilgi

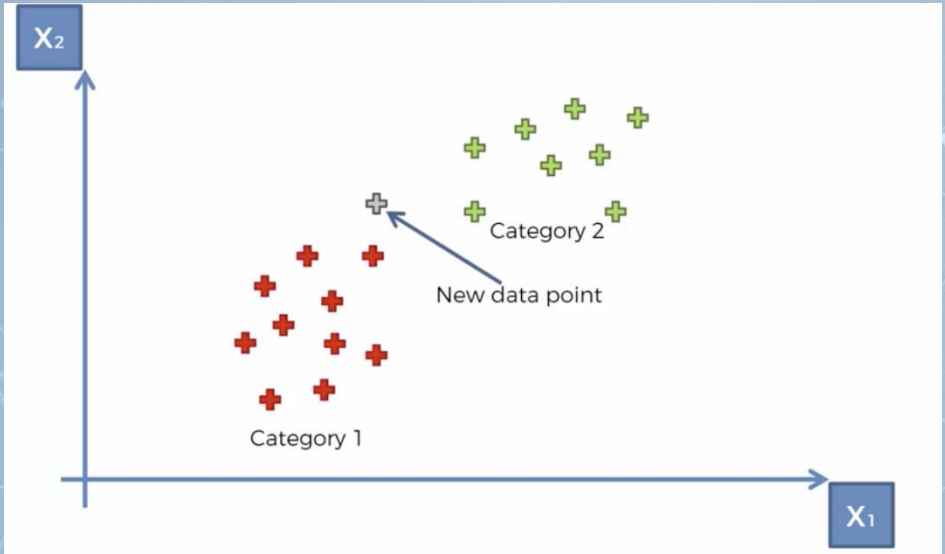
1) Hastadan alınan örneğin numarası
2) Clump Thickness
3) Uniformity of Cell Size
4) Uniformity of Cell Shape
5) Marginal Adhesion
6) Single Epithelial Cell Size
7) Bare Nuclei
8) Bland Chromatin
9) Normal Nucleoli
10) Mitoses
11) Sınıf Bilgisi (2 ise hasta değil, 4 ise hasta'yı ifade etmektedir)

- Eğitim verisetinde 600 örnek vardır.
- Test verisetinde ise 83 örnek vardır.

KNN (En yakın k komşu)

kNN sınıflandırma için kullanılan en basit algoritmalardan biridir. Bu algoritmada sınıflandırılmak istenen yeni verinin daha önceki verilerden k tanesine yakınlığına bakılır. Örneğin k = 3 için yeni bir veri sınıflandırılmak istensin. bu durumda eski sınıflandırılmış verilerden en yakın 3 tanesi alınır. Bu veriler hangi sınıfa dahilse, yeni veri de o sınıfa dahil edilir. Mesafe hesabında genelde öklit mesafesi (euclid distance) kullanılır.

KNN (En yakın k komşu)



Algoritmanın uygulanacağı veri, seçilen k sayısına göre (Tek sayı olmalı) iki kategoriden birine yerleştirilir.

İzlediğimiz Adımlar

- Eğitim ve test verilerini dosyadan okutma.
- Test verilerinin sonuçlarının bulunması için fonksiyon yapılması.
- Kullanıcıdan alınan değerlerin sonucunu bulmak için fonksiyon yapılması.
- Seçenekler için menü yapılması.

Test Verilerinin Sonuçların Bulan Fonksiyon

```
for (j = 0; j < 600; j++)
{
    Testsonuc = 0;

    for (z = 0; z < 9; z++)
    {

        int arrEgitimtemp = arrEgitim[j][z];
        int arrTesttemp = arrTest[a][z];

        Testsonuc = Testsonuc + pow((arrEgitimtemp - arrTesttemp), 2);

    }
    arrTestSonuc[j][0] = Testsonuc;
    HastalikDegeri = arrEgitim[j][9];
    arrTestSonuc[j][1] = HastalikDegeri;
}
```

- Eğitim ve Test verisinin Hasta Numarası ve Sınıf Bilgisi hariç her verileri için uzaklıkları bulan bir döngü yazdık. Bu döngüden çıkan uzaklık değerlerini arrTestSonuc dizisinin 0. indeksine atıyoruz, bulunan uzaklık değerinin hesaplanmasında kullanılan eğitim verisinin Sınıf Bilgisi de arrTestSonuc'un 1. indeksine kaydediliyor.

