Titanic - Kaggle Challenge

Antonella Azzarello

3/7/2021

Data Source - Kaggle

Source of the Titanic Kaggle Challenge: https://www.kaggle.com/c/titanic/overview.

Extras:

Analysis completed while listening to the Titanic: Motion Picture Soundtrack by James Horner, for feels.

Libraries Needed:

```
library(kableExtra)   # Tables
library(ggplot2)      # Visualization
library(Amelia)       # Missing Viz
library(mice)         # Imputation
library(Hmisc)        # Moar Imputation
library(dplyr)        # Data Manipulation
library(gridExtra)    # Plot Grids
library(scales)       # Visualization Assistance
library(corrplot)     # Correlation Plot
library(tidyverse)    # All the things
library(summarytools) # Summary Tools
library(caret)        # Model Training/K-Fold
library(MLmetrics)    # Quick ML Metrics (Accuracy, F1, etc)
library(e1071)        # Support Vector Machines
```

The Goal

Per Kaggle, the goal of this challenge is to use Machine Learning to create a model that predicts which passengers survived the Titanic shipwreck.

Data Exploration

We begin by loading in the training dataset and taking a quick look at the data. For the purposes of condensing the table view, the attribute `Name` was excluded.

As some of the data had outright missing values, these were replaced with 'NA's for quantifying purposes later on.

We have a total of 891 observations within our Training dataset. We see that about 62% of the observations within the `Train` dataset did not survive.

PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	male	22	1	0	A/5 21171	7.2500	NA	S
2	1	1	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	female	26	0	0	STON/O2. 3101282	7.9250	NA	S
4	1	1	female	35	1	0	113803	53.1000	C123	S
5	0	3	male	35	0	0	373450	8.0500	NA	S
6	0	3	male	NA	0	0	330877	8.4583	NA	Q

Survived	Passengers
0	549
1	342

Data Dictionary

- PassengerId = ID of passenger
- Survived = if passenger survived; 0 = No, 1 = Yes
- Pclass = Ticket class; 1 = 1st/Upper, 2 = 2nd/Middle, 3 = 3rd/Lower
- Name = Name of passenger
- Sex = Passgener's sex/gender; male/female
- Age = Age of passenger in years
- Sibsp = # of siblings / spouses aboard with passenger
- Parch = # of parents / children aboard with passenger
- Ticket = Ticket number
- Fare = Passenger fare
- Cabin = Cabin number
- Embarked = Port of embarkation; C = Cherbourg, Q = Queenstown, S = Southampton

---

Continuing to explore the data, the focus will remain on the `Train` dataset, as for predictive purposes, it is expected that the records within the `Test` dataset have remained unseen. All learnings of the data will be based purely on the `Train` dataset.

Worth noting, I did read some methods in which the ***`Train`*** *and* ***`Test`*** *datasets were combined for better imputation power. I disagree with this approach, because based on the Kaggle Challenge parameters, the* ***`Test`*** *data is to remain unseen. In the real-world, we would not be able to use true test data to better impute missing values within our training dataset. This would pose a data leakage risk.*

