SAPIENZA
UNIVERSITÀ DI ROMA

FACOLTÀ DI INGEGNERIA CIVILE E INDUSTRIALE

CORSO DI LAUREA IN INGEGNERIA DELLA SICUREZZA E PROTEZIONE CIVILE

Project in:

MACHINE LEARNING FOR SAFETY SYSTEMS

(Prof. Michele Scarpiniti)

PREDICTING THE RECURRENCE OF THYROID CANCER BY USING MACHINE LEARNING APPROACH

Student:

Victor NADOLSKI

Student ID number : 2116554

June 6, 2024

CONTENTS

CHAPTER 1

INTRODUCTION

The purpose of the following report is to be able to predict the recurrence of thyroid cancer using a machine learning approach. In the report we will discuss different approaches and see under which configuration we have the highest chances to give the best prediction. We will present three different views of the dataset and three different algorithms. And we will decide which configuration is the best by using statistical tests.

In our report, we decided to tackle the problem by using a classification approach, thus using supervised machine learning.

CHAPTER 2

DATASET

The dataset that was used in this report is a thyroid disease dataset that we can find on the kaggle web page [4].

2.1 Dataset structure

The data set that we evaluated contains 17 features, where the last one is the target feature, that aim to predict recurrence of well differentiated thyroid cancer. The data set was collected for a duration of 15 years and each patient was followed for at least 10 years. In total, it represents 383 patients. In the following, you can find the description of all the features of the dataset.

1. Age: The age of the patient at the time of diagnosis or treatment. This is a numerical discrete feature.
2. Gender: The gender of the patient (male or female). This is a categorical nominal feature.
3. Smoking: Whether the patient is a smoker or not. This is a binary feature, "Yes", "No".
4. Hx Smoking: Smoking history of the patient (e.g., whether they have ever smoked). This is a binary feature, "Yes", "No".
5. Hx Radiotherapy: History of radiotherapy treatment for any condition. This is a binary feature, "Yes", "No".
6. Thyroid Function: The status of thyroid function, possibly indicating if there are any abnormalities. This is a categorical nominal feature.

7. Physical Examination: Findings from a physical examination of the patient, which may include palpation of the thyroid gland and surrounding structures. This is a categorical nominal feature.

8. Adenopathy: Presence or absence of enlarged lymph nodes (adenopathy) in the neck region. This is a binary feature, "Yes", "No".

9. Pathology: Specific types of thyroid cancer as determined by pathology examination of biopsy samples. This is a categorical nominal feature.

10. Focality: Whether the cancer is unifocal (limited to one location) or multifocal (present in multiple locations). This is a categorical nominal feature.

11. Risk: The risk category of the cancer based on various factors, such as tumor size, extent of spread, and histological type. This is a categorical ordinal feature.

12. T: Tumor classification based on its size and extent of invasion into nearby structures. This is a categorical nominal feature.

13. N: Nodal classification indicating the involvement of lymph nodes. This is a categorical nominal feature.

14. M: Metastasis classification indicating the presence or absence of distant metastases. This is a nominal categorical feature.

15. Stage: The overall stage of the cancer, typically determined by combining T, N, and M classifications. This is a categorical ordinal feature.

16. Response: Response to treatment, indicating whether the cancer responded positively, negatively, or remained stable after treatment. This is a categorical ordinal feature.

17. Recurred: Indicates whether the cancer has recurred after initial treatment. This is our target, "Yes", "No".

Therefore, from all of our features we have only one, the age, that is a proper numerical one. In consequence, we had to transform all the other features into numerical ones to be able to perform our algorithms. Since most of our features are nominal categorical, we decided to use a one-hot encoding method, which transforms every possible value of a feature into a specific vector. We did so because the order of feature's value is not important and using ordered numbers would likely confuse the learning algorithm. The downsize of this approach is that we increased the dimensionality of the feature vector and thus the running time of our code.

We also had to check if there were some missing data and in that case how we handle it. From a first view it seemed that no data were missing, but we still performed a calculation over all the features to be sure of our first intuition. From the calculation we were indeed right, there are no missing data so no need to handle them.

2.2 Dataset views

For our report we decided to use three different views of the dataset, which are the following : original, normalized and standardized. Let's delve into each one to understand better what do they mean.

