Econ 412 Final Project: Survival on Titanic

Yinuo Song

1. Abstract

This report focuses on the prediction of survival on Titanic using five algorithms including logistic regression, support vector machine, decision tree, random forests and naïve Bayes. I first check the missing data and do the data imputation using representative statistics. Next I do the feature engineering and exploratory analysis to better know about the relationship between different variables. I then apply five algorithms to make prediction by building different models. And I find that the accuracy of Logistic Regression is 0.8539; the accuracy of Linear SVM is 0.8764; the accuracy of Non-linear Radial SVM is 0.8820; the accuracy of Decision Tree is 0.8371 without cv and 0.8427 with cv; the accuracy of Random Forest is 0.8371; the accuracy of Naive Bayes is 0.8427. The best performer of the five algorithms is from the non-linear SVM. And Logistic Regression, Linear SVM, Decision Tree, Random Forest and Naive Bayes executed roughly lower and close accuracy score.

2. Introduction

Titanic was a British passenger liner operated by the White Star Line. It sank in the North Atlantic Ocean on 15th April 1912 after striking an iceberg during her maiden voyage from Southampton to New York City. More than 1,500 of the estimated 2,224 passengers and crew died. It's one of modern history's deadliest peacetime commercial marine disasters. While there was some element of luck involved in surviving, it seems some groups of people Ire more likely to survive than others. Therefore, I are interested in the relationship between passengers' survival rates and personal characteristics and write this report to explore the relationship. Studying this topic helps us understand what will happen among different people when disasters happen.

There have been several studies on the survival on Titanic. Hall (1986) finds that females Ire more likely to survive than males, and the chances of survival are linked to the class in which the passengers traveled. Gleicher and Stevans (2004) demonstrate that women and children had higher survival rates than others. Frey, Savage and Torgler (2011) find that social norms Ire a key element in Titanic disaster, a life-or-death situation.

3. Methodology

3.1 Workflow Diagram

I draw the workflow diagram in Figure 1. First, I check the missing data of our original dataset to see if our data is balanced. Then I do the missing data imputation by choosing the representative statistics of the dataset. Third, I do the feature engineering, including the titles of passengers and family size. Next, I do the exploratory data analysis of the variables that I am interested in. The most important step is predicting the survival on Titanic using five algorithms, including logistic regression, support vector machine, decision tree, random forests and Naïve Bayes. Lastly, I discuss about the results of the predictions and compare between different methods.

3.2 Algorithms and Econometric Approaches

The five algorithms I mentioned above are popular when studying the survival in disasters. Therefore, I want to reproduce the results of previous studies by using these algorithms.

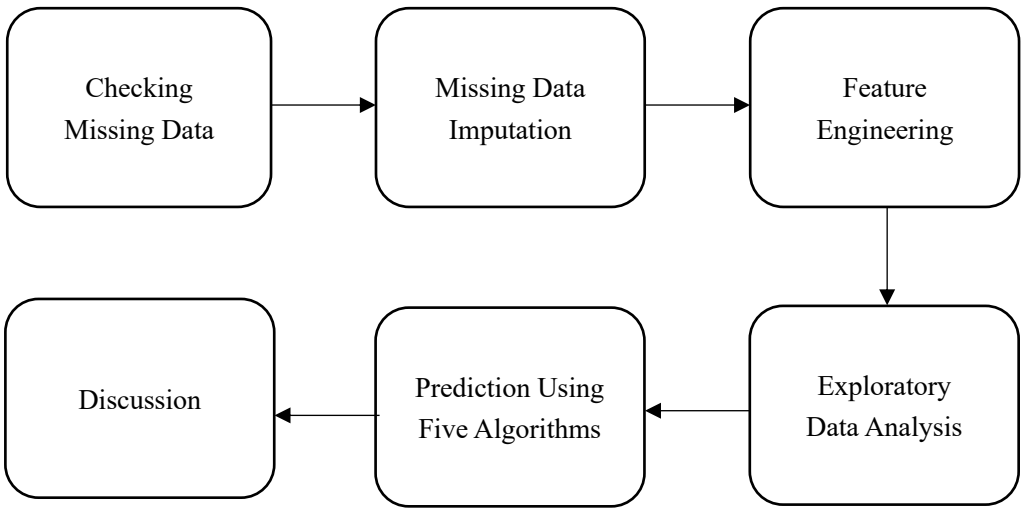
In statistics, the logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick (Wikipedia, n.d.). This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1 and the sum adding to one.

In machine learning, support-vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis (Wikipedia, n.d.). Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning (Wikipedia, n.d.). It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

Figure 1

Workflow Diagram



Random forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees (Wikipedia, n.d.). Random decision forests correct for decision trees' habit of overfitting to their training set.

In machine learning, naïve Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features (Wikipedia, n.d.). They are among the simplest Bayesian network models. But they could be coupled with Kernel density estimation and achieve higher accuracy levels.

4. Data

4.1 Dataset

I download the dataset from Kaggle, a machine learning and data science website. There're 1309 observations and 11 variables in total. The variables are: Survived, Pclass, name, sex, age, SibSp, parch, ticket, fare, cabin and embarked. The descriptions of the variables are shown in Table 1.

4.2 Checking Missing Data

I first use R to check if there're any missing data in our dataset. According to the result, I find that there're 418 missing values in survived, 263 missing values in age, 1 missing value in fare, 1014 missing values in cabin, and 2 missing values in embarked. I then draw the missing map of the dataset by using "missmap" function, which considers "NA" values as missing values, but it does not consider empty values as missing values (Figure 2). The missing map plot shows some of the age data is missing. However, the plot does not show cabin has missing values since missing cabin data are stored as empty values not NA values.

4.3 Missing Data Imputation

I first do the missing fare data imputation. I find that the passenger with missing fare data had a third-class ticket and embarked from Southampton. I then draw the fare distribution of

third-class passengers embarked from Southampton (Figure 3), and I decide to impute the missing fare by the median fare of third-class passengers embarked from Southampton.

Table 1

Variable Description

Variable	Description
Survived	Survival of the passenger. Yes = 1, No = 0
Pclass	Ticket class. 1st class = 1, 2nd class = 2, 3rd class = 3
Name	Name of the passenger
Sex	Sex of the passenger
Age	Age of the passenger in years
SibSp	Number of siblings / spouses aboard the Titanic
Parch	Number of parents / children aboard the Titanic
Ticket	Ticket number
Fare	Passenger fare
Cabin	Cabin number
Embarked	Port of embarkation. Cherbourg = C, Queenstown = Q, Southampton = S

Figure 2

Missing Map of Dataset

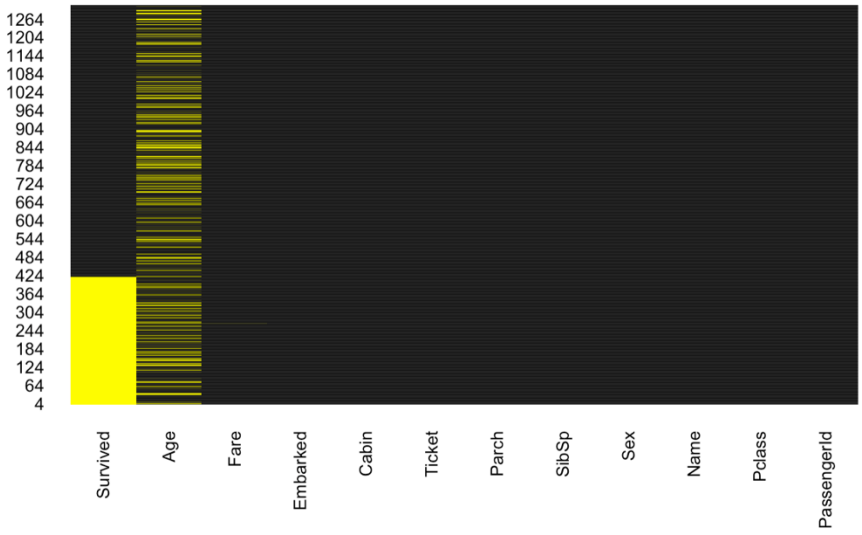
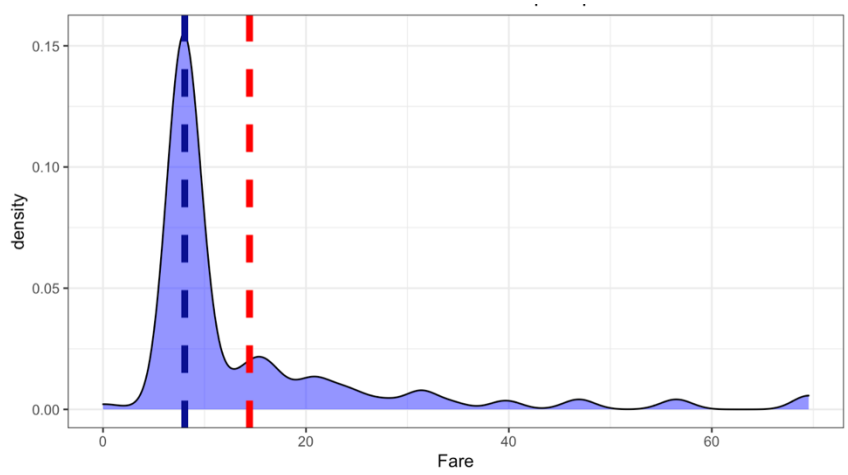


Figure 3

Fare Distribution of Third-Class Passengers Embarked from Southampton



After that, I deal with the missing values of embarked data. I find that both passengers with missing embarked data are first-class females with $80 fare and stayed in the same cabin B28. I then explore the embarked data of first-class passengers, and I find that Southampton is the most popular port and Cherbourg is the second-popular port. After investigating the fare distribution of first-class passengers (Figure 4), I find that the median fare for Cherbourg port passengers and $80 fare paid by two embarkment-deficient passengers almost coincide. Therefore, I impute the missing embarked data using Cherbourg.

I then deal with the missing values of age data. To analyze it, I first draw the boxplot of age distribution grouped by Pclass (Figure 5). I find that there're differences between median age among different classes, and the passengers in higher classes are older than others. Hence, I choose to use average age of each class to impute missing age data.

4.4 Feature Engineering

Since the title of the passengers is contained within the passenger name feature, I create a new variable of passengers' titles. I find that there're 264 passengers with title "Miss", 198 with title "Mrs", 61 with title "Master", 757 with title "Mr" and 29 with other titles.

I also create a new variable named FamilySize. There're too many categories of family size, and I merge them into three classes: Large, small and single. There're 82 large families, 437 small families and 790 singles.

Figure 4

Fare Distribution of First-Class Passengers

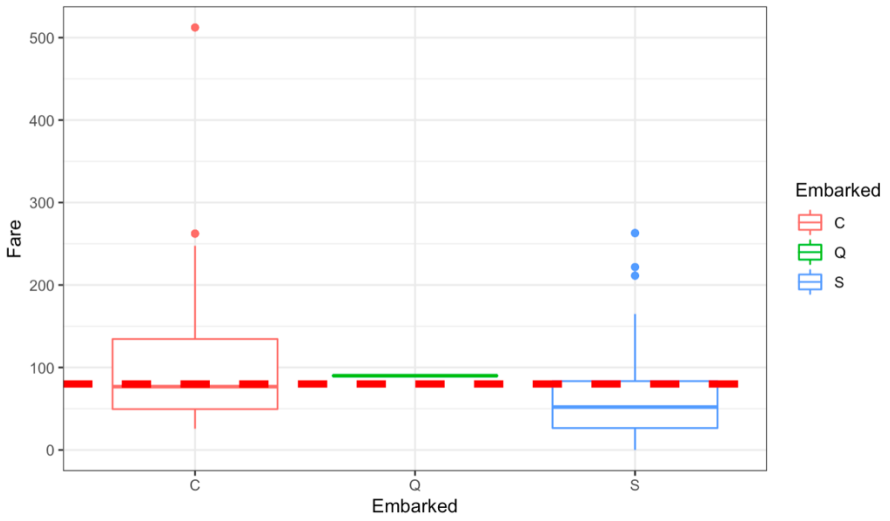
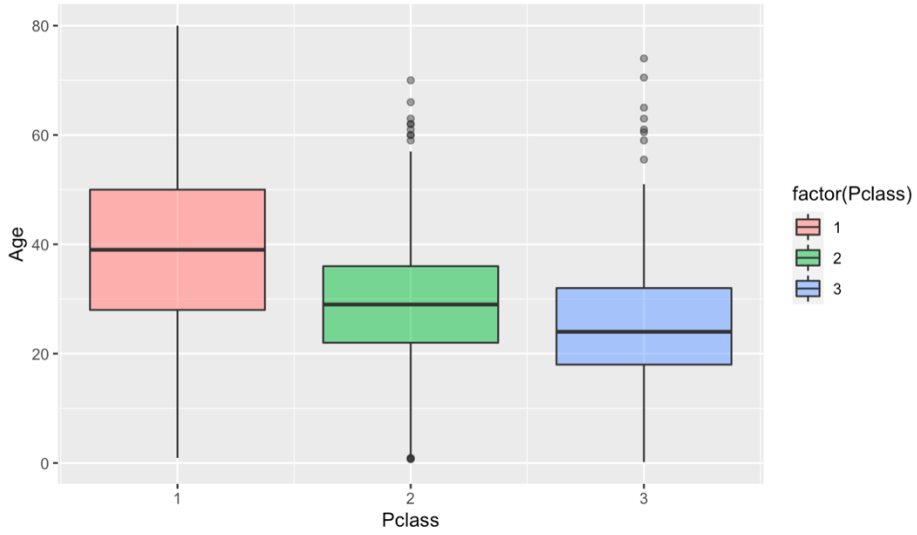


Figure 5

Boxplot of Age Distribution Grouped by Pclass



4.5 Exploratory Data Analysis

I first analyze the relationship between Pclass and survival rate. As I can see in Figure 6, the survival rate for first-class passengers is the highest. In Figure 7, I find that females have much higher survival rates than males among all classes. Overall, the passengers in the higher classes tend to be older disregard to whether they survived or not (Figure 8).

