



Escola de Camins
Escola Tècnica Superior d'Enginyeria de Camins, Canals i Ports
UPC BARCELONATECH

Models for Decision Making and Optimization in Engineering

Project Work: Titanic sinking

Report

Students: David Grandal Rejo
Guillermo Lahuerta Piñeiro

Place and date: Barcelona, May 25th, 2018

Content

1. Introduction.....	3
2. Kaggle	4
3. Objective.....	5
4. Hypothesis.....	5
5. Procedure.....	6
6. Data	7
6.1 Data preparation	8
6.1.1 Age.....	8
6.1.2 Fare.....	9
6.1.3 Cabin.....	9
6.1.4 Embarked	10
6.1.5 Relatives	10
6.2 Data analysis.....	11
7. Principal Component Analysis	12
8. Artificial Neural Network.....	15
9. Prediction.....	17
10. Conclusions	18
11. References	20

1. Introduction

On April 15th, 1912, during its **inaugural voyage** from Southampton to New York, the Titanic sank after **colliding with an iceberg** killing 1502 out of the 2224 passengers and crew.

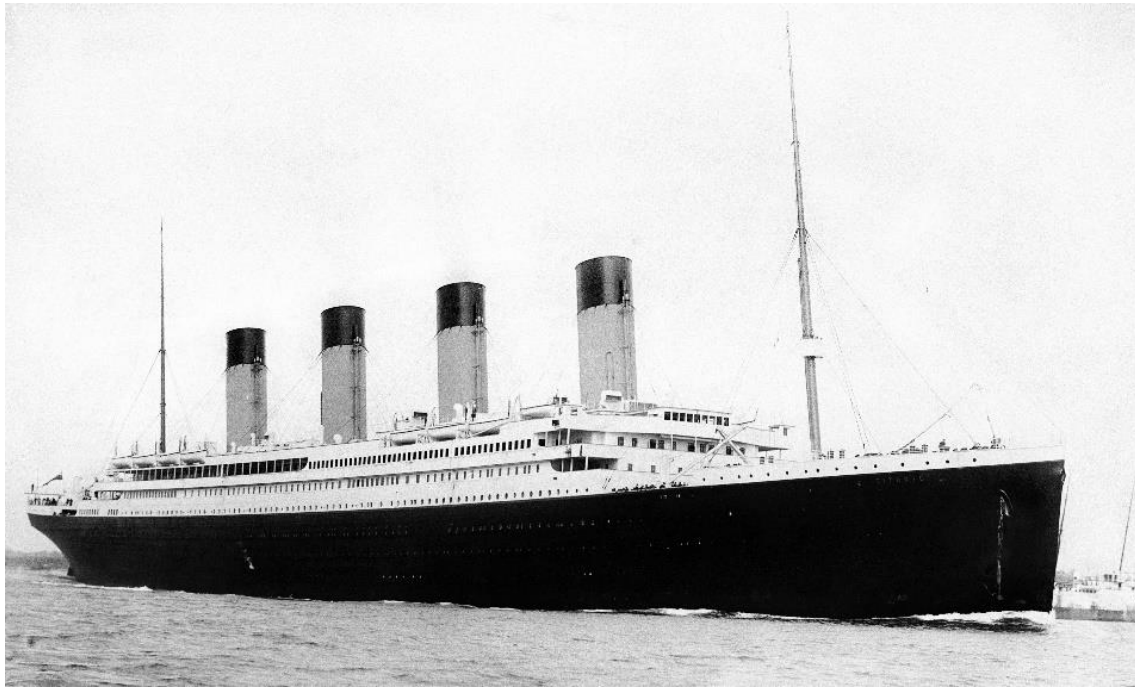


Figure 1. *Picture of Titanic leaving Southampton [1].*

The main reason leading to this high amount of deaths was the **lack of lifeboats** for all the passengers and the crew.



Figure 2. *Titanic inaugural route [2].*

2. Kaggle

Kaggle is one of the most popular platforms for **Data Science competitions**. Among these competitions, one of the most popular is the so-called: “**Titanic: Machine Learning from the disaster**”.

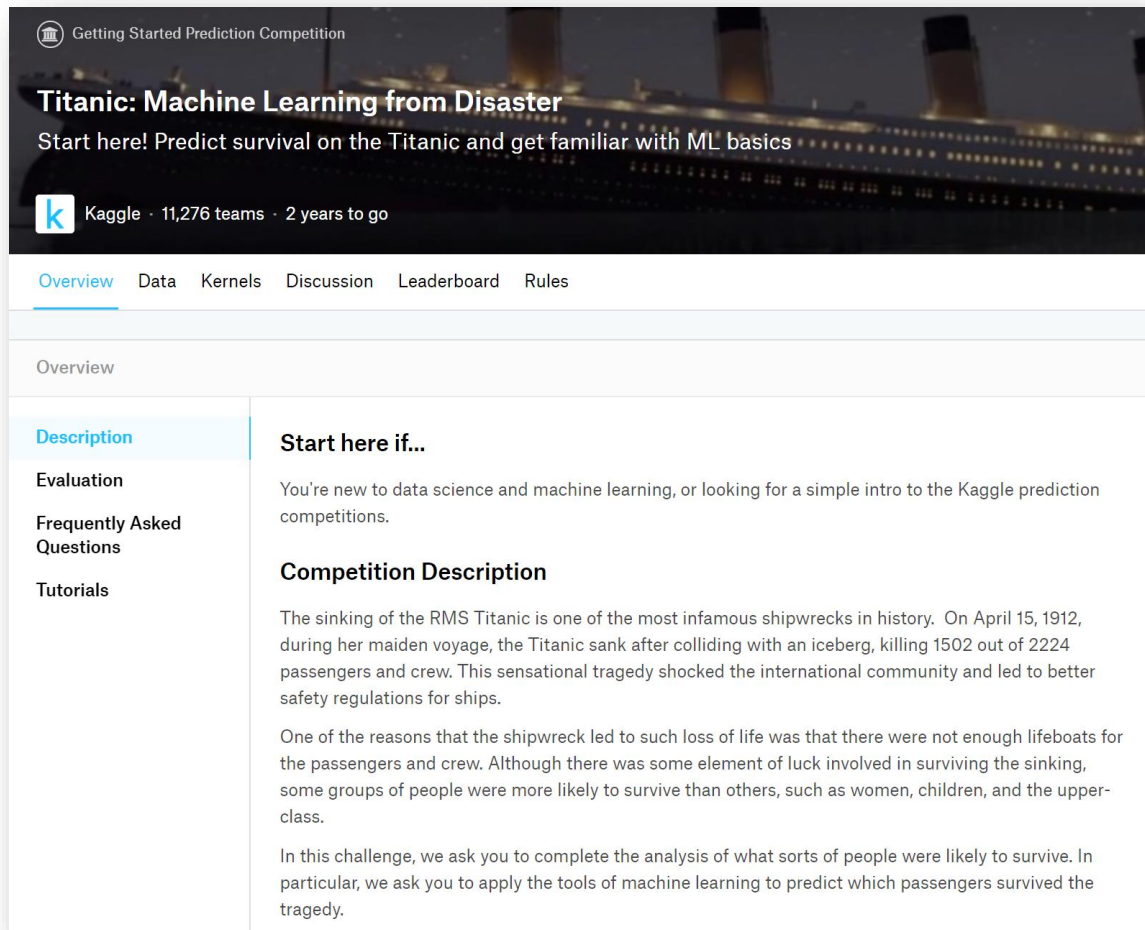


Figure 3. Main page of Kaggle [3].

In this challenge, participants must apply tools of **machine learning** to **predict which passengers survived** the tragedy.

At the moment of writing this report, there are **11.276 teams** participating in this challenge, with an average score (percentage of accuracy in the prediction submitted) of **77%**.

3. Objective

The objective in this project is to **predict the fate of the passengers** aboard the Titanic.

The **machine learning tools** chosen to create the **predictive model** will be two of the tools taught in the course “**Models for Decision Making and Optimization in Engineering**” during the academic course 2017/18 at the *Universitat Politècnica de Catalunya* in the framework of the *Máster de Ingeniería de Caminos*:

1. **Principal Component Analysis.**
2. **Artificial Neural Networks.**

Considering the chaos in the final hours of the Titanic, it is assumed that the **accuracy of the model will be limited**, and the average accuracy of the competition (77%) will be used as a reference.

