– Statistical Modeling – Prof Fulvia Pennoni

DAVIDE RONCHI - 903320

2024-04-24

Exercise 1

What might affect the chance of losing a customer? A big cellular phone company planned a study addressing this issue. Key variables collected in the sample data are the following:

- **churn**: 0 = no, 1 = yes
- **account.length**: how long has the account been active (in months)
- **voice.plan**: 0 = No, 1 = Yes
- **intl.plan**: 0 = No, 1 = yes
- **intl.mins**: minutes used during international calls
- **intl.charge**: total international charge (in dollars)
- **day.mins**: minutes used during the day
- **eve.mins**: minutes used during the evening
- **night.mins**: minutes used during the night

Data are in the file **phone.Rdata**.

1.1

Illustrate the data reporting and commenting on some descriptive plots.

Data loading and preliminary analysis

Loading the dataset from "*heating.RData*" file using the *load()* function.

```
load("phone.Rdata")
```

We use the functions:

- *class()* that returns what R considers as the *class* of the object;
- *typeof()* that returns what R considers as the *type* of the object.

```
class(phone)
```

```
## [1] "data.frame"
```

```
typeof(phone)
```

```
## [1] "list"
```

We can see that "*phone*" has a *data.frame* class and a *list* type.
In order to better investigate the types of the variables contained in the dataframe we can use the *str()*

function as follows:

```
str(phone)
```

```
## 'data.frame':    5000 obs. of  9 variables:
##  $ churn         : Factor w/ 2 levels "1","0": 2 2 2 2 2 2 2 2 2 2 ...
##  $ account.length: int  128 107 137 84 75 118 121 147 117 141 ...
##  $ voice.plan    : Factor w/ 2 levels "1","0": 1 1 2 2 2 2 1 2 2 1 ...
##  $ intl.plan     : Factor w/ 2 levels "1","0": 2 2 2 1 1 1 2 1 2 1 ...
##  $ intl.mins     : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
##  $ intl.charge   : num  2.7 3.7 3.29 1.78 2.73 1.7 2.03 1.92 2.35 3.02 ...
##  $ day.mins      : num  265 162 243 299 167 ...
##  $ eve.mins      : num  197.4 195.5 121.2 61.9 148.3 ...
##  $ night.mins    : num  245 254 163 197 187 ...
```

We can see that it contains 5000 observations (rows) and 9 variables (columns)

- **churn**: is a *fctr* (factor) type variable. *Factor* is a categorical variable that can assume both *numeric* and *string* value types. In this case we assume that data are *numeric*, since it has two the levels are "0" and "1". The levels of the variable can also be evaluated via *levels(phone$churn)*, that will return all the possible values assumed by this variable.

```
levels(phone$churn)
```

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

As expected the levels are "1", that corresponds to "yes" and "0" to "no". It means that if a customer is associated with a value of $churn = 1$ he is a churner and viceversa.

- **account.length**: is an *integer* type variable, meaning that its values are integer numbers. It represents the number of months since the account has been active.
- **voice.plan** and **intl.plan**: both of them are again *fctr* class variables. Also in this case the two possible levels are "0" and "1"

```
levels(phone$voice.plan)
```

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

```
levels(phone$intl.plan)
```

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

Before continuing, we change the order of these levels for all three factor variables to make them easier to interpret. This is done because the function we are going to use later to fit the model considers the first level as *failure*, and we want it to be *not-churn*. In this way, when the returned probability for a particular unit is near to 1, the unit will be classified as 1 -> *churn*; while when the returned probability is near to 0, the unit will be classified as 0 -> *not-churn*. In order to change the levels' order we simply reassign to each variable, the variable itself as a *factor*, but specifying the levels explicitly. For consistency, although it wouldn't be necessary, we'll do the same with the *voice.plan* and *intl.plan* variables:

```
phone$churn <- factor(phone$churn, levels = c(0,1))
phone$voice.plan <- factor(phone$voice.plan, levels = c(0,1))
phone$intl.plan <- factor(phone$intl.plan, levels = c(0,1))
levels(phone$churn)
```

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

```
levels(phone$voice.plan)
```

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

```
levels(phone$intl.plan)
```

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

As we can see now the levels are ordered as 0 and 1, instead of 1 and 0.

- **intl.mins**, **intl.charge**, **day.mins**, **eve.mins**, **night.mins**: all these remaining variables are of type *numeric*, meaning that their values can contain decimals.

Descriptive statistics

Now we can explore the dataframe via some descriptive statistics. To do so we use the function *skim_without_charts()* from the package *skimr* on the whole dataframe.

```
library(skimr)
skim_without_charts(phone)
```

Table 1: Data summary

Name	phone
Number of rows	5000
Number of columns	9
Column type frequency:	
factor	3
numeric	6
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
churn	0	1	FALSE	2	0: 4293, 1: 707
voice.plan	0	1	FALSE	2	0: 3677, 1: 1323
intl.plan	0	1	FALSE	2	0: 4527, 1: 473

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
account.length	0	1	100.26	39.69	1	73.00	100.00	127.00	243.0
intl.mins	0	1	10.26	2.76	0	8.50	10.30	12.00	20.0
intl.charge	0	1	2.77	0.75	0	2.30	2.78	3.24	5.4
day.mins	0	1	180.29	53.89	0	143.70	180.10	216.20	351.5
eve.mins	0	1	200.64	50.55	0	166.38	201.00	234.10	363.7
night.mins	0	1	200.39	50.53	0	166.90	200.40	234.70	395.0

As previously observed, we have 3 *factor* type variables and 6 *numeric*

Factor variables All of the factor variables have zero missing values and aren't ordered;

- **churn**: it's the response variable, it has 4293 occurrences for *not-churn* and 707 occurrences for *churn*
- **voice.plan**: indicates if the customer has a voice plan or not, we have 3677 customers *without voice plan* and 1323 *with voice plan*;
- **intl.plan**: indicates if the customer has an international plan or not, we have 4527 customers *without international plan* and 473 *with international plan*.

Numeric variables All of them have zero missing values.

account.length represents the number of months since the account has been active.

- The **mean** value of **account.length** is around 100.26 months, with a **standard deviation** of 39.7. It has an high variability, considering also that the **minimum** value (p_0) is **1** month and the **maximum** (p_{100}) is **243.0** months.;
- The **range** is given by $p_{100} - p_0 = 243.0 - 1 = 242.0$.
- The 1st **quartile** (p_{25}) is $p_{25} = 73.00$, which means that a quarter of the *account.length* values are below 73 months.
- The 3rd **quartile** (p_{75}) is $p_{75} = 127.00$, which means that a quarter of the *account.length* values are above 127 months.
- The **median** value is represented by p_{50} and has a value of $p_{50} = 100.00$ months, which is really similar to the **mean**. This suggest that data are symmetric.

intl.mins are the minutes used during international calls, while **intl.charge** is the associated charge. Both of them assume low values

