Assignment 1 - BDBA

Loic Roldan Waals, 409763 & Iris Bominaar, 411495

March 25, 2018

9.1 Conceptual

Question 1

In a regression tree the target variable does not have classes, therefore a regression model is fit to the target variable using each of the independent variables.

Entropy is used to calculate the amount of disorder of a specific node, meaning calculating the presence of one class of the target variable compared to the other class. The purer the data, the higher the presence of one specific class. Since, for regression tree the target variable does not have clear classes, entropy cannot be calculated and thus not used as a measure when splitting nodes. Instead other measures such as the sum of squared errors or cross validation need to be used.

Question 2

Part (a)

The corresponding tree to the partition of the predictor space is shown in figure 1 below.

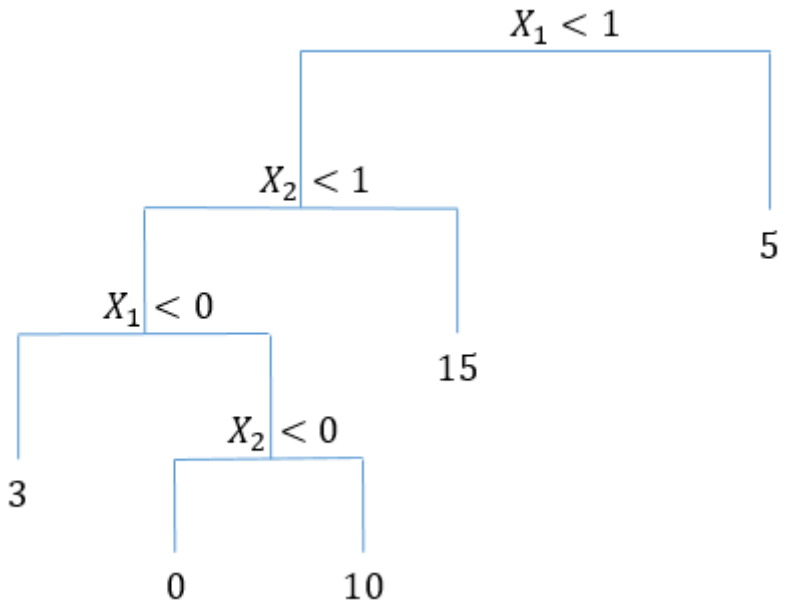


Figure 1: Decision Tree

Part (b)

The corresponding diagram to the right-hand panel is shown in figure 2 below.

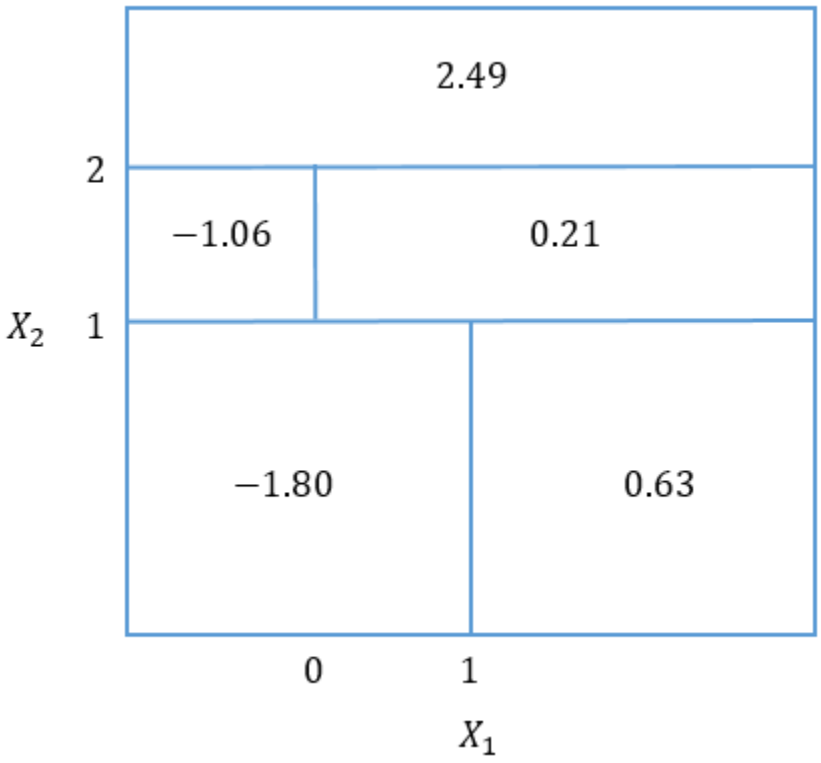


Figure 2: Diagram of predictor space

9.2 Applied

Question 3

Part (a)