Test Verilerinin Sonuçların Bulan Fonksiyon

```
for (i = 0; i < 599; i++)
{
    min_idx = i;
    for (j = i + 1; j < 600; j++)
        if (arrTestSonuc[j][0] < arrTestSonuc[min_idx][0])
            min_idx = j;
    //Uzaklik
    TempVal = arrTestSonuc[min_idx][0];
    arrTestSonuc[min_idx][0] = arrTestSonuc[i][0];
    arrTestSonuc[i][0] = TempVal;
    //Hastalik degeri
    TempVal = arrTestSonuc[min_idx][1];
    arrTestSonuc[min_idx][1] = arrTestSonuc[i][1];
    arrTestSonuc[i][1] = TempVal;
}
```

- Bulduğumuz 600 uzaklık değerini Selection Sort algoritmasını kullanarak sıraladık. Algoritmada arrTestSonucun 0. indeksine bağlı olarak sıraladık.

Test Verilerinin Sonuçların Bulan Fonksiyon

```
hasta = 0;
hastadegil = 0;
for (i = 0; i < k; i++)
{
    TempVal2 = arrTestSonuc[i][1];
    if (TempVal2 == 2) hastadegil++;
    else hasta++;

}

printf("\n%d. Test Verisi Sonuclari: Hasta: %d, Hasta degil: %d\n\n", a + 1, hasta, hastadegil);

if (hastadegil > hasta) arrTestSonucDY[i] = 2;
else arrTestSonucDY[i] = 4;

if (arrTestSonucDY[i] == arrTest[a][9]) arrTestSonucDYCHAR[a] = 'D';
else arrTestSonucDYCHAR[a] = 'Y';
```

- En yakın k tane arrTestSonuc değerinin 1. indeksine bakarak eğer 2 ise hastadegil 4 ise hasta değerini arttırdık. Eğer hastadegil değerleri fazla ise arrTestSonucDY dizisine 2 , hasta değerleri fazla ise 4 yazdırdık. Sonra arrTestSonucDYdeki değeri Test verisindeki değer ile karşılaştırdık eğer aynılarsa arrTestSonucDYCHAR dizisine ‘D’, farklı ise ‘Y’ yazıyoruz.

Test Verilerinin Sonuçların Bulan Fonksiyon

```
for (int g = 0; g < 83; g++) //sonuclari ekrana yazdiriyor.
{
    printf("%d. veri == %c\n", g + 1, arrTestSonucDYCHAR[g]);
    if (arrTestSonucDYCHAR[g] == 'D') dogruSayisi++;
    else yanlisSayisi++;
}

printf("\nDogru sayisi: %.0f\nYanlis sayisi: %.0f\n", dogruSayisi, yanlisSayisi);
printf("Basari orani yuzde: %.2f\n", ((dogruSayisi * 100) / 83));
```

- Son olarak sonuçları yani arrTestSonucDYCHAR'ın içindek değerleri ve başarı oranını ekrana yazdırdık.

Kullanıcıdan alınan değerlerin sonucunu bulan fonksiyon

```
for (i = 0; i < 9; i++)
{
    printf("%d. degeri giriniz: ", i + 1);
    scanf_s("%d", &TempVal);
    arrKullaniciGirisi[i] = TempVal;
}
```

- İlk olarak kullanıcıdan 9 tane değer aldık ve bu değerleri arrKullaniciGirisi adlı diziye kaydettik.

Kullanıcıdan alınan değerlerin sonucunu bulan fonksiyon

```
for (j = 0; j < 600; j++)
{
    Testsonuc = 0;
    for (int z = 0; z < 9; z++)
    {
        Testsonuc = Testsonuc + pow((arrEgitim[j][z] - arrKullaniciGirisi[z]), 2);
        HastalikDegeri = arrEgitim[j][9];

        arrKullaniciUzaklik[j][0] = Testsonuc;
        arrKullaniciUzaklik[j][1] = HastalikDegeri;
    }
}
```

- Kullanıcıdan aldığımız 9 değerin eğitim verisindeki değerlere olan uzaklığını bulduk ve arrKullaniciUzaklik'ın 0. indeksine kaydettik. Uzaklıkları hesaplanan eğitim verisinin Sınıf Bilgisini de 1. indeksine kaydettik.

