# INTRODUCTION

In this challenge, the aim is to find out that what sorts of people were likely to survive. We apply data mining and machine learning methods to predict which passengers survived. This is my first stab at a Kaggle script. I have chosen to work with the Titanic dataset after spending some time poking around on the site and looking at other scripts made by other Kagglers for inspiration. I will also focus on doing some illustrative data visualizations along the way. I’ll then use randomForest to create a model predicting survival on the Titanic. I am new to machine learning and hoping to learn a lot, so feedback is very welcome!

There are three parts to my script as follows:

* Feature engineering
* Missing value imputation
* Prediction!

# DATASET

I use Kaggle Titanic dataset for my project. Kaggle describes this dataset as; the sinking of the Titanic is one of the most infamous shipwrecks in history. The Titanic sank after colliding with an iceberg on April 15, 1912, killing 1502 out of 2224 passengers and crew. This tragedy shocked the international community and led to better safety regulations for ships. One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others.

# EXPLORATORY DATA ANALYSİS

## Load Libraries

Load libraries into R;

library(Hmisc)

library(knitr)

library(ggplot2)

library(dplyr)

library(caret)

library(randomForest)

library(gridExtra)

library(ROCR)

library(corrplot)

## Read CSV Files

Reading csv files as dataframes;

train <- read.csv("../input/train.csv", stringsAsFactors = F, na.strings = c("NA", ""))

test <- read.csv("../input/test.csv", stringsAsFactors = F, na.strings = c("NA", ""))

## Data size and structure

Describe the dataset;

str(train) # describe the dataset

The description of the training dataset is shown in the Figure 1.

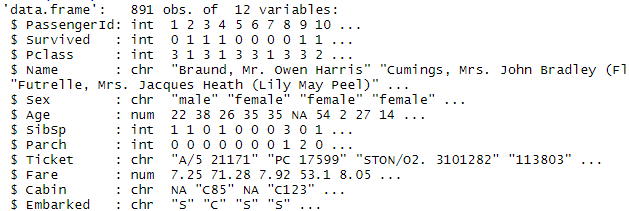


Figure 1. Output of Train

The training set consists of 891. The test set consist of 418 observations. Note that “Survived” variable is missing from test dataset compared to the training dataset. Creating this variable is the purpose of this project.

## Description of Variables

Description of all the variables is shown in the Table 1.

Table 1. Description of the Dataset

| Variable Name | Description |
| --- | --- |
| Survived | Survived (1) or died (0) |
| Pclass | Passenger’s class |
| Name | Passenger’s name |
| Sex | Passenger’s sex |
| Age | Passenger’s age |
| SibSp | Number of siblings/spouses aboard |
| Parch | Number of parents/children aboard |
| Ticket | Ticket number |
| Fare | Fare |
| Cabin | Cabin |
| Embarked | Port of embarkation |

## Merge Train and Test Datasets

Here we merge the train and the test datasets. This is required for data cleaning and feature engineering.

test$Survived <- NA # add “Survived” column to test dataset.

all <- rbind(train, test) # merge train and test datasets.

Combined training and test datasets consists of 1309 variables in total.

## Completeness of the data

We would like to see which variables contain missing values.

sapply(all, function(x) {sum(is.na(x))}) # check which columns contains missing values.

The output of this command is shown in the Figure 2.



Figure 2. The Output of sapply(all, function(x) {sum(is.na(x))})

The 418 observations is missing the Survived column in the test dataset. Cabin is sparsely populated. Age is also missing in some observations. In addition, Embarked is missing two values and one Fare value is missing.

## Exploring Some of the Variables

Sex, Pclass and Survived variables are all complete. Since they are complete and tidy, we can convert them into factors.

all$Sex <- as.factor(all$Sex)

all$Survived <- as.factor(all$Survived)

all$Pclass <- as.ordered(all$Pclass) #because Pclass is ordinal

### Survived

Survived variable describes the information of “how many people survived, and how many died”.

ggplot(all[!is.na(all$Survived),], aes(x = Survived, fill = Survived)) +

geom\_bar(stat='count') +

labs(x = 'How many people died and survived on the Titanic?') +

geom\_label(stat='count',aes(label=..count..), size=7) +

theme\_grey(base\_size = 18)

The number of people survived and died in training dataset is shown in the Figure 3.

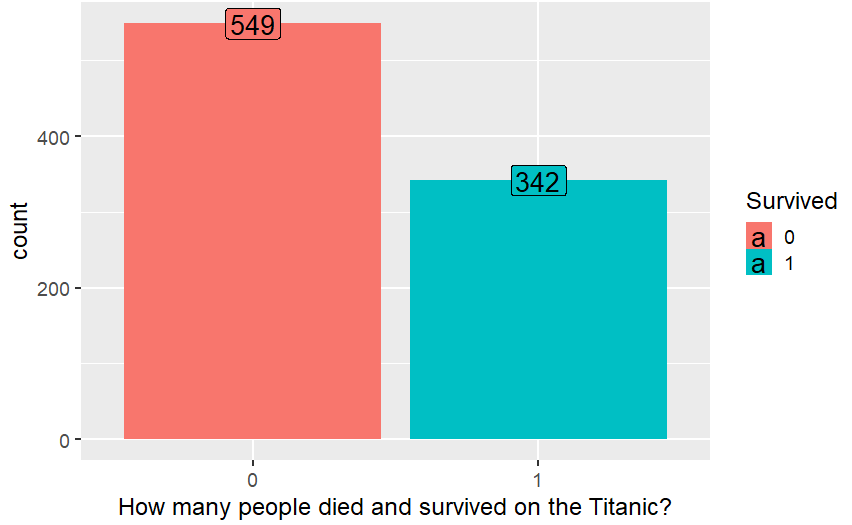


Figure 3. How many people died and survived on the Titanic?

### Sex/gender

Of the 1309 people on the Titanic, 64.4% was male. Within the training data 81.1% of the men died, and 25.8% of the women died.

p1 <- ggplot(all, aes(x = Sex, fill = Sex)) +

geom\_bar(stat='count', position='dodge') + theme\_grey() +

labs(x = 'All data') +

geom\_label(stat='count', aes(label=..count..)) +

scale\_fill\_manual("legend", values = c("female" = "pink", "male" = "green"))

p2 <- ggplot(all[!is.na(all$Survived),], aes(x = Sex, fill = Survived)) +

geom\_bar(stat='count', position='dodge') + theme\_grey() +

labs(x = 'Training data only') +

geom\_label(stat='count', aes(label=..count..))

grid.arrange(p1,p2, nrow=1)

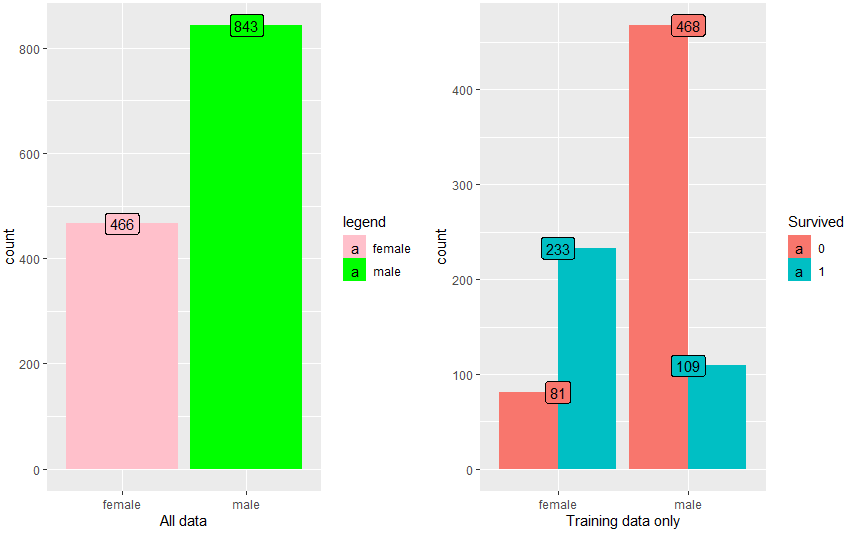


Figure 4 This percentage of Male/Female with Survived Information

### Passenger Class

Most people in Titanic traveled in 3rd class. Also, as expected, survival is strongly correlated with the passenger class. A majority of first class passengers survived, and majority of 3rd class died. It is noticeable that almost all women in 1st and 2nd class survived. For men, 2nd class was is as bad as 3rd class.