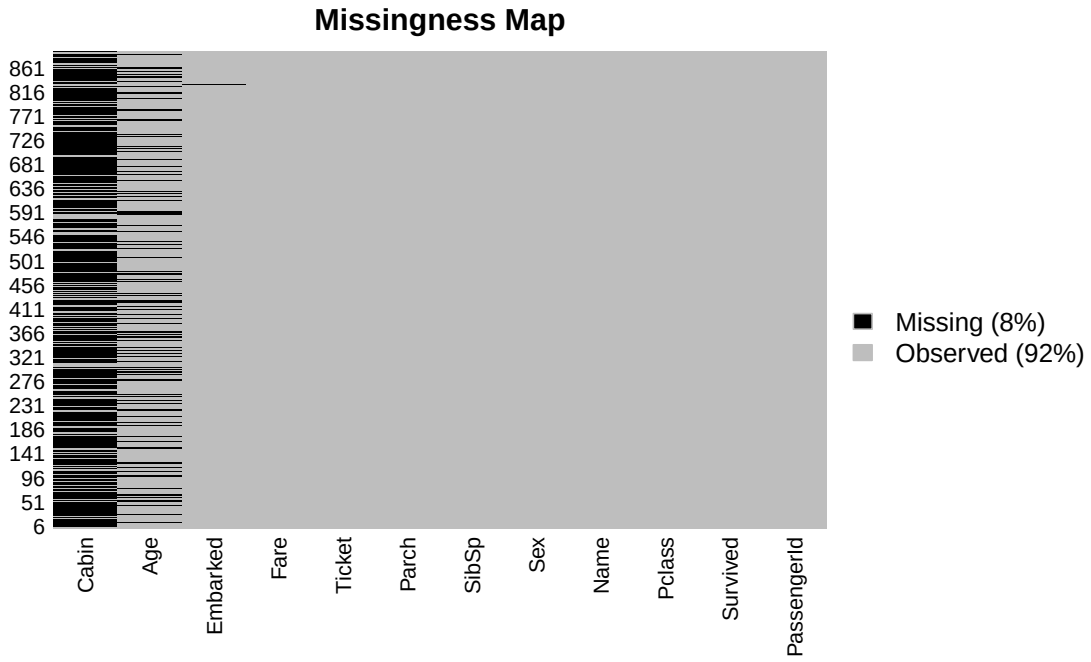
Quick Data Adjustments

Given our variable of interest, we need to ensure `Survived` is categorized as a factor. With the information known, another quick change is to categorize `Pclass` as an ordinal categorical variable. This is needed because `Pclass` 1 is ranked higher than `Pclass` 2, and so forth.

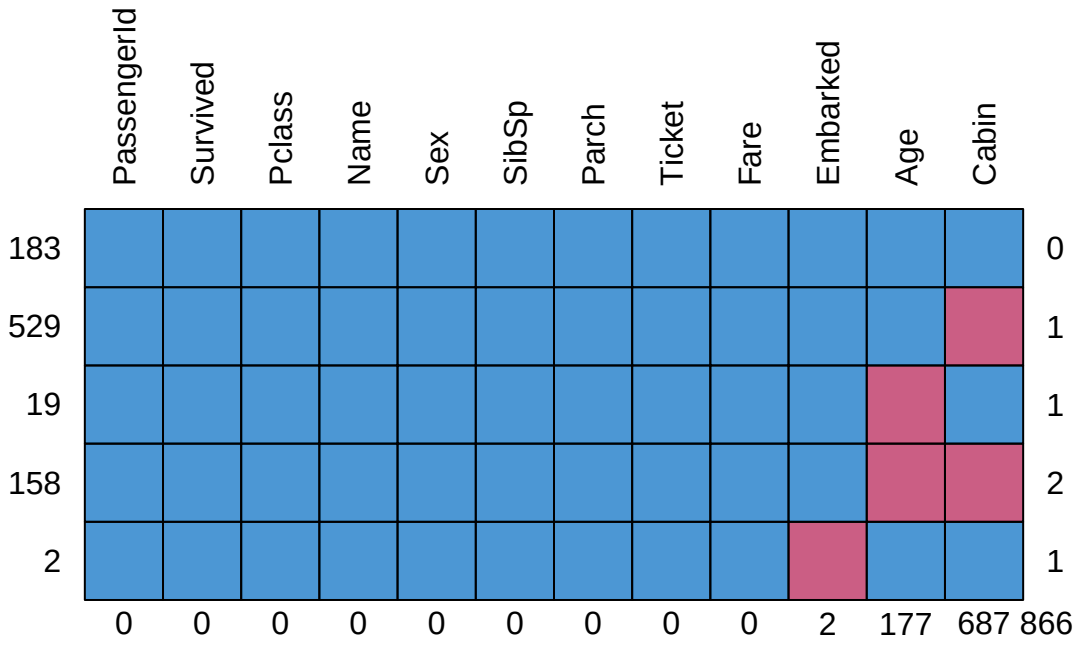
Missing Values

We visualize the missing values a couple of ways. First we look across all records to determine which attributes contain missing values.