4. Hypothesis

The main hypothesis/assumptions about the relationship among the variables provided in the data are the following ones:

- **Every passenger** in the Titanic **paid for his ticket**, therefore, if a 0 appears in the ticket variable, this is treated as a missing value.
- **The price** that each passenger paid (variable Fare) **is correlated with his class** (1st, 2nd or 3rd).
- The **cabins** of the 3 classes **were separated in different decks**.
- All the passengers with the **same ticket code** were travelling together and belonged to the **same family**.

These hypotheses are used during the **preparation of the data**, in order to fill the missing values.

With respect to the survive rate, it is expected that **children, women and 1st class passengers** are more prone to survive.

5. Procedure

The procedure followed in this project is the following:

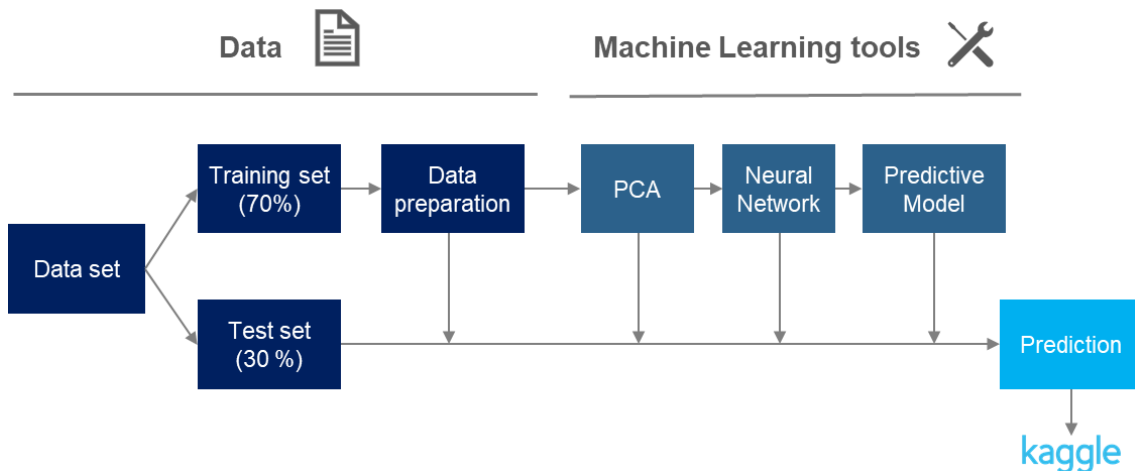


Figure 4. Procedure followed during the project.

- The **data set** provided by Kaggle will be used as the starting point. This data set consists of **1309 observations** (passengers) and **12 variables** for each observation.
- This data set is already divided by Kaggle into a **training set** (with the **70%** of the data) and the **test set** (with the other **30%**).
- The training set will be used to perform the **data preparation**, the **PCA** and the **Neural Network** in order to create the **predictive model**.
- Once the predictive model is created, all the steps defined before will be applied to the test set and **a prediction will be obtained**.
- This prediction will be **submitted to Kaggle** where a score will be assigned to the prediction. **The score is equal to the accuracy** obtained in the test set. As said before, the average accuracy of the competition is around 77%.

Note that in the validation of the Neural Network, the original training set will be subdivided into another training and test sets to calibrate the network.

6. Data

The data provided by Kaggle consists of two CSV files, one for the training set and another for the test set.

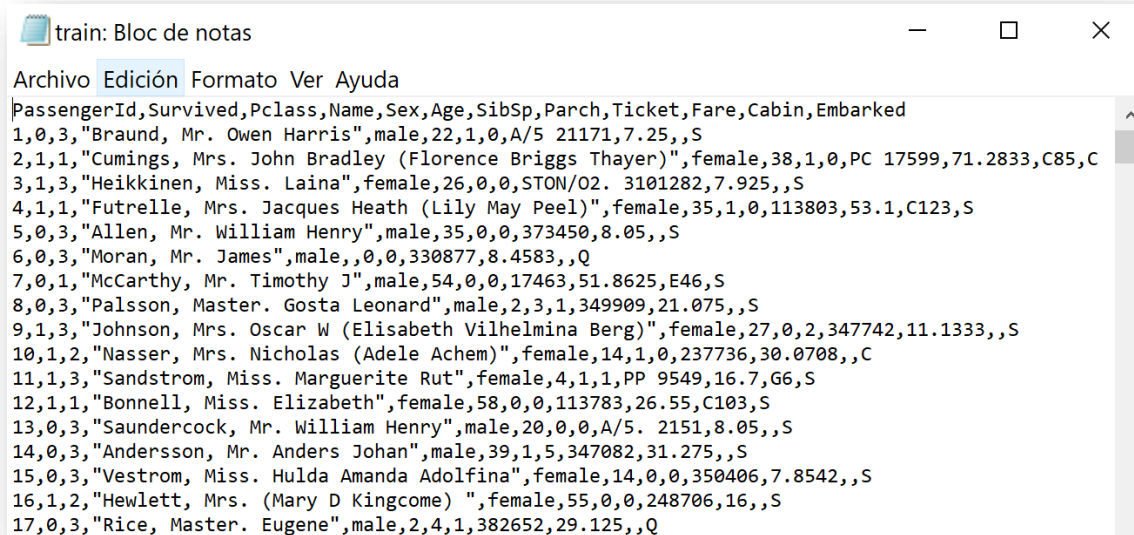


Figure 5. CSV file with the training data.

The variables included for each passenger are the following ones:

- **PassengerId.** Indicates the passenger number (1, 2, 3, ..., 1309).
- **Survival.** Indicates if the passenger survived or not (0 = No; 1 = Yes). This variable only applies to the Training set.
- **Pclass.** Indicates the passenger class (1 = 1st; 2 = 2nd; 3 = 3rd).
- **Name.** Indicates the full name of the passenger.
- **Sex.** Indicates the gender (Female/Male).
- **Age.** Indicates the age.
- **Sibsp.** Indicates the number of siblings/spouses aboard.
- **Parch.** Indicates the number of parents/children aboard.
- **Ticket.** Indicates the ticket code (for example PP9549)
- **Fare.** Indicates the cost of the ticket.
- **Cabin.** Indicates the deck and the number of the room.
- **Embarked.** Indicates at which Port the passenger embarked (C = Cherbourg; Q = Queenstown; S = Southampton).

6.1 Data preparation

The first task consists on preparing the data. To do so, we need to know which the missing values are to act on them. The next plot shows in red those missing values:

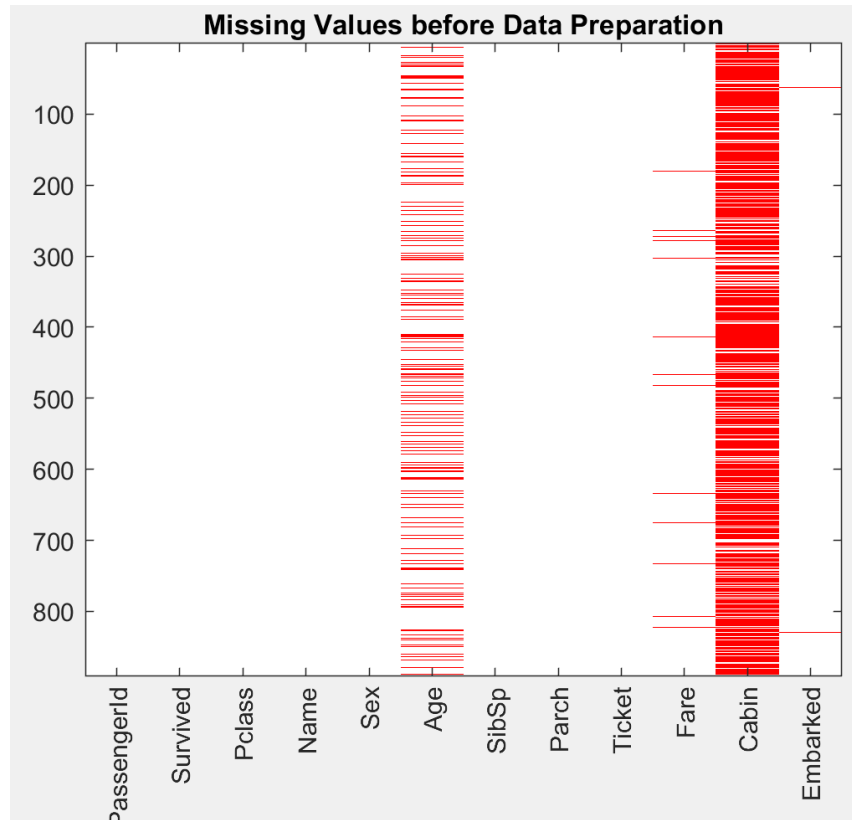


Figure 6. *Missing values.*

The following missing data is detected for 4 variables:

- Age: 19 % of missing values.
- Fare: 5 % of missing values.
- Cabin: 77 % of missing values.
- Embarked: 1 % of missing values.