p3 <- ggplot(all, aes(x = Pclass, fill = Pclass)) +

geom\_bar(stat='count', position='dodge') +

labs(x = 'Pclass, All data') + geom\_label(stat='count', aes(label=..count..)) +

theme(legend.position="none") + theme\_grey()

p4 <- ggplot(all[!is.na(all$Survived),], aes(x = Pclass, fill = Survived)) +

geom\_bar(stat='count', position='dodge') + labs(x = 'Training data only') +

theme(legend.position="none") + theme\_grey()

p5 <- ggplot(all[!is.na(all$Survived),], aes(x = Pclass, fill = Survived)) +

geom\_bar(stat='count', position='stack') +

labs(x = 'Training data only', y= "Count") + facet\_grid(.~Sex) +

theme(legend.position="none") + theme\_grey()

p6 <- ggplot(all[!is.na(all$Survived),], aes(x = Pclass, fill = Survived)) +

geom\_bar(stat='count', position='fill') +

labs(x = 'Training data only', y= "Percent") + facet\_grid(.~Sex) +

theme(legend.position="none") + theme\_grey()

grid.arrange(p3, p4, p5, p6, ncol=2)

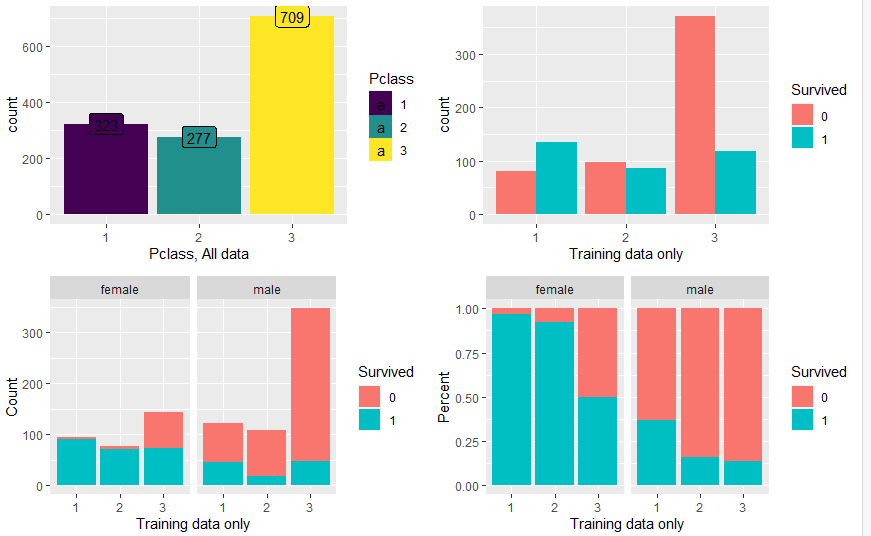


Figure 5. Passenger Class Information

Pclass 1 and 2 are almost guaranteed survival for women, and Pclass 2 is almost as bad as Pclass 3 for men. It is better to work on Class and Sex variables together than working on them as separate predictors.

all$PclassSex[all$Pclass=='1' & all$Sex=='male'] <- 'P1Male'

all$PclassSex[all$Pclass=='2' & all$Sex=='male'] <- 'P2Male'

all$PclassSex[all$Pclass=='3' & all$Sex=='male'] <- 'P3Male'

all$PclassSex[all$Pclass=='1' & all$Sex=='female'] <- 'P1Female'

all$PclassSex[all$Pclass=='2' & all$Sex=='female'] <- 'P2Female'

all$PclassSex[all$Pclass=='3' & all$Sex=='female'] <- 'P3Female'

all$PclassSex <- as.factor(all$PclassSex)

# FEATURE ENGINEERING

## Creating the Title variable

The name variable has no missing values, but it actually contains more than just a first name and surname. It also contains a Title for each person. The effects of family names can match with siblings-spouses and parents-children. Therefore, we need to extract Title and Surname from the Name variable.

all$Surname <- sapply(all$Name, function(x) {strsplit(x, split='[,.]')[[1]][1]})

#correcting some surnames that also include a maiden name

all$Surname <- sapply(all$Surname, function(x) {strsplit(x, split='[-]')[[1]][1]})

all$Title <- sapply(all$Name, function(x) {strsplit(x, split='[,.]')[[1]][2]})

all$Title <- sub(' ', '', all$Title) #removing spaces before title

kable(table(all$Sex, all$Title))

We can reduce the number of titles to create better Titles that can be used for prediction. Ms. is usually used for younger married women. I will therefore join this one with Miss. There has to be some assumptions, such as; Mlle stands for Mademoiselle and Mme stands for Madame. For the titles with low frequecies, we can create one new category.

all$Title[all$Title %in% c("Mlle", "Ms")] <- "Miss"

all$Title[all$Title== "Mme"] <- "Mrs"

all$Title[!(all$Title %in% c('Master', 'Miss', 'Mr', 'Mrs'))] <- "Rare Title"

all$Title <- as.factor(all$Title)

kable(table(all$Sex, all$Title))

ggplot(all[!is.na(all$Survived),], aes(x = Title, fill = Survived)) +

geom\_bar(stat='count', position='stack') +

labs(x = 'Title') +theme\_grey()

Title-person information can be seen in Figure 6.

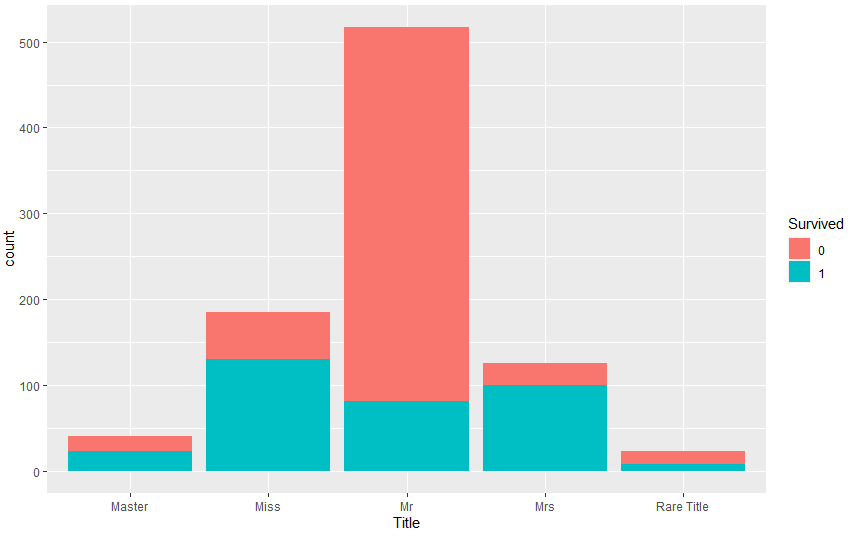


Figure 6. Titles Information

## Find Groups

### Families

In order to create the family size for each person on the boat, we need to add his/her number of parents/children, his/her siblings/spouses, and himself/herself.

all$Fsize <- all$SibSp+all$Parch +1 #creating family size variable (Fsize)

Below, it is clear that single travelers had a much higher chance to die. In addition, people traveling in families of 2-4 people actually had a relatively high chance to survive. It is significantly lower among more crowded families.

ggplot(all[!is.na(all$Survived),], aes(x = Fsize, fill = Survived)) +

geom\_bar(stat='count', position='dodge') +

scale\_x\_continuous(breaks=c(1:11)) +

labs(x = 'Family Size') + theme\_grey()

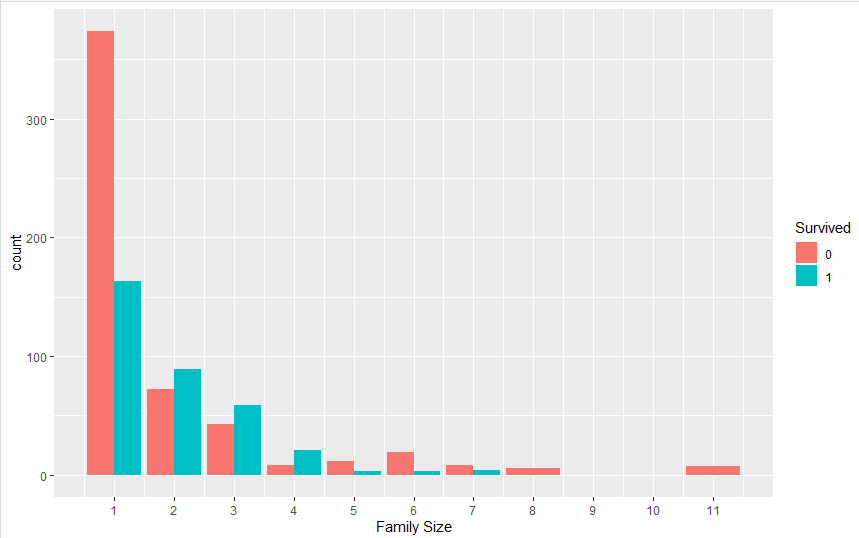


Figure 7. Family Size Information

Now, we can convert family sizes into categories such as; solo, small family, large family.