Figure 6

Survival Rate Based on Pclass

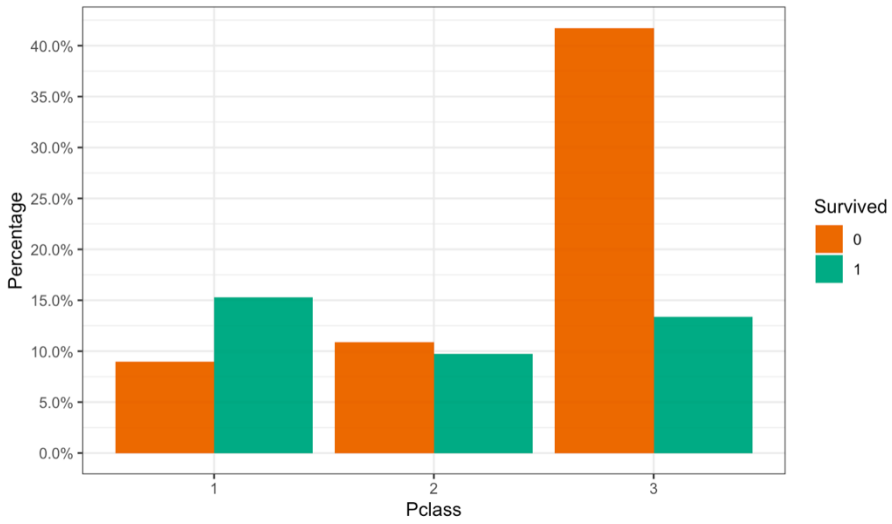


Figure 7

Survival Rate Based on Pclass and Sex

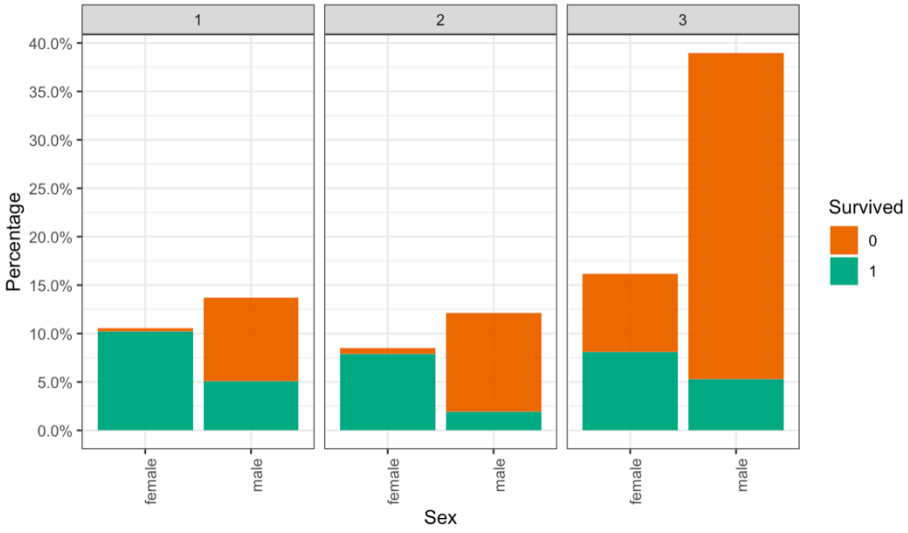
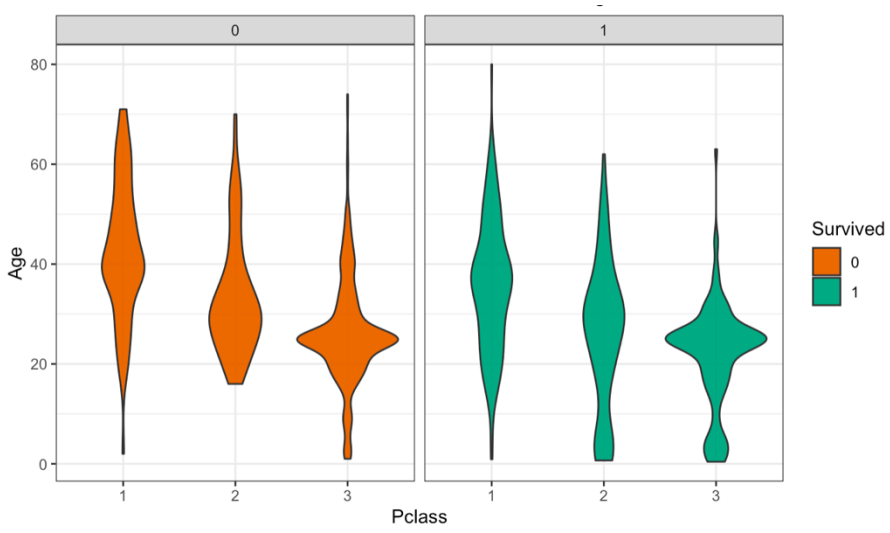


Figure 8

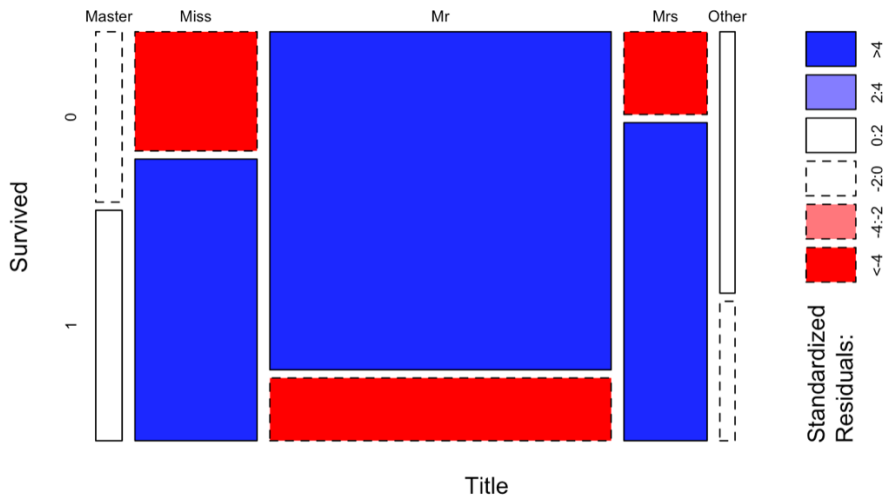
Survival Rate Based on Pclass and Age



I then do the EDA of passengers' title (Figure 9). I find that passengers with title "Mr" have the lowest survival rate, which is consistent with the history that women and children used the emergency boat first.

Figure 9

Survival Rate Based on Pclass and Title



I then analyze the survival rate based on family size (Figure 10). I notice that large families have the lowest survival rates and small families have higher survival rates. From Figure 11, I find that there is a substantial variation of fares in the survived category, especially from Cherbourg and Southampton ports.

Figure 10

Survival Rate Based on Pclass and Family Size

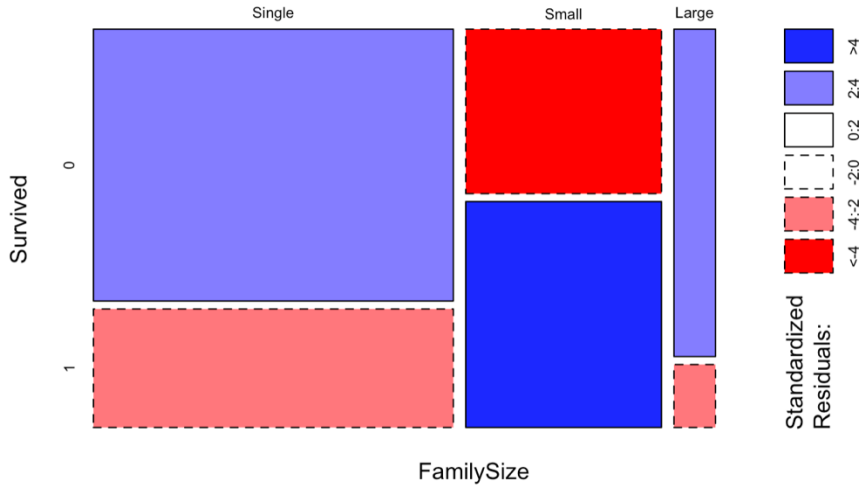
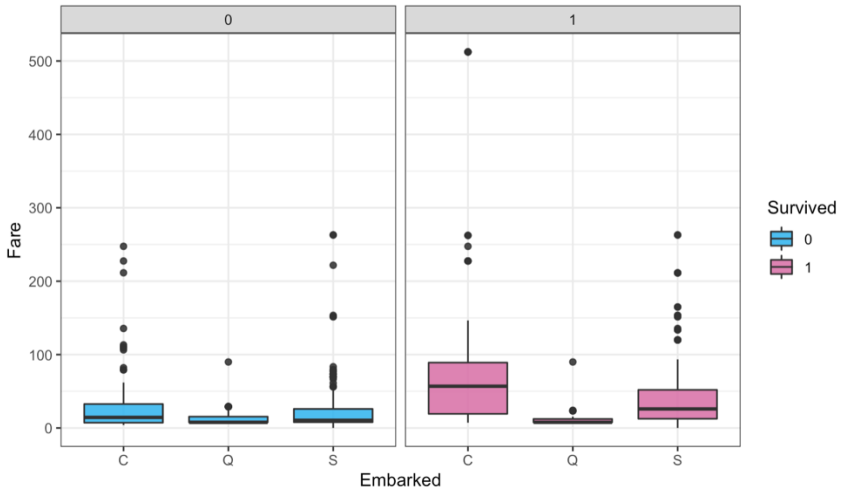


Figure 11

Survival Rate Based on Embarked and Fare



5. Prediction Using Five Algorithms

5.1 Logistic Regression

I split the dataset into two sets: Training set and testing set. After that, I calculate the correlation matrix of continuous variables (Table 2). None of the continuous variables are strongly correlated. I then use Chi Square test to test the independence of factors/categorical features. Since all the p-values < 0.05, I reject each Ho: Two factors are independent at 5% significance level and indeed at any reasonable level of significance. This violates the

independence assumption of features and can be confirmed that multicollinearity does exist among factors.

Table 2

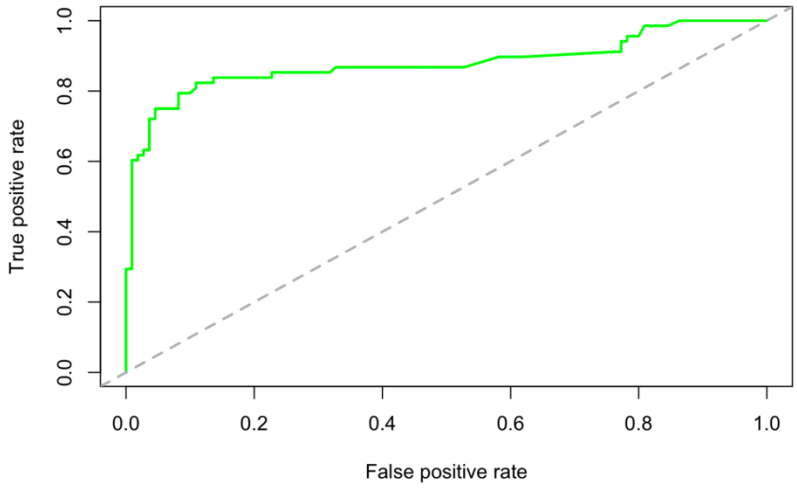
Correlation Matrix

	Age	SibSp	Parch	Fare
Age	1.00	-0.28	-0.21	0.11
SibSp	-0.28	1.00	0.45	0.16
Parch	-0.21	0.45	1.00	0.24
Fare	0.11	0.16	0.24	1.00

According to the best model, the features Pclass, Age, Title and FamilySize significantly contribute to the model in predicting survival. I will see how Ill the model predicts on new data in the validation set.

Figure 12

Survival Rate Based on Embarked and Fare



The ROC (Receiver Operating Characteristics) curve is a graphical representation of the performance of the classifier and it shows the performance of our model rises Ill above the diagonal line (Figure 12). This indicates that our logistic regression model performs better than

just a random guess. The logistic regression model delivers a whooping 0.8539 accuracy in terms of predicting the survival.

5.2 Support Vector Machines

Secondly, I use Support Vector Machines (SVM) for classification. In order to use SVM, I need to do feature scaling, because the SVM classifier predicts the class of a given test observation by identifying the observations that are nearest to it, the scale of the variables matters. Then I fit both linear SVM and non-linear SVM models to predict survival. Compared to the linear model, the accuracy of non-linear has been improved by 1% from 0.8764 to 0.8820. Also, I note that both linear SVM and non-linear SVM accuracies are higher than the accuracy for logistic regression model (Table 3).

Table 3

SVM Accuracy Matrix

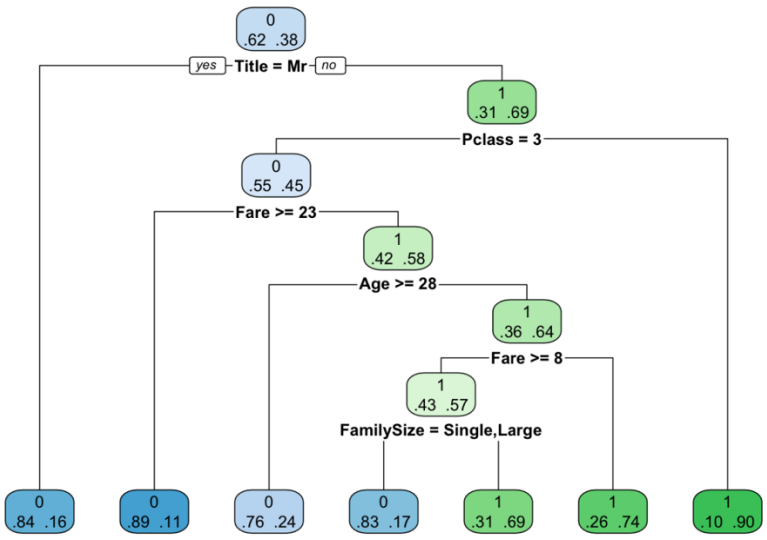
	Accuracy
Linear SVM	0.8764
Non-Linear SVM	0.8820

5.3 Decision Tree

Random Forest is a more powerful algorithm over just a single tree. However, the Decision Tree classification preserve the interpretability which the random forest algorithm lacks. The Decision Tree does not require feature scaling. I directly fit a decision tree model to our training data (Figure 13 & Figure 14). The single tree uses five features Title, Pclass, Fare, Age and FamilySize for classification. Then I try to find how Ill our model performs with the data in the validation set.

Figure 13

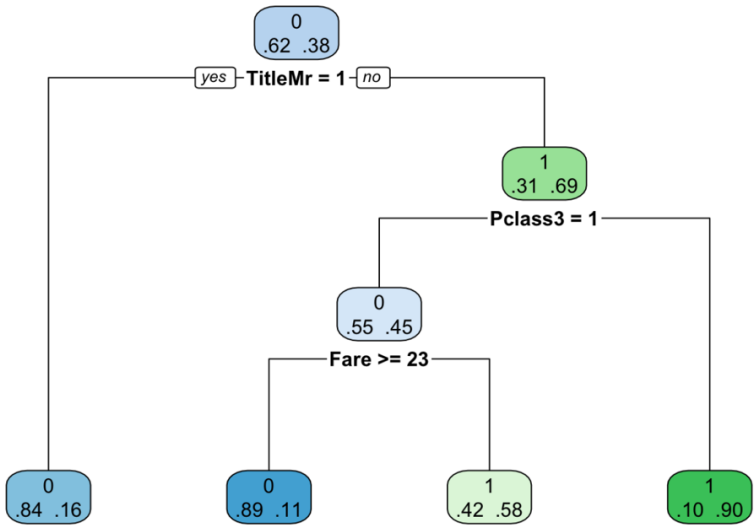
Decision Tree Without k-Fold Cross Validation



Accuracy of a single tree without k-Fold Cross Validation is 0.8371 (Figure 13). Overfitting can easily occur in Decision Tree classification. Therefore, I can identify that evaluating the model using k-Fold Cross Validation. Or I might be able to improve the model. I could do 10-fold cross validation to find out whether I could improve the model.

