Predicting Disaster

Kathryn Beck, Lindsay Jacobs, Nicole Manno, and Becca Shipper

Abstract—Kaggle's "Titanic- Machine Learning from Disaster" competition provides an interesting challenge of accurately predicting the survival of passengers aboard the historic Titanic, while using a data set with many missing values. We utilize techniques of feature engineering and data cleaning in order to remedy the missing cabin, age, and embarked values. For our model, we first compare the performance of eight different machine learning models. We use cross-validation scores to decide on features to drop that have low correlation with the results. Lastly, we hypertune the four top-performing models by applying a gridsearch in order to determine the hyper-parameters that result in the strongest model.

1 Introduction

The sinking of the Titanic is one of the most infamous historical disasters to date. On April 14-15, 1912, the Royal Mail Ship Titanic, or Titanic for short, sank during its voyage from Southampton, England to New York City due to the collision of the Titanic with an iceberg. Of the 2,200 people onboard the ship, it is estimated that 1,500 passengers and crew members had died. It is presumed that the massive death toll was due to the Titanic's lack of lifeboats, being only 20. Furthermore, of those 20 lifeboats, many were only half-full or sometimes even less. As a result of this tragedy, many laws were created that protected the lives of passengers and crew members on seas [1].

More than 100 years later, surviving the sinkage of the Titanic has become an interesting concept. Kaggle, an online community of data scientists and machine learning practitioners, has a data set called "Titanic- Machine Learning from Disaster" that contains passenger details and survival information accessible to its users [2]. Based off the data set provided by Kaggle, we want to predict survival of a given passenger. To do this, we will first evaluate and visualize the data set for better understanding and to see which features have the largest impact. Then, we will use eight distinct classification algorithms with their default parameters to create eight models that predict survival. Once the models have been created, we will then select the top four performing models and adjust their parameters to improve accuracy.

2 Related Work

In addition to Kaggle offering data sets for their users to explore and build models using data science and machine learning techniques, Kaggle offers competitions. Users are scored on the accuracy of their model; the higher the accuracy, the higher they rank on the scoreboard. The Titanic data set is one of many competitions offered by Kaggle. However, it is an extremely popular one as it is considered to be a beginner-friendly competition. With that being said, a lot of models have been created to predict survival on the Titanic. On the leadership board, there are about 100 competitors who have created a perfect or near-perfect model for predicting survival.

IEEE Demo Template for Computer Science Journals was used in this article.

Many competitors have written reports, blogs, and how-to guides, detailing their methods and experiment that they submitted to the Kaggle competition. We built off of some of this previous work, particularly in our process of data cleaning and feature engineering. A common theme in previous work is to focus on novel designs for feature engineering, as well as how to appropriately clean the data set, as there is a large amount of data missing. We attempted similar techniques to fill in the missing data as discussed in [5] and [7]. We also referred to [6] to give ideas for our initial analysis of the data set.

3 Data

Creating a model for a data set requires understanding the data and the different features. For our data set, we are attempting to accurately predict if a passenger survived the sinking of the Titanic. The data set provided through Kaggle is already split into two subsets of testing and training data. As the label is meant to be predicted by the final model entered into the competition, the testing set does not have the Survival feature. However, we decided to work only with the training data, and split it into new training and testing data in order to use supervised learning techniques to develop our model. For the remainder of the paper, when we refer to the data set, we are referring to the original training data set provided by Kaggle.

First, we notice that our data set includes 891 passengers, where 549 of the passengers had died and the remaining 342 had survived. In Table 1, you will notice the different features that the data set has, which will be used to create our models.

When analyzing the percentages of survival based on a specific feature, we observed that class, sex, and age seem to influence a passenger's survival. In Figure 1, notice that children under the age of ten had a much larger chance of surviving than if they were older. Furthermore, Figure 2 provides even more detail by stratifying the data in Figure 1 by sex. The dramatic difference in the two bar graphs in Figure 2 implies that an individual was more likely to survive if they were female. Next, Figure 3 depicts the survival rate by class. Notice that class 3 has a large death rate. This aligns with our intuition that wealthier individuals were the more likely to survive based on cultural norms and privilege.

