Grado en ciencia e ingeniería de datos

Introduction to data science - Academic year: 2020/21

Report II - Supervised classification techniques

LUCÍA CORDERO SÁNCHEZ

ALONSO MADROÑAL DE MESA

Instructors: ***David Delgado Gómez***

Danae Carreras García

Date: 4th December, 2020

Contents

1 Supervised techniques: purpose, pros and drawbacks

Supervised techniques are based on a set of input variables and their goal is to predict the output variables. They are both previously known and we create a model from a training data set, dividing the data base in training data and test or evaluation data.

With the first one we build and fit our model, while the second one is used to test how accurate it is to make predictions. Finally, we work with the label or class we already have from the beginning to compare the accuracy rates.

The reason why we chose these techniques to work with them was simple: as our training data had dependent and independent variables or a label (in this case, **Survived**), we could create clusters with our data (because the variables did not have the same importance).

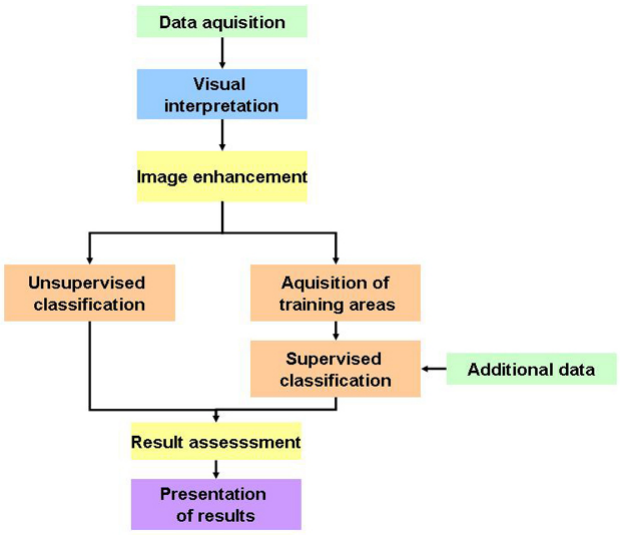


Figure 1: Essential steps in the data management process

Supervised and unsupervised classification are clearly used for different purposes, they do not only distinguish in the treatment of variables. Among the reasons why we could not adapt our data to the unsupervised techniques, some of them are:

- In unsupervised classification, the user cannot provide the class or label, since it is considered "prior knowledge" of the data. Data is classified in a particular way: first the algorithm looks for the similarities with each other and, and once it determines which of them are related it groups them into classes.

  In this sense, the existence of a class or label we needed to take into account did not allow clusters to be formed.

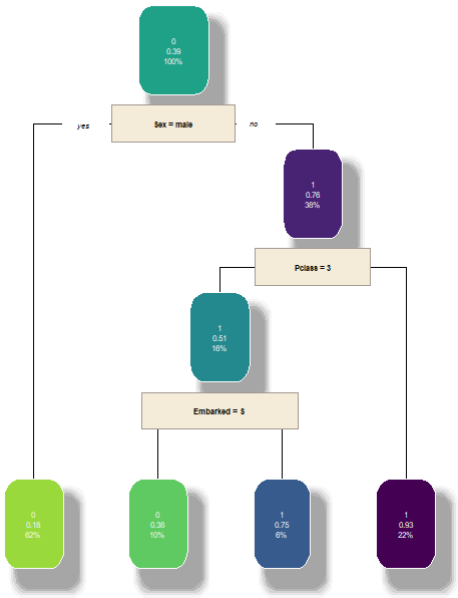
- Supervised classification is human-guided, while unsupervised methods are mainly calculated by software. It represents a huge drawback for us since the tools we could control were not limitless, even if it is much faster and easy to carry out.

  Being controlled by software also minimises the human error, but these techniques does not consider any spatial relationship.

2 Exploratory Data Analysis (EDA)

To fulfill our objective of predicting whether a passenger survived or not after the sink, we made an EDA task.

First, we depured our data base by removing the doubled values and the variables in which there were rows that remained missing or misvalues. Recovering the data and conclusions from our last project, we built up a new tree taking all into account except from the variables that we did not consider important.