### Family Size Inconsistencies

To check the family size data for inconsistencies, we create a variable that combines the surname and the Fsize. After that, we can check where these combinations lead to strange numbers of families.

all$FsizeName <- paste(as.character(all$Fsize), all$Surname, sep="") # combine Fsize and Surname

SizeCheck <- all %>%

group\_by(FsizeName, Fsize) %>%

summarise(NumObs=n())

SizeCheck$NumFam <- SizeCheck$NumObs/SizeCheck$Fsize

SizeCheck$modulo <- SizeCheck$NumObs %% SizeCheck$Fsize

SizeCheck <- SizeCheck[SizeCheck$modulo !=0,]

sum(SizeCheck$NumObs) #total number of Observations with inconsistencies

kable(SizeCheck[SizeCheck$FsizeName %in% c('3Davies', '5Hocking', '6Richards', '2Wilkes', '3Richards', '4Hocking'),]) #only display some inconsistencies that are discussed in the text

This check does not always add up to a round number of families (for a total of 93 passengers). Some inconsistencies can likely be explained by cancellations. For instance: there must have been two Davies families on board, while the SizeCheck only shows five obervations for FsizeName ‘3Davies’.

kable(all[all$FsizeName=='3Davies',c(2,3,14,5,6,7,8,17,9,15)])

The error seems to be that Mrs Davies [1222] was supposed to travel with 2 children, but eventually only traveled with one son (Master Davies [550]). This info can be corrected.

all$FsizeName[c(550, 1222)] <- '2Davies'

all$SibSp[550] <- 0

all$Parch[1222] <- 1

all$Fsize[c(550, 1222)] <- 2

kable(all[all$FsizeName=='2Davies',c(2,3,14,5,6,7,8,17,9,15)])

There may be more cancellations that were not administered fully correctly.

### Families; what about uncles, aunts, cousins, nieces, grandparents, brothers/sisters-in law?

I found out that there is something ‘hidden’ in this information that seems more interesting than a few cancellations. For instance, it turns out that the Hockings and the Richards’ are related. The connection here is that passenger 438 travels with 2 children, 1 parent, a brother and a sister. For her, all these people count are direct family. However, other people are linked indirectly. For the 2 children for instance, only the brother and mother count as direct family. This leads to Fsizes that cannot be compared to most families. Their Fsizes are generally too high, as it is likely that those people have split up into smaller groups. The mother may have stayed with her children, while the brother and sister probably have stayed with the grandmother.

Note: this family is actually even more complex, as the grandmother also travels with a sister with the same maiden name. However, as this really seems an exception and the other Mrs Needs already has the very reasonable Fsize of 2 (it’s Mrs Wilkes Needs), I am not taking her into consideration.

kable(all[all$Ticket %in% c('29104', '29105', '29106'),c(2,3,4,5,6,7,8,9,15)])

In order to fix this, we have to ‘glue’ those families together using maiden names.

NC <- all[all$FsizeName %in% SizeCheck$FsizeName,] #create data frame with only relevant Fsizenames

#extracting maiden names

NC$Name <- sub("\\s$", "", NC$Name) #removing spaces at end Name

NC$Maiden <- sub(".\*[^\\)]$", "", NC$Name) #remove when not ending with ')'

NC$Maiden <- sub(".\*\\s(.\*)\\)$", "\\1", NC$Maiden)

NC$Maiden[NC$Title!='Mrs'] <- "" #cleaning up other stuff between brackets (including Nickname of a Mr)

NC$Maiden <- sub("^\\(", '', NC$Maiden) #removing opening brackets (sometimes single name, no spaces between brackets)

#making an exceptions match

NC$Maiden[NC$Name=='Andersen-Jensen, Miss. Carla Christine Nielsine'] <- 'Jensen'

#take only Maiden names that also exist as surname in other Observations

NC$Maiden2[NC$Maiden %in% NC$Surname] <- NC$Maiden[NC$Maiden %in% NC$Surname]

#create surname+maiden name combinations

NC$Combi[!is.na(NC$Maiden2)] <- paste(NC$Surname[!is.na(NC$Maiden2)], NC$Maiden[!is.na(NC$Maiden2)])

#create labels dataframe with surname and maiden merged into one column

labels1 <- NC[!is.na(NC$Combi), c('Surname','Combi')]

labels2 <- NC[!is.na(NC$Combi), c('Maiden','Combi')]

colnames(labels2) <- c('Surname', 'Combi')

labels1 <- rbind(labels1, labels2)

NC$Combi <- NULL

NC <- left\_join(NC, labels1, by='Surname')

#Find the maximum Fsize within each newly found 'second degree' family

CombiMaxF <- NC[!is.na(NC$Combi),] %>%

group\_by(Combi) %>%

summarise(MaxF=max(Fsize)) #summarise(MaxF=n())

NC <- left\_join(NC, CombiMaxF, by = "Combi")

#create family names for those larger families

NC$FsizeCombi[!is.na(NC$Combi)] <- paste(as.character(NC$Fsize[!is.na(NC$Combi)]), NC$Combi[!is.na(NC$Combi)], sep="")

#find the ones in which not all Fsizes are the same

FamMaid <- NC[!is.na(NC$FsizeCombi),] %>%

group\_by(FsizeCombi, MaxF, Fsize) %>%

summarise(NumObs=n())

FamMaidWrong <- FamMaid[FamMaid$MaxF!=FamMaid$NumObs,]

kable(unique(NC[!is.na(NC$Combi) & NC$FsizeCombi %in% FamMaidWrong$FsizeCombi, c('Combi', 'MaxF')]))

As you can see, 7 combinations (total of 28 passengers) are found as families with not all members having the same Fsize, which means that they are broader families with non-direct family links included. Before I decided what to do with these, I first have to find the families who are similarly linked on the ‘male’ side.

NC$MaxF <- NULL #erasing MaxF column maiden combi's

#Find the maximum Fsize within remaining families (no maiden combi's)

FamMale <- NC[is.na(NC$Combi),] %>%

group\_by(Surname) %>%

summarise(MaxF=max(Fsize))

NC <- left\_join(NC, FamMale, by = "Surname")

NCMale <- NC[is.na(NC$Combi),] %>%

group\_by(Surname, FsizeName, MaxF) %>%

summarise(count=n()) %>%

group\_by(Surname, MaxF) %>%

filter(n()>1) %>%

summarise(NumFsizes=n())

NC$Combi[NC$Surname %in% NCMale$Surname] <- NC$Surname[NC$Surname %in% NCMale$Surname]

kable(NCMale[, c(1,2)])

Example. Mr Julius Vander Planke is traveling with a spouse and 2 siblings. His spouse and siblings (brothers/sisters-in-law) are ‘indirectly’ related to each other.

kable(all[all$Surname=='Vander Planke', c(2,3,4,5,6,7,8,9,15)])

This means that altogether, there are 9 families (37 passengers) that include ‘second degree’ family members. What I want to do is give each member in such family the same Fsize (which gives everybody in these families the same survival chances with regards to the group variable). I have chosen to make this the average of the Fsize (which are based on siblings/spouse/parents/children only).

#selecting those 37 passengers In Not Correct dataframe

NC <- NC[(NC$FsizeCombi %in% FamMaidWrong$FsizeCombi)|(NC$Surname %in% NCMale$Surname),]

#calculating the average Fsize for those 9 families

NC1 <- NC %>%

group\_by(Combi) %>%

summarise(Favg=mean(Fsize))

kable(NC1)

result is that for instance the Fsize is 4 for all 6 people in the Richards-Hockings family. This exactly what I wanted, as I wanted to combine those people into a group with all members having the same Fsize (to give equal survival chances to all members within the group) but also not the maximum size as they are less likely to stay together than first degree families.

