The Sinking of the RMS Titanic: An Analysis of Survivor Data Using Machine Learning Classification Algorithms

Rania Mardini

Mission College

Introduction to Machine Learning

Prof. Jahan Ghofraniha

5/15/2023

Executive Summary

This paper will leverage various techniques from Data Analysis and Machine Learning to explore Kaggle's Competition for the Titanic Dataset to build a prediction model. A large portion of the effort will be paid towards preprocessing the data, feature analysis, and feature engineering. The dataset contains categorical features and a significant amount of nulls which will need to be carefully considered and processed accordingly. Once preliminary data analysis has been completed, and the preprocessing stage is concluded, transformations will be considered. Careful attention will be paid on the comparison of various classification models and their accuracy. Precision/Recall scores will be considered along with confusion matrices and K-Fold cross validation, before making a selection of optimal models to perform hyperparameter optimization and further cross validation. Finally, once optimal hyperparameter values have been chosen, the models showing promise for prediction of the output 'Survival' will be discussed.

Keywords: machine learning, titanic, data analysis, feature engineering

Table of Contents

Background

On April 15, 1912, the RMS Titanic collided with an iceberg, on the fourth day of its maiden voyage from Southampton to New York. Within approximately three hours, the Olympic-class ocean liner sank, killing 1502 of the estimated 2224 passengers and crew. The tragedy reverberated across the globe, as the ship was considered to be unsinkable due to its waterlocks for each boiler room section. Unfortunately, without a mechanism to prevent water infiltration to the upper levels, the sections were not actually watertight. After striking the iceberg and causing damage to approximately 300 ft. along the hull, the water flooded into the forward sections and the Titanic began to sink. In the end, public inquiries led to regulating safety practices and legislation to monitor icebergs in the North Atlantic.

The Titanic dataset is organized by Kaggle for their Titanic competition and is separated into a train and test set. The output for "Survival" is withheld because of the nature of the competition, and Kaggle only returns prediction accuracy when the competition is submitted. Thus, the output for y-test utilized in this model will be organized from the training set in order to assess the accuracy of the models.

Preliminary Data Analysis

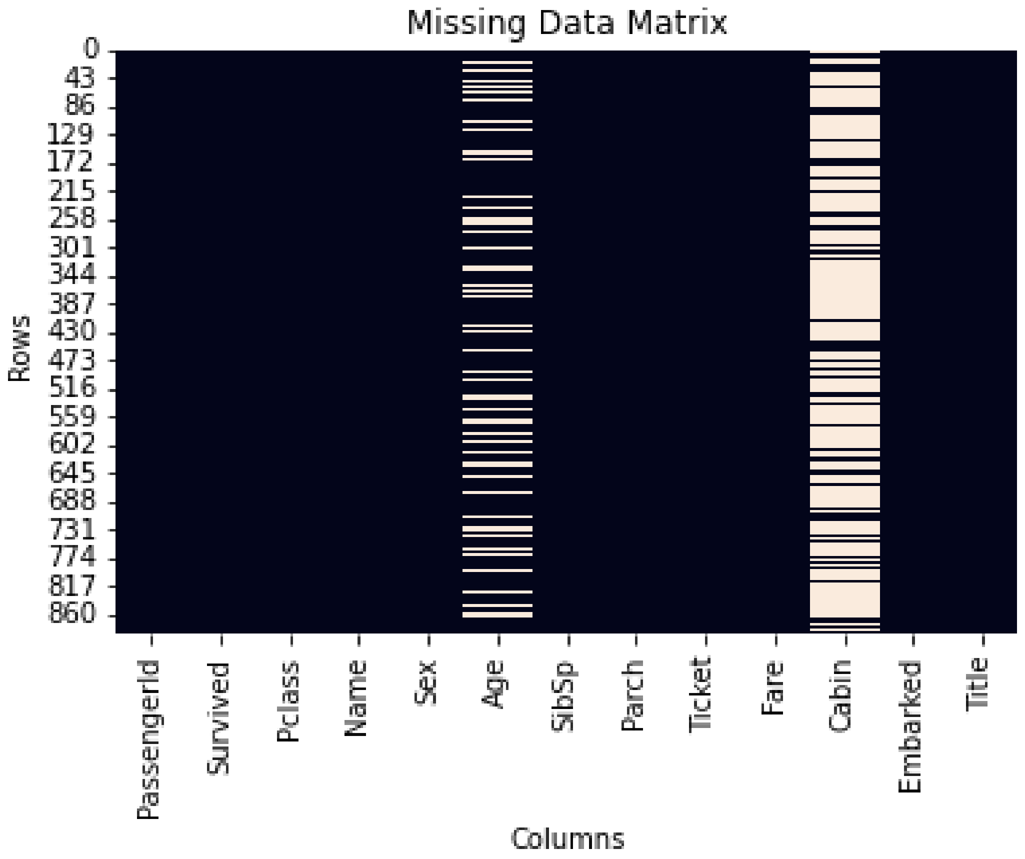
Dataset and Data Dictionary

The Titanic dataset contains 11 total input features with 'Survival' as the output. Pclass features ordinal ordering of the values; 1 denoting 1st class and so on. Five columns are categorical; Name, Sex, Ticket, Cabin, and Embarked.

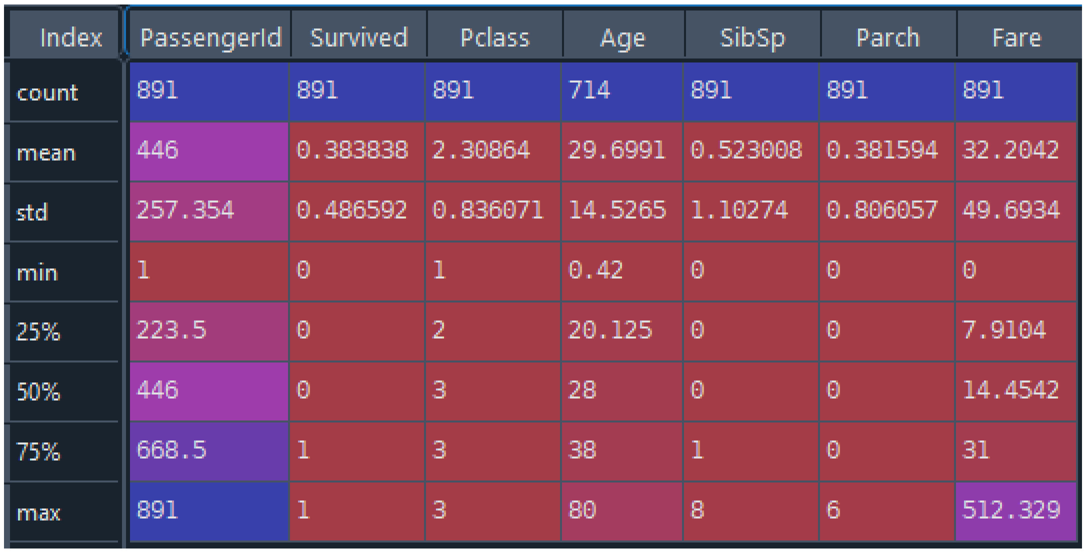
Variable	Definition	Key	Notes
survival	Survival	0 = No, 1 = Yes	
passengerId	Index/ID of each passenger		
pclass	Ticket Class	1 = 1st, 2 = 2nd, 3 = 3rd	Proxy for socio-economic status (SES). 1st = Upper, 2nd = Middle, 3rd = Lower
name	Name of each passenger		
sex	Sex		
age	Age in years		Age is fractional if less than 1. If the age is estimated, it is in the form of xx.5
sibsp	Number of siblings/spouses aboard the Titanic		How the dataset defines family relations: Sibling = brother, sister, stepbrother, stepsister Spouse = Husband, wife. Mistresses and fiancés were ignored.
parch	Number of parents/children aboard the Titanic		How the dataset defines family relations: Parent = mother, father Child = daughter, son, stepdaughter, stepson Some children traveled with a nanny, so parch = 0 for them.
ticket	Ticket number		
fare	Passenger fare		
cabin	Cabin number		
embarked	Port of Embarkation	C = Cherbourg	

		Q = Queenstown
		S = Southampton	

Name, Ticket, and Cabin will be dropped, particularly since Cabin features so many nulls:

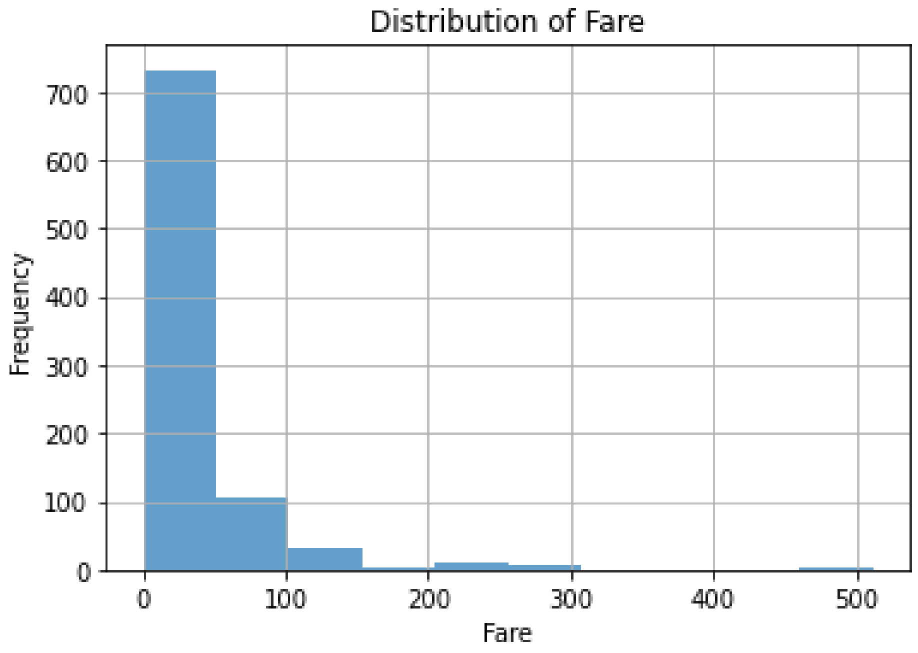


It is interesting to note that the Cabin feature could be imputed utilizing Pclass along with Ticket, though for the sake of this assignment's scope, it will be dropped. A description of the dataset's distributions before any processing is as follows:

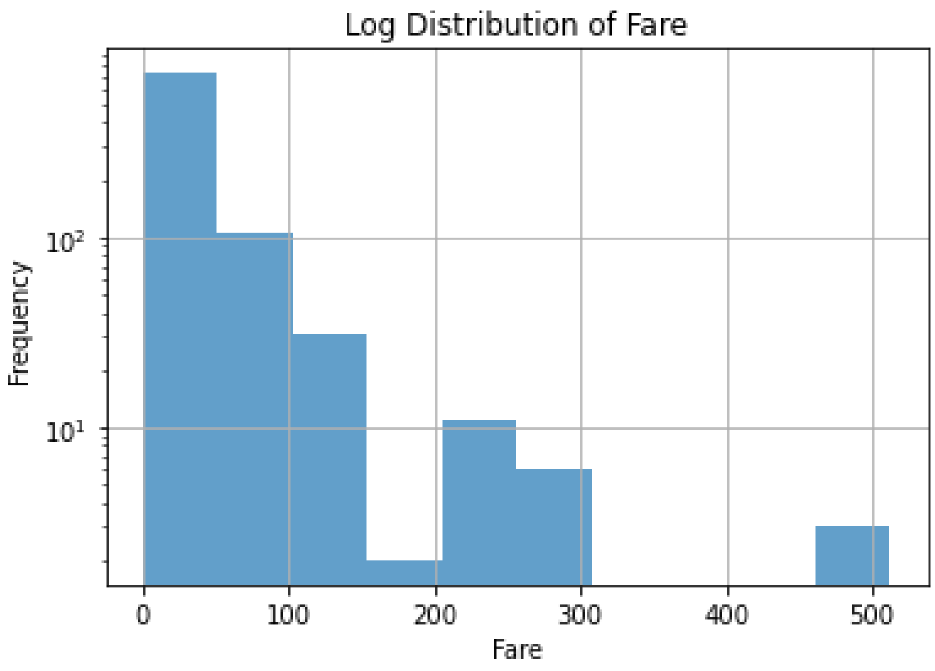


Index	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891	891	891	714	891	891	891
mean	446	0.383838	2.30864	29.6991	0.523008	0.381594	32.2042
std	257.354	0.486592	0.836071	14.5265	1.10274	0.806057	49.6934
min	1	0	1	0.42	0	0	0
25%	223.5	0	2	20.125	0	0	7.9104
50%	446	0	3	28	0	0	14.4542
75%	668.5	1	3	38	1	0	31
max	891	1	3	80	8	6	512.329

PassengerId will need to be dropped as well, since it is an identifying feature, along with Name. Fare seems to have an outlier:

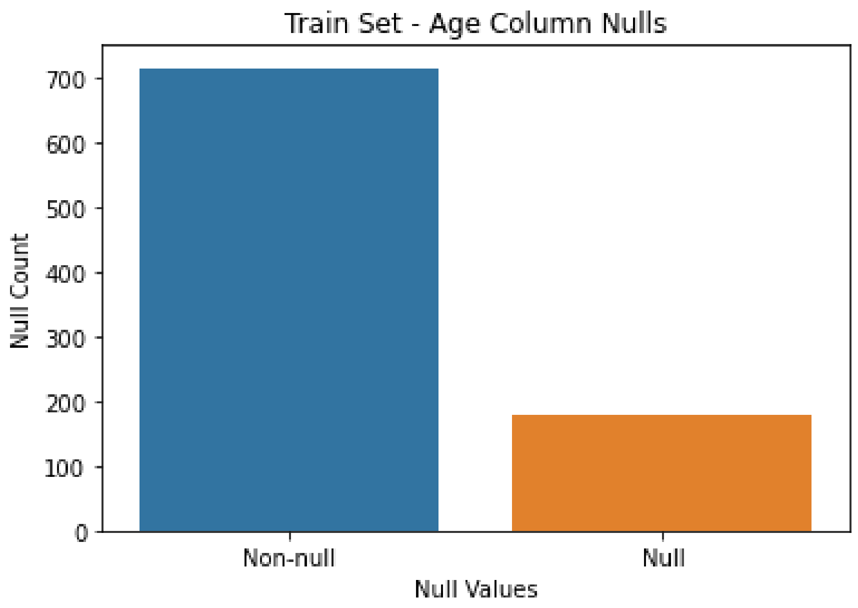


Looking at it on a logarithmic scale can help visualize that outlier and the distribution. This gives us some insight as to how to handle this column and could be noted for feature scaling. However, because the fare can technically have a value of 0, and applying a log transformation of 0 can result with infinite values, and so scaling or using a Box-Cox transformation could be an appropriate solution. However, for the scope of the project, I will continue with normal transformation techniques.



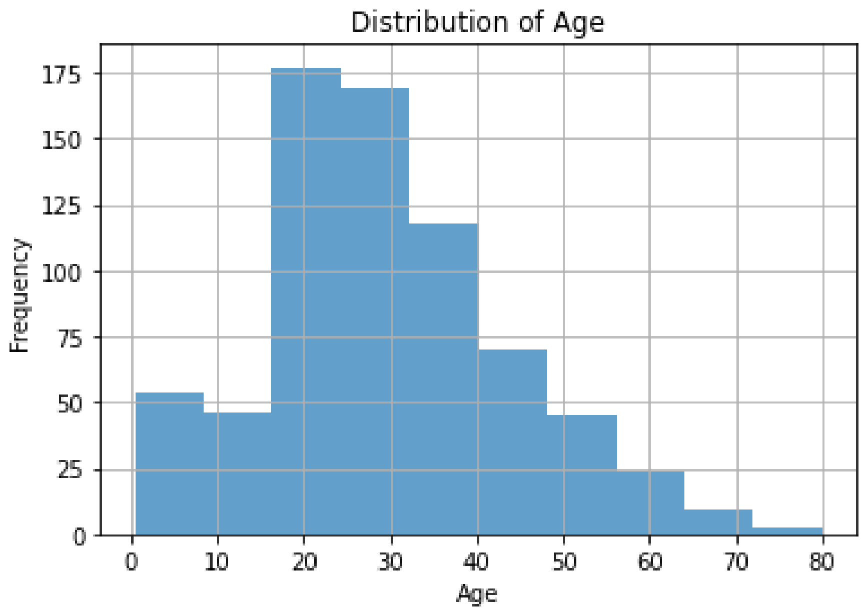
Feature Engineering

The column for Age in the dataset required some analysis for consideration of how to handle the NaN or missing values. One of the first questions when encountering NaNs in a dataset is whether it is appropriate to simply drop the rows with the missing values, dropping the column altogether, or considering a method like imputation. Again, there was a considerable amount of nulls in this column, but not as significant as Cabin. Further analysis of the nulls and their characteristics seemed necessary, especially as imputation seemed to be the ideal solution.

