STAT 151A HW5 Solutions

Billy Fang

1

I wrote this rather hastily. This is by no means the best way to analyze this dataset and my modeling choices might not be great under further inspection.

I got a few questions about how one should deal with the missing `Age` variables. There are a lot of techniques to deal with missing data. The process of replacing missing data with certain values is called imputation. The simplest type of imputation one could do is mean imputation or median imputation, where you replace the missing value with the mean/median of that variable. Of course, this has drawbacks (see the linked Wikipedia page). There are other more advanced methods listed on the page, each with their own drawbacks. One method a student mentioned to me was nearest neighbor imputation: in this dataset, if you have a person whose age is missing, you would find the most similar people in the dataset (in terms of the other variables), and then use their ages to guess a value of the age-less person. I unfortunately do not know much about missing data and imputation, but I hope these keywords can help you search for further reading material if you are interested.

I also should mention that you should not pay attention to the people on Kaggle that get extremely high prediction accuracy. There are ways to artificially climb the leaderboard (Moritz Hardt is a newly hired professor in Berkeley EECS) since you are allowed to submit many predictions. You should also not make the assumption that a model with higher test accuracy is necessarily "better"; if you make many submissions, you are in some sense "overfitting to the test dataset."

Exploration

Let's explore the data a little bit.

```r
train <- read.csv("train.csv")
test <- read.csv("test.csv")
ntrain <- dim(train)[1]
ntrain
```

```
## [1] 891
```

```r
ntest <- dim(test)[1]
ntest
```

```
## [1] 418
```

```r
full <- rbind(train[,-2], test)
nfull <- dim(full)[1]

ncol <- dim(full)[2]
for (j in 1:ncol) {
  curr <- full[,j]
  message(colnames(full)[j], ": ", class(curr))
  if (class(curr) == "factor") {
    message("    number of levels: ", length(levels(curr)))
  }
}
```

```
## PassengerId: integer
## Pclass: integer
## Name: factor
##     number of levels: 1307
## Sex: factor
##     number of levels: 2
## Age: numeric
## SibSp: integer
## Parch: integer
## Ticket: factor
##     number of levels: 929
## Fare: numeric
## Cabin: factor
##     number of levels: 187
## Embarked: factor
##     number of levels: 4
```

Let us discuss some issues with the predictors.

- `Pclass` is an integer, but we might consider making it categorical.
- `PassengerId` is just a row number, so it will not be helpful as a predictor.
- The `Name` variable is unique for each datapoint, so it will not be a useful predictor.
- Some inspection reveals that there are 263 (out of 1309 datapoints) missing values. All but one of them are in the `Age` column; the other missing value is inthe `Fare` column. We need to decide how to handle this.

```
sum((rowSums(is.na(full)) > 0)) # rows having at least one missing value
```

```
## [1] 264
```

```
sum(is.na(full$Age)) # rows with missing Age
```

```
## [1] 263
```

```
which(is.na(full$Fare))
```

```
## [1] 1044
```

- For `SibSp` and `Parch` we see the counts are very high for the low values of `SibSp` and `Parch`. We might consider treating these as categorical variables later.

```
table(full$SibSp)
```

```
##
##   0   1   2   3   4   5   8
## 891 319  42  20  22   6   9
```

```
table(full$Parch)
```

```
##
##    0    1    2    3    4    5    6    9
## 1002  170  113    8    6    6    2    2
```

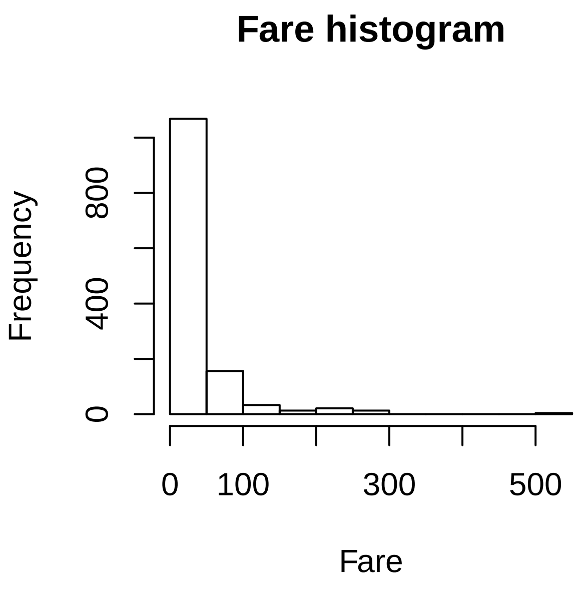
- As we see below, there are 681 ticket numbers for 1309 datapoints, so it might not be a great idea to use it. Each ticket number appears between 1 and 7 times, with the majority appearing 1 or 2 times.

