Assignment 2

Julien Duquesne, Paul Couturier

December 2, 2018

1 Introduction

In order to reach the best score possible, we have decided to split the work. We worked separately during the first weeks to explore more ways to solve the problem. Finally we compared our results two weeks ago and created a program using our two experiences.

Name on kaggle : Les Torcheurs

2 Feature engineering

2.1 Treatment of the existing features

First of all, we treated the existing features in order to avoid blank or text, which can prevent the algorithm from working. Thus, we filled the blank of Age, Parch, Sibsp, Fare. To be sure to complete these series with all the information we had, we used the mean of the series. For categorical features such as Embarked or Sex, we created dummy variables to make them understandable by algorithms. The advantage of dummy variables over only transforming the variable into numbers is that it doesn't create order between categories.

Finally, for the feature Cabin, we filled the blank with 'T' because we had to keep a string nature, to use after the constrains notion to create Cabinlevel.

2.2 Creation of new features

To create these features we reflected on which information the algorithm couldn't capture. These information where mostly hidden in text data, such as Name or Ticket. Actually the algorithm couldn't capture relationships between passengers, which is a fundamental need to predict whether if a passenger survived or not. We will come back later to that point.

We created these new features :
- "Cabnum" : with intend to know whereas the passenger had a cabin or not. This information help us classify passengers between the richest and the poorest

- "Cabinlevel": then, we found that the letter on the cabin letter represents floors of the boat. So, the letter A represents the highest floor whereas the letter G represents the lowest. We created Cabinlevel feature using the letter present in the Cabin feature from 0 to 2. The 2 represented the higher floors and the 0 the lower floors. When we found a T that meant that we didn't have information and so we put 2 to put them on the top of the boat thinking that people will try to escape

the water flood and thus reach higher floors. But this feature doesn't increase any of the algorithms so we eliminated it. We think that this was due either to the lack of information in the cabin feature or either to the fact that passengers were not in their cabin during the beginning of the disaster.

- "Status" : We there deal with the name of passenger and categorize them thanks to their designation. We found there were several designation that underlie their social status of age. Once again, we created dummy variables for each of those status.

-"Individual Fare" : The fare presented in data is a global Fare for a ticket and not an individual Fare. We then wanted to divide this global Fare by the number of people owning this ticket. However we don't have all the people that own a same ticket in the dataset. Then we assumed people were more or less grouped by family and we then divided the Fare by the number of people in the family, which is accessible thanks to SibSp and Parch features.

"Ticket Grouped" so we can link people that have close tickets such as 17502 and 17501. Their fate may be linked during the sinking.

'Parch' and 'SibSp' don't make differences between being a parent or a child nor being a sibling or a spouse. We have to differentiate those so we created 4 new boolean features :
-"isChild" : 1 if the passenger is under k_1 years old and is Parch
-"isParent" : 1 if the passenger is over k_1 years old and is Parch
-"isSpouse": 1 if the passenger is a women and his status is not 'Miss' or if it is a men over k_2 years old, the whole if is SibSp
-"isSibling": complentary of isSpouse over SibSp

k_1 and k_2 are thresholds to be determined like others model parameters. We know this model isn't perfect at all since some adults can be Siblings but it's the best we have found so far.

2.3 Feature selection

We had to select which features were the most relevant regarding to the score we obtained. First we decided to apply an LDA. LDA seemed to be the most appropriate since we aimed at categorize passengers.We also tried a PCA algorithm but the results we obtained further were less satisfying. We think that this result is due to the fact that PCA doesn't use the class of the train set, because it's an unsupervised algorithm.

For every classifier, we decided which feature select or not by using a backward search algorithm over features to decide which features were the most efficient and which ones we should remove.

3 Model tuning and comparison

We have used a lots of different model in order to choose the best one.
Every model has been tested using a cross validation on the train set, which ensure us to not have overfitting.
To tune model parameters, we used repetitively the function Grid Search CV from scikit learn, that test many parameters simultaneously and return the best choice of parameters. We started by testing a large range of parameters and progressively arrived to shrink the range and find the

best parameters.

First we tried a tree decision, which is a quite simple model but easy to train with relatively few samples as we have here. Moreover, the female/male division seems to work pretty well.
We obtained a score of 0,69 on Kaggle.

To improve this score, we used a random forest classifier, which gave us a first cross-validated score of 0.794. After tuning the algorithm's parameters especially the $n_estimators$ and the $max_features$ parameters, we reached a score of 0.837 on the train set which is much better. However this algorithm scored only 0.78 on Kaggle.

Secondly, the Logistic Classifier seemed to be adapted in this context, we then decided to use it, which gave us a cross-validated score of 0.834 and a score of 0.78 once again on Kaggle after tuning parameters of both the classifier and LDA, with the Grid Search CV. Thus it wasn't better than Random Forest.

Finally we tried a SVM Classifier (especially the SCV of Scikit Learn), which gave us after tuning with the Grid Search CV, a cross validated score of 83.9 which is a bit better than with Random Forest and which doesn't over fit the data. We obtained a Kaggle score of 0.799 which was a real improvement.

3.1 A hand-made classifier

We had hard time improving our score on Kaggle, tuning our models wasn't sufficient and we realize we needed another point of view. We then thought about what information were present in data but that the algorithms couldn't capture. We found it was pretty difficult to link passenger to each others, whereas it is a key information. Indeed in the panic that followed the hit of the iceberg, it didn't matter if you had paid your ticket 9 or 20. What prevail above everything was bounds between persons, especially in families. We then had the idea to build our own classifier which would run in parallel with the ones that we had build before.
This classifier is not classic because it has to make bounds between the test and the train test. We then needed to think different. We explored data to understand some key bounds, and found interesting ones using the features created :

-If a child is dead, it is very much alike that his parents have died too
-If a parent has survived, it is very much alike that his children have survived too
-If a husband has survived, it is very much alike that his wife has survived
-If a wife has died, it is very much alike that her husband died too
-We can predict if a child has died or not regarding what his siblings became
We surely could have found more interesting relationships between passenger, such as nationality, but we ran out of time. We then created a classifier that go over the test set, establish bounds with the train test and then predict their fates for people who were concerned by these relationships. We replaced the related result predicted by the first classifier by the results predicted by the relationships-based classifier, that we had taken for reliable.
This second classifier improved by one point the score we had on cross validation score, and more importantly improved our Kaggle score by more than one point to reach 81.339%.