- Original :
  We don't change the data, we leave them as they are provided.
- Normalized :
  We converted all the actual range of values into a standard range of values, typically in the interval [-1,1] or [0,1], in our case we chose the [0,1] scale.
  We used the following formula :

  $$X^i_{norm} = \frac{X^i - X^i_{min}}{X^i_{max} - X^i_{min}} \tag{2.1}$$

  Where X^i_{min} and X^i_{max} are, respectively the minimum and the maximum value of the feature i in the dataset.
- Standardized :
  We scale the values of the feature so that they have the properties of a standard normal distribution. Each feature will have a mean of 0 and a standard deviation of 1.
  We used the following formula :

  $$X^i_{std} = \frac{X^i - \mu^i}{\sigma^i} \tag{2.2}$$

  Where μ^i and σ^i are respectively the mean and standard deviation of the feature i in the dataset.

We could ask ourselves, why do we need to do dataset views. In fact, performing dataset views is crucial in machine learning for several reasons. Certain algorithms work better under certain type of dataset. For example, logistic regression model perform better when the features are on a similar scale. When features are on different scales, the optimization process can become inefficient, leading to slower convergence or getting stuck in local minima.

2.3 Training and test sets

In machine learning it's very important to have a training set and a test set of our datasets. We usually split our datasets into two parts, where the training set is bigger than the test set. The training set is used to train our models whereas the test set it used to evaluate the performance of the trained models. Optionally a validation set can be used, which is helpful to set the hyper parameters of our models, but we didn't do it in this exercise.

Since the goal of this exercise was to find the best model out of the three that we will present in the later part of the report, we decided to use a 10-fold cross validation approach. The method consist in the following. The dataset is randomly partitioned into 10 equal-sized folds. In the first iteration, the model is trained on the first 9 folds and tested on the 10th fold. In the second iteration, the model is trained on the second set of 9 folds and tested on the fold left out in the first iteration. This process is repeated until each fold has been used as the test set once. After completing the 10 iterations, the performance metrics from each iteration are averaged to produce a final score.

This method helps ensure that the model's performance is not dependent on the particular train-test split, providing a more robust evaluation.

CHAPTER 3

MODELS

In our exercise, we used the supervised machine learning approach. We chose three different models of learning algorithms which use the classification method. The goal of a classifying algorithm is to predict the class to which an object belongs based on it features. In the training phase, the algorithm detects patterns for features of the trained dataset. And since it has the corresponding target for each of the data instances the algorithm can associate each pattern to a specific target. The classification algorithms are really helpful in disease diagnoses because they are able to predict if a certain patient has a disease or not based on his medical record. That is why we chose this approach.
The most common algorithms are : decision trees, random forest, support vector machines, neural networks, logistic regression. In our case we chose decision trees, logistic regression and k-Nearest Neighbors (k-NN).

3.1 Model 1 - k-Nearest Neighbors (k-NN) [1]

The k-Nearest Neighbors algorithm is an instance-based learning algorithm, which means that it constructs hypotheses directly from the training instances. It compares new problem instances with instances seen in the training. The algorithm operates on the principle of similarity, where it predicts the label of a new data point by considering the labels of its K nearest neighbors in the training dataset. It can work on both numerical and categorical data. Here's the step by step explanation of how it works :

1. Step 1: Selecting the optimal value of K

   K represents the number of nearest neighbors that needs to be considered while making prediction.

2. Step 2: Calculating distance

   To measure the similarity between target and training data points, Euclidean distance is used. Distance is calculated between each of the data points in the dataset and target point.
   The Euclidian distance is given by the following formula :

   $$d(x_i, x_j) = \sqrt{\sum_p (x_{ip} - x_{jp})^2}$$

   Where x_{ip} represents the p-th feature of x_i.

3. Step 3: Finding Nearest Neighbors

   The k data points with the smallest distances to the target point are the nearest neighbors.

4. Step 4: Voting for Classification

   In the classification problem, the class labels are determined by performing majority voting. The class with the most occurrences among the neighbors becomes the predicted class for the target data point. Therefore it is good practice to put as the number of neighbors, k, an odd number, to avoid ties.

