IDS572 HW4

Kritika Raghuwanshi;Shixun Jiang;Yaze Gao

2023-04-14

Reading the Data Set

To proceed with the analysis, we load the data set into R using the file.choose() function, which allows the user to select a file following a prompt. For ease of calling it, we store the values in a variable called **"rm"**, which stands for retention_model.

```
library(readxl)
travel_retention <- read_excel("C:/Travel Company Retention Model.xlsx")
dim(travel_retention)
```

```
## [1] 2392   56
```

A closer look at the data using the "View" function shows us that cells **2390 to 2392** are all **NA**, which isn't useful for our model. These NAs are across all variables and not specific to certain variables, so removing them from the data set for the purpose of further analysis would cause no harm. We also make use of the **"drop = FALSE"** function here to make the output a one dimensional matrix and not convert it into a Vector.

```
ogdata <- travel_retention #to keep a copy of the original data set as is
tr <- travel_retention[-(2390:2392),,drop = FALSE]#to remove the NA cells
dim(tr)
```

```
## [1] 2389   56
```

Data Modification

We have been informed in this data set's description that **"Retained in 2012"** is the target variable, but since it's currently in binary format (0 or 1), we convert it to something more meaningful using the **IfElse"** condition as follows:

```
library(knitr)
tr$target <- as.factor(ifelse(tr$Retained.in.2012. == 1,
                        "Returned", " Didn't Return"))
kable(table(tr$target))
```

Var1	Freq
Didn't Return	938
Returned	1451

```
sum(is.na(tr$target))
```

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

Similarly, we can convert two more binary variables into categorical, which are **"Is.Non.Annual"** & **"School.Sponsor"**, using the IfElse condition like above. This will help us with Exploratory Data Analysis (EDA) in the future:

```
tr$Is.Non.Annual. <- as.factor(ifelse(tr$Is.Non.Annual. == 1, "Yes", "No"))
tr$School.Sponsor <- as.factor(ifelse(tr$School.Sponsor == 1,
                                "Sponsoring", "Not Sponsoring"))
```

After this, we've got a few variables, whose values are quite large, for example, **"SchoolGradeType"** has values called **"Elementary->Elementary"**, which we can convert to **"E->E"** etc. for better data visualization, and similarly for the variable **"Income.Level"**, we've got many levels, which don't quite mean anything, as it's got levels A to Z, and then some for P1, P2 etc., which are not coherent. In order to fix this, we can use the **"Mutate"** and the **"Recode"** function in R to convert these values as shown below:

```
library(dplyr)
tr <- tr %>% mutate(SchoolGradeType = recode(SchoolGradeType,
                                "Elementary->Elementary" = "E->E",
                                "Middle->Middle" = "M->M",
                                "High->High" = "H->H",
                                "Undefined->Undefined" = "U->U",
                                "Middle->Undefined" = "M->U",
                                "Elementary->Middle" = "E->M",
                                "Middle->High" = "M->H",
                                "Elementary->High" = "E->H",
                                "Elementary->Undefined" = "E->U"))
unique(tr$SchoolGradeType)
```

```
## [1] "E->E" "M->M" "H->H" "U->U" "M->U" "E->M" "M->H" "E->H" "E->U"
```

For the Income.Level variable, we can map the values in such a way:

1. A, B, C, D, and E = Low
2. F, G, H, I, J, K, and L = Medium
3. M, N, O, P, Q, and Z = High
4. P1, P2, P3, and P4 = High (as it's a variation of P category)

```
tr <- tr %>% mutate(Income.Level = recode(Income.Level,
                                "A" = "Low",
                                "B" = "Low",
                                "C" = "Low",
                                "D" = "Low",
                                "E" = "Low",
                                "F" = "Medium",
                                "G" = "Medium",
                                "H" = "Medium",
                                "I" = "Medium",
                                "J" = "Medium",
                                "K" = "Medium",
                                "L" = "Medium",
                                "M" = "High",
                                "N" = "High",
                                "O" = "High",
                                "P" = "High",
                                "Q" = "High",
                                "Z" = "Unclassified",
                                "P1" = "High",
                                "P3" = "High",
                                "P4" = "High",
                                "P5" = "High"))
unique(tr$Income.Level)
```

```
## [1] "High"         "Low"          NA             "Medium"       "Unclassified"
```

Data Removal / Cleaning

A closer look at the data set shows us that we've got a lot of redundant columns or irrelevant information, which is better removed than kept, so we can do better analysis overall. See the below table for column names that we've removed, along with their description:

```
library(knitr)

tbl <- data.frame(
ColumnNumber = c("Column 1", "Column 2", "Column 5", "Column 9",
                 "Column 10", "Column 11", "Column 12", "Column 17",
                 "Column 18", "Column 21", "Column 37", "Column 39",
                 "Column 40", "Column 56"),

ColumnName = c("ID", "Program.code", "Group.state", "Departure.Date",
               "Return.Date", "Deposit.Date", "Special.Pay", "Early.RPL",
               "Latest.RPL", "Initial.System.Date",  "SPR.Group.Revenue",
               "FirstMeeting", "LastMeeting", "Retained.in.2012"),

ReasonForRemoval = c("Irrelevant Variable", "Irrelevant Variable",
                     "Correlated Variable", "Correlated & Date Variable",
                     "Correlated & Date Variable", "Date Variable",
                     "Missing Variables (majority data missing)",
                     "Date Variable", "Date Variable", "Date Variable",
                     "Irrelevant Variable", "Correlated Variable",
                     "Correlated Variable", "Variable replaced with 'target'"))
kable(tbl)
```

ColumnNumber	ColumnName	ReasonForRemoval
Column 1	ID	Irrelevant Variable
Column 2	Program.code	Irrelevant Variable
Column 5	Group.state	Correlated Variable
Column 9	Departure.Date	Correlated & Date Variable
Column 10	Return.Date	Correlated & Date Variable
Column 11	Deposit.Date	Date Variable
Column 12	Special.Pay	Missing Variables (majority data missing)
Column 17	Early.RPL	Date Variable
Column 18	Latest.RPL	Date Variable
Column 21	Initial.System.Date	Date Variable
Column 37	SPR.Group.Revenue	Irrelevant Variable
Column 39	FirstMeeting	Correlated Variable
Column 40	LastMeeting	Correlated Variable
Column 56	Retained.in.2012	Variable replaced with 'target'

Code for removing the variable and the dimension of the data set after removal

```
dim(tr)
```

```
## [1] 2389   57
```

```
tr <- tr[,-c(1, 2, 5, 9, 10, 11, 12, 17, 18, 21, 37, 39, 40, 56)]#removal
dim(tr)
```

```
## [1] 2389   43
```

Now that we've removed some of the variables, we also need to clean the data set to remove **"NAs"** from all the variables. In order to see which variables still have NAs, we write the following code:

```
sum(is.na(tr))#to get the sum of NAs in the data set
```

```
## [1] 911
```

```
kable(colSums(is.na(tr)))#to get the column names having NA values within them
```

	x
From.Grade	0
To.Grade	0
Is.Non.Annual.	0
Days	0
Travel.Type	0
Tuition	0
FRP.Active	0
FRP.Cancelled	0
FRP.Take.up.percent.	0
Cancelled.Pax	0
Total.Discount.Pax	0
Poverty.Code	599
Region	0
CRM.Segment	0
School.Type	0
Parent.Meeting.Flag	0
MDR.Low.Grade	68
MDR.High.Grade	0
Total.School.Enrollment	91
Income.Level	62
EZ.Pay.Take.Up.Rate	0
School.Sponsor	0
SPR.Product.Type	0
SPR.New.Existing	0
FPP	0
Total.Pax	0
NumberOfMeetingswithParents	0
DifferenceTraveltoFirstMeeting	0
DifferenceTraveltoLastMeeting	0

	x
SchoolGradeTypeLow	0
SchoolGradeTypeHigh	0
SchoolGradeType	0
DepartureMonth	0
GroupGradeTypeLow	0
GroupGradeTypeHigh	0
GroupGradeType	0
MajorProgramCode	0
SingleGradeTripFlag	0
FPP.to.School.enrollment	0
FPP.to.PAX	0
Num.of.Non_FPP.PAX	0
SchoolSizeIndicator	91
target	0

However, the above code only shows NA values which are simply missing values within the data set. It does not account for all those variables where NAs are written in the following formats:

1. "NA" or
2. 'NA'

In order to fix this issue, we converted all **single and double quoted "NAs"** to simple NAs as follows:

```
tr[tr == "NA"] <- NA
sum(is.na(tr))
```

```
## [1] 2025
```

```
kable(colSums(is.na(tr)))
```

	x
From.Grade	127
To.Grade	150
Is.Non.Annual.	0
Days	0
Travel.Type	0
Tuition	0
FRP.Active	0
FRP.Cancelled	0
FRP.Take.up.percent.	0
Cancelled.Pax	0
Total.Discount.Pax	0
Poverty.Code	599
Region	0
CRM.Segment	4
School.Type	0
Parent.Meeting.Flag	0
MDR.Low.Grade	68
MDR.High.Grade	68
Total.School.Enrollment	91
Income.Level	62
EZ.Pay.Take.Up.Rate	0
School.Sponsor	0
SPR.Product.Type	0
SPR.New.Existing	0
FPP	0
Total.Pax	0
NumberOfMeetingswithParents	0
DifferenceTraveltoFirstMeeting	337
DifferenceTraveltoLastMeeting	337
SchoolGradeTypeLow	0
SchoolGradeTypeHigh	0
SchoolGradeType	0
DepartureMonth	0
GroupGradeTypeLow	0
GroupGradeTypeHigh	0

	x
GroupGradeType	0
MajorProgramCode	0
SingleGradeTripFlag	0
FPP.to.School.enrollment	91
FPP.to.PAX	0
Num.of.Non_FPP.PAX	0
SchoolSizeIndicator	91
target	0

As we can see from the above results, we've got about 2025 NA values that we need to clean up in our data set, where it would not be prudent to remove them as is. Instead, we can replace the NA values with the mode of all the values in a variable. For this, we can utilize a function to calculate the mode as shown below. The mode function calculates the mode for replacing these NA values for numerical variables while replacing the most occurring value for the categorical ones.

```
Modes <- function(x) {
  ux <- unique(x)
  tab <- tabulate(match(x, ux))
  ux[tab == max(tab)]
}
```

Now that we've created a function to find the statistical mode, we can convert the NA values to modes for each of those variables that have missing values. We go in the order listed above, starting with **From.Grade** and going to **"SchoolSizeIndicator"**:

1. Removing NAs from the "From.Grade" variable:

```
mode_FromGrade <- Modes(tr$From.Grade)

tr$From.Grade <- replace(tr$From.Grade,is.na(tr$From.Grade), mode_FromGrade)

sum(is.na(tr$From.Grade))
```

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

2. Removing NAs from the "To.Grade" variable:

```
mode_ToGrade <- Modes(tr$To.Grade)

tr$To.Grade <- replace(tr$To.Grade,is.na(tr$To.Grade), mode_ToGrade)

sum(is.na(tr$To.Grade))
```

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

3. Removing "N" values from "Travel.Type" variable:

Since the Travel.Type variable has 2 "N" values, which are anomalies, as the variable should only contain three modes of travel: Air (A), Bus (B), and Train (T). We try to remove them using the following method, where we search for entries with "N" and the remove them:

```
sum((tr$Travel.Type) == 'N')
```

```
## [1] 2
```

```
mode_TravelType <- Modes(tr$Travel.Type)

tr$Travel.Type <- replace(tr$Travel.Type, tr$Travel.Type == 'N',
                          mode_TravelType)

sum((tr$Travel.Type) == 'N')
```

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

4. Removing NAs from the "Poverty.Code" variable:

```
mode_PovertyCode <- Modes(tr$Poverty.Code)

tr$Poverty.Code <- replace(tr$Poverty.Code,is.na(tr$Poverty.Code),
                           mode_PovertyCode)

sum(is.na(tr$Poverty.Code))
```

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

5. Removing NAs from the "CRM.Segment" variable:

```
mode_CRMSegment <- Modes(tr$CRM.Segment)

tr$CRM.Segment <- replace(tr$CRM.Segment,is.na(tr$CRM.Segment),
                           mode_CRMSegment)

sum(is.na(tr$CRM.Segment))
```

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

6. Removing NAs from the "MDR.Low.Grade" variable:

```
mode_MDRLowGrade <- Modes(tr$MDR.Low.Grade)

tr$MDR.Low.Grade <- replace(tr$MDR.Low.Grade,is.na(tr$MDR.Low.Grade),
                            mode_MDRLowGrade)

sum(is.na(tr$MDR.Low.Grade))
```

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

7. Removing NAs from the "MDR.High.Grade" variable:

```
mode_MDRHighGrade <- Modes(tr$MDR.High.Grade)

tr$MDR.High.Grade <- replace(tr$MDR.High.Grade,is.na(tr$MDR.High.Grade),
                            mode_MDRHighGrade)

sum(is.na(tr$MDR.High.Grade))
```

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

8. Removing NAs from the "Total.School.Enrollment" variable:

Here we calculate the mean v/s the mode given the data within this variable.

```r
mean_TotalSchoolEnrollment <- round(mean(tr$Total.School.Enrollment,
                                         na.rm = TRUE))

tr$Total.School.Enrollment <- replace(tr$Total.School.Enrollment,
                              is.na(tr$Total.School.Enrollment),
                              mean_TotalSchoolEnrollment)

sum(is.na(tr$Total.School.Enrollment))
```

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

9. Removing NAs from the "Income.Level" variable:

```
mode_IncomeLevel <- Modes(tr$Income.Level)

tr$Income.Level <- replace(tr$Income.Level,is.na(tr$Income.Level),
                           mode_IncomeLevel)

sum(is.na(tr$Income.Level))
```

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

```
unique(tr$Income.Level)
```

```
## [1] "High"         "Low"          "Medium"       "Unclassified"
```

10. Removing NAs from the "DifferenceTraveltoFirstMeeting" variable:

Here we calculate the mean v/s the mode given the data within this variable.

```
mean_DifferenceTraveltoFirstMeeting <- round(mean(as.numeric(
tr$DifferenceTraveltoFirstMeeting),na.rm=TRUE))

tr$DifferenceTraveltoFirstMeeting <- replace(tr$DifferenceTraveltoFirstMeeting,
                                     is.na(tr$DifferenceTraveltoFirstMeeting),
                                     mean_DifferenceTraveltoFirstMeeting)

sum(is.na(tr$DifferenceTraveltoFirstMeeting))
```

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

11. Removing NAs from the "DifferenceTraveltoLastMeeting" variable:

Here we calculate the mean v/s the mode given the data within this variable.

```
mean_DifferenceTraveltoLastMeeting <- round(mean(as.numeric(
tr$DifferenceTraveltoLastMeeting),na.rm=TRUE))

tr$DifferenceTraveltoLastMeeting <- replace(tr$DifferenceTraveltoLastMeeting,
                                     is.na(tr$DifferenceTraveltoLastMeeting),
                                     mean_DifferenceTraveltoLastMeeting)

sum(is.na(tr$DifferenceTraveltoLastMeeting))
```

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

12. Removing NAs from the "FPP.to.School.enrollment" variable:

Here we calculate the mean v/s the mode given the data within this variable.

```
mean_FPPToSchoolEnrollment <- round(mean(as.numeric(tr$FPP.to.School.enrollment)
                                        ,na.rm=TRUE))

tr$FPP.to.School.enrollment <- replace(tr$FPP.to.School.enrollment,
                                is.na(tr$FPP.to.School.enrollment),
                                mean_FPPToSchoolEnrollment)

sum(is.na(tr$FPP.to.School.enrollment))
```

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

13. Removing NAs from the "SchoolSizeIndicator" variable:

```
tr$SchoolSizeIndicator <- sapply(tr$SchoolSizeIndicator,
                                 as.character, na.rm=TRUE)

mode_SchoolSizeIndicator <- Modes(tr$SchoolSizeIndicator)

tr$SchoolSizeIndicator <- replace(tr$SchoolSizeIndicator,
                                  is.na(tr$SchoolSizeIndicator),
                                  mode_SchoolSizeIndicator)


sum(is.na(tr$SchoolSizeIndicator))
```

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

```
unique(tr$SchoolSizeIndicator)
```

```
## [1] "L"   "S-M" "M-L" "S"
```

Post removing all the NAs, quoted and unquoted, our table looks like the following:

```
kable(colSums(is.na(tr)))
```

	x
From.Grade	0
To.Grade	0
Is.Non.Annual.	0
Days	0
Travel.Type	0
Tuition	0
FRP.Active	0
FRP.Cancelled	0
FRP.Take.up.percent.	0
Cancelled.Pax	0
Total.Discount.Pax	0
Poverty.Code	0
Region	0
CRM.Segment	0
School.Type	0
Parent.Meeting.Flag	0
MDR.Low.Grade	0
MDR.High.Grade	0
Total.School.Enrollment	0
Income.Level	0
EZ.Pay.Take.Up.Rate	0
School.Sponsor	0
SPR.Product.Type	0
SPR.New.Existing	0
FPP	0
Total.Pax	0
NumberOfMeetingswithParents	0
DifferenceTraveltoFirstMeeting	0
DifferenceTraveltoLastMeeting	0
SchoolGradeTypeLow	0
SchoolGradeTypeHigh	0
SchoolGradeType	0
DepartureMonth	0
GroupGradeTypeLow	0
GroupGradeTypeHigh	0
GroupGradeType	0
MajorProgramCode	0
SingleGradeTripFlag	0
FPP.to.School.enrollment	0
FPP.to.PAX	0
Num.of.Non_FPP.PAX	0
SchoolSizeIndicator	0
target	0

Converting variables to Factors

Now that we've cleaned the entire data set, we need to convert some variables into factors for the portion to follow, where we'll be doing Exploratory Data Analysis (EDA):

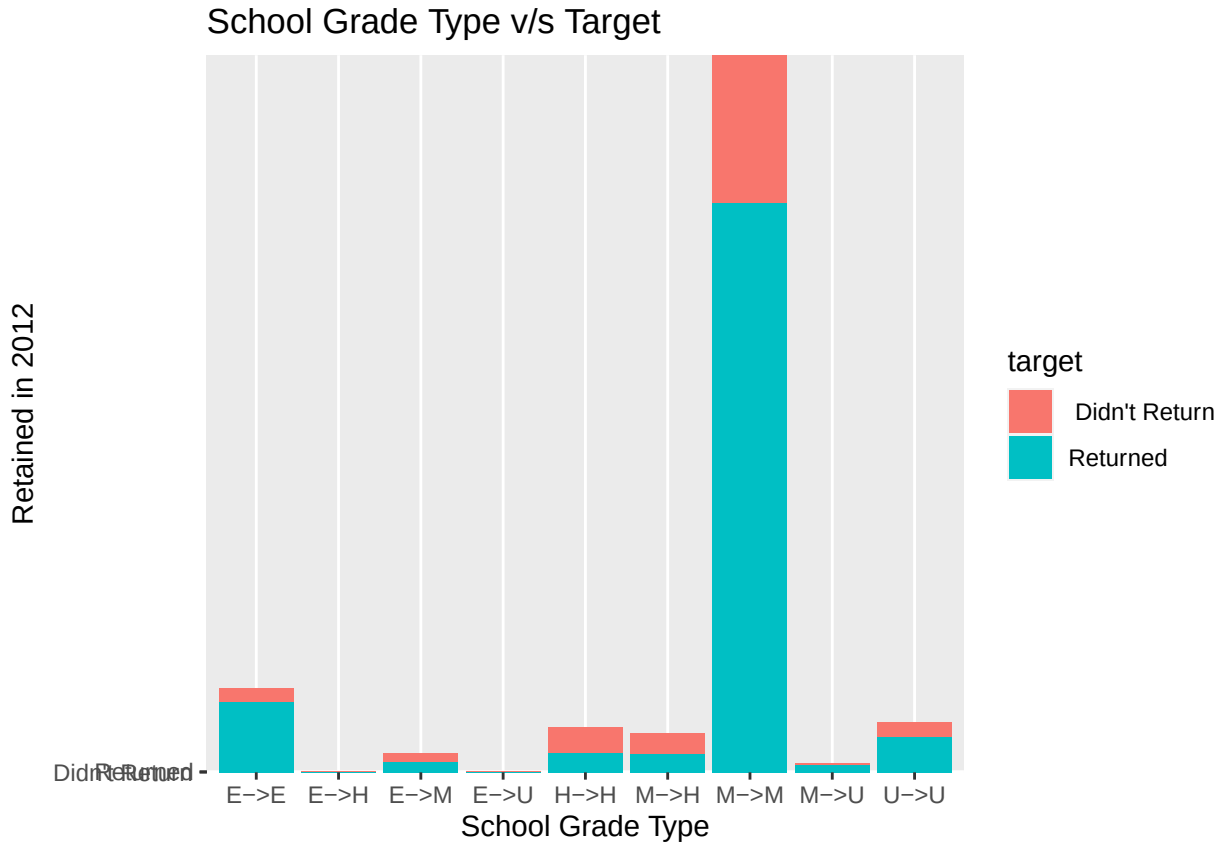
```
tr$From.Grade <- as.factor(tr$From.Grade)
tr$To.Grade <- as.factor(tr$To.Grade)
tr$Travel.Type <- as.factor(tr$Travel.Type)
tr$Poverty.Code <- as.factor(tr$Poverty.Code)
tr$Region <- as.factor(tr$Region)
tr$CRM.Segment <- as.factor(tr$CRM.Segment)
tr$School.Type <- as.factor(tr$School.Type)
tr$MDR.Low.Grade <- as.factor(tr$MDR.Low.Grade)
tr$MDR.High.Grade  <- as.factor(tr$MDR.High.Grade)
tr$Income.Level  <- as.factor(tr$Income.Level)
tr$SPR.Product.Type  <- as.factor(tr$SPR.Product.Type)
tr$SPR.New.Existing  <- as.factor(tr$SPR.New.Existing)

tr$DifferenceTraveltoFirstMeeting <- as.numeric(
tr$DifferenceTraveltoFirstMeeting)

tr$DifferenceTraveltoLastMeeting <- as.numeric(
tr$DifferenceTraveltoLastMeeting)

tr$SchoolGradeTypeLow <- as.factor(tr$SchoolGradeTypeLow)
tr$SchoolGradeTypeHigh <- as.factor(tr$SchoolGradeTypeHigh)
tr$SchoolGradeType <- as.factor(tr$SchoolGradeType)
tr$GroupGradeTypeLow <- as.factor(tr$GroupGradeTypeLow)
tr$GroupGradeTypeHigh <- as.factor(tr$GroupGradeTypeHigh)
tr$GroupGradeType <- as.factor(tr$GroupGradeType)
tr$DepartureMonth <- as.factor(tr$DepartureMonth)
tr$MajorProgramCode <- as.factor(tr$MajorProgramCode)

tr$FPP.to.School.enrollment <- as.numeric(tr$FPP.to.School.enrollment)

tr$MajorProgramCode <- as.factor(tr$MajorProgramCode)
tr$SchoolSizeIndicator <- as.factor(tr$SchoolSizeIndicator)
```

Exploratory Data Analysis (EDA)

In order to perform EDA, we can use some of the key variables and measure them against our target variable called **"target"**.

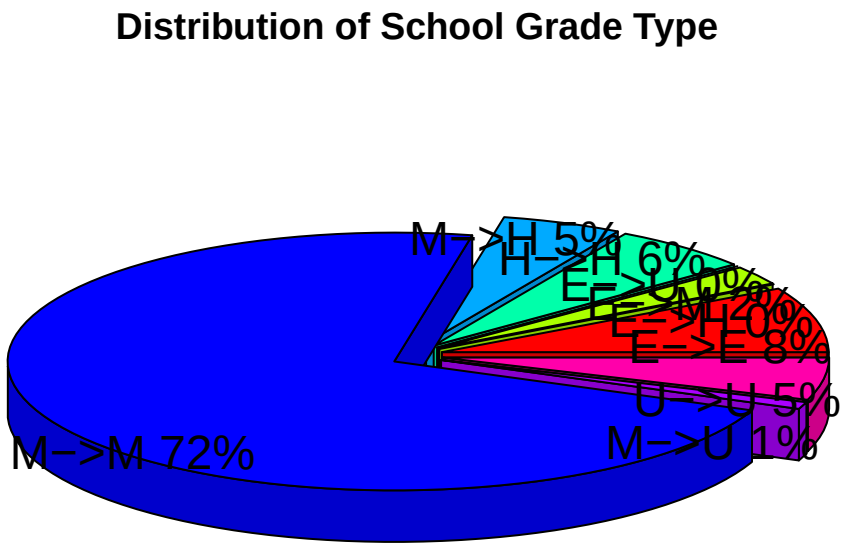
1. SchoolGradeType v/s target

```
library(ggplot2)
library(scales)
ggplot(tr, aes(SchoolGradeType, target, fill = target)) +
geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
labs(title = "School Grade Type v/s Target", x = "School Grade Type",
             y= "Retained in 2012")
```



Pie chart for School Grade Type distribution

```
library(ggplot2)
library(plotrix)
pietable <- table(tr$SchoolGradeType)
percent <- round(pietable/sum(pietable)*100)
label1 <- paste(names(pietable),percent)
label2 <- paste(label1, "%", sep="")
pie3D(pietable, labels = label2, explode = 0.1,
main="Distribution of School Grade Type", radius = 1.5)
```

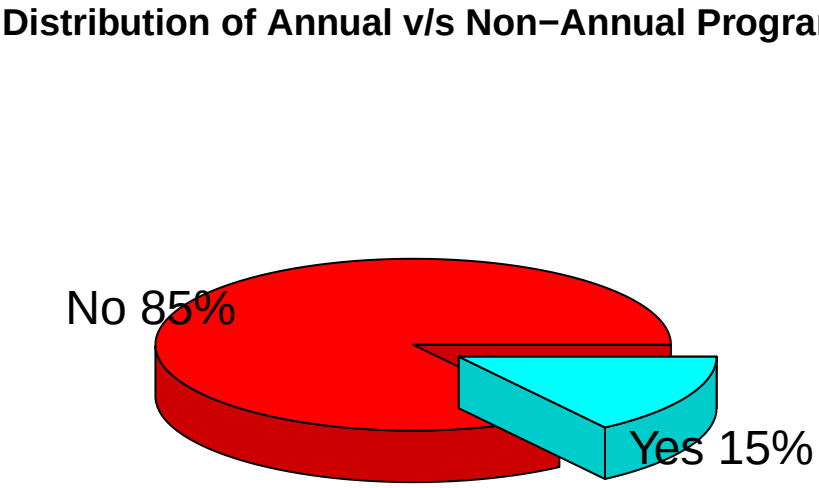


Analysis

From the first chart, we notice that **"Middle->Middle School"** grade type return to the Scholastic Travel Company (STC) more frequently compared to other grade types, with the next best being **Elementary->Elementary** grade type. However, we also notice that the highest ratio of not returning is also for the middle school grade type. For this reason, we pull a pie chart of the school grade type distribution. The pie chart shows us that the above result is due to "Middle School" grade type occupying 72% of the data, while the rest of the school grade types don't have much spread across the data set. **Hence our data shows a spike for middle school grade type.**

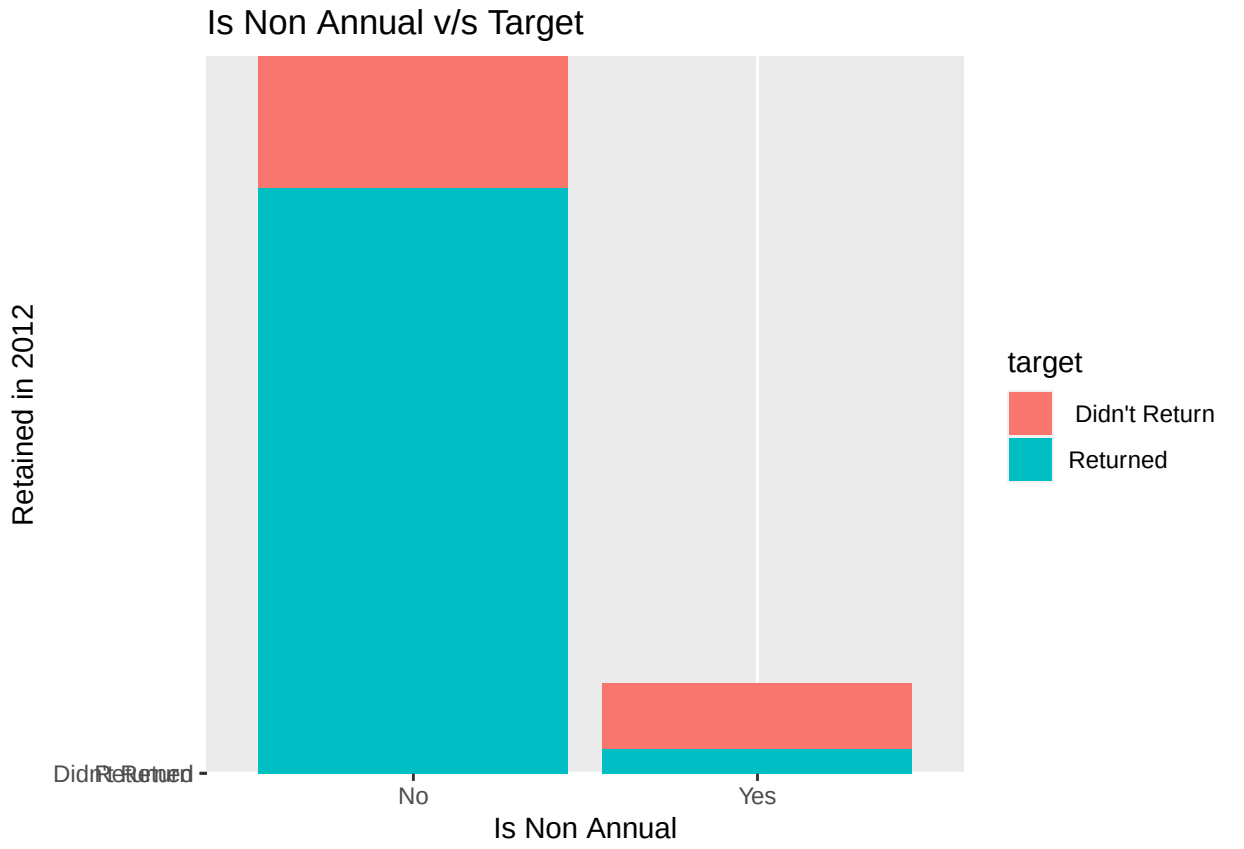
Pie chart to understand the impact of the program being annual on retention

```
pietable <- table(tr$Is.Non.Annual.)
percent <- round(pietable/sum(pietable)*100)
label1 <- paste(names(pietable),percent)
label2 <- paste(label1, "%", sep="")
pie3D(pietable, labels = label2, explode = 0.1,
      main="Distribution of Annual v/s Non-Annual Programs", radius = 1)
```



2. Is.Non.Annual v/s target

```
ggplot(tr, aes(Is.Non.Annual., target, fill = target)) +
geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
labs(title = "Is Non Annual v/s Target", x = "Is Non Annual",
             y= "Retained in 2012")
```

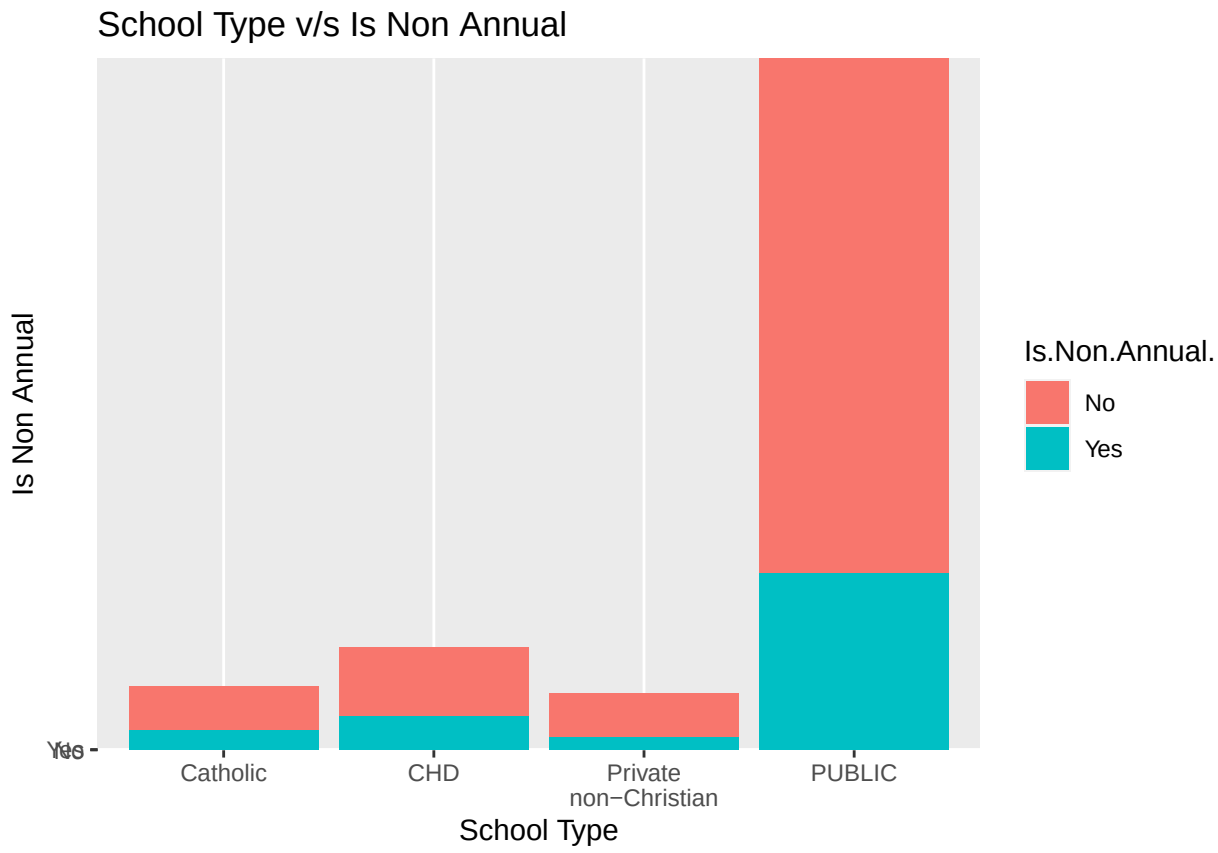


Analysis

Based on the two graphs above, we notice that certain groups within schools tend to repeat the tour yearly while some groups don't. We can see clearly that the group that repeats this annually prefers going back to the travel company and retain them, while the groups not performing these tours annually, tend to not repeat this travel company.

