Machine Learning from Disaster

Austin Bouchard and Gabrielle Luney
CSCE 5063: Machine Learning
`aaboucha@uark.edu and gluney@uark.edu`

Abstract

This project uses machine learning to create a models that predict which passengers survived the Titanic shipwreck.

1 Importing the Data

We started with the Kaggle data provided for the Titanic Project. The dataset used for this project was composed of 11 input features and the goal was to predict whether or not a passenger would survive base on each of the values.

1.1 Libraries

We used the following libraries to support our project.

Scikit-learn - Machine learning framework Pandas - Data analysis tool(Dataframes) Numpy - Array manipulation Matplotlib - Graphical visualization Seaborn - High level graphical library based on Matplotlib

1.2 Preprocessing

Once we imported the data, the next step was to ensure it was ready to use for training. The first step was to deal with empty values in the dataset.

Handling Missing Data Our training data contained some empty values in three of its feature vectors. The age and cabin features made the largest contribution to the empty values. Since over 2/3 of the cabin values were missing we decided to remove the feature, as it wouldn't have likely contributed to the performance of our model. For the age feature, we filled the missing values with the mean of the known values since the distribution of feature values didn't contain any outliers.

Binning the Data Some of our categorical features had many possible values, which may add too much complexity to our models and lead to over-fitting. In an attempt to prevent over-fitting we applied "bins" to two of our input features("Age" and "Fare"). Values contained in specified ranges for each feature are grouped together into a specified number of groups.The main drawback to this technique is that there is a loss of information.

Feature Reduction For features that have a large number of missing values, there is a level of uncertainty in filling these values using one of the previously described methods. These features oftentimes need to be removed from dataset. We removed the "Ticket", "Cabin", and "Name" features as they likely wouldn't add any additional useful information. Feature reduction is another way to prevent over-fitting your models.

2 Exploratory Data Analysis

Next, we used statistical information alongside graphical visualization to find correlations among the features.

2.1 Correlation Matrix

By graphing a correlation matrix, we can visualize which features are more likely to impact our target feature. The correlation matrix is composed of correlation coefficients between each pair of features that tell the linear relationship between the features.

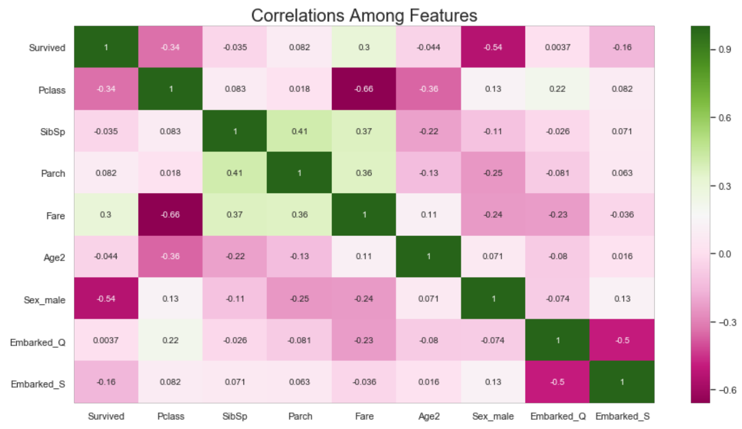

Figure 1: Correlation Matrix

2.2 Further Observations

Females were more likely to survive the crash than males. First class passengers were more likely to survive than second and third class passengers. This is also reflected through “Fare”(Price of a ticket).