Figure 14

Decision Tree With k-Fold Cross Validation



I are able to improve the model after 10-fold cross validation (Figure 14). The accuracy has been improved to 0.8427 but note the improved model uses only three features Title, Pclass and Fare for classification (Table 4).

Table 4

Decision Tree Accuracy Matrix

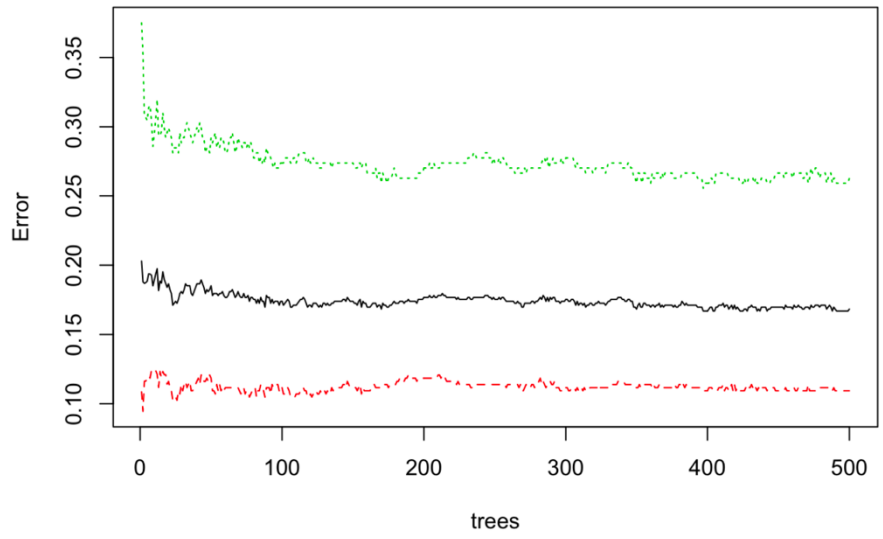
	Accuracy
Decision Tree Without k-Fold Cross Validation	0.8371
Decision Tree With k-Fold Cross Validation	0.8427

5.4 Random Forests

Random forests improve predictive accuracy by generating a large number of bootstrapped trees (based on random samples of variables). Random Forest is a powerful machine learning algorithm which holds a relatively high classification accuracy.

Figure 15

Random Forests Classifier



The green, black and red lines represent error rate for death, overall and survival, respectively. The overall error rate converges to around 17% (Figure 15). Interestingly, our model predicts death better than survival. Since the overall error rate converges to a constant and does not seem to further decrease, our choice of default 500 trees in the Random Forest function is a good choice.

The accuracy is 0.8371 and which is equal to the accuracy of just a single tree 0.8371 without 10-fold cross validation (Table 5). Then I try to find whether the 10-fold cross validation can improve our model as it did for the Decision Tree classification. Accuracy Int down to 0.8034. I am not able to improve the random forest model using 10-fold cross validation (Table 5).

Table 5

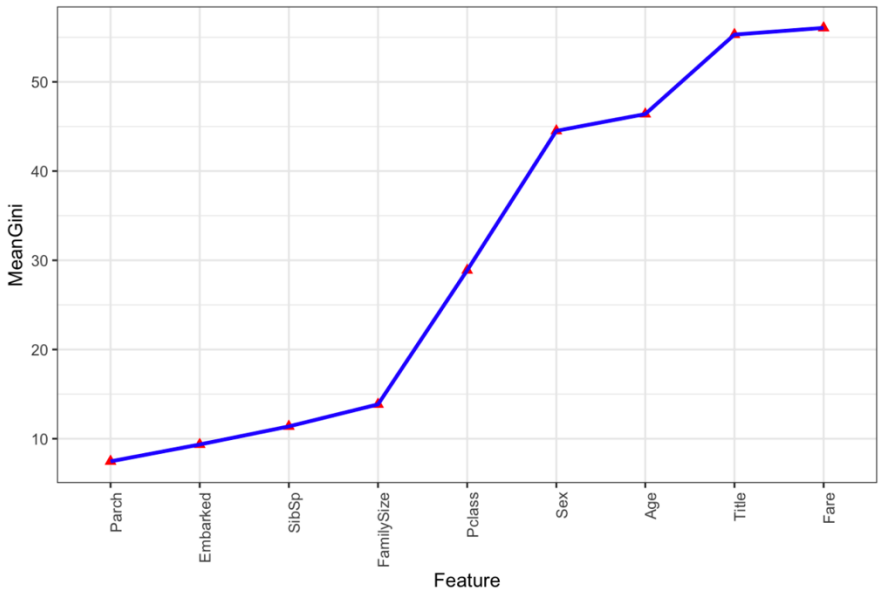
Random Forests Accuracy Matrix

	Accuracy
Random Forests Without k-Fold Cross Validation	0.8371
Random Forests With k-Fold Cross Validation	0.8034

As mentioned previously in the Decision Tree section, the random Forest classification suffers in terms of interpretability. I are unable to visualize the 500 trees and identify important features of the model. However, I can assess the Feature Importance using the Gini index measure. Therefore, I then plot the mean Gini index across all trees and identify important features (Figure 17). The feature Title has the highest mean Gini index, hence the highest importance. Fare is also relatively high important, and it is followed by Age of the passengers.

Figure 16

Random Forests' Mean Gini Index of Features



5.5 Naive Bayes

Lastly, I use Naive Bayes algorithm to predict survival. Naive Bayes classification is a simple but effective algorithm; it is faster compared to many other iterative algorithms; it does not need feature scaling; and its foundation is the Bayes Theorem.

However, Naive Bayes is based on the assumption that conditional probability of each feature given the class is independent of all the other features. The assumption of independent conditional probabilities means the features are completely independent of each other. This assumption was already checked in the Logistic Regression section and I have found that numeric features are independent to each other, however, the categorical features are not. By assuming the independence assumption of all the features, let's fit a naive Bayes model to our training data.

Table 6

Naive Bayes Accuracy Matrix

	Accuracy
Naive Bayes Accuracy	0.8427

Naive Bayes classification performs Ill for our validation data with an accuracy of 0.8427. Note that the accuracy is identical to the decision tree with 10-Fold cross validation accuracy (Table 6).

6. Discussion

6.1 Comparison of models

Logistic Regression: The assumptions are hold and the accuracy of the best model is 0.8539; Four features, Title, Pclass, Age and FamilySize, significantly contribute to the model in predicting survival; Confusion matrix consists of 9 false positives and 17 false negatives.

Linear SVM: Scaled some features and gained an accuracy of 0.8764; Confusion matrix consists of 4 false positives and 18 false negatives. Non-linear Radial SVM: Scaled some features and secured the highest accuracy of 0.8820; Parameter tuning did not improve the model accuracy; Confusion matrix consists of 5 false positives and 16 false negatives.

Decision Tree: Ire able to improve the model using 10-fold cross validation; The accuracy was improved from 0.8371 to 0.8427 and the improved model uses three features, Title, Fare and Pclass, for classification; Confusion matrix consists of 11 false positives and 17 false negatives.

Random Forest: Gained an accuracy of 0.8371 and 10-fold cross validation did not improve the model accuracy; The first five important features based on their importance (highest to lowest) are as follows: Title, Fare, Age, Sex and Pclass; Confusion matrix consists of 10 false positives and 18 false negatives.

Naive Bayes: Gained an accuracy of 0.8427 which is identical to the Random Forest accuracy; The independence among features was assumed; Confusion matrix consists of 11 false positives and 17 false negatives.

6.2 Conclusion

The best performer of the five algorithms is from the non-linear SVM. And Logistic Regression, Linear SVM, Decision Tree, Random Forest and Naive Bayes executed roughly lower and close accuracy score.

7. References

Hall, W. (1986). Social class and survival on the SS Titanic. *Social science & medicine, 22(6), 687-690.*

Gleicher, D., & Stevans, L. K. (2004). Who survived Titanic? A logistic regression analysis. *International Journal of Maritime History*, *16*(2), 61-94.

Frey, B. S., Savage, D. A., & Torgler, B. (2011). Who perished on the Titanic? The importance of social norms. *Rationality and society*, *23*(1), 35-49.

Wikipedia. (n.d.). Decision tree learning. https://en.wikipedia.org/wiki/Decision_tree_learning

Wikipedia. (n.d.). Logistic regression. https://en.wikipedia.org/wiki/Logistic_regression

Wikipedia. (n.d.). Naïve Bayes classifier. https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Wikipedia. (n.d.). Random forest. https://en.wikipedia.org/wiki/Random_forest

Wikipedia. (n.d.). Support vector machine. https://en.wikipedia.org/wiki/Support_vector_machine

Titanic Survival Prediction

Yinuo Song

5/15/2020

```
rm(list=ls())
```

```
getwd()
```

```
## [1] "/Users/yinuo/Desktop"
```

```
library(dplyr)
library(Amelia)
library(ggplot2)
library(scales)
library(caTools)
library(car)
library(ROCR)
library(e1071)
library(rpart)
library(rpart.plot)
library(randomForest)
library(caret)
```

Import data

```
train <- read.csv('train.csv', stringsAsFactors = F)
test  <- read.csv('test.csv', stringsAsFactors = F)
titanic  <- bind_rows(train, test)
str(titanic)
```

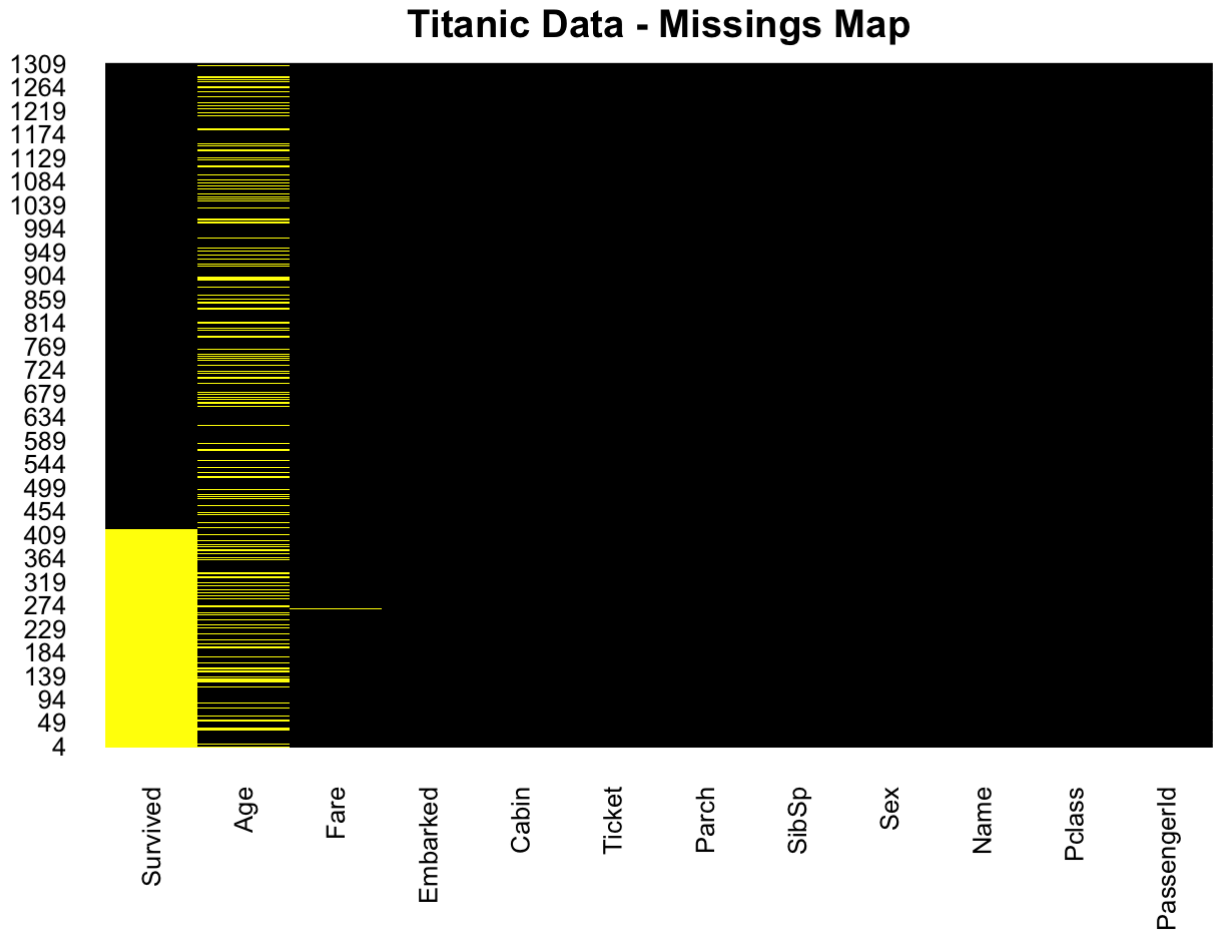
```
## 'data.frame':    1309 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence
Briggs Thayer)" "Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques Heath (Lily May Peel)"
...
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

Check missing data

```
colSums(is.na(titanic)|titanic=='')
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0         418           0           0           0         263
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           1        1014           2
```

