Final Project Report:
Titanic: Machine Learning from Disaster

Fengfei Zheng, Ming-hung Shih, Shotaro Matsui

December 3, 2017

1 Introduction

"Titanic: Machine Learning from Disaster" is an active Kaggle competition on `http://www.kaggle.com/titanic`.

The purpose of the problem is to predict what sorts of people were likely to survive, using their personal information. So this problem is a binary classification problem. The output is $\{0, 1\}$, whether survive or not, and the input is features listed below:

pclass Ticket class

name passenger's name with title

sex sex

age Age in years

sibsp # of siblings and spouses aboard the Titanic

parch # of parents and children aboard the Titanic

ticket Ticket number

fare Passenger fare

cabin Cabin number

embarked Port of Embarkation

There are 891 training examples (with survived/not-survived label) and 418 test data (without that label), so 1309 data for total. The survival rate on training set was about 38%.

2 Existing Approach

Since this is a classical binary classification problem, this problem seems to be a practice problem or playground to study machine learning basics. So there are nice tutorials on the overview section on the Kaggle web page, and also countless Kernels which we cannot check all of them. So we decided to choose some Kernels and go through, combine methods or try new methods so that we could improve the accuracy. The Kernels are on the discussion board on Kaggle.

3 Our Approach

3.1 Data Understanding

We start by tabulating the statistics of the data We can see that some features have missing values. This

	Age	Fare	Parch	PassengerId	Pclass	SibSp	Survived
count	714.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	29.699118	32.204208	0.381594	446.000000	2.308642	0.523008	0.383838
std	14.526497	49.693429	0.806057	257.353842	0.836071	1.102743	0.486592
min	0.420000	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000
25%	20.125000	7.910400	0.000000	223.500000	2.000000	0.000000	0.000000
50%	28.000000	14.454200	0.000000	446.000000	3.000000	0.000000	0.000000
75%	38.000000	31.000000	0.000000	668.500000	3.000000	1.000000	1.000000
max	80.000000	512.329200	6.000000	891.000000	3.000000	8.000000	1.000000

Figure 1: Data description

problem can be solved later by a technique called imputation.

We can plot a heat map showing the relative importance of the features present in the dataset.

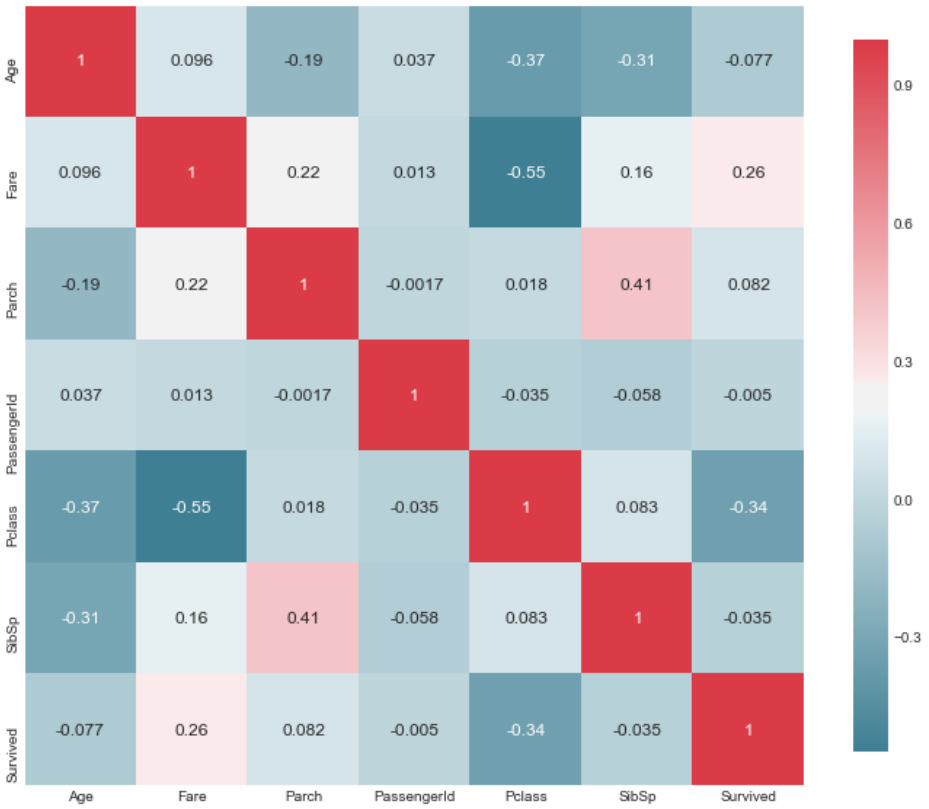

Figure 2: Heat Map

This map will be very useful when we later decide on the kinds of features we want to include in our model.

3.2 Feature analysis/engineering

3.2.1 Feature Preparation

outliers

At first, extracted outliers. For each numerical value features (age, sibsp, parch and fare), compute interquartile range (IQR), and ommit examples which has a value outside of this range. There are 10 of outliers. So the total number of data (to analyze) is 1299.

missing values

Also, there are some missed data in the data-set and we should predict them with corresponding data.

pclass(categorical)

Categorical value that ranges 1st to 3rd class. The higher the class, the more likely to survive.

Name(categorical)

Passengers' names contains their titles, which may affect survivals. Besides titles to differenciate genders (i.e. Mr., Mrs., Mlle, etc.), there are some titles like Master., Dr., or else. People who has those titles were more likely to survive. So extracting those titles and form a new categorical value
$(Male : Mr.)$
$(Female : Miss., Ms., Mrs., Mme., Mlle)$
$(Master : Master.)$
$(Other\text{-}titles : Lady., Countess., Dr., Capt., Rev., Col., Don., Major., Sir., Jonkheer., Dona.)$
The reason why Master is differenciated from Others-titles is because it was much more frequent than others (Master:62, Other-titles:29 for total), and also Masters. was more likely to survive than Other-titles passengers.

sex(categorical)

Female is more likely to survive. Actually, 74% of female is survived while male survival rate is only 19%.

age(numerical)

By comparing survival and non-survival group, very young (around 0-5) children are more likely to survive, and very old (around 60-80) people are less likely to survive.