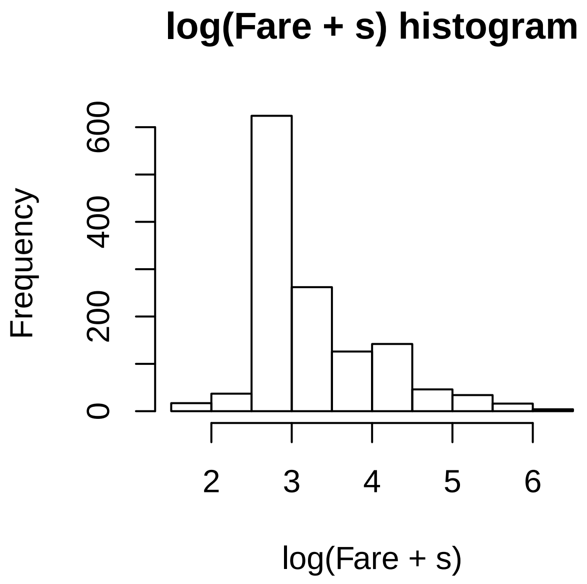
```
table(table(full$Ticket))
```

```
##
##   1   2   3   4   5   6   7   8  11
## 713 132  49  16   7   4   5   2   1
```

- The distribution of `Fare` values is also a bit skewed, and it might be better to use a log transformation.

```
hist(full$Fare, main="Fare histogram", xlab="Fare")
s <- 5 # arbitrarily chosen adjustment to deal with log(0)
hist(log(full$Fare + s), main="log(Fare + s) histogram", xlab="log(Fare + s)")
```





- By running `table(full$Cabin)` we can see that 1014 of the 1309 datapoints have no label for `Cabin`, and each other `Cabin` label has at most six datapoints. Thus `Cabin` might not be a useful predictor.
- `Embarked` (location of embarkment) does not seem to be relevant to survival status, so I will ignore it.

Cleaning, creating new variables

Let us modify the datasets accordingly. We consider `Pclass` as a categorical variable. We use mean imputation for the missing `Age` values, and median imputation for the missing `Fare` value. We also create a categorical variable for `SibSp` and `Parch` that groups all values ≥ 2 into one level.

