ENSF 611 – Final Project

Titanic: Machine Learning from Disaster

Stephanie Walsh, Toya Okeke and Chelsea Johnson
3 April 2020

Introduction

The main goal of this project is to analyze the Titanic data set provided by Kaggle to build a model that predicts if a passenger survived the shipwreck. We based our process for predicting this on the models that we focused on in class.

Methods

The first step we took to decide how to analyze the data was to visualize it. To do this, we started out with a heatmap that outlined what null values existed in our data. This heatmap can be seen in Figure 1.

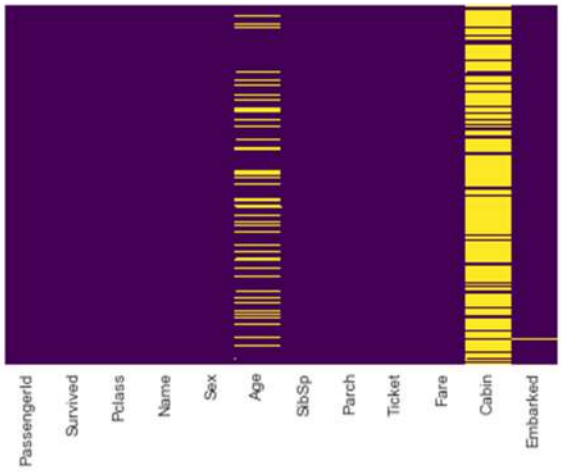

Figure 1 – Features Heatmap

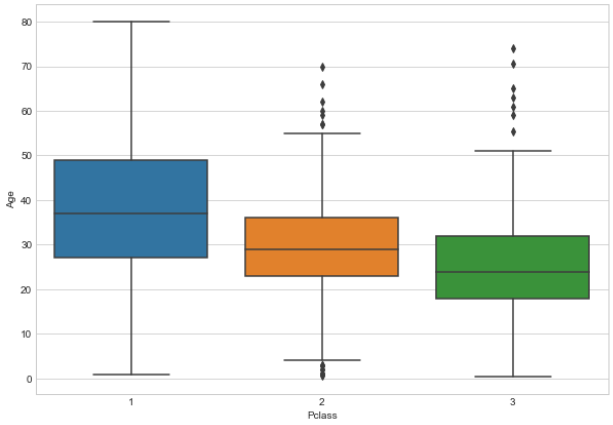

Figure 2 – Age Boxplot

It is obvious that there is a lot of missing data from the Cabin column. Initially we dropped Cabin from our data frame but later decided it may be relevant and chose to treat it as a discrete feature of either 1 (had a cabin) or 0 (did not have a cabin). Another column that had a lot of missing data was age. We plotted a boxplot, which can be seen Figure 2, and realized that the age range for each passenger class was different. We took the average for each class and filled in the missing ages based on those averages / classes. Through other visualizations, other columns that we deemed to be irrelevant were Name and Ticket Number.

Once we had the data filled in, each model used the same data set and employed test_train_split with 90% training data and 10% remaining for validation.

Our next step was to use different models to find the best accuracy that we could. The models we chose were: Decision Tree, SVM, KNN, Logistic Regression, Neural Networks, and Ensemble Methods.

For each of these we tuned the parameters to the best that we could before adding them to the Ensemble Method to try and get the best accuracy from that method.

Logistic Regression

The logistic regression model was built using all the default parameters. The results of the logistic regression model can be seen in Table 1 and Figure 3.

	precision	recall	f1-score	support
Did Not Survive	0.85	0.91	0.88	57
Survived	0.82	0.72	0.77	32
accuracy			0.84	89
macro avg	0.84	0.82	0.82	89
weighted avg	0.84	0.84	0.84	89

Table 1 – Logistic Regression Classification Report

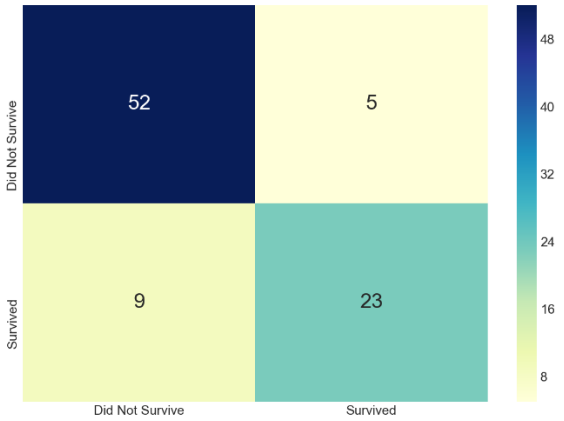

Figure 3 – Logistic Regression Confusion Matrix

Decision Tree

For the Decision Tree model, the main hyperparameter to tune is the min_sample_split. For this we chose to build separate models starting with 40, then 20 and 50. The results of the min_sample_split of 50 model can be seen in Table 2 and Figure 4. The results indicate a great recall score and our highest accuracy score.

