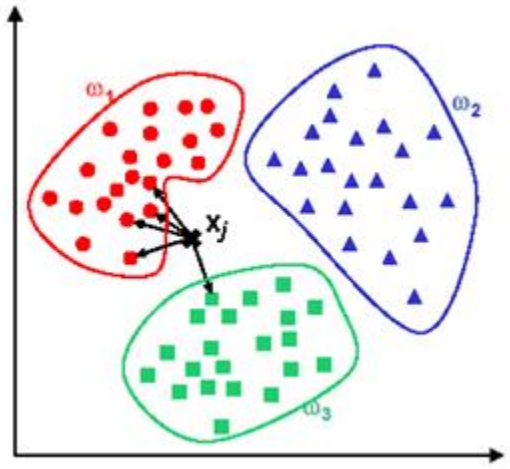
کلاس بندی با استفاده از الگوریتم K Nearest Neighbor

K نزدیکترین همسایه(KNN) یک الگوریتم یادگیری می باشد که در روش بازشناسی الگو طی چندین دهه مطالعه شده است. KNN به عنوان یکی از کاراترین روش ها شناخته شده است، و مطالعات زیادی KNN را روی اسناد آزمایشی به کار برده اند، که این مطالعات پیشنهاد می کنند که KNN با SVM، از روش هایی مانند تقریب خطی کوچکترین مربعات، Naive Bayes و شبکه های عصبی بهتر عمل می نماید.



KNN یک روش کارا و ساده برای دسته بندی می باشد. ایده KNN مطابق زیر است: یک سند آموزشی برای دسته بندی وجود دارد، الگوریتم K نزدیک در میان سندهای آموزشی پیش دسته بندی شده، بر اساس یک معیار شباهت پیدا کرده و دسته های این K همسایه نزدیک برای پیش بینی دسته سند آزمایشی به وسیله امتیازدهی سندهای هر دسته منتخب، استفاده می شود. اگر بیشتر از یک همسایه به مجموعه تعلق داشته باشد، مجموع امتیاز آنها به عنوان وزن آن دسته استفاده می شود و دسته با بالاترین امتیاز به سند مورد آزمایش انتساب می یابد.

مشکلاتی که در روش KNNوجود دارد، یکی تعیین مقدار k می باشد و برای تعیین آن باید یک سری آزمایشات با مقادیر مختلف k انجام شود، تا بهترین مقدار برای k را تعیین کند. عیب دیگر این روش، پیچیدگی زمانی محاسباتی مورد نیاز برای پیمایش همه سندهای آموزشی می باشد.

میزان k برای دسته بندی بستگی به تعداد و حجم داده ها دارد ولی برای این پروژه بین ۷ تا ۱۵ خوب است.

نحوه ی پیاده سازی پروژه:

برای پیاده سازی این پروژه، از ۴ کلاس استفاده شده که همگی از نوع `public` تعریف شده اند. کلاس اصلی در این برنامه کلاس `KnnClassifer` هست.

ابتدا در تابع main یک instance از این کلاس ساخته شده و constructor آن فراخوانی میشود. در آنجا از کاربر خواسته می شود تا از طریق کنسول آدرس فایل train و test خود را وارد سازد.

```
Scanner console = new Scanner(System.in);
       System.out.print("enter the address of training data file please:");
         String train_doc = console.next();
        System.out.print("enter the address of test data file please:");
              String test = console.next();
```

با استفاده از InputStream می توانیم اطلاعات را از فایل بخوانیم. بدین صورت که کلیه اطلاعات را خوانده و در متغیر is قرار می دهد و سپس در بافری ریخته و با بافر خط به خط فایل را می خواند و هر سطر خوانده شده از فایل را به عنوان آرگومان ورودی به تابع `fold` می فرستد..

```
try{
                InputStream is = new FileInputStream(train_doc);
                InputStreamReader instrm = new InputStreamReader(is);
                BufferedReader br = new BufferedReader(instrm);
                String strLine;
                String strLine2;
                InputStream is2= new FileInputStream(test_doc);
                InputStreamReader instrm2 = new InputStreamReader(is2);
                BufferedReader br2 = new BufferedReader(instrm2);

              for(int i=0; i<1500; i++)
                {strLine = br.readLine();
                fold(strLine);}
```