3. SchoolType v/s Is.Non.Annual / Target

```
ggplot(tr, aes(School.Type, Is.Non.Annual., fill = Is.Non.Annual.)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "School Type v/s Is Non Annual", x = "School Type",
         y= "Is Non Annual")
```

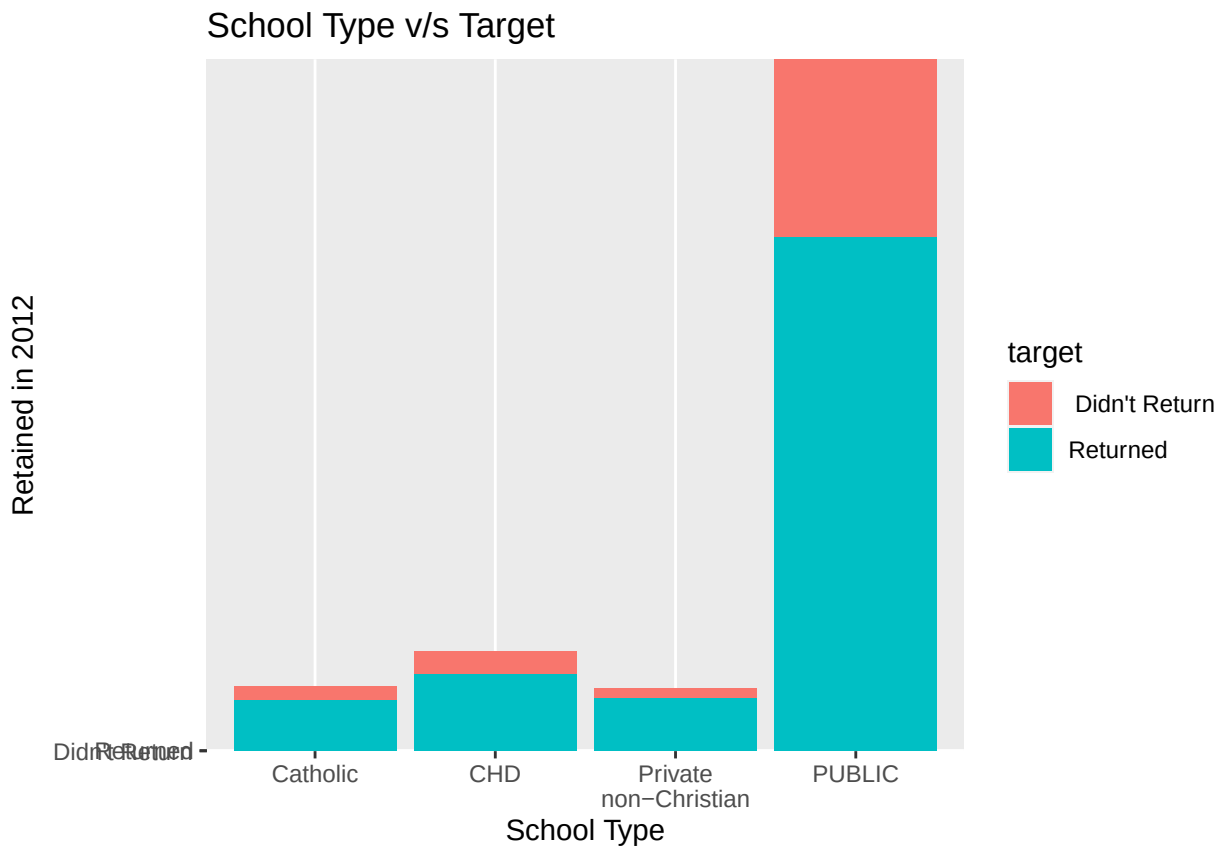


Analysis

In the above graph we notice that out of the different type of schools, Public, Catholic, CHD, and Private (non-christian), the groups within Public schools tend to use Scholastic Travel Company (STC) for tours more than the groups within other types of schools. In fact, they tend to prefer doing these annually v/s every other year.

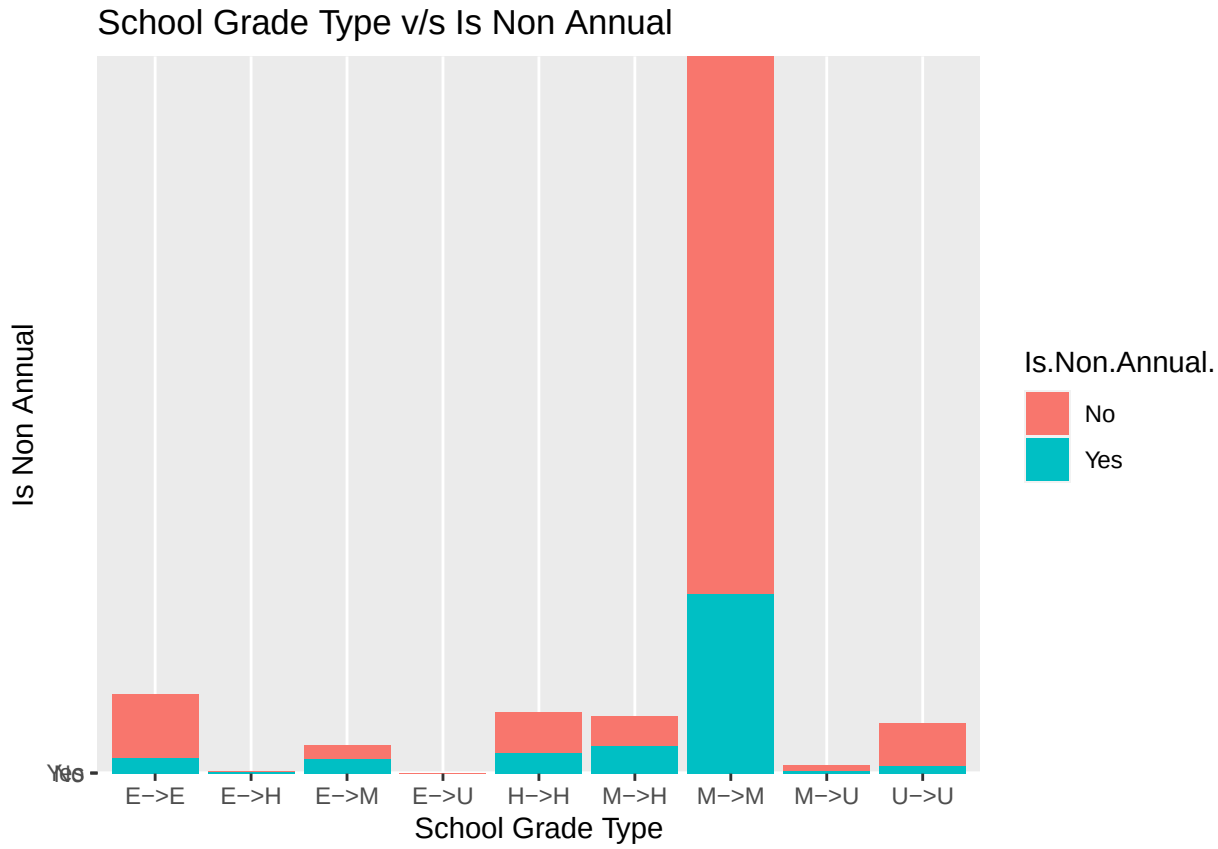
The same theory can also be confirmed with the graph below, where an analysis of School Type v/s target shows us that Public schools return to STC more than any other type of school.

```
ggplot(tr, aes(School.Type, target, fill = target)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "School Type v/s Target", x = "School Type",
         y= "Retained in 2012")
```



4. SchoolGradeType v/s Is.Non.Annual

```
ggplot(tr, aes(SchoolGradeType, Is.Non.Annual., fill = Is.Non.Annual.)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "School Grade Type v/s Is Non Annual", x = "School Grade Type",
         y= "Is Non Annual")
```

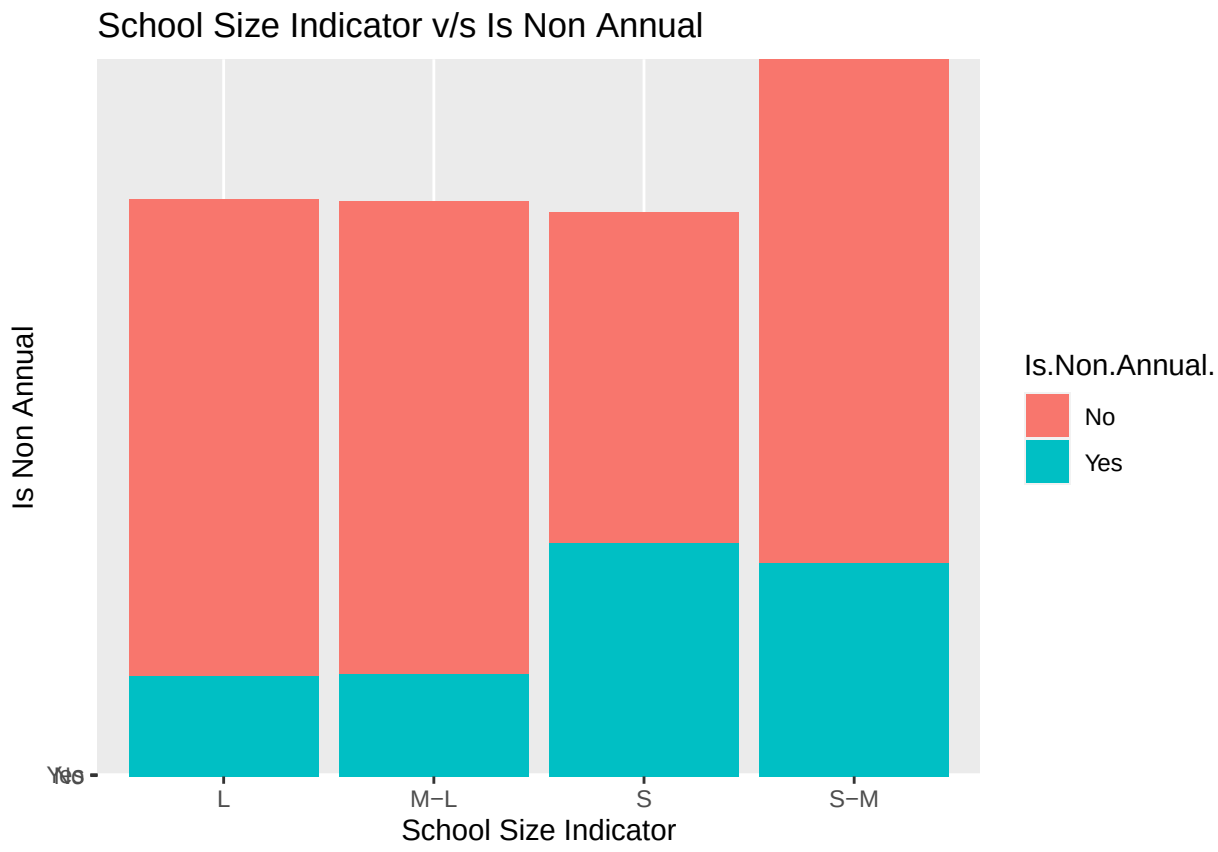


Analysis

Just like the previous analysis, we see that groups from the Middle school grade types repeat the tours annually most frequently with STC compared to other grade types. This theory was also corroborated with the very first analysis we did where we compared SchoolGradeType with the target variable, and noticed that Middle School grade types had higher chances to returning to STC compared to other grade types.

5. SchoolSizeIndicator v/s Is.Non.Annual / Target

```
ggplot(tr, aes(SchoolSizeIndicator, Is.Non.Annual., fill = Is.Non.Annual.)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "School Size Indicator v/s Is Non Annual",
         x = "School Size Indicator",
         y= "Is Non Annual")
```

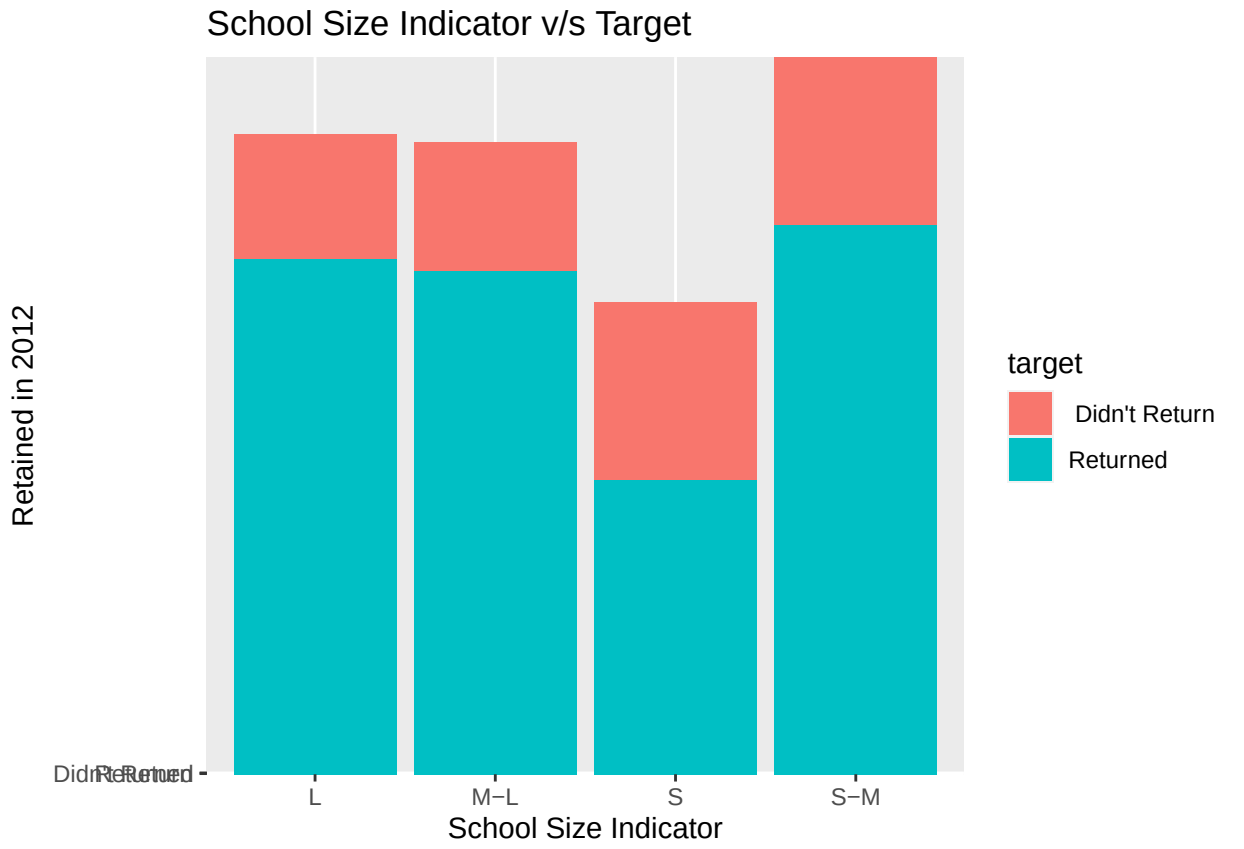


Analysis

Firstly, there are different types of school sizes - Small (S), Large (L), Small to Medium (S-M), and Medium to Large (M-L). Per this analysis, we notice that groups from **Small** sized schools are more likely to go with a tour **every other year** and not **annually**. However, groups from Small to Medium sized schools are more likely to **return to STC annually** compared to other sized schools.

This theory can be further validated in the below graph, where we compare the school size with the target variable. We notice that **Small sized schools did not return to STC as frequently** as the **Small to Medium** sized schools, who returned the most to STC, simply because they wanted the tours to be annual.

```
ggplot(tr, aes(SchoolSizeIndicator, target, fill = target)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "School Size Indicator v/s Target",
         x = "School Size Indicator",
         y= "Retained in 2012")
```

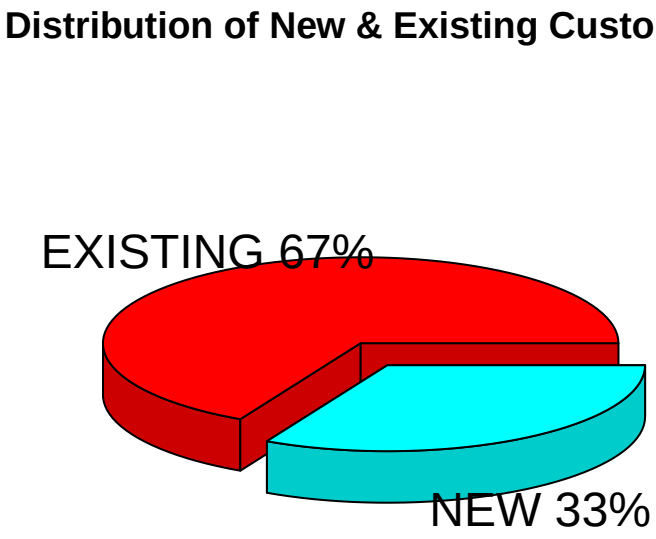


6. SPR.New.Existing v/s Target

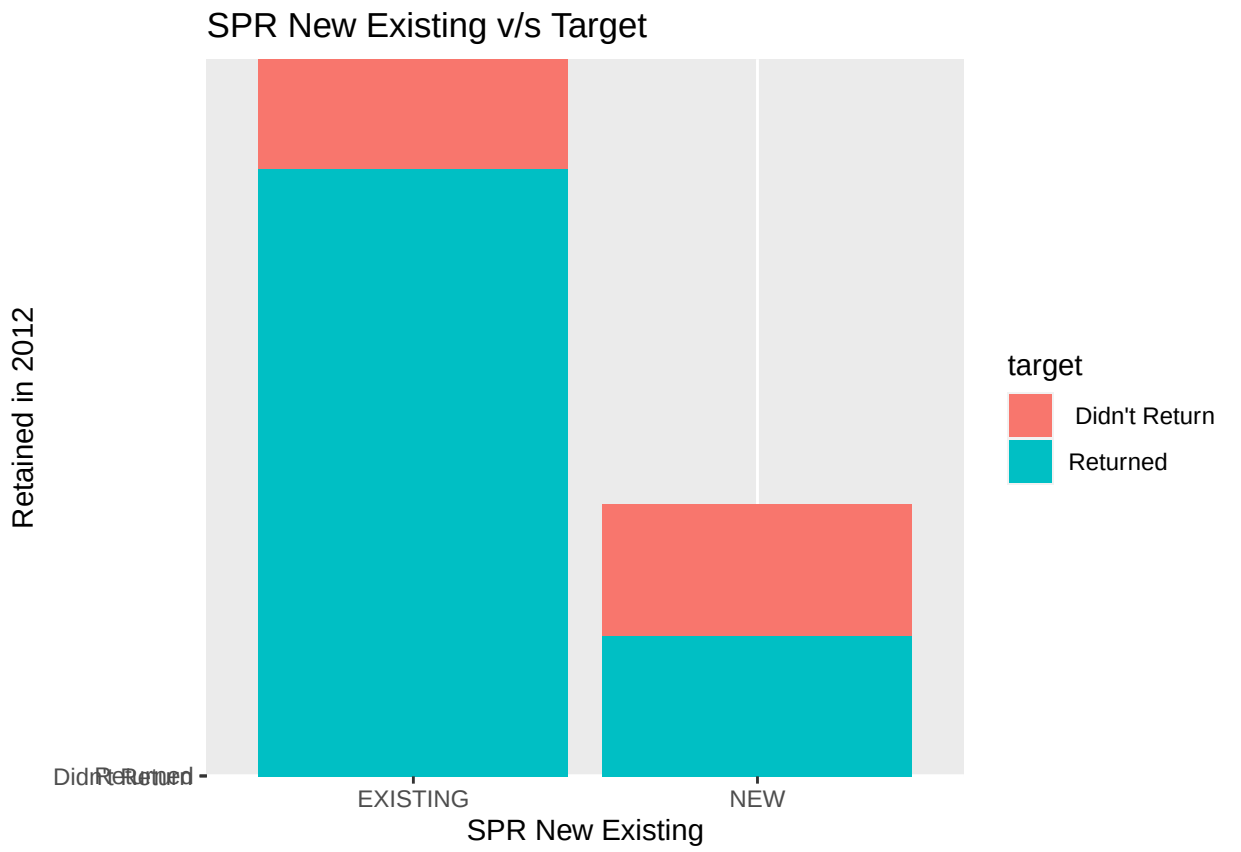
Pie Chart

In order to better understand the distribution of existing v/s new groups, we use the pie chart as we've done in above examples:

```
pietable <- table(tr$SPR.New.Existing)
percent <- round(pietable/sum(pietable)*100)
label1 <- paste(names(pietable),percent)
label2 <- paste(label1, "%", sep="")
pie3D(pietable, labels = label2, explode = 0.1,
      main="Distribution of New & Existing Customers", radius = 1)
```



```
ggplot(tr, aes(SPR.New.Existing, target, fill = target)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "SPR New Existing v/s Target",
         x = "SPR New Existing",
         y= "Retained in 2012")
```



Analysis

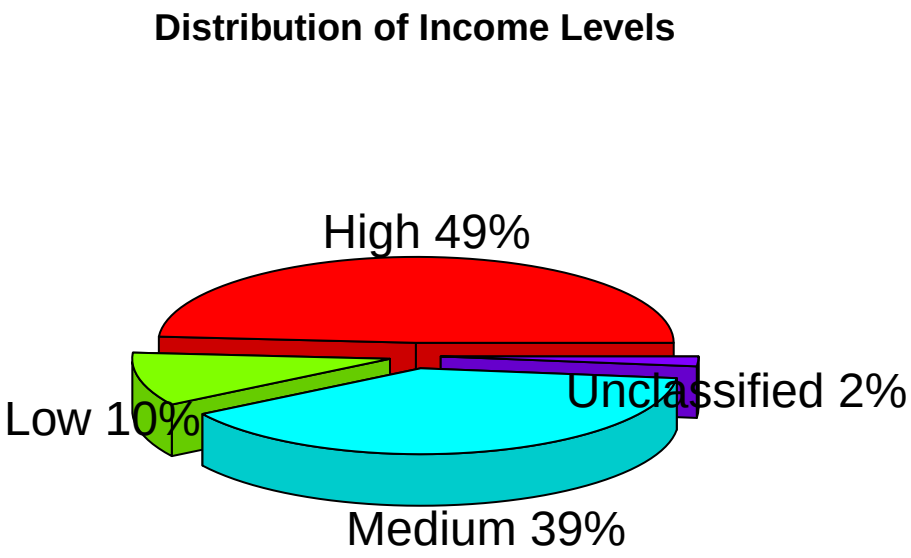
Firstly, we notice that there are 67% existing groups / customers for STC while 33% (with some exceptions) are new. This is also seen in the next graph, where we compare this metric with the target variable. In the second graph, we notice that existing customers returned the most to STC compared to new customers, which shows that existing customers were loyal to the travel company.

7. Income.Level v/s Poverty.Code / Target

Pie chart

In order to better understand the distribution of parents' income level, we use the pie chart as we've done in above examples. From the below graph, We notice that majority of the parents fall in the **"high"** income range, followed by the **"medium"** income range.

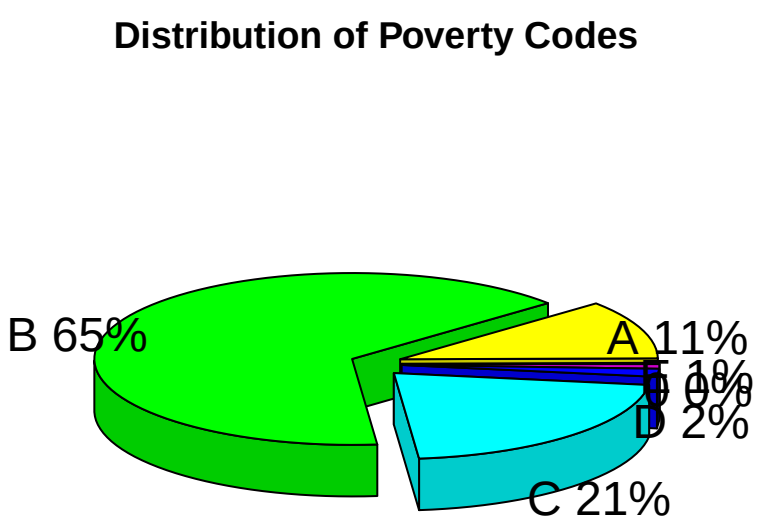
```
pietable <- table(tr$Income.Level)
percent <- round(pietable/sum(pietable)*100)
label1 <- paste(names(pietable),percent)
label2 <- paste(label1, "%", sep="")
pie3D(pietable, labels = label2, explode = 0.1,
      main="Distribution of Income Levels", radius = 1)
```



In order to see the distribution of these income level in relation to the poverty codes, we first look at the distribution of the codes. Please note the following distribution of poverty codes per the data set:

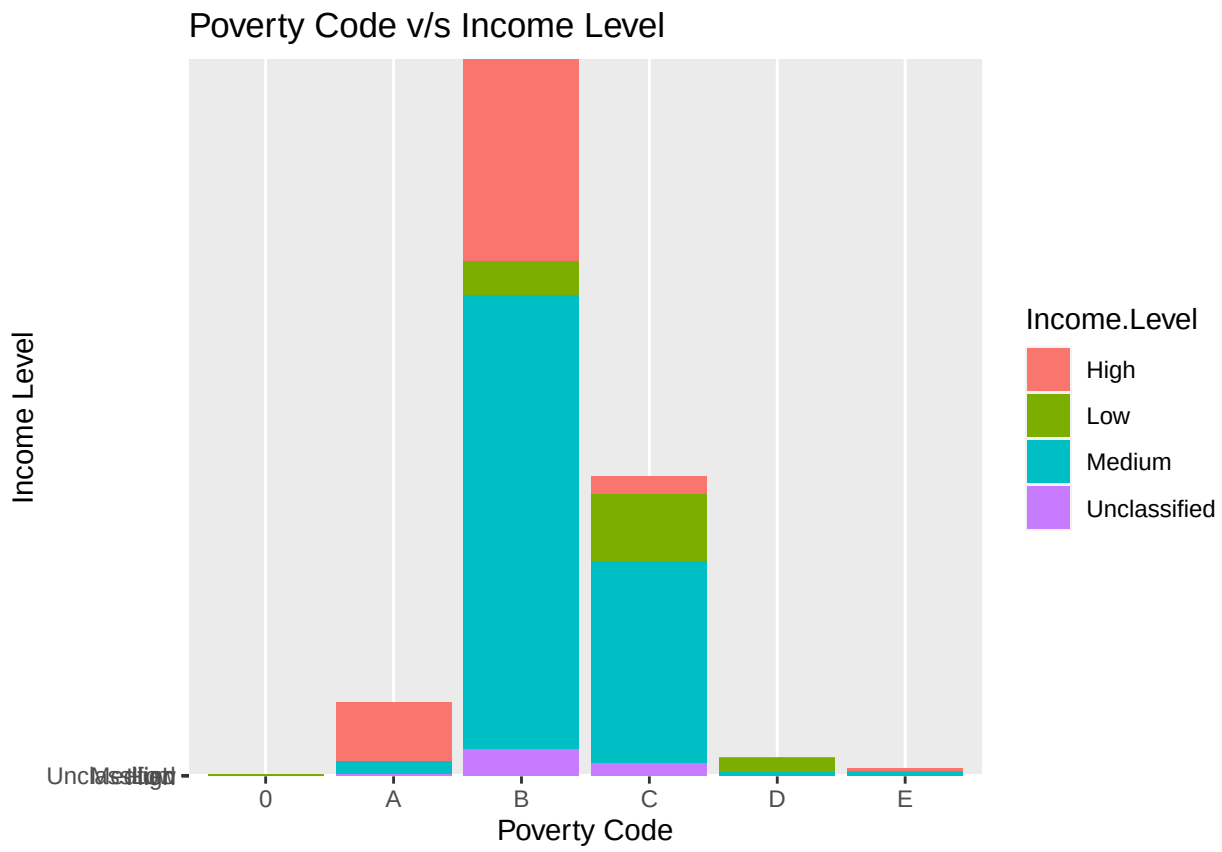
1. A -> 0 to 5.9
2. B -> 6 to 15.9
3. C -> 16 to 30.9
4. D -> 31 or more
5. E -> Unclassified

```
pietable <- table(tr$Poverty.Code)
percent <- round(pietable/sum(pietable)*100)
label1 <- paste(names(pietable),percent)
label2 <- paste(label1, "%", sep="")
pie3D(pietable, labels = label2, explode = 0.1,
      main="Distribution of Poverty Codes", radius = 1)
```



In the above graph, we notice that the highest chunk of poverty code falls in B, with 65%, followed by C and then A. Unclassified or E is just 1%.

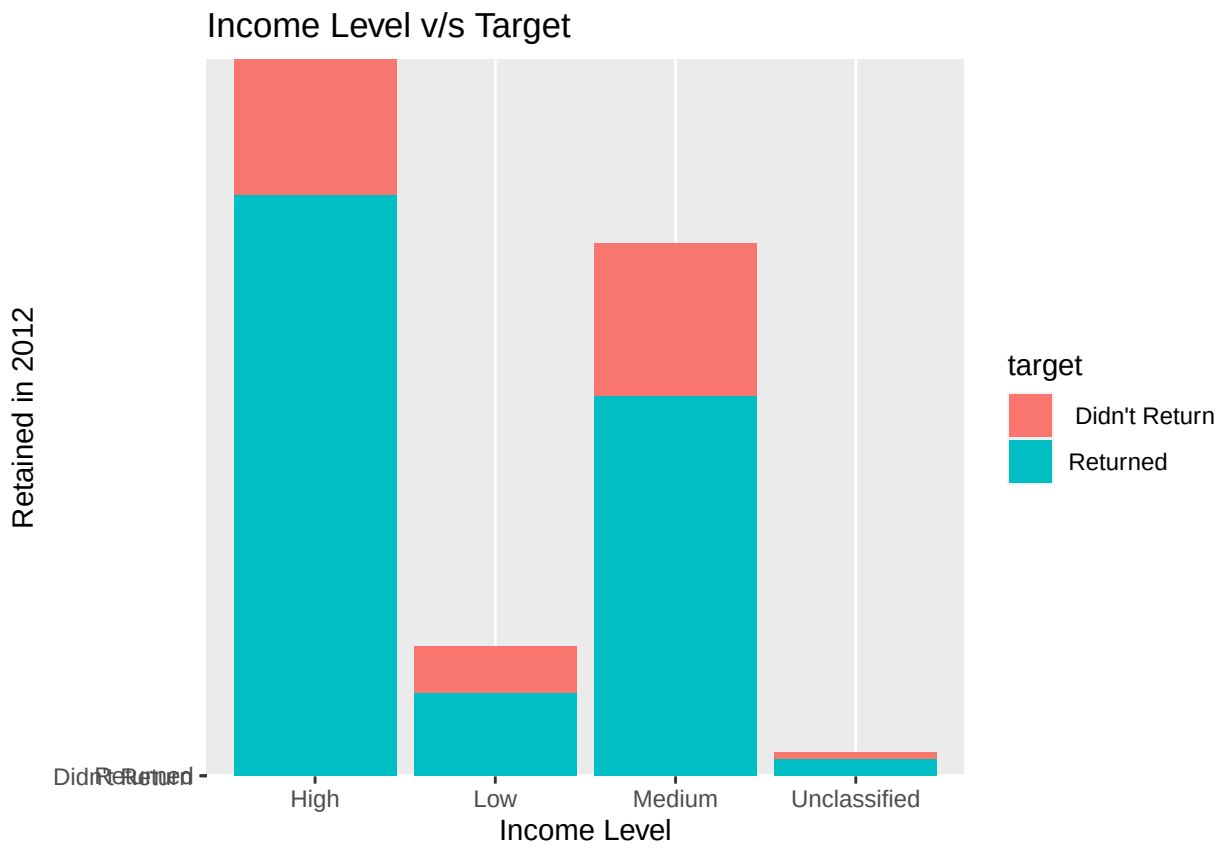
```
ggplot(tr, aes(Poverty.Code, Income.Level, fill = Income.Level)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Poverty Code v/s Income Level",
         x = "Poverty Code",
         y= "Income Level")
```



Analysis

In the below graph, we notice that poverty code B has the most distribution of **medium** and **high** salary ranges, among other salary types. This means that parents earning high or medium salary normally fall in areas with a poverty level of 6 to 15.9, which isn't too bad. As expected, areas with poverty level of 0 to 5.9, which is denoted by A, have a lot of parents with "high" salaries but very few within the medium and low ranges.

```
ggplot(tr, aes(Income.Level, target, fill = target)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Income Level v/s Target",
         x = "Income Level",
         y= "Retained in 2012")
```

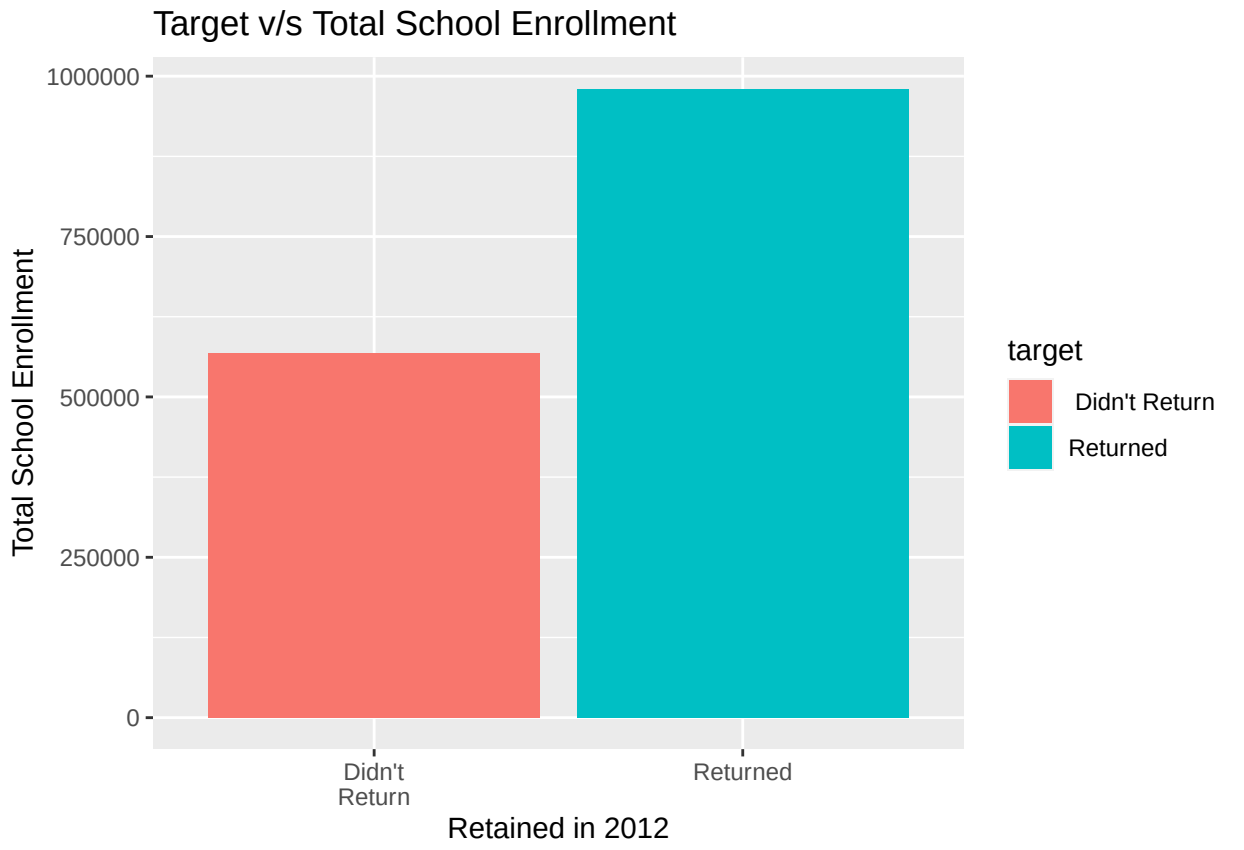


Analysis

In the above graph, we can finally conclude that those parents with a **high** or **medium** salary, are the ones who retain the travel company and continue paying for the tours for their children. However, the ones with low or unclassified salaries are less likely to utilize the travel company, which could mean that the price is quite high for those families, for them to afford the same.