NC <- left\_join(NC, NC1, by = "Combi") #adding Favg to NC dataframe

NC$Favg <- round(NC$Favg) #rounding those averages to integers

NC <- NC[, c('PassengerId', 'Favg')]

all <- left\_join(all, NC, by='PassengerId')

#replacing Fsize by Favg

all$Fsize[!is.na(all$Favg)] <- all$Favg[!is.na(all$Favg)]

4.2.4 Can we still find more second degree families?

Am I still missing some second degree families? Yes, at it appears that some people traveling solo with the same surname have tickets with almost the same number!

all$Ticket2 <- sub("..$", "xx", all$Ticket) # creating a variable with almost the same ticket numbers (only last 2 digits varying)

As they have no sibling/spouses and no parents/children, these people are likely cousins/uncles. If you look deeper into the data, you will see that these groups of cousins/uncles sometimes also travel with (first degree) families. However, I think the key to this exercise is not to find the absolute largest groups that people may have stayed together with. I think it should be to detect smaller groups that actually stayed together. It sounds reasonable to assume that first degree families stayed together, and that uncles/cousins also took care of each other (this is consistent with the averaging of the Fsizes in the previous section). Altogether, I have found another 56 passengers that I can assign a group size to.

rest <- all %>%

select(PassengerId, Title, Age, Ticket, Ticket2, Surname, Fsize) %>%

filter(Fsize=='1') %>%

group\_by(Ticket2, Surname) %>%

summarise(count=n())

rest <- rest[rest$count>1,]

rest1 <- all[(all$Ticket2 %in% rest$Ticket2 & all$Surname %in% rest$Surname & all$Fsize=='1'), c('PassengerId', 'Surname', 'Title', 'Age', 'Ticket', 'Ticket2', 'Fsize', 'SibSp', 'Parch')]

rest1 <- left\_join(rest1, rest, by = c("Surname", "Ticket2"))

rest1 <- rest1[!is.na(rest1$count),]

rest1 <- rest1 %>%

arrange(Surname, Ticket2)

kable(rest1[1:12,])

#replacing Fsize size in my overall dataframe with the count numbers in the table above

all <- left\_join(all, rest1)

## Joining, by = c("PassengerId", "Age", "SibSp", "Parch", "Ticket", "Surname", "Title", "Fsize", "Ticket2")

for (i in 1:nrow(all)){

if (!is.na(all$count[i])){

all$Fsize[i] <- all$count[i]

}

}

## Did people book together?

Besides families, groups of friends can off course also travel together. A nice example of this is the ticket below.

kable(all[all$Ticket=='1601', c('Survived', 'Pclass', 'Title', 'Surname', 'Age', 'Ticket', 'SibSp', 'Parch', 'Fsize')])

Below, I am adding the number of people on each ticket as variable.

#composing data frame with group size for each Ticket

TicketGroup <- all %>%

select(Ticket) %>%

group\_by(Ticket) %>%

summarise(Tsize=n())

all <- left\_join(all, TicketGroup, by = "Ticket")

Very similarly to the family group sizes, small groups of 2-4 people traveling together on the same ticket have a higher chance of survival.

ggplot(all[!is.na(all$Survived),], aes(x = Tsize, fill = Survived)) +

geom\_bar(stat='count', position='dodge') +

scale\_x\_continuous(breaks=c(1:11)) +

labs(x = 'Ticket Size') + theme\_grey()

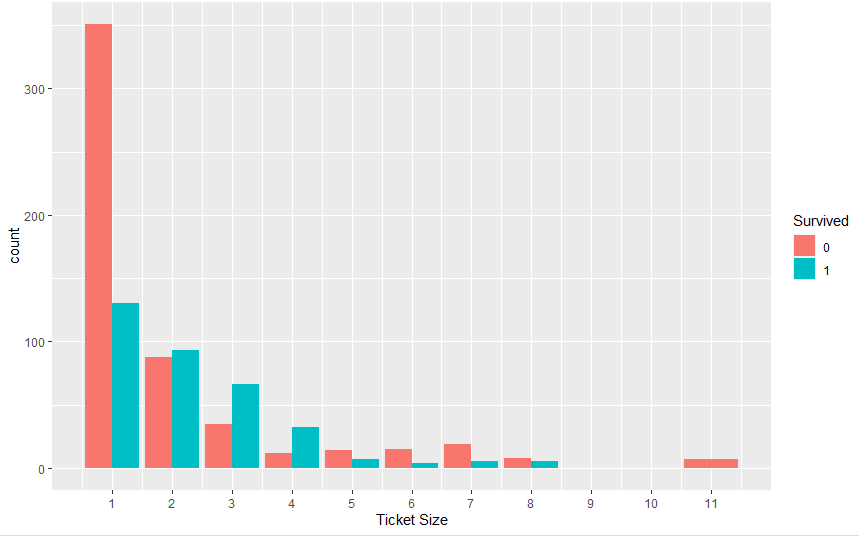


Figure . Book Together Information

As there is so much overlap between family size and ticket size, I am consolidating these two variables into one group variable. Now I can finally created my factorized variable for the group sizes.

#taking the max of family and ticket size as the group size

all$Group <- all$Fsize

for (i in 1:nrow(all)){

all$Group[i] <- max(all$Group[i], all$Tsize[i])

}

#Creating final group categories

all$GroupSize[all$Group==1] <- 'solo'

all$GroupSize[all$Group==2] <- 'duo'

all$GroupSize[all$Group>=3 & all$Group<=4] <- 'group'

all$GroupSize[all$Group>=5] <- 'large group'

all$GroupSize <- as.factor(all$GroupSize)

As ‘1’ and ‘2’ are large groups with their own typical survival rates, I am keeping them as separate groups. Sizes ‘3’ and ‘4’ clearly have the best survival chances, and the groups of 5 and more clearly have worse chances.

g1 <- ggplot(all[!is.na(all$Survived),], aes(x = Group, fill = Survived)) +

geom\_bar(stat='count', position='dodge') +

scale\_x\_continuous(breaks=c(1:11)) +

labs(x = 'Final Group Sizes') + theme\_grey()

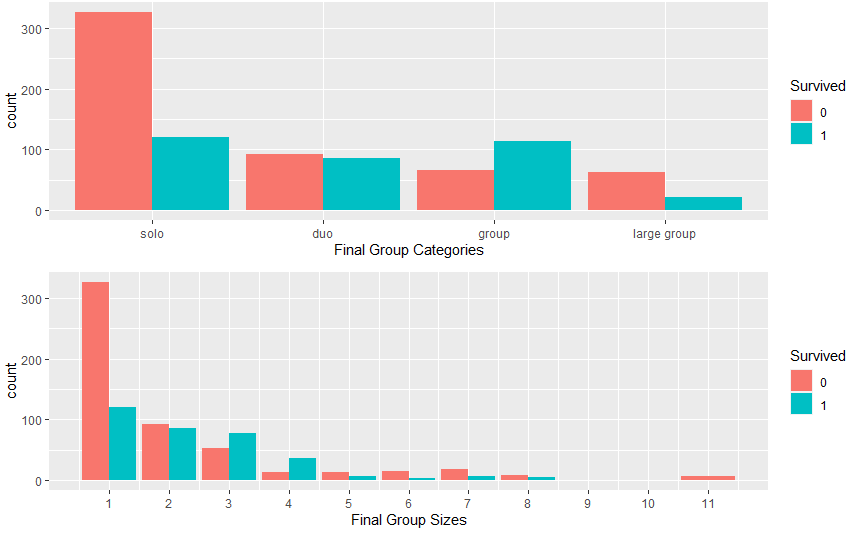
g2 <- ggplot(all[!is.na(all$Survived),], aes(x = GroupSize, fill = Survived)) +

geom\_bar(stat='count', position='dodge') +

labs(x = 'Final Group Categories') + theme\_grey() +

scale\_x\_discrete (limits = c('solo', 'duo', 'group', 'large group'))

grid.arrange(g2, g1)



## Dealing with the Fare variable

### Which data relevant to fare are missing?

There are two missing values in Embarked, and one in Fare. Embarked could be important to Fare, as different Embarkement cities mean longer or shorter journeys.

#display passengers with missing Embarked

kable(all[which(is.na(all$Embarked)),c('Surname', 'Title', 'Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked', 'Group') ])

Both women are traveling solo from a family perspective (Fsize=1), but must be friends as they are both are traveling on ticket 113572. Both also have the same fare, but this fare might still have been per person. I came to the conclusion that prices are indeed per ticket. As the explanation was getting lengthy, I will now just continue under the assumption that fares are per person.

I want to impute the missing embarkement city with the median Fare Per Person for each Embarkement city, and per Pclass.

all$FarePP <- all$Fare/all$Tsize #creating the Fare Per Person variable

tab2 <- all[(!is.na(all$Embarked) & !is.na(all$Fare)),] %>%

group\_by(Embarked, Pclass) %>%

summarise(FarePP=median(FarePP))

kable(tab2)

As the FarePP of those two women is 40, they most likely embarked at Cherbourgh.

#imputing missing Embarked values

all$Embarked[all$Ticket=='113572'] <- 'C'

#converting Embarked into a factor

all$Embarked <- as.factor(all$Embarked)

I can actually use the same table to find a sensible fare for Mr Story. As you can see below, he traveled 3rd class and embarked at Southampton.