And there are 256 missing values, so estimated them from other numerical features, sibsp, parch and pclass. Filled missing ages using median of rows which have same(sibsp, parch and pclass). If there is not such row, fill with median age.

sibsp, parch(numerical) $\rightarrow$ fsize(categorical)

Both of the value has similar effects on survival. Passengers who has middle size (1-2) of sibsp/parch are more likely to survive. So combined these two features and introduced a new feature "fsize (family size)", which is $(fsize) = (sibsp) + (parch) + 1$, where $+1$ means the passenger himself. And then turned the numerical value to categorical value, $Single = 1, SmallF = 2, MedF = (3,4), LargeF = (5, \cdots)$.

ticket(categorical)

The ticket number would contains information about pclass or cabin. As shown in pclass section, higher class passengers are more likely to survive. So, to extract that information, using only prefix of ticket number and dropping its numbers. Some ticket number are duplicate and it shows that this passenger took Titanic with others. We tried mark the passenger whose ticket number are the same as a group and no duplicate number as the other group to predict their survival rate.

fare(numerical)

There is one missing value, so decided to fill it with midian value. The distribution of fare is ranged around 0-500, however most of passengers are around between 0-25, so decided to use $log(fare)$ value to differenciate those values more precisely.

cabin(categorical)

1007/1299 of passengers do not have this value (missing value), which would mean that they did not have cabins. So introducing new category "X" to express that. Also omitting cabin number from the prefix (e.g. C123 $\rightarrow$ C) to reduce unnecessary dimentionality.

embarked(categorical)

The two missing values are filled out with the most frequent value, S (C=Cherbourg, Q=Queenstown, S=Southampton). The place of embarkment is related to pclass. Indeed, the survival rate was $C > Q > S$, and over half of S and Q embarked passengers are 3rd class, while half of C passengers are 1st class.

age*pclass(combination)

To further boost our accuracy, we added some combination features to make the data more separable. This gave significant improvement over the version without it, by a Kaggle ranking of about 4000.

3.2.2 Feature Selection

With all the data preparation, we end up having a lot of features, which can be bad for multiple reasons, such as slowing down the training, overfitting, etc. We tried to select the most important features for training, and had 12 features in the end.

3.3 Hyperparameter Tuning

Since there are numerous parameters to tune for each classifier we tried, we used some automation to find out the optimal parameters for the individual classifiers, improving the efficiency. The specific methods we used are grid search.

4 Results and Discussion

4.1 Cross-Validation

To evaluate the result before submitting to Kaggle, we tried cross validation and found the error rate is lower to 8% because of overfitting. With the submission limit of Kaggle, we should do more data analysis to make the cross validation more meaningful.

4.2 Analysis conclusion

The most effective features were sex, fare and name (title). Female, very young children, people who has Social status (title) and having higher ticket class are likely to survive. Male, old people, and lower class ticket holders are less likely to survive. It meets intuition that female, children and socially important people are priorly evacuated.

4.3 Individual Classifier

4.3.1 SVMs

Linear SVM For linear kernel, with different C it took about 1 minute to complete 891 training data. The error rate is about 74% to 77%.

Quadratic SVM For quadratic kernel, with different C it took about 5 minutes and the error rate is about 74% to 78%.

Gaussian SVM For gaussian kernel, with different C it took about 0.1 minutes and the error rate is about 60% to 74%. (too long)

4.3.2 Other Advanced Classifiers

We also tried GradientBoosting, Kneighbors, GaussianNB, LogisticRegression, etc. There's nothing really impressive befor doing feature engineering and model tuning (At first we ranked about 7800 without tuning).

Using the RandomForestClassifier, and with the 12 selected features, we performed grid search to determine the optimal parameters for it. This gave us significant improvement on accuracy, and we achieved 0.894060995185 on trainning and 0.787313432836 on dev.

We submitted it to Kaggle and got a test accuracy of 79.9%, which is a ranking of about $1700/9550$.

4.4 Voting Classifier

It turns out that voting classifer didn't perform as well as the individual classifer. Using scikit-learn tools, trained five different algorithms, SVM (SVC) with gaussian kernel ('rbf'), Random Forest, Extremely Random Tree, Ada Boosting and Gradient Boosting with Grid Search function to find the best tuned ones. The result was that all of these are around 82-83% accuracy on cross-validation, so constructed a voting classifier using VotingClassifier() function (with 'soft' option) and got 92% cross validation accuracy.

The Kaggle score was 78% accuracy, which was $4013/9550$. Some other approaches like not omitting outliers, or combining Masters. and Other-titles in the name feature, did not help improving both cross-validation score or test score (Kaggle submission).

Also we have to state the top rankers on Kaggle are 100% accuracy, which is doubtful. Probably because the test set has only 418 examples, they could got that accuracy?

5 Conclusion

We tried to solve a basic binary classification problem, and found out that the most part of efforts on machine learning is feature engineering. How to transform and treat training examples is difficult problem, especially if training takes very long time, because we would not be able to check results so frequently. So we would have to try figuring out what relationship of cause and effect is behind features (and label) before actually trying learning methods. Overall, our results seemed pretty good, since we are ranked among the top 20 % using the best model prediction we had. We should also state that the predicted values change every time we run the program, so it might or might not generate the best results when run a single time.