```
missmap(titanic, main="Titanic Data - Missings Map",
        col=c("yellow", "black"), legend=FALSE)
```



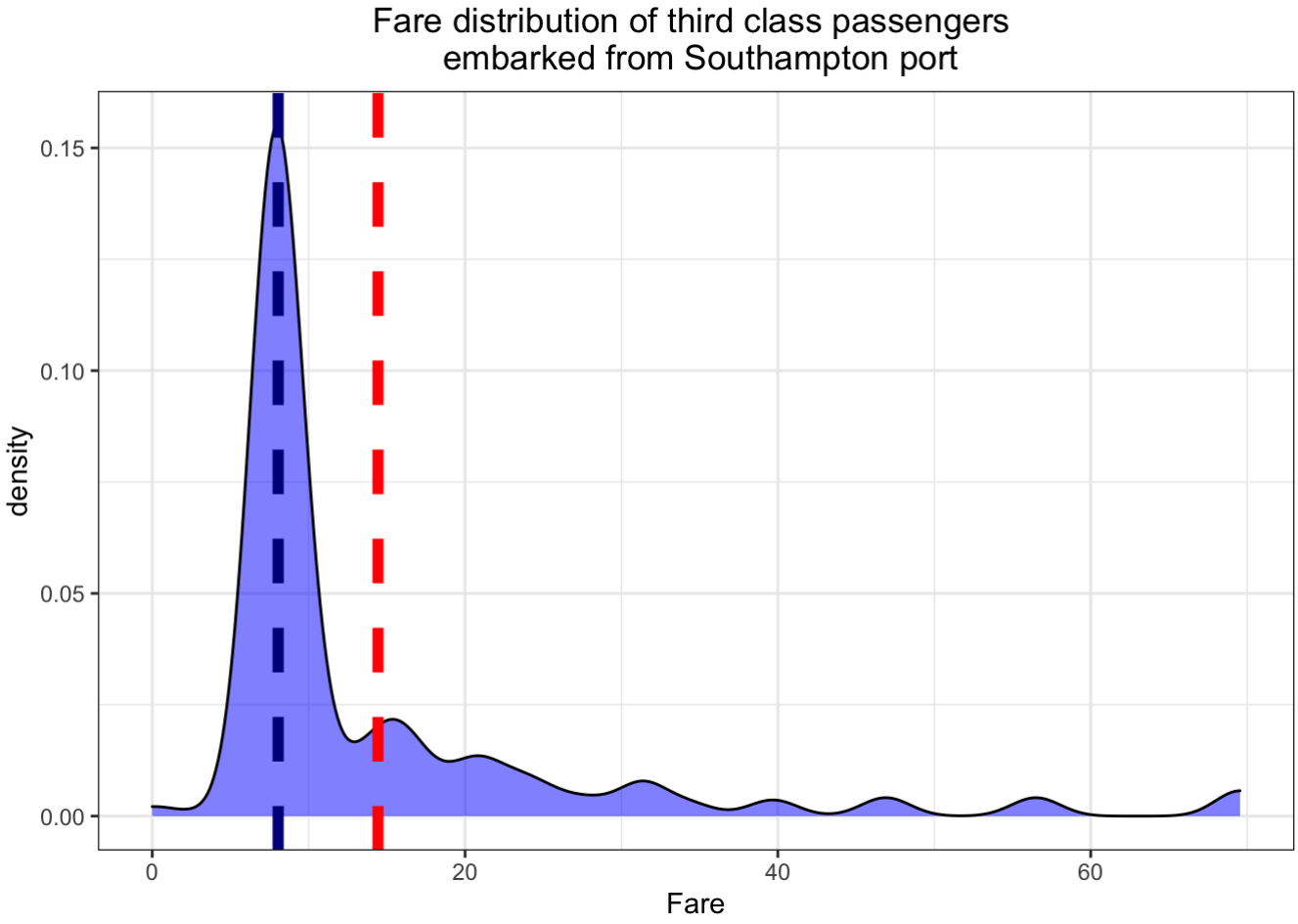
Missing fare data imputation

```
filter(titanic, is.na(Fare)==TRUE|Fare=='')
```

PassengerId	Survived	Pclass	Name	Sex	Age	Sib…	Par…	Ticket	F…
<int>	<int>	<int>	<chr>	<chr>	<dbl>	<int>	<int>	<chr>	<dbl>
1044	*NA*	3	Storey, Mr. Thomas	male	60.5	0	0	3701	*NA*

1 row | 1-10 of 12 columns

```
ggplot(filter(titanic, Pclass==3 & Embarked=="S"), aes(Fare)) +
  geom_density(fill="blue", alpha=0.5) +
  geom_vline(aes(xintercept=median(Fare, na.rm=T)), colour='darkblue', linetype='dashed'
, size=2) +
  geom_vline(aes(xintercept=mean(Fare, na.rm=T)), colour='red', linetype='dashed', size=
2) +
  ggtitle("Fare distribution of third class passengers \n embarked from Southampton por
t") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```




```
titanic$Fare[is.na(titanic$Fare)==TRUE] = median(filter(titanic, Pclass==3 & Embarked==
"S")$Fare, na.rm=TRUE)
colSums(is.na(titanic)|titanic=='')
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0         418           0           0           0         263
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           0        1014           2
```

Missing embarked data imputation

```
filter(titanic, is.na(Embarked)==TRUE|Embarked=='')
```

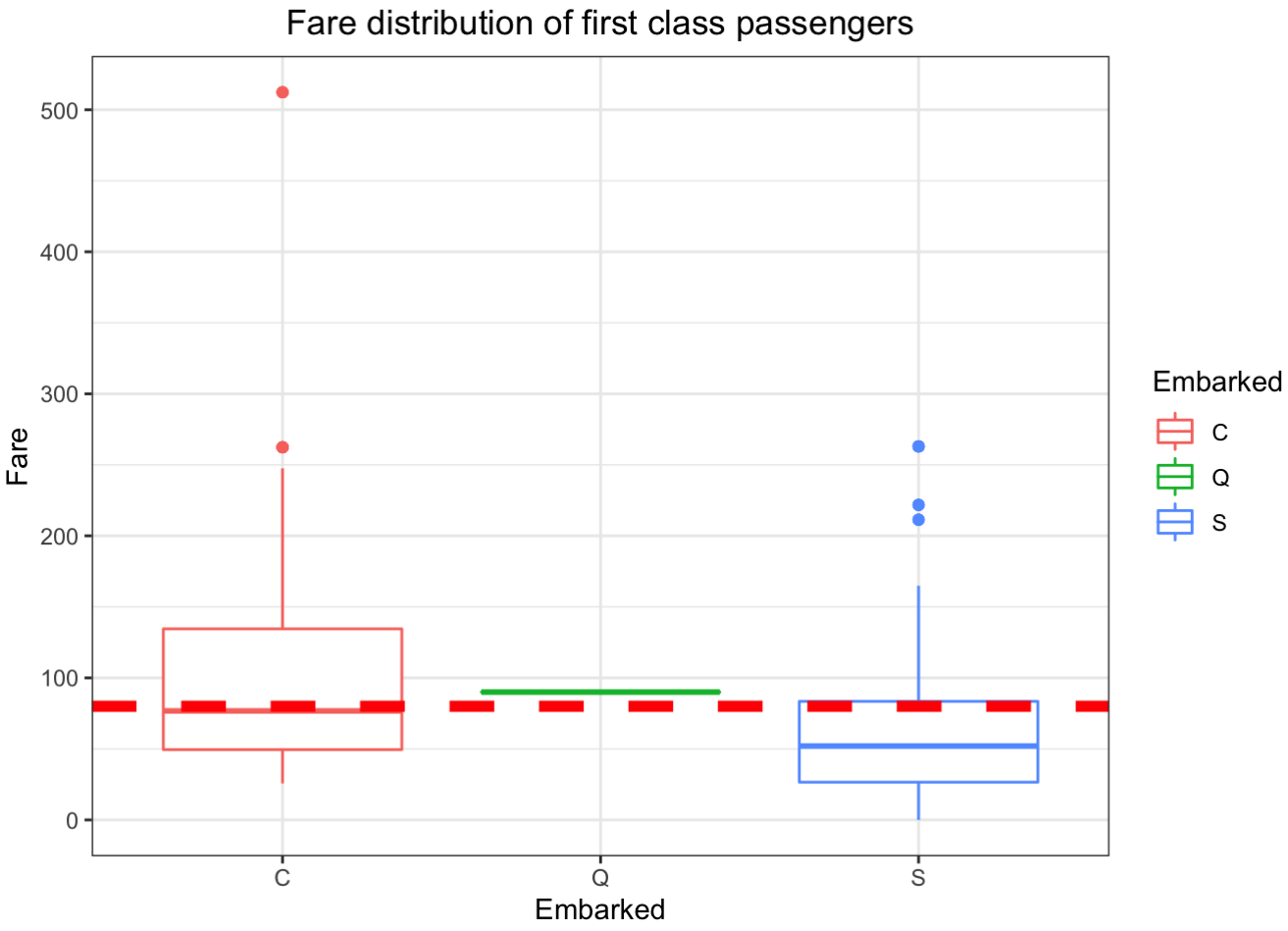
PassengerId	Survived	Pcla...	Name	Sex	...	Si
<int>	<int>	<int>	<chr>	<chr>	<dbl>	<in
62	1	1	Icard, Miss. Amelie	female	38	
830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62	

2 rows | 1-8 of 12 columns

```
table(filter(titanic, Pclass==1)$Embarked)
```

```
##
##       C   Q   S
##   2 141   3 177
```

```
ggplot(filter(titanic, is.na(Embarked)==FALSE & Embarked!='' & Pclass==1),
       aes(Embarked, Fare)) +
  geom_boxplot(aes(colour = Embarked)) +
  geom_hline(aes(yintercept=80), colour='red', linetype='dashed', size=2) +
  ggtitle("Fare distribution of first class passengers") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```

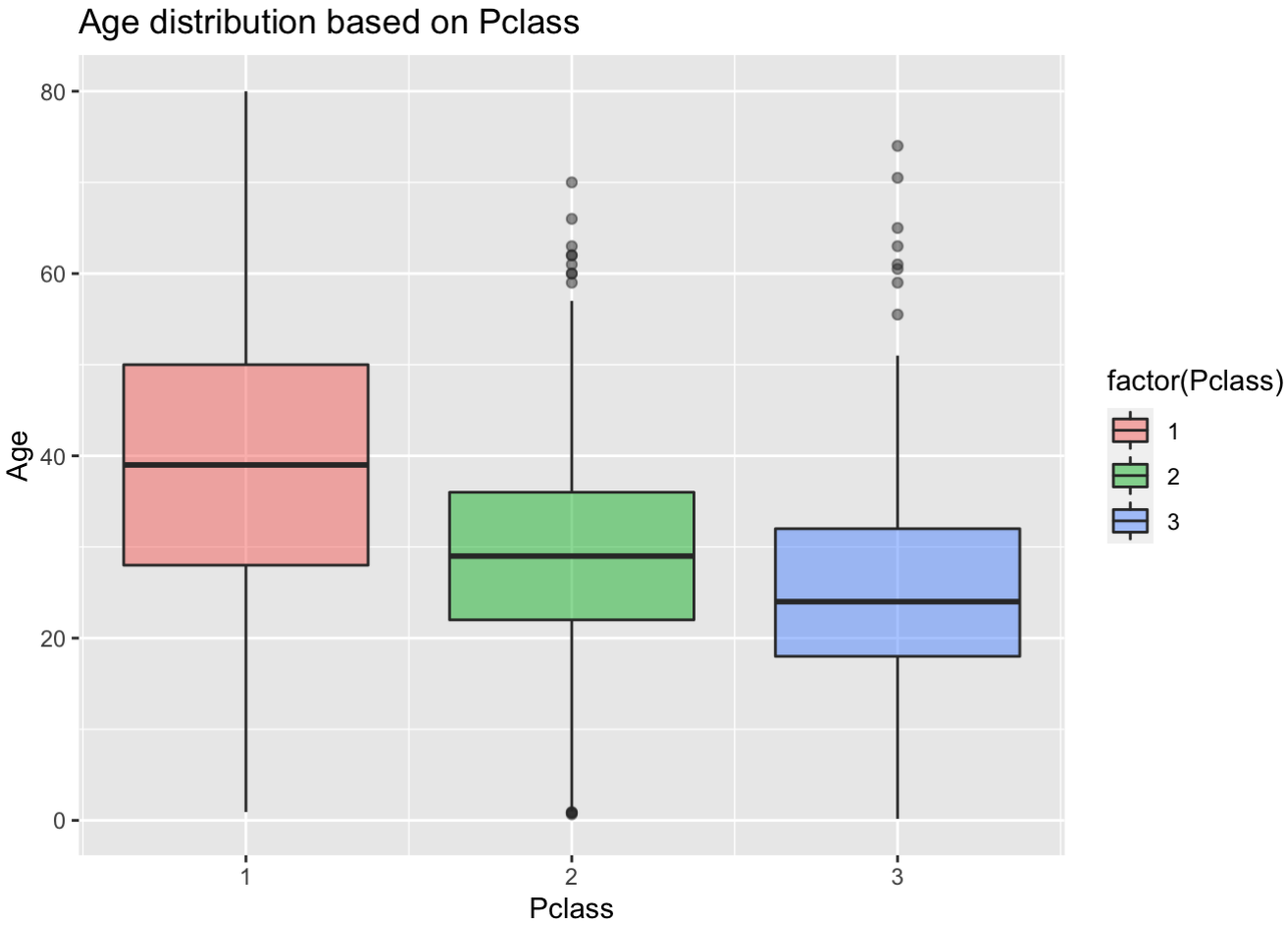


```
titanic$Embarked[titanic$Embarked==""] = "C"
colSums(is.na(titanic)|titanic=='')
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0         418           0           0           0         263
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           0        1014           0
```

Missing age data imputation

```
ggplot(titanic,aes(Pclass,Age)) +
  geom_boxplot(aes(fill=factor(Pclass)),alpha=0.5) +
  ggtitle("Age distribution based on Pclass")
```



```
impute.age <- function(age,class){
  vector <- age
  for (i in 1:length(age)){
    if (is.na(age[i])){
      if (class[i] == 1){
        vector[i] <- round(mean(filter(titanic,Pclass==1)$Age, na.rm=TRUE),0)
      }else if (class[i] == 2){
        vector[i] <- round(mean(filter(titanic,Pclass==2)$Age, na.rm=TRUE),0)
      }else{
        vector[i] <- round(mean(filter(titanic,Pclass==3)$Age, na.rm=TRUE),0)
      }
    }else{
      vector[i]<-age[i]
    }
  }
  return(vector)
}
imputed.age <- impute.age(titanic$Age,titanic$Pclass)
titanic$Age <- imputed.age
```

```
colSums(is.na(titanic)|titanic=='')
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0         418           0           0           0           0
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           0        1014           0
```

```
head(titanic$Name)
```

```
## [1] "Braund, Mr. Owen Harris"
## [2] "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
## [3] "Heikkinen, Miss. Laina"
## [4] "Futrelle, Mrs. Jacques Heath (Lily May Peel)"
## [5] "Allen, Mr. William Henry"
## [6] "Moran, Mr. James"
```

```
titanic$Title <- gsub("^.*, (.*?)\\..*$", "\\1", titanic$Name)
table(titanic$Sex, titanic$Title)
```

```
##
##          Capt Col Don Dona  Dr Jonkheer Lady Major Master Miss Mlle Mme  Mr Mrs
##   female    0   0   0    1   1        0    1     0      0  260    2   1   0 197
##   male      1   4   1    0   7        1    0     2     61    0    0   0 757   0
##
##           Ms Rev Sir the Countess
##   female   2   0   0            1
##   male     0   8   1            0
```

Title

```
titanic$Title[titanic$Title == 'Mlle' | titanic$Title == 'Ms'] <- 'Miss'
titanic$Title[titanic$Title == 'Mme']  <- 'Mrs'
Other <- c('Dona', 'Dr', 'Lady', 'the Countess','Capt', 'Col', 'Don', 'Jonkheer', 'Majo
r', 'Rev', 'Sir')
titanic$Title[titanic$Title %in% Other]  <- 'Other'
table(titanic$Sex, titanic$Title)
```

```
##
##          Master Miss  Mr Mrs Other
##   female      0  264   0 198     4
##   male       61    0 757   0    25
```

Family size

```
FamilySize <- titanic$SibSp + titanic$Parch + 1
table(FamilySize)
```

```
## FamilySize
##   1   2   3   4   5   6   7   8  11
## 790 235 159  43  22  25  16   8  11
```

