Final Report: Data Modeling and Analysis Results for the "Titanic: Machine Learning from Disaster" Competition (Term Project)

Submitted to:

Professor. Marek J. Druzdzel
School of Computing and Information and Intelligent Systems Program
University of Pittsburgh

Report Prepared By:

Zhiming Yang (zhy61@pitt.edu)
Chao Guo (chg101@pitt.edu)
Gokulnath Krishnan (gok11@pitt.edu)
Pooja Ghatge (pog4@pitt.edu)
Rohit Surana (rss80@pitt.edu)
Graduate Student, School of Computing and Information
University of Pittsburgh

December 3, 2017

Table of Content

Executive Summary

As one of the most popular shipwrecks in history, the sinking of RMS Titanic inflicted heavy casualties on the ship industry and shocked the international community. On April 15, 1912, the sinking of Titanic hit front-page headlines, killing tons of people and leading to better safety regulations for ships. Based on the group of survivors, except for lucky composition, our team will analyze the prediction that particular groups of people had better chance to survive, such as females, children, and the upper-class. More precisely, we will apply the techniques of machine learning to predict which passengers survived the misfortune.

Introduction

In the "Titanic: Machine Learning from Disaster" competition raised by Kaggle Inc., this report will address out the accurate prediction of survival in the test set (test.csv) given the training set (train.csv), by utilizing binary classification, Python and R basic skills and neural networks techniques.

We initially concentrate on the certain variables which influence mostly on the probability of survival. In other words, some of the unrelated variables are ignored while cleansing the data, which will be explained in the following section of the report.

After processing both sets, it is observed that there are 891 values with 12 columns in the training set and 418 values with 11 columns in the test set. Before implementing the analysis, it is necessary to convert the characters in the columns named as "Sex" and "Embarked" into numbers. The data processing will be shown in the following section.

Afterwards, four methods are considered to model the prediction of survival: Linear discriminant analysis and XGBoost in Python, Artificial Neural Network (ANN) in Python, Support Vector Machines (SVM) in Python, and Random Forest in R. The specific procedures and description will be provided in the following section.

Finally, the test set will evaluate each model that will generate different accuracy of analysis because of the different variables chosen. The submission on Kaggle's website will return the score, which is the percentage of passengers we correctly predict, known as "accuracy".

Background

There are 342 survivors in the training set, which occupies 38.3838% of all passengers on board. Among the total survivors, there are 109 males (18.8908%) and 233 females (25.7962%). In extreme case, our team assumes that all females in the test set survive and all males in the test set die. The following are the features in both sets and the statistical description of each feature generated in R.

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Table 1.1 The variables and definitions of each variable from Kaggle's website

pclass: A proxy for socio-economic status (SES)
1st = Upper
2nd = Middle
3rd = Lower
age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5
sibsp: The dataset defines family relations in this way...
Sibling = brother, sister, stepbrother, stepsister
Spouse = husband, wife (mistresses and fiancés were ignored)
parch: The dataset defines family relations in this way...
Parent = mother, father
Child = daughter, son, stepdaughter, stepson
Some children travelled only with a nanny, therefore parch=0 for them.

Table 1.2 The variable notes and explanations from Kaggle's website

```
> summary(dataset)
  PassengerId       Survived          Pclass      Sex           Age
 Min.   :  1.0   Min.   :0.0000   Min.   :1.000   1:577   Min.   : 0.42
 1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000   2:314   1st Qu.:22.00
 Median :446.0   Median :0.0000   Median :3.000           Median :28.00
 Mean   :446.0   Mean   :0.3838   Mean   :2.309           Mean   :29.36
 3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000           3rd Qu.:35.00
 Max.   :891.0   Max.   :1.0000   Max.   :3.000           Max.   :80.00

     SibSp           Parch           Ticket         Fare        Embarked
 Min.   :0.000   Min.   :0.0000   1601    :  7   Min.   :  0.00   1:  2
 1st Qu.:0.000   1st Qu.:0.0000   2343    :  7   1st Qu.:  7.91   2:168
 Median :0.000   Median :0.0000   347082  :  7   Median : 14.45   3: 77
 Mean   :0.523   Mean   :0.3816   2144    :  6   Mean   : 32.20   4:644
 3rd Qu.:1.000   3rd Qu.:0.0000   347088  :  6   3rd Qu.: 31.00
 Max.   :8.000   Max.   :6.0000   3101295 :  6   Max.   :512.33
                                  (Other):852
```