TABLE 1
Titanic Data Set Features

Feature	Definition	Key
survival	Survival	0 for No and 1 for Yes
pclass	Ticket class	1 is 1st, 2 is 2nd, and 3 for 3rd
sex	Sex	N/A
age	Age in years	N/A
sbsp	Number of sibilings & spouses aboard	N/A
parch	Number of parents & children aboard	N/A
ticket	Ticket number	N/A
fare	Passenger fare	N/A
cabin	Cabin number	N/A
embarked	Port of Embarkation	C for Cherbourg, Q for Queenstown, and S for Southampton

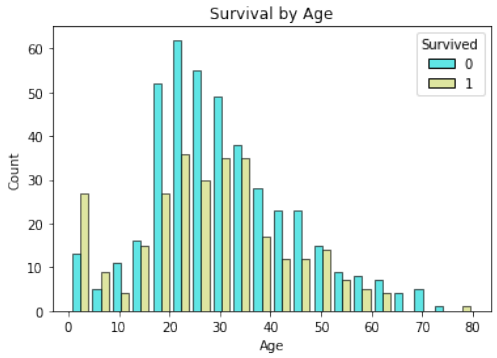

Fig. 1. Survival Count by Age

Finally, Figure 4 displays the data from Figure 3 stratified by sex. Again, similar to the observation made from Figure 2, sex has a large impact on survival. Very few women died in the first two classes. Even for the 3rd class, women still had a greater chance for survival.

Another important objective when evaluating a data set before modeling is to identify any missing data. Notable data that was missing were values for the age and cabin features. There were 177 values (approximately 20%) missing and 681 values (approximately 77%) missing for the age and cabin features, respectively. Since age has been observed to be a great indicator of survival and cabin can indicate class,

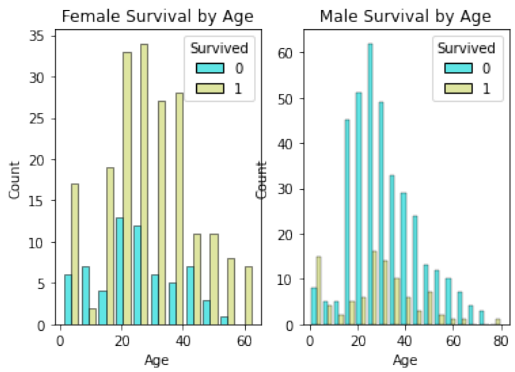

Fig. 2. Survival Count by Age and Sex

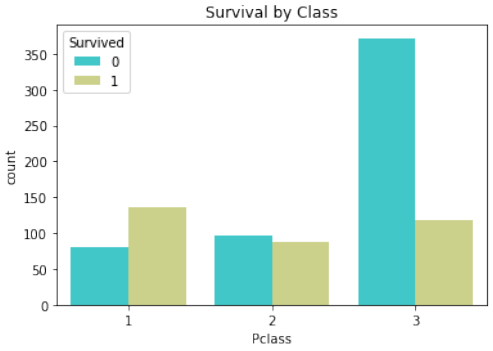

Fig. 3. Survival Count by Class

Fig. 4. Survival Count by Class and Sex

TABLE 2
Values Used to Fill Missing Ages

Title	Median Age
Master	3.5
Miss	21
Ms.	28
Mr.	31
Mrs.	35
Dr.	36

which also has a large impact, it was important to remediate the missing data.

Every passenger onboard had a title recorded. Given the era, we estimated that the milestones relating to a person's title would be largely age based. Thus, the method to remediate the missing age values was to first calculate the median age for passengers with each title. Then, passengers with missing ages were assigned the median age corresponding to their title. Table 2 displays the median ages used to rectify the missing data.