در تابع `fold` ابتدا هر سطری را با `delimiter` "،" `Tokenize` می نماییم و تک تک مقادیر را که به صورت رشته هستند به مقدار عددی اعشاری تبدیل می نماییم.

```
private void fold(String strLine) {
       j=0;
       StringTokenizer st = new StringTokenizer(strLine, ",");
             String line[] = new String[7];
        while(st.hasMoreTokens()){
```

```
        str = st.nextToken();
        line[j] = str;
    if(j<6) {
        train[train_row_num][j]=Float.parseFloat(line[j]);
```

هر سطر که اکنون در آرایه ۷ تایی `line` ذخیره هست را با تبدیل بعد هفتم آنها به مقادیر عددی ۱،۲،۳و۴ ذخیره می نماییم تا کل مقادیر را بتوانیم در آرایه ای از جنس `float` نگه داری می کنیم و کل فایل `train` را در یک ماتریس دوبعدی `train` ذخیره می نماییم.

```
        if(t==0)
         fold1[row_num][j] = Float.parseFloat(line[j]);
        else if(t==1)
         fold2[row_num][j] = Float.parseFloat(line[j]);
        else if(t==2)
         fold3[row_num][j] = Float.parseFloat(line[j]);
            }
```

فایل data را که به عنوان داده آموزشی برای سیستم در نظر گرفته شده را به سه قسمت تقسیم می نماییم(3-fold cross validation).

```
    if(j==6){
          if(line[6].equals("acc"))
           train[train_row_num][6]=1;
          else if(line[6].equals("unacc"))
           train[train_row_num][6]=2;
          else if(line[6].equals("good"))
           train[train_row_num][6]=3;
          else if(line[6].equals("vgood"))
           train[train_row_num][6]=4;

          if(t==0){
              if(line[6].equals("acc"))
                fold1[row_num][6]=1;
              else if(line[6].equals("unacc"))
                fold1[row_num][6]=2;
              else if(line[6].equals("good"))
                fold1[row_num][6]=3;
              else if(line[6].equals("vgood"))
                fold1[row_num][6]=4;}

          else if(t==1){
                if(line[6].equals("acc"))
                 fold2[row_num][6]=1;
               else if(line[6].equals("unacc"))
                 fold2[row_num][6]=2;
               else if(line[6].equals("good"))
                 fold2[row_num][6]=3;
               else if(line[6].equals("vgood"))
                 fold2[row_num][6]=4;}

          else if(t==2){
                 if(line[6].equals("acc"))
                  fold3[row_num][6]=1;
                 else if(line[6].equals("unacc"))
                  fold3[row_num][6]=2;
```

```
            else if(line[6].equals("good"))
             fold3[row_num][6]=3;
            else if(line[6].equals("vgood"))
             fold3[row_num][6]=4;}        }

                j++;}

          train_row_num++;

          if(row_num < 499)
                row_num++;
          else if(row_num == 499)
          {row_num = 0;
          t++;}
                }
```

برای محاسبه ی میزان صحت پیش بینی این الگوریتم با استفاده از الگوریتم cross validation، و هر بار یکی از آنها را به عنوان آرایه دوبعدی test و دو تای دیگر را پس از ترکیب، به عنوان آرایه دوبعدی train استفاده می نماییم و تابع process را با این مقادیر test و train فراخوانی نموده و به ازای مقدار k از ۱ تا ۱۵، مقادیر precision، recall، *accuracy* و f_measure را محاسبه می نماییم.

```
for(int k=1;k<=15;k++){

 precision = Process(k,fold1, merge(fold2, fold3, train1),1) +
Process(k,fold2, merge(fold1, fold3, train2),1)+Process(k,fold3, merge(fold1,
fold2, train3),1);
 recall = Process(k,fold1, merge(fold2, fold3, train1),2) + Process(k,fold2,
merge(fold1, fold3, train2),2)+Process(k,fold3, merge(fold1, fold2,
train3),2);
 accuracy = Process(k,fold1, merge(fold2, fold3, train1),3) +
Process(k,fold2, merge(fold1, fold3, train2),3)+Process(k,fold3, merge(fold1,
fold2, train3),3);
       System.out.println("k:" + k + "   ,precision: " + (precision/3)*100 +
"%" + "  ,recall: " + (recall/3)*100+"%" +"  ,accuracy:"+(accuracy)/3*100+"%"
+ "  ,f_measure: " + (((recall*precision)/((recall+precision)/2))/3)*100 +
"%");
             }
```