- The **mean** value of **intl.mins** is around 10.26 minutes, with a **standard deviation** of 2.76. For **intl.charge** we have a mean value of 2.77$ a standard deviation of 0.75. For both of them we have low variability, in particular lower for *intl.charge* relatively to the mean value.
- For the minutes the **minimum** value (p_0) is **0** minutes and the **maximum** (p_{100}) is **20** minutes, hence the **range** is given by $p_{100} - p_0 = 20 - 0 = 20$ minutes. The charge have a smaller range of $p_{100} - p_0 = 5.4 - 0 = 5.4\$$.
- The 1st **quartile** (p_{25}) is $p_{25} = 8.5$, which means that a quarter of the customers spend less than 8.5 minutes in international calls.
- The 3rd **quartile** (p_{75}) is $p_{75} = 12.00$, which means that just a quarter of the customers spend more than 12.0 minutes in international calls.
- The **median** value is represented by p_{50} and has a value of $p_{50} = 10.30$ minutes, which is really similar to the **mean**. This suggest that data are symmetric.

day.mins, **eve.mins** and **night.mins** are minutes spent on calls respectively during the day, the evening and the night. All of them have high mean values, in particular, **night.mins** and **eve.mins** have really similar values of mean (200.64 and 200.39) and standard deviations (50.55 and 50.23). The same holds for the **minimum**(0 for both), the **median** (201.0 and 200.4) and the quartiles. The main difference is represented by the **maximum** value which is higher in **night.mins**, probably due to outliers. Both distributions seems to be symmetric. Regarding **day.mins** we have a little lower **mean** value of 180.29, but an higher variability, with a **standard deviation** of 53.9 and also an high range of variation (0-351.5). The skewness seems to be small, since the median and the mean are quite the same.
By looking at these distribution we can conclude that most of the customers spend more minutes on calls during the evening and the night rather than during the day. This difference although is not that high in mean.
We can also show these variables in relationship with the response: for example we can create boxplots for *day.mins*, *night.mins*, *eve.mins* and *intl.mins* by *churn* status, grouping *churners* and *non churners* customers and evaluate the distributions. To do so we

- First create a plot area of 2 rows and 2 columns with *par(mfrow=c(2,2))*, than we fill each area with a couple of boxplot of day, evening, night and international minutes grouped by churn status. This is achieved by, e.g. day.mins ~ churn, in order to obtain two boxplots for "churn status = 0" (not-churn) and "churn status = 1" (churn);

- Specify the variable from which data should be taken via *data = phone*;
- Assign color to each boxplot using *col = c("lightblue", "lightsalmon")*, in this way we will have a light blue boxplot for churn = 0 a light salmon boxplot for churn = 1;
- Set x and y axis labels using *xlab =* and *ylab =*
- Set a title for each plot area using *main =*
- Set *horizontal* = TRUE in order to plot them horizontally.

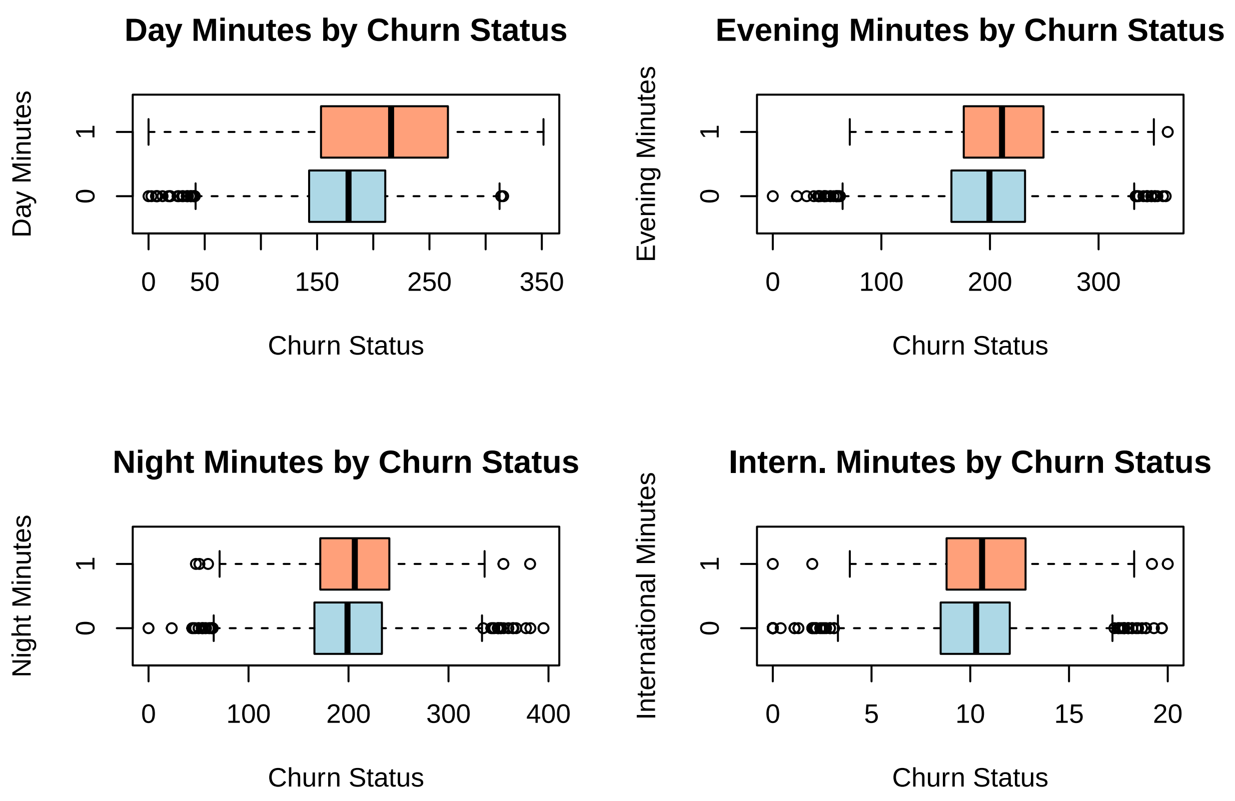
```r
# Set up the plotting area
par(mfrow = c(2,2))

# Boxplot of day minutes by churn status
boxplot(day.mins ~ churn, data = phone, col = c("lightblue", "lightsalmon"),
        xlab = "Churn Status", ylab = "Day Minutes",
        main = "Day Minutes by Churn Status", horizontal = TRUE)

# Boxplot of evening minutes by churn status
boxplot(eve.mins ~ churn, data = phone, col = c("lightblue", "lightsalmon"),
        xlab = "Churn Status", ylab = "Evening Minutes",
        main = "Evening Minutes by Churn Status", horizontal = TRUE)

# Boxplot of night minutes by churn status
boxplot(night.mins ~ churn, data = phone, col = c("lightblue", "lightsalmon"),
        xlab = "Churn Status", ylab = "Night Minutes",
        main = "Night Minutes by Churn Status", horizontal = TRUE)

# Boxplot of international minutes by churn status
boxplot(intl.mins ~ churn, data = phone, col = c("lightblue", "lightsalmon"),
        xlab = "Churn Status", ylab = "International Minutes",
        main = "Intern. Minutes by Churn Status", horizontal = TRUE)
```



As we can for all of them we have a distribution more shifted to high values when *churn = 1*. This is true in particular for *day.mins*, where churners distribution is more spread out and have a visibly higher median value. We can conclude that customers that are more active during the day and spend more minutes in daytime calls are more prone to churn. This is true also for night and evening calls, but less evident. We also notice that data of non churners seems to be more concentrated around the median, but present also more extreme values.

