Report: Predict Car Features

## 1. Best Submission with Score 1.18934

### Multinomial logit

library(mlogit)  
train\_raw <- read.csv("train.csv")  
test\_raw <- read.csv("test.csv")  
S <- mlogit.data(train\_raw,shape="wide",choice="Choice",varying=c(4:83), sep="", alt.levels=c("Ch1","Ch2","Ch3","Ch4"), id.var="Case")  
test\_mlogit <- mlogit.data(test\_raw,shape="wide",choice="Choice",varying=c(4:83), sep="", alt.levels=c("Ch1","Ch2","Ch3","Ch4"), id.var="Case")  
test\_mlogit[is.na(test\_mlogit)] <- 100  
  
M <- mlogit(Choice~CC+GN+NS+BU+FA+LD+BZ+FC+FP+RP+PP+KA+SC+TS+NV+MA+LB+AF+HU+Price-1|segment+miles+ppark+age, data=S)  
# M <- mlogit(Choice~CC+GN+NS+BU+FA+LD+BZ+FC+FP+RP+PP+KA+SC+TS+NV+MA+LB+AF+HU+Price-1, data=S)  
P <- predict(M,newdata=test\_mlogit)  
# subset(test\_mlogit,select=c(CC,GN,NS,BU,FA,LD,BZ,FC,FP,RP,PP,KA,SC,TS,NV,MA,LB,AF,HU,Price))  
#write.csv(P,file="mlogit-fendy.csv",row.names=FALSE,sep=",")

## 2. Next Best Submission with Score 1.23357

### NaiveBayes

library(e1071)  
train <- read.csv("train.csv", stringsAsFactors = FALSE)  
train8 <- train  
train8 <- factorise(train8)  
train8 <- convertincometointeger(convertincome\_train(train8)) # converts '$290,000 to $299,999' to '$300,000 and over', then take the middle value.  
train8$Choice <- factor(train8$Choice)  
train8$income <- factor(train8$income)  
train8$Price1 <- factor(train8$Price1)  
train8$Price2 <- factor(train8$Price2)  
train8$Price3 <- factor(train8$Price3)  
train8$Price4 <- factor(train8$Price4)  
set.seed(100)  
library(caTools)  
spl <- sample.split(train8$Choice, SplitRatio = 0.7)  
trainoftrain8 <- subset(train8, spl == TRUE)  
trainoftest8 <- subset(train8, spl == FALSE)  
trainoftrainnew8 <- subset(trainoftrain8, select = - c(1,2,3,64:83,95,96,97,98))  
trainoftestnew8 <- subset(trainoftest8, select = - c(1,2,3,64:83,95,96,97,98))  
  
nb1 <- naiveBayes(Choice ~ ., data = trainoftrainnew8)  
prednb1 <- predict(nb1, newdata = trainoftestnew8, type = "raw")  
head(prednb1, 1)  
N <- nrow(prednb1)  
sum(prednb1[1,])  
logloss<- -(sum(log(prednb1)\*trainoftest8[,95:98]))/N  
logloss  
#write.csv(prednb1, "naivebayes3.csv")

## 3. Other Attempted Models

### 3.1 Neural Networks

# Kaggle score: 1.24106  
library(caret)  
library(neuralnet)  
train <- read.csv("train.csv", stringsAsFactors = FALSE)  
test <- read.csv("test.csv", stringsAsFactors = FALSE)  
trainNN <- train[,c(4:83,99)]  
testNN <- test[,4:83]  
set.seed(100)  
modelNN <- neuralnet(Choice~CC1+GN1+NS1+BU1+LD1+BZ1+FC1+PP1+KA1+SC1+TS1+NV1+MA1+LB1+AF1+HU1+Price1+  
 CC2+GN2+NS2+BU2+LD2+BZ2+FC2+PP2+KA2+SC2+TS2+NV2+MA2+LB2+AF2+HU2+Price2+  
 CC3+GN3+NS3+BU3+LD3+BZ3+FC3+PP3+KA3+SC3+TS3+NV3+MA3+LB3+AF3+HU3+Price3+  
 CC4+GN4+NS4+BU4+LD4+BZ4+FC4+PP4+KA4+SC4+TS4+NV4+MA4+LB4+AF4+HU4+Price4,data=trainNN, hidden=9)  
probs1 <- predict(modelNN, newdata=testNN, type='prob')  
#write.table(probs1,file="nn.csv",sep=",",row.names=FALSE)

### 3.2 C4.5

library(RWeka)  
library(FSelector)  
train <- read.csv("train.csv", stringsAsFactors = FALSE)  
  
train2 <- train  
train2 <- factorise(train2)  
train2 <- convertincometointeger(convertincome\_train(train2)) # converts '$290,000 to $299,999' to '$300,000 and over', then take the middle value.  
train2$Choice <- factor(train2$Choice)  
train2$income <- factor(train2$income)  
train2$Price1 <- factor(train2$Price1)  
train2$Price2 <- factor(train2$Price2)  
train2$Price3 <- factor(train2$Price3)  
train2$Price4 <- factor(train2$Price4)  
  
set.seed(100)  
spl <- sample.split(train2$Choice, SplitRatio = 0.7)  
trainoftrain <- subset(train2, spl == TRUE)  
trainoftest <- subset(train2, spl == FALSE)  
trainoftrainnew <- subset(trainoftrain, select = - c(1,2,3,95,96,97,98))  
trainoftestnew <- subset(trainoftest, select = - c(1,2,3,95,96,97,98))  
  
information.gain(Choice ~ ., data = trainoftrainnew)  
modelC45 <- J48(as.factor(Choice) ~ ., data = trainoftrainnew, control = Weka\_control(A = TRUE, R = FALSE))  
predC45 <- predict(modelC45, newdata = trainoftestnew, type = "probability")  
head(predC45, 1)  
N <- nrow(predC45)  
sum(predC45[1,])  
logloss<- -(sum(log(predC45)\*trainoftest[,95:98]))/N  
logloss