We have missing values within the attributes, `Cabin`, `Age`, and `Embarked`. Given there appears to be a large number of missing values for `Cabin`, imputation may not make sense here.



Next, we look at another quick visual to understand the combinations of missing values. While this doesn't tell us too much more, it helps visualize a bit more clearly where there is overlap of missing values across attributes.



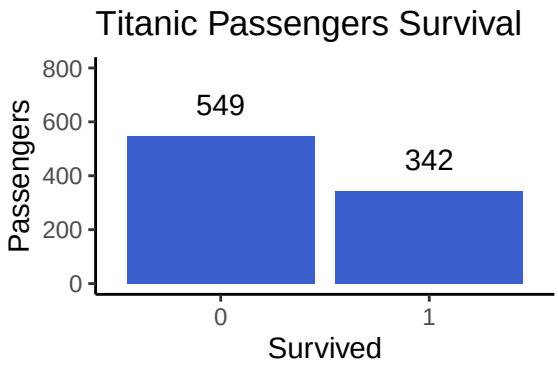
df	Cabin	Age	Embarked
	687	177	2

Quantifying the amount of missing values, `Cabin` has 687 missing values, `Age` with 177, and `Embarked` missing only 2 values. Given there are 891 observations in our Training data set, the missing values for `Cabin` are substantial; about 77% of `Cabin` values are missing.

Predictors/Features

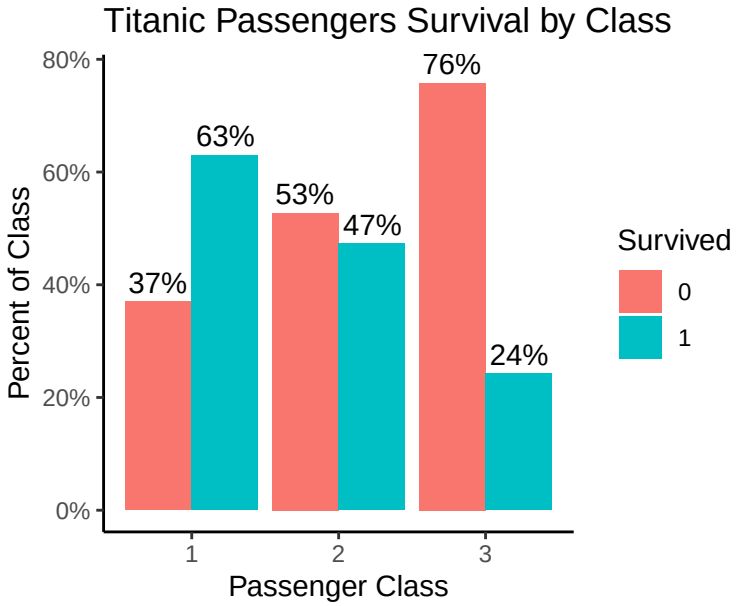
Before we address how to handle the missing values within the data, we first look at how each predictor relates to the survival of a passenger.

As mentioned earlier, based on the data within the `Train` dataset, about 62% of passengers did not survive.



