Iris database - Effectiveness of selected classifiers

Paulina Hałatek[1], Katarzyna Wiltos[1] and Mariusz Wróbel[1]

[1] *Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, Poland*

Abstract

Machine learning and artificial intelligence are crucial tools in the vast majority of different fields, but mainly in computer science and technology. Classifiers play a vital role in this field, especially in predicting the class membership of a sample under consideration. An example of the practical use of classifiers is the spam filter of email messages. The following paper aims to determine the most efficient classifier from selected: kNN, Soft set, and Naive Bayes on Iris database. Different versions of each of the classifiers have been considered. For kNN, the performance of various metrics was compared, for the Soft set, two approaches for establishing intervals during the classification, and for Naive Bayes, the normal and triangular distributions were compared. The most effective versions of the classifiers have been selected for the final comparison.

Keywords

Artificial intelligence, Iris, classifiers, kNN, Soft set, Naive Bayes

1. Introduction

Artificial intelligence is an important aspect in today's world. It brings a communication between human and machine. Thanks to that we are able to teach our computer how to process a given data and get a response from it. This is a called machine learning.

Machine learning is a field of study that uses diverse algorithms to make some analysis in the given data. That can be for example:

1. variety recognition,
2. weather and disease prediction,
3. puzzle or sudoku solver,
4. building a movie recommendation system.

This is actually a small fraction of the immeasurable possibilities in this field of study. Machine learning models are used to learn the patterns in data. Machine learning algorithms can be used for example to gather information about data, split data for two parts and try to identify unknown sample as a data element. The methods which determine this, are called classifiers. We have various types of classifiers applicable to different task. In machine learning models neural networks based ideas are very efficient in complex data analysis. In [4] was presented how to use them in low-dimensional data feature learning. The idea presented in [8] proposed neural network for analytical purposes of data recorded form high-speed train. There are also very efficient, however simple in construction, classifiers based on approaches sourced in data analytics. In [1] was proposed a model of soft set to approximate reasoning from input data. A model based on kNN classifier for big data analytics was presented in [7]. In decision processes we also very often use bayesian approaches which analyze probability of possible situations. In [2] was presented an wildfire risk assessment from data of remote sensing. Transmission of sensor readings for classifiers is an important topic, and there are many interesting models to support this process [5]. Classification model also depends on the type of the input information. In [6] was discussed how to use bayesian model for text classification. This kind of processes also need efficient data aggregation to improve efficiency of the classifier [3].

In our paper we are going to compare three classifiers: kNN, Soft set, Naive Bayes and decide which of those is the most accurate and effective. We will be trying to use different methods for each classifiers to determine the most reliable results. Each of these classifiers is different in some aspects. But the thing which connects them is that all of theirs main purpose is to identify a given sample by learning from a database using different types of identification. And in this paper we want to bring each classifier closer in the meaning, we are going to explain each of those three classifiers, explain which techniques of identification we were used and make overall conclusion which one is the best classifier.

We decided to use Iris data base to compare the accuracy of those classifiers and identify the most suitable.

2. Iris database

Before we started working on our base, we made sure that our Iris base didn't have null or NaN values. We have created additional class called DataProcessing which

IVUS 2022: 27th International Conference on Information Technology, May 12, 2022, Kaunas, Lithuania

paulhal456@student.polsl.pl (P. Hałatek); katawil756@student.polsl.pl (K. Wiltos); mariwro279@student.polsl.pl (M. Wróbel)

helped us to shuffle, normalize and split database by dividing it into 70% as a training set and 30% as a validation set. As we can see, the setos class is significantly separated from the rest of the classes. This will result in a high proportion of correctly recognized objects for this class.