In specific, we removed **Ticket** because it is a variable randomly created (a factor with many levels), and we could not find any relationship between the ticket letters and the survival rate. Also **Cabin**, as this variable is not useful for predictions in this case because in each case it has a format and it was registered in a different way (more letters than numbers or the other way around; sometimes it has lower letters and in other passengers, capital ones, or a passenger may have more than one room assigned). Afterwards, we identified the label and established the relationships between all the variables.

```
#As the cabin variable is incomplete we will eliminate it from the data
library("dplyr")
data <- slice(data, -c(which(duplicated(data))))
```

3 Decision trees

3.1 Cross Validation hypertuning

Cross validation is a resampling procedure used to evaluate models (which means, predicting data that was not used in fitting the model) avoiding overfitting problems. In general, it is a method to give an insight on how to create an model that will generalize to a independent data set; which means, an algorithm that can be used to solve a problem in real life.

First, we created a grid for the parameters of the decision tree before applying cross validation.

```
1
2 grid = expand.grid(minsplit = seq(10,60,10),
3                    cp = 2^(0:-10),
```

Secondly, we established the number of folds (5), created the corresponding sample, split the data so that it did not overlap, fit the tree with each of the values of the tree and made the prediction with the test data. We kept the best accuracy from all that were created throughout each parameters' combination. As we expected, the accuracy was reasonably good but not the best.

```
1  n_fold = 5; grid$accuracy = 0
2  folds = sample(rep(1:n_fold, length.out = n))
3  for (j in 1:n_fold){
4    # split
5    train = data[folds != j,]; test = data[folds == j,]
6    for (i in 1:nrow(grid)){
7      # fit and evaluate
8      tree = rpart(Survived~., train,
9                   control = rpart.control(minsplit = grid$minsplit[i],
10                                          maxdepth = grid$maxdepth[i],
11                                          cp = grid$cp[i]))
12     pred = predict(tree, test, type = "class")
13     acc = (sum(pred == test$Survived)/length(pred))
14     # average
15     grid$accuracy[i] = grid$accuracy[i] + acc/n_fold
16   }}
17 maxacc = grid[which.max(grid$accuracy),]
```

These were the chosen parameters depending on the best accuracy for cross validation.

```
> maxacc = grid[which.max(grid$accuracy),]
> maxacc
    minsplit        cp maxdepth  accuracy
511       10 0.00390625        8 0.8041162
```

3.2 Repeated Cross Validation for hypertuning

Repeated cross validation represented a way to "improve" how accurate the predictions were in cross validation, whose splits repeated only once and not overlapping may have very different results.

Consequently, we expected repeated cross validation to have a better performance, since repeating the cross-validation procedure multiple times and reporting the mean result across all folds from all runs would reduce the error.[1]

```
> table[which.max(table$accuracy),]
   minsplit       cp maxdepth  accuracy
41       50 0.015625        1 0.8727273
```

In this case, we see a **small improvement** compared to the accuracy in 3.1., as we already thought it would be. In our hypothesis, we stated that increasing the k would perform better because it reduces the difference dependent on the model and on the data set from one time to the next one.

[1] All the graphics and numbers we display are related to a specific set.seed due to the random factor, so in reality this numbers will change every time we execute the code, this is an approximation.

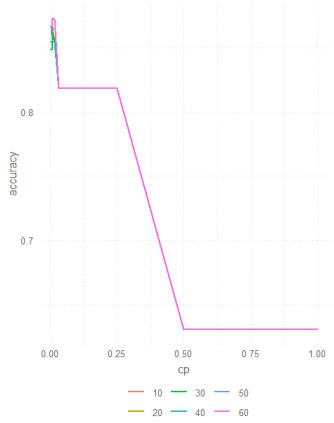


Figure 2: Accuracy depending on the minsplit (colors) and complexity parameter

Each repeat of the k-fold cross-validation process is performed on the same data set split into different folds.

This method works well in simple and small models (*e.g.* linear), like this one, which is not computationally complex to fit and evaluate.

3.3 Decision tree pruning

About the problematic when implementing pruning on our trees. As our data set is not really that large, our best decision trees had little to no depth, which made pruning not so interesting and not really useful (We should remember that, after all,pruning is a technique used to remove redundant sections of our model).