Pclass

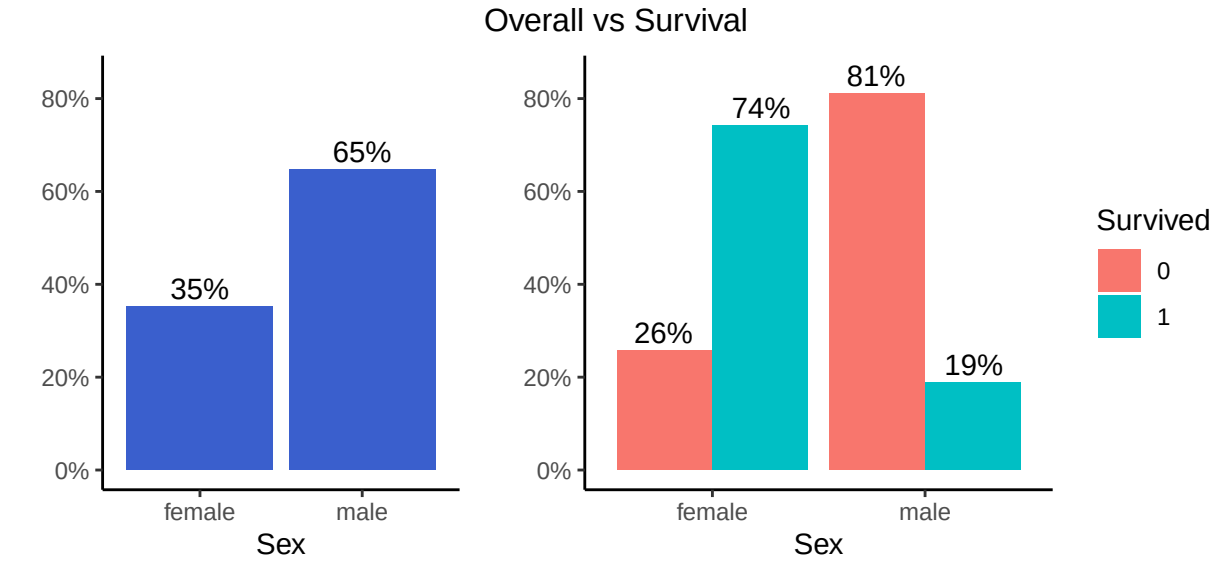
We observe that survival of the passengers varies across `PClass`. There are far more survivors within `PClass` 1 than there are compared to the lower classes. Based on this behavior difference across classes, `Pclass` is most likely a good predictor to use to determine survival.



Sex

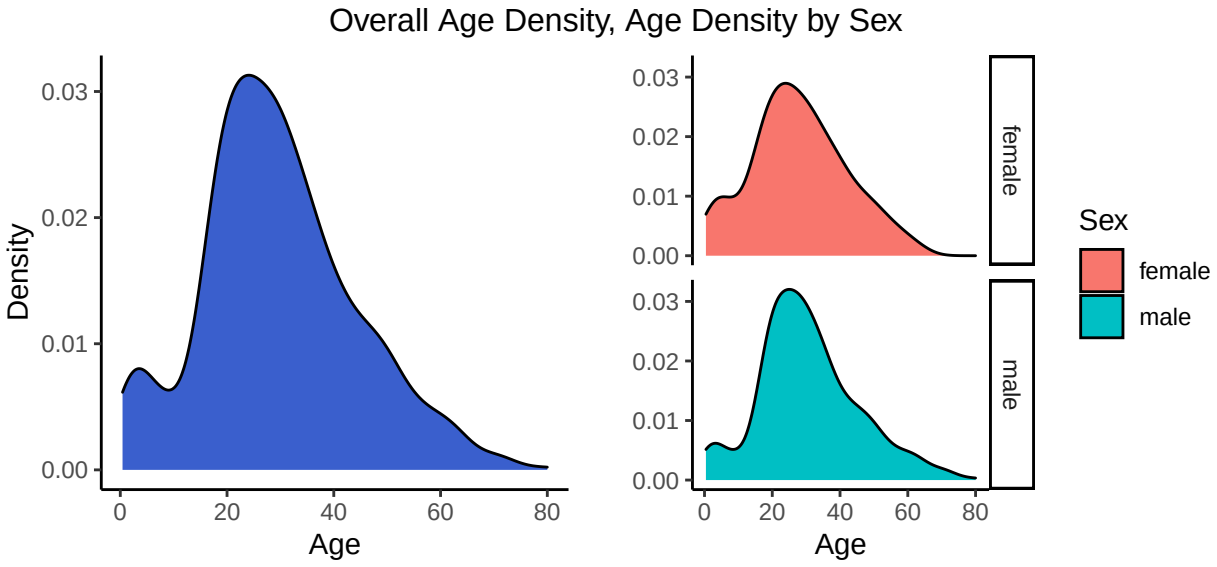
While there were more males aboard the Titanic, fewer survived, in comparison to females. 81% of males on the ship did not survive, compared to 26% of non-surviving females.

`Sex` also appears to be a good predictor of Survival, based on this quick glance.