8. Target v/s Total.School.Enrollment

```
ggplot(tr, aes(target, Total.School.Enrollment, fill = target)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Target v/s Total School Enrollment",
         x = "Retained in 2012",
         y= "Total School Enrollment")
```



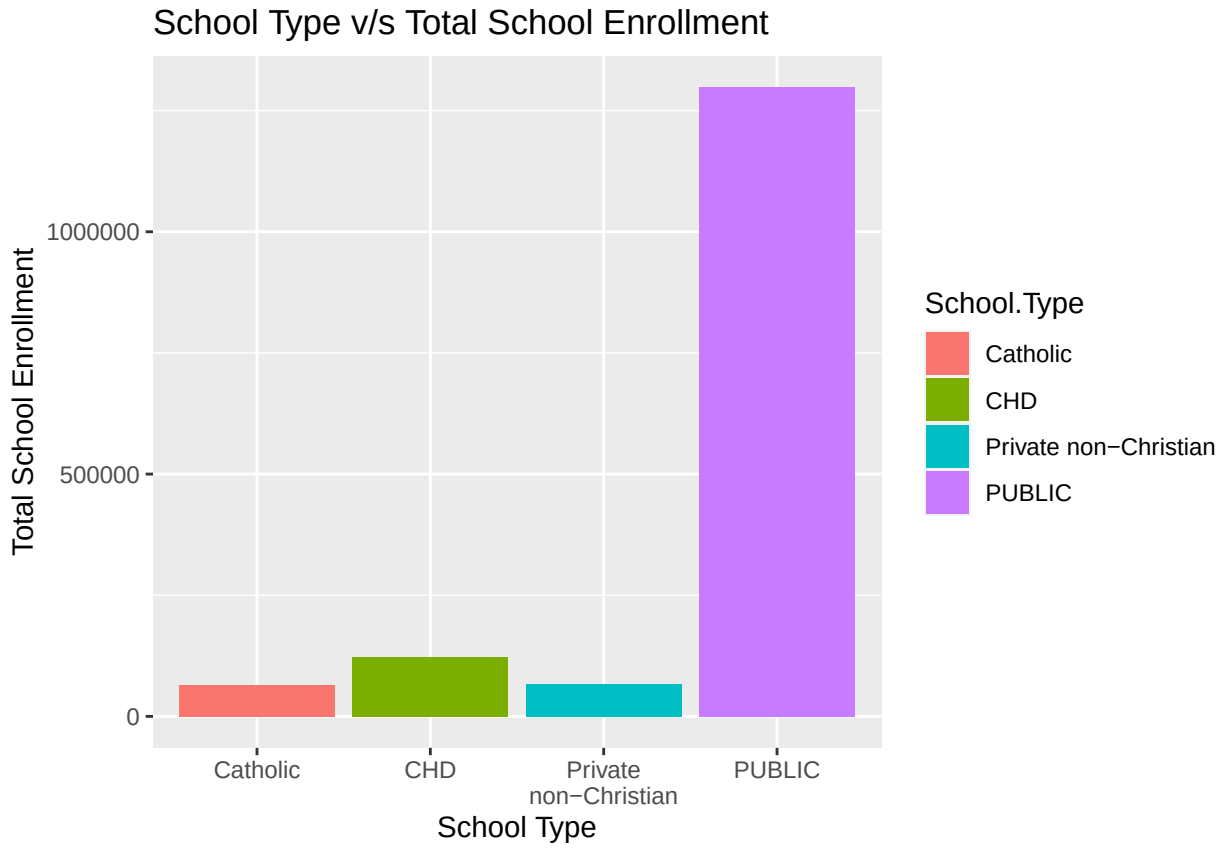
Analysis

In the above graph, we notice that those schools that have more student enrollments (big schools), returned to the travel company compared to those schools that didn't have enough enrollment (small schools).

9. School.Type v/s Total.School.Enrollment

To see the enrollments based on the school type, we use the following code:

```
options(scipen = 5)
ggplot(tr, aes(School.Type, Total.School.Enrollment, fill = School.Type)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "School Type v/s Total School Enrollment",
         x = "School Type",
         y= "Total School Enrollment")
```

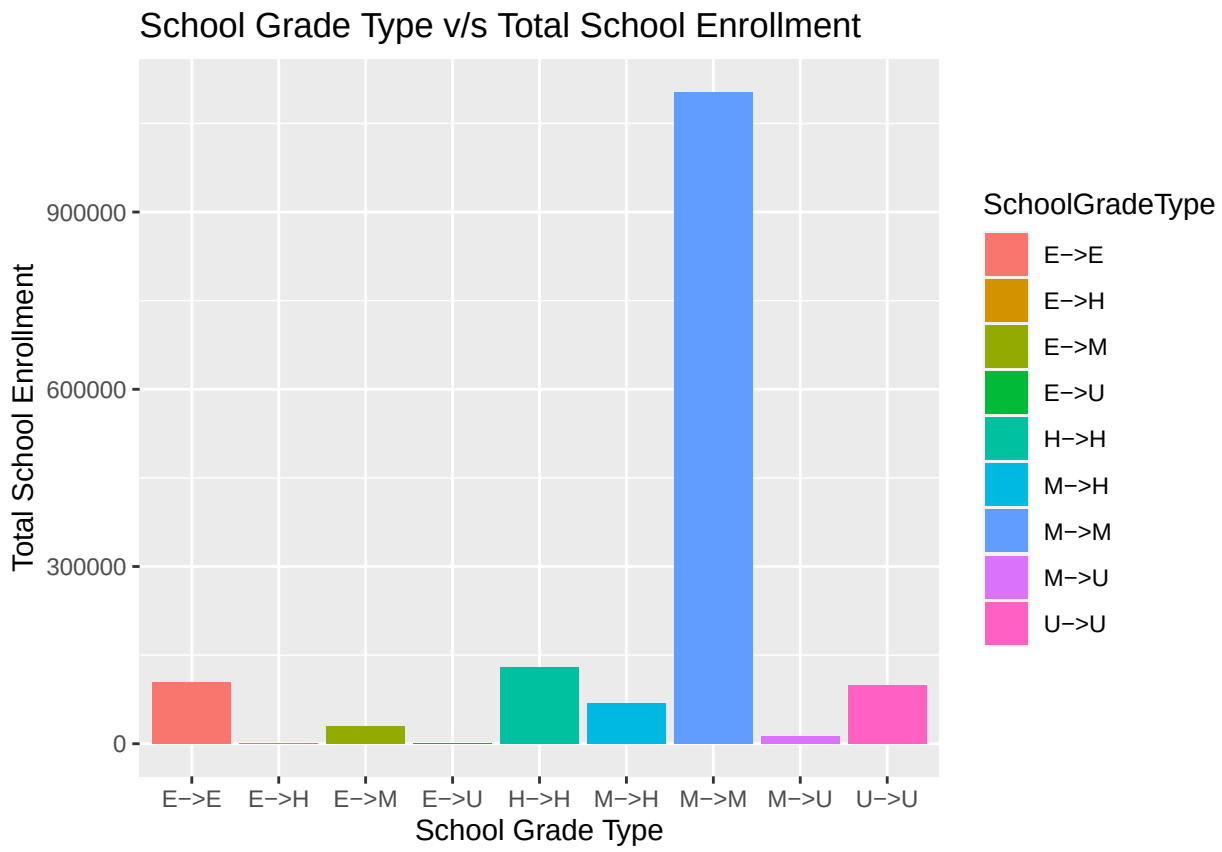


Analysis

Based on the above graph, we notice that public schools in the area have the most student enrollment compared to the private and catholic schools. However, we also see a decent spike in CHD school, which is likely private.

10. SchoolGradeType v/s Total.School.Enrollment

```
options(scipen = 5)
ggplot(tr, aes(SchoolGradeType, Total.School.Enrollment, fill = SchoolGradeType)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "School Grade Type v/s Total School Enrollment",
         x = "School Grade Type",
         y= "Total School Enrollment")
```

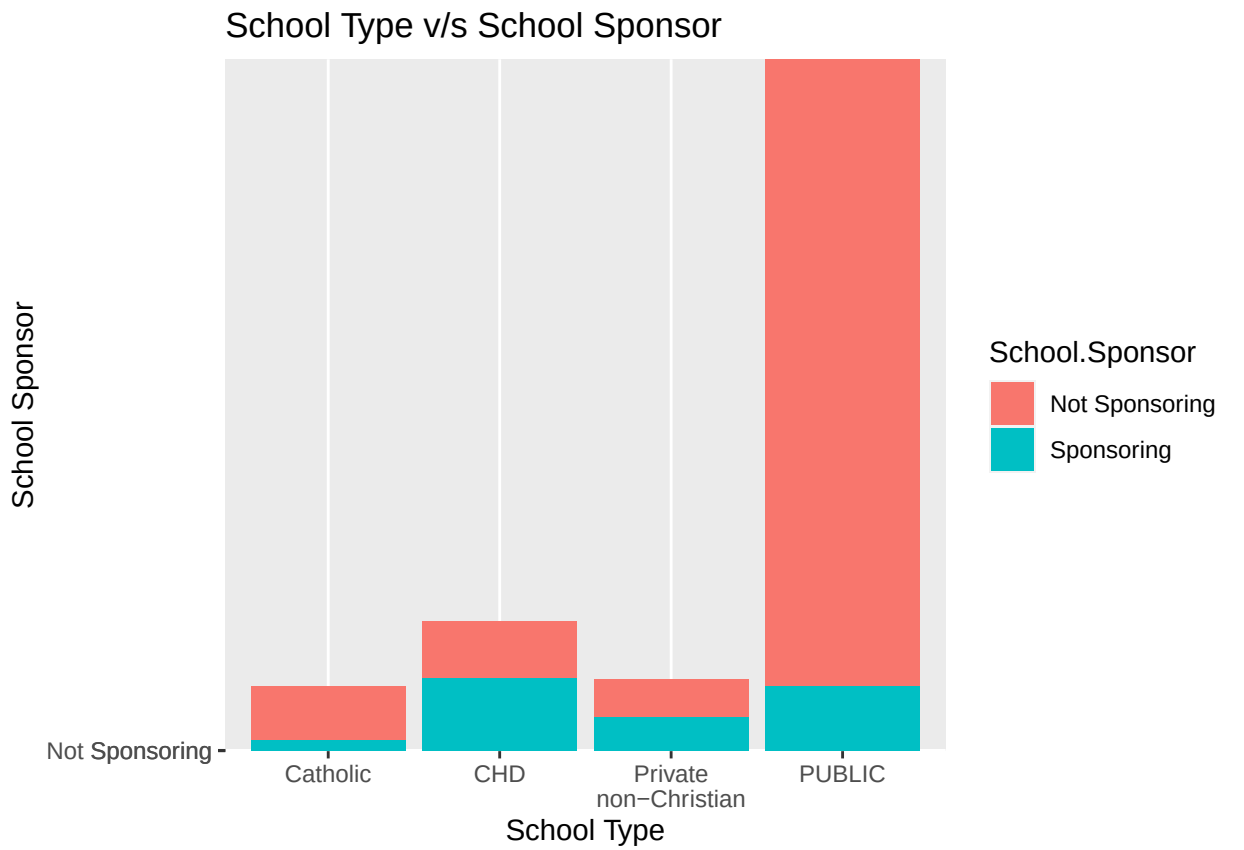


Analysis

In the above graph, we can see that at a grade type level, most enrollments are in the middle->middle schools compared to others. The next best is High->High school.

11. School Sponsor v/s School Type

```
ggplot(tr, aes(School.Type, School.Sponsor, fill = School.Sponsor)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "School Type v/s School Sponsor",
         x = "School Type",
         y= "School Sponsor")
```



Analysis

Per the above graph and analysis, we can see that **Public** schools do not sponsor the tours with the travel company. However, we noticed from the above graphs that they tend to return to those tours annually, which means **Public schools are asking parents to sponsor majority of the trips v/s sponsoring it themselves**. On the other hand, **Private and CHD** schools are sponsoring more trips for their students, followed by Public schools, which makes sense as private schools draw more money compared to public schools, which are generally free.

Summary of Exploratory Data Analysis

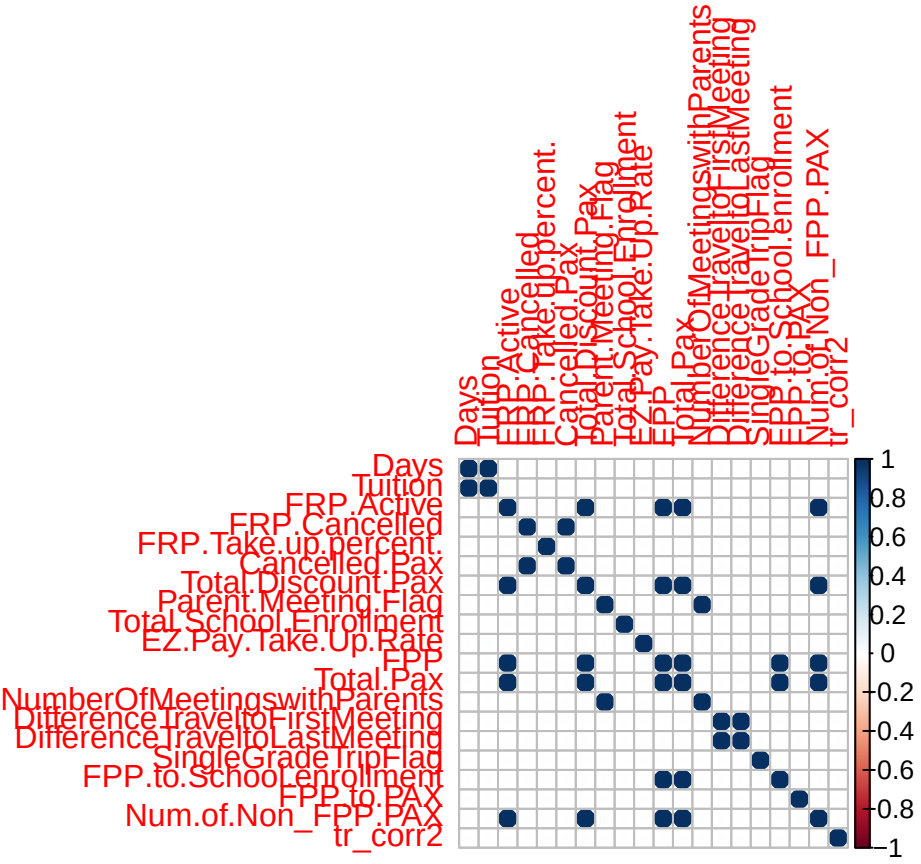
1. Within the SchoolGradeType variable, highest distribution is of the **Middle->Middle** grade types at **72%**
2. **"Middle->Middle School"** grade type return to the Scholastic Travel Company (STC) more frequently compared to other grade types, with the next best being **Elementary->Elementary** grade type
3. **85%** of the programs / tours are held annually while **15%** of them happen every other year
4. The groups that prefer to **hold tours annually**, return to Scholastic Travel Company (STC) compared to the group that **doesn't prefer** annual tours
5. **Public** schools tend to go for the most tours compared to **Private** schools, while also using the travel company for **annual** tours v/s every other year
6. **Middle->Middle** school grade types return the most to STC compared to other grade types, especially **annual** tours
7. **Small** sized schools prefer to go for tours with STC every other year and not annually. However, **Small to Medium** schools go more frequently for annual tours with STC than other sized schools.
8. **67%** of the customers / groups are existing while **33%** are new for Scholastic Travel Company (STC)
9. The **existing** customers returned the most to STC compared to **new** customers.
10. **49%** of the parents fall in the **High** salary category, followed by **39%** in the **Medium** salary category
11. **65%** of the areas fall under Poverty Code of **B**, while **21%** fall under code **C**, followed by **11%** in code **A**
12. Areas under poverty code **B** has the most distribution of **High** and **Medium** salaries, while areas under poverty code **A** have a lot of **High** salaried parents compared to other income levels
13. Parents with **High** and **Medium** salaries returned the most to STC for tours and continued paying annually compared to parents with **Low** and **Unclassified** salaries
14. Schools with the **high student enrollments** returned to STC compared to ones with **low student enrollments**
15. The highest student enrollment is in **Public** schools compared to **Private** schools
16. Out of those public schools, the most enrollment is in **Middle->Middle** school grade types, followed by **High->High** grade type
17. Public schools are **not sponsoring** most tours for their students and likely depending on parents to pay for the tour. **Private** schools are **sponsoring** slightly more than the public schools

Correlation

In order to show the correlation between the numeric variables, we do the following:

```
library(corrplot)
tr_corr1 <- select_if(tr, is.numeric)# to select only numeric variables in tr
tr_corr2 <- as.numeric(tr$target)# converts target into numeric variable
tr_corr3 <- cbind(tr_corr1, tr_corr2)# binds the above two variables
```

```
corrplot(round(cor(tr_corr3)), digits = 1) #draws the correlation plot
```



We can see from the above correlation matrix that highly correlated variables are:

1. FRP.Active
2. FPP
3. Total.Discount.Pax
4. Total.Pax
5. Num.of.Non_FPP.PAX

Predictive Models

Decision Tree

Decision Tree with all variables and default cp

We start first with the Decision tree model, for which we need to partition the data set into train and test data. In this iteration, we're taking the target variable and comparing it to all the input variables, and also going with a default cp of 0.1.

```
set.seed(30)
index_dt1 <-  sample(2, nrow(tr) , replace = T , prob = c(0.6 , 0.4))
train_dt1 <- tr[index_dt1 == 1, ]
test_dt1 <- tr[index_dt1 == 2, ]
```

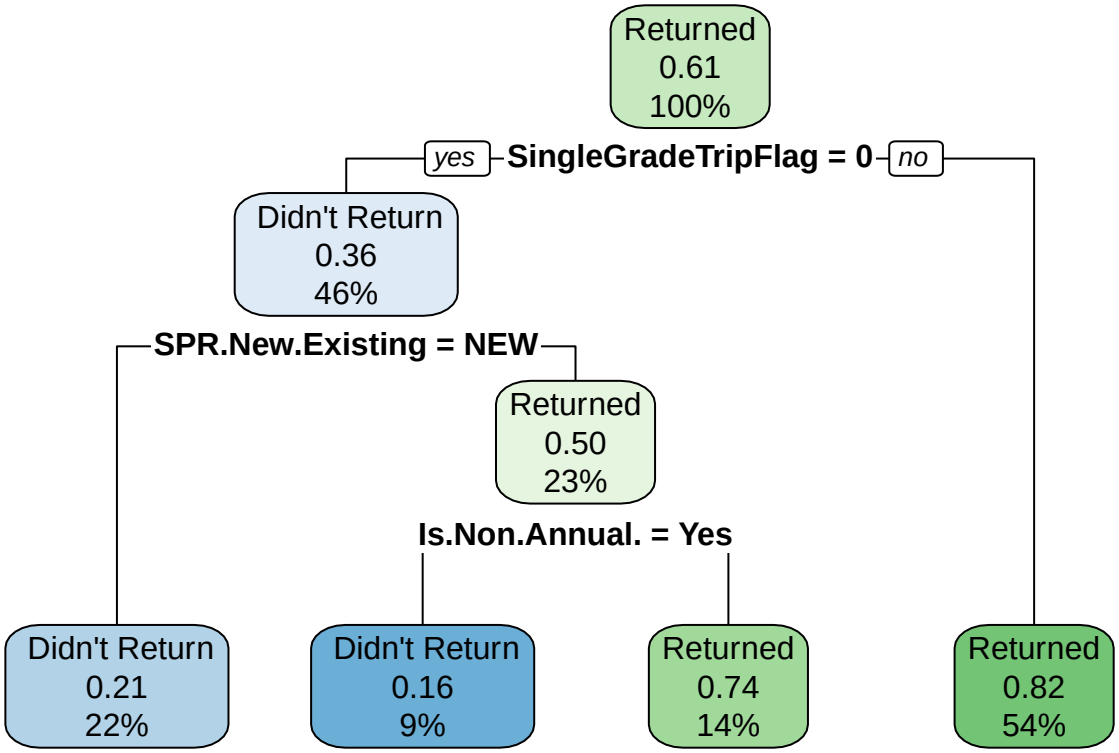
Once we're done partitioning, we proceed to with creating the model and printing the decision tree, its rules, and its summary. This can be done using the below chunk of codes:

```
library(rpart)
tree_model1 <- rpart(target ~., train_dt1)
print(tree_model1) # to print the decision rules
```

```
## n= 1481
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 1481 579 Returned (0.3909521 0.6090479)
##    2) SingleGradeTripFlag< 0.5 674 242  Didn't Return (0.6409496 0.3590504)
##      4) SPR.New.Existing=NEW 330  69  Didn't Return (0.7909091 0.2090909) *
##      5) SPR.New.Existing=EXISTING 344 171 Returned (0.4970930 0.5029070)
##       10) Is.Non.Annual.=Yes 140  23  Didn't Return (0.8357143 0.1642857) *
##       11) Is.Non.Annual.=No 204  54 Returned (0.2647059 0.7352941) *
##    3) SingleGradeTripFlag>=0.5 807 147 Returned (0.1821561 0.8178439) *
```

To plot an rpart decision tree we can use the "rpart.plot()" function from "rpart.plot" package:

```
library(rpart.plot)
rpart.plot(tree_model1)
```

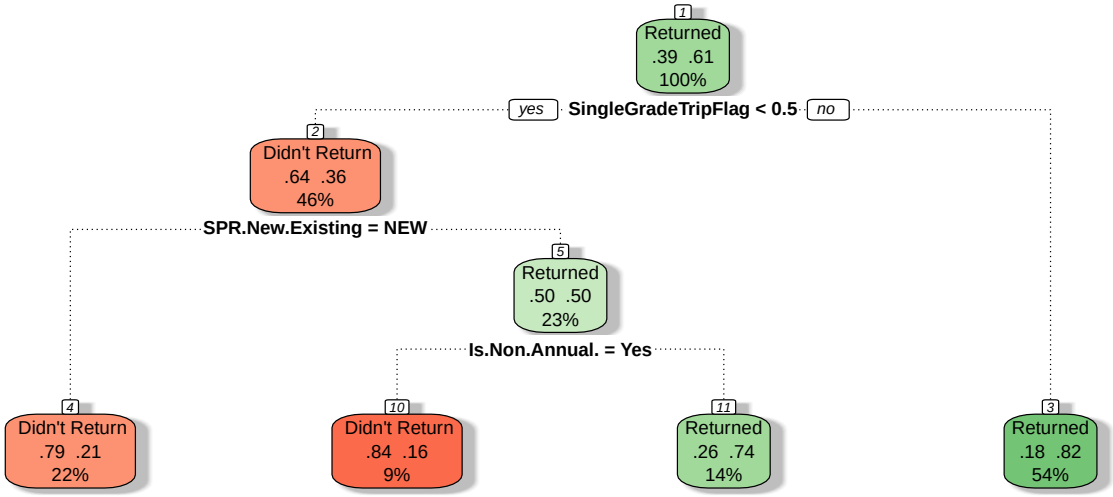



```
rpart.rules(tree_model1)
```

```
##  target
##    0.16 when SingleGradeTripFlag is 0 & SPR.New.Existing is EXISTING & Is.Non.Annual. is Yes
##    0.21 when SingleGradeTripFlag is 0 & SPR.New.Existing is      NEW
##    0.74 when SingleGradeTripFlag is 0 & SPR.New.Existing is EXISTING & Is.Non.Annual. is  No
##    0.82 when SingleGradeTripFlag is 1
```

In order to see a more fancier version of rpart.plot, we also have the option of fancyRpartPlot() function, which is part of the rattle library. It can be run as follows:

```
library(rattle)
fancyRpartPlot(tree_model1, palettes=c("Reds", "Greens"), sub="")
```



To obtain the error rate for the training and test data, we run the following sets of commands:

```
#Error rate for training data

pred_train1 <- predict(tree_model1, data = train_dt1, type = "class")
mean(train_dt1$target != pred_train1)
```

```
## [1] 0.1978393
```

```
# Error rate for the test data

pred_test1 <- predict(tree_model1, data1 = test_dt1, type = "class")
mean(test_dt1$target != pred_test1)
```

```
## [1] 0.4476705
```

Final summary is as follows:

```
summary(tree_model1)
```

```
## Call:
## rpart(formula = target ~ ., data = train_dt1)
##   n= 1481
##
##           CP nsplit rel error    xerror       xstd
## 1 0.32815199      0 1.0000000 1.0000000 0.03243295
## 2 0.08290155      1 0.6718480 0.6718480 0.02925028
## 3 0.01000000      3 0.5060449 0.5060449 0.02647805
##
## Variable importance
##      SingleGradeTripFlag               From.Grade           Is.Non.Annual.
##                       26                       17                       16
##         SPR.New.Existing          SchoolGradeType      SchoolGradeTypeHigh
##                       12                       10                        8
##           GroupGradeType  Total.School.Enrollment      SchoolSizeIndicator
##                        2                        1                        1
## FPP.to.School.enrollment            MDR.Low.Grade                      FPP
##                        1                        1                        1
##                Total.Pax       Num.of.Non_FPP.PAX       Total.Discount.Pax
##                        1                        1                        1
##               FRP.Active
##                        1
##
## Node number 1: 1481 observations,    complexity param=0.328152
##   predicted class=Returned        expected loss=0.3909521  P(node) =1
##     class counts:   579   902
##    probabilities: 0.391 0.609
##   left son=2 (674 obs) right son=3 (807 obs)
##   Primary splits:
##       SingleGradeTripFlag < 0.5        to the left,  improve=154.61180, (0 missing)
##       From.Grade          splits as  LLLLRLLLRL, improve=117.02960, (0 missing)
##       Is.Non.Annual.      splits as  RL, improve=116.68360, (0 missing)
##       SPR.New.Existing    splits as  RL, improve= 99.50450, (0 missing)
##       Total.Pax           < 25.5       to the left,  improve= 68.60379, (0 missing)
##   Surrogate splits:
##       From.Grade          splits as  LLRLRLLLRL, agree=0.851, adj=0.672, (0 split)
##       SchoolGradeType     splits as  RLLLLLRLL,  agree=0.718, adj=0.381, (0 split)
##       SchoolGradeTypeHigh splits as  RLRL,       agree=0.692, adj=0.323, (0 split)
##       Is.Non.Annual.      splits as  RL,         agree=0.665, adj=0.264, (0 split)
##       SPR.New.Existing    splits as  RL,         agree=0.663, adj=0.260, (0 split)
##
## Node number 2: 674 observations,    complexity param=0.08290155
##   predicted class= Didn't Return  expected loss=0.3590504  P(node) =0.4550979
##     class counts:   432   242
##    probabilities: 0.641 0.359
##   left son=4 (330 obs) right son=5 (344 obs)
##   Primary splits:
##       SPR.New.Existing splits as  RL, improve=29.07994, (0 missing)
##       Is.Non.Annual.   splits as  RL, improve=29.02286, (0 missing)
##       GroupGradeType   splits as  LLRLRLLLRLLLR, improve=15.48832, (0 missing)
```

```
##       MDR.High.Grade   splits as  L-LL-RRRRRRR, improve=13.83927, (0 missing)
##       MDR.Low.Grade    splits as  LR-LRRRRRLLL, improve=12.34334, (0 missing)
##   Surrogate splits:
##       FPP                < 16.5       to the left,  agree=0.614, adj=0.212, (0 split)
##       Total.Pax          < 17.5       to the left,  agree=0.614, adj=0.212, (0 split)
##       Total.Discount.Pax < 1.5        to the left,  agree=0.608, adj=0.200, (0 split)
##       Num.of.Non_FPP.PAX < 1.5        to the left,  agree=0.608, adj=0.200, (0 split)
##       FRP.Active         < 8.5        to the left,  agree=0.604, adj=0.191, (0 split)
##
## Node number 3: 807 observations
##   predicted class=Returned        expected loss=0.1821561  P(node) =0.5449021
##     class counts:   147   660
##    probabilities: 0.182 0.818
##
## Node number 4: 330 observations
##   predicted class= Didn't Return  expected loss=0.2090909  P(node) =0.2228224
##     class counts:   261    69
##    probabilities: 0.791 0.209
##
## Node number 5: 344 observations,    complexity param=0.08290155
##   predicted class=Returned        expected loss=0.497093  P(node) =0.2322755
##     class counts:   171   173
##    probabilities: 0.497 0.503
##   left son=10 (140 obs) right son=11 (204 obs)
##   Primary splits:
##       Is.Non.Annual.          splits as  RL, improve=54.13956, (0 missing)
##       Total.School.Enrollment < 325.5      to the left,  improve=16.39473, (0 missing)
##       SchoolSizeIndicator     splits as  RRLR, improve=16.38189, (0 missing)
##       SchoolGradeTypeHigh     splits as  LLLR, improve=11.45325, (0 missing)
##       SchoolGradeType         splits as  LLLRLLLRR, improve=11.45325, (0 missing)
##   Surrogate splits:
##       GroupGradeType           splits as  RLRLRLLRRRLLR, agree=0.663, adj=0.171, (0 split)
##       Total.School.Enrollment  < 315        to the left,  agree=0.660, adj=0.164, (0 split)
##       SchoolSizeIndicator      splits as  RRLR, agree=0.651, adj=0.143, (0 split)
##       MDR.Low.Grade            splits as  ---RRLRRRLLL, agree=0.648, adj=0.136, (0 split)
##       FPP.to.School.enrollment < 0.05276934 to the right, agree=0.648, adj=0.136, (0 split)
##
## Node number 10: 140 observations
##   predicted class= Didn't Return  expected loss=0.1642857  P(node) =0.09453072
##     class counts:   117    23
##    probabilities: 0.836 0.164
##
## Node number 11: 204 observations
##   predicted class=Returned        expected loss=0.2647059  P(node) =0.1377448
##     class counts:    54   150
##    probabilities: 0.265 0.735
```

Here we notice that one of the key variables being considered by the model is **SingleGradeTripFlag**, which indicates if there was a trip taken by students from the same grade or not. If this variable is false, that is if the students don't belong to the same grade, there's a **55%** chance of groups returning to the Scholastic Travel Company (STC).