در این تابع، ماتریس test را سطر به سطر به همراه ماتریس train و مقدار r که شماره مربوط به کلاس هر سطر از testهست را به عنوان ورودی به توابع reverseDist و sqReverseDist و nDist می دهیم تا محاسبه فاصله بین هر سطر از ماتریس train و test را انجام دهند و سپس مقادیر برگردانده شده توسط آنها که آرایه ای از کلاس نگه دارنده مقدار فاصله و برچسب کلاس آنها می باشد، به عنوان ورودی به تابع sort هدایت می شوند تا به ترتیب صعودی مشخص شوند. مقادیر مربوط به هر کدام از این مدل ها در آرایه ای از کلاس ها ذخیره می شود که حاوی دو مقدار dist و lable می باشد.

```
private float Process(int k, float test[][], float[][] train, int e) {//TODO

                float r=0;
```

```
                float[] test_rows = new float[7];
            for(int i=0;i<test.length;i++){
                for(int j=0;j<7;j++){
                test_rows[j] = test[i][j];

                }

                r = test_rows[6];
                rev = sort(reverseDist(train, test_rows , r),
reverseDist(train, test_rows , r).length);
                list = new float[k];
            for(int c=0;c<k;c++){
                list[c] = rev[c].lable;
               }

             if(find_max(list)==r)
               {relevant++;
                  if(find_max(list)!=0)
                  tp++;}
              else if(find_max(list) != r && find_max(list)!=0)fp++;
                        retrieved = fp+tp;

            }
            if(e==1)
                return evaluatePrec(tp, retrieved);
            else if(e==2)
                return evaluateRecall(tp, relevant);
            else if(e==3)
                return evaluateAcc(tp, fp, retrieved);
            else return 0;
                }
```

در انتهای این تابع سه تابع محاسبه ی precision، Recall و accuracy فراخوانی می شوند و مقدار f_measure نیز به کمک دو مقدار precision و recall محاسبه میگردد.

```
    private void nDist(float[][] train, float[] test_rows, float r) {
                float distance3[] = new float[train.length];
                float[] ndist = new float [train.length];
                float sub[] = new float[6];
                float math[] = new float[6];
                float res = 0;
                nDist[] rev = new nDist[train.length];
                for(int i=0;i<train.length;i++){
                      rev[i] = new nDist();
                      for(int j=0;j<6;j++){
                            sub[j] =test_rows[j] - train[i][j];
                      math[j]=(float) Math.pow((sub[j]), 2);
                      res =
math[0]+math[1]+math[2]+math[3]+math[4]+math[5];
                      }
                      distance3[i] = (float) (Math.sqrt(res));
                      ndist[i]=1-distance3[i];
                }
    }
```

```
            private sqReverseDist[] sqReverseDist(float[][] train, float[]
test_rows, float r) {
                  float distance2[] = new float[train.length];
                  float[] revd = new float [train.length];
                  float[] sqrevd = new float [train.length];
                  float sub[] = new float[6];
                  float math[] = new float[6];
                  float res = 0;
                  sqReverseDist[] sqrev = new sqReverseDist[train.length];
                  for(int i=0;i<train.length;i++){
                        sqrev[i] = new sqReverseDist();
                        for(int j=0;j<6;j++){
                              sub[j] =test_rows[j] - train[i][j];
                        math[j]=(float) Math.pow((sub[j]), 2);
                        res =
math[0]+math[1]+math[2]+math[3]+math[4]+math[5];
                        }
                        distance2[i] = (float) (Math.sqrt(res));
                        revd[i] = 1/distance2[i];
                        sqrevd[i] = (float) Math.pow(revd[i], 2);
                        sqrev[i].lable = r;
                        sqrev[i].dist =  distance2[i];
                        //sqrev[i].sq_rev_dist = (float)
Math.pow(sqrevd[i],2);
                  }
                  return sqrev;
            }
      private reverseDist_value[] reverseDist(float[][] train, float[]
test_rows, float r) {
            float distance[] = new float[train.length];
            float[] revd = new float [train.length];
            float sub[] = new float[6];
            float math[] = new float[6];
            float res = 0;
            reverseDist_value[] rev = new reverseDist_value[train.length];
            for(int i=0;i<train.length;i++){
                  rev[i] = new reverseDist_value();
                  for(int j=0;j<6;j++){
                        sub[j] = test_rows[j] - train[i][j];
                  math[j]=(float) Math.pow((sub[j]), 2);
                  res = math[0]+math[1]+math[2]+math[3]+math[4]+math[5];
                  }
                  distance[i] = (float) (Math.sqrt(res));
                  revd[i] = 1/distance[i];
                  rev[i].lable = train[i][6];
                  rev[i].dist =  distance[i];
                  rev[i].rev_dist =revd[i];
            }
            return rev;
                                                                  }
```