Here's an image to visually see how the algorithm works :

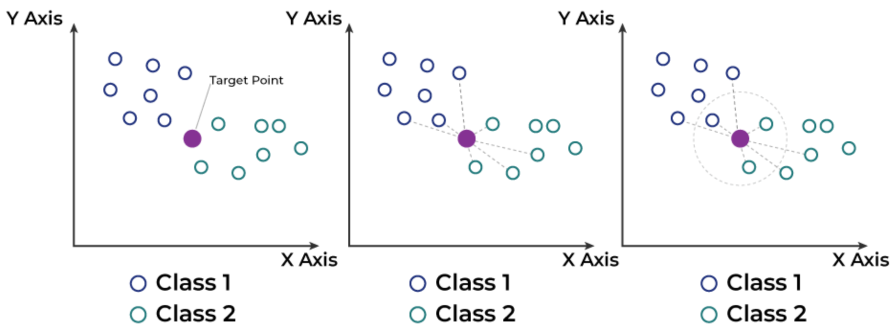

Fig. 3.1: Image showing how the k-nearest-neighbors algorithm works.

The advantage of this algorithm are the following :

- simple to implement
- training is efficient

- robust to noisy data

The disadvantage of this algorithm are the following :

- sensitive to range of features
- curse of dimensionality - the higher the number of data the longer the algorithm needs to run because it needs to compute the distance between the data we consider with all the the training data.

Therefore because of the simplicity of the algorithm and the results that it provides, we decided to choose it, as one of the three models that we want to implement to our dataset.

3.2 Model 2 - Logistic Regression [2]

In our case, since our target feature "Recurred" is binary, "Yes" or "No", we can simplify the explanation of the logistic regression model to a Bayesian classification. But the reader needs to be aware that there are differences between a binomial logistic regression and a multinomial logistic regression, in particular in the use of functions.

In the case of a binomial logistic regression model, the algorithm considers that there is a linear model of our training data and that after finding the optimum parameters of the linear model, it can pass the test data on this model. After having past the test data, by the use of a sigmoid function it changes the result into a probability between 0 and 1. Then by using the probability result, it compares it to a threshold value. If the probability is higher than the threshold value then the data point belongs to the class y=1, that can represent "Yes" in our case. If it's below the data point belongs to the class y=0, that can represent "No" in our case. Here's a more detailed step by step explanation

1. Linear model:

   We have our train data $\boldsymbol{X}^{(\mathrm{n^*m})}$ and based on our trained data, the algorithm assumes a linear model to describe them of the following form :
   $$z = \boldsymbol{w} * \boldsymbol{X} + w_0 \tag{3.1}$$
   With the training data the goal is to find the $\boldsymbol{w}$ = [w_1,w_2,..., w_m] the weight vector and w_0, the bias parameter. Those parameters are found by maximizing the likelihood function.

2. Implementing the test data and using the sigmoid function :

   After the trained phase, the algorithm should have found the weight vector and the bias parameter. Therefore it can calculate the linear combination 3.1 with the test set and implement the sigmoid function to have number between 0 and 1. The sigmoid function is the following :

$$\sigma(z) = \frac{1}{1 + e^{-(\boldsymbol{w} * \boldsymbol{x} + w_0)}} \tag{3.2}$$

   And can be visualize below :

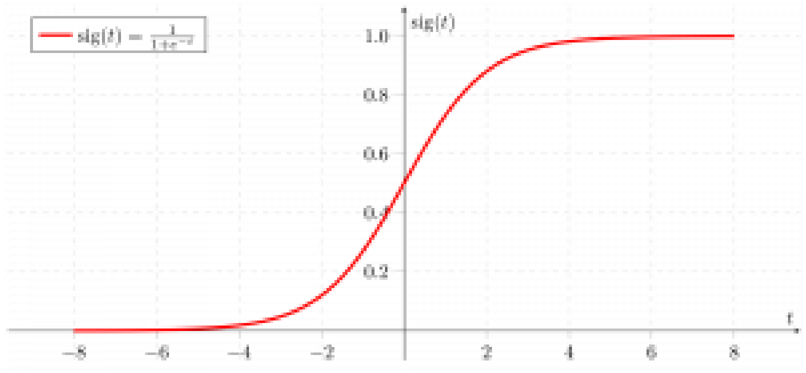

Fig. 3.2: Graph showing the sigmoid function.

   The sigmoid function converts the continuous variable data into the probability i.e. between 0 and 1.

