Titanic: Machine Learning from Disaster

Assignment 2: Foundations of Machine Learning Course - Fall 2018
Ecole CentraleSupelec, Paris

Ayush K. Rai, Louis DeVitry and Alami C. Mohamed
ayush.rai2512@student-cs.fr, louis.devitry@student-cs.fr, m.alamichehboune@student-cs.fr
Kaggle Team Name: Team_CS_AI

December 2, 2018

By turning this assignment, we declare that this is our own work.

The aim of the project is to predict whether or not a given passenger had died during the disaster or not.
For this, we will first perform a detailed analysis of the different features in order to combine them or create new ones to gather as much relevant information as possible before performing our prediction models.

1 Feature Engineering

1.1 Strategy

- **Missing values**: There are several variables (**Age, Cabin**) for which there are missing values. If not taken care of, this will have a detrimental effect on building predictive models. Some kind of models indeed need no missing values. However, we did not handle those missing values immediately as we needed some more information on the dataset to compute them more effectively.

- **Categorical variables**: Some predictors are not suited for direct exploitation. For instance, variables like **Name** need feature engineering/extracting. We will therefore proceed with common sense and data-driven choices to create relevant features from these raw data. The features concerned are: Name, Cabin and Ticket.

- **Coninuous variables**: Variables like **Fare** need some transformation before being used by a Machine Learning model. Specifically, the data need to be scaled (standard/min-max scaler). Furthermore, we will be looking for relevant data transformation (such as log transform) to further make the information digestable by ML models.

1.2 Encoding categorical features

Several features cannot be exploited right away. It is the case for the name, the cabin and the ticket of each passenger. We therefore had to retrieve informations and perform sound feature engineering of the data.

1.2.1 Sex

After plotting the survival probability depending on passenger's sex, it appeared women class passengers have more chance to survive than men.

1.2.2 Name

After a careful analysis of the Name features, we observed the occurrence of titles in lots of samples (Mr, Mrs, Lady, Dr, ...). These socio-economic markers probably have an importance. For instance, Lady and Countess might have a higher likelihood of getting on a lifeboat because of their status. The exhaustive list of found titles is: **Jonkheer, Ms, Mlle, Mme, Capt, Don, Major, Col, Sir, Dona, Lady, the Countess**. We therefore parsed the names and retrieved those keywords.

We noticed that most of the titles beside Mr, Mrs, Miss and Master have very low occurrences. As the most common way to deal with such categorical features is to create dummy variables, the resulting imbalanced features will undermine Machine Learning models abilities to learn. We therefore grouped some under broader categories:

- Ms, Mlle becomes Miss (Obvious)
- Mme becomes Mrs (Obvious)
- Jonkheer, Sir, Dona, Lady, the Countess become Nobles (Jonkheer denotes the lowest rank within the nobility in the Low countries)
- Capt, Don, Major, Col become Military (**Military individuals are likely to help others when the ship is sinking, i.e. sense of duty**)

After a careful analysis, we see the following:

- **Almost all nobles survived**
- **Military and Reverents did not survive for the majority and the same goes for the doctors**

Remark: These features are highly correlated with the Sex of the individual. 'Military', 'Reverents' and 'Doctors' are all males (except for one female doctor).

Rev and Nobles seem to be strong indicators of survival. We will consequently retain them in our dataset. However, despite the above average survival likelihood, military titles and Doctors do not seem discriminating enough. We therefore group the two of them under 'Officer' (by lack of a better name).

1.2.3 Cabin

The cabin number is a tricky predictor as it has many missing values and is not directly usable. Furthermore it needs feature engineering. We should note that there are some samples with more than one cabin reservation. **Intuitively, the residents of the cabin are likely to know each other, or at least the person who bought the ticket. It is also likely that people who know each other tend to stick together when in distress. We will take this into account in the following analysis.**

Deck: First letter of the string, it corresponds to the floor of the titanic, as per the plan below. We observe a few things:

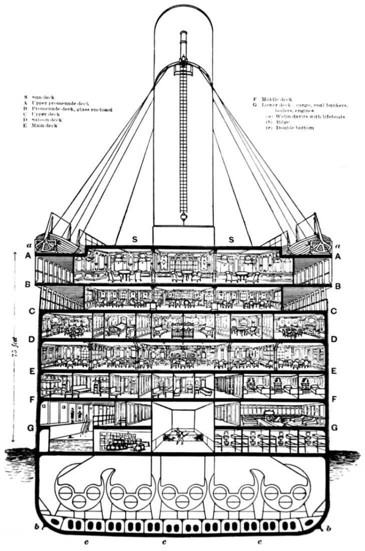
1. Some categories don't have enough information and should therefore be merged together

2. There are different likelihood of survival depending on the category.

We chose the groups based on the likelihood of survival

- **T, GF, U : Low (0 to 0.3 likelihood)**
- **A, C, G : Medium (between 0.3 and 0.7 likelihood)**
- **EF, F, D, E : High (superior to 0.7)**

Deck: According to the plan of the deck below, the number is distributed with regards to the boat length (low numbers at the front of the boat and high ones at the end). **We decided to consider the room number as a continuous variables. It will make our task easier for both the preprocessing and the the imputation.**



Number of cabins: A careful inspection of the number of cabins vs survival likelihood shows **that people with more than two cabin have a higher survival rate. We**

added a Boolean variable that indicates whether the reservation has two cabins or more.

1.2.4 Pclass

After plotting the survival probability for each class, **it appeared that first class passengers have more chances to survive than the others**.

1.3 Continuous Features

There are several continuous variables that need special attention. **These features are the Age, the Fare, SibSp and parch. In particular, we explored transformation and scaling.**

1.3.1 SibSp and Parch

Family: The two ordinal features Parch and SibSp both correspond to the number of family individuals. We therefore **merged** the two by summing them into a **new feature: 'Family'**.

