INF582 - Data Science and Mining
École Polytechnique

Final Project

Predicting Survival on the Titanic Wreck

Jean-Baptiste Bordes, Balázs Kégl, Fragkiskos Malliaros and Michalis Vazirgiannis

February 2015

1 Description of the Project

The goal of the project is to apply data mining and machine learning techniques in order to predict the passengers that survived at the shipwreck of Titanic. Titanic was a British passenger liner that sank in the North Atlantic Ocean in the early morning of 15 April 1912 after colliding with an iceberg during her maiden voyage from Southampton to New York City. The sinking resulted in the loss of more than 1500 passengers and crew making it one of the deadliest commercial peacetime maritime disasters in modern history. One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class. In this project, we ask you to complete the analysis of what sorts of people were likely to survive. The problem can be expressed as a two-class classification problem (survive or not) and the goal is to design a model that achieves high classification accuracy.

This project constitutes a competition on *Kaggle* (`www.kaggle.com/`), a platform that hosts predictive modeling and analytics competitions. As we will discuss later, the final evaluation of your model will be done using the Kaggle platform. More information about the competition can be found on the Kaggle website: `https://www.kaggle.com/c/titanic-gettingStarted`.

2 Dataset Description

Next we describe the structure of the train dataset (`train.csv`) that will be used for training your model. The size of the dataset is 891×12. Each of the 891 rows of the dataset corresponds to a passenger and the columns describe the following set of features:

- `PassengerId`: The unique identifier of each passenger.
- `Survived`: If the passenger survived or not (0=No; 1=Yes). This is also the class label that we want to predict.
- `Pclass`: Passenger Class in the ship (1=1st; 2=2nd; 3=3rd).
- `Name`: Name of the passenger.

- `Sex`: Sex of the passenger (male/female).
- `Age`: Age is in years; Fractional if Age less than one. If the age is estimated, it is in the form `xx.5`.
- `SibSp`: Number of siblings/spouses aboard.
- `Parch`: Number of parents/children aboard.
- `Ticket`: Ticket Number.
- `Fare`: Passenger fare.
- `Cabin`: Cabin number.
- `Embarked`: Port of embarkation (C=Cherbourg; Q=Queenstown; S=Southampton).

