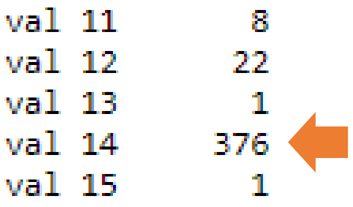
All steps and implementations of this section in the Q3 KNN.ipynb file

I.

According to the specifications, this dataset has 452 data rows and 280 columns. These data are classified into 16 groups, of course, the frequency of 3 groups is zero, so we are actually dealing with 13 diagnostic groups, which of course are not balanced.

According to the dataset, missing values are entered with '?'. To identify these, we first replace their value with NAN so that more tools are available to manipulate them. It is clear that the only columns with missing values are numbers 11 to 15 and the number of missing values is as follows:



```
val 11      8
val 12     22
val 13      1
val 14    376
val 15      1
```

Because the data is not balanced, before deleting the rows with missing values, I want to make sure that the data I'm missing is not manipulating the number of classes. (For example, if most of them are from class 1, which has the highest frequency, it is not a big problem, but if, for example, they are all from the same class and their frequency is significantly reduced, it is not acceptable to remove them.) It can be checked, but because the removal Data is not a good method in general, I choose another mode, for example, insert with average and fill these 5 columns with the average column wherever there is an empty value.

II.

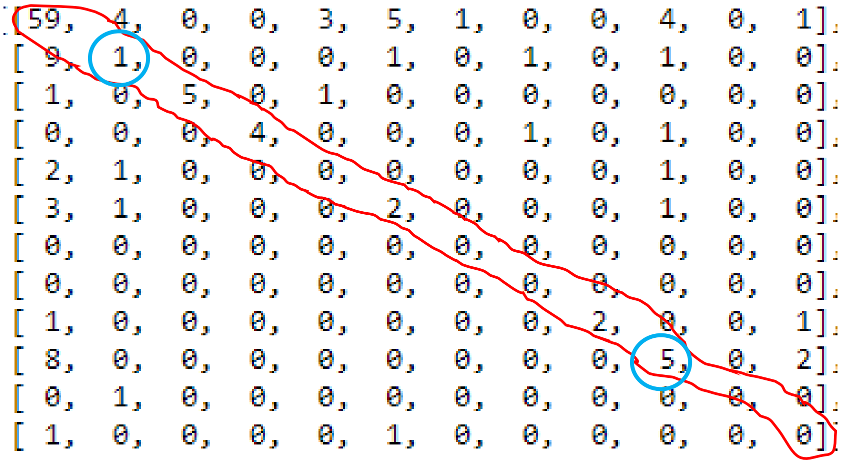
Because KNN is dependent and sensitive to the value of the variables, in order to equalize the influence of the predictive factors, I have to transform the data. Among the transformations, I use the minimum and maximum transformation. According to what was said in the book and considering that inside these 279 columns we have both qualitative variables (converted into numbers) and continuous measured values, the best transformation that was introduced in the class and can be applied to all columns Implemented using MinMaxScaler().

$$\frac{x - x_{min}}{x_{max} - x_{min}}$$

First, we will separate two datasets, one including features and one containing single column of the target variable from the main dataset. Then, with a ratio of 30 to 70, I divide these data into two sets of testing and training.

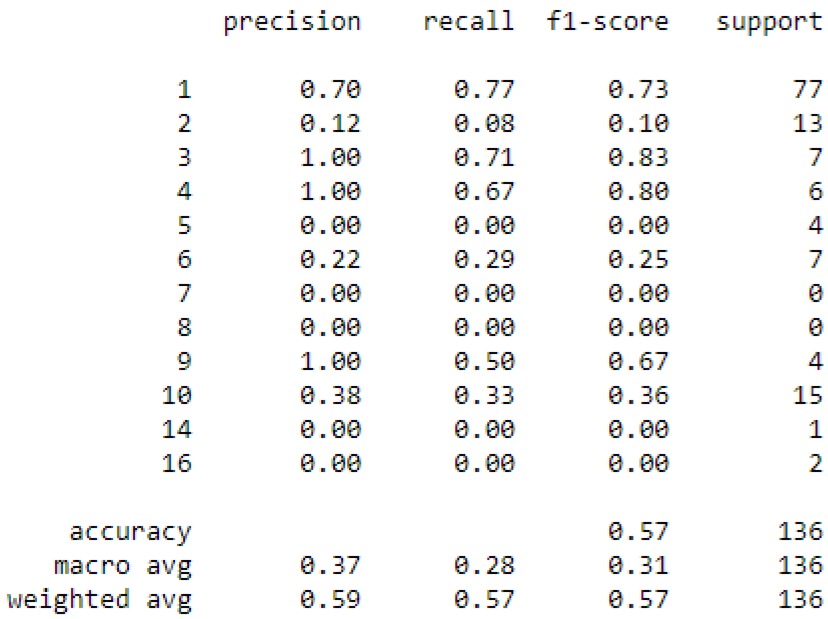
Now, according to the request of the question, I will create a KNN class model that takes k equal to 1 and give it the training data. After training the model, I use the test data for evaluation. (I have 126 test data)

The results obtained from this model for test data:



```
[59,  4,  0,  0,  3,  5,  1,  0,  0,  4,  0,  1],
[ 9,  1,  0,  0,  0,  1,  0,  1,  0,  1,  0,  0],
[ 1,  0,  5,  0,  1,  0,  0,  0,  0,  0,  0,  0],
[ 0,  0,  0,  4,  0,  0,  0,  1,  0,  1,  0,  0],
[ 2,  1,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0],
[ 3,  1,  0,  0,  0,  2,  0,  0,  0,  1,  0,  0],
[ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
[ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
[ 1,  0,  0,  0,  0,  0,  0,  0,  2,  0,  0,  1],
[ 8,  0,  0,  0,  0,  0,  0,  0,  0,  5,  0,  2],
[ 0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
[ 1,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0]
```