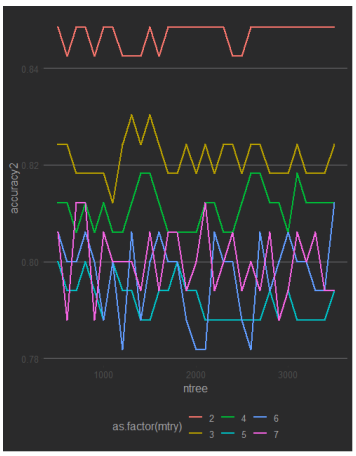
We saw that pruning was not the best option (not even a good one), we decided not to implement it on our code but to comment as a curiosity in the report to show that not all methods are always useful (as seen in the *free lunch theorem*).

4 Random Forest

Random forest (as its own name says) is a considerable set of trees trained individually with slightly different parameters. The new prediction is made taking into account the predictions of all the trees that are in the model. This method seems to be very accurate and easy to apply in a wide range of problems. Some of their advantages are:

- Random forest is not very influenced by outliers. Even if in this case, there is no possibility if we are predicting if passengers survived or not, when referring to predicting other variables such as Fare, it would be useful.
- They do not require a complex task of preprocessing or cleaning the data compared to other method, e.g. no need to standarize data.
- It is easy and fast to identify the most important variables.

- They are easy to interpret even if there were complex relations among variables.
- Random forests "solve" an issue of decision trees: its tendency to high variance and overfitting, by repeating the same process many times and reducing the margin of error.




```
> max(accuracy2)
[1] 0.8484848
```

In our case, the random forest technique worked always better when the *mtry* (number of variables randomly sampled as candidates at each split) was 2 and the number of tress suffered some ups and downs but at the end it stabilized (in $mtry = 2$). In spite of this, the improvement in the performance from the random forest with the hyperparameters tunned compared with the one with standard parameters is not very appreciated.

We built up the table comparing the predictions for the default parameters of random forest and the real references. Moreover, we obtained the **true positive rate, true negative rate** and **correct classification rate** or accuracy:

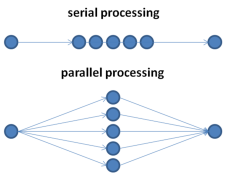
```
> t = table(pred, test$Survived);
> t

pred  0  1
   0 90 10
   1 12 53


       ccr       tpr       fpr
 0.8363636 0.7142857 0.07368421
```

4.1 Boosting for Random Forest

Parallel processing consists on running two or more processors (CPUs) to handle **separate parts of an overall task**; in other words, breaking a task into small tasks and using multiple processors in order to optimize the execution time.



In this code, we realised it was almost impossible to reach a standard efficiency without using Parallel Processing, because it took a while for the program to execute all the code and make all the calculations. After incorporating the *foreach* function, execution time reduced.

5 K-nearest neighbors algorithm (KNN)

KNN is a technique that bases the predictions in the data from the environment around them: it classifies data depending on the "neighbors" or the closest data around them. This method is very important when talking about machine learning in data mining, .e.g. in the system of recommendations of social media.

We created four data frames: one for the train and test without the label, and two for the factor of true classifications of training and test set.

Apart from applying parallel programming, we had the hypothesis that normalizing the data would be a good idea and the output would not be biased. We had variables such as sex (male or female), thus normalizing variables was very problematic and we could not work with them. Moreover, when we had the variables standarized the accuracy for all neighbors was almost perfect (between 0.97 and 1) and perfect for small k, which is not recommendable if we want to avoid overfitting.

The main advantages and disadvantages we found for this technique are:

- No training period: KNN does not include a data training period because the data itself is a model that will be a reference for future predictions. New data can be always added without affecting the model.
- It is easy to implement because it only requires the calculation of distances between the points of the data,and there are formulas (like Manhattan or Euclidean) for it.
- It does not work well with large data sets, high dimensionality (the more dimensions, the harder it is to calculate distances) and it is sensitive to "noisy" or missing data.

First, we turned all the factor type of variables into numeric variables. Likewise, we proceeded to do some hypertunning with KNN incorporated. This is the accuracy we obtained:

```
> acc = grid3$accuracy[which.max(grid3$accuracy)]
> acc
[1] 0.8484848
```

We did not expect a good accuracy rate because our data was already very "labeled" or classified, but the assumption was mistaken.

