Nicolas Drizard, Eloi Zablocki

March, 1st 2015

PREDICTING SURVIVAL ON THE TITANIC WRECK

Introduction

In the report we present our approach to the *Kaggle* competition[1]. The goal is to predict the passengers that survived at the shipwreck of Titanic, given some caracteristics such as the genre, the age, etc.
First, we will explain the preprocessing task that has been made in order to clean, extract and create features. Then we will go through the classification task, with the algorithms used and tried.

Première partie

Data pre-processing and *Feature Engineering*

1 Data pre-processing

As given with the assignment, the data of *train.csv* and *test.csv* were composed of raws and there was some work to be done in order to clean the data. For example, there are missing values in the dataset, some features are not useful such as the passengerId, some features are string with very useful information such as the passenger name (containing the title and the family name).

Use of pandas To easily clean up the data set and create new features, we have used the python library *pandas*. This library allowed to visualize and explore data.

Data visualization One of the first tasks we made was data visualization. Indeed, in order to have some insights on the general pipeline, it is very convenient to visualize the data. For example, our first observation was the part of women who have survived compared to the part of the men who have survived. Here you can find some visualizations (Figure 1, Figure 2, Figure 3) made on Weka (the software used will be presented later in this report). The color blue represents the people who died and the red color the survivors.

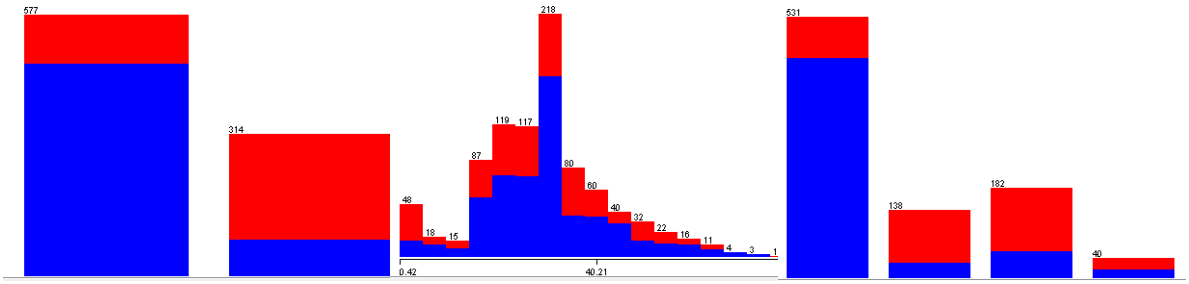

Figure 1 – Dead and surviving people repartition functions of the sex. Left : Male, Right : Female

Figure 2 – Dead and surviving people repartition functions of the age

Figure 3 – Dead and surviving people repartition functions of the title. From the left to the right : Mr, Mrs, Miss, Master

1. More infos to be found at `https://www.kaggle.com/c/titanic-gettingStarted/`

2 Feature Engineering

2.1 Creation of features

Feature extraction Some features such as the name and the cabin number contain precious information such as the title or the deck number. Nevertheless most of the algorithms would have troubles to use those features since they are strings. Thus, a "hand-made" extraction of the features have to be made.

Title extraction The title can give useful information on a person. A Lord, a Master (children), and a Miss have all three different chances to survive. With some appropriate function of *pandas* we have extracted the title from the name of the person. This gives us about 15 titles in the training set. We have aggregated the titles into four representative categories so that our algorithm is less likely to overfit the data. The categories are Mr., Miss., Mrs. and Master. To sum up, we have created a four-categories feature "Title" and removed the feature "Name"

Deck extraction Even though there are many missing values for the cabin number, those present give an information on which deck the person was. For example "C123" means that the cabin was on the deck "C". Again we have created a categorical feature "Deck" (values A, B, C, D, E, F, T, G or missing value) and removed the feature "Cabin".

New features Because we can always add or multiply two numbers to create a new number, here are the features that we have added to our dataset. Those features depends and are somehow redondant with other features.

Family size We have created a "Family size" feature. This is simply equal to the sum of "Sibsp" (number or siblings and spouse) and "Parch" (number of parents and children) and it represents the size of the family.

Fare per person We have created a "Fare per person" feature that is equal to the quotient of the "Fare" by the "Family size". The goal of this feature is to emphasize the difference between a single rich person and a poor person in a large family.

Age class interaction We have created a "Age*class" feature. This feature is simply the product of the age of the person with his class. The goal of this feature is complementary to the "Fare per person" feature : it emphasizes the difference between an old rich man and a young poor person.