First the data was downloaded from the website. We then import the file and change the date variables to actual dates so that R recognizes them. The data is quickly inspected with the *summary* function. This is performed with the following command:

```
setwd("C:/Users/Loic RW/Google Drive/Big Data and Business Analytics/Assignments/Assignment 1")

library(readr)
train_users_2 <- read_csv(file="train_users_2.csv",
    col_types = cols(date_account_created = col_date(format = "%Y-%m-%d"),
        date_first_booking = col_date(format = "%Y-%m-%d"),
        gender = col_character()))

summary(train_users_2$date_account_created)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.
## "2010-01-01" "2012-12-26" "2013-09-11" "2013-06-25" "2014-03-06"
##         Max.
## "2014-06-30"
```

```
summary(train_users_2$timestamp_first_active)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 2.009e+13 2.012e+13 2.013e+13 2.013e+13 2.014e+13 2.014e+13
```

```
summary(train_users_2$date_first_booking)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.
## "2010-01-02" "2012-12-02" "2013-09-11" "2013-07-04" "2014-04-04"
##         Max.         NA's
## "2015-06-29"     "124543"
```

```
summary(train_users_2$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1.00   28.00   34.00   49.67   43.00 2014.00   87990
```

```
summary(train_users_2$signup_flow)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   3.267   0.000  25.000
```

```
summary(as.factor(train_users_2$gender))
```

```
## -unknown-    FEMALE      MALE     OTHER
##     95688     63041     54440       282
```

```
summary(as.factor(train_users_2$signup_method))
```

```
##    basic facebook   google
##   152897    60008      546
```

```
summary(as.factor(train_users_2$language))
```

```
##     ca     cs     da     de     el     en     es     fi     fr     hr
##      5     32     58    732     24 206314    915     14   1172      2
##     hu     id     is     it     ja     ko     nl     no     pl     pt
##     18     22      5    514    225    747     97     30     54    240
##     ru     sv     th     tr     zh
##    389    122     24     64   1632
```

```
summary(as.factor(train_users_2$affiliate_channel))
```

```
##           api       content        direct         other   remarketing
##          8167          3948        137727          8961          1096
##     sem-brand sem-non-brand           seo
##         26045         18844          8663
```

```
summary(as.factor(train_users_2$affiliate_provider))
```

```
##               baidu                bing          craigslist
##                  29                2328                3471
##                daum              direct     email-marketing
##                   1              137426                 166
##            facebook facebook-open-graph              google
##                2273                 545               51693
##                 gsp              meetup               naver
##                 453                 347                  52
##               other           padmapper                vast
##               12549                 768                 829
##                wayn               yahoo              yandex
##                   8                 496                  17
```

```
summary(as.factor(train_users_2$first_affiliate_tracked))
```

```
##        linked     local ops     marketing           omg       product
##         46287            34           139         43982          1556
## tracked-other     untracked          NA's
##          6156        109232          6065
```

```
summary(as.factor(train_users_2$signup_app))
```

```
## Android     iOS   Moweb     Web
##    5454   19019    6261  182717
```

```
summary(as.factor(train_users_2$first_device_type))
```

```
##      Android Phone     Android Tablet    Desktop (Other)
##               2803               1292               1199
##               iPad             iPhone        Mac Desktop
##              14339              20759              89600
##      Other/Unknown SmartPhone (Other)    Windows Desktop
##              10667                 76              72716
```

```
summary(as.factor(train_users_2$first_browser))
```

```
##            -unknown-      Android Browser         AOL Explorer
##                27266                  851                  245
##           Apple Mail                Arora        Avant Browser
##                   36                    1                    4
##   BlackBerry Browser               Camino               Chrome
##                   53                    9                63845
```

```
##        Chrome Mobile              Chromium            CometBird
##                 1270                    73                   11
##        Comodo Dragon             Conkeror             CoolNovo
##                    2                     1                    6
##        Crazy Browser                 Epic              Firefox
##                    2                     1                33655
##                Flock         Google Earth            Googlebot
##                    2                     1                    1
##            IceDragon            IceWeasel                   IE
##                    1                    13                21068
##            IE Mobile                 Iron       Kindle Browser
##                   36                    17                    1
##              Maxthon       Mobile Firefox        Mobile Safari
##                   46                    30                19274
##              Mozilla          NetNewsWire              OmniWeb
##                    3                     1                    2
##               Opera           Opera Mini         Opera Mobile
##                  188                     4                    2
##         Outlook 2007            Pale Moon Palm Pre web browser
##                    1                    12                    1
##      PS Vita browser              RockMelt               Safari
##                    1                    24                45169
##            SeaMonkey                 Silk            SiteKiosk
##                   11                   124                   24
##          SlimBrowser       Sogou Explorer            Stainless
##                    2                    33                    1
##           TenFourFox     TheWorld Browser           wOSBrowser
##                    8                     2                    6
##       Yandex.Browser
##                   11
```

```
summary(as.factor(train_users_2$country_destination))
```

```
##     AU     CA     DE     ES     FR     GB     IT    NDF     NL  other
##    539   1428   1061   2249   5023   2324   2835 124543    762  10094
##     PT     US
##    217  62376
```

The variables with missing values include:

- *date_first_booking*
- *age*
- *first_affiliate_tracked*

i. As can be seen, the *age* variable has 87,990 missing values, with *2014* being the highest age and *1* being the lowest age. To clean this data, the mean imputation technique is used to replace:

- Any value smaller than 18 years old.
- Any value larger than 100 years old.
- All missing values for *age*.

The range of 18-100 years old is arbitrary and considered to be a reasonable range for the ages of users. It is to be noted that you need to be 18 years old to use Airbnb. To perform what has been described above, the following code is used:

```
cleaned_train <- train_users_2

cleaned_train$age[is.na(cleaned_train$age)] <- mean(cleaned_train$age, na.rm = TRUE)

cleaned_train$age[cleaned_train$age < 18 | cleaned_train$age > 100] <- mean(cleaned_train$age)
```

ii. For the other missing values, namely *date_first_booking*, with 124,543 missing values, and *first_affiliate_tracked* with 6,065 missing values, two approaches are used. With *first_affiliate_tracked*, all missing values are deleted, this is mainly due to the small percentage of missing values (2.8%).

