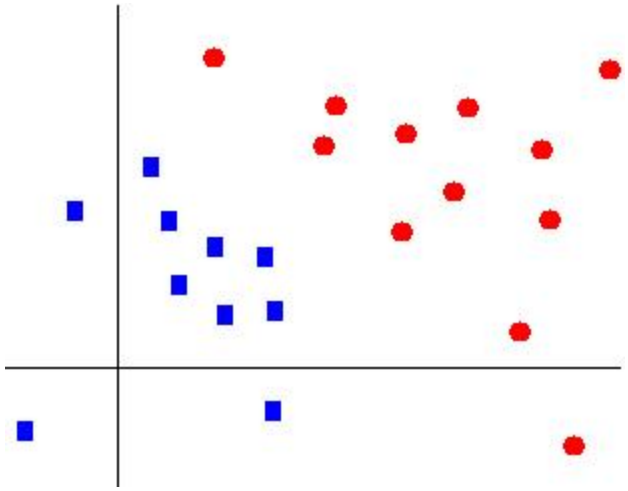
K Nearest Neighbor

Algorithm

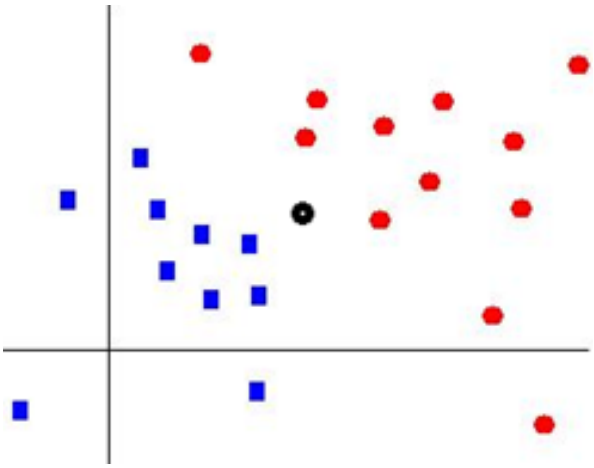
KNN algoritması, T. M. Cover ve P. E. Hart tarafından önerilen basitçe bir sınıflandırma(classification) algoritmasıdır. KNN algoritması nesneleri özelliklerine göre sınıflandırır. KNN algoritmasının çalışma prensibini basitçe anlatacak olursak, sınıflandırılmak istenilen yeni nesne, özelliklerine göre sınıflandırılmış en yakın K tane komşu nesnenin yakınlığına bakılarak sınıflandırılır.

Bir örnekle anlatacak olursak ;

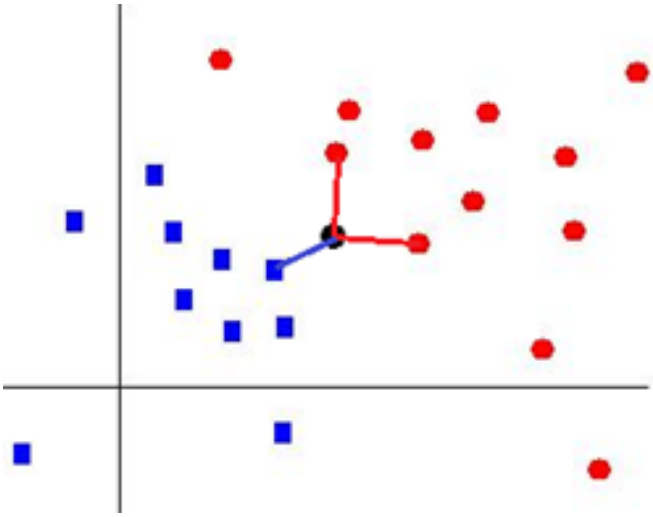
örneğin K =3 için yeni bir nesne sınıflandırılmak istensin. O halde eski sınıflandırılmış olan elemanlardan en yakın 3 tanesine bakılarak karar verilir. Bu en yakın 3 eleman hangi sınıfa daha çok dahil ise yeni nesnemiz de o sınıfa dahil edilir.