#display passengers with missing Fare

kable(all[which(is.na(all$Fare)), c('Surname', 'Title', 'Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked', 'Group')])

#imputing FarePP (as the Fare will be dropped later on anyway)

all$FarePP[1044] <- 7.8

### The Fare Per Person Variable

Although there now are no missing FarePP’s anymore, I also noticed that 17 Fares actually have the value 0. These people are not children that might have traveled for free. I think the information might actually be correct, however I also think that the zero-Fares might confuse the algorithm. For instance, there are zero-Fares within the 1st class passengers. To avoid this possible confusion, I am replacing these values by the median FarePP’s for each Pclass.

tab3 <- all[(!is.na(all$FarePP)),] %>%

group\_by(Pclass) %>%

summarise(MedianFarePP=median(FarePP))

all <- left\_join(all, tab3, by = "Pclass")

all$FarePP[which(all$FarePP==0)] <- all$MedianFarePP[which(all$FarePP==0)]

Below you can see that the FarePP is very skewed. I know that this is not desirable for some algorithms, and can be solved by taking the logarithm or normalisation (preprocessing with centering and scaling).

ggplot(all, aes(x=FarePP)) +

geom\_histogram(binwidth = 5, fill='blue') + theme\_grey() +

scale\_x\_continuous(breaks= seq(0, 150, by=10))

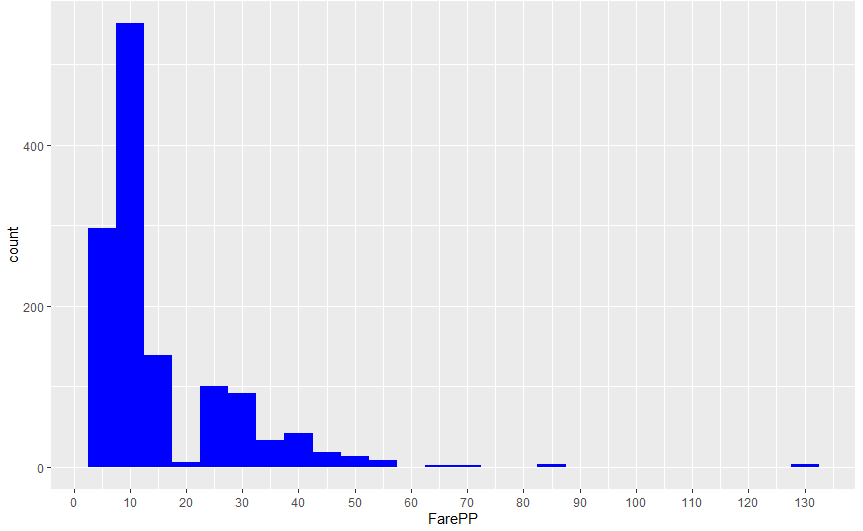


Figure . Fare Per Person

Another option is to use Fare Bins instead of keeping the FarePP as a numeric variable. I am using Fare Bin in the GBM model. As there are more FareBins than Pclasses, there is of course some overlap between FareBins and Pclasses.

#Note Hmisc needs to be loaded before dplyr, as the other way around errors occured due to the kernel using the Hmisc summarize function instead of the dplyr summarize function

all$FareBins <- cut2(all$FarePP, g=5)

ggplot(all[!is.na(all$Survived),], aes(x=FareBins, fill=Survived))+

geom\_bar(stat='count') + theme\_grey() + facet\_grid(.~Pclass)+

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

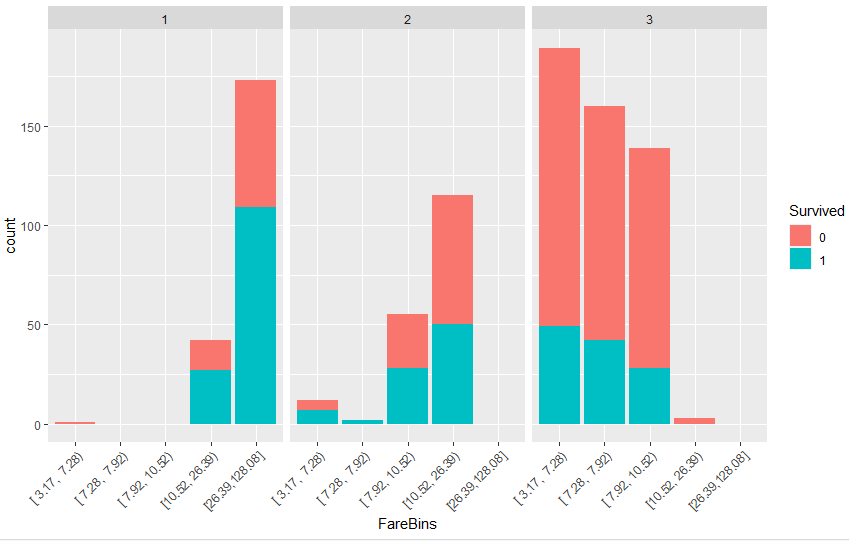


Figure . Fare Bins

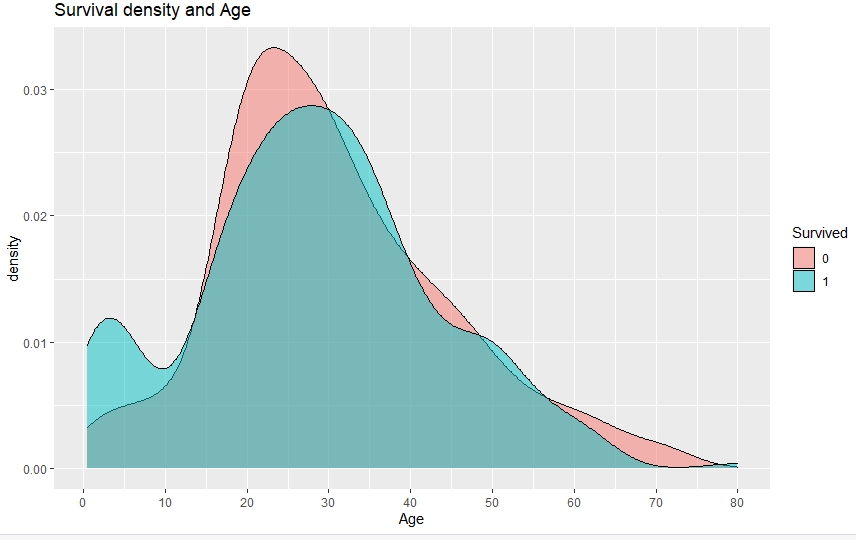
## Predicting missing Age values

The density plot below shows that survival chances of children are relatively high. Survival chances of ages 20-30 are below average, and I see less significant differences in the 30+ region. I think there may be a lot of solo travelers in the 20-30 category, which could explain the below averages survival chances. A possible use case of Age could be to use it to identify children. Therefore, I will focus on good looking Age imputations in the region 0-18 years old.

ggplot(all[(!is.na(all$Survived) & !is.na(all$Age)),], aes(x = Age, fill = Survived)) +

geom\_density(alpha=0.5, aes(fill=factor(Survived))) + labs(title="Survival density and Age") +

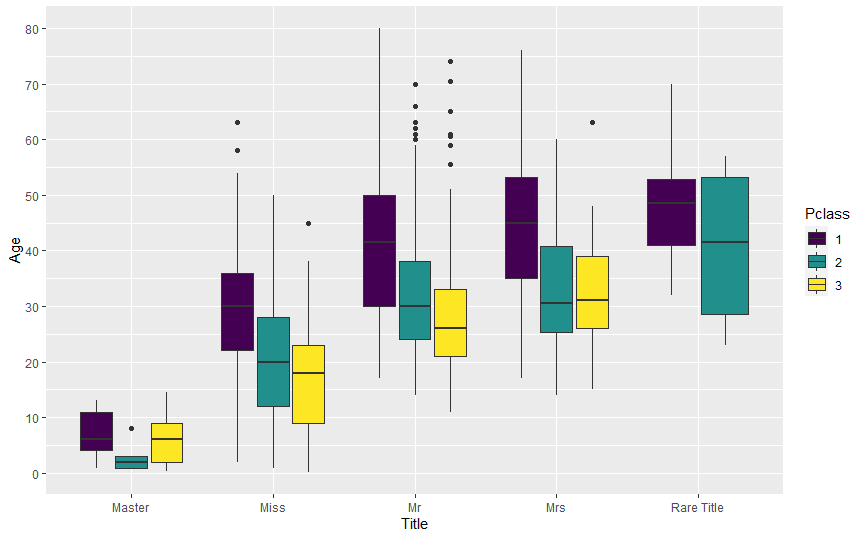
scale\_x\_continuous(breaks = scales::pretty\_breaks(n = 10)) + theme\_grey()



I first want to visualize the relation between the Age. Title and Pclass seem the most important predictors for Age to me. As you can see below, there are significant differences in Age across the Titles (By the way, this graph tells me that “Masters” are all very young. I did not know what a master was, but googling it tells me that a master was used as a title for the eldest son only.). Similarly, differences in Age when looking at the Title/Passenger Class combinations.

ggplot(all[!is.na(all$Age),], aes(x = Title, y = Age, fill=Pclass )) +

geom\_boxplot() + scale\_y\_continuous(breaks = scales::pretty\_breaks(n = 10)) + theme\_grey()



The title Master seems to be a good predictor for male children. However, female children are included in the Miss title, and of the 263 missing age values, 51 are Misses. If I would just take the median Age of the Titles (possibly also by Pclass), I would at least not predict the missing ages of female children well. I tried both Mice imputation and Linear Regression, and focused on how good the imputations for children looked. The Mice imputations looked reasonable, but I preferred Linear Regression.