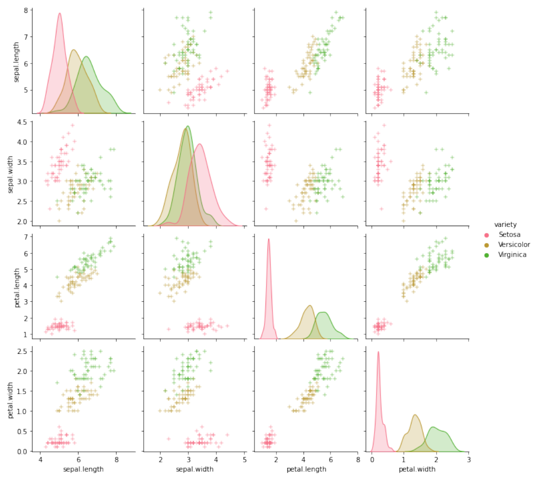
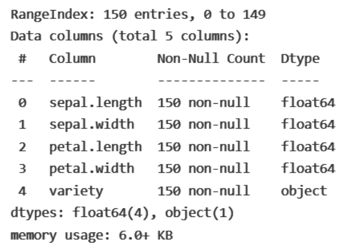


Figure 1: Iris dataset graphs



```
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal.length  150 non-null    float64
 1   sepal.width   150 non-null    float64
 2   petal.length  150 non-null    float64
 3   petal.width   150 non-null    float64
 4   variety       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

Figure 2: Iris dataset information

2.1. Normalization

For all classifiers we were normalizing a given database by taking all values from a specific column, determining of the lowest and highest value in the specific column and changing all values in the column according to the formula which is given below:

$$newValue[x][y] = \frac{oldValue[x][y] - min}{max - min} \tag{1}$$

x	a current row
y	a current column
min	a minimal value in the current column
max	a maximal value in the current column

3. Methods

3.1. kNN

3.1.1. Formulas

In order to function properly, the kNN algorithm needs functions that calculate the distance of the object for which we are looking for a class to the objects of classes already known to us. It is on the basis of this distance that the kNN algorithm decides to which class a given object may belong. There are many ways to calculate distances, each with its pros and cons. When calculating distances, we can, for example, use one of the known metrics, e.g. Euclid:

$$||x - x^i||^2 = \sum_{j}^{n}(x_j - x_j^i)^2 \tag{2}$$

Or Minkowski distance:

$$L_m(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^m)^{\frac{1}{m}} \tag{3}$$

In our algorithm, we chose the Minkowski metric.

3.1.2. Algorithm

The kNN (k-Nearest Neighbors) classifier is one of the most important non-parametric classification methods. The kNN algorithm does not create an internal representation of the training data, but looks for a solution only when the testing pattern appears. It consists in assigning an object to a given class by checking to which representatives a given object has the shortest distance. The algorithm works as follows. First, a sample is taken from the validation set. Next for a given sample, the distance to each object in the test set is calculated. Then list is created containing the given test file object and the distance to the sample which then is sorted from shortest distance to longest.

After that from this list, the k objects in the shortest distance from the sample are analyzed. At the end the sample is assigned to the class with the most objects.

Algorithm 1 kNN algorithm

Input: Test set, validate set, k, m
Output: The class to which the sample may belong

```
while $i < len(validate\ set)$ do
    while $j < len(test\ set)$ do
        Calculate the distance using Minkowski
        distance of test object j to the sample $i$
        and add the result to the list of distances.
        $j++$
    Sort the list of distances in ascending order.
    Take the k objects with the smallest distance
    and return the class x with the most objects.