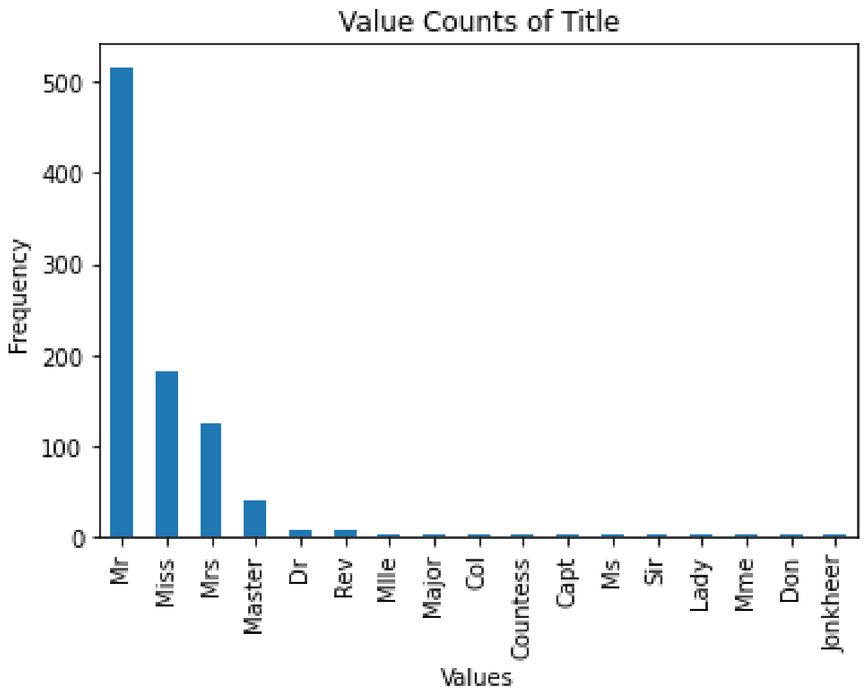


Viewing the distributions of ages shows a few in the upper age bracket, but data was mostly present in the younger to middle age.

Extracting Features



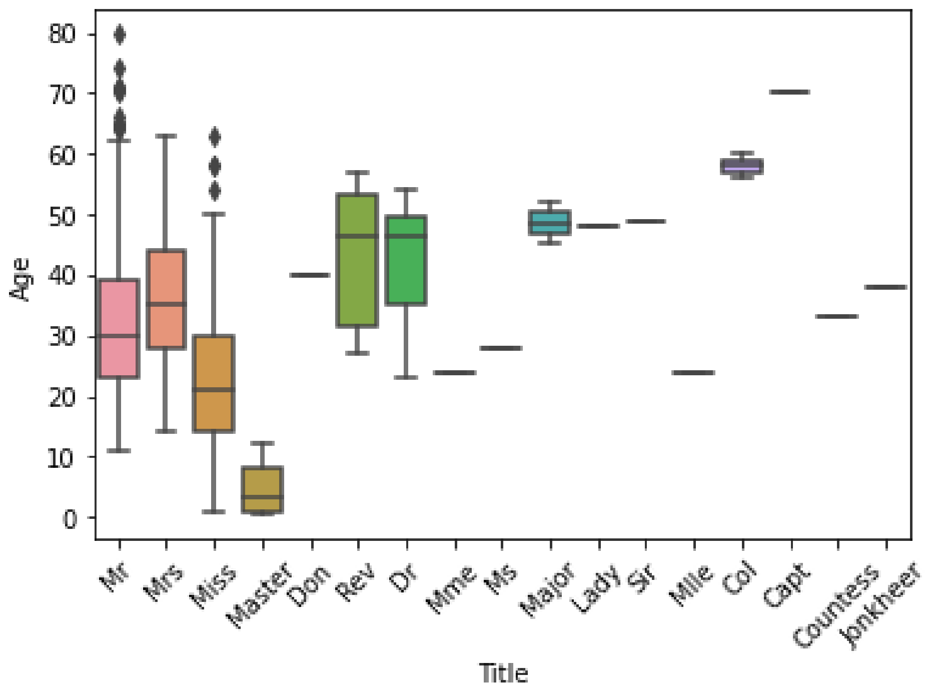
For a little more context, regular expressions were utilized to extract title information from the 'Name' column and compare the resulting data to the relative age data. That helped visualize a little more information both about the distribution of ages across the dataset and specifically across titles, but also helped contextualize how the missing data in the 'Age' column could be appropriately imputed, using the information from the age ranges across specific titles:

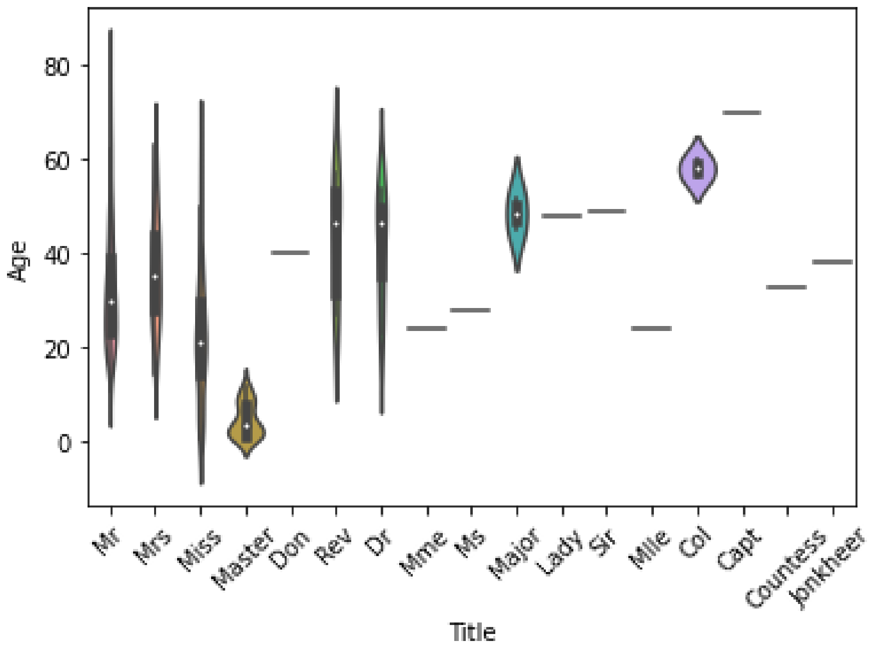


Outliers in 'Age'

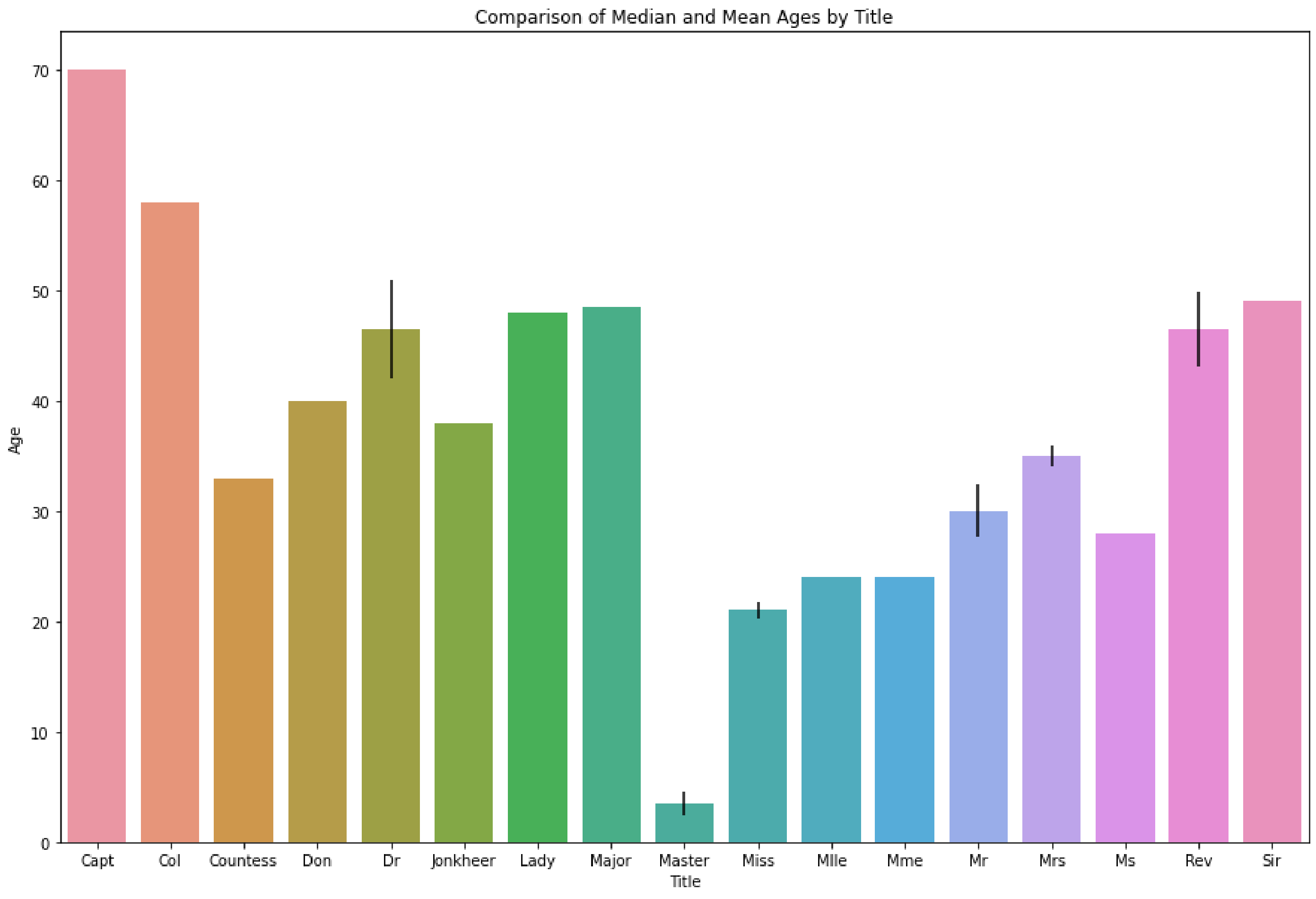
Before making any further decisions, I needed to consider whether or not a mean or median of the age calculation would be more appropriate to impute, particularly since the mean can be sensitive to outliers, while the median can be more robust against them.

