I'll Never Let Go, Jack
Stat 151A

Michael Knopf

May 6, 2015

1 Introduction

After performing an exhaustive search, I am frustrated that I have found neither Jack nor Rose in the Titanic dataset. Clearly, there are missing observations. I have refrained from adding 2 rows - one with $Survived = 1$ and the other with $Survived = 0$ - to reflect Jack and Rose in the data, since I would have to rewatch the entire movie to determine the other values for these observations.

From my initial intuition, I would expect *Age*, *Parch*, and *SibSp* to be useful in prediction. However, they may need to be transformed in order to be useful, and their interactions should be considered as well. For instance, if a young child was on-board with a parent, it is likely that the parent would try to ensure the child's survival, even to the point of disregarding his or her own (unless they are a terrible parent). However, a child on board without his or her parents might be at a disadvantage.

2 Data

A description of the provided data is tabulated below.

Variable	Description
PassengerId	A unique ID for each passenger in the dataset
Survived	0: Died, 1: Survived
Pclass	A proxy for passenger class, 1 being the highest class and 3 the lowest.
SibSp	The sum total of the number of siblings or spouses aboard with the passenger
Parch	The sum total of the number of parents or children aboard with the passenger
Fare	Fare price
Embarked	Port departed from: Cherbourg, Queenstown, Southhampton

The first issue with the data is missing values for *Age*. Because *Age* proved to be an important explanatory variable, I did not want to simply exlude it from the model. One method of dealing with this that I considered was to first predict missing values of *Age* from the other provided variables (excluding *Survived*), and then use the predicted values as substitutes in the main model. However, this method would compound errors and increase multicolinearity. Since there are a significant number of missing values (177 out of 891 in the training set, 86 outof 418 in the test set), this is not an acceptable solution.

The solution I decided on was to use two separate models: one including *Age*, and one not including it. For the model including *Age*, I fit the response to the set of observations where *Age* was recorded. For the model without *Age*, I fit the response to the entire training set.

A second question that arose was whether certain continuous variables should be convereted to categorical. In many models, the significance of *Parch* was improved by transforming it to $(Parch == 0)$, and *Embarked* was only useful when looked at as the variable $(Embarked ==' S')$. However, in the model I ended up choosing, neither of these were useful.

The only change I ended up making was on the variable *Age*. It turned out that the given continuous variable was less important than the categories "child," "young adult," and "old adult." The question of what age ranges define these categories was something I had to decide after inspecting several models, but the ranges that eventually proved most effective were $Age \leq 9$, $10 \leq Age \leq 36$, and $Age \geq 37$.

The last topic worth mentioning is interaction terms. The only interaction that seemed worth using was that between the *Age* categories and the *SibSp* variable. This interation term was significant in the model, however I chose not to use it because I was very concerned with overfitting. With the amount of data present, I was more concerned with limiting model size at that point.

3 Variable Selection

Due to the small number of explanatory variables, it is computationally possible to simply compare all possible models. This simplified the analysis greatly. I compared the *AIC* for every possible model and found that the best scoring models (up to whether or not they include *Age*) were the two I had already suspected were the best fits (*Age.cat* is the categorical variable described in section 2):