*Return from dictionary variety with the highest probability*
```

3.2. Soft sets

The soft set term as a mathematical model offers a tool for analysing vaguely defined objects. Soft set theory is a generalisation of fuzzy set theory that was introduced in 1999 by Dmitri Molodtsov.

3.2.1. Formulas

There are a few ways in which soft set may be implemented, for example: including weight or not. In this case weight was included. Pearson correlation coefficients were calculated to properly choose the most appropriate weight values for particular characteristics.

$$r = \frac{(\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}} \tag{4}$$

x_i	characteristic value for i = 0,1,...,n
x	mean value for particular characteristic
y_i	value of compared characteristic for i = 0,1,...,n
y	mean value for compared characteristic
r	Pearson correlation coefficient value

3.2.2. Algorithm

Prior to classification data was prepared through shuffling, normalizing, and splitting the database into a test set and validation set in the ratio of 70 to 30.

In the developed implementation of the soft set, the minimum and maximum values for each species of the test set were calculated. On their basis, the middle values of species range values were determined.

The classifier considers samples from the validation set together with the selected characteristic weight.

In the first approach implementation for each sample, the distance from the center of the interval is calculated and the minimum value is chosen, which determines sample classification.

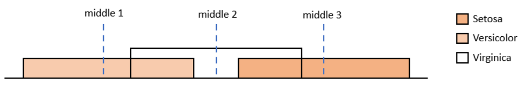


Figure 3: First approach algorithm visualization

Algorithm 2 Soft set algorithm - first approach

Input: Test set, validation set, weight
Output: The class to which the sample was classified

```
$centers \leftarrow centre\ value\ of\ each\ iris\ type$

for $index < len(validation\ set)$ do

    Creates nested list with iris type name, minimal
    and maximal values for each iris type in test set.

    Creates nested list with iris type name, centre
    value for each iris type based on minimal
    and maximal values.

    $row \leftarrow list\ of\ traits\ values\ of\ one\ sample$
    $sampleType \leftarrow iristypename$
    $sampleValue \leftarrow 0$
    $i \leftarrow 0$
    for $t\ in\ row$ do    ▷ Add all trait values for sample
        $sampleValue += t * weight[i]$
        $i += 1$
    for $value\ in\ centers$ do    ▷ Calculate distances
        $centre = value[0]$
        $distance = |centre - sampleValue|$

*Chooses minimal distance and corresponding iris type.*
*Returns classified type.*
```

In the second approach algorithm, overlapping intervals are considered and mean value is calculated to create new intervals. Based on new intervals each sample is being classified accordingly to these measures.

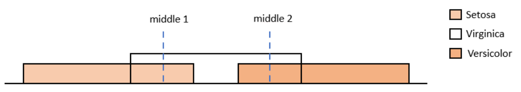


Figure 4: Second approach algorithm - calculating mean value for overlapping intervals

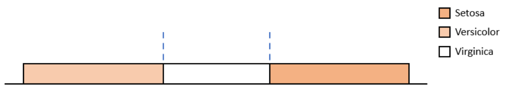


Figure 5: Second approach algorithm - determined intervals

Algorithm 3 Soft set algorithm - second approach

Input: Test set, validation set, weight
Output: The class to which the sample was classified

```
Create sorted list of all minimal and maximal values
of each iris type form test set.

Creates nested list of new ranges for each iris type
where mean value was calculated for overlapping sets
and taken as new edge value for set.