With that information, both a box plot and violin plot were used to better visualize any possible outliers in the distribution of Titles across Ages. In addition to the plots, I continued with a calculation of Z-scores, which can help quantify whether or not outliers might exist in the requisite column, as that could affect the quality of imputation.





To confirm the results of possible outliers, using z_scores and a threshold of 2.0, the percentage of outliers in the 'Age' column were approximately 3.25%. After assessing both the median and mean values of ages in each Title, however, the difference between the two appeared to be very slight for most of the titles, with only a few of the titles showing much variability. Using a bar plot with error bars, we can visualize that variability:



Ultimately, because of the robustness of the median against possible outliers, in the context of this dataset and problem, the median value across the titles was chosen to impute the missing NaNs in the Ages column. The information gathered from this process was completed on the training set only, and the resulting median was then used to impute missing Ages across both the training and test sets to limit any danger of data leakage. Before moving onto encoding, the columns ‘PassengerId’, ‘Cabin’, ‘Ticket’, ‘Name’, and ‘Title’ were dropped, and any further nulls in the dataset were dropped and confirmed to have been removed.

Categorical Encoding

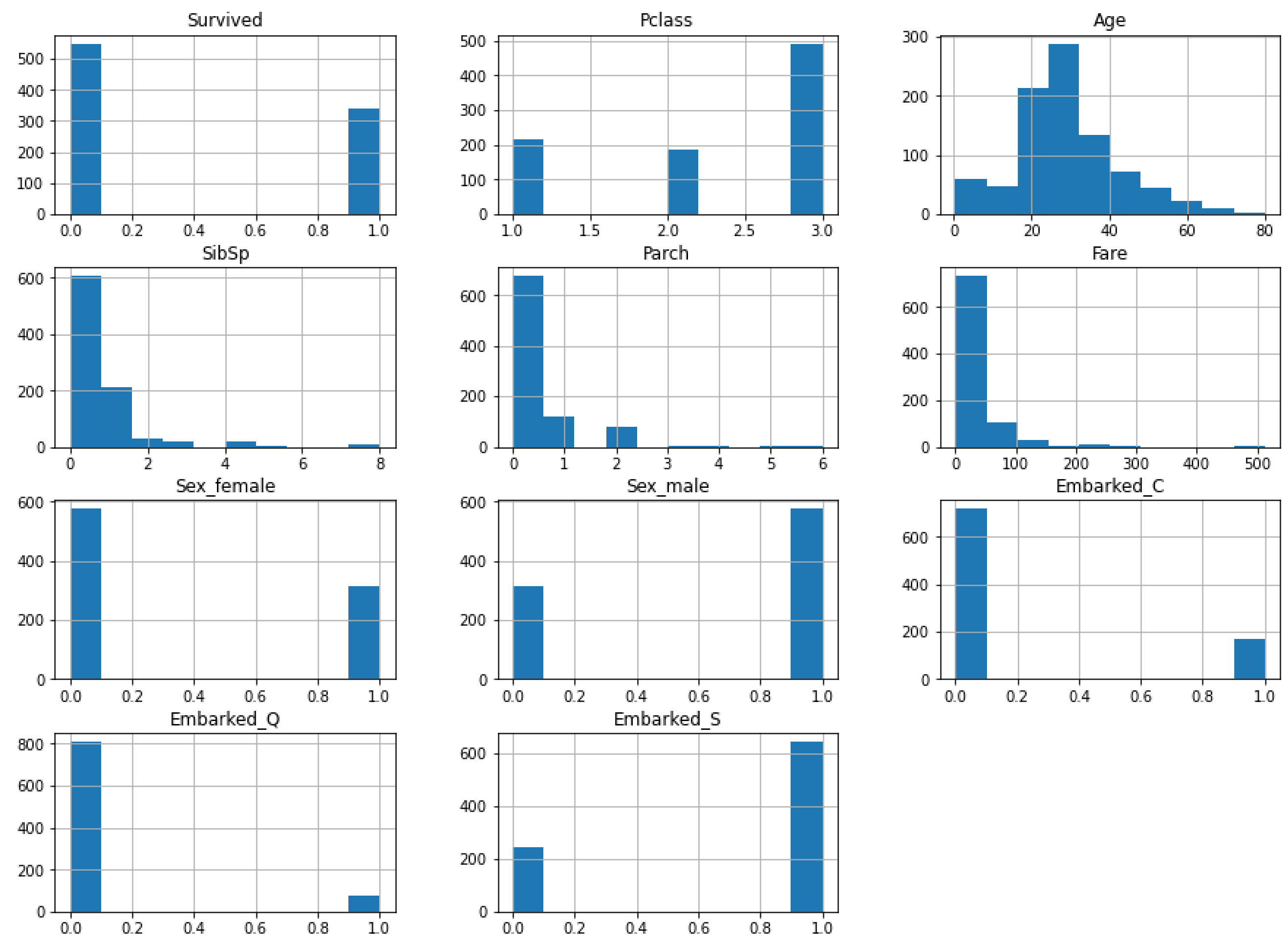
Two columns in the dataset contained categorical data and required encoding. However, careful attention was necessary in processing these columns as neither followed an ordinal relationship. For example, in the case of 'Sex', there is no default between Male or Female, and so OneHotEncoding was an appropriate application.

The other categorical column, 'Embarked', could be argued for in both ways; there is technically an order to which the Titanic arrived at each point of Embarkation before continuing onto its maiden voyage. However, in the context of the dataset, and when considering what we are looking at – the information about each passenger, of those who survived and didn't – the ordering of this movement might not be relevant. Therefore, OneHotEncoder was also used to label the categorical data across the points of Embarkation: Southampton, Cherbourg, and Queenstown.

Label naming was retrieved from the columns themselves to maintain consistency.

Exploratory Data Analysis

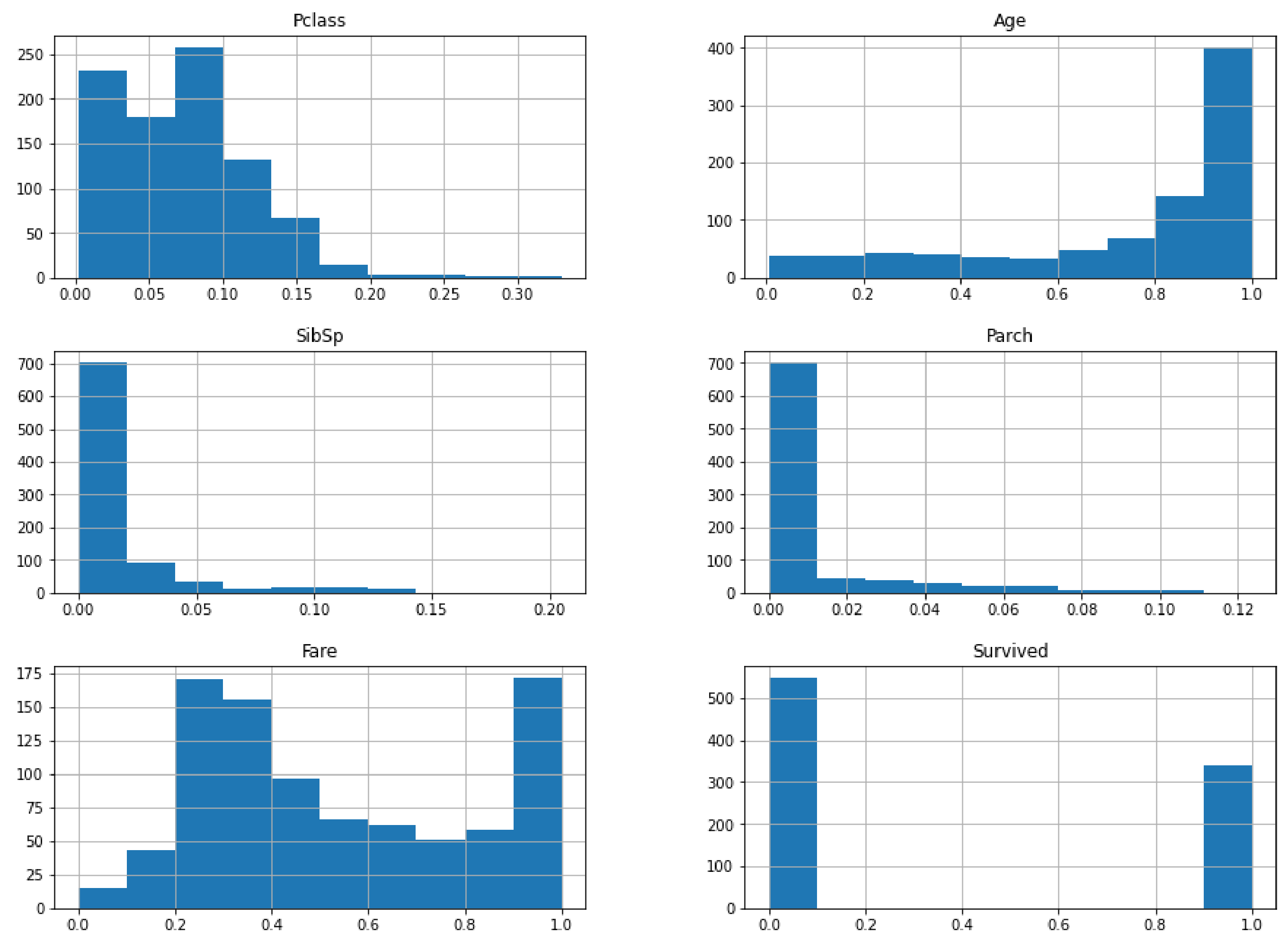
Once the basic preprocessing and feature engineering was completed, and all nulls and NaNs were accounted for, a histogram was produced to help visualize the distributions of the data and help steer me towards any transformations that might be necessary.