More information about the data can be found on: `www.kaggle.com/c/titanic-gettingStarted/data`. Figure 1 gives an example of the first five rows of the dataset.

```
In [13]: train_data = pd.read_csv('train.csv', header=0) # Load the data
         print "Size of the dataset: ", train_data.shape
         print train_data.head()
```

```
Size of the dataset:  (891, 12)
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex  Age  SibSp  \
0                            Braund, Mr. Owen Harris    male   22      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female   38      1
2                             Heikkinen, Miss. Laina  female   26      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female   35      1
4                           Allen, Mr. William Henry    male   35      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

Figure 1: First five rows of the dataset (using the `pandas` module).

As you can observe, the dataset has some missing values for some attributes (`NaN`). In the preprocessing task, you should take care of these cases. In the case of features that take numerical values, one approach could be to replace the missing values with the mean value of this feature. Some other features may not be useful at the classification task. For example, the `PassengerId` feature is just an identifier that takes values from 1 to 891, and thus it can be ignored. It would be helpful to explore the dataset and try to deal with such cases. Additionally, some of the features take values that correspond to a string (e.g., the `sex` feature takes values `male` and `female`). In such cases, we can replace the values with numeric ones (e.g., `male`: 0; `female`: 1). Lastly, normalizing the data could also be useful in the classification task.

As part of the preprocessing step, you can also apply feature selection techniques to keep a subset with the most informative features or dimensionality reduction methods (e.g., Linear Discriminat Analysis). It is also possible to create new features that do not exist in the dataset, but can be useful in the classification task. For example, although the name of each passenger may not be that useful in classification, the additional information contained in the `Name` feature regarding the title-status of the passengers (Mr., Mrs., Miss., Master., Mlle., Dr., ...), may be more relevant. Thus, you can create a new feature (i.e., add a new column to the data matrix) to represent this information (this is known as feature engineering or generation).

3 Summary of the Pipeline

The pipeline that will be followed in the project is similar to the one followed in the labs. Next we briefly describe each part of the pipeline.

- *Data pre-processing*: After loading the data, a preprocessing task should be done to transform the data into an appropriate format. In the previous section, we described some of these points.
- *Feature engineering - Dimensionality reduction*: The next step involves the feature engineering task, i.e., how to select a subset of the features that will be used in the learning task (feature selection) or how to create new features from the already existing ones (see also previous section). Moreover, it is possible to apply dimensionality reduction tecniques in order to improve the performance of the algorithms.
- *Learning algorithm*: The next step of the pipeline involves the selection of the appropriate learning (i.e., classification) algorithm for the problem. Here, you can test the performance of different algorithms and choose the best one. Additionally, you can follow an ensemble learning approach, combining many classification algorithms.
- *Evaluation*: In Section 4 we describe in detail how the evaluating will be performed.

4 Evaluation

You will build your classification model based on the training data contained in the `train.csv` file. To do this, you will apply *cross-validation* techniques[1]. The goal of cross-validation is to define a dataset to test the model in the training phase, in order to limit problems like overfitting and have an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, like the test dataset that will be used to assess your model).

In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k - 1$ subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation (i.e., test) data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation (average accuracy of the model).

Of course, having a good model that achieves good accuracy under cross validation does not guarantee that the same accuracy will be also achieved for the test data. Thus, the final evaluation of your model will be done on the test dataset contained in the `test.csv` file. So, after having a model that performs well under cross-validation, you should train the model using the whole training dataset (i.e., all the instances of the `train.csv` file) and test it on the test dataset as described below.

How to evaluate your model on the test dataset?

For the test data (`test.csv`) that contains 418 instances (passengers), we do not have information about the class labels (survived or not), and thus the final assessment will be done in the Kaggle platform. Note that, the same preprocessing tasks that were applied on the training data should also be applied on the test data. The evaluation process can be summarized as follows:

1. Run your model on the test data (`test.csv`).

[1] Wikipedia's lemma for *Cross-validation*: `http://en.wikipedia.org/wiki/Cross-validation_(statistics)`.

2. Get the predicted class labels (survived or not) for each instance of the test dataset.

3. Create a new file, called `predicted_class.csv`, that contains the predicted class label for each instance. The file must have exactly 2 comma separated columns: `PassengerId` (which can be sorted in any order), and `Survived` which contains your binary predictions: 1 for survived, 0 for did not.

4. Create an account on Kaggle (the same for each member of your team - see also Section 6) and make a new submission by simply uploading the `predicted_class.csv`. Then, Kaggle will evaluate your predictions and the evaluation score as well as your position (with respect to the rest users) will appear in the Leaderboard. Note that, you can submit up to 10 entries per day. Your final score will be the best one that you have achieved.

5 Useful Python Libraries

In this section, we briefly discuss some tools that can be useful in the project and you are encouraged to use.

- For the preprocessing task which also involves some initial data exploration, you may use the `pandas` Python library for data analysis[2].
- A very powerful machine learning library in Python is `scikit-learn`[3]. It can be used in the preprocessing step (e.g., for feature selection) and in the classification task (a plethora of classification algorithms have been implemented in `scikit-learn`.) Recall that we have already used the decision tree classifier of `scikit-learn` in Lab 6 (Ensemble methods).
- Finally, you are always encouraged to develop your own implementation of learning algorithms or use the ones developed in the labs.

6 Details about the Submission of the Project

Your final evaluation for the project will be based on both the accuracy that will be achieved on the test dataset (as reported by the Kaggle platform), as well as on your total approach to the problem. Regarding the evaluation in the Kaggle platform, please follow the details descibed in Section 4 and on the website of the competition[4].

- Form a team of 2-3 people. Each team will create one Kaggle account with the following name: `INF582_xx`, where `xx` should be 01, 02, 03, etc. (e.g., the first team that will create an account, will talke the `INF582_01` identifier).
- Follow the instructions provided in Kaggle platform. Each team can do a specific number of new submissions per day to Kaggle.

As part of the project, you have to submit the following:

[2] `http://pandas.pydata.org/`.
[3] `http://scikit-learn.org/`
[4] `https://www.kaggle.com/c/titanic-gettingStarted/details/submission-instructions`

- A 2-3 pages report, in which you should describe the approach and the methods that you used in the project. Since this is a real data science task, we are interested to know how you dealt with each part of the pipeline, e.g., if you have created new features and why, which classification algorithms did you use and why, their performance (accuracy and training time), approaches that finally didn't work but is interesting to present them, and in general, whatever you think that is interesting to report). Also, in the report, please provide the names of the team members and the identifier of your team (e.g., `INF582_10`).
- A directory with the code of your implementation and the `predicted_class.csv` file with the predicted class of the test data (the one that corresponds to your final submission to Kaggle).
- Create a `.zip` file with name `lastname1_lastname2.zip` (the lastnames of all team members), containing the code and the report and submit it to moodle.
- **Deadline: Sunday, March 1, 23:55.**