When we check the confusion matrix, I will first check the elements on the diagonal that indicate correctness. The element on directory ii shows how many of the test data are in class i and are correctly recognized. Intuitively, if we comment on this matrix, it does not seem that we have a good model. For example, the performance of this model was bad for the class such as row 2. Among the 10 cases that belonged to this class, 9 were wrongly attributed to the first class. 4 of the data that were in class 1 were mistakenly transferred to this class and only 1 is correct. Or, for example, in a class such as line 10, he made only 5 diagnoses correctly and mistakenly assigned the rest to class 1, and in addition, he assigned a total of 8 items from other classes to this class by mistake. If I look more generally, most of the mistakes are recorded in the first row and column, especially in the first column. That is, a large number of mistakes were due to the fact that something that is in another class was taken to class one. This is due to the lack of balance of the data and the significant frequency of class one, but on the other hand, the number 1 for k is not a good choice considering this characteristic of the data. For a better review of the obtained criteria, let's see:

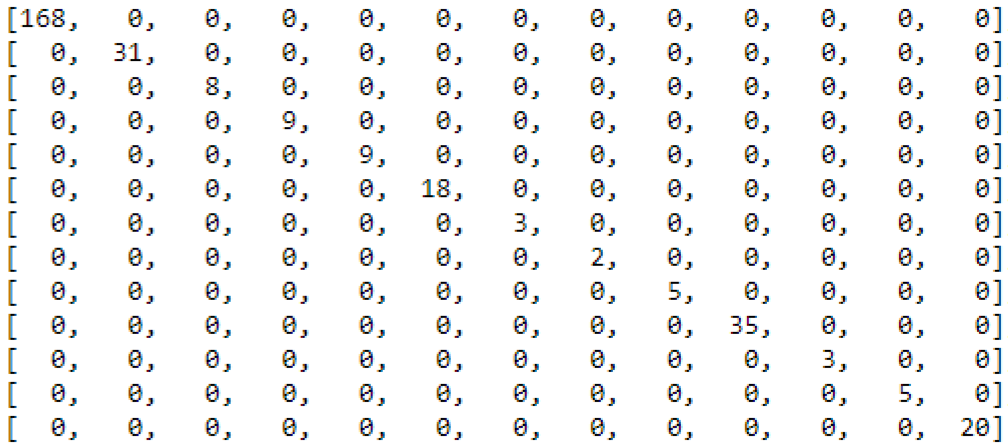


```
              precision    recall  f1-score   support

           1       0.70      0.77      0.73        77
           2       0.12      0.08      0.10        13
           3       1.00      0.71      0.83         7
           4       1.00      0.67      0.80         6
           5       0.00      0.00      0.00         4
           6       0.22      0.29      0.25         7
           7       0.00      0.00      0.00         0
           8       0.00      0.00      0.00         0
           9       1.00      0.50      0.67         4
          10       0.38      0.33      0.36        15
          14       0.00      0.00      0.00         1
          16       0.00      0.00      0.00         2

    accuracy                           0.57       136
   macro avg       0.37      0.28      0.31       136
weighted avg       0.59      0.57      0.57       136
```

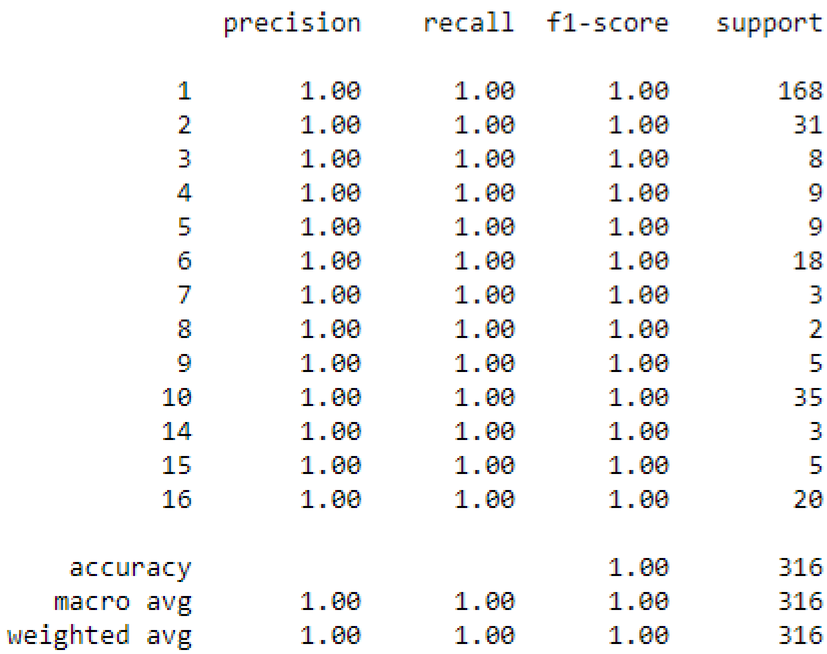
Well, we still don't have a good model! Let me start with accuracy. It is not a very good measure, but it shows that it did only 45% of the total data it was supposed to classify correctly. If the data here was balanced, probably the same number would not have been obtained and the value would have been much lower, and the magnitude of this number is due to the large number of first class data. The next measure of precision is strangely set to 1, i.e. the maximum value for some of the classes, while it is zero in some other places. This imbalance is first related to the way this criterion is calculated, and secondly, the number of 16 classes (13 existing ones) for 126 data is too much, and for example, if we have one frequency of one class and it is wrongly recognized, the precision equals becomes zero On the other hand, the large number of this number in other lines is due to the fact that the evaluation of false diagnoses in this class is considered. For example, if we take all 126 items into one class, this criterion shows a completely unfavorable result. While the accuracy may not change much (in the current data). If it is important that an item is not mistakenly assigned to class 3 when it does not belong to this class, this model has a good result. But if we have the same importance for class 6, the model is completely unfavorable. On the other hand, there is the recall criterion, which measures the number of correct detections among all real values. If it is important to recognize all class 1 cases correctly, this criterion is suitable (for example, if the equivalent of class 1 is the patient's emergency situation), if we have the same importance for classes 16 and 6 or... which have a small amount, the model is good. is not. (Because I don't know exactly how good or bad a class is, so I only interpret the criteria in general and without medical considerations) Now, if I want to merge the results of the previous two columns with the f1-score criteria, the result is that again, the model for different classes It has many ups and downs. And on average, its performance is poor.