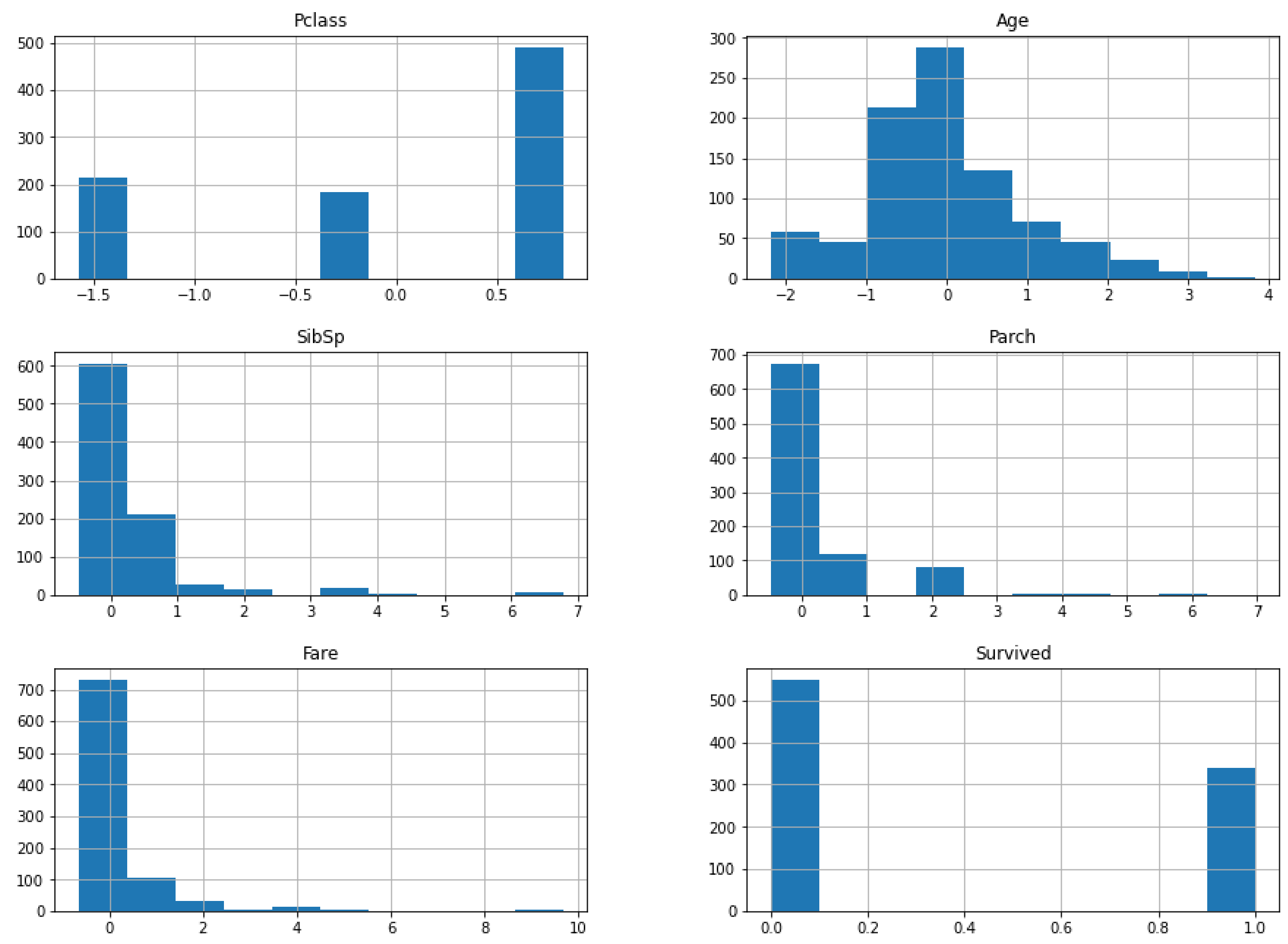
This gives some insight as to what transformations may be useful. Data that has been OneHotEncoded will not be transformed as it is unnecessary when data is not continuous.

Transformations

Two transformation techniques were considered – normalization and standardization. The output of each are as follows.

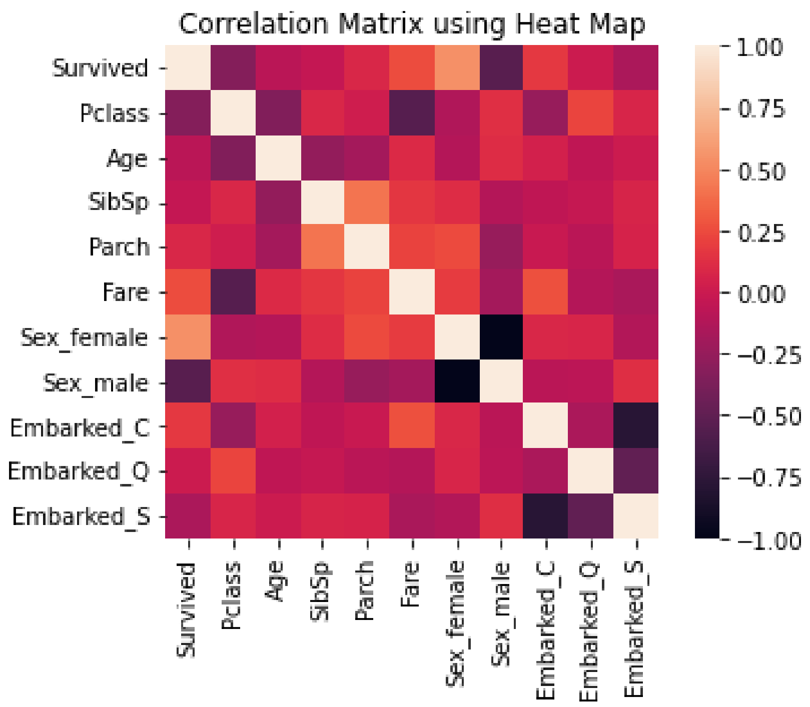


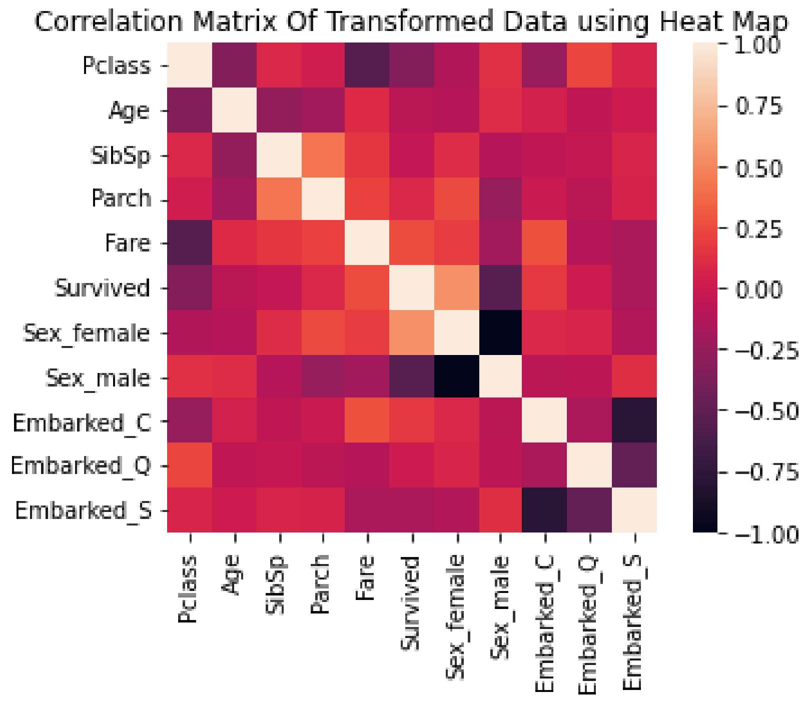
Standardized:



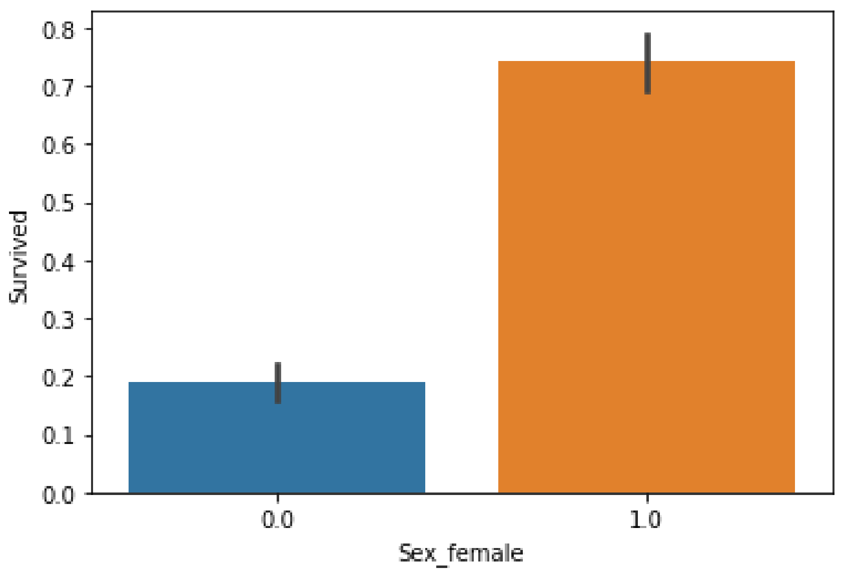
Standardization seems to apply slightly better on the Age column, though not dramatically. However, I will move forward using that. The same transformation was then applied to the test set to ensure consistency between the train and test sets, and to ensure the model will perform well on unseen data. It is worth pointing out that such transformations can really impact the end model, and so this is a good point to rework in the future.

Finally a Pearson correlation heat map was produced to compare, both in the untransformed training set, and after the Standardization technique was applied:

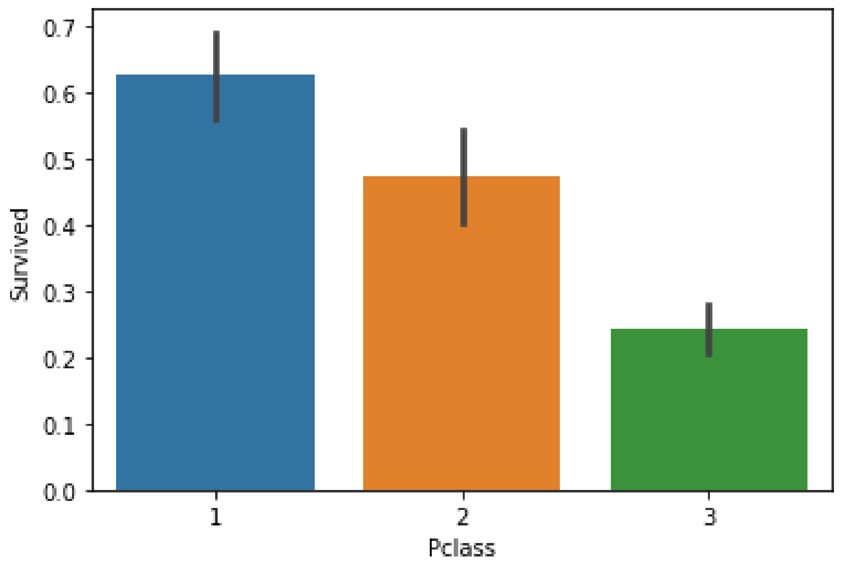




We can see high positive correlation between Sex_female and Survived, and strong negative correlation between being Sex_male and Survival.



Let’s look closer at the Pclass and Survival:

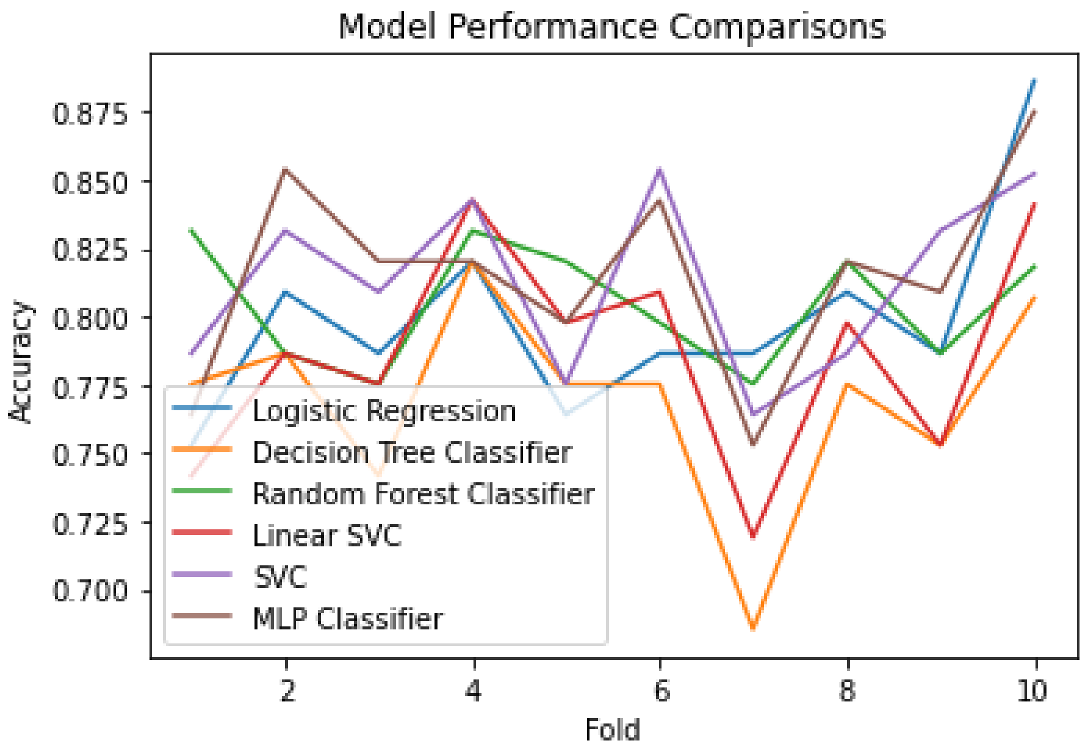


There is a higher chance of survival when a passenger was from First class, in comparison to Second or Third.

After data cleaning, categorical encoding, and feature engineering, 11 features – including those that were OneHotEncoded – will be considered in the predictive model.

Classification Models

Several classification models will be considered and analyzed in terms of their accuracy: Logistic Regression, Decision Tree, Random Forest, Linear SVC, SVC, and MLP. In the first run of classification methods and comparisons, default parameters will be chosen, with optimization taking place later. Using KFold Cross Validation, we can visualize the accuracy comparison of the results here:

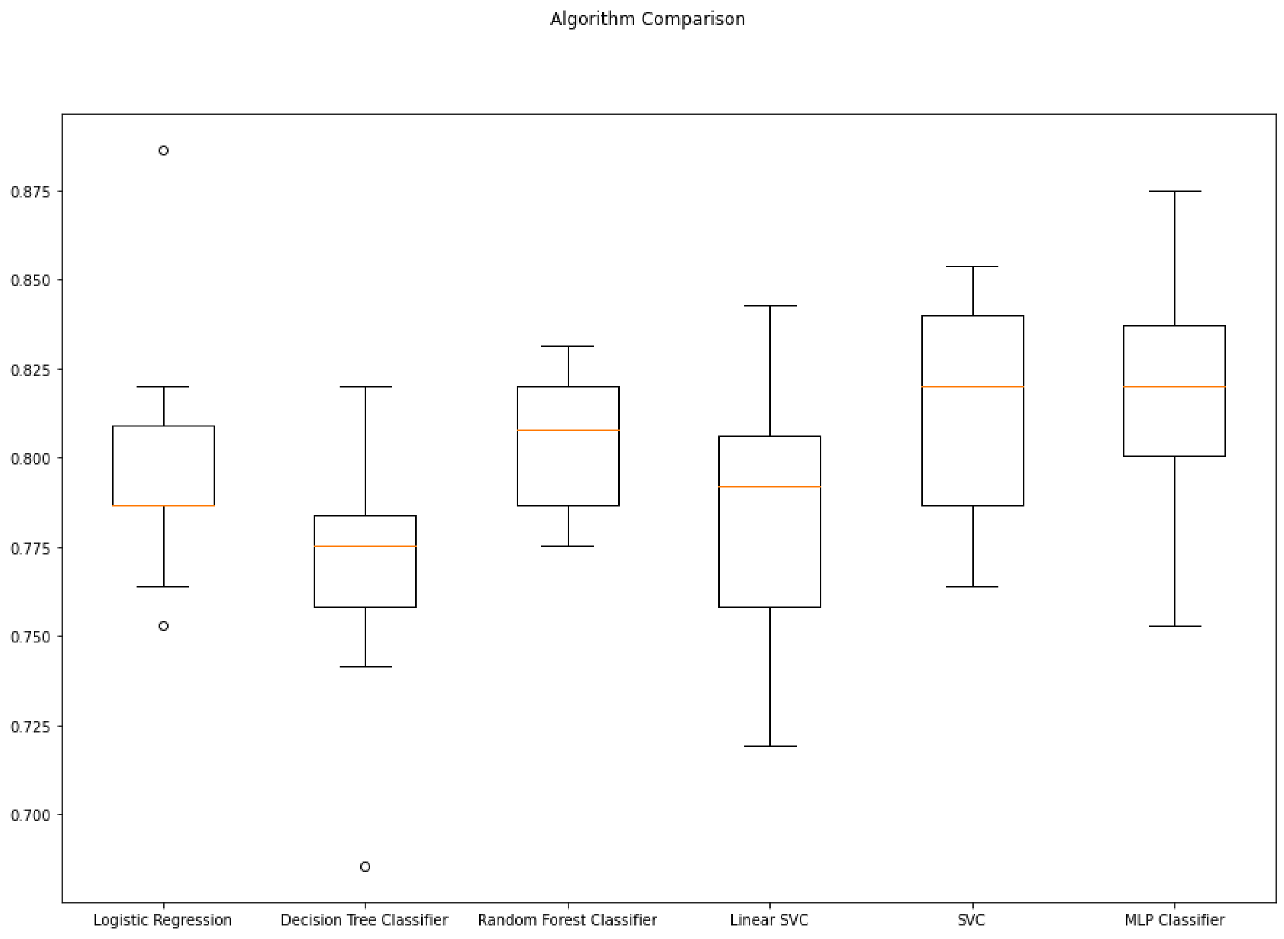


Precision/Recall

Following up with the Precision/Recall analysis[1], the top three performing models were Logistic Regression with an accuracy around 81%, Random Forest Classifier with an accuracy around 78%, and SVC with an accuracy around 81%. When looking at the F1 scores, Logistic Regression received a score of 76%, Random Forest

[1] See Precision/Recall output.

71%, and SVC 72%. Of these, Logistic Regression performed best. An algorithm comparison was used to further validate:



We can see outliers are a possibility with Logistic Regression and Decision Tree Classifier, which if present, can impact model performance. However, with all information considered, the three top performing models of Logistic Regression, Random Forest, and SVC will be considered for hyperparameter optimization.

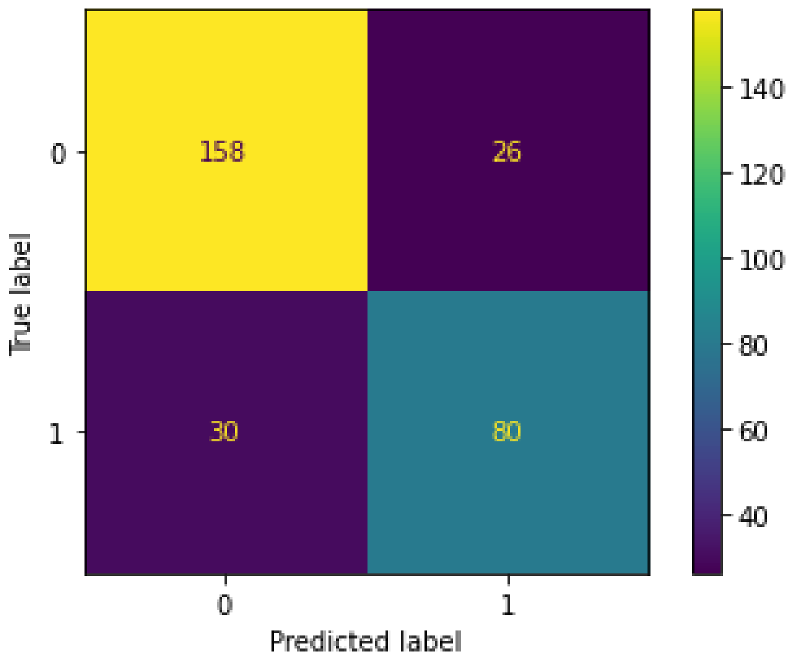
Hyperparameter Optimization

Using GridSearchCV to allow for the most iterations of hyperparameters to be considered, the three selected models were analyzed with the following results and Confusion Matrix Displays to cross validate:

Logistic Regression Optimal Hyperparameters

C: 0.1

Penalty: L2

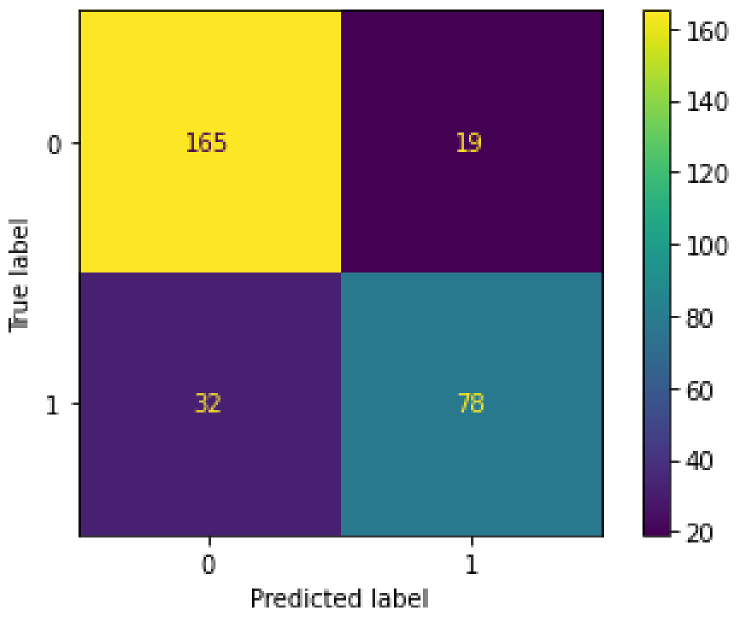


Random Forest Optimal Hyperparameters

Class Weight: Balanced

Max Depth: 11

N Estimators: 20

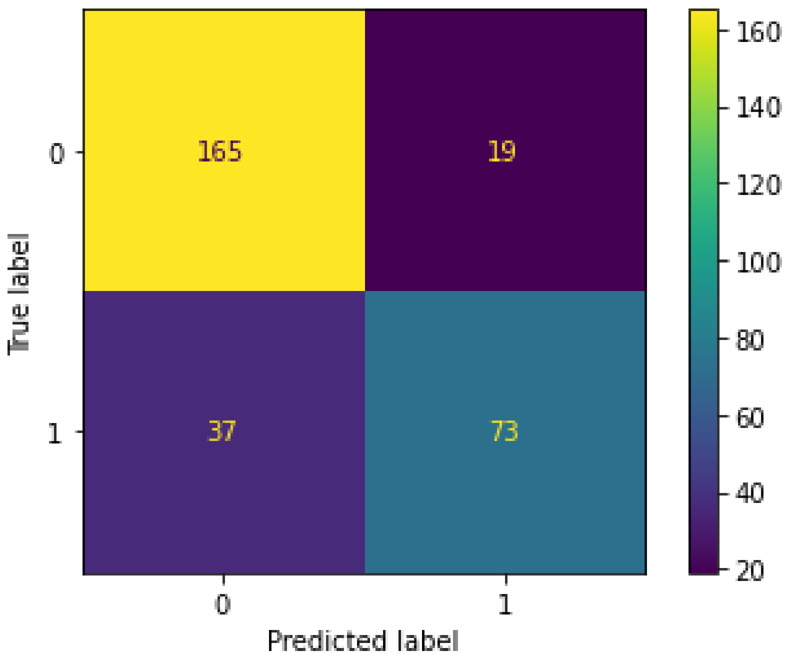


SVC Optimal Hyperparameters

C: 10

Gamma: Scale

Kernel: RBF



Results

Once the hyperparameters were optimized using GridSearchCV, the models were further compared using a variety of techniques to cross validate the results and compare algorithm accuracy.