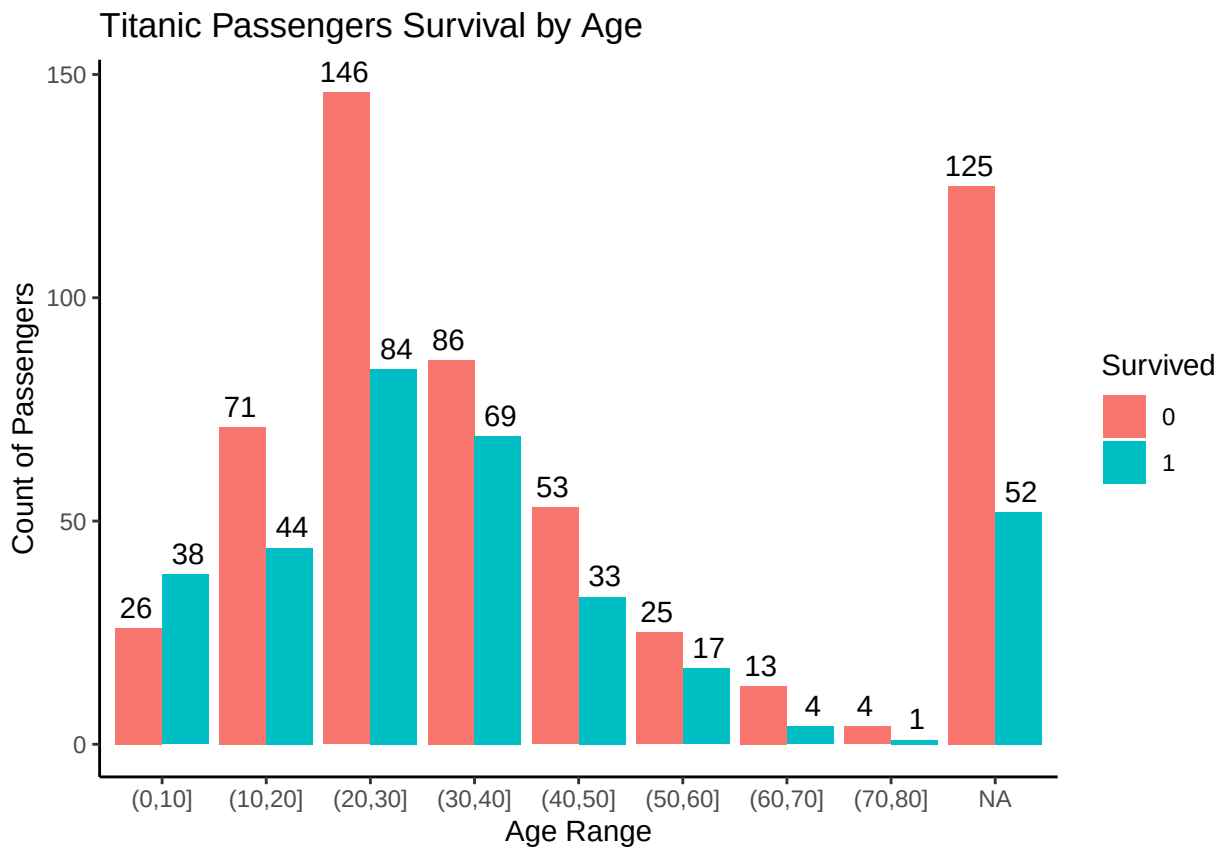


Age

Most passengers were between the ages of 20 and 40. There also appears to have been a larger share of very young female passengers, in comparison to male – while males had a wider spread of ages, ranging to 80.