Decision tree with "information" split

Here we're going to follow the same steps as above, but instead of taking all input variables for our consideration, we'll only take numeric variables of note.

```
set.seed(30)
index_dt2 <-  sample(2, nrow(tr) , replace = T , prob = c(0.75 , 0.25))
train_dt2 <- tr[index_dt2 == 1, ]
test_dt2 <- tr[index_dt2 == 2, ]
```

Once we're done partitioning, we proceed to with creating the model and printing the decision tree, its rules, and its summary. This can be done using the below chunk of codes:

```
library(rpart)
tree_model2 <- rpart(target ~., train_dt2, parms = list(split = "information"),
control = rpart.control(minbucket = 0, minsplit = 0, cp = 0.03))
print(tree_model2) # to print the decision rules
```

```
## n= 1804
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 1804 715 Returned (0.3963415 0.6036585)
##    2) SingleGradeTripFlag< 0.5 818 296  Didn't Return (0.6381418 0.3618582)
##      4) Is.Non.Annual.=Yes 250  35  Didn't Return (0.8600000 0.1400000) *
##      5) Is.Non.Annual.=No 568 261  Didn't Return (0.5404930 0.4595070)
##       10) SPR.New.Existing=NEW 325  83  Didn't Return (0.7446154 0.2553846) *
##       11) SPR.New.Existing=EXISTING 243  65 Returned (0.2674897 0.7325103) *
##    3) SingleGradeTripFlag>=0.5 986 193 Returned (0.1957404 0.8042596) *
```

To calculate the BestCp, We calculate the optimal xerror by adding min_xerror + min_xstd, which we do as follows:

```
mincp_i_travel <- which.min(tree_model2$cptable[, 'xerror'])
optError_travel <- tree_model2$cptable[mincp_i_travel, "xerror"] +
tree_model2$cptable[mincp_i_travel, "xstd"]
```

After this, we find the row(index) of the xerror value which is closest to optError calculated above, using the following code:

```
optCP_i_travel <-
which.min(abs(tree_model2$cptable[,"xerror"] - optError_travel))
```

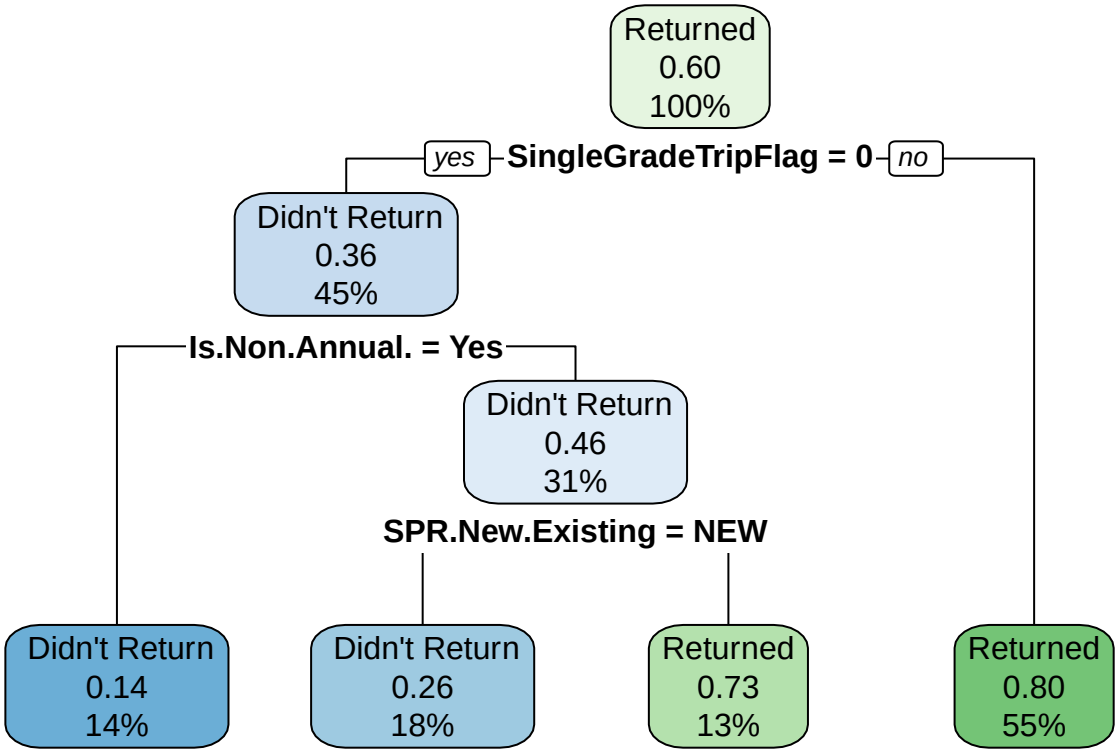
Finally, to get the best CP, we find the cp value corresponding to optCP_i calculated above:

```
optCP_travel <- tree_model2$cptable[optCP_i_travel, "CP"]
print(optCP_travel)
```

```
## [1] 0.03
```

To plot an rpart decision tree we can use the "rpart.plot()" function from "rpart.plot" package:

```
library(rpart.plot)
rpart.plot(tree_model2)
```

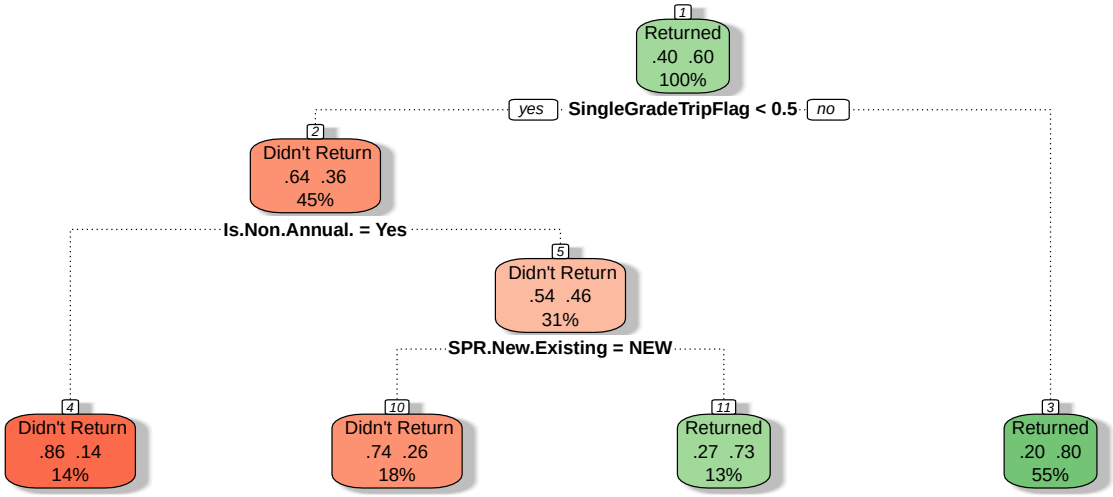



```
rpart.rules(tree_model2)
```

```
##  target
##    0.14 when SingleGradeTripFlag is 0 & Is.Non.Annual. is Yes
##    0.26 when SingleGradeTripFlag is 0 & Is.Non.Annual. is  No & SPR.New.Existing is      NEW
##    0.73 when SingleGradeTripFlag is 0 & Is.Non.Annual. is  No & SPR.New.Existing is EXISTING
##    0.80 when SingleGradeTripFlag is 1
```

In order to see a more fancier version of rpart.plot, we also have the option of fancyRpartPlot() function, which is part of the rattle library. It can be run as follows:

```
library(rattle)
fancyRpartPlot(tree_model2, palettes=c("Reds", "Greens"), sub="")
```



To obtain the error rate for the training and test data, we run the following sets of commands:

```
#Error rate for training data

pred_train2 <- predict(tree_model2, data = train_dt2, type = "class")
mean(train_dt2$target != pred_train2)
```

```
## [1] 0.2084257
```

```
# Error rate for the test data

tree_model2_pruned <- prune(tree_model2, cp = optCP_travel)

pred_test2 <- predict(tree_model2_pruned, data1 = test_dt2, type = "class")
mean(test_dt2$target != pred_test2)
```

```
## [1] 0.4534368
```

Final summary is as follows:

```
summary(tree_model2)
```

```
## Call:
## rpart(formula = target ~ ., data = train_dt2, parms = list(split = "information"),
##     control = rpart.control(minbucket = 0, minsplit = 0, cp = 0.03))
##   n= 1804
##
##           CP nsplit rel error    xerror       xstd
## 1 0.31608392      0 1.0000000 1.0000000 0.02905646
## 2 0.07902098      1 0.6839161 0.6839161 0.02640542
## 3 0.03000000      3 0.5258741 0.5258741 0.02412869
##
## Variable importance
## SingleGradeTripFlag          From.Grade    SPR.New.Existing      Is.Non.Annual.
##                  27                  18                  17                  13
##     SchoolGradeType SchoolGradeTypeHigh         CRM.Segment          FRP.Active
##                  10                   9                   1                   1
##      GroupGradeType  Num.of.Non_FPP.PAX  Total.Discount.Pax
##                   1                   1                   1
##
## Node number 1: 1804 observations,    complexity param=0.3160839
##   predicted class=Returned        expected loss=0.3963415  P(node) =1
##     class counts:   715  1089
##    probabilities: 0.396 0.604
##   left son=2 (818 obs) right son=3 (986 obs)
##   Primary splits:
##       SingleGradeTripFlag < 0.5   to the left,   improve=188.50400, (0 missing)
##       Is.Non.Annual.      splits as  RL,         improve=150.89090, (0 missing)
##       From.Grade          splits as  LLLLRLLLRL, improve=146.80510, (0 missing)
##       SPR.New.Existing    splits as  RL,         improve=114.96120, (0 missing)
##       FPP                 < 24.5  to the left,   improve= 92.15013, (0 missing)
##   Surrogate splits:
##       From.Grade          splits as  LLRLRLLLRL, agree=0.848, adj=0.664, (0 split)
##       SchoolGradeType     splits as  RLLLLLRLL,  agree=0.722, adj=0.386, (0 split)
##       SchoolGradeTypeHigh splits as  RLRL,       agree=0.696, adj=0.329, (0 split)
##       SPR.New.Existing    splits as  RL,         agree=0.666, adj=0.263, (0 split)
##       Is.Non.Annual.      splits as  RL,         agree=0.663, adj=0.257, (0 split)
##
## Node number 2: 818 observations,    complexity param=0.07902098
##   predicted class= Didn't Return  expected loss=0.3618582  P(node) =0.4534368
##     class counts:   522   296
##    probabilities: 0.638 0.362
##   left son=4 (250 obs) right son=5 (568 obs)
##   Primary splits:
##       Is.Non.Annual.          splits as  RL, improve=42.28076, (0 missing)
##       SPR.New.Existing        splits as  RL, improve=32.54186, (0 missing)
##       GroupGradeType          splits as  LLRLRLLLRLLLR, improve=19.28068, (0 missing)
##       Total.School.Enrollment < 341   to the left,  improve=18.07152, (0 missing)
##       MDR.High.Grade          splits as  LRLL-RRRLRRR, improve=17.21145, (0 missing)
##   Surrogate splits:
##       School.Sponsor                  splits as  RL,         agree=0.702, adj=0.024, (0 split)
##       DifferenceTraveltoFirstMeeting < 439.5 to the right,  agree=0.702, adj=0.024, (0 split)
```

```
##       MajorProgramCode               splits as  LRRR,       agree=0.702, adj=0.024, (0 split)
##       From.Grade                     splits as  RL-LRRRRRR, agree=0.699, adj=0.016, (0 split)
##       Total.School.Enrollment        < 88    to the left,   agree=0.699, adj=0.016, (0 split)
##
## Node number 3: 986 observations
##   predicted class=Returned        expected loss=0.1957404  P(node) =0.5465632
##     class counts:   193   793
##    probabilities: 0.196 0.804
##
## Node number 4: 250 observations
##   predicted class= Didn't Return  expected loss=0.14  P(node) =0.1385809
##     class counts:   215    35
##    probabilities: 0.860 0.140
##
## Node number 5: 568 observations,    complexity param=0.07902098
##   predicted class= Didn't Return  expected loss=0.459507  P(node) =0.3148559
##     class counts:   307   261
##    probabilities: 0.540 0.460
##   left son=10 (325 obs) right son=11 (243 obs)
##   Primary splits:
##       SPR.New.Existing splits as  RL,           improve=66.06511, (0 missing)
##       FPP              < 21.5  to the left,     improve=23.44394, (0 missing)
##       FRP.Active       < 18.5  to the left,     improve=23.21982, (0 missing)
##       Total.Pax        < 24.5  to the left,     improve=22.92100, (0 missing)
##       MDR.High.Grade   splits as  -RLL-RRRRRRR, improve=19.93734, (0 missing)
##   Surrogate splits:
##       FRP.Active         < 18.5  to the left,  agree=0.625, adj=0.123, (0 split)
##       CRM.Segment        splits as  LLLLRLRLLLL, agree=0.625, adj=0.123, (0 split)
##       GroupGradeType     splits as  L-RLRLLLRLLLL, agree=0.618, adj=0.107, (0 split)
##       Total.Discount.Pax < 2.5   to the left,  agree=0.614, adj=0.099, (0 split)
##       Num.of.Non_FPP.PAX < 2.5   to the left,  agree=0.614, adj=0.099, (0 split)
##
## Node number 10: 325 observations
##   predicted class= Didn't Return  expected loss=0.2553846  P(node) =0.1801552
##     class counts:   242    83
##    probabilities: 0.745 0.255
##
## Node number 11: 243 observations
##   predicted class=Returned        expected loss=0.2674897  P(node) =0.1347007
##     class counts:    65   178
##    probabilities: 0.267 0.733
```

Decision tree with "gini" split and cp = 0

Here we're going to follow the same steps as above, but instead of taking all input variables for our consideration, we'll only take numeric variables of note.

```
set.seed(30)
index_dt3 <-  sample(2, nrow(tr) , replace = T , prob = c(0.75 , 0.25))
train_dt3 <- tr[index_dt3 == 1, ]
test_dt3 <- tr[index_dt3 == 2, ]
```

Once we're done partitioning, we proceed to with creating the model and printing the decision tree, its rules, and its summary. This can be done using the below chunk of codes:

```
library(rpart)
tree_model3 <- rpart(target ~., train_dt3, parms = list(split = "gini"),
control = rpart.control(minbucket = 6, minsplit = 6, cp = 0))
```

To calculate the BestCp, We calculate the optimal xerror by adding min_xerror + min_xstd, which we do as follows:

```
mincp_i_travel1 <- which.min(tree_model3$cptable[, 'xerror'])
optError_travel1 <- tree_model3$cptable[mincp_i_travel1, "xerror"] +
tree_model3$cptable[mincp_i_travel1, "xstd"]
```

After this, we find the row(index) of the xerror value which is closest to optError calculated above, using the following code:

```
optCP_i_travel1 <-
which.min(abs(tree_model3$cptable[,"xerror"] - optError_travel1))
```

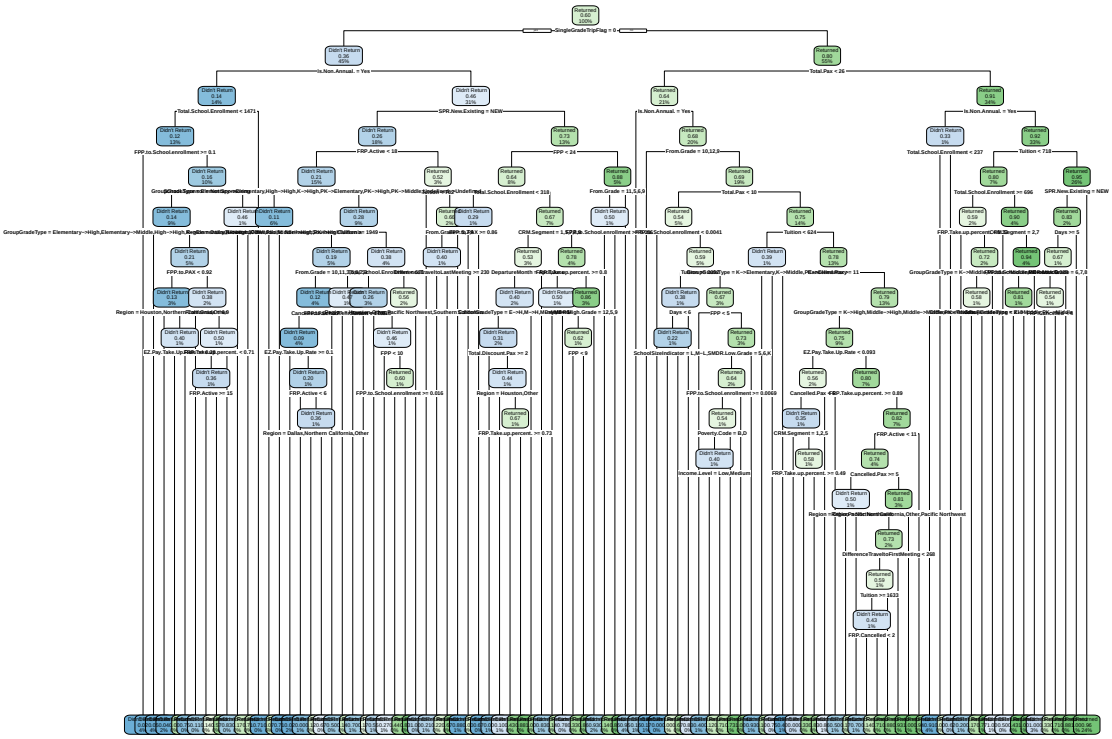
Finally, to get the best CP, we find the cp value corresponding to optCP_i calculated above:

```
optCP_travel1 <- tree_model3$cptable[optCP_i_travel1, "CP"]
print(optCP_travel1)
```

```
## [1] 0.004195804
```

To plot an rpart decision tree we can use the "rpart.plot()" function from "rpart.plot" package:

```
library(rpart.plot)
rpart.plot(tree_model3)
```



```
rpart.rules(tree_model3)
```

target
0.00 when SingleGradeTripFlag is 0 & Is.Non.Annual. is Yes & GroupGradeType is Middle->Middle or PK->Middle or Undefined->Undefined & Total.School.Enrollment < 1471 & FPP.to.School.enrollment < 0.1028 & Region is Dallas or Pacific Northwest or Southern California & EZ.Pay.Take.Up.Rate >= 0.280 & School.Sponsor is Not Sponsoring & FPP.to.PAX < 0.92
0.00 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition >= 1949 & From.Grade is 10 or 11 or 3 or 5 or 6 or 7 or 9 & GroupGradeType is Elementary->Middle or K->Elementary or K->Middle or Middle->High or Middle->Middle & SPR.New.Existing is NEW & FRP.Active < 6 & Cancelled.Pax < 17 & EZ.Pay.Take.Up.Rate < 0.103
0.00 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Total.School.Enrollment < 318 & SPR.New.Existing is EXISTING & FPP < 24 & FPP.to.PAX >= 0.86
0.00 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Total.School.Enrollment >= 318 & SPR.New.Existing is EXISTING & FPP < 24 & CRM.Segment is 1 or 5 or 7 or 8 or 9 & DepartureMonth is April or June & SchoolGradeType is E->H or M->H or M->M or M->U & Total.Discount.Pax >= 2
0.00 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & Total.Pax < 10 & FPP.to.School.enrollment < 0.0041
0.00 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 2002 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & Total.Pax < 10 & FPP.to.School.enrollment >= 0.0041 & Days < 6 & SchoolSizeIndicator

is L or M-L or S
0.00 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 624 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is K->High or Middle->High or Middle->Middle or PK->Middle & Total.Pax is 10 to 26 & Cancelled.Pax < 6 & EZ.Pay.Take.Up.Rate < 0.093 & CRM.Segment is 1 or 2 or 5
0.00 when SingleGradeTripFlag is 1 & Is.Non.Annual. is Yes & Total.Pax >= 26 & Total.School.Enrollment < 237
0.02 when SingleGradeTripFlag is 0 & Is.Non.Annual. is Yes & Total.School.Enrollment < 1471 & FPP.to.School.enrollment >= 0.1028
0.02 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition >= 1949 & From.Grade is 10 or 11 or 3 or 5 or 6 or 7 or 9 & GroupGradeType is Elementary->Middle or K->Elementary or K->Middle or Middle->High or Middle->Middle & SPR.New.Existing is NEW & FRP.Active < 18 & Cancelled.Pax < 17 & EZ.Pay.Take.Up.Rate >= 0.103
0.04 when SingleGradeTripFlag is 0 & Is.Non.Annual. is Yes & GroupGradeType is Middle->Middle or PK->Middle or Undefined->Undefined & Total.School.Enrollment < 1471 & FPP.to.School.enrollment < 0.1028 & Region is Houston or Northern California or Other & School.Sponsor is Not Sponsoring & FPP.to.PAX < 0.92
0.05 when SingleGradeTripFlag is 0 & Is.Non.Annual. is Yes & GroupGradeType is Elementary->High or Elementary->Middle or High->High or K->Elementary or K->High or K->Middle or Middle->High or PK->High & Total.School.Enrollment < 1471 & FPP.to.School.enrollment < 0.1028 & School.Sponsor is Not Sponsoring
0.07 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & GroupGradeType is Elementary->Elementary or High->High or K->High or PK->Elementary or PK->High or PK->Middle or Undefined->Undefined & SPR.New.Existing is NEW & FRP.Active < 18 & Region is Dallas or Houston or Other or Pacific Northwest or Southern California
0.10 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Total.School.Enrollment >= 318 & SPR.New.Existing is EXISTING & FPP < 24 & Region is Houston or Other & CRM.Segment is 1 or 5 or 7 or 8 or 9 & DepartureMonth is April or June & SchoolGradeType is E->H or M->H or M->M or M->U & Total.Discount.Pax < 2
0.10 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 624 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is K->Elementary or K->Middle or PK->Elementary & Total.Pax is 10 to 26
0.11 when SingleGradeTripFlag is 0 & Is.Non.Annual. is Yes & From.Grade is 6 or 9 & GroupGradeType is Middle->Middle or PK->Middle or Undefined->Undefined & Total.School.Enrollment < 1471 & FPP.to.School.enrollment < 0.1028 & School.Sponsor is Not Sponsoring & FPP.to.PAX >= 0.92
0.12 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition >= 1949 & From.Grade is 10 or 11 or 3 or 5 or 6 or 7 or 9 & GroupGradeType is Elementary->Middle or K->Elementary or K->Middle or Middle->High or Middle->Middle & SPR.New.Existing is NEW & FRP.Active is 6 to 18 & Region is Dallas or Northern California or Other & Cancelled.Pax < 17 & EZ.Pay.Take.Up.Rate < 0.103
0.12 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 2002 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & Total.Pax < 10 & FPP.to.School.enrollment >= 0.0069 & FPP >= 5 & MDR.Low.Grade is 5 or 6 or K & Poverty.Code is B or D & Income.Level is Low or Medium 0.14 when SingleGradeTripFlag is 0 & Is.Non.Annual. is Yes & From.Grade is 7 or 8 & GroupGradeType is Middle->Middle or PK->Middle or Undefined->Undefined & Total.School.Enrollment < 1471 & FPP.to.School.enrollment < 0.1028 & FRP.Active >= 15 & FRP.Take.up.percent. < 0.71 & School.Sponsor is Not Sponsoring & FPP.to.PAX >= 0.92
0.14 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition >= 1949 & From.Grade is 8 & GroupGradeType is Elementary->Middle or K->Elementary or K->Middle or Middle->High or Middle->Middle & SPR.New.Existing is NEW & FPP.to.School.enrollment >= 0.0177 & FRP.Active < 18
0.14 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Total.School.Enrollment >= 318 & SPR.New.Existing is EXISTING & FPP < 24 & FRP.Take.up.percent. >= 0.80 & CRM.Segment is 10 or 2 or 4 or 6 & Days >= 5
0.14 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & From.Grade is 11 or 5 or 6 or 9 & SPR.New.Existing is EXISTING & FPP.to.School.enrollment >= 0.0562 & FPP >= 24
0.14 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 1633 & From.Grade is 11 or 4

or 5 or 6 or 7 or 8 & GroupGradeType is K->High or Middle->High or Middle->Middle or PK->Middle & Total.Pax is 10 to 26 & FRP.Active < 11 & FRP.Take.up.percent. < 0.89 & Region is Northern California or Other or Pacific Northwest & Cancelled.Pax < 5 & EZ.Pay.Take.Up.Rate >= 0.093 & DifferenceTraveltoFirstMeeting < 268 & FRP.Cancelled < 2
0.16 when SingleGradeTripFlag is 1 & Is.Non.Annual. is Yes & Total.Pax < 26
0.17 when SingleGradeTripFlag is 0 & Is.Non.Annual. is Yes & Tuition < 1730 & Total.School.Enrollment < 1471 & FPP.to.School.enrollment < 0.1028 & School.Sponsor is Sponsoring
0.17 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & From.Grade is 10 or 12 or 9 & Total.Pax < 26
0.17 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 624 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is K->High or Middle->High or Middle->Middle or PK->Middle & Total.Pax is 10 to 26 & FRP.Active < 11 & FRP.Take.up.percent. < 0.89 & Region is Other or Pacific Northwest & Cancelled.Pax is 5 to 11 & EZ.Pay.Take.Up.Rate >= 0.093
0.17 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 718 & GroupGradeType is K->Middle or Middle->Middle or PK->Middle & Total.Pax >= 26 & Total.School.Enrollment >= 696 & FRP.Take.up.percent. >= 0.33 & DifferenceTraveltoFirstMeeting < 218
0.17 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition < 1786 & GroupGradeType is Elementary->Middle or K->Elementary or K->Middle or Middle->High or Middle->Middle & Total.School.Enrollment < 631 & SPR.New.Existing is NEW & FRP.Active < 18
0.20 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 718 & Total.Pax >= 26 & Total.School.Enrollment >= 696 & FRP.Take.up.percent. < 0.33
0.21 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition < 762 & SPR.New.Existing is NEW & FRP.Active >= 18
0.22 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition >= 762 & From.Grade is 5 or 7 or 9 & SPR.New.Existing is NEW & FRP.Active >= 18 & DifferenceTraveltoLastMeeting >= 230
0.27 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition < 1949 & GroupGradeType is Elementary->Middle or K->Elementary or K->Middle or Middle->High or Middle->Middle & Total.School.Enrollment >= 631 & SPR.New.Existing is NEW & FRP.Active < 18 & FPP < 10 & Region is Houston or Other or Pacific Northwest or Southern California
0.33 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Total.School.Enrollment >= 318 & SPR.New.Existing is EXISTING & FPP < 9 & FRP.Take.up.percent. < 0.80 & CRM.Segment is 10 or 2 or 4 or 6 & MDR.High.Grade is 12 or 5 or 9
0.33 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 624 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is K->High or Middle->High or Middle->Middle or PK->Middle & Total.Pax is 10 to 26 & FRP.Take.up.percent. >= 0.49 & Cancelled.Pax < 6 & EZ.Pay.Take.Up.Rate < 0.093 & CRM.Segment is 10 or 4 or 6
0.33 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 718 & Total.Pax >= 26 & SPR.New.Existing is NEW & Days >= 5 & MDR.Low.Grade is 6 or 7 or 8 & FRP.Cancelled < 4
0.40 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 2002 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & Total.Pax < 10 & FPP.to.School.enrollment >= 0.0041 & FPP < 5
0.40 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 624 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & Total.Pax is 10 to 26 & Cancelled.Pax >= 11
0.43 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Total.School.Enrollment >= 318 & SPR.New.Existing is EXISTING & FPP < 24 & FRP.Take.up.percent. >= 0.73 & Region is Dallas or Northern California or Pacific Northwest or Southern California & CRM.Segment is 1 or 5 or 7 or 8 or 9 & DepartureMonth is April or June & SchoolGradeType is E->H or M->H or M->M or M->U & Total.Discount.Pax < 2
0.43 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 718 & GroupGradeType is K->Middle or PK->Middle & Total.Pax >= 26 & Total.School.Enrollment < 696 & FPP.to.School.enrollment < 0.0880 & CRM.Segment is 10 or 11 or 3 or 4 or 5
0.44 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition < 1949 & GroupGradeType is Elementary->Middle or K->Elementary or K->Middle or Middle->High or Middle->Middle & Total.School.Enrollment >= 631 & SPR.New.Existing is NEW & FPP.to.School.enrollment >= 0.0157 & FRP.Active < 18 & FPP >= 10 & Region is Houston or Other or Pacific Northwest or Southern California

0.50 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition >= 1949 & From.Grade is 10 or 11 or 3 or 5 or 6 or 7 or 9 & GroupGradeType is Elementary->Middle or K->Elementary or K->Middle or Middle->High or Middle->Middle & SPR.New.Existing is NEW & FRP.Active < 18 & Cancelled.Pax >= 17
0.50 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 624 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is K->High or Middle->High or Middle->Middle or PK->Middle & Total.Pax is 10 to 26 & FRP.Take.up.percent. >= 0.89 & Cancelled.Pax < 11 & EZ.Pay.Take.Up.Rate >= 0.093
0.50 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 718 & Total.Pax >= 26 & Total.School.Enrollment < 696 & CRM.Segment is 2 or 7
0.55 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition is 1786 to 1949 & GroupGradeType is Elementary->Middle or K->Elementary or K->Middle or Middle->High or Middle->Middle & Total.School.Enrollment < 631 & SPR.New.Existing is NEW & FRP.Active < 18
0.57 when SingleGradeTripFlag is 0 & Is.Non.Annual. is Yes & From.Grade is 7 or 8 & GroupGradeType is Middle->Middle or PK->Middle or Undefined->Undefined & Total.School.Enrollment < 1471 & FPP.to.School.enrollment < 0.1028 & FRP.Active < 15 & FRP.Take.up.percent. < 0.71 & School.Sponsor is Not Sponsoring & FPP.to.PAX >= 0.92
0.62 when SingleGradeTripFlag is 1 & Is.Non.Annual. is Yes & Total.Pax >= 26 & Total.School.Enrollment >= 237
0.67 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition >= 1949 & From.Grade is 10 or 11 or 3 or 5 or 6 or 7 or 9 & GroupGradeType is Elementary->Middle or K->Elementary or K->Middle or Middle->High or Middle->Middle & SPR.New.Existing is NEW & FRP.Active is 6 to 18 & Region is Houston or Pacific Northwest or Southern California & Cancelled.Pax < 17 & EZ.Pay.Take.Up.Rate < 0.103
0.67 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition >= 762 & From.Grade is 5 or 7 or 9 & SPR.New.Existing is NEW & FRP.Active >= 18 & DifferenceTraveltoLastMeeting < 230
0.67 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Total.School.Enrollment < 318 & SPR.New.Existing is EXISTING & FPP < 24 & FPP.to.PAX < 0.86
0.67 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 2002 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & Total.Pax < 10 & FPP.to.School.enrollment >= 0.0041 & Days < 6 & SchoolSizeIndicator is S-M
0.70 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition >= 1949 & From.Grade is 8 & GroupGradeType is Elementary->Middle or K->Elementary or K->Middle or Middle->High or Middle->Middle & SPR.New.Existing is NEW & FPP.to.School.enrollment < 0.0177 & FRP.Active < 18
0.70 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 624 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is K->High or Middle->High or Middle->Middle or PK->Middle & Total.Pax is 10 to 26 & FRP.Active < 11 & FRP.Take.up.percent. < 0.89 & Region is Dallas or Houston or Northern California or Southern California & Cancelled.Pax is 5 to 11 & EZ.Pay.Take.Up.Rate >= 0.093
0.71 when SingleGradeTripFlag is 0 & Is.Non.Annual. is Yes & Tuition >= 1730 & Total.School.Enrollment < 1471 & FPP.to.School.enrollment < 0.1028 & School.Sponsor is Sponsoring
0.71 when SingleGradeTripFlag is 0 & Is.Non.Annual. is Yes & Total.School.Enrollment >= 1471
0.71 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & GroupGradeType is Elementary->Elementary or High->High or K->High or PK->Elementary or PK->High or PK->Middle or Undefined->Undefined & SPR.New.Existing is NEW & FRP.Active < 18 & Region is Northern California
0.71 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 2002 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & Total.Pax < 10 & FPP.to.School.enrollment >= 0.0069 & FPP >= 5 & MDR.Low.Grade is 5 or 6 or K & Poverty.Code is B or D & Income.Level is High 0.71 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 1633 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is K->High or Middle->High or Middle->Middle or PK->Middle & Total.Pax is 10 to 26 & FRP.Active < 11 & FRP.Take.up.percent. < 0.89 & Region is Northern California or Other or Pacific Northwest & Cancelled.Pax < 5 & EZ.Pay.Take.Up.Rate >= 0.093 & DifferenceTraveltoFirstMeeting < 268 & FRP.Cancelled >= 2
0.71 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 718 & Total.Pax >= 26 & SPR.New.Existing is NEW & Days >= 5 & MDR.Low.Grade is 6 or 7 or 8 & FRP.Cancelled >= 4

0.73 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 2002 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & Total.Pax < 10 & FPP.to.School.enrollment >= 0.0069 & FPP >= 5 & MDR.Low.Grade is 5 or 6 or K & Poverty.Code is A or C
0.75 when SingleGradeTripFlag is 0 & Is.Non.Annual. is Yes & GroupGradeType is Middle->Middle or PK->Middle or Undefined->Undefined & Total.School.Enrollment < 1471 & FPP.to.School.enrollment < 0.1028 & Region is Dallas or Pacific Northwest or Southern California & EZ.Pay.Take.Up.Rate < 0.280 & School.Sponsor is Not Sponsoring & FPP.to.PAX < 0.92
0.75 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 624 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is Middle->High or Middle->Middle or PK->Middle or Undefined->Undefined & Total.Pax is 10 to 26
0.77 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 718 & GroupGradeType is K->Middle or Middle->Middle or PK->Middle & Total.Pax >= 26 & Total.School.Enrollment >= 696 & FRP.Take.up.percent. >= 0.33 & DifferenceTraveltoFirstMeeting >= 218
0.78 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Total.School.Enrollment >= 318 & SPR.New.Existing is EXISTING & FPP < 24 & FRP.Take.up.percent. >= 0.80 & CRM.Segment is 10 or 2 or 4 or 6 & Days < 5
0.83 when SingleGradeTripFlag is 0 & Is.Non.Annual. is Yes & From.Grade is 7 or 8 & GroupGradeType is Middle->Middle or PK->Middle or Undefined->Undefined & Total.School.Enrollment < 1471 & FPP.to.School.enrollment < 0.1028 & FRP.Take.up.percent. >= 0.71 & School.Sponsor is Not Sponsoring & FPP.to.PAX >= 0.92
0.83 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition < 1949 & GroupGradeType is Elementary->Middle or K->Elementary or K->Middle or Middle->High or Middle->Middle & Total.School.Enrollment >= 631 & SPR.New.Existing is NEW & FPP.to.School.enrollment < 0.0157 & FRP.Active < 18 & FPP >= 10 & Region is Houston or Other or Pacific Northwest or Southern California
0.83 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Total.School.Enrollment >= 318 & SPR.New.Existing is EXISTING & FPP < 24 & CRM.Segment is 1 or 5 or 7 or 8 or 9 & DepartureMonth is February or March or May
0.83 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 2002 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & Total.Pax < 10 & FPP.to.School.enrollment >= 0.0041 & Days >= 6
0.83 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 624 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is K->High or Middle->High or Middle->Middle or PK->Middle & Total.Pax is 10 to 26 & FRP.Take.up.percent. < 0.49 & Cancelled.Pax < 6 & EZ.Pay.Take.Up.Rate < 0.093 & CRM.Segment is 10 or 4 or 6
0.86 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Total.School.Enrollment >= 318 & SPR.New.Existing is EXISTING & FPP is 9 to 24 & FRP.Take.up.percent. < 0.80 & CRM.Segment is 10 or 2 or 4 or 6 & MDR.High.Grade is 12 or 5 or 9
0.86 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & From.Grade is 11 or 5 or 6 or 9 & SPR.New.Existing is EXISTING & FPP.to.School.enrollment < 0.0562 & FPP >= 24
0.86 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 624 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is K->High or Middle->High or Middle->Middle or PK->Middle & Total.Pax is 10 to 26 & Cancelled.Pax is 6 to 11 & EZ.Pay.Take.Up.Rate < 0.093
0.88 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Total.School.Enrollment >= 318 & SPR.New.Existing is EXISTING & FPP < 24 & FRP.Take.up.percent. < 0.73 & Region is Dallas or Northern California or Pacific Northwest or Southern California & CRM.Segment is 1 or 5 or 7 or 8 or 9 & DepartureMonth is April or June & SchoolGradeType is E->H or M->H or M->M or M->U & Total.Discount.Pax < 2
0.88 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition is 624 to 1633 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is K->High or Middle->High or Middle->Middle or PK->Middle & Total.Pax is 10 to 26 & FRP.Active < 11 & FRP.Take.up.percent. < 0.89 & Region is Northern California or Other or Pacific Northwest & Cancelled.Pax < 5 & EZ.Pay.Take.Up.Rate >= 0.093 & DifferenceTraveltoFirstMeeting < 268
0.88 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 718 & Total.Pax >= 26 & SPR.New.Existing is NEW & Days >= 5 & MDR.Low.Grade is 3 or 5 or 9 or K or PK
0.88 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition >= 762 & From.Grade is 6 or 8 &

SPR.New.Existing is NEW & FRP.Active >= 18
0.91 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 624 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is Elementary->Elementary or Elementary->Middle or High->High or K->Elementary or K->Middle or PK->Elementary or PK->High or Undefined->Undefined & Total.Pax is 10 to 26 & Cancelled.Pax < 11
0.93 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Total.School.Enrollment >= 318 & SPR.New.Existing is EXISTING & FPP < 24 & FRP.Take.up.percent. < 0.80 & CRM.Segment is 10 or 2 or 4 or 6 & MDR.High.Grade is 3 or 4 or 6 or 7 or 8
0.93 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 2002 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & Total.Pax < 10 & FPP.to.School.enrollment >= 0.0041 & FPP >= 5 & MDR.Low.Grade is 4 or 7 or PK
0.93 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 624 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is K->High or Middle->High or Middle->Middle or PK->Middle & Total.Pax is 10 to 26 & FRP.Active < 11 & FRP.Take.up.percent. < 0.89 & Region is Northern California or Other or Pacific Northwest & Cancelled.Pax < 5 & EZ.Pay.Take.Up.Rate >= 0.093 & DifferenceTraveltoFirstMeeting >= 268
0.94 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 624 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is K->High or Middle->High or Middle->Middle or PK->Middle & Total.Pax is 10 to 26 & FRP.Active >= 11 & FRP.Take.up.percent. < 0.89 & Cancelled.Pax < 11 & EZ.Pay.Take.Up.Rate >= 0.093
0.95 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & From.Grade is 4 or 7 or 8 & SPR.New.Existing is EXISTING & FPP >= 24
0.96 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 718 & Total.Pax >= 26 & SPR.New.Existing is EXISTING
1.00 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Tuition < 1949 & GroupGradeType is Elementary->Middle or K->Elementary or K->Middle or Middle->High or Middle->Middle & Total.School.Enrollment >= 631 & SPR.New.Existing is NEW & FRP.Active < 18 & Region is Dallas or Northern California
1.00 when SingleGradeTripFlag is 0 & Is.Non.Annual. is No & Total.School.Enrollment >= 318 & SPR.New.Existing is EXISTING & FPP < 24 & CRM.Segment is 1 or 5 or 7 or 8 or 9 & DepartureMonth is April or June & SchoolGradeType is E->M or H->H or U->U
1.00 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 2002 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & Total.Pax < 10 & FPP.to.School.enrollment is 0.0041 to 0.0069 & FPP >= 5 & MDR.Low.Grade is 5 or 6 or K
1.00 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 624 & From.Grade is 11 or 4 or 5 or 6 or 7 or 8 & GroupGradeType is K->High or Middle->High or Middle->Middle or PK->Middle & Total.Pax is 10 to 26 & FRP.Active < 11 & FRP.Take.up.percent. < 0.89 & Region is Dallas or Houston or Southern California & Cancelled.Pax < 5 & EZ.Pay.Take.Up.Rate >= 0.093
1.00 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 718 & GroupGradeType is Elementary->Middle or K->Elementary or K->High or Middle->High or PK->High & Total.Pax >= 26 & Total.School.Enrollment >= 696 & FRP.Take.up.percent. >= 0.33
1.00 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 718 & GroupGradeType is Elementary->Elementary or Elementary->High or Elementary->Middle or K->Elementary or Middle->High or Middle->Middle or PK->Elementary & Total.Pax >= 26 & Total.School.Enrollment < 696 & FPP.to.School.enrollment < 0.0880 & CRM.Segment is 10 or 11 or 3 or 4 or 5
1.00 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition < 718 & Total.Pax >= 26 & Total.School.Enrollment < 696 & FPP.to.School.enrollment >= 0.0880 & CRM.Segment is 10 or 11 or 3 or 4 or 5
1.00 when SingleGradeTripFlag is 1 & Is.Non.Annual. is No & Tuition >= 718 & Total.Pax >= 26 & SPR.New.Existing is NEW & Days < 5

To obtain the error rate for the training and test data, we run the following sets of commands:

```
#Error rate for training data

pred_train3 <- predict(tree_model3, data = train_dt3, type = "class")
mean(train_dt3$target != pred_train3)
```

```
## [1] 0.1025499
```

```
# Error rate for the test data

tree_model3_pruned <- prune(tree_model3, cp = optCP_travel1)

pred_test3 <- predict(tree_model3_pruned, data1 = test_dt3, type = "class")
mean(test_dt3$target != pred_test3)
```

```
## [1] 0.463969
```

Final summary is that for cp=0, we see the best error rate on training data, which is **0.10 (10%)**, but for default cp or any other value, we see the error rate on training data to be in the ballpark of **0.20 (20%)**. Regardless of the cp value, we see that the error rate for the test data in all scenarios remains in the ballpark of **0.45 (45%)**.