Now let's see the results of the training data:

As expected from k=1, a model with all criteria at the perfect level for the training data. If we look at the matrix, there is not even one case of wrong classification, and of course, for this reason, all the numbers for all criteria are equal to 1.



```
[168,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  0,  31,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  0,   0,   8,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  0,   0,   0,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  0,   0,   0,   0,   9,   0,   0,   0,   0,   0,   0,   0,   0]
[  0,   0,   0,   0,   0,  18,   0,   0,   0,   0,   0,   0,   0]
[  0,   0,   0,   0,   0,   0,   3,   0,   0,   0,   0,   0,   0]
[  0,   0,   0,   0,   0,   0,   0,   2,   0,   0,   0,   0,   0]
[  0,   0,   0,   0,   0,   0,   0,   0,   5,   0,   0,   0,   0]
[  0,   0,   0,   0,   0,   0,   0,   0,   0,  35,   0,   0,   0]
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,   0,   0]
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   5,   0]
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  20]
```

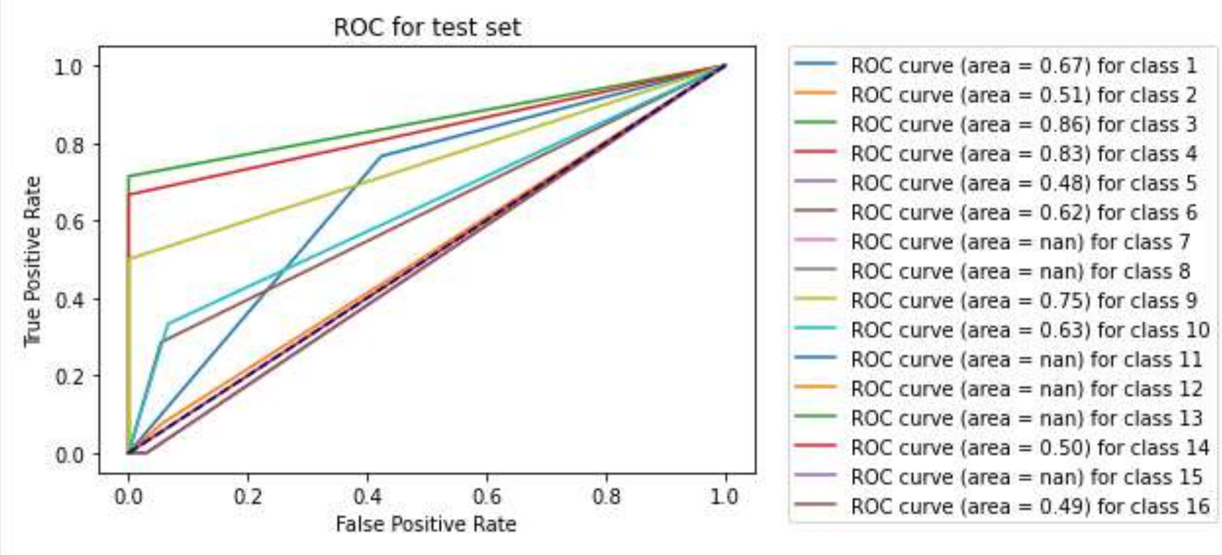


```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00       168
           2       1.00      1.00      1.00        31
           3       1.00      1.00      1.00         8
           4       1.00      1.00      1.00         9
           5       1.00      1.00      1.00         9
           6       1.00      1.00      1.00        18
           7       1.00      1.00      1.00         3
           8       1.00      1.00      1.00         2
           9       1.00      1.00      1.00         5
          10       1.00      1.00      1.00        35
          14       1.00      1.00      1.00         3
          15       1.00      1.00      1.00         5
          16       1.00      1.00      1.00        20

    accuracy                           1.00       316
   macro avg       1.00      1.00      1.00       316
weighted avg       1.00      1.00      1.00       316
```

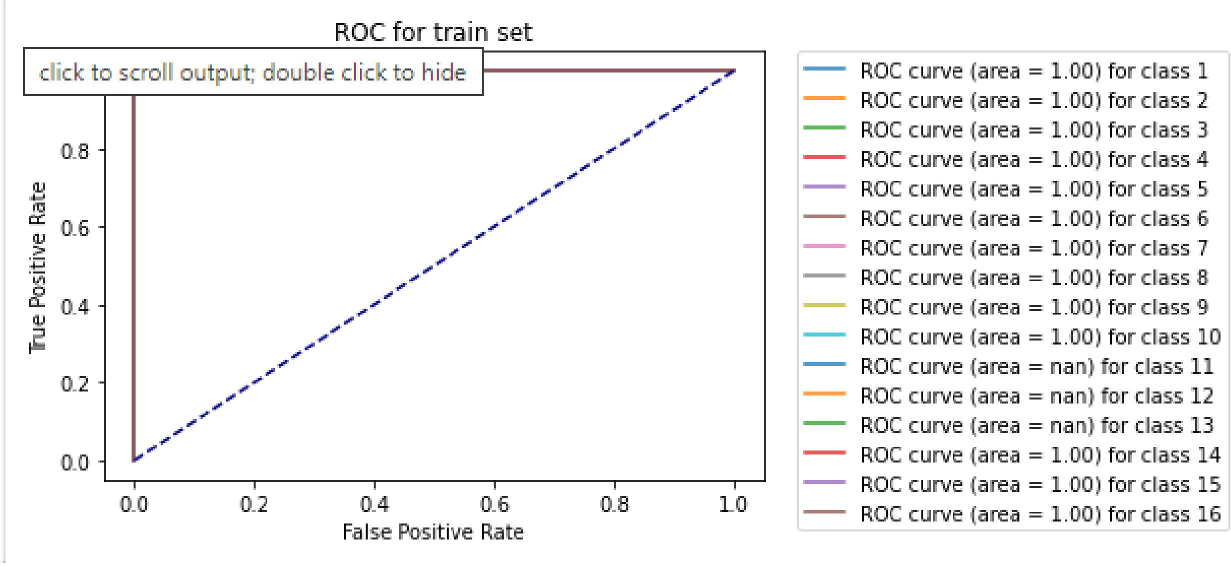
If we compare this with the results of the test data, it is clear that the model is overfit. The more k we increase, the complexity of the model decreases, so by increasing k, we can solve the overfitting problem so that by accepting an error value on the training part, it can have a good result on the test data. (bias variance trade-off)

Finally, check the ROC chart, which I drew on one chart for easy comparison of all 16 classes.