#predicting Age with Linear Regression

set.seed(12000)

AgeLM <- lm(Age ~ Pclass + Sex + SibSp + Parch + Embarked + Title + GroupSize, data=all[!is.na(all$Age),])

summary(AgeLM)

all$AgeLM <- predict(AgeLM, all)

As expected, the most significant predictors according to Linear Regression were Passenger Class and Title. Below you can see that the histogram of the predicted values versus the shape of the known ages. The Mice histogram actually looked nicer, but I was wondering how it could predict high ages well given the sparseness of these ages in the original data?

par(mfrow=c(1,2))

hist(all$Age[!is.na(all$Age)], main='Original data, non-missing', xlab='Age', col='green')

hist(all$AgeLM[is.na(all$Age)], main= 'LM NA predictions', xlab='Age', col='orange', xlim=range(0:80))

As mentioned before, I especially looked at young predicted ages. Both mice and Linear Regression predicted all Masters with missing ages to be children indeed (the one in Linear Regression with a negative age did not bother me that much, as it is categorized as a child anyway). Mice predicted some Mr.’s to be 14 years old, which is too young. As Linear Regression also predicted a reasonable number of Misses to be children, I eventually chose Linear Regression.

#display which passengers are predicted to be children (age<18) with Linear Regression.

all[(is.na(all$Age) & all$AgeLM <18), c('Sex', 'SibSp', 'Parch', 'Title', 'Pclass', 'Survived', 'AgeLM')]

#imputing Linear Regression predictions for missing Ages

indexMissingAge <- which(is.na(all$Age))

indexAgeSurvivedNotNA<- which(!is.na(all$Age) & (!is.na(all$Survived))) #needed in sections 4.6 and 4.7

all$Age[indexMissingAge] <- all$AgeLM[indexMissingAge]

So now all missing data have been imputed. Am I going to use Age as a predictor in my model? I am not sure yet, as the substantial number of imputations will also add noise. I will look at using it to create a Child predictor later on.

## What to do with Cabin?

Cabin is very sparsely populated. So I either have to ignore it, or use it somehow without making it too specific. On the internet, you can find that that the first letter corresponds to the Deck. Decks A-E are the topdecks and cabins on those decks are mostly first class.

#replacing NAs with imaginary Deck U, and keeping only the first letter of each Cabin (=Deck)

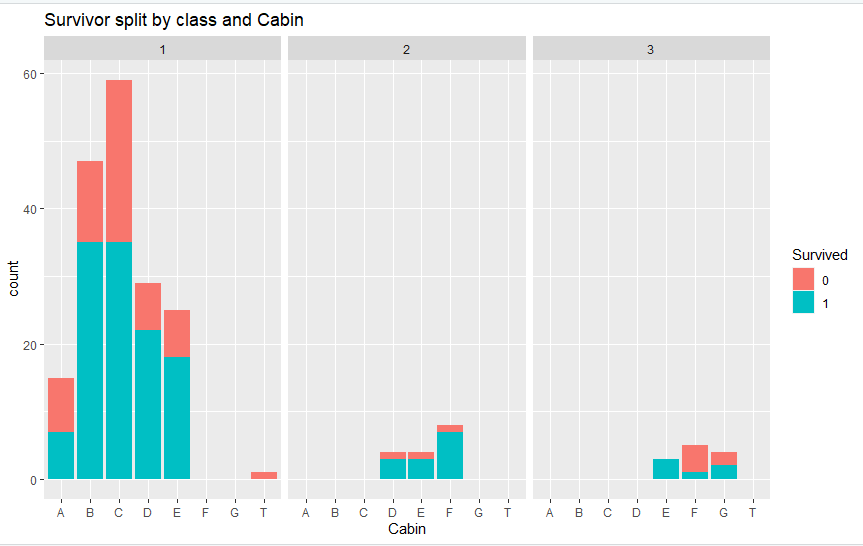
all$Cabin[is.na(all$Cabin)] <- "U"

all$Cabin <- substring(all$Cabin, 1, 1)

all$Cabin <- as.factor(all$Cabin)

ggplot(all[(!is.na(all$Survived)& all$Cabin!='U'),], aes(x=Cabin, fill=Survived)) +

geom\_bar(stat='count') + theme\_grey() + facet\_grid(.~Pclass) + labs(title="Survivor split by class and Cabin")



Below, you can see that there are interesting difference among Decks. For instance, the top Deck (A) was not best place to be. Even Deck F had better survival rates.

c1 <- round(prop.table(table(all$Survived[(!is.na(all$Survived)&all$Cabin!='U')], all$Cabin[(!is.na(all$Survived)&all$Cabin!='U')]),2)\*100)

kable(c1)

Although I feel that Deck and Deck sections (front/back of boat, sections close to stairs et cetera) would be great predictors, I am not using Cabin due to the sparseness of the data.

## How to deal with Children in the model?

The survival density plot in the Age section shows that Children below roughly 14.5 (which is also the maximum Age of Masters in the data) have a better survival rate than then other Ages. However, if you look at the imputed Ages below 14.5, you will also see that all these age imputation are for Pclass 3 and most of these children actually died (10 out of 13).

This makes me wonder if I should add a survival ‘bonus’ for all Pclasses. Below you can see that most children in P3 actually die. As these children in P3 also include age imputations which may add noise, I decided to exclude P3 from the Child predictor.

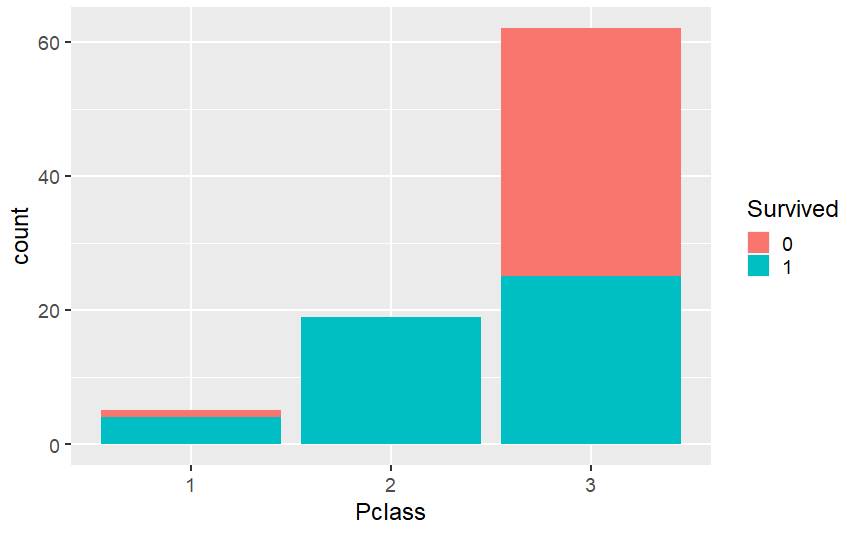
ggplot(all[all$Age<14.5 & !is.na(all$Survived),], aes(x=Pclass, fill=Survived))+

geom\_bar(stat='count') + theme\_grey(base\_size = 18)

all$IsChildP12 <- 'No'

all$IsChildP12[all$Age<=14.5 & all$Pclass %in% c('1', '2')] <- 'Yes'

all$IsChildP12 <- as.factor(all$IsChildP12)



## What does Embarked tell us?

Although I feel that the city of Embarked should not be related to survival rates, I still wanted to check it. As you can see below, there somehow are significant differences between the three ports of embarkment.

d1 <- ggplot(all[!is.na(all$Survived),], aes(x = Embarked, fill = Survived)) +

geom\_bar(stat='count') + theme\_grey() + labs(x = 'Embarked', y= 'Count')

d2 <- ggplot(all[!is.na(all$Survived),], aes(x = Embarked, fill = Survived)) +

geom\_bar(stat='count', position= 'fill') + theme\_grey() + labs(x = 'Embarked', y= 'Percent')

grid.arrange(d1, d2, nrow=1)