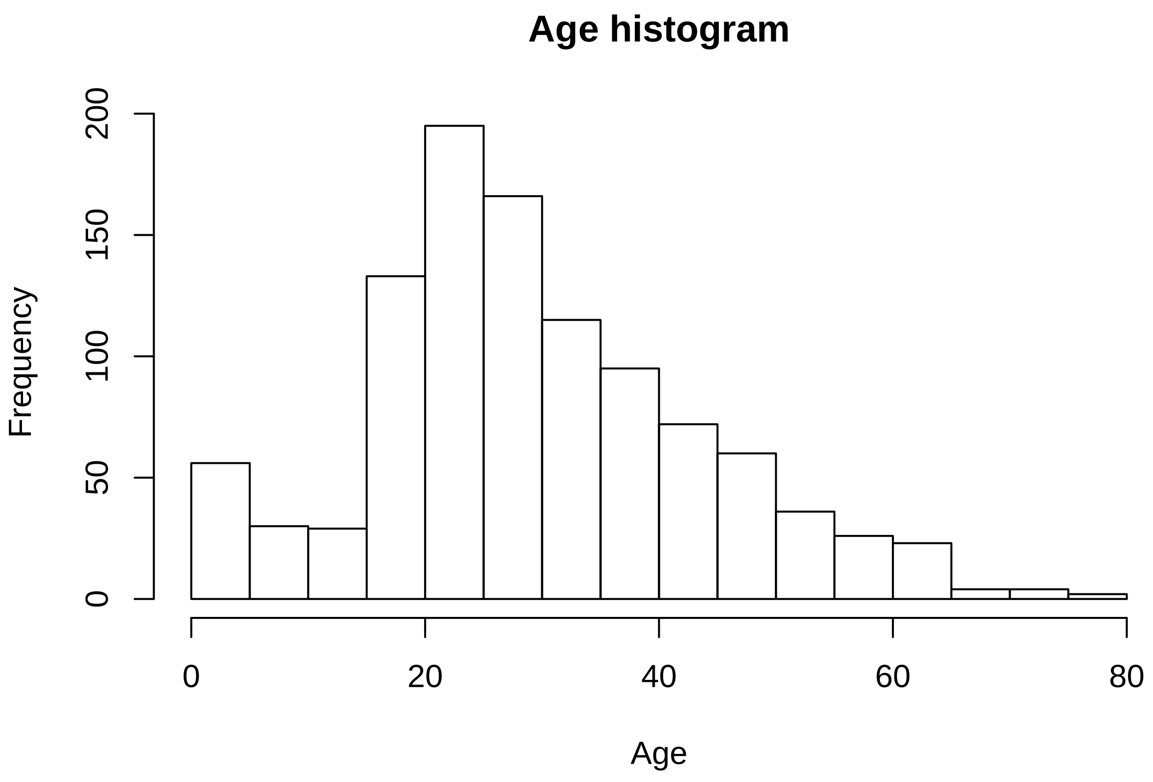
```
# Pclass
full$PclassCat <- as.factor(full$Pclass)
```

```
# SibSp and Parch
# version where >= 2 is grouped into one level
# https://stackoverflow.com/a/31679237/
full$SibSpCat <- factor(full$SibSp)
levels(full$SibSpCat) <- list("0"=0, "1"=1, ">=2"=c(2, 3, 4, 5, 8))
full$ParchCat <- factor(full$Parch)
levels(full$ParchCat) <- list("0"=0, "1"=1, ">=2"=c(2, 3, 4, 5, 6, 9))
```

```
# log Fare
# median imputation for missing value
```

```r
full$Fare[which(is.na(full$Fare))] <- median(full$Fare, na.rm=T)
full$logFare <- log(full$Fare + s)
```

```r
# use mean imputation to deal with missing Age
# (not necessarily the best way to handle this)
hist(full$Age, main="Age histogram", xlab="Age")
```



```r
mu <- mean(full$Age, na.rm=T)
full$Age2 <- full$Age
full$Age2[which(is.na(full$Age2))] <- mu
```

Let us split back into training and testing again.

```r
train <- data.frame(Survived=train$Survived, full[1:ntrain,])
test <- data.frame(full[-(1:ntrain),])
```

Models

Let us review what variables we have.

```r
colnames(train)
```

```
##  [1] "Survived"    "PassengerId" "Pclass"      "Name"        "Sex"
##  [6] "Age"         "SibSp"       "Parch"       "Ticket"      "Fare"
## [11] "Cabin"       "Embarked"    "PclassCat"   "SibSpCat"    "ParchCat"
## [16] "logFare"     "Age2"
```

I will consider a few models, and use 5-fold cross validation to decide between them. The models differ in how they treat `Age`, `SibSp`, `Parch` (as numerical or categorical) and `Fare` (with or without log transformation), as well as which variables are included.

```r
formulas <- list(
  "Survived ~ PclassCat + Sex + Age2 + SibSp + Parch + Fare",
  "Survived ~ PclassCat + Sex + Age2 + SibSpCat + ParchCat + Fare",
  "Survived ~ PclassCat + Sex + Age2 + SibSp + Parch + logFare",
  "Survived ~ PclassCat + Sex + Age2 + SibSpCat + ParchCat + logFare",
  "Survived ~ Pclass + Sex + Age2 + SibSp + Parch + Fare",
  "Survived ~ Pclass + Sex + Age2 + SibSpCat + ParchCat + Fare",
  "Survived ~ Pclass + Sex + Age2 + SibSp + Parch + logFare",
  "Survived ~ Pclass + Sex + Age2 + SibSpCat + ParchCat + logFare"
)
```