```
titanic$FamilySize <- sapply(1:nrow(titanic), function(x)
                         ifelse(FamilySize[x]==1, "Single",
                         ifelse(FamilySize[x]>4, "Large", "Small")))

table(titanic$FamilySize)
```

```
##
##  Large Single  Small
##     82    790    437
```

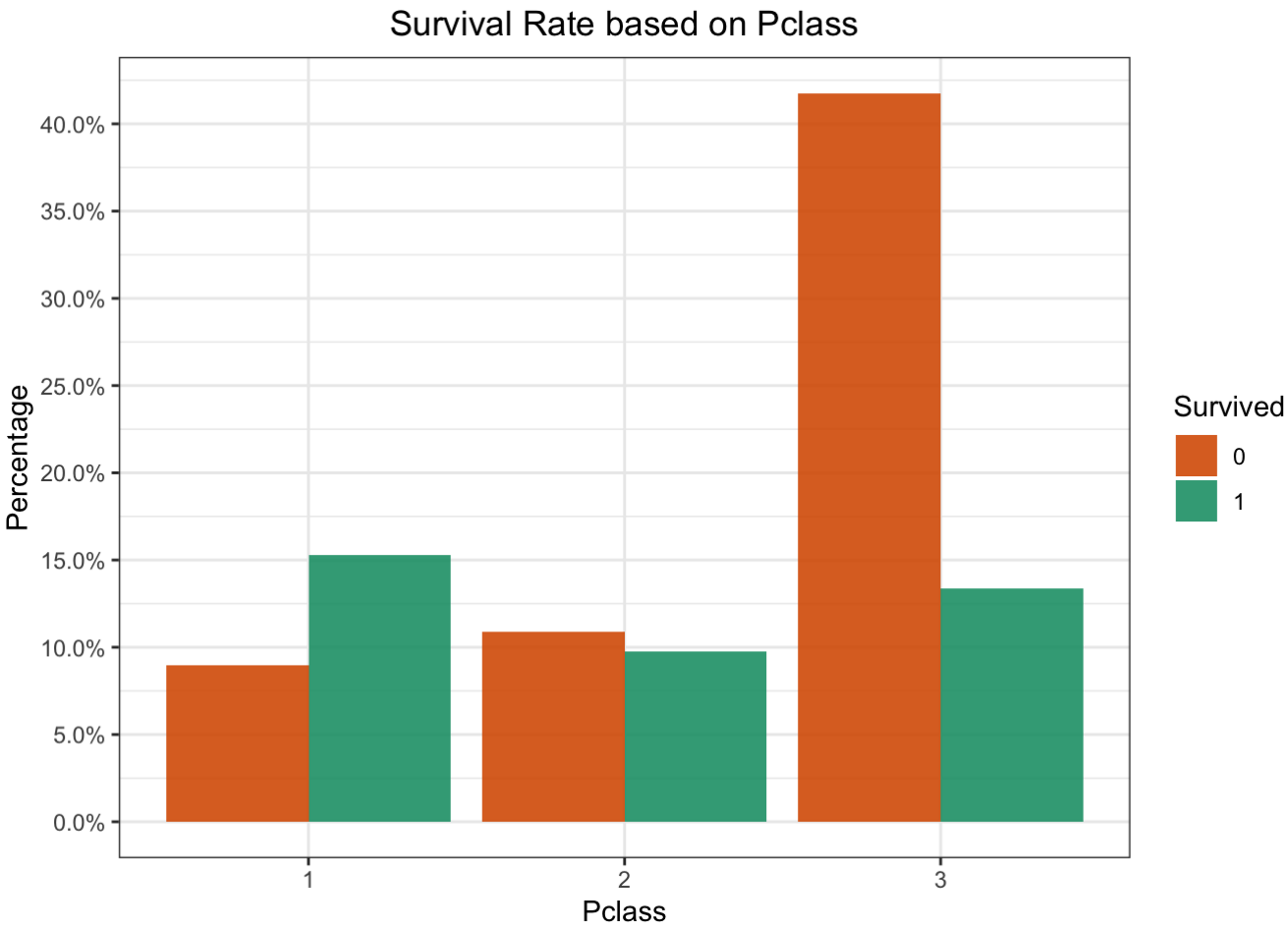
```
titanic$Survived = factor(titanic$Survived)
titanic$Pclass = factor(titanic$Pclass)
titanic$Sex = factor(titanic$Sex)
titanic$Embarked = factor(titanic$Embarked)
titanic$Title = factor(titanic$Title)
titanic$FamilySize = factor(titanic$FamilySize, levels=c("Single","Small","Large"))

str(titanic)
```

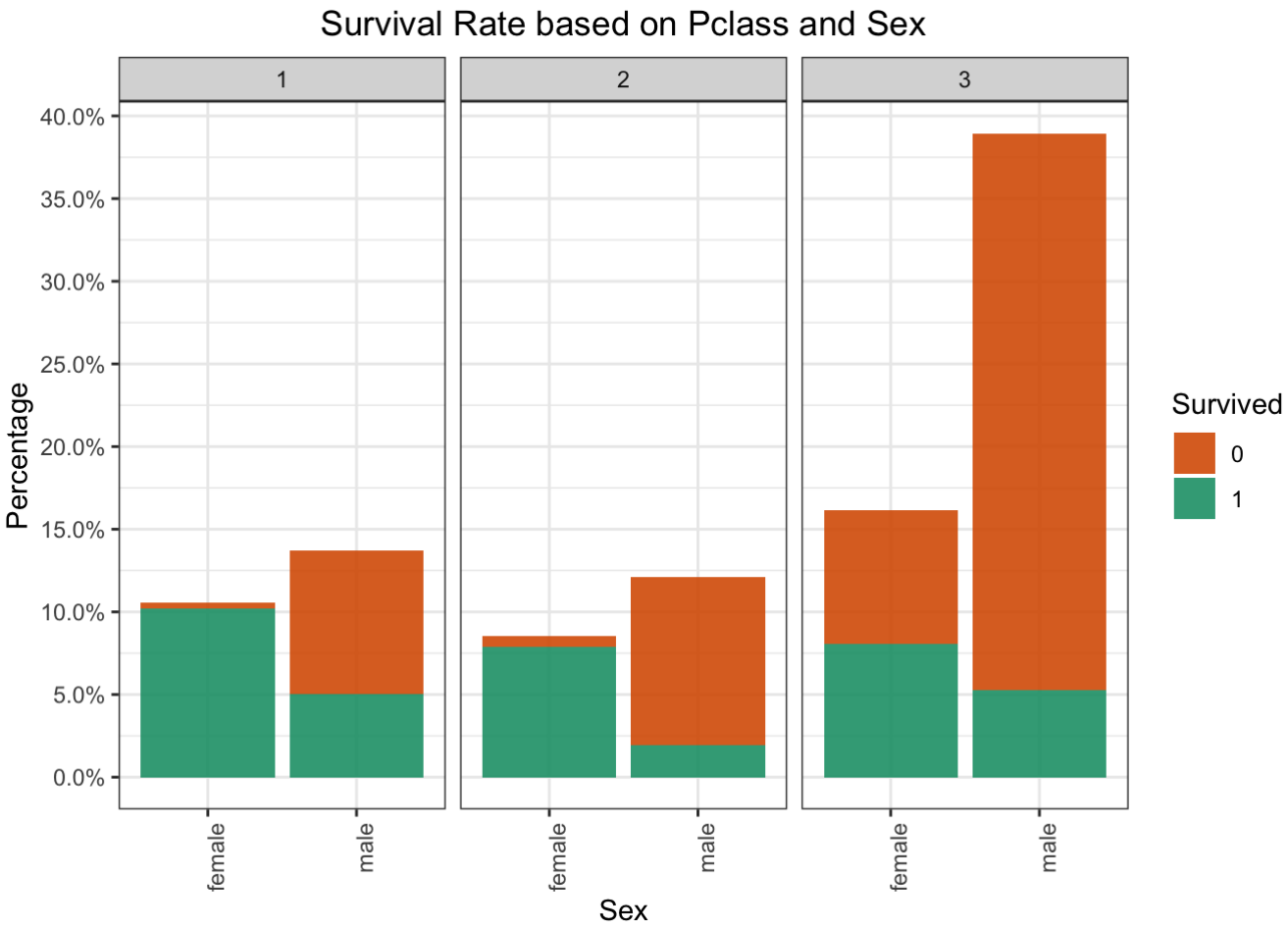
```
## 'data.frame':    1309 obs. of  14 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ Pclass     : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence
Briggs Thayer)" "Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques Heath (Lily May Peel)"
...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 25 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : Factor w/ 3 levels "C","Q","S": 3 1 3 3 3 2 3 3 3 1 ...
##  $ Title      : Factor w/ 5 levels "Master","Miss",..: 3 4 2 4 3 3 3 1 4 4 ...
##  $ FamilySize : Factor w/ 3 levels "Single","Small",..: 2 2 1 2 1 1 1 3 2 2 ...
```

EDA

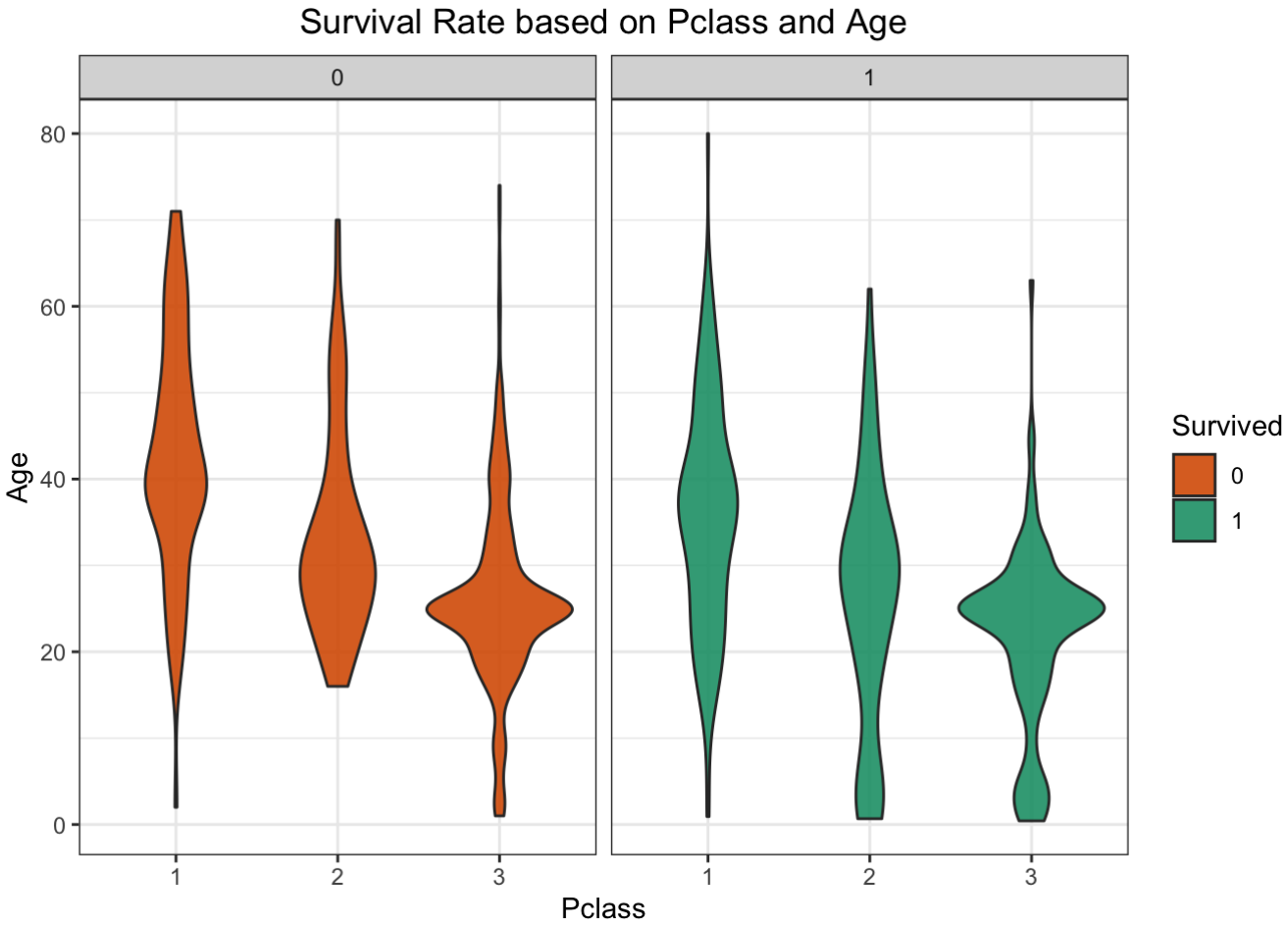
```
ggplot(filter(titanic, is.na(Survived)==FALSE), aes(Pclass, fill=Survived)) +
  geom_bar(aes(y = (..count..)/sum(..count..)), alpha=0.9, position="dodge") +
  scale_fill_brewer(palette = "Dark2", direction = -1) +
  scale_y_continuous(labels=percent, breaks=seq(0,0.6,0.05)) +
  ylab("Percentage") +
  ggtitle("Survival Rate based on Pclass") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```