1.2

Show the relative frequencies of the contingency tables referred to the bivariate association between voice.plan and the response variable churn. Compute the odds of churn for customers having and not having a voice plan and the corresponding odds ratio. Comment on the results.

The contingency table is useful to evaluate the joint frequencies for different classes simultaneously. It is computed as follows:

	Variable y				
Variable x	d_1	..	d_j	..	d_k
c_1	n_{11}	..	n_{1j}	..	n_{1k}
..	..		..		..
c_i	n_{i1}	..	n_{ij}	..	n_{jk}
..	..		..		..
c_h	n_{h1}	..	n_{hj}	..	n_{hk}

The frequency n_{ij} is the number of statistical units that have simultaneously the levels c_i and d_j.
To compute the contingency table we can do as follows: We can use the function *table*, providing the 2 columns of interest of the *phone* dataframe: *phone$voice.plan* and *phone$churn*. Then we specify the names to be given to the dimensions in the result for a better interpretability by using *colnames()* and *rownames()* functions to assign the names respectively to the columns and to the rows

```
contingency_tab <- table(phone$voice.plan, phone$churn)
colnames(contingency_tab) <- c("Churners", "Non churners")
rownames(contingency_tab) <- c("Voice Plan", "No Voice Plan")
contingency_tab
```

```
##
##                  Churners Non churners
##   Voice Plan         3072          605
##   No Voice Plan      1221          102
```

We can see that for customers with a voice plan we have 1221 churners and 102 non-churners. While for customers without a voice plan we have 605 churners and 3072 non-churners.
To better interpret these results we could calculate the relative proportions. The contingency table with the relative frequency is obtained by dividing each term of the previous one with n, i.e. the total number of observations as follows:

	Variable y				
Variable x	d_1	..	d_j	..	d_k
c_1	f_{11}	..	f_{1j}	..	f_{1k}
..	..		..		..
c_i	f_{i1}	..	f_{ij}	..	f_{jk}
..	..		..		..
c_h	f_{h1}	..	f_{hj}	..	f_{hk}

The relative frequency $f_{ij} = n_{ij}/n$ is then a fraction of the observations that have simultaneously the levels c_i and d_j We use the function *prop.table()* to calculate the proportion of each element of the contingency table, providing as input the contingency table itself. The proportions are calculated as follows

```
relative_frequencies <- prop.table(contingency_tab)
relative_frequencies
```

```
##
##                  Churners Non churners
##   Voice Plan       0.6144       0.1210
##   No Voice Plan    0.2442       0.0204
```

By looking at the table with the proportions we can see that customers without a voice plan have a higher churn rate (12.10%) than the one of the customers with a voice plan (2.04%). This suggests that customers without a voice plan tends to be more likely to churn with respect to customers with a voice plan. Having a voice plan could be an important factor to avoid customers churning.

To have another measure of the association between *churn* and *voice.plan* we can compute the **Odds** and the **Odds Ratio** (**OR**).We start from the first one, which, in probability, is defined as follows: Odds of an event is the likelihood that an event will occur, expressed as a proportion of the likelihood that the event will not occur.

$$O = \frac{P(Y_i = 1)}{P(Y_i = 0)} = \frac{p}{1-p}$$

It is the ratio between the probability of success and failure. In our case can be computed by looking at the joint frequencies in the contingency table. Odds of churn for customers having a voice plan:

$$O_{\text{v.p.}} = \frac{\text{Proportion of Churners with Voice Plan}}{\text{Proportion of Non-churners with Voice Plan}}$$

We can calculate it by directly accessing the values in the contingency table (or the proportion table) as follows:

```
odds_voice_plan <- relative_frequencies["Voice Plan", "Churners"] /
                   relative_frequencies["Voice Plan", "Non churners"]
odds_voice_plan
```

```
## [1] 5.077686
```

The Odds of churn for customers not having a voice plan is instead calculated as follows:

$$O_{\text{no v.p.}} = \frac{\text{Proportion of Churners without Voice Plan}}{\text{Proportion of Non-churners without Voice Plan}}$$

```
odds_no_voice_plan <- relative_frequencies["No Voice Plan", "Churners"] /
                      relative_frequencies["No Voice Plan", "Non churners"]
odds_no_voice_plan
```

```
## [1] 11.97059
```

By observing the result we can conclude that a customer without a voice plan is more likely to churn with respect to one that has a voice plan. This confirm our first observation on the contingency tables. Despite that it's difficult to compare quantitatively these two values; to make it easier we can compute the **Odds Ratio**, which is defined as the ratio of the odds of an event occurring in one group to the odds of it occurring in another group:

$$\text{OR} = \frac{O_1}{O_2} = \frac{p_1/(1-p_1)}{p_2/(1-p_2)}$$

In our case it becomes:

$$\text{OR} = \frac{\text{Odds of Churn with Voice Plan}}{\text{Odds of Churn without Voice Plan}}$$

The OR is always between $(0, +\infty)$ and indicates how much more likely is the event (churn) to occur in the first group than it is in the second. When the OR is greater than 1 it indicates that the event (churn) is more likely to occur in the first group (voice plan) than it is in the second (no voice plan). Otherwise, when it is between $(0, 1)$ it means that the event (churn) is less likely to occur in the first group (voice plan) than it is in the second (no voice plan). To compute it we just need to compute the ratio between the two previously calculated odds:

```
odds_ratio <- odds_voice_plan / odds_no_voice_plan
odds_ratio
```

```
## [1] 0.4241802
```

We see that the Odds Ratio OR $\approx$ 0.424 which is ≤ 1. It means that the quota of customers having a voice plan and churning is 0.424 times the quota of the ones not having a voice plan, reinforcing our initial hypothesis.

1.3

Fit a logistic regression model first performing model selection. Comment on the results obtained according to the Akaike information criterion.

In order to fit a **logistic regression model** we need to use the *glm()* function, providing as input

- The formula, which will contain all the variables in the df, with *churn* $\sim$.;
- The data with $data = phone$;
- A description of the error distribution and link function to be used in the model with $family = binomial$.

```r
mod1 <- glm(churn ~ ., data = phone, family = binomial)
```

The estimated model above is given by the following formula:

$$\begin{aligned} logit(\hat{p}_i) = \ln\left(\frac{p_i}{1-p_i}\right) &= \hat{\beta}_0 + \hat{\beta}_1 \cdot \text{account.length} + \hat{\beta}_2 \cdot \text{voice.plan} + \hat{\beta}_3 \cdot \text{intl.plan} + \hat{\beta}_4 \cdot \text{intl.mins} + \\ &+ \hat{\beta}_5 \cdot \text{intl.charge} + \hat{\beta}_6 \cdot \text{day.mins} + \hat{\beta}_7 \cdot \text{eve.mins} + \hat{\beta}_8 \cdot \text{night.mins} \end{aligned}$$