Table 1.3 The statistical description of each variable made in R

Limitations

During the process of analysis, one of the obstacles is cleansing data appropriately. Firstly, for the values with numbers and characters, such as the variable “Ticket”, we simply discard all the characters and signs, and keep the numbers only. Secondly, for the missing values in some variables, we directly assign the median of all values in the variable “Age” in the training set and the variable “Fare” in the test set, which might influence skewed distributions (the same cause and effect for assigned mean of all values). We are planning to find a more elaborate method (XGBoost is an option) to perfect models. The other obstacle is choosing relevant variables properly. According to the feature’s importance valued by Decision Tree classification and Random Forest classification, we have explored the comparatively low relevance of “SibSp”, “Parch”, “Ticket” with the probability of survival.

Decision Tree							
	Pclass	Age	Sex	Fare	SibSp	Parch	Embarked
Feature Importance	0.141303	0.17906	0.416167	0.179387	0.050397	0.019238	0.014448
Score	0.905723906						

Random Forest							
	Pclass	Sex	Age	Fare	SibSp	Parch	family_size
Feature Importance	0.103847	0.20139	0.319893	0.246029	0.052727	0.041592	0.03452128
Score	0.939393939						

Preparation

Cleansing Data

Firstly, we ignore the variable “Name” and “Cabin”.

Reasons: According to the calculation of importance of each feature by applying Decision Tree in Python, the column “Name” and “Cabin” are the least influential factors leading to survive eventually.

Secondly, we handle the missing values by using numeric feature and categorical features. After checking Filtered columns in Microsoft Excel, the column “Age”, and “Embarked” have blank cells. To deal with the missing values, we adopt R and Python to assign the median of all ages in the column “Age”. Lastly, in R, replace with “1” in the column “Embarked”, and in Python, assign “S” for missing values in the column “Embarked”.

Thirdly, we convert characterized values into integer numbers by the following three methods.

1. Microsoft Excel.

Description:

In the column “Sex”, we create two new columns, namely “male” and “female”. Then the column “male” will show “1” if the value “male” matches the cell in the column “Sex” and the same row, otherwise, “0” will be shown. Similarly, the column “female” will show “0” and “1” when condition is satisfied.

For the column “male”: =IF(E2 = "male",1,0); For the column “female”: =IF(E2 = "female",1,0)

In the column “Embarked”, we create three new columns, namely “Q”, “C” and “S”, representing “Queenstown”, “Cherbourg” and “Southampton” respectively. By using the same technique, it is shown “0” and “1” in each column when condition is satisfied.

For the column “Q”, =IF(L2 = "Q",1,0); For the column “C”, =IF(L2="C",1,0); For the column “S”, =IF(L2="S",1,0)

2. R

Description:

In the column “Sex”, we convert the “male” to “1” and the “female” to “2”. Then in the column “Ticket”, Lastly, in the column “Embarked”, we convert the “C” to “2”, the “Q” to “3” and the “S” to “4”. Also, many customers had same ‘ticket number”. The ticket number column had characters, we removed the characters and assigned same labels to all customers who were on the same ticket.

3. Python

Description:

In each column, we generate its median value and then replace missing values(NaN) with this median. And last, we use feature scaling for those columns having different numerical ranges so that they will have similar scales.

Methodology

We implement four analysis techniques: Support Vector Machines (SVM) in Python, Artificial Neural Network (ANN) in Python, Random Forest in R, and Linear Discriminant Analysis and XGBoost in Python.

I. Support Vector Machines (SVM) in Python

SVM is a supervised machine learning algorithm which is often used when solving classification problems. Since we don't know the boundary that our model calculates, it is possible that the boundary will not be a straight line. One important technique in SVM is called kernel trick. It helps to transform the data set and based on the transformations, SVM algorithm will be able to find an optimal boundary between features, hence SVM can capture a more complex relationship between data.