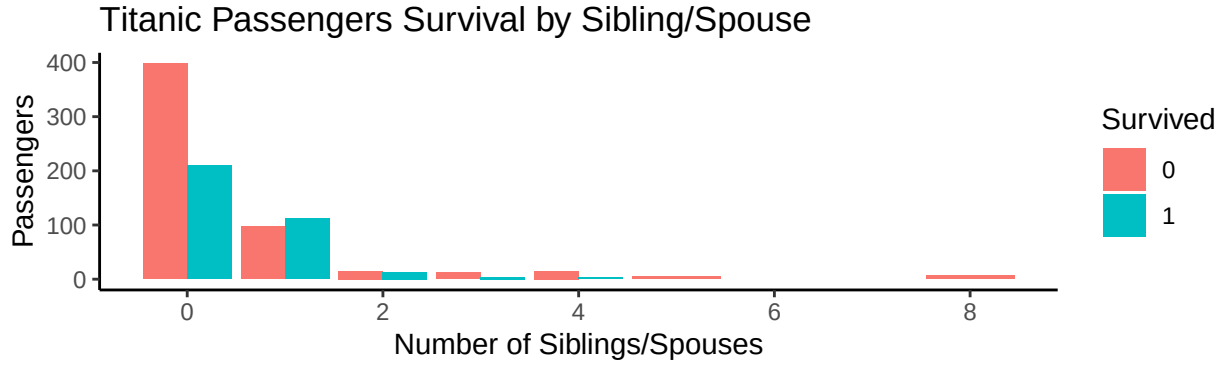


Grouping the `Age` into ranges, we see the group to have the largest amount of non-survivors is 20-30. Also included in the plot, is a reminder that we have several missing ages within our passengers, for both survivors, and non-survivors.



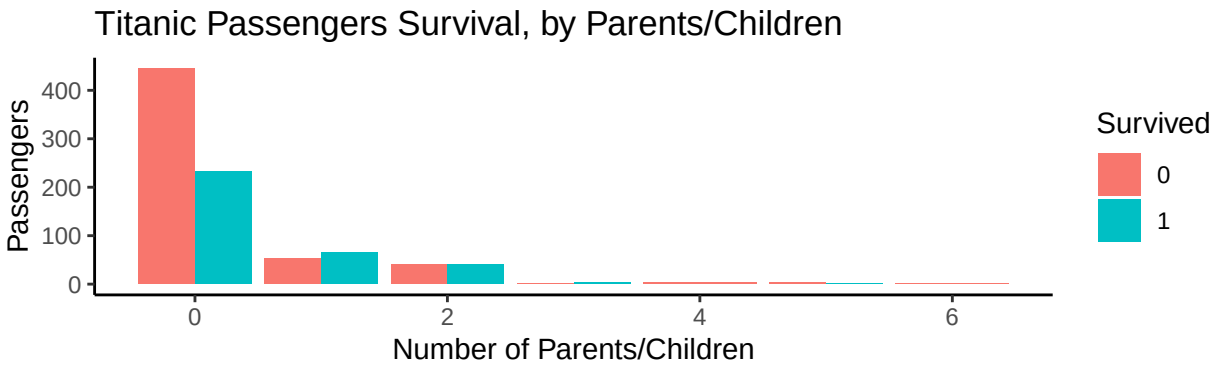
SibSp

The majority of passengers traveled alone, as seen in the column with 0 `SibSp`, indicating they did not have any Siblings nor Spouses traveling with them. Survival, if you were alone was higher than if you were not, however the largest survival difference is seen from 1 to 2 siblings or spouses. Anyone traveling with more than 1 sibling/spouse, had a far less likelihood of surviving.



Parch

`Parch` represents the number of parents or children traveling with each passenger. This is similar to `SibSp`, but not quite the same. Again, across the different groups, the passengers traveling alone had the highest amount of survivors. However, the dip in survivor quantity is after 2 or more `Parch`, rather than more than 1, as seen with `SibSp`. This is most likely children that were traveling, as the only child, with their 2 parents, and would have categorized in Group 1 of `SibSp`.