Random Forest

In order to create a random forest model for our data set, we run the following chunk of code:

```
library(randomForest)
rf <- randomForest(target ~ ., data = tr, mtry = sqrt(ncol(tr)-1),
                   ntree = 300, proximity = T, importance = T)

print(rf)
```

```
##
## Call:
##  randomForest(formula = target ~ ., data = tr, mtry = sqrt(ncol(tr) -      1), ntree = 300, proximity
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 19.88%
## Confusion matrix:
##                 Didn't Return Returned class.error
##  Didn't Return            658      280   0.2985075
## Returned                  195     1256   0.1343901
```

```
names(rf)
```

```
##  [1] "call"            "type"            "predicted"       "err.rate"
##  [5] "confusion"       "votes"           "oob.times"       "classes"
##  [9] "importance"      "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"          "y"
## [17] "test"            "inbag"           "terms"
```

The OOB error rate for our random forest model is **0.1988**

Attributes

To view all the attributes we can call upon using our model, we utilize the attributes() function as follows:

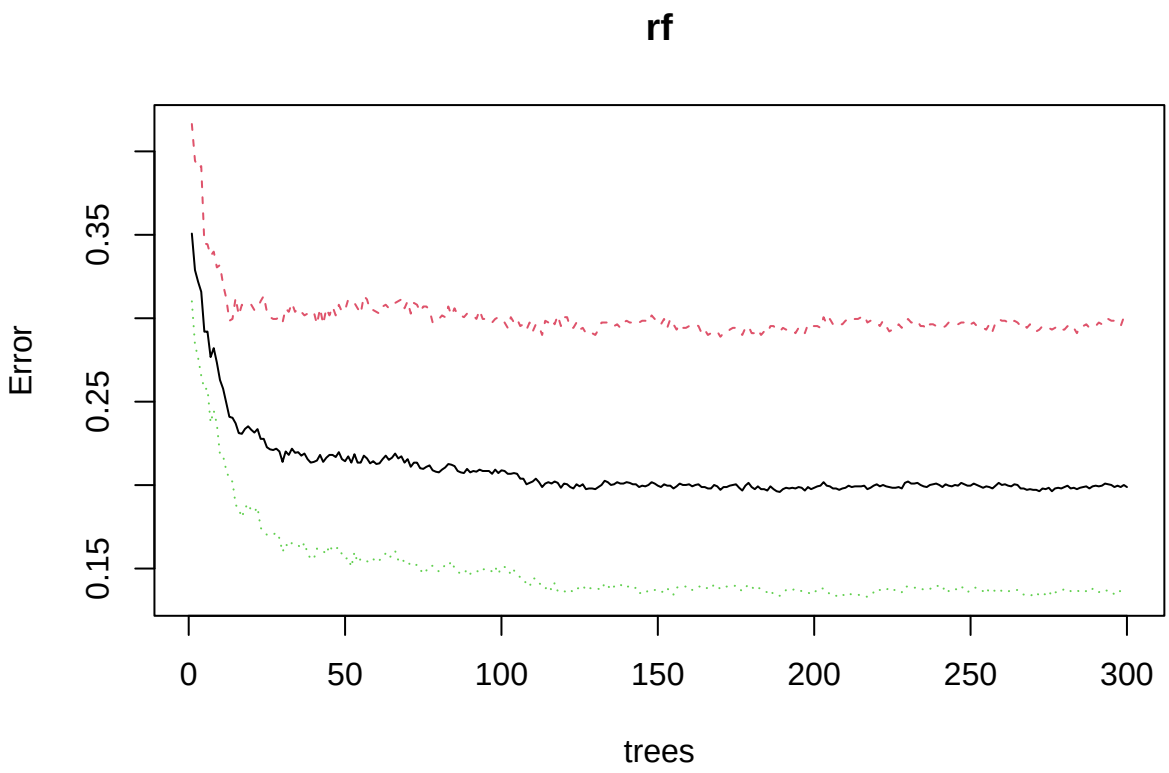
```
attributes(rf)
```

```
## $names
##  [1] "call"            "type"            "predicted"       "err.rate"
##  [5] "confusion"       "votes"           "oob.times"       "classes"
##  [9] "importance"      "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"          "y"
## [17] "test"            "inbag"           "terms"
##
## $class
## [1] "randomForest.formula" "randomForest"
```

Plot

We plot the error rates with various number of trees using the plot() function. In this result, we'll see a red curve, which is the error rate for the positive class that is **"Returned"** in our case, green curve is for the negative class that is **"Didn't Return"**, and the black curve indicates the error rate on OOB.

```
plot(rf)
```



MeanDecreaseAccuracy & MeanDecreaseGini

Get the importance of variables by the function "importance()". Include type = 1 in the importance function is to get the important variables based on MeanDecreaseAccuracy. Type=2 is for MeanDecreaseGini. Just selecting a subset of results, where the value is greater than 10, so we don't get many variables back.

```
imp1 <- importance(rf, type = 1)
imp2 <- importance(rf, type = 2)

imp1
```

```
##                                MeanDecreaseAccuracy
## From.Grade                                16.452642
## To.Grade                                   7.893186
## Is.Non.Annual.                            37.478343
## Days                                       4.391495
## Travel.Type                                3.931284
## Tuition                                    9.775860
## FRP.Active                                12.114944
## FRP.Cancelled                              5.884096
## FRP.Take.up.percent.                       6.731775
## Cancelled.Pax                              6.131307
## Total.Discount.Pax                         9.464729
## Poverty.Code                               5.707389
## Region                                     4.440006
## CRM.Segment                                8.931584
## School.Type                                5.512021
## Parent.Meeting.Flag                        1.750913
## MDR.Low.Grade                              5.993363
## MDR.High.Grade                             8.984999
## Total.School.Enrollment                   14.149196
## Income.Level                               1.665292
## EZ.Pay.Take.Up.Rate                        6.129636
## School.Sponsor                             5.276202
## SPR.Product.Type                           3.301831
## SPR.New.Existing                          29.307253
## FPP                                       15.431048
## Total.Pax                                 15.595479
## NumberOfMeetingswithParents                1.616548
## DifferenceTraveltoFirstMeeting             5.043118
## DifferenceTraveltoLastMeeting              4.622114
## SchoolGradeTypeLow                         4.915135
## SchoolGradeTypeHigh                        7.427218
## SchoolGradeType                            8.543191
## DepartureMonth                             5.083045
## GroupGradeTypeLow                          6.570347
## GroupGradeTypeHigh                         6.328809
## GroupGradeType                             9.397924
## MajorProgramCode                           2.554465
## SingleGradeTripFlag                       20.766058
## FPP.to.School.enrollment                  12.661887
## FPP.to.PAX                                10.068933
## Num.of.Non_FPP.PAX                         8.959834
## SchoolSizeIndicator                        7.785244
```

```
imp2
```

```
##                                 MeanDecreaseGini
## From.Grade                             62.880651
## To.Grade                               18.598423
## Is.Non.Annual.                         95.765599
## Days                                   14.318569
## Travel.Type                             3.402045
## Tuition                                40.335320
## FRP.Active                             41.391146
## FRP.Cancelled                          22.059072
## FRP.Take.up.percent.                   35.754406
## Cancelled.Pax                          25.516921
## Total.Discount.Pax                     19.681902
## Poverty.Code                           15.559215
## Region                                 25.591333
## CRM.Segment                            31.083818
## School.Type                             9.499462
## Parent.Meeting.Flag                     4.121841
## MDR.Low.Grade                          22.199935
## MDR.High.Grade                         14.263169
## Total.School.Enrollment                45.634244
## Income.Level                           12.680348
## EZ.Pay.Take.Up.Rate                    32.612817
## School.Sponsor                          2.691971
## SPR.Product.Type                        4.718385
## SPR.New.Existing                       59.926367
## FPP                                    46.609571
## Total.Pax                              51.401428
## NumberOfMeetingswithParents             9.432138
## DifferenceTraveltoFirstMeeting         35.013471
## DifferenceTraveltoLastMeeting          34.791248
## SchoolGradeTypeLow                      5.074731
## SchoolGradeTypeHigh                     8.911156
## SchoolGradeType                        17.668816
## DepartureMonth                         20.845130
## GroupGradeTypeLow                      11.437321
## GroupGradeTypeHigh                      5.909934
## GroupGradeType                         33.861697
## MajorProgramCode                        4.233346
## SingleGradeTripFlag                    77.725569
## FPP.to.School.enrollment               40.820111
## FPP.to.PAX                             34.474282
## Num.of.Non_FPP.PAX                     20.796909
## SchoolSizeIndicator                    18.598129
```

To see just a subset of the important variables, we can set a threshold on MeanDecreaseAccuracy and MeanDecreaseGini > 10

```
subset(imp1, imp1[] >  10)
```

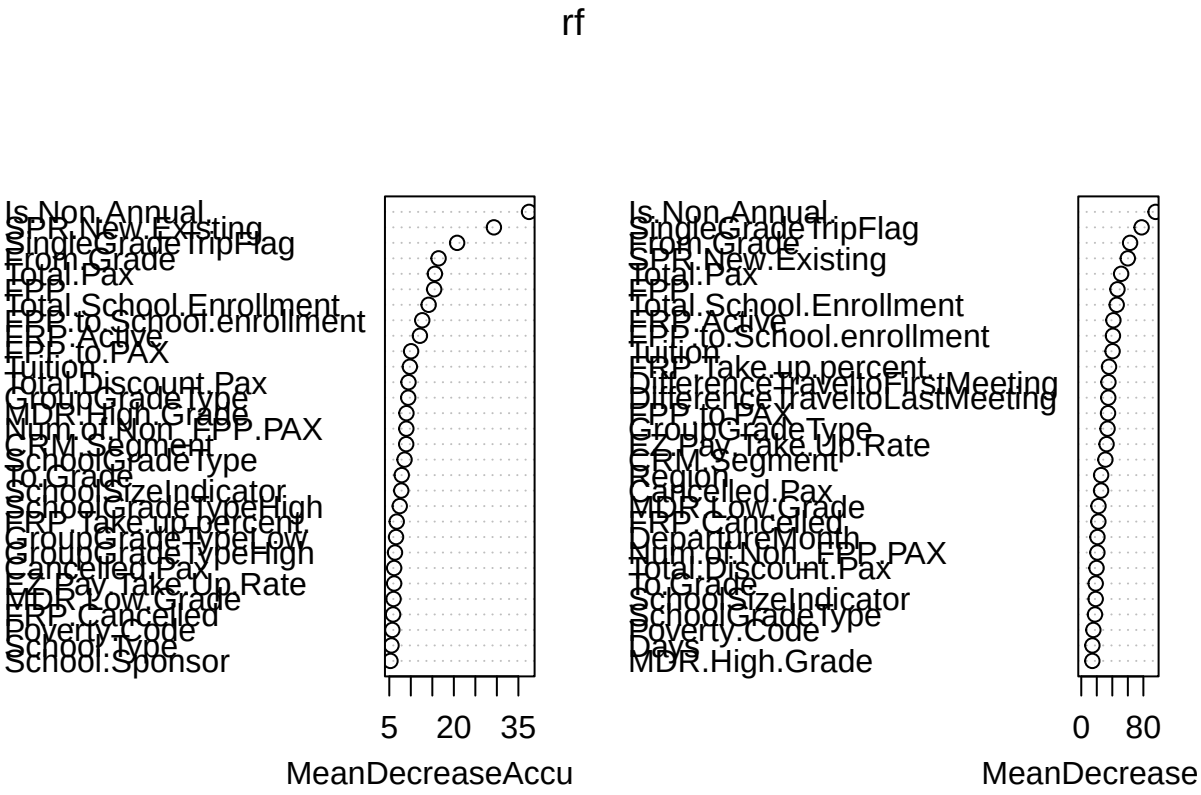
```
##                           MeanDecreaseAccuracy
## From.Grade                            16.45264
## Is.Non.Annual.                        37.47834
## FRP.Active                            12.11494
## Total.School.Enrollment               14.14920
## SPR.New.Existing                      29.30725
## FPP                                   15.43105
## Total.Pax                             15.59548
## SingleGradeTripFlag                   20.76606
## FPP.to.School.enrollment              12.66189
## FPP.to.PAX                            10.06893
```

```
subset(imp2, imp2[] >  10)
```

```
##                                 MeanDecreaseGini
## From.Grade                             62.88065
## To.Grade                               18.59842
## Is.Non.Annual.                         95.76560
## Days                                   14.31857
## Tuition                                40.33532
## FRP.Active                             41.39115
## FRP.Cancelled                          22.05907
## FRP.Take.up.percent.                   35.75441
## Cancelled.Pax                          25.51692
## Total.Discount.Pax                     19.68190
## Poverty.Code                           15.55921
## Region                                 25.59133
## CRM.Segment                            31.08382
## MDR.Low.Grade                          22.19993
## MDR.High.Grade                         14.26317
## Total.School.Enrollment                45.63424
## Income.Level                           12.68035
## EZ.Pay.Take.Up.Rate                    32.61282
## SPR.New.Existing                       59.92637
## FPP                                    46.60957
## Total.Pax                              51.40143
## DifferenceTraveltoFirstMeeting         35.01347
## DifferenceTraveltoLastMeeting          34.79125
## SchoolGradeType                        17.66882
## DepartureMonth                         20.84513
## GroupGradeTypeLow                      11.43732
## GroupGradeType                         33.86170
## SingleGradeTripFlag                    77.72557
## FPP.to.School.enrollment               40.82011
## FPP.to.PAX                             34.47428
## Num.of.Non_FPP.PAX                     20.79691
## SchoolSizeIndicator                    18.59813
```

Importance Plot

```
varImpPlot(rf)
```



Based on the above plots, we can see that **Is.Non.Annual** is the most important variable in both MeanDecreaseAccuracy and MeanDecreaseGini categories. However, the second most important variable in the first category is **SPR.New.Existing**, which makes sense as they're both the measures of new and existing customers and whether they go for a tour annually or not. In the second category, the second most important variable is **SingleGradeTripFlag**, which is what we saw in our Decision Tree model, as that's the flag for student groups belonging to the same grade, going for a tour. All in all, these variables are indeed quite important in predicting the outcome of our model and whether customers will return to the travel company or not.

Predicted Classes & Probablities

We can also obtain the predicted classes and predicted probabilities using the following codes:

```
head(rf$predicted)
```

```
##              1              2              3              4              5
##       Returned       Returned       Returned  Didn't Return       Returned
##              6
##  Didn't Return
## Levels:  Didn't Return Returned
```

```
head(rf$votes)
```

```
##    Didn't Return  Returned
## 1      0.1101695 0.8898305
## 2      0.3333333 0.6666667
## 3      0.2166667 0.7833333
## 4      0.5267857 0.4732143
## 5      0.1792453 0.8207547
## 6      0.7796610 0.2203390
```

Best mtry

To obtain the best value of mtry we can use the validation set. In particular, we can check the performance of the model for different values of mtry and check which one works best on a validation set.

```
ind_rf <- sample(2, nrow(tr), replace = T, prob = c(0.7, 0.3))
train_rf <- tr[ind_rf == 1, ]
validation_rf <- tr[ind_rf == 2, ]

pr.err <- c()
for(mt in seq(1,ncol(train_rf))) {
  library(randomForest)
  rf1 <- randomForest(target~., data = train_rf, ntree = 100,
                mtry = ifelse(mt == ncol(train_rf), mt-1, mt))

  predicted_rf <- predict(rf1, newdata = validation_rf, type = "class")
  pr.err <- c(pr.err, mean(validation_rf$target != predicted_rf))
}

bestmtry <- which.min(pr.err)

bestmtry
```

```
## [1] 4
```

Confusion Matrix

To obtain a confusion matrix, we can also use the confusionMatrix() function from the "caret" package. Similar to the table() function, confusionMatrix() also receives the predicted and actual labels as inputs.

```
library(caret)
confusionMatrix(rf$predicted, tr$target, positive = "Returned")
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction        Didn't Return Returned
##    Didn't Return            658      195
##   Returned                  280     1256
##
##                Accuracy : 0.8012
##                  95% CI : (0.7846, 0.817)
##     No Information Rate : 0.6074
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5763
##
##  Mcnemar's Test P-Value : 0.0001161
##
##             Sensitivity : 0.8656
##             Specificity : 0.7015
##          Pos Pred Value : 0.8177
##          Neg Pred Value : 0.7714
##              Prevalence : 0.6074
##          Detection Rate : 0.5257
##    Detection Prevalence : 0.6429
##       Balanced Accuracy : 0.7836
##
##        'Positive' Class : Returned
##
```

Evaluation Charts

To draw the evaluation charts we use "ROCR" package. There are two function in this package that we require to draw all different charts discussed in class: prediction and performance. The prediction() function receives two inputs:

1. The predicted probability of the positive class and
2. The true labels

The output of the prediction function will be given to the performance() function to draw the charts

```
library(ROCR)
score <- rf$votes[, 2]
pred <- prediction(score, tr$target)

pred
```

```
## A prediction instance
##   with 2389 data points
```

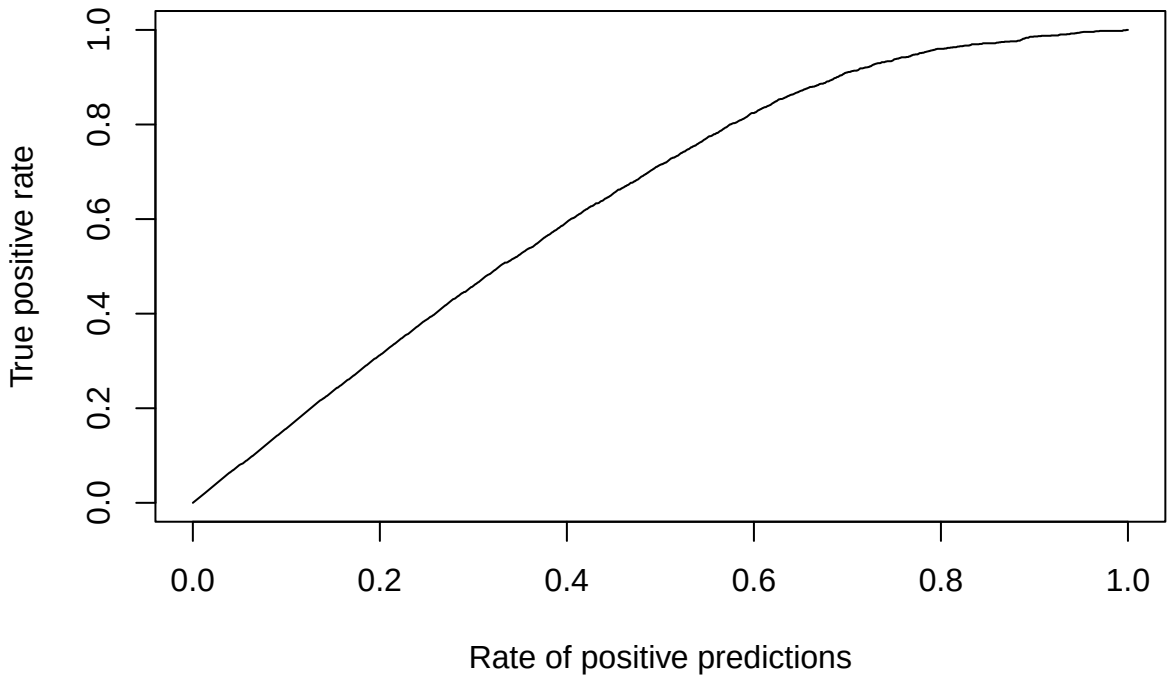
Gain chart

The gain chart for our model is:

```
perf <- performance(pred, "tpr", "rpp")

perf
```

```
## A performance instance
##   'Rate of positive predictions' vs. 'True positive rate' (alpha: 'Cutoff')
##   with 1588 data points
```

```
plot(perf)
```



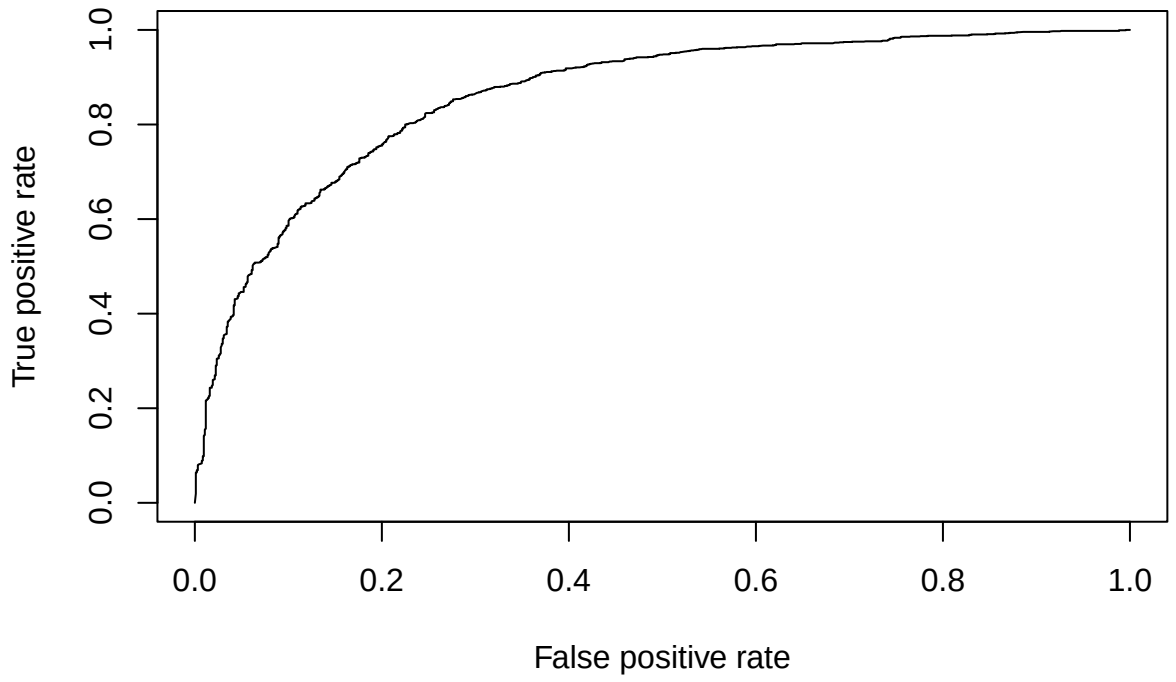
ROC Curve

The ROC curve for our model is:

```
perf1 <- performance(pred, "tpr", "fpr")

perf1
```

```
## A performance instance
##   'False positive rate' vs. 'True positive rate' (alpha: 'Cutoff')
##   with 1588 data points
```

```
plot(perf1)
```



Area under the curve

The area under the curve of our ROC curve is:

```
auc <- unlist(slot(performance(pred, "auc"), "y.values"))
```

The area under the curve is **0.8624245.**

Determining the best cut-off point

The performance() function for ROC curve returns tpr, fpr and alpha-values (cut-off points). We can use the following code to write a function that received these and return the best cut-off point as the point closest to the corner [0, 1]. The input argument to this function is perf (the output of the performance() function).

The mapply function applies the function FUN to all **perf@x.values**, **perf@y.values**, and **perf@alpha.values**. In the function FUN(), we first compute the distance of all the points on the ROC curve from the corner point [0,1]. These distance values are stored in the vector "d". We then find the index of the point that is the closest point to the corner. This index is stored in the variable named "ind". The output of this function is then the tpr, fpr and the probability threshold corresponding to this index.

```
cut.ind <- mapply(FUN = function(x,y,p) {
  d=(x-0)^2+(y-1)^2
  ind<- which(d==min(d))
  c(recall = y[[ind]], specificity = 1-x[[ind]],cutoff = p[[ind]])
  }, perf@x.values, perf@y.values, perf@alpha.values)

cut.ind
```

```
##                  [,1]
## recall      0.6271537
## specificity 0.5742989
## cutoff      0.7238095
```

Logistic regression

The original data is first divided into a test set and a training set based on a ratio of 8:2.

```
set.seed(123)
trainIndex <- createDataPartition(tr$target, p = 0.8, list = FALSE)
training <- tr[trainIndex, ]
testing <- tr[-trainIndex, ]
```