In this formula $\hat{p}_i = P(Y_i = 1)$ is the estimated probability of success, where the success is represented by the $churn = yes$ event. It is important to notice that the model is multiplicative, so each x has a multiplicative impact of e^{β} on the result.
Since the model contains many variables we can perform a variable selection, in order to select the best predictor to explain the variability of the response *churn*. To do that we use the function *step()*, providing as input the previously fitted model. This function perform a *stepwise selection*; at each step, evaluates the removal of one predictor variable based on the **Akaike Information Criterion (AIC)**.

```r
options(digits = 7)
step(mod1)
```

```
## Start:  AIC=3456.86
## churn ~ account.length + voice.plan + intl.plan + intl.mins +
##     intl.charge + day.mins + eve.mins + night.mins
##
##                  Df Deviance    AIC
## - intl.charge     1   3438.9 3454.9
## - intl.mins       1   3438.9 3454.9
## <none>                3438.9 3456.9
## - account.length  1   3440.9 3456.9
## - night.mins      1   3455.8 3471.8
## - eve.mins        1   3487.5 3503.5
## - voice.plan      1   3523.9 3539.9
## - day.mins        1   3669.8 3685.8
## - intl.plan       1   3695.7 3711.7
##
## Step:  AIC=3454.87
## churn ~ account.length + voice.plan + intl.plan + intl.mins +
##     day.mins + eve.mins + night.mins
##
##                  Df Deviance    AIC
## <none>                3438.9 3454.9
## - account.length  1   3441.0 3455.0
## - night.mins      1   3455.8 3469.8
## - intl.mins       1   3460.4 3474.4
## - eve.mins        1   3487.5 3501.5
## - voice.plan      1   3524.0 3538.0
## - day.mins        1   3669.8 3683.8
## - intl.plan       1   3695.9 3709.9
```

```
##
## Call:  glm(formula = churn ~ account.length + voice.plan + intl.plan +
##     intl.mins + day.mins + eve.mins + night.mins, family = binomial,
##     data = phone)
##
## Coefficients:
##    (Intercept)  account.length     voice.plan1      intl.plan1       intl.mins
##      -7.197652        0.001597       -1.026300        1.888075        0.074602
##       day.mins        eve.mins      night.mins
##       0.012515        0.006120        0.003557
##
## Degrees of Freedom: 4999 Total (i.e. Null);  4992 Residual
## Null Deviance:       4075
## Residual Deviance: 3439  AIC: 3455
```

From the output of the *step()* function we can observe what follows:

- **Full Model**: The full model contains all the predictors according to the previously presented formula. The AIC of the starting model is 3456.86 By looking at the deviance table we can see that:
  - The column **Df** (Degrees of freedom) represents the degrees of freedom associated to each variable. It represent the reduction in the number of parameters when the corresponding predictor is eliminated from the model and, as expected, is 1 for everyone of them.
  - The **Deviance** is a measure of goodness of fit: the smaller the deviance, the better the fit. The values at each row represents the deviance after the removal of each predictor from the model.
  - The last columns contains the values of **AIC** after the removal of each predictor from the full model. The AIC is a trade-off measure between model fit and complexity. As stated before, the lower is the AIC, the better it is. For this reason we are interested in the rows that have a lower AIC than the full model, since variables with a negative change (reduction) in AIC when removed from the full model contributes positively in the goodness of the fit and viceversa.

As we can see the variables *intl.charge* and *intl.mins* both have lower **deviance** (3438.9) and **AIC** (3454.9) than the full model (which corresponds to the voice). It means that by removing them from the full model will lead to a better fit of the data.

- **Step 2** Is performed by removing the variable *intl.charge* from the full model. As we can see the **AIC** lowered from the previous step to a value of AIC = 3454.87, therefore the model benefited from the removal of the variable *intl.Charge*. By looking at the deviance table, however, no other variables now have a lower AIC than the current model, meaning that is not profitable removing any of them, and the current one represent the best trade-of between fit and complexity.

The selected model is therefore given by:

$$logit(\hat{p}_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \hat{\beta}_0 + \hat{\beta}_1 \cdot \text{account.length} + \hat{\beta}_2 \cdot \text{voice.plan} + \hat{\beta}_3 \cdot \text{intl.plan} + \hat{\beta}_4 \cdot \text{intl.mins} + \\ + \hat{\beta}_5 \cdot \text{day.mins} + \hat{\beta}_6 \cdot \text{eve.mins} + \hat{\beta}_7 \cdot \text{night.mins}$$

Provided by the formula *glm(formula = churn ~ account.length + voice.plan + intl.plan + intl.mins + day.mins + eve.mins + night.mins, family = binomial, data = phone)*, with a *Residual Deviance* of 3439 and an *AIC* of 3455.

1.4

Fit the model selected in the previous step and comment on the summary of the model. Interpret the estimated regression coefficients in exponential terms.

After making the model selection we can proceed with the fitting of the chosen model and analyze the result: To do that we use the formula provided by the output of the step function:

```
mod2 <- glm(formula = churn ~ account.length + voice.plan + intl.plan + intl.mins +
              day.mins + eve.mins + night.mins, family = binomial, data = phone)
```

Then we can show a the results using the function *summary()*.

```
summary(mod2)
```

```
##
## Call:
## glm(formula = churn ~ account.length + voice.plan + intl.plan +
##     intl.mins + day.mins + eve.mins + night.mins, family = binomial,
##     data = phone)
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -7.1976517  0.3955861 -18.195  < 2e-16 ***
## account.length  0.0015968  0.0011060   1.444    0.149
## voice.plan1    -1.0263004  0.1205146  -8.516  < 2e-16 ***
## intl.plan1      1.8880753  0.1141062  16.547  < 2e-16 ***
## intl.mins       0.0746023  0.0161956   4.606 4.10e-06 ***
## day.mins        0.0125149  0.0008623  14.514  < 2e-16 ***
## eve.mins        0.0061195  0.0008861   6.906 4.98e-12 ***
## night.mins      0.0035568  0.0008674   4.101 4.12e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4075.0  on 4999  degrees of freedom
## Residual deviance: 3438.9  on 4992  degrees of freedom
## AIC: 3454.9
##
## Number of Fisher Scoring iterations: 5
```

The summary returns the estimates, but we are more interested in commenting on the probability associated to each of them, which are given by:

$$P(Y_i = 1|x_i) = p_i = \frac{\exp x_i'\beta}{1 + \exp x_i'\beta}$$

They represents the multiplicative change in the odds of the response variable (in this case, churn) associated with a one-unit increase in the predictor variable corresponding to that coefficient, holding all other predictor variables constant.
We can calculate them as follows:

- First we extract the coefficients of the model and assign them to a new variable called *coefficients2*,
- We apply the *exp()* function to this variable and assign the result to a new one called *exp_coefficients2*, that will contain all the exponential of each estimated β
- Then we show the results.

```
coefficients2 <- coef(mod2)
exp_coefficients2 <- exp(coefficients2)
exp_coefficients2
```

```
##   (Intercept) account.length    voice.plan1     intl.plan1      intl.mins
##  0.0007483411   1.0015980582   0.3583301861   6.6066404020   1.0774555206
##      day.mins       eve.mins     night.mins
##  1.0125935283   1.0061382832   1.0035631584
```

We can observe the following:

- $e^{\hat{\beta}_0} = e^{-7.198} = 0.00075$ is the ratio between the odds in favor of success (churn) and failure (not churn) if all the explanatory variables in the model have a value of zero. In this case we can say that the odds of the response variable being in the "success" category (churning) are about 0.00075 times smaller than the odds of it being in the "failure" category (not churning). By looking at the summary instead we see that the *Wald* test produced a *p-value* $p < 0.001$ which means it's highly significant. The realized value of the test statistic is:

$$Z = \frac{\hat{\beta}_0}{Std.Error(\hat{\beta}_0)} = \frac{-7.198}{0.396} = -18.195$$

It means that the estimated coefficient for the intercept is around 18 std. deviations away from zero with a significance less than 0.001 (at any significance level), as indicated also by the '***'.