Although we were successful in amending the missing age values for our data set, the same cannot be said for the cabin feature. One method attempted was to find passengers with the same ticket number and available cabin values. However, this resulted in populating only ten missing values. As a result, we decided to drop the cabin feature in favor of a generated feature titled 'Deck' that used the first letter of the cabin feature. For example, a passenger in cabin A30 would then be on deck A. Those with remaining null cabin values were assigned to deck N as a place holder. A logistic regression model taking into account class and fare

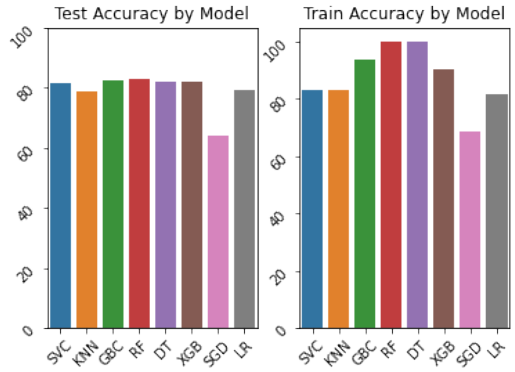

Fig. 5. Performance for each Classification Algorithm

TABLE 3
Top Four Performing Models

Model	Train Accuracy	Test Accuracy
Random Forest	99.85%	82.96%
Decision Tree	99.85%	82.05 %
Gradient Boosting	93.86 %	82.51 %
XGBoost	90.42 %	82.06 %

TABLE 4
Cross-Validation Mean Accuracies

Model	Original CV Mean	New CV Mean
Random Forest	83.84%	83.28%
Decision Tree	78.68%	79.24 %
Gradient Boosting	82.94 %	83.61 %
XGBoost	83.50 %	83.61 %

values was then used to assign a deck value for those we could not remediate via our original cabin search. Moving forward, we dropped the cabin feature and utilized the deck feature as a proxy.

4 Methods

Eight classification algorithms were chosen to create models that predict survival on the Titanic. They were

- SVM (Support Vector Machine),
- KNN (K-Nearest Neighbors),
- GBC (Gradient Boosting Classifier),
- RF (Random Forest),
- DT (Decision Tree),
- XGB (XGBoost),
- SGD (Stochastic Gradient Descent), and
- LR (Logistic Regression).

Before we created the models from the eight above algorithms, a pre-processing procedure took place. First, it is common practice to initially split the data set into testing and training subsets. We split the original training data set provide by Kaggle into new subsets of testing and training data. We split the 891 elements of training data into 25% for the new testing set and 75% for the new training set. The next step was to create two subsets, X and Y, from both the test and training sets. The X set contains all of the features except name, passenger ID, and cabin number since name and passenger ID does not have any correlation with survival and cabin number is missing over three quarters of its values. On the other hand, the Y set contains the survival data.

Once the training and testing X and Y sets had been established, we then used One-Hot Encoding for the categorical features and scaled the numerical data. Again, this is common practice since machine learning algorithms requires numerical input and output. Finally, we ran the eight classification algorithms with their default settings to produce eight models for predicting survival. The results are shown in Figure 5.

From Figure 5, what we can immediately see is that the four bar graphs in the center (representing Gradient Boosting, Random Forest, Decision Tree, and XGBoost) performed the best. The numerical values for their accuracies are displayed in Table 3. However, it may be the case that some overfitting may have occurred since the training accuracy for Random Forest and Decision Tree are nearly perfect. Nonetheless, of the four top performing models with default parameters, Random Forest had the highest accuracy.

5 Experiments

Once we had evaluated the top performing models, we wanted to see how the models would be affected when the parameters were adjusted for optimal accuracy. The process of choosing a set of hyper-parameters for a machine learning algorithm is known as hyper-tuning [3].