Use the glm() function to create a logistic regression model. The predicted variable is 'target'. 'family = "binomial" ' specifies the type of probability distribution used in the logistic regression model, in this case binomial, which is suitable for binary classification problems.

```
glm <- glm(target ~ ., data = training, family = "binomial")
summary(glm)
```

```
##
## Call:
## glm(formula = target ~ ., family = "binomial", data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9130  -0.6011   0.3076   0.6197   3.1024
##
## Coefficients: (27 not defined because of singularities)
##                                    Estimate    Std. Error z value Pr(>|z|)
## (Intercept)                     -3.37595877 1185.92870078  -0.003 0.997729
## From.Grade11                     0.50744672    0.83672607   0.606 0.544205
## From.Grade12                    -1.47243772    1.21958593  -1.207 0.227307
## From.Grade3                    -16.50242551  878.76465155  -0.019 0.985017
## From.Grade4                     -2.03808535    2.14790370  -0.949 0.342686
## From.Grade5                     -1.54367462    1.97272949  -0.783 0.433917
## From.Grade6                     -1.76342742    1.28240950  -1.375 0.169104
## From.Grade7                     -1.87275411    1.26530704  -1.480 0.138852
## From.Grade8                     -1.33787687    1.22226173  -1.095 0.273696
## From.Grade9                     -0.38041683    0.82376055  -0.462 0.644221
## To.Grade11                       1.43391900    1.34082782   1.069 0.284877
## To.Grade12                       1.86609925    1.22059897   1.529 0.126304
## To.Grade3                       31.39119758 1700.12074234   0.018 0.985269
## To.Grade4                        3.74970450    2.32605664   1.612 0.106952
## To.Grade5                        4.10855579    2.19496777   1.872 0.061233
## To.Grade6                        3.40855234    1.56127886   2.183 0.029023
## To.Grade7                        4.36336174    1.52788605   2.856 0.004293
## To.Grade8                        3.71671361    1.47675199   2.517 0.011842
## To.Grade9                        2.81664294    1.29017825   2.183 0.029025
## Is.Non.Annual.Yes               -2.58540034    0.21310999 -12.132  < 2e-16
## Days                             0.08833716    0.10622587   0.832 0.405636
## Travel.TypeB                    -0.08918803    0.33558207  -0.266 0.790415
## Travel.TypeT                     1.10722842    1.31654520   0.841 0.400342
## Tuition                         -0.00009532    0.00032104  -0.297 0.766550
## FRP.Active                       0.03569459    0.01037753   3.440 0.000583
## FRP.Cancelled                   -0.04821486    0.04030527  -1.196 0.231602
## FRP.Take.up.percent.            -0.58577566    0.39847123  -1.470 0.141546
```

```
## Cancelled.Pax                          0.03531605    0.02891781   1.221 0.221989
## Total.Discount.Pax                     0.10830856    0.06762265   1.602 0.109231
## Poverty.CodeA                          2.77678914    1.55016579   1.791 0.073248
## Poverty.CodeB                          2.50256715    1.52692696   1.639 0.101222
## Poverty.CodeC                          2.37541797    1.53275579   1.550 0.121197
## Poverty.CodeD                          2.40411822    1.61073526   1.493 0.135553
## Poverty.CodeE                          3.69688612    1.73148762   2.135 0.032753
## RegionHouston                         -1.02063207    0.36235521  -2.817 0.004853
## RegionNorthern California             -1.07572194    0.36207473  -2.971 0.002968
## RegionOther                           -0.69705258    0.28254790  -2.467 0.013624
## RegionPacific Northwest               -1.30105623    0.36654622  -3.550 0.000386
## RegionSouthern California             -0.31942479    0.35606524  -0.897 0.369668
## CRM.Segment10                          0.61419387    0.38159651   1.610 0.107499
## CRM.Segment11                          0.10867841    0.99629269   0.109 0.913137
## CRM.Segment2                          -0.56875272    0.56995642  -0.998 0.318334
## CRM.Segment3                           1.09440360    0.99914236   1.095 0.273366
## CRM.Segment4                           1.97506983    0.80879515   2.442 0.014607
## CRM.Segment5                           0.51831795    0.41098787   1.261 0.207254
## CRM.Segment6                           1.51992852    0.81276695   1.870 0.061475
## CRM.Segment7                           0.65178845    0.76244254   0.855 0.392624
## CRM.Segment8                           0.05161468    0.90695737   0.057 0.954617
## CRM.Segment9                          -1.79581343    1.08979124  -1.648 0.099383
## School.TypeCHD                        -0.17738191    0.38829693  -0.457 0.647800
## School.TypePrivate non-Christian       0.71382523    0.41617414   1.715 0.086307
## School.TypePUBLIC                     -0.36487758    0.33464105  -1.090 0.275557
## Parent.Meeting.Flag                   -0.03770023    0.40528106  -0.093 0.925886
## MDR.Low.Grade10                      -15.50874628  597.42830281  -0.026 0.979290
## MDR.Low.Grade2                       -16.29471216 1455.39918924  -0.011 0.991067
## MDR.Low.Grade3                         0.25414735    1.60878515   0.158 0.874477
## MDR.Low.Grade4                        -0.21850385    1.37647197  -0.159 0.873872
## MDR.Low.Grade5                         0.42366773    1.23739541   0.342 0.732060
## MDR.Low.Grade6                       -14.88122750  597.42554257  -0.025 0.980128
## MDR.Low.Grade7                       -14.71030410  597.42558019  -0.025 0.980356
## MDR.Low.Grade8                       -15.24540785  597.42600942  -0.026 0.979641
## MDR.Low.Grade9                       -14.94558223  597.42575145  -0.025 0.980042
## MDR.Low.GradeK                       -14.97129268  597.42516641  -0.025 0.980007
## MDR.Low.GradePK                      -15.19450073  597.42509303  -0.025 0.979709
## MDR.High.Grade10                      14.93904021 1024.45522248   0.015 0.988365
## MDR.High.Grade11                      13.71908090 1024.45333594   0.013 0.989315
## MDR.High.Grade12                      13.90649369 1024.45202849   0.014 0.989169
## MDR.High.Grade2                       26.17643651 1779.79890690   0.015 0.988266
## MDR.High.Grade3                       27.38323543 1779.79927891   0.015 0.987725
## MDR.High.Grade4                       27.35770079 1235.89670947   0.022 0.982340
## MDR.High.Grade5                       12.38861474 1024.45198934   0.012 0.990351
## MDR.High.Grade6                       12.11968107 1024.45199895   0.012 0.990561
## MDR.High.Grade7                       14.66224047 1024.45219989   0.014 0.988581
## MDR.High.Grade8                       14.41311537 1024.45196890   0.014 0.988775
## MDR.High.Grade9                       14.31571566 1024.45215080   0.014 0.988851
## Total.School.Enrollment                0.00017227    0.00031433   0.548 0.583646
## Income.LevelLow                        0.02972352    0.27023000   0.110 0.912415
## Income.LevelMedium                    -0.02679064    0.16720916  -0.160 0.872706
## Income.LevelUnclassified              -0.33925508    0.62801523  -0.540 0.589058
## EZ.Pay.Take.Up.Rate                   -0.34657275    0.43760594  -0.792 0.428376
## School.SponsorSponsoring               0.06654371    0.32328386   0.206 0.836918
```

```
## SPR.Product.TypeCosta Rica           -2.70092294    1.17371816  -2.301 0.021382
## SPR.Product.TypeEast Coast           -0.23452683    0.72437485  -0.324 0.746116
## SPR.Product.TypeIL History            0.38992695    1.24456050   0.313 0.754049
## SPR.Product.TypeInternational        17.14947448 1455.40019112   0.012 0.990598
## SPR.Product.TypeScience              -1.94668380    1.06592319  -1.826 0.067807
## SPR.New.ExistingNEW                  -1.40377095    0.15096353  -9.299  < 2e-16
## FPP                                  -0.01391518    0.00709445  -1.961 0.049830
## Total.Pax                                    NA            NA      NA       NA
## NumberOfMeetingswithParents           0.08746374    0.27195178   0.322 0.747744
## DifferenceTraveltoFirstMeeting       -0.00115482    0.00201083  -0.574 0.565766
## DifferenceTraveltoLastMeeting         0.00093512    0.00240457   0.389 0.697355
## SchoolGradeTypeLowHigh                       NA            NA      NA       NA
## SchoolGradeTypeLowMiddle              0.97952800    0.79166406   1.237 0.215975
## SchoolGradeTypeLowUndefined                  NA            NA      NA       NA
## SchoolGradeTypeHighHigh                      NA            NA      NA       NA
## SchoolGradeTypeHighMiddle            -1.03252205    0.80016716  -1.290 0.196918
## SchoolGradeTypeHighUndefined                 NA            NA      NA       NA
## SchoolGradeTypeE->H                -14.07355151  667.74097564  -0.021 0.983185
## SchoolGradeTypeE->M                  1.35198337    1.85254285   0.730 0.465513
## SchoolGradeTypeE->U                          NA            NA      NA       NA
## SchoolGradeTypeH->H                          NA            NA      NA       NA
## SchoolGradeTypeM->H                          NA            NA      NA       NA
## SchoolGradeTypeM->M                          NA            NA      NA       NA
## SchoolGradeTypeM->U                          NA            NA      NA       NA
## SchoolGradeTypeU->U                          NA            NA      NA       NA
## DepartureMonthFebruary               2.00337918    0.89906070   2.228 0.025860
## DepartureMonthJanuary                0.73782597    2.16792937   0.340 0.733603
## DepartureMonthJune                  -0.06578709    0.19877328  -0.331 0.740671
## DepartureMonthMarch                  0.34906406    0.23256168   1.501 0.133368
## DepartureMonthMay                    0.26625991    0.21456770   1.241 0.214638
## GroupGradeTypeLowHigh                        NA            NA      NA       NA
## GroupGradeTypeLowK                           NA            NA      NA       NA
## GroupGradeTypeLowMiddle             -0.07377596    0.69772285  -0.106 0.915790
## GroupGradeTypeLowPK                          NA            NA      NA       NA
## GroupGradeTypeLowUndefined                   NA            NA      NA       NA
## GroupGradeTypeHighHigh                       NA            NA      NA       NA
## GroupGradeTypeHighMiddle                     NA            NA      NA       NA
## GroupGradeTypeHighUndefined                  NA            NA      NA       NA
## GroupGradeTypeElementary->High     -16.25273913  597.42610614  -0.027 0.978297
## GroupGradeTypeElementary->Middle   -14.68980485  597.42395241  -0.025 0.980383
## GroupGradeTypeHigh->High                     NA            NA      NA       NA
## GroupGradeTypeK->Elementary          0.81830169    0.73764734   1.109 0.267284
## GroupGradeTypeK->High               -0.87617681    0.63668535  -1.376 0.168774
## GroupGradeTypeK->Middle                      NA            NA      NA       NA
## GroupGradeTypeMiddle->High          -0.18496060    0.53013906  -0.349 0.727171
## GroupGradeTypeMiddle->Middle                 NA            NA      NA       NA
## GroupGradeTypePK->Elementary                 NA            NA      NA       NA
## GroupGradeTypePK->High                       NA            NA      NA       NA
## GroupGradeTypePK->Middle                     NA            NA      NA       NA
## GroupGradeTypeUndefined->Undefined           NA            NA      NA       NA
## MajorProgramCodeH                   -1.10347877    0.41000905  -2.691 0.007116
## MajorProgramCodeI                  -20.03544513 1455.39946881  -0.014 0.989016
## MajorProgramCodeS                            NA            NA      NA       NA
## SingleGradeTripFlag                  0.70631657    0.39464879   1.790 0.073497
```

```
## FPP.to.School.enrollment             0.06600473    1.22095752   0.054 0.956888
## FPP.to.PAX                           2.18500933    1.78099067   1.227 0.219879
## Num.of.Non_FPP.PAX                          NA            NA      NA       NA
## SchoolSizeIndicatorM-L              -0.38918040    0.22975593  -1.694 0.090287
## SchoolSizeIndicatorS                -0.56573522    0.35726532  -1.584 0.113304
## SchoolSizeIndicatorS-M              -0.02617103    0.27898992  -0.094 0.925263
##
## (Intercept)
## From.Grade11
## From.Grade12
## From.Grade3
## From.Grade4
## From.Grade5
## From.Grade6
## From.Grade7
## From.Grade8
## From.Grade9
## To.Grade11
## To.Grade12
## To.Grade3
## To.Grade4
## To.Grade5                          .
## To.Grade6                          *
## To.Grade7                          **
## To.Grade8                          *
## To.Grade9                          *
## Is.Non.Annual.Yes                  ***
## Days
## Travel.TypeB
## Travel.TypeT
## Tuition
## FRP.Active                         ***
## FRP.Cancelled
## FRP.Take.up.percent.
## Cancelled.Pax
## Total.Discount.Pax
## Poverty.CodeA                      .
## Poverty.CodeB
## Poverty.CodeC
## Poverty.CodeD
## Poverty.CodeE                      *
## RegionHouston                      **
## RegionNorthern California          **
## RegionOther                        *
## RegionPacific Northwest            ***
## RegionSouthern California
## CRM.Segment10
## CRM.Segment11
## CRM.Segment2
## CRM.Segment3
## CRM.Segment4                       *
## CRM.Segment5
## CRM.Segment6                       .
## CRM.Segment7
```

```
## CRM.Segment8
## CRM.Segment9                        .
## School.TypeCHD
## School.TypePrivate non-Christian    .
## School.TypePUBLIC
## Parent.Meeting.Flag
## MDR.Low.Grade10
## MDR.Low.Grade2
## MDR.Low.Grade3
## MDR.Low.Grade4
## MDR.Low.Grade5
## MDR.Low.Grade6
## MDR.Low.Grade7
## MDR.Low.Grade8
## MDR.Low.Grade9
## MDR.Low.GradeK
## MDR.Low.GradePK
## MDR.High.Grade10
## MDR.High.Grade11
## MDR.High.Grade12
## MDR.High.Grade2
## MDR.High.Grade3
## MDR.High.Grade4
## MDR.High.Grade5
## MDR.High.Grade6
## MDR.High.Grade7
## MDR.High.Grade8
## MDR.High.Grade9
## Total.School.Enrollment
## Income.LevelLow
## Income.LevelMedium
## Income.LevelUnclassified
## EZ.Pay.Take.Up.Rate
## School.SponsorSponsoring
## SPR.Product.TypeCosta Rica          *
## SPR.Product.TypeEast Coast
## SPR.Product.TypeIL History
## SPR.Product.TypeInternational
## SPR.Product.TypeScience             .
## SPR.New.ExistingNEW                 ***
## FPP                                 *
## Total.Pax
## NumberOfMeetingswithParents
## DifferenceTraveltoFirstMeeting
## DifferenceTraveltoLastMeeting
## SchoolGradeTypeLowHigh
## SchoolGradeTypeLowMiddle
## SchoolGradeTypeLowUndefined
## SchoolGradeTypeHighHigh
## SchoolGradeTypeHighMiddle
## SchoolGradeTypeHighUndefined
## SchoolGradeTypeE->H
## SchoolGradeTypeE->M
## SchoolGradeTypeE->U
```

```
## SchoolGradeTypeH->H
## SchoolGradeTypeM->H
## SchoolGradeTypeM->M
## SchoolGradeTypeM->U
## SchoolGradeTypeU->U
## DepartureMonthFebruary             *
## DepartureMonthJanuary
## DepartureMonthJune
## DepartureMonthMarch
## DepartureMonthMay
## GroupGradeTypeLowHigh
## GroupGradeTypeLowK
## GroupGradeTypeLowMiddle
## GroupGradeTypeLowPK
## GroupGradeTypeLowUndefined
## GroupGradeTypeHighHigh
## GroupGradeTypeHighMiddle
## GroupGradeTypeHighUndefined
## GroupGradeTypeElementary->High
## GroupGradeTypeElementary->Middle
## GroupGradeTypeHigh->High
## GroupGradeTypeK->Elementary
## GroupGradeTypeK->High
## GroupGradeTypeK->Middle
## GroupGradeTypeMiddle->High
## GroupGradeTypeMiddle->Middle
## GroupGradeTypePK->Elementary
## GroupGradeTypePK->High
## GroupGradeTypePK->Middle
## GroupGradeTypeUndefined->Undefined
## MajorProgramCodeH                  **
## MajorProgramCodeI
## MajorProgramCodeS
## SingleGradeTripFlag                .
## FPP.to.School.enrollment
## FPP.to.PAX
## Num.of.Non_FPP.PAX
## SchoolSizeIndicatorM-L             .
## SchoolSizeIndicatorS
## SchoolSizeIndicatorS-M
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2562.0  on 1911  degrees of freedom
## Residual deviance: 1585.3  on 1798  degrees of freedom
## AIC: 1813.3
##
## Number of Fisher Scoring iterations: 14
```

Calculate the residual deviance of the logistic regression model

```
rd <- summary(glm)$deviance
```

```
1-pchisq(rd, 10)
```

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

```
Pred_glm <- predict(glm, newdata = testing, type = "response")
Pred_glm
```

```
##                1                2                3                4
## 0.943549395130744 0.239182638013544 0.108762299883053 0.980407330318662
##                5                6                7                8
## 0.989002660467714 0.997020134860928 0.996234055601536 0.990066102778185
##                9               10               11               12
## 0.976579784961268 0.974553860889194 0.988587629458595 0.980237709034740
##               13               14               15               16
## 0.960469664715716 0.175853976583844 0.783739589654620 0.614794391348061
##               17               18               19               20
## 0.685955122745975 0.856445930907250 0.997866278712733 0.970407385675307
##               21               22               23               24
## 0.863581026073298 0.616028744714555 0.893764420066341 0.678025532370937
##               25               26               27               28
## 0.908281338784264 0.257306506075458 0.846849862336561 0.914867479964259
##               29               30               31               32
## 0.752662147731452 0.788020893657631 0.676182438223754 0.950991018049791
##               33               34               35               36
## 0.666717847307360 0.979799036653464 0.321041154710799 0.825646905882494
##               37               38               39               40
## 0.968212209168926 0.856016380973707 0.854743389547701 0.271433342273535
##               41               42               43               44
## 0.861487944882053 0.079943237712096 0.749432005386640 0.962622190756532
##               45               46               47               48
## 0.132131587901951 0.604232056670226 0.625027042682266 0.905328096536996
##               49               50               51               52
## 0.726828532147976 0.626673833756808 0.654503390234970 0.951181652086082
##               53               54               55               56
## 0.946035609854008 0.907200029098772 0.651336874823446 0.999993245730082
##               57               58               59               60
## 0.971357610440126 0.906484125075764 0.999999916915052 0.890703144012248
##               61               62               63               64
## 0.791234153312159 0.196933019654696 0.999999912256089 0.956487559319492
##               65               66               67               68
## 0.828122847630010 0.010322709336315 0.663978026017563 0.342235586265546
##               69               70               71               72
## 0.938636642538723 0.906195814174384 0.028693497831758 0.883860780702507
##               73               74               75               76
## 0.966805435875010 0.549567563667628 0.465050902629198 0.947938894075476
##               77               78               79               80
## 0.886835644253170 0.027009054221888 0.399430941772040 0.747664686495688
##               81               82               83               84
```

```
## 0.680683381072975 0.631238016452861 0.392937570414664 0.935729829798918
##                85                86                87                88
## 0.544758655107662 0.917617923383777 0.755843019598037 0.492121148643967
##                89                90                91                92
## 0.860516188573199 0.917864888056006 0.770965554257756 0.019742356648541
##                93                94                95                96
## 0.412140024591746 0.585622737229283 0.702843439020569 0.448176841917436
##                97                98                99               100
## 0.841045650571339 0.706367936669277 0.612761165029347 0.918139223430978
##               101               102               103               104
## 0.863154772848697 0.955363453112704 0.876885323515480 0.205416167228884
##               105               106               107               108
## 0.878583280315950 0.575203797858352 0.418178974128445 0.629645253742194
##               109               110               111               112
## 0.102741025735532 0.796758102540633 0.538461045437892 0.149963431087716
##               113               114               115               116
## 0.322097421950279 0.617759827483875 0.265529500662911 0.809453083979464
##               117               118               119               120
## 0.305629343762405 0.846143031306650 0.784048624506773 0.939757113327650
##               121               122               123               124
## 0.944184998431825 0.951349644750789 0.583815709636325 0.138091415937468
##               125               126               127               128
## 0.180083471673547 0.120204678299727 0.891089490310088 0.733234862475493
##               129               130               131               132
## 0.545840734725719 0.581942147487357 0.919888639232885 0.180048491291751
##               133               134               135               136
## 0.901565844118636 0.813493590414075 0.852286649705313 0.739730351415301
##               137               138               139               140
## 0.614033374415664 0.799196025516390 0.969724077239082 0.498262792774597
##               141               142               143               144
## 0.902055974634408 0.905030742877859 0.871818338199550 0.919767244825480
##               145               146               147               148
## 0.944436870124281 0.945565041334918 0.871054580241951 0.143586412974224
##               149               150               151               152
## 0.064516798355681 0.419205018306344 0.239497217848746 0.922763974716168
##               153               154               155               156
## 0.370334052756902 0.915412809527879 0.945441604843032 0.950371981982137
##               157               158               159               160
## 0.803901423351882 0.474945964215043 0.959146371287870 0.120840241918898
##               161               162               163               164
## 0.794040813982932 0.682093746251384 0.759226701778580 0.000000011160121
##               165               166               167               168
## 0.982526284561257 0.686737289178471 0.204270924101139 0.034193443976677
##               169               170               171               172
## 0.912737372842896 0.745288874777402 0.698050073189565 0.312723372400661
##               173               174               175               176
## 0.463265883601841 0.103720525959033 0.093930140533799 0.942205536447622
##               177               178               179               180
## 0.426720412758522 0.801451943573801 0.570747873698113 0.423339938835134
##               181               182               183               184
## 0.191252350415612 0.636961564577050 0.571619492408457 0.955457718310350
##               185               186               187               188
## 0.818729026899701 0.807301058314707 0.953840804145413 0.828809994723997
##               189               190               191               192
```

```
## 0.961237993356450 0.000000012047223 0.991147320441768 0.433134828482719
##               193               194               195               196
## 0.960360486859722 0.950215250832419 0.987798523465084 0.359329817144928
##               197               198               199               200
## 0.230920371734811 0.955421875131027 0.942146151291734 0.939866794185385
##               201               202               203               204
## 0.055729942854804 0.933224822100780 0.936979389767389 0.940005037233552
##               205               206               207               208
## 0.396819668881103 0.069133866777884 0.936170292690158 0.893216095353640
##               209               210               211               212
## 0.972545984477288 0.919284020555551 0.410274304472037 0.756983208083906
##               213               214               215               216
## 0.936166945366165 0.917134606765963 0.900764478050368 0.978164983473950
##               217               218               219               220
## 0.294895114703965 0.862289294346518 0.657253100471737 0.890332244689119
##               221               222               223               224
## 0.129987624079612 0.939012363215957 0.913650464042068 0.523417901144869
##               225               226               227               228
## 0.955757595307187 0.966612063676483 0.978661835255792 0.870222684381501
##               229               230               231               232
## 0.877580364903844 0.833901905005780 0.808178903052834 0.911465798485652
##               233               234               235               236
## 0.923477995665646 0.919699278218245 0.946073889921409 0.962843776825801
##               237               238               239               240
## 0.642315986121397 0.787437526453480 0.931646041457379 0.478174962755468
##               241               242               243               244
## 0.912492028948328 0.959927187825454 0.981305137784908 0.893915259496651
##               245               246               247               248
## 0.151700036584927 0.759689466408973 0.921185916994093 0.485144851313988
##               249               250               251               252
## 0.893628693265045 0.956358381469000 0.971467214607187 0.876695722325074
##               253               254               255               256
## 0.938685569294265 0.946730688677525 0.259170129221439 0.907965105573008
##               257               258               259               260
## 0.888465244171172 0.588014444252893 0.922580317108206 0.876379513668443
##               261               262               263               264
## 0.958715785741927 0.968061452359660 0.951721362600522 0.050707550468668
##               265               266               267               268
## 0.786447764121746 0.755993928795590 0.495229726913357 0.456528880074536
##               269               270               271               272
## 0.136677632199660 0.301505794255497 0.918575639130879 0.066361296352971
##               273               274               275               276
## 0.057202653384743 0.913496234885466 0.991196359788070 0.828930458852129
##               277               278               279               280
## 0.134149315039172 0.826694621337353 0.859481470479070 0.216777930217499
##               281               282               283               284
## 0.660119638376081 0.222790685701594 0.118184763012116 0.911275126939128
##               285               286               287               288
## 0.931423400352712 0.940289002868559 0.374905391507132 0.016524968724286
##               289               290               291               292
## 0.037610047067895 0.642419452868131 0.126436552922618 0.230351643369495
##               293               294               295               296
## 0.351278157942021 0.677498503893210 0.869159048152137 0.673271996404077
##               297               298               299               300
```

```
## 0.449777201706239 0.433959071707087 0.931591254698567 0.958869730315863
##               301               302               303               304
## 0.475575510631028 0.668461811629166 0.470539911018268 0.832841945121818
##               305               306               307               308
## 0.401737699035964 0.892097025473945 0.854065549188202 0.939290301212359
##               309               310               311               312
## 0.892384956533648 0.054462686115135 0.212132181148210 0.921326796932284
##               313               314               315               316
## 0.944878991736659 0.910748778559204 0.604802610846618 0.844414893033881
##               317               318               319               320
## 0.743560618198189 0.941260231677789 0.466434771054169 0.184389986413618
##               321               322               323               324
## 0.127258895910480 0.849780269799034 0.728930129512134 0.697233698100129
##               325               326               327               328
## 0.764478543120742 0.790982507562724 0.146014757223082 0.321215999854068
##               329               330               331               332
## 0.679303511792973 0.018733445339230 0.123270204099210 0.979434141181763
##               333               334               335               336
## 0.901484548624760 0.283963145478658 0.859793118314169 0.571647719966459
##               337               338               339               340
## 0.216583683413073 0.477584292913919 0.142098789541748 0.883533041877725
##               341               342               343               344
## 0.027038311652638 0.448196357133293 0.059895212093148 0.839165512615069
##               345               346               347               348
## 0.700370775782001 0.645659304997757 0.262872557533151 0.344518578216755
##               349               350               351               352
## 0.504916601750659 0.273257669378689 0.900095978307629 0.707056387858336
##               353               354               355               356
## 0.860843141264472 0.186082785272322 0.839028508085374 0.296140088681750
##               357               358               359               360
## 0.537510906411920 0.082406330802067 0.380086372435329 0.509642496259755
##               361               362               363               364
## 0.133192744688786 0.757565542908642 0.052044179010070 0.597622264820706
##               365               366               367               368
## 0.090685095050115 0.764780623672590 0.046166553637982 0.288460834394461
##               369               370               371               372
## 0.551430941659818 0.186982030134144 0.626506599214848 0.788665333310228
##               373               374               375               376
## 0.827912894757957 0.118226172025652 0.207964248968784 0.862174797868090
##               377               378               379               380
## 0.826095888804048 0.051094869940977 0.723028051757931 0.437911331806582
##               381               382               383               384
## 0.173531119956571 0.171433899890088 0.189284247727966 0.901311703618432
##               385               386               387               388
## 0.516372606486622 0.114207773964774 0.192643285392641 0.018423594770390
##               389               390               391               392
## 0.446389779671742 0.083625055665414 0.203224361355279 0.866621743076963
##               393               394               395               396
## 0.925222335800069 0.139156163500612 0.083682608457924 0.811973054281091
##               397               398               399               400
## 0.902893237740020 0.069866617902446 0.916490836302422 0.055426625644935
##               401               402               403               404
## 0.884508762800081 0.662812508295650 0.877047045873314 0.709882787827173
##               405               406               407               408
```

```
## 0.401555242084139 0.071355633169594 0.168262324920424 0.673134498462098
##               409               410               411               412
## 0.047127248538317 0.929658691987951 0.853799104307045 0.342567506777037
##               413               414               415               416
## 0.809053801999902 0.772002935309682 0.455551974998226 0.539406051860059
##               417               418               419               420
## 0.420186917790957 0.519294411249082 0.481386979595327 0.794369626791222
##               421               422               423               424
## 0.199194354081189 0.020862986632721 0.281074393633301 0.118002402631075
##               425               426               427               428
## 0.522342871450800 0.886652558842714 0.022561454358701 0.718240383705063
##               429               430               431               432
## 0.769231730982746 0.533515742610146 0.382956854143697 0.492026760071994
##               433               434               435               436
## 0.656440089320591 0.126801460774929 0.465047546470798 0.138264764470487
##               437               438               439               440
## 0.009351696497699 0.759438105240704 0.916289188232523 0.898661987630693
##               441               442               443               444
## 0.830439498383471 0.911159587204372 0.425024855569400 0.073533276281426
##               445               446               447               448
## 0.220346505008971 0.944963808313009 0.939033301676470 0.671820996011272
##               449               450               451               452
## 0.754884272714543 0.009792084071684 0.068135813254198 0.143244771778702
##               453               454               455               456
## 0.251432235688260 0.704703466286405 0.319162678943655 0.288955770368273
##               457               458               459               460
## 0.739924107697722 0.556196936345183 0.154478847171673 0.137778077378265
##               461               462               463               464
## 0.738010804442429 0.166713532701196 0.602776118069890 0.526701499168514
##               465               466               467               468
## 0.107724502934701 0.471250939882711 0.143672030430142 0.473835334669340
##               469               470               471               472
## 0.672471806543425 0.000000009835378 0.022828923250808 0.899157189702597
##               473               474               475               476
## 0.730483055488314 0.225607927208495 0.393051987730514 0.802425288329903
##               477
## 0.570629485928371
```

```
Class_glm <- ifelse(Pred_glm >= 0.5, "YES", "NO")
Class_glm
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## "YES"  "NO"  "NO" "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES"
##    14    15    16    17    18    19    20    21    22    23    24    25    26
##  "NO" "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES"  "NO"
##    27    28    29    30    31    32    33    34    35    36    37    38    39
## "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES"  "NO" "YES" "YES" "YES" "YES"
##    40    41    42    43    44    45    46    47    48    49    50    51    52
##  "NO" "YES"  "NO" "YES" "YES"  "NO" "YES" "YES" "YES" "YES" "YES" "YES" "YES"
##    53    54    55    56    57    58    59    60    61    62    63    64    65
## "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES"  "NO" "YES" "YES" "YES"
##    66    67    68    69    70    71    72    73    74    75    76    77    78
##  "NO" "YES"  "NO" "YES" "YES"  "NO" "YES" "YES" "YES"  "NO" "YES" "YES"  "NO"
##    79    80    81    82    83    84    85    86    87    88    89    90    91
##  "NO" "YES" "YES" "YES"  "NO" "YES" "YES" "YES" "YES"  "NO" "YES" "YES" "YES"
##    92    93    94    95    96    97    98    99   100   101   102   103   104
##  "NO"  "NO" "YES" "YES"  "NO" "YES" "YES" "YES" "YES" "YES" "YES" "YES"  "NO"
##   105   106   107   108   109   110   111   112   113   114   115   116   117
## "YES" "YES"  "NO" "YES"  "NO" "YES" "YES"  "NO"  "NO" "YES"  "NO" "YES"  "NO"
##   118   119   120   121   122   123   124   125   126   127   128   129   130
## "YES" "YES" "YES" "YES" "YES" "YES"  "NO"  "NO"  "NO" "YES" "YES" "YES" "YES"
##   131   132   133   134   135   136   137   138   139   140   141   142   143
## "YES"  "NO" "YES" "YES" "YES" "YES" "YES" "YES" "YES"  "NO" "YES" "YES" "YES"
##   144   145   146   147   148   149   150   151   152   153   154   155   156
## "YES" "YES" "YES" "YES"  "NO"  "NO"  "NO"  "NO" "YES"  "NO" "YES" "YES" "YES"
##   157   158   159   160   161   162   163   164   165   166   167   168   169
## "YES"  "NO" "YES"  "NO" "YES" "YES" "YES"  "NO" "YES" "YES"  "NO"  "NO" "YES"
##   170   171   172   173   174   175   176   177   178   179   180   181   182
## "YES" "YES"  "NO"  "NO"  "NO"  "NO" "YES"  "NO" "YES" "YES"  "NO"  "NO" "YES"
##   183   184   185   186   187   188   189   190   191   192   193   194   195
## "YES" "YES" "YES" "YES" "YES" "YES" "YES"  "NO" "YES"  "NO" "YES" "YES" "YES"
##   196   197   198   199   200   201   202   203   204   205   206   207   208
##  "NO"  "NO" "YES" "YES" "YES"  "NO" "YES" "YES" "YES"  "NO"  "NO" "YES" "YES"
##   209   210   211   212   213   214   215   216   217   218   219   220   221
## "YES" "YES"  "NO" "YES" "YES" "YES" "YES" "YES"  "NO" "YES" "YES" "YES"  "NO"
##   222   223   224   225   226   227   228   229   230   231   232   233   234
## "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES" "YES"
##   235   236   237   238   239   240   241   242   243   244   245   246   247
## "YES" "YES" "YES" "YES" "YES"  "NO" "YES" "YES" "YES" "YES"  "NO" "YES" "YES"
##   248   249   250   251   252   253   254   255   256   257   258   259   260
##  "NO" "YES" "YES" "YES" "YES" "YES" "YES"  "NO" "YES" "YES" "YES" "YES" "YES"
##   261   262   263   264   265   266   267   268   269   270   271   272   273
## "YES" "YES" "YES"  "NO" "YES" "YES"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"
##   274   275   276   277   278   279   280   281   282   283   284   285   286
## "YES" "YES" "YES"  "NO" "YES" "YES"  "NO" "YES"  "NO"  "NO" "YES" "YES" "YES"
##   287   288   289   290   291   292   293   294   295   296   297   298   299
##  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO" "YES" "YES" "YES"  "NO"  "NO" "YES"
##   300   301   302   303   304   305   306   307   308   309   310   311   312
## "YES"  "NO" "YES"  "NO" "YES"  "NO" "YES" "YES" "YES" "YES"  "NO"  "NO" "YES"
##   313   314   315   316   317   318   319   320   321   322   323   324   325
## "YES" "YES" "YES" "YES" "YES" "YES"  "NO"  "NO"  "NO" "YES" "YES" "YES" "YES"
```

```
##   326   327   328   329   330   331   332   333   334   335   336   337   338
## "YES"  "NO"  "NO" "YES"  "NO"  "NO" "YES" "YES"  "NO" "YES" "YES"  "NO"  "NO"
##   339   340   341   342   343   344   345   346   347   348   349   350   351
##  "NO" "YES"  "NO"  "NO"  "NO" "YES" "YES" "YES"  "NO"  "NO" "YES"  "NO" "YES"
##   352   353   354   355   356   357   358   359   360   361   362   363   364
## "YES" "YES"  "NO" "YES"  "NO" "YES"  "NO"  "NO" "YES"  "NO" "YES"  "NO" "YES"
##   365   366   367   368   369   370   371   372   373   374   375   376   377
##  "NO" "YES"  "NO"  "NO" "YES"  "NO" "YES" "YES" "YES"  "NO"  "NO" "YES" "YES"
##   378   379   380   381   382   383   384   385   386   387   388   389   390
##  "NO" "YES"  "NO"  "NO"  "NO"  "NO" "YES" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"
##   391   392   393   394   395   396   397   398   399   400   401   402   403
##  "NO" "YES" "YES"  "NO"  "NO" "YES" "YES"  "NO" "YES"  "NO" "YES" "YES" "YES"
##   404   405   406   407   408   409   410   411   412   413   414   415   416
## "YES"  "NO"  "NO"  "NO" "YES"  "NO" "YES" "YES"  "NO" "YES" "YES"  "NO" "YES"
##   417   418   419   420   421   422   423   424   425   426   427   428   429
##  "NO" "YES"  "NO" "YES"  "NO"  "NO"  "NO"  "NO" "YES" "YES"  "NO" "YES" "YES"
##   430   431   432   433   434   435   436   437   438   439   440   441   442
## "YES"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO" "YES" "YES" "YES" "YES" "YES"
##   443   444   445   446   447   448   449   450   451   452   453   454   455
##  "NO"  "NO"  "NO" "YES" "YES" "YES" "YES"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"
##   456   457   458   459   460   461   462   463   464   465   466   467   468
##  "NO" "YES" "YES"  "NO"  "NO" "YES"  "NO" "YES" "YES"  "NO"  "NO"  "NO"  "NO"
##   469   470   471   472   473   474   475   476   477
## "YES"  "NO"  "NO" "YES" "YES"  "NO"  "NO" "YES" "YES"
```

Neural network

Normalize data before training a neural network.

```
library(dplyr)
myscale <- function(x) {
(x - min(x)) / (max(x) - min(x))
}
tr_nnet <- tr %>% mutate_if(is.numeric, myscale)
```

Split our normalized data into a training set and a test set.

```
set.seed(1234)
ind <- sample(2, nrow(tr_nnet), replace = T, prob = c(0.7, 0.3))
train_nnet <- tr_nnet[ind == 1, ]
test_nnet <- tr_nnet[ind == 2, ]
```