### 3.3 Random FOrest

#kaggle score: 1.23523  
library(randomForest)  
train\_all <- train  
train\_all <- factorise(train\_all)  
train\_all <- convertincometointeger(convertincome\_train(train\_all)) # converts '$290,000 to $299,999' to '$300,000 and over', then take the middle value.  
train\_all$Choice <- factor(train\_all$Choice)  
train\_all$income <- factor(train\_all$income)  
train\_all$Price1 <- factor(train\_all$Price1)  
train\_all$Price2 <- factor(train\_all$Price2)  
train\_all$Price3 <- factor(train\_all$Price3)  
train\_all$Price4 <- factor(train\_all$Price4)  
  
train\_allnew <- subset(train\_all, select = - c(1,2,3,95,96,97,98))  
  
rf <- randomForest(Choice ~ ., data = train\_allnew, importance = TRUE, ntree = 2000)  
predrf <- predict(rf, newdata = test\_allnew, type = "prob")  
head(predrf)  
sum(predrf[1,])  
N <- nrow(predrf)  
logloss<- -(sum(log(predrf)\*trainoftest[,95:98]))/N  
logloss  
  
predrfMatrix <- matrix(predC50[,2], nrow = N, ncol = 4, byrow = TRUE)  
head(predrfMatrix)  
for (i in 1:N){  
 d <- sum(predrfMatrix[i,])  
 for (j in 1:4){  
 predrfMatrix[i,j] <- predrfMatrix[i,j]/d  
 }  
}  
sum(predfMatrix[1,])  
predrfMatrix\_log <- log(predrfMatrix)  
logloss<- -(sum(predrfMatrix\_log\*train[,95:98]))/N  
logloss

## 4. Functions we created to preprocess data

# convert all income to integer  
convertincometointeger <- function(object){   
 object$income <- ifelse(object$income == "Under $29,999", 15, ifelse(object$income == "$30,000 to $39,999", 35, ifelse(object$income =="$40,000 to $49,999", 45, ifelse(object$income =="$50,000 to $59,999", 55, ifelse(object$income =="$60,000 to $69,999", 65, ifelse(object$income =="$70,000 to $79,999", 75, ifelse(object$income =="$80,000 to $89,999", 85, ifelse(object$income =="$90,000 to $99,999", 95, ifelse(object$income =="$100,000 to $109,999", 105, ifelse(object$income == "$110,000 to $119,999", 115, ifelse(object$income == "$120,000 to $129,999", 125, ifelse(object$income == "$130,000 to $139,999", 135, ifelse(object$income == "$140,000 to $149,999", 145, ifelse(object$income == "$150,000 to $159,999", 155, ifelse(object$income == "$160,000 to $169,999", 165, ifelse(object$income == "$170,000 to $179,999", 175, ifelse(object$income == "$190,000 to $199,999", 195, ifelse(object$income == "$200,000 to $209,999", 205, ifelse(object$income == "$220,000 to $229,999", 225, ifelse(object$income == "$250,000 to $259,999", 255, ifelse(object$income == "$300,000 & Over", 320, NA)))))))))))))))))))))  
 return (object)  
}  
  
# convert all price to integer  
convertpricetointeger <- function(object){   
 object <- ifelse(object == 1, 500, ifelse(object == 2, 1000, ifelse(object ==3, 1500, ifelse(object == 4, 2000, ifelse(object == 5, 2500, ifelse(object == 6, 3000, ifelse(object == 7, 4000, ifelse(object == 8, 5000, ifelse(object == 9, 7500, ifelse(object == 10, 10000, ifelse(object == 11, 12000, 1)))))))))))  
 return (object)  
}  
  
# manage inconsistencies in income (only for training)  
convertincome\_train <- function(object){  
 N <- nrow(object)  
 for (i in 1:N){  
 if (object[i,93] == "$290,000 to $299,999"){  
 object[i,93] <- as.character("$300,000 & Over")  
 print ("converted!")  
 }  
 }  
 return (object)  
}  
  
# manage inconsistencies in income (only for test)  
convertincome\_test <- function(object){  
 N <- nrow(object)  
 for (i in 1:N){  
 if (object[i,93] == "$180,000 to $189,999"){  
 object[i,93] <- as.character("$170,000 to $179,999")  
 print ("converted!")  
 } else {  
 if (object[i,93] == "$230,000 to $239,999"){  
 object[i,93] <- "$220,000 to $229,999"  
 print ("converted!")  
 } else {  
 if (object[i,93] == "$270,000 to $279,999"){  
 object[i,93] <- ("$250,000 to $259,999")  
 print ("converted")  
 }  
 }  
 }  
 }  
 return (object)  
}  
  
# factorise the variables (segment, year .. ppark)  
factorise <- function(object){  
 object$segment <- factor(object$segment)  
 object$year <- factor(object$year)  
 object$miles <- factor(object$miles)  
 object$night <- factor(object$night)  
 object$gender <- factor(object$gender)  
 object$age <- factor(object$age)  
 object$educ <- factor(object$educ)  
 object$region <- factor(object$region)  
 object$Urb <- factor(object$Urb)  
 object$ppark <- factor(object$ppark)  
 return (object)  
}