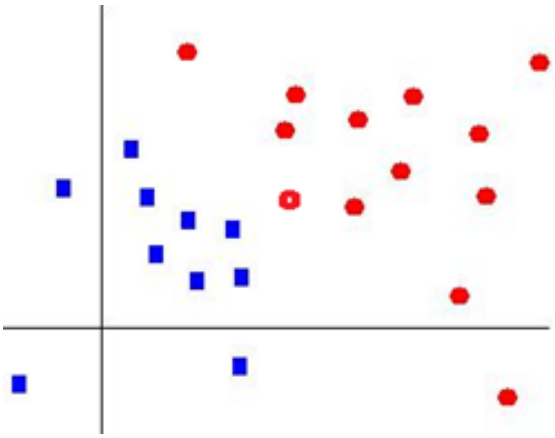
Örneğin yukardaki grafikte sınıflandırılmış olan elemanlara bir eleman daha ekleyelim ve sınıflandırma işlemini gerçekleştirelim.



İlk olarak yeni nesneye en yakın K=3 tane komşu seçilir.



Yukarda görüldüğü üzere eklenen yeni elemana en yakın 3 komşu elemanı seçtik ve bu komşulardan 2 tanesi kırmızı renk ile sınıflandırılmış 1 tanesi de mavi ile sınıflandırılmıştır. o halde yeni üyemizin de kırmızı renk ile sınıflandırılması gerekir.



KNN Algoritması; eski, basit ve gürültülü eğitim verilerine karşı dirençli olması sebebiyle en popüler makine öğrenme algoritmalarından biridir. Fakat bunun uzaklık hesabı yaparken bütün durumları sakladığından, büyük veriler için kullanıldığında çok sayıda bellek alanına gereksinim duymaktadır.

Senaryo: Dünya dışı bulunan bir elementin radyoaktif olup olmadığına karar veren bir program yazalım.

1.Adım;

Algoritmanın mesafe hesabı yapabilmesi için bir veri seti oluşturmamız gerekiyor.

Yazdığım veri setinde 1. Değer proton 2. Değer nötron sayısını 3. Değer ise elementin kararlı olup olmadığını belirtiyor

Proton	nötron	kararlılık özelliği (1 ise kararlı 0 ise radyoaktif-kararsız)	
1	0	1	
1	1	1	
1	2	0	
2	2	1	
3	3	1	
3	4	1	yazdığım veri setinde 490 element özellik
4	3	0	mevcut
4	5	1	
4	6	0	
5	5	1	
...	..	..	
..	..	..	
..	..	..	