Cross-validation

We perform cross-validation to evaluate the performance of each of the models. We choose a threshold to minimize the error on the training data, before using it to predict the survival status on the fold.

```r
bestThres <- function(phat, y, thres.vec) {
  ### Function for choosing best threshold based on training data
  minerr <- Inf
  best <- -1
  for (i in 1:length(thres.vec)) {
    thres <- thres.vec[i]
    pred <- as.numeric(phat > thres)
    err <- sum(pred != y)
    if (err < minerr) {
      minerr <- err
      best <- thres
    }
  }
  return(best)
}
```

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
k <- 20
set.seed(0)
folds <- createFolds(train$Survived, k=k)
thres.vec <- seq(0, 1, by=0.05)
errs <- rep(0, length(formulas))
for (i in 1:length(folds)) {
  for (j in 1:length(formulas)) {
    mod <- glm(formula=formulas[[j]], family=binomial, data=train[-folds[[i]],])
    phat <- fitted(mod)

    # choose between one of the following two lines
    # thres <- 0.5
    thres <- bestThres(phat, train$Survived[-folds[[i]]], thres.vec)

    phatpred <- predict(mod, newdata=train[folds[[i]],], type="response")
    pred <- as.numeric(phatpred > thres)
    err <- sum(pred != train$Survived[folds[[i]]]) / length(folds[[i]])
    errs[j] <- errs[j] + err

  }
}
errs <- errs / length(folds)
errs
```

```
## [1] 0.1976515 0.1908081 0.1943182 0.1909091 0.1965404 0.1896717 0.1965909
## [8] 0.1998485
```

```
bestidx <- which.min(errs)
bestidx
```

```
## [1] 6
```

```
formulas[[bestidx]]
```

```
## [1] "Survived ~ Pclass + Sex + Age2 + SibSpCat + ParchCat + Fare"
```

This is the model that cross-validation selected. It uses class (as a numerical variable), gender, age (with mean imputation for missing values), sibling-spouse count (as a categorical variable), parent-child count (as a categorical variable), and logarithm of fare shifted by 5.

Prediction

We now fit this model on all the training data, and predict for the test data.

```
mod <- glm(formula=formulas[[bestidx]], data=train, family=binomial)
summary(mod)
```

```
##
## Call:
## glm(formula = formulas[[bestidx]], family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7248  -0.6312  -0.4321   0.6024   2.6838
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.804906   0.540108   8.896  < 2e-16 ***
## Pclass      -1.077924   0.138955  -7.757 8.67e-15 ***
## Sexmale     -2.714044   0.199378 -13.613  < 2e-16 ***
## Age2        -0.040093   0.007966  -5.033 4.82e-07 ***
## SibSpCat1    0.068130   0.221039   0.308  0.75791
## SibSpCat>=2 -1.286655   0.384921  -3.343  0.00083 ***
## ParchCat1    0.336575   0.285628   1.178  0.23865
## ParchCat>=2 -0.378336   0.318059  -1.190  0.23424
## Fare         0.002315   0.002283   1.014  0.31048
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.66  on 890  degrees of freedom
## Residual deviance:  785.13  on 882  degrees of freedom
## AIC: 803.13
##
## Number of Fisher Scoring iterations: 5
```

```
phat <- fitted(mod)

# choose between one of the following two lines
# thres <- 0.5
thres <- bestThres(phat, train$Survived, thres.vec)

phatpred <- predict(mod, test, type="response")
pred <- as.numeric(phatpred > thres)
sum(is.na(pred))
```