The larger the area under the curve of a class is (closer to one), it means we have a better model to determine whether a data belongs to that class or not. Here, for example, there is a better model for the test data for class 3.



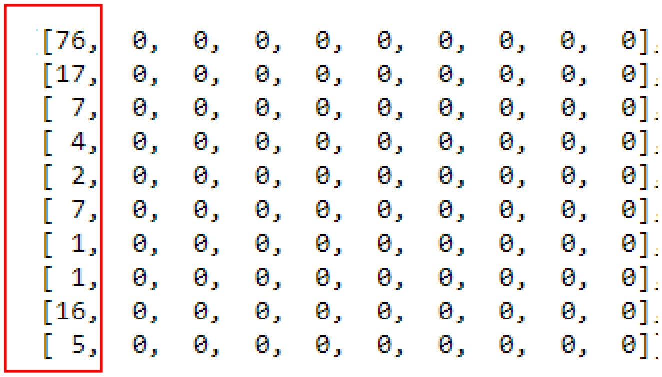
In the case of the test chart, all the curves are superimposed and have the same shape. (except for the classes whose abundance in the dataset is zero) and all of them have a value equal to one (area under the graph), which clearly shows overfitting in comparison with the test data.



Now the same reports obtained for the KNN model with 30 neighbors:

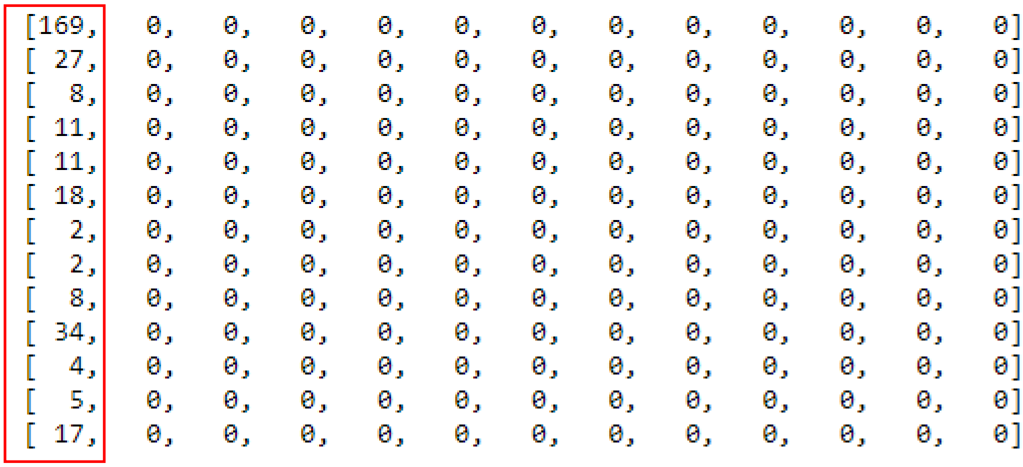
Test data matrix:

It is clear from the matrices that the model works in such a way that it takes all the diagnoses to class 1! And due to the lack of balance of the test data (it shouldn't be balanced!) toward class 1, the accuracy is not worse and it is no different from the previous state!!



```
[76,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[17,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[ 7,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[ 4,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[ 2,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[ 7,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[ 1,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[ 1,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[16,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[ 5,  0,  0,  0,  0,  0,  0,  0,  0,  0]
```

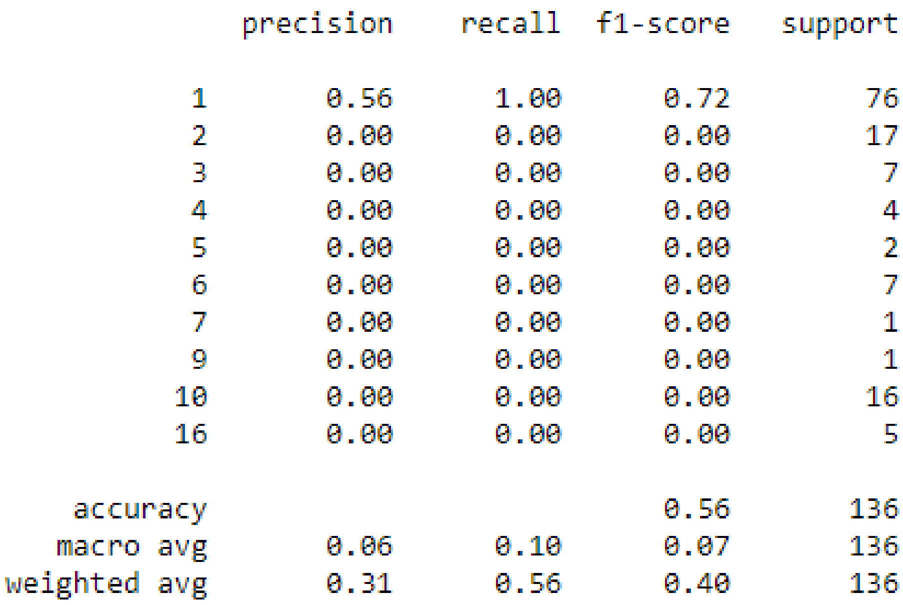
Training data confusion matrix:



```
[169,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[ 27,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  8,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[ 11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[ 11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[ 18,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  8,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[ 34,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  4,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  5,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[ 17,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
```