Neural network Model

Then create neural network model,change the size=3, maxit=100.

```
library(nnet)
nnModel <- nnet(target ~ ., data = train_nnet, linout = FALSE,
                size = 3, decay = 0.01, maxit = 100)
```

```
## # weights:  427
## initial  value 1192.460537
## iter  10 value 826.265979
## iter  20 value 704.712470
## iter  30 value 636.292712
## iter  40 value 607.275689
## iter  50 value 594.225092
## iter  60 value 587.163914
## iter  70 value 581.897615
## iter  80 value 566.956406
## iter  90 value 556.272286
## iter 100 value 550.496222
## final  value 550.496222
## stopped after 100 iterations
```

```
summary(nnModel)
```

```
## a 140-3-1 network with 427 weights
## options were - entropy fitting  decay=0.01
##    b->h1   i1->h1   i2->h1   i3->h1   i4->h1   i5->h1   i6->h1   i7->h1
##    -2.25     0.31     0.14    -0.03    -0.96     1.48     1.11    -2.27
##   i8->h1   i9->h1  i10->h1  i11->h1  i12->h1  i13->h1  i14->h1  i15->h1
##     0.08    -1.38     2.33     0.55     0.00     0.49    -1.73     3.32
##  i16->h1  i17->h1  i18->h1  i19->h1  i20->h1  i21->h1  i22->h1  i23->h1
##    -1.99     0.40    -4.72    -6.49    -1.40    -5.16    -2.77    -6.08
##  i24->h1  i25->h1  i26->h1  i27->h1  i28->h1  i29->h1  i30->h1  i31->h1
##     8.53     0.10    -4.14    -1.25     2.59     1.41     2.30    -5.54
##  i32->h1  i33->h1  i34->h1  i35->h1  i36->h1  i37->h1  i38->h1  i39->h1
##    -0.96     4.23     2.65    -1.84     0.02    -3.84     3.82     4.03
##  i40->h1  i41->h1  i42->h1  i43->h1  i44->h1  i45->h1  i46->h1  i47->h1
##    -1.32    -4.41     2.53    -3.22     0.05     8.37     4.48     1.88
##  i48->h1  i49->h1  i50->h1  i51->h1  i52->h1  i53->h1  i54->h1  i55->h1
##     3.22    -0.55     2.37    -2.12     2.28    -0.20    -1.40     4.13
##  i56->h1  i57->h1  i58->h1  i59->h1  i60->h1  i61->h1  i62->h1  i63->h1
##    -3.62     0.37    -1.49    -1.22     3.48     1.96    -1.61    -1.01
##  i64->h1  i65->h1  i66->h1  i67->h1  i68->h1  i69->h1  i70->h1  i71->h1
##     1.11    -0.01    -6.40    -0.01     0.24     0.22    -1.84    -3.24
##  i72->h1  i73->h1  i74->h1  i75->h1  i76->h1  i77->h1  i78->h1  i79->h1
##     5.14    -1.38     4.05     2.37     0.25     3.71     2.60     2.42
##  i80->h1  i81->h1  i82->h1  i83->h1  i84->h1  i85->h1  i86->h1  i87->h1
##    -0.52     0.67     0.78    -0.54    -1.08     0.65    -6.40     5.30
##  i88->h1  i89->h1  i90->h1  i91->h1  i92->h1  i93->h1  i94->h1  i95->h1
##     5.24    -0.25     0.77     4.21    -1.64    -0.87    -0.24    -2.62
##  i96->h1  i97->h1  i98->h1  i99->h1 i100->h1 i101->h1 i102->h1 i103->h1
##     0.64     1.03    -1.42     3.55    -0.31    -1.67     0.38    -2.85
## i104->h1 i105->h1 i106->h1 i107->h1 i108->h1 i109->h1 i110->h1 i111->h1
##     1.57    -0.22     8.55    -0.46    -2.48     4.80    -0.71     1.84
## i112->h1 i113->h1 i114->h1 i115->h1 i116->h1 i117->h1 i118->h1 i119->h1
##    -1.64     0.26    -1.06     0.48    -1.27    -0.03     0.45    -1.39
## i120->h1 i121->h1 i122->h1 i123->h1 i124->h1 i125->h1 i126->h1 i127->h1
##    -0.15     1.78     0.82    -2.34    -0.19    -0.26     0.54    -1.79
## i128->h1 i129->h1 i130->h1 i131->h1 i132->h1 i133->h1 i134->h1 i135->h1
##     1.01    -0.24     0.46    -4.06    -1.08     1.37     4.92     0.65
## i136->h1 i137->h1 i138->h1 i139->h1 i140->h1
##    -0.89     2.52     2.35     5.83     4.99
##    b->h2   i1->h2   i2->h2   i3->h2   i4->h2   i5->h2   i6->h2   i7->h2
##     0.02    -0.57    -0.93    -0.04     0.11     0.11     0.10    -0.76
##   i8->h2   i9->h2  i10->h2  i11->h2  i12->h2  i13->h2  i14->h2  i15->h2
##     0.55    -3.66     0.67    -0.34    -0.01     0.65     0.61    -2.99
##  i16->h2  i17->h2  i18->h2  i19->h2  i20->h2  i21->h2  i22->h2  i23->h2
##     2.44     0.49     1.36    -6.28    -0.39    -0.25     2.90    -3.76
##  i24->h2  i25->h2  i26->h2  i27->h2  i28->h2  i29->h2  i30->h2  i31->h2
##     1.86     0.35    -1.21    -1.84     4.39    -0.17     0.02     1.20
##  i32->h2  i33->h2  i34->h2  i35->h2  i36->h2  i37->h2  i38->h2  i39->h2
##     0.42    -1.29    -5.84    -2.03    -3.33    -4.03    -2.31     0.08
##  i40->h2  i41->h2  i42->h2  i43->h2  i44->h2  i45->h2  i46->h2  i47->h2
##     0.40    -0.62     0.63     5.94     0.22    -0.52    -1.00    -2.48
##  i48->h2  i49->h2  i50->h2  i51->h2  i52->h2  i53->h2  i54->h2  i55->h2
```

```
##    -1.98    -3.10     0.48    -0.06    -1.55    -0.55    -2.76     0.27
##  i56->h2  i57->h2  i58->h2  i59->h2  i60->h2  i61->h2  i62->h2  i63->h2
##     2.79     2.73     0.98     2.13    -2.16    -2.05     0.39     1.81
##  i64->h2  i65->h2  i66->h2  i67->h2  i68->h2  i69->h2  i70->h2  i71->h2
##    -0.11     0.10    -0.65    -0.01     0.01    -0.19    -1.33    -3.87
##  i72->h2  i73->h2  i74->h2  i75->h2  i76->h2  i77->h2  i78->h2  i79->h2
##     1.02     3.40     1.77     8.52    -0.74    -0.66     1.85     2.17
##  i80->h2  i81->h2  i82->h2  i83->h2  i84->h2  i85->h2  i86->h2  i87->h2
##     2.90     0.19     2.17     0.95    -0.25    -0.26    -3.40    -0.84
##  i88->h2  i89->h2  i90->h2  i91->h2  i92->h2  i93->h2  i94->h2  i95->h2
##    -0.11     2.41    -4.85    -0.65    -0.11     0.04    -0.15    -1.23
##  i96->h2  i97->h2  i98->h2  i99->h2 i100->h2 i101->h2 i102->h2 i103->h2
##     0.91    -0.95    -0.40     0.74    -1.40    -0.13    -0.72     0.13
## i104->h2 i105->h2 i106->h2 i107->h2 i108->h2 i109->h2 i110->h2 i111->h2
##     0.56    -0.15    -1.67    -0.10    -1.00    -2.59    -0.66    -2.54
## i112->h2 i113->h2 i114->h2 i115->h2 i116->h2 i117->h2 i118->h2 i119->h2
##     0.36     0.49     1.75     0.48     1.07     0.03     0.46    -1.75
## i120->h2 i121->h2 i122->h2 i123->h2 i124->h2 i125->h2 i126->h2 i127->h2
##    -0.52    -2.61    -1.42     0.74     1.01     0.62    -0.13    -1.89
## i128->h2 i129->h2 i130->h2 i131->h2 i132->h2 i133->h2 i134->h2 i135->h2
##     4.01    -0.34     0.46    -0.03    -0.37    -0.09     1.11     2.52
## i136->h2 i137->h2 i138->h2 i139->h2 i140->h2
##     3.46     4.39     0.09    -1.39     1.45
##    b->h3   i1->h3   i2->h3   i3->h3   i4->h3   i5->h3   i6->h3   i7->h3
##     0.00    -1.99    -0.02     0.68     1.90    -0.15     1.73    -1.89
##   i8->h3   i9->h3  i10->h3  i11->h3  i12->h3  i13->h3  i14->h3  i15->h3
##     0.57    -1.75    -1.41     0.67     0.02     1.84     0.01    -1.04
##  i16->h3  i17->h3  i18->h3  i19->h3  i20->h3  i21->h3  i22->h3  i23->h3
##     1.03     0.29     0.28     0.21    -5.47    -1.50    -0.13    -0.17
##  i24->h3  i25->h3  i26->h3  i27->h3  i28->h3  i29->h3  i30->h3  i31->h3
##    -0.47    -2.08    -5.19    -2.13     1.04    -3.64     2.18     1.56
##  i32->h3  i33->h3  i34->h3  i35->h3  i36->h3  i37->h3  i38->h3  i39->h3
##    -1.80     1.68    -0.89    -5.87    -2.38    -2.23     4.81    -3.20
##  i40->h3  i41->h3  i42->h3  i43->h3  i44->h3  i45->h3  i46->h3  i47->h3
##    -2.68     6.07     0.10    -4.88    -0.51    -1.47     2.73    -1.37
##  i48->h3  i49->h3  i50->h3  i51->h3  i52->h3  i53->h3  i54->h3  i55->h3
##     0.61     1.60     0.57     2.04     4.93     0.00     0.59     1.02
##  i56->h3  i57->h3  i58->h3  i59->h3  i60->h3  i61->h3  i62->h3  i63->h3
##    -0.34    -0.55    -5.22     1.83     3.39    -3.74     0.97     1.35
##  i64->h3  i65->h3  i66->h3  i67->h3  i68->h3  i69->h3  i70->h3  i71->h3
##     0.00     0.71    -5.59     0.02    -0.03     0.08    -2.39    -0.82
##  i72->h3  i73->h3  i74->h3  i75->h3  i76->h3  i77->h3  i78->h3  i79->h3
##     3.29     0.05     4.69    -3.21     0.30     2.61    -0.16    -3.34
##  i80->h3  i81->h3  i82->h3  i83->h3  i84->h3  i85->h3  i86->h3  i87->h3
##     1.18     1.42     3.55     3.47     0.55     0.22    -1.82     1.24
##  i88->h3  i89->h3  i90->h3  i91->h3  i92->h3  i93->h3  i94->h3  i95->h3
##     1.35     1.47     0.05    -1.88    -2.85     1.68    -1.28    -2.24
##  i96->h3  i97->h3  i98->h3  i99->h3 i100->h3 i101->h3 i102->h3 i103->h3
##     3.16    -2.82     0.26     0.57    -0.22    -2.88     0.37     2.63
## i104->h3 i105->h3 i106->h3 i107->h3 i108->h3 i109->h3 i110->h3 i111->h3
##    -1.30    -1.30     0.65     0.47    -2.15    -6.60    -3.55    -3.82
## i112->h3 i113->h3 i114->h3 i115->h3 i116->h3 i117->h3 i118->h3 i119->h3
##     0.97    -0.11     1.35     0.15    -0.23     2.37     0.12     1.23
## i120->h3 i121->h3 i122->h3 i123->h3 i124->h3 i125->h3 i126->h3 i127->h3
```

```
##     3.67    -3.82     0.50     0.41     0.08     0.76    -0.93     0.72
## i128->h3 i129->h3 i130->h3 i131->h3 i132->h3 i133->h3 i134->h3 i135->h3
##     1.07    -0.47     0.15     1.07     0.65     1.63     1.93     1.03
## i136->h3 i137->h3 i138->h3 i139->h3 i140->h3
##     1.25     1.09    -1.28    -2.02     9.37
##  b->o h1->o h2->o h3->o
## -1.46  4.98  6.41 -4.16
```

Use wts to get the best weights found and fitted.values to get the fitted values on training data

```
nnModel$wts
```

```
##   [1] -2.250486353  0.311888828  0.144685472 -0.028760727 -0.961285710
##   [6]  1.480709348  1.113064774 -2.269894196  0.078709930 -1.384568765
##  [11]  2.325443799  0.545633642  0.002386341  0.488057402 -1.733997941
##  [16]  3.324335580 -1.987076291  0.395910853 -4.724608424 -6.487916089
##  [21] -1.396421678 -5.160977518 -2.766600205 -6.080817356  8.526991165
##  [26]  0.095490855 -4.143409847 -1.252519835  2.591487804  1.407643712
##  [31]  2.296178550 -5.543570615 -0.961506881  4.234733669  2.646310711
##  [36] -1.836570280  0.015764964 -3.844331648  3.822808616  4.033183857
##  [41] -1.322122052 -4.412548577  2.533192966 -3.220561492  0.054701988
##  [46]  8.366174981  4.483332282  1.883605383  3.220934195 -0.554362445
##  [51]  2.370555925 -2.117795515  2.278994459 -0.196441124 -1.401949025
##  [56]  4.128352093 -3.624260214  0.371587501 -1.489137467 -1.221145427
##  [61]  3.479695434  1.960202974 -1.608537720 -1.005551587  1.109220983
##  [66] -0.007269331 -6.400821488 -0.010761077  0.239076015  0.218876730
##  [71] -1.839125218 -3.235697771  5.136815276 -1.384743304  4.046333840
##  [76]  2.368708675  0.245743701  3.709952467  2.600657290  2.419041952
##  [81] -0.518593671  0.670410242  0.781581054 -0.540246768 -1.081145817
##  [86]  0.649538229 -6.397770930  5.301846580  5.239455281 -0.251493416
##  [91]  0.769659933  4.210556144 -1.641193434 -0.865022505 -0.240990707
##  [96] -2.623912463  0.641053539  1.034451549 -1.416018944  3.546344791
## [101] -0.309738867 -1.668068994  0.375790088 -2.850998500  1.572757390
## [106] -0.224960876  8.545652077 -0.459342601 -2.478207915  4.797704507
## [111] -0.705606492  1.840817322 -1.640215800  0.262510920 -1.060409010
## [116]  0.479273897 -1.267154969 -0.026897669  0.448447183 -1.388694069
## [121] -0.148438601  1.783929675  0.822302007 -2.337216925 -0.192532339
## [126] -0.261017802  0.540121848 -1.785985963  1.010259825 -0.236791945
## [131]  0.459873847 -4.061896659 -1.080464383  1.368293607  4.921006777
## [136]  0.645509560 -0.891964451  2.520603315  2.354901368  5.830244813
## [141]  4.986198790  0.018435213 -0.568639819 -0.931944474 -0.039034436
## [146]  0.105305270  0.113481453  0.103839903 -0.758291153  0.550515775
## [151] -3.663173435  0.667128913 -0.344175302 -0.008565268  0.649961230
## [156]  0.607540882 -2.990962222  2.438274395  0.494021723  1.362028656
## [161] -6.280089639 -0.387976403 -0.252142648  2.899538542 -3.758222003
## [166]  1.859440888  0.349942959 -1.209241202 -1.843625036  4.385690316
## [171] -0.170826710  0.022073249  1.201195734  0.419770830 -1.289750749
## [176] -5.838514107 -2.027411735 -3.331442229 -4.029357925 -2.311050000
## [181]  0.079714072  0.395592297 -0.615864408  0.625514067  5.943921827
## [186]  0.222681300 -0.520815258 -0.998845374 -2.480168512 -1.976031262
## [191] -3.098018544  0.476480901 -0.064667028 -1.545657921 -0.552566364
## [196] -2.762262532  0.273866272  2.794085616  2.731641940  0.983862500
## [201]  2.129619082 -2.163695284 -2.047699405  0.387807871  1.812246663
## [206] -0.114357297  0.102321854 -0.647191642 -0.008800943  0.011186677
## [211] -0.187296149 -1.328969459 -3.874404594  1.020518249  3.403279646
## [216]  1.771179284  8.520112073 -0.744248477 -0.657148823  1.845419409
## [221]  2.173154278  2.896511559  0.188279823  2.165646252  0.947478412
## [226] -0.252574998 -0.261580134 -3.404849383 -0.842710158 -0.107312326
## [231]  2.414322690 -4.848406464 -0.645251359 -0.112318672  0.042791744
## [236] -0.154954349 -1.229633709  0.912913991 -0.952989674 -0.399871458
## [241]  0.740698777 -1.404003379 -0.133048417 -0.723205521  0.129804959
## [246]  0.560585488 -0.147332445 -1.669553783 -0.096813624 -1.003020157
```

```
## [251] -2.591118459 -0.657858331 -2.535816795  0.363252571  0.491305564
## [256]  1.753229091  0.479755982  1.074959022  0.030533315  0.464100679
## [261] -1.752900586 -0.523421489 -2.612140143 -1.422585609  0.735602537
## [266]  1.014930050  0.617770205 -0.125775355 -1.885528555  4.007970279
## [271] -0.338303838  0.458967025 -0.029164118 -0.367627586 -0.092805640
## [276]  1.113797104  2.520094895  3.460650985  4.394951753  0.093990301
## [281] -1.386210719  1.446042669 -0.004709097 -1.989449690 -0.020159095
## [286]  0.681067407  1.904005274 -0.150353948  1.731198455 -1.893745871
## [291]  0.573828162 -1.746620879 -1.408855862  0.669693554  0.023661852
## [296]  1.836279007  0.006925520 -1.041944334  1.031904181  0.291065694
## [301]  0.282333917  0.212305365 -5.468695664 -1.499119903 -0.132923762
## [306] -0.167697703 -0.471733742 -2.081289764 -5.193082014 -2.129550862
## [311]  1.044820336 -3.641323921  2.180883012  1.561634596 -1.797084265
## [316]  1.679991954 -0.889787978 -5.865481316 -2.377598272 -2.225194986
## [321]  4.811914830 -3.196157497 -2.677799195  6.068158421  0.103470614
## [326] -4.880698239 -0.505884928 -1.469964145  2.727871444 -1.372552125
## [331]  0.607974677  1.602074813  0.569348397  2.037240307  4.927598789
## [336] -0.004907194  0.591524504  1.022146836 -0.343141562 -0.551447121
## [341] -5.218850743  1.828166499  3.389229641 -3.743668044  0.970119352
## [346]  1.347448401 -0.002555834  0.714914252 -5.593522798  0.017785316
## [351] -0.026608954  0.078020957 -2.393694526 -0.820849224  3.285255231
## [356]  0.047446089  4.687234087 -3.212259879  0.296777997  2.607359251
## [361] -0.162896099 -3.336584485  1.180215258  1.424635260  3.554767966
## [366]  3.466702191  0.548653944  0.224410316 -1.818784069  1.239773562
## [371]  1.353495591  1.471125283  0.049575343 -1.875409038 -2.850303556
## [376]  1.677208972 -1.284634984 -2.236288224  3.163978859 -2.824459616
## [381]  0.258377749  0.572087666 -0.216082731 -2.877643106  0.370237996
## [386]  2.625072759 -1.304257114 -1.296738923  0.653338069  0.473445707
## [391] -2.148919576 -6.602901273 -3.553140794 -3.823208841  0.965011007
## [396] -0.109691774  1.354389853  0.145730388 -0.232108997  2.366465133
## [401]  0.121725594  1.231536323  3.667779658 -3.816097852  0.496525603
## [406]  0.408674443  0.078749684  0.759184325 -0.927821835  0.721037112
## [411]  1.074554850 -0.473876782  0.151717319  1.067020164  0.652002879
## [416]  1.629839644  1.934912833  1.032682470  1.249283844  1.094026100
## [421] -1.281747188 -2.017195330  9.371892874 -1.456861675  4.979857930
## [426]  6.413766311 -4.156421260
```

```
nnModel$fitted.values
```

```
##             [,1]
## 1    0.870711093
## 2    0.689392953
## 3    0.690083000
## 4    0.004974415
## 5    0.188948716
## 6    0.281165662
## 7    0.005189271
## 8    0.992038230
## 9    0.996853956
## 10   0.996534648
## 11   0.996905763
## 12   0.996197074
## 13   0.996864985
## 14   0.995547475
```

```
## 15   0.994767993
## 16   0.996879206
## 17   0.994546591
## 18   0.996896147
## 19   0.883193329
## 20   0.996853805
## 21   0.997332360
## 22   0.990227875
## 23   0.961609504
## 24   0.999931393
## 25   0.995797020
## 26   0.999927670
## 27   0.862811460
## 28   0.988920875
## 29   0.996896683
## 30   0.996909215
## 31   0.986285579
## 32   0.996906638
## 33   0.996606128
## 34   0.243544666
## 35   0.996780467
## 36   0.996750104
## 37   0.996919102
## 38   0.208098893
## 39   0.996921106
## 40   0.944460548
## 41   0.996918860
## 42   0.995206267
## 43   0.996496985
## 44   0.974650995
## 45   0.860613618
## 46   0.991996635
## 47   0.996827510
## 48   0.996520229
## 49   0.188947793
## 50   0.005643259
## 51   0.997913926
## 52   0.996315881
## 53   0.999109948
## 54   0.996917981
## 55   0.683378496
## 56   0.983125960
## 57   0.949528525
## 58   0.903008260
## 59   0.818289008
## 60   0.991418754
## 61   0.996560800
## 62   0.996825742
## 63   0.996097107
## 64   0.180823878
## 65   0.996292499
## 66   0.960814566
## 67   0.826810736
## 68   0.997914475
```

```
## 69   0.836405165
## 70   0.246984017
## 71   0.998595354
## 72   0.994359434
## 73   0.140463785
## 74   0.996873571
## 75   0.995729277
## 76   0.277502308
## 77   0.666613841
## 78   0.996919262
## 79   0.984417221
## 80   0.996703971
## 81   0.996548506
## 82   0.996842746
## 83   0.924017152
## 84   0.919395601
## 85   0.009221751
## 86   0.004624147
## 87   0.984000964
## 88   0.796171733
## 89   0.996990095
## 90   0.994322667
## 91   0.981841482
## 92   0.994859971
## 93   0.994350591
## 94   0.868412763
## 95   0.101258115
## 96   0.991297035
## 97   0.995995417
## 98   0.030011047
## 99   0.694114412
## 100  0.961436295
## 101  0.992582530
## 102  0.996809485
## 103  0.053710028
## 104  0.996866631
## 105  0.756665067
## 106  0.971332723
## 107  0.990303755
## 108  0.995772917
## 109  0.994728824
## 110  0.206667440
## 111  0.732019432
## 112  0.996311517
## 113  0.966445073
## 114  0.060130638
## 115  0.022251428
## 116  0.996863051
## 117  0.999430823
## 118  0.993190133
## 119  0.780036710
## 120  0.996223885
## 121  0.053083424
## 122  0.964119036
```

```
## 123  0.003640058
## 124  0.995701872
## 125  0.004574390
## 126  0.999849616
## 127  0.968849248
## 128  0.015843728
## 129  0.609702036
## 130  0.802125154
## 131  0.998729186
## 132  0.721707370
## 133  0.138035692
## 134  0.092584922
## 135  0.235766593
## 136  0.677644697
## 137  0.825480452
## 138  0.846375172
## 139  0.674422464
## 140  0.272887295
## 141  0.989967466
## 142  0.996804616
## 143  0.964053416
## 144  0.746726523
## 145  0.806654001
## 146  0.996914334
## 147  0.996266368
## 148  0.970209730
## 149  0.448242026
## 150  0.991062541
## 151  0.003706137
## 152  0.996728935
## 153  0.003818942
## 154  0.720809642
## 155  0.023292255
## 156  0.916551915
## 157  0.841730603
## 158  0.864701048
## 159  0.989789806
## 160  0.401306195
## 161  0.006824608
## 162  0.214407222
## 163  0.995907161
## 164  0.990418072
## 165  0.991050211
## 166  0.996509274
## 167  0.987586055
## 168  0.189063934
## 169  0.188968088
## 170  0.904808934
## 171  0.997537782
## 172  0.989802131
## 173  0.987791358
## 174  0.986786702
## 175  0.188947793
## 176  0.996012535
```

```
## 177  0.970177144
## 178  0.970348741
## 179  0.995517138
## 180  0.406407088
## 181  0.921020578
## 182  0.678698638
## 183  0.997497532
## 184  0.457016132
## 185  0.983335405
## 186  0.958755927
## 187  0.996591080
## 188  0.577783317
## 189  0.987597740
## 190  0.684688051
## 191  0.995803955
## 192  0.004134771
## 193  0.355709085
## 194  0.223785813
## 195  0.018302940
## 196  0.998114703
## 197  0.996918279
## 198  0.097332264
## 199  0.710662376
## 200  0.195817524
## 201  0.189057158
## 202  0.266610079
## 203  0.998233055
## 204  0.200004748
## 205  0.128241727
## 206  0.003777575
## 207  0.934359832
## 208  0.996911046
## 209  0.996936668
## 210  0.999885601
## 211  0.973485569
## 212  0.146799778
## 213  0.003749682
## 214  0.991968379
## 215  0.997709395
## 216  0.221861513
## 217  0.030022070
## 218  0.429732354
## 219  0.186374512
## 220  0.966054445
## 221  0.003638753
## 222  0.880240559
## 223  0.214157889
## 224  0.125999539
## 225  0.994990835
## 226  0.263088274
## 227  0.767879178
## 228  0.176565850
## 229  0.990740351
## 230  0.996833627
```

```
## 231  0.090837878
## 232  0.950127059
## 233  0.188947793
## 234  0.996913189
## 235  0.996873968
## 236  0.996854494
## 237  0.996863835
## 238  0.997059419
## 239  0.999941645
## 240  0.996858825
## 241  0.999943801
## 242  0.189007293
## 243  0.953532259
## 244  0.188949232
## 245  0.996902626
## 246  0.968723620
## 247  0.528395061
## 248  0.999933839
## 249  0.996010861
## 250  0.996839777
## 251  0.682356284
## 252  0.974608406
## 253  0.997307253
## 254  0.094074985
## 255  0.996218518
## 256  0.944570420
## 257  0.976318596
## 258  0.013662841
## 259  0.194336648
## 260  0.465805796
## 261  0.996699549
## 262  0.007370663
## 263  0.994014061
## 264  0.817591345
## 265  0.988245251
## 266  0.204542495
## 267  0.423335020
## 268  0.245273507
## 269  0.136857786
## 270  0.884971164
## 271  0.996347477
## 272  0.996520096
## 273  0.007915288
## 274  0.827121384
## 275  0.712213214
## 276  0.996889688
## 277  0.018455867
## 278  0.176650337
## 279  0.940659690
## 280  0.995924995
## 281  0.522710536
## 282  0.075072121
## 283  0.989163341
## 284  0.265814456
```

```
## 285  0.915132573
## 286  0.999949620
## 287  0.996682555
## 288  0.999798991
## 289  0.190056054
## 290  0.164316145
## 291  0.511080369
## 292  0.874774077
## 293  0.706730124
## 294  0.186915537
## 295  0.004135375
## 296  0.971437310
## 297  0.996921046
## 298  0.062661803
## 299  0.934545736
## 300  0.218915038
## 301  0.197313812
## 302  0.995940540
## 303  0.127964748
## 304  0.005130426
## 305  0.005200346
## 306  0.189659442
## 307  0.043663247
## 308  0.982605666
## 309  0.008339705
## 310  0.996913308
## 311  0.808521806
## 312  0.169603266
## 313  0.786962046
## 314  0.986437871
## 315  0.682485201
## 316  0.684843678
## 317  0.985793397
## 318  0.670313922
## 319  0.273626151
## 320  0.691398104
## 321  0.991528705
## 322  0.939575989
## 323  0.738280212
## 324  0.716038638
## 325  0.996669747
## 326  0.991585730
## 327  0.233741136
## 328  0.945788182
## 329  0.003840736
## 330  0.614963550
## 331  0.007210647
## 332  0.275352848
## 333  0.681719542
## 334  0.633263667
## 335  0.194222504
## 336  0.736389867
## 337  0.714097670
## 338  0.672638470
```

```
## 339  0.864008689
## 340  0.690543478
## 341  0.691900992
## 342  0.995775743
## 343  0.733233963
## 344  0.458324987
## 345  0.955876464
## 346  0.490215900
## 347  0.690521357
## 348  0.059561562
## 349  0.686770948
## 350  0.689695126
## 351  0.687975248
## 352  0.696143195
## 353  0.686517843
## 354  0.003730962
## 355  0.522737070
## 356  0.797760970
## 357  0.655616627
## 358  0.687725368
## 359  0.700247567
## 360  0.884024188
## 361  0.992812315
## 362  0.995987755
## 363  0.991493499
## 364  0.856775193
## 365  0.961973238
## 366  0.237412371
## 367  0.909550707
## 368  0.003978485
## 369  0.605809226
## 370  0.686203124
## 371  0.996908394
## 372  0.678316742
## 373  0.764062864
## 374  0.678850582
## 375  0.996515921
## 376  0.672342787
## 377  0.033289606
## 378  0.686319218
## 379  0.993911674
## 380  0.110262675
## 381  0.646067055
## 382  0.005559333
## 383  0.926181544
## 384  0.995509532
## 385  0.083098327
## 386  0.591063675
## 387  0.992317984
## 388  0.996907054
## 389  0.254470740
## 390  0.004146795
## 391  0.004982857
## 392  0.691394672
```

```
## 393  0.999661745
## 394  0.703364256
## 395  0.240309579
## 396  0.981601182
## 397  0.694001652
## 398  0.177577332
## 399  0.698808549
## 400  0.710322064
## 401  0.709566790
## 402  0.027911058
## 403  0.686074281
## 404  0.981788084
## 405  0.996878044
## 406  0.692398195
## 407  0.282388812
## 408  0.962167544
## 409  0.979796042
## 410  0.823704794
## 411  0.023639739
## 412  0.004880812
## 413  0.983179265
## 414  0.996831134
## 415  0.020201534
## 416  0.996388878
## 417  0.417426822
## 418  0.750262884
## 419  0.826503469
## 420  0.967397835
## 421  0.996225154
## 422  0.188947793
## 423  0.894786769
## 424  0.992424287
## 425  0.995146555
## 426  0.986873418
## 427  0.992065181
## 428  0.707382186
## 429  0.497579115
## 430  0.731055061
## 431  0.690767237
## 432  0.874090163
## 433  0.701510508
## 434  0.914510306
## 435  0.934482955
## 436  0.884472631
## 437  0.137337971
## 438  0.094420354
## 439  0.996758051
## 440  0.697860117
## 441  0.918554209
## 442  0.879694484
## 443  0.965245534
## 444  0.231394089
## 445  0.007257731
## 446  0.018385991
```

```
## 447  0.996917657
## 448  0.690074321
## 449  0.993840649
## 450  0.696609007
## 451  0.053744610
## 452  0.037308894
## 453  0.815726231
## 454  0.990507570
## 455  0.674048337
## 456  0.686814511
## 457  0.660286890
## 458  0.908753154
## 459  0.993646147
## 460  0.291876867
## 461  0.996827087
## 462  0.996850528
## 463  0.706696931
## 464  0.479283702
## 465  0.008180470
## 466  0.715893719
## 467  0.986547621
## 468  0.996835160
## 469  0.187818117
## 470  0.984113927
## 471  0.690156884
## 472  0.996667538
## 473  0.051801014
## 474  0.785515333
## 475  0.996808544
## 476  0.996911227
## 477  0.996921784
## 478  0.994527555
## 479  0.922310652
## 480  0.976609413
## 481  0.003636988
## 482  0.734030871
## 483  0.996739238
## 484  0.999889288
## 485  0.995793988
## 486  0.735721245
## 487  0.996518560
## 488  0.996893148
## 489  0.004473082
## 490  0.559262431
## 491  0.024883319
## 492  0.959399131
## 493  0.776865956
## 494  0.981562905
## 495  0.996919233
## 496  0.024441771
## 497  0.938132325
## 498  0.994682242
## 499  0.993444970
## 500  0.004023117
```

```
## 501  0.694733256
## 502  0.188947793
## 503  0.004017443
## 504  0.259384310
## 505  0.690636402
## 506  0.526146360
## 507  0.016863099
## 508  0.986710838
## 509  0.980986776
## 510  0.760254821
## 511  0.297102764
## 512  0.031905599
## 513  0.192933616
## 514  0.951445704
## 515  0.685512637
## 516  0.955866441
## 517  0.384671783
## 518  0.972588854
## 519  0.992595978
## 520  0.690801214
## 521  0.682283639
## 522  0.686358893
## 523  0.013926229
## 524  0.918578174
## 525  0.821697789
## 526  0.969499628
## 527  0.008671758
## 528  0.408222772
## 529  0.680199377
## 530  0.692626863
## 531  0.705691528
## 532  0.003693964
## 533  0.950791941
## 534  0.034823253
## 535  0.188948336
## 536  0.008388037
## 537  0.045539256
## 538  0.060484171
## 539  0.645139271
## 540  0.994448487
## 541  0.690199684
## 542  0.965808603
## 543  0.062178697
## 544  0.019155451
## 545  0.749909529
## 546  0.954422399
## 547  0.467402067
## 548  0.978821897
## 549  0.703560300
## 550  0.003848903
## 551  0.998822193
## 552  0.348204652
## 553  0.686840552
## 554  0.005002211
```

```
## 555  0.687704221
## 556  0.672225163
## 557  0.087596290
## 558  0.352082955
## 559  0.685808964
## 560  0.996858139
## 561  0.058510484
## 562  0.003808024
## 563  0.188947793
## 564  0.433603550
## 565  0.996841017
## 566  0.996842349
## 567  0.996898316
## 568  0.073624594
## 569  0.579093429
## 570  0.688867907
## 571  0.028968160
## 572  0.871087792
## 573  0.683836614
## 574  0.004976917
## 575  0.977013317
## 576  0.719301646
## 577  0.004390067
## 578  0.028281777
## 579  0.996882237
## 580  0.827385009
## 581  0.005063812
## 582  0.018887220
## 583  0.698869426
## 584  0.996317572
## 585  0.007206087
## 586  0.026709746
## 587  0.996556084
## 588  0.360359744
## 589  0.039355959
## 590  0.188966578
## 591  0.693688535
## 592  0.005320863
## 593  0.004071993
## 594  0.003678249
## 595  0.004051796
## 596  0.732859193
## 597  0.688395918
## 598  0.919120091
## 599  0.689408928
## 600  0.958842775
## 601  0.329500336
## 602  0.717972200
## 603  0.056904920
## 604  0.995398672
## 605  0.990718789
## 606  0.994107578
## 607  0.786273725
## 608  0.974492788
```

```
## 609  0.996622551
## 610  0.691319560
## 611  0.563445095
## 612  0.509960119
## 613  0.003647716
## 614  0.985207081
## 615  0.180847439
## 616  0.784285260
## 617  0.989540475
## 618  0.692177606
## 619  0.675089954
## 620  0.996778415
## 621  0.152775859
## 622  0.195600672
## 623  0.698786064
## 624  0.926460013
## 625  0.969055189
## 626  0.843009933
## 627  0.994210262
## 628  0.380120763
## 629  0.701915773
## 630  0.594963717
## 631  0.843232880
## 632  0.005768139
## 633  0.010406735
## 634  0.674331377
## 635  0.709984352
## 636  0.996813118
## 637  0.685409962
## 638  0.996516839
## 639  0.682147505
## 640  0.488066385
## 641  0.925505822
## 642  0.556744173
## 643  0.876970937
## 644  0.971553552
## 645  0.630225830
## 646  0.654927129
## 647  0.047272862
## 648  0.100733233
## 649  0.188947793
## 650  0.995786719
## 651  0.287884951
## 652  0.004887801
## 653  0.471933861
## 654  0.686023555
## 655  0.982739398
## 656  0.698073243
## 657  0.996901389
## 658  0.993914194
## 659  0.998854066
## 660  0.989321258
## 661  0.633846271
## 662  0.994042370
```

```
## 663  0.937310900
## 664  0.996921151
## 665  0.703648602
## 666  0.996921806
## 667  0.991447615
## 668  0.991655700
## 669  0.999874832
## 670  0.996857602
## 671  0.996753961
## 672  0.996732647
## 673  0.960731716
## 674  0.717679913
## 675  0.197832051
## 676  0.691225057
## 677  0.717173711
## 678  0.978163454
## 679  0.979819281
## 680  0.998404056
## 681  0.996788513
## 682  0.701524686
## 683  0.202936616
## 684  0.003756049
## 685  0.980039673
## 686  0.994528063
## 687  0.913807503
## 688  0.950574674
## 689  0.832135658
## 690  0.996919597
## 691  0.996408621
## 692  0.698660261
## 693  0.950081428
## 694  0.996745968
## 695  0.011165691
## 696  0.995145767
## 697  0.996573669
## 698  0.999918515
## 699  0.996184899
## 700  0.990977251
## 701  0.689797289
## 702  0.992051604
## 703  0.004469076
## 704  0.035274896
## 705  0.996930685
## 706  0.390916197
## 707  0.750875969
## 708  0.996978778
## 709  0.996190732
## 710  0.996780176
## 711  0.687275481
## 712  0.689812699
## 713  0.998847096
## 714  0.749475106
## 715  0.996921144
## 716  0.382944797
```

```
## 717  0.722269914
## 718  0.009050157
## 719  0.996841462
## 720  0.961915148
## 721  0.965037844
## 722  0.997556210
## 723  0.990614039
## 724  0.991450764
## 725  0.003887572
## 726  0.995343714
## 727  0.996920896
## 728  0.007898014
## 729  0.197240866
## 730  0.891055230
## 731  0.745194495
## 732  0.971329194
## 733  0.996707609
## 734  0.992187179
## 735  0.973725586
## 736  0.994192351
## 737  0.711503331
## 738  0.974360060
## 739  0.949534605
## 740  0.996835131
## 741  0.004049907
## 742  0.706878429
## 743  0.834855659
## 744  0.816597782
## 745  0.996173924
## 746  0.797405458
## 747  0.752380517
## 748  0.938631973
## 749  0.996593072
## 750  0.994232530
## 751  0.716226924
## 752  0.003668741
## 753  0.005314737
## 754  0.986880549
## 755  0.995501667
## 756  0.968541364
## 757  0.985074850
## 758  0.096278843
## 759  0.688759496
## 760  0.004329982
## 761  0.874846050
## 762  0.995749796
## 763  0.901399301
## 764  0.793174311
## 765  0.689994593
## 766  0.780187772
## 767  0.996833011
## 768  0.003650233
## 769  0.994355046
## 770  0.689869440
```

```
## 771  0.084237093
## 772  0.986865608
## 773  0.996860326
## 774  0.971728936
## 775  0.995315126
## 776  0.996910013
## 777  0.803266913
## 778  0.722405595
## 779  0.979863861
## 780  0.996914015
## 781  0.047021337
## 782  0.344829405
## 783  0.996892206
## 784  0.069988823
## 785  0.929541464
## 786  0.689538976
## 787  0.996854010
## 788  0.996146136
## 789  0.683928827
## 790  0.856193113
## 791  0.689713308
## 792  0.678619352
## 793  0.188059020
## 794  0.996920552
## 795  0.996891360
## 796  0.007046498
## 797  0.973359607
## 798  0.003848119
## 799  0.996919781
## 800  0.996911818
## 801  0.139468990
## 802  0.013323988
## 803  0.995905937
## 804  0.996920919
## 805  0.996826878
## 806  0.925387566
## 807  0.996579372
## 808  0.509567709
## 809  0.996915317
## 810  0.996891701
## 811  0.703092654
## 812  0.119898303
## 813  0.938914555
## 814  0.996918891
## 815  0.992567026
## 816  0.996254239
## 817  0.003639316
## 818  0.996860239
## 819  0.996919780
## 820  0.996913582
## 821  0.683983490
## 822  0.690160573
## 823  0.690230610
## 824  0.996792548
```

```
## 825  0.995212042
## 826  0.996921209
## 827  0.695342459
## 828  0.996895847
## 829  0.996703697
## 830  0.666102267
## 831  0.003691182
## 832  0.674187536
## 833  0.835557693
## 834  0.996832557
## 835  0.689387640
## 836  0.996829495
## 837  0.178664605
## 838  0.039075999
## 839  0.613388488
## 840  0.995597393
## 841  0.996900309
## 842  0.766713784
## 843  0.738251448
## 844  0.999714117
## 845  0.752757909
## 846  0.996912915
## 847  0.692703189
## 848  0.996922210
## 849  0.004853158
## 850  0.940241151
## 851  0.761901786
## 852  0.991090615
## 853  0.984428339
## 854  0.762631814
## 855  0.691290853
## 856  0.993486009
## 857  0.996908556
## 858  0.154524116
## 859  0.702202216
## 860  0.996915541
## 861  0.833854800
## 862  0.697316825
## 863  0.994634512
## 864  0.729918956
## 865  0.966520910
## 866  0.996369696
## 867  0.086915137
## 868  0.008303978
## 869  0.674111108
## 870  0.996915214
## 871  0.965931075
## 872  0.684780783
## 873  0.849351725
## 874  0.712531578
## 875  0.953440453
## 876  0.652160579
## 877  0.005065887
## 878  0.108892007
```

```
## 879  0.688056033
## 880  0.964275379
## 881  0.709640329
## 882  0.996678570
## 883  0.846777729
## 884  0.639662335
## 885  0.872178164
## 886  0.996900715
## 887  0.989777891
## 888  0.938971289
## 889  0.085926831
## 890  0.996869365
## 891  0.006909662
## 892  0.994271869
## 893  0.692328495
## 894  0.690440283
## 895  0.989848117
## 896  0.906079187
## 897  0.800529601
## 898  0.691225964
## 899  0.270851130
## 900  0.969761814
## 901  0.834674487
## 902  0.970312762
## 903  0.687200884
## 904  0.003942366
## 905  0.996125830
## 906  0.785631694
## 907  0.684843411
## 908  0.999824941
## 909  0.996728212
## 910  0.996151259
## 911  0.996718836
## 912  0.207182093
## 913  0.003812397
## 914  0.694279748
## 915  0.990694156
## 916  0.993222407
## 917  0.090235558
## 918  0.238593947
## 919  0.708426684
## 920  0.668310556
## 921  0.996312570
## 922  0.690037682
## 923  0.290956424
## 924  0.689704612
## 925  0.986508414
## 926  0.996791075
## 927  0.007055978
## 928  0.688092878
## 929  0.760811761
## 930  0.996895381
## 931  0.687410869
## 932  0.012515390
```

```
## 933  0.588345754
## 934  0.995217186
## 935  0.996885028
## 936  0.996917783
## 937  0.978469179
## 938  0.188947793
## 939  0.688568228
## 940  0.690692601
## 941  0.562751814
## 942  0.684243704
## 943  0.995138636
## 944  0.987643963
## 945  0.987505620
## 946  0.938444391
## 947  0.688198339
## 948  0.279098416
## 949  0.532286826
## 950  0.794181477
## 951  0.740410811
## 952  0.978455533
## 953  0.208454442
## 954  0.999255611
## 955  0.902278528
## 956  0.690523287
## 957  0.978161549
## 958  0.306379441
## 959  0.996912311
## 960  0.005108105
## 961  0.959683384
## 962  0.688450802
## 963  0.989559669
## 964  0.878760731
## 965  0.049539633
## 966  0.161153306
## 967  0.188947793
## 968  0.004070977
## 969  0.088750545
## 970  0.190787258
## 971  0.188953203
## 972  0.857919456
## 973  0.448459220
## 974  0.995865516
## 975  0.996913677
## 976  0.696568801
## 977  0.004389658
## 978  0.188956178
## 979  0.362066102
## 980  0.448584142
## 981  0.661232533
## 982  0.115007640
## 983  0.070666820
## 984  0.212637984
## 985  0.188595921
## 986  0.971268307
```

```
## 987  0.907957987
## 988  0.122886784
## 989  0.980117604
## 990  0.987081475
## 991  0.822956019
## 992  0.722685744
## 993  0.891774214
## 994  0.995527576
## 995  0.189966158
## 996  0.682669390
## 997  0.685990680
## 998  0.854847945
## 999  0.996550566
## 1000 0.026155730
## 1001 0.188947793
## 1002 0.996552818
## 1003 0.198489567
## 1004 0.003899972
## 1005 0.006816278
## 1006 0.758234980
## 1007 0.996903650
## 1008 0.991015525
## 1009 0.996863790
## 1010 0.996866790
## 1011 0.275323382
## 1012 0.003719086
## 1013 0.687889963
## 1014 0.690281706
## 1015 0.992640864
## 1016 0.992426677
## 1017 0.999724297
## 1018 0.989771450
## 1019 0.999815712
## 1020 0.632008336
## 1021 0.004791240
## 1022 0.188947793
## 1023 0.996916663
## 1024 0.996840422
## 1025 0.003670504
## 1026 0.468862274
## 1027 0.694812664
## 1028 0.994744460
## 1029 0.182969283
## 1030 0.982565030
## 1031 0.996594631
## 1032 0.065121308
## 1033 0.399175757
## 1034 0.188948900
## 1035 0.005212293
## 1036 0.995573720
## 1037 0.996890462
## 1038 0.004002266
## 1039 0.700329669
## 1040 0.994882801
```

```
## 1041 0.683666460
## 1042 0.689743991
## 1043 0.368690169
## 1044 0.188979164
## 1045 0.188947793
## 1046 0.244824046
## 1047 0.683437477
## 1048 0.976418715
## 1049 0.915371144
## 1050 0.887862544
## 1051 0.918797831
## 1052 0.970603603
## 1053 0.994358954
## 1054 0.996916497
## 1055 0.753765171
## 1056 0.688014393
## 1057 0.877684776
## 1058 0.996537893
## 1059 0.028090981
## 1060 0.722906784
## 1061 0.007496836
## 1062 0.729915570
## 1063 0.005942561
## 1064 0.687161203
## 1065 0.688064311
## 1066 0.011056746
## 1067 0.688796304
## 1068 0.007795125
## 1069 0.164730738
## 1070 0.445112913
## 1071 0.510024967
## 1072 0.124043077
## 1073 0.003777948
## 1074 0.135197050
## 1075 0.019495979
## 1076 0.006623979
## 1077 0.048027952
## 1078 0.670921873
## 1079 0.169951515
## 1080 0.996484910
## 1081 0.984424232
## 1082 0.831903228
## 1083 0.188977523
## 1084 0.923544107
## 1085 0.003854704
## 1086 0.004260354
## 1087 0.224986780
## 1088 0.568775280
## 1089 0.689906279
## 1090 0.984574258
## 1091 0.004703101
## 1092 0.046083360
## 1093 0.047354973
## 1094 0.386541730
```

```
## 1095 0.670695220
## 1096 0.628738130
## 1097 0.686362818
## 1098 0.112702232
## 1099 0.050313257
## 1100 0.681120756
## 1101 0.683359854
## 1102 0.689575938
## 1103 0.006249442
## 1104 0.003812188
## 1105 0.995924447
## 1106 0.898682378
## 1107 0.688065510
## 1108 0.841114731
## 1109 0.363578145
## 1110 0.662285704
## 1111 0.757388359
## 1112 0.996198836
## 1113 0.569739606
## 1114 0.144873493
## 1115 0.733522710
## 1116 0.710797308
## 1117 0.996871011
## 1118 0.088088340
## 1119 0.996891042
## 1120 0.687642014
## 1121 0.026770719
## 1122 0.605391659
## 1123 0.690560841
## 1124 0.070194218
## 1125 0.031493022
## 1126 0.685533655
## 1127 0.003662992
## 1128 0.048332946
## 1129 0.867218051
## 1130 0.681210415
## 1131 0.997004999
## 1132 0.257419661
## 1133 0.458364890
## 1134 0.970110948
## 1135 0.968881689
## 1136 0.660199361
## 1137 0.989245237
## 1138 0.672456642
## 1139 0.451870076
## 1140 0.640969995
## 1141 0.693720269
## 1142 0.004361955
## 1143 0.881782734
## 1144 0.690599622
## 1145 0.902441790
## 1146 0.033486652
## 1147 0.985533936
## 1148 0.996903197
```

```
## 1149 0.003842212
## 1150 0.195695144
## 1151 0.989356604
## 1152 0.306544536
## 1153 0.833917845
## 1154 0.005797371
## 1155 0.003653767
## 1156 0.003659951
## 1157 0.982922820
## 1158 0.022567523
## 1159 0.004550257
## 1160 0.677878476
## 1161 0.690218106
## 1162 0.696653624
## 1163 0.992977009
## 1164 0.429282945
## 1165 0.996639882
## 1166 0.004112468
## 1167 0.996914956
## 1168 0.003935908
## 1169 0.005911138
## 1170 0.996160816
## 1171 0.996819513
## 1172 0.685197116
## 1173 0.987288174
## 1174 0.691945582
## 1175 0.939682210
## 1176 0.188958395
## 1177 0.160816146
## 1178 0.029246204
## 1179 0.696527387
## 1180 0.892175892
## 1181 0.880527809
## 1182 0.690461711
## 1183 0.698298481
## 1184 0.246576151
## 1185 0.972207526
## 1186 0.996918932
## 1187 0.948845060
## 1188 0.004392605
## 1189 0.698969067
## 1190 0.007014616
## 1191 0.701379376
## 1192 0.931316673
## 1193 0.555215056
## 1194 0.952308989
## 1195 0.999335590
## 1196 0.004050989
## 1197 0.707958199
## 1198 0.708976169
## 1199 0.006750325
## 1200 0.706135464
## 1201 0.006404636
## 1202 0.802732098
```

```
## 1203 0.533956084
## 1204 0.212845040
## 1205 0.030626761
## 1206 0.357278375
## 1207 0.688799272
## 1208 0.681808328
## 1209 0.702296483
## 1210 0.984088600
## 1211 0.834435797
## 1212 0.367964873
## 1213 0.649088104
## 1214 0.004550528
## 1215 0.945312414
## 1216 0.727869137
## 1217 0.003676188
## 1218 0.970693333
## 1219 0.004399592
## 1220 0.982580098
## 1221 0.004228770
## 1222 0.434667533
## 1223 0.429214463
## 1224 0.980953447
## 1225 0.021737359
## 1226 0.388537181
## 1227 0.004545555
## 1228 0.150894058
## 1229 0.008027110
## 1230 0.131070020
## 1231 0.703228520
## 1232 0.188947793
## 1233 0.038378467
## 1234 0.189132361
## 1235 0.694635894
## 1236 0.188947793
## 1237 0.692349368
## 1238 0.182917211
## 1239 0.189244393
## 1240 0.868233314
## 1241 0.043270909
## 1242 0.740717094
## 1243 0.858456239
## 1244 0.201106529
## 1245 0.989317717
## 1246 0.189346072
## 1247 0.003767935
## 1248 0.345358392
## 1249 0.689846090
## 1250 0.996719691
## 1251 0.060182789
## 1252 0.698291930
## 1253 0.992529425
## 1254 0.837119050
## 1255 0.006845376
## 1256 0.996431864
```

```
## 1257 0.006005262
## 1258 0.193676458
## 1259 0.707031096
## 1260 0.996204479
## 1261 0.996853886
## 1262 0.738381142
## 1263 0.003773327
## 1264 0.833219376
## 1265 0.762905054
## 1266 0.191283966
## 1267 0.963401980
## 1268 0.203108337
## 1269 0.374356057
## 1270 0.355121290
## 1271 0.199929296
## 1272 0.173505402
## 1273 0.682659077
## 1274 0.664476815
## 1275 0.003672071
## 1276 0.672383909
## 1277 0.013159419
## 1278 0.376462730
## 1279 0.674297538
## 1280 0.989843239
## 1281 0.008643407
## 1282 0.999940275
## 1283 0.712144079
## 1284 0.993492138
## 1285 0.996505124
## 1286 0.779486553
## 1287 0.689960463
## 1288 0.304143541
## 1289 0.098374338
## 1290 0.188196386
## 1291 0.770282900
## 1292 0.821221176
## 1293 0.664504891
## 1294 0.029501144
## 1295 0.646650440
## 1296 0.006772816
## 1297 0.806855829
## 1298 0.014882075
## 1299 0.188947793
## 1300 0.708117973
## 1301 0.855169373
## 1302 0.482423915
## 1303 0.686216694
## 1304 0.006848412
## 1305 0.304781177
## 1306 0.672769086
## 1307 0.687965969
## 1308 0.011796595
## 1309 0.679571483
## 1310 0.977870959
```

```
## 1311 0.572435673
## 1312 0.689492616
## 1313 0.011961583
## 1314 0.688672827
## 1315 0.689059807
## 1316 0.005269126
## 1317 0.617587820
## 1318 0.989852860
## 1319 0.910076774
## 1320 0.003658211
## 1321 0.992842739
## 1322 0.625971819
## 1323 0.003899744
## 1324 0.617783182
## 1325 0.786239732
## 1326 0.659574958
## 1327 0.693456822
## 1328 0.376032693
## 1329 0.432575615
## 1330 0.987471639
## 1331 0.918815568
## 1332 0.275302835
## 1333 0.354854623
## 1334 0.154996845
## 1335 0.188971198
## 1336 0.604284895
## 1337 0.006512699
## 1338 0.619269631
## 1339 0.010691453
## 1340 0.686243861
## 1341 0.690100270
## 1342 0.600680228
## 1343 0.009284564
## 1344 0.692026089
## 1345 0.688377849
## 1346 0.691140063
## 1347 0.668391214
## 1348 0.463263081
## 1349 0.813753521
## 1350 0.994210834
## 1351 0.959465870
## 1352 0.003866190
## 1353 0.983281478
## 1354 0.690174261
## 1355 0.003801523
## 1356 0.985287577
## 1357 0.140777657
## 1358 0.004247708
## 1359 0.695983535
## 1360 0.996733863
## 1361 0.689821340
## 1362 0.687974181
## 1363 0.799776913
## 1364 0.687399281
```

```
## 1365 0.689680954
## 1366 0.694017356
## 1367 0.029287544
## 1368 0.046505575
## 1369 0.994317857
## 1370 0.057992918
## 1371 0.721385843
## 1372 0.188947793
## 1373 0.026486402
## 1374 0.688490941
## 1375 0.003803849
## 1376 0.996912573
## 1377 0.664812315
## 1378 0.004115528
## 1379 0.688993193
## 1380 0.689526250
## 1381 0.822218828
## 1382 0.186189934
## 1383 0.432504588
## 1384 0.943747960
## 1385 0.753544902
## 1386 0.352112375
## 1387 0.003948816
## 1388 0.578560956
## 1389 0.008041622
## 1390 0.690298019
## 1391 0.996768075
## 1392 0.993920805
## 1393 0.186694450
## 1394 0.996119684
## 1395 0.690644908
## 1396 0.657008332
## 1397 0.282957238
## 1398 0.707564634
## 1399 0.933477156
## 1400 0.023027224
## 1401 0.992029238
## 1402 0.455507067
## 1403 0.680018256
## 1404 0.079820823
## 1405 0.004606872
## 1406 0.689157436
## 1407 0.688997909
## 1408 0.689459954
## 1409 0.444102715
## 1410 0.786284406
## 1411 0.985070448
## 1412 0.188947793
## 1413 0.188947793
## 1414 0.955878904
## 1415 0.192567744
## 1416 0.021289295
## 1417 0.051545942
## 1418 0.003679604
```

```
## 1419 0.826304255
## 1420 0.029386494
## 1421 0.646274587
## 1422 0.354011380
## 1423 0.974664048
## 1424 0.197968746
## 1425 0.004246016
## 1426 0.956910546
## 1427 0.062943665
## 1428 0.688725300
## 1429 0.986976778
## 1430 0.695222314
## 1431 0.770741770
## 1432 0.990552830
## 1433 0.764104251
## 1434 0.873249267
## 1435 0.687139997
## 1436 0.691872331
## 1437 0.688984435
## 1438 0.705291419
## 1439 0.579027245
## 1440 0.691664512
## 1441 0.834720452
## 1442 0.738997979
## 1443 0.872391748
## 1444 0.727089978
## 1445 0.354465369
## 1446 0.003948037
## 1447 0.015001649
## 1448 0.004179966
## 1449 0.996837417
## 1450 0.667676475
## 1451 0.015910883
## 1452 0.809135322
## 1453 0.003755254
## 1454 0.696712260
## 1455 0.141986204
## 1456 0.649667263
## 1457 0.996882496
## 1458 0.689261732
## 1459 0.990907823
## 1460 0.400383087
## 1461 0.698800559
## 1462 0.814507806
## 1463 0.603549992
## 1464 0.012268975
## 1465 0.112097634
## 1466 0.674546720
## 1467 0.003764217
## 1468 0.009616352
## 1469 0.992844934
## 1470 0.006987086
## 1471 0.991646407
## 1472 0.235385570
```

```
## 1473 0.189000754
## 1474 0.993312183
## 1475 0.996870137
## 1476 0.610547788
## 1477 0.197382883
## 1478 0.198694018
## 1479 0.996701357
## 1480 0.771557524
## 1481 0.734852515
## 1482 0.607836183
## 1483 0.654455993
## 1484 0.993238480
## 1485 0.996865740
## 1486 0.996878673
## 1487 0.995178496
## 1488 0.910524301
## 1489 0.007166110
## 1490 0.383575606
## 1491 0.682603791
## 1492 0.573056714
## 1493 0.188954426
## 1494 0.692280611
## 1495 0.811666827
## 1496 0.980554384
## 1497 0.688738988
## 1498 0.188947793
## 1499 0.208060409
## 1500 0.650945370
## 1501 0.766096378
## 1502 0.690058307
## 1503 0.996372660
## 1504 0.666388674
## 1505 0.512764802
## 1506 0.188947793
## 1507 0.006497019
## 1508 0.994723927
## 1509 0.081854253
## 1510 0.003965325
## 1511 0.003680442
## 1512 0.052692960
## 1513 0.004205268
## 1514 0.995356245
## 1515 0.741115181
## 1516 0.004058638
## 1517 0.715056987
## 1518 0.511640116
## 1519 0.003645947
## 1520 0.010313253
## 1521 0.076909101
## 1522 0.017571697
## 1523 0.034121938
## 1524 0.631279832
## 1525 0.988369006
## 1526 0.988770715
```

```
## 1527 0.990734525
## 1528 0.188952861
## 1529 0.694811403
## 1530 0.966611681
## 1531 0.996855222
## 1532 0.015085643
## 1533 0.997185498
## 1534 0.529757315
## 1535 0.998979856
## 1536 0.689818140
## 1537 0.693986648
## 1538 0.003904016
## 1539 0.980154635
## 1540 0.003643624
## 1541 0.044046093
## 1542 0.003786751
## 1543 0.772660015
## 1544 0.004087469
## 1545 0.905290108
## 1546 0.048044472
## 1547 0.680356526
## 1548 0.442449631
## 1549 0.204390450
## 1550 0.665418466
## 1551 0.328385491
## 1552 0.790929640
## 1553 0.687907323
## 1554 0.011523466
## 1555 0.786495214
## 1556 0.006598443
## 1557 0.007031908
## 1558 0.010611320
## 1559 0.993505359
## 1560 0.941254164
## 1561 0.695571715
## 1562 0.688930044
## 1563 0.025904151
## 1564 0.916226553
## 1565 0.976659775
## 1566 0.267734345
## 1567 0.734041013
## 1568 0.007637716
## 1569 0.196627743
## 1570 0.834428535
## 1571 0.572646418
## 1572 0.959528749
## 1573 0.188947793
## 1574 0.004621674
## 1575 0.024368342
## 1576 0.607002751
## 1577 0.254413713
## 1578 0.702230592
## 1579 0.689859813
## 1580 0.690528625
```