For *date_first_booking*, deleting the large number of missing values can pose a problem, given that we have some variables with many factors (e.g. *language* and *first_browser*) and that the missing values account for almost half the data set. Hence, the missing values are imputed with the average date.

With other variables, if there is a factor for *unknown* or *other*, these are left as they may contain valuable information. Finally the variable *id* is removed as it does not help in our analysis. To delete the missing values the following code is used:

```
cleaned_train$date_first_booking[is.na(cleaned_train$date_first_booking)] <-
  mean(cleaned_train$date_first_booking, na.rm = TRUE)
cleaned_train <- cleaned_train[complete.cases(cleaned_train),]

cleaned_train$id <- NULL
```

This leaves us with 207,386 observations, or 97.2% of the original data set.

Part (b)

The new values for the descriptive summary are as follows:

```
summary(cleaned_train$date_account_created)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.
## "2010-01-01" "2013-01-17" "2013-09-18" "2013-07-08" "2014-03-10"
##         Max.
## "2014-06-30"
```

```
summary(cleaned_train$timestamp_first_active)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 2.009e+13 2.013e+13 2.013e+13 2.013e+13 2.014e+13 2.014e+13
```

```
summary(cleaned_train$date_first_booking)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.
## "2010-01-02" "2013-07-04" "2013-07-04" "2013-07-10" "2013-07-04"
##         Max.
## "2015-06-29"
```

```
summary(cleaned_train$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.00   32.00   49.00   42.01   49.67  100.00
```

```
summary(cleaned_train$signup_flow)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   3.151   0.000  25.000
```

```
summary(as.factor(cleaned_train$gender))
```

```
## -unknown-    FEMALE      MALE     OTHER
##     91783     61978     53347       278
```

```
summary(as.factor(cleaned_train$signup_method))
```

```
##    basic facebook   google
##   148297    58543      546
```

```
summary(as.factor(cleaned_train$language))
```

```
##     ca     cs     da     de     el     en     es     fi     fr     hr
##      5     32     58    715     24 200415    891     14   1149      2
##     hu     id     is     it     ja     ko     nl     no     pl     pt
##     18     22      5    490    224    722     95     30     54    235
##     ru     sv     th     tr     zh
##    379    122     23     63   1599
```

```
summary(as.factor(cleaned_train$affiliate_channel))
```

```
##           api       content        direct         other   remarketing
##          7749          3787        134179          8343          1058
##     sem-brand sem-non-brand           seo
##         25787         18028          8455
```

```
summary(as.factor(cleaned_train$affiliate_provider))
```

```
##               baidu                bing          craigslist
##                  29                2261                2990
##                daum              direct     email-marketing
##                   1              133935                 163
##            facebook facebook-open-graph              google
##                2201                 545               50459
##                 gsp              meetup               naver
##                 453                 347                  52
##               other           padmapper                vast
##               11909                 768                 752
##                wayn               yahoo              yandex
##                   8                 496                  17
```

```
summary(as.factor(cleaned_train$first_affiliate_tracked))
```

```
##        linked     local ops     marketing           omg       product
##         46287            34           139         43982          1556
## tracked-other     untracked
##          6156        109232
```

```
summary(as.factor(cleaned_train$signup_app))
```

```
## Android     iOS   Moweb     Web
##    5388   17871    5793  178334
```

```
summary(as.factor(cleaned_train$first_device_type))
```

```
##      Android Phone     Android Tablet    Desktop (Other)
##               2803               1292               1199
##               iPad             iPhone        Mac Desktop
##              14339              20759              89600
```

```
##      Other/Unknown SmartPhone (Other)    Windows Desktop
##               4602                 76              72716
```

```
summary(as.factor(cleaned_train$first_browser))
```

```
##            -unknown-      Android Browser         AOL Explorer
##                21201                  851                  245
##           Apple Mail               Arora        Avant Browser
##                   36                    1                    4
##   BlackBerry Browser              Camino               Chrome
##                   53                    9                63845
##        Chrome Mobile             Chromium            CometBird
##                 1270                   73                   11
##        Comodo Dragon             Conkeror             CoolNovo
##                    2                    1                    6
##        Crazy Browser                 Epic              Firefox
##                    2                    1                33655
##                Flock         Google Earth            Googlebot
##                    2                    1                    1
##            IceDragon            IceWeasel                   IE
##                    1                   13                21068
##            IE Mobile                 Iron       Kindle Browser
##                   36                   17                    1
##              Maxthon       Mobile Firefox        Mobile Safari
##                   46                   30                19274
##              Mozilla          NetNewsWire              OmniWeb
##                    3                    1                    2
##                Opera           Opera Mini         Opera Mobile
##                  188                    4                    2
##         Outlook 2007            Pale Moon Palm Pre web browser
##                    1                   12                    1
##      PS Vita browser              RockMelt               Safari
##                    1                   24                45169
##            SeaMonkey                 Silk            SiteKiosk
##                   11                  124                   24
##          SlimBrowser       Sogou Explorer            Stainless
##                    2                   33                    1
##           TenFourFox     TheWorld Browser           wOSBrowser
##                    8                    2                    6
##       Yandex.Browser
##                   11
```

```
summary(as.factor(cleaned_train$country_destination))
```

```
##     AU     CA     DE     ES     FR     GB     IT    NDF     NL  other
##    527   1391   1041   2213   4899   2295   2791 120216    751   9981
##     PT     US
##    214  61067
```

It appears that no factor was disproportionately affected by the deletion of the missing values of *first_affiliate_tracked*. Some highlights from the descriptive analysis:

- The biggest group in the *gender* variable is *unknown* (44.3%).
- The basic sign up method is by far the most common (71.5%).
- English is by far the most common language (96.6%).
- There are 25 selected language preferences.