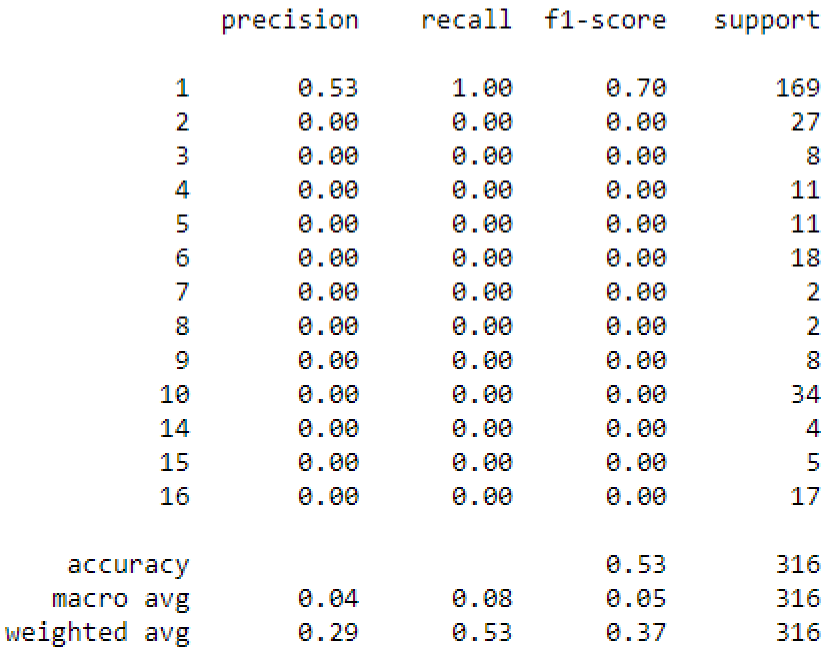
Since the model has taken only one classification class and referred everyone to it, we only have the results for class 1, because only the detection number of this class has a non-zero value for the standard fraction form (Russian numbers of Qatar). In addition to the fact that the data of this class are abundant in the test data, the model seems to have obtained good results for the first class. But in fact, the model is so bad that it doesn't do classification work at all! It is true that the error difference is less compared to the previous state, but here the complexity of the model is so low that even on the training data it has a lot of error and the model is clearly underfit. We need a much lower value of k.

Test data:



```
              precision    recall  f1-score   support

           1       0.56      1.00      0.72        76
           2       0.00      0.00      0.00        17
           3       0.00      0.00      0.00         7
           4       0.00      0.00      0.00         4
           5       0.00      0.00      0.00         2
           6       0.00      0.00      0.00         7
           7       0.00      0.00      0.00         1
           9       0.00      0.00      0.00         1
          10       0.00      0.00      0.00        16
          16       0.00      0.00      0.00         5

    accuracy                           0.56       136
   macro avg       0.06      0.10      0.07       136
weighted avg       0.31      0.56      0.40       136
```

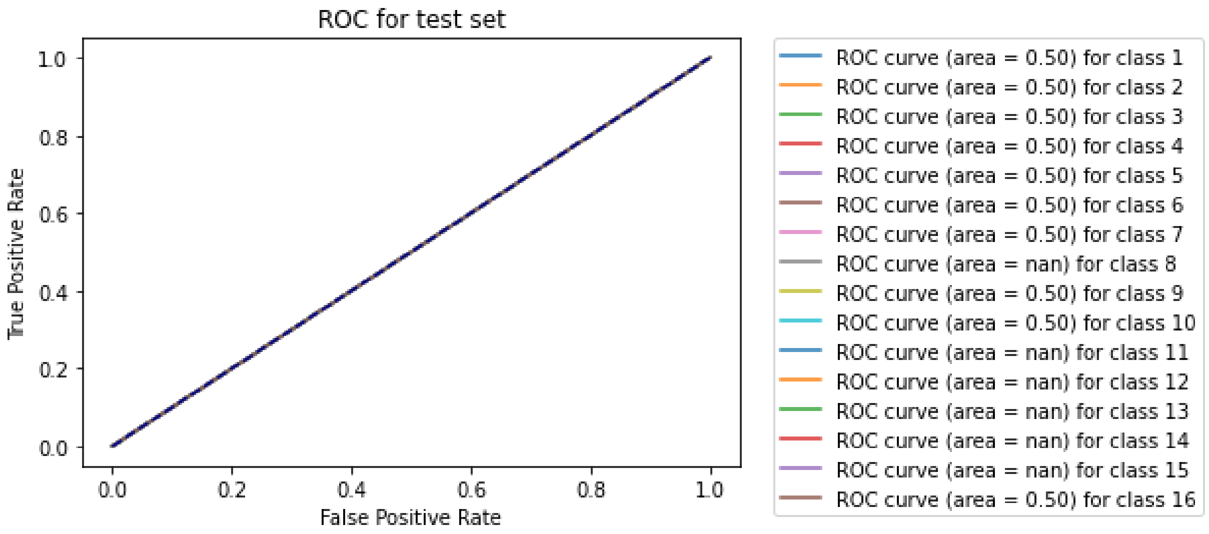
Training data:

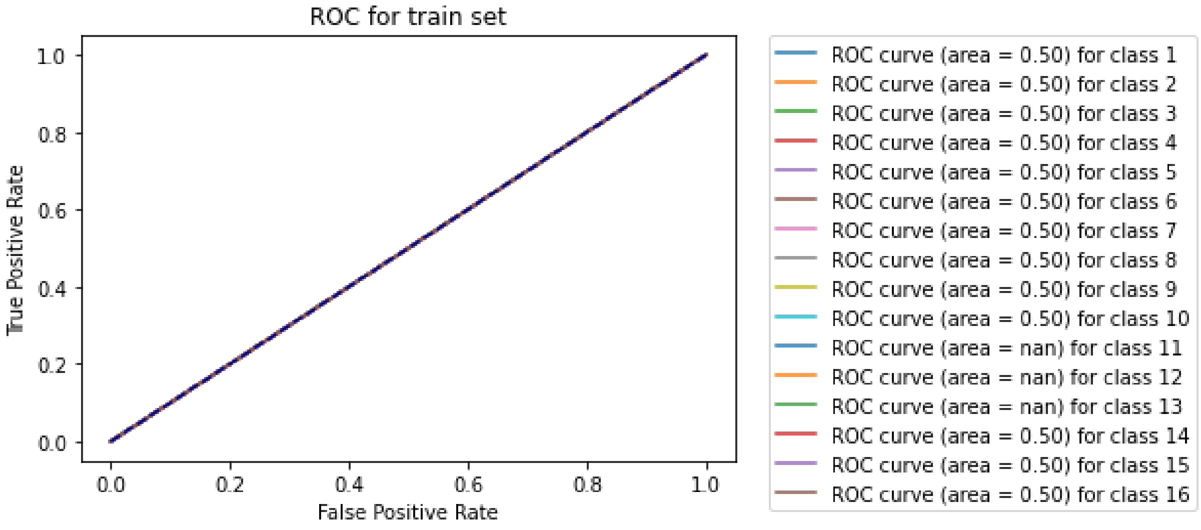


```
              precision    recall  f1-score   support

           1       0.53      1.00      0.70       169
           2       0.00      0.00      0.00        27
           3       0.00      0.00      0.00         8
           4       0.00      0.00      0.00        11
           5       0.00      0.00      0.00        11
           6       0.00      0.00      0.00        18
           7       0.00      0.00      0.00         2
           8       0.00      0.00      0.00         2
           9       0.00      0.00      0.00         8
          10       0.00      0.00      0.00        34
          14       0.00      0.00      0.00         4
          15       0.00      0.00      0.00         5
          16       0.00      0.00      0.00        17

    accuracy                           0.53       316
   macro avg       0.04      0.08      0.05       316
weighted avg       0.29      0.53      0.37       316
```

