Comparative Analysis of Data Mining Algorithms with Titanic Survival Dataset

By Hang Jiang, Qinyi She, Wenyun Fan

I. Motivation and Contributions

Motivation

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912 during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and eventually led to better regulations for ships.

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there sure existed some element of luck involved in surviving the sinking, some groups of people might be more likely to survive than the others, such as women, children, and the upper-class. So exactly what sorts of people were likely to survive? In this project, we want to investigate the factors that lead to their survival. We would apply several tools of data mining algorithms to predict which passengers survived the tragedy. The result of our predictions can become a good source to improve the survival rate of passengers and crew.

Objective

The first goal of this project is to investigate the factors behind surviving in this tragedy. We would like to use the training data set to build either a tree or a model, and then use the test data set to predict whether people with certain features survived the tragedy. Moreover, to learn more about our result, we will look into the trees and models we built to find the factors that contribute more to the survival of passengers and crew.

The second goal is to implement or improve and compare different algorithms that we learned in this class. Besides, we are interested in how each algorithm can be applied to analyze this data set, what are the efficiency and accuracy of these algorithms comparing with others, and why is an algorithm better than the other. Eventually, we will choose the top 5 classifiers and analyse the pros and cons of each. In general, we aim to understand deeper about how the algorithms work from a more applicational perspective.

Contribution

Hang Jiang implemented perceptron and K-modes clustering algorithms; Qinyi She implemented KNN and Naive Bayesian algorithms; Wenyun Fan implemented decision tree algorithms; and together, we did comparative analysis of several machine learning algorithms we have learned in class and even applied more models from online libraries, to find the best models for this data set.

II. Approach and Methodology

Data Preprocess

The dataset involved in this project records the passenger information of RMS Titanic in the tragedy. The Titanic dataset obtained from Kaggle includes a training dataset with 891 instances and a test dataset with 419 instances. The entire dataset contains 12 variables with 4 numeric attributes and 8 categorical attributes, including passenger ID, survival (the class attribute), pclass (the class of ticket), name, sex, age, sibsp (number of siblings or spouses aboard the Titanic), parch (number of parents or children aboard the Titanic), ticket (ticket number), fare, cabin, embarked (port of embarkation).

To begin with, we evaluate each attribute's importance and their potential influence in this classification problem. The attribute passenger ID is dropped because it is simply a row number without special meanings. Moreover, the attribute ticket that records the ticket number of each passenger is also dropped because it is unique for each passenger and it is hard to understand the meaning behind it.

The attribute name seems to be irrelevant information at first. However, the name includes the title of the passenger, which might be an interesting factor in the classification problem. Hence, we replace the attribute name by a new attribute title and group the titles that rarely appears (Lady, Countess, Capt, Col, Don, Dr, Major, Rev, Sir, Jonkheer, Dona) in the dataset as Rare, and group Mlle and Ms together as Miss. The resulting title attribute has five categories including Mr, Miss, Mrs, Master, and Rare.

Then, we work on the missing values in this dataset. We found that the attribute cabin has missing values in 687 out of 891 records of the training dataset. Because of the assumption that the higher the fare, the better the cabin, the attribute fare is a good indicator of the cabin attribute. Hence, the attribute cabin is dropped. Moreover, we have missing values in age and embarked. We fill the missing values using the average for numeric attribute age, and the mode for categorical attribute embarked.

Because most of the attribute are categorical, for simplicity, we transform the numeric attributes into categorical attributes. Using equi-width binning, we separate age into five categories and fare into four categories, where the number of bins is determined by the existence of a significant change in survival rate between different bins. For attributes sibsp and parch, we convert them into binary categories with 1 indicating has sibling or spouse and parent or children, and 0 indicating no sibling or spouse and parent or children. The nominal attributes sex and embarked are also converted to ordinal-like numbers using integers.

In addition to these preprocessing techniques, we also generate another table using one hot encoding. For each attribute, a new column is created for each category so the entries in the table can only be 0 or 1. This technique is suggested to improve the accuracy of classification because it equalizes the distance between a pair of categories.

Model Building

In this project, we compared the performance of 11 data mining algorithms. First, we implemented some models we learned in class as well as sourcing from Scikit library. Then, we picked the top performing models and tuned them to see which one performs the best on the test set.

In algorithm implementation, specifically, we implemented KNN, Decision Tree, Naive Bayes, perceptron and K-modes+KNN. The Decision Tree model is improved by adding a pruning function. Perceptron is implemented using gradient descent. K-modes+KNN is to use K-modes to cluster points into K groups and predict the label of a new point by comparing the distance between the new point and the centroids of the clusters. Those models, though found to perform reasonably, were later found to be less robust than the models from Scikit library.

Among algorithms sourced from Scikit library, most models can be applied with their default parameters, while some models need to be tuned to perform better. For example, to optimize the Random Forest Classifier, we first select only the important features for the model. To do so, two features out of 26 features in the one-hot encoded data are abandoned. We then used grid search to decide the the max_depth, n_estimators and criterion parameters of the model in order to find the optimal model. Grid search approach is also used in K-Neighbors Classifier to find the best K.