- $e^{\hat{\beta}_1} = e^{0.0016} = 1.0016$ means that for every one-unit increase in the predictor variable x_1 (for every additional month in *account.length*) the odds of the response variable (churn) increase by a factor of approximately 1.0016, holding all other predictor variables constant. The fact that it is greater than 1 suggest that there is a positive association between the predictor *account.length* and the likelihood of churn, but since the magnitude is really close to 1 the effect on it is minimal. The *Wald* test produced a *p-value* $p = 0.149$, which means that the estimated coefficient is not statistically significant. It means there is not enough evidence to refuse the hypothesis that the estimated coefficient $\hat{\beta}_1 = 0$. The Z value is:

$$Z = \frac{\hat{\beta}_1}{Std.Error(\hat{\beta}_1)} = \frac{0.0016}{0.0011} = 1.444$$

- $e^{\hat{\beta}_2} = e^{-1.026} = 0.358$ means that for the customers who have the predictor $x_2 = 1$ (which is for customers that have a voice plan), the odds of the response variable *churn* is 0.358 times the odds of the customers that have $x_2 = 0$ (that don't have a voice plan), while holding all the other variables constant. This indicates that having a voice plan is associated with a lower likelihood of churn among customers. - $e^{\hat{\beta}_3} = e^{1.888} = 6.607$ means that customers with $x_3 = 1$ (which is for customers that have a international plan), the odds of the response variable *churn* is 6.607 times higher the odds of the customers that have $x_3 = 0$ (that don't have an international plan), while holding all the other variables constant. This indicates that having an international plan is associated with a higher likelihood of churn among customers.

Both *voice.plan* and *intl.plan* are statistically significant with a *p-value* $p < 0.001$. At any significance level the two estimated coefficients are different from zero

- All the remaining have similar values: *intl.mins* $\rightarrow e^{\hat{\beta}_4} = 1.077$, *day.mins* $\rightarrow e^{\hat{\beta}_6} = 1.013$, *eve.mins* $\rightarrow e^{\hat{\beta}_7} = 1.006$, *night.mins* $\rightarrow e^{\hat{\beta}_8} = 1.003$. Since their values are similar they have similar interpretations. All of them have values > 1, it means that they are positively associated with the response, i.e., for each of them, an unit increase in the predictor is associated with an increased likelihood of churn, but since their values are really close to one, their effect will be minimal. All of them are also statistically significant at any significance level.
- The **null deviance** $= 4075$ is substantially higher than the **Residual deviance** $= 3439$, indicating that the estimated model explain a lot more variability than the one explained with a model with all the coefficients $\hat{\beta}_i = 0$.

1.5

Report the estimated probabilities of churn and comment on them.

The estimated probabilities of churn are in fact the fitted values of the model and can be can be calculated as follows:

$$P(Y_i = 1|x_i) = p_i = \frac{\exp x_i'\beta}{1 + \exp x_i'\beta}$$

We extract the *fitted.values* of the model and assign them to a new variable called *predicted_probabilities*. Then we show the first 6 elements of it

```r
predicted_probabilities <- mod2$fitted.values
head(predicted_probabilities)
```

```
##          1          2          3          4          5          6
## 0.13267681 0.05178815 0.15412664 0.53569503 0.31479268 0.55474650
```

These values are the predicted probabilities of churning for the first 6 customers in the dataframe. We can obtain the same result by using the function *predict()*. We provide as input the model and set *type= "response"* in order to return the probability of having *churn = 1*. Than we can create a new dataframe using the fu

```r
predicted_probabilities <- predict(mod2, type = "response")
head(predicted_probabilities)
```

```
##          1          2          3          4          5          6
## 0.13267681 0.05178815 0.15412664 0.53569503 0.31479268 0.55474650
```

We can now compare the predicted probabilities with the actual values in the *phone* dataframe. To do so we use the function *mutate()* from the *dplyr* library to create a new dataframe called *dt*. It will contain the original *phone* dataframe and a new column called *prob* that will contain the predicted probabilities we previously calculated using the model. Then we show the first 6 element of the columns *prob*, the predicted probabilities, and of *churn*, the actual class.

```r
library("dplyr")
```

```
##
## Caricamento pacchetto: 'dplyr'

## I seguenti oggetti sono mascherati da 'package:stats':
##
##     filter, lag

## I seguenti oggetti sono mascherati da 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
dt <- mutate(phone,
             prob = predicted_probabilities)
head(dt$prob)
```

```
##          1          2          3          4          5          6
## 0.13267681 0.05178815 0.15412664 0.53569503 0.31479268 0.55474650
```

```r
head(dt$churn)
```

```
## [1] 0 0 0 0 0 0
## Levels: 0 1
```

We see that the actual values are all zero and are represented by a categorical variable, while the predicted probabilities are continuous values, all between $(0, 1)$. We notice that the first, second, third and fifth have values below 0.5, while the other are above this threshold. This indicates that if we choose a threshold of 0.5 the first, second, third and fifth will be classified as *not-churn*, which corresponds to the actual value, while the remaining as *churn*, which would be wrong.

1.6

Produce the classification table, report the values of specificity and sensitivity and comment on the results.

In order to produce the **classification table** (**confusion matrix**) we first need to extract the *fitted values* from the model, that are probabilities calculated by the model for each unit of the dataset, according to the estimated coefficients. We already extracted them and put in a new dataframe called *dt*, so we can add a new column called *predout* that will contain the output of the prediction. It consists in the class obtained from the predicted probabilities by setting a threshold for the classification at 0.5. In order to classify the elements in the dataframe according to their predicted probability, we use the function *ifelse()*, that assign "0" if the predicted probability is < 0.5 and assigns "1" otherwise. After that we reassign the variable *predout* as a factor variable, explicitly specifying the levels, in this way we can compare it more easily with the actual class *churn*

```r
dt$predout <- ifelse(dt$prob < 0.5, 0,1)
dt$predout <- factor(dt$predout, levels = c("0", "1"))
tail(dt)
```

```
##      churn account.length voice.plan intl.plan intl.mins intl.charge day.mins
## 4995     0             75          0         0       6.9        1.86    170.7
## 4996     0             50          1         0       9.9        2.67    235.7
## 4997     1            152          0         0      14.7        3.97    184.2
## 4998     0             61          0         0      13.6        3.67    140.6
## 4999     0            109          0         0       8.5        2.30    188.8
## 5000     0             86          1         0       9.3        2.51    129.4
##      eve.mins night.mins       prob predout
## 4995    193.1      129.1 0.05808650       0
## 4996    223.0      297.5 0.11578324       0
## 4997    256.8      213.6 0.22761738       0
## 4998    172.8      212.4 0.07494208       0
## 4999    171.7      224.4 0.10176197       0
## 5000    267.1      154.8 0.02690210       0
```