$$Survived = Pclass + Sex + Age.cat + SibSp$$

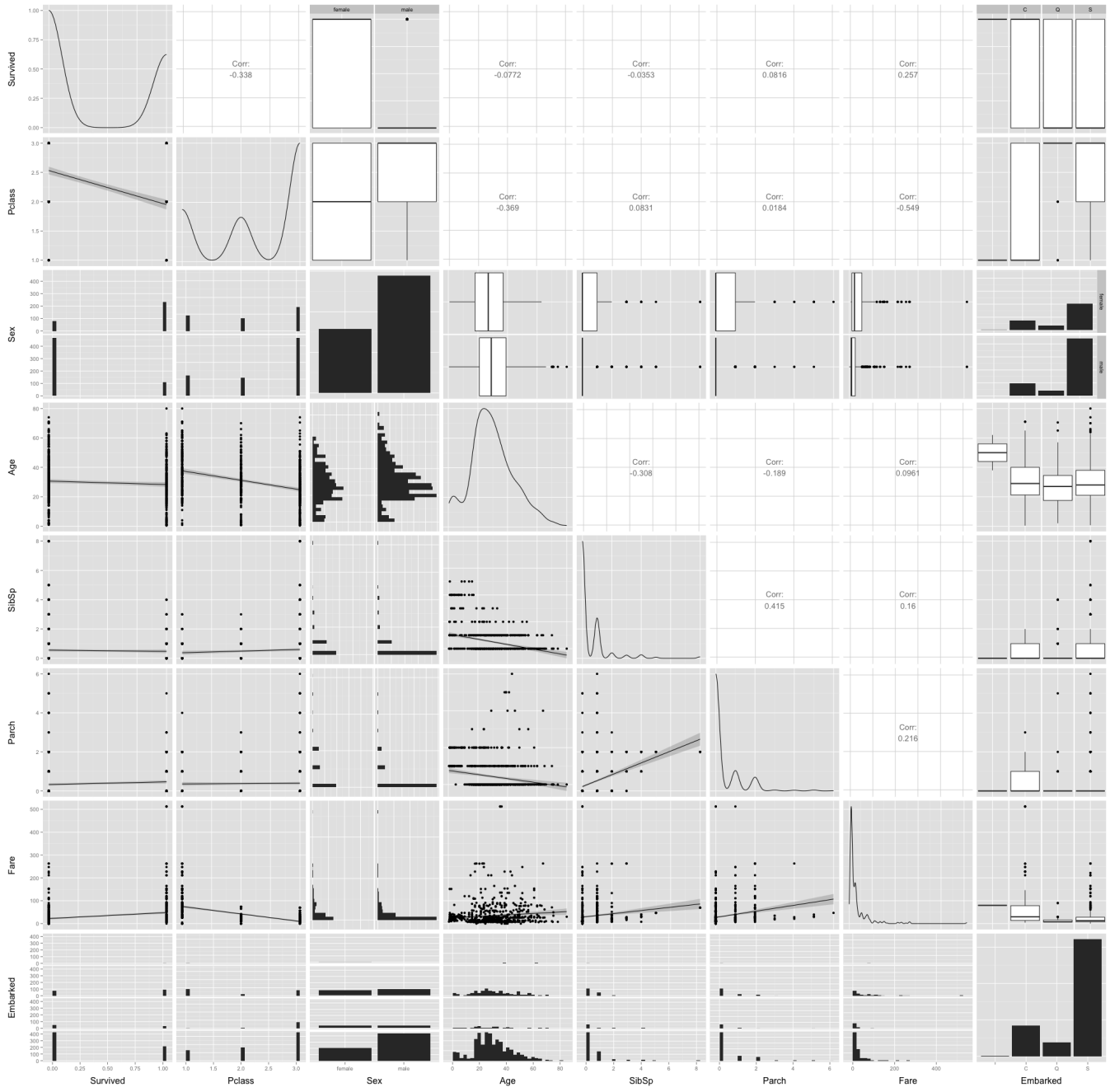

Figure 1: Pairwise plots of all present variables. Relationships between the variables used in the chosen models are minimal.

$$Survived = Pclass + Sex + SibSp$$

The way I originally found these models was by backward selection based on p-value, so I was satisfied that they had the best AIC score as well.

Next, I used a confusion matrix to find the best threshold for prediction. I went through a set of 1000 possible thresholds, looking for which produced the best results. All such values were consecutive, so I took their mean as my final cutoff. The thresholds I chose were .52 and .62, respectively.

After looking at many aspects of other candidate models, I finally performed 7-fold cross-

validation on these models, as well as on all others possible, to evaluate their performance. I used two methods: in one, I measured deviance to be the difference between the predicted probability of survival and the observed survival status; in the other, I simply looked at the number of incorrect predictions, where I predict based on whether the probability is above the threshold or not. The statistics I took into account were

1. Sum of the deviances for each test set.

2. Variance of deviances across the test sets.

Of course, we want the total deviance of the model to be low. However, if the model's performance varies greatly between test sets, this is a problem as well. The models I chose had the best score in the deviance category, and their scores were very low in the variance category. Overall, they were the best performing models. Summaries of the cross-validation are given below in Figure 2.