Daha sonra keşfedilen elementin proton ve nötron değerini veri setindeki tüm elementlerle karşılaştırarak aralarındaki mesafeyi öklit hesabı ile ölçüyor ve ardından keşfedilen elemente en yakın K tane element seçilip kararlılıklarına bakılıyor. Sayıca üstün olan özellik seçilip keşfedilen elementin radyoaktiflik özelliğine karar veriliyor.

```
#include<stdio.h>
#include <stdlib.h>
#include<sys/time.h>
#include <math.h>
int veri_s[480][3],veri_sayi=0;
// veri_sayi  değeri elimizdeki sınıflandırılmış element sayisini gösterir
void Load_sample(){
        int i,j,k,f,counter;
        FILE *veri_seti=fopen("test_veri_seti.txt","r");// hazırladığım veri seti açilir
        if(veri_seti==NULL){
                printf("cannot open veri seti");// veri seti açılmaz ise error çiktisi
        }
        k=0;
        j=0;
        f=0;
        char tmp[3];
        char veri[10];
        while(fgets(veri,sizeof(veri),veri_seti)!=NULL){
 // bu döngüde değerler satır satır çekilip
                i=0;
   // integer a çevirilir

                        while(veri[f]!='\0'&&veri[f]!='\n'){

                                while(veri[f]!=32&&veri[f]!='\0'){
                                        tmp[i]=veri[f];
                                        i++;
                                        f++;
                                }
```

```
                        i=0;
                        if(veri[f]==32){
                                f++;
                        }
                        veri_s[j][k]=atoi(tmp);
                        if(veri_s[j][0]>100){
                                veri_s[j][0]=veri_s[j][0]-veri_s[j][k]%10;
                                veri_s[j][0]=veri_s[j][0]/10;
                        }
                if(veri_s[j][1]>200){
                        veri_s[j][k]=veri_s[j][k]-veri_s[j][k]%10;
                        veri_s[j][k]=veri_s[j][k]/10;
                }
                k++;
                }
        if(veri[f]!='\n'){
        f=0;
        k=0;
        j++;
        }
    }
}
void Load_test(int test_int[][30]){
 // bu fonksiyonda ise sınıflandırılacak elementlerin veri seti ara belleğe alınıyor
    int i,j,k,f;
    char test_dat[10];
    char tmp[3];
    FILE *test=fopen("kesif_veri_seti.txt","r");
    if(test==NULL){
            printf("test file cannot open");
    }
    k=0;
    i=0;
    j=0;
    f=0;
    while(fgets(test_dat,sizeof(test_dat),test)!=NULL){
```

```
64              while(test_dat[f]!='\n'&&test_dat[f]!='\0'){
65                      while(test_dat[f]!=32&&test_dat[f]!='\0'){
66                              tmp[i]=test_dat[f];
67                              i++;
68                              f++;
69                      }
70                      if(test_dat[f]==32){
71                              f++;
72                      }
73                      i=0;
74                      test_int[j][k]=atoi(tmp);
75                      if(test_int[j][0]>100){
76                                      test_int[j][0]=test_int[j][0]-test_int[j][k]%10;
77                                      test_int[j][0]=test_int[j][0]/10;
78                              }
79                      if(test_int[j][1]>200){
80                              test_int[j][1]=test_int[j][1]-test_int[j][1]%10;
81                              test_int[j][1]=test_int[j][1]/10;
82                      }
83                      k++;
84              }
85              if(test_dat[f]!='\n'){
86                      f=0;
87                      k=0;
88                      j++;
89              }
90      }
91      /*for(i=0;i<30;i++){
92              for(j=0;j<3;j++){
93                      printf("%d  ",test_int[i][j]);
94              }
95              printf("\n");
96      }*/
97 }
98 int decision(int dist[][480],int k){
99 //bu fonksiyonda en yakın mesafediki komşu elementler
```

```
100  //k sayısı kadar seçilip sınıflandırma işlemi yapılır
101     float strange,normal;
102     float percentage;
103     int i,j;
104     strange=0;
105     normal=0;
106     for(i=0;i<k;i++){
107             if(veri_s[dist[1][i]][2]==0){
108                     strange++;
109             }else if(veri_s[dist[1][i]][2]==1){
110                     normal++;
111             }
112     }
113     if(strange>normal){ //strange kararsızlığını belirtiyor
114             percentage=normal/k*100;
115             printf("this element is a  radioactive element\n");
116             printf("I am srure about %f  percentage\n",percentage);
117             printf("strange vealude is %f normal value is %f",strange,normal);
118
119             return 0;
120     }else{
121             percentage=normal/k*100;
122             printf("this element is a non radioactive element\n");
123             printf("I am srure about %f  percentage\n",percentage);
124             printf("strange vealude is %f normal value is %f",strange,normal);
125             return 1;
126     }
127 }
128 int distance(int p,int n,int k){
129 //bu fonksiyonda yeni elementin her bir eski elemente uxaklığı belirlenir
130     int i,j,min,ind_2,tmp,tmp_1;
131     int dist[2][480];//2. satır her bir elementin sırasını tutar
132     for(i=0;i<480-1;i++){
133 //böylece en kısa mesafenin hengi elemntler arası olduğunu bulabiliriz
134             dist[0][i]=(p-veri_s[i][0])*(p-veri_s[i][0])+(n-veri_s[i][1])*(n-veri_s[i][1]);