6.1.1 Age

To fill the missing values of the age, the regular approach is to use the **average age of all the passengers**. However, we have tried to use a more accurate method.

At the beginning, the variable NAME was discarded because no useful information could be extracted from it. However, paying more attention to this variable, it is observed that we can extract the **title of each passenger**. Four examples are presented below:

- “Giglio, **Mr.** Victor”.
- “Hakkarainen, **Mrs.** Pekka Pietari”.
- “Ford, **Miss.** Robina Maggie”.
- “Panula, **Master.** Eino Viljami”.
- “Stahelin-Maeglin, **Dr.** Max”

This information is **stored in a new variable called TITLE that is used as a proxy variable** of the age. Then, the average age of each TITLE group is assigned to the passengers with that title that have no age.

6.1.2 Fare

For the FARE variable (amount of money that each passenger has payed), it is assumed that **every passenger in the Titanic payed** for his ticket, therefore, if a 0 appears in the ticket variable, this is treated as a missing value.

Then, the **average fare of each passenger class** (1st, 2nd, and 3rd class) is used, assuming that the price that each passenger payed is correlated with his class.

6.1.3 Cabin

Variable CABIN is the most complicated one:

- There are a lot of missing values (77%).
- Cannot assign an average room to all the missing values.

Therefore, first of all, it is assumed that the **room number doesn't play a role**, only the deck in which the cabin was located matters. Then, the **average deck of each passenger class** is assigned to the missing values, assuming that classes were separated in different decks.

6.1.4 Embarked

Finally, the EMBARKED variable is filled using the **most common Port of embarkation** (Southampton). Since there are only 2 missing values for this variable, there won't be any effect if we chose any other approach.

6.1.5 Relatives

No distinction is made between the variables SIBSP and PARCH, and a new variable RELATIVES is created. This new variable captures the **number of relatives aboard** for each passenger.

Note that there is no way to know if, for instance, when the PARCH variable (accounting for the number of parents/children aboard) is equal to 0, is because it was actually equal to 0, or it is just a missing value. To solve this, the TICKET variable is used, assuming that all the passengers with the same ticket code were travelling together and belonged to the same family.

Once all the variables are filled, the next figure shows that there are no missing values:

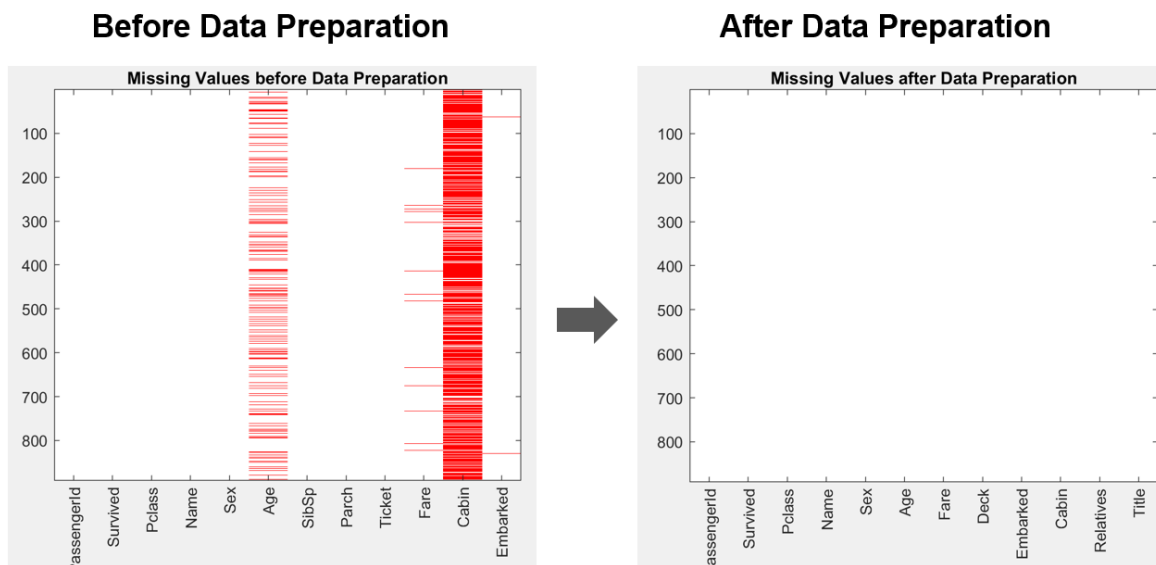


Figure 7. Missing values before and after data preparation.

6.2 Data analysis

At this point, it is worth to plot the data with respect to the main variables to see if one of the main hypothesis is correct: children, women and 1st class passengers were more prone to survive.

In Figure 8, the distribution of the passengers with respect to the **age**, the **gender** and the **class** is presented with **absolute values**.

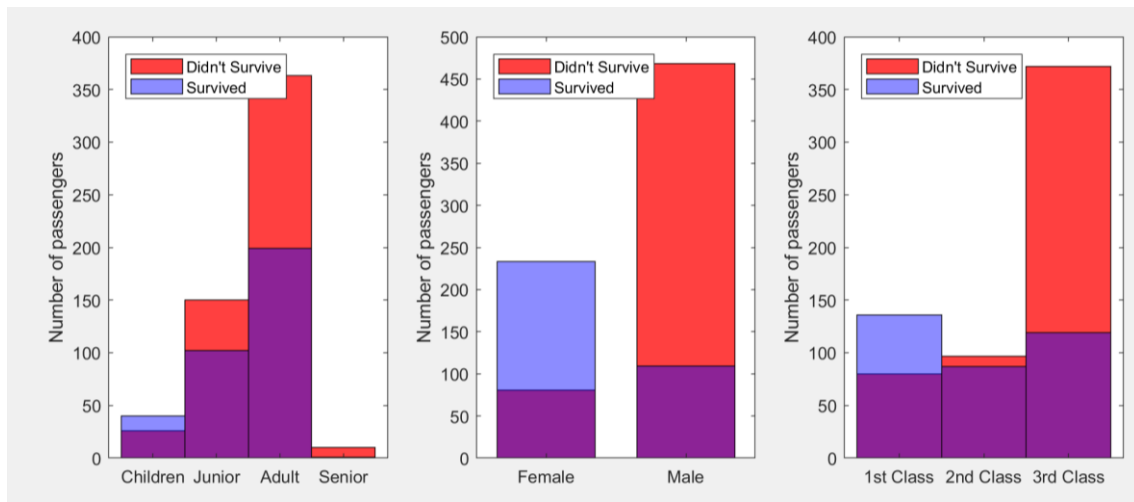


Figure 8. *Distribution of the passengers' fate (absolute values).*

On the other hand, in Figure 9, the same distribution is presented with **relative values** and **only for the survivors**.