- Bookings are mainly provided directly as the direct group in the *affiliate_provider* variable comprises of 64.6%.
- The majority of the first affiliates are not tracked (52.7%).
- The web is by far the most popular sign-up app (86.0%).
- The three most common browsers (Chrome, Safari and Firefox) account for 68.8% of all the instances in the *first_browser* variable.
- There are over 20 different first browsers
- The majority of the sessions did not lead to a booking (58.0%).
- The most booked place was the US with 70.1% of all bookings

Part (c)

Now the data set is divided into a training set and a test set. But first the data set is reduced to a size of 20,000 to make it easier and quicker to compute. As is convention, 80% will be the training set and 20% will be the test set. The following code is used:

```
set.seed(12345)

library(caTools)
smaller = sample.split(cleaned_train$country_destination, SplitRatio = 0.0964385)
smallSet = subset(cleaned_train, smaller == TRUE)


sample <- sample.split(smallSet$country_destination, SplitRatio = 0.8)

train = subset(smallSet, sample == TRUE)
test = subset(smallSet, sample == FALSE)
```

To check whether the train and test set have similar proportions of destination countries, the descriptive statistics of the target variable of each are checked again.

```
summary(as.factor(train$country_destination))
```

```
##    AU    CA    DE    ES    FR    GB    IT   NDF    NL other    PT    US
##    41   107    80   170   378   177   215  9274    58   770    17  4711
```

```
summary(as.factor(test$country_destination))
```

```
##    AU    CA    DE    ES    FR    GB    IT   NDF    NL other    PT    US
##    10    27    20    43    94    44    54  2319    14   193     4  1178
```

It appears that the train set has about 4 times as many of each country as the test set, this is what we expected.

Part (d)

Now the classification tree will be trained and its performance will be evaluated, all possible variables were included, except for *id*. The one variable that was not possible to include was *language* as the training set contained languages that were not in the test set and vice versa. The cause of this is likely due to English being a dominating language and the abundance of language preferences with few instances.

```
library("rpart")
library("rpart.plot")
library("caret")
```

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

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

```
library("e1071")

tree = rpart(country_destination ~ date_account_created + timestamp_first_active +
               date_first_booking + gender + age + signup_method + signup_flow +
               affiliate_channel + affiliate_provider + first_affiliate_tracked +
               signup_app + first_device_type + first_browser,
             data = train,
             method = "class",
             parms = list(split = "information"))

testPredict = test
testPredict$country_destination = NULL
testPredict$language = NULL

predictions = as.character(predict(tree, testPredict, type = "class"))
actual = test$country_destination

confusionMatrix(predictions, actual)
```

```
## Warning in confusionMatrix.default(predictions, actual): Levels are not in
## the same order for reference and data. Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   AU   CA   DE   ES   FR   GB   IT  NDF   NL other   PT   US
##      AU       0    0    0    0    0    0    0    0    0     0    0    0
##      CA       0    0    0    0    0    0    0    0    0     0    0    0
##      DE       0    0    0    0    0    0    0    0    0     0    0    0
##      ES       0    0    0    0    0    0    0    0    0     0    0    0
##      FR       0    0    0    0    0    0    0    0    0     0    0    0
##      GB       0    0    0    0    0    0    0    0    0     0    0    0
##      IT       0    0    0    0    0    0    0    0    0     0    0    0
##      NDF      0    0    0    0    0    0    0 2319    0     0    0    0
##      NL       0    0    0    0    0    0    0    0    0     0    0    0
##      other    0    0    0    0    0    0    0    0    0     0    0    0
##      PT       0    0    0    0    0    0    0    0    0     0    0    0
##      US      10   27   20   43   94   44   54    0   14   193    4 1178
##
## Overall Statistics
##
##                Accuracy : 0.8742
##                  95% CI : (0.8636, 0.8844)
##     No Information Rate : 0.5798
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7672
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: AU Class: CA Class: DE Class: ES Class: FR
## Sensitivity             0.0000   0.00000     0.000   0.00000    0.0000
```