$sortedValues \leftarrow list\ of\ each\ iris\ type\ range$
$newRanges \leftarrow calculated\ iris\ type\ ranges$
for $index < len(validation\ set)$ do
    $row \leftarrow list\ of\ traits\ values\ of\ one\ sample$
    $sampleType \leftarrow iristypename$
    $sampleValue \leftarrow 0$
    $i \leftarrow 0$
    for $t\ in\ row$ do
        $sampleValue += t * weight[i]$
        $i += 1$
    if $sampleValue \in newRanges[0]$ then
        $Classified\ as\ Setosa$
    else if $sampleValue \in newRanges[1]$ then
        $Classified\ as\ Versicolor$
    else if $sampleValue \in newRanges[2]$ then
        $Classified\ as\ Virginica$
Returns classified type.
```

3.3. Naive Bayes

Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. At the beginning it reduces database by splitting an Iris database to three smaller databases according to their variety. After that classifier assigns the initial probability of a given species appearing in the database. Next, it takes a sample and counts a probability for each reduced database. It uses one of two considered distribution formulas. Subsequently, it multiplies the initial probability with all partial probabilities (with all attributes that a reduced database has). And at the end it compares which probability of three possible is the highest.

3.3.1. Formulas

Normal distribution:

$$P(a_i|V) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(s_{a_i}-\mu)^2}{2\sigma^2}) \tag{5}$$

Triangular distribution:

$$P(a_i|V) = \begin{cases} 0, & s_{ai} < \mu - \sqrt{6}\sigma_V \\ \frac{s_{ai}-\mu}{6\sigma^2} + \frac{1}{\sqrt{6}\sigma}, & \mu - \sqrt{6}\sigma \leq s_{ai} \leq \mu \\ -\frac{s_{ai}-\mu}{6\sigma^2} + \frac{1}{\sqrt{6}\sigma}, & \mu \leq s_{ai} \leq \mu + \sqrt{6}\sigma) \\ 0 & s_{ai} > \mu + \sqrt{6}\sigma) \end{cases} \tag{6}$$

a_i	a current attribute in reduced database
s_{ai}	a current attribute of a sample
σ	a standard deviation of an attribute
μ	a mean of an attribute in reduced database
V	current reduced variety database

Counting probability:

$$P(Variety) = P(Init) * \prod_{i=1}^{4} P(a_i|Variety) \tag{7}$$

3.3.2. Algorithm

To simplify how Naive Bayes actually works I will explain everything based on Iris database.

At the beginning, the Naive Bayes algorithm takes two parameters. First is a test set and the second is a validation set.

Afterwards, it splits the test set to three reduced databases according to their varieties.

Subsequently, it calculates a initial probability by counting the number of elements in reduced database divided by the number of all elements in the main database. Next, for a given sample it calculates a partial probability for each attributes in each reduced database.

To do so, it takes a list of elements in each attribute and then it calculates a mean and a standard deviation.

After that, it calls a distribution function which passes the given sample's current calculated attribute, the standard deviation and the mean.

Next, it multiplies the initial probability with four partial probabilities.

At the end, it returns the variety of the highest probability.

Algorithm 4 Naive Bayes algorithm

Input: Test set, sample
Output: The class to which the sample may belong

```
Make three reduced databases according to theirs varieties;
Make a list of attributes names in reduced databases;
Make a empty dictionary;
  $i \leftarrow 0$
  while $i < len(reducedDatabases)$ do
      $initProb \leftarrow \frac{len(reducedDatabases[i])}{len(dataBase)}$
      $probability \leftarrow initProb$
      $j \leftarrow 0$

      while $j < len(attributes)$ do
      Get a list of all values in the column[j]
      Calculate mean and standard deviation from the list
          $partialProb \leftarrow$ DistributionFunction(
        sample[j],
        standardDeviation,
        mean;)
       $probability \leftarrow probability * partialProb$;