3. Assigning values :

   After finding the probability value for our test data we can compare it to a threshold value, usually 0.5. If the probability is higher than 0.5 we assign the value one to this test data, otherwise we assign the value 0.

The advantages of this algorithm are the following :

- Simple and interpretable
- Probabilistic output
- Computationally efficient
- Works well with linear separable data

The disadvantages of this algorithm are the following :

- Assumes linear relationship

- Overfitting with high dimensionality

Therefore because of the simplicity of the algorithm, the probabilistic output that it provides and the computational efficiency, we decided to choose it, as one of the three models that we want to implement to our dataset.

3.3 Model 3 - Decision tree [3]

A decision tree is a hierarchical data structure implementing the divide and conquer strategy. It can be seen as a flow chart to take a decision. It is a acyclic graph that is used to take decisions. In each branching node of the graph, a specific feature j of the feature vector is examined. If the feature is below a specific threshold the the left branch is followed, otherwise the right branch is followed. The recursive split is made until the reach of the terminal leaves. Then a decision is made about the class to which the example belongs. The decision node divides the input space into two, which is called a binary split. Therefore the prediction cost of using a tree is fairly low $\Theta(logN)$, where N is the number of datapoints. You can find below a visual representation of how a decision tree works.

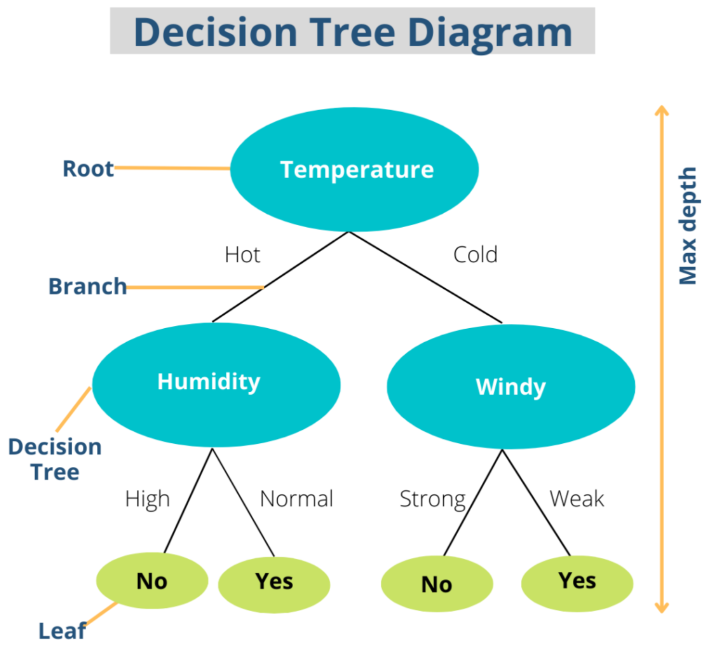

Fig. 3.3: Picture showing a decision tree diagram, explaining how it works with the terminology.

Therefore to make a decision tree work well there are 2 main criteria that needs to be considered. The depth of the tree, which means the number of branches, and the splitting criterion for each of the nodes.

The depth of the tree represents the maximum number of level in the tree, from roots to the leaves. Therefore a decision tree with a high depth can capture more details and complexities in the training data, potentially leading to a model that fits the training data very well. But the risk is that it leads to overfitting. On the contrary if the tree is not deep enough the risk is that it the tree might not capture enough details from the training data, resulting in underfitting. Therefore choosing wisely the depth of the tree is crucial.

For the splitting criterion there are multiple options like the Gini impurity or the Information Gain. In our case we will present the formula for the Gini impurity because it is the one we chose in our model.
The Gini index is computed as followed :

$$GI = 1 - \sum_{i=1}^{K} (p_m^i)^2 \tag{3.3}$$

Where p_m^i is the proportion of instances belonging to class i at the node we are looking at. To choose the data under which we are splitting we have to choose the split under which we minimise Gini's impurity.

The advantages of this algorithm are the following :

- Simplicity and Interpretability : mimics human decision-making process
- No need for feature scaling
- Handles non linear relationships between the feaures and the target

The disadvantages of this algorithm are the following :

- Overfitting

Therefore because of the simplicity of the algorithm and the generality of its use, we decided to choose it as one of the three models that we want to implement to our dataset.