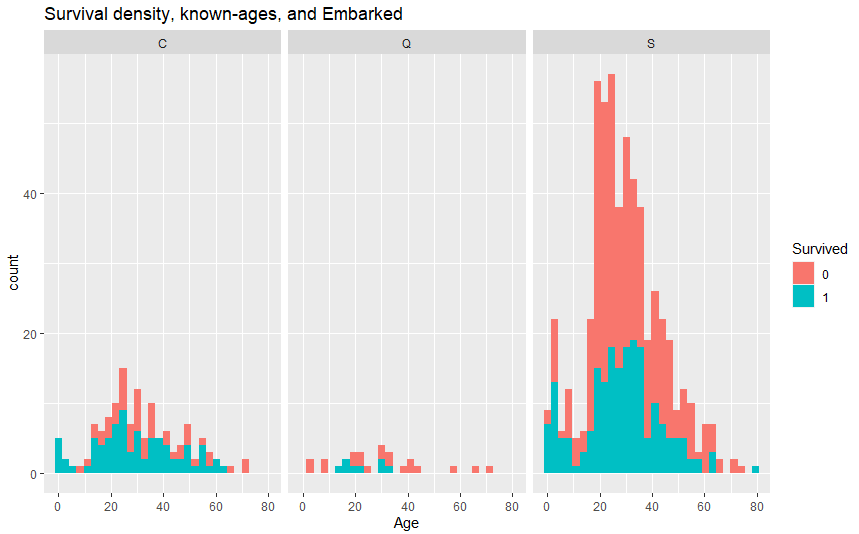
To get a feel for where this differences may come from, I plotted them against Sex and Pclass. Roughly, differences were: Southampton survival rates are worse than Cherbourg in all Pclass/Sex combinations. Cherbourg survival rates are better than Queenstown as many 1st class passengers boarded at Cherbourgh, while almost all Queenstown passengers boarded 3rd class (but within 3rd class, female survival rate is better than Cherbourg and male survival rate is worse than Cherbourgh).

My conclusion is that at least the lower survival rate of Southampton compared to Cherbourg cannot be explained by Pclass or Sex. One thing that I want to look at is the relation between Embarked, Age and Survived, because Linear Regression surprisingly enough also labeled Embarked at Queenstown as a significant predictor for Age. Below I am only using the known Ages of the training data (714 observation = training set - 177 observations with missing Age).

ggplot(all[indexAgeSurvivedNotNA,], aes(x = Age, fill = Survived)) +

geom\_histogram(aes(fill=factor(Survived))) + labs(title="Survival density, known-ages, and Embarked") +

scale\_x\_continuous(breaks = scales::pretty\_breaks(n = 5)) + theme\_grey() + facet\_grid(.~Embarked)



This shows that is very little data for especially Queenstown when looking at known Ages. Below you can see that the total number of people who embarked at Queenstown is low indeed, but especially the high percentage of missing ages in Queenstown is really high. Using imputed ages will therefore add too much noise, and combining Age and Embarked as a predictor is a bad idea.

tab1 <- rbind(table(all$Embarked[!is.na(all$Survived)]),table(all$Embarked[indexAgeSurvivedNotNA]))

tab1 <- cbind(tab1, (rowSums(tab1)))

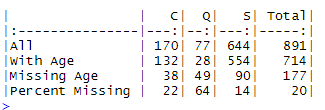
tab1 <- rbind(tab1, tab1[1,]-tab1[2,])

tab1 <- rbind(tab1, round((tab1[3,]/tab1[1,])\*100))

rownames(tab1) <- c("All", "With Age", "Missing Age", "Percent Missing")

colnames(tab1) <- c("C", "Q", "S", "Total")

kable(tab1)



The only other thing that I can think of that might explain the differences is that probably people from the different embarkement cities are somehow grouped on certain sections of the decks.

I kept Embarked in my model in early versions. However, it gradually became clear that Embarked does not add anything and I am not using it anymore.

## Ticket survivors

This variable checks if any people in a group survived. The idea is that if anyone in a certain group survived, chances of others also surviving are higher. I did this using the Ticket information, and it improved the scores.

TicketSurvivors <- all %>%

group\_by(Ticket) %>%

summarize(Tsize = length(Survived),

NumNA = sum(is.na(Survived)),

SumSurvived = sum(as.numeric(Survived)-1, na.rm=T))

all <- left\_join(all, TicketSurvivors)

## Joining, by = c("Ticket", "Tsize")

all$AnySurvivors[all$Tsize==1] <- 'other'

all$AnySurvivors[all$Tsize>=2] <- ifelse(all$SumSurvived[all$Tsize>=2]>=1, 'survivors in group', 'other')

all$AnySurvivors <- as.factor(all$AnySurvivors)

kable(x=table(all$AnySurvivors), col.names= c('AnySurvivors', 'Frequency'))

4.9 Adding an “Is Solo” variable" based on Siblings and Spouse (SibSp) only

In an earlier version, I experimented with an “IsSolo” predictor that was based on the Group size. However, this double counted the Group categories too much and did not work. Eventually, I added an IsSolo predictor that is only based on the SibSp information. Using this predictor in the SVM model leads to slightly better results.

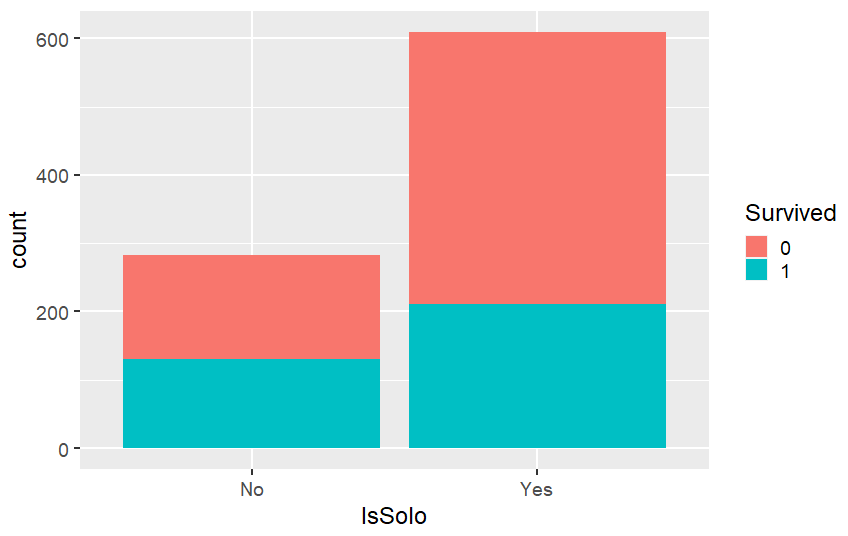
all$IsSolo[all$SibSp==0] <- 'Yes'

all$IsSolo[all$SibSp!=0] <- 'No'

all$IsSolo <- as.factor(all$IsSolo)

ggplot(all[!is.na(all$Survived),], aes(x = IsSolo, fill = Survived)) +

geom\_bar(stat='count') + theme\_grey(base\_size = 18)



# PREDICTIONS

Altogether, I created predictions with 3 different algorithms. In addition, I tried to combine (ensemble) the models in 3 different ways. This ensemble further improved the scores.

#splitting data into train and test set again

trainClean <- all[!is.na(all$Survived),]

testClean <- all[is.na(all$Survived),]

## Random Forest model

I started this analysis with just a Random Forest model, as it is known for high accuracy and limiting overfitting. Although the formula function must be used with many algorithms, it is better to not use it with Random Forest as this causes issues with weights of predictors. I am just using 5 predictors.

set.seed(2017)

caret\_matrix <- train(x=trainClean[,c('PclassSex', 'GroupSize', 'FarePP', 'AnySurvivors', 'IsChildP12')], y=trainClean$Survived, data=trainClean, method='rf', trControl=trainControl(method="cv", number=5))

caret\_matrix

caret\_matrix$results

#extracting variable importance and make graph with ggplot (looks nicer that the standard varImpPlot)

rf\_imp <- varImp(caret\_matrix, scale = FALSE)

rf\_imp <- rf\_imp$importance

rf\_gini <- data.frame(Variables = row.names(rf\_imp), MeanDecreaseGini = rf\_imp$Overall)

ggplot(rf\_gini, aes(x=reorder(Variables, MeanDecreaseGini), y=MeanDecreaseGini, fill=MeanDecreaseGini)) +

geom\_bar(stat='identity') + coord\_flip() + theme(legend.position="none") + labs(x="") +

ggtitle('Variable Importance Random Forest') + theme(plot.title = element\_text(hjust = 0.5))

#using the model to make Survival predictions on the test set

solution\_rf <- predict(caret\_matrix, testClean)

