TITANIC: MACHINE LEARNING FROM DISASTER

FINDING THE MACHINE LEARNING MODEL WITH HIGH ACCURACY TO PREDICT.

Overview

The sinking of the Titanic is one of the most infamous shipwrecks in history.
On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.
While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.
In this challenge, we ask you to build a predictive model that answers the question: “what sorts of people were more likely to survive?” using passenger data (ie name, age, gender, socio-economic class, etc).

Text from:
Titanics Taggle Competition.
https://www.kaggle.com/c/titanic

DATASET

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embar
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 31	7.925		S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075		S
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742	11.1333		S
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	30.0708		C
11	1	3	Sandstrom, Miss. Marguerite Rut	female	4	1	1	PP 9549	16.7	G6	S
12	1	1	Bonnell, Miss. Elizabeth	female	58	0	0	113783	26.55	C103	S
13	0	3	Saundercock, Mr. William Henry	male	20	0	0	A/5. 2151	8.05		S
14	0	3	Andersson, Mr. Anders Johan	male	39	1	5	347082	31.275		S
15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14	0	0	350406	7.8542		S
16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55	0	0	248706	16		S
17	0	3	Rice, Master. Eugene	male	2	4	1	382652	29.125		Q
18	1	2	Williams, Mr. Charles Eugene	male		0	0	244373	13		S
19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)	female	31	1	0	345763	18		S
20	1	3	Masselmani, Mrs. Fatima	female		0	0	2649	7.225		C
21	0	2	Fynney, Mr. Joseph J	male	35	0	0	239865	26		S
22	1	2	Beesley, Mr. Lawrence	male	34	0	0	248698	13	D56	S
23	1	3	McGowan, Miss. Anna "Annie"	female	15	0	0	330923	8.0292		Q
24	1	1	Sloper, Mr. William Thompson	male	28	0	0	113788	35.5	A6	S
25	0	3	Palsson, Miss. Torborg Danira	female	8	3	1	349909	21.075		S
26	1	3	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia Johansson)	female	38	1	5	347077	31.3875		S
27	0	3	Emir, Mr. Farred Chehab	male		0	0	2631	7.225		C
28	0	1	Fortune, Mr. Charles Alexander	male	19	3	2	19950	263	C23 C25 C27	S
29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female		0	0	330959	7.8792		Q
30	0	3	Todoroff, Mr. Lalio	male		0	0	349216	7.8958		S
31	0	1	Uruchurtu, Don. Manuel E	male	40	0	0	PC 17601	27.7208		C
32	1	1	Spencer, Mrs. William Augustus (Marie Eugenie)	female		1	0	PC 17569	146.5208	B78	C

We have two files, 'train.csv' & 'test.csv' we use the data from train.csv to create a model and after with this model we have to predict if the passenger survived or not, using the feature data from test.csv. To begin we only use the features marked in yellow to train the model, maybe after we have to eliminate one or more features to get more accuracy or reduce a potencially high bias issue. we are not considering use a cross-validation test yet

DATASET TREATMENT

```python
import csv
import torch
import numpy as np
import pandas as pd


def main():
    Dataframe = pd.read_csv('train.csv')

    sex = Dataframe[['Sex']].dropna(axis =0, how = 'any')
    embarked = Dataframe[['Embarked']]

    Sexo = np.zeros((np.size(sex),1))
    Embarked = np.zeros((np.size(embarked) ,1))

    for i in range(np.size(sex)):
        if sex.values[i] == 'male':
            Sexo[i] = 1.0
        else:
            Sexo[i] = 2.0
        if embarked.values[i] == 'C':
            Embarked[i] = -1.0
        elif embarked.values[i] == 'Q':
            Embarked[i] = 0.0
        elif embarked.values[i] == 'S':
            Embarked[i] = 1.0
        else:
            Embarked[i] = 0.5

    myFile = open('sex.csv', 'w')
    with myFile:
        writer = csv.writer(myFile)
        writer.writerows(Sexo)

    myFile = open('Embarked.csv', 'w')
    with myFile:
        writer = csv.writer(myFile)
        writer.writerows(Embarked)


if __name__ == "__main__":
```

First, I created a python script to transform some features like a 'Sex', 'Embarked' from strings to a numerical value, for example if one example(i) is = 'male' therefore Sexo(i) = 1.0.

After of this I saved the numerical information in a csv file to treat this in Octave. (sex.csv & embarked.csv)

Note that I choose close values to guarantee the normalization in this features, maybe after we have to apply a algorithm to normalize other numerical features but first we will focus in build a fast model.

Part 1.- Readfile in Octave

```
#------------ Part 1-------------#
#Load data from the csv files

[y, X] = readfile();
```

```
function [y, X] = readfile()

#read Files csv using function csvread
Dataset = csvread('train.csv');
sex = csvread('sex.csv');
embarked = csvread('Embarked.csv');

#Vector y getting from csv file
y = Dataset(2:end, 2);
X = [];
#using the diferents matrix getting from the csv file we join this in a only matrix X with n features
X = [X, Dataset(2:end, 3), sex,  Dataset(2:end, 7:9), Dataset(2:end, 11), embarked];

end
```