4 Predictions

Using the thresholds determined by confusion matrices, I predicted the value of *Survived* for the test set observations. For those which inclued *Age*, I used the first model. For those that did not, I use the second.

Next, I used classification trees to predict these values as well. I will not go into the methods I used to construct the trees, because it was not the main focus of my analysis. However, the predictions only differed on 34 observations, and for the vast majority of these, the predicted probability of survival was very close to the threshold. Also, whether or not *Age* was present in these observations was insignificant, which is good (the proportion of observations without *Age* was .2057416 in the test set .2058824 in the set of observations on which the two prediction methods disagreed).

In order to predict for these observations, I looked at them individually, adjusting the threshold and sometimes looking at how a model with one extra variable would predict their survival. Ultimately, I split them into groups based on the glm predicted probability and made decisions for each group on a case-by-case basis, since there were not very many of them.
The prediction success rate was 0.77751.

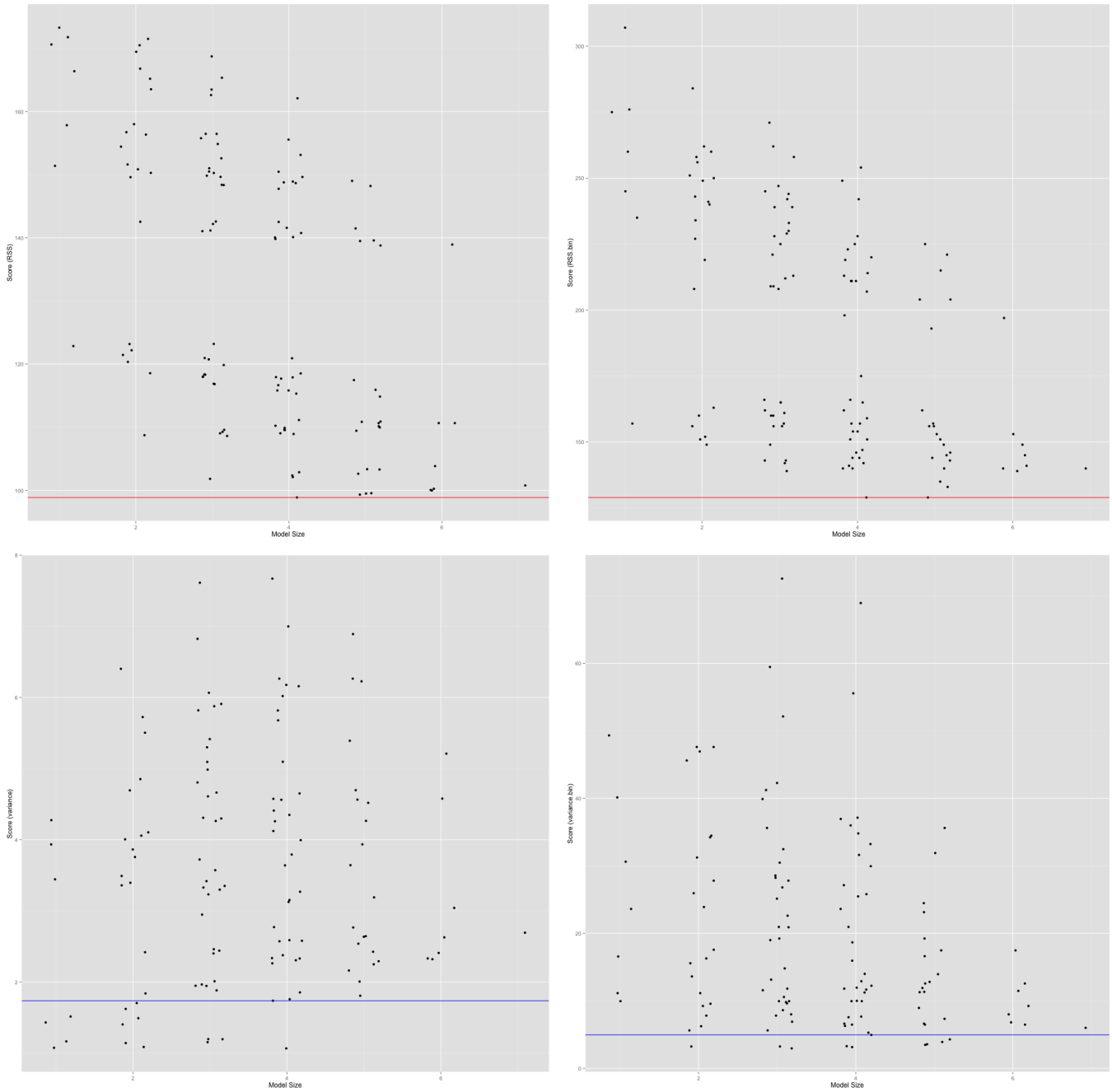

Figure 2: On the left are CV scores using predicted probabilities. On the right are CV scores using pure predictions. The top plots are total deviance against model size, the bottom are variance of deviances across the test sets against model size. The lines mark the scores of the chosen model (including *Age*).

5 Code

The following code is not commented well. I did not have as much time to spend on this project as I did on the second midterm. Many times, I went back and edited old code in order to reuse it. Also, some variables have been instantiated or updated in one script, but are also being used in another. Ultimately, the code worked perfectly for my purposes, but is no longer very readable.

First inspection of data:

```
library(mlbench)
library(ggplot2)
library(plyr)
library(reshape2)
library(grid)
library(lattice)
library(GGally)
library(rpart)

setwd("~/Desktop/Stat 151/Titanic")
test = read.csv('test.csv')
train = read.csv('train.csv')

allvars = c('Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked')
data = train[,allvars]

# Create pairwise plots of explanatory variables
ggpairs(data, lower=list(continuous="smooth"))

ft = glm(Survived ~ ., family = "binomial", data = data)
summary(ft)

ft = glm(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare, family = "binomial", data = data)