One important thing about implementing the SVM algorithm is to choose a kernel. The kernels we used in our case are the default kernel "rbf" and another kernel "linear". Rbf, also known as Radial basis function, is used for nonlinear classification, while linear kernel is used for linear classification. Since we don't know whether we should use linear or nonlinear classifier, both kernels are implemented and tested in our code. The classifier in scikit-learn is called SVC (Support Vector Classifier). The objective SVC is to fit to the data we provide and return a "best fit" hyperplane that divides, or categorizes the data set like this:

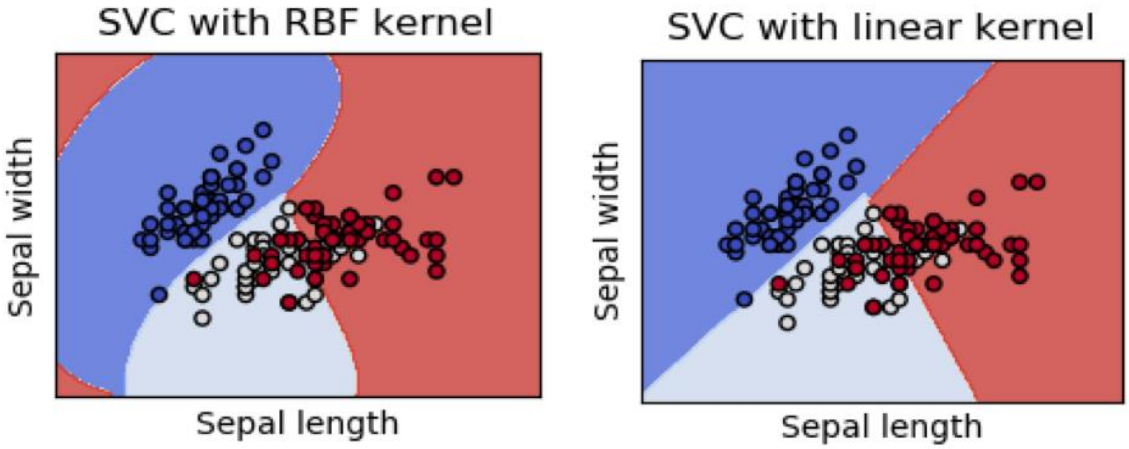


Figure.1 Support Vector Machines with different kernels

SVC can receive many parameters, but we mainly focus on 3 parameters: C, kernel and random_state. C, the penalty factor, is critical here, if we set the value too big, those points may not be able to separate, and we may store more support vectors than we want and the classifier will be overfitting. And if we choose this value too small, the classifier may be underfitting. Here we use the default value 0 for the parameter C. Kernel, like mentioned above, will be rbf and linear. The random_state is used when the classifier shuffles the data set. The value represents the seed of the generator the model used to generate random numbers, and we will use "None" here as the value.

After SVC learns our training set, we call predicting method on the classifier to generate the predictions.

II. Artificial Neural Network (ANN) in Python

ANN are computing systems inspired by the biological neural network. ANN progressively improve performance by considering examples from training data without task-specific programming. In our titanic project we have created a 3 layered ANN with 1 hidden layer, 1 output layer & 1 input layer. The input provided to the input layer are all the 22 features of all the 891 training examples. The hidden layer is assigned initially random weights and a bias = 0. We have changed the hidden layers gradually from 25 to 5 and the best results were obtained when hidden layers = 10. ANN basically does the job of finding the best weights and biases so that to fit the training data in the best possible way. We have used the following functions in our ANN:

- initialize_parameters(n_x, n_h, n_y)
- forward_propagation(X, parameters)
- compute_cost(A2, Y, parameters)
- backward_propagation(parameters, cache, X, Y)
- update_parameters(parameters, grads, learning_rate = 0.01)
- nn_model(X, Y, n_h, num_iterations=10000, print_cost=True)
- predict(parameters, X)

Our main function in nn_model which calls all the other functions. The parameters passed to nn_model are X - the training set input, Y - the training set output, n_h - number of hidden layers, num_iterations - the number of times we run the network for improving performance(i.e. parameters) and print_cost which indicates whether we have to print the computed cost by every 1000 iterations or not.

nn_model then first calls initialize_parameters which takes the size of input(total number of features in training data i.e. 22) = n_x, size of hidden layers (this can vary from 25 to 5) and the size of output layer(total neurons in output layer = 1) = n_y. initialize_parameters returns the random weight matrices and bias matrices in a dictionary called parameters.

nn_model then starts a loop from 0 to num_iterations and calculates the cost using forward propagation and then calls the backpropagation algorithm to improve the weights and bias to reduce cost. Forward_propagation calculates the output using equations z1 = w1 * x + b1, A1 = tanh(Z1), z2 = w2 * a1 + b2 and A2 = sigmoid(Z2)

After forward propagation we calculate the cost by comparing our obtained output and given output for training data using the cost function.
Then we call backward_propagation and update_parameters to change the weights and bias so that our cost is reduced.