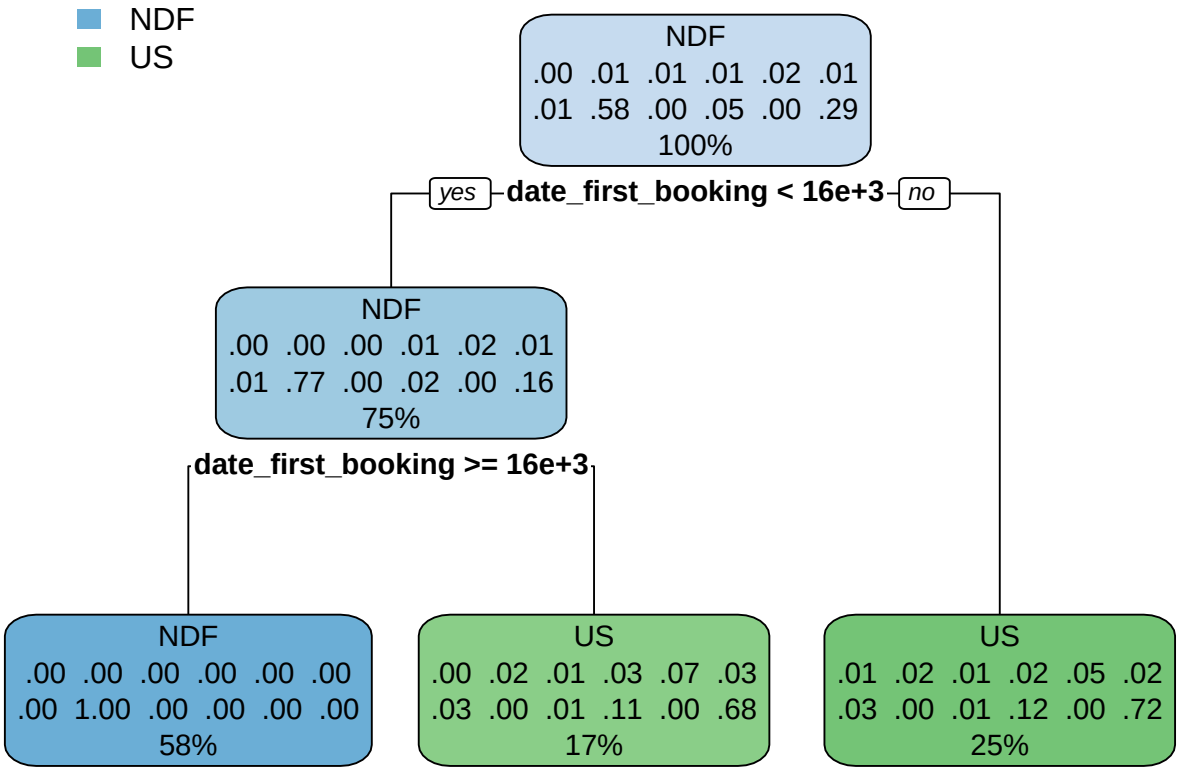
```
## Specificity             1.0000   1.00000     1.000   1.00000    1.0000
## Pos Pred Value             NaN       NaN       NaN       NaN       NaN
## Neg Pred Value          0.9975   0.99325     0.995   0.98925    0.9765
## Prevalence              0.0025   0.00675     0.005   0.01075    0.0235
## Detection Rate          0.0000   0.00000     0.000   0.00000    0.0000
## Detection Prevalence    0.0000   0.00000     0.000   0.00000    0.0000
## Balanced Accuracy       0.5000   0.50000     0.500   0.50000    0.5000
##                      Class: GB Class: IT Class: NDF Class: NL Class: other
## Sensitivity              0.000    0.0000     1.0000    0.0000      0.00000
## Specificity              1.000    1.0000     1.0000    1.0000      1.00000
## Pos Pred Value             NaN       NaN     1.0000       NaN          NaN
## Neg Pred Value           0.989    0.9865     1.0000    0.9965      0.95175
## Prevalence               0.011    0.0135     0.5797    0.0035      0.04825
## Detection Rate           0.000    0.0000     0.5797    0.0000      0.00000
## Detection Prevalence     0.000    0.0000     0.5797    0.0000      0.00000
## Balanced Accuracy        0.500    0.5000     1.0000    0.5000      0.50000
##                      Class: PT Class: US
## Sensitivity              0.000    1.0000
## Specificity              1.000    0.8218
## Pos Pred Value             NaN    0.7008
## Neg Pred Value           0.999    1.0000
## Prevalence               0.001    0.2945
## Detection Rate           0.000    0.2945
## Detection Prevalence     0.000    0.4203
## Balanced Accuracy        0.500    0.9109
```

With we can see above that the accuracy is 87.4%. The model appears to have focused on only getting *NDF* and *US* right. This seems logical as these comprise of 87.4% of the sample. The model did not bother trying to learn any other destinations, likely because the next largest destination, that is not *other*, is *FR* with only 2.4% of the sample.

Part (e)

The decision tree looks as follows:

```
rpart.plot(tree)
```



It appears that the model bases its decision solely on the date of the first booking. It is not clear from the rule set what the dates are, but there appear to only be 2 categories:

- At a certain date.
- Any other date.

With just these three rules the model was able to perfectly predict all *NDF* instances. This result is intriguing as the majority of the *date_first_booking* values were missing. The fact that such a perfect cutoff is possible is very suspicious as logically *date_first_booking* does not relate to country booking choice.

Further investigation shows that the number of missing values for *date_first_booking* in **Part (a)** is exactly the same as the number of *NDF* occurrences (124,543). These missing values were imputed to the mean of *date_first_booking*, this was *2013-07-04*. When we look in the data set itself we see all *NDF* occurrences happen on that same date. The most likely explanation for this relationship is that all sessions were first-time sessions, hence if someone does not book, there will not be a *date_first_booking*. The exact date that was then imputed was only the same for the instances of *NDF*. Although this model scores relatively high in accuracy, its validity is quite low due to a technicality of how the *date_first_booking* are used.