summary(ft)

ft = glm(Survived ~ Pclass + Sex + Age + SibSp + Fare, family = "binomial", data = data)
summary(ft)

ft = glm(Survived ~ Pclass + Sex + Age + SibSp, family = "binomial", data = data)
summary(ft)

qplot(x=jitter(ft$y), y=ft$fitted.values, geom = c("point", "smooth"), method = "lm")

summary((ft$fitted.values)[ft$y == 0])
summary((ft$fitted.values)[ft$y == 1])

plots.simple = list()
plots.simple[[1]] = qplot(x=jitter(Pclass), y=jitter(Survived), data = data, geom = c("point", "
    smooth"), method = "lm", xlab = 'Pclass', ylab = "")
plots.simple[[2]] = qplot(x=jitter(as.numeric(Sex)%%2), y=jitter(Survived), data = data, geom = c("
    point", "smooth"), method = "lm", xlab = 'Sex', ylab = "")
plots.simple[[3]] = qplot(x=na.omit(data$Age), y=jitter(data$Survived[!is.na(data$Age)]), geom = c(
    "point", "smooth"), method = "lm", xlab = 'Age', ylab = "")
plots.simple[[4]] = qplot(x=jitter(SibSp), y=jitter(Survived), data = data, geom = c("point", "
    smooth"), method = "lm", xlab = 'SibSp', ylab = "")
plots.simple[[5]] = qplot(x=jitter(Parch), y=jitter(Survived), data = data, geom = c("point", "
    smooth"), method = "lm", xlab = 'Parch', ylab = "")
plots.simple[[6]] = qplot(x=Fare, y=jitter(Survived), data = data, geom = c("point", "smooth"),
    method = "lm", xlab = 'Fare', ylab = "")

multiplot(plotlist = plots.simple, cols = 3)

ft = glm(Survived ~ Embarked, family = "binomial", data = data)
summary(ft)
# Embarked appears highly insignificant


# Should SibSp be categorical?
# We will use age, so remove observations where age is NA
data = data[!is.na(data$Age),]
sum(data$SibSp > 1)
plot(data$SibSp[data$SibSp > 1])


# Check the difference in means of survival between SibSp groups
data$SibSp1 = data$SibSp == 1
data$SibSp2 = data$SibSp == 2
data$SibSp3 = data$SibSp == 3
data$SibSp4 = data$SibSp == 4
```

```
ft = lm(Survived ~ SibSp1 + SibSp2 + SibSp3 + SibSp4, data = data)
summary(ft)

mean(data$Survived)
mean(data$SibSp == 0)
mean(data$SibSp1)
mean(data$SibSp2)
mean(data$SibSp3)
mean(data$SibSp4)

data$SibSp.new = (data$SibSp == 1 | data$SibSp == 2)

ft1 = glm(Survived ~ Pclass + Sex + Age + SibSp.new, family = "binomial", data = data)
summary(ft)
summary(ft1)

data$SibSp.new = (data$SibSp == 0)
ft1 = glm(Survived ~ Pclass + Sex + Age + SibSp.new, family = "binomial", data = data)
summary(ft)
summary(ft1)

data$SibSp.new = data$SibSp
ft1 = glm(Survived ~ Pclass + Sex + Age + SibSp.new, family = "binomial", data = data)
summary(ft)
summary(ft1)
```