    Add to dictionary a new record

Return from dictionary variety with the highest probability
```

4. Experiments

In order to properly analyze individual classifiers, we will perform a series of tests that will allow us to select the best classifier using the confusion matrix. Confusion matrix is used in assessing the quality of a binary classification. It describes how well the classifier classified given samples. It also gives us information about several things about the classifier such as:

1. Accuracy

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

2. Sensitivity

$$\frac{TP}{TP + FN} \tag{9}$$

3. Precision

$$\frac{TP}{TP + FP} \tag{10}$$

4. F1 Score

$$\frac{2TP}{2TP + FP + FN} \tag{11}$$

5. Specificity

$$\frac{TN}{TN + FP} \tag{12}$$

For this purpose we will make one confusion matrix based on one of three abstract class called Setosa. Based on that class the all three classifiers will be identify how well they classified given samples.

	Setosa	Versicolor	Virginica
Setosa	TP	FP	FP
Versicolor	FN	TN	FN
Virginica	FN	FN	TN

4.1. kNN

After testing, we noticed that the results for any k are very similar to each other. This may be due to the fact that the Setos class is significantly distant from the other two classes, which means that there is a very high probability that the objects closest to the sample will also be Stetosa. Below are the results for k equal to 1 2 3 and 4, respectively.

Table 1
Table for k = 1

k	TP	FP	TN	FN
1	16	0	27	2
AC	SEN	PRE	F1	SPE
0.96	0.89	1.00	0.94	1.00

Table 2
Table for k = 2

k	TP	FP	TN	FN
2	19	0	22	4

AC	SEN	PRE	F1	SPE
0.91	0.83	1.00	0.90	1.00

Table 3
Table for k = 3

k	TP	FP	TN	FN
3	14	0	29	2

AC	SEN	PRE	F1	SPE
0.96	0.88	1.00	0.93	1.00

Table 4
Table for k = 4

k	TP	FP	TN	FN
4	18	0	26	1

AC	SEN	PRE	F1	SPE
0.98	0.95	1.00	0.97	1.00

4.2. Naive Bayes

We considered in our article two distribution formulas. In this section we decide which of these two are the best for our database.

For the Normal distribution function results are:

Table 5
Results for Setosa class

Accuracy	Sensitivity	Precision	F1	Specificity
0.96	0.88	1.00	0.93	1.00

TP	FP	TN	FN
14	0	29	2

Table 6
Results for Versicolor class

Accuracy	Sensitivity	Precision	F1	Specificity
0.96	0.89	1.00	0.94	1.00

TP	FP	TN	FN
16	0	27	2

Table 7
Results for Virginica class

Accuracy	Sensitivity	Precision	F1	Specificity
0.96	1.00	0.87	0.93	0.94

TP	FP	TN	FN
13	2	30	3

For the Triangular distribution function results are:

Table 8
Results for Setosa class

Accuracy	Sensitivity	Precision	F1	Specificity
0.91	0.78	1.00	0.88	1.00

TP	FP	TN	FN
14	0	27	4

Table 9
Results for Versicolor class

Accuracy	Sensitivity	Precision	F1	Specificity
0.91	0.83	0.94	0.88	0.96

TP	FP	TN	FN
15	1	26	3

Table 10
Results for Virginica class

Accuracy	Sensitivity	Precision	F1	Specificity
0.91	0.92	0.80	0.86	0.91

TP	FP	TN	FN
12	3	29	1

After analyzing above tables we decided that Normal distribution is the best distribution function and it will be considered in the final test.

At the end we performed 100 test and we obtained the following results:

Table 11
Results for Setosa class

Accuracy	Sensitivity	Precision	F1	Specificity
0.95	0.87	1.00	0.93	1.00

Table 12
Results for Versicolor class

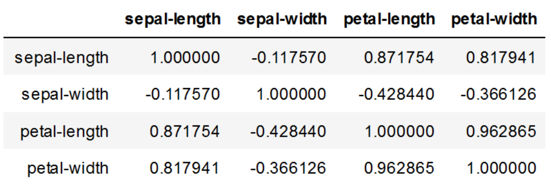
Accuracy	Sensitivity	Precision	F1	Specificity
0.95	0.93	0.94	0.93	0.97

Table 13
Results for Virginica class

Accuracy	Sensitivity	Precision	F1	Specificity
0.95	0.93	0.92	0.93	0.96

4.3. Soft sets