```
## [1] 0
```

```
pred.dat <- data.frame(PassengerId=test$PassengerId, Survived=pred)
write.csv(pred.dat, file="prediction5.csv", row.names=F)
```

Kaggle reports that the "public" prediction accuracy for this model is `0.7799`. [Note that this is the accuracy on a *subset* of the test data.]

2

The gradient of the log-likelihood is $\nabla\ell(\beta) = X^\top(Y - p)$ (see lecture notes), where $\log\frac{p_i}{1-p_i} := x_i^\top\beta$. The fitted probabilities $\widehat{p}$ therefore satisfy $X^\top(Y - \widehat{p}) = 0$.

3

Remark. Since the numbers in the output are rounded, the result of any computation that uses them may be slightly imprecise.

The `z value` column is $\frac{\widehat{\beta}_i}{\text{s.e.}(\widehat{\beta}_i)}$, i.e. the `Estimate` column divided by the `Std. Error` column. Thus the two missing standard errors are

```
0.6864 / 0.313
```

```
## [1] 2.192971
```

and

```
-0.9050 / (-4.349)
```

```
## [1] 0.2080938
```

respectively. Alternatively, you could take the square roots of the first two diagonal entries of the $(X^\top WX)^{-1}$ output; see below for more detail.

In the intercept-only model, there is a common probability $p_i = p$ for each observation, so one can check that the p that maximizes the log likelihood

$$\sum_{i=1}^{n}[y_i \log p + (1 - y_i)\log(1 - p)]$$

is $p = \overline{y}$, by setting the derivative [with respect to p] to zero, for example. Thus the null deviance is

$$-2n[\overline{y} \log \overline{y} + (1 - \overline{y}) \log(1 - \overline{y})].$$

In the statement of the problem we are given $\sum_{i=1}^{n} y_i = 79$ observations where the response variable is 1, so $\overline{y} = 79/212$ since we have $n = 212$ observations. Plugging this in yields

```r
n <- 212
ybar <- 79/212
- 2 * n * (ybar * log(ybar) + (1 - ybar) * log (1 - ybar))
```

```
## [1] 279.987
```

for the null deviance.

There are $n = 212$ observations, so there are $n - 1 = 211$ degrees of freedom in the null deviance.

Note that AIC can be computed by adding residual deviance with $2(3 + 1)$ since there are 3 explanatory variables. Thus, the residual deviance is

```r
222.18 - 2 * (3+1)
```

```
## [1] 214.18
```

There are 3 explanatory variables, there are $n - 3 - 1 = 208$ degrees of freedom in the residual deviance.

The missing off-diagonal entry in the $(X^\top WX)^{-1}$ output can be found by simply noting that this matrix is symmetric. So this missing entry is -0.255928180.

Because $(X^\top WX)^{-1}$ is the estimated asymptotic covariance of $\widehat{\beta}$, the square root of each diagonal entry yields the `Std. Error` column of the `glm` output. Thus the missing diagonal entry is

```r
0.3131^2
```

```
## [1] 0.09803161
```

We can check our answers with the real output.

```r
library(DAAG)
frogs.glm <- glm(pres.abs ~ log(distance) + log(NoOfPools) + meanmin,
                 family=binomial,
                 data=frogs)
summary(frogs.glm)
```

```
##
## Call:
## glm(formula = pres.abs ~ log(distance) + log(NoOfPools) + meanmin,
##     family = binomial, data = frogs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9642  -0.7657  -0.4619   0.8728   2.3219
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)      0.6864     2.1918   0.313 0.754146
## log(distance)   -0.9050     0.2081  -4.349 1.37e-05 ***
## log(NoOfPools)   0.5027     0.2004   2.509 0.012102 *
## meanmin          1.1153     0.3131   3.562 0.000369 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 279.99  on 211  degrees of freedom
## Residual deviance: 214.18  on 208  degrees of freedom
## AIC: 222.18
##
## Number of Fisher Scoring iterations: 5
```

```
X <- model.matrix(frogs.glm)
W <- diag(frogs.glm$fitted.values * (1 - frogs.glm$fitted.values))
solve(t(X) %*% W %*% X)
```