135     /* pisagor hesabı ile mesafe ölçümü */
```

```
136             dist[1][i]=i;
137     }
138     //sırada mesafeyi sıralamak var
139     min=dist[0][0];
140     for(i=0;i<479;i++){
141             for(j=i+1;j<479;j++){
142                     if(dist[0][i]>dist[0][j]){
143                             tmp=dist[0][i];
144                             dist[0][i]=dist[0][j];
145                             dist[0][j]=tmp;
146                             tmp_1=dist[1][i];
147                             dist[1][i]=dist[1][j];
148                             dist[1][j]=tmp_1;
149                     }
150             }
151     }
152     i=decision(dist,k);
153     return i;//karar verme fonksiyonuna gider
154 }
155 void time_graph(int test_int[][30],int k){
156  //bu fonksiyonda çalışma zamanı grafiği bastırılıyor
157     int i,j,q,a;
158     j=20;
159     float counter,micro,d;
160     struct timeval start,stop;
161     printf("Siniflandirilacak test_elements iterasyon k Micro
    sec.    Accurracy\n");
162     for(i=100;i<1000;i+=100){
163             gettimeofday(&start,NULL);
164             for(a=0;a<i;a++){
165                     for(q=0;q<j;q++){
166                     if(distance(test_int[q][0],test_int[q][1],k)){
167                     if(test_int[q][2]==1){
168                             counter++;
169                      }}else{
170                     if(test_int[q][2]==0){
```

```
171                             counter++;
172                     }
173                     }
174                     }
175             }
176             gettimeofday(&stop,NULL);
177     int
    total=(((stop.tv_sec-start.tv_sec)*1000000)+(stop.tv_usec-start.tv_usec));
178     printf("%d\t\t\t  580 \t\t  %d \t\t%d \t %d
    \t%f\n",q,a,k,total,counter/(q*a)*100);
179     if(i>800&&j==20){;
180     i=100;
181     j=25;
182     }else if(i>800&&j==25){
183     i=100;
184     j=30;
185     }else if(i>800&&j==30&&k!=5){
186             i=100;
187             j=20;
188             k=5;
189     }
190             counter=0;
191     }
192     printf("el X itr X k   Zaman Grafiği (tek birim yaklasik 0.1 sn)\n\n");
193     for(i=100;i<1000;i+=100){
194             gettimeofday(&start,NULL);
195             for(a=0;a<i;a++){
196                     for(q=0;q<j;q++){
197                             if(distance(test_int[q][0],test_int[q][1],k)){
198                                     if(test_int[q][2]==1){
199                                             counter++;
200                                     }
201                             }else{
202                                     if(test_int[q][2]==0){
203                             counter++;
204                                     }
```

```
205                             }
206                     }
207             }
208             gettimeofday(&stop,NULL);
209     int
    total=(((stop.tv_sec-start.tv_sec)*1000000)+(stop.tv_usec-start.tv_usec));
210             micro=(float)total/1000000;
211             total=round(micro);
212             printf("%d X %d X %d = ",q,a,k);
213             for(d=0;d<micro;d+=0.1){
214                     printf("#");
215             }
216             printf("\n");
217             if(i>800&&j==20){;
218             i=100;
219             j=25;
220             printf("\n");
221             }else if(i>800&&j==25){
222             i=100;
223             j=30;
224             printf("\n");
225             }else if(i>800&&j==30&&k!=5){
226                     i=100;
227                     j=20;
228                     k=5;
229                     printf("\n");
230             }
231             counter=0;
232
233             }
234
235 }
236 int main(){
237     int p,n,k;
238     int i,j;
239     int test_int[3][30];
```

```
240     char graph[200][200];
241     for(i=0;i<200;i++){
242             for(j=0;j<200;j++){
243                     graph[i][j]=' ';
244             }
245     }
246     printf("k degerini giriniz");
247     scanf("%d",&k);
248     Load_sample();
249  // bu fonksiyonda hazırlanan veri test seti ara belleğe alınıyor
250     Load_test(test_int);
251  // bu fonksiyonda ise sınıflandırılacak elementlerin veri seti ara belleğe alınıyor
252  //time_graph(test_int,k);//bu fonksiyonda ise zaman karmaşıklığı için grafik
    bastırılıyor
253     printf("yeni elemtin proton sayisini giriniz");
254     scanf("%d",&p);
255     printf("yini elementin notron sayisini giriniz");
256     scanf("%d",&n);
257     if(distance(p,n,k)){
258             printf("this element is non radioactive\n");
259     }else{
260             printf("this element is radioactive \n");
261     }
262     for(i=0;i<480;i++){
263             if(veri_s[i][2]==0){//atomların kararlılık kusagı grafik haline getiriliyor
264                     graph[veri_s[i][1]][veri_s[i][0]]=48;
265             }if(veri_s[i][2]==1){
266                     graph[veri_s[i][1]][veri_s[i][0]]=49;
267             }
268     }
269     for(i=70;i>=0;i--){
270             for(j=0;j<100;j++){
271                     if(i==n&&j==p){                         //grafik bastırılıyor
272                             printf("E");
273                     }else{
274                             printf("%c ",graph[i][j]);
```

```
275                 }
276
277             }
278             printf("\n");
279     }
280
281 }
```