The determined values of Pearson correlation coefficients for each characteristic of iris flowers allowed choosing the weight of the features for the optimal classifier accuracy. An association between individual features was considered and their influence on the classifier efficiency. Based on these factors different weights were applied to select the most suitable solution.



	sepal-length	sepal-width	petal-length	petal-width
sepal-length	1.000000	-0.117570	0.871754	0.817941
sepal-width	-0.117570	1.000000	-0.428440	-0.366126
petal-length	0.871754	-0.428440	1.000000	0.962865
petal-width	0.817941	-0.366126	0.962865	1.000000

Figure 6: Pearson correlation coefficients for iris characteristics

Considering obtained correlation values, it was concluded that the most important characteristics are the following in descending order: petal-length, petal-width, sepal-length, and sepal-width. According to these observations, successively assigning different weight values, the best results were observed with weight $w = [0.1, 0, 0.5, 0.4]$.

The analysis of the results for both the first and the second algorithm showed that the first algorithm is a more effective soft set implementation. After performing 100 tests, the following results were obtained.

Table 14
Statistical results for first approach

Accuracy	F1	Sensitivity	Precision	Specificity
0.96	0.94	0.92	0.96	0.98

Table 15
Statistical results for second approach

Accuracy	F1	Sensitivity	Precision	Specificity
0.96	0.93	0.92	0.95	0.97

For the most efficient implementation, the following results were obtained for individual types of iris flowers. After performing 100 tests, the following results were obtained.

Table 16
Results for Setosa class

Accuracy	Sensitivity	Precision	F1	Specificity
0.98	0.95	1.00	0.97	1.00

TP	FP	TN	FN
18	0	26	1

Table 17
Results for Versicolor class

Accuracy	Sensitivity	Precision	F1	Specificity
0.98	0.93	1.00	0.97	1.00

TP	FP	TN	FN
14	0	30	1

Table 18
Results for Virginica class

Accuracy	Sensitivity	Precision	F1	Specificity
0.98	1.00	0.92	0.96	0.97

TP	FP	TN	FN
12	1	32	0

5. Conclusion

In order to establish the most efficient classifier, the prepared implementations were compared on the same partition of the Iris database. In comparison, the following results were statistically calculated.

Results for kNN:

Table 19
Results for Setosa class

Accuracy	Sensitivity	Precision	F1	Specificity
0.96	0.87	1.00	0.93	1.00

TP	FP	TN	FN
13	0	30	2

Table 20
Results for Versicolor class

Accuracy	Sensitivity	Precision	F1	Specificity
0.96	0.89	1.00	0.94	1.00

TP	FP	TN	FN
17	0	26	2

Table 21
Results for Virginica class

Accuracy	Sensitivity	Precision	F1	Specificity
0.96	1.00	0.87	0.93	0.94

TP	FP	TN	FN
13	2	30	0

Results for Naive Bayes:

Table 22
Results for Setosa class

Accuracy	Sensitivity	Precision	F1	Specificity
0.96	0.87	1.00	0.93	1.00

TP	FP	TN	FN
13	0	30	2

Table 23
Results for Versicolor class

Accuracy	Sensitivity	Precision	F1	Specificity
0.96	0.89	1.00	0.94	1.00

TP	FP	TN	FN
17	0	26	2

Table 24
Results for Virginica class

Accuracy	Sensitivity	Precision	F1	Specificity
0.96	1.00	0.87	0.93	0.94

TP	FP	TN	FN
13	2	30	0

Results for Soft sets:

Table 25
Results for Setosa class

Accuracy	Sensitivity	Precision	F1	Specificity
0.98	0.93	1.00	0.96	1.00

TP	FP	TN	FN
13	0	31	1

Table 26
Results for Versicolor class

Accuracy	Sensitivity	Precision	F1	Specificity
0.98	0.93	1.00	0.96	1.00

TP	FP	TN	FN
17	0	27	1

Table 27
Results for Virginica class

Accuracy	Sensitivity	Precision	F1	Specificity
0.98	1.00	0.93	0.97	0.97

TP	FP	TN	FN
14	1	30	0

Through analysis of attained results for all classes of all classifiers, it can be noted that the level of accuracy is the highest for the soft set classifier. The values of other statistically obtained characteristics also reach the highest levels for the previously mentioned classifier. It is worth mentioning that the obtained results are similar for particular characteristics of the kNN and Naive Bayes classifiers.