CHAPTER 4

TOOLS

For this exercise, we used the Python software tool to implement all the code and study our models on the dataset. The code was written in Visual Studio Code application.

4.1 Preprocessing

To do the preprocessing of the data we imported the sklearn modules in our code, with the following line of code :

```
from sklearn.preprocessing import MinMaxScaler,
                                  StandardScaler
```

Since we are only interested in original, normalised and standardized data views it's the only one preprocessing module that we imported. Then to effectively use this module we did the following :

```
# Extracting the feature matrix
# and the target vector (original view)
X1 = dataset2.drop('Recurred',axis=1).values
y  = dataset2['Recurred'].values

# Normalization
n_scaler = MinMaxScaler(feature_range=(0, 1))
X2 = n_scaler.fit_transform(X1)

# Standardization
s_scaler = StandardScaler()
X3 = s_scaler.fit_transform(X1)
```

But generally speaking, sklearn modules are available modules very useful for machine learning and not only for preprocessing of the data. With this module we also imported the different algorithms and performance indices.

4.2 Implementation

After preprocessing of the data we also imported our three algorithms, k-NN, Logistic Regression and Decision Tree as it can be seen below :

```
    # Set the parameters of the chosen classifiers
classifiers = [
    KNeighborsClassifier(n_neighbors=5),
    LogisticRegression(max_iter=3000,tol=0.01),
    DecisionTreeClassifier(max_depth=5)]
```

For the k-NN we decided to take 5 number of neighbors in the implementation of the model, since we have 383 patients. With 3 neighbors the results were less accurate and with 7 neighbors there was a risk of overfitting and the running time was longer.

For the logistic regression the 'tol' hyperparameter represents the tolerance under which the algorithm should stop running. The algorithm stops when the change in the cost function between iterations is less than the specified 'tol' value, here 0.01. As for the 'max iter' hyperparameter, during logistic regression, the optimization algorithm iteratively updates the model parameters to minimize the cost function. If the model does not converge to a solution within the specified number of iterations, here 3 000, the process will stop once 'max iter' is reached. Therefore settings those hyperparameters are important while regarding the runnning time of the algorithm.

For the decision tree, we took a maximum depth of 5 because the results were better with this number and we didn't risk overfitting, because we have 16 features. Also the default splitting criterion here is the Gini index. Describe how the selected models have been implemented. In particular, it is appropriate to carefully discuss the setting of the hyper-parameters for the selected models.

4.3 Performance Indices

Performance indices are crucial statistical tools to see the performance of our models on the dataset. There are many of them and for our case for the comparison of the models we chose to look at the following ones : accuracy, precision, recall, f1-score, with a particular look on recall. To understand better those performance indices we first need to understand the results of models. There are four possibilities :

- True Positive (TP): The number of positive instances correctly classified by the model.
- True Negative (TN): The number of negative instances correctly classified by the model.
- False Positive (FP): The number of negative instances incorrectly classified as positive by the model.
- False Negative (FN): The number of positive instances incorrectly classified as negative by the model.

With those results this is how we compute our performance indices :

$$\begin{aligned} Accuracy &= \frac{TP + TN}{TP + TN + FP + FN} \\ Precision &= \frac{TP}{TP + FP} \\ Recall &= \frac{TP}{TP + FN} \\ F1 - Score &= 2 * \frac{Precision * Recall}{Precision + Recall} \end{aligned}$$

Therefore in our case, for a medical prediction we are particularly interested by the recall value. Indeed, we want to have as less as possible false negative results and the most possible true positive results. Here false negative would mean that we didn't identify the recurrence of the cancer in our diagnoses and true positive would mean that we did predict well the recurrence of the cancer. Therefore, we want to look at the configuration that maximizes the recall, and this was the key performance indicator for our case. After running all the configuration we looked at which one had the best recall and decided to chose it as the best data view/ algorithm configuration.