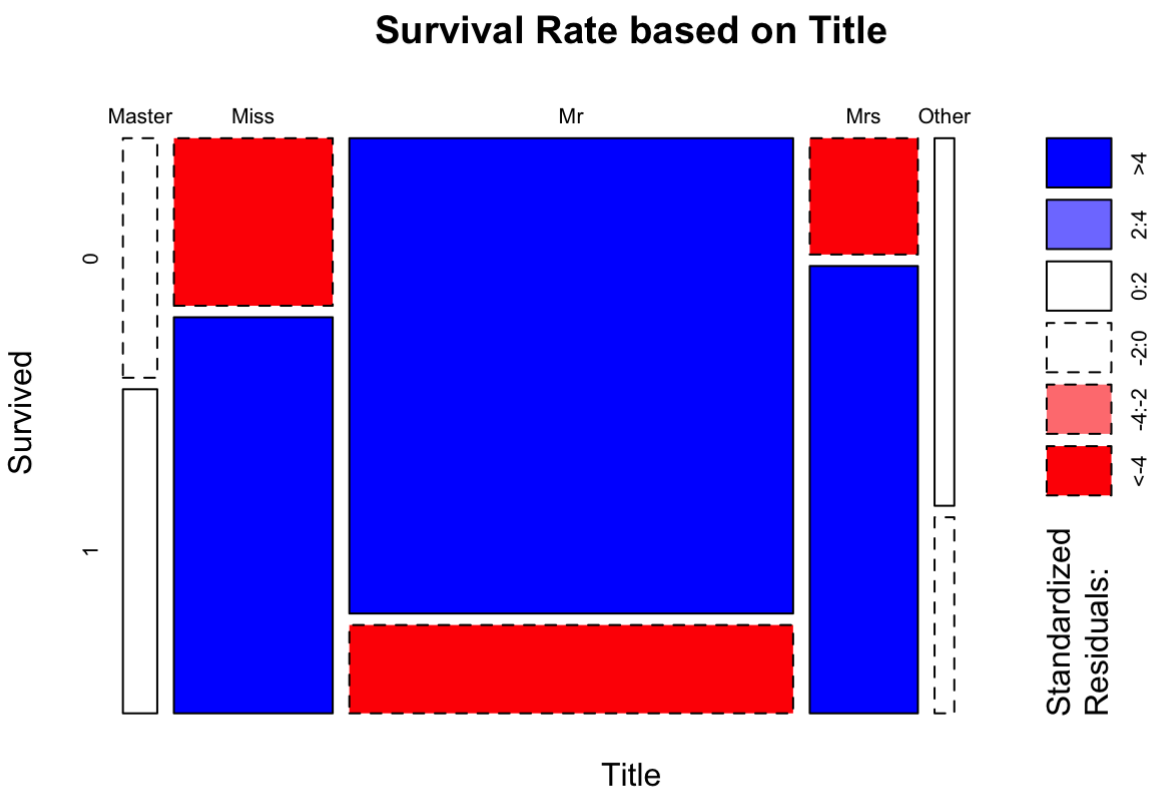

```
ggplot(filter(titanic, is.na(Survived)==FALSE), aes(Sex, fill=Survived)) +
  geom_bar(aes(y = (..count..)/sum(..count..)), alpha=0.9) +
  facet_wrap(~Pclass) +
  scale_fill_brewer(palette = "Dark2", direction = -1) +
  scale_y_continuous(labels=percent, breaks=seq(0,0.4,0.05)) +
  ylab("Percentage") +
  ggtitle("Survival Rate based on Pclass and Sex") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



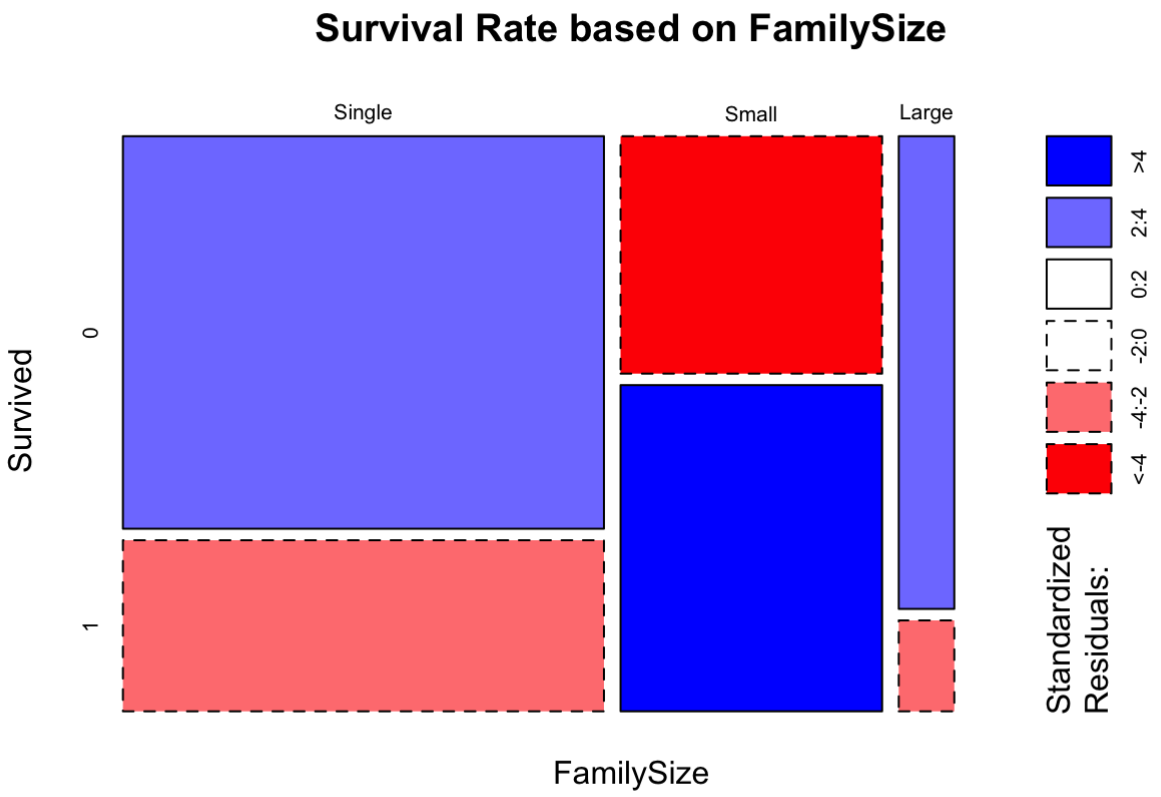

```
ggplot(filter(titanic, is.na(Survived)==FALSE), aes(Pclass, Age)) +
  geom_violin(aes(fill=Survived), alpha=0.9) +
  facet_wrap(~Survived) +
  scale_fill_brewer(palette = "Dark2", direction = -1) +
  ggtitle("Survival Rate based on Pclass and Age") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```



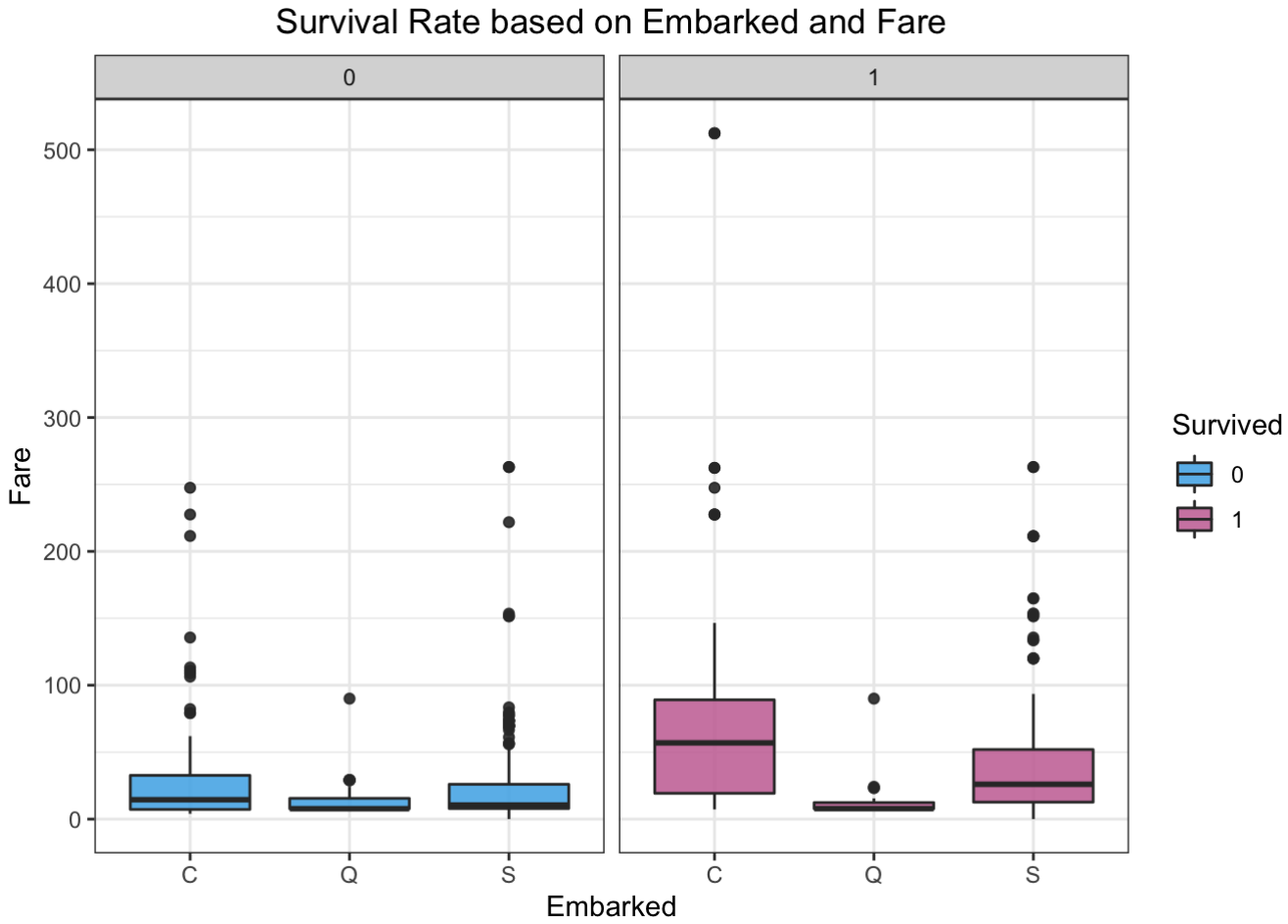

```
mosaicplot(~ Title + Survived, data=titanic, main='Survival Rate based on Title', shade=
TRUE)
```




```
mosaicplot(~ FamilySize + Survived, data=titanic, main='Survival Rate based on FamilySiz
e', shade=TRUE)
```




```
ggplot(filter(titanic, is.na(Survived)==FALSE), aes(Embarked, Fare)) +
  geom_boxplot(aes(fill=Survived), alpha=0.9) +
  facet_wrap(~Survived) +
  scale_fill_manual(values=c("#56B4E9", "#CC79A7")) +
  ggtitle("Survival Rate based on Embarked and Fare") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```




```
train_original <- titanic[1:891, c("Survived","Pclass","Sex","Age","SibSp","Parch","Far
e","Embarked","Title","FamilySize")]
test_original <- titanic[892:1309, c("Pclass","Sex","Age","SibSp","Parch","Fare","Embark
ed","Title","FamilySize")]
```

```
set.seed(789)
split = sample.split(train_original$Survived, SplitRatio = 0.8)
train = subset(train_original, split == TRUE)
test = subset(train_original, split == FALSE)
```

Logistic regression

```
cor(train[,unlist(lapply(train,is.numeric))])
```

```
##              Age      SibSp      Parch      Fare
## Age    1.0000000 -0.2758417 -0.2079948 0.1107712
## SibSp -0.2758417  1.0000000  0.4529568 0.1571153
## Parch -0.2079948  0.4529568  1.0000000 0.2361560
## Fare   0.1107712  0.1571153  0.2361560 1.0000000
```

```
ps = chisq.test(train$Pclass, train$Sex)$p.value
pe = chisq.test(train$Pclass, train$Embarked)$p.value
pt = chisq.test(train$Pclass, train$Title)$p.value
pf = chisq.test(train$Pclass, train$FamilySize)$p.value
se = chisq.test(train$Sex, train$Embarked)$p.value
st = chisq.test(train$Sex, train$Title)$p.value
sf = chisq.test(train$Sex, train$FamilySize)$p.value
et = chisq.test(train$Embarked, train$Title)$p.value
ef = chisq.test(train$Embarked, train$FamilySize)$p.value
tf = chisq.test(train$Title, train$FamilySize)$p.value
cormatrix = matrix(c(0, ps, pe, pt, pf,
                     ps, 0, se, st, sf,
                     pe, se, 0, et, ef,
                     pt, st, et, 0, tf,
                     pf, sf, ef, tf, 0),
                   5, 5, byrow = TRUE)
row.names(cormatrix) = colnames(cormatrix) = c("Pclass", "Sex", "Embarked", "Title", "Fa
milySize")
cormatrix
```

```
##                  Pclass           Sex     Embarked         Title   FamilySize
## Pclass     0.000000e+00  2.532566e-03 1.053100e-23  5.962301e-10 1.108964e-10
## Sex        2.532566e-03  0.000000e+00 1.321593e-02 1.116723e-150 4.591649e-15
## Embarked   1.053100e-23  1.321593e-02 0.000000e+00  1.383169e-04 2.490631e-06
## Title      5.962301e-10 1.116723e-150 1.383169e-04  0.000000e+00 2.204782e-51
## FamilySize 1.108964e-10  4.591649e-15 2.490631e-06  2.204782e-51 0.000000e+00
```

```
classifier = glm(Survived ~ ., family = binomial(link='logit'), data = train)
classifier <- step(classifier)
```

```
## Start:  AIC=612.29
## Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked +
##     Title + FamilySize
##
##              Df Deviance    AIC
## - SibSp       1   580.29 610.29
## - Embarked    2   582.69 610.69
## - Fare        1   580.81 610.81
## - Parch       1   581.59 611.59
## <none>            580.29 612.29
## - Sex         1   584.37 614.37
## - Age         1   585.69 615.69
## - FamilySize  2   590.68 618.68
## - Title       4   616.14 640.14
## - Pclass      2   624.73 652.73
##
## Step:  AIC=610.29
## Survived ~ Pclass + Sex + Age + Parch + Fare + Embarked + Title +
##     FamilySize
##
##              Df Deviance    AIC
## - Embarked    2   582.71 608.71
## - Fare        1   580.82 608.82
## - Parch       1   582.05 610.05
## <none>            580.29 610.29
## - Sex         1   584.37 612.37
## - Age         1   585.70 613.70
## - FamilySize  2   609.33 635.33
## - Title       4   616.76 638.76
## - Pclass      2   624.87 650.87
##
## Step:  AIC=608.71
## Survived ~ Pclass + Sex + Age + Parch + Fare + Title + FamilySize
##
##              Df Deviance    AIC
## - Fare        1   583.56 607.56
## - Parch       1   584.41 608.41
## <none>            582.71 608.71
## - Sex         1   586.56 610.56
## - Age         1   588.11 612.11
## - FamilySize  2   615.00 637.00
## - Title       4   619.32 637.32
## - Pclass      2   628.03 650.03
##
## Step:  AIC=607.56
## Survived ~ Pclass + Sex + Age + Parch + Title + FamilySize
##
##              Df Deviance    AIC
## - Parch       1   585.45 607.45
## <none>            583.56 607.56
## - Sex         1   587.31 609.31
## - Age         1   589.24 611.24
## - FamilySize  2   615.02 635.02
```

```
## - Title       4   619.43 635.43
## - Pclass      2   662.23 682.23
##
## Step:  AIC=607.45
## Survived ~ Pclass + Sex + Age + Title + FamilySize
##
##              Df Deviance    AIC
## <none>            585.45 607.45
## - Sex         1   589.15 609.15
## - Age         1   591.29 611.29
## - Title       4   623.47 637.47
## - FamilySize  2   622.80 640.80
## - Pclass      2   664.64 682.64
```

```
summary(classifier)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age + Title + FamilySize,
##     family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6190  -0.5712  -0.3802   0.5462   2.4571
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)      18.72233  506.79974   0.037 0.970531
## Pclass2          -1.42636    0.32191  -4.431 9.38e-06 ***
## Pclass3          -2.55761    0.31174  -8.204 2.32e-16 ***
## Sexmale         -14.63628  506.79929  -0.029 0.976960
## Age              -0.02518    0.01062  -2.370 0.017789 *
## TitleMiss       -15.04068  506.79960  -0.030 0.976324
## TitleMr          -3.36409    0.60123  -5.595 2.20e-08 ***
## TitleMrs        -14.59001  506.79969  -0.029 0.977033
## TitleOther       -2.96720    0.85828  -3.457 0.000546 ***
## FamilySizeSmall  -0.23457    0.25791  -0.909 0.363087
## FamilySizeLarge  -2.65324    0.50371  -5.267 1.38e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 949.90  on 712  degrees of freedom
## Residual deviance: 585.45  on 702  degrees of freedom
## AIC: 607.45
##
## Number of Fisher Scoring iterations: 13
```