Another tree was made where *date_first_booking* is removed, forcing the model to use other variables. This model only achieved an accuracy of 62.6%, it only used *age* and *signup_method* to decide the target variable. Although this model performs worse, it has higher validity. Again this model only chooses *NDF* or *US* and ignores all other options for *country_destination*. Another option, rather than disabling *date_first_booking* is to delete all *NDF* instances This option will be explored in **Part (f)** and **Part (g)**. The code for the the model without *date_first_booking* is shown below:

```
library("rpart")
library("rpart.plot")
library("caret")
library("e1071")
```

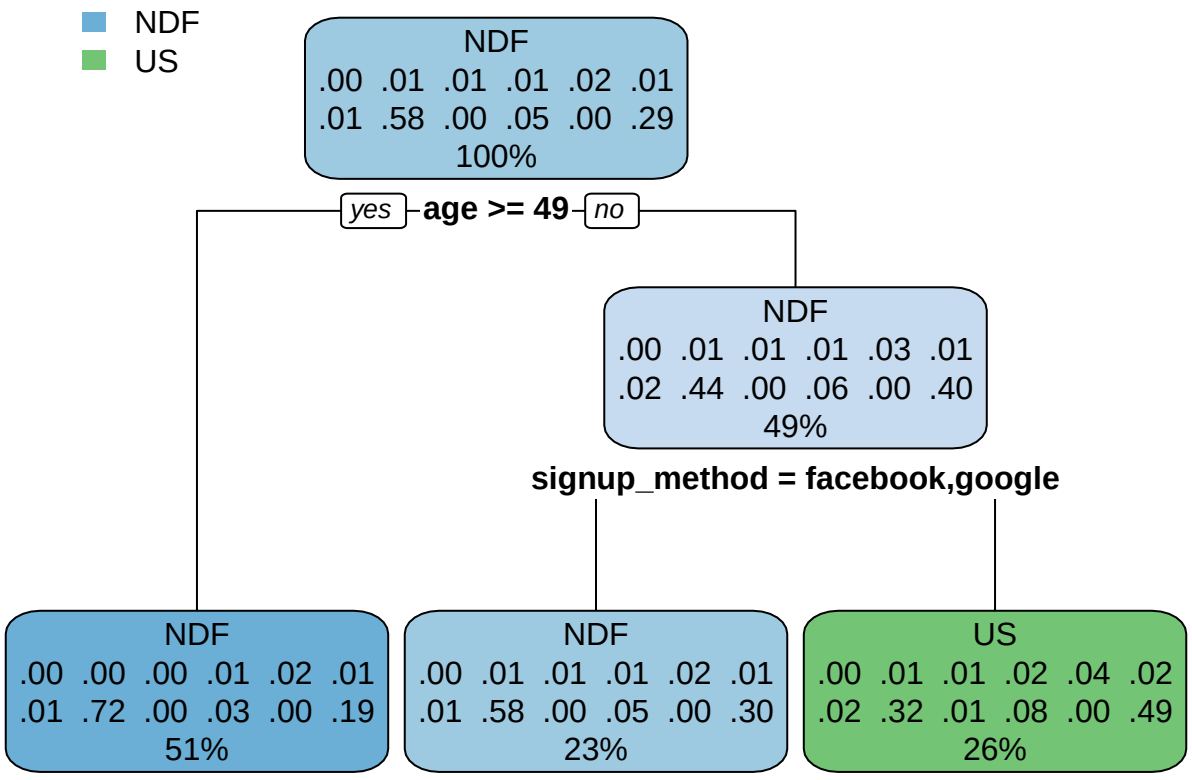
```
tree = rpart(country_destination ~ date_account_created + timestamp_first_active +
               gender + age + signup_method + signup_flow +
               affiliate_channel + affiliate_provider + first_affiliate_tracked +
               signup_app + first_device_type + first_browser,
             data = train,
             method = "class",
             parms = list(split = "information"))

testPredict = test
testPredict$country_destination = NULL
testPredict$language = NULL
testPredict$date_first_booking = NULL

predictions = as.character(predict(tree, testPredict, type = "class"))
actual = test$country_destination

confusionMatrix(predictions, actual)
```

```
## Warning in confusionMatrix.default(predictions, actual): Levels are not in
## the same order for reference and data. Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   AU   CA   DE   ES   FR   GB   IT  NDF   NL other   PT   US
##      AU       0    0    0    0    0    0    0    0    0     0    0    0
##      CA       0    0    0    0    0    0    0    0    0     0    0    0
##      DE       0    0    0    0    0    0    0    0    0     0    0    0
##      ES       0    0    0    0    0    0    0    0    0     0    0    0
##      FR       0    0    0    0    0    0    0    0    0     0    0    0
##      GB       0    0    0    0    0    0    0    0    0     0    0    0
##      IT       0    0    0    0    0    0    0    0    0     0    0    0
##      NDF      5   14   14   25   53   28   34 2003    6   119    3  677
##      NL       0    0    0    0    0    0    0    0    0     0    0    0
##      other    0    0    0    0    0    0    0    0    0     0    0    0
##      PT       0    0    0    0    0    0    0    0    0     0    0    0
##      US       5   13    6   18   41   16   20  316    8    74    1  501
##
## Overall Statistics
##
##                Accuracy : 0.626
##                  95% CI : (0.6108, 0.641)
##     No Information Rate : 0.5798
##     P-Value [Acc > NIR] : 1.394e-09
##
##                   Kappa : 0.2413
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: AU Class: CA Class: DE Class: ES Class: FR
## Sensitivity             0.0000   0.00000     0.000   0.00000    0.0000
## Specificity             1.0000   1.00000     1.000   1.00000    1.0000
## Pos Pred Value             NaN       NaN       NaN       NaN       NaN
```

```
## Neg Pred Value          0.9975   0.99325     0.995   0.98925    0.9765
## Prevalence              0.0025   0.00675     0.005   0.01075    0.0235
## Detection Rate          0.0000   0.00000     0.000   0.00000    0.0000
## Detection Prevalence    0.0000   0.00000     0.000   0.00000    0.0000
## Balanced Accuracy       0.5000   0.50000     0.500   0.50000    0.5000
##                      Class: GB Class: IT Class: NDF Class: NL Class: other
## Sensitivity              0.000    0.0000     0.8637    0.0000      0.00000
## Specificity              1.000    1.0000     0.4182    1.0000      1.00000
## Pos Pred Value             NaN       NaN     0.6719       NaN          NaN
## Neg Pred Value           0.989    0.9865     0.6899    0.9965      0.95175
## Prevalence               0.011    0.0135     0.5797    0.0035      0.04825
## Detection Rate           0.000    0.0000     0.5008    0.0000      0.00000
## Detection Prevalence     0.000    0.0000     0.7452    0.0000      0.00000
## Balanced Accuracy        0.500    0.5000     0.6410    0.5000      0.50000
##                      Class: PT Class: US
## Sensitivity              0.000    0.4253
## Specificity              1.000    0.8164
## Pos Pred Value             NaN    0.4917
## Neg Pred Value           0.999    0.7729
## Prevalence               0.001    0.2945
## Detection Rate           0.000    0.1252
## Detection Prevalence     0.000    0.2547
## Balanced Accuracy        0.500    0.6209
```

```
rpart.plot(tree)
```