III. Evaluation and Results

We first evaluated our implementations by directly submitting our predictions to the Leaderboard. The table 1 below shows that KNN and DecisionTree C4.5 are the best classifiers among our implementations. This is reasonable because the only tricky part of KNN is to decide K's value. For two DecisionTree models, we added pruning and tuned the threshold, so they also gave us good results. However, other models need more optimizations. For example, Kmodes+KNN can be optimized by bettering the initialization method similar to KMeans. Perceptron can also be optimized by adding regularization or changing the Gradient Descent to Stochastic Gradient Descent. We didn't have chance of implementing all the optimizations for individual implementation. As a result, we used models from Scikit Library to pursue better performances.

Classifier	Kaggle Submission Score (accuracy)
KNN	0.78469
Decision Tree Gain Ratio	0.78469
Decision Tree Information Gain	0.77033
Multinomial Naive Bayers	0.74641
Perceptron	0.74641
Kmodes with KNN	0.73206

Table 1: Kaggle Submission Score of our Implementations

For models sourced from Scikit Library, they are first evaluated locally by accuracy using 10-fold cross validation on the two training sets (categorical and one-hot encoded datasets) shown below (Table 2). It is worth noting that Gaussian Naive

Bayesian is Naive Bayesian model based on Gaussian distribution. It does not have a performance on one-hot encoded data because the model does not take sparse vectors (for example, the one-hot encoded vectors) as input.

	Accuracy % (categorical)	Accuracy % (one-hot)
MLP Classifier	**79.48**	78.58
Logistic Regression	79.80	**81.27**
SVC	**81.03**	79.34
K-Neighbors Classifier	81.38	**81.83**
Multinomial Naive Bayesian	76.78	77.56
Gaussian Naive Bayesian	74.88	--
Perceptron	64.10	69.38
SGD Classifier	70.62	72.83
Decision Tree Classifier	80.82	**81.04**
Random Forest Classifier	80.71	**82.60**

Table 2: Accuracy by 10-fold Cross Validation of Scikit Models

Comparing the performances of all the classifiers we built, we picked top 5 classifiers. The top five classifiers are Random Forest Classifier (82.6%), K-Neighbors Classifier (81.83%), Logistic Regression (81.27%), SVC (81.03%), and MLP (80.48%). We did not choose Decision Tree algorithm even it has a better score than MLP because Random Forest algorithm is the same type of classifiers as Decision Tree algorithm. After applying those classifiers to the test data, we submitted the predictions to Kaggle Leaderboard to obtain the performance, which is evaluated by the percentage of passengers correctly predicted as accuracy. The table 3 below shows the accuracy score of each classifier. **The best score achieved by SVC algorithm places 987th out of all 6912 submissions, top 15%.**

Classifier	Kaggle Submission Score (accuracy)
SVC	0.80383
Random Forest Classifier	0.78947
KNeighborsClassifier (KNN)	0.78947

Multiple-Layer Perceptron (MLP)	0.77512
LogisticRegression	0.75598

Table 3: Kaggle Submission Score of Top 5 Models Selected from Table 2

Although Random Forest Classifier has the best score on the training dataset, it probably overfits the training dataset and does not scale well to the test dataset. The overfitting issue is even worse on MLP and Logistic Regression because the two algorithms both make use of gradient descent to update the weights. Also, MLP and Logistic Regression are unstable in performance because the order of instances to train actually makes a difference in the final separation line drawn in the dataset. SVC, on the other hand, is very stable because it needs all the training data to draw the global optimal separation line while MLP and Logistic Regression usually rest at some local optimal solution.

For KNeighbors Classifier, the difference in scores on training set and dataset also suggests an overfitting of the model to the training set. There are two things that result in the bad performance of KNN. First, It is hard to decide the best K value for the model. The best K for the training set does not guarantee a robust model for test data. Second, KNN only performance well when we choose the right bucket of features. Obviously, the features we chose here are not the best for KNN.

IV. Conclusion and Future Work

In this project, we were able to complete a data mining task from data preprocessing to selecting the best performing data mining algorithm. It provides a chance for us to apply the concepts and techniques we learned in class including data preprocessing techniques such as filling in missing values and feature selection, and data mining algorithms such as decision tree, random forest, naive bayesian classifier, KNN, SVM and perceptron.

Moreover, during data preprocessing, we learned more about how to perform feature selection by understanding the underlying information of each attribute and to be able to extract only the useful information from the attribute in order to replace it with a more concise new attribute (e.g. the transformation from name to title in this project).

In addition, we had the chance to implement the optimization algorithms discussed in class, and through the implementation process, we had a better understanding of the performance of each algorithm. By sourcing online libraries to apply more advanced algorithms to this dataset, we got familiar with other popular classifiers such as neural network.

To further improve the result, there are two major aspects can be worked on. First, more features can be extracted from the dataset. According to the advice given by other participants, creating more features from existing dataset can improve the result of classification. For example, a new attribute created by multiplying sibsp and parch is suggested to be an useful feature in classification. Furthermore, instead of transforming all numeric attributes to categories as in this project, manipulating the algorithm so that it is able to process numeric and categorical attributes together might further improves the result of classification due to the reduction of information loss.