در تابع sort مقادیر فاصله ها به ترتیب نزولی مرتب می شوند(به وسیله الگوریتم binary search)

وآرایه list ایجاد میشود که به ازای kاز ۱ تا ۱۵ مقادیر k تا بیشترین مقادیر را نگهداری می کند. در

تابع find_max بررسی می شود که بیشترین فراوانی list مربوط به کدام کلاس است و آن کلاس به عنوان کلاس پیش بینی شده برای آن سطر از ماتریس test مشخص می شود.

```
private static reverseDist_value[] sort(reverseDist_value[] rev, int n) {
              int i, j=0;
              float y=0;
              for(i = 0; i < n; i++){
              for(j = 1; j < (n-i); j++){
              if(rev[j-1].rev_dist < rev[j].rev_dist){
              y = rev[j-1].rev_dist;
              rev[j-1].rev_dist=rev[j].rev_dist;
              rev[j].rev_dist=y;
              }
              }

                                                                  }

private static int find_max(reverseDist_value[] list) {
            // TODO Auto-generated method stub
            int max = 0, max1=0,max2=0;
            int acc_num = 0 , unacc_num=0, good_num=0, vgood_num=0;
            for(int i=1;i<list.length;i++)
                  {if(list[i].lable==1)
                        acc_num++;
                  else if(list[i].lable == 2)
                        unacc_num++;
                  else if(list[i].lable == 3)
                        good_num++;
                  else if(list[i].lable == 4)
                        vgood_num++;
                  }
            if(acc_num > unacc_num) max1=1;else max1=2;
            if(good_num > vgood_num) max2=3;else max2=4;
            if(max2>max1)
                  return max2;
            else
            return vgood_num;
                                                                  }
```

در تابع evaluateAccو EvaluateRec و evaluateAcc با مقایسه ی lable مربوط به سطر test و مقدار پیش بینی شده، تعداد حدس های درست و حدس های نادرست را به دست آورده می شود و با تقسیم تعداد حدس های صحیح بر تعداد کل پیش بینی ها مقدار دقت به دست می آید. نتایج به دست آمده در فایل ذخیره می گردد.

```
private float evaluatePrec(float tp, float total) {
      precision = (tp/total);
            return precision;
                  }

      private float evaluateRecall(float tp, float total) {
```

```
            recall = (tp/total);
            return recall;
                  }

      private static float evaluateAcc(float tp2, float fp2, float total) {
            accuracy = (tp2/total)+(fp2/total);
            return accuracy;
                  }
```

میزان precision در بهترین وضعیت (k=15) برابر با 65.75% و f-measure آن برابر 79.34% به دست می آید.

در نهایت نیز خروجی هر یک از ۴۵ حالت در فایل های خروجی ذخیره می گردد. (در فولدر output).

```
             for(int k=1;k<=15;k++){
                   String k2 = Float.toString(k);
                   String file_name = s.concat(k2).concat(txt0);
                   BufferedWriter bw = new BufferedWriter(new FileWriter(new
File(file_name), true));

             for(test_row_num=0;test_row_num<140;test_row_num++) {//TODO
                  strLine2 = br2.readLine();
                  if(strLine2!=null){testProcess(k,strLine2,train);
                  float lable2 = find_max(list2);

                  String test_class = null;
                  if(lable2==1)test_class=",acc";
                  if(lable2==2)test_class=",unacc";
                  if(lable2==3)test_class=",good";
                  if(lable2==4)test_class=",vgood";

                  String file_line = strLine2.concat(test_class);
                   bw.write(file_line);
                   bw.newLine();
                  }
                  }
                   bw.close();
                  }
```