Second inspection of data:

```
library(mlbench)
library(ggplot2)
library(plyr)
library(reshape2)
library(grid)
library(lattice)
library(GGally)
library(rpart)

setwd("~/Desktop/Stat 151/Titanic")
test = read.csv('test.csv')
train = read.csv('train.csv')

allvars = c('Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked')
data = train[,allvars]

# Create pairwise plots of explanatory variables
ggpairs(data, lower=list(continuous="smooth"))

# Age is not very significant
ft = glm(Survived ~ Age, family = "binomial", data = data)
summary(ft)
# Optimal cutoff for Age is 13
ft = glm(Survived ~ (Age<=13), family = "binomial", data = data)
summary(ft)

# Pclas is significant
ft = glm(Survived ~ Pclass, family = "binomial", data = data)
summary(ft)


# Parch alone is not very significant
ft = glm(Survived ~ Parch, family = "binomial", data = data)
summary(ft)
# Optimal cutoff for Parch is 0.
ft = glm(Survived ~ (Parch==0), family = "binomial", data = data)
summary(ft)

# Fare is significant
ft = glm(Survived ~ Fare, family = "binomial", data = data)
summary(ft)

# Embarked is not significant
ft = glm(Survived ~ Embarked, family = "binomial", data = data)
```

```
summary(ft)
# Embarked == 'S' is significant.  Most people embarked from this location.
ft = glm(Survived ~ (Embarked=='S'), family = "binomial", data = data)
summary(ft)

data$Age.13 = data$Age < 13
data$Parch.0 = data$Parch == 0
data$Embarked.S = data$Embarked == 'S'

ft = glm(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked, family = "binomial", data
    )
summary(ft)

ft = glm(Survived ~ Pclass + Sex + Age.13 + SibSp + Parch.0 + Fare + Embarked.S, family = "binomial
    ", data)
summary(ft)

ft = glm(Survived ~ Pclass + Sex + Age.13 + SibSp + Parch.0 + Fare, family = "binomial", data)
summary(ft)

ft = glm(Survived ~ Pclass + Sex + Age.13 + SibSp + Parch.0, family = "binomial", data)
summary(ft)

ft = glm(Survived ~ Pclass + Sex + Age.13 + SibSp, family = "binomial", data)
summary(ft)

ft = glm(Survived ~ Pclass + Sex + Age + SibSp, family = "binomial", data)
summary(ft)

ft = glm(Survived ~ Pclass + Sex + (Age<9) + SibSp, family = "binomial", data)
summary(ft)

ft = glm(Survived ~ Pclass + Sex + cut(Age, c(-1,9,36,100)) + SibSp, family = "binomial", data)
summary(ft)

data$Age.cat = cut(data$Age, c(-1,9,36,100))
data$Parch.0 = data$Parch == 0
data$Embarked.S = data$Embarked == 'S'

train$Age.cat = cut(train$Age, c(-1,9,36,100))
train$Parch.0 = train$Parch == 0
train$Embarked.S = train$Embarked == 'S'

test$Age.cat = cut(test$Age, c(-1,9,36,100))
test$Parch.0 = test$Parch == 0
test$Embarked.S = test$Embarked == 'S'



### Model not using Age

ft = glm(Survived ~ Pclass + Sex + SibSp + Parch.0 + Fare + Embarked.S, family = "binomial", data)
summary(ft)

ft = glm(Survived ~ Pclass + Sex + SibSp + Parch.0 + Embarked.S, family = "binomial", data)
summary(ft)

ft = glm(Survived ~ Pclass + Sex + SibSp + Embarked.S, family = "binomial", data)
summary(ft)

ft = glm(Survived ~ Pclass + Sex + SibSp, family = "binomial", data)
summary(ft)
```