With the outputs of the models we can also build a confusion matrix, that presents the results in a visual way as follows :

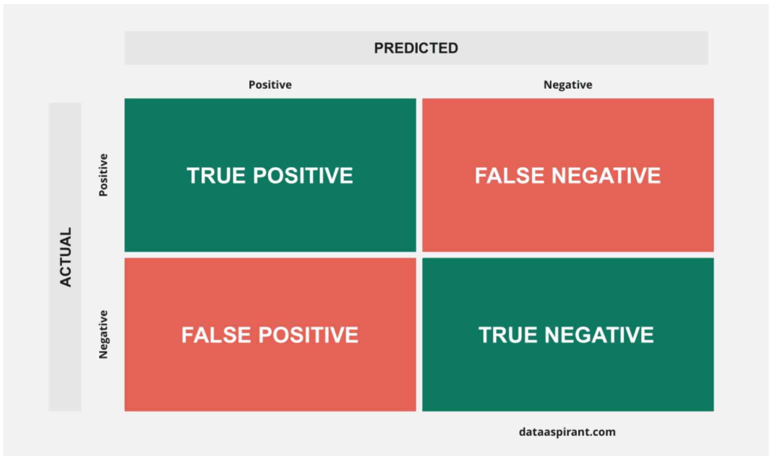

Fig. 4.1: Picture showing the confusion matrix. The more we have numbers on the diagonal the better it is. In ideal case there are no numbers on the antidiagonal.

We also decided to compute the confusion matrix after having chose which configuration was the best. This helped us to visually verify if the model was a good one. Another good visual approach is to make box plots.

CHAPTER 5

RESULTS

For each model we computed the accuracy, precision, recall and F1-score and look at the mean result and the corresponding standard deviation. We presented all the results in tables.

5.1 Results of model 1 - k-Nearest Neighbors (k-NN)

Model: kNN								
Orig	Acc	86.45 +/- 6.63	Prec	87.18 +/- 6.33	Rec	81.10 +/- 10.44	F1	81.52 +/- 9.81
Norm	Acc	87.21 +/- 10.83	Prec	89.22 +/- 10.26	Rec	83.84 +/- 12.49	F1	83.52+/-13.34
Std	Acc	88.53 +/- 9.29	Prec	91.10 +/- 8.99	Rec	83.63 +/- 12.58	F1	84.29 +/- 12.57

Table 5.1: Table showing the accuracy, the precision, the recall and the F1 score for the k-Nearest Neighbors model.

For the case of the k-NN model if we first look at the recall we can see that the normalised and standardised data views have a higher value than the original one. But then when we look precisely between normalised and standardised recall values we can see that they are similar, order of tenth place, so hard to separate between them. Therefore then if we compare the two data views with the other statistical tests we see that the standardised data views have a better result in all the other categories. Therefore, for the k-NN model the standardised data views seem to score the best.

5.2 Results of model 2 - Logistic Regression

Model: LR								
Orig	Acc	91.63 +/- 7.62	Prec	92.74 +/- 7.61	Rec	90.53 +/- 6.18	F1	90.19 +/- 7.90
Norm	Acc	90.34 +/- 7.79	Prec	92.04 +/- 7.45	Rec	88.25 +/- 8.40	F1	88.09 +/- 8.93
Std	Acc	91.10 +/- 8.01	Prec	92.01 +/- 7.84	Rec	90.12 +/- 6.21	F1	89.63 +/- 8.23

Table 5.2: Table showing the accuracy, the precision, the recall and the F1 score for the logistic regression model.

For the case of the logistic regression model if we first look at the recall we can see that the original and standardised data views have a higher value than the original one. However the difference of the recall values between the two is not very high, order of tenth place, so hard to separate between them. Therefore then if we compare the two data views with the other statistical tests we see that the original data views have a better result in all the other categories. Therefore, for the logistic regression model the original data views seem to score the best.

5.3 Results of model 3 - Decision tree

Model: DT								
Orig	Acc	90.30 +/- 15.36	Prec	91.42 +/- 11.38	Rec	87.88 +/- 11.17	F1	87.76 +/- 13.44
Norm	Acc	90.30 +/- 13.12	Prec	92.12 +/- 11.12	Rec	87.92 +/- 12.08	F1	87.29 +/- 15.01
Std	Acc	89.51 +/- 15.17	Prec	91.94 +/- 10.95	Rec	88.83 +/- 11.58	F1	88.20 +/- 14.06

Table 5.3: Table showing the accuracy, the precision, the recall and the F1 score for the decision tree model.