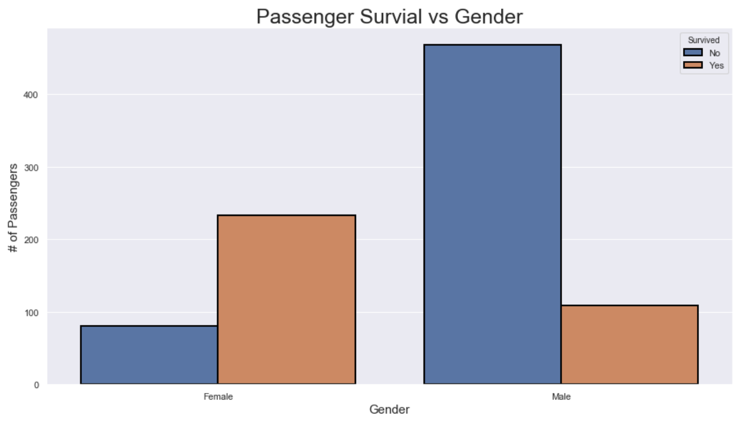

Figure 2: Passenger Survival vs Gender

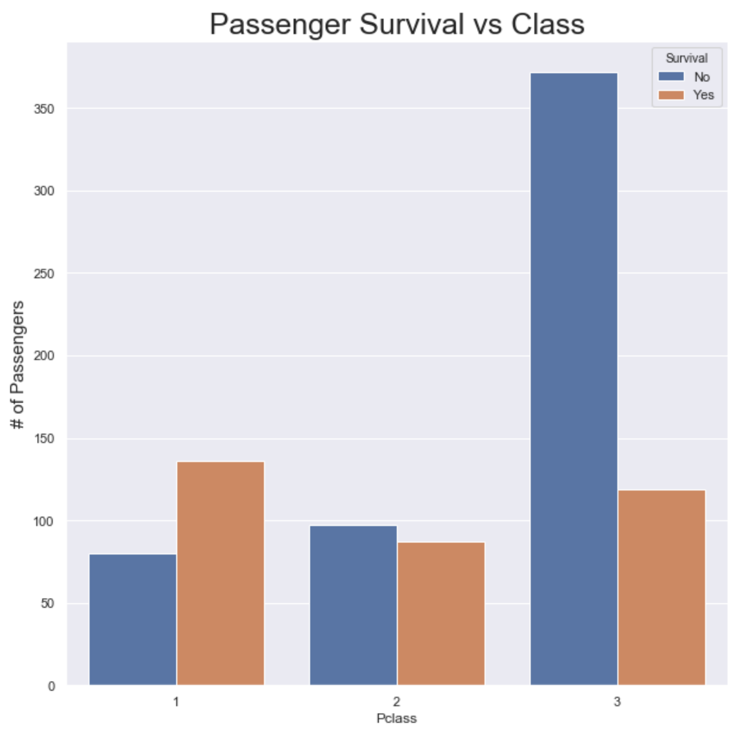

Figure 3: Passenger Survival vs Class

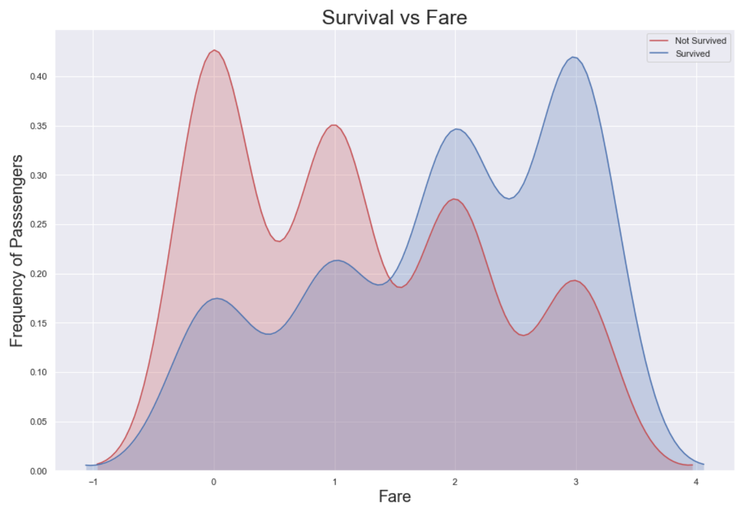

Figure 4: Passenger Survival vs Fare

3 Classifers

Scikit-learn has a library called Classifier Comparison which will allow us to choose which model best for the given data.

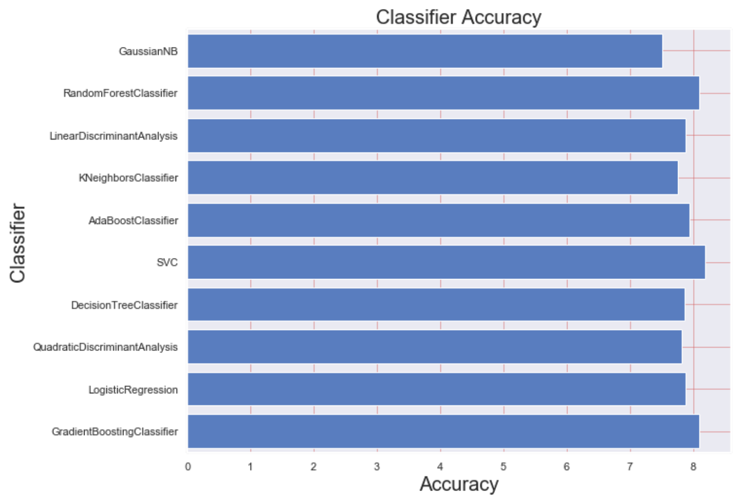

Figure 5: Classifier Models

3.1 Support Vector Classifier(SVC)

Based on the classifier data, the Support Vector Classifier would be the most accurate model. In an attempt to further optimize the SVC we performed a grid search using Scikit-learn's GridSearchCV class. This class fits the training data using all combinations of provided hyperparameters in training and testing of the model. We decided to explore more into using a SVC for classification. The SVC was able to classify the training data correctly around 84 percent of the time with a cross validation score of approximately 0.823 and correctly predicted the test data around 73 percent of the time.

3.2 K-Neighbors Classifier(KNN)

Again to tune hyperparameters, we attempt an exhaustive search over a set of possible options. The KNN model was able to achieve around 84 percent accuracy on the training data, around 75 percent accuracy on the test data, and had a cross-validation score of 0.808.

3.3 Logistic Regression

Following the same techniques we could that Logistic regression gave an accuracy of 79 percent with a cross-validations score of 0.786. However, this model performed the best on the testing data. After submission to Kaggle it was found that the logistic regression model performed best on our dataset with around 77 percent accuracy on the testing data.

Broader Impact

Through the use of machine learning with help from some of the many available libraries and packages, we can oftentimes obtain very reliable and consistent results that can generalize outside of our dataset. To obtain the best results, we need to cross validate our training results with a testing dataset. If we don't perform cross validation our models will likely overfit the data. Dimensionality reduction and hyperparameter tuning can reduce the computational cost of training our models. Choosing a good dataset and performing proper preprocessing is critical for reliable results. Overcomplicating your model doesn't always lead to better results. Going forward we would like to see if we can improve our results.

References

[1]Innat, Mohammed. “Kaggle Titanic: A Machine Learning from Disaster: Feature Eng. Part 1.” Codementor. Accessed May 1, 2020. https://www.codementor.io/@innat2k14/titanic-kaggle-machine-learning-from-disaster-part-1-10gbvqzguc.

[2]“Titanic: Machine Learning from Disaster.” Kaggle. Accessed May 1, 2020. https://www.kaggle.com/c/titanic.