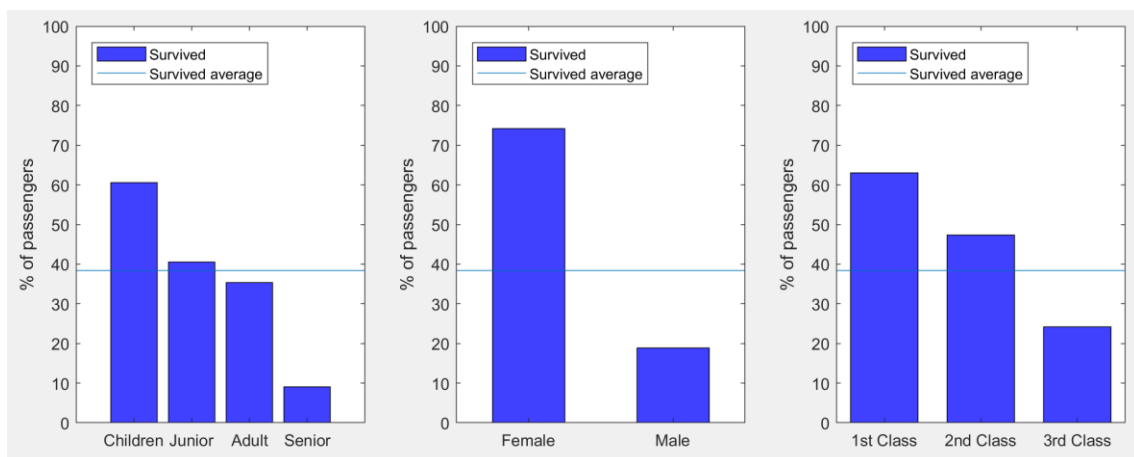


Figure 9. *Distribution of the passengers' fate (relative values).*

These statistics confirm the hypothesis that **children, women and 1st class passengers were more prone to survive.**

7. Principal Component Analysis

As presented before, the original data set had 12 dimensions. However, **after data preparation**, these **12 dimensions are down to 8**: AGE, FARE, TITLE, RELATIVES, SEX, EMBARKED, DECK and PCLASS.

Nevertheless, it is assumed that most of the variables have strong correlations. Thus, a Principal Component Analysis is carried out to **reduce these variables** to a new set of components such that some of them **collect most of the variance**. The steps followed to perform the PCA are:

1. **Scale** the data. This is an important step since, for instance, the fare is 2 orders of magnitude higher than most of the variables. To scale the data, the mean normalization is used:

$$x' = \frac{x - \text{mean}(x)}{\text{max}(x) - \text{min}(x)}$$

Where x is the original value and x' is the normalized one.

2. **Center** the data. This is done by subtracting the mean to each value.
3. Compute the **covariance matrix** of the 8 variables.
4. Get **eigenvalues and eigenvectors**. The calculation of eigenvalues and eigenvectors of the covariance matrix can be very ill-conditioned and can be very expensive computationally. Instead, what is done in practice is to compute the Singular Value Decomposition (SVD). In this project, both methods were used, performing identical results.
5. Compute the **transformation matrix** to be able to **project the original data into the new space** defined by the Principal Components.

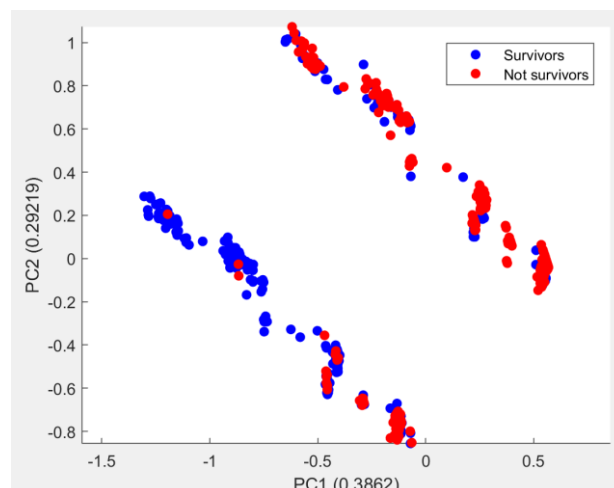


Figure 10. Projection of the data onto the space defined by the first 2 PC.

In Figure 10, it is clear that **there are two main clusters**; the red one collecting the non-survivors and the blue one collecting the survivors. However, there is a little noise, especially in the bottom of the plot, where we can see a little group of 'reds'. These passengers are basically women in the 1st class; women that were supposed to live, but for some reason, they didn't.

Figure 11 shows the same plot but with the 3 PC:

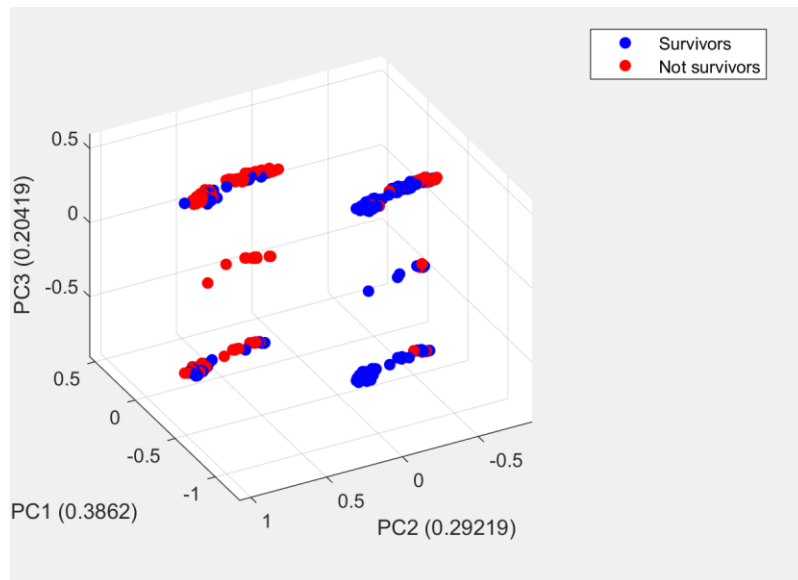


Figure 11. *Projection of the data onto the space defined by the first 3 PC.*

With respect to the variance collected by each PC, the next figure shows the distribution:

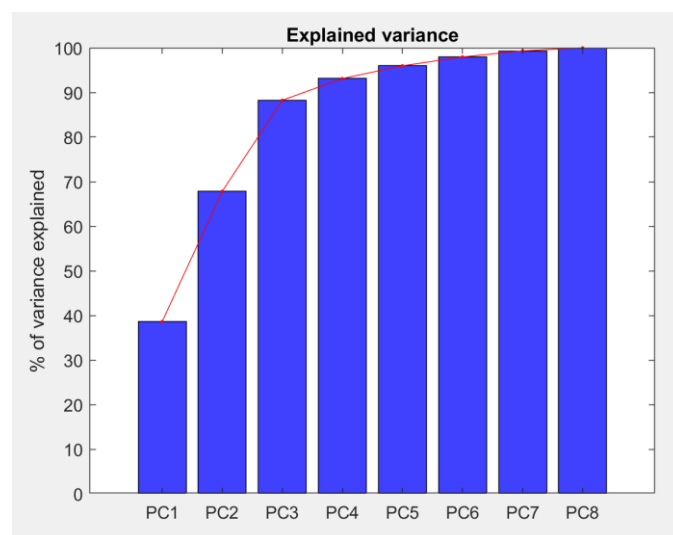


Figure 12. *Explained accumulated variance for each PC.*

It is shown that **the 3 first PC collect almost the 90% of the variance**. Therefore, **these 3 PC will be kept to perform the Neural Network**.

Finally, in the next figure, the vectors of the original variables are projected onto the new space defined by the 2 PCs:

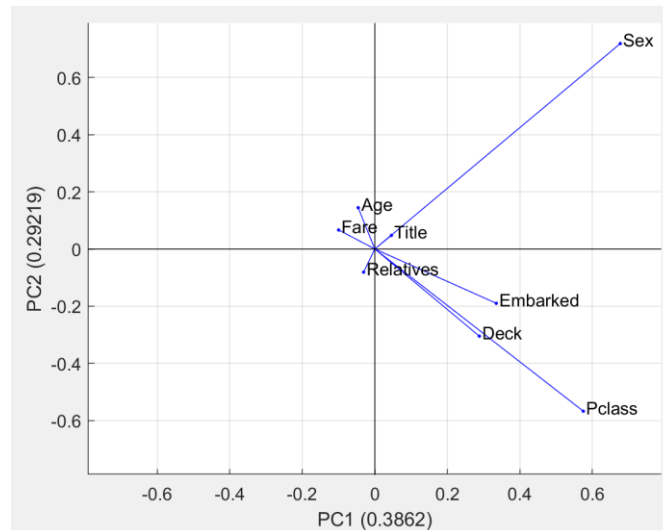


Figure 13. *Original variables projected onto the new space defined by the 2 PCs.*

It is shown that **SEX, TITLE, DECK AND PCLASS, contribute 50-50% to both Principal Components** since these variables have a perfect slope of 45°. On the other hand, variables like EMBARKED are more correlated with the 1st PC.