We can see that the last 6 element of *dt* contains only one customer which actually is a churner, while all the others are non churners. The *predout* column instead contain all zeros, meaning that 5 out of 6 of these customers have been correctly classified.
We can have a better overview of the performance of the model by computing the confusion matrix. To do so we simply use the function *table()* to cross classify the elements in the column *predout*, i.e. the "Predictions", and the column *churn*, i.e. the "Ground Truth". After that we assign names to the dimensions of the table for a better interpretability.

```
conf_matrix <- table(dt$predout, dt$churn)
dimnames(conf_matrix) <- list(Predictions = row.names(conf_matrix),
                              Ground_Truth = colnames(conf_matrix))
conf_matrix
```

```
##            Ground_Truth
## Predictions    0    1
##           0 4219  605
##           1   74  102
```

On the main diagonal of the matrix we find the count of the occurrences that have been correctly predicted, outside of it we have the number of occurrences wrongly predicted. In particular we have:

- The **True Negatives** (**TN**) in position (0,0), which are values predicted as Negative ($predout = 0$) that actually are Negative ($churn = 0$), so they represent the correctly predicted as negative; $TN = 4219$.
- The **True Positives** (**TP**) in position (1,1), that are values predicted as Positive ($predout = 1$) that actually are Positive ($churn = 1$), so they represent the correctly predicted as positive; $TP = 102$.
- The **False Negatives** (**FN**) in position (0,1), that are values predicted as Negative ($predout = 0$) that actually are Positive ($churn = 1$), so they represent the wrongly predicted as negatives; $FN = 605$.
- The **False Positives** (**TP**) in position (1,0), that are values predicted as Positive ($predout = 1$) that actually are Negative ($churn = 0$), so they represent the wrongly predicted as positive; $FP = 74$.

Although we can already notice that most of the data is on the diagonal, it can be difficult to fully interpret these result without the appropriate metrics. Useful metrics are **Accuracy**, **Specificity** and **Sensitivity**. They are defined as follows

- **Accuracy** indicates the fraction of correctly predicted over the total number of occurrences, that is $\text{Accuracy} = \frac{\text{TP + TN}}{\text{TP+TP+FN+FP}}$.
- **Specificity** indicates how good the model is at identifying elements in the positive class, that is the probability that a customer is not a churner given that the model predicted him as a non-churner: $P(\hat{Y} = 0|y = 0)$. In our case is represented by$\text{Specificity} = \frac{\text{TN}}{\text{TN+FP}}$
- **Sensitivity** indicates how good the model is at identifying elements in the positive class, that is the probability that a customer is a churner given that the model predicted him as a churner Sensitivity of the test is the probability $P(\hat{Y} = 1|y = 1)$. In our case it is represented by $\text{Sensitivity} = \frac{\text{TP}}{\text{TP+FN}}$

To calculate these we can do as follows: first we assign the values of the confusion matrix to the respective variables and then we calculate the metrics.

```
TN <- conf_matrix[1, 1]
FP <- conf_matrix[2, 1]
FN <- conf_matrix[1, 2]
TP <- conf_matrix[2, 2]
```

```r
accuracy <- (TN + TP)/(TN+TP+FN+FP)
specificity <- TN / (TN + FP)
sensitivity <- TP / (TP + FN)

print(conf_matrix)
```

```
##            Ground_Truth
## Predictions    0    1
##           0 4219  605
##           1   74  102
```

```r
cat("Accuracy:", round(accuracy, 4),"\n")
```

```
## Accuracy: 0.8642
```

```r
cat("Specificity:", round(specificity, 4), "\n")
```

```
## Specificity: 0.9828
```

```r
cat("Sensitivity:", round(sensitivity, 4), "\n")
```

```
## Sensitivity: 0.1443
```

We see that

- The **Accuracy** value is 0.8642, meaning that overall 86.42% of the predictions are correct. This is informative but does not provide a complete picture, since the dataset is highly imbalanced: 4293 observations are truly non-churners, while only 707 are churners.
- **Specificity** has a really high value of 0.9828, meaning that 98.28% of the actual negative instances have been recognized as negative by the model, so the model is really good in detecting non-churners.
- **Sensitivity** has instead a low value or 0.1443, meaning that just 14.43% of the actual positive instances have been recognized as positive by the model, so it is not so good in detecting churners. This can be due to the fact that the class are unbalanced: we have many more observations of 'non-churners' than of 'churners'. This may have led the model to be better at recognizing the negative class (the over represented one).

1.7

Show the plot of the receiving operating curve (ROC), calculate the area under the curve (AUC) and comment on them.

Another way of evaluating the classification model is the **Receiving Operating Curve**, which plots the **True Positive Rate**, i.e. the *Sensitivity*, against the **False Positive Rate**, i.e. $1 - Specificity$.

$$TPR = \frac{TP}{TP + FN}$$
$$FPR = \frac{FP}{FP + TN}$$

The TPR is on the vertical axis, while the FPR is on the horizontal axis. To plot this curve we use the function *rocit()* from the *ROCit* package. This function creates a ROC Curve object and we assign it to a variable called *roc1*. We provide it two arguments: the *score*, which is the predicted probabilities by the model(dt$prob), and *class*, which is the actual classes (the ground truth). Then we plot the summary to evaluate the results.

```r
require(ROCit)
```

```
## Caricamento del pacchetto richiesto: ROCit
```

```r
roc1 <- rocit(score = dt$prob, class = dt$churn)
summary(roc1)
```

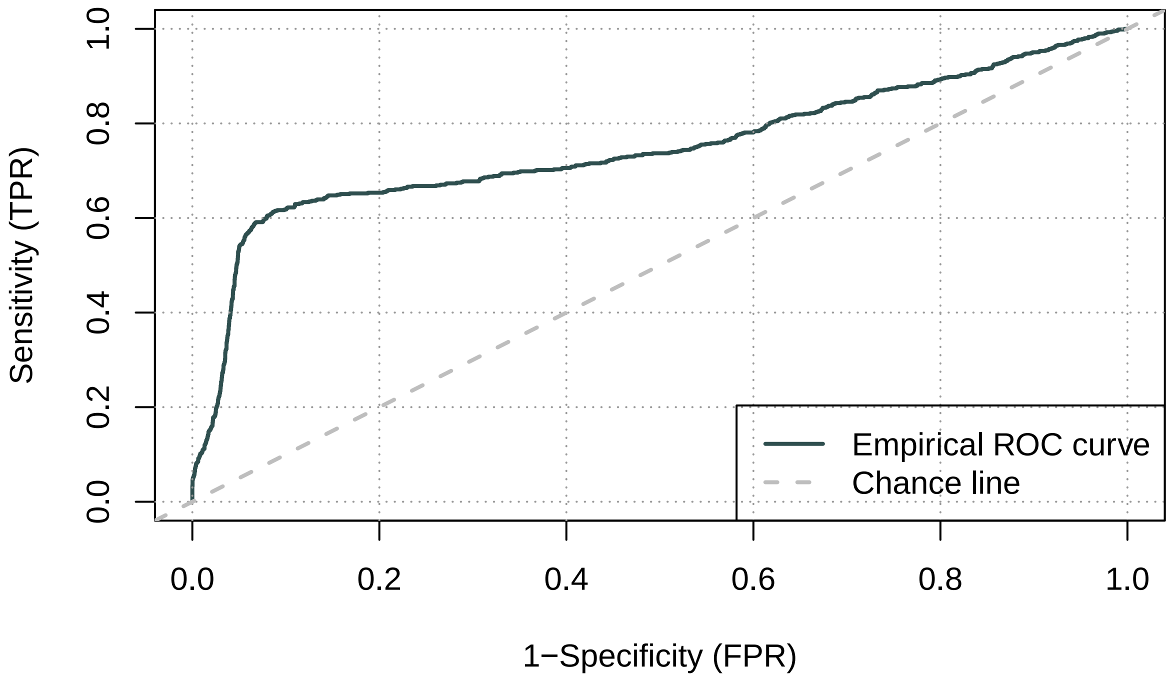
```
##
##  Method used: empirical
##  Number of positive(s): 707
##  Number of negative(s): 4293
##  Area under curve: 0.7482
```

