MACHINE LEARNING
M.SC. IN DATA SCIENCES AND BUSINESS ANALYTICS
M.SC. IN ARTIFICIAL INTELLIGENCE
CENTRALESUPÉLEC

Assignment 1

Instructor: Fragkiskos Malliaros

Due: **December 2, 2018 at 23:00**

1 Description of the Project

The goal of the project is to apply 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 $1,500$ 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 (survived 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`.

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: `https://www.kaggle.com/c/titanic/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. The pipeline is contained in the `main.py` python script.

- *Data preprocessing*: 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 the tasks that need to be performed.
- *Feature engineering and 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 techniques in order to improve the performance of the algorithms. For example, in the `classify.py` file, you can see that by retaining only the sex of the passengers, we can achieve fairly good performance ($\sim 78\%$).
- *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 (e.g., logistic regression, SVMs, decision trees, neural networks). 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 such as overfitting, and to get insights about how the model will generalize to an unseen dataset. As we have already seen in the class, a commonly used approach is the one of k-fold cross-validation.

Of course, having a good model that achieves good accuracy under cross validation does not guarantee that the same accuracy will also be 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. 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 have been 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`).
2. Get the predicted class labels (survived or not) for each instance of the test dataset.

[1] Cross-validation in *scikit-learn*: `http://scikit-learn.org/stable/modules/cross_validation.html`.

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) 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`).

6 Grading

Grading will be on 100 points total. Each team should deliver:

- A submission on the Kaggle competition webpage. **30 points** will be allocated based on raw performance only, provided that the results are reproducible. That is, using only your code and the data provided on the competition page, we should be able to train your final model and use it to generate the predictions you submitted for scoring.
- A 3-pages report (see details below). Please ensure that both your real names and the name of your Kaggle team appear on the report.

The 3-page report should include the following sections (in this specific order):

- *Section 1: Feature engineering* **[40 points]**. Regardless of the performance achieved, we will reward the research efforts and creativity put into the feature engineering step (e.g., creation of new features). You are expected to:

  1. Explain the motivation and intuition behind each feature. How did you come up with new feature? What is it intended to capture? Did you discard other features?
  2. Rigorously report your experiments about the impact of various combinations of features on predictive performance, and, depending on the classifier, how you tackled the task of feature selection.

[2] `http://pandas.pydata.org/`.
[3] `http://scikit-learn.org/stable/`

- *Section 2: Model tuning and comparison* **[20 points].** You are expected to:

    1. Compare multiple classifiers (e.g., SVMs, Random Forest, Boosting, Logistic regression, Nearest neighbors).
    2. For each classifier, explain the procedure that was followed to tackle parameter tuning and prevent overfitting.
    3. Report the cross-validated performance (on the training data) of the models you have explored, as well as the score on the test set (given by Kaggle).
    4. Discuss about any additional models that you have tested but did not perform well.

Report and code **completeness, organization and readability will be worth 10 points**. Best submissions will (i) clearly deliver the solution, providing detailed explanations of each step, and (ii) provide clear, well organized and commented code.

7 How to Submit

Please complete the second assignment in groups of **3-4** students (preferably, the same teams as in the project of the course). No late assignments will be accepted.

1. **Kaggle submission:** submission of your solution in the kaggle platform (team submission – pick also a name for your team).
2. **Report:** *typeset* your report (**PDF** file only). The submission of the report (max **3 pages**) should be made on **gradescope** as group submission (Assignment 2; Entry Code: 9PW55K).
3. **Code:** prepare a .zip file (`code_name_of_your_team.zip`) containing the code that is needed reproduce your submission. The file should be send by email to: *fragkiskos.me@gmail.com*, subject: "ML-DSBA-AI - Assignment 2 Code - Name of Your Team".