Once the reduction of the dimensions (from 8 to 3) is acquired, we can move on to the Neural Network model to create the predictive model.

8. Artificial Neural Network

The ANN is used to classify the Survivors and the Non-survivors. The **input** matrix of the model is the **projection of the 891 passengers** onto the space defined by the first 3 principal components.

After calibrating the network to obtain the best accuracy in both the training and the test set, the main parameters defining the final Neural Network are:

- Pattern recognition: patternnet
- Training function: 'trainlm'
- Hidden layers: 1
- Neurons: 2
- Training ratio: 0,90
- Validation ratio: 0,10
- Training Set accuracy: 81,9 %

The wizard selected to perform this analysis was the *Pattern recognition and classification neural network* from Matlab with a Levenberg-Marquardt backpropagation optimization function (the training function used to optimize weights and biases of the network).

Talking about the structure, it was chosen two layers (one hidden layer) with two neurons each one. Nevertheless, some other configuration structures were tested, increasing the complexity of the network.

However, when complexity grows, it can lead to overfit the problem. Looking at this specific problem, there are clearly two clusters of data. As commented before, there are also some individuals within a cluster that should not “belong” to them (see Figure 15). So, if the decision boundary tries to fit all of these “misclassified” individuals, you probably will incur into a worse accuracy once the prediction is performed.

Bearing this aspect in mind, several configurations were checked with the test set in order to obtain their accuracy; resulting that a simple structure of one hidden layer with two neurons would be the best solution.

The next figures show the scheme of the final network and the classification model in 2-D:

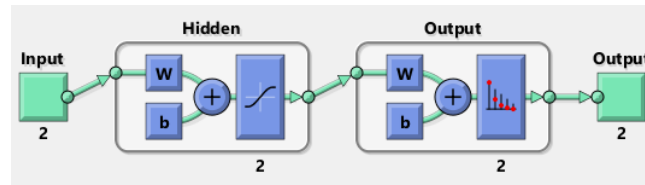


Figure 14. *Scheme of the Neural Network.*

It can be seen in next figure, that the decision boundary tries to capture the small group of casualties in the bottom, but, as explained before, there will be some misclassified individuals in both big clusters.

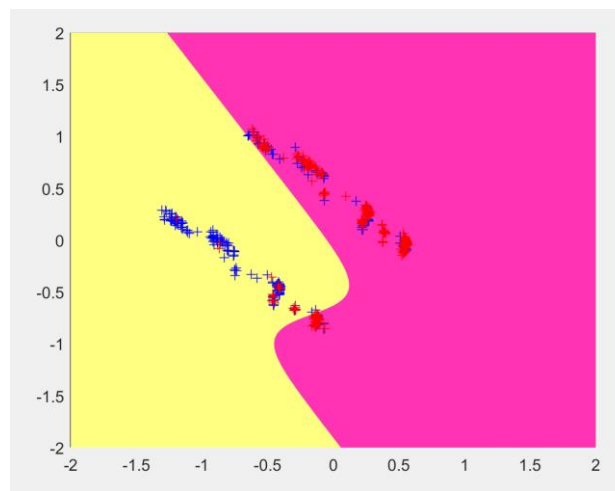


Figure 15. *Neural Network classifier model.*

9. Prediction

Once the predictive model is created, all the previous steps (preparation data and PCA) are applied to the test set in order to obtain the prediction.

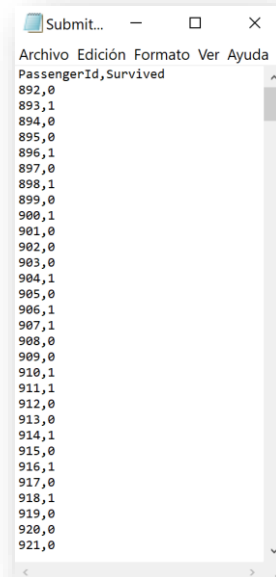


Figure 16. Prediction file with the test set.

This prediction is saved in a CSV file and uploaded to Kaggle, where **a score is given according to the accuracy**. The following figure shows the score acquired and the position in the ranking:

3458	▼ 403	mwyo00	0.78468	13	2mo
4453	▲ 2510	Madison Rosen	0.78468	3	6h
4454	new	UPC - Team	0.78468	10	2h

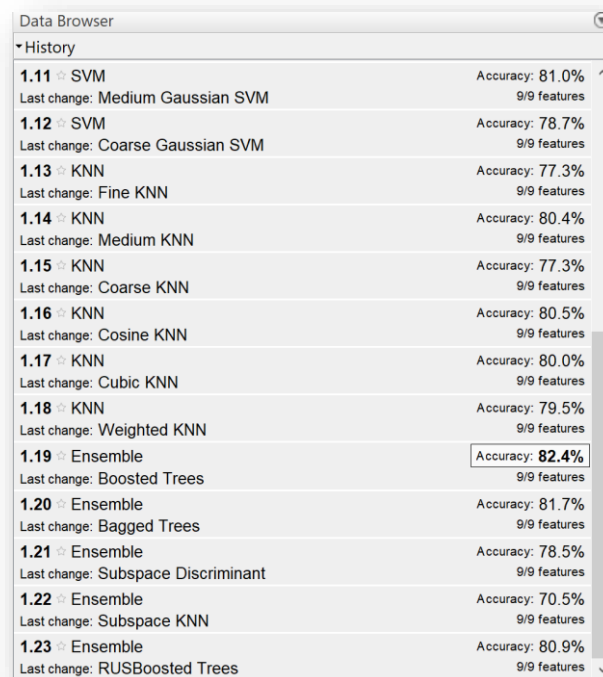
Figure 17. Ranking and score achieved.

As shown in Figure 11, **the accuracy of the model corresponds to a 78,468%**. This puts the model in position 4.454th. However, note that all the teams from this position until 3.458th have the same accuracy. If we recall that there are 11.276 teams, this means that **only the 30% of the teams** participating in this competition **have a higher score**.

10. Conclusions

These are the main conclusions of the project:

1. Checking the forums about this challenge, it seems that **PCA + NN is not really the best option** for this specific problem. There are some participants that used many machine learning tools to compare their accuracy and PCA + NN is always one of the combinations with less accuracy.
2. However, we used this combination to **practice** the methods that we have learnt at class.
3. Just out of curiosity, using **Classification Learner** in Matlab we get a much better accuracy (**82,4%**) using Boosted Tres.



Data Browser		
History		
1.11	SVM	Accuracy: 81.0%
Last change: Medium Gaussian SVM		9/9 features
1.12	SVM	Accuracy: 78.7%
Last change: Coarse Gaussian SVM		9/9 features
1.13	KNN	Accuracy: 77.3%
Last change: Fine KNN		9/9 features
1.14	KNN	Accuracy: 80.4%
Last change: Medium KNN		9/9 features
1.15	KNN	Accuracy: 77.3%
Last change: Coarse KNN		9/9 features
1.16	KNN	Accuracy: 80.5%
Last change: Cosine KNN		9/9 features
1.17	KNN	Accuracy: 80.0%
Last change: Cubic KNN		9/9 features
1.18	KNN	Accuracy: 79.5%
Last change: Weighted KNN		9/9 features
1.19	Ensemble	Accuracy: 82.4%
Last change: Boosted Trees		9/9 features
1.20	Ensemble	Accuracy: 81.7%
Last change: Bagged Trees		9/9 features
1.21	Ensemble	Accuracy: 78.5%
Last change: Subspace Discriminant		9/9 features
1.22	Ensemble	Accuracy: 70.5%
Last change: Subspace KNN		9/9 features
1.23	Ensemble	Accuracy: 80.9%
Last change: RUSBoosted Trees		9/9 features

Figure 18. *Classification Learner results.*

4. With this accuracy, we would have jumped to **206th** position (percentile of the best 2% scores).
5. Anyway, considering the chaos in the final hours of the Titanic, a **78% accuracy rate is decent**.