1.3.2 Age

Age column contained 256 missing values in the whole dataset. Since there are sub-populations that have more chance to survive (children for example), it is preferable to keep the age feature and to impute the missing values.

Age distribution seems to be a tailed distribution, maybe a gaussian distribution. We noticed that age distributions are not the same in the survived and not survived sub-populations. Indeed, there is a peak corresponding to young passengers, that have survived. We also noticed that passengers between 60-80 have less survived.So, even if "Age" is not correlated with "Survived", we can see that there is age categories of passengers that of have more or less chance to survive. **It seems that very young passengers have more chance to survive.**

In the dataset we observe that age attribute for many passengers has missing values. An intelligent way of filling this missing value is to find the most likely group of that passenger based on other attributes like gender, title and class and then take the median age of that group. We choose median because it is a better measure of centrality when it comes to robustness against outliers. So we implemented a function that groups passengers that have the same features as mentioned above and computes the median for each group. Then it replaces each missing value with the corresponding mean

1.3.3 Fare

Fare distribution appeared to be very skewed. This can lead to overweight very high values in the model, even if it is scaled. **Therefore, we decided to transform Fare with the log function to reduce the skew.**

1.4 Outliers analysis

If not taken care of, outliers have detrimental effects on predictive models. We will use the local outlier factor, an algorithm for finding anomalous data points by measuring the local deviation of a given data point with respect to its neighbours.

The **Local Outlier Factor score quantifies the typical distance at which a point can be "reached" from its neighbors**. This technique works well when dealing with low dimensional data (such as our use case) because it does not suffer much from the curse of dimensionality.

After inspections of the scores density among training samples, we **remove the ten observations with the highest scores** (heavy tail of the LOF score distribution).

1.5 Feature selection

1.5.1 Why does feature selection matter?

This has three benefits. **First, we make our model more simple to interpret. Second, we can reduce the variance of the model, and therefore we reduce the risk of overfitting (especially with the small dimensions of our problem). Finally, we can reduce the computational cost (and time) of training a model. The process of identifying only the most relevant features is called feature selection.**

1.5.2 Feature importance and Random Forests

Random Forests are often used for feature selection in a data science workflow. **The reason is because the tree-based strategies used by random forests naturally rank by how well they improve the purity of the node.**

1.6 Normalization of Continuous Features

In order to normalize the continuous features, we used Scikit's Min-Max Scalar normalizer..

2 Model Comparison and Tuning

2.1 Setup

- We began the process of model evaluation by creating a **new training dataset with 70%** of the data points (in original training dataset) and a **validation dataset with 30%** of the remaining data points (in the original training dataset).
- As the rules of the competition clearly specify that **Accuracy** is the sole criteria for model evaluation therefore we decided to examine the performance the classifier only based on it and ignore other evaluation metrics like F1 Score and Area under ROC curve.
- We decided to use various classifiers like Logistic Regression Classifiers, Random Forest Classifier, Support Vector Classifier, Gradient Boosting Ensemble Classifier (Both from Scikit and XgBoost) and Adaboost Classifier.
- Since the training dataset is really small and therefore this problem is not a good candidate for applying multi-layer perceptron or neural networks.

2.2 Validation

We trained the classifiers mentioned in the **Setup** section on the **New Training Dataset** and measured their accuracy on **Validation Dataset**. The results are expressed in the following table:

Machine_Learning_Methods	**Accuracy**
Logistic Regression Classifier	0.7894
Random Forest Classifier	0.8458
Support Vector Classifier	0.8329
Gradient Boosting Classifier (Scikit)	0.8345
Gradient Boosting Classifier (XgBoost)	0.8646
Adaboost Classifier	0.8157

Table 1: Results of Various Machine Learning Models on Validation Dataset

In our multiple attempts in the **model evaluation phase** we observed that ensemble methods perform much better than the other methods like Logistic Regression and Support Vector Classifiers and especially the **Gradient Boosting Classifier from XgBoost Library was consistently producing good results** therefore we wanted to perform **Hyperparameter selection particularly on it**.

2.3 Hyperparameter Selection

- Usually KFold Cross-Validation technique is a common technique for model tuning and estimating hyperparameters of the model but **the drawback of KFold Cross**

Validation is that it doesn't preserve the percentage of sample of each class while creating the splits. To handle this, we used **Stratified KFold cross validation technique which ensures that every split of the training dataset. doesn't suffer from class imbalance issue**.

- In order to better tune the model, we used the **entire training dataset** (New Training Dataset + Validation Dataset) for this step.
- We also used **exhaustive grid search** from Scikit's hyperparameter tuning module to find the best parameters for Gradient Boosting Classifier (from XgBoost)
- After a series of experimentation, we were able to get the **best parameters** for our Gradient Boosting Classifier.

The best parameters are: XGBClassifier(base$_{s}core = 0.5, booster =' gbtree', colsample_{b}ylevel = 1, colsample_{b}ytree = 0.8, gamma = 0.5, learning_{r}ate = 0.02, max_{d}elta_{s}tep = 0, max_{d}epth = 5, min_{c}hild_{w}eight = 1, missing = None, n_{e}stimators = 600, n_{j}obs = 1, nthread = 1, objective =' binary : logistic', random_{s}tate = 0, reg_{a}lpha = 0, reg_{l}ambda = 1, scale_{p}os_{w}eight = 1, seed = None, silent = True, subsample = 0.6)$

3 Conclusions

Based on this model, we were able to achieve a public score of **0.73 on Kaggle Platform.** However, on one of our submissions, we reached our best score of 0.78