Part (f)

Now we will build another classification tree, but after removing all instances where no booking was made (*NDF*). In this model we could not use *first_browser* for the same reason as *language* in **Part (d)**. Again we limit the total size of data set to 20,000. The following code is used:

```
clean_noNDA = cleaned_train[!(cleaned_train$country_destination == "NDF"),]

set.seed(12345)

library(caTools)
smaller = sample.split(clean_noNDA$country_destination, SplitRatio = 0.2294367)
smallSet = subset(clean_noNDA, smaller == TRUE)


sample <- sample.split(smallSet$country_destination, SplitRatio = 0.8)

train = subset(smallSet, sample == TRUE)
test = subset(smallSet, sample == FALSE)


library("rpart")
library("rpart.plot")
library("caret")
library("e1071")

tree = rpart(country_destination ~ date_account_created + timestamp_first_active +
               date_first_booking + gender + age + signup_method + signup_flow +
               affiliate_channel + affiliate_provider + first_affiliate_tracked +
               signup_app + first_device_type,
             data = train,
             method = "class",
             parms = list(split = "information"))

testPredict = test
testPredict$country_destination = NULL
testPredict$language = NULL

predictions = as.character(predict(tree, testPredict, type = "class"))
actual = test$country_destination

confusionMatrix(predictions, actual)
```

```
## Warning in confusionMatrix.default(predictions, actual): Levels are not in
## the same order for reference and data. Refactoring data to match.
```

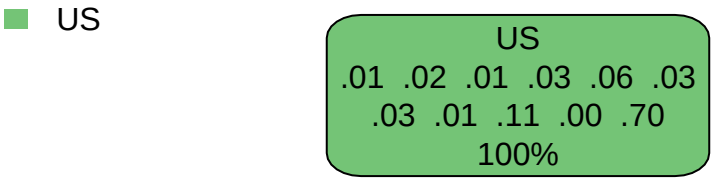
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   AU   CA   DE   ES   FR   GB   IT   NL other   PT   US
##      AU       0    0    0    0    0    0    0    0     0    0    0
##      CA       0    0    0    0    0    0    0    0     0    0    0
##      DE       0    0    0    0    0    0    0    0     0    0    0
##      ES       0    0    0    0    0    0    0    0     0    0    0
##      FR       0    0    0    0    0    0    0    0     0    0    0
```