As general conclusions:

6. One of the main tasks of the project was the **data preparation**. As shown at the beginning of this report, there were **a lot of missing values**; dealing with them was one of the most **challenging and time-consuming** parts of this project.
7. As expected, **female, children and 1st class** passengers have higher probability of **surviving**. This hypothesis was confirmed analyzing the statistics of the data.
8. It is important **not to discard any variable**, even when seems useless (e. g., Name) we can create **proxy variables** that are useful (e. g., Title).
9. **Principal Component Analysis** is a powerful tool when dealing with **high-dimension problems**. The 12 initial dimensions were reduced to 3 Principal Components that collected the **90% of the variance**
10. **Artificial Neural Network** is a useful tool to classify big data problems. However, in this specific one, seems **not to be the best option**.
11. Finally, it is very important **not to overfit** the model. We saw that higher accuracies in the Training set, led to worse results in the Test set.

11. References

- [1] "Mormon," 21 May 2018. [Online]. Available:
<https://zonamormon.wordpress.com/2015/04/17/mormones-en-el-titanic/>. [Accessed 7th April 2018].

- [2] "Wikia," 21 May 2018. [Online]. Available:
http://foreverknight.wikia.com/wiki/File:Route_of_Titanic.svg. [Accessed 7th April 2018].

- [3] "Kaggle," 21 May 2018. [Online]. Available: <https://www.kaggle.com/c/titanic>. [Accessed 21 May 2018].