Kullanıcıdan alınan değerlerin sonucunu bulan fonksiyon

```
for (i = 0; i < 599; i++) // Siralama
{

    min_idx = i;
    for (j = i + 1; j < 600; j++)
        if (arrKullaniciUzaklik[j][0] < arrKullaniciUzaklik[min_idx][0])
            min_idx = j;

    TempVal = arrKullaniciUzaklik[min_idx][0];
    arrKullaniciUzaklik[min_idx][0] = arrKullaniciUzaklik[i][0];
    arrKullaniciUzaklik[i][0] = TempVal;

    TempVal = arrKullaniciUzaklik[min_idx][1];
    arrKullaniciUzaklik[min_idx][1] = arrKullaniciUzaklik[i][1];
    arrKullaniciUzaklik[i][1] = TempVal;
}
```

- Selection Sort kullanarak bu 600 değeri sıraladık.

Kullanıcıdan alınan değerlerin sonucunu bulan fonksiyon

```
for (i = 0; i < k; i++)
{
    TempVal2 = arrKullaniciUzaklik[i][1];
    if (TempVal2 == 2) hastadegil++;
    else hasta++;
}

if (hastadegil > hasta) printf("\nHasta degilsiniz. :D");
else printf("\nKansersiniz. :(");
```

- En yakın k tane değere bakarak, hastadegil ve hasta değerlerini bulduk. Bu değerlere göre ekrana eğer hastadegil fazla ise ‘‘Hasta Değilsiniz.’’ hasta değeri fazla ise ‘‘Kansersiniz.’’ yazdırdık.

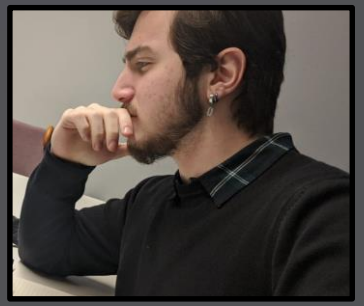
Seçenekler için menü hazırlanması

```
    //                    Menu kismi                        //
    printf("Merhabalar kNN algoritmasi ile kanser tespiti programina hosgeldiniz.\n");
kontrol1:
    printf("\nTest dosyasini hesaplamak icin => 1 giriniz.\nKullanicidan parametre alip hesaplamak icin => 2 giriniz.\nProgrami sonlandirmak icin => 0 giriniz.\nGiris: ");
    scanf_s("%d", &komut);
    if (komut == 1) {
        TestSonucBulma(arrEgitim, arrTest);
        printf("\nTekrar islem yapmak istiyorsaniz => 1 giriniz.\nProgrami sonlandirmak icin => 0 giriniz.\nGiris: ");
    kontrol3:
        scanf_s("%d", &komut);
        if (komut == 1) goto kontrol1;
        else if (komut == 0) goto kontrol2;
        else {
            printf("HATALI KOMUT TEKRAR GIRIN\n ");
            goto kontrol3;
        }
    }
    else if (komut == 2) {
        KullaniciGirisiHesaplama(arrEgitim);
        printf("\nTekrar islem yapmak istiyorsaniz => 1 giriniz.\nProgrami sonlandirmak icin => 0 giriniz.\nGiris: ");
    kontrol4:
        scanf_s("%d", &komut);
        if (komut == 1) goto kontrol1;
        else if (komut == 0) goto kontrol2;
        else {
            printf("HATALI KOMUT TEKRAR GIRIN\n ");
            goto kontrol4;
        }
    }
    else if (komut == 0) goto kontrol2;
    else {
        printf("HATALI KOMUT TEKRAR GIRIN");
        goto kontrol1;
    }

kontrol2:
    printf("\n\nYine bekleriz gorusmek uzere...\n\n");
    return 0;
}
```

Hazırlayanlar

Arda ALHAN



Evrim Arda KALAFAT

Teşekkürler