```
##      GB       0    0    0    0    0    0    0    0     0    0    0
##      IT       0    0    0    0    0    0    0    0     0    0    0
##      NL       0    0    0    0    0    0    0    0     0    0    0
##      other    0    0    0    0    0    0    0    0     0    0    0
##      PT       0    0    0    0    0    0    0    0     0    0    0
##      US      24   64   48  102  225  105  128   34   458   10 2802
##
## Overall Statistics
##
##                Accuracy : 0.7005
##                  95% CI : (0.686, 0.7147)
##     No Information Rate : 0.7005
##     P-Value [Acc > NIR] : 0.5078
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: AU Class: CA Class: DE Class: ES Class: FR
## Sensitivity              0.000     0.000     0.000    0.0000   0.00000
## Specificity              1.000     1.000     1.000    1.0000   1.00000
## Pos Pred Value             NaN       NaN       NaN       NaN       NaN
## Neg Pred Value           0.994     0.984     0.988    0.9745   0.94375
## Prevalence               0.006     0.016     0.012    0.0255   0.05625
## Detection Rate           0.000     0.000     0.000    0.0000   0.00000
## Detection Prevalence     0.000     0.000     0.000    0.0000   0.00000
## Balanced Accuracy        0.500     0.500     0.500    0.5000   0.50000
##                      Class: GB Class: IT Class: NL Class: other Class: PT
## Sensitivity            0.00000     0.000    0.0000       0.0000    0.0000
## Specificity            1.00000     1.000    1.0000       1.0000    1.0000
## Pos Pred Value             NaN       NaN       NaN          NaN       NaN
## Neg Pred Value         0.97375     0.968    0.9915       0.8855    0.9975
## Prevalence             0.02625     0.032    0.0085       0.1145    0.0025
## Detection Rate         0.00000     0.000    0.0000       0.0000    0.0000
## Detection Prevalence   0.00000     0.000    0.0000       0.0000    0.0000
## Balanced Accuracy      0.50000     0.500    0.5000       0.5000    0.5000
##                      Class: US
## Sensitivity             1.0000
## Specificity             0.0000
## Pos Pred Value          0.7005
## Neg Pred Value             NaN
## Prevalence              0.7005
## Detection Rate          0.7005
## Detection Prevalence    1.0000
## Balanced Accuracy       0.5000
```

Part (g)

The decision tree looks as following:

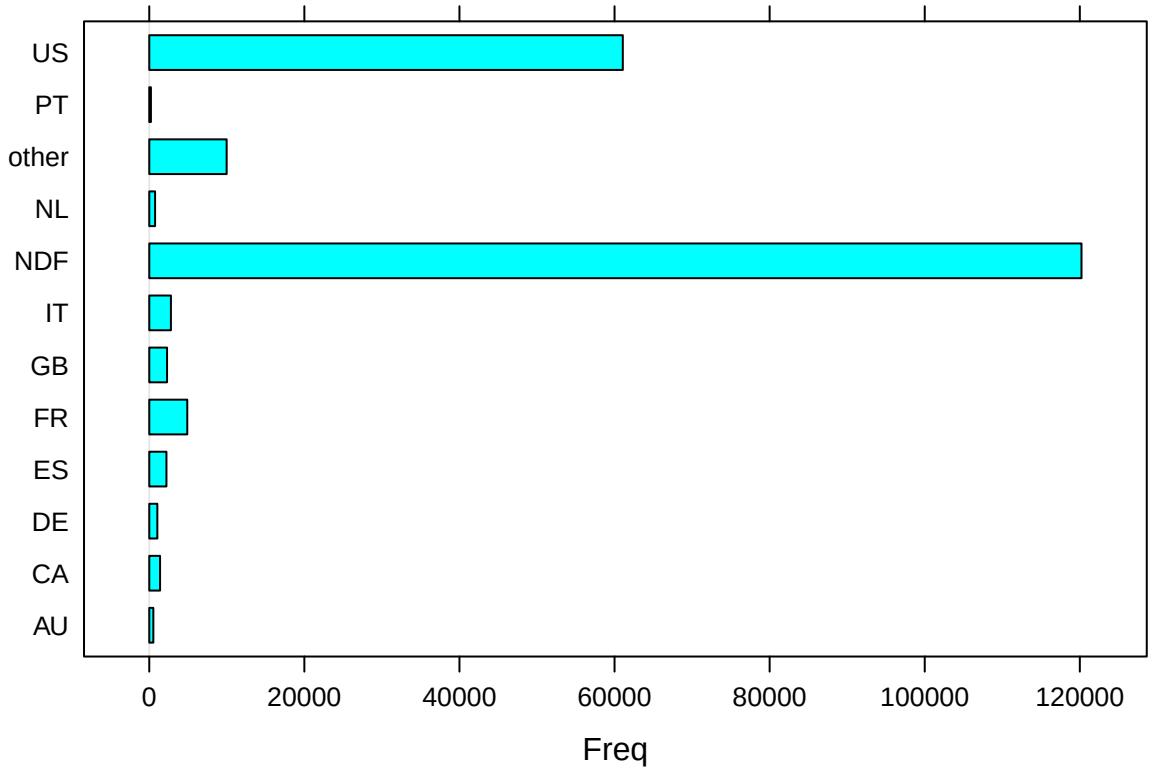
```
rpart.plot(tree)
```



As we can see, this model does not make any decisions. It simply always chooses *US* as this is the largest category (70.1%). The next largest category, that is not *other*, is again *FR*, this time with 5.6%. The same behavior is observed when the sample size is increased.

Compared to the second model with *NDF*, this model performs better. The performance of either model is mostly influenced by the unbalanced classes and the fact that there are many countries to choose from, a small proportion of which accounts for the majority of the instances. This unbalance becomes even more apparent when you plot a bar chart of all possible destinations.

```
counts = table(cleaned_train$country_destination)
barchart(counts)
```


US
PT
other
NL
NDF
IT
GB
FR
ES
DE
CA
AU
0
20000
40000
60000
80000
100000
120000
Freq