Variable selection:

```
allvars = c('Survived', 'Pclass', 'Sex', 'Age.cat', 'SibSp', 'Parch', 'Fare', 'Embarked')
var.sets = sapply(1:7, function(n) combn(allvars[-1],n))
sum(sapply(var.sets, ncol))
# 127 possible models

mods = list()
```

```
for (n in 1:7) {
  for (set in 1:choose(7,n)) {
    mods[[length(mods) + 1]] = glm(Survived ~ ., family = "binomial", data = data[,c('Survived',
        var.sets[[n]][,set])])
  }
}

aic = sapply(mods, AIC)
fts = mods[which(aic < 650)]
dev = sapply(fts, deviance)

for (ft in fts) {
  print(summary(ft))
}
```

Choosing thresholds with confusion matrices:

```
prediction = function(ft, threshold) {
  as.numeric(sapply(ft$fitted.values, function(x) x > threshold))
}

confusion = function(ft, threshold) {
  pred = prediction(ft, threshold)
  obs = ft$y
  a = sum((1-pred)*(1-obs))
  b = sum(pred*(1-obs))
  c = sum((1-pred)*obs)
  d = sum(pred*obs)
  matrix(c(a,c,b,d), nrow = 2)
}

findthresh.ad = function(ft, thresholds) {
  sapply(thresholds, function(x) {
    mat = confusion(ft, x)
    return(sum(diag(mat)))
  })
}

findthresh.bc = function(ft, thresholds) {
  sapply(thresholds, function(x) {
    mat = confusion(ft, x)
    return(mat[2,1] + mat[1,2])
  })
}

thresholds = (0:1000)/1000

a = findthresh.ad(ft, thresholds)
b = findthresh.bc(ft, thresholds)

which(a == max(a))
which(b == min(b))
### Note that which(a == max(a)) equals which(b == min(b))


qplot(rep(thresholds,2), c(a,b), ylab = "AIC / BIC", xlab = "Index") +
  geom_vline(x= thresholds[median(which(a == max(a)))], col = "blue") +
  geom_hline(y = max(a), col = "red") + geom_hline(y = min(b), col = "red")

threshold = .52
```