2.2 Filling missing values

Most of the algorithm implemented in Weka accept missing values. Thus we have only filled the missing values that are really important. Given that the age of the passenger is very important and that we do not want weka to replace the age of children with the average age of all the passengers, we have computed an approached age which is the mean of the age of people who have the same title. For example the average age of the Master children is 4.57 whereas the average age of Reverend people is 43.17.
This modification was extremely helpful to improve our score.

Deuxième partie

Classification algorithms

1 Weka

We chose to use an already built library to classify our cleaned data. Weka is a library developed by the Machine Learning group form the Waikato University [2] in the Java programming language. Weka is a powerful tool for data analysis such as classification, regression, data vizualization or dimensionality reduction. In each of these machine learning fields, it has many implemented algorithms (50+ classification algorithms for example). Weka's input are **.arff** files which are very close to **.csv** files but with more information on the features.
The whole library accepts missing values and different kind of features (categorical, numerical, date...) yet some implemented algorithms may be more restrictive concerning the data type (some do not treat categorical features for example).
The preprocessing tasks explained in the part 1, gave us **.csv** files that we converted into **.arff** files, by hand. When Weka makes predictions on the test set, it returns an **.arff** file that we convert into **.csv** files, the accepted format for the *Kaggle* submissions.

2 Considered algorithms

The advantage of using the library Weka is that we can test our data set with many different algorithms. We first tried it with the basic algorithms such as oneR which is considering only the most weighted feature. We had a result of 78 % in the cross validation test. Then we considered different algorithms seen in the coursework such as AdaBoost and random Forest. But the best result were made with the Bagging algorithm which is presented in the next paragraph.

3 Chosen algorithm : Bagging

As Boosting, Bagging (standing for Bootstrap Aggregating) is an ensemble method. It creates separate samples of the training dataset and creates a classifier for each sample. The results of these multiple classifiers are then combined (such as averaged or majority voting). The trick is that each sample of the training dataset is different, giving each classifier that is trained, a subtly different focus and perspective on the problem. Averaging on the different basis enables to decrease the variance, which is important for our data set which is prone to over-fitting with its several features.

2. More infos to be found at `http://www.cs.waikato.ac.nz/ml/`

Troisième partie

Perspective

1 Difficulties

The most important lesson of this project is that building an acceptable classifier needs to process step by step. The first classification we tried was with only five features and had only result around 70 % in the cross validation test. Then we need to think about which feature was relevant to consider and how to format it. Some of our initial guesses were irrelevant and made us waste time, as using the logarithm of fare (the logarithm is often used with financial variables).

http ://trevorstephens.com/post/73461351896/titanic-getting-started-with-r-part-4-feature

http ://gertlowitz.blogspot.com.au/2013/06/where-am-i-up-to-with-titanic-competion.html

http ://www.joyofdata.de/blog/titanic-challenge-kaggle-svms-kernlab-decision-trees-party/

si t'as la foi, teste SVM en implementation python c'est performant sur notre dataset je crois.

2 Future

Using Weka enable us to test built-in Machine Learning algorithms and to focus on the feature engineering part but we can try to built our own Bagging algorithm to obtain better result. Or we could even test with another method, such as AdaBoost or neural networks.
Nevertheless, the feature engineering part is not over. We drop the family name where as we could cluster the data with it as a person is more likely to have survived is one of its relative has survived, this become all the more true with children from the same family for instance. Maybe some other interaction between numerical values can provide relevant results as we haven't tried all of them.

Code Explanation

faut faire le dossier qu'on va rendre. avec les bons programmes et les bons fichiers. pourquoi pas rendre des programmes qui marchent moins bien en disant que c'etait nos premieres pistes de recherche, pour souligner la progression de notre taf (quitte a les creer a partir de ce qui marche bien) ; expliquer dans cette partie le contenu du dossier qu'on rend

Conclusion

This project gives a good overview of the job of data engineers today. Our approach to deal with the subject is a common pipeline to handle many classification problems encountered in many companies or in the public research field. This project made us realize that all classification problems are different, that there are no best algorithms in general since it relies a lot on the type, the number and the range of the data. Moreover, not only don't we have a best classification algorithm in general, but also there are no best feature engineering method in general. Indeed, one always has to keep in mind the nature of the data he is dealing with. For example, thinking that the first letter of the cabin number might

be the deck and thus might be a classification criterion is something that cannot easily be done by a computer whereas human can make the feature engineering task given a specific context of the data.