Then at last we call “predict” (parameters, X) to use the parameter dictionary and the test input - X to calculate test output.

III. Linear discriminant analysis and XGBoost in Python

Linear discriminant analysis are machine learning algorithms used for classification. They assume gaussian distribution of the features. They can be used for multiclass classification as well. In our case, we need to classify in two classes.

Our main goal is to reduce overfitting, to avoid underfitting and to choose a generalized model which is accurate for new test examples. To test accuracy of the fitted model we apply cross validation techniques like k-fold cross validation and leave-one-out cross validation. We also research on ensemble learning to improve the accuracy of the fit.

XGBoost (Extreme gradient boosting) is a gradient boosting algorithm. The principle used in XGboost is similar to the working of gradient boosted trees which uses additive training approach to boost the performance of the algorithm. The model uses combination of Regression and classification trees (CART). In such models the leaf nodes are associated with additional weights of positive classification known as leaf scores. The ultimate respective leaf score of the ensemble tree is the respective sum of the leaf scores in set of trees under consideration. We need to find set of trees that optimizes the leaf scores. What additive learning does is to add up one tree at a time, and to add up one new tree at another time.

$T = (t_1, t_2, t_3)$ where T represents the whole tree with three subset trees
$X_1 = x_1(t_1) + x_1(t_2) + x_1(t_3)$ where X_1 represents one feature of T and $x_1(t_1)$, $x_1(t_2)$, $x_1(t_3)$ represents features of tree t1, t2, t3 respectively.
Similarly, $X_2 = x_2(t_1) + x_2(t_2) + x_2(t_3)$ and $X_3 = x_2(t_1) + x_2(t_2) + x_2(t_3)$

This can be mathematically written as,

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

Where, K is the number of trees, f is a function in function space F, and F is a subset of all possible CARTs.

In order to find the combination of trees which optimized the leaf scores we add one new tree to the set at a time. The tree which we add should optimize the objective function.

$$\text{Obj(t)} \quad = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))$$

Where t is tree which is added, and 'l' is the cost function like MSE (mean square error)

IV. Random Forest in R

Random Forest is a type of classification and regression algorithm which can be implemented with the enormous data. It is an ensemble learning method where two or many models are generated and merged to solve a specific computational analytical problem. Firstly, the tree structure of the titanic project would be,

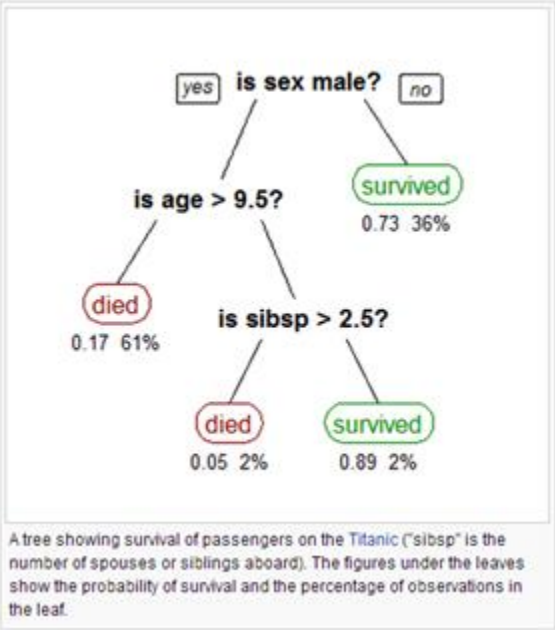


Figure.2 Decision Tree structure of Titanic survival representation

The data depicts that 36% of passengers are female and they are likely to have higher survival rate as 0.73. By falling into this category, she is likely to be survived and her survival status is predicted as 1. Then we will be narrowing down through many subcategories to predict the survival status and make them into small groups. In the random forest model, we generate N number of trees. This N value can be between few hundreds to several thousands. The resulting N trees are said to be the random forest. The average prediction of all the decision trees is the prediction of the random forest.