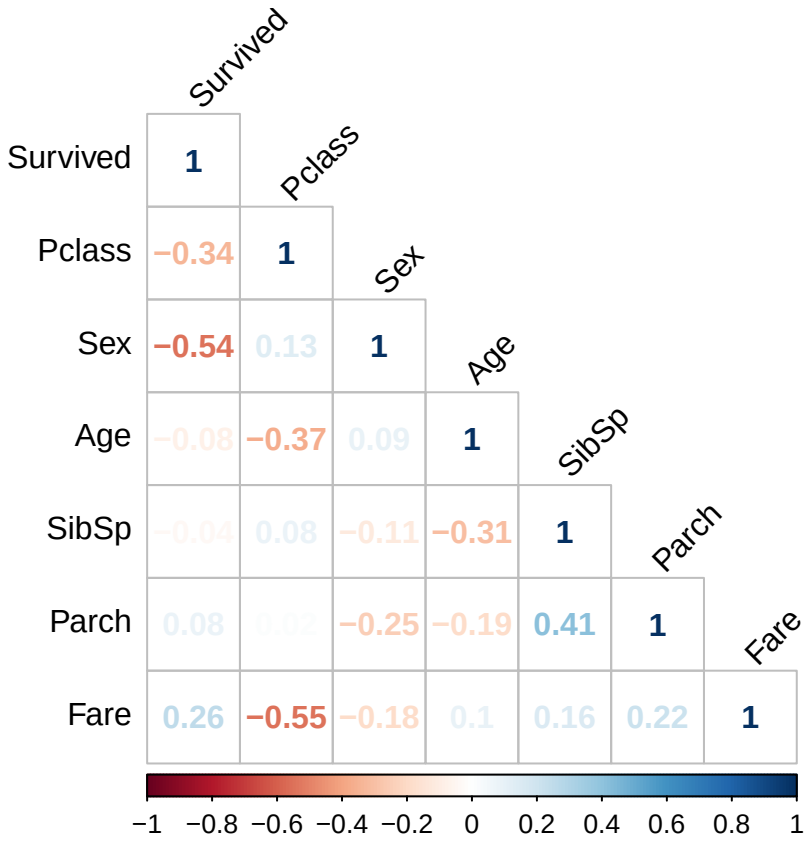
Other Viz

There are many other iterations of visualizations that can be created, however, based on the high level overview, we have a good understanding of some of the main predictors we will use to build the Survival Prediction Model.

Correlations

After sub-setting the numeric and some of the categorical variable, and adjusting the categorical variables to dummy variables, we were able to look at a correlation matrix of the following variables: `Survived`, `Pclass`, `Sex`, `Age`, `SibSp`, `Parch` and `Fare`. Overall, there doesn't appear to have very strong correlations across many of the variables.

The strongest relationships among the listed predictors and `Survived`, are `Pclass` and `Sex`. There also appears to be a relationship between `Fare` and `Pclass`, which makes sense, given the higher class passengers most likely paid a higher fare price to board.



Data Imputation & Cleaning

Cabin

As `Cabin` had such a large amount of missing data, I decided to drop this predictor. I initially felt I could use `Fare` to predict the missing values of `Cabin` - though then I would be concerned about introducing unnecessary multicolinearity. Given `Fare` has a complete set of data within the training dataset, I moved forward with this and dropped `Cabin` from the dataset going forward.

Embark & Age

Since `Embark` only had a couple missing values, I will replace these with the most common value seen within `Embark`. As seen within the frequency table, the most common value for `Embark` is "S" (Southampton).

Given the `Age` variable is close to normally distributed, I used `method = norm` for imputation using the MICE algorithm. This takes into account all the values within the dataset, excluding the `PassengerId` attribute.

	Freq	% Valid	% Valid Cum.	% Total	% Total Cum.
C	168	18.90	18.90	18.86	18.86
Q	77	8.66	27.56	8.64	27.50
S	644	72.44	100.00	72.28	99.78
<NA>	2	NA	NA	0.22	100.00
Total	891	100.00	100.00	100.00	100.00

We also take a quick pulse of our mean and standard deviation for the `Age` variable, so we can verify these remain marginally close once we have completed imputation of this variable.