Here is the classification report of the Decision Tree Model:

	precision	recall	f1-score	support
Did Not Survive	0.93	0.86	0.89	58
Survived	0.77	0.87	0.82	31
accuracy			0.87	89
macro avg	0.85	0.87	0.86	89
weighted avg	0.87	0.87	0.87	89

Table 2 – Decision Tree Classifier Classification Report

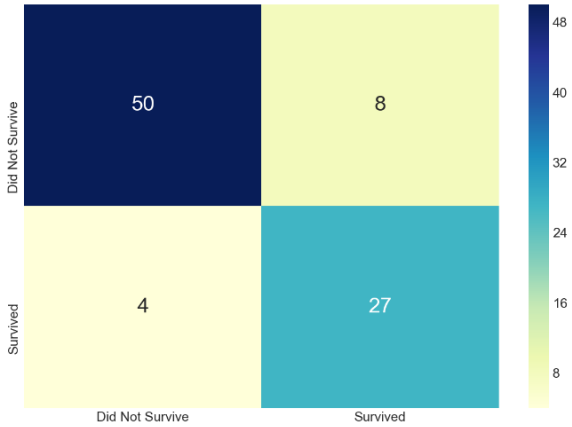

Figure 4 – Decision Tree Classifier Confusion Matrix

SVM Model

For the SVM model, the hyperparameters to tune were C, kernel, and degree. The first step was to load the training data and exclude unwanted features for training. After normalizing the loaded training set, a grid search was performed to find the best hyperparameters. After testing C values from 1 to 50, all the built-in kernels and a maximum polynomial degree of 4, the best hyperparameters were: C = 1, kernel = "rbf", and degree = 2. The results of the of the SVM model can be seen in Table 3 and Figure 5.

Here is the classification report of the SVM Model:

	precision	recall	f1-score	support
Did Not Survive	0.85	0.85	0.85	54
Survived	0.78	0.78	0.78	36
accuracy			0.82	90
macro avg	0.81	0.81	0.81	90
weighted avg	0.82	0.82	0.82	90

Table 3 – SVM Classifier Classification Report

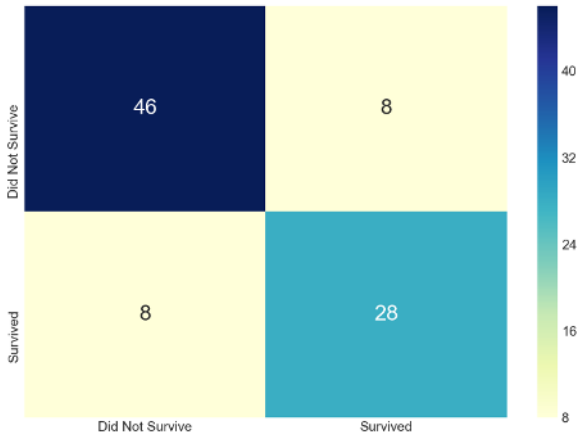

Figure 5 – SVM Classifier Confusion Matrix

KNN Model

For the KNN Model, the hyperparameters to tune are the n_neighbors and weights. We tested weights = "uniform" and "distance". For n_neighbors, the default is 5 however we tried 10 as well, which provided worse results. Because the KNN can be sensitive to the number of features evaluate, we ran the model on different numbers of features. The best results we found to be when all features except those removed above, 'Embarked', and 'Cabin' were used with weights="distance" and n_neighbors=5. The results of the KNN Model can be seen in Table 4 and Figure 6.

Here is the classification report of the KNN Model

	precision	recall	f1-score	support
Did Not Survive	0.81	0.79	0.80	58
Survived	0.62	0.65	0.63	31
accuracy			0.74	89
macro avg	0.72	0.72	0.72	89
weighted avg	0.74	0.74	0.74	89

Table 4 – KNN Classifier Classification Report

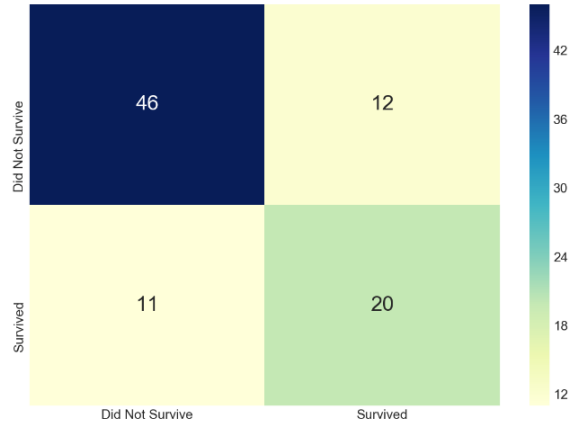

Figure 6 – KNN Classifier Confusion Matrix

Neural Network

Using the methods learned in Lab 7, we built a Neural Network. The ultimate model that we chose used an activation of 'relu' for the hidden layers and 'sigmoid' for the final layer, dropout of 0.2, 'adam' optimizer, and epochs of 100. With some research we realized that for a binary classification that best loss category to use while compiling the model was "binary_crossentropy." You can see the accuracy plot in Figure 7.