```
##                (Intercept) log(distance) log(NoOfPools)     meanmin
## (Intercept)      4.8038479  -0.363947754   -0.255928180 -0.49698440
## log(distance)   -0.3639478   0.043313307    0.008053415  0.01562971
## log(NoOfPools)  -0.2559282   0.008053415    0.040141698  0.02678507
## meanmin         -0.4969844   0.015629708    0.026785069  0.09805609
```

b

The new datapoint, after transformations according to our model, is $x_0 = (\log(265), \log(26), 3.5)$. Using $\widehat{\beta}$ from the `Estimate` column of the `glm` output, we can compute $\widehat{p}_0 = \frac{\exp(x_0^\top \widehat{\beta})}{1+\exp(x_0^\top \widehat{\beta})}$ to be

```
x0 <- c(256, 26, 3.5)
x0.tr <- c(1, log(x0[1]), log(x0[2]), x0[3])
betahat <- c(0.6864, -0.9050, 0.5027, 1.1153)
theta <- x0.tr %*% betahat
phat <- exp(theta) / (1 + exp(theta))
phat
```

```
##           [,1]
## [1,] 0.7701945
```

We can check our work with the real model. (The slightly different answer is likely due to the fact that the values for $\widehat{\beta}$ that we get from the `glm` output are rounded.)

```
newdata <- data.frame(cbind(distance=265, NoOfPools=26, meanmin=3.5))
predict(frogs.glm, newdata, type="response")
```

```
##         1
## 0.7645755
```

c

The residual deviance will decrease [or stay the same], for the same reason that RSS decreases in linear regresison under the analogous scenario. Specifically, let $(\widehat{\beta}_0, \ldots, \widehat{\beta}_3)$ be the coefficients obtained above for the original model. If we consider the larger model, using coefficients $(\widehat{\beta}_0, \ldots, \widehat{\beta}_3, 0)$ will produce the same model as before, with the same likelihood as before. But these coefficients (which may not be the MLE in the larger model) will have likelihood less than or equal to the maximized likelihood for this larger model. Thus the residual deviance for the larger model will be smaller than [or equal to] that of the original model.

The null deviance will not be affected because it only depends on the intercept-only model.

4

a

The log likelihood can be written as

$$\begin{aligned}\ell(\beta) &= \sum_{i=1}^{n}[y_i(x_i^\top \beta) - \log(1+\exp(x_i^\top \beta))] \\ &= Y^\top X\beta - \sum_{i=1}^{n}\log(1+\exp(x_i^\top \beta)).\end{aligned}$$

We see that Y enters the log likelihood only through $X^\top Y$.

Alternatively you can note that the gradient $X^\top(Y-p)$ only depends on Y through $X^\top Y$.

Alternatively you can note that in all the Newton-Rhapson / IWLS updates, Y only appears via $X^\top Y$.

b

By definition

$$\widehat{p}_i = \frac{\exp(\widehat{\beta}_0 + \widehat{\beta}_1 x_{i1} + \cdots + \widehat{\beta}_p x_{ip})}{1+\exp(\widehat{\beta}_0 + \widehat{\beta}_1 x_{i1} + \cdots + \widehat{\beta}_p x_{ip})} = \frac{\exp(x_i^\top \widehat{\beta})}{1+\exp(x_i^\top \widehat{\beta})} = \frac{1}{1+\exp(-x_i^\top \widehat{\beta})}.$$

c

From problem 2 and the fact that X contains the all ones vector $\mathbf{1}$, we have

$$\sum_{i=1}^{n} y_i = Y^\top \mathbf{1} = \widehat{p}^\top \mathbf{1} = \sum_{i=1}^{n}\widehat{p}_i.$$

Because the y_i are either 0 or 1, the left-hand side is the number of y_i that are equal to 1.

d

The log likelihood for some β is

$$\ell(\beta) = \sum_{i=1}^{n}[y_i \log p_i + (1-y_i)\log(1-p_i)]$$

where $p_i := \frac{\exp(x_i^\top \beta)}{1+\exp(x_i^\top \beta)}$, so the residual deviance (up to an additive term) is

$$-2\ell(\widehat{\beta}) = -2\sum_{i=1}^{n}[y_i \log \widehat{p}_i + (1-y_i)\log(1-\widehat{p}_i)]$$

where $\widehat{p}_i := \frac{\exp(x_i^\top \widehat{\beta})}{1+\exp(x_i^\top \widehat{\beta})}$.

5

The `z value` for the ith coefficient is $\frac{\widehat{\beta}_i}{\text{s.e.}(\widehat{\beta}_i)}$, i.e. the `Estimate` column divided by the `Std. Error` column. Thus the missing z-value is