Aşşağıdaki tabloda ise sırasıyla sınıflandırılacak element sayısını, test eleman sayısını iterasyon sayısını,K değeri,geçen süre ve doğruluk oranını gösteriyor.

Siniflandirilacak	test_elements	iterasyon	k	Micro sec.	Accurracy
20	580	100	3	667147	80.000000
20	580	200	3	1362179	80.000000
20	580	300	3	2093200	80.000000
20	580	400	3	2640471	80.000000
20	580	500	3	3287607	80.000000
20	580	600	3	4010155	80.000000
20	580	700	3	4679178	80.000000
20	580	800	3	5355909	80.000000
20	580	900	3	5924086	80.000000
25	580	200	3	1705761	80.000000
25	580	300	3	2568565	80.000000
25	580	400	3	3403452	80.000000
25	580	500	3	4226319	80.000000
25	580	600	3	5109213	80.000000
25	580	700	3	5968021	80.000000
25	580	800	3	6826847	80.000000
25	580	900	3	7773648	80.000000
30	580	200	3	1993339	80.000000
30	580	300	3	3007977	80.000000
30	580	400	3	3994663	80.000000
30	580	500	3	4993296	80.000000
30	580	600	3	5992028	80.000000
30	580	700	3	6998650	80.000000
30	580	800	3	8005302	80.000000
30	580	900	3	9179369	80.000000
20	580	200	5	1327234	65.000000
20	580	300	5	1971870	65.000000
20	580	400	5	2634690	65.000000
20	580	500	5	3297608	65.000000
20	580	600	5	3890801	65.000000
20	580	700	5	4513969	65.000000
20	580	800	5	5173088	65.000000
20	580	900	5	5804246	65.000000
25	580	200	5	1697731	64.000000
25	580	300	5	2520633	64.000000
25	580	400	5	3415470	64.000000
25	580	500	5	4258277	64.000000
25	580	600	5	5057277	64.000000
25	580	700	5	5908107	64.000000
25	580	800	5	6710999	64.000000
25	580	900	5	7541923	64.000000
30	580	200	5	1957385	63.333332
30	580	300	5	2964040	63.333332
30	580	400	5	3918764	63.333332
30	580	500	5	4905446	63.333332
30	580	600	5	5888133	63.333332
30	580	700	5	6918789	63.333332
30	580	800	5	7932489	63.333332
30	580	900	5	9019915	63.333332