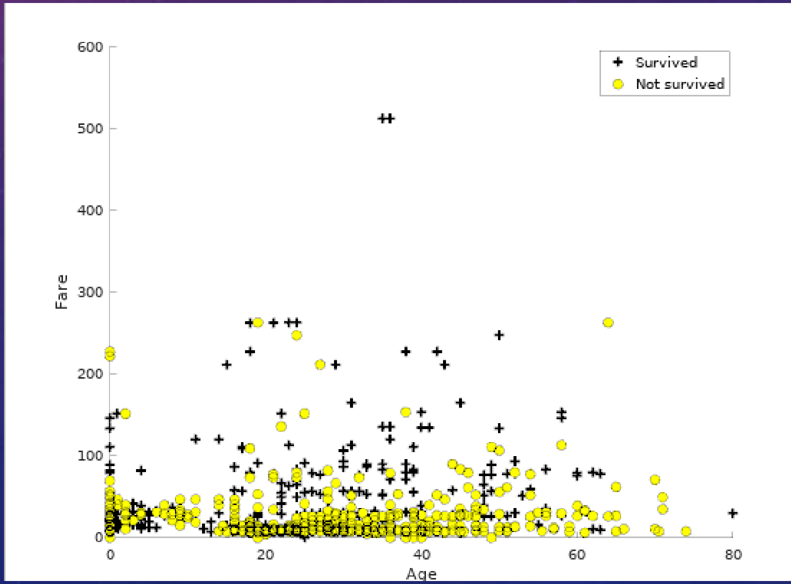
Using the script readfile we get the y values a X matrix. We use the csv files generated previously in our script in python. To get the searched structure of data more fast.

Part 2.- Plot 2 features

```
#------------ Part 2-------------#
#Plot Data

fprintf(['Plotting data with + indicating (y = 1) examples and o ' 'indicating (y = 0) examples.\n']);

PlotData(X, y);

% Put some labels
hold on;
% Labels and Legend
xlabel('Age')
ylabel('Fare')

% Specified in plot order
legend('Survived', 'Not survived')
hold off;
```

```
function PlotData(X, y)

figure;
hold on;

pos = find(y == 1);
neg = find(y == 0);

plot(X(pos, 3), X(pos, 6),  'k+', 'LineWidth', 2, 'MarkerSize', 7);
plot(X(neg, 3), X(neg, 6), 'ko', 'MarkerFaceColor', 'y', 'MarkerSize', 7);

hold off;


end
```

I will plot only 2 features in this example we print Age vs Fare to view the distribution.



Part 3.- Compute cost J and gradient

```
#---------------Part 3 ------------#
#Compute Cost (J) & Gradient

#Initialize useful values
[m, n] = size(X);

#add  column x0 to complete the matrix X
X = [ones(m, 1), X];

#Initialize values of theta according with number of features x
init_theta = zeros(n + 1, 1);

#Compute cost and gradient function that we will used to train the model
[cost, grad] = Cost_grad(init_theta, X, y);
```

```
function [cost, grad] = Cost_grad(theta, X, y)
#initialize useful values
cost = 0;
grad = zeros(size(theta));
m = length(y);

#Compute cost J
cost = (1.0/m)*(-y'*log(sigmoid(X*theta)) - (1-y)'*log(1-(sigmoid(X*theta))));

#Compute gradient.
grad = (1.0/m)*X'*(sigmoid(X*theta)-y);

end
```

The function Cost_grad is useful to train the model after.

$$h = g(X\theta)$$

$$J(\theta) = \frac{1}{m} \cdot \left(-y^T \log(h) - (1-y)^T \log(1-h)\right)$$

Part 4.- Compute gradient descent

```
#--------------Part 4 ------------#
#Compute Gradient Descent (Manual Repeat method)

fprintf('Printing optimal theta values using gradien descent\n');
#Set numer of iterations and alpha value
iterations = 1500;
alpha = 0.01;

#calculate optimal theta values using repeat method
[theta] = descent_grad(X, y, init_theta, alpha, iterations)
```

```
function theta1 = descent_grad(X, y, theta, alpha, iterations)

#Initialize useful values
m = length(y);

for i = 1:iterations,
  theta1 = theta1 - (alpha/m)*X'*(sigmoid(X*theta)-y);
endfor

end
```

In this part we calculate the optimal values of theta using the gradient descent in 1500 iterations

$$\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y})$$

Vectorized implementation

```
theta1 =

  -0.1745343
  -0.8105008
   1.1729786
  -0.0676387
  -0.5887735
  -0.0386071
   0.0065384
  -0.3497570
```

Part 5.- Compute gradient descent using fmcing

```
#-----------Part 5  Compute grad using fmcing-----------#

options = optimset('GradObj', 'on', 'MaxIter', 400);

[theta, cost] = ...
   fminunc(@(t)(Cost_grad(t, X, y)), init_theta, options);
```

```
theta =

  -1.4988574
  -0.9812012
   2.7324930
  -0.0162132
  -0.2746127
  -0.0497049
   0.0023175
  -0.1841445
```

In this part we will use fmcing function to train the model and get the optimal value thetas.

Part 6.- Predict using test data 'test.csv'

```
#-----------Part 6 Predict------------#

[Xtest, ind] = readfile2();

m = size(Xtest,1);

yval = zeros(m, 1);

Xtest = [ones(m, 1), Xtest];

for i = 1:m,
  if sigmoid(theta'*Xtest(i, 1:end)') >= 0.5,
    yval(i, 1) = 1;
  endif

endfor

submission = [];
submission = [submission, ind, yval];

csvwrite('submission.csv', submission);
```

In the part 6 using the data from test.csv we calculate the y values predicted using the theta optimal values.

After we create a file csv to submit to the kaggle platform and evaluate the model accuracy

Evaluatin the model accurracy in kaggle platform

Name	Submitted	Wait time	Execution time	Score
submission.csv	19 hours ago	0 seconds	0 seconds	0.74641
Complete				

We getting a accurracy of 74.6% we have to make adjust in model to improve this perfomance.

Next steps

-Apply Feature normalization.
-Add regularization to the model
-Evaluate again the model.

If the acuuracy is not desired, we have to evaluate with learning curve if we have a problem of high variance or high bias and implement news methods to prevente this.

Contact:
fernando.aguilar1010@gmail.com