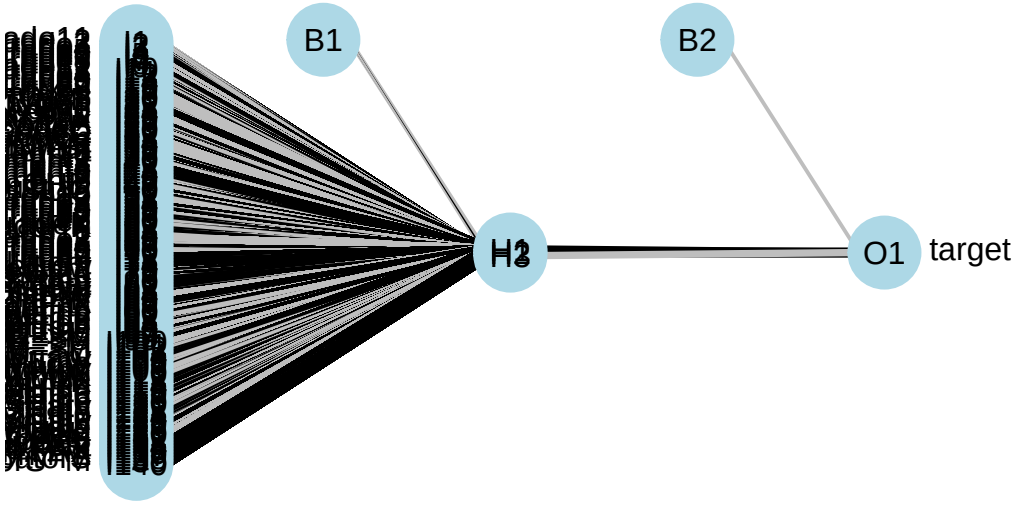
```
## 1581 0.317807402
## 1582 0.685712616
## 1583 0.704852435
## 1584 0.996748582
## 1585 0.640918052
## 1586 0.625755440
## 1587 0.383768214
## 1588 0.700474166
## 1589 0.684353826
## 1590 0.003703210
## 1591 0.007050400
## 1592 0.687182977
## 1593 0.545724594
## 1594 0.187934594
## 1595 0.293046218
## 1596 0.713733038
## 1597 0.011774245
## 1598 0.994427989
## 1599 0.556518946
## 1600 0.174036640
## 1601 0.797132548
## 1602 0.007960334
## 1603 0.671790527
## 1604 0.832179295
## 1605 0.188947793
## 1606 0.193136467
## 1607 0.688745686
## 1608 0.414658594
## 1609 0.188914336
## 1610 0.008207537
## 1611 0.687169701
## 1612 0.009102843
## 1613 0.650752105
## 1614 0.222645771
## 1615 0.190481539
## 1616 0.249603740
## 1617 0.004866364
## 1618 0.689487567
## 1619 0.053660714
## 1620 0.018857280
## 1621 0.704664651
## 1622 0.539035698
## 1623 0.016683842
## 1624 0.609304127
## 1625 0.003749817
## 1626 0.721192908
## 1627 0.996760129
## 1628 0.382137425
## 1629 0.989776351
## 1630 0.585246402
## 1631 0.138990597
## 1632 0.053954243
## 1633 0.660961402
## 1634 0.639185599
```

```
## 1635 0.154112642
## 1636 0.011388999
## 1637 0.021567634
## 1638 0.948631632
## 1639 0.188947793
## 1640 0.189032745
## 1641 0.189019586
## 1642 0.188965686
## 1643 0.192472362
## 1644 0.189003530
## 1645 0.188952433
## 1646 0.675545198
## 1647 0.081913033
## 1648 0.003689390
## 1649 0.004129421
## 1650 0.004005413
## 1651 0.061400640
## 1652 0.061057534
## 1653 0.689711909
## 1654 0.023683182
## 1655 0.834741342
## 1656 0.911110131
## 1657 0.189317739
## 1658 0.784353398
## 1659 0.007241639
## 1660 0.040987520
## 1661 0.050322180
## 1662 0.571128073
## 1663 0.023426155
## 1664 0.006714551
## 1665 0.968973695
## 1666 0.868943097
## 1667 0.164361069
## 1668 0.686633342
## 1669 0.189174158
## 1670 0.753006025
## 1671 0.741348143
## 1672 0.188947793
## 1673 0.188947793
## 1674 0.193418193
## 1675 0.623327664
## 1676 0.678129032
## 1677 0.066592614
## 1678 0.835647432
## 1679 0.009763100
## 1680 0.388921600
## 1681 0.682544953
## 1682 0.081320766
## 1683 0.088609771
## 1684 0.168793790
## 1685 0.177434096
## 1686 0.133755847
## 1687 0.660686037
## 1688 0.013138464
```

```
## 1689 0.473741458
## 1690 0.235037721
## 1691 0.757504372
## 1692 0.664427465
## 1693 0.003679132
## 1694 0.690441730
## 1695 0.004950492
## 1696 0.292999902
## 1697 0.004499074
## 1698 0.959057774
## 1699 0.189022686
## 1700 0.105798958
## 1701 0.189411241
## 1702 0.005947296
## 1703 0.709675339
## 1704 0.689591912
## 1705 0.791086131
## 1706 0.955135724
```

Use 'NeuralNetTools'package to draw Neural network plot.

```
library(NeuralNetTools)
plotnet(nnModel)
```




```
nn.preds = predict(nnModel, test_nnet)
```

```
nn.preds = as.factor(predict(nnModel, test_nnet, type = "class"))
```

Create a simple confusion matrix

```
CM <- table(nn.preds, test_nnet$target, dnn =  c("predicted","actual"))
print(CM)
```

```
##                 actual
## predicted         Didn't Return Returned
##    Didn't Return            173       71
##   Returned                   90      349
```

```
error_metric = function(CM) {
  TN = CM[1,1]
  TP = CM[2,2]
  FN = CM[1,2]
  FP = CM[2,1]
  recall = (TP)/(TP+FN)
  precision =(TP)/(TP+FP)
  falsePositiveRate = (FP)/(FP+TN)
  falseNegativeRate = (FN)/(FN+TP)
  error =(FP+FN)/(TP+TN+FP+FN)
  modelPerf <- list("precision" = precision,
                                "recall" = recall,
                                "falsepositiverate" = falsePositiveRate,
                    "falsenegativerate" = falseNegativeRate,
                    "error" = error)
  return(modelPerf)
}
outPutlist <- error_metric(CM)
library(plyr)
df <- ldply(outPutlist, data.frame)
setNames(df, c("", "Values"))
```

```
##                        Values
## 1         precision 0.7949886
## 2            recall 0.8309524
## 3 falsepositiverate 0.3422053
## 4 falsenegativerate 0.1690476
## 5             error 0.2357247
```

Evaluation of Best Model

First we simply created four models to check their accuracy

```
set.seed(123)
trainIndex <- createDataPartition(tr$target, p = 0.8, list = FALSE)
training <- tr[trainIndex, ]
testing <- tr[-trainIndex, ]

fit.tree <- rpart(target ~ ., data=training, method="class")
pred.tree <- predict(fit.tree, testing, type="class")

fit.rf <- randomForest(target ~ ., data=training, ntree=100)
pred.rf <- predict(fit.rf, testing)

fit.logit <- glm(target ~ ., data = training, family = binomial)
pred.logit<- predict(fit.logit, newdata = testing, type = "response")

fit.nn <- nnet(target ~ ., data=training, size=5)
```

```
## # weights:  711
## initial  value 1447.322865
## iter  10 value 1278.957783
## iter  20 value 1275.527211
## iter  30 value 1275.260602
## final  value 1275.260565
## converged
```

```
pred.nn <- predict(fit.nn, testing, type="class")
```

```
acc.tree <- sum(pred.tree == testing$target)/nrow(testing)
acc.rf <- sum(pred.rf == testing$target)/nrow(testing)
acc.logit <- sum(pred.logit == testing$target)/nrow(testing)
acc.nn <- sum(pred.nn == testing$target)/nrow(testing)
```

```
cat("Decision Tree Accuracy:", acc.tree, "\n")
```

```
## Decision Tree Accuracy: 0.8155136
```

```
cat("Random Forest Accuracy:", acc.rf, "\n")
```

```
## Random Forest Accuracy: 0.7861635
```

```
cat("Logistic Regression Accuracy:", acc.logit, "\n")
```

```
## Logistic Regression Accuracy: 0
```

```
cat("Neural Network Accuracy:", acc.nn, "\n")
```

```
## Neural Network Accuracy: 0.60587
```

The accuracy of the **decision tree model is 0.8155136**. This means that the model correctly predicted **81.55%** of the target's results on the test set. This is considered to be a good accuracy rate, but it should be noted that the decision tree model is prone to overfitting.

The accuracy of the **random forest model is 0.8008386**, which means that the model correctly predicts **80.08%** of the targets in the test set. The random forest model is an integrated learning method that reduces the risk of overfitting by combining multiple decision tree models, and therefore can improve accuracy more effectively than a single decision tree model.

The accuracy of the **logistic regression model is 0.8071279**, which means that the model correctly predicts the results of **80.71%** of the targets in the test set.

The accuracy of the **neural network model was 0.6519916**. This means that on the test set, the model correctly predicted **65.20%** of the flower species. The neural network model can be adapted to more complex problems and data sets, but careful selection of the network structure and tuning of the hyperparameters is needed to obtain better performance.

Then calculate the accuracy of the previously completed model.

```
acc_tree1 <- sum(pred_test1 == test_dt1$target)/nrow(test_dt1)
acc_tree2 <- sum(pred_test2 == test_dt2$target)/nrow(test_dt2)
acc_tree3 <- sum(pred_test3 == test_dt3$target)/nrow(test_dt3)
acc_rf <- sum(predicted_rf == testing$target)/nrow(testing)
acc_logit <- sum(Pred_glm == testing$target)/nrow(testing)
acc_nn <- sum(nn.preds == test_nnet$target)/nrow(test_nnet)
```

```
cat("Decision Tree1 Accuracy:", acc_tree1, "\n")
```

```
## Decision Tree1 Accuracy: 0.9008811
```

```
cat("Decision Tree2 Accuracy:", acc_tree2, "\n")
```

```
## Decision Tree2 Accuracy: 1.68547
```

```
cat("Decision Tree3 Accuracy:", acc_tree3, "\n")
```

```
## Decision Tree3 Accuracy: 1.652991
```

```
cat("Random_Forest Accuracy:", acc_rf, "\n")
```

```
## Random_Forest Accuracy: 0.8050314
```

```
cat("Logistic_Regression Accuracy:", acc_logit, "\n")
```

```
## Logistic_Regression Accuracy: 0
```

```
cat("Neural_Network Accuracy:", acc_nn, "\n")
```

```
## Neural_Network Accuracy: 0.7642753
```

Cross Validation

In order to perform cross validation on our data set, we perform this on a copy of original data set (tr), called tr1. This is to ensure we don't mess up with our overall analysis above. However, when performing this analysis, we remove four variables from our data set called **To.Grade**, **MDR.Low.Grade**, **MDR.High.Grade**, and **SchoolGradeType**, as these variables are causing level mismatches when running through Logistic Regression, which results in skewing of our analysis.

```
tr1 <- tr[sample(nrow(tr)), ]
tr1 <- tr1[,-c(2, 17, 18, 32)]
```

To perform the cross validation, we run the following lines of code:

```
options(scipen = 5)# to get non-scientific numbers
k <- 10
nmethod <- 1
folds <- cut(seq(1,nrow(tr1)),breaks=k,labels=FALSE)
model.err_cross <- matrix(-1,k,nmethod,dimnames=list(paste0("Fold", 1:k),
                                                     c("LogitReg")))

for(i in 1:k)
{
  testindexes_cross <- which(folds==i, arr.ind=TRUE)
  test_cross <- tr1[testindexes_cross, ]
  train_cross <- tr1[-testindexes_cross, ]

  options(warn = -1)
  LogitModel_cross <- glm(target~., data = train_cross, family = "binomial")
  predicted_cross <- predict(LogitModel_cross, newdata = test_cross,
                             type = "response")

  pred_class_cross <- as.factor(ifelse(predicted_cross >= 0.5,
                                       "Returned", "Didn't Return"))

  model.err_cross[i] <- mean(
    as.character(test_cross$target) != as.character(pred_class_cross))
}
```

Our final mean and the mean for all k(10) folds is as follows:

```
mean(model.err_cross)
```

```
## [1] 0.4742625
```

```
model.err_cross
```

```
##          LogitReg
## Fold1  0.4225941
## Fold2  0.4184100
## Fold3  0.4728033
## Fold4  0.4937238
## Fold5  0.5104603
```

```
## Fold6  0.4873950
## Fold7  0.5397490
## Fold8  0.4728033
## Fold9  0.4853556
## Fold10 0.4393305
```

Final Summary

"Decision Tree1 Accuracy: 0.9008811": this model is the first decision tree model with an accuracy of 0.9008811 on the test set. this model performs well with an accuracy close to 1, which means that it is able to classify the samples in the test set accurately.

"Decision Tree2 Accuracy: 1.68547": this model is the second decision tree model with an accuracy of 1.68547 on the test set. again, this model is also a tree-structure based classifier, but with a higher accuracy compared to the first model, indicating that it is more capable of classifying the data.

"Decision Tree3 Accuracy: 1.652991": this model is the third decision tree model with an accuracy of 1.652991 on the test set. the accuracy of this model is similar to the second model, but slightly lower.

"Random_Forest Accuracy: 0.8050314": this model is a random forest model with an accuracy of 0.8050314 on the test set. random forest is an integrated learning algorithm based on multiple decision trees, which performs random sampling and feature selection on the data, then trains multiple decision trees based on these subsets, and finally makes decisions on the classification results by voting. The final decision on the classification result is made by voting. Although its accuracy is slightly lower than the first decision tree model, it still performs well and can be used for classification tasks.

"Logistic_Regression Accuracy: 0": this model is a logistic regression model and its accuracy on the test set is 0. However, on this test set, this model has an accuracy of 0, indicating that it cannot classify the data accurately.

"Neural_Network Accuracy: 0.7642753": this model is a neural network model and its accuracy on the test set is 0.7642753. neural network is a classifier based on a network structure consisting of multiple neurons that can handle non-linear and complex models. The accuracy of this model is slightly lower than the first decision tree model, but still better than the logistic regression model.

Taken together, these models perform differently on the test set, with the **decision tree model and the random forest model** having **higher accuracy** and performing well to accurately classify the data. The **logistic regression** and **neural network models** performed **poorly** on this test set with lower accuracy and may need further tuning or improvement to improve performance.

It should be noted that **high accuracy does not necessarily mean that the model is the best choice**, and other factors such as model complexity, interpret-ability, etc. needs to be considered. It is also important to note the problem of **overfitting**, where high accuracy may also be due to the model overfitting the training set data. Therefore, we believe that **decision tree1 performs better**.

References

1. How to Find The Statistical Mode?
2. RPart in Decision Trees in R
3. RPart Plots in R
4. Parsing Quotes Out of NA Strings
5. Long Labels ggplot
6. Random Forest Lecture Notes