Instrumentation

Excel built-in function:
IF function

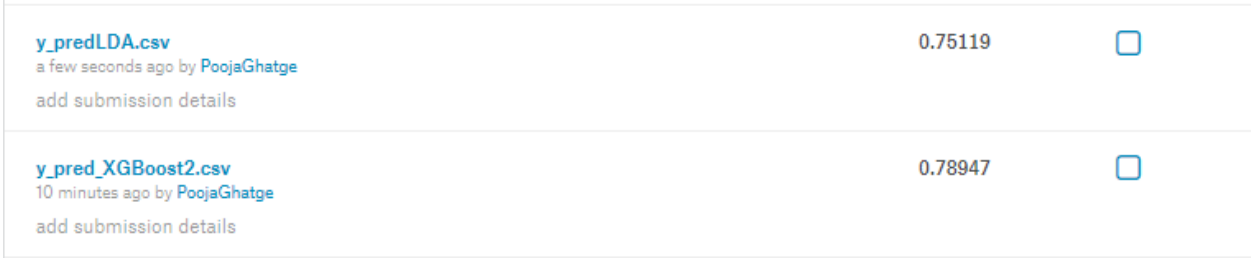
Python built-in library:
Pandas (read the files), NumPy (create N-dimensional array object), Scikit-Learn (machine learning), Time (create random numbers)

Machine Learning Algorithms:
Decision Tree classifier, Random Forest classifier, Binary Tree, Neural Networks

Results

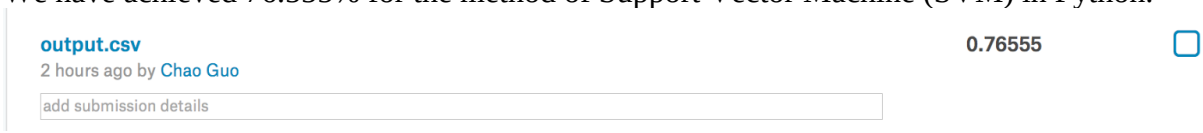
The following results display the scores by each methodology returned on Kaggle's website.

We have achieved 78.95% for the method XGBoost and 75.12% for the method LDA in Python.

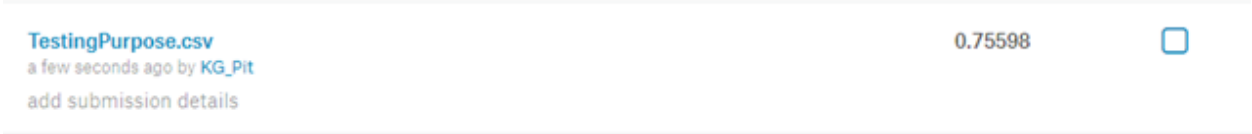


We have achieved 77.251% for the method of Artificial Neural Network (ANN) in Python.

We have achieved 76.555% for the method of Support Vector Machine (SVM) in Python.



We have achieved 75.598% for the method of Random Forest in R.



Conclusion

LDA works best when the features follow gaussian normal distribution. Most of the features like the ticket number, Q, C, S were discrete and didn't follow Gaussian distribution which resulted in only 75% accuracy. Whereas, XGBoost gave comparatively more accuracy because it iteratively adds the best ensemble model which optimizes the objective function to the additive training model. Accuracy can be further improved if we figure out, how is cabin related to the survival rate and how to handle missing values in it.

The accuracy of neural network is average because of less training data. For neural networks to be successful the training data should be large. Hence, we get only 77% accuracy since we only have 891 training examples. We tested the network by changing various parameters - learning rate varying from 0.01 to 0.1, number_of_iterations varying from 10000 to 500000 and hidden layers varying from 22 to 5.

One of the common issues in all tree-based algorithms is overfitting. If the tree is too large, it might not have an ability to predict the correct output. The most important factors affecting the performance of Random Forest are number of trees, and number of observations.

Although our team spends plenty of time dealing with data cleansing, there are reasons to believe that inappropriate procedures of data cleansing, such as simply assigning missing values as one unique number, mainly lead to comparatively low accuracy (less than 80%) other than methodologies themselves. We attempt to create categorial features to replace missing values, but the outcomes do not prove us positively. Therefore, the "Limitations" section briefly describes the obstacles we face and the challenge we can conquer to achieve higher accuracy.