The first attempt to improve the accuracy of our models was to evaluate the features based on their importance and remove the 3 least effective features in the hopes of simplifying our model. For the Decision Tree, XGBoost, and Gradient Boosting Classifier algorithms, the features that were removed were parch, embarked, and deck. On the other hand, the features removed from Random Forest were sibsp, parch, and embarked. Once we had removed each of the three least effective features for each of the models, we again evaluated their performance, this time using cross-validation scores. Table 4 displays the cross validation-scores for each model before and after the least impactful features were removed. Notice that the cross-validation score for Random Forest is slightly lower than before the features were removed. Despite this decrease in the mean cross-validation score, when hyper-tuning we were able to achieve a better result with the Random Forest with fewer features. For the remainder of the paper, each model will be further tuned without including their respective lowest 3 features. To do this, we turned to a new method of hyper-tuning called grid search.

Grid search is a tuning technique that finds optimal hyper-parameters for a model. For instance, for the K-Nearest Neighbors algorithm, grid search finds a value k that improves the performance of the model. In essence, it is an automation of trial and error for picking hyper-parameters. Although it is deemed as effective among the data science community, this certainly does not mean it doesn't have its qualms. Because it is an exhaustive method, it can take quite a long time to finish the operation if the hyper-parameter range or number is high [4].

To test grid search on our models, we used GridSearchCV from sklearn. We then provided a list of possible

TABLE 5
Cross-Validation Mean Accuracies

Model	Previous Best CV Mean	New CV Mean
Random Forest	83.84%	84.85%
Decision Tree	79.24%	82.60 %
Gradient Boosting	83.61 %	84.84 %
XGBoost	83.61 %	84.40 %

values for a few hyper-parameters for each model and ran the grid search on 5 folds. When we integrated grid search into our experiment, we found that each model's previous best accuracy had improved, with Random Forest, XGBoost, and Gradient Boosting improving by 1% and Decision Tree improving by 3%. These results are displayed in Table 5. Once again, we find Random Forest to be the top performer.

6 Conclusion

From our initial observations, although many features can have an impact on survival, the most influential feature was sex. That is, you were far more likely to survive if you were female than if you were male. Following the visualization of our data set, we found that the top performing models that gave the best accuracy with their default parameters were Random Forest, XGBoost, Gradient Boosting, and Decision Tree. Of these four, Random Forest performed the best. From there, we were able to improve each models accuracy by approximately 1% by using grid search to find optimal hyper-parameters for each model. While our models did not yield incredibly accurate results, we were faced with a relatively small data set that required a good deal of cleaning and modeling to remedy the null values. In addition, this event was filled with human emotions and decision making that we may not be able to capture in data alone. Everything considered, Kaggle's Titanic data set is a great data set for introducing individuals to machine learning binary classification and the scientific process of using data to tell a story.

References

[1] A. Gavin and C. Zarr, *They Said it Couldn't Sink*, available at the URL: https://www.archives.gov/publications/prologue/2012/spring/titanic.html

[2] A. Goldbloom, *Titanic- Machine Learning from Disaster*, available at the URL: https://www.kaggle.com/competitions/titanic/data

[3] J. Jordan, *Hyperparameter Tuning for Machine Learning Models*, available at the URL: https://www.jeremyjordan.me/hyperparameter-tuning/

[4] F. Malik, *What is Grid Search?*, available at the URL: https://medium.com/fintechexplained/what-is-grid-search-c01fe886ef0a

[5] A. Besbes, *How To Start with Kaggle - An Introduction to the Titanic Challenge*, available at the URL: https://medium.datadriveninvestor.com/start-with-kaggle-a-comprehensive-guide-to-solve-the-titanic-challenge-8ac5815b0473

[6] S. Mukhija, *A beginner's guide to Kaggle's Titanic problem*, available at the URL:https://towardsdatascience.com/a-beginners-guide-to-kaggle-s-titanic-problem-3193cb56f6ca

[7] *Basic Feature Engineering with the Titanic Data*, available at the URL:https://triangleinequality.wordpress.com/2013/09/08/basic-feature-engineering-with-the-titanic-data/