Escola de Camins

Escola Tècnica Superior d'Enginyeria de Camins, Canals i Ports
UPC BARCELONATECH

Models for decision making and optimization in engineering

Annex A: MATLAB Code

Content

1. PCA_and_NN.....	2
--------------------	---

1. PCA_and_NN

```
clear;
clc;
close all;

% VARIABLES:
% survival      Survival (0 = No; 1 = Yes)
% pclass        Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
% name          Name
% sex           Sex
% age           Age
% sibsp         Number of Siblings/Spouses Aboard
% parch         Number of Parents/Children Aboard
% ticket        Ticket Number
% fare          Passenger Fare
% cabin         Cabin
% embarked      Port of Embarkation (C = Cherbourg; Q = Queenstown;
S = Southampton)

%% Previous tasks
% Import the data
Train = readtable('train.csv','Format','%f%f%f%q%C%f%f%f%q%f%q%C');
Test = readtable('test.csv','Format','%f%f%q%C%f%f%f%q%f%q%C');

%% Detect missing values

% The fare cannot be 0 assuming that all the passengers payed for
their
% ticket. Therefore, if a 0 appears on the fare column, the value is
% considered incorrect and will be calculated later on. For now, we
just
% remove the 0 values and replace them by NaN:
Train.Fare(Train.Fare == 0) = NaN;
Test.Fare(Train.Fare == 0) = NaN;

% Plot the missing values
figure(1)
subplot(1,2,1)
M=ismissing(Train);
imagesc(M)
ax=gca;
ax.XTick = 1:width(Train);
ax.XTickLabel= Train.Properties.VariableNames;
ax.XTickLabelRotation = 90;
colormap([1 1 1 ; 1 0 0]);
title('Missing Values before Data Preparation')

%% Data preparation

% Deal with EMBARKED missing
values
% This variable corresponds to port at which the passenger embarked.
% We replace the missing values with the most common (mode) Embarked
Port:
Train.Embarked= renamecats(Train.Embarked,{'C' 'Q' 'S'},{'Cherbourg'
'Queenstown' 'Southampton'});
```

```
Train.Embarked(isundefined(Train.Embarked)) = mode(Train.Embarked);
Test.Embarked= renamecats(Test.Embarked,{ 'C' 'Q' 'S'},{ 'Cherbourg'
'Queenstown' 'Southampton'});
Test.Embarked(isundefined(Test.Embarked)) = mode(Test.Embarked);

% Deal with FARE missing
values
% The missing values are filled with the average of each Pclass.
for i=1:3
    Train.Fare(Train.Pclass==i & isnan(Train.Fare))=
nanmean(Train.Fare(Train.Pclass==i));
    Test.Fare(Test.Pclass==i & isnan(Test.Fare))=
nanmean(Test.Fare(Test.Pclass==i));
end

% Deal with CABIN missing
values
% It is assumed that the room number doesn't play a role: only the
deck in
% which the cabin was located matters.
% The passengers with no cabin assigned are identified according to
their Pclass:
No_cabin=cell2mat(cellfun(@(x)
any(isempty(x)),Train.Cabin,'UniformOutput',false));
No_cabin_Pclass_1=logical(No_cabin.*(Train.Pclass==1));
No_cabin_Pclass_2=logical(No_cabin.*(Train.Pclass==2));
No_cabin_Pclass_3=logical(No_cabin.*(Train.Pclass==3));
% The passengers with cabin assigned are identified according to their
Pclass:
Yes_cabin=cell2mat(cellfun(@(x)
any(~isempty(x)),Train.Cabin,'UniformOutput',false));
Yes_cabin_Pclass_1=logical(Yes_cabin.*(Train.Pclass==1));
Yes_cabin_Pclass_2=logical(Yes_cabin.*(Train.Pclass==2));
Yes_cabin_Pclass_3=logical(Yes_cabin.*(Train.Pclass==3));
% The number of the cabin is deleted, and we only keep the deck
Cabin_vector=(char(Train.Cabin));
Train.Cabin=Cabin_vector(:,1);
% Then, it is calculated the most common deck for each Pclass
[unique, ~,
string_map]=unique(Cabin_vector(logical(Yes_cabin_Pclass_1)));
Most_common_deck_Pclass_1=unique(mode(string_map));
clear unique
[unique, ~,
string_map]=unique(Cabin_vector(logical(Yes_cabin_Pclass_2)));
Most_common_deck_Pclass_2=unique(mode(string_map));
clear unique
[unique, ~,
string_map]=unique(Cabin_vector(logical(Yes_cabin_Pclass_3)));
Most_common_deck_Pclass_3=unique(mode(string_map));
clear unique
% Then, the most common Deck is assigned to the passengers of each
Pclass
Train.Cabin(No_cabin_Pclass_1)=Most_common_deck_Pclass_1;
Train.Cabin(No_cabin_Pclass_2)=Most_common_deck_Pclass_2;
Train.Cabin(No_cabin_Pclass_3)=Most_common_deck_Pclass_3;
% Finally, we rename the variable Cabin to Deck
Train.Properties.VariableNames('Cabin')= {'Deck'};

% Do the same for the Test set
```



```
% The passengers with no cabin assigned are identified according to
their Pclass:
No_cabin=cell2mat(cellfun(@(x)
any(isempty(x)),Test.Cabin,'UniformOutput',false));
No_cabin_Pclass_1=logical(No_cabin.*(Test.Pclass==1));
No_cabin_Pclass_2=logical(No_cabin.*(Test.Pclass==2));
No_cabin_Pclass_3=logical(No_cabin.*(Test.Pclass==3));
% The passengers with cabin assigned are identified according to their
Pclass:
Yes_cabin=cell2mat(cellfun(@(x)
any(~isempty(x)),Test.Cabin,'UniformOutput',false));
Yes_cabin_Pclass_1=logical(Yes_cabin.*(Test.Pclass==1));
Yes_cabin_Pclass_2=logical(Yes_cabin.*(Test.Pclass==2));
Yes_cabin_Pclass_3=logical(Yes_cabin.*(Test.Pclass==3));
% The number of the cabin is deleted, and we only keep the deck
Cabin_vector=(char(Test.Cabin));
Test.Cabin=Cabin_vector(:,1);
% Then, it is calculated the most common deck for each Pclass
[unique,~,
string_map]=unique(Cabin_vector(logical(Yes_cabin_Pclass_1)));
Most_common_deck_Pclass_1=unique(mode(string_map));
clear unique
[unique,~,
string_map]=unique(Cabin_vector(logical(Yes_cabin_Pclass_2)));
Most_common_deck_Pclass_2=unique(mode(string_map));
clear unique
[unique,~,
string_map]=unique(Cabin_vector(logical(Yes_cabin_Pclass_3)));
Most_common_deck_Pclass_3=unique(mode(string_map));
% Then, the most common Deck is assigned to the passengers of each
Pclass
Test.Cabin(No_cabin_Pclass_1)=Most_common_deck_Pclass_1;
Test.Cabin(No_cabin_Pclass_2)=Most_common_deck_Pclass_2;
Test.Cabin(No_cabin_Pclass_3)=Most_common_deck_Pclass_3;
% Finally, we rename the variable Cabin to Deck
Test.Properties.VariableNames('Cabin')= {'Deck'};

%% Feature engineering

% We will create another variable from 'ticket','sibsp' and 'parch'
that will be
% RELATIVES. This new variable will capture the number of relatives
onboard
% From variable Ticket it is assumed that if the Ticket is repeated,
the person with that ticket had
% relatives aboard
for i=1:length(Train.Ticket)
    idx = strfind(Train.Ticket, Train.Ticket(i));
    idx = find(not(cellfun('isempty', idx)));
    N = length(idx);
    Train.Relatives(i)=N-1;
end
for i=1:length(Test.Ticket)
    idx = strfind(Test.Ticket, Test.Ticket(i));
    idx = find(not(cellfun('isempty', idx)));
    N = length(idx);
    Test.Relatives(i)=N-1;
end
```

```
Train.Relatives(logical((Train.SibSp>0) |  
(Train.Parch>0)==1))=Train.SibSp(logical((Train.SibSp>0) |  
(Train.Parch>0)==1))+ Train.Parch(logical((Train.SibSp>0) |  
(Train.Parch>0)==1));  
Test.Relatives(logical((Test.SibSp>0) |  
(Test.Parch>0)==1))=Test.SibSp(logical((Test.SibSp>0) |  
(Test.Parch>0)==1))+ Test.Parch(logical((Test.SibSp>0) |  
(Test.Parch>0)==1));  
% Finally, we delete 'ticket','sibsp' and 'parch'  
Train(:,{'SibSp','Parch','Ticket'}) = [];  
Test(:,{'SibSp','Parch','Ticket'}) = [];  
  
% We obtain the TITLE of each passenger  
Title = cellfun(@(x) strsplit(x,' '),  
Train.Name(:),'UniformOutput',false);  
for i=1:length(Title)  
    Title{i}=Title{i}{2};  
end  
Title= cellfun(@(x) strsplit(x,' '), Title(:),'UniformOutput',false);  
for i=1:length(Title)  
    Title{i}=Title{i}{1};  
end  
Train.Title=categorical(Title);  
Title = cellfun(@(x) strsplit(x,' '),  
Test.Name(:),'UniformOutput',false);  
for i=1:length(Title)  
    Title{i}=Title{i}{2};  
end  
Title= cellfun(@(x) strsplit(x,' '), Title(:),'UniformOutput',false);  
for i=1:length(Title)  
    Title{i}=Title{i}{1};  
end  
Test.Title=categorical(Title);  
% We assign an age for the missing values according to the Title  
categories_Train=length(categories(Train.Title));  
categories_Test=length(categories(Test.Title));  
for i=1:categories_Train  
    Train.Age(isnan(Train.Age) & double(Train.Title)==i)=  
nanmean(Train.Age(double(Train.Title)==i));  
end  
for i=1:categories_Test  
    Test.Age(isnan(Test.Age) & double(Test.Title)==i)=  
nanmean(Test.Age(double(Test.Title)==i));  
end  
Test.Age(89)=nanmean(Test.Age);  
  
% Check that now there aren't missing  
values  
figure(1)  
subplot(1,2,2)  
imagesc(ismissing(Train))  
ax=gca;  
ax.XTick = 1:width(Train);  
ax.XTickLabel= Train.Properties.VariableNames;  
ax.XTickLabelRotation = 90;  
colormap([1 1 1 ; 1 0 0]);  
title('Missing Values after Data Preparation')  
  
%% Plot the basic statistics
```

```
% We also use categories insted of numerical values according to the
age to
% plot the class
Age_class = Train.Age;
Age_class = discretize(Age_class, [0 10 25 65 100], ...
    'categorical',{'Children','Junior','Adult','Senior'});
Train.Age_class=double(Age_class);

% Extract some basic
statistics
survived_statistics=grpstats(Train(:,{'Survived'}), 'Survived');
sex_statistics = grpstats(Train(:,{'Survived','Sex'}), 'Sex');
age_statistics = grpstats(Train(:,{'Survived','Age_class'}),
    'Age_class');
Pclass_statistics = grpstats(Train(:,{'Survived','Pclass'}),
    'Pclass');
relatives_statistics = grpstats(Train(:,{'Survived','Relatives'}),
    'Relatives');

% Fate of the
passengers
% Plot the age statistics
figure(2)
p = uipanel('Parent',figure(2),'BorderType','none');
p.Title = 'Fate of the passengers';
p.TitlePosition = 'centertop';
p.FontSize = 12;
p.FontWeight = 'bold';
% set(gcf,'position',[500 450 900 400])
subplot(1,3,1,'Parent',p)
histogram(Train.Age_class(Train.Survived ==
0), 'FaceColor', 'r', 'facealpha',0.75)
hold on
histogram(Train.Age_class(Train.Survived ==
1), 'FaceColor', 'b', 'facealpha',0.45)
hold off
ax=gca;
ax.XTick = 1:4;