The last task is to compare the ROC charts and their area, which is written next to the chart for each class:

In both cases, the test data and the training performance are equally bad! Some classes, where there is not enough data, are still zero and have no value, and the rest are equal to 0.50, which is like randomly determining whether this data is a member of class one or not. (all colors and all charts are on one line)





III.

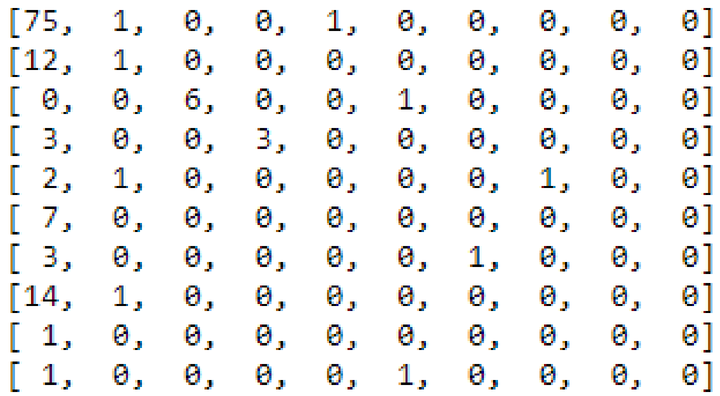
First, let's determine the parameters, then I will use GridSearchCV to find the appropriate values with the mutual evaluation algorithm. In the previous episode, we had the results of 1 and 30, and one was too much and one was too little. So we have to find the right number for k within this range. The distance criterion is the same as mentioned in the question. If a certain evaluation criterion is a priority for us, we should also determine it, but here I did not use it and left it blank.

After searching, the best mode for KNN on these data:

```
{'metric': 'cosine', 'n_neighbors': 5}
```

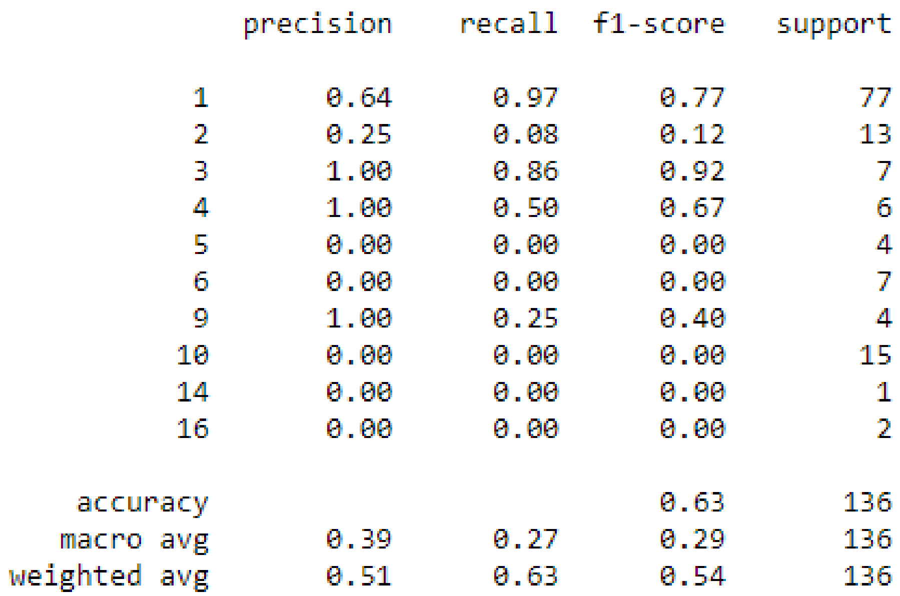
That is, cosine distance and k value equal to 5, and based on the test and training data with the new model, we have:

test :



```
[75,  1,  0,  0,  1,  0,  0,  0,  0,  0]
[12,  1,  0,  0,  0,  0,  0,  0,  0,  0]
[ 0,  0,  6,  0,  0,  1,  0,  0,  0,  0]
[ 3,  0,  0,  3,  0,  0,  0,  0,  0,  0]
[ 2,  1,  0,  0,  0,  0,  0,  1,  0,  0]
[ 7,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[ 3,  0,  0,  0,  0,  0,  1,  0,  0,  0]
[14,  1,  0,  0,  0,  0,  0,  0,  0,  0]
[ 1,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[ 1,  0,  0,  0,  0,  1,  0,  0,  0,  0]
```

The result obtained from the test data is very close to the test result for k=30, and as it is clear from the difference in the numbers, it is actually improved. It means that the complexity of the model is increased to create a better balance between testing and training.