Cross Validation:

```
allvars = c('Survived', 'Pclass', 'Sex', 'Age.cat', 'SibSp', 'Parch', 'Fare', 'Embarked')
data = train[!is.na(train$Age),allvars]  #Remove unused observations
n = nrow(data)
### n=714 is divisible by 7
k = 714/7
```

```
# Split data into test sets of size 102
testsets = list()
for (i in 0:6) {
  testsets[[i+1]] = 1:k + k*i
}

# Variables used in current model of choice
vars = c('Survived', 'Pclass', 'Sex', 'Age.cat', 'SibSp')

# Cross validation function.  Returns the sum of squared errors for each test set.
cross = function(vars, binary = FALSE) {
  models = list()

  for (i in 1:7) {
    models[[i]] = glm(Survived ~ ., family = "binomial", data = data[-testsets[[i]],vars])
  }

  if (binary == FALSE) {
    dev = sapply(1:7, function(i) {
      sum((predict.glm(models[[i]], data[testsets[[i]],], type = "response") - data$Survived[
          testsets[[i]]])^2)
    })

  } else {

    dev = sapply(1:7, function(i) {
      pred = predict.glm(models[[i]], data[testsets[[i]],], type = "response")
      thresholds = (25:75)/100
      a = findthresh.ad(models[[i]], thresholds)
      thresh = (mean(which(a == max(a))) + 24) /100
      return(sum(((pred >= thresh) - data$Survived[testsets[[i]]])^2))
    })

  }

  return(dev)
}

# Check total sum of squared error for chosen model and full model
dev.ft = cross(vars)
sum(dev.ft)
var(dev.ft)

dev.full = cross(allvars)
sum(dev.full)
var(dev.full)

# Create list of cross-validation scores for every possible model
dev = list()
for (n in 1:7) {
  for (set in 1:choose(7,n)) {
    dev[[length(dev) + 1]] = cross(c('Survived', var.sets[[n]][,set]))
  }
}

# List of variables used in each model (to see which score corresponds to which model)
varlist = list()
for (n in 1:7) {
  for (set in 1:choose(7,n)) {
    varlist[[length(varlist) + 1]] = var.sets[[n]][,set]
  }
}

# Get total rss of each model
rss = sapply(dev,sum)
variances = sapply(dev,var)


## Identify chosen model's index
which(rss == sum(dev.ft))

# Plot scores against model size
```

```
plots = list()
plots[[1]] = qplot(jitter(sapply(varlist,length)), rss, xlab = "Model Size", ylab = "Score (RSS)")
    +
  geom_hline(y = sum(dev.ft), col = "red")
plots[[2]] = qplot(jitter(sapply(varlist,length)), variances, xlab = "Model Size", ylab = "Score (
    variance)") +
  geom_hline(y = var(dev.ft), col = "blue")




############## BINARY CROSS-VALIDATION ##############

# Check total sum of squared error for chosen model and full model
dev.ft.bin = cross(vars, binary = TRUE)
sum(dev.ft.bin)
var(dev.ft.bin)

dev.full.bin = cross(allvars, binary = TRUE)
sum(dev.full.bin)
var(dev.full.bin)

# Create list of cross-validation scores for every possible model
dev.bin = list()
for (n in 1:7) {
  for (set in 1:choose(7,n)) {
    dev.bin[[length(dev.bin) + 1]] = cross(c('Survived', var.sets[[n]][,set]), binary = TRUE)
  }
}

# Get total rss of each model
rss.bin = sapply(dev.bin,sum)
variances.bin = sapply(dev.bin,var)


## Identify chosen model's index
which(rss.bin == sum(dev.ft.bin))
# 64

# Plot scores against model size
plots[[3]] = qplot(jitter(sapply(varlist,length)), rss.bin,
      xlab = "Model Size", ylab = "Score (RSS.bin)") +
  geom_hline(y = sum(dev.ft.bin), col = "red")
plots[[4]] = qplot(jitter(sapply(varlist,length)), variances.bin,
      xlab = "Model Size", ylab = "Score (variance.bin)") +
  geom_hline(y = var(dev.ft.bin), col = "blue")

multiplot(plotlist = plots, cols = 2)
```