The output returns:

- The method used for calculations, which is *empirical*, meaning that it used the actual data points.
- Number of positives = 707. This number represent the number of occurrences that have as actual class *churn=1*.
- Number of negatives = 4293. This number represent the number of occurrences that have as actual class *churn=0*.
- Area Under curve $AUC = 0.7482$. This is the area under the ROC curve and is a measure of the ability of the model to discriminate between positive and negative classes. It assumes values between 0 and 1 and an higher value corresponds to a better result. In this case the value 0.7482 suggest that the model have quite a good discrimination ability between positive and negative values. As a reference an $AUC = 0.5$ corresponds to the value of a random guessing model, while 1 to a perfect discrimination model.

To have a better picture of the measure we can obviously plot the ROC curve using the *plot()* function:

```r
plot(roc1,
     YIndex = FALSE)
```



In the plot we can see the ROC curve plotted with the continuous line and the "Chance line" as a dotted line, which represent the random guessing model. We can see that the ROC curve is above the Chance line, meaning that our model performs better than the random guessing. We also see that the ROC curve has a spike at the beginning, meaning that it has better performance for low values of FPR.

Exercise 2

Consider the **carData::Womenlf** data used to illustrate the logistic model in the teaching notes. Consider the following variables: - **partic**: a factor indicating labor force participation with levels: no work, part-time work, and full-time work; - **hincome**: husband's income in thousands of dollars; - **children**: a factor indicating the presence of children in the household.

2.1

Fit a multinomial logit model for the response variable partic, using hincome and children as covariates. Comment on the summary of the model.

First of all we load the data, contained in the library *carData* using the function *data()*.

```r
library(carData)
data("Womenlf")
```

In this way we have the data in a dataframe variable called *Womenlf*.
In order to fit a multinomial logit model we use the function *multinom()* from the library *nnet*. It takes as input the formula and the data. At first we estimate the model with *partic* as response and *hincome* and *children* as covariates.

First of all we look at the levels of the variable *partic* using the *levels()* function:

```r
levels(Womenlf$partic)
```

```
## [1] "fulltime" "not.work" "parttime"
```

We see that the levels are “fulltime”, “not.work” and “parttime”, in this order. Since the *multinom()* function takes as reference level the first one and computes the odds with respect to that, we prefer to set *not.work* as reference level. To do that we simply reassign to the original variable, the variable itself as a *factor*, but specifying the levels explicitly.

```r
Womenlf$partic <- factor(Womenlf$partic, levels = c("not.work", "parttime", "fulltime"))
levels(Womenlf$partic)
```

```
## [1] "not.work" "parttime" "fulltime"
```

Now we see that the levels have the desired order and we can fit the model

```r
library(nnet)
mmod1 <- multinom(partic ~ hincome + children, data = Womenlf)
```

```
## # weights:  12 (6 variable)
## initial  value 288.935032
## iter  10 value 211.454772
## final  value 211.440963
## converged
```

Since the optimization process of the objective function of the model converged, we can show the summary and comment on the results:

```r
summary(mmod1)
```

```
## Call:
## multinom(formula = partic ~ hincome + children, data = Womenlf)
##
## Coefficients:
##          (Intercept)      hincome childrenpresent
## parttime   -1.432321  0.006893838      0.02145558
## fulltime    1.982842 -0.097232073     -2.55860537
##
## Std. Errors:
##          (Intercept)    hincome childrenpresent
## parttime   0.5924627 0.02345484       0.4690352
## fulltime   0.4841789 0.02809599       0.3621999
##
## Residual Deviance: 422.8819
## AIC: 434.8819
```

We see that coefficients and corresponding standard errors are organized in a table, since we have a coefficient for each covariate and for each class other than the reference one in the response variable. The estimated coefficients represents the log-odds of the outcome variable associated with a one-unit change in the predictor variable, holding all other variables constant. Since these values can be difficult to interpret, we prefer to evaluate the associated probabilities. We can calculate them as follows:

- First we extract the coefficients of the model and assign them to a new variable called *cc1*,
- We apply the *exp()* function to this variable and assign the result to a new one called *exp_cc1*, that will contain all the exponential of each estimated β
- Then we show the results.

```
cc1 <- coef(mmod1)
exp_cc1 <- exp(cc1)
exp_cc1
```

```
##          (Intercept)   hincome childrenpresent
## parttime    0.238754 1.0069177      1.02168740
## fulltime    7.263353 0.9073454      0.07741263
```

-**(intercept)**: the odds ratio associated with the intercept for the class *parttime* is 0.2388. It means that the odds of being in the class *parttime* with respect to the one of being in the reference class, i.e. *not.work*, when all the predictors are zero is 0.2388. Similarly for the class *fulltime* we have that the odds of being in this class compared to the odds of being in the reference is 7.2633. - **hincome**: The values under the voice *hincome* represents the odds ratio associated with a one-unit increase in husband's income. For example, for the *parttime* category, the odds of being in the *parttime* category increase by a factor of approximately 1.0069 for every 1000$ increase in husband's income, holding other variables constant. The fact that it is greater than 1, suggest that there is a positive association between the predictor *hincome* and the likelihood of working part time, but since the magnitude is really close to 1 the effect on it is minimal. Similarly, for the *fulltime* class, the odds of being in the *fulltime* class decrease by a factor of approximately 0.9073 for every 1000$ increase in husband's income, holding other variables constant. In this case the value is less than 1, suggesting a negative association between *hincome* and the likelihood of working full time, meaning that when the husband income increase, the likelihood of working full time decrease by a factor of 0.9073. - **childrenpresent**: these represents the odds ratio associated with the presence of a child. Meaning that when children are present the likelihood of working *parttime* increases of a factor 1.0217, with respect to the one when children are absent. We see that the value is greater than one, despite it's really close to it, suggesting that there is a positive association between the presence of children and the likelihood of working partime. Similarly for the class *fulltime* we have that the odds of working fulltime decreases by a factor 0.0774 when the children are present with respect to when they are absent. So mothers have much less likelihood of working fulltime than non-mothers.

Overall we can conclude that both *hincome* and *children* are meaningful predictor, in particular for determining the likelihood of working fulltime.