```
4.11947 / 0.36342
```

```
## [1] 11.33529
```

and the missing coefficient is

```
12.345 * 0.028
```

```
## [1] 0.34566
```

We have $n = 4601$ emails and 1813 of them have response variable equal to 1, so $\overline{y} = 1813/4601$. Using the expression for null deviance computed in Problem 3, we have

```
n <- 4601
ybar <- 1813 / 4601
- 2 * n * (ybar * log(ybar) + (1 - ybar) * log(1 - ybar))
```

```
## [1] 6170.153
```

We have $n = 4601$ emails, so the degrees of freedom in the null deviance is $n - 1 = 4600$.

We have 6 explanatory variables, so the degrees of freedom in the residual deviance is $n - 6 - 1 = 4594$.

The AIC is the residual deviance plus $2 * (6 + 1)$ since we have 6 explanatory variables.

```
3245.1 + 2 * (6 + 1)
```

```
## [1] 3259.1
```

We can check our work with the real output.

```
data(spam7)
spam <- spam7
n <- dim(spam)[1]

s <- 1e-3

M1 <- glm(yesno ~ log(crl.tot) + log(dollar+s) + log(bang+s)
          +log(money+s) + log(n000+s) + log(make+s),
          family=binomial, data=spam)
summary(M1)
```

```
##
## Call:
## glm(formula = yesno ~ log(crl.tot) + log(dollar + s) + log(bang +
##     s) + log(money + s) + log(n000 + s) + log(make + s), family = binomial,
##     data = spam)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1657  -0.4367  -0.2863   0.3609   2.7152
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      4.11947    0.36342  11.335  < 2e-16 ***
## log(crl.tot)     0.30228    0.03693   8.185 2.71e-16 ***
## log(dollar + s)  0.32586    0.02365  13.777  < 2e-16 ***
## log(bang + s)    0.40984    0.01597  25.661  < 2e-16 ***
## log(money + s)   0.34563    0.02800  12.345  < 2e-16 ***
## log(n000 + s)    0.18947    0.02931   6.463 1.02e-10 ***
## log(make + s)   -0.11418    0.02206  -5.177 2.25e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6170.2  on 4600  degrees of freedom
## Residual deviance: 3245.1  on 4594  degrees of freedom
## AIC: 3259.1
##
## Number of Fisher Scoring iterations: 6
```

b

Our new datapoint (after transformation) is $x_0 = (\log(157+s), \log(0.868+s), \log(2.894+s), \log(s), \log(s), \log(s))$. Using $\widehat{\beta}$ from the Estimate column of the glm output, we can compute $\widehat{p}_0 = \frac{\exp(x_0^\top \widehat{\beta})}{1+\exp(x_0^\top \widehat{\beta})}$ to be

```r
s <- 0.001
x0 <- c(157, 0.868, 2.894, 0, 0, 0)
x0.tr <- c(1, log(x0+s))
betahat <- c(4.11947, 0.30228, 0.32586, 0.40984, 0.34566, 0.18947, -0.11418)
theta <- x0.tr %*% betahat
phat <- exp(theta) / (1 + exp(theta))
phat
```

```
##           [,1]
## [1,] 0.9581115
```

We can check our work with the real model. (The slightly different answer is likely due to the fact that the values for $\widehat{\beta}$ that we get from the glm output are rounded.)

```r
newdata <- data.frame(cbind(crl.tot=157, dollar=0.868,
                            bang=2.894, money=0, n000=0, make=0))
predict(M1, newdata, type="response")
```

```
##         1
## 0.9581203
```

c

In other words, M1 has a higher maximized likelihood than M2, so (purely on the basis of comparing residual deviance) M1 is preferable because it fits the data better.

Note that since both models have the same number of variables, M1 has a lower AIC than M2 as well, so in this particular example choosing the model with the smaller residual deviance is the same as choosing the model with the smaller AIC.