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   0.42   20.12   28.00   29.70   38.00   80.00     177
```

```
[1] 14.53
```

** ***Imputation Magic Happening...*** **

We confirm with a quick view of the summary statistics, as well as the standard deviation, both the mean and stdev look to be close to what they were prior to imputing the data for `Age`. This is the expected result, and we are good to move forward.

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.42   21.00   29.00   29.48   36.00   80.00
```

```
[1] 13.57
```

We check one last time to ensure all NA values have been addressed through imputation. We can confirm this is now true for both `Age` and `Embarked`.

```
PassengerId    Survived      Pclass        Name         Sex         Age
          0           0           0           0           0           0
      SibSp       Parch      Ticket        Fare    Embarked
          0           0           0           0           0
```

One final look at the data...

```
 Survived Pclass      Name              Sex                 Age
 0:549    3:491   Length:891         Length:891         Min.   : 0.42
 1:342    2:184   Class :character   Class :character   1st Qu.:21.00
          1:216   Mode  :character   Mode  :character   Median :29.00
                                                        Mean   :29.48
                                                        3rd Qu.:36.00
                                                        Max.   :80.00
     SibSp           Parch           Ticket              Fare
 Min.   :0.000   Min.   :0.0000   Length:891         Min.   :  0.00
 1st Qu.:0.000   1st Qu.:0.0000   Class :character   1st Qu.:  7.91
 Median :0.000   Median :0.0000   Mode  :character   Median : 14.45
 Mean   :0.523   Mean   :0.3816                      Mean   : 32.20
 3rd Qu.:1.000   3rd Qu.:0.0000                      3rd Qu.: 31.00
 Max.   :8.000   Max.   :6.0000                      Max.   :512.33
   Embarked
 Length:891
 Class :character
 Mode  :character
```

Modeling Building

Split into Training and Validation datasets

We are moving forward with the following predictors:

- Pclass
- Sex
- Age
- Sibsp
- Parch
- Fare
- Embarked

Since there aren't *many* predictors to begin with, and they are not highly correlated, for the purposes of this analysis, I'm including them all in the model build.

The first model attempt will be a simple Logistic Regression, using K=10 Cross Validation Folds.

To assess the Logistic Regression model's performance, I am looking at the **F1-Score**, as well as the **Accuracy**.

While **Accuracy** looks at the overall *True Positives* and *True Negatives* over all possible cases, **F1-Score** does a better job looking at the incorrectly classified cases. **F1-Score** may be a better metric to compare when looking at this data, given the slight imbalance of class distribution within the `Survived` variable.

For example, it wouldn't be ideal if we classified a Passenger as non-survived, when they in fact, did survive. Though, I am not entirely sure the purposes of how this model would be used - perhaps if trying to take a headcount after a similar sized ship sank, then we would definitely want to pay more attention to the incorrectly classified Passengers.

F1-Score

```
[1] 0.8500382
```

Accuracy

```
[1] 0.8114726
```

The next model build type is a Support Vector Machine, using a radial kernel. SVM classifier model helps us work with non-linear class boundaries; specifically for binary classification.

The radial kernel works with local behavior; only nearby training observations have an effect on the class label of a test observation.

Ways to improve this model would be to find the best cost parameter for the SVM using cross-validation. The cost parameter impacts the margin of the boundaries; the smaller the cost the wider the margins, resulting in a higher chance of misclassifications. However, we don't want the cost to be too much higher, as we run the risk of overfitting the data, and creating an overly rigid boundary.

F1-Score

```
[1] 0.8613618
```

Accuracy

```
[1] 0.8193666
```

Conclusion

When looking at these two models, Logistic Regression and SVM, the SVM model performed better both on the **F1-Score** and **Accuracy** counts. However, the 'better' performance is only marginally better. If I had to choose between these two models, I would most likely go with the more simpler of the two, Logistic Regression.

Next Steps

This was my first attempt at solving this problem. With more time, I think there are some additional things that could be done to further improve the model performance.

Here are a few that I would explore first:

1) Imputing data for `Cabin`, and taking a closer look at this variable.
2) Create new predictors with a combination of existing predictors, sometimes referred to as Feature Engineering.
3) Experiment with removing all records that have missing values; no imputation.
4) Try out a few other imputation methods, to see if different `Age` values are imputed, providing a better performing model.
5) Stacking multiple model types into one model for better performance.
6) Perform some sort of Feature Selection/Dimension Reduction; such as best subset.