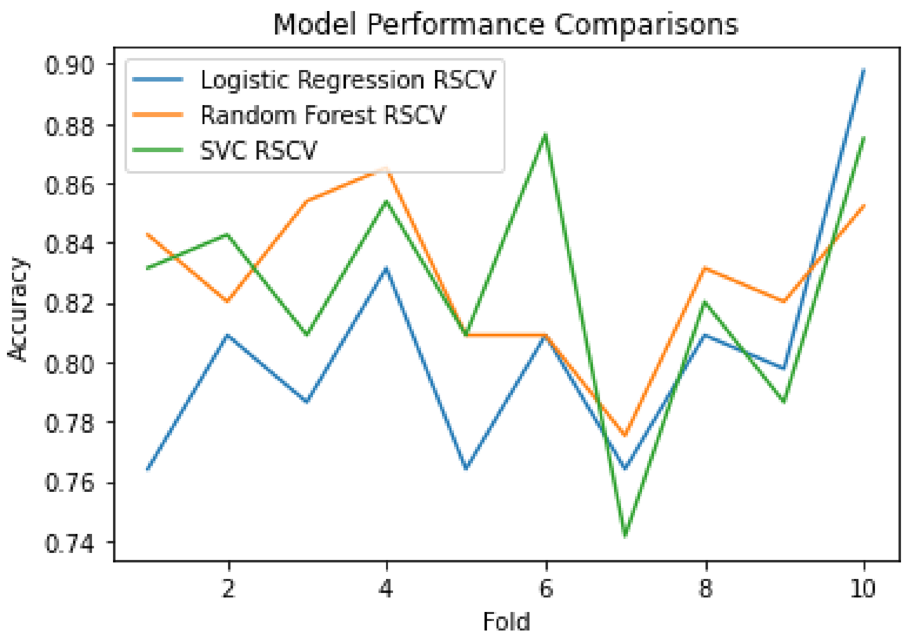
Precision/Recall

Another Precision/Recall[2] was produced to analyze once parameterization has taken place, with all three classification models achieving a weighted avg. accuracy of 81%.

When we look at the model's accuracy in predicting Class 1, Survival, we can see they're all quite similar: Logistic Regression at 77%, Random Forest at 77%, and SVC at 79%. Looking at F1 Scores, Logistic Regression has the highest score at Class 1, Survival, at 75%; however, Random Forest is very close, at 74%, and SVC is not far behind, at 72%. In terms of performance on prediction Class 1, these models seem to perform quite similarly – but let's consider more information before making any conclusions.

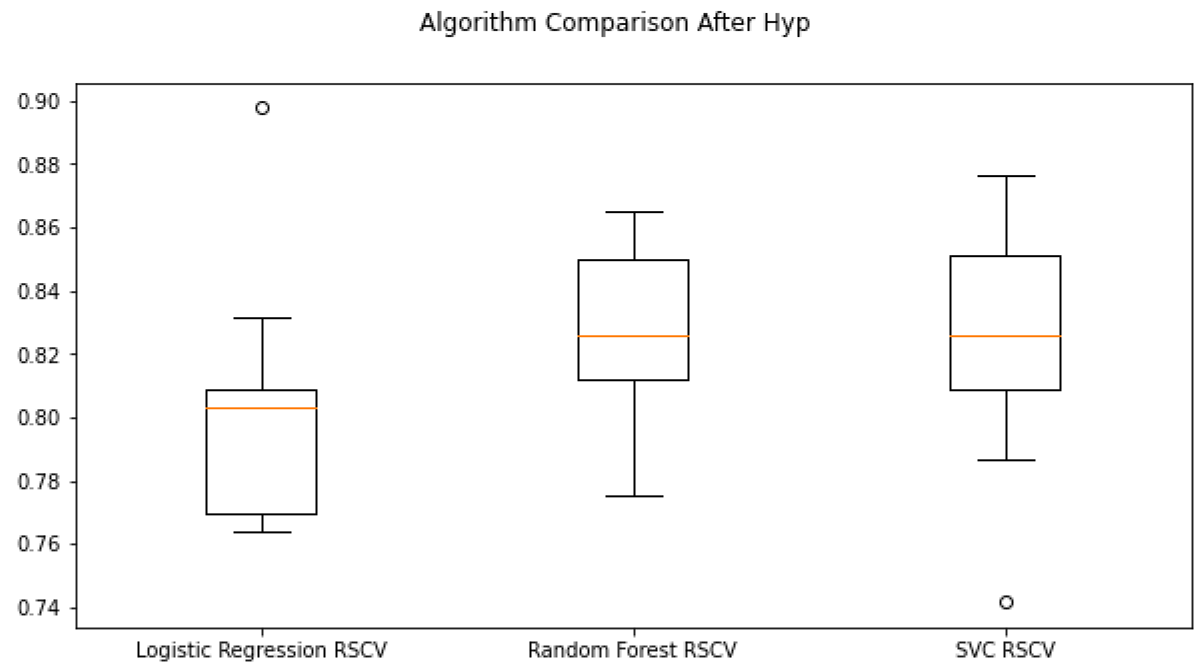
By looking at the KFold Cross Validation, we can also note information about the accuracy of each model in the cross-validation folds and in particular, note the variance in model performance across folds:

[2] See the Precision/Recall output.



Interestingly, we can see that the models perform differently across different sets of data, showing that there might not be as much consistency across model comparison, though the model with the most variability is SVC. The high amount of variability in particular with SVC and Logistic Regression point to possible low bias and possible overfitting. In comparison, Random Forest features less extreme fluctuations across folds, and might perform more stable in comparison to the other two models.

Finally, a box plot algorithm comparison helps elucidate a little more still on how these models performed, where we can more readily visualize the median accuracy values, as well as any outliers, along with the range of accuracy values:



Conclusion

When considering these results altogether, I would rule out Logistic Regression due to the outlier viewed in the algorithm comparison, as well as the extreme variance in the KFold Cross Validation, and SVC would similarly be ruled out. I would consider Random Forest as a potential model for predicting the Survival of passengers in the RMS Titanic.

However, it is worthy to note that when conducting algorithm comparison before running GridSearchCV, the MLP or Neural Network model appeared to perform in a similar fashion to the three I chose. Therefore, it is worth noting that in future investigations of model selection, MLP might be a viable contender and is a great option to later test and compare against Random Forest.

Ultimately, due to the limitations of the dataset, and the competition guidelines, the accuracy of the machine learning models are not optimal enough, generally.

However, within the scope of the assignment, and when compared to the results of others in the Titanic competition, Random Forest performed quite decently, and could be further analyzed and compared with other models if more data preprocessing steps can be taken.

Future Considerations

Working on the Kaggle Titanic competition allowed me to experience what Machine Learning can look like. The process of data analysis, exploratory statistics, and data processing became a priority from the beginning of the final, when it became clear to me that this was not a clean nor perfect dataset, which is exciting as a student. Much has to be taken into account, from considering what features might be important, to what combination of features or information could be useful in building a predictive model. There were several features that I felt could have used more feature engineering, such as Cabin, Sibsp, and Parch, all of which could elucidate more context in the dataset, and allow for a better predictive model by providing more information. Other considerations include what type of information could be gleaned from tickets, information about survivors from geographical locations or points of Embarkation, and other types of transformations and scaling that could help with certain features.

Most of my attention during this project was spent on feature engineering, appropriate encoding, and other steps in the data preprocessing pipeline, as it felt so important to make sure my models were built on well categorized and cleaned data. When finally arriving at the Machine Learning section of the assignment, I was able to not only test a few classification algorithms we have studied over the semester, but I was also challenged to use multiple forms of cross validation and ensure that I was getting an accurate view of model performance. With a variety of information, both visual and from code outputs, I could make clearer decisions towards choosing a model. Finally, throughout this process, my skills in visualization, coding, and version control were really put to the test, to make sure that I stayed organized, on task, and could easily weed away and troubleshoot various parts of my code.

References

Abid, A. (n.d.). *Titanic - machine learning from disaster*. Kaggle. Titanic - Machine Learning from Disaster | Kaggle

Brownlee, J. (2020, August 14). *Data leakage in machine learning*. MachineLearningMastery.com. https://machinelearningmastery.com/data-leakage-machine-learning/

Gе́ron, A. (2023). *Hands-on machine learning with scikit-learn, keras and tensorflow: Concepts, tools, and techniques to build Intelligent Systems*. O'Reilly.

Mathur, D. (2020, August 3). *Outliers handling with Titanic example*. Kaggle. https://www.kaggle.com/code/max22112019/outliers-handling-with-titanic-example

Mukhija, S. (2019, June 22). *A beginner's guide to kaggle's titanic problem*. Medium. https://towardsdatascience.com/a-beginners-guide-to-kaggle-s-titanic-problem-3193cb56f6ca

Supratim Haldar, kabochkov, Noordeen, & Swati. *Feature names from onehotencoder*. Stack Overflow. https://stackoverflow.com/questions/54570947/feature-names-from-onehotencoder

Titanic - machine learning from disaster. Kaggle. (n.d.). https://www.kaggle.com/competitions/titanic/data