6 SVM

Support Vector machine is a supervised technique related to classification and regression, making a prediction model through a hyperplane or a set of high-dimensional hyperplanes defined as the vector between two points (from the two classes) which are closer to the support vector. When new data enter the model, they are classified in one or another class depending on the area where they are set into.

For this method, we tried to figure out which kernel fitted better and gave the best accuracy: first we used linear kernel, later polynomial kernel and finally radial. In all of them we intented to observe for which degree or gamma (in polynomial and radial kernel, respectively) the accuracy was better.

The main advantages we found were that SVM is very effective in high dimensional spaces or when the number of dimensions is greater than the number of samples, and it works well for data with a clear margin of separation. In contrast, it does not work very well for large amount of data or when data have a lot of noise.

When applying polynomial kernel, we created a grid of degree between 1 and 100 and applied support vector machine. This wide interval of degrees may be the reason why the most accurate model is a polynomial kernel type.

This is the accuracy we obtained in linear kernel:

```
> accuracy2 = sum(pred == test$Survived)/length(pred)
> accuracy2
[1] 0.830303
```

Figure 3: Accuracy depending on linear kernel in svm

With the polynomial kernel, we appreciate the best accuracy is the best from all the models when the hyperplane is bidimensional.

```
> grid4[which.max(grid4$accuracy),]
  degree  accuracy
2      2 0.8606061
```

Figure 4: Accuracy for the best polynomial kernel

Finally, in radial kernel we iterate the model for *gamma* taking values between 2**(-70) and 2**(90) and the best result we obtain is:

```
> grid5[which.max(grid5$accuracy),]
    gamma  accuracy
67 0.0625 0.8424242
```

Figure 5: Accuracy for the best gamma in radial kernel

7 Final conclusions

To cover up all the research we have made, we would like to make some important appointments:
First, we got to the conclusion that the most accurate methods had a strong correlation with the noise of the data; the more resistant techniques were fighting noise, the more accurate they were.

Although unsupervised classification is very useful in some specific cases, in small data bases it is easier to work with supervised techniques; in the first place, we have more control over them, and secondly they fit better in models where all variables do not have the same importance or whether all variables depend on one.

We estimate that, based on the previous work, the **Repeated Cross Validation applied to Random Forest** is the technique that will fit the future data the best because Random Forest is very noise-resistant.

8 References

Setting seeds:
https://youtu.be/zAYzAZwufKI

Slicing datasets:
https://genomicsclass.github.io/book/pages/dplyr$_t$*utorial.html*

Decision Trees:
https://bradleyboehmke.github.io/HOML/DT.html
https://youtu.be/tU3Adlru1Ng
https://youtu.be/JFJIQ0$_2$*ijg*
https : *//youtu.be/*7*VeUPuFGJHk*
https : *//www.reddit.com/r/MachineLearning/comments/*2*ur*5*w*4/*question*$_w$*hen*$_m$*aking*$_{ad}$*ecision*$_t$*ree*$_h$*ow*$_d$*o*$_y$*ou/*

Random Forests:
https://bradleyboehmke.github.io/HOML/random-forest.html
https://youtu.be/J4Wdy0Wc$_x$*Q*
https : *//youtu.be/*6*EXPYzbfLCE*
Robnik−*ŠikonjaM.*(2004)*ImprovingRandomForests.Retrievedfromtheacademicdatabase SpringerLinkhttps* : *//doi.org/*10.1007/978 − 3 − 540 − 30115 − 8$_3$4

KNN:
https://bradleyboehmke.github.io/HOML/knn.html
https://youtu.be/HVXime0nQeI
https://youtu.be/UqYde-LULfs
https://youtu.be/MDniRwXizWo
https://discuss.analyticsvidhya.com/t/how-to-resolve-error-na-nan-inf-in-foreign-function-call-arg-6-in-knn/7280
https://www.edureka.co/blog/knn-algorithm-in-r/How

SVM:
https://youtu.be/efR1C6CvhmE
https://youtu.be/Toet3EiSFcM
https://youtu.be/Qc5IyLW$_h$*ns*