Predictions:

```
library(mlbench)
library(ggplot2)
library(plyr)
library(reshape2)
library(grid)
library(lattice)
library(GGally)
library(rpart)

setwd("~/Desktop/Stat 151/Titanic")
test = read.csv('test.csv')
train = read.csv('train.csv')

allvars = c('Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked')

tit = rpart(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked,  method="class", data=
    train[!is.na(train$Age),])
tit2 = rpart(Survived ~ Pclass + Sex + SibSp + Parch + Fare + Embarked,  method="class", data=train
    )

test$Survived.Rpart[!is.na(test$Age)] = as.numeric(predict(tit, test[!is.na(test$Age),])[,2] > .5)
```

```
test$Survived.Rpart[is.na(test$Age)] = as.numeric(predict(tit2, test[is.na(test$Age),])[,2] > .5)

ft = glm(Survived ~ Pclass + Sex + cut(Age, c(-1,9,36,100)) + SibSp, family = "binomial", train[!is
    .na(train$Age),])
# ft = glm(Survived ~ Pclass + Sex + cut(Age, c(-1,9,36,100)) + SibSp + cut(Age, c(-1,9,36,100))*
    SibSp, family = "binomial", train[!is.na(train$Age),])
ft2 = glm(Survived ~ Pclass + Sex + SibSp, family = "binomial", train)

test$Surv.prob[!is.na(test$Age)] = predict.glm(ft, test[!is.na(test$Age),], type = "response")
test$Surv.prob[is.na(test$Age)] = predict.glm(ft2, test[is.na(test$Age),], type = "response")

test$Survived.glm[!is.na(test$Age)] = as.numeric(test$Surv.prob[!is.na(test$Age)] > .52)
# test$Survived.glm[!is.na(test$Age)] = as.numeric(test$Surv.prob[!is.na(test$Age)] > .5455)
test$Survived.glm[is.na(test$Age)] = as.numeric(test$Surv.prob[is.na(test$Age)] > .62)

test$Survived[test$Survived.Rpart == test$Survived.glm] = test$Survived.Rpart[test$Survived.Rpart
    == test$Survived.glm]

sum(test$Survived.Rpart != test$Survived.glm)
#34

bad = test[test$Survived.Rpart != test$Survived.glm,-c(1,3,8)]
probs = test$Surv.prob[test$Survived.Rpart != test$Survived.glm]

# Whether or not Age is NA does not affect whether the predictions agree.
# This is good!
sum(is.na(test$Age)) / nrow(test)
#0.2057416
sum(is.na(bad$Age)) / nrow(bad)
#0.2058824

numbad = list()
for (thresh1 in (480:560)/1000) {
  for (thresh2 in (580:660)/1000) {

    test$Survived.glm[!is.na(test$Age)] = as.numeric(test$Surv.prob[!is.na(test$Age)] > thresh1)
    test$Survived.glm[is.na(test$Age)] = as.numeric(test$Surv.prob[is.na(test$Age)] > thresh2)

    test$Survived[test$Survived.Rpart == test$Survived.glm] = test$Survived.Rpart[test$Survived.
        Rpart == test$Survived.glm]

    numbad[[length(numbad) + 1]] = c(sum(test$Survived.Rpart != test$Survived.glm), thresh1,
        thresh2)
  }
}

numbad = matrix(unlist(numbad), byrow = TRUE, ncol = 3)
#34 is the minimum number of disagreements achievable by adjusting thresholds



train$Surv.prob[!is.na(train$Age)] = predict.glm(ft, train[!is.na(train$Age),], type = "response")
train$Surv.prob[is.na(train$Age)] = predict.glm(ft2, train[is.na(train$Age),], type = "response")
train$Survived.glm[!is.na(train$Age)] = as.numeric(train$Surv.prob[!is.na(train$Age)] > .52)
train$Survived.glm[is.na(train$Age)] = as.numeric(train$Surv.prob[is.na(train$Age)] > .62)

data = train[,c('Survived', 'Survived.glm' ,'Surv.prob')]
data[data$Survived != data$Survived.glm,]

test$Survived[as.numeric(rownames(bad[!is.na(bad$Age),]))] = as.numeric(test$Surv.prob[as.numeric(
    rownames(bad[!is.na(bad$Age),]))] > .6)
test$Survived[as.numeric(rownames(bad[is.na(bad$Age),]))] = 0

bad[is.na(bad$Age),]
test$Survived[128] = 0
ft3 = glm(Survived ~ Pclass + Sex + SibSp + Fare, family = "binomial", train)

thresholds = (0:1000)/1000
a = findthresh.ad(ft3, thresholds)
b = findthresh.bc(ft3, thresholds)
which(a == max(a)) - 1
which(b == min(b)) - 1
```

```
thresh = .59

predict.glm(ft3, bad[is.na(bad$Age),], type = "response")
test$Survived[as.numeric(names(predict.glm(ft3, bad[is.na(bad$Age),], type = "response")))] =
  predict.glm(ft3, bad[is.na(bad$Age),], type = "response") > thresh




test$Survived[is.na(test$Survived) & abs(test$Surv.prob - .52) > .07] =
  test$Survived.glm[is.na(test$Survived) & abs(test$Surv.prob - .52) > .07]


bad = test[is.na(test$Survived),]

test$Survived[as.numeric(rownames(bad))] = 1


frame = test[,c('PassengerId', 'Survived')]
write.table(frame, "results.csv", row.names=FALSE, sep=",")

# test$Survived[as.numeric(rownames(bad)[which(bad$Surv.prob < .65)])] = 0
# bad = test[is.na(test$Survived),]
# test$Survived[as.numeric(rownames(bad)[which(bad$Surv.prob > .6)])] = 1
#
# frame = test[,c('PassengerId', 'Survived')]
# names(frame)[2] = 'Survived'
# write.table(frame, "results2.csv", row.names=FALSE, sep=",")
```