For the case of the decision tree model if we first look at the recall we can see that the standardised data views have a higher value than the other ones. Moreover with a lower variance than for the normalised values. The difference of the recall values is of the order of the unit place. Therefore, for the decision tree model the standardised data views seem to score the best.

5.4 Results comparisons

In order to do results comparisons we can start with the use of box plots. The box plots can give us a first visual approximation of which of the model is the best, with which data view. Indeed, the box plots are a standardized way of displaying the distribution of data based on a five-number summary: minimum, first quartile (Q1), median, third quartile (Q3), and maximum. It provides a graphical representation of the central tendency, spread, and skewness of a dataset. So in our case, we can see how accurate are each of our models with each of the data view. The smaller is the box and the higher is the median value, the more accurate is our configuration.

You can find below the three box plots for our three data view.

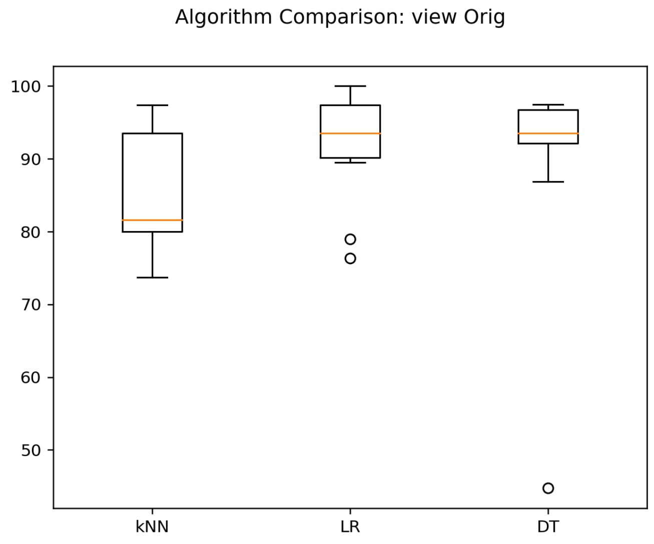

Fig. 5.1: Box plot of the three different models on the original view of the trained data.

For the original data view the logistic regression model seems to be the best one. Indeed the median is high, above 90% and the variance is small. The decision tree could also be a good candidate because it has similar parameters.

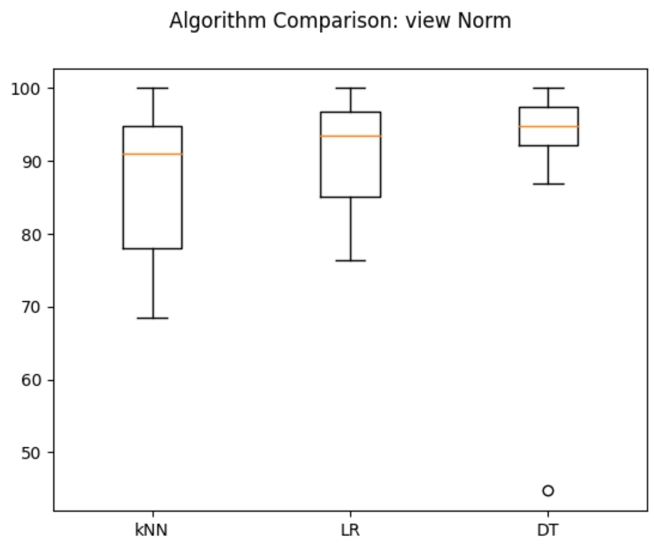

Fig. 5.2: Box plot of the three different models on the normalised view of the trained data.

For the normalised view it is clear that the decision tree is the best model. Highest median value and low variance.

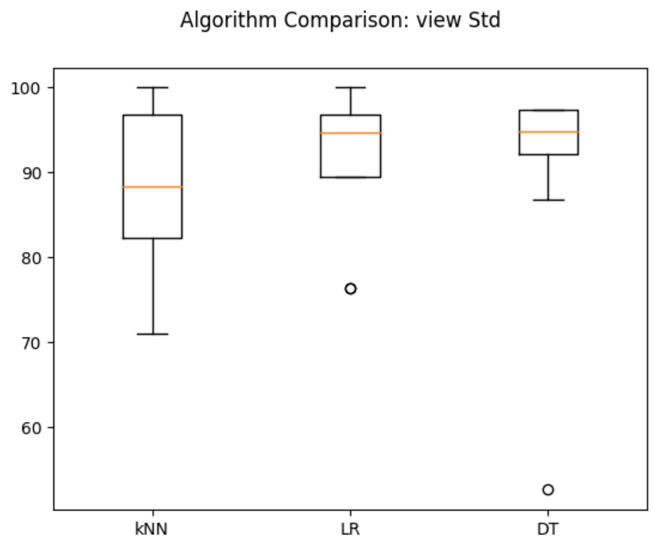

Fig. 5.3: Box plot of the three different models on the standardised view of the trained data.