ax.XTickLabel= {'Children','Junior','Adult','Senior'};
legend('Didn''t Survive', 'Survived')
legend('Location','northwest')
ylabel('Number of passengers')

% Plot the Pclass statistics
figure(2)
subplot(1,3,3,'Parent',p)
histogram(Train.Pclass(Train.Survived ==
0), 'FaceColor', 'r', 'facealpha',0.75)
hold on
histogram(Train.Pclass(Train.Survived ==
1), 'FaceColor', 'b', 'facealpha',0.45)
hold off
legend('Didn''t Survive', 'Survived')
legend('Location','northwest')
ax=gca;
ax.XTick = 1:3;
ax.XTickLabel= {'1st Class','2nd Class','3rd Class'};
ylabel('Number of passengers')
```

```
legend('Location','northwest')

% Plot the sex statistics
figure(2)
subplot(1,3,2,'Parent',p)
histogram(Train.Sex(Train.Survived ==
0), 'FaceColor','r', 'facealpha',0.75, 'BarWidth', 0.7)
hold on
histogram(Train.Sex(Train.Survived ==
1), 'FaceColor','b', 'facealpha',0.45, 'BarWidth', 0.7)
hold off
legend('Didn't Survive', 'Survived')
legend('Location','northwest')
ax=gca;
ax.XTick = {'female','male'};
ax.XTickLabel= {'Female','Male'};
ylabel('Number of passengers')
legend('Location','northwest')

% Percentage of passengers that didn't
survived
% Plot the percentage of survivors depending on the age
figure(3)
j = uipanel('Parent',figure(3),'BorderType','none');
j.Title = 'Percentage of survivors';
j.TitlePosition = 'centertop';
j.FontSize = 12;
j.FontWeight = 'bold';
% set(gcf,'position',[10 50 1200 400])
subplot(1,3,1,'Parent',j)
bar([1 2 3
4], (age_statistics.mean_Survived*100), 'b', 'facealpha',0.75);
ylabel('% of passengers')
average_survived =
survived_statistics.GroupCount('1')/(survived_statistics.GroupCount('0
')+survived_statistics.GroupCount('1'))*100;
ax=gca;
ax.XTick = 1:4;
ax.XTickLabel= {'Children','Junior','Adult','Senior'};
line([0,5],[average_survived,average_survived])
xlim([0 5]);
ylim([0 100]);
legend('Survived', 'Survived average')
legend('Location','northwest')

% Plot the gender statistics
figure(3)
subplot(1,3,2,'Parent',j)
bar([1
2], (sex_statistics.mean_Survived*100), 'b', 'facealpha',0.75, 'BarWidth',
0.5);
ax=gca;
ax.XTick = 1:2;
ax.XTickLabel= {'Female','Male'};
ylabel('% of passengers')
xlim([0.5 2.5]);
ylim([0 100]);
line([0,3],[average_survived,average_survived])
legend('Survived', 'Survived average')
```

```
legend('Location','northwest')

% Plot the percentage of survivors depending on the Pclass
figure(3)
subplot(1,3,3,'Parent',j)
bar([1 2
3],(Pclass_statistics.mean_Survived*100),'b','facealpha',0.75,'BarWidtht
h', 0.7);
ax=gca;
ax.XTick = 1:3;
ax.XTickLabel= {'1st Class','2nd Class','3rd Class'};
xlim([0.5 3.5]);
ylim([0 100]);
ylabel('% of passengers')
line([0,4],[average_survived,average_survived])
legend('Survived','Survived average')
legend('Location','northwest')

%% Preparation for the model
% Delete the variables that don't perform any rol
Train(:,{'Name','Age_class'}) = [];
Test(:,{'PassengerId','Name'}) = [];

% Before using any model, we transform the
categorical/logical/character
% variables into numerical 'double' values
Train.Deck=double(categorical(cellstr(Train.Deck)));
Train.Sex=double(Train.Sex);
Train.Fare=double(Train.Fare);
Train.Embarked=double(Train.Embarked);
Train.Title=double(Train.Title);
Test.Deck=double(categorical(cellstr(Test.Deck)));
Test.Sex=double(Test.Sex);
Test.Fare=double(Test.Fare);
Test.Embarked=double(Test.Embarked);
Test.Title=double(Test.Title);

% Transform the Table to a Matrix
Data_Train=table2array(Train);
Data_Test=table2array(Test);

%% PCA
% Note that Data must be (measurements x dimensions)
Survive_index=2;

% Divide the data between those who survived and those who didn't
Survived_interval=Data_Train(:,Survive_index)==1;
Not_Survived_interval=Data_Train(:,Survive_index)==0;
% For each group, sort according to the Survived
Survived=Data_Train(Survived_interval,:);
Not_Survived=Data_Train(Not_Survived_interval,:);

% Put all the data together
Data_Train=[Survived;Not_Survived];

% Save the list of the passengers sorted
List_Train=Data_Train(:,1);
```

```
% Redefine the intervals for each group for a latter use
Survived_interval=Data_Train(:,Survive_index)==1;
Not_Survived_interval=Data_Train(:,Survive_index)==0;

Data_Train=Data_Train(:,Survive_index+1:end);

% First, we scale all the values between 0 and 1
for i=1:size(Data_Train,2)
    Data_Train(:,i)=( Data_Train(:,i)-mean(Data_Train(:,i)) ) / (
max(Data_Train(:,i))-min(Data_Train(:,i)) );
end

% % Then, we center the data by subtracting the mean
Data_centered = bsxfun(@minus, Data_Train, mean(Data_Train));

% Compute the Covariance matrix of Data
C=cov(Data_centered);

% Compute the eigenvalues and eigenvectors of the covariance matrix
[eigenvectors,eigenvalues_matrix] = eig(C);

% Sort the eigenvectors matrix in descending order according to the
% eigenvalues
[D,rindices]=sort(diag(eigenvalues_matrix),'descend');
eigenvectors=eigenvectors(:,rindices);

% Compute the transformation matrix W
W=eigenvectors';

% Project the data with the PC's
Data_projected=(W*Data_centered)';

% Keep the first 3 PC
PC1=Data_projected(:,1);
PC2=Data_projected(:,2);
PC3=Data_projected(:,3);

% At this point we want to calculate how much variance do these PC
capture
Variance_PC1=num2str(D(1)/sum(D));
Variance_PC2=num2str(D(2)/sum(D));
Variance_PC3=num2str(D(3)/sum(D));
fprintf('Percentatge of variance collected: PC1: %s, PC2: %s, PC3:
%s\n',Variance_PC1,Variance_PC2,Variance_PC3)
Cumulative=cumsum(D)/sum(D);

% Separate the projected data for each subgroup
PC1_survived=PC1(Survived_interval);
PC2_survived=PC2(Survived_interval);
PC3_survived=PC3(Survived_interval);
PC1_not_survived=PC1(Not_Survived_interval);
PC2_not_survived=PC2(Not_Survived_interval);
PC3_not_survived=PC3(Not_Survived_interval);

% Plot the projected data for each
group
figure(4)
scatter3(PC1_survived,PC2_survived,PC3_survived,'b', 'filled')
```

```
hold on
scatter3(PC1_not_survived,PC2_not_survived,PC3_not_survived,'r',
'filled')
hold off
axis equal
xlabel(['PC1 (' Variance_PC1 ')'])
ylabel(['PC2 (' Variance_PC2 ')'])
zlabel(['PC3 (' Variance_PC3 ')'])
legend('Survivors','Not survivors')

figure(5)
biplot(eigenvectors(:,1:2),'Scores',[],'Varlabels',{ 'Pclass' 'Sex'
'Age' 'Fare' 'Deck' 'Embarked' 'Relatives' 'Title'});
xlabel(['PC1 (' Variance_PC1 ')'])
ylabel(['PC2 (' Variance_PC2 ')'])

figure(6)
bar(1:length(Cumulative),Cumulative*100,'b','facealpha',0.75);
hold on
plot(Cumulative*100,'.-r')
ax=gca;
ax.XTick = 1:length(Cumulative);
ax.XTickLabel= {'PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8'};
ylabel('% of variance explained')
title('Explained variance')

%% Neural Network

% Define 2 clusters of input data
A = [PC1_survived'; PC2_survived'];
B = [PC1_not_survived'; PC2_not_survived'];
% A = [PC1_survived'; PC2_survived'; PC3_survived'];
% B = [PC1_not_survived'; PC2_not_survived'; PC3_not_survived'];

% plot clusters
figure(7)
plot(A(1,:),A(2:,:), 'b+')
hold on
grid on
plot(B(1,:),B(2:,:), 'r+')

% coding (+1/-1) of 2 separate classes
a = [0 1]';
b = [1 0]';

% define inputs
Input = [A B];

% define targets
Target = [repmat(a,1,length(A)) repmat(b,1,length(B))];

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';
```

```
hiddenLayerSize = 2; % Define number of layers
trainRatio = 0.9; % training set [%]
valRatio = 0.1; % validation set [%]
testRatio = 0.0; % test set [%]

% Create a Pattern Recognition Network
net = patternnet(hiddenLayerSize, trainFcn);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = trainRatio;
net.divideParam.valRatio = valRatio;
net.divideParam.testRatio = testRatio;

% Train the Network
[net,tr] = train(net,Input,Target);

% Test the Network
y = net(Input);
e = gsubtract(Target,y);
performance = perform(net,Target,y);
tind = vec2ind(Target);
yind = vec2ind(y);
percentErrors = sum(tind ~= yind)/numel(tind);

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotconfusion(t,y)
%figure, plotroc(t,y)

% evaluate performance: decoding network response
[~,i] = max(Target); % target class
[m,j] = max(y); % predicted class
N = length(y); % number of all samples
k = 0; % number of missclassified samples
if find(i-j) % if there exist missclassified samples
    k = length(find(i-j)); % get a number of missclassified samples
end
fprintf('Correct classified samples: %.1f%% samples\n', 100*(N-k)/N)

%% Test the model with the test data
% Note that Data must be (measurements x dimensions)

% First, we scale all the values between 0 and 1
for i=1:size(Data_Test,2)
    Data_Test(:,i)=( Data_Test(:,i)-mean(Data_Test(:,i)) ) / (
max(Data_Test(:,i))-min(Data_Test(:,i)) );
end

%% Then, we center the data by subtracting the mean
Data_centered = bsxfun(@minus, Data_Test, mean(Data_Test));
```



```
% Project the data with the PC's
Data_projected=(W*Data_centered')';

% Keep the first 2 PC
PC1=Data_projected(:,1);
PC2=Data_projected(:,2);
PC3=Data_projected(:,3);

% define the test data
Xpred = [PC1'; PC2'];
%Xpred = [PC1'; PC2'; PC3'];
Test_Prediction = net(Xpred);

for i=1:size(Test_Prediction,2)
    if Test_Prediction(1,i)<Test_Prediction(2,i)
        Survived_prediction(i)=1;
    else
        Survived_prediction(i)=0;
    end
end

Survived_prediction=Survived_prediction';
Result=[[1:length(Survived_prediction)]'+891 Survived_prediction];

% generate a grid
figure(7)
span = -2:.005:2;
[P1,P2] = meshgrid(span,span);
pp = [P1(:) P2(:)]';
% simulate neural network on a grid
aa = net(pp);
% plot classification regions
m = mesh(P1,P2,reshape(aa(1,:),length(span),length(span))-5);
set(m,'facecolor',[1 0.2 .7],'linestyle','none');
hold on
m = mesh(P1,P2,reshape(aa(2,:),length(span),length(span))-5);
set(m,'facecolor',[1 1.0 0.5],'linestyle','none');
view(2)
```