## Support Vector Machine (SVM) model

The second algorithm that I want to use is SVM, as it is known to work well with small datasets. As I am only having a few predictors and relatively many observation, I am choosing svmRadial (Gaussian) over svmLinear.

set.seed(2017)

caret\_svm <- train(Survived~ PclassSex + FarePP + AnySurvivors + IsChildP12 + IsSolo, data=trainClean, method='svmRadial', preProcess= c('center', 'scale'), trControl=trainControl(method="cv", number=5))

caret\_svm

caret\_svm$results

#using the model to make Survival predictions on the test set

solution\_svm <- predict(caret\_svm, testClean)

## Gradient Boosting Machine (GBM) model

As I am already having a model that uses Bagging, I want the 3rd model to be a boosting model. Of the possible boosting algorithms, I am choosing GBM.

set.seed(2017)

caret\_boost <- train(Survived~ PclassSex + GroupSize + FareBins + AnySurvivors + IsChildP12, data=trainClean, method='gbm', preProcess= c('center', 'scale'), trControl=trainControl(method="cv", number=7), verbose=FALSE)

print(caret\_boost)

solution\_boost <- predict(caret\_boost, testClean)

## Combining models

### Majority vote ensemble of the three models

A simple majority vote a multiple good models can help to increase accuracy. This idea is really well explained in Kaggle Ensembling Guide. It works best with multiple good models that are as uncorrelated as possible.

#adding model predictions to test dataframe

testClean$RF <- as.numeric(solution\_rf)-1

testClean$SVM <- as.numeric(solution\_svm)-1

testClean$Boost <- as.numeric(solution\_boost)-1

#compose correlations plot

corrplot.mixed(cor(testClean[, c('RF', 'SVM', 'Boost')]), order="hclust", tl.col="black")

Given the fact that all three models have decent public scores, especially the correlation between SVM and the GBM model is surprisingly low. The most likely explanation is that SVM really is a different algorithm (both other models are tree-based).

The idea is very simple:

If 0 or 1 model predicts ‘Survived’, the overall prediction will be ‘Died’

If 2 or 3 models predict ‘Survived’, the overall prediction will be ‘Survived’

testClean$Sum <- testClean$RF + testClean$SVM + testClean$Boost

testClean$Majority <- ifelse(testClean$Sum<=1, 0, 1)

### Taking predictions from one model, unless the others both disagree

Although I have done my best to avoid over fitting as much as possible (by for instance not using the Titles), the high cross validation scores of both RF and GBM are an indication that these 2 models still overfit somewhat.

The best kernels on Kaggle show that a public score of around 0.82 - 0.83 is likely to be the maximum achievable test accuracy. The cross validation score of the SVM model is exactly in this range. Probably even more importantly, a low AccuracySD seems more important than the Accuracy itself regarding Public Scores. As the SVM also has a really low AccuracySD, I am going to use SVM as my best model. In this second framework, I am taking the SVM predictions unless both RF and GBM disagree with the SVM prediction.

testClean$DisagreeSVM <- ifelse(testClean$RF==testClean$Boost & testClean$SVM != testClean$RF, testClean$RF, testClean$SVM)

### Selectively combining models for PclassSex combinations

Another idea is to select the model that seems best for each PclassSex combination. When looking at predictions on the training set, issues seem similar in all models. All produce tend to produce False Negatives for men (men predicted to die, but survived), and False Positives for women (predicted to survive, but died).

#predictions of the models on the training set

trainClean$RF <- predict(caret\_matrix, trainClean)

trainClean$SVM <- predict(caret\_svm, trainClean)

trainClean$Boost <- predict(caret\_boost, trainClean)

#plot differences between actual survived and predictions

f1 <- ggplot(trainClean[trainClean$Survived != trainClean$RF,], aes(x=PclassSex, fill=RF)) +

geom\_bar(stat='count') + labs(title="FP and FN, RF model") + theme\_grey() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +

theme(legend.position="none") + xlab("")

f2 <- ggplot(trainClean[trainClean$Survived != trainClean$SVM,], aes(x=PclassSex, fill=SVM)) +

geom\_bar(stat='count')+ labs(title="FP and FN, SVM") + theme\_grey() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +

theme(legend.position="none") + xlab("")

f3 <- ggplot(trainClean[trainClean$Survived != trainClean$Boost,], aes(x=PclassSex, fill=Boost)) +

geom\_bar(stat='count')+ labs(title="FP and FN, GBM") + theme\_grey() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +

theme(legend.position="none") + xlab("")

grid.arrange(f1, f2, f3, nrow = 1)

A noticeable difference however is that the SVM model also produced a significant amount of False Positives in P3Male (the blue part). As the survival rate of men in P3 was really low, this large amount of False Positives seems a bad sign. What I now could do is take SVM as the base model, but take GBM for the P3 predictions.

#selecting SVM prediction, and GMB predictions for P3

testClean$Select <- ifelse(testClean$Pclass != 3, testClean$SVM, testClean$Boost)

The best framework turned out to be the combination of SVM for P1 and P2, and GBM for P3. The majority voting and ‘take SVM unless both RF and GBM disagree’ did not improve my score. However, chances are that these methods will also improve the scores if I add a few more good models. This should decrease of 2 (alternative) models both getting a prediction wrong.

#writing final submission file

submission\_select <- data.frame(PassengerId = test$PassengerId, Survived = testClean$Select)

write.csv(submission\_select, file = 'Titanic\_select.csv', row.names = F)

## Where is room for improvement?

### How to fix Male in 1st class issues?

This class is hard to predict, as 37% of men in P1 survived while men overall have a much lower survival rate. As shown in the previous section, predictions include many False Negatives. Many False Negatives means that the predicted number of survivors on the test set is likely to be too low. As you can see below, all models predict the survival rate of this group to be significantly lower than 37% indeed (.37\*57=21 survivors).

cat('Total number of Male passengers in P1 in the test set is', length(testClean$Survived[testClean$PclassSex=='P1Male']))

## Total number of Male passengers in P1 in the test set is 57

p1m\_surv <- as.data.frame(sapply(testClean[testClean$PclassSex=='P1Male', c('RF', 'SVM', 'Boost')], function(x) {sum(x)}))

kable(x=p1m\_surv, col.names = c('Predicted number of survivors'))

Although SVM only predicts 2 survivors, these 2 are in fact the 2 children in this class. This might actually be the best bet, if survival is hard to predict (predicting dead is better when unclear as the majority died). I do not know if the survival rate of P1Male in the test set is also 37%, but it seems likely that the predicted survival rate for this group is too low in all models. Therefore, I have looked for angles to ‘reliably’ increase this number. As you can see below, there is a significant difference in survival density of men below and above approximately 40. So the survival ‘bonus’ that children have in general, is for P1Male ‘extended’ to 40.

I have actually tried to include this exception in the models, but It did not improve the scores. I assume this is because the survival rate in P1Male under 40 now really gets close to 50/50, which makes it very hard to predict with some degree of certainty.

p1m1 <- ggplot(all[indexAgeSurvivedNotNA,] %>% filter(PclassSex=='P1Male'), aes(x = Age, fill = Survived)) + geom\_density(alpha=0.5, aes(fill=factor(Survived))) + labs(title="Survival density and Age P1 Male") + theme\_grey()

all$P1AgeMale[indexAgeSurvivedNotNA=T & all$PclassSex=='P1Male' & all$Age<40] <- 'Under40'

all$P1AgeMale[indexAgeSurvivedNotNA=T & all$PclassSex=='P1Male' & all$Age>=40] <- 'Over40'

p1m2 <- ggplot(all[!is.na(all$Survived) & !is.na(all$P1AgeMale),], aes(x=P1AgeMale, fill=Survived))+

geom\_bar(stat = 'count', position = 'fill') + theme(legend.position="none")

grid.arrange(p1m1, p1m2, widths=c(2,1))

## How to fix Female in 3rd class issues?

Females in 3rd class have an overall 50/50 survival chance, which is hard to predict. In the previous section, P3Female showed many False Positives. This was expected, as women generally survived. Similar to the males in P1, I found something that is specific to this group.

As you can see below, solo women (based on SibSp) had better chances of survival in all Pclasses. This is especially relevant to the women in P3, as differences in P2Female and P1Female are small while they mostly survived anyway. In previous versions, I tried a different IsSolo predictor (based on Fsize), and did not manage to model this exception succesfully. However, it may be worth trying a Male/Female split with the IsSolo predictor based on SibSp only.

ggplot(all[!is.na(all$Survived),], aes(x=IsSolo, fill=Survived))+

geom\_bar(stat='count', position='fill') + facet\_grid(.~Pclass+Sex)