As for the standardised data both the logistic regression and the decision tree are good. They both have a high median value and a small variance.

After doing the box plot we can have a first idea of which model is going to be the best for us. In our case there is a debate between the logistic regression and the decision tree. But since we are mostly interested by the recall for our exercise we decided to compute the recall for each of the configurations. Below you can find the results.

Recall	**kNN**	**LR**	**DT**
Orig	78.28 +/- 10.86	90.53 +/- 6.18	87.88 +/- 11.17
Norm	83.84 +/- 12.49	88.25 +/- 8.40	87.92 +/- 12.08
Std	83.63 +/- 12.58	90.12 +/- 6.21	88.83 +/- 11.58

Table 5.4: Table showing the comparison results in terms of recall (in percentage) of the three considered models with the three considered data views.

As we can see from the table the best configuration is the logistic regression with the original data views. Indeed it is where we observe the highest recall and the lowest variance of all the possibilities. Therefore this is the configuration we chose.
After having chose this model, we then performed another run with the dataset to do a confusion matrix and see the output. We took a test size of 33% size of the full dataset, which corresponds to 127 cases.

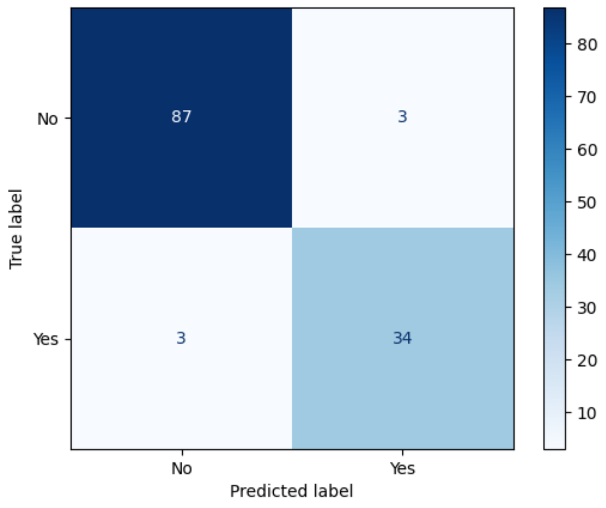

Fig. 5.4: Confusion matrix for the original data views and the logistic regression algorithm.

As we can see from the confusion matrix most of our prediction are on the diagonal, only 6 of them are on the anti diagonal. We have an overall accuracy of 95.28 % and a recall of 96.66 %. Therefore our configuration is indeed very efficient.

CHAPTER 6

CONCLUSIONS

For our problem of predicting the recurrence of thyroid cancer we saw that the machine learning approach is a very powerful one. Indeed it is quite easy to implement and allows us to use different algorithms and models and at the end predict with a good accuracy if a patient is recurred from the thyroid cancer or not. In this report we tried different approaches to tackle the problem and then to determine which one is the best. We used three data views : original data, standardised and normalised one. As well as three algorithms : k-NN, logistic regression and decision tree. To determine which configuration was the best we looked at different statistical tests: accuracy, precision, recall, F1 score, with a particular attention to recall, since we do a medical prediction. We also displayed some visual tools like the box plots and the confusion matrix, to visually see if our models are efficient or not. From all the configuration we used and based on the results of the recall tests we saw that the original data view with the logistic regression algorithm is the best configuration for our problem.

BIBLIOGRAPHY

[1] https://www.geeksforgeeks.org/k-nearest-neighbours/.

[2] https://www.geeksforgeeks.org/understanding-logistic-regression/.

[3] https://www.geeksforgeeks.org/decision-tree/.

[4] https://www.kaggle.com/datasets/jainaru/thyroid-disease-data/.