```
vif(classifier)
```

```
##                    GVIF Df GVIF^(1/(2*Df))
## Pclass     1.667655e+00  2        1.136388
## Sex        5.751701e+06  1     2398.270329
## Age        1.894599e+00  1        1.376444
## Title      1.224494e+07  4        7.691208
## FamilySize 1.812345e+00  2        1.160273
```

```
classifier = glm(Survived ~ . -Sex, family = binomial(link='logit'), data = train)
```

```
classifier <- step(classifier)
```

```
## Start:  AIC=614.37
## Survived ~ (Pclass + Sex + Age + SibSp + Parch + Fare + Embarked +
##     Title + FamilySize) - Sex
##
##              Df Deviance    AIC
## - SibSp       1   584.37 612.37
## - Embarked    2   586.52 612.52
## - Fare        1   584.83 612.83
## - Parch       1   585.58 613.58
## <none>            584.37 614.37
## - Age         1   589.95 617.95
## - FamilySize  2   594.60 620.60
## - Pclass      2   629.59 655.59
## - Title       4   772.00 794.00
##
## Step:  AIC=612.37
## Survived ~ Pclass + Age + Parch + Fare + Embarked + Title + FamilySize
##
##              Df Deviance    AIC
## - Embarked    2   586.56 610.56
## - Fare        1   584.84 610.84
## - Parch       1   586.10 612.10
## <none>            584.37 612.37
## - Age         1   589.95 615.95
## - FamilySize  2   613.53 637.53
## - Pclass      2   629.78 653.78
## - Title       4   772.02 792.02
##
## Step:  AIC=610.56
## Survived ~ Pclass + Age + Parch + Fare + Title + FamilySize
##
##              Df Deviance    AIC
## - Fare        1   587.31 609.31
## - Parch       1   588.22 610.22
## <none>            586.56 610.56
## - Age         1   592.09 614.09
## - FamilySize  2   618.78 638.78
## - Pclass      2   632.93 652.93
## - Title       4   783.98 799.98
##
## Step:  AIC=609.31
## Survived ~ Pclass + Age + Parch + Title + FamilySize
##
##              Df Deviance    AIC
## - Parch       1   589.15 609.15
## <none>            587.31 609.31
## - Age         1   593.14 613.14
## - FamilySize  2   618.78 636.78
## - Pclass      2   667.53 685.53
## - Title       4   785.52 799.52
##
## Step:  AIC=609.15
## Survived ~ Pclass + Age + Title + FamilySize
```

```
##
##              Df Deviance    AIC
## <none>            589.15 609.15
## - Age         1   595.17 613.17
## - FamilySize  2   626.64 642.64
## - Pclass      2   669.92 685.92
## - Title       4   793.80 805.80
```

```
summary(classifier)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Age + Title + FamilySize, family = binomial(link =
"logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6304  -0.5750  -0.3792   0.5637   2.4607
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      4.11423    0.69575   5.913 3.35e-09 ***
## Pclass2         -1.46793    0.32032  -4.583 4.59e-06 ***
## Pclass3         -2.58021    0.31125  -8.290  < 2e-16 ***
## Age             -0.02543    0.01059  -2.403  0.01627 *
## TitleMiss       -0.40382    0.55940  -0.722  0.47037
## TitleMr         -3.36730    0.60132  -5.600 2.15e-08 ***
## TitleMrs         0.05208    0.63190   0.082  0.93431
## TitleOther      -2.56210    0.81122  -3.158  0.00159 **
## FamilySizeSmall -0.23202    0.25621  -0.906  0.36516
## FamilySizeLarge -2.65769    0.50374  -5.276 1.32e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 949.90  on 712  degrees of freedom
## Residual deviance: 589.15  on 703  degrees of freedom
## AIC: 609.15
##
## Number of Fisher Scoring iterations: 5
```

```
vif(classifier)
```

```
##                GVIF Df GVIF^(1/(2*Df))
## Pclass     1.688094  2        1.139854
## Age        1.909003  1        1.381667
## Title      2.618396  4        1.127858
## FamilySize 1.805038  2        1.159102
```

```
durbinWatsonTest(classifier)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      0.02148659      1.956557    0.55
##  Alternative hypothesis: rho != 0
```

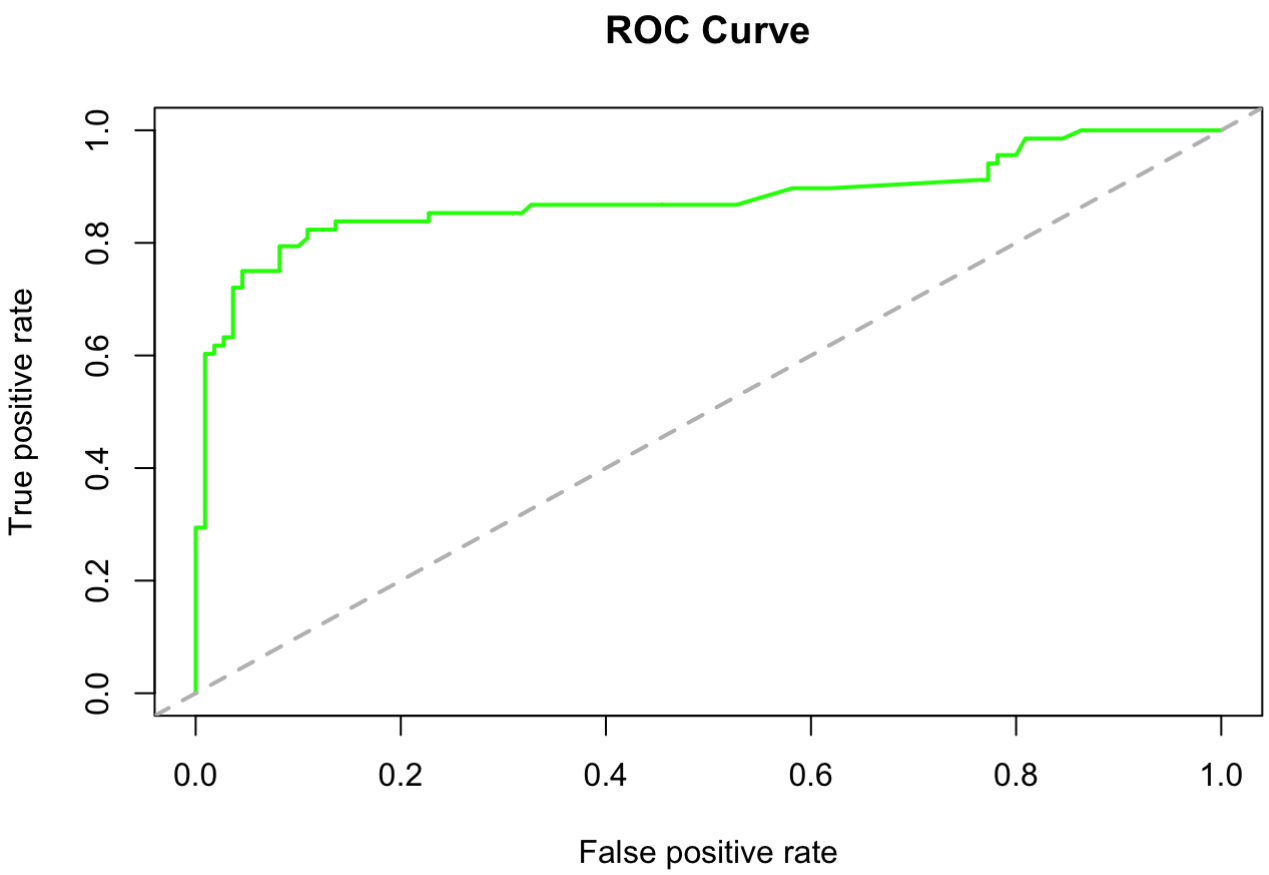
```
prob_pred = predict(classifier, type = 'response', newdata = test)
y_pred = ifelse(prob_pred > 0.5, 1, 0)
table(test$Survived, y_pred > 0.5)
```

```
##
##     FALSE TRUE
##   0   101    9
##   1    17   51
```

```
error <- mean(test$Survived != y_pred)
paste('Accuracy',round(1-error,4))
```

```
## [1] "Accuracy 0.8539"
```

```
fitpred = prediction(prob_pred, test$Survived)
fitperf = performance(fitpred,"tpr","fpr")
plot(fitperf,col="green",lwd=2,main="ROC Curve")
abline(a=0,b=1,lwd=2,lty=2,col="gray")
```



Support Vector Machines

```
paste('Age variance: ',var(train$Age),', SibSp variance: ',var(train$SibSp),', Parch var
iance: ',var(train$Parch),', Fare variance: ',var(train$Fare))
```

```
## [1] "Age variance:  173.992892791181 , SibSp variance:  1.26512441495816 , Parch vari
ance:  0.604594449784894 , Fare variance:  2291.51426667939"
```

```
standardized.train = cbind(select(train, Survived, Pclass, Sex, SibSp, Parch, Embarked,
Title, FamilySize), Age = scale(train$Age), Fare = scale(train$Fare))
paste('Age variance: ',var(standardized.train$Age),', Fare variance: ',var(standardized.
train$Fare))
```

```
## [1] "Age variance:  1 , Fare variance:  1"
```

```
standardized.test = cbind(select(test, Survived, Pclass, Sex, SibSp, Parch, Embarked, Ti
tle, FamilySize), Age = scale(test$Age), Fare = scale(test$Fare))
paste('Age variance: ',var(standardized.test$Age),', Fare variance: ',var(standardized.t
est$Fare))
```

```
## [1] "Age variance:  1 , Fare variance:  1"
```

```
classifier = svm(Survived ~ .,
                 data = standardized.train,
                 type = 'C-classification',
                 kernel = 'linear')

y_pred = predict(classifier, newdata = standardized.test[,-which(names(standardized.tes
t)=="Survived")])

table(test$Survived, y_pred)
```

```
##    y_pred
##       0   1
##   0 106   4
##   1  18  50
```

```
error <- mean(test$Survived != y_pred)
paste('Accuracy',round(1-error,4))
```

```
## [1] "Accuracy 0.8764"
```

```
classifier = svm(Survived ~ .,
                 data = standardized.train,
                 type = 'C-classification',
                 kernel = 'radial')

y_pred = predict(classifier, newdata = standardized.test[,-which(names(standardized.tes
t)=="Survived")])

table(test$Survived, y_pred)
```

```
##    y_pred
##       0   1
##   0 105   5
##   1  16  52
```

```
error <- mean(test$Survived != y_pred)
paste('Accuracy',round(1-error,4))
```

```
## [1] "Accuracy 0.882"
```

```
tune.results <- tune(svm,
                     Survived ~ .,
                     data = standardized.train,
                     kernel='radial',
                     ranges=list(cost=2^(-2:2), gamma=2^(-6:-2)))
summary(tune.results)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     4 0.125
##
## - best performance: 0.1794992
##
## - Detailed performance results:
##    cost    gamma     error dispersion
## 1  0.25 0.015625 0.2033646 0.02818232
## 2  0.50 0.015625 0.1936033 0.03253985
## 3  1.00 0.015625 0.1851721 0.03958896
## 4  2.00 0.015625 0.1851526 0.03885835
## 5  4.00 0.015625 0.1893388 0.03426396
## 6  0.25 0.031250 0.1865806 0.03383293
## 7  0.50 0.031250 0.1865415 0.03425425
## 8  1.00 0.031250 0.1865610 0.03670215
## 9  2.00 0.031250 0.1879695 0.03914563
## 10 4.00 0.031250 0.1865610 0.03729787
## 11 0.25 0.062500 0.1837441 0.03694279
## 12 0.50 0.062500 0.1851526 0.03718315
## 13 1.00 0.062500 0.1865806 0.03627857
## 14 2.00 0.062500 0.1838224 0.03791697
## 15 4.00 0.062500 0.1894171 0.04495805
## 16 0.25 0.125000 0.1865415 0.03492662
## 17 0.50 0.125000 0.1837637 0.03710588
## 18 1.00 0.125000 0.1865610 0.04113898
## 19 2.00 0.125000 0.1795579 0.04752411
## 20 4.00 0.125000 0.1794992 0.04997202
## 21 0.25 0.250000 0.1949531 0.03390228
## 22 0.50 0.250000 0.1823161 0.04411747
## 23 1.00 0.250000 0.1795579 0.05355751
## 24 2.00 0.250000 0.1837050 0.06249995
## 25 4.00 0.250000 0.1976330 0.06657420
```

```
classifier = svm(Survived ~ .,
                 data = standardized.train,
                 type = 'C-classification',
                 kernel = 'radial',
                 cost = 4,
                 gamma = 0.125)

y_pred = predict(classifier, newdata = standardized.test[,-which(names(standardized.tes
t)=="Survived")])

table(test$Survived, y_pred)
```

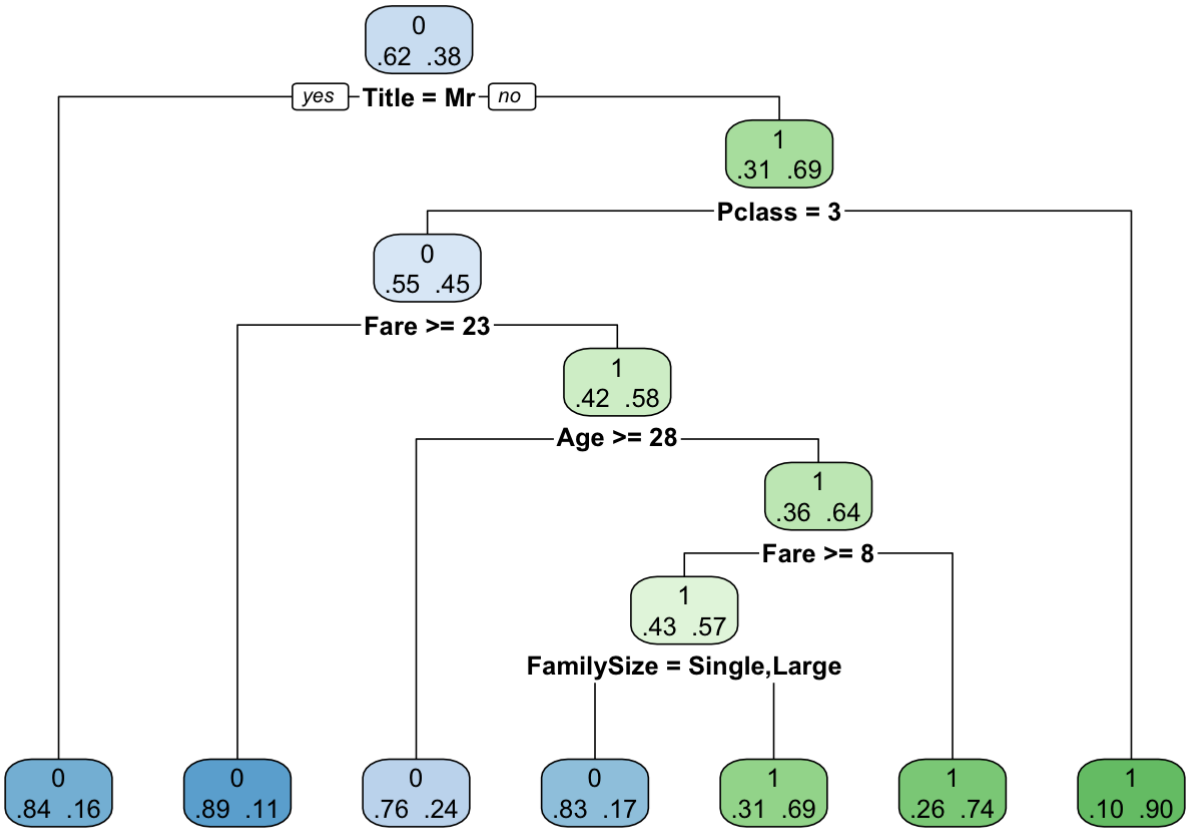
```
##     y_pred
##        0   1
##    0 104   6
##    1  24  44
```