2.2

Provide the table of predicted classifications cross-classified by the presence of children. Comment on the results.

Before calculating the predicted classifications table by the presence of children, we can have a first overview of how the model works by evaluating the probability of each record of being associated to a particular *partic* class.
To do so we first extract the **fitted values**, which represent the estimated probabilities of each unit in the dataframe according to the model. We extract them with the function *fitted()*, providing as input the model *mmod1*. Then we assign the result to a new variable called *pred_prob*.

```
pred_prob <- fitted(mmod1)
head(pred_prob)
```

```
##    not.work  parttime    fulltime
## 1 0.7136302 0.1930418 0.093327986
## 2 0.7014320 0.1871439 0.111424051
## 3 0.7464180 0.2483012 0.005280805
## 4 0.7429845 0.2123780 0.044637474
## 5 0.7316843 0.2034593 0.064856417
## 6 0.6490673 0.1661560 0.184776690
```

In each column we have, for each woman in the dataframe, the estimated probability of belonging to the correspondent class. After that we use the function *mutate()* from the *dplyr* library to create a new dataframe called *df*. It will contain the original *Womenlf* dataframe and three new columns called *prob_parttime*, *prob_fulltime*, *prob_notwork* that will contain the predicted probabilities we previously calculated (*pred_prob*). Then we show the first 6 element of the columns containing the predicted probabilities, and of *partic*, the actual class.

```
library("dplyr")
df <- mutate(Womenlf,
             prob_parttime = pred_prob[, "parttime"],
             prob_fulltime = pred_prob[, "fulltime"],
             prob_notwork = pred_prob[, "not.work"])
head(df[c("prob_parttime","prob_fulltime", "prob_notwork")])
```

```
##   prob_parttime prob_fulltime prob_notwork
## 1     0.1930418   0.093327986    0.7136302
## 2     0.1871439   0.111424051    0.7014320
## 3     0.2483012   0.005280805    0.7464180
## 4     0.2123780   0.044637474    0.7429845
## 5     0.2034593   0.064856417    0.7316843
## 6     0.1661560   0.184776690    0.6490673
```

```
head(df$partic)
```

```
## [1] not.work not.work not.work not.work not.work not.work
## Levels: not.work parttime fulltime
```

We see that all the first 6 elements in the dataframe have the highest value of probability associated to *not.work* and have as actual class *not.work*.
Now we can classify all the elements in *df* according to their predicted probability by using the function *predict()*. We provide as input the model *mmod1* and specify *type = "class"* such that the function return a classification instead of the probabilities associated with each class.

```
predicted <- predict(mmod1, type = "class")
df$predout <- predicted
head(df[c("partic", "prob_parttime","prob_fulltime", "prob_notwork", "predout")])
```

```
##     partic prob_parttime prob_fulltime prob_notwork  predout
## 1 not.work     0.1930418   0.093327986    0.7136302 not.work
## 2 not.work     0.1871439   0.111424051    0.7014320 not.work
## 3 not.work     0.2483012   0.005280805    0.7464180 not.work
## 4 not.work     0.2123780   0.044637474    0.7429845 not.work
## 5 not.work     0.2034593   0.064856417    0.7316843 not.work
## 6 not.work     0.1661560   0.184776690    0.6490673 not.work
```

We see that according to the probabilities calculated by the model the most probable class is *not.work* and they have been correctly classified. Now we can use these data to print the table of predicted classifications cross-classified by the presence of children. To do so we use the *table()* function, providing as rows input *predicted* which is the variable containing the predicted classes, and *Womenlf$children* that contains information if the woman has children or not.

```
cross_table <- table(predicted, Womenlf$children)
cross_table
```

```
##
## predicted  absent present
##   not.work     14     184
##   parttime      0       0
##   fulltime     65       0
```

- *not.work*: For observations predicted to be in the "not.work" class, there are 14 cases where children are absent and 184 cases where children are present. This indicates that the model predicts a significant number of women not working, in particular for mothers.
- *parttime*: We see that there are no units predicted to belong to this classes, for either the presence or absence of children. This suggests that maybe the model doesn't perform well in predicting this class. This could be due to the fact that the class is under represented with respect to others.
- *fulltime*: For observations predicted to be in the "fulltime" class, there are 65 cases where children are absent and no cases where children are present, meaning that the model predicts an important number of women working fulltime when children are absent, but it does not predict any women working fulltime when they have children.

2.3

Print the predicted probabilities for the woman in row 21 of the dataframe. What is the actual working position of the woman in question?

In order to print the predicted probabilities for the 21st woman in the dataframe we can just print the values we previously calculated in the *df* dataframe for the 21st row:

```
df[21, c("prob_notwork", "prob_parttime","prob_fulltime")]
```

```
##    prob_notwork prob_parttime prob_fulltime
## 21     0.413702     0.1125961     0.4737018
```

We see that the estimated probability of not working is 0.4137, for working part-time is 0.1126 and for working full-time is 0.4737. This suggest that maybe the woman have been classified as *fulltime*. We can varify it by showing the corresponding value in the column *predout*.

```
df[21, "predout"]
```

```
## [1] fulltime
## Levels: not.work parttime fulltime
```

The output in fact is "fulltime". If we want instead to verify the actual class for this woman we can show the corresponding value in the column *partic*.

```
df[21, "partic"]
```

```
## [1] parttime
## Levels: not.work parttime fulltime
```

We see that the actual class is *parttime*, not full time. This means that the woman has been wrongly classified by the model. We are not surprised, since no occurrence has been classified as *parttime* by the model.

2.4

Fix the value of hincome and children for a new out-of-sample unit and provide the predicted probabilities for this new unit. Comment on the results.

The function *predict()* can also be used for predicting probabilities and therefore the belonging class of new data. To do so we create a new unit containing specific values of *hincome* and *children*. In particular we choose an husband income of 20000$ and no children, so *children* = "absent". To do so we use the function *data.frame()* to create a dataframe containing two columns: *hincome* = 20 and *children* = "absent". In particular we pass the value "absent" as a factor, specifying the levels to be "absent" and "present".

```
new_unit <- data.frame(hincome = 20, children = factor("absent", levels = c("absent", "present")))
new_unit
```

```
##   hincome children
## 1      20   absent
```

After that we can calculate the probabilities for this unit using the function *predict()*. In this case we provide the model *mmod1*, the variable *new_unit* and we specify the *type = "probs"*, such that it returns the probabilities instead of classifying the unit

```
nu_pred_probs <- predict(mmod1, new_unit, type = "probs")
nu_pred_probs
```

```
##  not.work  parttime  fulltime
## 0.4323410 0.1184831 0.4491760
```

We see that the probabilities are 0.4323 for *not.work*, 0.1184 for *parttime* and 0.4492 for *fulltime*. This means that for a woman with no children and an husband income of 20000$, there is a greater probability of working full-time, despite it is reaaly similar to the one of not working at all. The probability of working partime instead is really low. In this case the unit would have been classified as a full-time worker and we can verify it by using the function *predict()*, specifying *type = "class"*:

```
predict(mmod1, new_unit, type = "class")
```

```
## [1] fulltime
## Levels: not.work parttime fulltime
```