Surviving the Titanic - Kaggle Group Write-Up

Alec Guertin, Michael Ball, Peter Sujan, Alex Caravan
Team Draw Me Like One of Your French Girls

April 2, 2015

1 Introduction

As part of our process to find a good model for predicting which passengers survived the Titanic, we worked on optimizing our results for several different methods. We developed multiple models using support vector machines, random forests, neural networks and AdaBoost. We started by improving these models individually and then moved to combine our results in an ensemble approach. In our final approach, we also incorporated some preprocessing techniques and included some new features to improve the accuracy of our analysis. This report outlines in detail our approach for each of these steps and explains the reasoning behind our decisions as well as our findings.

2 Preprocessing and New Features

We added several new features in this final step. We wanted to incorporate more of the source data. However, some of these features were not conducive to the models that we were training. We modified the existing data for the fare, cabin and home.dest features to create new features to train our model on. We also created new features from predictions.

2.1 Cleanup

In all of our alanysis, we dropped both the "Name" and "Ticket" columns because these values were unique to each person on the Titanic.

2.2 Fare

We noticed that the range of ticket fares had large gaps. We made a histogram of the values and discovered that it had a large tail extending to the right (higher ticket prices). We researched methods to eliminate skew in data and found that using a variable transformation such as Box-Cox [1] is simple and effective. We used $\log(fare)$ to approximate the Box-Cox transformation. We realized, however, that the range of values for the fare feature included zero so we settled on modifying this transformation to $\log(fare + 1)$ to ensure valid values. This transformation helped to significantly mitigate the negative effects of skew in our analysis.

[1] http://onlinestatbook.com/2/transformations/box-cox.html

2.3 Cabin

We initially had trouble including the values of the cabin feature in our analysis because of the lack of uniform representation. Multiple sections of the same deck are sometimes listed for a single person. The raw values for the same deck may be different for each occurence. We decided to make a new feature by simplifying the values of the cabin column. We took the prefix of each value instead of the whole string. For example, "B6 B5 B20" and "B4" would both become simply "B". These values both refer to the same deck, but would have been treated as different values in the original model. This modification helped analysis with the cabin values, since there was very little grouping originally. Our new feature only contained the values "A" through "G" where as the original feature contained 184 non-empty entries with 183 unique values in the training set.

2.4 Home.Dest

We were faced with a similar problem regarding the *home.dest* feature. The values for this feature contained too much variation to analyze groups. We decided to transform this feature by grouping values. After examining the values of *home.dest*, we decided to create three groups. The first group includes all entries with values that contain the strings "London" or "England". The second group includes the values containing "NY". And the last group contains entries that do not fit into either of the previous categories. Having only three possible values for this feature was more conducive to training our model.

2.5 Predictions

During our individual work and early group work, we saved the values of our predictions after training several different models. We concatenated the values of the survival predictions to make a set of new features to train on. Our intention was that our final model would learn from the predictions of other models as well as our more conventional set of features.

3 Models

We used the following models in our predictions. Each of these four models was used in both the ensemble methods we tried. For each individual learning method, we optimized various parameters, using cross-validation to estimate test error.

3.1 SVMs

Our support vector machines were all trained using the `e1071` [2] (`R`) package. We tried two different kernels: radial and polynomial. For each one, we optimized the cost parameter C. For the radial kernel we optimized *gamma*, and for polynomial we optimized the degree d. The grid search over these parameters gave $C = 10$ for both kernels, $gamma = 0.1$ for radial, and $d = 2$ for polynomial.

[2] http://cran.r-project.org/web/packages/e1071/index.html

3.2 Random Forests

We trained our random forests in `R` using the `randomForest` package. The only parameter to optimize was the number of decision trees to grow, and the optimal value turned out to be 1500.

3.3 Neural Nets

For our neural nets, we used the `nnet` package. We trained nets with one hidden layer, and used CV to choose the number of hidden nodes. The optimal number was 5.

3.4 AdaBoost

For adaboost, we used the `ada` package and optimized over the shrinkage parameter *nu*. The optimal value of *nu* was 1/2.

4 Ensemble Methods

We tried several different ensemble methods during our work to see which would give us the best result. Below we describe some of our findings with each method we tried.

To get a rough baseline for our ensemble methods, we first tried ensembling using only our models from the individual phase of the project. Our approach here was to concatenate the previous predictions into one matrix, and use logistic regression as a simple way of "averaging" these predictions. This gave an accuracy of around 76%, which was actually slightly worse than some of our individual predictions.

Our next step was to incorporate our new features. We started by adding these to the ensemble matrix of previous predictions. This, however, did not really make a difference - adding the new features on top of the already-trained models did not seem to add any information. Instead, we retrained the four models from above with our new features included in the original training data matrix. Then, we tried two ensemble methods on the matrix of these predictions: First, we tried a simple logistic regression model as before. We then tried an SVM, once again optimizing over C and *gamma*. Both ensemble methods achieved approximately 79% test error, which was better than any of our previous individual submissions.

5 Final Results

Though we were able to slightly improve over our individual submissions, we ended up fairly low on the leaderboard. Possible ways to improve our results would be to consider more models in our ensembling phase, or improve our parameter tuning. In addition, creating more complicated feature transformations might improve accuracy.