References

Coursera Inc. (2017, December 3). Neural Networks and Deep Learning. Retrieved from Coursera: https://www.coursera.org/learn/neural-networks-deep-learning

Data Science Central. (2017, December 3). Introduction to Classification & Regression Trees (CART). Retrieved from Data Science Central: https://www.datasciencecentral.com/profiles/blogs/introduction-to-classification-regression-trees-cart

DMLC. (2017, December 3). Introduction to Boosted Trees. Retrieved from XGBoost : http://xgboost.readthedocs.io/en/latest/model.html

Kaggle Inc. (2017, December 3). Titanic: Machine Learning from Disaster. Retrieved from kaggle: https://www.kaggle.com/c/titanic

scikit-learn developers. (2017, December 3). Support Vector Machines. Retrieved from scikit-learn: http://scikit-learn.org/stable/modules/svm.html

Support vector machine. (2017, December 3). Wikipedia. Retrieved from The Free Encyclopedia.: https://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=813219411

Appendix I

First 28 rows in the table of Data Cleansing used in SVM, ANN and Random Forest.

Passenger Id	Pclass	male	female	Age	SibSp	Parch	Fare	Q	C	S	Survived
1	3	1	0	22	1	0	7.25	0	0	1	0
2	1	0	1	38	1	0	71.283	0	1	0	1
3	3	0	1	26	0	0	7.925	0	0	1	1
4	1	0	1	35	1	0	53.1	0	0	1	1
5	3	1	0	35	0	0	8.05	0	0	1	0
6	3	1	0	28	0	0	8.4583	1	0	0	0
7	1	1	0	54	0	0	51.862	0	0	1	0
8	3	1	0	2	3	1	21.075	0	0	1	0
9	3	0	1	27	0	2	11.133	0	0	1	1
10	2	0	1	14	1	0	30.070	0	1	0	1
11	3	0	1	4	1	1	16.7	0	0	1	1
12	1	0	1	58	0	0	26.55	0	0	1	1
13	3	1	0	20	0	0	8.05	0	0	1	0
14	3	1	0	39	1	5	31.275	0	0	1	0
15	3	0	1	14	0	0	7.8542	0	0	1	0
16	2	0	1	55	0	0	16	0	0	1	1
17	3	1	0	2	4	1	29.125	1	0	0	0
18	2	1	0	28	0	0	13	0	0	1	1
19	3	0	1	31	1	0	18	0	0	1	0
20	3	0	1	28	0	0	7.225	0	1	0	1
21	2	1	0	35	0	0	26	0	0	1	0
22	2	1	0	34	0	0	13	0	0	1	1
23	3	0	1	15	0	0	8.0292	1	0	0	1
24	1	1	0	28	0	0	35.5	0	0	1	1
25	3	0	1	8	3	1	21.075	0	0	1	0
26	3	0	1	38	1	5	31.3875	0	0	1	1
27	3	1	0	28	0	0	7.225	0	1	0	0
28	1	1	0	19	3	2	263	0	0	1	0

Appendix II

First 23 rows in the table of Data Cleansing used in LDA and XGboost.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Pclass	Age	SibSp	Parch	Fare	Ticket	Q	C	S	male	female	Survived
2	3	22	1	0	7.25	521171	0	0	1	1	0	0
3	1	38	1	0	71.2833	17599	0	1	0	0	1	1
4	3	26	0	0	7.925	23101282	0	0	1	0	1	1
5	1	35	1	0	53.1	113803	0	0	1	0	1	1
6	3	35	0	0	8.05	373450	0	0	1	1	0	0
7	3	28	0	0	8.4583	330877	1	0	0	1	0	0
8	1	54	0	0	51.8625	17463	0	0	1	1	0	0
9	3	2	3	1	21.075	349909	0	0	1	1	0	0
10	3	27	0	2	11.1333	347742	0	0	1	0	1	1
11	2	14	1	0	30.0708	237736	0	1	0	0	1	1
12	3	4	1	1	16.7	9549	0	0	1	0	1	1
13	1	58	0	0	26.55	113783	0	0	1	0	1	1
14	3	20	0	0	8.05	52151	0	0	1	1	0	0
15	3	39	1	5	31.275	347082	0	0	1	1	0	0
16	3	14	0	0	7.8542	350406	0	0	1	0	1	0
17	2	55	0	0	16	248706	0	0	1	0	1	1
18	3	2	4	1	29.125	382652	1	0	0	1	0	0
19	2	28	0	0	13	244373	0	0	1	1	0	1
20	3	31	1	0	18	345763	0	0	1	0	1	0
21	3	28	0	0	7.225	2649	0	1	0	0	1	1
22	2	35	0	0	26	239865	0	0	1	1	0	0
23	2	34	0	0	13	248698	0	0	1	1	0	1