Based on the obtained results, it can be concluded that the soft set classifier implementation classifies most effectively. All of the implemented classifiers have been properly implemented. The results of the best classifier differ only by a few percentage points from each other.

The work and effort that was applied to completing this article are practical and applicable. This research offered an opportunity to learn and expand knowledge about the different approaches to assessing and teaching chosen classifiers as well as through the process of identifying the best solution. The analysis allowed acquiring practical experience in implementing machine learning algorithms.

In the future, the project could be extended and followed with further analysis of other classifiers, for instance through rebuilding the current classifiers in a more advanced way and selecting even more efficient solutions.

Table 28
Results

Classifier		Setosa	Versicolor	Virginica
kNN	ACC	0.96	0.96	0.96
	SEN	0.87	0.89	1.00
	PRE	1.00	1.00	0.87
	F1	0.93	0.94	0.93
	SPE	1.00	1.00	0.94
	TP	13	17	13
	FP	0	0	2
	TN	30	26	30
	FN	2	2	0
Naive Bayes	ACC	0.96	0.96	0.96
	SEN	0.87	0.89	1.00
	PRE	1.00	1.00	0.87
	F1	0.93	0.94	0.93
	SPE	1.00	1.00	0.94
	TP	13	17	13
	FP	0	0	2
	TN	30	26	30
	FN	2	2	0
Soft sets	ACC	0.98	0.98	0.98
	SEN	0.93	0.93	1.00
	PRE	1.00	1.00	0.93
	F1	0.96	0.96	0.97
	SPE	1.00	1.00	0.97
	TP	13	17	14
	FP	0	0	1
	TN	31	27	30
	FN	1	1	0

References

[1] Akram, M., Ali, G., Butt, M. A., Alcantud, J. C. R. (2021). Novel MCGDM analysis under m-polar fuzzy soft expert sets. Neural Computing and Applications, 33(18), 12051-12071.

[2] Chen, W., Zhou, Y., Zhou, E., Xiang, Z., Zhou, W., Lu, J. (2021). Wildfire risk assessment of transmission-line corridors based on Naïve Bayes network and remote sensing data. Sensors, 21(2), 634.

[3] Dong, W., Wozniak, M., Wu, J., Li, W., Bai, Z. (2022). De-Noising Aggregation of Graph Neural Networks by Using Principal Component Analysis. IEEE Transactions on Industrial Informatics.

[4] Dong, W., Wu, J., Bai, Z., Hu, Y., Li, W., Qiao, W., Woźniak, M. (2021). MobileGCN applied to low-dimensional node feature learning. Pattern Recognition, 112, 107788.

[5] Rani, P., Verma, S., Kaur, N., Wozniak, M., Shafi, J., Ijaz, M. F. (2021). Robust and secure data transmission using artificial intelligence techniques in ad-hoc networks. Sensors, 22(1), 251.

[6] Ruan, S., Chen, B., Song, K., Li, H. (2022). Weighted Naïve Bayes text classification algorithm based on improved distance correlation coefficient. Neural Computing and Applications, 34(4), 2729-2738.

[7] Shokrzade, A., Ramezani, M., Tab, F. A., Mohammad, M. A. (2021). A novel extreme learning machine based kNN classification method for dealing with big data. Expert Systems with Applications, 183, 115293.

[8] Siłka, J., Wieczorek, M., Wozniak, M. (2022). Recurrent neural network model for high-speed train vibration prediction from time series. Neural Computing and Applications, 1-14.

[9] https://c3.ai/glossary/data-science/classifier/

[10] https://www.sas.com/en_th/insights/articles/big-data/artificial-intelligence-machine-learning-deep-learning-and-beyond.html

[11] https://www.sciencedirect.com/topics/computer-science/machine-learning

[12] https://www.sciencedirect.com/science/article/pii/S0898122199000565

[13] https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

[14] https://monkeylearn.com/blog/what-is-a-classifier/