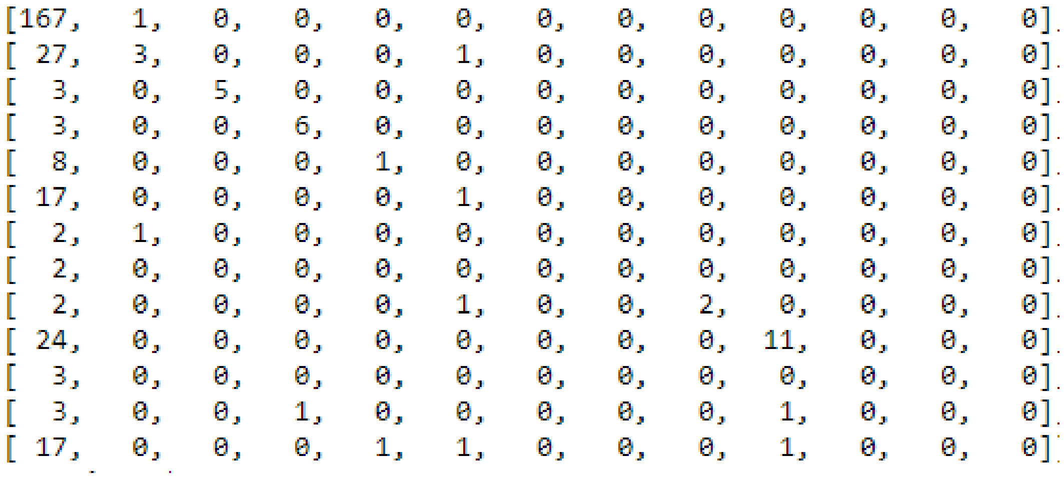


```
              precision    recall  f1-score   support

           1       0.64      0.97      0.77        77
           2       0.25      0.08      0.12        13
           3       1.00      0.86      0.92         7
           4       1.00      0.50      0.67         6
           5       0.00      0.00      0.00         4
           6       0.00      0.00      0.00         7
           9       1.00      0.25      0.40         4
          10       0.00      0.00      0.00        15
          14       0.00      0.00      0.00         1
          16       0.00      0.00      0.00         2

    accuracy                           0.63       136
   macro avg       0.39      0.27      0.29       136
weighted avg       0.51      0.63      0.54       136
```

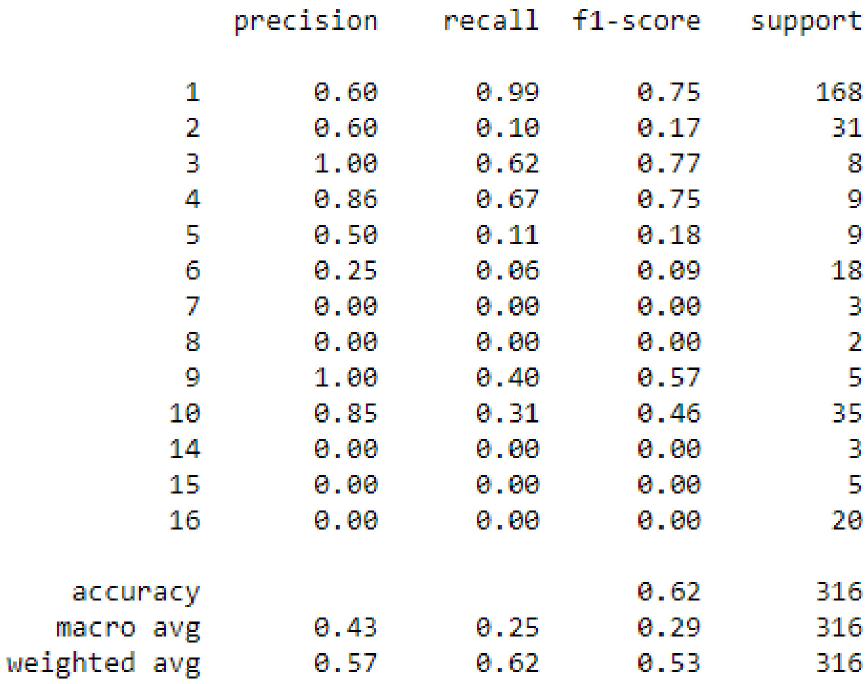
The fluctuation of recall, precision, f1 criteria is high on different groups and the reason is the same issues raised on the previous part. Depending on, for example, the 9th group and the correct diagnosis of this class, what is the importance and cost for us, we can choose the appropriate criteria for evaluating the model. For example, the model for class 9 has performed well in precision, which means that the probability that the model will assign an item from other classes to class 49 is low, but instead, it has a weak recall criterion, which means that the number of elements from this class that fall into other classes by mistake is high. . If the condition of someone who is in class 9 in reality requires immediate measures and care, this model cannot identify these patients (for example, it may classify their condition in the healthy or non-emergency class), a usable model. It is not and it is very risky if it is used in this direction.

But if it is the other way around, for example, the condition of patient 9 is such that the necessary care for him is harmful for other categories, or if it has a heavy cost, and I don't want to send the wrong patients to this department, this precision number is good! But in general, this big difference in evaluation criteria is not desirable. The higher all the numbers are and the closer they are to each other, it means that the model gives a more uniform and favorable result for the members of all the classes. If we consider class 3, it performs well in all the above criteria.

In training data:



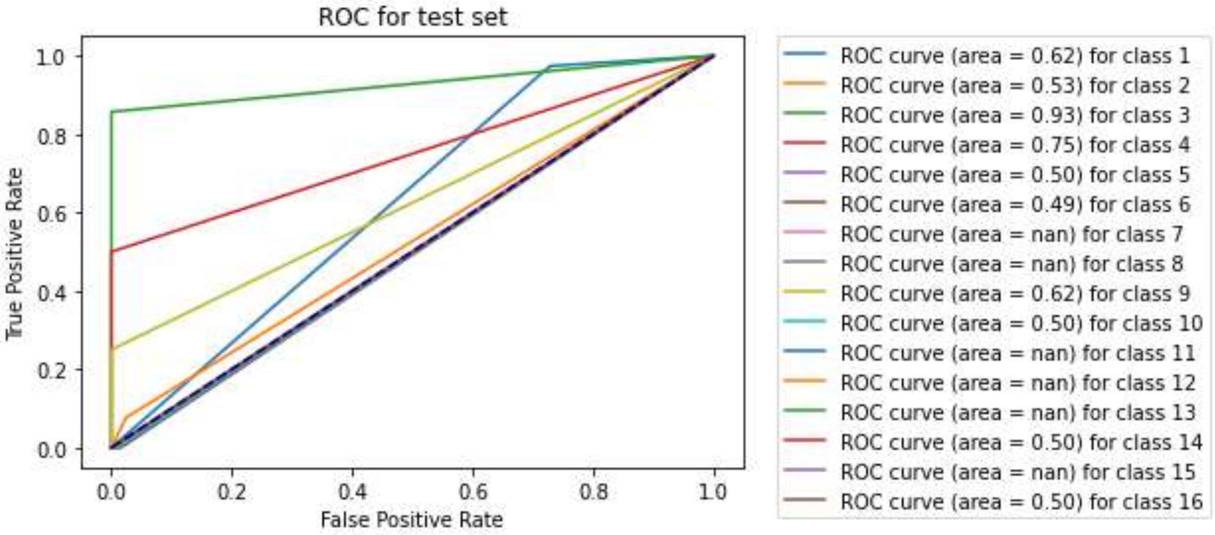
```
[167,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[ 27,   3,   0,   0,   0,   1,   0,   0,   0,   0,   0,   0,   0]
[  3,   0,   5,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  3,   0,   0,   6,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  8,   0,   0,   0,   1,   0,   0,   0,   0,   0,   0,   0,   0]
[ 17,   0,   0,   0,   0,   1,   0,   0,   0,   0,   0,   0,   0]
[  2,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  2,   0,   0,   0,   0,   1,   0,   0,   2,   0,   0,   0,   0]
[ 24,   0,   0,   0,   0,   0,   0,   0,   0,  11,   0,   0,   0]
[  3,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]
[  3,   0,   0,   1,   0,   0,   0,   0,   0,   1,   0,   0,   0]
[ 17,   0,   0,   0,   1,   1,   0,   0,   0,   1,   0,   0,   0]
```

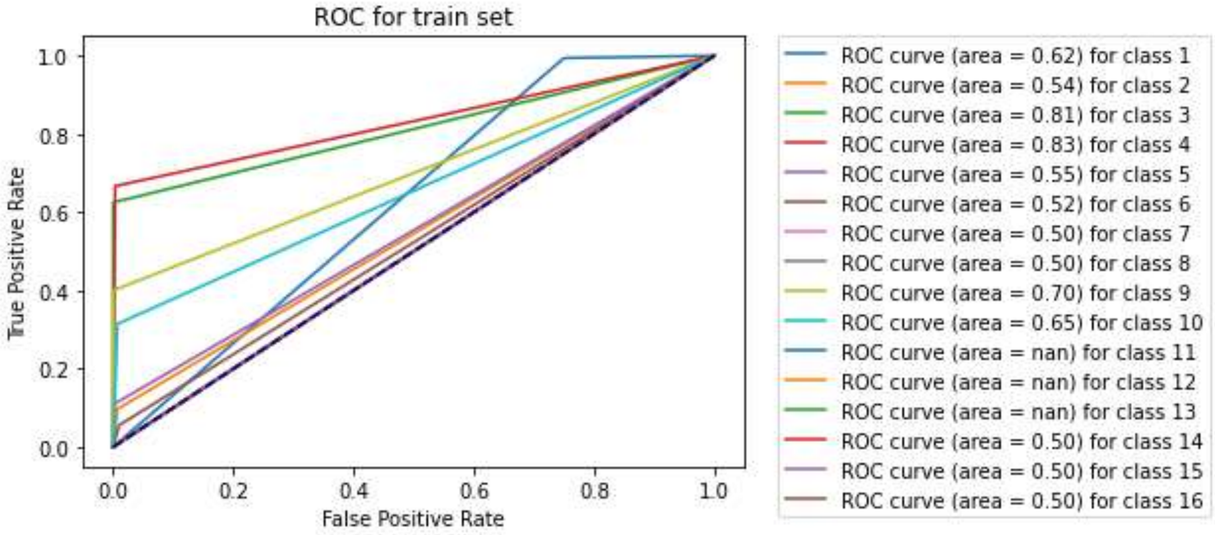


```
              precision    recall  f1-score   support

           1       0.60      0.99      0.75       168
           2       0.60      0.10      0.17        31
           3       1.00      0.62      0.77         8
           4       0.86      0.67      0.75         9
           5       0.50      0.11      0.18         9
           6       0.25      0.06      0.09        18
           7       0.00      0.00      0.00         3
           8       0.00      0.00      0.00         2
           9       1.00      0.40      0.57         5
          10       0.85      0.31      0.46        35
          14       0.00      0.00      0.00         3
          15       0.00      0.00      0.00         5
          16       0.00      0.00      0.00        20

    accuracy                           0.62       316
   macro avg       0.43      0.25      0.29       316
weighted avg       0.57      0.62      0.53       316
```

Finally, comparing the ROC value of the area of the graphs for this optimal model that was found:

For about half of the classes we have performance above 0.5, which is good. The homogeneity of test and training results is quite noticeable here. We have the best performance for class 3 detection. The rest of the classes that remained at 0.5 show that if we use the model to recognize these classes, the recognition will not be very valid.





The difference between test and training data is much better than k=1. The difference between the results obtained for the training and test sets is much less, and in my opinion, the accuracy is at an acceptable level and we don't have the problem of underfit. But in my opinion, the heterogeneity of the model's performance for different classes makes the validity of the model not very high, and I don't think it is a good model. (Especially since the subject of its use is medicine) One of the main problems of KNN is that it completely depends on the data, and for example, maybe if the data was a more accurate and suitable set in terms of the distribution of classes, a better model could be made. Of course, by performing pre-processing and methods of data balancing or data addition, etc., it is possible to build a model with better performance on the same data.

P.N.: For example, during 3 different divisions of test and training data with 3 different training sets using GridSearchCV, I reached 3 different results for the desired model, where k was equal to 3, 5, and 4, and they had different metrics. And the results and numbers obtained were naturally different. I did not record the results, but this difference shows the high dependence of the KNN model on the data.