```
error <- mean(test$Survived != y_pred)
paste('Accuracy',round(1-error,4))
```

```
## [1] "Accuracy 0.8315"
```

Decision Tree

```
classifier = rpart(Survived ~ ., data = train, method = 'class')
rpart.plot(classifier, extra=4)
```



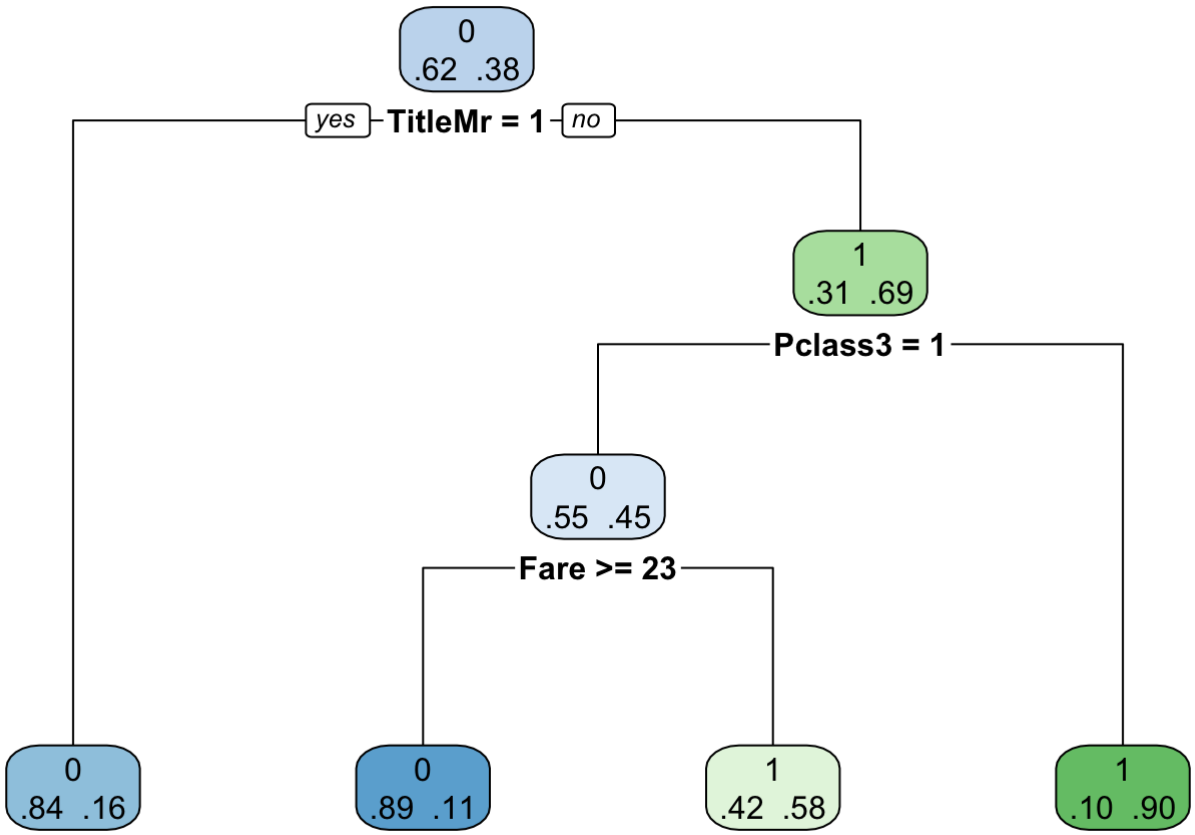

```
y_pred = predict(classifier, newdata = test[,-which(names(test)=="Survived")], type='cla
ss')
table(test$Survived, y_pred)
```

```
##     y_pred
##        0   1
##    0 102   8
##    1  21  47
```

```
error <- mean(test$Survived != y_pred)
paste('Accuracy',round(1-error,4))
```

```
## [1] "Accuracy 0.8371"
```

```
set.seed(789)
folds = createMultiFolds(train$Survived, k = 10, times = 5)
control <- trainControl(method = "repeatedcv", index = folds)
classifier_cv <- train(Survived ~ ., data = train, method = "rpart", trControl = contro
l)
rpart.plot(classifier_cv$finalModel, extra=4)
```




```
y_pred = predict(classifier_cv, newdata = test[,-which(names(test)=="Survived")])
table(test$Survived, y_pred)
```

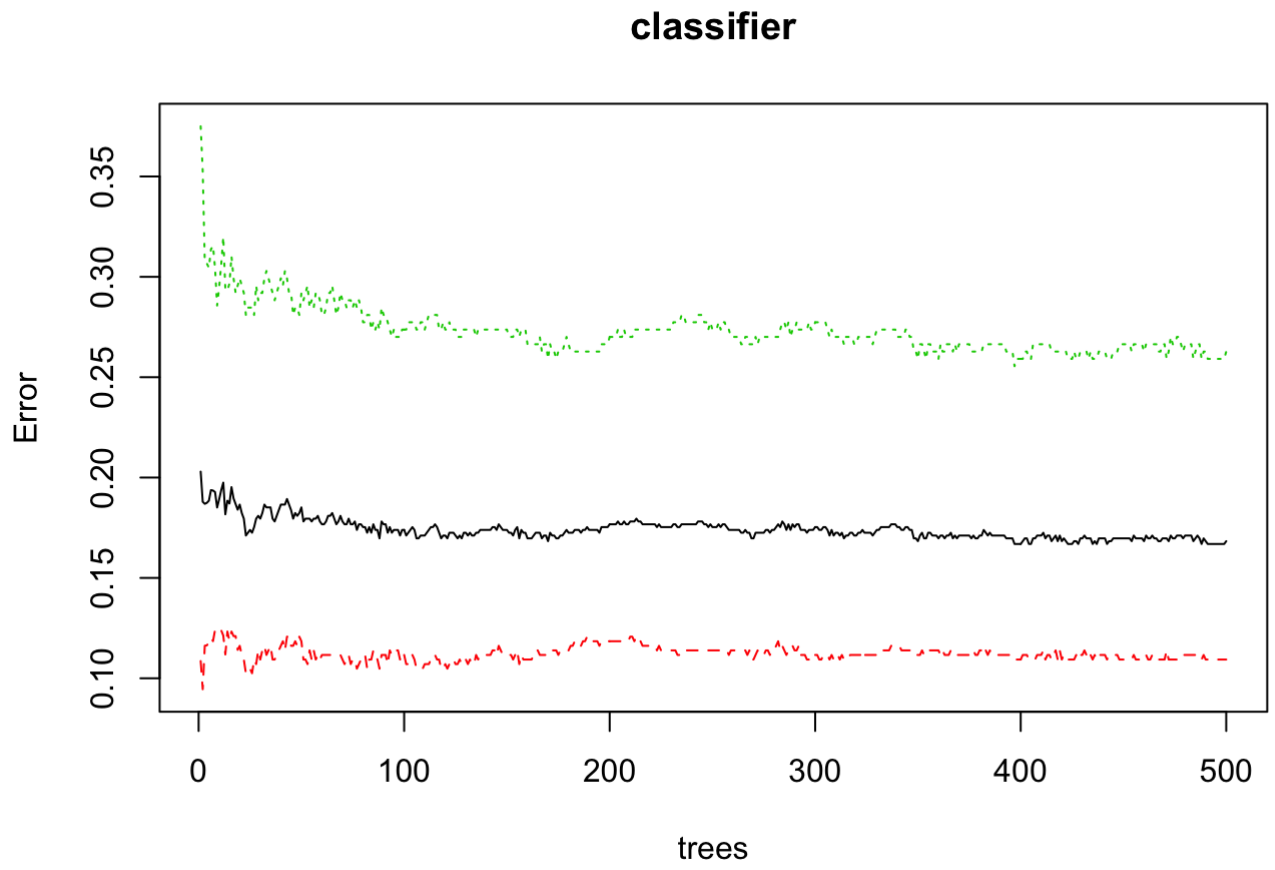
```
##     y_pred
##       0  1
##    0 99 11
##    1 17 51
```

```
error <- mean(test$Survived != y_pred)
paste('Accuracy',round(1-error,4))
```

```
## [1] "Accuracy 0.8427"
```

Random Forests

```
set.seed(432)
classifier = randomForest(Survived ~ ., data = train)
plot(classifier)
```




```
y_pred = predict(classifier, newdata = test[,-which(names(test)=="Survived")])
table(test$Survived, y_pred)
```

```
##    y_pred
##       0  1
##   0 99 11
##   1 18 50
```

```
error <- mean(test$Survived != y_pred)
paste('Accuracy',round(1-error,4))
```

```
## [1] "Accuracy 0.8371"
```

```
set.seed(651)
folds = createMultiFolds(train$Survived, k = 10)
control <- trainControl(method = "repeatedcv", index = folds)
classifier_cv <- train(Survived ~ ., data = train, method = "rf", trControl = control)
y_pred = predict(classifier_cv, newdata = test[,-which(names(test)=="Survived")])
table(test$Survived, y_pred)
```

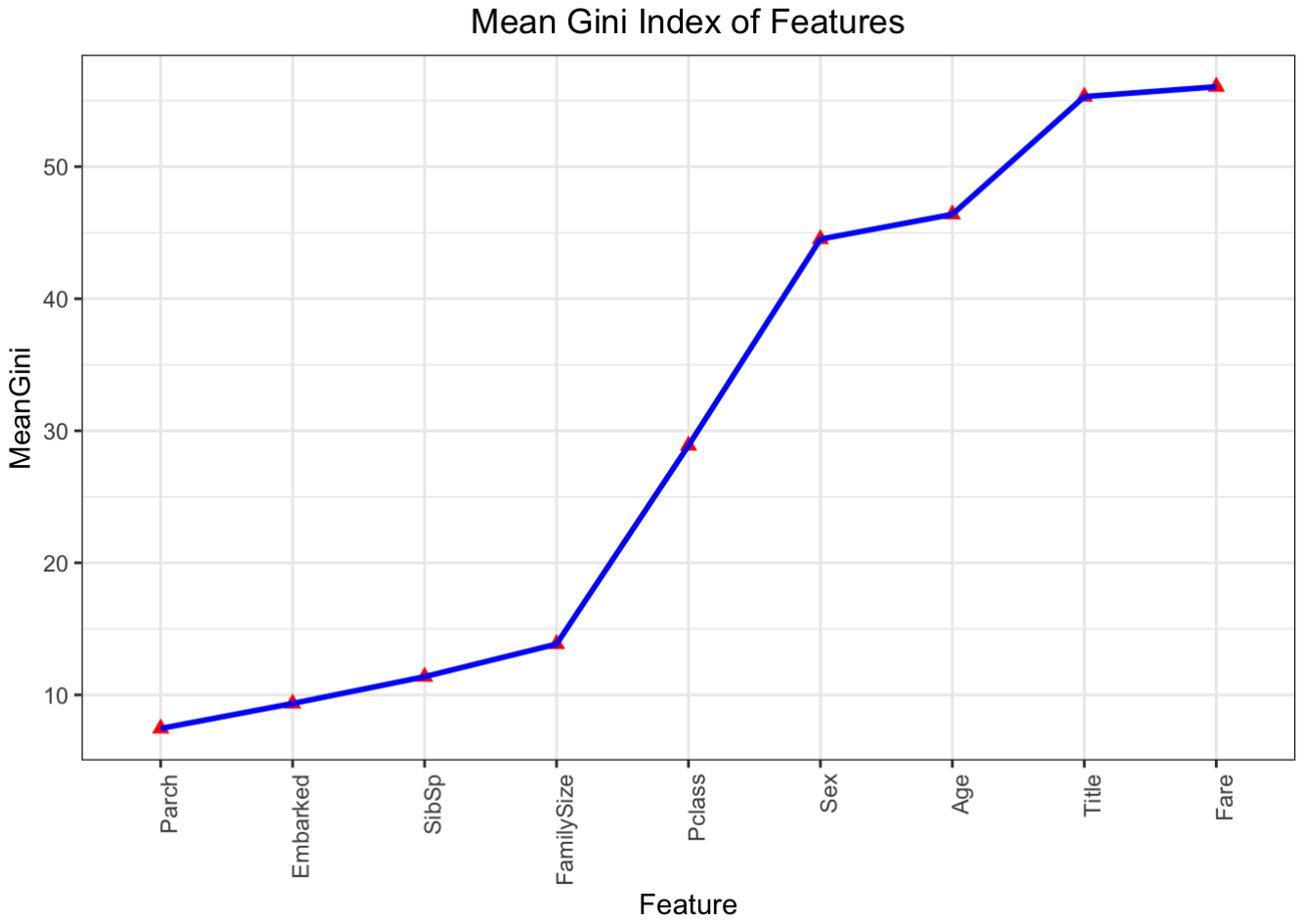
```
##    y_pred
##       0  1
##   0 94 16
##   1 19 49
```

```
error <- mean(test$Survived != y_pred)
paste('Accuracy',round(1-error,4))
```

```
## [1] "Accuracy 0.8034"
```

```
gini = as.data.frame(importance(classifier))
gini = data.frame(Feature = row.names(gini),
                  MeanGini = round(gini[ ,'MeanDecreaseGini'], 2))
gini = gini[order(-gini[,"MeanGini"]),]

ggplot(gini,aes(reorder(Feature,MeanGini), MeanGini, group=1)) +
  geom_point(color='red',shape=17,size=2) +
  geom_line(color='blue',size=1) +
  scale_y_continuous(breaks=seq(0,60,10)) +
  xlab("Feature") +
  ggtitle("Mean Gini Index of Features") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Naive Bayes

```
classifier = naiveBayes(Survived ~ ., data = train)
y_pred = predict(classifier, newdata = test[,-which(names(test)=="Survived")])
table(test$Survived, y_pred)
```

```
##    y_pred
##       0  1
##   0 99 11
##   1 17 51
```

```
error <- mean(test$Survived != y_pred)
paste('Accuracy',round(1-error,4))
```

```
## [1] "Accuracy 0.8427"
```

Results

```
y_pred = predict(classifier, newdata = test_original)
results <- data.frame(PassengerID = titanic[892:1309,"PassengerId"], Survived = y_pred)
write.csv(results, file = 'PredictingTitanicSurvival.csv', row.names = FALSE, quote=FALS
E)
```