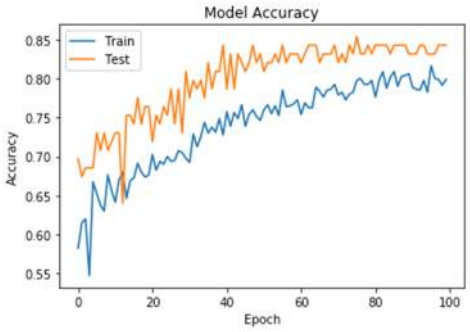

Figure 7– Neural Network Training and Validation Accuracy Plot

Ensemble Methods

We chose the best hyperparameters from each model to build an ensemble method using both hard and soft voting to the classifiers. This method was not as successful as we were expecting. Having used the best combinations of hyperparameters from each model we thought we would get an equal if not better result for accuracy. What we found was a drop of every accuracy score compared to those determined by the models alone. As such we did not submit it to Kaggle for evaluation.

Experimental Results

When looking at the best models of each type, we thought we had done a pretty good job with training our models, however when we uploaded our predictions to Kaggle, the scores were lower for all of our models and we had surprising results for some of them. The results are below:

- Decision Tree: accuracy of 87% for validation, a score of 57.894% on Kaggle
- SVM: accuracy of 82% for validation, a score of 77.03% on Kaggle
- KNN: accuracy of 74% for validation, a score of 64.114% on Kaggle
- Logistic Regression: accuracy of 84% for validation, a score of 75.598% on Kaggle
- Neural Network: accuracy of 83.5% for validation, a score of 75.598% on Kaggle

We believe the reason behind the test scores on Kaggle is the precision and recall of the models used. Many of the models had lower than desired precision and/or recall which means in a low f1-score. To improve out models further we can re-tune them and test different feature combinations for training so the number of false positives (type 1 errors) and false negatives (type 2 errors) are minimized.

Conclusion

This project was the perfect platform to bring together all the concepts that we learned through out the semester. It has offered us a working example of some of the challenges that we were taught to look out for and be able to help mitigate depending on the model we were using. Some of the most impactful things that we will take away from this project are:

- One of the biggest challenges was the lack of available data for both training and testing. In the training set there were 891 total data points and when using train_test_split that brought number of training samples down to 801.
- Missing data in some features meant we had to populate that data based on educated assumptions or dropped the feature all together.
- There was a severe lack of strong positive correlation between any given feature and survival, meaning many feature combinations needed to be tested. Correlations did become more evident when some of the features were transformed to discrete values rather than continuous (e.g. Cabin, SibSp and Parch). Correlation improvements were seen with the transformed data but remained they weak. The correlation matrices for both sets of data with, positive correlations highlighted, can be seen in Figure 8 and Figure 9.

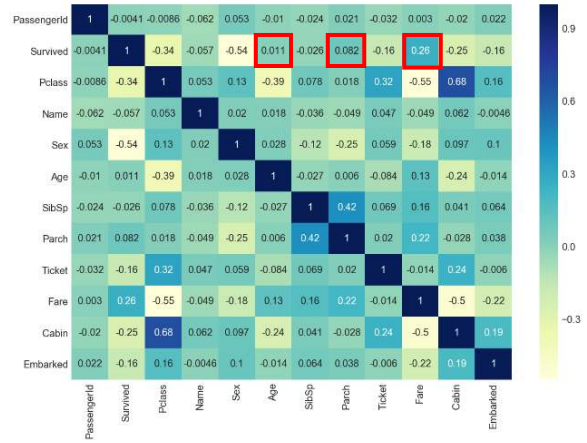

Figure 8 – Correlation Matrix of Untransformed Date

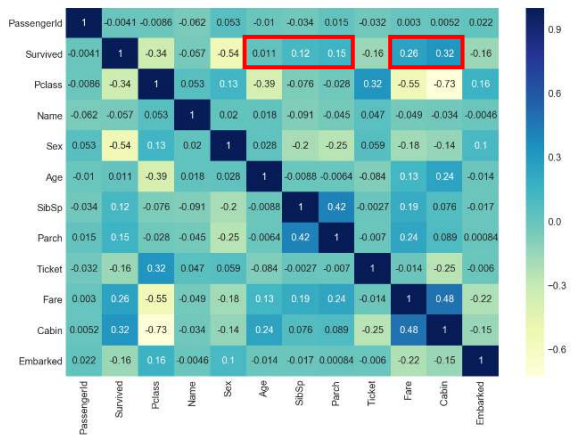

Figure 9 – Correlation Matrix of Transformed Data

- This data is imbalanced for the training data. We believe, because the titanic is well known, this is true for the test data as well. There are significantly more people who did not survive, leading us to think that any model we train will have a decent accuracy, as there is a disproportionate number of people who did not survive compared to those who did.

Contributions

We approached this project as a team. We each took on a different model to tune to the best accuracy that we could. We worked together to come up with the features that we felt best represented the data using visualizations and to help each other with our respective models.