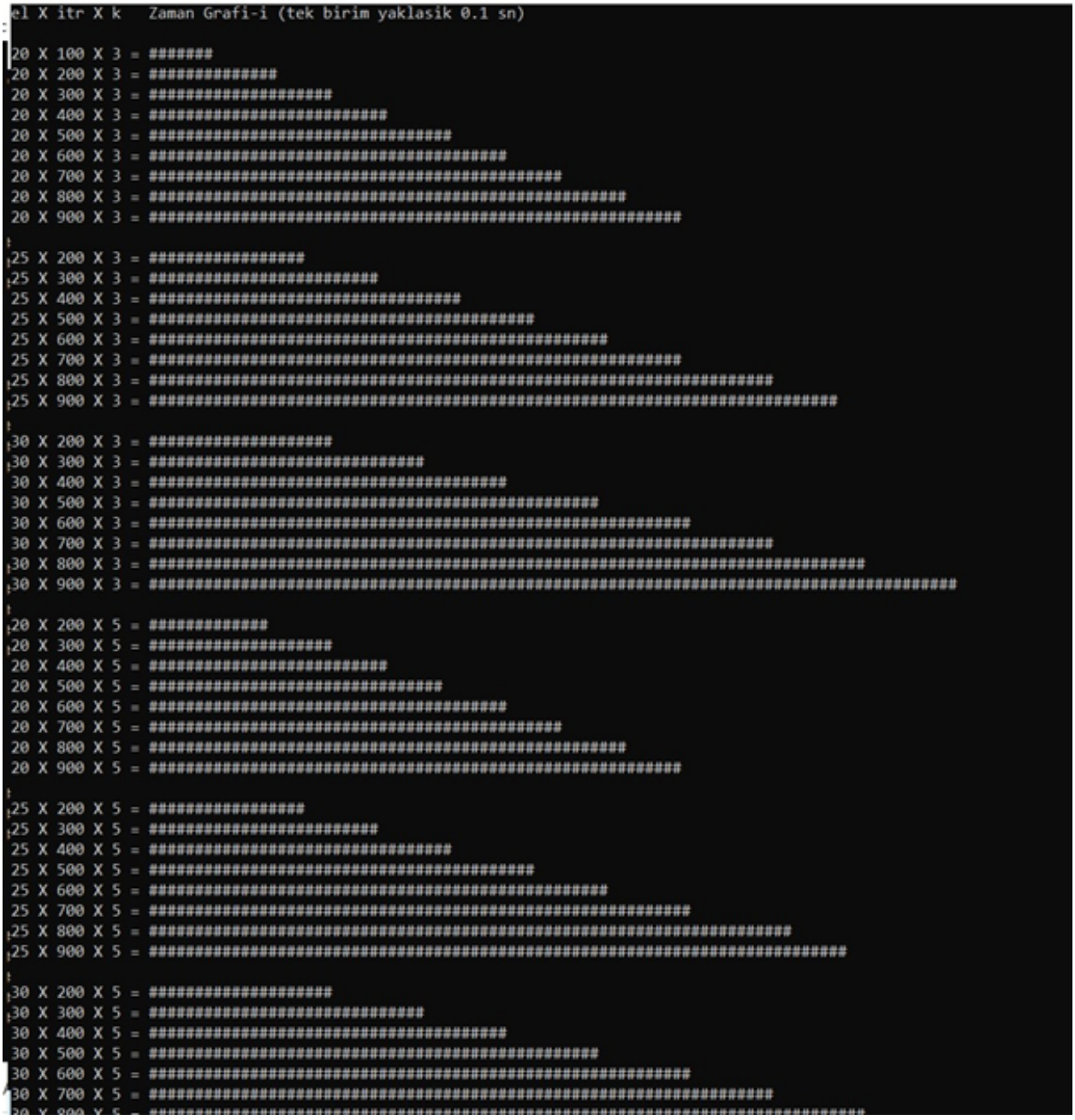
Yukardaki tablodan yola çıkacak olursak algoritmanın çalışma zamanının tüm etmenlere bağlı olduğunu görebiliriz. Sınıflandırılacak element sayısına,test veri setindeki element sayısına,tabiki iterasyon sayısına ve çok az da olsa k değerine bağlı olduğunu görüyoruz.

Bu değerlerin hepsi ile doğru orantılı yani bu değerlerden herhangi biri artıkça çalışma zamanı da artıcaktır.

Fakat bu tabloda gözümüze ise çarpan K değerinin artmasıyla birlikte doğruluk oranının

düşmesidir. K değeri artıkça algoritmanın da karar vermesi zorlaşıyor.

Aşşağıdaki tabloda ise çalışma zamanının farklı değerler ve iterasonlara ait bir grafiği gösteriliyor.



Barış Karataş 18011029
youtube video link : https://www.youtube.com/watch?v=FeR0DQJ-Cvs

kaynakça;
http://bilgisayarkavramlari.sadievrenseker.com/2008/11/17/knn-k-nearest-neighborhood-en-yakin-k-komsu/

https://www.geeksforgeeks.org/k-nearest-neighbours/

https://erdincuzun.com/makine_ogrenmesi/k-nn-algoritmasi/